COPYRIGHT NOTICE

Thank you for your integrity by not copying any portion of this* ExcelCEO Access 2016 and SQL *complete self-study course, including the course manual and/or practice files (in whole or in part) for others to use.

Jim Cline

Tips. Tricks. Training. Mastery.

The Master Guide to Financial Reporting and Analysis

ExcelCEO believes the information contained in this manual is accurate. We have taken much care in its preparation. However, we do not accept any responsibility, financial or otherwise, for any consequences coming from the use of this material, including loss of profit and any other indirect, special, or consequential damages. No warranties extend beyond this course specification. If you do not agree to these terms, do not use this course. Using this course for Access training and reference is considered agreement to these terms.

You should exercise care to assure that the use of the concepts taught in this course is in full compliance with federal, state, and local laws, rules, and regulations, and meets your certification requirements.

Rev. 1.02
All right reserved
© September 2019, ClineSys/ExcelCEO.com
Printed in Houston, TX, United States of America

Access® 2016 and SQL

Complete Self-study Course

ExcelCEO

Chief Excel Officer

Introduction

Most accountants and financial folks have a relatively decent knowledge of Excel before they take this course, particularly if they completed the ExcelCEO Excel course. Excel seems to be THE program for financial analysis, partly due to how easy it is to use. Almost anyone can open Excel and start typing numbers onto a spreadsheet and immediately see results. Not so with Access. In its simplest form, Access is a *relational database management system*. As such, you use it to create relationships between tables, query data from those relationships, and use or display the results in forms or reports. If you have no experience with relational databases, don't worry. I go through everything step-by-step, just like we did in the Excel course. It is important, however, for you to read every word and complete every exercise in this course. There is nothing herein that is unimportant. To teach you what you need to know about Access, I need to have your complete attention in each chapter.

It is my opinion that Access is the most under-utilized program in the market today. At the same time, it is the most over-utilized program on the market, particularly for the accounting and finance community. Few accountants and financial people know how to use Access, but once they figure it out, they seem to use it for EVERYTHING! An ex-employee of a certain unnamed, defunct energy trading company based in Houston (whose name rhymes with TENRON) told me he had created an Access database that paid out millions of dollars in bonuses to some of the company's employees. It seemed strange to me that a large company like that would depend on a "simple desktop program" like Access to calculate those kinds of important bonuses, but it happens. I sincerely believe that Access is the best Windows-based relational database program on the market, but it can be over-used.

To learn Access, you must first explore its basics: Tables, Queries, Forms, and Reports. You will then delve into more "advanced basics", like complex queries and reports. There are other tools in Access, like Macros and Modules, but I will touch only lightly on those subjects. I consider those to be the bells and whistles of Access. By the time you finish this course, you will have a very good grasp on the nuts and bolts of Access and you'll know if you need to go into the bells and whistles for your particular needs. By that time, you'll be able to figure it all out for yourself.

I will assume that you have already completed the Excel course or that you have a very good grasp on Excel. This Access course is a continuation of the Excel course in that I will use the same company, Nitey-Nite Mattresses, in all of my examples. If you haven't taken the Excel course, I strongly encourage you to do so, as many database concepts that you will need to know for Access are taught in that course. However, if you insist on taking this course without the Excel course, let me fill you in about Nitey-Nite Mattresses.

In creating this course, I wanted to create a company whose accounting system is complex enough to simulate real world activity, but simple enough to be used in the examples and projects. I didn't want you to have to take another course just to learn the accounting system of this company, but I had to make it complex enough to teach the necessary concepts. As such, I created a fictitious company called Nitey-Nite Mattresses. Nitey-Nite's business is to operate stores in retail centers across the United States to sell its line of mattresses, pillows, and other related merchandise. It owns and operates 29 stores along the East Coast. Nitey-Nite purchases mattresses and pillows from four vendors and ships them to the 29 stores.

> ***Please note:*** *The financial data contained herein is purely fictitious and does not resemble the activity in any way of any similar retail store today. The accounting methods used in these examples and projects do not necessarily conform with GAAP (Generally Accepted Accounting Principles) requirements for the industry, although the financial statements and accounting practices herein reflect standard double-entry accounting methodology.*

This course is written specifically for Microsoft Office Access 2016 using the Windows 10 operating system. While the skills and tips taught generally translate to other versions of Excel, this training program should be taken with a copy of Microsoft Access 2016, which is not included in ExcelCEO training.

In Access 2007, the Microsoft engineers introduced the concept of the Office Ribbon, which is located at the top of the program window. All of the toolbars and menus prior to Access 2007 have been reorganized into the Office Ribbon tabs and galleries. If you are an experienced user for any version prior to 2007, you will initially find yourself very frustrated. Access 2016 expands on the capabilities of Access 2007 — 2013, although there haven't been many major changes since Access 2010 for Access. If you are new to Access, you will find it very easy to unlock the powerful data analysis capabilities as compared to previous versions.

I sincerely hope that you will enjoy this course. If you find any errors that should be corrected, or if you would like to send me any feedback for improving this course, successes you have achieved as a result of your training, etc., please email Customer.Service@ExcelCEO.com. Thank you.

Jim Cline
Founder, Developer, and Author

Derek Mecham
Editor, Designer, and Technical Support

Getting Started

Welcome to the ExcelCEO Access 2016 and SQL training course! Whether or not your are taking this course for CPE credit, we are confident you will gain significant and useful skills as you follow the detailed, step-by-step instructions provided in this course manual, and work through these 15 chapters of projects designed to teach you beginning to advanced Access. This course will challenge you, but the gains will be worth the effort!

ExcelCEO courses are classified as self-study and are not considered to be online courses. This means you will need access to a copy of the Microsoft Access® program correlated to this course — in this case, Microsoft® Office Access 2016 — to follow along with the examples outlined in this course manual. While ExcelCEO does provide some complementary tutorial examples of some ExcelCEO exercises — located at www.ExcelCEO.com — this course is not provided as audio/video-based instruction. The tutorials are designed to show you skills you should be able to gain for yourself as you progress toward ExcelCEO Excel Master status by allowing you to see the instructions, and work hands-on with the projects given.

When you registered for this course, you received a case-sensitive **Password**. You will input your registered Email Address as your User ID with your Password to login at www.ExcelCEO.com to download the database files, take Review Questions and chapter exams, and gain general access to your ExcelCEO profile. Keep your User ID and Password in a secure place, so you can refer back to them as needed. A [**Forgot Your Password?**] link is available at ExcelCEO.com in case you forget your password.

To work the many examples illustrated in this course, you need to download the Practice Files from the **Main Menu** in your student profile. To download the **Practice Files** for your selected course, log in to www.ExcelCEO.com with your User ID and Password, and click on the **Downloads** link on the Main Menu. Step-by-step instructions are provided to show you what to expect as the files download to the computer you are using. If you choose to save the Practice Files somewhere besides the default location, remember where you saved them to ensure convenient access later. If you do forget the location, you can click on the Windows button (Windows 7), the Start Menu icon (Windows 8/8.1), or the Cortana button (Windows 10), then type Access 2016 in the search box, and select the File folder from that menu.

Review Questions

As you work through the course manual, you will periodically be instructed to sign on to www.ExcelCEO.com and complete the Review Questions. Review Questions are formatted in the same way as the actual examination questions, but are provided for review purposes as well as for extended learning opportunities. Review Questions are not graded, though the program will indicate whether or not the chosen answer is correct. Have the Access program open for hands-on learning when completing Review Questions. If you choose an incorrect answer, a message will appear indicating why the chosen answer is incorrect, and the program will allow you to choose another answer. You must choose the correct answer before continuing to the next question. Login to your student profile at www.ExcelCEO.com to complete the Review Questions when you see a note paragraph like the following:

> ***Review Questions****: It is now time to complete the hands-on Review Questions. Log on to www.ExcelCEO.com with your Email address and Password, click on the* **Access 2016 and SQL Review Questions, Chapter 1, Section 1 of 2** *option in your Main Menu, and complete the Review Questions.*

The program will guide you to the correct Review Questions section or Chapter Exam based on the sections you have already completed. Each chapter contains between one and two sections of Review Questions. Review Questions checkpoints for each chapter must be completed in order to progress to the chapter exams.

Chapter Examinations

There are 15 chapters in the Access 2016 course. After completing the exercises in each chapter, you will be instructed to go to www.ExcelCEO.com, log in, and take an exam. After you log in with your email address and password, and after you have completed all of the Review Questions, you will click on the Chapter xx Exam link which will navigate you to the appropriate test. All tests are administered sequentially after you successfully pass the previous chapter's exam and Review Questions. The Chapter Exams use questions from a database of hundreds of possible questions, so it is highly unlikely that any two people will get the same exam. Some of the exam questions are based on the completed examples in the practice files, so I would encourage you to complete all of the exercises in the course. For example, if you complete an exercise that calculates net income by store, one exam question may ask you, "*What is the result in qry10Net_Income if you change the criteria to include January and February?*" Each question is in a multiple-choice format, and you will have four choices from which to choose.

The only recorded score for each exam is the passing grade. Your ExcelCEO student profile contains a Certificates section where exams you complete are tracked by date and score. You must score a 70% or better on each exam to pass. After obtaining a passing grade, you will be able to complete a short survey of the chapter, then print out a Certificate of Completion as evidence that you've read the material, worked the hands-on project examples, completed the Review Questions, and passed the chapter exam.

Continuing Professional Education (CPE) Credits

To claim CPE credits under NASBA guidelines, the CPA must successfully complete any chapters in the course within one year from the date of purchase to claim credit for them. The CPE credits for each chapter in this manual are listed on the following page. Upon completion of a chapter, sign on to www.ExcelCEO.com with your email address and password and click on the Certificates link to print a copy of the certificate of completion. As of December 31, 2010, the student can take up to two retests on final exams that were failed. After the second retest (or the third time to take the final exam), the student cannot receive CPE credits for the course (in this case, the applicable individual chapter).

National Association of State Boards of Accountancy (NASBA)

ClineSys is registered with the National Association of State Boards of Accountancy (NASBA) as a sponsor of continuing professional education on the National Registry of CPE Sponsors. State boards of accountancy have final authority on the acceptance of individual courses for CPE credit. Complaints regarding registered sponsors may be submitted to the National Registry of CPE Sponsors through its website: www.nasbaregistry.org

CPE Credit Schedule

Chapter	Title	CPE Credits
1	Access 2016 Basics	2.0
2	Beginning Queries	2.0
3	Intermediate Queries	2.0
4	Advanced Queries	3.0
5	Action Queries and Macros	3.0
6	Beginning Forms	3.0
7	Intermediate Forms	2.0
8	Beginning Reporting	3.0
9	Intermediate Reporting	3.0
10	DSN and External Data Sources	2.0
11	Advanced Reporting (Part I)	3.0
12	Advanced Reporting (Part II)	3.0
13	Introduction to SQL	3.0
14	Intermediate SQL	3.0
15	Combining Access, Excel, and SQL	3.0
	Total	**40.0**

***Note**: Completion of Access 2016 and SQL is required to earn an* **ExcelCEO Access Master** *certificate.*

Conventions Used In This Course

The basis behind this course is learning by example. As such, I have included hundreds of tasks, examples and projects. Steps to complete a task are numbered, and are shown in italicized and bold text. Objects or buttons you can see are in Bold font, and names or formulas you should type are bolded and italicized. Workbook tabs have vertical tab lines around them, and are bolded. Here are some examples:

1. *Click on the* **Quantity** *field in the* **Design View** *of* **qry01NetIncome**.
2. *Enter the formula to read* ***=SUM([Qty])****, and press* **[Enter]**.

To assist you, I have included hundreds of screen shots and pictures of the icons used in the examples, including simple assists like this **Save As** icon: 💾 Screen shots are from **Access 2016** using **Windows 10**.

Action keys (keys that do something other than type a character on the screen) on the keyboard are referred to in brackets, such as the Enter key **[Enter]** and the F2 key **[F2]** (possibly **Fn**+**[F2]**, depending on your keyboard setup). Sometimes it is required to hold down one or more keys on the keyboard to perform a certain action. For example, to make a cell bold, you press and hold the Control key [Ctrl] and then type the "***b***" key. I will refer to this action as **[Ctrl]**+***b***. Action keys can also be sequential, by typing one key at a time. For example, to execute the process to set a column width using action keys would be to press the [Alt] key, then the o, c, and w keys, each separated by a comma. I will refer to this action as [Alt], o, c, w. These work in Access 2016 just as they did in Access 2003 — 2013. If you are accustomed to using action keys in previous versions of Access, you'll feel at home with Access 2016.

Keyboard Shortcuts

Keyboard shortcuts are keyboard strokes that execute functionalities without having to choose the options from the Office ribbon with the mouse. Keyboard shortcuts generally include the use of the [Ctrl], [Alt], and/or [Shift] keys. For those of you who liked to use keyboard shortcuts in earlier versions of Access, you'll be happy to know that the same shortcuts exist in Access 2016.

Prerequisites

Prerequisites for taking the Access course include a basic knowledge of a Windows operating system, and knowing how to use the keyboard, mouse, and general knowledge of how to access the Internet. Although it is not necessary, I highly recommend taking the Excel course before the Access course, as many concepts in the Excel course prepare you to take the Access course. This course is written specifically for people with a financial background, so I will assume you know the basics of income statements and the transactions (the debits and credits) that make up the statements. With that said, let's get started.

CHAPTER ONE — ACCESS BASICS

In this chapter, you will:

- Identify the new features of Access 2016
- Determine Access basics (types of databases, navigation, and screen appearance)
- Choose ways to create a new database
- Identify how to open an existing database
- Recognize how to open, filter, and sort a table using icons within the Office Ribbon
- Determine the various parts of Table Design (data types and field properties)
- Identify the Primary Key in Table Design
- Recognize the appropriate way to add a new record to a table

CPE Credits possible for this chapter: 2

What's New in Access 2016?

If you are converting from Access 2003 or 2007 to Access 2016, the answer is a lot, probably too many new features to show you all of them in this course. The Microsoft engineers have made a lot of changes from Access 2003 to 2007, but if you are upgrading from Access 2010 or 2013 to 2016, you won't see many changes. The release of Office 2016 was geared more towards Excel and Word and Access was left alone. I developed this course to show accounting and financial professionals how to use Access for data analysis and reporting, and most of the new changes are more cosmetic in nature. Many of the new features are included in the examples you will work throughout this course, and I have listed most of them in the following bullet points. If any of them interests you, feel free to explore these new features on your own. More about these features can also be found at http://office.microsoft.com/en-us/access/default.aspx.

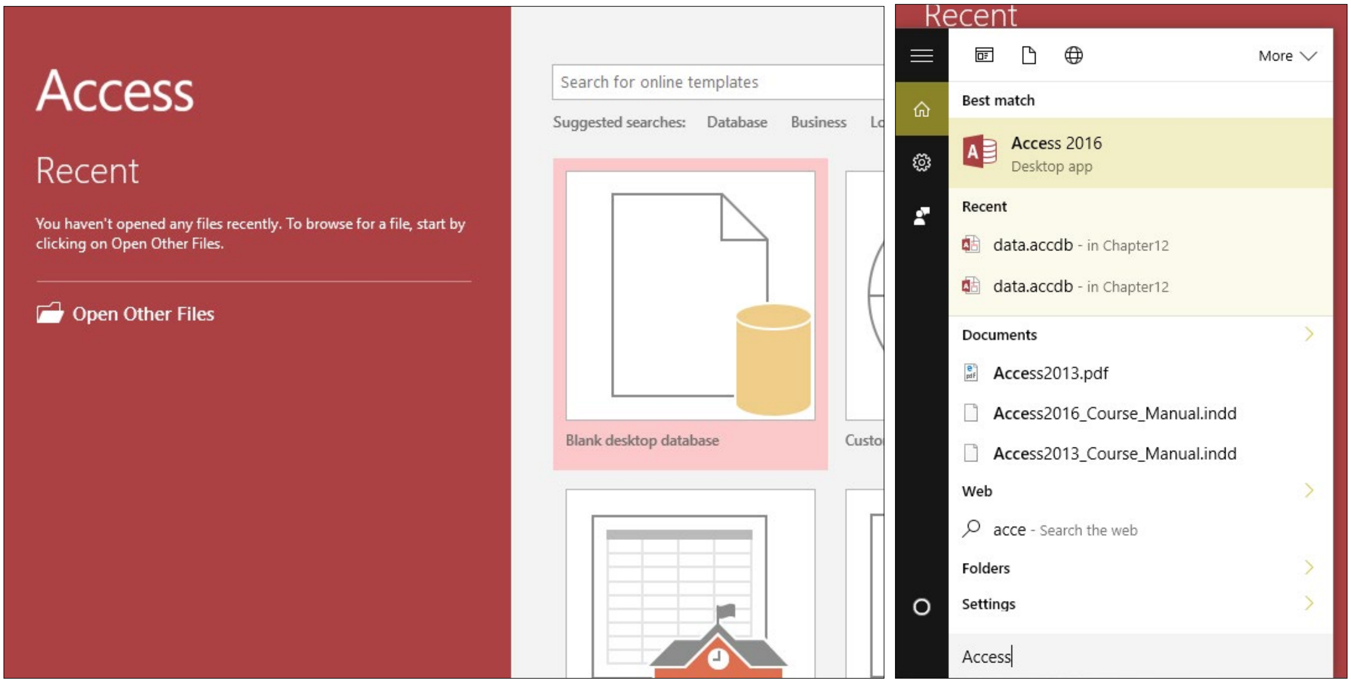
- Ribbon: Replaces toolbars used in earlier versions of Access. It is similar to the Ribbon used in Excel 2016.
- Quick Access Toolbar: Allows the user to place frequently used icons in one toolbar.
- Navigation Pane: Replaces the Database window from Access 2003. Allows you to hide, unhide, and/or organize objects such as Tables, Queries, Forms, and Reports.
- View Toolbar: Considered to be a super-set of the tabs or views in the database object. Users can quickly navigate between Datasheet View, Design View, PivotTable View, PivotChart View, Form View, Layout View, Report View, and other views.
- Tabbed Documents: Creates a tab for each opened object. This is one of my favorite enhancements.
- Template Library: An entire library of professionally designed templates.
- Layout View: Edit a report or form while in View mode.
- Calendar: The date/time data type now includes an optional calendar control which allows the user to simply click on a date instead of typing it in.
- Rich Text: Memo fields now support most formatting, including fonts, color and text formatting.
- Create Tab: Quickly create Access objects.
- Totals Function: Automatically calculate totals and aggregate values in a query.
- Field List Pane: Drag and drop related fields on to your active table.
- Attachment Data Type: Import or attach photos and other files to a database record.
- Embedded Macros: Take advantage of better security by embedding macros in a form or report.
- Enhanced Microsoft Help: Easily search developer and end-user help within Access.
- Sharing Information: Enhanced collaboration between Access and other Office applications, such as Excel, Outlook, and SharePoint, as well as creating XML, HTML, PDF, and dBase files.
- Enhanced Security: Adding password protection to a database automatically encrypts it when closed, and decrypts it when opened.

Access 2016 Basics

Access is a relational database. A database is simply a tool where you can store and manipulate data. The data is stored in tables. The term "relational" is used because in order to use the data in the tables, you need to create relationships between the tables. An Access file is called a database. An Access 2016 database is stored as an .accdb file. Access databases before Access 2007 were stored with the .mdb extension.

In this course, I will use Microsoft Access 2016, but most of the concepts taught can be applied to any version of Access. To follow along with the screenshots, I recommend you use Access 2016 software for this course as Access 2016 has a slightly different interface than any of its predecessors before Access 2010. Let's open Access.

1. *Open* **Access 2016** *(from the* **Windows button** *on your* **Taskbar**, *choose* **All Apps**, **Microsoft Office 2016**, **Access 2016**). *The* **Cortana** *search box is shown below to the right.*



You haven't opened any files recently. To browse for a file, start by clicking on Open Other Files.

Access
Recent
Open Other Files
Search for online templates
Suggested searches: Database Business Lo
Blank desktop database
Custo
Recent
More
Best match
Access 2016
Desktop app
Recent
data.accdb - in Chapter12
data.accdb - in Chapter12
Documents
Access2013.pdf
Access2016_Course_Manual.indd
Access2013_Course_Manual.indd
Web
acce - Search the web
Folders
Settings
Access

Figure 1.1

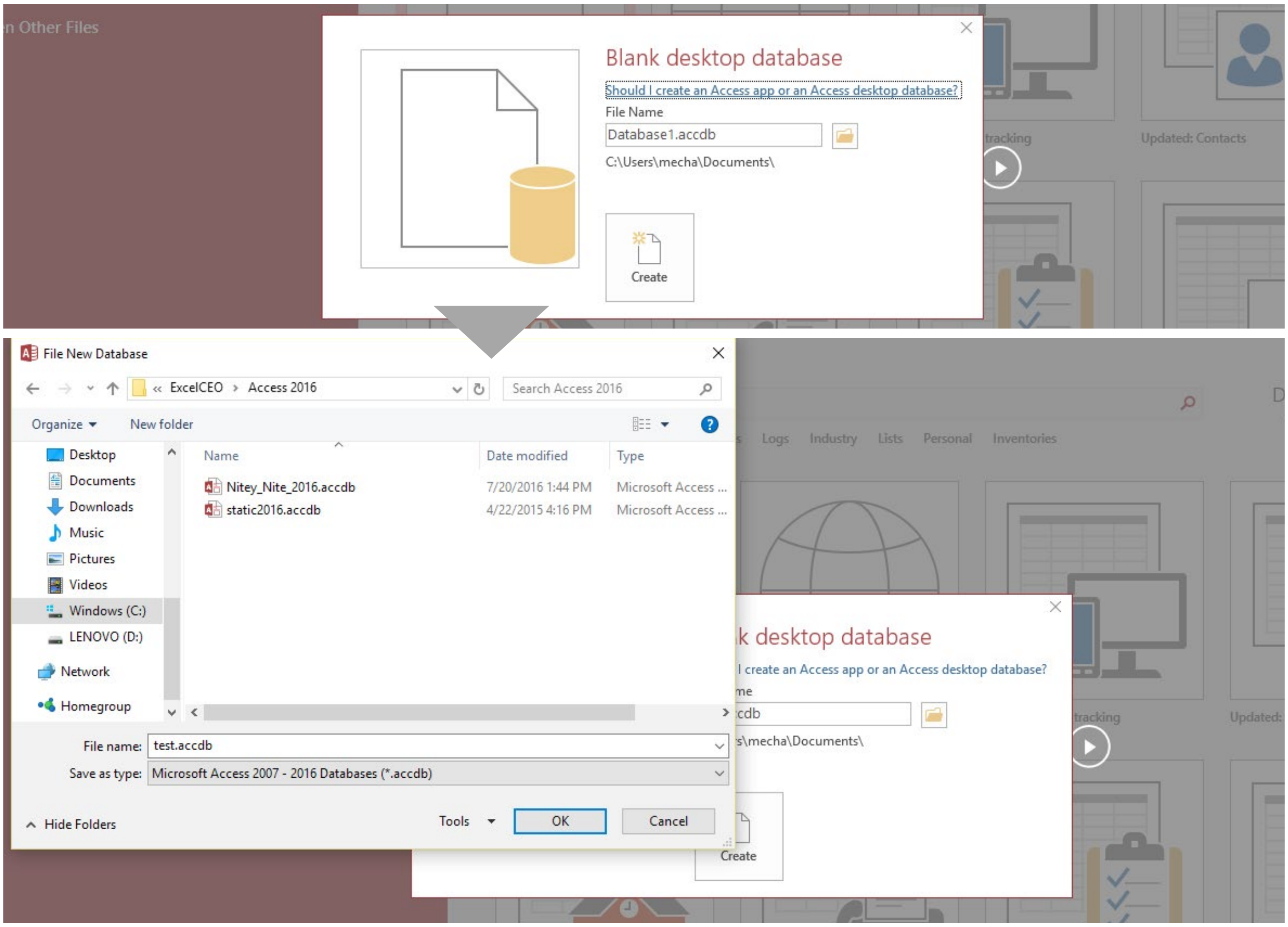
When you first open Access 2016 without opening a database, Access displays the ***Backstage View***, as displayed in Figure 1.1. You can display the Backstage View at any time by simply clicking the File tab along the top of the ribbon. It is in the Backstage view where you can create a new database, use database templates, view recently used databases, and change Access options, among other things. In this course, we touch only lightly on using templates. ***Templates*** are databases that use predefined tables, queries, forms, and reports that are used when you want to create a database that will serve a specific purpose, such as a database that contains all of your Contacts, Issues and Tasks, or Projects. These templates can be wonderful tools. They are professionally designed and look great. I strongly encourage you to use templates after you complete this course. Once you finish this course, you will be very good at the complex "behind the scenes" programming, and that instruction is not provided very well in the templates.

In this course, I will teach you how to make your database very functional. You can then use templates to make your database look "pretty". I have written this course specifically for accounting and financial professionals. For the most part in this course, you will create your own tables, queries, forms, and reports that are specific to accounting and finance projects. In the middle of the screen is the option to create a new database, which you will do later in this chapter. On the left side of the Backstage View is a list of the most recent databases you have opened. To open a recently used database, simply click on it.

Create a New Database

The first step in working with Access is to create a new ***database*** or open an existing one. Let's create a new database.

2. *Click on the* **Blank database** *icon (to the right of the* **Recent** *title) on the right side of the* **Backstage Templates** *view.*



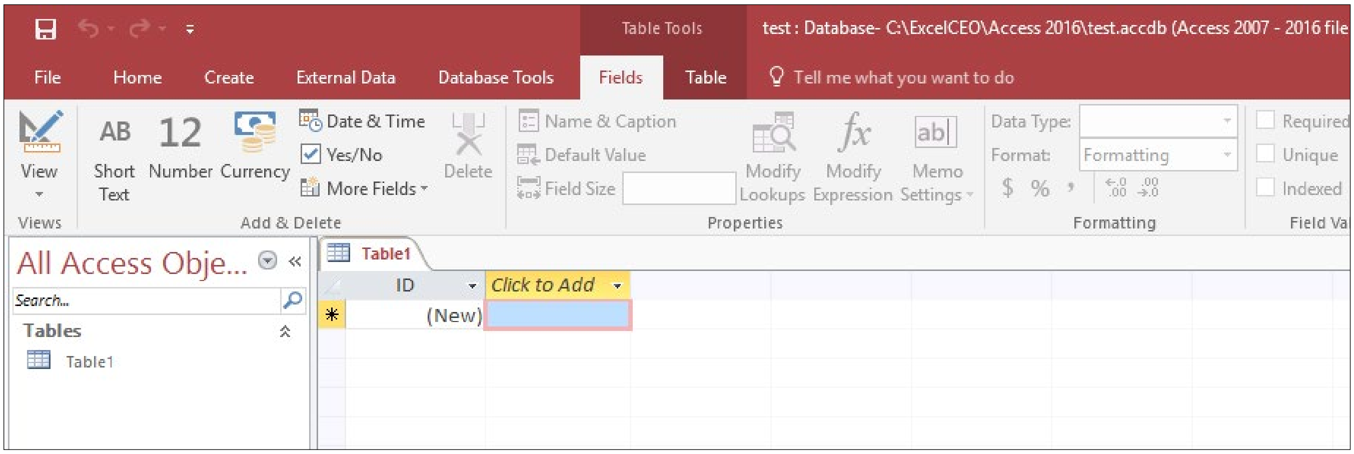
File Name: Database1.accdb
C:\Users\mecha\Documents\

File name: test.accdb
Save as type: Microsoft Access 2007 - 2016 Databases (*.accdb)

Figure 1.2

The File New Database dialog box opens. This is basically asking you where you want to store your new database. It is here where you will name the database and store it where you want it.

3. *Navigate to the* **C:\ExcelCEO\Access 2016** *folder.*
4. *Replace* **Database1** *(or the name of the default database that appears in the text box) with* ***test.accdb****.*
5. *In* **Backstage View***, make sure the path below the* **File Name** *box reads* **C:\ExcelCEO\Access 2016\** *(if not, repeat steps 2 – 5), and click the* **Create** *icon.*
6. *Click on the* **Enable Content** *command button if it appears (we'll discuss that in a minute).*



test : Database- C:\ExcelCEO\Access 2016\test.accdb (Access 2007 - 2016 file

Figure 1.3

A new Access database called test is created and it opens to a new table. You can create a simple database like we just did, or you can use the various templates and wizards that are available to assist you.

Notice that the title bar at the top reads, "test: Database" and then the file path followed by the file format in parentheses. It includes Access 2007 in the title because the underlying format of the database is in Access 2007; this is done to distinguish compatibility. Databases created with Access 2007 — 2016 can generally be used interchangeably. Note that you can open an Access 2003 database with Access 2007 or newer software, but you cannot open an Access 2007 — 2016 database using Access 2003 software.

Once you have a blank database, you need to bring data into the table, either by importing it from a database, spreadsheet, or other data source, or typing it in manually. Then you can build or import queries, forms, and/or reports so you can manipulate and show the results and analysis of that data. We'll go over those topics in later chapters.

Object Naming Conventions

At this point, let me explain about object naming conventions. When programming within an Access database, you will need to refer to many of the objects on different screens. Sometimes Access shows you the names of the objects, but it doesn't tell you if the object is a table, query, form, or report. When I create an object in an Access database, I like to name the object with an identifier that tells me what kind of object it is. Therefore, all of the tables, queries, forms, and reports that you will create in this course will begin with the following TLAs (TLA stands for "three-letter acronym" – just an example of my "unique" sense of humor):

Item Type	TLA
Table:	tbl
Query:	qry
Form:	frm
Report:	rpt
Macro:	mcr

You will see how I use these TLAs in later exercises. If you already have a naming convention that works for you, by all means continue with that method. But in this course, please use my method, as it will help you when taking the tests.

Open an Existing Database

Now we will close the blank database and open an existing database.

1. *Close the* **Access 2016 test.accdb** *database by clicking on the* **Close** *icon* × *in the upper-right corner of the screen.*
2. *In the* **Backstage View**, *click on the* **Open Other Files** *icon* Open Other Files, *navigate to* **C:\ExcelCEO\Access 2016** *and open the* **Nitey_Nite_2016.accdb** *file.*

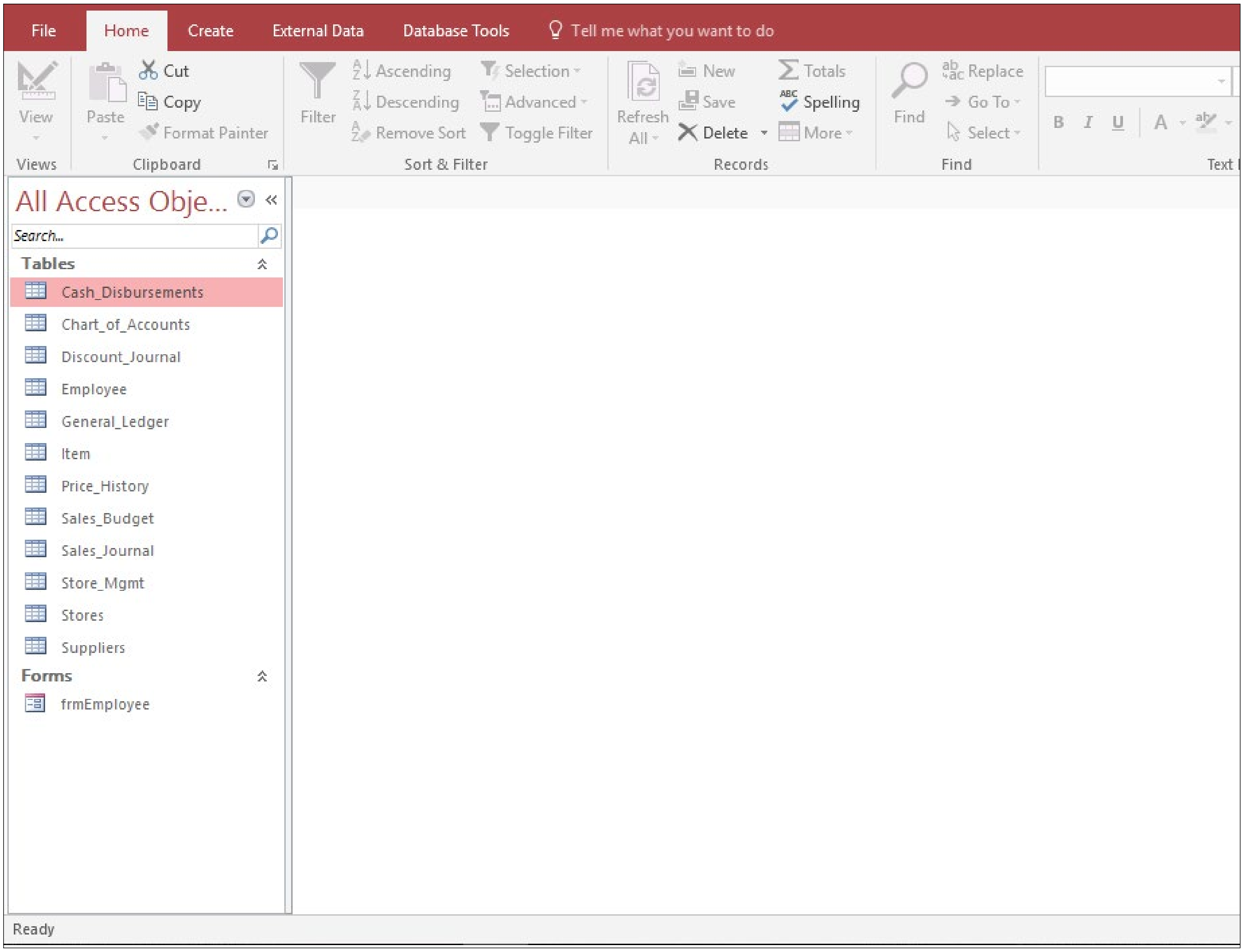


Figure 1.4

This is the database you will use throughout this course. If you've previously used Access 2003 or earlier versions of Access, you will notice that the Object Pane looks different. In this version there are two major differences: 1) There are no Tables, Queries, Forms, etc. sections on the left, and 2) a Security Warning appears just below the Ribbon. The Security Warning tells the user there is some content in the database that has been disabled. This would be things like macros, some VBA code, etc. that could potentially be harmful to your computer. The database is fine, so let's turn the Security Warning off.

3. *Click on the* **Enable Content** *command button (if it appears) and click* **OK**.

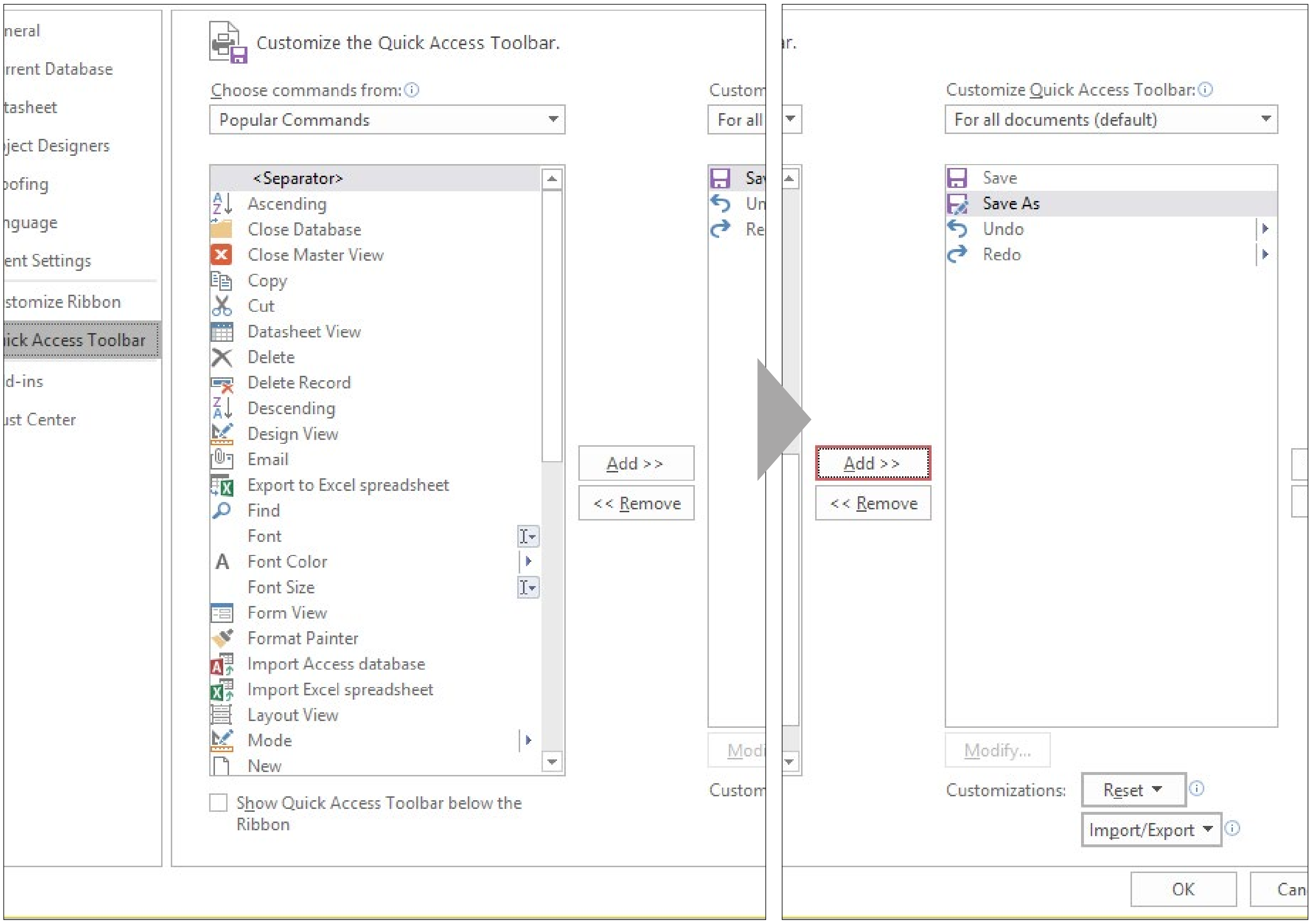
The Security Warning message disappears. After the first time you click the Enable Content button, that message will no longer appear.

Quick Access Toolbar

The first thing I want to do is introduce you to a headache reliever that will become more obvious in Chapter 2 and beyond, but let's hurry and get it out of the way. You will be saving a lot of objects throughout your Access 2016 and SQL training and to avoid frustration, we're going to make a quick change to the Quick Access Toolbar by adding the Save As icon with the Save icon. Trust me, you'll thank me for this one!

4. *Click on the* **File** *tab, click* **Options**, *then* **Quick Access Toolbar**.
5. *Under* **Popular Commands**, *scroll down to the* **Save As** Save As *icon, highlight it, then click* **Add** Add >>.
6. *Highlight the* **Save As** *icon below the* **Customize Quick Access Toolbar:** *section and click the* **Move Up** *arrow icon to move the* **Save As** *icon above the* **Undo** *icon, if necessary.*

Your Quick Access Toolbar is now setup with the Save As icon to the left of the Undo and Redo icons. The Save As icon allows you save your new Objects with a chance to name them as you want and what type of object, which is extremely valuable when you are creating variations of different objects. If you want, you can use the up and down arrows to the far-right to change the order of the Quick Access Toolbar icons.



Choose commands from: Popular Commands

Customize Quick Access Toolbar: For all documents (default)

□ Show Quick Access Toolbar below the Ribbon

Customizations:

Customize Quick Access Toolbar

7. *Click* **OK** *to save the changes and exit the* **Access Options** *dialog box.*

Now let's review this Access database. Access shows us the two existing types of objects currently in the Nitey_Nite_2016.accdb database, Tables and Forms, under the title of All Access Objects. In previous versions of Access, there was not a lot of organization to the Navigation Pane and we would have to organize the objects ourselves. To see what organization objects are possible, let's click the circled drop-down arrow to explore.

8. *Click on the circled drop-down menu arrow to the right of* **All Access Objects** *label.*

Here you see the menu with Object Type checked in the Navigate to Category section and All Access Objects checked in the Filter by Group section of the menu. Other Access object types in addition to Tables and Forms include Queries and Reports, both of which we will use in-depth later. You can also organize your object types by Tables and Related Views, Created Date, Modified Date, or as Custom categories.

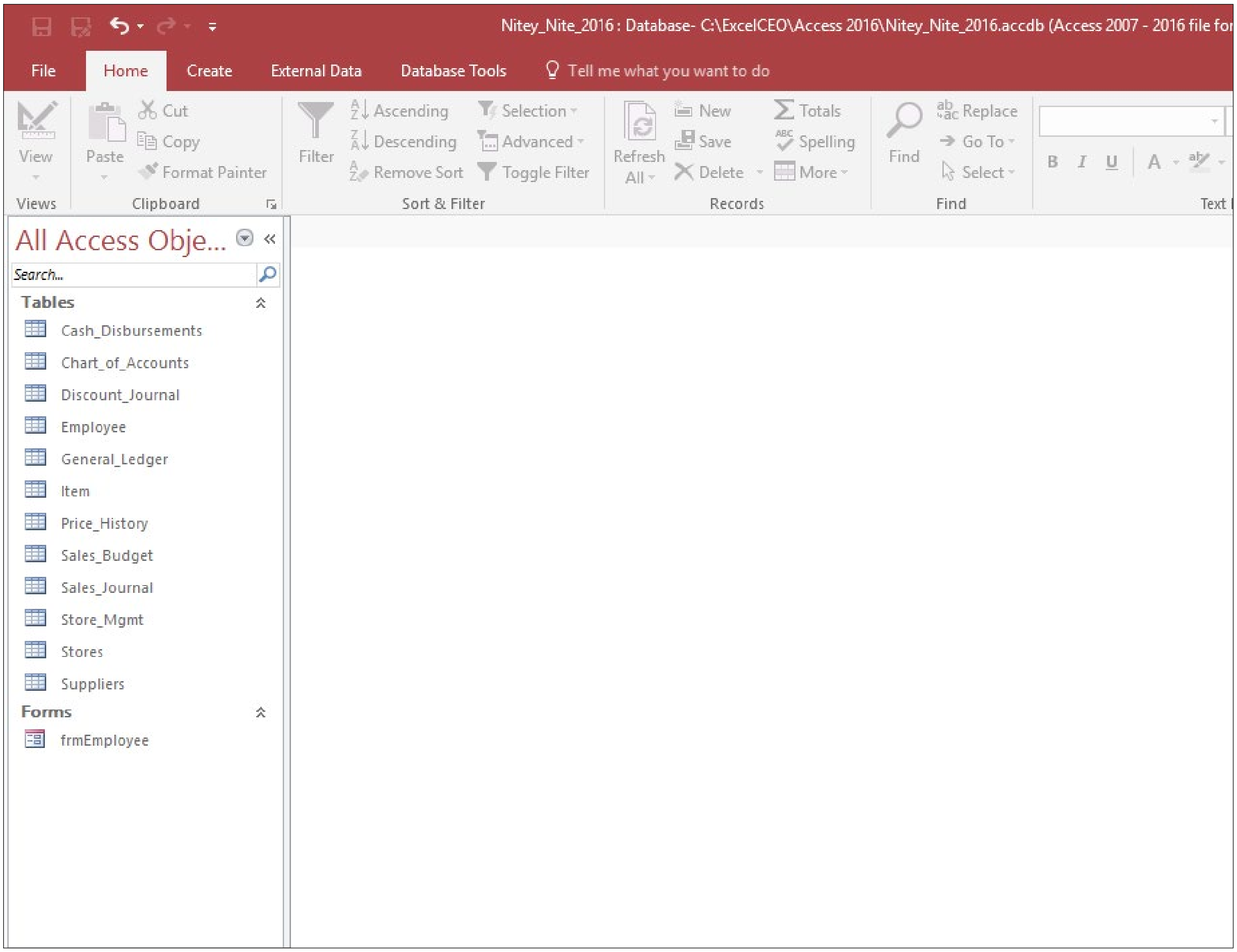


Figure 1.5

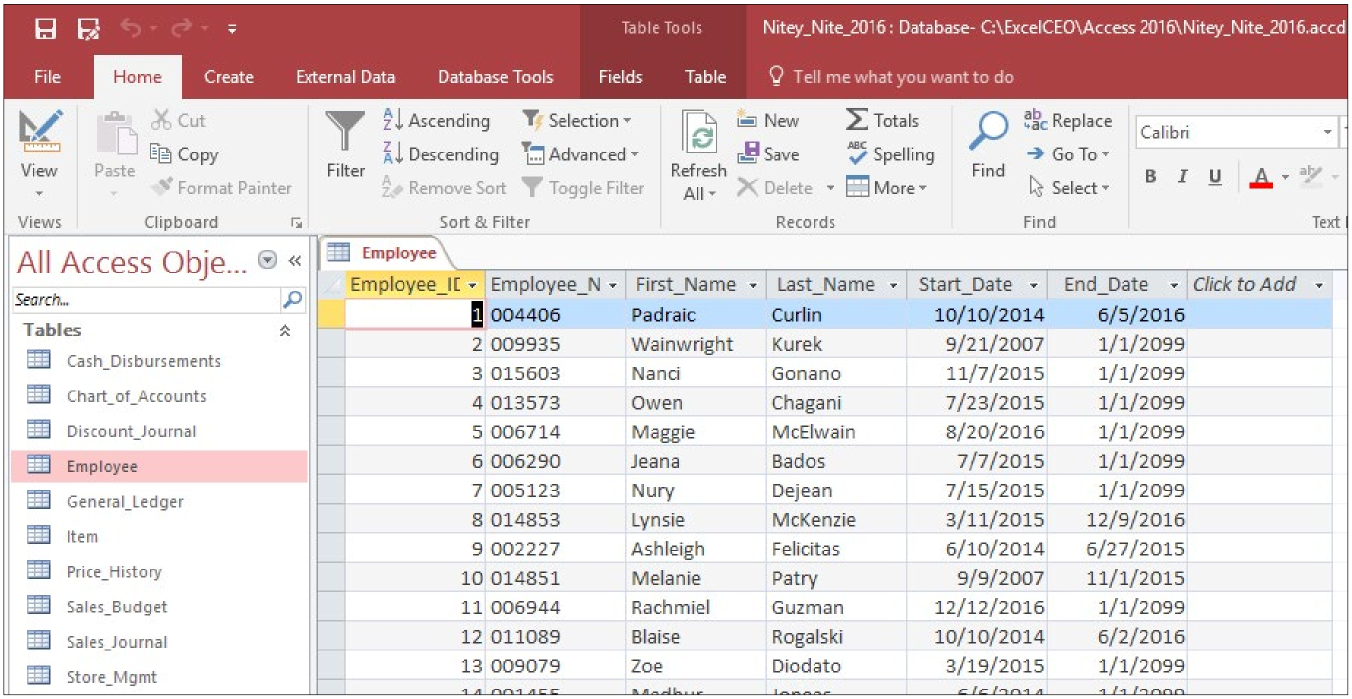
In the Tables section, there are 12 tables. Let's briefly review each of the tables.

- Cash_Disbursements, a journal of various payments.
- Chart_of_Accounts, the company's account rollup structure.
- Discount_Journal, a separate record of discounts not captured by the Sales system.
- Employee, a list of all current and former employees.
- General_Ledger, a partial general ledger for the company.
- Item, a table containing descriptions of each major item the company sells.
- Price_History, a historical reference of standard sale prices and costs per item.
- Sales_Budget, a store-by-store listing of each store's sales budget for three years;
- Sales_Journal, a detail listing of each sale of mattresses, pillows, warranties, delivery charges and promotional discounts;
- Store_Mgmt, a list of the managers at each store, by Employee ID;
- Stores, a list of each store and pertinent information;
- Suppliers, a list of suppliers and their contact information.

Open a Table

Let's open the Employee table.

9. *Double-click on the* **Employee** *table.*



Employee_II	Employee_N	First_Name	Last_Name	Start_Date	End_Date	Click to Add
1	004406	Padraic	Curlin	10/10/2014	6/5/2016	
2	009935	Wainwright	Kurek	9/21/2007	1/1/2099	
3	015603	Nanci	Gonano	11/7/2015	1/1/2099	
4	013573	Owen	Chagani	7/23/2015	1/1/2099	
5	006714	Maggie	McElwain	8/20/2016	1/1/2099	
6	006290	Jeana	Bados	7/7/2015	1/1/2099	
7	005123	Nury	Dejean	7/15/2015	1/1/2099	
8	014853	Lynsie	McKenzie	3/11/2015	12/9/2016	
9	002227	Ashleigh	Felicitas	6/10/2014	6/27/2015	
10	014851	Melanie	Patry	9/9/2007	11/1/2015	
11	006944	Rachmiel	Guzman	12/12/2016	1/1/2099	
12	011089	Blaise	Rogalski	10/10/2014	6/2/2016	
13	009079	Zoe	Diodato	3/19/2015	1/1/2099	
14	001455	Madhur	Jonas	6/6/2014	1/1/2099	

Figure 1.6

This view of a table is called the ***Datasheet View***. Notice that a separate tab appears at the top of the table that shows the name of the table, and many of the icons in the Home tab are now activated. This is a simple table that lists all of the past and present employees of Nitey_Nite Mattresses. Doesn't it look just like an Excel spreadsheet? In a database table, columns are called ***fields*** and rows are called ***records***.

> ***Tip***: *You can select the various cells, records, or fields with your mouse, or by using the* **up arrow**, **down arrow**, **PgUp**, **PgDn**, *and/or* **Tab** *keys on your keyboard. Select the first record in a table by pressing* **[Ctrl]+[Home]** *or the last record by pressing* **[Ctrl]+[End]**. *You can also use the record selectors at the bottom of the screen.*

The Employee_No field contains the employee's identification number as assigned by the Nitey-Nite Human Resources department. Notice how part of the name of that field is cut off. To see the entire name of the field, double-click the margin line between Employee_No and First_Name, just like how you adjust the margins on an Excel spreadsheet.

10. *Double-click on the column line between* **Employee_No** *and* **First_Name**.

Now you can see the entire field name. Even though the Employee_No looks like a number, it is actually a ***text field***. The database managers of Nitey-Nite Mattresses use the Employee_No field in another application (a payroll application) and that system requires the Employee_No field to be a text field. You can usually tell if a number in a field is formatted as a number or as text because: 1) typically number fields are right-justified and text fields are left-justified (unless changed by the database designer), and 2) a number field cannot contain leading zeroes. Look at the first employee record, Padraic Curlin, whose Employee_No is 004406. Since it has leading zeros, you know it is a text field. Another "number" that should be formatted as text is a social security number. Even though it is a number, a social security number can have leading zeros, like 003-34-9876. We talked a lot about number and text fields in Excel, and that discussion holds true here as well, so I won't elaborate on it any further.

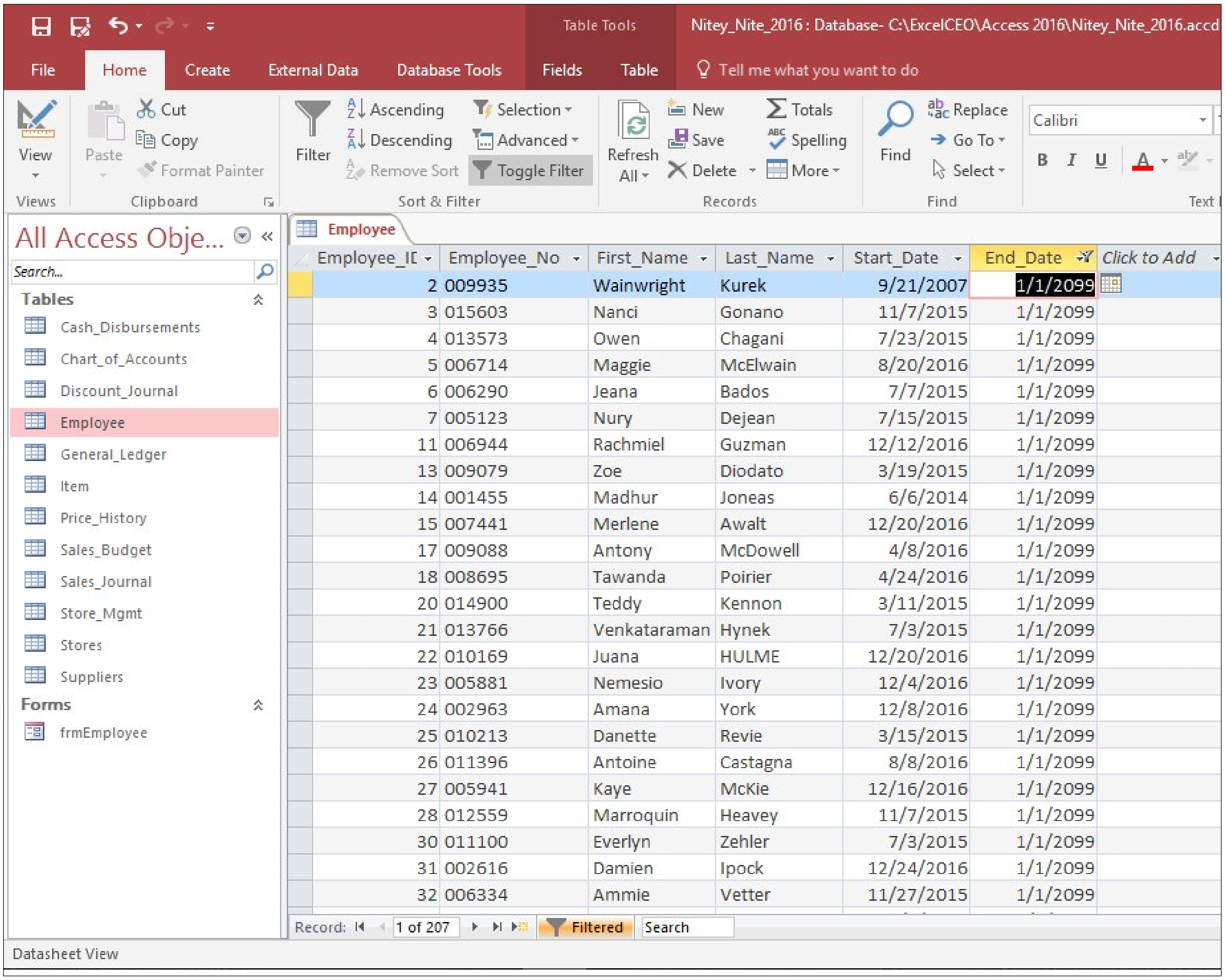
The First_Name and Last_Name fields should be self-explanatory. These are text fields and are the first and last names of the employee. The Start_Date and End_Date fields should also be easy to understand. These fields are formatted as dates, which is more closely associated to a number than a text. A date in an Access table, as with Excel, is a number that is formatted as a date. January 1, 1900, is represented as the number one, January 2, 1900, is two, and so forth, just like in Excel. Calendar functions can assist with this. When you set up the field as a date type, it automatically displays the data as a date. In this table, the Start_Date is the date the person became an employee of Nitey-Nite. The End_Date is the date the person left Nitey-Nite as an employee. Notice that many of the end dates are 1/1/2099. This was done to indicate that the employee is a current employee of Nitey-Nite. The database designers at Nitey-Nite, in their infinite wisdom, know that programs sometimes don't handle ***NULL*** fields (or fields with no value in them) very well, and it is usually good database practice to put some type of value in every field. In this case, they picked a date far enough into the future so that users could clearly see that it is not a true end date. January 1, 2099, was as good of a date as any other, so that was used.

Filter a Table

You can do many things with a table that is open. A common functionality you can do within a table is to ***filter*** the data. Let's suppose that you want to filter the Employee table to show only the current employees. That's easy, as all current employees have an End_Date of 1/1/2099. You do that by using the AutoFilter functionality. The ***AutoFilter*** in Access works just like it does in Excel – simply click on the drop-down arrow and choose the values you want.

11. *Click on the drop-down arrow next to the* **End_Date** *field, uncheck the* **(Select All)** *box, scroll down to the bottom of the list, check the* **1/1/2099** *box, and click* **OK**.

You should get a record set that looks like this:



Employee_IL	Employee_No	First_Name	Last_Name	Start_Date	End_Date	Click to Add
2	009935	Wainwright	Kurek	9/21/2007	1/1/2099	
3	015603	Nanci	Gonano	11/7/2015	1/1/2099	
4	013573	Owen	Chagani	7/23/2015	1/1/2099	
5	006714	Maggie	McElwain	8/20/2016	1/1/2099	
6	006290	Jeana	Bados	7/7/2015	1/1/2099	
7	005123	Nury	Dejean	7/15/2015	1/1/2099	
11	006944	Rachmiel	Guzman	12/12/2016	1/1/2099	
13	009079	Zoe	Diodato	3/19/2015	1/1/2099	
14	001455	Madhur	Joneas	6/6/2014	1/1/2099	
15	007441	Merlene	Awalt	12/20/2016	1/1/2099	
17	009088	Antony	McDowell	4/8/2016	1/1/2099	
18	008695	Tawanda	Poirier	4/24/2016	1/1/2099	
20	014900	Teddy	Kennon	3/11/2015	1/1/2099	
21	013766	Venkataraman	Hynek	7/3/2015	1/1/2099	
22	010169	Juana	HULME	12/20/2016	1/1/2099	
23	005881	Nemesio	Ivory	12/4/2016	1/1/2099	
24	002963	Amana	York	12/8/2016	1/1/2099	
25	010213	Danette	Revie	3/15/2015	1/1/2099	
26	011396	Antoine	Castagna	8/8/2016	1/1/2099	
27	005941	Kaye	McKie	12/16/2016	1/1/2099	
28	012559	Marroquin	Heavey	11/7/2015	1/1/2099	
30	011100	Everlyn	Zehler	7/3/2015	1/1/2099	
31	002616	Damien	Ipock	12/24/2016	1/1/2099	
32	006334	Ammie	Vetter	11/27/2015	1/1/2099	

Figure 1.7

Notice the ***Calendar*** icon, ▦, that appears to the right of the selected record. This is a new feature since Access 2007 that allows you to choose a date from a calendar instead of typing the date into the record. This is very beneficial for people who can't remember, "*30 days hath September, April, June, and November...*", and try to input 4/31/2016 as a valid date. To use this icon, just click on it and a Calendar will appear. To choose a date, just click on the appropriate date and the record will be changed. Be careful though. Microsoft sometimes makes it way too easy to change things.

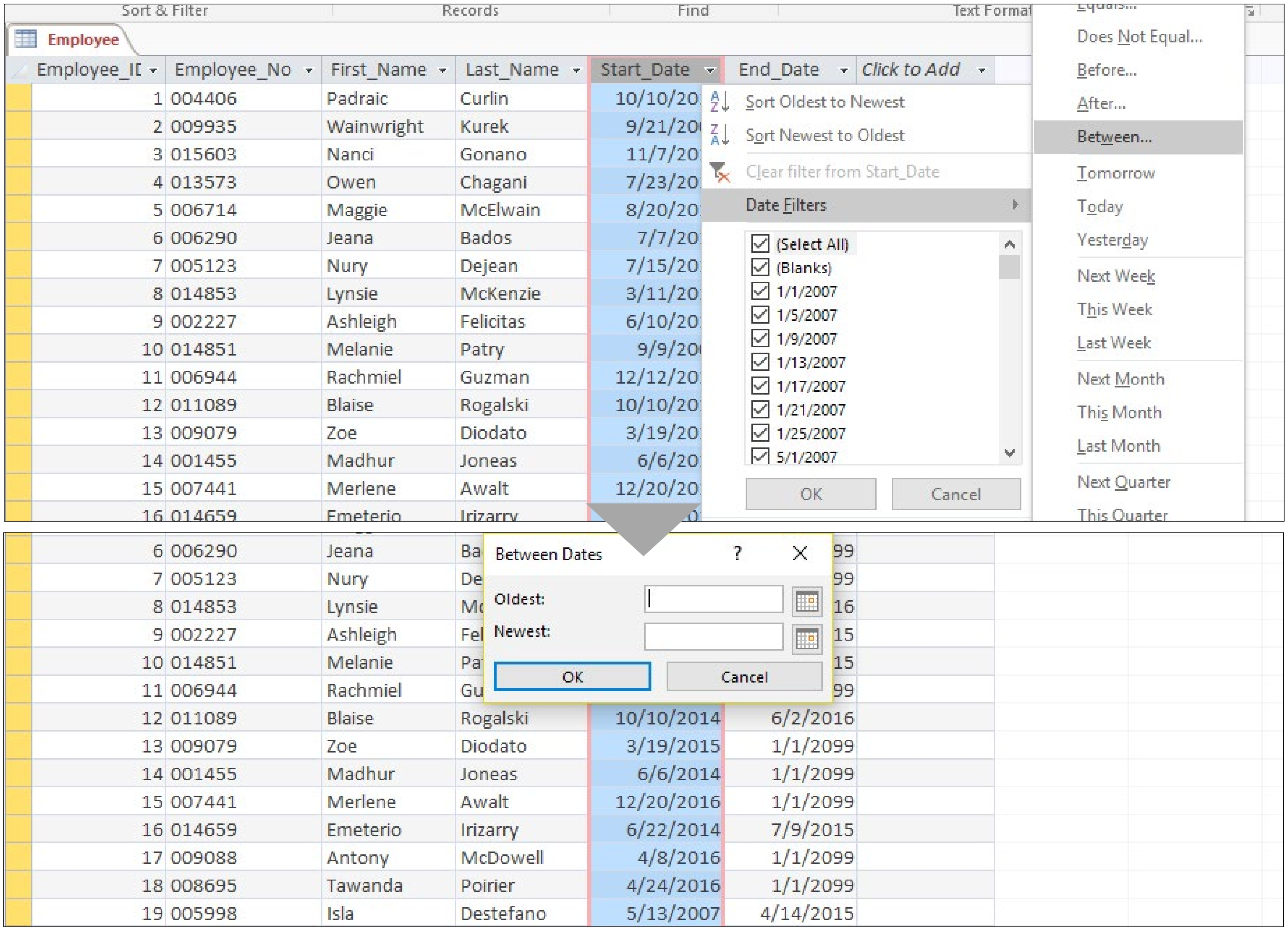
Notice that the End_Date field now displays a filter icon, indicating that the field is filtered. To turn the filter off (or to show all records), click the drop-down arrow and check the (Select All) box.

12. *Click on the* **End_Date** *drop-down menu, check the* **Select All** *box and click* **OK**.

The table returns to its original look.

You can also filter a table based on more than one criterion. Let's suppose you want to have a list of all the employees who started with Nitey-Nite between 12/1/2016 and 12/15/2016. You want those employees to attend a New Employee seminar and you need that list of employees. You can specify such criteria by using the AutoFilter functionality.

13. *Click on the drop-down arrow next to* **Start_Date**, *point to* **Date Filters** *and choose* **Between…**

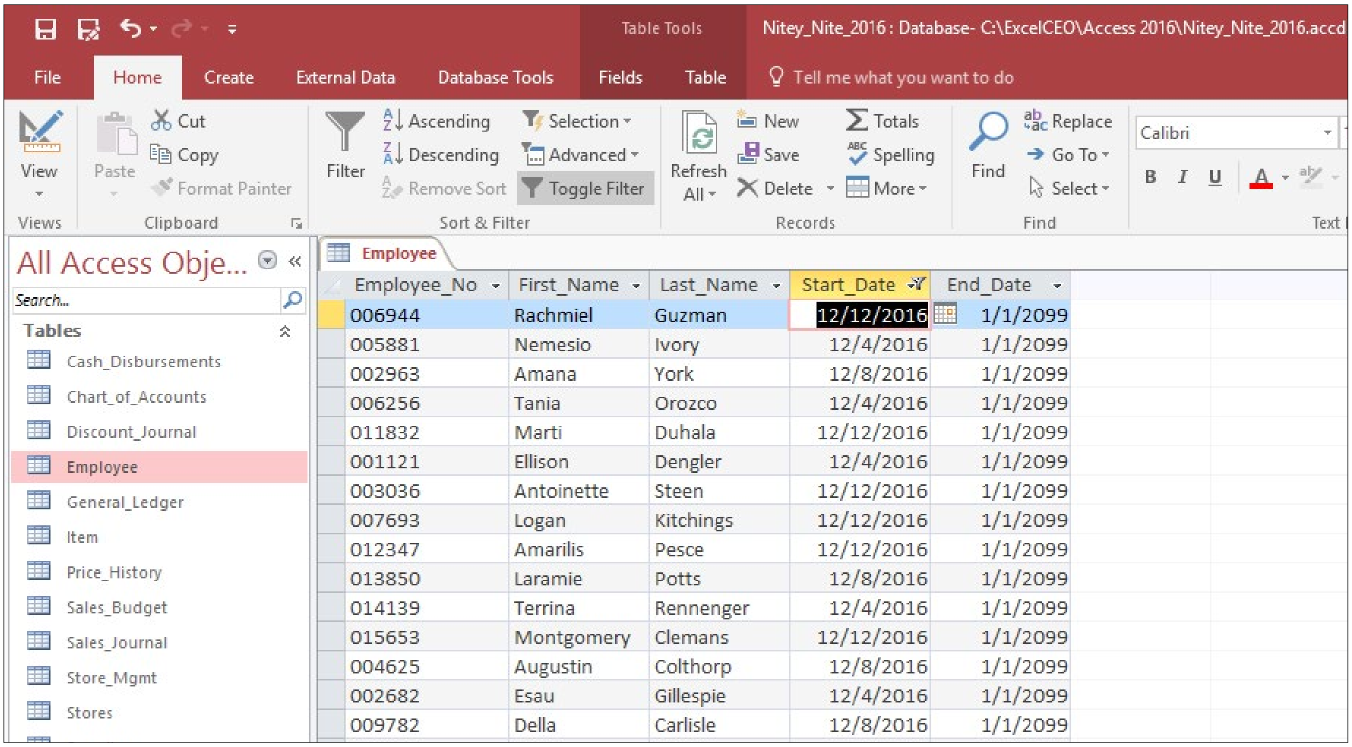


Employee_ID	Employee_No	First_Name	Last_Name	Start_Date	End_Date	Click to Add
1	004406	Padraic	Curlin	10/10/20		
2	009935	Wainwright	Kurek	9/21/20		
3	015603	Nanci	Gonano	11/7/20		
4	013573	Owen	Chagani	7/23/20		
5	006714	Maggie	McElwain	8/20/20		
6	006290	Jeana	Bados	7/7/20		
7	005123	Nury	Dejean	7/15/20		
8	014853	Lynsie	McKenzie	3/11/20		
9	002227	Ashleigh	Felicitas	6/10/20		
10	014851	Melanie	Patry	9/9/20		
11	006944	Rachmiel	Guzman	12/12/20		
12	011089	Blaise	Rogalski	10/10/20		
13	009079	Zoe	Diodato	3/19/20		
14	001455	Madhur	Joneas	6/6/20		
15	007441	Merlene	Awalt	12/20/20		
16	014659	Emeterio	Irizarry			

6	006290	Jeana	Ba		99
7	005123	Nury	De		99
8	014853	Lynsie	Mc		16
9	002227	Ashleigh	Fel		15
10	014851	Melanie	Pa		15
11	006944	Rachmiel	Gu		99
12	011089	Blaise	Rogalski	10/10/2014	6/2/2016
13	009079	Zoe	Diodato	3/19/2015	1/1/2099
14	001455	Madhur	Joneas	6/6/2014	1/1/2099
15	007441	Merlene	Awalt	12/20/2016	1/1/2099
16	014659	Emeterio	Irizarry	6/22/2014	7/9/2015
17	009088	Antony	McDowell	4/8/2016	1/1/2099
18	008695	Tawanda	Poirier	4/24/2016	1/1/2099
19	005998	Isla	Destefano	5/13/2007	4/14/2015

Figure 1.8

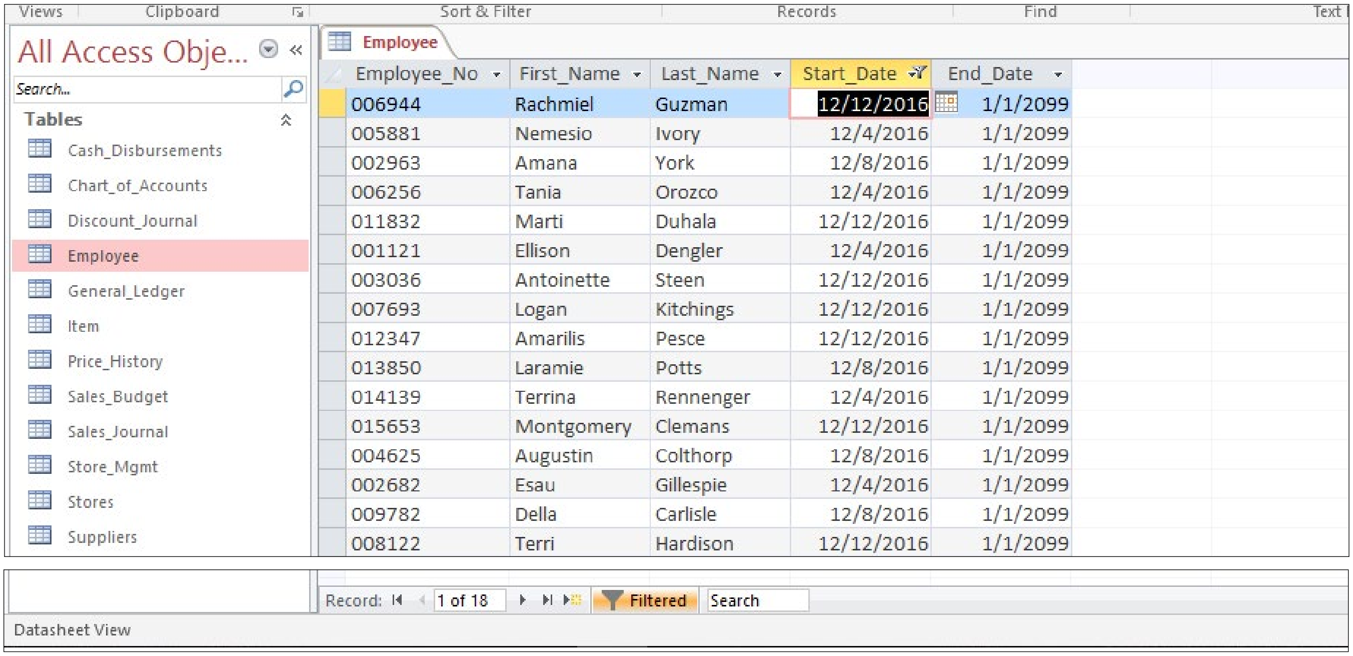
14. *In the* **Oldest** *box, type* ***12/1/2016****, type* ***12/15/2016*** *in the* **Newest** *box, and click* **OK***.*



Employee_No	First_Name	Last_Name	Start_Date	End_Date
006944	Rachmiel	Guzman	12/12/2016	1/1/2099
005881	Nemesio	Ivory	12/4/2016	1/1/2099
002963	Amana	York	12/8/2016	1/1/2099
006256	Tania	Orozco	12/4/2016	1/1/2099
011832	Marti	Duhala	12/12/2016	1/1/2099
001121	Ellison	Dengler	12/4/2016	1/1/2099
003036	Antoinette	Steen	12/12/2016	1/1/2099
007693	Logan	Kitchings	12/12/2016	1/1/2099
012347	Amarilis	Pesce	12/12/2016	1/1/2099
013850	Laramie	Potts	12/8/2016	1/1/2099
014139	Terrina	Rennenger	12/4/2016	1/1/2099
015653	Montgomery	Clemans	12/12/2016	1/1/2099
004625	Augustin	Colthorp	12/8/2016	1/1/2099
002682	Esau	Gillespie	12/4/2016	1/1/2099
009782	Della	Carlisle	12/8/2016	1/1/2099

Figure 1.9

Now you have a list of all the employees that started between 12/1/2016 and 12/15/2016. There are 18 records in that list. You can see the number of records in the filtered list by looking at the bottom of the screen. You should see something like the following:



Employee_No	First_Name	Last_Name	Start_Date	End_Date
006944	Rachmiel	Guzman	12/12/2016	1/1/2099
005881	Nemesio	Ivory	12/4/2016	1/1/2099
002963	Amana	York	12/8/2016	1/1/2099
006256	Tania	Orozco	12/4/2016	1/1/2099
011832	Marti	Duhala	12/12/2016	1/1/2099
001121	Ellison	Dengler	12/4/2016	1/1/2099
003036	Antoinette	Steen	12/12/2016	1/1/2099
007693	Logan	Kitchings	12/12/2016	1/1/2099
012347	Amarilis	Pesce	12/12/2016	1/1/2099
013850	Laramie	Potts	12/8/2016	1/1/2099
014139	Terrina	Rennenger	12/4/2016	1/1/2099
015653	Montgomery	Clemans	12/12/2016	1/1/2099
004625	Augustin	Colthorp	12/8/2016	1/1/2099
002682	Esau	Gillespie	12/4/2016	1/1/2099
009782	Della	Carlisle	12/8/2016	1/1/2099
008122	Terri	Hardison	12/12/2016	1/1/2099

Record: 1 of 18

Figure 1.10

Record Selectors

Let's talk about ***Record Selectors***. With your cursor on the first record, the record selector shows Record 1 of 18. The Filtered indicator reminds you that the list you are looking at is filtered by some criteria. If you click on the next record (or row in the table), it will show 2 of 18, and so forth. To go to the last record in the table, you can click on the Last record icon ⏭, or you can use the keyboard by pressing [Ctrl]+[End]. [Ctrl]+[Home], or clicking on the First record icon ⏮, brings you to the first record of the table. The New record icon ▶✱, takes you to a new row to input a new record. The Next ▶, and Previous ◀, record icons take you one record forward and one record back, respectively. You can also choose a record by simply clicking on it. Press the [PgUp] (Page Up) and [PgDn] (Page Down) buttons on your keyboard to scroll up one page and down one page for lists contained on multiple pages.

> *Review Questions: It is now time to complete the hands-on Review Questions. Log on to www.ExcelCEO.com with your Email address and Password, click on the* **Access 2016 and SQL Review Questions, Chapter 1, Section 1 of 2** *option in your Main Menu and complete the Review Questions.*

The Office Ribbon, Groups, and Icons

To use most of the functionality that Access 2016 offers, you will most likely use the icons within the Office Ribbon. The Office Ribbon appears at the top of the screen and generally replaces the Toolbars used in Access 2003. When the Home tab of the Office Ribbon is selected, it appears like this:

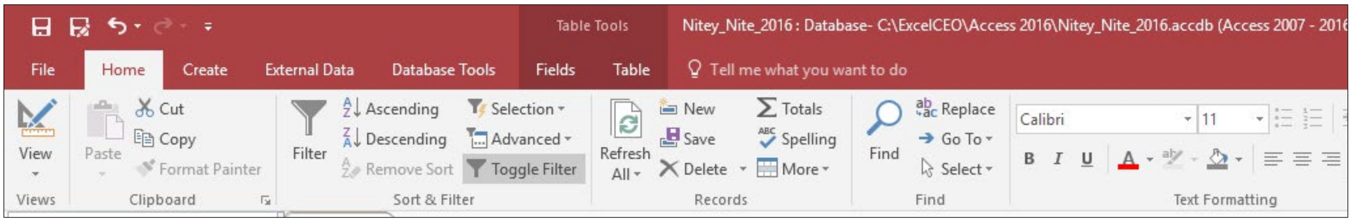


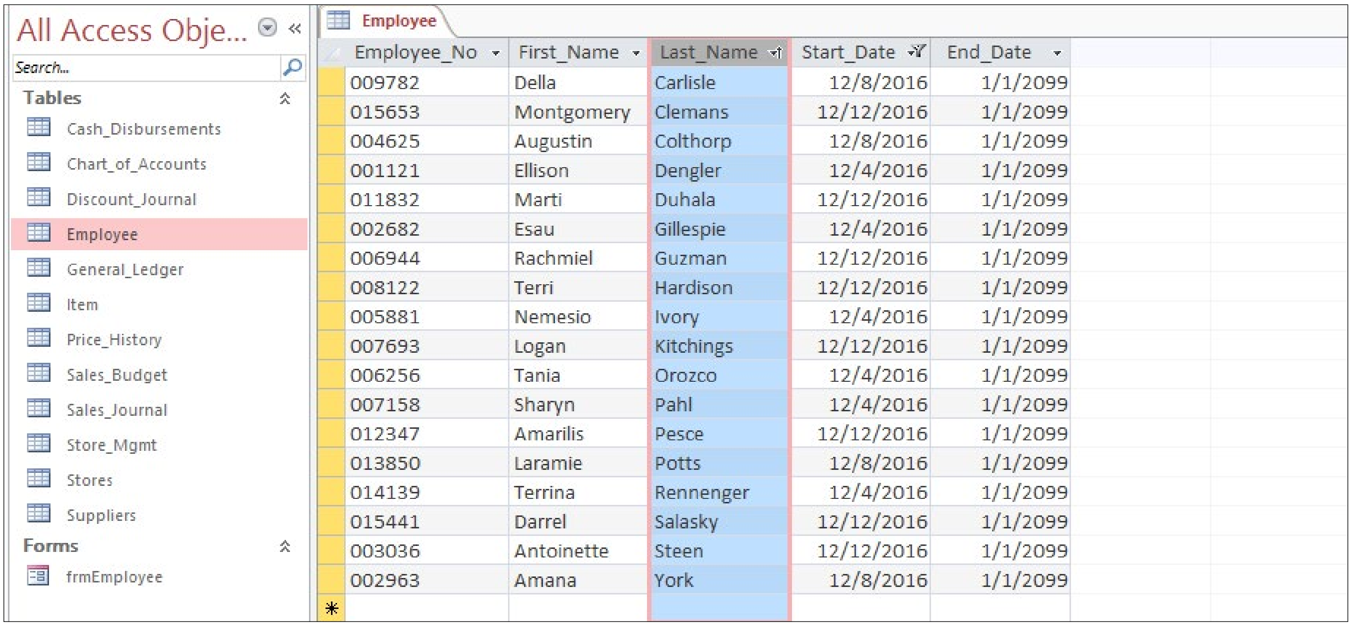
Figure 1.11

There are four parts to the Office Ribbon: The Tabs (Home, Create, External Data, and Database Tools appear in every "view" of the Ribbon), ***Contextual Tabs*** (which appear above the standard tabs. For example, when working with tables, the Table Tools contextual tab displays the Datasheet tab), ***Groups*** (such as View, Clipboard, and Font within the Home tab), and ***Icons*** (like the Paste, Cut, Copy, and Format Painter icons within the Clipboard Group of the Home tab). With a table open, many of the icons are activated and are available for use. Some of the functionality is not activated (meaning it is not available), such as the Bullet and Numbering icons in the Text Formatting group of the Home tab.

Sorting a Table

Now that you have a filtered list, you may want to sort the list by last name in ascending order. To do this, simply click on the field or any record within the field you want to sort, and then click the ***Ascending*** icon in the Sort & Filter group of the Home tab.

15. *Click on the* **Last_Name** *field (or any record within that field) and click the* **Ascending** *icon in the* **Sort & Filter** *group of the* **Home** *tab.*



Employee_No	First_Name	Last_Name	Start_Date	End_Date
009782	Della	Carlisle	12/8/2016	1/1/2099
015653	Montgomery	Clemans	12/12/2016	1/1/2099
004625	Augustin	Colthorp	12/8/2016	1/1/2099
001121	Ellison	Dengler	12/4/2016	1/1/2099
011832	Marti	Duhala	12/12/2016	1/1/2099
002682	Esau	Gillespie	12/4/2016	1/1/2099
006944	Rachmiel	Guzman	12/12/2016	1/1/2099
008122	Terri	Hardison	12/12/2016	1/1/2099
005881	Nemesio	Ivory	12/4/2016	1/1/2099
007693	Logan	Kitchings	12/12/2016	1/1/2099
006256	Tania	Orozco	12/4/2016	1/1/2099
007158	Sharyn	Pahl	12/4/2016	1/1/2099
012347	Amarilis	Pesce	12/12/2016	1/1/2099
013850	Laramie	Potts	12/8/2016	1/1/2099
014139	Terrina	Rennenger	12/4/2016	1/1/2099
015441	Darrel	Salasky	12/12/2016	1/1/2099
003036	Antoinette	Steen	12/12/2016	1/1/2099
002963	Amana	York	12/8/2016	1/1/2099

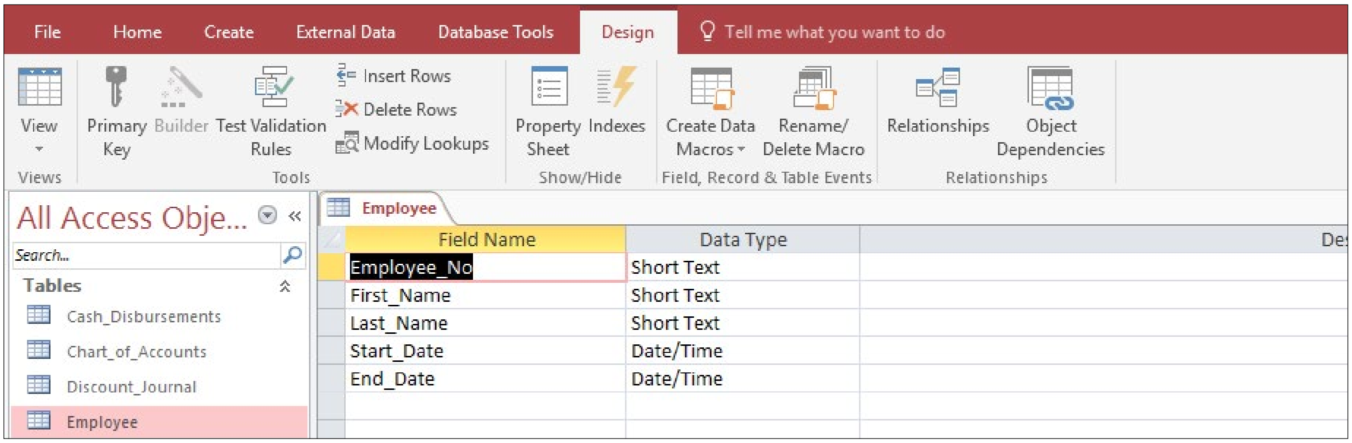
Figure 1.12

Since you've probably already taken the Excel course, you know how to use the Ascending and Descending icons, so I won't explain it in detail. If you don't know how to use the Sort icons by now, you should probably review the ExcelCEO Excel 2016 course. The primary difference between using the sort icons in Access and Excel is that you can choose the entire field in Access OR you can choose any single record within that field, then click on the icon to perform the sort. If you select an entire field in Excel, it will sort ONLY that field, which can do major damage if there is more than one field in the table. Note that there is a small ascending arrow next to the Last_Name field which indicates that this field is sorted in ascending order.

Table Design

When designing a table, ***Field Properties*** are critically important. Now that you've seen the data in the Employee table, let's look at the behind-the-scenes design of the table.

16. *Click on the* **View** *icon (Be sure to click on the image, not the down arrow).*



Field Name	Data Type	Des
Employee_No	Short Text	
First_Name	Short Text	
Last_Name	Short Text	
Start_Date	Date/Time	
End_Date	Date/Time	

Figure 1.13

This view of the design behind a table is called Table ***Design View***. The Table Design View is divided into two sections: The first section contains the Field Name, Data Type and Description of each field. The ***Field Name*** should be a name that adequately describes the data in the field. You wouldn't want to name a field that contains vendor phone numbers something like "empl_type" or "FieldXYZ", so try to be descriptive but concise when naming fields. As with naming fields in Excel, I encourage you not to use spaces in field names. Trust me, it will simplify your programming life if you follow this simple advice.

A ***Data Type*** is the characteristic of a field that determines what type of data it can contain. Data types in an Access table include Short and Long Text, Number, Date/Time, Currency, AutoNumber, Yes/No, OLE Object, Hyperlink, Attachment (new in Access 2010), Calculated, and Lookup Wizard. The next few paragraphs discuss each data type available.

Short Text: This data type is the most common and can contain up to 255 characters (numbers, letters, and/or special characters). A good general rule of thumb is that if the field could contain combinations of numbers and characters, it will most likely be a Text data type. Another good rule of thumb is that if the field contains numbers that you don't perform calculations on (like phone numbers and Social Security numbers), you will probably use a Text data type. An exception to this general rule is if you want to sort using that field (like sorting by Zip code), you may want to use a number field. If you had numbers like 8, 49 and 157 in a text field, it would perform an ascending sort in the reverse order (157, 49 and 8) as the "1" in 157 comes before the "4" in 49, similar to sorting on alpha characters.

Long Text: The Long Text type replaced the Memo fields in previous versions of Access. Long Text fields are used to store long text strings that could contain more than 255 characters. Entries in this field could be things like descriptive comments. A Long Text field can contain any number, letter, or special character, and can hold up to 65,535 characters.

Number: You use a Number type when you plan to do calculations on the entries in the field, like number of hours worked or summing the number of products sold. You can also use it to store dollar amounts, like sales or expenses. You should use a Number type whenever possible in fields that are designated as Primary Keys. Using a Number type as a Primary Key helps speed up processing when you link queries and tables. We'll discuss primary keys later in this chapter.

Date/Time: As implied, this field contain date and time data. As such, you can sort the values in this field chronologically by date. You can also perform calculations on this data type to calculate differences between dates, like calculating someone's age ((now() – birthdate)/365.25 = age). You can use a variety of formats available to make the date appear just like you want it. Calendar icons are available to assist with Date/Time fields.

Currency: A Currency field works just like a number field except it maintains a currency format on the numbers in the field. It can contain up to 15 characters to the left of the decimal point, and four digits to the right of the decimal point, so the largest number you can have in a Currency field is $999,999,999,999,999.9999.

AutoNumber: When you use an AutoNumber type, Access will force each new record in that field to be a new number, one higher than the last one created. You cannot physically change the numbers in this field, thus Access helps to ensure that each record has a unique number. I like to use an AutoNumber as a Primary Key whenever I can. There are two types of AutoNumber: Long Integer and Replication ID. A Long Integer is the most common type used, and you can have it create the new number incrementally or randomly. Incrementally is typically the best choice.

Yes/No: I really like this field. It contains simple Yes/No data, like a check box. When you create a check box, the default data type in the field is Yes/No. You can choose to make it display Yes/No, On/Off, or True/False.

OLE Object: You use this data type when you want to embed (or link) an object in your database. But be careful when using this data type. It can take up a lot of processing resources. There are typically other ways to get data stored in this format, but it is available if you need to use it.

Hyperlink: You already know about Hyperlinks from the Excel course, so I won't review the definition here. A hyperlink reference in Access can contain up to four parts, separated by a pound sign (#):

- **Display Text**: This is text that is displayed instead of the full hyperlink address (like "Microsoft" instead of www.microsoft.com).
- **Address**: The URL (Universal Resource Locator) path (like http://www.microsoft.com).
- **SubAddress**: A site within the referenced page (like http://www.microsoft.com/**windows/default.mspx**)
- **ScreenTip**: The message that is displayed when the mouse is positioned over the hyperlink.

When creating a hyperlink, only the address is required, unless you need to point to a certain page within the website; then the subaddress is required as well.

Attachment: This data type was new in Access 2010 files. You can attach images, spreadsheet files, documents, charts, and other types of supported files to the records in your database, much like you attach files to e-mail messages. You can also view and edit attached files, depending on how the database designer sets up the Attachment field. Attachment fields provide greater flexibility than

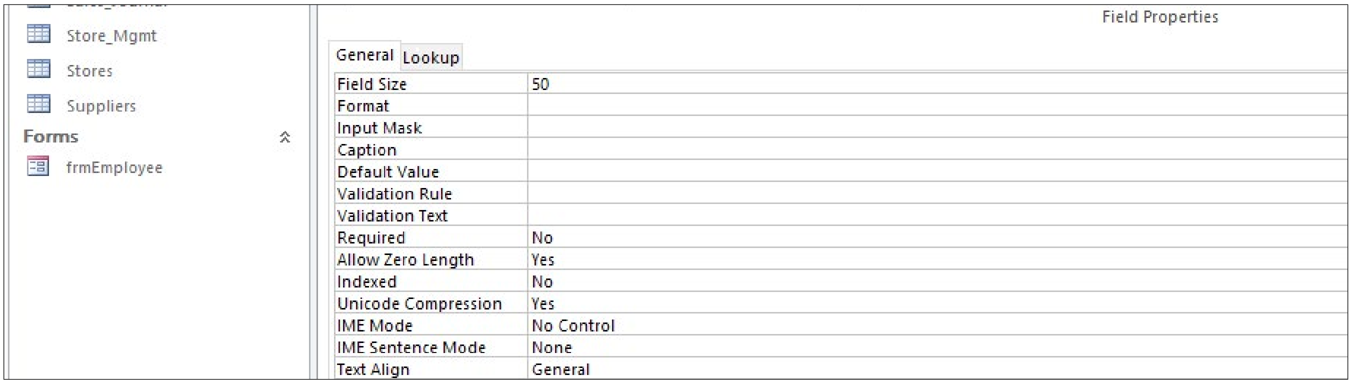
OLE Object fields, and they use storage space more efficiently because they don't create a bitmap image of the original file.

Calculated: This data type was new in Access 2010. It allows the database designer to create a calculated field in the table, which was previously available only in a query, form, or report.

Lookup Wizard…: The Lookup Wizard contains values that are limited to a list of values you enter. When you use this data type, it launches a wizard to help you design the field.

Field Properties

The second section of the table design screen contains the ***Field Properties*** of the field that is selected. Different properties exist for the various data types, and I won't review all of them here. However, there are a few I want to touch on. In our table, the Employee_No field is selected, and the Field Properties section reflects the properties of that field.



Field Size	50
Format	
Input Mask	
Caption	
Default Value	
Validation Rule	
Validation Text	
Required	No
Allow Zero Length	Yes
Indexed	No
Unicode Compression	Yes
IME Mode	No Control
IME Sentence Mode	None
Text Align	General

Figure 1.14

The ***Field Size*** property indicates the maximum number of characters you can input into any record in the field. The default value for this field (when using a Short Text data type) is 50, but you can use up to 255 characters. If the data type is a Number, you can choose the field size from a list of options, including Byte (one byte), Integer, Long Integer, Single, Double, Replication ID, and Decimal.

The ***Format*** property determines the appearance of the value, like forcing upper- and lower-case letters. With a Number data type, Format can include General Number, Currency, Euro, Fixed, Standard, Percent and Scientific. There is no default format for a Text data type.

The ***Input Mask*** is an interesting property. It's kind of a template that allows you to enter data in a certain way, but stores the data without the format. A good example is a phone number. Typically, phone numbers appear like (123) 456-7890. When inputting a phone number into the table, with a phone number input mask, you can type in the numbers like 1234567890, but it appears like (123) 456-7890. The data is stored as 1234567890, cutting down on the number of characters stored in the table. This is a very good feature to use when you have inexperienced users inputting data into your tables.

You can create a ***Validation Rule*** property to check data against a certain criterion. For example, if you have a birth date field, you can create a validation rule that says that the birth date cannot be greater than today's date. A ***Validation Text*** is a message that appears when the data entered fails the validation rule. In the birth date example, you can create a message that says something like, "*You can't enter a birth date for someone who hasn't been born yet.*"

The ***Required*** property can be set to Yes if you require the user to input some value in the field. In an Employee database, you may require certain fields such as first name, last name, and hire date.

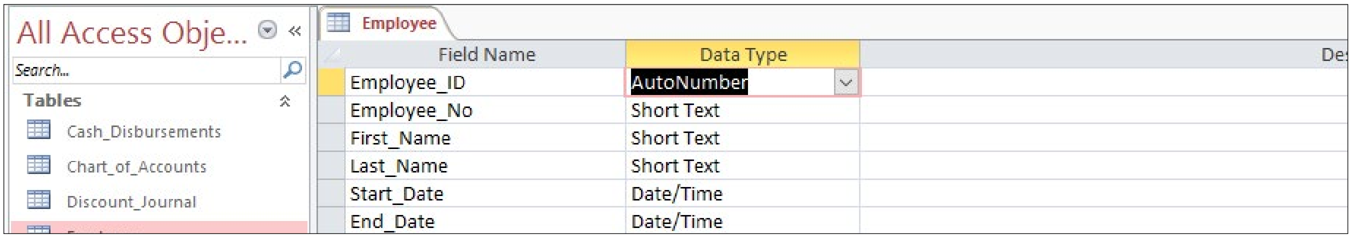
The ***Indexed*** property should be used with caution. It is used to speed up performance when querying a large table. However, if the index is not set on the right fields, it can severely slow down performance when updating, adding or deleting records from the table. Once you get to a point where performance is an issue, then you can research more about this property, but an in-depth discussion of indexing is outside the scope of this course.

Primary Key

Every table should have at least one field (or combination of fields) that contains a unique value for each record. Such a field is called a ***Primary Key***. You can set a Primary Key in the Table Design View. This is good database design practice, and I would encourage you to ALWAYS have a primary key field in each of your tables. In Access, it's very easy to set up a table with unique records using the AutoNumber feature in Table Design. In my tables, I like to create a field called the name of the table followed by "_ID", design that field as an AutoNumber, and use that field as the Primary Key. Even if you think you may not need it, trust me, you will.

In the Employee table, you will create an AutoNumber field called Employee_ID. Whenever records are added to the table, a new Employee_ID number is automatically populated with the next number in the sequence. If a record is deleted, the AutoNumber is permanently deleted and can never be re-used. As its name implies, this field is a number field, meaning that the data in this field is restricted only to numbers. So if you tried to input "123XYZ" into this field, you would get an error message, as the field type is a number. AutoNumber values are automatically generated, and you can't input a number in that field even if you tried. Let's create the AutoNumber field now.

17. *With* **Employee_No** *field selected, click the* **Insert Rows** *icon* Insert Rows *in the* **Tools** *group of the* **Design** *tab.*
18. *Under* **Field Name**, *type* ***Employee_ID*** *and press* **[Enter]**.
19. *Under* **Data Type**, *click on the drop-down arrow and choose* **AutoNumber**.



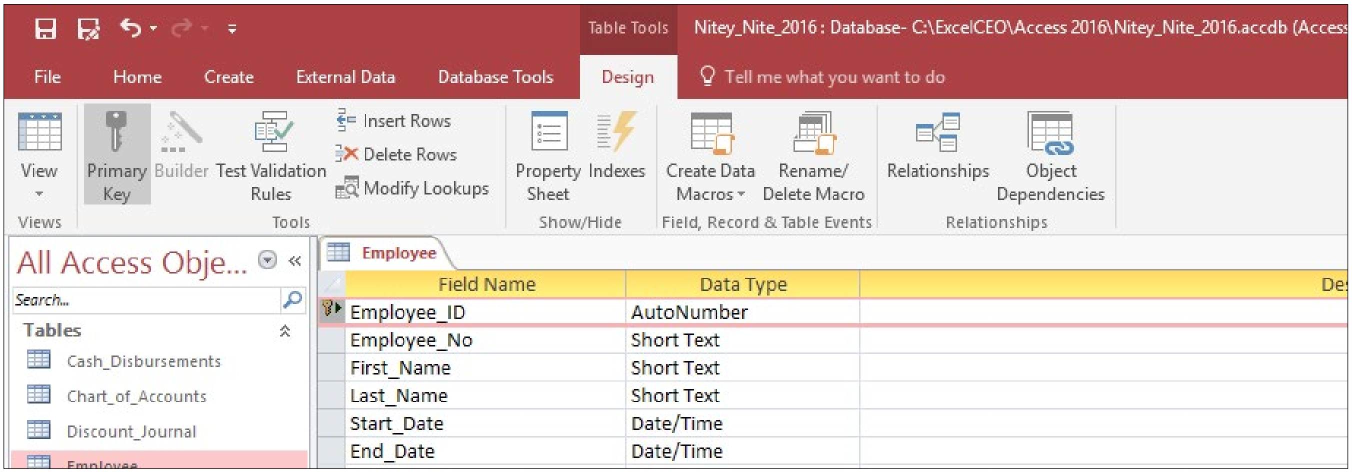
Field Name	Data Type	Des
Employee_ID	AutoNumber	
Employee_No	Short Text	
First_Name	Short Text	
Last_Name	Short Text	
Start_Date	Date/Time	
End_Date	Date/Time	

Figure 1.15

When you have a field that you want to use as a primary key, you have to tell Access that the field is the primary key. You do that by selecting that field in Design View and clicking on the Primary Key icon.

20. *With your cursor on the* **Employee_ID** *field, click the* **Primary Key** *icon* .

A small key symbol appears to the left of the Employee_ID field, indicating that field is being used as a Primary Key. The Field Properties section of Table Design View should look like this:



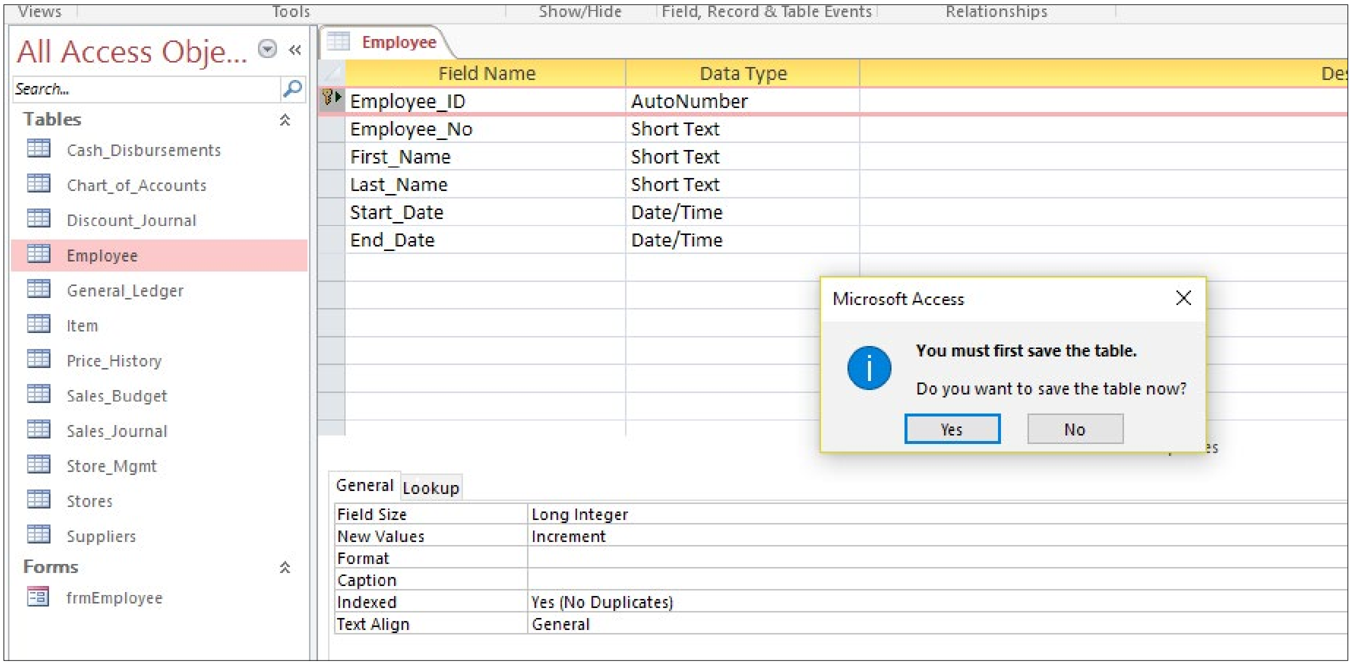
Employee

Field Name	Data Type	Des
Employee_ID	AutoNumber	
Employee_No	Short Text	
First_Name	Short Text	
Last_Name	Short Text	
Start_Date	Date/Time	
End_Date	Date/Time	

Figure 1.16

The Indexed property of Yes (No Duplicates) means there is an index on the field and that no two records can have the same value.

21. *Click on the* **View** *icon* *(Be sure to click on the image, not the down arrow).*



Field Name	Data Type	Des
Employee_ID	AutoNumber	
Employee_No	Short Text	
First_Name	Short Text	
Last_Name	Short Text	
Start_Date	Date/Time	
End_Date	Date/Time	

Microsoft Access

You must first save the table.

Do you want to save the table now?

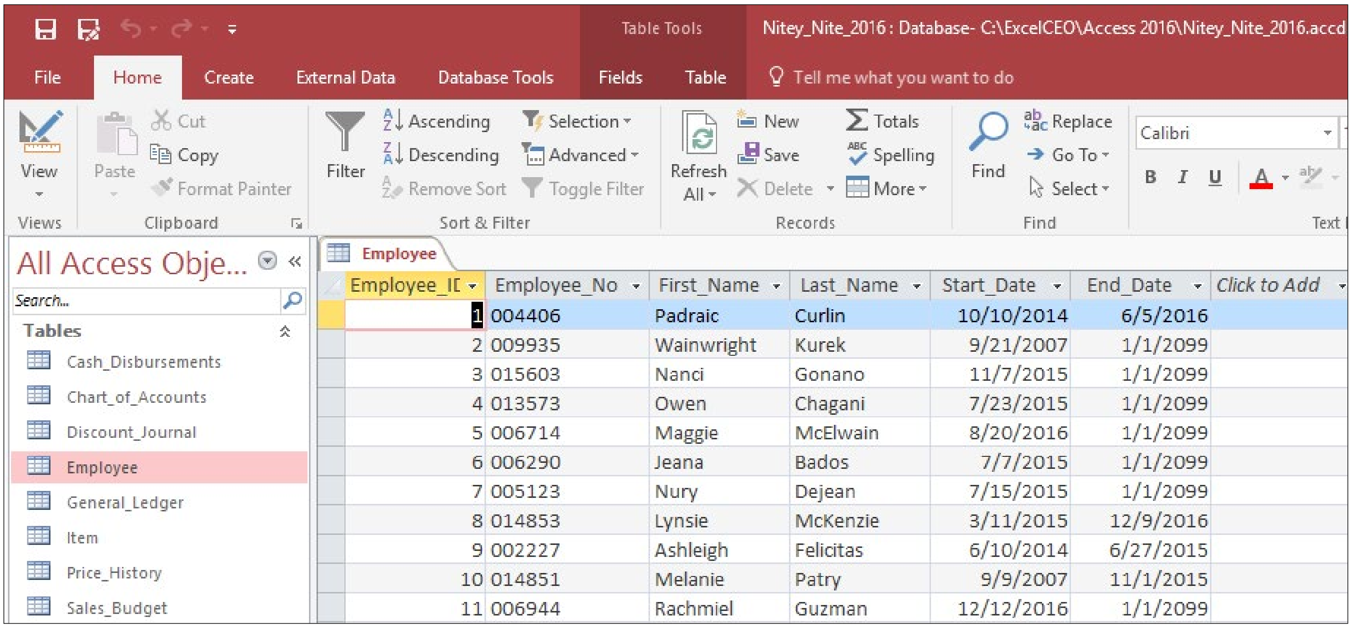
Field Size	Long Integer
New Values	Increment
Format	
Caption	
Indexed	Yes (No Duplicates)
Text Align	General

Figure 1.17

In order to view the data after you've made changes to the design of the table, you must first save it.

> ***IMPORTANT NOTE****: When you "save" a table, it saves the design of the table, such as field width adjustments. Whenever you enter data into an Access table, the data is automatically saved after you move to another row. Therefore, it is not necessary to "save" data after you enter it. You save only when there is a change to the design of the table.*

22. *Click* **Yes** *to save the table design.*



Employee_ID	Employee_No	First_Name	Last_Name	Start_Date	End_Date	Click to Add
1	004406	Padraic	Curlin	10/10/2014	6/5/2016	
2	009935	Wainwright	Kurek	9/21/2007	1/1/2099	
3	015603	Nanci	Gonano	11/7/2015	1/1/2099	
4	013573	Owen	Chagani	7/23/2015	1/1/2099	
5	006714	Maggie	McElwain	8/20/2016	1/1/2099	
6	006290	Jeana	Bados	7/7/2015	1/1/2099	
7	005123	Nury	Dejean	7/15/2015	1/1/2099	
8	014853	Lynsie	McKenzie	3/11/2015	12/9/2016	
9	002227	Ashleigh	Felicitas	6/10/2014	6/27/2015	
10	014851	Melanie	Patry	9/9/2007	11/1/2015	
11	006944	Rachmiel	Guzman	12/12/2016	1/1/2099	

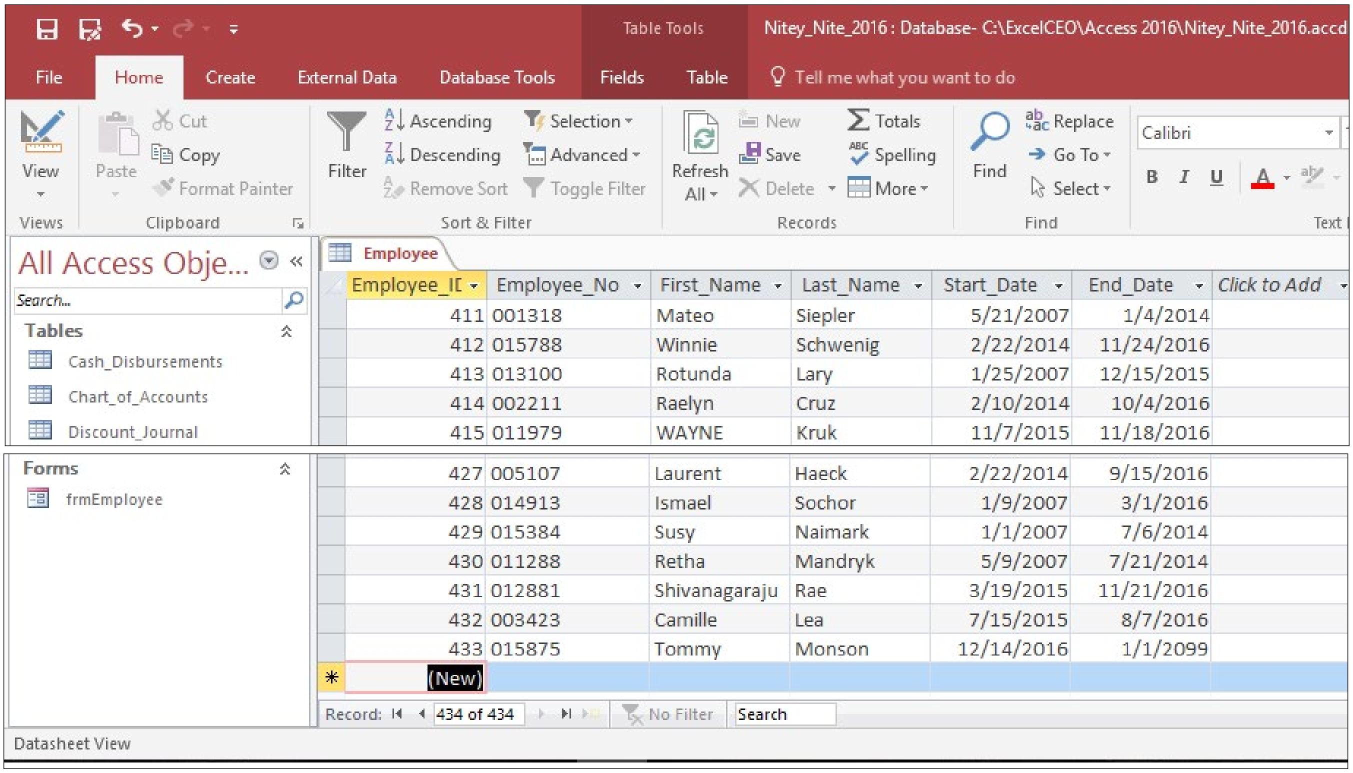
Figure 1.18

Notice that when you return to the Datasheet view of the table, the filter has been removed. By default, Access 2016 creates the “Click to Add” column that allows you to create a new field within the Datasheet view of the table, so you don’t have to always go back to the Design view to create a field. I don’t prefer this “enhancement” much, as I like to define everything about the field before it appears in Datasheet view, so we won’t do anything with that field.

Add a Record

Now you will add a record to the table.

1. *Click the* **New (blank) record** *icon* ▶✳ *in the* **Record Selector** *section to add a new record.*
2. *In the* **Employee_No** *field, type:* **015875** *and press* **[Enter]**.
3. *In the* **First_Name** *field, type your own first name and press the* **[Tab]** *key.*
4. *In the* **Last_Name** *field, type your own last name.*
5. *Click in the* **Start_Date** *field and type:* **12/14/2016**.
6. *In the* **End_Date** *field, type:* **1/1/2099** *and press* **[Enter]**.



Employee_II	Employee_No	First_Name	Last_Name	Start_Date	End_Date	Click to Add
411	001318	Mateo	Siepler	5/21/2007	1/4/2014	
412	015788	Winnie	Schwenig	2/22/2014	11/24/2016	
413	013100	Rotunda	Lary	1/25/2007	12/15/2015	
414	002211	Raelyn	Cruz	2/10/2014	10/4/2016	
415	011979	WAYNE	Kruk	11/7/2015	11/18/2016	
427	005107	Laurent	Haeck	2/22/2014	9/15/2016	
428	014913	Ismael	Sochor	1/9/2007	3/1/2016	
429	015384	Susy	Naimark	1/1/2007	7/6/2014	
430	011288	Retha	Mandryk	5/9/2007	7/21/2014	
431	012881	Shivanagaraju	Rae	3/19/2015	11/21/2016	
432	003423	Camille	Lea	7/15/2015	8/7/2016	
433	015875	Tommy	Monson	12/14/2016	1/1/2099	
(New)						

Record: 434 of 434 No Filter Search

Figure 1.19

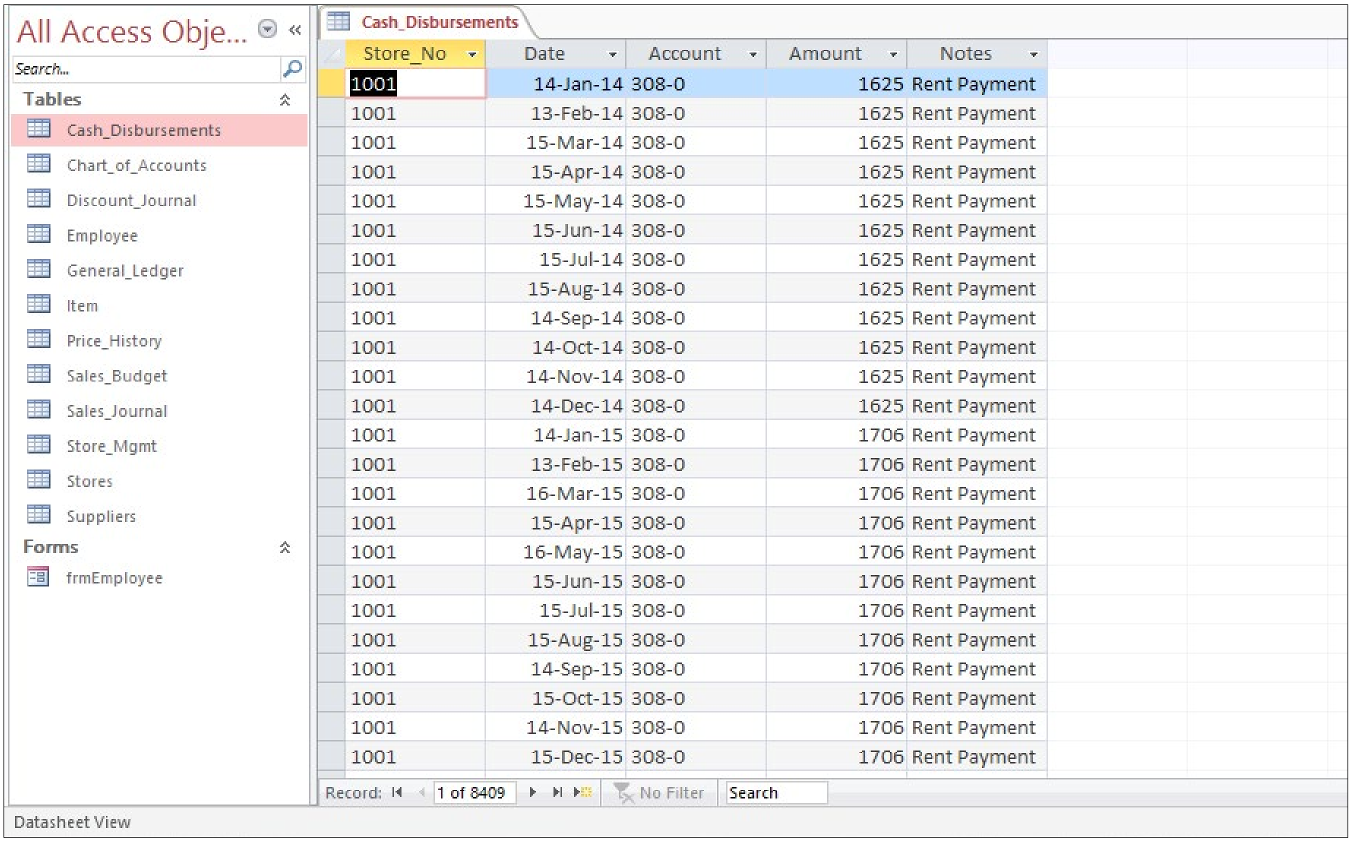
Congratulations! You are now the newest employee of Nitey-Nite Mattresses. Did you see how the AutoNumber was added when you started to type the Employee_No? When you got to the end of the record and pressed [Enter], your cursor moved to the next record (as if you were going to input another record), the record you input automatically saved, and the AutoNumber was automatically created.

We're now finished with the Employee table so you can close it.

7. *Click the* **Close Window** *icon* × *in the right section of the screen at the same level as the* **Employee** *tab.*

Let's briefly review some of the other tables in the database, as you will be using them throughout this course.

1. *Double-click on the* **Cash_Disbursements** *table (to open it).*

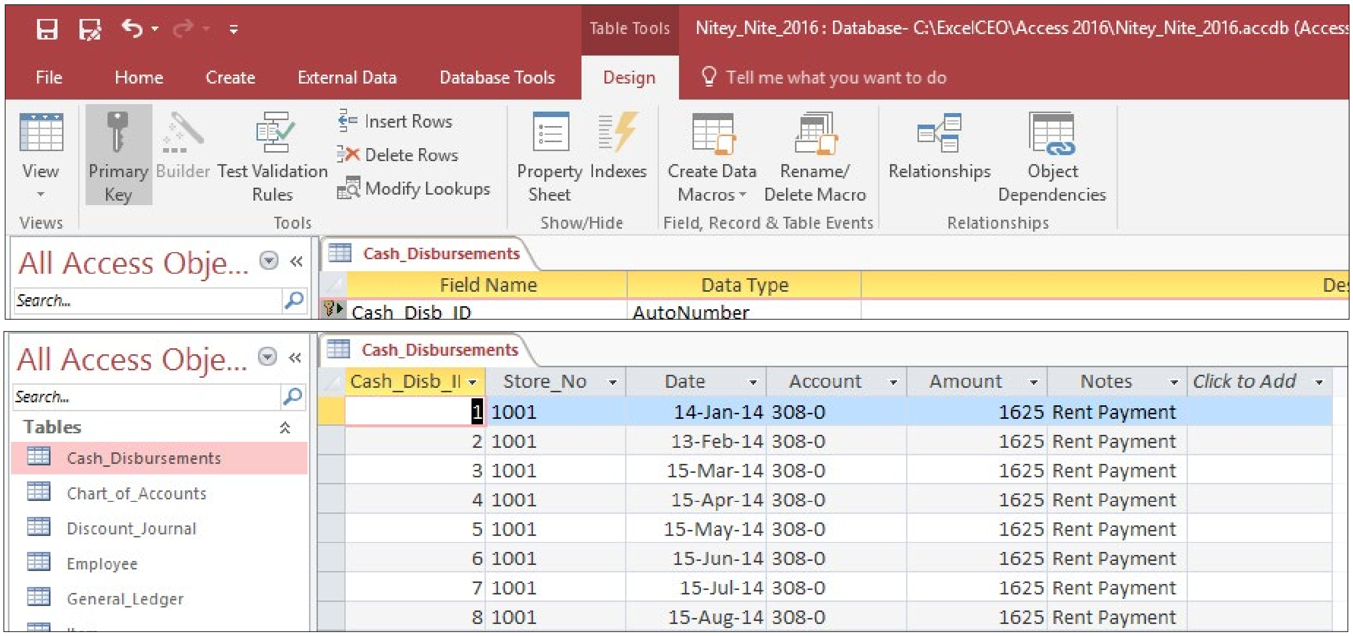


Store_No	Date	Account	Amount	Notes
1001	14-Jan-14	308-0	1625	Rent Payment
1001	13-Feb-14	308-0	1625	Rent Payment
1001	15-Mar-14	308-0	1625	Rent Payment
1001	15-Apr-14	308-0	1625	Rent Payment
1001	15-May-14	308-0	1625	Rent Payment
1001	15-Jun-14	308-0	1625	Rent Payment
1001	15-Jul-14	308-0	1625	Rent Payment
1001	15-Aug-14	308-0	1625	Rent Payment
1001	14-Sep-14	308-0	1625	Rent Payment
1001	14-Oct-14	308-0	1625	Rent Payment
1001	14-Nov-14	308-0	1625	Rent Payment
1001	14-Dec-14	308-0	1625	Rent Payment
1001	14-Jan-15	308-0	1706	Rent Payment
1001	13-Feb-15	308-0	1706	Rent Payment
1001	16-Mar-15	308-0	1706	Rent Payment
1001	15-Apr-15	308-0	1706	Rent Payment
1001	16-May-15	308-0	1706	Rent Payment
1001	15-Jun-15	308-0	1706	Rent Payment
1001	15-Jul-15	308-0	1706	Rent Payment
1001	15-Aug-15	308-0	1706	Rent Payment
1001	14-Sep-15	308-0	1706	Rent Payment
1001	15-Oct-15	308-0	1706	Rent Payment
1001	14-Nov-15	308-0	1706	Rent Payment
1001	15-Dec-15	308-0	1706	Rent Payment

Figure 1.20

This table is a journal of cash disbursements. Think of it as a checkbook register. It does not have a primary key (naughty, naughty, Nitey-Nite). Remember, the Primary Key is a field that contains a unique value for each record.

2. *Create a* **Primary Key** *in the* **Cash_Disbursements** *table using an* **AutoNumber** *format and name the field* ***Cash_Disb_ID****.*



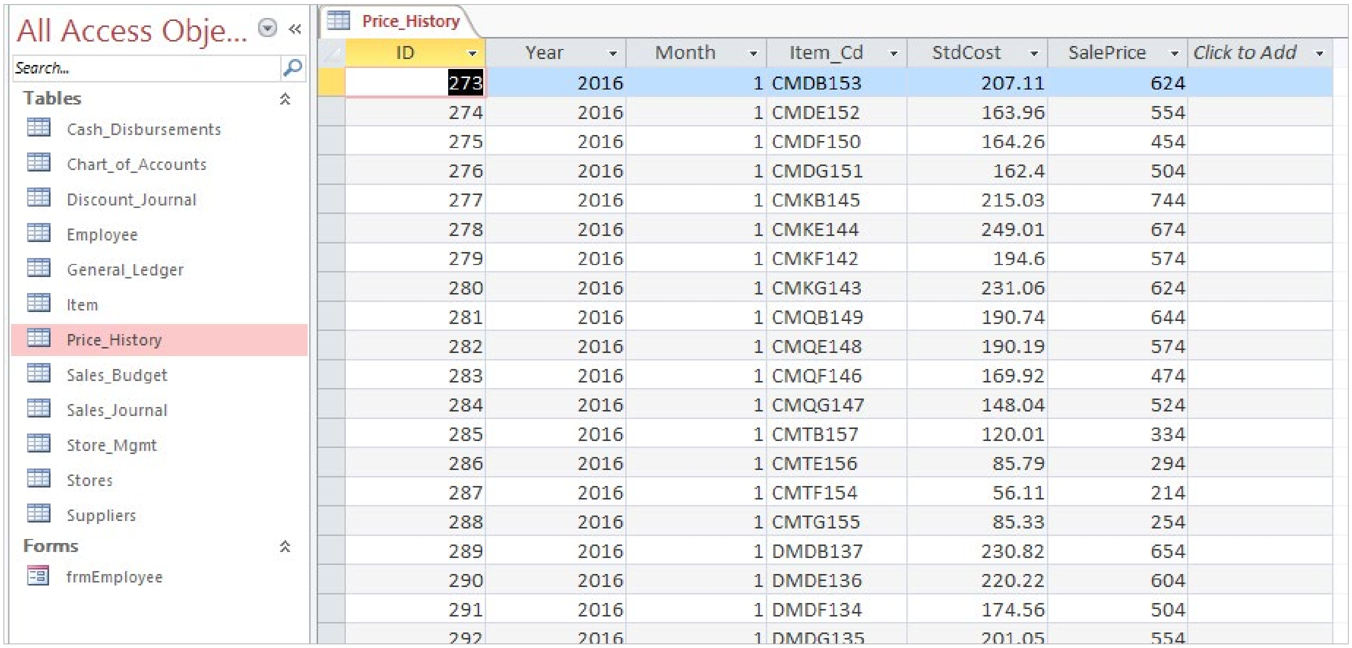
Field Name	Data Type	De
Cash_Disb_ID	AutoNumber	

Cash_Disb_I	Store_No	Date	Account	Amount	Notes	Click to Add
1	1001	14-Jan-14	308-0	1625	Rent Payment	
2	1001	13-Feb-14	308-0	1625	Rent Payment	
3	1001	15-Mar-14	308-0	1625	Rent Payment	
4	1001	15-Apr-14	308-0	1625	Rent Payment	
5	1001	15-May-14	308-0	1625	Rent Payment	
6	1001	15-Jun-14	308-0	1625	Rent Payment	
7	1001	15-Jul-14	308-0	1625	Rent Payment	
8	1001	15-Aug-14	308-0	1625	Rent Payment	

Figure 1.21

As you scroll down through the data in this table, you will see many different types of expenditures, like Rent, Utility, Office Supplies, and others. The payments in this table have already been uploaded to the General_Ledger table, so you could use this table to reconcile back to the General_Ledger table. The Cash_Disbursements table contains 8,409 records.

3. **Save** *and close the* **Cash_Disbursements** *table, then open the* **Price_History** *table.*



ID	Year	Month	Item_Cd	StdCost	SalePrice	Click to Add
273	2016	1	CMDB153	207.11	624	
274	2016	1	CMDE152	163.96	554	
275	2016	1	CMDF150	164.26	454	
276	2016	1	CMDG151	162.4	504	
277	2016	1	CMKB145	215.03	744	
278	2016	1	CMKE144	249.01	674	
279	2016	1	CMKF142	194.6	574	
280	2016	1	CMKG143	231.06	624	
281	2016	1	CMQB149	190.74	644	
282	2016	1	CMQE148	190.19	574	
283	2016	1	CMQF146	169.92	474	
284	2016	1	CMQG147	148.04	524	
285	2016	1	CMTB157	120.01	334	
286	2016	1	CMTE156	85.79	294	
287	2016	1	CMTF154	56.11	214	
288	2016	1	CMTG155	85.33	254	
289	2016	1	DMDB137	230.82	654	
290	2016	1	DMDE136	220.22	604	
291	2016	1	DMDF134	174.56	504	
292	2016	1	DMDG135	201.05	554	

Figure 1.22

This table is a historical listing of the standard cost (used in the calculation for cost of merchandise) and the retail sale price, or the price per item used in all stores. The costs and sale prices of all of Nitey-Nite's items change every year, and this table makes a reconciliation of cost and sale prices very easy to do. This table has an ID field, but it is not yet set as the Primary Key. The Item_Cd field in this table is a unique identifier for each item Nitey-Nite sells (mattresses and pillows), but since one item can be repeated several times in this table, it should not be used as a Primary Key. The Item_Cd itself doesn't tell you much about the item unless you know how the Item_Cd is derived. We'll see that data in the Item table. For now, let's set the ID field in the Price_History table as the Primary Key.

4. *Set the* **ID** *field in the* **Price_History** *table as the* **Primary Key**.

Note that the ID field is an AutoNumber field, but it starts with the number 273. That means that the records before 273 (1 – 272) were deleted from the table at some time in the past.

5. **Save** *and close the* **Price_History** *table.*

We will review the other tables as necessary as we continue through the course.

> ***Review Questions****: It is now time to complete the hands-on Review Questions. Log on to www.ExcelCEO.com with your Email and Password, click on the* **Access 2016 and SQL Review Questions, Chapter 1, Section 2 of 2** *option and complete the review questions.*

Conclusion

In this chapter, you were introduced to Access. You learned about the enhancements that Microsoft developed in Access 2016. We reviewed the different types of databases, how to create a new database, and how to open and navigate around an existing database. Tables are the most basic objects in Access and you learned to open and view tables, create a new field, and input a new record. You learned about data types and field properties in a table's design. You learned how to filter and sort a table, and you saw how to create a new field and create a Primary Key. In the next chapter, we'll begin exploring my favorite function of Access — Queries.

Chapter Exam

You can now go to www.ExcelCEO.com, click on Sign In, log in and take the exam. Make sure that you take the exam on the same device on which you completed the practice files, as some of the questions on the exam may refer to some of the completed projects. Chapter exams are intended to be hands-on.

Access® 2016 and SQL

Complete Self-study Course

ExcelCEO

Chief Excel Officer

CHAPTER TWO — BEGINNING QUERIES

In this chapter, you will:

- Identify the correct way to create a simple query using the Query Wizard and from Design View
- Recognize the various ways to add and delete fields from the query Design Grid
- Choose the appropriate criteria to add to a query
- Select the correct formula format in the Criteria section of a query
- Identify the times to use the "OR" criteria in a query

CPE Credits possible for this chapter: 2

Introduction to Queries

If there is any one concept in this book that I want you to learn inside and out, backwards and forwards, up and down, it is Queries. ***Queries*** will become the basis for all of your database development, so it is absolutely essential that you know this material very well. In my opinion, queries are the most important task in performing financial analysis with Access. Queries are very powerful tools. Getting the right data is 80% of your job, and once you have the right data, slicing and dicing it in Excel, Access, SQL Server, putting it on the web, etc., is the easy part. Knowing how to query is essential in securing the data behind your reports. First, let's talk about some of the basics of a query.

Create a Simple Query

What is a query? A ***query*** is a virtual table. It doesn't contain any data in itself, but it is simply the procedure to pull data from one or more tables or other queries. You design a query to get data from one or more sources, and that source doesn't have to be an Access database. You can pull data from an Excel spreadsheet, an Oracle or Paradox database, SQL Server database, .txt and .csv files, and a host of other ***data sources***. When executed, the query pulls and displays the data for you to view and manipulate. A query can be both very simple and extremely complex, depending on how you set it up. Let's create a simple query to show you what it can do.

1. *In the* **Nitey_Nite_2016** *database, click on the* **Create** *tab.*

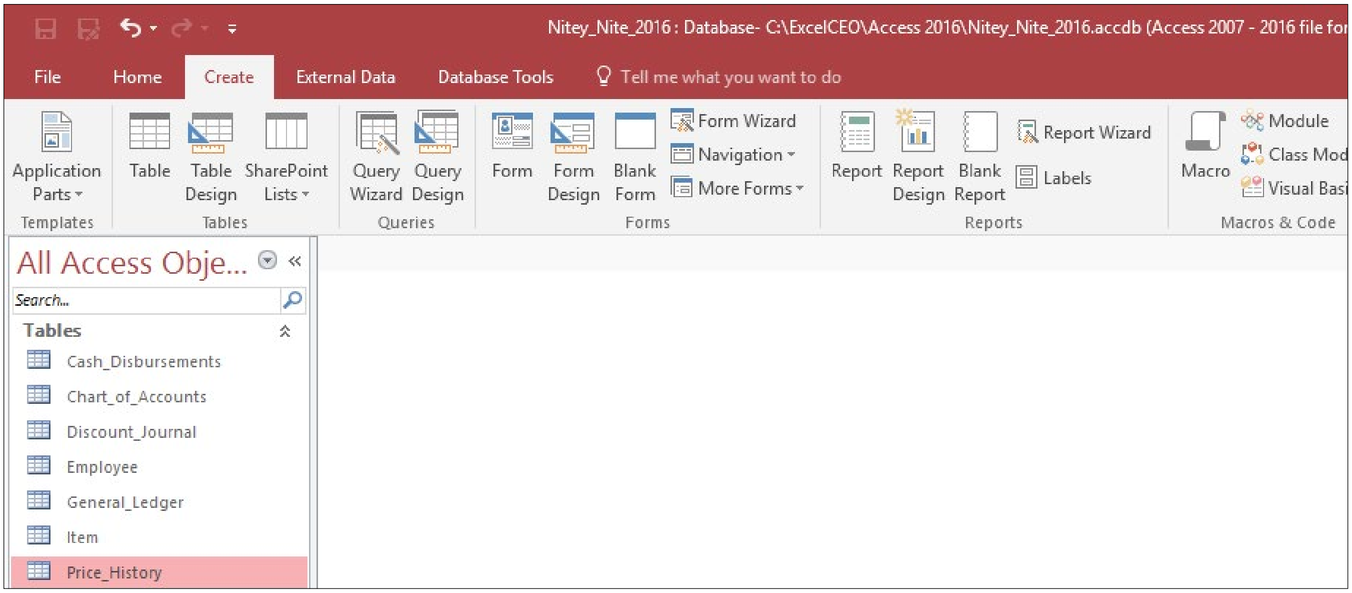


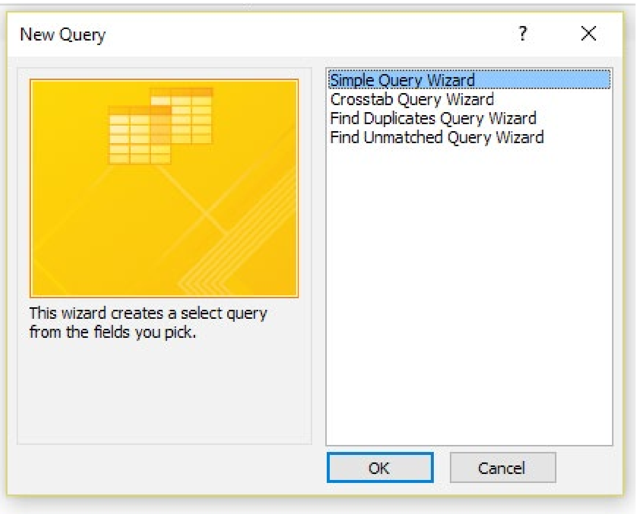
Figure 2.1

In the ***Create*** tab, you can create templates, tables, queries, forms, and reports, and there are many tools to help you do so. There are no existing queries in the Nitey_Nite_2016 database, so your basic options are to create a query from Design View (by clicking the Query Design icon) or by using the Query Wizard. In creating your first query, you will use the Query Wizard. For all others, you will use Query Design.

Using the Query Wizard

Let's walk through a simple example of how the wizard can help you in designing queries. Suppose your manager asks you about one of your vendors, Sleepwell. He wants to see details about all the products that Sleepwell offers. That data is contained in the Item table. You will now use the ***Query Wizard*** to design a query to extract all products from the Item table.

2. *Click on the* **Query Wizard** *icon in the* **Queries** *group of the* **Create** *tab.*



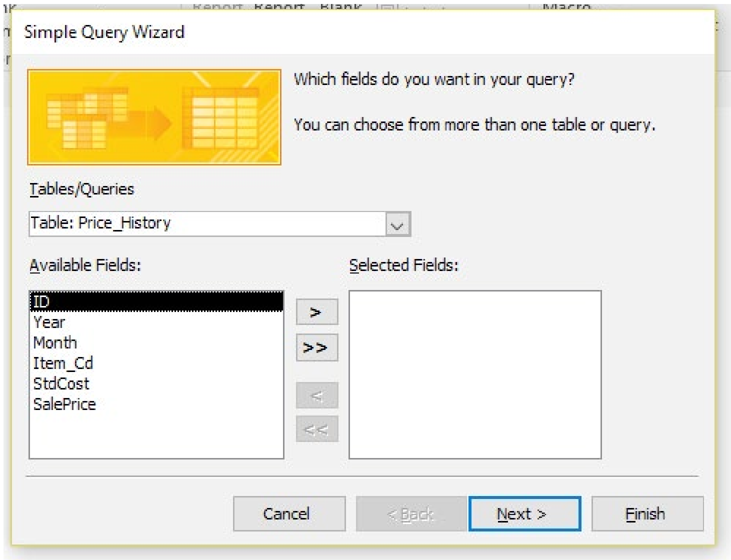
This wizard creates a select query from the fields you pick.

Figure 2.2

The first screen of the wizard asks you to choose the type of query you want to create. In this example, we'll create a simple query. We'll create other query types later. The Find Duplicates and Find Unmatched queries help you to create queries for those specific purposes.

3. *With* **Simple Query Wizard** *selected, click* **OK**.

You may get a security notice dialog box advising you that the query may not come from a reliable source. If you get this message, click Open to continue with the Query Wizard.



Simple Query Wizard

Which fields do you want in your query?

You can choose from more than one table or query.

Tables/Queries

Table: Price_History

Available Fields:

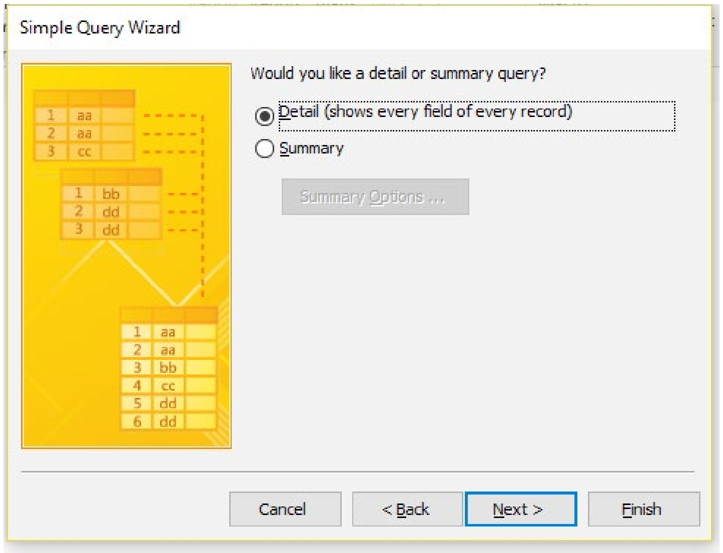
ID
Year
Month
Item_Cd
StdCost
SalePrice

Selected Fields:

Figure 2.3

In the next step of the Query Wizard, you will choose the table or query you want to work with, and the fields within the table or query.

4. *Click on the drop-down menu under* **Tables/Queries** *and choose* **Table: Item**.
5. *Click on the* **>>** *button to move all fields into the* **Selected Fields:** *section.*
6. *Click* **Next >**.



Simple Query Wizard

Would you like a detail or summary query?

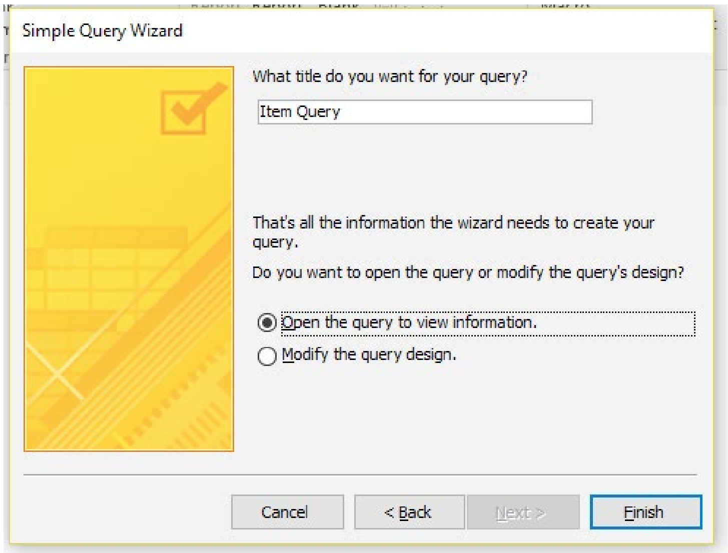
● Detail (shows every field of every record)

○ Summary

Figure 2.4

The next screen asks if you want to show the detailed records or if you want a summary of the data. We want the default option, Detail.

7. *Click* **Next >**.



Simple Query Wizard

What title do you want for your query?

Item Query

That's all the information the wizard needs to create your query.

Do you want to open the query or modify the query's design?

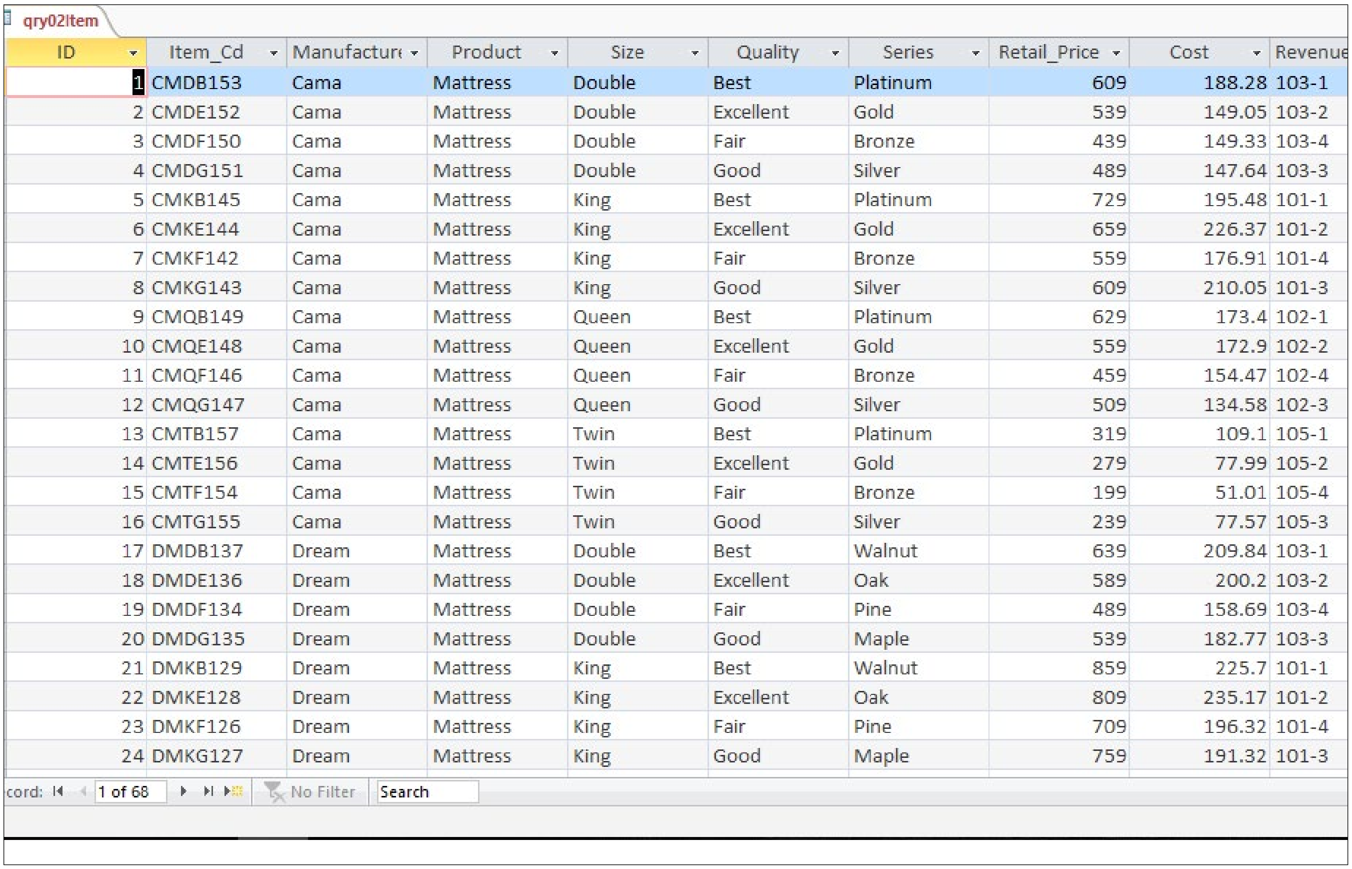
● Open the query to view information.

○ Modify the query design.

Figure 2.5

The last step asks you to name your query, and saves it when finished. Sometimes, you'll need to save the query so you can use it in another analysis, or use it at a later date. When you save a query, you must give it a name. The name should be logical. For example, we could name this query Sleepwell_Items or something like that. However, in Access you sometimes can't tell if an object is a table or query unless it is specified in the name, or if you just happen to know. To solve that problem, I like to begin my query names with "qry". That lets me know it is a query. I talked about this in the Object Naming Conventions section of Chapter One. It may not seem important now, but I promise you that using this or a similar naming convention will save you lots of headaches in the future, so I encourage you to do it. From here on out, I will save all queries with a name beginning with qry, followed by the chapter number it came from, and then a description of the data it is extracting.

8. *Name the query* ***qry02Item*** *and click* **Finish**.



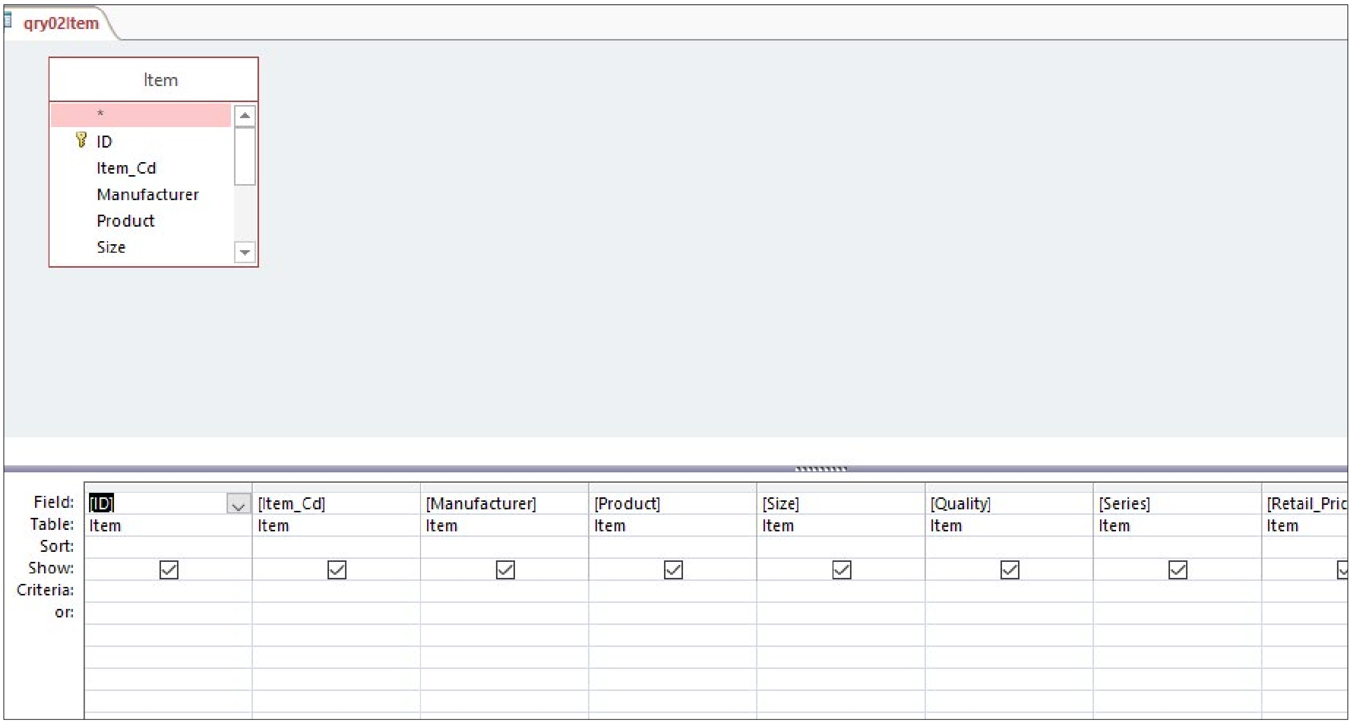
qry02Item

ID	Item_Cd	Manufacture	Product	Size	Quality	Series	Retail_Price	Cost	Revenue
1	CMDB153	Cama	Mattress	Double	Best	Platinum	609	188.28	103-1
2	CMDE152	Cama	Mattress	Double	Excellent	Gold	539	149.05	103-2
3	CMDF150	Cama	Mattress	Double	Fair	Bronze	439	149.33	103-4
4	CMDG151	Cama	Mattress	Double	Good	Silver	489	147.64	103-3
5	CMKB145	Cama	Mattress	King	Best	Platinum	729	195.48	101-1
6	CMKE144	Cama	Mattress	King	Excellent	Gold	659	226.37	101-2
7	CMKF142	Cama	Mattress	King	Fair	Bronze	559	176.91	101-4
8	CMKG143	Cama	Mattress	King	Good	Silver	609	210.05	101-3
9	CMQB149	Cama	Mattress	Queen	Best	Platinum	629	173.4	102-1
10	CMQE148	Cama	Mattress	Queen	Excellent	Gold	559	172.9	102-2
11	CMQF146	Cama	Mattress	Queen	Fair	Bronze	459	154.47	102-4
12	CMQG147	Cama	Mattress	Queen	Good	Silver	509	134.58	102-3
13	CMTB157	Cama	Mattress	Twin	Best	Platinum	319	109.1	105-1
14	CMTE156	Cama	Mattress	Twin	Excellent	Gold	279	77.99	105-2
15	CMTF154	Cama	Mattress	Twin	Fair	Bronze	199	51.01	105-4
16	CMTG155	Cama	Mattress	Twin	Good	Silver	239	77.57	105-3
17	DMDB137	Dream	Mattress	Double	Best	Walnut	639	209.84	103-1
18	DMDE136	Dream	Mattress	Double	Excellent	Oak	589	200.2	103-2
19	DMDF134	Dream	Mattress	Double	Fair	Pine	489	158.69	103-4
20	DMDG135	Dream	Mattress	Double	Good	Maple	539	182.77	103-3
21	DMKB129	Dream	Mattress	King	Best	Walnut	859	225.7	101-1
22	DMKE128	Dream	Mattress	King	Excellent	Oak	809	235.17	101-2
23	DMKF126	Dream	Mattress	King	Fair	Pine	709	196.32	101-4
24	DMKG127	Dream	Mattress	King	Good	Maple	759	191.32	101-3

Figure 2.6

The query is saved and opens up to show all of the records in the Item table. This view of the data is called the ***Datasheet View***. I will also refer to this view as the record set. But didn't we want the records just for Sleepwell? To do that, we have to modify the design of the query.

9. *Click on the* **View** *icon in the* **Views** *group of the* **Home** *tab (make sure you click on the graphic and not the down arrow).*

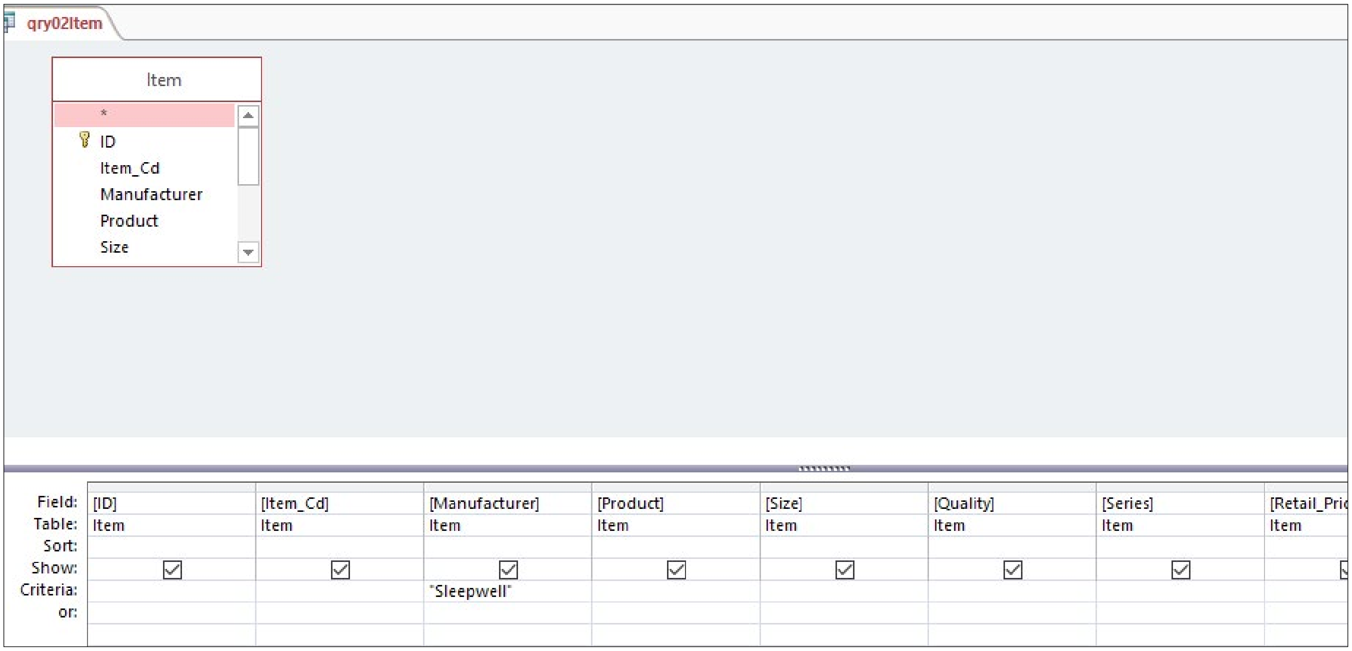


Field:	[ID]	[Item_Cd]	[Manufacturer]	[Product]	[Size]	[Quality]	[Series]	[Retail_Pric
Table:	Item	Item	Item	Item	Item	Item	Item	Item
Sort:								
Show:	☑	☑	☑	☑	☑	☑	☑	☑
Criteria:								
or:								

Figure 2.7

The Design View of the query appears. Notice that the field names appear in brackets. In Access 2016, all field names appear in brackets. For now we just want to filter the query for all Sleepwell products.

10. *Under* [**Manufacturer**] *on the* **Criteria** *line, type* ***Sleepwell*** *and press* [**Enter**].

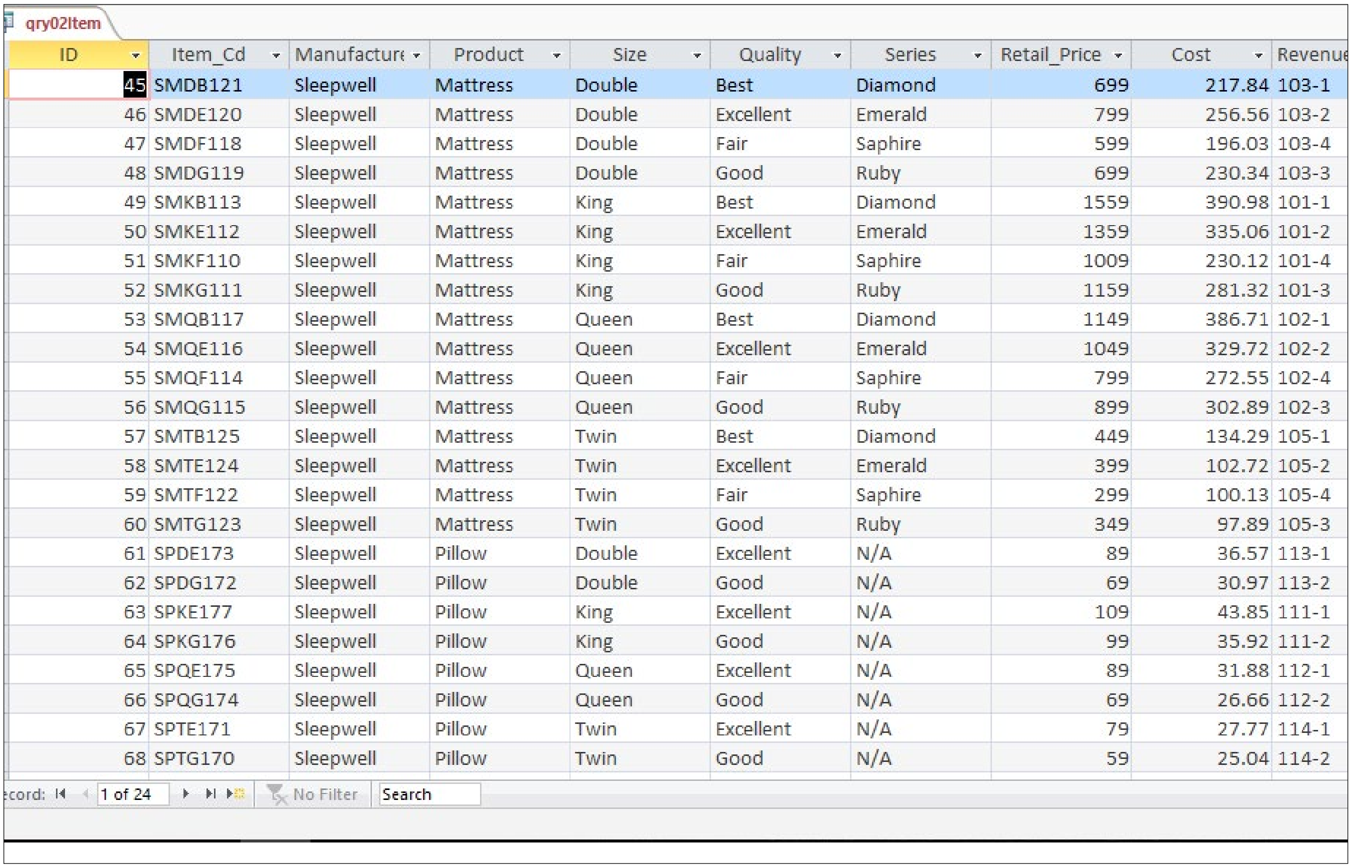


Field:	[ID]	[Item_Cd]	[Manufacturer]	[Product]	[Size]	[Quality]	[Series]	[Retail_Pric
Table:	Item	Item	Item	Item	Item	Item	Item	Item
Sort:								
Show:	☑	☑	☑	☑	☑	☑	☑	☑
Criteria:			"Sleepwell"					
or:								

Figure 2.8

As you enter data on the Criteria line and press [Enter], Access may enclose the typed criteria with certain characters. If the field is a text field, Access will assume the criteria is also text, and will enclose the criteria with quotes (“”). Dates are enclosed with pound signs (#), and number criteria are not enclosed with any character. In this case, the Manufacturer field is a text field, so Access enclosed the word “Sleepwell” with quotes.

11. *To display the query in* **Datasheet View**, *click the* **View** *icon in the* **Results** *group.*



ID	Item_Cd	Manufacture	Product	Size	Quality	Series	Retail_Price	Cost	Revenue
45	SMDB121	Sleepwell	Mattress	Double	Best	Diamond	699	217.84	103-1
46	SMDE120	Sleepwell	Mattress	Double	Excellent	Emerald	799	256.56	103-2
47	SMDF118	Sleepwell	Mattress	Double	Fair	Saphire	599	196.03	103-4
48	SMDG119	Sleepwell	Mattress	Double	Good	Ruby	699	230.34	103-3
49	SMKB113	Sleepwell	Mattress	King	Best	Diamond	1559	390.98	101-1
50	SMKE112	Sleepwell	Mattress	King	Excellent	Emerald	1359	335.06	101-2
51	SMKF110	Sleepwell	Mattress	King	Fair	Saphire	1009	230.12	101-4
52	SMKG111	Sleepwell	Mattress	King	Good	Ruby	1159	281.32	101-3
53	SMQB117	Sleepwell	Mattress	Queen	Best	Diamond	1149	386.71	102-1
54	SMQE116	Sleepwell	Mattress	Queen	Excellent	Emerald	1049	329.72	102-2
55	SMQF114	Sleepwell	Mattress	Queen	Fair	Saphire	799	272.55	102-4
56	SMQG115	Sleepwell	Mattress	Queen	Good	Ruby	899	302.89	102-3
57	SMTB125	Sleepwell	Mattress	Twin	Best	Diamond	449	134.29	105-1
58	SMTE124	Sleepwell	Mattress	Twin	Excellent	Emerald	399	102.72	105-2
59	SMTF122	Sleepwell	Mattress	Twin	Fair	Saphire	299	100.13	105-4
60	SMTG123	Sleepwell	Mattress	Twin	Good	Ruby	349	97.89	105-3
61	SPDE173	Sleepwell	Pillow	Double	Excellent	N/A	89	36.57	113-1
62	SPDG172	Sleepwell	Pillow	Double	Good	N/A	69	30.97	113-2
63	SPKE177	Sleepwell	Pillow	King	Excellent	N/A	109	43.85	111-1
64	SPKG176	Sleepwell	Pillow	King	Good	N/A	99	35.92	111-2
65	SPQE175	Sleepwell	Pillow	Queen	Excellent	N/A	89	31.88	112-1
66	SPQG174	Sleepwell	Pillow	Queen	Good	N/A	69	26.66	112-2
67	SPTE171	Sleepwell	Pillow	Twin	Excellent	N/A	79	27.77	114-1
68	SPTG170	Sleepwell	Pillow	Twin	Good	N/A	59	25.04	114-2

Figure 2.9

There are two icons you can use to run an Access query while in Design View — the View icon and the Run icon. If you are running a simple query like we’ve built (called a Select query), you can use either icon. If you are in an action query (discussed in Chapter Five), the View icon will view the results, but will not perform the action. The Run icon will execute the action.

You now have a list of 24 records which represent all of the products that Sleepwell provides to Nitey-Nite. This is an example of a very simple query, and you now know how the Query Wizard works. When you save the query, you will see its name appear under Queries in the All Access Objects section.

12. **Save** *and close the* **qry02Item** *query.*

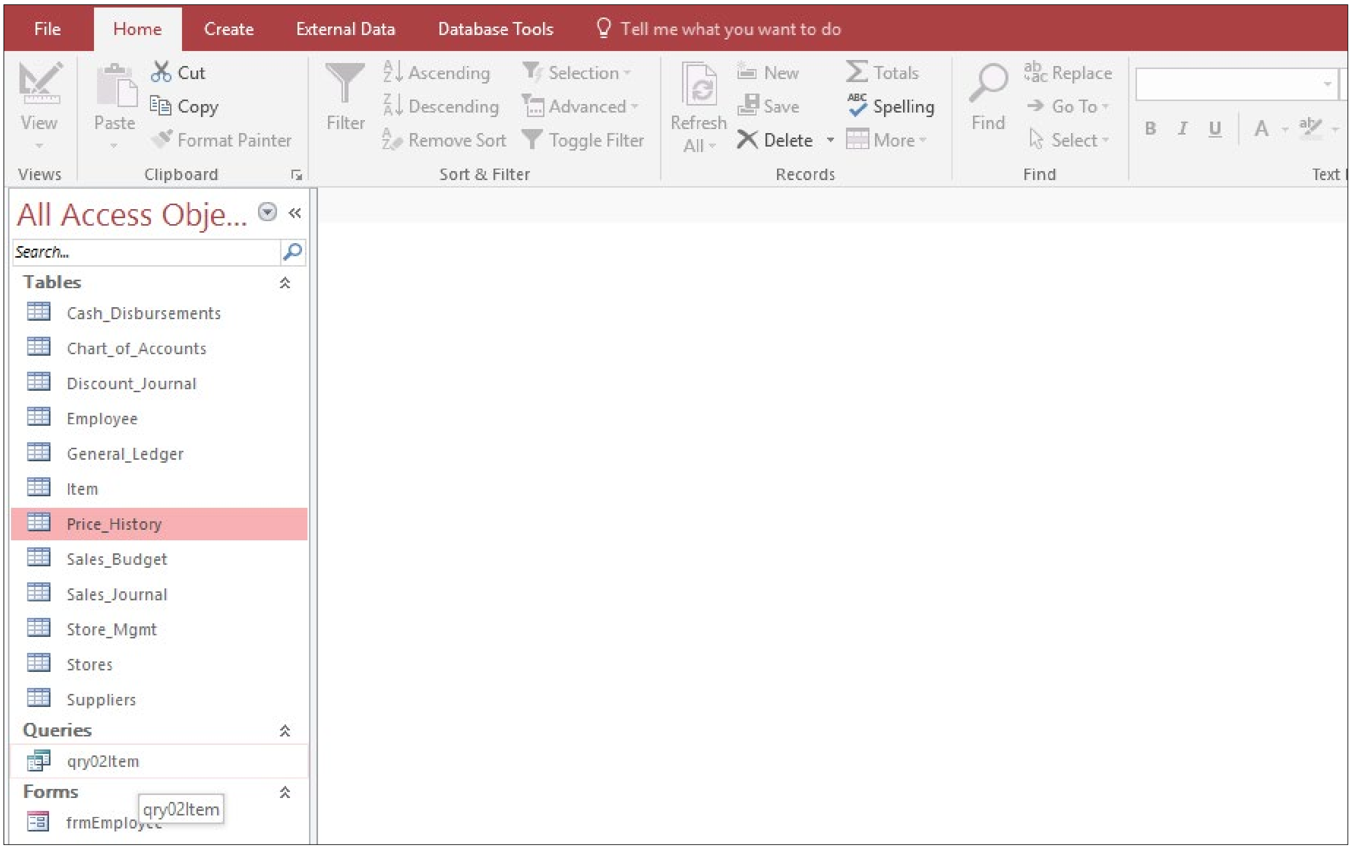


Figure 2.10

You can now see the qry02Item query in the new Queries section appear in the Navigation Pane.

> ***Review Questions****: It is now time to complete the hands-on Review Questions. Log on to www.ExcelCEO.com with your Email and Password, click on the* **Access 2016 and SQL Review Questions, Chapter 2, Section 1 of 2** *option and complete the review questions.*

Query Design View

Now you will create a new query by going directly into the Design View.

1. *In the* **Queries** *group of the* **Create** *tab, click on the* **Query Design** *icon* Query Design .

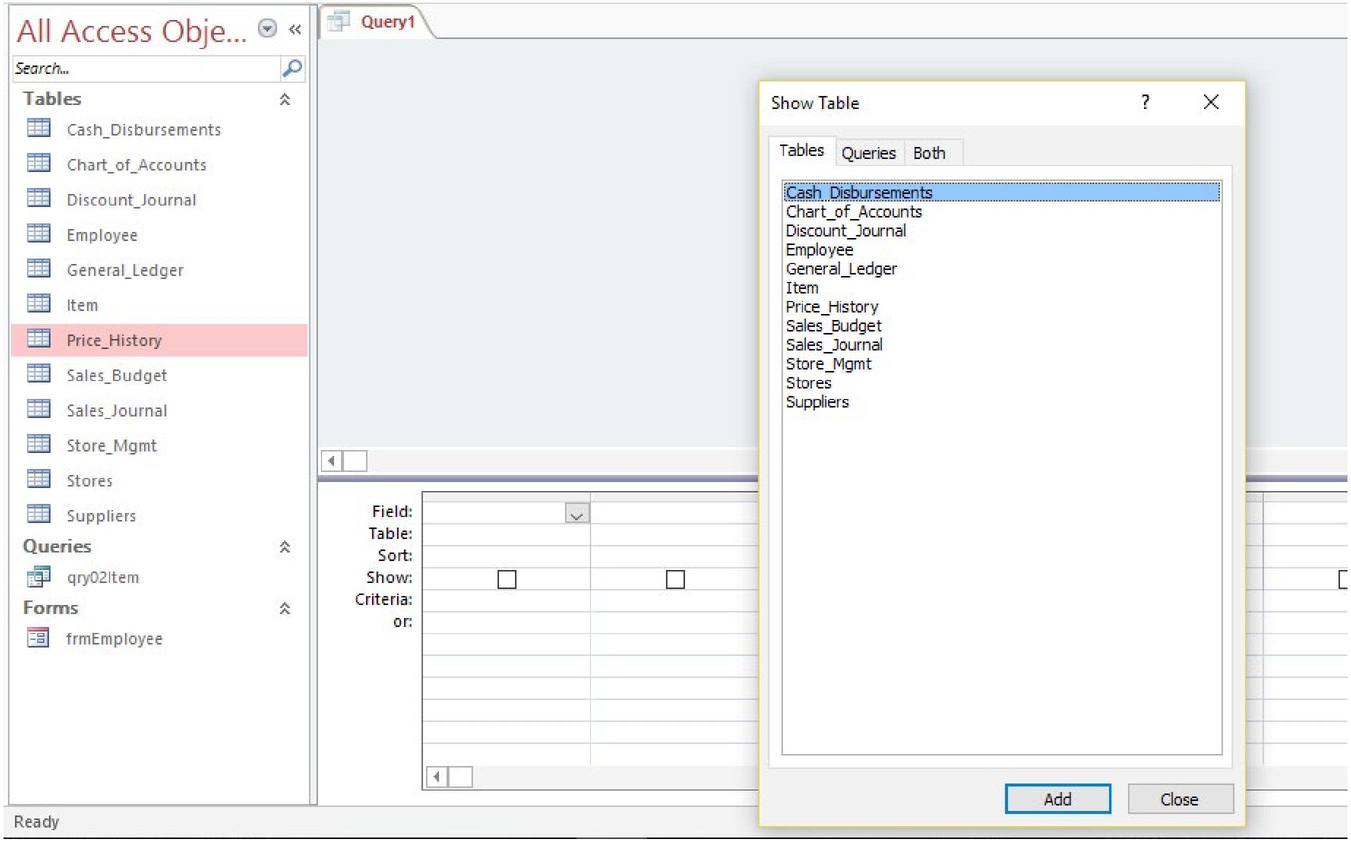


Figure 2.11

Access opens to a new query in Design View and displays the ***Show Table*** dialog box. At this point, Access asks you which table(s) and/or queries you want to use in your query. In this example, I will repeat portions of the filtering and sorting exercise you did in Chapter One using the Employee table. Note that the Show Table dialog box may not appear in a convenient place on your screen. If you want, you can click on the title section of that dialog box and move it to any other position on the screen.

2. *Click once on the* **Employee** *table, click* **Add**, *then click* **Close**.

Your screen should now look like this:

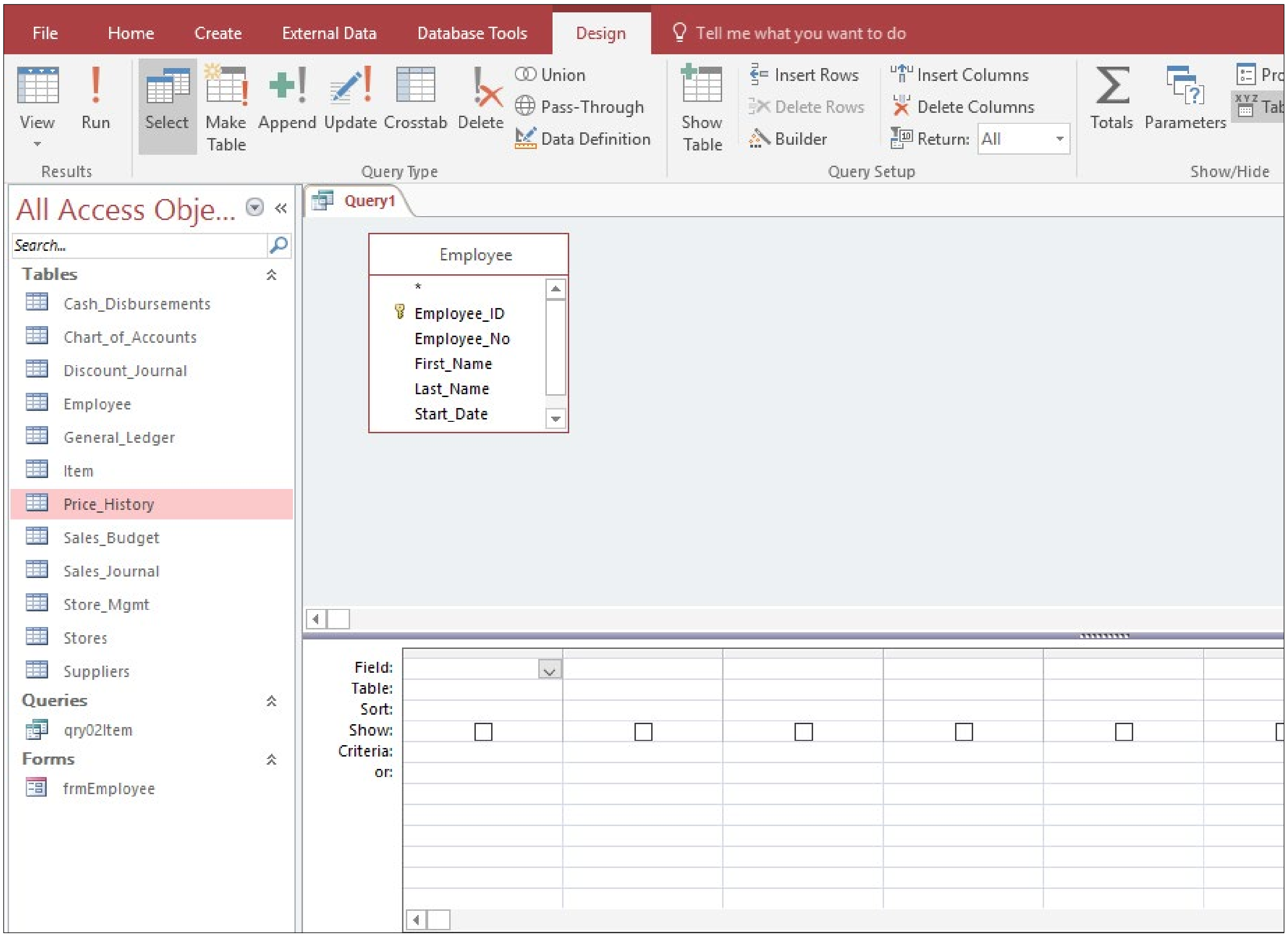


Figure 2.12

Now you are ready to design your query. The Query Design View is separated into two sections: the ***Table Area***, or the gray area in which the Employee and other tables/queries reside, and the ***Design Grid***, the section where Field:, Table:, Sort:, Show:, Criteria:, and or: lines appear. Depending on your screen, you may have to scroll up and down on the Employee table to see all of the fields. Also, there is a lot of space between the design grid and the table area. I like to able to see all of the fields at a glance whenever possible, so let's resize the Design grid and the table so we can see all of the fields in the Employee table.

3. *Click on the bottom-right corner of the* **Employee** *table (your cursor turns to diagonal arrows) and drag it down and to the right until you can see the* **End_Date** *field as well as the entire names of all of the fields in the table, then release.*
4. *Place your cursor over the area just below the horizontal scroll bar in the* **Table** *area. Your cursor will turn to an up-and-down arrow with a horizontal line. Then click and drag the* **Design Grid** *up until it is just below the* **Employee** *table and release.*

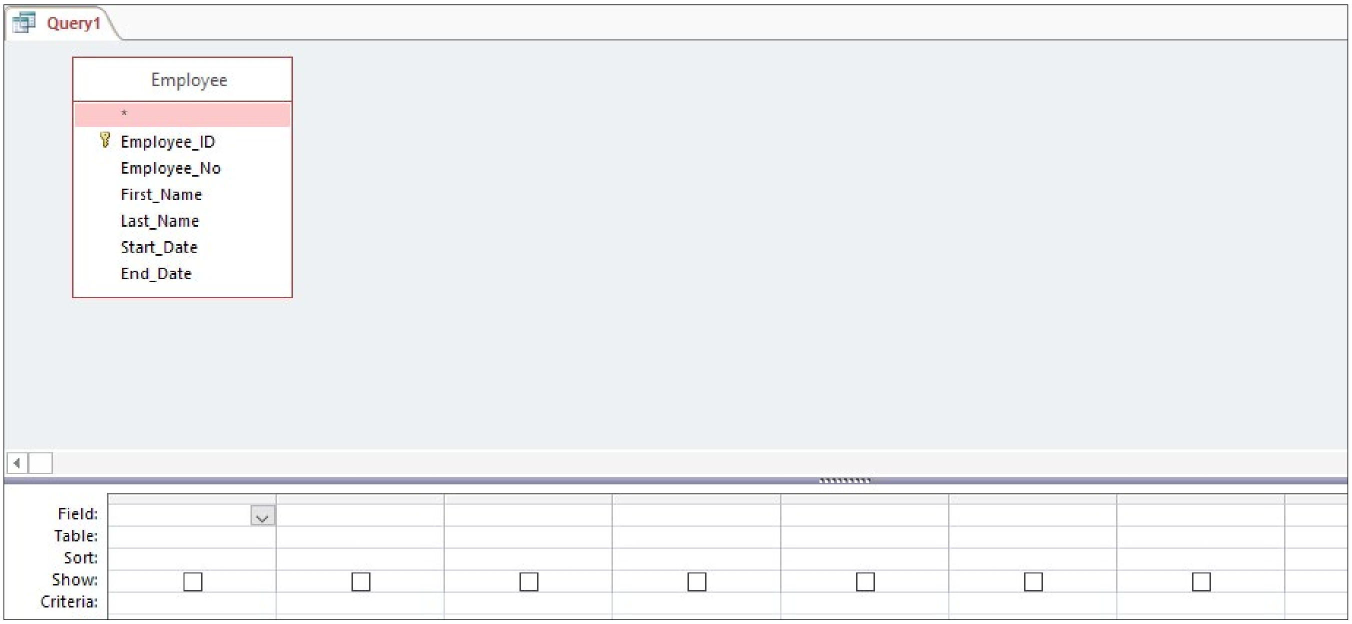
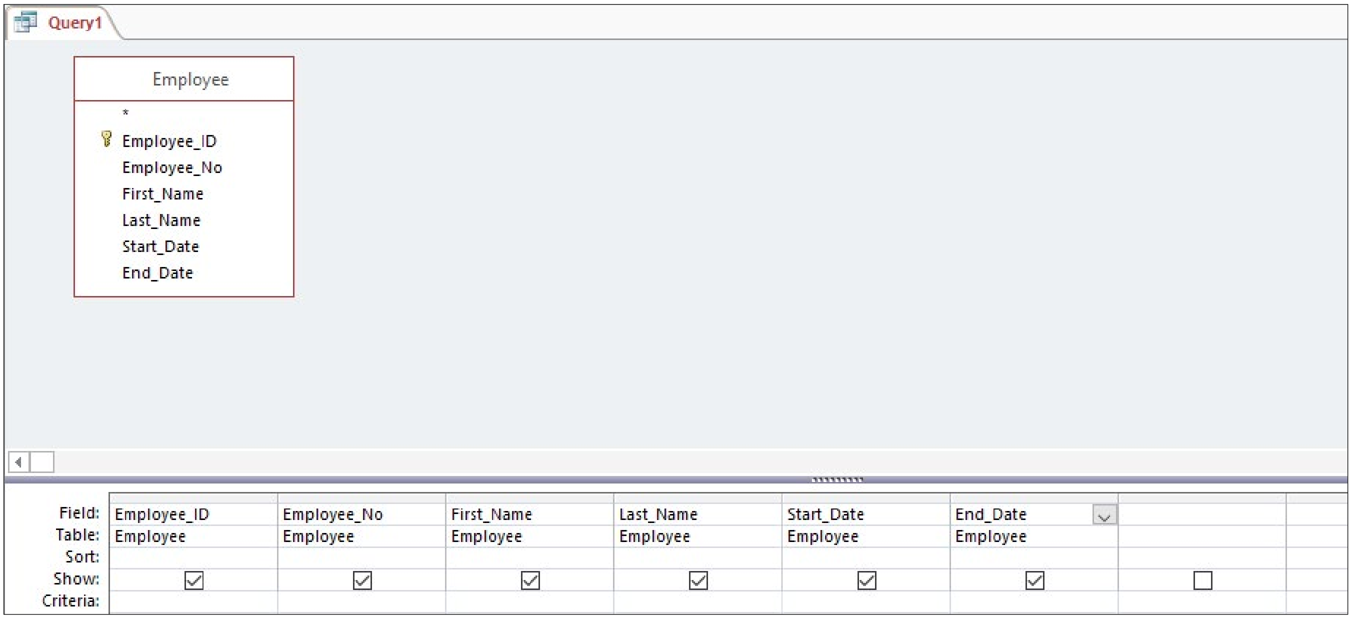


Figure 2.13

Now you are ready to bring fields down to the Design Grid. You do this by dragging each field down, or by double-clicking the field.

5. *Click on the* **Employee_ID** *field and drag it down to the first column in the* **Design Grid** *next to the word* **Field***, and release.*
6. *Double-click the* **Employee_No** *field to bring it down into the* **Design Grid***.*
7. *Bring down each of the remaining fields in the* **Employee** *table to the* **Design Grid***.*

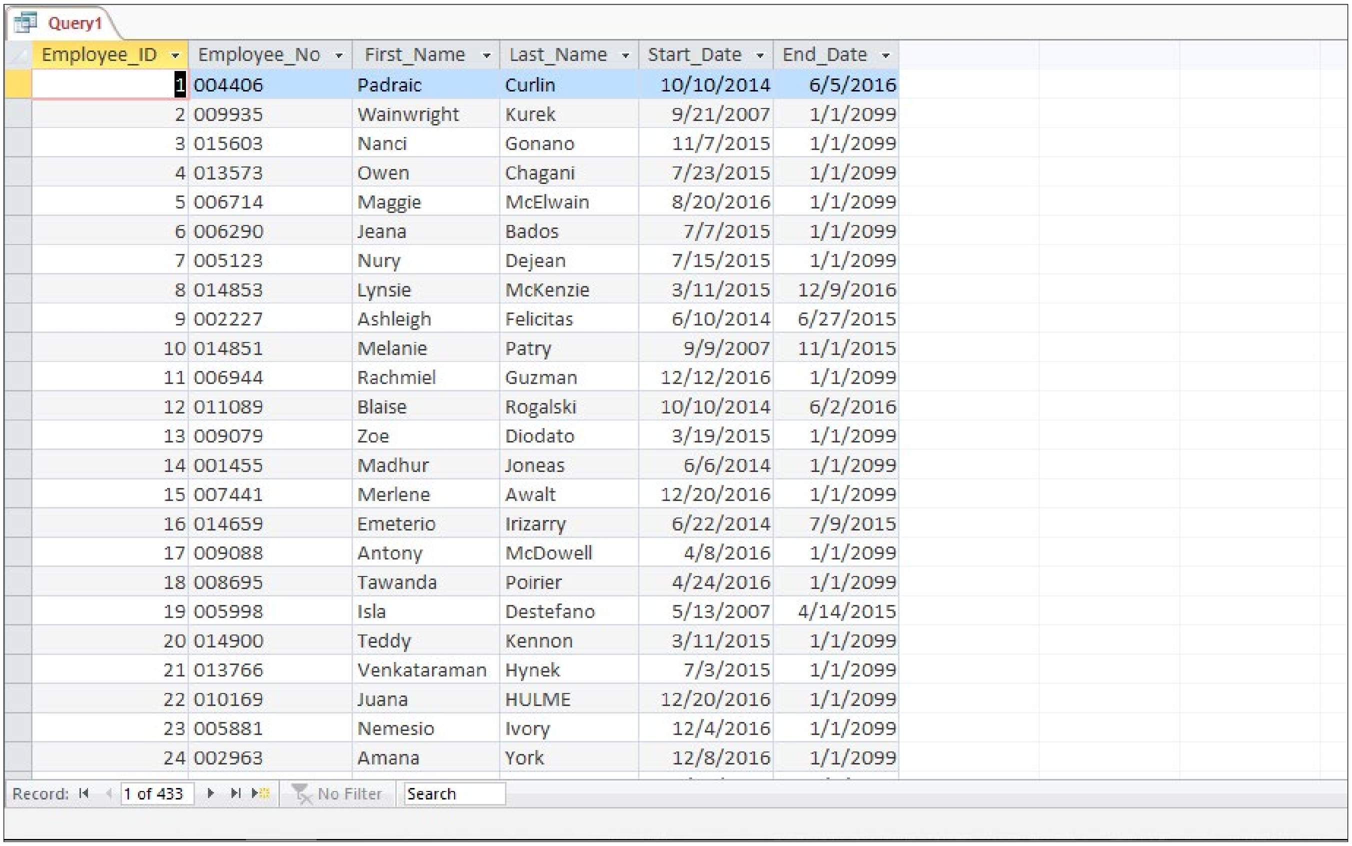


Field:	Employee_ID	Employee_No	First_Name	Last_Name	Start_Date	End_Date	
Table:	Employee	Employee	Employee	Employee	Employee	Employee	
Sort:							
Show:	☑	☑	☑	☑	☑	☑	□
Criteria:							

Figure 2.14

All you've done is create a query that will show you every field and record in the Employee table. The result will be exactly the same as if you had opened the table in the Table screen.

8. *Click on the* **View** *icon to see the query in* **Datasheet View**.
9. *Adjust the margins of each field to where you can see the entire field name.*



Employee_ID	Employee_No	First_Name	Last_Name	Start_Date	End_Date
1	004406	Padraic	Curlin	10/10/2014	6/5/2016
2	009935	Wainwright	Kurek	9/21/2007	1/1/2099
3	015603	Nanci	Gonano	11/7/2015	1/1/2099
4	013573	Owen	Chagani	7/23/2015	1/1/2099
5	006714	Maggie	McElwain	8/20/2016	1/1/2099
6	006290	Jeana	Bados	7/7/2015	1/1/2099
7	005123	Nury	Dejean	7/15/2015	1/1/2099
8	014853	Lynsie	McKenzie	3/11/2015	12/9/2016
9	002227	Ashleigh	Felicitas	6/10/2014	6/27/2015
10	014851	Melanie	Patry	9/9/2007	11/1/2015
11	006944	Rachmiel	Guzman	12/12/2016	1/1/2099
12	011089	Blaise	Rogalski	10/10/2014	6/2/2016
13	009079	Zoe	Diodato	3/19/2015	1/1/2099
14	001455	Madhur	Joneas	6/6/2014	1/1/2099
15	007441	Merlene	Awalt	12/20/2016	1/1/2099
16	014659	Emeterio	Irizarry	6/22/2014	7/9/2015
17	009088	Antony	McDowell	4/8/2016	1/1/2099
18	008695	Tawanda	Poirier	4/24/2016	1/1/2099
19	005998	Isla	Destefano	5/13/2007	4/14/2015
20	014900	Teddy	Kennon	3/11/2015	1/1/2099
21	013766	Venkataraman	Hynek	7/3/2015	1/1/2099
22	010169	Juana	HULME	12/20/2016	1/1/2099
23	005881	Nemesio	Ivory	12/4/2016	1/1/2099
24	002963	Amana	York	12/8/2016	1/1/2099

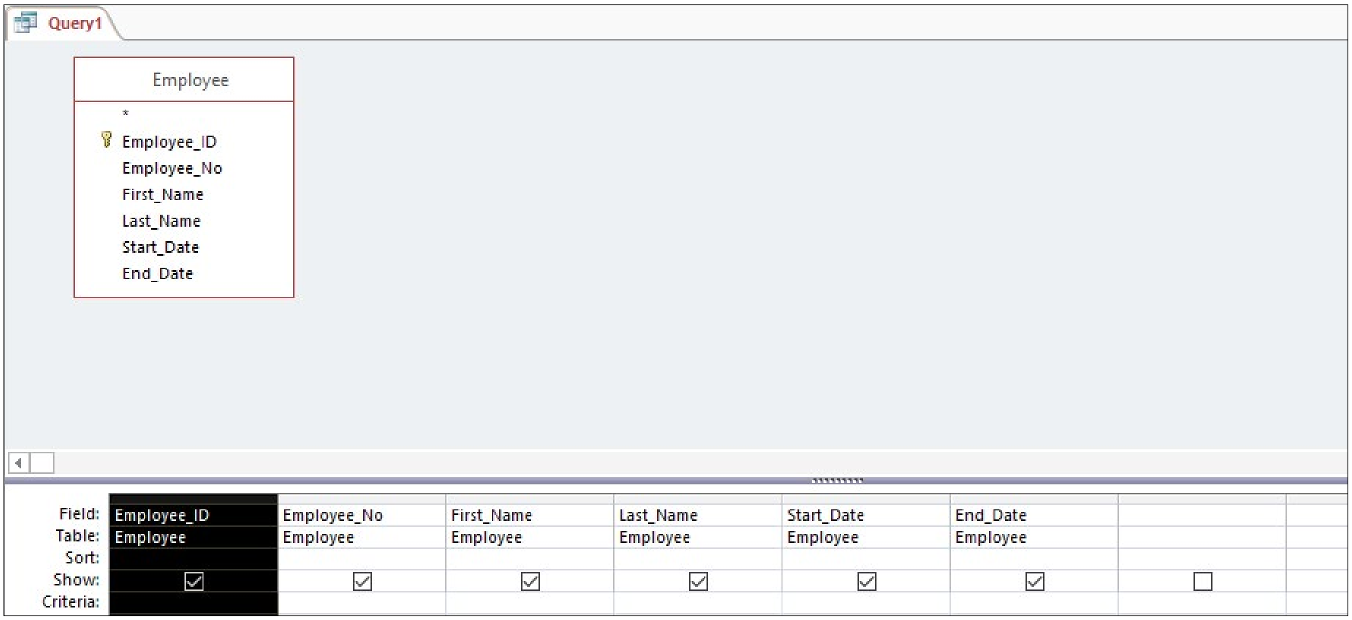
Figure 2.15

Looks familiar, doesn't it? It should. The great thing about a query is that you can design the look of the data anyway you want without altering the actual data in the table.

Deleting Fields from the Query Design Grid

For this exercise, you don't need the Employee_ID field. Removing it is easy.

1. *Return to the* **Design View** *of the query.*
2. *Place your cursor in the thin, gray rectangular box above the* **Employee_ID** *field in the* **Design Grid**. *Your cursor will turn to a down arrow.*
3. *Click on the down arrow to select the* **Employee_ID** *field.*



Employee

*
Employee_ID
Employee_No
First_Name
Last_Name
Start_Date
End_Date

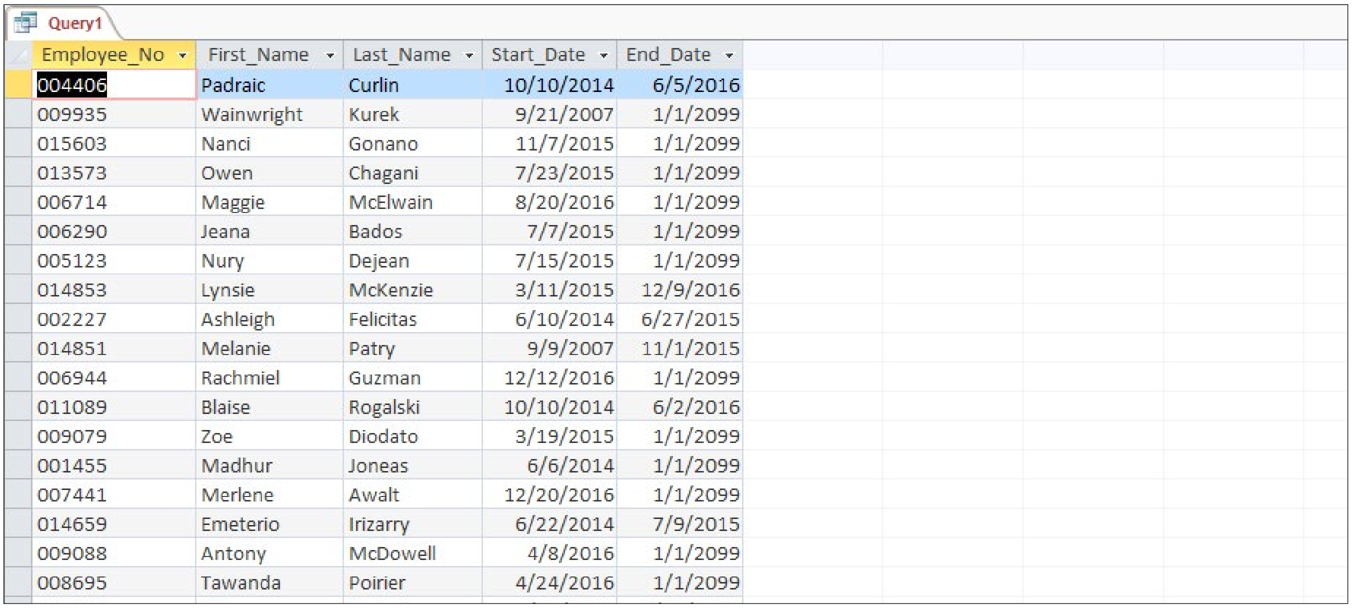
Field:	Employee_ID	Employee_No	First_Name	Last_Name	Start_Date	End_Date	
Table:	Employee	Employee	Employee	Employee	Employee	Employee	
Sort:							
Show:	☑	☑	☑	☑	☑	☑	□
Criteria:							

Figure 2.16

4. *Press the* **Delete** *key on your keyboard (you can also right-click and choose* **Cut***).*

The Employee_ID field is removed from the query Design Grid, but not from the Employee table.

5. *Click on the* **View** *icon to go to* **Datasheet View** Datasheet View .



Employee_No	First_Name	Last_Name	Start_Date	End_Date
004406	Padraic	Curlin	10/10/2014	6/5/2016
009935	Wainwright	Kurek	9/21/2007	1/1/2099
015603	Nanci	Gonano	11/7/2015	1/1/2099
013573	Owen	Chagani	7/23/2015	1/1/2099
006714	Maggie	McElwain	8/20/2016	1/1/2099
006290	Jeana	Bados	7/7/2015	1/1/2099
005123	Nury	Dejean	7/15/2015	1/1/2099
014853	Lynsie	McKenzie	3/11/2015	12/9/2016
002227	Ashleigh	Felicitas	6/10/2014	6/27/2015
014851	Melanie	Patry	9/9/2007	11/1/2015
006944	Rachmiel	Guzman	12/12/2016	1/1/2099
011089	Blaise	Rogalski	10/10/2014	6/2/2016
009079	Zoe	Diodato	3/19/2015	1/1/2099
001455	Madhur	Joneas	6/6/2014	1/1/2099
007441	Merlene	Awalt	12/20/2016	1/1/2099
014659	Emeterio	Irizarry	6/22/2014	7/9/2015
009088	Antony	McDowell	4/8/2016	1/1/2099
008695	Tawanda	Poirier	4/24/2016	1/1/2099

Figure 2.17

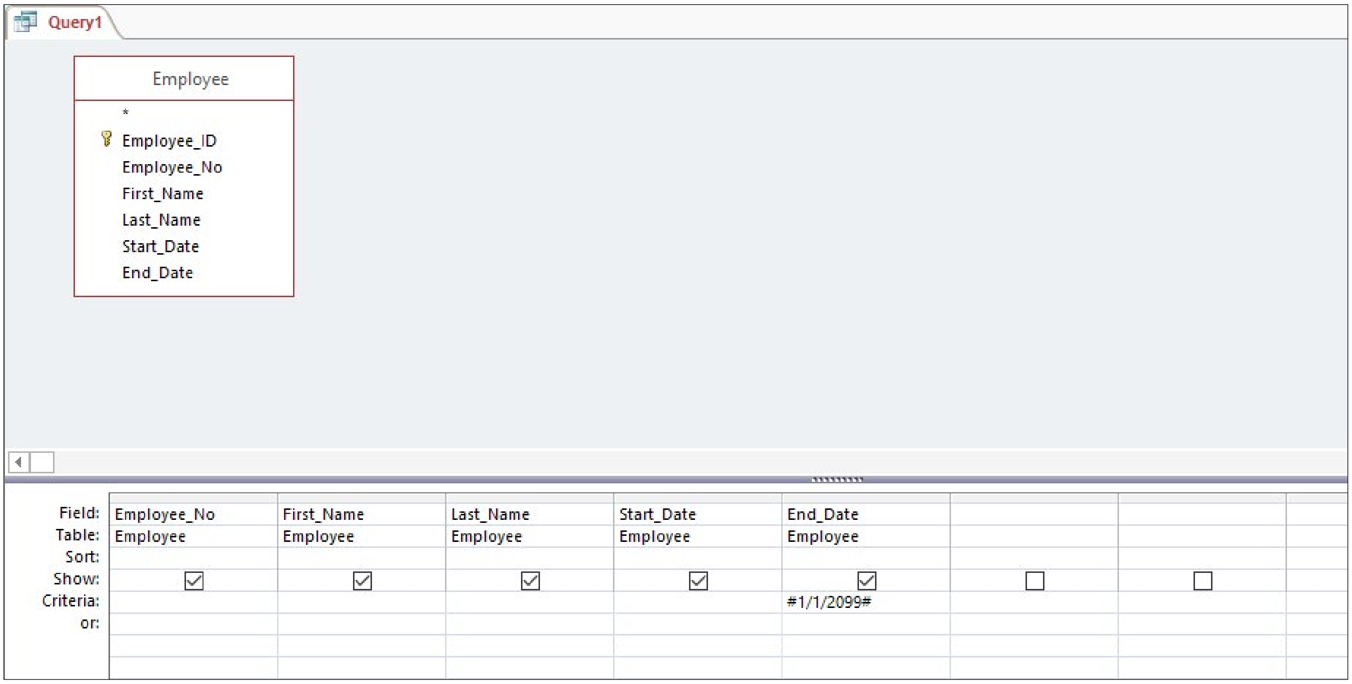
The Employee_ID field no longer appears in the data.

Filter a Query with Criteria

You can also use the Design Grid to filter data. We did a similar example in the first chapter using the Employee table.

1. *Go back to the query's* **Design View**.
2. *On the* **Criteria:** *line of the* **Design Grid** *under* **End_Date**, *type* ***1/1/2099*** *and press* **[Enter]**.

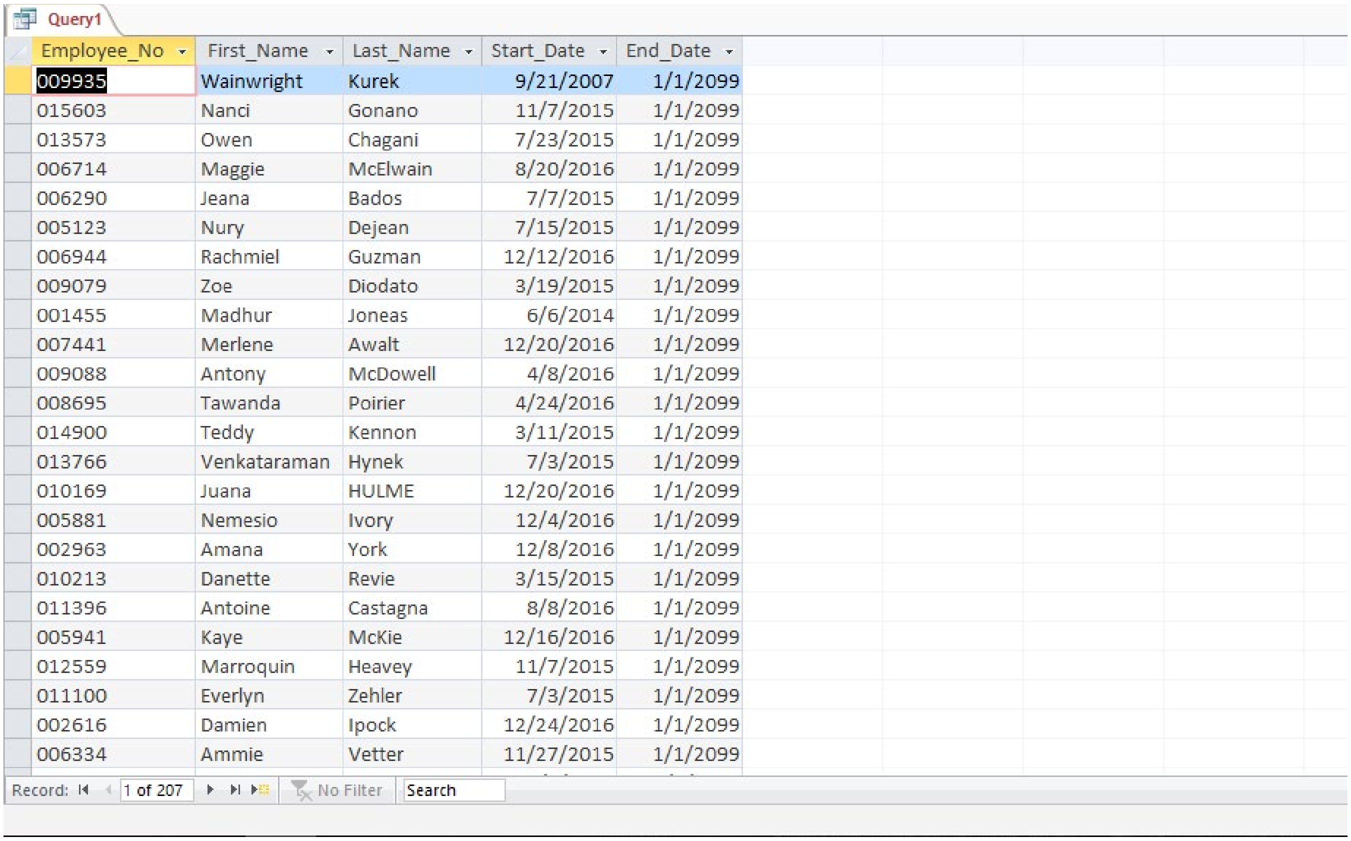
After you press [Enter], Access automatically surrounds 1/1/2099 with # signs. The Design Grid should look like this:



Field:	Employee_No	First_Name	Last_Name	Start_Date	End_Date		
Table:	Employee	Employee	Employee	Employee	Employee		
Sort:							
Show:	☑	☑	☑	☑	☑	□	□
Criteria:					#1/1/2099#		
or:							

Figure 2.18

3. *Go to* **Datasheet View**.



Employee_No	First_Name	Last_Name	Start_Date	End_Date
009935	Wainwright	Kurek	9/21/2007	1/1/2099
015603	Nanci	Gonano	11/7/2015	1/1/2099
013573	Owen	Chagani	7/23/2015	1/1/2099
006714	Maggie	McElwain	8/20/2016	1/1/2099
006290	Jeana	Bados	7/7/2015	1/1/2099
005123	Nury	Dejean	7/15/2015	1/1/2099
006944	Rachmiel	Guzman	12/12/2016	1/1/2099
009079	Zoe	Diodato	3/19/2015	1/1/2099
001455	Madhur	Joneas	6/6/2014	1/1/2099
007441	Merlene	Awalt	12/20/2016	1/1/2099
009088	Antony	McDowell	4/8/2016	1/1/2099
008695	Tawanda	Poirier	4/24/2016	1/1/2099
014900	Teddy	Kennon	3/11/2015	1/1/2099
013766	Venkataraman	Hynek	7/3/2015	1/1/2099
010169	Juana	HULME	12/20/2016	1/1/2099
005881	Nemesio	Ivory	12/4/2016	1/1/2099
002963	Amana	York	12/8/2016	1/1/2099
010213	Danette	Revie	3/15/2015	1/1/2099
011396	Antoine	Castagna	8/8/2016	1/1/2099
005941	Kaye	McKie	12/16/2016	1/1/2099
012559	Marroquin	Heavey	11/7/2015	1/1/2099
011100	Everlyn	Zehler	7/3/2015	1/1/2099
002616	Damien	Ipock	12/24/2016	1/1/2099
006334	Ammie	Vetter	11/27/2015	1/1/2099

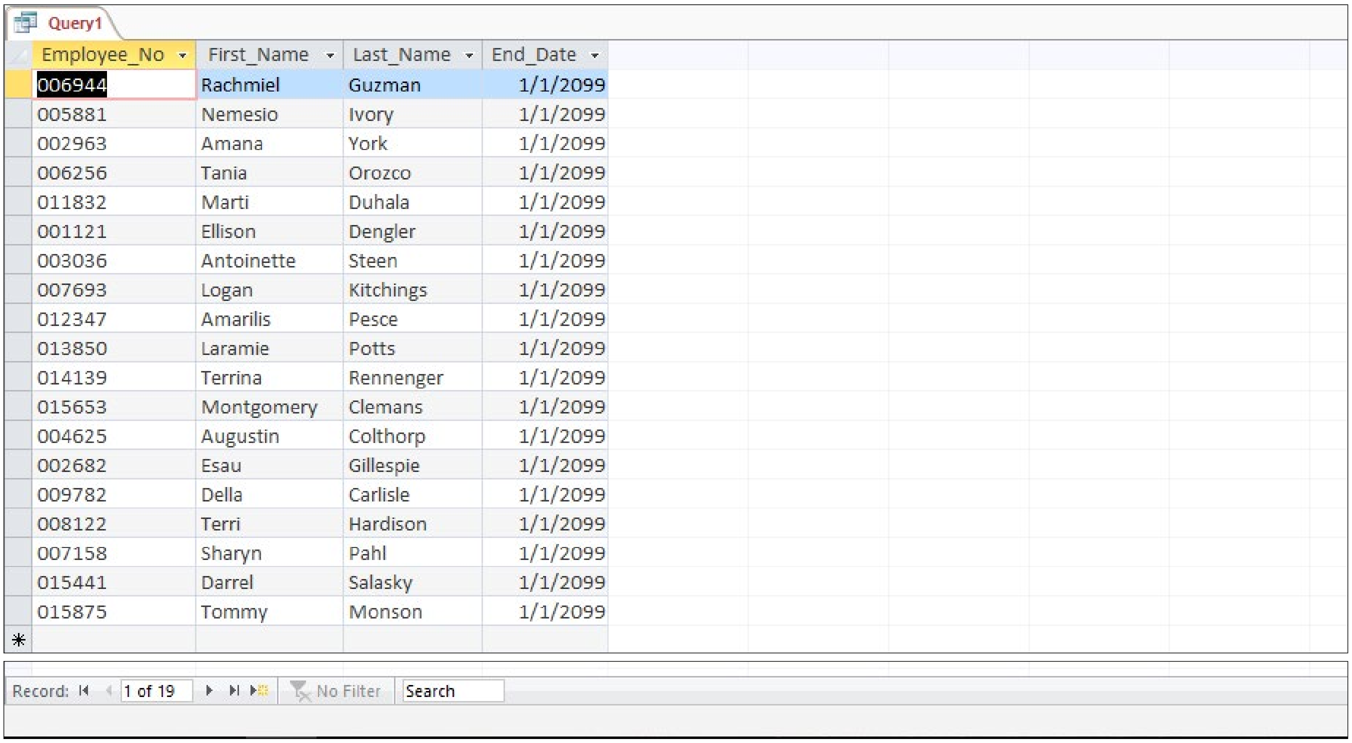
Figure 2.19

The data is now filtered to include only records with an End_Date of 1/1/2099. As you can see, there are 207 records.

Writing Formulas

You can also write formulas in Design View on the Field:, Criteria:, and or: lines. Let's repeat the Chapter One example and filter the Employee table for all employees who started between 12/1/2016 and 12/15/2016.

1. *Go to the query's* **Design View**.
2. *Delete the criteria under* **End_Date**.
3. *On the* **Criteria:** *line for* **Start_Date**, *type* ***>=12/1/2016 and <=12/15/2016*** *and press* **[Enter]**.
4. *Go to* **Datasheet View**.



Query1

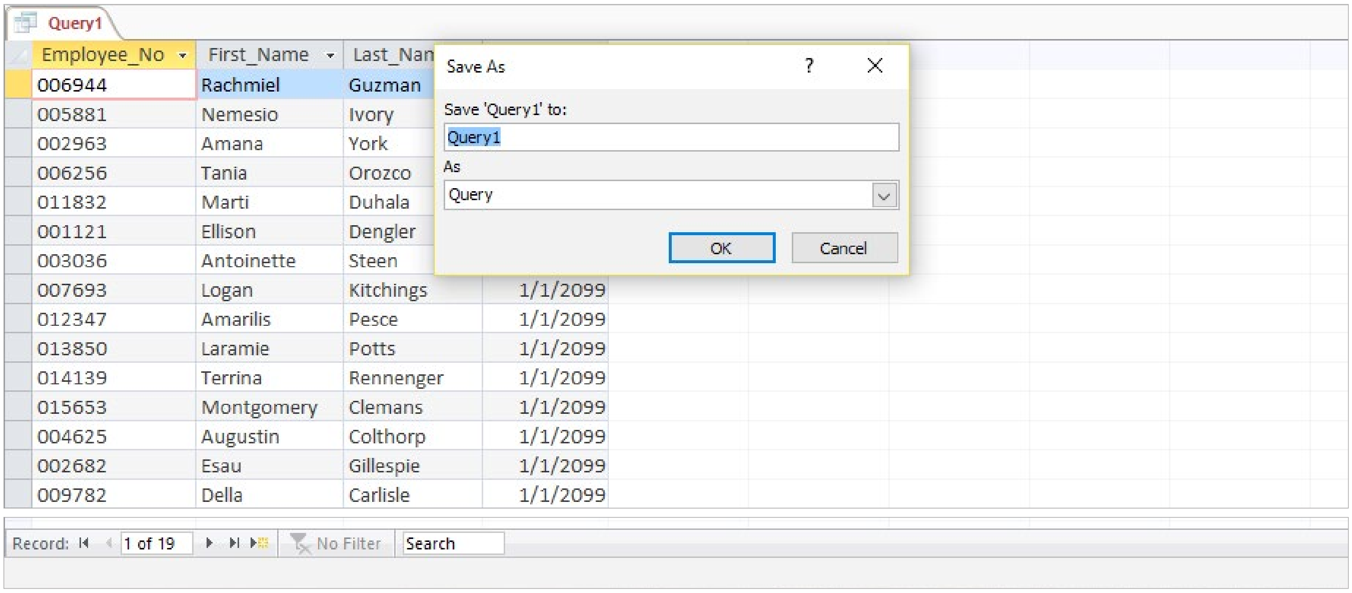
Employee_No	First_Name	Last_Name	End_Date
006944	Rachmiel	Guzman	1/1/2099
005881	Nemesio	Ivory	1/1/2099
002963	Amana	York	1/1/2099
006256	Tania	Orozco	1/1/2099
011832	Marti	Duhala	1/1/2099
001121	Ellison	Dengler	1/1/2099
003036	Antoinette	Steen	1/1/2099
007693	Logan	Kitchings	1/1/2099
012347	Amarilis	Pesce	1/1/2099
013850	Laramie	Potts	1/1/2099
014139	Terrina	Rennenger	1/1/2099
015653	Montgomery	Clemans	1/1/2099
004625	Augustin	Colthorp	1/1/2099
002682	Esau	Gillespie	1/1/2099
009782	Della	Carlisle	1/1/2099
008122	Terri	Hardison	1/1/2099
007158	Sharyn	Pahl	1/1/2099
015441	Darrel	Salasky	1/1/2099
015875	Tommy	Monson	1/1/2099

Record: 1 of 19 No Filter Search

Figure 2.20

There are 19 records in the query. In the exercise in Chapter One, there were 18 records. Why the difference? Remember, you added your name as a new record to the Employees table, and so the record of you is also visible. That is the extra record.

5. *Click on the* **Save As** *icon in the* **Quick Access Toolbar**.



Query1

Employee_No	First_Name	Last_Nan	
006944	Rachmiel	Guzman	
005881	Nemesio	Ivory	
002963	Amana	York	
006256	Tania	Orozco	
011832	Marti	Duhala	
001121	Ellison	Dengler	
003036	Antoinette	Steen	
007693	Logan	Kitchings	1/1/2099
012347	Amarilis	Pesce	1/1/2099
013850	Laramie	Potts	1/1/2099
014139	Terrina	Rennenger	1/1/2099
015653	Montgomery	Clemans	1/1/2099
004625	Augustin	Colthorp	1/1/2099
002682	Esau	Gillespie	1/1/2099
009782	Della	Carlisle	1/1/2099

Save As

Save 'Query1' to: Query1

As: Query

Record: 1 of 19 No Filter Search

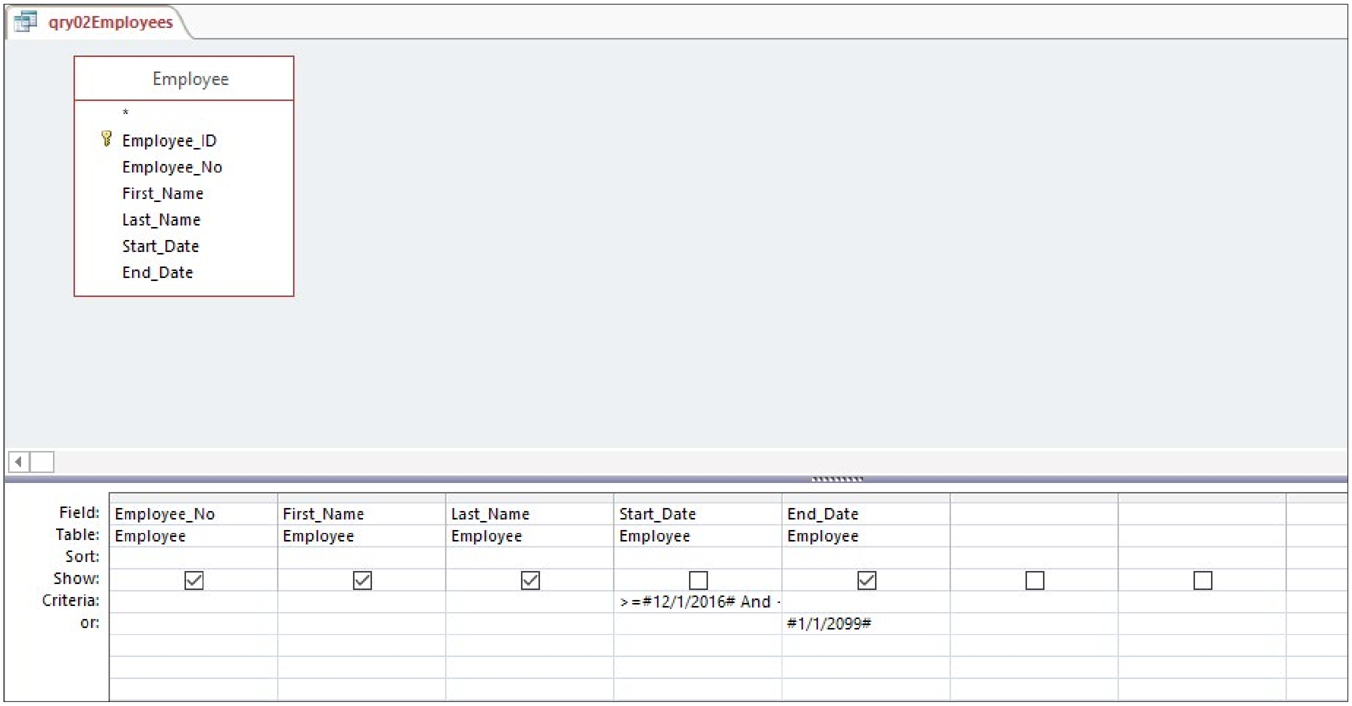
Figure 2.21

6. *Name the query* ***qry02Employees****, leave the* **As** *box set to* **Query***, and click* **OK***.*

OR Criteria

Now that you've had a little experience in writing formulas on the Criteria line, let's explore the next line in the Design Grid, the "***or:***" line. You can think of this line as similar to an OR() function in Excel. You use the or: line when you want to specify criteria that could meet either of two or more conditions. Let's say that you want to modify the qry02Employees query to give you a list of employees with a start date between 12/1/2016 and 12/15/2016, and you also want all of the current employees. You can use the or: line to do this.

7. *Go to the* **Design View** *of* **qry02Employee** *and move or type* ***1/1/2099*** *on the* **or:** *line under the* **End_Date** *field (removing it from the* **Criteria:** *line).*
8. *Under* **Start Date***, type* ***>=12/1/2016 and <=12/15/2016*** *on the* **Criteria:** *line.*

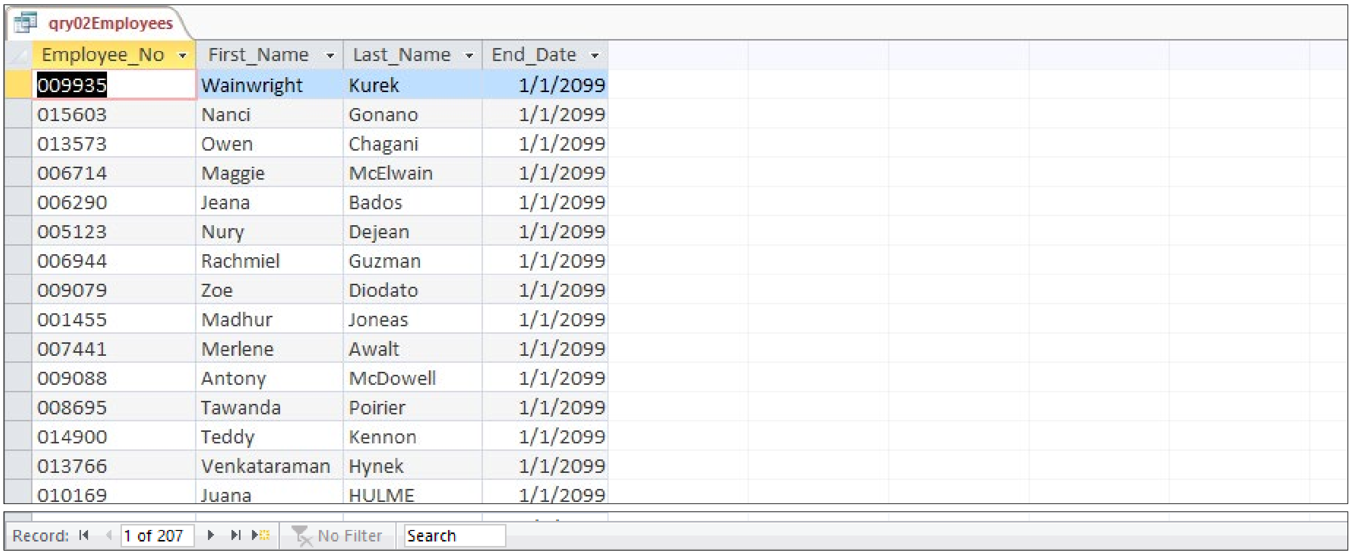


Field:	Employee_No	First_Name	Last_Name	Start_Date	End_Date		
Table:	Employee	Employee	Employee	Employee	Employee		
Sort:							
Show:	☑	☑	☑	□	☑	□	□
Criteria:				>=#12/1/2016# And ·			
or:					#1/1/2099#		

Figure 2.22

With this query, you are telling Access that you want a list that contains all employees with a start date between 12/1/2016 and 12/15/2016, as well as all of the current employees. If you were to write both criteria on the same Criteria line, it would return records that meet both criteria, which would be all current employees who started between 12/1/2016 and 12/15/2016. That logic is similar to the AND() function in Excel. Make sure that you have the logic sorted out in your mind as to how this works.

9. **Run** ! Run *the query.*



Employee_No	First_Name	Last_Name	End_Date
009935	Wainwright	Kurek	1/1/2099
015603	Nanci	Gonano	1/1/2099
013573	Owen	Chagani	1/1/2099
006714	Maggie	McElwain	1/1/2099
006290	Jeana	Bados	1/1/2099
005123	Nury	Dejean	1/1/2099
006944	Rachmiel	Guzman	1/1/2099
009079	Zoe	Diodato	1/1/2099
001455	Madhur	Joneas	1/1/2099
007441	Merlene	Awalt	1/1/2099
009088	Antony	McDowell	1/1/2099
008695	Tawanda	Poirier	1/1/2099
014900	Teddy	Kennon	1/1/2099
013766	Venkataraman	Hynek	1/1/2099
010169	Juana	HULME	1/1/2099

Figure 2.23

Now you get a list that is much larger than 19 records. In addition to the new employees, it contains all of the current employees. There should be 207 records in this record set.

10. **Save** *and close the* **qry02Employee** *query.*

Now that you know what a simple (and I mean VERY simple) query can do, the next chapter will build on that knowledge and focus on more intermediate skills of writing queries.

> ***Review Questions****: It is now time to complete the hands-on Review Questions. Log on to www.ExcelCEO.com with your Email and Password, click on the* **Access 2016 and SQL Review Questions, Chapter 2, Section 2 of 2** *option and complete the review questions.*

Conclusion

In this chapter, you created a simple Access query from the Query Wizard and from Design View. You learned how to add and delete fields from the query design grid. You practiced filtering a query with criteria you wrote on the Criteria line. You also saw how to write formulas on the Criteria line, whose syntax is similar to writing formulas in Excel. You learned about the OR line of the query design grid, and how writing criteria on the same line or on different lines can dramatically change your queried record set.

Chapter Exam

You can now go to www.ExcelCEO.com, click on Sign In, log in and take the exam. Make sure that you take the exam on the same device on which you completed the practice files, as some of the questions on the exam may refer to some of the completed projects. Chapter exams are intended to be hands-on.

Access® 2016 and SQL

Complete Self-study Course

ExcelCEO

Chief Excel Officer

CHAPTER THREE — INTERMEDIATE QUERIES

In this chapter, you will:

- Identify the methods to pull all fields from a table into a query
- Recognize how to join (or create relationships between) multiple tables
- Select the options to format fields in the Design grid of a query
- Choose the correct method for creating an Alias
- Identify the differences between Flat Files vs. Hierarchical tables

CPE Credits possible for this chapter: 2

More Query Skills

Here's a tidbit of information you'll probably never use, but it's kind of interesting to know. One time I had an extremely complex Access 97 database. We were doing thousands of calculations in the database. Some of the queries were so complex that we started to get error messages like, "*The query is too complex. Simplify it.*" Once we created a query and then tried to select it from the Queries pane. We knew we created it, but the query name didn't appear in the Queries pane along with the hundreds of other queries we built. We could see it if we tried to use it in a query Design View, but it didn't appear in the Queries tab. We later found out that you can have approximately 600 queries in one database before it doesn't display the more recently created ones in the Queries tab. We had discovered that little-known limitation of Access 97. Access 2016 has significantly expanded the capabilities, and now you can have up to 50 'nested' queries (something you certainly don't need to know for this course).

In this chapter, you will build on the simple query skills you acquired in Chapter 2. This chapter is also extremely important to your future database design education, so if you don't catch on to all of the concepts taught in this chapter, go over them again and again until you understand everything. In this chapter, we will concentrate on writing formulas in queries and joining tables.

1. *In the* **Nitey_Nite_2016** *database, create a new query in* **Design View** *using the* **Cash_ Disbursements** *table.*
2. *Modify the* **Design View** *to make it look like this:*

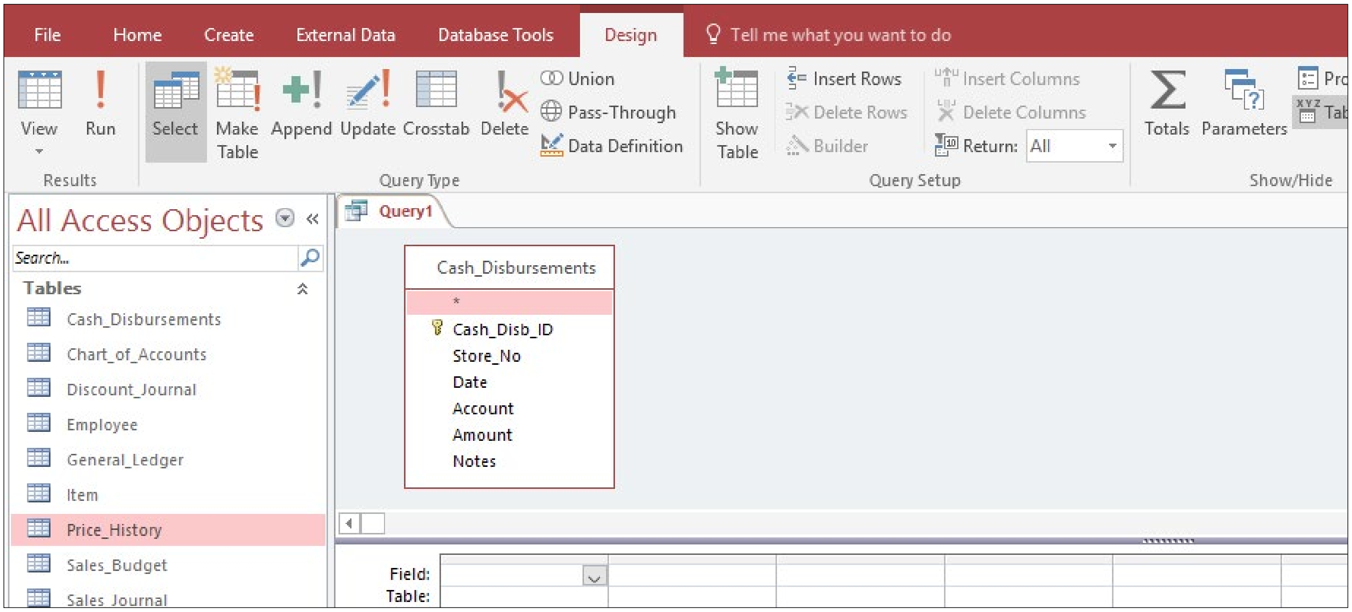
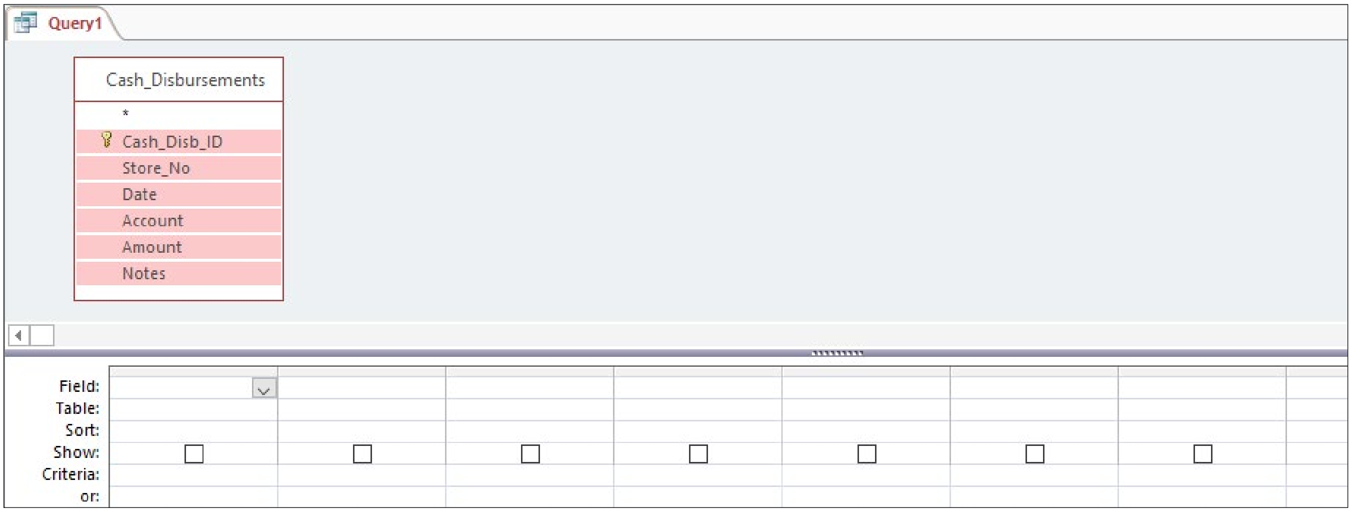


Figure 3.1

As explained in Chapter One, the Cash_Disbursements table is kind of like a checkbook register. When expenses are paid, the accountant at Nitey-Nite records the transactions in this journal. You created the primary key (Cash_Disb_ID) for this table in Chapter One. The table also contains the Store_No, Date (of the transaction), the GL Account it was booked to, the Amount of the transaction, and Notes giving more details on the transaction. Let's review the data in this table.

3. *Double-click on the table name* **Cash_Disbursements** *in the* **Table Area** *of the* **Design Grid**.

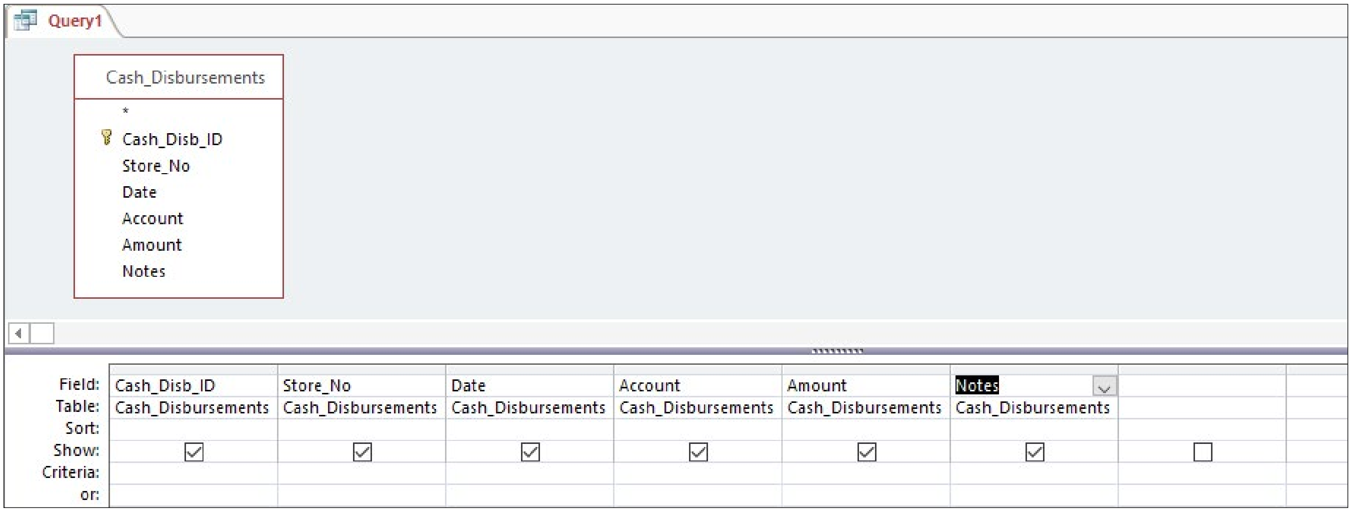


Field:							
Table:							
Sort:							
Show:	☐	☐	☐	☐	☐	☐	☐
Criteria:							
or:							

Figure 3.2

All of the fields in the table are selected.

4. *Click and hold on any of the selected fields and drag the group down to the first column in the* **Design Grid** *and release.*

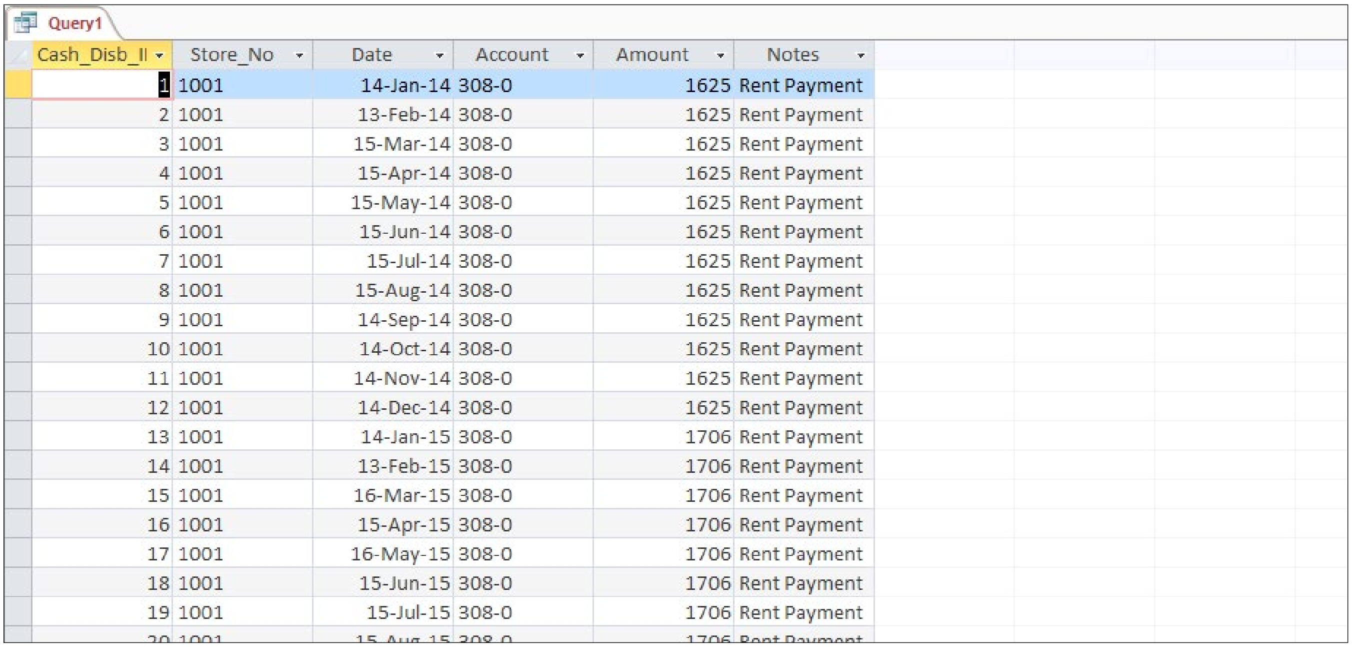


Field:	Cash_Disb_ID	Store_No	Date	Account	Amount	Notes	
Table:	Cash_Disbursements	Cash_Disbursements	Cash_Disbursements	Cash_Disbursements	Cash_Disbursements	Cash_Disbursements	
Sort:							
Show:	☑	☑	☑	☑	☑	☑	☐
Criteria:							
or:							

Figure 3.3

This is a quick and easy way to bring all of the fields in a table into the Design Grid. Another way is to click and drag the asterisk (*) from the table into the Design Grid. I typically don't like to do that because only the asterisk appears in the Design Grid, even though all fields appear when you run the query.

5. *View the data in* **Datasheet View**.



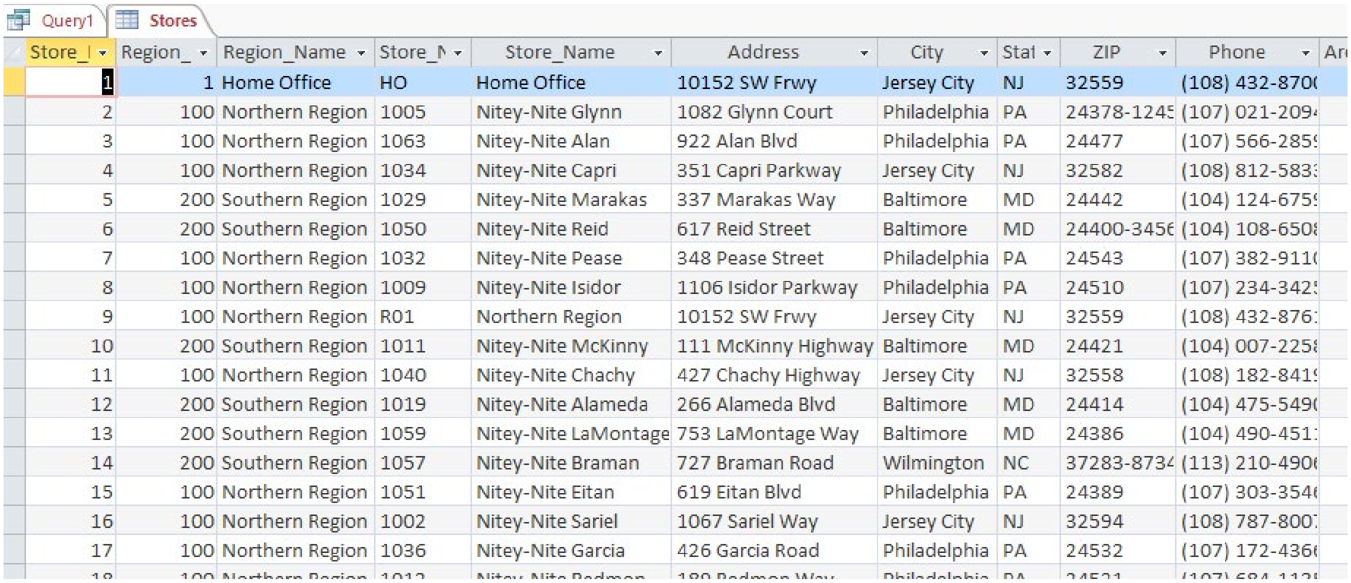
Query1

Cash_Disb_II	Store_No	Date	Account	Amount	Notes
1	1001	14-Jan-14	308-0	1625	Rent Payment
2	1001	13-Feb-14	308-0	1625	Rent Payment
3	1001	15-Mar-14	308-0	1625	Rent Payment
4	1001	15-Apr-14	308-0	1625	Rent Payment
5	1001	15-May-14	308-0	1625	Rent Payment
6	1001	15-Jun-14	308-0	1625	Rent Payment
7	1001	15-Jul-14	308-0	1625	Rent Payment
8	1001	15-Aug-14	308-0	1625	Rent Payment
9	1001	14-Sep-14	308-0	1625	Rent Payment
10	1001	14-Oct-14	308-0	1625	Rent Payment
11	1001	14-Nov-14	308-0	1625	Rent Payment
12	1001	14-Dec-14	308-0	1625	Rent Payment
13	1001	14-Jan-15	308-0	1706	Rent Payment
14	1001	13-Feb-15	308-0	1706	Rent Payment
15	1001	16-Mar-15	308-0	1706	Rent Payment
16	1001	15-Apr-15	308-0	1706	Rent Payment
17	1001	16-May-15	308-0	1706	Rent Payment
18	1001	15-Jun-15	308-0	1706	Rent Payment
19	1001	15-Jul-15	308-0	1706	Rent Payment
20	1001	15-Aug-15	308-0	1706	Rent Payment

Figure 3.4

According to the Notes field, the first few records of this table appear to be rent payments for Store No 1001. The payments were made around the 15th of each month, and we know that the amount of the rent payment was $1,675. As you scroll down this record set, you will see different store numbers, dates, accounts, amounts, and notes on each record. The Store_No field doesn't tell us much about the store – we know only that it is Store_No 1001. Let's find out more about that particular store.

6. *Double-click on the* **Stores** *table.*



Query1 | Stores

Store_I	Region_	Region_Name	Store_N	Store_Name	Address	City	Stat	ZIP	Phone	Ar
1	1	Home Office	HO	Home Office	10152 SW Frwy	Jersey City	NJ	32559	(108) 432-870	
2	100	Northern Region	1005	Nitey-Nite Glynn	1082 Glynn Court	Philadelphia	PA	24378-124	(107) 021-209	
3	100	Northern Region	1063	Nitey-Nite Alan	922 Alan Blvd	Philadelphia	PA	24477	(107) 566-285	
4	100	Northern Region	1034	Nitey-Nite Capri	351 Capri Parkway	Jersey City	NJ	32582	(108) 812-583	
5	200	Southern Region	1029	Nitey-Nite Marakas	337 Marakas Way	Baltimore	MD	24442	(104) 124-675	
6	200	Southern Region	1050	Nitey-Nite Reid	617 Reid Street	Baltimore	MD	24400-345	(104) 108-650	
7	100	Northern Region	1032	Nitey-Nite Pease	348 Pease Street	Philadelphia	PA	24543	(107) 382-911	
8	100	Northern Region	1009	Nitey-Nite Isidor	1106 Isidor Parkway	Philadelphia	PA	24510	(107) 234-342	
9	100	Northern Region	R01	Northern Region	10152 SW Frwy	Jersey City	NJ	32559	(108) 432-876	
10	200	Southern Region	1011	Nitey-Nite McKinny	111 McKinny Highway	Baltimore	MD	24421	(104) 007-225	
11	100	Northern Region	1040	Nitey-Nite Chachy	427 Chachy Highway	Jersey City	NJ	32558	(108) 182-841	
12	200	Southern Region	1019	Nitey-Nite Alameda	266 Alameda Blvd	Baltimore	MD	24414	(104) 475-549	
13	200	Southern Region	1059	Nitey-Nite LaMontage	753 LaMontage Way	Baltimore	MD	24386	(104) 490-451	
14	200	Southern Region	1057	Nitey-Nite Braman	727 Braman Road	Wilmington	NC	37283-873	(113) 210-490	
15	100	Northern Region	1051	Nitey-Nite Eitan	619 Eitan Blvd	Philadelphia	PA	24389	(107) 303-354	
16	100	Northern Region	1002	Nitey-Nite Sariel	1067 Sariel Way	Jersey City	NJ	32594	(108) 787-800	
17	100	Northern Region	1036	Nitey-Nite Garcia	426 Garcia Road	Philadelphia	PA	24532	(107) 172-436	
18	100	Northern Region	1012	Nitey-Nite Redmon	189 Redmon Way	Philadelphia	PA	24521	(107) 684-112	

Figure 3.5

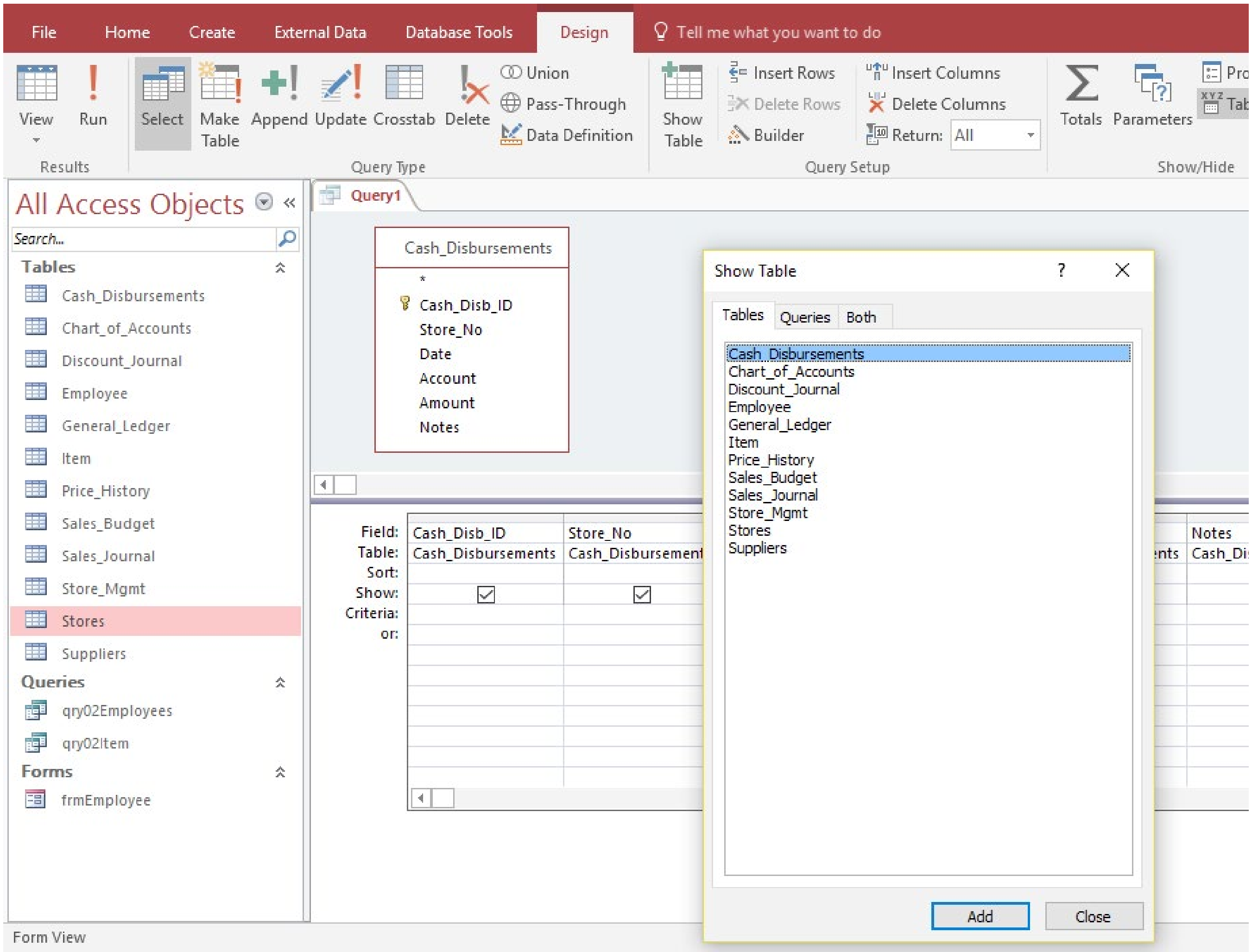
Joining Tables

A new tab opens up that contains the data from the Stores table. The Stores table contains information about each store. In the table, you see that Store_No 1001 is called Nitey-Nite Miami, and is located at 10101 Miami St. in New York. The table also shows the State, ZIP, Phone number, and the Area of the store in square feet. It also shows that the Store_ID is 19 and that it's in the Northern Region, but we'll talk about those fields later. This gives you some really good information about the store, but how do you get this data into the query we're building? In Excel, you could do a VLOOKUP() function and bring in whatever data you need with no problem. But how do you do it in Access? The answer is to create a join, or a relationship, between the two tables. Remember that Access is a relational database, meaning you can store data in separate tables and create relationships between the tables linking the data.

This is a major hurdle that financial people must overcome: understanding why and how to join tables in a relational database. It's really not that hard, if you think of a join as the VLOOKUP() of Access. If we were to store all the necessary information pertinent to cash disbursements in one table, we would have to include all of the store information from the Stores table, as well as all of the account information from the Chart_of_Accounts table, and information from any other necessary table. That would result in a lot of duplicate data in one monster file. One of the goals of a relational database is to store data only once, thus minimizing storage space required. You should store data in separate, compact tables, and create relationships between the tables to pull the data together. Creating a relationship in Access is easy: just click and drag.

In this scenario, let's say we want to see the store number, store name, the date of the transaction, the account, and the amount of the transaction all in one query. The store number and store name come from the Stores table, and the remaining fields will come from the Cash_Disbursements table. The common field between these two tables is the store number, so we will create a join between the store number in the Stores table and the store number in the Cash_Disbursements table. That would be the field we would do the VLOOKUP() on if we were doing it in Excel.

7. *Close the* **Stores** *table.*
8. *Return to* **Design View** *of* **Query1**.
9. *Click on the* **Show Table** *icon in the* **Query Setup** *group of the* **Query Tools Design** *tab (to bring in another table).*

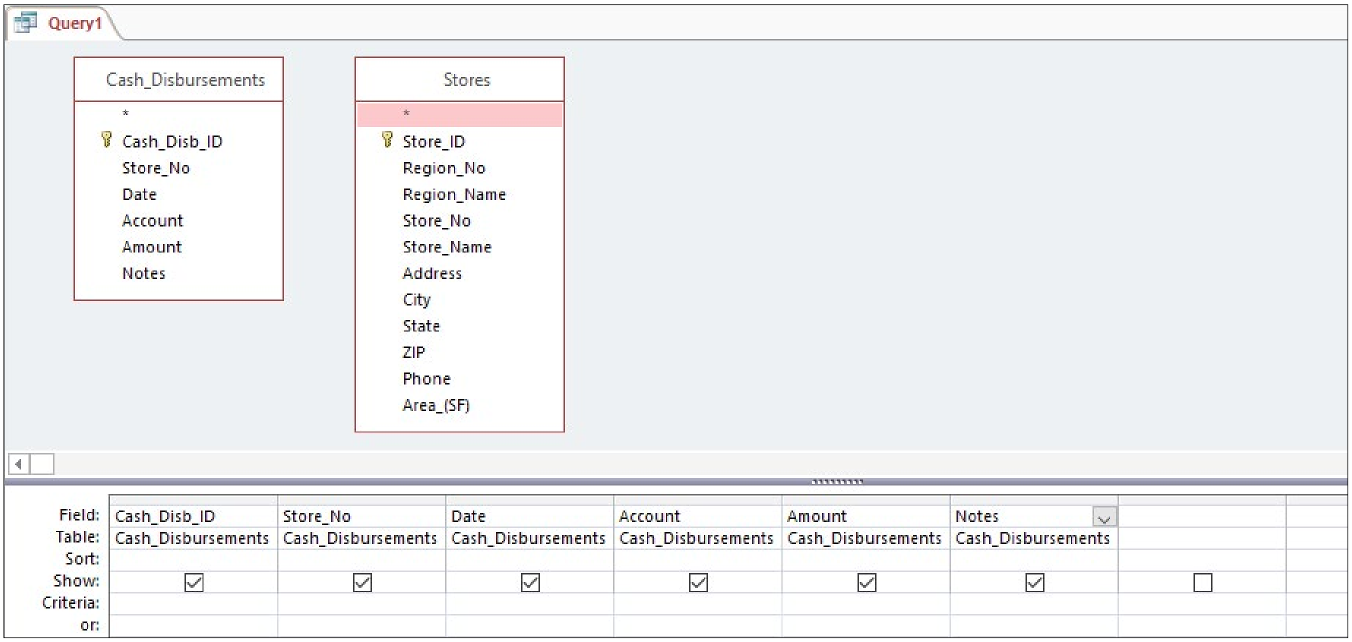


Field:	Cash_Disb_ID	Store_No		Notes
Table:	Cash_Disbursements	Cash_Disbursement	ents	Cash_Di
Sort:				
Show:	☑	☑		
Criteria:				
or:				

Figure 3.6

The Show Table dialog box appears. This allows you to bring other tables into the query. You will now add the Stores table.

10. *Click on the* **Stores** *table, and click the* **Add** *button (You can also double-click on the table/ query name).*
11. *When the* **Stores** *table appears in the* **Table Area**, *click the* **Close** *button.*
12. *If necessary, adjust the margins of the* **Stores** *table where you can see all of the fields.*



Query1

Cash_Disbursements
*
Cash_Disb_ID
Store_No
Date
Account
Amount
Notes

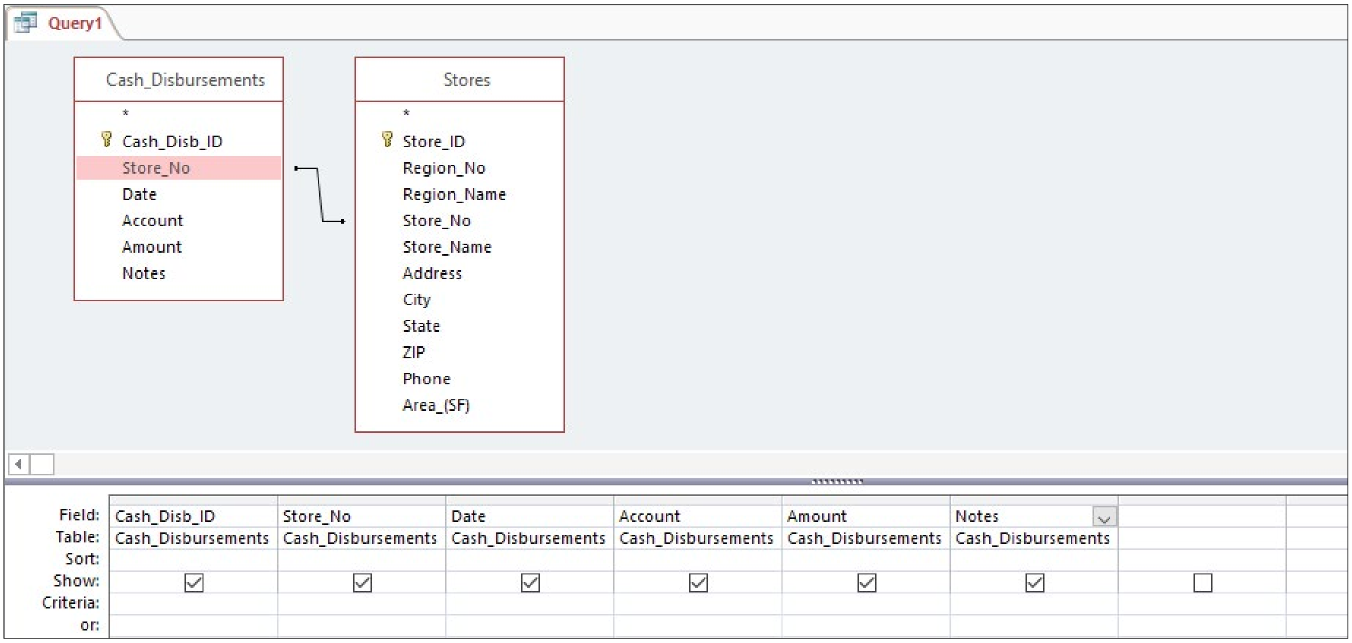
Stores
*
Store_ID
Region_No
Region_Name
Store_No
Store_Name
Address
City
State
ZIP
Phone
Area_(SF)

Field:	Cash_Disb_ID	Store_No	Date	Account	Amount	Notes		
Table:	Cash_Disbursements	Cash_Disbursements	Cash_Disbursements	Cash_Disbursements	Cash_Disbursements	Cash_Disbursements		
Sort:								
Show:	☑	☑	☑	☑	☑	☑	□	
Criteria:								
or:								

Figure 3.7

Now you will create a join (or relationship) between the two tables on the Store_No field.

13. *Click on the* **Store_No** *field in the* **Cash_Disbursements** *table, drag it over to the* **Store_No** *field on the* **Stores** *table, and release.*



Query1

Cash_Disbursements
*
Cash_Disb_ID
Store_No
Date
Account
Amount
Notes

Stores
*
Store_ID
Region_No
Region_Name
Store_No
Store_Name
Address
City
State
ZIP
Phone
Area_(SF)

Field:	Cash_Disb_ID	Store_No	Date	Account	Amount	Notes		
Table:	Cash_Disbursements	Cash_Disbursements	Cash_Disbursements	Cash_Disbursements	Cash_Disbursements	Cash_Disbursements		
Sort:								
Show:	☑	☑	☑	☑	☑	☑	□	
Criteria:								
or:								

Figure 3.8

There is now a line connecting the two Store_No fields. You have just created your first join.

Congratulations! You have just fully stepped into the world of relational databases. That wasn't too painful, was it? These two tables contain different types of information, and relational databases are designed to store different data in separate tables. You then join the tables on one or more common field to extract the data you need. Remember that you can think of a join in Access like a VLOOKUP() in Excel on steroids.

In the 12 tables that make up the Nitey_Nite_2016 database, there are more than sixty fields of data, containing hundreds of thousands of records. It would be unreasonable to have one table that contains all of the data you could possibly need. If we input all of the Store data (Store_ID, Store_No, Store_Name, etc.) in the Cash_Disbursements table, there would be a lot of repeating information. Just for Store No 1001, there are 263 records of information in the Cash_Disbursements table, so you would have to repeat all of the Store data 263 times. A relational database management system like Access makes it possible and efficient to store the data in separate tables.

The most common type of relationship is called a one-to-many relationship because there is one unique value in one table which is tied to another table which stores many of that same value. Take for example the Stores and Cash_Disbursements tables. There is only one Store_No per store in the Stores table, but Store_No appears many times for each store in the Cash_Disbursements table. Other types of relationships are called one-to-one and many-to-many relationships. In a one-to-one relationship, each record in the first table has only one corresponding record in the second table. A one-to-one relationship between tables in not very common because one-to-one relationship data are typically stored in the same table. A many-to-many relationship is very uncommon and generally reflects poor database design, but sometimes it is necessary. That kind of relationship typically exists when two one-to-many relationships from separate tables are joined together through a third table, called a union table.

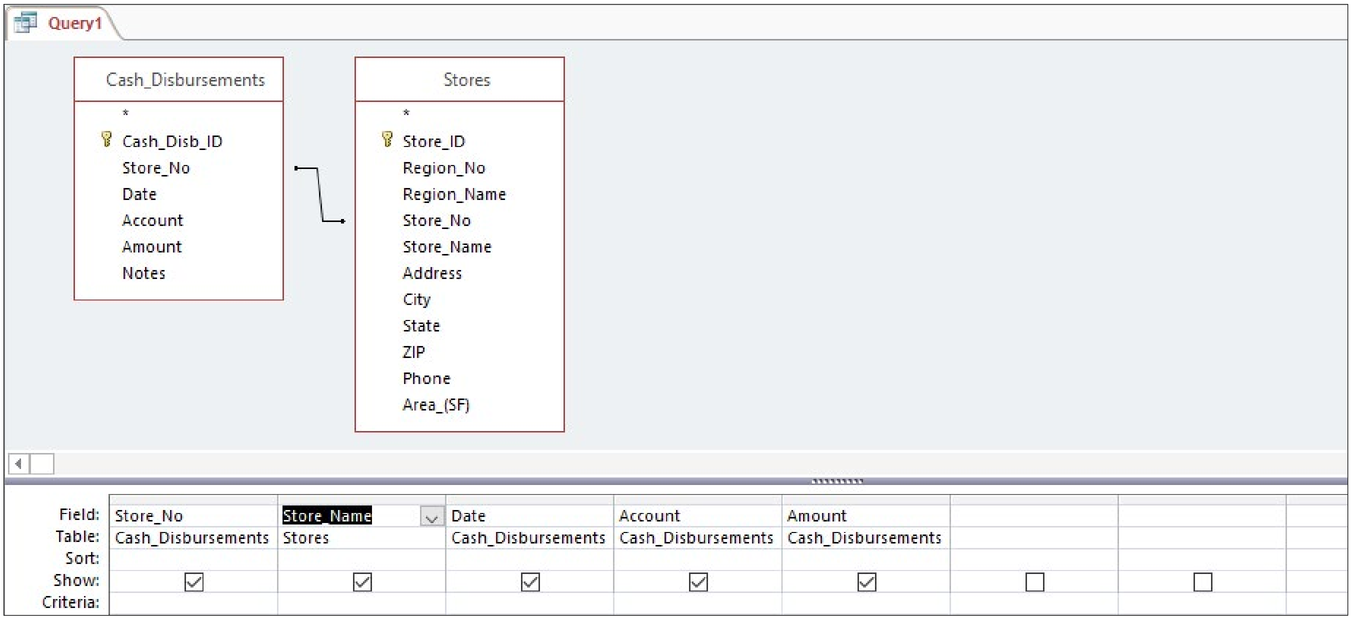
Another good argument for storing data in separate tables involves updating the data. Let's suppose the area code for the phone number at Store No. 1001 changed. That can happen quite often, particularly in large metropolitan areas. Particularly with the advent of cell phones, area codes are continuously added or changed. Using a join, you can update the area code in one table (the Stores table) and that is the only place where the information is contained. All queries that are joined to the Stores table would be automatically updated after making that one change. That is much better than changing it 263 times in the Cash_Disbursements table, isn't it?

Lastly, when data are repeated, there is more of a chance it could be entered differently. Take for example a table of Employee names. "Robert" could be entered in one place as "Robert", another place as "Bob" and yet another place as "Bobby". How would you like to keep track of Elizabeth Taylor's name each time she was married or divorced? If you store the employee's name in one table, you should create an ID for that record and use a join to pull in the name from the Employee table. That way, the name will always be consistent, even after you make a change to it.

The main point I want to stress in this discussion is that you should never have unique data contained in more than one table in your relational database. The only exception to that is perhaps a table that is used specifically for reporting purposes, and that is done only for speed. As we discussed in Chapter 2, each table should contain at least one field that can be used as a primary key, or a unique identifier, for each record. Sometimes you can use this primary key as the field on which to join to other tables. In the Nitey_Nite_2016 database, I've used primary keys in some tables, but not all. I did this because database designers who create tables sometimes don't always make them as efficient as they should be, and I wanted you to see that kind of "bad" ***database design*** so you can have the experience of working with it. Trust me — you will encounter bad database design numerous times in your career.

Now let's pull the necessary fields into the Design Grid and run it.

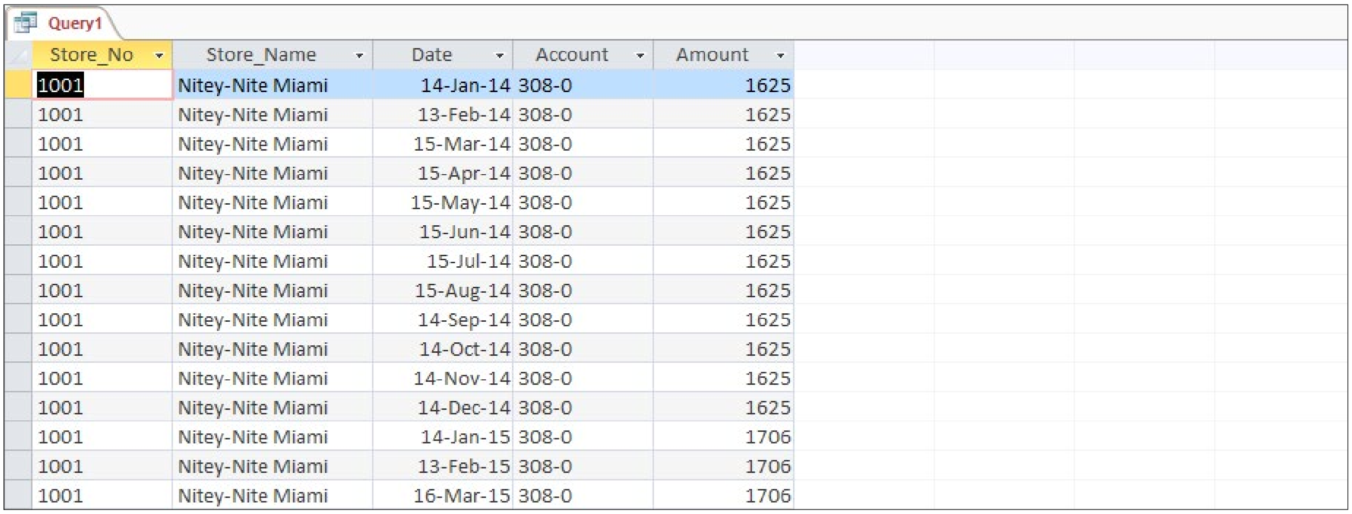
14. *In the* **Design Grid** *of* **Query1**, *delete the* **Cash_Disb_ID** *and* **Notes** *fields.*
15. *Click and drag the* **Store_Name** *field from the* **Stores** *table and place it on top of the* **Date** *field in the* **Design Grid** *and release.*



Field:	Store_No	Store_Name	Date	Account	Amount		
Table:	Cash_Disbursements	Stores	Cash_Disbursements	Cash_Disbursements	Cash_Disbursements		
Sort:							
Show:	☑	☑	☑	☑	☑	□	□
Criteria:							

Figure 3.9

16. *View the data in* **Datasheet View**.



Store_No	Store_Name	Date	Account	Amount
1001	Nitey-Nite Miami	14-Jan-14	308-0	1625
1001	Nitey-Nite Miami	13-Feb-14	308-0	1625
1001	Nitey-Nite Miami	15-Mar-14	308-0	1625
1001	Nitey-Nite Miami	15-Apr-14	308-0	1625
1001	Nitey-Nite Miami	15-May-14	308-0	1625
1001	Nitey-Nite Miami	15-Jun-14	308-0	1625
1001	Nitey-Nite Miami	15-Jul-14	308-0	1625
1001	Nitey-Nite Miami	15-Aug-14	308-0	1625
1001	Nitey-Nite Miami	14-Sep-14	308-0	1625
1001	Nitey-Nite Miami	14-Oct-14	308-0	1625
1001	Nitey-Nite Miami	14-Nov-14	308-0	1625
1001	Nitey-Nite Miami	14-Dec-14	308-0	1625
1001	Nitey-Nite Miami	14-Jan-15	308-0	1706
1001	Nitey-Nite Miami	13-Feb-15	308-0	1706
1001	Nitey-Nite Miami	16-Mar-15	308-0	1706

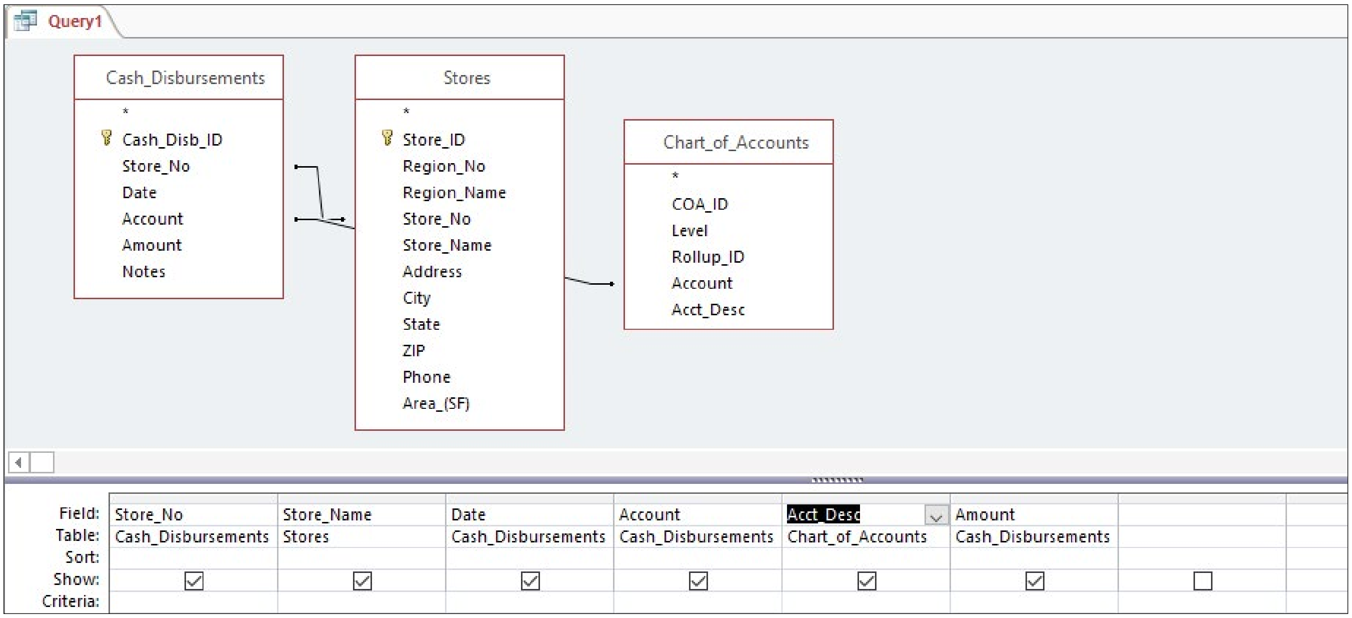
Figure 3.10

Since you created the relationship (or join) between the Cash_Disbursements table and the Stores table using the Store_ID, you can now query any information you want about stores in the same query that you built using the Cash_Disbursements table. This is a simple join. You will see that this query produces 8,409 records. If you open the Cash_Disbursements table, you'll see that table also contains 8,409 records. In a simple one-to-many join like we did, this is a very good check to do. If you get a different number of records than is in the "many" table, you need to check your query or the data in the tables.

Come to think of it, the Account number in the Cash_Disbursements table doesn't really tell us a lot either. The name of the account would be much more meaningful than just the account number by itself. All you have to do to accomplish that is to create another join to the table that lists the accounts and account names. That information is found in the Chart_ of_Accounts table.

17. *Go back to* **Design View** *of* **Query1**.
18. *Bring in the* **Chart_of_Accounts** *table to the* **Table Area** *of the* **Design Grid** *and adjust the margins so you can read all of the data.*
19. *Create a relationship between the* **Account** *field in the* **Cash_Disbursements** *table and the* **Account** *field in the* **Chart_of_Accounts** *table.*
20. *Bring the* **Acct_Desc** *field from the* **Chart_of_Accounts** *table and place it in between the* **Account** *and* **Amount** *fields in the* **Design Grid**.

The Design Grid should look like this:



Field:	Store_No	Store_Name	Date	Account	Acct_Desc	Amount	
Table:	Cash_Disbursements	Stores	Cash_Disbursements	Cash_Disbursements	Chart_of_Accounts	Cash_Disbursements	
Sort:							
Show:	☑	☑	☑	☑	☑	☑	□
Criteria:							

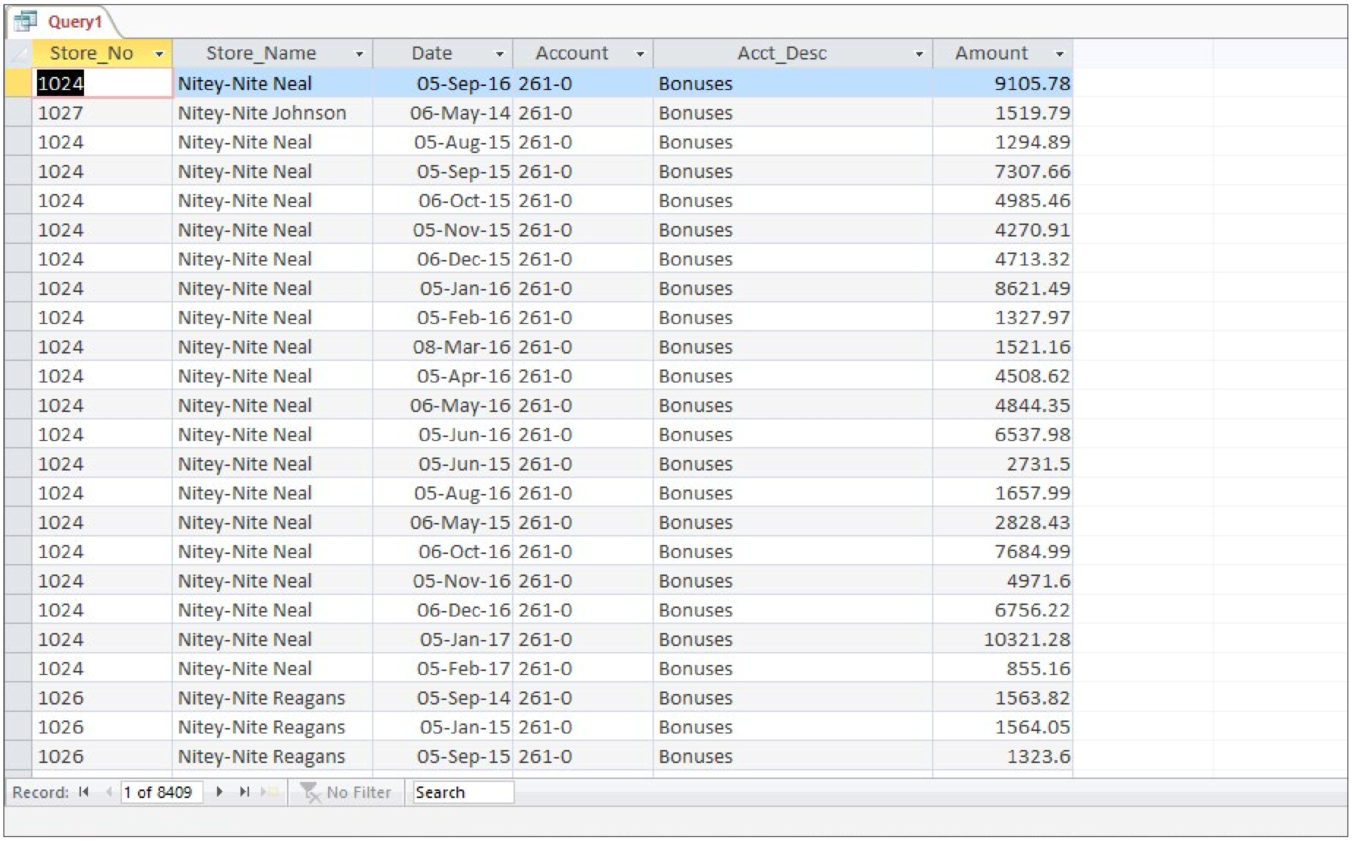
Figure 3.11

***Note**: The* **Chart_of_Accounts** *table was moved down to better display the relationship since all relationships shown are currently the fourth field down.*

Notice that the line that connects the Cash_Disbursements table with the Chart_of_Accounts table goes behind the Stores table. In Figure 3.11, it kind of looks like the account is connected to the Region_Name in the Stores table, but the line has points or dots that show which fields are actually connected.

You can create multiple joins on tables, as long as the fields between the tables correspond to one another. The number of joins you can create is limited to the number of fields contained in all the tables in the query, or 16, whichever is lower. But remember that creating too many joins can negatively affect performance. The point here is to make the tables and joins as efficient as possible. You should constantly test and retest your queries to make sure they are performing as efficiently as possible.

21. **Run** *the query.*



Store_No	Store_Name	Date	Account	Acct_Desc	Amount
1024	Nitey-Nite Neal	05-Sep-16	261-0	Bonuses	9105.78
1027	Nitey-Nite Johnson	06-May-14	261-0	Bonuses	1519.79
1024	Nitey-Nite Neal	05-Aug-15	261-0	Bonuses	1294.89
1024	Nitey-Nite Neal	05-Sep-15	261-0	Bonuses	7307.66
1024	Nitey-Nite Neal	06-Oct-15	261-0	Bonuses	4985.46
1024	Nitey-Nite Neal	05-Nov-15	261-0	Bonuses	4270.91
1024	Nitey-Nite Neal	06-Dec-15	261-0	Bonuses	4713.32
1024	Nitey-Nite Neal	05-Jan-16	261-0	Bonuses	8621.49
1024	Nitey-Nite Neal	05-Feb-16	261-0	Bonuses	1327.97
1024	Nitey-Nite Neal	08-Mar-16	261-0	Bonuses	1521.16
1024	Nitey-Nite Neal	05-Apr-16	261-0	Bonuses	4508.62
1024	Nitey-Nite Neal	06-May-16	261-0	Bonuses	4844.35
1024	Nitey-Nite Neal	05-Jun-16	261-0	Bonuses	6537.98
1024	Nitey-Nite Neal	05-Jun-15	261-0	Bonuses	2731.5
1024	Nitey-Nite Neal	05-Aug-16	261-0	Bonuses	1657.99
1024	Nitey-Nite Neal	06-May-15	261-0	Bonuses	2828.43
1024	Nitey-Nite Neal	06-Oct-16	261-0	Bonuses	7684.99
1024	Nitey-Nite Neal	05-Nov-16	261-0	Bonuses	4971.6
1024	Nitey-Nite Neal	06-Dec-16	261-0	Bonuses	6756.22
1024	Nitey-Nite Neal	05-Jan-17	261-0	Bonuses	10321.28
1024	Nitey-Nite Neal	05-Feb-17	261-0	Bonuses	855.16
1026	Nitey-Nite Reagans	05-Sep-14	261-0	Bonuses	1563.82
1026	Nitey-Nite Reagans	05-Jan-15	261-0	Bonuses	1564.05
1026	Nitey-Nite Reagans	05-Sep-15	261-0	Bonuses	1323.6

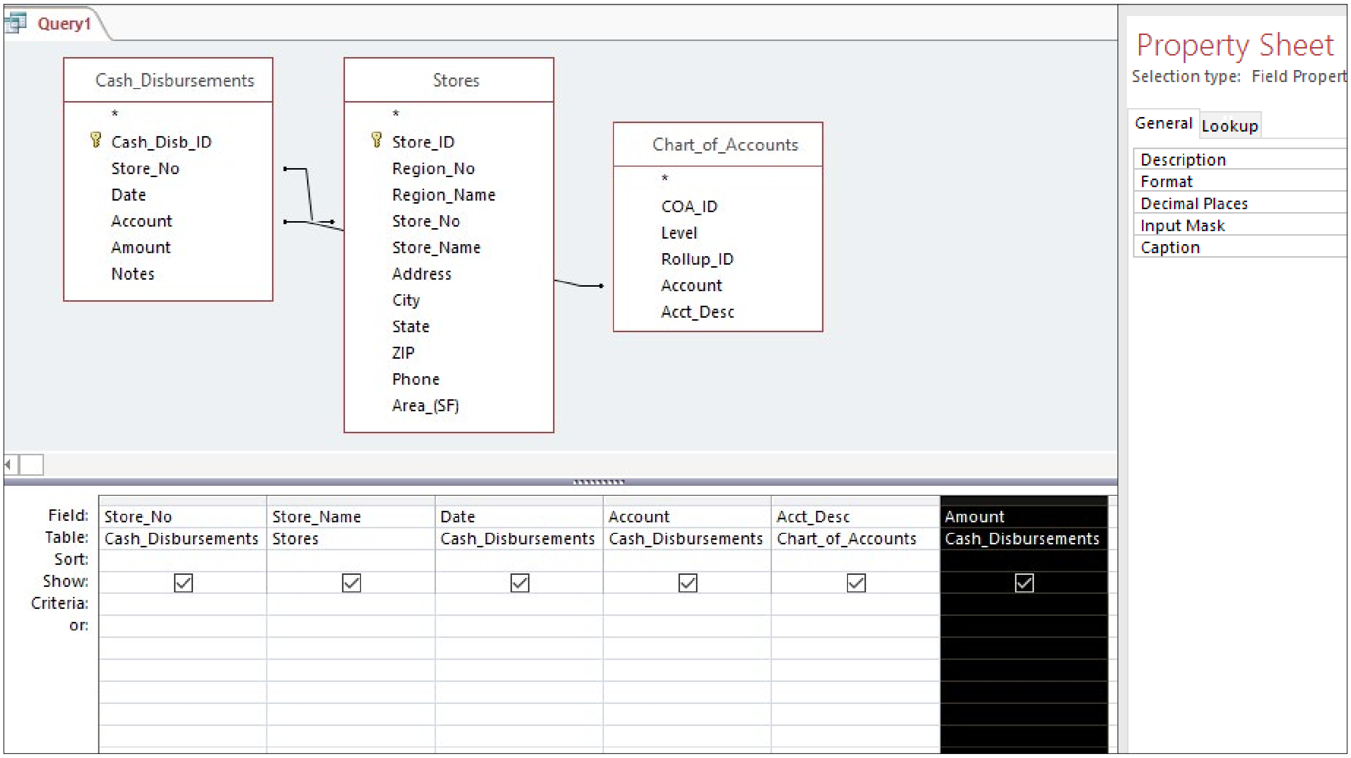
Figure 3.12

As you continue to add fields to the query, the order of your results may change, and depending how you've sorted tables in your database, your results may not look like the ones in the screenshots.

Formatting Query Fields

Now let's suppose you want to see all of the Cash_Disbursements transactions for Store No. 1021 in the month of June 2016. Additionally, the amount field is not formatted, and you would like to see it in a Number format with no decimal places. Let's do it.

22. *Return to* **Design View***.*
23. *Click on the* **Amount** *field and click the* **Property Sheet** Property Sheet *icon in* **Show/ Hide** *group of the* **Query Tools Design** *tab.*



Field:	Store_No	Store_Name	Date	Account	Acct_Desc	Amount
Table:	Cash_Disbursements	Stores	Cash_Disbursements	Cash_Disbursements	Chart_of_Accounts	Cash_Disbursements
Sort:						
Show:	☑	☑	☑	☑	☑	☑
Criteria:						
or:						

Property Sheet

Selection type: Field Propert

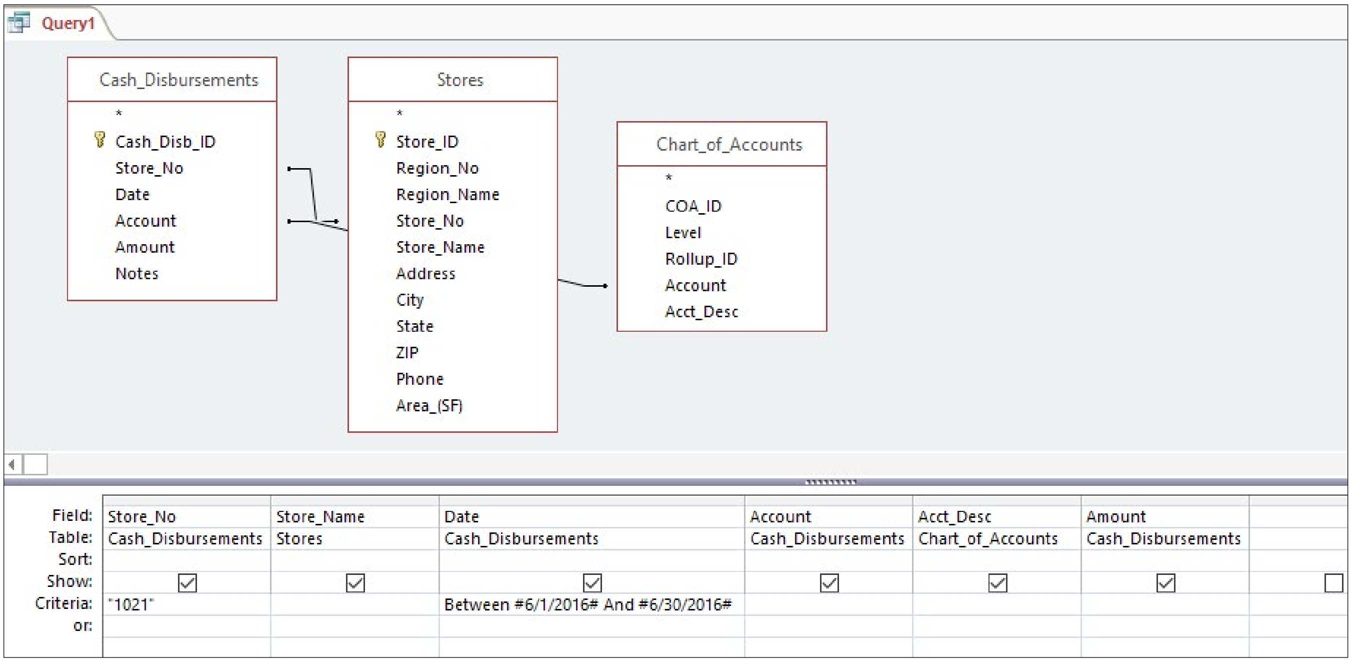
General Lookup

Description
Format
Decimal Places
Input Mask
Caption

Figure 3.13

The ***Property Sheet*** dialog box appears.

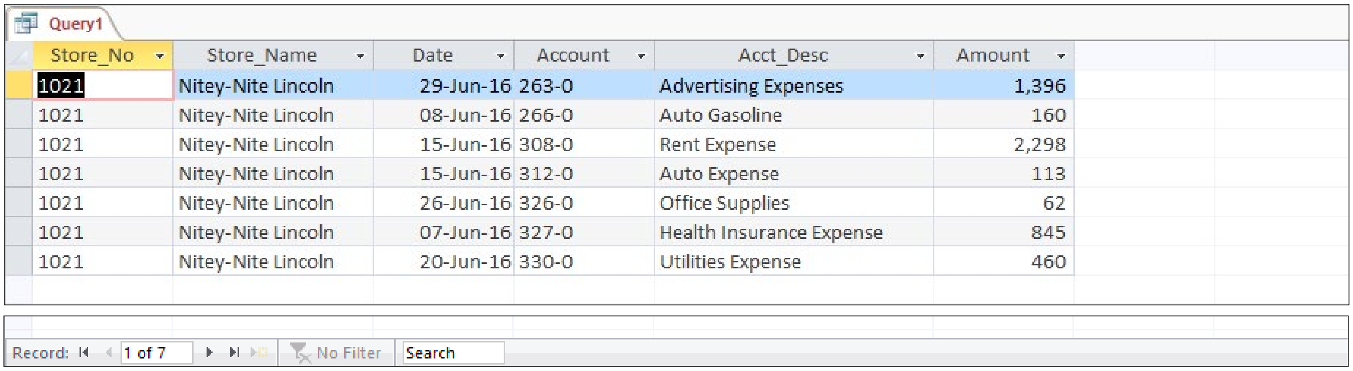
24. *Click on the* **Format** *line, and from the drop-down menu, choose* **Standard***.*
25. *In the* **Decimal Places** *box, type* ***0****.*
26. *Close the* **Property Sheet** *box and input a* **Criteria** *to filter for Store No.* ***1021*** *with dates* ***between 6/1/2016 and 6/30/2016****.*
27. *Adjust the margins of the* **Date** *field so you can see the criteria string.*



Field:	Store_No	Store_Name	Date	Account	Acct_Desc	Amount	
Table:	Cash_Disbursements	Stores	Cash_Disbursements	Cash_Disbursements	Chart_of_Accounts	Cash_Disbursements	
Sort:							
Show:	☑	☑	☑	☑	☑	☑	□
Criteria:	"1021"		Between #6/1/2016# And #6/30/2016#				
or:							

Figure 3.14

28. *View the record set in* **Datasheet View**.



Store_No	Store_Name	Date	Account	Acct_Desc	Amount
1021	Nitey-Nite Lincoln	29-Jun-16	263-0	Advertising Expenses	1,396
1021	Nitey-Nite Lincoln	08-Jun-16	266-0	Auto Gasoline	160
1021	Nitey-Nite Lincoln	15-Jun-16	308-0	Rent Expense	2,298
1021	Nitey-Nite Lincoln	15-Jun-16	312-0	Auto Expense	113
1021	Nitey-Nite Lincoln	26-Jun-16	326-0	Office Supplies	62
1021	Nitey-Nite Lincoln	07-Jun-16	327-0	Health Insurance Expense	845
1021	Nitey-Nite Lincoln	20-Jun-16	330-0	Utilities Expense	460

Figure 3.15

There should be seven records that appear in the query.

Naming an Alias

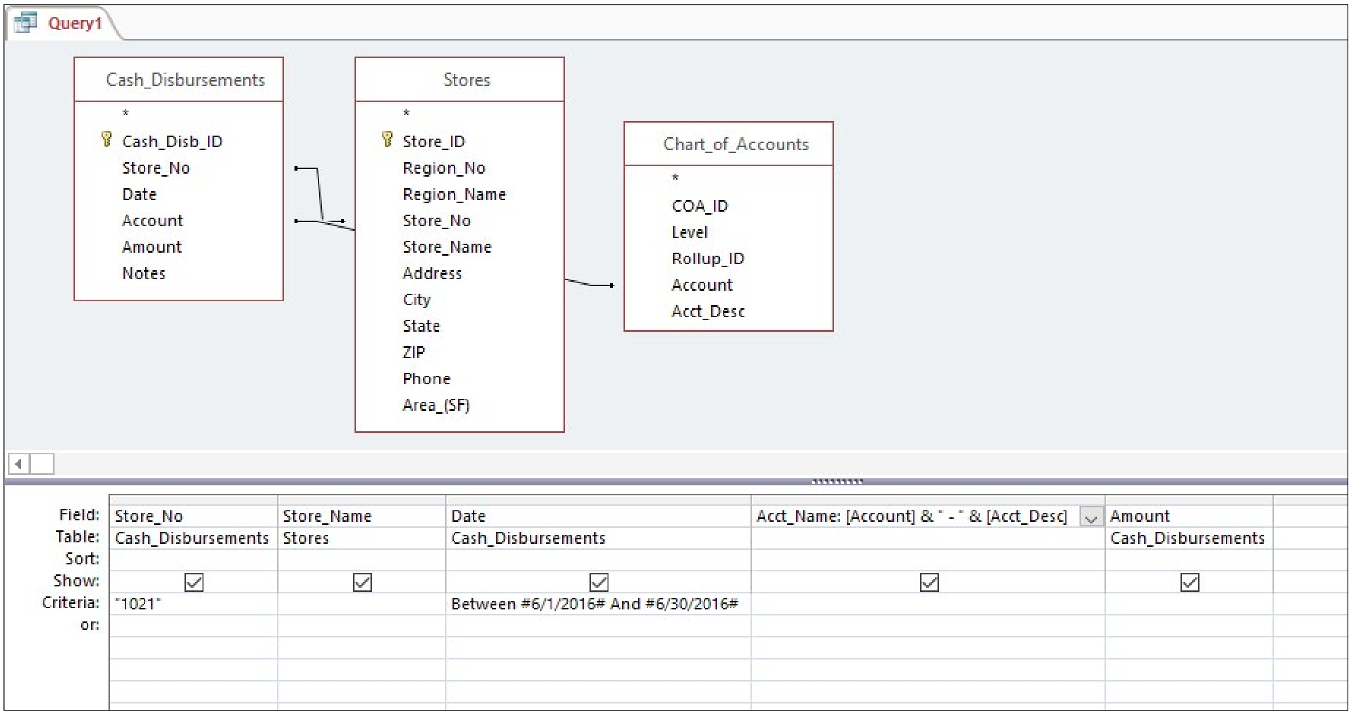
In Nitey-Nite's detailed financial statements, management likes to see the account and the name of the account in the same field, separated by a space, a dash, and another space. Just like in Excel, you can combine, or concatenate, fields in Access. When you do this in Access, you have to create a new name, or an ***Alias***, for the new field. You can also rename an existing field with an alias. One of the rules in creating an alias in Access is that you cannot use a name that already exists in the table(s) or query(ies) you are using. Since we will combine the Account and Acct_Desc fields into one field, let's call it Acct_Name. You do this by typing the new name of the field, then a colon followed by a space, and then the formula, or concatenation. Let’s do that now.

29. *Return to* **Design View***.*
30. *Insert a column before the* **Amount** *field (select the* **Amount** *field and click on the* **Insert Columns** *icon from the* **Query Tools Design** *tab.)*
31. *In the new field, type:* ***Acct_Name: Account&“ – ”&Acct_Desc****, then press* **[Enter]***.*

After you press [Enter], brackets appear around the fields account and acct_desc.

32. *Adjust the margins of the new field in* **Design View** *to where you can see the entire formula.*
33. *Delete the* **Account** *and* **Acct_Desc** *fields from the* **Design Grid***.*

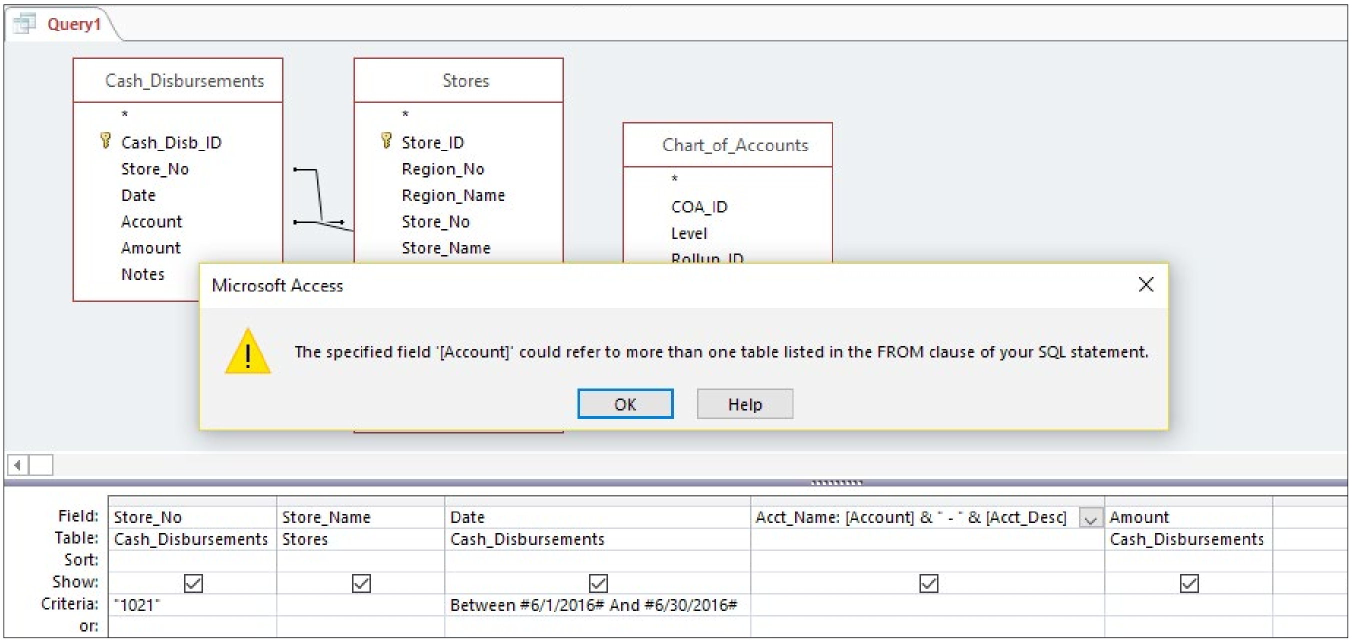
The Design Grid should look like this:



Field:	Store_No	Store_Name	Date	Acct_Name: [Account] & " - " & [Acct_Desc]	Amount	
Table:	Cash_Disbursements	Stores	Cash_Disbursements		Cash_Disbursements	
Sort:						
Show:	☑	☑	☑	☑	☑	
Criteria:	"1021"		Between #6/1/2016# And #6/30/2016#			
or:						

Figure 3.16

34. *Run the query in* **Datasheet View***.*



The specified field '[Account]' could refer to more than one table listed in the FROM clause of your SQL statement.

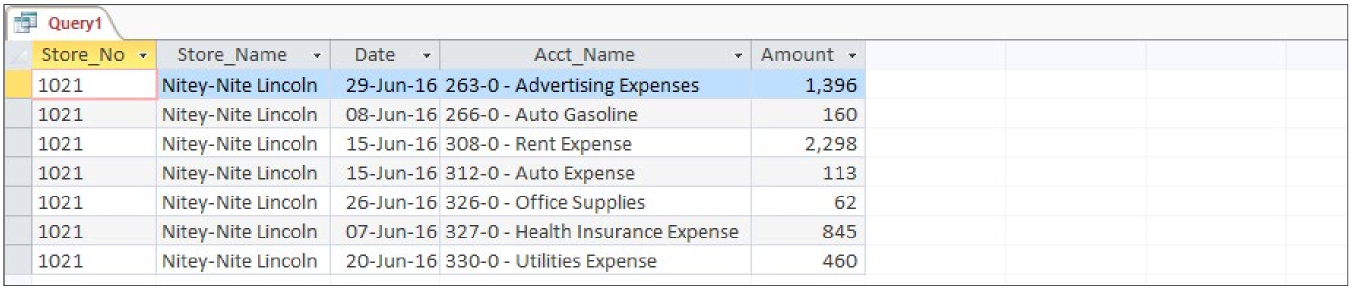
Field:	Store_No	Store_Name	Date	Acct_Name: [Account] & " - " & [Acct_Desc]	Amount
Table:	Cash_Disbursements	Stores	Cash_Disbursements		Cash_Disbursements
Sort:					
Show:	☑	☑	☑	☑	☑
Criteria:	"1021"		Between #6/1/2016# And #6/30/2016#		
or:					

Figure 3.17

Whoa-ho! What is that? This is an error message that's telling you that there is more than one table in your query that has a field named "Account" in it. If you look at the tables you're using, you'll see that the Chart_of_Accounts and the Cash_Disbursements tables both have a field named "Account". When you tried to run the query, Access didn't know which "Account" to use. To correct it, we need to specify which table we want to pull the account field from. That is accomplished by typing the name of the table before the name of the field, separated by a period. This is called the field's ***path***. In this case, it doesn't matter which table it comes from, so we'll just use the Chart_of_Accounts table.

35. *Click* **OK** *to return to* **Design View**.
36. *Edit the* **Acct_Name** *formula to look like this:*

    Acct_Name: [Chart_of_Accounts.Account] & " - " & [Acct_Desc]
37. **Run** *the query.*
38. *Adjust columns as needed.*



Store_No	Store_Name	Date	Acct_Name	Amount
1021	Nitey-Nite Lincoln	29-Jun-16	263-0 - Advertising Expenses	1,396
1021	Nitey-Nite Lincoln	08-Jun-16	266-0 - Auto Gasoline	160
1021	Nitey-Nite Lincoln	15-Jun-16	308-0 - Rent Expense	2,298
1021	Nitey-Nite Lincoln	15-Jun-16	312-0 - Auto Expense	113
1021	Nitey-Nite Lincoln	26-Jun-16	326-0 - Office Supplies	62
1021	Nitey-Nite Lincoln	07-Jun-16	327-0 - Health Insurance Expense	845
1021	Nitey-Nite Lincoln	20-Jun-16	330-0 - Utilities Expense	460

Figure 3.18

You will probably have to resize the Acct_Name column for the entire field to appear.

39. *Save the query as* ***qry03Cash_Disb*** *and close the query.*

After you close the query, your manager tells you all amount fields should appear with two decimal places. Since you've already built the query, changing it is easy.

40. *In the* **Queries** *section of the* **Navigation Pane**, *right-click* **qry03Cash_Disb** *and choose* **Design View**.

You return to the Design View of the query.

41. *Click on the* **Amount** *field, click on the* **Property Sheet** *icon and change the* **Decimal Places** *to* **2**.
42. *Close the* **Property Sheet** *dialog box.*
43. **Run** *the query to make sure the* **Amount** *field appears with two decimal places.*
44. **Save** *and close the* **qry03Cash_Disb** *query.*

> ***Review Questions****: It is now time to complete the hands-on Review Questions. Log on to www.ExcelCEO.com with your Email and Password, click on the* **Access 2016 and SQL Review Questions, Chapter 3, Section 1 of 2** *option and complete the review questions.*

Relationships View

Another great tool you can use, which could help facilitate query development and creating joins, is the Relationships View. In the Relationships view, you pre-determine any relationship you want instead of creating the joins on the fly in each query. Let's do an example using the Stores and General_Ledger tables. Whenever you use those two tables, the Store_ID in the General_Ledger table will ALWAYS tie to the Store_ID in the Stores table. When you create that permanent relationship in the Relationships View, the join will automatically be created whenever you use those same two tables in any query. Let's do an example now.

1. *Click on the* **Database Tools** *tab and click on the* **Relationships** Relationships *icon.*
2. *If the Show Table dialog box doesn't appear, click on the* **Show Table** *icon.*
3. *Add the* **Stores** *and* **General_Ledger** *tables and click* **Close**.
4. *Adjust the table margins appropriately.*

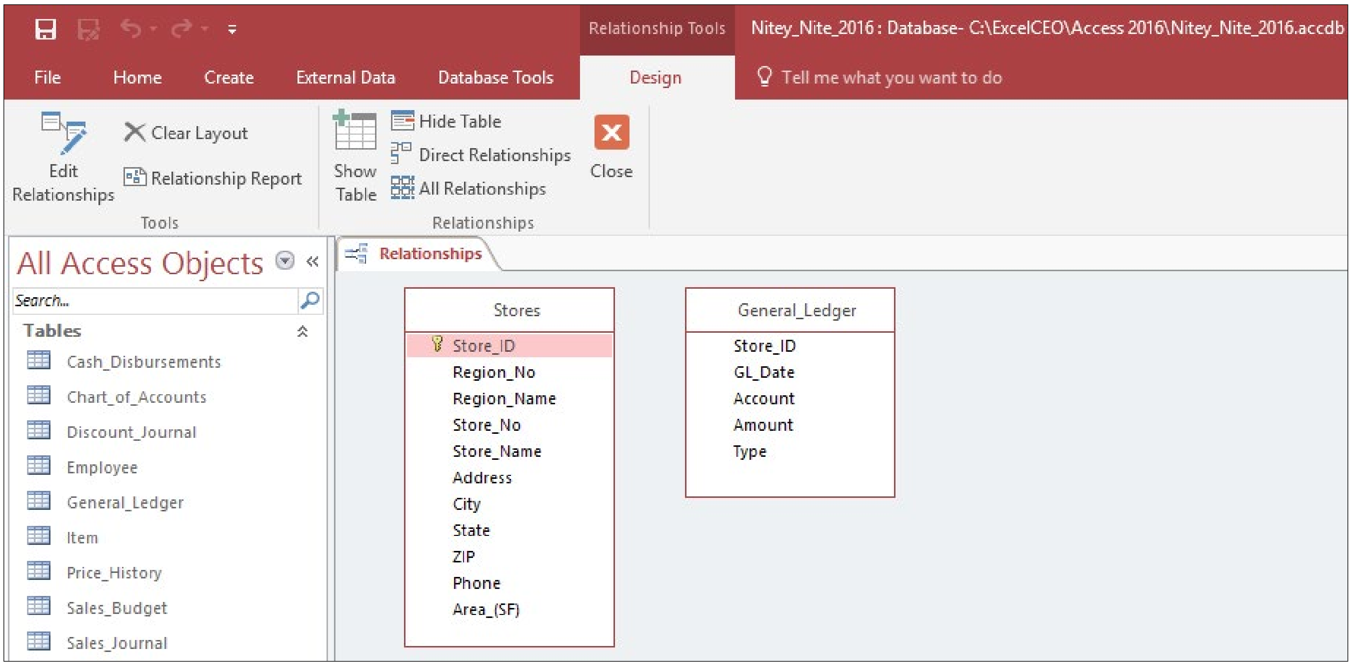
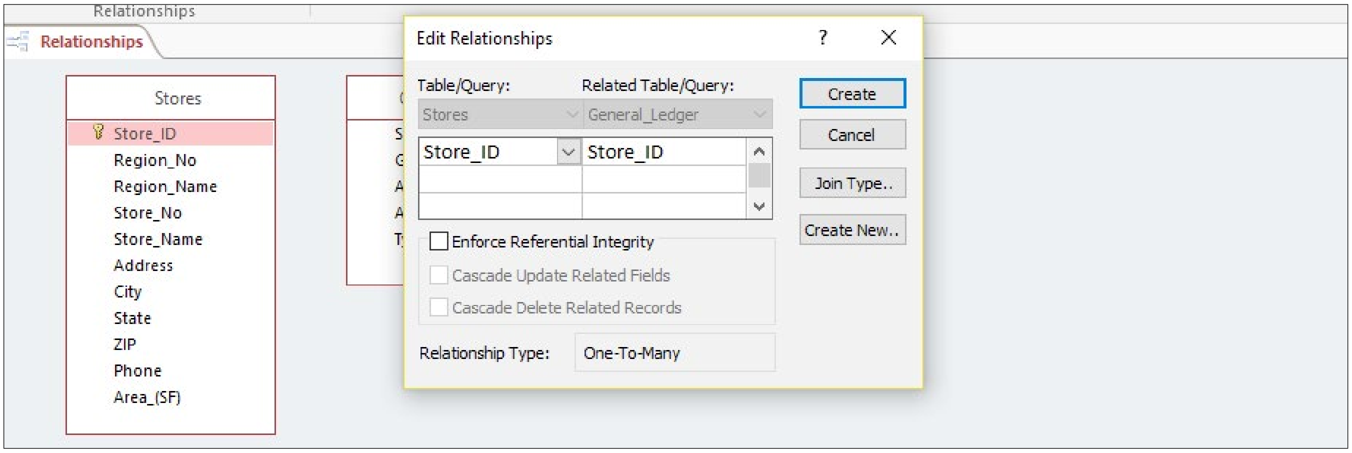


Figure 3.19

The Stores and General_Ledger tables appear just like they do in a query Design View. Now all you do is create a relationship just like you did in Design View.

5. *Click on the* **Store_ID** *field in the* **Stores** *table and drag it over to the* **Store_ID** *in the* **General_Ledger** *table.*



Table/Query: Stores

Related Table/Query: General_Ledger

Store_ID | Store_ID

□ Enforce Referential Integrity

□ Cascade Update Related Fields

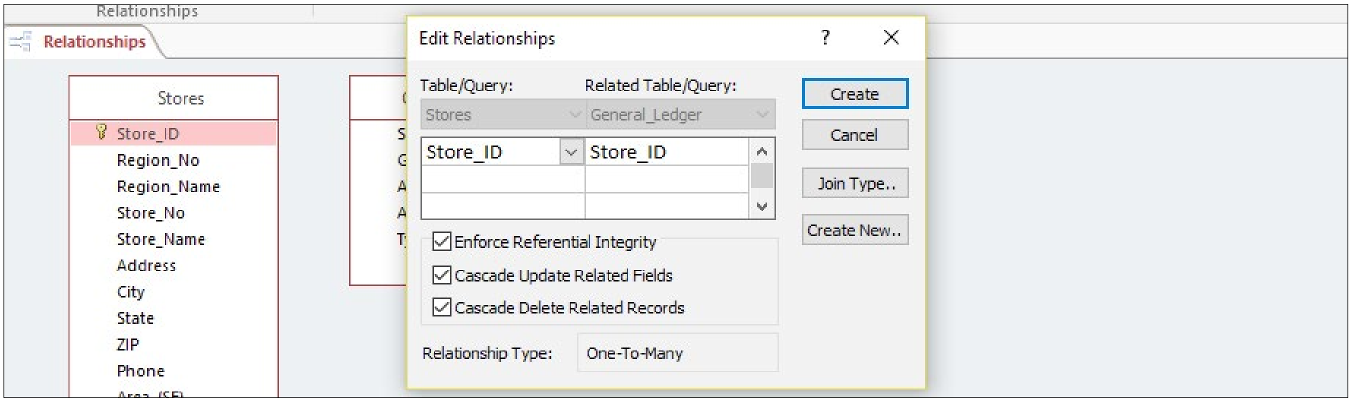
□ Cascade Delete Related Records

Relationship Type: One-To-Many

Edit Relationships dialog box

Once you release your mouse, the ***Edit Relationships*** dialog box appears. In this dialog box, you create the type of relationship you want between the two tables. At the bottom of the dialog box, it shows you that the relationship type is a one-to-many, as the Store_ID in the Stores table is the primary key, and there are many corresponding values in the General_Ledger table. You can also select "***Enforce Referential Integrity***". Selecting referential integrity ensures that the relationships between the records in the two tables are valid, and helps to make sure you don't delete or modify related data. Think of how you could mess up a query if you changed the Store_ID in the Stores table and didn't change it in the General_Ledger table. When you check Cascade Update Related Fields (after you check the Enforce Referential Integrity) box, changing a Primary Key value in the primary table will automatically update all related records in the other table. When Cascade Delete Related Records is checked, deleting a record in the primary table will also delete all records in the related table. Both of these functionalities help to ensure the database is kept intact. It's a good idea to check all of these boxes if you want to keep your database with the related records.

6. *Check the* **Enforce Referential Integrity**, *the* **Cascade Update Related Fields** *and* **Cascade Delete Related Fields** *boxes, then click* **Create**.



Table/Query: Stores
Related Table/Query: General_Ledger

Table/Query	Related Table/Query
Store_ID	Store_ID

☑ Enforce Referential Integrity
☑ Cascade Update Related Fields
☑ Cascade Delete Related Records

Relationship Type: One-To-Many

Figure 3.20

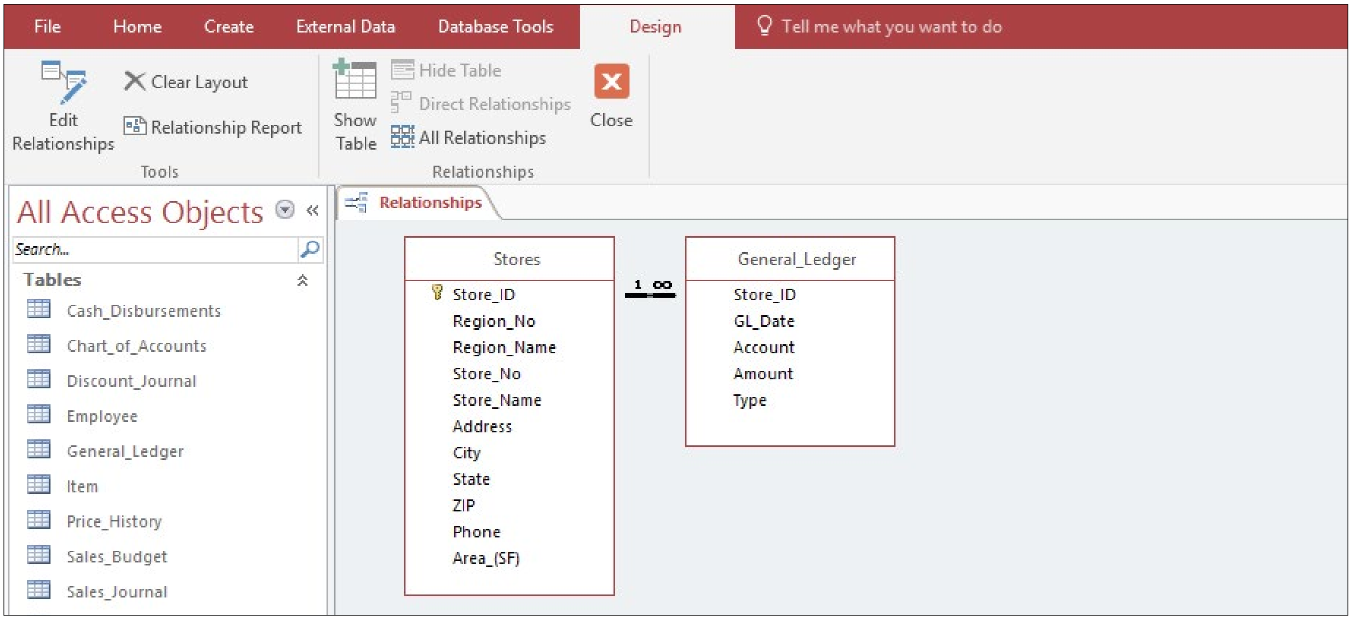


Figure 3.21

A line connects the Store_ID fields in both tables. There is a small "1" above the connection point at the Stores table (signifying the one in one-to-many), and an infinity sign (meaning many records) at the connection point to the General_Ledger table. Now whenever you bring the Stores and General_Ledger tables into the Design view of a query, this relationship will automatically be created.

7. *Close* ☒ *the* **Relationships** *window, and choose* **Yes** *to save changes.*

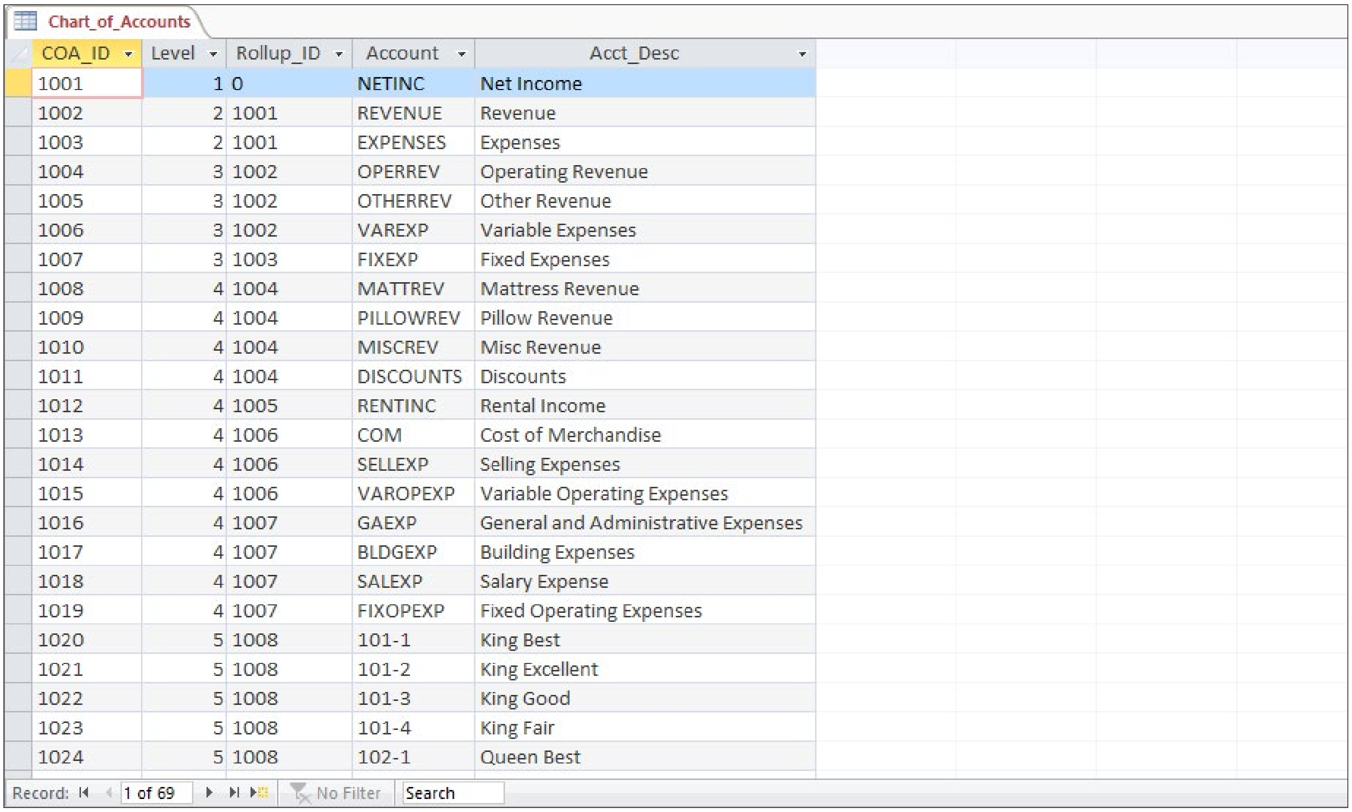
> ***Tip****: If you check the* **Stores** *table, a* **Plus** *icon appears to the left of the records to give you drill-down capability because of the relationship you just established.*

Flat Files vs. Hierarchical Tables

The next topic we will discuss is the difference between a ***flat file*** and a ***hierarchical table***. Up to this point in this Access course, you have been working with flat files. In a flat file, records are maintained in rows, and fields are contained in columns. The more levels at which you can roll up the data, the more fields or columns you have. Take, for example, a theoretical chart of accounts. At the lowest level, an expense account (say Telephone Expense) may roll up to a subcategory of expenses (Office Expenses), and that subcategory may roll up to another subcategory (General and Administrative Expenses), which may roll up to Fixed Expenses, which could roll up to Total Expenses, and then to Net Income. In that example, you would have six columns, one for each level. In a hierarchical table, there are two fields that contain all rollup data: a ***parent field*** and a ***child field*** (the child is under, or rolls up to, the parent). The rollups of the accounts are maintained in the ***parent/child relationships*** within the table.

To illustrate, let's open the Chart_of_Accounts table.

1. *Open the* **Chart_of_Accounts** *table and adjust all margins.*



Chart_of_Accounts

COA_ID	Level	Rollup_ID	Account	Acct_Desc
1001	1	0	NETINC	Net Income
1002	2	1001	REVENUE	Revenue
1003	2	1001	EXPENSES	Expenses
1004	3	1002	OPERREV	Operating Revenue
1005	3	1002	OTHERREV	Other Revenue
1006	3	1002	VAREXP	Variable Expenses
1007	3	1003	FIXEXP	Fixed Expenses
1008	4	1004	MATTREV	Mattress Revenue
1009	4	1004	PILLOWREV	Pillow Revenue
1010	4	1004	MISCREV	Misc Revenue
1011	4	1004	DISCOUNTS	Discounts
1012	4	1005	RENTINC	Rental Income
1013	4	1006	COM	Cost of Merchandise
1014	4	1006	SELLEXP	Selling Expenses
1015	4	1006	VAROPEXP	Variable Operating Expenses
1016	4	1007	GAEXP	General and Administrative Expenses
1017	4	1007	BLDGEXP	Building Expenses
1018	4	1007	SALEXP	Salary Expense
1019	4	1007	FIXOPEXP	Fixed Operating Expenses
1020	5	1008	101-1	King Best
1021	5	1008	101-2	King Excellent
1022	5	1008	101-3	King Good
1023	5	1008	101-4	King Fair
1024	5	1008	102-1	Queen Best

Figure 3.22

In the Chart_of_Accounts table, there are five fields. The COA_ID (COA is the acronym for Chart of Accounts) field is the unique identifier, or primary key field. From the very bottom level (the account level) to the highest level (Net Income), there are five levels. You see this in the Level field. The accountants at Nitey-Nite wanted to keep the Chart of Accounts very simple.

Often times, the ***Chart of Accounts*** table is simply a listing of each general ledger account and its corresponding name or description, such as 261-0, Bonuses. In this course, we combine the account, account name, and the rollup of the accounts to their respective categories (like Fixed Expenses or Revenue) to be able to generate financial statements. Many companies have a very complex chart-of-accounts rollup structure, but Nitey-Nite's structure is relatively simple.

To help understand the concept of a hierarchical table, look at Account 101-2, King Excellent. Its Level is 5, the lowest level, so this is a child account that no other account rolls up to. Its parent account, or the account it rolls up to, is indicated in the Roll_Up_ID field — it rolls up to Rollup_ID 1008. To see what Rollup_ID 1008 is, look up 1008 in the COA_ID field, and you can see that it is Mattress Revenue. COA_ID 1008 has a Rollup_ID of 1004, Operating Revenue, which rolls up to Revenue, which finally rolls up to the top level, Net Income.

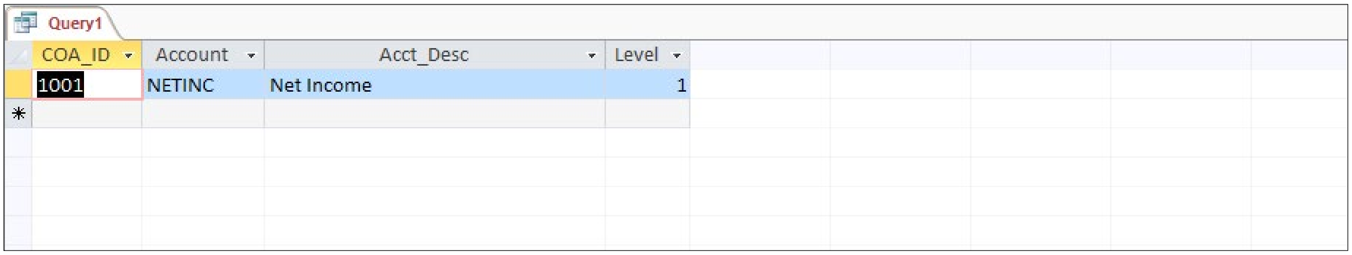
All Accounts in the Chart_of_Accounts table roll up to one higher level, except for the top level, Net Income (Level 1). The Level field is provided as a convenience to show the user which level the account or category is on, and we will use it in the next exercise. It is critically important to understand this type of table, as it is very common in database systems today.

Hierarchical tables generally have better performance and take up less space than flat files. However, it is often easier for financial/accounting people like us to understand and work with the chart-of-account rollup type of data in flat files rather than in hierarchical files. With a query, we can turn the hierarchical table into a flat file without tampering with the structure of the hierarchical table. Since a query is a virtual table which is based on a table or query, any change in the base table will be automatically reflected in the query. To turn this hierarchical table into an understandable flat file, what we need is a query that incorporates the account ID (COA_ID), the Account, and its description (Acct_Desc) for all levels in separate fields. To do that, we will have to make the table relate to itself five times, once for each level. This is a complex concept to understand, so make sure you repeat it enough times until you understand it thoroughly. Let's build the query.

2. *Close the* **Chart_of_Accounts** *table (save the changes if you adjusted column widths) and create a new query.*
3. *Bring in the* **Chart_of_Accounts** *table.*
4. *Bring the* **COA_ID**, **Account**, **Acct_Desc**, *and* **Level** *fields into the* **Design Grid**.

The highest level of accounts, Net Income, is the first level, so let's start with that level.

5. *In the* **Criteria** *section for the* **Level** *field, type* ***1***.
6. **Run** *the query.*

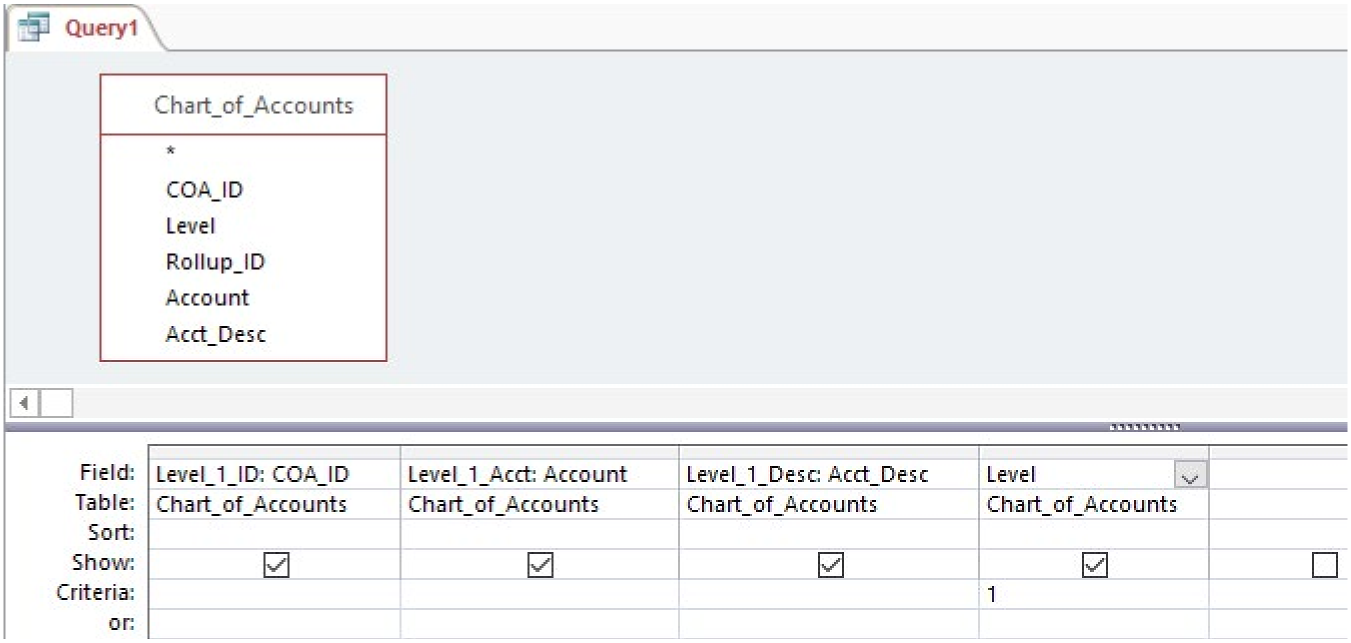


COA_ID	Account	Acct_Desc	Level
1001	NETINC	Net Income	1

Figure 3.23

This query should return only one record. We will be bringing in the same information for all five levels, so we need to use a consistent naming convention for each level.

7. *Return to* **Design View**.
8. *Create the following aliases for each field in the query:*
   Level_1_ID: **COA_ID**
   Level_1_Acct: **Account**
   Level_1_Desc: **Acct_Desc**



Chart_of_Accounts

- *
- COA_ID
- Level
- Rollup_ID
- Account
- Acct_Desc

Field:	Level_1_ID: COA_ID	Level_1_Acct: Account	Level_1_Desc: Acct_Desc	Level	
Table:	Chart_of_Accounts	Chart_of_Accounts	Chart_of_Accounts	Chart_of_Accounts	
Sort:					
Show:	☑	☑	☑	☑	☐
Criteria:				1	
or:					

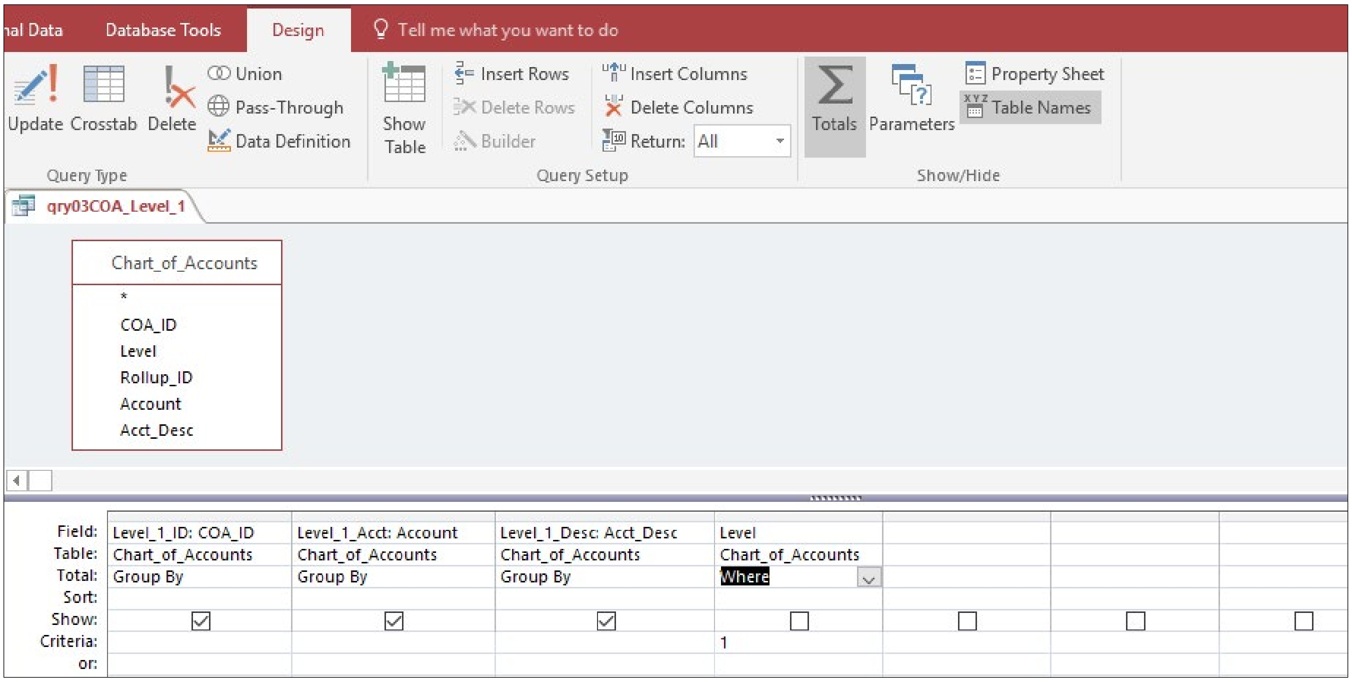
Graphical Representation of Step 8

With these descriptions for each field, it won't be necessary to show the Level number, but we still need to filter for it. To filter for criteria in a field that we don't need to show in the results, you can choose Where from the Total line, instead of using a Group By. You should learn when to use a Where clause very well, as it will be critically important in your programming education. In this next exercise, you will use the Totals icon. For now, I'll show you how to use it with a Where clause. We will discuss more about grouping and the Totals icon in the next chapter.

9. *Click on the* **Totals** Σ Totals *icon in the* **Query Tools Design** *tab.*

A new line appears beneath the Table line called Total.

10. *Choose* **Where** *from the drop-down menu on the* **Total:** *line for the* **Level** *field.*
11. **Save** *the query and name it* ***qry03COA_Level_1***.

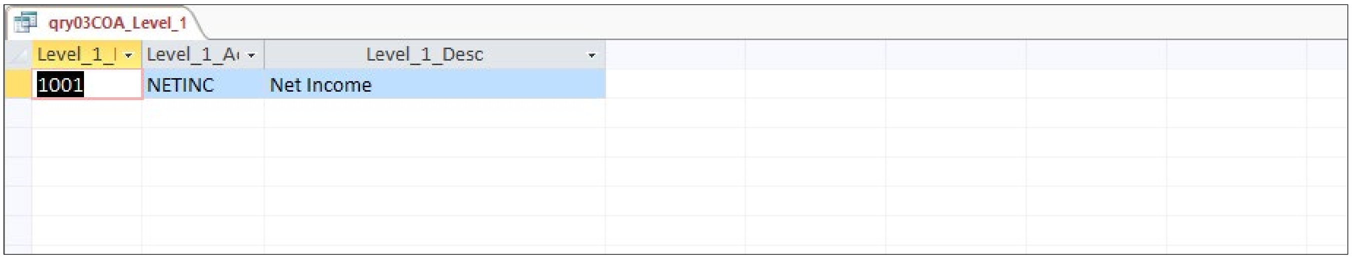


Field:	Level_1_ID: COA_ID	Level_1_Acct: Account	Level_1_Desc: Acct_Desc	Level			
Table:	Chart_of_Accounts	Chart_of_Accounts	Chart_of_Accounts	Chart_of_Accounts			
Total:	Group By	Group By	Group By	Where			
Sort:							
Show:	☑	☑	☑	☐	☐	☐	☐
Criteria:				1			
or:							

Figure 3.24

Note that the Show: box below the Level field is unchecked. That's because we changed it to be have a Where grouping, and the default is to have the Show box for a Where clause unchecked. If you want to show it, simply check the Show box.

12. **Run** *the query.*



Level_1_I	Level_1_A	Level_1_Desc
1001	NETINC	Net Income

Figure 3.25

Joins on Multiple Tables

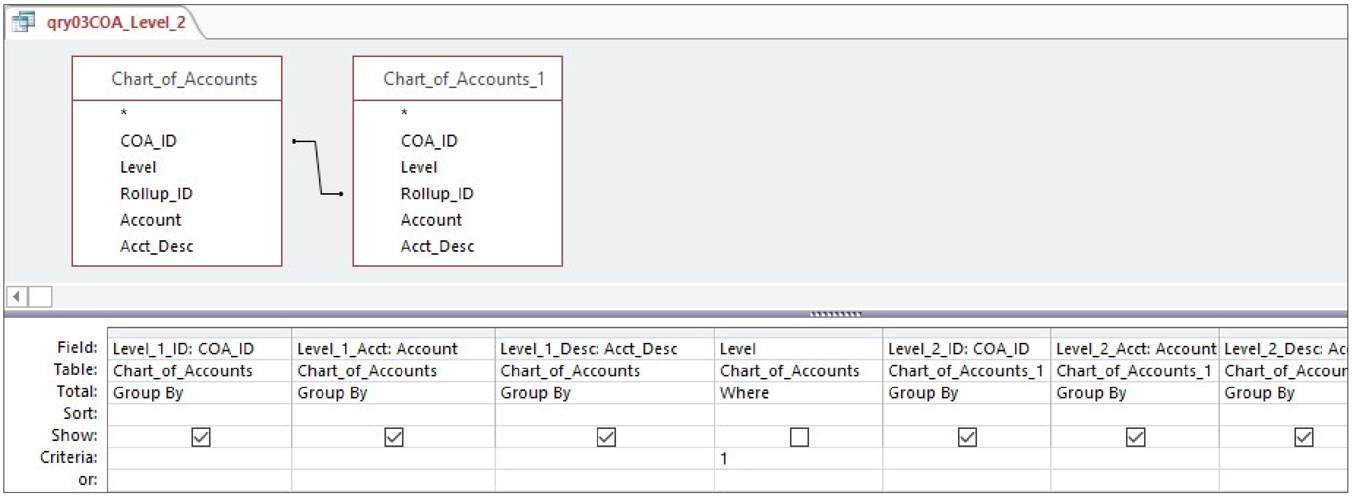
Now we'll bring in the next level of accounts. If you examine the Chart_of_Accounts table, you'll see that Net Income comprises Revenue (COA_ID 1002) and Expenses (COA_ID 1003). Try to stay with me on this part. It's difficult for some people to grasp this concept, but you HAVE to understand it to work with hierarchical tables.

13. Return to **Design View** *of the query.*

14. *Click the* **Show Table** *icon and bring in the* **Chart_of_Accounts** *table again.*

Note that the name of the new Chart_of_Accounts table is Chart_of_Accounts_1, to distinguish it from the first table you brought in, even though they are the same table being used twice.

15. *Close the* **Show Table** *window.*
16. *Create a join between* **COA_ID** *in the* **Chart_of_Accounts** *table and* **Rollup_ID** *in the* **Chart_of_Accounts_1** *table.*
17. *Bring in the* **COA_ID**, **Account**, **Acct_Desc**, *and* **Level** *fields from the* **Chart_of_Accounts_1** *table.*
18. *Create the same aliases for these fields as you did for* **Level_1**, *but name them* ***Level_2_ID:***, ***Level_2_Acct:*** *… and so forth.*
19. *Use a criteria of* **2** *in the* **Level** *field using* **Chart_of_Accounts_1** *and make it a* **Where** *field.*
20. *Save the query and name it* ***qry03COA_Level_2***

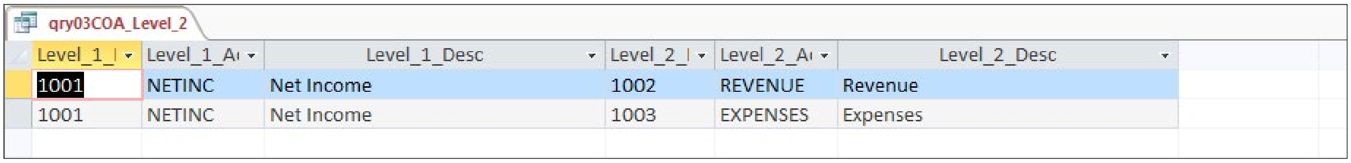


Field:	Level_1_ID: COA_ID	Level_1_Acct: Account	Level_1_Desc: Acct_Desc	Level	Level_2_ID: COA_ID	Level_2_Acct: Account	Level_2_Desc: Ac
Table:	Chart_of_Accounts	Chart_of_Accounts	Chart_of_Accounts	Chart_of_Accounts	Chart_of_Accounts_1	Chart_of_Accounts_1	Chart_of_Accour
Total:	Group By	Group By	Group By	Where	Group By	Group By	Group By
Sort:							
Show:	☑	☑	☑	☐	☑	☑	☑
Criteria:				1			
or:							

Figure 3.26

In Figure 3.26, keep in mind there are fields to the left of the Level 1 criteria.

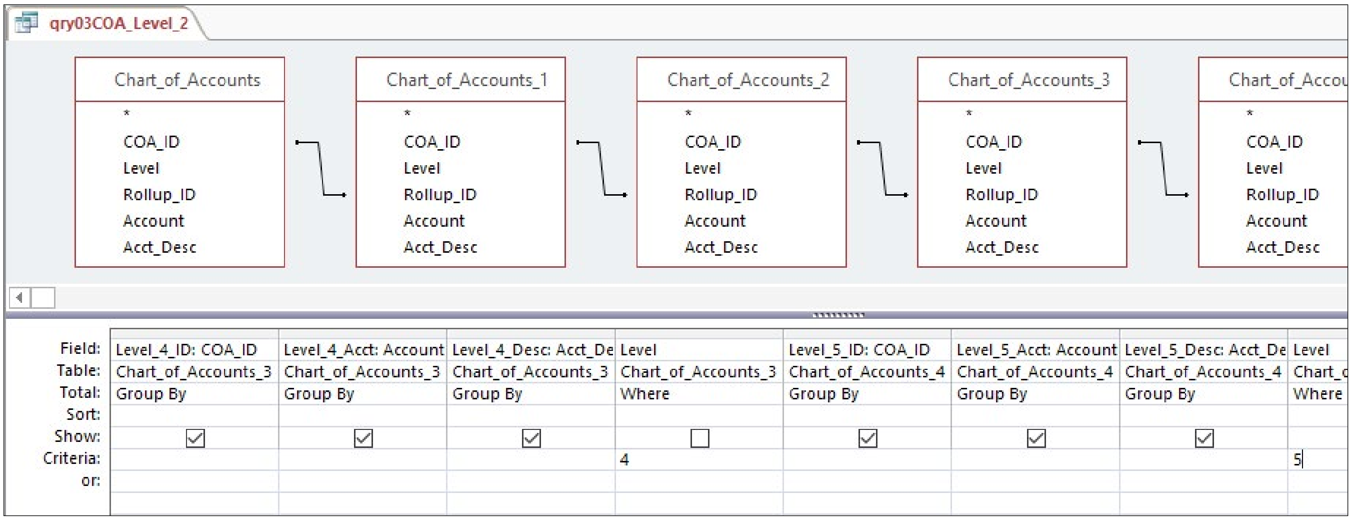
21. **Run** *the query. Resize the grid, if needed.*



Level_1_I	Level_1_A	Level_1_Desc	Level_2_I	Level_2_A	Level_2_Desc
1001	NETINC	Net Income	1002	REVENUE	Revenue
1001	NETINC	Net Income	1003	EXPENSES	Expenses

Figure 3.27

22. *Repeat the same process for* **Levels 3**, **4**, *and* **5**.

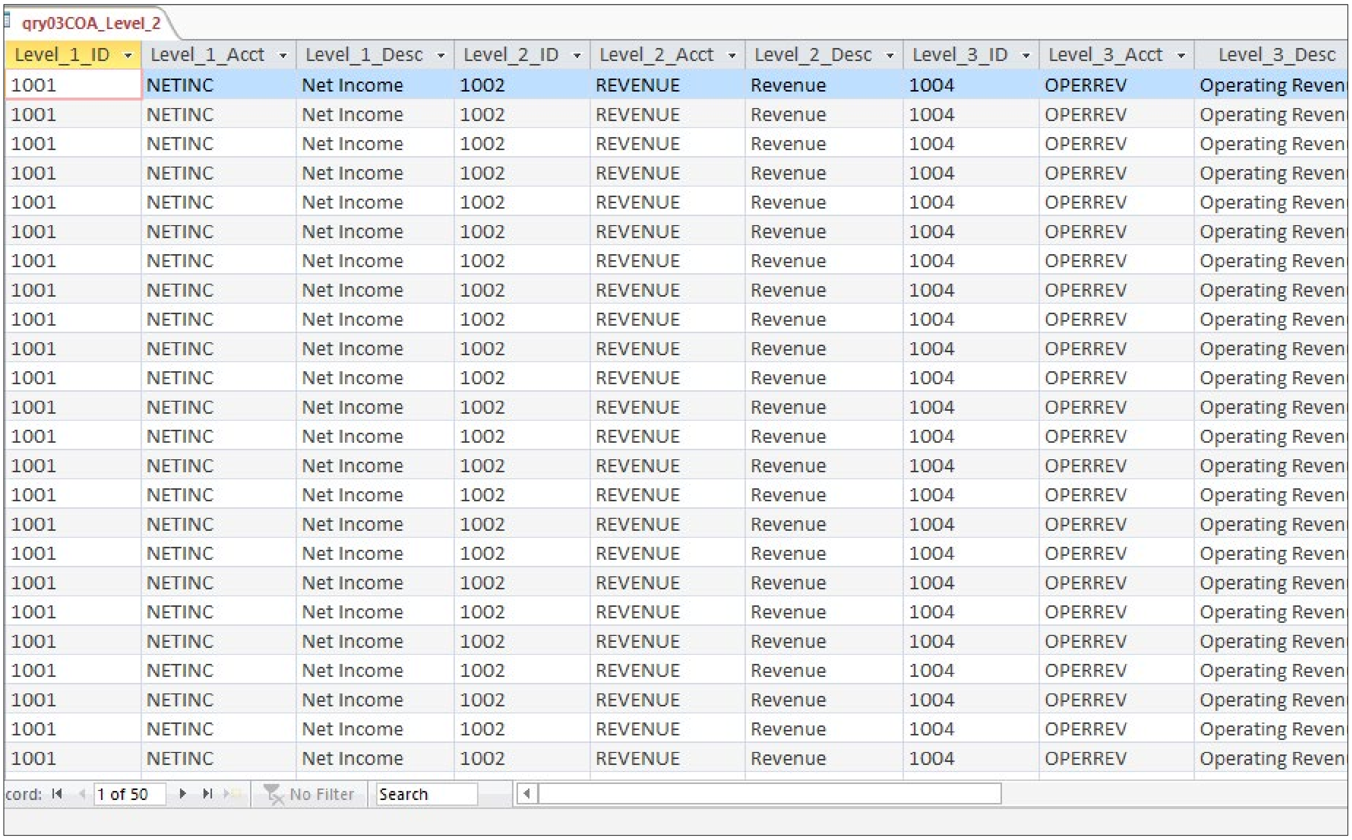


Field:	Level_4_ID: COA_ID	Level_4_Acct: Account	Level_4_Desc: Acct_De	Level	Level_5_ID: COA_ID	Level_5_Acct: Account	Level_5_Desc: Acct_De	Level
Table:	Chart_of_Accounts_3	Chart_of_Accounts_3	Chart_of_Accounts_3	Chart_of_Accounts_3	Chart_of_Accounts_4	Chart_of_Accounts_4	Chart_of_Accounts_4	Chart_c
Total:	Group By	Group By	Group By	Where	Group By	Group By	Group By	Where
Sort:								
Show:	☑	☑	☑	☐	☑	☑	☑	
Criteria:				4				5
or:								

Figure 3.28

23. **Run** *the query.*

You should end up with a record set of 50 records (because there are 50 records at the fifth level) that looks like this:



Level_1_ID	Level_1_Acct	Level_1_Desc	Level_2_ID	Level_2_Acct	Level_2_Desc	Level_3_ID	Level_3_Acct	Level_3_Desc
1001	NETINC	Net Income	1002	REVENUE	Revenue	1004	OPERREV	Operating Reven
1001	NETINC	Net Income	1002	REVENUE	Revenue	1004	OPERREV	Operating Reven
1001	NETINC	Net Income	1002	REVENUE	Revenue	1004	OPERREV	Operating Reven
1001	NETINC	Net Income	1002	REVENUE	Revenue	1004	OPERREV	Operating Reven
1001	NETINC	Net Income	1002	REVENUE	Revenue	1004	OPERREV	Operating Reven
1001	NETINC	Net Income	1002	REVENUE	Revenue	1004	OPERREV	Operating Reven
1001	NETINC	Net Income	1002	REVENUE	Revenue	1004	OPERREV	Operating Reven
1001	NETINC	Net Income	1002	REVENUE	Revenue	1004	OPERREV	Operating Reven
1001	NETINC	Net Income	1002	REVENUE	Revenue	1004	OPERREV	Operating Reven
1001	NETINC	Net Income	1002	REVENUE	Revenue	1004	OPERREV	Operating Reven
1001	NETINC	Net Income	1002	REVENUE	Revenue	1004	OPERREV	Operating Reven
1001	NETINC	Net Income	1002	REVENUE	Revenue	1004	OPERREV	Operating Reven
1001	NETINC	Net Income	1002	REVENUE	Revenue	1004	OPERREV	Operating Reven
1001	NETINC	Net Income	1002	REVENUE	Revenue	1004	OPERREV	Operating Reven
1001	NETINC	Net Income	1002	REVENUE	Revenue	1004	OPERREV	Operating Reven
1001	NETINC	Net Income	1002	REVENUE	Revenue	1004	OPERREV	Operating Reven
1001	NETINC	Net Income	1002	REVENUE	Revenue	1004	OPERREV	Operating Reven
1001	NETINC	Net Income	1002	REVENUE	Revenue	1004	OPERREV	Operating Reven
1001	NETINC	Net Income	1002	REVENUE	Revenue	1004	OPERREV	Operating Reven
1001	NETINC	Net Income	1002	REVENUE	Revenue	1004	OPERREV	Operating Reven
1001	NETINC	Net Income	1002	REVENUE	Revenue	1004	OPERREV	Operating Reven
1001	NETINC	Net Income	1002	REVENUE	Revenue	1004	OPERREV	Operating Reven
1001	NETINC	Net Income	1002	REVENUE	Revenue	1004	OPERREV	Operating Reven
1001	NETINC	Net Income	1002	REVENUE	Revenue	1004	OPERREV	Operating Reven

Figure 3.29

24. **Save** *the query as* ***qry03COA_Flat*** *and close it.*

Now you have a COA flat file that you can and will use many times. If the COA chart changes, the query will automatically be updated. Please remember this concept in all of your database programming. You should be able to make a change in ONE place and have it populate everywhere it needs to.

> ***Review Questions****: It is now time to complete the hands-on Review Questions. Log on to www.ExcelCEO.com with your Email and Password, click on the* **Access 2016 and SQL Review Questions, Chapter 3, Section 2 of 2** *option and complete the review questions.*

Conclusion

In this chapter, you fully stepped into the world of relational databases. You created a query, set filters, and created relationships (or joins) to other tables. You learned the difference between flat files and hierarchical files, and even created a query changing a hierarchical file into a flat file format. I hope you are beginning to see the real power of relational databases. In the next chapter, we'll continue to enhance your query-building skills and introduce other types of joins and functionality.

Chapter Exam

You can now go to www.ExcelCEO.com, click on Sign In, log in and take the exam. Make sure that you take the exam on the same device on which you completed the practice files, as some of the questions on the exam may refer to some of the completed projects. Chapter exams are intended to be hands-on.

CHAPTER FOUR — ADVANCED QUERIES

In this chapter, you will:

- Determine how to use an IIF() Function
- Recognize how to design a Parameter Query
- Select the Grouping option in a Query Design Grid
- Determine when to use one-way joins
- Identify how subqueries are used

CPE Credits possible for this chapter: 3

Advanced Queries

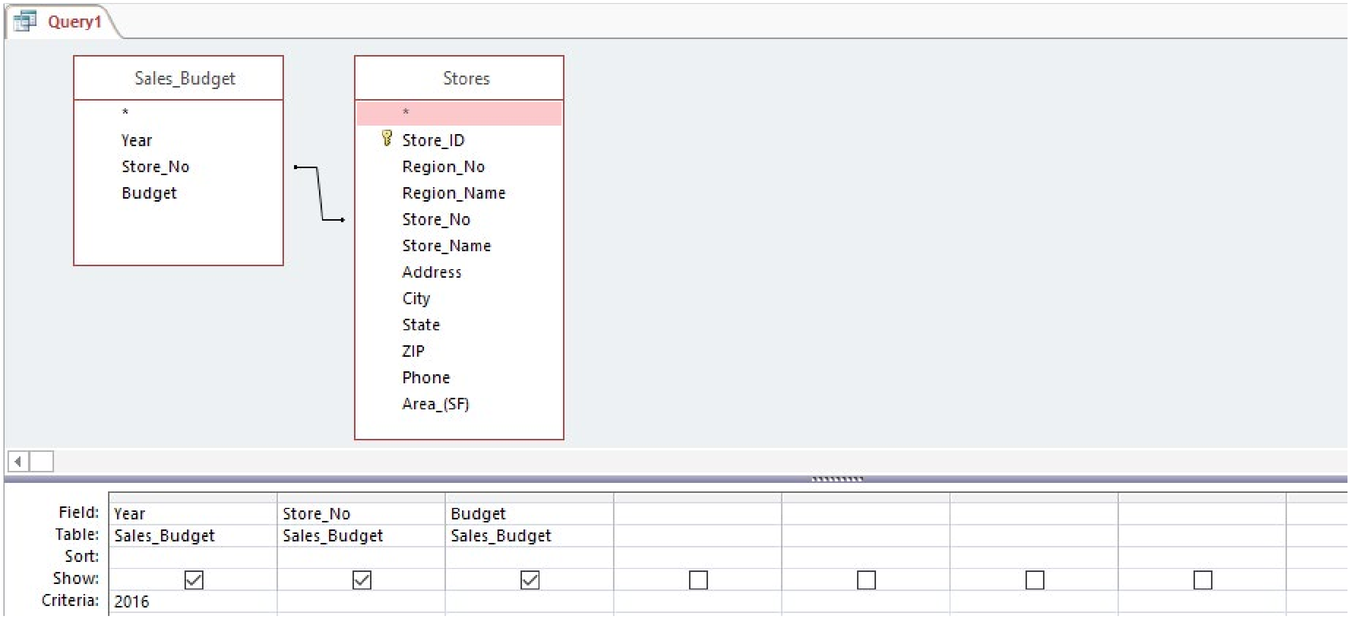
A co-worker and I once took a SQL Server programming course. In the course, the instructor was talking about left and right joins, sometimes referred to as an outer or one-way join, to be discussed later in this chapter. He told us that you should never use more than two one-way joins in a query. When I asked why, he said it was "unstable". My co-worker and I had recently created a query with twelve one-way joins (one for each month of the year), so we printed out the code and showed it to the teacher. I commented that it was not unstable, because we always got the expected result when we ran the query. He had never seen a query with so many one-way joins, but that was commonplace in our world. Don't you just love showing the teacher something new?

In this chapter, you will enhance your query-building skills. We'll explore the IIF() function and other functions, and talk more about grouping and one-way joins. But first, let's talk a little about functions. I'm not going to go into functions in-depth as much as I did in the Excel course. Functions work the same way in Access as they do in Excel. Most of the syntax (or the coding) is even the same, except you refer to the field names instead of the cell references.

The IIF() Function

One function that is a little different in Access than in Excel is the IIF() function. The ***IIF() function*** in Access is basically the same as the IF() function in Excel, except that you refer to field names in the formula rather than cell addresses, as in Excel. Let's try it out using one of the same exercises that we did in Excel.

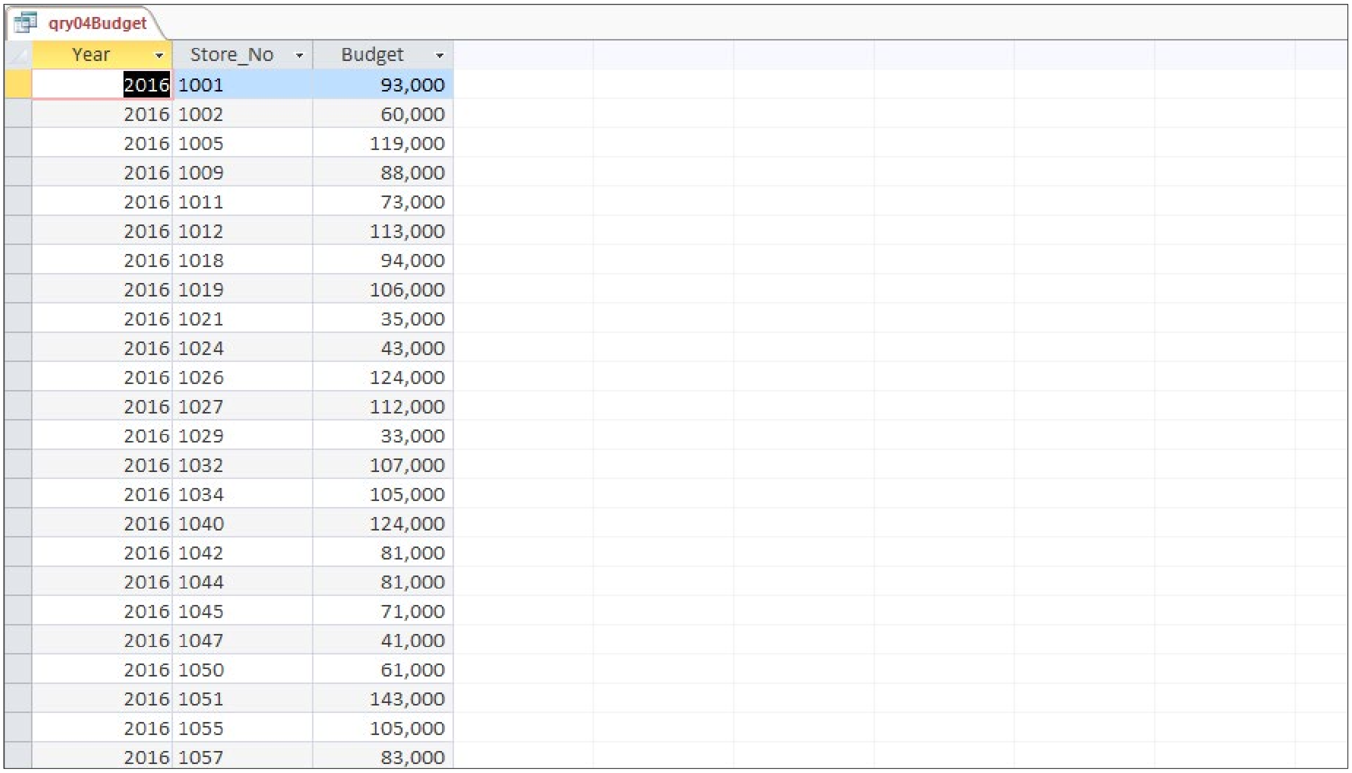
1. *Open the* **Nitey_Nite_2016** *database.*
2. *Create a query that uses the* **Sales_Budget** *and* **Stores** *tables, create a join on* **Store_No**, *and bring the* **Year**, **Store_No** *and* **Budget** *fields into the query* **Design Grid**.
3. *Filter the data for the Year* ***2016****.*



Field:	Year	Store_No	Budget				
Table:	Sales_Budget	Sales_Budget	Sales_Budget				
Sort:							
Show:	☑	☑	☑	☐	☐	☐	☐
Criteria:	2016						

Figure 4.1

4. *Using the* **Property Sheet**, *format the* **Budget** *field to be* **Standard** *with* **Zero decimal places**.
5. *Save the query as* ***qry04Budget*** *and* **Run** *it.*

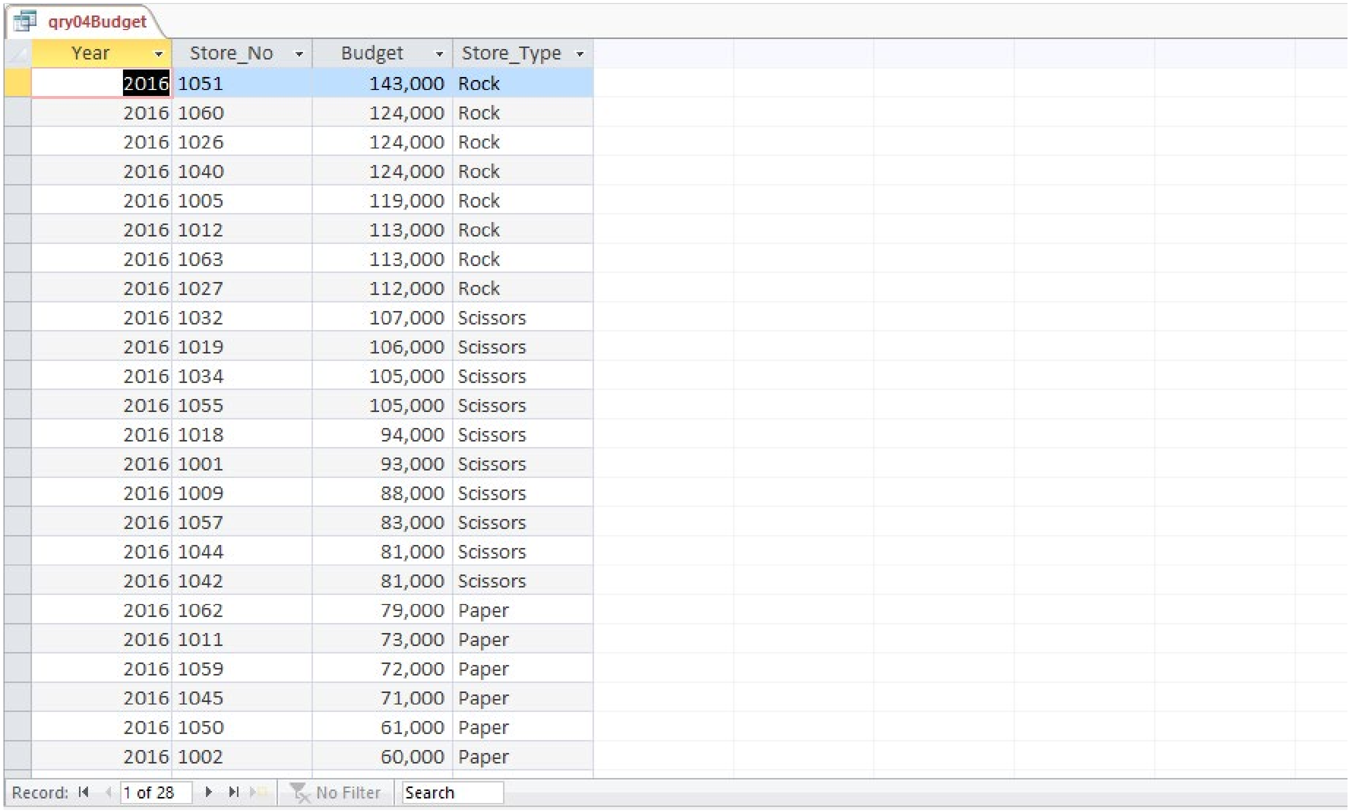


Year	Store_No	Budget
2016	1001	93,000
2016	1002	60,000
2016	1005	119,000
2016	1009	88,000
2016	1011	73,000
2016	1012	113,000
2016	1018	94,000
2016	1019	106,000
2016	1021	35,000
2016	1024	43,000
2016	1026	124,000
2016	1027	112,000
2016	1029	33,000
2016	1032	107,000
2016	1034	105,000
2016	1040	124,000
2016	1042	81,000
2016	1044	81,000
2016	1045	71,000
2016	1047	41,000
2016	1050	61,000
2016	1051	143,000
2016	1055	105,000
2016	1057	83,000

Figure 4.2

Now you have a simple query. It shows the sales budgets for each store in 2016. Let's suppose that any store that has a budget of less than $80,000 is called a Paper store, a store with a budget between $80,000 and $110,000 is a Scissors store, and a store with a budget of more than $110,000 is a Rock store. To show these store classifications, you will create an alias field and write an IIF() function with these assumptions. To make the auditing of the formula a bit easier, you can sort the data on the Budget field in Descending order. As such, all of the Rock stores should appear first, followed by the Scissors stores, and then by the Paper stores.

6. *Return to the* **Design View** *of* **qry04Budget**.
7. *In a new field next to* **Budget**, *type the following alias and formula:*
   Store_Type: IIF(Budget<80000,”Paper”,IIF(Budget<110000,”Scissors”,”Rock”))
8. *Click in the* **Sort:** *row under* **Budget** *and choose* **Descending**.
9. *Save* **qry04Budget** *and* **Run** *it.*



Year	Store_No	Budget	Store_Type
2016	1051	143,000	Rock
2016	1060	124,000	Rock
2016	1026	124,000	Rock
2016	1040	124,000	Rock
2016	1005	119,000	Rock
2016	1012	113,000	Rock
2016	1063	113,000	Rock
2016	1027	112,000	Rock
2016	1032	107,000	Scissors
2016	1019	106,000	Scissors
2016	1034	105,000	Scissors
2016	1055	105,000	Scissors
2016	1018	94,000	Scissors
2016	1001	93,000	Scissors
2016	1009	88,000	Scissors
2016	1057	83,000	Scissors
2016	1044	81,000	Scissors
2016	1042	81,000	Scissors
2016	1062	79,000	Paper
2016	1011	73,000	Paper
2016	1059	72,000	Paper
2016	1045	71,000	Paper
2016	1050	61,000	Paper
2016	1002	60,000	Paper

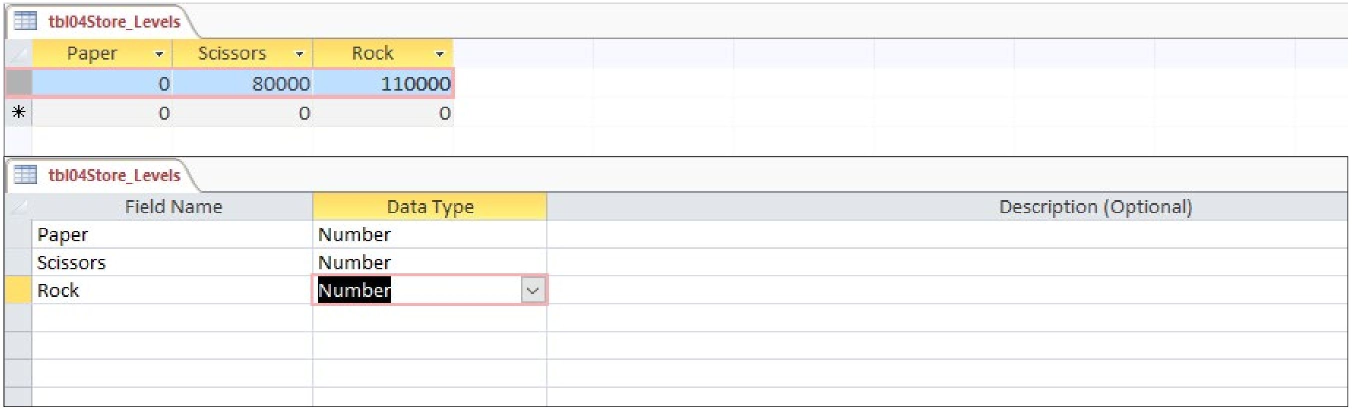
Figure 4.3

That's how to use the IIF() function and sorting within a query. Remember that Access solves the formulas from left to right, just like Excel.

Assumptions Table

One of the problems in hard-coding the stores' types in a formula is that if the dollar amounts change, you would have to go into the formula and change it. In the Excel course, we solved that issue by having an Assumptions tab that contains all of the workbook's criteria and assumptions. In Access, you can create an ***Assumptions Table*** that contains that data, but you need to be careful how you set it up. Let's create an assumptions table that has this budget criteria in it. All of these fields should have a field size of Long Integer with no default value.

1. *Close* **qry04Budget***.*
2. **Create** *a new table (with no primary key) called* ***tbl04Store_Levels*** *and make it look like the following:*



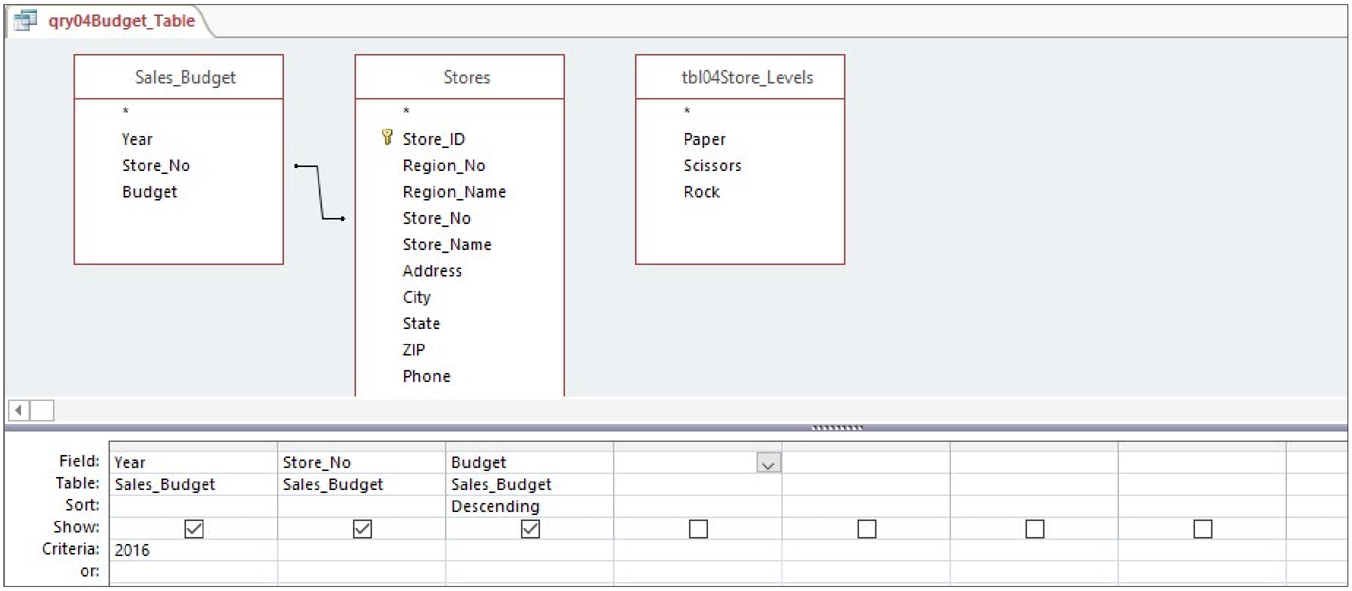
	Paper	Scissors	Rock
	0	80000	110000
*	0	0	0

Field Name	Data Type	Description (Optional)
Paper	Number	
Scissors	Number	
Rock	Number	

Figure 4.4

This table simply stores data in three fields with the minimum budget amount for each level.

3. **Save** *and close* **tbl04Store_Levels.**
4. *In the* **Navigation Pane** *under* **Queries**, *click on the* **qry04Budget** *query and* **copy** *and* **paste** *it.*
5. *When the* **Paste As** *dialog box appears, type* ***qry04Budget_Table*** *(to remind yourself it is based on a table) as the* **Query Name** *and click* **OK**.
6. *In the* **Design View** *of* **qry04Budget_Table**, *delete the field containing the formula for* **Store_Type**.
7. *Bring in the* **tbl04Store_Levels** *table, but do not create a relationship with either of the other tables.*

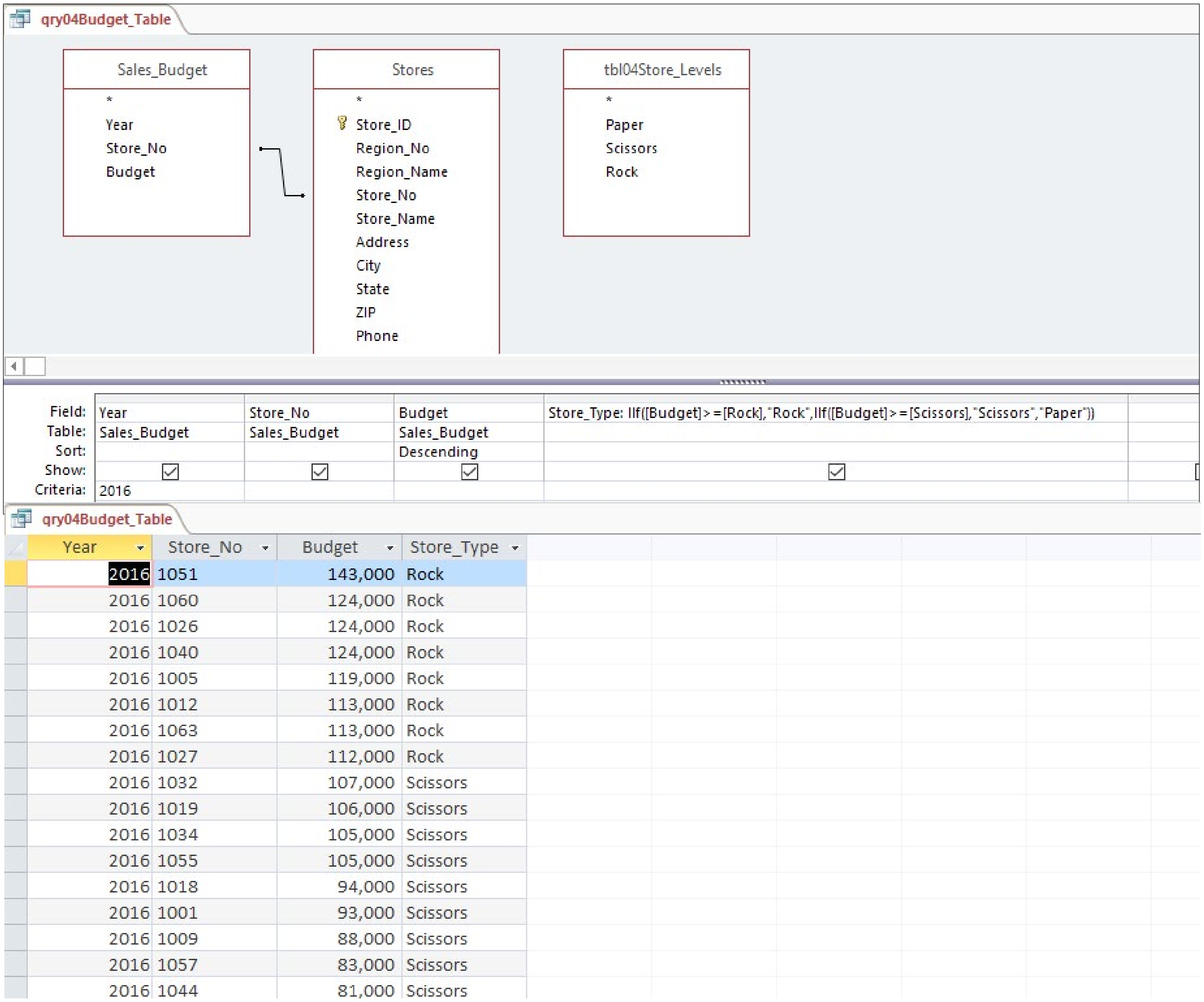


Field:	Year	Store_No	Budget				
Table:	Sales_Budget	Sales_Budget	Sales_Budget				
Sort:			Descending				
Show:	☑	☑	☑	□	□	□	□
Criteria:	2016						
or:							

Figure 4.5

We didn’t establish any joins with the new table because there are no fields on which you could create a join. Remember that there is only one row of data in the table. If there were two rows of data, the record set returned would contain double the information (two rows for each unique record in the dataset). Therefore, an assumptions table that has no fields that could be joined with other tables should contain only one record.

8. *In a new field, write the following formula with the alias* ***Store_Type: IIF(Budget>=Rock,"Rock",IIF(Budget>=Scissors, "Scissors","Paper"))***
9. **Run** *the query and then save it.*



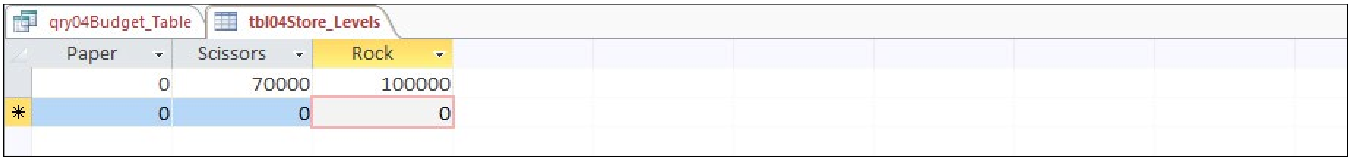
Field:	Year	Store_No	Budget	Store_Type: IIf([Budget]>=[Rock],"Rock",IIf([Budget]>=[Scissors],"Scissors","Paper"))
Table:	Sales_Budget	Sales_Budget	Sales_Budget	
Sort:			Descending	
Show:	☑	☑	☑	☑
Criteria:	2016			

Year	Store_No	Budget	Store_Type
2016	1051	143,000	Rock
2016	1060	124,000	Rock
2016	1026	124,000	Rock
2016	1040	124,000	Rock
2016	1005	119,000	Rock
2016	1012	113,000	Rock
2016	1063	113,000	Rock
2016	1027	112,000	Rock
2016	1032	107,000	Scissors
2016	1019	106,000	Scissors
2016	1034	105,000	Scissors
2016	1055	105,000	Scissors
2016	1018	94,000	Scissors
2016	1001	93,000	Scissors
2016	1009	88,000	Scissors
2016	1057	83,000	Scissors
2016	1044	81,000	Scissors

Figure 4.6

Your query result should be exactly the same as in qry04Budget, with eight Rock stores, ten Scissors stores, and ten Paper stores. Whenever possible, I like to store variables (or values that could possibly change) in tables rather than in formulas or code. Maintaining this kind of data in tables is much easier than changing the formulas. Sometimes it's a bit more work to setup in tables, but once it's there, it's much easier to change later. With the data now in a table, let's change the Scissors amount to $70,000 and the Rock amount to $100,000.

10. *Open the* **tbl04Store_Levels** *table and change* **Scissors** *to* ***70000*** *and* **Rock** *to* ***100000****.*

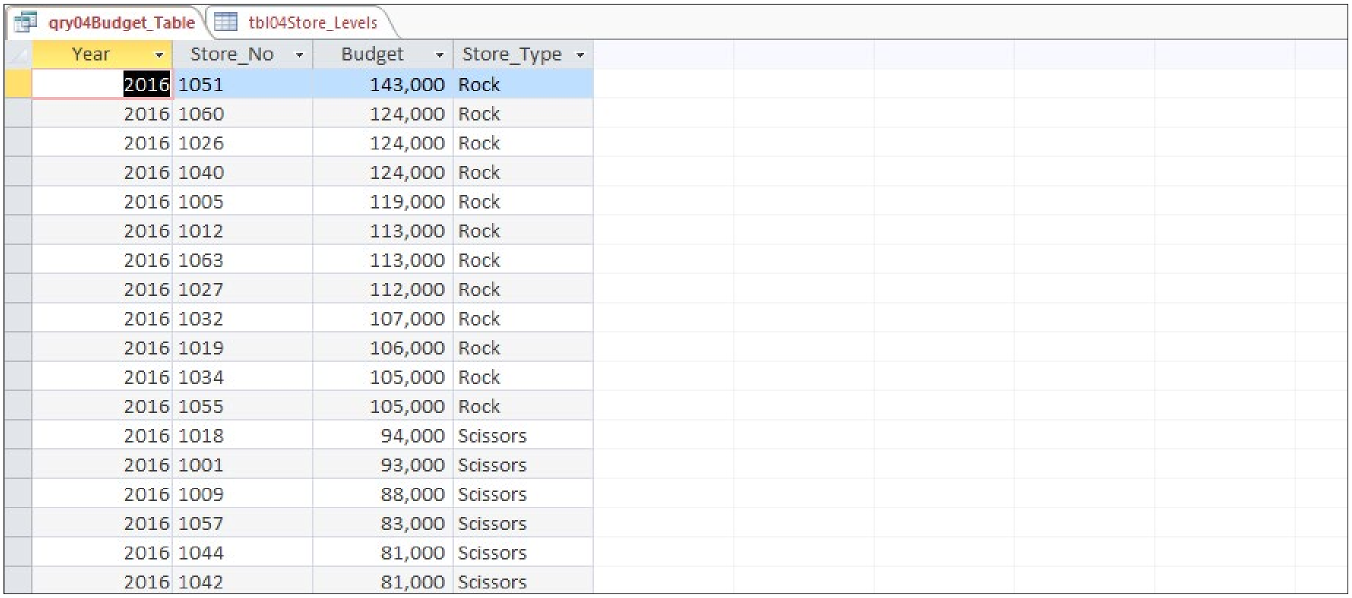


Paper	Scissors	Rock
0	70000	100000
0	0	0

Figure 4.7

11. *Rerun* **qry04Budget_Table***.*

> ***Note****: To rerun an open query, click on the* **Home** *tab, then the drop-down arrow on the* **Refresh All** *icon in the* **Records** *group, and click on* **Refresh***. Click on* **Refresh All** *to run all of the queries.*



Year	Store_No	Budget	Store_Type
2016	1051	143,000	Rock
2016	1060	124,000	Rock
2016	1026	124,000	Rock
2016	1040	124,000	Rock
2016	1005	119,000	Rock
2016	1012	113,000	Rock
2016	1063	113,000	Rock
2016	1027	112,000	Rock
2016	1032	107,000	Rock
2016	1019	106,000	Rock
2016	1034	105,000	Rock
2016	1055	105,000	Rock
2016	1018	94,000	Scissors
2016	1001	93,000	Scissors
2016	1009	88,000	Scissors
2016	1057	83,000	Scissors
2016	1044	81,000	Scissors
2016	1042	81,000	Scissors

Figure 4.8

Now your query returns twelve Rock stores, ten Scissors stores, and six Paper stores.

> ***IMPORTANT NOTE****: When you* **update** *a record in* **Access***, you must click outside of the row for the table to update. If you had changed Scissors to 70,000, then tabbed over to the Rock field and changed it to 100,000 without getting out of that row, the query would not have picked up the new values.*

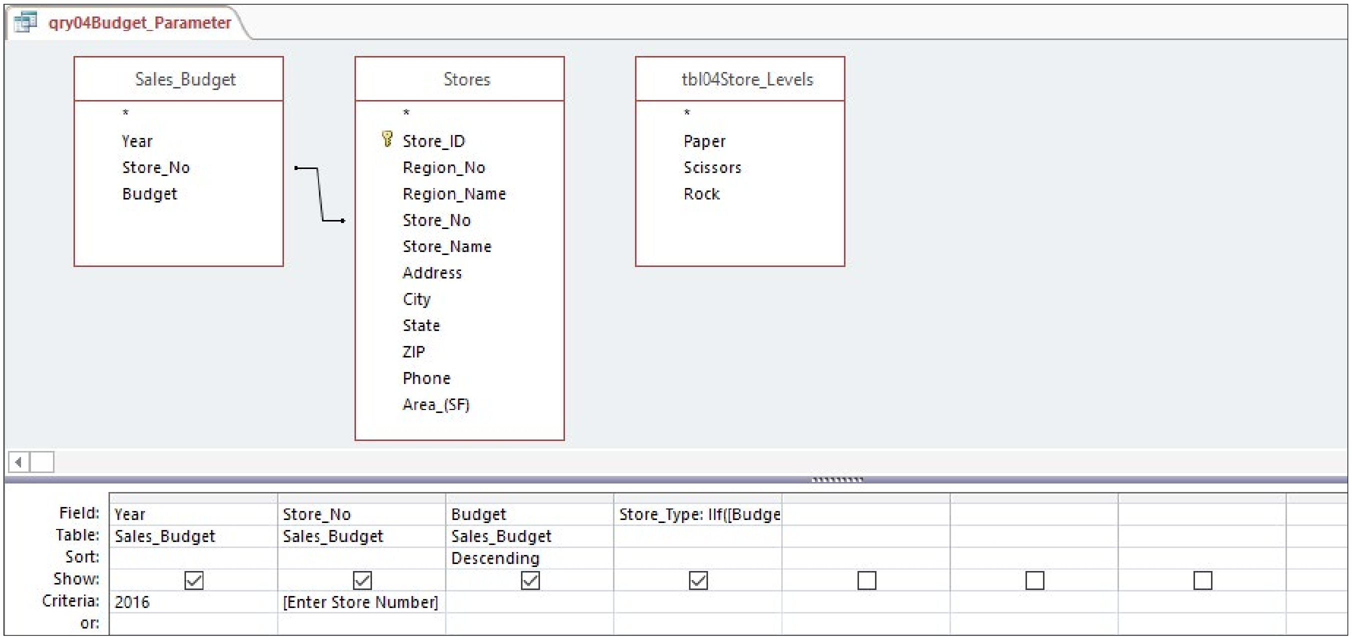
12. *Close the* **tbl04Store_Levels** *table and* **save** *and close* **qry04Budget_Table**.

Parameter Queries

Now let's talk about one of my favorite types of queries, Parameter Queries. A ***Parameter Query*** is a query that is dependent on user input. When it is run, it stops at a certain point and requires the user to input a value that will be used in the query. Using qry04Budget_Table as an example, let's say that you want to know the budget and store type for Store_No. 1009. You could run the query and look for Store_No. 1009 (or sort by Store_No to make it a little easier to find that particular store), or you could enter (hard code) the store number on the Criteria line. Alternatively, you could create a parameter query that asks you to input the store number, and returns the results for that store only.

To set up a Parameter query on the Store_No field, you type in a text string (this will be the prompt verbiage) on the Criteria line and surround it with brackets ([]). The only condition to a parameter query is that the text string cannot be the same as any existing field name you have in the table(s) used in the query. Let's create a Parameter query.

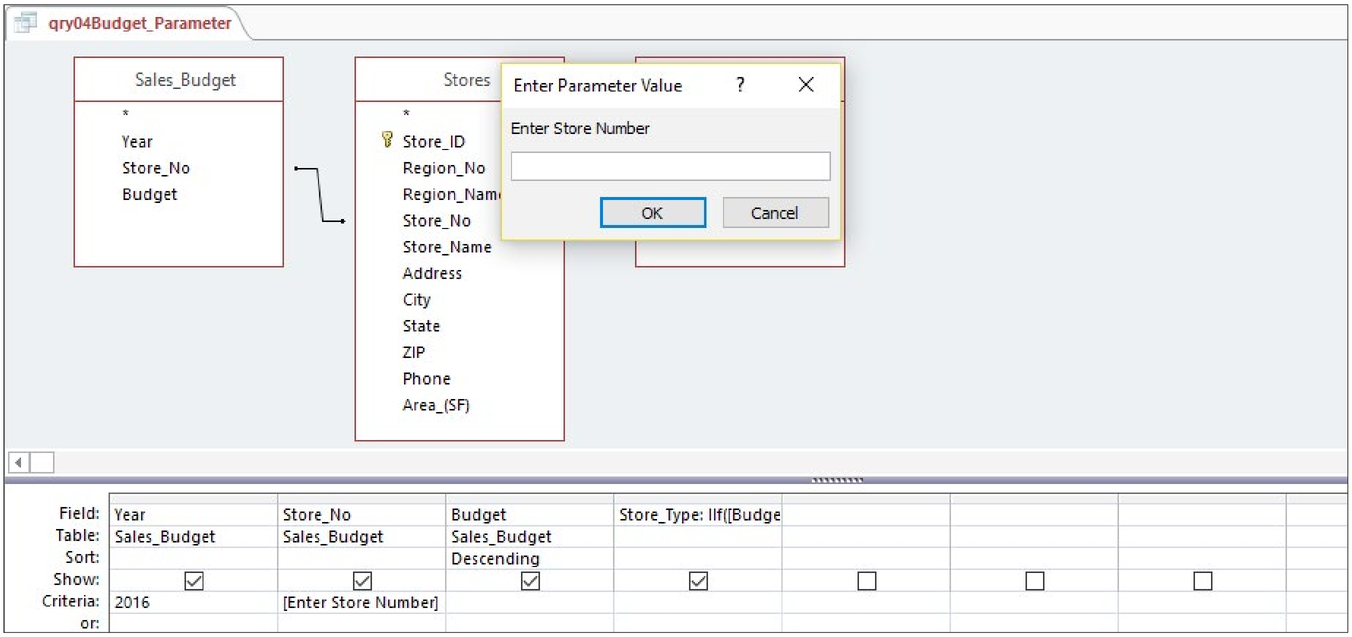
1. *In the* **Navigation Pane** *under* **Queries**, *copy* **qry04Budget_Table** *and save the new query as* ***qry04Budget_Parameter***.
2. *In* **Design View** *of* **qry04Budget_Parameter**, *type* ***[Enter Store Number]*** *in the* **Criteria** *line under the* **Store_No** *field.*



Field:	Year	Store_No	Budget	Store_Type: IIf([Budge			
Table:	Sales_Budget	Sales_Budget	Sales_Budget				
Sort:			Descending				
Show:	☑	☑	☑	☑	□	□	□
Criteria:	2016	[Enter Store Number]					
or:							

Figure 4.9

3. **Run** *the query.*

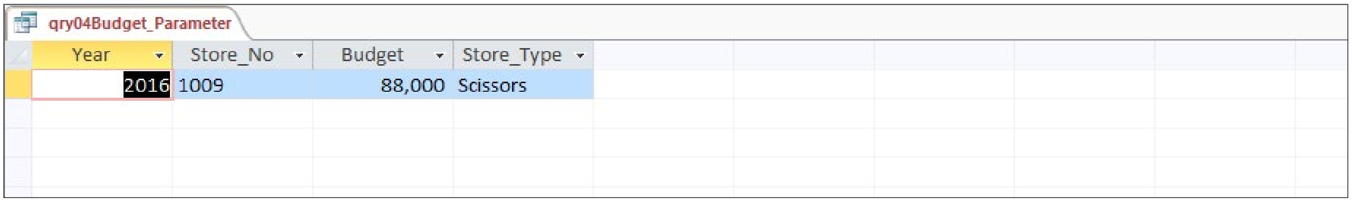


Field:	Year	Store_No	Budget	Store_Type: IIf([Budge			
Table:	Sales_Budget	Sales_Budget	Sales_Budget				
Sort:			Descending				
Show:	☑	☑	☑	☑	☐	☐	☐
Criteria:	2016	[Enter Store Number]					
or:							

Figure 4.10

A dialog box pops up with the prompt "*Enter Store Number*" — that was the text string you typed in brackets on the Criteria line.

4. *Input **1009** and click* **OK**.

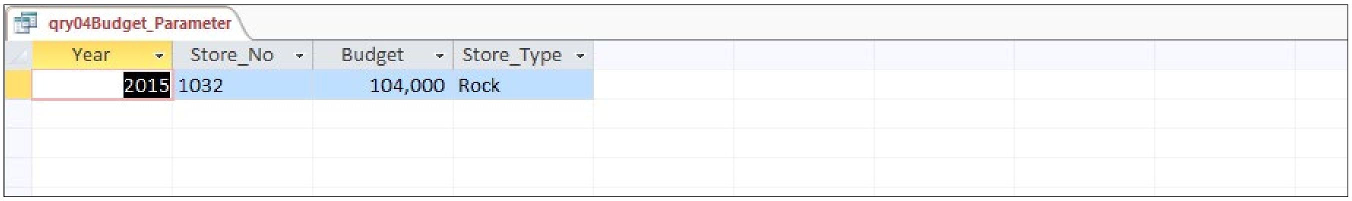


Year	Store_No	Budget	Store_Type
2016	1009	88,000	Scissors

Figure 4.11

The query returns one record, which is the budget and store type for Store_No 1009. Parameter queries typically take less time to run because they filter for one value as opposed to returning all the records. It would be the same if you had hard-coded the store number in the Criteria line. You can also input multiple parameters in one query.

5. *In* **Design View** *of* **qry04Budget_Parameter**, *replace* **2016** *under the* **Year** *field with* ***[Enter Year]*** *and* **Run** *the query.*
6. *Enter* ***2015*** *for the* **year** *and* ***1032*** *for the* **Store_No**.



Year	Store_No	Budget	Store_Type
2015	1032	104,000	Rock

Figure 4.12

7. **Save** *and close* **qry04Budget_Parameter**.

You can have as many parameter strings as you have parameters in the query. You can use parameter strings in alias formulas as well.

> ***Review Questions****: It is now time to complete the hands-on Review Questions. Log on to www.ExcelCEO.com with your Email and Password, click on the* **Access 2016 and SQL Review Questions, Chapter 4, Section 1 of 2** *option and complete the review questions.*

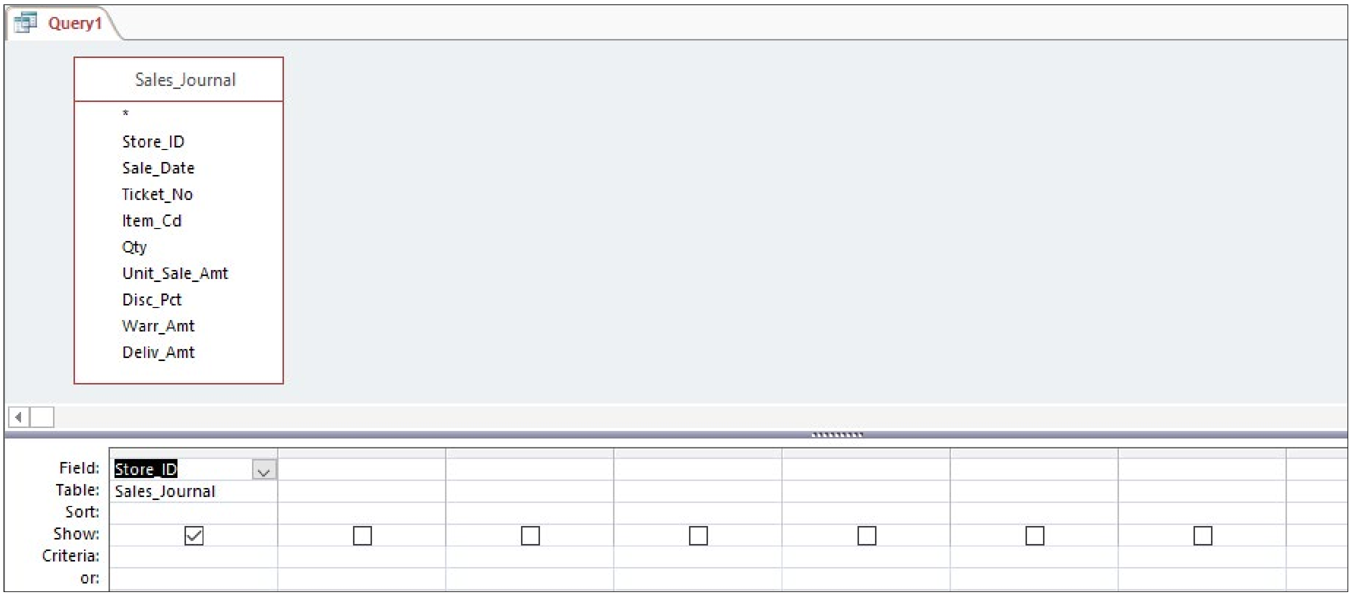
Grouping

Next we'll talk about grouping. ***Grouping*** in Access is a pre-cursor to grouping in SQL coding (explored in Chapters 13 and 14), so it is very important that you grasp the concept. You've already grouped data by using the Total button in Chapter 3. Let's start off with a simple example and then I'll talk more about it.

In the next example, we will use the Sales_Journal table. The Sales_Journal table is a detailed description of every sale made at Nitey-Nite. It contains fields showing the Store_ID, Sale_Date, Ticket_No (or invoice number), Item_Cd (which when joined with the Item table gives a complete description of the item sold), Qty (quantity, or number of items sold for each item on that ticket), and various amount fields describing the sale price of each unit (Unit_Sale_Amt), the percent discount given on each item sold, Warr_Amt (the amount of the warranty sale), and Deliv_Amt (the amount of the delivery charge).

Suppose that you want to find all of the store numbers in the Sales_Journal table that had at least one sale, or in other words, a unique set of the store numbers that are in the Sales_Journal table.

1. **Create** *a new query using the* **Sales_Journal** *table.*
2. *Bring only the* **Store_ID** *field into the* **Design Grid**.



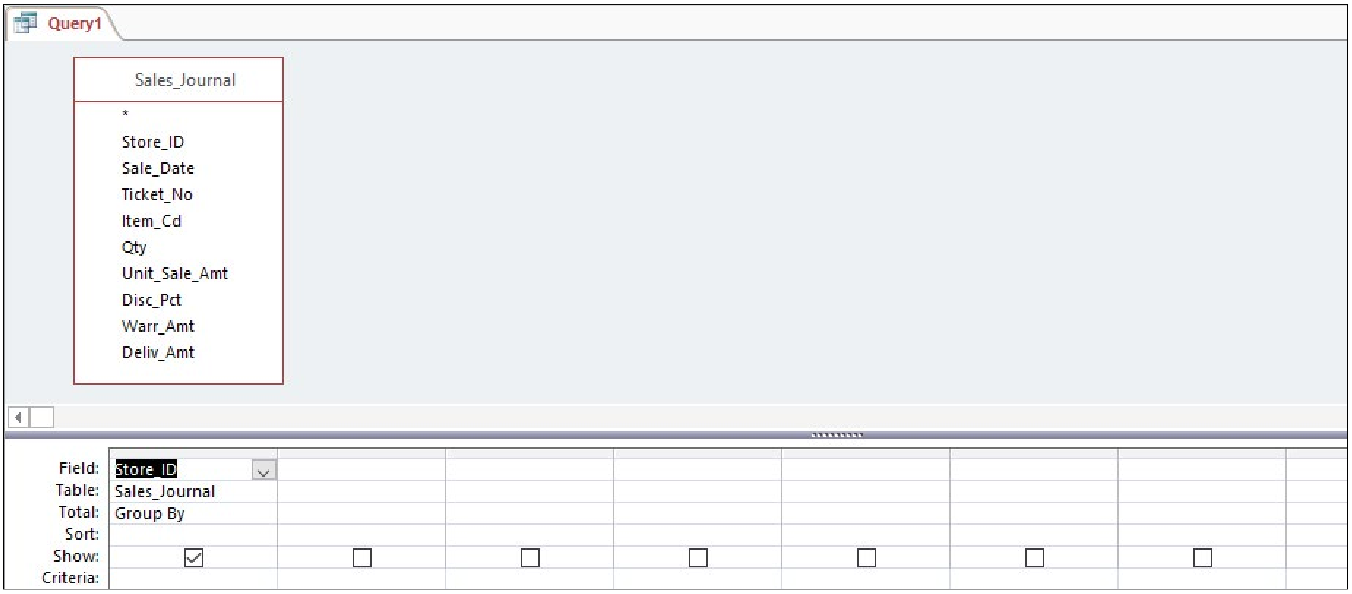
Field: Store_ID
Table: Sales_Journal
Sort:
Show: ☑ □ □ □ □ □ □
Criteria:
or:

Figure 4.13

3. **Run** *the query.*

All you did was to bring in one field, the Store_ID, from the Sales_Journal table. If you open the Sales_Journal table, you will see that there are 141,609 records, which is also the number of records produced by this query. If you want to get a unique list of the store IDs, just group the data.

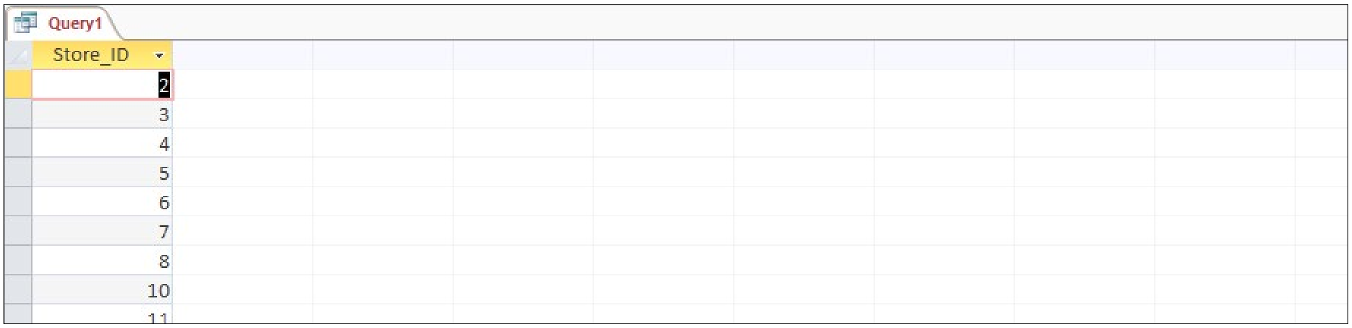
4. *Return to* **Design View**.
5. *Click the* **Totals** *icon* Σ *and* **Run** *the query.*



Field: Store_ID
Table: Sales_Journal
Total: Group By
Sort:
Show: ☑ □ □ □ □ □ □
Criteria:

Figure 4.14

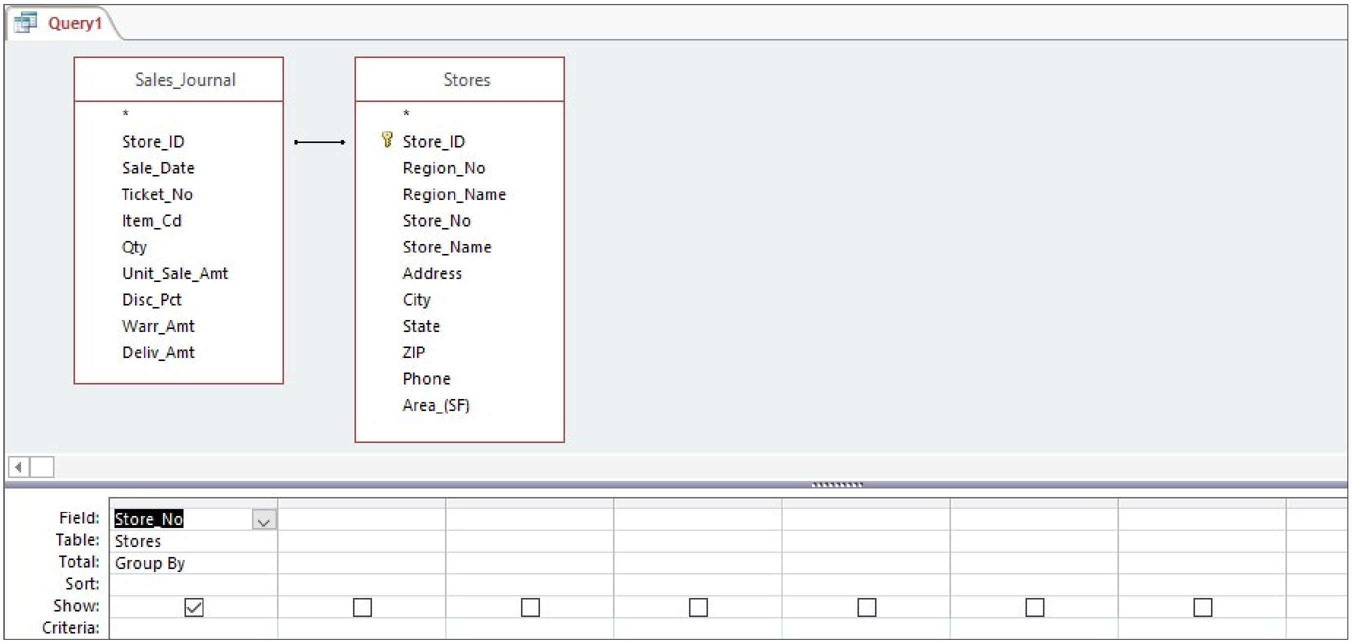
The Total row appears with a default value of Group By under Store_ID. When you run the query, you see that only 29 records appear, meaning that each of those Store_IDs appear at least once in the Sales_Journal table. The Store_ID in itself doesn't tell you much, but you can now create a join to the Stores table to get the Store_No.



Store_ID
2
3
4
5
6
7
8
10
11

Figure 4.15

6. *Return to* **Design View**.
7. *Bring in the* **Stores** *table and make sure a join is intact on both tables using the* **Store_ID**.
8. *Replace* **Store_ID** *in the grid with* **Store_No**.
9. **Run** *the query.*



Field:	Store_No						
Table:	Stores						
Total:	Group By						
Sort:							
Show:	☑	□	□	□	□	□	□
Criteria:							

Figure 4.16

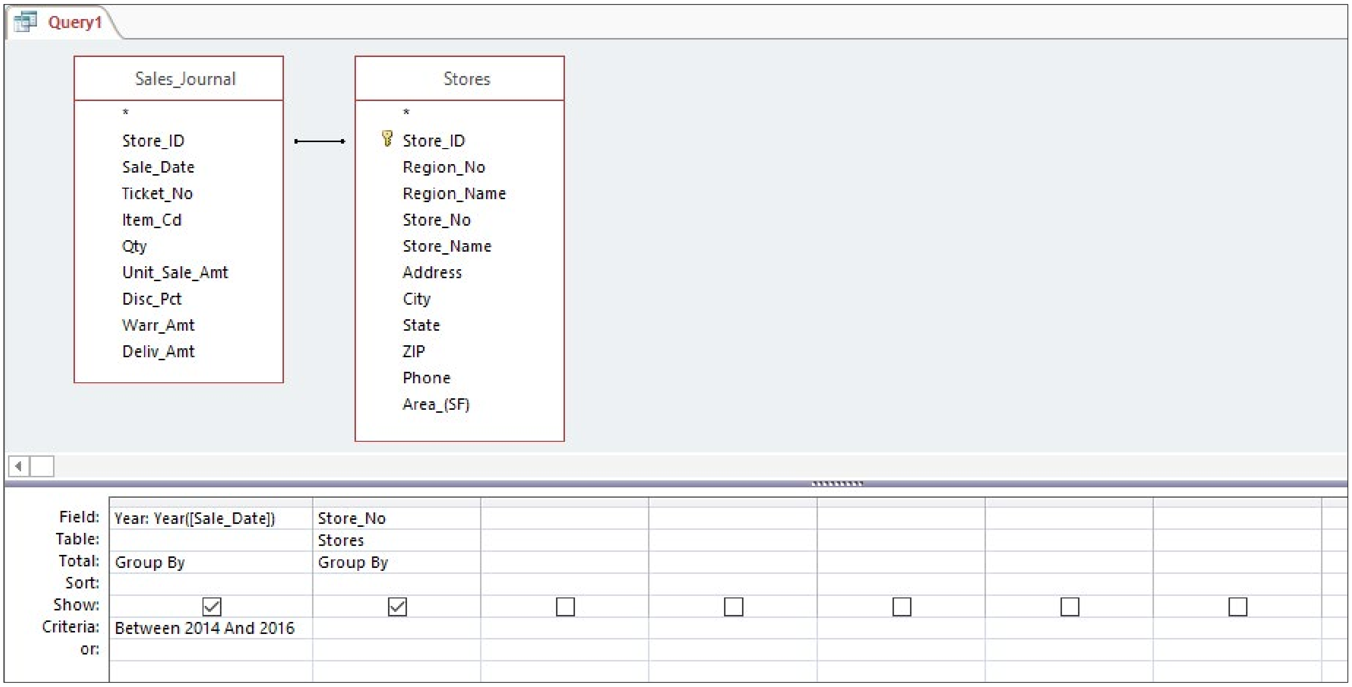
You now see that the record set produced is the same number of records (29), but now it contains the store numbers. This is a good example of what is called a one-to-many relationship, as discussed in Chapter 3.

There are many Store_ID records in the Sales_Journal table, but only one unique Store_ID for each store in the Stores table. When you create a relationship on two tables like this, it will return the number of records in both tables. Had there been two records for each Store_ID in the Stores table, the query would have produced 283,218 records, or exactly double. So if you are ever auditing a query and the data is exactly doubled or tripled, you probably have an issue with one of your joins. This is also why the ID fields in tables should be unique identifiers, or Primary Keys.

> ***Note****: Instead of storing the* **Store_No** *in the* **Sales_Journal***, the database designer decided to use the* **Store_ID***, probably because there are so many records in that table, and performing joins on numbers (the Store_ID is formatted as a* **Number***) works much faster than joins on text strings (the Store_No is a* **Text String***). This is good database design. ID fields should be in a* **Number** *format as it enhances performance.*

You can now add fields from the Sales_Journal table. Do you remember calculating total sales from Nitey-Nite's database in Excel? We're going to do the same thing here. Let's suppose you want to find the total sales at each store for the years 2014 – 2016. All you need to do is to bring in the right data.

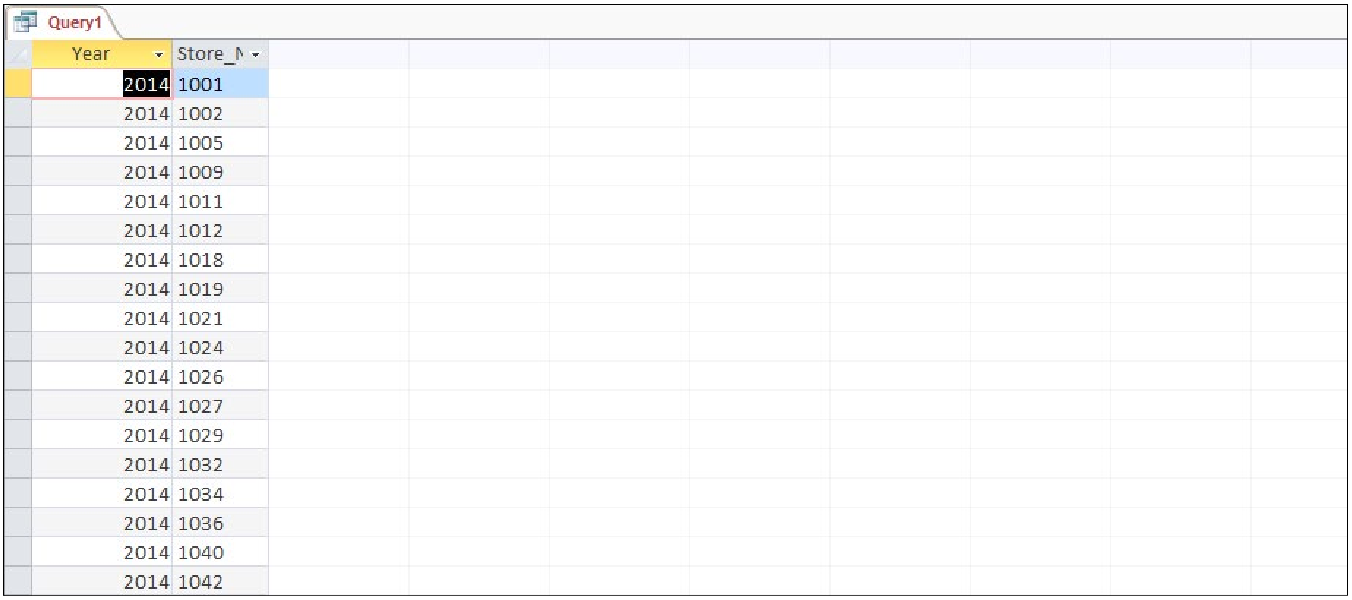
10. *Return to* **Design View** *of the query.*
11. *Drag the* **Sale_Date** *field in the* **Sales_Journal** *table to the left of the* **Store_No** *field.*
12. *Edit* **Sale_Date** *to be as follows:* ***Year: Year(Sale_Date)***
13. *In the* **Year** *field, input the following criteria:* ***Between 2014 and 2016***



Field:	Year: Year([Sale_Date])	Store_No					
Table:		Stores					
Total:	Group By	Group By					
Sort:							
Show:	☑	☑	□	□	□	□	□
Criteria:	Between 2014 And 2016						
or:							

Figure 4.17

14. **Run** *the query.*

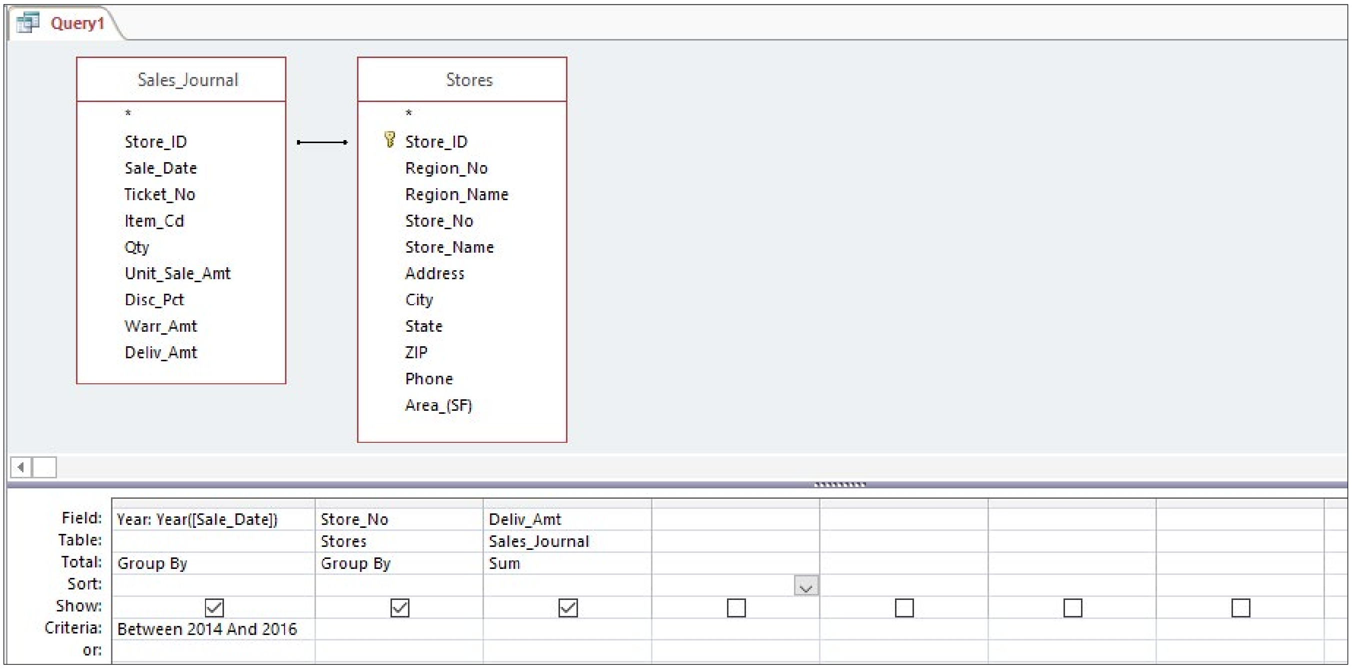


Year	Store_N
2014	1001
2014	1002
2014	1005
2014	1009
2014	1011
2014	1012
2014	1018
2014	1019
2014	1021
2014	1024
2014	1026
2014	1027
2014	1029
2014	1032
2014	1034
2014	1036
2014	1040
2014	1042

Figure 4.18

The query now shows you 87 records by year and by store. But you want to see the total sales by year and by store. So now all you have to do is to bring in the appropriate amount fields and sum them up.

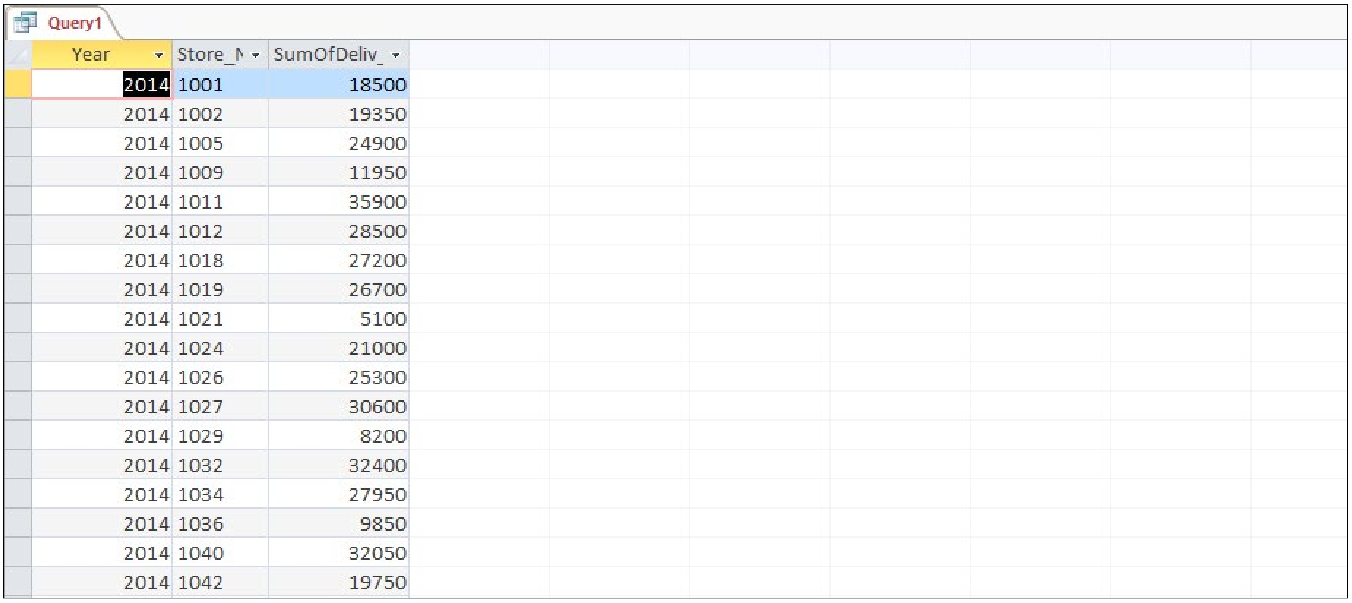
15. *Return to* **Design View***.*
16. *Drag the* **Deliv_Amt** *field next to* **Store_No** *and release.*
17. *Under the* **Deliv_Amt** *field, change the* **Group By** *criteria to* **Sum***.*



Field:	Year: Year([Sale_Date])	Store_No	Deliv_Amt				
Table:		Stores	Sales_Journal				
Total:	Group By	Group By	Sum				
Sort:							
Show:	☑	☑	☑	□	□	□	□
Criteria:	Between 2014 And 2016						
or:							

Figure 4.19

18. **Run** *the query.*

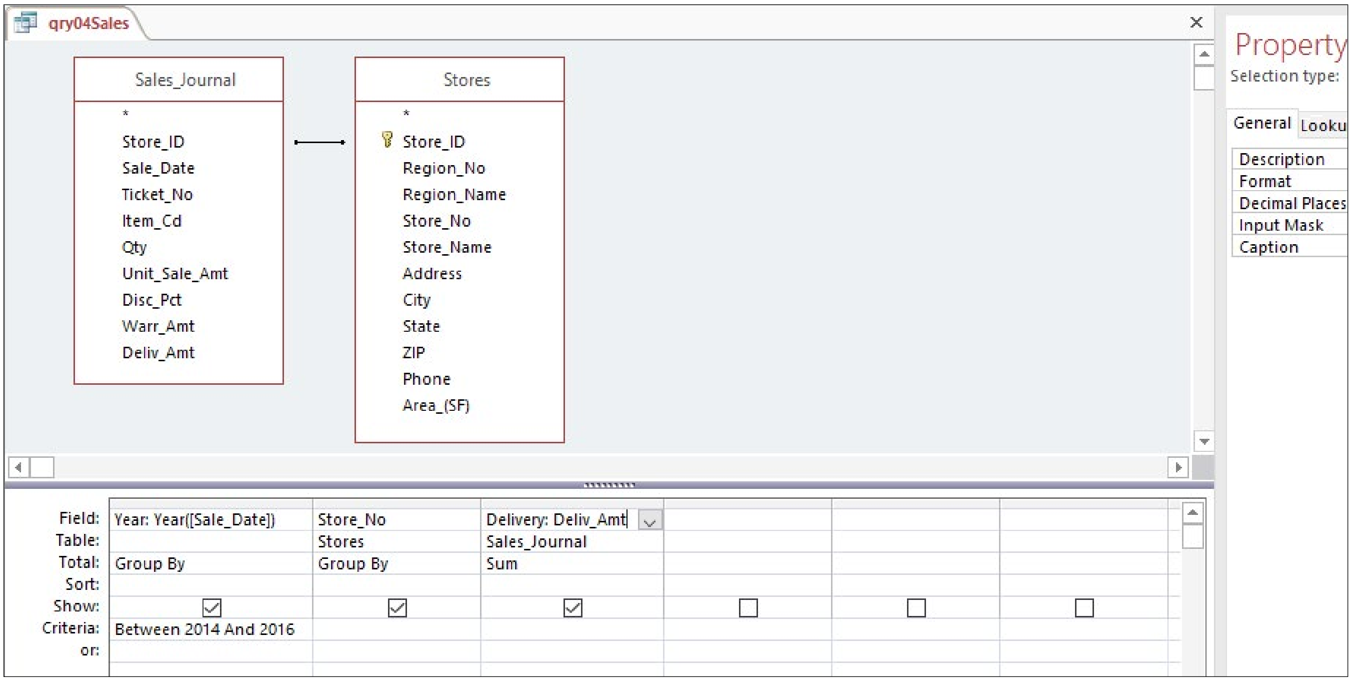


Year	Store_N	SumOfDeliv_
2014	1001	18500
2014	1002	19350
2014	1005	24900
2014	1009	11950
2014	1011	35900
2014	1012	28500
2014	1018	27200
2014	1019	26700
2014	1021	5100
2014	1024	21000
2014	1026	25300
2014	1027	30600
2014	1029	8200
2014	1032	32400
2014	1034	27950
2014	1036	9850
2014	1040	32050
2014	1042	19750

Figure 4.20

Notice that the sum of the delivery amount field is now called SumOfDeliv_Amt. Whenever you change the Total line in the Design Grid to an ***aggregate value*** (like Sum, Max, Min, First, etc.), the name of the field is changed to include the aggregate type you used. You can replace that title by creating an alias and writing a formula for the field. Remember that you cannot create an alias with the same name as an existing field in any of the tables you are using in the query. So in this example, you can't use the name Deliv_Amt as the alias. Therefore, we'll use something a little different. We're starting to do a lot of work on this query, so we'd better save it.

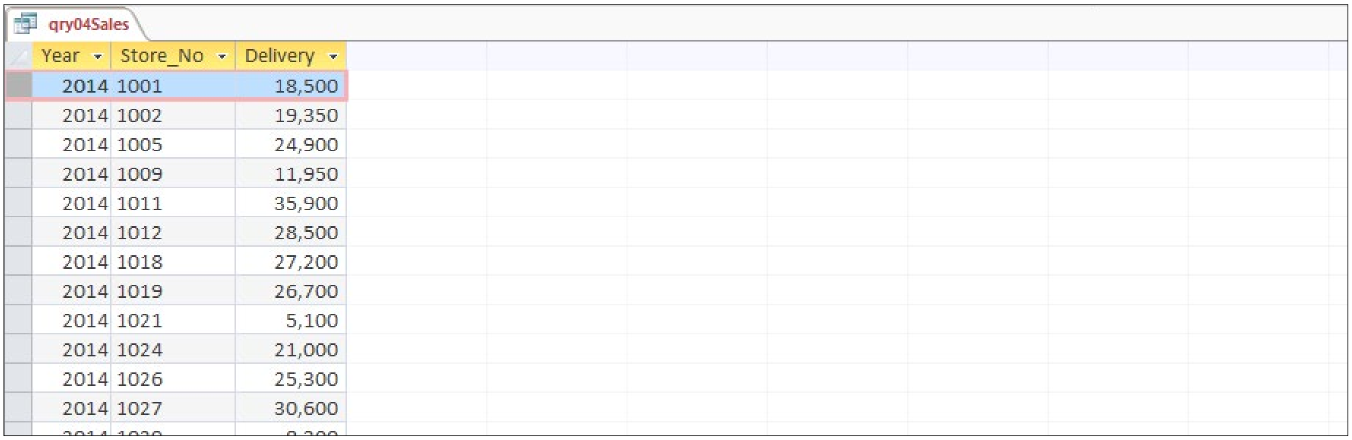
19. *Return to* **Design View**.
20. *Save the query as* ***qry04Sales***.
21. *Give the* **Deliv_Amt** *field an alias of* ***Delivery***
22. **Format** *that field to be* **Standard**, **zero decimal places**.



Field:	Year: Year([Sale_Date])	Store_No	Delivery: Deliv_Amt			
Table:		Stores	Sales_Journal			
Total:	Group By	Group By	Sum			
Sort:						
Show:	☑	☑	☑	□	□	□
Criteria:	Between 2014 And 2016					
or:						

Figure 4.21

23. **Run** *the query and adjust the field margins.*

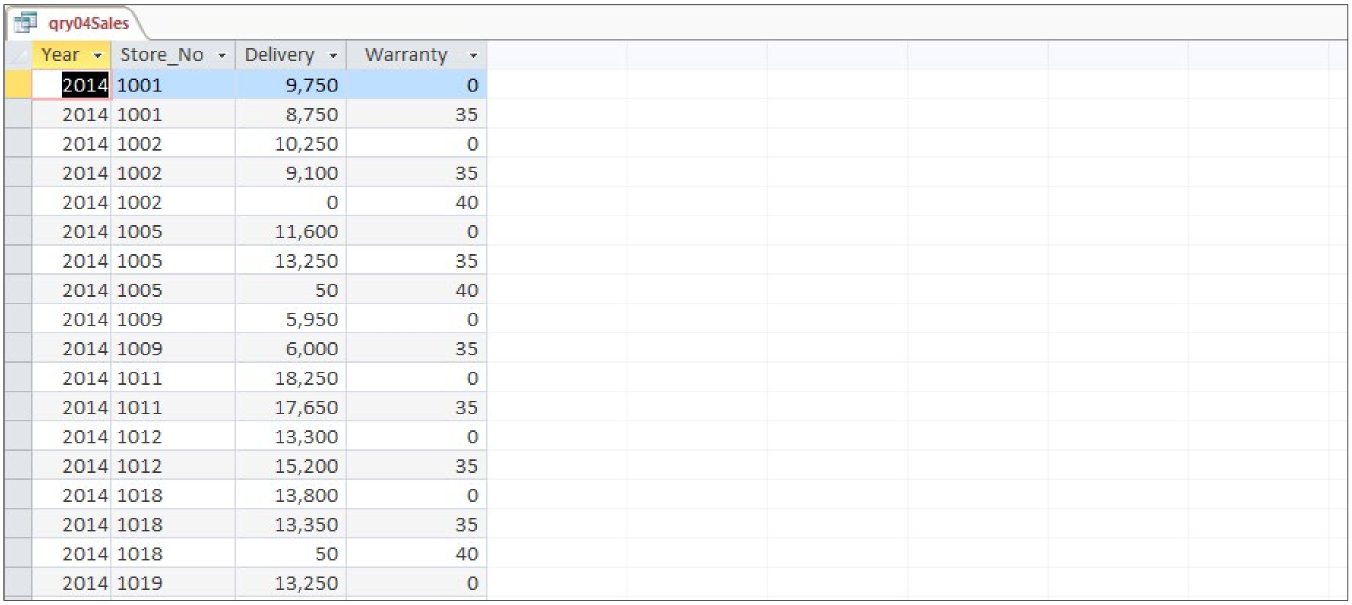


Year	Store_No	Delivery
2014	1001	18,500
2014	1002	19,350
2014	1005	24,900
2014	1009	11,950
2014	1011	35,900
2014	1012	28,500
2014	1018	27,200
2014	1019	26,700
2014	1021	5,100
2014	1024	21,000
2014	1026	25,300
2014	1027	30,600

Figure 4.22

Now the field is called Delivery.

24. *In* **qry04Sales**, *create a field called* ***Warranty*** *which is the summation of the* **Warr_Amt** *field.*
25. **Format** *the field as* **Standard**, **zero decimal places**.
26. **Run** *the query.*



Year	Store_No	Delivery	Warranty
2014	1001	9,750	0
2014	1001	8,750	35
2014	1002	10,250	0
2014	1002	9,100	35
2014	1002	0	40
2014	1005	11,600	0
2014	1005	13,250	35
2014	1005	50	40
2014	1009	5,950	0
2014	1009	6,000	35
2014	1011	18,250	0
2014	1011	17,650	35
2014	1012	13,300	0
2014	1012	15,200	35
2014	1018	13,800	0
2014	1018	13,350	35
2014	1018	50	40
2014	1019	13,250	0

Figure 4.23

Now we come to a tricky part. Your manager wants to see the Mattress sales, Pillow sales, and Other sales in their own columns. Other sales shouldn't be too terribly difficult, as there is an Item_Cd in the Sales_Journal table that reads "Other". But how can you distinguish between Mattress sales and Pillow sales? If you open the Sales_Journal table and look at the Item_Cd, the second letter in every code (except for Other) is either an "M" for Mattress or "P" for Pillow. So all you have to do is sum up all of the records where the second letter in the Item_Cd is M or P in different columns. Remember using the LEFT(), RIGHT() and MID() text functions in the Excel course? If not, it may be useful to review those before doing this next exercise. You can use text functions here to identify that letter from the Item_Cd. In the Sales_Journal, Other sales are recorded in Item_Cd field as Other. Let's do the Other sales first, then we'll tackle the Mattress and Pillow sales.

27. *Return to* **Design View**.
28. *In the column following* **Warranty**, *create an alias called* **Other** *and write the following formula:* ***IIF(Item_Cd="Other",Unit_Sale_Amt,0)***
29. **Format** *the field as* **Standard**, **zero decimal places** *and change the* **Total** *line to* **Sum**.

> ***Note***: *When writing a complex formula such as using an* **IIF()** *function, sometimes it is necessary to run the query before you can change the formatting of the field.*

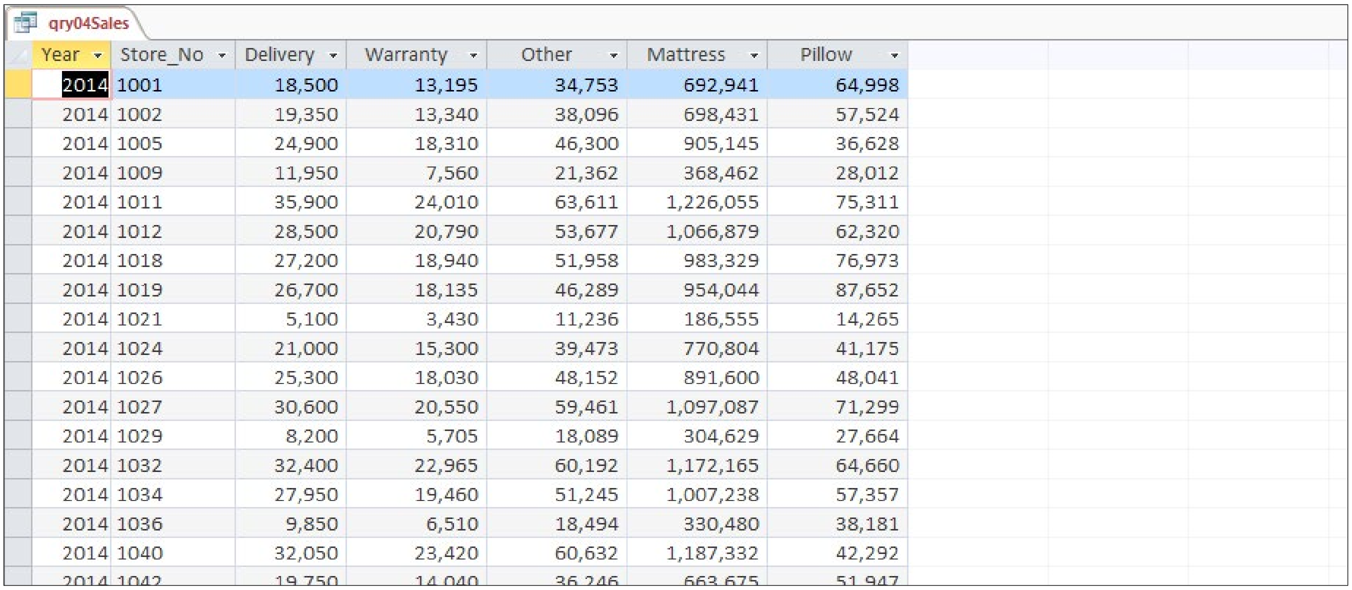
30. *In the next column, create the alias called* **Mattress** *and write the following formula:* ***IIF(Mid(Item_Cd,2,1)="M",Unit_Sale_Amt*qty*(1-disc_pct),0)***
31. *Format the* **Mattress** *field like the other amount fields and choose* **Sum** *on the* **Total** *line.*

Let's review this formula. Since you are already an expert in writing formulas in Excel, this one should pose no problem for you. This formula has basically the same syntax as it would in Excel. Here we're telling Access to return all records where the second letter in the Item_Cd is M. After we get those records, multiply the unit sale amount by the quantity multiplied by the inverse of the discount percent, which results in the sale amount net of discounts. Then we'll sum up all of the records using the Sum choice on the Total line. When I'm writing long formulas like this in the Design grid, it's convenient to press [Shift]+[F2] to zoom in and see the formula in a bigger screen. You can also right-click on the field and choose Zoom.

32. **Copy** *the* **Mattress** *formula over to the next column and change the criteria to reflect* ***Pillow*** *sales*

> ***Hint****: the second letter in the* **Item_Cd** *for* **Pillow** *sales is* **P**

33. **Format** *the field as* **Standard**, **zero decimal places** *and choose* **Sum** *on the* **Total** *line.*
34. **Run** *the query.*



Year	Store_No	Delivery	Warranty	Other	Mattress	Pillow
2014	1001	18,500	13,195	34,753	692,941	64,998
2014	1002	19,350	13,340	38,096	698,431	57,524
2014	1005	24,900	18,310	46,300	905,145	36,628
2014	1009	11,950	7,560	21,362	368,462	28,012
2014	1011	35,900	24,010	63,611	1,226,055	75,311
2014	1012	28,500	20,790	53,677	1,066,879	62,320
2014	1018	27,200	18,940	51,958	983,329	76,973
2014	1019	26,700	18,135	46,289	954,044	87,652
2014	1021	5,100	3,430	11,236	186,555	14,265
2014	1024	21,000	15,300	39,473	770,804	41,175
2014	1026	25,300	18,030	48,152	891,600	48,041
2014	1027	30,600	20,550	59,461	1,097,087	71,299
2014	1029	8,200	5,705	18,089	304,629	27,664
2014	1032	32,400	22,965	60,192	1,172,165	64,660
2014	1034	27,950	19,460	51,245	1,007,238	57,357
2014	1036	9,850	6,510	18,494	330,480	38,181
2014	1040	32,050	23,420	60,632	1,187,332	42,292
2014	1042	19,750	14,040	36,246	663,675	51,947

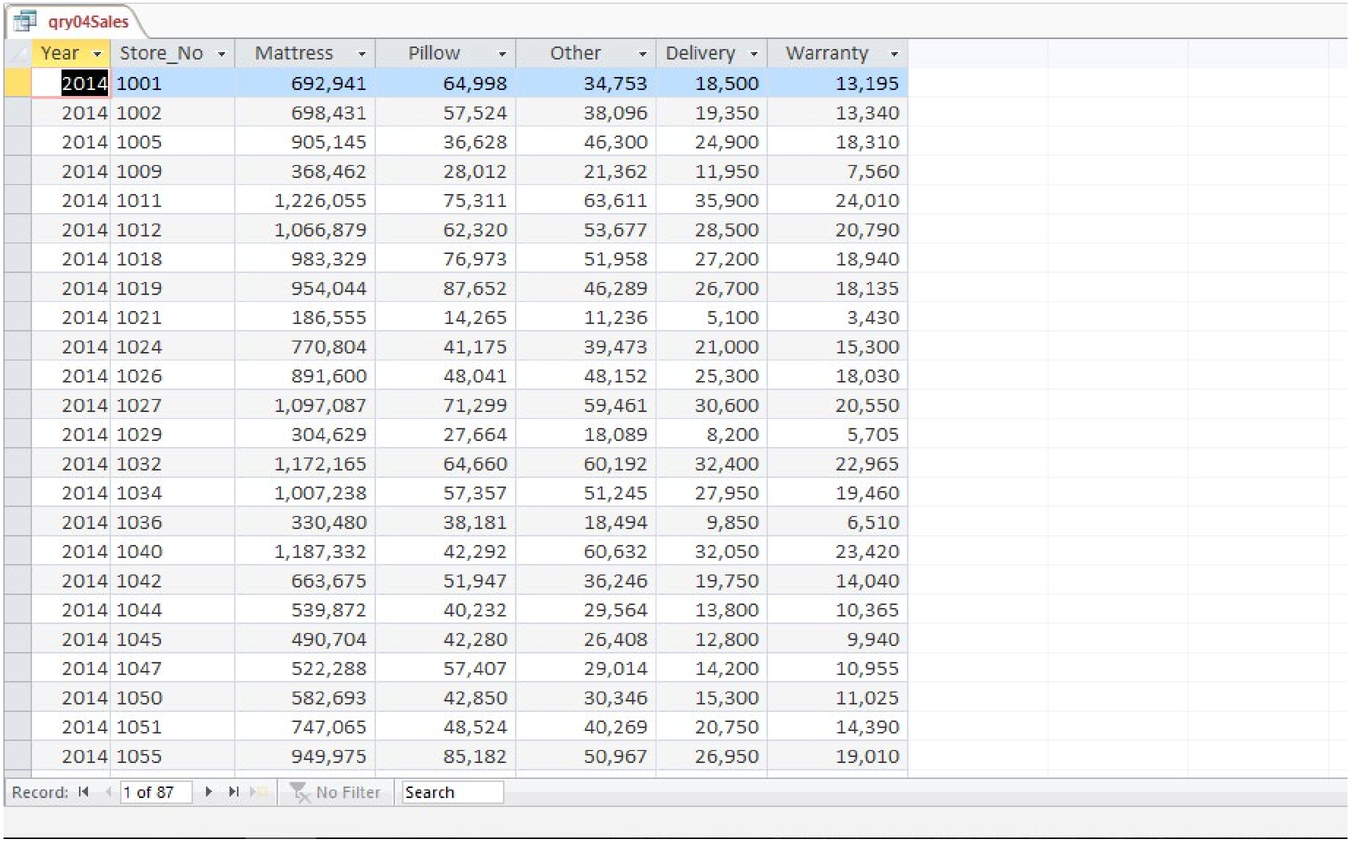
Figure 4.24

After you write the formula, you may have to run the query once and then go back to Design View to do the formatting. Access sometimes likes for you to run the query first, and then it will decide on which formatting options it will make available to you, depending on the type of data returned by the query.

Now let's move some of the columns around so that the more important sources of sales appear first.

35. *Return to* **Design View**.
36. *Place your cursor in the thin gray box above the* **Mattress** *calculation and it will turn to a down arrow. Click on the down arrow to select the entire field.*

37. *Drag the* **Mattress** *field to the left to where it is the first field after* **Store_No**.
38. *Move the other fields where they are in this order:* **Pillow**, **Other**, **Delivery**, **Warranty**, *and* **Run** *the query.*



qry04Sales

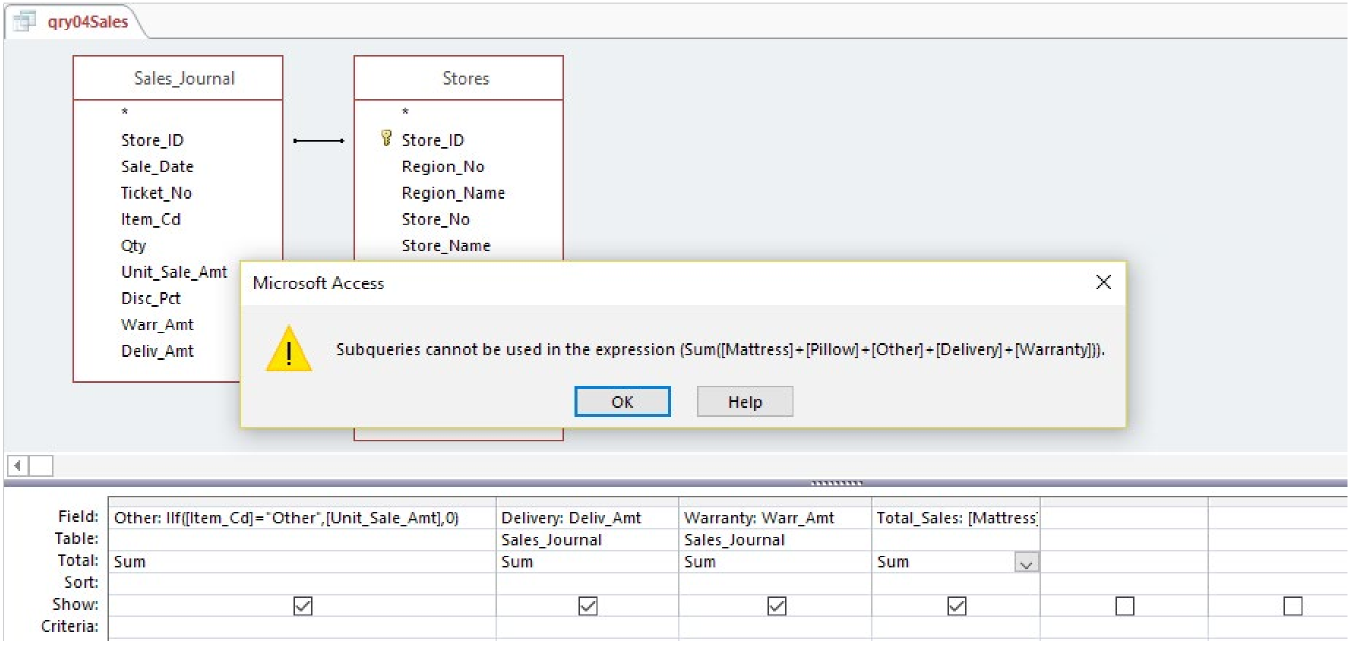
Year	Store_No	Mattress	Pillow	Other	Delivery	Warranty
2014	1001	692,941	64,998	34,753	18,500	13,195
2014	1002	698,431	57,524	38,096	19,350	13,340
2014	1005	905,145	36,628	46,300	24,900	18,310
2014	1009	368,462	28,012	21,362	11,950	7,560
2014	1011	1,226,055	75,311	63,611	35,900	24,010
2014	1012	1,066,879	62,320	53,677	28,500	20,790
2014	1018	983,329	76,973	51,958	27,200	18,940
2014	1019	954,044	87,652	46,289	26,700	18,135
2014	1021	186,555	14,265	11,236	5,100	3,430
2014	1024	770,804	41,175	39,473	21,000	15,300
2014	1026	891,600	48,041	48,152	25,300	18,030
2014	1027	1,097,087	71,299	59,461	30,600	20,550
2014	1029	304,629	27,664	18,089	8,200	5,705
2014	1032	1,172,165	64,660	60,192	32,400	22,965
2014	1034	1,007,238	57,357	51,245	27,950	19,460
2014	1036	330,480	38,181	18,494	9,850	6,510
2014	1040	1,187,332	42,292	60,632	32,050	23,420
2014	1042	663,675	51,947	36,246	19,750	14,040
2014	1044	539,872	40,232	29,564	13,800	10,365
2014	1045	490,704	42,280	26,408	12,800	9,940
2014	1047	522,288	57,407	29,014	14,200	10,955
2014	1050	582,693	42,850	30,346	15,300	11,025
2014	1051	747,065	48,524	40,269	20,750	14,390
2014	1055	949,975	85,182	50,967	26,950	19,010

Record: 1 of 87 No Filter Search

Figure 4.25

Now that you have all of the individual sales categories, you need to have a Total column.

39. *Return to* **Design View** *and create a new field using the following formula:* ***Total_Sales: Mattress+Pillow+Other+Delivery+Warranty***
40. *Make the* **Total** *line* **Sum** *and* **Run** *the query.*



Subqueries cannot be used in the expression (Sum([Mattress]+[Pillow]+[Other]+[Delivery]+[Warranty])).

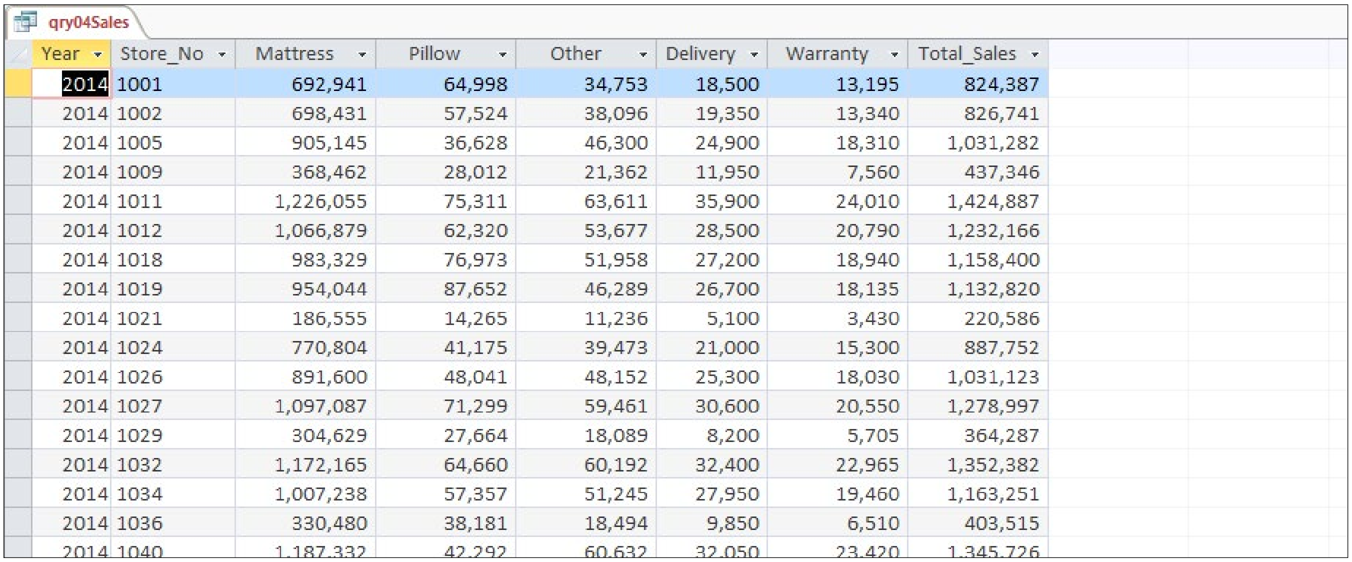
Field:	Other: IIf([Item_Cd]="Other",[Unit_Sale_Amt],0)	Delivery: Deliv_Amt	Warranty: Warr_Amt	Total_Sales: [Mattress]		
Table:		Sales_Journal	Sales_Journal			
Total:	Sum	Sum	Sum	Sum		
Sort:						
Show:	☑	☑	☑	☑	□	□
Criteria:						

Figure 4.26

Subqueries

Oops! Another error. This one is telling us that we can't have subqueries in the formula. A ***subquery*** is a query within a query. But that is essentially what we want to do, right? To solve this problem, you have to copy the formula we wrote for each of those fields, and use the formula in place of the field name.

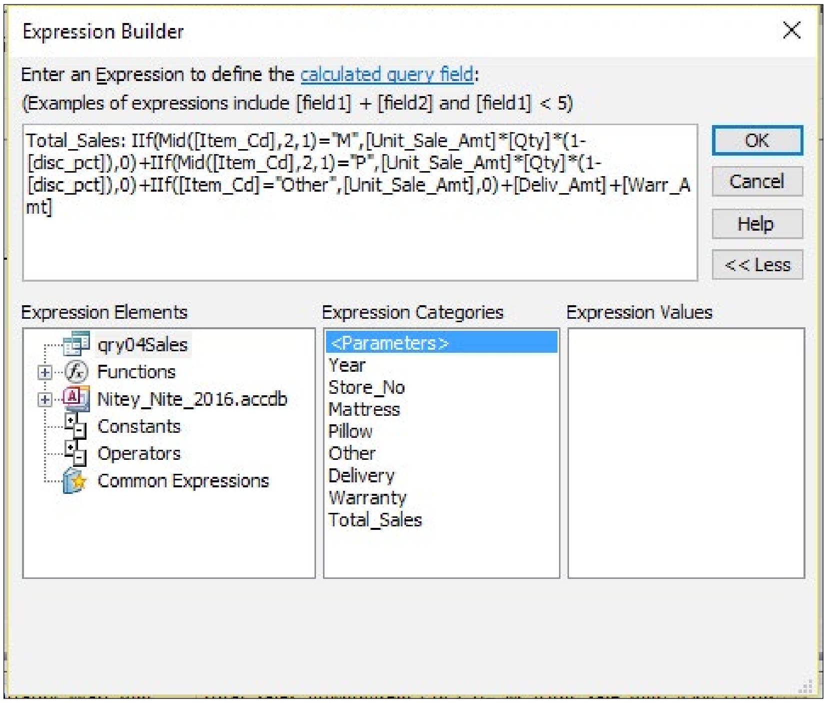
1. *Click* **OK** *to return to* **Design View**.
2. *Copy the formula in the* **Mattress** *field and replace "***[Mattress]***" in the* **Total_Sales** *field with the formula.*
3. *Do the same for each corresponding field and formula.*
4. *Use* **Sum** *in the* **Total** *line.*
5. **Format** *like the other fields.*
6. **Save** *and* **Run** *the query.*



Year	Store_No	Mattress	Pillow	Other	Delivery	Warranty	Total_Sales
2014	1001	692,941	64,998	34,753	18,500	13,195	824,387
2014	1002	698,431	57,524	38,096	19,350	13,340	826,741
2014	1005	905,145	36,628	46,300	24,900	18,310	1,031,282
2014	1009	368,462	28,012	21,362	11,950	7,560	437,346
2014	1011	1,226,055	75,311	63,611	35,900	24,010	1,424,887
2014	1012	1,066,879	62,320	53,677	28,500	20,790	1,232,166
2014	1018	983,329	76,973	51,958	27,200	18,940	1,158,400
2014	1019	954,044	87,652	46,289	26,700	18,135	1,132,820
2014	1021	186,555	14,265	11,236	5,100	3,430	220,586
2014	1024	770,804	41,175	39,473	21,000	15,300	887,752
2014	1026	891,600	48,041	48,152	25,300	18,030	1,031,123
2014	1027	1,097,087	71,299	59,461	30,600	20,550	1,278,997
2014	1029	304,629	27,664	18,089	8,200	5,705	364,287
2014	1032	1,172,165	64,660	60,192	32,400	22,965	1,352,382
2014	1034	1,007,238	57,357	51,245	27,950	19,460	1,163,251
2014	1036	330,480	38,181	18,494	9,850	6,510	403,515
2014	1040	1,187,332	42,292	60,632	32,050	23,420	1,345,726

Figure 4.27

It may take a few seconds to run because you're doing a lot of calculations on more than 140,000 records. That ain't no walk in the park, so Access may take some time to process the calculations. Mine took five seconds to calculate the eighty seven records, which represents the sales for 29 stores for three years. Of course, since I am a big cheapskate, I have a computer that I believe was manufactured just after the United States declared its independence from England. If you have a problem with the Total_Sales formula, check it with this one:



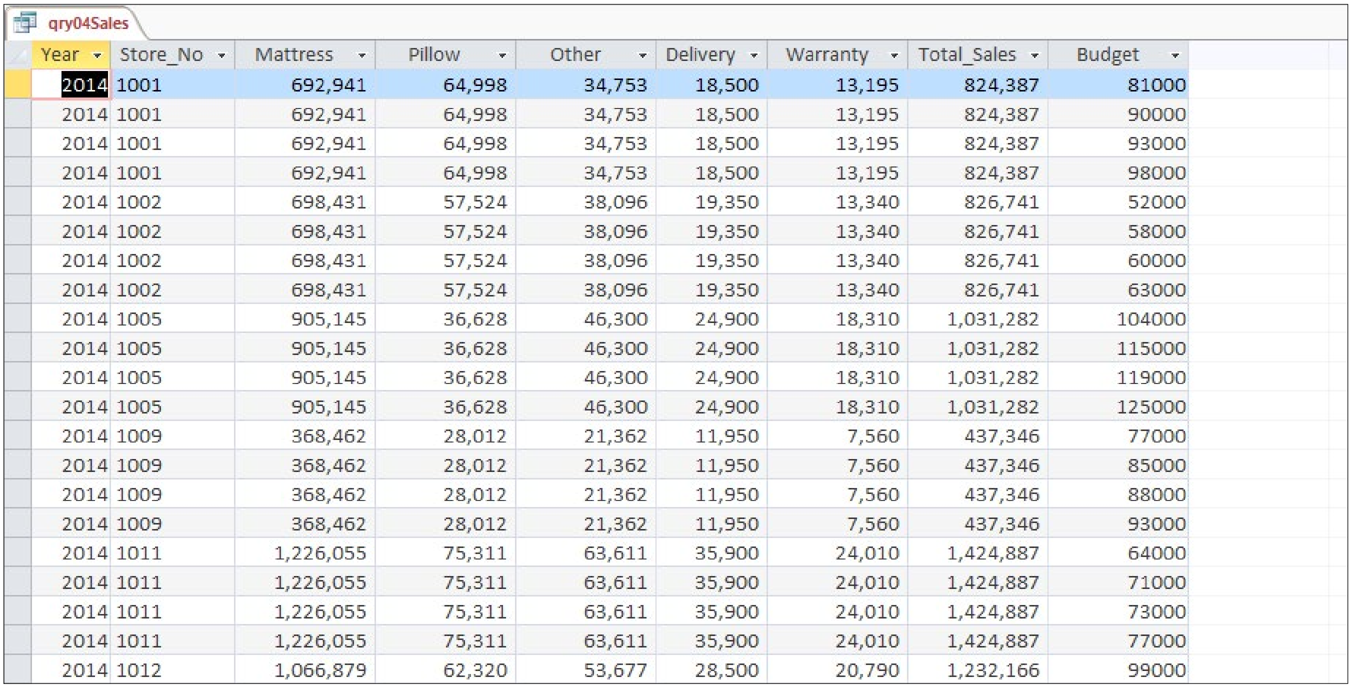
Enter an Expression to define the calculated query field:
(Examples of expressions include [field1] + [field2] and [field1] < 5)

```
Total_Sales: IIf(Mid([Item_Cd],2,1)="M",[Unit_Sale_Amt]*[Qty]*(1-
[disc_pct]),0)+IIf(Mid([Item_Cd],2,1)="P",[Unit_Sale_Amt]*[Qty]*(1-
[disc_pct]),0)+IIf([Item_Cd]="Other",[Unit_Sale_Amt],0)+[Deliv_Amt]+[Warr_A
mt]
```

Figure 4.28

At this point, we have a query with all of the sales by year from each store. Now we want to see how each store performed as compared with its budget. We have the budget in a table (Sales_Budget), so all we have to do is to bring in the Sales_Budget table, create a relationship, and we're done, right? Let's do it.

7. *Return to* **Design View**.
8. *Bring in the* **Sales_Budget** *table and create a join on the* **Store_No** *with the* **Stores** *table.*
9. *Bring in the* **Budget** *field next to* **Total_Sales** *(leave the* **Total** *line on* **Group By** *for the Budget)*
10. **Run** *the query.*



Year	Store_No	Mattress	Pillow	Other	Delivery	Warranty	Total_Sales	Budget
2014	1001	692,941	64,998	34,753	18,500	13,195	824,387	81000
2014	1001	692,941	64,998	34,753	18,500	13,195	824,387	90000
2014	1001	692,941	64,998	34,753	18,500	13,195	824,387	93000
2014	1001	692,941	64,998	34,753	18,500	13,195	824,387	98000
2014	1002	698,431	57,524	38,096	19,350	13,340	826,741	52000
2014	1002	698,431	57,524	38,096	19,350	13,340	826,741	58000
2014	1002	698,431	57,524	38,096	19,350	13,340	826,741	60000
2014	1002	698,431	57,524	38,096	19,350	13,340	826,741	63000
2014	1005	905,145	36,628	46,300	24,900	18,310	1,031,282	104000
2014	1005	905,145	36,628	46,300	24,900	18,310	1,031,282	115000
2014	1005	905,145	36,628	46,300	24,900	18,310	1,031,282	119000
2014	1005	905,145	36,628	46,300	24,900	18,310	1,031,282	125000
2014	1009	368,462	28,012	21,362	11,950	7,560	437,346	77000
2014	1009	368,462	28,012	21,362	11,950	7,560	437,346	85000
2014	1009	368,462	28,012	21,362	11,950	7,560	437,346	88000
2014	1009	368,462	28,012	21,362	11,950	7,560	437,346	93000
2014	1011	1,226,055	75,311	63,611	35,900	24,010	1,424,887	64000
2014	1011	1,226,055	75,311	63,611	35,900	24,010	1,424,887	71000
2014	1011	1,226,055	75,311	63,611	35,900	24,010	1,424,887	73000
2014	1011	1,226,055	75,311	63,611	35,900	24,010	1,424,887	77000
2014	1012	1,066,879	62,320	53,677	28,500	20,790	1,232,166	99000

Figure 4.29

Hmmm. Something doesn't look right. It looks like the stores have some repeating information. Look at Store_No 1001. All of the fields have exactly the same data, except the Budget field. There were also 336 records returned by the query, which is many more than the 87 records we got before we added in the Sales_Budget table.

As I stated before, if you have repeating data in a query, you probably have a problem with a join somewhere. That is precisely the problem here. If you examine the Sales_Budget table, you will see that for each Store_No, there are four years of budgets. When you created the join with the Sales_Budget, you needed to also create a join on the year. We didn't do that, so you got one record from the query and one record for each of the four years per location from the Sales_Budget table. The answer is to create a join on the Year field. That's kind of hard to do in this query, as the only table that has a Year field is the Sales_Budget table. However, we WILL come up with a solution.

In this next exercise, we need to create a query similar to the Sales_Journal table with a Year field. The qry04Sales query already has a Year field, so we will save that query without the Sales_Budget table and create a relationship between qry04Sales and the Sales_Budget table. Previously, I said that a query is simply a virtual table, and you can create a join on a query just like you can on a table. Let's try it.

1. *Return to* **Design View***.*
2. *Delete the* **Sales_Budget** *table from the* **Table Area** *by clicking anywhere in the* **Sales_Budget** *table and press the* **[Delete]** *key. You can also right-click on the table and choose* **Remove Table***.*
3. *Save* **qry04Sales***.*
4. **Create** *a new query.*
5. *In the* **Show Table** *dialog box, bring in the* **Sales_Budget** *table, but don't exit out of the* **Show Table** *dialog box yet.*
6. *Next, click on the* **Queries** *tab of the* **Show Table** *dialog box, choose* **qry04Sales***, click on the* **Add** *button, then click on the* **Close** *button.*

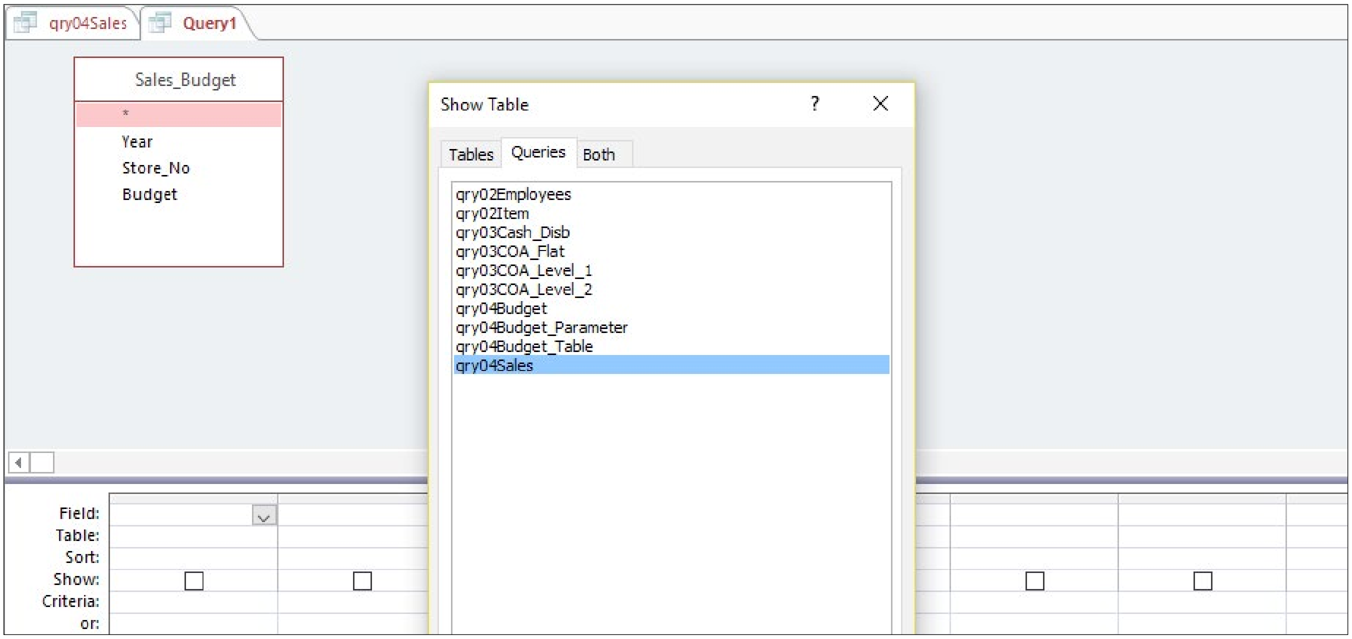
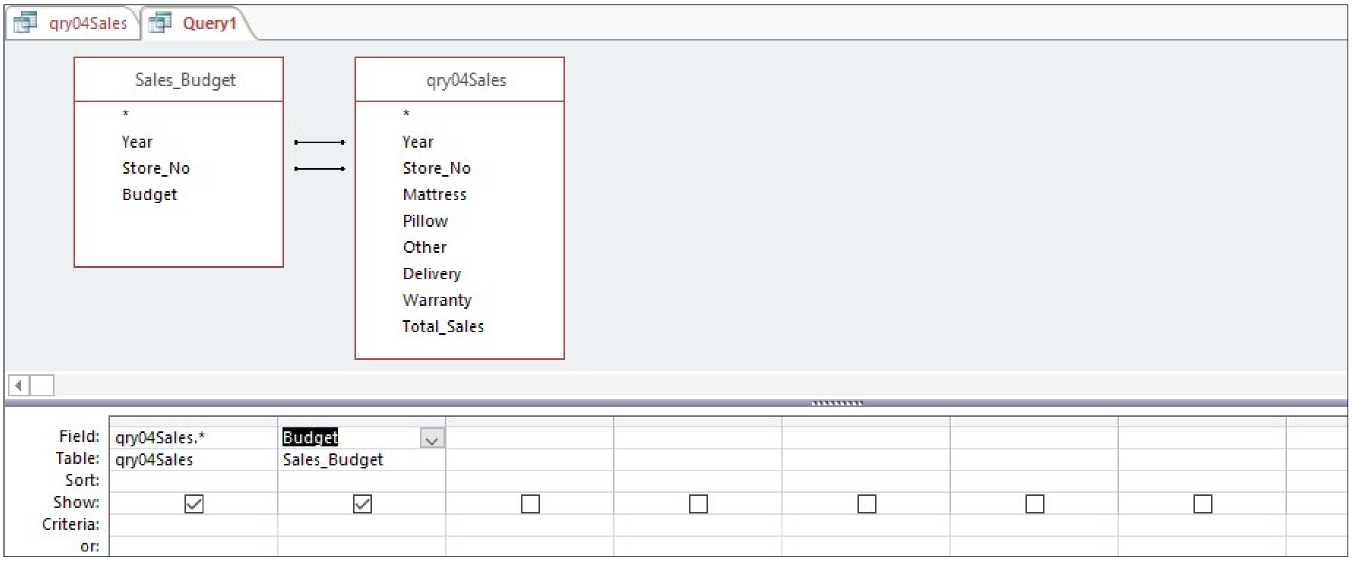


Figure 4.30

7. *Create joins between the* **Sales_Budget** *table and the* **qry04Sales** *query on* **Year** *and* **Store_No***.*
8. *Bring in all fields from* **qry04Sales** *and the* **Budget** *field from* **Sales_Budget***.*

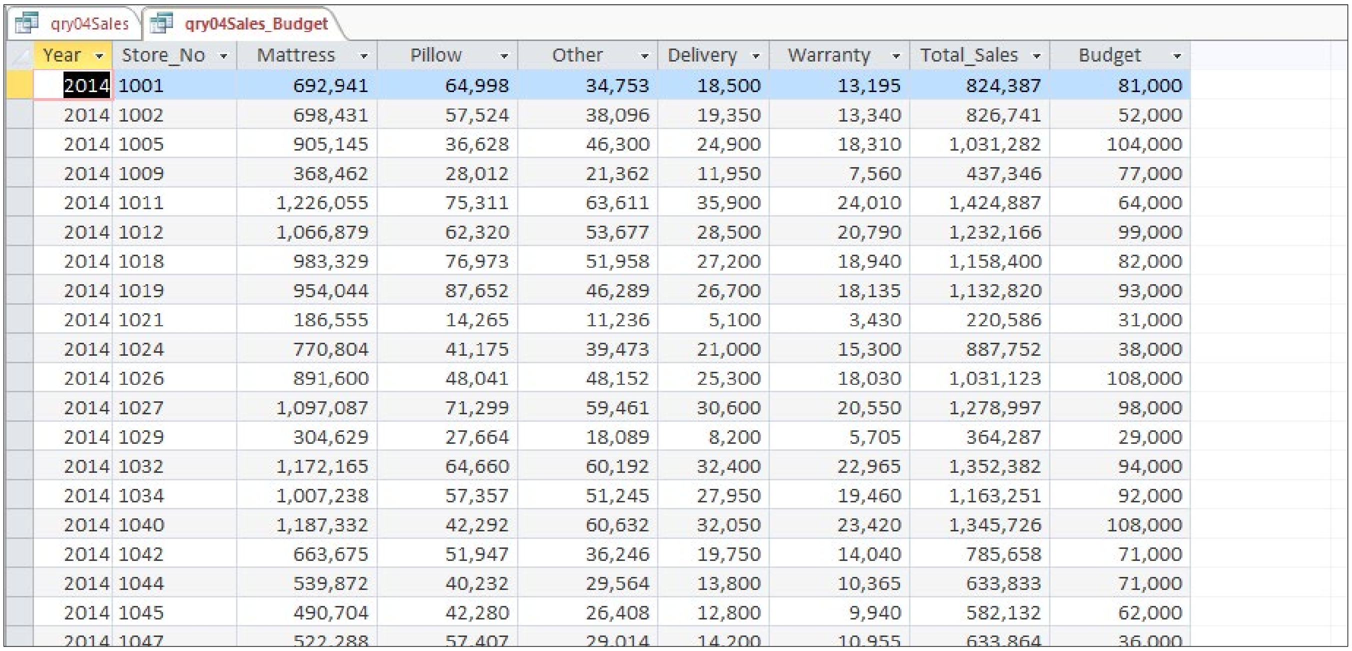
> ***Note****: Double-clicking the * above the field names brings in all fields under the title of* **qry04Sales.*** *within a single column for a compact view. If you run the query, all fields populate. Only use this option when you want all fields from a query or table included.*



Field:	qry04Sales.*	Budget					
Table:	qry04Sales	Sales_Budget					
Sort:							
Show:	☑	☑	☐	☐	☐	☐	☐
Criteria:							
or:							

Figure 4.31

9. **Format** *all amount fields as* **Standard**, **zero decimal places**.
10. *Save the query as* ***qry04Sales_Budget***.
11. **Run** *the query.*



Year	Store_No	Mattress	Pillow	Other	Delivery	Warranty	Total_Sales	Budget
2014	1001	692,941	64,998	34,753	18,500	13,195	824,387	81,000
2014	1002	698,431	57,524	38,096	19,350	13,340	826,741	52,000
2014	1005	905,145	36,628	46,300	24,900	18,310	1,031,282	104,000
2014	1009	368,462	28,012	21,362	11,950	7,560	437,346	77,000
2014	1011	1,226,055	75,311	63,611	35,900	24,010	1,424,887	64,000
2014	1012	1,066,879	62,320	53,677	28,500	20,790	1,232,166	99,000
2014	1018	983,329	76,973	51,958	27,200	18,940	1,158,400	82,000
2014	1019	954,044	87,652	46,289	26,700	18,135	1,132,820	93,000
2014	1021	186,555	14,265	11,236	5,100	3,430	220,586	31,000
2014	1024	770,804	41,175	39,473	21,000	15,300	887,752	38,000
2014	1026	891,600	48,041	48,152	25,300	18,030	1,031,123	108,000
2014	1027	1,097,087	71,299	59,461	30,600	20,550	1,278,997	98,000
2014	1029	304,629	27,664	18,089	8,200	5,705	364,287	29,000
2014	1032	1,172,165	64,660	60,192	32,400	22,965	1,352,382	94,000
2014	1034	1,007,238	57,357	51,245	27,950	19,460	1,163,251	92,000
2014	1040	1,187,332	42,292	60,632	32,050	23,420	1,345,726	108,000
2014	1042	663,675	51,947	36,246	19,750	14,040	785,658	71,000
2014	1044	539,872	40,232	29,564	13,800	10,365	633,833	71,000
2014	1045	490,704	42,280	26,408	12,800	9,940	582,132	62,000
2014	1047	522,288	57,407	29,014	14,200	10,955	633,864	36,000

Figure 4.32

Now it looks better. Whenever I do queries like this, I ALWAYS check my numbers back to a base file, just in case one of those annoying Sarbanes-Oxley auditors wants to check my numbers. In this case, it is probably a good idea, so let's do it.

12. *Click on the blank box in the upper-left corner of the record set to select all records.*
13. **Copy** *the records ([Ctrl]+[c] or right-click and choose Copy) and* **Paste** *them into an* **Excel** *spreadsheet.*
14. *Open* **qry04Sales** *and copy the results from that query onto an* **Excel** *spreadsheet.*
15. **Sum** *up the* **Total_Sales** *numbers in both spreadsheets and compare the totals.*

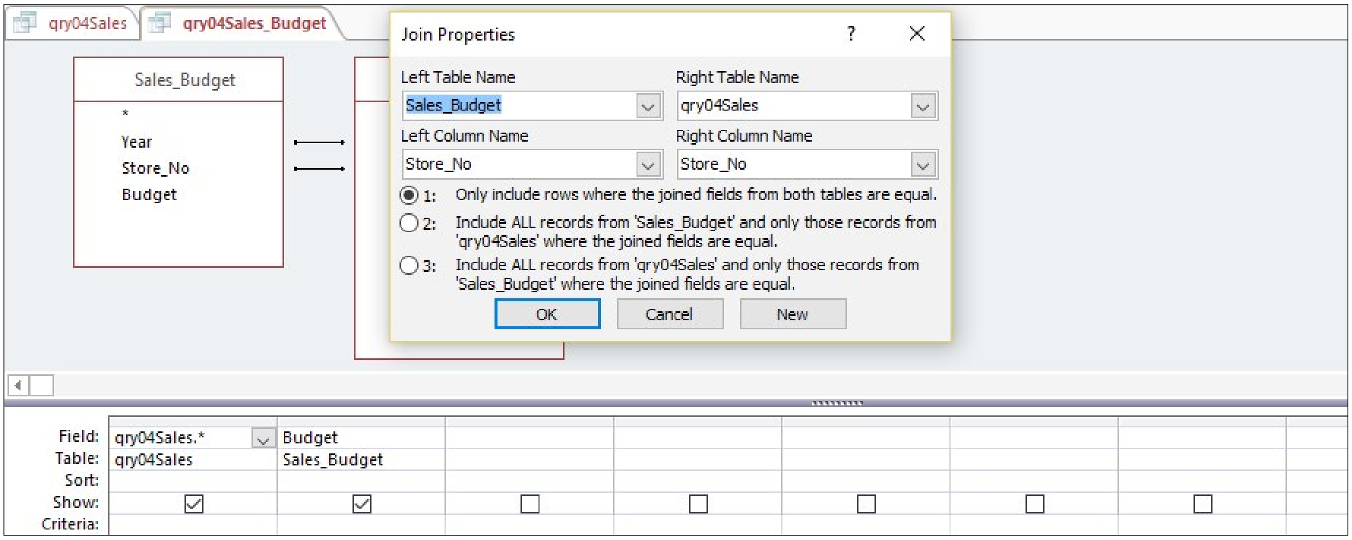
One-Way Joins

If you did everything correctly, the numbers will NOT match. "Huh?", you say? Why don't they match? That's where your expert auditing skills come in. If you compare the years and store numbers from both sets, you will find that there is one store number, 1036, that does not exist in the results from qry04Sales_Budget. Why not? Answer: If you open the Sales_Budget table, you'll see that Store_No 1036 does not exist in that table. That is because there is not a budget established for that store. Also notice that there were only 84 records returned in qry04Sales_Budget, but there were 87 records returned in qry04Sales. When you create a simple join (also called an inner join), it tells Access to return a record set of the matched records in both tables. In this case, you want to return all of the records from qry04Sales and only the matching records from the Sales_Budget table. This is where you employ the one-way join.

A one-way join selects all of the records from one table or query and the matching records from the other query. Let me show you how to do that.

1. *Go back to the* **Design View** *of* **qry04Sales_Budget***.*
2. *Right-click on the line that connects the* **Store_No** *fields and choose* **Join Properties***.*

Note that you have to click directly on the line to get the correct right-click options to display. You may have to try it a few times to get the right option list to appear. Sometimes it helps to reposition one table and click on the join line when it is offset a bit. You can also double-click on the line to display the Join Properties dialog box.



Left Table Name: Sales_Budget
Right Table Name: qry04Sales
Left Column Name: Store_No
Right Column Name: Store_No

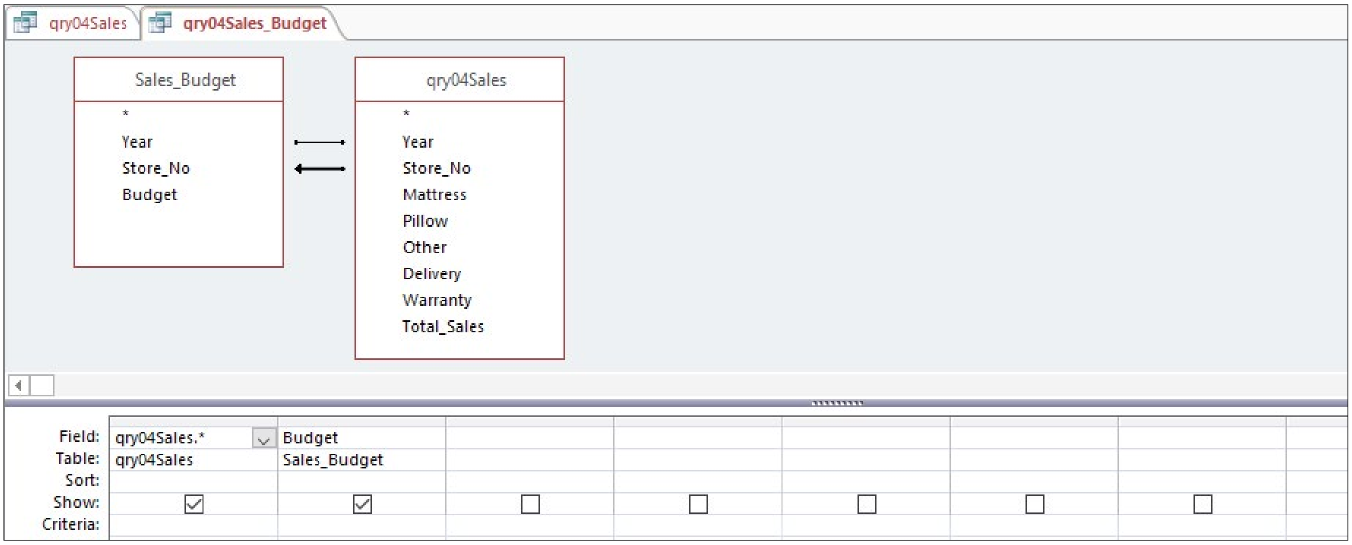
● 1: Only include rows where the joined fields from both tables are equal.
○ 2: Include ALL records from 'Sales_Budget' and only those records from 'qry04Sales' where the joined fields are equal.
○ 3: Include ALL records from 'qry04Sales' and only those records from 'Sales_Budget' where the joined fields are equal.

Field:	qry04Sales.*	Budget					
Table:	qry04Sales	Sales_Budget					
Sort:							
Show:	☑	☑	□	□	□	□	□
Criteria:							

Figure 4.33

3. *In the* **Join Properties** *dialog box, choose option* **3** *and click* **OK**.

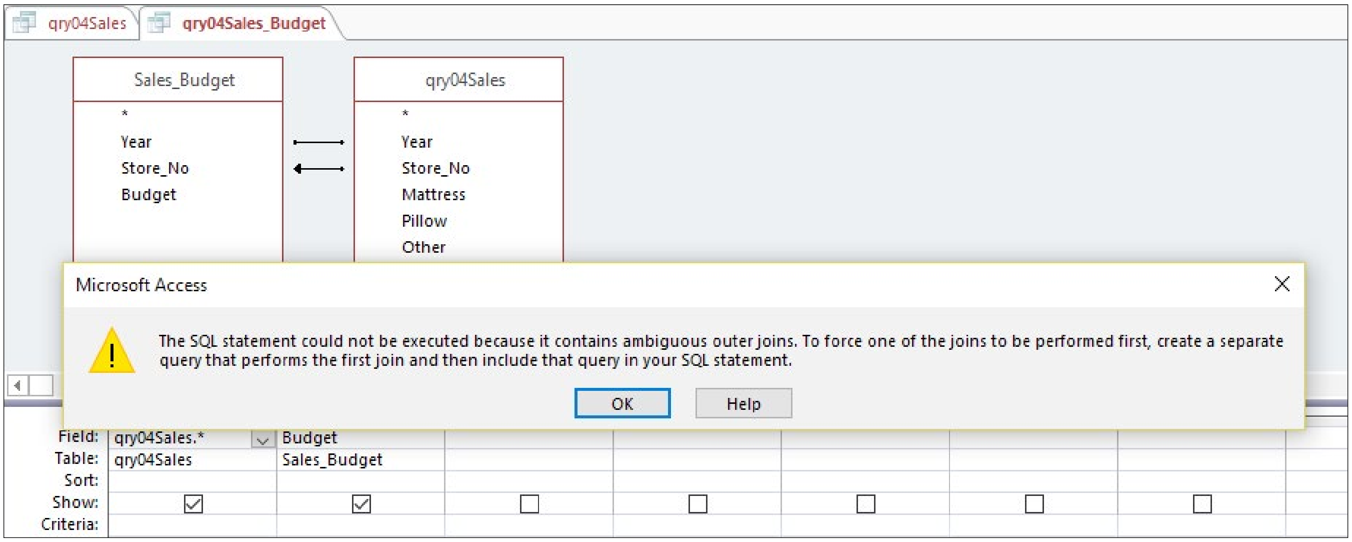
In the Join Properties box, you can do a one-way join between either table. In our case, we want to include all of the records in qry04Sales and the matched (or corresponding) records in the Sales_Budget table.



Field:	qry04Sales.*	Budget					
Table:	qry04Sales	Sales_Budget					
Sort:							
Show:	☑	☑	□	□	□	□	□
Criteria:							

Figure 4.34

4. **Run** *the query.*



Microsoft Access

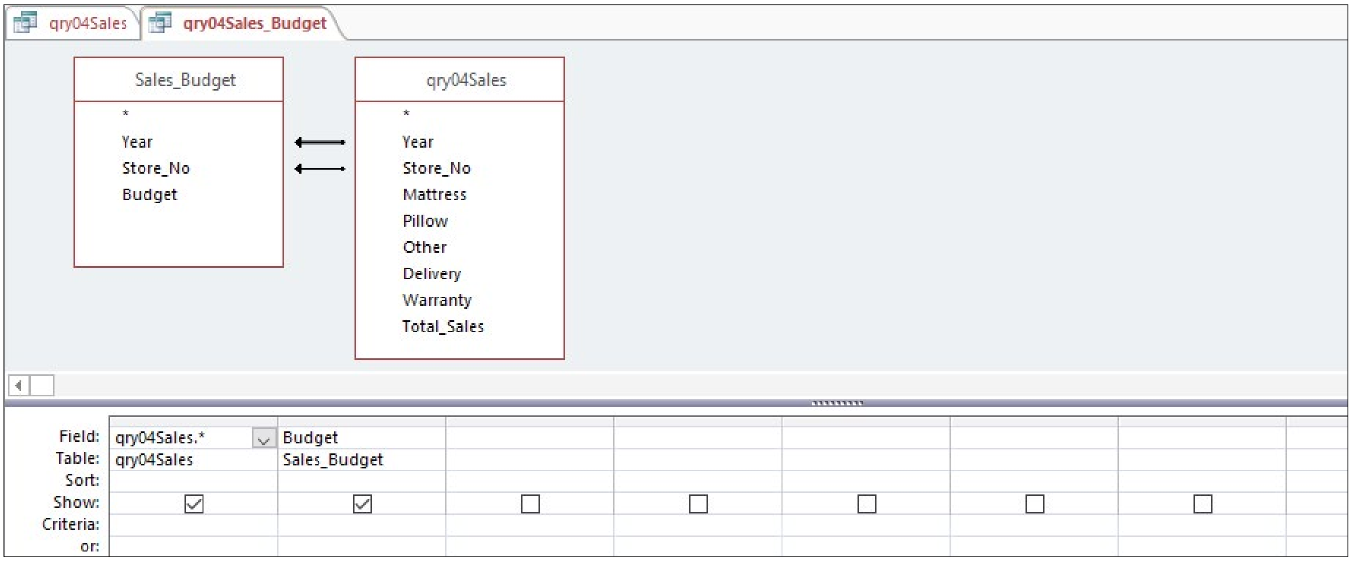
The SQL statement could not be executed because it contains ambiguous outer joins. To force one of the joins to be performed first, create a separate query that performs the first join and then include that query in your SQL statement.

Field:	qry04Sales.*	Budget					
Table:	qry04Sales	Sales_Budget					
Sort:							
Show:	☑	☑	□	□	□	□	□
Criteria:							

Figure 4.35

Dang it! Another error! A one-way join is also called an outer join, and you can’t have different types of joins between the tables in the same query. You either have to create different queries or do the same type of joins in a single query.

5. *Return to* **Design View** *and create a one-way join on the* **Year** *field.*



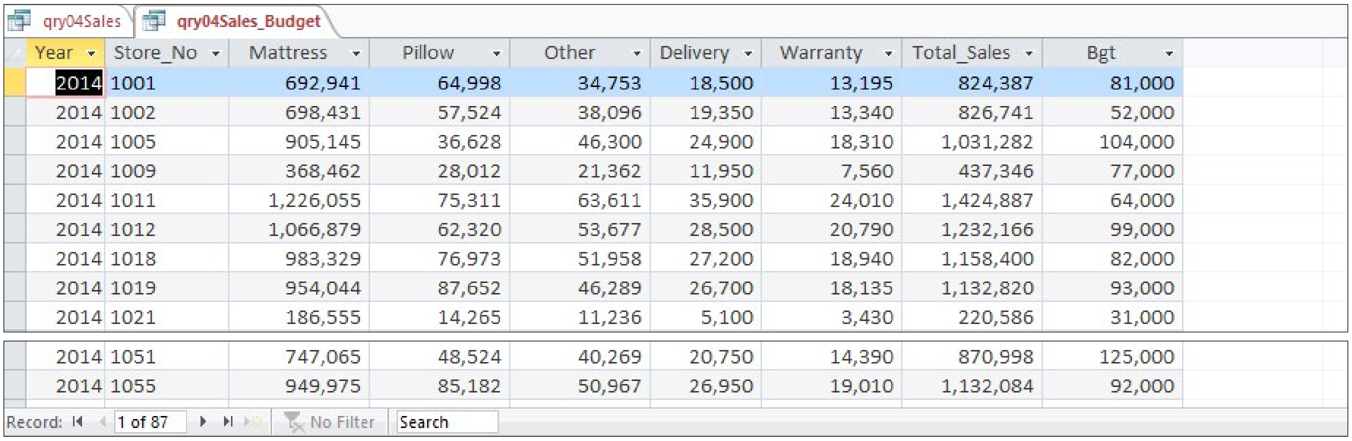
Field:	qry04Sales.*	Budget					
Table:	qry04Sales	Sales_Budget					
Sort:							
Show:	☑	☑	☐	☐	☐	☐	☐
Criteria:							
or:							

Figure 4.36

6. **Run** *the query.*

You should now see a record set of 87 records, and the Budget field for Store_No 1036 is NULL for all years. Remember that a NULL field contains nothing – it is not a zero. We want each Budget field to contain some number (as NULL values can be problematic when performing calculations on the field), so all you have to do is write a formula to replace the NULL values.

7. *Return to* **Design View**.
8. *In place of the* **Budget** *field, create an alias called* **Bgt** *and write the following formula:* ***IIF(ISNULL([Budget]),0,[Budget])***
9. **Run** *the query.*



Year	Store_No	Mattress	Pillow	Other	Delivery	Warranty	Total_Sales	Bgt
2014	1001	692,941	64,998	34,753	18,500	13,195	824,387	81,000
2014	1002	698,431	57,524	38,096	19,350	13,340	826,741	52,000
2014	1005	905,145	36,628	46,300	24,900	18,310	1,031,282	104,000
2014	1009	368,462	28,012	21,362	11,950	7,560	437,346	77,000
2014	1011	1,226,055	75,311	63,611	35,900	24,010	1,424,887	64,000
2014	1012	1,066,879	62,320	53,677	28,500	20,790	1,232,166	99,000
2014	1018	983,329	76,973	51,958	27,200	18,940	1,158,400	82,000
2014	1019	954,044	87,652	46,289	26,700	18,135	1,132,820	93,000
2014	1021	186,555	14,265	11,236	5,100	3,430	220,586	31,000
2014	1051	747,065	48,524	40,269	20,750	14,390	870,998	125,000
2014	1055	949,975	85,182	50,967	26,950	19,010	1,132,084	92,000

Figure 4.37

You should now get a record set with 87 records, and Store_No 1036 should have a zero budget.

10. **Save** *and close the* **qry04Sales** *and* **qry04Sales_Budget** *queries.*

> ***Note****: When closing the queries, you may get a message that tells you that you've copied a large amount of data onto the* **Clipboard** *and asks if you want to save it on the Clipboard. If this happens, click* **No** *as you don't need to save the data.*
>
> ***Review Questions****: It is now time to complete the hands-on Review Questions. Log on to www.ExcelCEO.com with your Email and Password, click on the* **Access 2016 and SQL Review Questions, Chapter 4, Section 2 of 2** *option and complete the review questions.*

Conclusion

This was a hard chapter, but a very important one. I can't stress enough the importance of making sure that your joins are correct when you are writing complex formulas. This is one of the more difficult chapters in this course, but critically important in your programming education. In the next chapter, we'll go over action queries and macros.

In this chapter, you learned Access's version of the IF() function, the IIF() function. You learned how to customize a query to accept user-inputted values by creating a Parameter Query. You learned how to create an Assumptions table similar to the Assumptions page you did in Excel, and to use the table in a query. You learned about grouping using the Total icon in the query design grid. You explored subqueries, and saw how creating a one-way join can help in auditing tables and queries.

Chapter Exam

You can now go to www.ExcelCEO.com, click on Sign In, log in and take the exam. Make sure that you take the exam on the same device on which you completed the practice files, as some of the questions on the exam may refer to some of the completed projects. Chapter exams are intended to be hands-on.

CHAPTER FIVE — ACTION QUERIES AND MACROS

In this chapter, you will:

- Identify the different types of Action Queries
- Recognize how to create a Crosstab Query
- Identify the different examples of Action Queries, including Make Table, Update, Append, and Delete Queries
- Select the correct syntax for writing SQL Code by creating a Drop Table Query
- Determine the actions to automate the execution of several Action Queries in one Macro

CPE Credits possible for this chapter: 3

Introduction

The first three words in a successful IT system are automation, automation, and automation. When I went to a new company in 1996 as Manager of Reporting, I took over an Excel report that someone else had developed. This person would download data from an AS/400 system and then import it into an Excel spreadsheet. He would sort the database by social security number, then manually insert a row beneath each change in social security number, and then write a formula to add up four columns of data. This was a very manual process which took twenty-four hours (three business days) to do every month. This was in the dark ages (before PivotTables), and I showed him how to use Data Tables (the predecessor to PivotTables). That trick alone cut the development time for the report from twenty-four hours to five hours. After I learned how to manipulate data and tables with Access, I cut it down from five hours to about ninety seconds, and that was almost all computer run time.

The goal with any report generation system is to eliminate as much human intervention as possible, and action queries and macros in Access can make that process much easier and faster.

If you look in any query Design View, you will see that there are nine types of queries:

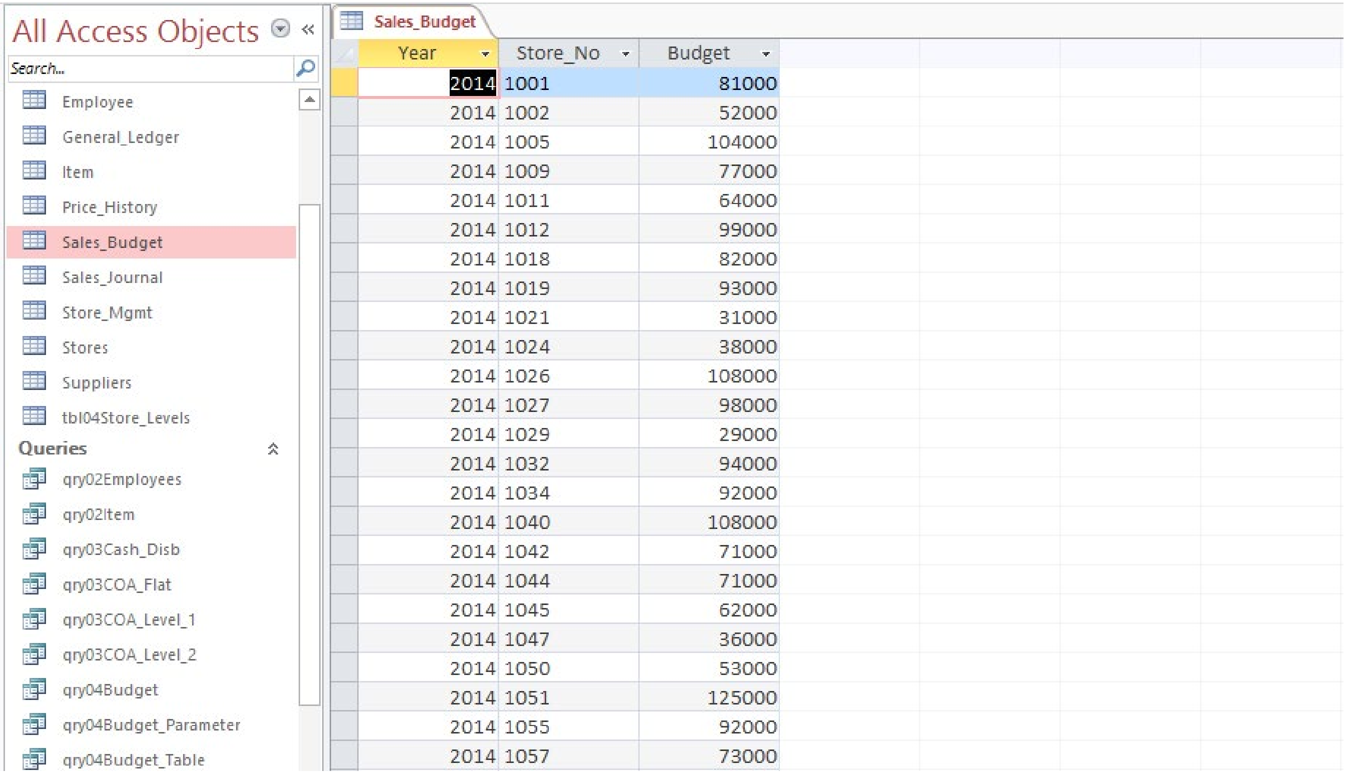
- **Select Query**: The most common type and the only type we've used up to this point.
- **Crosstab Query**: Allows you to pivot data from columns into rows or vice versa.
- **Make Table Query**: Creates a table when executed.
- **Update Query**: Modifies data in a table based on criteria you specify.
- **Append Query**: Inserts additional rows of data into a table.
- **Delete Query**: Deletes rows of data from a table.
- **Union Query**: Combines the results of two or more tables and/or queries in to one dataset (we will do this query in Chapter 14).
- **Pass-Through Query**: Allows users to execute SQL statements directly against tables in an external database, like SQL Server or Oracle.
- **Data Definition Query**: A custom query written with SQL code.

You have already done several exercises using a standard query, called a Select query, in this course already. We will be doing examples of all of these queries (except for Pass-Through and Union) in this chapter.

Crosstab Query

A ***Crosstab Query*** in Access works kind of like a PivotTable in Excel. Since you are a PivotTable expert, this concept should be easy to understand. In the next example, we will take the Sales_Budget table and pivot it to move the Year field from being inside rows to being column headers by using the Crosstab Query Wizard. As we walk through the steps in the Wizard, pay attention to the various options that are available.

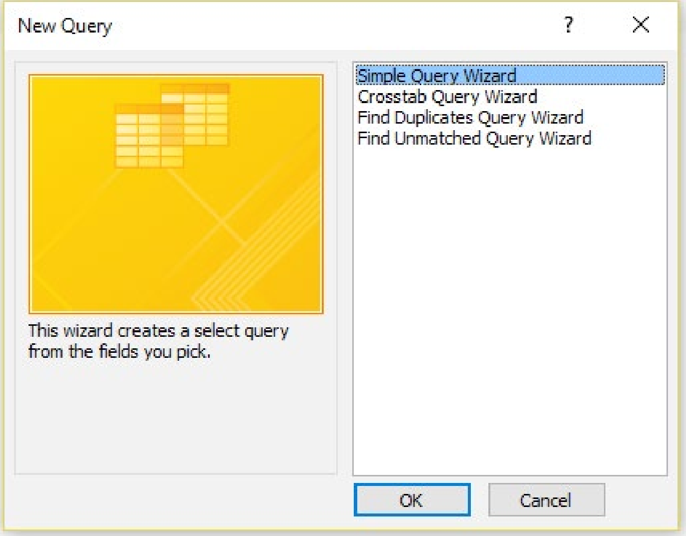
1. *In the* **Nitey-Nite 2016** *database, open the* **Sales_Budget** *table. Notice how the years and store numbers are displayed.*



Year	Store_No	Budget
2014	1001	81000
2014	1002	52000
2014	1005	104000
2014	1009	77000
2014	1011	64000
2014	1012	99000
2014	1018	82000
2014	1019	93000
2014	1021	31000
2014	1024	38000
2014	1026	108000
2014	1027	98000
2014	1029	29000
2014	1032	94000
2014	1034	92000
2014	1040	108000
2014	1042	71000
2014	1044	71000
2014	1045	62000
2014	1047	36000
2014	1050	53000
2014	1051	125000
2014	1055	92000
2014	1057	73000

Figure 5.1

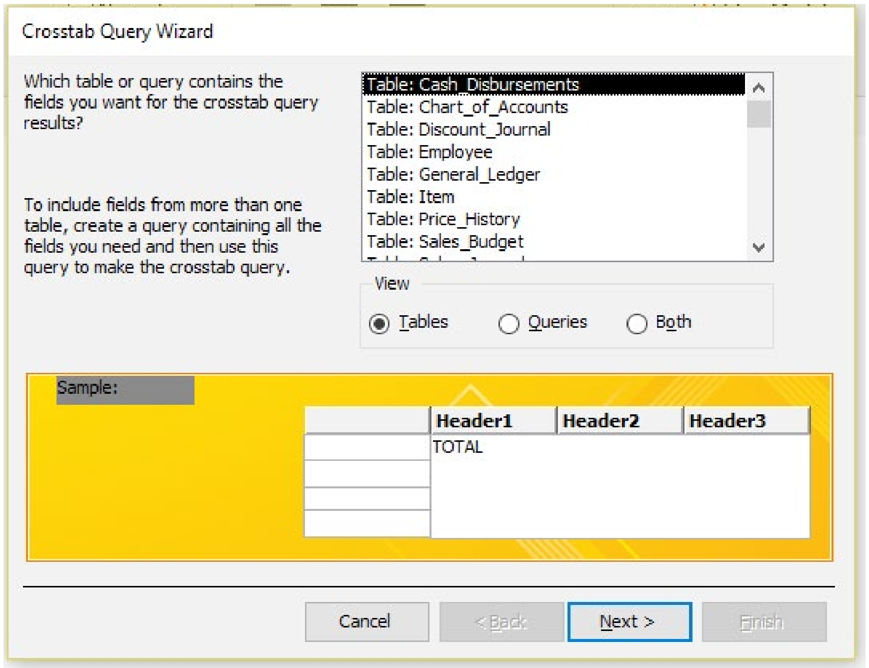
2. *Close the* **Sales_Budget** *table, click on the* **Create** *tab, and then click the* **Query Wizard** *icon.*



This wizard creates a select query from the fields you pick.

Figure 5.2

3. *In the* **New Query** *dialog box, choose* **Crosstab Query Wizard** *and click* **OK**.



Crosstab Query Wizard

Which table or query contains the fields you want for the crosstab query results?

To include fields from more than one table, create a query containing all the fields you need and then use this query to make the crosstab query.

Table: Cash_Disbursements
Table: Chart_of_Accounts
Table: Discount_Journal
Table: Employee
Table: General_Ledger
Table: Item
Table: Price_History
Table: Sales_Budget

View

● Tables ○ Queries ○ Both

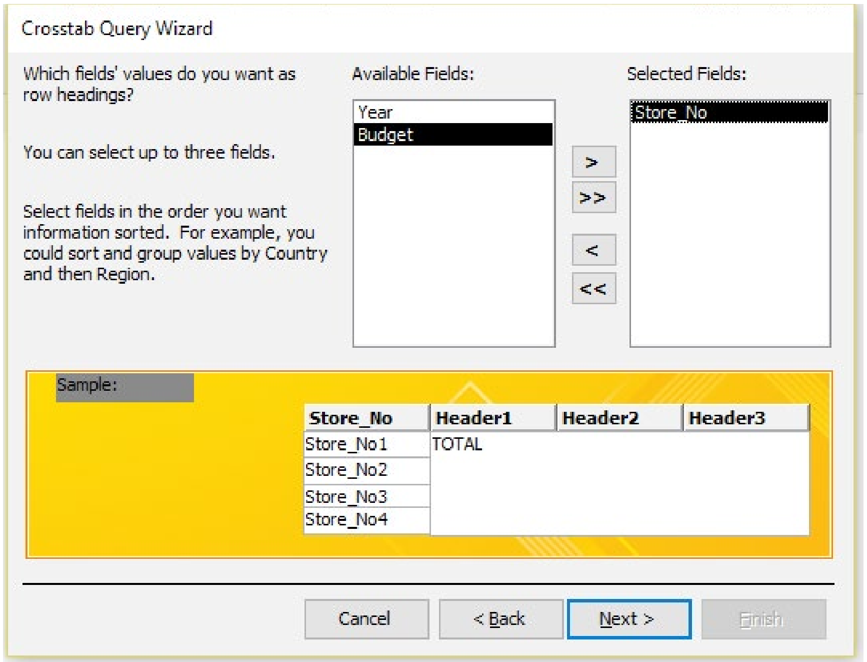
Sample:

	Header1	Header2	Header3
	TOTAL		

Figure 5.3

The first step of the Crosstab Query Wizard asks you to choose the data source of the Crosstab Query, and it lists all of the tables and queries in the open Access database.

4. *Make sure the* **Tables** *or* **Both** *radio button is selected and choose* **Table: Sales_Budget** *and click* **Next***.*
5. *In the next step of the wizard, choose* **Store_No** *as the field you want to use as row headings, click the* > *button (to move* **Store_No** *over to the* **Selected Fields** *area).*



Crosstab Query Wizard

Which fields' values do you want as row headings?

You can select up to three fields.

Select fields in the order you want information sorted. For example, you could sort and group values by Country and then Region.

Available Fields: Year, Budget

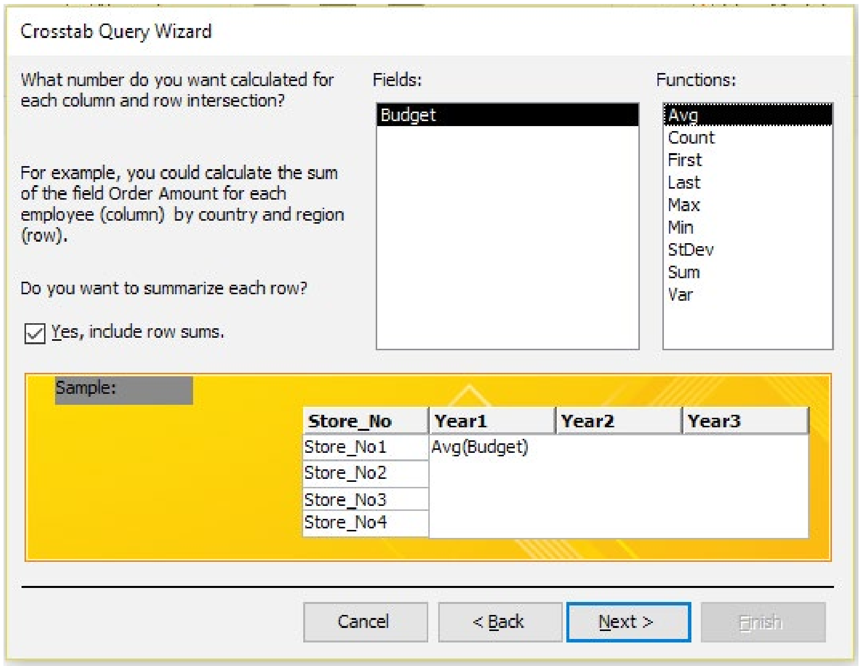
Selected Fields: Store_No

Sample:

Store_No	Header1	Header2	Header3
Store_No1	TOTAL		
Store_No2			
Store_No3			
Store_No4			

Figure 5.4

6. *Click* **Next***.*
7. *Choose* **Year** *as the field you want to use as column headings and click* **Next***.*



Crosstab Query Wizard

What number do you want calculated for each column and row intersection?

For example, you could calculate the sum of the field Order Amount for each employee (column) by country and region (row).

Do you want to summarize each row?

☑ Yes, include row sums.

Fields: Budget

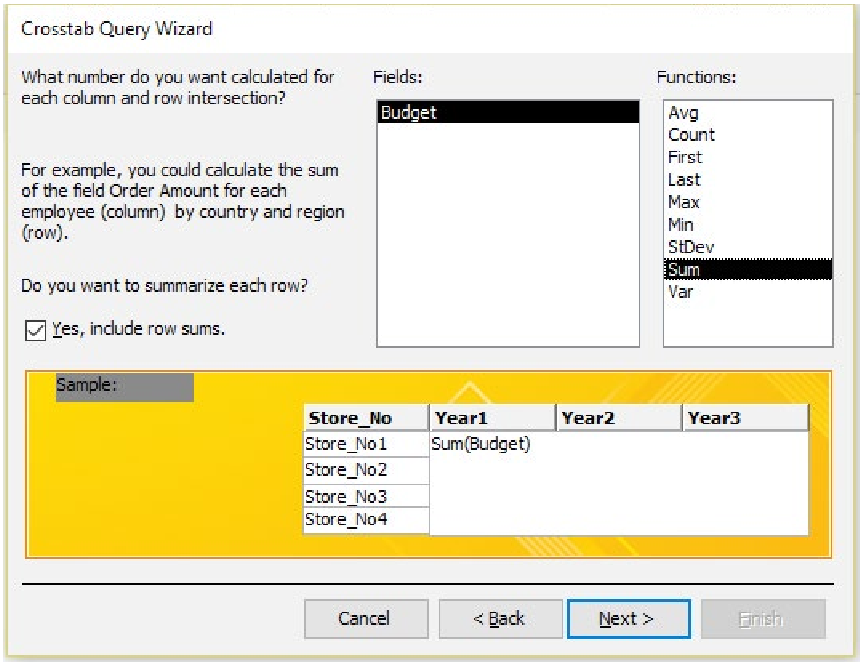
Functions: Avg, Count, First, Last, Max, Min, StDev, Sum, Var

Sample:

Store_No	Year1	Year2	Year3
Store_No1	Avg(Budget)		
Store_No2			
Store_No3			
Store_No4			

Figure 5.5

8. *Under* **Functions:** *choose the* **Sum** *function to be used as the calculation.*



Crosstab Query Wizard

What number do you want calculated for each column and row intersection?

For example, you could calculate the sum of the field Order Amount for each employee (column) by country and region (row).

Do you want to summarize each row?

☑ Yes, include row sums.

Fields: Budget

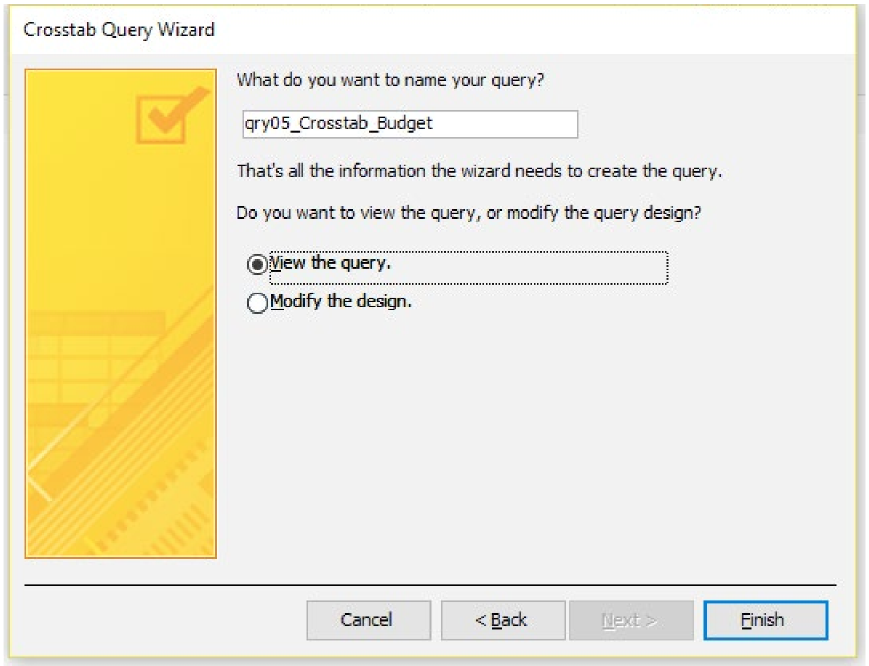
Functions: Avg, Count, First, Last, Max, Min, StDev, Sum, Var

Sample:

Store_No	Year1	Year2	Year3
Store_No1	Sum(Budget)		
Store_No2			
Store_No3			
Store_No4			

Figure 5.6

9. *Click* **Next***.*
10. *Change the default query name to* **qry05Crosstab_Budget** *and select the* **View the query** *radio button.*



Crosstab Query Wizard

What do you want to name your query?

qry05_Crosstab_Budget

That's all the information the wizard needs to create the query.

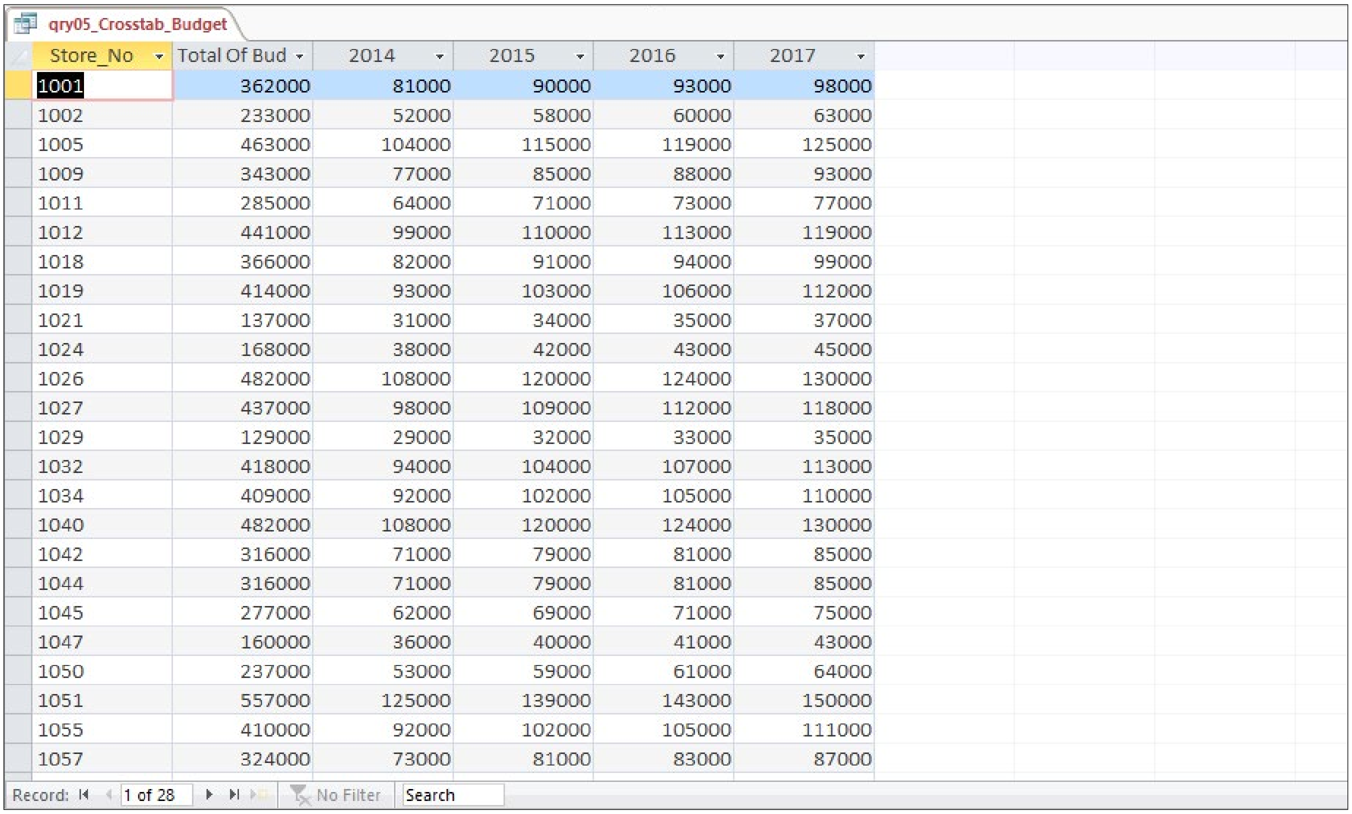
Do you want to view the query, or modify the query design?

● View the query.

○ Modify the design.

Figure 5.7

11. *Click* **Finish**.



qry05_Crosstab_Budget

Store_No	Total Of Bud	2014	2015	2016	2017
1001	362000	81000	90000	93000	98000
1002	233000	52000	58000	60000	63000
1005	463000	104000	115000	119000	125000
1009	343000	77000	85000	88000	93000
1011	285000	64000	71000	73000	77000
1012	441000	99000	110000	113000	119000
1018	366000	82000	91000	94000	99000
1019	414000	93000	103000	106000	112000
1021	137000	31000	34000	35000	37000
1024	168000	38000	42000	43000	45000
1026	482000	108000	120000	124000	130000
1027	437000	98000	109000	112000	118000
1029	129000	29000	32000	33000	35000
1032	418000	94000	104000	107000	113000
1034	409000	92000	102000	105000	110000
1040	482000	108000	120000	124000	130000
1042	316000	71000	79000	81000	85000
1044	316000	71000	79000	81000	85000
1045	277000	62000	69000	71000	75000
1047	160000	36000	40000	41000	43000
1050	237000	53000	59000	61000	64000
1051	557000	125000	139000	143000	150000
1055	410000	92000	102000	105000	111000
1057	324000	73000	81000	83000	87000

Figure 5.8

The years have moved from the values in rows (records) to being column (field) headers. Although this is a simple example, I hope you can see the power of a crosstab query. Just remember that it works very similar to a PivotTable in Excel.

12. **Save** *and close* **qry05Crosstab_Budget**.

Action Queries

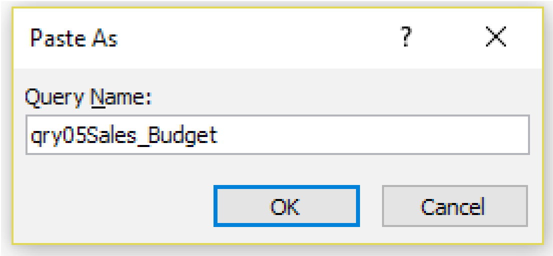
The other types of queries (Make Table, Update, Append, and Delete queries) are called ***Action Queries*** because they perform actions on the data. With action queries, you can do things like create and delete tables, modify the data in a table, and delete records from tables. These action queries allow you to manipulate data and perform analyses that may not be possible otherwise. In this chapter, we will discuss and work examples of each of these types of queries.

Make Table Query

The first type of query I want to discuss is a ***Make Table Query***. As its name implies, this type of query makes or creates a table. As you've seen before, sometimes you have to join numerous tables or queries to get the desired record set. Typically, the more tables and queries you use in a query, the more time it takes to run. My general rule of thumb is that if it takes more than ten seconds or so to run a query, it's too long — users don't like to sit around waiting much longer than that. Ten seconds to run doesn't seem like a lot of time, unless YOU'RE the one running the query — then it seems to take forever. This is particularly true when the query is interactive, or when you can "drill down" on the data. If there are seven levels of data (which is very common), a query that takes ten seconds to run can take more than a minute to get down to the lowest level. If that doesn't seem like a long time, try holding your breath for that long.

Using a Make Table Query can cut the run time for a query or report from several minutes to almost instantaneous. When I create reports, I try to base the reports on tables or queries that are as small and efficient as possible. This way, I get the quickest response time. Take, for example, the query we created in the previous chapter, qry04Sales_Budget. When I run that query on my old computer, it takes about seventeen seconds. Your time may be more or less, depending on the speed at which your computer processes information. I personally think seventeen seconds is too long to wait for numbers, so I could create a Make Table Query and base my analysis on the new table instead of the query. One problem with using a new table created by a Make Table query is that if something changes in the data and the Make Table Query does not run for some reason, you could possibly get an incorrect result. However, you can build procedures so that the probability of that happening is minimal. Let's create a Make Table Query. We'll use the query we built in the last chapter as a starting point.

1. **Copy** *and* **paste** *the* **qry04Sales_Budget** *query.*
2. *In the* **Paste As** *dialog box, type* ***qry05Sales_Budget*** *and click* **OK**.



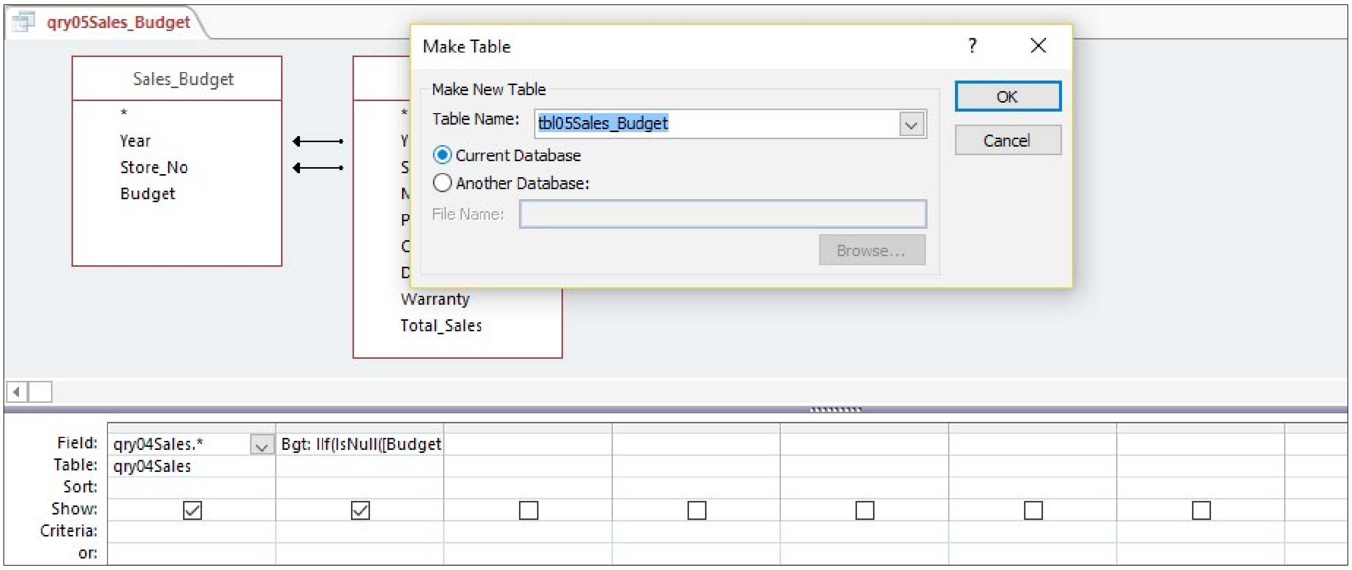
Query Name: qry05Sales_Budget

Figure 5.9

3. *Go to the* **Design View** *of* **qry05Sales_Budget**.
4. *Click on the* **Make Table** *icon on the* **Query Tools Design** *tab.*

The Make Table dialog box appears.

5. *In the* **Make Table** *dialog box, type* ***tbl05Sales_Budget***.



Make Table

Make New Table

Table Name: tbl05Sales_Budget

● Current Database

○ Another Database:

File Name:

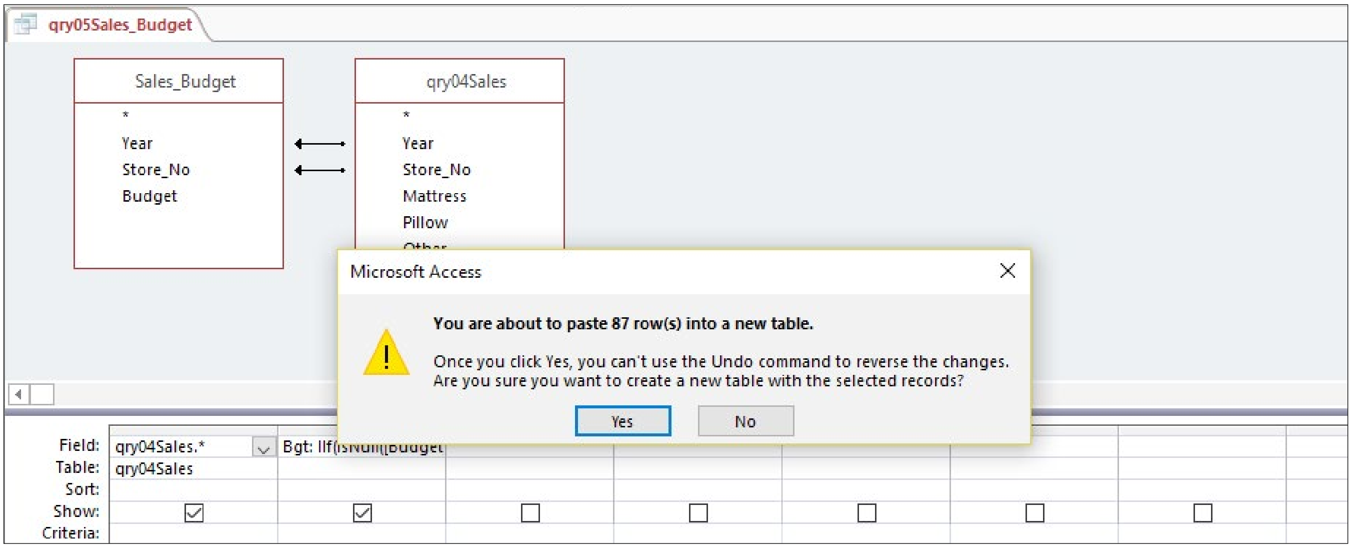
Field:	qry04Sales.*	Bgt: IIf(IsNull([Budget					
Table:	qry04Sales						
Sort:							
Show:	☑	☑	□	□	□	□	□
Criteria:							
or:							

Figure 5.10

6. *Click* **OK**.

The name tbl05Sales_Budget is the name of the table you will create. Whenever I create a table, I like to begin its name with "tbl". This lets me know that the table is one that I created. I also like to name the table the same as the Make Table query that created it. Now you are ready to create the table.

7. *Click the* **Run** *icon image.*



Microsoft Access

You are about to paste 87 row(s) into a new table.

Once you click Yes, you can't use the Undo command to reverse the changes.
Are you sure you want to create a new table with the selected records?

Field:	qry04Sales.*	Bgt: IIf(IsNull([Budget					
Table:	qry04Sales						
Sort:							
Show:	☑	☑	□	□	□	□	□
Criteria:							

Figure 5.11

Access warns you that you are about to create a new table that will contain 87 rows of data. This is what you want to do, so click Yes.

8. *Click* **Yes***.*

You return to the Design View of qry05Sales_Budget.

> ***Note****: To run a* **Make Table** *query or any* **Action Query***, you must click on the* **Run** *icon, not the* **View** *icon. The* **View** *icon shows the results of the query in* **Datasheet View***, but it does not execute an action query — you must click on the* **Run** *icon to do that.*

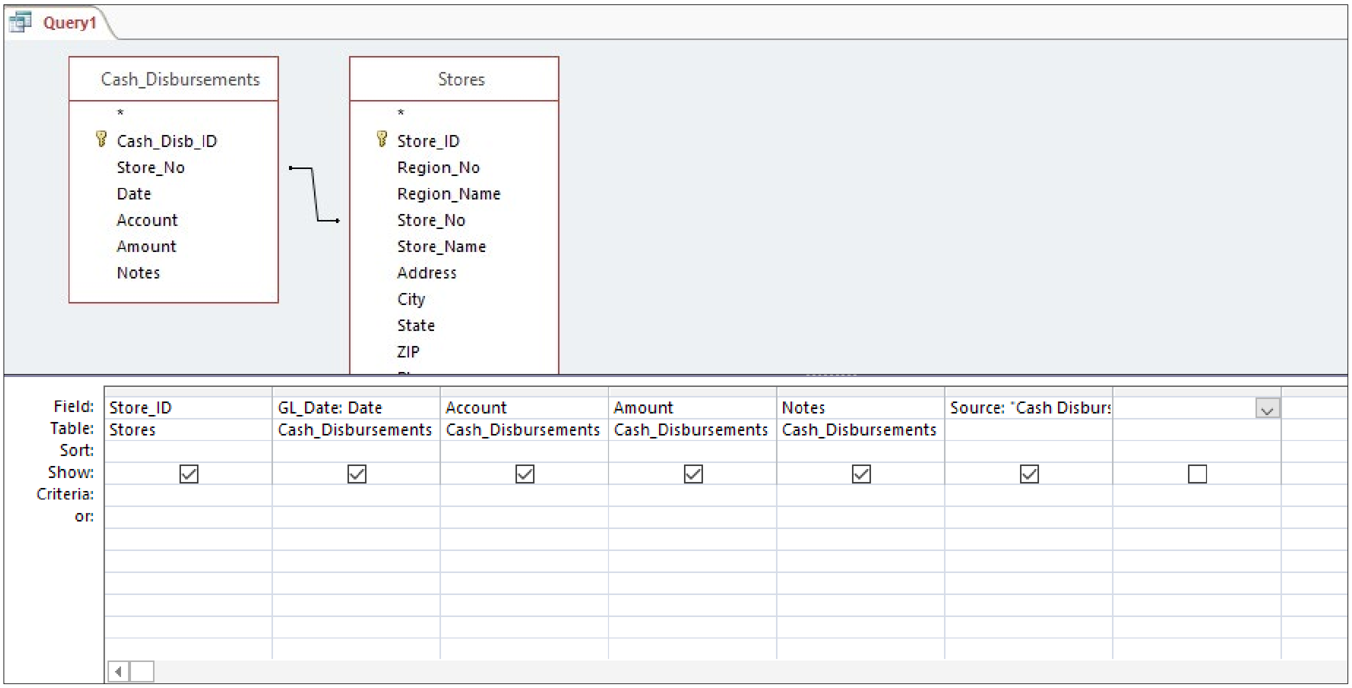
9. **Save** *and close* **qry05Sales_Budget***.*

You now see that there is a new table called tbl05Sales_Budget in the Navigation Pane. Open the table and you will see that it opens immediately instead of taking several seconds to run the query.

I have found Make Table Queries to be very useful. If you run reports daily, weekly, or monthly, you will find that creating tables instead of using queries that take a long time to run can save a lot of time. But be careful! Make sure you have the appropriate procedures in place to ensure that all of the tables are kept up-to-date.

In the next few examples, we are going to re-create the journal entries booked to the General Ledger based on the Cash_Disbursements table and the Discount_Journal table. The tables in the Nitey_Nite_2016 database make up the base data behind some of the entries to the General Ledger, and we will perform an audit of these tables to make sure the amounts in the tables are recorded correctly in the General Ledger. To do this audit, we will first create a table with a Make Table query, then append data to that table, next we'll update data in the table, and finally delete records that we don't need from the table. All of this is done through action queries.

1. *Create a query that shows all of the records in the* **Cash_Disbursements** *table with the following fields:*
   - **Store_ID***: (you will have to join to the* **Stores** *table to get the* **Store_ID***)*
   - **GL_Date***: (from the* **Date** *field in* **Cash_Disbursements***)*
   - **Account***: (from* **Cash_Disbursements***)*
   - **Amount***: (from* **Cash_Disbursements***)*
   - **Notes***: (from* **Cash_Disbursements***)*
   - **Source***: (use the reference* ***"Cash_Disbursements"****)*



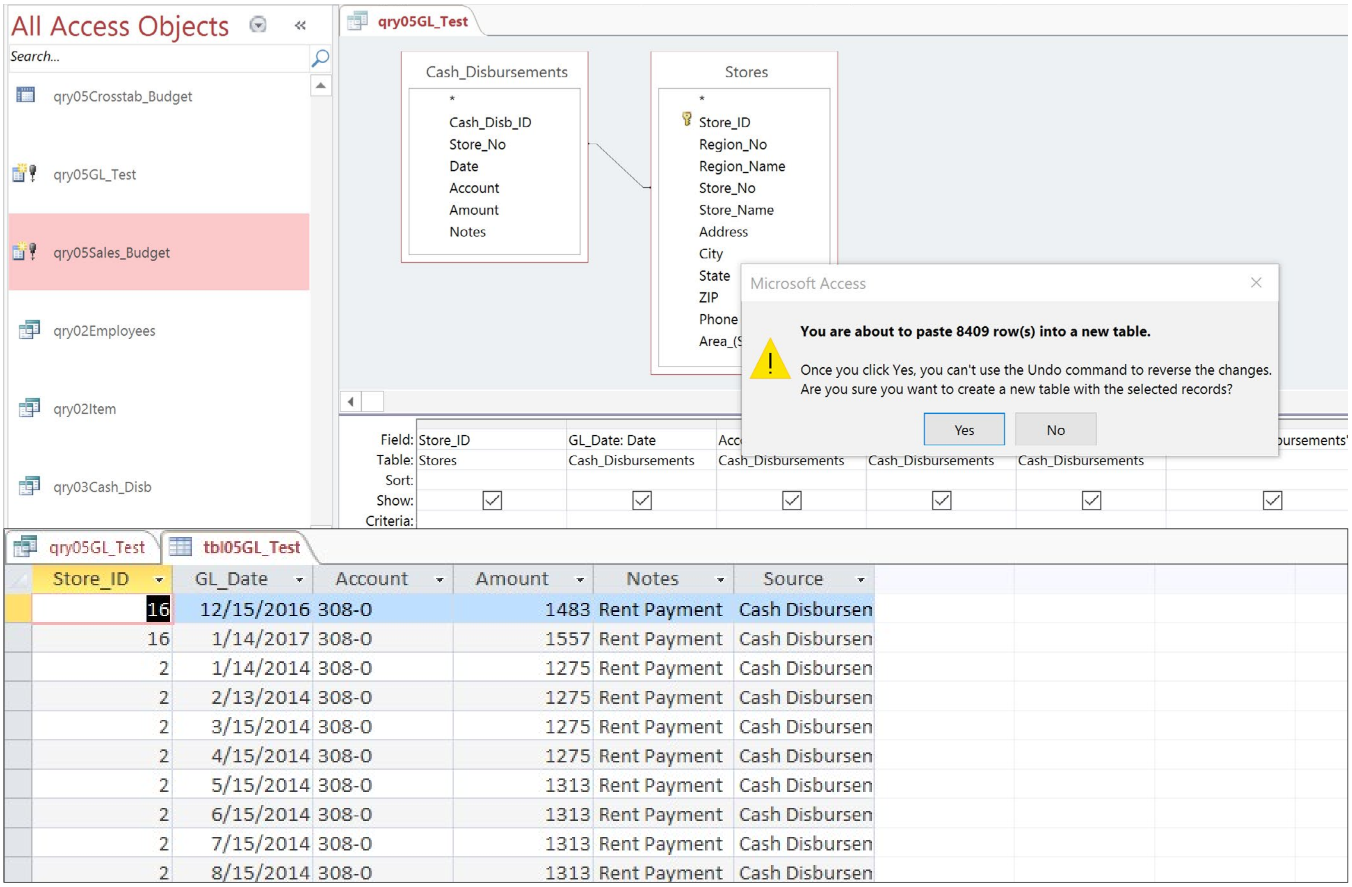
Field:	Store_ID	GL_Date: Date	Account	Amount	Notes	Source: "Cash Disburs	
Table:	Stores	Cash_Disbursements	Cash_Disbursements	Cash_Disbursements	Cash_Disbursements		
Sort:							
Show:	☑	☑	☑	☑	☑	☑	□
Criteria:							
or:							

Figure 5.12

2. *Make the query a* **Make Table Query** *and call the new table* ***tbl05GL_Test****.*
3. *Name the query* ***qry05GL_Test*** *and save it.*
4. **Run** *the query.*

There should be 8,409 records in the new table produced by your query, as there are also 8,409 records in the Cash_Disbursements table.

5. *Click* **Yes** *to create the table.*
6. *Check* **tbl05GL_Test** *to make sure the table was created correctly.*



You are about to paste 8409 row(s) into a new table.

Once you click Yes, you can't use the Undo command to reverse the changes. Are you sure you want to create a new table with the selected records?

Field:	Store_ID	GL_Date: Date	Acc			ursements'
Table:	Stores	Cash_Disbursements	Cash_Disbursements	Cash_Disbursements	Cash_Disbursements	
Sort:						
Show:	☑	☑	☑	☑	☑	☑
Criteria:						

Store_ID	GL_Date	Account	Amount	Notes	Source
16	12/15/2016	308-0	1483	Rent Payment	Cash Disbursen
16	1/14/2017	308-0	1557	Rent Payment	Cash Disbursen
2	1/14/2014	308-0	1275	Rent Payment	Cash Disbursen
2	2/13/2014	308-0	1275	Rent Payment	Cash Disbursen
2	3/15/2014	308-0	1275	Rent Payment	Cash Disbursen
2	4/15/2014	308-0	1275	Rent Payment	Cash Disbursen
2	5/15/2014	308-0	1313	Rent Payment	Cash Disbursen
2	6/15/2014	308-0	1313	Rent Payment	Cash Disbursen
2	7/15/2014	308-0	1313	Rent Payment	Cash Disbursen
2	8/15/2014	308-0	1313	Rent Payment	Cash Disbursen

Figure 5.13

7. **Save** *and close the* **qry05GL_Test** *query.*

Now that you have the beginnings of a table, we need to insert more data into the table from different sources. We can do this with an Append Query, as explained in the next section.

> ***Review Questions****: It is now time to complete the hands-on Review Questions. Log on to www.ExcelCEO.com with your Email and Password, click on the* **Access 2016 and SQL Review Questions, Chapter 5, Section 1 of 2** *option and complete the review questions.*

Append Query

As the name implies, an ***Append Query*** appends, inserts, or adds rows of data to an existing table. To perform an Append Query, the added data must be in the same format as the table in which you're inserting the data. For example, you shouldn't append a store number into a store ID field. A store number is formatted as text, and a store ID is formatted as a number. Often the data you want to append is resident in another data source, like Excel or in a text file, and you can simply copy the data and paste it into the Access table. You still have to make sure the data being copied is in the same format as the

Access table you are copying the data into. For the purposes of this exercise, however, we'll stick with using Access tables.

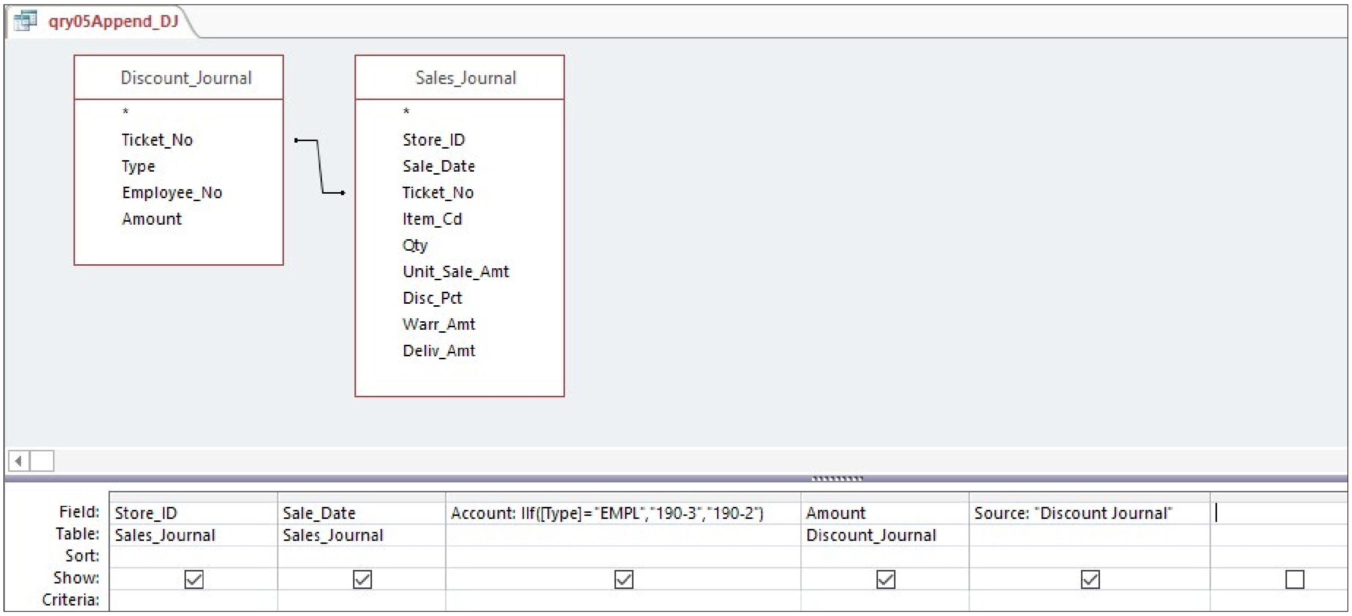
The Discount_Journal table at Nitey-Nite records the atypical discounts given in the Nitey-Nite stores. There are two types of discounts other than the normal promotional discount (which is already recorded in the Sales_Journal): Employee Discounts and Charity Discounts. Stating the obvious, Employee discounts are given to employees of Nitey-Nite, and Charity discounts are discounts that are given, with the store manager's permission, to purchasers from charitable organizations (churches, not-for-profit hospitals, etc.). An Employee Discount is booked to account 190-3 and a Charity Discount is booked to account 190-2.

The first field we need to query for is the Store_ID, which is the first field in the tbl05GL_Test table. If you open the Discount_Journal, you will see there is no store id or a store number field. All we have in that table are the Ticket_No, Type (Employee or Charity), Employee_No, and Amount fields. To get the Store_ID, we need to create a join to the Sales_Journal on the field Ticket_No. We can also get the GL_Date from the Sales_Journal. Let's do that.

1. *Create a new query based on the* **Discount_Journal** *and* **Sales_Journal** *tables.*
2. *Create a join between the* **Discount_Journal** *and* **Sales_Journal** *using the* **Ticket_No** *field.*
3. *Bring the* **Store_ID** *and* **Sale_Date** *fields from the* **Sales_Journal** *into the* **Design Grid**.
4. *Create an alias field called* ***Account****, and write an* ***IIF()*** *statement that indicates that if the* ***Type*** *is* **equal** *to* ***EMPL****, then use account* ***190-3****. If not, use account* ***190-2****.*

   Note*: Those are the only types of discounts in that table.*

5. *Bring in the* **Amount** *from the* **Discount_Journal** *table.*
6. *Define the* ***Source*** *as* ***Discount Journal****.*
7. *Save the query as* ***qry05Append_DJ****.*

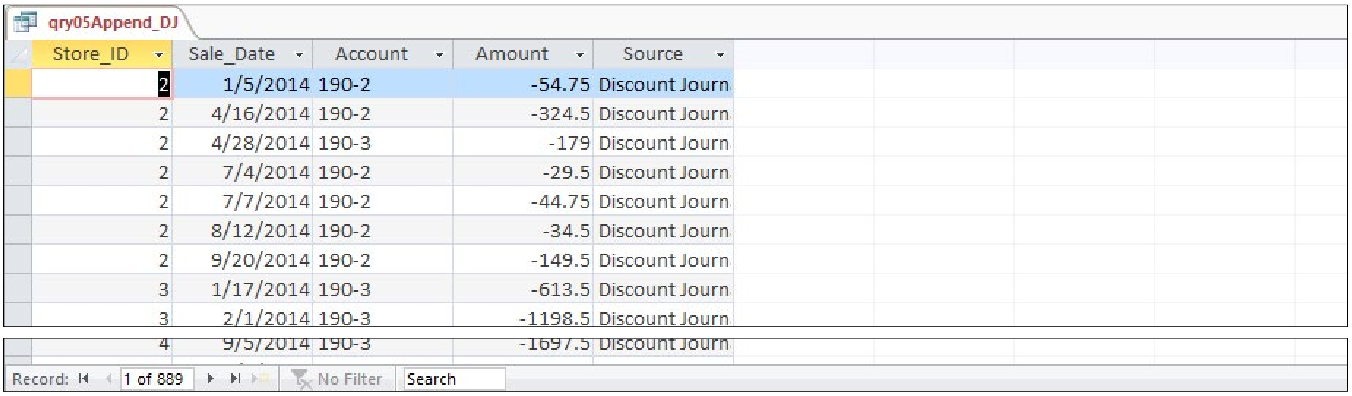


qry05Append_DJ

Field:	Store_ID	Sale_Date	Account: IIf([Type]="EMPL","190-3","190-2")	Amount	Source: "Discount Journal"	
Table:	Sales_Journal	Sales_Journal		Discount_Journal		
Sort:						
Show:	☑	☑	☑	☑	☑	☐
Criteria:						

Figure 5.14

8. **Run** *the query.*



qry05Append_DJ

Store_ID	Sale_Date	Account	Amount	Source
2	1/5/2014	190-2	-54.75	Discount Journ
2	4/16/2014	190-2	-324.5	Discount Journ
2	4/28/2014	190-3	-179	Discount Journ
2	7/4/2014	190-2	-29.5	Discount Journ
2	7/7/2014	190-2	-44.75	Discount Journ
2	8/12/2014	190-2	-34.5	Discount Journ
2	9/20/2014	190-2	-149.5	Discount Journ
3	1/17/2014	190-3	-613.5	Discount Journ
3	2/1/2014	190-3	-1198.5	Discount Journ
4	9/5/2014	190-3	-1697.5	Discount Journ

Record: 1 of 889 No Filter Search

Figure 5.15

Auditing with One-Way Joins

Things seem to be working, but check this out. The total number of records returned by the qry05Append_DJ query is 889. If you open the Discount_Journal table, you'll see that there are 893 records in the table. We're missing four records. Why? That's easy to find out. Since you are an expert in manipulating tables in Access, finding the answer should be no problem. We'll just use the one-way join trick you learned in Chapter 4.

1. **Create** *a new query using the* **Discount_Journal** *and* **Sales_Journal** *tables.*
2. *Do a one-way join to include all records from the* **Discount_Journal** *table and the corresponding records from the* **Sales_Journal** *table on the* **Ticket_No** *field.*

3. *Bring in the* **Ticket_No** *fields from both tables into the* **Design Grid**.
4. *In the* **Criteria** *section for the* **Sales_Journal.Ticket_No** *field, type* ***Is Null***

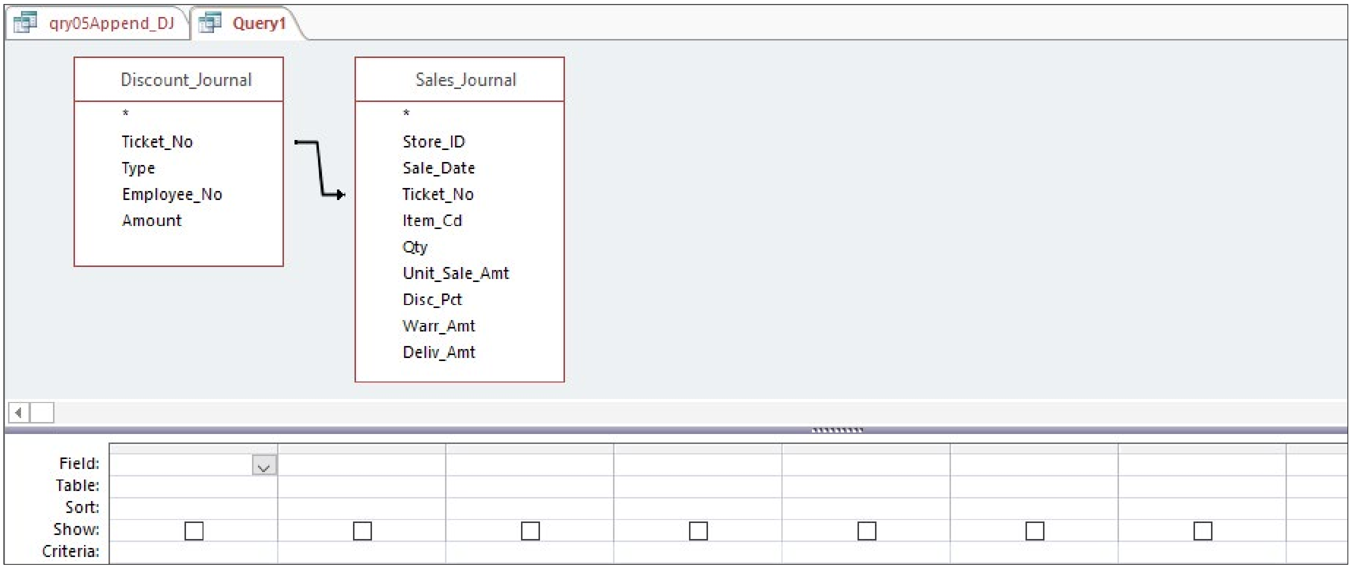
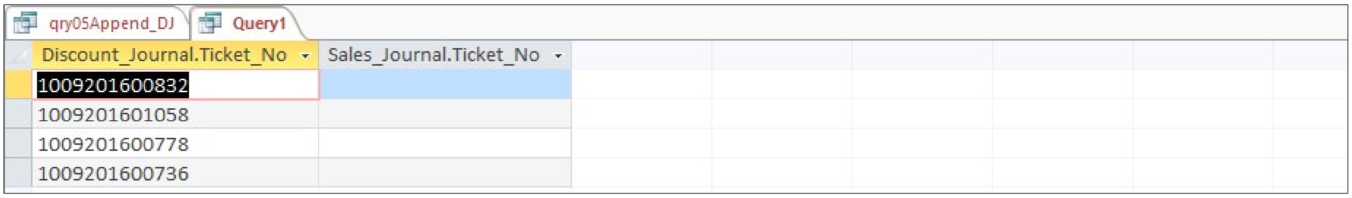


Figure 5.16

5. **Run** *the query and expand the fields to see the full titles.*



Discount_Journal.Ticket_No	Sales_Journal.Ticket_No
1009201600832	
1009201601058	
1009201600778	
1009201600736	

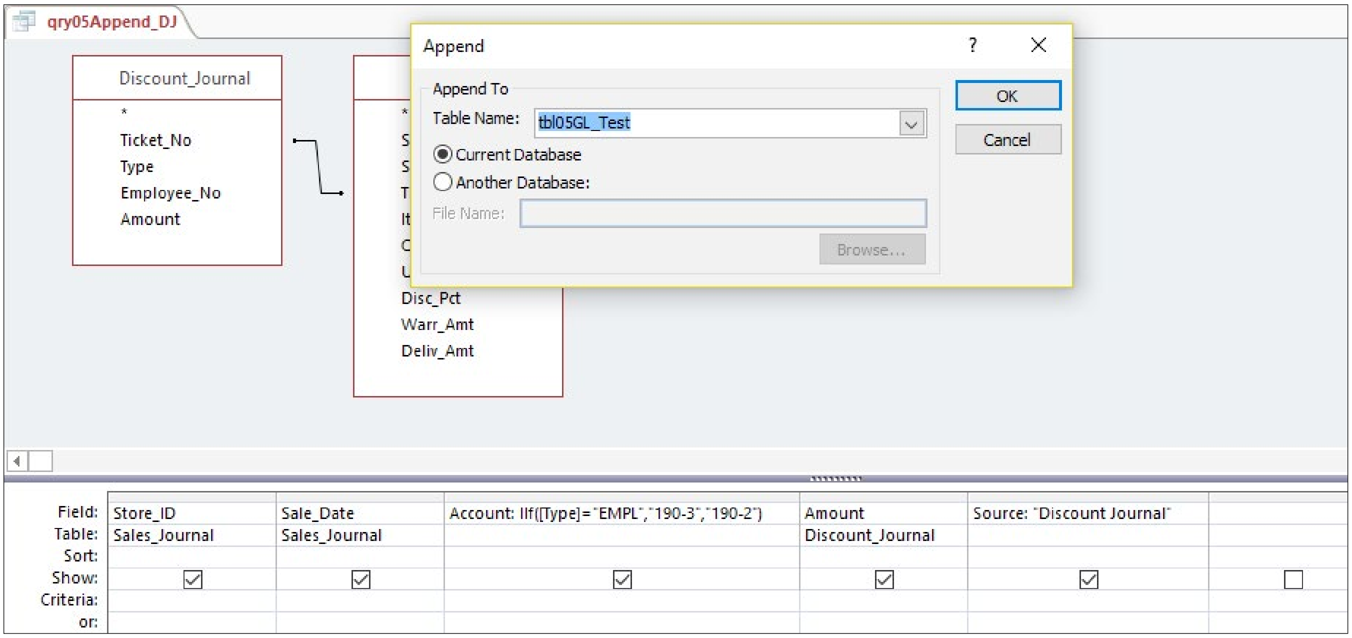
Figure 5.17

There's your answer. These four ticket numbers do not exist in the Sales_Journal table. There is probably some type of miscoding on the part of the person or system who input the numbers into the Discount_ Journal table. At this point, your manager has determined that this error is an immaterial reconciling item, so let's move on. In reality, you would probably manually input these records into the Sales_Journal.

6. *Close* **Query1** *without saving it.*

Your qry05Append_DJ query is now ready to upload or append. Remember, we created the table tbl05GL_Test, and now we want to append the data from your new query to that table. Basically, we're recreating a portion of the General Ledger.

7. *In the* **Design View** *of* **qry05Append_DJ**, *click on the* **Append** Append *query icon in the* **Query Tools Design** *tab.*
8. *In the* **Append** *dialog box, click on the drop-down menu and choose* **tbl05GL_Test**.



qry05Append_DJ

Discount_Journal
- *
- Ticket_No
- Type
- Employee_No
- Amount

Disc_Pct
Warr_Amt
Deliv_Amt

Append

Append To

Table Name: tbl05GL_Test

● Current Database

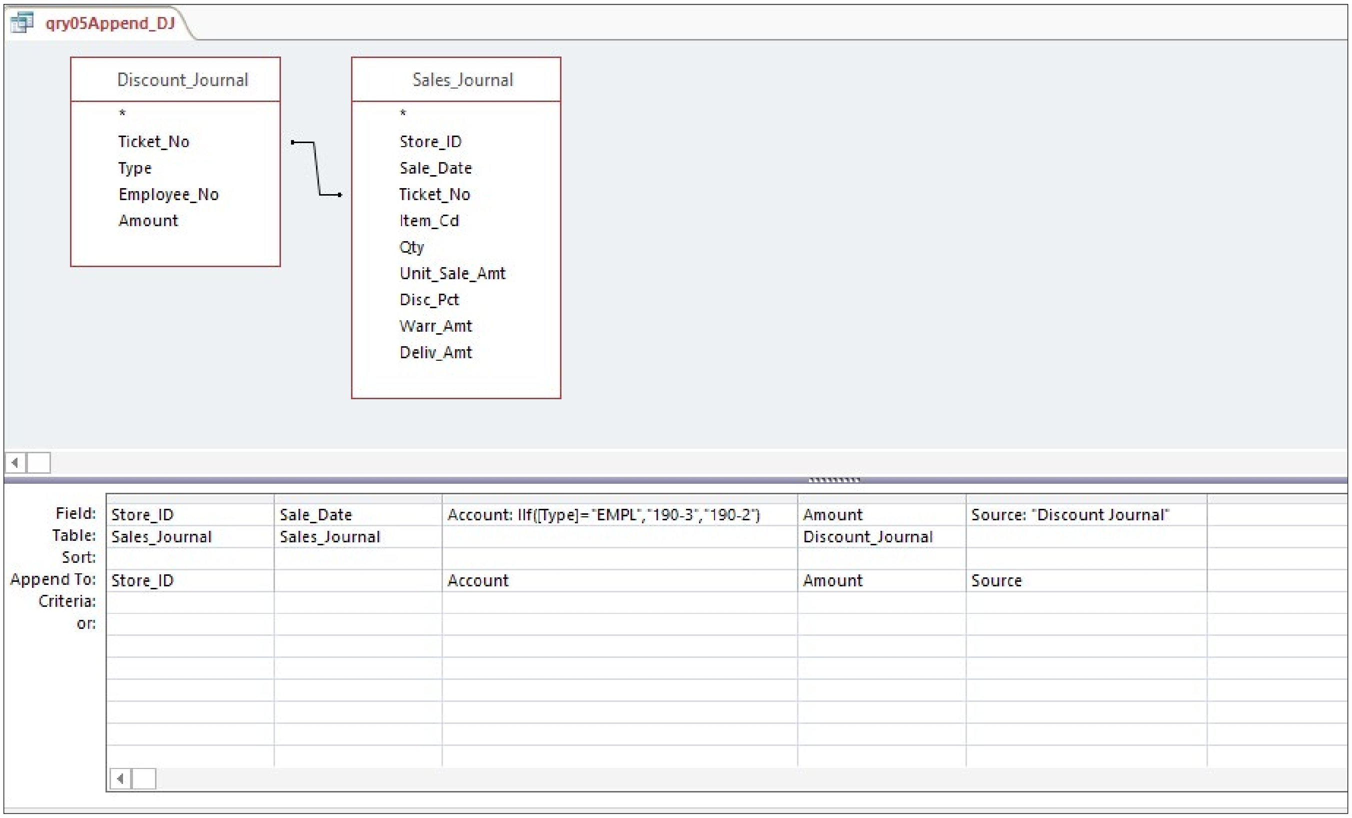
○ Another Database:

File Name:

Field:	Store_ID	Sale_Date	Account: IIf([Type]="EMPL","190-3","190-2")	Amount	Source: "Discount Journal"	
Table:	Sales_Journal	Sales_Journal		Discount_Journal		
Sort:						
Show:	☑	☑	☑	☑	☑	□
Criteria:						
or:						

Figure 5.18

9. *Click* **OK**.



qry05Append_DJ

Discount_Journal
- *
- Ticket_No
- Type
- Employee_No
- Amount

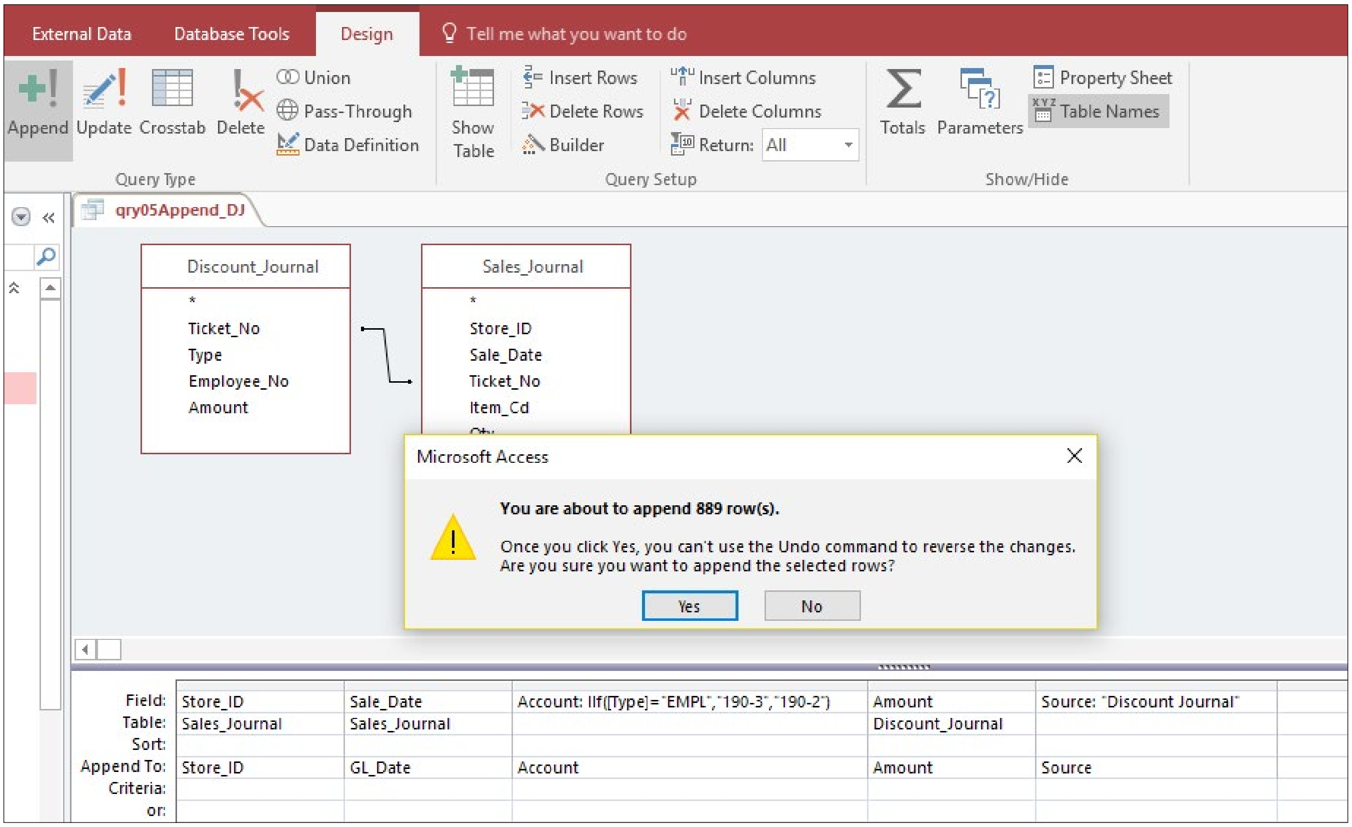
Sales_Journal
- *
- Store_ID
- Sale_Date
- Ticket_No
- Item_Cd
- Qty
- Unit_Sale_Amt
- Disc_Pct
- Warr_Amt
- Deliv_Amt

Field:	Store_ID	Sale_Date	Account: IIf([Type]="EMPL","190-3","190-2")	Amount	Source: "Discount Journal"
Table:	Sales_Journal	Sales_Journal		Discount_Journal	
Sort:					
Append To:	Store_ID		Account	Amount	Source
Criteria:					
or:					

Figure 5.19

A new row, Append To:, appears beneath the Sort row. This row tells you which field to append the data to in tbl05GL_Test. Notice that there is no Append To field under the Sale_Date field. That is because tbl05GL_Test does not have a Sale_Date field. We want to append the Sale_Date data to the GL_Date field in tbl05GL_Test.

10. *Click in the* **Append To:** *box under* **Sale_Date**, *and choose* **GL_Date** *from the drop-down menu provided.*
11. **Run** *the query by clicking the Run icon.*



You are about to append 889 row(s).

Once you click Yes, you can't use the Undo command to reverse the changes.
Are you sure you want to append the selected rows?

Field:	Store_ID	Sale_Date	Account: IIf([Type]="EMPL","190-3","190-2")	Amount	Source: "Discount Journal"
Table:	Sales_Journal	Sales_Journal		Discount_Journal	
Sort:					
Append To:	Store_ID	GL_Date	Account	Amount	Source
Criteria:					
or:					

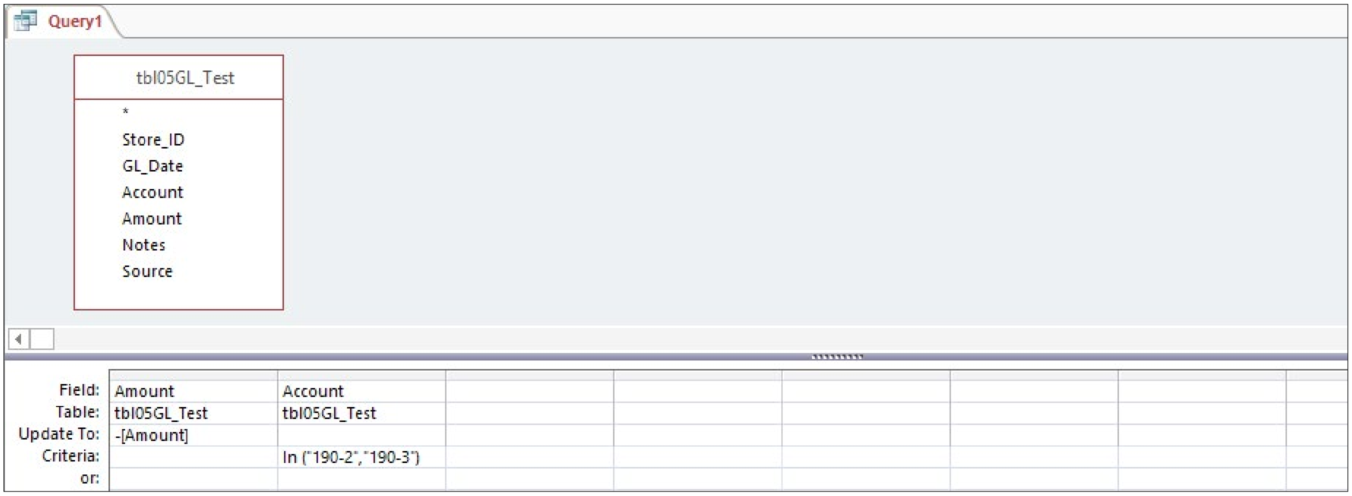
Figure 5.20

12. *Click* **Yes**.
13. **Save** *and close* **qry05Append_DJ**.

Update Query

An ***Update Query*** will change the data in a table based on criteria you specify. In case you didn't notice, when we created tbl05GL_Test table from the Cash_Disbursements table, the amounts were positive amounts, or debit amounts. When we appended the data from the Discount_Journal, the data was in negative amounts, or credits. If you don't know the difference between a debit and a credit, just trust me that discounts to revenue are posted as debits to (or a reduction from) revenue. As such, we accidentally uploaded the data with the wrong sign. We now need to reverse the sign for all of the entries we posted to the discount accounts (190-2 and 190-3). This is easily accomplished with an Update Query.

1. **Create** *a new query based on* **tbl05GL_Test**.
2. *Bring the* **Amount** *and* **Account** *fields into the query* **Design Grid**.
3. *Choose* **Update Query** *on the* **Query Tools Design** *tab.*
4. *On the* **Criteria** *line of the* **Account** *field, type* ***In("190-2","190-3")***. *(This means that we want to bring in only accounts 190-2 and 190-3.)*
5. *In the* **Update To:** *section of the* **Amount** *field, type* ***-[Amount]***

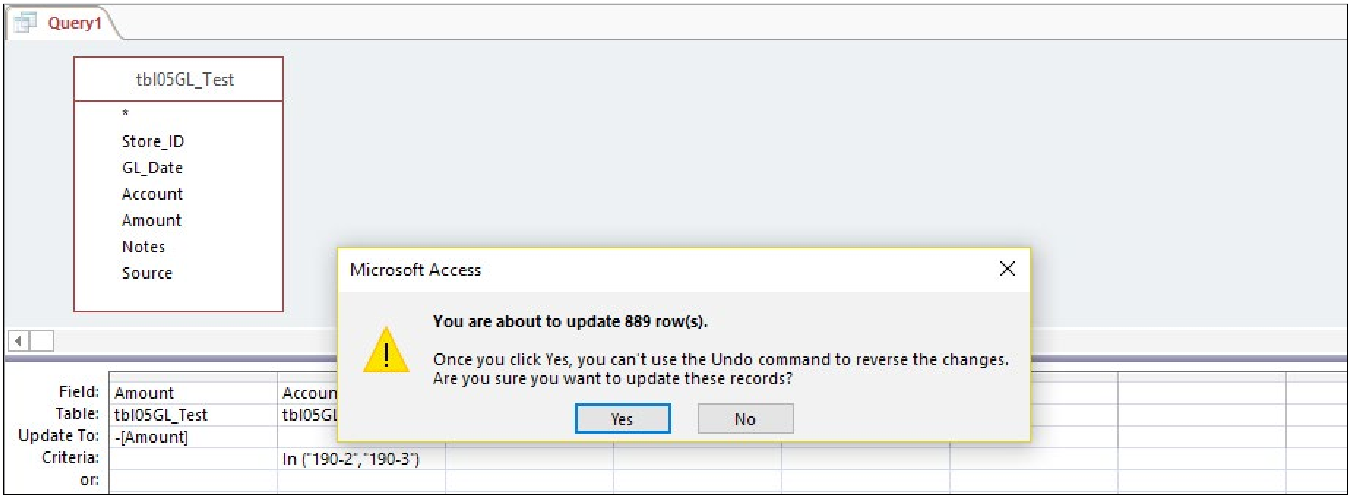


Field:	Amount	Account
Table:	tbl05GL_Test	tbl05GL_Test
Update To:	-[Amount]	
Criteria:		In ("190-2","190-3")
or:		

Figure 5.21

Once you run the query, it will take the numbers in the amount field where the account is either 190-2 or 190-3 and reverse the sign. You can also take the amount and multiply it by it -1. Note that you cannot view the updated values by clicking the View icon. The values will only update when the query is run by clicking the Run icon.

6. **Run** *the query.*



Field:	Amount	Accoun
Table:	tbl05GL_Test	tbl05GL
Update To:	-[Amount]	
Criteria:		In ("190-2","190-3")
or:		

You are about to update 889 row(s).

Once you click Yes, you can't use the Undo command to reverse the changes.
Are you sure you want to update these records?

Figure 5.22

7. *Click* **Yes**.

The values of accounts 190-2 and 190-3 have now been reversed.

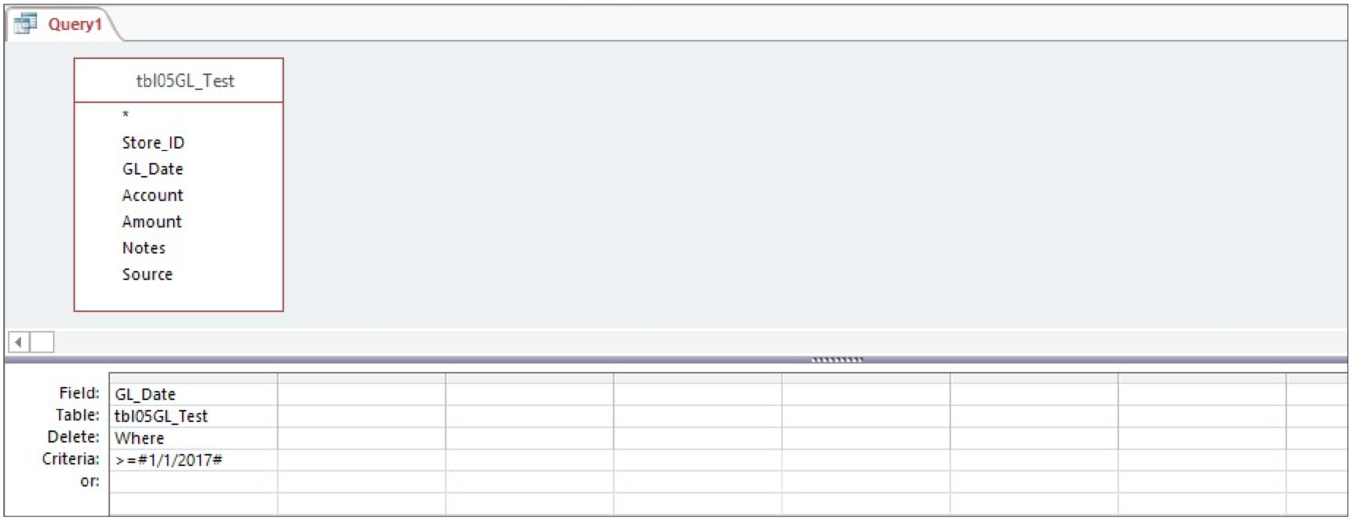
8. **Save** *the query as* ***qry05Update_GL*** *and close it.*
9. *To make sure you don't repeat the same mistake again, edit the* **Amount** *field in the* **qry05Append_DJ** *to* ***Amt: -[Amount]***
10. **Save** *and close* **qry05Append_DJ**.

Delete Query

The next action query we'll review is the ***Delete Query***. This query type deletes specified records within a table or all of the records in a table, but does not delete the table itself. Just as with the other action queries, you need to specify which records you want to affect in the Criteria section. Be careful with this one, because if you make no specification, it will delete ALL records.

In our table, we don't need any GL data in the year 2017, so we'll write a Delete Query which will delete all records with GL dates on or after 1/1/2017.

1. **Create** *a new query based on* **tbl05GL_Test**.
2. *Bring* **GL_Date** *into the* **Design Grid**.
3. *Choose* **Delete** *query* ✕ *from the* **Query Tools Design** *tab.*
4. *In the* **Criteria** *section, type* ***>=1/1/2017*** *(Access will change the code to* **>=#1/1/2017#***)*



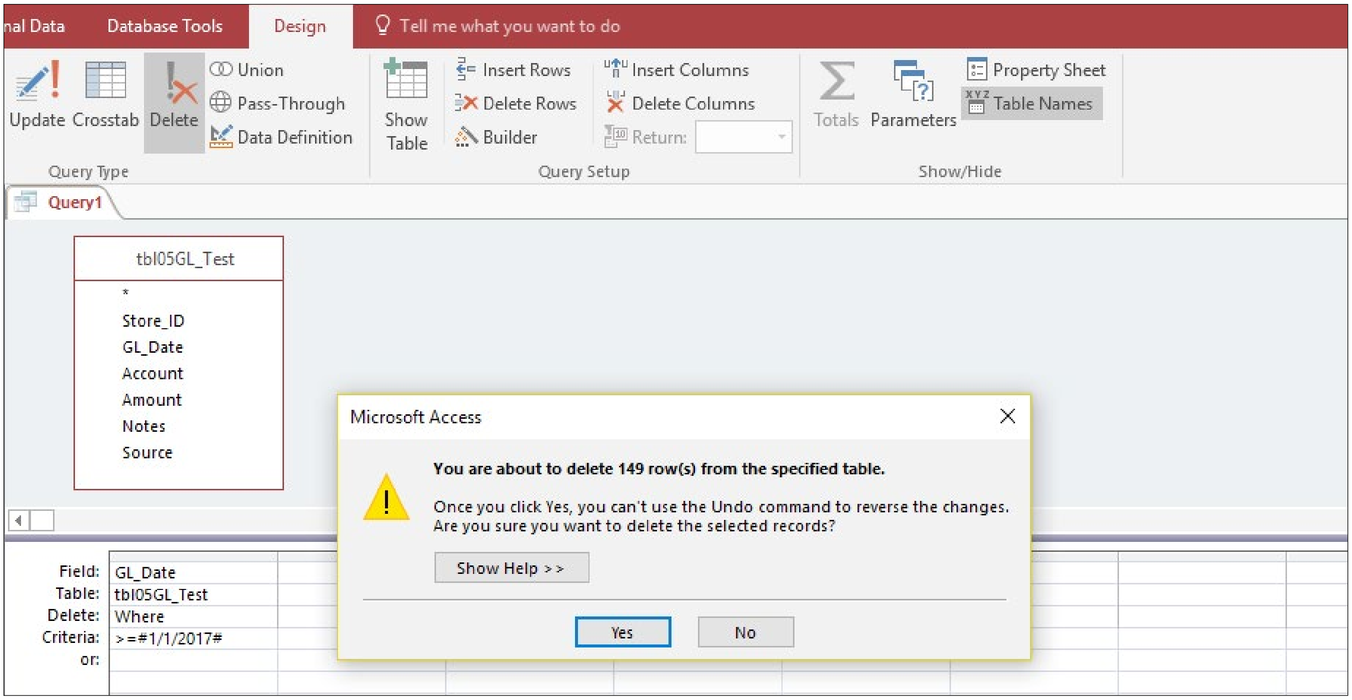
Field:	GL_Date
Table:	tbl05GL_Test
Delete:	Where
Criteria:	>=#1/1/2017#
or:	

Figure 5.23

5. *View the record set in* **View** *mode, but don't Run the query yet to see the bottom of* **Figure 5.23**.

You should get 149 records with GL dates greater than or equal to 1/1/2017. These are the records that will be deleted.

6. *Return to* **Design View***.*
7. *Click the* **Run** *icon.*



Microsoft Access

You are about to delete 149 row(s) from the specified table.

Once you click Yes, you can't use the Undo command to reverse the changes. Are you sure you want to delete the selected records?

Field:	GL_Date
Table:	tbl05GL_Test
Delete:	Where
Criteria:	>=#1/1/2017#
or:	

Figure 5.24

8. *Click* **Yes***.*
9. **Save** *the query as* ***qry05Delete_2017****.*
10. *Close the* **qry05Delete_2017** *query.*

> ***Note:*** *In practice, you shouldn't name a table or query with a year number in the name. That tends to be problematic when the database is updated in future years.*

Data Definition Query

The next type of action query I want to discuss is a ***Data Definition***, or a customized SQL query. This is a type of query that you need to create using SQL (Structured Query Language) code as there are no pre-designed query templates for it. Sometimes you will need to delete a table entirely from the database. You'll now write a ***Drop Table Query*** using SQL code. Don't get too worried – you'll learn much more about SQL in detail in the last few chapters of this course. For now, just do exactly what I tell you and we'll talk about the theory of it later. Let's first create the query that will delete or drop tbl05GL_Test (to "drop" a table means to delete the entire table).

1. *Open a new query in* **Design View***.*
2. *Close the* **Show Table** *dialog box without choosing a table.*

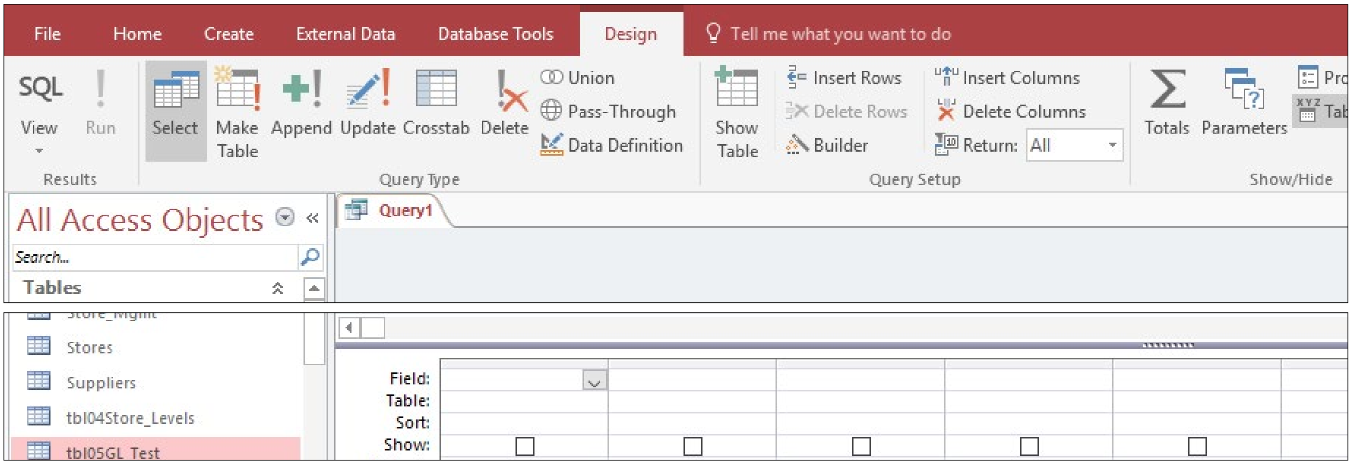
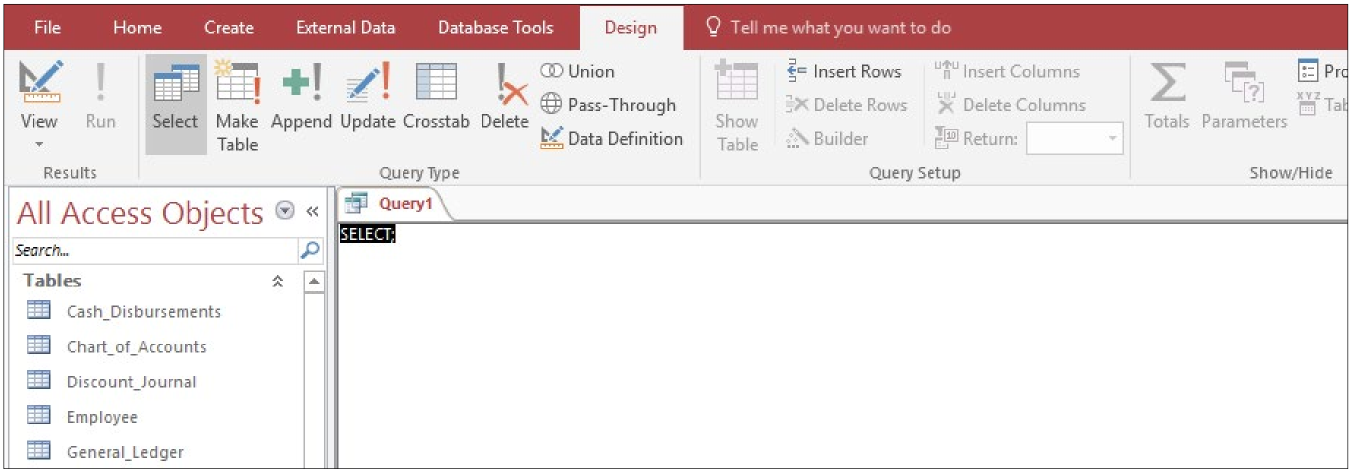


Figure 5.25

If you look on the left side of the Office Ribbon, you'll see the View icon changed to SQL. Click on the SQL icon and it will open up an almost blank screen with the word SELECT; in it.

3. *Click once on the* **SQL** SQL View *icon.*

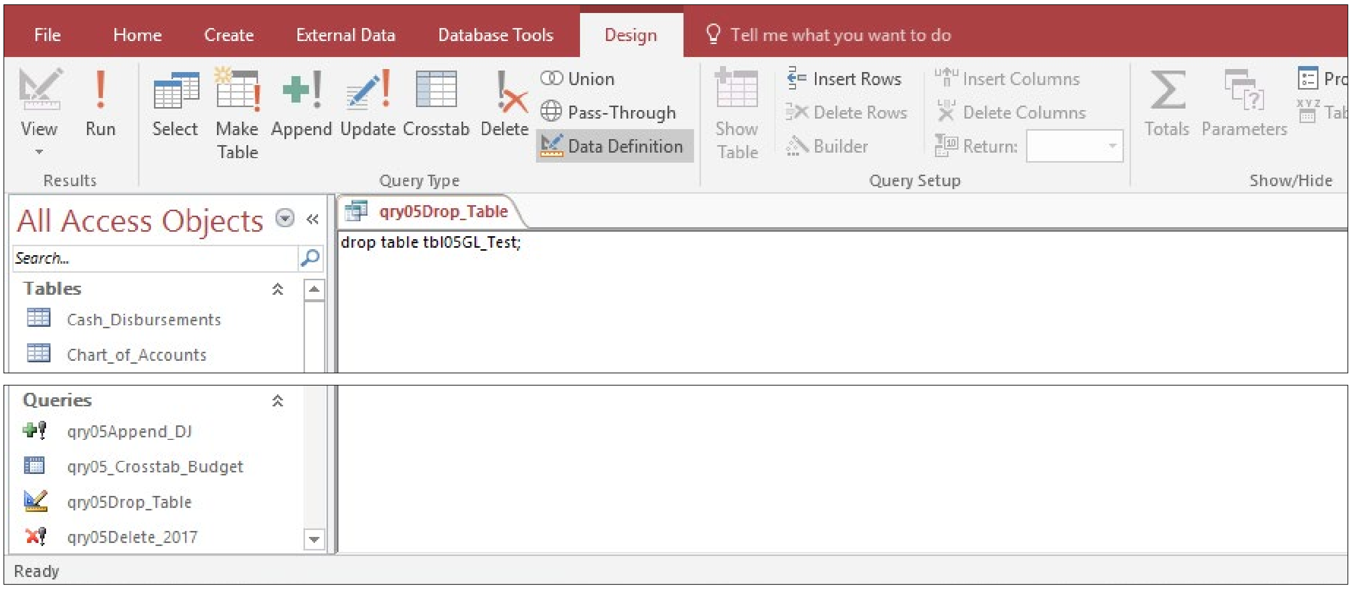


```
SELECT;
```

Figure 5.26

This is the screen where you will write SQL code. For now, I just want you to type the code explained in the following steps:

4. **Delete** *the* **Select;** *text and type* ***drop table tbl05GL_Test;***
5. *Click on the* **Save As** *icon and save the query as* ***qry05Drop_Table****.*



```
drop table tbl05GL_Test;
```

Figure 5.27

6. *Close the* **qry05Drop_Table** *query.*

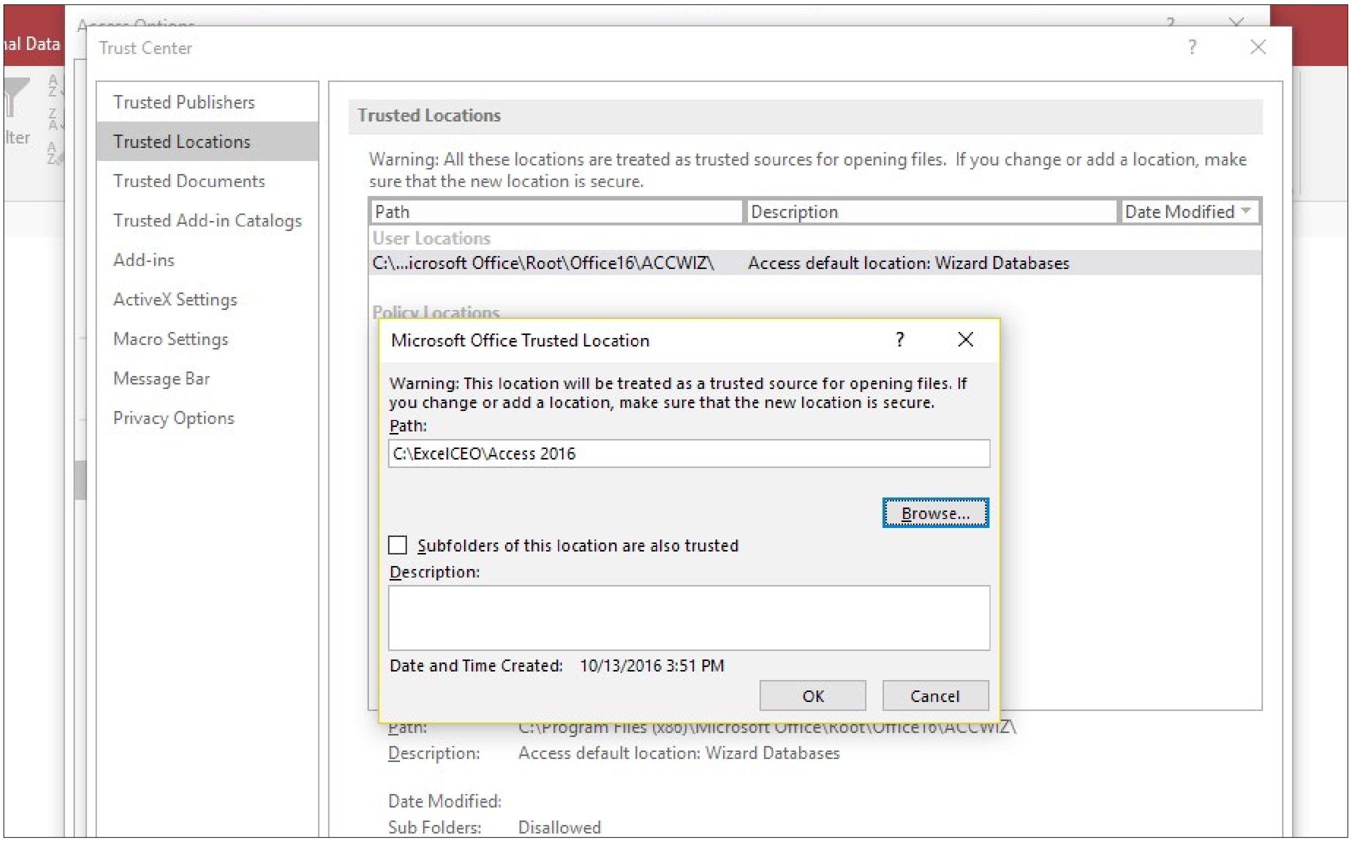
This code executes the command to drop or delete the entire table — not just the data in the table, but the entire table itself. Be careful with this code. As you can imagine, code like this can do some major damage to your databases, if not used properly. If you look at qry05Drop_Table in the Navigation Pane, you'll see the icon next to the query name is different — it looks like the Design View icon. That is because this type of query, called a Data Definition Query, can only be designed in SQL mode. We'll run this query in the next exercise.

Macros

Now that you've seen the things that action queries can do, it's up to your own imagination on how you can use them. One problem with action queries that I've seen is that users may forget to run one or more of the action queries, and people may think the tables are updated correctly, when in actuality they are not. To help reduce the risk of the queries not being run, you can create a macro that someone runs, or you can even set it to run every time the database is opened. A ***Macro*** is simply an action, or set of actions, that you program to automate tasks. You can create macros that make tables, append rows, delete rows, update data, and a host of other actions. Let's briefly go over macros in Access.

Before we create a macro, we need to make sure the database is located in a trusted location.

1. *Click on the* **File** *tab, and click on the* **Options** *button.*
2. *Click on* **Trust Center***, then click on the* **Trust Center Settings…** *button.*
3. *Click on* **Trusted Locations***. Make sure the* **Allow Trusted Locations on my network (not recommended)** *check box is checked, and click the* **Add new location…** *button.*
4. *Browse to the* **C:\ExcelCEO\Access 2016** *folder (or whichever file path on which you stored this course) and click* **OK***.*



Trusted Locations

Warning: All these locations are treated as trusted sources for opening files. If you change or add a location, make sure that the new location is secure.

Path	Description	Date Modified
User Locations		
C:\...icrosoft Office\Root\Office16\ACCWIZ\	Access default location: Wizard Databases	
Policy Locations		

Microsoft Office Trusted Location

Warning: This location will be treated as a trusted source for opening files. If you change or add a location, make sure that the new location is secure.

Path: C:\ExcelCEO\Access 2016

Browse...

□ Subfolders of this location are also trusted

Description:

Date and Time Created: 10/13/2016 3:51 PM

OK Cancel

Path: C:\Program Files (x86)\Microsoft Office\Root\Office16\ACCWIZ\
Description: Access default location: Wizard Databases

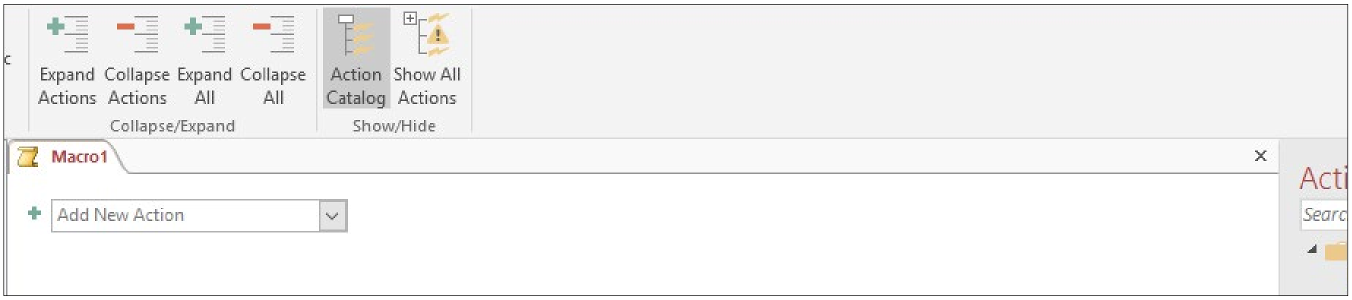
Date Modified:
Sub Folders: Disallowed

Trust Center

5. *Click* **OK** *in the next few boxes to exit out of* **Access Options**.

Let's suppose you want to create a macro that will run the qry05GL_Test Make Table query, then it will append data using the qry05Append_DJ data. One problem you may have is that you need to delete (or drop) the tbl05GL_Test table before you can run the make table query so you can start with a new table. If the table to be dropped does not exist, it will generate an error message. We will now create a macro that will run the drop table query, then we'll run the query to create the new table, and lastly, we'll run the Append query.

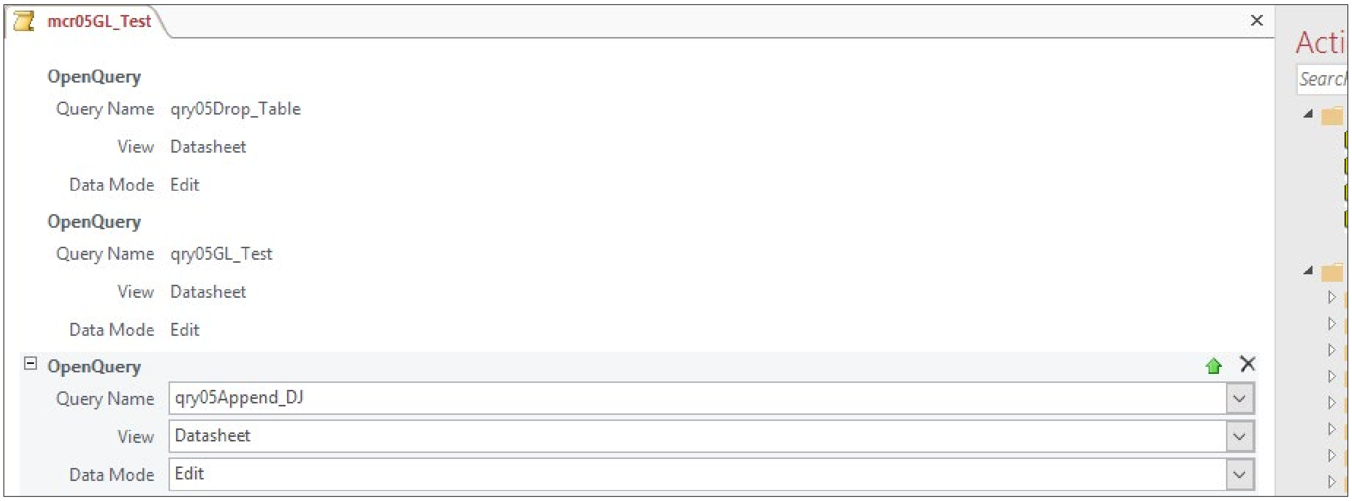
6. *Click on the* **Create** *tab and choose* **Macro** Macro *from the* **Macros & Code** *group.*



Add New Action

Figure 5.28

7. *Click on the* **Add New Action** *drop-down box and choose* **OpenQuery**.
8. *In the* **Query Name** *drop-down box, choose* **qry05Drop_Table** *(leave the* **View** *option as* **Datasheet** *and the* **Data Mode** *as* **Edit***).*
9. *Click in the new* **Add New Action** *drop-down box under the first* **OpenQuery** *item, choose another* **OpenQuery***, and choose* **Query Name: qry05GL_Test**.
10. *Create another* **OpenQuery** *action and choose* **Query Name: qry05Append_DJ**.
11. *Click the* **Save As** *icon and save the macro as* ***mcr05GL_Test***.


mcr05GL_Test

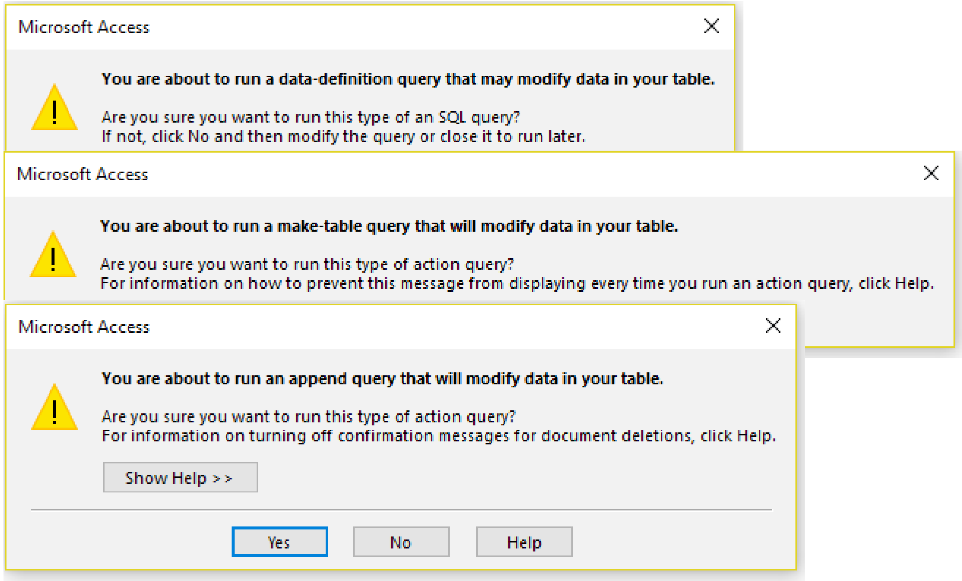
OpenQuery
Query Name qry05Drop_Table
View Datasheet
Data Mode Edit

OpenQuery
Query Name qry05GL_Test
View Datasheet
Data Mode Edit

OpenQuery
Query Name qry05Append_DJ
View Datasheet
Data Mode Edit

Figure 5.29

12. *Close the* **mcr05GL_Test** *macro.*
13. *In the newly created* **Macros** *section (located below the* **Forms** *section) in the* **Navigation Pane***, double-click* **mcr05GL_Test**.


Microsoft Access

You are about to run a data-definition query that may modify data in your table.

Are you sure you want to run this type of an SQL query?
If not, click No and then modify the query or close it to run later.

Microsoft Access

You are about to run a make-table query that will modify data in your table.

Are you sure you want to run this type of action query?
For information on how to prevent this message from displaying every time you run an action query, click Help.

Microsoft Access

You are about to run an append query that will modify data in your table.

Are you sure you want to run this type of action query?
For information on turning off confirmation messages for document deletions, click Help.

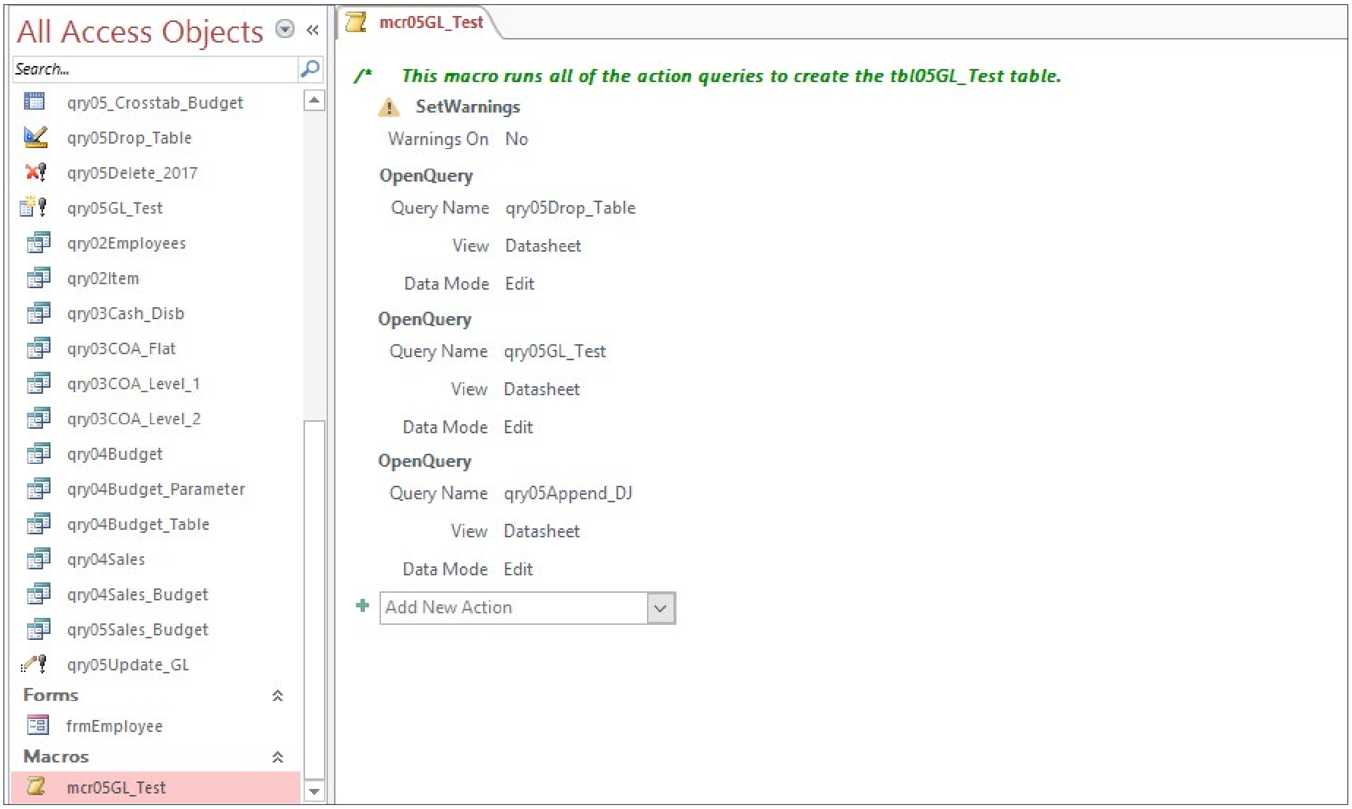
Warning boxes

After you double-click the macro, you will see a series of dialog boxes asking you to make sure you want to perform those individual actions. If you are like me, you don't want to have to click Yes in every one of those dialog boxes. We're really confident in our macro, so let's disable those warning messages.

14. *Click* **No** *in each of the three warning boxes until the messages disappear.*

Now let's set the SetWarnings action.

15. *Return to* **Design View** *of* **mcr05GL_Test**.
16. *In the* **Office Ribbon** *under the* **Macro Tools Design** *contextual tab, click on the* **Show All Actions** *icon (this shows some of the options that are not typically available as a macro action).*
17. *Click on the* **Add New Action** *drop-down box and choose* **SetWarnings** *(leave* **Warnings On** *set to* **No**).
18. *Click on the green* **Move Up** *arrow to the far-right of the* **SetWarnings** *action. Move the* **SetWarnings** *action up to the top of the action list.*
19. *Click on the* **Add New Action** *drop-down box, choose* **Comment**, *and write the following comment:* ***This macro runs all of the action queries to create the tbl05GL_Test table.***
20. *Move the* **Comment** *action to the top of the macro.*
21. *Click in a blank area of the* **Macro Design** *screen.*



/* This macro runs all of the action queries to create the tbl05GL_Test table.

SetWarnings
Warnings On No

OpenQuery
Query Name qry05Drop_Table
View Datasheet
Data Mode Edit

OpenQuery
Query Name qry05GL_Test
View Datasheet
Data Mode Edit

OpenQuery
Query Name qry05Append_DJ
View Datasheet
Data Mode Edit

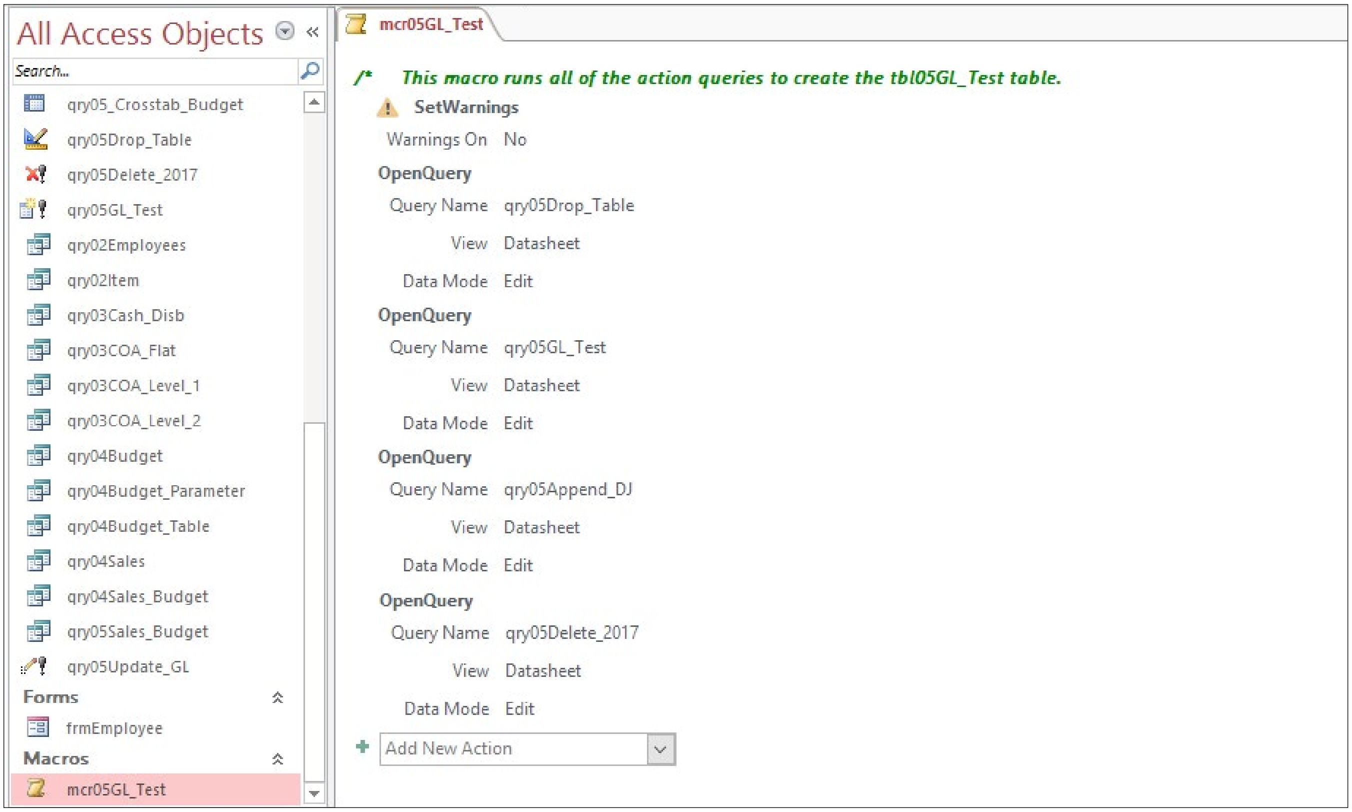
Add New Action

Figure 5.30

22. **Save** *and* **Run** *the* **mcr05GL_Test** *macro.*

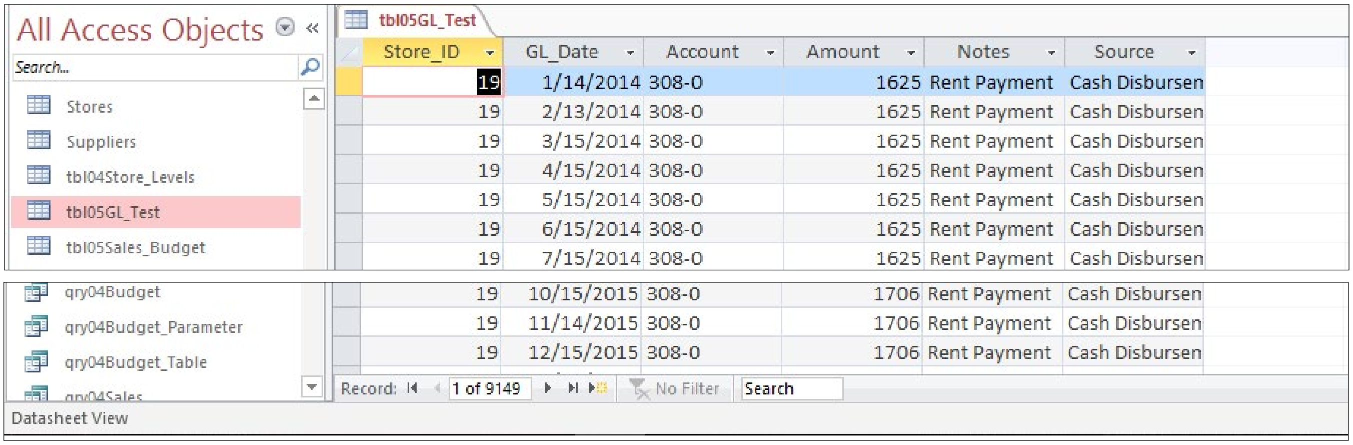
If you open tbl05GL_Test, it should contain 9,298 records. We forgot to run the query that takes out the 2017 data, but you can easily edit the macro to include that query.

23. *Edit the macro to include* **qry05Delete_2017** *as the last action.*
24. **Save**, *close, and test the macro.*



```
/*  This macro runs all of the action queries to create the tbl05GL_Test table.
  SetWarnings
    Warnings On  No
  OpenQuery
    Query Name  qry05Drop_Table
          View  Datasheet
     Data Mode  Edit
  OpenQuery
    Query Name  qry05GL_Test
          View  Datasheet
     Data Mode  Edit
  OpenQuery
    Query Name  qry05Append_DJ
          View  Datasheet
     Data Mode  Edit
  OpenQuery
    Query Name  qry05Delete_2017
          View  Datasheet
     Data Mode  Edit
+ Add New Action
```

Figure 5.31



Store_ID	GL_Date	Account	Amount	Notes	Source
19	1/14/2014	308-0	1625	Rent Payment	Cash Disbursen
19	2/13/2014	308-0	1625	Rent Payment	Cash Disbursen
19	3/15/2014	308-0	1625	Rent Payment	Cash Disbursen
19	4/15/2014	308-0	1625	Rent Payment	Cash Disbursen
19	5/15/2014	308-0	1625	Rent Payment	Cash Disbursen
19	6/15/2014	308-0	1625	Rent Payment	Cash Disbursen
19	7/15/2014	308-0	1625	Rent Payment	Cash Disbursen
19	10/15/2015	308-0	1706	Rent Payment	Cash Disbursen
19	11/14/2015	308-0	1706	Rent Payment	Cash Disbursen
19	12/15/2015	308-0	1706	Rent Payment	Cash Disbursen

Figure 5.32

You should now have 9,149 records in tbl05GL_Test, exactly 149 records less than 9,298. IT WORKS!!!

> ***Review Questions**: It is now time to complete the hands-on Review Questions. Log on to www.ExcelCEO.com with your Email and Password, click on the* **Access 2016 and SQL Review Questions, Chapter 5, Section 2 of 2** *option and complete the review questions.*

Conclusion

You may think that I'm spending too much time on queries, but it is crucial to learning how to manipulate data. Getting the right data and doing the right calculations is the majority of the work. Once you get the right data set, presenting it in a report, or an Excel file, or even on an Internet site is the easy part. If you've made it through this far in the Access course and you understand all of the concepts, you're now on the downhill side. The rest should be a piece of cake.

In this chapter, you learned about action queries. We started off with crosstab queries, which are a type of PivotTable in Access. Then we went through each of the action queries (make table, update, append, and delete). You got a sneak preview of writing SQL code when we created a drop table query. Finally, we put all of the action queries into one macro for automation purposes.

Chapter Exam

You can now go to www.ExcelCEO.com, click on Sign In, log in and take the exam. Make sure that you take the exam on the same device on which you completed the practice files, as some of the questions on the exam may refer to some of the completed projects. Chapter exams are intended to be hands-on.

Access® 2016 and SQL

Complete Self-study Course

ExcelCEO

Chief Excel Officer

CHAPTER SIX — FORMS

In this chapter, you will:

- Determine the correct way to open an existing form and to create a Bound Form
- Recognize how to add records into a table using a form as an interface
- Select and use Form Headers, Footers, and Detail
- Identify Form Rulers and how to use Snap-to-Grid functionality
- Determine how to create a form using the Form Wizard
- Identify the various components of the Design View of a form
- Select graphics and include them in a form
- Recognize Labels and Text Boxes within a form
- Identify the Tab Order of Text Boxes

CPE Credits possible for this chapter: 3

Introduction to Forms

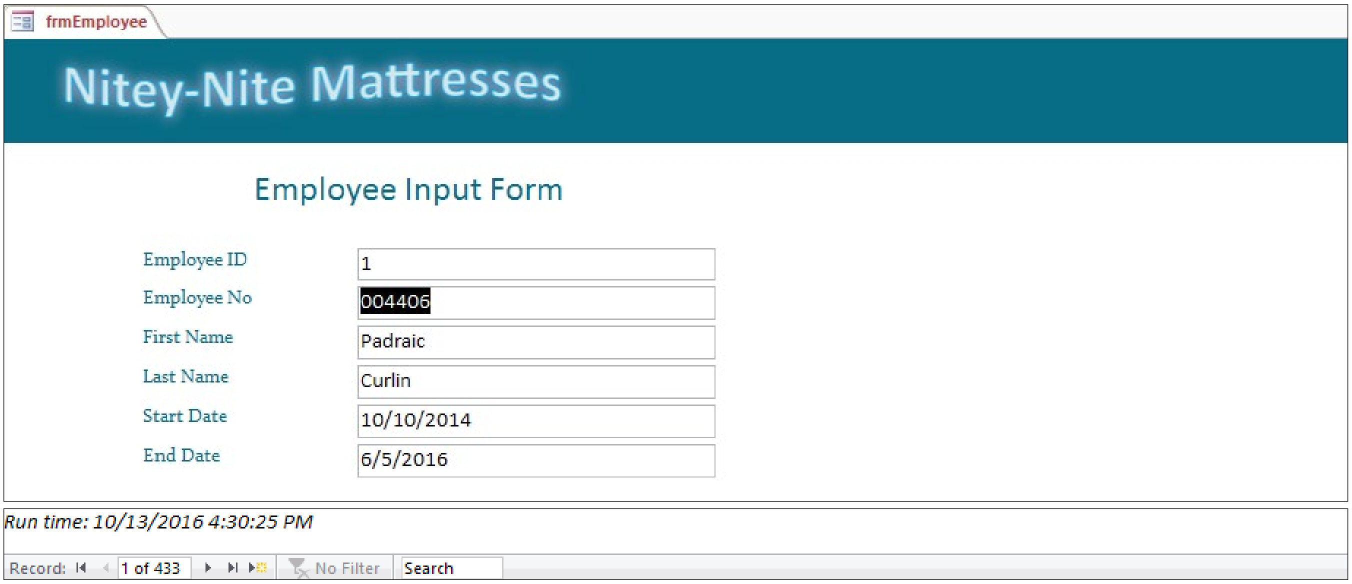
As long as you are the only person who will ever use your database (which is most likely not the case), you will probably be comfortable inputting data directly into tables and navigating around the different tabs. However, if you need to get someone to help you with entering data or running reports, they may not be familiar with Access and it will be clunky at best for them to use it. To make the task easier, you can create forms to facilitate data entry and for navigation within the database. Once I had an assistant who I tried to teach how to use Access. She wasn't familiar with action queries and macros, and the analysis we had to run over and over again was full of complex queries, macros, and SQL code. The experience for her was disastrous at best. If I had had enough foresight, I could have used forms to facilitate the process, and might have convinced her that Access is a good tool. I've since repented.

A ***form*** is basically an organized view of the fields from one or more tables or queries. A ***bound form*** is based on a specific table or query, and it works interactively with that table or query in the database. There are various controls that you can create on the form to assist in entering new data, viewing data, editing or deleting data, or finding information. Some of the available controls include labels, text boxes, drop-down menus (also called combo boxes), and check boxes. You need to be creative in designing the form so the user can efficiently input and view the data displayed by the form. You can also use an ***unbound form*** (one that is not based on a table or query) to display menus that help users navigate within the database.

In this chapter, you will learn about the different kinds of forms, how to create and edit a form, and input data into a form. The first form that we will work with is an example of a bound form, or a form that is tied to a specific table. Probably the best way to start off our exploration of forms is to see one in action.

Bound Forms

1. *In the* **Nitey_Nite_2016** *database under the* **Forms** *tab, double-click on* **frmEmployee** *to open it.*



Nitey-Nite Mattresses

Employee Input Form

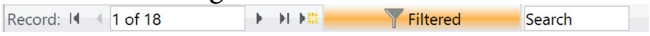
Employee ID	1
Employee No	004406
First Name	Padraic
Last Name	Curlin
Start Date	10/10/2014
End Date	6/5/2016

Run time: 10/13/2016 4:30:25 PM

Figure 6.1

The form opens up to a view similar to Figure 6.1. This bound form is tied directly to the Employee table. Instead of viewing and inputting data directly into that ugly table (like when you input your name as the newest Nitey-Nite employee in Chapter 1), you can use a form like this that looks a lot nicer and is easier to use. When designing a form to be used by others, you need to keep in mind the skill level of the users. Most of the time, people who input data into databases aren't advanced Access users, and I therefore try to make the forms as easy to use as I can. Let's review this form.

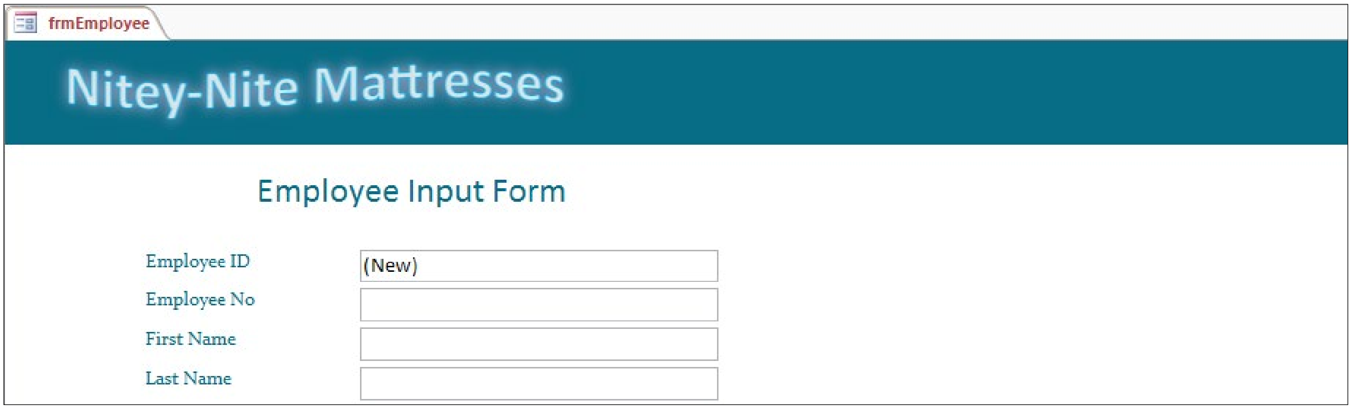
When you open the form, it displays the data for the first employee, Padraic Curlin. From the data in the form, we know her Employee ID is 1, her Employee Number is 004406, she started on October 10, 2014, and she ended her employment with Nitey-Nite on June 5, 2016.

In the upper-left corner of the form, you see the same graphic that you created in the Excel course. All the form designer did was to import that jpeg file into the design view of the form. It doesn't really do anything except to make the form look more official. Next is the label, "Employee Input Form". This is the title of the form. It's not absolutely necessary, but it helps the users know what the form is and does. The next few items are the meat of the form. The labels to the left (Employee ID, Employee No, First Name, Last Name, Start Date, and End Date) are simply descriptors of the data contained in the text boxes to the right. The text boxes contain the data in the Employee table. The ***record navigators***  are located at the bottom-left of the form. It is similar to the tab selectors in Excel, but it displays (or sets focus on) one individual record at a time. You can click the first record, previous record, next record, last record, or new record selectors to scroll through each record in the table. Lastly, the designer of this form created a message at the bottom of the form that tells the date and time when the form was last used. This is helpful when someone prints out a form or a report and you are reviewing the printout later. Information can change, and knowing when a form or report was printed helps in auditing the database.

Adding Records

Using a form like this to input records is easy. The form is already linked to the Employee table, and inputting records is just like inputting them directly into the Employee table. Let's input a few records so you can see how it works.

2. *Click the* **New Record** *selector to begin a new record.*



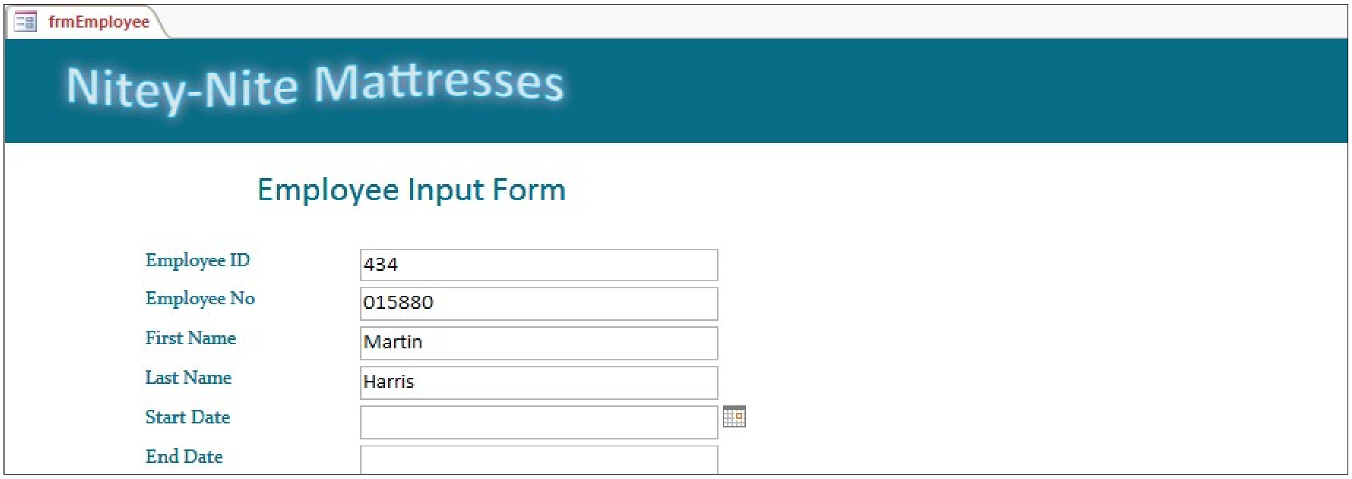
Employee Input Form

Employee ID	(New)
Employee No	
First Name	
Last Name	

Figure 6.2

Notice that the Employee ID field reads (New). That is because that field is tied to the Employee_ID field in the Employee table, which is an AutoNumber field. As such, you can't input anything into that field. If you try, it will display an error message at the bottom of the page.

3. *Click on the* **Employee No** *text box (or tab over to it) and type* ***015880****, and click on the* **First Name** *text box.*
4. *Type* ***Martin*** *in the* **First Name** *box,* ***Harris*** *in the* **Last Name** *box, then tab over to the* **Start Date** *box*



Employee ID	434
Employee No	015880
First Name	Martin
Last Name	Harris
Start Date	
End Date	

Figure 6.3

Notice that when your cursor is in the Start Date text box, an icon of a calendar appears to the right. This icon (or control) allows the user to choose a date instead of typing it in. Some users may unintentionally input a date like February 30, so instead of inputting an invalid or incorrect date, the user can use this control to choose a date. This was a new feature in Access 2010.

5. *Click on the* **Calendar** *control next to the* **Start Date** *box and choose* **January 12, 2016**.
6. *In the* **End Date** *box, type* ***1/1/2099*** *(to indicate Martin is a current employee), and press the* **[Tab]** *key.*

You have just input a new record into the Employee table by using the form. When you press the [Tab] key, the form allows you to begin another new record. Let's input a couple more new employees.

7. *Input the following employees using the* **Employee Input Form***:*

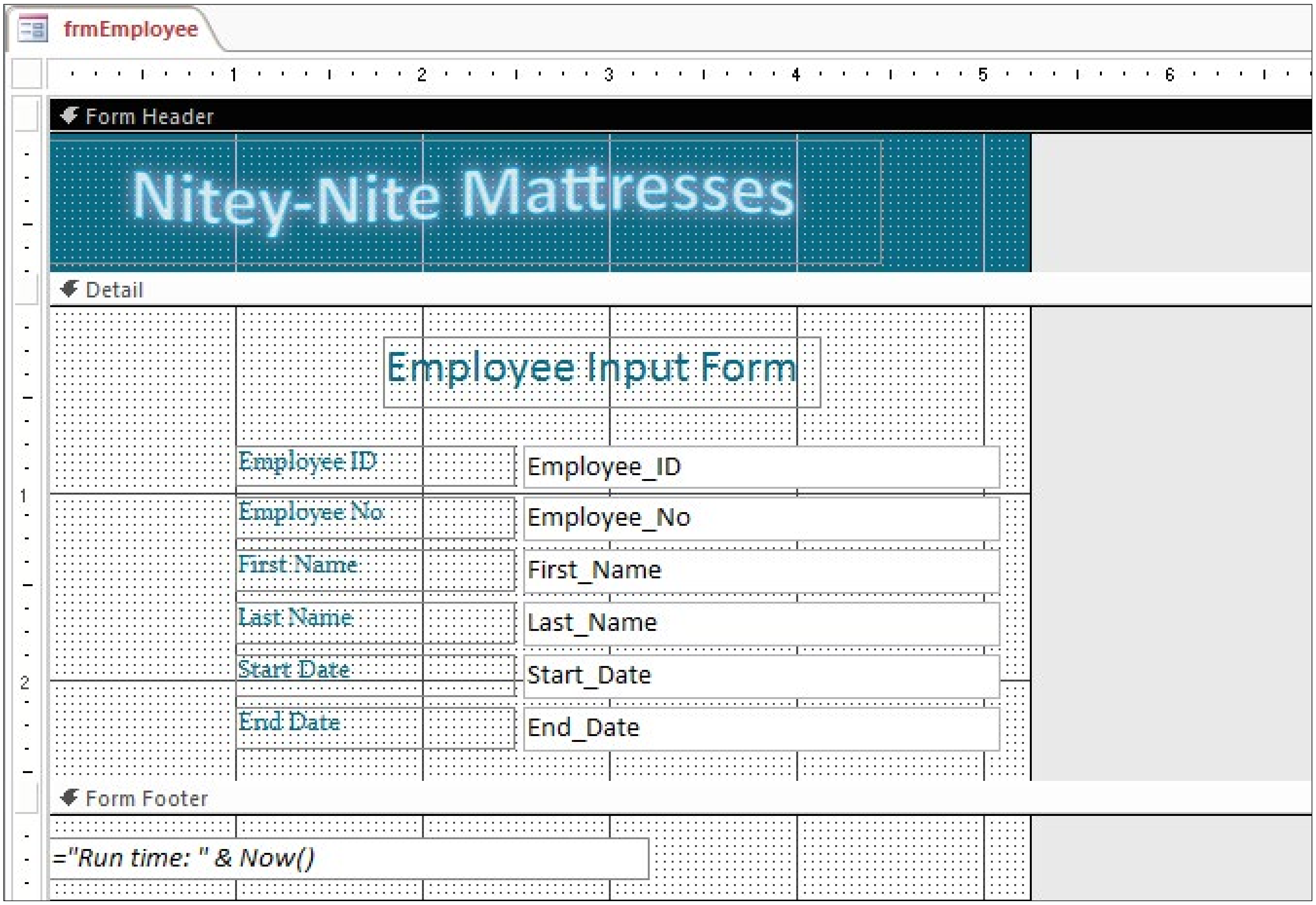
Field	Employee	Employee
Employee No	***015884***	***015892***
First Name	***Emma***	***Oliver***
Last Name	***Smith***	***Cowdery***
Start Date	***1/15/2016***	***1/20/2016***
End Date	***1/1/2099***	***1/1/2099***

> ***Note**: Be careful! You must start a new record to input a new record. If you have an existing record displayed when you input the new record, it will overwrite the data on that record.*

You should now have 436 employees (or records) in the Employee table (if you clicked Tab to finish the last record, the record selector should show 437 of 437 for the new record). Isn't this a lot easier to input data into this form instead of directly into the table?

Now that you know how to input data using a form, you are ready to learn how to design a form. But before you create one on your own, let's explore the behind-the-scenes design view of this existing form.

8. *Click the drop-down arrow under the* **View** *icon on the* **Home** *tab and choose* **Design View***.*



="Run time: " & Now()

Figure 6.4

Form Header, Detail, and Form Footer

It may look kind of scary, but don't worry – we'll go over everything in detail. In a basic form, there are three sections: the Form Header, Detail, and Form Footer. As the names imply, the Form Header and

Form Footer sections appear at the top and bottom of the form, respectively. The meat of the form is in the ***Detail section***.

In the ***Form Header*** section of this form, there is only one object. It should look familiar to you — it's the Nitey-Nite Mattresses logo you used in the Excel course. As it is in the Form Header section, it will appear at the top of the screen. As you already know, the image is a simple .png file that can be imported into a variety of applications, including Access forms and reports. Here the form designer imported the logo.png file and placed it in the Form Header section.

In the Detail section, the designer tied all of the fields in the Employee table to text boxes, and created descriptive labels to the left of each text box to let the user know what each text box is. A ***text box*** is a control used in forms or reports that displays data or serves as a mechanism to input or edit data. A text box can be bound (tied to a table or query), unbound (not tied to a table or query), or calculated (displays the result of an expression). A well-designed form should be simple enough for any user to figure out, with minimal instruction (if any at all), what information is contained in the form and how to use it.

In the ***Form Footer*** section, there is only one object. It is a text box that contains a formula. The formula is ="Run time: "&Now(). You may not notice that it is a text box because of its formatting. The designer formatted the text box to be transparent with no lines surrounding it. Therefore, in the View mode of the form, it appears to be text that is simply typed onto the bottom of the form with no background or lines around it. In reality, it is a formula that returns the current date and time.

Form Rulers

You will notice in the Design View of the form that there are rulers across the top and left sides of the screen. Rulers give the designer a good idea of where each object will appear when the form is shown in View mode. Another nifty feature of the rulers is that you can place your cursor over the ruler and it changes to an arrow. When it changes to an arrow, you can click and drag to select objects in that section of the form. We'll explore that functionality when you create your first form.

Snap to Grid

Another great tool in Forms (and in designing reports as well) is the ***Snap-to-Grid*** functionality. If you notice, there are small dots within the body of the form Design View. The dots make up the "grid" of the form, and assist the designer in aligning the labels, text boxes, and other objects used in the form. You can move form objects around by simply clicking and dragging the objects. Instead of having free-floating objects, you can turn on the Snap-to-Grid functionality so that the objects will automatically align within the grid as they are moved around.

Toolbars and Icons

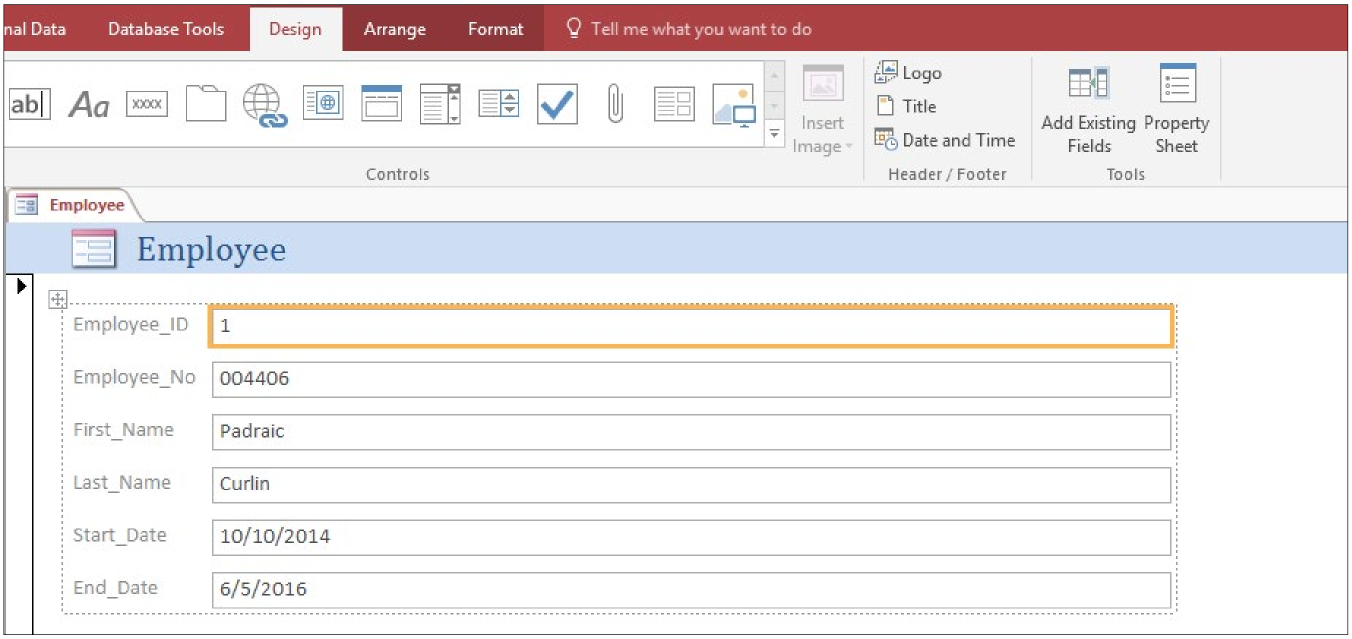
As with Excel, Access 2016 contains a ribbon that houses the various icons available for use within the form Design View. You are already very familiar with the Office Ribbon and using icons, so I won't go into an in-depth explanation. Most of the icons you should already know, and it will not be a difficult task to learn how to use some of the new ones.

Form Creation

There are a number of ways you can create a form, and we'll review many of them here. In the next few exercises, you will create a form using the various icons in the Create tab on the Office Ribbon. For all of these examples, we'll base the forms on the Employee table. Since these forms you'll create are based on a table, by definition they will be bound forms.

In Access 2016, Microsoft has made it much easier to create forms. In Access 2003, you basically had two choices: create a form from scratch or use the Form Wizard. In Access 2016, the Office Ribbon gives you a number of other choices with which to build a form. These choices, indicated by icons in the Forms section of the Create tab, work like a wizard, making a number of assumptions and creating the form for you in one step. Let's create a form by using the Form icon.

1. *Close the form* **frmEmployee** *(if it asks you if you want to save, choose* **No** *as you should not have made any changes).*
2. *Under the* **Tables** *section, click on the* **Employee** *table, but do not open it.*
3. *In the* **Forms** *section of the* **Create** *tab, click on the* **Form** *icon.*



Employee_ID 1
Employee_No 004406
First_Name Padraic
Last_Name Curlin
Start_Date 10/10/2014
End_Date 6/5/2016

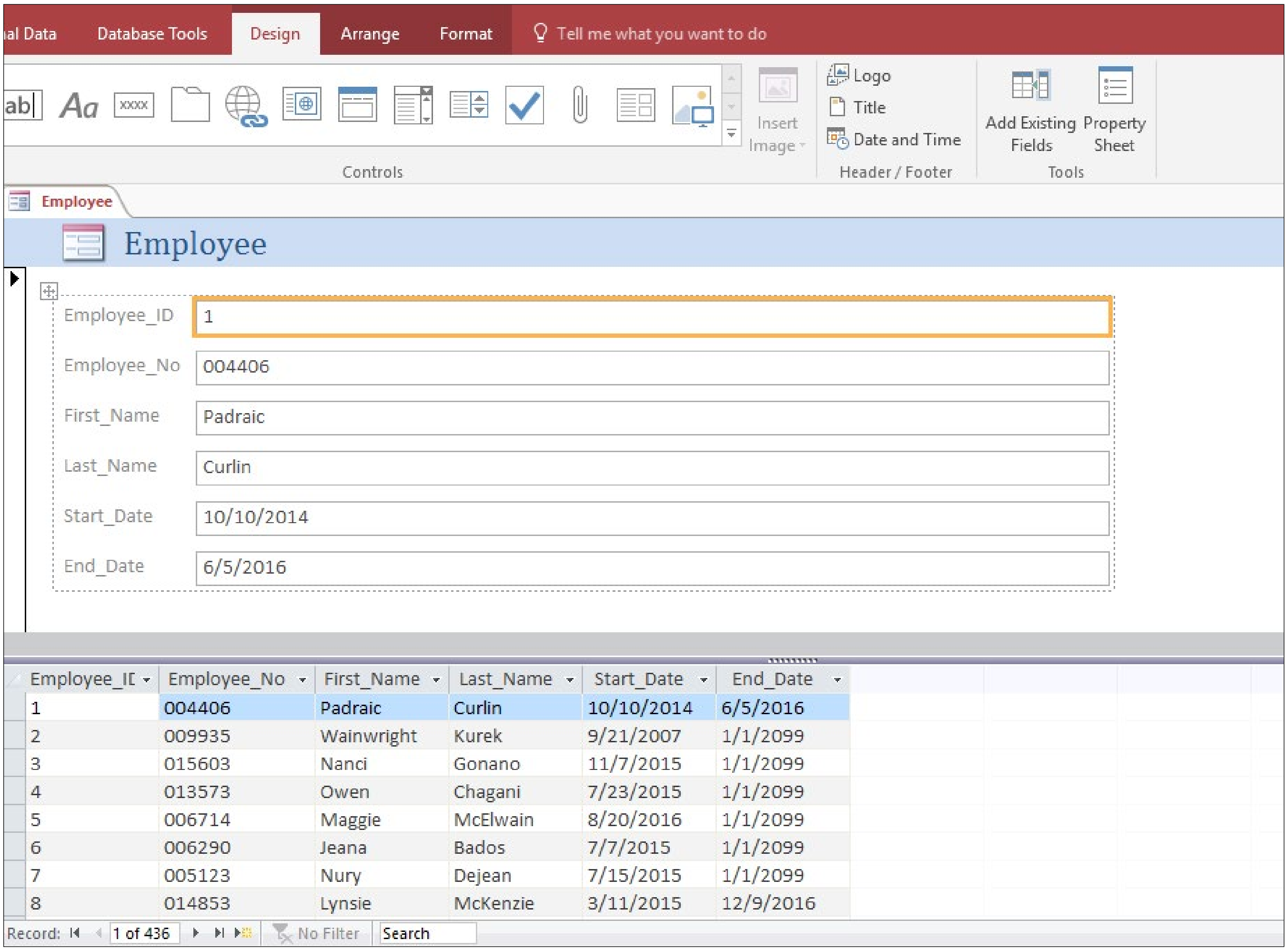
Figure 6.5

With the Employee table selected, Access assumed that you want to create a bound form on that table, so it created labels to the left that contain the names of the fields, and text boxes next to them containing the data in the Employee table. In addition, it provided a record selector down the left margin and navigation buttons at the bottom, as well as a title for the form (the name of the table).

Another important upgrade over Access 2003 is the inclusion of the Layout View. The ***Layout View*** allows the user to change the design of the form as it actually appears when the user is using it. It's kind of a cross between Form View and Design View. It contains the same drag and drop functionality as in

Design View with the look and feel of the finished form in Form View, containing actual data. We won't do much in Layout View in the following exercises, but I wanted you to see it at least once. It can help an inexperienced user make changes to a form without the complexity and intimidation of doing it in Design View.

4. *Close the* **Employee** *form without saving it.*
5. *In the* **Create** *tab, click on the* **Split Form** Split Form *icon that is located under the* **More Forms** *drop-down box.*



Employee

Employee_ID	1
Employee_No	004406
First_Name	Padraic
Last_Name	Curlin
Start_Date	10/10/2014
End_Date	6/5/2016

Employee_ID	Employee_No	First_Name	Last_Name	Start_Date	End_Date
1	004406	Padraic	Curlin	10/10/2014	6/5/2016
2	009935	Wainwright	Kurek	9/21/2007	1/1/2099
3	015603	Nanci	Gonano	11/7/2015	1/1/2099
4	013573	Owen	Chagani	7/23/2015	1/1/2099
5	006714	Maggie	McElwain	8/20/2016	1/1/2099
6	006290	Jeana	Bados	7/7/2015	1/1/2099
7	005123	Nury	Dejean	7/15/2015	1/1/2099
8	014853	Lynsie	McKenzie	3/11/2015	12/9/2016

Record: 1 of 436 No Filter Search

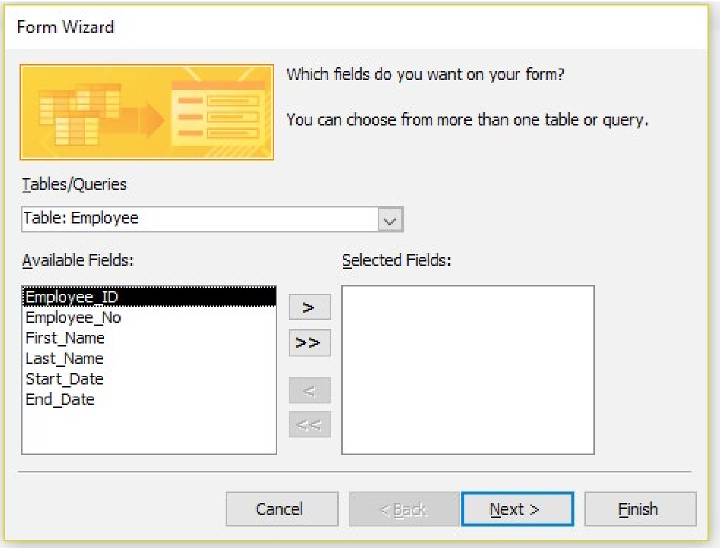
Figure 6.6

Access creates another form with a different layout. This form is split into two sections: the top section that allows the user to input and view data in a typical Form View, and the bottom section in a tabular view of the data. This kind of form can be very useful in certain situations.

6. *Close the* **Employee** *form without saving it.*

In this next exercise, you will create a form by using the Form Wizard. The Form Wizard works like other wizards in Microsoft Office 2016 by walking you through a series of choices and building the form based on those choices. Since you're already familiar with the Employee Input form, we will replicate that form. Let's start it now.

1. *Click on the* **Form Wizard** *icon.*



Form Wizard

Which fields do you want on your form?

You can choose from more than one table or query.

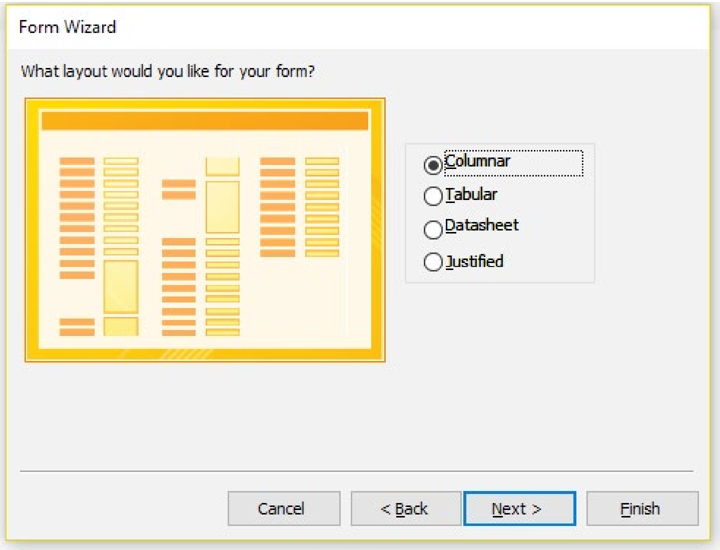
Tables/Queries: Table: Employee

Available Fields:
- Employee_ID
- Employee_No
- First_Name
- Last_Name
- Start_Date
- End_Date

Selected Fields:

Figure 6.7

2. *Click on the* >> *button to move all fields from the* **Available Fields** *section to the* **Selected Fields** *section and click* **Next >**.



Form Wizard

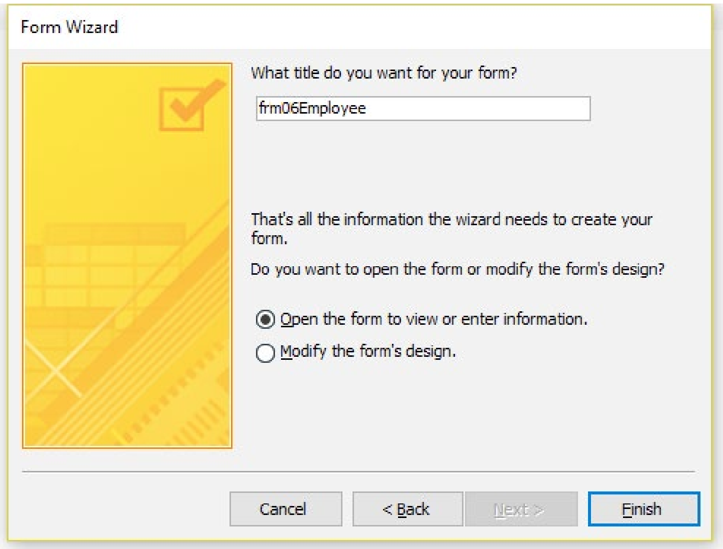
What layout would you like for your form?

● Columnar
○ Tabular
○ Datasheet
○ Justified

Figure 6.8

3. *Make sure the* **Columnar** *radio button is selected and click* **Next >**.

4. *Name the form* ***frm06Employee****.*



What title do you want for your form?

frm06Employee

That's all the information the wizard needs to create your form.

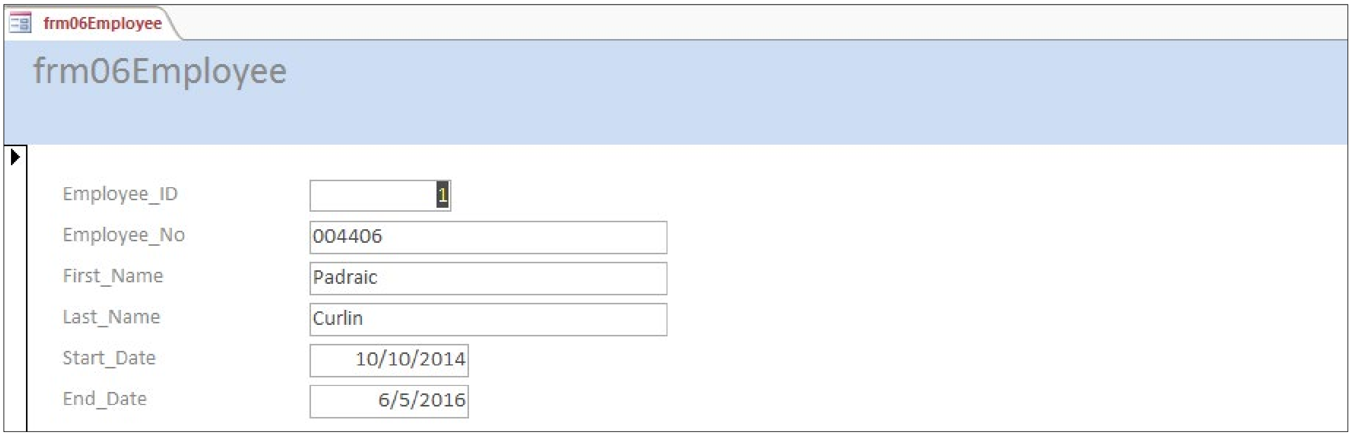
Do you want to open the form or modify the form's design?

● Open the form to view or enter information.

○ Modify the form's design.

Figure 6.9

5. *Click* **Finish**.



frm06Employee

Employee_ID	1
Employee_No	004406
First_Name	Padraic
Last_Name	Curlin
Start_Date	10/10/2014
End_Date	6/5/2016

Figure 6.10

> ***Review Questions****: It is now time to complete the hands-on Review Questions. Log on to www.ExcelCEO.com with your Email and Password, click on the* ***Access 2016 and SQL Review Questions, Chapter 6, Section 1 of 2*** *option and complete the review questions.*

Form Design View

You should get a form that looks like Figure 6.11, but that one doesn't look a lot like the one in Figure 6.1, does it? We used the wizard just to get us the shell of the form, but now we have to go in and modify it to look like Figure 6.1. The wizard has done all of the basics in creating the form – all we have to do now are the cosmetics.

6. *Go to the* **Design View** *of* **frm06Employee***.*

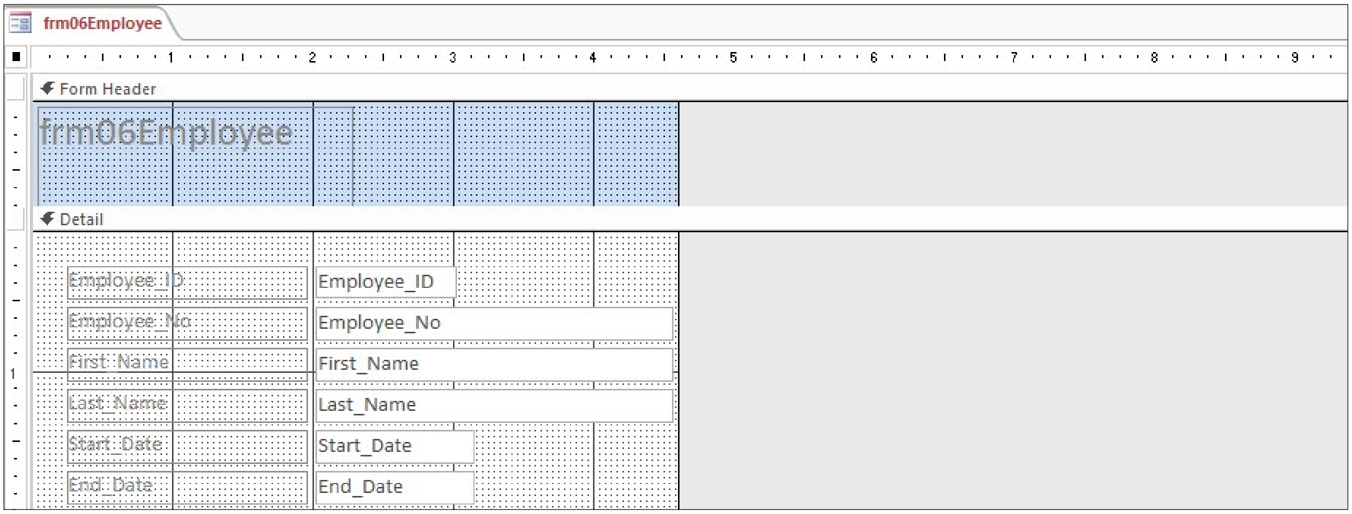


Figure 6.11

It's quite different than the Design View in Figure 6.4, but don't worry – we'll clean it up a bit.

Insert a Graphic

Let's start at the top. The first thing we need to do is to insert the Nitey-Nite Mattresses logo into the Form Header section. As you can see, there is already a label in that section that reads "frm06Employee". We don't need that label, so let's delete it.

7. *Click once on the* **frm06Employee** *label in the* **Form Header** *section of the form and press the* **[Del]** *or* **Delete** *key on your keyboard (you can also right-click on the label and choose Delete).*
8. *Click on the* **Form Header** *bar.*

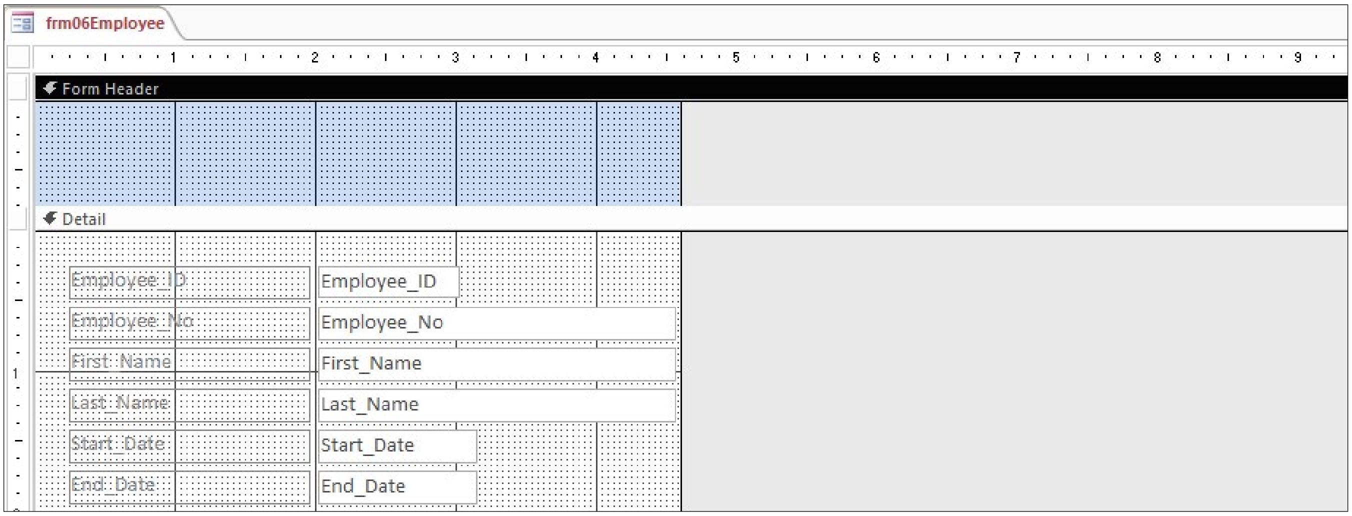
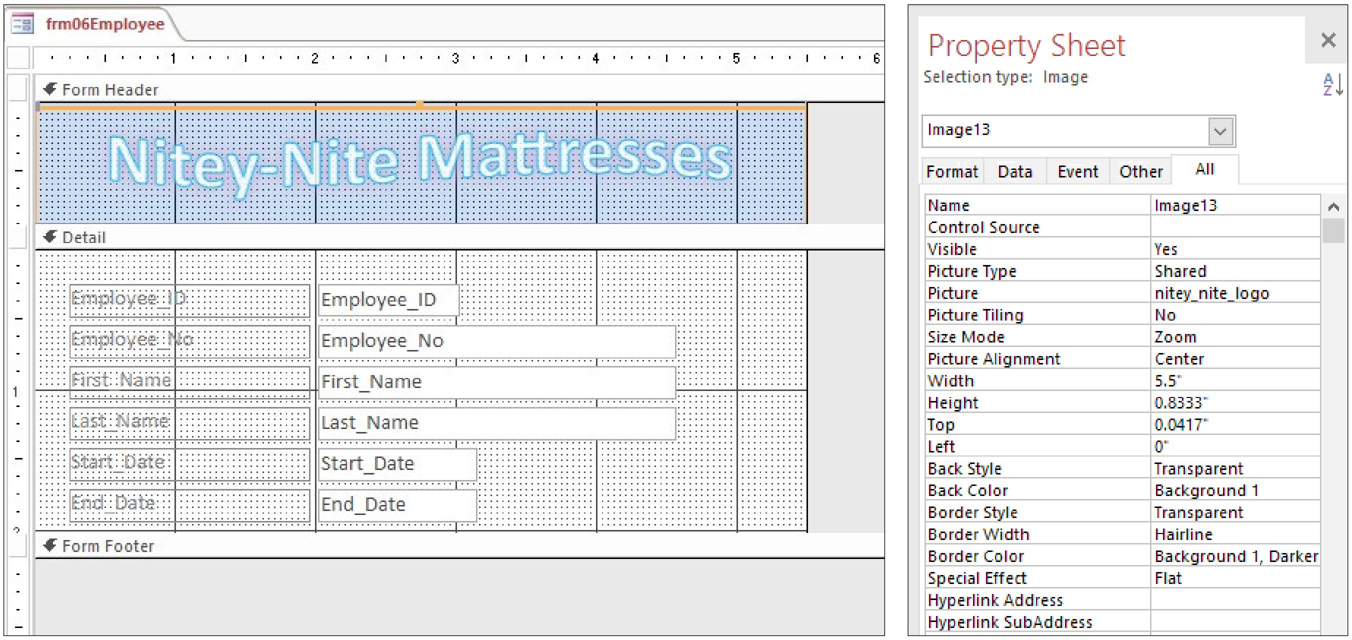


Figure 6.12

You click on the Form Header bar to make that section active. You need to do this because you will now import the .png file and you want to place it into that section.

9. *Under the* **Form Design Tools Design** *contextual tab, click the* **Insert Image** Insert Image *icon in the* **Controls** *group.*
10. *Navigate to the* **C:\ExcelCEO\Access 2016** *folder and double-click on the* **logo.png** *file.*
11. *Using the crosshair cursor, draw a box to make the logo about ¾" from the left of the form to about 5½" using the top ruler. Size the logo to look like* **Figure 6.13**.



Selection type: Image

Image13

Name	Image13
Control Source	
Visible	Yes
Picture Type	Shared
Picture	nitey_nite_logo
Picture Tiling	No
Size Mode	Zoom
Picture Alignment	Center
Width	5.5"
Height	0.8333"
Top	0.0417"
Left	0"
Back Style	Transparent
Back Color	Background 1
Border Style	Transparent
Border Width	Hairline
Border Color	Background 1, Darker
Special Effect	Flat
Hyperlink Address	
Hyperlink SubAddress	

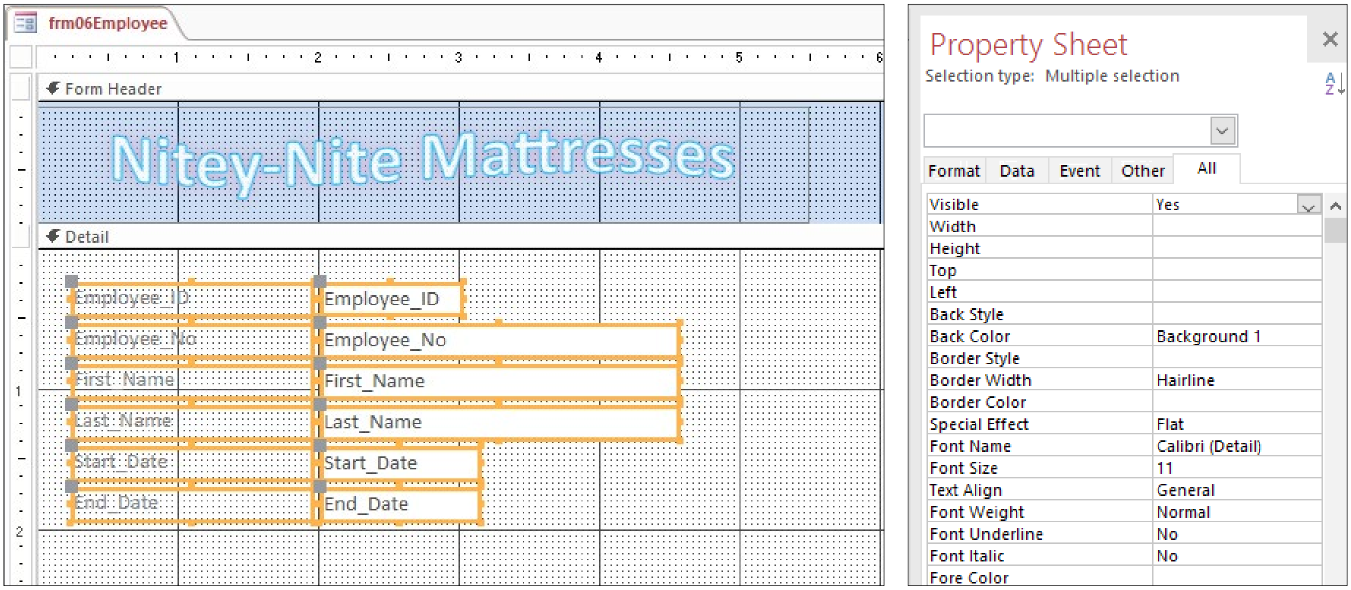
Figure 6.13

> ***Note**: You could also use the **Logo** icon to bring in images. That would import a label in addition to the logo. Also, you would notice the size of the form increases to the right when elements within the **Form View** expand outside the borders. Overall, I find **Insert Image** simpler to use.*

Next, we have to create a little more space to work with in the Detail section.

12. *Place your cursor over the vertical line that makes up the right edge of the **Design Grid**.*
13. *Drag that line over to the **6½"** mark on the top ruler.*
14. *Activate the **Detail** section and expand down until it is about **2½"** in height.*
15. *Place your cursor in the left ruler. Your cursor will change to a right arrow.*
16. *Click in the ruler to the left of the **Employee_ID** label. Hold and drag down until the selection encompasses all labels and text boxes in the **Detail** section, then release.*

This is an easy way to select all of the objects within a form. Using the ruler to select objects will work on the top ruler as well. You will know all of the objects have been selected when each object is surrounded by a bold orange line.



Property Sheet

Selection type: Multiple selection

Visible	Yes
Width	
Height	
Top	
Left	
Back Style	
Back Color	Background 1
Border Style	
Border Width	Hairline
Border Color	
Special Effect	Flat
Font Name	Calibri (Detail)
Font Size	11
Text Align	General
Font Weight	Normal
Font Underline	No
Font Italic	No
Fore Color	

Figure 6.14

Create a Label

Now you need to move the selection of labels and text boxes over to the right and down a little. You can do this with your mouse or the arrow keys on your keyboard. Then you will create a new label.

17. *With your mouse, click and drag the selection until the upper-left corner of the **Employee_ ID** label is at approximately the ¾" mark on the left ruler and the **1"** mark on the top ruler.*

18. *Deselect the selection by clicking anywhere outside of the selection.*
19. *Click on the* **Label** *icon* Aa *in the* **Form Design Tools Design** *contextual tab.*
20. *With your mouse, draw a rectangular box from just below the* **Detail** *section bar at about the* **1½”** *line to about the* **3½”** *line on the top ruler, with a height of about half an inch.*
21. *Inside the label, type* ***Employee Input Form****.*
22. *Deselect the label by clicking outside of the label somewhere in the* **Design Grid**.

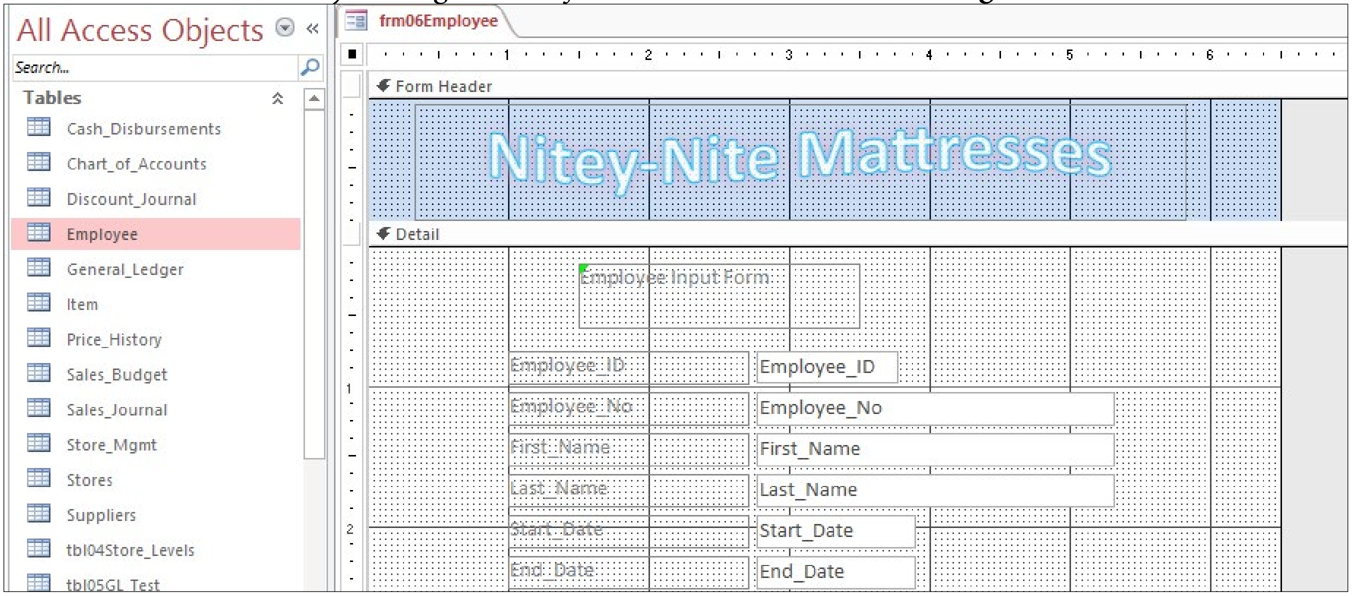
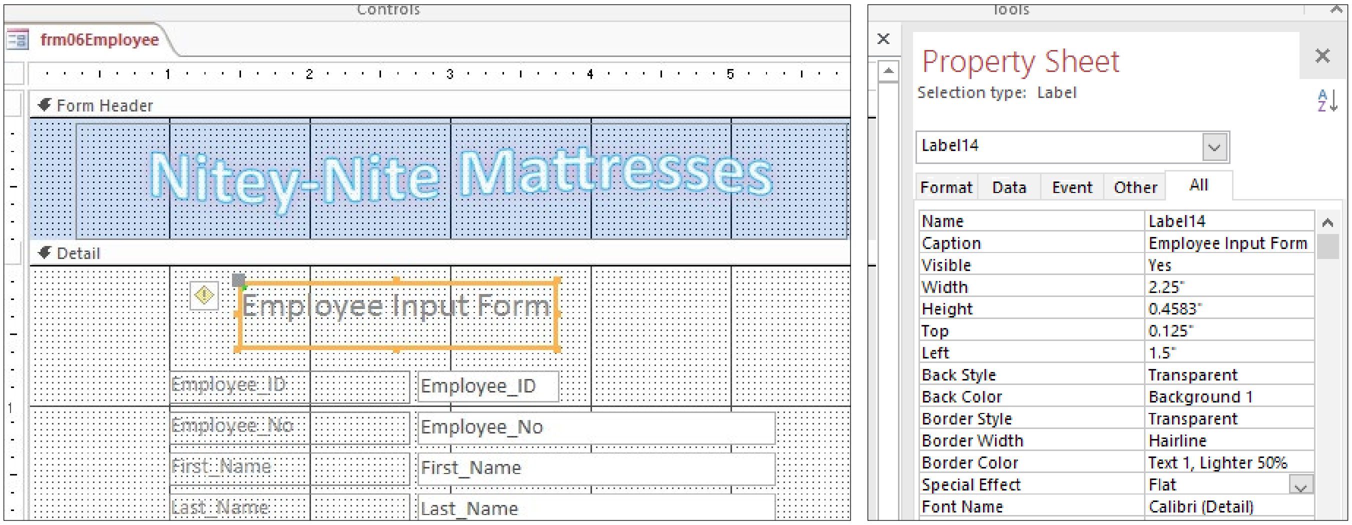


Figure 6.15

Now we need to format the Employee Input Form label.

23. *Select the* **Employee Input Form** *label by clicking once on the label.*
24. *Open the* **Property Sheet***, make sure the* **Font Name** *is* **Calibri (Detail)***, and change the* **Font Size** *to* **18***.*
25. *Resize the label to include all of the text and center it over the labels and text boxes.*



Property Sheet
Selection type: Label

Label14

Name	Label14
Caption	Employee Input Form
Visible	Yes
Width	2.25"
Height	0.4583"
Top	0.125"
Left	1.5"
Back Style	Transparent
Back Color	Background 1
Border Style	Transparent
Border Width	Hairline
Border Color	Text 1, Lighter 50%
Special Effect	Flat
Font Name	Calibri (Detail)

Figure 6.16

Now let's start working with the labels next to the text boxes. When you used the wizard to create the bound form, it assumed that the labels for each field would be the same as the corresponding field names in the Employee table. I like to show more common names for the fields. For example, the Employee_ID label doesn't need to have the underscore character in the label. We're not using the Employee_ID label for programming — it's just a label. So our next step is to rename each of the labels with names that are easier to read.

26. *Click on the* **Employee_ID** *label (not the text box) and change the text to "***Employee ID***".*
27. *Change all of the other labels as shown in* **Figure 6.17**.

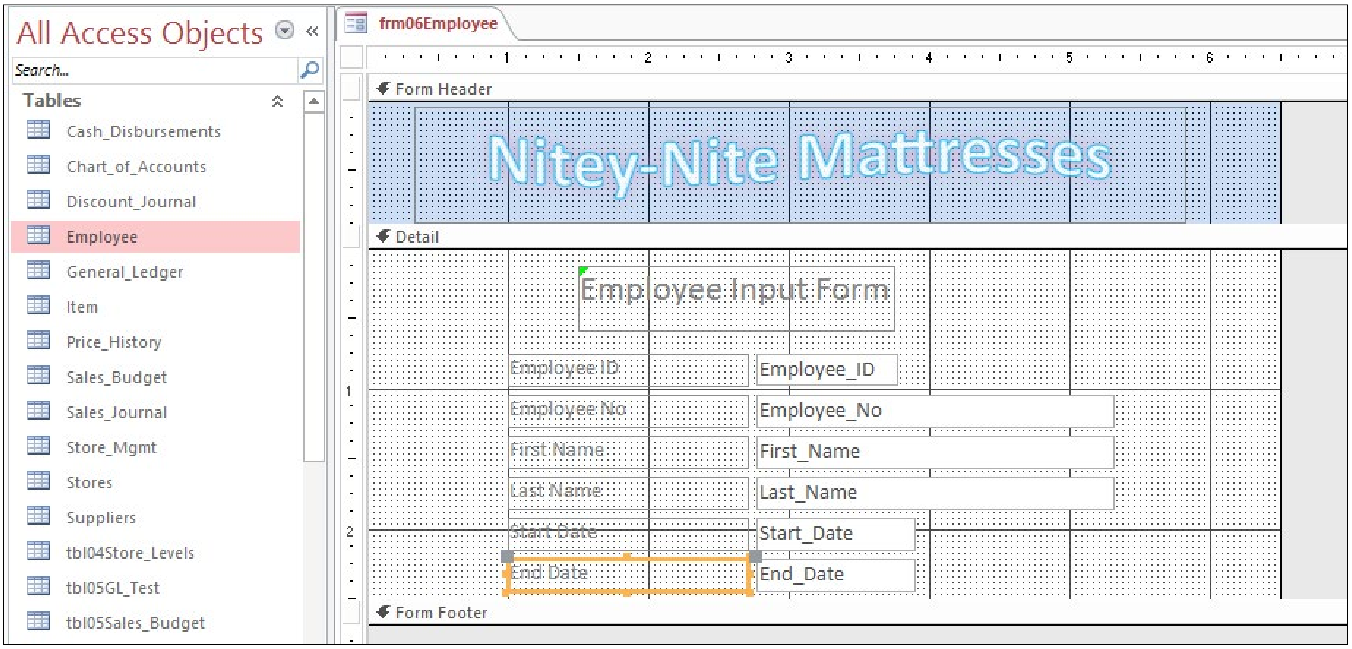
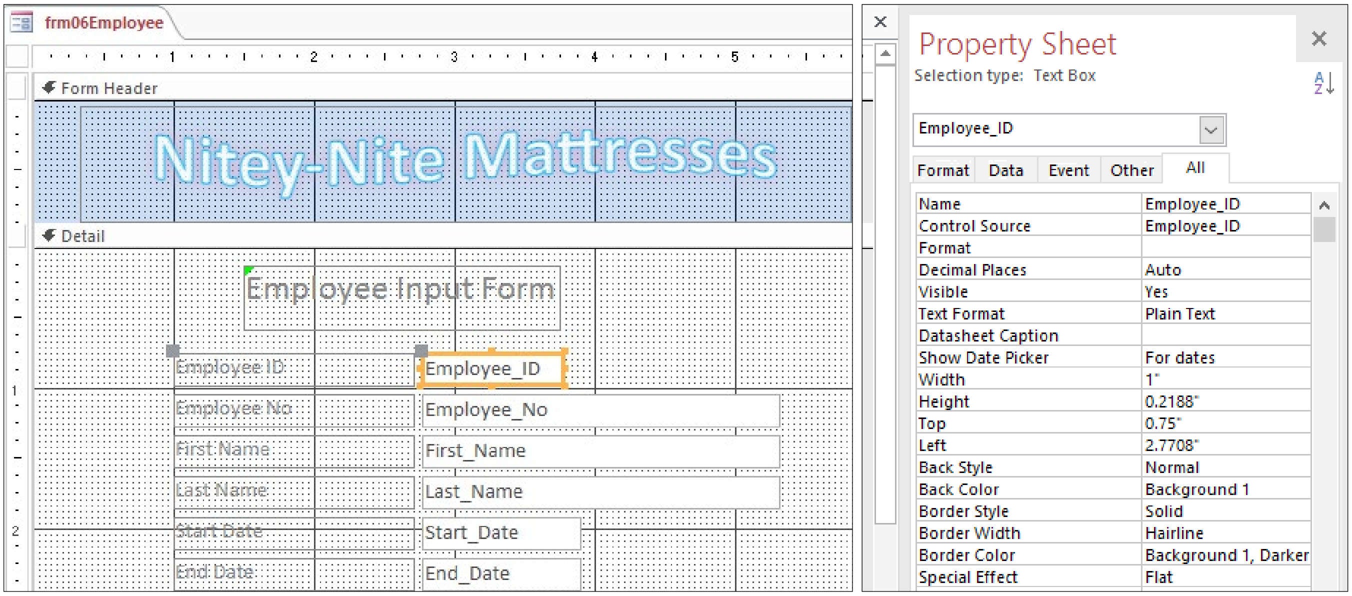


Figure 6.17

Now let's look at the text boxes. We've already discussed what each field is, but let's think about it from a user's perspective. The user will input data into each field in the form, except for the Employee_ID. The Employee_ID is an AutoNumber field, and therefore it can't be changed. It would be helpful to show it on the form, but only for informational purposes. So let's keep it on the form but not allow the user to access it with the tab key. We can do this by changing the object's Properties.

28. *Click on the* **Employee_ID** *text box and make sure the* **Property Sheet** *is displayed.*



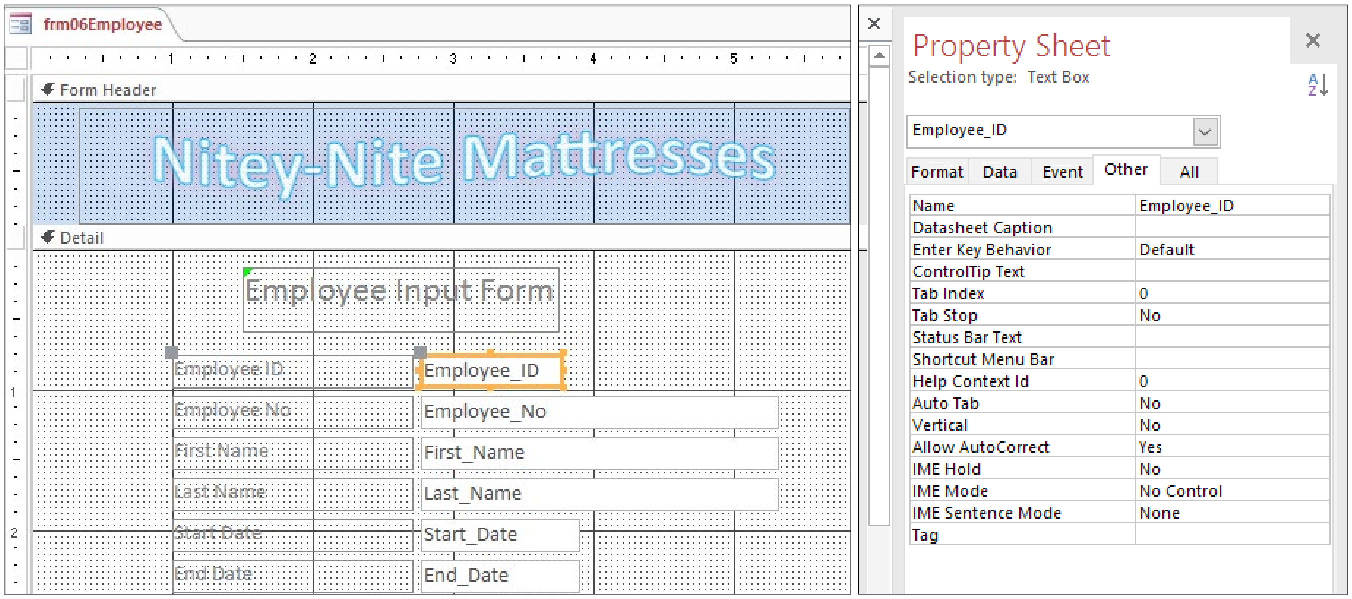
Selection type: Text Box

Employee_ID

Name	Employee_ID
Control Source	Employee_ID
Format	
Decimal Places	Auto
Visible	Yes
Text Format	Plain Text
Datasheet Caption	
Show Date Picker	For dates
Width	1"
Height	0.2188"
Top	0.75"
Left	2.7708"
Back Style	Normal
Back Color	Background 1
Border Style	Solid
Border Width	Hairline
Border Color	Background 1, Darker
Special Effect	Flat

Figure 6.18

29. *Click on the* **Other** *tab in the* **Property Sheet** *dialog box, scroll down to* **Tab Stop**, *and change it to* **No**.



Employee ID	Employee_ID
Employee No	Employee_No
First Name	First_Name
Last Name	Last_Name
Start Date	Start_Date
End Date	End_Date

Property Sheet
Selection type: Text Box

Employee_ID

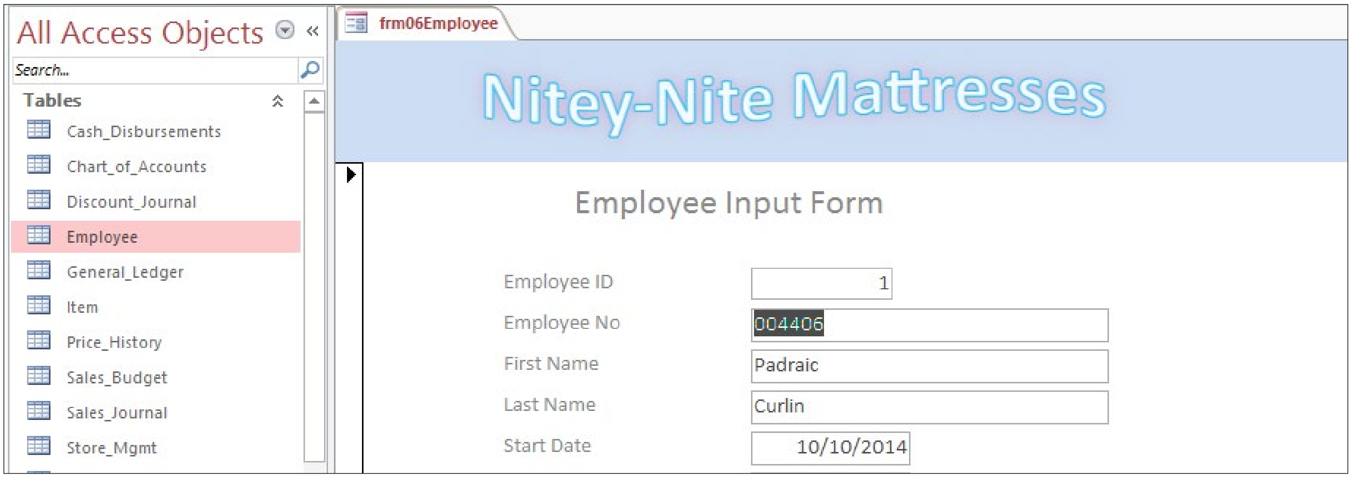
Name	Employee_ID
Datasheet Caption	
Enter Key Behavior	Default
ControlTip Text	
Tab Index	0
Tab Stop	No
Status Bar Text	
Shortcut Menu Bar	
Help Context Id	0
Auto Tab	No
Vertical	No
Allow AutoCorrect	Yes
IME Hold	No
IME Mode	No Control
IME Sentence Mode	None
Tag	

Figure 6.19

With the Tab Stop property set to No, that field is skipped over when the user uses the [Tab] key on the keyboard to move between text boxes.

There are many properties that are associated with each control. I encourage you to scroll through the list of properties to review what properties are available.

30. *Click the* **View** *icon to display the form in* **Form View***.*



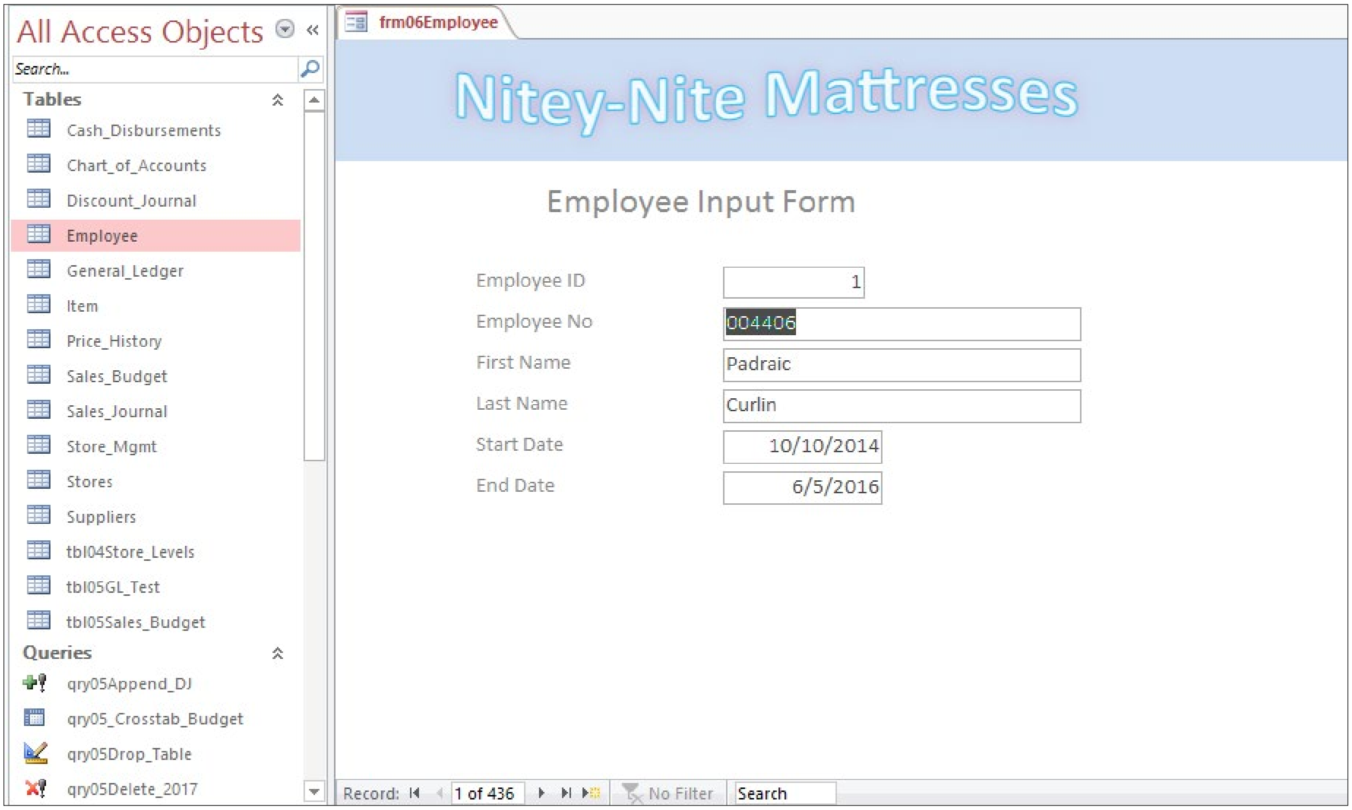
Employee Input Form

Employee ID	1
Employee No	004406
First Name	Padraic
Last Name	Curlin
Start Date	10/10/2014

Figure 6.20

It's getting to look more like the form in Figure 6.1, but we're not there yet. Notice the vertical gray bar on the left side of the form. This is called a ***Record Selector***. When you have a form that displays many records, the record selector is visible when you have a certain record in focus, or when it is chosen. In this form, we see only one record at a time. We really don't need the record selector, so we'll turn it off.

31. *Return to the* **Design View** *of the form.*
32. *To activate the properties for the entire form, click in the* **gray box** *at the upper-left corner of the Design View where the two rulers intersect.*
33. *In the* **Property Sheet** *for the form under the* **All** *tab, choose* **No** *for* **Record Selectors**.
34. *View the form in* **Form View**.



Nitey-Nite Mattresses

Employee Input Form

Employee ID	1
Employee No	004406
First Name	Padraic
Last Name	Curlin
Start Date	10/10/2014
End Date	6/5/2016

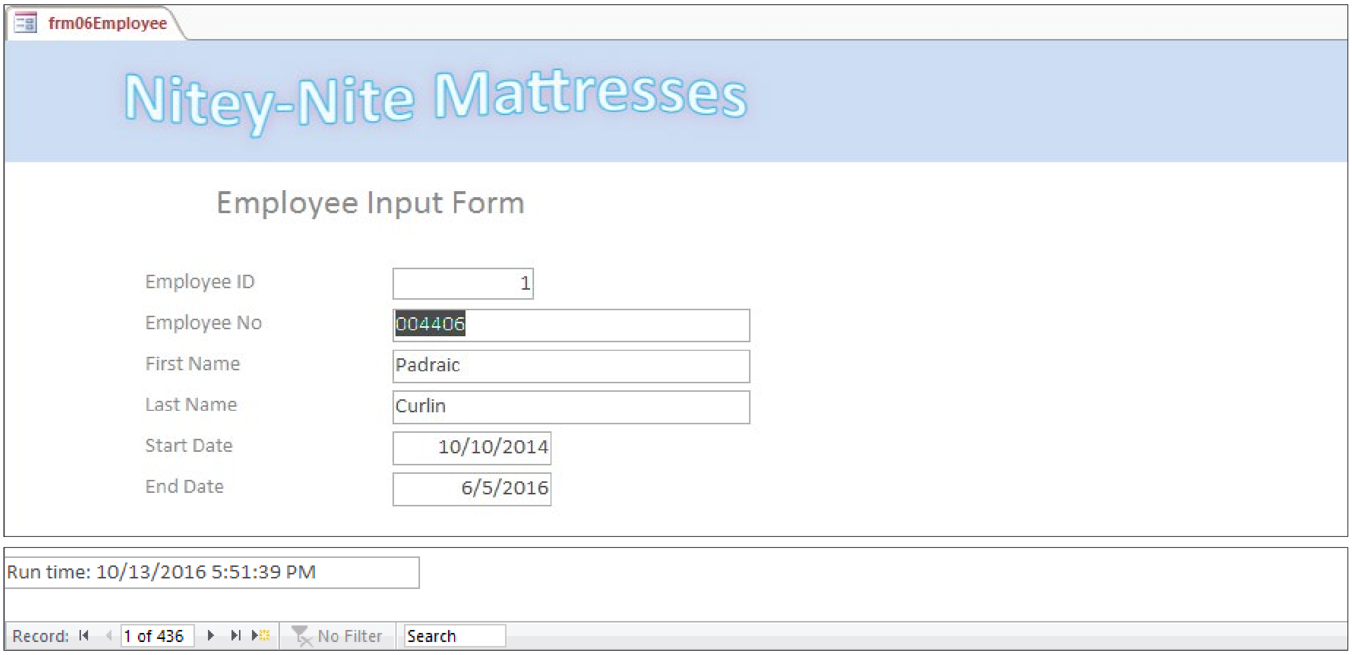
Figure 6.21

The next thing we need to do is to add in the Run Time box in the Form Footer. To do this, you will create a text box and write a formula in it that will display the date and time. The text box will not be tied to any field in the Employee table, and will show the date and time of the computer that's running the form.

35. *Go back to the* **Design View** *of the form.*
36. *Expand the* **Form Footer** *section where you have about half an inch of additional space.*
37. *Click the* **Text Box** *icon in the* **Controls** *group of the* **Form Design Tools Design** *tab and draw a rectangular box in the* **Form Footer** *section to expand across the center of the form.*
38. *Click once inside the new* **Unbound** *text box and type the following formula: =****”Run time: “ & Now()*** *and press* **[Enter]**.

This should look very familiar to you. All you're doing here is concatenating the text "Run time: " with the current date and time. When you created the text box, Access also created a label for that text box and called it something like Text21. You don't need that label, so you can delete it.

39. **Delete** *the* **label** *created in front of the* **Text box** ab|.
40. *Move the* **Text box** *over to the left side of the form.*
41. *View the form in* **Form View**.



Nitey-Nite Mattresses

Employee Input Form

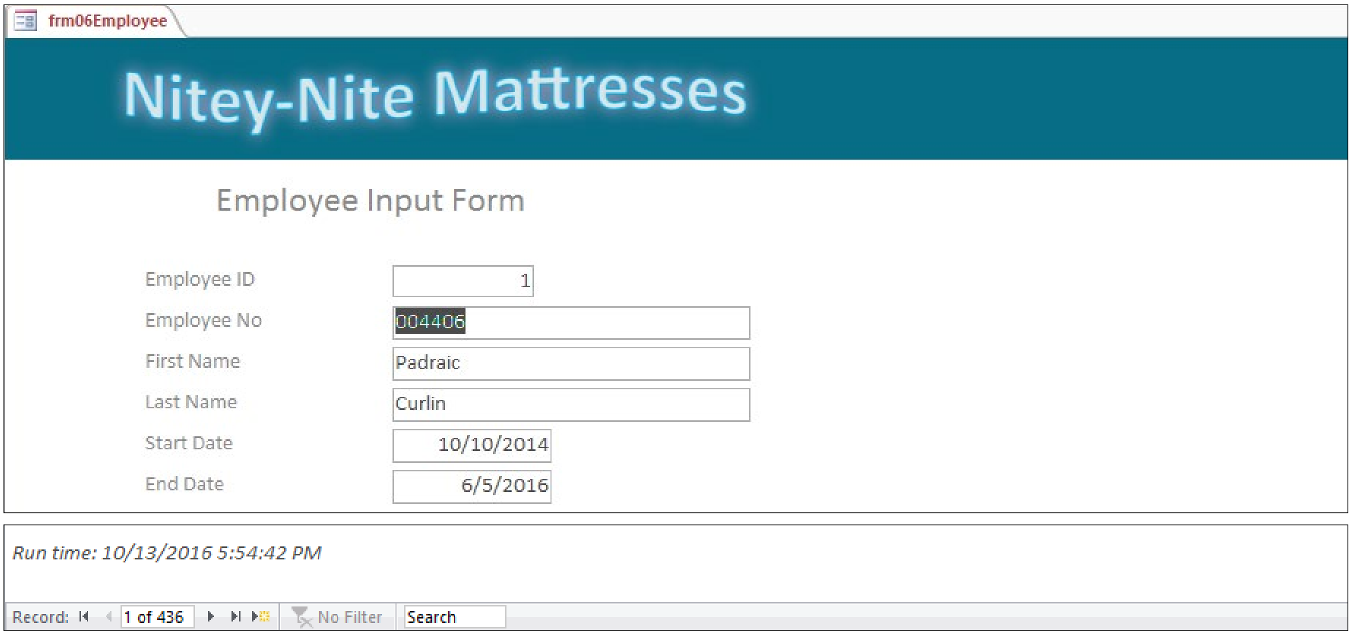
Employee ID	1
Employee No	004406
First Name	Padraic
Last Name	Curlin
Start Date	10/10/2014
End Date	6/5/2016

Run time: 10/13/2016 5:51:39 PM

Figure 6.22

The text box in the footer doesn't look quite yet like the one in Figure 6.1, but all we have to do is to make the text italics and remove the border around the text box. The Header color is not right yet, either.

42. *Return to the* **Design View** *of the form.*
43. *With the text box in the* **Form Footer** *selected, click on the* **Italics** *I icon in the* **Text Formatting** *group of the* **Home** *tab (the keyboard shortcut* **[Ctrl]+*i*** *also works).*
44. *Change the* **Border Style** *property in the* **Property Sheet** *to* **Transparent**.
45. *Make sure the* **Run time** *text box is wide enough to display all the results.*
46. *If you check the* **frmEmployee** *you started with, the* **Header** *color is darker. Change the* **Header Back Color** *to* ***#04617B*** *in the* **Property Sheet** *to match Figure 6.1.*
47. *View the form in* **Form View**.



Nitey-Nite Mattresses

Employee Input Form

Employee ID	1
Employee No	004406
First Name	Padraic
Last Name	Curlin
Start Date	10/10/2014
End Date	6/5/2016

Run time: 10/13/2016 5:54:42 PM

Figure 6.23

NOW it looks ready to show to your manager. When you show him the form, he really likes it. He even calls in the clerk who's going to be using it to show it to him. When the clerk looks at it, he says, "*You know, I usually enter new employee data in order of last name, first name, start date, end date, then employee number. And do I have to type in the End Date even though it's ALWAYS going to be 1/1/2099 when I'm entering a new employee?*"

I really like to get users involved in the design phase of my projects because sometimes they can come up with things like this which will help them out in the long run, and many times the changes are easy to do. Don't ever discount input by users.

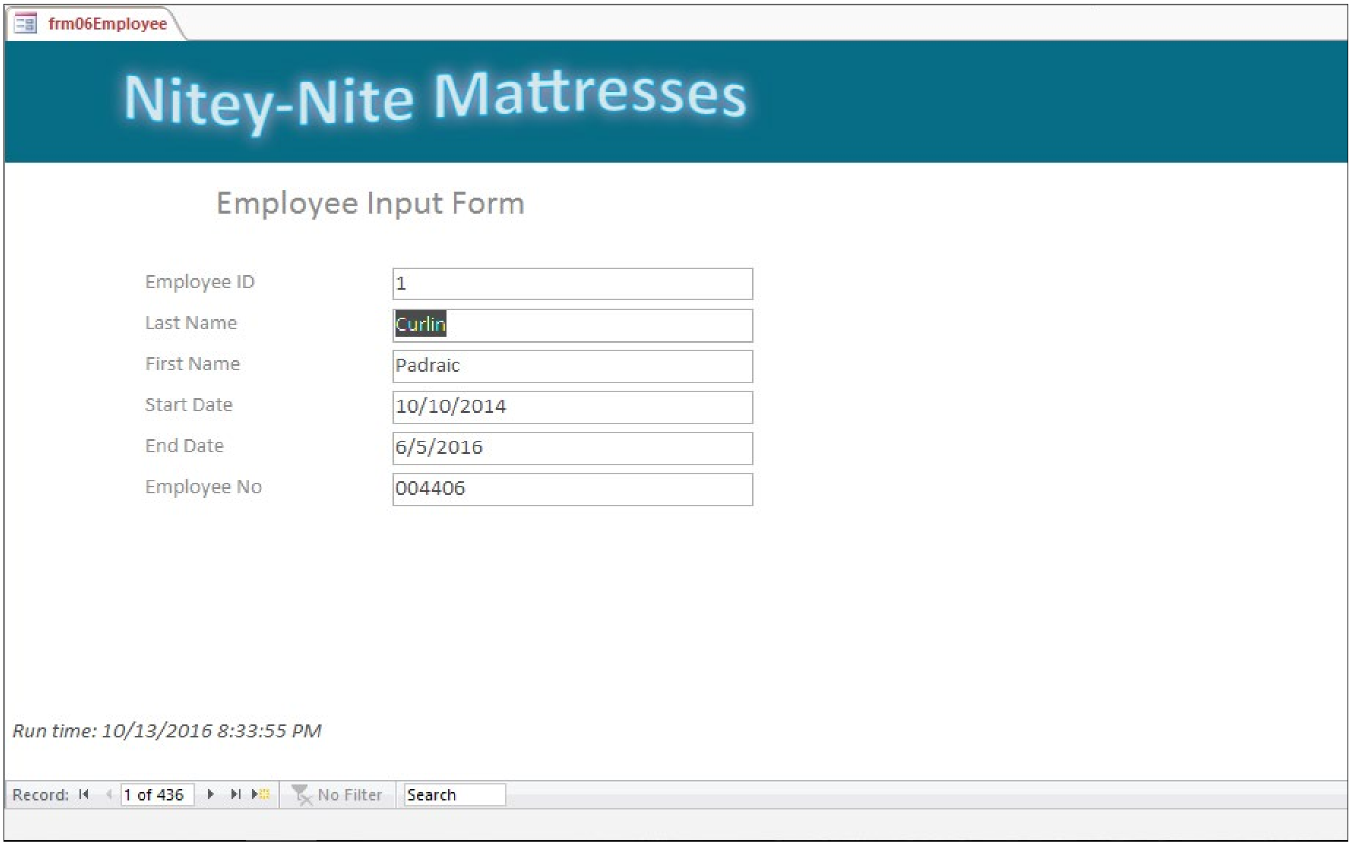
Text boxes are very easy to move around – just click and drag them to the place where you want them.

48. *Return to the* **Design View** *of the form.*
49. *Move the text boxes around in the order the clerk wanted.*
50. *In the* **Property Sheet** *dialog box, edit the properties of the* **End_Date** *text box to have a default value of* **1/1/2099** *and the* **Tab Stop** *value set to* **No***.*
51. *View the form in* **Form View** *and tab through a few records.*

You notice that as you tab through the records, the tabs move through the form in the order as the text boxes originally appeared. Also the tab stops on the Run time text box, which isn't necessary.

52. *Return to the* **Design View** *of the form, click on the* **Last Name** *text box, and click on the* **Other** *tab in the* **Property Sheet***.*
53. *Change the* **Tab Index** *to* **1** *and make sure the* **Tab Stop** *is set to* ***Yes****.*

54. *Change the tab indexes for the* **First Name** *to* **2**, **Start Date** *to* **3**, **End Date** *to* **4**, *and* **Employee No** *to* **5**, *with* **Tab Stop** = **Yes** *for each text box.*
55. *Change the* **Tab Stop** *for the* **Run time** *text box to* **No**.
56. *Click and drag to expand the bound text boxes to be the same width.*
57. *In the* **Format** *tab of the* **Property Sheet**, *change* **Text Align** *to* **Left** *for each text box (default is* **General**).
58. *View the form in* **Form View** *and make sure everything is working properly.*



Nitey-Nite Mattresses

Employee Input Form

Employee ID	1
Last Name	Curlin
First Name	Padraic
Start Date	10/10/2014
End Date	6/5/2016
Employee No	004406

Run time: 10/13/2016 8:33:55 PM

Figure 6.24

59. *Save and close the* **frm06Employee** *form.*

> ***Review Questions****: It is now time to complete the hands-on Review Questions. Log on to www.ExcelCEO.com with your Email and Password, click on the* ***Access 2016 and SQL Review Questions, Chapter 6, Section 2 of 2*** *option and complete the review questions.*

Conclusion

In this chapter, you learned about bound forms, or a form that is bound (tied) to a specific table or query. You looked at the Form, Layout, and Design views of an existing form and learned how to create a new bound form. You input new records into a table using a form as an interface. You learned how to use the rulers, Snap-to-Grid functionality, and used some of the icons available in the Office Ribbon. You created a form by using the wizard. You used and modified form properties and object properties. You inserted a graphic into a form as well as created labels and text boxes using the icons on the Office Ribbon. You worked with the Tab Order to be able to tab to each field in the correct order.

Chapter Exam

You can now go to www.ExcelCEO.com, click on Sign In, log in and take the exam. Make sure that you take the exam on the same device on which you completed the practice files, as some of the questions on the exam may refer to some of the completed projects. Chapter exams are intended to be hands-on.

CHAPTER SEVEN — INTERMEDIATE FORMS

In this chapter, you will:

- Identify and create an Unbound form
- Recognize Custom Groups
- Determine how and when to use use Subforms
- Identify how to use Startup Functionality

CPE Credits possible for this chapter: 2

Unbound Forms

The form in the previous chapter was an example of a bound form, or a form that is tied to a specific table or query. Sometimes I like to use unbound forms as a way to help users navigate to tables, queries, reports, or other forms in the database. An ***unbound form*** is simply a form that is not tied to a specific table or query. Many times, these types of forms are used as menus and navigational tools to help make the database easier to use. An example of an unbound form is a Main Menu. It is usually the first screen the user sees when the Access database is opened. We'll first set up an unbound form and then I'll show you some interesting tweaking you can do to make it more user-friendly. Let's get started.

1. *Open the* **Nitey_Nite_2016** *database.*
2. *In the* **Create** *tab, click on the* **Form Design** *icon in the* **Forms** *group.*

Access opens up a simple form in Design View. There is no table that is attached to this form. This is the form we will use as a Main Menu for the database.

3. **Save** *the form as* ***frm07MainMenu***.

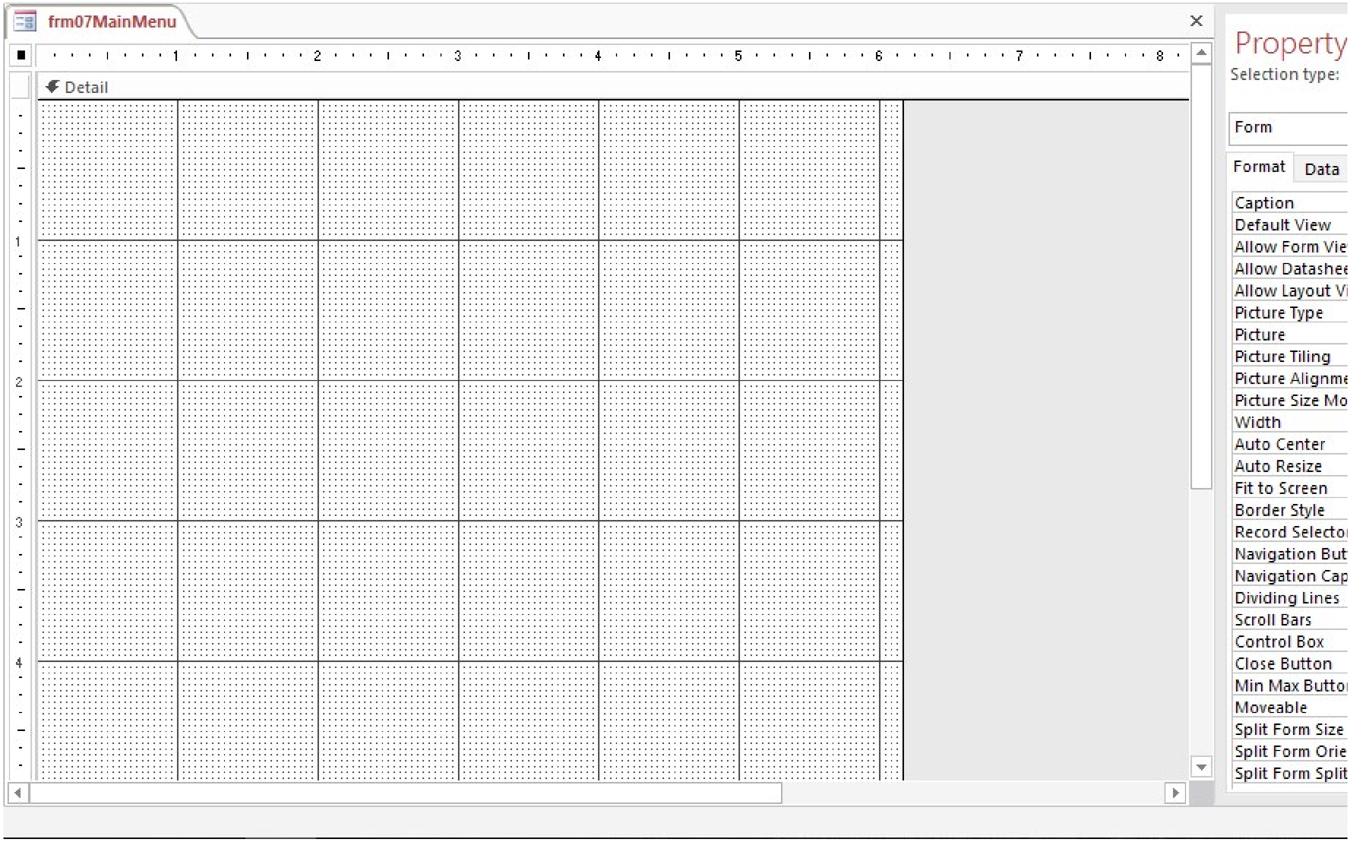


Figure 7.1

4. *View the form in* **Form View** *mode.*

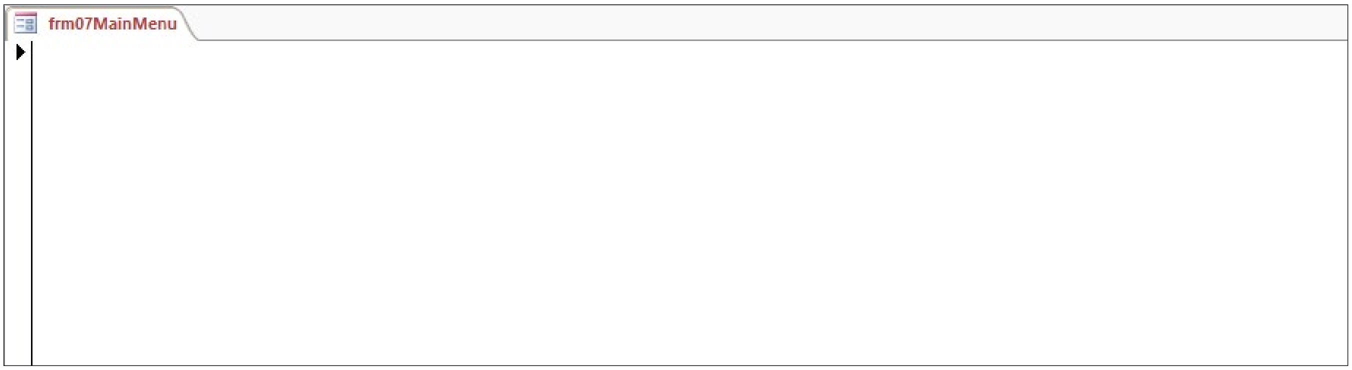


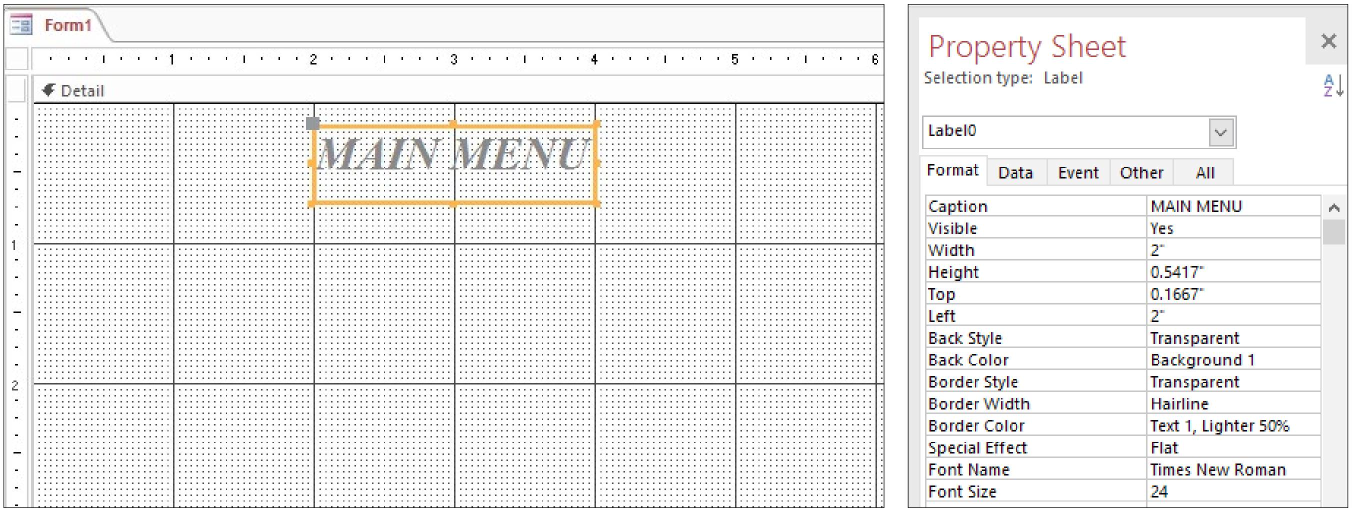
Figure 7.2

Not much to look at, is there? Not to worry – we'll make it more useful. The first thing we should do is to take out the record selector, just like we did in the first form we created.

5. *Return to the* **Design View** *of the form.*
6. *Make sure the form is selected (click in the gray box at the intersection where the top and left rulers meet).*
7. *On the* **Format** *tab of the* **Property Sheet**, *set the* **Record Selectors** *property to* **No**.

Now you have a completely blank form. Let's create a label that says "MAIN MENU".

8. *Click on the* **Label** *icon in the* **Controls** *group of the* **Form Design Tools Design** *contextual tab.*
9. *Create a label at the top-center portion of the form called* ***MAIN MENU***.
10. *In the* **Property Sheet**, *change the* **Font Name** *to* **Times New Roman** *and the* **Font Size** *to* ***24***, *making the* **Font Weight Bold** *and changing* **Font Italic** *to* **Yes**.



Selection type: Label

Label0

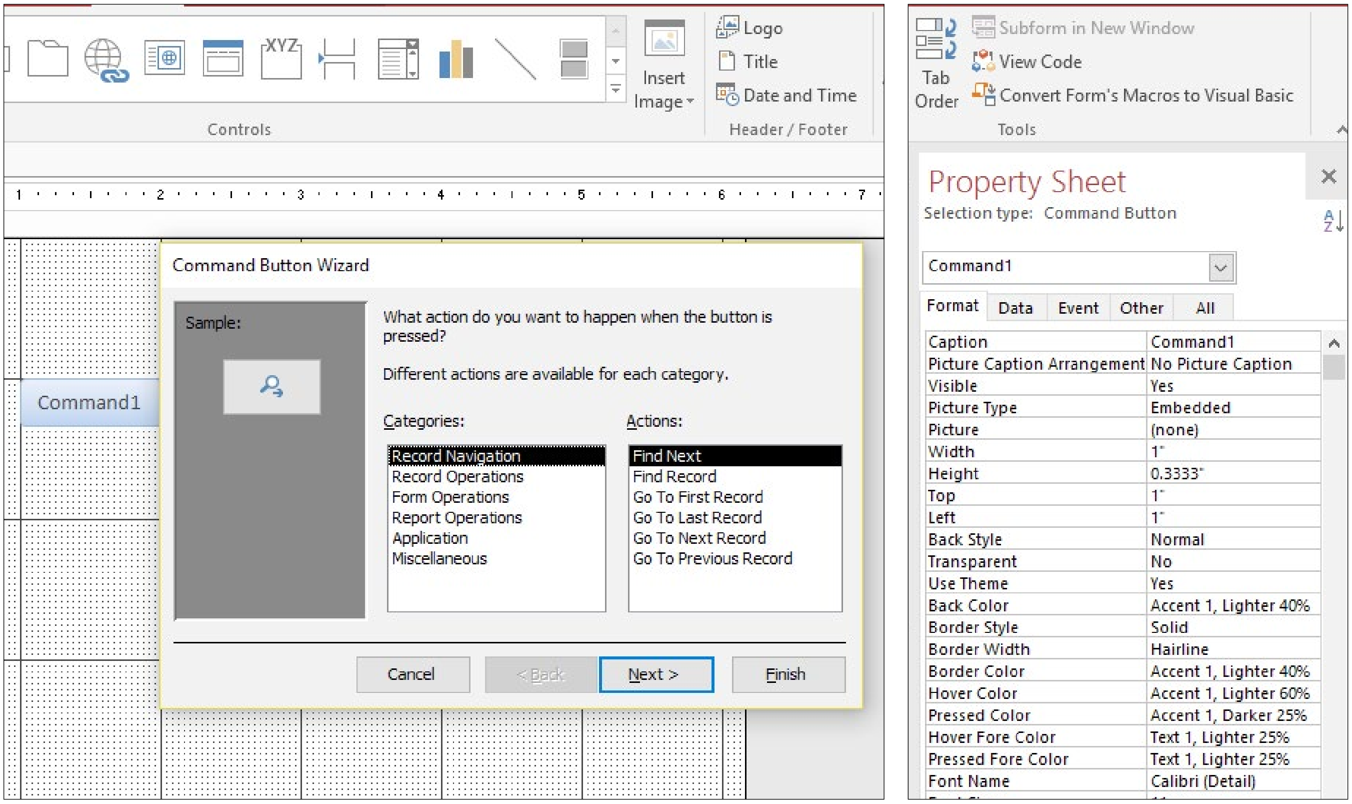
Caption	MAIN MENU
Visible	Yes
Width	2"
Height	0.5417"
Top	0.1667"
Left	2"
Back Style	Transparent
Back Color	Background 1
Border Style	Transparent
Border Width	Hairline
Border Color	Text 1, Lighter 50%
Special Effect	Flat
Font Name	Times New Roman
Font Size	24

Figure 7.3

Command Buttons

Next we will add a Command Button to navigate the user to the Employee Input Form. A Command Button in Access is a graphical tool that initiates an action when you click it. When you begin to create a Command Button in Access, a wizard pops up to guide you through the programming assumptions.

11. *Click on the* **Button** XXXX *icon in the* **Controls** *group of the* **Form Design Tools Design** *tab and draw a rectangle on the left side of the form below the* **MAIN MENU** *label.*



Command Button Wizard

Sample:

What action do you want to happen when the button is pressed?

Different actions are available for each category.

Categories:
Record Navigation
Record Operations
Form Operations
Report Operations
Application
Miscellaneous

Actions:
Find Next
Find Record
Go To First Record
Go To Last Record
Go To Next Record
Go To Previous Record

Cancel | < Back | Next > | Finish

Property Sheet

Selection type: Command Button

Command1

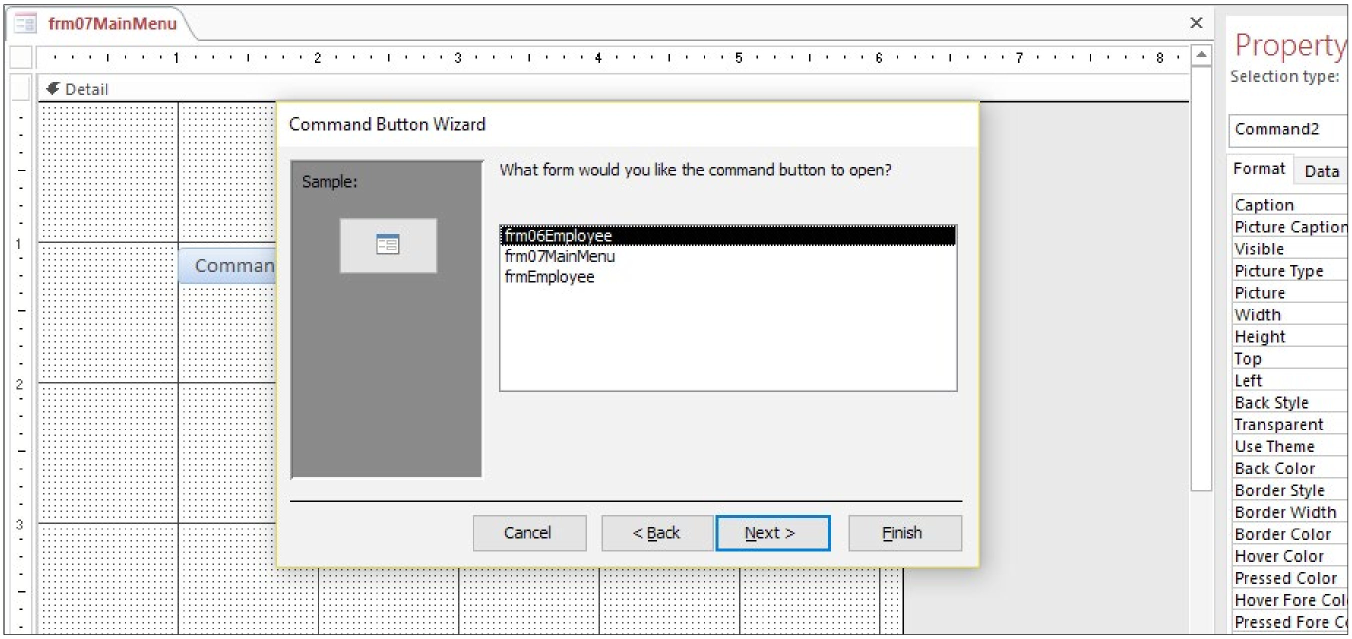
Format | Data | Event | Other | All

Caption	Command1
Picture Caption Arrangement	No Picture Caption
Visible	Yes
Picture Type	Embedded
Picture	(none)
Width	1"
Height	0.3333"
Top	1"
Left	1"
Back Style	Normal
Transparent	No
Use Theme	Yes
Back Color	Accent 1, Lighter 40%
Border Style	Solid
Border Width	Hairline
Border Color	Accent 1, Lighter 40%
Hover Color	Accent 1, Lighter 60%
Pressed Color	Accent 1, Darker 25%
Hover Fore Color	Text 1, Lighter 25%
Pressed Fore Color	Text 1, Lighter 25%
Font Name	Calibri (Detail)

Figure 7.4

When you release the mouse, the Command Button Wizard opens up. In this wizard, you will set the properties for the command button. All we want this button to do is to navigate to the Employee Input Form, or in other words, open frm06Employee.

12. *In the* **Command Button Wizard**, *click on* **Form Operations** *in the* **Categories** *section and* **Open Form** *in the* **Actions** *section, and click* **Next** >.



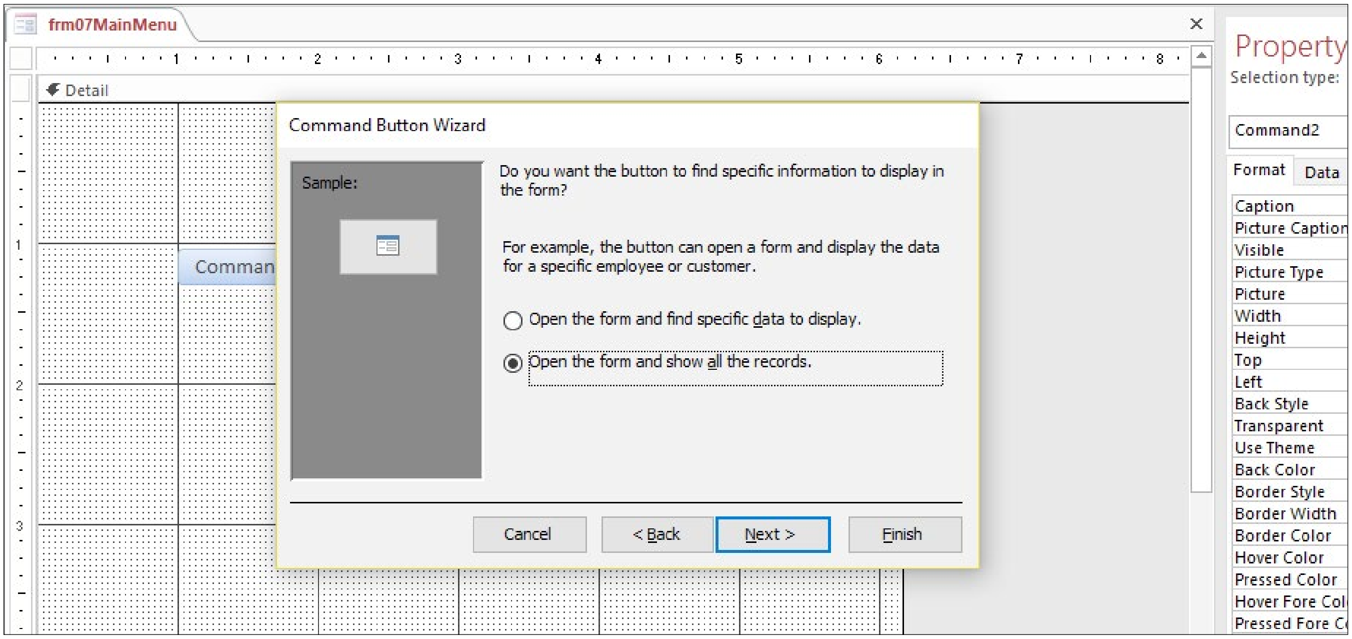
Command Button Wizard

What form would you like the command button to open?

- frm06Employee
- frm07MainMenu
- frmEmployee

Figure 7.5

13. *In the next step of the wizard, make sure* **frm06Employee** *is selected, and click* **Next >**.



Command Button Wizard

Do you want the button to find specific information to display in the form?

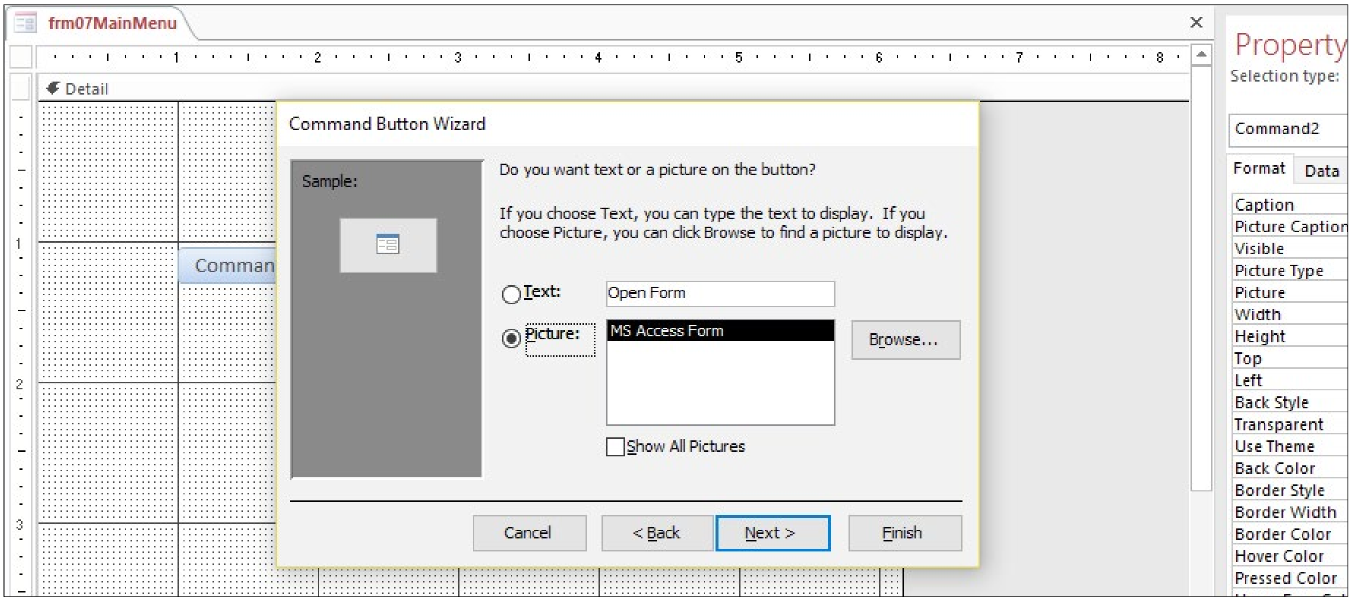
For example, the button can open a form and display the data for a specific employee or customer.

○ Open the form and find specific data to display.

● Open the form and show all the records.

Figure 7.6

14. *Even though* **frm06Employee** *displays only one record at a time, we don't want to find a specific record, so leave the default value "***Open the form and show all the records***" option selected and click* **Next >**.



Command Button Wizard

Do you want text or a picture on the button?

If you choose Text, you can type the text to display. If you choose Picture, you can click Browse to find a picture to display.

○ Text: Open Form

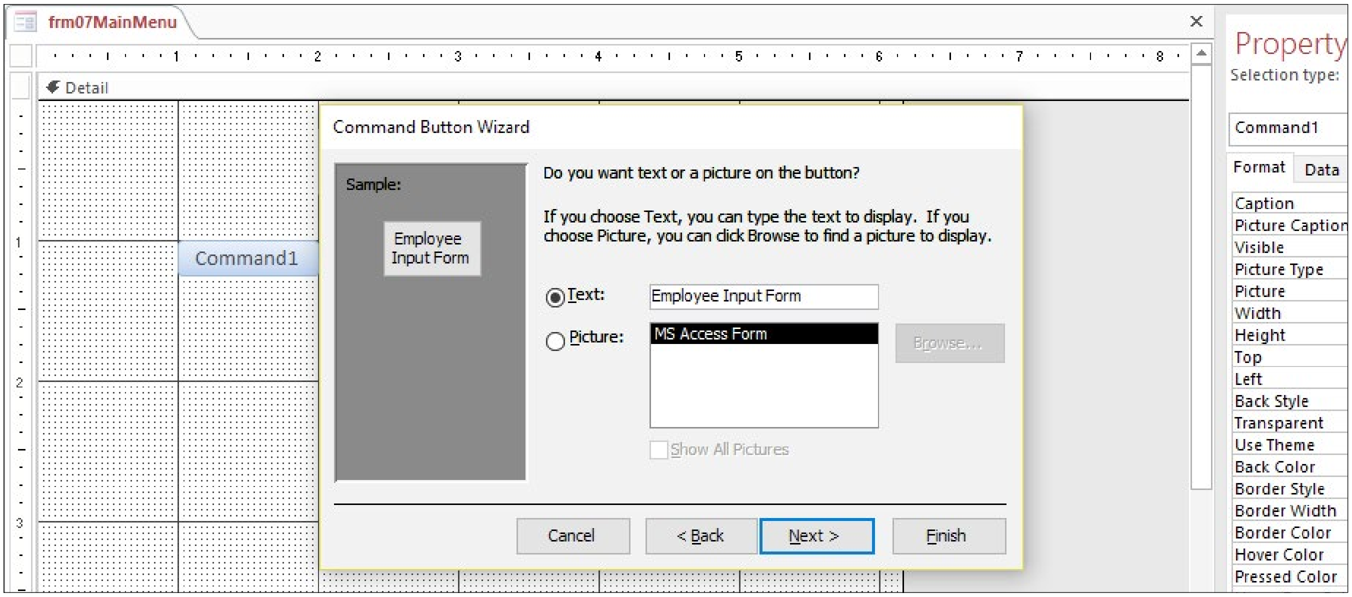
● Picture: MS Access Form

□ Show All Pictures

Figure 7.7

In the next step of the form, you choose whether to type text or to put a picture in the command button. One time I was developing a Delete Record command button and I put on it a picture of a toilet. I also imported a wave file that made the sound of a toilet flushing when the user clicked the button. It was a lot of fun to create, but since the database would be used in front of clients, we decided against that idea. We'll just use the text option in our example.

15. *Click on the* **Text:** *option and type* ***Employee Input Form****, then click* **Next >***.*



Command Button Wizard

Do you want text or a picture on the button?

If you choose Text, you can type the text to display. If you choose Picture, you can click Browse to find a picture to display.

● Text: Employee Input Form

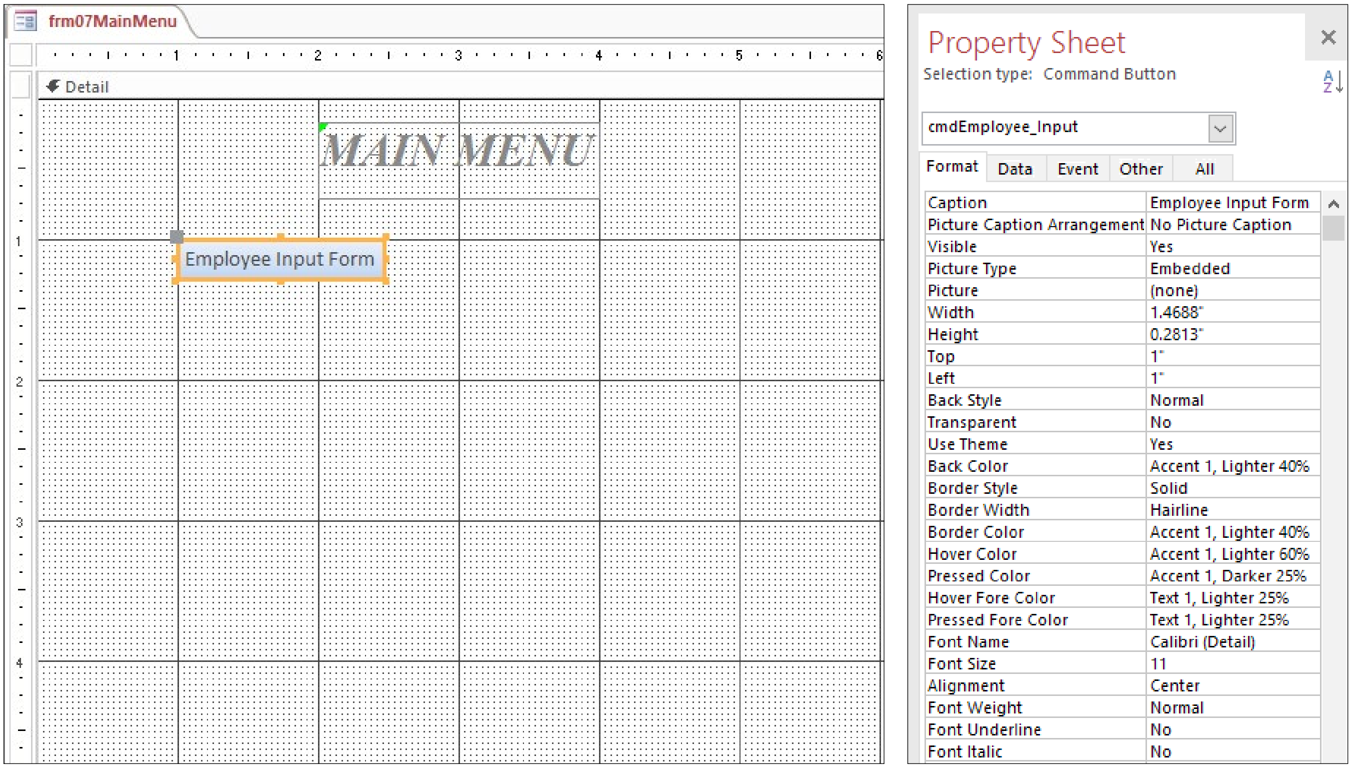
○ Picture: MS Access Form

□ Show All Pictures

Figure 7.8

The last step of the wizard asks you to name the button. You can name it anything you want, but it would be helpful to name it with a logical name just in case you need to refer to it in some of your programming.

16. *Name the command* **Button** ***cmdEmployee_Input*** *and click* **Finish**.



Property Sheet

Selection type: Command Button

cmdEmployee_Input

Format | Data | Event | Other | All

Caption	Employee Input Form
Picture Caption Arrangement	No Picture Caption
Visible	Yes
Picture Type	Embedded
Picture	(none)
Width	1.4688"
Height	0.2813"
Top	1"
Left	1"
Back Style	Normal
Transparent	No
Use Theme	Yes
Back Color	Accent 1, Lighter 40%
Border Style	Solid
Border Width	Hairline
Border Color	Accent 1, Lighter 40%
Hover Color	Accent 1, Lighter 60%
Pressed Color	Accent 1, Darker 25%
Hover Fore Color	Text 1, Lighter 25%
Pressed Fore Color	Text 1, Lighter 25%
Font Name	Calibri (Detail)
Font Size	11
Alignment	Center
Font Weight	Normal
Font Underline	No
Font Italic	No

Figure 7.9

17. *Open the form in* **Form View** *and click the* **Employee Input Form** *command* **Button**.

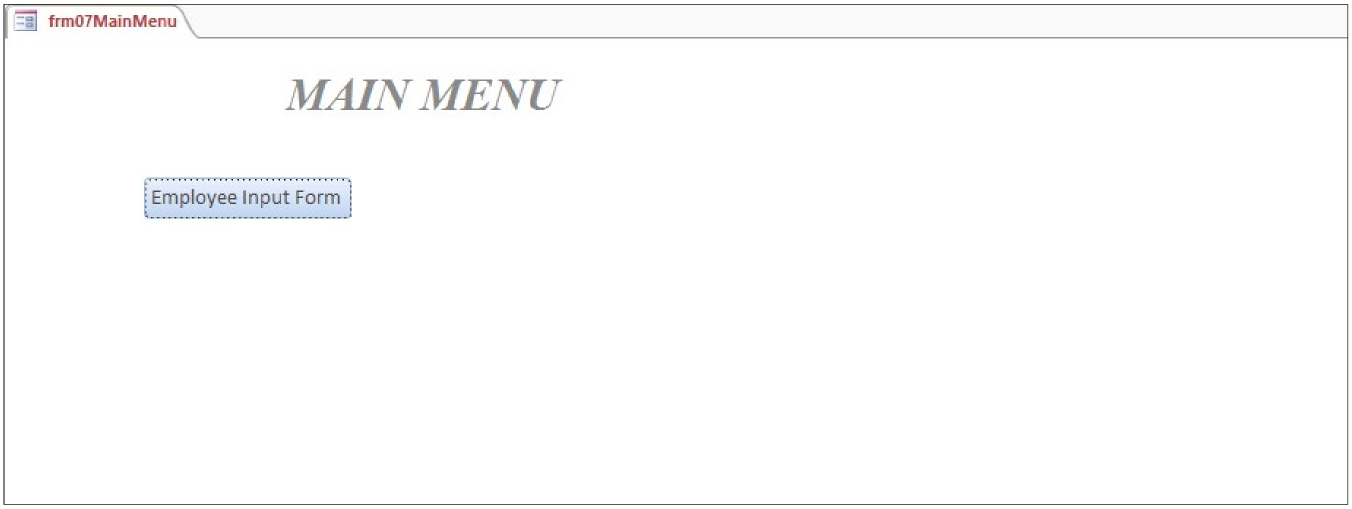
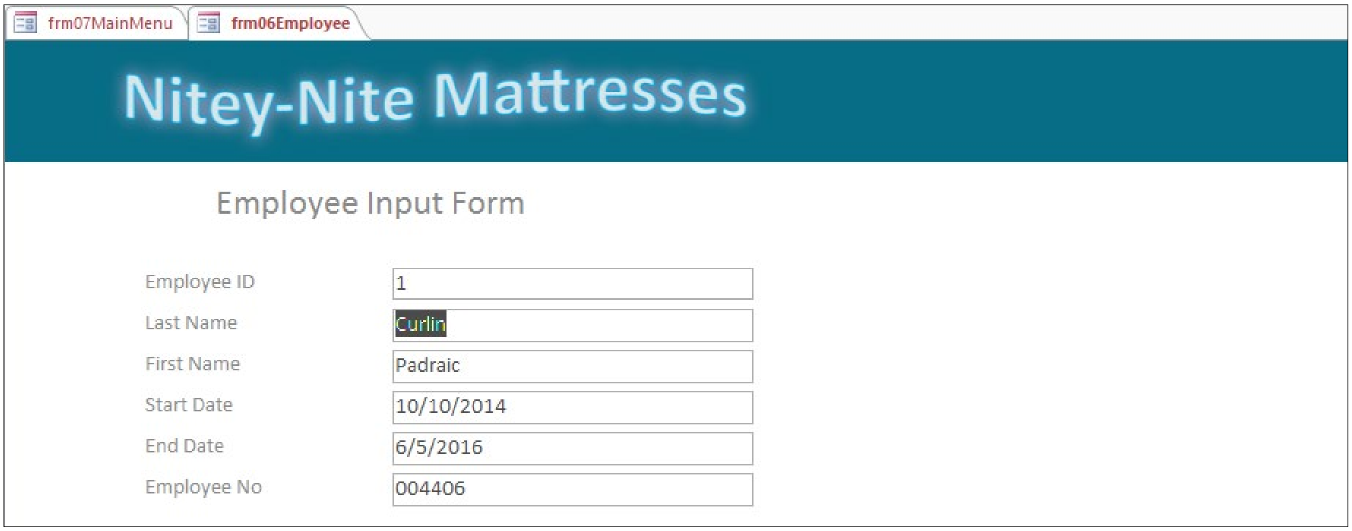


Figure 7.10

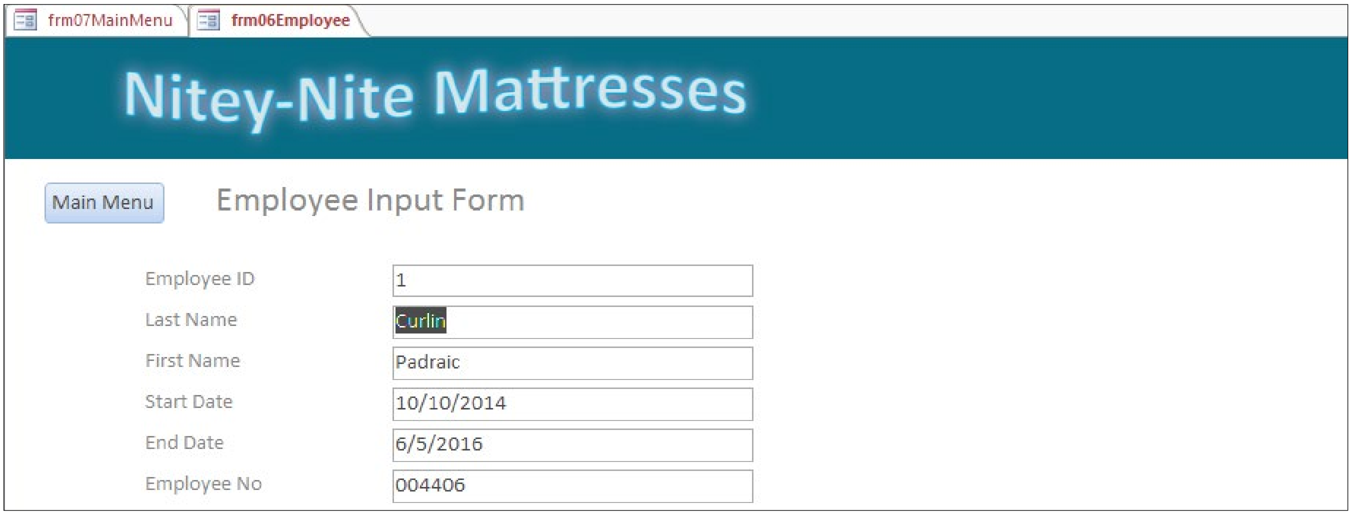


Employee ID	1
Last Name	Curlin
First Name	Padraic
Start Date	10/10/2014
End Date	6/5/2016
Employee No	004406

Figure 7.11

The Employee Input Form opens up. Wouldn't it be great if you had a similar type of button on the Employee Input Form to send the user back to the Main Menu? You should now be able to do that one on your own.

18. *Go to the* **Design View** *of the* **Employee Input Form** *and create a command* **Button** *with a caption of* **Main Menu** *that sends the user back to the* **frm07MainMenu**. *Place this command button to the left of the* **Employee Input Form** *label.*
19. **Save** *the changes to* **frm06Employee**, *then view the form in* **Form View**.



Employee ID	1
Last Name	Curlin
First Name	Padraic
Start Date	10/10/2014
End Date	6/5/2016
Employee No	004406

Figure 7.12

While you are clicking on these command buttons and seeing how cool it is to go back and forth between forms, you may notice that each form stays open after you go to the other form. This is illustrated by the frm07MainMenu and frm06Employee tabs at the top of the form. Frm06Employee is displayed with a white background, meaning it is the one currently displayed. I like to have only one form open at a time. To do this, you need to close a form before opening a new one, something we were unable to program in the Command Button Wizard. Let's add that functionality. We'll do that by creating a macro.

20. *Go to the* **Design View** *of* **frm06Employee**.
21. *Click on the* **Main Menu** *button and view the* **Property Sheet** *for the button.*
22. *Click on the* **Event** *tab of the* **Property Sheet**.

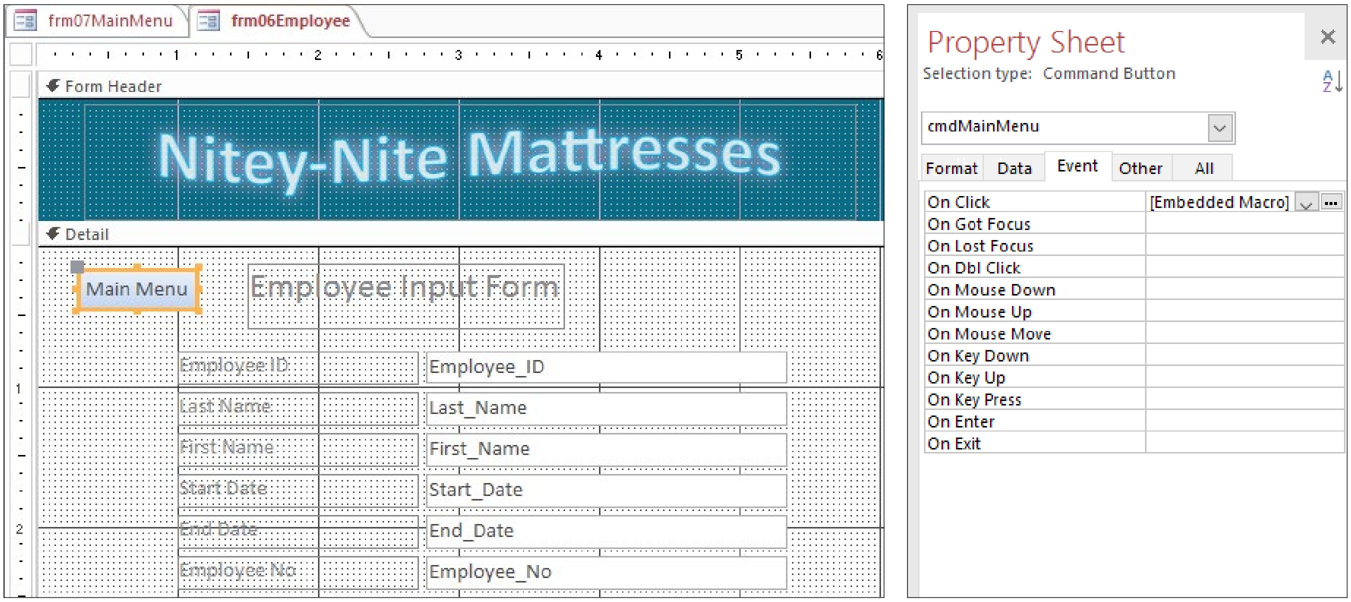
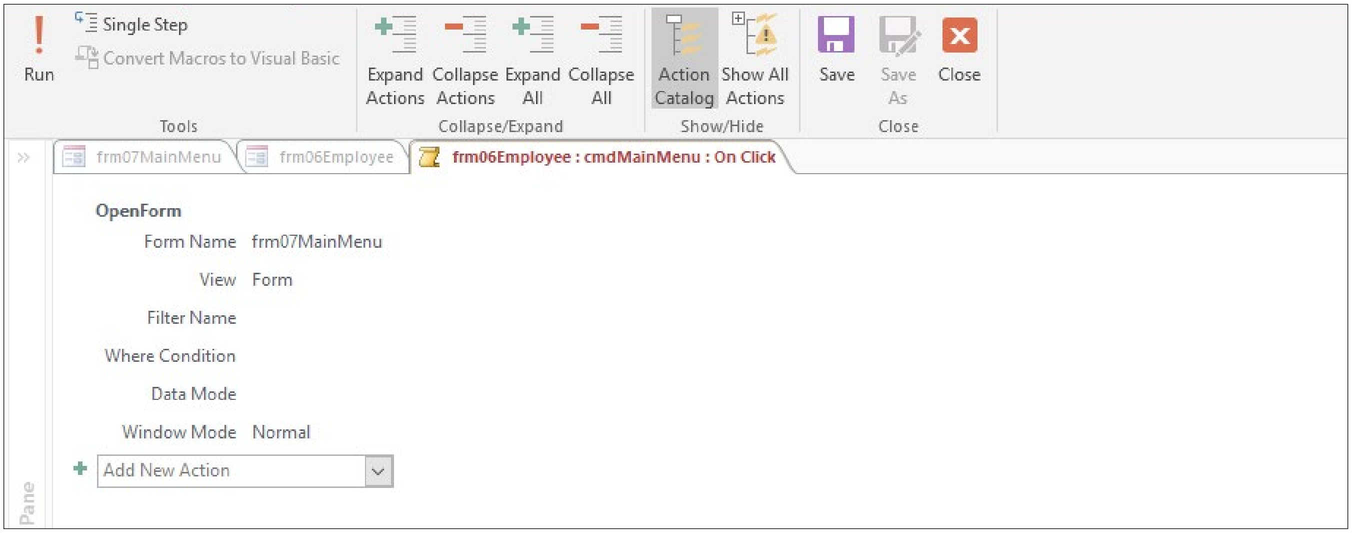


Figure 7.13

23. *Click on the ellipses* ⌄ ··· *button (also called a* **Build** *button) to the right of the* **On Click** *property.*

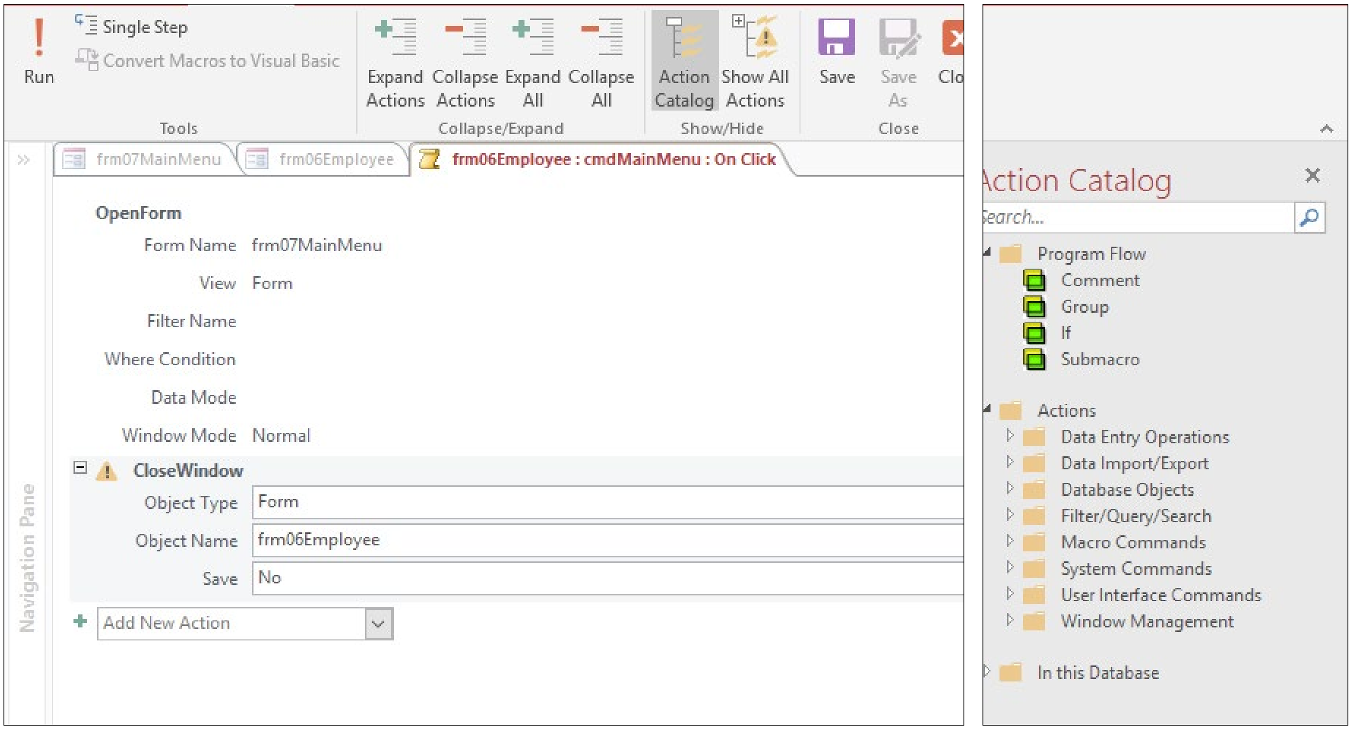


OpenForm
Form Name frm07MainMenu
View Form
Filter Name
Where Condition
Data Mode
Window Mode Normal

Figure 7.14

After you click on the build button, you should get a screen that looks like Figure 7.14. This is the screen where you can add to the macro that the Command Button Wizard automatically created to open frm07MainMenu. Currently, the On Click property simply opens the MAIN MENU form. We want for it to also close frm06Employee. That is done by using the Close action. Let's try it.

24. *Click in the* **Add New Action** *box under the* **OpenForm** *action and choose* **CloseWindow**.
25. *In the* **Close Window** *box on the* **Object Type** *line, choose* **Form**.
26. *On the* **Object Name** *line, choose* **frm06Employee**.
27. *On the* **Save** *line, choose* **No**.



OpenForm
Form Name frm07MainMenu
View Form
Filter Name
Where Condition
Data Mode
Window Mode Normal

CloseWindow
Object Type Form
Object Name frm06Employee
Save No

Figure 7.15

The button will now open the MAIN MENU form, then close the frm06Employee form.

28. *Close the macro window by clicking on the* **Close** *button in the* **Office Ribbon**.
29. *Click* **Yes** *when it asks if you want to save the changes made to the macro and update the property. If it doesn't prompt you,* **save** *the form anyway.*
30. **Save** *the* **frm06Employee** *and* **frm07MainMenu** *forms.*

The form frm06Employee opens up in Design View. Now you'll test the button to see if it works.

31. *Go to the* **Form View** *of the* **frm06Employee** *form and click on the* **Main Menu** *button.*
32. *If asked if you want to save the changes to the design of form* **frm06Employee**, *click* **Yes**.

The Main Menu form then opens up, and you will notice that the form frm06Employee tab no longer appears.

33. *Click on the* **Employee Input Form** *button on the* **Main Menu** *form.*

Now you see that the frm06Employee opens up, but the frm07MainMenu tab still shows up. All you have to do now is to edit the On Click property of the Employee Input Form command button to close the MAIN MENU form (frm07MainMenu) before the frm06Employee form opens up.

34. *Modify the* **On Click** *property of the* **Employee Input Form** *command button in* **frm07MainMenu** *to close the form before* **frm06Employee** *opens up.*
35. *Save the form.*
36. *Test the functionality of both command buttons to make sure they work as expected.*
37. *Save and close* **frm07MainMenu** *and* **frm06Employee**.

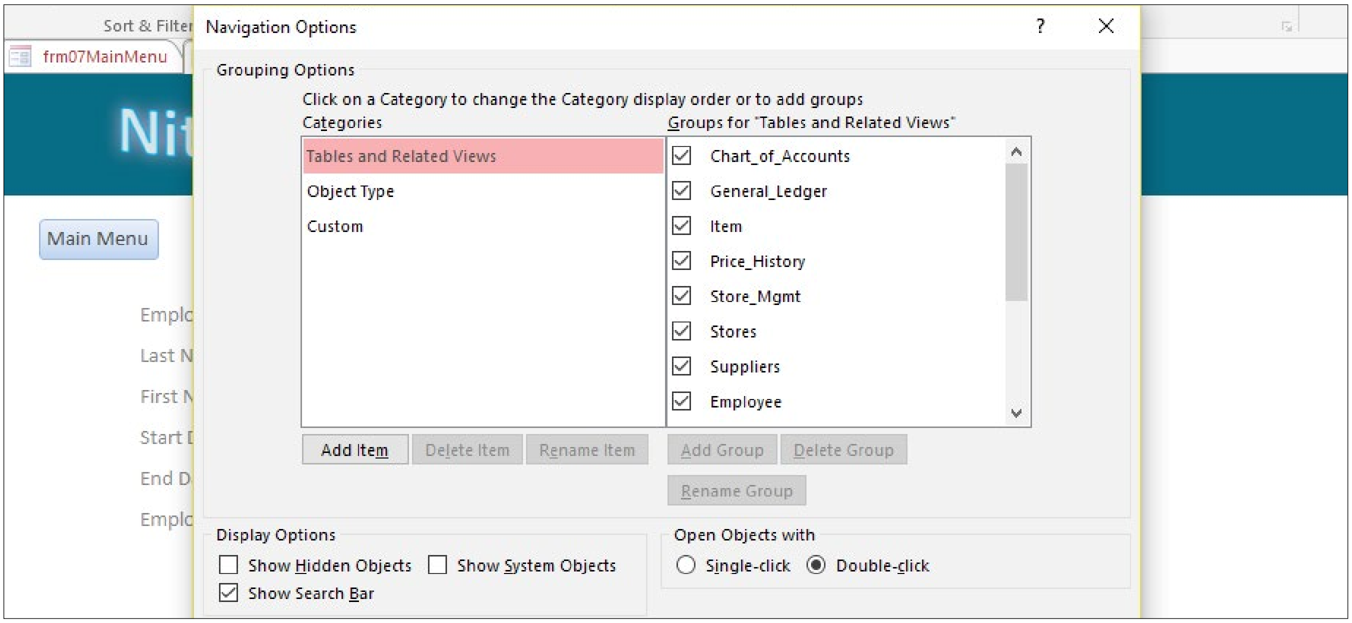
> ***Review Questions****: It is now time to complete the hands-on Review Questions. Log on to www.ExcelCEO.com with your Email and Password, click on the* **Access 2016 and SQL Review Questions, Chapter 7, Section 1 of 2** *option and complete the review questions.*

Creating Custom Categories

In the past, Access has used the concept of a "switchboard" to navigate around a database. The switchboard was being phased out in Access 2010 and totally eliminated since Access 2013 and Microsoft has replaced it with ***Custom Categories***. One reason is that custom categories contain almost all of the functionality of switchboards, with much less complexity to set them up. Access switchboards are a thing of the past, so this course will not explore them. We will now introduce you to using Custom Categories.

You have already used the Navigation Pane on the left side of the database. In Chapter 1, you changed the category heading to read "All Access Objects" instead of "All Tables". The smaller bars beneath the "All Access Objects" heading are groups, and "All Access Objects" is the category. There are a number of features that allow you to change the categories and groups, and that is what we will explore in this exercise. You can have up to ten custom categories and each category can contain multiple groups. Once a group is created, you can drag and drop Access objects (like tables, queries, forms, reports, and macros) into each group. The objects will remain in their original categories (table, queries, etc.), and the custom categories serve as shortcuts to those objects. Let's create a custom category.

1. *At the top of the* **Navigation Pane** *on the left side of the database, right-click on* **All Access Objects** *and choose* **Navigation Options…**



Navigation Options

Grouping Options

Click on a Category to change the Category display order or to add groups

Categories	Groups for "Tables and Related Views"
Tables and Related Views	☑ Chart_of_Accounts
Object Type	☑ General_Ledger
Custom	☑ Item
	☑ Price_History
	☑ Store_Mgmt
	☑ Stores
	☑ Suppliers
	☑ Employee

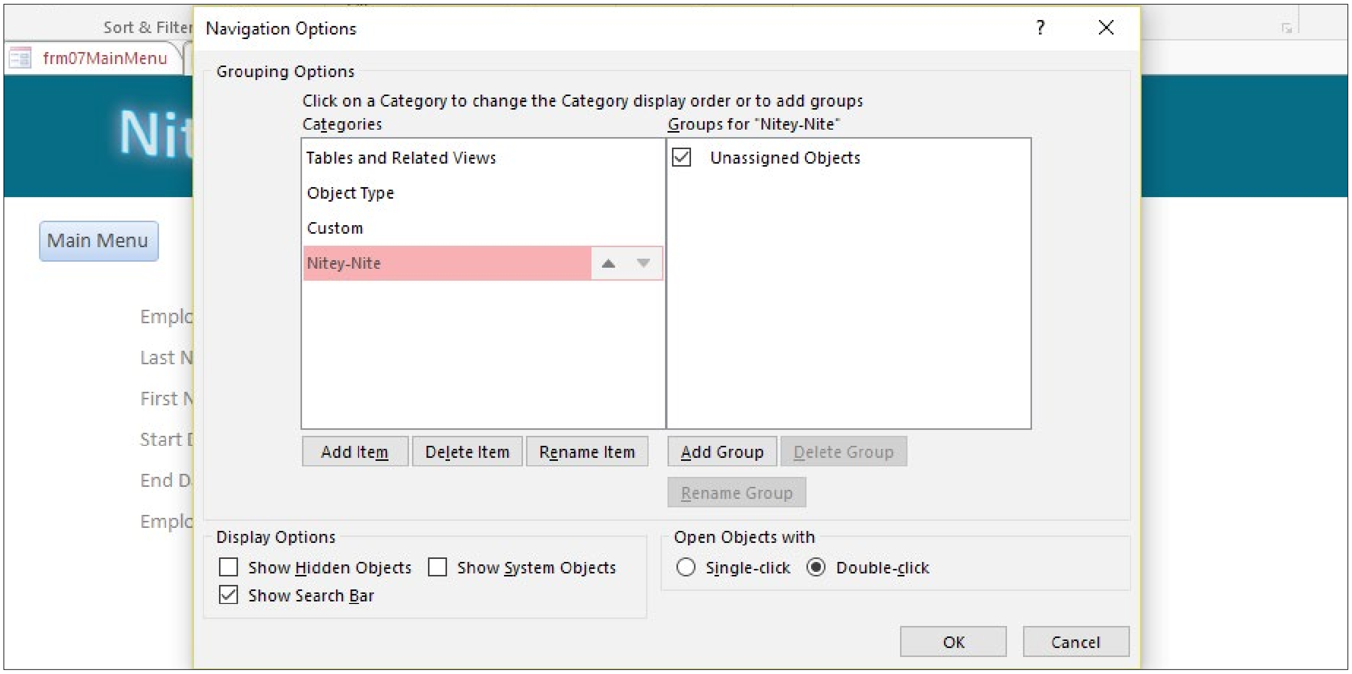
Display Options: □ Show Hidden Objects □ Show System Objects ☑ Show Search Bar

Open Objects with: ○ Single-click ● Double-click

Figure 7.16

The Navigation Options dialog box appears. On the left side of the dialog box are the existing categories. On the right are the groups in which the objects are contained. We will create a new custom category, then create groups, and then add objects to the groups.

2. *Below the* **Categories** *box, click the* **Add Item** *button.*
3. *Click* **Rename Item** *to rename* **Custom Category 1** *as* ***Nitey-Nite*** *and press* **[Enter]**.



Navigation Options

Grouping Options

Click on a Category to change the Category display order or to add groups

Categories	Groups for "Nitey-Nite"
Tables and Related Views	☑ Unassigned Objects
Object Type	
Custom	
Nitey-Nite	

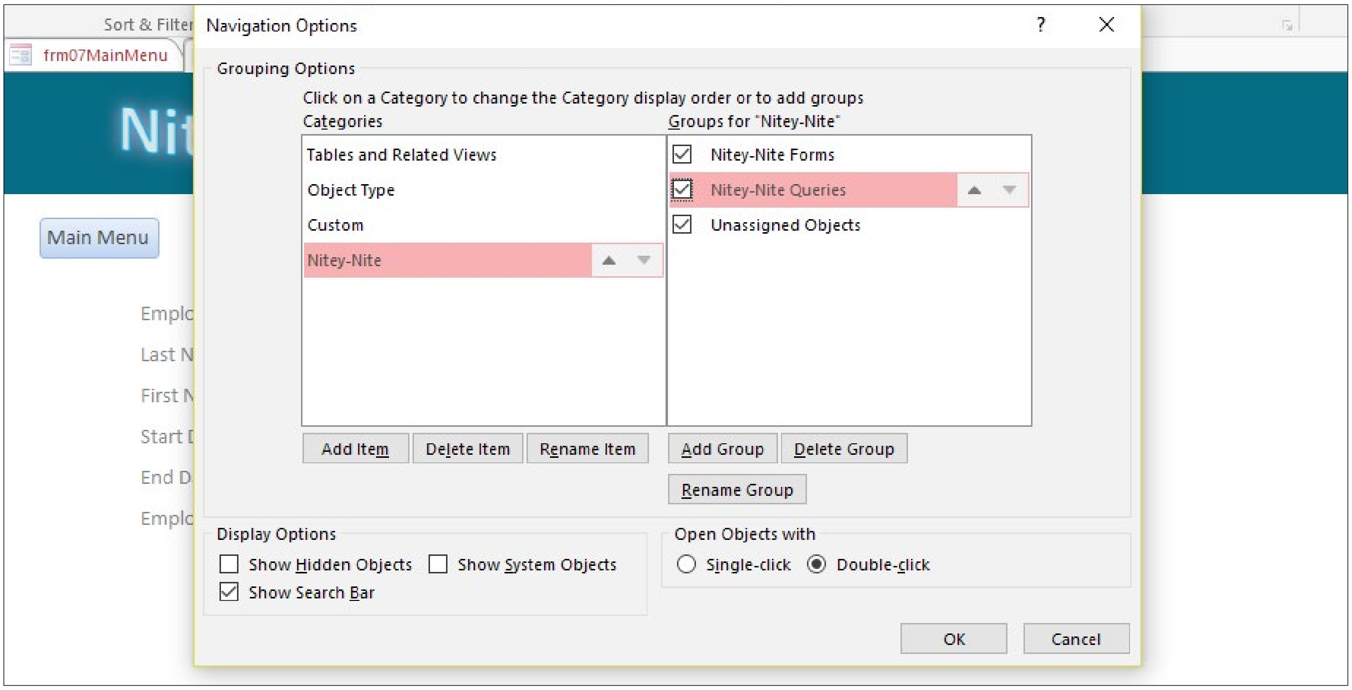
Display Options: □ Show Hidden Objects □ Show System Objects ☑ Show Search Bar

Open Objects with: ○ Single-click ● Double-click

Figure 7.17

With the Nitey-Nite category created, you can now create groups under it.

4. *Click on the* **Add Group** *button below* **Groups for "Nitey-Nite"**.
5. *Rename* **Custom Group 1** *with* ***Nitey-Nite Forms***.
6. *Create a second group called* **Nitey-Nite Queries**.



Navigation Options

Grouping Options

Click on a Category to change the Category display order or to add groups

Categories	Groups for "Nitey-Nite"
Tables and Related Views	☑ Nitey-Nite Forms
Object Type	☑ Nitey-Nite Queries
Custom	☑ Unassigned Objects
Nitey-Nite	

Display Options

□ Show Hidden Objects □ Show System Objects

☑ Show Search Bar

Open Objects with

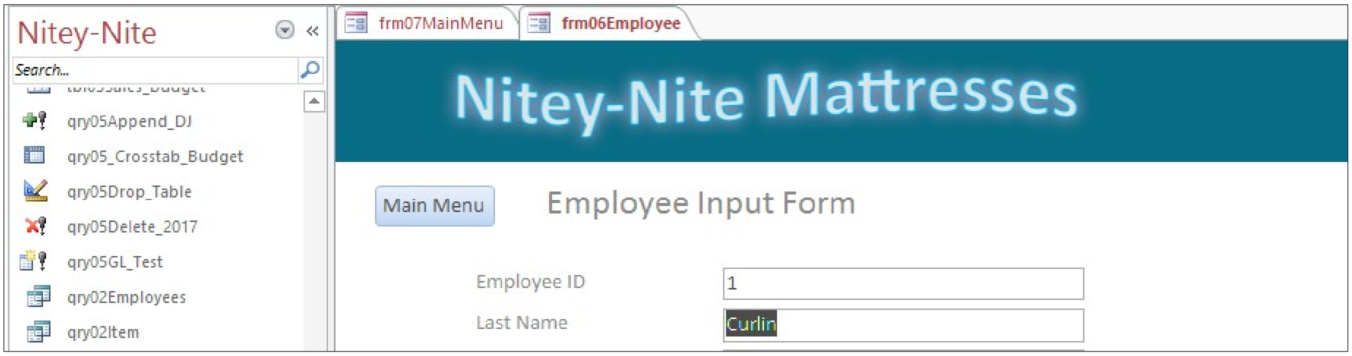
○ Single-click ● Double-click

Figure 7.18

7. *Click* **OK** *in the* **Navigation Options** *dialog box.*

You can use the up and down arrows to the right of each group or category to reposition it if you like, but we won't do that in this exercise. Once you click OK, the dialog box disappears, but the groups are still there – you just have to display them in the Navigation Pane.

8. *Click on the drop-down arrow on the* **Navigation Pane** *next to* **All Access Objects** *and choose* **Nitey-Nite**.

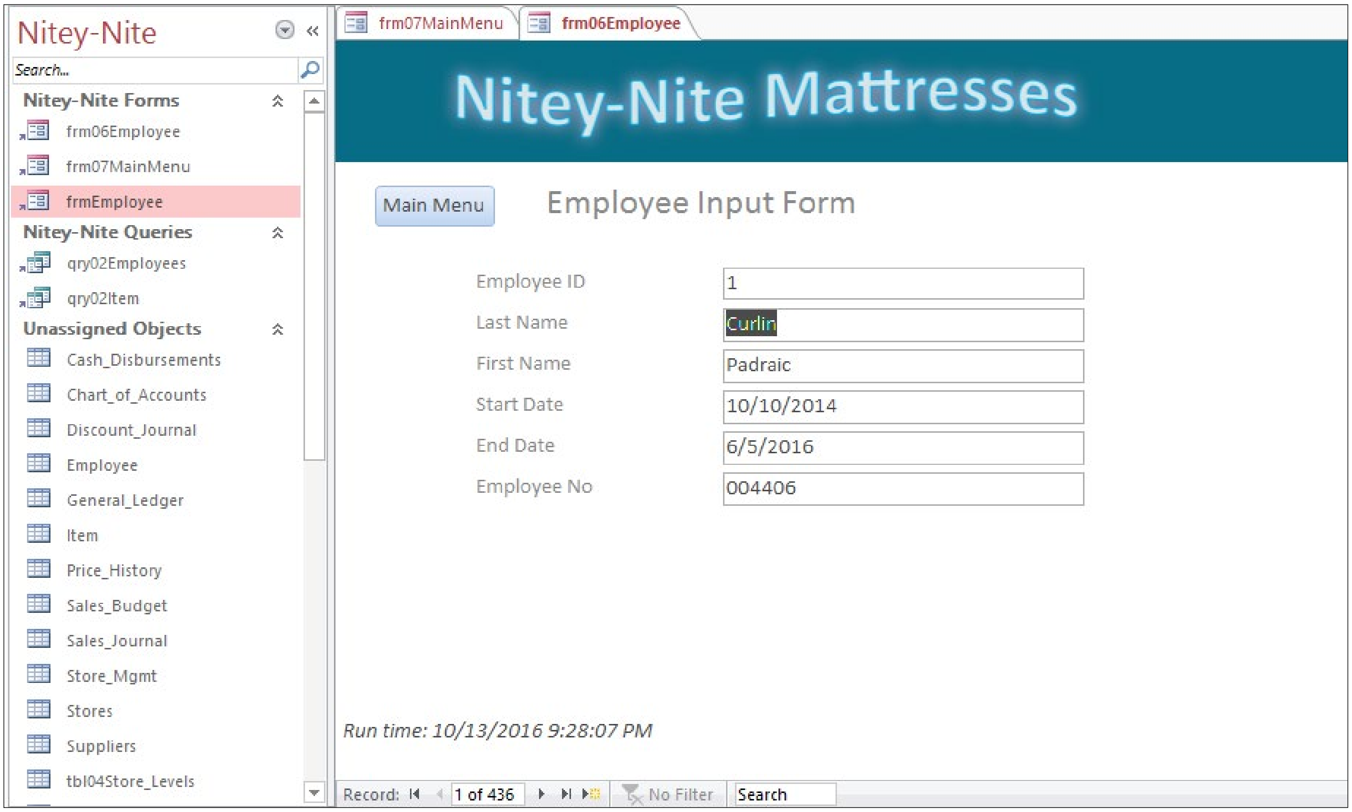


Employee ID: 1
Last Name: Curlin

Figure 7.19

Above the tables, you see there are now three bars under Nitey-Nite (which are the groups), Nitey-Nite Forms, Nitey-Nite Queries, and Unassigned Objects. All of the objects are currently under the Unassigned Objects group, and all you have to do is to drag and drop them in their appropriate places.

9. *Click and drag* **qry02Item** *and drop it on top of the* **Nitey-Nite Queries** *group.*
10. *Repeat the process with* **qry02Employees***.*
11. *Click on* **frm06Employee***, hold down the* **[Ctrl]** *key, click on* **frm07MainMenu***, then on* **frmEmployee** *(to select all three forms) and drag the group up and drop it on the* **Nitey-Nite Forms** *group.*

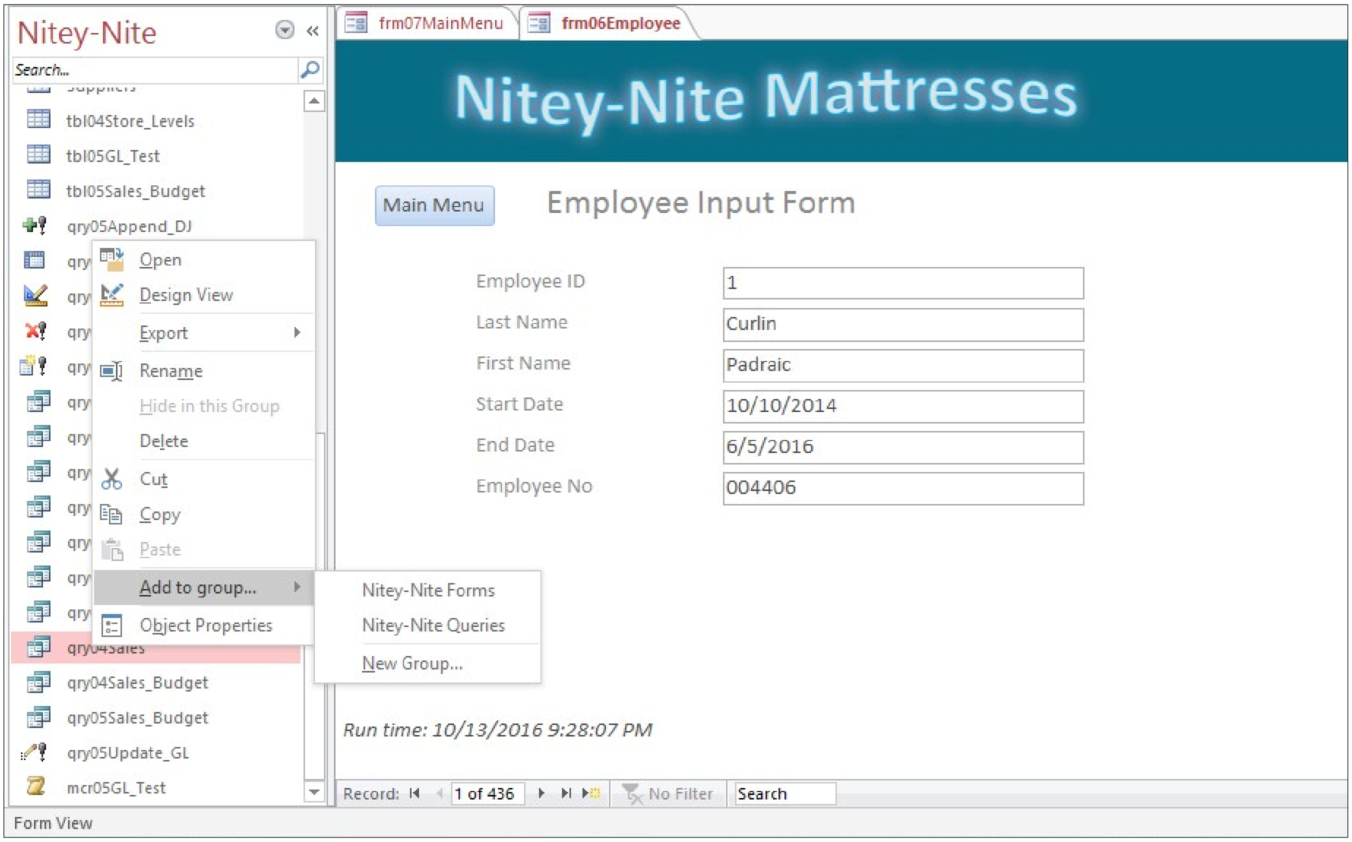


Employee ID: 1
Last Name: Curlin
First Name: Padraic
Start Date: 10/10/2014
End Date: 6/5/2016
Employee No: 004406

Run time: 10/13/2016 9:28:07 PM

Figure 7.20

12. *Right-click on* **qry04Sales**, *point to* **Add to group…**, *and select* **Nitey-Nite Queries**.



Employee Input Form

Employee ID	1
Last Name	Curlin
First Name	Padraic
Start Date	10/10/2014
End Date	6/5/2016
Employee No	004406

Run time: 10/13/2016 9:28:07 PM

Figure 7.21

You can see there are several ways to create shortcuts in the Navigation Pane. Notice that there are small arrows at the bottom-left corner of each object's icon. These small arrows tell the user that this object is a shortcut and not the object itself. You can delete the shortcut with no fear of the object being deleted. You can alternatively move the object out of any group and place it in any other group, including Unassigned Objects.

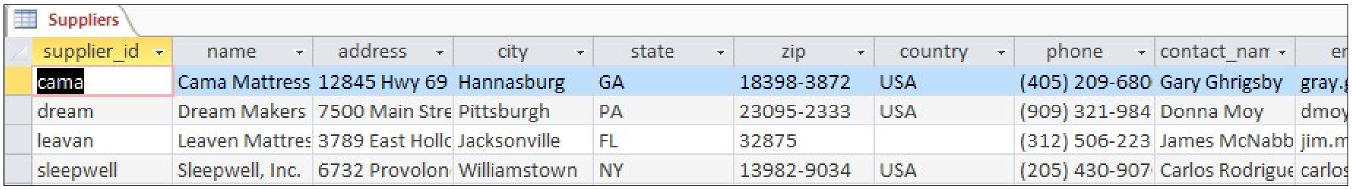
13. *Right-click on the* **Nitey-Nite** *category, point to* **Category**, *and click on* **Object Type** *to have the* **Navigation Pane** *return to* **All Access Objects**.

SubForms

Sometimes it is necessary to view data from multiple tables in one form. When you need to do this, you should give careful thought to the relationships between those tables or queries. A ***SubForm*** can help you immensely in this regard.

In the next exercise, your manager has asked you to create a form where he can see the supplier (or manufacturer) information as well as a listing of all of the products they provide. The top portion of the form should contain the supplier information and the bottom portion should be a list of all of the items. Let's first review the tables on which the forms will be built.

1. *Open the* **Suppliers** *table.*



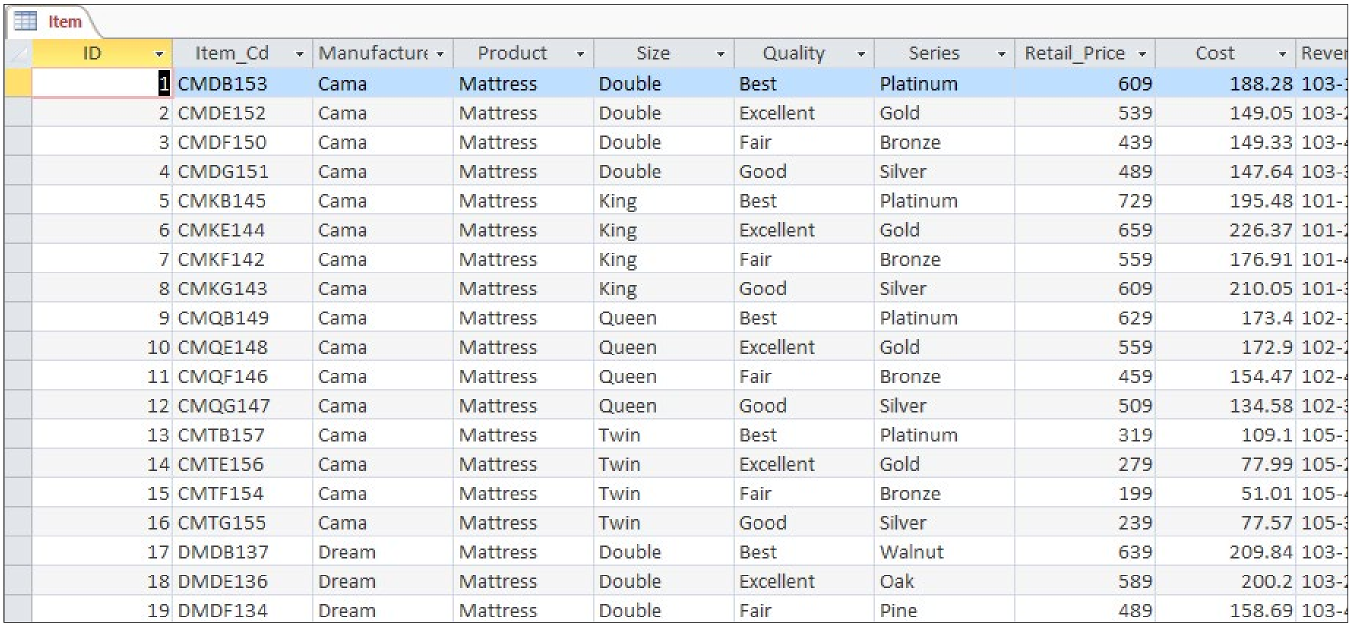
Suppliers

supplier_id	name	address	city	state	zip	country	phone	contact_nam	er
cama	Cama Mattress	12845 Hwy 69	Hannasburg	GA	18398-3872	USA	(405) 209-680	Gary Ghrigsby	gray.g
dream	Dream Makers	7500 Main Stre	Pittsburgh	PA	23095-2333	USA	(909) 321-984	Donna Moy	dmoy
leavan	Leaven Mattres	3789 East Hollc	Jacksonville	FL	32875		(312) 506-223	James McNabb	jim.m
sleepwell	Sleepwell, Inc.	6732 Provolon	Williamstown	NY	13982-9034	USA	(205) 430-907	Carlos Rodrigue	carlos

Figure 7.22

This is a simple table that contains only four records with the information about each of the manufacturers that Nitey-Nite uses.

2. *Close the* **Suppliers** *table and open the* **Item** *table.*



Item

ID	Item_Cd	Manufacture	Product	Size	Quality	Series	Retail_Price	Cost	Rever
1	CMDB153	Cama	Mattress	Double	Best	Platinum	609	188.28	103-1
2	CMDE152	Cama	Mattress	Double	Excellent	Gold	539	149.05	103-2
3	CMDF150	Cama	Mattress	Double	Fair	Bronze	439	149.33	103-4
4	CMDG151	Cama	Mattress	Double	Good	Silver	489	147.64	103-3
5	CMKB145	Cama	Mattress	King	Best	Platinum	729	195.48	101-1
6	CMKE144	Cama	Mattress	King	Excellent	Gold	659	226.37	101-2
7	CMKF142	Cama	Mattress	King	Fair	Bronze	559	176.91	101-4
8	CMKG143	Cama	Mattress	King	Good	Silver	609	210.05	101-3
9	CMQB149	Cama	Mattress	Queen	Best	Platinum	629	173.4	102-1
10	CMQE148	Cama	Mattress	Queen	Excellent	Gold	559	172.9	102-2
11	CMQF146	Cama	Mattress	Queen	Fair	Bronze	459	154.47	102-4
12	CMQG147	Cama	Mattress	Queen	Good	Silver	509	134.58	102-3
13	CMTB157	Cama	Mattress	Twin	Best	Platinum	319	109.1	105-1
14	CMTE156	Cama	Mattress	Twin	Excellent	Gold	279	77.99	105-2
15	CMTF154	Cama	Mattress	Twin	Fair	Bronze	199	51.01	105-4
16	CMTG155	Cama	Mattress	Twin	Good	Silver	239	77.57	105-3
17	DMDB137	Dream	Mattress	Double	Best	Walnut	639	209.84	103-1
18	DMDE136	Dream	Mattress	Double	Excellent	Oak	589	200.2	103-2
19	DMDF134	Dream	Mattress	Double	Fair	Pine	489	158.69	103-4

Figure 7.23

You should already be familiar with this table. It is a listing of every item that is in Nitey-Nite's inventory. Your job is to create a form that shows all of the information from both of these tables. As such, you first have to think about joining the two tables. The key fields in these tables would be the supplier_id field in the Suppliers table and the Manufacturer field in the Item table. Whenever you use these two tables, this join should always exist, so setting it up in the Relationships view maintains those relationships, and whenever you use these tables in a query, the relationship will be built automatically.

3. *Close the* **Item** *table and open the* **Relationships** *view on the* **Database Tools** *tab.*
4. *Bring in the* **Item** *table and the* **Suppliers** *table, and create a join on the* **Manufacturer** *field and the* **supplier_id** *field.*
5. *In the join,* **Enforce referential integrity**.

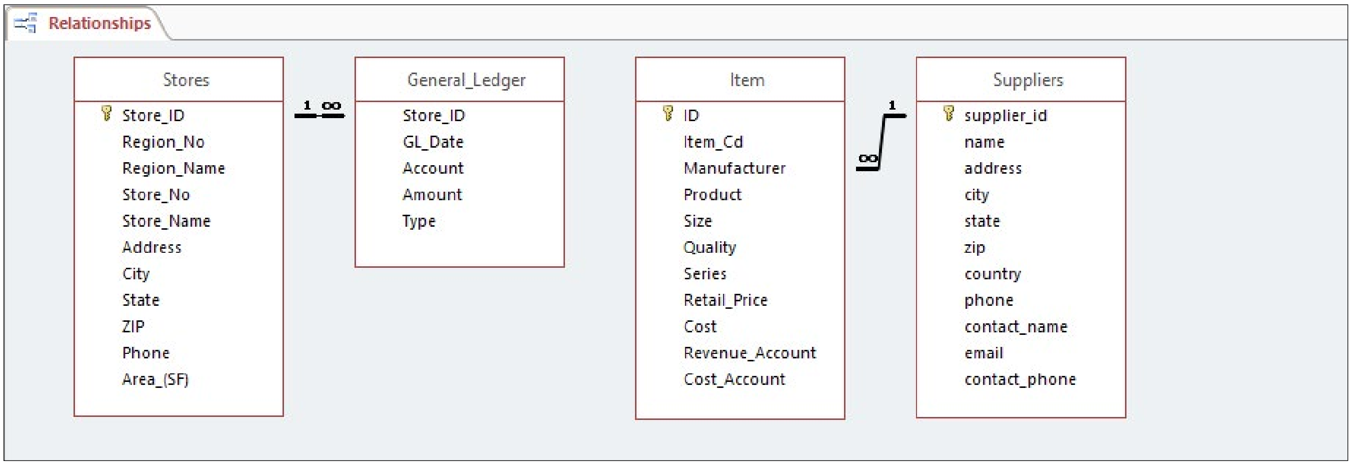


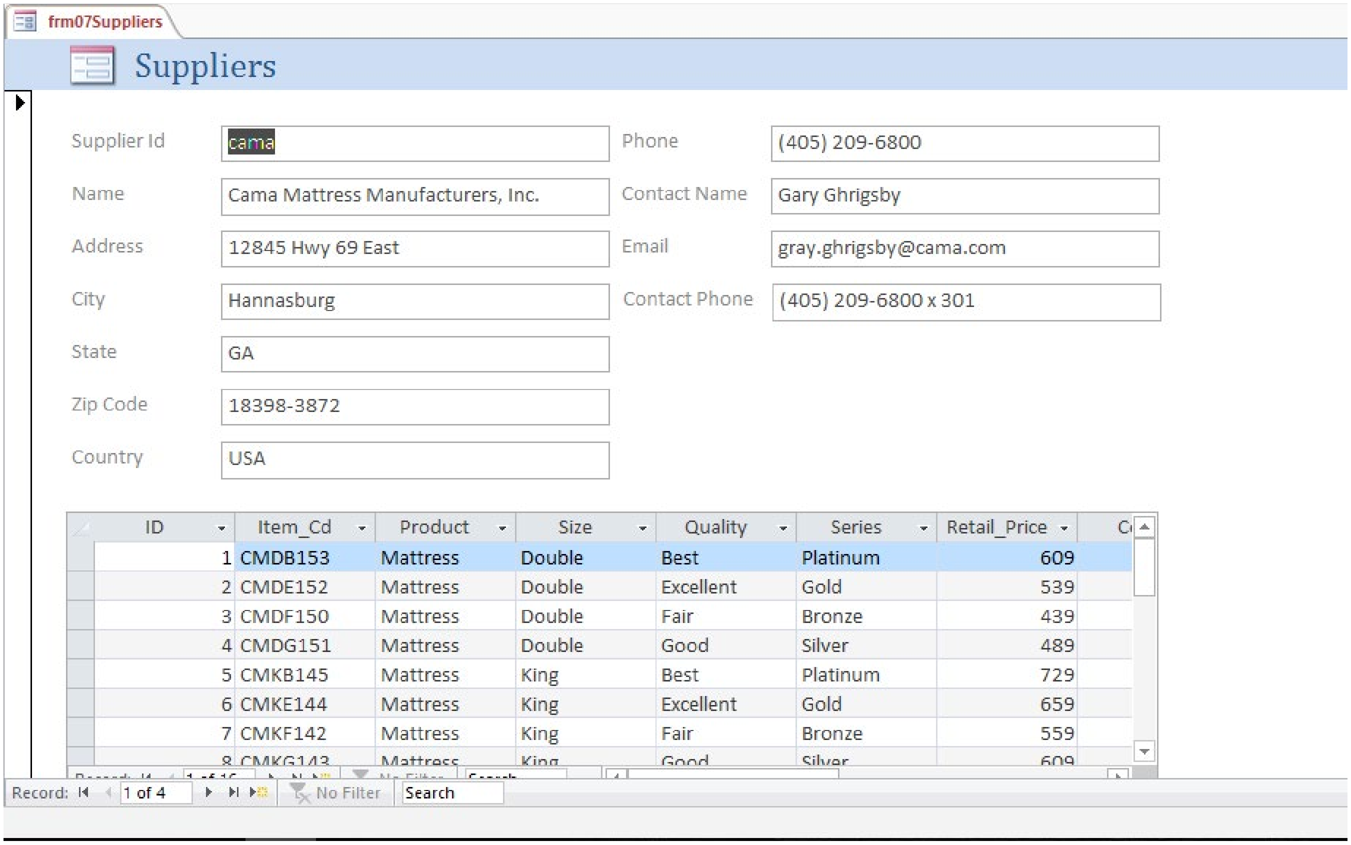
Figure 7.24

6. *Close the* **Relationships** *view. Choose* **Yes** *when asked if you want to save it.*

Notice in the Suppliers table how the expand option appears because of this relationship you just setup. Now that the tables are related to each other, you can create the form.

7. *Click on the* **Suppliers** *table, then click on the* **Form** *icon in the* **Forms** *group of the* **Create** *tab.*
8. *Edit the form's design to look like* **Figure 7.25**.
9. **Save** *the form as* ***frm07Suppliers***.

> ***Note****: You may have to delete the* **Table.Item** *table at the bottom of the form, take off the* **Record Selectors***, and resize and reposition the text boxes and labels within the form. Also, you may have to right-click the text boxes you want to move, click* **Layout***, then* **Remove Layout** *from the menu to separate the text boxes from their labels.*



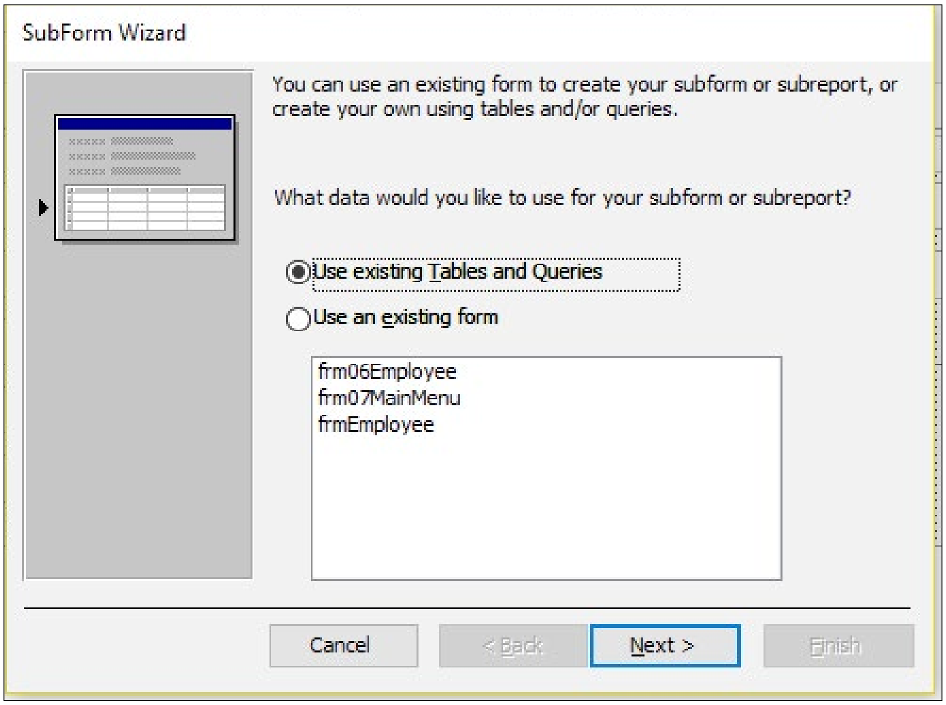
Supplier Id	cama	Phone	(405) 209-6800
Name	Cama Mattress Manufacturers, Inc.	Contact Name	Gary Ghrigsby
Address	12845 Hwy 69 East	Email	gray.ghrigsby@cama.com
City	Hannasburg	Contact Phone	(405) 209-6800 x 301
State	GA		
Zip Code	18398-3872		
Country	USA		

ID	Item_Cd	Product	Size	Quality	Series	Retail_Price	C
1	CMDB153	Mattress	Double	Best	Platinum	609	
2	CMDE152	Mattress	Double	Excellent	Gold	539	
3	CMDF150	Mattress	Double	Fair	Bronze	439	
4	CMDG151	Mattress	Double	Good	Silver	489	
5	CMKB145	Mattress	King	Best	Platinum	729	
6	CMKE144	Mattress	King	Excellent	Gold	659	
7	CMKF142	Mattress	King	Fair	Bronze	559	
8	CMKG143	Mattress	King	Good	Silver	609	

Figure 7.25

Now you need to add the list of items to the bottom of the form, or create a form within a form. This is done through the SubForm/SubReport control.

10. *In the* **Design View** *of the form, make sure you have about* **3”** *of space below the last text box to work with below the* **Suppliers** *fields in the form.*
11. *Scroll down in the* **Controls** *group of the* **Form Design Tools Design** *tab and click on the* **SubForm/SubReport** *control* .
12. *With your mouse, draw a rectangle starting just below the last text box extending down about* **2”** *and over to the right margin of the text boxes on the right and release the mouse.*



SubForm Wizard

You can use an existing form to create your subform or subreport, or create your own using tables and/or queries.

What data would you like to use for your subform or subreport?

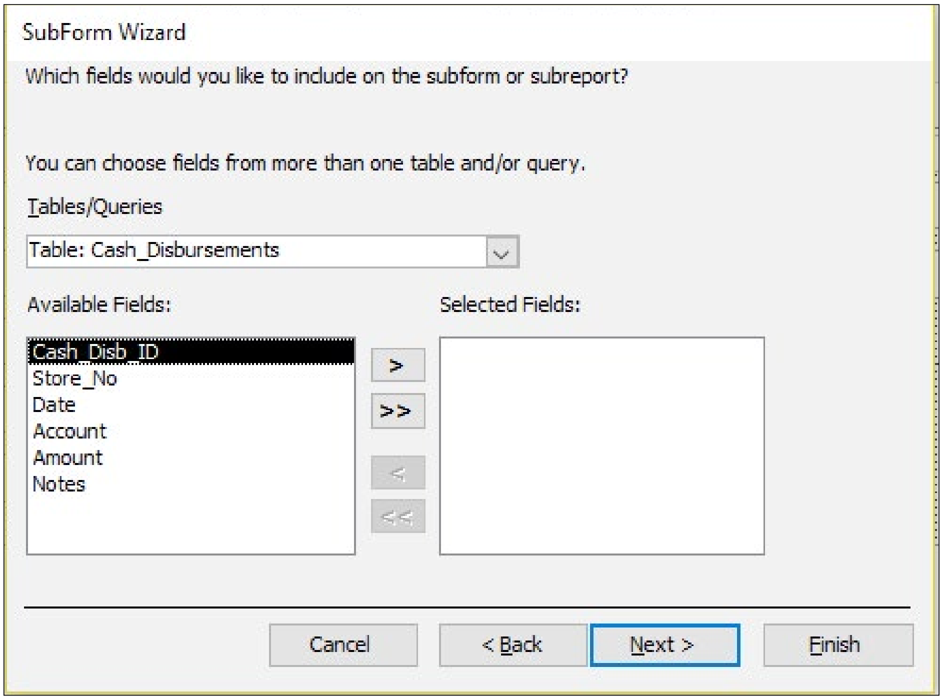
● Use existing Tables and Queries

○ Use an existing form

frm06Employee
frm07MainMenu
frmEmployee

Figure 7.26

14. *Make sure the* **Use existing Tables and Queries** *radio button is selected and click* **Next >**.



SubForm Wizard

Which fields would you like to include on the subform or subreport?

You can choose fields from more than one table and/or query.

Tables/Queries

Table: Cash_Disbursements

Available Fields:

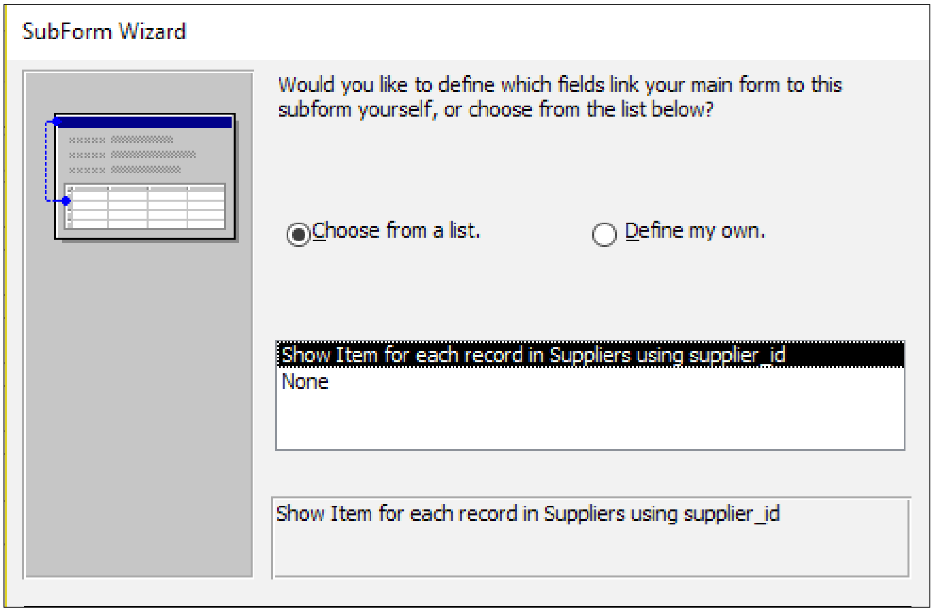
Cash_Disb_ID
Store_No
Date
Account
Amount
Notes

Selected Fields:

Figure 7.27

15. *In the* **Tables/Queries** *drop-down menu, choose* **Table: Item**.

16. *Move all of the fields over to the* **Selected Fields** *area, except for* **ID**, **Revenue_Account**, *and* **Cost_Account** *and click* **Next >**.



Would you like to define which fields link your main form to this subform yourself, or choose from the list below?

● Choose from a list. ○ Define my own.

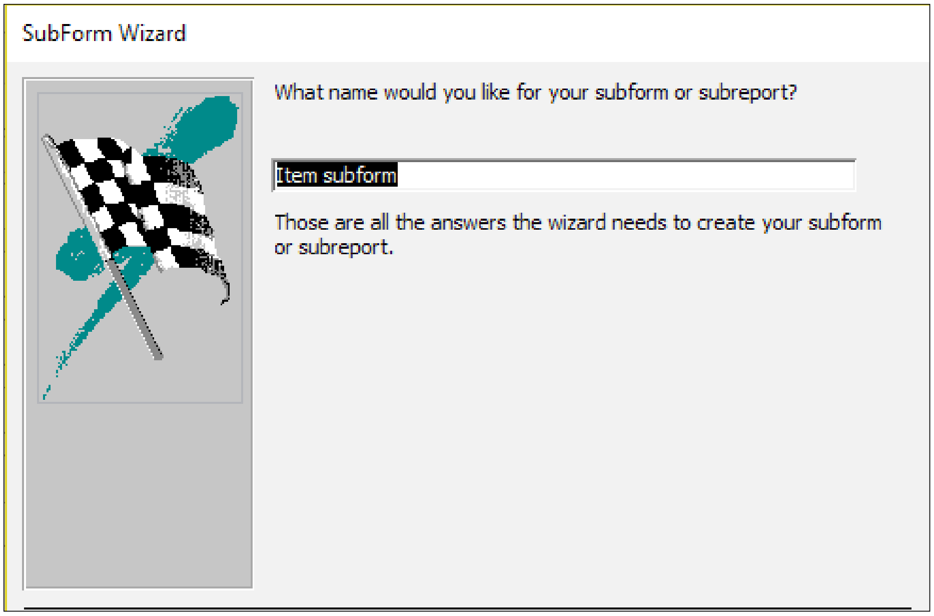
Show Item for each record in Suppliers using supplier_id
None

Show Item for each record in Suppliers using supplier_id

Figure 7.28

The next screen is a really cool thing about Access 2016. Since we have already defined a relationship between the Suppliers table and the Item table, Access knows that we will probably want to keep that relationship here in this screen.

16. *Make sure the* **Choose from a list.** *radio button is selected, then click* **Next >**.



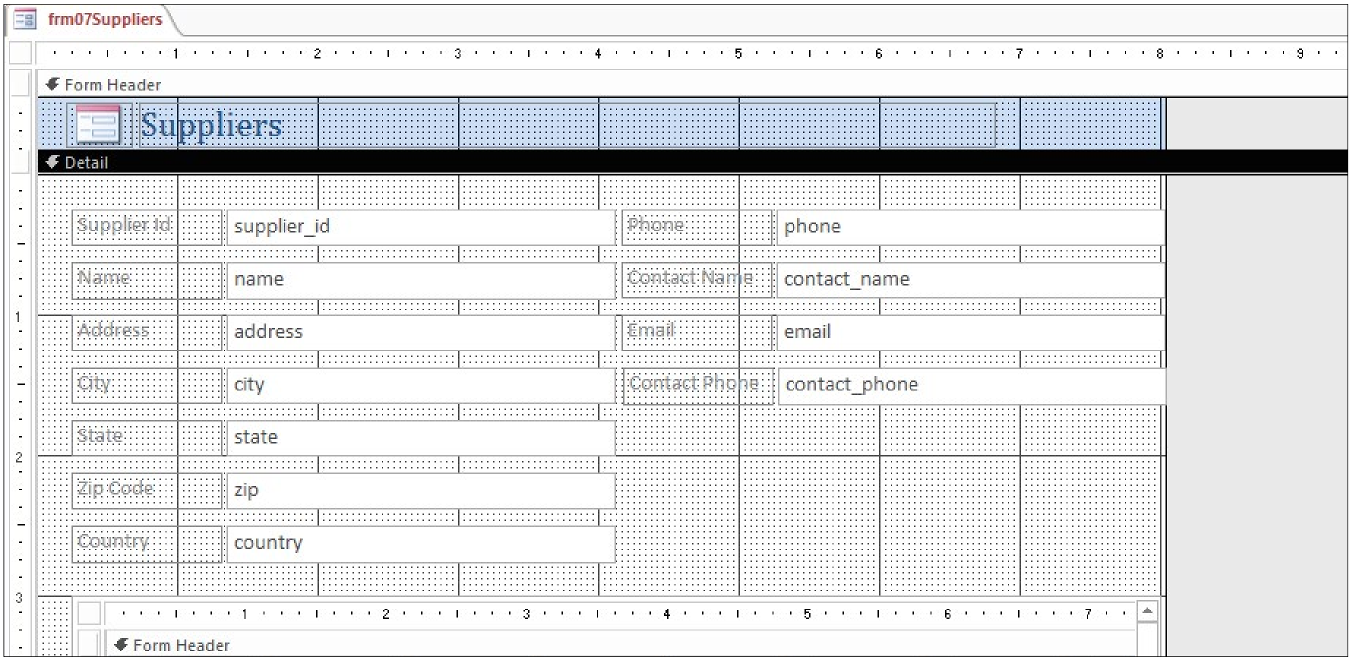
What name would you like for your subform or subreport?

Item subform

Those are all the answers the wizard needs to create your subform or subreport.

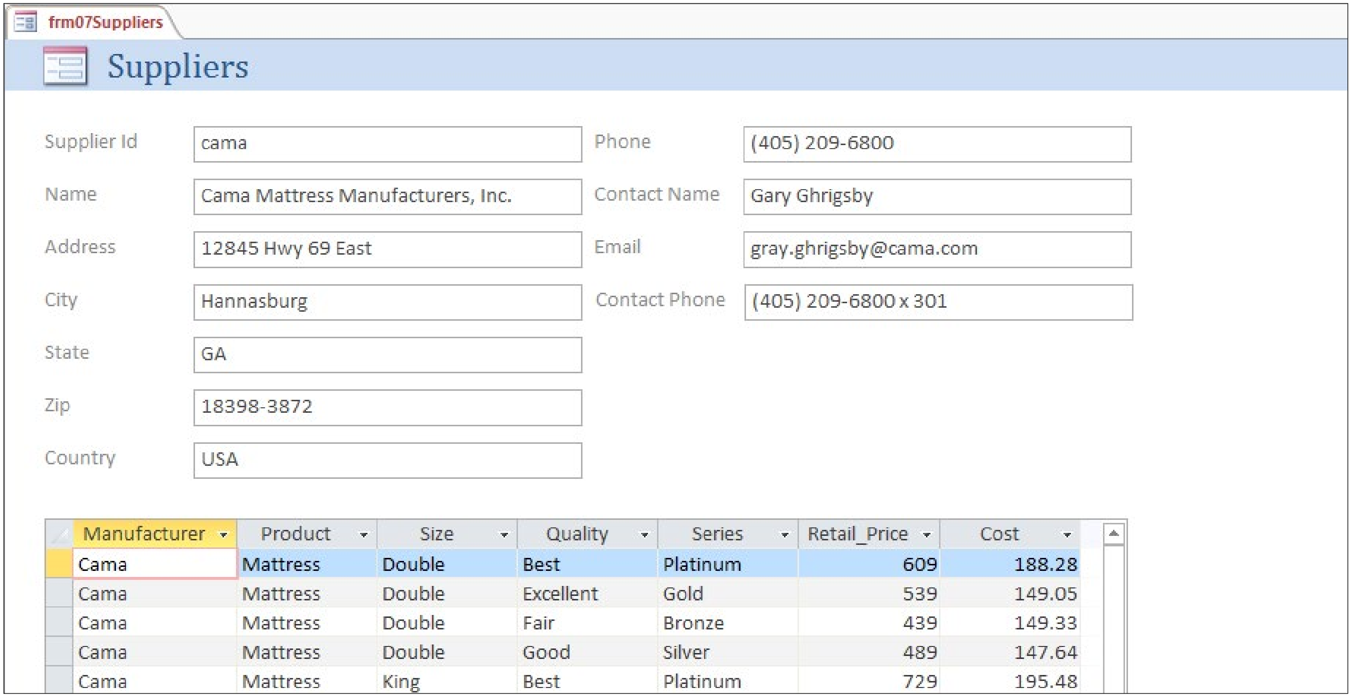
Figure 7.29

17. *Keep the default subform name and click* **Finish**.
18. *Delete the label* **Item subform** *and rearrange the subform to make it look like* **Figures 7.30** *and* **7.31**.



Supplier Id	supplier_id	Phone	phone
Name	name	Contact Name	contact_name
Address	address	Email	email
City	city	Contact Phone	contact_phone
State	state		
Zip Code	zip		
Country	country		

Figure 7.30



Supplier Id	cama	Phone	(405) 209-6800
Name	Cama Mattress Manufacturers, Inc.	Contact Name	Gary Ghrigsby
Address	12845 Hwy 69 East	Email	gray.ghrigsby@cama.com
City	Hannasburg	Contact Phone	(405) 209-6800 x 301
State	GA		
Zip	18398-3872		
Country	USA		

Manufacturer	Product	Size	Quality	Series	Retail_Price	Cost
Cama	Mattress	Double	Best	Platinum	609	188.28
Cama	Mattress	Double	Excellent	Gold	539	149.05
Cama	Mattress	Double	Fair	Bronze	439	149.33
Cama	Mattress	Double	Good	Silver	489	147.64
Cama	Mattress	King	Best	Platinum	729	195.48

Figure 7.31

It may take a little work to get it to look just right, but such a form can be very powerful. One downside to this method is that the Subform column widths readjust once you close the form, but you can fix this in the Item Subform form that was created. You can also change the data source to be based on a query or a table. SubReports can be created in the same way. Play around with it a little bit. Click on the navigation button for the main form (Record 1 of 4) and see how the data in the subform correspondingly changes.

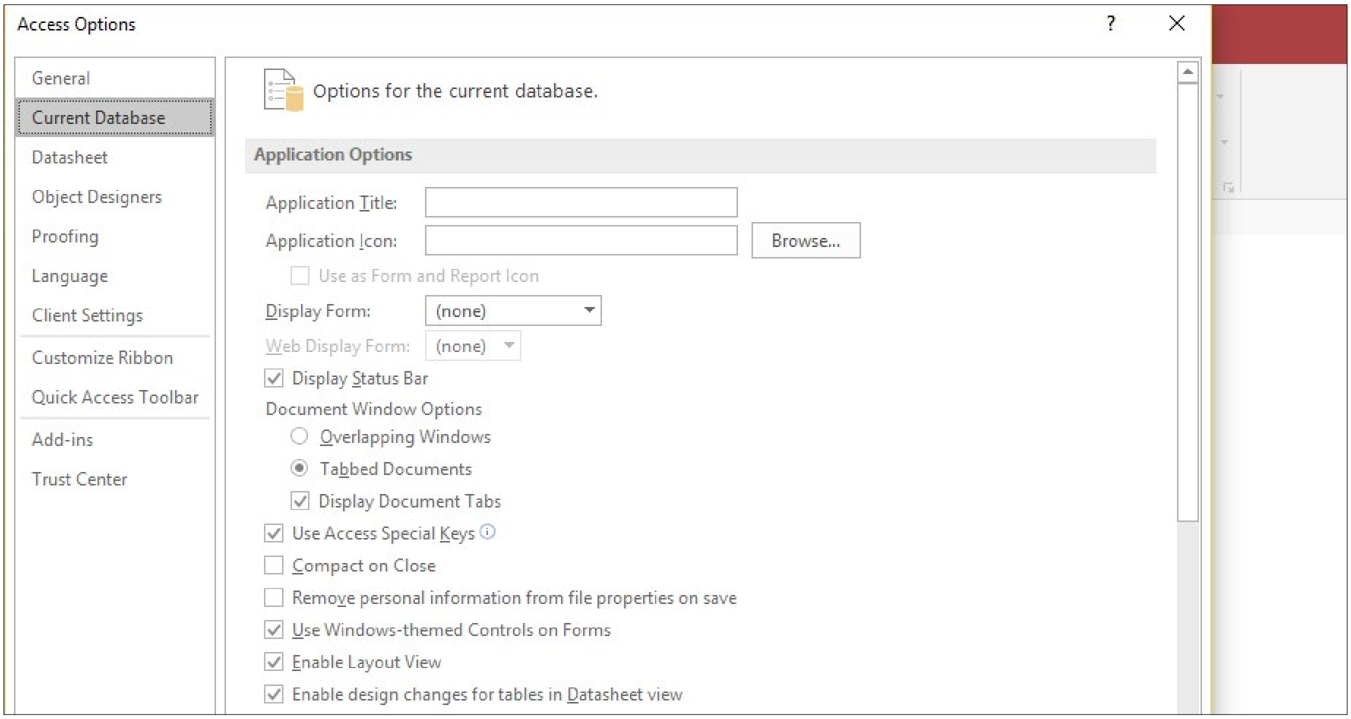
19. **Save** *and close* **frm07Suppliers***.*

Using the linked table Subform option has some benefits, namely that it is quick and relatively esasy to customize. Just remember that the base table or query formatting will carry in to the Subform and changes you make in the Subform, such as hiding columns, will be reflected in the base table or query.

Startup Functionality and Special Features

The last item I want to cover in forms is the Startup functionality. You can program Access to do certain things whenever the database is opened. For example, you may want for the database to run a macro, open a form, or run a report. This is sometimes a good idea, as you may have some inexperienced users that don't know what needs to be done when they first open the database. I usually like to have the Main Menu form open up whenever anyone opens the database. Sometimes you'll want to be like the government (pardon my political humor), and totally control what the user can see and do. In the next exercise, you will make the Main Menu form open immediately when the database opens and restrict some other features in the database.

1. *Close all forms and any other Access objects you may have open.*
2. *Click on the* **File** *tab, and then click on the* **Options** *button.*
3. *On the left side of the* **Access Options** *dialog box, click on* **Current Database***.*



Options for the current database.

Application Options

Application Title:

Application Icon:

□ Use as Form and Report Icon

Display Form: (none)

Web Display Form: (none)

☑ Display Status Bar

Document Window Options

○ Overlapping Windows

● Tabbed Documents

☑ Display Document Tabs

☑ Use Access Special Keys ⓘ

□ Compact on Close

□ Remove personal information from file properties on save

☑ Use Windows-themed Controls on Forms

☑ Enable Layout View

☑ Enable design changes for tables in Datasheet view

Figure 7.32

4. *In the* **Display Form:** *drop-down menu, choose* **frm07MainMenu**.
5. *Make sure the* **Display Navigation Pane** *check box is cleared in the* **Navigation** *section.*
6. *In the* **Ribbon and Toolbar Options** *section, clear the* **Allow Full Menus** *and* **Allow Default Shortcut Menus** *boxes and click* **OK** *in the* **Access Options** *dialog box.*
7. *Click* **OK** *when the warning message,* **You must close and reopen the current database for the specified option to take effect**, *appears.*
8. *Close and reopen the* **Nitey_Nite_2016** *database.*

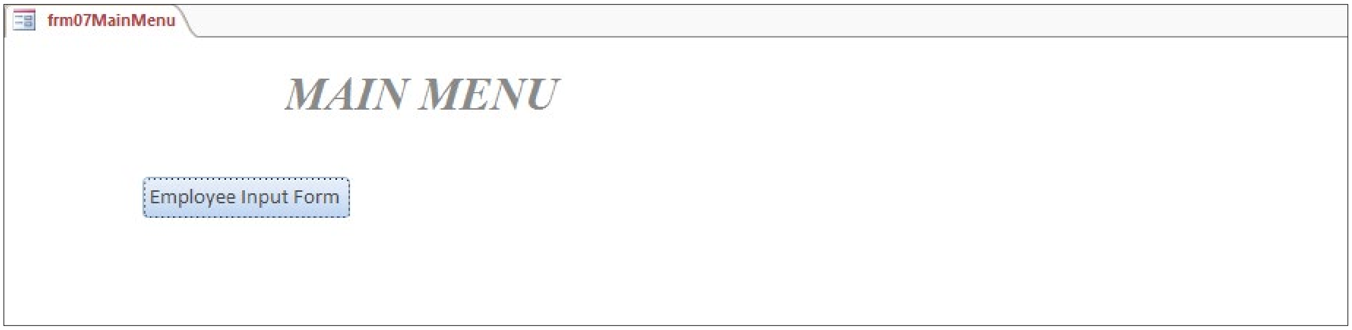


Figure 7.33

Notice that a lot of functionality has been taken away. The Main Menu form (frm07MainMenu) automatically opens, the Navigation Pane to the left of the form does not appear, and the menu tabs across the top of the ribbon are gone. Also notice that the right-click does not work. This is what was turned off when you unchecked the Allow Default Shortcut Menus box. The only choice the user has is to click on the Employee Input Form. You can get the Navigation Pane to reappear by pressing [F11].

9. *Press* **[F11]**.

The Navigation Pane reappears. Typically, only an experienced Access user will know about this special key ([F11] hides or displays the Navigation Pane), but if you want to make sure the user can't see the Navigation Pane, you can turn off Use Access Special Keys in the Access Options dialog box. If at any time you have programmed something into the startup options and you want to bypass those options when opening the database, simply hold down the [Shift] key while the database is opening.

10. *Close the* **Nitey_Nite_2016** *database.*
11. *Reopen the database while holding down the* **[Shift]** *key.*
12. *If a safety warning message appears, continue to hold down the* **[Shift]** *key when you click on the warning message.*

Notice how the database now opens as if none of the startup functions you changed were executed.

13. *Go back into the* **Access Options** *for the* **Current Database**, *check the* **Display Navigation Pane** *box, as well as the* **Allow Full Menus** *and* **Allow Default Shortcut Menus** *boxes.*
14. *Click* **OK** *in the* **Access Options** *dialog box.*

Your database now returns to normal with the Main Menu form opening when the database opens.

> ***Review Questions****: It is now time to complete the hands-on Review Questions. Log on to www.ExcelCEO.com with your Email and Password, click on the* **Access 2016 and SQL Review Questions, Chapter 7, Section 2 of 2** *option and complete the review questions.*

Conclusion

In this chapter, you created an unbound form to use as the Main Menu. You created a Command Button to make it easier for users to navigate around the database. You were exposed to a little more macro programming to tweak the Command Button to do a little more than what's available in the form wizard. You created a custom group which gives structure on navigating around the database. You created a form that has a subform so the user can look at a more detailed level of data in the same form. Finally, you learned how to automate a procedure to occur every time you open the database by using Startup functionality and learned how to maximize database forms using the On Open property.

Chapter Exam

You can now go to www.ExcelCEO.com, click on Sign In, log in and take the exam. Make sure that you take the exam on the same device on which you completed the practice files, as some of the questions on the exam may refer to some of the completed projects. Chapter exams are intended to be hands-on.

CHAPTER EIGHT — BEGINNING REPORTING

In this chapter, you will:

- Determine how to create a simple report using the Report Wizard
- Identify and use the Design View of a Report
- Select the various properties and formatting of Labels and Text Boxes
- Recognize the Sorting and Grouping dialog boxes
- Identify subtotals within a report
- Select objects within a report horizontally and vertically

CPE Credits possible for this chapter: 3

Reporting Basics

Once I trained an employee who was a recent college graduate with a double major in Accounting and Finance. She was very excited to learn about Excel. I worked with her for a couple of months and she took to Excel like a duck after a June bug! She was good! Then we started on Access. Her first project in Access was to develop a simple report based on a simple table. That report ate her proverbial lunch. After that experience, she made great strides and is today one of the best Access developers I know. Hopefully you too will join the ranks of being the best.

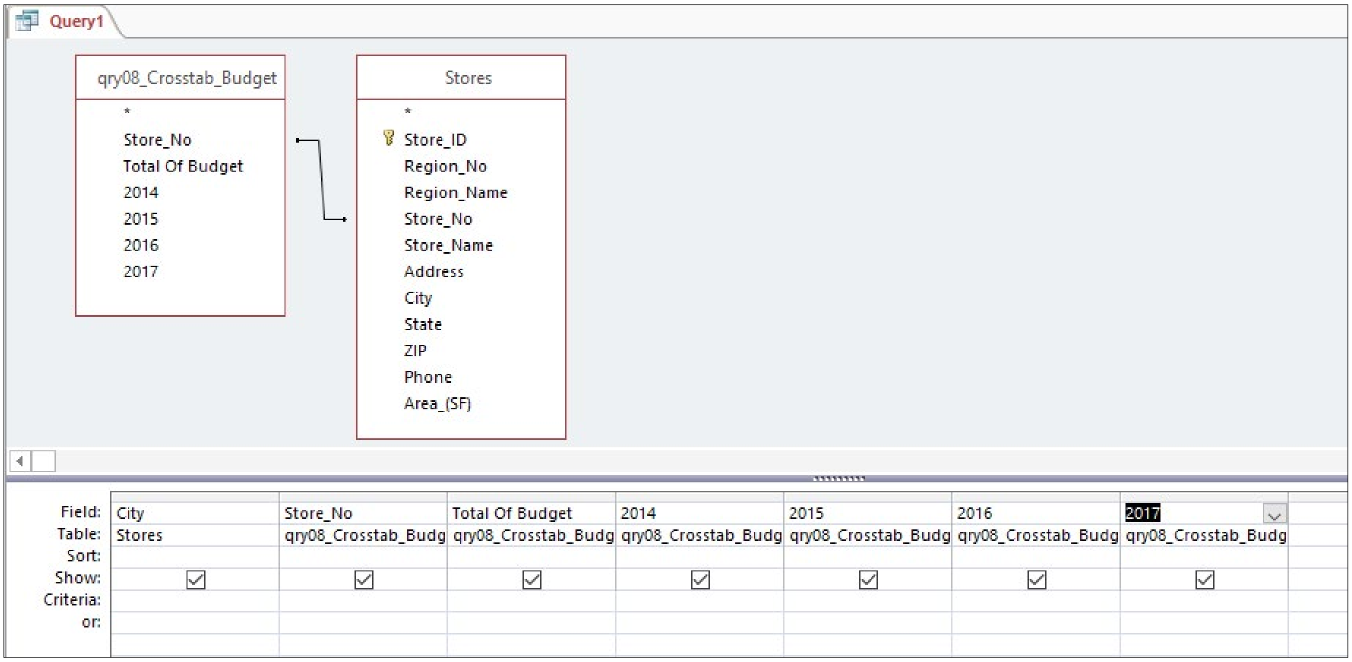
In this chapter, you will develop a very simple report, so you can learn the basics of creating reports. In subsequent chapters, we'll get into the cool stuff. This chapter will concentrate on reporting basics.

Create a Simple Report

The first thing that I want to stress about Access Report Writing is that you should NEVER base a report on a table. It should always be based on a query, even if the query just runs the data in the table. The reason I say that is because whenever you develop a report, someone will look at the final product and say, "*This is great! Do you think you can make it include yaddy, yaddy, yadda...?*" People, particularly managers, will almost always want to change something in the report, and it is much easier to change data in a query than in a table.

In Chapter 5, you created a crosstab query that was based on the Sales_Budget table. Management now wants us to create a nice clean report that they can show to the board of directors. The report should show each city and store along the left side of the report, and the Years 2014, 2015, 2016, and 2017 as columns with the budget numbers as the data. They also want to see totals by city and for the entire company. The base query that we'll use is the crosstab query, and we'll create a join to the Stores table to get the city name. With that said, let's get started.

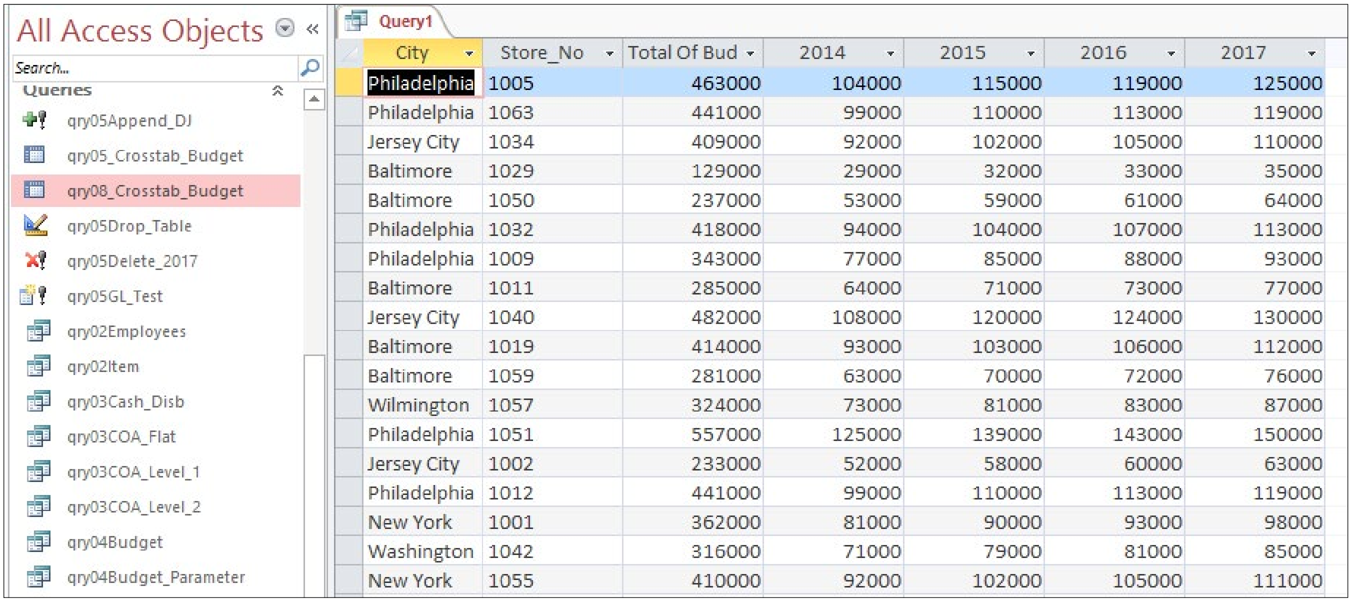
1. *Open the* **Nitey_Nite_2016** *database and close the* **Main Menu** *form.*
2. **Copy qry05Crosstab_Budget** *and name the new query* ***qry08Crosstab_Budget***.
3. **Create** *a new query that joins* **qry08Crosstab_Budget** *with the* **Stores** *table on* **Store_No**.
4. *Bring the* **City** *field from the* **Stores** *table, and all fields from* **qry08Crosstab_Budget**.



Field:	City	Store_No	Total Of Budget	2014	2015	2016	2017
Table:	Stores	qry08_Crosstab_Budg	qry08_Crosstab_Budg	qry08_Crosstab_Budg	qry08_Crosstab_Budg	qry08_Crosstab_Budg	qry08_Crosstab_Budg
Sort:							
Show:	☑	☑	☑	☑	☑	☑	☑
Criteria:							
or:							

Figure 8.1

5. **Run** *the query.*



City	Store_No	Total Of Bud	2014	2015	2016	2017
Philadelphia	1005	463000	104000	115000	119000	125000
Philadelphia	1063	441000	99000	110000	113000	119000
Jersey City	1034	409000	92000	102000	105000	110000
Baltimore	1029	129000	29000	32000	33000	35000
Baltimore	1050	237000	53000	59000	61000	64000
Philadelphia	1032	418000	94000	104000	107000	113000
Philadelphia	1009	343000	77000	85000	88000	93000
Baltimore	1011	285000	64000	71000	73000	77000
Jersey City	1040	482000	108000	120000	124000	130000
Baltimore	1019	414000	93000	103000	106000	112000
Baltimore	1059	281000	63000	70000	72000	76000
Wilmington	1057	324000	73000	81000	83000	87000
Philadelphia	1051	557000	125000	139000	143000	150000
Jersey City	1002	233000	52000	58000	60000	63000
Philadelphia	1012	441000	99000	110000	113000	119000
New York	1001	362000	81000	90000	93000	98000
Washington	1042	316000	71000	79000	81000	85000
New York	1055	410000	92000	102000	105000	111000

Figure 8.2

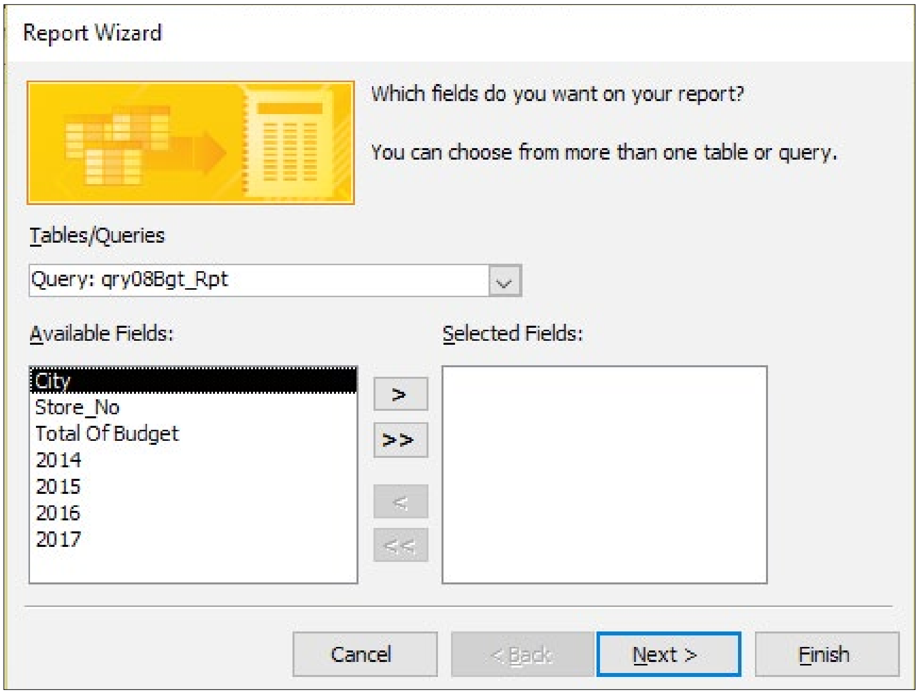
It has all of the correct information in it, but it's not sorted or formatted, and it just doesn't look pretty. Using this query as our data source, we will create a report.

6. **Save** *the query as* ***qry08Bgt_Rpt*** *and close it.*

Using the Report Wizard

As with Forms, there are a number of ways to create a report. The first way is to click on the table or query you want to base the report on in the Navigation pane, and click on the Report icon in the Reports group of the Create tab. The Report icon, however, gives a very basic report. You can experiment with that on your own. For the purposes of this exercise, we'll launch the Report Wizard and walk through the basic components of a report.

7. *Click on the* **Create** *tab in the* **Office Ribbon**.
8. *Make sure that* **qry08Bgt_Rpt** *is selected in the* **Navigation Pane**.
9. *In the* **Reports** *group of the* **Create** *tab, click on the* **Report Wizard** *icon* Report Wizard .



Report Wizard

Which fields do you want on your report?

You can choose from more than one table or query.

Tables/Queries

Query: qry08Bgt_Rpt

Available Fields:

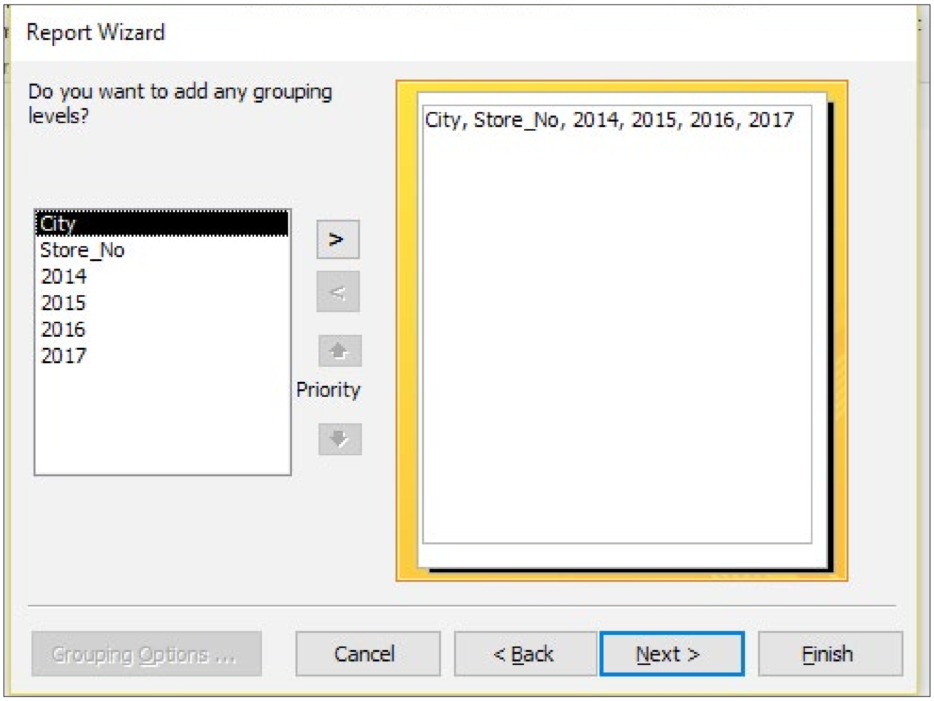
- City
- Store_No
- Total Of Budget
- 2014
- 2015
- 2016
- 2017

Selected Fields:

Figure 8.3

The first screen of the Report Wizard asks you to define the data source and which fields you will be using in the report. Since you had already selected qry08Bgt_Rpt, Access assumes you want to use that query as your data source.

10. *Click the* >> *button to move all fields from the* **Available Fields** *area to the* **Selected Fields** *area, then click on the* **Total of Budget** *field and click the* < *button (to take it out of the* **Selected Fields** *area), then click* **Next >**.



Report Wizard

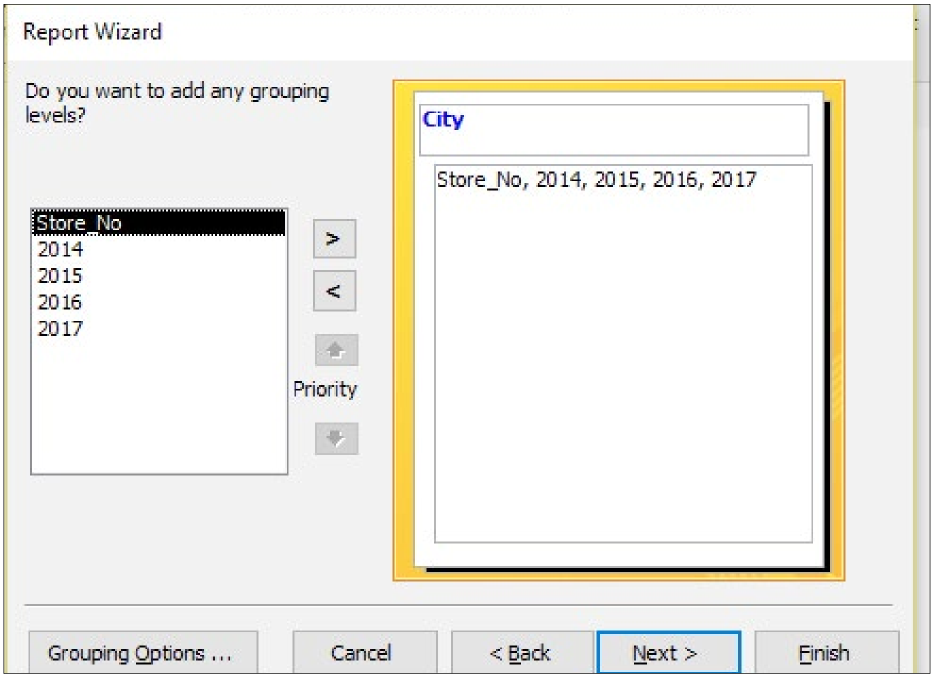
Do you want to add any grouping levels?

City, Store_No, 2014, 2015, 2016, 2017

Figure 8.4

In the next screen of the reporting wizard, it asks if we want to do any grouping (for subtotaling). We want to group by City.

11. *Click on* **City**, *then click the* > *button.*



Report Wizard

Do you want to add any grouping levels?

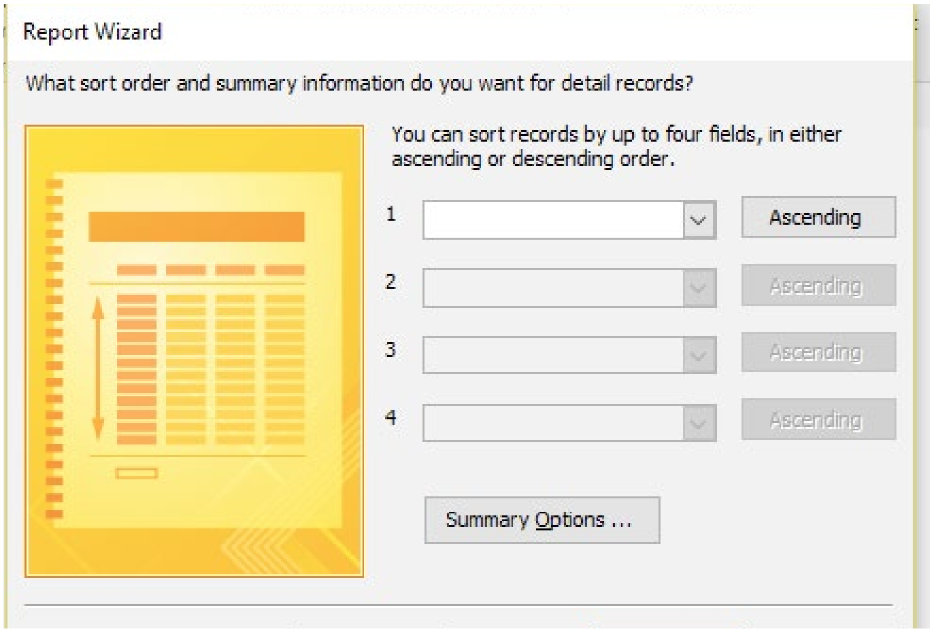
City

Store_No, 2014, 2015, 2016, 2017

Figure 8.5

12. *Click* **Next >**.

The next screen in the wizard asks you to order (or sort) the records in the report. You want to order by store number.



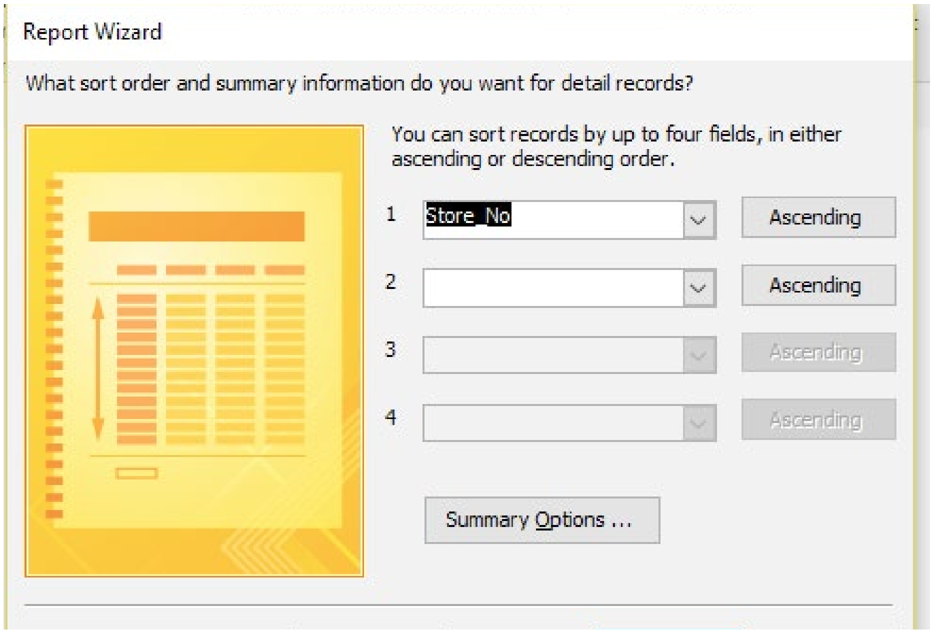
Report Wizard

What sort order and summary information do you want for detail records?

You can sort records by up to four fields, in either ascending or descending order.

Figure 8.6

13. *Click on the first drop-down arrow, choose* **Store_No**, *make sure the* **Order by** *button reads* **Ascending**.



Report Wizard

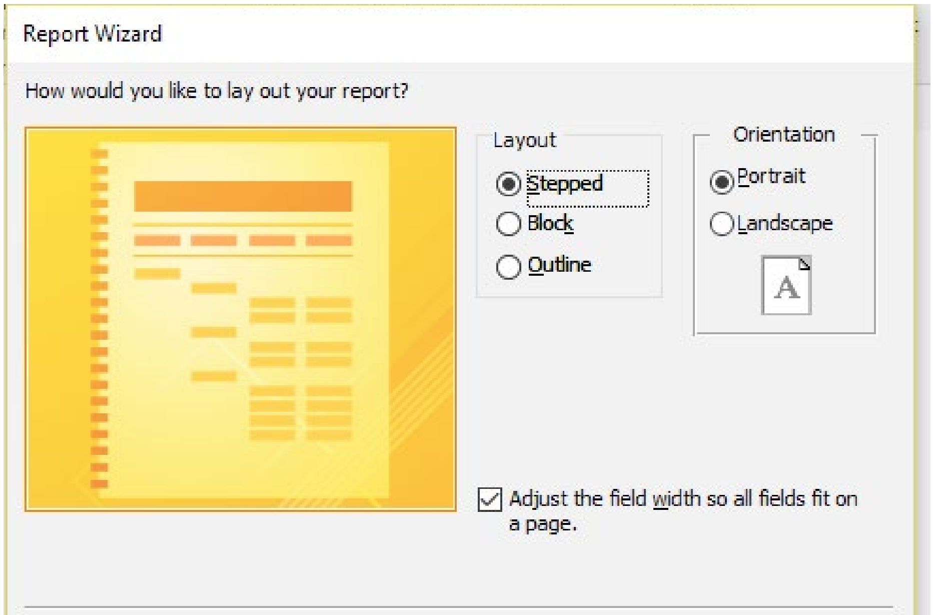
What sort order and summary information do you want for detail records?

You can sort records by up to four fields, in either ascending or descending order.

1 Store_No — Ascending

Figure 8.7

14. *Click* **Next** >.



Report Wizard

How would you like to lay out your report?

Layout

● Stepped

○ Block

○ Outline

Orientation

● Portrait

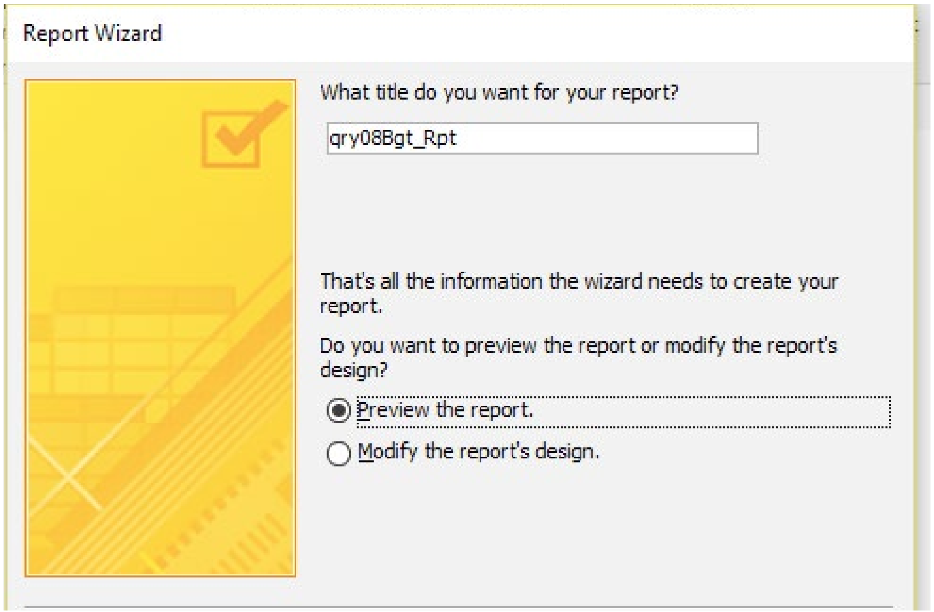
○ Landscape

☑ Adjust the field width so all fields fit on a page.

Figure 8.8

The next screen asks you what layout you want to use. As you click on the various options, the graphic to the left shows you generally what it looks like. In this report, we'll use the Stepped option. This report is probably small enough for everything to fit on a Portrait report, so let's leave the Portrait radio button checked under the Orientation section.

15. *Make sure the* **Stepped** *and* **Portrait** *radio buttons are checked, and click* **Next** >.



Report Wizard

What title do you want for your report?

qry08Bgt_Rpt

That's all the information the wizard needs to create your report.

Do you want to preview the report or modify the report's design?

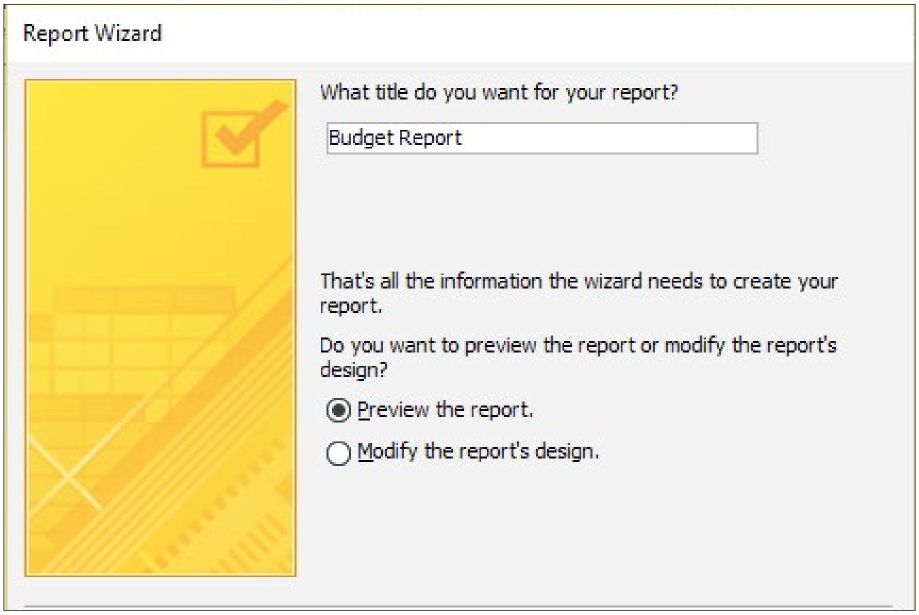
● Preview the report.

○ Modify the report's design.

Figure 8.9

The last screen of the wizard asks you what you want to name the report. Let's pick something real innovative, like "Budget Report".

16. *Type* ***Budget Report*** *as the report title, make sure the* **Preview the report** *radio button is selected.*



Report Wizard

What title do you want for your report?

Budget Report

That's all the information the wizard needs to create your report.

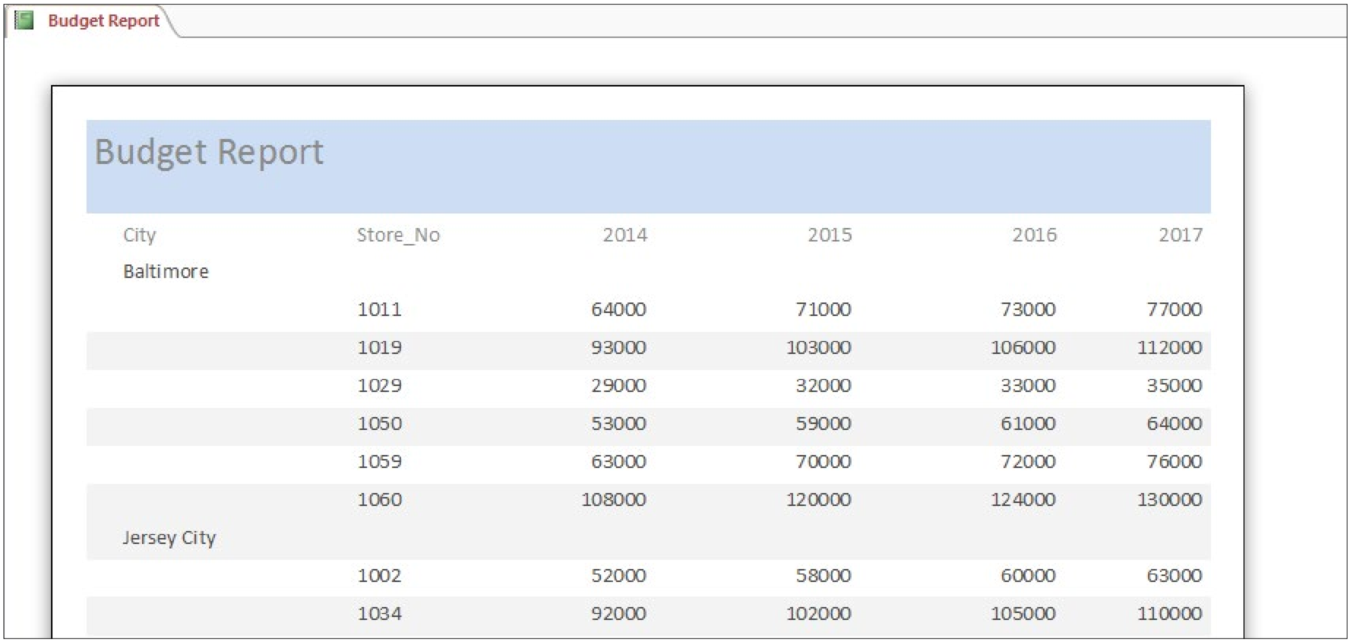
Do you want to preview the report or modify the report's design?

● Preview the report.

○ Modify the report's design.

Figure 8.10

17. *Click* **Finish.**



Budget Report

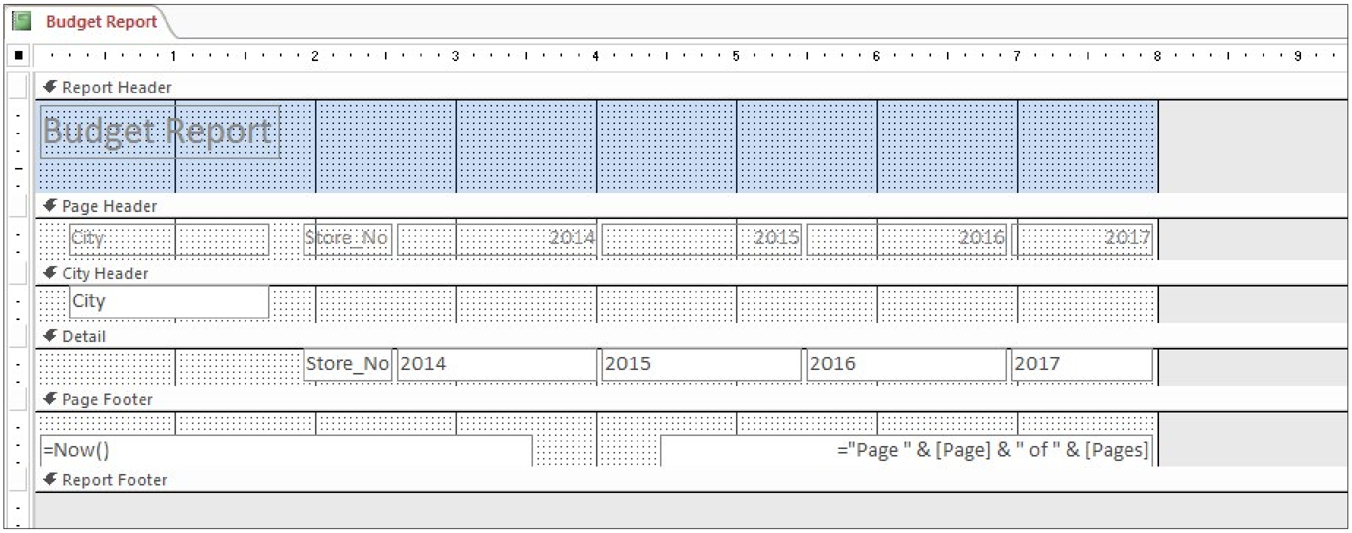
City	Store_No	2014	2015	2016	2017
Baltimore					
	1011	64000	71000	73000	77000
	1019	93000	103000	106000	112000
	1029	29000	32000	33000	35000
	1050	53000	59000	61000	64000
	1059	63000	70000	72000	76000
	1060	108000	120000	124000	130000
Jersey City					
	1002	52000	58000	60000	63000
	1034	92000	102000	105000	110000

Figure 8.11

Report Design View

It's a simple-looking report. Not much pizzazz. Not much formatting. If you click on the page selectors at the bottom of the page, you'll see that most of the report fits on one page. In the Report Wizard on the page where we selected Stepped and Portrait layouts, there was a check box that said, "*Adjust the field width so all fields fit on a page*". This kept all the information it can on one page by adjusting margin widths, but the report still doesn't tell us much. If you turn it in looking like this, you may not have a job as a reporting guru for very long. Let's look at the Design View of the report and clean it up.

18. *Right-click anywhere in the report and choose* **Design View**.



=Now()

="Page " & [Page] & " of " & [Pages]

Figure 8.12

Once you click on Design View, the Design Grid of the report opens. You will probably see the ***Field List*** or Property Sheet dialog box as well. It may look complex, but once you start working with the report in Design View, you'll soon see how it works and it will become second nature to you. The Design Grid of the report looks very similar to the Design View of a form, and is split into sections. There are two types of sections in the report Design Grid: standard and custom. A standard section is one that is available in every report. A custom section is one that you define based on data in your data source (query or table). Let's discuss each of the sections.

Report Header (standard): The first section in the report design is the Report Header. This section of the report appears on only the first page of the report. You can put things here like the name of the company, the report name, and the as-of date. I personally like for that information to repeat on every page, and not just appear on the first page of the report, so I usually collapse this section and place the report information in the Page Header section.

Page Header (standard): The Page Header section of the report is the text or data that appear at the top of every page. This is where I like to put information about the report, such as the name of the company, title of the report, and column headers so they will repeat on every page.

City Header (custom): Not every report you write will have a category for city, so this section of the report is a custom section. Typically, I make this section of the report include only the title of the data group (in this case, city).

Detail (standard): This is where the meat of the report resides. The lowest level of data in the report appears in this section.

Page Footer (standard): This section usually contains subtotals and the corresponding labels. I like to include things like the date the report was run and the author or group that created the report. This way, when any page of the printed report goes outside of my department, anyone looking at it will know who to talk to about the report. Anything in the Page Footer section will print on every page of the report.

Report Footer (standard): This section also usually contains subtotals and the corresponding labels. It also contains controls that appear only on the last page of the report.

Rulers: Across the top of the grid and along the left side of the grid is the Report Ruler. Although rulers are not included in a "section" of the Design View, they warrant mentioning here. Rulers in reports work the same as rulers in forms. Rulers appear in the design grid so we can estimate where the various sections, labels, and text boxes are placed in relation to the page.

Report Labels and Text Boxes

Within a standard Access report, you can have information contained in controls like labels and text boxes. As with forms, labels in reports are simply text strings. Text boxes can contain data from a data source (bound), or unbound values or calculations. In our example, the Budget Report box in the Report Header section is a label. Other labels in our example include the City, Store_No, 2014, 2015, 2016, and 2017 boxes in the Page Header section. Label boxes appear on the report just as they do in the design grid.

As I mentioned earlier, I don't use the Report Header section of the report design grid much. I prefer to have the entire title of the report appear on every page, which is done in the Page Header section. Let's move the Budget Report label into the Page Header section of the report. To do that, we first have to create a space big enough in the Page Header section to hold the label.

19. *Place your cursor over the top of the* **City Header** *section in the* **Design Grid**. *It will turn to a vertical up and down arrow once the header bar is clicked.*
20. *Click and drag that section down to approximately the bottom of the* **Detail** *gray bar and release.*

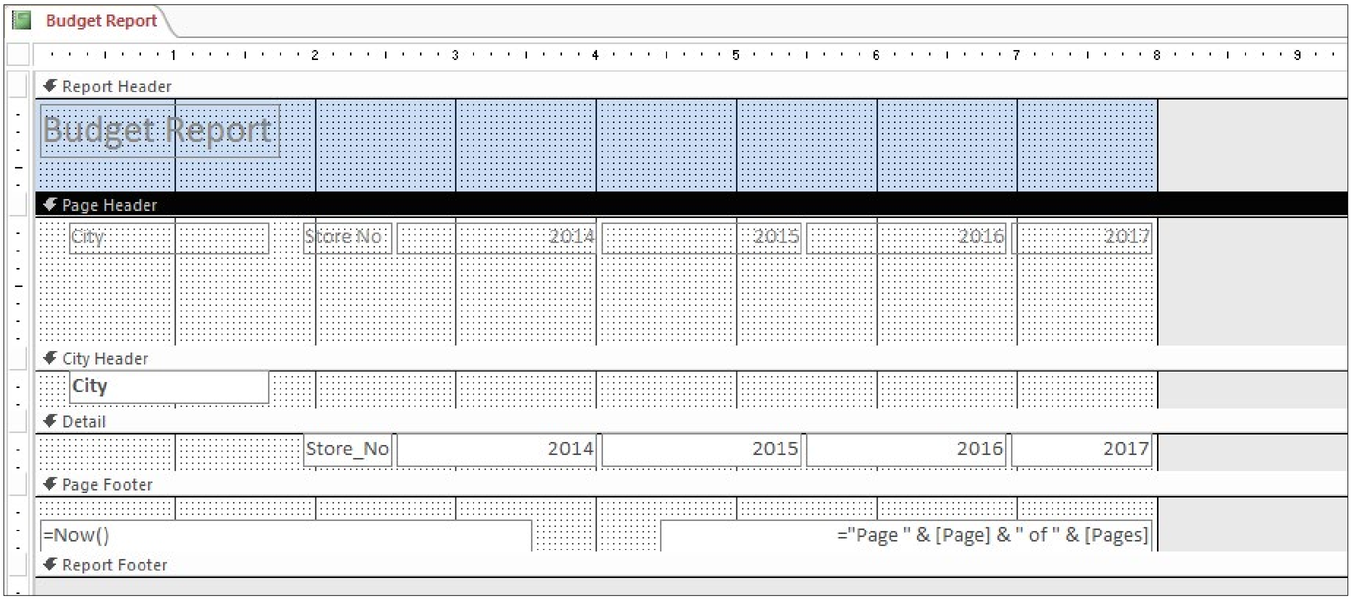


Figure 8.13

Now there's a little more room in the Page Header section to work with. Next you will reposition objects in the Report and Page Header sections.

21. *Position your cursor in the* **Left Ruler***, just under the* **Page Header** *gray bar. Your cursor will turn to a right arrow.*
22. *Click and hold and you will see a horizontal line going through all of the label boxes at the top of the* **Page Header** *section. Drag your mouse down a little to select all text boxes and the horizontal line and release.*

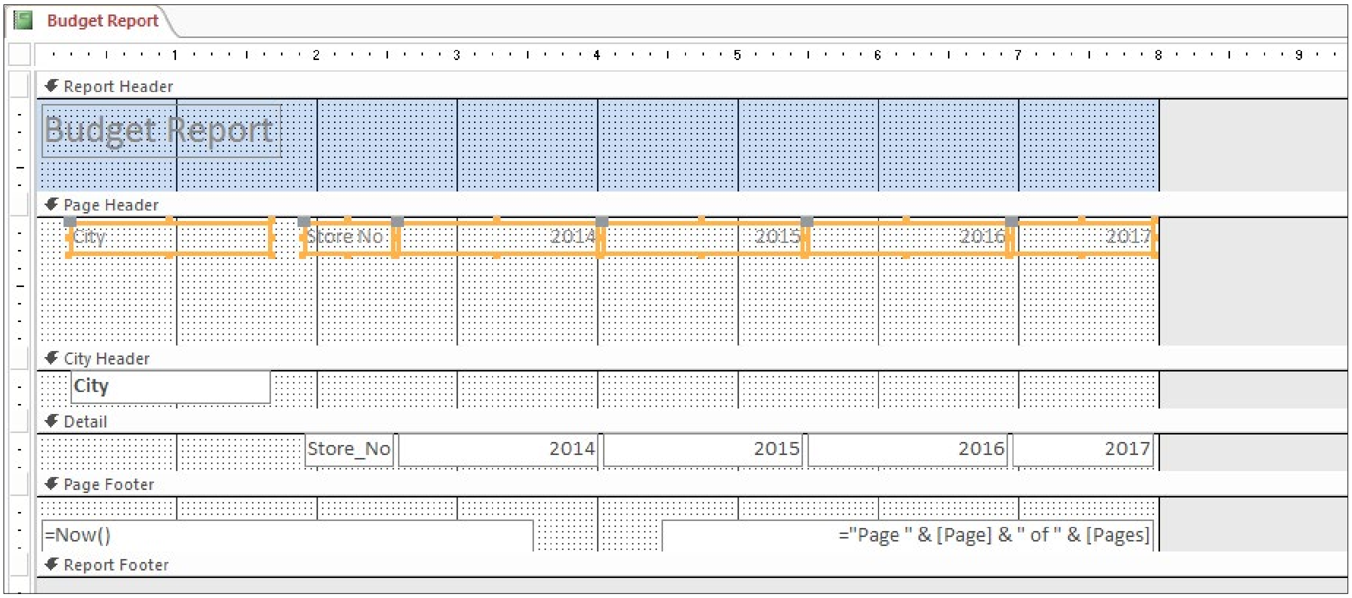


Figure 8.14

All of the label boxes are selected in the Page Header section.

> ***Note**: If you prefer, you can click on the arrow keys on your keyboard to move the selection.*

23. *Click and hold on any of the selected items and drag the entire group down to the bottom section of the* **Page Header** *section and release.*

As you drag the entire selection down, you may notice that the edges of the label boxes remain in line with the small dots that appear in the grid. This is because the Snap to Grid option is enabled. I really like to have this feature turned on, because it makes it much easier to align the objects. Unfortunately, Snap to Grid in Access 2016 is somewhat hidden. If your report doesn't have Snap to Grid turned on, click on the Arrange tab under the Report Design Tools contextual tab, click on the Size/Space icon in the Sizing and Ordering group, and click on Snap to Grid. Let's continue editing the design of the report.

24. *Click on the* **Budget Report** *label box and drag it down to the top of the* **Page Header** *section and release. Make sure that the left edge of the* **Budget Report** *label box is in line with the* **City** *label box.*

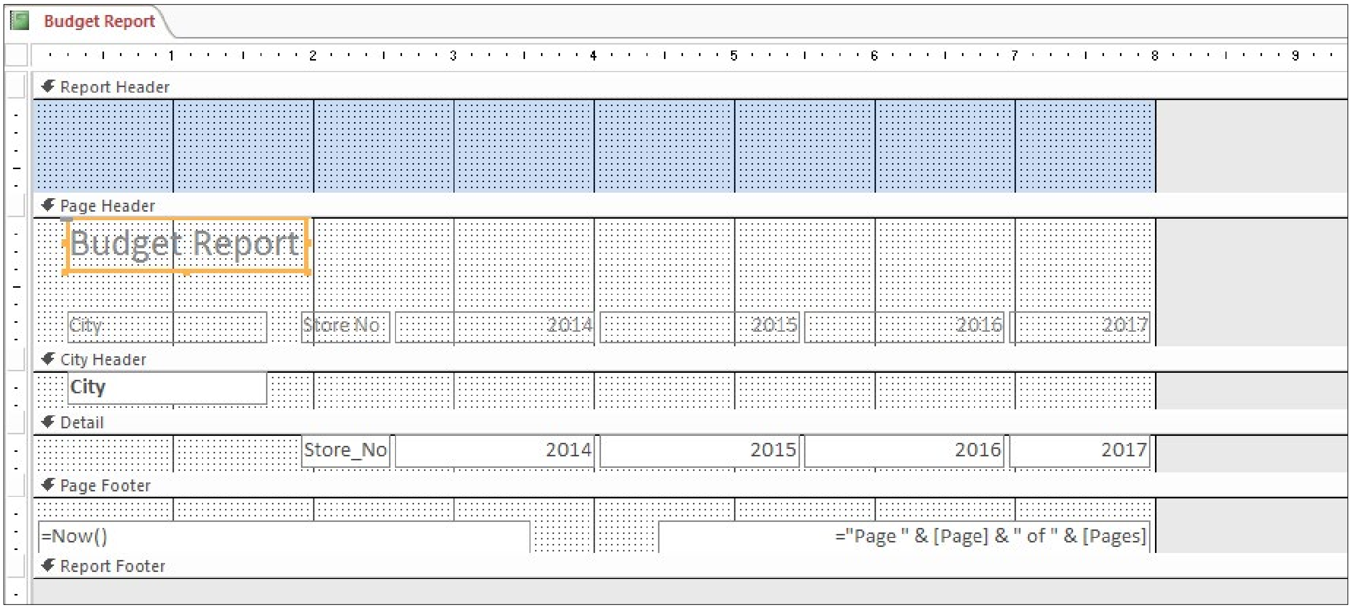


Figure 8.15

Now we're going to disable the Report Header section. You do this by dragging up the gray box at the top edge of the Page Header.

25. *Click on top edge of the* **Page Header** *section bar and drag it up until the blue space disappears.*

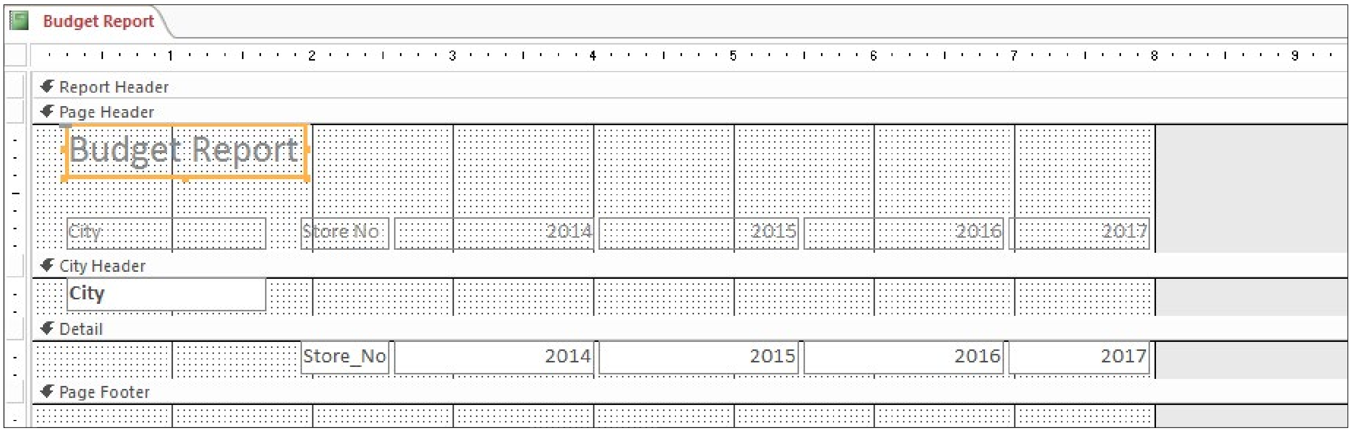
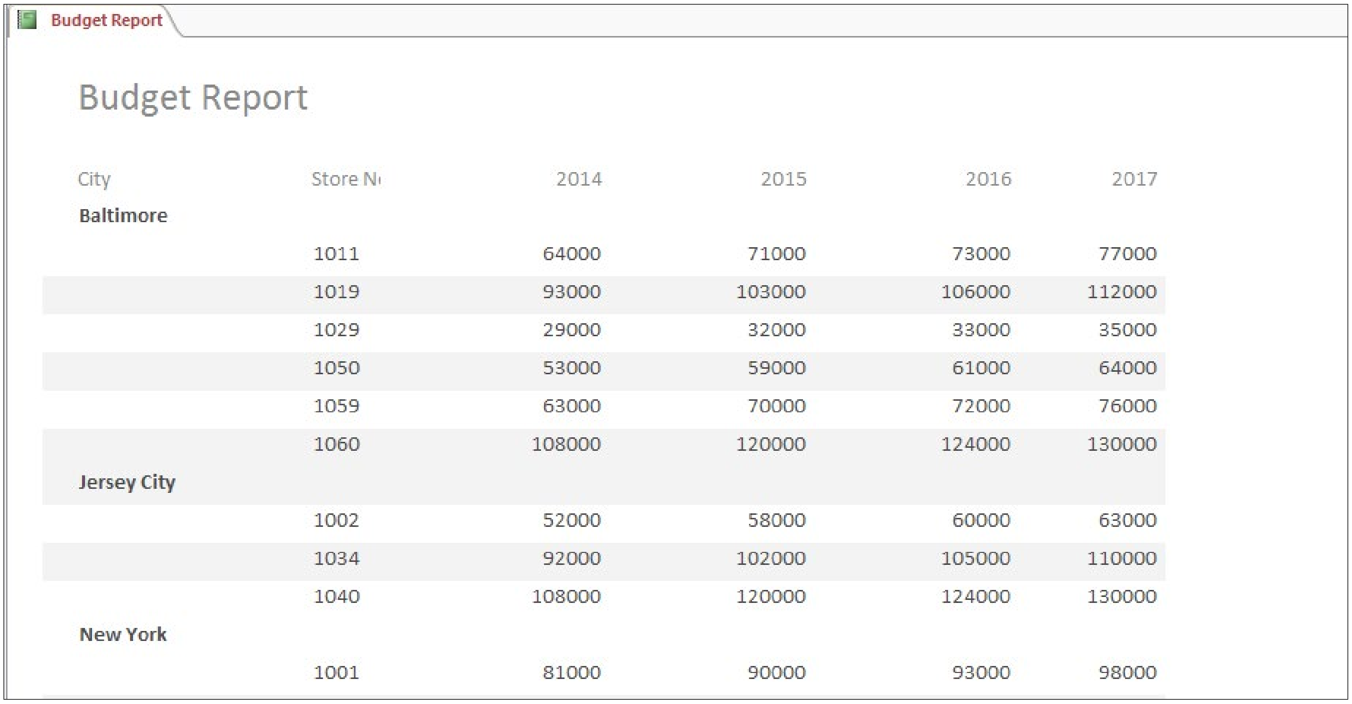


Figure 8.16

> ***Note**: Sometimes when working with reports in this manner, there are small objects that are almost invisible, and which, if they exist, will not allow you to close the* **Report Header** *(or any other section). Therefore, if you try to close a section and Access doesn't allow it, you probably have an object somewhere in the section you're trying to close.*

When you click on the View icon to see the report, the top of the report should look like the figure below.

26. *Click on the* **Design** *tab, then click on the* **View** *icon to view the report.*


Budget Report

Budget Report

City	Store N	2014	2015	2016	2017
Baltimore					
	1011	64000	71000	73000	77000
	1019	93000	103000	106000	112000
	1029	29000	32000	33000	35000
	1050	53000	59000	61000	64000
	1059	63000	70000	72000	76000
	1060	108000	120000	124000	130000
Jersey City					
	1002	52000	58000	60000	63000
	1034	92000	102000	105000	110000
	1040	108000	120000	124000	130000
New York					
	1001	81000	90000	93000	98000

Figure 8.17

As I explain how to do different commands to edit and design the report, feel free to click on the View icon in go to Design View to see how the report looks, and then go back to Report View. To go to Design View, either right-click in Print Preview and choose Design View, or click on the View drop-down menu and choose Design View.

The Store_No label looks screwy because 1) some of the label text is a bit crowded, and 2) most reports don't show an underscore character for a space. You can edit that out easily enough.

27. *In* **Design View***, click on the* **Store_No** *label, take out the underscore and replace it with a space.*
28. *Expand the* **Store No** *label so that the entire phrase is visible then click somewhere outside of the label to deselect it.*

> ***Review Questions****: It is now time to complete the hands-on Review Questions. Log on to www.ExcelCEO.com with your Email and Password, click on the* **Access 2016 and SQL Review Questions, Chapter 8, Section 1 of 2** *option and complete the review questions.*

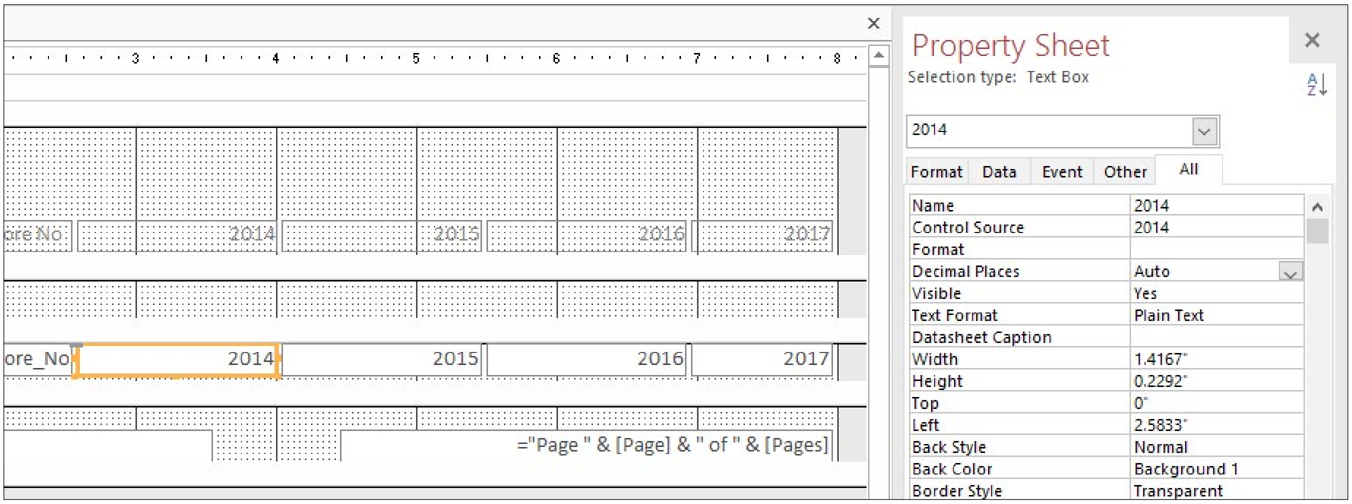
Object Properties

So far in this report, we've been working mostly with label boxes. Now we'll start working with the data contained in the text boxes. The budget numbers need some formatting. You do formatting in the Property Sheet of each item, just like you do in forms. Let's format the budget numbers as Standard, zero decimal places.

1. *Click on the* **2014** *text box (not the label box).*
2. *Click on the* **Property Sheet** *icon (if the Property Sheet is not already displayed).*

The Property Sheet dialog box that contains all of the information about the text box selected appears. At some time, scroll through the various options available, as there is no way I can review all of them in this course. For now, we'll just change the formatting.

3. *Click on the* **Format** *tab.*



Property Sheet

Selection type: Text Box

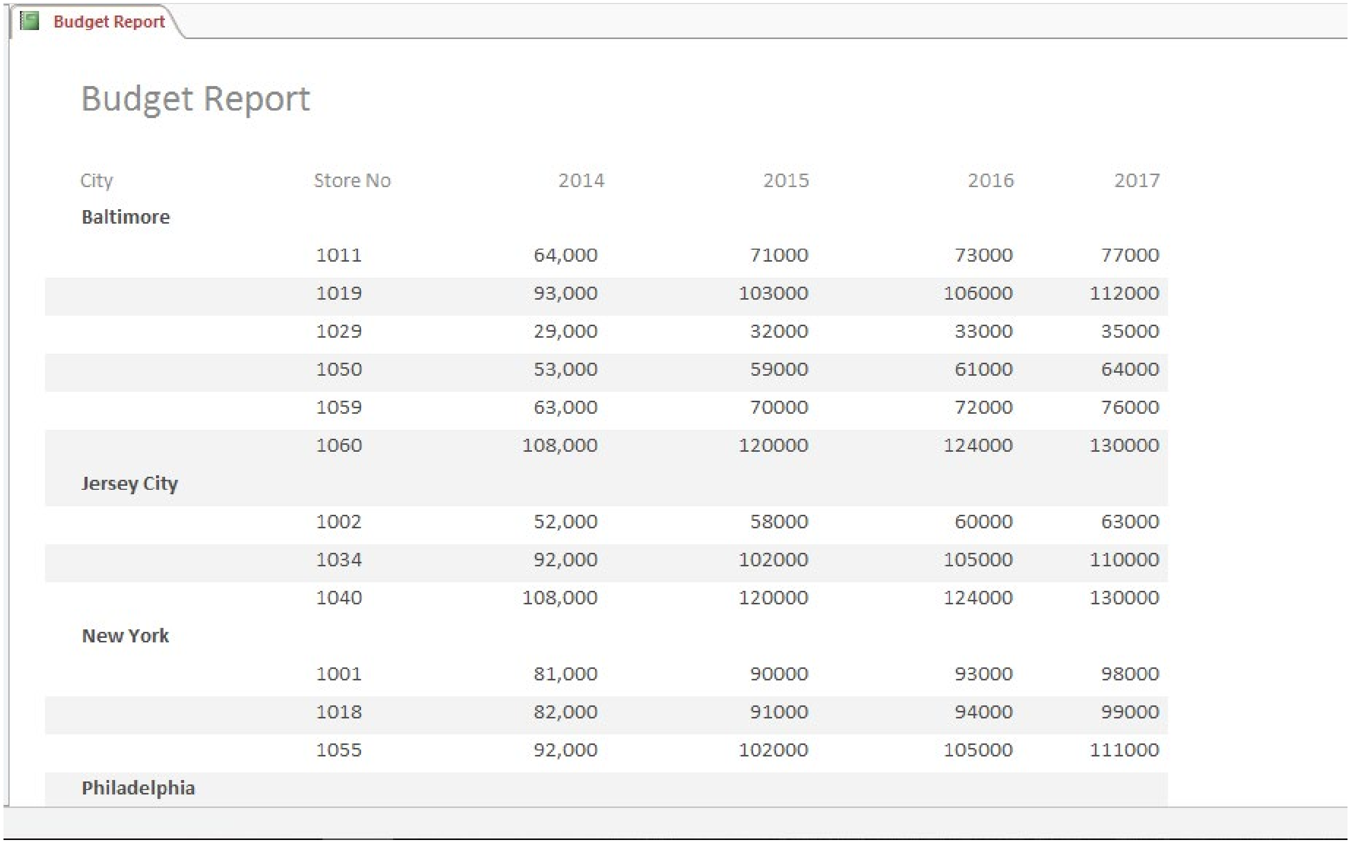
2014

Format | Data | Event | Other | All

Name	2014
Control Source	2014
Format	
Decimal Places	Auto
Visible	Yes
Text Format	Plain Text
Datasheet Caption	
Width	1.4167"
Height	0.2292"
Top	0"
Left	2.5833"
Back Style	Normal
Back Color	Background 1
Border Style	Transparent

Figure 8.18

4. *Click in the* **Format** *drop-down menu and choose* **Standard**.
5. *In the* **Decimal Places** *box, choose or type* **0**.
6. **View** *the report.*



Budget Report

City	Store No	2014	2015	2016	2017
Baltimore					
	1011	64,000	71000	73000	77000
	1019	93,000	103000	106000	112000
	1029	29,000	32000	33000	35000
	1050	53,000	59000	61000	64000
	1059	63,000	70000	72000	76000
	1060	108,000	120000	124000	130000
Jersey City					
	1002	52,000	58000	60000	63000
	1034	92,000	102000	105000	110000
	1040	108,000	120000	124000	130000
New York					
	1001	81,000	90000	93000	98000
	1018	82,000	91000	94000	99000
	1055	92,000	102000	105000	111000
Philadelphia					

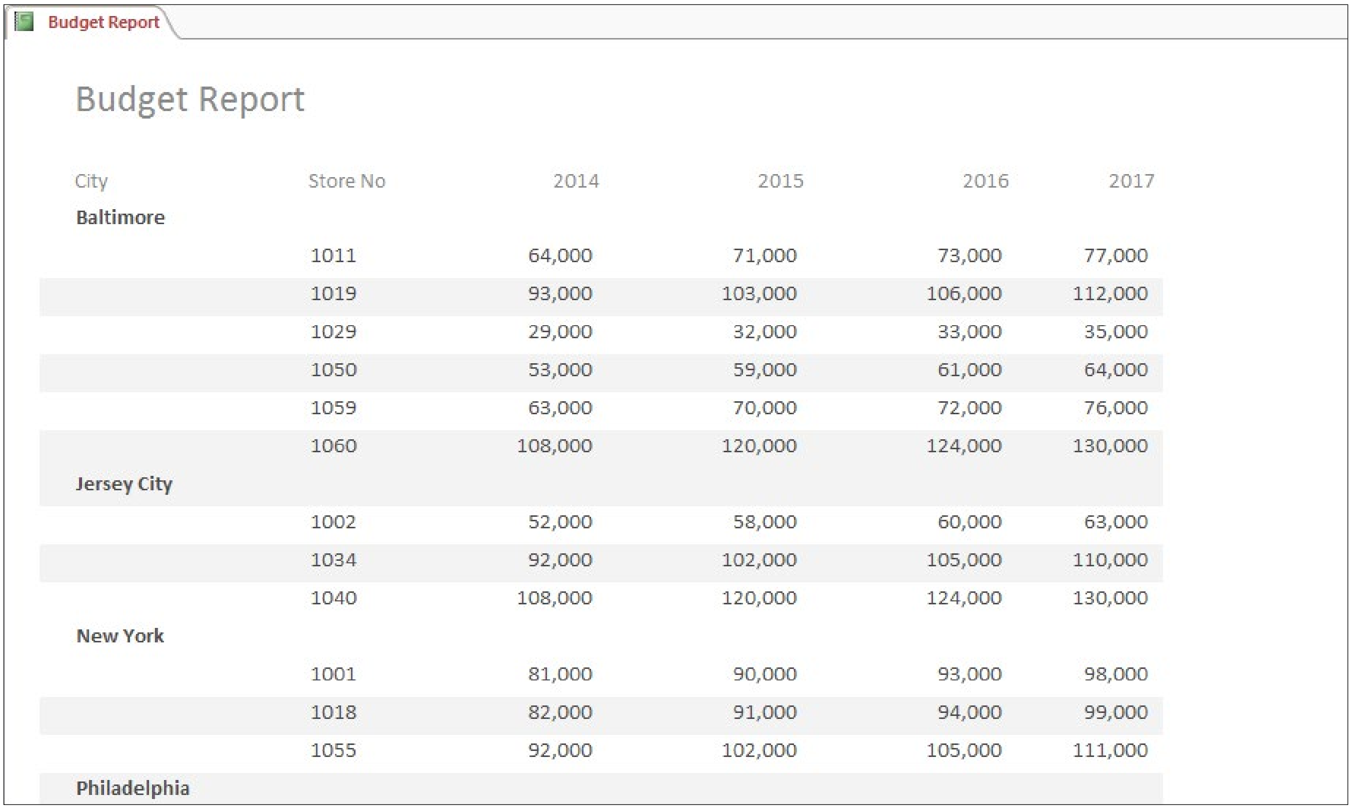
Figure 8.19

See how the budget numbers under the year 2014 are in the new format? All we have to do now is format the others in the same way. Keep in mind that you don't have to do the formatting one box at a time. You can select numerous boxes and do the formatting all at once.

7. *In* **Design View**, *click on the* **2015** *text box, hold down the* **[Shift]** *key, and click on the* **2016** *and* **2017** *text boxes.*

> ***Tip****: When you click on the* **2016** *and* **2017** *text boxes, make sure you don't accidentally move the text boxes to be out of alignment with the others. That happens to me a lot, especially when I try to do it really fast.*

8. *With the* **2015**, **2016**, *and* **2017** *text boxes selected and the* **Property Sheet** *box still open, choose* **Standard** *format and* **zero decimal places**.
9. **View** *the report.*



Budget Report

City	Store No	2014	2015	2016	2017
Baltimore					
	1011	64,000	71,000	73,000	77,000
	1019	93,000	103,000	106,000	112,000
	1029	29,000	32,000	33,000	35,000
	1050	53,000	59,000	61,000	64,000
	1059	63,000	70,000	72,000	76,000
	1060	108,000	120,000	124,000	130,000
Jersey City					
	1002	52,000	58,000	60,000	63,000
	1034	92,000	102,000	105,000	110,000
	1040	108,000	120,000	124,000	130,000
New York					
	1001	81,000	90,000	93,000	98,000
	1018	82,000	91,000	94,000	99,000
	1055	92,000	102,000	105,000	111,000
Philadelphia					

Figure 8.20

10. *In* **Design View**, *select the* **Store_No** *text box as well as all of the* **Year** *text boxes in the* **Detail** *section, and then select all of the* **label boxes** *in the* **Page Header** *section with the exception of the* **City** *label.*
11. *Move all of those objects as a group to the left to where the left side of the* **Store_No** *text box and label are aligned with the* **1"** *mark.*

12. *Move the top section bar of the of the* **Page Footer** *section up as far as you can to take out that space in the* **Detail** *section.*

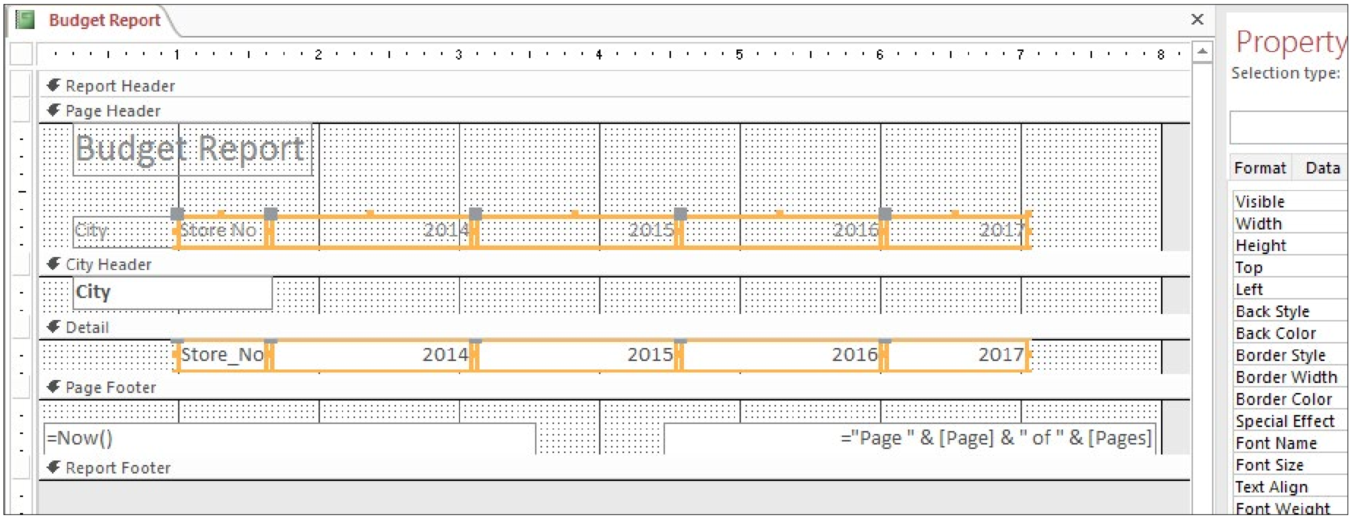
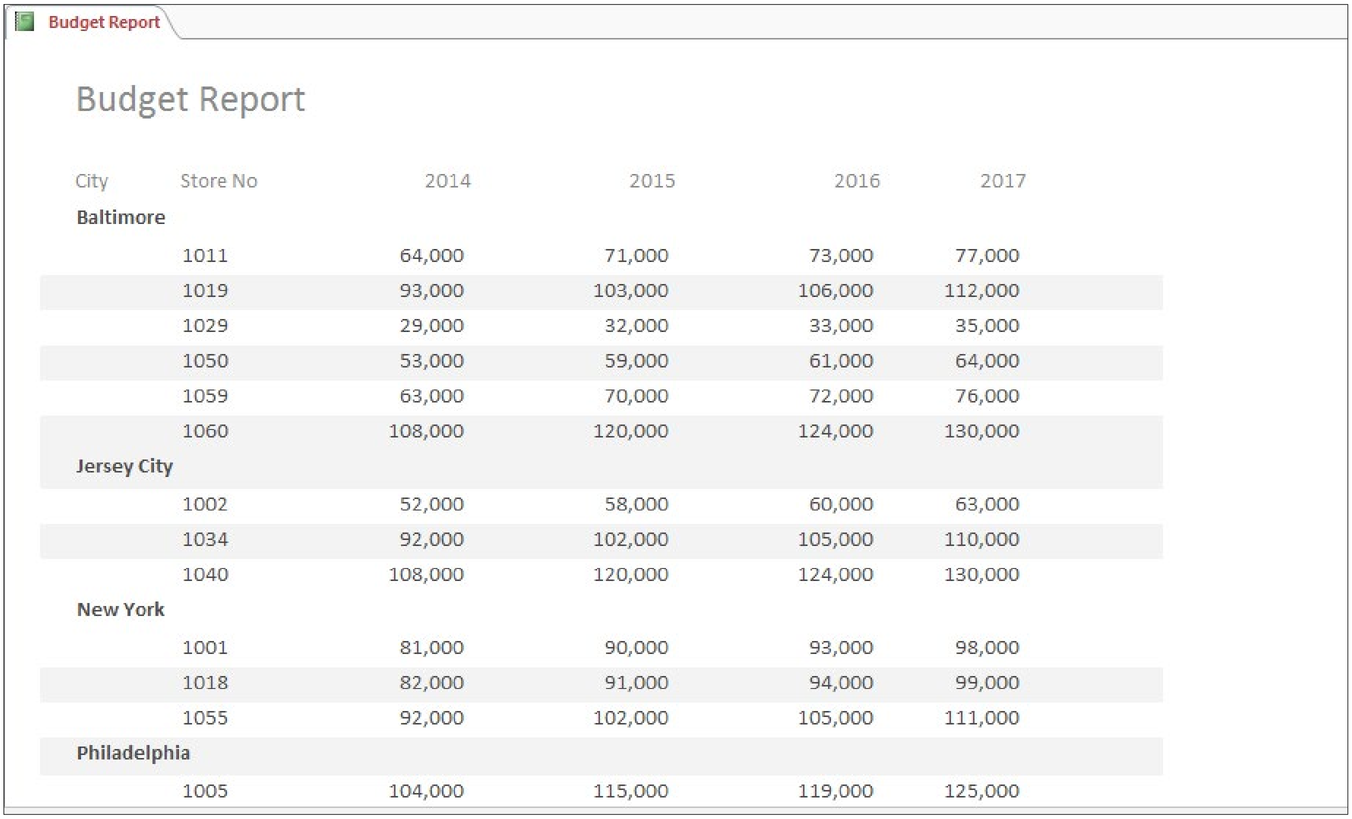


Figure 8.21

13. **View** *the report.*



Budget Report

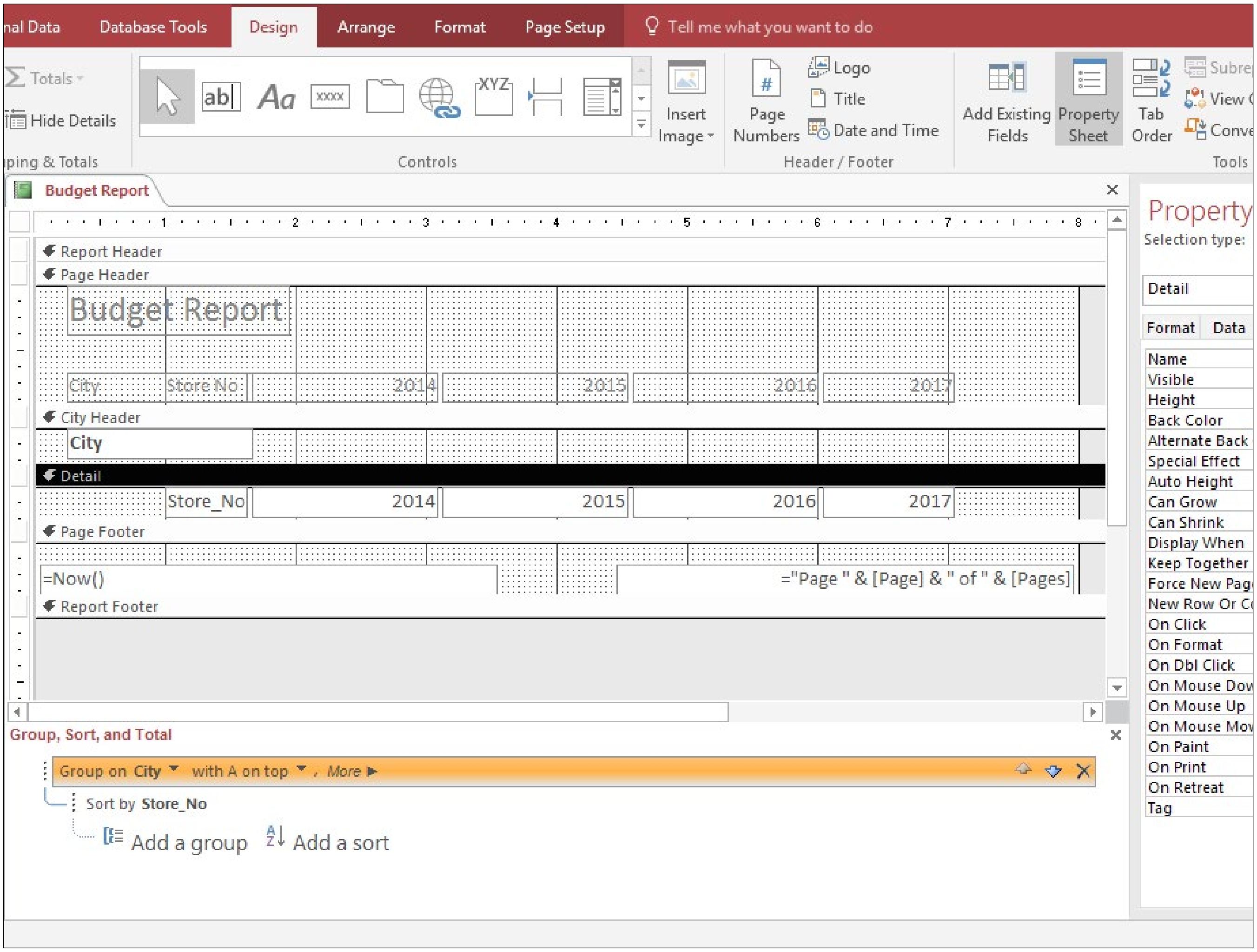
City	Store No	2014	2015	2016	2017
Baltimore					
	1011	64,000	71,000	73,000	77,000
	1019	93,000	103,000	106,000	112,000
	1029	29,000	32,000	33,000	35,000
	1050	53,000	59,000	61,000	64,000
	1059	63,000	70,000	72,000	76,000
	1060	108,000	120,000	124,000	130,000
Jersey City					
	1002	52,000	58,000	60,000	63,000
	1034	92,000	102,000	105,000	110,000
	1040	108,000	120,000	124,000	130,000
New York					
	1001	81,000	90,000	93,000	98,000
	1018	82,000	91,000	94,000	99,000
	1055	92,000	102,000	105,000	111,000
Philadelphia					
	1005	104,000	115,000	119,000	125,000

Figure 8.22

The Grouping and Sorting Section

Now the report is starting to look better. But notice that the report doesn't give subtotals by city. It just has the budget amount for every store. Subtotals are typically done in the Footer section of the report, although they can be done in a header section as well. If you look in the Design Grid, there is no footer for City. We need a City footer so that we can create subtotals for each City. You can create a new section by using the ***Grouping and Sorting*** dialog box.

1. *In* **Design View**, *click on the* **Group & Sort** *icon in the* **Grouping & Totals** *section of the* **Design** *tab.*



```
=Now()
="Page " & [Page] & " of " & [Pages]
```

Group on City ▼ with A on top ▼ , More ▶

Sort by Store_No

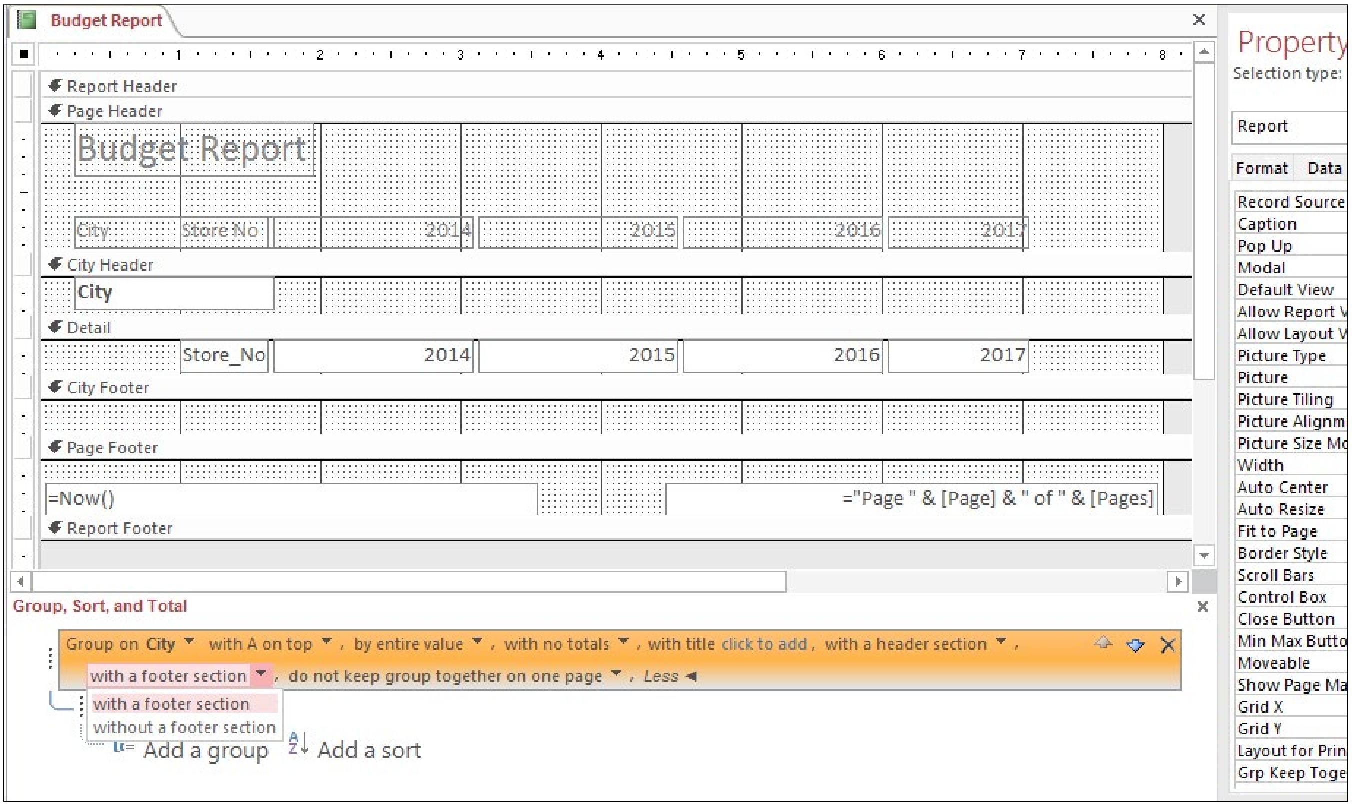
Figure 8.23

The Group, Sort, and Total section appears below the Design Grid of the report. Access already knows that the report is grouped and/or sorted on the City and/or Store_No levels, and provides this section for you to refine it. To create a City Footer, all we have to do is expand the City section.

2. *Click on the* More ▶ *icon in the* **Group on City** *section.*

3. *Click on the drop-down arrow next to* **without a footer section** *and choose* **with a footer section**.

As soon as you make the change, the City Footer section appears in the Design Grid of the report.



Group on City ▾ with A on top ▾ , by entire value ▾ , with no totals ▾ , with title click to add , with a header section ▾ , with a footer section ▾ , do not keep group together on one page ▾ , Less ◂

with a footer section
without a footer section

Figure 8.24

4. *Close the* **Group, Sort, and Total** *section.*

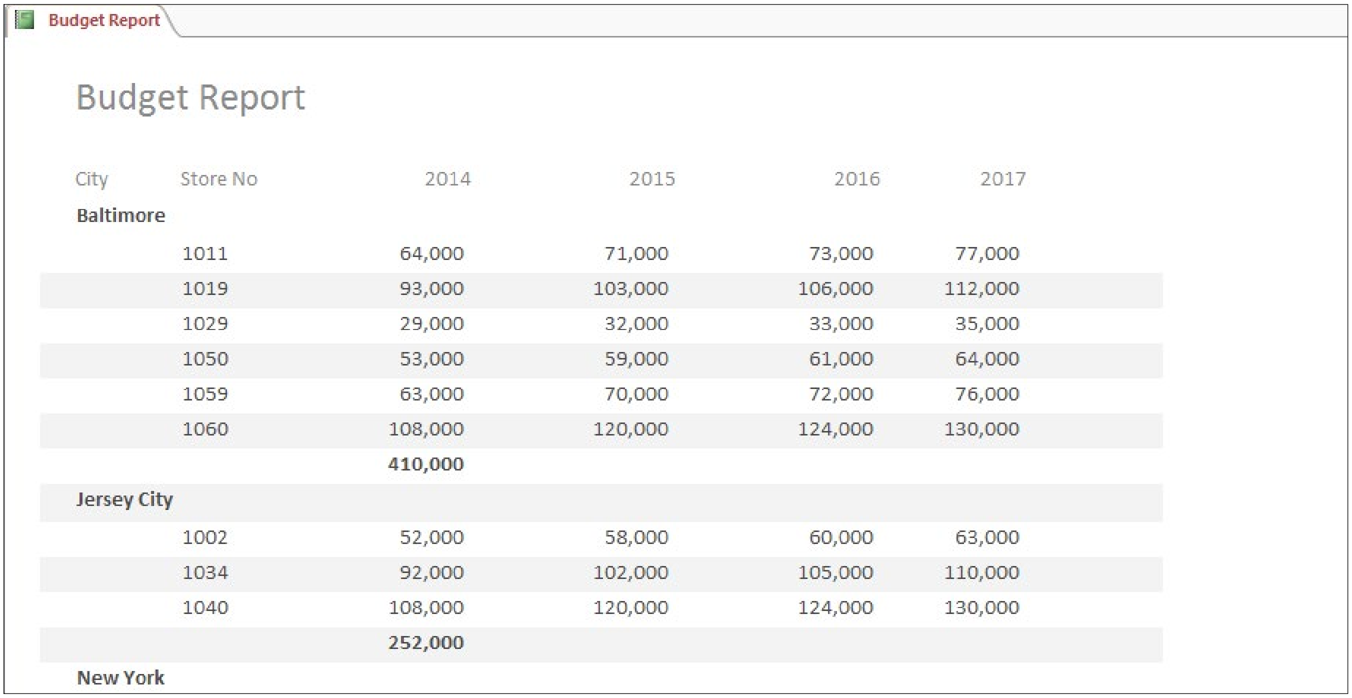
Creating Subtotals

Now, you need to pay really close attention to the next part. I want to make sure you understand this very well. You will now create budget subtotals by city. In the following exercise, we will copy the text boxes 2014, 2015, 2016, and 2017 from the Detail section to the City Footer section. Then you will go into each text box and write a SUM() function around it.

5. *Click on the* **2014** *text box and* **copy** *it ([Ctrl]+c or right-click and choose Copy). You may have to click somewhere in the grid to deselect the selected objects.*
6. *Click on the* **City Footer** *section bar and* **paste** *the text box ([Ctrl]+v or right-click and choose Paste).*

A 2014 text box appears in the City Footer section of the report, aligned all the way to the top and left.

7. *Edit the text in the* **2014** *text box in the* **City Footer** *section to read:* **=*Sum([2014])*** *and press* **[Enter]**.
8. *With that text box selected, place your cursor over an edge and move it until the cursor becomes a crosshair.*
9. *Click and drag the* **Sum([2014])** *text box to be aligned under the* **2014** *text box in the* **Detail** *section.*
10. *Press the* **B** *(Bold) icon in the* **Font** *group of the* **Format** *contextual tab.*
11. **View** *the report.*



Budget Report

Budget Report

City	Store No	2014	2015	2016	2017
Baltimore					
	1011	64,000	71,000	73,000	77,000
	1019	93,000	103,000	106,000	112,000
	1029	29,000	32,000	33,000	35,000
	1050	53,000	59,000	61,000	64,000
	1059	63,000	70,000	72,000	76,000
	1060	108,000	120,000	124,000	130,000
		410,000			
Jersey City					
	1002	52,000	58,000	60,000	63,000
	1034	92,000	102,000	105,000	110,000
	1040	108,000	120,000	124,000	130,000
		252,000			
New York					

Figure 8.25

> ***Note***: *If you do not see the subtotal below each group where you placed it using* **Report View**, *try clicking in that area to select the subtotal. To verify it is there, you can also view the report in* **Print Preview** *mode.*

It is important to put the brackets around 2014 in your formula so Access knows 2014 is the field and not the number 2014. If you add up the six values under the 2014 column for Baltimore, you will see that they add up to $410,000. Now you just have to copy that 2014 formula over to change 2015, 2016, and 2017.

12. *In* **Design View**, *make sure the* **=Sum([2014])** *text box is selected.*
13. *Copy and paste the text box in the* **City Footer** *section.*
14. *Change the new text box to read* ***2015***.
15. *Do the same for* ***2016*** *and* ***2017***.
16. *Resize all of the text boxes that contain amounts for* **2014** - **2017** *in the* **Detail** *and* **City**

Footer *sections to have a width of* **1.2"**. *(You can do this by selecting all of the text boxes and changing the* **Width** *property in the* **Property Sheet** *window).*

17. *Make the* **2014 - 2017** *Page Header labels be the same width as the* **text boxes**.
18. *Align the text boxes under their respective years.*
19. *Move the* **Page Footer** *section bar up a bit to take out some of the white space.*

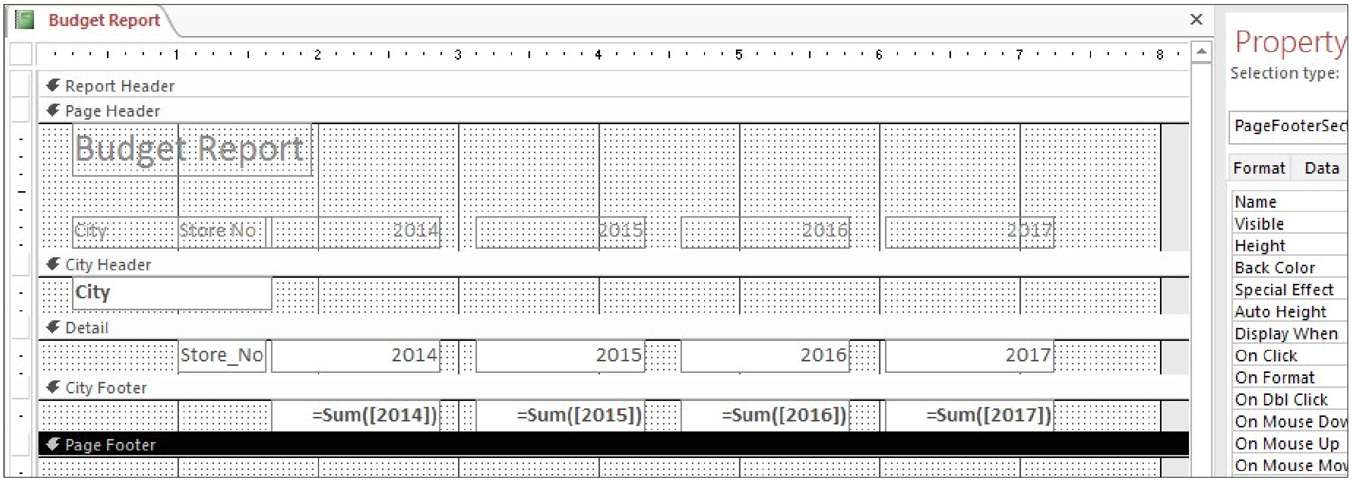


Figure 8.26

Now you need to create a label indicating the total for each city. All you need to do here is to write a concatenated formula like you learned in the Excel course (assuming you took the Excel course. If you didn't, I highly recommend it). To do this, you need to use a text box, not a label. Remember, concatenation is a formula, and you write formulas in reports within text boxes, not labels. We'll use one of the text boxes we created in the City Footer section as a starting point.

20. *Copy and paste the* **Sum([2014])** *text box (you could actually copy ANY text box or create a new one) in the* **City Footer** *section.*
21. *Edit the formula in the new text box to read:* ***=[City]&" Total"***
22. *Expand the* **City Total** *text box as shown in the figure below.*
23. *Align the text box over to the left under the* **City** *label in the* **City Header** *section.*

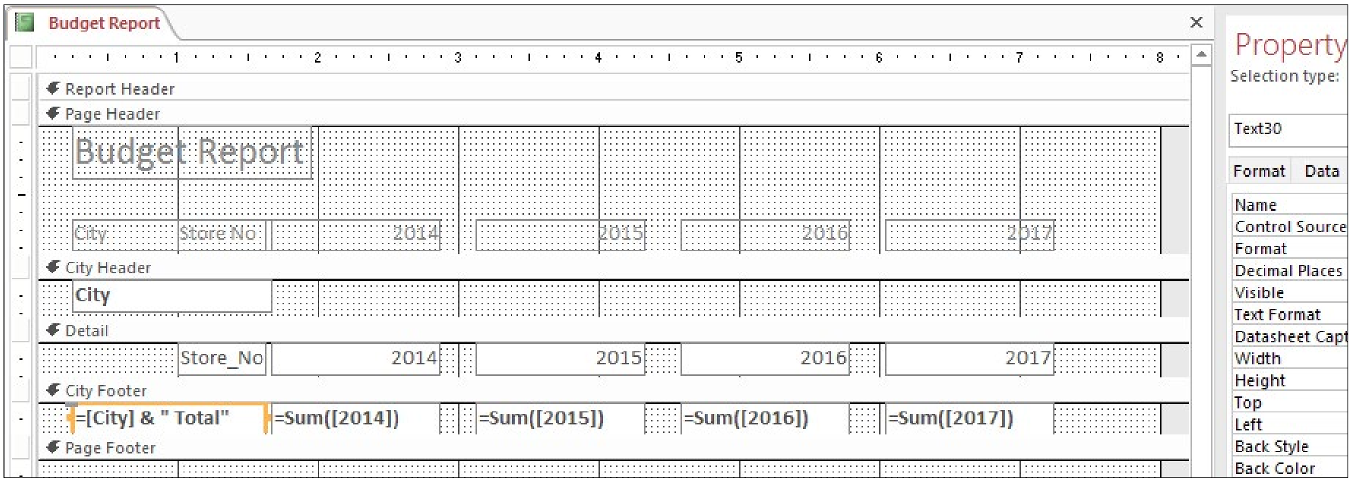
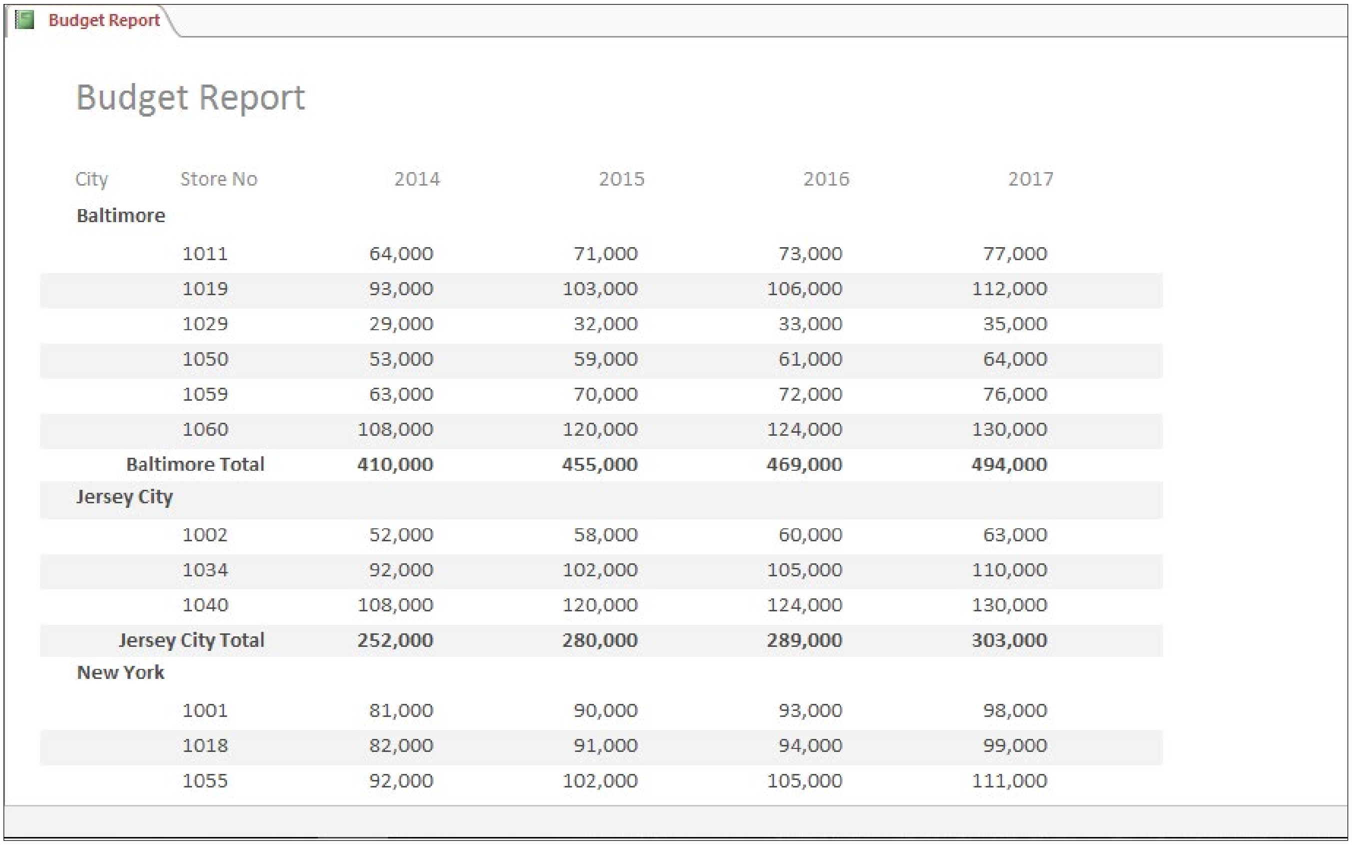


Figure 8.27

24. **View** *the report.*


Budget Report

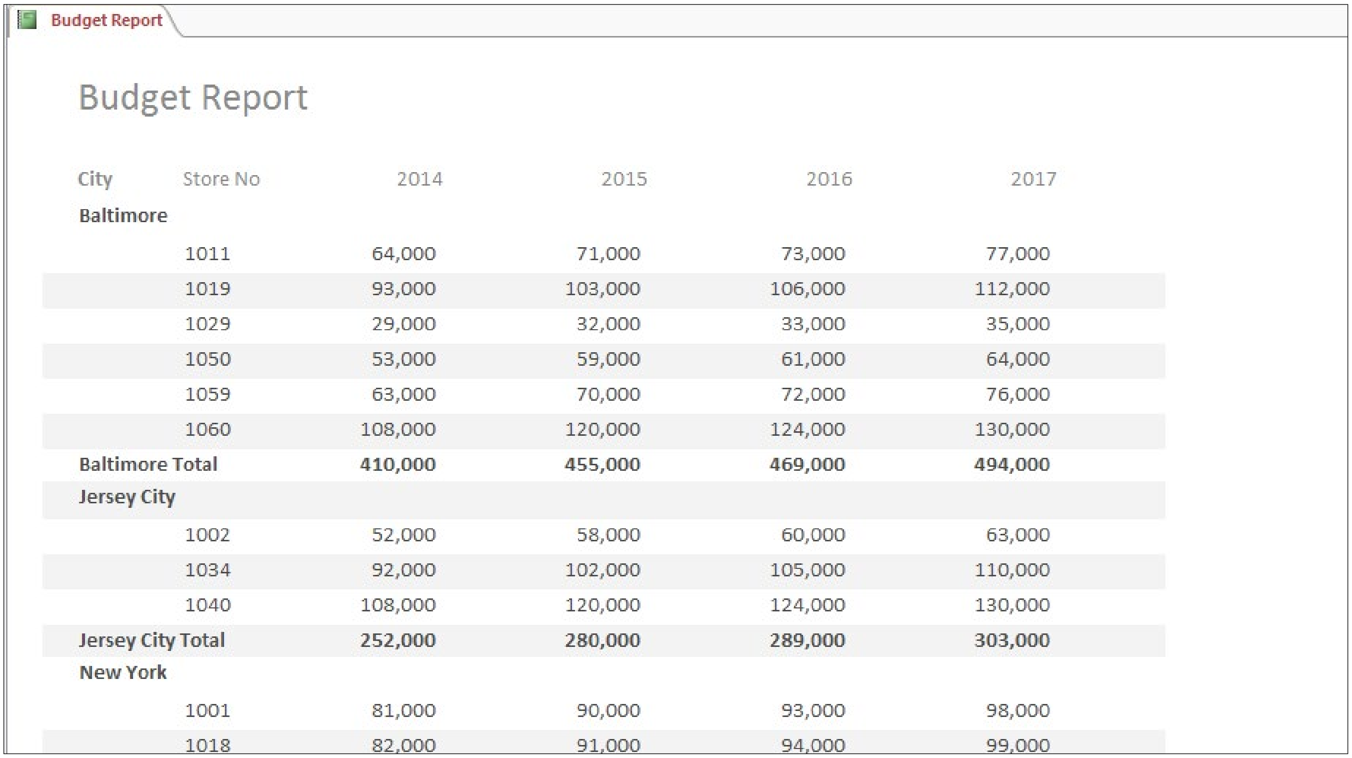
Budget Report

City	Store No	2014	2015	2016	2017
Baltimore					
	1011	64,000	71,000	73,000	77,000
	1019	93,000	103,000	106,000	112,000
	1029	29,000	32,000	33,000	35,000
	1050	53,000	59,000	61,000	64,000
	1059	63,000	70,000	72,000	76,000
	1060	108,000	120,000	124,000	130,000
	Baltimore Total	**410,000**	**455,000**	**469,000**	**494,000**
Jersey City					
	1002	52,000	58,000	60,000	63,000
	1034	92,000	102,000	105,000	110,000
	1040	108,000	120,000	124,000	130,000
	Jersey City Total	**252,000**	**280,000**	**289,000**	**303,000**
New York					
	1001	81,000	90,000	93,000	98,000
	1018	82,000	91,000	94,000	99,000
	1055	92,000	102,000	105,000	111,000

Figure 8.28

Notice how "Baltimore Total" appears under the Store No label. This is because we used a text box that was calculating a number and was formatted to be right-justified. We actually want to make that text box left-justified with the same formatting as the City text in the City Header section is.

25. *In* **Design View**, *click on the* **City Total** *text box and click the* **Align Left** *icon* ≡ *in the* **Font** *group of the* **Format** *tab.*
26. **Bold** *the* **City** *text box in the* **City Header** *section and the* **City Total** *text box in the* **City Footer** *section (if it's not bolded already).*
27. **View** *the report.*



Budget Report

Budget Report

City	Store No	2014	2015	2016	2017
Baltimore					
	1011	64,000	71,000	73,000	77,000
	1019	93,000	103,000	106,000	112,000
	1029	29,000	32,000	33,000	35,000
	1050	53,000	59,000	61,000	64,000
	1059	63,000	70,000	72,000	76,000
	1060	108,000	120,000	124,000	130,000
Baltimore Total		**410,000**	**455,000**	**469,000**	**494,000**
Jersey City					
	1002	52,000	58,000	60,000	63,000
	1034	92,000	102,000	105,000	110,000
	1040	108,000	120,000	124,000	130,000
Jersey City Total		**252,000**	**280,000**	**289,000**	**303,000**
New York					
	1001	81,000	90,000	93,000	98,000
	1018	82,000	91,000	94,000	99,000

Figure 8.29

If you go to the bottom of the report in Print Preview mode, you will see that it ends with Washington or Wilmington (this may vary slightly depending on your Margins settings). Management wants to see a total budget number for every year on the last page of the report. Hey, we can do that! Just create a Report Footer and write the formulas. Instead of copying labels and text boxes from another section, we'll create them using the icons in the ribbon.

28. *In* **Design View**, *place your cursor over the bottom line of the* **Report Footer** *section.*
29. *Click and drag the bottom border down about half an inch.*
30. *Click on the* **Label** *icon in the* **Controls** *group of the* **Design** *tab.*
31. *Draw a rectangular box on the left side of the* **Report Footer** *section and in it type:* ***REPORT TOTAL*** *and press* **[Enter]**.
32. *Click on the* **Text Box** *icon in the* **Controls** *group of the* **Design** *tab.*
33. *Draw a rectangular box in the* **Report Footer** *section which aligns under the Year* **2014**.

When you draw the text box, two boxes appear. The first one is a text box label. It should say something like Text21. The other box is a text box that should read "Unbound". You need only the unbound text box.

34. *Click on the* **Text Box** *label (which reads something like* **Text21***) and delete it.*
35. *Click inside the* **Unbound** *text box (you may have to click twice – once to select it and again to put it in* **Edit** *mode). The word* **Unbound** *should disappear.*
36. *Inside the new text box, type* **=Sum([2014])** *and press* **[Enter]**.
37. *Create other text boxes for the years* **2015**, **2016**, *and* **2017**.

The Report Footer section should now look something like this:

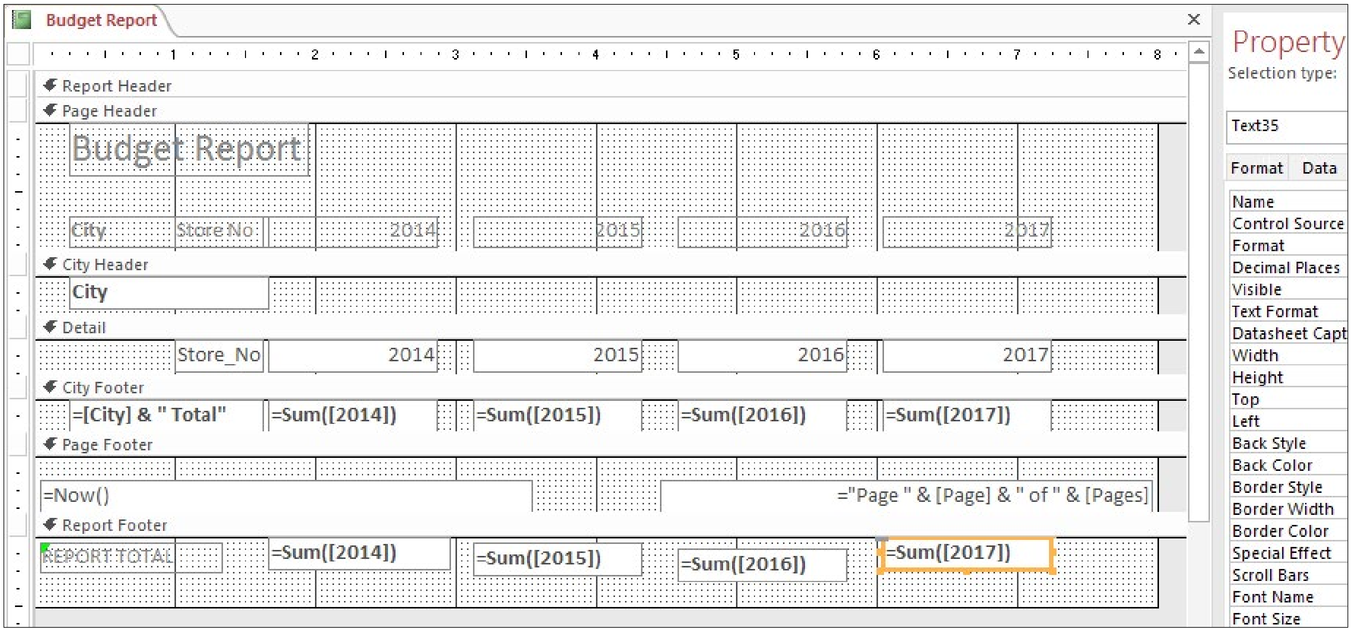


Figure 8.30

Aligning Objects

When you create text boxes and labels like this, particularly when they are not part of a layout grid, sometimes they get out of alignment. When there are numerous text boxes and labels, it can be difficult to manually get them to all align correctly. You need to make sure everything is in alignment from top to bottom AND from right to left. Fortunately, there is a tool you can use to make sure everything is in a similar alignment. Let's do the top to bottom alignment first, as well as some formatting in the Report Footer section. These controls are located in the Arrange tab.

38. *Click in the* **Left Ruler** *and select all of the* **text boxes** *and* **labels** *in the* **Report Footer** *section.*
39. *Click on the* **Arrange** *tab of the* **Report Design Tools** *contextual tab, and click on the* **Size/ Space** *icon in the* **Sizing and Ordering** *group.*
40. *Click on the* **To Shortest** To Shortest *icon under* **Size**.
41. *Then click on the* **Align** *icon in the same group and click on* **Top** Top .

All of the text boxes and the label are now aligned to the top and all should have the same height. Depending on how you drew the label and text boxes, you may have to play around with the icons in the Arrange tab to get them to look like the illustrations provided. Let's continue on with some more formatting.

42. ***Format** all text boxes (not the label) as **Standard** with **zero decimal places** (remember to use the **Property Sheet** dialog box to do the formatting).*
43. *Select the **2014** label in the **Page Header**, hold down the **[Shift]** key and click on the **2014** text box in the **Detail** section, the **Sum([2014])** text box in the **City Footer** section, and the **Sum([2014])** text box in the **Report Footer** section.*
44. *Click the **Align Right** icon in the **Font** group of the **Format** tab.*
45. *To make sure all of the objects are aligned with each other, click on the **Align** icon in the **Sizing & Ordering** group and choose **Right**.*
46. *Do the same for the text boxes and labels under **2015**, **2016**, and **2017**.*

The Design Grid of the report should look like the figure below.

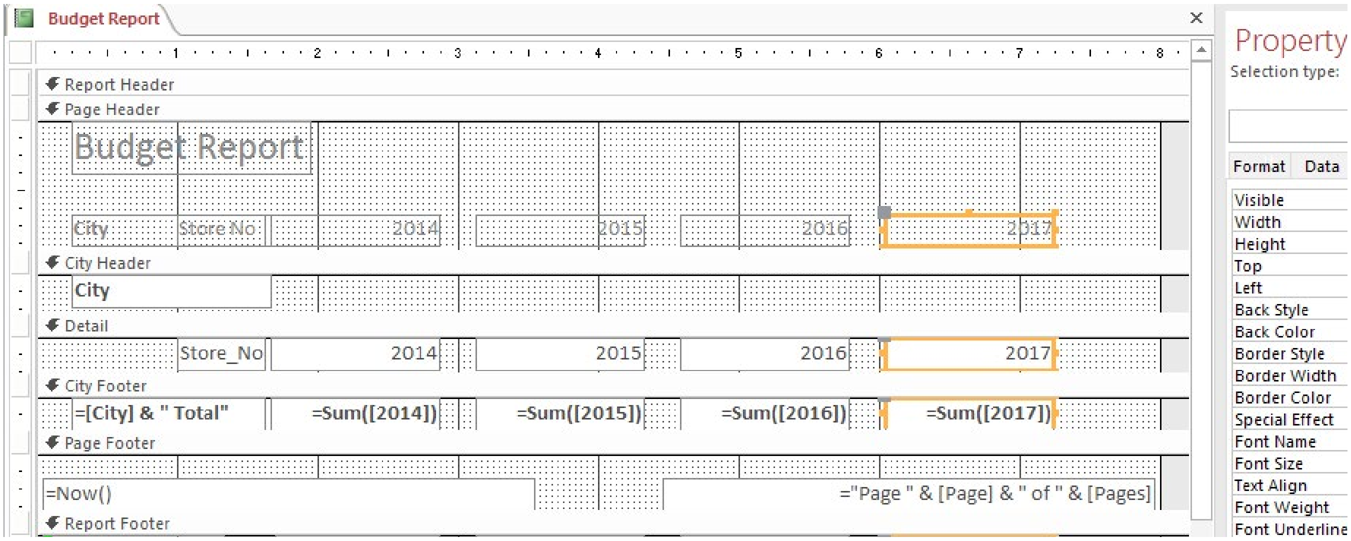
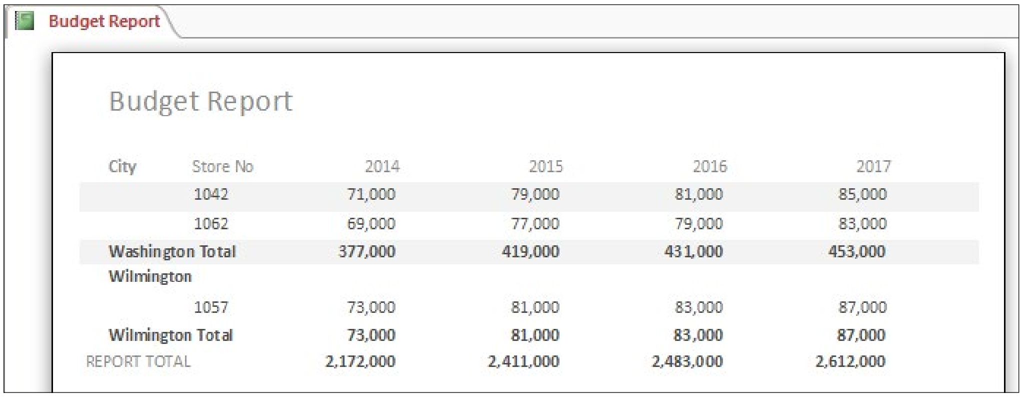


Figure 8.31

47. ***Run** the report in **Print Preview** and go to the **Page 2**.*



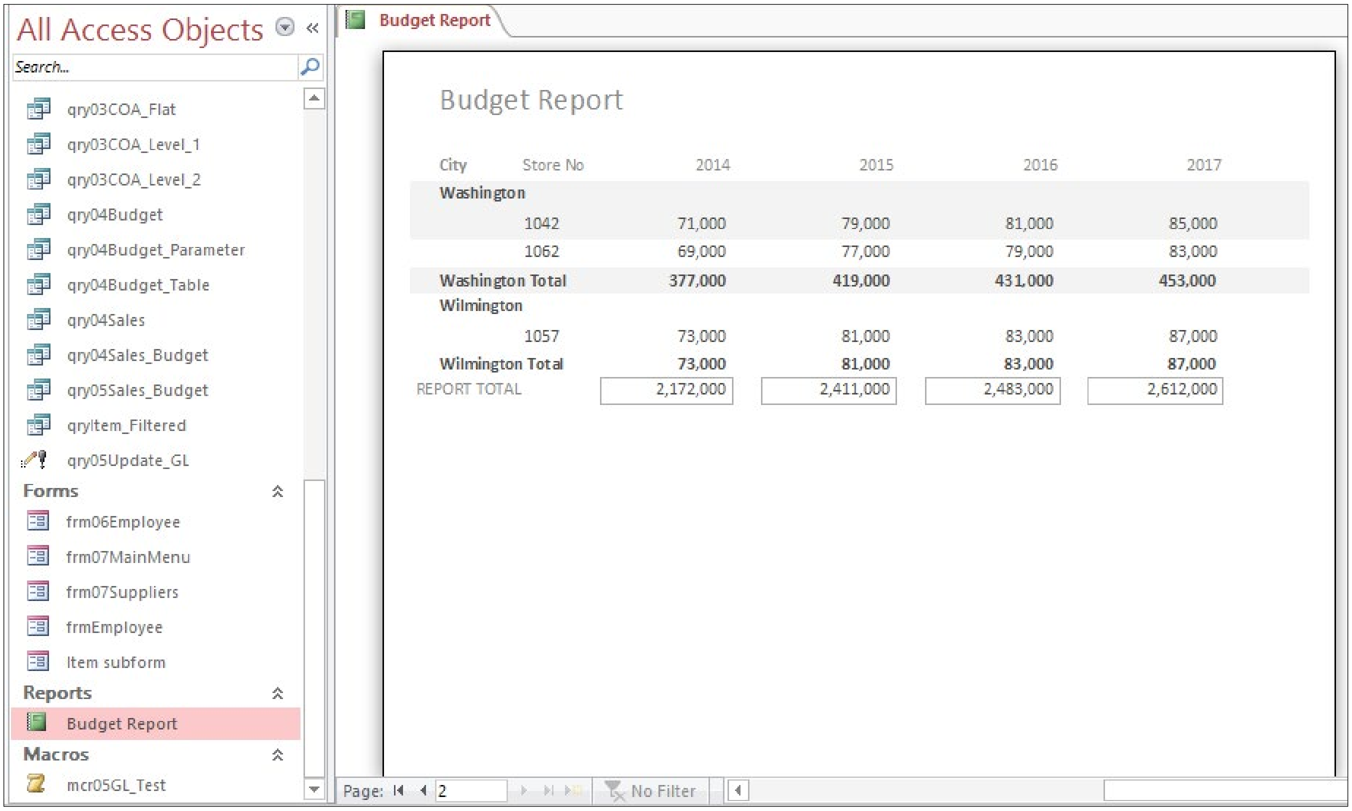
Budget Report

City	Store No	2014	2015	2016	2017
	1042	71,000	79,000	81,000	85,000
	1062	69,000	77,000	79,000	83,000
Washington Total		**377,000**	**419,000**	**431,000**	**453,000**
Wilmington					
	1057	73,000	81,000	83,000	87,000
Wilmington Total		**73,000**	**81,000**	**83,000**	**87,000**
REPORT TOTAL		2,172,000	2,411,000	2,483,000	2,612,000

Figure 8.32

Do you see how there is not a city name in the first row below the City heading? The City heading for this group (Washington) begins on the first page. I like to have the name of the City (or other similar group) repeat on pages where the data are split. You can accomplish this by using the Repeat Section property.

48. *Go to the* **Design View** *of the report, right-click on the* **City Header** *gray section bar and choose* **Properties** *(if the Property Sheet is not already displayed)*
49. *On the* **Format** *tab, scroll down to the* **Repeat Section** *property and change it to* **Yes***.*
50. **Run** *the report in* **Print Preview** *and scroll to the bottom of the report.*



Budget Report

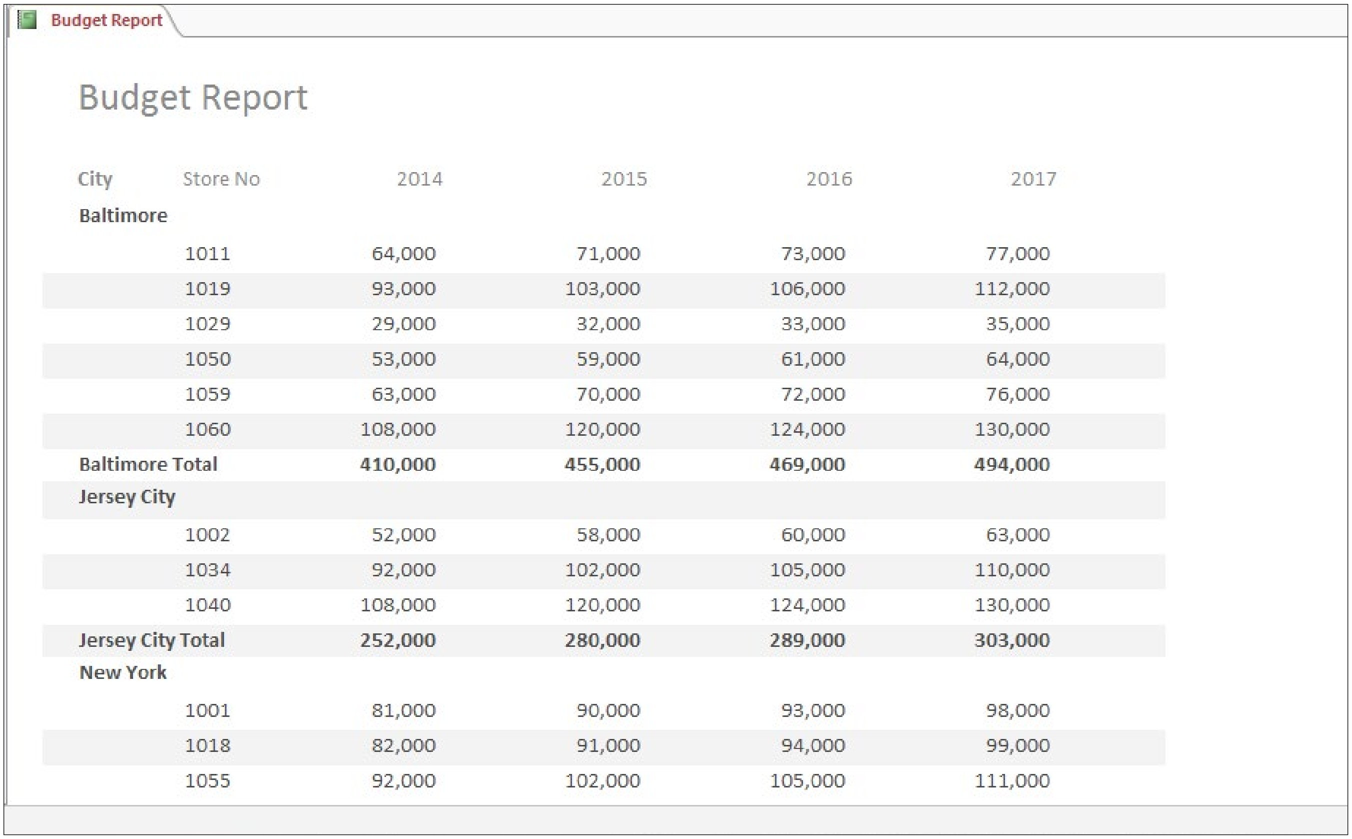
City	Store No	2014	2015	2016	2017
Washington					
	1042	71,000	79,000	81,000	85,000
	1062	69,000	77,000	79,000	83,000
Washington Total		**377,000**	**419,000**	**431,000**	**453,000**
Wilmington					
	1057	73,000	81,000	83,000	87,000
Wilmington Total		**73,000**	**81,000**	**83,000**	**87,000**
REPORT TOTAL		2,172,000	2,411,000	2,483,000	2,612,000

Figure 8.33

There's just a little more clean-up to do before it's perfect. We just need to bold the REPORT TOTAL line and take out those borders around the text boxes.

51. *In* **Design View***, select all objects in the* **Report Footer** *section.*
52. *In the* **Property Sheet***, click on the drop-down arrow next to* **Border Style** *and select* **Transparent***.*
53. *Make all of the* **text boxes** *and the* **label bold***.*

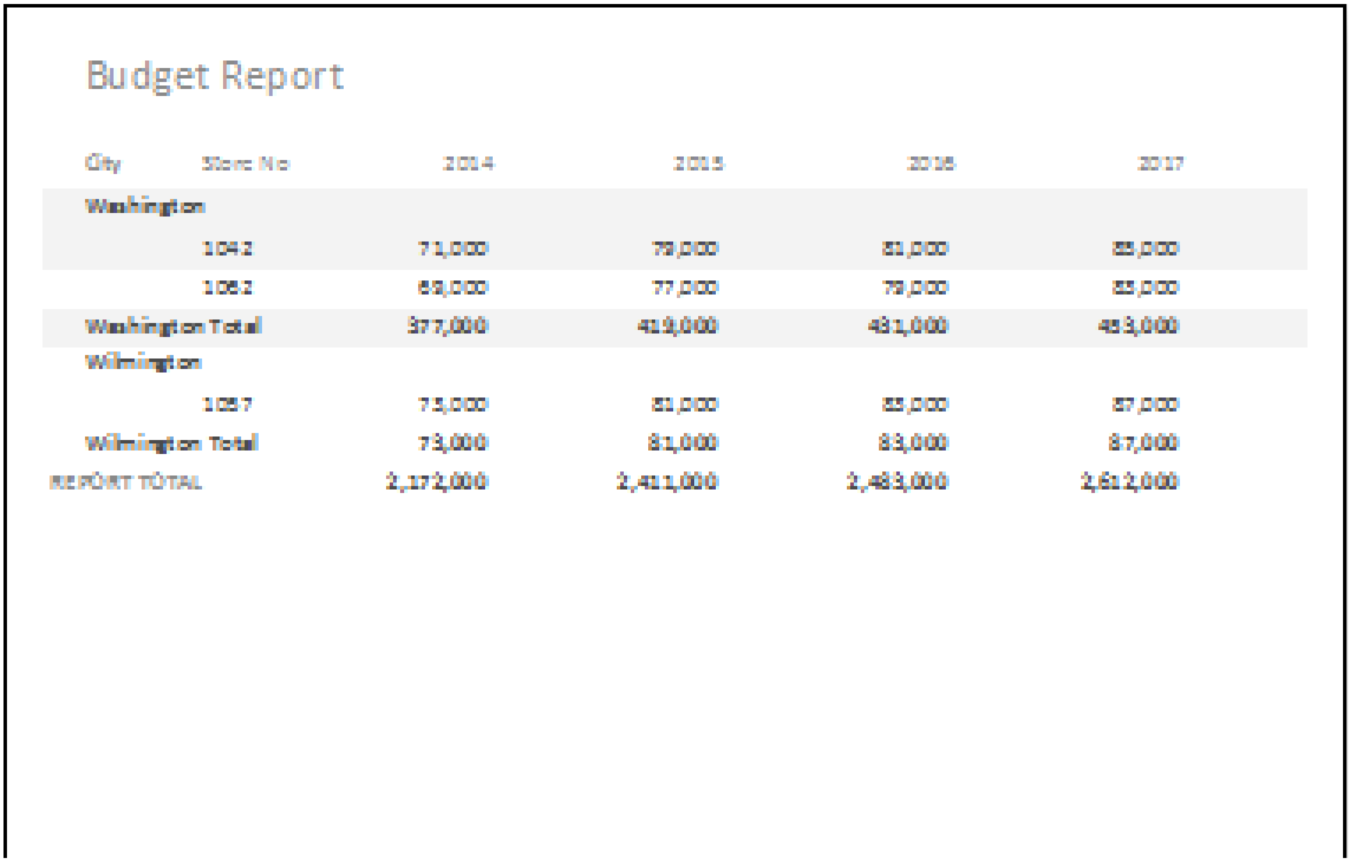
> ***Note****: You may recognize that the page numbering differs between* **Report View** *and* **Print Preview***.*



Budget Report

City	Store No	2014	2015	2016	2017
Baltimore					
	1011	64,000	71,000	73,000	77,000
	1019	93,000	103,000	106,000	112,000
	1029	29,000	32,000	33,000	35,000
	1050	53,000	59,000	61,000	64,000
	1059	63,000	70,000	72,000	76,000
	1060	108,000	120,000	124,000	130,000
Baltimore Total		410,000	455,000	469,000	494,000
Jersey City					
	1002	52,000	58,000	60,000	63,000
	1034	92,000	102,000	105,000	110,000
	1040	108,000	120,000	124,000	130,000
Jersey City Total		252,000	280,000	289,000	303,000
New York					
	1001	81,000	90,000	93,000	98,000
	1018	82,000	91,000	94,000	99,000
	1055	92,000	102,000	105,000	111,000

Figure 8.34



Budget Report

City	Store No	2014	2015	2016	2017
Washington					
	1042	71,000	79,000	81,000	85,000
	1062	69,000	77,000	79,000	85,000
Washington Total		377,000	419,000	431,000	453,000
Wilmington					
	1057	73,000	81,000	85,000	87,000
Wilmington Total		73,000	81,000	83,000	87,000
REPORT TOTAL		2,172,000	2,411,000	2,483,000	2,612,000

Figure 8.35

54. **Save** *and close* **Budget Report**.
55. *Rename* **Budget Report** *as* ***rpt08Bgt_Rpt***.

Wow! Looks impressive. Your manager takes one look at it and nominates you for Employee of the Year! It may take you awhile to get it to look like the report shown here, but it will be a fantastic sense of accomplishment once you get it completed.

Reports in Access can sometimes be clunky to work with. However, once you get used to working with reports, it will be second nature to you. This type of Design Grid is similar to other reporting programs, like Crystal Reports® and Microsoft SQL Server Reporting Services®. Once you learn to create reports in Access, learning to design reports in these other tools will be much easier.

> ***Review Questions****: It is now time to complete the hands-on Review Questions. Log on to www.ExcelCEO.com with your Email and Password, click on the* **Access 2016 and SQL Review Questions, Chapter 8, Section 2 of 2** *option and complete the review questions.*

Conclusion

In this chapter, you learned the basics of Access report writing. You connected a report to a query and used the Report Wizard to create a simple report that you then edited. You explored the Properties of the various text boxes and labels and learned how to use Sorting and Grouping in your report. You used the Sizing & Ordering group to get all labels and text boxes the same size and to align with each other. In the next chapter, you'll expand on your report-writing skills by doing even more modifications to this same report.

Chapter Exam

You can now go to www.ExcelCEO.com, click on Sign In, log in and take the exam. Make sure that you take the exam on the same device on which you completed the practice files, as some of the questions on the exam may refer to some of the completed projects. Chapter exams are intended to be hands-on.

Access® 2016 and SQL

Complete Self-study Course

ExcelCEO

Chief Excel Officer

CHAPTER NINE — INTERMEDIATE REPORTING

In this chapter, you will:

- Identify how to make a copy of an existing report and modify it
- Recognize the new header and footer sections to a report
- Select the options to resize and reorient a report
- Determine how to calculate Subtotals within a report
- Identify how to create a Mailing Labels Report

CPE Credits possible for this chapter: 3

Editing a Report

One of the most frustrating things in our job can be the first time we present a report. You'd think it would be a joyous occasion – to finally present the report that you've been working so hard on. But that day comes, you bring the report in to your manager, he looks at it and exclaims, "*This looks GREAT!*", and he starts to thumb through it. Then he follows his first statement with, "*Do you think you can …?*" It will happen almost every time. So the more you can prepare yourself for that eventuality, the better off you will be when it happens.

That scenario happens with the report you created in Chapter Eight. You bring the really good-looking report into your manager and he says, after he has nominated you for the Employee of the Year, "*You know, it really would be great to see it broken out by Region, as well as by City. Can you do that?*" You say, "*No problem.*" It really isn't that hard. All you need are the data in the query that support the information he needs, to make a few modifications, and you have it.

Copy an Existing Report

Before we get started, let's copy the last query and report you built so you can use them in this chapter.

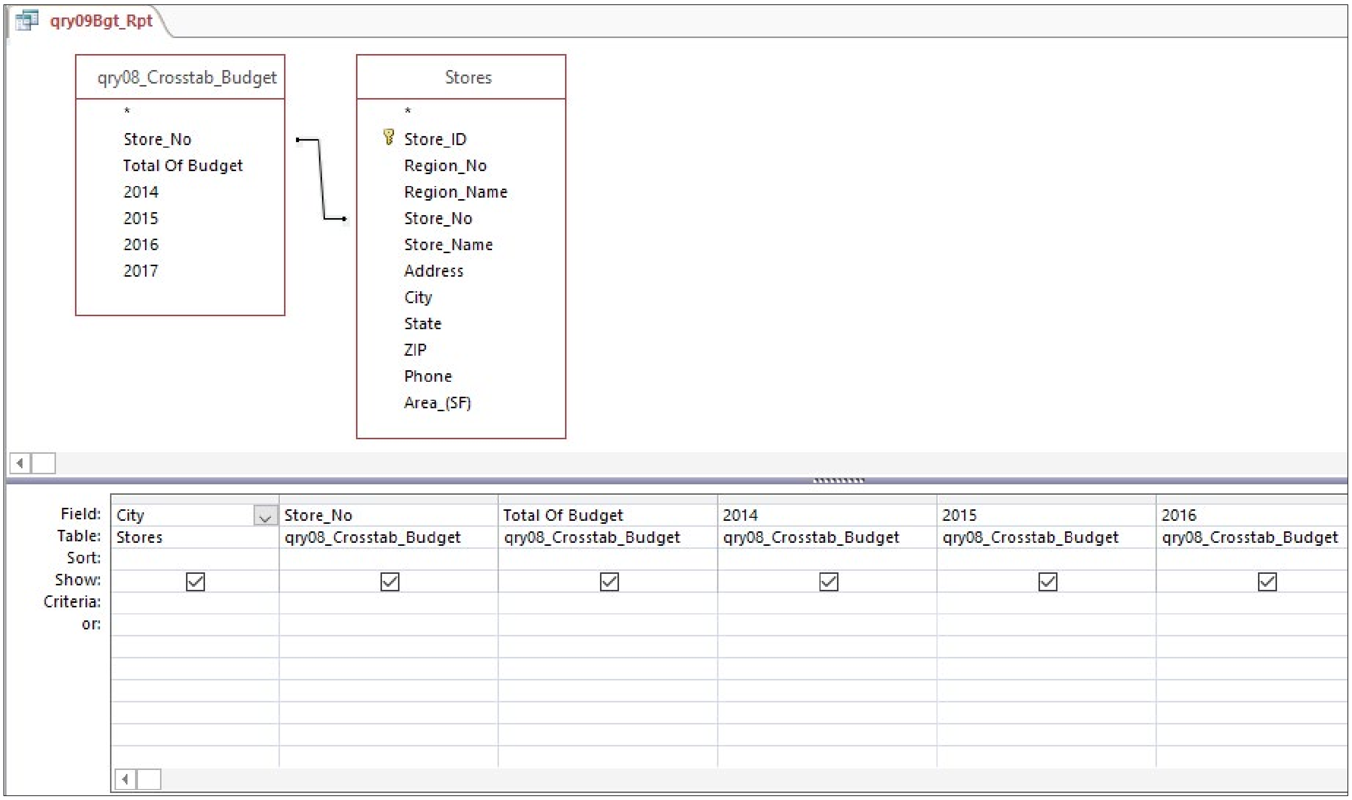
1. *Open the* **Nitey_Nite_2016** *database, close the* **Main Menu** *form,* **copy qry08Bgt_Rpt** *and name the copy* ***qry09Bgt_Rpt***.
2. **Copy rpt08Bgt_Rpt** *and name it* ***rpt09Bgt_Rpt***.

You can leave qry09Bgt_Rpt connected to the crosstab query we did in Chapter Eight, as we will be using that base data. The next housekeeping thing we need to do for our new report is to connect to the new data source. This is a new report, and we want to make sure it is connected to qry09Bgt_Rpt.

3. *Open* **rpt09Bgt_Rpt** *in* **Design View**.
4. *Make sure the gray box at the intersection of the two rulers is selected.*
5. *Click on the* **Property Sheet** *icon and on the* **Data** *tab change the* **Record Source** *from* **qry08Bgt_Rpt** *to* **qry09Bgt_Rpt**.

Now we can concentrate on the problem at hand, which is to bring the Region Name into the report and do the respective subtotals. The first thing we have to do is to modify the query. Aren't you glad you based the report on a query and not a table? You just changed the record source of the report to be qry09Bgt_Rpt. Let's open that query.

6. *Open* **qry09Bgt_Rpt** *in* **Design View**. *Adjust the grid, if needed.*



Field:	City	Store_No	Total Of Budget	2014	2015	2016
Table:	Stores	qry08_Crosstab_Budget	qry08_Crosstab_Budget	qry08_Crosstab_Budget	qry08_Crosstab_Budget	qry08_Crosstab_Budget
Sort:						
Show:	☑	☑	☑	☑	☑	☑
Criteria:						
or:						

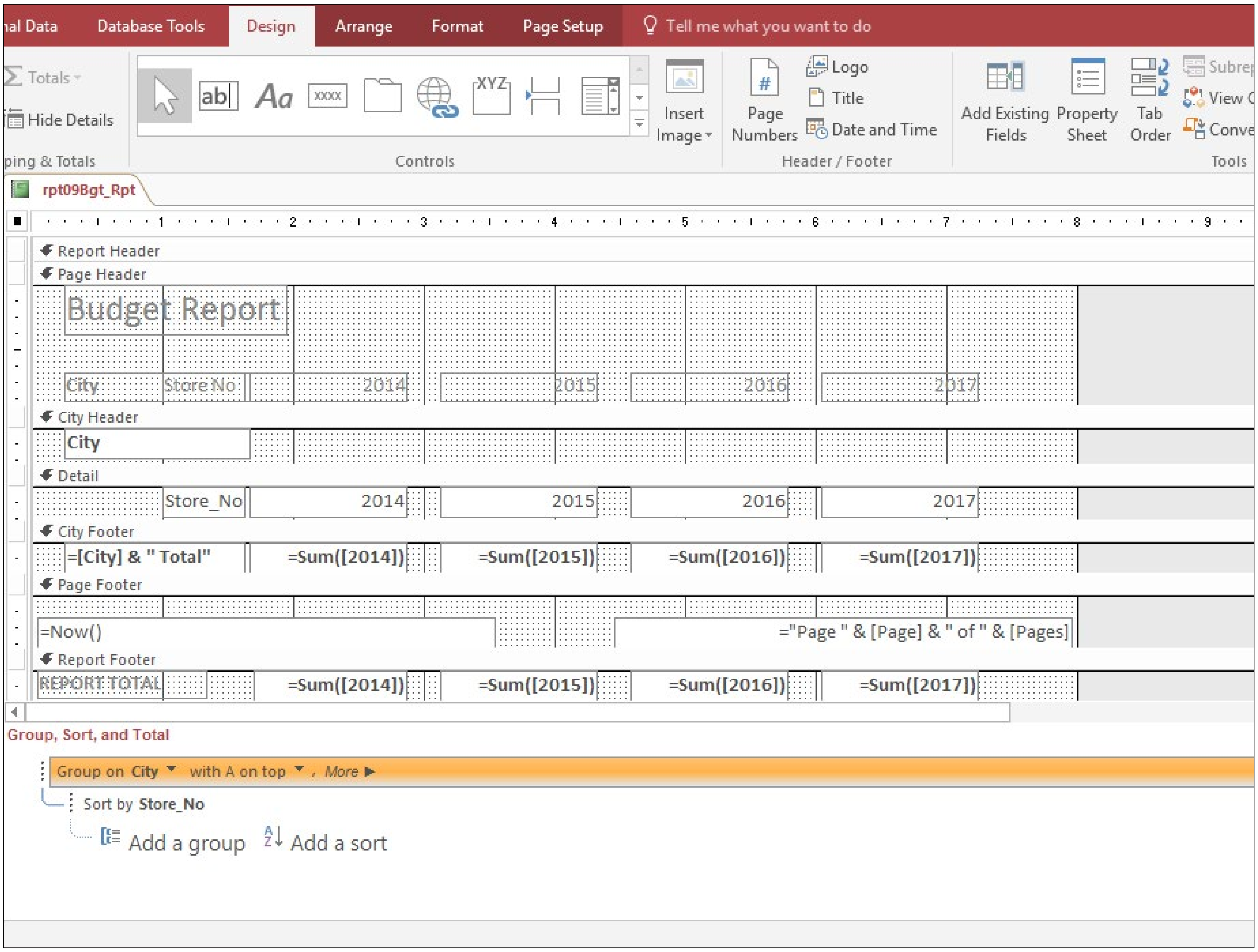
Figure 9.1

As you can see, all the data is already here for you to include the Region Name. All you have to do is to bring the Region_Name field down into the Design grid.

7. *Bring the* **Region_Name** *field down and place it to the left of the* **City** *field.*
8. **Save** *and close* **qry09Bgt_Rpt**.

Now that the field is in the query, you need to create a report header and footer for that level.

9. *In the* **Report Design Tools Design** *contextual tab of* **rpt09Bgt_Rpt**, *click on the* **Group & Sort** *icon.*



Report Header

Page Header

Budget Report

City | Store No | 2014 | 2015 | 2016 | 2017

City Header

City

Detail

Store_No | 2014 | 2015 | 2016 | 2017

City Footer

=[City] & " Total" | =Sum([2014]) | =Sum([2015]) | =Sum([2016]) | =Sum([2017])

Page Footer

=Now() | ="Page " & [Page] & " of " & [Pages]

Report Footer

REPORT TOTAL | =Sum([2014]) | =Sum([2015]) | =Sum([2016]) | =Sum([2017])

Group, Sort, and Total

Group on **City** ▼ with A on top ▼ , *More* ▶

Sort by **Store_No**

Add a group   Add a sort

Figure 9.2

Adding New Header and Footer Sections

Currently, you have only two fields in the Group, Sort, and Total section: City and Store_No. In order to have the Region_Name level, you need to add it in this area. In the Group, Sort, and Total area, the Region_Name field needs to appear first because it is the most comprehensive grouping category. Let's add the group for Region Name.

10. *Click on the* **Add a group** *button below* **Sort by Store_No** *and choose* **Region_Name**.
11. *When the* **Region_Name** *group appears below* **Sort by Store_No**, *click and drag it above the* **Group on City** *group (or use the* **Move up** *arrow)*

> **Note**: *You may have to click and drag on the vertical dots icon to the left of the group.*

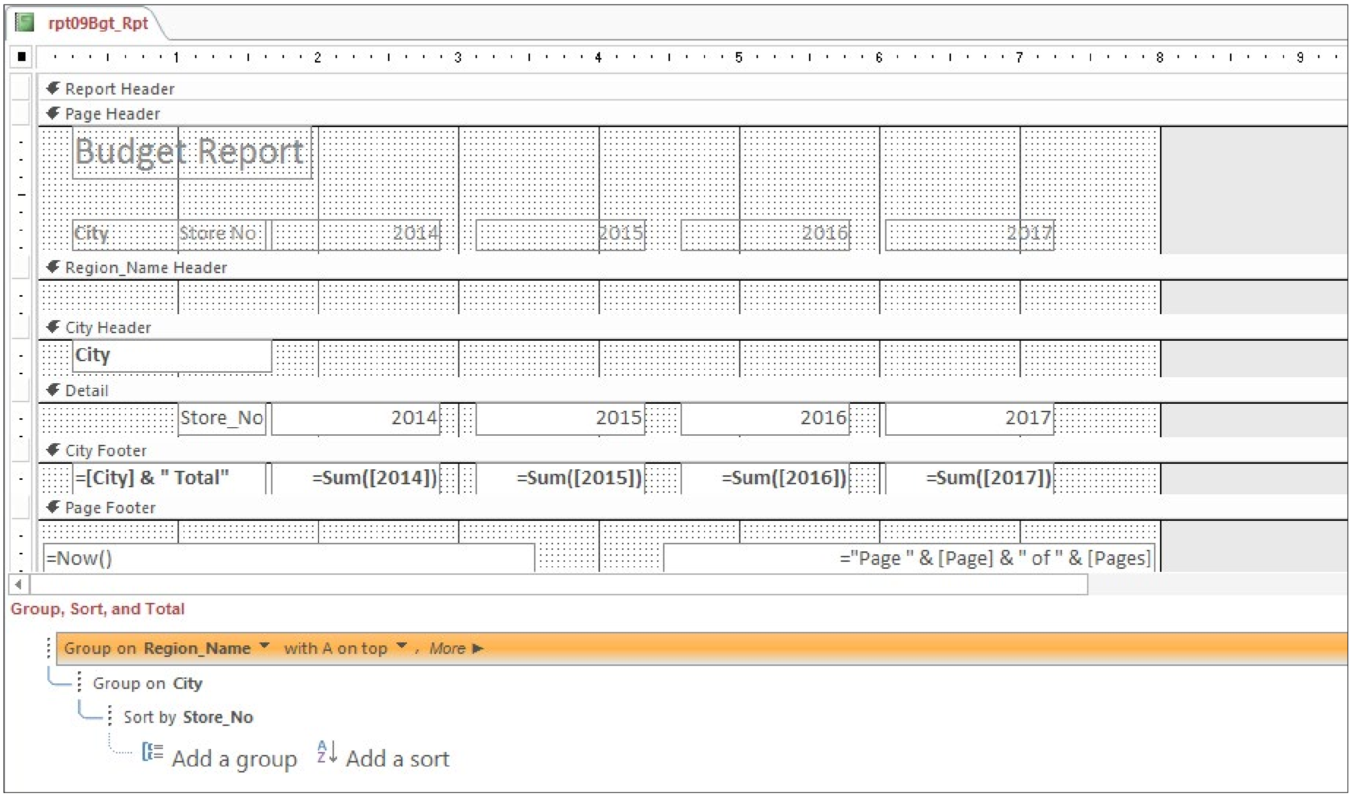


Figure 9.3

Notice that a Region_Name Header section appears in the Report Grid, but the footer does not. To see the properties of the group, click on the More icon.

12. *In the* **Group on Region_Name** *group, click on the* More ▶ *icon.*
13. *Change the* **without a footer section** *property to* **with a footer section** *and click the* Less ◀ *icon.*

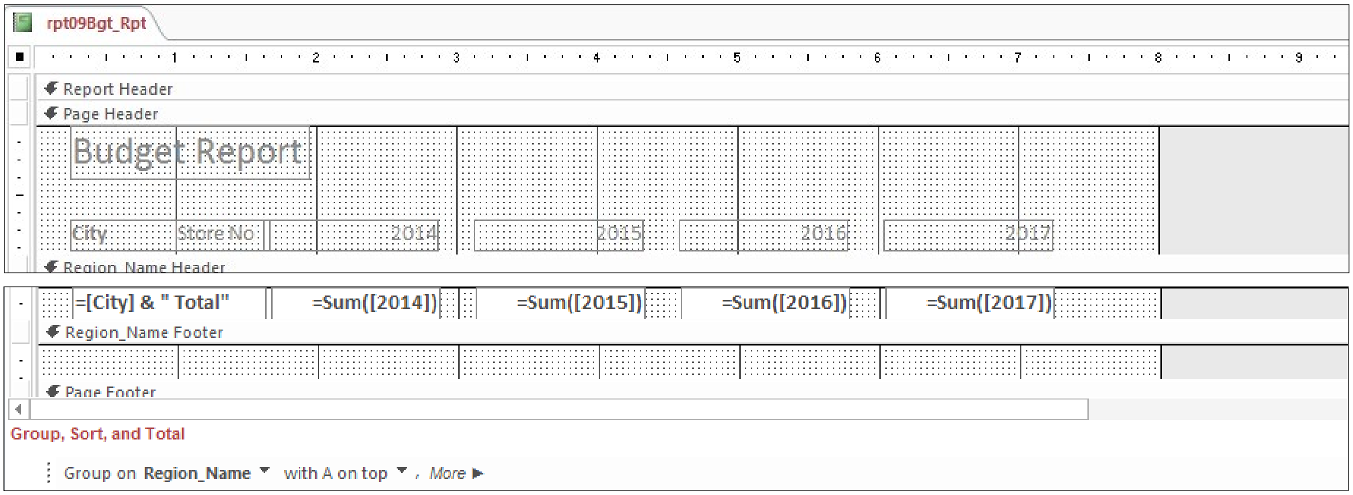


Figure 9.4

The Region_Name Header and Region_Name Footer sections appear in the Design Grid of the report.

Resizing and Reorienting a Report

Now that you have more objects in the Design Grid, you will see that you will have to increase the size of the grid to have ample space to include the remaining objects. Resizing a grid is easy – just click and drag.

1. *Close the* **Group, Sort, and Total** *section.*
2. *Increase the size of the* **Design Grid** *by dragging its right edge to the right until it reaches about* **8¾ inches**.
3. *Move all* **labels** *and* **text boxes** *in the* **Design Grid** *(except for the* **Budget Report** *and* **REPORT TOTAL** *labels and the* **Now()** *text box) to the right to make room for the* **Region_Name** *label and data.*
4. *In the* **Page Header** *section, create a new* **label** *with the caption* ***Region Name*** *and place it to the left of the* **City** *label.*
5. *In the* **Region_Name Header** *section, create a text box similar to the one in the* **City** *section and name it* ***Region_Name***.
6. *In the* **Region_Name Footer** *section, write formulas that sum up the subtotals by* **Region_Name**, *similar to the ones in the* **City Footer** *section.*
7. **Create** *a* ***Region_Name Total*** *text box, similar to the one in the* **City** *footer.*
8. *Position the* **City** *and the* **City Total** *text boxes to make them more indented than their* **Region_Name** *equivalents.*
9. *Align all of the objects in the report to look like the following figure:*

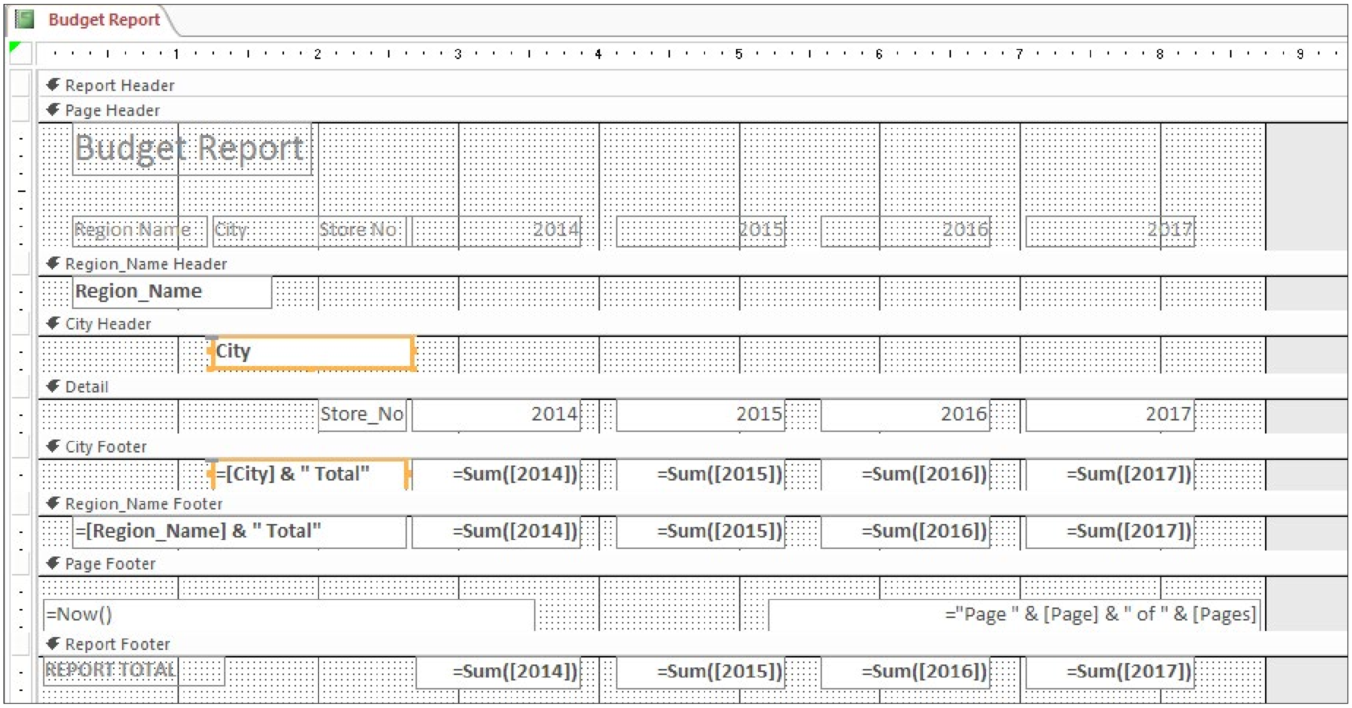
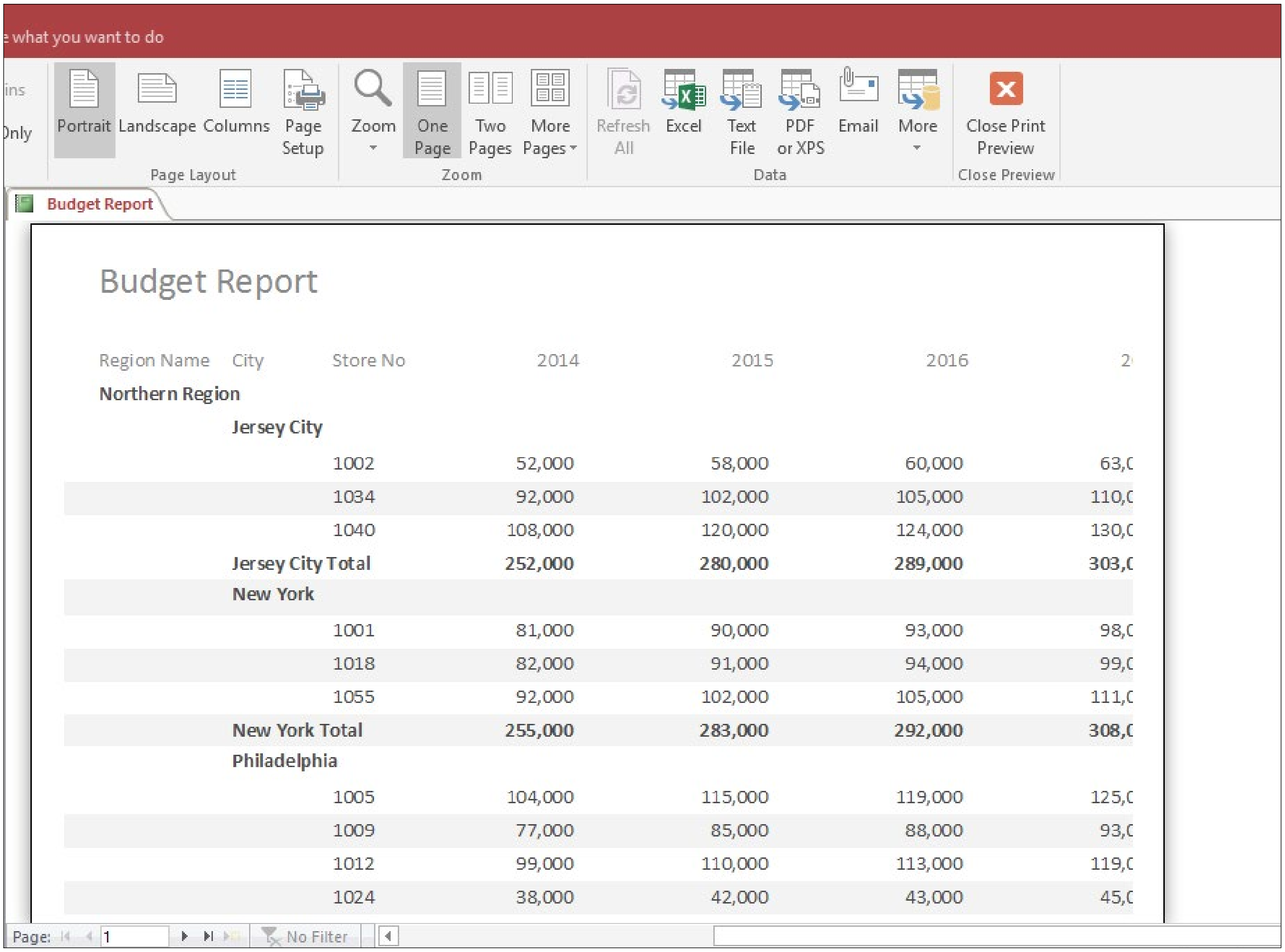


Figure 9.5

It will probably take you awhile to accomplish this, but it is necessary for you to have this experience in developing reports.

10. *Run the report in* **Print Preview***.*



Budget Report

Region Name	City	Store No	2014	2015	2016	2(
Northern Region						
	Jersey City					
		1002	52,000	58,000	60,000	63,(
		1034	92,000	102,000	105,000	110,(
		1040	108,000	120,000	124,000	130,(
	Jersey City Total		252,000	280,000	289,000	303,(
	New York					
		1001	81,000	90,000	93,000	98,(
		1018	82,000	91,000	94,000	99,(
		1055	92,000	102,000	105,000	111,(
	New York Total		255,000	283,000	292,000	308,(
	Philadelphia					
		1005	104,000	115,000	119,000	125,(
		1009	77,000	85,000	88,000	93,(
		1012	99,000	110,000	113,000	119,(
		1024	38,000	42,000	43,000	45,(

Figure 9.6

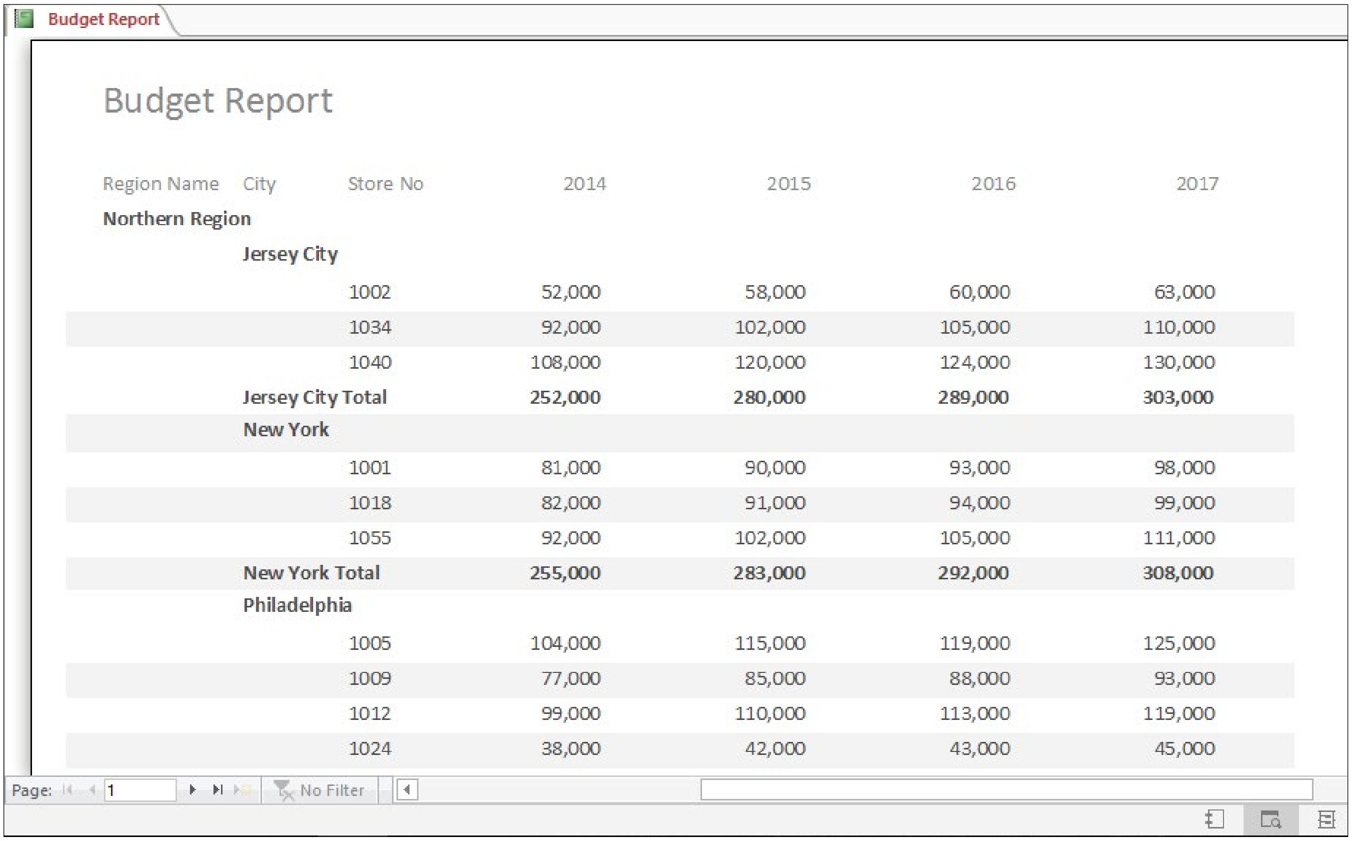
Changing Page Orientation

Whoa there! Where did the data for the year 2017 go? If you click on the page selectors at the bottom of the report, you'll see that the report is now four pages long instead of two. That is because the report is too wide (8¾ inches) to display in Portrait layout. Not to worry, we can easily change the Page Setup.

1. *Click on the* **Page Setup** *icon in the* **Page Layout** *group.*
2. *In the* **Page Setup** *dialog box, click on the* **Page** *tab and choose the* **Landscape** *radio button, then click* **OK***.*

You can also click on the Margins icon to adjust the top, bottom, left, and/or right margins to fit, but since the report is only 8¾ inches wide, just changing it to a Landscape orientation will most likely do the trick. You can change the margins or orientation either in Print Preview, Design View, or Layout View, but not in Report View.

> ***Note**: You could also directly choose* **Landscape** *from the* **Page Layout** *group, change margins, etc. Either way, you should be able to see the updates since you are still in* **Print Preview** *mode. Seeing the real-time updates is a nice time-saving feature!*



Budget Report

Region Name	City	Store No	2014	2015	2016	2017
Northern Region						
	Jersey City					
		1002	52,000	58,000	60,000	63,000
		1034	92,000	102,000	105,000	110,000
		1040	108,000	120,000	124,000	130,000
	Jersey City Total		252,000	280,000	289,000	303,000
	New York					
		1001	81,000	90,000	93,000	98,000
		1018	82,000	91,000	94,000	99,000
		1055	92,000	102,000	105,000	111,000
	New York Total		255,000	283,000	292,000	308,000
	Philadelphia					
		1005	104,000	115,000	119,000	125,000
		1009	77,000	85,000	88,000	93,000
		1012	99,000	110,000	113,000	119,000
		1024	38,000	42,000	43,000	45,000

Figure 9.7

The report should now appear on two pages, with all columns appearing on each page.

3. *Close* **Print Preview** *mode.*

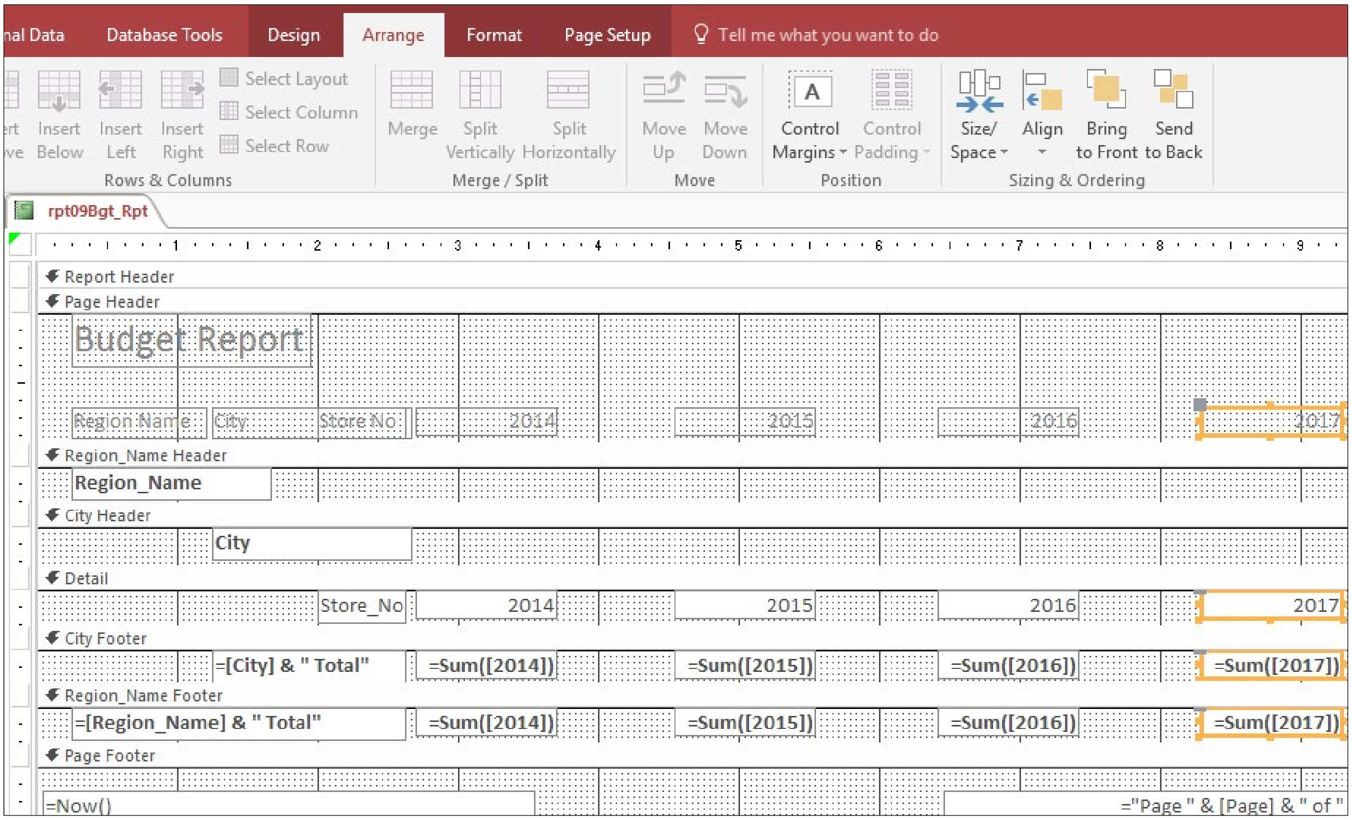
> ***Review Questions**: It is now time to complete the hands-on Review Questions. Log on to www.ExcelCEO.com with your Email and Password, click on the* **Access 2016 and SQL Review Questions, Chapter 9, Section 1 of 2** *option and complete the review questions.*

Calculating Subtotals and Percentages

After you show the report to your manager, he has another request. It appears he's been thinking a lot about this report, and how good you are at building reports in Access. He has a simple request (at least it's simple in his mind). He wants to see the percentage of budget contributed by each store within each city and region. For example, Jersey City has three stores: 1002, 1034, and 1040. These stores have budgets in 2014 of $52,000, $92,000, and $108,000, respectively, for a city total of $252,000. He would like to see the percentage contributed by each store (20.6% for Store 1002, 36.5% for Store 1034, and 42.9% for Store 1040).

Next, he wants to see the percentage of each region contributed by each city in the region. Jersey City, for example, has a total 2014 budget of $252,000, which is 22.0% of the Northern Region budget of $1,143,000. Finally, he wants to see the percentage contributed by the Northern and Southern Regions as compared with that of the entire company. Of course, the entire company's number would be 100%. If all we're doing is calculations, it should be relatively easy to do, right? It is, but we're going to have to do some reformatting and moving things around a bit to get everything to fit on the two pages. Let's get started.

1. *In* **Design View** *of the report, increase the width of the* **Design Grid** *to* **10 inches**.
2. *Select all* **text boxes** *under and including the headings* **2014 – 2017**, *except the* **Page Numbers** *text box.*
3. *In the* **Property Sheet**, *make all year text boxes and labels have a width of* **1"** *and a height of* **0.2"**.
4. *Make sure all objects under each heading are right-aligned.*
5. *Reposition all objects to appear as in the following figure:*



Budget Report

Region Name | City | Store No | 2014 | 2015 | 2016 | 2017

Region_Name

City

Store_No | 2014 | 2015 | 2016 | 2017

=[City] & " Total" | =Sum([2014]) | =Sum([2015]) | =Sum([2016]) | =Sum([2017])

=[Region_Name] & " Total" | =Sum([2014]) | =Sum([2015]) | =Sum([2016]) | =Sum([2017])

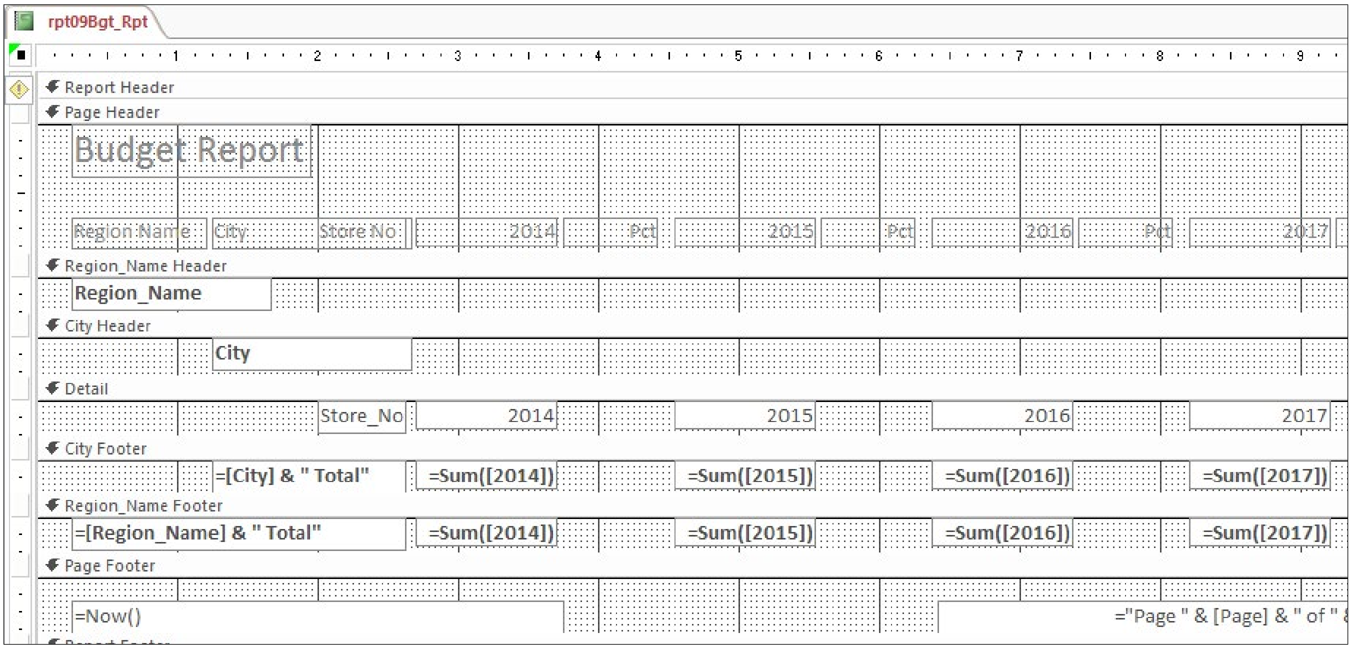
=Now() | ="Page " & [Page] & " of "

Figure 9.8

Setting Report Margins

When you resize text boxes that contain data, make sure you go back and forth between Design View, Report View and Print Preview to make sure the text boxes are large enough. If the text box is too small, it may not display all of the data. A good check is to verify the Report Total number, as it is usually the largest number with the largest font. Make sure they are all in the appropriate alignment, both horizontally and vertically. Your goal in the next few exercises is to put a Percent column after each year, and you need to make sure you have enough room in between each year's data to do so.

6. *Create* **labels** *that read* **Pct** *to the right of each* **Year label**.
7. *Click on the* **Page Setup** *tab and click on the* **Page Setup** *icon. On the* **Print Options** *tab, make sure* **Left** *and* **Right margins** *are set to* **0.5"** *and the* **Top** *and* **Bottom margins** *are set to* **0.25"**.



Report Header

Page Header

Budget Report

Region Name | City | Store No | 2014 | Pct | 2015 | Pct | 2016 | Pct | 2017

Region_Name Header

Region_Name

City Header

City

Detail

Store_No | 2014 | 2015 | 2016 | 2017

City Footer

=[City] & " Total" | =Sum([2014]) | =Sum([2015]) | =Sum([2016]) | =Sum([2017])

Region_Name Footer

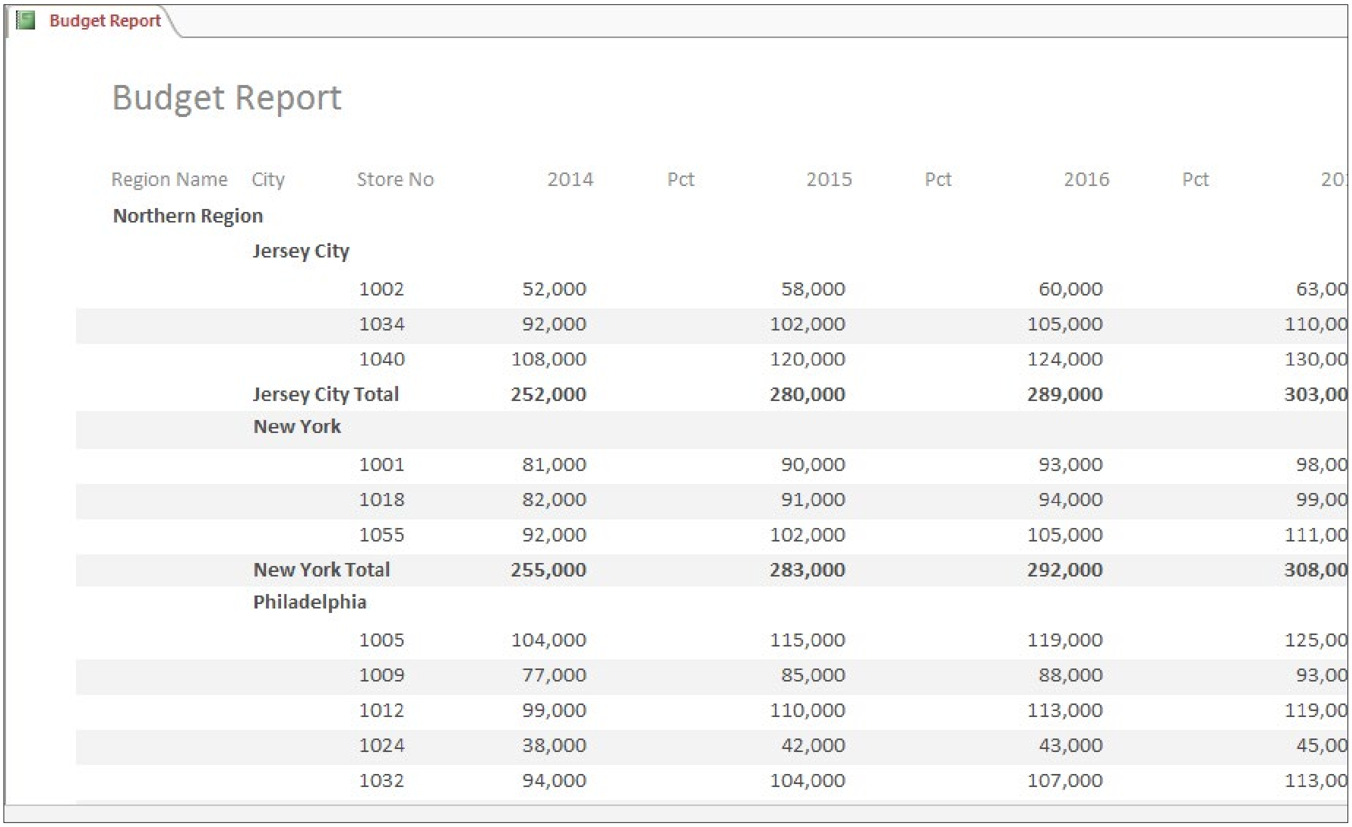
=[Region_Name] & " Total" | =Sum([2014]) | =Sum([2015]) | =Sum([2016]) | =Sum([2017])

Page Footer

=Now() | ="Page " & [Page] & " of " &

Figure 9.9

8. **View** *the report.*



Budget Report

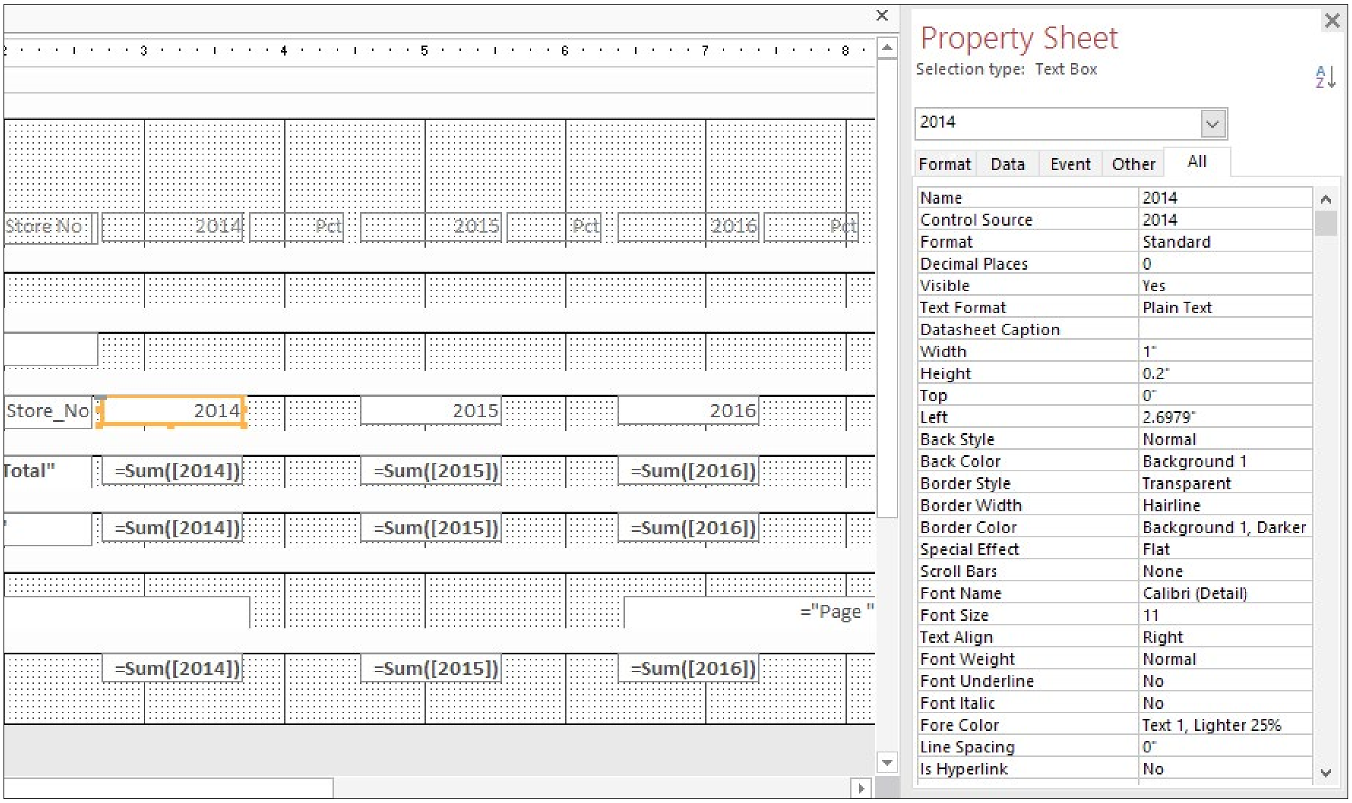
Region Name	City	Store No	2014	Pct	2015	Pct	2016	Pct	20
Northern Region									
	Jersey City								
		1002	52,000		58,000		60,000		63,00
		1034	92,000		102,000		105,000		110,00
		1040	108,000		120,000		124,000		130,00
	Jersey City Total		252,000		280,000		289,000		303,00
	New York								
		1001	81,000		90,000		93,000		98,00
		1018	82,000		91,000		94,000		99,00
		1055	92,000		102,000		105,000		111,00
	New York Total		255,000		283,000		292,000		308,00
	Philadelphia								
		1005	104,000		115,000		119,000		125,00
		1009	77,000		85,000		88,000		93,00
		1012	99,000		110,000		113,000		119,00
		1024	38,000		42,000		43,000		45,00
		1032	94,000		104,000		107,000		113,00

Figure 9.10

It may take you awhile to get to this point. Make sure that you go back and forth between Design View and Print Preview to make sure the settings are working as anticipated. If you think you need to change your report up a little, feel free to do so. I'm not so interested in your completing this example EXACTLY as mine, but rather that you learn how to design your own report the way you like it. I do, however, want you to learn the basics, so for this exercise try to stay as close to my example as possible.

Now you need to insert the text boxes for the calculations. These calculations are just like you would do in Excel, except that you refer to the name of the text box instead of the cell address. The first text box you'll create is to calculate the 2014 percent contributed for each city, but first you need to look at the names of the text boxes you need to use in our calculation.

9. *In* **Design View***, click on the* **2014** *text box in the* **Detail** *section and view the* **Properties** *of that box.*



Property Sheet

Selection type: Text Box

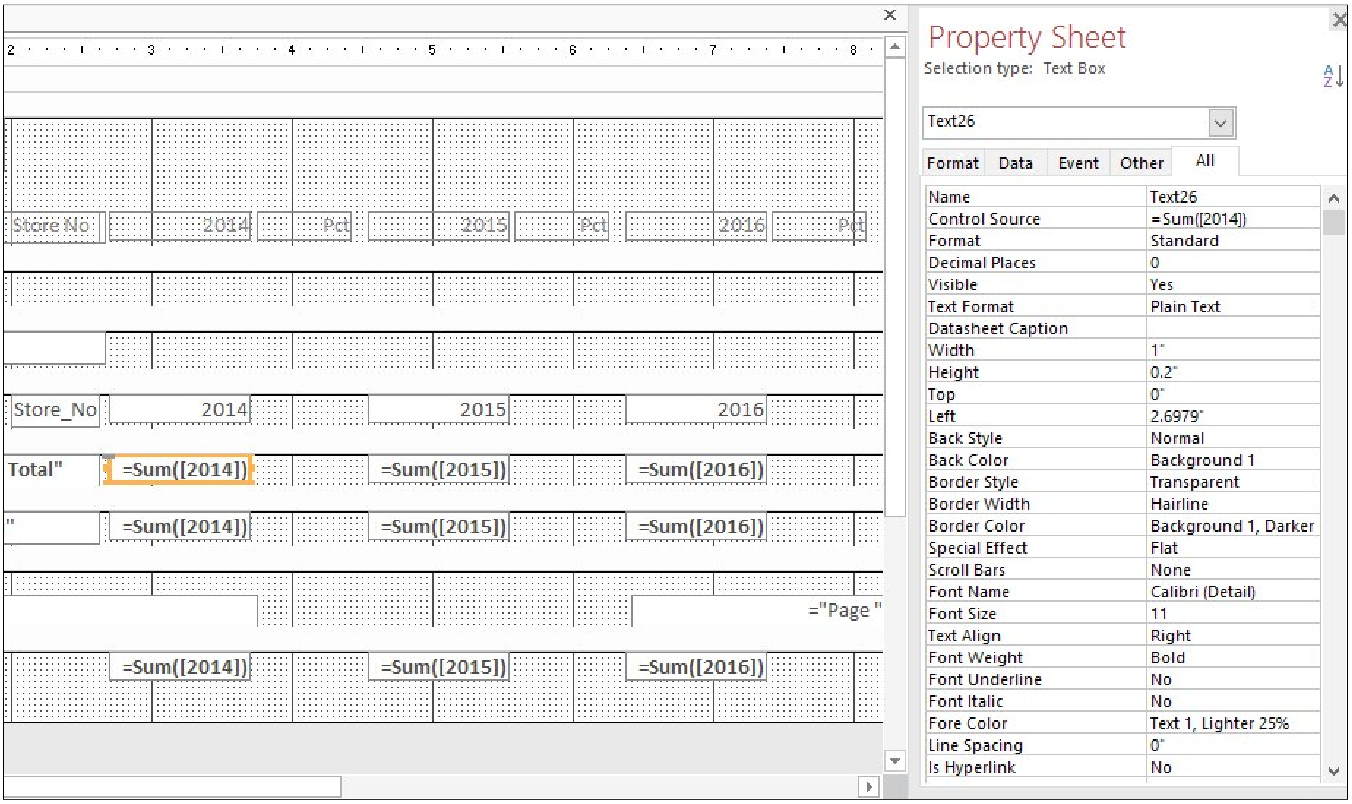
2014

Name	2014
Control Source	2014
Format	Standard
Decimal Places	0
Visible	Yes
Text Format	Plain Text
Datasheet Caption	
Width	1"
Height	0.2"
Top	0"
Left	2.6979"
Back Style	Normal
Back Color	Background 1
Border Style	Transparent
Border Width	Hairline
Border Color	Background 1, Darker
Special Effect	Flat
Scroll Bars	None
Font Name	Calibri (Detail)
Font Size	11
Text Align	Right
Font Weight	Normal
Font Underline	No
Font Italic	No
Fore Color	Text 1, Lighter 25%
Line Spacing	0"
Is Hyperlink	No

Figure 9.11

The name of the text box is "2014". When you use this text box in a calculation, simply refer to that name in brackets. This text box will be the numerator in our first calculation.

10. *Click on the* **Sum([2014])** *text box in the* **City Footer** *section and look at its* **Properties**.



Property Sheet

Selection type: Text Box

Text26

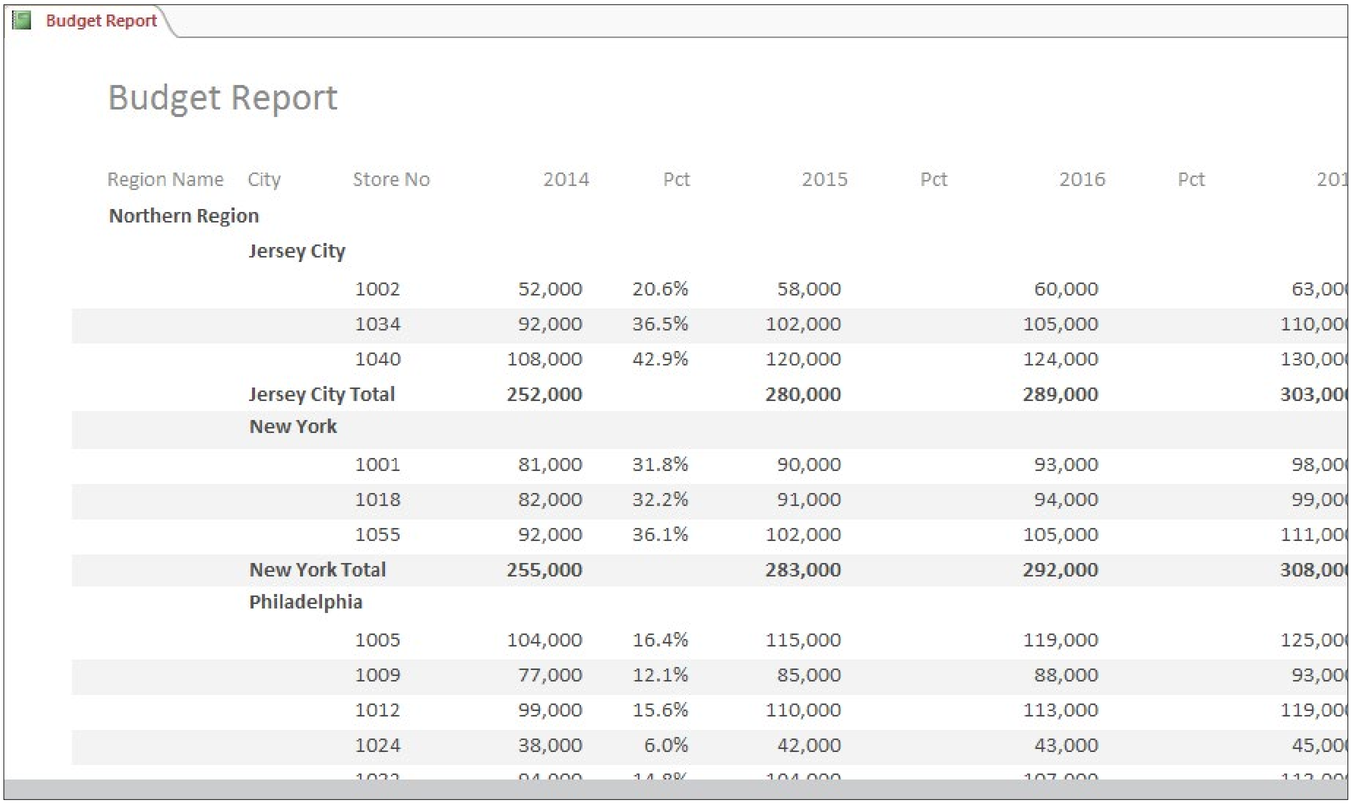
Name	Text26
Control Source	=Sum([2014])
Format	Standard
Decimal Places	0
Visible	Yes
Text Format	Plain Text
Datasheet Caption	
Width	1"
Height	0.2"
Top	0"
Left	2.6979"
Back Style	Normal
Back Color	Background 1
Border Style	Transparent
Border Width	Hairline
Border Color	Background 1, Darker
Special Effect	Flat
Scroll Bars	None
Font Name	Calibri (Detail)
Font Size	11
Text Align	Right
Font Weight	Bold
Font Underline	No
Font Italic	No
Fore Color	Text 1, Lighter 25%
Line Spacing	0"
Is Hyperlink	No

Figure 9.12

This text box in my report was named Text26. This text box will be the denominator in the calculation.

> ***Important Note***: *The names of the* **Text boxes** *in my report may be different from your report. As such, you will need to modify your formulas to reflect the names of the text boxes in your particular report.*

11. *In* **Design View**, *create a* **text box** *and place it under the* **2014 Pct** *label in the* **Detail** *section.*
12. **Delete** *the* **label** *created for the new text box.*
13. *In the new* **text box**, *input the formula* **=[2014]/[Text26]** *(or the name of the text box in your report).*
14. **Format** *the new text box as* **Percent** *with* **one decimal place**.
15. *Make sure there are* **no borders** *around the text box, it is right-aligned, and* **View** *the report.*



Budget Report

Region Name	City	Store No	2014	Pct	2015	Pct	2016	Pct	201
Northern Region									
	Jersey City								
		1002	52,000	20.6%	58,000		60,000		63,00
		1034	92,000	36.5%	102,000		105,000		110,00
		1040	108,000	42.9%	120,000		124,000		130,00
	Jersey City Total		**252,000**		**280,000**		**289,000**		**303,00**
	New York								
		1001	81,000	31.8%	90,000		93,000		98,00
		1018	82,000	32.2%	91,000		94,000		99,00
		1055	92,000	36.1%	102,000		105,000		111,00
	New York Total		**255,000**		**283,000**		**292,000**		**308,00**
	Philadelphia								
		1005	104,000	16.4%	115,000		119,000		125,00
		1009	77,000	12.1%	85,000		88,000		93,00
		1012	99,000	15.6%	110,000		113,000		119,00
		1024	38,000	6.0%	42,000		43,000		45,00
		1022	94,000	14.8%	104,000		107,000		112,00

Figure 9.13

You should see these formatted numbers appearing to the right of the 2014 column as in the figure above. If not, you did something wrong and need to correct it. Check Print Preview as well since Report View occasionally does not display data that is otherwise there.

16. *Make sure the font size and style of the new* **Text box** *is the same as the* **2014** *text box. Resize as necessary. Be sure to* **Bold** *the Footer section* **Pct** *calculations.*
17. *Repeat the same process to create the calculations for the* **City**, **Region**, *and* **Company** *contributions (the Company contribution will be 100%) for* **Year 2014**.

Remember that the calculation for the city would be the city total budget divided by the budget for the region. The region percent is the region budget divided by the company's budget.

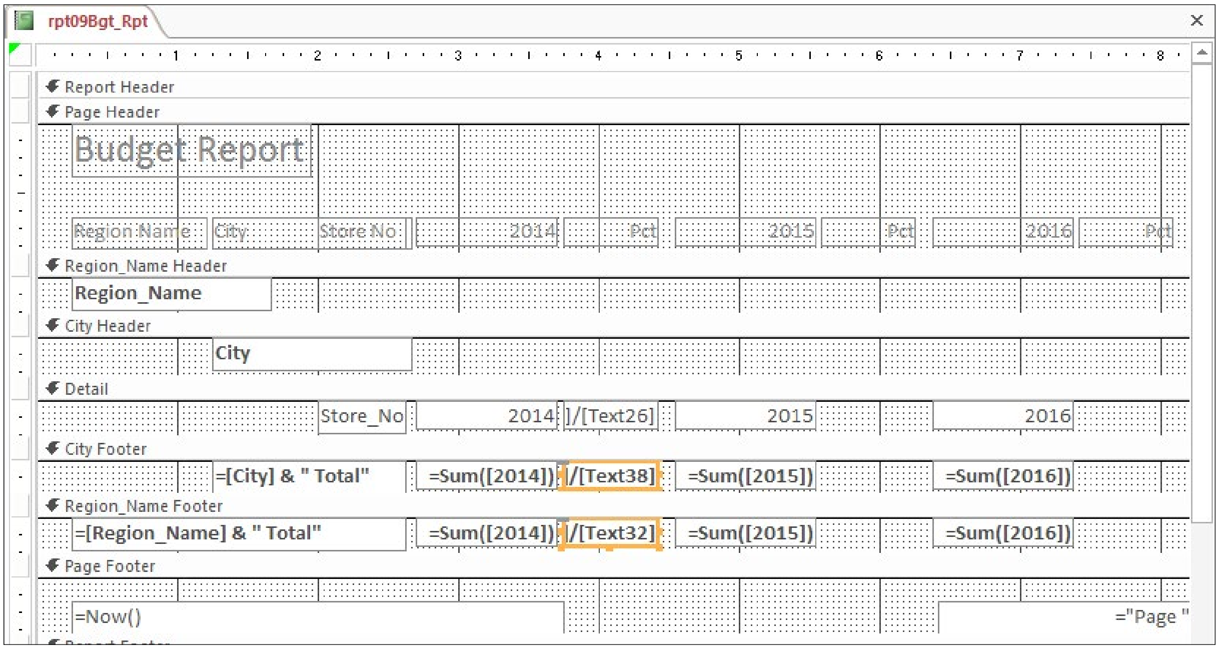
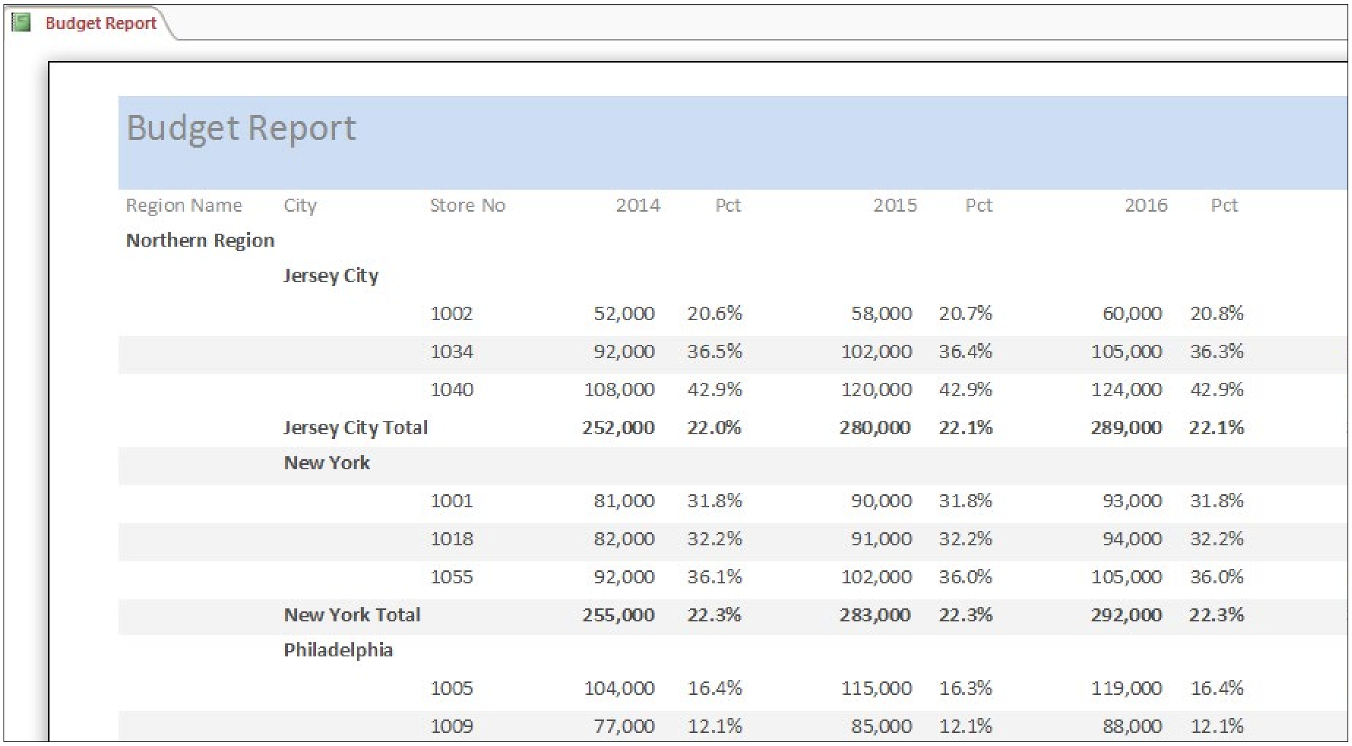


Figure 9.14

In my report, the calculation for the 2014 City Pct was =([Text26])/[Text38], the Region Pct was =([Text38])/[Text32], and for the Report Total was =Sum([2014])/[Text32]. Remember, the calculations and text box names in your report may be different, so you will have to do your own calculations to make the numbers correct. Check the figures for 2014 in your report with these figures:

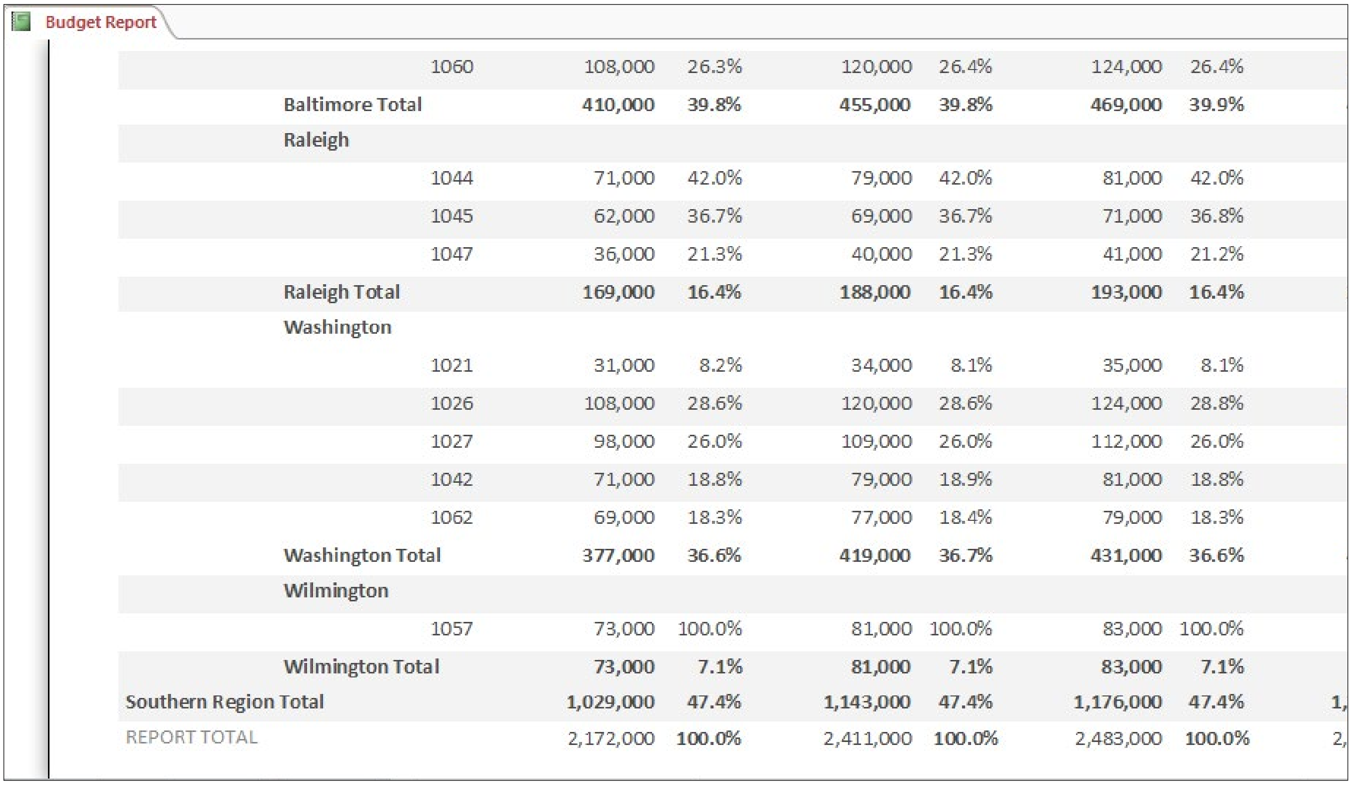


Budget Report

Region Name	City	Store No	2014	Pct	2015	Pct	2016	Pct
Northern Region								
	Jersey City							
		1002	52,000	20.6%	58,000	20.7%	60,000	20.8%
		1034	92,000	36.5%	102,000	36.4%	105,000	36.3%
		1040	108,000	42.9%	120,000	42.9%	124,000	42.9%
	Jersey City Total		**252,000**	**22.0%**	**280,000**	**22.1%**	**289,000**	**22.1%**
	New York							
		1001	81,000	31.8%	90,000	31.8%	93,000	31.8%
		1018	82,000	32.2%	91,000	32.2%	94,000	32.2%
		1055	92,000	36.1%	102,000	36.0%	105,000	36.0%
	New York Total		**255,000**	**22.3%**	**283,000**	**22.3%**	**292,000**	**22.3%**
	Philadelphia							
		1005	104,000	16.4%	115,000	16.3%	119,000	16.4%
		1009	77,000	12.1%	85,000	12.1%	88,000	12.1%

Figure 9.15

Here are some numbers on the second page of the report.

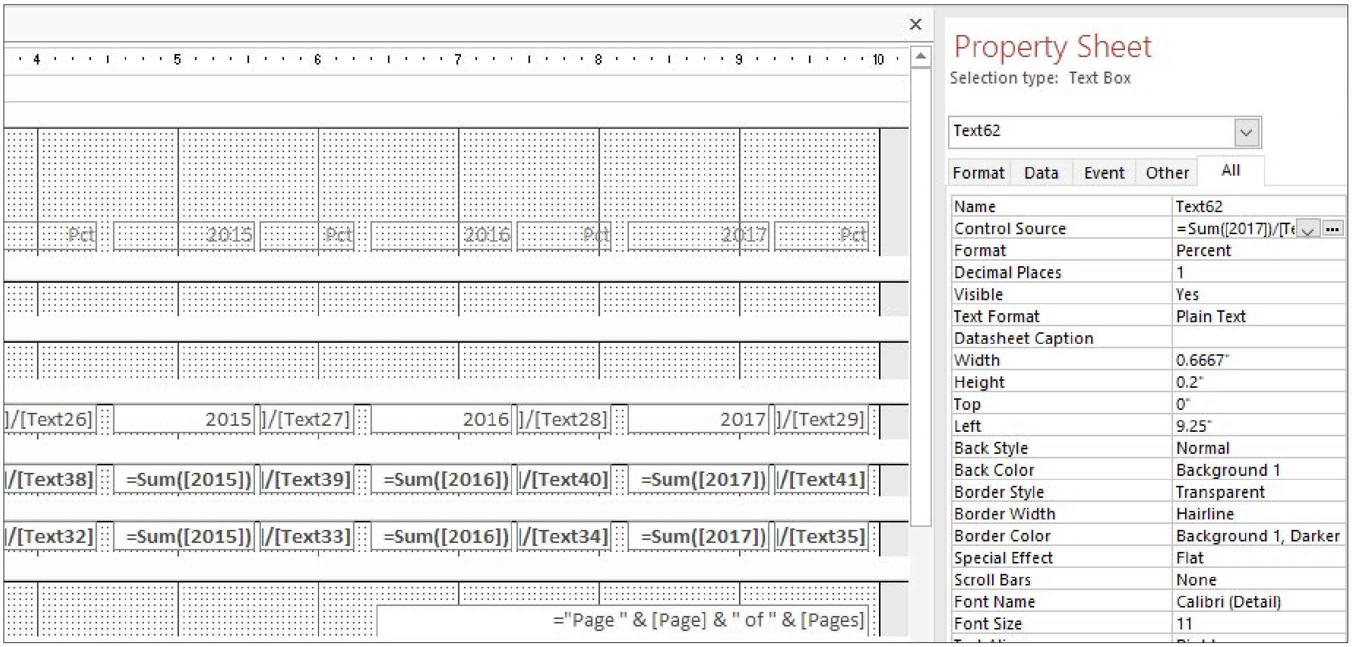


Budget Report

		1060	108,000	26.3%	120,000	26.4%	124,000	26.4%	
	Baltimore Total		**410,000**	**39.8%**	**455,000**	**39.8%**	**469,000**	**39.9%**	
	Raleigh								
		1044	71,000	42.0%	79,000	42.0%	81,000	42.0%	
		1045	62,000	36.7%	69,000	36.7%	71,000	36.8%	
		1047	36,000	21.3%	40,000	21.3%	41,000	21.2%	
	Raleigh Total		**169,000**	**16.4%**	**188,000**	**16.4%**	**193,000**	**16.4%**	
	Washington								
		1021	31,000	8.2%	34,000	8.1%	35,000	8.1%	
		1026	108,000	28.6%	120,000	28.6%	124,000	28.8%	
		1027	98,000	26.0%	109,000	26.0%	112,000	26.0%	
		1042	71,000	18.8%	79,000	18.9%	81,000	18.8%	
		1062	69,000	18.3%	77,000	18.4%	79,000	18.3%	
	Washington Total		**377,000**	**36.6%**	**419,000**	**36.7%**	**431,000**	**36.6%**	
	Wilmington								
		1057	73,000	100.0%	81,000	100.0%	83,000	100.0%	
	Wilmington Total		**73,000**	**7.1%**	**81,000**	**7.1%**	**83,000**	**7.1%**	
Southern Region Total			**1,029,000**	**47.4%**	**1,143,000**	**47.4%**	**1,176,000**	**47.4%**	**1,**
REPORT TOTAL			2,172,000	**100.0%**	2,411,000	**100.0%**	2,483,000	**100.0%**	2,

Figure 9.16

18. *Repeat the same process to create the calculations for the Years* **2015**, **2016**, *and* **2017**.



]/[Text26] | 2015 |]/[Text27] | 2016 |]/[Text28] | 2017 |]/[Text29]

|/[Text38] | =Sum([2015]) | |/[Text39] | =Sum([2016]) | |/[Text40] | =Sum([2017]) | |/[Text41]

|/[Text32] | =Sum([2015]) | |/[Text33] | =Sum([2016]) | |/[Text34] | =Sum([2017]) | |/[Text35]

="Page " & [Page] & " of " & [Pages]

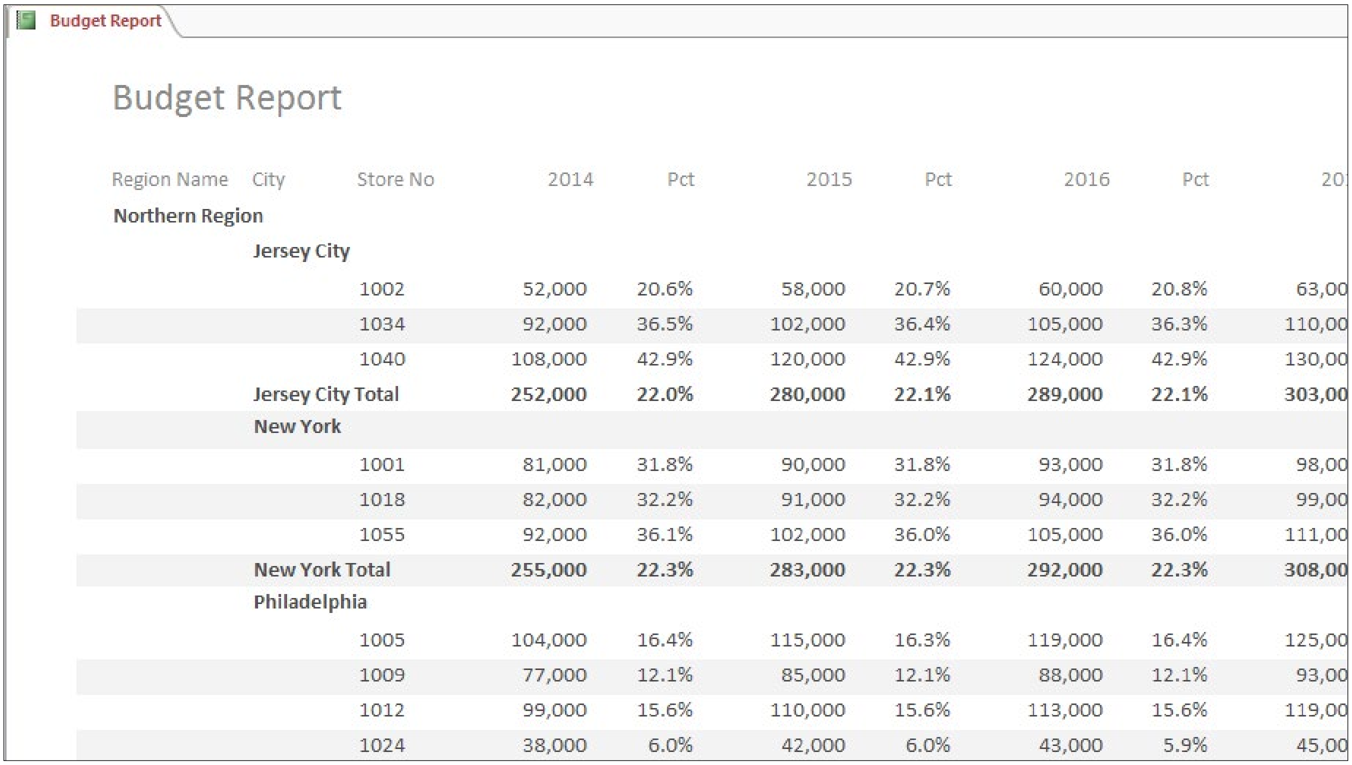
Property Sheet

Selection type: Text Box

Text62

Name	Text62
Control Source	=Sum([2017])/[Te
Format	Percent
Decimal Places	1
Visible	Yes
Text Format	Plain Text
Datasheet Caption	
Width	0.6667"
Height	0.2"
Top	0"
Left	9.25"
Back Style	Normal
Back Color	Background 1
Border Style	Transparent
Border Width	Hairline
Border Color	Background 1, Darker
Special Effect	Flat
Scroll Bars	None
Font Name	Calibri (Detail)
Font Size	11

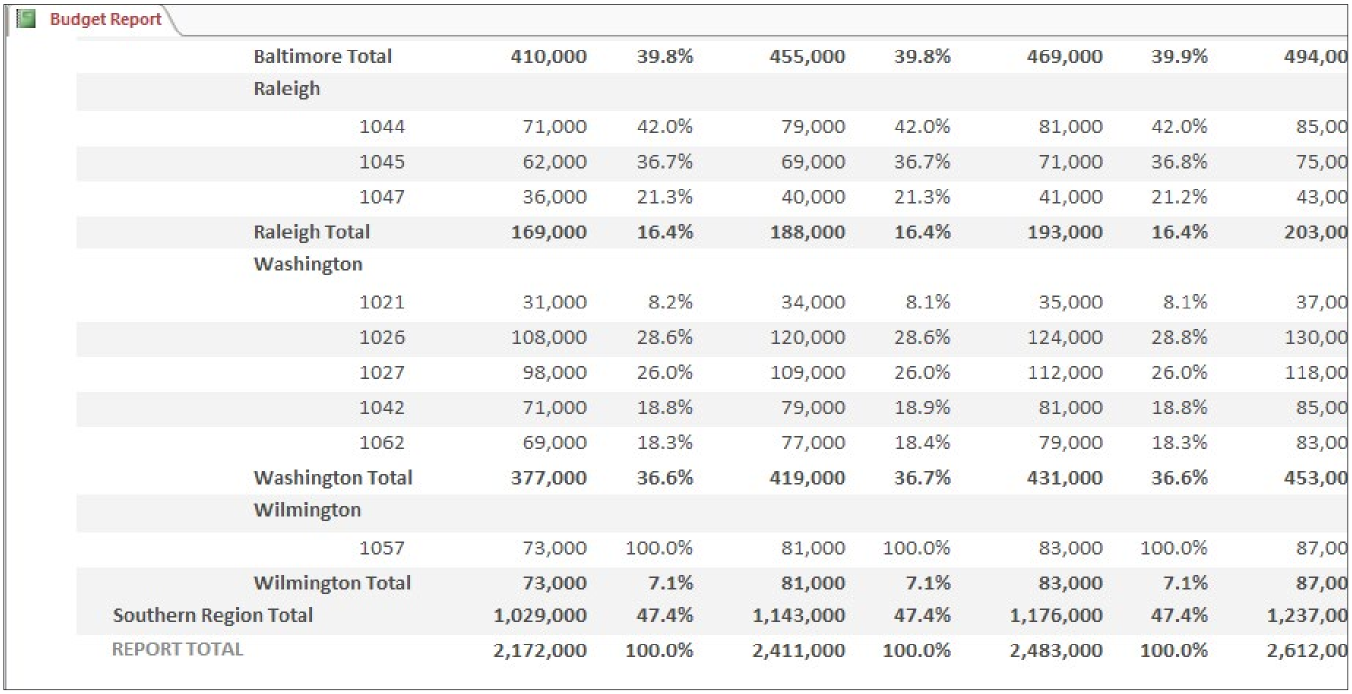
Figure 9.17



Budget Report

Region Name	City	Store No	2014	Pct	2015	Pct	2016	Pct	20
Northern Region									
	Jersey City								
		1002	52,000	20.6%	58,000	20.7%	60,000	20.8%	63,00
		1034	92,000	36.5%	102,000	36.4%	105,000	36.3%	110,00
		1040	108,000	42.9%	120,000	42.9%	124,000	42.9%	130,00
	Jersey City Total		**252,000**	**22.0%**	**280,000**	**22.1%**	**289,000**	**22.1%**	**303,00**
	New York								
		1001	81,000	31.8%	90,000	31.8%	93,000	31.8%	98,00
		1018	82,000	32.2%	91,000	32.2%	94,000	32.2%	99,00
		1055	92,000	36.1%	102,000	36.0%	105,000	36.0%	111,00
	New York Total		**255,000**	**22.3%**	**283,000**	**22.3%**	**292,000**	**22.3%**	**308,00**
	Philadelphia								
		1005	104,000	16.4%	115,000	16.3%	119,000	16.4%	125,00
		1009	77,000	12.1%	85,000	12.1%	88,000	12.1%	93,00
		1012	99,000	15.6%	110,000	15.6%	113,000	15.6%	119,00
		1024	38,000	6.0%	42,000	6.0%	43,000	5.9%	45,00

Figure 9.18



	Baltimore Total		**410,000**	**39.8%**	**455,000**	**39.8%**	**469,000**	**39.9%**	**494,00**
	Raleigh								
		1044	71,000	42.0%	79,000	42.0%	81,000	42.0%	85,00
		1045	62,000	36.7%	69,000	36.7%	71,000	36.8%	75,00
		1047	36,000	21.3%	40,000	21.3%	41,000	21.2%	43,00
	Raleigh Total		**169,000**	**16.4%**	**188,000**	**16.4%**	**193,000**	**16.4%**	**203,00**
	Washington								
		1021	31,000	8.2%	34,000	8.1%	35,000	8.1%	37,00
		1026	108,000	28.6%	120,000	28.6%	124,000	28.8%	130,00
		1027	98,000	26.0%	109,000	26.0%	112,000	26.0%	118,00
		1042	71,000	18.8%	79,000	18.9%	81,000	18.8%	85,00
		1062	69,000	18.3%	77,000	18.4%	79,000	18.3%	83,00
	Washington Total		**377,000**	**36.6%**	**419,000**	**36.7%**	**431,000**	**36.6%**	**453,00**
	Wilmington								
		1057	73,000	100.0%	81,000	100.0%	83,000	100.0%	87,00
	Wilmington Total		**73,000**	**7.1%**	**81,000**	**7.1%**	**83,000**	**7.1%**	**87,00**
Southern Region Total			**1,029,000**	**47.4%**	**1,143,000**	**47.4%**	**1,176,000**	**47.4%**	**1,237,00**
REPORT TOTAL			**2,172,000**	**100.0%**	**2,411,000**	**100.0%**	**2,483,000**	**100.0%**	**2,612,00**

Figure 9.19

19. **Save** *and close* **rpt09Bgt_Rpt**.

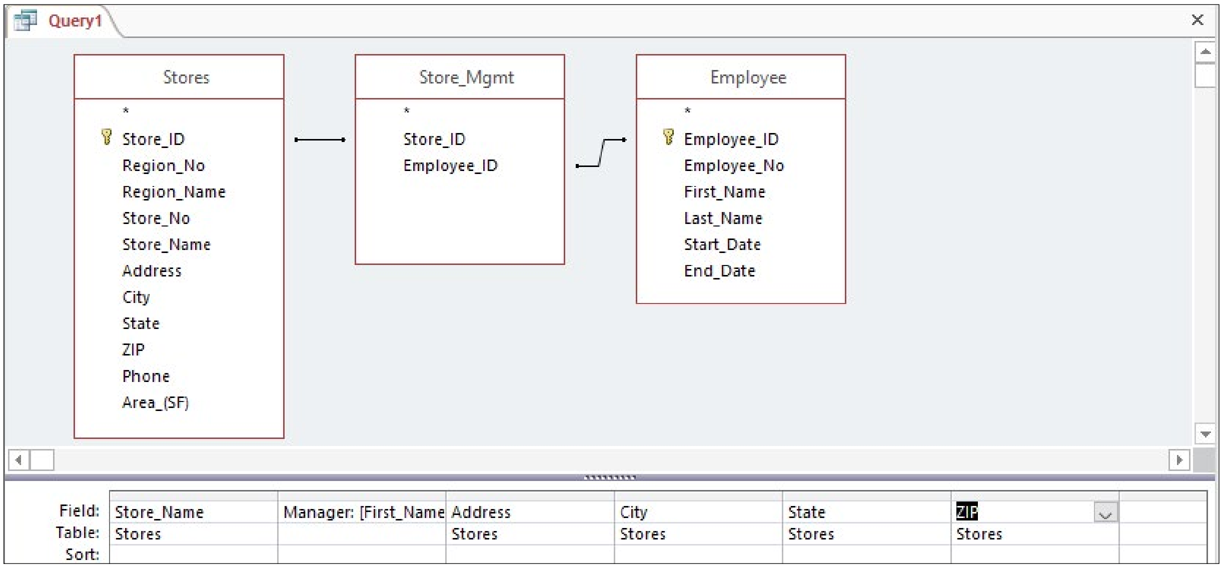
Wow! You did it! That wasn't too hard, was it? I guess it was, but you're a better person for going through all of that. I promise that if you made a few mistakes, all the better. I assume you had to play around with some of the formatting or placing of the text boxes and labels to make it come out just right. In the reporting world, there are a billion and one ways to do these kinds of reports. This is just one way, but it establishes a basis of how you can do reports in Access and other software packages. If you can develop a report like this in Access, I would say that you can now use just about any reporting package out there.

The Mailing Label Report

There is just one more kind of report I want to review with you in this chapter. It's one that is extremely useful in all businesses – mailing labels. I like to use Access for making mailing labels because I can tie the addresses and names of people to a database and the list is always updated. It's also very easy to learn how to do, once you have experience in creating reports.

In this report, you have been asked to create mailing labels with the name of the store on the first line, the manager's name on the second line, the store address on the third line, and the city, state, and ZIP code on the fourth line. The first thing we have to do it to create a query that gives us all of that data. The Stores table gives the names and addresses of the stores, and that is most of the data we need. The Store_Mgmt table is a record of who currently manages each store. Since that table is composed solely of IDs, we have to create joins to other tables to get the right data. Each record in the query results will be the data for one store. Let's get started.

1. ***Create** a new query, and bring in the **Stores**, **Store_Mgmt**, and **Employee** tables.*
2. *Create joins on **Store_ID** and **Employee_ID**.*
3. *Bring in **Store_Name** as the first field, concatenate **First_Name** and **Last_Name** as a field called **Manager**, then bring in **Address**, **City**, **State**, and **ZIP**.*



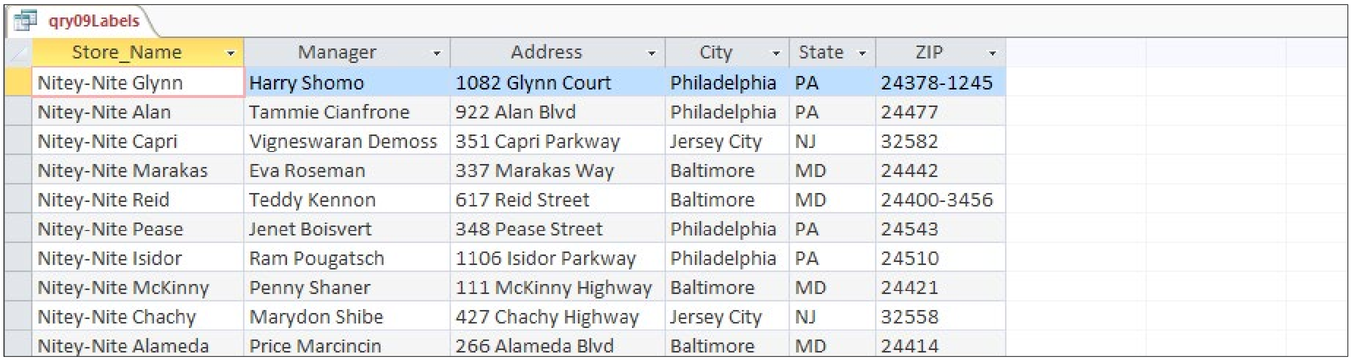
Field:	Store_Name	Manager: [First_Name	Address	City	State	ZIP
Table:	Stores		Stores	Stores	Stores	Stores
Sort:						

Figure 9.20

4. ***Save** the query as **qry09Labels** and **Run** it. Adjust field widths, as needed.*

As a note, a well-designed database has the joins between tables already established in the Relationships view. We created two such relationship in this database, and when you brought in the Store_Mgmt and Employee table, it should have automatically created that join in the Query Design. You then had to create the join between the Stores and Store_Mgmt table manually, which could have already been done in the Relationships view.

You should get the following record set:



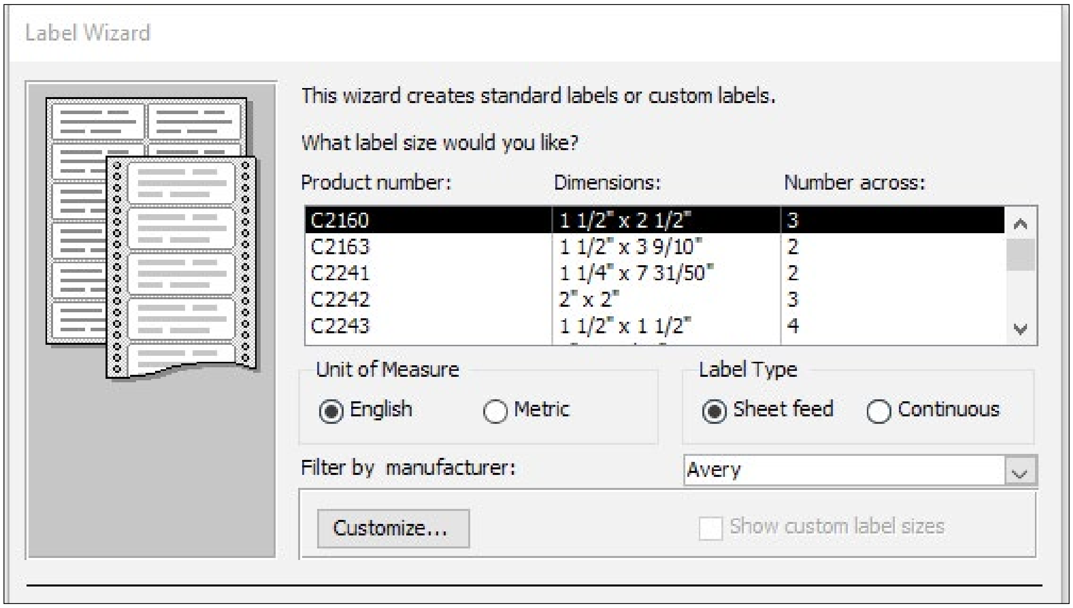
qry09Labels

Store_Name	Manager	Address	City	State	ZIP
Nitey-Nite Glynn	Harry Shomo	1082 Glynn Court	Philadelphia	PA	24378-1245
Nitey-Nite Alan	Tammie Cianfrone	922 Alan Blvd	Philadelphia	PA	24477
Nitey-Nite Capri	Vigneswaran Demoss	351 Capri Parkway	Jersey City	NJ	32582
Nitey-Nite Marakas	Eva Roseman	337 Marakas Way	Baltimore	MD	24442
Nitey-Nite Reid	Teddy Kennon	617 Reid Street	Baltimore	MD	24400-3456
Nitey-Nite Pease	Jenet Boisvert	348 Pease Street	Philadelphia	PA	24543
Nitey-Nite Isidor	Ram Pougatsch	1106 Isidor Parkway	Philadelphia	PA	24510
Nitey-Nite McKinny	Penny Shaner	111 McKinny Highway	Baltimore	MD	24421
Nitey-Nite Chachy	Marydon Shibe	427 Chachy Highway	Jersey City	NJ	32558
Nitey-Nite Alameda	Price Marcincin	266 Alameda Blvd	Baltimore	MD	24414

Figure 9.21

5. *Close* **qry09Labels**.
6. *In the* **Navigation Pane**, *make sure that* **qry09Labels** *is selected.*
7. *In the* **Create** *tab in the* **Reports** *group, click on the* **Labels** *icon.*

The Label Wizard opens. You can use just about any type of label manufacturer and the Product number will show up here.



Label Wizard

This wizard creates standard labels or custom labels.

What label size would you like?

Product number:	Dimensions:	Number across:
C2160	1 1/2" x 2 1/2"	3
C2163	1 1/2" x 3 9/10"	2
C2241	1 1/4" x 7 31/50"	2
C2242	2" x 2"	3
C2243	1 1/2" x 1 1/2"	4

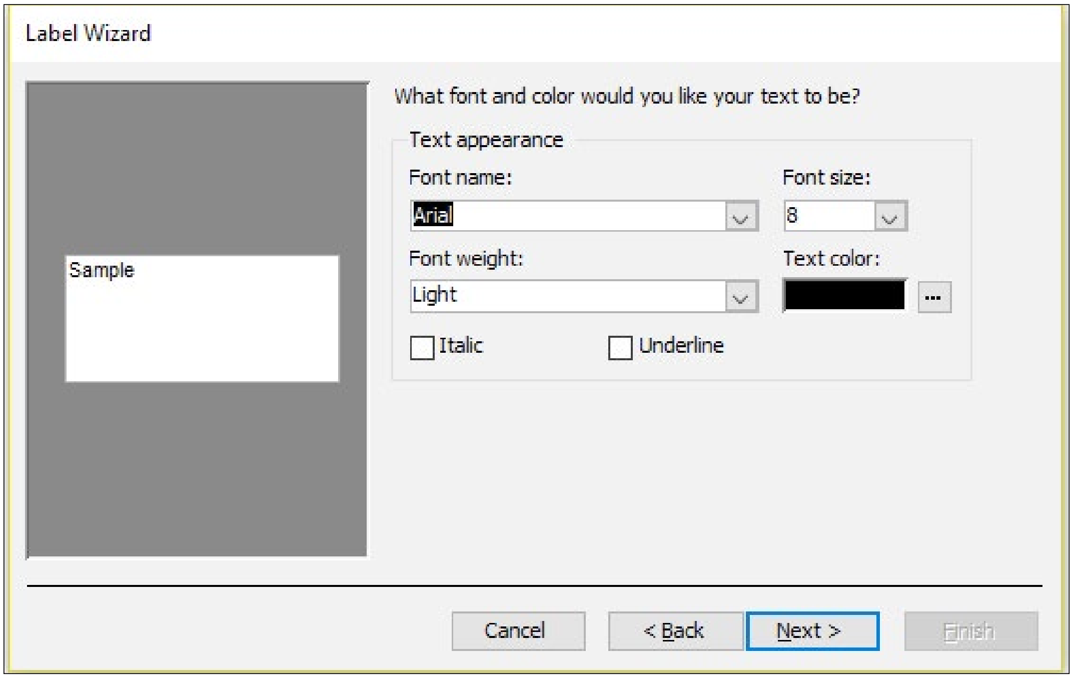
Unit of Measure: ● English ○ Metric

Label Type: ● Sheet feed ○ Continuous

Filter by manufacturer: Avery

Figure 9.22

1. *In the first screen of the* **Label Wizard**, *make sure* **Filter by manufacturer** *is set to* **Avery** *(to use Avery labels), select* **5160** *from the list of products and click* **Next >**.



Label Wizard

What font and color would you like your text to be?

Text appearance

Font name: Arial

Font size: 8

Font weight: Light

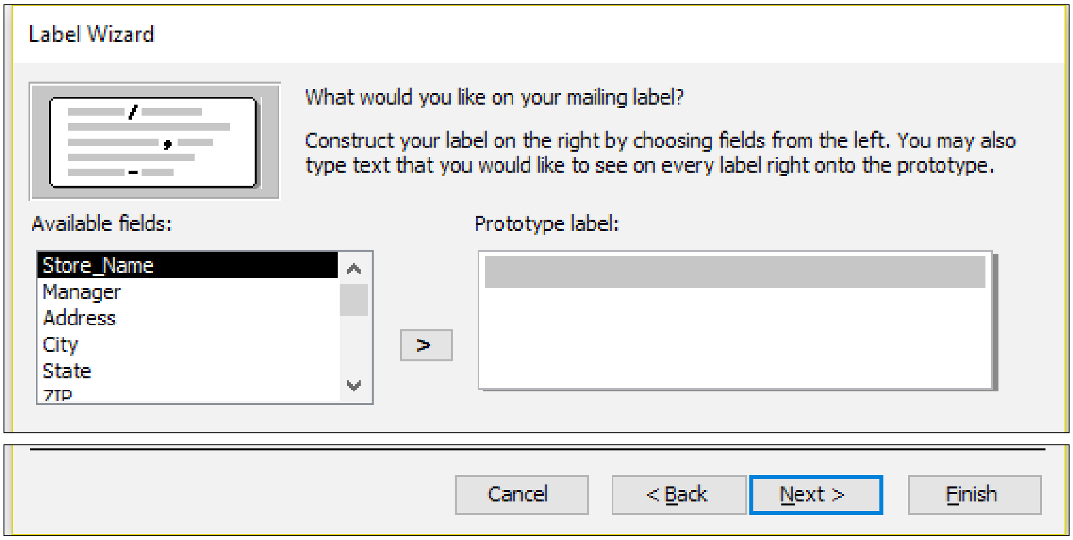
Text color: ■

□ Italic □ Underline

Figure 9.23

2. *Accept these default values in the next step of the wizard and click* **Next >**.

The next step of the wizard is where you design how each label will look.



Label Wizard

What would you like on your mailing label?

Construct your label on the right by choosing fields from the left. You may also type text that you would like to see on every label right onto the prototype.

Available fields:

Store_Name
Manager
Address
City
State
ZIP

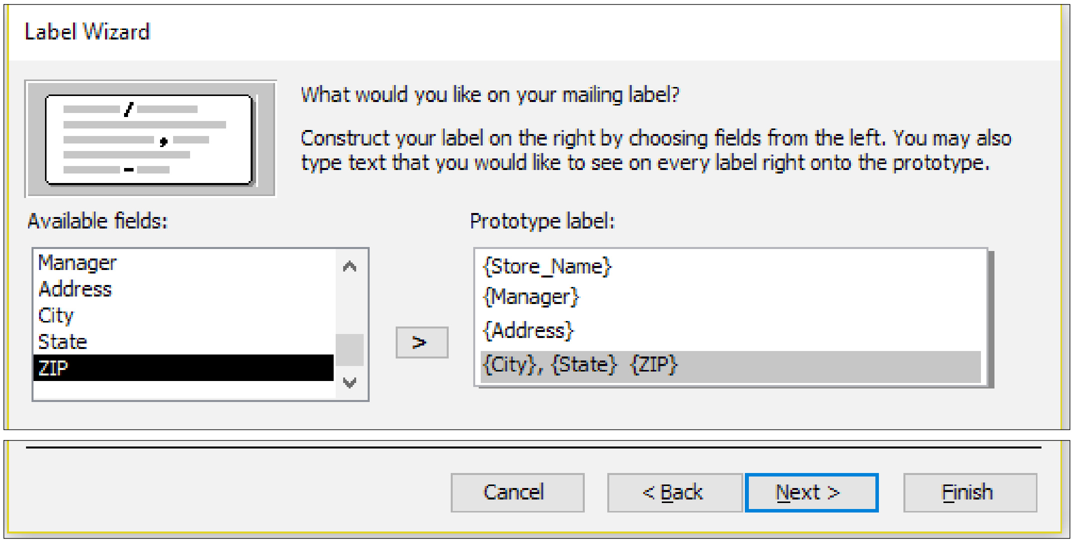
Prototype label:

Figure 9.24

3. *In the* **Available fields:**, *click on* **Store_Name** *and bring that field into the* **Prototype label:**

section by clicking the > *icon.*

4. *After you bring in the* **Store_Name** *field, press the* **[Enter]** *key to move to the next line.*
5. *On the second and third lines, bring in* **Manager** *and* **Address**, *respectively.*
6. *On the fourth line, bring in* **City**, *then type a* **comma** *and a* **space**, *then bring in* **State**, *then* **two spaces**, *and then bring in* **ZIP**.



Label Wizard

What would you like on your mailing label?

Construct your label on the right by choosing fields from the left. You may also type text that you would like to see on every label right onto the prototype.

Available fields:

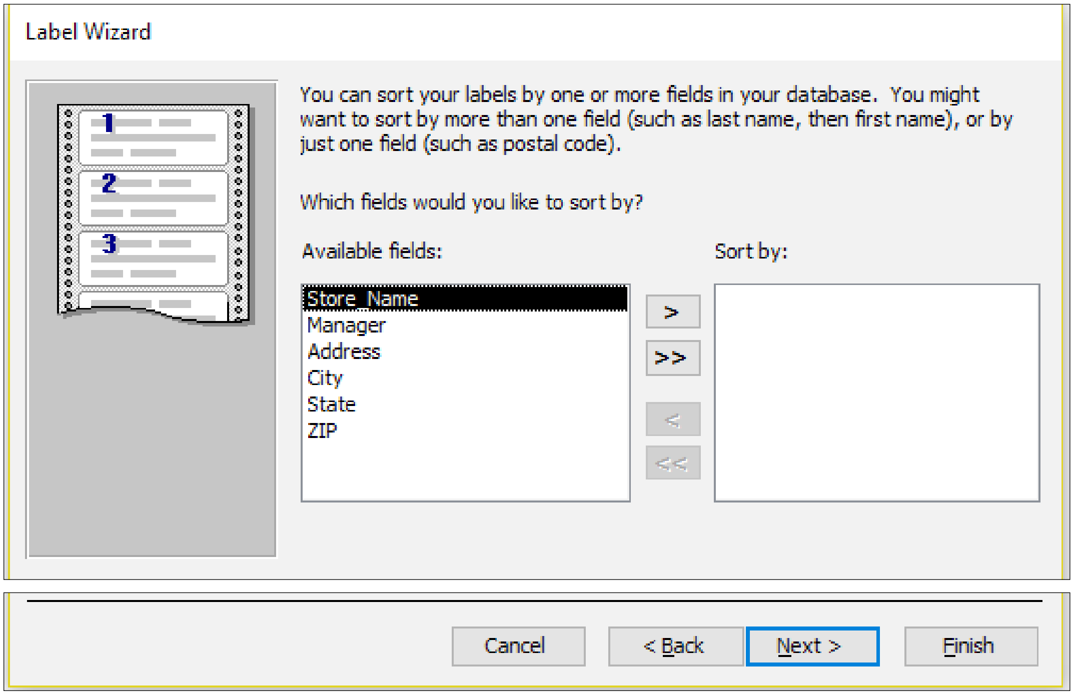
Manager
Address
City
State
ZIP

Prototype label:

{Store_Name}
{Manager}
{Address}
{City}, {State} {ZIP}

Figure 9.25

7. *Click* **Next >**.



Label Wizard

You can sort your labels by one or more fields in your database. You might want to sort by more than one field (such as last name, then first name), or by just one field (such as postal code).

Which fields would you like to sort by?

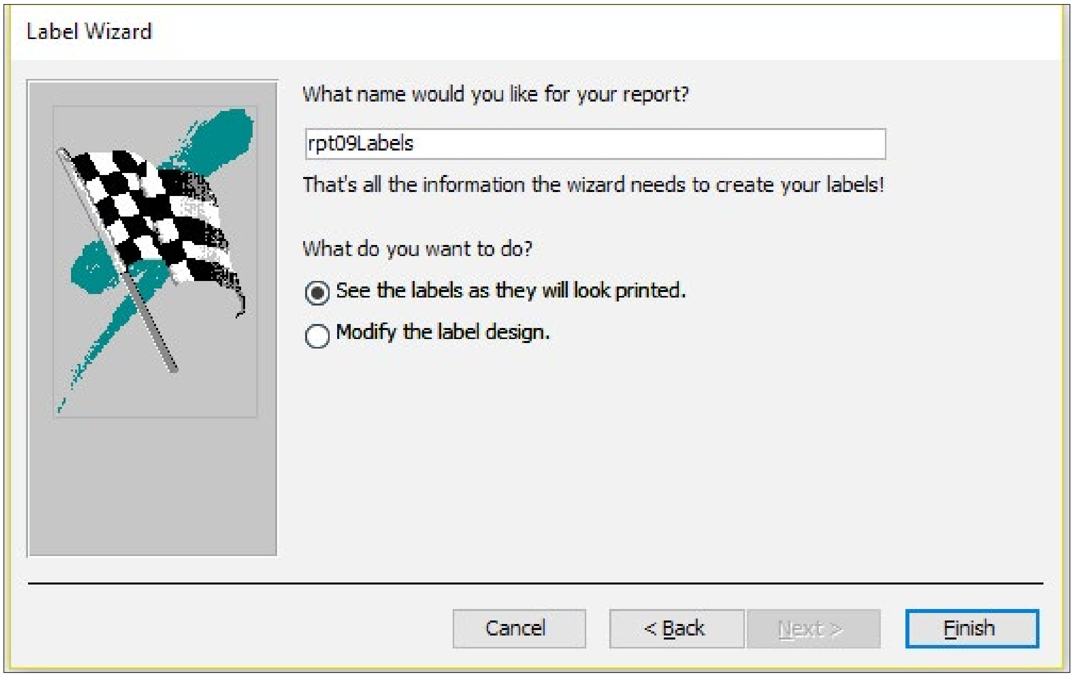
Available fields:

Store_Name
Manager
Address
City
State
ZIP

Sort by:

Figure 9.26

8. *There is no sorting necessary so click* **Next >** *again.*
9. *In the last screen of the wizard, name the report* ***rpt09Labels*** *and click* **Finish**.



Label Wizard

What name would you like for your report?

rpt09Labels

That's all the information the wizard needs to create your labels!

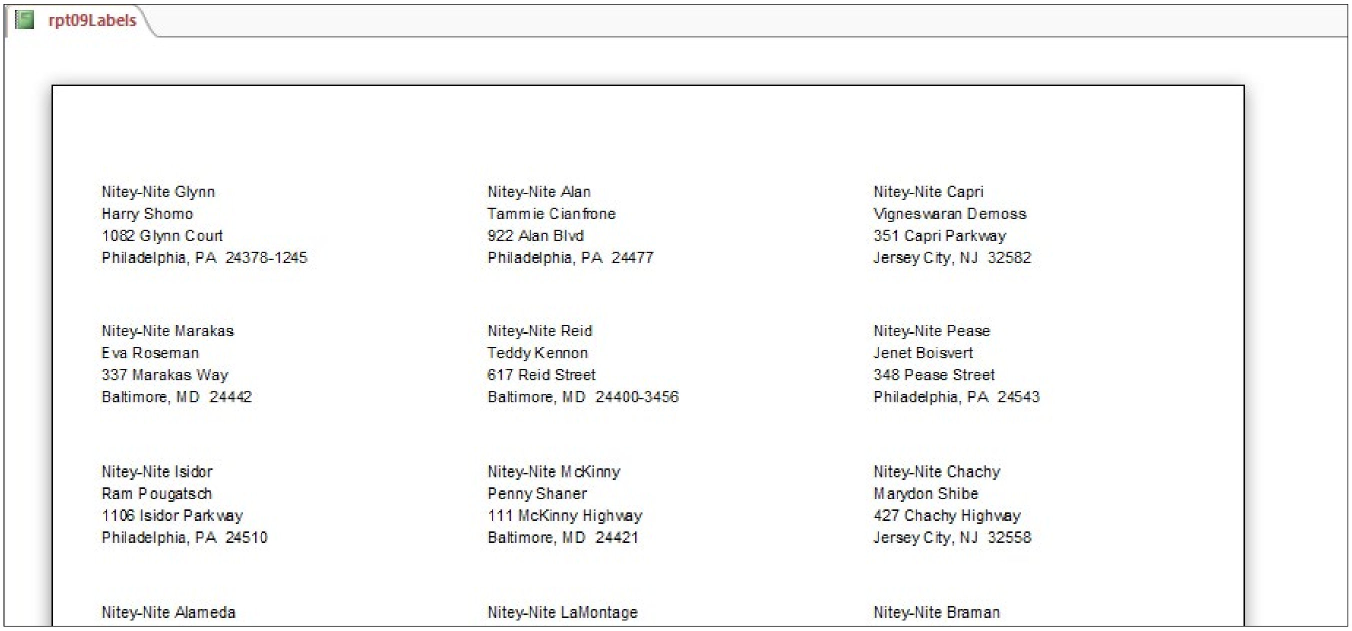
What do you want to do?

● See the labels as they will look printed.

○ Modify the label design.

Figure 9.27

You may get a message that says there is not horizontal space to create the labels, but just click OK and see if it works. If it doesn't, you may have to reduce the font or choose another Avery label type. You can see the Avery label number printed on the box of Avery labels.



rpt09Labels

Nitey-Nite Glynn
Harry Shomo
1082 Glynn Court
Philadelphia, PA 24378-1245

Nitey-Nite Alan
Tammie Cianfrone
922 Alan Blvd
Philadelphia, PA 24477

Nitey-Nite Capri
Vigneswaran Demoss
351 Capri Parkway
Jersey City, NJ 32582

Nitey-Nite Marakas
Eva Roseman
337 Marakas Way
Baltimore, MD 24442

Nitey-Nite Reid
Teddy Kennon
617 Reid Street
Baltimore, MD 24400-3456

Nitey-Nite Pease
Jenet Boisvert
348 Pease Street
Philadelphia, PA 24543

Nitey-Nite Isidor
Ram Pougatsch
1106 Isidor Parkway
Philadelphia, PA 24510

Nitey-Nite McKinny
Penny Shaner
111 McKinny Highway
Baltimore, MD 24421

Nitey-Nite Chachy
Marydon Shibe
427 Chachy Highway
Jersey City, NJ 32558

Nitey-Nite Alameda

Nitey-Nite LaMontage

Nitey-Nite Braman

Figure 9.28

Creating a mailing label report is another very useful tool to have available. I can almost guarantee you will use it many times in your career.

10. **Save** *and close* **rpt09Labels**.

> ***Review Questions****: It is now time to complete the hands-on Review Questions. Log on to www.ExcelCEO.com with your Email and Password, click on the* **Access 2016 and SQL Review Questions, Chapter 9, Section 2 of 2** *option and complete the review questions.*

Conclusion

In this chapter you learned how to make a copy of an existing report and the query that generated it, and used them as a base for a new report. You added new header and footer sections to a report by using the Sorting and Grouping icon. You made the report larger and changed it from Portrait to Landscape orientation. You learned how to work with the Repeat Section to make section headers repeat at page breaks. You calculated lots of subtotals within a report and finally created a report that you can use for mailing labels.

Chapter Exam

You can now go to www.ExcelCEO.com, click on Sign In, log in and take the exam. Make sure that you take the exam on the same device on which you completed the practice files, as some of the questions on the exam may refer to some of the completed projects. Chapter exams are intended to be hands-on.

Access® 2016 and SQL

Complete Self-study Course

ExcelCEO

Chief Excel Officer

CHAPTER TEN — DSN AND EXTERNAL DATA SOURCES

In this chapter, you will:

- Identify and learn how to create a Data Source Name (DSN)
- Recognize a link to an external data source using a DSN
- Determine how to import objects from another Access database
- Identify the different types of data that can be imported or linked to an Access database

CPE Credits possible for this chapter: 2

External Data Sources

Without a doubt, the most challenging issue in reporting and analysis is to get the right data. Once you have the right data, slicing and dicing it in a report or analysis is easy, IF you know what to do. There have been countless projects in my career that I have spent more time trying to get the right data than I have in developing the report or analysis. Getting the right data is critical, and the skills you will learn in this chapter will help you in solving that problem. As linking to an external database is such an important topic, I've dedicated an entire chapter to the matter.

Many Access developers will query information out of a database and import or copy the data into an Access table. The problem with that approach is that once the data changes, the developer must go out to get the data again. You can solve this problem by ***linking*** to the database instead of importing from it. Most companies will not allow non-IT personnel direct access to company data, and rightfully so. However, non-IT employees who have proven their skills can sometimes convince management to grant them Read Only (RO) access, which means that they can access and read the data, but they can't make any changes to it. Read Write (RW) access is the capability to not only read, but also update, insert, and delete data, but that level of access is rarely given.

There are typically three levels of data within an organization: Production, Development, and Test. Production data are the "real" data of the company. In large organizations, there are many controls around this data that the company has certified as "official". Very few individuals have RW access to that data. Development data are used for the development of reports and other analyses. Many times, companies will periodically make a copy of production data and put it in a development environment to allow developers to have access to data that are as close to production as possible. Test data is generally a database that mirrors production and is used in UAT (User Acceptance Testing) right before it is migrated to production. Generally this would include specific subsets of the data. Some companies have only development or test data, but not both, as they can arguably be used for the same purpose.

Companies can have very complex IT systems. Some of the larger and more robust database systems include SAP, Oracle, and Microsoft SQL Server, just to name a few. As it is impractical to train you in all systems, I have decided to use Microsoft SQL Server (pronounced Sequel Server) because: 1) it is a very popular Microsoft-based system, and 2) the concepts I will teach in SQL Server are easily translated into the other systems. SQL Server is a relational database very similar to Access, but is much more robust. Access is a desktop tool, whereas SQL Server is a server-based tool. I like to refer to SQL Server as "Access on steroids".

Generally speaking, you can't develop forms and reports within SQL Server itself (though there are programs to handle this). However, you can use other tools (like Access or Excel) to write reports based on SQL Server data. In this chapter, you will connect an Access database to three tables in SQL Server. You don't even need SQL Server on your computer – you can use just Access.

For the purposes of the exercises in this chapter, we will assume that Nitey-Nite has created a test environment in a SQL Server database system. In this environment, you will connect to the three tables you need to run the report. The beautiful thing about linking to data is that if the data changes, the Access database will be automatically updated. There are generally two ways to connect to data: a DSN connection and a DSN-less connection. DSN stands for Data Source Name. In this chapter, we will use a DSN connection.

> ***Note**: I have had a few students that could not make the connection to the Nitey-Nite database on the hosted server no matter what they tried. Sometimes, when students are taking this course at work, the IT security at their office would not allow them to make external connections. But even at home, some students are unable to make the connection, perhaps because of a setting from their Internet Service Provider (ISP) or on their firewall software. I would encourage you to make all attempts to get the appropriate access to be able to connect to the hosted server I describe in this chapter, either at work or at home. You will use the database created in the chapter as a basis of data in the following two chapters. After you have exhausted all resources trying to make the connection, if you are still unable to do it, you can use a database called* **Static_2016** *I have provided when you download the Nitey_Nite_2016 database. You will find that database in the C:\ExcelCEO\Access 2016 folder. You can use this database with the same tables as the SQL Server tables. Please use that database only as a last resort.*

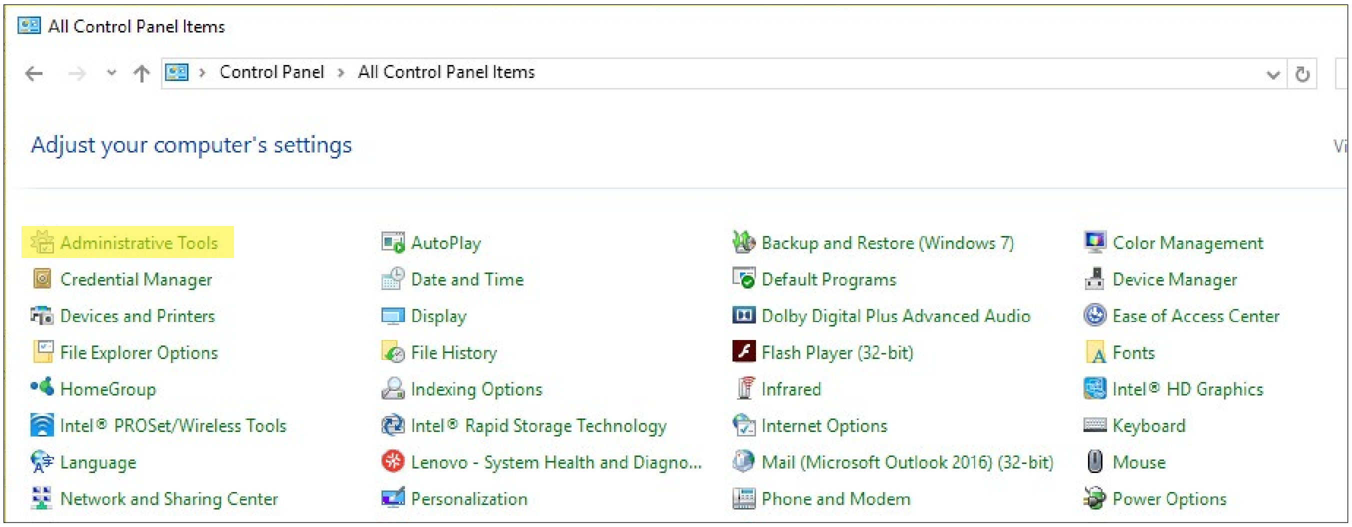
As I mentioned before, SQL Server data is stored on data "servers". You can think of a server as a computer that is always turned on and connected to the World Wide Web, which is the "www" that you so often type (or used to type) in the URL when accessing a website. A SQL Server system can exist in your company's IT infrastructure, or you can use a "hosted server". A hosted server is a server that is owned by a company that rents out databases, usually for a monthly fee. It is usually a good idea to have your SQL Server hosted instead of trying to maintain it yourself, if you have your own small company. It can be very expensive and time-consuming to maintain it yourself, and companies can rent database space for pennies on the dollar for what it would cost to run the same servers themselves. In this chapter, you will connect to a hosted server for the SQL Server tables.

Creating a DSN

The first thing we need to do to be able to connect to the SQL Server tables is to create a DSN. You can think of a DSN as the gateway to retrieve data. There are many types of databases in the world today, and a DSN creates a conduit by which you can access the data. In the exercises that follow, you will create a DSN on your computer. You only have to create a DSN once for each computer that will use it. The steps to creating a DSN can be a little confusing, but I have outlined all of them here in detail.

Make sure that you are connected to the Internet before you create the DSN. If you have a firewall, you may need to change its settings to allow you to connect to an external database. This is generally accomplished by turning off the firewall temporarily. If you are creating the DSN on your work computer and you experience difficulties in making the connection, you may have to contact your system administrator. They may have to open a port for you to connect through, or you may have a firewall issue that prohibits such a connection. Typically, all of these issues can be resolved so you can connect. With those points in mind, let's create your first DSN.

1. *Click on* **Cortana** ○, *type* ***Control Panel*** *in the* **Search bar**, *then click on* **Control Panel Desktop App** *(or just navigate to Control Panel, if you have a preferred way to search for programs/apps on your device).*



All Control Panel Items

Control Panel > All Control Panel Items

Adjust your computer's settings

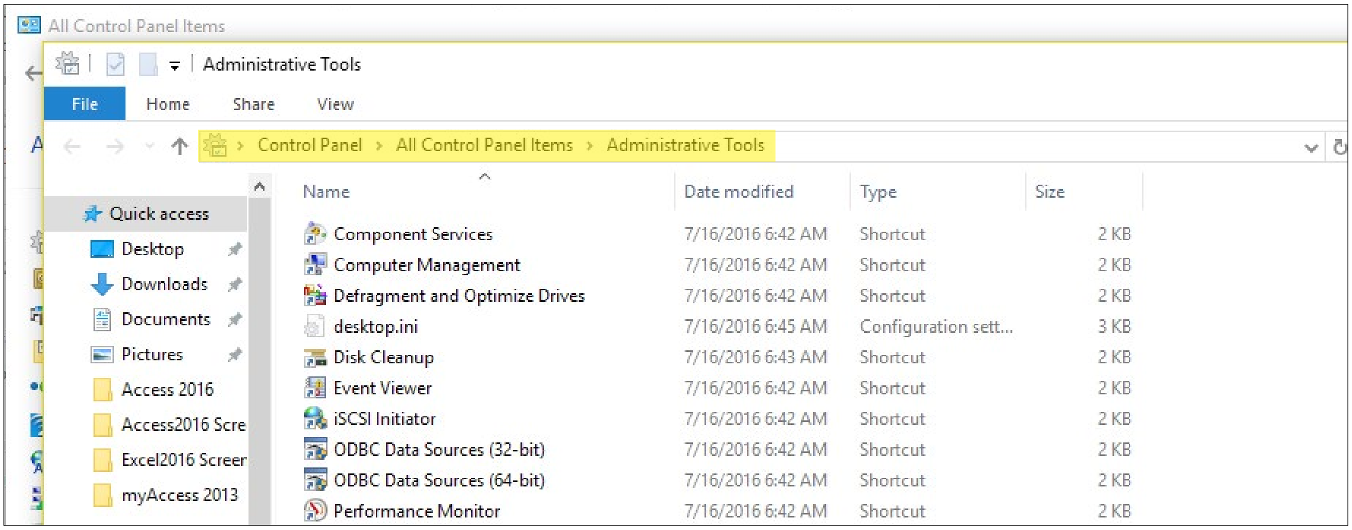
Administrative Tools
AutoPlay
Backup and Restore (Windows 7)
Color Management
Credential Manager
Date and Time
Default Programs
Device Manager
Devices and Printers
Display
Dolby Digital Plus Advanced Audio
Ease of Access Center
File Explorer Options
File History
Flash Player (32-bit)
Fonts
HomeGroup
Indexing Options
Infrared
Intel® HD Graphics
Intel® PROSet/Wireless Tools
Intel® Rapid Storage Technology
Internet Options
Keyboard
Language
Lenovo - System Health and Diagno...
Mail (Microsoft Outlook 2016) (32-bit)
Mouse
Network and Sharing Center
Personalization
Phone and Modem
Power Options

Figure 10.1

Your screen should look similar to Figure 10.1. If not, you may have to click on "Small icons" in the View by: drop-down menu on the top-right of the window. Even if you want to use the view you have, the names of the icons/items should be the same.

2. *Click on* **Administrative Tools***.*

> ***Note****: In* **Windows 10** *you may have to click on the* **System and Security** *link to display* **Administrative Tools***.*



All Control Panel Items

Administrative Tools

File Home Share View

Control Panel > All Control Panel Items > Administrative Tools

Quick access
Desktop
Downloads
Documents
Pictures
Access 2016
Access2016 Scre
Excel2016 Screer
myAccess 2013

Name	Date modified	Type	Size
Component Services	7/16/2016 6:42 AM	Shortcut	2 KB
Computer Management	7/16/2016 6:42 AM	Shortcut	2 KB
Defragment and Optimize Drives	7/16/2016 6:42 AM	Shortcut	2 KB
desktop.ini	7/16/2016 6:45 AM	Configuration sett...	3 KB
Disk Cleanup	7/16/2016 6:43 AM	Shortcut	2 KB
Event Viewer	7/16/2016 6:42 AM	Shortcut	2 KB
iSCSI Initiator	7/16/2016 6:42 AM	Shortcut	2 KB
ODBC Data Sources (32-bit)	7/16/2016 6:42 AM	Shortcut	2 KB
ODBC Data Sources (64-bit)	7/16/2016 6:42 AM	Shortcut	2 KB
Performance Monitor	7/16/2016 6:42 AM	Shortcut	2 KB

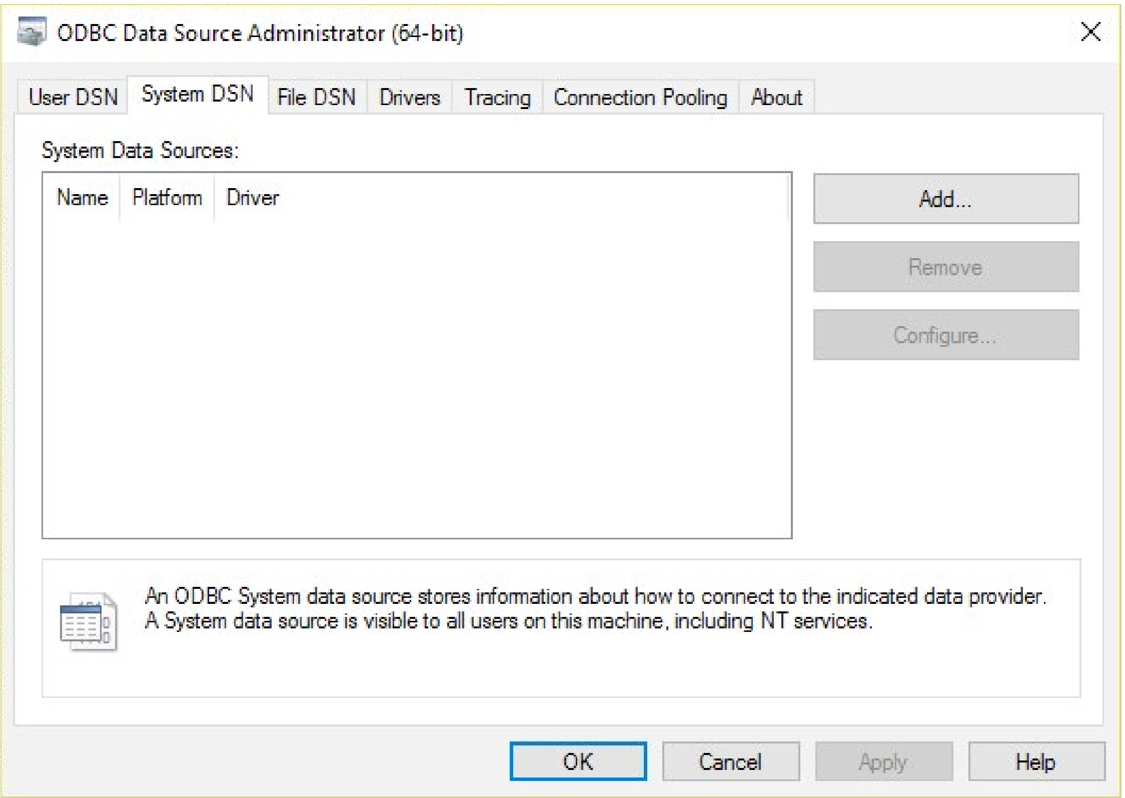
Figure 10.2

3. *Open* **Data Sources (ODBC)** *(this may also display as* **ODBC Data Sources***) (32-bit or 64-bit). To find out which bit version of* **Windows** *your device is using, click on the* **Cortana**

*button, type **System Type**, click the **System Control Panel** icon, and then **Enter**.*

> ***Note**: Many newer devices are using **64-bit versions** of **Windows**. A **32-bit ODBC Data Sources** connection should still be compatible, if you find out later that you chose 32-bit for a 64-bit operating system version. However, a 64-bit version ODBC Data Source connection would not be compatible with a **32-bit operating system**, so it is still worth checking.*

ODBC stands for Open Database Connectivity. It is the tool that allows you to connect to a host of different database types.

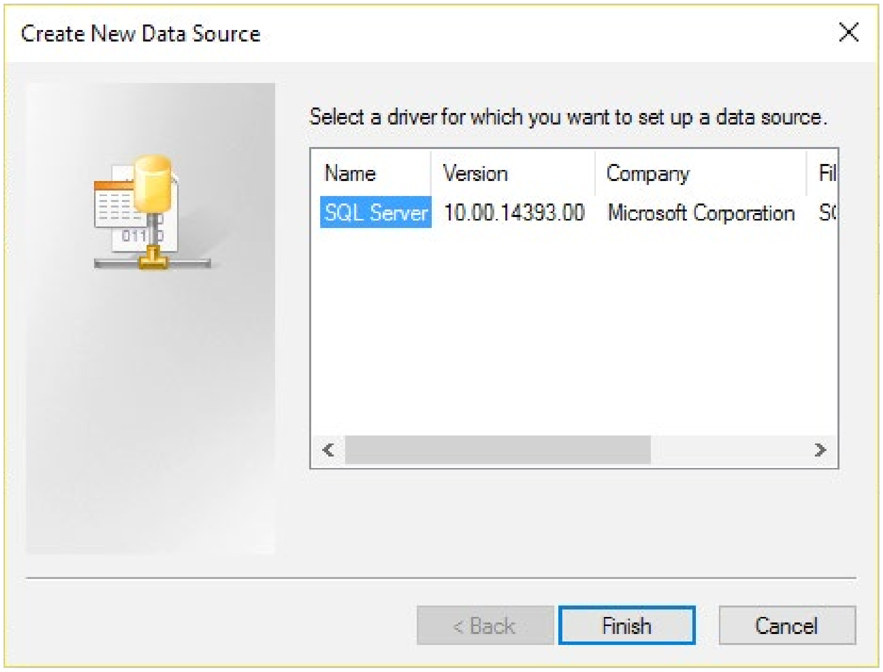


An ODBC System data source stores information about how to connect to the indicated data provider. A System data source is visible to all users on this machine, including NT services.

Figure 10.3

The ODBC Data Source Administrator dialog box appears. If you have no DSNs created on your computer, there will be nothing listed under Name and Driver. If you have one or more DSNs created, they will show up here. In either case, you want to create a DSN to connect to the database name that I will give you, so you need to create a new DSN.

4. *Click on the **User DSN** tab and click on the **Add…** button.*
5. *In the **Create New Data Source** dialog box, scroll down and click on the **SQL Server** option and click **Finish**. (If you are using a 64-bit version, SQL Server may be the only option.)*



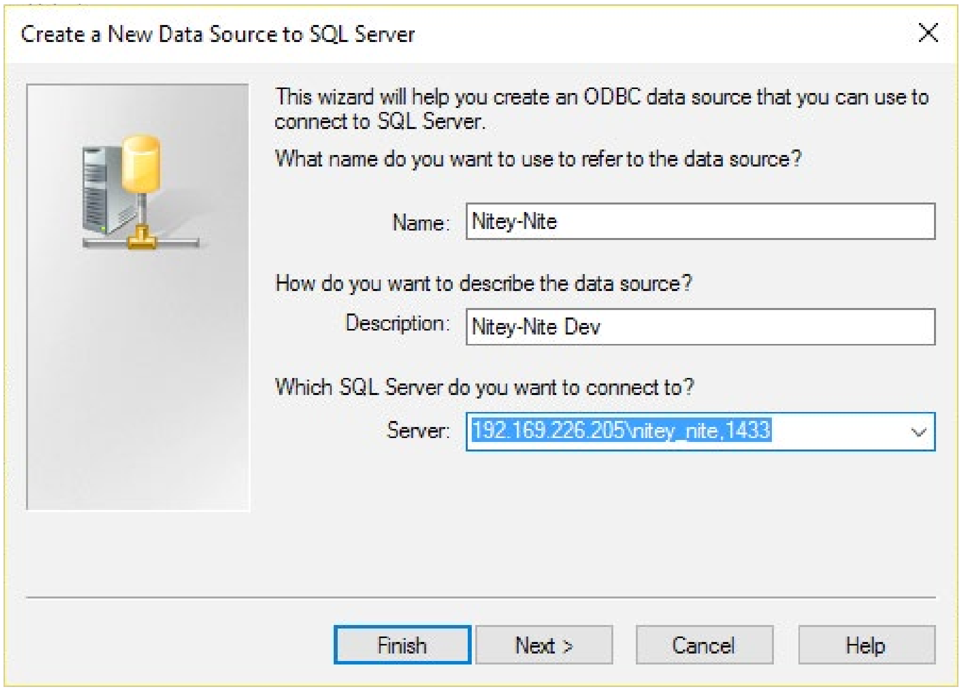
Create New Data Source

Select a driver for which you want to set up a data source.

Name	Version	Company	Fil
SQL Server	10.00.14393.00	Microsoft Corporation	S(

Figure 10.4

In this step of the wizard, you give the DSN a name, description, and tell it which server you will use. The IP address for the server that is available for this exercise is: **192.169.226.205\nitey_nite,1433**.



Create a New Data Source to SQL Server

This wizard will help you create an ODBC data source that you can use to connect to SQL Server.

What name do you want to use to refer to the data source?

Name: Nitey-Nite

How do you want to describe the data source?

Description: Nitey-Nite Dev

Which SQL Server do you want to connect to?

Server: 192.169.226.205\nitey_nite,1433

Figure 10.5

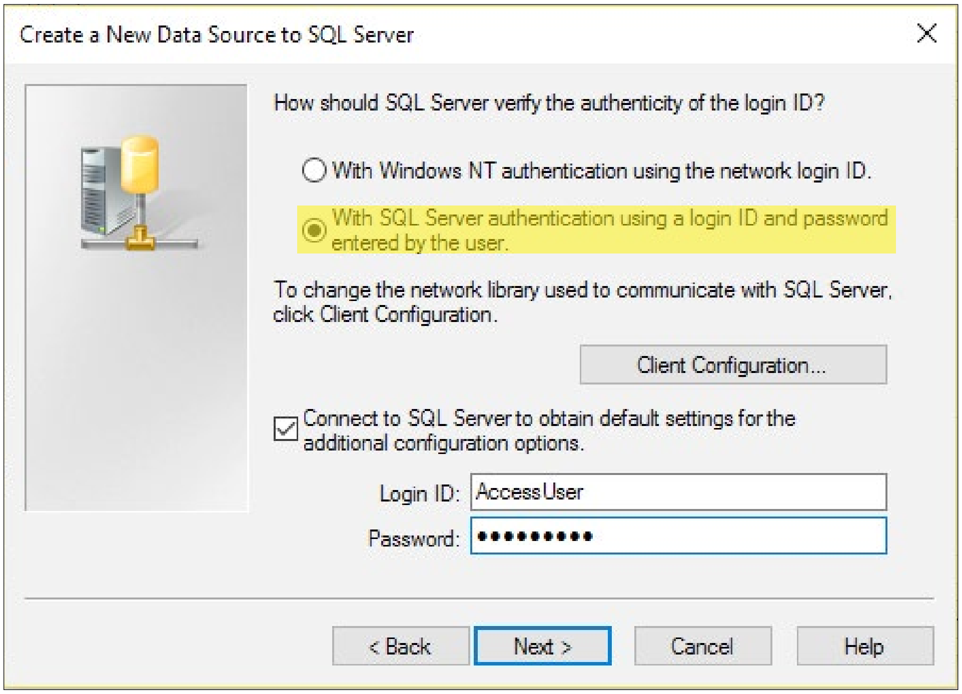
6. *Type in the name, description, and server for the DSN exactly as it appears in Figure 10.5.*

> ***Note**: Occasionally, it is necessary for us to change the server for this exercise. If you are unable to make a connection to the server, please log on to www.ExcelCEO.com/server_name.asp to check the current server name. The user name and password used to log on will remain the same as in this exercise.*

7. *Click* **Next >**.

In this step of the wizard, you will verify you have the authority to connect to the database. If you create a DSN on your company's server, you will most likely use a trusted connection (Windows NT authentication). Whenever you connect to a database using a trusted connection, it uses the login ID and password you used when you logged on to the computer. It will not force you to verify your connection to the DSN every time you log on. As with most hosted servers, you will be required to verify your user name and password each time you access the database, as is the case in this exercise. The user ID and password set up for this exercise is read-only access.

8. *Under the* **SQL Server** *authenticity question, click on the* **With SQL Server authentication…** *option.*
9. *For the* **Login ID**, *type* ***AccessUser***
10. *For the* **Password** *type* ***clinesys1***



Create a New Data Source to SQL Server

How should SQL Server verify the authenticity of the login ID?

○ With Windows NT authentication using the network login ID.

● With SQL Server authentication using a login ID and password entered by the user.

To change the network library used to communicate with SQL Server, click Client Configuration.

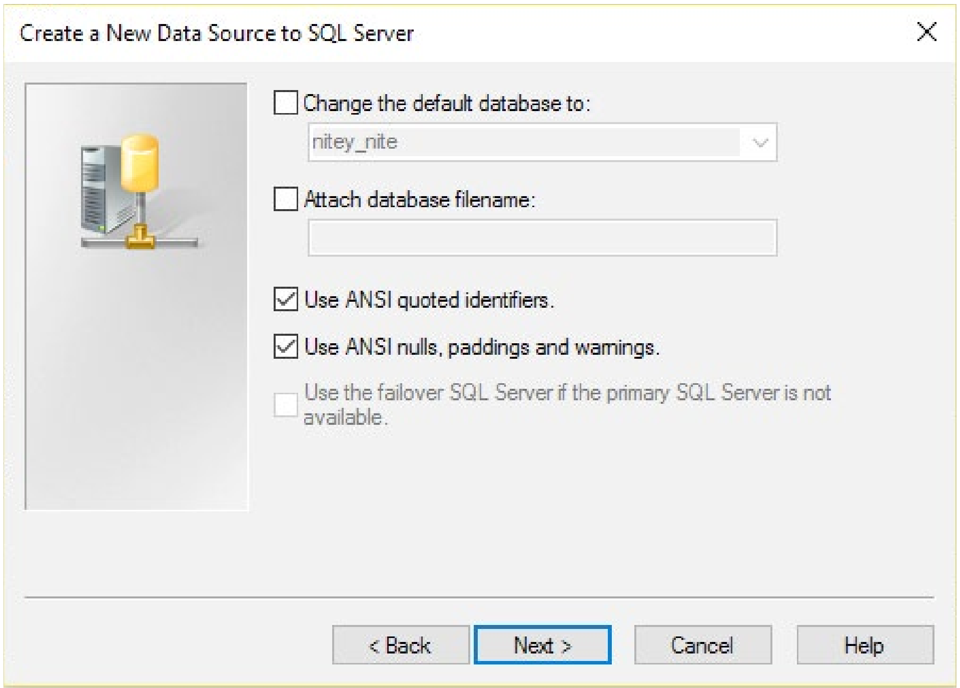
☑ Connect to SQL Server to obtain default settings for the additional configuration options.

Login ID: AccessUser

Password: ●●●●●●●●●

Figure 10.6

11. *Click* **Next >**.



Create a New Data Source to SQL Server

□ Change the default database to:

nitey_nite

□ Attach database filename:

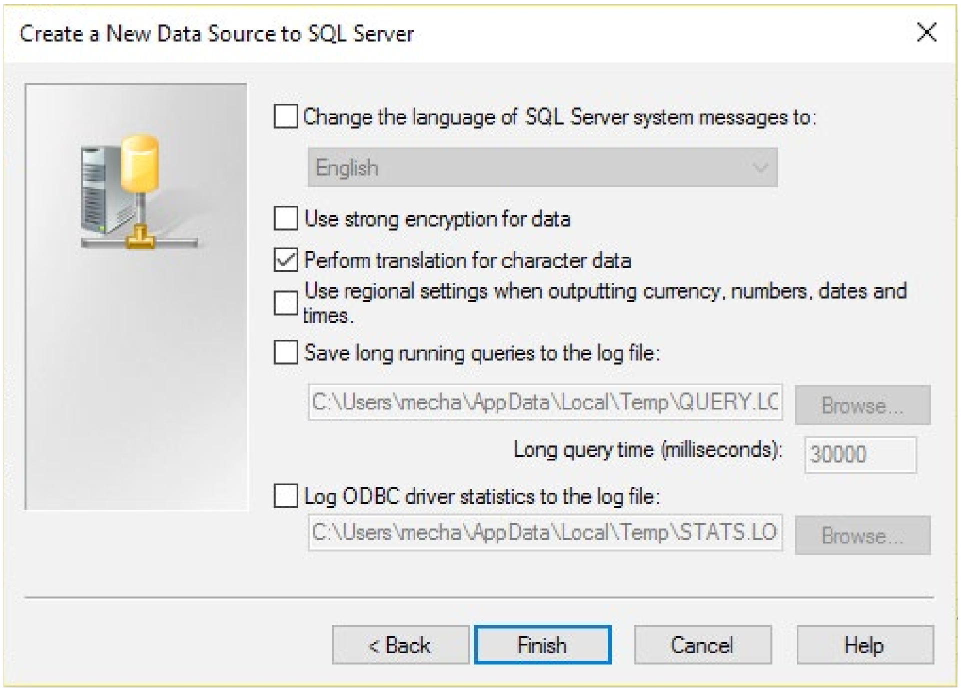
☑ Use ANSI quoted identifiers.

☑ Use ANSI nulls, paddings and warnings.

□ Use the failover SQL Server if the primary SQL Server is not available.

Figure 10.7

12. *Make sure the default database is set to* **nitey_nite** *and click* **Next >**.



Create a New Data Source to SQL Server

□ Change the language of SQL Server system messages to:

English

□ Use strong encryption for data

☑ Perform translation for character data

□ Use regional settings when outputting currency, numbers, dates and times.

□ Save long running queries to the log file:

C:\Users\mecha\AppData\Local\Temp\QUERY.LC

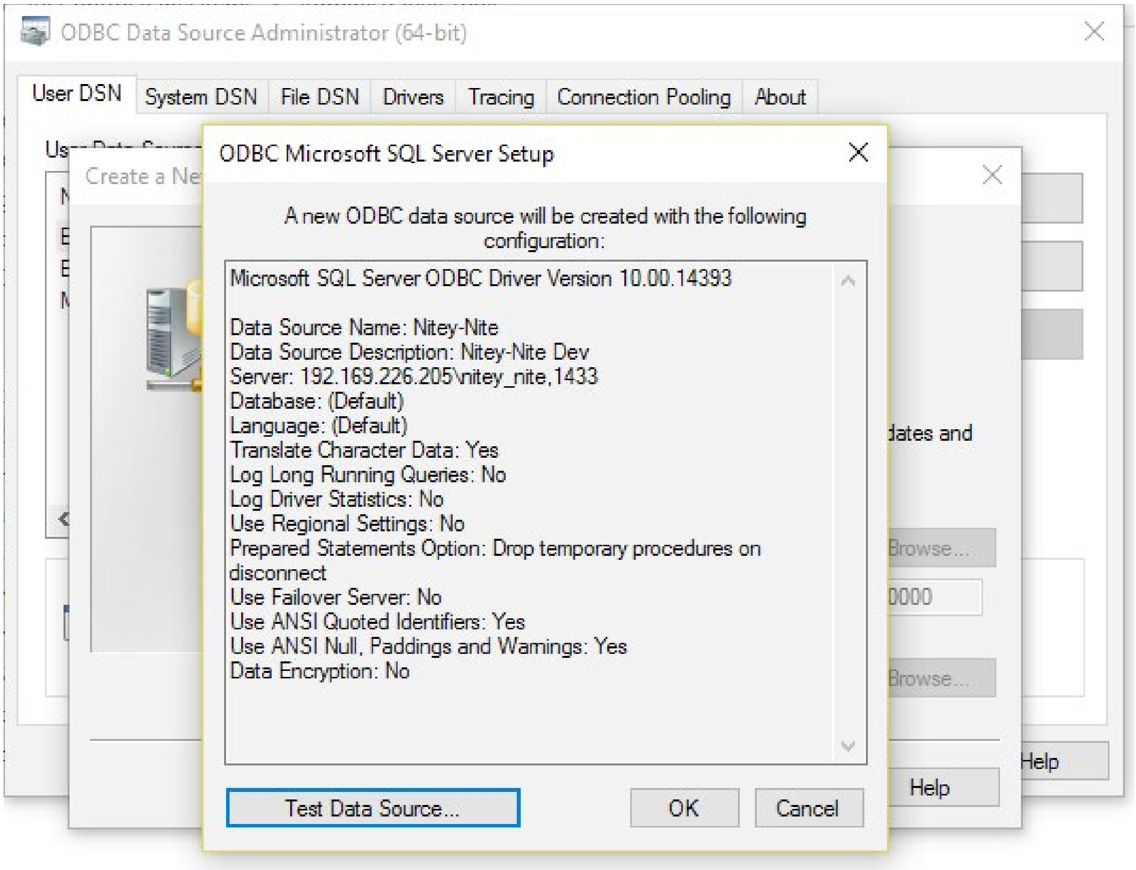
Long query time (milliseconds): 30000

□ Log ODBC driver statistics to the log file:

C:\Users\mecha\AppData\Local\Temp\STATS.LO

Figure 10.8

13. *Click* **Finish**.



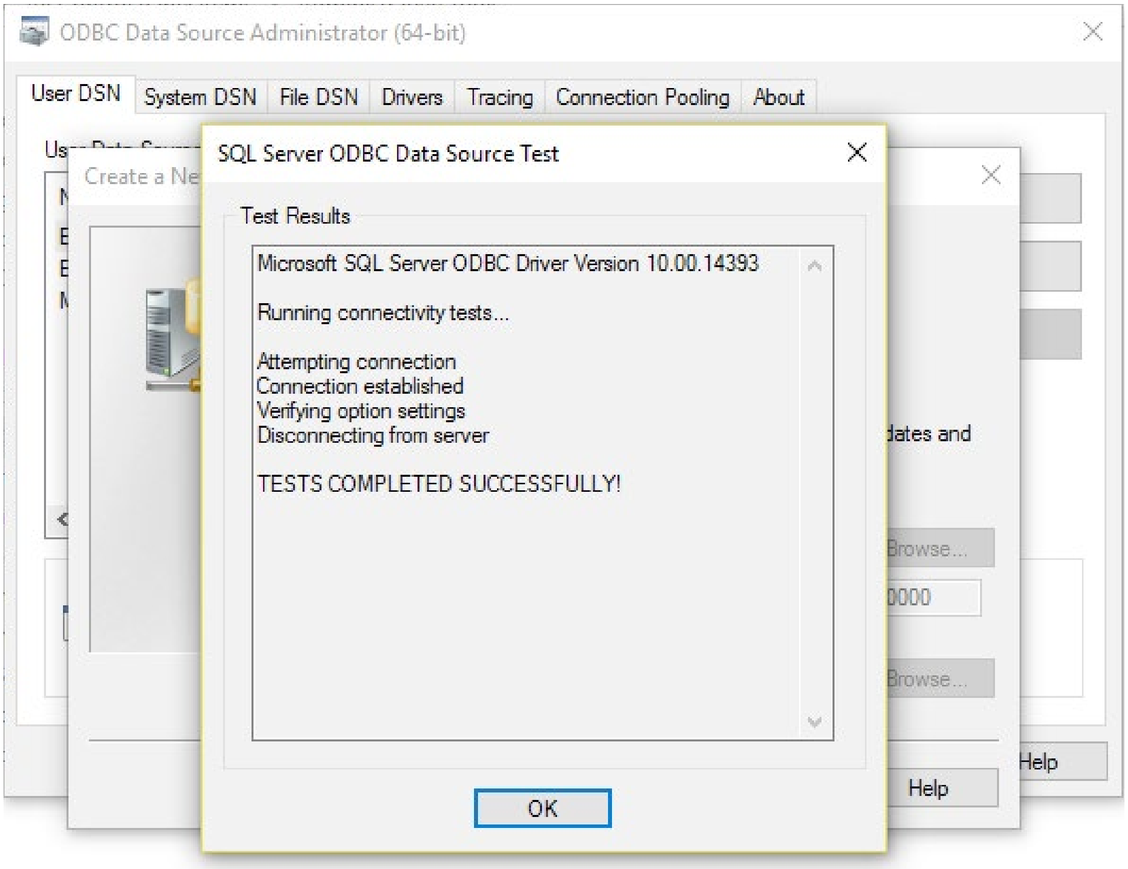
A new ODBC data source will be created with the following configuration:

```
Microsoft SQL Server ODBC Driver Version 10.00.14393

Data Source Name: Nitey-Nite
Data Source Description: Nitey-Nite Dev
Server: 192.169.226.205\nitey_nite,1433
Database: (Default)
Language: (Default)
Translate Character Data: Yes
Log Long Running Queries: No
Log Driver Statistics: No
Use Regional Settings: No
Prepared Statements Option: Drop temporary procedures on
disconnect
Use Failover Server: No
Use ANSI Quoted Identifiers: Yes
Use ANSI Null, Paddings and Warnings: Yes
Data Encryption: No
```

Figure 10.9

You should now get this dialog box where you can test the connection.

14. *Click the* **Test Data Source…** *button.*



```
Microsoft SQL Server ODBC Driver Version 10.00.14393

Running connectivity tests...

Attempting connection
Connection established
Verifying option settings
Disconnecting from server

TESTS COMPLETED SUCCESSFULLY!
```

Figure 10.10

You should get this message indicating that the connection was established and that it completed successfully.

15. *Click* **OK** *on the* **SQL Server ODBC Data Source Test** *dialog box.*
16. *Click* **OK** *in the next few boxes to exit out of the DSN creation wizard and close the* **Control Panel**.

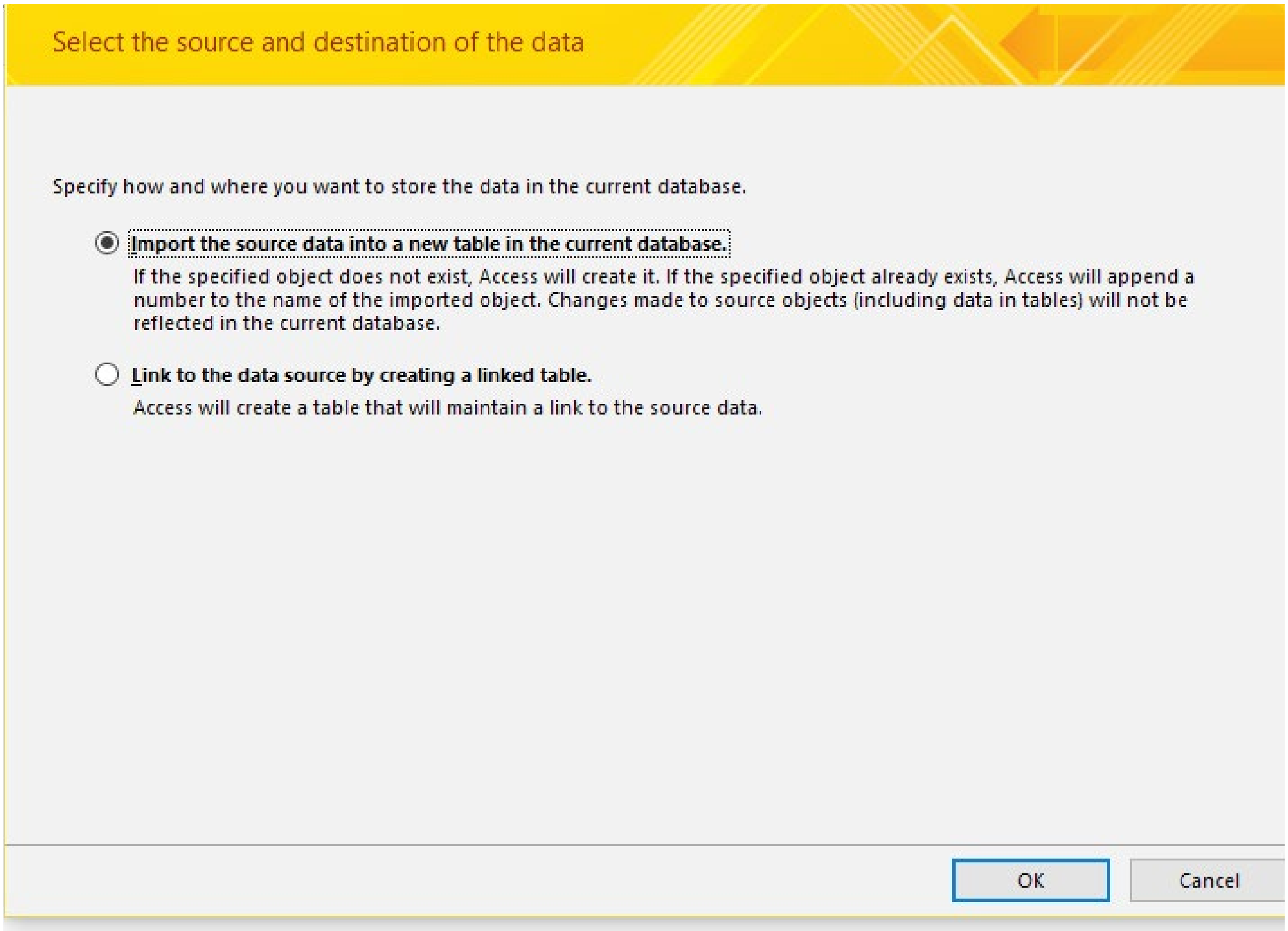
Now that you have the DSN created, you can connect to the Nitey-Nite SQL Server database. This server is physically located somewhere in California, but when you create the connection, it connects your computer to the server and allows you to manipulate the data and perform calculations as if they were on your own workstation.

Linking Tables Using a DSN

In the next few tasks, you will create a new Access database and connect to the three tables you need to create an income statement.

1. *Create a* **Blank database** *in* **Access** *and* **save** *it as* **C:\ExcelCEO\Access 2016*IncStmt.accdb***.

2. *Close out of the new table that automatically opens when you create a new* **Access 2016** *database.*
3. *Click on the* **External Data** *tab and click on the* **ODBC Database** *icon in the* **Import & Link** *group.*



Select the source and destination of the data

Specify how and where you want to store the data in the current database.

● Import the source data into a new table in the current database.

If the specified object does not exist, Access will create it. If the specified object already exists, Access will append a number to the name of the imported object. Changes made to source objects (including data in tables) will not be reflected in the current database.

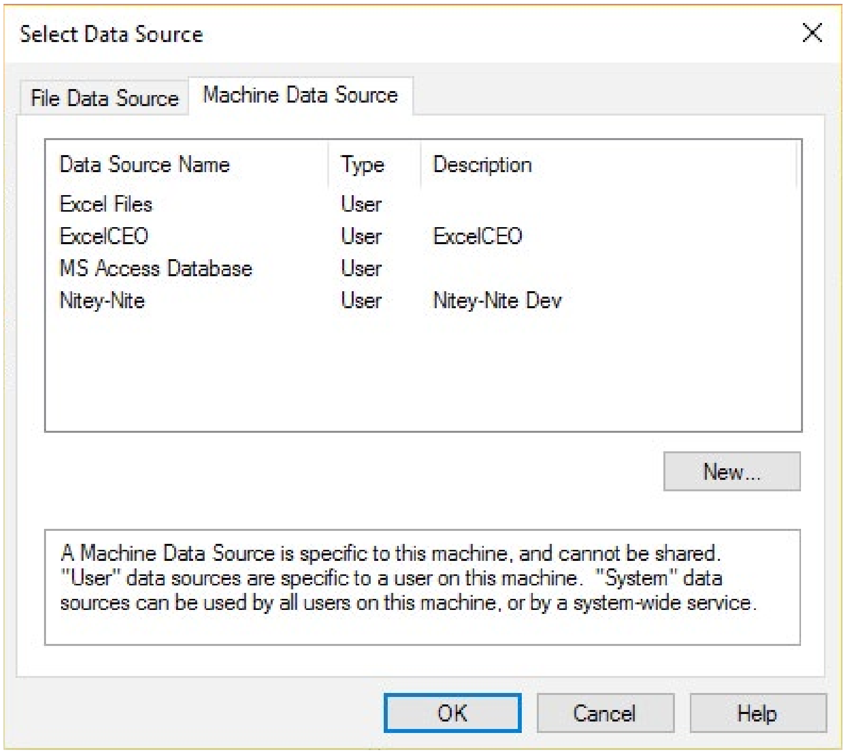
○ Link to the data source by creating a linked table.

Access will create a table that will maintain a link to the source data.

Figure 10.11

This is a list of all of the connections you can make in Access. The nitey-nite DSN is an ODBC database connection.

4. *In the* **Select the source and destination of the data** *dialog box, click on the* **Link to the data source by creating a linked table** *radio button and click* **OK***.*
5. *In the* **Select Data Source** *dialog box, click on the* **Machine Data Source** *tab.*



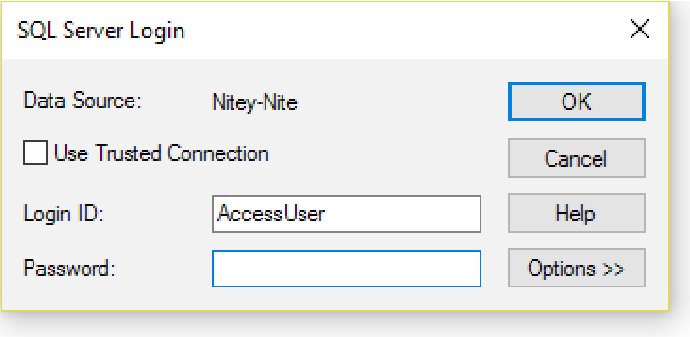
Data Source Name	Type	Description
Excel Files	User	
ExcelCEO	User	ExcelCEO
MS Access Database	User	
Nitey-Nite	User	Nitey-Nite Dev

A Machine Data Source is specific to this machine, and cannot be shared. "User" data sources are specific to a user on this machine. "System" data sources can be used by all users on this machine, or by a system-wide service.

Figure 10.12

You may have other DSNs available for use on your machine. You need to scroll to the Nitey_nite DSN and select it.

6. *Click on the* **Nitey-Nite** *DSN and click* **OK***.*



Data Source: Nitey-Nite

□ Use Trusted Connection

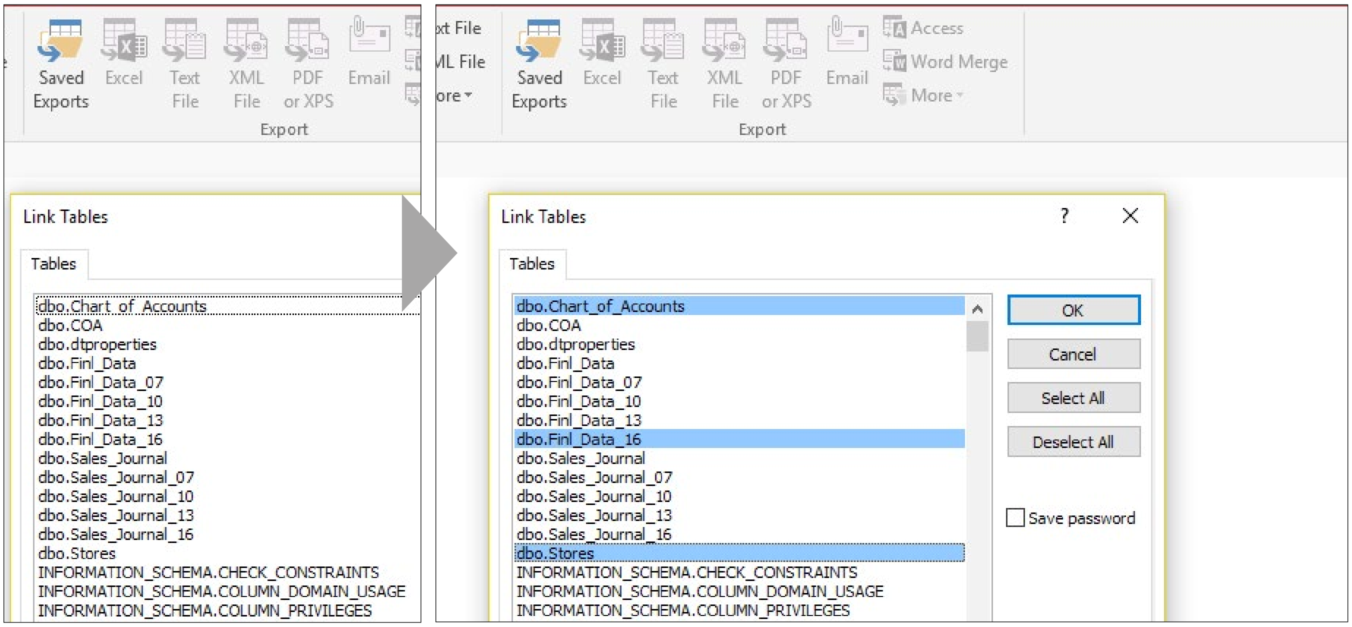
Login ID: AccessUser

Password:

Figure 10.13

As you have a SQL Server authentication connection, Access asks you for the user ID and password.

7. *Enter* ***AccessUser*** *as the* **Login ID** *and* ***clinesys1*** *as the* **Password** *and click* **OK***.*

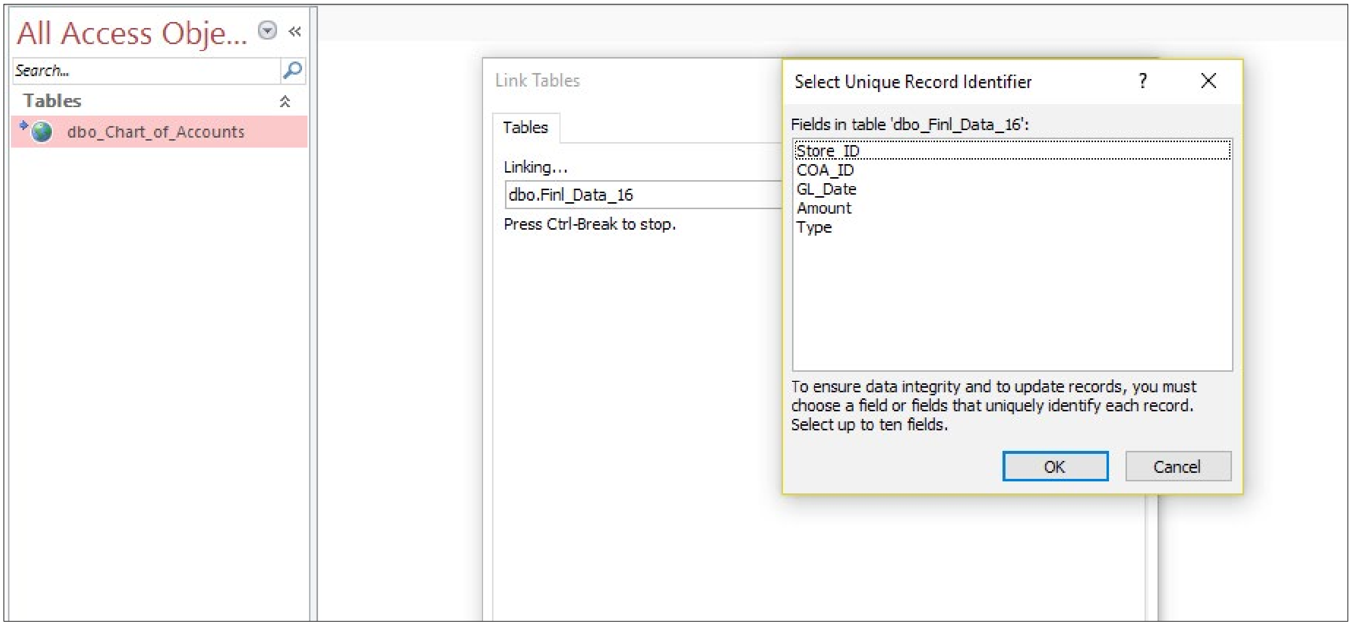


dbo.Chart_of_Accounts
dbo.COA
dbo.dtproperties
dbo.Finl_Data
dbo.Finl_Data_07
dbo.Finl_Data_10
dbo.Finl_Data_13
dbo.Finl_Data_16
dbo.Sales_Journal
dbo.Sales_Journal_07
dbo.Sales_Journal_10
dbo.Sales_Journal_13
dbo.Sales_Journal_16
dbo.Stores
INFORMATION_SCHEMA.CHECK_CONSTRAINTS
INFORMATION_SCHEMA.COLUMN_DOMAIN_USAGE
INFORMATION_SCHEMA.COLUMN_PRIVILEGES

Figure 10.14

The three tables to which you want to link are Chart_of_Accounts, Finl_Data_13, and Stores. Make sure you don't connect to the other tables as those tables are used for other Access courses.

8. *Click on* **dbo.Chart_Of_Accounts**, **dbo.Finl_Data_16**, *and* **dbo.Stores** *and click* **OK**.



Linking...
dbo.Finl_Data_16
Press Ctrl-Break to stop.

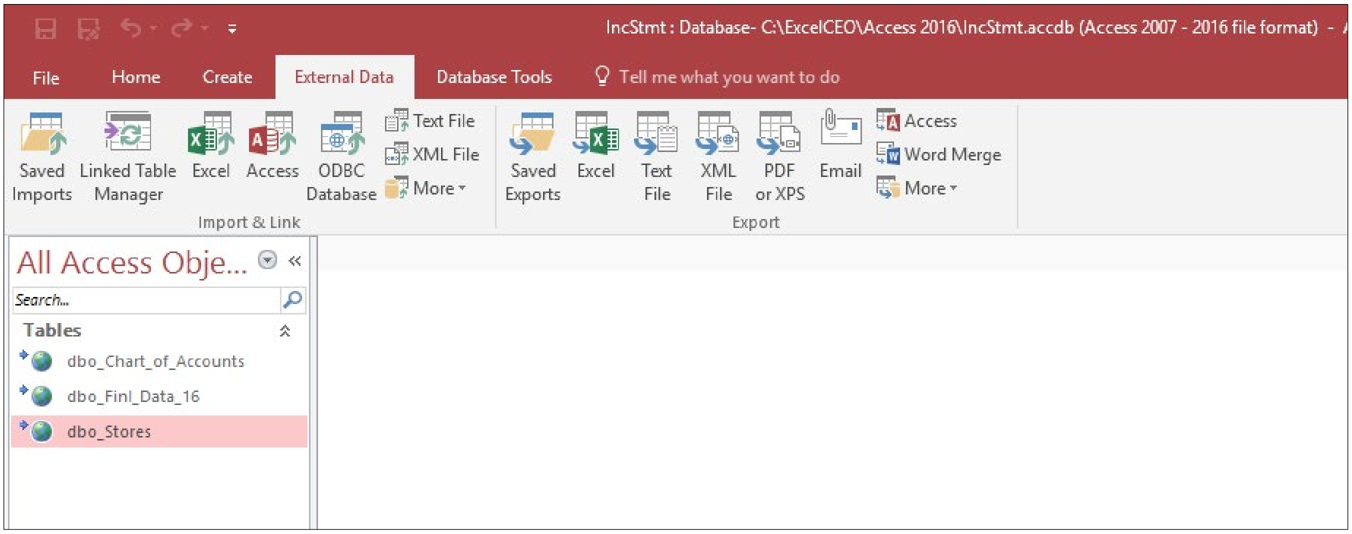
Fields in table 'dbo_Finl_Data_16':
Store_ID
COA_ID
GL_Date
Amount
Type

To ensure data integrity and to update records, you must choose a field or fields that uniquely identify each record. Select up to ten fields.

Figure 10.15

Since the Finl_Data_16 table doesn't have a Primary Key set, Access asks you which field should be set as the Primary Key, or unique identifier. The database designer should have already set up at least one field as the Primary Key. Typically, a Primary Key is a single field that uniquely identifies each record in the table, but in this table, there is no such field. We can create a combination of fields that will represent a unique record. In this case, we'll use all of the fields except for the Type field.

9. *Click on* **Store_ID**, **COA_ID**, **GL_Date**, *and* **Amount** *fields and click* **OK**.



IncStmt : Database- C:\ExcelCEO\Access 2016\IncStmt.accdb (Access 2007 - 2016 file format)

Figure 10.16

> ***Note**: If your **Navigation Pane** opens up too narrow to see the linked tables, and will not adjust at first, close **Access** and reopen and the pane should be adjustable.*

In the Table pane, you should now see links to three tables. The right arrow on the far left of the graphic indicates that the table is linked. The world graphic indicates that it is linked to a SQL Server table, ODBC connection, or view. Whenever you connect to a SQL Server table, Access places the "owner" of the table in front of the table name, separated by an underscore. The owner for these tables is dbo (database owner), meaning that anyone who has the appropriate permissions to this database can read these files. You can just assume that these tables are generated and updated by the system, and all you can do is link to them and read them, but you can't change them. Even if you tried to change the data in the tables, you couldn't, as the login ID and password you used to access them has read-only permissions.

Importing Objects

With the tables linked into the navigation pane, you can treat them like any other table or data source. Do you remember in Chapter 3 when you took the hierarchical COA table and turned it into a flat file using a query? We're going to use that same query in this exercise. Instead of re-creating the query, all we have to do is import it. There's just one little trick to it. The table we used in Chapter 3 was called Chart_of_Accounts, but the link we have to the SQL Server table is called dbo_Chart_of_Accounts. You can rename a link just like you can a table, query, or any other object. Let's rename the link first, and then we'll import the query.

1. *Right-click on the* **dbo_Chart_of_Accounts** *link and choose* **Rename** *(this puts the object into* **Edit** *mode).*
2. *Rename the link as* ***Chart_of_Accounts*** *and press* **[Enter]**.
3. *In the* **External Data** *tab, click on the* **Access** *icon in the* **Import & Link** *group.*
4. *In the* **Get External Data – Access Database** *dialog box, make sure the* **Import tables, queries, forms, reports, macros, and modules into the current database.** *radio button is selected.*
5. *Click on the* **Browse…** *button and navigate to* **C:\ExcelCEO\Access 2016\Nitey_Nite_2016.accdb**, *click* **Open**, *then click on* **OK**.

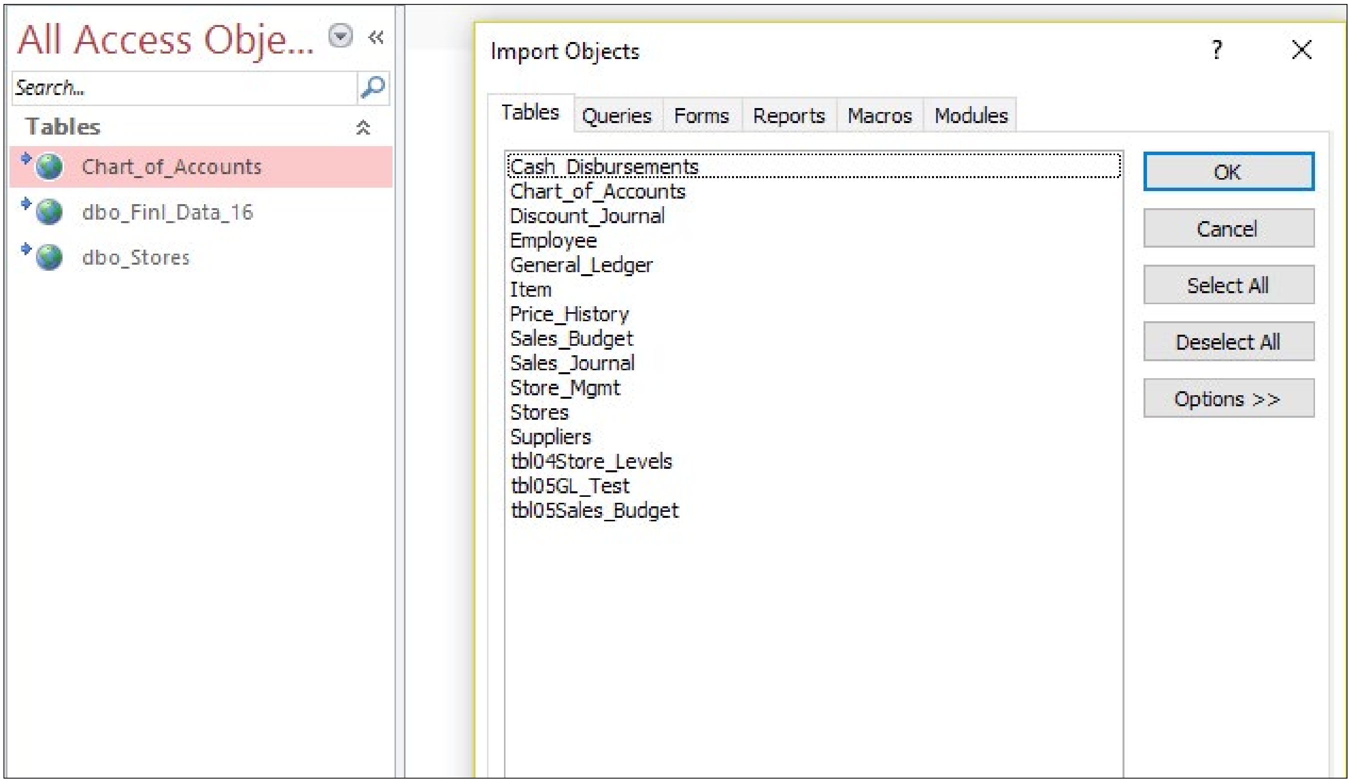


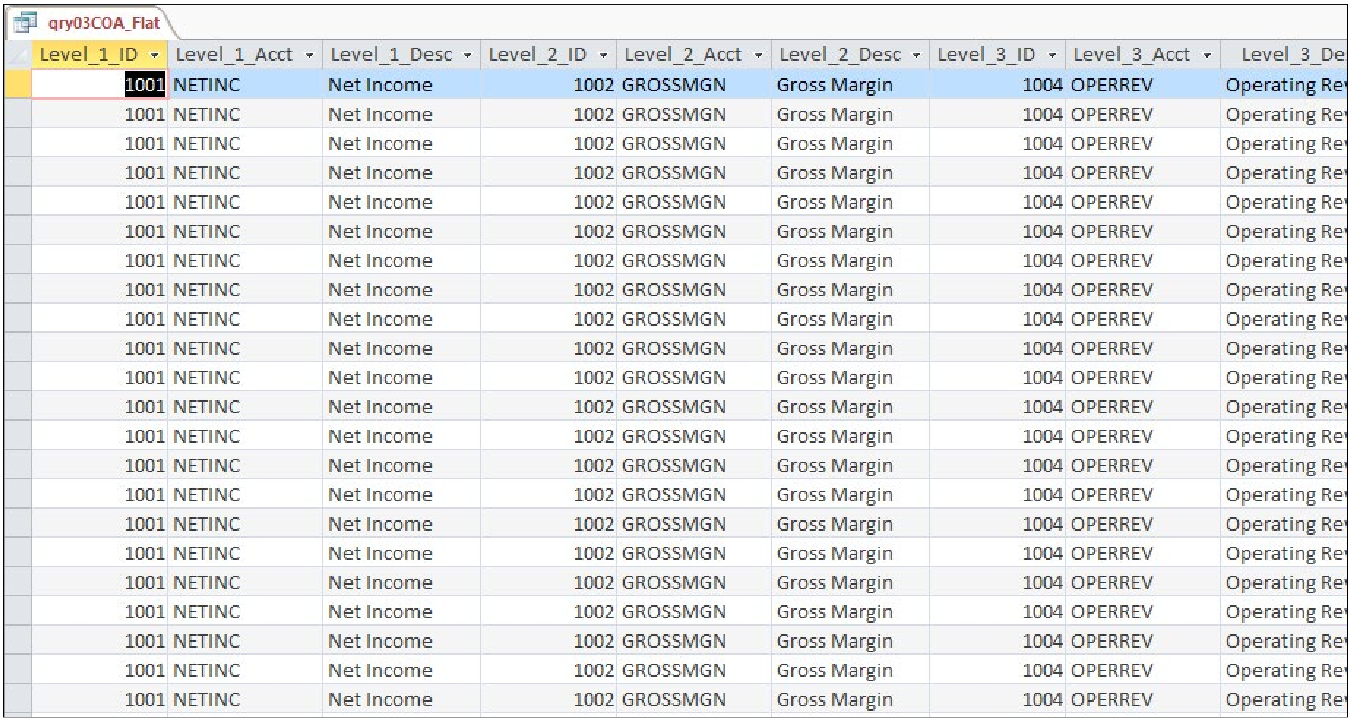
Figure 10.17

The Import Objects dialog box appears. The tabs along the top represent each pane in the Access database. You can import any object from any pane, as long as the object is not linked. If it is linked, you have to go to the base file and import it from there. We want to import the query called qry03COA_Flat, so just click on the Queries tab and import it.

6. *Click on the* **Queries** *tab, click on* **qry03COA_Flat**, *and click* **OK**.
7. *Click* **Close** *in the* **Get External Data – Access Database** *dialog box.*

Access imports the query and places it under the Queries section of the Navigation Pane. To test the query to make sure it's working, just open the query. You may need to login to open the connection again.

8. *Open* **qry03COA_Flat** *in* **Datasheet View**.



qry03COA_Flat

Level_1_ID	Level_1_Acct	Level_1_Desc	Level_2_ID	Level_2_Acct	Level_2_Desc	Level_3_ID	Level_3_Acct	Level_3_De
1001	NETINC	Net Income	1002	GROSSMGN	Gross Margin	1004	OPERREV	Operating Re
1001	NETINC	Net Income	1002	GROSSMGN	Gross Margin	1004	OPERREV	Operating Re
1001	NETINC	Net Income	1002	GROSSMGN	Gross Margin	1004	OPERREV	Operating Re
1001	NETINC	Net Income	1002	GROSSMGN	Gross Margin	1004	OPERREV	Operating Re
1001	NETINC	Net Income	1002	GROSSMGN	Gross Margin	1004	OPERREV	Operating Re
1001	NETINC	Net Income	1002	GROSSMGN	Gross Margin	1004	OPERREV	Operating Re
1001	NETINC	Net Income	1002	GROSSMGN	Gross Margin	1004	OPERREV	Operating Re
1001	NETINC	Net Income	1002	GROSSMGN	Gross Margin	1004	OPERREV	Operating Re
1001	NETINC	Net Income	1002	GROSSMGN	Gross Margin	1004	OPERREV	Operating Re
1001	NETINC	Net Income	1002	GROSSMGN	Gross Margin	1004	OPERREV	Operating Re
1001	NETINC	Net Income	1002	GROSSMGN	Gross Margin	1004	OPERREV	Operating Re
1001	NETINC	Net Income	1002	GROSSMGN	Gross Margin	1004	OPERREV	Operating Re
1001	NETINC	Net Income	1002	GROSSMGN	Gross Margin	1004	OPERREV	Operating Re
1001	NETINC	Net Income	1002	GROSSMGN	Gross Margin	1004	OPERREV	Operating Re
1001	NETINC	Net Income	1002	GROSSMGN	Gross Margin	1004	OPERREV	Operating Re
1001	NETINC	Net Income	1002	GROSSMGN	Gross Margin	1004	OPERREV	Operating Re
1001	NETINC	Net Income	1002	GROSSMGN	Gross Margin	1004	OPERREV	Operating Re
1001	NETINC	Net Income	1002	GROSSMGN	Gross Margin	1004	OPERREV	Operating Re
1001	NETINC	Net Income	1002	GROSSMGN	Gross Margin	1004	OPERREV	Operating Re
1001	NETINC	Net Income	1002	GROSSMGN	Gross Margin	1004	OPERREV	Operating Re
1001	NETINC	Net Income	1002	GROSSMGN	Gross Margin	1004	OPERREV	Operating Re
1001	NETINC	Net Income	1002	GROSSMGN	Gross Margin	1004	OPERREV	Operating Re

Figure 10.18

It looks just like it did in Chapter 3.

9. *Close* **qry03COA_Flat** *and close the database.*

> ***Review Questions****: It is now time to complete the hands-on Review Questions. Log on to www.ExcelCEO.com with your Email and Password, click on the* **Access 2016 and SQL Review Questions, Chapter 10, Section 1 of 1** *option and complete the review questions.*

Conclusion

In this chapter you learned about Data Source Names (DSN). You set up a DSN to a database that is on a hosted server using SQL Server authentication. After the DSN was created, you used it to link to tables in that database. You learned how to import objects from one database to another. You also saw the different types of data sources that can be imported into an Access database.

Chapter Exam

You can now go to www.ExcelCEO.com, click on Sign In, log in and take the exam. Make sure that you take the exam on the same device on which you completed the practice files, as some of the questions on the exam may refer to some of the completed projects. Chapter exams are intended to be hands-on.

CHAPTER ELEVEN — ADVANCED REPORTING (PART I)

In this chapter, you will:

- Determine how to build a query on which you will base a financial statement report
- Identify the basic financial statement concepts of Gross Margin, Fixed and Variable Expenses
- Select a Chart of Accounts (COA) table to perform creative sorting
- Choose a report style within the Report Wizard and modify it
- Choose a form to allow users to change the variables to run a report

CPE Credits possible for this chapter: 3

Advanced Reporting (Part I)

There are so many things I want to cover in Advanced Reporting that I had to break them up into two chapters. Up to this point, you've learned the basics and some intermediate Access reporting skills. You can create a somewhat complex report, and you should have mastered queries and forms. Chapters 11 and 12 are the capstone of everything we work towards in Access. A word of caution – these next two chapters, and particularly Chapter 12, are hard. If you can get through these next two chapters, you will have made it. Bear with me as we go through these chapters together. It will get tedious at times, but in the end, you will have unmatched reporting skills.

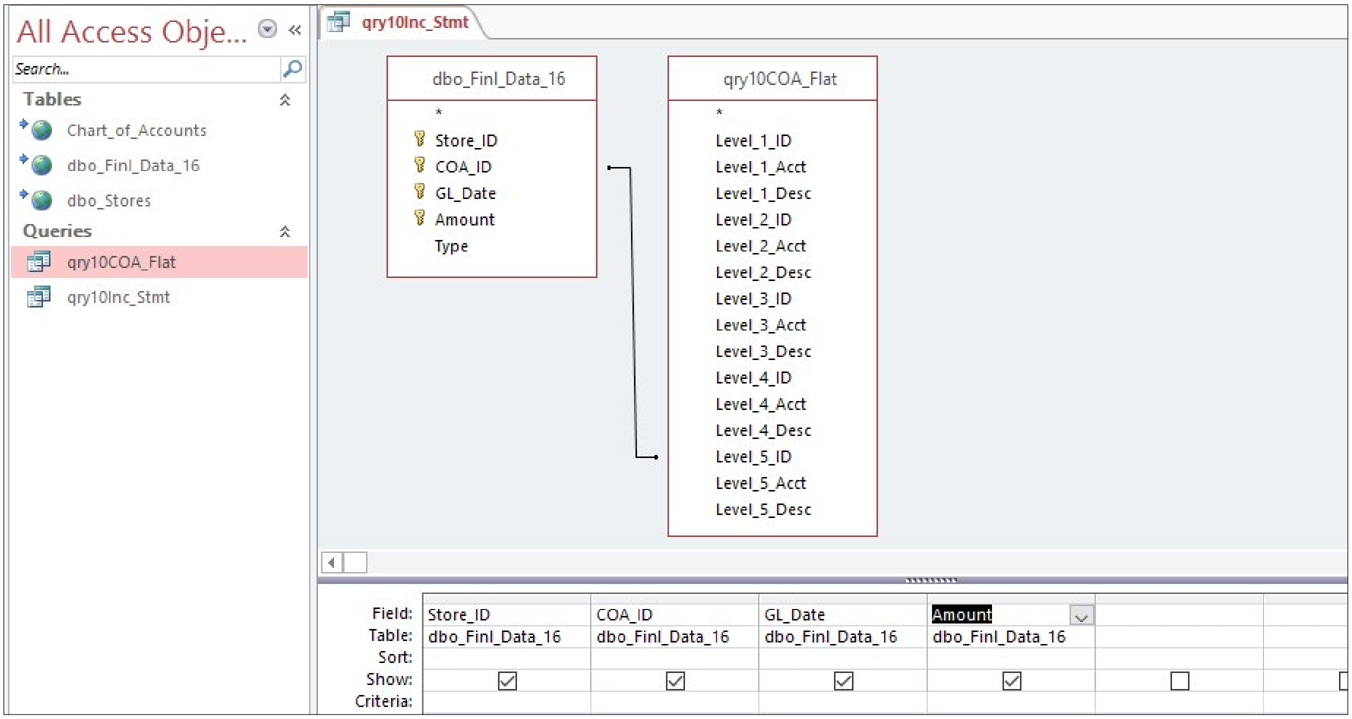
In this chapter, you will create the beginnings of a reporting environment, not just a report. You will begin by creating the necessary queries to populate your report (which will populate an income statement), create the report, then create a form to run the report. When you create the form, you will add in some advanced features to allow the user to choose which report he/she wants to run. In the next chapter, you will add the functionality for the user to change the criteria for the report (store number, city, state, region, year, month) and have it automatically populate the report without building separate reports.

For the basis of this report, you will use the database you created in Chapter 10, Inc_Stmt.accdb. Now that you have the flat file (qry03COA_Flat), the Stores table and the financial data in the Inc_Stmt database, you can begin to build a query that combines these three tables, which will serve as the data source behind the income statement report. Remember that since this query is linked directly to the data in SQL Server, your report will automatically update anytime the data updates.

> ***Note**: If you were unable to connect to the* **SQL Server** *database, don't worry. You can use the* **Static_2016.accdb** *database to complete the rest of the exercises in this course.*

In this exercise, you will tie the dbo_Finl_Data_16 table to a query. Remember that a query is simply a virtual table, and you can create joins on a query just like you can on a table.

1. *Open the* **IncStmt** *database.*
2. *Rename* **qry03COA_Flat** *as* ***qry10COA_Flat***.
3. **Create** *a new query and pull in* **dbo_Finl_Data_16** *and* **qry10COA_Flat**.
4. *Link the* **COA_ID** *field in* **dbo_Finl_Data_16** *to the* **Lvl5_ID** *field in* **qry10COA_Flat**.
5. *Bring the* **Store_ID**, **COA_ID**, **GL_Date**, *and* **Amount** *fields into the* **Design View**.
6. **Save** *the query as* ***qry10Inc_Stmt***.



Field:	Store_ID	COA_ID	GL_Date	Amount		
Table:	dbo_Finl_Data_16	dbo_Finl_Data_16	dbo_Finl_Data_16	dbo_Finl_Data_16		
Sort:						
Show:	☑	☑	☑	☑	☐	☐
Criteria:						

Figure 11.1

If you closed the IncStmt database after you completed Chapter Ten, and if you are using the external database connections, you will be prompted to input the user name and password provided in Chapter 10 when you run the query. This re-establishes the connection to the hosted server. It may take a minute or two (depending on your Internet connection speed), but you should soon see the results. If you run the query and go to the last record, you should see the record set has 293,150 records. Generally speaking, queries take less time to run when there is less data returned in the record set. Therefore, we will pick one store in the query on which to build and test the report. Once the report is built, I will show you how to change the store number criteria (as well as the other criteria) in a form. Let's continue building the query.

At this point, we need to bring in the Stores table.

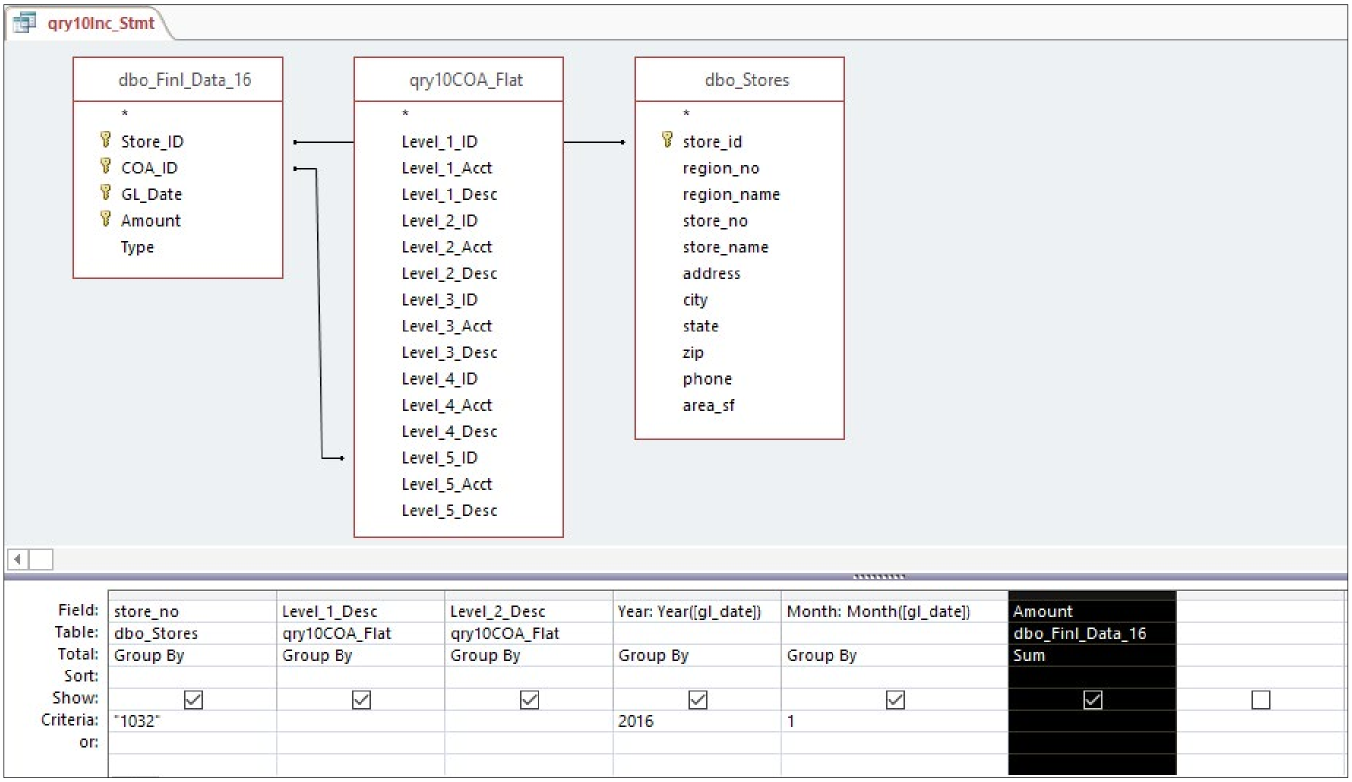
7. *In the* **Design View** *of* **qry10Inc_Stmt**, *bring in the* **dbo_Stores** *table.*
8. *Create a join on* **Store_ID** *between the* **dbo_Stores** *table and the* **dbo_Finl_Data_16** *table.*
9. *Bring down the following fields in the* **Design Grid***:*

   store_no
   Level_1_Desc
   Level_2_Desc
   Year *as* ***year(gl_date)***
   Month *as* ***month(gl_date)***

10. *Group all fields (by clicking on the* **Totals** *icon) and change the* **Group By** *on the* **Amount**

field to **Sum***.*

11. *Filter* **Year** *for* ***2016****,* **Month** *for* ***1****, and* **Store_No** ***1032****.*
12. *Take out the* **Store_ID***,* **COA_ID***, and* **GL_Date** *fields, and move the* **Amount** *field to be the last field in the query.*

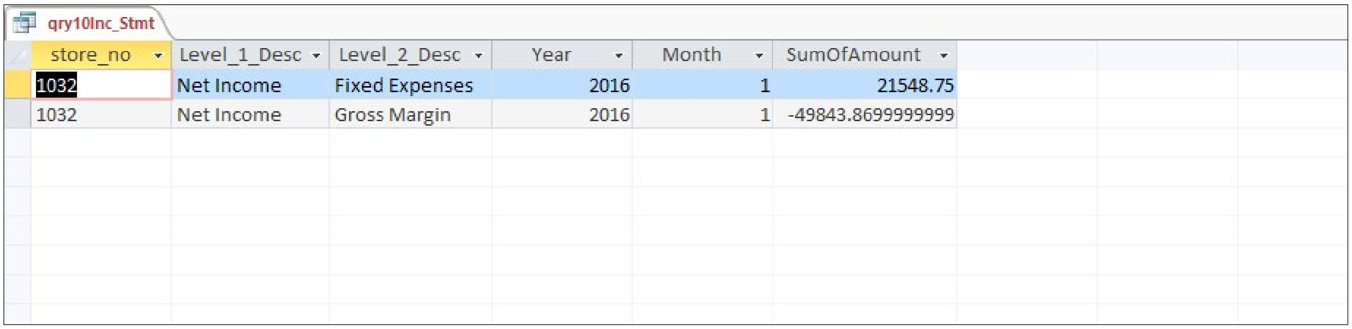


Field:	store_no	Level_1_Desc	Level_2_Desc	Year: Year([gl_date])	Month: Month([gl_date])	Amount	
Table:	dbo_Stores	qry10COA_Flat	qry10COA_Flat			dbo_Finl_Data_16	
Total:	Group By	Group By	Group By	Group By	Group By	Sum	
Sort:							
Show:	☑	☑	☑	☑	☑	☑	□
Criteria:	"1032"			2016	1		
or:							

Figure 11.2

13. **Run** *the query.*

Note*: You may have to adjust the column width of the* **SumOfAmount** *column to see the entire value.*



store_no	Level_1_Desc	Level_2_Desc	Year	Month	SumOfAmount
1032	Net Income	Fixed Expenses	2016	1	21548.75
1032	Net Income	Gross Margin	2016	1	-49843.8699999999

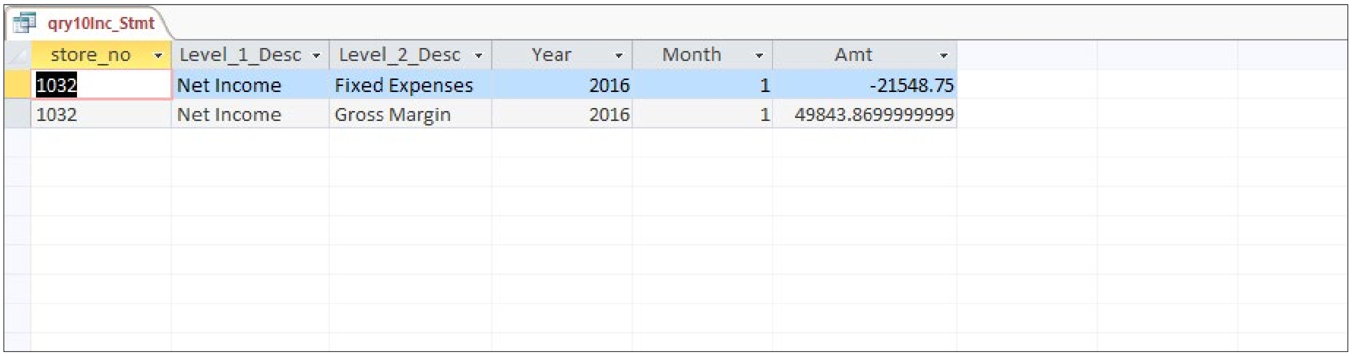
Figure 11.3

This is the basis for our financial statement. The query contains the store structure rollup (via the dbo_Stores table), the account rollup (via the Chart_of_Accounts table) and the financial data (via the dbo_Finl_Data_16 table). It is currently filtered for Store 1032, January 2016. The lowest account level currently displayed in the query is for Level 2, which splits out all amounts between Fixed Expenses and Gross Margin. If you remember your Accounting 101 course, Gross Margin (sometimes called Contribution Margin) is Total Revenue less Variable Expenses. The Level 2 categories roll up to Level 1, which is Net Income.

Sorting on Multiple Levels

Let's discuss the Amount field. The data in the Amount field comes directly from the dbo_Finl_Data_16 table, which carries GAAP signs (i.e., revenues are normally credits, or a minus sign, and expenses are debits, generally a positive number). Management at Nitey-Nite likes to see revenues as positive numbers and expenses as negative numbers, so all we have to do is to change or flip the signs on all amounts. Doing that in the query is the easiest way, and the change keeps the original data intact.

14. *In* **Design View**, *change the* **Amount** *field to have an alias called* ***Amt*** *(remember you can't name an alias with the same name as an existing field) and the formula as "****–Amount****".*
15. **Run** *the query.*



qry10Inc_Stmt

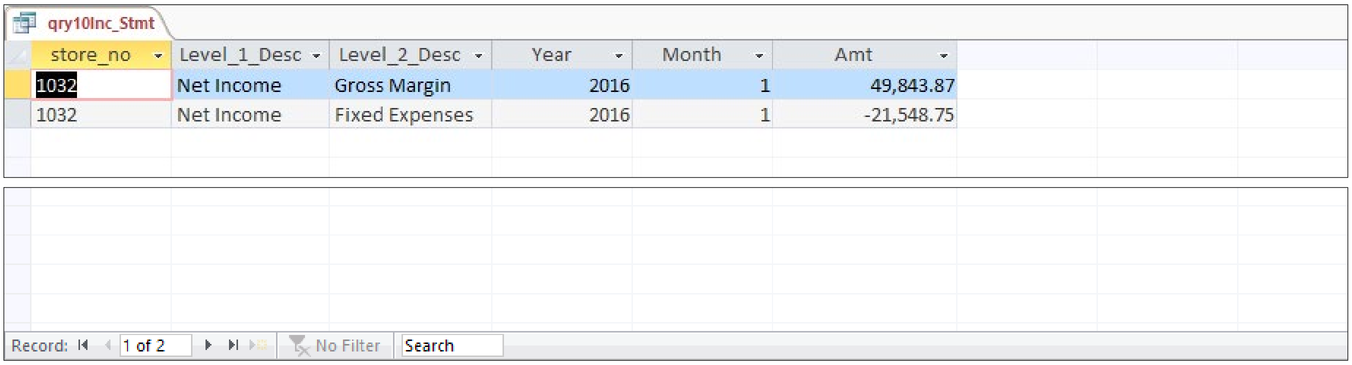
store_no	Level_1_Desc	Level_2_Desc	Year	Month	Amt
1032	Net Income	Fixed Expenses	2016	1	-21548.75
1032	Net Income	Gross Margin	2016	1	49843.8699999999

Figure 11.4

The signs are now reversed where the fixed expense amounts are negative and the gross margin amount is positive. Note that the amounts for the variable expenses that are rolling up to gross margin are also negative. The revenues are positive, and are more than the variable expenses, leaving a net positive amount for gross margin. Now that the amount signs are correct for reporting purposes, we'll continue.

The next issue is ordering the accounts and subtotals. Notice how Fixed Expenses appear before Gross Margin. In the Operating Statement, we want Gross Margin to appear first, followed by Fixed Expenses. To make Gross Margin appear first, just change the sort on Level_2_Desc to Descending. There is no need to put a sort on Level_1_Desc as there is only one description, Net Income.

16. *In* **Design View**, *make the* **Level_2_Desc** *field sort in* **Descending** *order by choosing* **Descending** *on the* **Sort** *line.*
17. **Run** *the query.*

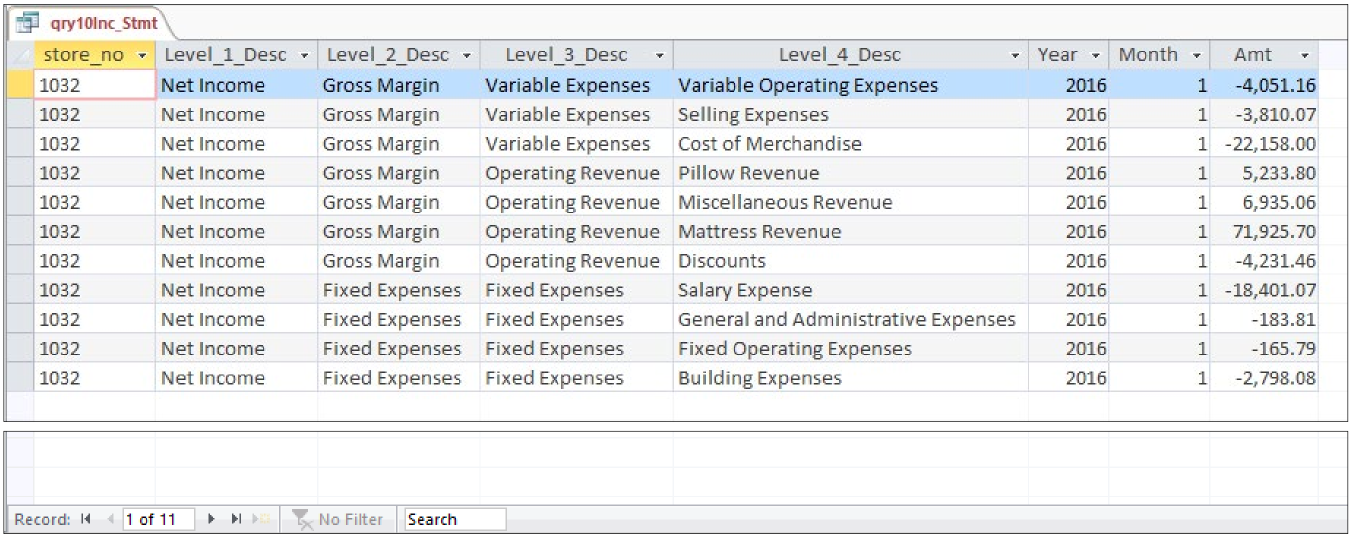


store_no	Level_1_Desc	Level_2_Desc	Year	Month	Amt
1032	Net Income	Gross Margin	2016	1	49,843.87
1032	Net Income	Fixed Expenses	2016	1	-21,548.75

Figure 11.5

Now let's add in the third and fourth layer of accounts. These levels will make up all accounts that will appear in the summary operating statement.

18. *In* **Design View***, place the* **Level_3_Desc** *and* **Level_4_Desc** *fields to the right of* **Level_2_Desc***.*
19. **Run** *the query and adjust the column widths as necessary.*

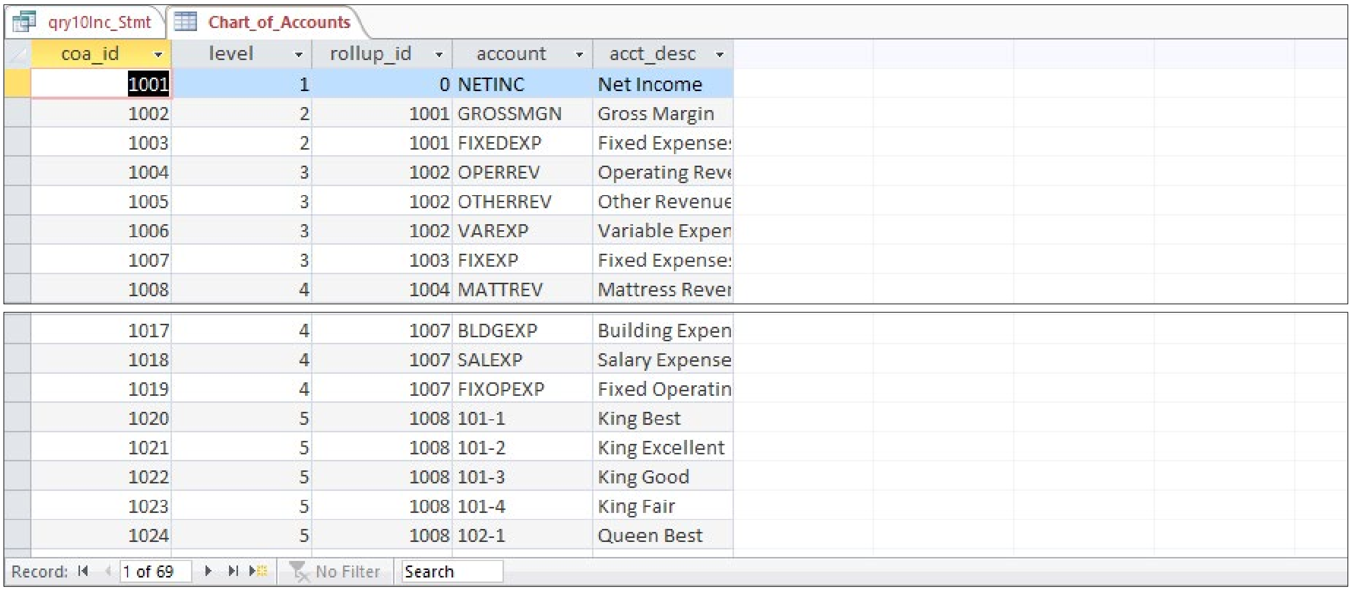


store_no	Level_1_Desc	Level_2_Desc	Level_3_Desc	Level_4_Desc	Year	Month	Amt
1032	Net Income	Gross Margin	Variable Expenses	Variable Operating Expenses	2016	1	-4,051.16
1032	Net Income	Gross Margin	Variable Expenses	Selling Expenses	2016	1	-3,810.07
1032	Net Income	Gross Margin	Variable Expenses	Cost of Merchandise	2016	1	-22,158.00
1032	Net Income	Gross Margin	Operating Revenue	Pillow Revenue	2016	1	5,233.80
1032	Net Income	Gross Margin	Operating Revenue	Miscellaneous Revenue	2016	1	6,935.06
1032	Net Income	Gross Margin	Operating Revenue	Mattress Revenue	2016	1	71,925.70
1032	Net Income	Gross Margin	Operating Revenue	Discounts	2016	1	-4,231.46
1032	Net Income	Fixed Expenses	Fixed Expenses	Salary Expense	2016	1	-18,401.07
1032	Net Income	Fixed Expenses	Fixed Expenses	General and Administrative Expenses	2016	1	-183.81
1032	Net Income	Fixed Expenses	Fixed Expenses	Fixed Operating Expenses	2016	1	-165.79
1032	Net Income	Fixed Expenses	Fixed Expenses	Building Expenses	2016	1	-2,798.08

Figure 11.6

Uh-oh. We may have a problem. On the third category level, management likes to see Operating Revenue first, followed by Variable Expenses and then by Fixed Expenses. That is OK in the query, but Level 4 presents another issue. Within Operating Revenue, management likes to see Mattress Revenue as the first category, followed by Pillow Revenue, Other Revenue, and then by Discounts. There is no alphabetic ordering using the Code or the Description, so we'll have to think of another way to do it. Let's open up the Chart_of_Accounts table to see if we can find a way around that.

20. *Open the* **Chart_of_Accounts** *table.*



qry10Inc_Stmt | Chart_of_Accounts

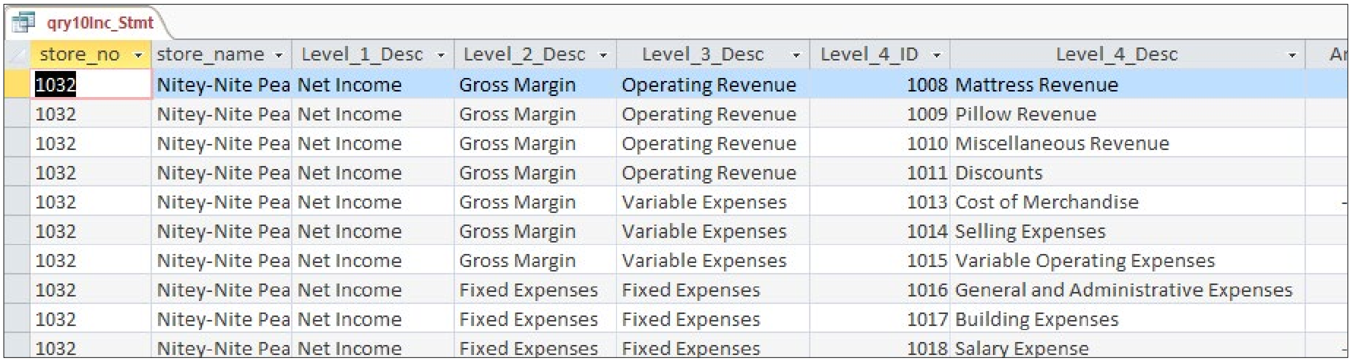
coa_id	level	rollup_id	account	acct_desc
1001	1	0	NETINC	Net Income
1002	2	1001	GROSSMGN	Gross Margin
1003	2	1001	FIXEDEXP	Fixed Expense:
1004	3	1002	OPERREV	Operating Reve
1005	3	1002	OTHERREV	Other Revenue
1006	3	1002	VAREXP	Variable Exper
1007	3	1003	FIXEXP	Fixed Expense:
1008	4	1004	MATTREV	Mattress Rever
1017	4	1007	BLDGEXP	Building Expen
1018	4	1007	SALEXP	Salary Expense
1019	4	1007	FIXOPEXP	Fixed Operatin
1020	5	1008	101-1	King Best
1021	5	1008	101-2	King Excellent
1022	5	1008	101-3	King Good
1023	5	1008	101-4	King Fair
1024	5	1008	102-1	Queen Best

Record: 1 of 69 No Filter Search

Figure 11.7

When you look at the accounts, you see that Mattress Revenue rolls up to coa_id 1004, Operating Revenue. Within rollup_id 1004, all of the accounts roll up in the correct order, if you base the rollup on coa_id field. So all we have to do is to bring in the coa_id field at level 4 and that should work. Let's try it. While we're at it, let's fix the formatting for Amt.

21. **Close** *the* **Chart_of_Accounts** *table.*
22. *In* **Design View** *of the query, sort* **Level_3_Desc** *in* **Ascending** *order (to make sure that sorting will remain in place).*
23. *Bring the* **Level_4_ID** *field into the grid to the left of* **Level_4_Desc**.
24. *Make* **Level_4_ID** *sort* **Ascending**.
25. **Format** *the* **Amt** *field to be* **Standard**, **zero decimal places**.
26. **Run** *the query.*

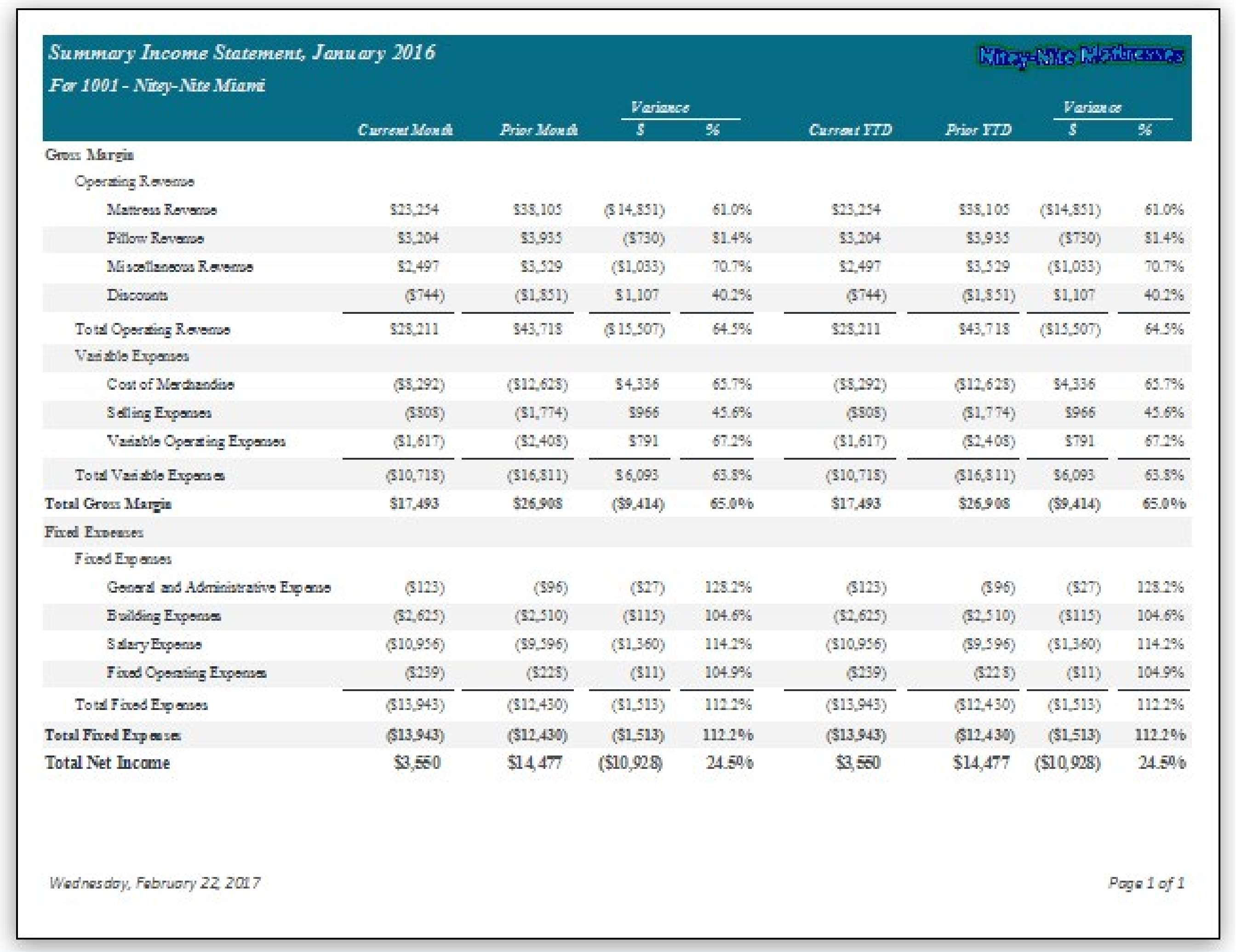


qry10Inc_Stmt

store_no	store_name	Level_1_Desc	Level_2_Desc	Level_3_Desc	Level_4_ID	Level_4_Desc	Ar
1032	Nitey-Nite Pea	Net Income	Gross Margin	Operating Revenue	1008	Mattress Revenue	
1032	Nitey-Nite Pea	Net Income	Gross Margin	Operating Revenue	1009	Pillow Revenue	
1032	Nitey-Nite Pea	Net Income	Gross Margin	Operating Revenue	1010	Miscellaneous Revenue	
1032	Nitey-Nite Pea	Net Income	Gross Margin	Operating Revenue	1011	Discounts	
1032	Nitey-Nite Pea	Net Income	Gross Margin	Variable Expenses	1013	Cost of Merchandise	-
1032	Nitey-Nite Pea	Net Income	Gross Margin	Variable Expenses	1014	Selling Expenses	
1032	Nitey-Nite Pea	Net Income	Gross Margin	Variable Expenses	1015	Variable Operating Expenses	
1032	Nitey-Nite Pea	Net Income	Fixed Expenses	Fixed Expenses	1016	General and Administrative Expenses	
1032	Nitey-Nite Pea	Net Income	Fixed Expenses	Fixed Expenses	1017	Building Expenses	
1032	Nitey-Nite Pea	Net Income	Fixed Expenses	Fixed Expenses	1018	Salary Expense	-

Figure 11.8

NOW it's starting to take shape. Before we get too far into building the query, do you think we should know what the report should look like when finished? Probably so. I really like the phrase, "*Begin with the end in mind.*" (with credit to the late Dr. Stephen R. Covey, one of the most brilliant business minds of our time), and that definitely applies here. Let's review what the summary income statement should look like. Figure 11.9 is the goal for what our statement should look like.



Summary Income Statement, January 2016

For 1001 - Nitey-Nite Miami

Nitey-Nite Mattresses

	Current Month	Prior Month	Variance $	Variance %	Current YTD	Prior YTD	Variance $	Variance %
Gross Margin								
Operating Revenue								
Mattress Revenue	$23,254	$38,105	($14,851)	61.0%	$23,254	$38,105	($14,851)	61.0%
Pillow Revenue	$3,204	$3,935	($730)	81.4%	$3,204	$3,935	($730)	81.4%
Miscellaneous Revenue	$2,497	$3,529	($1,033)	70.7%	$2,497	$3,529	($1,033)	70.7%
Discounts	($744)	($1,851)	$1,107	40.2%	($744)	($1,851)	$1,107	40.2%
Total Operating Revenue	$28,211	$43,718	($15,507)	64.5%	$28,211	$43,718	($15,507)	64.5%
Variable Expenses								
Cost of Merchandise	($8,292)	($12,628)	$4,336	65.7%	($8,292)	($12,628)	$4,336	65.7%
Selling Expenses	($808)	($1,774)	$966	45.6%	($808)	($1,774)	$966	45.6%
Variable Operating Expenses	($1,617)	($2,408)	$791	67.2%	($1,617)	($2,408)	$791	67.2%
Total Variable Expenses	($10,718)	($16,811)	$6,093	63.8%	($10,718)	($16,811)	$6,093	63.8%
Total Gross Margin	**$17,493**	**$26,908**	**($9,414)**	**65.0%**	**$17,493**	**$26,908**	**($9,414)**	**65.0%**
Fixed Expenses								
Fixed Expenses								
General and Administrative Expense	($123)	($96)	($27)	128.2%	($123)	($96)	($27)	128.2%
Building Expenses	($2,625)	($2,510)	($115)	104.6%	($2,625)	($2,510)	($115)	104.6%
Salary Expense	($10,956)	($9,596)	($1,360)	114.2%	($10,956)	($9,596)	($1,360)	114.2%
Fixed Operating Expenses	($239)	($228)	($11)	104.9%	($239)	($228)	($11)	104.9%
Total Fixed Expenses	($13,943)	($12,430)	($1,513)	112.2%	($13,943)	($12,430)	($1,513)	112.2%
Total Fixed Expenses	**($13,943)**	**($12,430)**	**($1,513)**	**112.2%**	**($13,943)**	**($12,430)**	**($1,513)**	**112.2%**
Total Net Income	**$3,550**	**$14,477**	**($10,928)**	**24.5%**	**$3,550**	**$14,477**	**($10,928)**	**24.5%**

Wednesday, February 22, 2017 *Page 1 of 1*

Figure 11.9

Hmmmmm. There are a lot of things in this statement that we don't have in our query. We should have probably looked at this statement before we started building our query. But that's the beauty of building a report that is based on a query – it's much easier to change a query than it is to change a table.

Let's review the statement. The title in the upper-left corner of the report says it is the Summary Income Statement for March 2016. Our data is set to January (Month 1), but that is an easy change. The report also shows the store number and the name of the store, so we have to bring the name of the store into our data source. The ever-so-popular Nitey-Nite logo appears in the upper-right corner of the report. All we have to do there is to bring the graphic into the report, so that's no problem. On the left side of the report, we see the same categories as we have in our query, so there's no need to modify any of those categories.

Now comes the tricky part. There are eight columns of numbers, divided into two sections: the monthly numbers and the year-to-date numbers. Under the monthly numbers, the statement shows a current month and prior month. The current month is March 2016 and the prior month means the prior year's month, which is March 2015. Since Nitey-Nite's business is seasonal, management likes to compare the current month with the same month in the prior year. The variance columns ($ variance and % variance) are simply calculations. We can do those either in the query or in the report. The Current YTD (year-to-date) and Prior YTD columns contain the data from January through March in 2016 and in 2015. Management always wants to see monthly and year-to-date numbers on the same report.

After our review of the income statement, we see that we need to bring in the store name and three additional columns of information into our query. Bringing in the Store Name is easy – just drag it in.

27. *In the* **Design View** *of the query, bring in the* **store_name** *field from the* **dbo_Stores** *table and place it to the right of* **store_no**.

Calculating the three additional columns of data is more challenging, but with the formula writing skills you have, along with a little Access logic, you can do it. Per the report, we need to have one column of March 2016 numbers, a column for March 2015 numbers, then two columns containing January through March numbers for 2016 and 2015. Here is a good piece of advice that you should always remember when building these kinds of reports or analyses. You should generally not name a column or field with a specific year or month. For example, we need to have a column with 2016 numbers and another column with 2015 numbers. Once one year goes by, we'll need 2017 numbers to compare with 2016 numbers. To avoid renaming the columns and revising the report every year, I like to create fields called Current Year, Current Month and Prior Year, Prior Month. This way, you can choose the year/month you want and not have to worry about the names of the fields. Let's start off by naming the Current Month and Current Year fields and changing the month from January to March.

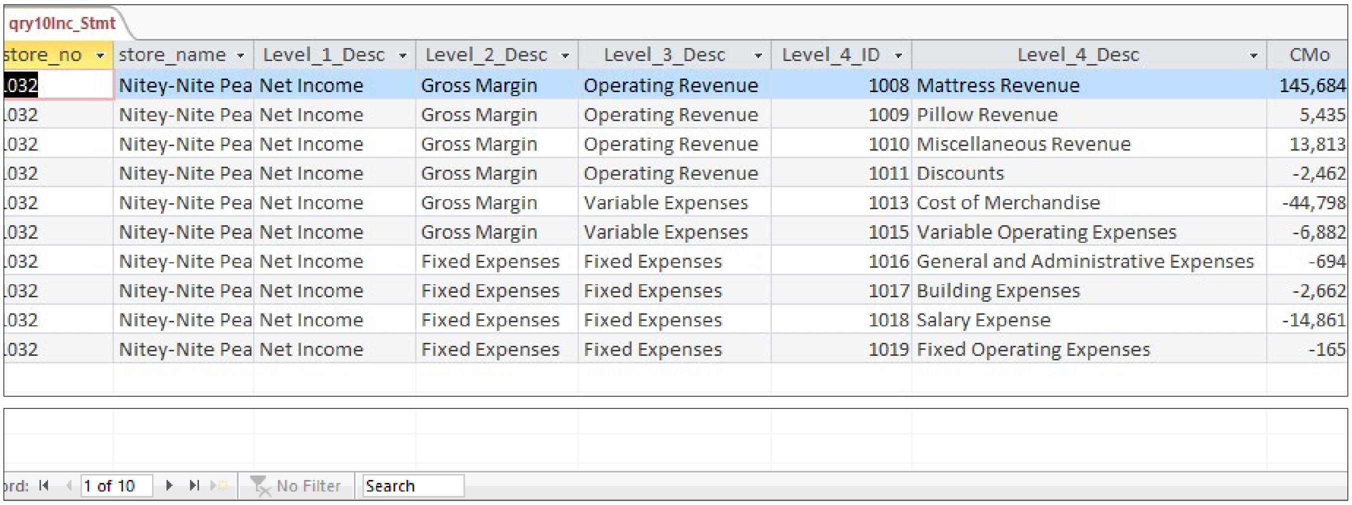
28. *Replace the* **Amt** *alias with* **CMo** *(which stands for Current Month).*
29. *Change the* **Month** *criteria to* **3**.
30. *Change the* **Month** *and* **Year** *fields to show* **Where** *on the* **Total** *line (this can also make the query run a little faster since you're not returning data for those fields).*

A Where clause filters the query for a specified criteria without returning the data in DataSheet View. By default, when a Where clause is used, it turns off the Show property and thus does not display those fields and corresponding values in the record set.

31. *Replace the* **CMo** *formula with the formula* ***IIf(Month([gl_date])=3 And Year([gl_date])=2016,-[amount],0)***.

This formula basically runs the query to return the amount field only when the Month is 3 and the Year is 2016. Trust me, you'll need this logic later on in this chapter, so please just bear with me at this point. You'll soon see why we're setting it up this way.

32. **Run** *the query.*



qry10Inc_Stmt

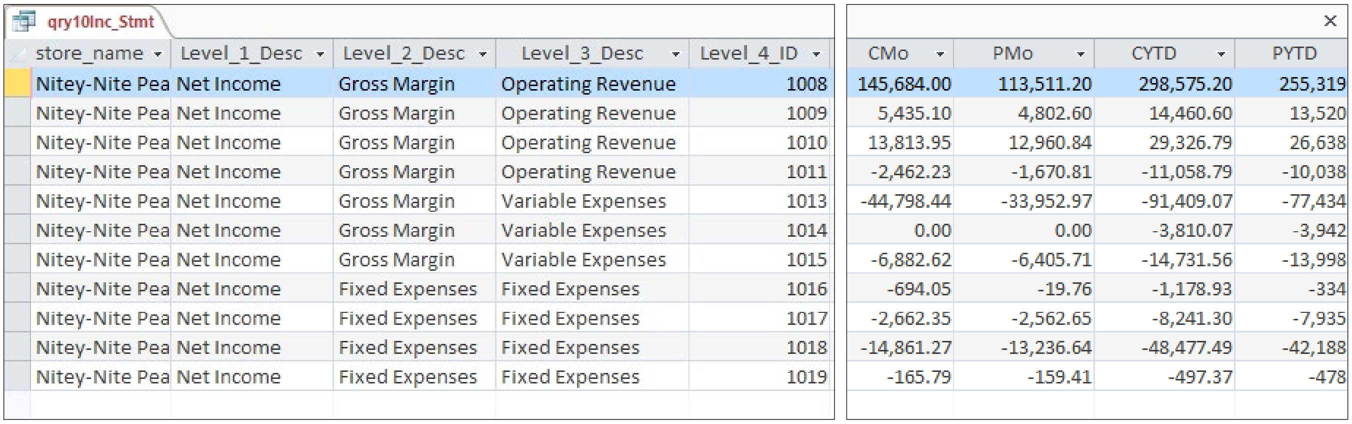
store_no	store_name	Level_1_Desc	Level_2_Desc	Level_3_Desc	Level_4_ID	Level_4_Desc	CMo
032	Nitey-Nite Pea	Net Income	Gross Margin	Operating Revenue	1008	Mattress Revenue	145,684
.032	Nitey-Nite Pea	Net Income	Gross Margin	Operating Revenue	1009	Pillow Revenue	5,435
.032	Nitey-Nite Pea	Net Income	Gross Margin	Operating Revenue	1010	Miscellaneous Revenue	13,813
.032	Nitey-Nite Pea	Net Income	Gross Margin	Operating Revenue	1011	Discounts	-2,462
.032	Nitey-Nite Pea	Net Income	Gross Margin	Variable Expenses	1013	Cost of Merchandise	-44,798
.032	Nitey-Nite Pea	Net Income	Gross Margin	Variable Expenses	1015	Variable Operating Expenses	-6,882
.032	Nitey-Nite Pea	Net Income	Fixed Expenses	Fixed Expenses	1016	General and Administrative Expenses	-694
.032	Nitey-Nite Pea	Net Income	Fixed Expenses	Fixed Expenses	1017	Building Expenses	-2,662
.032	Nitey-Nite Pea	Net Income	Fixed Expenses	Fixed Expenses	1018	Salary Expense	-14,861
.032	Nitey-Nite Pea	Net Income	Fixed Expenses	Fixed Expenses	1019	Fixed Operating Expenses	-165

rd: 1 of 10 No Filter Search

Figure 11.10

Now that you have a field containing the amounts for the current month, all you have to do is copy that formula and change the variables to return the desired results.

33. *Create the following fields:*
    PMo: IIf(Month([gl_date])=3 And Year([gl_date])=2015,-amount,0)
    CYTD: IIf(Month([gl_date]) Between 1 And 3 And Year([gl_date])=2016,-amount,0)
    PYTD: IIf(Month([gl_date]) Between 1 And 3 And Year([gl_date])=2015,-amount,0)
34. *Change the* **Total** *line on all the new formula fields to* **Expression**.
35. *Modify the criteria in the* **Year** *field to be* ***2016 or 2015***.
36. *Modify the criteria in the* **Month** *field to be* ***Between 1 and 3***.
37. **Format** *all amount fields to be* **Standard**, **zero decimal places**.
38. **Save** *and* **Run** *the query.*



qry10Inc_Stmt

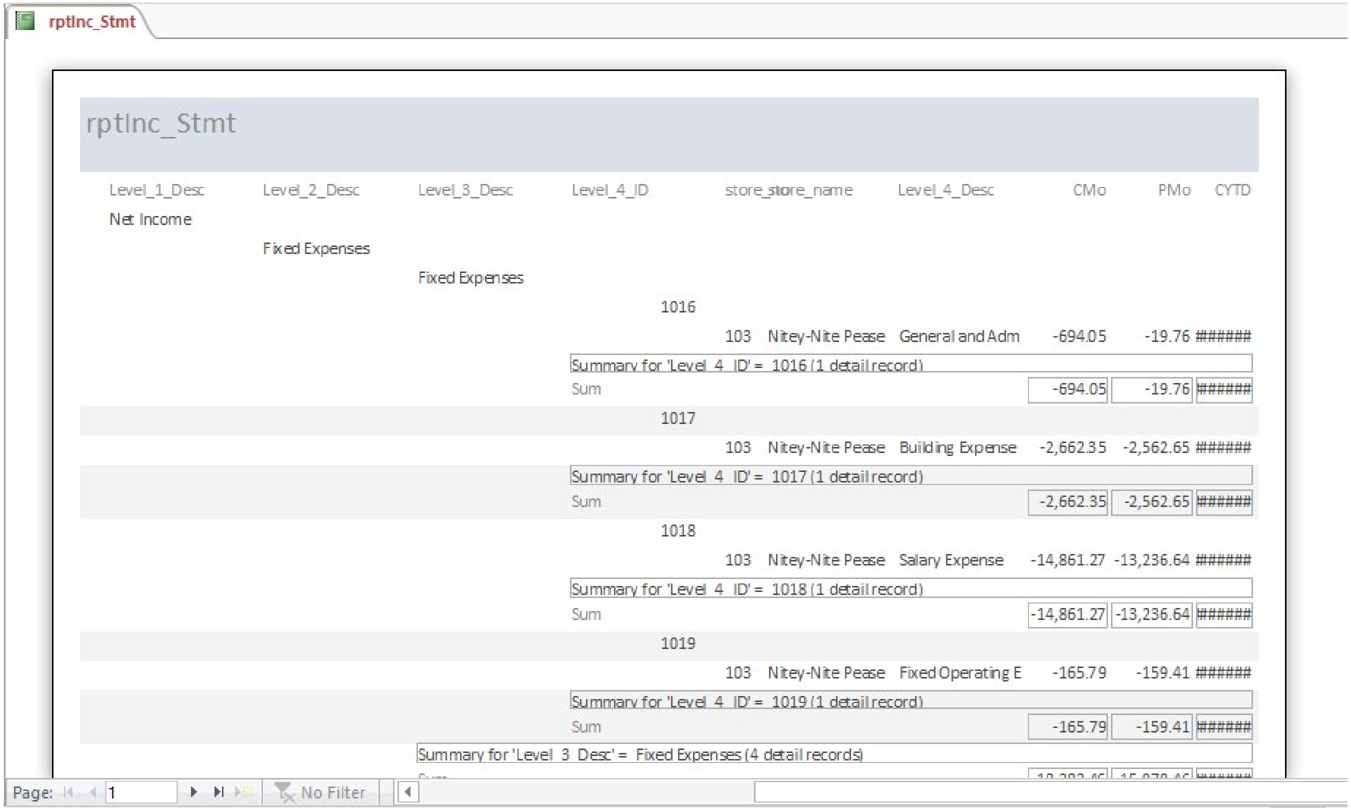
store_name	Level_1_Desc	Level_2_Desc	Level_3_Desc	Level_4_ID	CMo	PMo	CYTD	PYTD
Nitey-Nite Pea	Net Income	Gross Margin	Operating Revenue	1008	145,684.00	113,511.20	298,575.20	255,319
Nitey-Nite Pea	Net Income	Gross Margin	Operating Revenue	1009	5,435.10	4,802.60	14,460.60	13,520
Nitey-Nite Pea	Net Income	Gross Margin	Operating Revenue	1010	13,813.95	12,960.84	29,326.79	26,638
Nitey-Nite Pea	Net Income	Gross Margin	Operating Revenue	1011	-2,462.23	-1,670.81	-11,058.79	-10,038
Nitey-Nite Pea	Net Income	Gross Margin	Variable Expenses	1013	-44,798.44	-33,952.97	-91,409.07	-77,434
Nitey-Nite Pea	Net Income	Gross Margin	Variable Expenses	1014	0.00	0.00	-3,810.07	-3,942
Nitey-Nite Pea	Net Income	Gross Margin	Variable Expenses	1015	-6,882.62	-6,405.71	-14,731.56	-13,998
Nitey-Nite Pea	Net Income	Fixed Expenses	Fixed Expenses	1016	-694.05	-19.76	-1,178.93	-334
Nitey-Nite Pea	Net Income	Fixed Expenses	Fixed Expenses	1017	-2,662.35	-2,562.65	-8,241.30	-7,935
Nitey-Nite Pea	Net Income	Fixed Expenses	Fixed Expenses	1018	-14,861.27	-13,236.64	-48,477.49	-42,188
Nitey-Nite Pea	Net Income	Fixed Expenses	Fixed Expenses	1019	-165.79	-159.41	-497.37	-478

Figure 11.11

Build the Summary Report

Your numbers should tie with those in Figure 11.11. With this query as your data source, you are now ready to begin building the report. Since you already know how to use the wizard, let's use it to create the shell of the report then we'll go in and modify it as needed.

1. *Close* **qry10Inc_Stmt***.*
2. *In the* **Navigation Pane***, click on* **qry10Inc_Stmt***.*
3. *Click on the* **Create** *tab then click on the* **Report Wizard** Report Wizard *icon in the* **Reports** *group.*
4. *Base your report on* **qry10Inc_Stmt** *and choose all fields.*
5. *Group by* **Level_1_Desc***,* **Level_2_Desc***,* **Level_3_Desc***, and* **Level_4_ID** *(the wizard will allow up to four grouping levels – you can add more later).*
6. *Click* **Next***.*
7. *Don't choose a* **Sorting** *field, but click on the* **Summary Options…** *button and choose* **Sum** *for all of the amount fields, then click* **OK***.*
8. *Click* **Next** *in the* **Report Wizard** *dialog box, then choose a* **Stepped** *layout and* **Landscape** *orientation options.*
9. *Click* **Next***.*
10. *Name the report* ***rptInc_Stmt*** *and click* **Finish***.*



```
rptInc_Stmt

Level_1_Desc  Level_2_Desc   Level_3_Desc    Level_4_ID   store_store_name   Level_4_Desc        CMo        PMo  CYTD
Net Income
              Fixed Expenses
                             Fixed Expenses
                                                   1016
                                                          103  Nitey-Nite Pease  General and Adm    -694.05     -19.76 #######
                                             Summary for 'Level_4_ID' = 1016 (1 detail record)
                                             Sum                                                    -694.05     -19.76 #######
                                                   1017
                                                          103  Nitey-Nite Pease  Building Expense  -2,662.35  -2,562.65 #######
                                             Summary for 'Level_4_ID' = 1017 (1 detail record)
                                             Sum                                                  -2,662.35  -2,562.65 #######
                                                   1018
                                                          103  Nitey-Nite Pease  Salary Expense   -14,861.27 -13,236.64 #######
                                             Summary for 'Level_4_ID' = 1018 (1 detail record)
                                             Sum                                                 -14,861.27 -13,236.64 #######
                                                   1019
                                                          103  Nitey-Nite Pease  Fixed Operating E  -165.79    -159.41 #######
                                             Summary for 'Level_4_ID' = 1019 (1 detail record)
                                             Sum                                                    -165.79    -159.41 #######
                             Summary for 'Level_3_Desc' = Fixed Expenses (4 detail records)
```

Figure 11.12

Ooooo! That report looks uglier than when my three-eyed bull ate a bunch of bitterweed, swelled up like a hot air balloon, went crazy, and mangled himself in the barbed wire! It doesn't look anything like the report in Figure 11.9, but that's OK. It has many of the components necessary to build our report – we just have to put things in the right order, do some cleanup, and create a few formulas. We'll use the report in Figure 11.9 as a guide.

In the next few exercises, you're going to be moving around a lot of the labels and text boxes in the report's Design View.

11. *Go to the* **Design View** *of the report.*

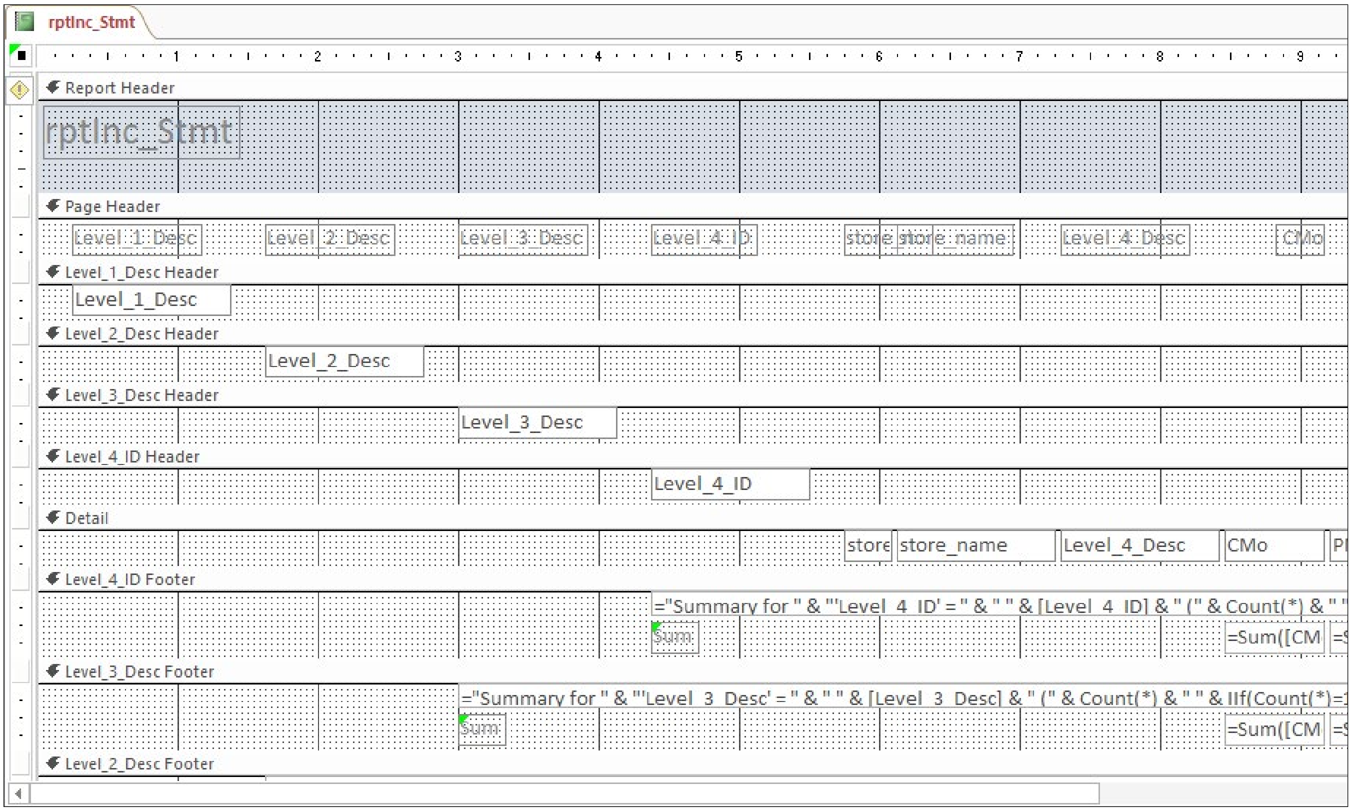


Figure 11.13

Let's start at the top. When you create a report using the wizard, Access creates a report header. Like I said before in the Forms section, I personally don't use the form (or report) header much, so let's take it out. We'll move the Report Header label down to the Page Header section. We'll then modify the label to read the name of the report and its "as of" date, and create another label that displays the store number and name of the store, and then import the Nitey-Nite logo. As you will notice, not all of the fields in the query came over in the design of the report, like the PYTD field, so we'll add in some fields. Finally, we'll create new labels, or edit existing ones to show the names of each column, and do a lot of rearranging.

As you're working through the design of the report, you will need to occasionally run the report to make sure it's doing what you want. From this point on, I won't tell you to run the report unless it's absolutely necessary – I'll assume you're doing that on your own.

12. *In* **Design View** *of the report, expand the* **Page Header** *section to be* **1"** *tall.*
13. *Close any dialog boxes that may appear.*
14. *Move all labels in the* **Page Header** *section down to the bottom part of that section.*
15. *Move the* **rptInc_Stmt** *label in the* **Report Header** *to the upper-left corner of the* **Page Header** *section and edit it to read* ***Summary Income Statement, March 2016****.*
16. **Format** *it to be* **Times New Roman***,* **14 pt***,* **bold***, and* **italicized***.*
17. *Rearrange the borders of the label to fit over its text, if necessary.*

> ***Tip****: You can right-click on the* **label***, click* **Size***, and select* **To Fit***.*

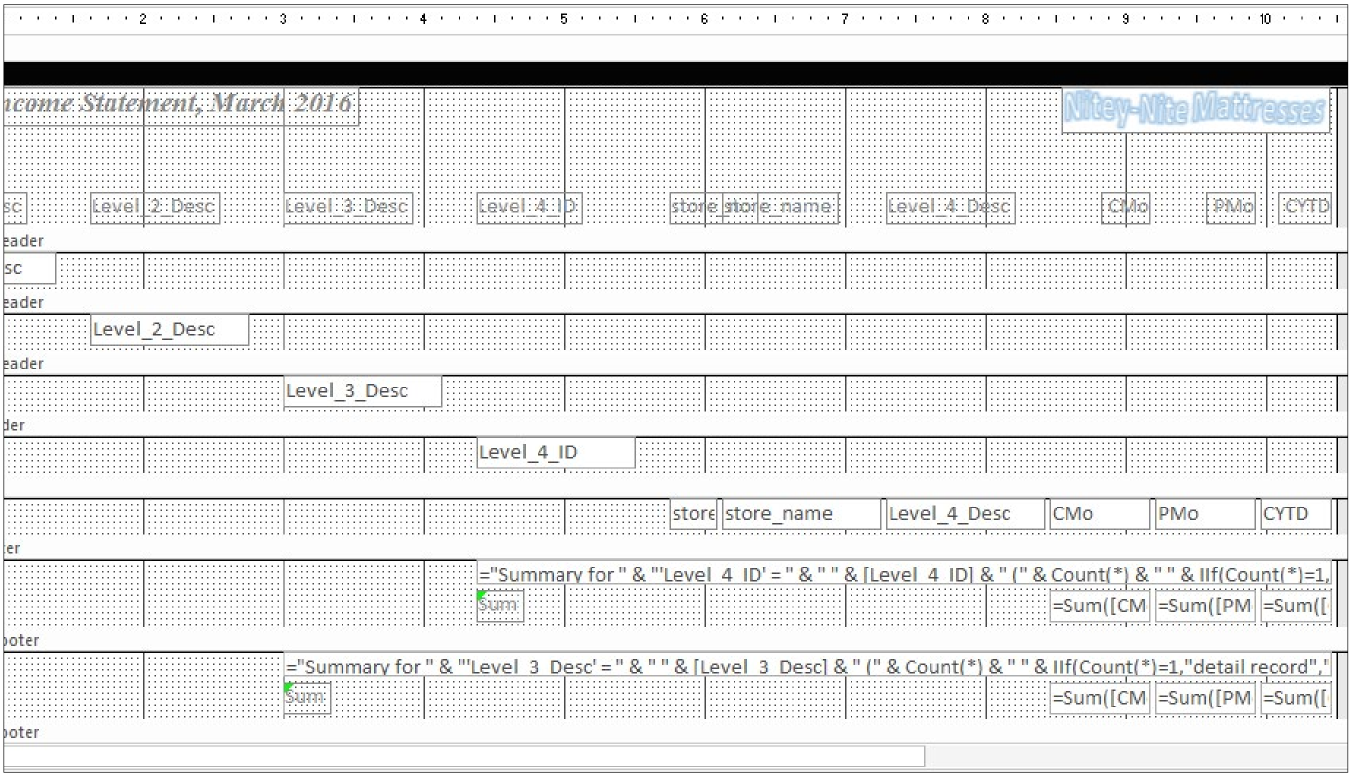
18. *Collapse the* **Report Header** *section.*
19. *Import the* **Nitey-Nite logo** *from the* **Access 2016** *folder into the* **Page Header** *section of the report* **Design View***.*

> ***Hint****: use the* **Insert Image** *icon in the* **Controls** *group of the* **Design** *tab.*

Layout Grouping

When you import the logo, you have to draw a rectangle where you want the logo to be and then adjust the size. If you were to use the Logo icon to import the logo, it may be very small and may have a dotted line surrounding the logo as well as some space to the right of the logo. This is called a ***Layout Grouping***. This generally occurs when the Report Wizard can identify certain objects that should be grouped together, but sometimes it happens with an individual object, like our logo image. With this layout in place, it is difficult to move the logo within the Design view of the report. Insert Image is an easier option to help speed up cleaning up this report.

20. *Adjust the size and position of the* **Logo.png** *image to match the following image:*



="Summary for " & "'Level_4_ID' = " & " " & [Level_4_ID] & " (" & Count(*) & " " & IIf(Count(*)=1,

="Summary for " & "'Level_3_Desc' = " & " " & [Level_3_Desc] & " (" & Count(*) & " " & IIf(Count(*)=1,"detail record",'

Figure 11.14

21. *Adjust the properties of the logo so there are* **no borders** *around it and* **no background colors**. *Make the logo be* **2"** *wide and* **0.38"** *tall.*
22. *Create another* **label** *under the report name to read* ***For Store 1032 – Nitey-Nite Pease***.
23. **Format** *that label to be* **Times New Roman**, **12 pt**, **bold** *and* **italicized**.
22. *Delete all of the labels in the bottom part of the* **Page Header** *section.*
23. *Create the labels that are shown in* **Figure 11.15**.
24. **Format** *all of those labels to be* **Times New Roman**, **10 pt**, **bold**, *and* **italicized**, **no background color**, **Text Light** *color, and position the labels as in* **Figure 11.15**.
25. *Using the* **Line** *icon, draw a line under each* **Variance** *label to expand over the* **$** *and* **%** *labels. Make the lines have a* **Border Style** *property of* **Solid**, *a* **Border Width** *of* **1 pt**, *and a* **Border Color** *of* **Text Light**.

> ***Hint***: *If you hold* **Ctrl** *while drawing the* **Line**, *you will have a much easier time making the line straight.*

26. *For the background color of the* **Page Header**, *use the* ***#04617B*** *color we used on the form* **frm06Employee** *earlier.*
27. *Make the* **"Summary Income Statement, March 2016"** *and the* **"For Store 1032 – Nitey-Nite Pease"** *labels a font color of* **Text Light**.

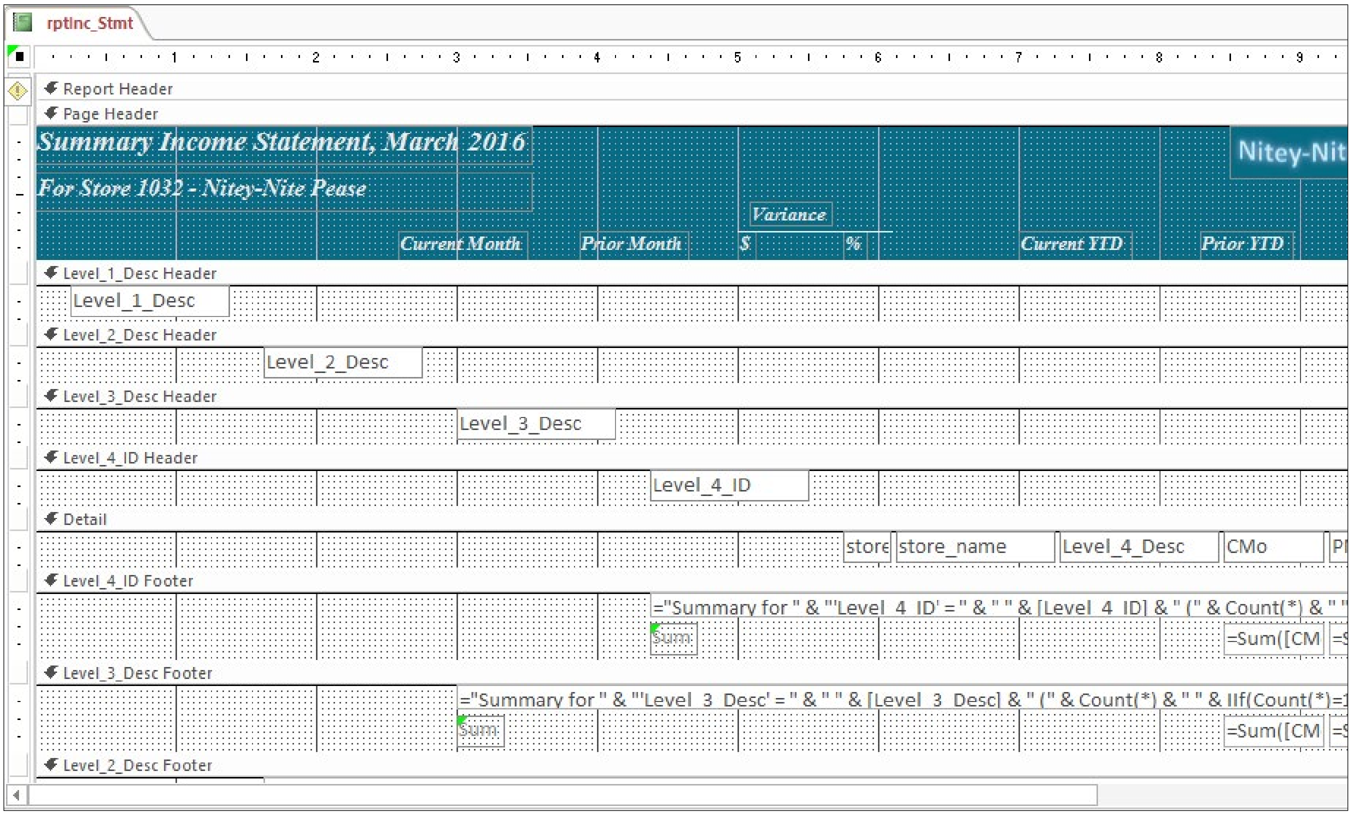
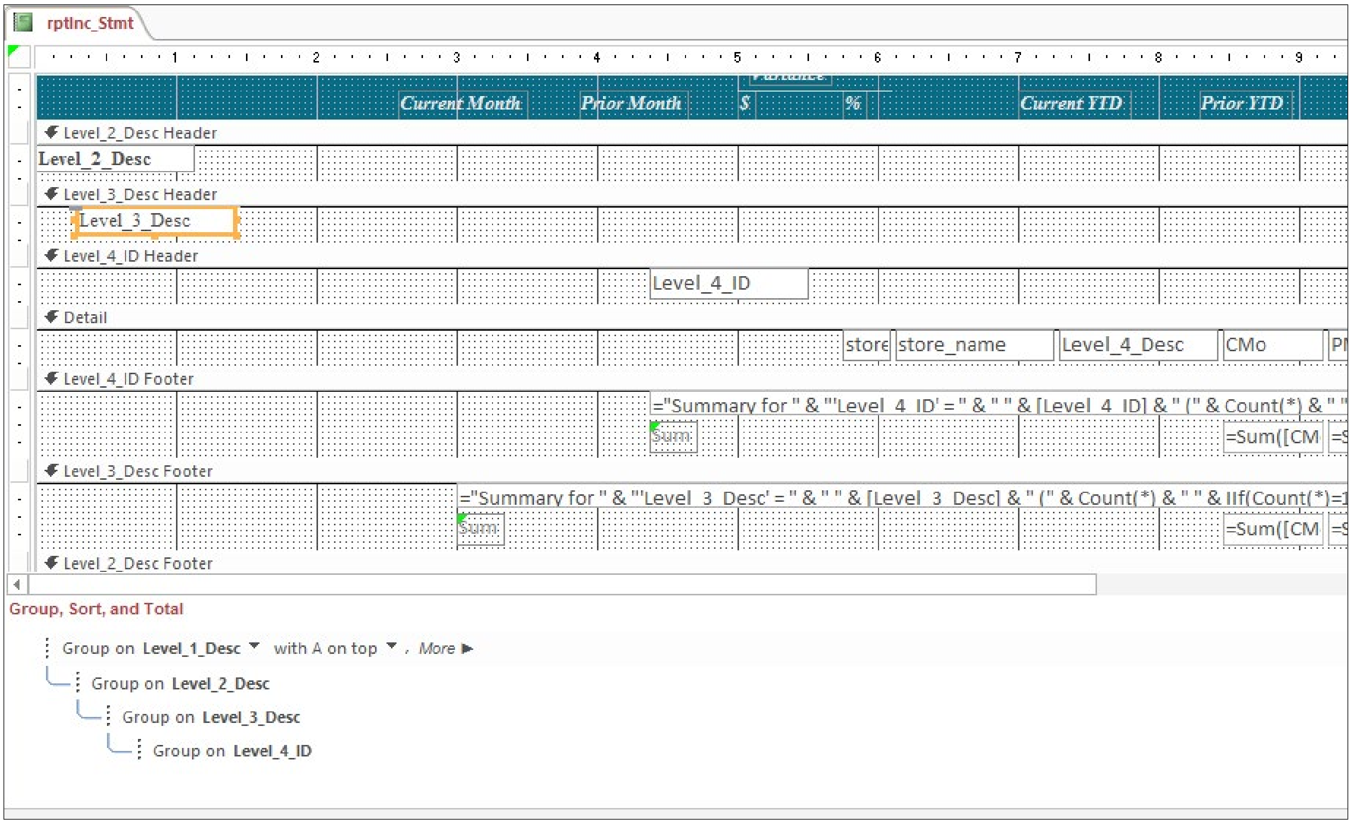


Figure 11.15

These are the header and labels that will appear on every page in the report. Next, we'll get into the body of the report. According to Figure 11.9, the first heading we want to show is Gross Margin, which is on a Level 2. We won't create a header for Level 1. We will then format the appropriate boxes for Level 2 and Level 3.

1. *Delete the* **Level_1_Desc** *text box in the* **Level_1_Desc Header** *section and delete the* **Level_1_Desc Header** *section (using the Group, Sort, and Total section).*
2. *In the* **Level_2_Desc** *section, move the* **Level_2_Desc** *text box all the way to the left and format it as* **Times New Roman, 10 pt, bold**.
3. *With the* **Level_2_Desc** *text box selected, click on the* **Arrange** *tab, click on the* **Size/Space** *icon, and choose* **To Fit** *(to make it fit to the data).*
4. *Turn on the* **Group, Sort, and Total** *section, click on the* **Group on Level_2_Desc** *group and change the* **with A on top** *to* **with Z on top** *(to sort the group in Descending order).*
5. *Repeat steps 2 – 4 for the* **Level_3_Desc** *group, except do not make it bold and offset it about* **7 grid dots** *to the right of* **Level_2_Desc**.
6. *Make sure your report looks like* **Figure 11.16**.



```
="Summary for " & "'Level_4_ID' = " & " " & [Level_4_ID] & " (" & Count(*) & " "
=Sum([CM
="Summary for " & "'Level_3_Desc' = " & " " & [Level_3_Desc] & " (" & Count(*) & " " & IIf(Count(*)=1
=Sum([CM
```

Group, Sort, and Total

Group on **Level_1_Desc** ▾ with A on top ▾ , *More* ▶
Group on **Level_2_Desc**
Group on **Level_3_Desc**
Group on **Level_4_ID**

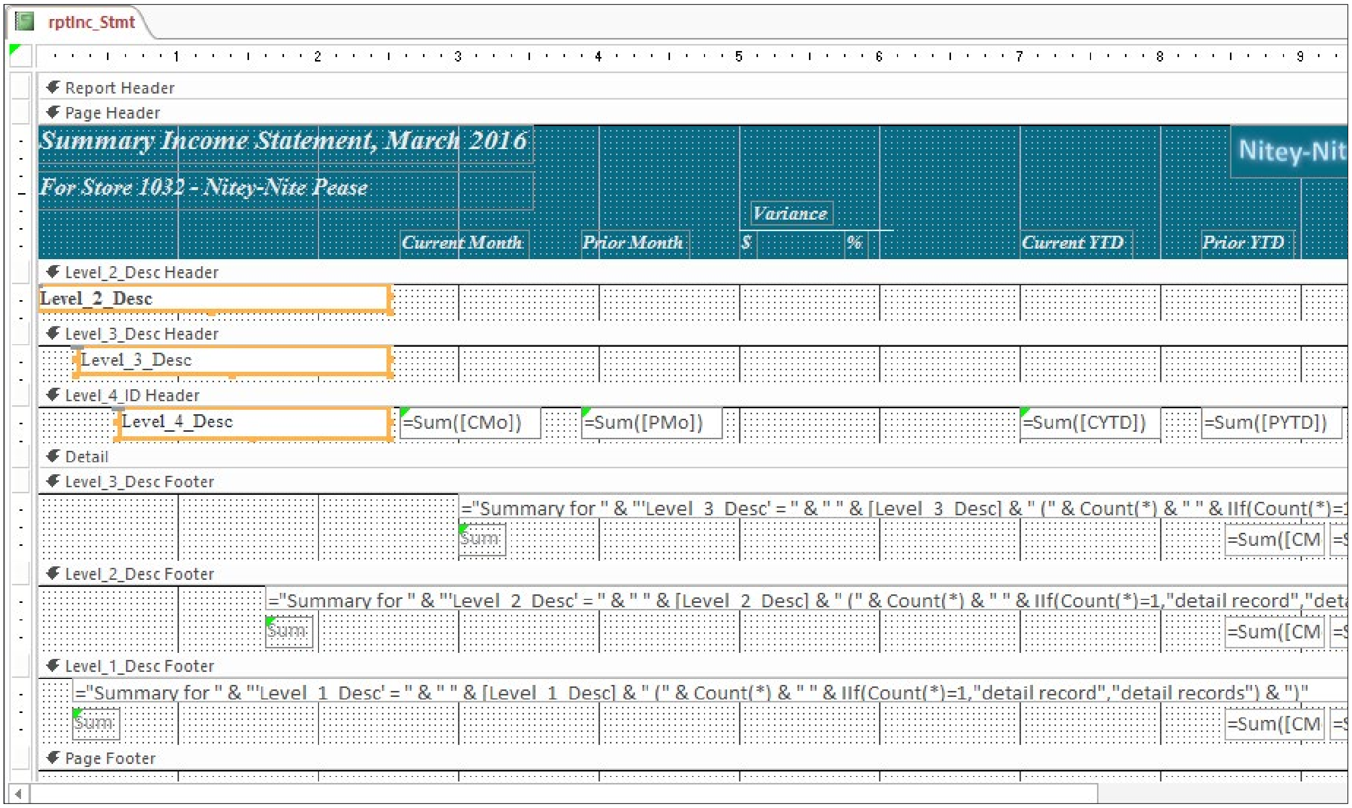
Figure 11.16

Next we'll modify the lowest level of account activity, Level 4. You need to pay close attention to this step as we're going to significantly change up the design that the wizard created. Remember, this is the Summary report. Later on we'll create the Detail report, which is one level lower than this report. We'll use the Summary report as a base from which we'll create the Detail report. Therefore, we're going to put the Level 4 data in the Level_4_ID section, sum the text boxes, and then take out the Detail section.

7. *In the* **Level_4_ID Header** *section, change the* **Level_4_ID** *text box to read* **Level_4_Desc** *and move it to the left, offset about* **7 grid dots** *to the right of* **Level_3_Desc***.*
8. *Make the* **Level_4_Desc** *text box* **Times New Roman***,* **10 pt***, no bold or italics and make the font* **fore color Text Black***.*
9. *In the* **Detail** *section, delete the* **store_no***,* **store_name***, and* **Level_4_Desc** *text boxes.*
10. *Move the* **CMo***,* **PMo***, and* **CYTD** *text boxes from the* **Detail** *section to the* **Level_4_ID Header** *section.*
11. *Modify each of the* **CMo***,* **PMo***, and* **CYTD** *text boxes to include a* **Sum()** *function (ex.,* ***=Sum([CMo])****).*
12. *Create a similar text box for* ***PYTD****.*
13. *Format all text boxes in the* **Level_4_ID** *section to be* **Times New Roman***,* **10 pt***.*
14. *Position the summed* **CMo***,* **PMo***,* **CYTD***, and* **PYTD** *text boxes under their appropriate*

columns in the **Page Header** *section, right justify the text boxes and format them to be* **Currency** *with* **no decimal places**.

15. *Expand the* **Amount** *text boxes to a width that would reasonably display the amounts (you may have to go back and forth between* **Print Preview** *and* **Design Views** *a few times to get it just right.).*
16. *With nothing in the* **Detail** *section, move the* **Level_4_ID Footer** *section up to completely close out the* **Detail** *section.*
17. *Delete everything in the* **Level_4_ID Footer** *section.*
18. *Take out the* **Level_4_ID Footer** *section.*
19. *Increase the width of the* **Level_2_Desc**, **Level_3_Desc**, *and* **Level_4_Desc** *text boxes to be close to the* **Sum([CMo])** *text box.*



rptInc_Stmt

Report Header

Page Header

Summary Income Statement, March 2016

For Store 1032 - Nitey-Nite Pease

Nitey-Nit

Variance

Current Month | Prior Month | $ | % | Current YTD | Prior YTD

Level_2_Desc Header

Level_2_Desc

Level_3_Desc Header

Level_3_Desc

Level_4_ID Header

Level_4_Desc | =Sum([CMo]) | =Sum([PMo]) | =Sum([CYTD]) | =Sum([PYTD])

Detail

Level_3_Desc Footer

```
="Summary for " & "'Level_3_Desc' = " & " " & [Level_3_Desc] & " (" & Count(*) & " " & IIf(Count(*)=1
Sum   =Sum([CM   =S
```

Level_2_Desc Footer

```
="Summary for " & "'Level_2_Desc' = " & " " & [Level_2_Desc] & " (" & Count(*) & " " & IIf(Count(*)=1,"detail record","deta
Sum   =Sum([CM   =S
```

Level_1_Desc Footer

```
="Summary for " & "'Level_1_Desc' = " & " " & [Level_1_Desc] & " (" & Count(*) & " " & IIf(Count(*)=1,"detail record","detail records") & ")"
Sum   =Sum([CM   =S
```

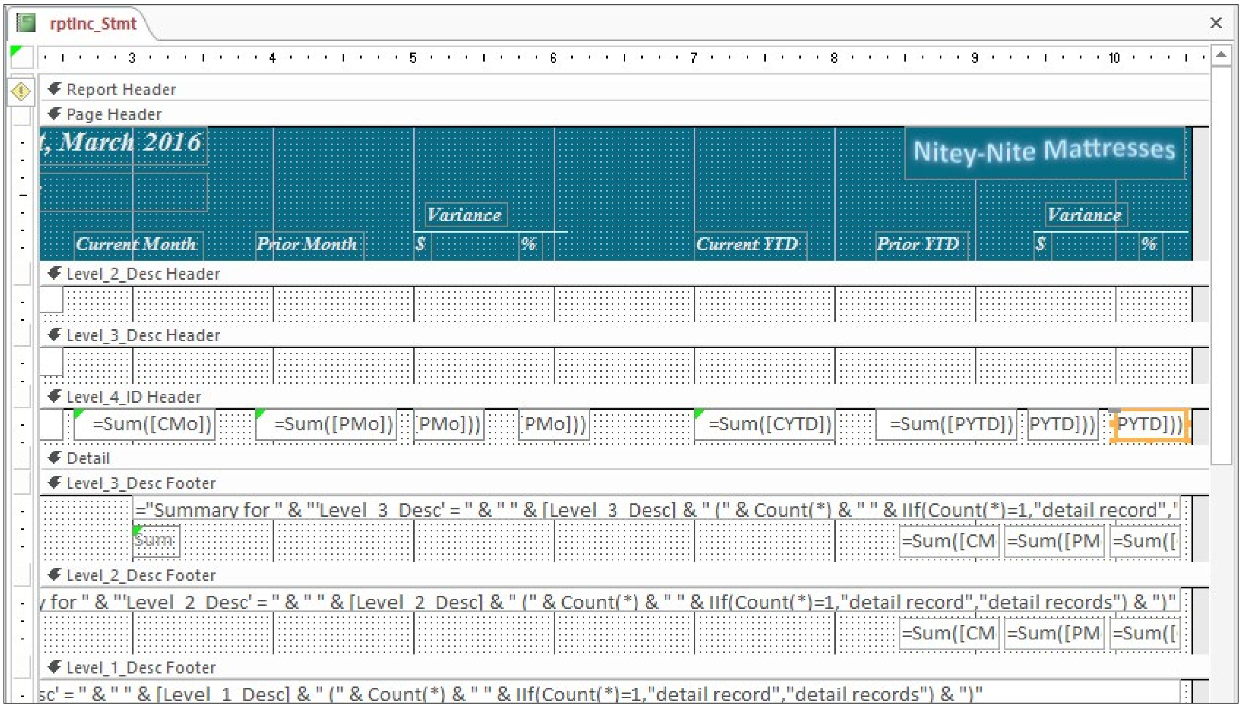
Page Footer

Figure 11.17

Now that you have numbers from the lowest level coming in, you need to calculate the variances. In the report, you need to show the dollar variance and the percent variance. The dollar variance is the current month minus the prior month, and current YTD minus prior year. The percent variance would include an IIF() function, where if the prior month (or YTD) equals zero, return a zero, else divide current month (or YTD) by prior month (or YTD). You should be able to do this one on your own. Refer back to the formulas you wrote in Chapter Nine, if need be.

20. **Create** *the variance* **text boxes** *for the* **dollar** *and* **percent** *variances for the* **Level_4_ID Header CMo**, **PMo**, **CYTD**, *and* **PYTD** *boxes.*

21. **Format** *the* **dollar** *variances as* **Currency**, **zero decimal places**, *and the* **percent** *variances as* **Percent**, **one decimal place**.



rptInc_Stmt

Report Header

Page Header

t, March 2016

Nitey-Nite Mattresses

Variance

Current Month | Prior Month | $ | %

Current YTD | Prior YTD

Variance

$ | %

Level_2_Desc Header

Level_3_Desc Header

Level_4_ID Header

=Sum([CMo]) | =Sum([PMo]) | PMo])) | PMo])) | =Sum([CYTD]) | =Sum([PYTD]) | PYTD])) | PYTD]))

Detail

Level_3_Desc Footer

="Summary for " & "'Level 3 Desc' = " & " " & [Level 3 Desc] & " (" & Count(*) & " " & IIf(Count(*)=1,"detail record","

Sum

=Sum([CM | =Sum([PM | =Sum([

Level_2_Desc Footer

for " & "'Level 2 Desc' = " & " " & [Level 2 Desc] & " (" & Count(*) & " " & IIf(Count(*)=1,"detail record","detail records") & ")"

=Sum([CM | =Sum([PM | =Sum([

Level_1_Desc Footer

sc' = " & " " & [Level 1 Desc] & " (" & Count(*) & " " & IIf(Count(*)=1,"detail record","detail records") & ")"

Figure 11.18

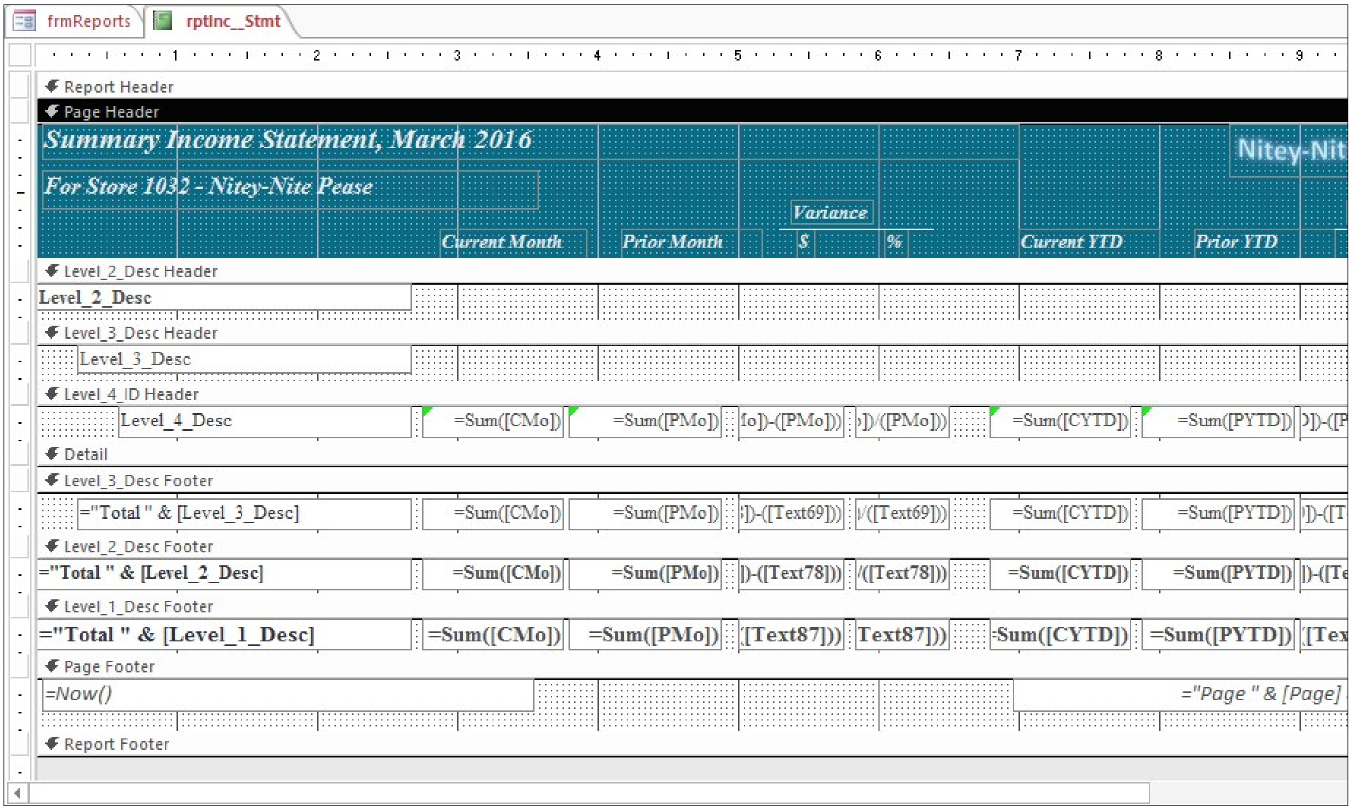
Next you will modify the formulas and text boxes in the Level_3_Desc footer section. All you want here are subtotals for the accounts that are flowing through in the Level_4_ID section.

22. *Delete all text boxes in the* **Level_3_Desc_Footer** *section.*
23. *Copy all of the text boxes in the* **Level_4_ID Header** *section and paste them into the* **Level_3_Desc Footer** *section.*
24. *Align the text boxes that contain number values under the appropriate headings.*
25. *Make sure all objects in the* **Level_3_Desc Footer** *section are formatted as* **Times New Roman**, **10 pt**.
26. *In the* **Level_3_Desc Footer**, *modify the text box that reads* **Level_4_Desc** *to* ***="Total " & [Level_3_Desc]*** *and left-align it with the* **Level_3_Desc** *text box in the* **Level_3_Desc Header** *section.*

You do this so that the word Total appears in front of (or concatenated to) the account name in the footer section.

27. *Repeat the same procedure for the* **Level_2_Desc** *and* **Level_1_Desc Footer** *sections.*
28. **Format** *all text boxes in the* **Level_2_Desc Footer** *section as* **Times New Roman**, **10 pt**, **bold**, *and all text boxes in the* **Level_1_Desc Footer** *section as* **Times New Roman**, **11 pt**, **bold**.

29. *Move the* **page count text box** *in the* **Page Footer** *section to align with the right side of the report, if necessary.*
30. *Delete all entries in the* **Report Footer** *section and collapse that section.*
31. *Make sure that all dollar amounts are* **Currency**, **zero decimal places**. *Percentage amounts should be formatted as* **Percent**, **one decimal place**.
32. *Italicize the text boxes in the* **Page Footer** *section.*
33. *Adjust the* **Header** *and* **Footer** *sections to have only two dots below the text boxes visible.*
34. *Close the* **Group, Sort, & Total** *section.*
35. *Go back and forth between* **Design View** *and* **Report View** *to make sure that all numbers and descriptions are displayed appropriately.*



Summary Income Statement, March 2016
For Store 1032 - Nitey-Nite Pease
Nitey-Nit
Variance
Current Month | Prior Month | $ | % | Current YTD | Prior YTD

Level_2_Desc

Level_3_Desc

Level_4_Desc | =Sum([CMo]) | =Sum([PMo]) | [o])-([PMo]) |)])/([PMo]) | =Sum([CYTD]) | =Sum([PYTD]) |)])-([P

="Total " & [Level_3_Desc] | =Sum([CMo]) | =Sum([PMo]) |])-([Text69]) |)/([Text69]) | =Sum([CYTD]) | =Sum([PYTD]) |])-([T

="Total " & [Level_2_Desc] | =Sum([CMo]) | =Sum([PMo]) |])-([Text78]) | /([Text78]) | =Sum([CYTD]) | =Sum([PYTD]) |])-([Te

="Total " & [Level_1_Desc] | =Sum([CMo]) | =Sum([PMo]) | [Text87]) | Text87]) | Sum([CYTD]) | =Sum([PYTD]) | [Tex

=Now() | ="Page " & [Page]

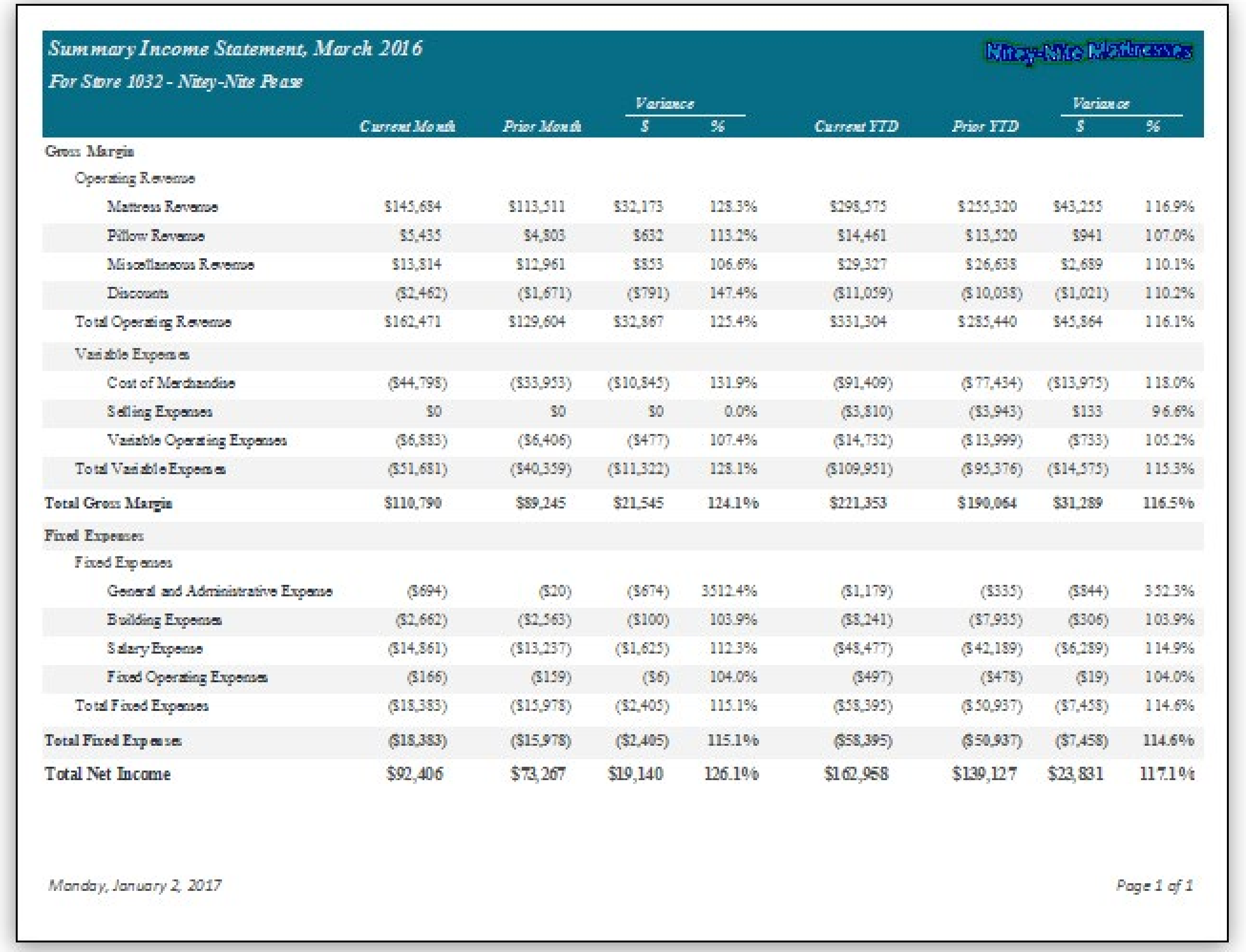
Figure 11.19

Now you are ready to view the report in Print Preview mode. Remember that in Print Preview mode, the report will appear exactly as it will print. The Report Wizard in Access 2016 usually does a very good job in identifying most of the print properties (like setting the appropriate margins, orientation, etc.) of a report and generally speaking the Print Preview should work just fine. At this point, you should view the report in Print Preview to make sure the report's print properties are set appropriately.

36. *View the report in* **Print Preview** *mode.*
37. *If the report doesn't appear correct in* **Print Preview** *mode, change the settings in the* **Print Preview** *ribbon or click on the* **Page Setup** *icon in the* **Page Layout** *group and change the*

necessary settings for the report to print on one page.

38. **Save** *the report and* **Run** *it again in* **Print Preview** *mode.*



Summary Income Statement, March 2016

For Store 1032 - Nitey-Nite Pease

	Current Month	Prior Month	Variance		Current YTD	Prior YTD	Variance	
			$	%			$	%
Gross Margin								
Operating Revenue								
Mattress Revenue	$145,684	$113,511	$32,173	128.3%	$298,575	$255,320	$43,255	116.9%
Pillow Revenue	$5,435	$4,803	$632	113.2%	$14,461	$13,520	$941	107.0%
Miscellaneous Revenue	$13,814	$12,961	$853	106.6%	$29,327	$26,638	$2,689	110.1%
Discounts	($2,462)	($1,671)	($791)	147.4%	($11,059)	($10,038)	($1,021)	110.2%
Total Operating Revenue	$162,471	$129,604	$32,867	125.4%	$331,304	$285,440	$45,864	116.1%
Variable Expenses								
Cost of Merchandise	($44,798)	($33,953)	($10,845)	131.9%	($91,409)	($77,434)	($13,975)	118.0%
Selling Expenses	$0	$0	$0	0.0%	($3,810)	($3,943)	$133	96.6%
Variable Operating Expenses	($6,883)	($6,406)	($477)	107.4%	($14,732)	($13,999)	($733)	105.2%
Total Variable Expenses	($51,681)	($40,359)	($11,322)	128.1%	($109,951)	($95,376)	($14,575)	115.3%
Total Gross Margin	$110,790	$89,245	$21,545	124.1%	$221,353	$190,064	$31,289	116.5%
Fixed Expenses								
Fixed Expenses								
General and Administrative Expense	($694)	($20)	($674)	3512.4%	($1,179)	($335)	($844)	352.3%
Building Expenses	($2,662)	($2,563)	($100)	103.9%	($8,241)	($7,935)	($306)	103.9%
Salary Expense	($14,861)	($13,237)	($1,625)	112.3%	($48,477)	($42,189)	($6,289)	114.9%
Fixed Operating Expenses	($166)	($159)	($6)	104.0%	($497)	($478)	($19)	104.0%
Total Fixed Expenses	($18,383)	($15,978)	($2,405)	115.1%	($58,395)	($50,937)	($7,458)	114.6%
Total Fixed Expenses	($18,383)	($15,978)	($2,405)	115.1%	($58,395)	($50,937)	($7,458)	114.6%
Total Net Income	$92,406	$73,267	$19,140	126.1%	$162,958	$139,127	$23,831	117.1%

Monday, January 2, 2017

Page 1 of 1

Figure 11.20

Now you have a report that is tied to a query and looks good.

> ***Review Questions****: It is now time to complete the hands-on Review Questions. Log on to www.ExcelCEO.com with your Email and Password, click on the* **Access 2016 and SQL Review Questions, Chapter 11, Section 1 of 2** *option and complete the review questions.*

I hope that you've asked the question, "*Isn't this report hard-coded for March 2016 and Store No 1032? How am I going to get reports for the other stores, or a rollup of all of the stores in a city, state, region, or for the whole company? Also, I need to be able to change the dates. How am I going to do that?*" If you've asked yourself these questions, bless you! That is exactly what I wanted you to think about. If you didn't, you should have. Let's talk about how we can get that accomplished.

Tie Report Variables to a Form

It should be obvious that you will probably not be the only one running these statements out of this database. When I create reports like this, I like to make them as "bone-stupid simple" as possible for any user to run. Not trying to insult the intelligence of users, but there aren't too many people who know how to use Access. Therefore, we have to make the report REAL simple to use. If anyone calls you and asks a question on how to navigate through the report, it's probably not easy enough, and you should make it easier. One time I was involved with a report that took a one-hour training session on how to read and navigate through the report. That was WAY too much training on how to use a report, as reports should be self-explanatory and intuitive.

What we need here is a type of Assumptions page, or a page where we can enter all of the variables to run the report. We can use a form for that. The most efficient way to pass variables to a report is through inputs in a form. Let's think about the inputs. To run this report, you need to know the store number or store name, city, state, region, or if you want to run it for the whole company. Then you need to know which dates you want to run it for. Since the report always shows a month and year-to-date numbers, we can have the user pick just one month and one year. The user should be able to pick one year (the current year), and calculate the prior year number. The month should appear in the current and prior month columns, and the year-to-date number will always be from January through the month chosen. So all we need is a form that contains those choices.

Hard-Coded Combo Boxes

When I create such a form, I like to use drop-down menus, or combo boxes, as they are called in Access. A ***combo box*** is simply a drop-down menu that contains the predetermined available choices. Let's create a reporting form where we can choose the parameters for a report and run it.

1. **Save** *and close* **rptInc_Stmt**.
2. **Create** *a blank unbound form (using the* **Blank Form** Blank Form *icon on the* **Create** *tab) and save it as* ***frmReports***.
3. *In* **Design View**, *create a* **label** *at the top of the form that reads* ***REPORTING MENU***.
4. *Format it as* **Times New Roman**, **28 pt**, **Text Black**, **bold**, *and* **italicized** *and position the label as shown in* **Figure 11.21**:

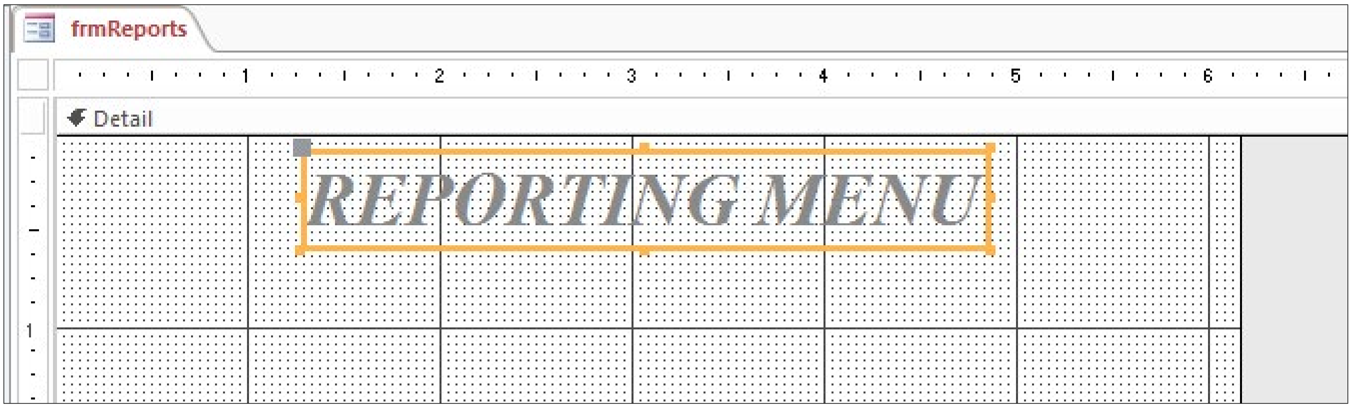
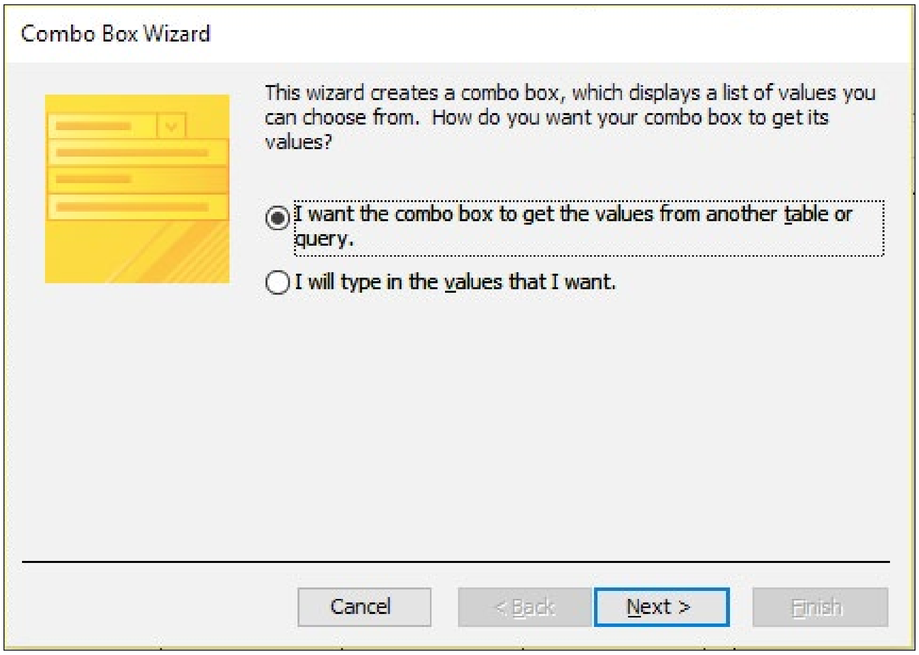


Figure 11.21

Now we'll add the combo boxes. There are two ways to populate data within a Combo Box. You can type in the values you want, or you can look them up in a table or query. We will do both in the following examples. Let's start off with a combo box where you type, or hard-code, the values for the months.

5. *In the* **Design** *tab, click on the* **Combo Box** *icon* .
6. *With your cursor, draw a rectangular box starting at the intersection of the* **1"** *mark on the* **left** *and* **top rulers**, *expanding down* **four grid dots** *and over to the* **2"** *line.*

When you release the mouse, the Combo Box Wizard dialog box appears.



Combo Box Wizard

This wizard creates a combo box, which displays a list of values you can choose from. How do you want your combo box to get its values?

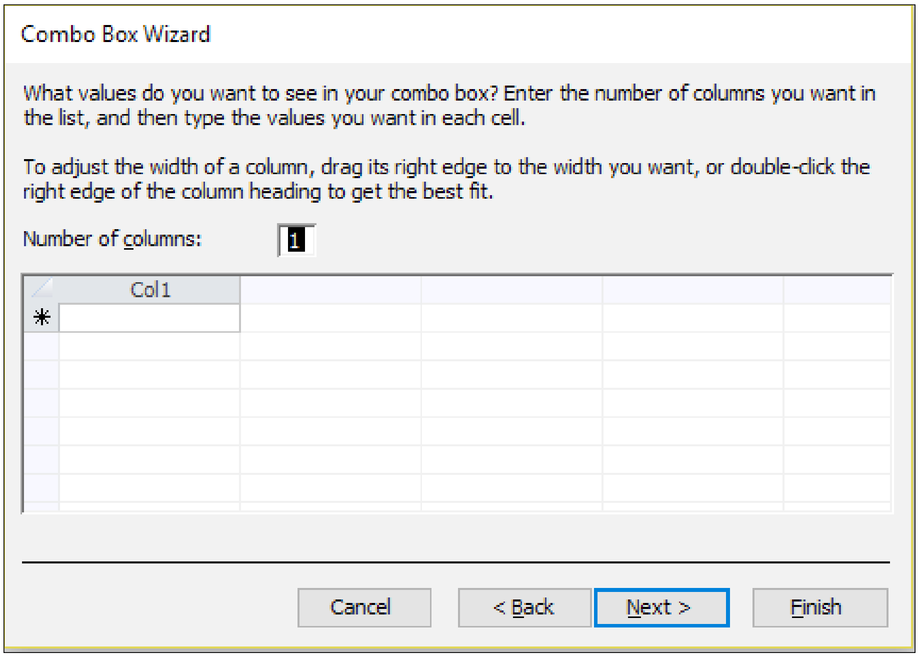
● I want the combo box to get the values from another table or query.

○ I will type in the values that I want.

Figure 11.22

7. *Choose the* **I will type in the values that I want** *option and click* **Next >**.

In the next step of the wizard, you will choose the number of columns you want in the combo box, how wide they should be, and what they should contain.



Combo Box Wizard

What values do you want to see in your combo box? Enter the number of columns you want in the list, and then type the values you want in each cell.

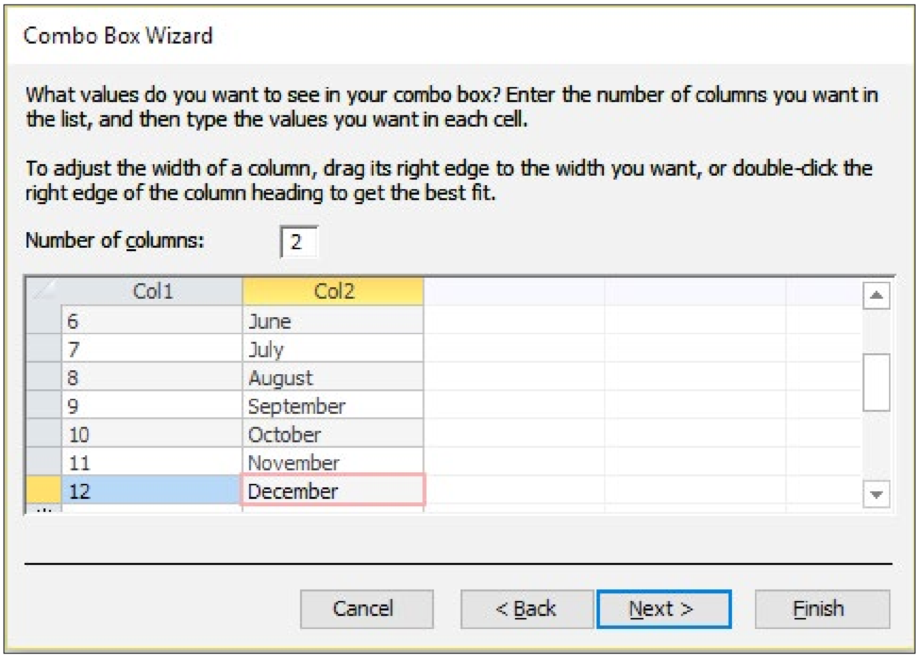
To adjust the width of a column, drag its right edge to the width you want, or double-click the right edge of the column heading to get the best fit.

Number of columns: 1

Col1

Figure 11.23

8. *Type* **2** *in the* **Number of columns** *box and press the* **[Tab]** *key.*
9. *Input the numbers* ***1 – 12*** *in* **Col 1** *and* ***January – December*** *in* **Col 2**.
10. *Adjust the widths of the columns to be similar to* **Figure 10.24**.



Combo Box Wizard

What values do you want to see in your combo box? Enter the number of columns you want in the list, and then type the values you want in each cell.

To adjust the width of a column, drag its right edge to the width you want, or double-click the right edge of the column heading to get the best fit.

Number of columns: 2

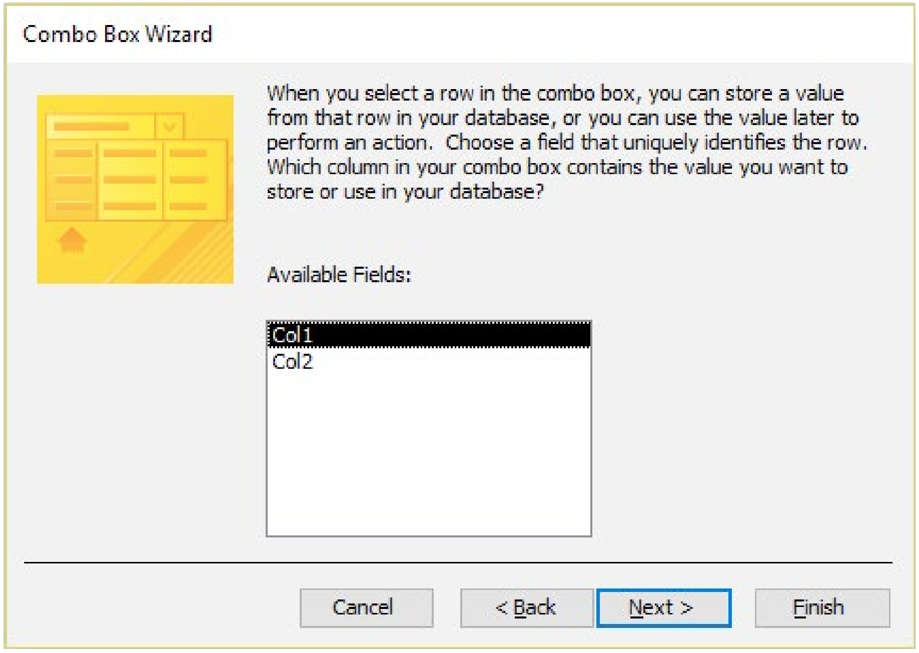
Col1	Col2
6	June
7	July
8	August
9	September
10	October
11	November
12	December

Figure 11.24

11. *Click* **Next >**.

The next step of the wizard asks you to pick a value that will be stored in the database. Since the months in our query are numbers (1 – 12), we'll choose the values in Col 1.

12. *Click on* **Col1**.



Combo Box Wizard

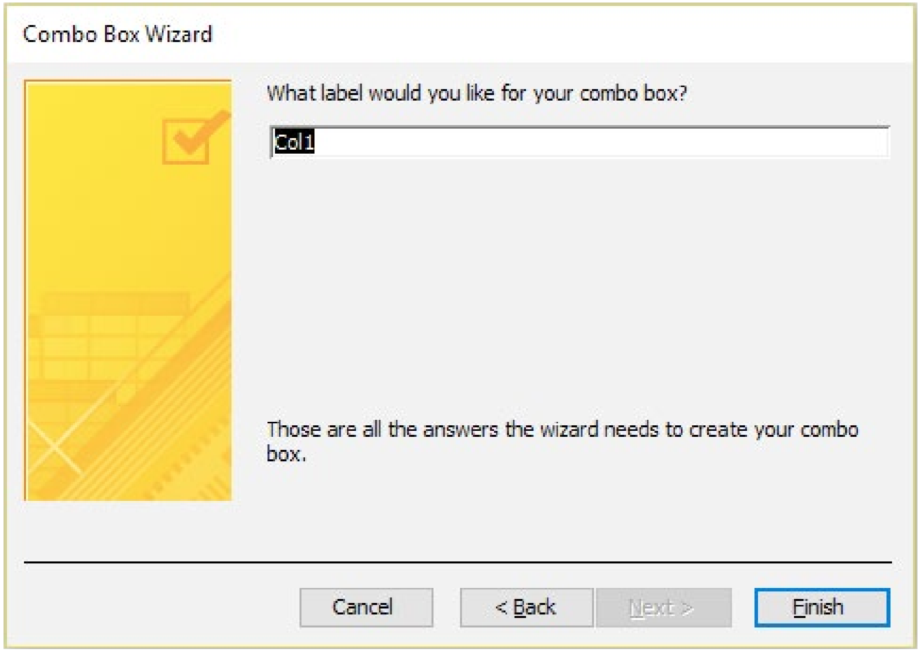
When you select a row in the combo box, you can store a value from that row in your database, or you can use the value later to perform an action. Choose a field that uniquely identifies the row. Which column in your combo box contains the value you want to store or use in your database?

Available Fields:

Col1
Col2

Figure 11.25

13. *Click* **Next >**.



Combo Box Wizard

What label would you like for your combo box?

Col1

Those are all the answers the wizard needs to create your combo box.

Figure 11.26

14. *In the last step of the wizard, change the* **label** *name to* ***Month*** *and click* **Finish**.

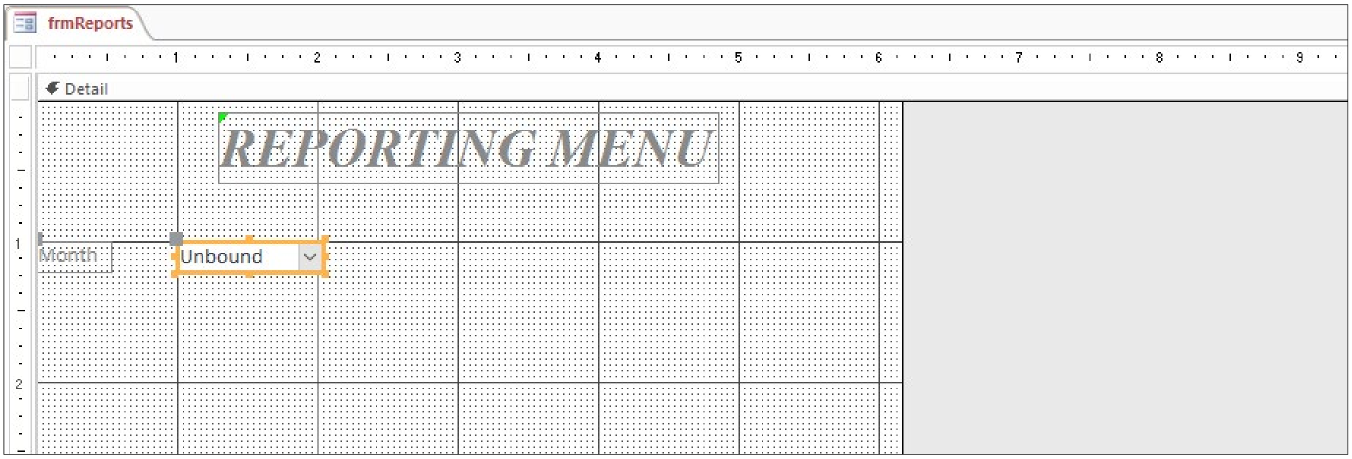
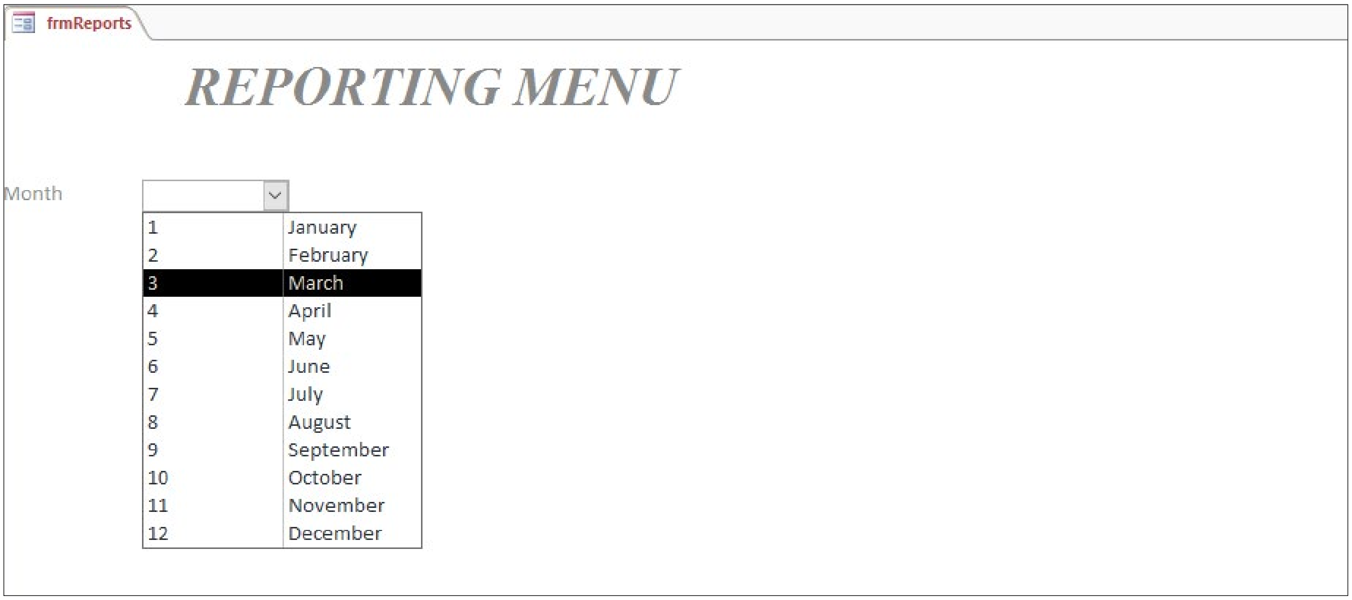


Figure 11.27

There is just one more step we need to take to make programming with combo boxes a little easier. That is to name the Combo Box.

15. *With the* **Month Combo box** *selected, display its* **Properties** *icon and change its name to* ***cmbMonth***.
16. *On the form* **Property Sheet**, *set* **Record Selectors** *and* **Navigation Buttons** *to* **No**.
17. **Save frmReports**.
18. *Run* **frmReports** *in* **Form View**.



1	January
2	February
3	March
4	April
5	May
6	June
7	July
8	August
9	September
10	October
11	November
12	December

Figure 11.28

If you click on the drop-down menu, you should see the month numbers and the name of each month. When you choose one of the months, you should see only the month number. We haven't programmed it to do anything yet, as we don't have any more controls built around it. Let's create another combo box for the Year. In our form, we'll hard-code in the Year. We'll use 2015, 2016, and 2017. Even though we have 2014 data in the table behind it, 2015 is the first year where we can do a year-over-year comparison, so it's not necessary to have 2014 as a choice.

19. *In* **Design View** *of the form, create another* **Combo Box** *below the* **Month** *combo box called* **Year** *populated with the Years* ***2015***, ***2016***, *and* ***2017***.
20. *Name it* ***cmbYear***, *save the form, and* **View** *it.*

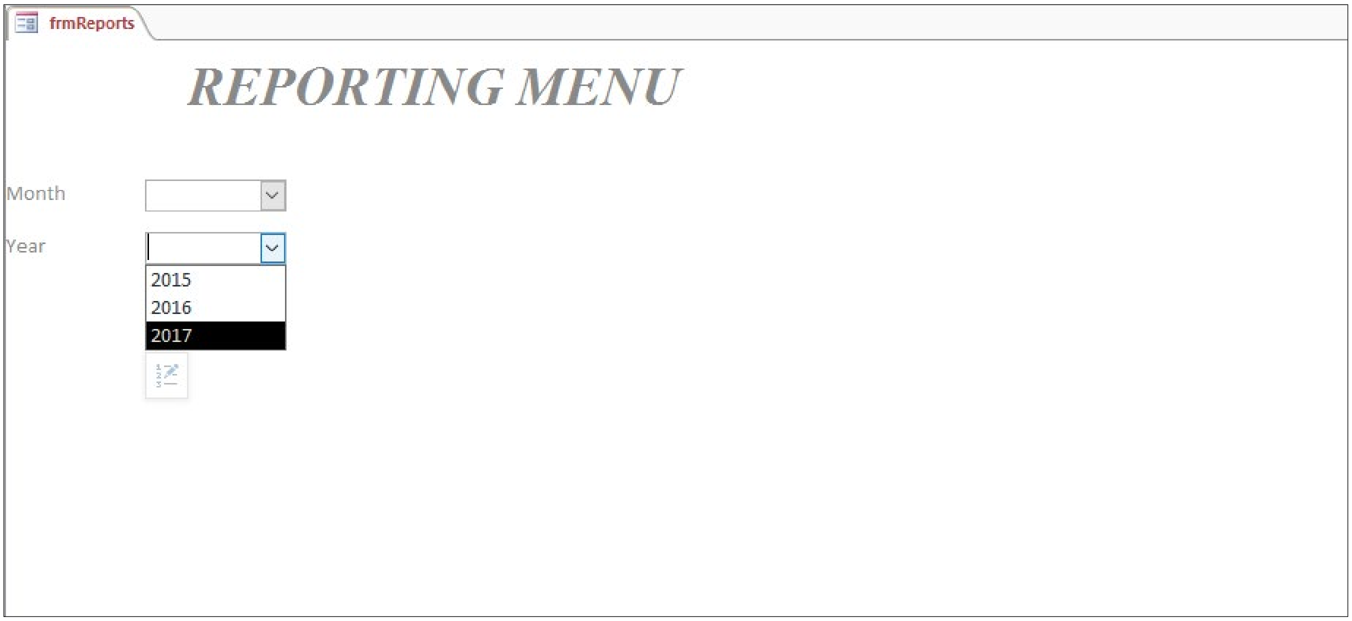
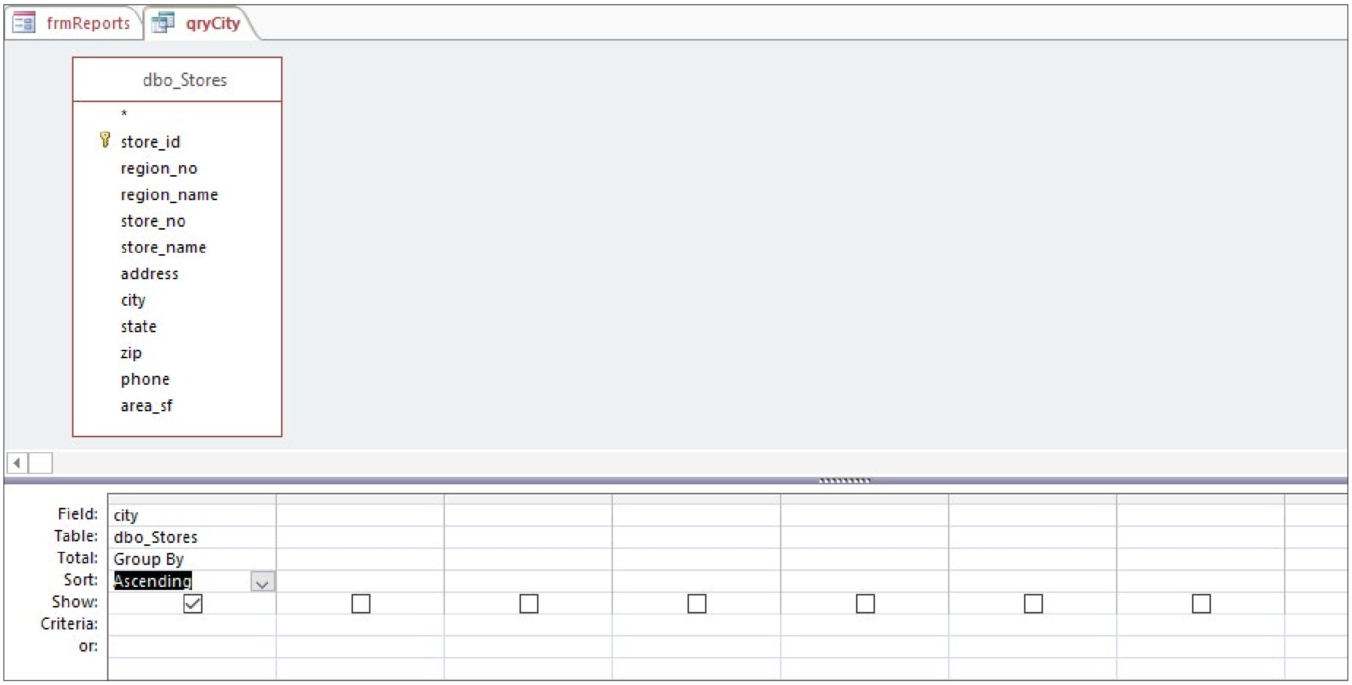


Figure 11.29

Query-Based Combo Boxes

Now that you have the Month and Year combo boxes done, you can create combo boxes for the Store, City, State, and Region. You'll first create a combo box for the City. But before you create the box for the city, you need to think ahead. What if Nitey-Nite opens a store in a new city? If you hard-code the names of all the cities, you'll have to remember to go back and input the name of the city in every combo box we create. That's not very efficient. Instead, since you're a database expert, you can create a query that goes into the Stores table to query the names of all the cities. Once you have that query created, you can open stores in any city, and as long as the data is updated in the Stores table, the name of the city will flow through automatically to the combo box.

1. **Create** *a simple query that lists the unique names of the cities using the* **dbo_Stores** *table as the data source, sorted in* **Ascending** *order.*
2. *Name the query* ***qryCity***.



dbo_Stores
- *
- store_id
- region_no
- region_name
- store_no
- store_name
- address
- city
- state
- zip
- phone
- area_sf

Field:	city						
Table:	dbo_Stores						
Total:	Group By						
Sort:	Ascending						
Show:	☑	☐	☐	☐	☐	☐	☐
Criteria:							
or:							

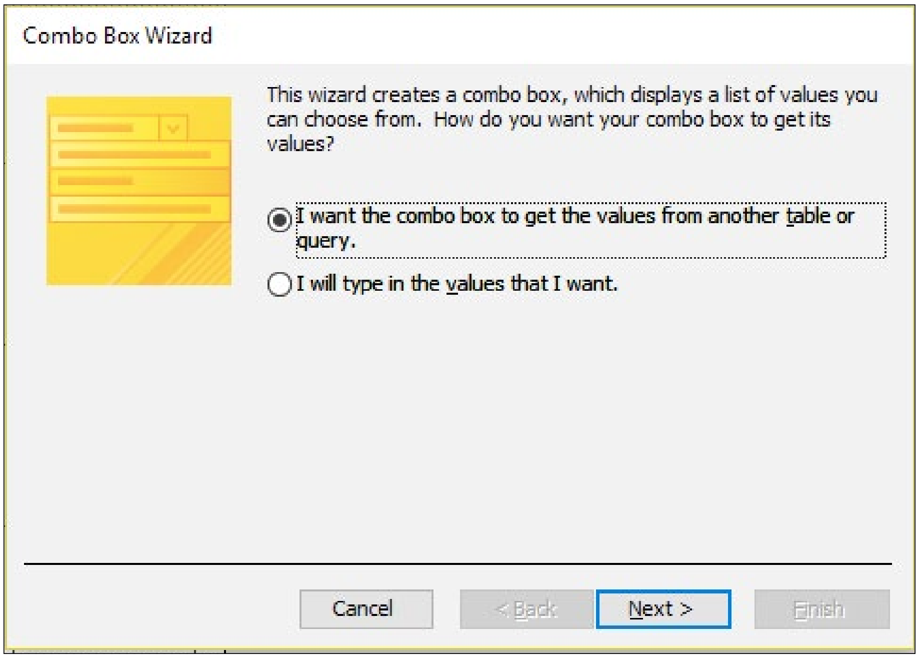
Figure 11.30

Pretty easy, huh? Now you'll create the other queries you need to serve as the data source behind the combo boxes.

3. **Create** *queries that show the names of the* **States** *and* **Regions** *using the* **dbo_Stores** *table as the data source.*
4. *Name the queries* ***qryState*** *and* ***qryRegion***.
5. **Create** *a query that lists the* **store number** *and* **store name** *in one column, with the* **store number** *appearing first then* **store name**, *separated by a* **space**, *a* **dash**, *and a* **space**. *Call that field* ***Store***.
6. *Name that query* ***qryStore***, *then* **save** *and close all the* **Combo boxes** *queries you created.*

Now that you have the queries built, all you have to do is create the combo boxes and tie them to each query. Let's create the Store combo box first, using the wizard.

1. *Go to the* **Design View** *of* **frmReports**.
2. **Create** *a* **Combo box** *and draw it somewhere to the right of the* **Month** *combo box (we'll reposition it later).*
3. *In the first step of the* **Combo Box Wizard**, *choose* **I want the combo box to look up the values in a table or query** *and click* **Next** >.
4. *In the next step of the wizard, click on the* **Queries** *option and choose* **Query: qryStore** *and click* **Next** >.



Combo Box Wizard

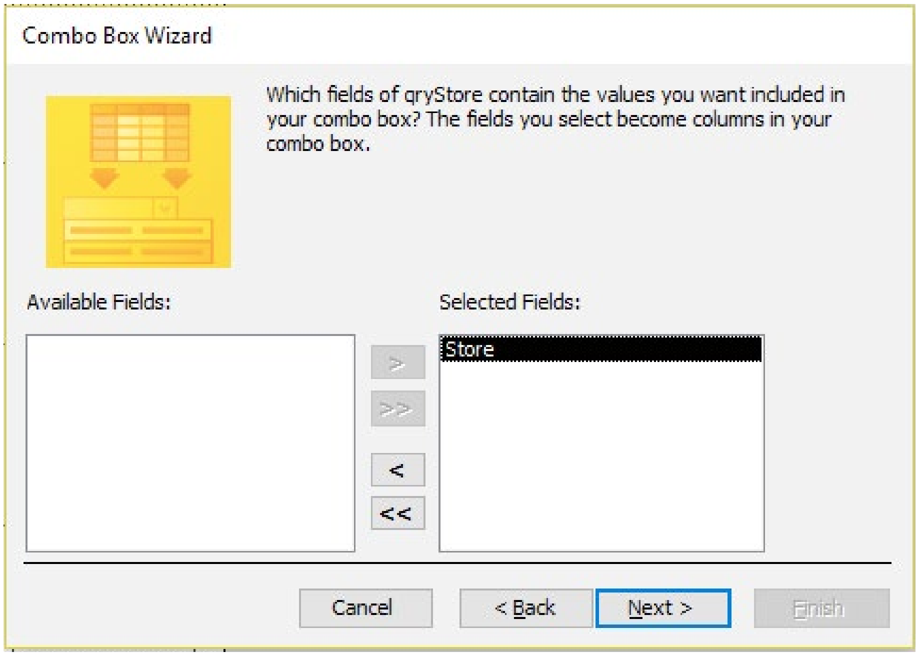
This wizard creates a combo box, which displays a list of values you can choose from. How do you want your combo box to get its values?

● I want the combo box to get the values from another table or query.

○ I will type in the values that I want.

Figure 11.31

5. *In the next box, move the* **Store** *field from the* **Available Fields:** *over to the* **Selected Fields:** *and click* **Next** *>.*



Combo Box Wizard

Which fields of qryStore contain the values you want included in your combo box? The fields you select become columns in your combo box.

Available Fields:

Selected Fields:

Store

Figure 11.32

6. *In the next step, sort on* **Store Ascending** *and click* **Next** *>.*
7. *Adjust the column margins to make sure every store has enough room and click* **Next** *>.*

8. *Make sure the label of the* **Combo Box** *reads* **Store** *and click* **Finish***.*
9. *Name the* **combo box** ***cmbStore****.*
10. *Resize the* **Combo box** *to where each has enough room to display all of the data in it.*
11. *Reposition the* **Combo box** *and corresponding label to appear as in* **Figure 11.33***.*

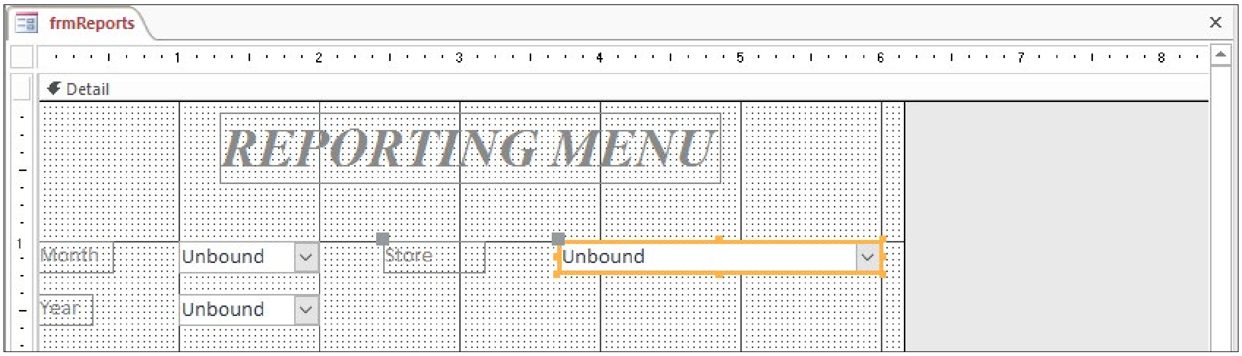


Figure 11.33

If you run the form, you should be able to click on the Store drop-down menu and choose any store. Let's now create combo boxes for the City, State and Region. You should be able to do all of that on your own now.

12. **Create** *combo boxes for the* **City**, **State**, *and* **Region**, *based upon the queries you created and name them like we did the others.*
13. *Design and align the* **combo boxes** *as show in* **Figure 11.34***:*

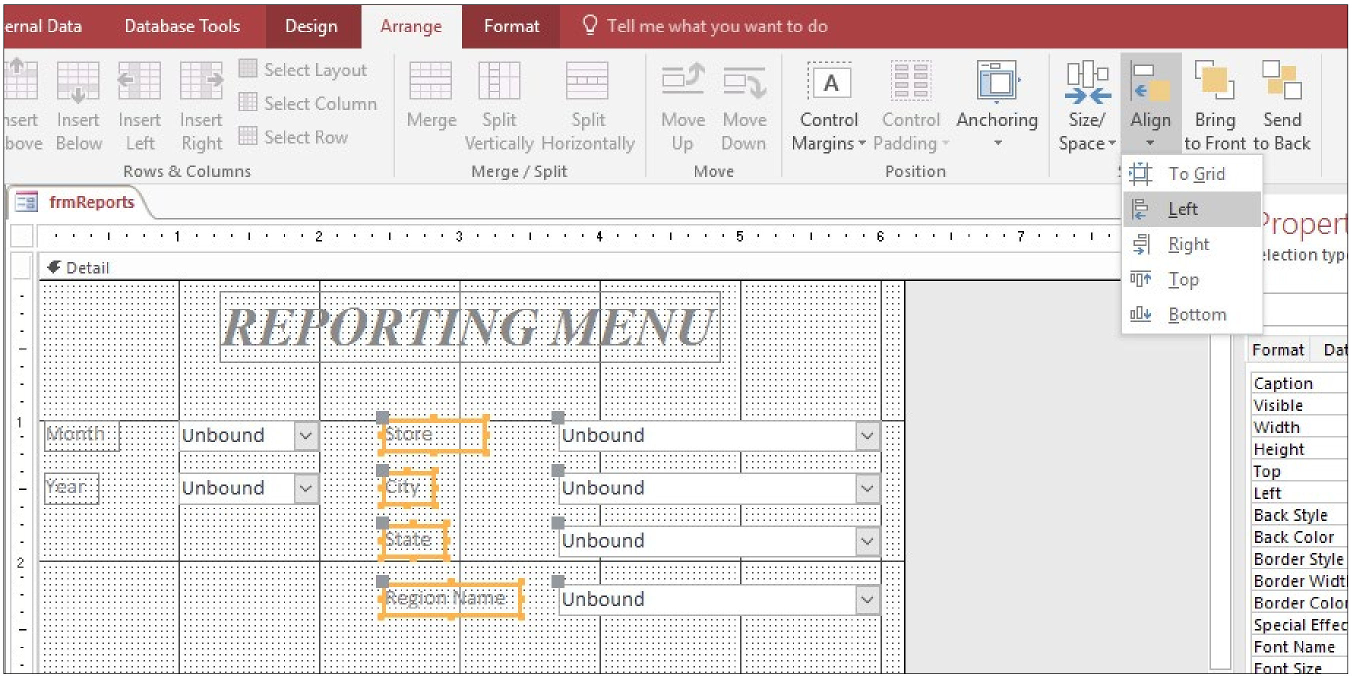


Figure 11.34

> ***Trick**: You can use the **Size/Space** options on the **Arrange** tab to adjust label boxes together for a more uniform look. When moving **Label boxes**, click the larger handle box on the upper-left to avoid moving the **Combo box** as well.*

14. **Save frmReports**, *then display in* **Form View**.

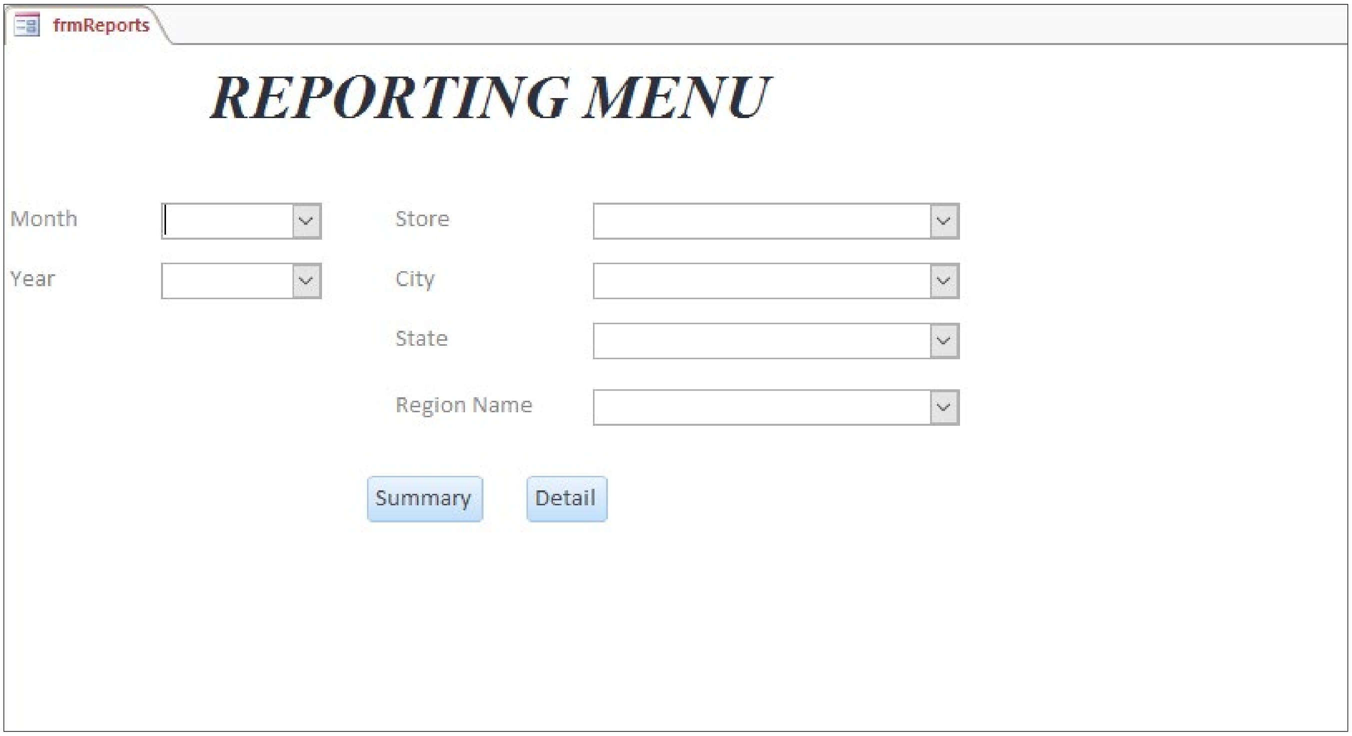


Figure 11.35

> ***Tip**: Make sure your **combo boxes** are named correctly. If you named your combo box <u>labels</u> instead, your queries in **Chapter 12** will not work for the affected boxes.*

15. **Save** *and close* **frmReports**.

Click on each of the combo boxes to make sure that each works like it should.

> ***Review Questions**: It is now time to complete the hands-on Review Questions. Log on to www.ExcelCEO.com with your Email and Password, click on the **Access 2016 and SQL Review Questions, Chapter 11, Section 2 of 2** option and complete the review questions.*

Conclusion

In this chapter, you first built a query based on linked tables that would serve as the basis behind creating a financial statement report. We reviewed the basic account concepts of Gross Margin and Fixed and Variable Expenses. You learned how to get creative in performing sorts on different fields by using a Chart of Accounts table to serve as the sort. You started a report built by the report wizard and extensively modified it to resemble a printed report. Lastly, you created a form, more queries and some combo boxes to allow users to choose the various variables to run the report for. The next chapter is a continuation of this chapter where you will tie the form to a query which is tied to the report.

Chapter Exam

You can now go to www.ExcelCEO.com, click on Sign In, log in and take the exam. Make sure that you take the exam on the same device on which you completed the practice files, as some of the questions on the exam may refer to some of the completed projects. Chapter exams are intended to be hands-on.

Access® 2016 and SQL

Complete Self-study Course

ExcelCEO

Chief Excel Officer

CHAPTER TWELVE — ADVANCED REPORTING (PART II)

In this chapter, you will:

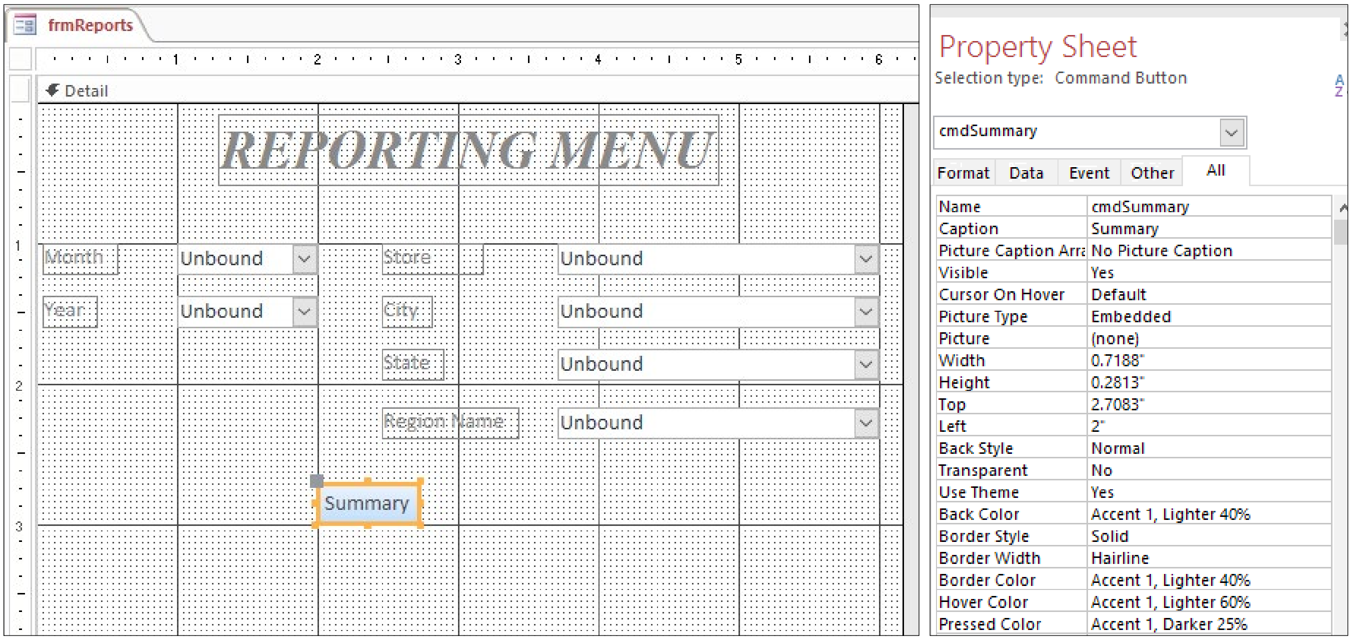
- Determine how to tie variables in a form to a query using Expression Builder
- Identify how to copy the summary report and modify the copy to create a detail report
- Select Database Utilities, including Compact and Repair Database, Performance Analyzer and Documenter, and use them to create and manage reports

CPE Credits possible for this chapter: 3

Advanced Reporting (Part II)

In Chapter 11, you built a very impressive income statement that you could be proud to show to anyone in your organization. But you're really only halfway there. The problem with the report is that all of the variables are hard-coded. You started to build a form that contained the variables that you need to run the report for, but they are not yet coded into the report or query. That is what you will accomplish in this chapter. First though, you need to create a button on the form to run the report. Remember command buttons from the Forms chapters? It should be easy for you. They are the same in Reports as they are in Forms.

1. *Open the* **Inc_Stmt** *database.*
2. *Create a command* **Button** *on* **frmReports** *that previews* **rptInc_Stmt**. *Place it in the center of the* **Reporting Menu** *below the combo boxes.*
3. *Make the text on the command* **Button** *read* ***Summary***.
4. *Name the* **button *cmdSummary***.
5. *Save* **frmReports**.



Property Sheet
Selection type: Command Button

cmdSummary

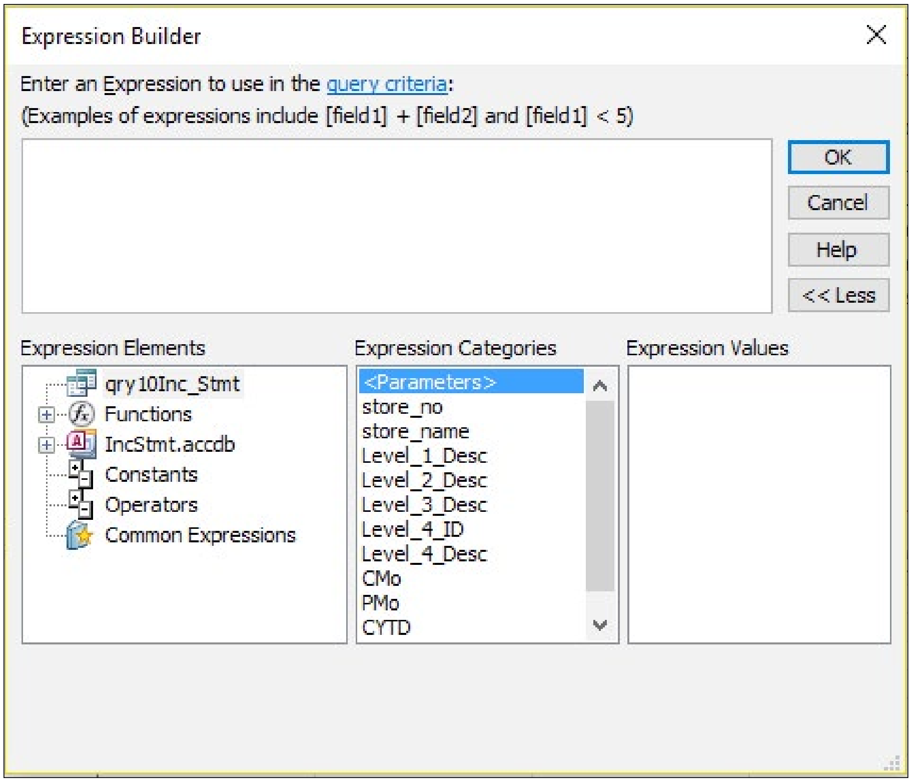
Name	cmdSummary
Caption	Summary
Picture Caption Arra	No Picture Caption
Visible	Yes
Cursor On Hover	Default
Picture Type	Embedded
Picture	(none)
Width	0.7188"
Height	0.2813"
Top	2.7083"
Left	2"
Back Style	Normal
Transparent	No
Use Theme	Yes
Back Color	Accent 1, Lighter 40%
Border Style	Solid
Border Width	Hairline
Border Color	Accent 1, Lighter 40%
Hover Color	Accent 1, Lighter 60%
Pressed Color	Accent 1, Darker 25%

Figure 12.1

6. *In* **Form View**, *choose* **January 2016** *from the* **Month** *and* **Year** *combo boxes and Store* **1005 - Nitey-Nite Glynn** *from the* **Store** *combo box and click on the* **Summary** *command button and open* **Print Preview**.

The Summary command button should run the report just fine, but with the variables changed to January 2016 and Store 1005, the report will still run only for March 2016, Store 1032. Why? Because those are the values you have hard-coded in the query. Now we need to tie the code in the query to the options on the form. Please stay with me on this one, as it is crucially important for you to understand the concept of passing variables from one object to another. All we want to do is to tell the query to use the options chosen on the form instead of the ones we hard-coded. To do that, we have to edit the query and make that change.

7. *Close the* **Print Preview** *of the report.*
8. *Open* **qry10Inc_Stmt** *in* **Design View***.*
9. *Delete the* **"1032"** *criteria under* **Store_No***.*
10. *While on the* **Store_No** *criteria, click the* **Builder** Builder *icon in the* **Query Setup** *group of the* **Design** *tab.*


Expression Builder

Enter an Expression to use in the query criteria:

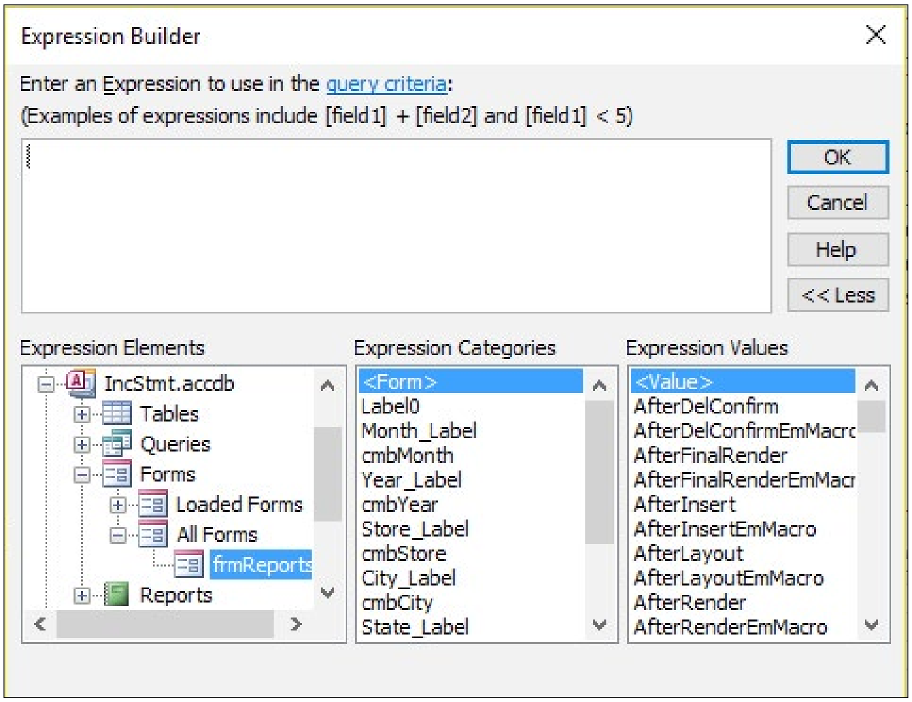
(Examples of expressions include [field1] + [field2] and [field1] < 5)

Figure 12.2

The Expression Builder dialog box appears. You can use this dialog box to help you build a number of expressions or formulas, but I particularly like to use it to help capture data from a form. Here, we'll tie the Store combo box value to the query.

11. *Expand* **Inc_Stmt.accdb** *in the* **Expression Elements** *column, double-click on* **Forms***, then again to expand* **All Forms***, then click on* **frmReports***.*

> ***Note:*** *Expanding on the options by clicking on the "+" sign and double-clicking on it will both open that option.*



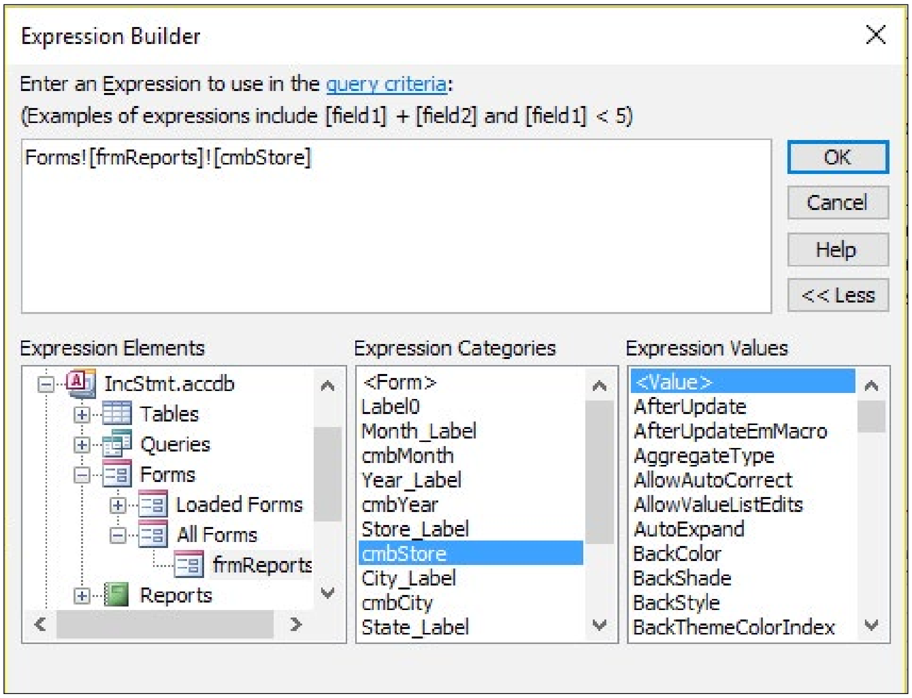
Expression Builder

Enter an Expression to use in the query criteria:
(Examples of expressions include [field1] + [field2] and [field1] < 5)

Figure 12.3

All of the objects in frmReports appear in the middle section under <Form>.

12. *Double-click on* **cmbStore** *in the* **Expression Categories** *section.*



Expression Builder

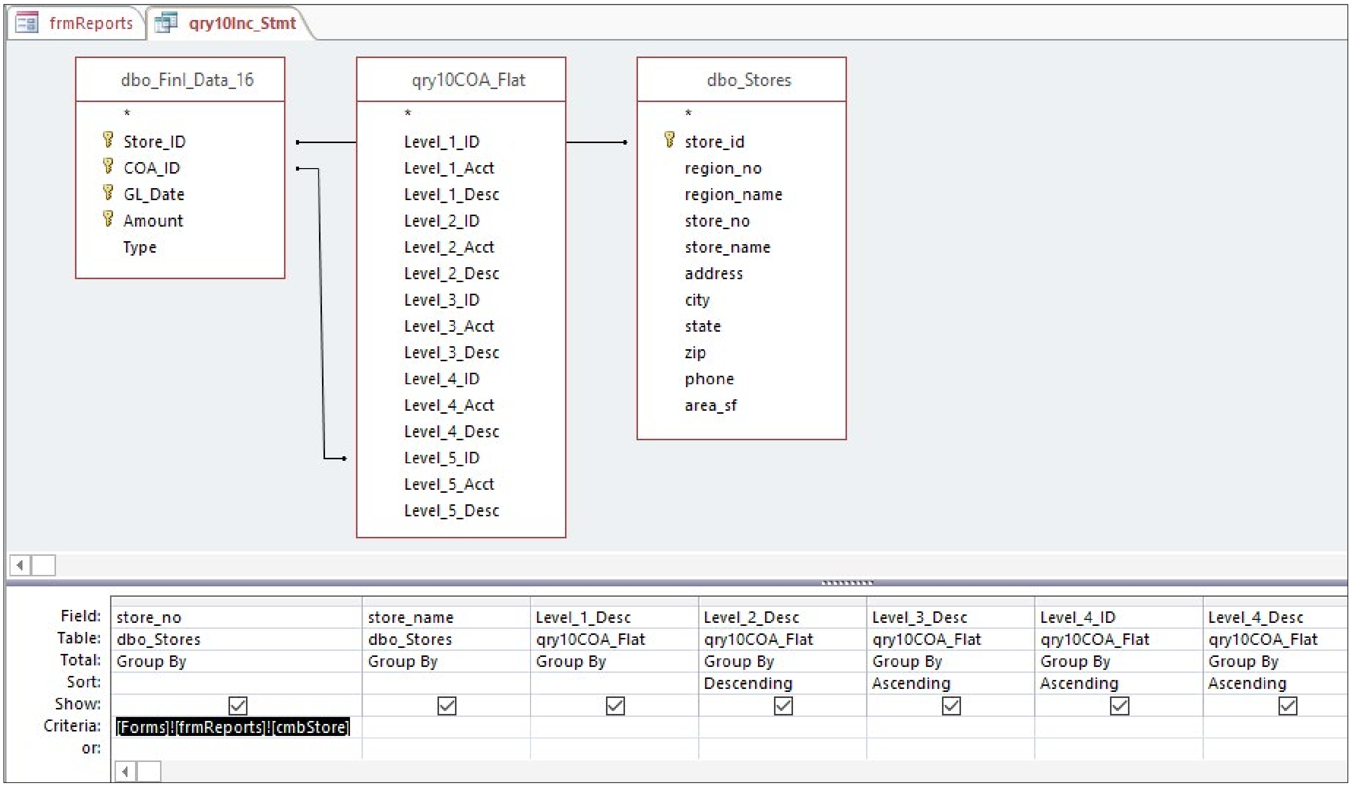
Enter an Expression to use in the query criteria:
(Examples of expressions include [field1] + [field2] and [field1] < 5)

Forms![frmReports]![cmbStore]

Figure 12.4

Just like that, the Expression Builder built a path to the cmbStore value on the form. You can write out the formula (or path to the combo box) if you prefer, but I think it's easier to use the builder. The formula in the Expression Builder tells Access to return the value in a Forms object in frmReports called cmbStore. This is the path to the Store combo box value, whose path is separated by exclamation points. As you have already chosen Store 1005-Nitey-Nite Glynn in the form, it should return that value.

13. *Click* **OK**.



dbo_Finl_Data_16: *, Store_ID, COA_ID, GL_Date, Amount, Type

qry10COA_Flat: *, Level_1_ID, Level_1_Acct, Level_1_Desc, Level_2_ID, Level_2_Acct, Level_2_Desc, Level_3_ID, Level_3_Acct, Level_3_Desc, Level_4_ID, Level_4_Acct, Level_4_Desc, Level_5_ID, Level_5_Acct, Level_5_Desc

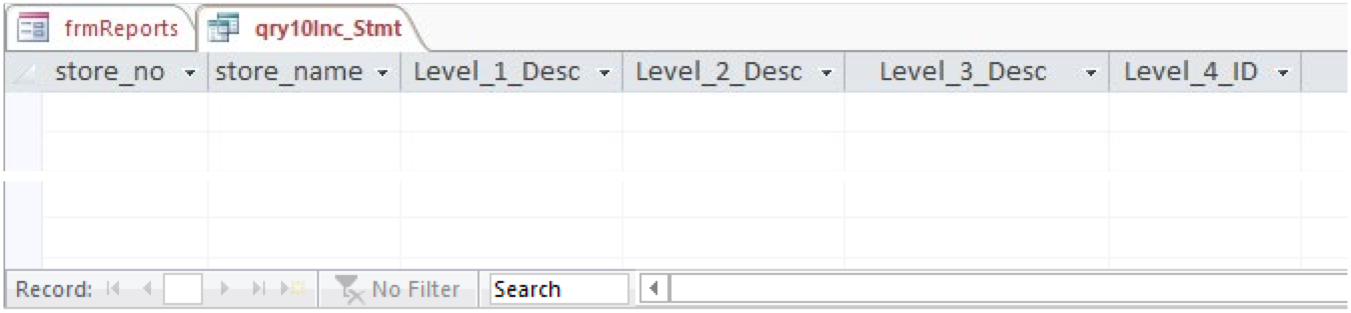
dbo_Stores: *, store_id, region_no, region_name, store_no, store_name, address, city, state, zip, phone, area_sf

Field:	store_no	store_name	Level_1_Desc	Level_2_Desc	Level_3_Desc	Level_4_ID	Level_4_Desc
Table:	dbo_Stores	dbo_Stores	qry10COA_Flat	qry10COA_Flat	qry10COA_Flat	qry10COA_Flat	qry10COA_Flat
Total:	Group By	Group By	Group By	Group By	Group By	Group By	Group By
Sort:				Descending	Ascending	Ascending	Ascending
Show:	☑	☑	☑	☑	☑	☑	☑
Criteria:	[Forms]![frmReports]![cmbStore]						
or:							

Figure 12.5

The builder inserts the formula into the Criteria section of the Store_No field.

14. **Run** *the query.*



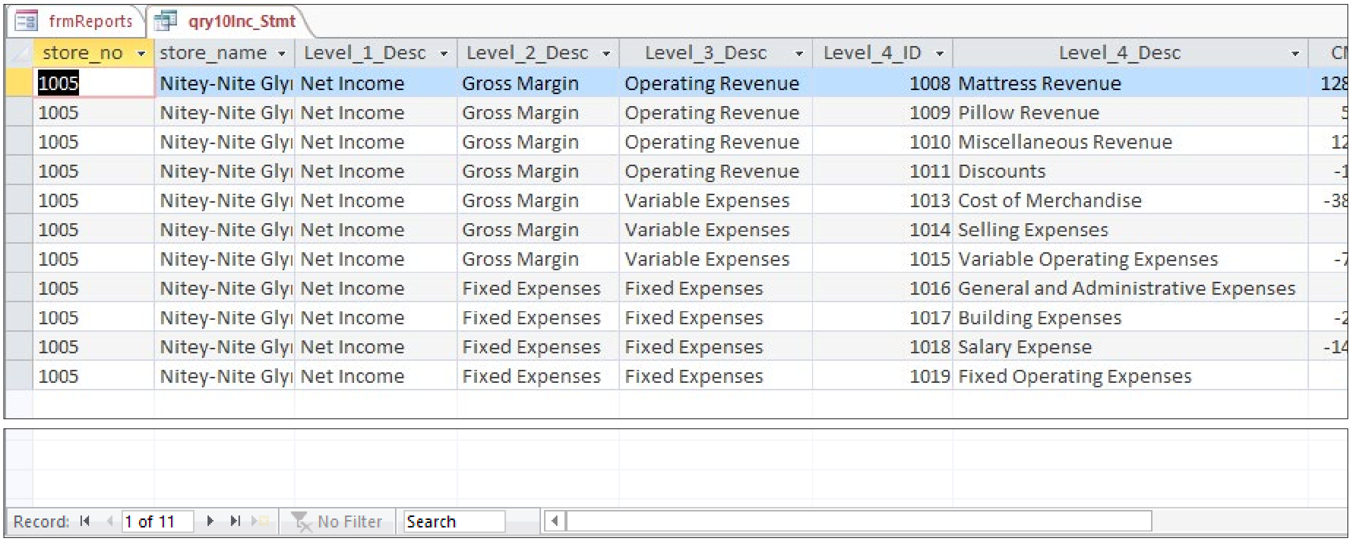
store_no	store_name	Level_1_Desc	Level_2_Desc	Level_3_Desc	Level_4_ID

Figure 12.6

Huh?!? If you did it right, it will return no records. That is because the value in the combo box is "1005-Nitey-Nite Glynn", and the Store_No field contains only the numbers, not the names. This is easy to fix – just choose the left 4 characters in the Store combo box. This should be easy for you since you are a formula-writing guru.

15. *In the* **Design View** *of the query, edit the formula in the* **Store_No** *criteria field to read as follows:* ***Left([Forms]![frmReports]![cmbStore],4)***
16. **Run** *the query.*

> ***Note****: If it still returns no records, you may have to save the form, then run the query.*



frmReports | qry10Inc_Stmt

store_no	store_name	Level_1_Desc	Level_2_Desc	Level_3_Desc	Level_4_ID	Level_4_Desc	Cl
1005	Nitey-Nite Gly	Net Income	Gross Margin	Operating Revenue	1008	Mattress Revenue	128
1005	Nitey-Nite Gly	Net Income	Gross Margin	Operating Revenue	1009	Pillow Revenue	5
1005	Nitey-Nite Gly	Net Income	Gross Margin	Operating Revenue	1010	Miscellaneous Revenue	12
1005	Nitey-Nite Gly	Net Income	Gross Margin	Operating Revenue	1011	Discounts	-1
1005	Nitey-Nite Gly	Net Income	Gross Margin	Variable Expenses	1013	Cost of Merchandise	-38
1005	Nitey-Nite Gly	Net Income	Gross Margin	Variable Expenses	1014	Selling Expenses	
1005	Nitey-Nite Gly	Net Income	Gross Margin	Variable Expenses	1015	Variable Operating Expenses	-7
1005	Nitey-Nite Gly	Net Income	Fixed Expenses	Fixed Expenses	1016	General and Administrative Expenses	
1005	Nitey-Nite Gly	Net Income	Fixed Expenses	Fixed Expenses	1017	Building Expenses	-2
1005	Nitey-Nite Gly	Net Income	Fixed Expenses	Fixed Expenses	1018	Salary Expense	-14
1005	Nitey-Nite Gly	Net Income	Fixed Expenses	Fixed Expenses	1019	Fixed Operating Expenses	

Figure 12.7

NOW it's working. If you look to the amount fields, you will see that it is returning the data for the hard-coded years and months. Let's fix that. We'll work with the Year first.

17. *In the* **Design View** *of the* **qry10Inc_Stmt** *query, use the builder to choose the* **Year** *from the* **cmbYear** *combo box, replacing the* **2016** *reference in the* **Year** *criteria.*
18. *Replace* **2015** *with the same expression for* **year**, **minus one**.

The formula you have in the Year criteria should look like the following:

[Forms]![frmReports]![cmbYear] Or [Forms]![frmReports]![cmbYear]-1

19. *Click* **OK** *and* **save qry10Inc_Stmt**.

Let's modify the Month criteria now.

20. *In the* **Design View** *of the query, use the builder to choose the* **Month** *from the* **cmbMonth** *combo box, replacing ONLY the* **3** *reference in the* **Month** *criteria.*

The formula you have in the Month criteria should look like this:

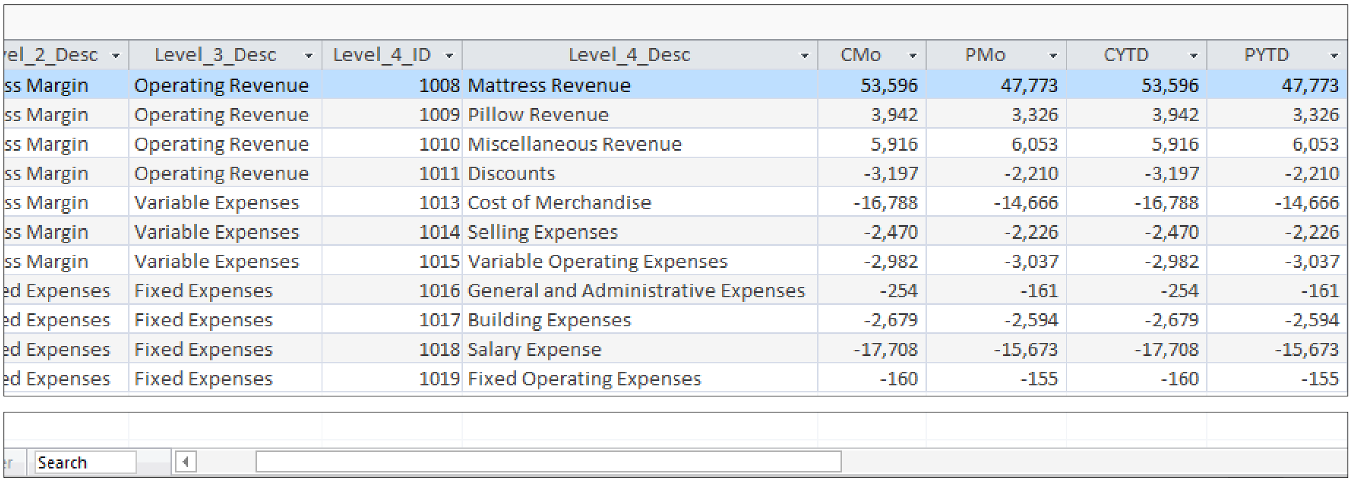
Between 1 And Forms![frmReports]![cmbMonth]

We have to treat the month field a little different, because in the YTD fields we will always choose from January (Month 1) through the month chosen.

The only thing left to do is to modify the Year and Month references in each of the CMo, PMo, CYTD and PYTD formulas. You should have the concept down now, so you should be able to modify those formulas on your own. Try it on your own, but I'll show you the correct formulas just in case you need a hint or two.

- *CMo: Sum(IIF(Month([gl_date])=[Forms]![frmReports]![cmbMonth] And Year([gl_date])=[Forms]![frmReports]![cmbYear],-[amount],0))*
- *PMo: Sum(IIF(Month([gl_date])=[Forms]![frmReports]![cmbMonth] And Year([gl_date])=[Forms]![frmReports]![cmbYear]-1,-[amount],0))*
- *CYTD: Sum(IIF(Month([gl_date]) Between 1 And [Forms]![frmReports]![cmbMonth] And Year([gl_date])=[Forms]![frmReports]![cmbYear],-[amount],0))*
- *PYTD: Sum(IIF(Month([gl_date]) Between 1 And [Forms]![frmReports]![cmbMonth] And Year([gl_date])=[Forms]![frmReports]![cmbYear]-1,-[amount],0))*

21. **Run** *the query.*



el_2_Desc	Level_3_Desc	Level_4_ID	Level_4_Desc	CMo	PMo	CYTD	PYTD
ss Margin	Operating Revenue	1008	Mattress Revenue	53,596	47,773	53,596	47,773
ss Margin	Operating Revenue	1009	Pillow Revenue	3,942	3,326	3,942	3,326
ss Margin	Operating Revenue	1010	Miscellaneous Revenue	5,916	6,053	5,916	6,053
ss Margin	Operating Revenue	1011	Discounts	-3,197	-2,210	-3,197	-2,210
ss Margin	Variable Expenses	1013	Cost of Merchandise	-16,788	-14,666	-16,788	-14,666
ss Margin	Variable Expenses	1014	Selling Expenses	-2,470	-2,226	-2,470	-2,226
ss Margin	Variable Expenses	1015	Variable Operating Expenses	-2,982	-3,037	-2,982	-3,037
ed Expenses	Fixed Expenses	1016	General and Administrative Expenses	-254	-161	-254	-161
ed Expenses	Fixed Expenses	1017	Building Expenses	-2,679	-2,594	-2,679	-2,594
ed Expenses	Fixed Expenses	1018	Salary Expense	-17,708	-15,673	-17,708	-15,673
ed Expenses	Fixed Expenses	1019	Fixed Operating Expenses	-160	-155	-160	-155

Figure 12.8

Please understand that this is complex programming. I don't expect you to get all of this immediately, but at least you've been exposed to it, and can come back to it whenever you need to. Notice that the CMo and CYTD fields (and the PMo and PYTD fields) return the same amounts. That is because we're running the report just for January.

22. **Save** *and close* **qry10Inc_Stmt**.

Now you can run the Summary income statement for any store and for any time period. However, notice that the report labels still show that the report is for March 2016, Store 1032. Again, that is because the labels are hard-coded with that date and store. Next, we'll create text boxes in the report to bring in the date and store name chosen on the form.

1. *In the* **Design View** *of the* **rptInc_Stmt**, *delete the* **Summary Income Statement, March 2016** *label and the* **For Store 1032 – Nitey-Nite Pease** *label.*
2. **Copy** *the* **Level_2_Desc** *text box to the* **Page Header** *section and replace the field with this formula: =****"Summary Income Statement, " & MonthName(Forms!frmReports!cmbMonth) & " " & Forms!frmReports!cmbYear***
3. **Format** *it as* **Times New Roman**, **14 pt**, **bold**, *and* **italicized**.
4. *Change the* **Back Style** *to* **Transparent** *and the* **Fore Color** *to* **Text Light**.
5. *Make sure it is wide enough to display the results.*

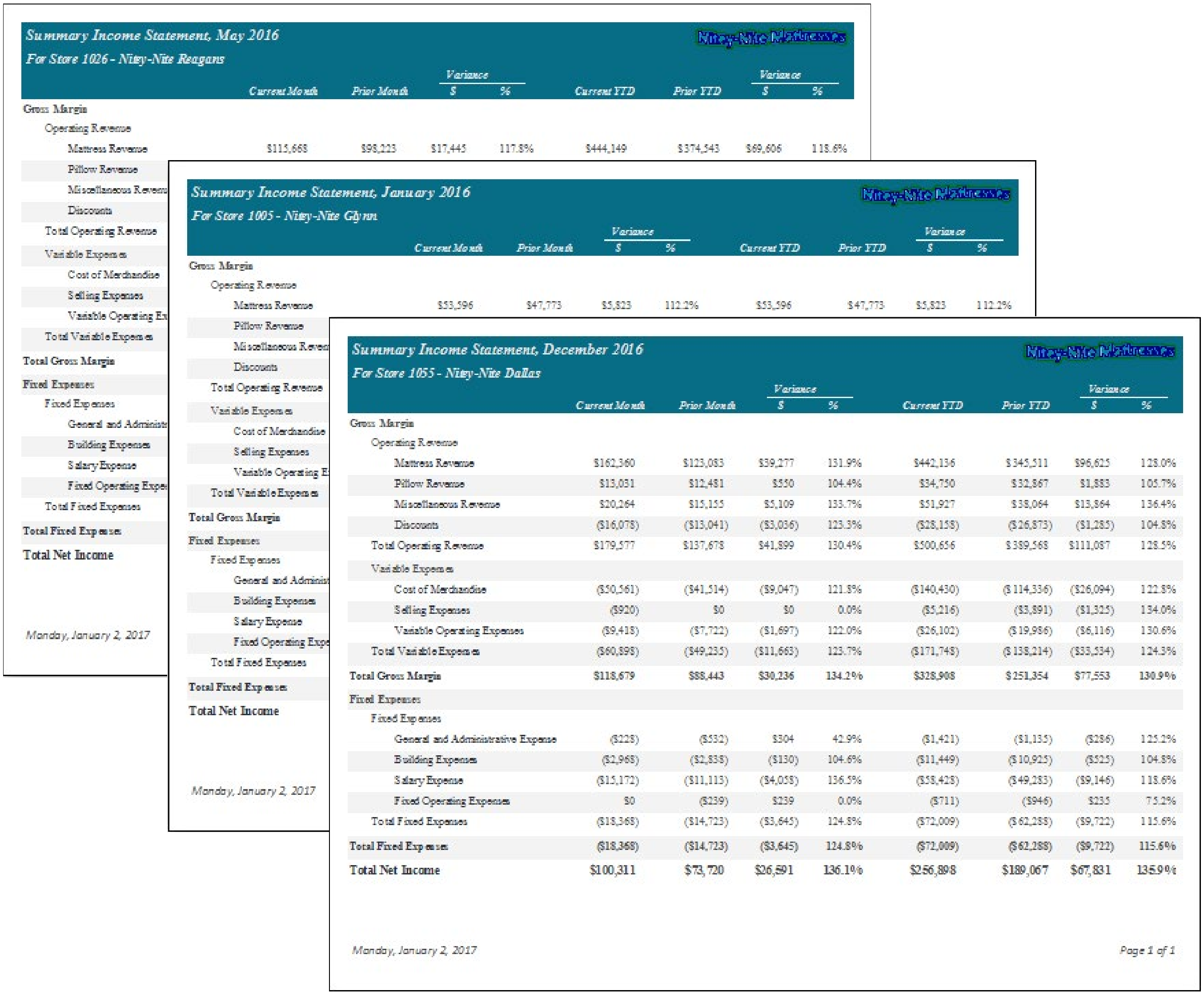
> ***Tip****: You could also right-click on the* **Label** *box and click* **Change to** *then Text box from the menu that appears. After that, you can type the formula and the text formatting should stay intact. Changing the box type can reset the border properties though.*

This formula will display the text "Summary Income Statement" followed by the name of the month and the year as chosen in the form.

6. **Copy** *the text box you just created and place the copy below the first* **Text box**.
7. *Modify the copied* **Text box***'s formula to be: =****"For Store " &[Forms]![frmReports]![cmbStore]***
8. **Format** *it as* **Times New Roman**, **12 pt**, **bold**, *and* **italicized**.
9. **Save rptInc_Stmt**.

This formula simply brings in the text "For Store" followed by the store number and name chosen in the combo box. Now you have a form and a report where you can choose any store for any time period, and the report prints out the choices you made. These choices are called variables.

10. **Run** *the report from* **frmReports** *to make sure everything is working correctly. This is a good time to make sure your text boxes are wide enough, including labels.*



Summary Income Statement, May 2016

For Store 1026 - Nitey-Nite Reagans

	Current Month	Prior Month	Variance $	Variance %	Current YTD	Prior YTD	Variance $	Variance %
Gross Margin								
Operating Revenue								
Mattress Revenue	$115,668	$98,223	$17,445	117.8%	$444,149	$374,543	$69,606	118.6%
Pillow Revenue								
Miscellaneous Revenu								
Discounts								
Total Operating Revenue								
Variable Expenses								
Cost of Merchandise								
Selling Expenses								
Variable Operating Ex								
Total Variable Expenses								
Total Gross Margin								
Fixed Expenses								
Fixed Expenses								
General and Administ								
Building Expenses								
Salary Expense								
Fixed Operating Expe								
Total Fixed Expenses								
Total Fixed Expenses								
Total Net Income								

Monday, January 2, 2017

Summary Income Statement, January 2016

For Store 1005 - Nitey-Nite Glynn

	Current Month	Prior Month	Variance $	Variance %	Current YTD	Prior YTD	Variance $	Variance %
Gross Margin								
Operating Revenue								
Mattress Revenue	$53,596	$47,773	$5,823	112.2%	$53,596	$47,773	$5,823	112.2%
Pillow Revenue								
Miscellaneous Reven								
Discounts								
Total Operating Revenue								
Variable Expenses								
Cost of Merchandise								
Selling Expenses								
Variable Operating E								
Total Variable Expenses								
Total Gross Margin								
Fixed Expenses								
Fixed Expenses								
General and Administ								
Building Expenses								
Salary Expense								
Fixed Operating Expe								
Total Fixed Expenses								
Total Fixed Expenses								
Total Net Income								

Monday, January 2, 2017

Summary Income Statement, December 2016

For Store 1055 - Nitey-Nite Dallas

	Current Month	Prior Month	Variance $	Variance %	Current YTD	Prior YTD	Variance $	Variance %
Gross Margin								
Operating Revenue								
Mattress Revenue	$162,360	$123,083	$39,277	131.9%	$442,136	$345,511	$96,625	128.0%
Pillow Revenue	$13,031	$12,481	$550	104.4%	$34,750	$32,867	$1,883	105.7%
Miscellaneous Revenue	$20,264	$15,155	$5,109	133.7%	$51,927	$38,064	$13,864	136.4%
Discounts	($16,078)	($13,041)	($3,036)	123.3%	($28,158)	($26,873)	($1,285)	104.8%
Total Operating Revenue	$179,577	$137,678	$41,899	130.4%	$500,656	$389,568	$111,087	128.5%
Variable Expenses								
Cost of Merchandise	($50,561)	($41,514)	($9,047)	121.8%	($140,430)	($114,336)	($26,094)	122.8%
Selling Expenses	($920)	$0	$0	0.0%	($5,216)	($3,891)	($1,325)	134.0%
Variable Operating Expenses	($9,418)	($7,722)	($1,697)	122.0%	($26,102)	($19,986)	($6,116)	130.6%
Total Variable Expenses	($60,898)	($49,235)	($11,663)	123.7%	($171,748)	($138,214)	($33,534)	124.3%
Total Gross Margin	$118,679	$88,443	$30,236	134.2%	$328,908	$251,354	$77,553	130.9%
Fixed Expenses								
Fixed Expenses								
General and Administrative Expense	($228)	($532)	$304	42.9%	($1,421)	($1,135)	($286)	125.2%
Building Expenses	($2,968)	($2,838)	($130)	104.6%	($11,449)	($10,925)	($525)	104.8%
Salary Expense	($15,172)	($11,113)	($4,058)	136.5%	($58,428)	($49,283)	($9,146)	118.6%
Fixed Operating Expenses	$0	($239)	$239	0.0%	($711)	($946)	$235	75.2%
Total Fixed Expenses	($18,368)	($14,723)	($3,645)	124.8%	($72,009)	($62,288)	($9,722)	115.6%
Total Fixed Expenses	($18,368)	($14,723)	($3,645)	124.8%	($72,009)	($62,288)	($9,722)	115.6%
Total Net Income	$100,311	$73,720	$26,591	136.1%	$256,898	$189,067	$67,831	135.9%

Monday, January 2, 2017 *Page 1 of 1*

Figure 12.9

11. **Save** *and close* **rptInc_Stmt**.

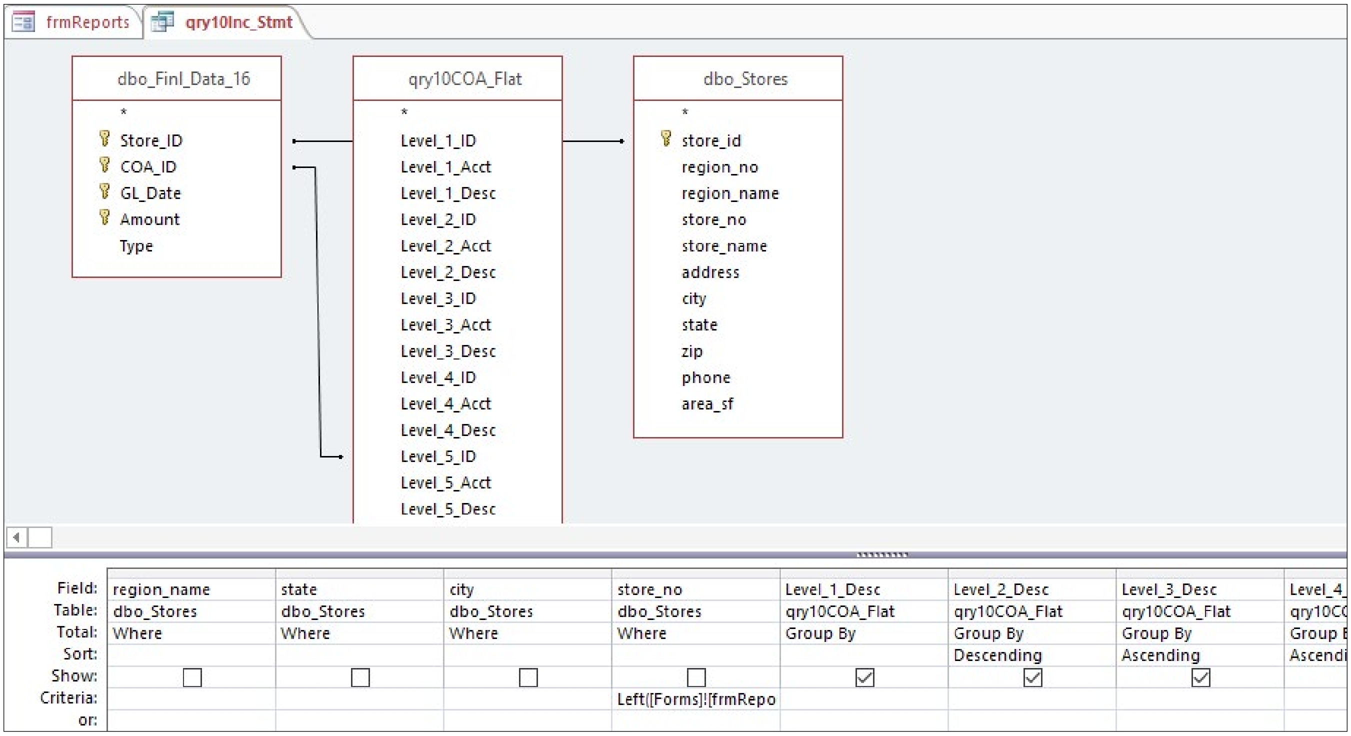
You show this database to your manager who absolutely loves it. He shows it to the Regional President of the Northern Region says, "*What if I want to see a rollup of all of the stores in my region? Also, I need to see the Detail statement, so I can look at all of the individual accounts. Can you do that?*" Your answer: "*Does a one-legged duck swim in circles? Of course I can!*" Then you start to work on it.

Since this entire report is based on the data from one query, let's look at the query and think about what we have to display.

1. *Open* **qry10Inc_Stmt** *in* **Design View**.

If the regional president wanted to see his region rolled up, you may as well plan on management asking for other levels of rollup, like at the City and State levels. Fortunately, you have the insight to plan for that – that is why you created the queries and combo boxes in the form. But before we start taking things out and adding things in to the query, let's think about it. If you simply brought the Region Name field into the query and ran it, all of the stores' data would appear. You would have to take out the Store_No and Store_Name fields and then run the query. The same problem exists if you want to run reports for the City or State. One answer is to bring all of those fields into the query and change the Group By option to Where. You could then write an IIF() statement in the report to detect whether the choice was a Store, City, State, Region, or the entire company, but I really don't like writing IIF() statements in a report. Rather, I think we'll create a field in the query that will give us the area chosen, and just refer to that field in the report. Let's do that.

2. *In* **Design View** *of* **qry10Inc_Stmt**, *change the* **Store_No** *field to have* **Where** *on the* **Total** *line instead of* **Group By**.
3. **Delete** *the* **Store_Name** *field (as you won't need it).*
4. *Bring in* **Region_Name**, **State**, *and* **City** *fields to the left of* **Store_No** *and make them all have* **Where** *clauses on the* **Total** *line.*



dbo_Finl_Data_16
*
Store_ID
COA_ID
GL_Date
Amount
Type

qry10COA_Flat
*
Level_1_ID
Level_1_Acct
Level_1_Desc
Level_2_ID
Level_2_Acct
Level_2_Desc
Level_3_ID
Level_3_Acct
Level_3_Desc
Level_4_ID
Level_4_Acct
Level_4_Desc
Level_5_ID
Level_5_Acct
Level_5_Desc

dbo_Stores
*
store_id
region_no
region_name
store_no
store_name
address
city
state
zip
phone
area_sf

Field:	region_name	state	city	store_no	Level_1_Desc	Level_2_Desc	Level_3_Desc	Level_4_
Table:	dbo_Stores	dbo_Stores	dbo_Stores	dbo_Stores	qry10COA_Flat	qry10COA_Flat	qry10COA_Flat	qry10C
Total:	Where	Where	Where	Where	Group By	Group By	Group By	Group E
Sort:						Descending	Ascending	Ascendi
Show:	□	□	□	□	☑	☑	☑	
Criteria:				Left([Forms]![frmRepo				
or:								

Figure 12.10

Now you will create a field that will display the report's selected area (Store, City, State, Region or entire Company). In this exercise, we'll create a field that analyzes the inputs from the form and then display the area the report is analyzing.

5. **Create** *a new field at the end of the* **Design Grid** *with an alias called* ***Report_Name***.

6. *Write a formula where if the* **Store** *input from the form* **is not null**, *then display that* **Store** *input. Else, if the* **City is not null**, *display that input.*
7. *Continue likewise for the* **State** *and if the* **Region is not null**, *display the* **Region**, *else write* ***Total Company***.

The formula I wrote looks like this:

```
Report_Name: IIF([Forms]![frmReports]![cmbStore]<>””,[Forms]![frmReports]![cmbStore],IIF(
[Forms]![frmReports]![cmbCity]<>””,[Forms]![frmReports]![cmbCity],IIF([Forms]![frmReports]
![cmbState]<>””,“State of ”&[Forms]![frmReports]![cmbState],IIF([Forms]![frmReports]!
[cmbRegion]<>””,[Forms]![frmReports]![cmbRegion],”Total Company”))))
```

Next, you need to write some logic in the criteria section of each area (region_name, state, city, and store_no) that says if the corresponding combo box is null, return all of the records, otherwise return the chosen value.

8. *Write formulas in each of the* **Store_No**, **City**, **State**, *and* **Region** *fields that says if the corresponding combo box is null, return all of the records, otherwise return the chosen value.*

This part can be tricky and, if you don't know the code, you will get very frustrated. Think through the logic of the formula first, and try to do it on your own. If you get stuck (which you probably will), try this code in the Criteria section for the City:

```
Like IIF([Forms]![frmReports]![cmbCity] Is Null,”*”,[Forms]![frmReports]![cmbCity])
```

Remember, the Store combo box from the form has the store number concatenated with the store name, so you have to choose the left four characters. You'll have to delete the existing criteria in the Store_No field.

```
Like IIF([Forms]![frmReports]![cmbStore] Is Null,”*”,Left([Forms]![frmReports]![cmbStore],4))
```

The State and Region criteria will be exactly like the one for the City, except replace City with State or Region. This is a very useful piece of code. Keep it somewhere accessible for if and when you need to refer to it again.

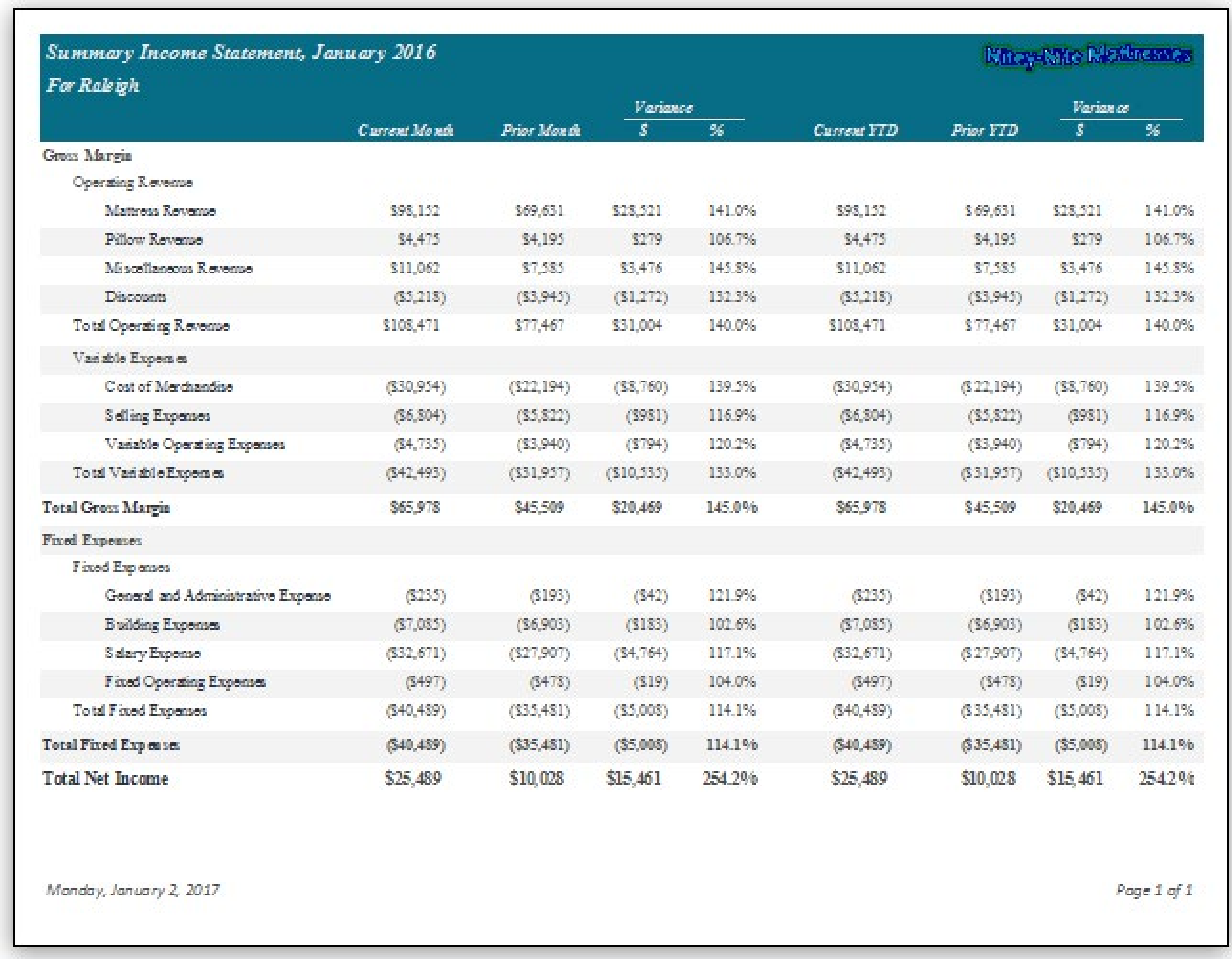
Next, you need to replace the formula in the text box in the report that contains the Store_No and Store_Name with a reference to the Report_Name field in the query.

9. *Replace the text box in the report that contains the reference to the* **Store_No** *and* **Store_Name** *with* = ***“For ”&[Report_Name]***
10. **Save** *and close both the* **rptInc_Stmt** *and* **qry10Inc_Stmt**.

It looks like we're finished. Let's run a few reports and see if they make sense.

11. *In the form, run the report for the* **City** *of* **Raleigh**, **January 2016**.

    Note: *To delete a value in a Combo box, simply select the value and press [Del].*



Summary Income Statement, January 2016

For Raleigh

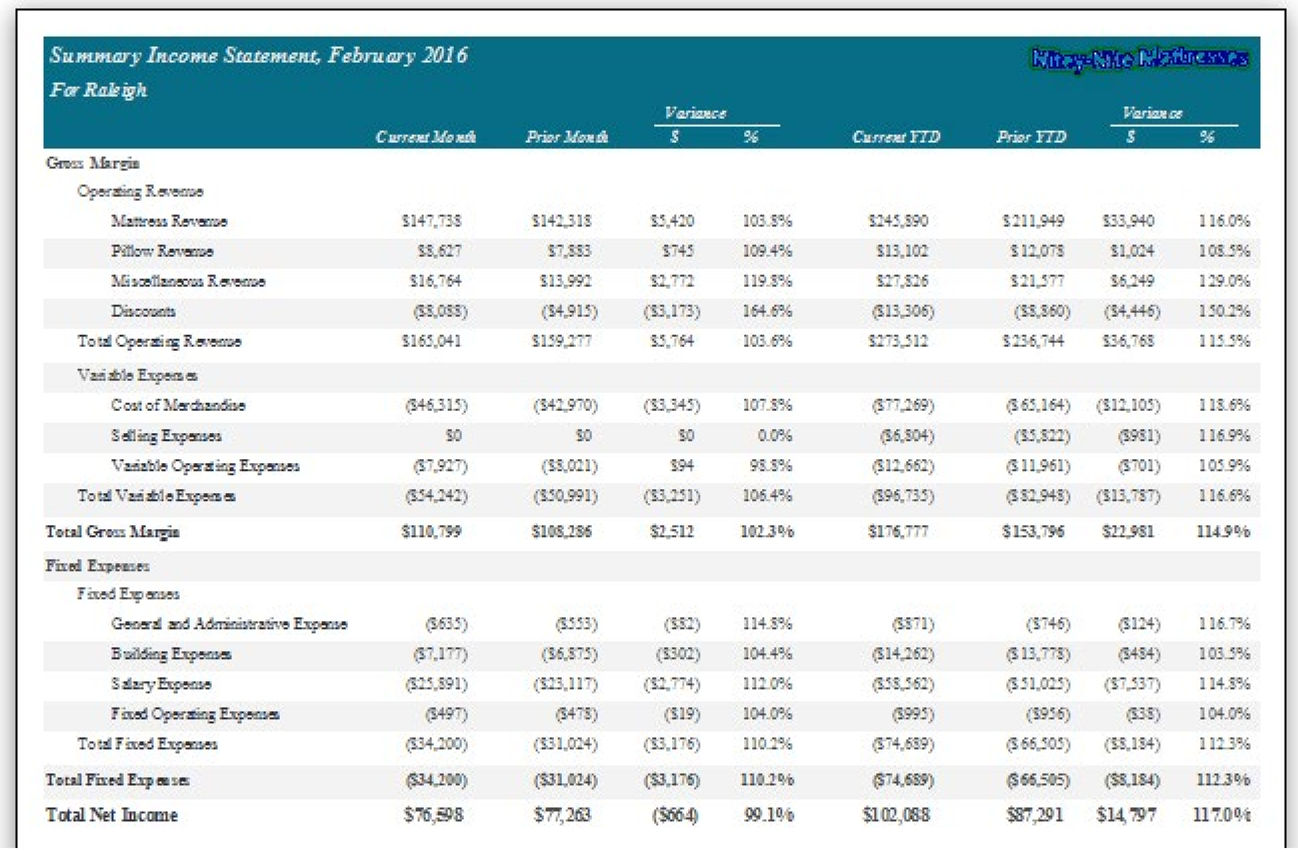
	Current Month	*Prior Month*	*Variance*		*Current YTD*	*Prior YTD*	*Variance*	
			$	*%*			*$*	*%*
Gross Margin								
Operating Revenue								
Mattress Revenue	$98,152	$69,631	$28,521	141.0%	$98,152	$69,631	$28,521	141.0%
Pillow Revenue	$4,475	$4,195	$279	106.7%	$4,475	$4,195	$279	106.7%
Miscellaneous Revenue	$11,062	$7,585	$3,476	145.8%	$11,062	$7,585	$3,476	145.8%
Discounts	($5,218)	($3,945)	($1,272)	132.3%	($5,218)	($3,945)	($1,272)	132.3%
Total Operating Revenue	$108,471	$77,467	$31,004	140.0%	$108,471	$77,467	$31,004	140.0%
Variable Expenses								
Cost of Merchandise	($30,954)	($22,194)	($8,760)	139.5%	($30,954)	($22,194)	($8,760)	139.5%
Selling Expenses	($6,804)	($5,822)	($981)	116.9%	($6,804)	($5,822)	($981)	116.9%
Variable Operating Expenses	($4,735)	($3,940)	($794)	120.2%	($4,735)	($3,940)	($794)	120.2%
Total Variable Expenses	($42,493)	($31,957)	($10,535)	133.0%	($42,493)	($31,957)	($10,535)	133.0%
Total Gross Margin	**$65,978**	**$45,509**	**$20,469**	**145.0%**	**$65,978**	**$45,509**	**$20,469**	**145.0%**
Fixed Expenses								
Fixed Expenses								
General and Administrative Expense	($235)	($193)	($42)	121.9%	($235)	($193)	($42)	121.9%
Building Expenses	($7,085)	($6,903)	($183)	102.6%	($7,085)	($6,903)	($183)	102.6%
Salary Expense	($32,671)	($27,907)	($4,764)	117.1%	($32,671)	($27,907)	($4,764)	117.1%
Fixed Operating Expenses	($497)	($478)	($19)	104.0%	($497)	($478)	($19)	104.0%
Total Fixed Expenses	($40,489)	($35,481)	($5,008)	114.1%	($40,489)	($35,481)	($5,008)	114.1%
Total Fixed Expenses	**($40,489)**	**($35,481)**	**($5,008)**	**114.1%**	**($40,489)**	**($35,481)**	**($5,008)**	**114.1%**
Total Net Income	**$25,489**	**$10,028**	**$15,461**	**254.2%**	**$25,489**	**$10,028**	**$15,461**	**254.2%**

Monday, January 2, 2017 *Page 1 of 1*

Figure 12.11

If you go back through the data, you will see these are the correct monthly and year-to-date numbers. Both the current month Net Income and YTD numbers are $25,489, which seems reasonable since we're running the report just for January. Let's change the month and see if it works.

12. *Close the* **Print Preview** *of the report.*
13. *Run the report for* **February 2016** *for the city of* **Raleigh**.



Summary Income Statement, February 2016
For Raleigh

Nitey-Nite Mattresses

	Current Month	Prior Month	Variance $	Variance %	Current YTD	Prior YTD	Variance $	Variance %
Gross Margin								
Operating Revenue								
Mattress Revenue	$147,738	$142,318	$5,420	103.8%	$245,890	$211,949	$33,940	116.0%
Pillow Revenue	$8,627	$7,883	$745	109.4%	$13,102	$12,078	$1,024	108.5%
Miscellaneous Revenue	$16,764	$13,992	$2,772	119.8%	$27,826	$21,577	$6,249	129.0%
Discounts	($8,088)	($4,915)	($3,173)	164.6%	($13,306)	($8,860)	($4,446)	150.2%
Total Operating Revenue	$165,041	$159,277	$5,764	103.6%	$273,512	$236,744	$36,768	115.5%
Variable Expenses								
Cost of Merchandise	($46,315)	($42,970)	($3,345)	107.8%	($77,269)	($65,164)	($12,105)	118.6%
Selling Expenses	$0	$0	$0	0.0%	($6,804)	($5,822)	($981)	116.9%
Variable Operating Expenses	($7,927)	($8,021)	$94	98.8%	($12,662)	($11,961)	($701)	105.9%
Total Variable Expenses	($54,242)	($50,991)	($3,251)	106.4%	($96,735)	($82,948)	($13,787)	116.6%
Total Gross Margin	$110,799	$108,286	$2,512	102.3%	$176,777	$153,796	$22,981	114.9%
Fixed Expenses								
Fixed Expenses								
General and Administrative Expense	($635)	($553)	($82)	114.8%	($871)	($746)	($124)	116.7%
Building Expenses	($7,177)	($6,875)	($302)	104.4%	($14,262)	($13,778)	($484)	103.5%
Salary Expense	($25,891)	($23,117)	($2,774)	112.0%	($58,562)	($51,025)	($7,537)	114.8%
Fixed Operating Expenses	($497)	($478)	($19)	104.0%	($995)	($956)	($38)	104.0%
Total Fixed Expenses	($34,200)	($31,024)	($3,176)	110.2%	($74,689)	($66,505)	($8,184)	112.3%
Total Fixed Expenses	($34,200)	($31,024)	($3,176)	110.2%	($74,689)	($66,505)	($8,184)	112.3%
Total Net Income	$76,598	$77,263	($664)	99.1%	$102,088	$87,291	$14,797	117.0%

Figure 12.12

You get a monthly Net Income number of $76,598, which seems reasonable because January is typically a very low sales month, and February is historically higher, but look at the year-to-date number. February YTD for the current year is $102,088. Does that seem right? It should be January's net income number of $25,489 plus February's number of $76,598 for a February YTD net income of $102,088. That is correct. Depending on your database field settings, you might see a Total Net Income of $616,349, which is not correct. In case you run into similar cases where Access does not correctly bring in the field formatting from the original table, let's look at what you could expect.

In previous versions of Access, the date filter would read the month numbers as Short text rather than as Date/Time and would pull in data from months 1 and 2 as planned, and 10, 11, and 12 as well. In that case, it looks like the YTD total would be adding in a lot more numbers than necessary. While the querying capability has improved in this case, look to see what the query can do in the background, as this could help with troubleshooting similar issues in the future. Let's audit our query to see if anything is amiss there.

14. *Close the* **rptInc_Stmt** *and open* **qry10Inc_Stmt** *in* **Design View**.

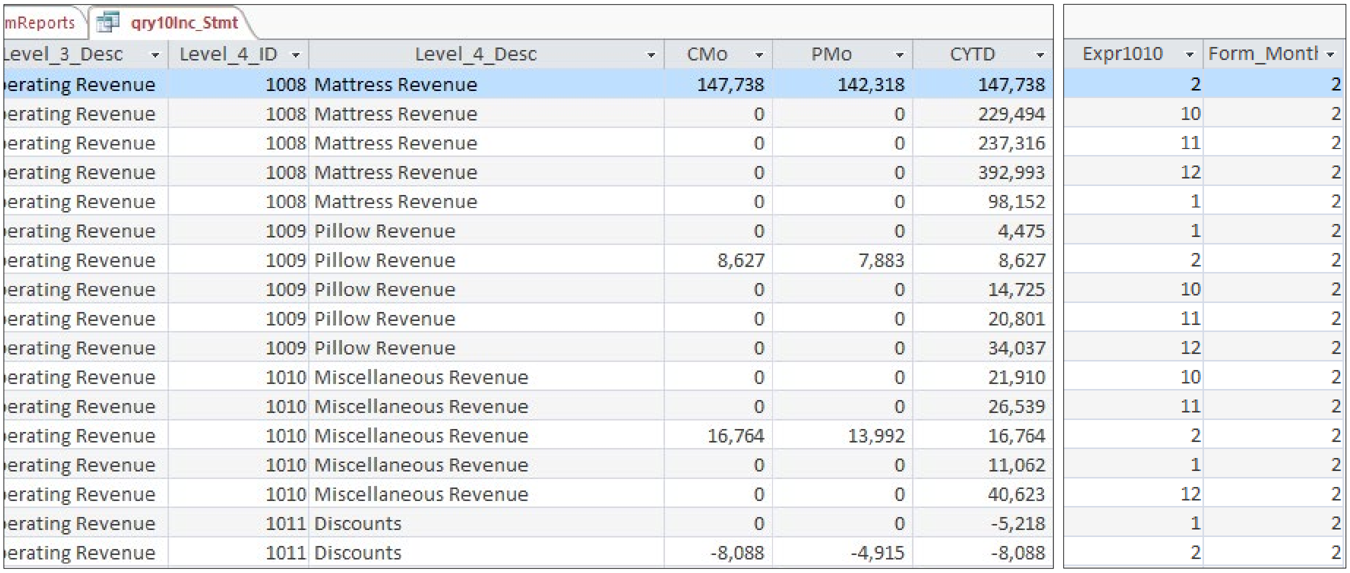
There are a lot of fields and calculations in the query and it would be real time consuming to go through every one of those. In the real world, that's what you have to do. But when you start to analyze this income statement, you think that maybe the query is somehow pulling in more months than it should.

Since we're Access experts, we've designed the query where it is real easy to find out which months the query is pulling in. It should be bringing in only months 1 and 2. That is easy to see. All we have to do is to change the Month Total line to a Where clause and run the query. Let's also create a field that brings in only the month value from the form.

15. *In the* **Design View** *of the query, change the* **Month([gl_date])** *field* **Total** *line to* **Group By**, *check the* **Show** *box.*
16. *To the right of the* **Month([gl_date])** *field, insert a column (use the* **Insert Columns** *icon in the* **Query Setup** *group of the* **Design** *tab) and write the following formula:* ***Form_Month: [Forms]![frmReports]![cmbMonth]***

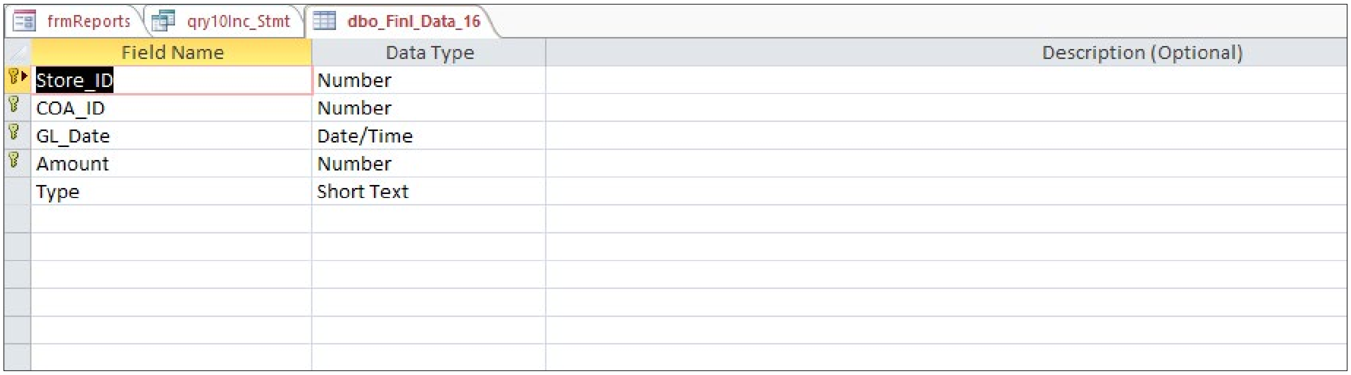
This formula will use the form to query only the data specified. You need to have the form still open with the variables set to February 2016 for the city of Raleigh for the query to run properly.

17. **Run** *the query without saving it.*



evel_3_Desc	Level_4_ID	Level_4_Desc	CMo	PMo	CYTD	Expr1010	Form_Montl
erating Revenue	1008	Mattress Revenue	147,738	142,318	147,738	2	2
erating Revenue	1008	Mattress Revenue	0	0	229,494	10	2
erating Revenue	1008	Mattress Revenue	0	0	237,316	11	2
erating Revenue	1008	Mattress Revenue	0	0	392,993	12	2
erating Revenue	1008	Mattress Revenue	0	0	98,152	1	2
erating Revenue	1009	Pillow Revenue	0	0	4,475	1	2
erating Revenue	1009	Pillow Revenue	8,627	7,883	8,627	2	2
erating Revenue	1009	Pillow Revenue	0	0	14,725	10	2
erating Revenue	1009	Pillow Revenue	0	0	20,801	11	2
erating Revenue	1009	Pillow Revenue	0	0	34,037	12	2
erating Revenue	1010	Miscellaneous Revenue	0	0	21,910	10	2
erating Revenue	1010	Miscellaneous Revenue	0	0	26,539	11	2
erating Revenue	1010	Miscellaneous Revenue	16,764	13,992	16,764	2	2
erating Revenue	1010	Miscellaneous Revenue	0	0	11,062	1	2
erating Revenue	1010	Miscellaneous Revenue	0	0	40,623	12	2
erating Revenue	1011	Discounts	0	0	-5,218	1	2
erating Revenue	1011	Discounts	-8,088	-4,915	-8,088	2	2

Figure 12.13



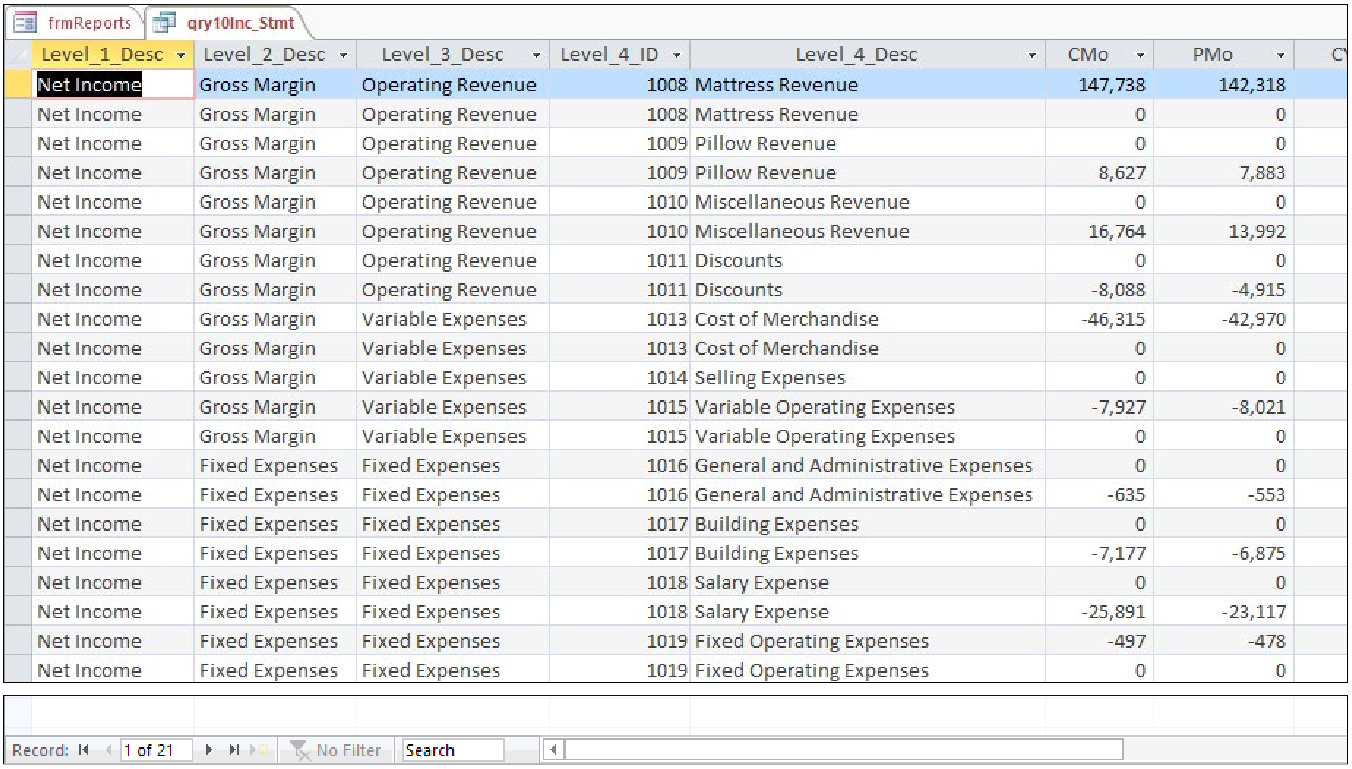
Field Name	Data Type	Description (Optional)
Store_ID	Number	
COA_ID	Number	
GL_Date	Date/Time	
Amount	Number	
Type	Short Text	

Figure 12.14

You should get 54 records of data, but look at the next-to-the-last column. For some reason, it is bringing in months 10, 11, and 12 into the query in addition to 1 and 2. Why would that happen? Remember about the many times I have told you about the difference between text and numbers (and proper database design)? When the month value is being passed from the form, it is being passed through as a text field. In the Month([gl_date]) criteria field, we wrote a formula to pull in values from the query that are between 1 and whatever value is being passed from the form. The value being passed from the form is the text "2", not the number 2. Therefore, it is querying 1, 2, 10, 11, and 12 as all as those values are between 1 and "2". This could get very tricky to figure out, and this is a great example to illustrate this issue.

Now the question is, how do you fix it? Well, you need to get the criteria of the Month([gl_date]) query to pull in only numbers, not text. If you've taken the Excel course, you will have learned the +0 trick. Hmmmmm… Will that work here? If you add a zero to the month value being passed through from the form, that may work. Let's try it.

18. *Change the criteria in the* **Month([gl_date])** *field to read* ***Between 1 And [Forms]![frmReports]![cmbMonth]+0*** *and run the query.*



Level_1_Desc	Level_2_Desc	Level_3_Desc	Level_4_ID	Level_4_Desc	CMo	PMo	C
Net Income	Gross Margin	Operating Revenue	1008	Mattress Revenue	147,738	142,318	
Net Income	Gross Margin	Operating Revenue	1008	Mattress Revenue	0	0	
Net Income	Gross Margin	Operating Revenue	1009	Pillow Revenue	0	0	
Net Income	Gross Margin	Operating Revenue	1009	Pillow Revenue	8,627	7,883	
Net Income	Gross Margin	Operating Revenue	1010	Miscellaneous Revenue	0	0	
Net Income	Gross Margin	Operating Revenue	1010	Miscellaneous Revenue	16,764	13,992	
Net Income	Gross Margin	Operating Revenue	1011	Discounts	0	0	
Net Income	Gross Margin	Operating Revenue	1011	Discounts	-8,088	-4,915	
Net Income	Gross Margin	Variable Expenses	1013	Cost of Merchandise	-46,315	-42,970	
Net Income	Gross Margin	Variable Expenses	1013	Cost of Merchandise	0	0	
Net Income	Gross Margin	Variable Expenses	1014	Selling Expenses	0	0	
Net Income	Gross Margin	Variable Expenses	1015	Variable Operating Expenses	-7,927	-8,021	
Net Income	Gross Margin	Variable Expenses	1015	Variable Operating Expenses	0	0	
Net Income	Fixed Expenses	Fixed Expenses	1016	General and Administrative Expenses	0	0	
Net Income	Fixed Expenses	Fixed Expenses	1016	General and Administrative Expenses	-635	-553	
Net Income	Fixed Expenses	Fixed Expenses	1017	Building Expenses	0	0	
Net Income	Fixed Expenses	Fixed Expenses	1017	Building Expenses	-7,177	-6,875	
Net Income	Fixed Expenses	Fixed Expenses	1018	Salary Expense	0	0	
Net Income	Fixed Expenses	Fixed Expenses	1018	Salary Expense	-25,891	-23,117	
Net Income	Fixed Expenses	Fixed Expenses	1019	Fixed Operating Expenses	-497	-478	
Net Income	Fixed Expenses	Fixed Expenses	1019	Fixed Operating Expenses	0	0	

Figure 12.15

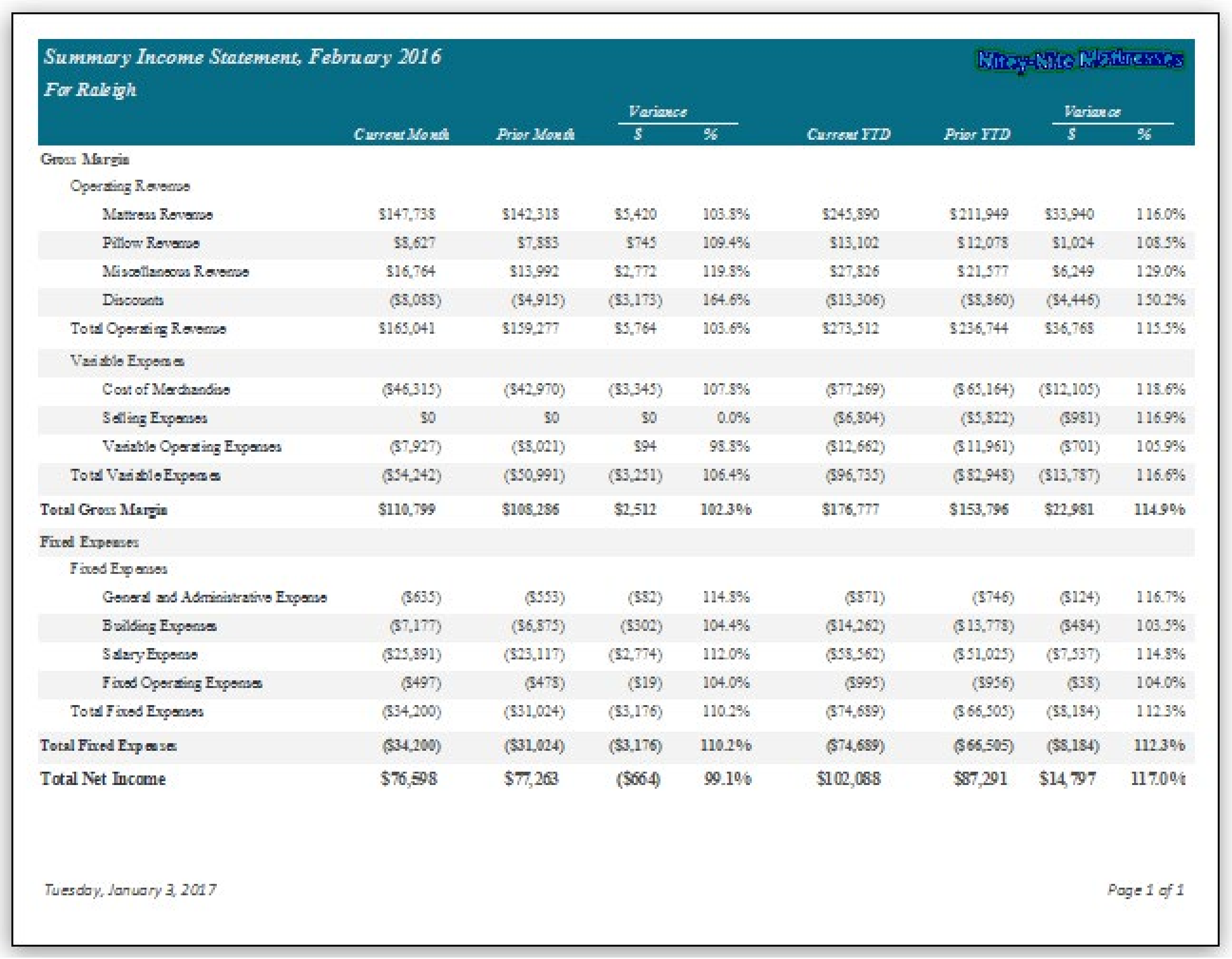
Woohoo!!! Now you get only 21 records and months 10, 11, and 12 are not in the record set. Life is beautiful!

19. *Change the* **Month([gl_date])** *field back to a* **Where** *clause and delete the* **Form_Month**

field (you don't need it in your query anymore). Make sure you keep the **+0** *in the query criteria.*

20. **Save** *and close* **qry10Inc_Stmt** *and run the report from the form again.*

> ***Tip****: In a case like this one, you could also check the base data table in* **Design View** *to see how the data is formatted. In the* **dbo_Finl_Data_16***, you can see that* **gl_data** *is formatted as* **Date/Time** *rather than as* **Text***, so the original query would pull the* **Month** *number as intended. If you end up with tables where the dates are imported and formatted as Text for whatever reason, the +0 trick can make all the difference without having to modify the original data.*



Summary Income Statement, February 2016

For Raleigh

Nitey-Nite Mattresses

			Variance				*Variance*	
	Current Month	*Prior Month*	*$*	*%*	*Current YTD*	*Prior YTD*	*$*	*%*
Gross Margin								
Operating Revenue								
Mattress Revenue	$147,738	$142,318	$5,420	103.8%	$245,890	$211,949	$33,940	116.0%
Pillow Revenue	$8,627	$7,883	$745	109.4%	$13,102	$12,078	$1,024	108.5%
Miscellaneous Revenue	$16,764	$13,992	$2,772	119.8%	$27,826	$21,577	$6,249	129.0%
Discounts	($8,088)	($4,915)	($3,173)	164.6%	($13,306)	($8,860)	($4,446)	150.2%
Total Operating Revenue	$165,041	$159,277	$5,764	103.6%	$273,512	$236,744	$36,768	115.5%
Variable Expenses								
Cost of Merchandise	($46,315)	($42,970)	($3,345)	107.8%	($77,269)	($65,164)	($12,105)	118.6%
Selling Expenses	$0	$0	$0	0.0%	($6,804)	($5,822)	($981)	116.9%
Variable Operating Expenses	($7,927)	($8,021)	$94	98.8%	($12,662)	($11,961)	($701)	105.9%
Total Variable Expenses	($54,242)	($50,991)	($3,251)	106.4%	($96,735)	($82,948)	($13,787)	116.6%
Total Gross Margin	**$110,799**	**$108,286**	**$2,512**	**102.3%**	**$176,777**	**$153,796**	**$22,981**	**114.9%**
Fixed Expenses								
Fixed Expenses								
General and Administrative Expense	($635)	($553)	($82)	114.8%	($871)	($746)	($124)	116.7%
Building Expenses	($7,177)	($6,875)	($302)	104.4%	($14,262)	($13,778)	($484)	103.5%
Salary Expense	($25,891)	($23,117)	($2,774)	112.0%	($58,562)	($51,025)	($7,537)	114.8%
Fixed Operating Expenses	($497)	($478)	($19)	104.0%	($995)	($956)	($38)	104.0%
Total Fixed Expenses	($34,200)	($31,024)	($3,176)	110.2%	($74,689)	($66,505)	($8,184)	112.3%
Total Fixed Expenses	**($34,200)**	**($31,024)**	**($3,176)**	**110.2%**	**($74,689)**	**($66,505)**	**($8,184)**	**112.3%**
Total Net Income	**$76,598**	**$77,263**	**($664)**	**99.1%**	**$102,088**	**$87,291**	**$14,797**	**117.0%**

Tuesday, January 3, 2017 *Page 1 of 1*

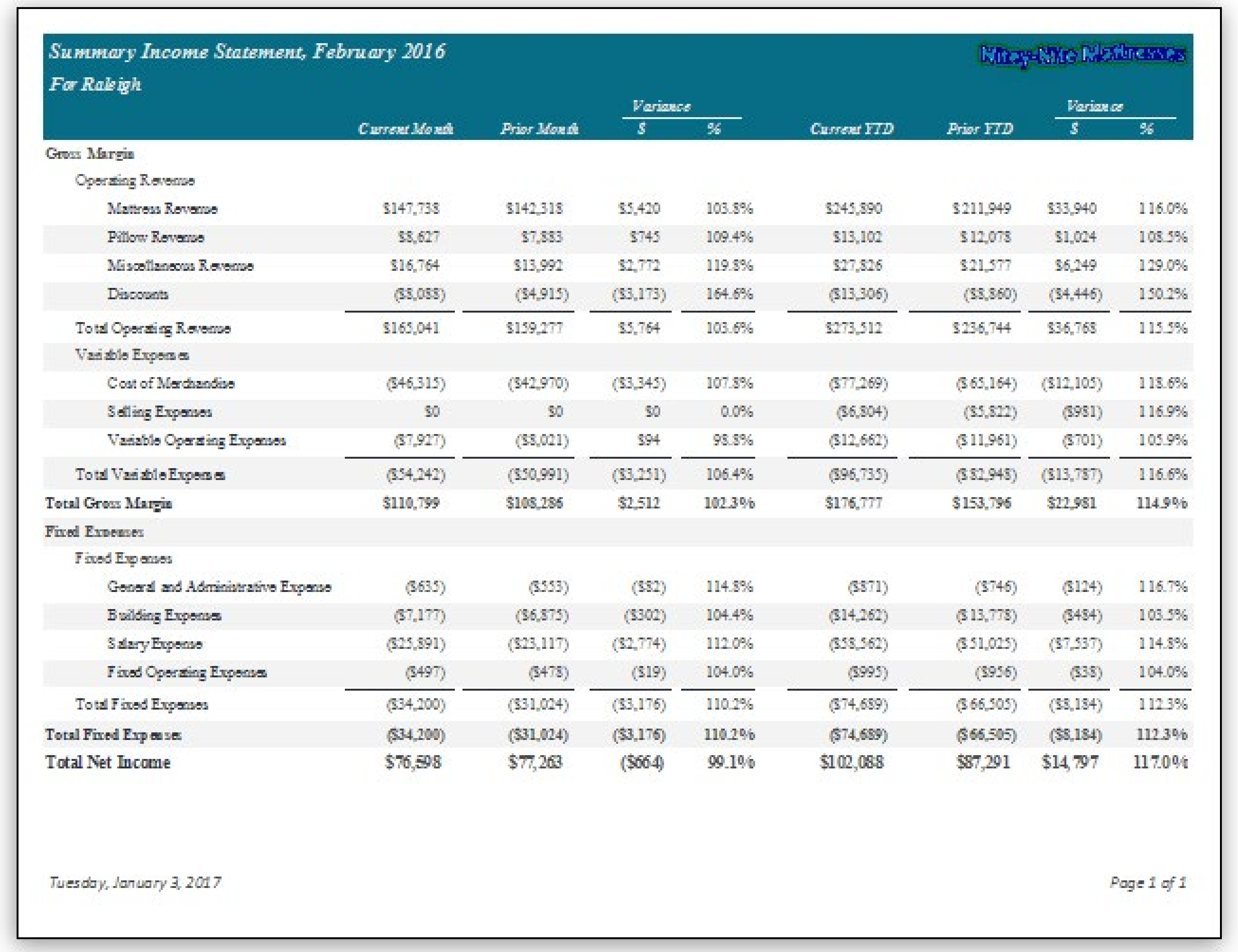
Figure 12.16

Now it works. February's YTD number is now $102,088 (it's a little different than our addition of the two months together due to rounding). This should be a good lesson for you. Whenever you are developing reports, check and recheck your numbers. Remember to act like a carpenter – measure twice and cut once. Few things are more embarrassing for a developer to deliver numbers that he or she has been working on for a long time and someone finds an error like this.

There's one more thing I'd like to do to the report before we call it quits. If you look at the report, there is no obvious break between Total Operating Revenue and the Discounts number just above it. It would be nice if we had the Discounts number underlined, but if you underline that number, each of the other Revenue numbers would also be underlined. A way to get around that is to put a line over the Total Operating Revenue number. Let's do that.

21. *Go to the* **Design View** *of the* **rptInc_Stmt** *and draw a line using the* **Line** *icon just above each text box in the* **Level_3_Desc Footer** *section.*
22. **Format** *all lines with a* **Border Style Solid**, **Border Width 1 pt** *and* **Border Color** *as* **Text Black**.

You may have to go back and forth between Print Preview and Design views to get it just right. Depending on the alignment of the text boxes in the Level_3_Desc Footer section, you may have to move those text boxes down a little to make room for the lines. Once you have that done, your report should look like Figure 12.17.



Summary Income Statement, February 2016
For Raleigh

Nitey-Nite Mattresses

			Variance				Variance	
	Current Month	Prior Month	$	%	Current YTD	Prior YTD	$	%
Gross Margin								
Operating Revenue								
Mattress Revenue	$147,738	$142,318	$5,420	103.8%	$245,890	$211,949	$33,940	116.0%
Pillow Revenue	$8,627	$7,883	$745	109.4%	$13,102	$12,078	$1,024	108.5%
Miscellaneous Revenue	$16,764	$13,992	$2,772	119.8%	$27,826	$21,577	$6,249	129.0%
Discounts	($8,088)	($4,915)	($3,173)	164.6%	($13,306)	($8,860)	($4,446)	150.2%
Total Operating Revenue	$165,041	$159,277	$5,764	103.6%	$273,512	$236,744	$36,768	115.5%
Variable Expenses								
Cost of Merchandise	($46,315)	($42,970)	($3,345)	107.8%	($77,269)	($65,164)	($12,105)	118.6%
Selling Expenses	$0	$0	$0	0.0%	($6,804)	($5,822)	($981)	116.9%
Variable Operating Expenses	($7,927)	($8,021)	$94	98.8%	($12,662)	($11,961)	($701)	105.9%
Total Variable Expenses	($54,242)	($50,991)	($3,251)	106.4%	($96,735)	($82,948)	($13,787)	116.6%
Total Gross Margin	$110,799	$108,286	$2,512	102.3%	$176,777	$153,796	$22,981	114.9%
Fixed Expenses								
Fixed Expenses								
General and Administrative Expense	($635)	($553)	($82)	114.8%	($871)	($746)	($124)	116.7%
Building Expenses	($7,177)	($6,875)	($302)	104.4%	($14,262)	($13,778)	($484)	103.5%
Salary Expense	($25,891)	($23,117)	($2,774)	112.0%	($58,562)	($51,025)	($7,537)	114.8%
Fixed Operating Expenses	($497)	($478)	($19)	104.0%	($995)	($956)	($38)	104.0%
Total Fixed Expenses	($34,200)	($31,024)	($3,176)	110.2%	($74,689)	($66,505)	($8,184)	112.3%
Total Fixed Expenses	($34,200)	($31,024)	($3,176)	110.2%	($74,689)	($66,505)	($8,184)	112.3%
Total Net Income	$76,598	$77,263	($664)	99.1%	$102,088	$87,291	$14,797	117.0%

Tuesday, January 3, 2017

Page 1 of 1

Figure 12.17

Create the Detail Statement

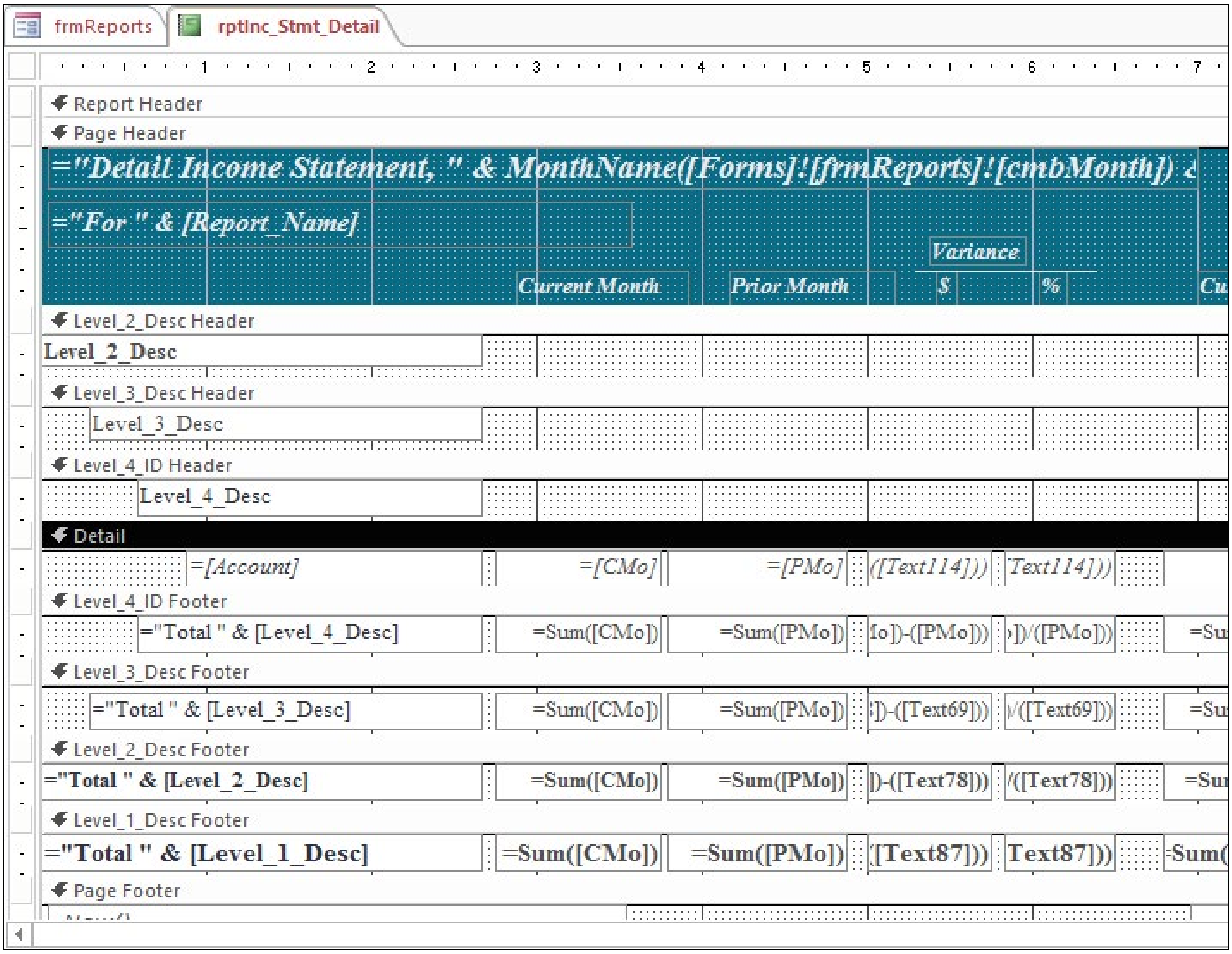
Now that you have your summary statement working as it should, creating the ***detail statement*** should be easy to do. First you will create the query, then create the report, and then create a command button on the form to run that report. You've already done all of that for the Summary report, so all you have to do now is to copy that query and report and modify them for the Detail report. Let's do it.

1. *Close all queries and* **frmReports** *(make sure to save everything).*
2. **Copy qry10Inc_Stmt** *and rename the copy* ***qry10Inc_Stmt_Detail****.*
3. *In* **qry10Inc_Stmt_Detail***, create a new field called* ***Account*** *placed to the right of* **Level_4_Desc** *with the following formula:* ***[Level_5_Acct] & " - " & [Level_5_Desc]****.*

All you're doing here is adding the last level of detail, the account level.

4. *Open* **frmReports***, choose any* **year** *and* **store***, and run the query to make sure it works.*

5. **Save** *and close* **qryInc_Stmt_Detail**.
6. **Copy rptInc_Stmt** *and rename as* ***rptInc_Stmt_Detail***.
7. *In the report design of* **rptInc_Stmt_Detail**, *change* **Record Source** *to* **qry10Inc_Stmt_Detail** *(done in the* **Property Sheet***).*
8. *Expand the* **Detail** *section.*
9. *Display the* **Footer** *section for* **Level_4_ID**.
10. **Copy** *everything in the* **Level_4_ID Header** *section to be in the* **Level_4_ID Footer** *section and align appropriately.*
11. *Modify the* **Level_4_Desc** *text box in the* **Level_4_ID Footer** *section to read =****"Total " &[Level_4_Desc]***
12. *Delete all text boxes in* **Level_4_ID Header** *section except for the* **Level_4_Desc** *text box.*
13. **Copy** *all objects in the* **Level_4_ID Footer** *section up to the* **Detail** *section and align appropriately.*
14. *Modify the* **Level_4_Desc** *text box in the* **Detail** *section to pull in the field called* **Account**.
15. *Modify the text boxes in the* **Detail** *section to be* ***CMo***, ***PMo***, ***CYTD***, *and* ***PYTD*** *(not "sum") and edit the variance formulas to work correctly.*
16. *Make everything in the* **Detail** *section italicized and indent the* **Account** *text box* **seven grid dots**.
17. *Change the formula that begins with "***Summary Income Statement…***" to "****Detail Income Statement…***"*
18. *Test, save, and close the report.*



Report Header

Page Header

="Detail Income Statement, " & MonthName([Forms]![frmReports]![cmbMonth]) &

="For " & [Report_Name]

Variance

Current Month | Prior Month | $ | % | Cu

Level_2_Desc Header

Level_2_Desc

Level_3_Desc Header

Level_3_Desc

Level_4_ID Header

Level_4_Desc

Detail

=[Account] | =[CMo] | =[PMo] | ([Text114])) | Text114]))

Level_4_ID Footer

="Total " & [Level_4_Desc] | =Sum([CMo]) | =Sum([PMo]) | Io])-([PMo])) |)])/([PMo])) | =Su

Level_3_Desc Footer

="Total " & [Level_3_Desc] | =Sum([CMo]) | =Sum([PMo]) |])-([Text69])) | /([Text69])) | =Su

Level_2_Desc Footer

="Total " & [Level_2_Desc] | =Sum([CMo]) | =Sum([PMo]) |])-([Text78])) | /([Text78])) | =Su

Level_1_Desc Footer

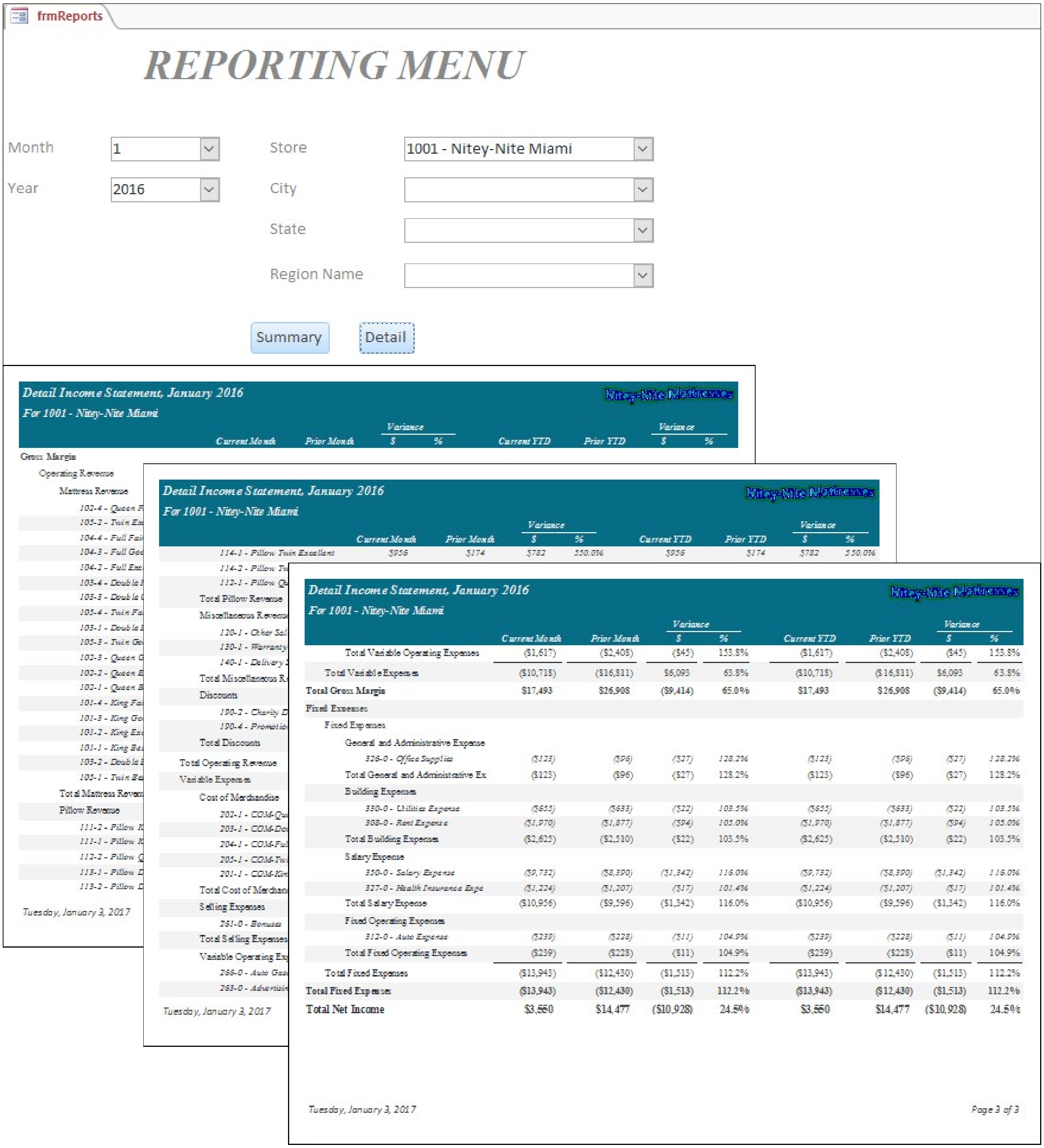
="Total " & [Level_1_Desc] | =Sum([CMo]) | =Sum([PMo]) | [Text87])) | Text87])) | Sum(

Page Footer

Figure 12.18

Now all that's left to do is to create a command button on the form to run the report.

19. *Open* **frmReports** *in* **Design View**.
20. *Create a command* **Button** *called* **Detail** *that previews* **rptInc_Stmt_Detail**.
21. *Position both command* **Buttons** *where they are in the middle of the form.*
22. *Test the* **Detail** *button's functionality, including scrolling through the report pages.*



REPORTING MENU

Month	1	Store	1001 - Nitey-Nite Miami
Year	2016	City	
		State	
		Region Name	

Detail Income Statement, January 2016
For 1001 - Nitey-Nite Miami

Gross Margin
Operating Revenue
Mattress Revenue

Tuesday, January 3, 2017

Detail Income Statement, January 2016
For 1001 - Nitey-Nite Miami

	Current Month	Prior Month	Variance $	Variance %	Current YTD	Prior YTD	Variance $	Variance %
114-1 - Pillow Twin Excellent	$956	$174	$782	550.0%	$956	$174	$782	550.0%

Tuesday, January 3, 2017

Detail Income Statement, January 2016
For 1001 - Nitey-Nite Miami

	Current Month	Prior Month	Variance $	Variance %	Current YTD	Prior YTD	Variance $	Variance %
Total Variable Operating Expenses	($1,617)	($2,408)	($45)	153.8%	($1,617)	($2,408)	($45)	153.8%
Total Variable Expenses	($10,718)	($16,811)	$6,093	63.8%	($10,718)	($16,811)	$6,093	63.8%
Total Gross Margin	$17,493	$26,908	($9,414)	65.0%	$17,493	$26,908	($9,414)	65.0%
Fixed Expenses								
Fixed Expenses								
General and Administrative Expense								
326-0 - Office Supplies	($123)	($96)	($27)	128.2%	($123)	($96)	($27)	128.2%
Total General and Administrative Ex	($123)	($96)	($27)	128.2%	($123)	($96)	($27)	128.2%
Building Expenses								
330-0 - Utilities Expense	($655)	($633)	($22)	103.5%	($655)	($633)	($22)	103.5%
308-0 - Rent Expense	($1,970)	($1,877)	($94)	105.0%	($1,970)	($1,877)	($94)	105.0%
Total Building Expenses	($2,625)	($2,510)	($22)	103.5%	($2,625)	($2,510)	($22)	103.5%
Salary Expense								
350-0 - Salary Expense	($9,732)	($8,390)	($1,342)	116.0%	($9,732)	($8,390)	($1,342)	116.0%
327-0 - Health Insurance Expe	($1,224)	($1,207)	($17)	101.4%	($1,224)	($1,207)	($17)	101.4%
Total Salary Expense	($10,956)	($9,596)	($1,342)	116.0%	($10,956)	($9,596)	($1,342)	116.0%
Fixed Operating Expenses								
312-0 - Auto Expense	($239)	($228)	($11)	104.9%	($239)	($228)	($11)	104.9%
Total Fixed Operating Expenses	($239)	($228)	($11)	104.9%	($239)	($228)	($11)	104.9%
Total Fixed Expenses	($13,943)	($12,430)	($1,513)	112.2%	($13,943)	($12,430)	($1,513)	112.2%
Total Fixed Expenses	($13,943)	($12,430)	($1,513)	112.2%	($13,943)	($12,430)	($1,513)	112.2%
Total Net Income	$3,550	$14,477	($10,928)	24.5%	$3,550	$14,477	($10,928)	24.5%

Tuesday, January 3, 2017 Page 3 of 3

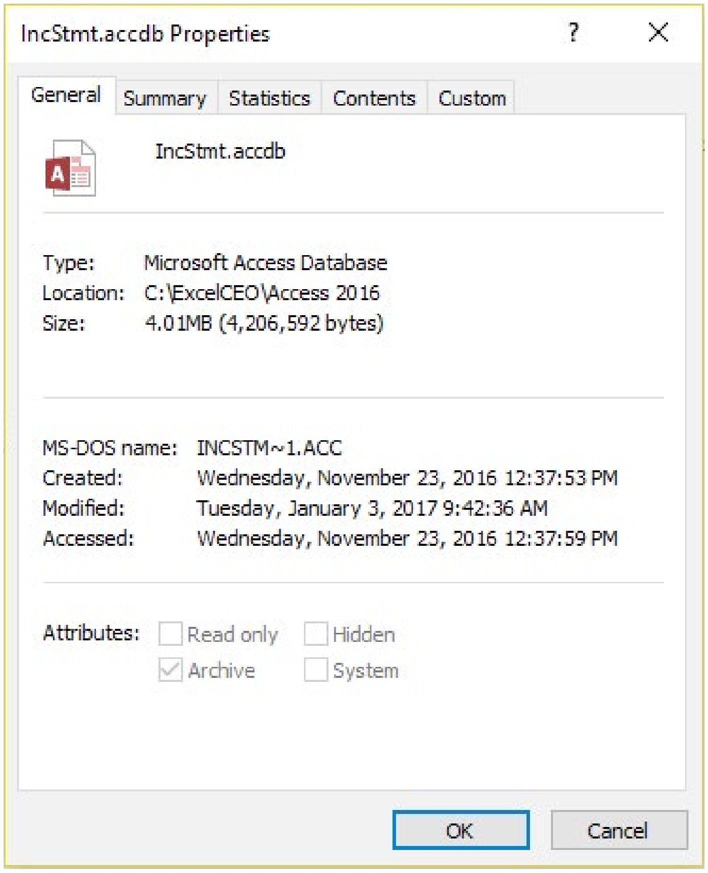
Figure 12.19

23. **Save** *and close* **frmReports**.

Database Utilities

Let's finish this chapter by reviewing some tips about the performance of the database and documenting your work. First we'll talk about performance. Every time you run a query in Access or bring in a new table, the database grows in size. When you delete data or objects from an Access database, it generally decreases the size, but some extra space is left over. I'll compare it to a parking lot with lots of cars in it. The parking lot (or database) is full when all spaces are taken up with cars. However, people don't generally leave a parking lot in an orderly fashion, and as cars leave the parking lot, it will be left with empty spaces in various spots throughout the lot. All of those spaces add to the size of the database, even though there's nothing there. To make the parking lot operate at its maximum efficiency, you could move all of the cars to the front or back of the lot in a nice orderly line. Unless you're in the valet parking business and you have some spare time on your hands, that's usually not practical to do. Access has a nifty utility to clean up that extra space. Let's use that utility to see if we can clean up the Inc_Stmt database a little.

1. *Close all queries, forms, and reports.*
2. *Click on the* **File** *tab and click on the* **Info** *section on the left side of the screen.*
3. *All the way to the right of the screen, click on the "***View and edit database properties***" link.*
4. *In the* **Inc_Stmt.accdb Properties** *dialog box, click on the* **General** *tab.*



IncStmt.accdb

Type: Microsoft Access Database
Location: C:\ExcelCEO\Access 2016
Size: 4.01MB (4,206,592 bytes)

MS-DOS name: INCSTM~1.ACC
Created: Wednesday, November 23, 2016 12:37:53 PM
Modified: Tuesday, January 3, 2017 9:42:36 AM
Accessed: Wednesday, November 23, 2016 12:37:59 PM

Attributes: □ Read only □ Hidden
☑ Archive □ System

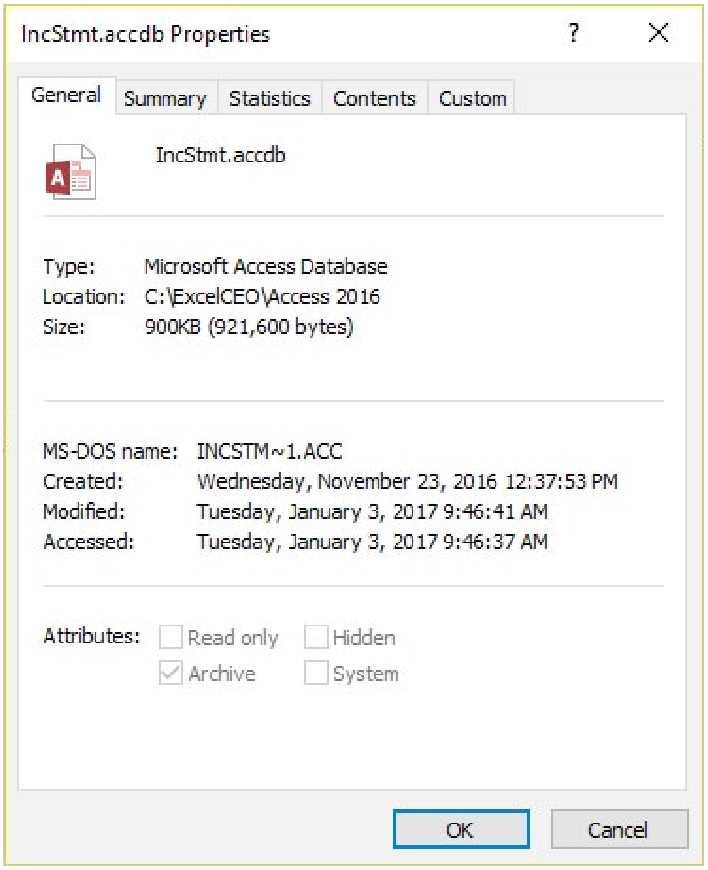
Figure 12.20

The Inc_Stmt Properties dialog box shows that my database contains 4.01 MB of space. This may be a different number than yours, as I may have used the database more or less than you did. We'll now run the utility to clean it up and we'll see if we save some space.

5. *Cancel out of the* **Inc_Stmt Properties** *dialog box.*
6. *Click on the* **Compact & Repair Database** *icon in the* **Info** *tab.*

The Compact and Repair Database utility will run very quickly and return you to the Home tab of the Access database.

7. *Display the* **Inc_Stmt.accdb Properties** *dialog box again.*



IncStmt.accdb Properties

IncStmt.accdb

Type: Microsoft Access Database
Location: C:\ExcelCEO\Access 2016
Size: 900KB (921,600 bytes)

MS-DOS name: INCSTM~1.ACC
Created: Wednesday, November 23, 2016 12:37:53 PM
Modified: Tuesday, January 3, 2017 9:46:41 AM
Accessed: Tuesday, January 3, 2017 9:46:37 AM

Attributes: □ Read only □ Hidden
☑ Archive □ System

Figure 12.21

The database now has about 900KB instead of 4.01MB. Frequently, you will have such a savings in space, and the database typically runs faster when it is smaller. One tidbit of information (which may or may not be on the test…): an Access database can generally increase in size to about 1.1 gigabyte. Once it gets that large, you have to compact it down or split it up into more than one database. When it gets larger than 1.1 GB, it doesn't have enough room to compact, and thus the database will keep increasing in size with no possibility of compacting it, so it is important that you occasionally compact and repair your Access database. One time I was consulting with a user who had a database that had never been compacted. The size of the database when I first saw it was 497 MB. After compacting it, it went down to 4 MB. You should periodically compact and repair ALL Access databases.

Analyze Performance

Now let's talk about another tool called Analyze Performance. The ***Analyze Performance*** tool analyzes the objects you choose and offers suggestions on how you can enhance their performance. Let's try one to see if we could improve the performance on something.

1. *Close the* **Inc_Stmt.accdb Properties** *dialog box.*
2. *Click on the* **Back arrow** *(to return to the database), click on the* **Database Tools** *tab then on* **qry10Inc_Stmt**.
3. *Click on the* **Analyze Performance** Analyze Performance *icon in the* **Analyze** *group.*

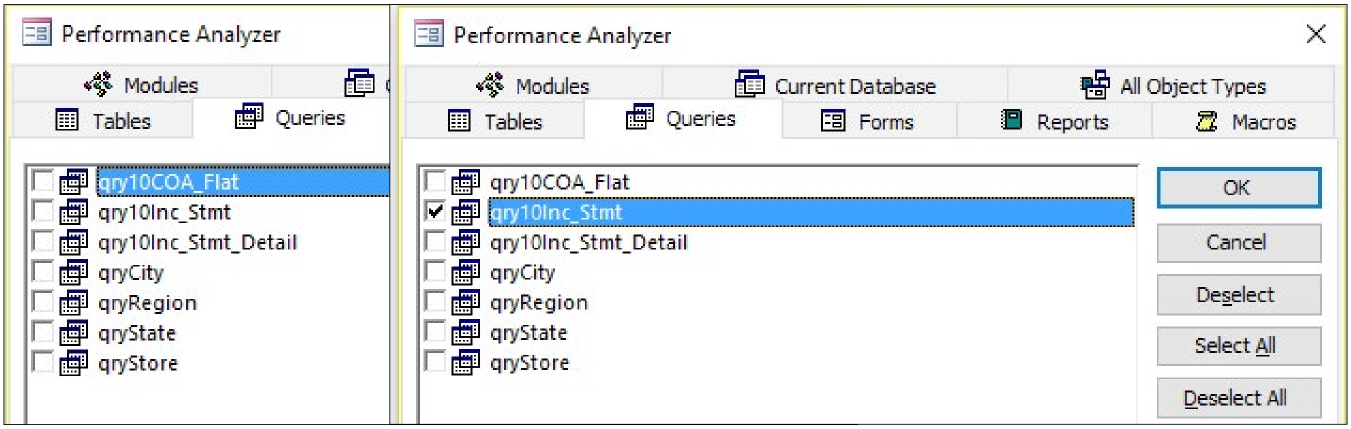
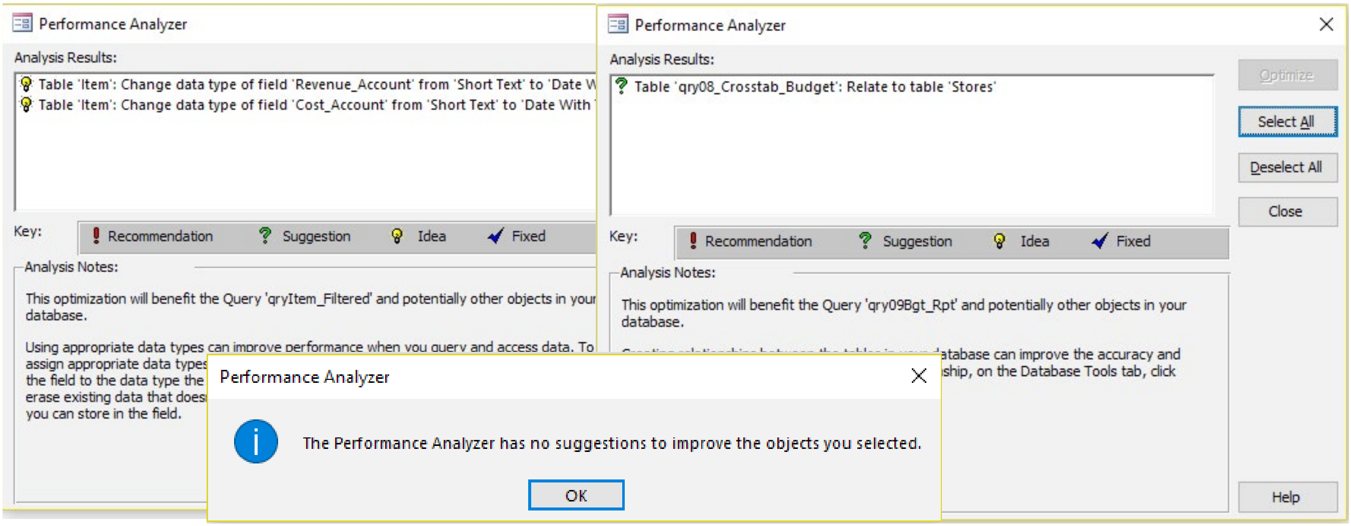


Figure 12.22

4. *Check the box next to* **qry10Inc_Stmt** *and click* **OK**.



Analysis Results:

Table 'Item': Change data type of field 'Revenue_Account' from 'Short Text' to 'Date W

Table 'Item': Change data type of field 'Cost_Account' from 'Short Text' to 'Date With

Analysis Notes:

This optimization will benefit the Query 'qryItem_Filtered' and potentially other objects in your database.

Using appropriate data types can improve performance when you query and access data. To assign appropriate data types the field to the data type the erase existing data that does you can store in the field.

Analysis Results:

Table 'qry08_Crosstab_Budget': Relate to table 'Stores'

Analysis Notes:

This optimization will benefit the Query 'qry09Bgt_Rpt' and potentially other objects in your database.

atabase can improve the accuracy and ship, on the Database Tools tab, click

The Performance Analyzer has no suggestions to improve the objects you selected.

Figure 12.23

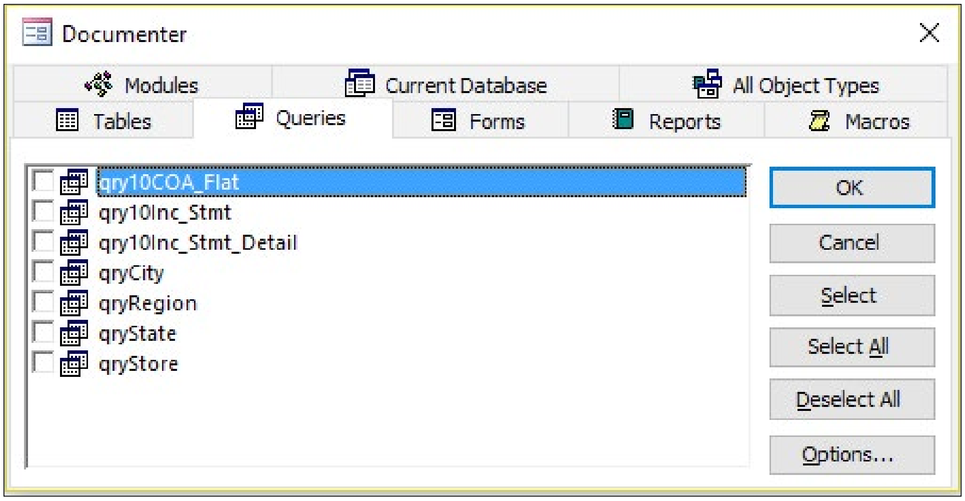
Depending how you have worked on your projects in this database so far, you could see one of the above Performance Analyzer suggestions. The only suggestion it is offering is that we should create relationships in the Relationships View joining the three tables we're using in the query. That is a good suggestion, but since we're done with the report and you know how to create relationships, I'll let you do that on your own, if you want. The other images from Performance Analyzer show what you could expect as related to other tables. Sometimes there are no suggestions to be made.

5. *Click* **Close**.

Documenter

Another nifty tool (especially good at annoying Sarbanes-Oxley auditors) is the Documenter. The ***Documenter*** works just like Analyze Performance, but it documents everything you ever wanted to know about the object, but were afraid to ask. Let's try the documenter on qry10Inc_Stmt.

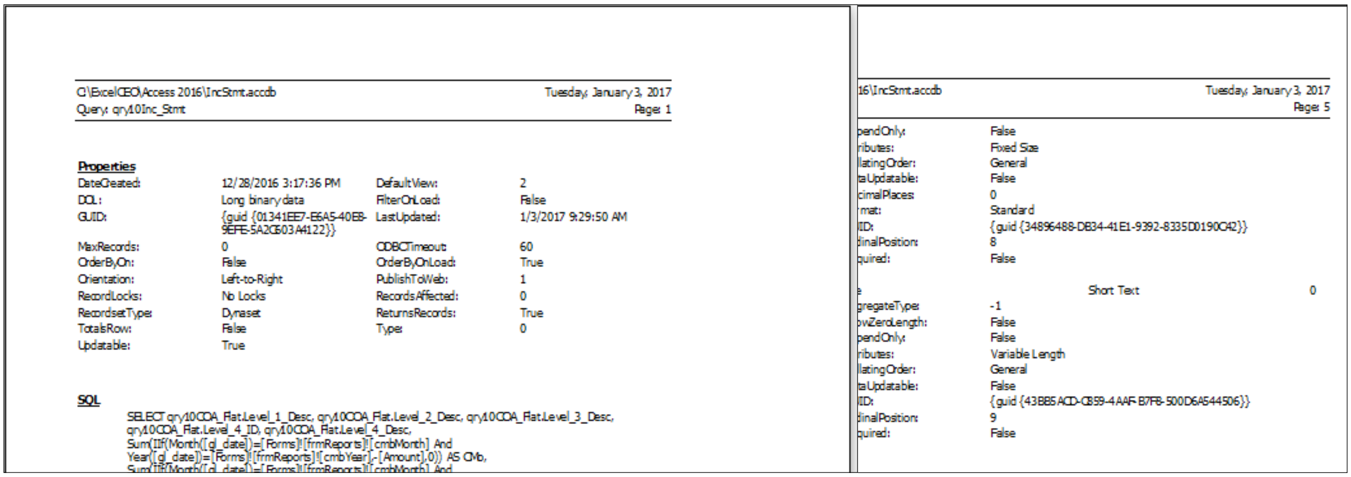
1. *Click on the* **Database Documenter** Database Documenter *icon in the* **Analyze** *group of the* **Database Tools** *tab.*



- □ qry10COA_Flat
- □ qry10Inc_Stmt
- □ qry10Inc_Stmt_Detail
- □ qryCity
- □ qryRegion
- □ qryState
- □ qryStore

Figure 12.24

2. *Click on* **qry10Inc_Stmt** *and click* **OK**.



```
C:\ExcelCEO\Access 2016\IncStmt.accdb                              Tuesday, January 3, 2017
Query: qry10Inc_Stmt                                                                Page: 1

Properties
DateCreated:     12/28/2016 3:17:36 PM      DefaultView:      2
DOL:             Long binary data           FilterOnLoad:     False
GUID:            {guid {01341EE7-E6A5-40EB- LastUpdated:      1/3/2017 9:29:50 AM
                 9EFE-5A2C603A4122}}
MaxRecords:      0                          ODBCTimeout:      60
OrderByOn:       False                      OrderByOnLoad:    True
Orientation:     Left-to-Right              PublishToWeb:     1
RecordLocks:     No Locks                   RecordsAffected:  0
RecordsetType:   Dynaset                    ReturnsRecords:   True
TotalsRow:       False                      Type:             0
Updatable:       True

SQL
        SELECT qry10COA_Flat.Level_1_Desc, qry10COA_Flat.Level_2_Desc, qry10COA_Flat.Level_3_Desc,
        qry10COA_Flat.Level_4_ID, qry10COA_Flat.Level_4_Desc,
        Sum(IIf(Month([gl_date])=[Forms]![frmReports]![cmbMonth] And
        Year([gl_date])=[Forms]![frmReports]![cmbYear],-[Amount],0)) AS CMo,
        Sum(IIf(Month([gl_date])=[Forms]![frmReports]![cmbMonth] And
```

```
16\IncStmt.accdb                                       Tuesday, January 3, 2017
                                                                        Page: 5
pendOnly:          False
ributes:           Fixed Size
llatingOrder:      General
taUpdatable:       False
cimalPlaces:       0
rmat:              Standard
ID:                {guid {34896488-DB34-41E1-9392-8335D0190C42}}
dinalPosition:     8
quired:            False

e                                  Short Text                         0
gregateType:       -1
owZeroLength:      False
pendOnly:          False
ributes:           Variable Length
llatingOrder:      General
taUpdatable:       False
ID:                {guid {43BB5ACD-CB59-4AAF-B7FB-500D6A544506}}
dinalPosition:     9
quired:            False
```

Figure 12.25

It returns a five page report (yours may vary) that gives you comprehensive documentation of everything about that query, including: its properties, SQL code, parameters, fields (columns), indexes (if any), and permissions. Although I don't use it a lot, it may come in handy for database documentation, if needed.

3. *Close out of* **Documenter**.

> ***Review Questions****: It is now time to complete the hands-on Review Questions. Log on to www.ExcelCEO.com with your Email and Password, click on the* **Access 2016 and SQL Review Questions, Chapter 12, Section 1 of 1** *option and complete the review questions.*

Conclusion

In this chapter, you tied the variables in the form to a query that ran the report using the Expression Builder in the criteria section of the query and also in the alias formulas. After that complex summary report was ready, you copied the report to create a similar detail report. Lastly, we reviewed how to use database Utilities (including Compact and Repair Database), Performance Analyzer and Documenter.

Access Conclusion

You have now finished the Access portion of the course. I would encourage you to buy a comprehensive Access reference manual. I've taught you the basics of Access reporting and analysis, but there is still much to learn. You will probably refer back to some of the exercises you did in this course quite often, so you should keep this book handy.

Chapter Exam

You can now go to www.ExcelCEO.com, click on Sign In, log in and take the exam. Make sure that you take the exam on the same device on which you completed the practice files, as some of the questions on the exam may refer to some of the completed projects. Chapter exams are intended to be hands-on.

CHAPTER THIRTEEN — INTRODUCTION TO SQL

In this chapter, you will:

- Recognize simple SELECT statements in SQL View
- Identify SQL reserved words
- Determine when to use the following reserved words in SQL code: SELECT, FROM, ORDER BY, and WHERE
- Determine when to use the various operators in SQL
- Identify the AND, OR, IN, NOT, and LIKE Operators in SQL code
- Determine the appropriate Order of Evaluation in SQL Code
- Recognize the various Wildcard Operators

CPE Credits possible for this chapter: 3

SQL (Structured Query Language)

In the first twelve chapters of this course, you studied databases, specifically Access. While Access is an excellent desktop relational database, sometimes you need to apply querying capabilities to databases other than Access. ***SQL*** (pronounced "sequel" or just S-Q-L, an acronym that stands for Structured Query Language) is the universally accepted language for communicating with databases, and you can use it with database management systems (***DBMS***) like Access, SQL Server, Oracle, Paradox, or even Excel. The language was invented around 1989 and its sole purpose is to communicate with databases. It quickly became THE database language, and the vast majority of databases now have incorporated the language into their systems.

SQL is a wonderful introduction to the third and final concept accountants and financial people need – learning how to write programming code. One of the reasons that I like to use SQL as an introductory programming language is that it is relatively easy to learn, and very powerful! I continue to see more and more of my clients wanting to learn how to write SQL code.

In the first exercise, I will teach you two words and a character (SELECT, FROM and *) and POOF – you are instantly a SQL programmer. All you have to do from that point is to learn a few more tricks. The words that are used in the SQL language are English words, so if you can read this text, you can learn SQL. In fact, the entire language consists of just a few words, about 200, and we will be using only a fraction of those. SQL is not proprietary software, meaning ANYONE can program with it without having to buy any special package. Even though it is very easy to use, it is also extremely powerful.

Even though SQL is universally accepted as THE database programming language, many DBMSs have altered the language a bit to better fit into their programs and add some other functionality. When we refer to the base SQL language (without any alterations from vendors), we call it ANSI SQL, as it is governed by the American National Standards Institute (ANSI). ANSI has about 1,000 member companies that help develop and support SQL as well as other national standards. Other versions of SQL include Microsoft SQL, MySQL, Transact-SQL, PL-SQL and others. For the purposes of this course, I will teach you mostly ANSI SQL, with a few modifications to allow it to work in a Microsoft Access environment. Each time I deviate from ANSI SQL, I will show both methods.

With that intro, I hope you're excited about learning SQL. In the following two chapters, we will continue to use the Nitey-Nite Access database and create queries using ANSI SQL and Microsoft SQL. As a note, a query written in a DBMS (like SQL Server or Oracle) is called a View. A View in SQL Server is the same thing as a Query in Access. If you ever explore more serious SQL programming, you will need to use another SQL manager, but for now we can use Access and Microsoft's query tool called Microsoft Query. With that said, let's begin SQL.

1. *Open the* **Nitey_Nite_2016** *Access database.*
2. *Exit out of the* **Main Menu** *form.*
3. **Create** *a new query, but don't add any tables in the* **Design View** *and click* **Close** *on the* **Show Tables** *window.*

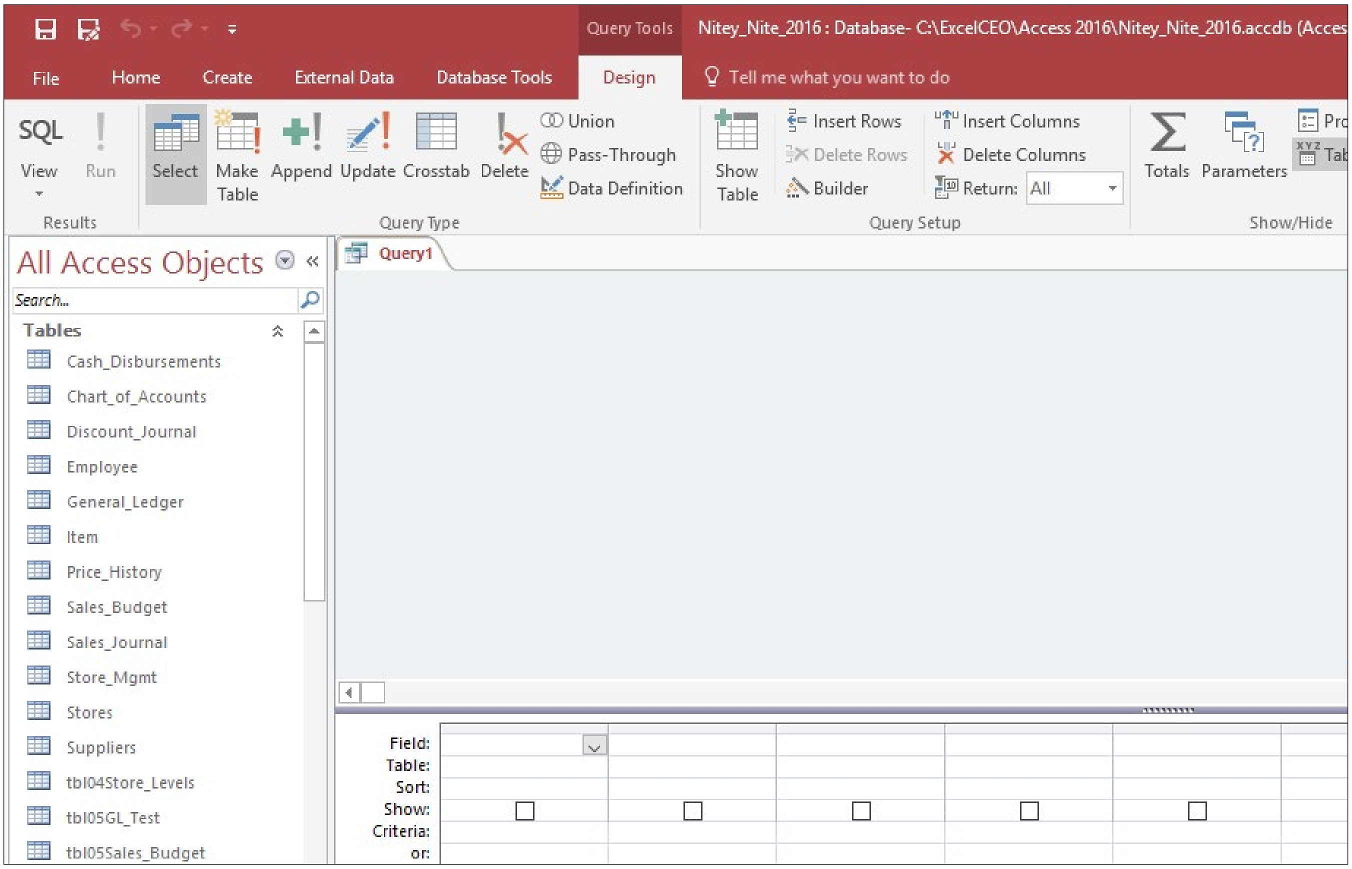
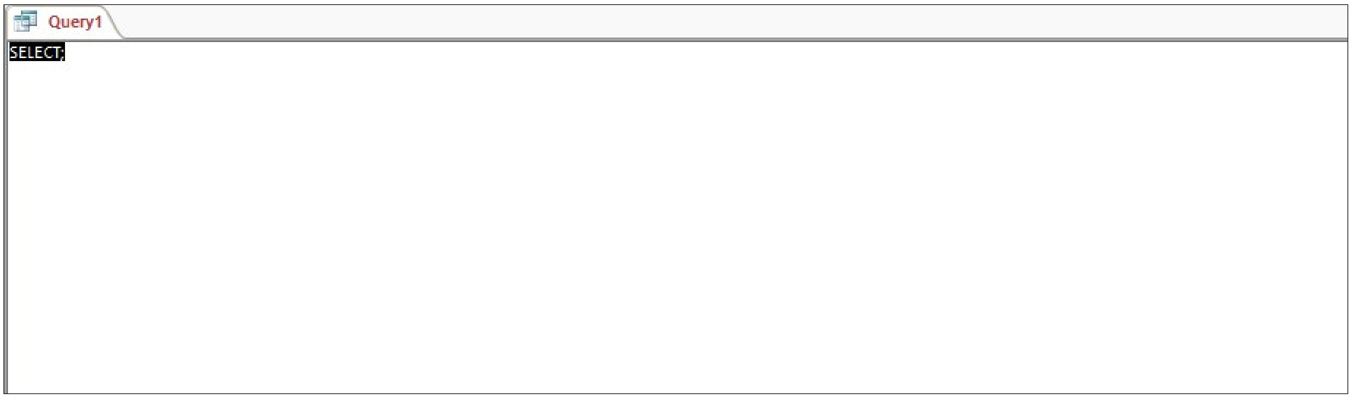


Figure 13.1

Notice on the left side of the Office ribbon the View icon reads SQL, instead of the normal View icon. It appears this way because there are no tables defined as part of the query, and you can't do much in the Design Grid without any tables. To start programming with SQL code, or to go to SQL View, click on the View icon when it reads SQL. Alternatively, you can click on the drop-down arrow at the bottom of the View icon and choose SQL view.

4. *Click on the* **SQL** *icon (or click on the drop-down arrow and choose* **SQL View***).*



```
SELECT;
```

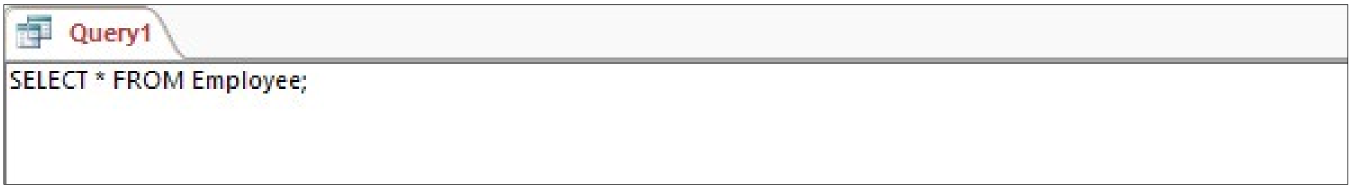
Figure 13.2

SELECT and FROM

At this point, you have an almost blank screen with only the word "*SELECT*" followed by a semi-colon. SELECT is your first SQL programming word, and FROM is your second. The simplest SQL code is the code to return everything from a specific table. The semi-colon is the character that indicates the end of the SQL statement.

You'll use the Employee table in this exercise. Let's say you want to return everything (all fields and records) from the Employee table.

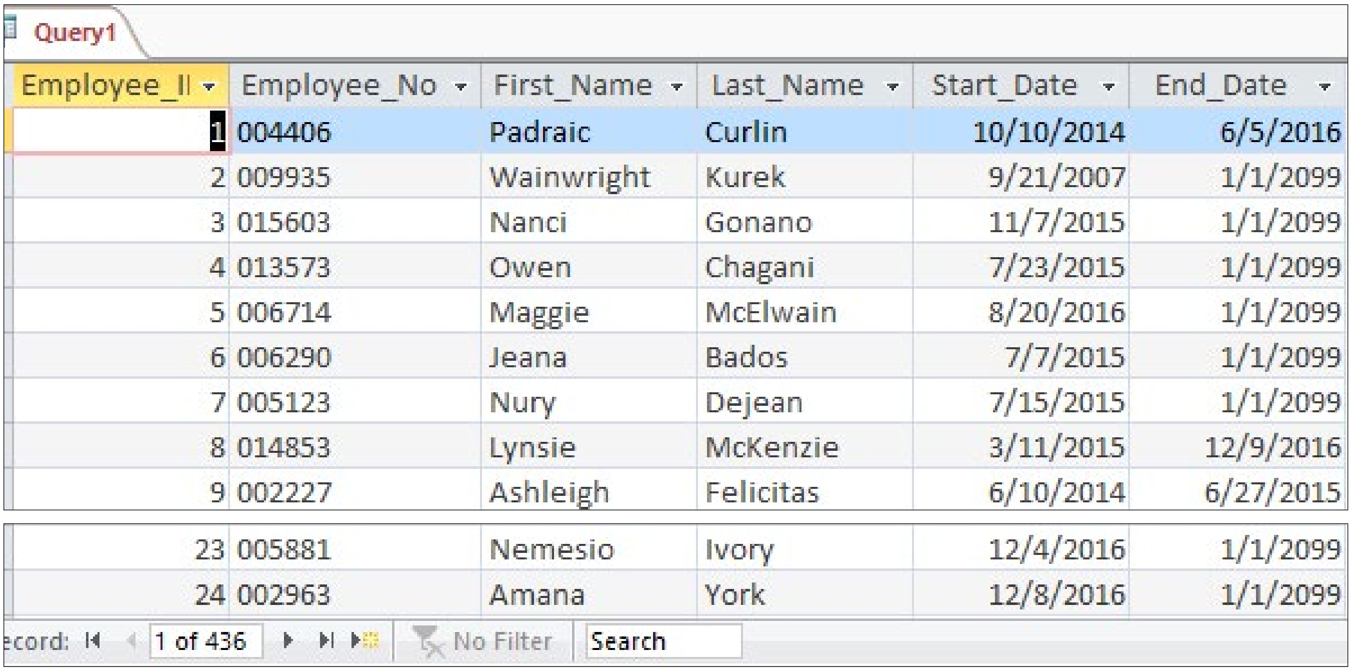
5. *In* **SQL View**, *type* ***SELECT * FROM Employee;***



```
SELECT * FROM Employee;
```

Figure 13.3

6. *Click the* **View** *icon (to see the results in* **Datasheet View***).*



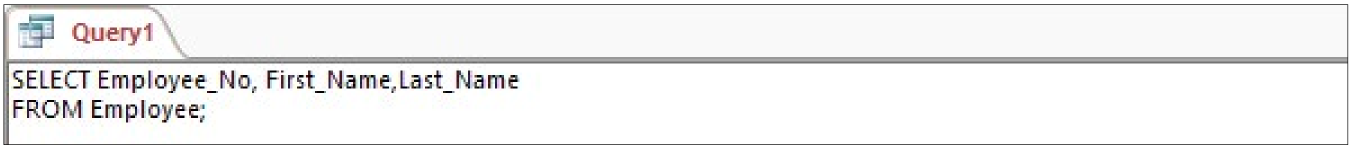
Employee_II	Employee_No	First_Name	Last_Name	Start_Date	End_Date
1	004406	Padraic	Curlin	10/10/2014	6/5/2016
2	009935	Wainwright	Kurek	9/21/2007	1/1/2099
3	015603	Nanci	Gonano	11/7/2015	1/1/2099
4	013573	Owen	Chagani	7/23/2015	1/1/2099
5	006714	Maggie	McElwain	8/20/2016	1/1/2099
6	006290	Jeana	Bados	7/7/2015	1/1/2099
7	005123	Nury	Dejean	7/15/2015	1/1/2099
8	014853	Lynsie	McKenzie	3/11/2015	12/9/2016
9	002227	Ashleigh	Felicitas	6/10/2014	6/27/2015
23	005881	Nemesio	Ivory	12/4/2016	1/1/2099
24	002963	Amana	York	12/8/2016	1/1/2099

Figure 13.4

Notice that if you click the View icon after you run the record set, it will go to the Design View of the query, not the SQL view. This is because Access has translated the SQL code in Design View. It will always go to Design View by default unless Access can't make the translation, then it will go directly to SQL view. The first few examples you will work are easily translated into Design View. Whenever I want you to go to SQL view, I'll instruct you to go to SQL view. That is done by clicking on the drop-down arrow and choosing SQL View. Feel free to look at the Design View of the query if you don't understand how something works. Sometimes it's easier (especially when beginning to learn SQL) to look at the code in Design View to understand the logic. However, the more SQL code you do, the easier it will become, and if you use it enough, you will come to a point where you will prefer to write your own code rather than to use the Design View in Access. One of my employees once told me that building a query in the Access Design View is like building a house with Legos™. Building a query with SQL is like building a house with clay. You can mold SQL code to do many things that the Access Design View can't do.

The resulting dataset from our query is simply all of the records from the Employee table. If you want to return only a few fields of data (not all of them), simply replace the asterisk with the names of the fields you want to return, separated by commas. The last field will have no comma after it. Let's query the Employee table to return the Employee_No, First_Name and Last_Name.

7. *Click the drop-down arrow on the* **View** *icon and choose* **SQL View** *(to return to* **SQL View**).
8. *Replace the* **asterisk** *with* ***Employee_No, First_Name, Last_Name***
9. *Move* **FROM Employee;** *to the second line.*

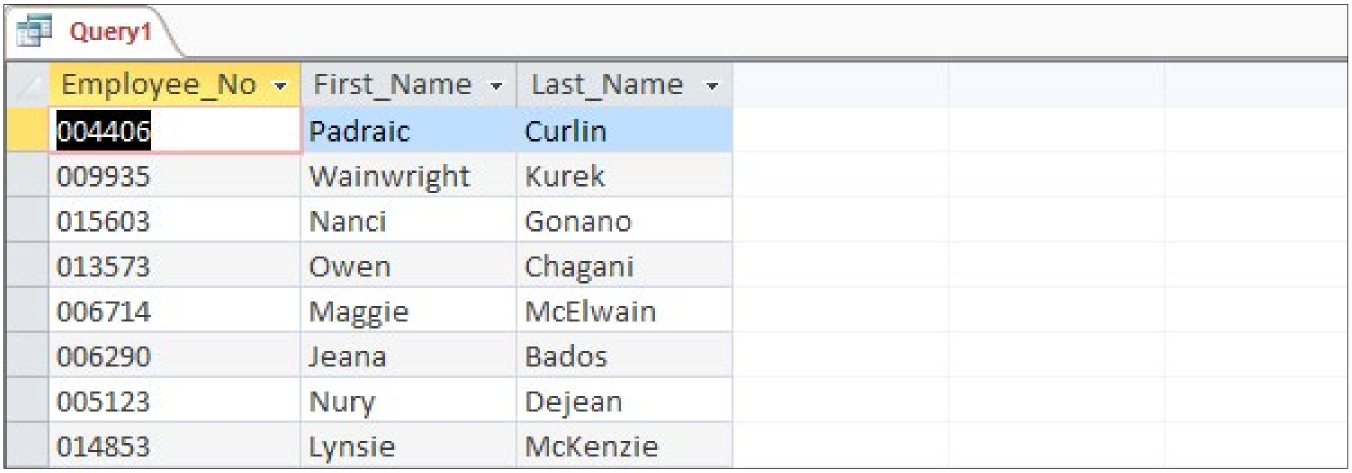


```
SELECT Employee_No, First_Name,Last_Name
FROM Employee;
```

Figure 13.5

Moving the FROM statement to the second line doesn't do anything except make the code a little easier to read and analyze.

10. **Run** *the query.*



Employee_No	First_Name	Last_Name
004406	Padraic	Curlin
009935	Wainwright	Kurek
015603	Nanci	Gonano
013573	Owen	Chagani
006714	Maggie	McElwain
006290	Jeana	Bados
005123	Nury	Dejean
014853	Lynsie	McKenzie

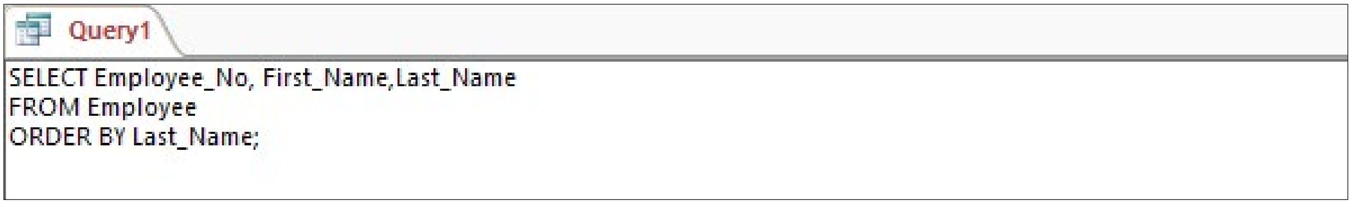
Figure 13.6

As a note, the words that make up SQL are called ***reserved words***. When I program in SQL, I like to type all reserved words in upper-case. This is not necessary, but I do it to distinguish between reserved words and all other words.

The ORDER BY Clause

Just like in Access and Excel, you may want to sort data. Sorting in SQL is done with the ORDER BY clause. Let's suppose you want to sort this record set by last name. All you do is type ORDER BY followed by the field name.

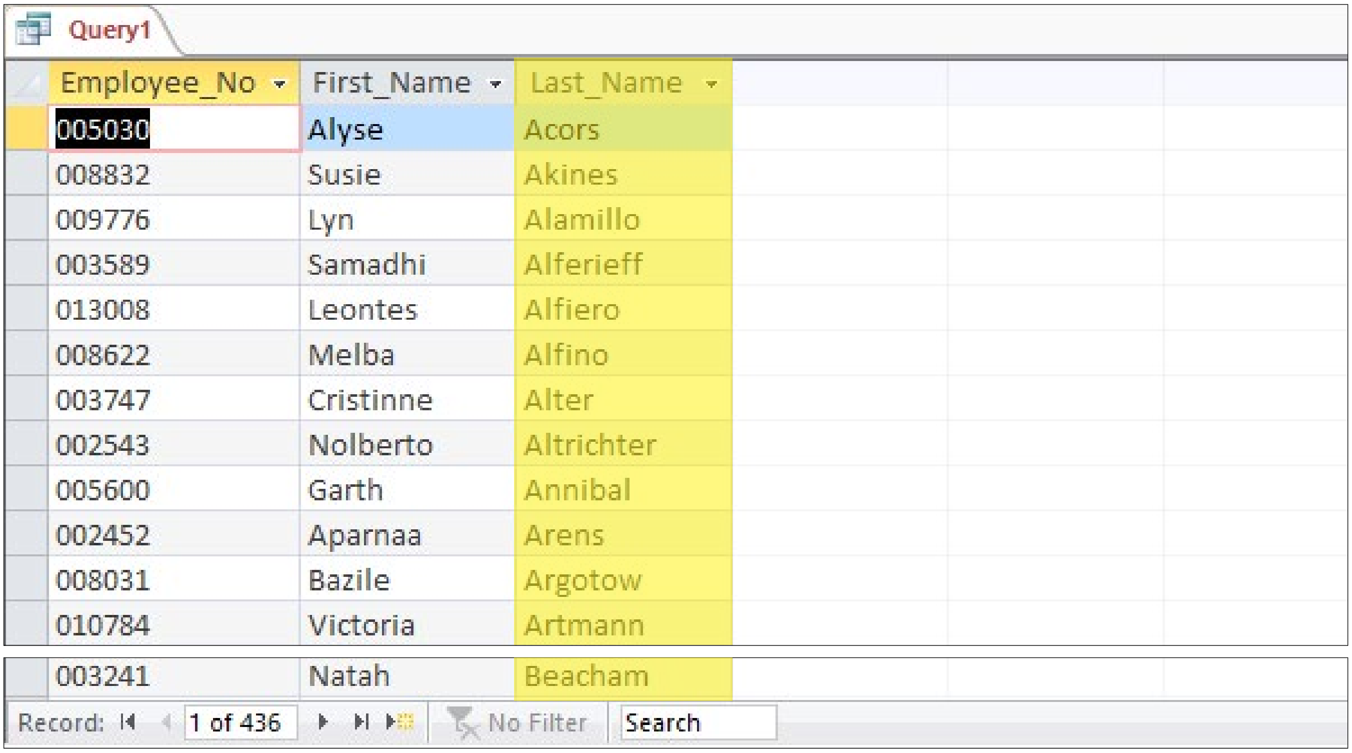
11. *Return to* **SQL View**.
12. *Edit the code as in* **Figure 13.7**:



```
SELECT Employee_No, First_Name,Last_Name
FROM Employee
ORDER BY Last_Name;
```

Figure 13.7

13. **Run** *the query.*



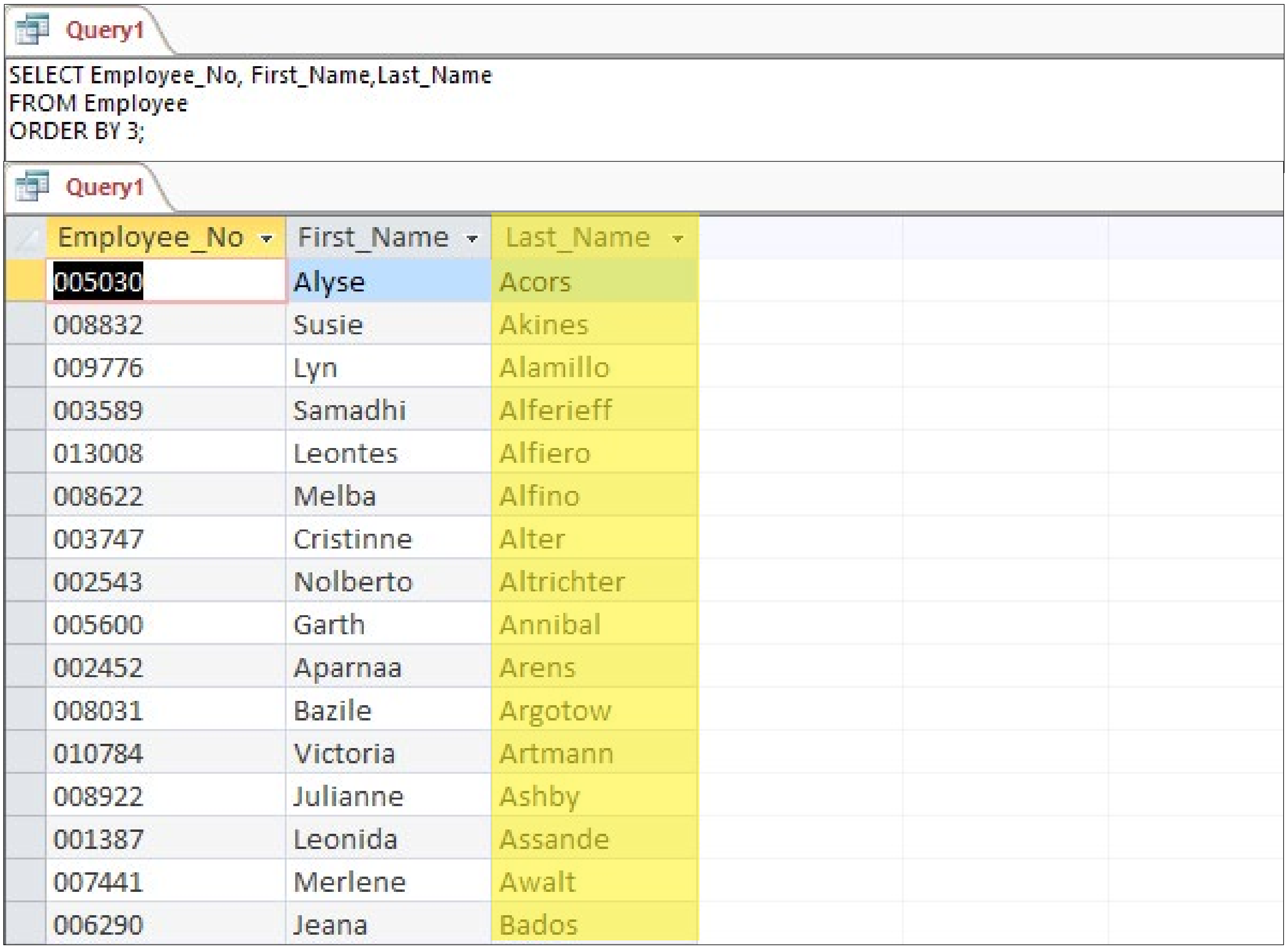
Employee_No	First_Name	Last_Name
005030	Alyse	Acors
008832	Susie	Akines
009776	Lyn	Alamillo
003589	Samadhi	Alferieff
013008	Leontes	Alfiero
008622	Melba	Alfino
003747	Cristinne	Alter
002543	Nolberto	Altrichter
005600	Garth	Annibal
002452	Aparnaa	Arens
008031	Bazile	Argotow
010784	Victoria	Artmann
003241	Natah	Beacham

Figure 13.8

From this point on, I won't always ask you to type in the code, or show a picture of the code then tell you to run it, then go back to the SQL view. Every time you change your code, you should run it to make sure it works, and that you're getting the expected results.

The data is now sorted by last name. Instead of typing in the name of the field in the ORDER BY clause, you can also type the position number of the field. The field Last_Name is in the third column or position (because that is the order in which you typed it), so you can change the ORDER BY code to ORDER BY 3 and it will run the same.

14. *Edit the code as in the example below, then* **Run** *the query.*



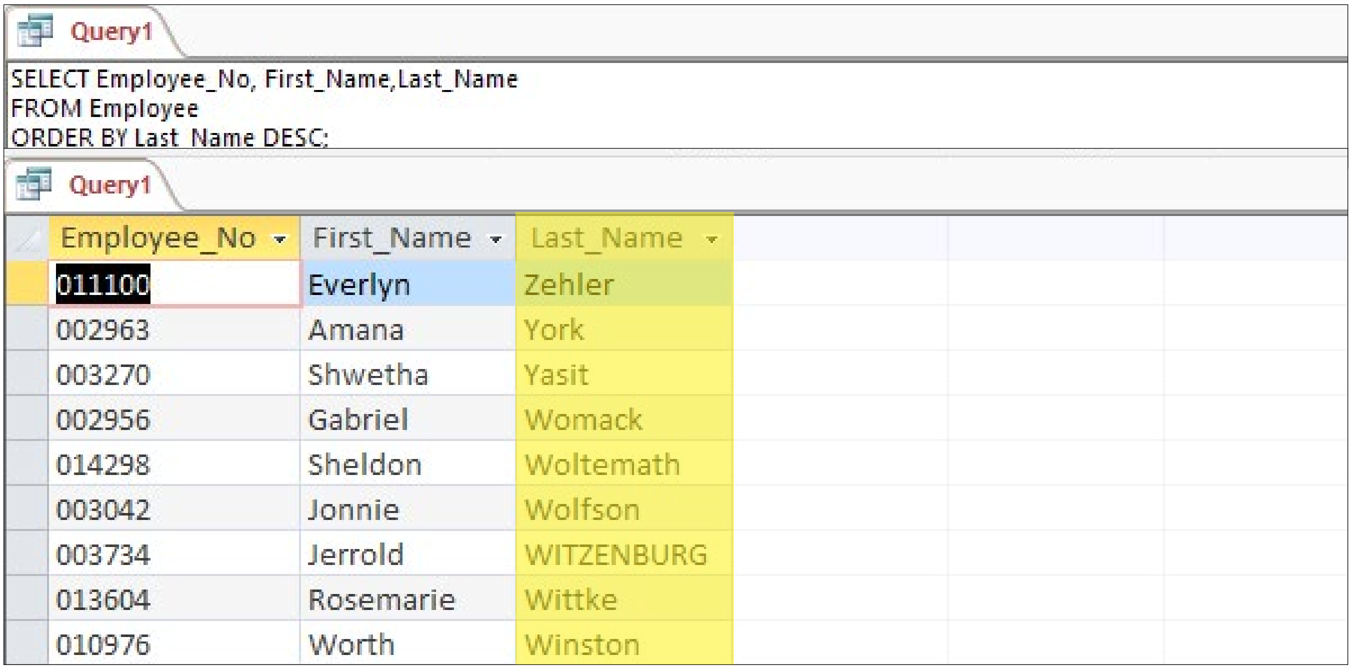
```
SELECT Employee_No, First_Name,Last_Name
FROM Employee
ORDER BY 3;
```

Employee_No	First_Name	Last_Name
005030	Alyse	Acors
008832	Susie	Akines
009776	Lyn	Alamillo
003589	Samadhi	Alferieff
013008	Leontes	Alfiero
008622	Melba	Alfino
003747	Cristinne	Alter
002543	Nolberto	Altrichter
005600	Garth	Annibal
002452	Aparnaa	Arens
008031	Bazile	Argotow
010784	Victoria	Artmann
008922	Julianne	Ashby
001387	Leonida	Assande
007441	Merlene	Awalt
006290	Jeana	Bados

Figure 13.9

By default, SQL assumes you want to sort in Ascending order. If you want to sort in Descending order, just type "desc" after the field or position, like this:

15. *Edit the code as in* **Figure 13.10** *and* **Run** *the query.*



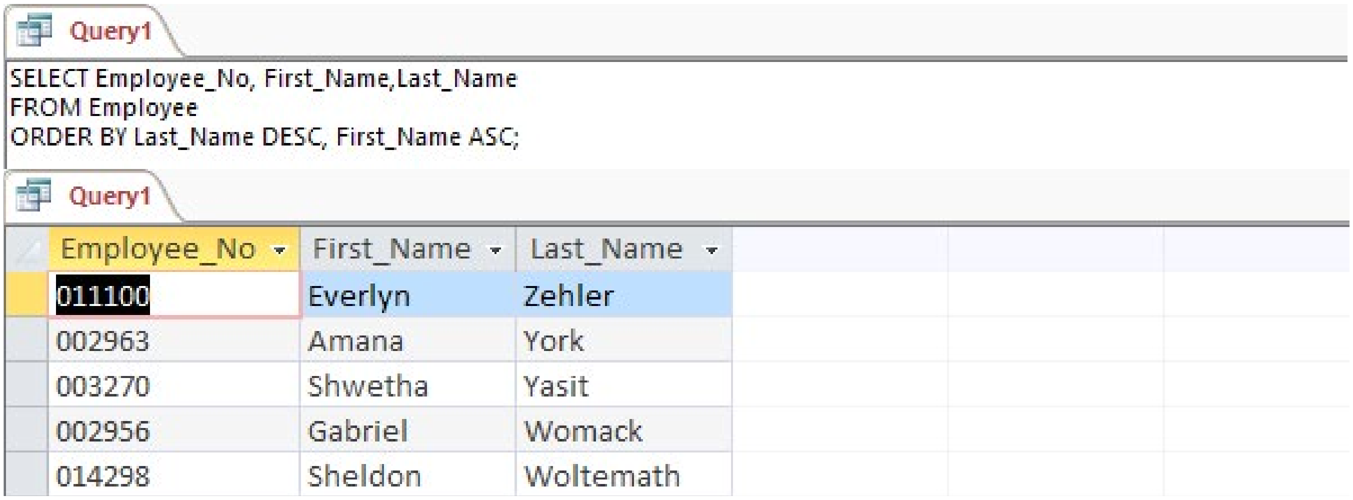
```
SELECT Employee_No, First_Name,Last_Name
FROM Employee
ORDER BY Last_Name DESC;
```

Employee_No	First_Name	Last_Name
011100	Everlyn	Zehler
002963	Amana	York
003270	Shwetha	Yasit
002956	Gabriel	Womack
014298	Sheldon	Woltemath
003042	Jonnie	Wolfson
003734	Jerrold	WITZENBURG
013604	Rosemarie	Wittke
010976	Worth	Winston

Figure 13.10

Use "ASC" if you want to specify to sort in Ascending order, but that is not usually necessary. The ORDER BY clause by default assumes you are sorting in an Ascending order. You can also perform sorts on multiple columns. A sort by last name in Descending order and then by first name in Ascending order can be done as follows:

16. *Type the code as in* **Figure 13.11** *and* **Run** *the query.*



```
SELECT Employee_No, First_Name,Last_Name
FROM Employee
ORDER BY Last_Name DESC, First_Name ASC;
```

Employee_No	First_Name	Last_Name
011100	Everlyn	Zehler
002963	Amana	York
003270	Shwetha	Yasit
002956	Gabriel	Womack
014298	Sheldon	Woltemath

Figure 13.11

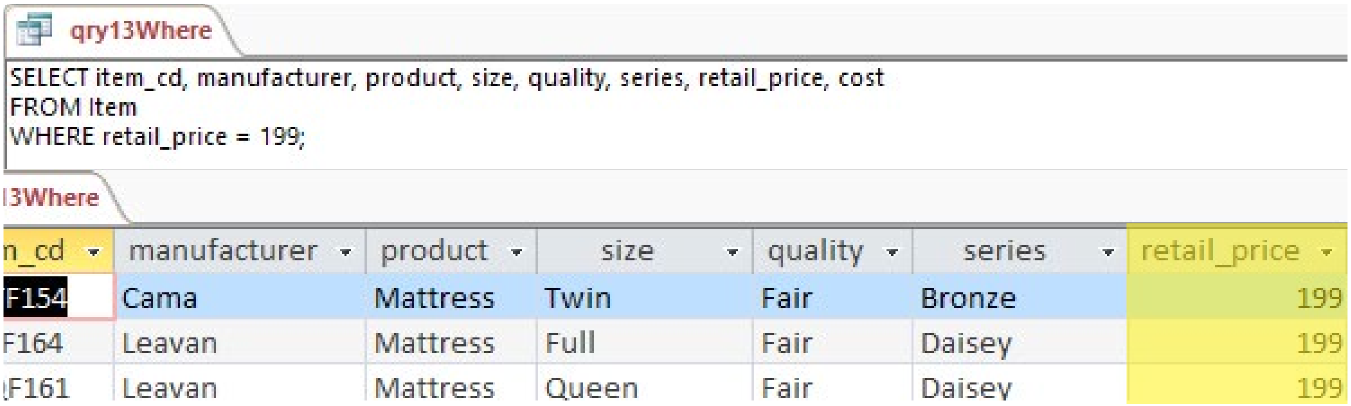
You should get the same results as in Figure 13.10, as all last names in the first few records are unique. Since there are no employees with the same last name, you don't see the Sort ASC on the first names.

17. **Save** *the query as* ***qry13Select_From_Order*** *and close the query.*

The WHERE Clause

In Access Design View, you can filter data in the Criteria line. With SQL, you filter data by using a WHERE clause within the SELECT statement. To work with the WHERE clause, we'll use the Item table. Let's suppose you want to find out all of the items in the Item table that have a retail price of $199.00. That's easy – just tell it you want the records where the retail_price=199.

1. **Create** *a new query called* ***qry13Where*** *and go to* **SQL View**.
2. *Type the code as shown in* **Figure 13.12** *and* **Run** *the query.*



```
SELECT item_cd, manufacturer, product, size, quality, series, retail_price, cost
FROM Item
WHERE retail_price = 199;
```

n_cd	manufacturer	product	size	quality	series	retail_price
F154	Cama	Mattress	Twin	Fair	Bronze	199
F164	Leavan	Mattress	Full	Fair	Daisey	199
F161	Leavan	Mattress	Queen	Fair	Daisey	199

Figure 13.12

Out of the 68 items in the Item table, only three have a retail price of $199.00.

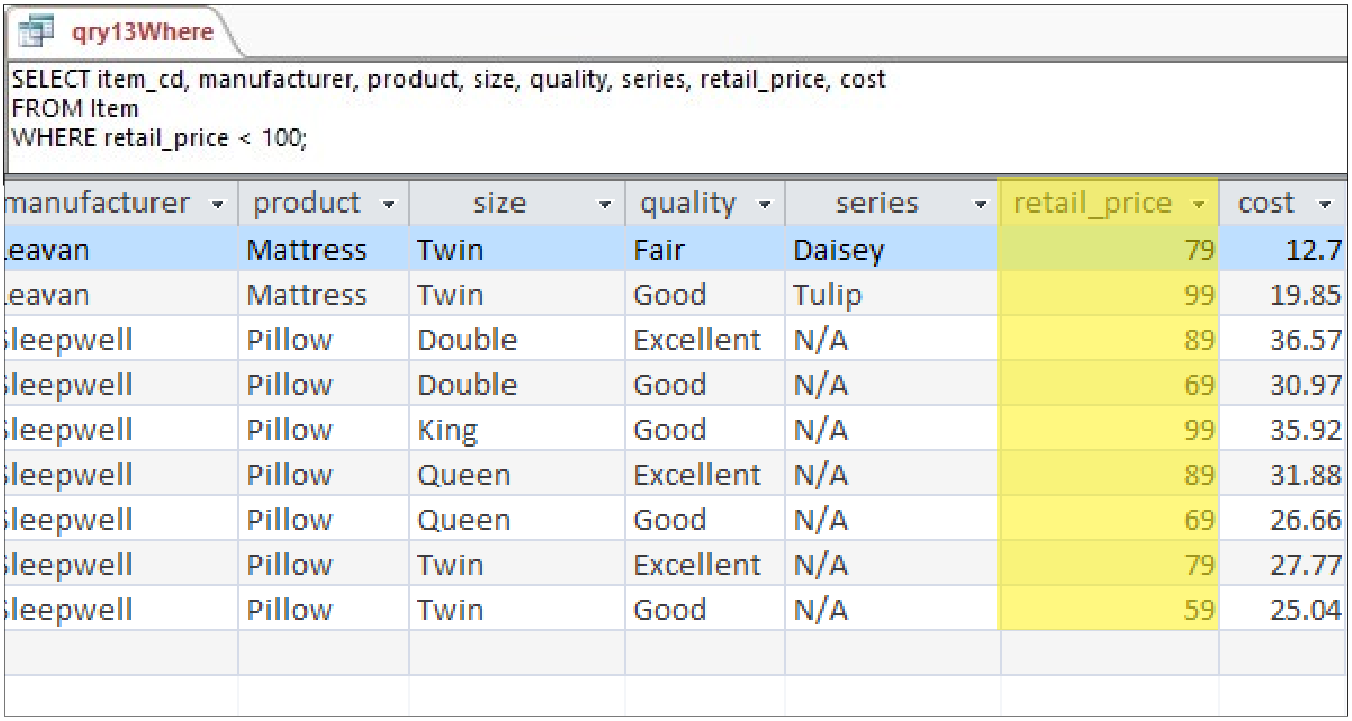
This exercise is a simple equality test (i.e., using the "=" sign), but SQL can do so much more than equality. There are numerous operators you can use in a WHERE clause in SQL. The table below lists those operators.

Operator	Description
=	*Equality*
<>	*Nonequality*
!=	*Nonequality*
<	*Less than*
<=	*Less than or equal to*
!<	*Not less than*
>	*Greater than*
>=	*Greater than or equal to*
!>	*Not greater than*
BETWEEN	*Between two values*
IS NULL	*Is a NULL value*

Not all of these operators are supported by all systems, but most work just fine. The only ones that I've found that sometimes cause problems are the ones with the exclamation point. However, there are almost always several ways to get around those through the use of simple logic.

Let's filter the database for all products with a retail price of less than $100.00.

3. *Edit the code as shown in* **Figure 13.13** *and* **Run** *the query.*



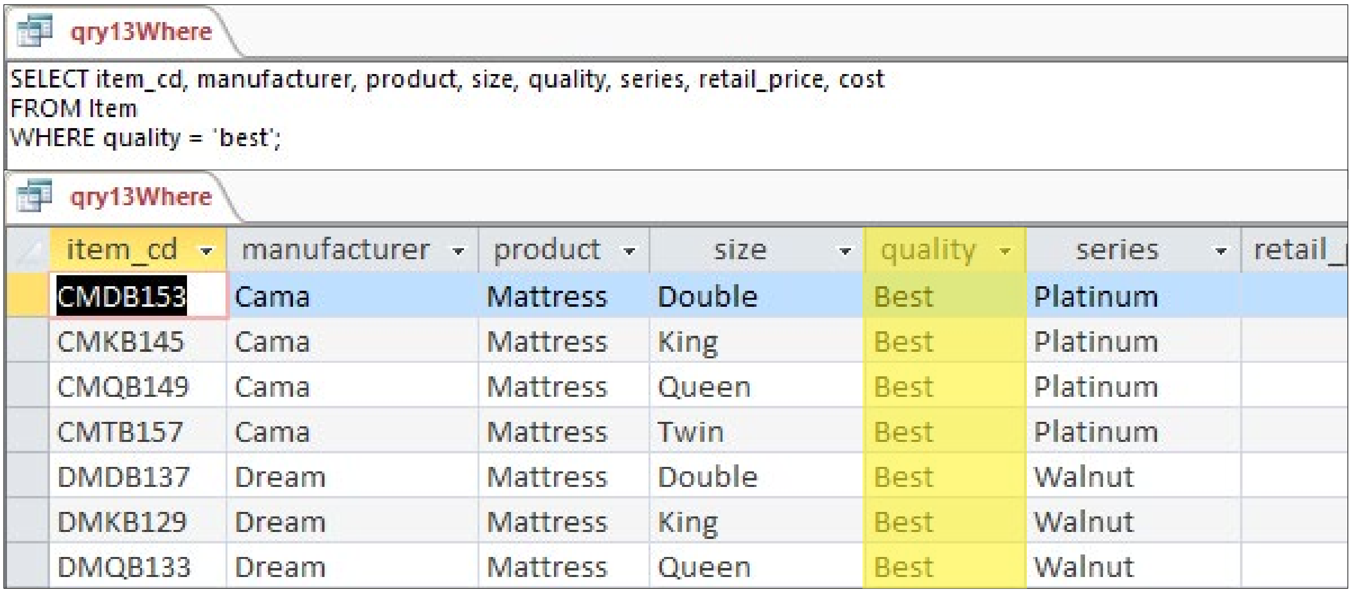
```
SELECT item_cd, manufacturer, product, size, quality, series, retail_price, cost
FROM Item
WHERE retail_price < 100;
```

manufacturer	product	size	quality	series	retail_price	cost
eavan	Mattress	Twin	Fair	Daisey	79	12.7
eavan	Mattress	Twin	Good	Tulip	99	19.85
leepwell	Pillow	Double	Excellent	N/A	89	36.57
leepwell	Pillow	Double	Good	N/A	69	30.97
leepwell	Pillow	King	Good	N/A	99	35.92
leepwell	Pillow	Queen	Excellent	N/A	89	31.88
leepwell	Pillow	Queen	Good	N/A	69	26.66
leepwell	Pillow	Twin	Excellent	N/A	79	27.77
leepwell	Pillow	Twin	Good	N/A	59	25.04

Figure 13.13

You can play around with using the different operators, as you've done in Excel and Access. You should get the picture now on how to filter for a single value. Let's now filter for all products that have a quality of Best.

4. *Edit the code as shown in* **Figure 13.14** *and* **Run** *the query.*



```
SELECT item_cd, manufacturer, product, size, quality, series, retail_price, cost
FROM Item
WHERE quality = 'best';
```

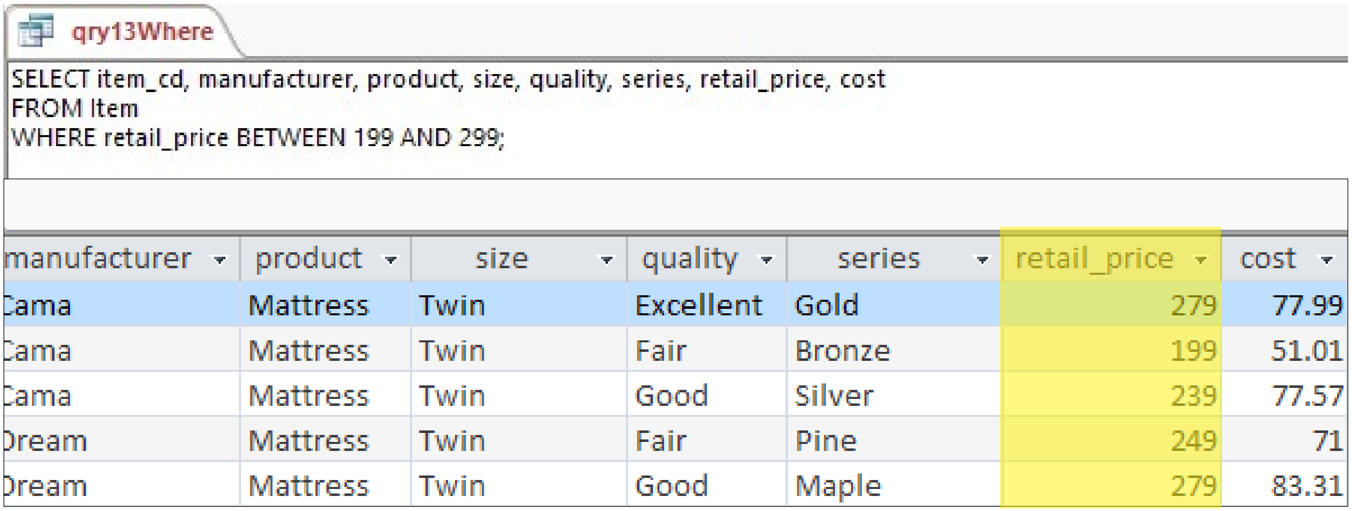
item_cd	manufacturer	product	size	quality	series	retail_
CMDB153	Cama	Mattress	Double	Best	Platinum	
CMKB145	Cama	Mattress	King	Best	Platinum	
CMQB149	Cama	Mattress	Queen	Best	Platinum	
CMTB157	Cama	Mattress	Twin	Best	Platinum	
DMDB137	Dream	Mattress	Double	Best	Walnut	
DMKB129	Dream	Mattress	King	Best	Walnut	
DMQB133	Dream	Mattress	Queen	Best	Walnut	

Figure 13.14

You should get a list of 12 records. Notice that the word best is surrounded by apostrophes. In the Access version of SQL, you can use an apostrophe or quote marks to surround text. As you remember, Access will surround a text string with quote marks in Design View. Most ANSI SQL code is written with text values surrounded by apostrophes. From here on out, I will use an apostrophe, as is required with ANSI SQL. Numeric values do not require an apostrophe.

You can use the BETWEEN operator to filter values that are between two values. Let's filter the Item table for all products that have a retail price between $199 and $299. Note that it will include those retail prices that equal $199 and $299 when you use the BETWEEN operator.

5. *Edit the code as shown in* **Figure 13.15** *and* **Run** *the query.*



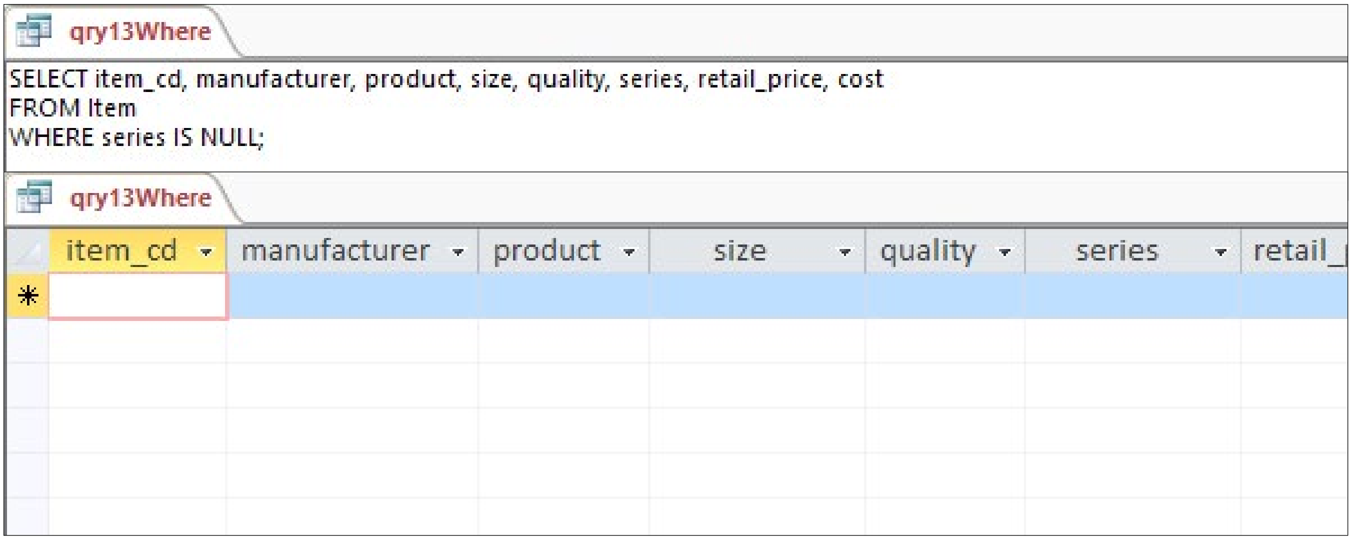
```
SELECT item_cd, manufacturer, product, size, quality, series, retail_price, cost
FROM Item
WHERE retail_price BETWEEN 199 AND 299;
```

manufacturer	product	size	quality	series	retail_price	cost
Cama	Mattress	Twin	Excellent	Gold	279	77.99
Cama	Mattress	Twin	Fair	Bronze	199	51.01
Cama	Mattress	Twin	Good	Silver	239	77.57
Dream	Mattress	Twin	Fair	Pine	249	71
Dream	Mattress	Twin	Good	Maple	279	83.31

Figure 13.15

Similarly, you can use the IS NULL operator to check to see if a field in a table or query contains a NULL value. You should not confuse a NULL value with a zero value or spaces. A NULL value has no data in it, whereas a zero or spaces are actually characters. Let's see if there are any NULL values in the series field of the Item table.

6. *Edit the code as shown in* **Figure 13.16** *and* **Run** *the query.*



```
SELECT item_cd, manufacturer, product, size, quality, series, retail_price, cost
FROM Item
WHERE series IS NULL;
```

Figure 13.16

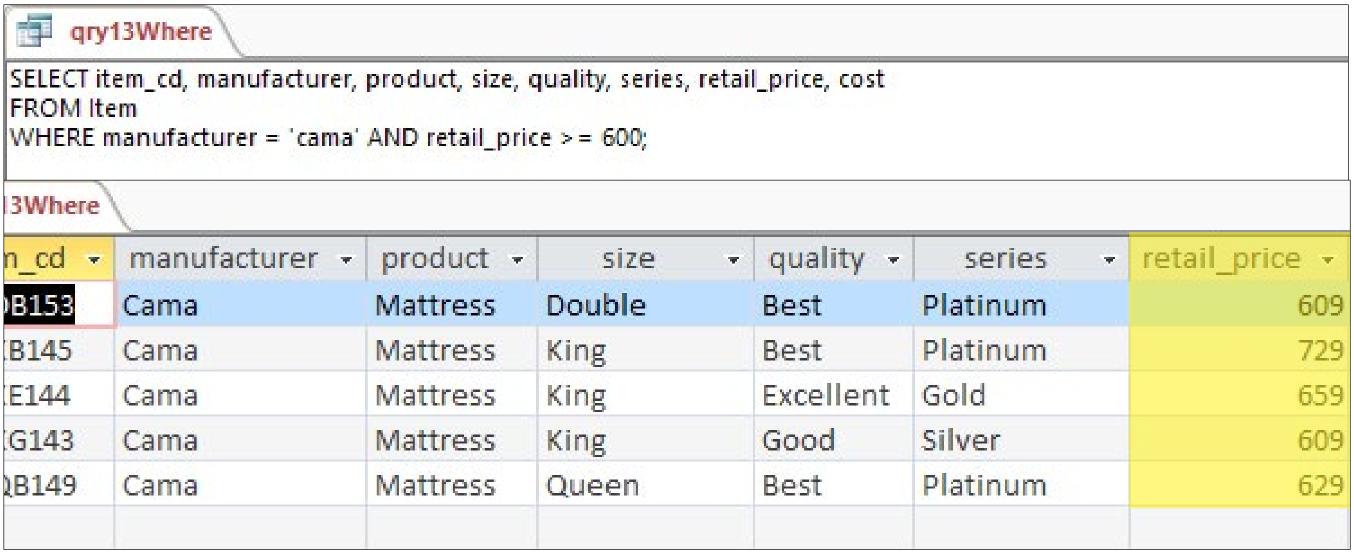
No records were returned, meaning there are no records with NULL values in the Series field.

Review Questions*: It is now time to complete the hands-on Review Questions. Log on to www.ExcelCEO.com with your Email and Password, click on the* ***Access 2016 and SQL Review Questions, Chapter 13, Section 1 of 2*** *option and complete the review questions.*

The AND Operator

Many times, you will need to include more than one criteria in the WHERE clause. We did that in many examples in the Access portion of the course. Suppose you want to run a query on the item table to find the items where the manufacturer is Cama and the retail price is greater than or equal to $600. Instead of writing two WHERE clauses (which wouldn't work anyway), you separate the arguments by using the AND operator.

7. *Open* **qry13Where** *(if you closed it) and edit the code as shown in* **Figure 13.17***, then* **Run** *the query.*



```
SELECT item_cd, manufacturer, product, size, quality, series, retail_price, cost
FROM Item
WHERE manufacturer = 'cama' AND retail_price >= 600;
```

n_cd	manufacturer	product	size	quality	series	retail_price
B153	Cama	Mattress	Double	Best	Platinum	609
B145	Cama	Mattress	King	Best	Platinum	729
E144	Cama	Mattress	King	Excellent	Gold	659
G143	Cama	Mattress	King	Good	Silver	609
B149	Cama	Mattress	Queen	Best	Platinum	629

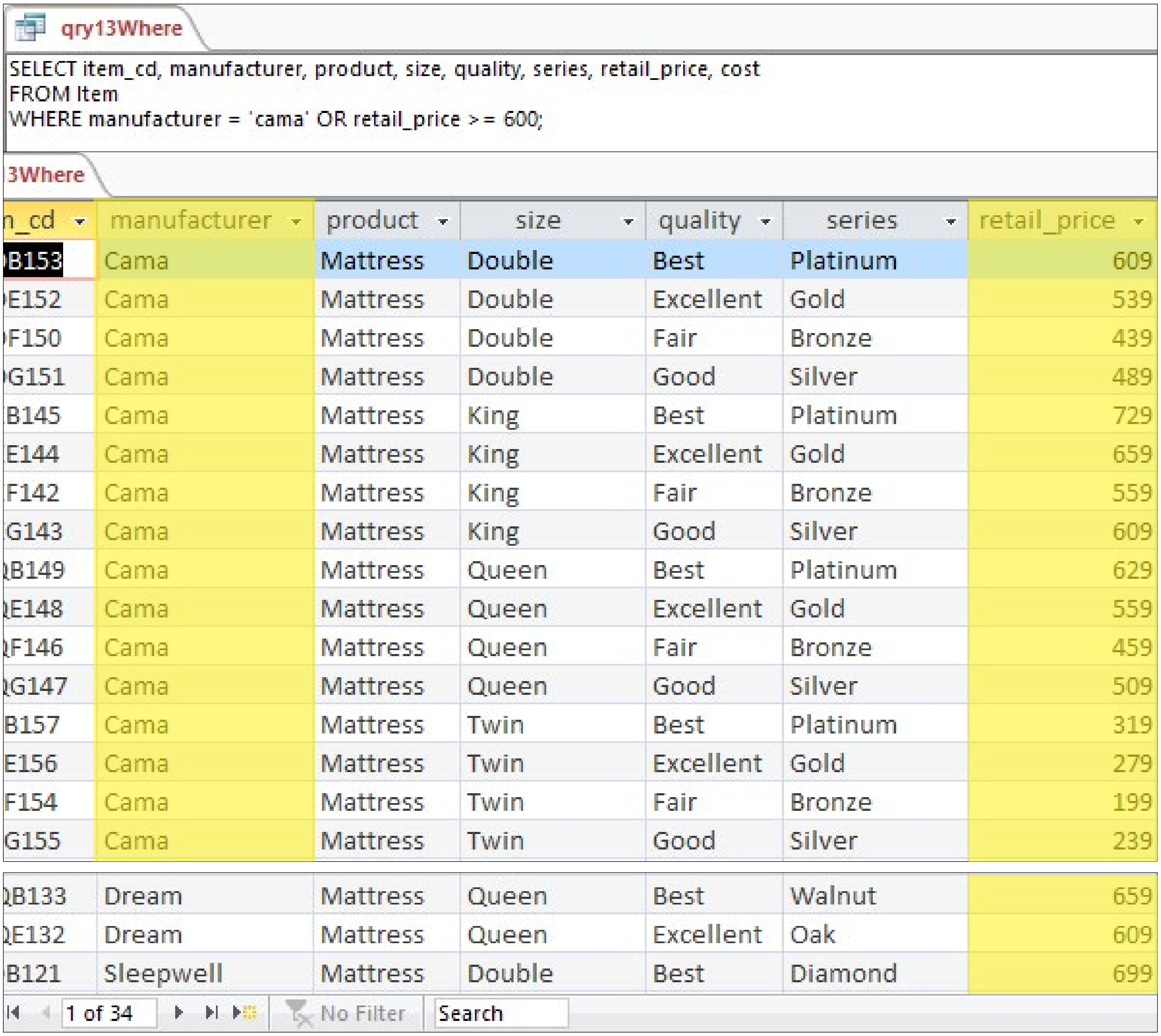
Figure 13.17

Five records were returned by this code. Remember that in this example, the WHERE clause has two conditions that are separated by the AND operator. The AND operator instructs the code to make sure that both conditions are met.

The OR Operator

There is power in words. That is especially true in SQL programming. One word or even one character can totally change the output; therefore, you must continually check and recheck your results. The OR operator in SQL is the opposite of the AND operator. The OR operator instructs SQL to return records that meet EITHER of the conditions specified in the WHERE clause. Let's use our previous example and change the AND to OR and see the results.

8. *Edit the code as shown in* **Figure 13.18** *and* **Run** *the query.*



```
SELECT item_cd, manufacturer, product, size, quality, series, retail_price, cost
FROM Item
WHERE manufacturer = 'cama' OR retail_price >= 600;
```

n_cd	manufacturer	product	size	quality	series	retail_price
B153	Cama	Mattress	Double	Best	Platinum	609
E152	Cama	Mattress	Double	Excellent	Gold	539
F150	Cama	Mattress	Double	Fair	Bronze	439
G151	Cama	Mattress	Double	Good	Silver	489
B145	Cama	Mattress	King	Best	Platinum	729
E144	Cama	Mattress	King	Excellent	Gold	659
F142	Cama	Mattress	King	Fair	Bronze	559
G143	Cama	Mattress	King	Good	Silver	609
B149	Cama	Mattress	Queen	Best	Platinum	629
E148	Cama	Mattress	Queen	Excellent	Gold	559
F146	Cama	Mattress	Queen	Fair	Bronze	459
G147	Cama	Mattress	Queen	Good	Silver	509
B157	Cama	Mattress	Twin	Best	Platinum	319
E156	Cama	Mattress	Twin	Excellent	Gold	279
F154	Cama	Mattress	Twin	Fair	Bronze	199
G155	Cama	Mattress	Twin	Good	Silver	239
B133	Dream	Mattress	Queen	Best	Walnut	659
E132	Dream	Mattress	Queen	Excellent	Oak	609
B121	Sleepwell	Mattress	Double	Best	Diamond	699

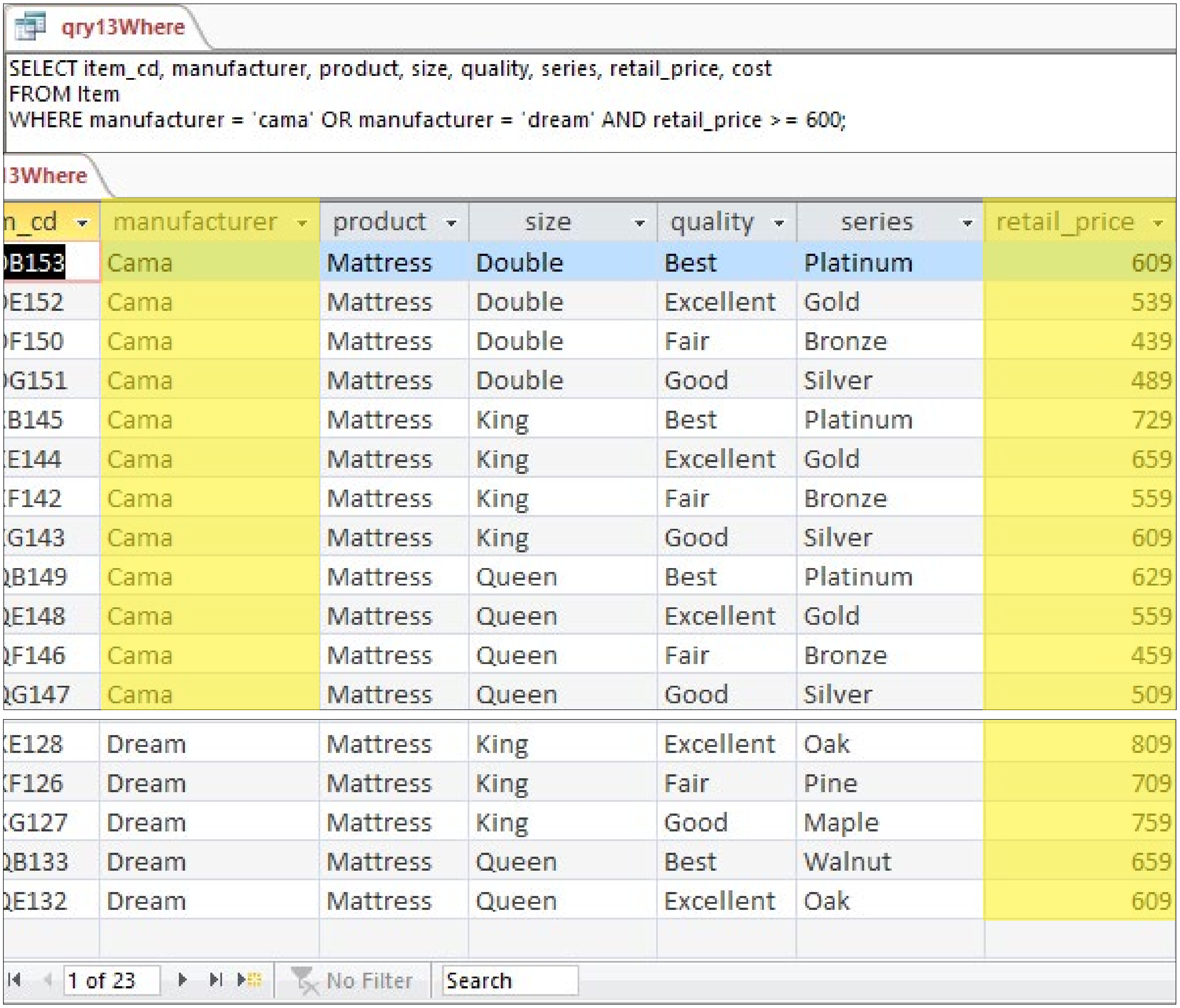
Figure 13.18

This time, there were 34 records returned. This dataset contains all records in the Item table where the manufacturer is Cama. Additionally, it contains all of the records where the retail price is greater than or equal to $600.

Order of Evaluation

You can use the AND and OR operators to perform some very complex queries, but you must be careful when using multiple AND and OR operators. As with writing formulas in Excel or Access, SQL also has an order of evaluation. Let's continue to build on the previous example but this time we want all of the items where the manufacturer is either Cama or Dream and the retail price is equal to or greater than $600.

9. *Edit the code as shown in* **Figure 13.19** *and* **Run** *the query.*



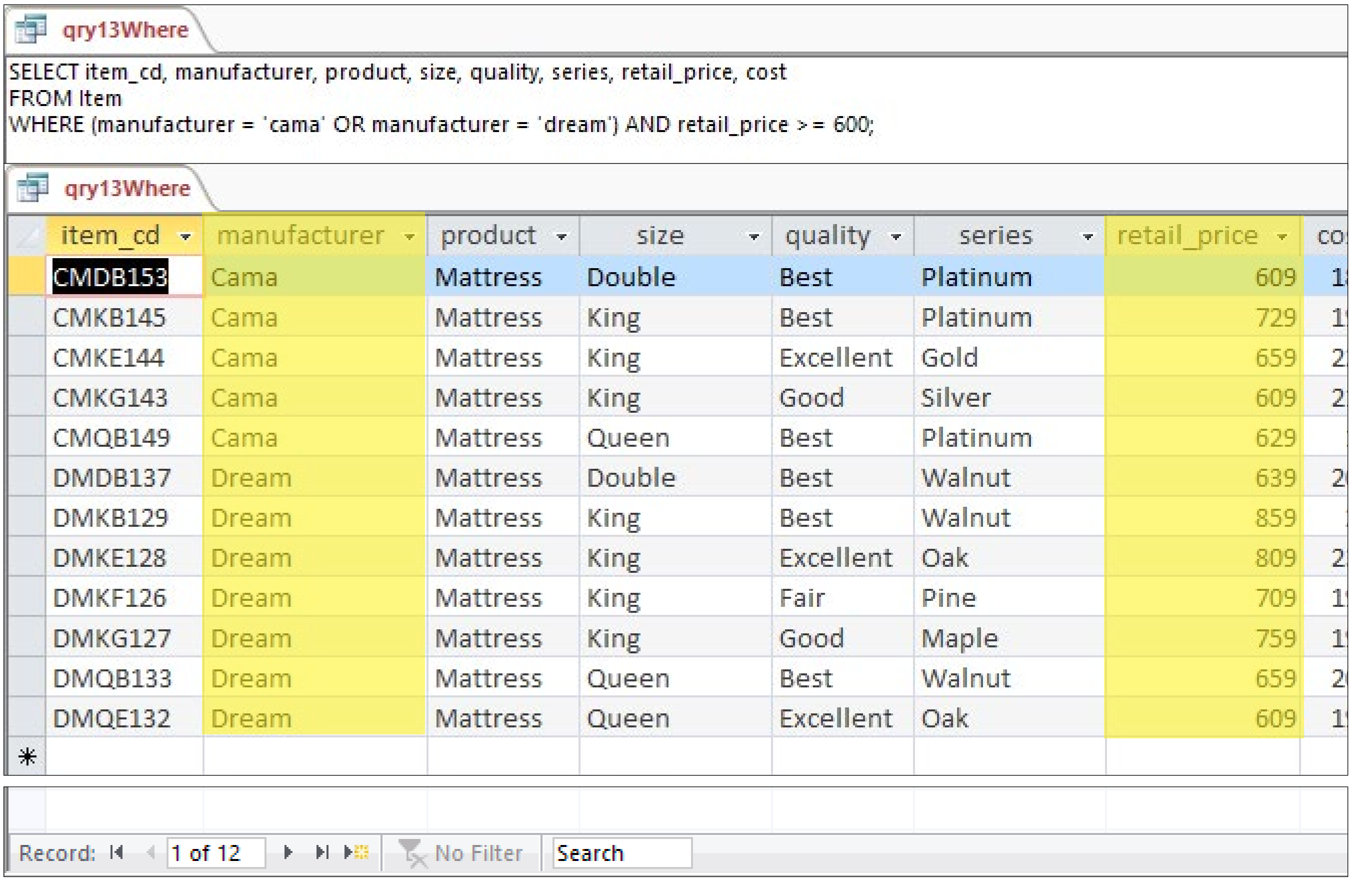
```
SELECT item_cd, manufacturer, product, size, quality, series, retail_price, cost
FROM Item
WHERE manufacturer = 'cama' OR manufacturer = 'dream' AND retail_price >= 600;
```

n_cd	manufacturer	product	size	quality	series	retail_price
B153	Cama	Mattress	Double	Best	Platinum	609
E152	Cama	Mattress	Double	Excellent	Gold	539
F150	Cama	Mattress	Double	Fair	Bronze	439
G151	Cama	Mattress	Double	Good	Silver	489
B145	Cama	Mattress	King	Best	Platinum	729
E144	Cama	Mattress	King	Excellent	Gold	659
F142	Cama	Mattress	King	Fair	Bronze	559
G143	Cama	Mattress	King	Good	Silver	609
B149	Cama	Mattress	Queen	Best	Platinum	629
E148	Cama	Mattress	Queen	Excellent	Gold	559
F146	Cama	Mattress	Queen	Fair	Bronze	459
G147	Cama	Mattress	Queen	Good	Silver	509
E128	Dream	Mattress	King	Excellent	Oak	809
F126	Dream	Mattress	King	Fair	Pine	709
G127	Dream	Mattress	King	Good	Maple	759
B133	Dream	Mattress	Queen	Best	Walnut	659
E132	Dream	Mattress	Queen	Excellent	Oak	609

Figure 13.19

In this dataset, you see that it contains only the manufacturers Cama and Dream, but there are many retail prices that are under $600. What happened? It happened because SQL evaluated the argument manufacturer='cama' separately from manufacturer='dream' AND retail_price<=600. Basically, it made the AND operator take precedence. It evaluated the manufacturer='cama' statement first, then it evaluated the manufacturer='dream' AND retail_price<=600 statement separately. The AND operator takes precedence over the OR operator. You can correct the order of evaluation by placing the OR statement in parentheses.

10. *Edit the code as shown in* **Figure 13.20** *and* **Run** *the query.*



```
SELECT item_cd, manufacturer, product, size, quality, series, retail_price, cost
FROM Item
WHERE (manufacturer = 'cama' OR manufacturer = 'dream') AND retail_price >= 600;
```

item_cd	manufacturer	product	size	quality	series	retail_price	co
CMDB153	Cama	Mattress	Double	Best	Platinum	609	1
CMKB145	Cama	Mattress	King	Best	Platinum	729	1
CMKE144	Cama	Mattress	King	Excellent	Gold	659	2
CMKG143	Cama	Mattress	King	Good	Silver	609	2
CMQB149	Cama	Mattress	Queen	Best	Platinum	629	
DMDB137	Dream	Mattress	Double	Best	Walnut	639	2
DMKB129	Dream	Mattress	King	Best	Walnut	859	
DMKE128	Dream	Mattress	King	Excellent	Oak	809	2
DMKF126	Dream	Mattress	King	Fair	Pine	709	1
DMKG127	Dream	Mattress	King	Good	Maple	759	1
DMQB133	Dream	Mattress	Queen	Best	Walnut	659	2
DMQE132	Dream	Mattress	Queen	Excellent	Oak	609	1
*							

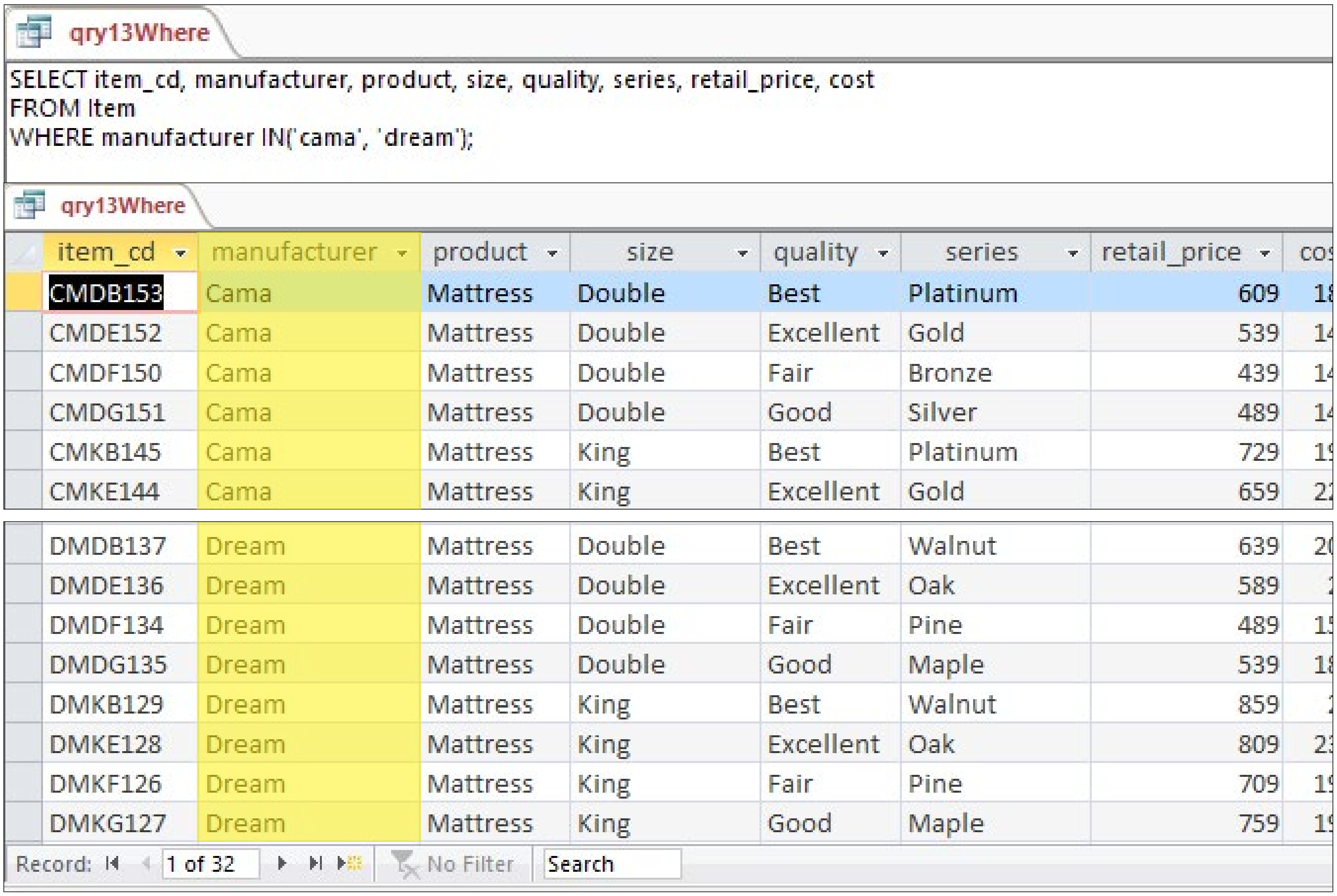
Figure 13.20

This code returned only 12 records, which is the correct answer for what you wanted to do. The only difference between the code in Figures 13.19 and 13.20 is the placement of the parentheses.

The IN Operator

The IN operator is used when you want to include a list of conditions to be used in the query, any of which are included in the results. You simply include all of the values, separated by commas and enclosed with parentheses. Let's suppose you want to simplify our example in Figure 13.19 to include all items that have a manufacturer of Cama or Dream. You could do it with an OR operator, but you would have to type the field name (manufacturer) each time. With an IN operator, you type the name of the field once followed by the values.

11. *Edit the code as shown in* **Figure 13.21** *and* **Run** *the query.*



```
SELECT item_cd, manufacturer, product, size, quality, series, retail_price, cost
FROM Item
WHERE manufacturer IN('cama', 'dream');
```

item_cd	manufacturer	product	size	quality	series	retail_price	cos
CMDB153	Cama	Mattress	Double	Best	Platinum	609	18
CMDE152	Cama	Mattress	Double	Excellent	Gold	539	14
CMDF150	Cama	Mattress	Double	Fair	Bronze	439	14
CMDG151	Cama	Mattress	Double	Good	Silver	489	14
CMKB145	Cama	Mattress	King	Best	Platinum	729	19
CMKE144	Cama	Mattress	King	Excellent	Gold	659	22
DMDB137	Dream	Mattress	Double	Best	Walnut	639	20
DMDE136	Dream	Mattress	Double	Excellent	Oak	589	2
DMDF134	Dream	Mattress	Double	Fair	Pine	489	15
DMDG135	Dream	Mattress	Double	Good	Maple	539	18
DMKB129	Dream	Mattress	King	Best	Walnut	859	2
DMKE128	Dream	Mattress	King	Excellent	Oak	809	23
DMKF126	Dream	Mattress	King	Fair	Pine	709	19
DMKG127	Dream	Mattress	King	Good	Maple	759	19

Figure 13.21

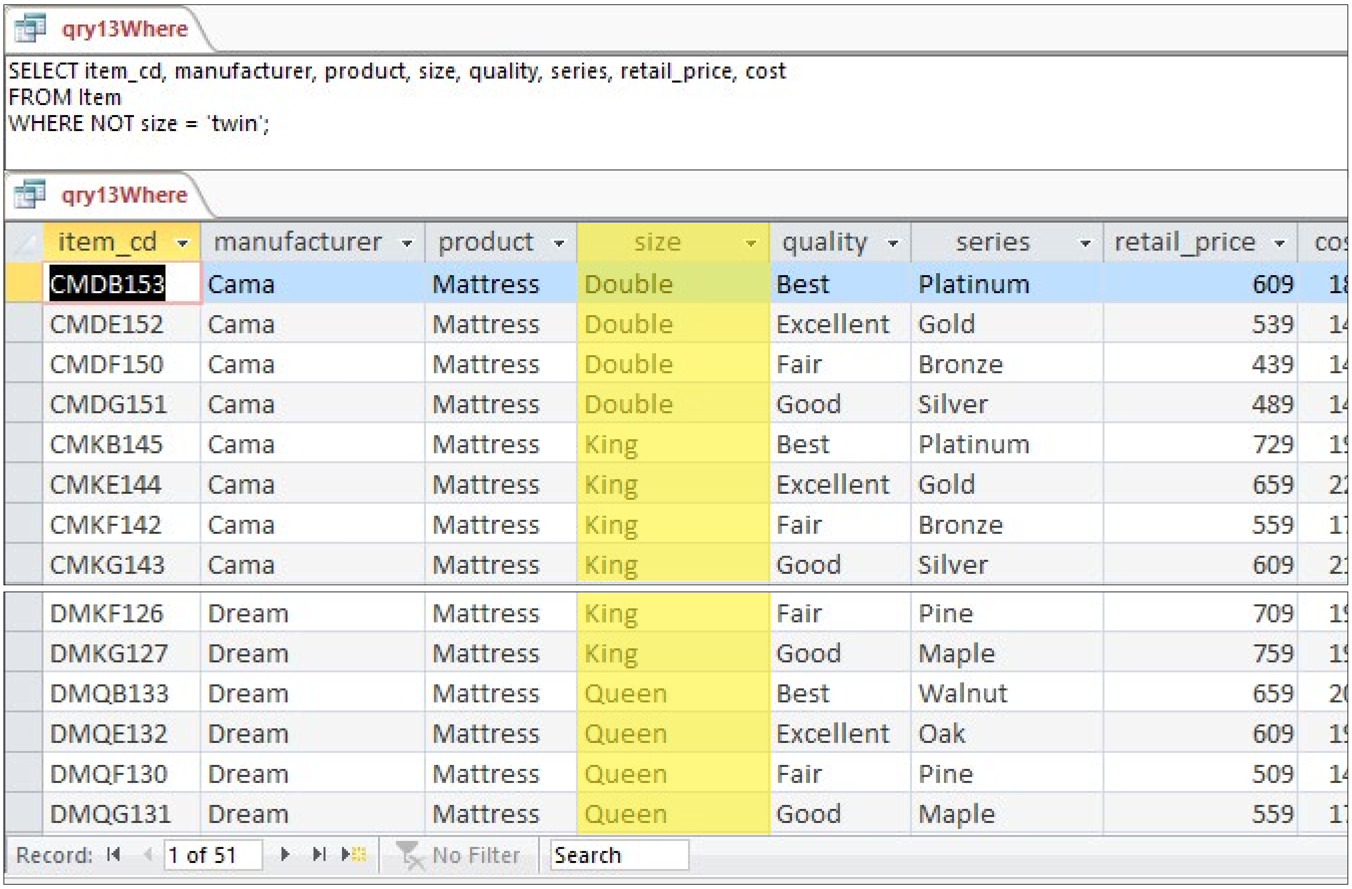
You now get a list of 32 items, which is ALL of the items that have a manufacturer of either Cama or Dream. Other advantages of using the IN operator as opposed to an OR operator include:

- The list of values in an IN operator are easier to read (rather than having each value separated by an OR operator),
- The order of evaluation is easier when IN is used,
- Most of the time, IN operators work faster than an OR operator (particularly when there is a long list of values), and
- An IN operator can contain another SELECT statement, with which you can build some powerful WHERE clauses. These are called ***subqueries***.

The NOT Operator

There is one purpose and one purpose only in the NOT operator's life – to negate a condition. NOT is used only when you don't want to include something. Normally, NOT is used with other operators in a WHERE statement, but it can be used by itself as well. Let's suppose you want to see a list of records from the Item table where the size is not Twin. That's easy:

12. *Edit the code as shown in* **Figure 13.22** *and* **Run** *the query.*



qry13Where

```
SELECT item_cd, manufacturer, product, size, quality, series, retail_price, cost
FROM Item
WHERE NOT size = 'twin';
```

qry13Where

item_cd	manufacturer	product	size	quality	series	retail_price	cos
CMDB153	Cama	Mattress	Double	Best	Platinum	609	18
CMDE152	Cama	Mattress	Double	Excellent	Gold	539	14
CMDF150	Cama	Mattress	Double	Fair	Bronze	439	14
CMDG151	Cama	Mattress	Double	Good	Silver	489	14
CMKB145	Cama	Mattress	King	Best	Platinum	729	19
CMKE144	Cama	Mattress	King	Excellent	Gold	659	22
CMKF142	Cama	Mattress	King	Fair	Bronze	559	17
CMKG143	Cama	Mattress	King	Good	Silver	609	21
DMKF126	Dream	Mattress	King	Fair	Pine	709	19
DMKG127	Dream	Mattress	King	Good	Maple	759	19
DMQB133	Dream	Mattress	Queen	Best	Walnut	659	20
DMQE132	Dream	Mattress	Queen	Excellent	Oak	609	19
DMQF130	Dream	Mattress	Queen	Fair	Pine	509	14
DMQG131	Dream	Mattress	Queen	Good	Maple	559	1

Figure 13.22

Probably the more common syntax is to write WHERE size<>'twin', meaning where size is not equal to twin. Either one will work in most simple queries, but I prefer to use the <> syntax. However, in more complex queries you may have to use the NOT operator.

Wildcard Operators

Do you remember playing poker and the dealer calls for five card stud with sevens and jacks wild? That means that you can use a seven or a jack for any card you want. The same concept holds true when using wildcard characters. You've already been exposed to one wildcard operator in SQL. At the beginning of this chapter, I had you write the SELECT * FROM Employee. The asterisk (*) is a wildcard character, just like we used in formulas in Access. When you ran that code, it gave you ALL of the records from the Employee table. A wildcard is simply a character used to match part(s) of a value. A wildcard character can be used by itself or with other characters to search for a value.

The LIKE Operator

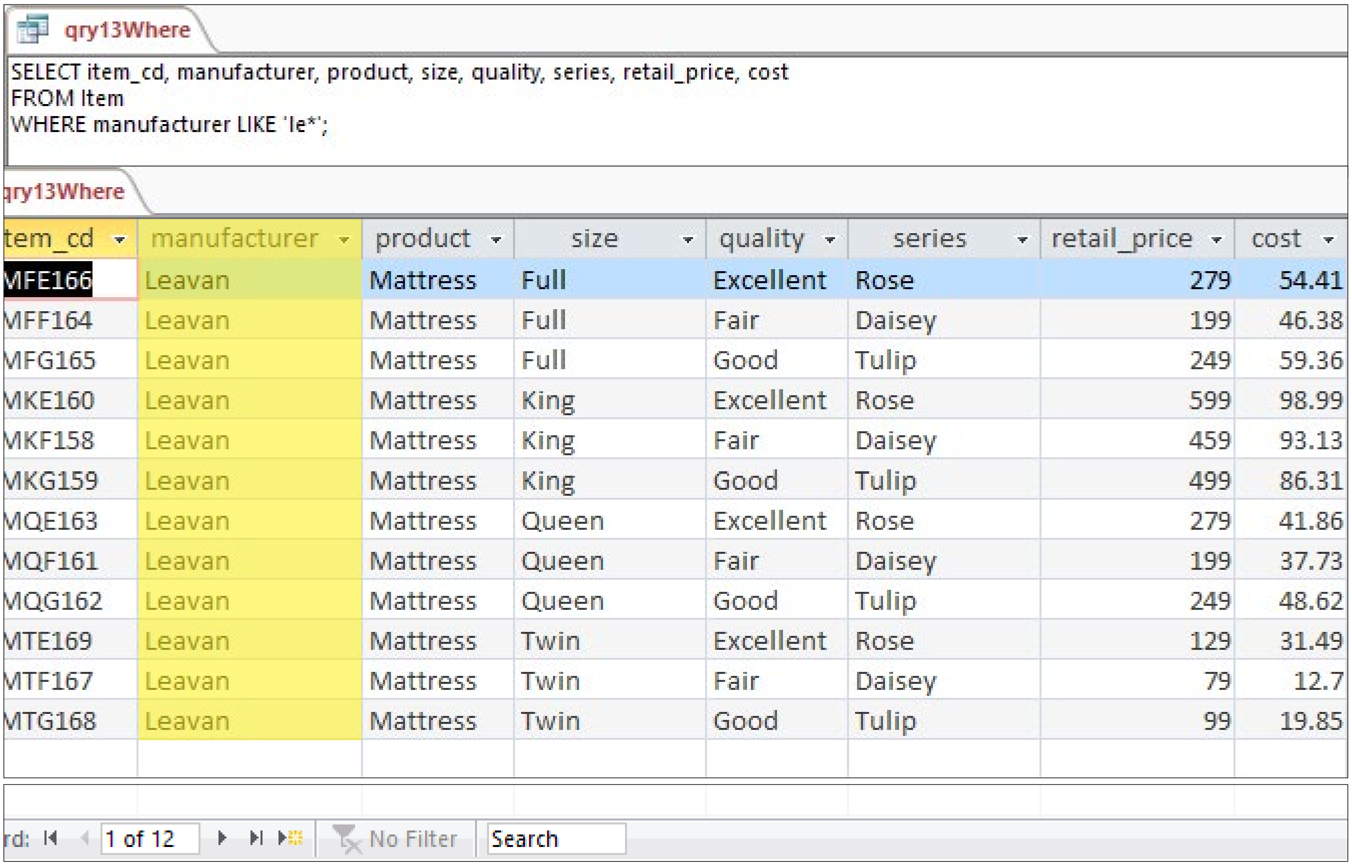
Whenever you use a wildcard character to search for a value in a WHERE clause, you must use the LIKE operator. The LIKE operator tells SQL to search for the pattern that follows where the wildcard character could stand for any character in the record. It comes in very handy for people who can't spell very well.

The Asterisk (*) and Percent Sign (%) Wildcards

Probably the most frequently used wildcards are the asterisk (*) and percent sign (%). So far in the SQL portion of this course, all of the SQL code has been ANSI SQL. As we're using Access SQL, I'm going to deviate from that a bit here. In the following examples, you will use the asterisk as the wildcard, but keep in mind when you are using ANSI SQL or Transact SQL, you will have to use the percent sign.

Placing an asterisk (or the percent sign in ANSI SQL) within a search string means that you want to match the number of occurrences of any character. Let's suppose in our example that you want to see all of the items from the manufacturer Leaven, but you can't remember if it is spelled Leaven, or Leven, or Levan. You can use the asterisk in the WHERE statement to help you out.

13. *Edit the code as shown in* **Figure 13.23** *and* **Run** *the query.*



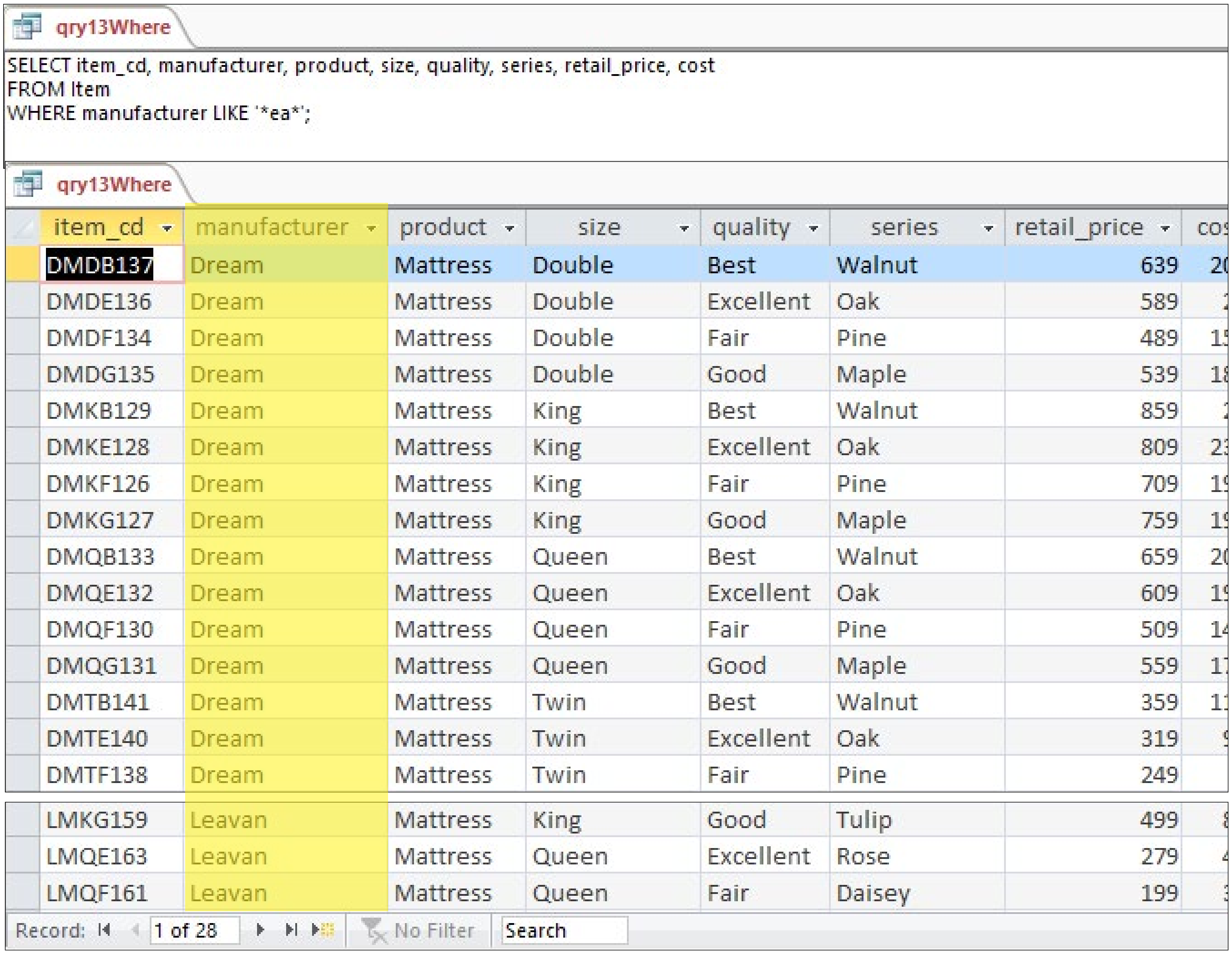
```
SELECT item_cd, manufacturer, product, size, quality, series, retail_price, cost
FROM Item
WHERE manufacturer LIKE 'le*';
```

tem_cd	manufacturer	product	size	quality	series	retail_price	cost
MFE166	Leavan	Mattress	Full	Excellent	Rose	279	54.41
MFF164	Leavan	Mattress	Full	Fair	Daisey	199	46.38
MFG165	Leavan	Mattress	Full	Good	Tulip	249	59.36
MKE160	Leavan	Mattress	King	Excellent	Rose	599	98.99
MKF158	Leavan	Mattress	King	Fair	Daisey	459	93.13
MKG159	Leavan	Mattress	King	Good	Tulip	499	86.31
MQE163	Leavan	Mattress	Queen	Excellent	Rose	279	41.86
MQF161	Leavan	Mattress	Queen	Fair	Daisey	199	37.73
MQG162	Leavan	Mattress	Queen	Good	Tulip	249	48.62
MTE169	Leavan	Mattress	Twin	Excellent	Rose	129	31.49
MTF167	Leavan	Mattress	Twin	Fair	Daisey	79	12.7
MTG168	Leavan	Mattress	Twin	Good	Tulip	99	19.85

Figure 13.23

You should get a record set of 12 records, which represents all of the items that are manufactured by Leaven. You can use the asterisk (or percent sign in ANSI SQL) at any place in the search string. Let's suppose that you remember that the name of the manufacturer has an "ea" somewhere in the name, but that's all you can remember. You can place the asterisk before and after the "ea" string and perform the search.

14. *Edit the code as shown in* **Figure 11.24** *and* **Run** *the query.*



```
SELECT item_cd, manufacturer, product, size, quality, series, retail_price, cost
FROM Item
WHERE manufacturer LIKE '*ea*';
```

item_cd	manufacturer	product	size	quality	series	retail_price	cos
DMDB137	Dream	Mattress	Double	Best	Walnut	639	20
DMDE136	Dream	Mattress	Double	Excellent	Oak	589	2
DMDF134	Dream	Mattress	Double	Fair	Pine	489	15
DMDG135	Dream	Mattress	Double	Good	Maple	539	18
DMKB129	Dream	Mattress	King	Best	Walnut	859	2
DMKE128	Dream	Mattress	King	Excellent	Oak	809	23
DMKF126	Dream	Mattress	King	Fair	Pine	709	19
DMKG127	Dream	Mattress	King	Good	Maple	759	19
DMQB133	Dream	Mattress	Queen	Best	Walnut	659	20
DMQE132	Dream	Mattress	Queen	Excellent	Oak	609	19
DMQF130	Dream	Mattress	Queen	Fair	Pine	509	14
DMQG131	Dream	Mattress	Queen	Good	Maple	559	17
DMTB141	Dream	Mattress	Twin	Best	Walnut	359	11
DMTE140	Dream	Mattress	Twin	Excellent	Oak	319	9
DMTF138	Dream	Mattress	Twin	Fair	Pine	249	
LMKG159	Leavan	Mattress	King	Good	Tulip	499	8
LMQE163	Leavan	Mattress	Queen	Excellent	Rose	279	4
LMQF161	Leavan	Mattress	Queen	Fair	Daisey	199	3

Figure 13.24

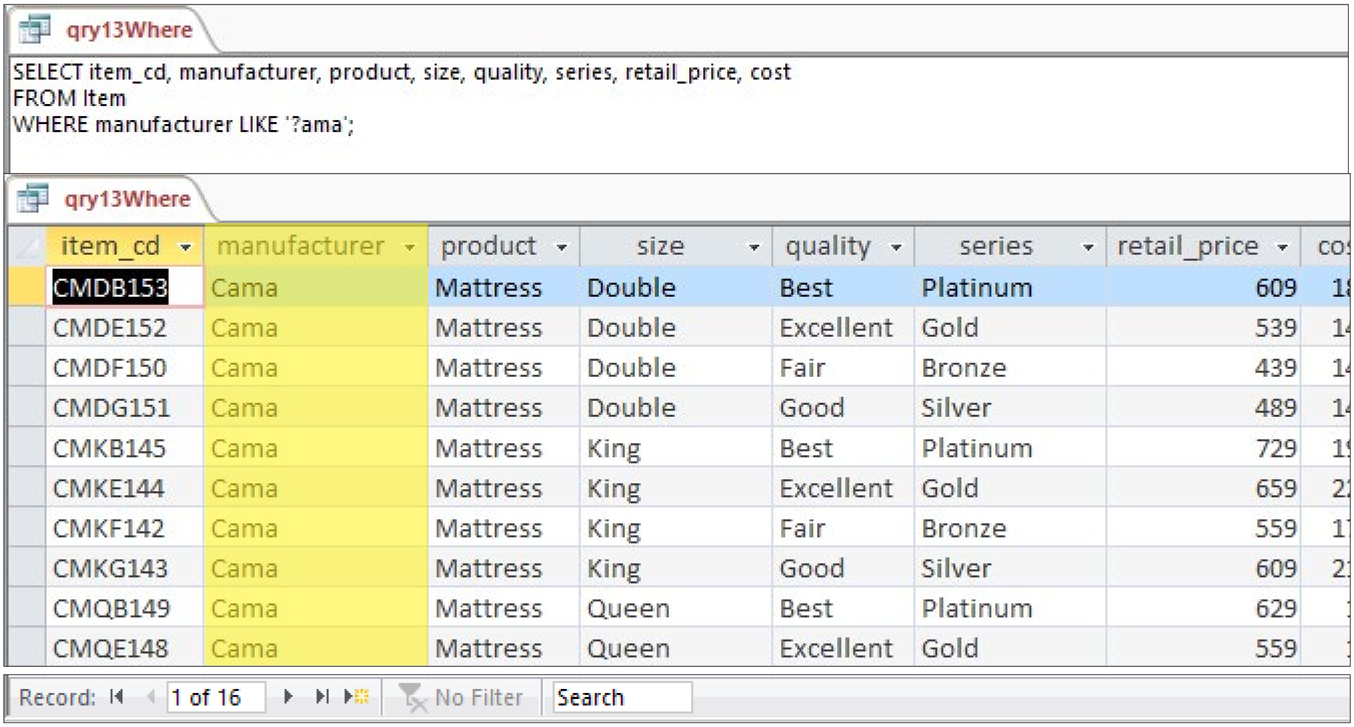
Oops. Now you have all of the manufacturers that have the string "ea" somewhere in their name, which includes Dream and Leaven. I once heard that if God wanted to destroy the world, all He would have to do is give everyone everything they prayed for. This is the text string you wanted, so this is what it gave to you, whether you like it or not. As such, you must be careful when using wildcards.

> ***Tip****: One issue we discussed in the Excel course is the trailing spaces that are contained in some data. Sometimes,* **DBMSs** *will include spaces after the last character of the text string. If the field calls for 20 characters and the text string contains only 12 characters, sometimes it will include eight spaces to take up the remaining characters. You can use the asterisk or percent sign at the end of the search string to account for the spaces. You can also use the* **RTRIM()** *function in* **ANSI SQL** *to take out the trailing spaces. We will discuss the RTRIM() function in the next chapter.*

The Question Mark (?) and Underscore (_) Operators

Sometimes you want to search for a string where you are unsure of just one character. The question mark in Access SQL (or the underscore in ANSI SQL) does exactly that – it matches a single character. Let's suppose that you want to search for the items of a certain manufacturer, but you can't remember if the name is Bama, Cama, or Lama.

15. Edit the code as shown in **Figure 13.25** *and* **Run** *the query.*



```
SELECT item_cd, manufacturer, product, size, quality, series, retail_price, cost
FROM Item
WHERE manufacturer LIKE '?ama';
```

item_cd	manufacturer	product	size	quality	series	retail_price	cos
CMDB153	Cama	Mattress	Double	Best	Platinum	609	1
CMDE152	Cama	Mattress	Double	Excellent	Gold	539	14
CMDF150	Cama	Mattress	Double	Fair	Bronze	439	14
CMDG151	Cama	Mattress	Double	Good	Silver	489	14
CMKB145	Cama	Mattress	King	Best	Platinum	729	1
CMKE144	Cama	Mattress	King	Excellent	Gold	659	2
CMKF142	Cama	Mattress	King	Fair	Bronze	559	1
CMKG143	Cama	Mattress	King	Good	Silver	609	2
CMQB149	Cama	Mattress	Queen	Best	Platinum	629	
CMQE148	Cama	Mattress	Queen	Excellent	Gold	559	

Figure 13.25

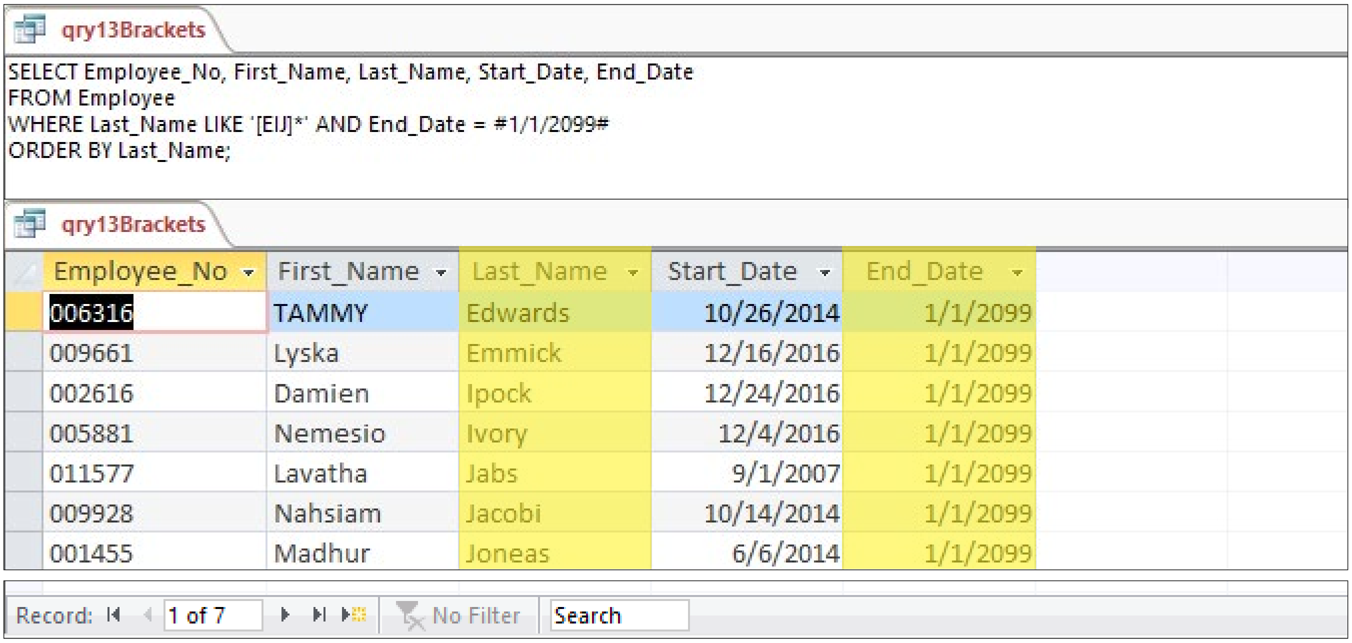
Here you get a record set of all of the items that are manufactured by Cama. Just remember that the asterisk (or percent sign in ANSI SQL) is used for any number of characters whereas the question mark (or underscore in ANSI SQL) searches for a single character.

16. **Save** *and close* **qry13Where**.

The Brackets ([]) Operator

The brackets operator ([]) is used to specify a set of characters where any of them could be used to match a character in a certain position. The brackets are not supported by all DBMSs, but it is supported by Access and SQL Server. Let's create a new query based on the Employee table and search for all current employees with a last name that begins with E, I, or J. We'll also throw in an ORDER BY clause. When you write an ORDER BY clause and a WHERE clause in the same code, the WHERE clause comes first, followed by the ORDER BY clause.

1. **Create** *a new query and call it* ***qry13Brackets***.
2. *Type the code as shown in* **Figure 13.26** *and* **Run** *the query.*



```
SELECT Employee_No, First_Name, Last_Name, Start_Date, End_Date
FROM Employee
WHERE Last_Name LIKE '[EIJ]*' AND End_Date = #1/1/2099#
ORDER BY Last_Name;
```

Employee_No	First_Name	Last_Name	Start_Date	End_Date
006316	TAMMY	Edwards	10/26/2014	1/1/2099
009661	Lyska	Emmick	12/16/2016	1/1/2099
002616	Damien	Ipock	12/24/2016	1/1/2099
005881	Nemesio	Ivory	12/4/2016	1/1/2099
011577	Lavatha	Jabs	9/1/2007	1/1/2099
009928	Nahsiam	Jacobi	10/14/2014	1/1/2099
001455	Madhur	Joneas	6/6/2014	1/1/2099

Figure 13.26

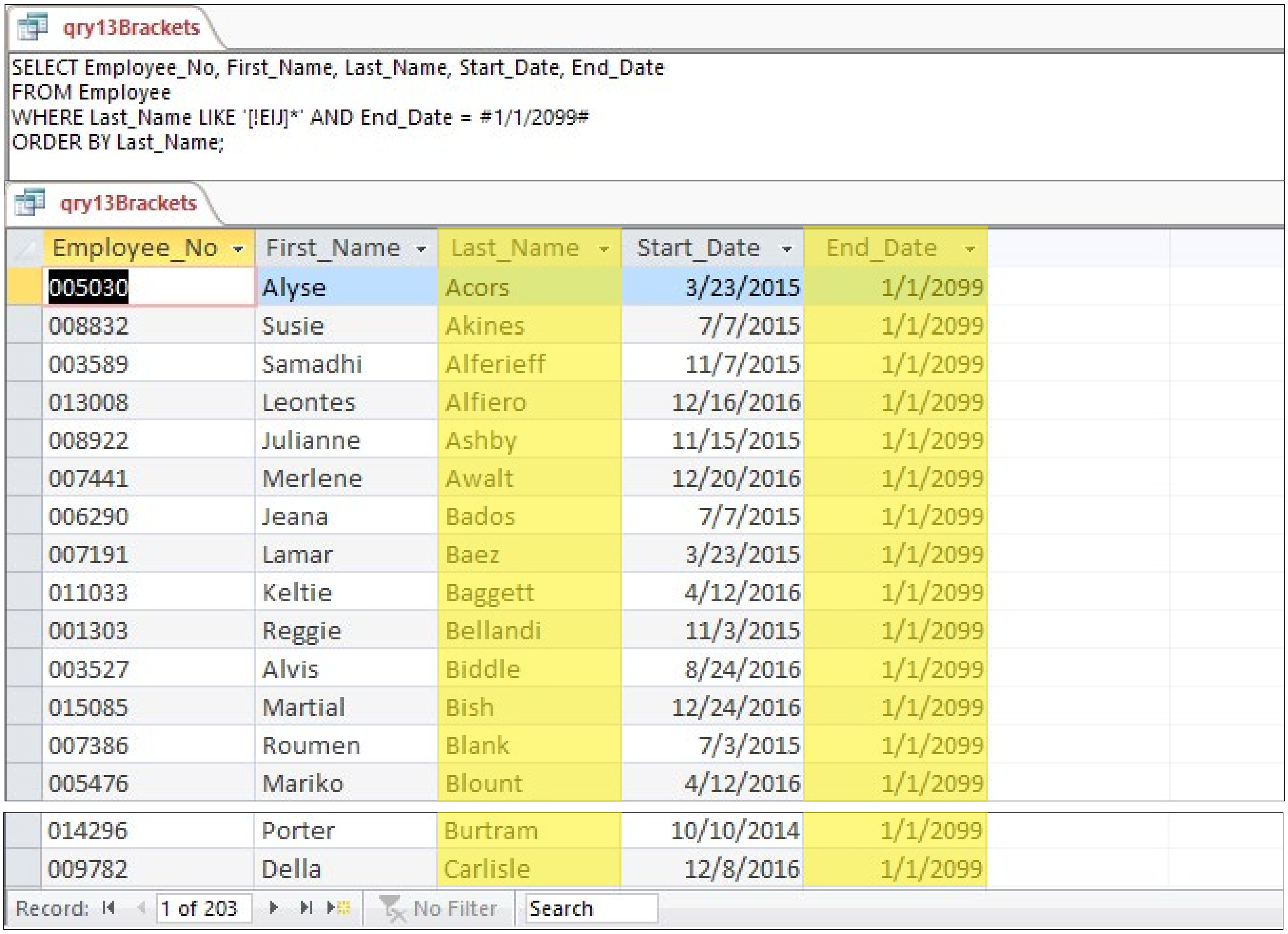
> ***Note****: In Access SQL, a date in the SQL code is surrounded by the "#" sign as in the Access query view.*

This code is actually using two different wildcard operators — the brackets and the asterisk. Using these two operators in conjunction with each other will return all records whose first character in the last name begins with E, I, or J. Note that if someone had a last name of E, I, or J (i.e., only ONE character in the last name), it would NOT return that record. You say, "No one has a last name with only one character." Not true, says I. I work with many foreign people, and I have seen people whose last name is one character only, so it does happen.

The Exclamation Point (!) and Carat (^) Characters

The last wildcard character I want to discuss is the exclamation point (!) (or the carat character (^) in ANSI SQL). These characters negate the other wildcards. For example, if you wanted to return a list of current employees whose names do not begin with E, I or J, you place the exclamation point inside the brackets in front of the search string.

3. *Edit the code as shown in* **Figure 13.27** *and* **Run** *the query.*



```
SELECT Employee_No, First_Name, Last_Name, Start_Date, End_Date
FROM Employee
WHERE Last_Name LIKE '[!EIJ]*' AND End_Date = #1/1/2099#
ORDER BY Last_Name;
```

Employee_No	First_Name	Last_Name	Start_Date	End_Date
005030	Alyse	Acors	3/23/2015	1/1/2099
008832	Susie	Akines	7/7/2015	1/1/2099
003589	Samadhi	Alferieff	11/7/2015	1/1/2099
013008	Leontes	Alfiero	12/16/2016	1/1/2099
008922	Julianne	Ashby	11/15/2015	1/1/2099
007441	Merlene	Awalt	12/20/2016	1/1/2099
006290	Jeana	Bados	7/7/2015	1/1/2099
007191	Lamar	Baez	3/23/2015	1/1/2099
011033	Keltie	Baggett	4/12/2016	1/1/2099
001303	Reggie	Bellandi	11/3/2015	1/1/2099
003527	Alvis	Biddle	8/24/2016	1/1/2099
015085	Martial	Bish	12/24/2016	1/1/2099
007386	Roumen	Blank	7/3/2015	1/1/2099
005476	Mariko	Blount	4/12/2016	1/1/2099
014296	Porter	Burtram	10/10/2014	1/1/2099
009782	Della	Carlisle	12/8/2016	1/1/2099

Figure 13.27

This record set returns a list of 203 records, which are all current employees with last names that do not begin with E, I, or J. You can also use the following code to accomplish the same results:

WHERE NOT Last_Name LIKE '[EIJ]' AND End_Date=#1/1/2099#*

On a personal note, I've used this wildcard (the exclamation point to negate a wildcard) not even three times in my career. But even though I may not use it much, it may come in handy in one of your analyses.

4. **Save** *and close* **qry13Brackets**.

Just a few ending notes about using wildcard characters:

- Don't overuse wildcards. They sometimes take up more processing resources than necessary, so if another search operator is available, use that instead.
- Try not to place wildcard characters at the beginning of the search string. That also takes up a lot of resources.

- Be careful! If you misplace wildcard characters, you can get bad results, so check and re-check your results.

As you can see, SQL code can be a powerful ally. All of the code that we wrote in this chapter can be written in Access Query Design view, but much of the code in the following chapter can't. Sometimes you can cheat in SQL by creating the query in Design View first, then editing the code in SQL View. Don't get used to that, as you can't do that in some instances.

> ***Review Questions**: It is now time to complete the hands-on Review Questions. Log on to www.ExcelCEO.com with your Email and Password, click on the* **Access 2016 and SQL Review Questions, Chapter 13, Section 2 of 2** *option and complete the review questions.*

Conclusion

In this chapter, you learned about the history of and the necessity for SQL. You started out by creating some simple SELECT/FROM statements and progressed to more complex code using ORDER BY and WHERE clauses. You learned about SQL reserved words and explored using the various operators available in SQL. You wrote code using operators like AND, OR, IN, NOT, and LIKE. You read about the order of evaluation when using the AND and OR operators. Finally, you learned about wildcard characters and how to use them in your code. In the next chapter, we'll build on the SQL skills you've gained in this chapter.

Chapter Exam

You can now go to www.ExcelCEO.com, click on Sign In, log in and take the exam. Make sure that you take the exam on the same device on which you completed the practice files, as some of the questions on the exam may refer to some of the completed projects. Chapter exams are intended to be hands-on.

CHAPTER FOURTEEN — INTERMEDIATE SQL

In this chapter, you will:

- Identify and learn how to create calculated fields in SQL
- Recognize how to create an alias field
- Identify SQL's concatenation techniques
- Determine when to use advanced functions like RTRIM() and LTRIM()
- Select and use other Text Functions
- Choose the appropriate Date and Time Functions
- Recognize when to use Aggregate Functions
- Identify Grouping concepts
- Determine when to filter on grouped data using a HAVING clause
- Identify how to create joins using SQL code
- Determine when to use SELECT DISTINCT to return a record set of unique records
- Determine when to use an IIF() Function in Access SQL and a CASE Statement in ANSI SQL
- Select and create a UNION query

CPE Credits possible for this chapter: 3

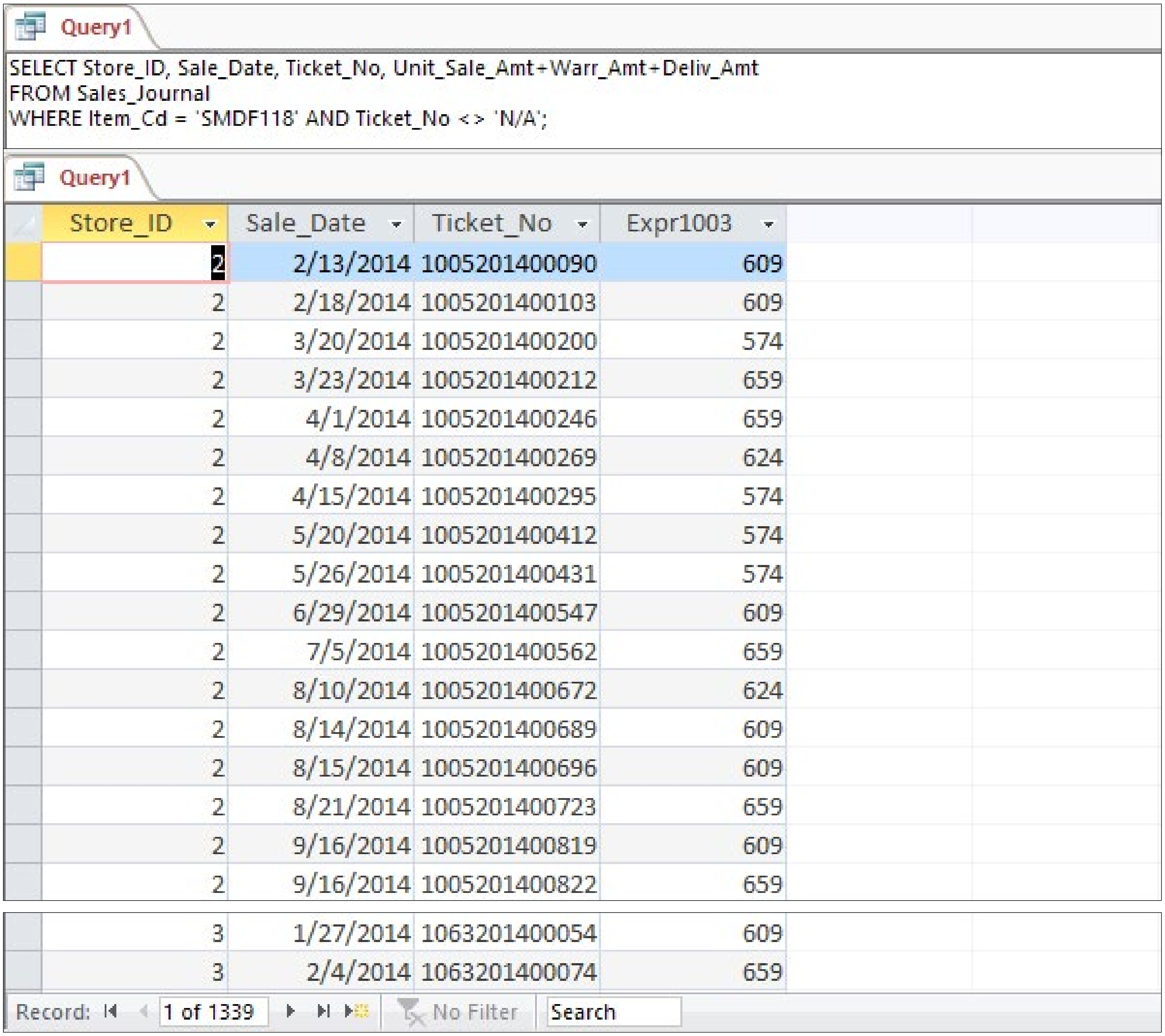
Calculated Fields

Sometimes you will need to perform calculations within your SQL code. This can be things like a summation of sales for a given month, calculating the percentage of budget to evaluate performance, or calculating an average sale. You have performed these calculations numerous times in Excel and Access, so I will assume you understand calculated fields. In SQL, these calculations are generally performed in the SELECT statement. In many cases, you will need to use functions, and many functions are very similar (if not exact) to the ones you've already learned in Excel and Access. However, they can be deceiving. The syntax for functions in the various DBMSs can be slightly different, and keeping them straight can be confusing in the different systems. For example, a MID() function in Excel and Access is a SUBSTRING() function in SQL Server and a SUBSTR() function in Oracle. A NOW() function in Excel and Access is a GETDATE() function in SQL Server and a CURDATE() function in Oracle. If you are using a function that you think should work and it doesn't, be sure to consult your DBMS's documentation to verify you're using the right syntax. In this chapter, I will use the functions that work in Access and show you the syntax for the functions in SQL Server.

At this point, let's talk a little about client-side processing verses server-side processing. Client-side processing basically means that the processing of the program is being performed on the user's computer. Server-side processing, on the other hand, is performed on the server and the results are sent back to the users' workstation. It is possible to perform SQL processing on the client side, but generally speaking, processing will be faster if you perform it on the server as DBMSs are designed to process the data more efficiently on a server than on an individual workstation. All of the code that we are processing within an Access database is done on the client side (as the database is stored on your local storage drive, or on a network or cloud-based drive located on a server), but once you progress to work with network, shared, or hosted servers, processing on the server side is preferred.

Let's start this section with mathematical operators. In Access and Excel, you've already learned how to use the plus sign (+, addition), minus sign (-, subtraction), asterisk (*, multiplication), and slash (/, division). These operators work the same in SQL code. Let's suppose you want to evaluate the sales for one item, Item No. SMDF118, the Sleepwell Double mattress, Fair quality, Saphire series. The first thing you are asked to do is to calculate the total amount sold for each sale of this item from the Sales_Journal table. Let's start by calculating the total sale price for each sale that does not have a ticket number of N/A.

1. *Open the* **Nitey_Nite_2016** *Access database and close the* **Main Menu** *form.*
2. *Using the* **Sales_Journal** *table, create a* **SQL** *query that brings in the* **Store_ID**, **Sale_Date**, *and* **Ticket_No**, *and adds the* **Unit Sale Amount plus** *the* **Warranty Amount plus** *the* **Delivery Amount**, *where the* **Item_Cd** *is* **SMDF118** *and the* **Ticket_No** *is <u>not</u>* **N/A**.



```
SELECT Store_ID, Sale_Date, Ticket_No, Unit_Sale_Amt+Warr_Amt+Deliv_Amt
FROM Sales_Journal
WHERE Item_Cd = 'SMDF118' AND Ticket_No <> 'N/A';
```

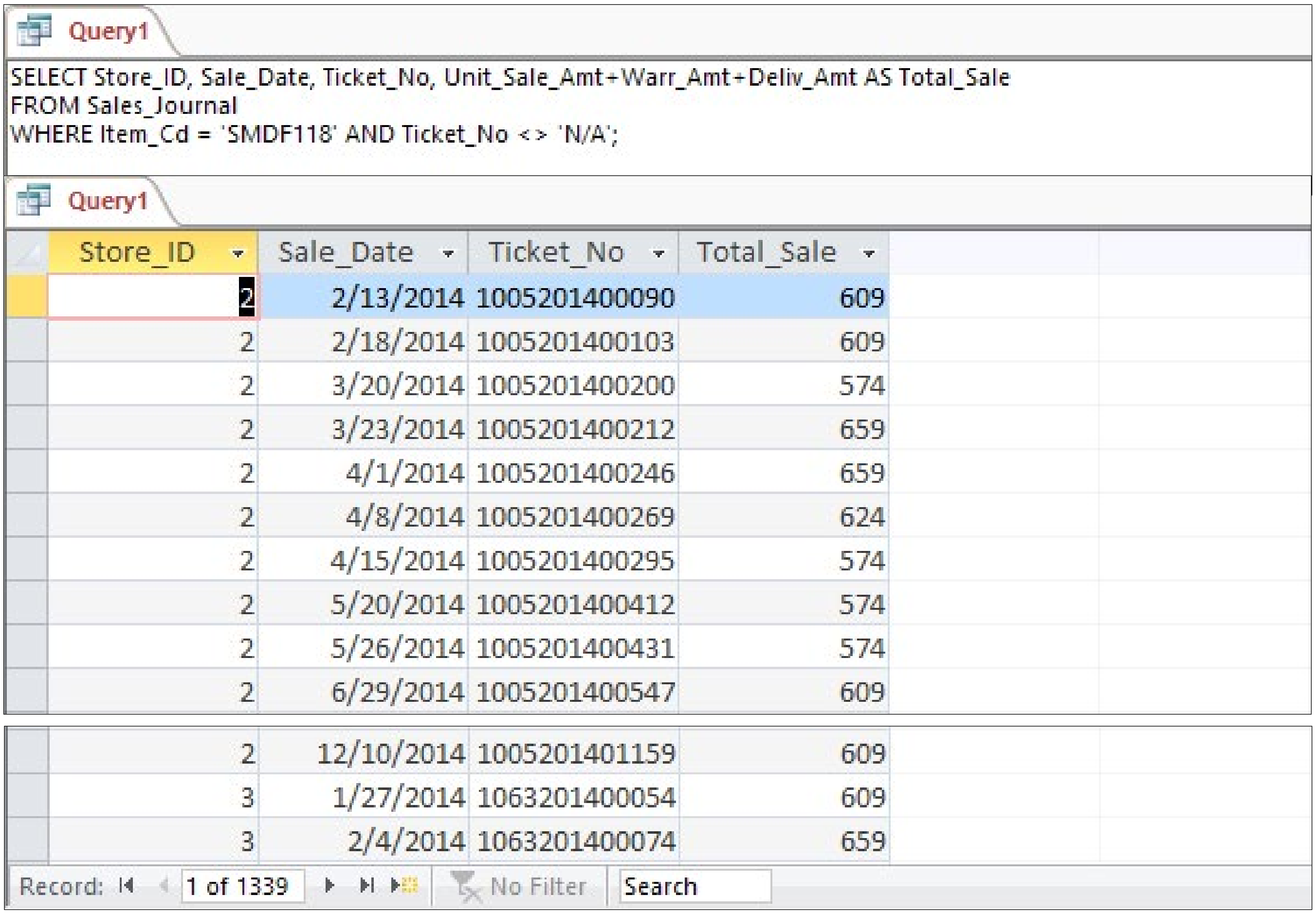
Store_ID	Sale_Date	Ticket_No	Expr1003
2	2/13/2014	1005201400090	609
2	2/18/2014	1005201400103	609
2	3/20/2014	1005201400200	574
2	3/23/2014	1005201400212	659
2	4/1/2014	1005201400246	659
2	4/8/2014	1005201400269	624
2	4/15/2014	1005201400295	574
2	5/20/2014	1005201400412	574
2	5/26/2014	1005201400431	574
2	6/29/2014	1005201400547	609
2	7/5/2014	1005201400562	659
2	8/10/2014	1005201400672	624
2	8/14/2014	1005201400689	609
2	8/15/2014	1005201400696	609
2	8/21/2014	1005201400723	659
2	9/16/2014	1005201400819	609
2	9/16/2014	1005201400822	659
3	1/27/2014	1063201400054	609
3	2/4/2014	1063201400074	659

Figure 14.1

Creating an Alias

The query returns a record set of 1,339 items, with the last field being named Expr1003. The name may be different name in your query. This field is a calculation, and you didn't assign a name to assign to it. Therefore, SQL names this field a random name beginning with Expr (for Expression) followed by a number to distinguish it from other fields without a given name. We need to give this field a name, or an alias. In the Access Design View, you put the alias name before the expression, separated by a colon. In SQL, you can put the alias name after the expression with the word AS between the expression and the alias. Let's give this expression the name Total_Sale.

3. *Type the code as shown in* **Figure 14.2** *and run the query.*



```
SELECT Store_ID, Sale_Date, Ticket_No, Unit_Sale_Amt+Warr_Amt+Deliv_Amt AS Total_Sale
FROM Sales_Journal
WHERE Item_Cd = 'SMDF118' AND Ticket_No <> 'N/A';
```

Store_ID	Sale_Date	Ticket_No	Total_Sale
2	2/13/2014	1005201400090	609
2	2/18/2014	1005201400103	609
2	3/20/2014	1005201400200	574
2	3/23/2014	1005201400212	659
2	4/1/2014	1005201400246	659
2	4/8/2014	1005201400269	624
2	4/15/2014	1005201400295	574
2	5/20/2014	1005201400412	574
2	5/26/2014	1005201400431	574
2	6/29/2014	1005201400547	609
2	12/10/2014	1005201401159	609
3	1/27/2014	1063201400054	609
3	2/4/2014	1063201400074	659

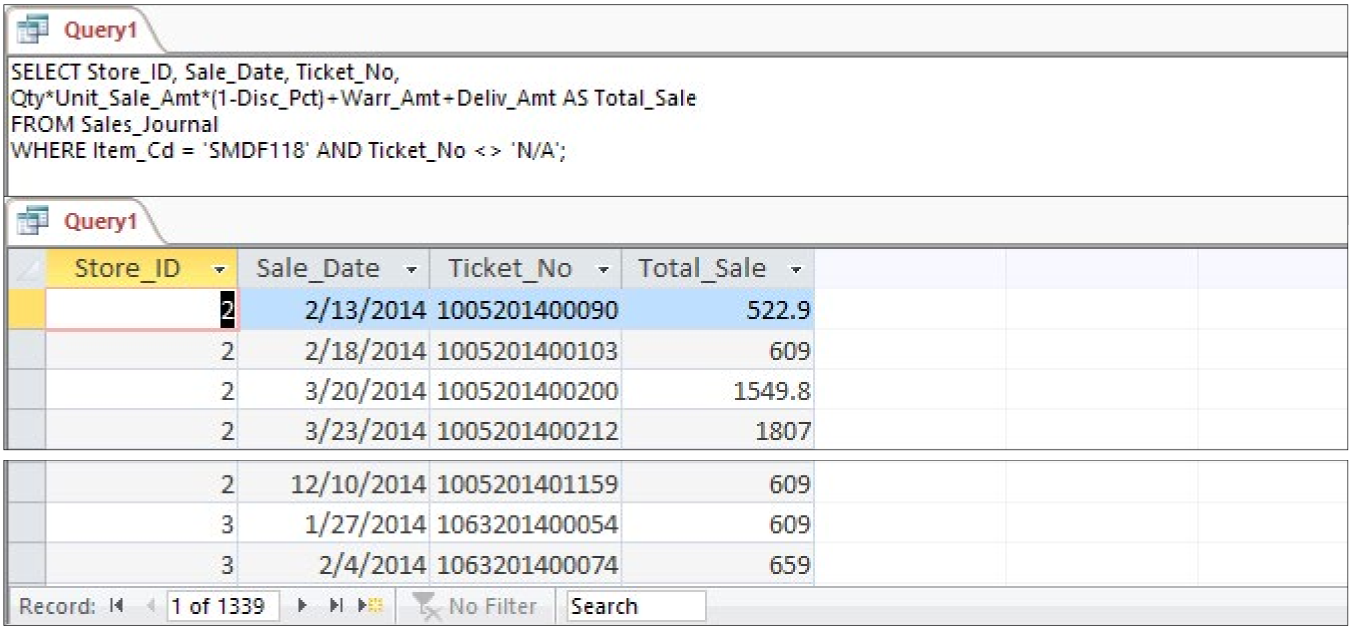
Figure 14.2

> ***Note**: You can also place the alias before the expression, separated by an equal sign, like this: Total_Sale=Unit_Sale_Amt+Warr_Amt+Deliv_Amt.*

> ***Note**: I find that **Microsoft Query** works better with the alias before the expression rather than after.*

All you did here was to change the calculated field to end with AS Total_Sale. I hope you've noticed that this calculation wasn't really the total sale amount, as you have to consider discounts and the number of units sold. Remember from the calculation you did previously that the Total Sale amount is the number of units sold times the sale amount per unit times (1-discount rate), plus the warranty and delivery amounts. Also remember that you need to put the parentheses in the correct places.

4. *Edit the code as shown in* **Figure 14.3** *and run the query.*



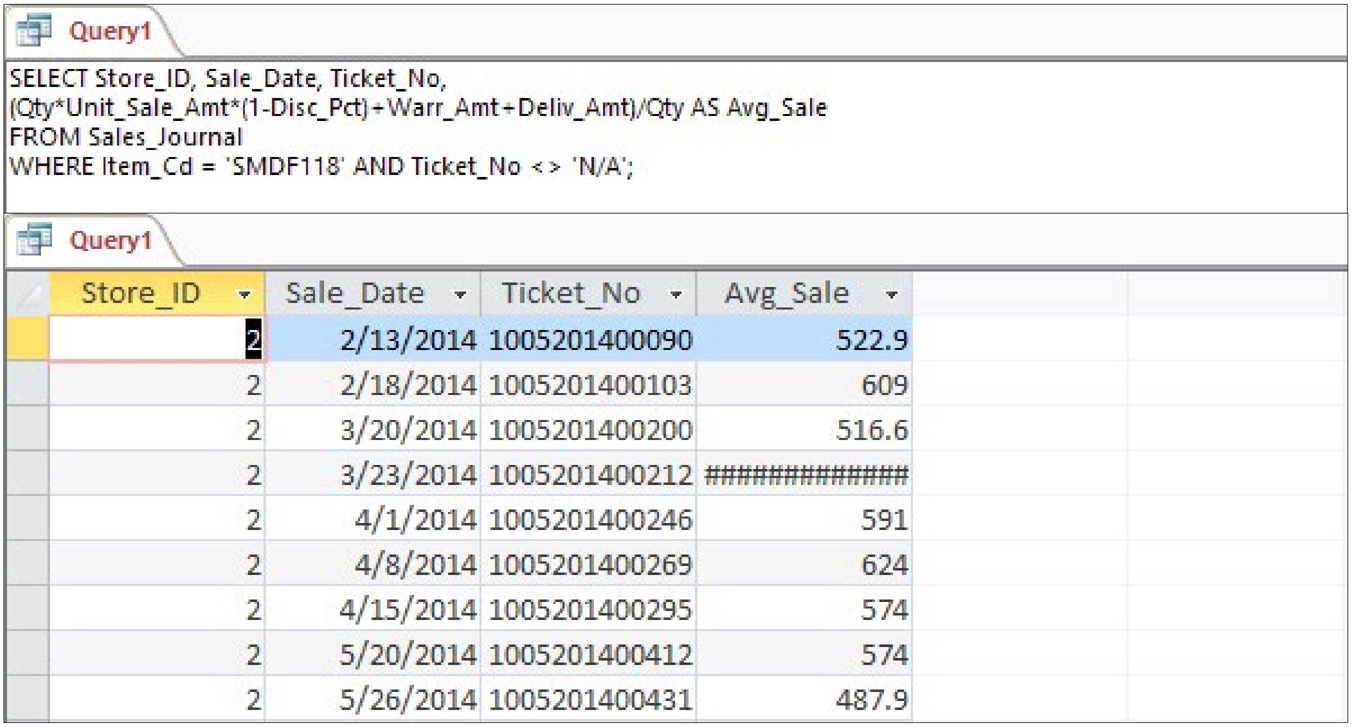
```
SELECT Store_ID, Sale_Date, Ticket_No,
Qty*Unit_Sale_Amt*(1-Disc_Pct)+Warr_Amt+Deliv_Amt AS Total_Sale
FROM Sales_Journal
WHERE Item_Cd = 'SMDF118' AND Ticket_No <> 'N/A';
```

Store_ID	Sale_Date	Ticket_No	Total_Sale
2	2/13/2014	1005201400090	522.9
2	2/18/2014	1005201400103	609
2	3/20/2014	1005201400200	1549.8
2	3/23/2014	1005201400212	1807
2	12/10/2014	1005201401159	609
3	1/27/2014	1063201400054	609
3	2/4/2014	1063201400074	659

Figure 14.3

To calculate the average sale, simply divide the Total_Sale calculation by the quantity. Let's perform that calculation and rename the Total_Sale field to Avg_Sale.

5. *Edit the code as shown in* **Figure 14.4** *and* **Run** *the query.*



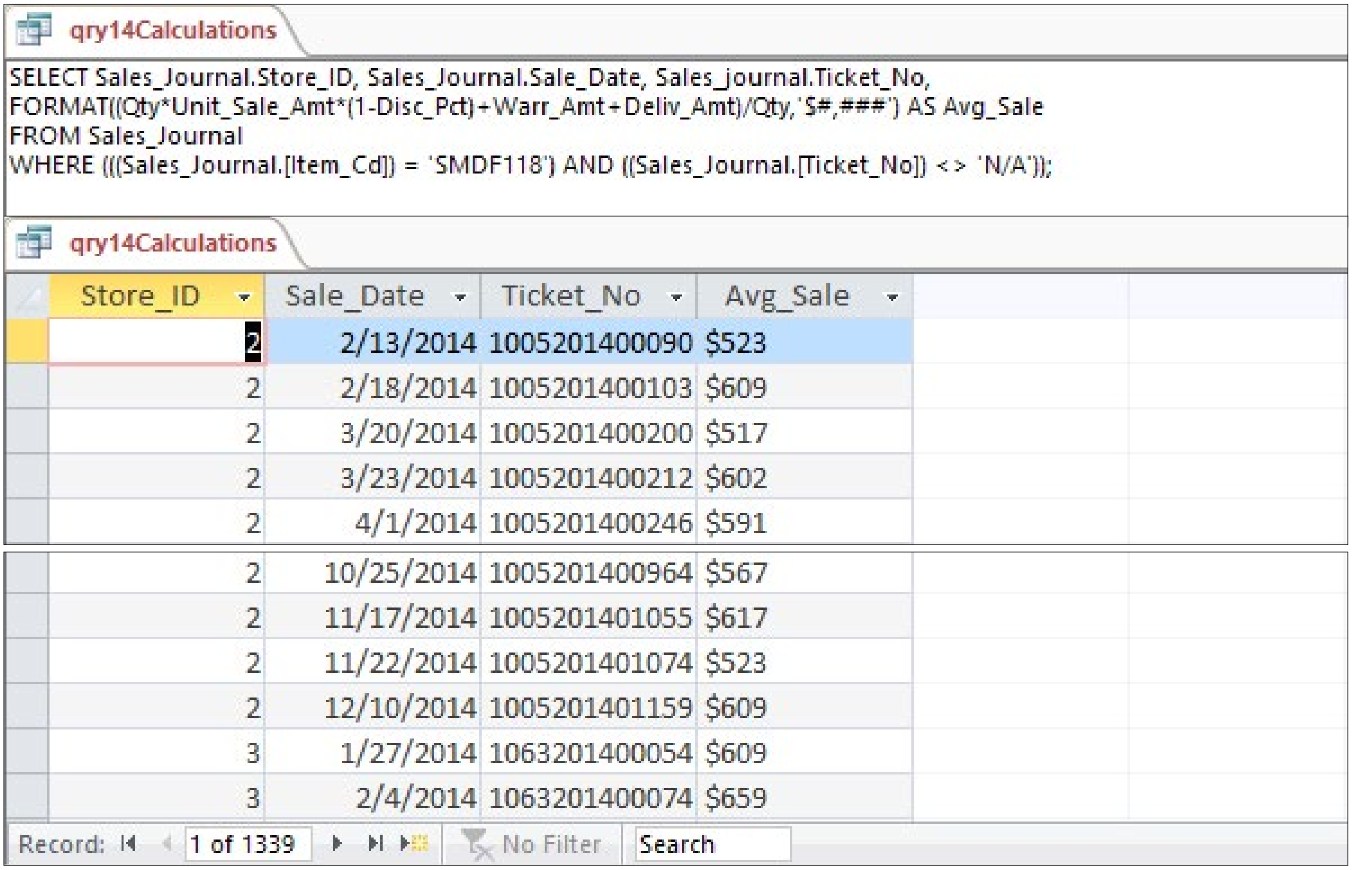
```
SELECT Store_ID, Sale_Date, Ticket_No,
(Qty*Unit_Sale_Amt*(1-Disc_Pct)+Warr_Amt+Deliv_Amt)/Qty AS Avg_Sale
FROM Sales_Journal
WHERE Item_Cd = 'SMDF118' AND Ticket_No <> 'N/A';
```

Store_ID	Sale_Date	Ticket_No	Avg_Sale
2	2/13/2014	1005201400090	522.9
2	2/18/2014	1005201400103	609
2	3/20/2014	1005201400200	516.6
2	3/23/2014	1005201400212	##############
2	4/1/2014	1005201400246	591
2	4/8/2014	1005201400269	624
2	4/15/2014	1005201400295	574
2	5/20/2014	1005201400412	574
2	5/26/2014	1005201400431	487.9

Figure 14.4

Since the values in the Avg_Sale field are not formatted, the numbers that are too big to fit inside the width of the field are represented as ############. To fix it, simply adjust the width of the column or apply formatting to the field. Again, with your experience in Excel and Access, you should have no problem understanding mathematical operators and how to use them. In SQL, they work just like they do in Access and Excel. Let's format the Avg_Sale field.

6. **Save** *the query as* **qry14Calculations**.
7. *Edit the code as shown in* **Figure 14.5** *and* **Run** *the query*



```
SELECT Sales_Journal.Store_ID, Sales_Journal.Sale_Date, Sales_journal.Ticket_No,
FORMAT((Qty*Unit_Sale_Amt*(1-Disc_Pct)+Warr_Amt+Deliv_Amt)/Qty,'$#,###') AS Avg_Sale
FROM Sales_Journal
WHERE (((Sales_Journal.[Item_Cd]) = 'SMDF118') AND ((Sales_Journal.[Ticket_No]) <> 'N/A'));
```

Store_ID	Sale_Date	Ticket_No	Avg_Sale
2	2/13/2014	1005201400090	$523
2	2/18/2014	1005201400103	$609
2	3/20/2014	1005201400200	$517
2	3/23/2014	1005201400212	$602
2	4/1/2014	1005201400246	$591
2	10/25/2014	1005201400964	$567
2	11/17/2014	1005201401055	$617
2	11/22/2014	1005201401074	$523
2	12/10/2014	1005201401159	$609
3	1/27/2014	1063201400054	$609
3	2/4/2014	1063201400074	$659

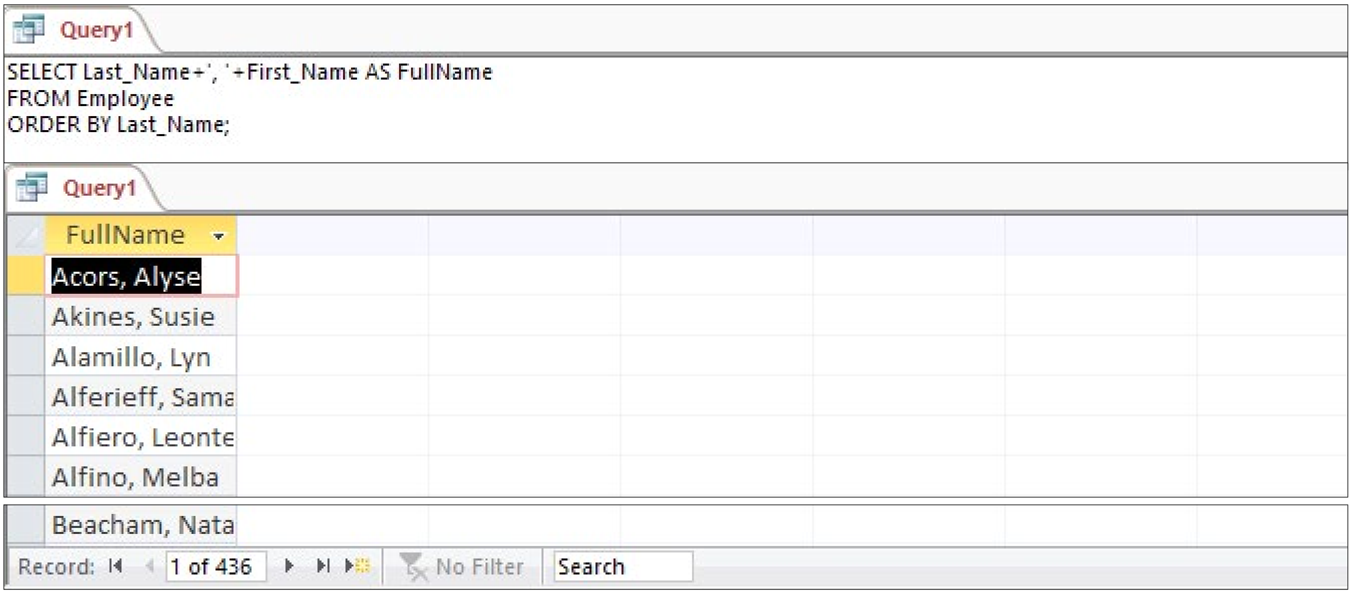
Figure 14.5

8. **Save** *and close* **qry14Calculations**.

Concatenation

Concatenation in SQL is basically the same as in Excel and Access, with one slight syntax difference – in SQL, you use the plus (+) and apostrophe (') signs in the place of ampersand (&) and quote ("). It is also a good idea to create an alias for any concatenated string. Let's use the Employee table and concatenate the Last_Name and First_Name fields, separated by a comma, to create the Full Name of the employee.

1. *Create a new* **SQL** *query with the code as shown in* **Figure 14.6** *and* **Run** *the query.*



```
SELECT Last_Name+', '+First_Name AS FullName
FROM Employee
ORDER BY Last_Name;
```

FullName
Acors, Alyse
Akines, Susie
Alamillo, Lyn
Alferieff, Sama
Alfiero, Leonte
Alfino, Melba
Beacham, Nata

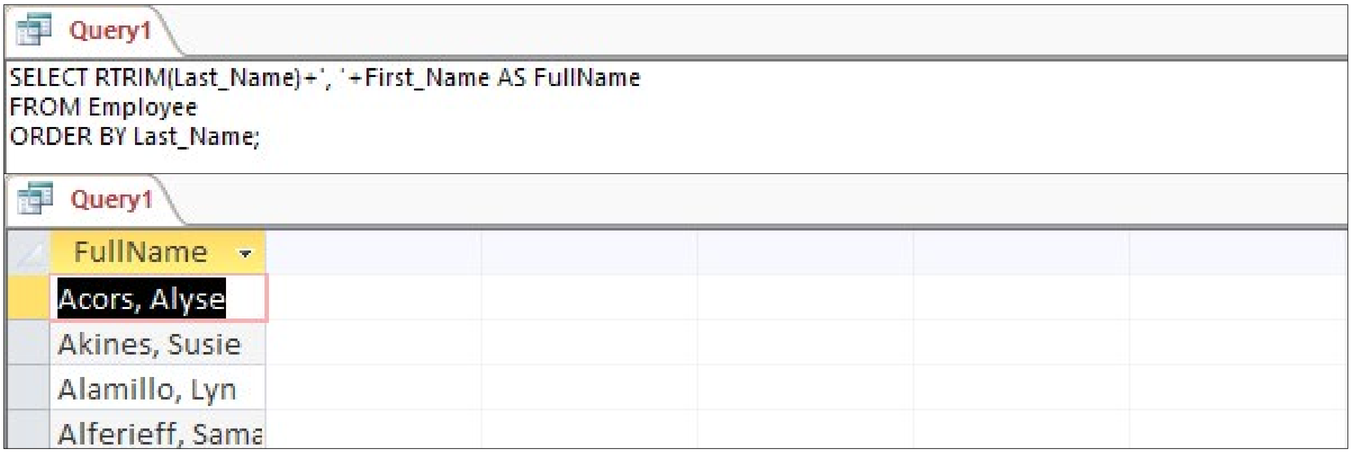
Figure 14.6

As a general rule, anything you type in between the apostrophes will appear as a text string. In this example, all we did was type a comma and a space to separate the last name from the first name.

The LTRIM() and RTRIM() Functions

Sometimes you may get records that have spaces before or after the text string. You worked some examples of this in Excel, where you used the TRIM() function. But in SQL, you have to define whether the spaces are before or after the text string. If the spaces are before the text string, you will use an LTRIM() function. You use the RTRIM() function (which is more frequently used) when the spaces appear after the text string. The syntax for both is the same, so let's do an example using the RTRIM() function.

2. *Edit the* **SQL** *code in the query as shown in* **Figure 14.7** *and* **Run** *it.*



```
SELECT RTRIM(Last_Name)+', '+First_Name AS FullName
FROM Employee
ORDER BY Last_Name;
```

FullName
Acors, Alyse
Akines, Susie
Alamillo, Lyn
Alferieff, Sama

Figure 14.7

The results should be the same as in Figure 14.6, as there were no spaces after any of the last names.

3. **Save** *the query as* ***qry14Concatenation*** *and close.*

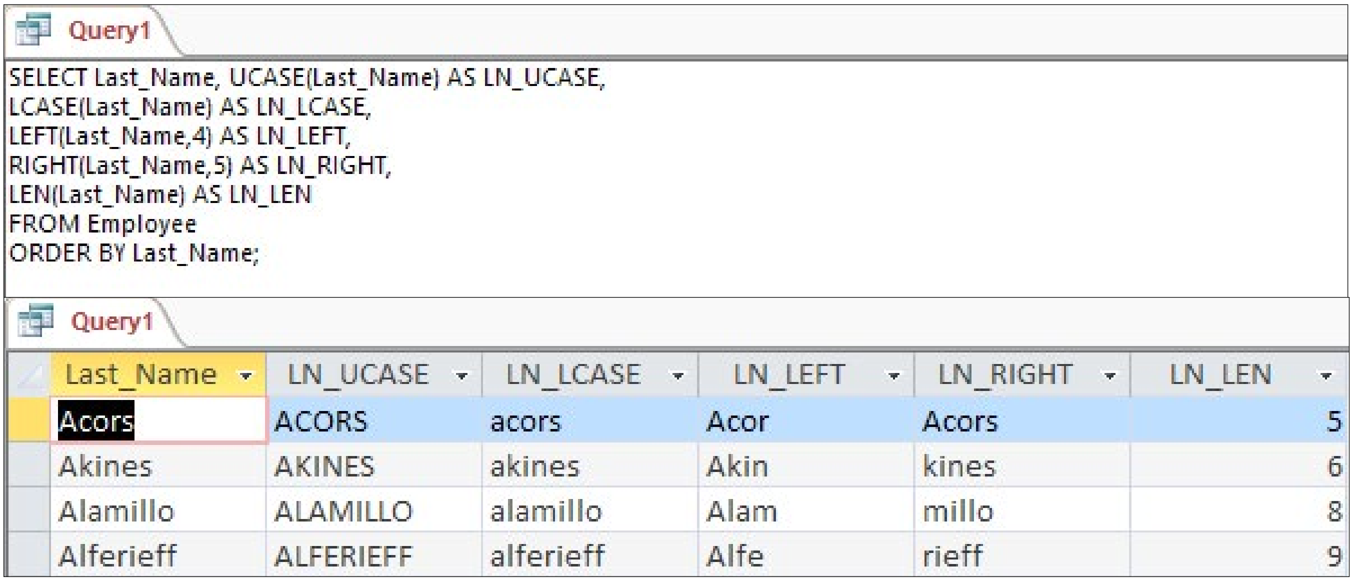
Text Functions

The RTRIM() and LTRIM() functions are examples of Text functions. In this section, we'll explore more text functions. They all work basically the same, so we won't do examples of each. The following table lists examples of other text functions.

Function	Description	Example	Result
LEFT()	Returns characters from the left of the text string	LEFT('Johnson',4)	John
RIGHT()	Returns characters from the right of the text string	RIGHT('Johnson',5)	hnson
LOWER() (use LCASE in Access)	Converts all text in the string to lower-case	LOWER('Apple')	apple
UPPER() (use UCASE in Access)	Converts all text in the string to upper-case	UPPER('Apple')	APPLE
LEN()	Returns the length, or the number of characters in the string	LEN('White')	5

Let's do a query that shows the effects of each of these functions.

4. *Create a new* **SQL** *query, type in the code as in* **Figure 14.8** *and* **Run** *it.*



```
SELECT Last_Name, UCASE(Last_Name) AS LN_UCASE,
LCASE(Last_Name) AS LN_LCASE,
LEFT(Last_Name,4) AS LN_LEFT,
RIGHT(Last_Name,5) AS LN_RIGHT,
LEN(Last_Name) AS LN_LEN
FROM Employee
ORDER BY Last_Name;
```

Last_Name	LN_UCASE	LN_LCASE	LN_LEFT	LN_RIGHT	LN_LEN
Acors	ACORS	acors	Acor	Acors	5
Akines	AKINES	akines	Akin	kines	6
Alamillo	ALAMILLO	alamillo	Alam	millo	8
Alferieff	ALFERIEFF	alferieff	Alfe	rieff	9

Figure 14.8

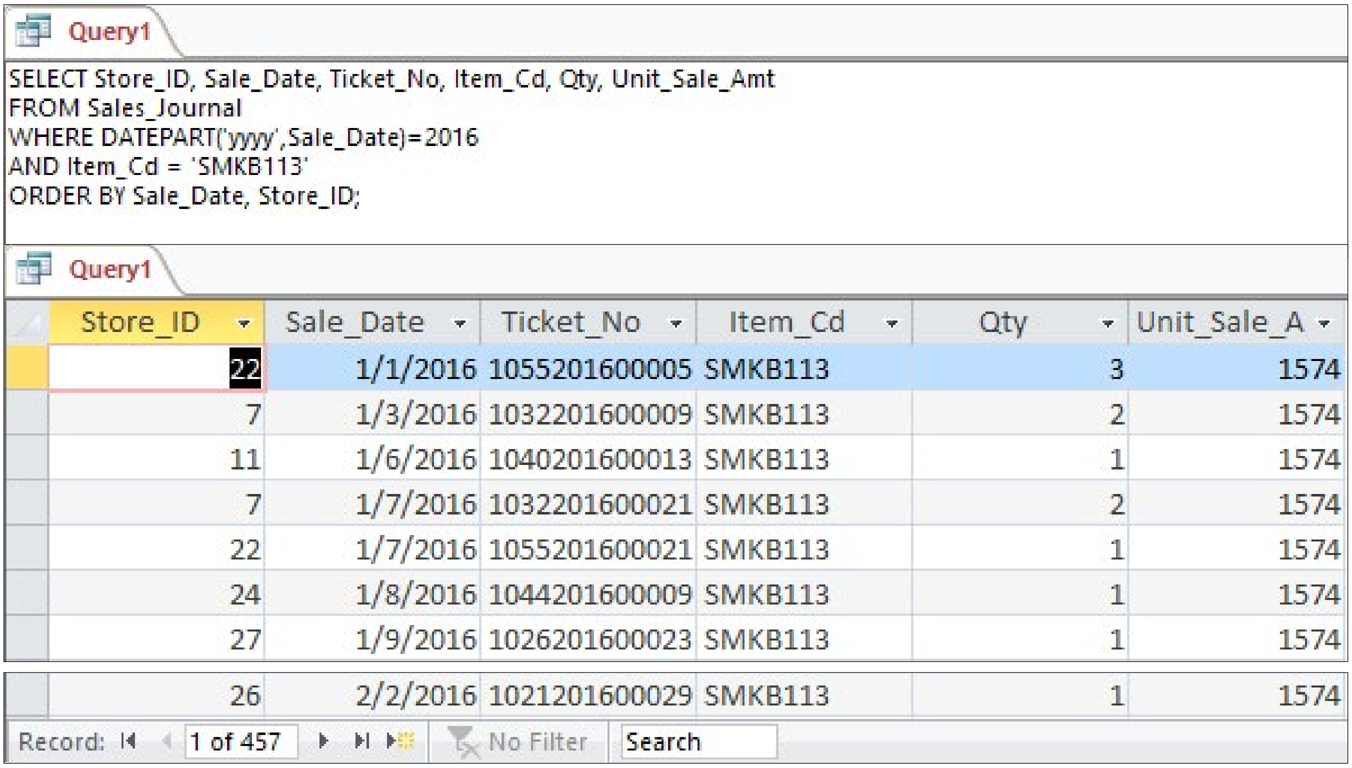
5. **Save** *the query as* ***qry14Text*** *and close.*

Date Functions

Since date and time data are stored in tables with their own data types, they require their own set of functions. The data is stored in its own format so it can be queried quickly and efficiently. Each DBMS has its own way of using date and time functions, and we will concentrate on the functions as used in Access and SQL Server. Again, consult your DBMS's documentation for date and time functions for something other than Access and SQL Server.

In the next example, you will query the Sales_Journal table for all of the sales in 2016 for the most expensive item, which is Item Code SMKB113. To do this in Access SQL, you'll use the DATEPART() function in the WHERE clause of the code.

1. **Create** *a new* **SQL** *query, type in the code as in* **Figure 14.9** *and* **Run** *it.*



```
SELECT Store_ID, Sale_Date, Ticket_No, Item_Cd, Qty, Unit_Sale_Amt
FROM Sales_Journal
WHERE DATEPART('yyyy',Sale_Date)=2016
AND Item_Cd = 'SMKB113'
ORDER BY Sale_Date, Store_ID;
```

Store_ID	Sale_Date	Ticket_No	Item_Cd	Qty	Unit_Sale_A
22	1/1/2016	1055201600005	SMKB113	3	1574
7	1/3/2016	1032201600009	SMKB113	2	1574
11	1/6/2016	1040201600013	SMKB113	1	1574
7	1/7/2016	1032201600021	SMKB113	2	1574
22	1/7/2016	1055201600021	SMKB113	1	1574
24	1/8/2016	1044201600009	SMKB113	1	1574
27	1/9/2016	1026201600023	SMKB113	1	1574
26	2/2/2016	1021201600029	SMKB113	1	1574

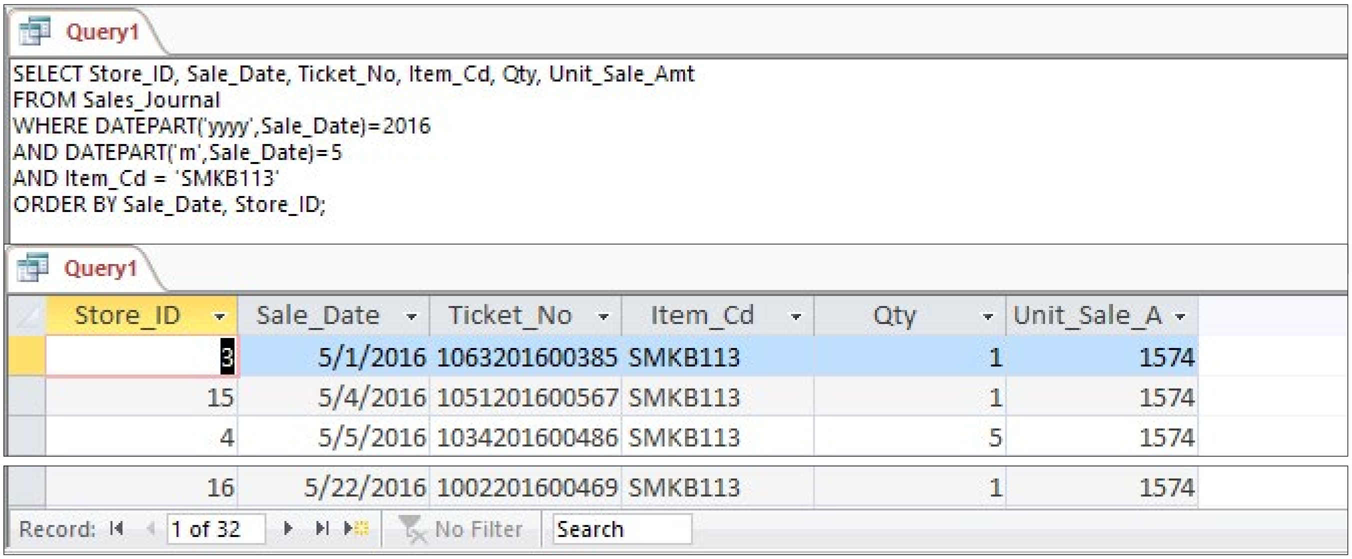
Figure 14.9

You should get a record set of 457 records. For SQL Server, you need to change up the syntax on the DATEPART() function to look like this:

WHERE DATEPART(yy, Sale_Date)=2016

To return the record set of all records in May 2016, you need to include another DATEPART() function and change the first argument to month instead of year. Let's perform an additional filter for the month of May.

2. *Edit the* **SQL** *code as in* **Figure 14.10** *and* **Run** *it.*



```
SELECT Store_ID, Sale_Date, Ticket_No, Item_Cd, Qty, Unit_Sale_Amt
FROM Sales_Journal
WHERE DATEPART('yyyy',Sale_Date)=2016
AND DATEPART('m',Sale_Date)=5
AND Item_Cd = 'SMKB113'
ORDER BY Sale_Date, Store_ID;
```

Store_ID	Sale_Date	Ticket_No	Item_Cd	Qty	Unit_Sale_A
3	5/1/2016	1063201600385	SMKB113	1	1574
15	5/4/2016	1051201600567	SMKB113	1	1574
4	5/5/2016	1034201600486	SMKB113	5	1574
16	5/22/2016	1002201600469	SMKB113	1	1574

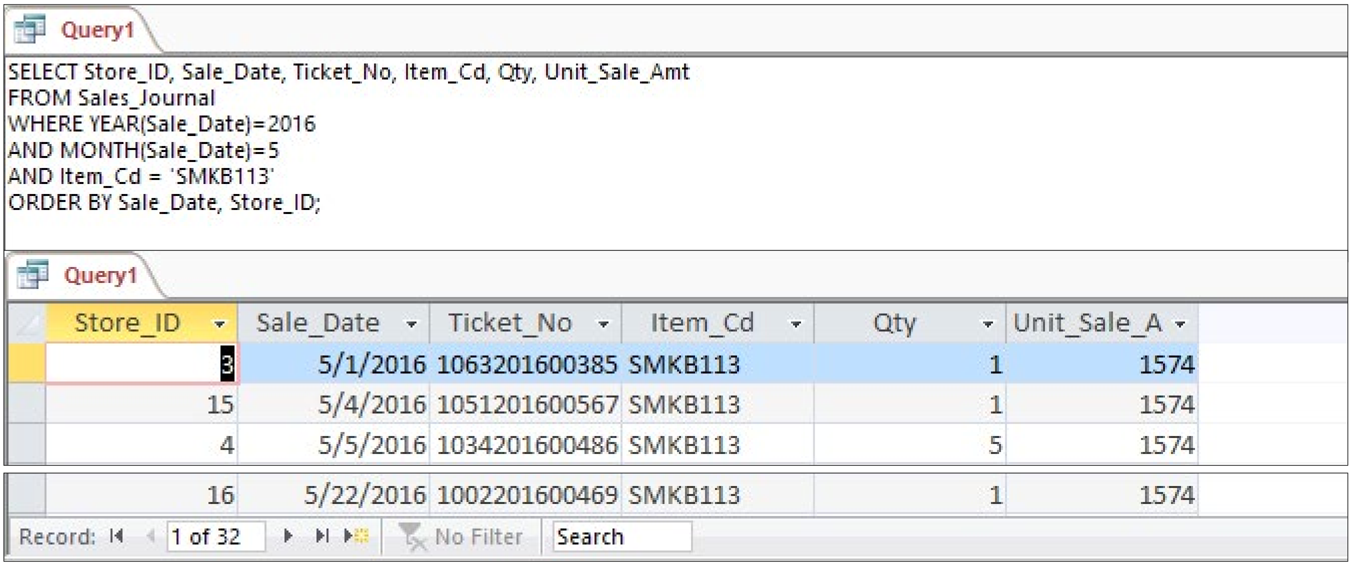
Figure 14.10

The SQL Server syntax for the month criteria is as follows:

AND DATEPART(m, Sale_Date)=5

This time, you should get 32 records returned. Alternatively, you can use the YEAR() and MONTH() functions to get the same results, if you are using Access.

3. *Edit the* **SQL** *code as in* **Figure 14.11** *and* **Run** *it.*



```
SELECT Store_ID, Sale_Date, Ticket_No, Item_Cd, Qty, Unit_Sale_Amt
FROM Sales_Journal
WHERE YEAR(Sale_Date)=2016
AND MONTH(Sale_Date)=5
AND Item_Cd = 'SMKB113'
ORDER BY Sale_Date, Store_ID;
```

Store_ID	Sale_Date	Ticket_No	Item_Cd	Qty	Unit_Sale_A
3	5/1/2016	1063201600385	SMKB113	1	1574
15	5/4/2016	1051201600567	SMKB113	1	1574
4	5/5/2016	1034201600486	SMKB113	5	1574
16	5/22/2016	1002201600469	SMKB113	1	1574

Figure 14.11

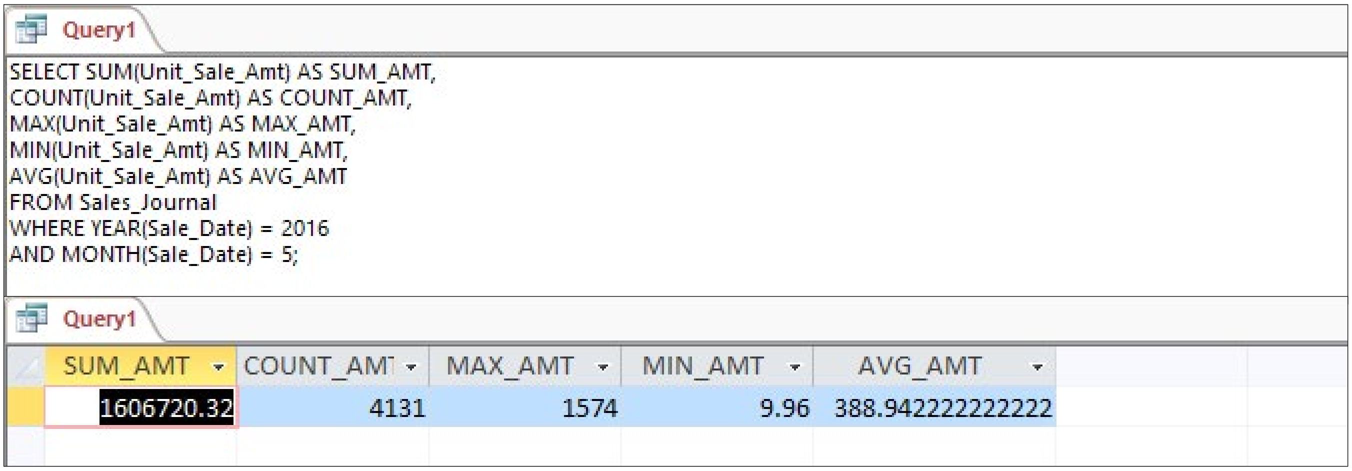
4. **Save** *the query as* ***qry14Dates*** *and close it.*

> ***Review Questions****: It is now time to complete the hands-on Review Questions. Log on to www.ExcelCEO.com with your Email and Password, click on the* **Access 2016 and SQL Review Questions, Chapter 14, Section 1 of 2** *option and complete the review questions.*

Aggregate Functions

In Excel and Access, you used the SUM(), COUNT(), MAX(), MIN(), and AVERAGE() ***aggregate functions***. All of these functions are the same in SQL, with the exception of AVERAGE(), where you use AVG(). Since you've had plenty of experience working with these functions in Excel and Access, I won't go into much detail about them. You'll use all of the above functions in one piece of code to make sure you understand how they all work. In this example, you'll run the sum, count, maximum, minimum, and average values for all sales in the Sales_Journal table in May 2016.

5. *Create a new* **SQL** *query, type in the code as in* **Figure 14.12** *and* **Run** *it.*



```
SELECT SUM(Unit_Sale_Amt) AS SUM_AMT,
COUNT(Unit_Sale_Amt) AS COUNT_AMT,
MAX(Unit_Sale_Amt) AS MAX_AMT,
MIN(Unit_Sale_Amt) AS MIN_AMT,
AVG(Unit_Sale_Amt) AS AVG_AMT
FROM Sales_Journal
WHERE YEAR(Sale_Date) = 2016
AND MONTH(Sale_Date) = 5;
```

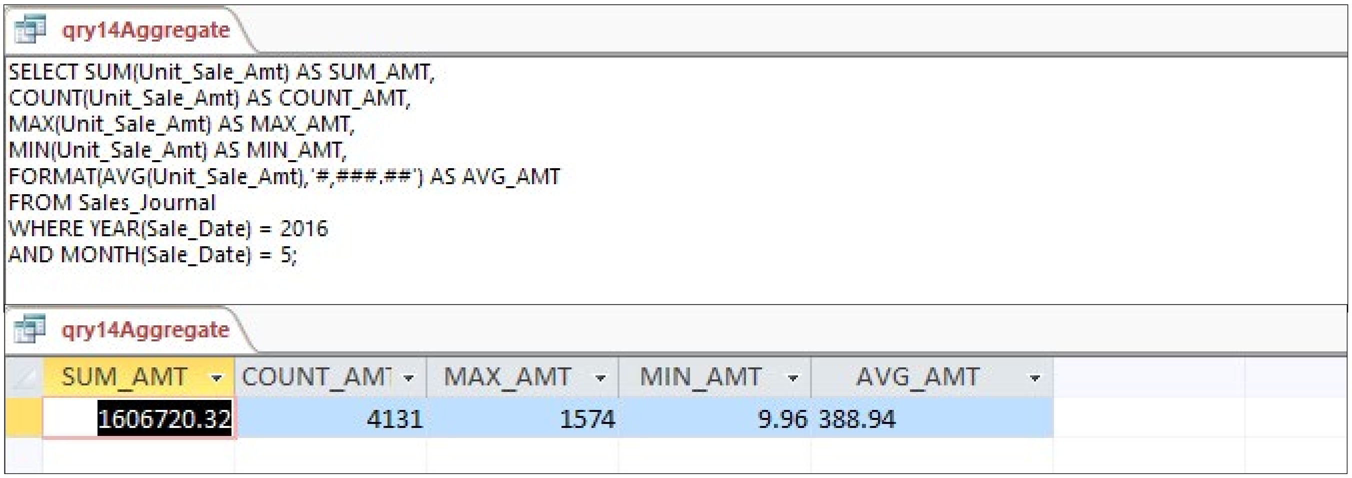
SUM_AMT	COUNT_AMT	MAX_AMT	MIN_AMT	AVG_AMT
1606720.32	4131	1574	9.96	388.942222222222

Figure 14.12

You should get one record that looks like Figure 14.12. Again, since you've had so much experience with aggregate functions already, you should be able to grasp this concept with no problems.

6. **Save** *the query as* ***qry14Aggregate****, but don't close it yet.*

Let's use some formatting to make the AVG_AMT a bit more readable since that is so easy to correct.



```
SELECT SUM(Unit_Sale_Amt) AS SUM_AMT,
COUNT(Unit_Sale_Amt) AS COUNT_AMT,
MAX(Unit_Sale_Amt) AS MAX_AMT,
MIN(Unit_Sale_Amt) AS MIN_AMT,
FORMAT(AVG(Unit_Sale_Amt),'#,###.##') AS AVG_AMT
FROM Sales_Journal
WHERE YEAR(Sale_Date) = 2016
AND MONTH(Sale_Date) = 5;
```

SUM_AMT	COUNT_AMT	MAX_AMT	MIN_AMT	AVG_AMT
1606720.32	4131	1574	9.96	388.94

Figure 14.13

7. **Save qry14Aggregate** *and close it.*

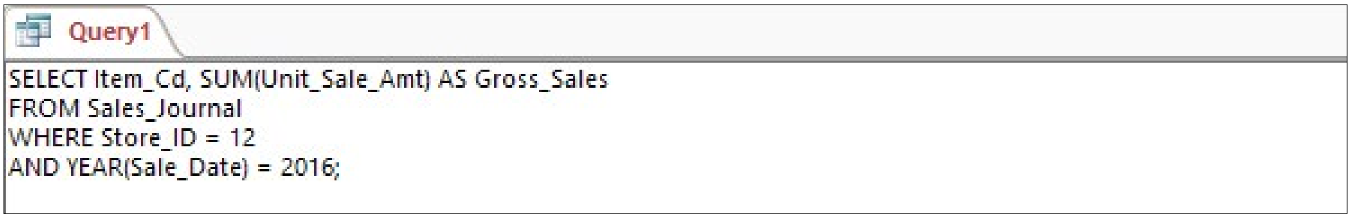
Grouping

The last section reviewed aggregate functions, which work very well when you want to perform those analyses on all records in the table or query. But in our example using the Sales_Journal, what if you want to summarize by item code? You could type the item number in the WHERE clause (such as WHERE Item_Cd='SMKB113'), but that would give you the aggregation for only that one item. Let's suppose you want to sum all the delivery amounts by item for all items in the same query. To do this, you will use ***Grouping***. Grouping allows you to segregate the data into groups so you can perform aggregate functions on each set.

Grouping in SQL code is the same as if you used the Totals icon in an Access query Design View. When you click on the Totals icon, a new Totals line appears that contains the default value of Group By. We use the same syntax in SQL code. If you understand how this works in Access, but are having a hard time understanding the SQL code, you can cheat. You can design a simple query in Access, run it to make sure it works, then look at the query in SQL View. A lot of beginning SQL students do that until they understand SQL coding. I also did that when I was learning to program using SQL. However, I got to a point where it was easier to program with SQL code than it was to design the query in Access Design View.

Of course, the best way to show you how it's done is through an example. Let's use the Sales_Journal again. We want to sum the gross sale amount for each item at Store_ID 12 in the year 2016.

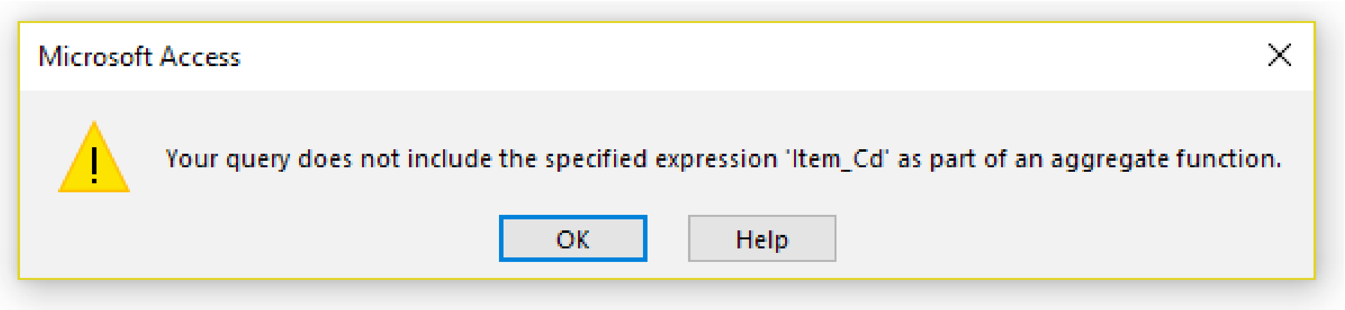
1. **Create** *a new* **SQL** *query, type in the code as in* **Figure 14.14** *and* **Run** *it.*



```
SELECT Item_Cd, SUM(Unit_Sale_Amt) AS Gross_Sales
FROM Sales_Journal
WHERE Store_ID = 12
AND YEAR(Sale_Date) = 2016;
```

Figure 14.14

When you run this code, you will get an error message that looks like this:

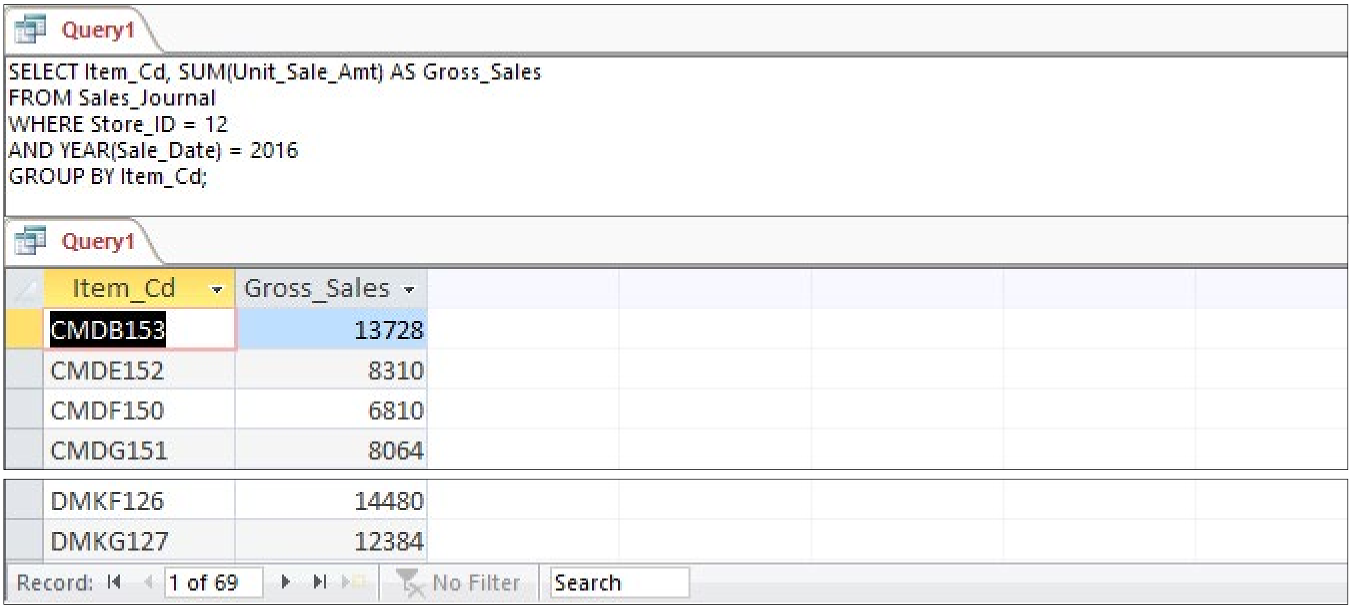


Your query does not include the specified expression 'Item_Cd' as part of an aggregate function.

Figure 14.15

The message is telling you that you tried to run a query and didn't group it correctly. As you are trying to return the gross sales for each item, you have to group by that item.

2. *Click* **OK***.*
3. *Edit the query to include a* ***GROUP BY*** *statement as show in* **Figure 14.16** *and* **Run** *it.*



```
SELECT Item_Cd, SUM(Unit_Sale_Amt) AS Gross_Sales
FROM Sales_Journal
WHERE Store_ID = 12
AND YEAR(Sale_Date) = 2016
GROUP BY Item_Cd;
```

Item_Cd	Gross_Sales
CMDB153	13728
CMDE152	8310
CMDF150	6810
CMDG151	8064
DMKF126	14480
DMKG127	12384

Figure 14.16

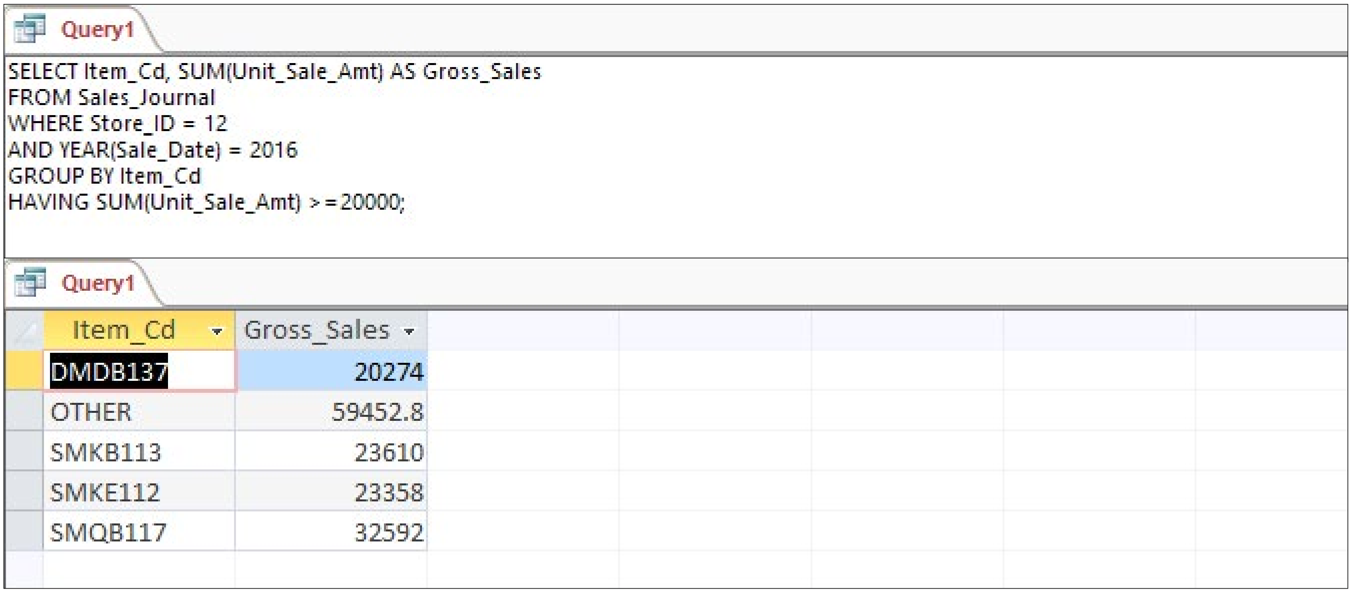
There should have been 69 records returned by this query. Because you used the GROUP BY clause, the query automatically included all items. You could have used an IN() function to include specific items, or if you wanted certain items excluded from the analysis. Here are a few rules that you need to keep in mind when using a GROUP BY clause:

- You can include as many fields as you want in your SELECT statement, but every field in the SELECT statement that is not an aggregate function needs to be included in the GROUP BY clause.
- An alias name cannot be referred to in the GROUP BY clause in the same SELECT query.
- If the value in a field is NULL, then NULL will be returned in the group, and all NULL values will be grouped together.
- The GROUP BY clause must be written after the WHERE clause, but before the ORDER BY clause (if any).

Filtering on Grouped Data

We've already reviewed the WHERE clause. A WHERE clause is always coded before the GROUP BY clause, but what happens if you want to filter the grouped dataset after the group runs? Let's suppose in the query you wrote in Figure 14.14 that you want to see only those item codes where the sum of the gross sales is greater than $20,000. You can't include that criteria in the WHERE clause because you want to perform the filter after it is first filtered for Store_ID and Year. In this situation, you will use a ***HAVING*** clause.

4. *Edit the query to include a **HAVING** clause statement as show in **Figure 14.17** and **Run** it.*



```
SELECT Item_Cd, SUM(Unit_Sale_Amt) AS Gross_Sales
FROM Sales_Journal
WHERE Store_ID = 12
AND YEAR(Sale_Date) = 2016
GROUP BY Item_Cd
HAVING SUM(Unit_Sale_Amt) >=20000;
```

Item_Cd	Gross_Sales
DMDB137	20274
OTHER	59452.8
SMKB113	23610
SMKE112	23358
SMQB117	32592

Figure 14.17

This query should return five records. A nickel's worth of free advice on using a HAVING clause – BE CAREFUL! It can be confusing to use. A SQL programmer friend of mine once told me that in all the years he had programmed with SQL, he had NEVER used a HAVING clause. Later, he came back and told me he had to use it later on that same day in a query. I suppose that's kind of like people who say they never get sick, and then end up in the hospital with food poisoning the next day.

Just remember to use a WHERE clause before the GROUP BY statement and use a HAVING clause after the GROUP BY statement. This is an important concept to remember as items filtered out by the WHERE clause will not appear in the record set. The HAVING clause simply takes the grouped results of the record set and further filters it.

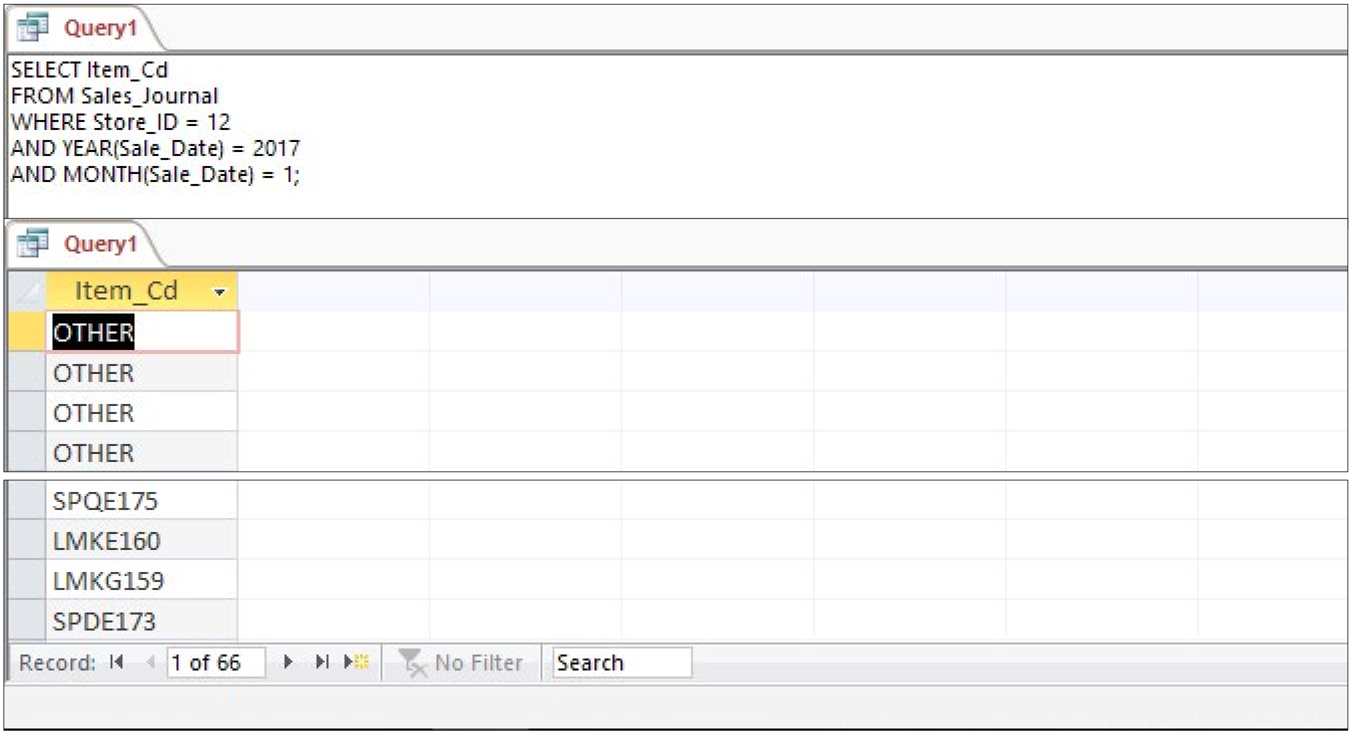
Additionally, don't forget the ORDER BY clause. Whenever you group data in a GROUP BY clause with a WHERE and/or HAVING clauses, it is probably a good idea to include an ORDER BY clause. Don't depend on the GROUP BY clause to sort your data for you. Sometimes a GROUP BY statement will order it correctly, but you shouldn't depend on it.

5. **Save** *the query as* ***qry14Grouping*** *and close it.*

SELECT DISTINCT

When you don't need to use an aggregate function (like SUM() or COUNT()) and all you want is to return distinct rows of data, you can use a ***SELECT DISTINCT*** statement. To illustrate, let's suppose that you want to return a dataset of only the item codes sold at Store ID 12 in January 2017.

6. **Create** *a new* **SQL** *query, type in the code as in* **Figure 14.18** *and* **Run** *it.*



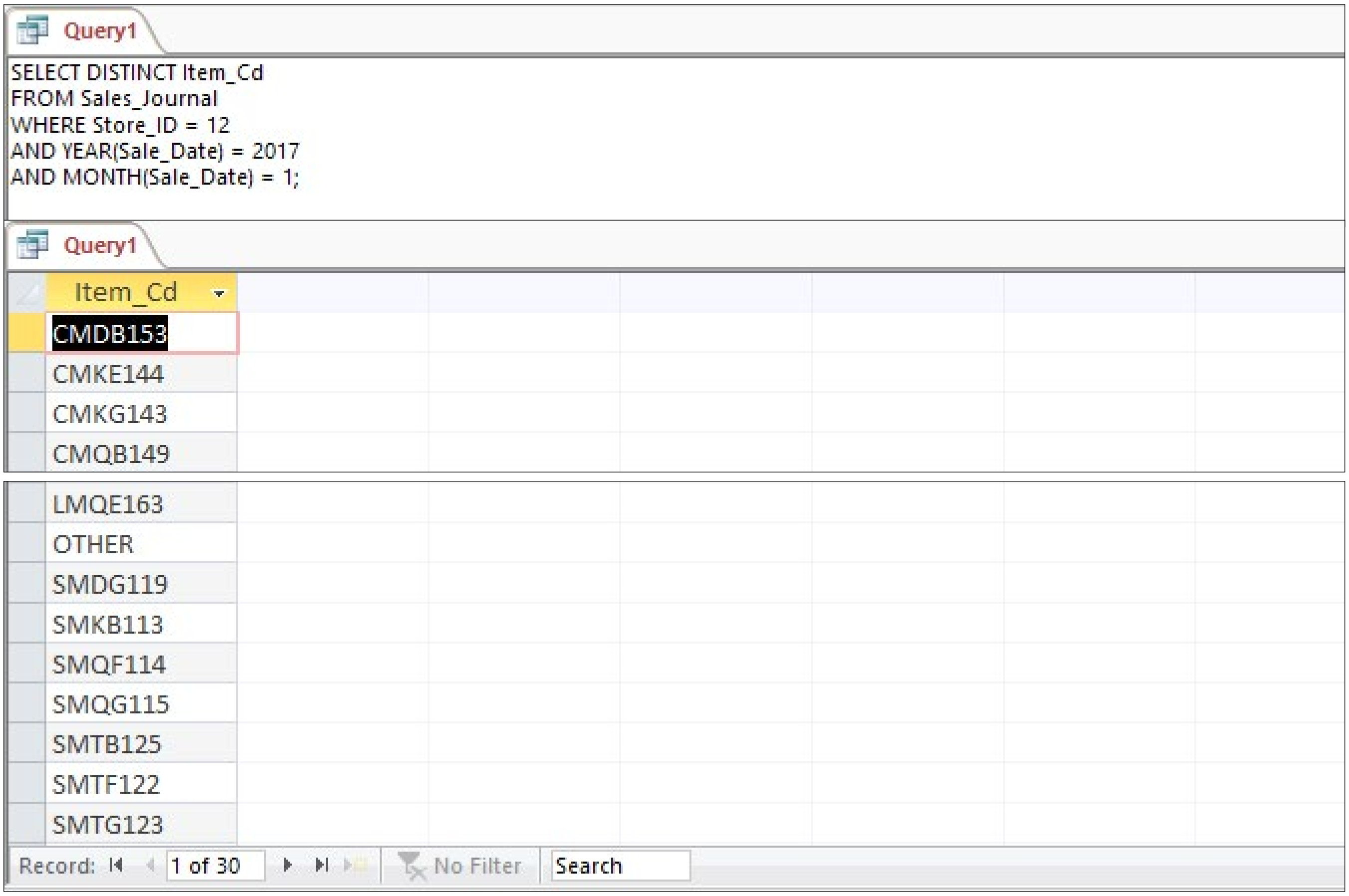
```
SELECT Item_Cd
FROM Sales_Journal
WHERE Store_ID = 12
AND YEAR(Sale_Date) = 2017
AND MONTH(Sale_Date) = 1;
```

Item_Cd
OTHER
OTHER
OTHER
OTHER
SPQE175
LMKE160
LMKG159
SPDE173

Figure 14.18

This query returned 66 records. Since you didn't group the data, it is returning one record for every row in the Sales_Journal table that meets the criteria. You see that the item code of "OTHER" appears several times, but you want each row to contain a unique value. Essentially, the record set should contain only the unique values that meet the specified criteria. Since you are not using an aggregate function, you can use SELECT DISTINCT.

7. *Edit the query to include the* ***DISTINCT*** *clause as shown in* **Figure 14.19** *and* **Run** *it.*



```
SELECT DISTINCT Item_Cd
FROM Sales_Journal
WHERE Store_ID = 12
AND YEAR(Sale_Date) = 2017
AND MONTH(Sale_Date) = 1;
```

Item_Cd
CMDB153
CMKE144
CMKG143
CMQB149
LMQE163
OTHER
SMDG119
SMKB113
SMQF114
SMQG115
SMTB125
SMTF122
SMTG123

Figure 14.19

Now the record set contains 30 distinct Item codes. You could have also used SELECT Item Code and GROUP BY Item Code. When you are running resource-intensive queries, I've found that using DISTINCT runs a bit faster than GROUP BY.

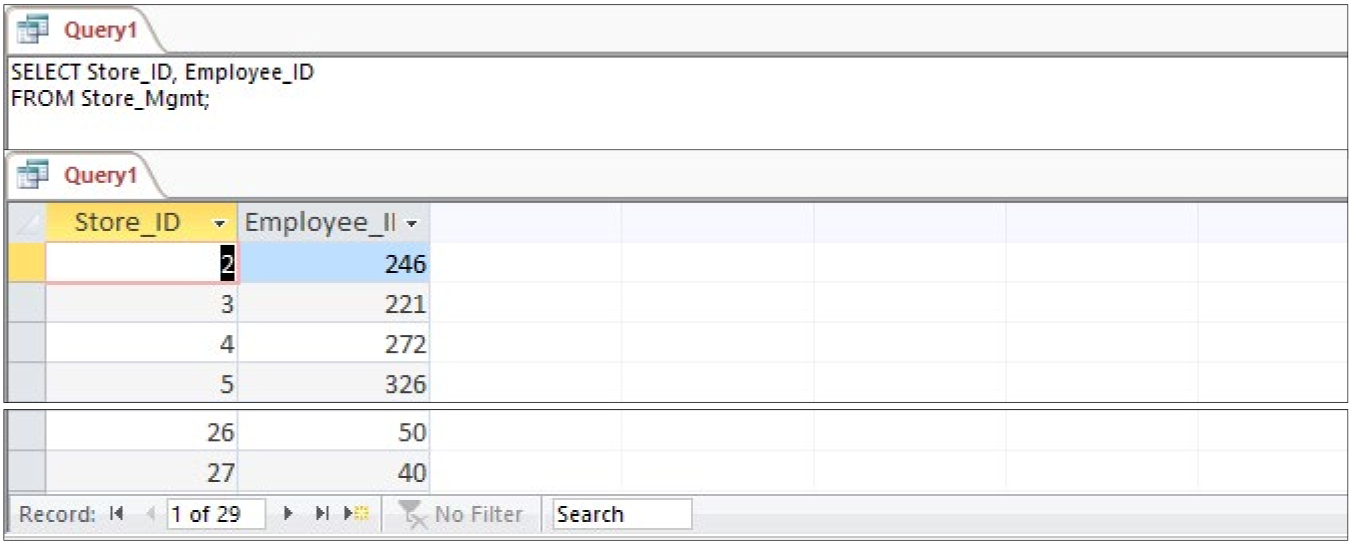
8. **Save** *the query as* ***qry14Distinct*** *and close it.*

Creating Relationships

You've already been educated in the concept of joins between tables in Access, so now I'll show you how to code it in ANSI SQL. Remember that the VLOOKUP() of Excel is the join of Access and SQL. In SQL, there are generally two ways to create a join. The first and easiest way (in my opinion) is to create the join in the WHERE clause. The second way is to write the join in the FROM statement. SQL purists will tell you to do joins in the FROM statement, but I believe it is easier to understand for non-programmers to create joins in the WHERE clause. However, most DBMSs today require the more complex joins (like one-way joins) to be written in the FROM clause. In the first example, I'll show you how to do it using both methods, but from there on out I'll do the joins in the FROM clause.

Let's begin with a simple query and build on it to illustrate how joins are coded. In this next example, you will write a query that pulls in the Store_ID and Employee_ID fields from the Store_Mgmt table. This table stores data that tells the name of the store manager of each store. The table stores ID numbers, not the actual store numbers or names of the employees, so we must create joins to find out the actual store numbers and names of the employees.

1. **Create** *a new query with* **SQL** *code as shown in Figure 14.20 and* **Run** *it.*



```
SELECT Store_ID, Employee_ID
FROM Store_Mgmt;
```

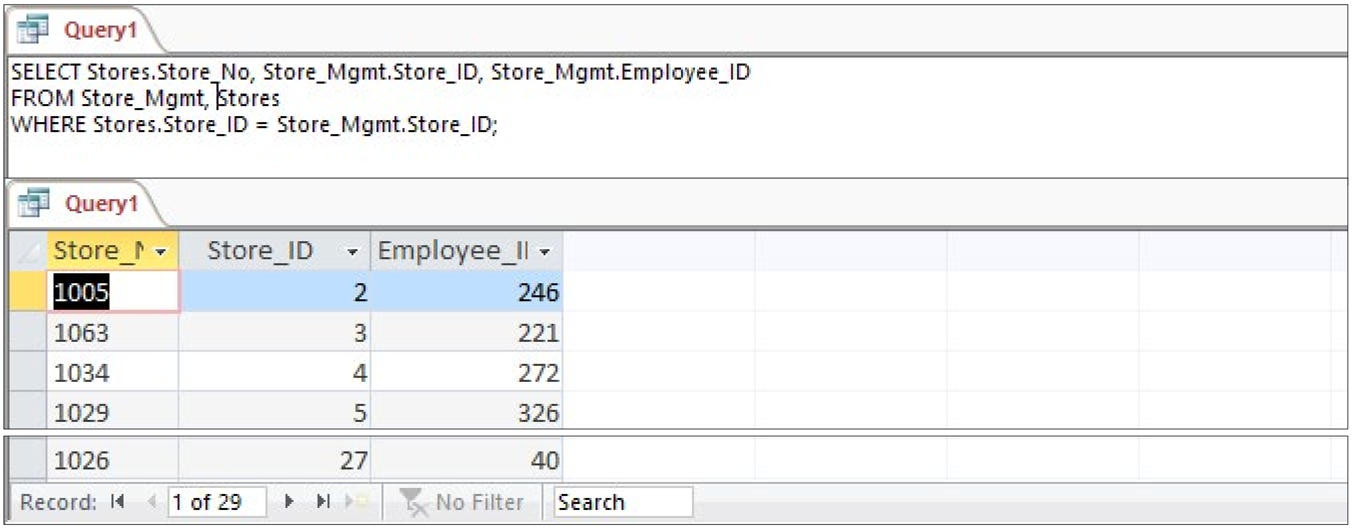
Store_ID	Employee_II
2	246
3	221
4	272
5	326
26	50
27	40

Figure 14.20

All you did was to write a query that brought in the only two fields in that table. The data in the table really doesn't tell you anything. To get the Store Number, you have to create a join with the Stores table, and the join will be done on the Store_ID. In doing this, you need to be familiar with the necessary coding when you bring in more than one table where the field names in both tables are the same. For example, in this next query, you will create a join on the Store_ID in the Store_Mgmt table with the Store_ID from the Stores table. When you have the same field names from different tables, you need to tell SQL which table you want to pull the field from. This is done by typing the table name followed by a period then the name of the field. This is called the field's path. This way, SQL knows which table to pull the data from. If the field name isn't the same as the other tables in the query, it isn't necessary to type in the field's path, but it does help programmers who aren't as familiar with the tables as you are. A table's full path is reflected as SERVER.OWNER.DATABASE.TABLE, but if the data is contained in the same database, all you have to type is the TABLE.FIELD_NAME, as you will see in the following exercise.

Let's first do the join in the WHERE statement as follows.

2. *Edit the code in the new query as shown in* **Figure 14.21** *and* **Run** *it.*



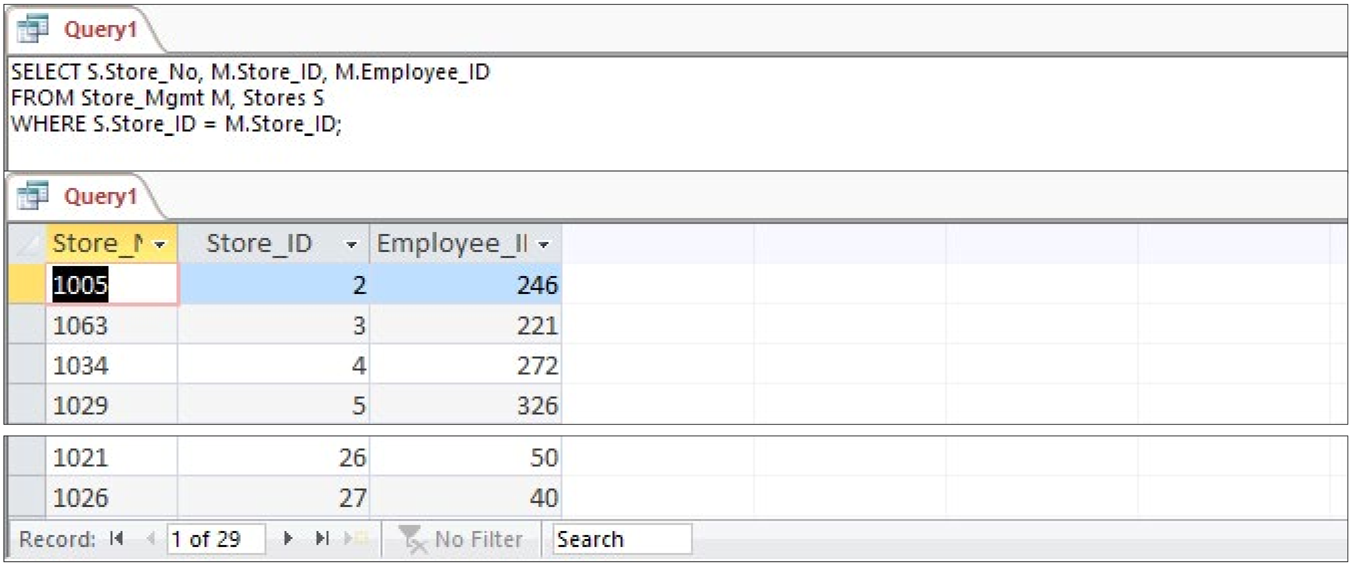
```
SELECT Stores.Store_No, Store_Mgmt.Store_ID, Store_Mgmt.Employee_ID
FROM Store_Mgmt, Stores
WHERE Stores.Store_ID = Store_Mgmt.Store_ID;
```

Store_N	Store_ID	Employee_I
1005	2	246
1063	3	221
1034	4	272
1029	5	326
1026	27	40

Figure 14.21

In this record set, you should get 29 records. As you may imagine, it would take a lot of typing the same table name over and over again if you had a lot of fields in your query. To cut down on the pain a bit, SQL allows you to use an alias for the name of a table. I like to use a one- or two-letter alias, as I don't like typing the same word over and over again. Let's call the Stores table "S" and the Store_Mgmt table "M". This way, we can use S and M in the code when referring to the tables. A programming purist will tell you that you need to use AS after the name of the table to create an alias. That isn't necessary unless there are fields, or other tables, with similar names. For this simple example, we won't use AS.

3. *Edit the code in the query as shown in* **Figure 14.22** *and* **Run** *it.*



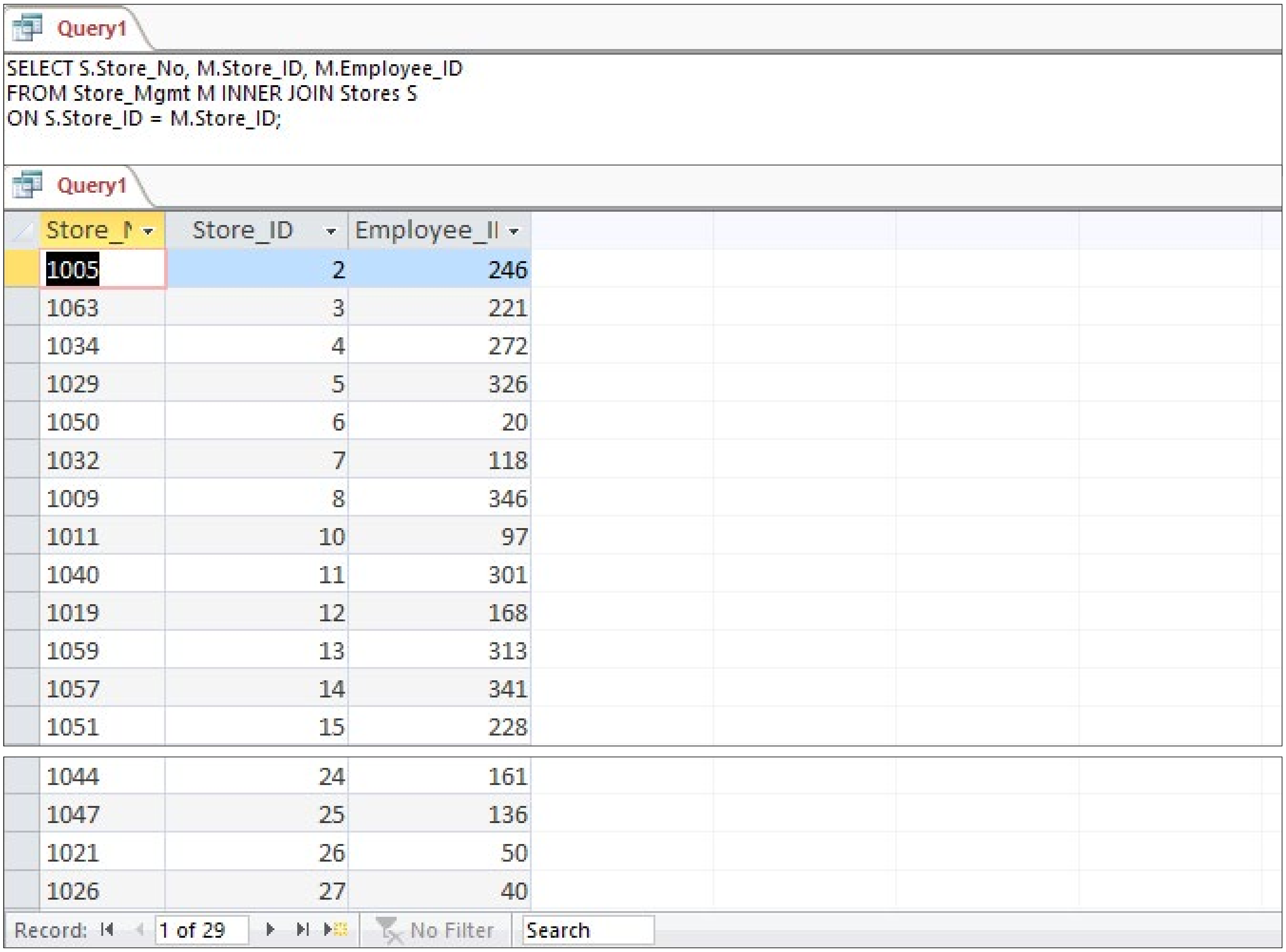
```
SELECT S.Store_No, M.Store_ID, M.Employee_ID
FROM Store_Mgmt M, Stores S
WHERE S.Store_ID = M.Store_ID;
```

Store_N	Store_ID	Employee_I
1005	2	246
1063	3	221
1034	4	272
1029	5	326
1021	26	50
1026	27	40

Figure 14.22

The result from the two queries should be exactly the same. Now let's convert the query to create the join in the FROM clause.

4. *Edit the code in the query as shown in* **Figure 14.23** *and* **Run** *it.*



```
SELECT S.Store_No, M.Store_ID, M.Employee_ID
FROM Store_Mgmt M INNER JOIN Stores S
ON S.Store_ID = M.Store_ID;
```

Store_N	Store_ID	Employee_I
1005	2	246
1063	3	221
1034	4	272
1029	5	326
1050	6	20
1032	7	118
1009	8	346
1011	10	97
1040	11	301
1019	12	168
1059	13	313
1057	14	341
1051	15	228
1044	24	161
1047	25	136
1021	26	50
1026	27	40

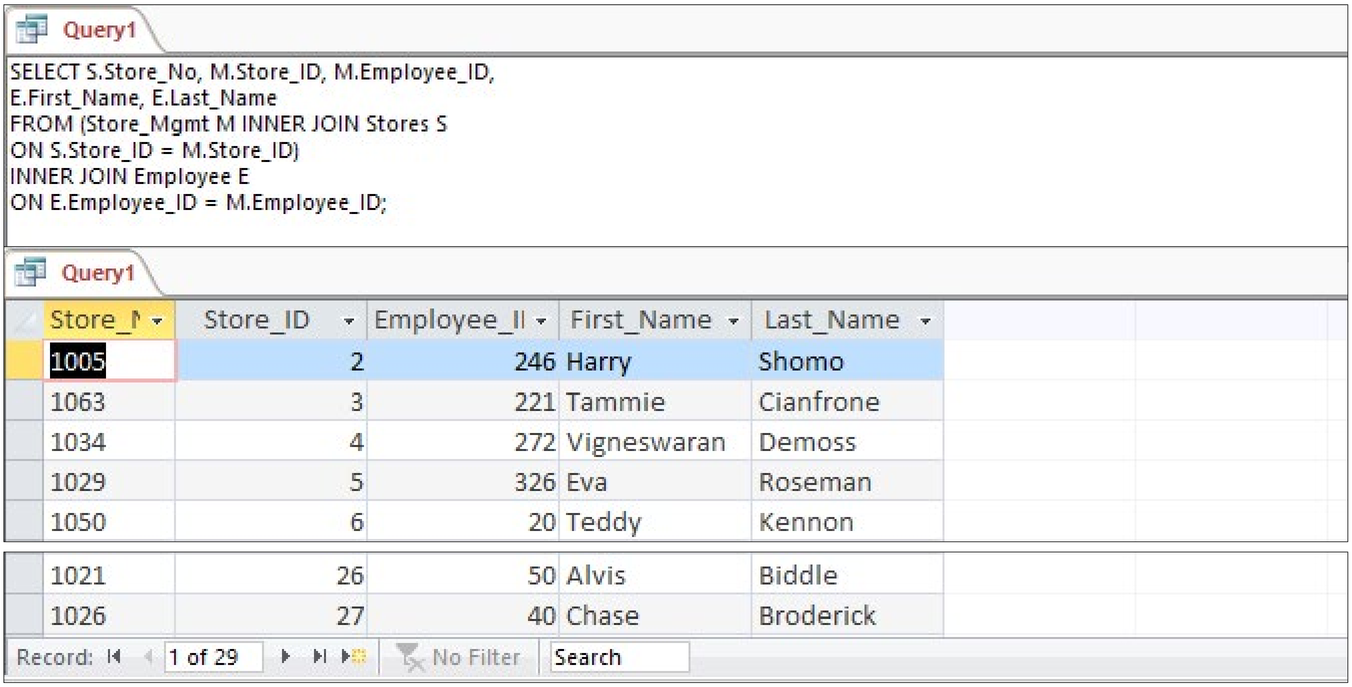
Figure 14.23

You should get exactly the same results as in Figure 14.22 and 14.23.

Joins on Multiple Tables

In SQL, there are few limits on how many joins you can do in one query. However, keep in mind that having too many joins could adversely affect the query's performance. The syntax for joining more than two tables is the same as on just two tables. Now let's enhance our query to bring in the Employee's name from the Employee table, and join to the Store_Mgmt table on Employee_ID. You will assign the Employee table an alias of "E".

5. *Edit the code in the query as shown in* **Figure 14.24** *and* **Run** *it.*



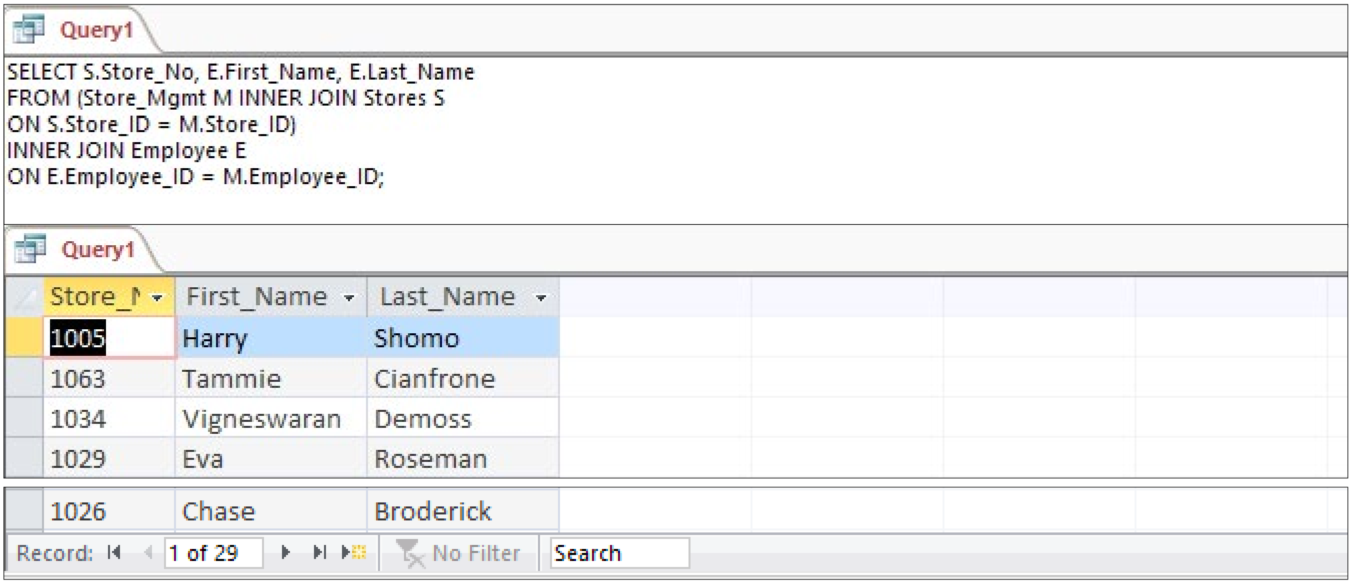
```
SELECT S.Store_No, M.Store_ID, M.Employee_ID,
E.First_Name, E.Last_Name
FROM (Store_Mgmt M INNER JOIN Stores S
ON S.Store_ID = M.Store_ID)
INNER JOIN Employee E
ON E.Employee_ID = M.Employee_ID;
```

Store_N	Store_ID	Employee_I	First_Name	Last_Name
1005	2	246	Harry	Shomo
1063	3	221	Tammie	Cianfrone
1034	4	272	Vigneswaran	Demoss
1029	5	326	Eva	Roseman
1050	6	20	Teddy	Kennon
1021	26	50	Alvis	Biddle
1026	27	40	Chase	Broderick

Figure 14.24

You should still get 29 records, but now the record set includes the first name and last name of the employee who is the manager of each store. Even though you are joining the tables on Store_ID and Employee_ID, it is not necessary to have those fields in the SELECT statement, as they really don't add anything to the query. Sometimes it's helpful to have them show up when you run the query to check your numbers, but once the query is functioning, you may not need them. Let's take them out.

6. *Edit the code in the query as shown in* **Figure 14.25** *and* **Run** *it.*



```
SELECT S.Store_No, E.First_Name, E.Last_Name
FROM (Store_Mgmt M INNER JOIN Stores S
ON S.Store_ID = M.Store_ID)
INNER JOIN Employee E
ON E.Employee_ID = M.Employee_ID;
```

Store_N	First_Name	Last_Name
1005	Harry	Shomo
1063	Tammie	Cianfrone
1034	Vigneswaran	Demoss
1029	Eva	Roseman
1026	Chase	Broderick

Figure 14.25

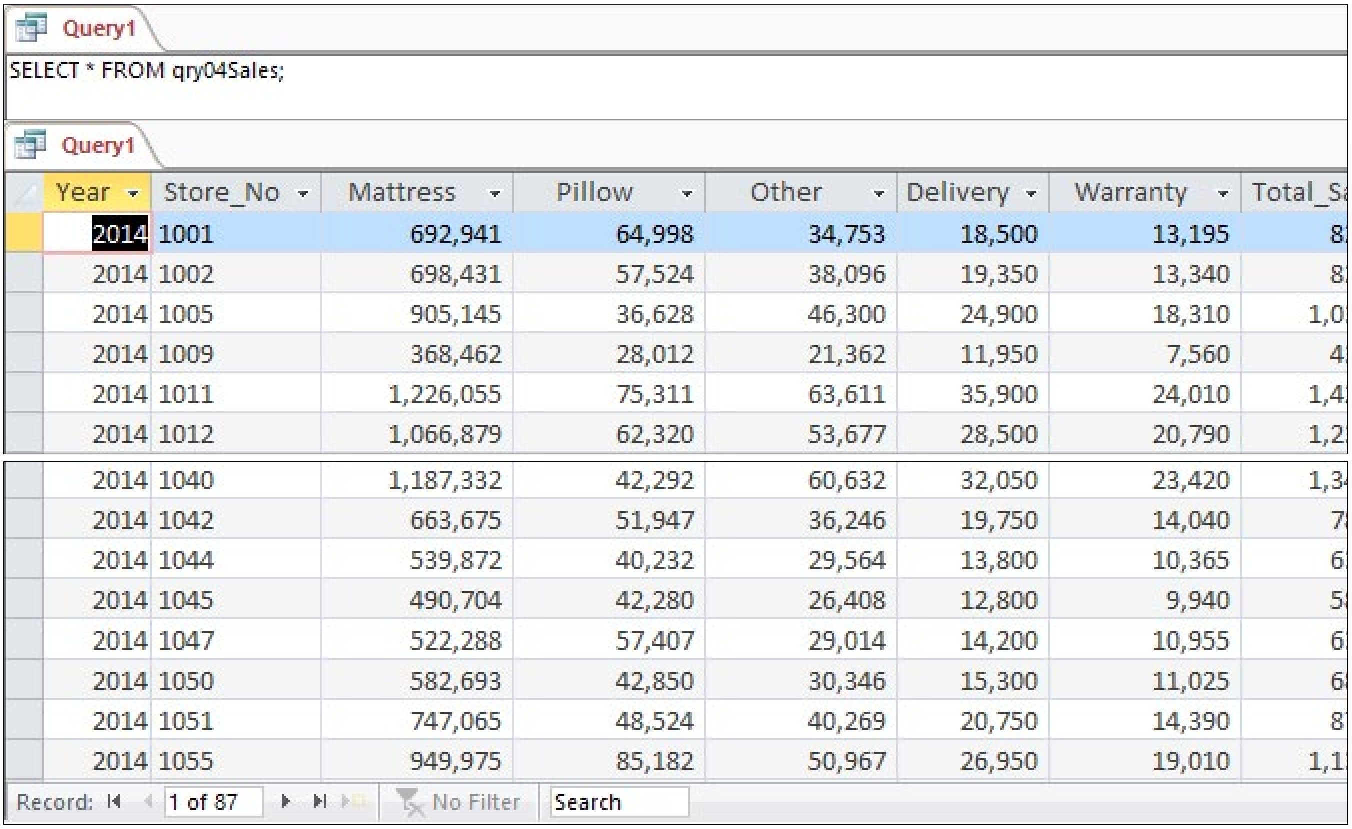
Now notice in this query that the primary table is the Store_Mgmt table, as that is the table that contains all the necessary information, yet we're not bringing in ANY of the fields from that table. This kind of table is referred to as a Union table, or a table that brings the necessary information together in one query.

7. **Save** *the query as* ***qry14Joins*** *and close it.*

One Way (or Outer) Joins

In Chapter Four, you learned about one-way or outer joins. I won't review the necessity of these kinds of joins, as you should already understand the concept. If you don't remember, review Chapter Four. In this next exercise, you'll use the same query (qry04Sales) and table (Sales_Budget) that you used to design qry4Sales_Budget, but here you'll design it using SQL code. You can use an Access query in SQL code just like to use a table, as long as you are writing the SQL code within the Access database. First let's review the data in qry04Sales.

8. **Create** *a new query with SQL code as illustrated in* **Figure 14.26** *and* **Run** *it.*



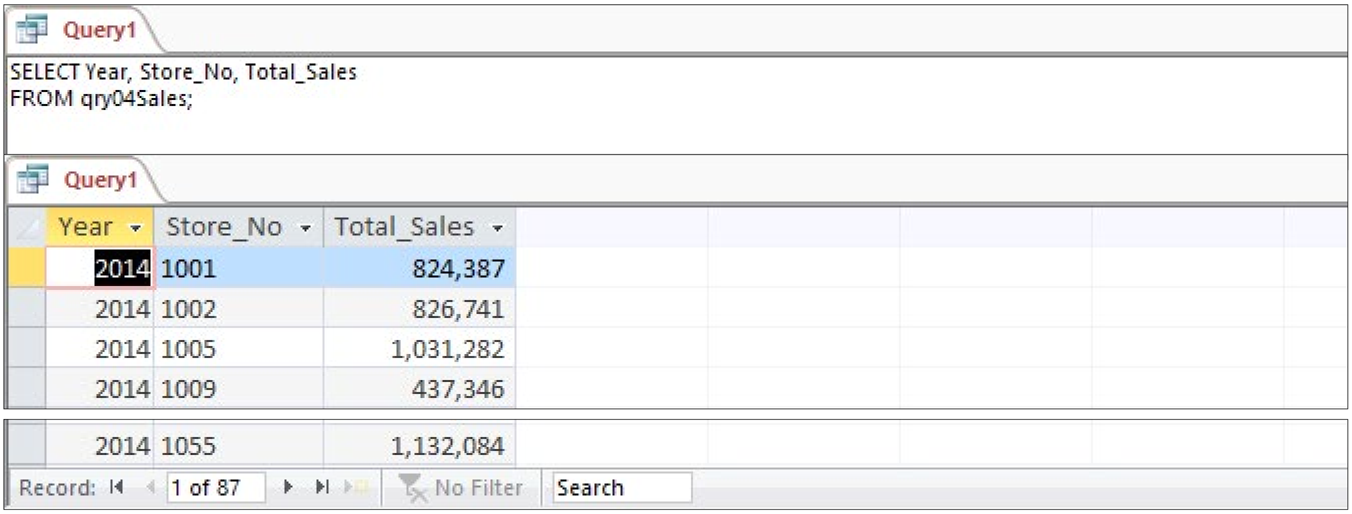
```
SELECT * FROM qry04Sales;
```

Year	Store_No	Mattress	Pillow	Other	Delivery	Warranty	Total_Sa
2014	1001	692,941	64,998	34,753	18,500	13,195	8
2014	1002	698,431	57,524	38,096	19,350	13,340	8
2014	1005	905,145	36,628	46,300	24,900	18,310	1,0
2014	1009	368,462	28,012	21,362	11,950	7,560	4
2014	1011	1,226,055	75,311	63,611	35,900	24,010	1,4
2014	1012	1,066,879	62,320	53,677	28,500	20,790	1,2
2014	1040	1,187,332	42,292	60,632	32,050	23,420	1,3
2014	1042	663,675	51,947	36,246	19,750	14,040	7
2014	1044	539,872	40,232	29,564	13,800	10,365	6
2014	1045	490,704	42,280	26,408	12,800	9,940	5
2014	1047	522,288	57,407	29,014	14,200	10,955	6
2014	1050	582,693	42,850	30,346	15,300	11,025	6
2014	1051	747,065	48,524	40,269	20,750	14,390	8
2014	1055	949,975	85,182	50,967	26,950	19,010	1,1

Figure 14.26

The dataset reflects annual sales by year and by store, and contains 87 rows of data. Let's edit the query to bring in just the Year, Store_No, and Total_Sales.

9. *Edit the code in the query as shown in* **Figure 14.27** *and* **Run** *it.*



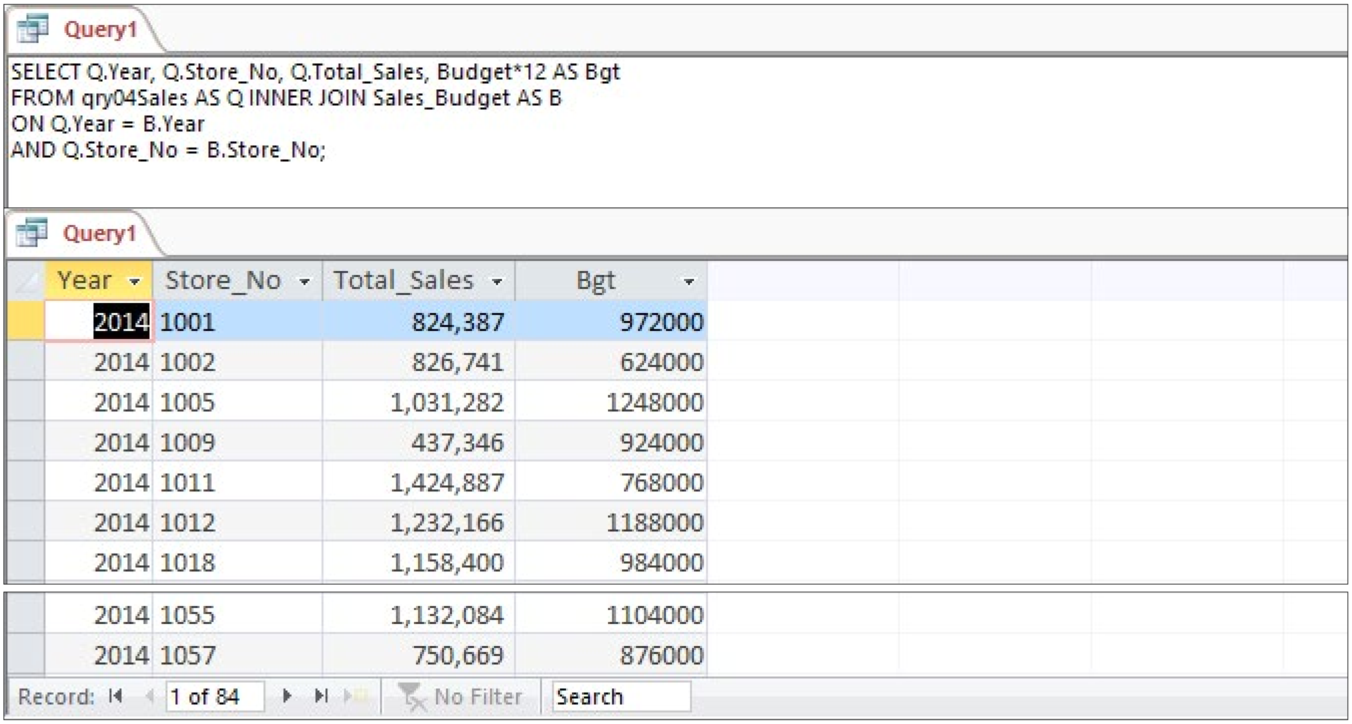
```
SELECT Year, Store_No, Total_Sales
FROM qry04Sales;
```

Year	Store_No	Total_Sales
2014	1001	824,387
2014	1002	826,741
2014	1005	1,031,282
2014	1009	437,346
2014	1055	1,132,084

Figure 14.27

The dataset still contains 87 records. Now you will bring in the Sales_Budget table and join it by Year and Store_No. You'll use "Q" as the alias for qry04Sales and "B" as the alias for the Sales_Budget table. You'll bring in the Budget field and make the budget numbers be annual. Remember that the budget numbers in the Sales_Budget table are monthly numbers, so you need to multiply the budget by 12 to get an annual number that is comparable to total annual sales.

10. *Edit the code in the query as shown in* **Figure 14.28** *and* **Run** *it.*



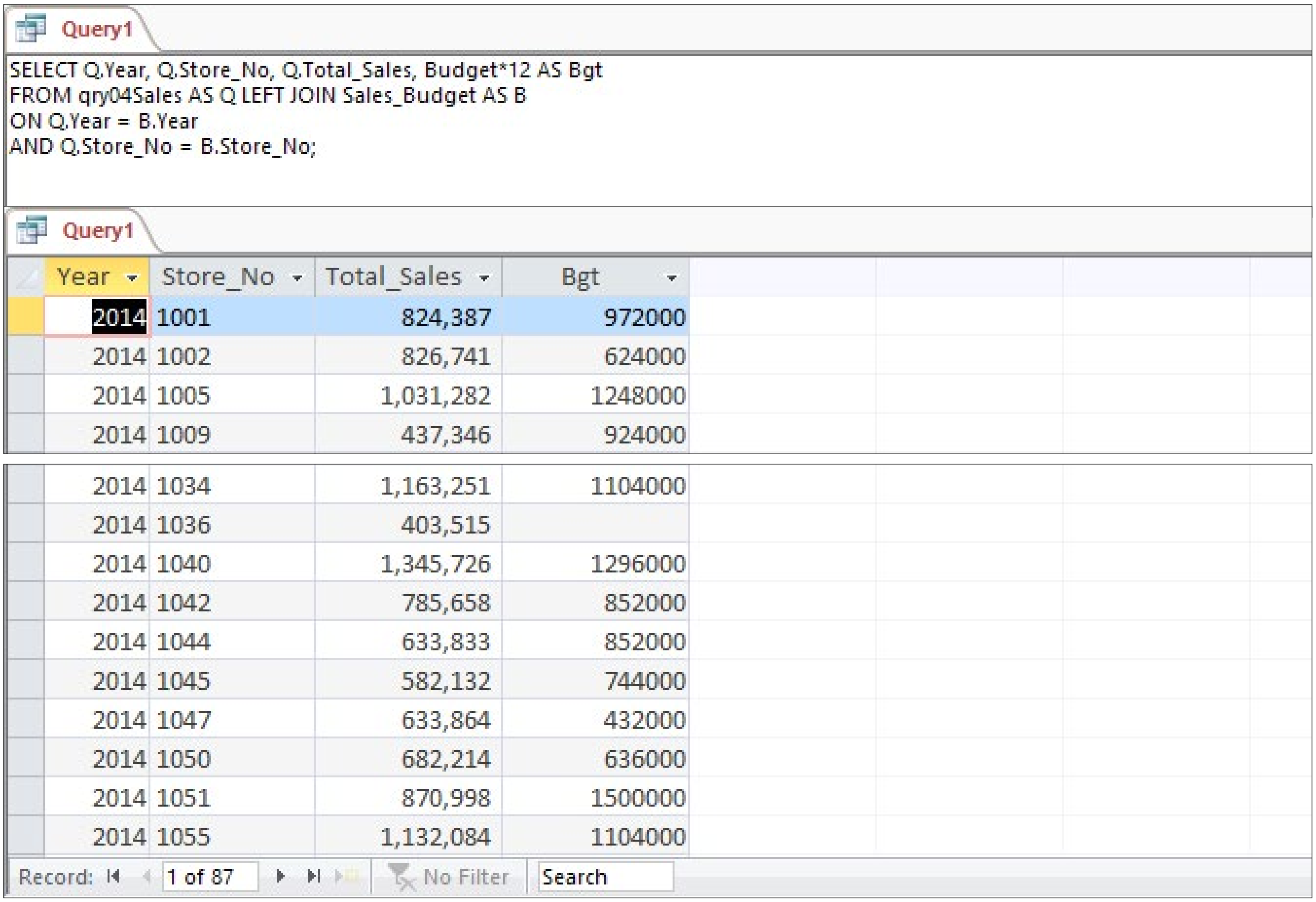
```
SELECT Q.Year, Q.Store_No, Q.Total_Sales, Budget*12 AS Bgt
FROM qry04Sales AS Q INNER JOIN Sales_Budget AS B
ON Q.Year = B.Year
AND Q.Store_No = B.Store_No;
```

Year	Store_No	Total_Sales	Bgt
2014	1001	824,387	972000
2014	1002	826,741	624000
2014	1005	1,031,282	1248000
2014	1009	437,346	924000
2014	1011	1,424,887	768000
2014	1012	1,232,166	1188000
2014	1018	1,158,400	984000
2014	1055	1,132,084	1104000
2014	1057	750,669	876000

Figure 14.28

You will notice that, just like in the example in Chapter Four, only 84 records appear. That is because there is no budget for Store_No 1036. To include ALL records from qry04Sales and the MATCHED records from the Sales_Budget table, you need to do an outer join, or in this case, a LEFT join.

11. *Edit the code in the query as shown in* **Figure 14.29** *and* **Run** *it.*



```
SELECT Q.Year, Q.Store_No, Q.Total_Sales, Budget*12 AS Bgt
FROM qry04Sales AS Q LEFT JOIN Sales_Budget AS B
ON Q.Year = B.Year
AND Q.Store_No = B.Store_No;
```

Year	Store_No	Total_Sales	Bgt
2014	1001	824,387	972000
2014	1002	826,741	624000
2014	1005	1,031,282	1248000
2014	1009	437,346	924000
2014	1034	1,163,251	1104000
2014	1036	403,515	
2014	1040	1,345,726	1296000
2014	1042	785,658	852000
2014	1044	633,833	852000
2014	1045	582,132	744000
2014	1047	633,864	432000
2014	1050	682,214	636000
2014	1051	870,998	1500000
2014	1055	1,132,084	1104000

Figure 14.29

As you can see, the budget for Store No 1036 is now showing up as a NULL value, and there are now 87 records in the recordset.

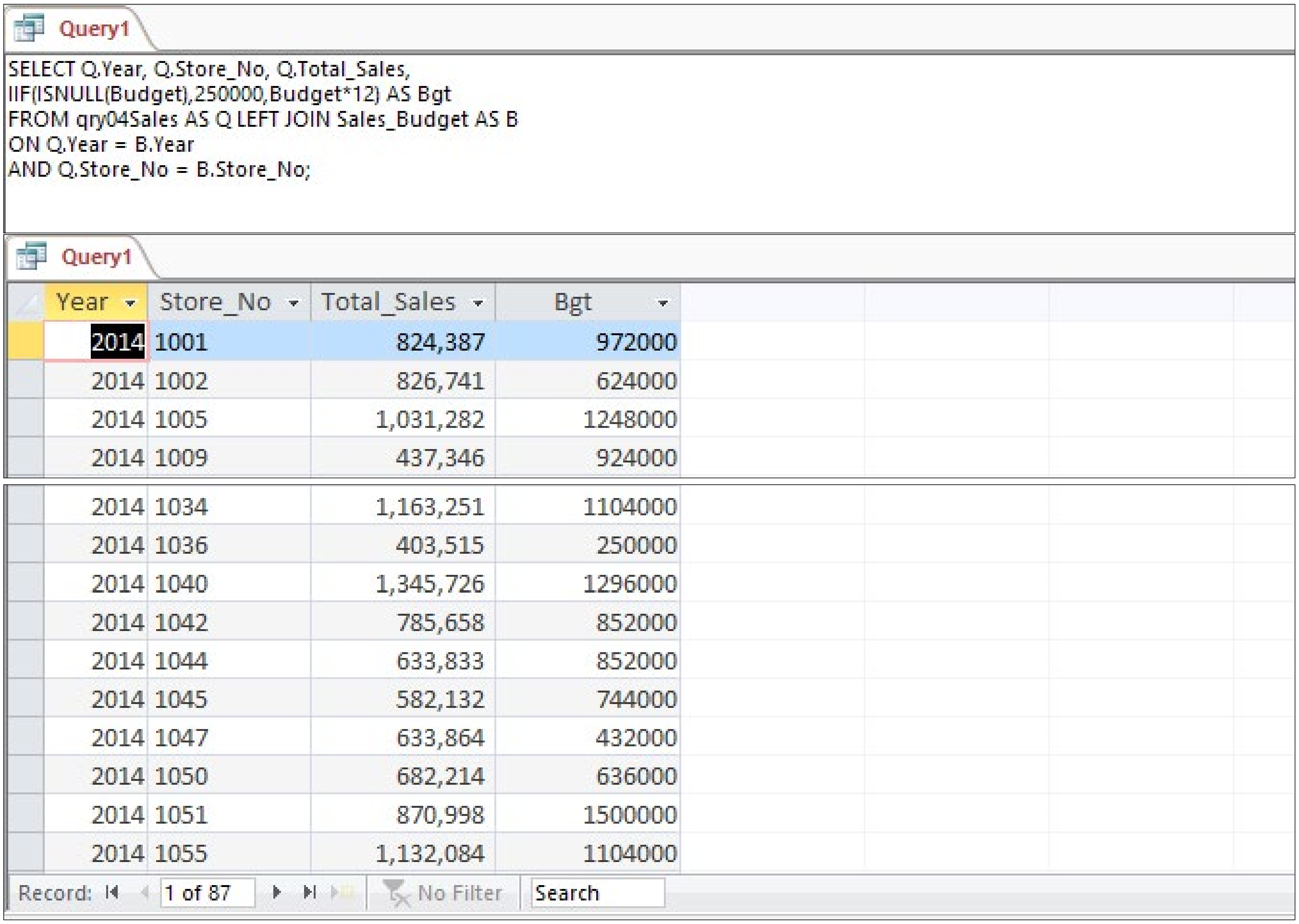
Some older versions of SQL allow you to do the one-way join in the WHERE clause where the syntax is as follows:

*SELECT Q.Year, Q.Store_No, Total_Sales, Budget*12 AS Bgt*
FROM qry4Sales Q, Sales_Budget B
*WHERE Q.Store_No *= B.Store_No*
*AND Q.Year *= B.Year;*

CASE and IIF() Statements

Another function that is extremely useful in SQL is the CASE statement. You've already learned how to use an IIF() function in Access, and you must use an IIF() statement when using SQL code in Access. But if you were coding an IIF() statement in SQL Server, you would use a CASE statement. Let's assume that in our example, we want to replace all NULL values in the Budget field with a budget of $250,000.

12. *Edit the code in the query as shown in* **Figure 14.30** *and* **Run** *it.*



```
SELECT Q.Year, Q.Store_No, Q.Total_Sales,
IIF(ISNULL(Budget),250000,Budget*12) AS Bgt
FROM qry04Sales AS Q LEFT JOIN Sales_Budget AS B
ON Q.Year = B.Year
AND Q.Store_No = B.Store_No;
```

Year	Store_No	Total_Sales	Bgt
2014	1001	824,387	972000
2014	1002	826,741	624000
2014	1005	1,031,282	1248000
2014	1009	437,346	924000
2014	1034	1,163,251	1104000
2014	1036	403,515	250000
2014	1040	1,345,726	1296000
2014	1042	785,658	852000
2014	1044	633,833	852000
2014	1045	582,132	744000
2014	1047	633,864	432000
2014	1050	682,214	636000
2014	1051	870,998	1500000
2014	1055	1,132,084	1104000

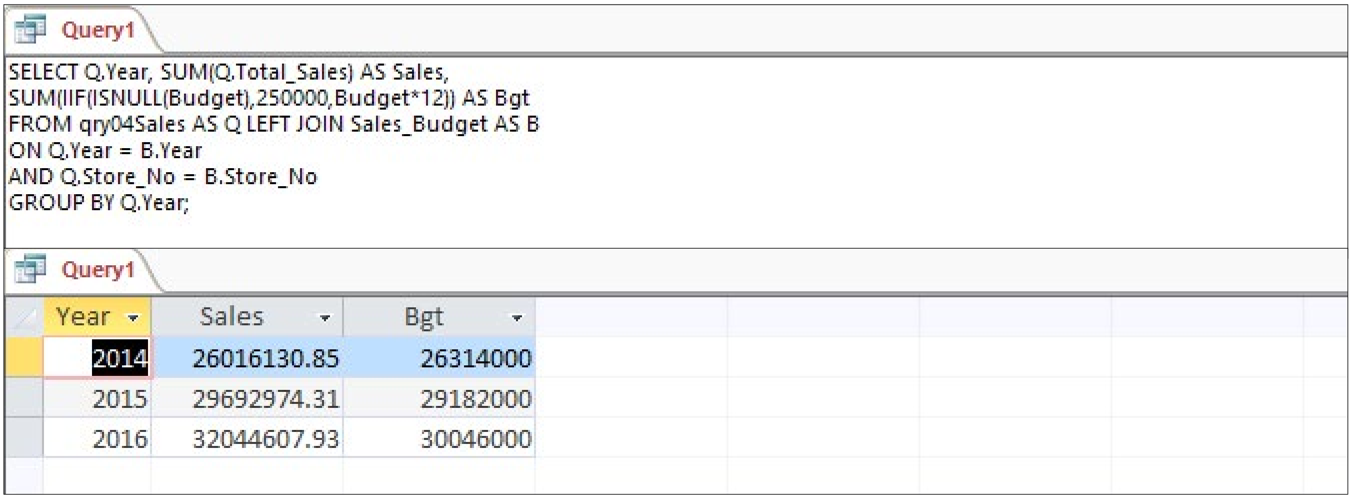
Figure 14.30

If you scroll down to Store_No 1036, you'll see that it has a budget of $250,000. I typically don't like to hard-code in these kinds of assumptions into my programming code, but I did it in this simple example to show you the power of an IIF() statement. You would use a CASE statement if you wanted to do the same IIF() statement in SQL. The syntax for that portion of the code is below:

CASE WHEN Budget =‘’ THEN 250000 ELSE Budget*12 END AS Bgt

You can also use a one-way join in combination with aggregate functions. Let's suppose you want to add up the total sales and budget for all stores by year.

13. *Edit the code in the query as shown in* **Figure 14.31** *and* **Run** *it.*



```
SELECT Q.Year, SUM(Q.Total_Sales) AS Sales,
SUM(IIF(ISNULL(Budget),250000,Budget*12)) AS Bgt
FROM qry04Sales AS Q LEFT JOIN Sales_Budget AS B
ON Q.Year = B.Year
AND Q.Store_No = B.Store_No
GROUP BY Q.Year;
```

Year	Sales	Bgt
2014	26016130.85	26314000
2015	29692974.31	29182000
2016	32044607.93	30046000

Figure 14.31

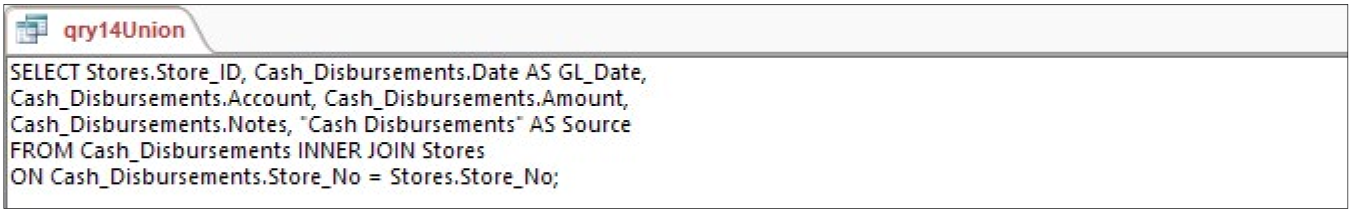
14. **Save** *the query as* ***qry14OneWay_IIF*** *and close.*

Union Queries

Sometimes you will want to query a table, or tables, multiple times to get one record set, or query different sources of data, and put it into one query. You saw an example of that in Chapter Five. In that chapter, you tested the entries booked to the GL from the Cash_Disbursements and the Discount_Journal tables. You first created a Make-Table query using the Cash_Disbursements table joined to the Stores table, then you appended Discounts from the Discount_Journal joined to the Sales_Journal table. If you were to do one query to return both sets of data in an Access SELECT query, you'd have a hard time doing so. However, in SQL, you can create two separate queries and join them with a UNION clause. It's simple enough to do – just run the two queries separated by a UNION clause.

To show you how a UNION query works, you will copy qry05GL_Test and qry05Append_DJ queries, turn them into SELECT statements, then combine the codes via a UNION query.

1. **Copy** *the* **qry05GL_Test** *query and name the new query* ***qry14Union****.*
2. *In* **Design View** *of* **qry14Union***, click on the* **Select** *icon (to make it a Select query).*
3. *View the* **SQL** *code of that query.*



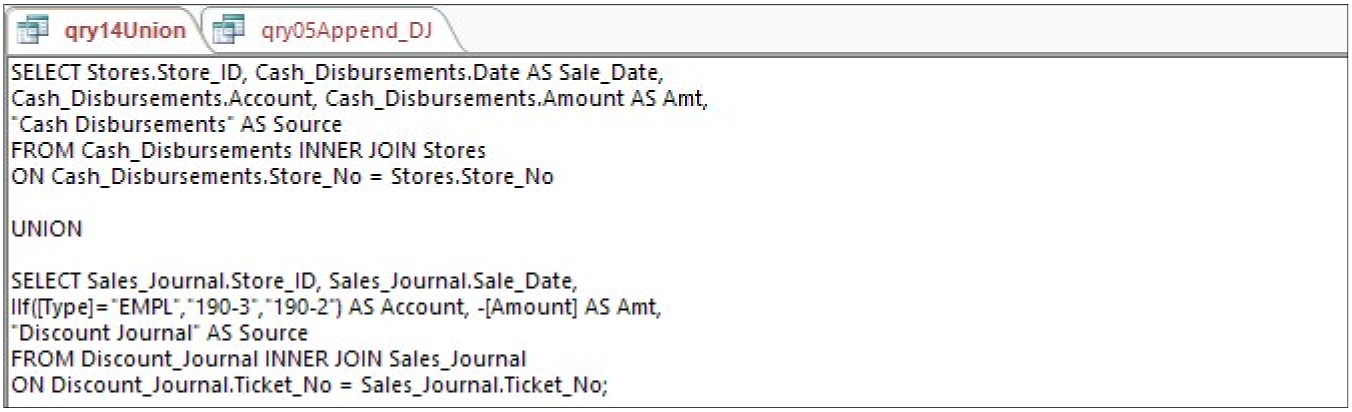
```
SELECT Stores.Store_ID, Cash_Disbursements.Date AS GL_Date,
Cash_Disbursements.Account, Cash_Disbursements.Amount,
Cash_Disbursements.Notes, "Cash Disbursements" AS Source
FROM Cash_Disbursements INNER JOIN Stores
ON Cash_Disbursements.Store_No = Stores.Store_No;
```

Figure 14.32

4. *Open* **qry05Append_DJ** *in* **Design View**.
5. *Make the query be a* **Select** *query and view the* **SQL** *code.*
6. **Copy** *the* **SQL** *code from* **qry05Append_DJ** *to below the* **SQL** *code in* **qry14Union**, *separated by the word* ***UNION***, *as in* **Figure 14.33**.

In a UNION query, both queries need to have the same column names, so you need to edit the SQL code from the qry14Union query to match the fields in the qry05Append_DJ query.

7. *Change the* **GL_Date** *alias to* ***Sale_Date***.
8. *Make the* **Cash_Disbursements.Amount** *field have an alias of "****Amt****".*
9. **Delete** *the* **Cash_Disbursements.Notes** *field.*

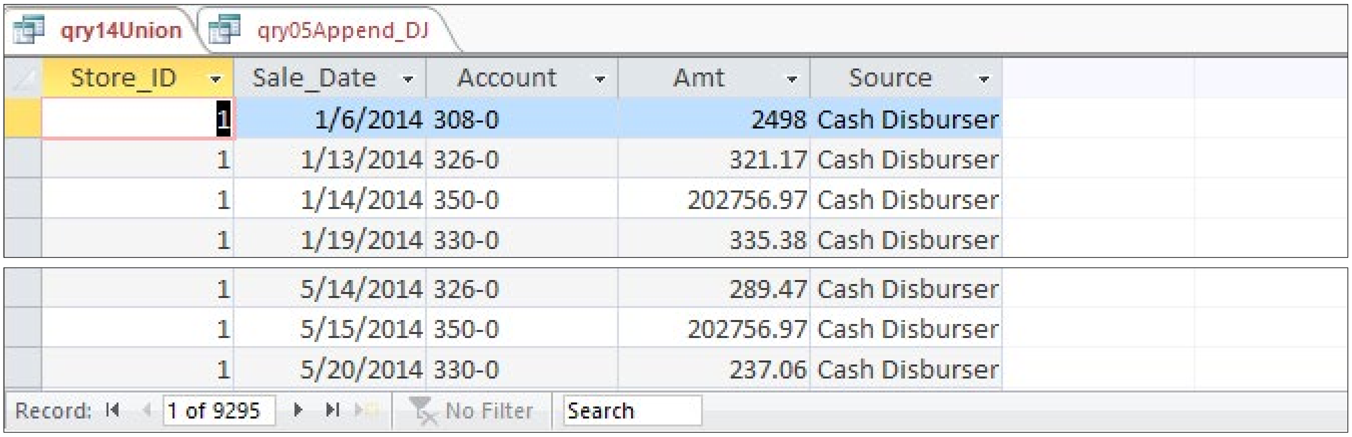


```
SELECT Stores.Store_ID, Cash_Disbursements.Date AS Sale_Date,
Cash_Disbursements.Account, Cash_Disbursements.Amount AS Amt,
"Cash Disbursements" AS Source
FROM Cash_Disbursements INNER JOIN Stores
ON Cash_Disbursements.Store_No = Stores.Store_No

UNION

SELECT Sales_Journal.Store_ID, Sales_Journal.Sale_Date,
IIf([Type]="EMPL","190-3","190-2") AS Account, -[Amount] AS Amt,
"Discount Journal" AS Source
FROM Discount_Journal INNER JOIN Sales_Journal
ON Discount_Journal.Ticket_No = Sales_Journal.Ticket_No;
```

Figure 14.33

Make sure the semi-colon appears AFTER the last line of the SQL Code. The semi-colon tells SQL that you are at the end of the SQL statement.

10. **Run** *the* **SQL** *Code for* **qry14Union**.



Store_ID	Sale_Date	Account	Amt	Source
1	1/6/2014	308-0	2498	Cash Disburser
1	1/13/2014	326-0	321.17	Cash Disburser
1	1/14/2014	350-0	202756.97	Cash Disburser
1	1/19/2014	330-0	335.38	Cash Disburser
1	5/14/2014	326-0	289.47	Cash Disburser
1	5/15/2014	350-0	202756.97	Cash Disburser
1	5/20/2014	330-0	237.06	Cash Disburser

Figure 14.34

If you scroll down, you will see you have 9,295 records, which contain a mixture of Cash Disbursement and Discount Journal records. A few rules about a UNION query:

- It must contain at least two SELECT statements, each separated by the word UNION;
- Each field in all SELECT statements must be named the same;
- The data in each field must be compatible (meaning it must have the same data type).

Not only is a UNION query easy to do, but as you can see, you can design the query in Access Design View, and use the SQL code it generated. But be careful! UNION queries can take up a lot of resources, so use them sparingly. Also, be aware that a UNION query will remove duplicate records that are produced by each query separately. If you want to include the duplicate records, be sure to use UNION ALL instead of just UNION by itself.

11. *Close* **qry05Append_DJ** *without saving it.*
12. **Save** *and close* **qry14Union**.

When you save the UNION query, look at the design of the query in the Queries pane. Since it is a UNION query, it appears like this: qry14Union

> ***Review Questions****: It is now time to complete the hands-on Review Questions. Log on to www.ExcelCEO.com with your Email and Password, click on the* **Access 2016 and SQL Review Questions, Chapter 14, Section 2 of 2** *option and complete the review questions.*

SQL Conclusion

There are many other things you can do with SQL. It is my intent to show you how learning a few phrases of SQL code can help you with extracting data from various databases. Typically, a non-programmer will not perform advanced procedures such as triggers and inserting, editing, and deleting data from a SQL Server database. Those functions are typically performed by the Database Administrators (DBAs). However, there is a great need for business people to know how to query the needed data, and to analyze and report on it. At this point, you should buy a comprehensive SQL reference book to keep in your personal IT library. You now know the basics of SQL, and anything else you should be able to find in such a reference manual (or from my good friend, Google), if the necessity arises.

Chapter Exam

You can now go to www.ExcelCEO.com, click on Sign In, log in and take the exam. Make sure that you take the exam on the same device on which you completed the practice files, as some of the questions on the exam may refer to some of the completed projects. Chapter exams are intended to be hands-on.

Access® 2016 and SQL

Complete Self-study Course

ExcelCEO

Chief Excel Officer

CHAPTER FIFTEEN — EXCEL AND MICROSOFT QUERY

In this chapter, you will:

- Identify SQL Code in Excel and Microsoft Query
- Determine how to use Microsoft Query's GUI screen
- Recognize SQL code in Microsoft Query's Design grid to include Parameters on an Excel file

CPE Credits possible for this chapter: 3

Excel and Microsoft Query

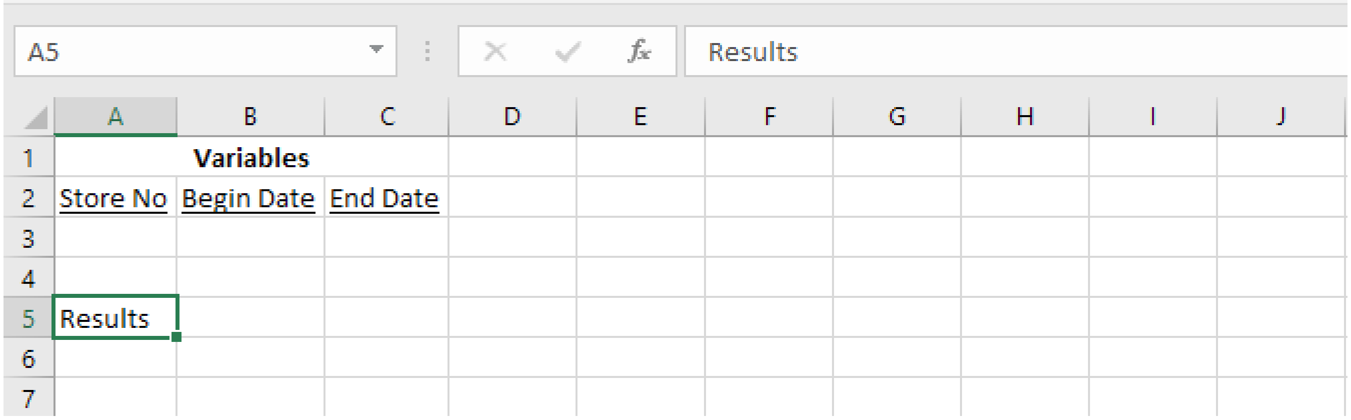
In the last couple of chapters, you've seen how to write SQL code in an Access database, and you may have thought to yourself, "*I don't see a whole lot of need for SQL in Access since you can do most of the query design using the Design tool.*" If you thought that, you are mostly right. The Access Query Design screen is a great place to design your query, if you are more comfortable with that GUI environment. When I started programming in SQL, I never thought I would prefer to write code than to use the Design View. However, the more I wrote code, the more I could see how it could help refine my queries. Simple things like UNION queries and writing the code for a Drop Table query helped me to program queries that are not available using the Access GUI screen. Similarly, you can use SQL code in Excel via Microsoft Query to query information directly from a SQL Server database or other DBMSs.

One time I had a project where I wanted to design an Excel file where I could input the variables onto the spreadsheet and have the variables passed back to the SQL code behind the spreadsheet and have the code filter the data based on the variables on the spreadsheet. I knew how to connect to a SQL Server database using a DSN, but I didn't know how to pass the variables back to the SQL code. I researched it various times on the Internet with no luck. There were a couple of programmers that wrote some extremely complex ADO code to accomplish the task, but that was way too complicated for what I wanted to do. I kept thinking, "*There's GOT to be an easier way to do this.*" Usually when I think that, there IS an easier way. So I started playing around with the code and found the answer. I'll explain how I did it in the following exercises.

In the Excel course, you learned about advanced PivotTables, and how to connect to an external database and use the data in a PivotTable. In Chapter Ten of this course, you learned how to create a DSN to connect to a SQL Server database. In this next exercise, you will connect to the SQL Server database and pull the results directly into an Excel spreadsheet. Then you will modify the SQL code behind the Excel file to filter the data.

Let's assume that there is an accountant at the Nitey-Nite Home Office who wants to query the Sales_Journal table and return detail sales by store for a particular date range. In this example, we have three variables: Store_No, Begin Date, and End Date. She doesn't have the SQL Server software (to allow direct connectivity to SQL Server) and doesn't know how to use Access. She just wants to be able to input the variables into an Excel spreadsheet and click a Run button to return a dataset directly into Excel below the variables. (She doesn't know what she's missing by not learning how to do this on her own, but people like that keep people like us fully employed.) Let's set up the basic spreadsheet.

1. *Open* **Excel** *to a* **Blank workbook**.
2. *Make* **Sheet1** *look like* **Figure 15.1**.



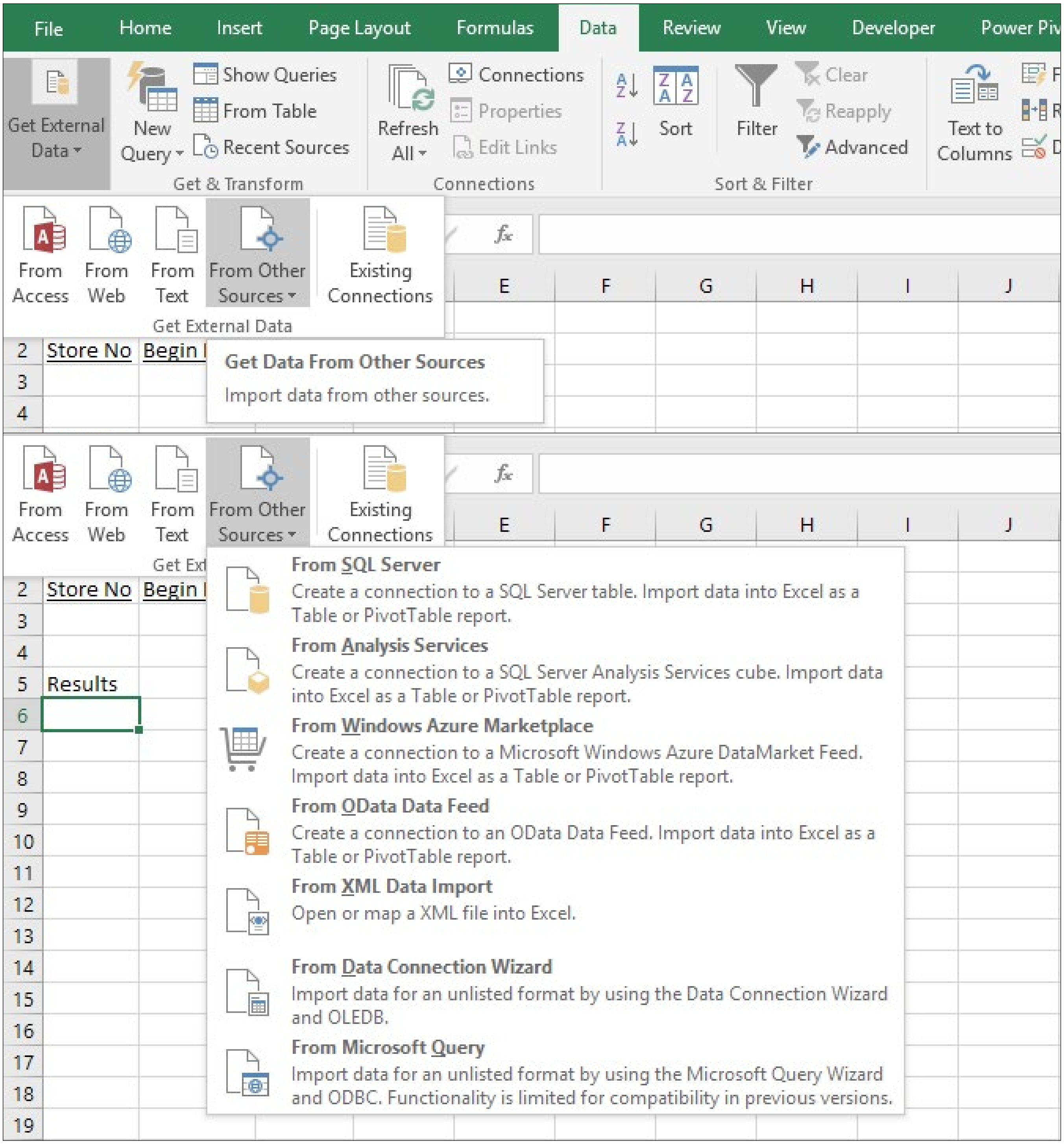
	A	B	C	D	E	F	G	H	I	J
1		**Variables**								
2	Store No	Begin Date	End Date							
3										
4										
5	Results									
6										
7										

Figure 15.1

3. **Save** *the workbook as* **C:\ExcelCEO\Access 2016\*Excel_Query.xlsx*.**

We want to input criteria into Cells A3, B3, and C3, click a button, and have the results populate in Cells A6 through the end of the file. Let's first work on getting some results populated, then we'll go back and adjust the query. To do this, we need to get external data through the Nitey_Nite DSN we created in Chapter 10.

4. *Click on* **Cell A6**.
5. *Click on the* **Data** *tab then click on the* **From Other Sources** From Other Sources *icon in the* **Get External Data** *group.*



Get Data From Other Sources
Import data from other sources.

From SQL Server
Create a connection to a SQL Server table. Import data into Excel as a Table or PivotTable report.

From Analysis Services
Create a connection to a SQL Server Analysis Services cube. Import data into Excel as a Table or PivotTable report.

From Windows Azure Marketplace
Create a connection to a Microsoft Windows Azure DataMarket Feed. Import data into Excel as a Table or PivotTable report.

From OData Data Feed
Create a connection to an OData Data Feed. Import data into Excel as a Table or PivotTable report.

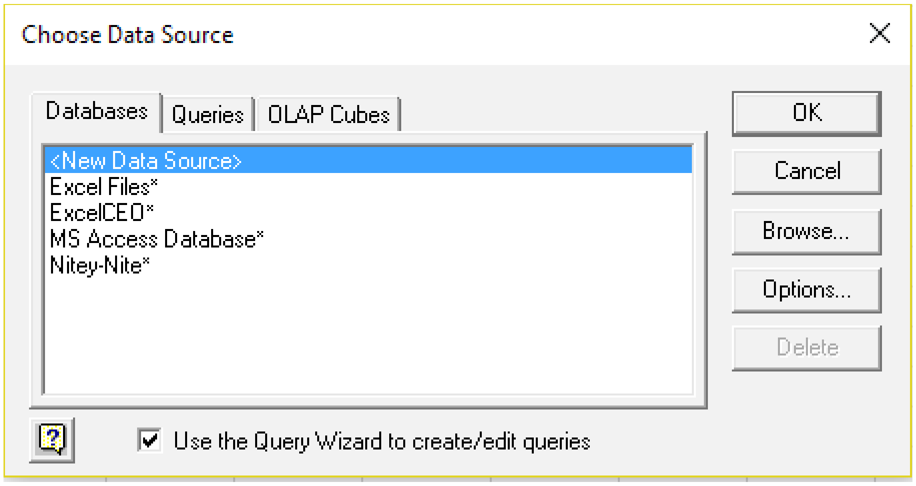
From XML Data Import
Open or map a XML file into Excel.

From Data Connection Wizard
Import data for an unlisted format by using the Data Connection Wizard and OLEDB.

From Microsoft Query
Import data for an unlisted format by using the Microsoft Query Wizard and ODBC. Functionality is limited for compatibility in previous versions.

Figure 15.2

6. *Click on the* **From Microsoft Query** *option.*



<New Data Source>
Excel Files*
ExcelCEO*
MS Access Database*
Nitey-Nite*

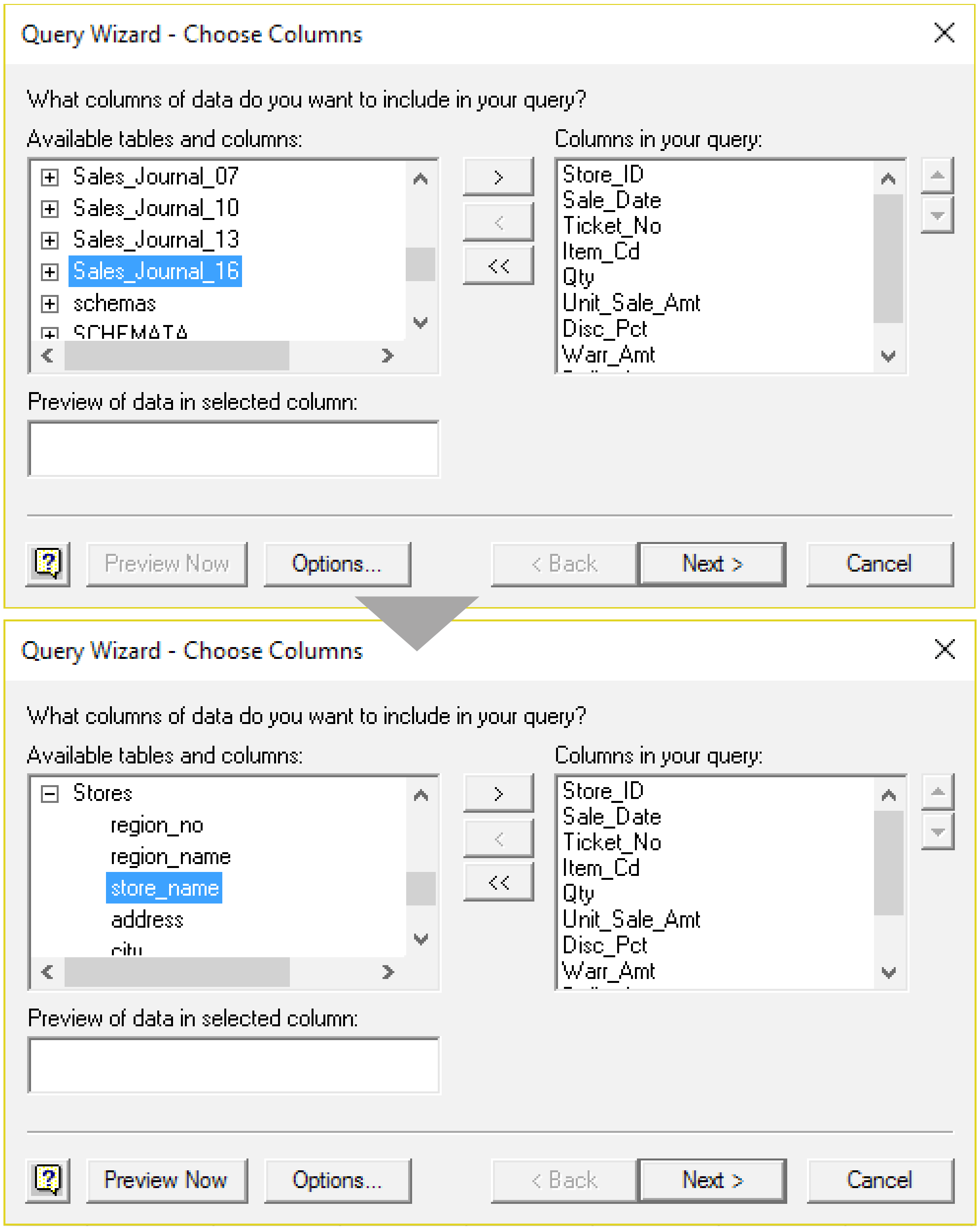
☑ Use the Query Wizard to create/edit queries

Figure 15.3

7. *In the* **Choose Data Source** *dialog box, click on* **Nitey-Nite*** *and click* **OK**.

> ***Note**: If you have been unable to make the connection to the* **SQL Server** *database, click on the* **MS Access Database DSN** *and navigate to* **C:\ExcelCEO\Access 2016 and choose the Static_2016.accdb** *database.*

8. *In the* **SQL Server Login** *dialog box, make sure* **AccessUser** *is the* **Login ID** *and type* ***clinesys1*** *as the* **Password** *and click* **OK**.
9. *In the* **Query Wizard - Choose Columns** *dialog box, scroll down and click on the* **Sales_Journal_16** *table and bring all columns over into the* **Columns in your query** *box.*
10. *Next, click on the* ⊞ *sign to expand the fields in the* **Stores** *table and bring over the* **Store_ID** *and* **Store_No** *fields, and click* **Next >**.



Query Wizard - Choose Columns

What columns of data do you want to include in your query?

Available tables and columns:

- ⊞ Sales_Journal_07
- ⊞ Sales_Journal_10
- ⊞ Sales_Journal_13
- ⊞ Sales_Journal_16
- ⊞ schemas
- ⊞ SCHEMATA

Columns in your query:

Store_ID
Sale_Date
Ticket_No
Item_Cd
Qty
Unit_Sale_Amt
Disc_Pct
Warr_Amt

Preview of data in selected column:

Query Wizard - Choose Columns

What columns of data do you want to include in your query?

Available tables and columns:

- ⊟ Stores
  - region_no
  - region_name
  - store_name
  - address
  - city

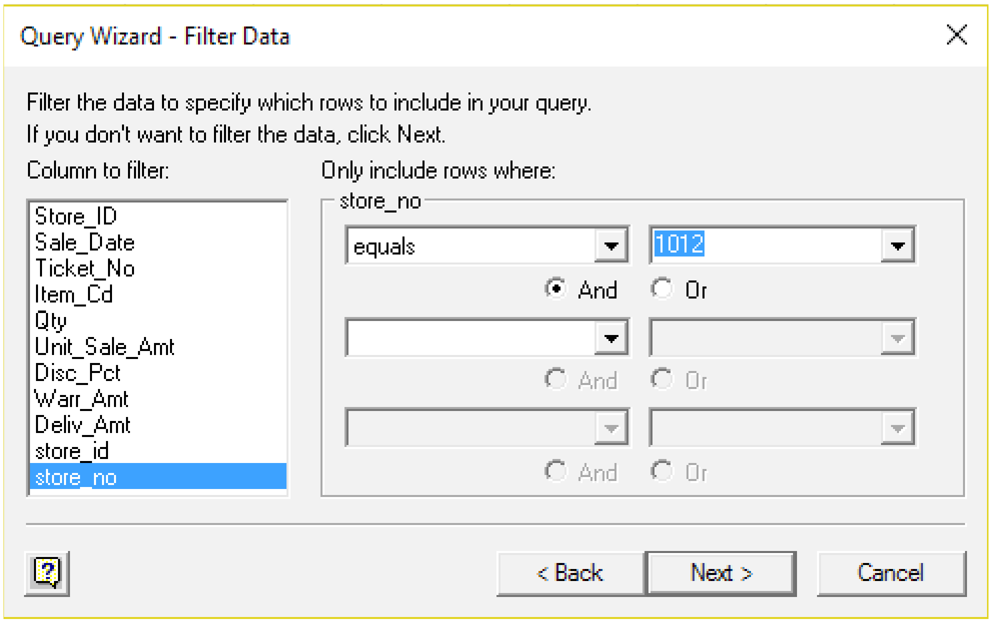
Columns in your query:

Store_ID
Sale_Date
Ticket_No
Item_Cd
Qty
Unit_Sale_Amt
Disc_Pct
Warr_Amt

Preview of data in selected column:

Figure 15.4

11. *In the* **Query Wizard - Filter Data** *dialog box, click on the* **store_no** *field on the left side of the screen to filter by* **Store_No**.
12. *On the right side of the dialog box, set the first filter to* **equals** *and the second filter to* **1012** *(to filter for Store_No 1012).*



Query Wizard - Filter Data

Filter the data to specify which rows to include in your query.
If you don't want to filter the data, click Next.

Column to filter:

- Store_ID
- Sale_Date
- Ticket_No
- Item_Cd
- Qty
- Unit_Sale_Amt
- Disc_Pct
- Warr_Amt
- Deliv_Amt
- store_id
- store_no

Only include rows where:

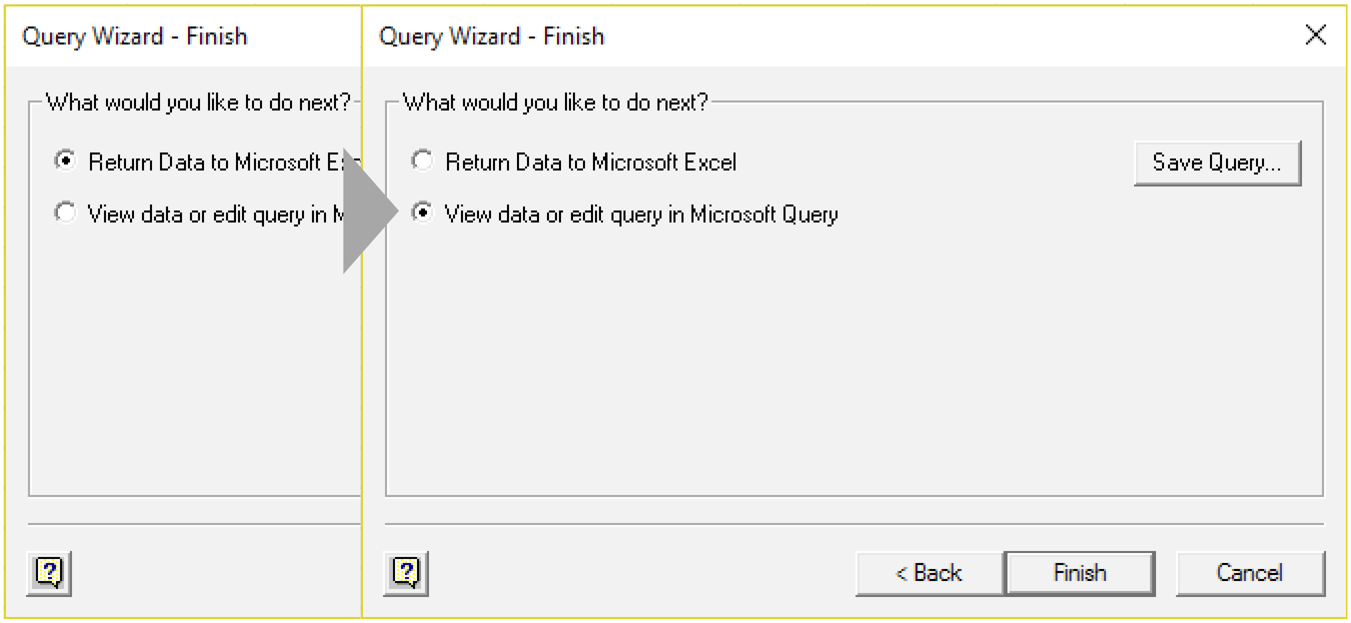
store_no

equals 1012

● And ○ Or

Figure 15.5

13. *Click* **Next >** *two times to get to the* **Query Wizard - Finish** *dialog box.*



Query Wizard - Finish

What would you like to do next?

● Return Data to Microsoft E

○ View data or edit query in M

Query Wizard - Finish

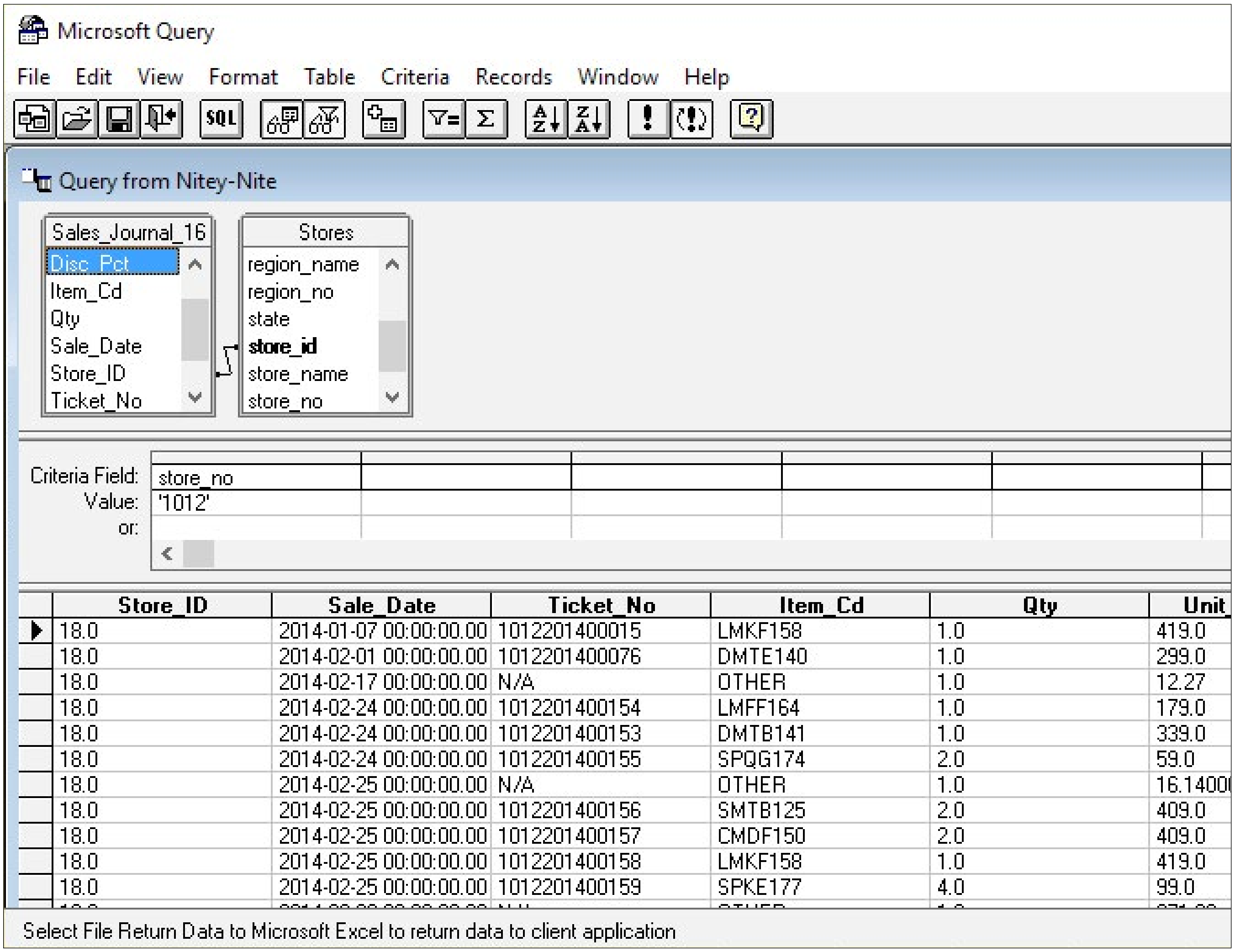
What would you like to do next?

○ Return Data to Microsoft Excel

● View data or edit query in Microsoft Query

Figure 15.6

14. *In the* **Query Wizard- Finish** *dialog box, click on the* **View data or edit query in Microsoft Query** *radio button and click* **Finish**.



Criteria Field: store_no
Value: '1012'
or:

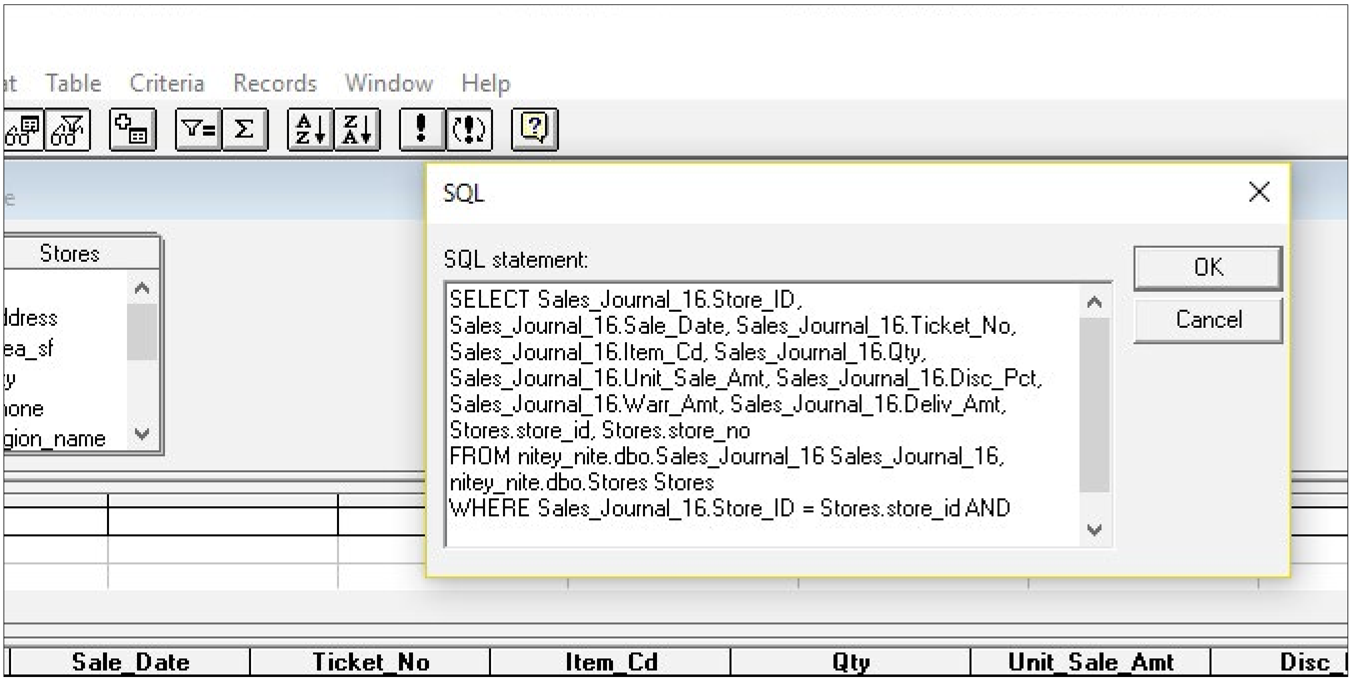
	Store_ID	Sale_Date	Ticket_No	Item_Cd	Qty	Unit
▶	18.0	2014-01-07 00:00:00.00	1012201400015	LMKF158	1.0	419.0
	18.0	2014-02-01 00:00:00.00	1012201400076	DMTE140	1.0	299.0
	18.0	2014-02-17 00:00:00.00	N/A	OTHER	1.0	12.27
	18.0	2014-02-24 00:00:00.00	1012201400154	LMFF164	1.0	179.0
	18.0	2014-02-24 00:00:00.00	1012201400153	DMTB141	1.0	339.0
	18.0	2014-02-24 00:00:00.00	1012201400155	SPQG174	2.0	59.0
	18.0	2014-02-25 00:00:00.00	N/A	OTHER	1.0	16.1400
	18.0	2014-02-25 00:00:00.00	1012201400156	SMTB125	2.0	409.0
	18.0	2014-02-25 00:00:00.00	1012201400157	CMDF150	2.0	409.0
	18.0	2014-02-25 00:00:00.00	1012201400158	LMKF158	1.0	419.0
	18.0	2014-02-25 00:00:00.00	1012201400159	SPKE177	4.0	99.0

Select File Return Data to Microsoft Excel to return data to client application

Figure 15.7

You should get a screen similar to the one in Figure 15.7. This screen is simply returning all of the records in the Sales_Journal table for Store No 1012. Doesn't it look a lot like the GUI screen in an Access query Design View? It should, and it does basically the same thing. To see the SQL code behind this query, click on the SQL icon.

15. *Click on the* **SQL** SQL *icon.*



```
SELECT Sales_Journal_16.Store_ID,
Sales_Journal_16.Sale_Date, Sales_Journal_16.Ticket_No,
Sales_Journal_16.Item_Cd, Sales_Journal_16.Qty,
Sales_Journal_16.Unit_Sale_Amt, Sales_Journal_16.Disc_Pct,
Sales_Journal_16.Warr_Amt, Sales_Journal_16.Deliv_Amt,
Stores.store_id, Stores.store_no
FROM nitey_nite.dbo.Sales_Journal_16 Sales_Journal_16,
nitey_nite.dbo.Stores Stores
WHERE Sales_Journal_16.Store_ID = Stores.store_id AND
```

Figure 15.8

The SQL dialog box appears. In this box, you can write the SQL code you want for your new query. Important point: If you write SQL code somewhere outside of this screen and paste it in, Microsoft Query may not be able to display the code in this view. If it can't display the SQL code in this view, you will not be able to code in parameters. Since we are going to input parameters, we need to design the query in the GUI screen. Using the GUI screen is just like designing the query with Access. I have found that Microsoft Query is quirky sometimes and won't accept some of the more complex code (like temp tables, CASE statements, and UNION queries) if you write it in straight SQL. Let's use the graphic interface to design this part of the query.

> ***Note**: If you are using the* **Static_2016.accdb** *database file, you will most likely go directly to the image in* **Figure 15.9**. *If that happens, you will need to manually create the* **Join** *on* **Store_ID** *in both tables. To show the Criteria section, click on the* **Criteria** *menu item, click* **Add Criteria**, *and in the* **Field** *drop-down menu, choose* **Store_No**, *and set it equal to* **1012**, *and* **Sale_Date BETWEEN 1/1/2016 AND 1/31/2016**.

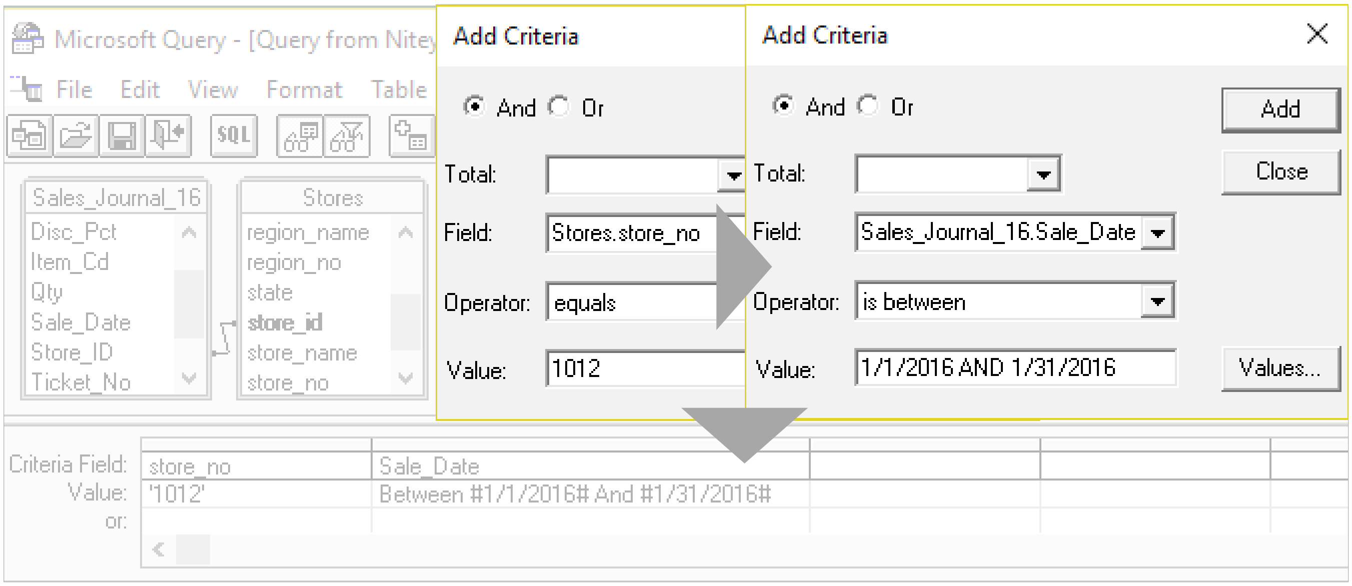
16. *Click* **OK** *on the* **SQL** *dialog box.*

Working with Microsoft Query Design View

In this analysis, we brought fields from two tables: Sales_Journal (where we will get all of the base information), and the Stores table (where we can join to the Sales_Journal table on Store_ID to get the Store number). When you brought in the two tables, Microsoft Query automatically assumed that there would be a join between the two tables on the Store_ID field, so it created the relationship. You can see this as there is a line connecting the Store_ID fields in both tables (If you used the Static2016.accdb database, you will have to manually create that join). If you ever use Microsoft Query and need to create another join, click and drag between the two fields, just like you do in Access.

You've already set the query to pull all of the data for Store No 1012. You'll now put in the variables for sales between 1/1/2016 and 1/31/2016, as well as make a couple of changes in Microsoft Query.

1. *Maximize the interior window of the* **Microsoft Query Design Grid**.
2. **Microsoft Query** *should automatically establish a relationship on* **Store_ID**. *If it didn't, do it now (you may have to move the* **Stores** *table window to see the relationship).*
3. *Take* **Store_ID** *out of the* **Design Grid** *in both places.*
4. *Move the* **Store_No** *to be the first field in the grid.*
5. *Click on the* **Criteria** *tab,* **Add Criteria…** *from the* **Microsoft Query Bar** *and input the criteria as shown in* **Figure 15.9**.



Add Criteria

● And ○ Or

Total:

Field: Stores.store_no

Operator: equals

Value: 1012

Add Criteria

● And ○ Or

Total:

Field: Sales_Journal_16.Sale_Date

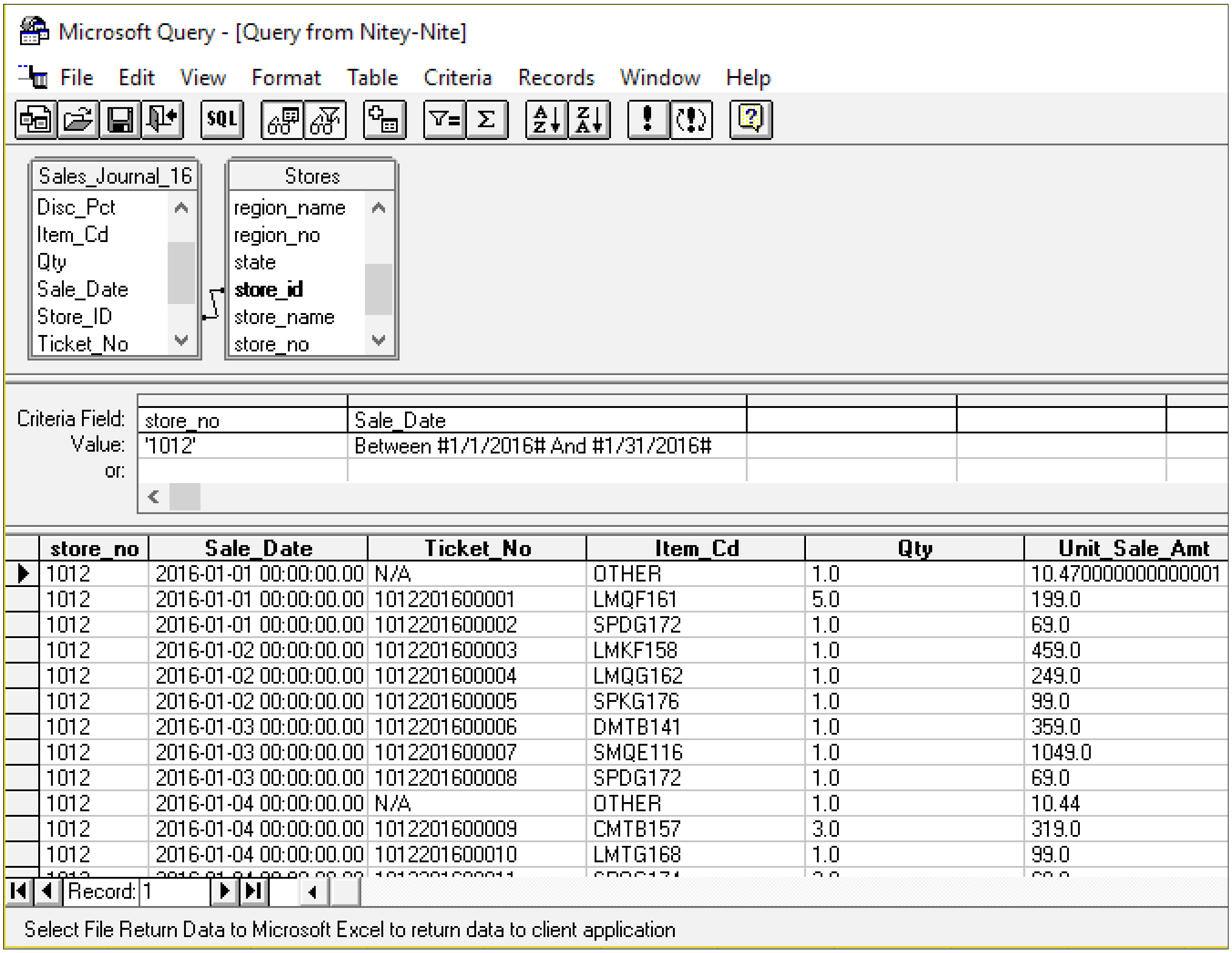
Operator: is between

Value: 1/1/2016 AND 1/31/2016

Criteria Field:	store_no	Sale_Date
Value:	'1012'	Between #1/1/2016# And #1/31/2016#
or:		

Figure 15.9

6. *Click the* **Add** *button then click the* **Close** *button.*



Criteria Field:	store_no	Sale_Date
Value:	'1012'	Between #1/1/2016# And #1/31/2016#
or:		

store_no	Sale_Date	Ticket_No	Item_Cd	Qty	Unit_Sale_Amt
1012	2016-01-01 00:00:00.00	N/A	OTHER	1.0	10.470000000000001
1012	2016-01-01 00:00:00.00	1012201600001	LMQF161	5.0	199.0
1012	2016-01-01 00:00:00.00	1012201600002	SPDG172	1.0	69.0
1012	2016-01-02 00:00:00.00	1012201600003	LMKF158	1.0	459.0
1012	2016-01-02 00:00:00.00	1012201600004	LMQG162	1.0	249.0
1012	2016-01-02 00:00:00.00	1012201600005	SPKG176	1.0	99.0
1012	2016-01-03 00:00:00.00	1012201600006	DMTB141	1.0	359.0
1012	2016-01-03 00:00:00.00	1012201600007	SMQE116	1.0	1049.0
1012	2016-01-03 00:00:00.00	1012201600008	SPDG172	1.0	69.0
1012	2016-01-04 00:00:00.00	N/A	OTHER	1.0	10.44
1012	2016-01-04 00:00:00.00	1012201600009	CMTB157	3.0	319.0
1012	2016-01-04 00:00:00.00	1012201600010	LMTG168	1.0	99.0

Select File Return Data to Microsoft Excel to return data to client application

Figure 15.10

The Microsoft Query Design View screen should look like Figure 15.10.

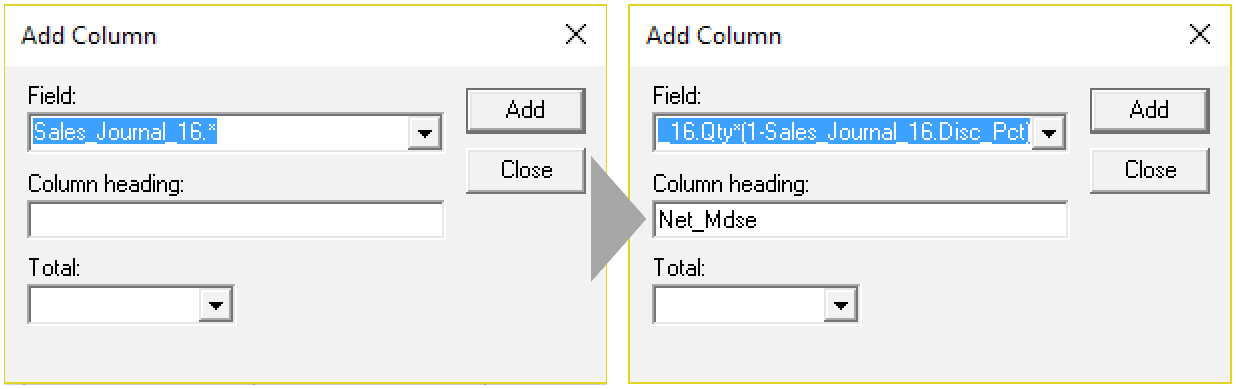
Adding Calculated Fields

At this point, you need to add two calculated fields: Net_Mdse (for Net Merchandise) and Total_Sale. You've calculated these fields in Excel and in Access, so I won't review the concept. I'll just show you how to do it in the Microsoft Query Design View.

1. *Click on* **Records**, **Add Column…**

> ***Note**: If your cursor was somewhere in the record set of the* **Design Grid**, *you may see the* **Insert Column** *options instead of the* **Add Column** *option. Either one will do for this exercise.)*

2. *In the* **Field:** *box, type:* ***Sales_Journal_16.Unit_Sale_Amt*Sales_Journal_16.Qty*(1-Sales_Journal_16.Disc_Pct)***



Field: Sales_Journal_16.*

Field: _16.Qty*(1-Sales_Journal_16.Disc_Pct)

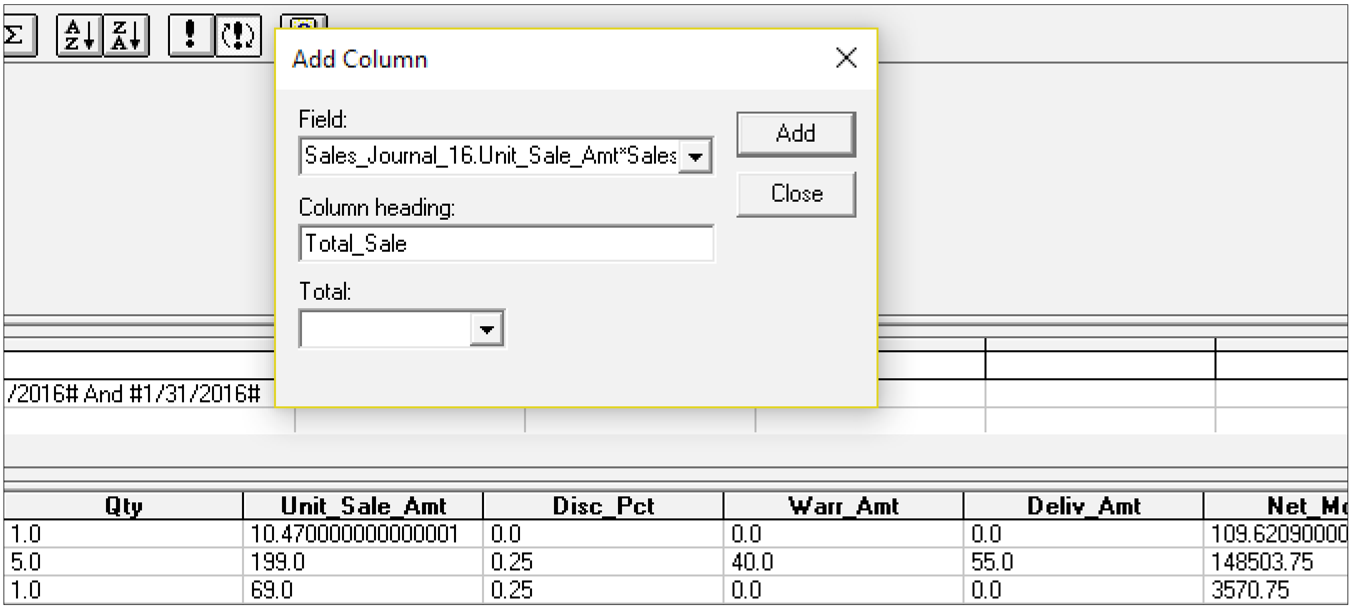
Column heading: Net_Mdse

Figure 15.11

3. *In the* **Column heading:** *box, type:* ***Net_Mdse***
4. *Leave the* **Total:** *box blank and click* **Add**.

Once you click Add, a new field called Net_Mdse appears in the grid below. Let's add the next field which calculates the total sale of the ticket.

5. *In the* **Field:** *box, replace the existing text with:* ***Sales_Journal_16.Unit_Sale_Amt*Sales_Journal_16.Qty*(1-Sales_Journal_16.Disc_Pct)+Deliv_Amt+Warr_Amt***
6. *In the* **Column heading:** *box, type:* ***Total_Sale***
7. *Click* **Add** *and close the* **Add Column** *dialog box.*



Field: Sales_Journal_16.Unit_Sale_Amt*Sales

Column heading: Total_Sale

/2016# And #1/31/2016#

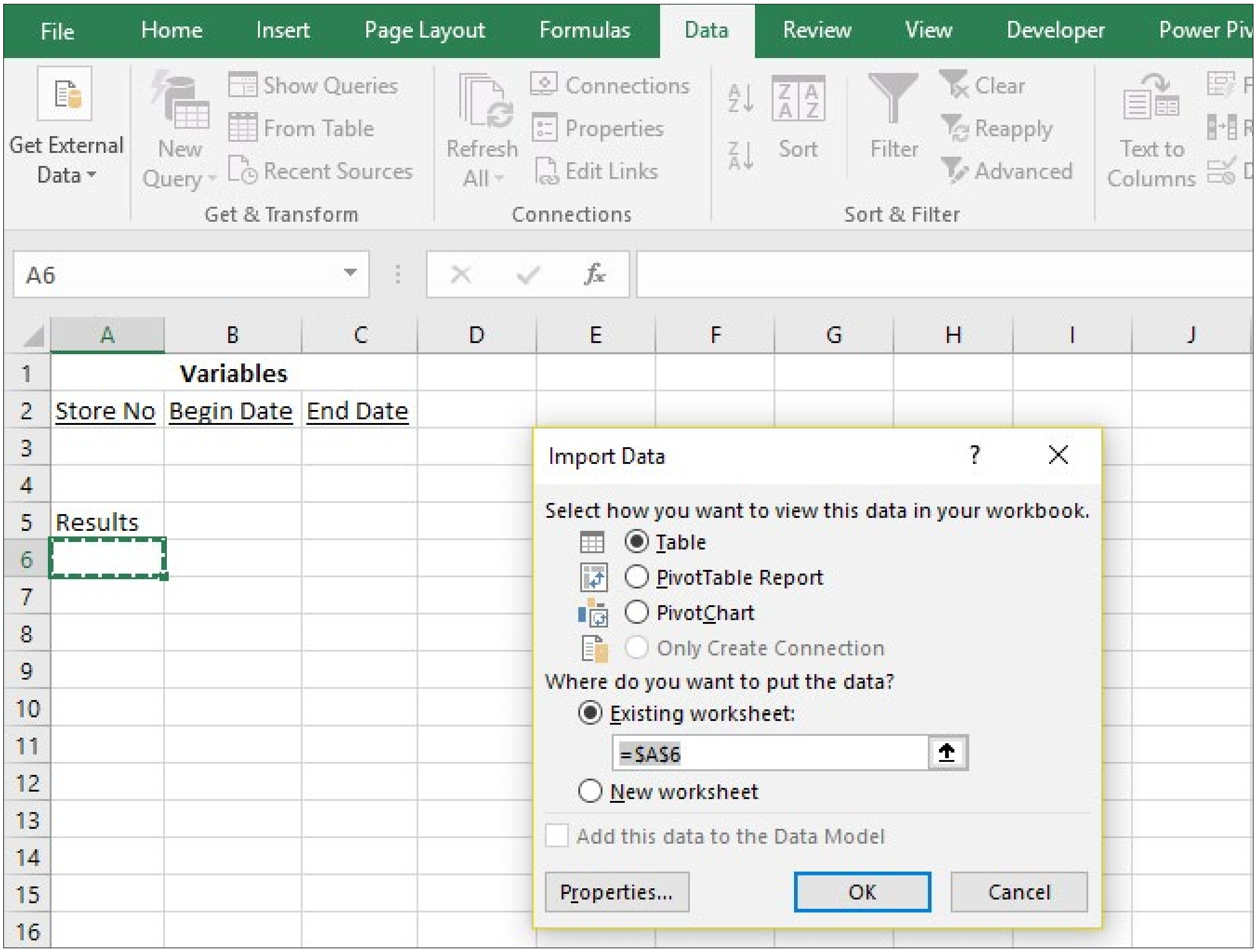
Qty	Unit_Sale_Amt	Disc_Pct	Warr_Amt	Deliv_Amt	Net_Md
1.0	10.470000000000001	0.0	0.0	0.0	109.62090000
5.0	199.0	0.25	40.0	55.0	148503.75
1.0	69.0	0.25	0.0	0.0	3570.75

Figure 15.12

If the Net_Mdse and/or the Total_Sale fields do not populate, click on the Query Now icon !.

Repositioning columns works the same in Access as it does in Microsoft Query.

8. *Click on the* **Net_Mdse** *column and drag it to the right of the* **Disc_Pct** *column.*
9. *Make sure the* **Total_Sale** *column is at the far-right of the grid.*
10. *Close out of* **Microsoft Query** *by clicking the black* ✕ *at the top-right of the Microsoft Query window.*



	A	B	C
1		Variables	
2	Store No	Begin Date	End Date
5	Results		

Import Data ? ✕

Select how you want to view this data in your workbook.

● Table

○ PivotTable Report

○ PivotChart

○ Only Create Connection

Where do you want to put the data?

● Existing worksheet:

=$A$6

○ New worksheet

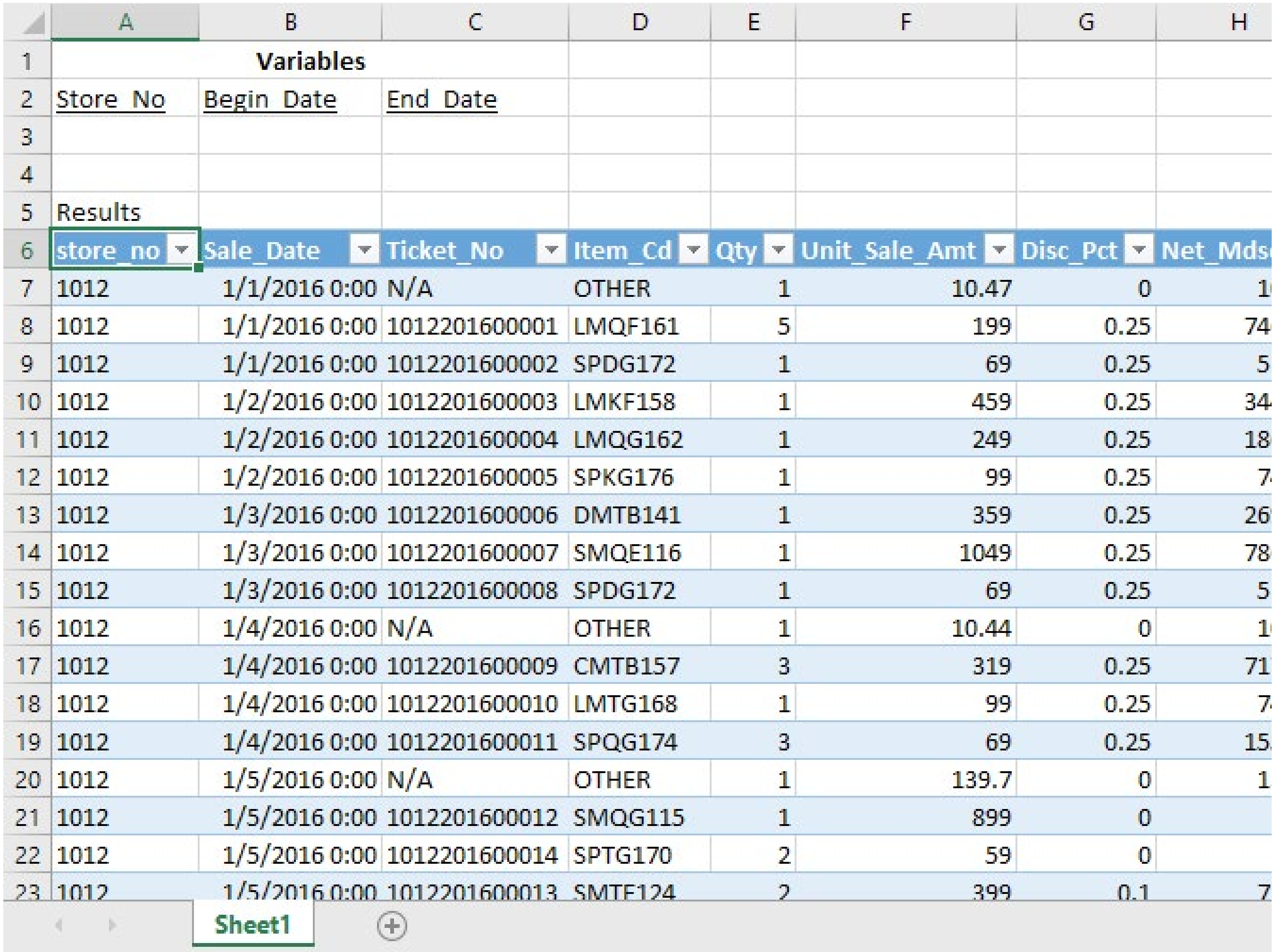
□ Add this data to the Data Model

Properties... OK Cancel

Figure 15.13

The last step of the Query Wizard asks you where to put the results of the query. The Import Data dialog box should read to put the data on Cell A6.

11. *Make sure the* **Import Data** *dialog box reads to put the data on* **Cell A6** *on the* **Existing worksheet:** *radio button and click* **OK**.



	A	B	C	D	E	F	G	H
1		Variables						
2	Store_No	Begin_Date	End_Date					
3								
4								
5	Results							
6	store_no	Sale_Date	Ticket_No	Item_Cd	Qty	Unit_Sale_Amt	Disc_Pct	Net_Mds
7	1012	1/1/2016 0:00	N/A	OTHER	1	10.47	0	1
8	1012	1/1/2016 0:00	1012201600001	LMQF161	5	199	0.25	74
9	1012	1/1/2016 0:00	1012201600002	SPDG172	1	69	0.25	5
10	1012	1/2/2016 0:00	1012201600003	LMKF158	1	459	0.25	34
11	1012	1/2/2016 0:00	1012201600004	LMQG162	1	249	0.25	18
12	1012	1/2/2016 0:00	1012201600005	SPKG176	1	99	0.25	7
13	1012	1/3/2016 0:00	1012201600006	DMTB141	1	359	0.25	26
14	1012	1/3/2016 0:00	1012201600007	SMQE116	1	1049	0.25	78
15	1012	1/3/2016 0:00	1012201600008	SPDG172	1	69	0.25	5
16	1012	1/4/2016 0:00	N/A	OTHER	1	10.44	0	1
17	1012	1/4/2016 0:00	1012201600009	CMTB157	3	319	0.25	71
18	1012	1/4/2016 0:00	1012201600010	LMTG168	1	99	0.25	7
19	1012	1/4/2016 0:00	1012201600011	SPQG174	3	69	0.25	15
20	1012	1/5/2016 0:00	N/A	OTHER	1	139.7	0	1
21	1012	1/5/2016 0:00	1012201600012	SMQG115	1	899	0	
22	1012	1/5/2016 0:00	1012201600014	SPTG170	2	59	0	
23	1012	1/5/2016 0:00	1012201600013	SMTF124	2	399	0.1	7

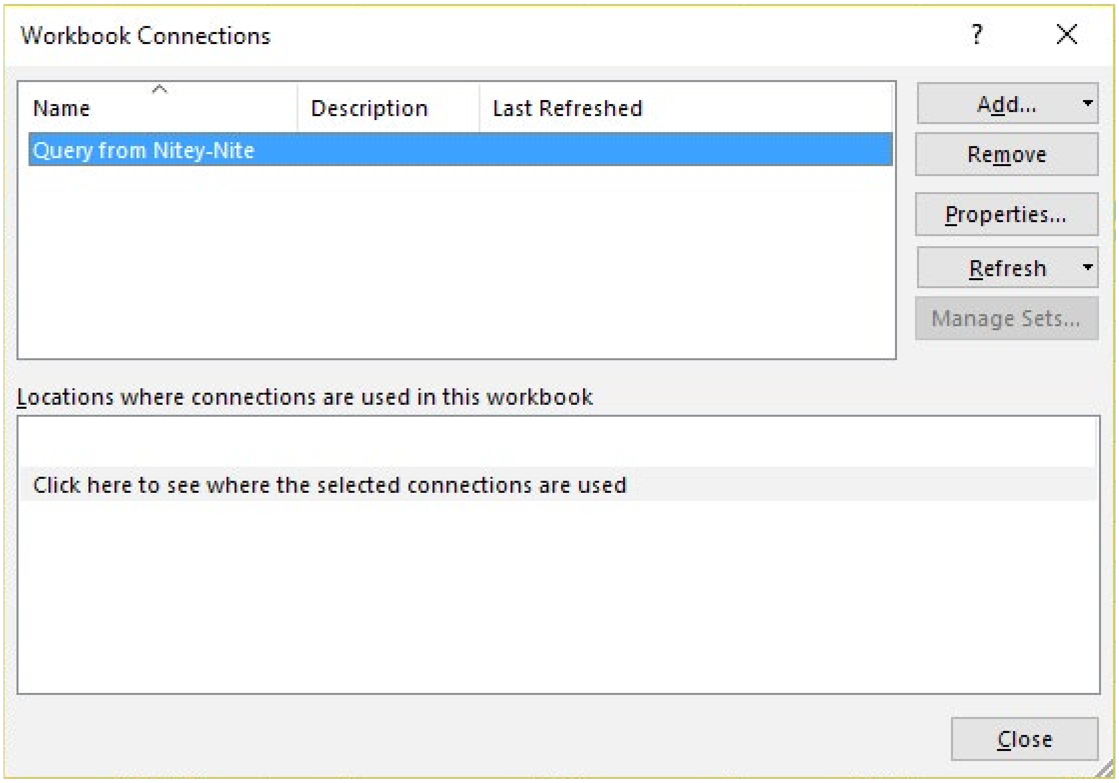
Figure 15.14

And like magic you get a record set containing 92 records imported onto the spreadsheet. When the data is imported, Excel formats it as a table. To refresh the data in the database, you can right-click anywhere in the table and choose Refresh. You can also use the Refresh All icon in the Connections group of the Data tab. When the data is in the process of refreshing, you will see a small turning world icon at the bottom-left corner of your screen. Data sets with a large number of records, or some complex SQL code, take longer to refresh. Make sure you don't make any changes to the worksheet when the data is refreshing.

Passing Parameters

This procedure gets us most of the way to where we want to be. However, it still doesn't allow you to pass the Store Number, Begin Date, and End Date variables back to the SQL code. To add those cells as parameters, you have to set them up in Microsoft Query.

1. *Click on the* **Connections** Connections *icon in the* **Data** *tab. Drag the dialog box border over, if needed.*


Workbook Connections

Name	Description	Last Refreshed
Query from Nitey-Nite		

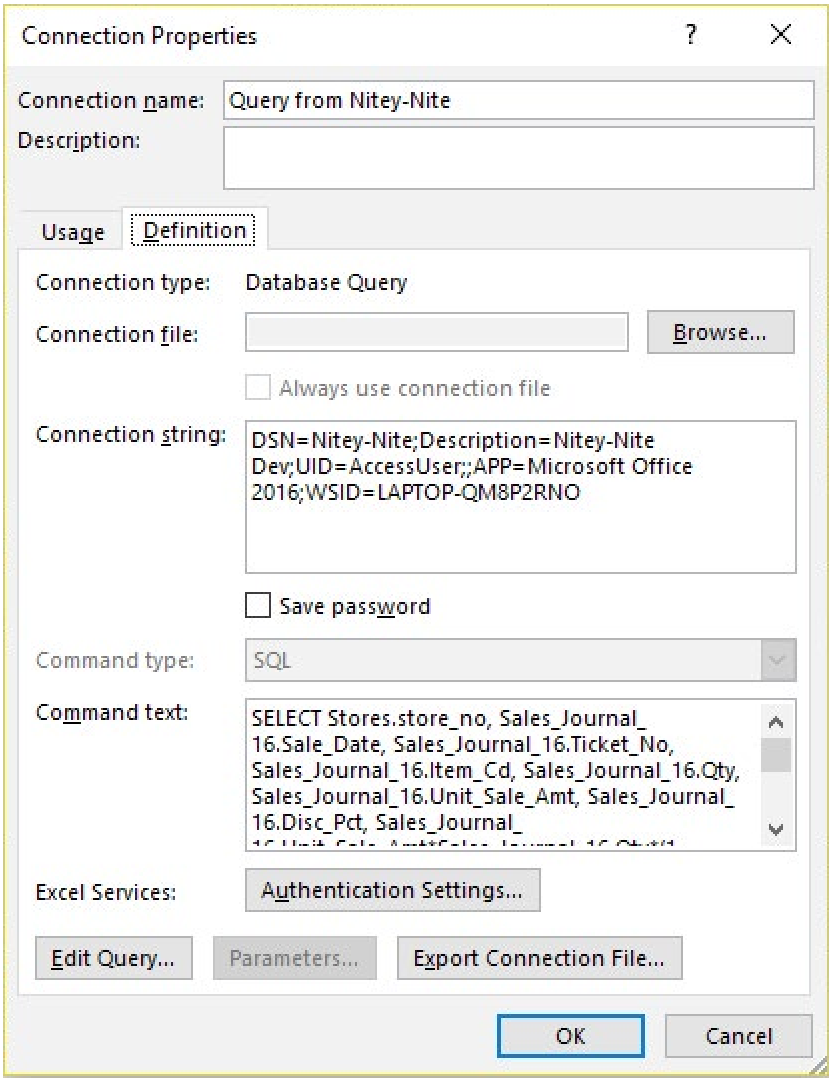
Locations where connections are used in this workbook

Click here to see where the selected connections are used

Figure 15.15

The Workbook Connections dialog box appears. This dialog box lists all of the connections you have created in this workbook. Since you've created only one connection, the Query from Nitey-Nite name appears.

2. *Click on the* **Properties…** *button then click on the* **Definition** *tab.*



Connection Properties

Connection name: Query from Nitey-Nite

Description:

Connection type: Database Query

Connection file:

□ Always use connection file

Connection string: DSN=Nitey-Nite;Description=Nitey-Nite Dev;UID=AccessUser;;APP=Microsoft Office 2016;WSID=LAPTOP-QM8P2RNO

□ Save password

Command type: SQL

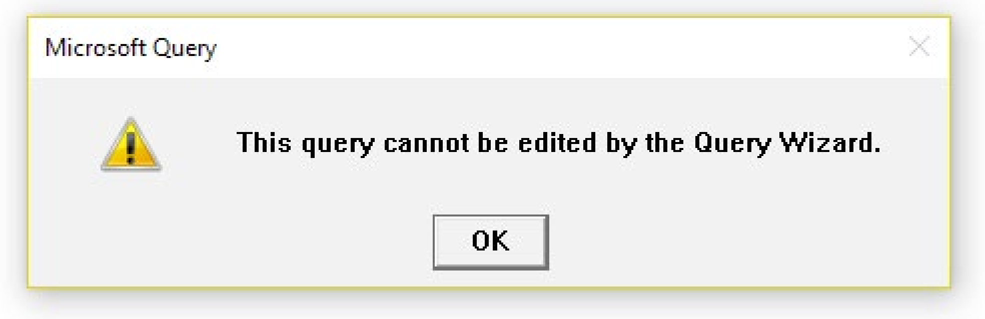
Command text: SELECT Stores.store_no, Sales_Journal_16.Sale_Date, Sales_Journal_16.Ticket_No, Sales_Journal_16.Item_Cd, Sales_Journal_16.Qty, Sales_Journal_16.Unit_Sale_Amt, Sales_Journal_16.Disc_Pct, Sales_Journal_

Excel Services:

Figure 15.16

In the Definition tab of the connection properties dialog box, you can change the connection file, connection string, and the command text. The command text is the SQL code created by the design grid. Notice the Parameters… button is grayed out. It cannot be used until you have some established parameters. To create parameters, you have to edit the query.

3. *Click on the* **Edit Query…** *button.*



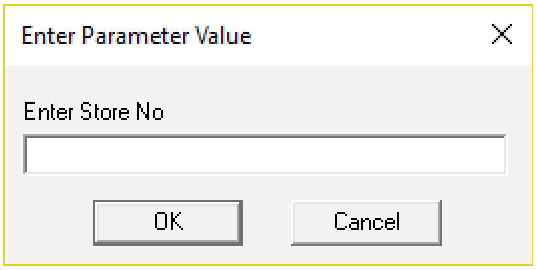
This query cannot be edited by the Query Wizard.

Figure 15.17

4. *Click* **OK**.

You should get a message box from Microsoft Query telling you that the query cannot be edited by the Query Wizard. That's OK since we're not using the wizard. Creating a parameter in Microsoft Query is basically the same as creating a parameter query in Access. Just replace the values in the Criteria line with named parameters surrounded by brackets.

5. *In the* **Store_no** *criteria field, replace* **'1012'** *with* ***[Enter Store No]*** *and press the* **Tab** *key.*



Enter Store No

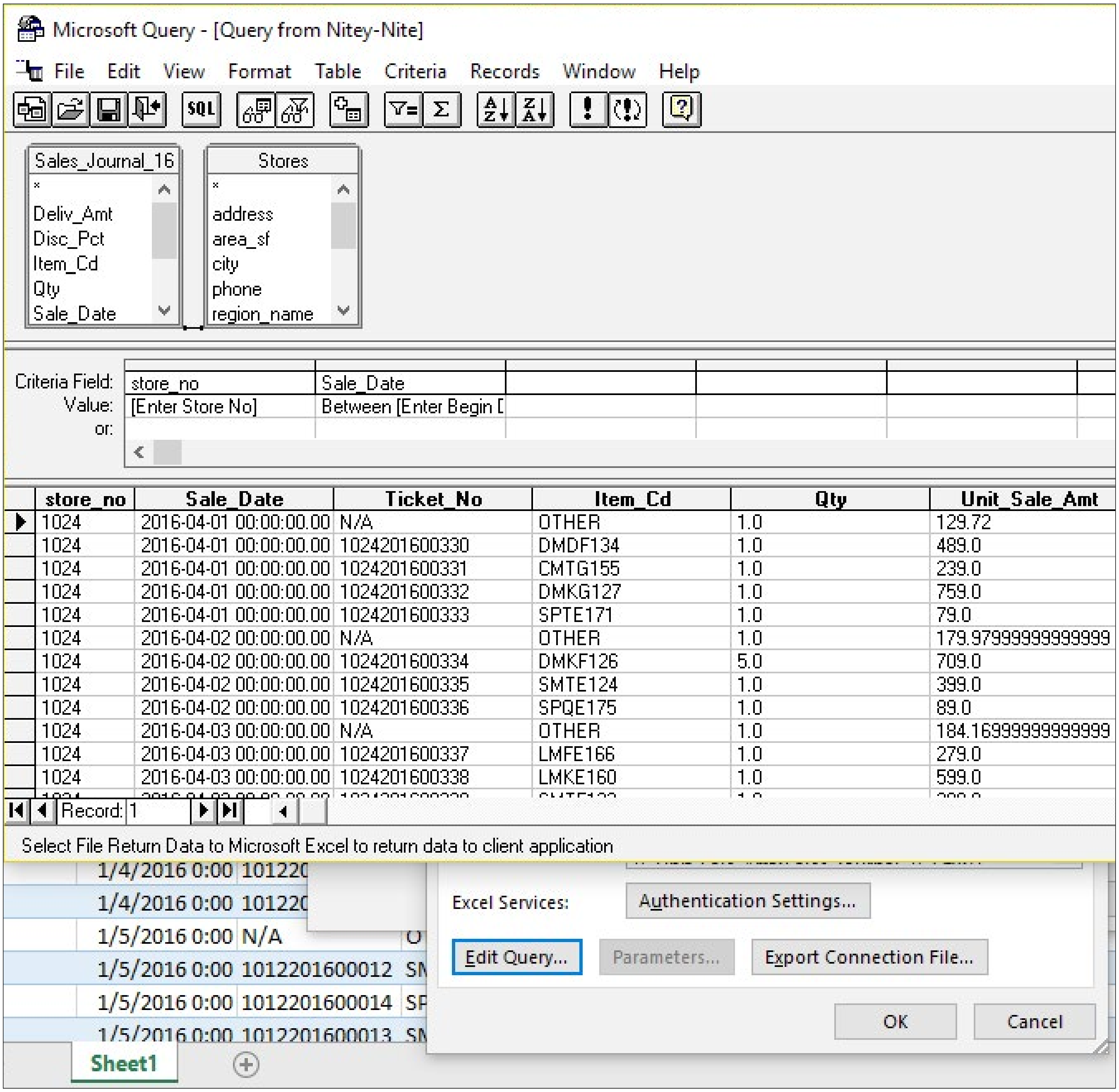
Figure 15.18

Once you press the Tab key, or click outside of the Criteria filed, Microsoft Query will prompt you to input the value for the parameter. If it doesn't, click the Query Now icon.

6. *Input* ***1024*** *for the* **Store No** *parameter and click* **OK**.

Notice that the data in the record set changed to reflect Store No 1024.

7. *In the* **Sale_Date** *criteria field, replace* **Between #1/1/2016# And #1/31/2016#** *with* **Between *[Enter Begin Date]* And *[Enter End Date]***.
8. *Tab out of the* **Sale_Date** *criteria field and input* ***4/1/2016*** *and* ***6/30/2016*** *in the* **[Enter Begin Date]** *and* **[Enter End Date]** *parameters when prompted.*



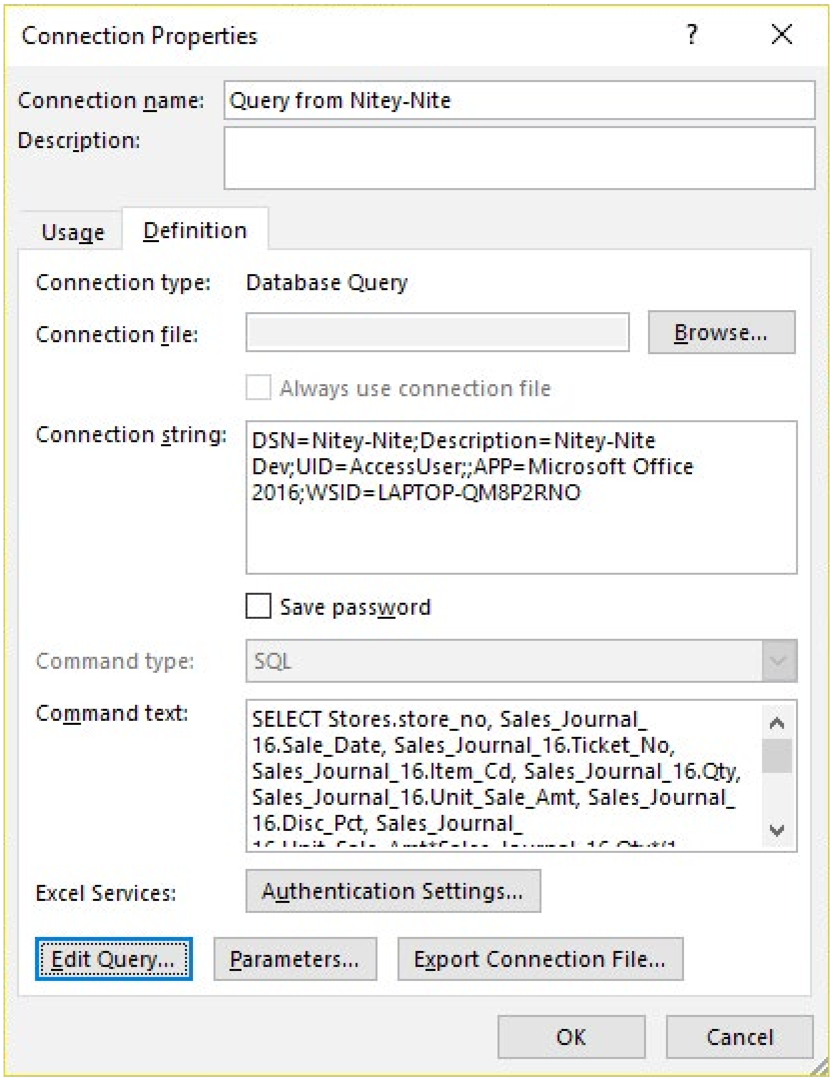
Criteria Field:	store_no	Sale_Date
Value:	[Enter Store No]	Between [Enter Begin [
or:		

store_no	Sale_Date	Ticket_No	Item_Cd	Qty	Unit_Sale_Amt
1024	2016-04-01 00:00:00.00	N/A	OTHER	1.0	129.72
1024	2016-04-01 00:00:00.00	1024201600330	DMDF134	1.0	489.0
1024	2016-04-01 00:00:00.00	1024201600331	CMTG155	1.0	239.0
1024	2016-04-01 00:00:00.00	1024201600332	DMKG127	1.0	759.0
1024	2016-04-01 00:00:00.00	1024201600333	SPTE171	1.0	79.0
1024	2016-04-02 00:00:00.00	N/A	OTHER	1.0	179.97999999999999
1024	2016-04-02 00:00:00.00	1024201600334	DMKF126	5.0	709.0
1024	2016-04-02 00:00:00.00	1024201600335	SMTE124	1.0	399.0
1024	2016-04-02 00:00:00.00	1024201600336	SPQE175	1.0	89.0
1024	2016-04-03 00:00:00.00	N/A	OTHER	1.0	184.16999999999999
1024	2016-04-03 00:00:00.00	1024201600337	LMFE166	1.0	279.0
1024	2016-04-03 00:00:00.00	1024201600338	LMKE160	1.0	599.0

Select File Return Data to Microsoft Excel to return data to client application

Figure 15.19

9. *Exit* **Microsoft Query***, then click* **OK** *in the* **Connection Properties** *to save changes.*



Connection Properties

Connection name: Query from Nitey-Nite

Description:

Connection type: Database Query

Connection file:

□ Always use connection file

Connection string: DSN=Nitey-Nite;Description=Nitey-Nite Dev;UID=AccessUser;;APP=Microsoft Office 2016;WSID=LAPTOP-QM8P2RNO

□ Save password

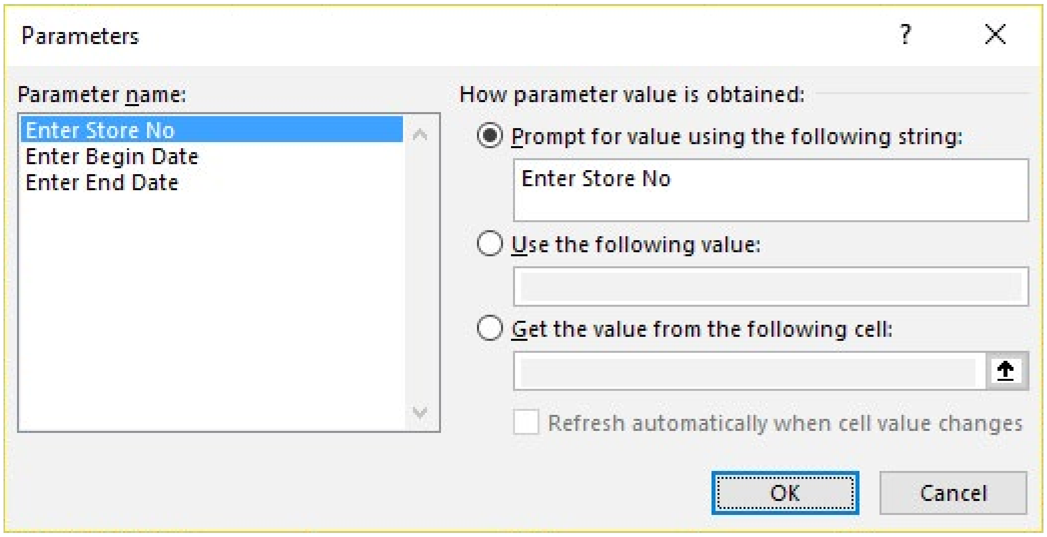
Command type: SQL

Command text: SELECT Stores.store_no, Sales_Journal_16.Sale_Date, Sales_Journal_16.Ticket_No, Sales_Journal_16.Item_Cd, Sales_Journal_16.Qty, Sales_Journal_16.Unit_Sale_Amt, Sales_Journal_16.Disc_Pct, Sales_Journal_

Figure 15.20

You now return to the Connection Properties dialog box. But now you can see that the Parameters… button is enabled, so you can now define the cells as part of the criteria. You can now type the criteria onto the spreadsheet.

10. *Close the* **Connection Properties** *dialog box and the* **Workbook Connections** *dialog box.*
11. *In* **Cell A3**, *type* ***1051***.
12. *In* **Cell B3**, *type* ***1/1/2015***.
13. *In* **Cell C3**, *type* ***1/31/2015***.
14. *Click on the* **Data** *tab,* **Connections** *icon, then on the* **Properties…** *button and the* **Definition** *tab, and lastly on the* **Parameters…** *button.*



Parameters

Parameter name:

Enter Store No
Enter Begin Date
Enter End Date

How parameter value is obtained:

● Prompt for value using the following string:

Enter Store No

○ Use the following value:

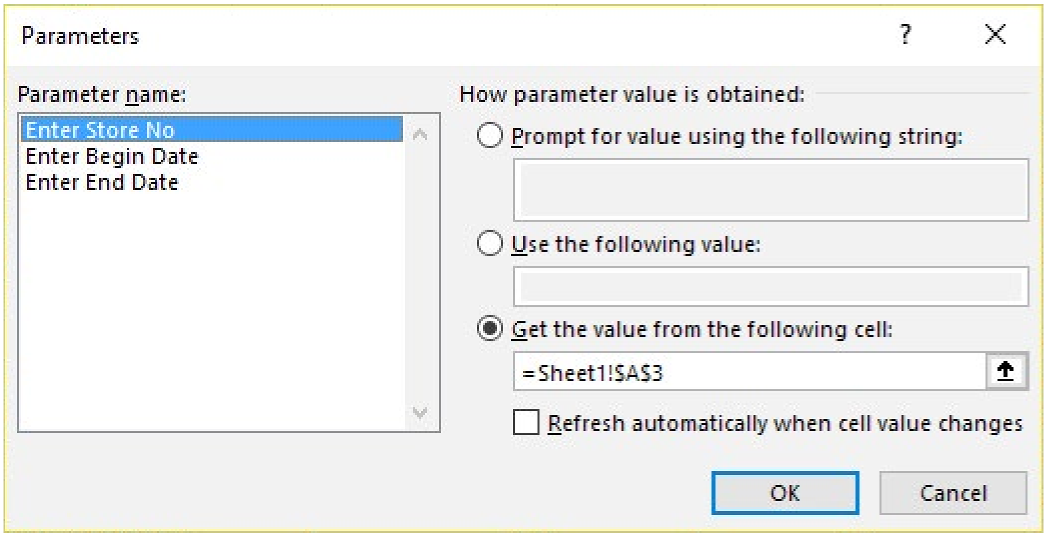
○ Get the value from the following cell:

□ Refresh automatically when cell value changes

Figure 15.21

In the Parameters dialog box, you can set up parameters in one of three ways. The first way is to Prompt for value using the following string:. The string is the value you entered between the brackets, just like a parameter query in Access. The second method is to Use the following value:. This one really isn't a parameter – it's just a hard-coded value. The third method is to Get the value from the following cell:. That's the one you want. You can either type in the cell reference you want or you can click inside the box then click on the cell.

15. *Click on the* **Get value from the following cell** *radio button.*
16. *Click inside the* **Get the value from the following cell** *box, then click on* **Cell A3**.



Parameters

Parameter name:

Enter Store No
Enter Begin Date
Enter End Date

How parameter value is obtained:

○ Prompt for value using the following string:

○ Use the following value:

● Get the value from the following cell:

=Sheet1!$A$3

□ Refresh automatically when cell value changes

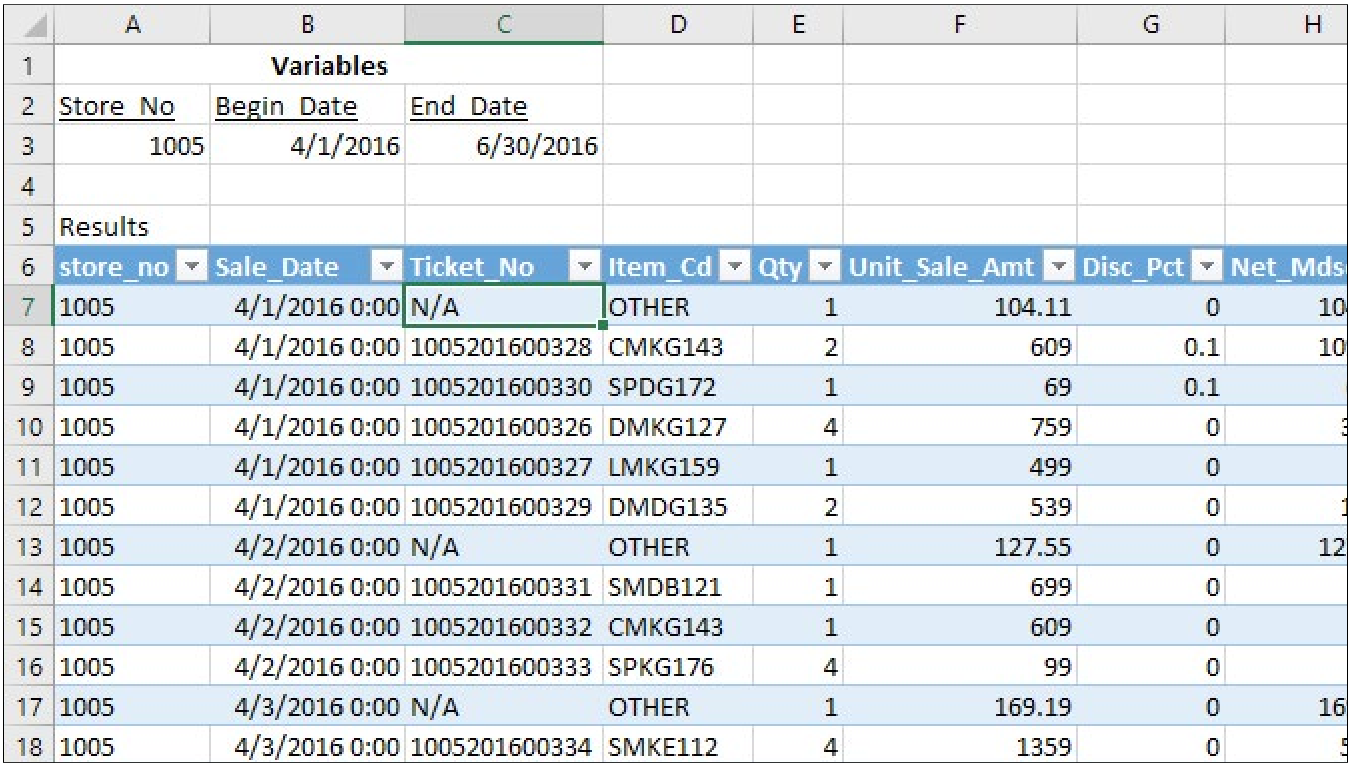
Figure 15.22

Notice that there is a check box that indicates you can refresh the data automatically when the cell value changes. I typically don't use this as my queries usually take a few seconds to run and a user may get confused when he tries to change cells quickly and it appears something is running in the background. If the query is very responsive and works quickly, this is something you may want to consider using, but you won't do it in this exercise.

17. *Select the* [**Enter Begin Date**] *parameter, click on the* **Get the value from the following cell** *radio button and choose* **Cell B3**.
18. *Select the* [**Enter End Date**] *parameter, click on the* **Get the value from the following cell** *radio button and choose* **Cell C3**.
19. *Click* **OK**, *then click* **OK** *or* **Close** *on any open dialog box.*

You now return to the spreadsheet and the record set reflects the updated data. You will be prompted at various times to input the user name (AccessUser) and password (clinesys1) to make a connection to the database. Let's play around with it a bit.

20. *On the spreadsheet, change the* **Store No** *to* ***1005***, *the* **Begin Date** *to* ***4/1/2016*** *and the* **End Date** *to* ***6/30/2016***.
21. *Right-click anywhere in the table and choose* **Refresh** Refresh.



	A	B	C	D	E	F	G	H
1		Variables						
2	Store_No	Begin_Date	End_Date					
3	1005	4/1/2016	6/30/2016					
4								
5	Results							
6	store_no	Sale_Date	Ticket_No	Item_Cd	Qty	Unit_Sale_Amt	Disc_Pct	Net_Mds
7	1005	4/1/2016 0:00	N/A	OTHER	1	104.11	0	10
8	1005	4/1/2016 0:00	1005201600328	CMKG143	2	609	0.1	10
9	1005	4/1/2016 0:00	1005201600330	SPDG172	1	69	0.1	
10	1005	4/1/2016 0:00	1005201600326	DMKG127	4	759	0	3
11	1005	4/1/2016 0:00	1005201600327	LMKG159	1	499	0	
12	1005	4/1/2016 0:00	1005201600329	DMDG135	2	539	0	1
13	1005	4/2/2016 0:00	N/A	OTHER	1	127.55	0	12
14	1005	4/2/2016 0:00	1005201600331	SMDB121	1	699	0	
15	1005	4/2/2016 0:00	1005201600332	CMKG143	1	609	0	
16	1005	4/2/2016 0:00	1005201600333	SPKG176	4	99	0	
17	1005	4/3/2016 0:00	N/A	OTHER	1	169.19	0	16
18	1005	4/3/2016 0:00	1005201600334	SMKE112	4	1359	0	5

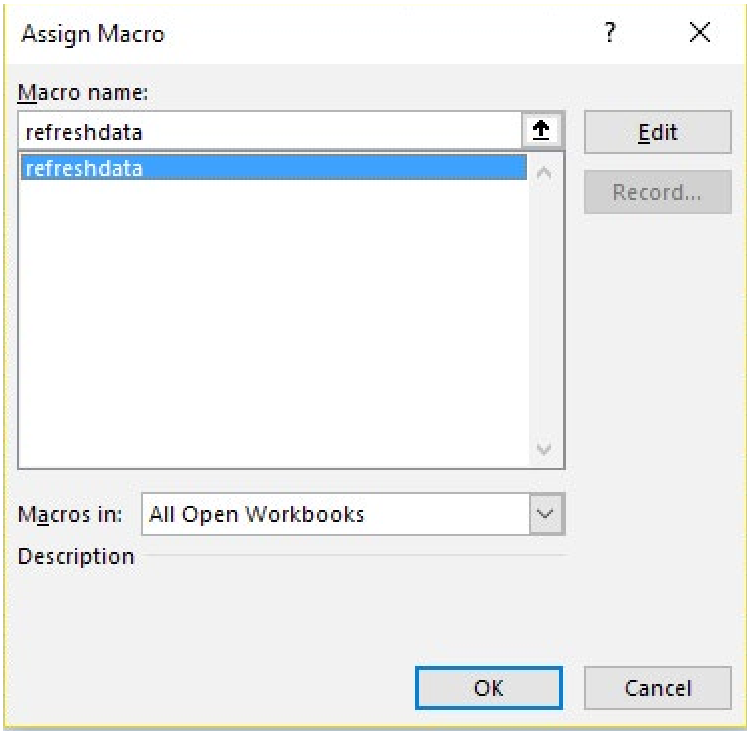
Figure 15.23

Let's do one more thing that will make this tool really powerful. Some Excel users may not know to right-click and refresh the spreadsheet to refresh the data. So let's make it really simple – let's create a command button for them that executes a macro to refresh the data. I'll walk you through how to do that. You'll first record the macro then you'll tie that macro to a command Button to it.

22. *Make sure you have the* **Developer** *tab displayed (If not, click on* **File***, choose* **Options***, click on* **Customize Ribbon***, and make sure the* **Developer** *box on the right side of the dialog box is checked), and click* **OK***.*
23. *Click on the* **View** *tab, then click on the drop-down arrow under* **Macros** *and choose* **Record Macro***.*
24. *Name the macro* ***refreshdata*** *and click* **OK** *(The macro is now recording).*
25. *Click on* **Cell A6***, right-click, and choose* **Refresh***.*
26. *After the data refreshes, click on the* **Stop Recording** *button* ■ *at the lower-left corner of your screen.*

The macro to refresh the data is now recorded. Now all you have to do is tie it to a Command button.

27. *Click on the* **Developer** *tab.*
28. *Click on the* **Insert** *button in the* **Form Controls** *group and click on the first image (***Button (Form Control)** XXXX *).*
29. *Move your cursor down to* **Cell D3** *and click, hold, and draw a small rectangle.*
30. *When you release the mouse, the* **Assign Macro** *dialog box appears. Click on the* **refreshdata** *macro and click* **OK***.*


Assign Macro

Macro name: refreshdata

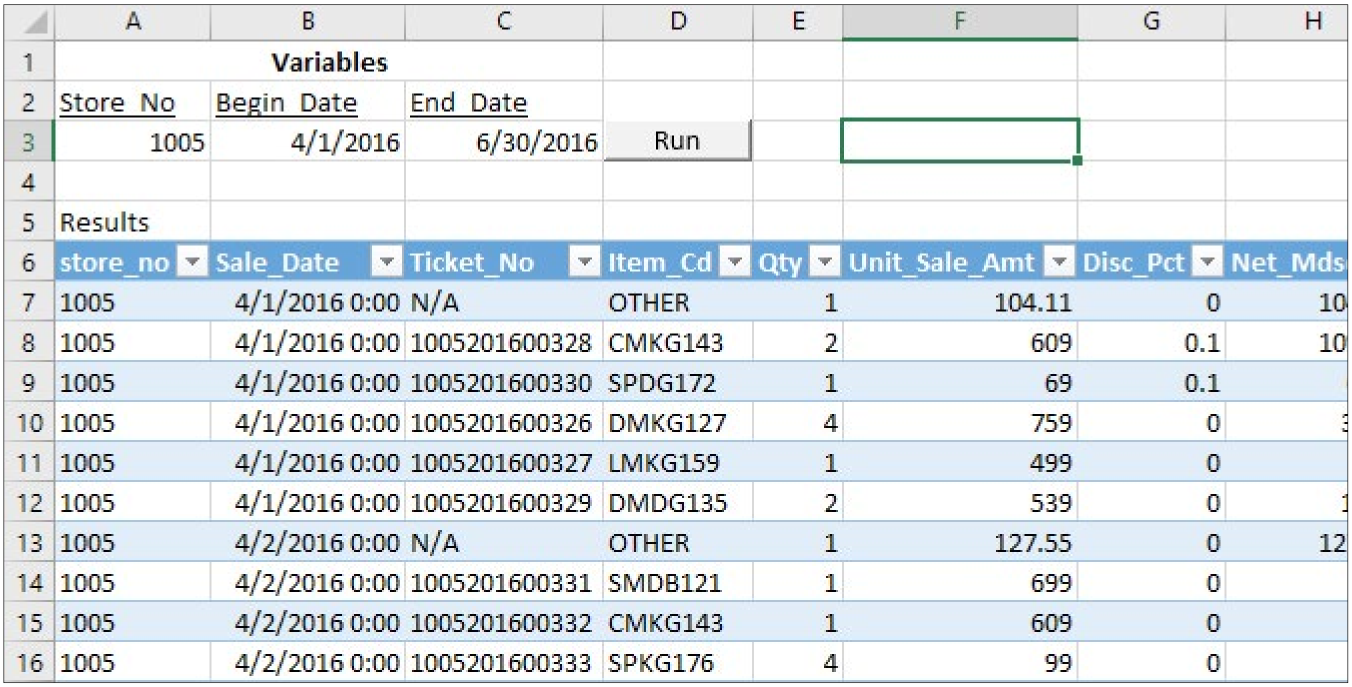
refreshdata

Macros in: All Open Workbooks

Description

Figure 15.24

31. *While the button is in* **Edit** *mode, replace* **Button 1** *with* ***Run****, and click outside of the Command* **Button***.*



	A	B	C	D	E	F	G	H
1		Variables						
2	Store No	Begin Date	End Date					
3	1005	4/1/2016	6/30/2016	Run				
4								
5	Results							
6	store_no	Sale_Date	Ticket_No	Item_Cd	Qty	Unit_Sale_Amt	Disc_Pct	Net_Mds
7	1005	4/1/2016 0:00	N/A	OTHER	1	104.11	0	10
8	1005	4/1/2016 0:00	1005201600328	CMKG143	2	609	0.1	10
9	1005	4/1/2016 0:00	1005201600330	SPDG172	1	69	0.1	
10	1005	4/1/2016 0:00	1005201600326	DMKG127	4	759	0	3
11	1005	4/1/2016 0:00	1005201600327	LMKG159	1	499	0	
12	1005	4/1/2016 0:00	1005201600329	DMDG135	2	539	0	1
13	1005	4/2/2016 0:00	N/A	OTHER	1	127.55	0	12
14	1005	4/2/2016 0:00	1005201600331	SMDB121	1	699	0	
15	1005	4/2/2016 0:00	1005201600332	CMKG143	1	609	0	
16	1005	4/2/2016 0:00	1005201600333	SPKG176	4	99	0	

Figure 15.25

You now have a fully functional data query tool in Excel! The data behind this record set is stored on a server somewhere in California, and it blows through it like it's on your work station. You can change the store number, begin date, and end date, click the Run button and see immediate results. I'm sure you can think of many applications for this tool. Make sure that it works before you take the chapter exam, as there are a few questions that ask you to run the query for certain criteria and give the right answer. If you have it set up the right way, you should have no problem getting the right answers.

32. **Save** *and close* **Excel_Query.xlsx** *(Remember, since the file contains a macro, you have to save it as a macro-enabled file).*

Let me just add one last comment. In this project, I asked you to create a Microsoft Query query from the Query Design Grid because you can't pass parameters to SQL code in Microsoft Query unless it can be displayed in the Design Grid. Many times, you will not need to pass parameters into your SQL Code. Sometimes you may just write the SQL code and copy it into the SQL area of Microsoft Query. That's OK. I have a very complex piece of SQL code that creates a trial balance for a very large corporation by pulling millions of records into a PivotTable. You can't have parameters based on a PivotTable and I couldn't design that code in the Design Grid (it had several UNION queries which can't be displayed in the grid). Even though it can't be displayed in the grid, knowing how to write the complex SQL code to pull a trial balance file was very handy, and it proved to be a very useful PivotTable.

Review Questions*: It is now time to complete the hands-on Review Questions. Log on to www.ExcelCEO.com with your Email and Password, click on the* **Access 2016 and SQL Review Questions, Chapter 15, Section 1 of 2** *option and complete the review questions.*

The Access Master Project

Just one more project, then you will have arrived! Once you complete this last project, you will have a skill set few people in this world have. This project is where you will bring Access, Excel, and Microsoft Query all together. There won't be anything in this project that you can't do, but it will require that you think about what you're doing. If you've taken the Excel course, you can compare this project to the Comprehensive Project. There is a significant difference though. You are not on your own near as much in this project as you were in the Excel Comprehensive Project. I did that because this project that you are about to complete is so intense that very few people could do it totally on their own. Although I give a lot of guidance in this project, you will still have to study the concepts to make sure you can do things like this on your own.

In this last project, you will use a lot of the skills you learned in Excel, Access, and Microsoft Query, all in one project. You will build a report in Excel called the Mattress Mix Report. The purpose of the report is to analyze how many (quantity, not dollars) mattresses sold. The report analyzes the mix of the mattresses sold, broken down by the size and quality of the mattresses as rows, the various vendors are listed in columns, and you can run the report by Region, State, City, or Store for any year and range of months. The report passes parameters via Microsoft Query to an Access database, runs the SQL code against the Access database, then dumps the data onto a tab in Excel called Data. The Report tab is a report in Excel written against the data in the Data tab. Doesn't that sound like an interesting project? It is, and I'm sure that if you learn the concepts taught in the project, you'll be able to use them time and time again in real world projects. Let's get started.

1. **Create** *a new* **Access** *database on* **C:\ExcelCEO\Access 2016** *called* ***Mattress_Mix.accdb****.*
2. *In the database, import the* **Item**, **Sales_Journal**, *and* **Stores** *tables from the* **Nitey_ Nite_2016** *Access database.*
3. *Change the* **Navigation Pane** *to view the* **Access objects** *by* **Object type** *(if necessary).*

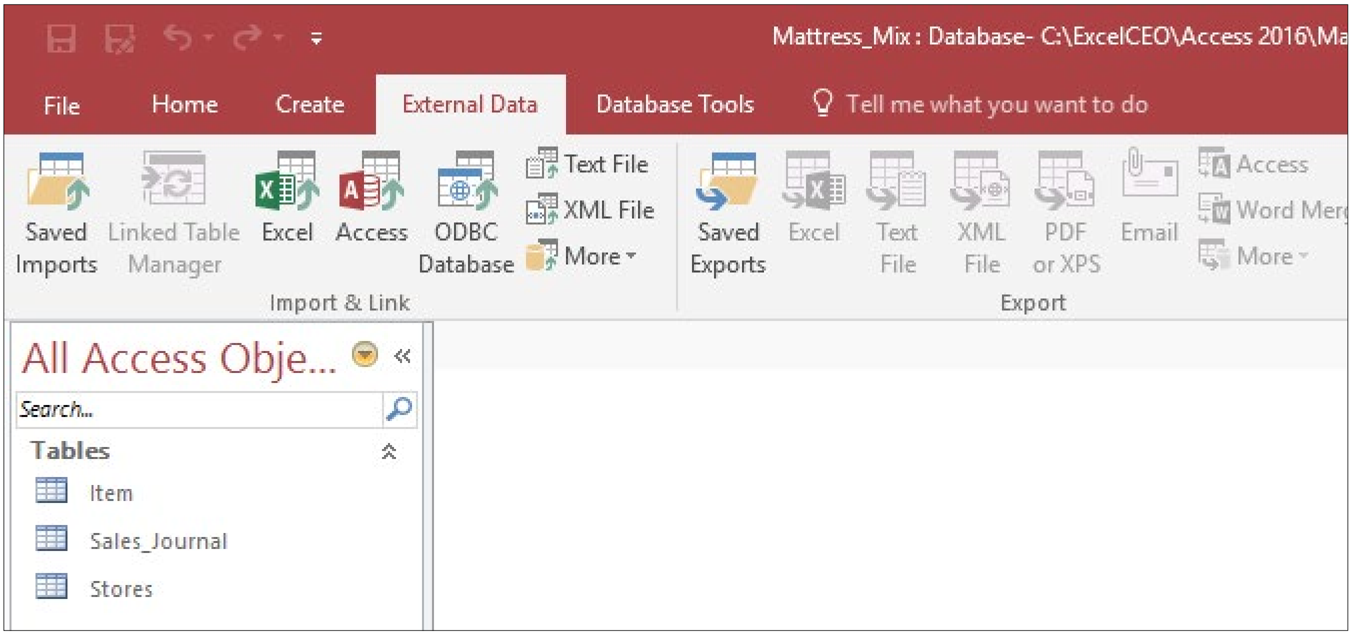
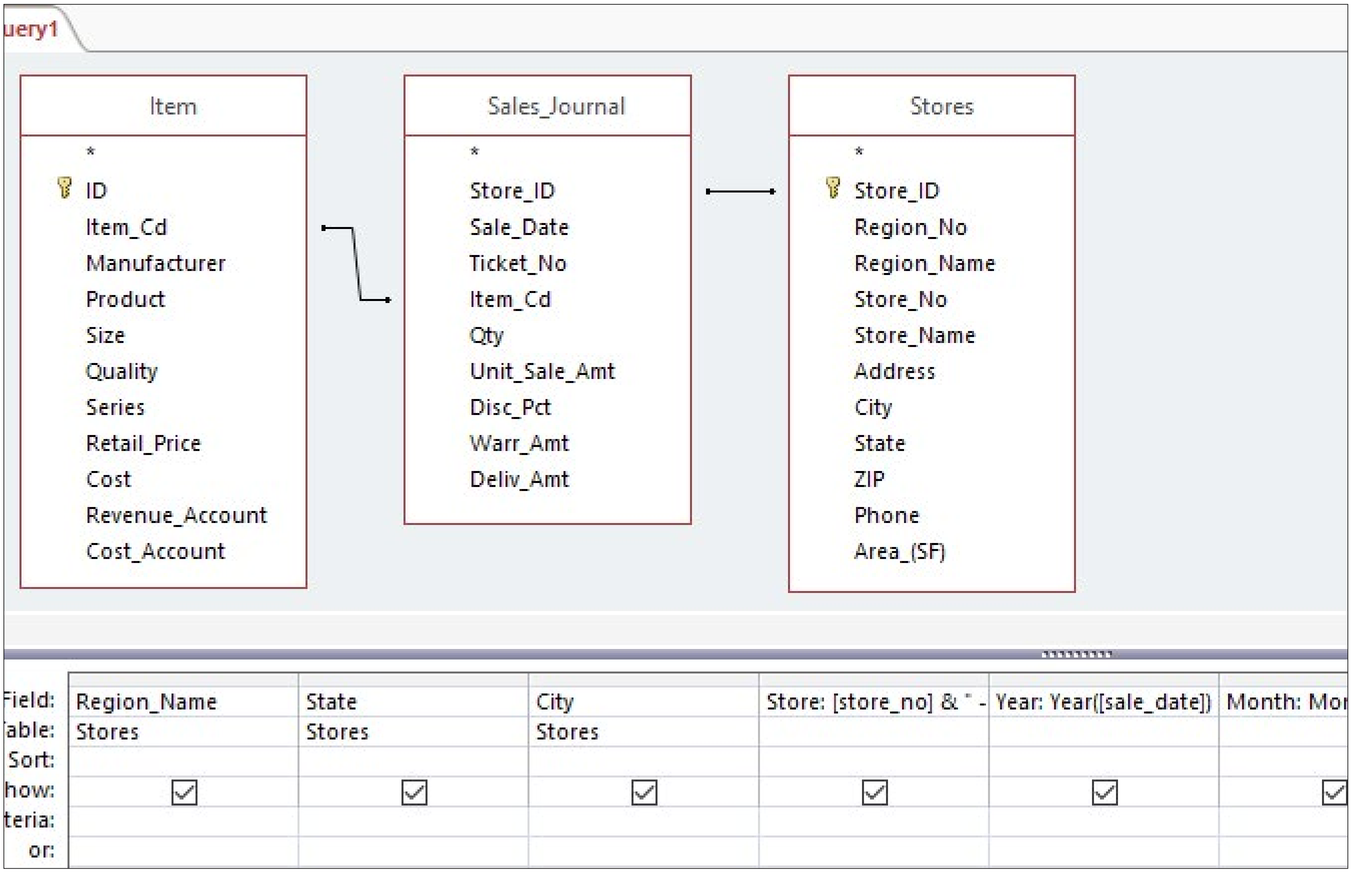


Figure 15.26

The Data tab in the Mattress_Mix.xlsx workbook should contain a "data dump" of all the mattress sales in the Sales_Journal table. Since we can work with Access and Excel in the same project, let's write the query in Access and pull the data from the query into Excel. First we need to write the Access query.

4. *In a new* **Access** *query, bring in the* **Item**, **Sales_Journal**, *and* **Stores** *tables.*
5. *Join the* **Item** *and* **Sales_Journal** *tables on* **Item_Cd** *and make sure the* **Sales_Journal** *and* **Stores** *tables are still joined on the* **Store_ID**.
6. *Bring the* **Region_Name**, **State**, *and* **City** *fields into the query.*
7. **Create** *a field called* **Store** *which concatenates the* **Store_No** *and* **Store_Name**, *separated by a* **space**, *a* **dash**, *and a* **space**, *from the* **Stores** *table.*
8. **Create** *a field called* ***Year*** *which is the* **YEAR()** *function based on the* **Sale_Date**.
9. **Create** *a field called* ***Month*** *which is the* **MONTH()** *function based on the* **Sale_Date**.

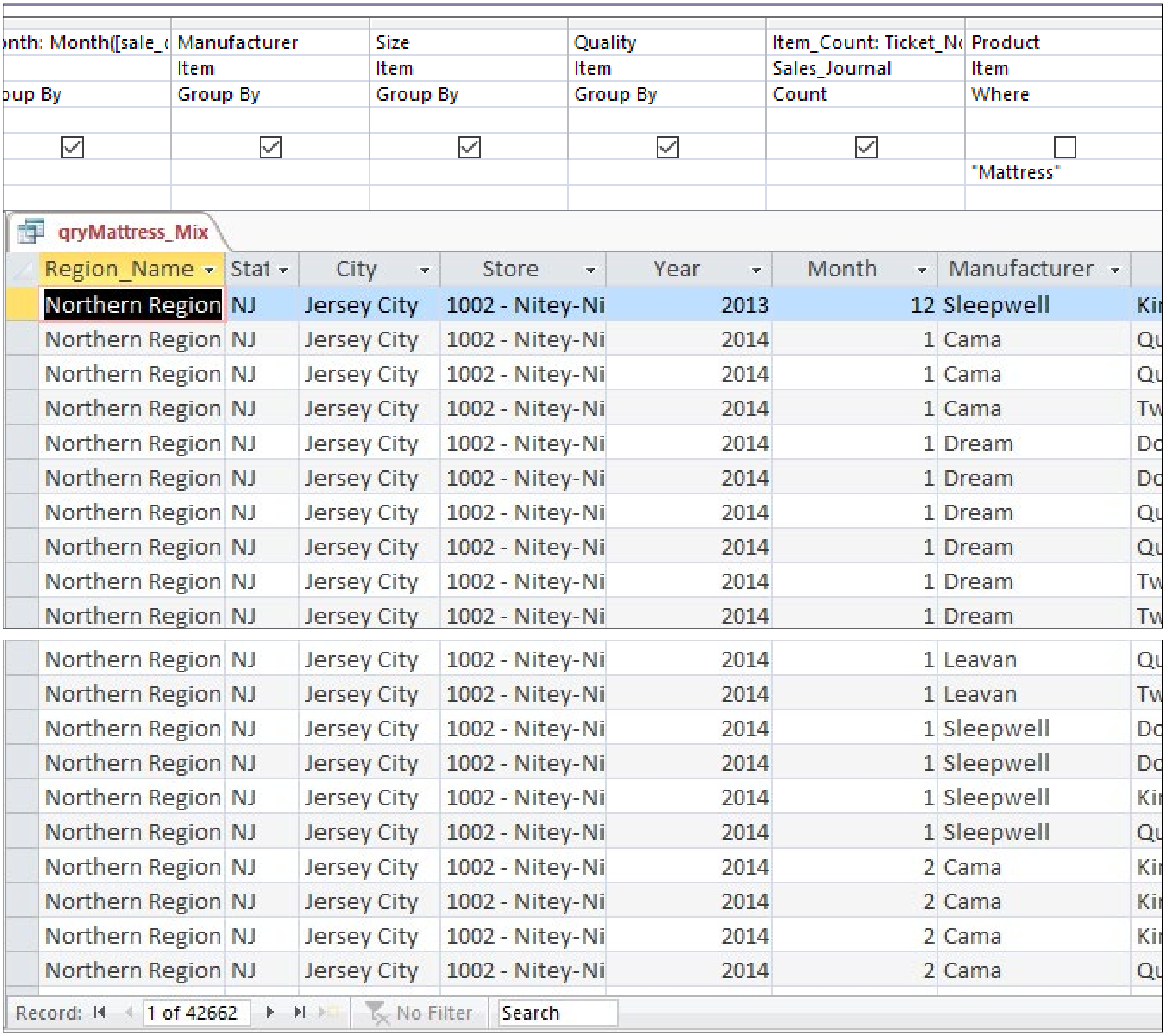


ield:	Region_Name	State	City	Store: [store_no] & " -	Year: Year([sale_date])	Month: Mor
able:	Stores	Stores	Stores			
Sort:						
how:	☑	☑	☑	☑	☑	☑
teria:						
or:						

Figure 15.27

10. *Bring in the* **Manufacturer**, **Size**, *and* **Quality** *fields from the* **Item** *table into the query.*
11. *Click on the* **Totals** *icon to group the fields.*
12. *Use the* **COUNT()** *function to count the* **Ticket_No** *field from the* **Sales_Journal** *table. Call that field* ***Item_Count***.
13. *Bring in the* **Product** *field. Make that field contain a* **Where** *clause and filter it for* ***Mattress***.

14. **Save** *the query as* ***qryMattress_Mix****.*



nth: Month([sale_(	Manufacturer	Size	Quality	Item_Count: Ticket_N(	Product
	Item	Item	Item	Sales_Journal	Item
oup By	Group By	Group By	Group By	Count	Where
☑	☑	☑	☑	☑	☐
					"Mattress"

qryMattress_Mix

Region_Name	Stat	City	Store	Year	Month	Manufacturer	
Northern Region	NJ	Jersey City	1002 - Nitey-Ni	2013	12	Sleepwell	Ki
Northern Region	NJ	Jersey City	1002 - Nitey-Ni	2014	1	Cama	Qu
Northern Region	NJ	Jersey City	1002 - Nitey-Ni	2014	1	Cama	Qu
Northern Region	NJ	Jersey City	1002 - Nitey-Ni	2014	1	Cama	Tw
Northern Region	NJ	Jersey City	1002 - Nitey-Ni	2014	1	Dream	Do
Northern Region	NJ	Jersey City	1002 - Nitey-Ni	2014	1	Dream	Do
Northern Region	NJ	Jersey City	1002 - Nitey-Ni	2014	1	Dream	Qu
Northern Region	NJ	Jersey City	1002 - Nitey-Ni	2014	1	Dream	Qu
Northern Region	NJ	Jersey City	1002 - Nitey-Ni	2014	1	Dream	Tw
Northern Region	NJ	Jersey City	1002 - Nitey-Ni	2014	1	Dream	Tw
Northern Region	NJ	Jersey City	1002 - Nitey-Ni	2014	1	Leavan	Qu
Northern Region	NJ	Jersey City	1002 - Nitey-Ni	2014	1	Leavan	Tw
Northern Region	NJ	Jersey City	1002 - Nitey-Ni	2014	1	Sleepwell	Do
Northern Region	NJ	Jersey City	1002 - Nitey-Ni	2014	1	Sleepwell	Do
Northern Region	NJ	Jersey City	1002 - Nitey-Ni	2014	1	Sleepwell	Ki
Northern Region	NJ	Jersey City	1002 - Nitey-Ni	2014	1	Sleepwell	Qu
Northern Region	NJ	Jersey City	1002 - Nitey-Ni	2014	2	Cama	Ki
Northern Region	NJ	Jersey City	1002 - Nitey-Ni	2014	2	Cama	Ki
Northern Region	NJ	Jersey City	1002 - Nitey-Ni	2014	2	Cama	Ki
Northern Region	NJ	Jersey City	1002 - Nitey-Ni	2014	2	Cama	Qu

Record: |◀ ◀ 1 of 42662 ▶ ▶| No Filter Search

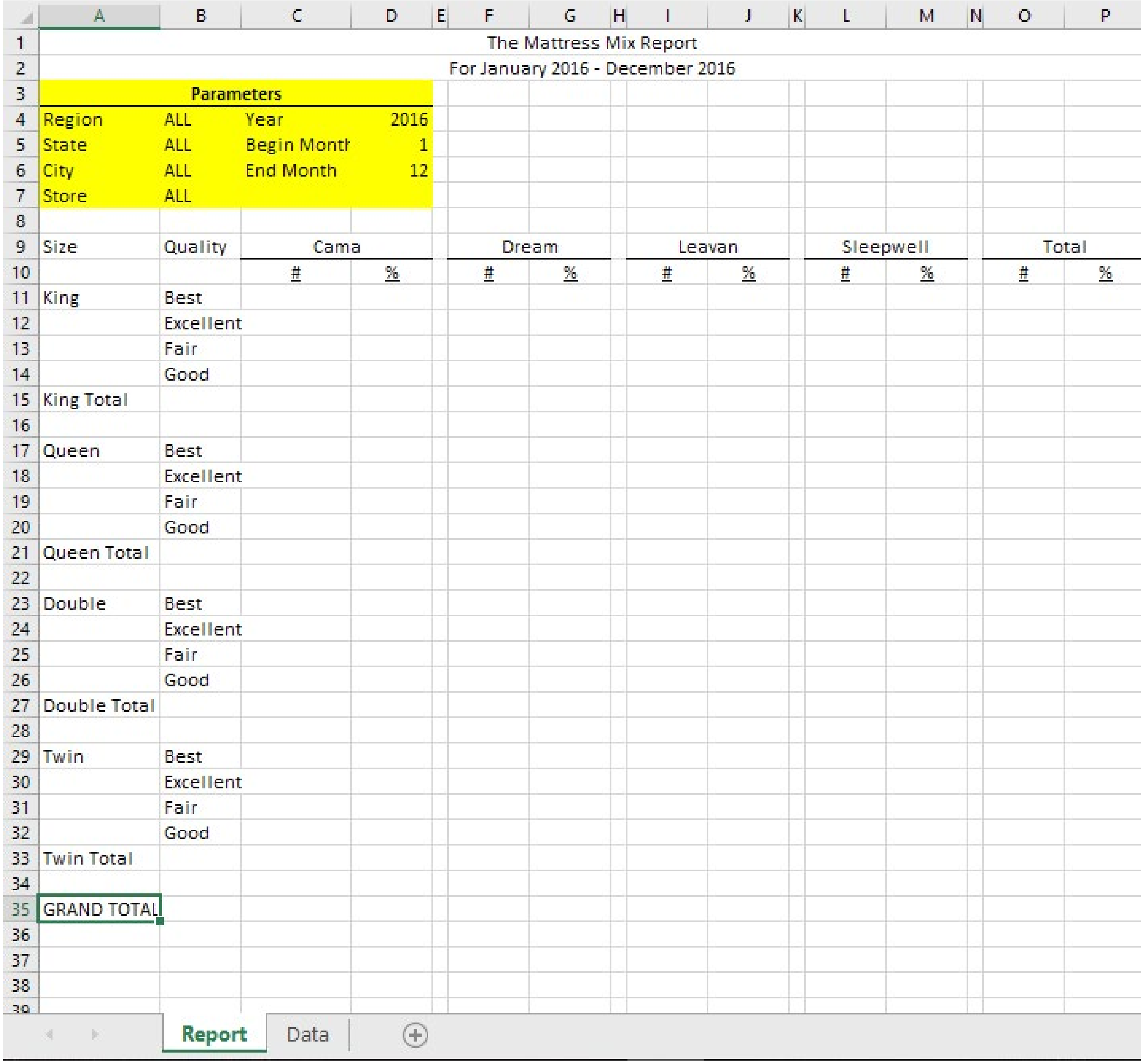
Figure 15.28

15. *Test the query to make sure it is working (you should have 42,662 rows of data).*
16. *Close* **qryMattress_Mix** *and close* **Access***.*

Now that you have your data source, you'll create the shell of the Excel file.

1. *Open a new Excel* **Blank workbook** *and save it as* **C:\ExcelCEO\Access 2016\*Mattress_Mix.xlsx***.*
2. *Rename* **Sheet1** *as* ***Report****, add* **Sheet2** *and rename as* ***Data****.*

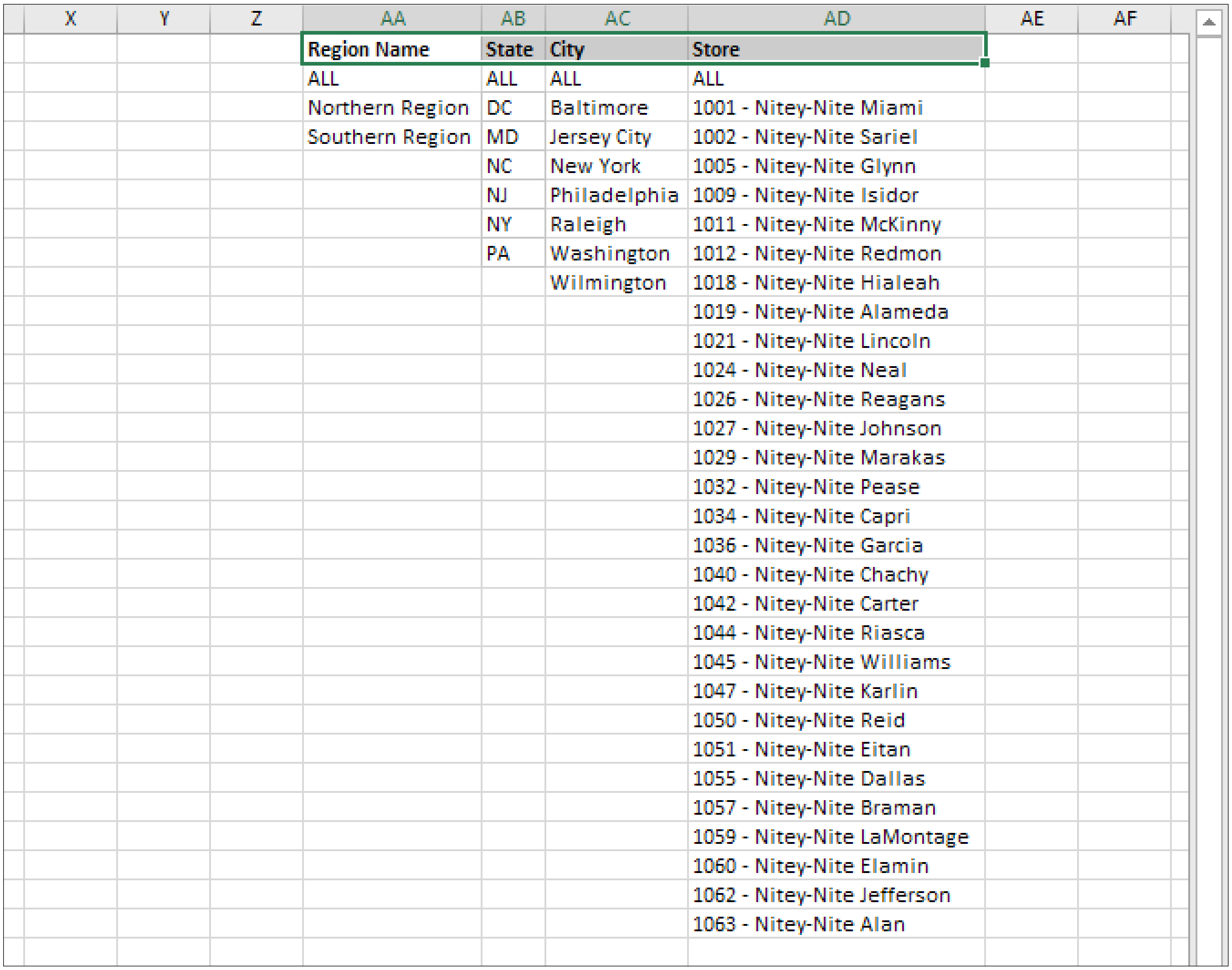
3. *On the* **Report** *tab, create a spreadsheet as in* **Figure 15.29**.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	The Mattress Mix Report															
2	For January 2016 - December 2016															
3	Parameters															
4	Region	ALL	Year	2016												
5	State	ALL	Begin Montr	1												
6	City	ALL	End Month	12												
7	Store	ALL														
8																
9	Size	Quality	Cama			Dream			Leavan			Sleepwell			Total	
10			#	%		#	%		#	%		#	%		#	%
11	King	Best														
12		Excellent														
13		Fair														
14		Good														
15	King Total															
16																
17	Queen	Best														
18		Excellent														
19		Fair														
20		Good														
21	Queen Total															
22																
23	Double	Best														
24		Excellent														
25		Fair														
26		Good														
27	Double Total															
28																
29	Twin	Best														
30		Excellent														
31		Fair														
32		Good														
33	Twin Total															
34																
35	GRAND TOTAL															
36																
37																
38																

Figure 15.29

4. *On the* **Report** *tab, go to* **Cell AA1** *and input the values according to* **Figure 15.30**.

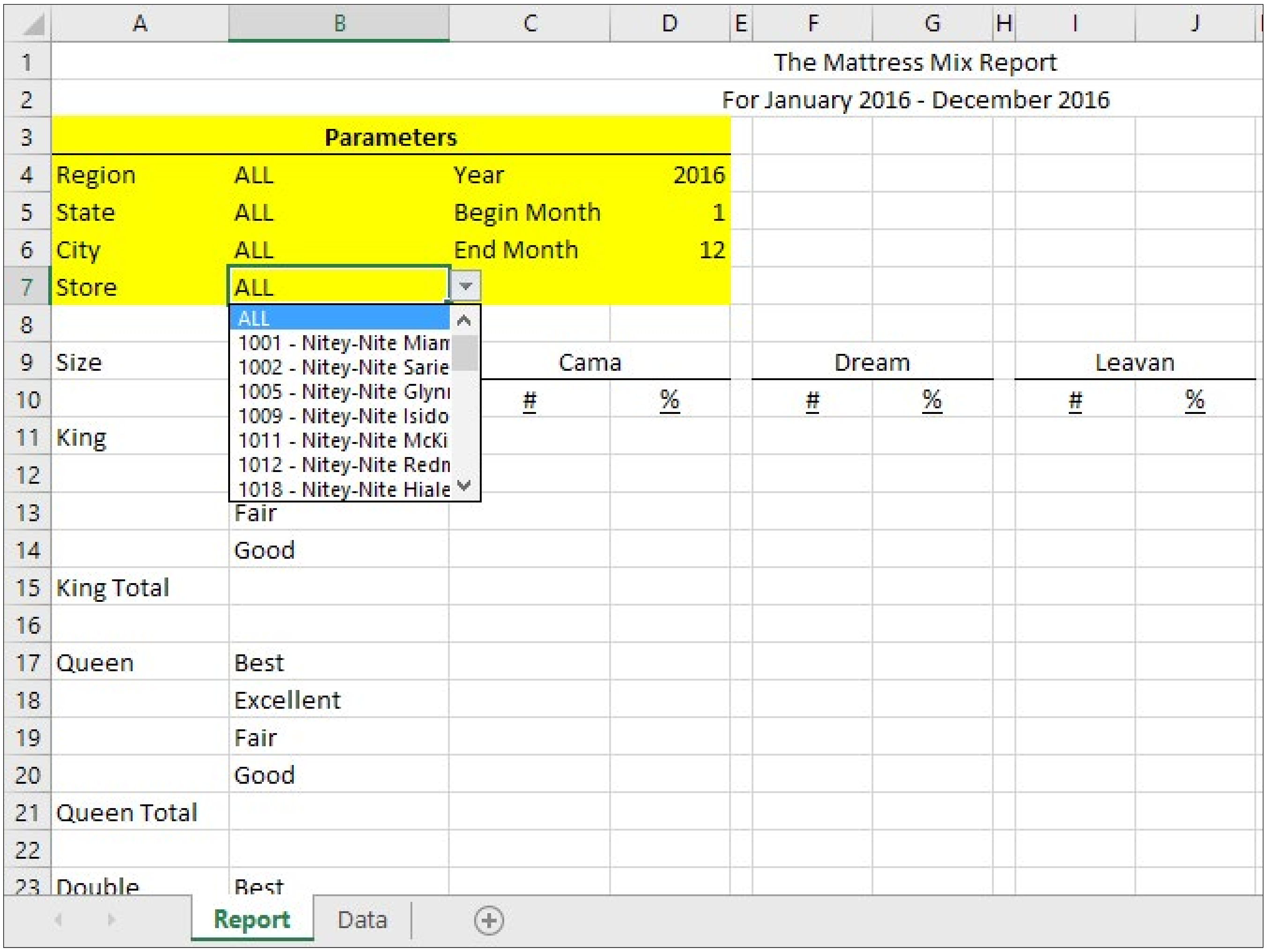


Region Name	State	City	Store
ALL	ALL	ALL	ALL
Northern Region	DC	Baltimore	1001 - Nitey-Nite Miami
Southern Region	MD	Jersey City	1002 - Nitey-Nite Sariel
	NC	New York	1005 - Nitey-Nite Glynn
	NJ	Philadelphia	1009 - Nitey-Nite Isidor
	NY	Raleigh	1011 - Nitey-Nite McKinny
	PA	Washington	1012 - Nitey-Nite Redmon
		Wilmington	1018 - Nitey-Nite Hialeah
			1019 - Nitey-Nite Alameda
			1021 - Nitey-Nite Lincoln
			1024 - Nitey-Nite Neal
			1026 - Nitey-Nite Reagans
			1027 - Nitey-Nite Johnson
			1029 - Nitey-Nite Marakas
			1032 - Nitey-Nite Pease
			1034 - Nitey-Nite Capri
			1036 - Nitey-Nite Garcia
			1040 - Nitey-Nite Chachy
			1042 - Nitey-Nite Carter
			1044 - Nitey-Nite Riasca
			1045 - Nitey-Nite Williams
			1047 - Nitey-Nite Karlin
			1050 - Nitey-Nite Reid
			1051 - Nitey-Nite Eitan
			1055 - Nitey-Nite Dallas
			1057 - Nitey-Nite Braman
			1059 - Nitey-Nite LaMontage
			1060 - Nitey-Nite Elamin
			1062 - Nitey-Nite Jefferson
			1063 - Nitey-Nite Alan

Figure 15.30

These columns contain the values we will use in a list for Cells B4 through B7. To create these ranges, all I did was open the queries in Mattress_Mix.accdb that we used for the Form combo boxes to group each of the Region, State and City fields. There is no point in doing work again that you have already done. It's probably easier to do in Access than to type in all the values, especially for the Store field (think about if store numbers were to change, or if Nitey-Nite opened new stores - it would be handy to have your query stay up-to-date). If you have a similar spreadsheet but with many more rows, you may want to consider having the list tied to an Access query.

5. *In* **Cells B4** *through* **B7**, *create data validation lists (***Data**, **Data Validation**, **Allow**, **List***) that use these ranges in the indicated cells.*



	A	B	C	D	E	F	G	H	I	J
1						The Mattress Mix Report				
2						For January 2016 - December 2016				
3		Parameters								
4	Region	ALL	Year	2016						
5	State	ALL	Begin Month	1						
6	City	ALL	End Month	12						
7	Store	ALL								
8										
9	Size		Cama			Dream			Leavan	
10			#	%		#	%		#	%
11	King									
12										
13		Fair								
14		Good								
15	King Total									
16										
17	Queen	Best								
18		Excellent								
19		Fair								
20		Good								
21	Queen Total									
22										
23	Double	Best								

ALL
1001 - Nitey-Nite Mian
1002 - Nitey-Nite Sarie
1005 - Nitey-Nite Glyni
1009 - Nitey-Nite Isido
1011 - Nitey-Nite McKi
1012 - Nitey-Nite Redn
1018 - Nitey-Nite Hiale

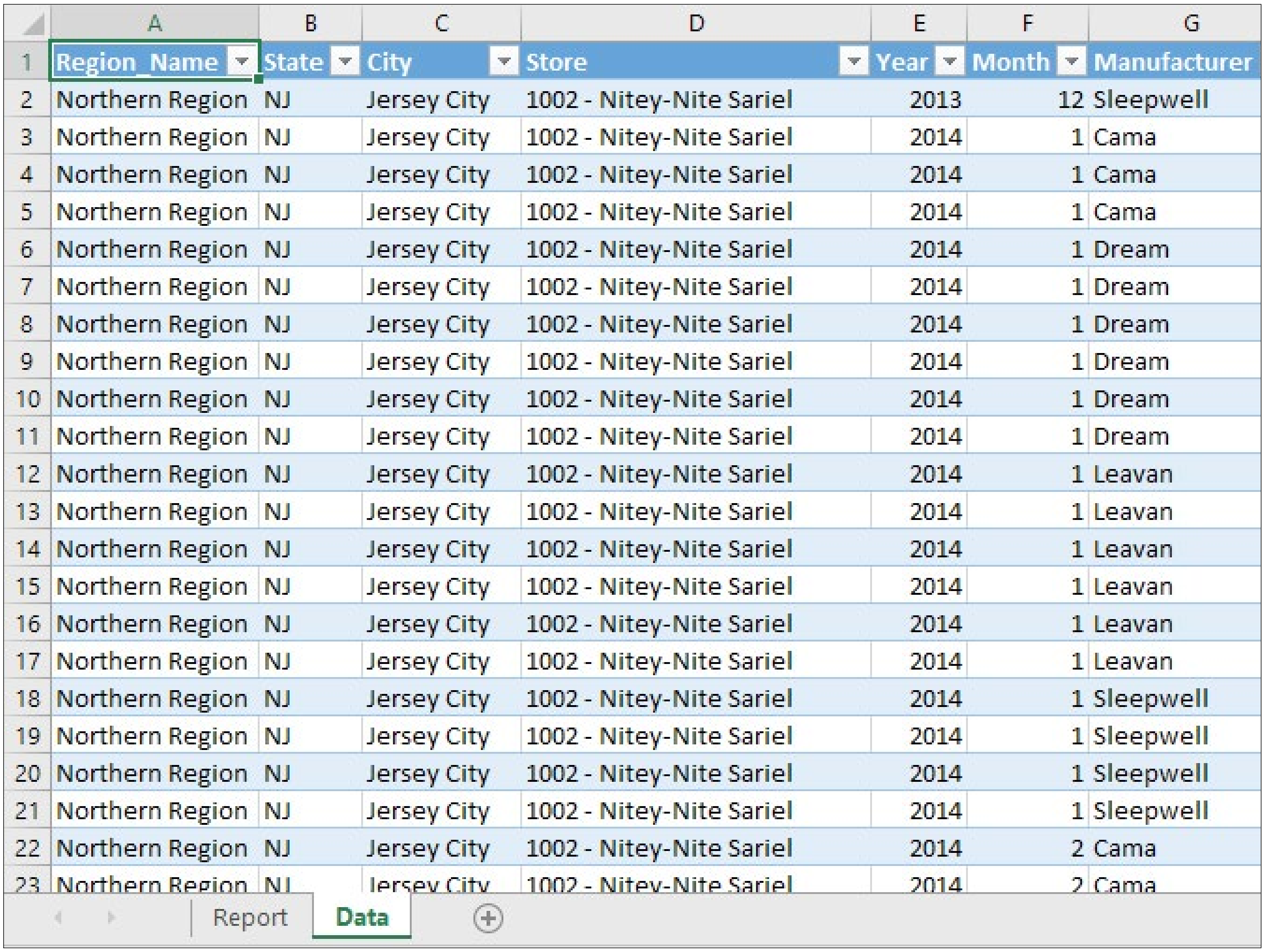
Figure 15.31

6. *For* **Cell D4**, *create a* **Data validation list** *that contains the values* **2014**, **2015**, **2016**, *and* **2017**.
7. *In* **Cells D5** *and* **D6**, *create a* **Data validation list** *that contains the values* ***1*** - ***12***.

Now that you have your parameters populated, you can write the procedure that pulls the data into the Data tab. Since the total record set contains only 42,662 records, we'll bring it all into the Data tab, then program the parameters.

1. *Click on* **Cell A1** *of the* **Data** *tab.*
2. *On the* **Office Ribbon Data** *tab, click on the* **From Other Sources** *icon in the* **Get External Data** *group and choose* **From Microsoft Query**.
3. *In the* **Choose Data Source** *dialog box in the* **Databases** *tab, click on* **MS Access Database*** *and click* **OK**.
4. *In the* **Select Database** *dialog box, navigate to the* **C:\ExcelCEO\Access 2016** *folder and click on the* **Mattress_Mix.accdb** *file and click* **OK**.

5. *In the* **Query Wizard - Choose Columns** *dialog box, bring in all fields from* **qryMattress_Mix**, *click* **Next >** *three times, then click* **Finish** *in the* **Query Wizard - Finish** *dialog box.*
6. *After a few seconds, the* **Import Data** *dialog box should respond to put the data in* **Cell A1**. *Click* **OK**.

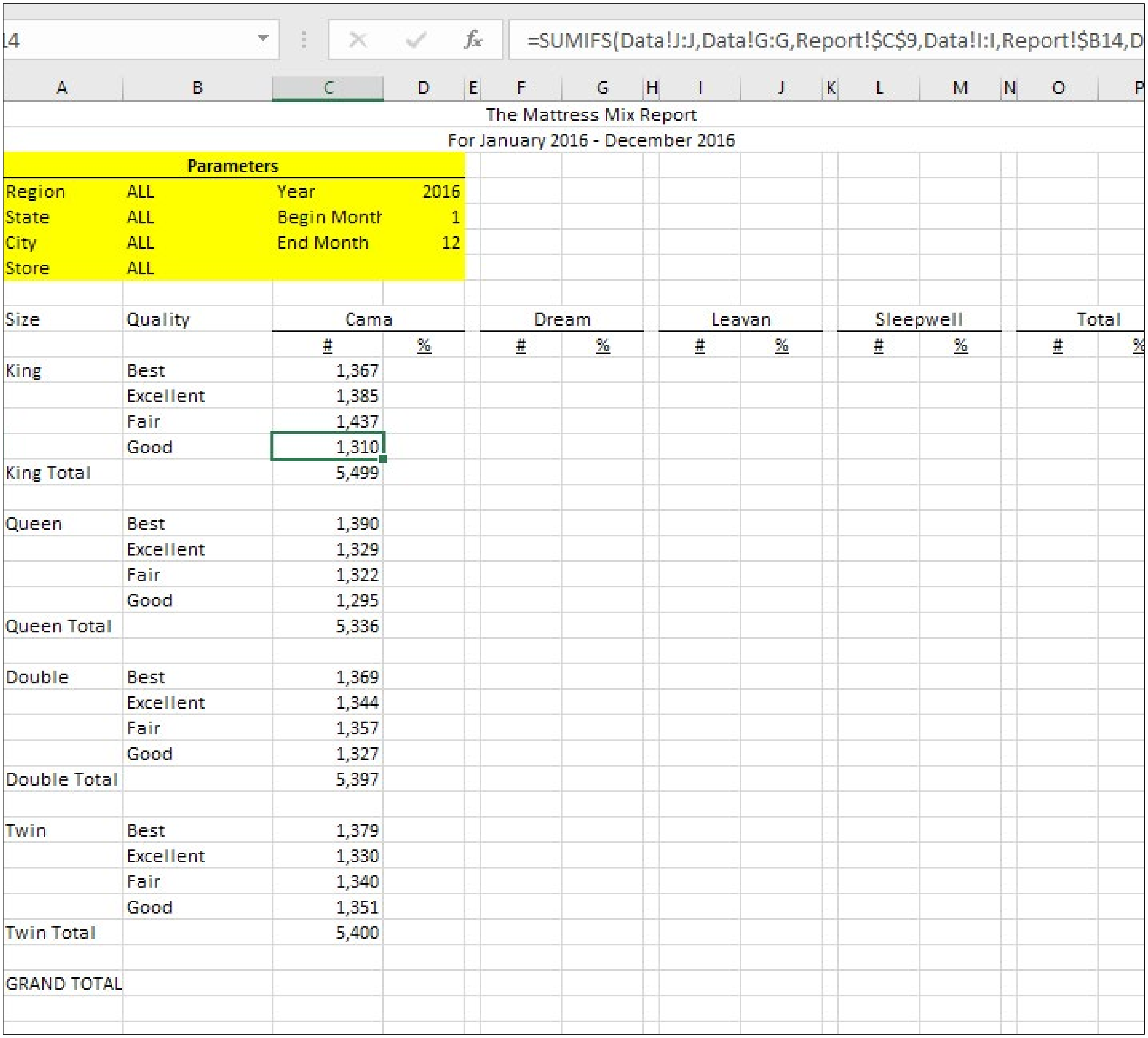


	A	B	C	D	E	F	G
1	Region_Name	State	City	Store	Year	Month	Manufacturer
2	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2013	12	Sleepwell
3	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2014	1	Cama
4	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2014	1	Cama
5	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2014	1	Cama
6	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2014	1	Dream
7	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2014	1	Dream
8	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2014	1	Dream
9	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2014	1	Dream
10	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2014	1	Dream
11	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2014	1	Dream
12	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2014	1	Leavan
13	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2014	1	Leavan
14	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2014	1	Leavan
15	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2014	1	Leavan
16	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2014	1	Leavan
17	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2014	1	Leavan
18	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2014	1	Sleepwell
19	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2014	1	Sleepwell
20	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2014	1	Sleepwell
21	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2014	1	Sleepwell
22	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2014	2	Cama
23	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2014	2	Cama

Figure 15.32

Now that you have all of the data in the Data tab and you know that procedure works, you can create all of the formulas in the report. Which formula would be the best to use to pull the data from the Data tab over into the report? Let's think about it. Once we have the parameters working, the data set will contain all of the data we need, filtered by Region, State, City, or Store. We need to distinguish between the manufacturer, the size and quality, and sum up the item_count column. It seems to me that a SUMIF() or a SUMIFS() function would do the trick. If we are to use a SUMIF() function, the manufacturer, size, and quality need to be in one field so we can use it in the criteria argument (the middle argument of a SUMIF() function). You could go back to your query and create a field that concatenates all of those fields, but if you use a SUMIFS() function (a new function from Excel 2010), you could probably do it without creating that additional field. So let's use the SUMIFS() function.

1. *Write a* **SUMIFS()** *function in* **Cell C11** *of the* **Report** *tab that pulls in the count of the items in the* **Data** *tab, based on the* **Manufacturer**, **Size**, *and* **Quality** *of the mattress sales.*
2. **Copy** *that formula down to the cells below, changing the formula as needed.*

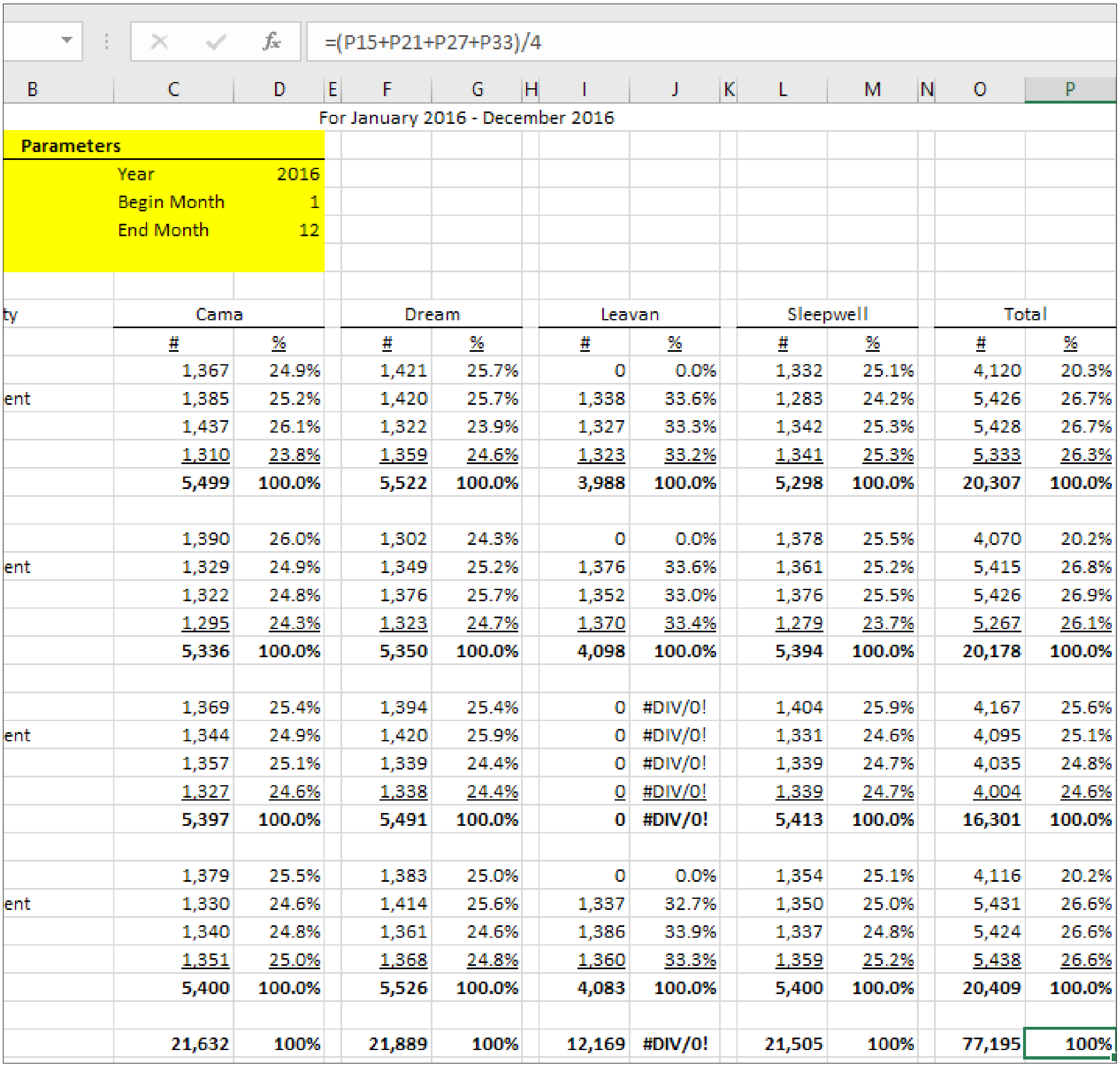


14 | =SUMIFS(Data!J:J,Data!G:G,Report!$C$9,Data!I:I,Report!$B14,D

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
					The Mattress Mix Report										
					For January 2016 - December 2016										
Parameters															
Region	ALL	Year	2016												
State	ALL	Begin Montł	1												
City	ALL	End Month	12												
Store	ALL														
Size	Quality	Cama			Dream			Leavan			Sleepwell			Total	
		#	%		#	%		#	%		#	%		#	%
King	Best	1,367													
	Excellent	1,385													
	Fair	1,437													
	Good	1,310													
King Total		5,499													
Queen	Best	1,390													
	Excellent	1,329													
	Fair	1,322													
	Good	1,295													
Queen Total		5,336													
Double	Best	1,369													
	Excellent	1,344													
	Fair	1,357													
	Good	1,327													
Double Total		5,397													
Twin	Best	1,379													
	Excellent	1,330													
	Fair	1,340													
	Good	1,351													
Twin Total		5,400													
GRAND TOTAL															

Figure 15.33

3. *Create the appropriate* % *of total formulas in* **Column D** *formatted as* **Percent, one decimal place**.
4. **Bold** *the* **Total** *lines and* **underline** *the* # *and* % *data above the* **Total** *lines.*
5. **Copy** *of the formulas to the remaining cells to tie with the figures in* **Figure 15.34**.



=(P15+P21+P27+P33)/4

B	C	D	F	G	I	J	L	M	O	P
			For January 2016 - December 2016							
Parameters										
	Year	2016								
	Begin Month	1								
	End Month	12								
ty	Cama		Dream		Leavan		Sleepwell		Total	
	#	%	#	%	#	%	#	%	#	%
	1,367	24.9%	1,421	25.7%	0	0.0%	1,332	25.1%	4,120	20.3%
ent	1,385	25.2%	1,420	25.7%	1,338	33.6%	1,283	24.2%	5,426	26.7%
	1,437	26.1%	1,322	23.9%	1,327	33.3%	1,342	25.3%	5,428	26.7%
	1,310	23.8%	1,359	24.6%	1,323	33.2%	1,341	25.3%	5,333	26.3%
	5,499	**100.0%**	**5,522**	**100.0%**	**3,988**	**100.0%**	**5,298**	**100.0%**	**20,307**	**100.0%**
	1,390	26.0%	1,302	24.3%	0	0.0%	1,378	25.5%	4,070	20.2%
ent	1,329	24.9%	1,349	25.2%	1,376	33.6%	1,361	25.2%	5,415	26.8%
	1,322	24.8%	1,376	25.7%	1,352	33.0%	1,376	25.5%	5,426	26.9%
	1,295	24.3%	1,323	24.7%	1,370	33.4%	1,279	23.7%	5,267	26.1%
	5,336	**100.0%**	**5,350**	**100.0%**	**4,098**	**100.0%**	**5,394**	**100.0%**	**20,178**	**100.0%**
	1,369	25.4%	1,394	25.4%	0	#DIV/0!	1,404	25.9%	4,167	25.6%
ent	1,344	24.9%	1,420	25.9%	0	#DIV/0!	1,331	24.6%	4,095	25.1%
	1,357	25.1%	1,339	24.4%	0	#DIV/0!	1,339	24.7%	4,035	24.8%
	1,327	24.6%	1,338	24.4%	0	#DIV/0!	1,339	24.7%	4,004	24.6%
	5,397	**100.0%**	**5,491**	**100.0%**	**0**	**#DIV/0!**	**5,413**	**100.0%**	**16,301**	**100.0%**
	1,379	25.5%	1,383	25.0%	0	0.0%	1,354	25.1%	4,116	20.2%
ent	1,330	24.6%	1,414	25.6%	1,337	32.7%	1,350	25.0%	5,431	26.6%
	1,340	24.8%	1,361	24.6%	1,386	33.9%	1,337	24.8%	5,424	26.6%
	1,351	25.0%	1,368	24.8%	1,360	33.3%	1,359	25.2%	5,438	26.6%
	5,400	**100.0%**	**5,526**	**100.0%**	**4,083**	**100.0%**	**5,400**	**100.0%**	**20,409**	**100.0%**
	21,632	**100%**	**21,889**	**100%**	**12,169**	**#DIV/0!**	**21,505**	**100%**	**77,195**	**100%**

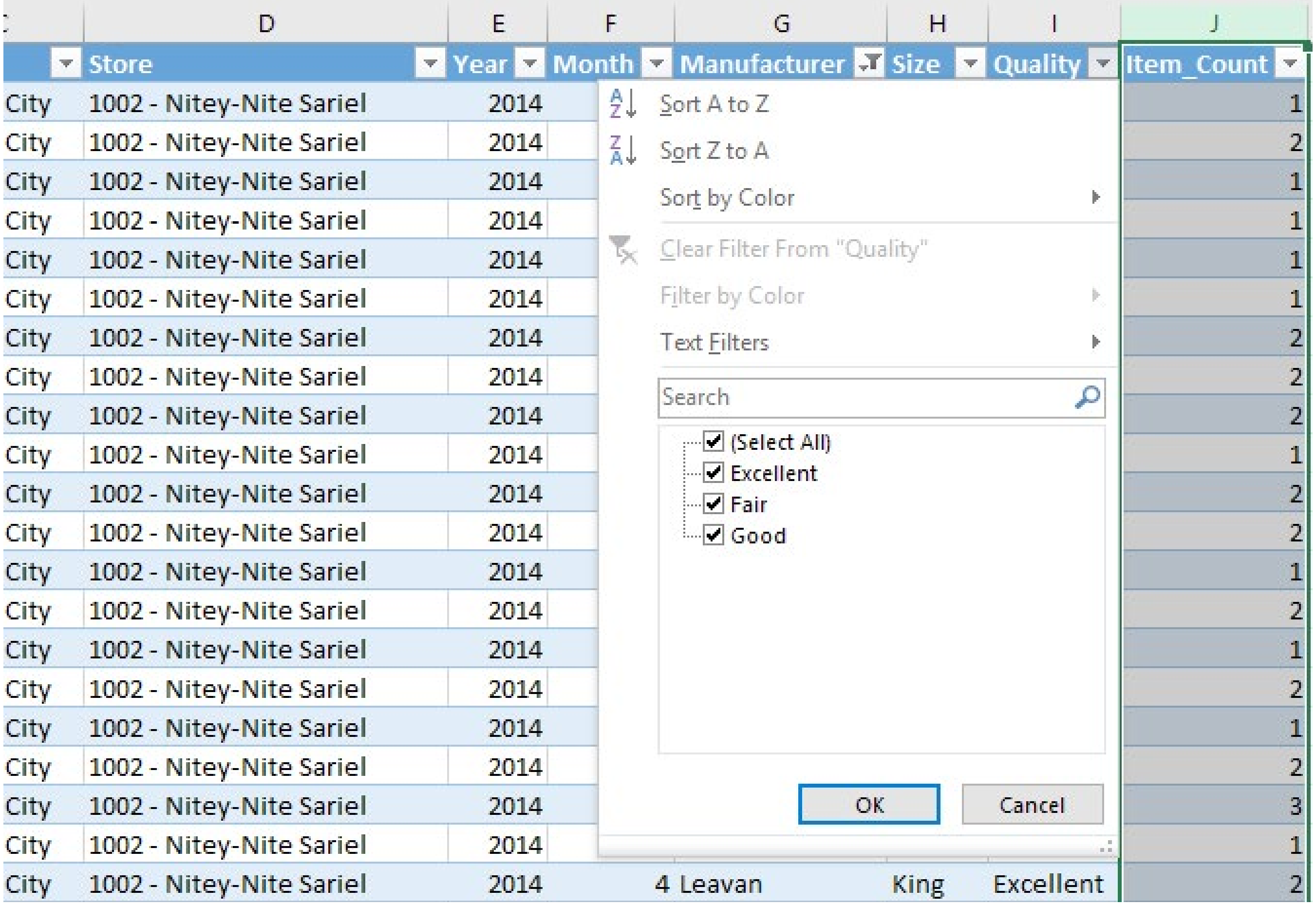
Figure 15.34

Looks pretty good, doesn't it? Just for grins, let's check our total number of 77,195 with the Data tab.

6. *Go to the* **Data** *tab and sum up all of* **Column J** *(Item_Count).*

You will see that it sums to 81,150, which is more than the 77,195 number we have in our report. If you look at the report, you see that Leavan has a lot of zeros. Two things stick out: 1) It does not have any mattresses sold in the Best category and 2) There are no Double size sales at all. Let's check it out on the Data tab.

7. *In the* **Data** *tab, filter the table for* **Leavan**.
8. *With the* **Leavan** *filter on, click on the* **Quality** *filter.*

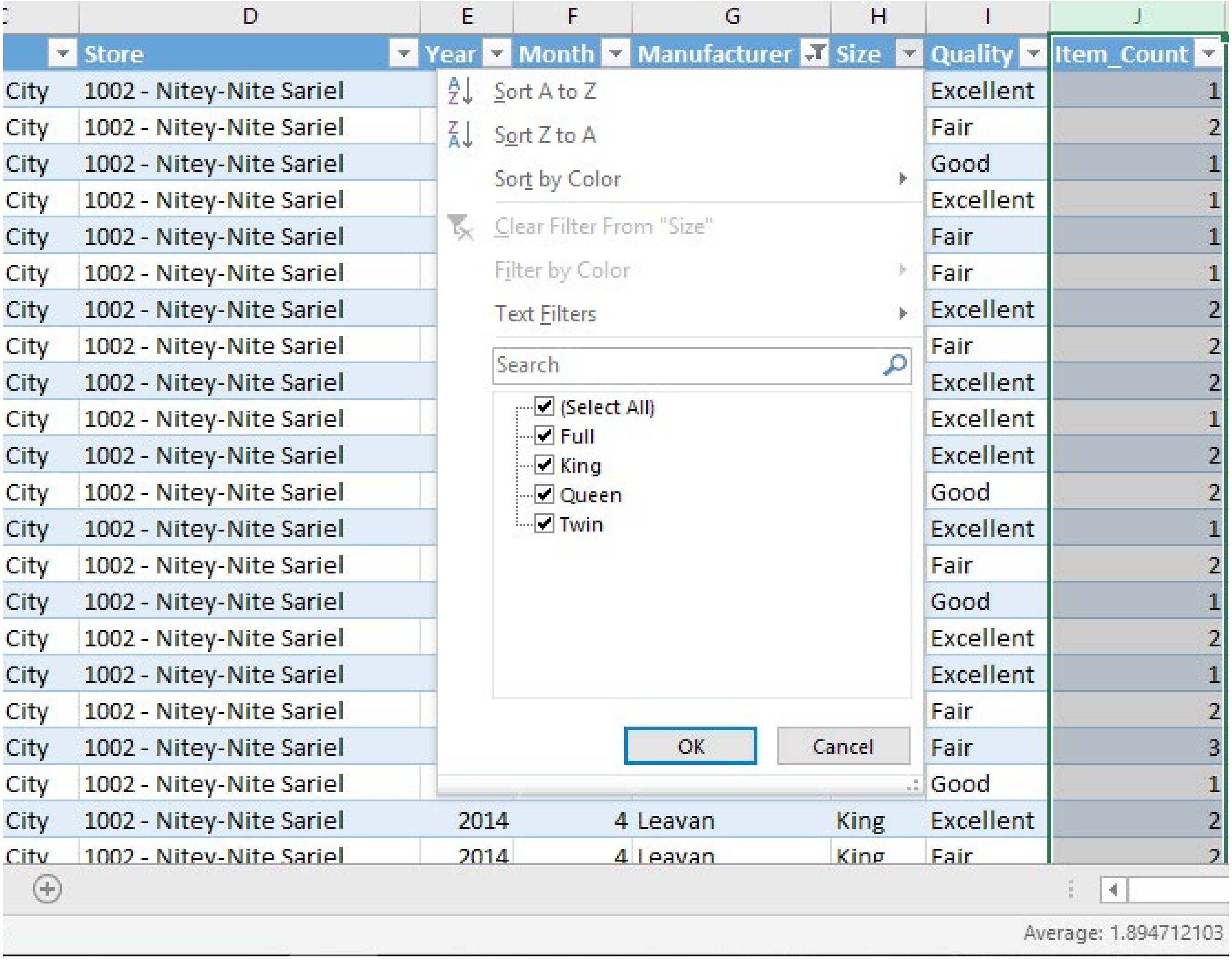


	Store	Year	Month	Manufacturer	Size	Quality	Item_Count
City	1002 - Nitey-Nite Sariel	2014					1
City	1002 - Nitey-Nite Sariel	2014					2
City	1002 - Nitey-Nite Sariel	2014					1
City	1002 - Nitey-Nite Sariel	2014					1
City	1002 - Nitey-Nite Sariel	2014					1
City	1002 - Nitey-Nite Sariel	2014					1
City	1002 - Nitey-Nite Sariel	2014					2
City	1002 - Nitey-Nite Sariel	2014					2
City	1002 - Nitey-Nite Sariel	2014					2
City	1002 - Nitey-Nite Sariel	2014					1
City	1002 - Nitey-Nite Sariel	2014					2
City	1002 - Nitey-Nite Sariel	2014					2
City	1002 - Nitey-Nite Sariel	2014					1
City	1002 - Nitey-Nite Sariel	2014					2
City	1002 - Nitey-Nite Sariel	2014					1
City	1002 - Nitey-Nite Sariel	2014					2
City	1002 - Nitey-Nite Sariel	2014					1
City	1002 - Nitey-Nite Sariel	2014					2
City	1002 - Nitey-Nite Sariel	2014					3
City	1002 - Nitey-Nite Sariel	2014					1
City	1002 - Nitey-Nite Sariel	2014	4	Leavan	King	Excellent	2

Figure 15.35

You see there is no Best quality option there, so those numbers must be correct. Let's look at the Size field.

9. *With the* **Leavan** *filter on, click on the* **Size** *filter.*



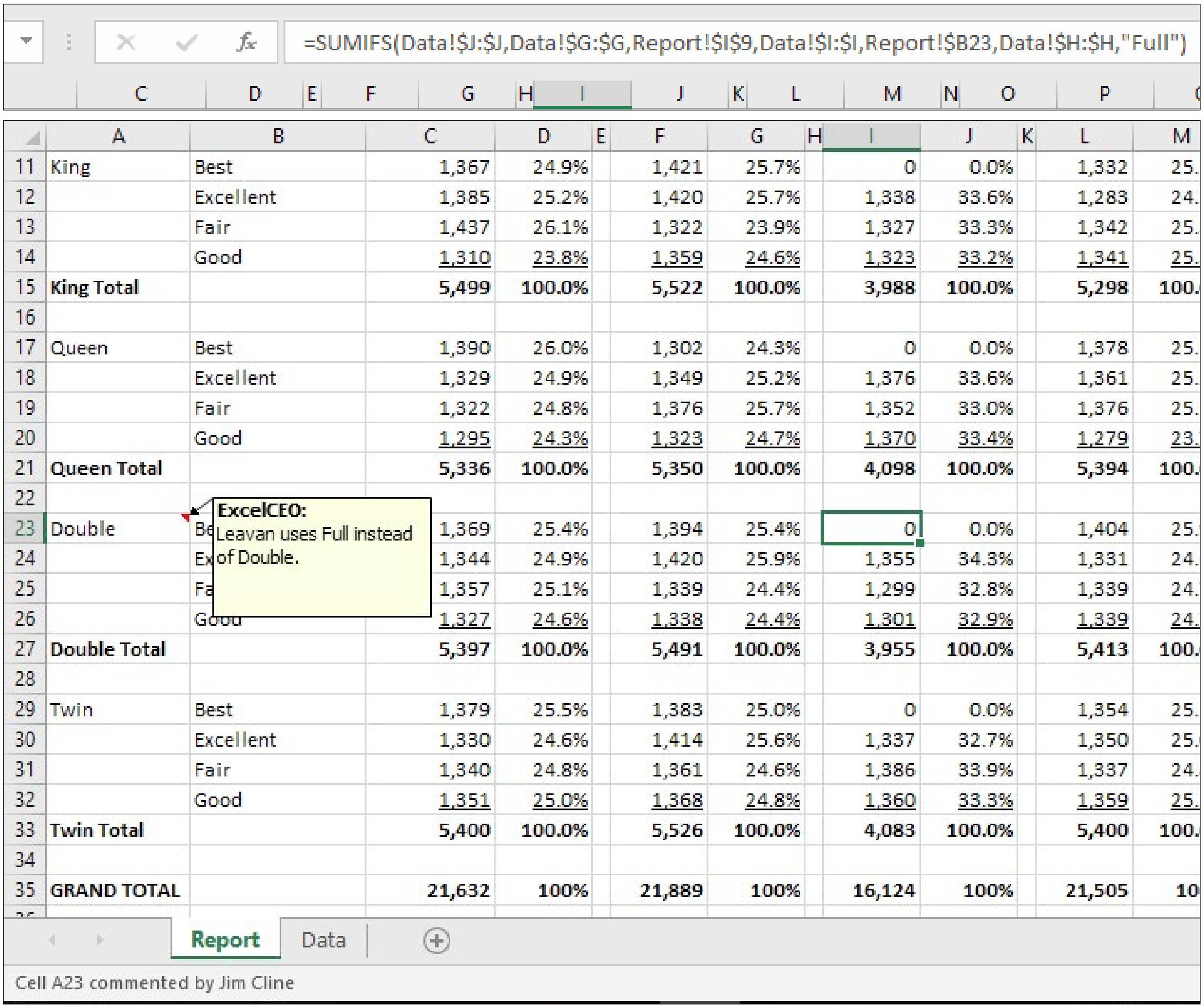
C	D	E	F	G	H	I	J
	Store	Year	Month	Manufacturer	Size	Quality	Item_Count
City	1002 - Nitey-Nite Sariel					Excellent	1
City	1002 - Nitey-Nite Sariel					Fair	2
City	1002 - Nitey-Nite Sariel					Good	1
City	1002 - Nitey-Nite Sariel					Excellent	1
City	1002 - Nitey-Nite Sariel					Fair	1
City	1002 - Nitey-Nite Sariel					Fair	1
City	1002 - Nitey-Nite Sariel					Excellent	2
City	1002 - Nitey-Nite Sariel					Fair	2
City	1002 - Nitey-Nite Sariel					Excellent	2
City	1002 - Nitey-Nite Sariel					Excellent	1
City	1002 - Nitey-Nite Sariel					Excellent	2
City	1002 - Nitey-Nite Sariel					Good	2
City	1002 - Nitey-Nite Sariel					Excellent	1
City	1002 - Nitey-Nite Sariel					Fair	2
City	1002 - Nitey-Nite Sariel					Good	1
City	1002 - Nitey-Nite Sariel					Excellent	2
City	1002 - Nitey-Nite Sariel					Excellent	1
City	1002 - Nitey-Nite Sariel					Fair	2
City	1002 - Nitey-Nite Sariel					Fair	3
City	1002 - Nitey-Nite Sariel					Good	1
City	1002 - Nitey-Nite Sariel	2014	4	Leavan	King	Excellent	2
City	1002 - Nitey-Nite Sariel	2014	4	Leavan	King	Fair	2

Average: 1.894712103

Figure 15.36

Ah-ha! There is no Double size for Leavan, but they have it listed as Full. We could go through a major project just to change the data using SQL code, but you're probably getting tired by now, so let's just correct it in our SUMIFS() formula.

10. *Undo all the filters in the* **Data** *tab then click on the* **Report** *tab.*
11. *Change the formula in* **Cell I23** *to reflect a* ***Full*** *size instead of* **Double** *and copy to the appropriate cells below.*
12. **Insert** *a* **comment** *in* **Cell A23** *to let the users know that for* **Leavan** *you used* **Full** *instead of* **Double**.



=SUMIFS(Data!$J:$J,Data!$G:$G,Report!$I$9,Data!$I:$I,Report!$B23,Data!$H:$H,"Full")

	A	B	C	D	E	F	G	H	I	J	K	L	M
11	King	Best	1,367	24.9%		1,421	25.7%		0	0.0%		1,332	25.
12		Excellent	1,385	25.2%		1,420	25.7%		1,338	33.6%		1,283	24.
13		Fair	1,437	26.1%		1,322	23.9%		1,327	33.3%		1,342	25.
14		Good	1,310	23.8%		1,359	24.6%		1,323	33.2%		1,341	25.
15	**King Total**		**5,499**	**100.0%**		**5,522**	**100.0%**		**3,988**	**100.0%**		**5,298**	**100.**
16													
17	Queen	Best	1,390	26.0%		1,302	24.3%		0	0.0%		1,378	25.
18		Excellent	1,329	24.9%		1,349	25.2%		1,376	33.6%		1,361	25.
19		Fair	1,322	24.8%		1,376	25.7%		1,352	33.0%		1,376	25.
20		Good	1,295	24.3%		1,323	24.7%		1,370	33.4%		1,279	23.
21	**Queen Total**		**5,336**	**100.0%**		**5,350**	**100.0%**		**4,098**	**100.0%**		**5,394**	**100.**
22													
23	Double	Be	1,369	25.4%		1,394	25.4%		0	0.0%		1,404	25.
24		Ex	1,344	24.9%		1,420	25.9%		1,355	34.3%		1,331	24.
25		Fa	1,357	25.1%		1,339	24.4%		1,299	32.8%		1,339	24.
26		Good	1,327	24.6%		1,338	24.4%		1,301	32.9%		1,339	24.
27	**Double Total**		**5,397**	**100.0%**		**5,491**	**100.0%**		**3,955**	**100.0%**		**5,413**	**100.**
28													
29	Twin	Best	1,379	25.5%		1,383	25.0%		0	0.0%		1,354	25.
30		Excellent	1,330	24.6%		1,414	25.6%		1,337	32.7%		1,350	25.
31		Fair	1,340	24.8%		1,361	24.6%		1,386	33.9%		1,337	24.
32		Good	1,351	25.0%		1,368	24.8%		1,360	33.3%		1,359	25.
33	**Twin Total**		**5,400**	**100.0%**		**5,526**	**100.0%**		**4,083**	**100.0%**		**5,400**	**100.**
34													
35	**GRAND TOTAL**		**21,632**	**100%**		**21,889**	**100%**		**16,124**	**100%**		**21,505**	**10**

ExcelCEO:
Leavan uses Full instead of Double.

Cell A23 commented by Jim Cline

Figure 15.37

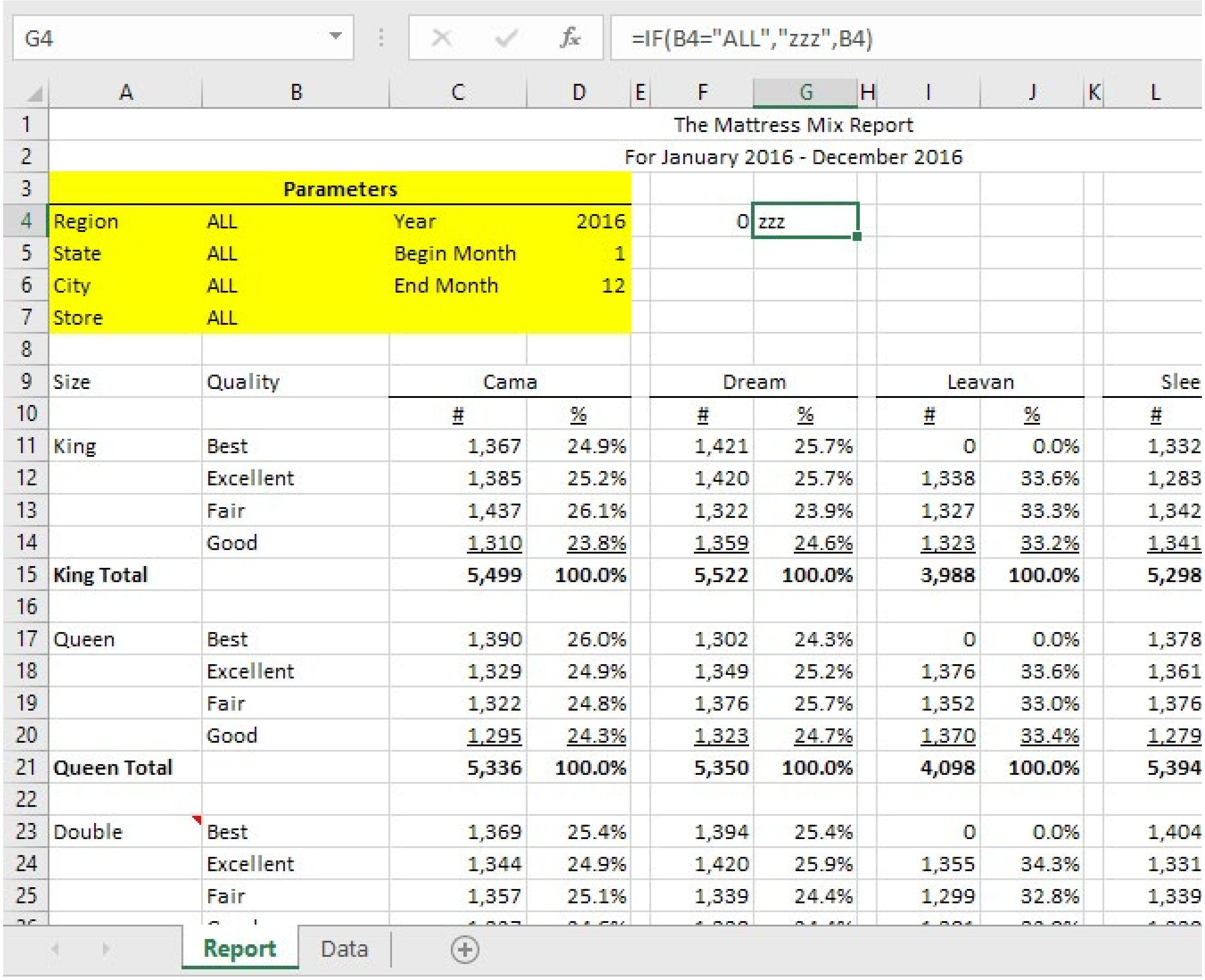
Now your totals should match the total on the Data tab, 81,150.

Now you are ready to program the parameters. Let's discuss those. The Year and Month parameters are easy to understand. When you design the query in the Microsoft Query grid, you just need to filter for the year that equals Cell D4 on the Report tab, and the month for between Cells D5 and D6. But the Region, State, City, and Store parameters are more difficult. You want the user to choose only ONE of those AND the year and month cells. When that happens, you will be forced to program AND() and OR() functions into the Design Grid. That can be very tricky, as you can end up with several OR() functions. Since there are four optional geography levels and three required date levels, you will end up with at least 16 OR() functions. If you don't believe me, just try to program it into Microsoft Query's Design Grid.

One practical way to cut down on the number of criteria that typically involves numerous OR() functions is to use a BETWEEN statement. Notice in the drop-down list for each geographic level I've input the word ALL as the first option. My logic for that is to create formulas in two cells that contain the parameters used in the BETWEEN statement. In the first criteria cell, you will write a simple IF() statement that says if the drop-down cell equals ALL then return a 0, else return the value of the drop-down cell. In the second criteria cell, write another IF() statement that says if the drop-down cell contains ALL then return 'zzz' else return the value of the cell. The BETWEEN statement will pick up the range 0 to 'zzz', which return all numbers and text strings or the value chosen in the drop-down cell.

To illustrate, suppose that you chose Raleigh in the City drop-down cell. The BETWEEN statements for Region, State, and Store would all be BETWEEN 0 and 'zzz' and the statement for City would be BETWEEN Raleigh and Raleigh, which is the same as being equal to Raleigh. Let's create the criteria cells.

1. *In* **Cell F4***, write the following formula:* ***=IF(B4="ALL",0,B4)***
2. *In* **Cell G4***, write the following formula:* ***=IF(B4="ALL","zzz",B4)***

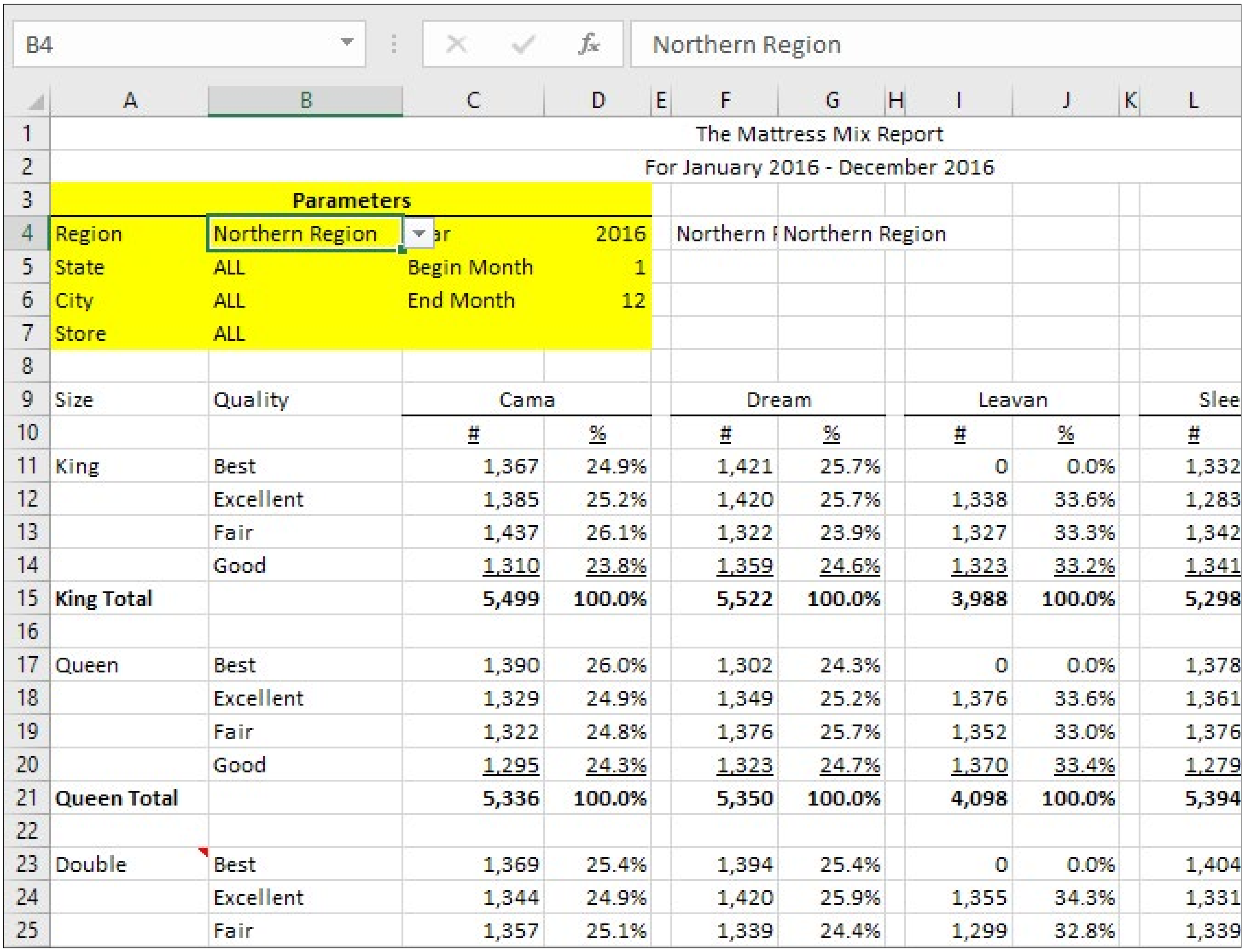


G4 =IF(B4="ALL","zzz",B4)

	A	B	C	D	E	F	G	H	I	J	K	L
1						The Mattress Mix Report						
2						For January 2016 - December 2016						
3		**Parameters**										
4	Region	ALL	Year	2016		0	zzz					
5	State	ALL	Begin Month	1								
6	City	ALL	End Month	12								
7	Store	ALL										
8												
9	Size	Quality	Cama			Dream			Leavan			Slee
10			#	%		#	%		#	%		#
11	King	Best	1,367	24.9%		1,421	25.7%		0	0.0%		1,332
12		Excellent	1,385	25.2%		1,420	25.7%		1,338	33.6%		1,283
13		Fair	1,437	26.1%		1,322	23.9%		1,327	33.3%		1,342
14		Good	1,310	23.8%		1,359	24.6%		1,323	33.2%		1,341
15	**King Total**		**5,499**	**100.0%**		**5,522**	**100.0%**		**3,988**	**100.0%**		**5,298**
16												
17	Queen	Best	1,390	26.0%		1,302	24.3%		0	0.0%		1,378
18		Excellent	1,329	24.9%		1,349	25.2%		1,376	33.6%		1,361
19		Fair	1,322	24.8%		1,376	25.7%		1,352	33.0%		1,376
20		Good	1,295	24.3%		1,323	24.7%		1,370	33.4%		1,279
21	**Queen Total**		**5,336**	**100.0%**		**5,350**	**100.0%**		**4,098**	**100.0%**		**5,394**
22												
23	Double	Best	1,369	25.4%		1,394	25.4%		0	0.0%		1,404
24		Excellent	1,344	24.9%		1,420	25.9%		1,355	34.3%		1,331
25		Fair	1,357	25.1%		1,339	24.4%		1,299	32.8%		1,339

Figure 15.38

3. *Change* **Cell B4** *to be* **Northern Region**.

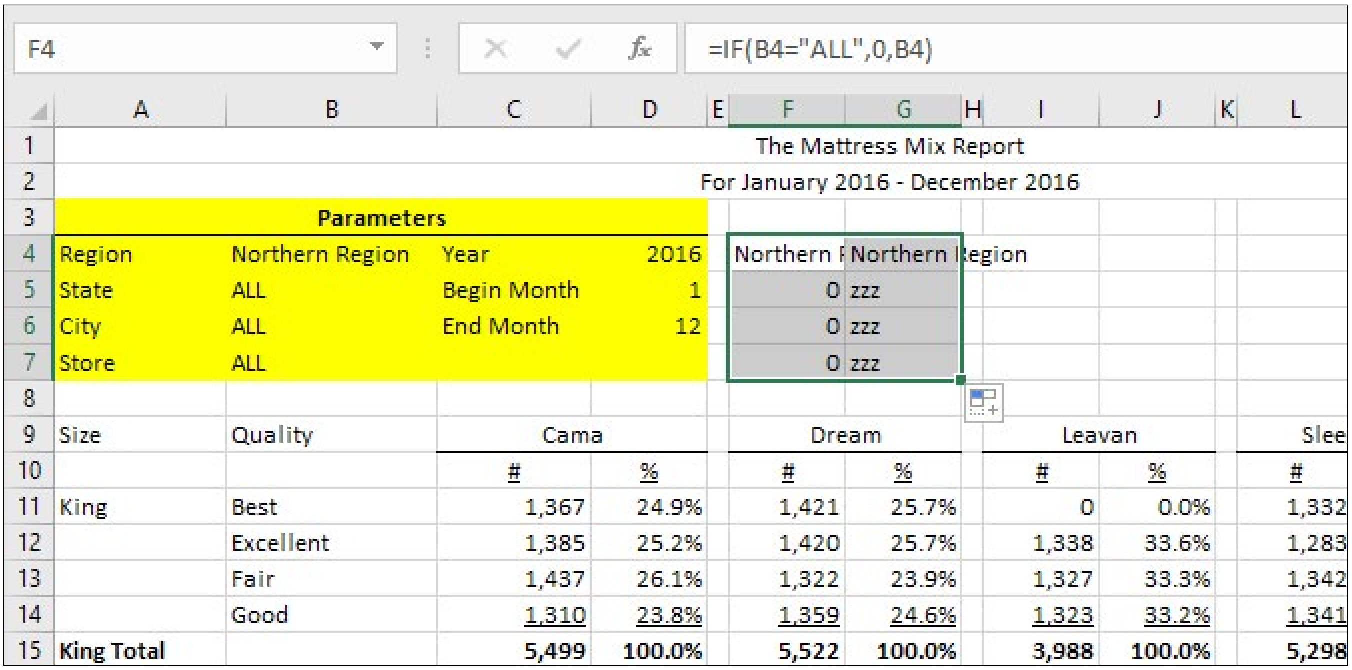


B4 | Northern Region

	A	B	C	D	E	F	G	H	I	J	K	L
1							The Mattress Mix Report					
2							For January 2016 - December 2016					
3		Parameters										
4	Region	Northern Region	ar	2016		Northern R	Northern Region					
5	State	ALL	Begin Month	1								
6	City	ALL	End Month	12								
7	Store	ALL										
8												
9	Size	Quality	Cama			Dream			Leavan			Slee
10			#	%		#	%		#	%		#
11	King	Best	1,367	24.9%		1,421	25.7%		0	0.0%		1,332
12		Excellent	1,385	25.2%		1,420	25.7%		1,338	33.6%		1,283
13		Fair	1,437	26.1%		1,322	23.9%		1,327	33.3%		1,342
14		Good	1,310	23.8%		1,359	24.6%		1,323	33.2%		1,341
15	**King Total**		**5,499**	**100.0%**		**5,522**	**100.0%**		**3,988**	**100.0%**		**5,298**
16												
17	Queen	Best	1,390	26.0%		1,302	24.3%		0	0.0%		1,378
18		Excellent	1,329	24.9%		1,349	25.2%		1,376	33.6%		1,361
19		Fair	1,322	24.8%		1,376	25.7%		1,352	33.0%		1,376
20		Good	1,295	24.3%		1,323	24.7%		1,370	33.4%		1,279
21	**Queen Total**		**5,336**	**100.0%**		**5,350**	**100.0%**		**4,098**	**100.0%**		**5,394**
22												
23	Double	Best	1,369	25.4%		1,394	25.4%		0	0.0%		1,404
24		Excellent	1,344	24.9%		1,420	25.9%		1,355	34.3%		1,331
25		Fair	1,357	25.1%		1,339	24.4%		1,299	32.8%		1,339

Figure 15.39

4. **Copy Cells F4** *and* **G4** *down to* **Cells F5 – G7**.



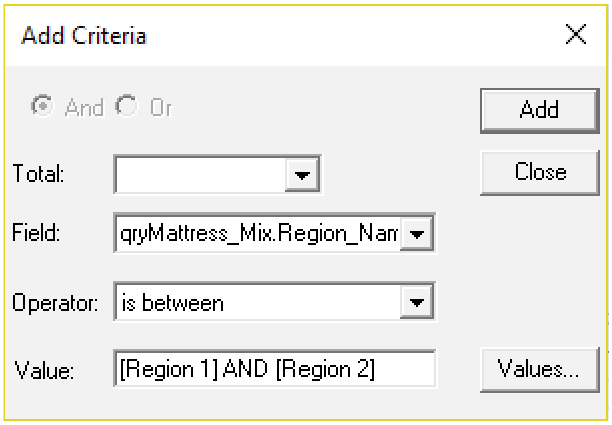
F4 =IF(B4="ALL",0,B4)

	A	B	C	D	E	F	G	H	I	J	K	L
1						The Mattress Mix Report						
2						For January 2016 - December 2016						
3		Parameters										
4	Region	Northern Region	Year	2016		Northern I	Northern Region					
5	State	ALL	Begin Month	1		0	zzz					
6	City	ALL	End Month	12		0	zzz					
7	Store	ALL				0	zzz					
8												
9	Size	Quality	Cama			Dream			Leavan			Slee
10			#	%		#	%		#	%		#
11	King	Best	1,367	24.9%		1,421	25.7%		0	0.0%		1,332
12		Excellent	1,385	25.2%		1,420	25.7%		1,338	33.6%		1,283
13		Fair	1,437	26.1%		1,322	23.9%		1,327	33.3%		1,342
14		Good	1,310	23.8%		1,359	24.6%		1,323	33.2%		1,341
15	King Total		5,499	100.0%		5,522	100.0%		3,988	100.0%		5,298

Figure 15.40

Now you are ready to program the parameters in Microsoft Query.

5. *Open* **Microsoft Query** *to the* **Design Grid** *(***Data** *tab,* **Connections**, **Properties…** *button,* **Definition** *tab,* **Edit Query…** *button)*
6. *Click* **OK** *if you get a message that the query cannot be edited by the* **Query Wizard**.
7. *Click* **Next >** *until you get to the last screen of the* **Query Wizard** *and select the* **View data or edit query in Microsoft Query** *radio button and click* **Finish**.
8. **Create** *the criteria that the* **Region_Name** *is between* ***[Region1] and [Region2]***



Add Criteria

● And ○ Or

Total:

Field: qryMattress_Mix.Region_Nam

Operator: is between

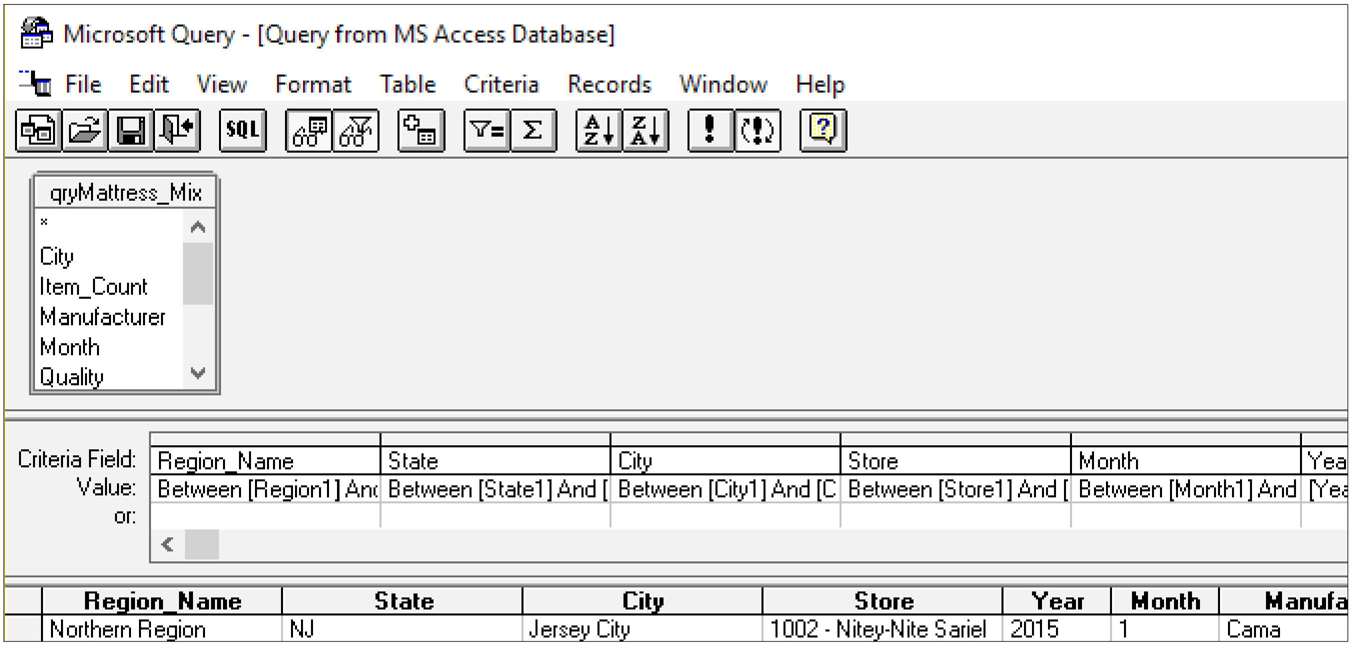
Value: [Region 1] AND [Region 2]

Figure 15.41

9. **Create** *similar criteria for the* **State**, **City**, **Store**, *and* **Month** *fields.*

10. **Create** *a criteria for the* **Year** *field where the year equals* ***[Year1]*** *(Remember, the parameter field name cannot be the same name as the field)*
11. **Run** *the query by clicking on the exclamation point (***Query Now***)* ! *icon.*
12. *Input the following values when prompted:*

Parameter	Text
Region1	***0***
Region2	***zzz***
State1	***0***
State2	***zzz***
City1	***0***
City2	***zzz***
Store1	***0***
Store2	***zzz***
Month1	***1***
Month2	***6***
Year	***2015***


Microsoft Query - [Query from MS Access Database]

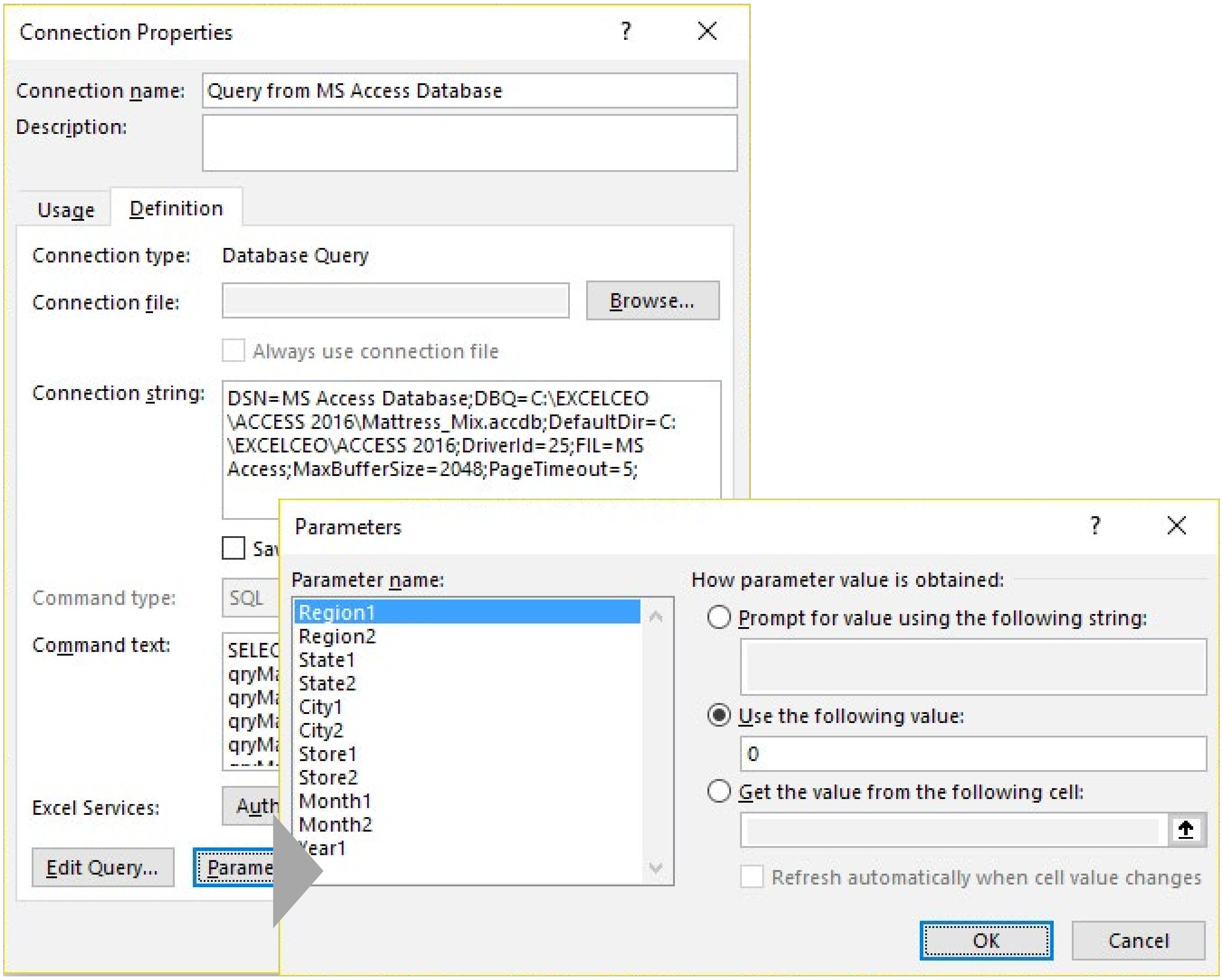
qryMattress_Mix
*
City
Item_Count
Manufacturer
Month
Quality

Criteria Field:	Region_Name	State	City	Store	Month	Yea
Value:	Between [Region1] An	Between [State1] And [	Between [City1] And [C	Between [Store1] And [	Between [Month1] And	[Yea
or:						

Region_Name	State	City	Store	Year	Month	Manufa
Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2015	1	Cama

Figure 15.42

13. *Exit out of* **Microsoft Query**.
14. *In the* **Connection Properties** *dialog box, click on the* **Parameters…** *button.*



Connection Properties

Connection name: Query from MS Access Database

Description:

Usage | Definition

Connection type: Database Query

Connection file: Browse...

□ Always use connection file

Connection string: DSN=MS Access Database;DBQ=C:\EXCELCEO\ACCESS 2016\Mattress_Mix.accdb;DefaultDir=C:\EXCELCEO\ACCESS 2016;DriverId=25;FIL=MS Access;MaxBufferSize=2048;PageTimeout=5;

□ Sa

Command type: SQL

Command text: SELEC qryMa qryMa qryMa qryMa

Excel Services: Auth

Edit Query... | Parame

Parameters

Parameter name:

- Region1
- Region2
- State1
- State2
- City1
- City2
- Store1
- Store2
- Month1
- Month2
- Year1

How parameter value is obtained:

○ Prompt for value using the following string:

● Use the following value: 0

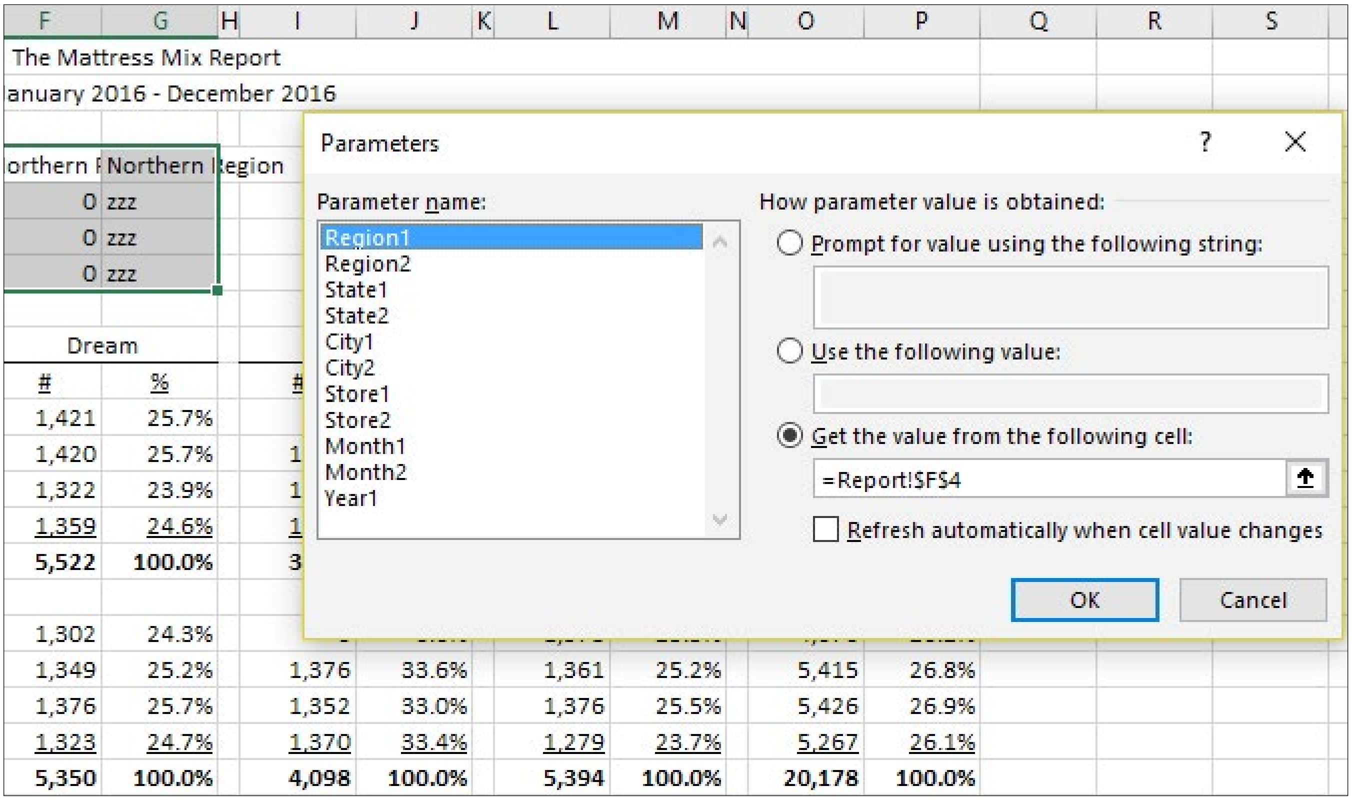
○ Get the value from the following cell:

□ Refresh automatically when cell value changes

OK | Cancel

Figure 15.43

15. *Set each parameter where it gets it value from the appropriate cell (example, Region1 = Report!$F$4, Region2 = Report!$G$4, etc.)*
16. *Make sure the* **Refresh automatically when cell value changes** *box is not checked.*



The Mattress Mix Report

anuary 2016 - December 2016

F	G	H	I	J	K	L	M	N	O	P
orthern I	Northern Region									
0	zzz									
0	zzz									
0	zzz									
Dream										
#	%		#							
1,421	25.7%									
1,420	25.7%		1							
1,322	23.9%		1							
1,359	24.6%		1							
5,522	100.0%		3							
1,302	24.3%									
1,349	25.2%		1,376	33.6%		1,361	25.2%		5,415	26.8%
1,376	25.7%		1,352	33.0%		1,376	25.5%		5,426	26.9%
1,323	24.7%		1,370	33.4%		1,279	23.7%		5,267	26.1%
5,350	100.0%		4,098	100.0%		5,394	100.0%		20,178	100.0%

Parameters

Parameter name: Region1, Region2, State1, State2, City1, City2, Store1, Store2, Month1, Month2, Year1 (Region1 selected)

How parameter value is obtained:

○ Prompt for value using the following string:

○ Use the following value:

● Get the value from the following cell: =Report!$F$4

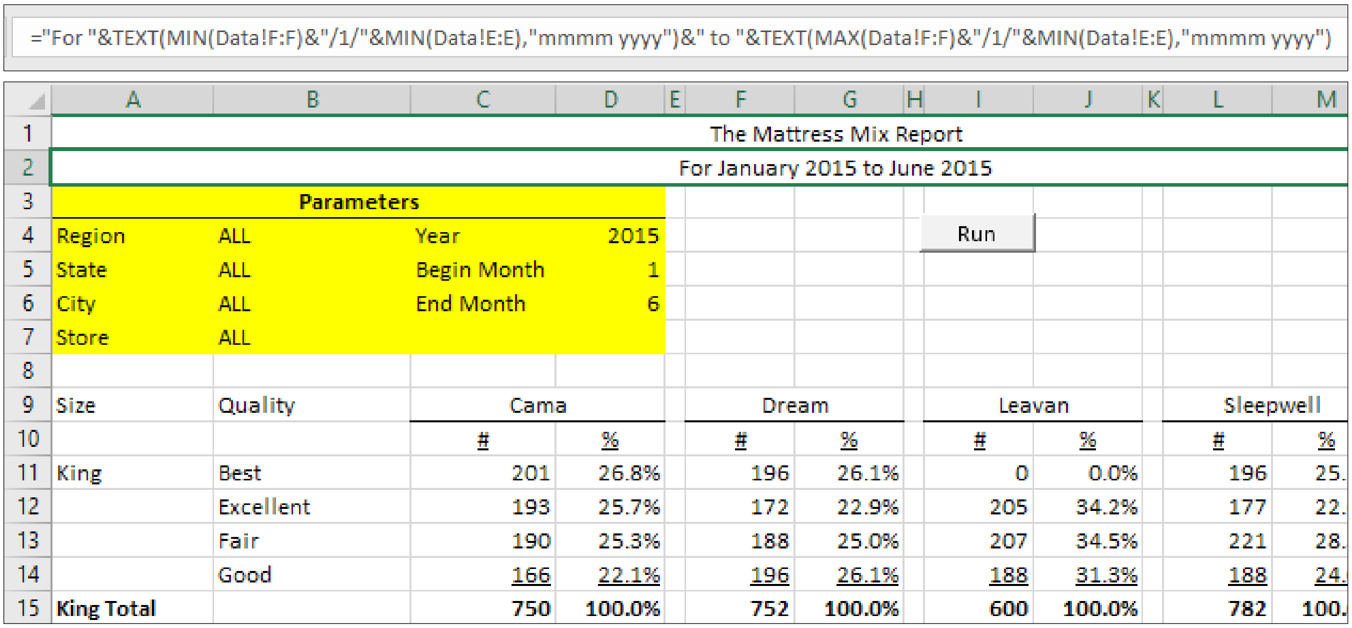
□ Refresh automatically when cell value changes

Figure 15.44

17. *Click* **OK** *and exit out of* **Microsoft Query Parameters** *and close the* **Workbook Connections** *dialog box.*
18. **Hide** *the contents of* **Cells F4 – G7** *(***Format Cells**, **Custom**, *Type=;;;)*

To tie a pretty bow around this project, you need to create a command button that refreshes the data. Additionally, the data range of the report should reflect the data in the Data tab, not the parameters chosen in the Parameters section. Think about it. If you change the date range in the Parameter section and don't click the Run button and print out the report, users will think that the data in the printed report reflects what is listed in the Parameters section. So the dates in the report title should reflect the dates in the data tab, not in the Parameter section.

19. **Create** *a macro that refreshes the data (just like you did in the previous project) and tie it to a Command* **Button**.
20. *Write a formula for the second line of the report title that reflects the date range in the* **Data** *tab.*



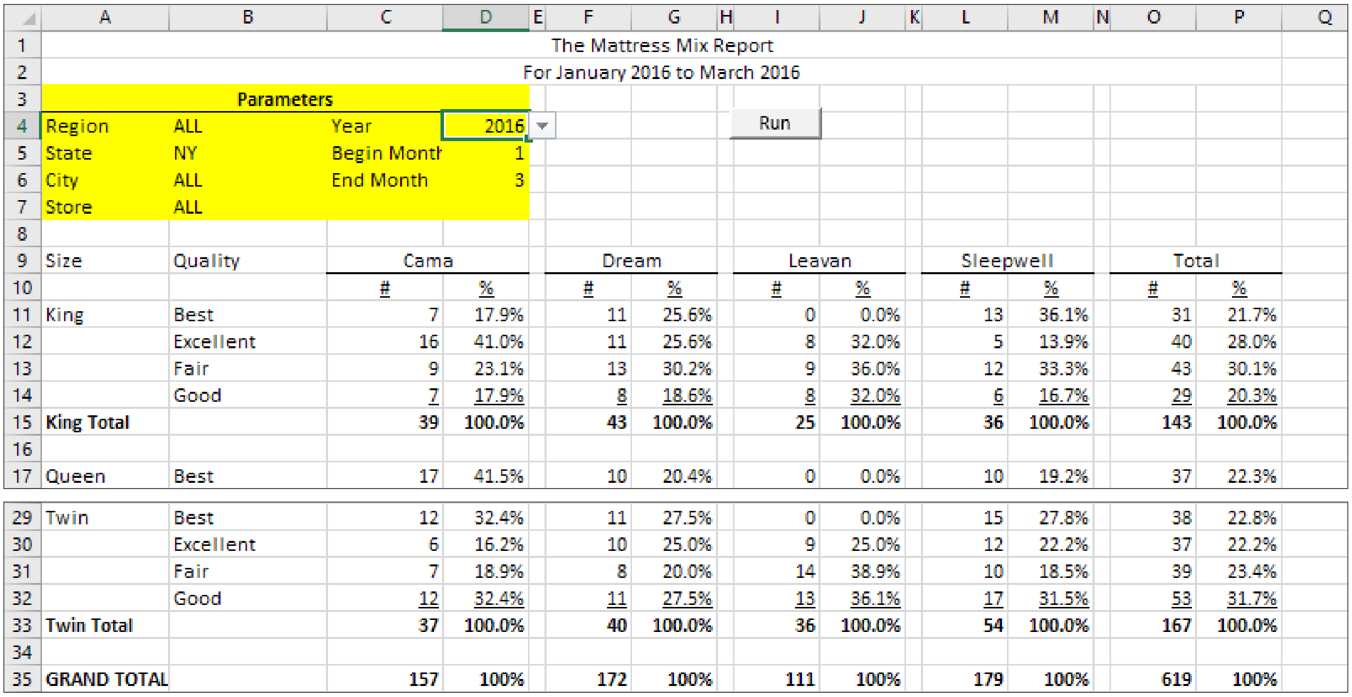
="For "&TEXT(MIN(Data!F:F)&"/1/"&MIN(Data!E:E),"mmmm yyyy")&" to "&TEXT(MAX(Data!F:F)&"/1/"&MIN(Data!E:E),"mmmm yyyy")

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	The Mattress Mix Report												
2	For January 2015 to June 2015												
3	Parameters												
4	Region	ALL	Year	2015					Run				
5	State	ALL	Begin Month	1									
6	City	ALL	End Month	6									
7	Store	ALL											
8													
9	Size	Quality	Cama			Dream			Leavan			Sleepwell	
10			#	%		#	%		#	%		#	%
11	King	Best	201	26.8%		196	26.1%		0	0.0%		196	25.
12		Excellent	193	25.7%		172	22.9%		205	34.2%		177	22.
13		Fair	190	25.3%		188	25.0%		207	34.5%		221	28.
14		Good	166	22.1%		196	26.1%		188	31.3%		188	24.
15	**King Total**		**750**	**100.0%**		**752**	**100.0%**		**600**	**100.0%**		**782**	**100.**

Figure 15.45

Let's run another report to make sure it is working right.

21. **Run** *a report for the state of* ***New York*** *for the* **first quarter** *of* ***2016****.*



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	The Mattress Mix Report																
2	For January 2016 to March 2016																
3	Parameters																
4	Region	ALL	Year	2016					Run								
5	State	NY	Begin Month	1													
6	City	ALL	End Month	3													
7	Store	ALL															
8																	
9	Size	Quality	Cama			Dream			Leavan			Sleepwell			Total		
10			#	%		#	%		#	%		#	%		#	%	
11	King	Best	7	17.9%		11	25.6%		0	0.0%		13	36.1%		31	21.7%	
12		Excellent	16	41.0%		11	25.6%		8	32.0%		5	13.9%		40	28.0%	
13		Fair	9	23.1%		13	30.2%		9	36.0%		12	33.3%		43	30.1%	
14		Good	7	17.9%		8	18.6%		8	32.0%		6	16.7%		29	20.3%	
15	**King Total**		**39**	**100.0%**		**43**	**100.0%**		**25**	**100.0%**		**36**	**100.0%**		**143**	**100.0%**	
16																	
17	Queen	Best	17	41.5%		10	20.4%		0	0.0%		10	19.2%		37	22.3%	
29	Twin	Best	12	32.4%		11	27.5%		0	0.0%		15	27.8%		38	22.8%	
30		Excellent	6	16.2%		10	25.0%		9	25.0%		12	22.2%		37	22.2%	
31		Fair	7	18.9%		8	20.0%		14	38.9%		10	18.5%		39	23.4%	
32		Good	12	32.4%		11	27.5%		13	36.1%		17	31.5%		53	31.7%	
33	**Twin Total**		**37**	**100.0%**		**40**	**100.0%**		**36**	**100.0%**		**54**	**100.0%**		**167**	**100.0%**	
34																	
35	**GRAND TOTAL**		**157**	**100%**		**172**	**100%**		**111**	**100%**		**179**	**100%**		**619**	**100%**	

Figure 15.46

YOU DID IT!!! Now you have a fully functioning reporting tool that ANYONE would be proud of.

22. **Save Mattress_Mix.xlsx** *as a* **macro-enabled** *file and close it.*

SQL Conclusion

Although Microsoft Query can open up worlds of functionality that you can use, it has its limitations. Let me end with a few closing remarks about Microsoft Query:

- If you can't edit the SQL code with the Query Design Grid, it will not let you pass parameters from an Excel spreadsheet.
- You can't pass spreadsheet variables into a PivotTable that is tied via Microsoft Query.

Over my years of working with Microsoft Query and writing SQL code, I've found MANY uses for it. I really like using a UNION query in Microsoft Query. Once, I assumed the responsibility to reconcile a particular general ledger account. I could get the detailed transactions from a subledger (whose data was copied into a SQL Server database) and the detailed transactions from the company's ERP system. It took the person who was previously reconciling the account quite a few hours (typically a day) to perform the reconciliation, and most of that time was used just to identify the differences. I was able to write a UNION query to import the activity from two separate tables and compare the balances in a PivotTable. To refresh the data, all I had to do was to click the Refresh Data button. It took me about an hour to write the query (along with all verifications it was pulling the right data), and I was able to save about five to seven man hours per month. Not bad for one hour's work! I'm sure you will have many similar opportunities. It just takes your creativity to figure it out.

> ***Review Questions****: It is now time to complete the hands-on Review Questions. Log on to www.ExcelCEO.com with your Email and Password, click on the* **Access 2016 and SQL Review Questions, Chapter 15, Section 2 of 2** *option and complete the review questions.*

Conclusion

You began this chapter by connecting to a SQL Server database, created a SQL statement in Microsoft Query's GUI screen and edited the SQL code to include parameters on the Excel spreadsheet. This "trick" is well hidden from the world of average users, but knowing how to do it is easily worth the cost of this course in itself. You then created an extremely functional report that incorporated your Excel, Access, and Microsoft Query knowledge all into one project. Yes, it was a long chapter, but now you have a skill set that is beyond 99% of accountants and financial professionals in the marketplace today. CONGRATULATIONS!!! The next step is to apply the training you have learned and the skills you have gained to projects that continue to stretch your imagination and creativity to improve your work and save time and resources for other pursuits.

Chapter Exam

You can now go to www.ExcelCEO.com, click on Sign In, log in and take the exam. Make sure that you take the exam on the same device on which you completed the practice files, as some of the questions on the exam may refer to some of the completed projects. Chapter exams are intended to be hands-on.

Access® 2016 and SQL

Complete Self-study Course

ExcelCEO

Chief Excel Officer

GLOSSARY

Term	**Definition**
#DIV/0! Error	An Excel error message returned when a numeric calculation is setup to divide by zero - a mathematical impossibility.
#N/A Error	An error message returned when the search value is not available as formatted, or not available at all.
Absolute Reference	A cell reference in a formula that does not change when copied to other cells. This is identified by having a $ before AND after the column letter, which acts as an anchor for both column and row. The $ anchors whichever cell reference part (column or row) before which it is placed.
Active Field	The highlighted or selected area of a spreadsheet, field, or record where content is in a state to be modified or edited.
Aggregate Function	A query tool that calculates or groups data that meets query- or user-generated criteria. In a SQL query, aggregate functions must be included in the GROUP BY statement.
Alias	An alternative name provided for a defined calculation or concatenated field which cannot be the same as an existing field within the table(s) or query/queries included in the Design View.
Align Left	Text formatting that locates cell contents toward the left-hand side of the cell.
Align Right	Text formatting that locates cell contents toward the right-hand side of the cell.
AutoFill	A text fill function, available through an icon that pops up after a tool like DataFill has been used, that allows the Excel user to define how data will be copied, with or without originating cell formatting, or using Flash Fill.
AutoFilter	An Excel filtering feature that allows you to select filtering options using a dialog box accessed from column filter drop-down arrows that allow you to select specific values using checkboxes and menus.
AutoNumber	An Access database field function that assigns a unique numeric value to a given record when it is created. Manual entries in an AutoNumber field are not allowed. If the record is deleted from the database, the given unique value will not be re-created in that table.
AutoSum	An Excel function on the Home tab used for quickly creating a sum for a contiguous group of numerically formatted cells. Average, Count, and other functions are also available in the AutoSum drop-down menu.
AVERAGE()	A statistical function for calculating and displaying the combined middle of a defined group of numbers in numeric formats.

AVERAGEIFS()	A multiple option criteria-based function for finding the mean of given arguments.
Blank workbook	An Excel workbook not containing any user-inputted data. This is also an option icon in the Templates section when opening Excel for users who want to start a new project or a quick calculation.
Bold	Text formatting that adds visual weight to the characters for the intent of standing out.
Borders	Formatting that applies line characteristics to the exterior of a cell, text label, text box, or other defined shape.
Bottom Align	A cell location format that causes cell content to be situated toward the bottom of that cell.
Brackets	Text characters that encase a table or query field within an alias or to represent a user-defined parameter to be inputted later when a query is used.
Calculated Field	A mathematical function that allows elements of a PivotTable or Access Text Box to be manipulated using formulas to develop new solutions not contained in the source data.
Cell	The intersection of a column and a row or between a field and a record row. A cell is a rectangular space — named first by column, then by row position in an Excel spreadsheet — where calculations or data can be inputted and analyzed.
Cell Style	A formatting option that allows the Excel user to create a custom, named format for quick access. Options include background color and font attributes.
Center	A cell location format that causes cell content to be situated toward the vertical middle of a defined cell.
Chapter Examination	An exam that is administered at the end of each chapter to test whether or not the participant has learned the material well enough from the given chapter. Each Review Questions checkpoint in the chapter must be completed prior to taking the exam. A passing grade of 70% or above must be obtained before continuing to the next chapter, and a maximum of two retakes are possible for claiming CPE credit for the specific chapter. Projects outlined and completed within the chapter will be referenced in the chapter exams, so having the projects accurate, completed, and available during the exam is vital to passing each exam.
Columns	Vertical groups of spreadsheet cells named with alphabetical letters above the spreadsheet. The Access equivalent is called a field.
Combo Box	Multi-functional box that can be tied to user-specified data or query and capable of passing these variables from a form to generate a report.
Command Button	An interactive overlay feature that can have interactive functions such as a simple or complex macro connected to carry out a defined action when clicked.

Conditional Formatting	Cell formatting applied based on pre-defined criteria, ranging from text formats to cell background colors.
Contextual Tab	A tab related to a specific function and represented above the Office Ribbon in Microsoft programs when the given tab is activated by a user of the Office program.
Copy	To record defined cell or worksheet content for placement in another location while leaving that same content in the original location.
COUNT() Function	A data function designed to return the number of cells containing a specified value.
COUNTIF()	A count function that allows the Excel user to select a range, and a specific criteria total based on a cell reference.
CPE	Continuing Professional Education. A course meeting criteria established by an authorized reporting board, or by NASBA, by which people maintain and expand their knowledge and skills for a particular profession to maintain professional accreditation. The trainee/student should be aware of requirements specific to their reporting board by which training will be evaluated including delivery method, credit quantity, and any applicable deadlines.
Custom Format	A user-defined object appearance that is displayed when certain criteria or inputs are recognized.
Custom Sort	Functionality in Excel that allows the user to organize data based on column, criteria, and order. Multiple levels are also available for more detailed sorting.
Cut	To record defined cell or worksheet content to the Clipboard for placement in another location, removing the content from the original location.
Database	A collection of related tables, queries, forms, reports, and/or macros which can allow for communication by query. A database is typically stored on a remote server or on a computer storage drive.
Data Consolidation	The process of combining worksheets from separate workbooks into a common workbook or file.
Data Labels	Descriptive or otherwise identifying information, usually for use in charts, forms, or reports to enhance the appearance of, section off, or draw attention to related or key details.
Data Source Name	A name given to the connection set up to a database from a server. The name is commonly used when creating a query to the database.
Data Validation	A method for requiring a user to select from pre-defined options in a drop-down menu, or from a chosen range. This is preferred when the choice affects formula performance.

DataFill

Related to AutoFill. A cell content feature shown as a bold black box on the bottom-right corner of an active cell, or group of cells, that allows for dragging formulas to adjacent cells, or through double-click on the AutoFill/DataFill box, copying the cell content formula or pattern to all adjacent cells in a specific column below.

Dates

A numeric text format that applies a calendar day format to a specified number from a base of one representing January 1, 1900.

Decrease Decimal

An icon on the Home tab Number group used for reducing the visible amount of numbers following the decimal.

Dialog Box

A labeled pop-up window on a computer screen that communicates information to the user and prompts them for a response. Dialog boxes typically allow for customization or interaction within the given program.

Dialog Expander

A small, downward facing arrow in a box letting the Excel user know more options are available in the tool group.

Expenses

A term representing outgoing payments, particularly in an income statement

External Data Sources

Data contained in a location outside of Excel or Access which can be linked to or imported for analysis.

Filter

A tool for narrowing and defining the search for data within a given range based on user-defined criteria.

Find and Replace

A tool that locates user-defined content within a selected range and replaces that matching content with content of the user's choosing into the active range.

Fixed Expenses

A financial outgo of capital that does not change based on product or service output, material costs, etc. These expenses remain constant, or sunk, regardless of ability to generate income.

Format Cells

A dialog box where you can apply number, text, and color formats to a cell, or group of cells within a workbook.

Form

A customizable interface that allows for indirect interaction between a user and linked database fields. Common features include labels, Combo boxes, command Buttons, text boxes, text labels, and/or hyperlinks. Data can be passed from a form into a database or data can be developed into a report based on variables passed from the form into a query.

Formula

A logic-based equation which evaluates specified arguments on values, dates, or text strings from referenced cells in an Excel worksheet, an Access database, or other logic-based application, and returns a value based on criteria provided.

Function Keys

Keys on a keyboard that allow the user to perform certain functions that allow the user to use the keyboard instead of a mouse to increase speed and maneuverability. Function keys are typically keys that begin with the letter "F" and are located along the top of the keyboard. Also called Action keys.

Graphics	Images that can be Inserted or created in a spreadsheet to customize the appearance of the work.
Handles	Small, opaque squares surrounding an active object that allow for adjustment in vertical or horizontal size, or a combination of the two. A circular handle can also be available for rotating some objects.
Hard-coded	Non-dynamic data inputted directly into a cell, text box, or label without formula. Hard-coded data does not update without manual adjustment.
Hide Detail	A box with a "-" sign inside used for covering low-level details, or removing them from view, often leaving summary levels.
Hierarchical Structure	A parent/child structure in a database that establishes relationships between given data in a rollup or structured layout.
HTML	Stands for Hyper Text Markup Language. HTML is a language, not particularly programming, that identifies how text or images should be displayed by a web, or other HTML-compatible browser.
Hyperlink	An interactive connection, through text or image, that allows a user to move from one virtual location to another through mouse click. Hyperlinks can be used to move between spreadsheets or cells within an Excel workbook, to internet pages, or to email client windows.
Icon	A pictorial representation of an object. When attached to a function, a clicked the icon performs a predesigned task.
Images	Another term for graphics, images are visible representations of data, scenery, etc., and can be inputted into Excel workbooks to enhance interaction, display, or presentation.
Increase Decimal	A number icon that expands the total count of numbers visible to the right of a decimal, up to the total amount available in the cell, or group of cells.
Insert Function	A button to the left of the formula bar identified with "*fx*", which can open a dialog box for explaining functions step-by-step, and inserting the selected formula into the selected cell.
Join	A user-defined relationship between database fields in a SQL or Access query. One-to-many, one-to-one, and many-to-many joins are possible.
Keyboard Shortcut	Defined keystroke combinations in a program that allow a user quick access to specific formulas, functions, menus, or macros associated with the combination. (Ex: [Ctrl]+c is the equivalent keyboard shortcut for clicking the Copy icon from the Clipboard group in the Excel Home tab, and selecting the cell contents to copy to the clipboard memory for pasting in another location.)
LEN()	A text function that returns the length in total characters in a defined cell, including spaces before, after, and between character groups (words, numbers, etc.)

LOOKUP()

A search function with two arguments: the value to be found, and the range for return value. The range will always be two columns, and the return value will always be the value in the second column to the right of the lookup value, or the next lowest value available.

LOWER()

A text function that returns an all lower-case equivalent of the text in a specified cell.

LTRIM()

A database function that eliminates spaces from the left side of a cell and returns the remaining value and and spaces that may exist to the right of the cell value.

Macro

A mini-program that executes a programmed task, which can be linked to buttons or keystrokes. Caution should be exercised in knowing the source and intended function of a macro before enabling it for use, as macros can perform functions extending beyond the program written in.

Macro Recorder

A small box icon (also available through a dialog box) that can be used to define a macro's function by executing it visually on the screen. Once recording has begun, a Stop box is available on the lower-left of the Excel window for ending step recording to be associated with the macro.

MATCH() Function

A text function that returns the relative row position in a table or array for a defined criteria.

MAX()

A statistical function that returns the greatest numeric value in a given range.

MEDIAN()

A statistical function that returns the value in the middle of a given range where half the values are more, and half are less. Not to be confused with the AVERAGE() function.

MID()

A text function that returns characters from a cell based on user-defined beginning point, and maximum count.

Middle Align

A text alignment tool that vertically locates cell contents in the middle of a given cell.

MIN()

A statistical function that returns the numerically least value in a specified range or array.

MODE()

A statistical function that returns the most commonly occurring numeric data point in a selected range.

MONTH()

A text function that returns the month number from a specified cell with the number content formatted as a any type of number, based on a beginning point of one being 01 JAN 1900.

Named Range

A table or database option for identifying a workbook range by a user-specified label in addition to column/row names for quick reference in formula-writing or locating. Names for ranges can be viewed and written in the Name Box to the left of the Insert Function icon and the Formula Bar.

Nesting Functions	Text functions that operate based on logical arguments when evaluating data for best solution to return. Nested functions are evaluated left to right and return the first value that evaluates as TRUE, or else a specified FALSE argument value, or error, displays.
Non-contiguous Ranges	Data ranges with column or row breaks between non-blank cells. When intentional, these provide useful barriers for separating non-related data. If breaks between related data are present, DataFill formula copying will not populate beyond the data breaks without manual intervention.
NOW()	A time function that returns the computer-provided time in the given cell. This function updates upon refresh.
Operator	Sometimes referred to as a Comparison Operator. Characters or a set of characters used in conditional statements to test arguments. The six operators are Equals (=), Less than (<), Greater than (>), Less than or equal to (<=), Greater than or equal to (>=), Not equal to (<>).
Orientation	A Page Layout tool divided between Portrait and Landscape used for displaying information as it would print on paper, etc.
Page Break	A Page Layout detail that defines where the end of a page would be when printed, based on margin values compared to cell dimensions. As a selected option, a spreadsheet will display Page Breaks with virtual perforations seen as dashed lines.
Parameter	Criteria within a query.
Parent/child Relationship	A relational database term for identifying how data is connected to other data between ranges.
Password	A case-sensitive login credential for accessing the student training portal at ExcelCEO.com. A computer-generated password is initially assigned for first-time access, which can be changed once logged into the specific profile using the access boxes to the upper-right of ExcelCEO.com. After periods of inactivity, or for other reasons, a registered ExcelCEO student can have their password sent to the email address associated with the profile using the [Forgot Your Password?] link below the login area.
Paste	A virtual function that takes cut or copied data from the Clipboard and locates it in a specified cell or cell range.
PivotTable	An advanced Excel feature for in-depth analyses and organization of data that does not alter the original data. Data can be displayed using filters, slicers, and modified column and row information identified in the original source data, or through user-created Calculated Fields. Similar to a query, PivotTables do not alter the base data, but a PivotTable retains the specified data.
PivotTable Fields	Column headers from the source data, or Calculated Fields, that can be inputted in the PivotTable organization grid through checking boxes, or drag-and-drop ability. Additional data ranges can be utilized through the Excel Data Model.

Primary Key	A user-specified relational database feature set for a field containing unique identifiers or values that can be linked to other tables or queried records.
Print Preview	A visual representation of the way data in a specified range would display on a printed page.
Progress Report	An emailed reported at a user-specified interval that tracks online progress through the course(s) for which the ExcelCEO student is registered.
PROPER()	A text function that takes text from a defined cell, returning the text with the first letter capitalized, and the remaining letters in lower-case format.
Property Sheet	A toolbox in the Design View of Access that contains all the display, criteria, format, and interaction features for a given Access object including: Combo, Text, or Label box; query field, form, or report feature.
Protect	An option within Excel for locking certain aspects of a cell or range against changes to formulas or data without inputting the assigned password.
Query	A virtual table populated with native fields, records, and aliased fields specified or defined by the user. Query types include: Select, Append, Update, Delete, and Drop Table.
Quick Access Toolbar	A customizable group of icons that enables the Excel user to quickly access commonly used features. Default icons include Save, Undo, and Redo
Quick Analysis	An Excel feature available through an icon that appears after a range is selected for seeing visual representations of the data from a group of tool options.
Record	A horizontal, linked set of data within an Access table whether in a single field or a combination of fields within the table. Related fields can be linked to like fields in other tables or queries through temporary or enforced relationships.
Record Selector	An Access feature that allows the user to navigate through existing database values. This feature can be enabled or disabled in a form.
Redo	A right-facing arrow icon with a curved shaft that is designed to reapply the last step that was removed.
Relationship	A connection established between related fields in separate tables or queries to simplify and compact a database. Relationships can be set temporarily within the query Design View or enforced through Relationships View.
Relative Reference	A column/row name in a formula without any anchors to specific cells, columns, or rows (anchors being identified as "$" before the column letter, or row number).

Report	Defined records populated through a combination of table, query, and/or form inputs that match given criteria within the query-specified database. Common features include Header, Footer, and Detail sections with custom sections being optional.
Revenue	A measurement of received payments that do not take into account any related fixed or variable expenses applied to the unit sold.
Review Questions	A hands-on learning component of the ExcelCEO student training experience in the online profile at ExcelCEO.com (for students registered for a training course) where the student is provided questions to test their knowledge of the program taught directly up to that point, or peripheral knowledge that can be gained at the time the question is given, in order to further enhance learning opportunities. Each section of Review Questions is delivered sequentially, not graded, but must be completed in order to move forward in the online portion of ExcelCEO training, and each section of Review Questions must be completed prior to the Main Menu text link updating to display that chapter's exam link. Each option selection provides an answer and explanation.
RTRIM()	A function that eliminates spaces from the right side of a record cell and returns the remaining value from the cell, including any spaces that may exist to the left of the cell value.
ROUND() Function	A number function that takes numeric values in cells, and provides the closest whole number to the specified space, filling in the rest with zeroes. The rounding space number is defined from the decimal placement, with positive numbers being to the right of the decimal, and negative numbers applying to number slots to the left of the decimal.
Rows	Horizontal groups of spreadsheet cells named numerically in Ascending order. The Access equivalent is known as a record.
Screentip	A temporary window seen when hovering over an object in the workbook. Screentips include shortcut key stroke combinations, and brief descriptions of object capabilities.
SEARCH()	An Excel text function that returns the numeric starting place in a defined cell where the user-defined text is located, from the given beginning point.
Show Detail	A box with a "+" sign inside used for showing (or expanding) low-level details, allowing the Excel user to see more than summary levels for the Data.
Show Table	An Access query feature and related dialog box that allows the user to select queries and/or tables with which to structure a query. A query can also be structured using SQL View.
Spreadsheet	A grid of columns and rows that can be used for organizing, calculating, sorting, summarizing, etc. In Microsoft Excel, Columns are identified alphabetically, and rows are identified numerically.

SQL	Acronym standing for Structured Query Language. SQL is a language consisting of Reserved Words and user inputs designed to communicate with databases and return values to which the user has authorized access.
SQL View	A section of Access wherein a user can use a combination of Reserved Words, database table, alias, aggregate function, and/or query fields to develop a query.
Status Bar	A customizable feature to the bottom of the Excel window that shows details such as the AutoSum of selected cells, average, and count. To access the Status Bar options list, right-click the colored bar on the Excel window.
Subtotal	A database function that can apply summary data labels and values for specific groups of data.
SUM()	A number function that provides the mathematical total for the cell range provided using addition or subtraction.
SUMIF()	A conditional number function that returns a mathematical total for cells in a range that meet a defined criteria.
SUMIFS()	A conditional number function that returns a mathematical total for cells in a range that meet a defined criteria or condition.
Table	A collection of related fields and records stored within a database.
Tags	HTML markers identified by web browsers to display the enclosed text or object(s) in a pre-defined way
Text Box	Highly flexible size and format boxes capable of displaying user-specified text, single records based specified formulas or criteria, or entire recordsets matching a given criteria within a report as provided through a query or field.
Text Functions	A group of Excel tools designed to interact with specified data using logical arguments and conditions to return the first true value available.
Text to Columns	An external database tool within Excel for inputting data from other sources and formats based on user-identifed separators, or delimiters, that determine column, row, and cell breaks. This method is not formula-based.
Title Bar	The area at the top-center of the workbook that identifies the file name and program. Unsaved File names will begin with “Book”, such as Book1.
TODAY()	A text function that identifies the calendar date when written, or updated, and is absent of a defined time, as provided with the NOW() function.
Top Align	A cell text feature that aligns text vertically at the top of the selected cell.
TRIM()	A Excel text function designed to remove seen and unseen spaces surrounding cell data, which can otherwise affect formula performance and/or displayed appearance.

Underline	A text feature that places a thin, solid line below a specified cell or text — used for separation, isolation, or attention purposes.
Underscore	A single-character displayed as a line at the bottom of the text field, often as the visual representation of a space, or to improve functionality for programs that do not easily recognize unmarked spaces.
Undo	A left-facing arrow icon with a curved shaft that is designed to remove the most recent function or input application. The default location for this icon is in the Quick Access Toolbar
Union	A SQL Reserved Word that connects related queries where the format and number of fields are matched.
UPPER()	A text function that returns the contents of a chosen text cell in all capital format.
User ID	An identifying credential for logging into a website. ExcelCEO students provide email addresses for User IDs, which can be changed in the student profile Main Menu, as needed. [Forgot Your Password?] requests are setup to send the information to that address as well.
Variable Expense	A financial outgo of capital that changes based on product or service output, material or labor costs, etc.
VBA	Acronym standing for Visual Basic for Applications. An editable language used for executing macros.
Validation Rules	User-defined criteria for inputting data in a cell that requires specific values for dependent Formulas to perform properly.
VALUE()	A text function that returns the static, displayed characters from a specified cell without formula dynamics applied.
VLOOKUP()	A vertical lookup function that searches for a defined number, text string, value, or formula result in a specified data range, and returns a related value from a specified column number within that range. The value returned can be user-defined as exact or relative.
View	The displayed results of a query.
Wildcard Character	Substitute character that can be used to assist in a search or query to expand the possible results identified. A wildcard character can represent a single character or a string of characters.
YEAR()	A date function that returns the year number from a numerically-formatted target cell, based on the number 1 representing 01 JAN 1900.

Access® 2016 and SQL

Complete Self-study Course

ExcelCEO

Chief Excel Officer

INDEX