ExcelCEO
Chief Excel Officer

THE MASTER GUIDE TO FINANCIAL REPORTING AND ANALYSIS

"Bridging the Gap between Accounting and I.T.", Access 2010 and SQL

Course Manual **www.ExcelCEO.com**

ExcelCEO

Chief Excel Officer

Access 2010 and SQL

Complete Self-study Course

Table of Contents

ExcelCEO

Chief Excel Officer

Access 2010 and SQL

Complete Self-study Course

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 <u>every word</u> and complete <u>every exercise</u> 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 on the market. 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 operates in 29 stores along the East Coast. Nitey-Nite purchases mattresses and pillows from four vendors and ships them to the 29 stores.

Please note that the financial data contained herein is purely fictitious and does not resemble the activity in any way of any particular 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 contained herein reflect standard double-entry accounting methodology.

I sincerely hope that you will enjoy this course. If you see any corrections that should be made, or if you would like to send me any feedback, please email us at Customer.Service@ExcelCEO.com. Thank you.

[illegible]

Jim Cline

Logging In

When you purchased this course, you received a password. You will use your email address and that password to download the practice files, take tests, and gain general access to the ExcelCEO system. Please write down that password in the inside cover of this manual or store in some other secure place so you can refer back to it when needed. To log on to the system, go to www.ExcelCEO.com and input your email address and password in the upper right corner of the page. If you forget your password, you can go to www.ExcelCEO.com and click on the Forgot Your Password link and your password will be emailed to you.

Practice Files

To work the many examples illustrated in this course, you need to download the Practice Files from the **ExcelCEO** website. To download the practice files, log in to the **ExcelCEO** website with your email address and password, and click on the **Downloads** link on the Main Menu. Follow the instructions in the dialog boxes to properly download the practice files on to your computer. I recommend you download the files exactly as the instructions indicate.

Review Questions

As you work through the course manual, you will periodically be instructed to sign on to the ExcelCEO website and complete the Review Questions. Review Questions are questions that are formatted in the same way as the actual examination questions, but are provided in the material for review and further instructional purposes. Review Questions are not graded, even though the program will indicate whether or not the chosen answer is correct or incorrect. If you choose a wrong answer, a message will appear that tells you why the chosen answer is wrong, and the program will allow you to choose another answer. You must choose the correct answer before continuing on to the next question. You should log on to the website to review the Review Questions when you see this paragraph in the manual:

> ***Review Questions***: *It is now time to complete the Review Questions. Log on to www.ExcelCEO.com with your Email and Password, click on the **Access 2010 Review Questions Chapter 1, Section 1 of 2** option 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 from two to four sections of review questions. Note that some of the review questions may cover material that is not specifically presented in the course manual. As such, you may benefit from intentionally choosing a wrong answer to see the explanation of why that choice was wrong. This is available ONLY on the review questions.

Chapter Examinations

There are 15 chapters in the Access 2010 course. At the end of 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 ***Take a Test (Access 2010)*** link which will navigate you to the appropriate test. All tests are administered sequentially after you successfully pass the previous 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 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.

After obtaining a passing grade, you will be able to print out a Certificate of Completion as proof that you've read the material, worked the examples, completed the review questions, and passed the exam. You must score a 70% or above on each exam to pass. Once you pass the last test you will become an ExcelCEO Access 2010 Master.

> ***Note****: If you are a CPA and are taking this course for CPE credits, AICPA rules state that you are allowed two retakes (meaning you can fail the test up to three times). You will not receive CPE credit for the given chapter, if you need to retake the test more than two times.*

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. Here is an example:

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

To further assist you, I have included hundreds of screen shots and pictures of the icons used in the examples, such as this Open icon: The screenshots that accompany this manual were taken using a Windows 7 operating system.

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 2010.
- 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.

What's New in Access 2010?

If you are converting from Access 2003 to Access 2010, 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 2007 to 2010, you won't see many changes. The release of Office 2010 was geared more towards Excel and PowerPoint than it was to Access. 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 2010.
- 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 XML, HTML, PDF, and dBase files.

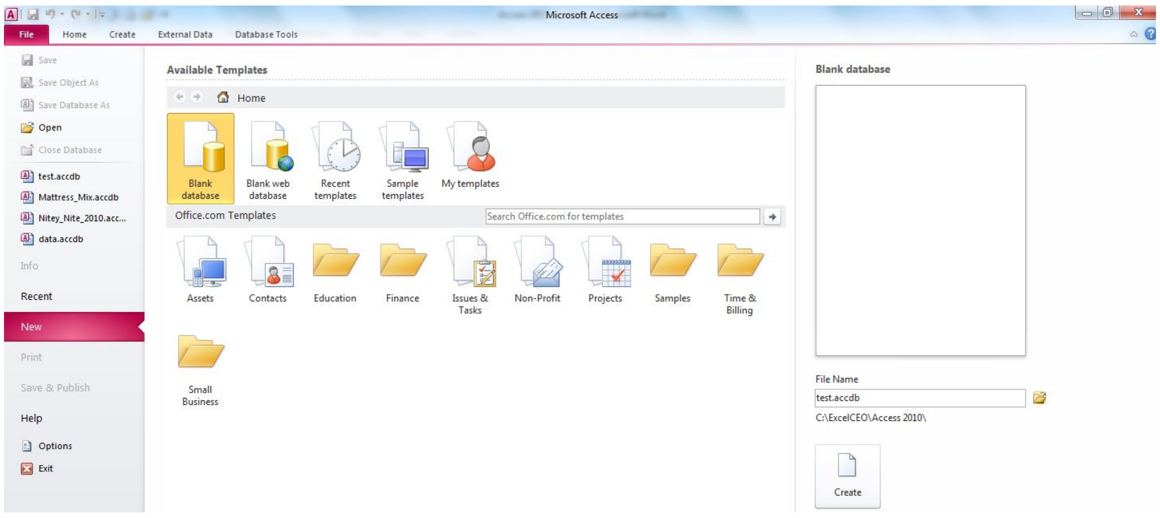
- <u>Enhanced Security</u>: Adding password protection to a database automatically encrypts it when closed, and decrypts it when opened.

Access 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 2010 database is stored as an **.accdb** file. Access databases before Access 2007 were stored with the **.mdb** extension.

In this course, I will use Access 2010, 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 2010 software for this course as Access 2010 has a different interface than any of its predecessors. Let's open Access.

1. *Open* **Access** *(from the* **Start** *menu, choose* **All Programs**, **Microsoft Office**, **Microsoft Access 2010***).*



test.accdb
C:\ExcelCEO\Access 2010\

Figure 1.1

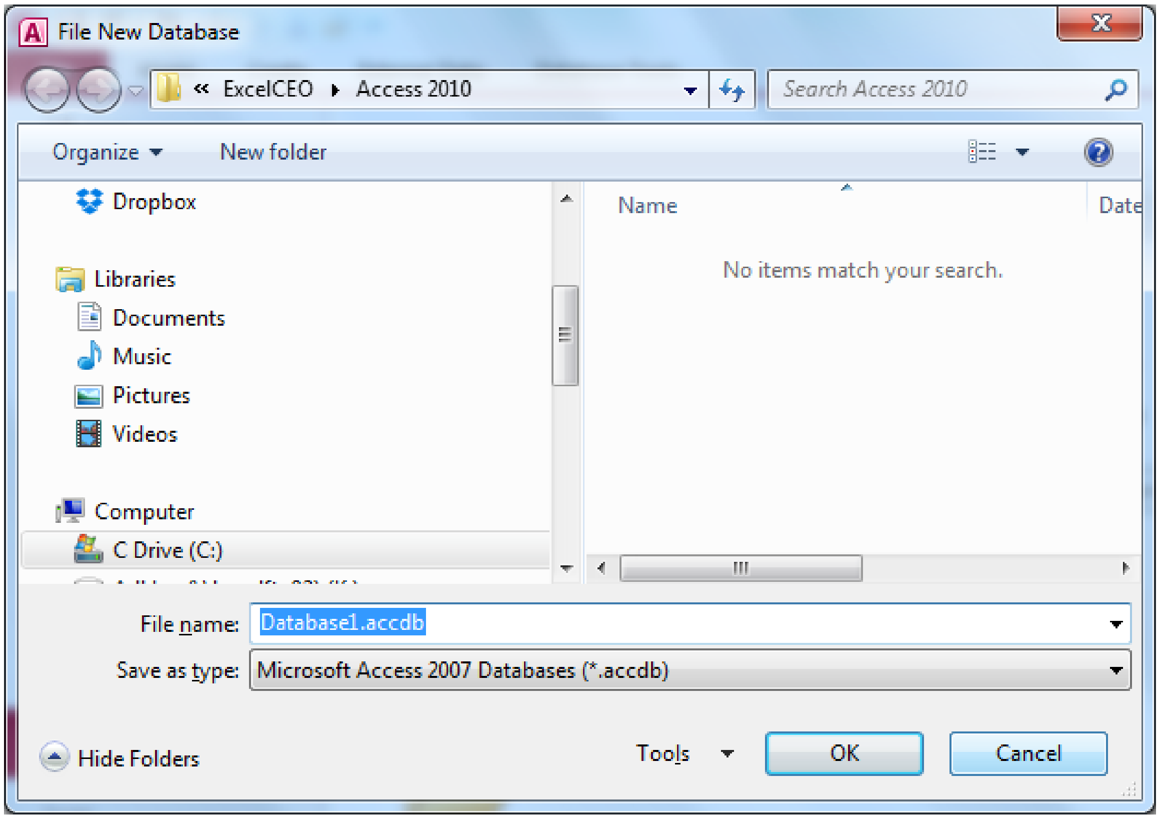
When you first open Access 2010 without opening a database, Access displays the Backstage view, as displayed in Figure 1.1. You can display the Backstage view at any time by simply clicking the File tab along the top of the ribbon. It is in the Backstage view where you can create a new database, use 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* **Browse** *icon* 📂 *to the right of* **File Name** *box on the right side of the* **Backstage** *view.*



No items match your search.

File name: Database1.accdb

Save as type: Microsoft Access 2007 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 2010** *folder.*
4. *Replace* **Database1** *(or the name of the default database that appears in the text box) with* **test.accdb**.
5. *Click* **OK**.
6. *In* **Backstage** *view, make sure the path below the* **File Name** *box reads* **C:\ExcelCEO\Access 2010** *(if not, repeat steps 2 – 5), and click the* **Create** *icon.*
7. *Click on the* **Enable Content** *command button if it appears (we'll discuss that in a minute).*

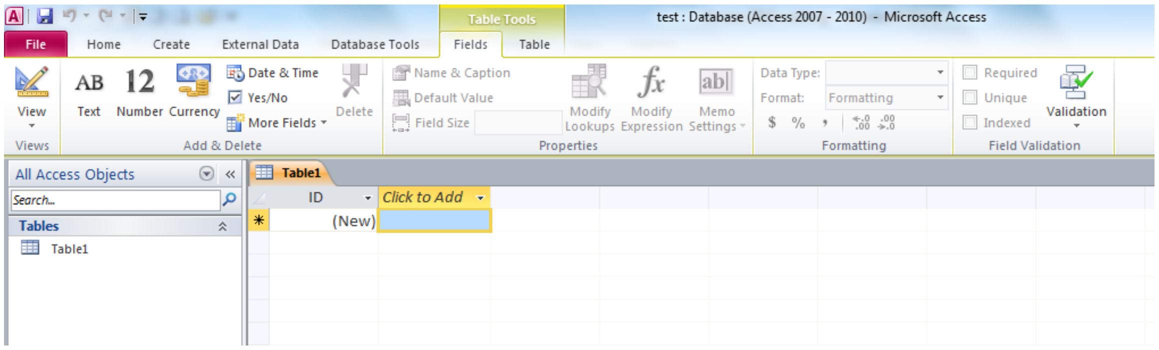


Figure 1.3

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 (Access 2007 - 2010) – Microsoft Access". 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 or Access 2010 can generally be used interchangeably. Note that you can open an Access 2003 database with Access 2007 or 2010 software, but you cannot open an Access 2007 or 2010 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):

Table:	tbl
Query:	qry
Form:	frm
Report:	rpt

You will see how I suggest to do this 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* **Access 2010** *and the* **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** *icon* *, navigate to* **C:\ExcelCEO\Access 2010** *and open the* **Nitey_Nite_2010.accdb** *file.*

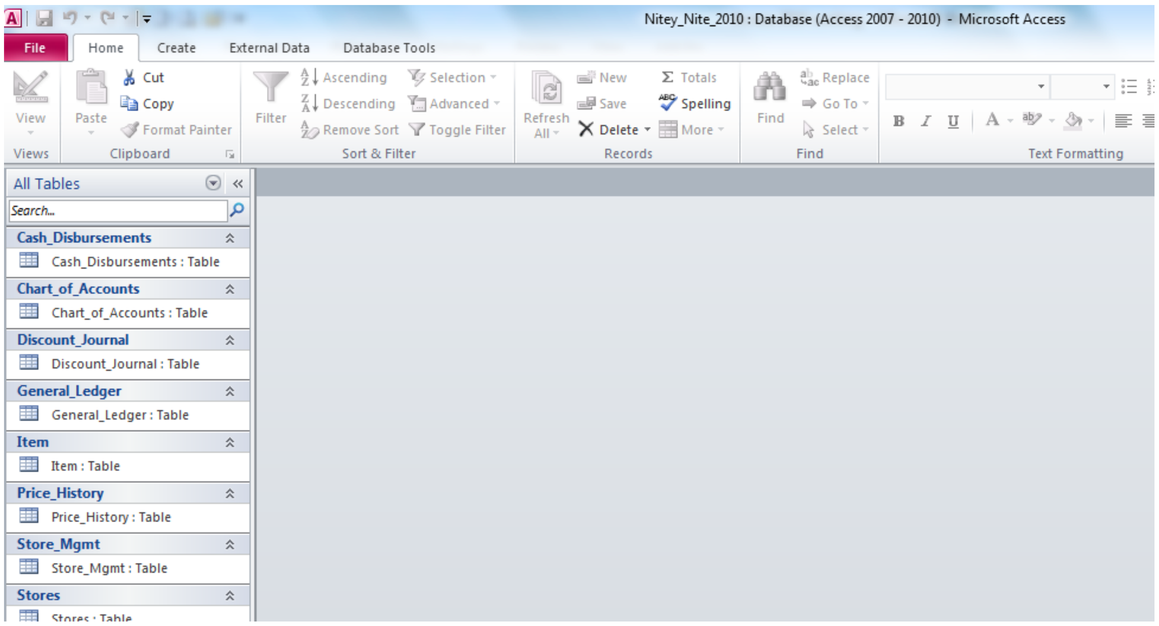


Figure 1.4

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, 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.

One issue in this database is that the objects don't appear to be very organized. I really like to see the objects as they appeared by default in earlier versions of Access (where they are organized by type of object), so let's change the way they appear in this pane.

4. *Click on the drop-down menu to the right of* **All Access Objects** *and choose* **Object Type** *(you can also right-click on* **All Access Objects**, *point to* **Category**, *and choose* **Object Type**).

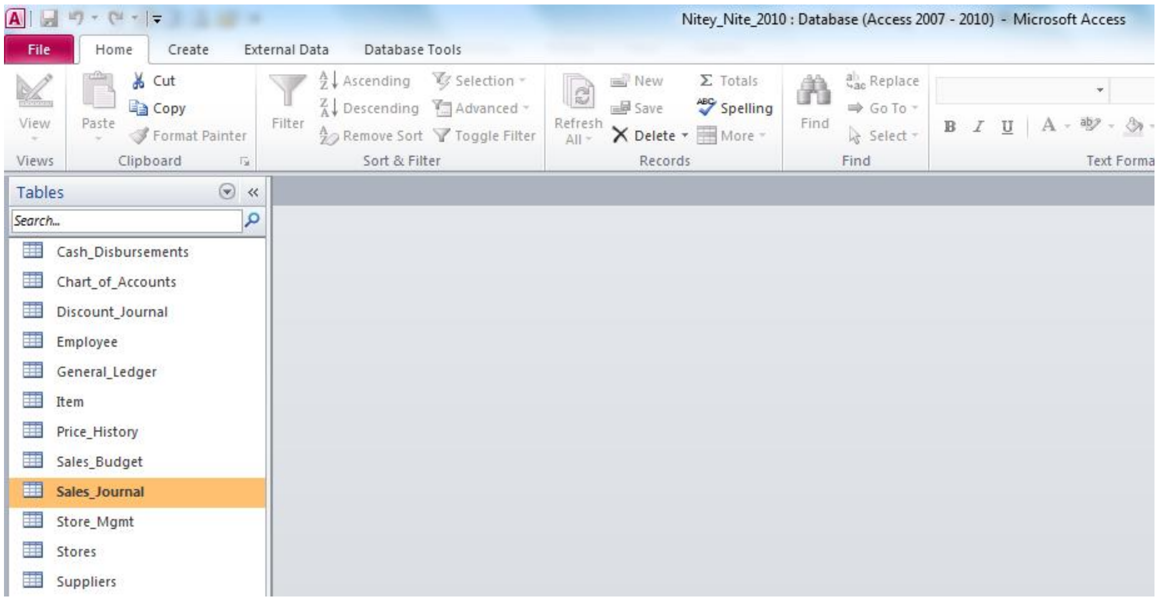


Figure 1.5

The objects in the database are now re-organized to appear in the Tables and Forms sections. These are the only sections in this database, as there are currently no objects in the other sections (Queries, Reports, and Modules).

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

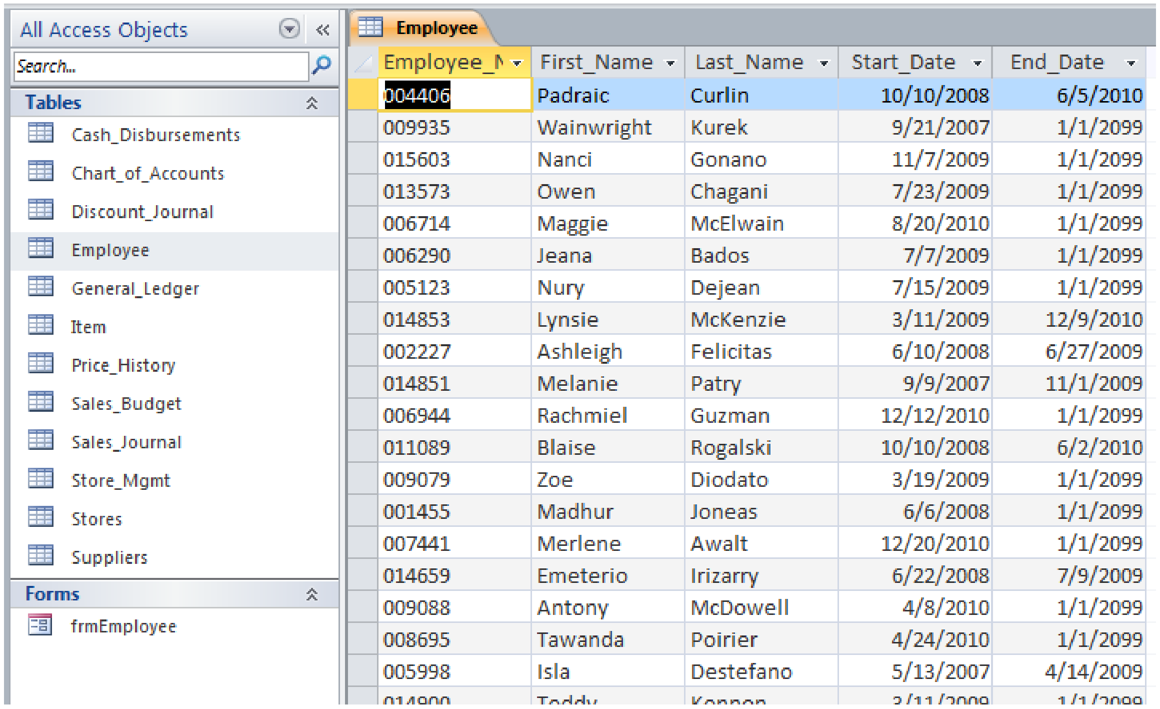
- Cash_Disbursements, a journal of various payments.
- Chart_of_Accounts, the company's account rollup structure.

- 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 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.

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



Employee_N	First_Name	Last_Name	Start_Date	End_Date
004406	Padraic	Curlin	10/10/2008	6/5/2010
009935	Wainwright	Kurek	9/21/2007	1/1/2099
015603	Nanci	Gonano	11/7/2009	1/1/2099
013573	Owen	Chagani	7/23/2009	1/1/2099
006714	Maggie	McElwain	8/20/2010	1/1/2099
006290	Jeana	Bados	7/7/2009	1/1/2099
005123	Nury	Dejean	7/15/2009	1/1/2099
014853	Lynsie	McKenzie	3/11/2009	12/9/2010
002227	Ashleigh	Felicitas	6/10/2008	6/27/2009
014851	Melanie	Patry	9/9/2007	11/1/2009
006944	Rachmiel	Guzman	12/12/2010	1/1/2099
011089	Blaise	Rogalski	10/10/2008	6/2/2010
009079	Zoe	Diodato	3/19/2009	1/1/2099
001455	Madhur	Joneas	6/6/2008	1/1/2099
007441	Merlene	Awalt	12/20/2010	1/1/2099
014659	Emeterio	Irizarry	6/22/2008	7/9/2009
009088	Antony	McDowell	4/8/2010	1/1/2099
008695	Tawanda	Poirier	4/24/2010	1/1/2099
005998	Isla	Destefano	5/13/2007	4/14/2009

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, rows, or columns with your mouse, or by using the up arrow, down arrow, Page Up, Page Down, 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.

6. *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.

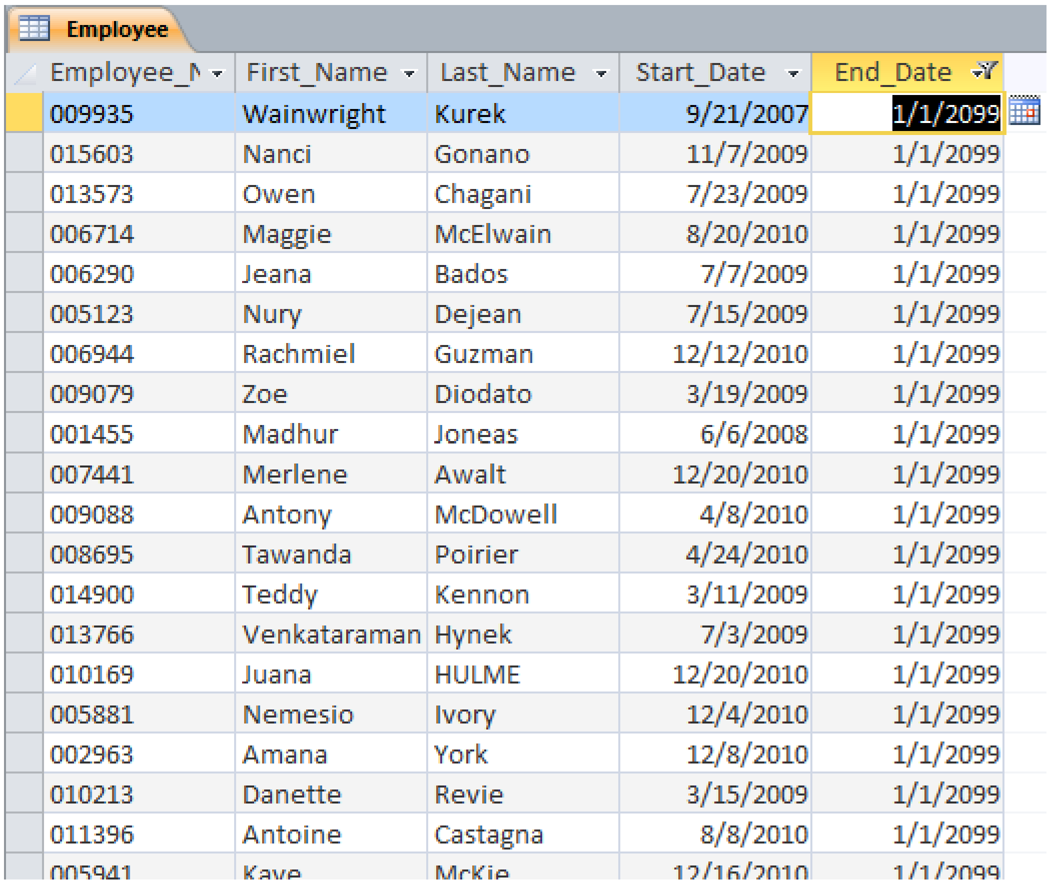
The **First_Name** and **Last_Name** fields should be self-explanatory. These are text fields and are the first and last names of the employee. The **Start_Date** and **End_Date** fields should also be easy to understand. These fields are formatted as **dates**, which is more closely associated to a 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. 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.

7. *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

Employee_N	First_Name	Last_Name	Start_Date	End_Date
009935	Wainwright	Kurek	9/21/2007	1/1/2099
015603	Nanci	Gonano	11/7/2009	1/1/2099
013573	Owen	Chagani	7/23/2009	1/1/2099
006714	Maggie	McElwain	8/20/2010	1/1/2099
006290	Jeana	Bados	7/7/2009	1/1/2099
005123	Nury	Dejean	7/15/2009	1/1/2099
006944	Rachmiel	Guzman	12/12/2010	1/1/2099
009079	Zoe	Diodato	3/19/2009	1/1/2099
001455	Madhur	Joneas	6/6/2008	1/1/2099
007441	Merlene	Awalt	12/20/2010	1/1/2099
009088	Antony	McDowell	4/8/2010	1/1/2099
008695	Tawanda	Poirier	4/24/2010	1/1/2099
014900	Teddy	Kennon	3/11/2009	1/1/2099
013766	Venkataraman	Hynek	7/3/2009	1/1/2099
010169	Juana	HULME	12/20/2010	1/1/2099
005881	Nemesio	Ivory	12/4/2010	1/1/2099
002963	Amana	York	12/8/2010	1/1/2099
010213	Danette	Revie	3/15/2009	1/1/2099
011396	Antoine	Castagna	8/8/2010	1/1/2099
005941	Kaye	McKie	12/16/2010	1/1/2099

Figure 1.7

Notice the Calendar icon, ▦, that appears to the right of the selected record. This is a new feature in Access 2007 and 2010 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 4/31/2010 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.

8. *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 12/1/2010 and 12/15/2010. 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.

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

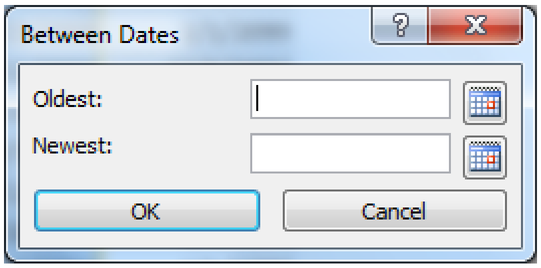
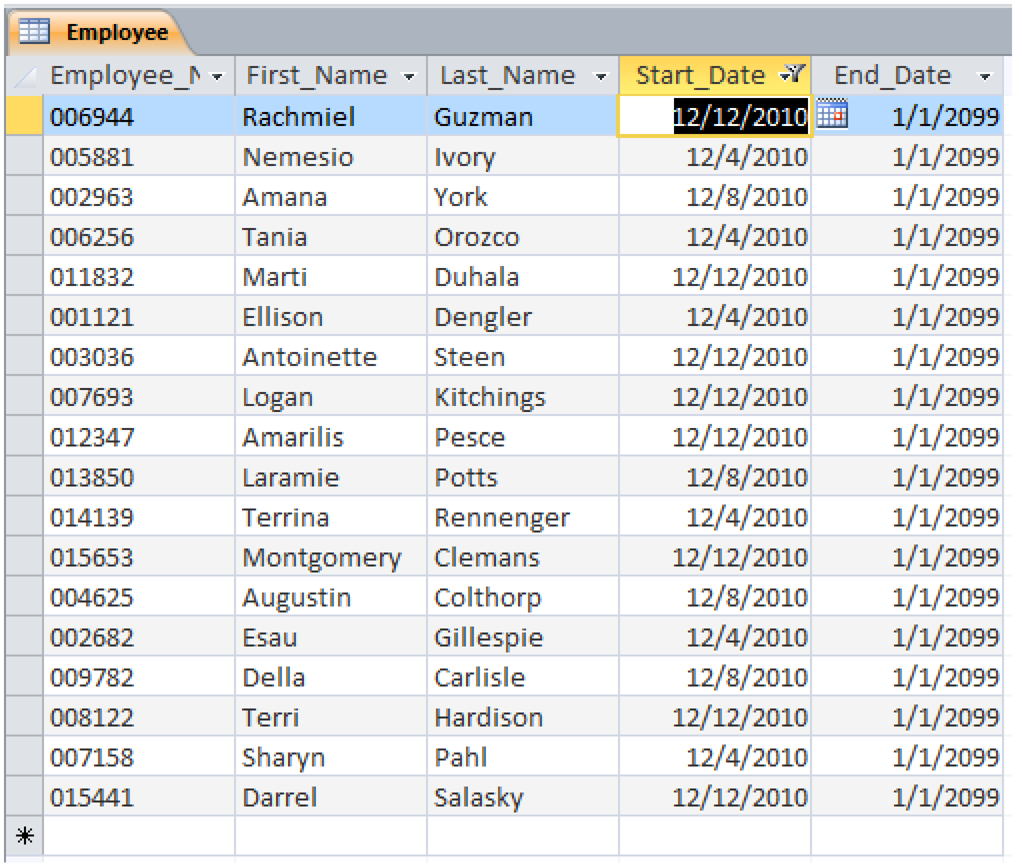


Figure 1.8

10. *In the* **Oldest** *box, type* **12/1/2010**, *type* **12/15/2010** *in the* **Newest** *box, and click* **OK**.



Employee_N	First_Name	Last_Name	Start_Date	End_Date
006944	Rachmiel	Guzman	12/12/2010	1/1/2099
005881	Nemesio	Ivory	12/4/2010	1/1/2099
002963	Amana	York	12/8/2010	1/1/2099
006256	Tania	Orozco	12/4/2010	1/1/2099
011832	Marti	Duhala	12/12/2010	1/1/2099
001121	Ellison	Dengler	12/4/2010	1/1/2099
003036	Antoinette	Steen	12/12/2010	1/1/2099
007693	Logan	Kitchings	12/12/2010	1/1/2099
012347	Amarilis	Pesce	12/12/2010	1/1/2099
013850	Laramie	Potts	12/8/2010	1/1/2099
014139	Terrina	Rennenger	12/4/2010	1/1/2099
015653	Montgomery	Clemans	12/12/2010	1/1/2099
004625	Augustin	Colthorp	12/8/2010	1/1/2099
002682	Esau	Gillespie	12/4/2010	1/1/2099
009782	Della	Carlisle	12/8/2010	1/1/2099
008122	Terri	Hardison	12/12/2010	1/1/2099
007158	Sharyn	Pahl	12/4/2010	1/1/2099
015441	Darrel	Salasky	12/12/2010	1/1/2099

Figure 1.9

Now you have a list of all the employees that started between 12/1/2010 and 12/15/2010. 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:

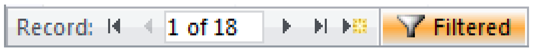


Figure 1.10

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 Page Up and 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 and Password, click on the **Access 2010 Review Questions Chapter 1, Section 1 of 2** option and complete the review questions.*

The Office Ribbon, Groups, and Icons

To use most of the functionality that Access 2010 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:

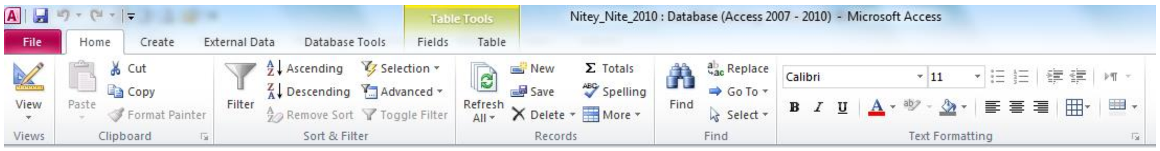


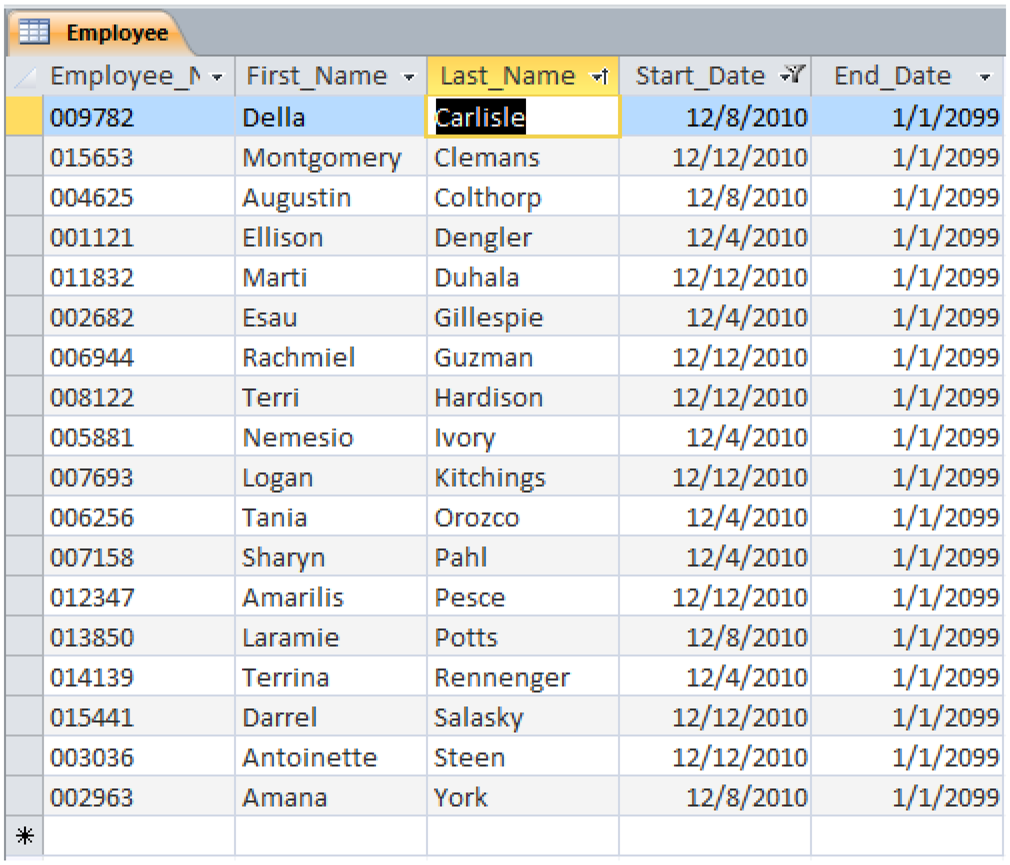
Figure 1.11

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 (as 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.

11. *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

Employee_N	First_Name	Last_Name	Start_Date	End_Date
009782	Della	Carlisle	12/8/2010	1/1/2099
015653	Montgomery	Clemans	12/12/2010	1/1/2099
004625	Augustin	Colthorp	12/8/2010	1/1/2099
001121	Ellison	Dengler	12/4/2010	1/1/2099
011832	Marti	Duhala	12/12/2010	1/1/2099
002682	Esau	Gillespie	12/4/2010	1/1/2099
006944	Rachmiel	Guzman	12/12/2010	1/1/2099
008122	Terri	Hardison	12/12/2010	1/1/2099
005881	Nemesio	Ivory	12/4/2010	1/1/2099
007693	Logan	Kitchings	12/12/2010	1/1/2099
006256	Tania	Orozco	12/4/2010	1/1/2099
007158	Sharyn	Pahl	12/4/2010	1/1/2099
012347	Amarilis	Pesce	12/12/2010	1/1/2099
013850	Laramie	Potts	12/8/2010	1/1/2099
014139	Terrina	Rennenger	12/4/2010	1/1/2099
015441	Darrel	Salasky	12/12/2010	1/1/2099
003036	Antoinette	Steen	12/12/2010	1/1/2099
002963	Amana	York	12/8/2010	1/1/2099

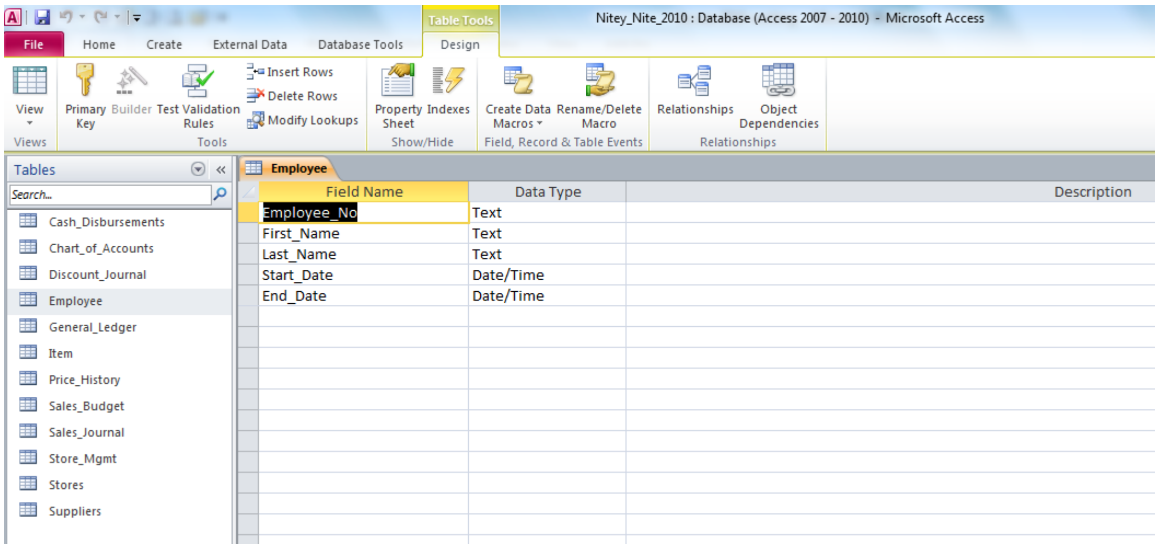
Figure 1.12

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 Excel 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.

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



Field Name	Data Type	Description
Employee_No	Text	
First_Name	Text	
Last_Name	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 will 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 *Text, Memo, 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.

> **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.

> **Memo**: Memo 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 Memo 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.

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.

Date/Time: As implied, this field stores numbers that are formatted in a date/time format. 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.

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.

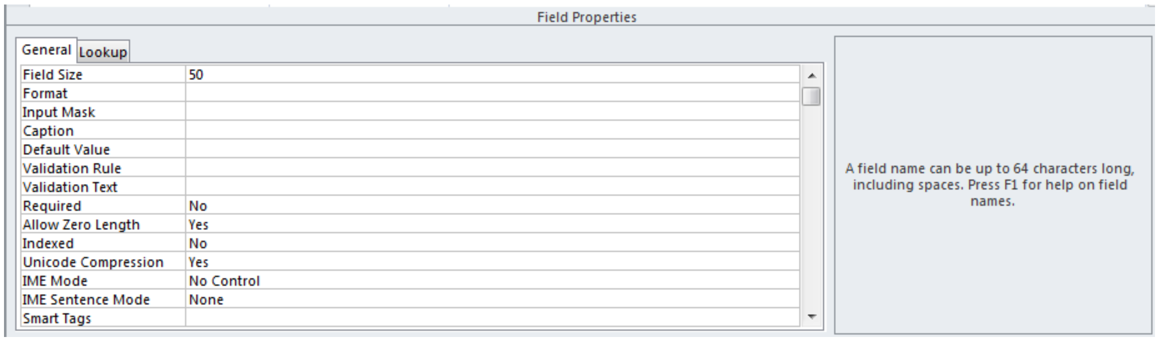
Attachment: This data type is new in Office Access 2010 files. You can attach images, spreadsheet files, documents, charts, and other types of supported files to the records in your database, much like you attach files to e-mail messages. You can also view and edit attached files, depending on how the 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 is 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
Smart Tags	

A field name can be up to 64 characters long, including spaces. Press F1 for help on field names.

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 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.

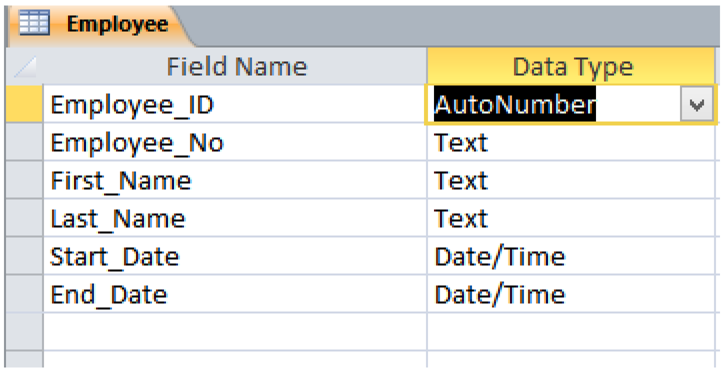
The **Indexed** property should be used with caution. It is used to speed up performance when querying a large table. However, if the index is not set on the right fields, it can severely slow down performance when updating, adding or deleting records from the table. Once you get to a point where performance 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.

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



Employee

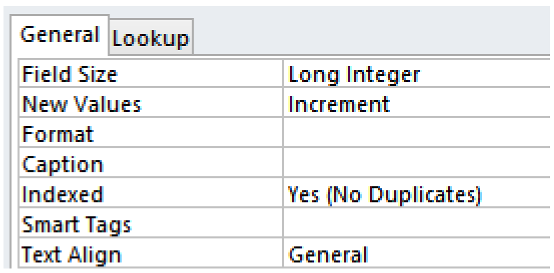
Field Name	Data Type
Employee_ID	AutoNumber
Employee_No	Text
First_Name	Text
Last_Name	Text
Start_Date	Date/Time
End_Date	Date/Time

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.

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

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 should look like this:



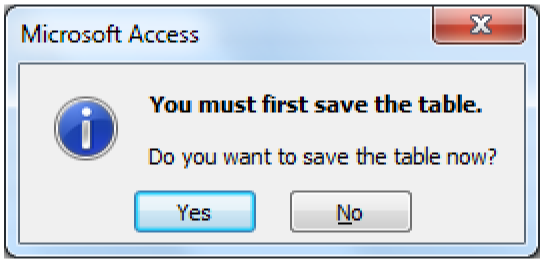
General | Lookup

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

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.

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


Microsoft Access

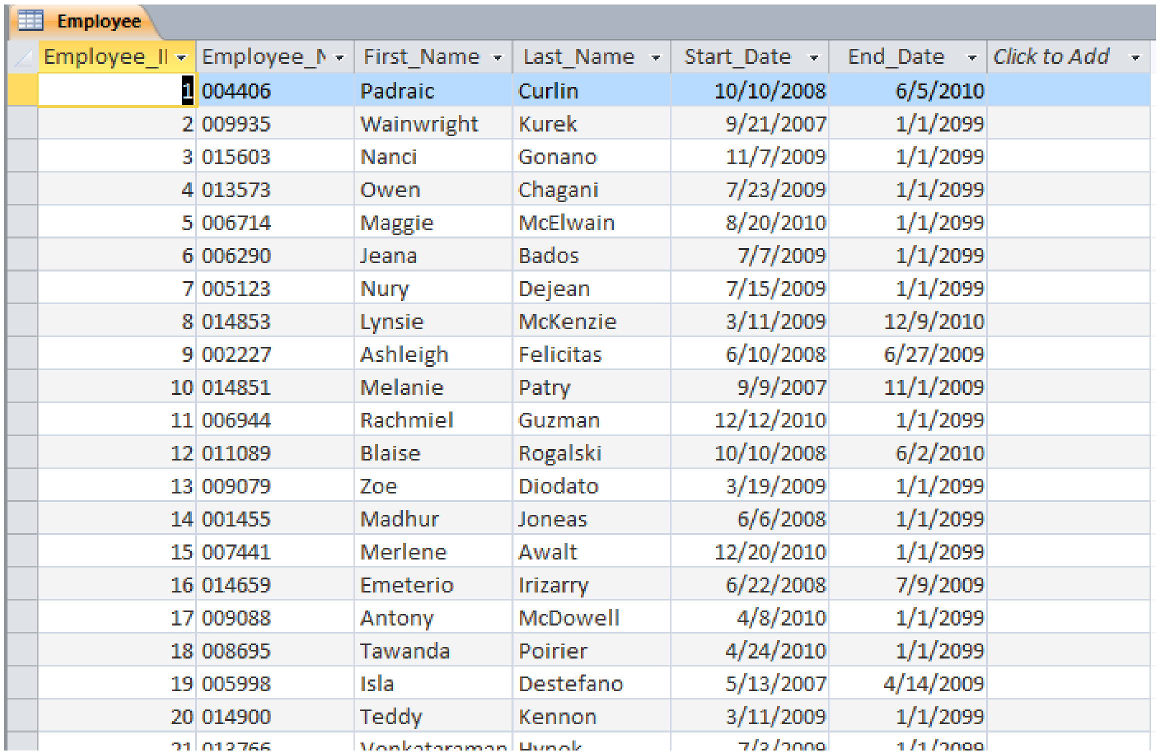
You must first save the table.

Do you want to save the table now?

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. 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.

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


Employee

Employee_I	Employee_N	First_Name	Last_Name	Start_Date	End_Date	Click to Add
1	004406	Padraic	Curlin	10/10/2008	6/5/2010	
2	009935	Wainwright	Kurek	9/21/2007	1/1/2099	
3	015603	Nanci	Gonano	11/7/2009	1/1/2099	
4	013573	Owen	Chagani	7/23/2009	1/1/2099	
5	006714	Maggie	McElwain	8/20/2010	1/1/2099	
6	006290	Jeana	Bados	7/7/2009	1/1/2099	
7	005123	Nury	Dejean	7/15/2009	1/1/2099	
8	014853	Lynsie	McKenzie	3/11/2009	12/9/2010	
9	002227	Ashleigh	Felicitas	6/10/2008	6/27/2009	
10	014851	Melanie	Patry	9/9/2007	11/1/2009	
11	006944	Rachmiel	Guzman	12/12/2010	1/1/2099	
12	011089	Blaise	Rogalski	10/10/2008	6/2/2010	
13	009079	Zoe	Diodato	3/19/2009	1/1/2099	
14	001455	Madhur	Joneas	6/6/2008	1/1/2099	
15	007441	Merlene	Awalt	12/20/2010	1/1/2099	
16	014659	Emeterio	Irizarry	6/22/2008	7/9/2009	
17	009088	Antony	McDowell	4/8/2010	1/1/2099	
18	008695	Tawanda	Poirier	4/24/2010	1/1/2099	
19	005998	Isla	Destefano	5/13/2007	4/14/2009	
20	014900	Teddy	Kennon	3/11/2009	1/1/2099	
21	013766	Venkataraman	Hynek	7/3/2009	1/1/2099	

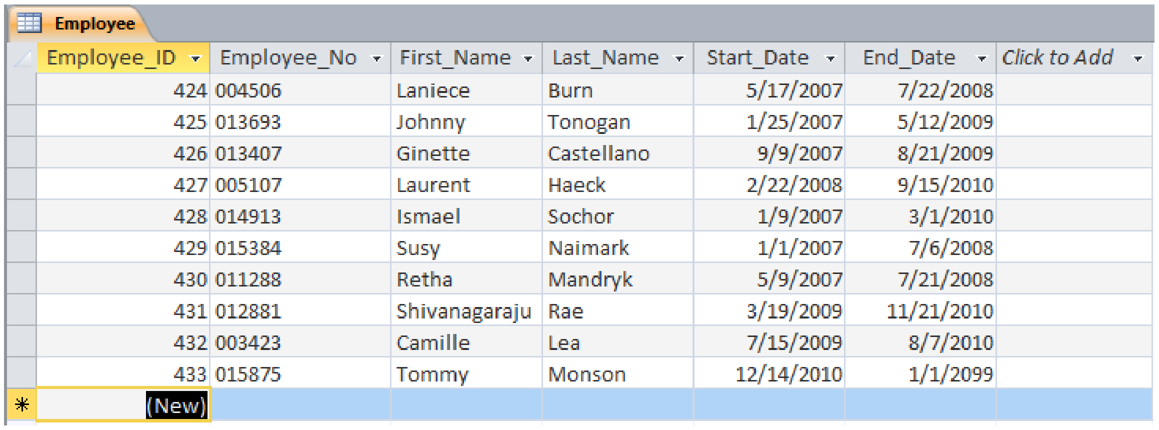
Figure 1.18

Notice that when you return to the Datasheet view of the table, the filter has been removed. By default, Access 2010 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 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:* **12/14/2010**.
6. *In the* **End_Date** *field, type:* **1/1/2099** *and press* **[Enter]**.


Employee

Employee_ID	Employee_No	First_Name	Last_Name	Start_Date	End_Date	*Click to Add*
424	004506	Laniece	Burn	5/17/2007	7/22/2008	
425	013693	Johnny	Tonogan	1/25/2007	5/12/2009	
426	013407	Ginette	Castellano	9/9/2007	8/21/2009	
427	005107	Laurent	Haeck	2/22/2008	9/15/2010	
428	014913	Ismael	Sochor	1/9/2007	3/1/2010	
429	015384	Susy	Naimark	1/1/2007	7/6/2008	
430	011288	Retha	Mandryk	5/9/2007	7/21/2008	
431	012881	Shivanagaraju	Rae	3/19/2009	11/21/2010	
432	003423	Camille	Lea	7/15/2009	8/7/2010	
433	015875	Tommy	Monson	12/14/2010	1/1/2099	
(New)						

Figure 1.19

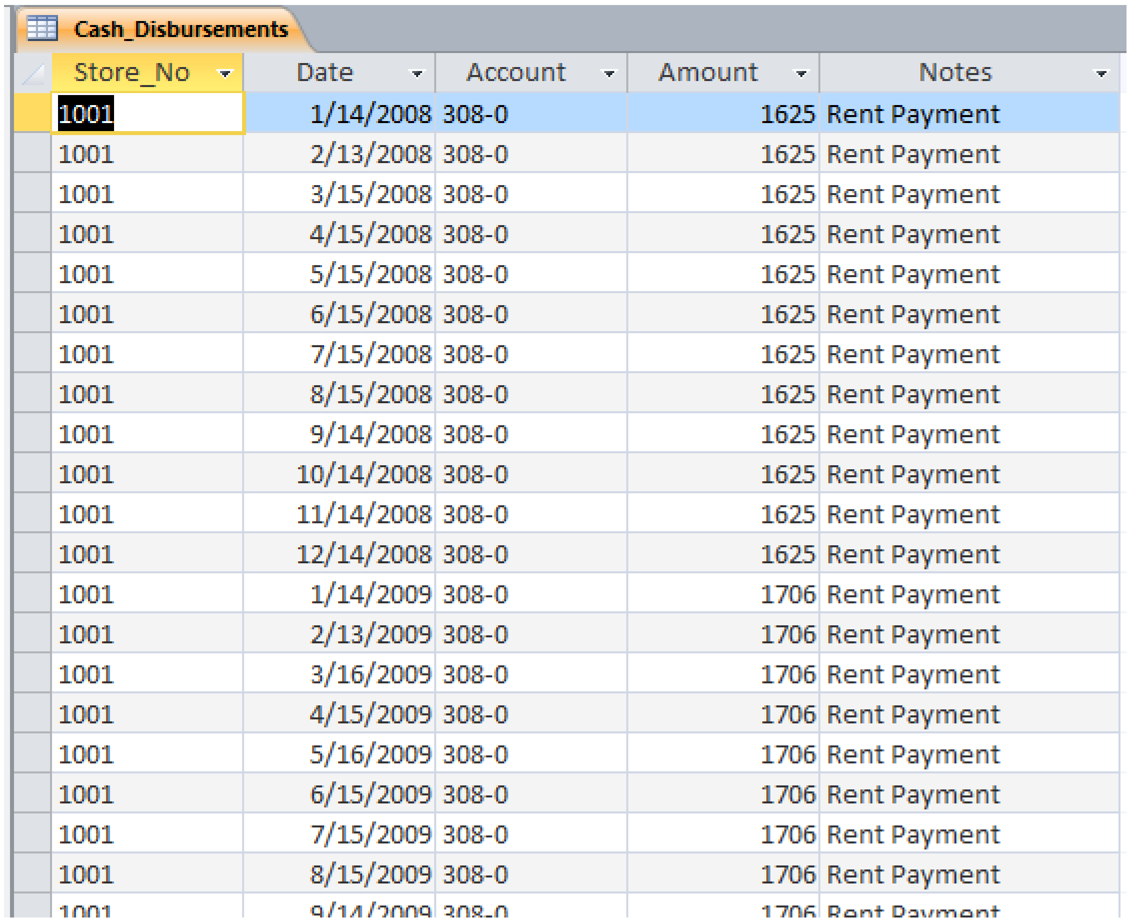
Congratulations! You are now the newest employee of Nitey-Nite Mattresses. Did you see how the AutoNumber was added when you started to type the Employee_No? When you got to the end of the record and pressed [Enter], your cursor moved to the next record (as if you were going to input another record),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

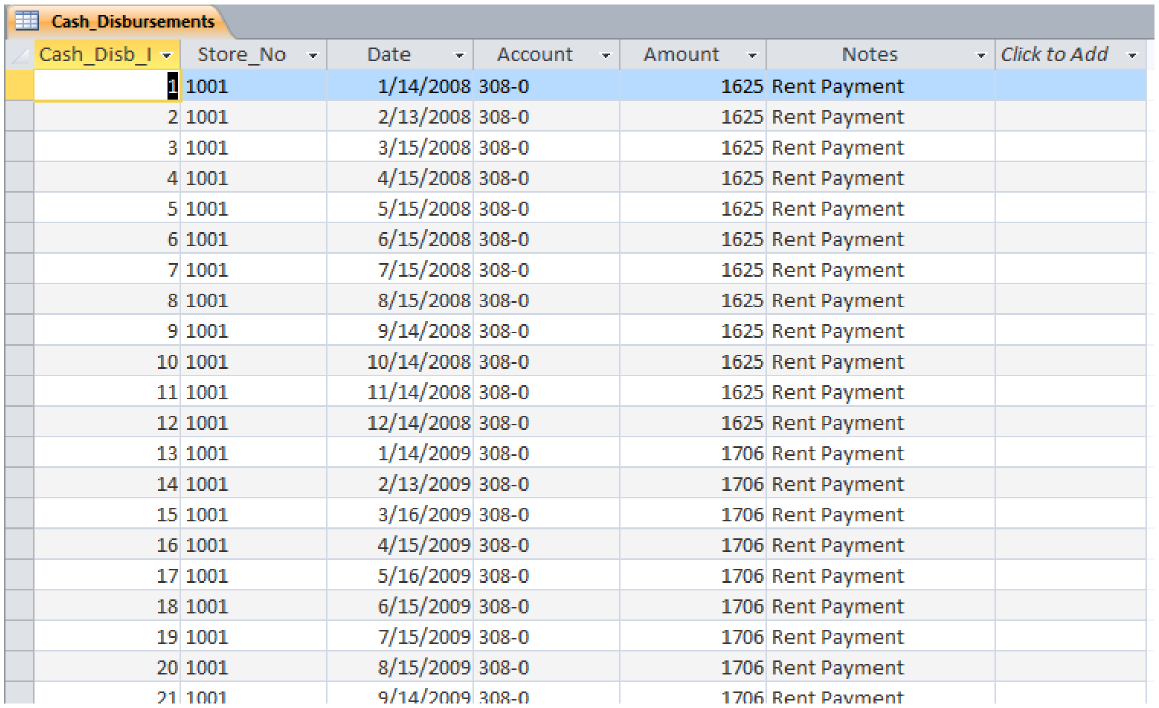


Store_No	Date	Account	Amount	Notes
1001	1/14/2008	308-0	1625	Rent Payment
1001	2/13/2008	308-0	1625	Rent Payment
1001	3/15/2008	308-0	1625	Rent Payment
1001	4/15/2008	308-0	1625	Rent Payment
1001	5/15/2008	308-0	1625	Rent Payment
1001	6/15/2008	308-0	1625	Rent Payment
1001	7/15/2008	308-0	1625	Rent Payment
1001	8/15/2008	308-0	1625	Rent Payment
1001	9/14/2008	308-0	1625	Rent Payment
1001	10/14/2008	308-0	1625	Rent Payment
1001	11/14/2008	308-0	1625	Rent Payment
1001	12/14/2008	308-0	1625	Rent Payment
1001	1/14/2009	308-0	1706	Rent Payment
1001	2/13/2009	308-0	1706	Rent Payment
1001	3/16/2009	308-0	1706	Rent Payment
1001	4/15/2009	308-0	1706	Rent Payment
1001	5/16/2009	308-0	1706	Rent Payment
1001	6/15/2009	308-0	1706	Rent Payment
1001	7/15/2009	308-0	1706	Rent Payment
1001	8/15/2009	308-0	1706	Rent Payment
1001	9/14/2009	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**.



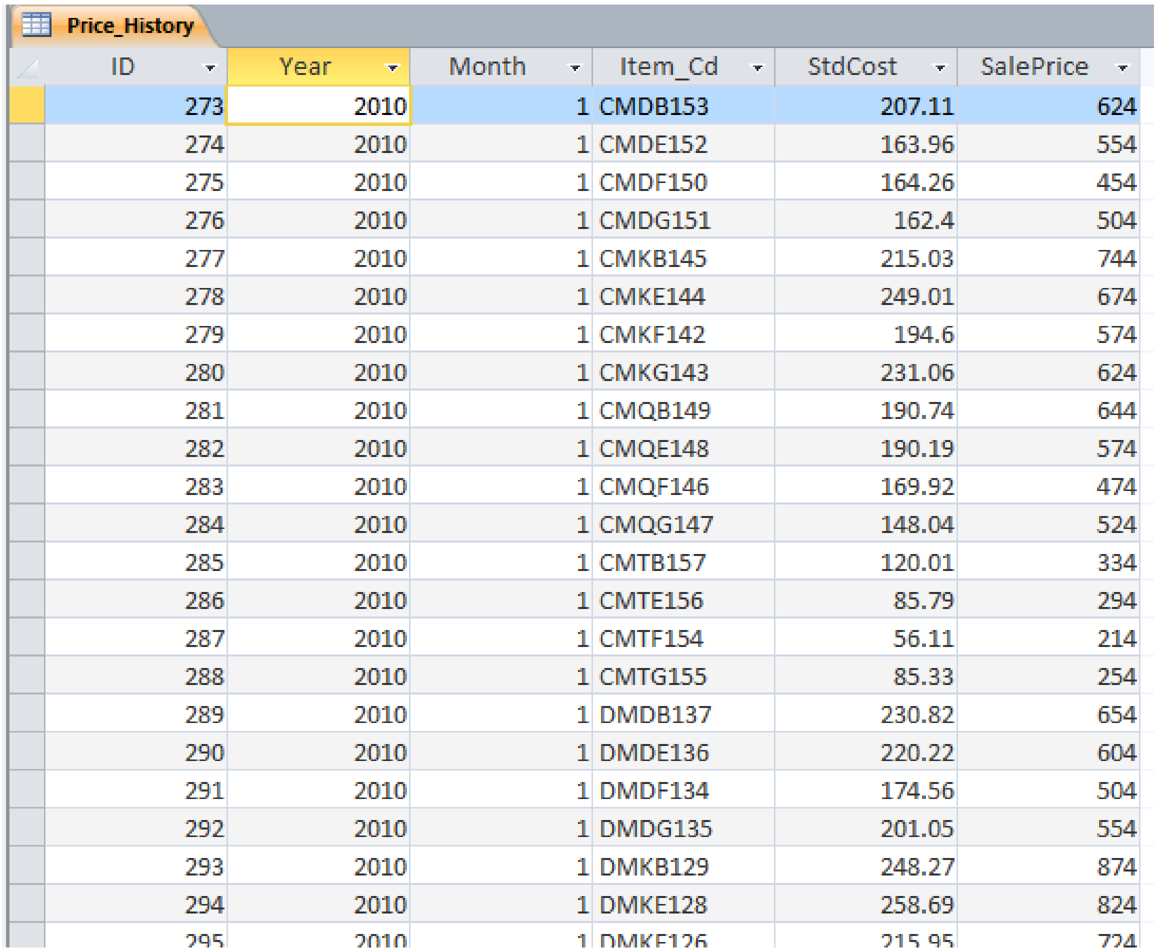
Cash_Disbursements

Cash_Disb_I	Store_No	Date	Account	Amount	Notes	Click to Add
1	1001	1/14/2008	308-0	1625	Rent Payment	
2	1001	2/13/2008	308-0	1625	Rent Payment	
3	1001	3/15/2008	308-0	1625	Rent Payment	
4	1001	4/15/2008	308-0	1625	Rent Payment	
5	1001	5/15/2008	308-0	1625	Rent Payment	
6	1001	6/15/2008	308-0	1625	Rent Payment	
7	1001	7/15/2008	308-0	1625	Rent Payment	
8	1001	8/15/2008	308-0	1625	Rent Payment	
9	1001	9/14/2008	308-0	1625	Rent Payment	
10	1001	10/14/2008	308-0	1625	Rent Payment	
11	1001	11/14/2008	308-0	1625	Rent Payment	
12	1001	12/14/2008	308-0	1625	Rent Payment	
13	1001	1/14/2009	308-0	1706	Rent Payment	
14	1001	2/13/2009	308-0	1706	Rent Payment	
15	1001	3/16/2009	308-0	1706	Rent Payment	
16	1001	4/15/2009	308-0	1706	Rent Payment	
17	1001	5/16/2009	308-0	1706	Rent Payment	
18	1001	6/15/2009	308-0	1706	Rent Payment	
19	1001	7/15/2009	308-0	1706	Rent Payment	
20	1001	8/15/2009	308-0	1706	Rent Payment	
21	1001	9/14/2009	308-0	1706	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.*



Price_History

ID	Year	Month	Item_Cd	StdCost	SalePrice
273	2010	1	CMDB153	207.11	624
274	2010	1	CMDE152	163.96	554
275	2010	1	CMDF150	164.26	454
276	2010	1	CMDG151	162.4	504
277	2010	1	CMKB145	215.03	744
278	2010	1	CMKE144	249.01	674
279	2010	1	CMKF142	194.6	574
280	2010	1	CMKG143	231.06	624
281	2010	1	CMQB149	190.74	644
282	2010	1	CMQE148	190.19	574
283	2010	1	CMQF146	169.92	474
284	2010	1	CMQG147	148.04	524
285	2010	1	CMTB157	120.01	334
286	2010	1	CMTE156	85.79	294
287	2010	1	CMTF154	56.11	214
288	2010	1	CMTG155	85.33	254
289	2010	1	DMDB137	230.82	654
290	2010	1	DMDE136	220.22	604
291	2010	1	DMDF134	174.56	504
292	2010	1	DMDG135	201.05	554
293	2010	1	DMKB129	248.27	874
294	2010	1	DMKE128	258.69	824
295	2010	1	DMKF126	215.95	724

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 2010 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 2010. 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, log in and take the exam. Make sure that you take the exam on the same computer that you completed the sample files on, as some of the questions on the exam may refer to some of the completed examples

ExcelCEO

Chief Excel Officer

Access 2010 and SQL

Complete Self-study Course

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.

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. Let's 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_2010** *database, click on the* **Create** *tab.*

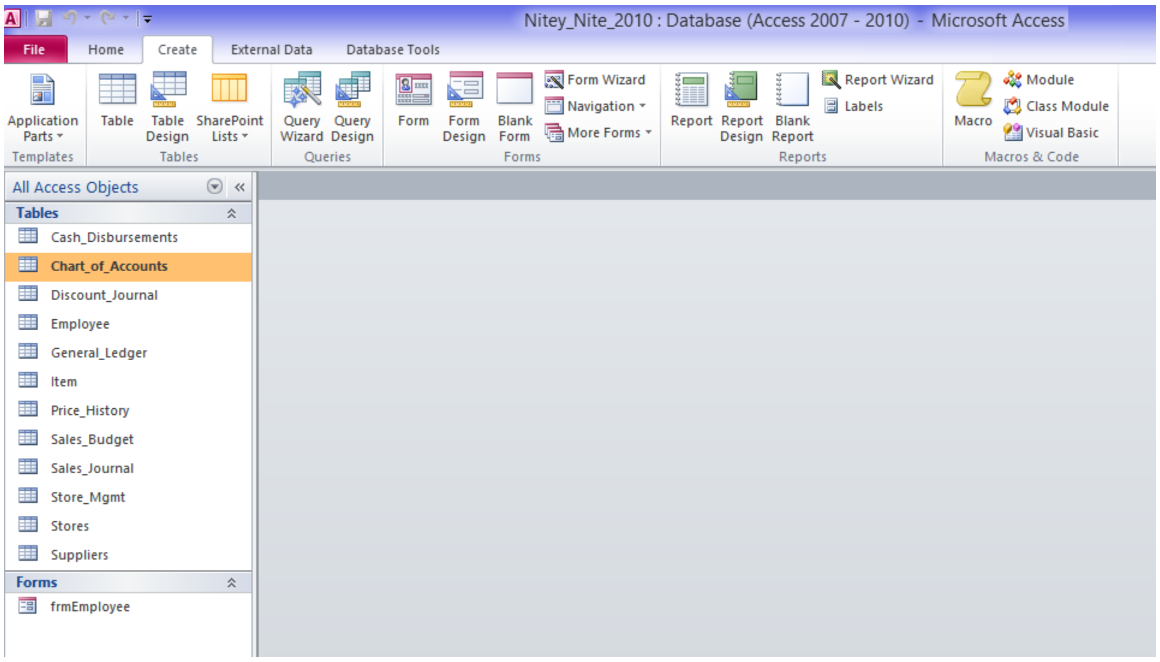


Figure 2.1

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_2010 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.*

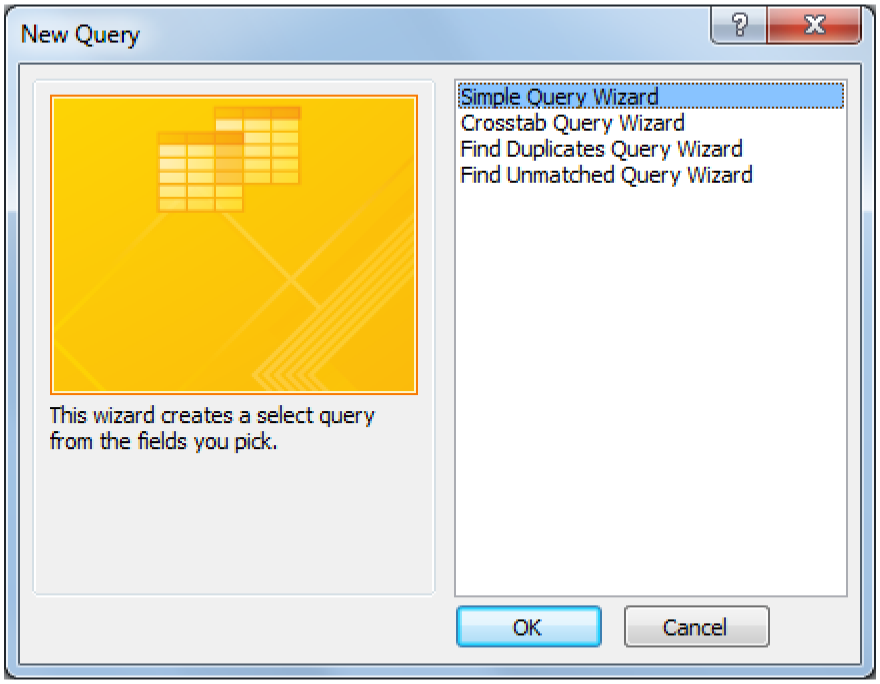
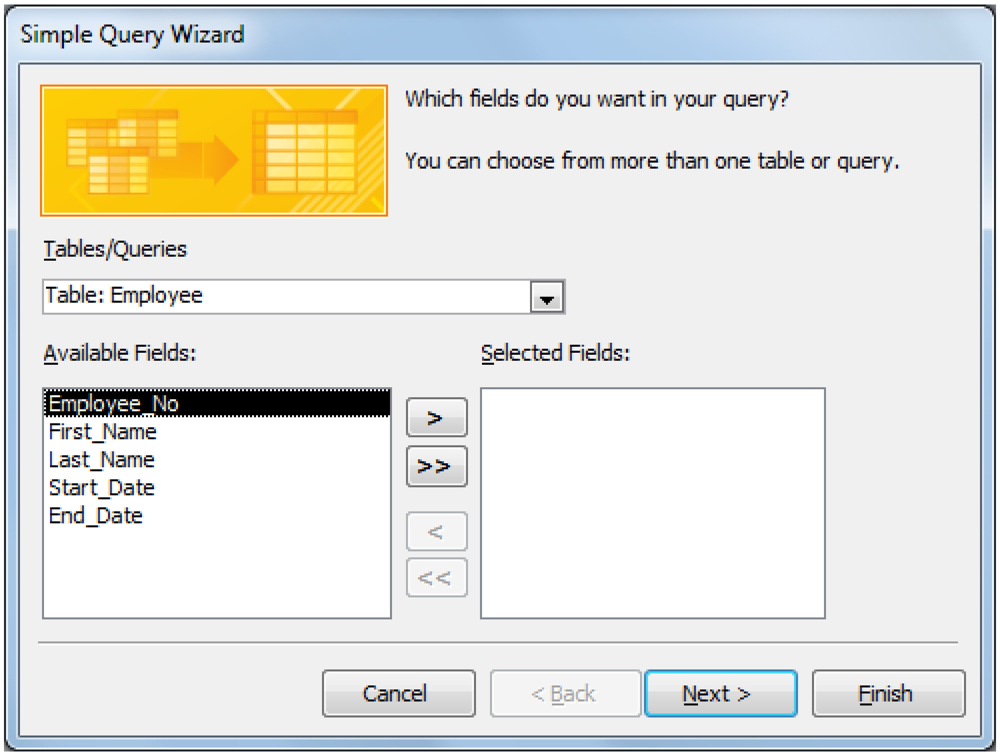

New Query
This wizard creates a select query from the fields you pick.
Simple Query Wizard
Crosstab Query Wizard
Find Duplicates Query Wizard
Find Unmatched Query Wizard
OK
Cancel

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 course. 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: Employee

Available Fields:

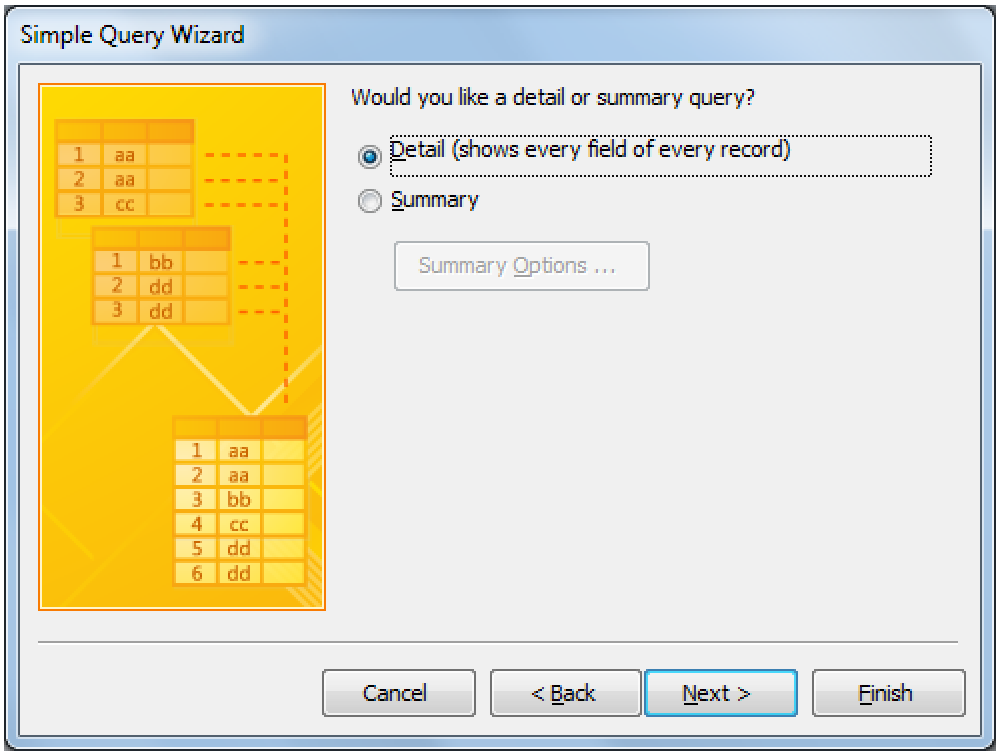
Employee_No
First_Name
Last_Name
Start_Date
End_Date

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 >.**



Would you like a detail or summary query?

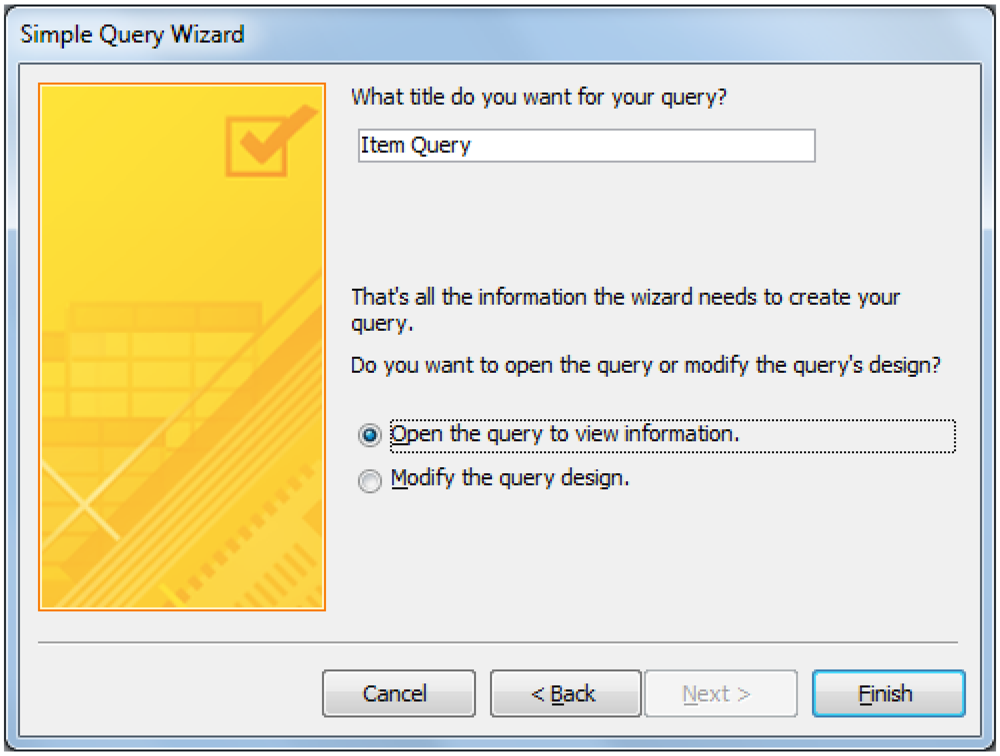
[✓ Detail (shows every field of every record)]

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**>.



What title do you want for your query?

Item Query

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

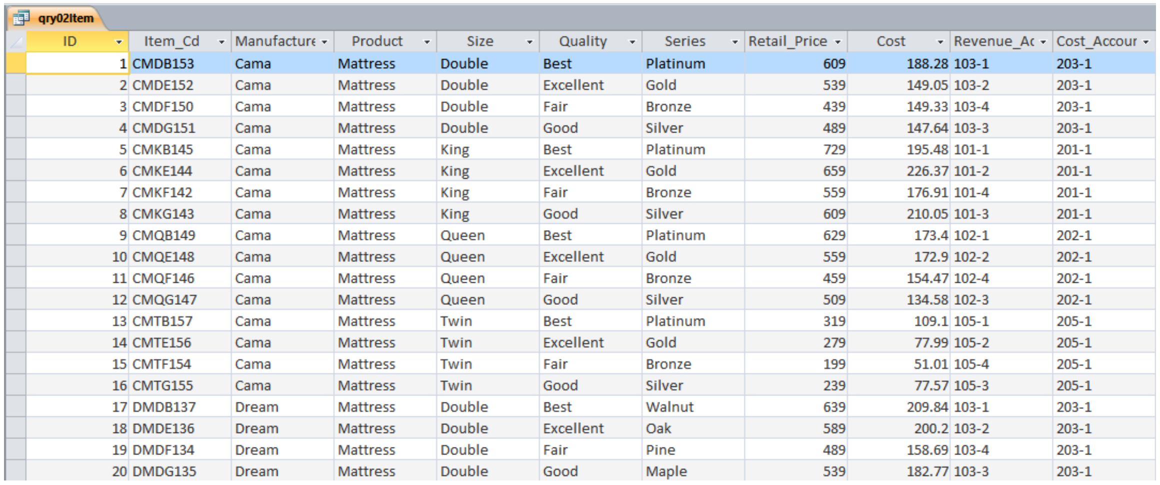
Do you want to open the query or modify the query's design?

- [✓ 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 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**.



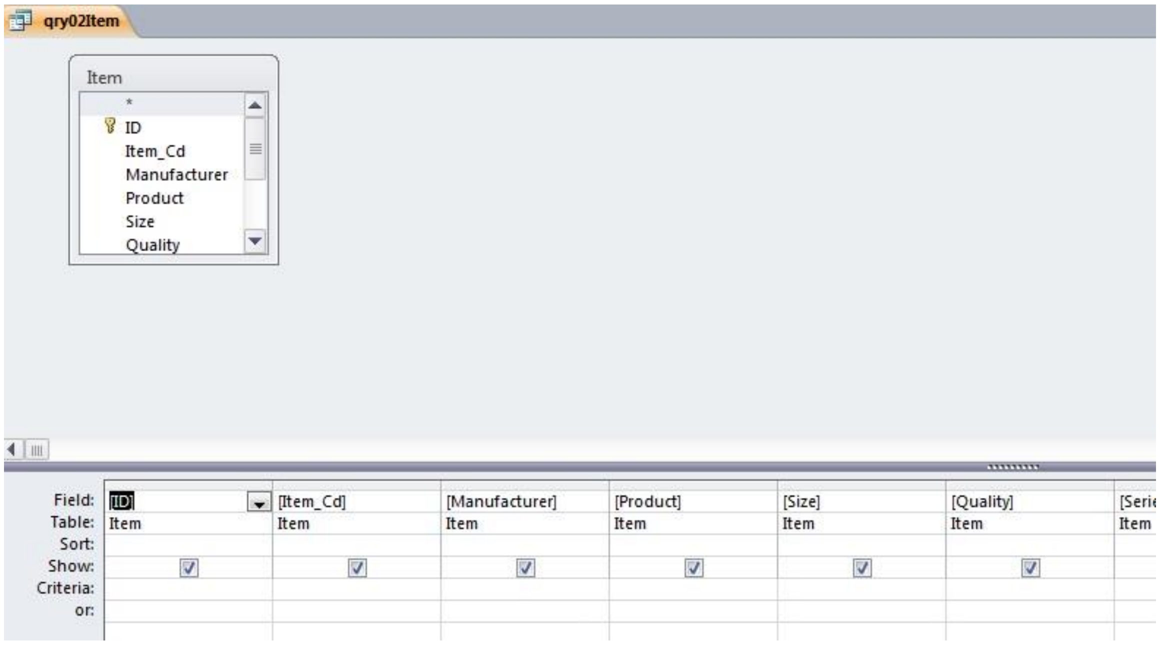
qry02Item

ID	Item_Cd	Manufacture	Product	Size	Quality	Series	Retail_Price	Cost	Revenue_Ac	Cost_Accour
1	CMDB153	Cama	Mattress	Double	Best	Platinum	609	188.28	103-1	203-1
2	CMDE152	Cama	Mattress	Double	Excellent	Gold	539	149.05	103-2	203-1
3	CMDF150	Cama	Mattress	Double	Fair	Bronze	439	149.33	103-4	203-1
4	CMDG151	Cama	Mattress	Double	Good	Silver	489	147.64	103-3	203-1
5	CMKB145	Cama	Mattress	King	Best	Platinum	729	195.48	101-1	201-1
6	CMKE144	Cama	Mattress	King	Excellent	Gold	659	226.37	101-2	201-1
7	CMKF142	Cama	Mattress	King	Fair	Bronze	559	176.91	101-4	201-1
8	CMKG143	Cama	Mattress	King	Good	Silver	609	210.05	101-3	201-1
9	CMQB149	Cama	Mattress	Queen	Best	Platinum	629	173.4	102-1	202-1
10	CMQE148	Cama	Mattress	Queen	Excellent	Gold	559	172.9	102-2	202-1
11	CMQF146	Cama	Mattress	Queen	Fair	Bronze	459	154.47	102-4	202-1
12	CMQG147	Cama	Mattress	Queen	Good	Silver	509	134.58	102-3	202-1
13	CMTB157	Cama	Mattress	Twin	Best	Platinum	319	109.1	105-1	205-1
14	CMTE156	Cama	Mattress	Twin	Excellent	Gold	279	77.99	105-2	205-1
15	CMTF154	Cama	Mattress	Twin	Fair	Bronze	199	51.01	105-4	205-1
16	CMTG155	Cama	Mattress	Twin	Good	Silver	239	77.57	105-3	205-1
17	DMDB137	Dream	Mattress	Double	Best	Walnut	639	209.84	103-1	203-1
18	DMDE136	Dream	Mattress	Double	Excellent	Oak	589	200.2	103-2	203-1
19	DMDF134	Dream	Mattress	Double	Fair	Pine	489	158.69	103-4	203-1
20	DMDG135	Dream	Mattress	Double	Good	Maple	539	182.77	103-3	203-1

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

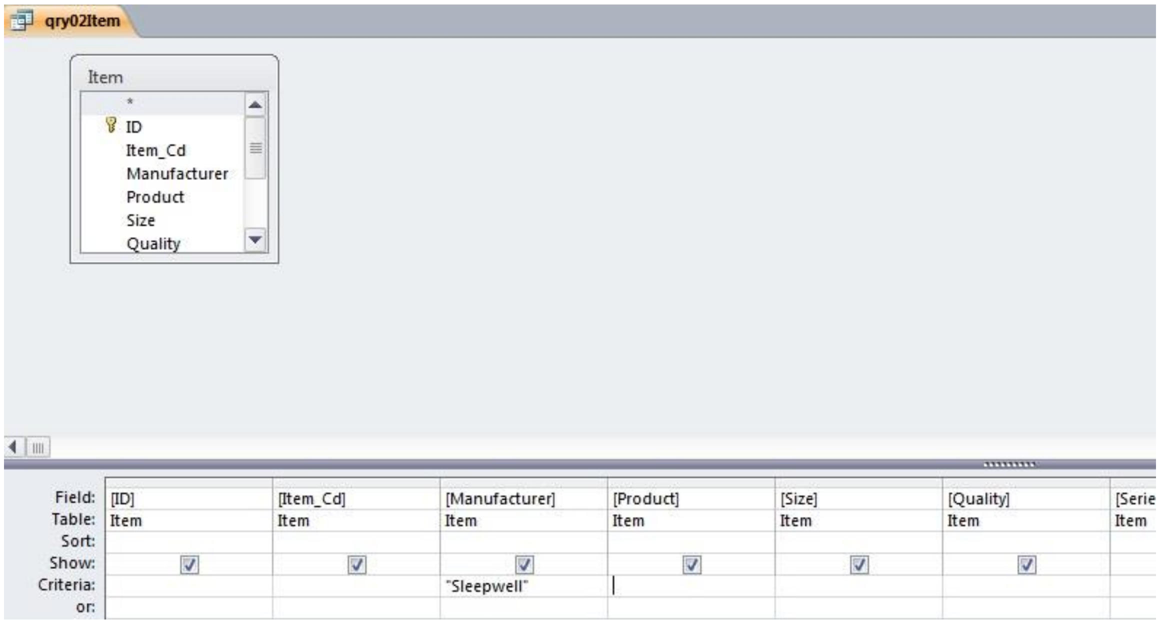


Field:	[ID]	[Item_Cd]	[Manufacturer]	[Product]	[Size]	[Quality]	[Serie
Table:	Item	Item	Item	Item	Item	Item	Item
Sort:							
Show:	✓	✓	✓	✓	✓	✓	
Criteria:							
or:							

Figure 2.7

The Design view of the query appears. Notice that the field names appear in brackets. In Access 2010, 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]**.

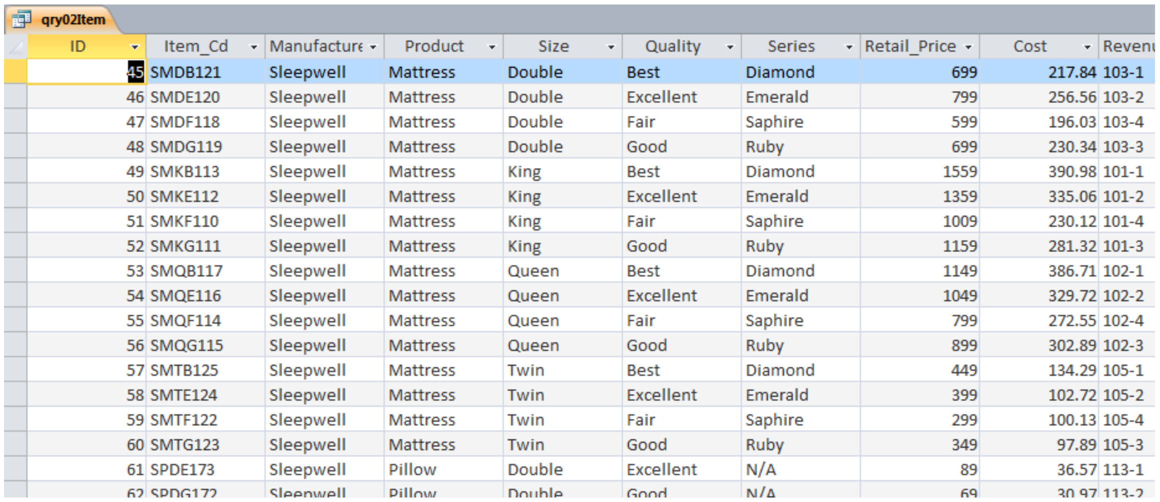


Field:	[ID]	[Item_Cd]	[Manufacturer]	[Product]	[Size]	[Quality]	[Serie
Table:	Item	Item	Item	Item	Item	Item	Item
Sort:							
Show:	✓	✓	✓	✓	✓	✓	
Criteria:			"Sleepwell"				
or:							

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	Revenu
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

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 query.*

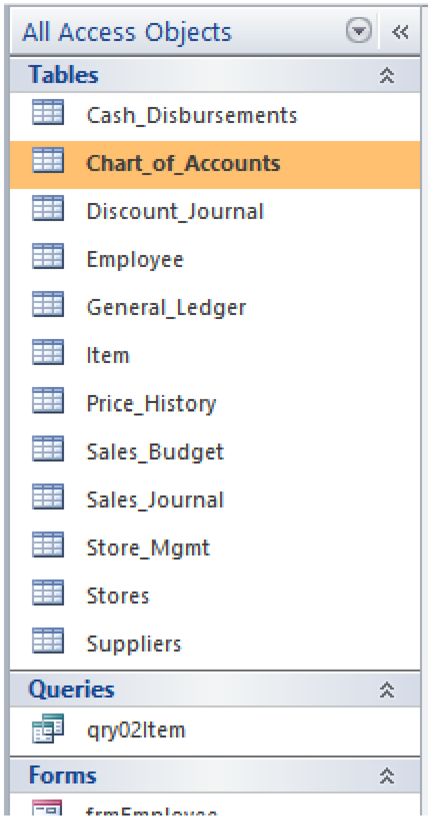


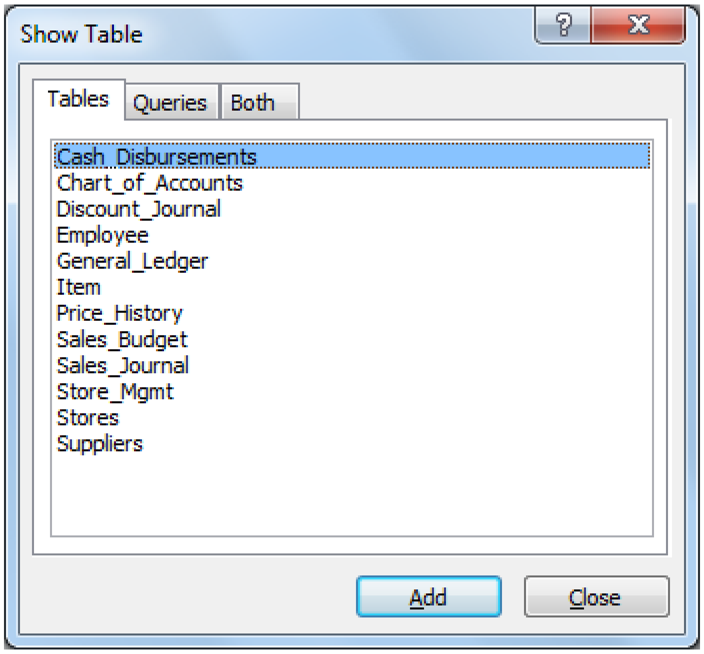
Figure 2.10

> ***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 2010 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.*



Cash_Disbursements
Chart_of_Accounts
Discount_Journal
Employee
General_Ledger
Item
Price_History
Sales_Budget
Sales_Journal
Store_Mgmt
Stores
Suppliers

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:

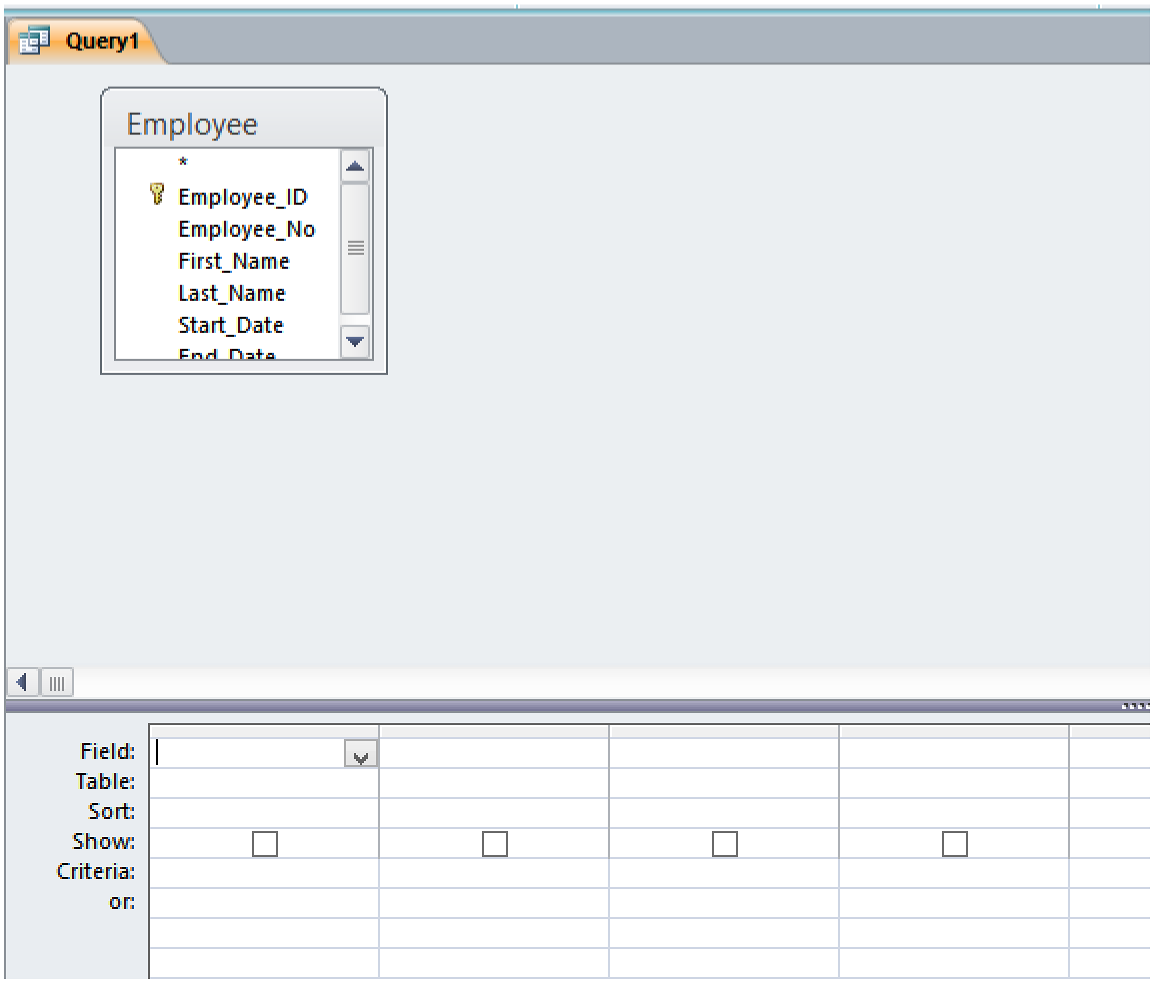


Figure 2.12

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 and 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.*

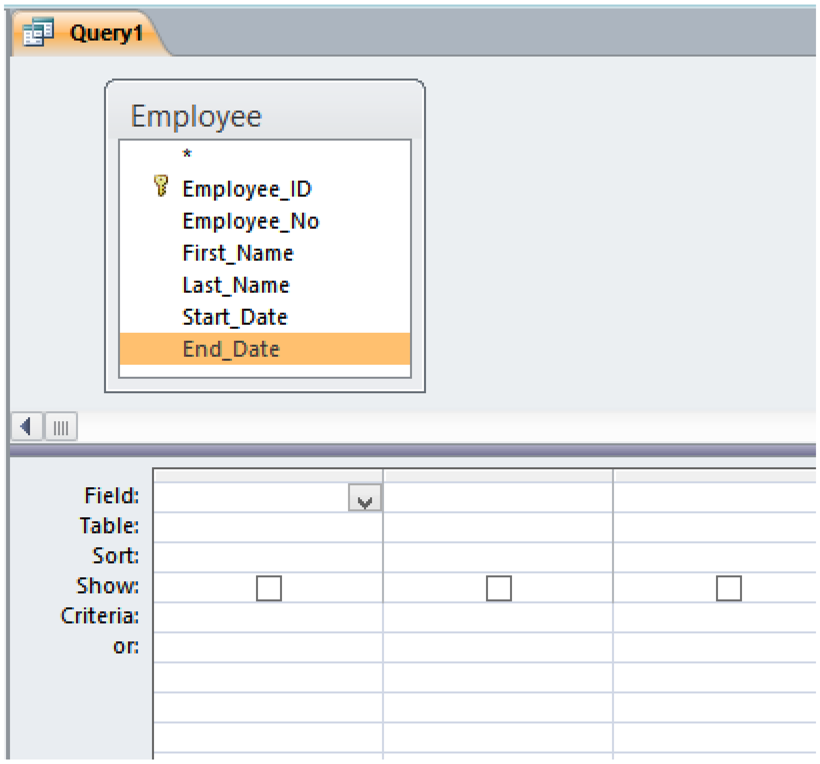
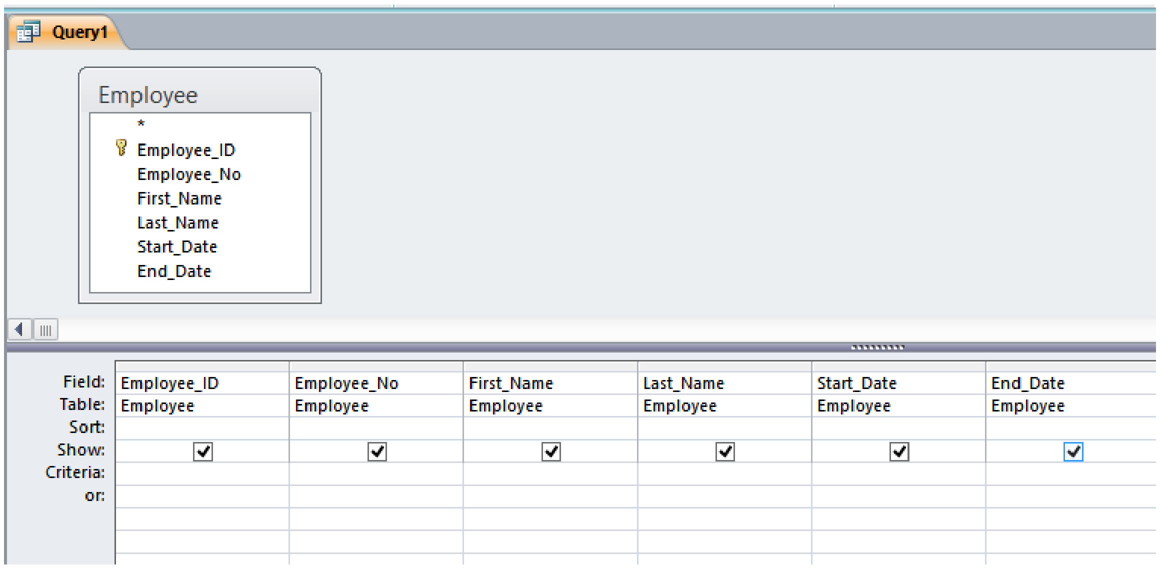


Figure 2.13

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**.

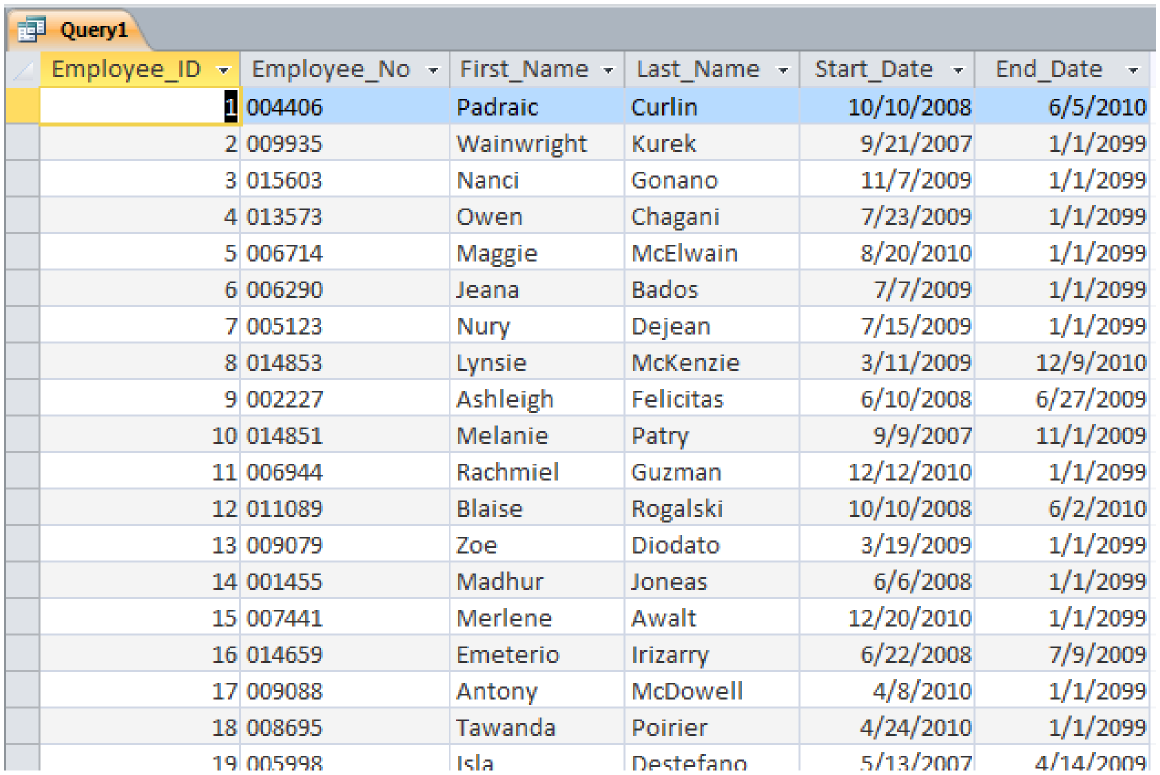


Field:	Employee_ID	Employee_No	First_Name	Last_Name	Start_Date	End_Date
Table:	Employee	Employee	Employee	Employee	Employee	Employee
Sort:						
Show:	✓	✓	✓	✓	✓	✓
Criteria:						
or:						

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	10/10/2008	6/5/2010
2	009935	Wainwright	Kurek	9/21/2007	1/1/2099
3	015603	Nanci	Gonano	11/7/2009	1/1/2099
4	013573	Owen	Chagani	7/23/2009	1/1/2099
5	006714	Maggie	McElwain	8/20/2010	1/1/2099
6	006290	Jeana	Bados	7/7/2009	1/1/2099
7	005123	Nury	Dejean	7/15/2009	1/1/2099
8	014853	Lynsie	McKenzie	3/11/2009	12/9/2010
9	002227	Ashleigh	Felicitas	6/10/2008	6/27/2009
10	014851	Melanie	Patry	9/9/2007	11/1/2009
11	006944	Rachmiel	Guzman	12/12/2010	1/1/2099
12	011089	Blaise	Rogalski	10/10/2008	6/2/2010
13	009079	Zoe	Diodato	3/19/2009	1/1/2099
14	001455	Madhur	Joneas	6/6/2008	1/1/2099
15	007441	Merlene	Awalt	12/20/2010	1/1/2099
16	014659	Emeterio	Irizarry	6/22/2008	7/9/2009
17	009088	Antony	McDowell	4/8/2010	1/1/2099
18	008695	Tawanda	Poirier	4/24/2010	1/1/2099
19	005998	Isla	Destefano	5/13/2007	4/14/2009

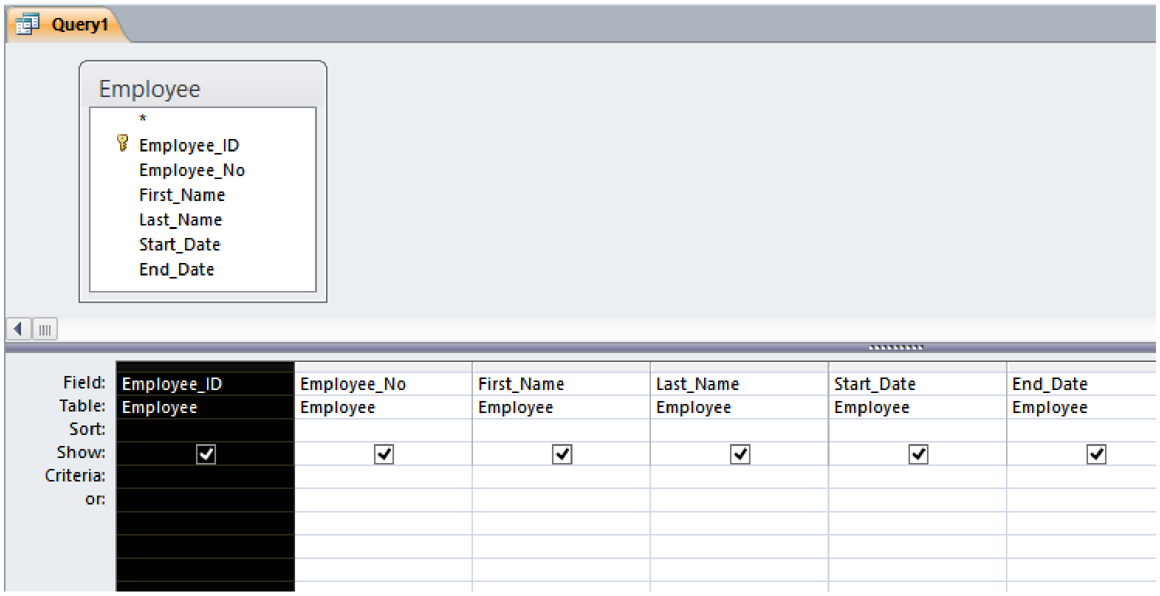
Figure 2.15

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.*



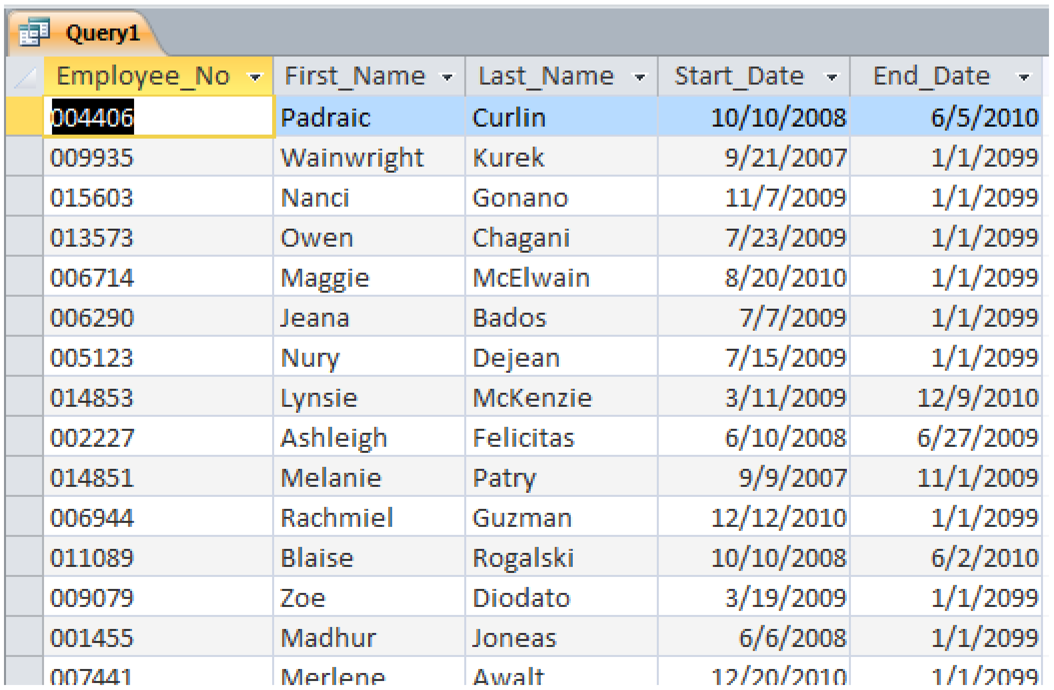
Field:	Employee_ID	Employee_No	First_Name	Last_Name	Start_Date	End_Date
Table:	Employee	Employee	Employee	Employee	Employee	Employee
Sort:						
Show:	✓	✓	✓	✓	✓	✓
Criteria:						
or:						

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**.



Query1

Employee_No	First_Name	Last_Name	Start_Date	End_Date
004406	Padraic	Curlin	10/10/2008	6/5/2010
009935	Wainwright	Kurek	9/21/2007	1/1/2099
015603	Nanci	Gonano	11/7/2009	1/1/2099
013573	Owen	Chagani	7/23/2009	1/1/2099
006714	Maggie	McElwain	8/20/2010	1/1/2099
006290	Jeana	Bados	7/7/2009	1/1/2099
005123	Nury	Dejean	7/15/2009	1/1/2099
014853	Lynsie	McKenzie	3/11/2009	12/9/2010
002227	Ashleigh	Felicitas	6/10/2008	6/27/2009
014851	Melanie	Patry	9/9/2007	11/1/2009
006944	Rachmiel	Guzman	12/12/2010	1/1/2099
011089	Blaise	Rogalski	10/10/2008	6/2/2010
009079	Zoe	Diodato	3/19/2009	1/1/2099
001455	Madhur	Joneas	6/6/2008	1/1/2099
007441	Merlene	Awalt	12/20/2010	1/1/2099

Figure 2.17

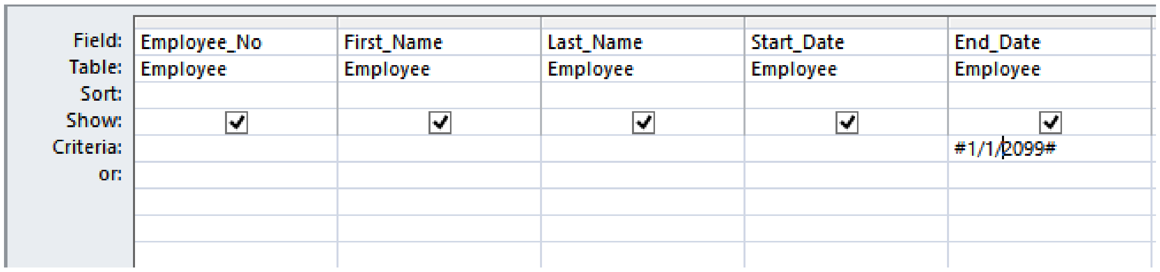
The Employee_ID field no longer appears in the data.

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]**.

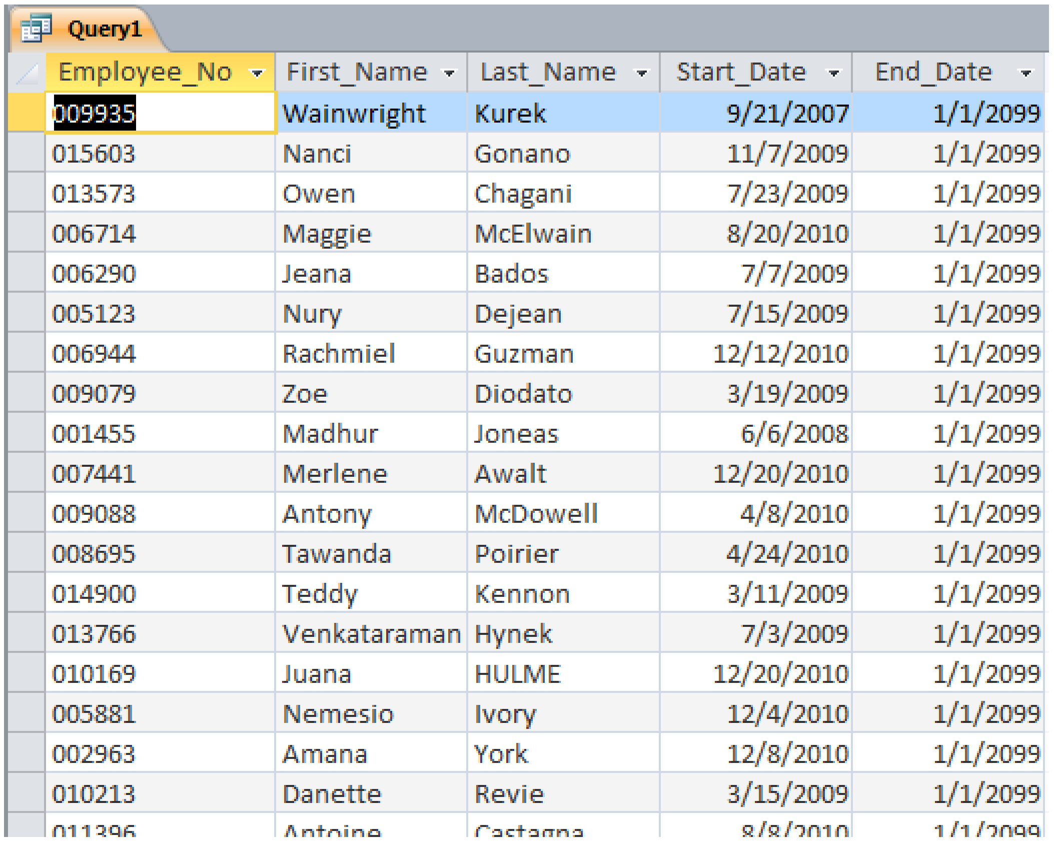
After you press [Enter], Access automatically surrounds 1/1/2099 with # signs. The Design Grid should look like this:



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**.



Query1

Employee_No	First_Name	Last_Name	Start_Date	End_Date
009935	Wainwright	Kurek	9/21/2007	1/1/2099
015603	Nanci	Gonano	11/7/2009	1/1/2099
013573	Owen	Chagani	7/23/2009	1/1/2099
006714	Maggie	McElwain	8/20/2010	1/1/2099
006290	Jeana	Bados	7/7/2009	1/1/2099
005123	Nury	Dejean	7/15/2009	1/1/2099
006944	Rachmiel	Guzman	12/12/2010	1/1/2099
009079	Zoe	Diodato	3/19/2009	1/1/2099
001455	Madhur	Joneas	6/6/2008	1/1/2099
007441	Merlene	Awalt	12/20/2010	1/1/2099
009088	Antony	McDowell	4/8/2010	1/1/2099
008695	Tawanda	Poirier	4/24/2010	1/1/2099
014900	Teddy	Kennon	3/11/2009	1/1/2099
013766	Venkataraman	Hynek	7/3/2009	1/1/2099
010169	Juana	HULME	12/20/2010	1/1/2099
005881	Nemesio	Ivory	12/4/2010	1/1/2099
002963	Amana	York	12/8/2010	1/1/2099
010213	Danette	Revie	3/15/2009	1/1/2099
011396	Antoine	Castagna	8/8/2010	1/1/2099

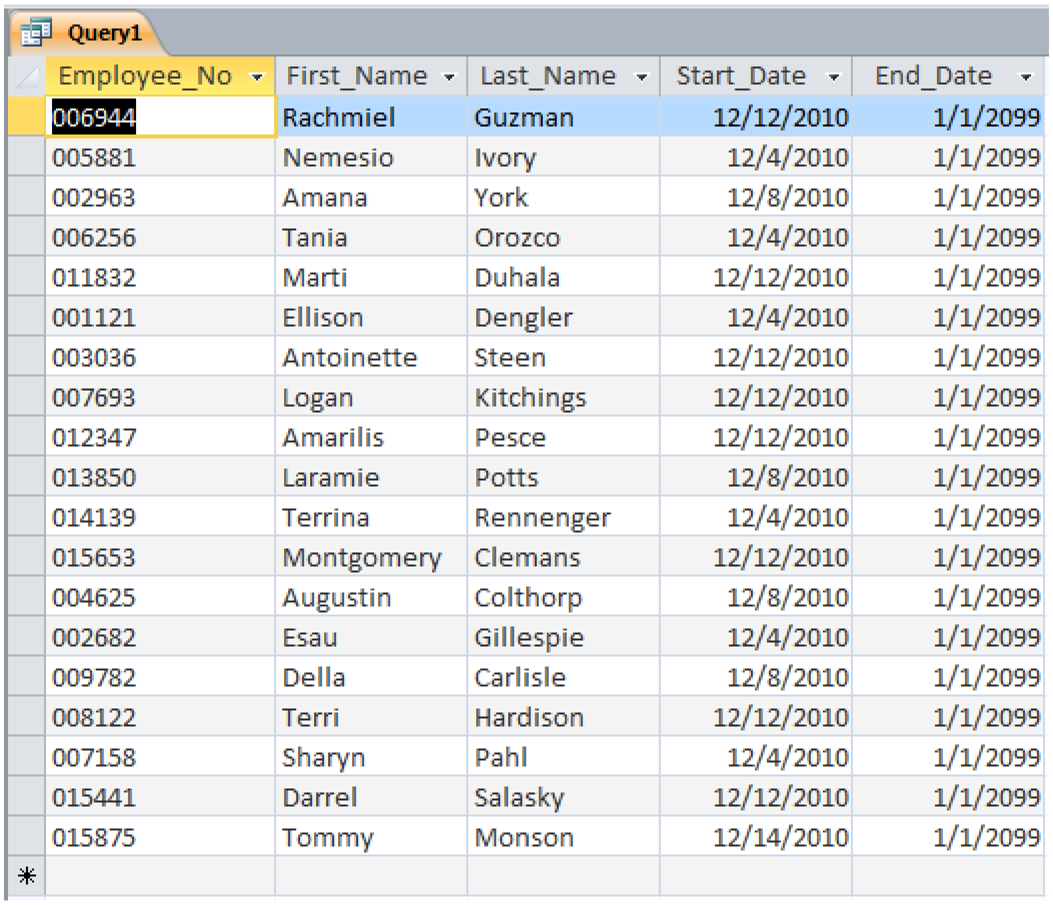
Figure 2.19

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 12/1/2010 and 12/15/2010.

1. *Go to the query's* **Design View**.
2. *Delete the criteria under* **End_Date**.
3. *On the* **Criteria:** *line for* **Start_Date**, *type* **>=12/1/2010 and <=12/15/2010** *and press* **[Enter]**.
4. *Go to* **Datasheet View**.



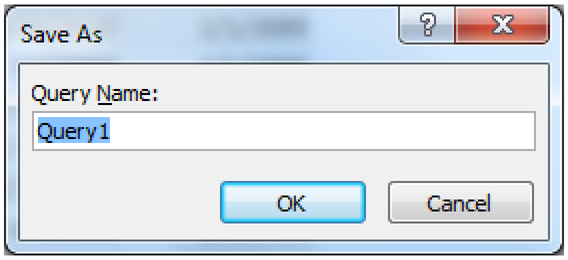
Query1

Employee_No	First_Name	Last_Name	Start_Date	End_Date
006944	Rachmiel	Guzman	12/12/2010	1/1/2099
005881	Nemesio	Ivory	12/4/2010	1/1/2099
002963	Amana	York	12/8/2010	1/1/2099
006256	Tania	Orozco	12/4/2010	1/1/2099
011832	Marti	Duhala	12/12/2010	1/1/2099
001121	Ellison	Dengler	12/4/2010	1/1/2099
003036	Antoinette	Steen	12/12/2010	1/1/2099
007693	Logan	Kitchings	12/12/2010	1/1/2099
012347	Amarilis	Pesce	12/12/2010	1/1/2099
013850	Laramie	Potts	12/8/2010	1/1/2099
014139	Terrina	Rennenger	12/4/2010	1/1/2099
015653	Montgomery	Clemans	12/12/2010	1/1/2099
004625	Augustin	Colthorp	12/8/2010	1/1/2099
002682	Esau	Gillespie	12/4/2010	1/1/2099
009782	Della	Carlisle	12/8/2010	1/1/2099
008122	Terri	Hardison	12/12/2010	1/1/2099
007158	Sharyn	Pahl	12/4/2010	1/1/2099
015441	Darrel	Salasky	12/12/2010	1/1/2099
015875	Tommy	Monson	12/14/2010	1/1/2099

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** *icon* *in the* **Quick Access Toolbar**.



Save As

Query Name: Query1

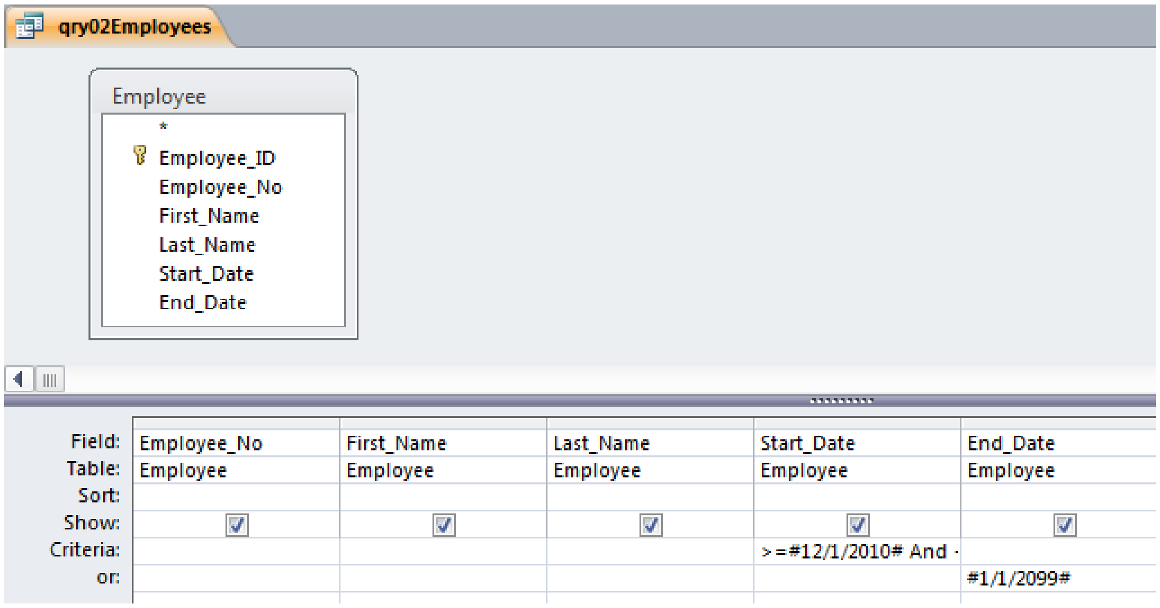
Figure 2.21

6. *Name the query* **qry02Employees** *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 12/1/2010 and 12/15/2010, 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* **qry02Employees** *and type* **1/1/2099** *on the* **Or** *line under the* **End_Date** *field.*

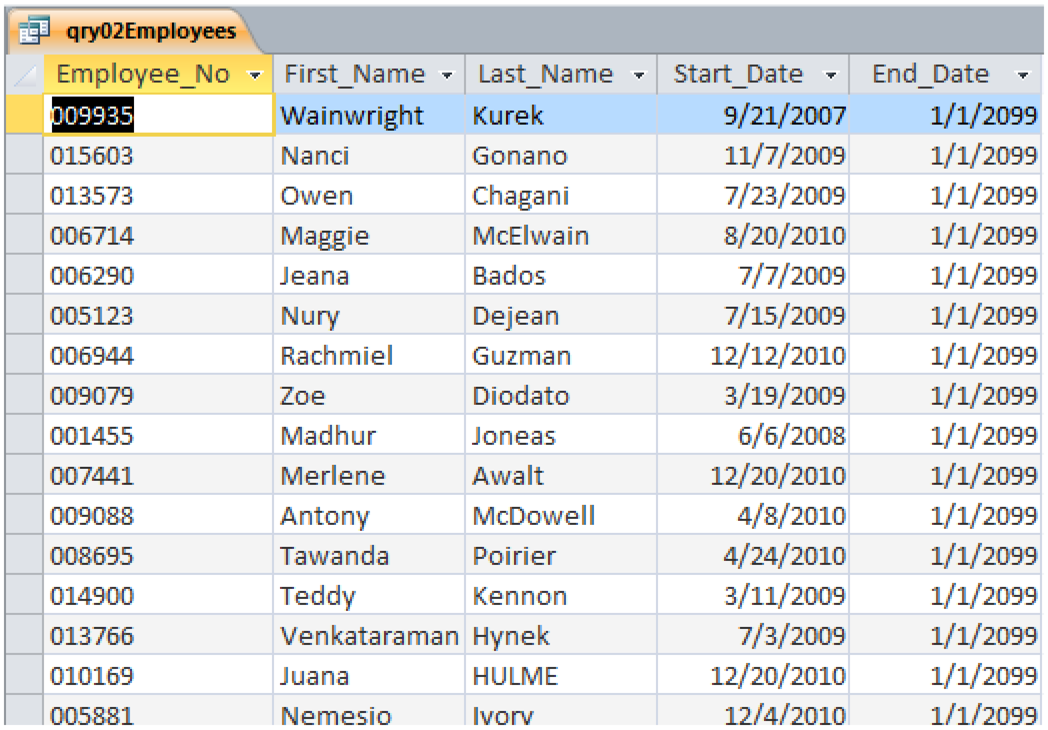


Field:	Employee_No	First_Name	Last_Name	Start_Date	End_Date
Table:	Employee	Employee	Employee	Employee	Employee
Sort:					
Show:	✓	✓	✓	✓	✓
Criteria:				>=#12/1/2010# And	
or:					#1/1/2099#

Figure 2.22

With this query, you are telling Access that you want a list that contains all employees with a start date between 12/1/2010 and 12/15/2010, 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 12/1/2010 and 12/15/2010. 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.

8. *Run the query.*


qry02Employees

Employee_No	First_Name	Last_Name	Start_Date	End_Date
009935	Wainwright	Kurek	9/21/2007	1/1/2099
015603	Nanci	Gonano	11/7/2009	1/1/2099
013573	Owen	Chagani	7/23/2009	1/1/2099
006714	Maggie	McElwain	8/20/2010	1/1/2099
006290	Jeana	Bados	7/7/2009	1/1/2099
005123	Nury	Dejean	7/15/2009	1/1/2099
006944	Rachmiel	Guzman	12/12/2010	1/1/2099
009079	Zoe	Diodato	3/19/2009	1/1/2099
001455	Madhur	Joneas	6/6/2008	1/1/2099
007441	Merlene	Awalt	12/20/2010	1/1/2099
009088	Antony	McDowell	4/8/2010	1/1/2099
008695	Tawanda	Poirier	4/24/2010	1/1/2099
014900	Teddy	Kennon	3/11/2009	1/1/2099
013766	Venkataraman	Hynek	7/3/2009	1/1/2099
010169	Juana	HULME	12/20/2010	1/1/2099
005881	Nemesio	Ivory	12/4/2010	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.

9. *Save and close the 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 2010 Review Questions Chapter 1, 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, log in and take the exam. Make sure that you take the exam on the same computer that you completed the sample files on, as some of the questions on the exam may refer to some of the completed examples.

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.

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 view it in 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 limit of Access 97. Access 2010 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. We will concentrate on writing formulas in queries and joining tables.

1. *In the* **Nitey_Nite_2010** *database, create a new query in* **Design** *view using the* **Cash_Disbursements** *table.*
2. *Modify the* **Design View** *to make it look like this:*

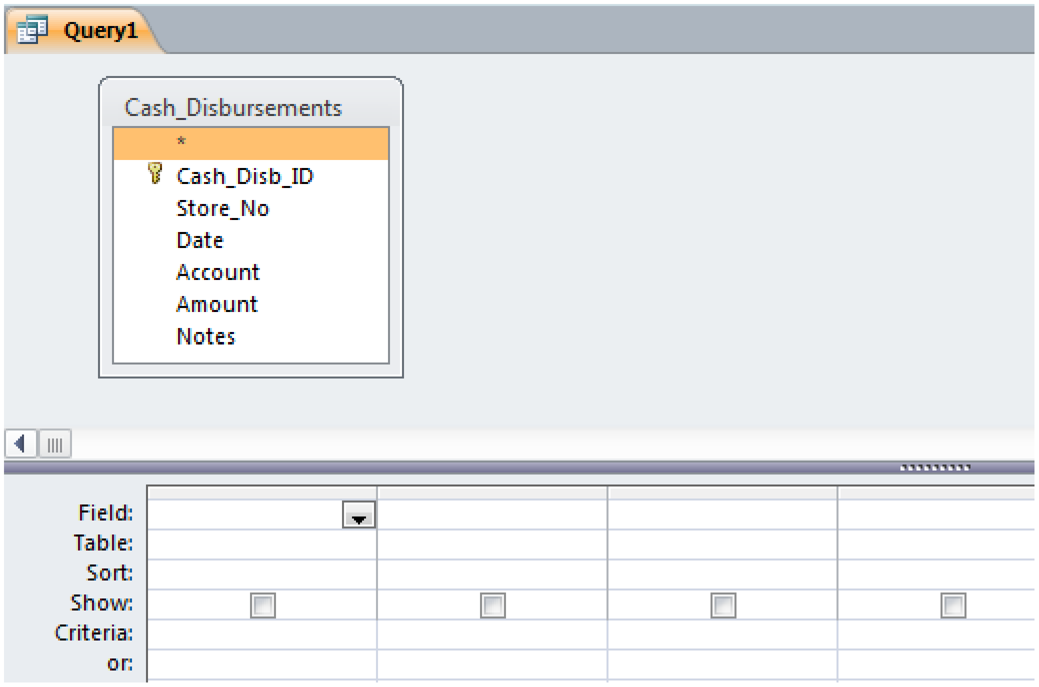


Figure 3.1

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**.

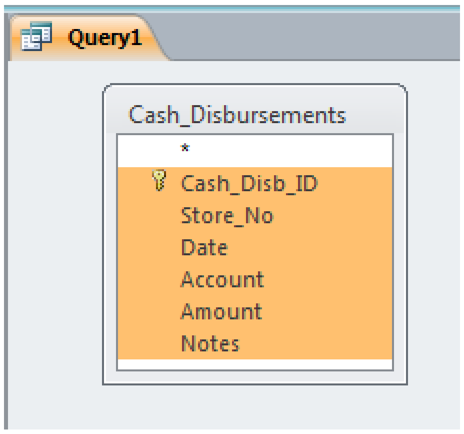
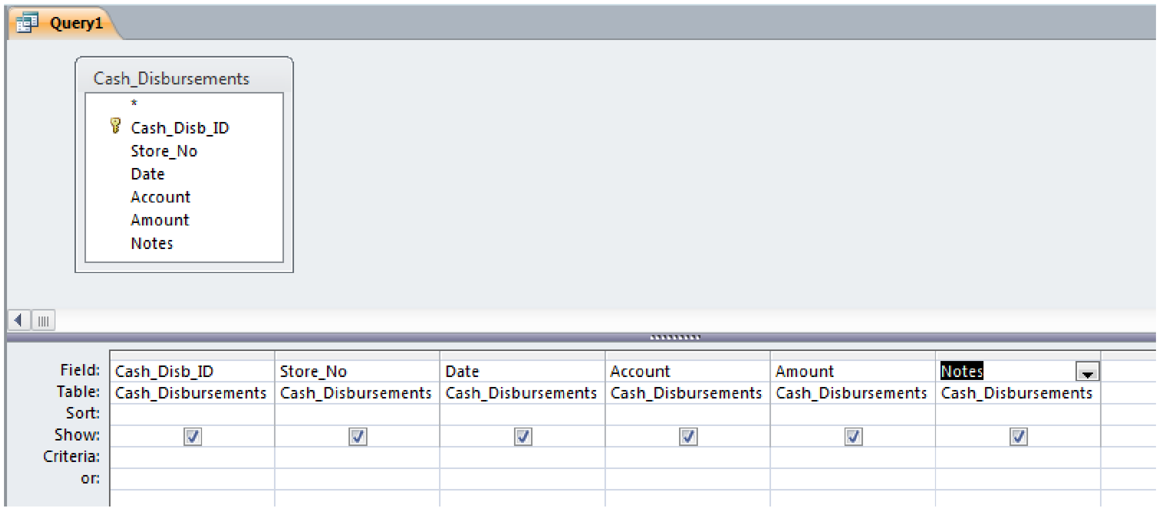


Figure 3.2

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.*

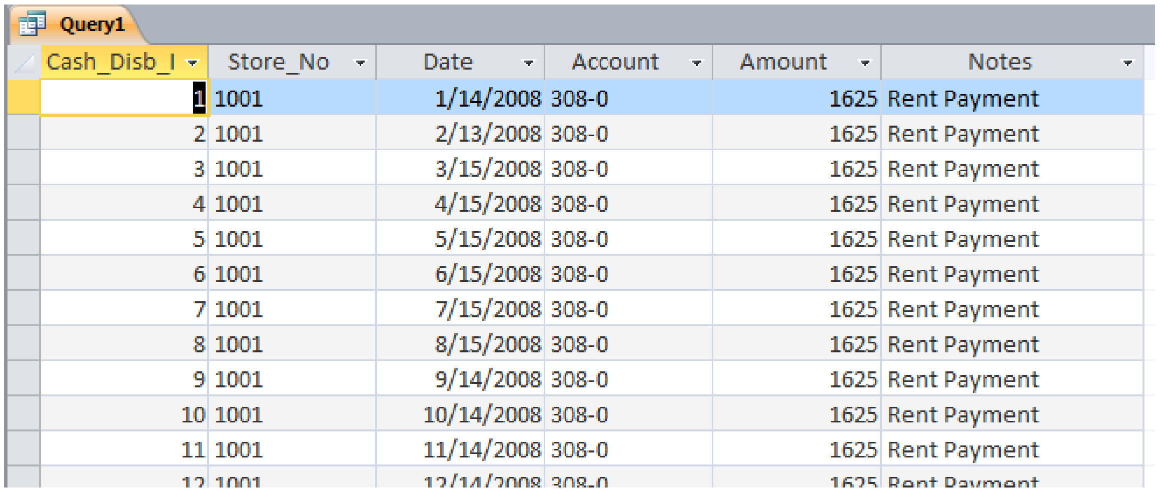


Field:	Cash_Disb_ID	Store_No	Date	Account	Amount	Notes
Table:	Cash_Disbursements	Cash_Disbursements	Cash_Disbursements	Cash_Disbursements	Cash_Disbursements	Cash_Disbursements
Sort:						
Show:	✓	✓	✓	✓	✓	✓
Criteria:						
or:						

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 (*) into the Design grid. I typically don't like to do that because only the asterisk appears in the Design grid, but all fields appear when you run the query.

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



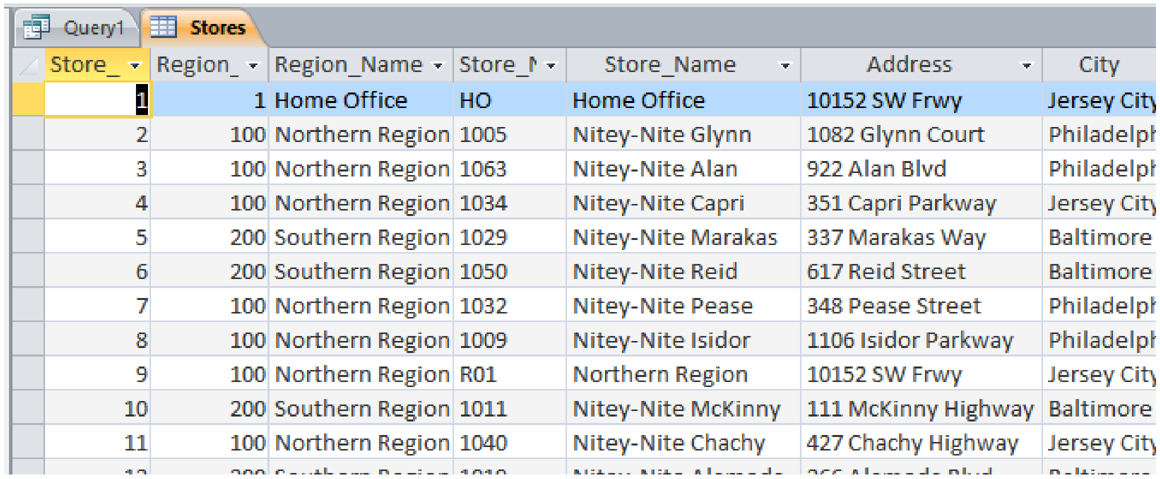
Query1

Cash_Disb_I	Store_No	Date	Account	Amount	Notes
1	1001	1/14/2008	308-0	1625	Rent Payment
2	1001	2/13/2008	308-0	1625	Rent Payment
3	1001	3/15/2008	308-0	1625	Rent Payment
4	1001	4/15/2008	308-0	1625	Rent Payment
5	1001	5/15/2008	308-0	1625	Rent Payment
6	1001	6/15/2008	308-0	1625	Rent Payment
7	1001	7/15/2008	308-0	1625	Rent Payment
8	1001	8/15/2008	308-0	1625	Rent Payment
9	1001	9/14/2008	308-0	1625	Rent Payment
10	1001	10/14/2008	308-0	1625	Rent Payment
11	1001	11/14/2008	308-0	1625	Rent Payment
12	1001	12/14/2008	308-0	1625	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 $1,625. 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_	Region_	Region_Name	Store_N	Store_Name	Address	City
1	1	Home Office	HO	Home Office	10152 SW Frwy	Jersey City
2	100	Northern Region	1005	Nitey-Nite Glynn	1082 Glynn Court	Philadelph
3	100	Northern Region	1063	Nitey-Nite Alan	922 Alan Blvd	Philadelph
4	100	Northern Region	1034	Nitey-Nite Capri	351 Capri Parkway	Jersey City
5	200	Southern Region	1029	Nitey-Nite Marakas	337 Marakas Way	Baltimore
6	200	Southern Region	1050	Nitey-Nite Reid	617 Reid Street	Baltimore
7	100	Northern Region	1032	Nitey-Nite Pease	348 Pease Street	Philadelph
8	100	Northern Region	1009	Nitey-Nite Isidor	1106 Isidor Parkway	Philadelph
9	100	Northern Region	R01	Northern Region	10152 SW Frwy	Jersey City
10	200	Southern Region	1011	Nitey-Nite McKinny	111 McKinny Highway	Baltimore
11	100	Northern Region	1040	Nitey-Nite Chachy	427 Chachy Highway	Jersey City

Figure 3.5

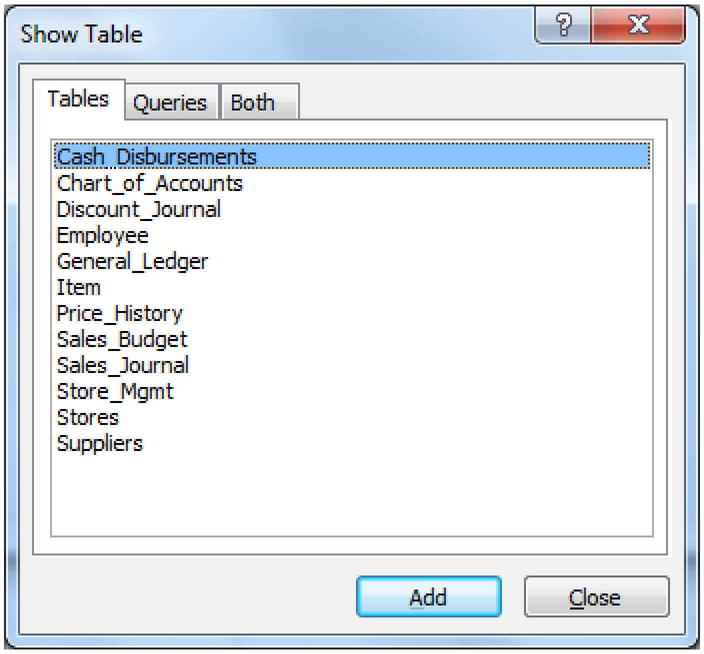
Joining Tables

A new tab opens up that contains the Stores table. The Stores table contains data 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* Show Table *in the* **Query Setup** *group of the* **Query Tools Design** *tab (to bring in another table).*

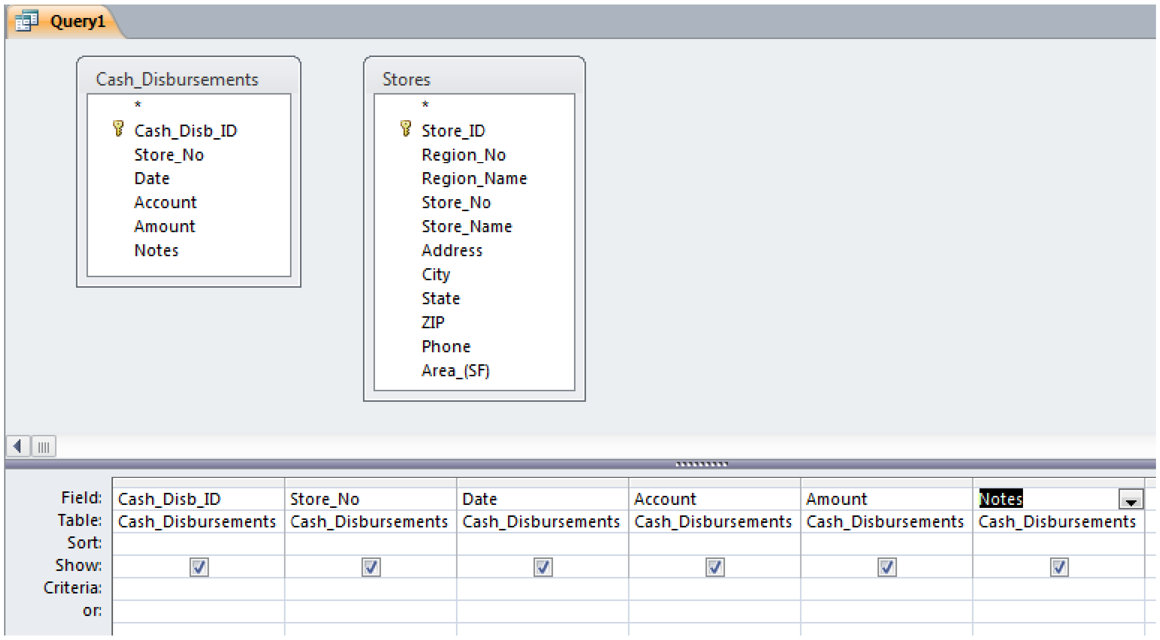


Cash_Disbursements
Chart_of_Accounts
Discount_Journal
Employee
General_Ledger
Item
Price_History
Sales_Budget
Sales_Journal
Store_Mgmt
Stores
Suppliers

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.*



Cash_Disbursements: *, Cash_Disb_ID, Store_No, Date, Account, Amount, Notes

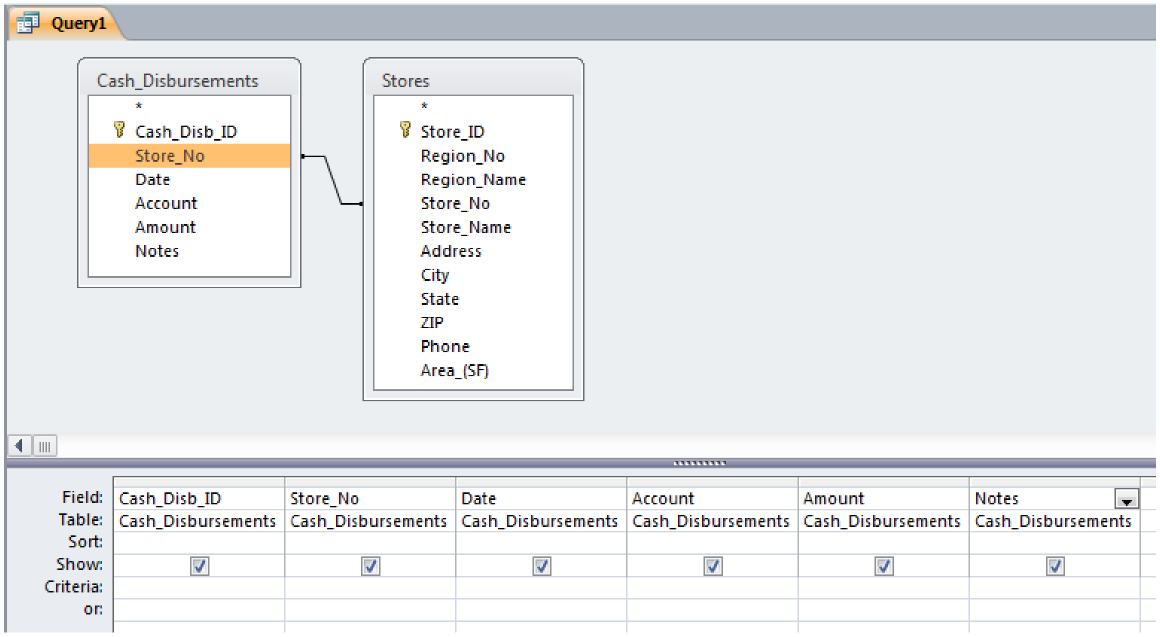
Stores: *, Store_ID, Region_No, Region_Name, Store_No, Store_Name, Address, City, State, ZIP, Phone, Area_(SF)

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.*



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 fields 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_2010 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.

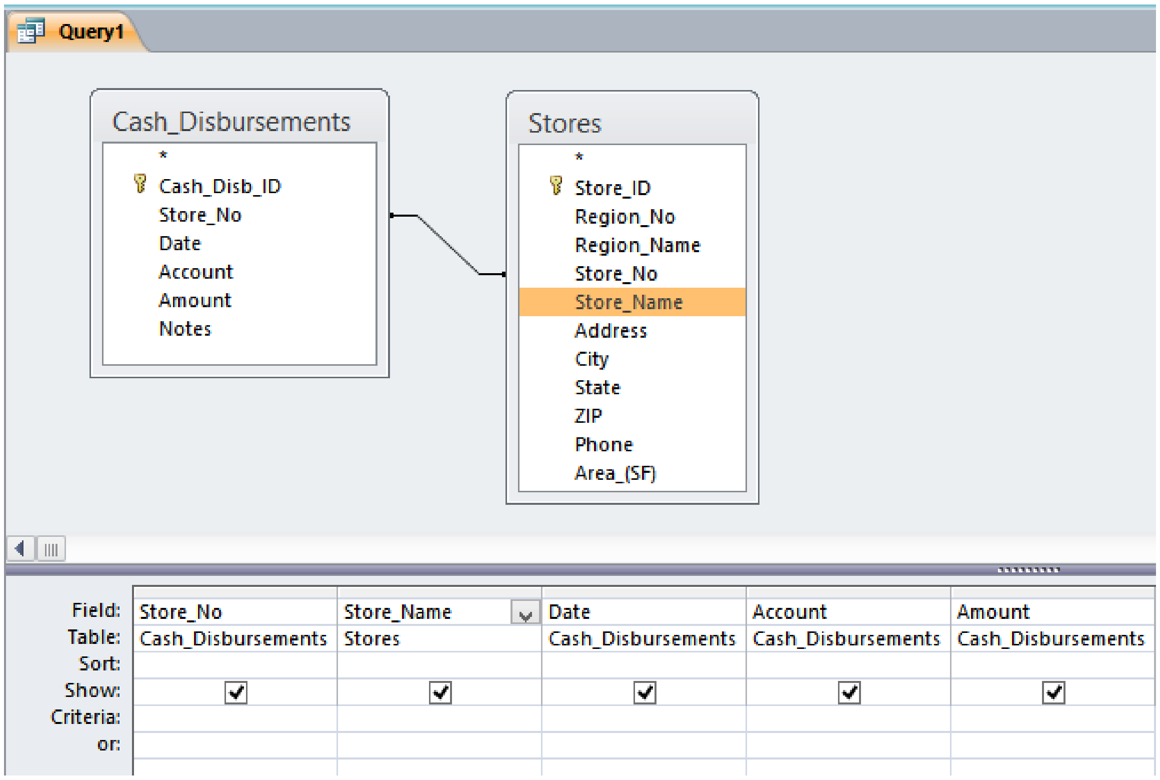
Another good argument for storing data in separate tables involves updating the data. Let's suppose the area code for the phone number at Store No. 1001 changed. That can happen quite often, particularly in large metropolitan areas. Particularly with the advent of cell phones, area codes are continuously 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_2010 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.

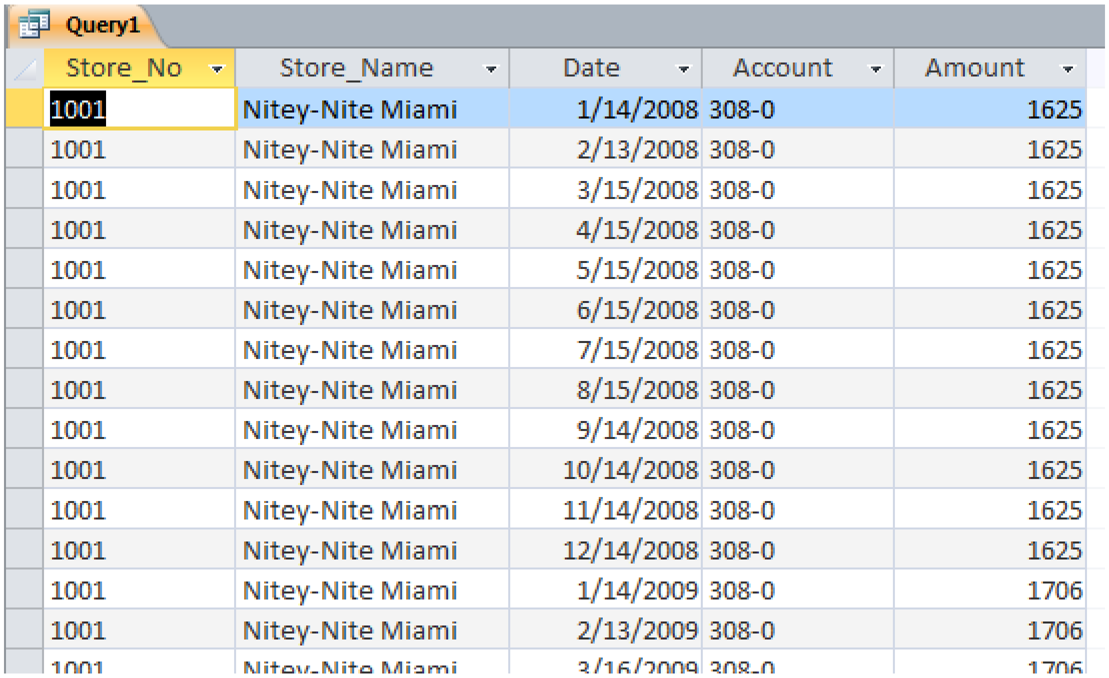
14. *In the* **Design grid** *of* **Query1**, *delete the* **Cash_Disb_ID** *and* **Notes** *fields.*
15. *Click and drag the* **Store_Name** *field from the* **Stores** *table and place it on top of the* **Date** *field in the* **Design grid** *and release.*



Field:	Store_No	Store_Name	Date	Account	Amount
Table:	Cash_Disbursements	Stores	Cash_Disbursements	Cash_Disbursements	Cash_Disbursements
Sort:					
Show:	✓	✓	✓	✓	✓
Criteria:					
or:					

Figure 3.9

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



Store_No	Store_Name	Date	Account	Amount
1001	Nitey-Nite Miami	1/14/2008	308-0	1625
1001	Nitey-Nite Miami	2/13/2008	308-0	1625
1001	Nitey-Nite Miami	3/15/2008	308-0	1625
1001	Nitey-Nite Miami	4/15/2008	308-0	1625
1001	Nitey-Nite Miami	5/15/2008	308-0	1625
1001	Nitey-Nite Miami	6/15/2008	308-0	1625
1001	Nitey-Nite Miami	7/15/2008	308-0	1625
1001	Nitey-Nite Miami	8/15/2008	308-0	1625
1001	Nitey-Nite Miami	9/14/2008	308-0	1625
1001	Nitey-Nite Miami	10/14/2008	308-0	1625
1001	Nitey-Nite Miami	11/14/2008	308-0	1625
1001	Nitey-Nite Miami	12/14/2008	308-0	1625
1001	Nitey-Nite Miami	1/14/2009	308-0	1706
1001	Nitey-Nite Miami	2/13/2009	308-0	1706
1001	Nitey-Nite Miami	3/16/2009	308-0	1706

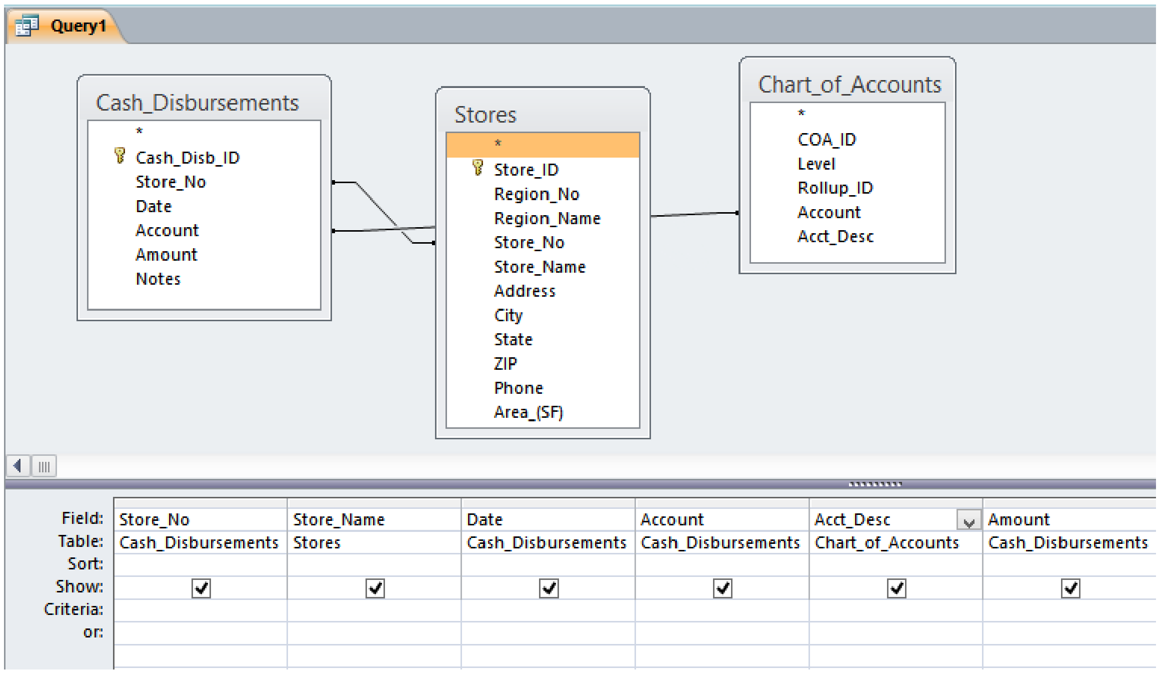
Figure 3.10

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:



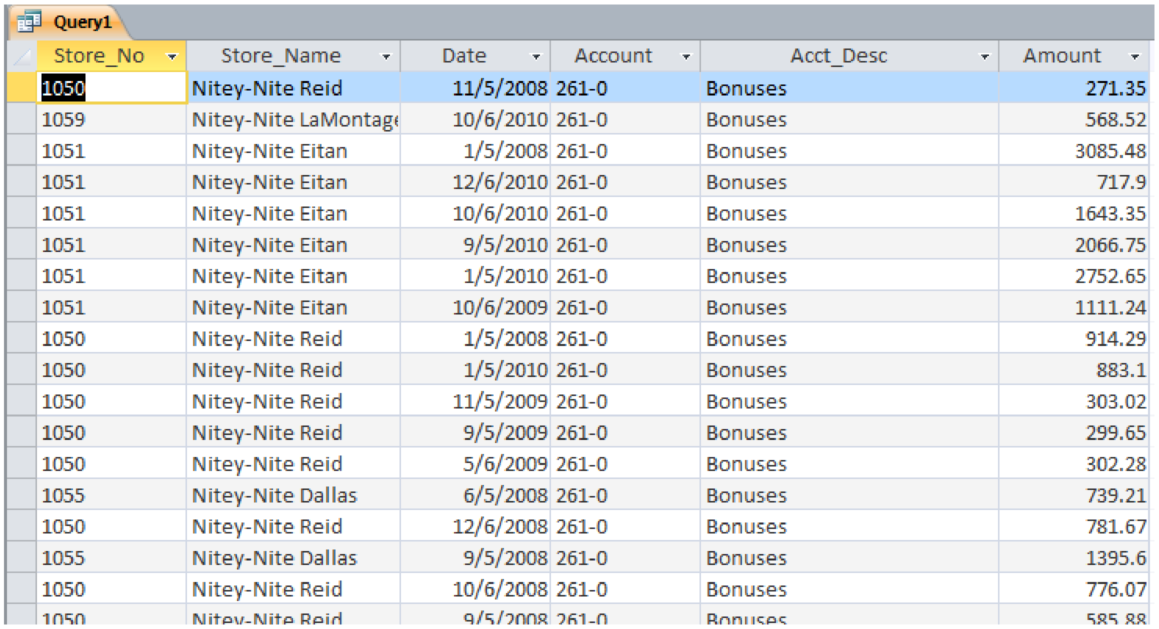
Field:	Store_No	Store_Name	Date	Account	Acct_Desc	Amount
Table:	Cash_Disbursements	Stores	Cash_Disbursements	Cash_Disbursements	Chart_of_Accounts	Cash_Disbursements
Sort:						
Show:	✓	✓	✓	✓	✓	✓
Criteria:						
or:						

Figure 3.11

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.


Query1

Store_No	Store_Name	Date	Account	Acct_Desc	Amount
1050	Nitey-Nite Reid	11/5/2008	261-0	Bonuses	271.35
1059	Nitey-Nite LaMontage	10/6/2010	261-0	Bonuses	568.52
1051	Nitey-Nite Eitan	1/5/2008	261-0	Bonuses	3085.48
1051	Nitey-Nite Eitan	12/6/2010	261-0	Bonuses	717.9
1051	Nitey-Nite Eitan	10/6/2010	261-0	Bonuses	1643.35
1051	Nitey-Nite Eitan	9/5/2010	261-0	Bonuses	2066.75
1051	Nitey-Nite Eitan	1/5/2010	261-0	Bonuses	2752.65
1051	Nitey-Nite Eitan	10/6/2009	261-0	Bonuses	1111.24
1050	Nitey-Nite Reid	1/5/2008	261-0	Bonuses	914.29
1050	Nitey-Nite Reid	1/5/2010	261-0	Bonuses	883.1
1050	Nitey-Nite Reid	11/5/2009	261-0	Bonuses	303.02
1050	Nitey-Nite Reid	9/5/2009	261-0	Bonuses	299.65
1050	Nitey-Nite Reid	5/6/2009	261-0	Bonuses	302.28
1055	Nitey-Nite Dallas	6/5/2008	261-0	Bonuses	739.21
1050	Nitey-Nite Reid	12/6/2008	261-0	Bonuses	781.67
1055	Nitey-Nite Dallas	9/5/2008	261-0	Bonuses	1395.6
1050	Nitey-Nite Reid	10/6/2008	261-0	Bonuses	776.07
1050	Nitey-Nite Reid	9/5/2008	261-0	Bonuses	585.88

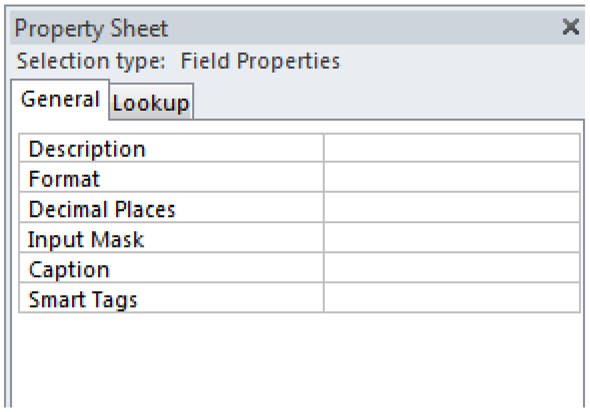
Figure 3.12

Formatting Query Fields

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.

Now let's suppose you want to see all of the Cash Disbursements transactions for Store No. 1021 in the month of June 2010. 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** *icon in* **Show/Hide** *group of the* **Query Tools**, **Design** *tab.*



Property Sheet

Selection type: Field Properties

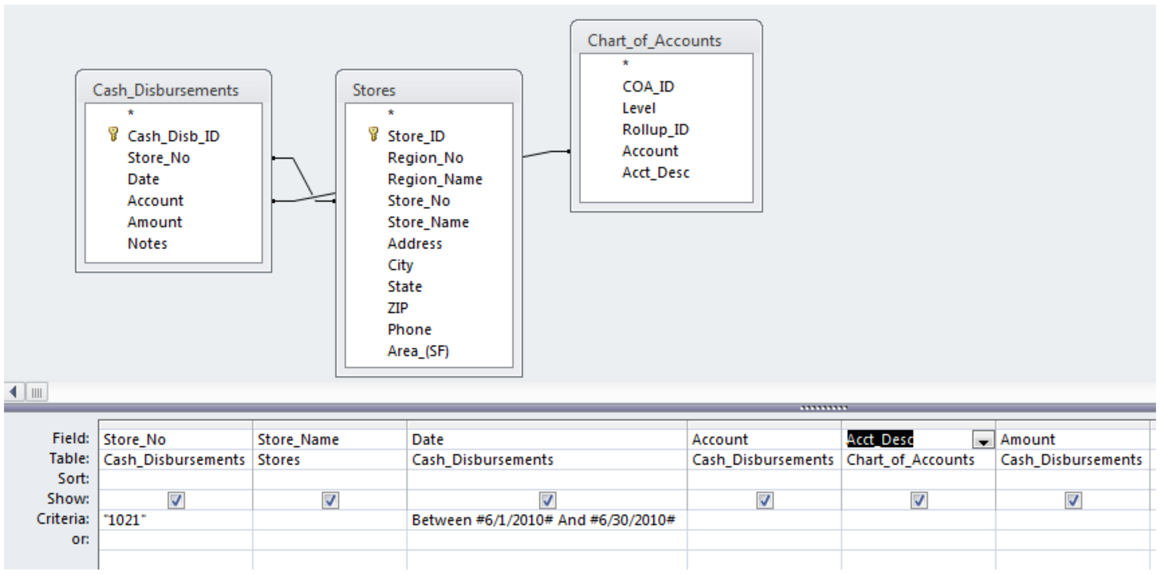
General | Lookup

Description	
Format	
Decimal Places	
Input Mask	
Caption	
Smart Tags	

Figure 3.13

The **Property Sheet** dialog box appears.

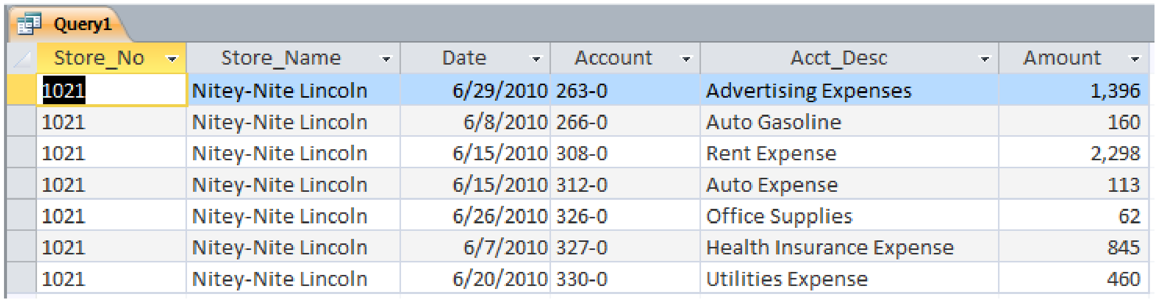
24. *Click on the* **Format** *line, and from the drop-down menu, choose* **Standard**.
25. *In the* **Decimal Places** *box, type* **0**.
26. *Close the* **Property Sheet** *box and input a criteria to filter for* **Store No. 1021** *with dates between* **6/1/2010** *and* **6/30/2010**.
27. *Adjust 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 #6/1/2010# And #6/30/2010#			
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	6/29/2010	263-0	Advertising Expenses	1,396
1021	Nitey-Nite Lincoln	6/8/2010	266-0	Auto Gasoline	160
1021	Nitey-Nite Lincoln	6/15/2010	308-0	Rent Expense	2,298
1021	Nitey-Nite Lincoln	6/15/2010	312-0	Auto Expense	113
1021	Nitey-Nite Lincoln	6/26/2010	326-0	Office Supplies	62
1021	Nitey-Nite Lincoln	6/7/2010	327-0	Health Insurance Expense	845
1021	Nitey-Nite Lincoln	6/20/2010	330-0	Utilities Expense	460

Figure 3.15

Naming an Alias

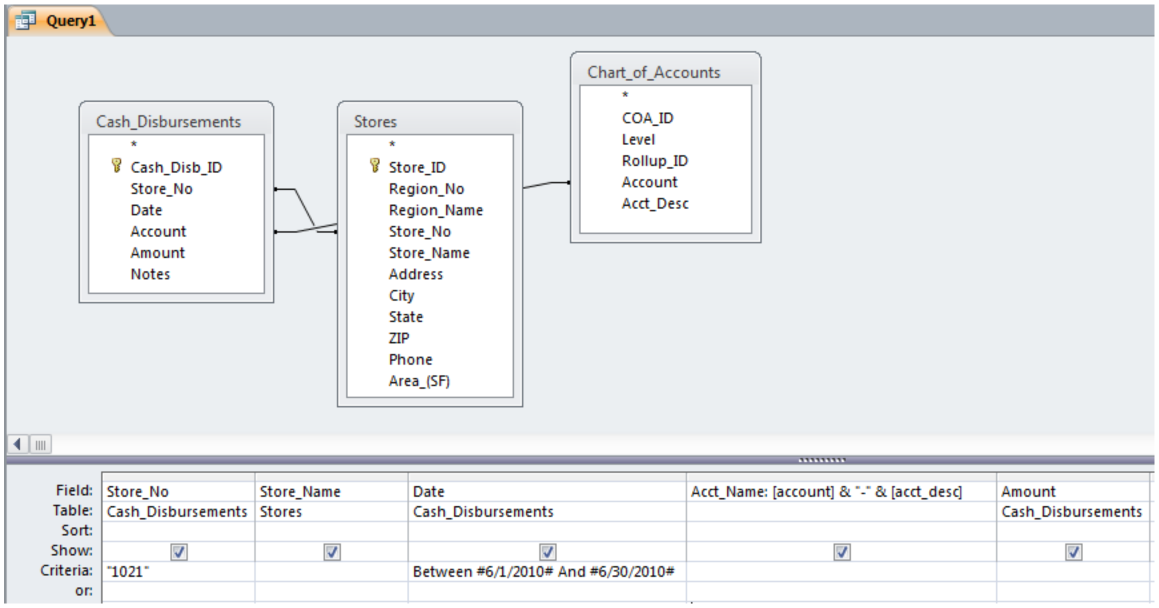
In Nitey-Nite's detailed financial statements, management likes to see the account and the name of the account in the same field, separated by a space, a dash, and another space. Just like in Excel, you can combine, or concatenate, fields in Access. When you do this in Access, you have to create a new 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) 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**.

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**.

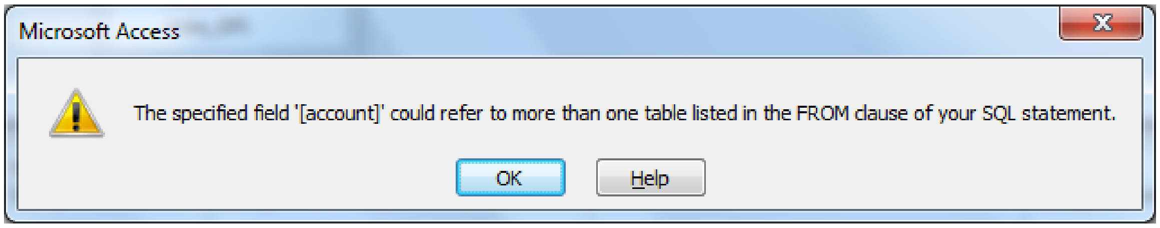
The design grid should look like this:



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 #6/1/2010# And #6/30/2010#		
or:					

Figure 3.16

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

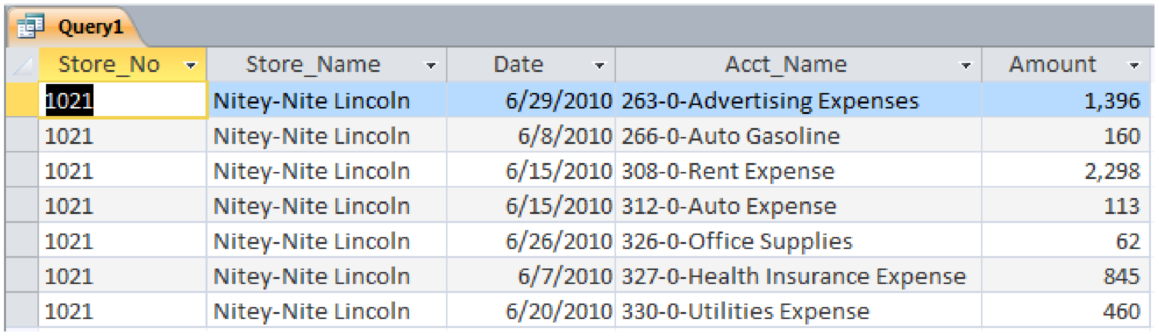


The specified field '[account]' could refer to more than one table listed in the FROM clause of your SQL statement.

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.



Query1

Store_No	Store_Name	Date	Acct_Name	Amount
1021	Nitey-Nite Lincoln	6/29/2010	263-0-Advertising Expenses	1,396
1021	Nitey-Nite Lincoln	6/8/2010	266-0-Auto Gasoline	160
1021	Nitey-Nite Lincoln	6/15/2010	308-0-Rent Expense	2,298
1021	Nitey-Nite Lincoln	6/15/2010	312-0-Auto Expense	113
1021	Nitey-Nite Lincoln	6/26/2010	326-0-Office Supplies	62
1021	Nitey-Nite Lincoln	6/7/2010	327-0-Health Insurance Expense	845
1021	Nitey-Nite Lincoln	6/20/2010	330-0-Utilities Expense	460

Figure 3.18

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

38. *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.

39. *In the* **Queries** *section, right-click* **qry03Cash_Disb** *and choose* **Design View**.

You return to the Design View of the query.

40. *Click on the* **Amount** *field, click on the* **Property Sheet** *icon and change the* **Decimal Places** *to* **2**.
41. *Close the* **Property Sheet** *dialog box.*
42. *Run the query to make sure the* **Amount** *field appears with* **two decimal places.**
43. *Save and close the 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 2010 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.

1. *Click on the* **Database Tools** *tab and click on the* **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.*

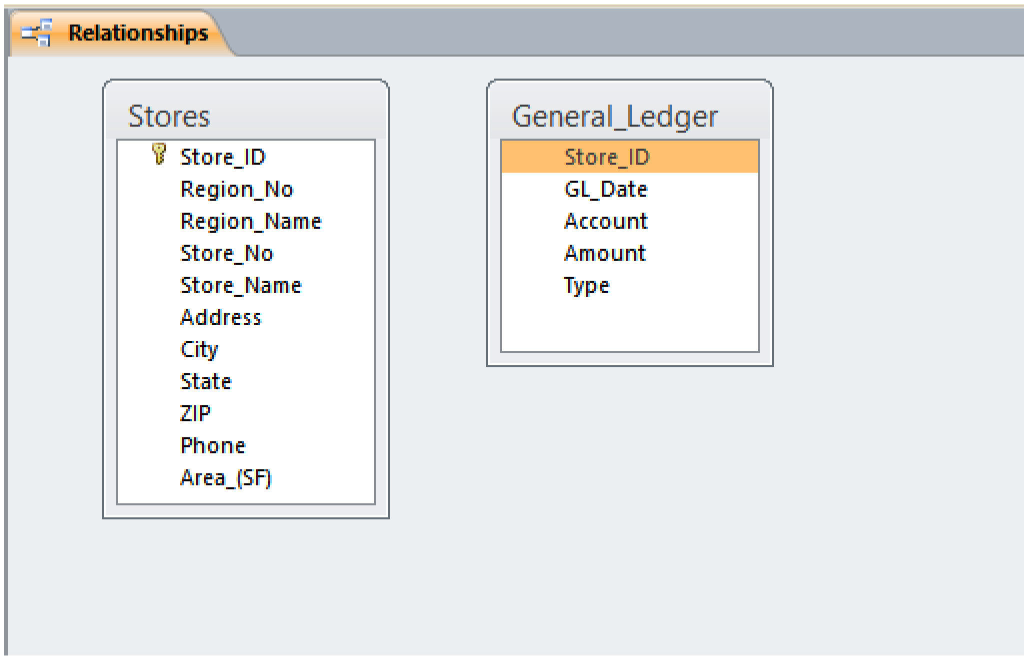


Figure 3.19

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.*

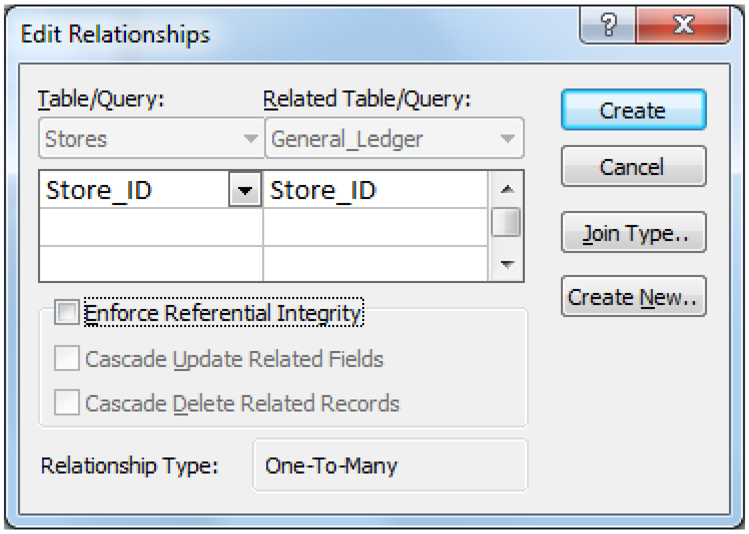


Figure 3.20

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**.

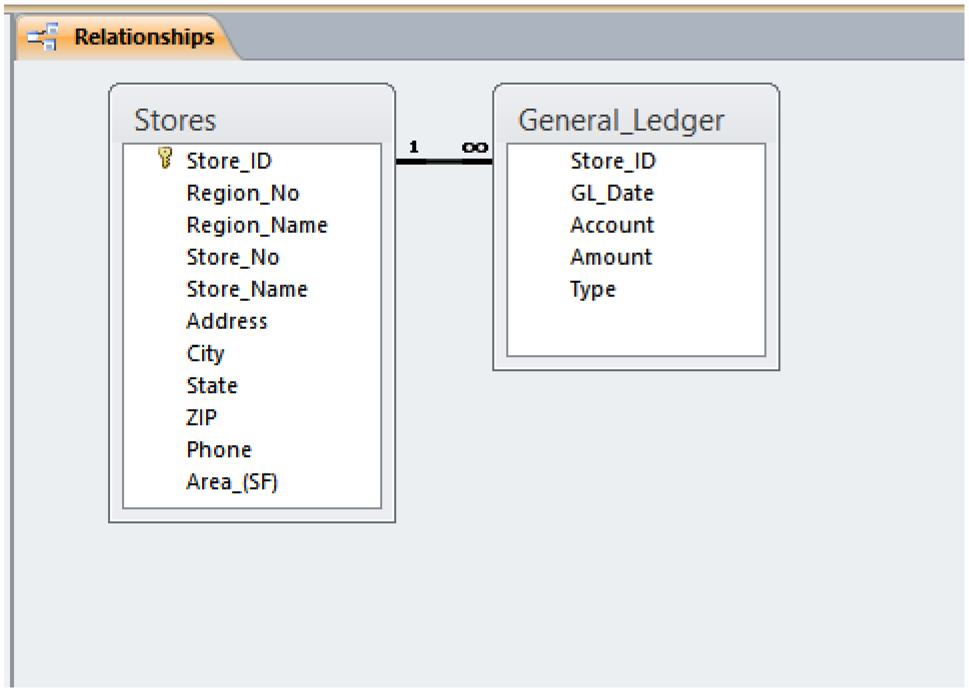


Figure 3.21

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.

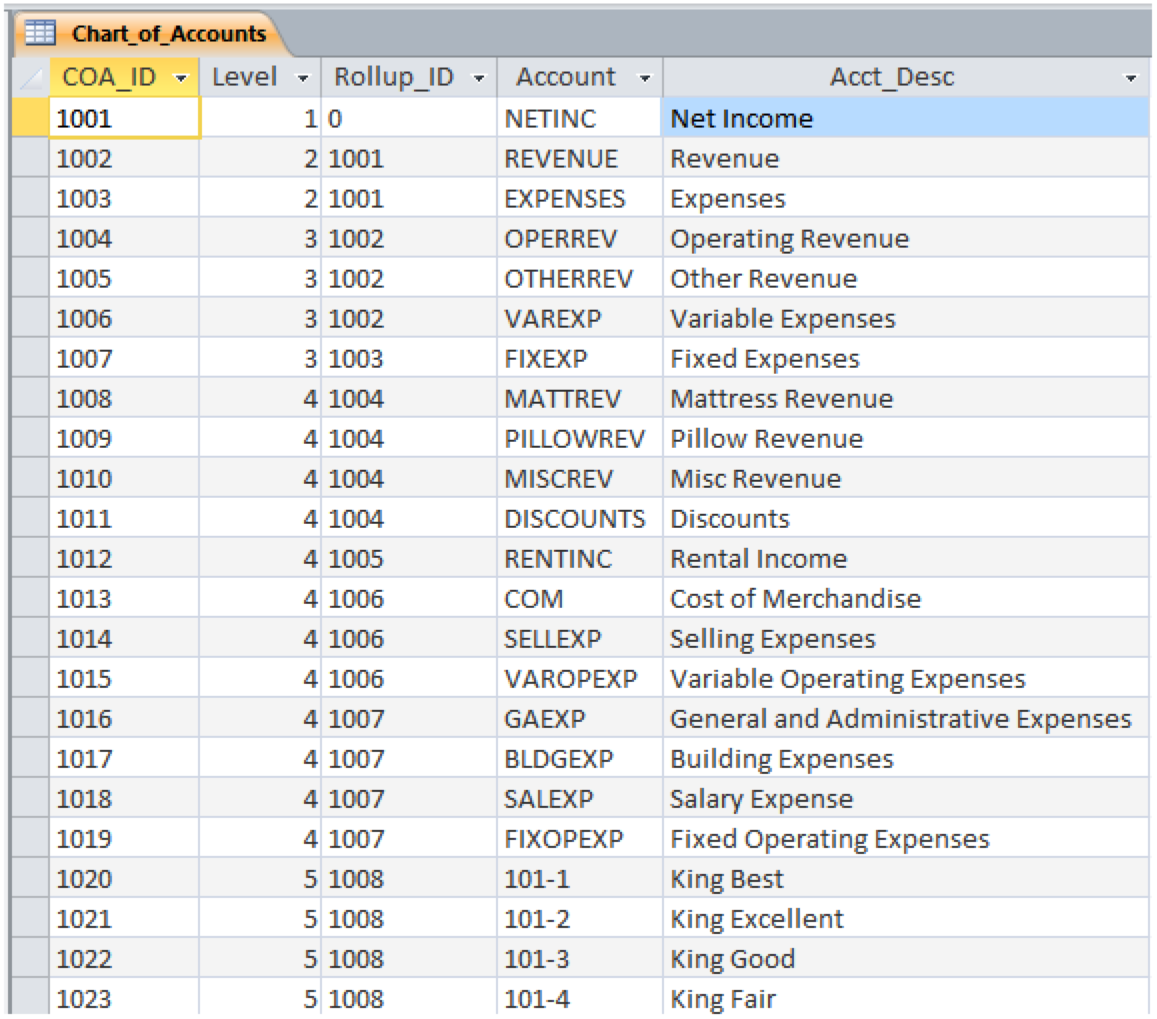
7. *Close the* **Relationships** *window, and choose* **Yes** *to save changes.*

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

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 1001-001, Cash. 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.

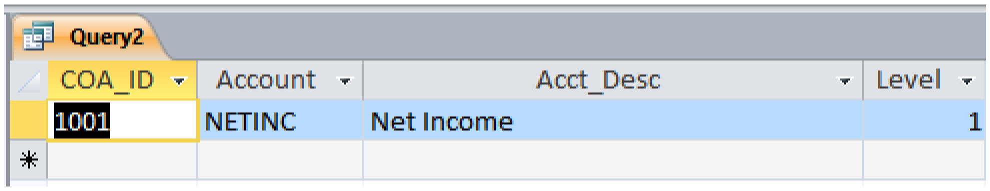
All Accounts in the Chart_of_Accounts table roll up to one higher level, except for the top level, Net Income (Level 1). The Level field is provided as a convenience to show the user which level the account or category is on, and we will use it in the next exercise. It is critically important to understand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

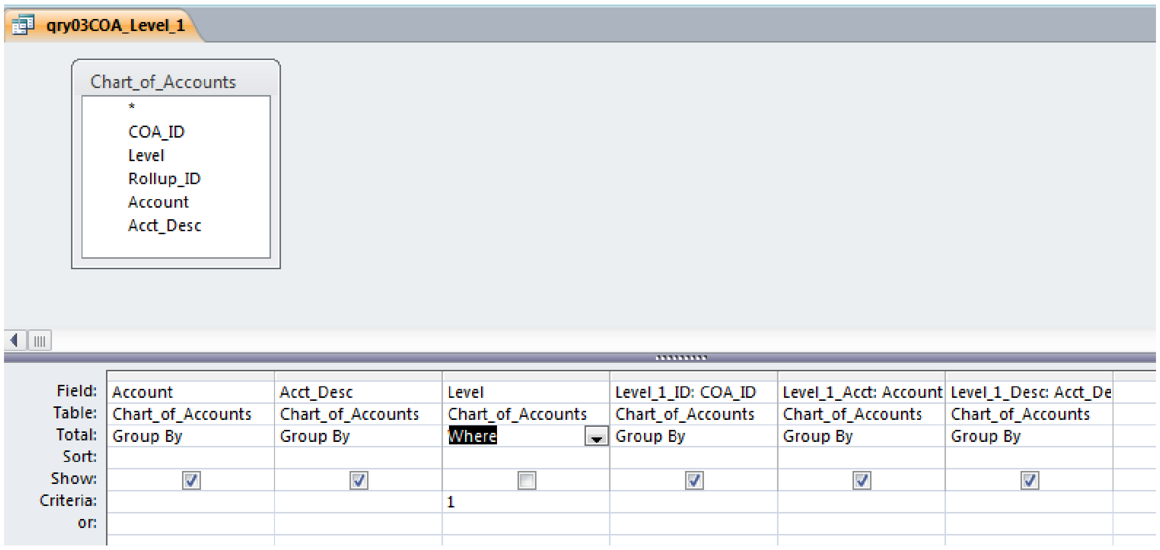
Level_1_ID: COA_ID
Level_1_Acct: Account
Level_1_Desc: Acct_Desc

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** *icon in the* **Query Tools**, **Design** *tab.*

A new line appears beneath the Table line called Total.

10. *Choose* **Where** *on the* **Total** *line for the* **Level** *field.*
11. *Save the query and name it* **qry03COA_Level_1**.

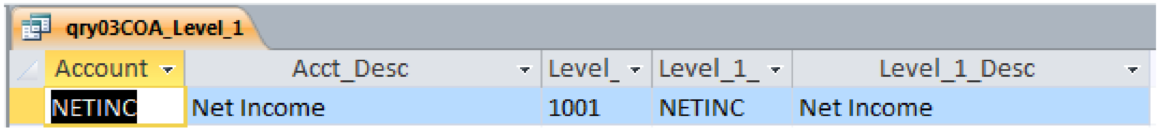


Field:	Account	Acct_Desc	Level	Level_1_ID: COA_ID	Level_1_Acct: Account	Level_1_Desc: Acct_De
Table:	Chart_of_Accounts	Chart_of_Accounts	Chart_of_Accounts	Chart_of_Accounts	Chart_of_Accounts	Chart_of_Accounts
Total:	Group By	Group By	Where	Group By	Group By	Group By
Sort:						
Show:	✓	✓	☐	✓	✓	✓
Criteria:			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.*



Account	Acct_Desc	Level_	Level_1_	Level_1_Desc
NETINC	Net Income	1001	NETINC	Net Income

Figure 3.25

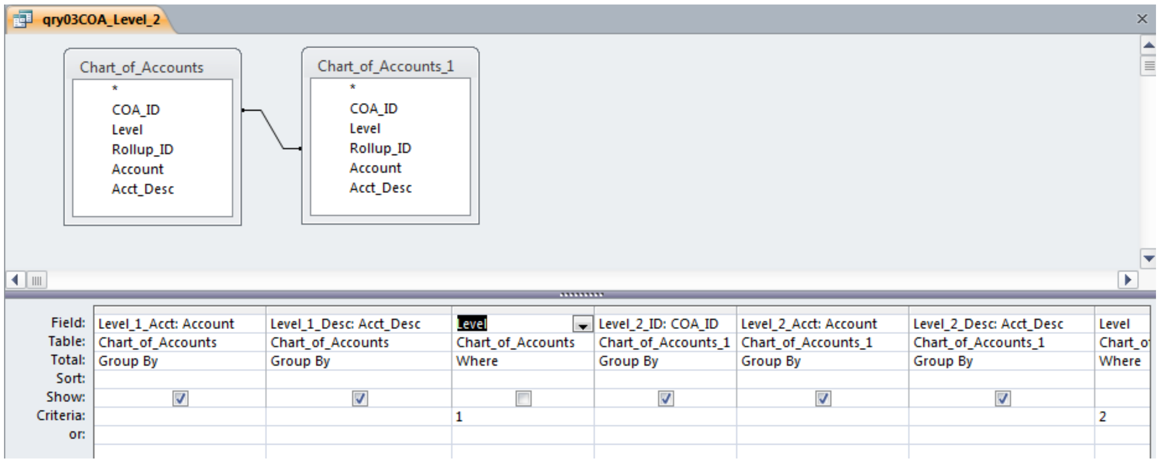
Joins on Multiple Tables

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_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**

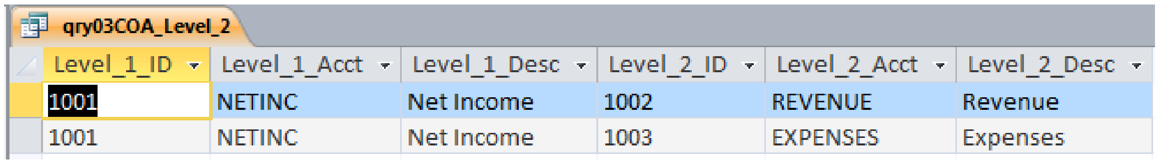


Field:	Level_1_Acct: Account	Level_1_Desc: Acct_Desc	Level	Level_2_ID: COA_ID	Level_2_Acct: Account	Level_2_Desc: Acct_Desc	Level
Table:	Chart_of_Accounts	Chart_of_Accounts	Chart_of_Accounts	Chart_of_Accounts_1	Chart_of_Accounts_1	Chart_of_Accounts_1	Chart_o
Total:	Group By	Group By	Where	Group By	Group By	Group By	Where
Sort:							
Show:	✓	✓	☐	✓	✓	✓	
Criteria:			1				2
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.*

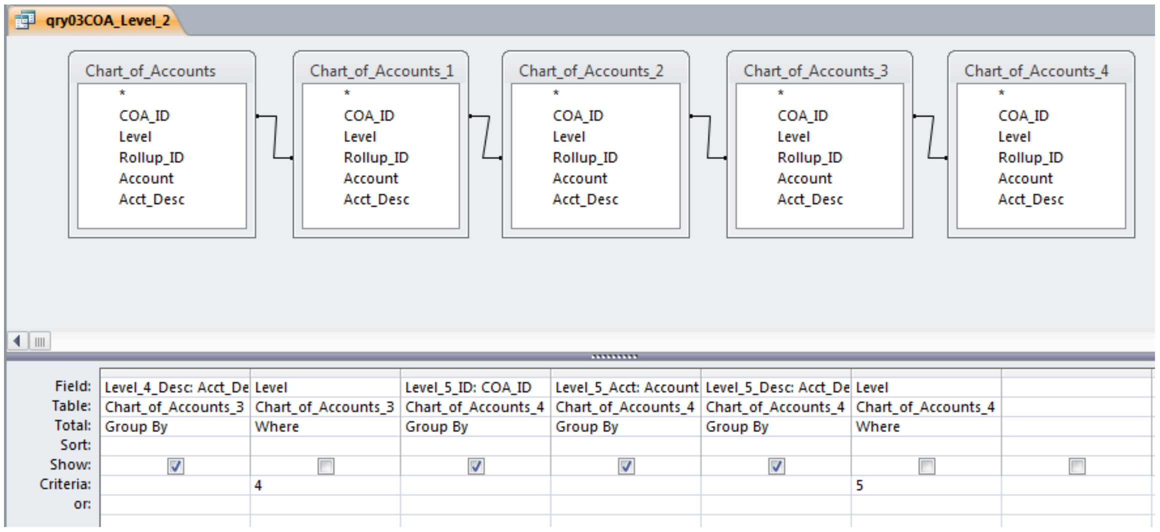


qry03COA_Level_2

Level_1_ID	Level_1_Acct	Level_1_Desc	Level_2_ID	Level_2_Acct	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***.*

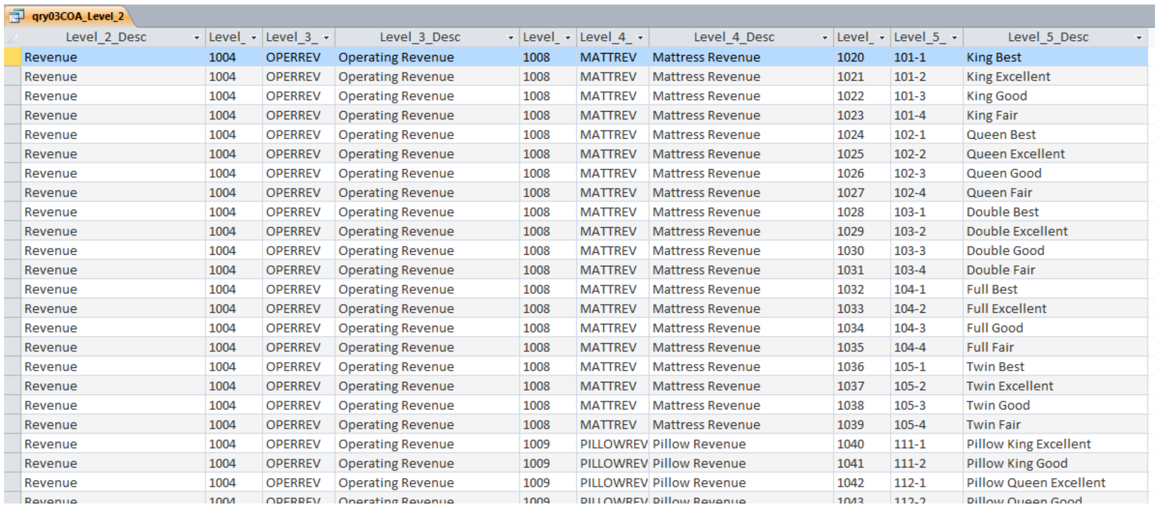


Field:	Level_4_Desc: Acct_De	Level	Level_5_ID: COA_ID	Level_5_Acct: Account	Level_5_Desc: Acct_De	Level	
Table:	Chart_of_Accounts_3	Chart_of_Accounts_3	Chart_of_Accounts_4	Chart_of_Accounts_4	Chart_of_Accounts_4	Chart_of_Accounts_4	
Total:	Group By	Where	Group By	Group By	Group By	Where	
Sort:							
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:



qry03COA_Level_2

Level_2_Desc	Level_	Level_3_	Level_3_Desc	Level_	Level_4_	Level_4_Desc	Level_	Level_5_	Level_5_Desc
Revenue	1004	OPERREV	Operating Revenue	1008	MATTREV	Mattress Revenue	1020	101-1	King Best
Revenue	1004	OPERREV	Operating Revenue	1008	MATTREV	Mattress Revenue	1021	101-2	King Excellent
Revenue	1004	OPERREV	Operating Revenue	1008	MATTREV	Mattress Revenue	1022	101-3	King Good
Revenue	1004	OPERREV	Operating Revenue	1008	MATTREV	Mattress Revenue	1023	101-4	King Fair
Revenue	1004	OPERREV	Operating Revenue	1008	MATTREV	Mattress Revenue	1024	102-1	Queen Best
Revenue	1004	OPERREV	Operating Revenue	1008	MATTREV	Mattress Revenue	1025	102-2	Queen Excellent
Revenue	1004	OPERREV	Operating Revenue	1008	MATTREV	Mattress Revenue	1026	102-3	Queen Good
Revenue	1004	OPERREV	Operating Revenue	1008	MATTREV	Mattress Revenue	1027	102-4	Queen Fair
Revenue	1004	OPERREV	Operating Revenue	1008	MATTREV	Mattress Revenue	1028	103-1	Double Best
Revenue	1004	OPERREV	Operating Revenue	1008	MATTREV	Mattress Revenue	1029	103-2	Double Excellent
Revenue	1004	OPERREV	Operating Revenue	1008	MATTREV	Mattress Revenue	1030	103-3	Double Good
Revenue	1004	OPERREV	Operating Revenue	1008	MATTREV	Mattress Revenue	1031	103-4	Double Fair
Revenue	1004	OPERREV	Operating Revenue	1008	MATTREV	Mattress Revenue	1032	104-1	Full Best
Revenue	1004	OPERREV	Operating Revenue	1008	MATTREV	Mattress Revenue	1033	104-2	Full Excellent
Revenue	1004	OPERREV	Operating Revenue	1008	MATTREV	Mattress Revenue	1034	104-3	Full Good
Revenue	1004	OPERREV	Operating Revenue	1008	MATTREV	Mattress Revenue	1035	104-4	Full Fair
Revenue	1004	OPERREV	Operating Revenue	1008	MATTREV	Mattress Revenue	1036	105-1	Twin Best
Revenue	1004	OPERREV	Operating Revenue	1008	MATTREV	Mattress Revenue	1037	105-2	Twin Excellent
Revenue	1004	OPERREV	Operating Revenue	1008	MATTREV	Mattress Revenue	1038	105-3	Twin Good
Revenue	1004	OPERREV	Operating Revenue	1008	MATTREV	Mattress Revenue	1039	105-4	Twin Fair
Revenue	1004	OPERREV	Operating Revenue	1009	PILLOWREV	Pillow Revenue	1040	111-1	Pillow King Excellent
Revenue	1004	OPERREV	Operating Revenue	1009	PILLOWREV	Pillow Revenue	1041	111-2	Pillow King Good
Revenue	1004	OPERREV	Operating Revenue	1009	PILLOWREV	Pillow Revenue	1042	112-1	Pillow Queen Excellent
Revenue	1004	OPERREV	Operating Revenue	1009	PILLOWREV	Pillow Revenue	1043	112-2	Pillow Queen Good

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 2010 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, log in and take the exam. Make sure that you take the exam on the same computer that you completed the sample files on, as some of the questions on the exam may refer to some of the completed examples.

ExcelCEO

Chief Excel Officer

Access 2010 and SQL

Complete Self-study Course

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.

Advanced Queries

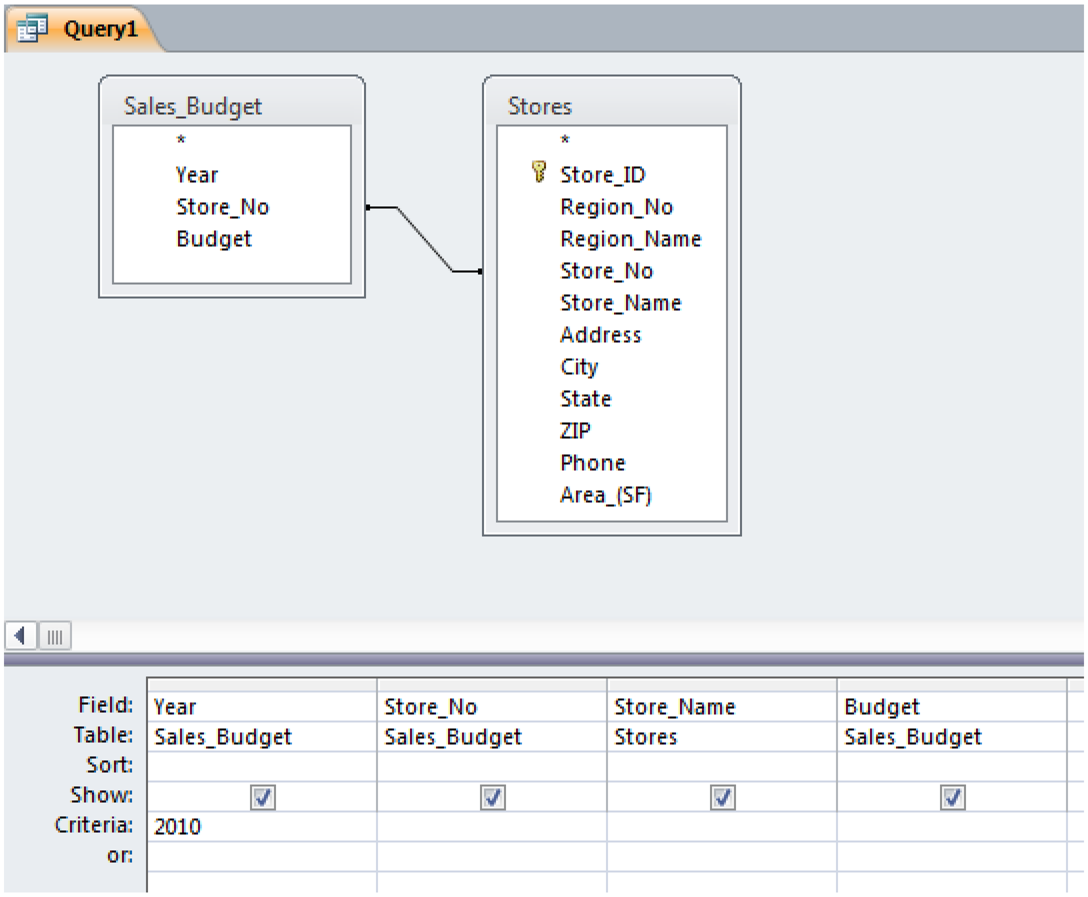
A co-worker and I once took a SQL Server programming course. In the course, the instructor was talking about left and right joins, sometimes referred to as an outer or one-way join, to be discussed later in this chapter. He told us that you should never use more than two one-way joins in a query. When 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.

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.

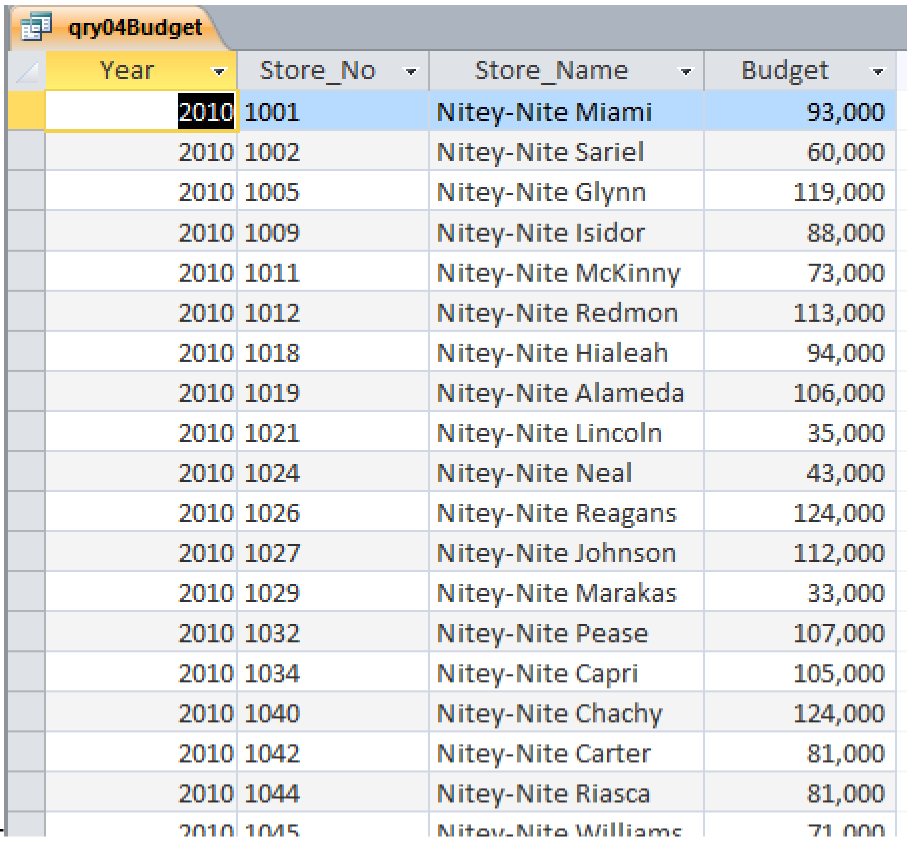
1. *Open the* **Nitey_Nite_2010** *database.*
2. *Create a query that uses the* **Sales_Budget** *and* **Stores** *tables, create a join on* **Store_No**, *and bring the* **Year**, **Store_No**, **Store_Name** *and* **Budget** *fields into the query design grid.*
3. *Filter the data for the* **Year 2010**.



Field:	Year	Store_No	Store_Name	Budget
Table:	Sales_Budget	Sales_Budget	Stores	Sales_Budget
Sort:				
Show:	✓	✓	✓	✓
Criteria:	2010			
or:				

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.*



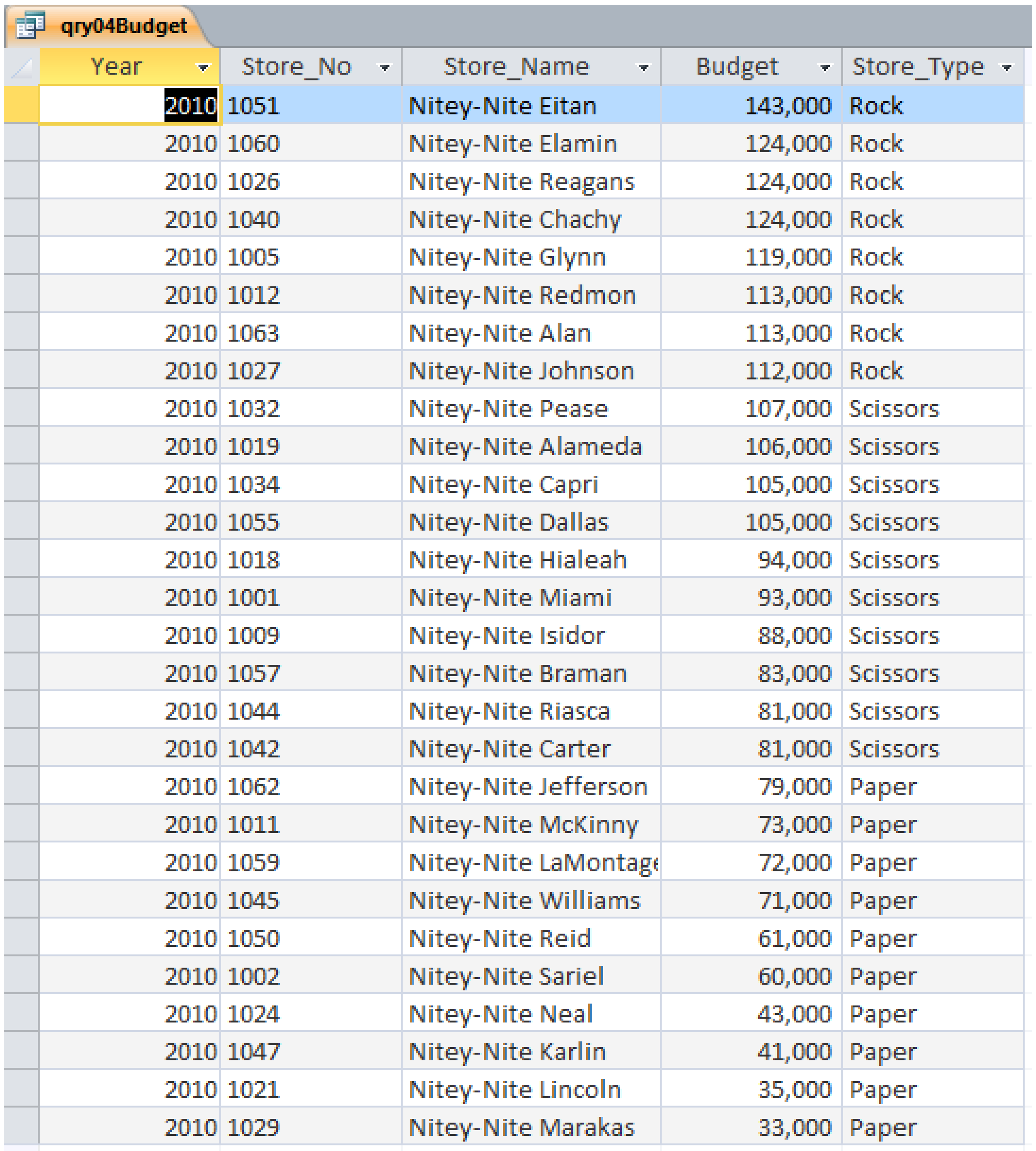
qry04Budget

Year	Store_No	Store_Name	Budget
2010	1001	Nitey-Nite Miami	93,000
2010	1002	Nitey-Nite Sariel	60,000
2010	1005	Nitey-Nite Glynn	119,000
2010	1009	Nitey-Nite Isidor	88,000
2010	1011	Nitey-Nite McKinny	73,000
2010	1012	Nitey-Nite Redmon	113,000
2010	1018	Nitey-Nite Hialeah	94,000
2010	1019	Nitey-Nite Alameda	106,000
2010	1021	Nitey-Nite Lincoln	35,000
2010	1024	Nitey-Nite Neal	43,000
2010	1026	Nitey-Nite Reagans	124,000
2010	1027	Nitey-Nite Johnson	112,000
2010	1029	Nitey-Nite Marakas	33,000
2010	1032	Nitey-Nite Pease	107,000
2010	1034	Nitey-Nite Capri	105,000
2010	1040	Nitey-Nite Chachy	124,000
2010	1042	Nitey-Nite Carter	81,000
2010	1044	Nitey-Nite Riasca	81,000
2010	1045	Nitey-Nite Williams	71,000

Figure 4.2

Now you have a simple query. It shows the sales budgets for each store in 2010. 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 the query and run it.*


qry04Budget

Year	Store_No	Store_Name	Budget	Store_Type
2010	1051	Nitey-Nite Eitan	143,000	Rock
2010	1060	Nitey-Nite Elamin	124,000	Rock
2010	1026	Nitey-Nite Reagans	124,000	Rock
2010	1040	Nitey-Nite Chachy	124,000	Rock
2010	1005	Nitey-Nite Glynn	119,000	Rock
2010	1012	Nitey-Nite Redmon	113,000	Rock
2010	1063	Nitey-Nite Alan	113,000	Rock
2010	1027	Nitey-Nite Johnson	112,000	Rock
2010	1032	Nitey-Nite Pease	107,000	Scissors
2010	1019	Nitey-Nite Alameda	106,000	Scissors
2010	1034	Nitey-Nite Capri	105,000	Scissors
2010	1055	Nitey-Nite Dallas	105,000	Scissors
2010	1018	Nitey-Nite Hialeah	94,000	Scissors
2010	1001	Nitey-Nite Miami	93,000	Scissors
2010	1009	Nitey-Nite Isidor	88,000	Scissors
2010	1057	Nitey-Nite Braman	83,000	Scissors
2010	1044	Nitey-Nite Riasca	81,000	Scissors
2010	1042	Nitey-Nite Carter	81,000	Scissors
2010	1062	Nitey-Nite Jefferson	79,000	Paper
2010	1011	Nitey-Nite McKinny	73,000	Paper
2010	1059	Nitey-Nite LaMontage	72,000	Paper
2010	1045	Nitey-Nite Williams	71,000	Paper
2010	1050	Nitey-Nite Reid	61,000	Paper
2010	1002	Nitey-Nite Sariel	60,000	Paper
2010	1024	Nitey-Nite Neal	43,000	Paper
2010	1047	Nitey-Nite Karlin	41,000	Paper
2010	1021	Nitey-Nite Lincoln	35,000	Paper
2010	1029	Nitey-Nite Marakas	33,000	Paper

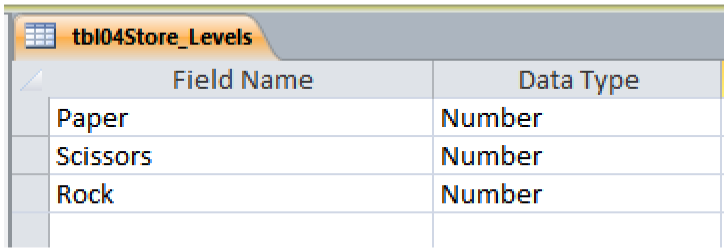
Figure 4.3

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

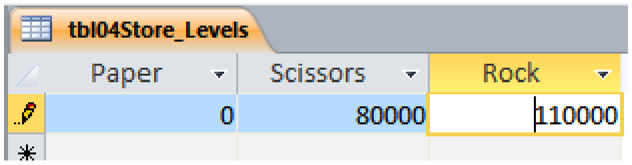
One of the problems in hard-coding the stores' types in a formula is that if the dollar amounts change, you would have to go into the formula and change it. In the Excel course, we solved that issue by having an Assumptions tab that contains all of the workbook's criteria and assumptions. In Access, 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:*



tbl04Store_Levels

Field Name	Data Type
Paper	Number
Scissors	Number
Rock	Number



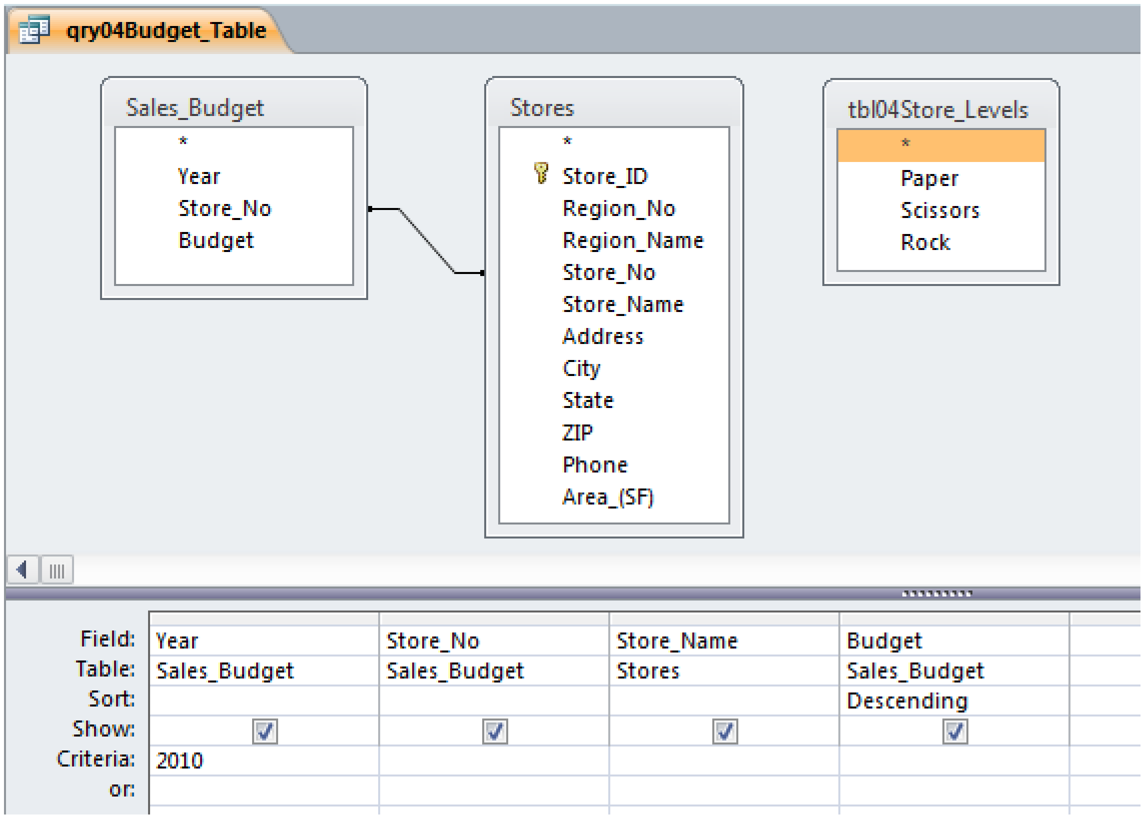
tbl04Store_Levels

Paper	Scissors	Rock
0	80000	110000

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.*

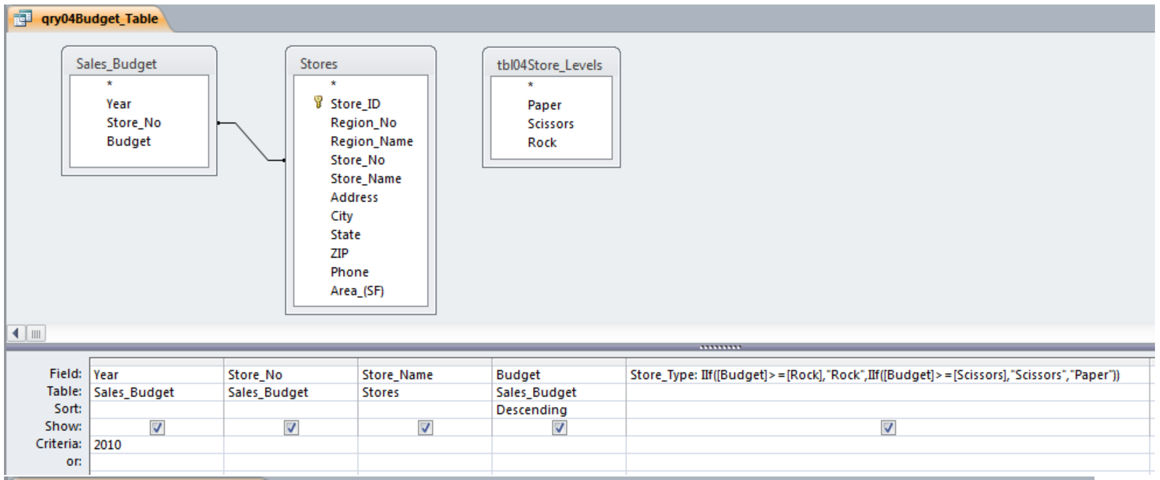


Field:	Year	Store_No	Store_Name	Budget
Table:	Sales_Budget	Sales_Budget	Stores	Sales_Budget
Sort:				Descending
Show:	✓	✓	✓	✓
Criteria:	2010			
or:				

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.*



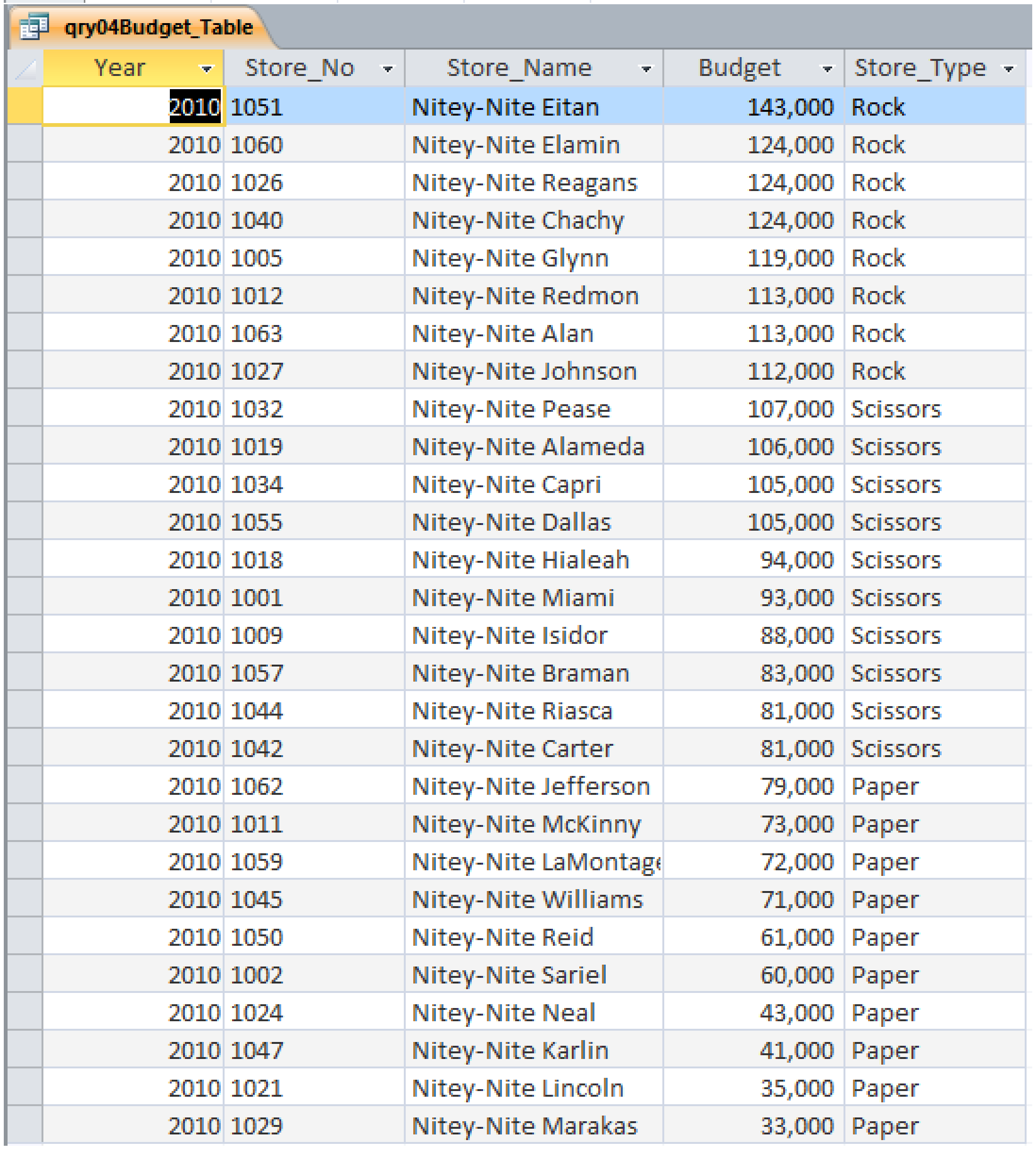
qry04Budget_Table

Sales_Budget: *, Year, Store_No, Budget

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

tbl04Store_Levels: *, Paper, Scissors, Rock

Field:	Year	Store_No	Store_Name	Budget	Store_Type: IIf([Budget]>=[Rock],"Rock",IIf([Budget]>=[Scissors],"Scissors","Paper"))
Table:	Sales_Budget	Sales_Budget	Stores	Sales_Budget	
Sort:				Descending	
Show:	✓	✓	✓	✓	✓
Criteria:	2010				
or:					



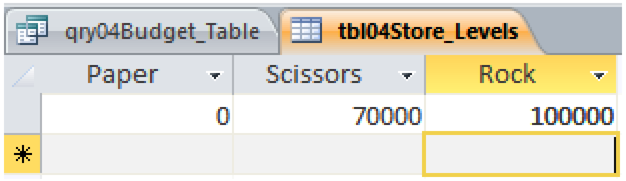
qry04Budget_Table

Year	Store_No	Store_Name	Budget	Store_Type
2010	1051	Nitey-Nite Eitan	143,000	Rock
2010	1060	Nitey-Nite Elamin	124,000	Rock
2010	1026	Nitey-Nite Reagans	124,000	Rock
2010	1040	Nitey-Nite Chachy	124,000	Rock
2010	1005	Nitey-Nite Glynn	119,000	Rock
2010	1012	Nitey-Nite Redmon	113,000	Rock
2010	1063	Nitey-Nite Alan	113,000	Rock
2010	1027	Nitey-Nite Johnson	112,000	Rock
2010	1032	Nitey-Nite Pease	107,000	Scissors
2010	1019	Nitey-Nite Alameda	106,000	Scissors
2010	1034	Nitey-Nite Capri	105,000	Scissors
2010	1055	Nitey-Nite Dallas	105,000	Scissors
2010	1018	Nitey-Nite Hialeah	94,000	Scissors
2010	1001	Nitey-Nite Miami	93,000	Scissors
2010	1009	Nitey-Nite Isidor	88,000	Scissors
2010	1057	Nitey-Nite Braman	83,000	Scissors
2010	1044	Nitey-Nite Riasca	81,000	Scissors
2010	1042	Nitey-Nite Carter	81,000	Scissors
2010	1062	Nitey-Nite Jefferson	79,000	Paper
2010	1011	Nitey-Nite McKinny	73,000	Paper
2010	1059	Nitey-Nite LaMontag	72,000	Paper
2010	1045	Nitey-Nite Williams	71,000	Paper
2010	1050	Nitey-Nite Reid	61,000	Paper
2010	1002	Nitey-Nite Sariel	60,000	Paper
2010	1024	Nitey-Nite Neal	43,000	Paper
2010	1047	Nitey-Nite Karlin	41,000	Paper
2010	1021	Nitey-Nite Lincoln	35,000	Paper
2010	1029	Nitey-Nite Marakas	33,000	Paper

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 harder to set up 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**.

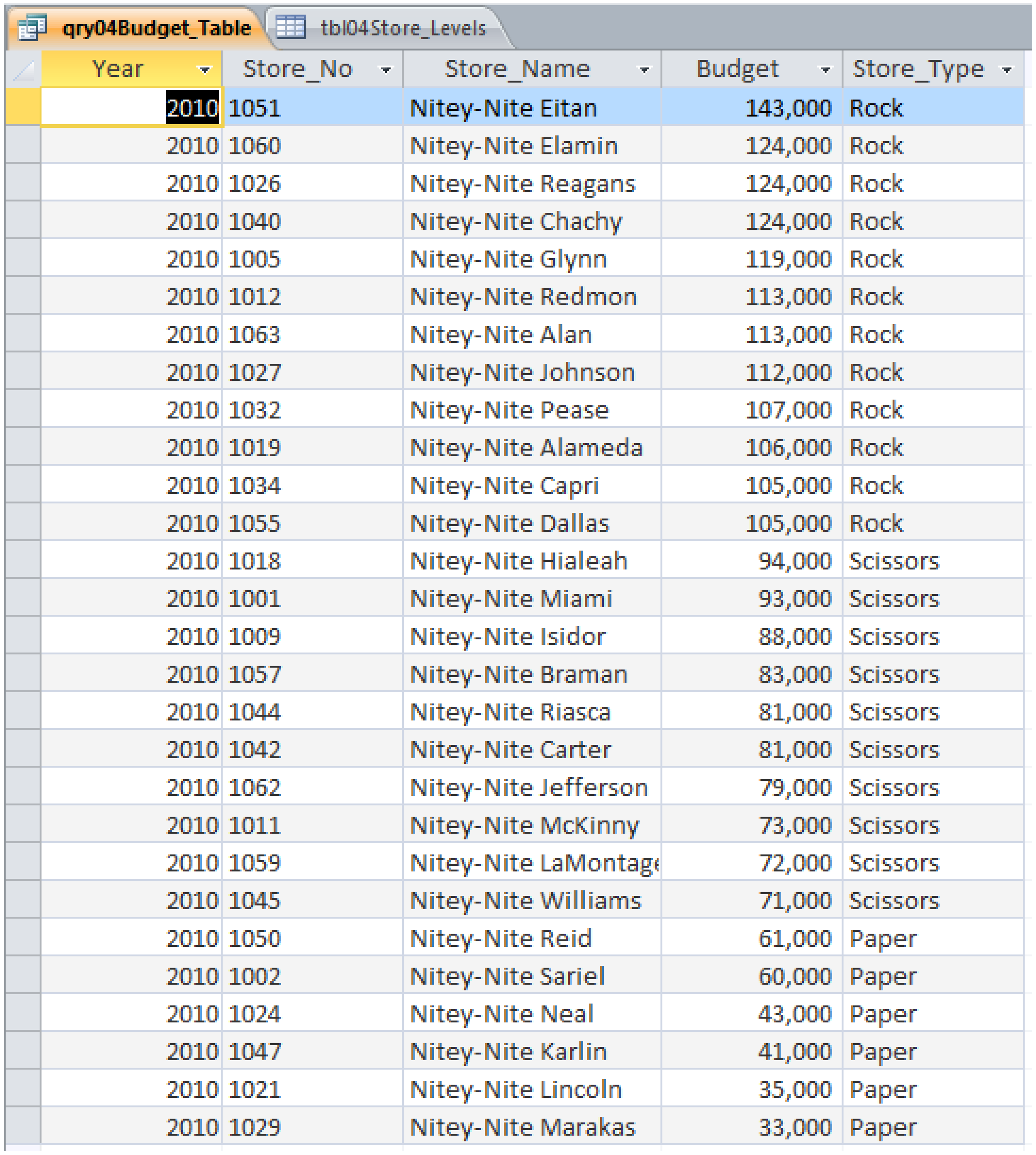


Paper	Scissors	Rock
0	70000	100000

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	Store_Name	Budget	Store_Type
2010	1051	Nitey-Nite Eitan	143,000	Rock
2010	1060	Nitey-Nite Elamin	124,000	Rock
2010	1026	Nitey-Nite Reagans	124,000	Rock
2010	1040	Nitey-Nite Chachy	124,000	Rock
2010	1005	Nitey-Nite Glynn	119,000	Rock
2010	1012	Nitey-Nite Redmon	113,000	Rock
2010	1063	Nitey-Nite Alan	113,000	Rock
2010	1027	Nitey-Nite Johnson	112,000	Rock
2010	1032	Nitey-Nite Pease	107,000	Rock
2010	1019	Nitey-Nite Alameda	106,000	Rock
2010	1034	Nitey-Nite Capri	105,000	Rock
2010	1055	Nitey-Nite Dallas	105,000	Rock
2010	1018	Nitey-Nite Hialeah	94,000	Scissors
2010	1001	Nitey-Nite Miami	93,000	Scissors
2010	1009	Nitey-Nite Isidor	88,000	Scissors
2010	1057	Nitey-Nite Braman	83,000	Scissors
2010	1044	Nitey-Nite Riasca	81,000	Scissors
2010	1042	Nitey-Nite Carter	81,000	Scissors
2010	1062	Nitey-Nite Jefferson	79,000	Scissors
2010	1011	Nitey-Nite McKinny	73,000	Scissors
2010	1059	Nitey-Nite LaMontage	72,000	Scissors
2010	1045	Nitey-Nite Williams	71,000	Scissors
2010	1050	Nitey-Nite Reid	61,000	Paper
2010	1002	Nitey-Nite Sariel	60,000	Paper
2010	1024	Nitey-Nite Neal	43,000	Paper
2010	1047	Nitey-Nite Karlin	41,000	Paper
2010	1021	Nitey-Nite Lincoln	35,000	Paper
2010	1029	Nitey-Nite Marakas	33,000	Paper

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.***

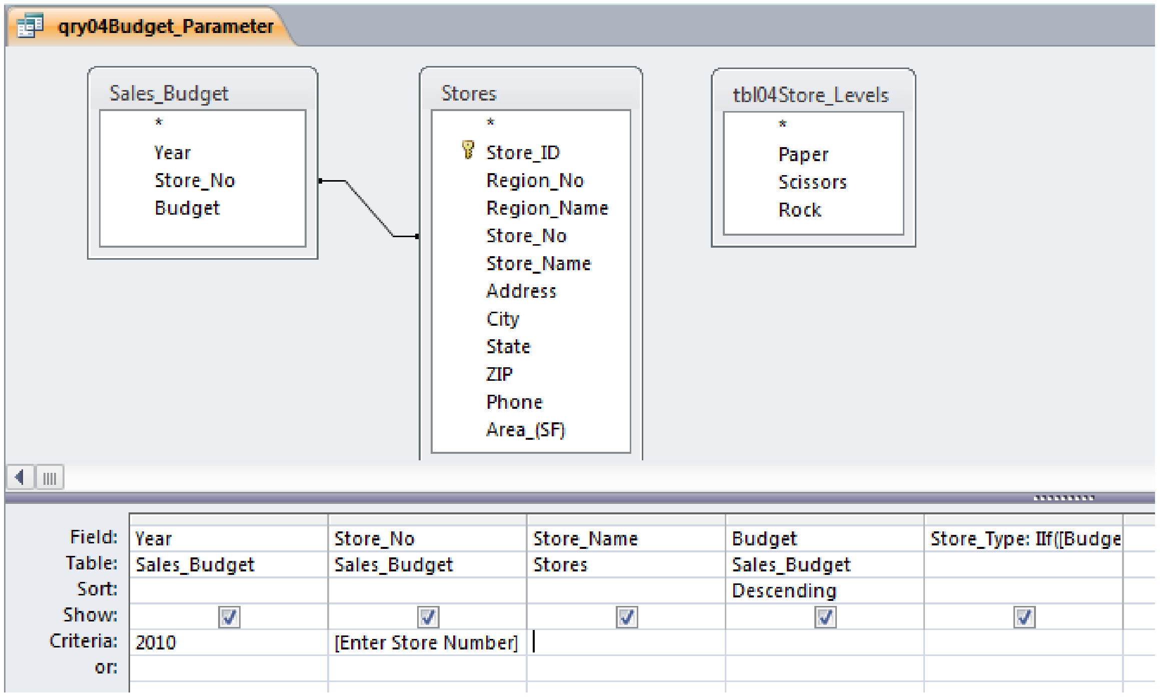
12. *Close the* **tbl04Store_Levels** *table and save and close* **qry04Budget_Table**.

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	Store_Name	Budget	Store_Type: IIf([Budge
Table:	Sales_Budget	Sales_Budget	Stores	Sales_Budget	
Sort:				Descending	
Show:	✓	✓	✓	✓	✓
Criteria:	2010	[Enter Store Number]			
or:					

Figure 4.9

3. *Run the query.*

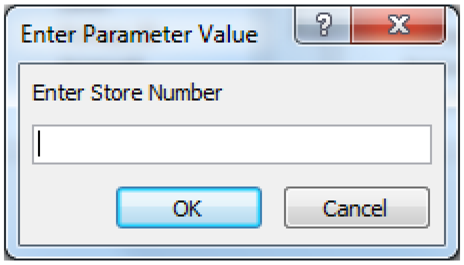
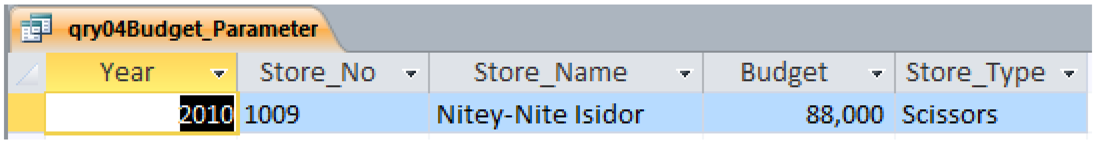


Figure 4.10

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**.



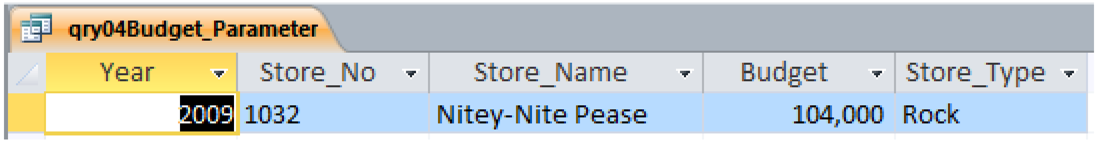
qry04Budget_Parameter

Year	Store_No	Store_Name	Budget	Store_Type
2010	1009	Nitey-Nite Isidor	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* **2010** *under the* **Year** *field with* **[Enter Year]** *and run the query.*
6. *Enter* **2009** *for the year and* **1032** *for the store number.*



qry04Budget_Parameter

Year	Store_No	Store_Name	Budget	Store_Type
2009	1032	Nitey-Nite Pease	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 2010 Review Questions Chapter 4, Section 1 of 2*** *option and complete the review questions.*

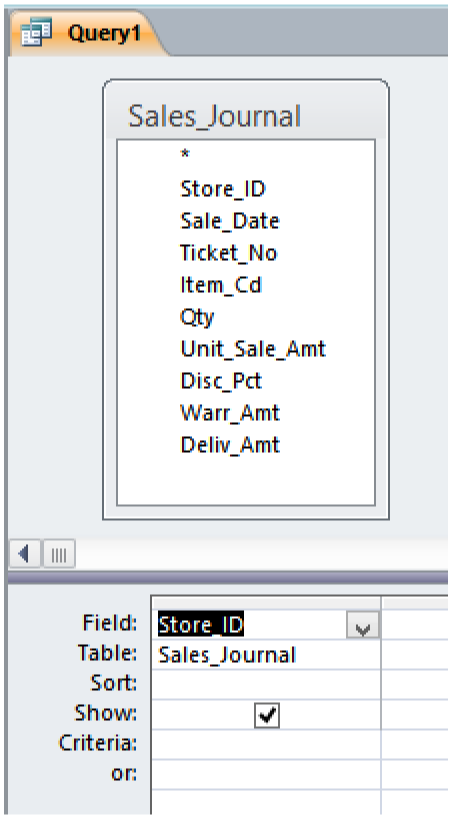
Grouping

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.*



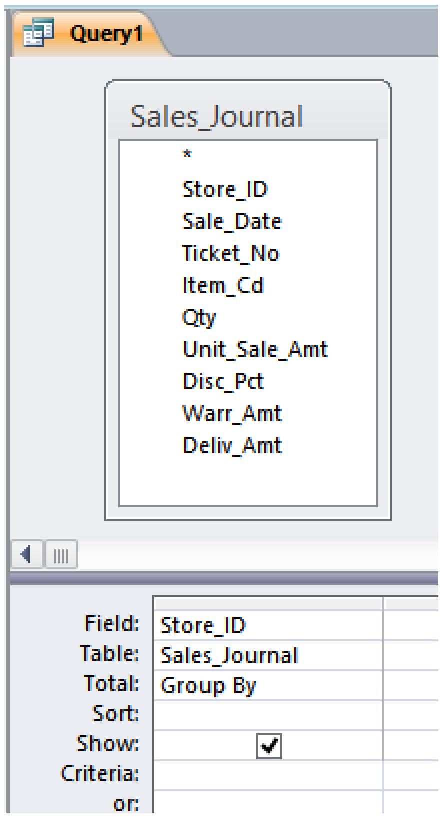
Field:	Store_ID
Table:	Sales_Journal
Sort:	
Show:	✓
Criteria:	
or:	

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.

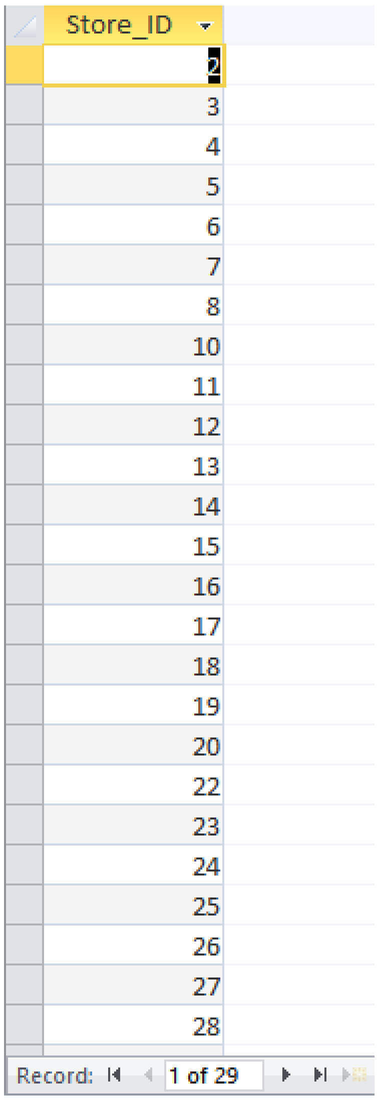
4. *Return to* **Design View**.
5. *Click the* **Totals** *icon* Σ *and run the query.*



Field:	Store_ID
Table:	Sales_Journal
Total:	Group By
Sort:	
Show:	✓
Criteria:	
or:	

Figure 4.14

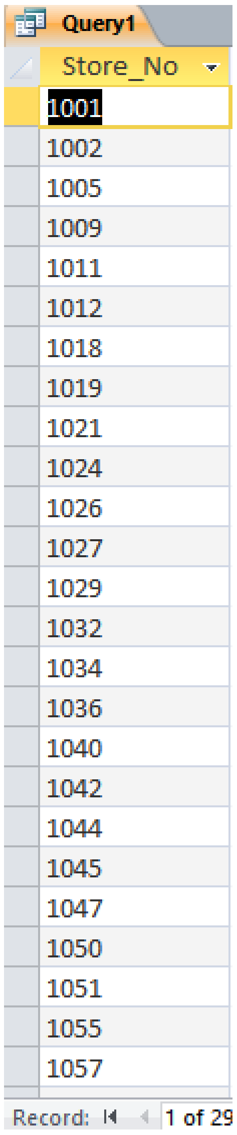
The **Total** row appears with a default value of **Group By** under Store_ID. When you run the query, you see that only 29 records appear, meaning that each of those Store_IDs appear at least once in the Sales_Journal table. The Store_ID in itself doesn't tell you much, but you can now create a join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



Store_ID
2
3
4
5
6
7
8
10
11
12
13
14
15
16
17
18
19
20
22
23
24
25
26
27
28

Figure 4.15

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


Query1

Store_No
1001
1002
1005
1009
1011
1012
1018
1019
1021
1024
1026
1027
1029
1032
1034
1036
1040
1042
1044
1045
1047
1050
1051
1055
1057

Record: 1 of 29

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.

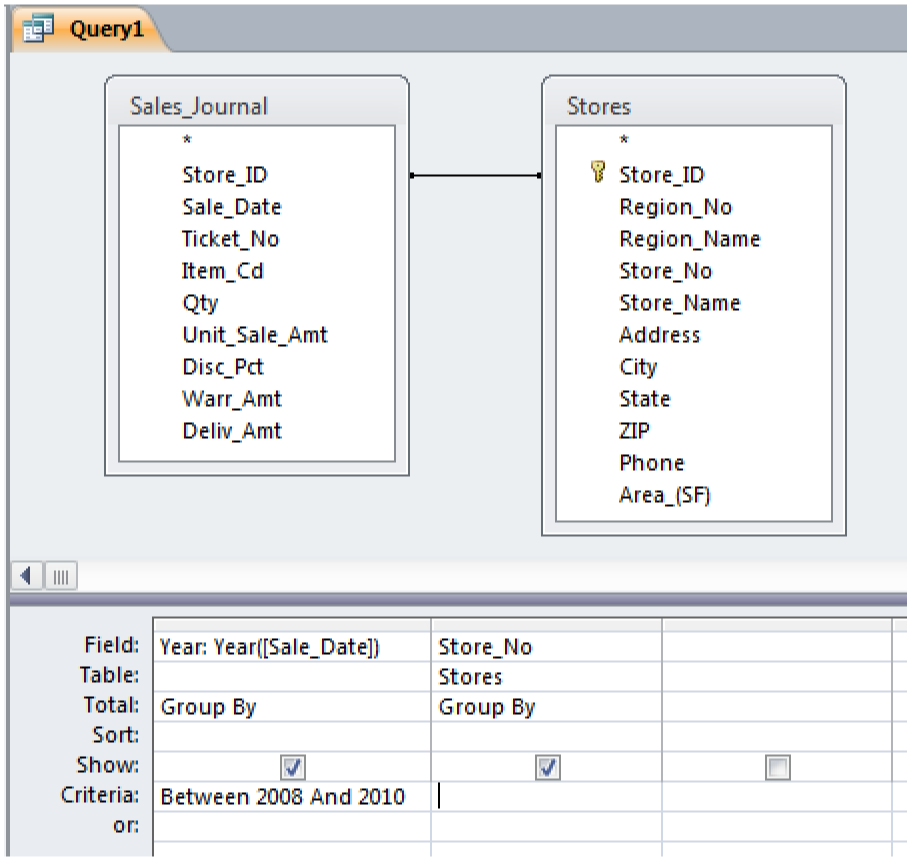
There are many Store_ID records in the Sales_Journal table, but only one unique Store_ID for each store in the Stores table. When you create a relationship on two tables like this, it will return the number of records in both tables. Had there been two records for each Store_ID in the Stores table, the 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.

> *(**As a 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 2008 – 2010. All you need to do is to bring in the right data.

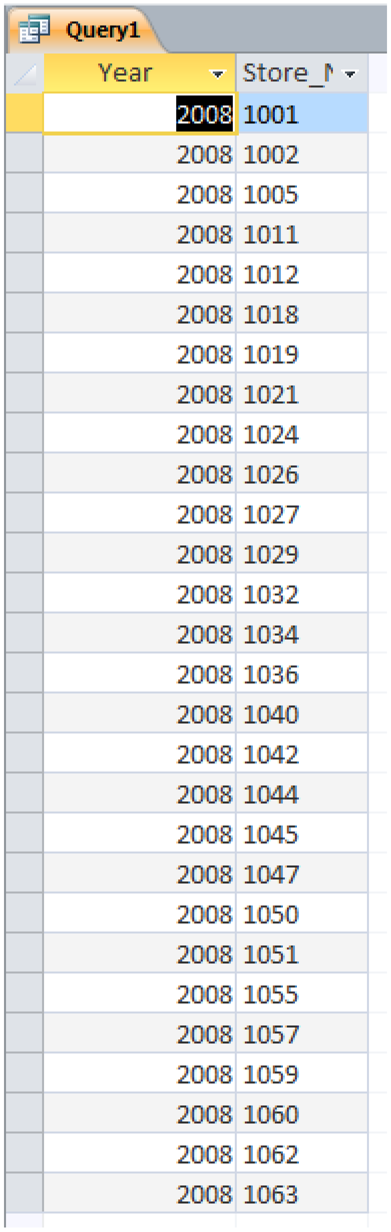
10. *Return to* **Design View** *of the query.*
11. *Drag the* **Sale_Date** *field in the* **Sales_Journal** *table to the left of the* **Store_No** *field.*
12. *Edit* **Sale_Date** *to be as follows:* **Year: Year(Sale_Date)**
13. *In the* **Year** *field, input the following criteria:* **Between 2008 and 2010**



Field:	Year: Year([Sale_Date])	Store_No	
Table:		Stores	
Total:	Group By	Group By	
Sort:			
Show:	☑	☑	☐
Criteria:	Between 2008 And 2010		
or:			

Figure 4.17

14. *Run the query.*

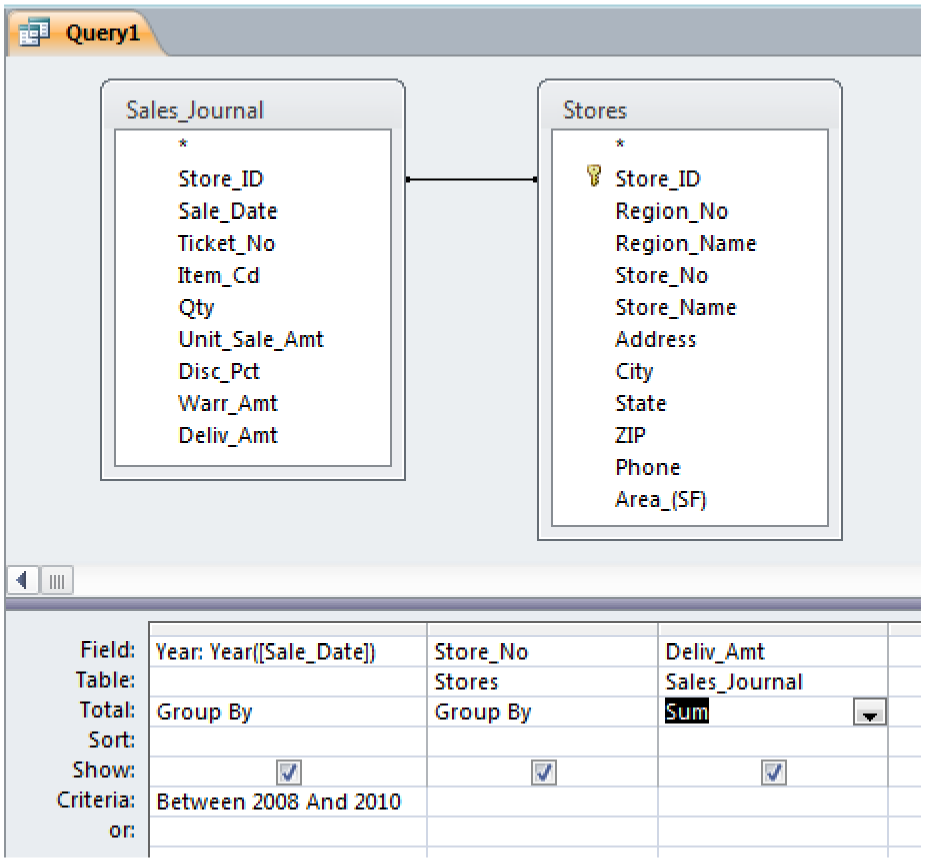

Query1

Year	Store_N
2008	1001
2008	1002
2008	1005
2008	1011
2008	1012
2008	1018
2008	1019
2008	1021
2008	1024
2008	1026
2008	1027
2008	1029
2008	1032
2008	1034
2008	1036
2008	1040
2008	1042
2008	1044
2008	1045
2008	1047
2008	1050
2008	1051
2008	1055
2008	1057
2008	1059
2008	1060
2008	1062
2008	1063

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.

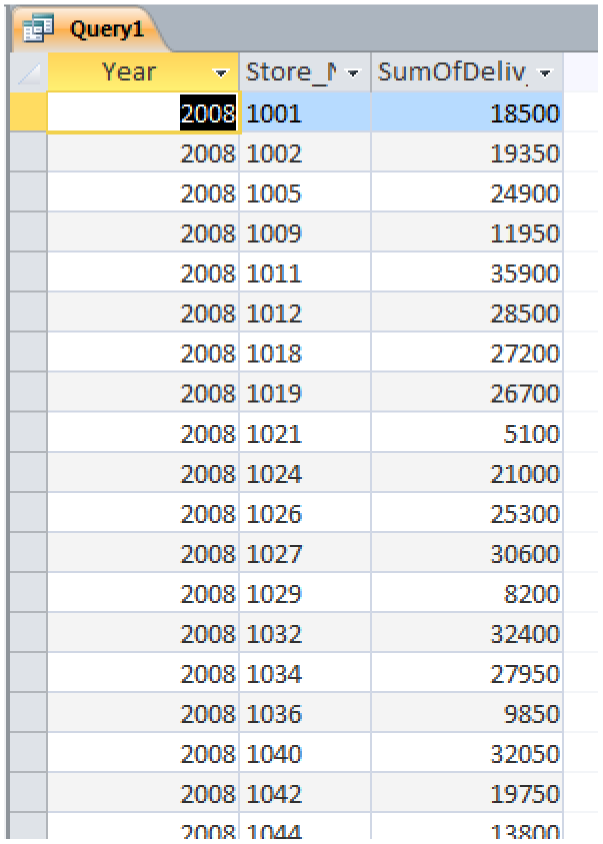
15. *Return to* **Design View**.
16. *Drag the* **Deliv_Amt** *field next to* **Store_No** *and release.*
17. *Under the* **Deliv_Amt** *field, change the* **Group By** *criteria to* **Sum**.



Field:	Year: Year([Sale_Date])	Store_No	Deliv_Amt
Table:		Stores	Sales_Journal
Total:	Group By	Group By	Sum
Sort:			
Show:	✓	✓	✓
Criteria:	Between 2008 And 2010		
or:			

Figure 4.19

18. Run the query.



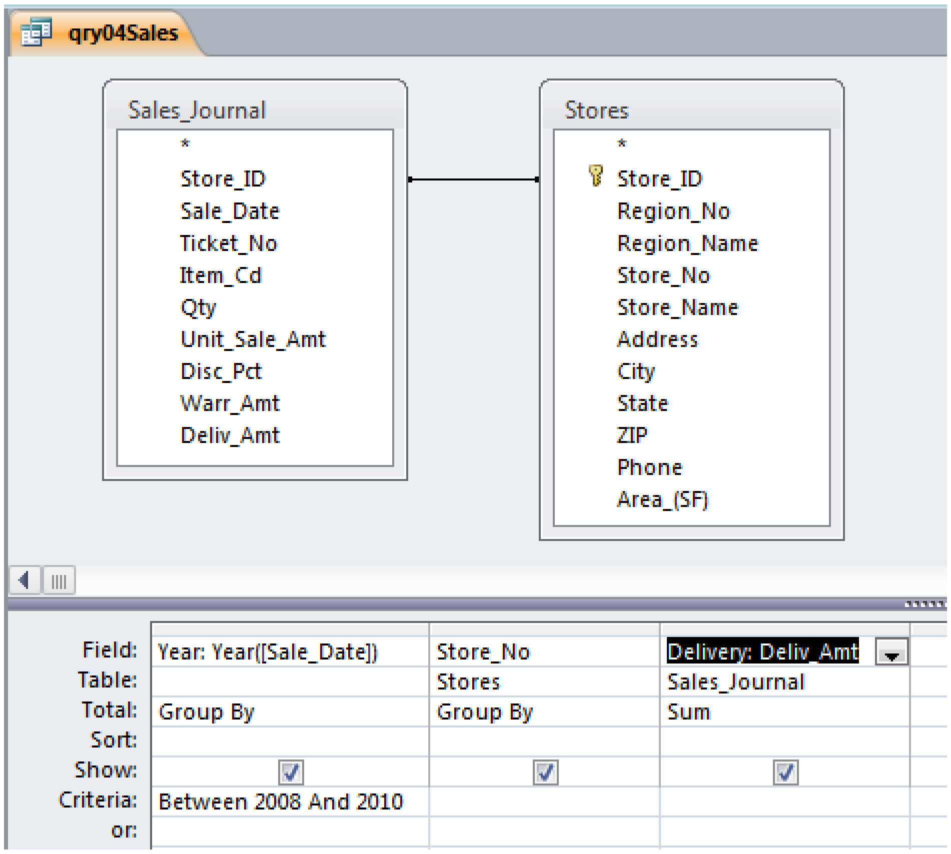
Query1

Year	Store_N	SumOfDeliv
2008	1001	18500
2008	1002	19350
2008	1005	24900
2008	1009	11950
2008	1011	35900
2008	1012	28500
2008	1018	27200
2008	1019	26700
2008	1021	5100
2008	1024	21000
2008	1026	25300
2008	1027	30600
2008	1029	8200
2008	1032	32400
2008	1034	27950
2008	1036	9850
2008	1040	32050
2008	1042	19750
2008	1044	13800

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.

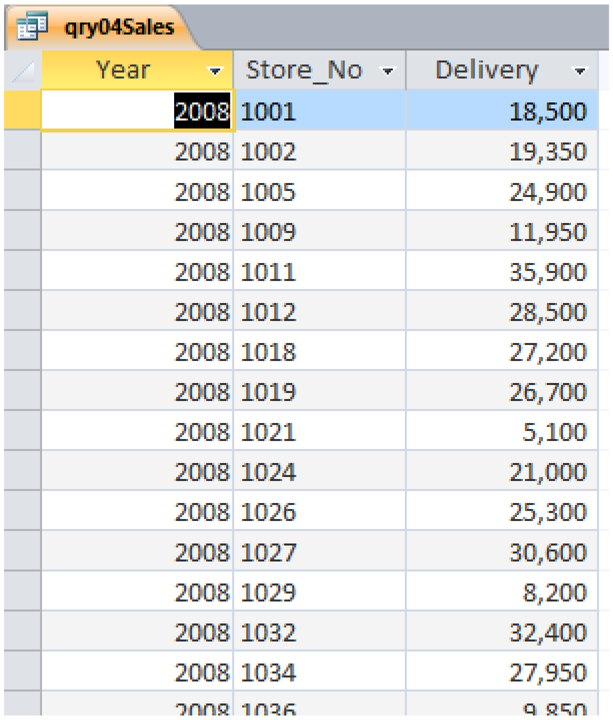
19. *Return to* **Design View**.
20. *Save the query as* **qry04Sales**.
21. *Give the* **Deliv_Amt** *field an alias of* **Delivery**
22. *Format that field to be* **Standard**, **zero** *decimal places.*



Field:	Year: Year([Sale_Date])	Store_No	Delivery: Deliv_Amt
Table:		Stores	Sales_Journal
Total:	Group By	Group By	Sum
Sort:			
Show:	✓	✓	✓
Criteria:	Between 2008 And 2010		
or:			

Figure 4.21

23. Run the query and adjust the field margins.



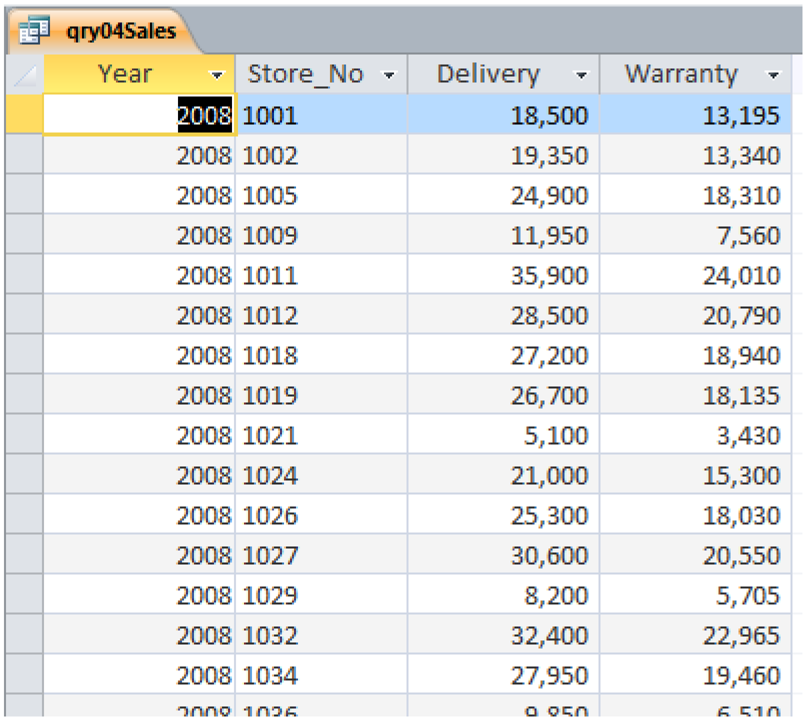
qry04Sales

Year	Store_No	Delivery
2008	1001	18,500
2008	1002	19,350
2008	1005	24,900
2008	1009	11,950
2008	1011	35,900
2008	1012	28,500
2008	1018	27,200
2008	1019	26,700
2008	1021	5,100
2008	1024	21,000
2008	1026	25,300
2008	1027	30,600
2008	1029	8,200
2008	1032	32,400
2008	1034	27,950
2008	1036	9,850

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.*



qry04Sales

Year	Store_No	Delivery	Warranty
2008	1001	18,500	13,195
2008	1002	19,350	13,340
2008	1005	24,900	18,310
2008	1009	11,950	7,560
2008	1011	35,900	24,010
2008	1012	28,500	20,790
2008	1018	27,200	18,940
2008	1019	26,700	18,135
2008	1021	5,100	3,430
2008	1024	21,000	15,300
2008	1026	25,300	18,030
2008	1027	30,600	20,550
2008	1029	8,200	5,705
2008	1032	32,400	22,965
2008	1034	27,950	19,460
2008	1036	9,850	6,510

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**.

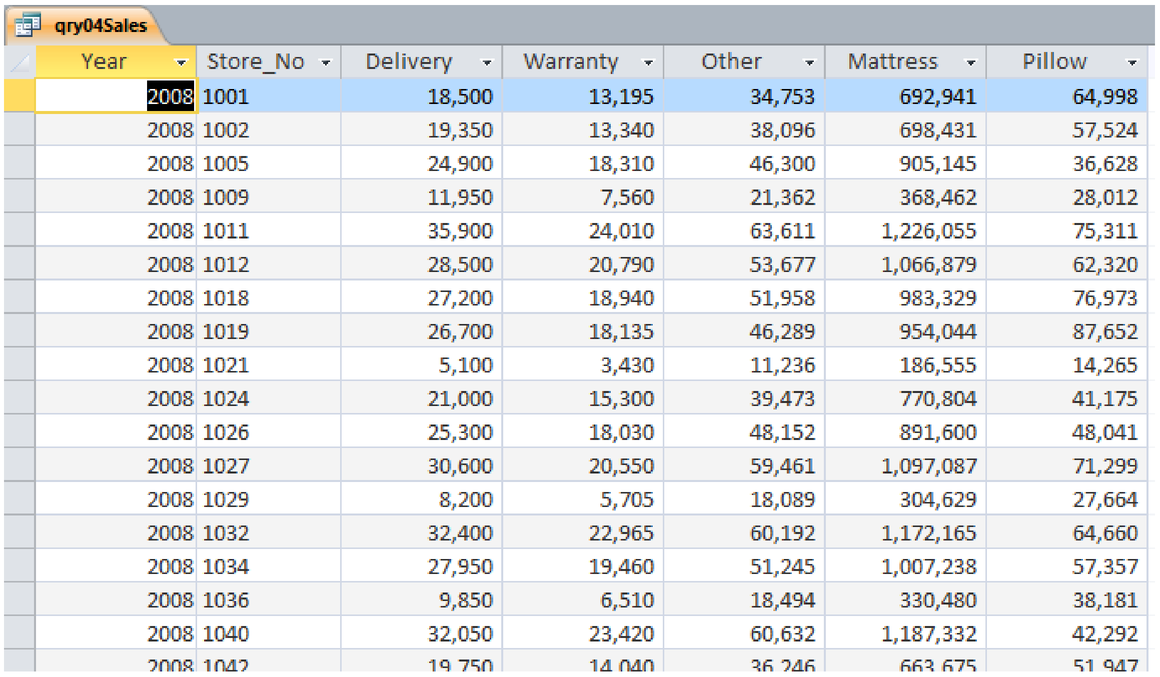
    *(**Note**: When writing a complex formula such as using an IIF() function, sometimes it is necessary to run the query before you can change the formatting of the field)*

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.*


qry04Sales

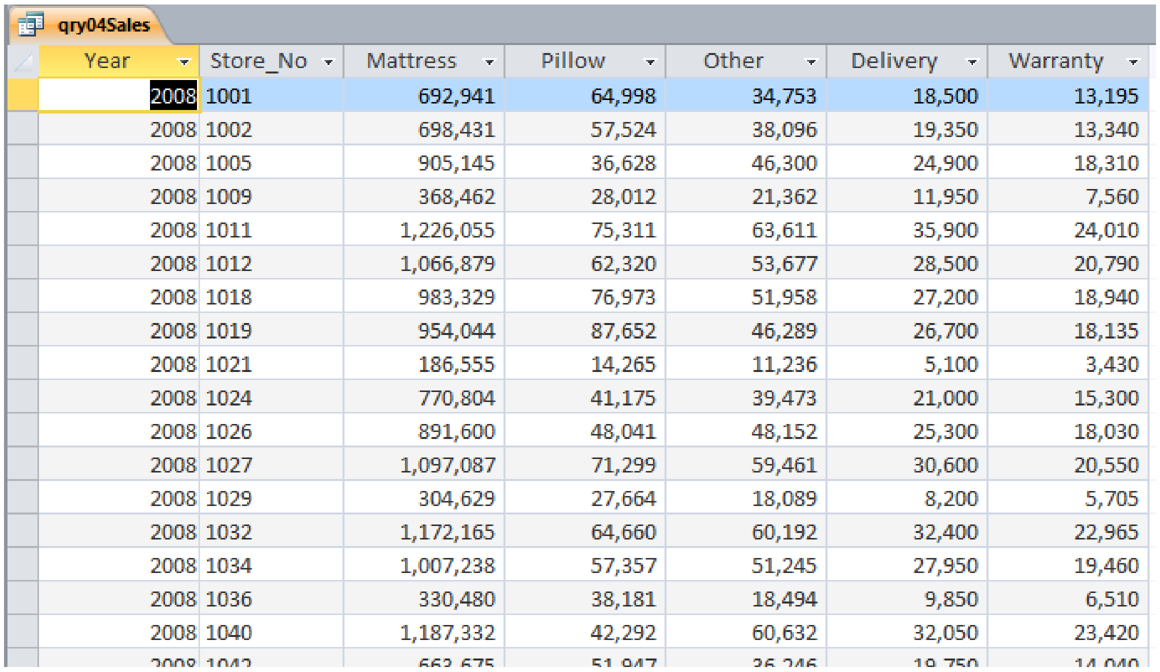
Year	Store_No	Delivery	Warranty	Other	Mattress	Pillow
2008	1001	18,500	13,195	34,753	692,941	64,998
2008	1002	19,350	13,340	38,096	698,431	57,524
2008	1005	24,900	18,310	46,300	905,145	36,628
2008	1009	11,950	7,560	21,362	368,462	28,012
2008	1011	35,900	24,010	63,611	1,226,055	75,311
2008	1012	28,500	20,790	53,677	1,066,879	62,320
2008	1018	27,200	18,940	51,958	983,329	76,973
2008	1019	26,700	18,135	46,289	954,044	87,652
2008	1021	5,100	3,430	11,236	186,555	14,265
2008	1024	21,000	15,300	39,473	770,804	41,175
2008	1026	25,300	18,030	48,152	891,600	48,041
2008	1027	30,600	20,550	59,461	1,097,087	71,299
2008	1029	8,200	5,705	18,089	304,629	27,664
2008	1032	32,400	22,965	60,192	1,172,165	64,660
2008	1034	27,950	19,460	51,245	1,007,238	57,357
2008	1036	9,850	6,510	18,494	330,480	38,181
2008	1040	32,050	23,420	60,632	1,187,332	42,292
2008	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.*

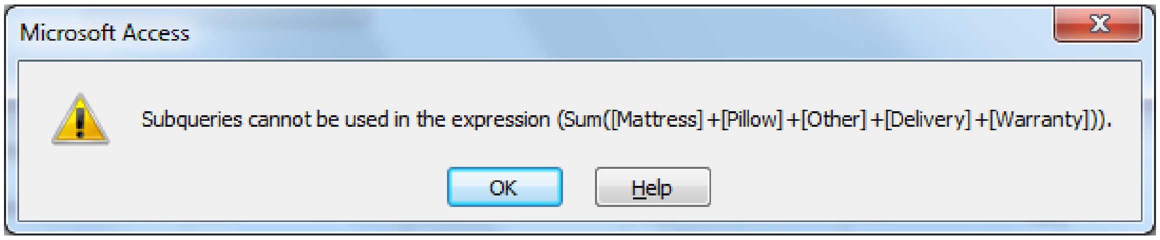


Year	Store_No	Mattress	Pillow	Other	Delivery	Warranty
2008	1001	692,941	64,998	34,753	18,500	13,195
2008	1002	698,431	57,524	38,096	19,350	13,340
2008	1005	905,145	36,628	46,300	24,900	18,310
2008	1009	368,462	28,012	21,362	11,950	7,560
2008	1011	1,226,055	75,311	63,611	35,900	24,010
2008	1012	1,066,879	62,320	53,677	28,500	20,790
2008	1018	983,329	76,973	51,958	27,200	18,940
2008	1019	954,044	87,652	46,289	26,700	18,135
2008	1021	186,555	14,265	11,236	5,100	3,430
2008	1024	770,804	41,175	39,473	21,000	15,300
2008	1026	891,600	48,041	48,152	25,300	18,030
2008	1027	1,097,087	71,299	59,461	30,600	20,550
2008	1029	304,629	27,664	18,089	8,200	5,705
2008	1032	1,172,165	64,660	60,192	32,400	22,965
2008	1034	1,007,238	57,357	51,245	27,950	19,460
2008	1036	330,480	38,181	18,494	9,850	6,510
2008	1040	1,187,332	42,292	60,632	32,050	23,420
2008	1042	663,675	51,947	36,246	19,750	14,040

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.*



Microsoft Access

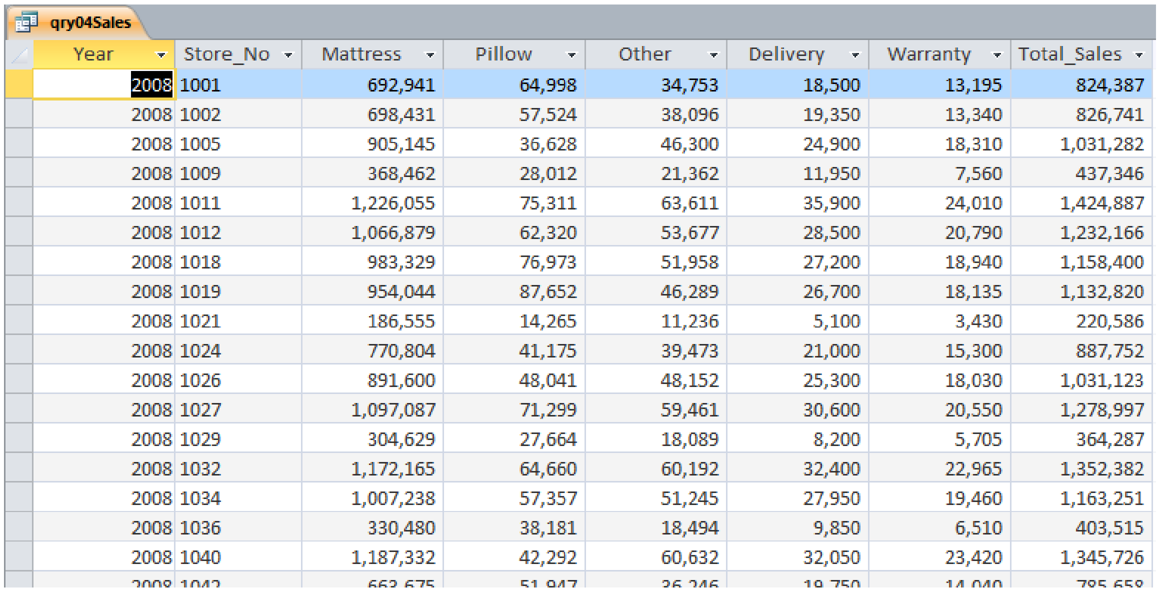
Subqueries cannot be used in the expression (Sum([Mattress]+[Pillow]+[Other]+[Delivery]+[Warranty])).

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.

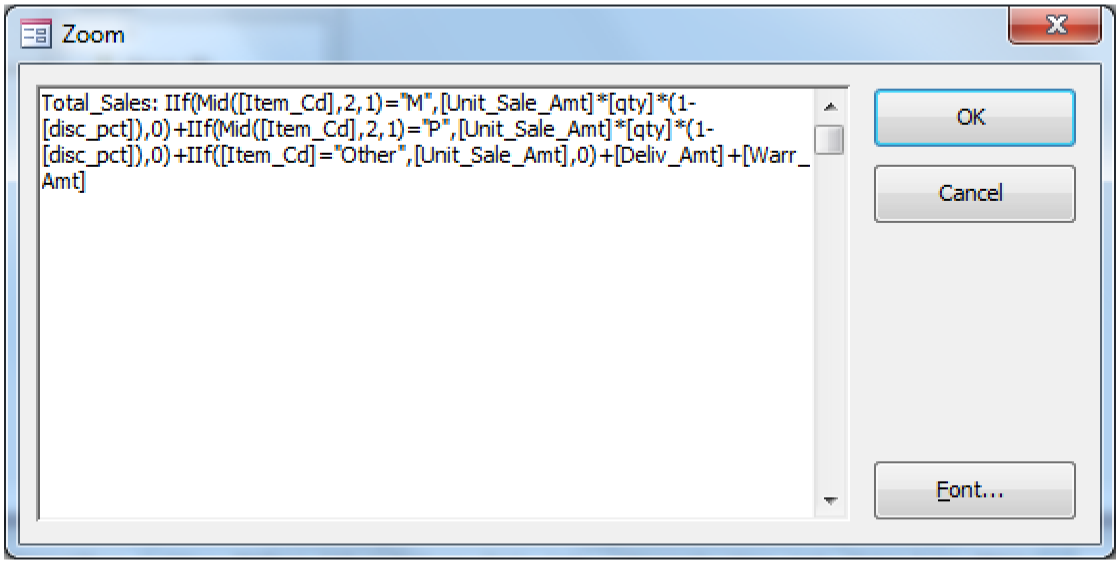
1. *Click* **OK** *to return to* **Design View**.
2. *Copy the formula in the* **Mattress** *field and replace "***[Mattress]***" in the* **Total_Sales** *field with the formula.*
3. *Do the same for each corresponding field and formula.*
4. *Use* **Sum** *in the* **Total** *line.*
5. *Format as the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



Year	Store_No	Mattress	Pillow	Other	Delivery	Warranty	Total_Sales
2008	1001	692,941	64,998	34,753	18,500	13,195	824,387
2008	1002	698,431	57,524	38,096	19,350	13,340	826,741
2008	1005	905,145	36,628	46,300	24,900	18,310	1,031,282
2008	1009	368,462	28,012	21,362	11,950	7,560	437,346
2008	1011	1,226,055	75,311	63,611	35,900	24,010	1,424,887
2008	1012	1,066,879	62,320	53,677	28,500	20,790	1,232,166
2008	1018	983,329	76,973	51,958	27,200	18,940	1,158,400
2008	1019	954,044	87,652	46,289	26,700	18,135	1,132,820
2008	1021	186,555	14,265	11,236	5,100	3,430	220,586
2008	1024	770,804	41,175	39,473	21,000	15,300	887,752
2008	1026	891,600	48,041	48,152	25,300	18,030	1,031,123
2008	1027	1,097,087	71,299	59,461	30,600	20,550	1,278,997
2008	1029	304,629	27,664	18,089	8,200	5,705	364,287
2008	1032	1,172,165	64,660	60,192	32,400	22,965	1,352,382
2008	1034	1,007,238	57,357	51,245	27,950	19,460	1,163,251
2008	1036	330,480	38,181	18,494	9,850	6,510	403,515
2008	1040	1,187,332	42,292	60,632	32,050	23,420	1,345,726
2008	1042	663,675	51,947	36,246	19,750	14,040	785,658

Figure 4.27

It may take a few seconds to run because you're doing a lot of calculations on over 140,000 records. That ain't no walk in the park, so Access may take a few seconds. Mine took seventeen 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:

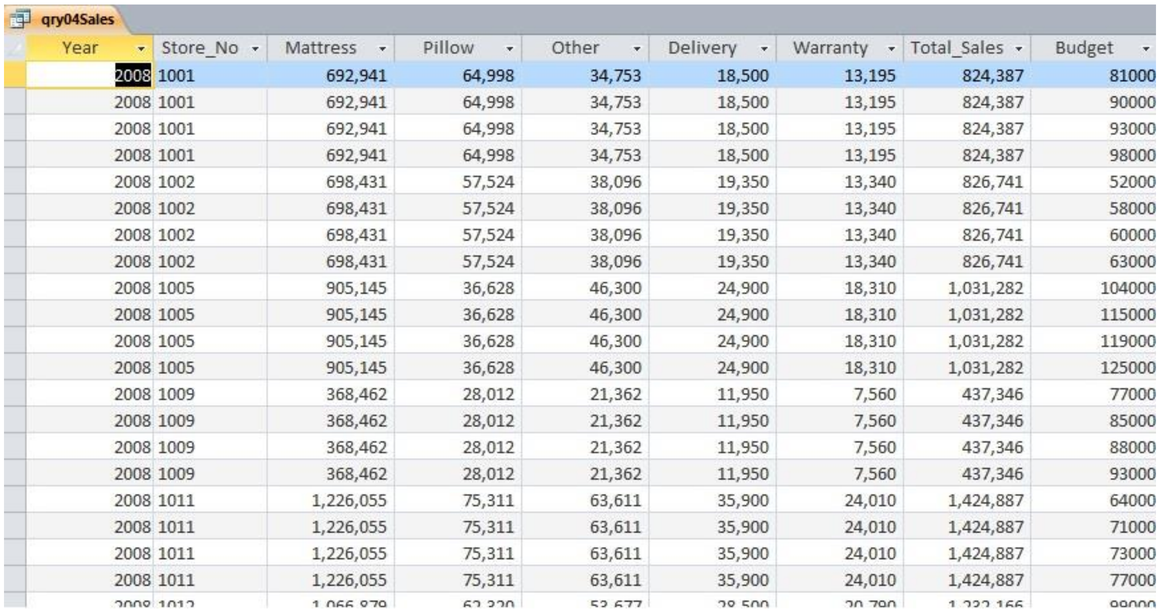


```
Total_Sales: IIf(Mid([Item_Cd],2,1)="M",[Unit_Sale_Amt]*[qty]*(1-[disc_pct]),0)+IIf(Mid([Item_Cd],2,1)="P",[Unit_Sale_Amt]*[qty]*(1-[disc_pct]),0)+IIf([Item_Cd]="Other",[Unit_Sale_Amt],0)+[Deliv_Amt]+[Warr_Amt]
```

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***)*
10. *Run the query.*



Year	Store_No	Mattress	Pillow	Other	Delivery	Warranty	Total_Sales	Budget
2008	1001	692,941	64,998	34,753	18,500	13,195	824,387	81000
2008	1001	692,941	64,998	34,753	18,500	13,195	824,387	90000
2008	1001	692,941	64,998	34,753	18,500	13,195	824,387	93000
2008	1001	692,941	64,998	34,753	18,500	13,195	824,387	98000
2008	1002	698,431	57,524	38,096	19,350	13,340	826,741	52000
2008	1002	698,431	57,524	38,096	19,350	13,340	826,741	58000
2008	1002	698,431	57,524	38,096	19,350	13,340	826,741	60000
2008	1002	698,431	57,524	38,096	19,350	13,340	826,741	63000
2008	1005	905,145	36,628	46,300	24,900	18,310	1,031,282	104000
2008	1005	905,145	36,628	46,300	24,900	18,310	1,031,282	115000
2008	1005	905,145	36,628	46,300	24,900	18,310	1,031,282	119000
2008	1005	905,145	36,628	46,300	24,900	18,310	1,031,282	125000
2008	1009	368,462	28,012	21,362	11,950	7,560	437,346	77000
2008	1009	368,462	28,012	21,362	11,950	7,560	437,346	85000
2008	1009	368,462	28,012	21,362	11,950	7,560	437,346	88000
2008	1009	368,462	28,012	21,362	11,950	7,560	437,346	93000
2008	1011	1,226,055	75,311	63,611	35,900	24,010	1,424,887	64000
2008	1011	1,226,055	75,311	63,611	35,900	24,010	1,424,887	71000
2008	1011	1,226,055	75,311	63,611	35,900	24,010	1,424,887	73000
2008	1011	1,226,055	75,311	63,611	35,900	24,010	1,424,887	77000

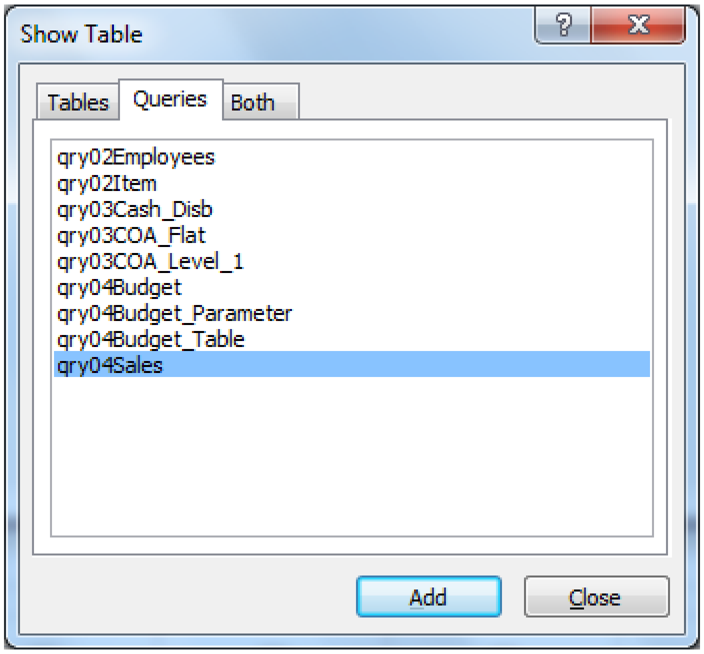
Figure 4.29

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.

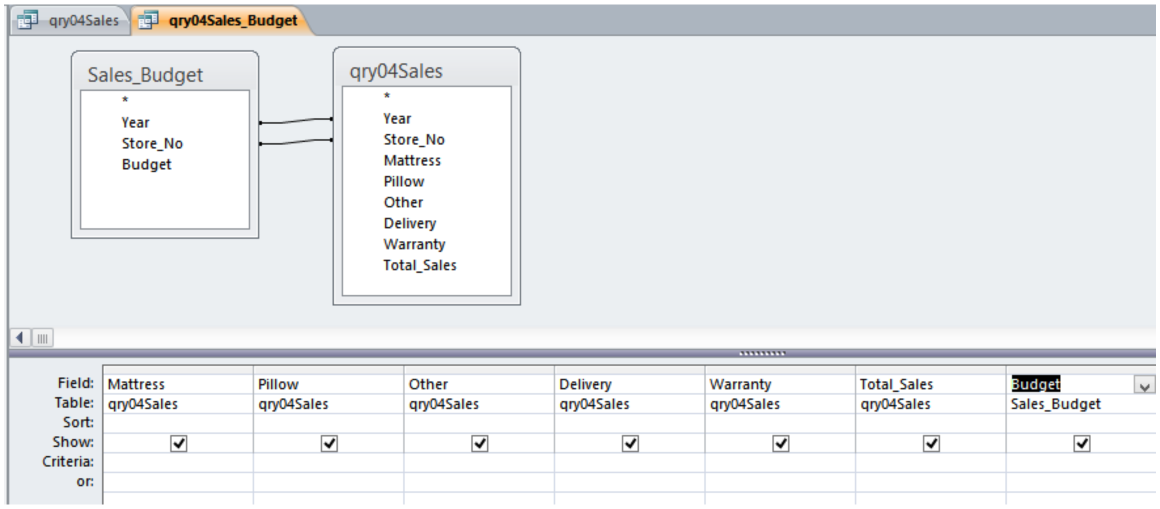
1. *Return to* **Design View**.
2. *Delete the* **Sales_Budget** *table from the* **Table** *area by clicking anywhere in the* **Sales_Budget** *table and press the* **[Del]** *key. You can also right-click on the table and choose* **Remove table**.
3. *Save* **qry04Sales**.
4. *Create a new query.*
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.*



qry02Employees
qry02Item
qry03Cash_Disb
qry03COA_Flat
qry03COA_Level_1
qry04Budget
qry04Budget_Parameter
qry04Budget_Table
qry04Sales

Figure 4.30

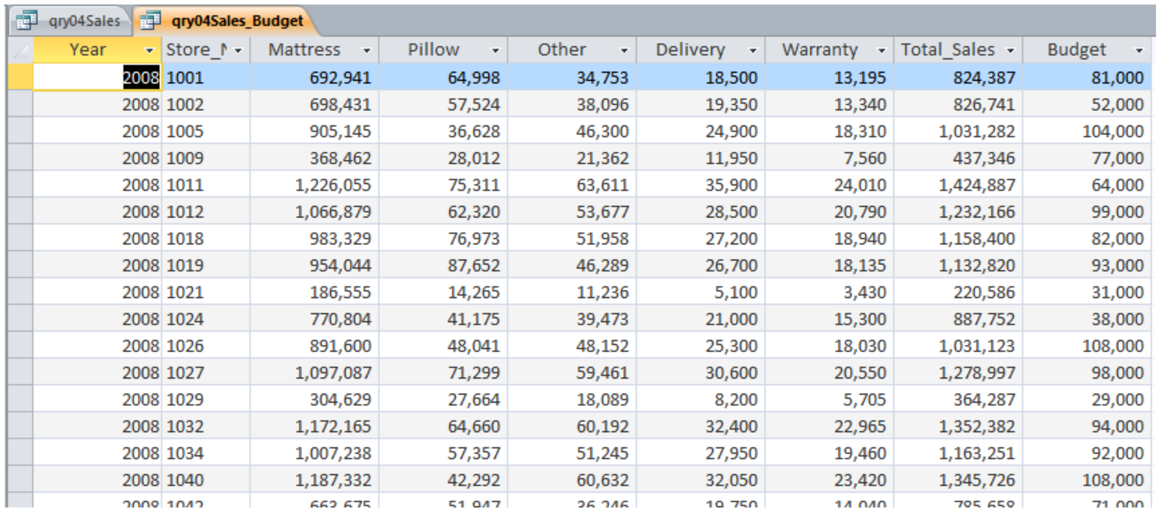
7. *Create joins between the* **Sales_Budget** *table and the* **qry04Sales** *query on* **Year** *and* **Store_No.**
8. *Bring in all fields from* **qry04Sales** *and the* **Budget** *field from* **Sales_Budget.**



Field:	Mattress	Pillow	Other	Delivery	Warranty	Total_Sales	Budget
Table:	qry04Sales	qry04Sales	qry04Sales	qry04Sales	qry04Sales	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.*



qry04Sales | qry04Sales_Budget

Year	Store_N	Mattress	Pillow	Other	Delivery	Warranty	Total_Sales	Budget
2008	1001	692,941	64,998	34,753	18,500	13,195	824,387	81,000
2008	1002	698,431	57,524	38,096	19,350	13,340	826,741	52,000
2008	1005	905,145	36,628	46,300	24,900	18,310	1,031,282	104,000
2008	1009	368,462	28,012	21,362	11,950	7,560	437,346	77,000
2008	1011	1,226,055	75,311	63,611	35,900	24,010	1,424,887	64,000
2008	1012	1,066,879	62,320	53,677	28,500	20,790	1,232,166	99,000
2008	1018	983,329	76,973	51,958	27,200	18,940	1,158,400	82,000
2008	1019	954,044	87,652	46,289	26,700	18,135	1,132,820	93,000
2008	1021	186,555	14,265	11,236	5,100	3,430	220,586	31,000
2008	1024	770,804	41,175	39,473	21,000	15,300	887,752	38,000
2008	1026	891,600	48,041	48,152	25,300	18,030	1,031,123	108,000
2008	1027	1,097,087	71,299	59,461	30,600	20,550	1,278,997	98,000
2008	1029	304,629	27,664	18,089	8,200	5,705	364,287	29,000
2008	1032	1,172,165	64,660	60,192	32,400	22,965	1,352,382	94,000
2008	1034	1,007,238	57,357	51,245	27,950	19,460	1,163,251	92,000
2008	1040	1,187,332	42,292	60,632	32,050	23,420	1,345,726	108,000
2008	1042	663,675	51,947	36,246	19,750	14,040	785,658	71,000

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.*

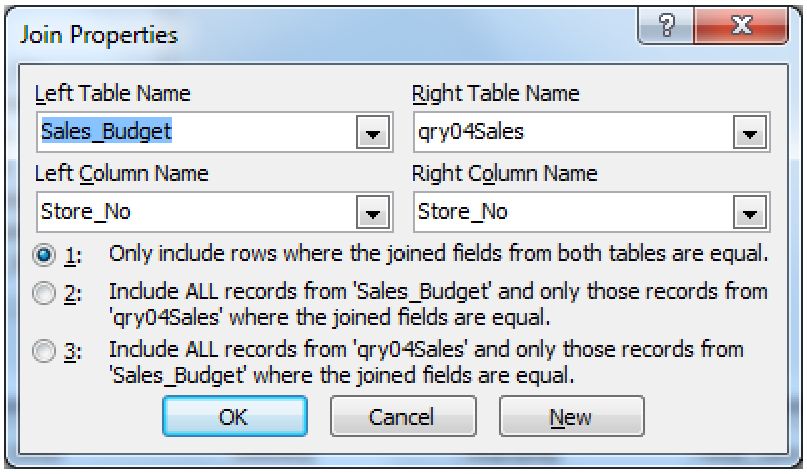
One-Way Joins

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 eighty four records returned in qry04Sales_Budget, but there were eighty seven 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.


Join Properties
?
X
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.
OK
Cancel
New
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.

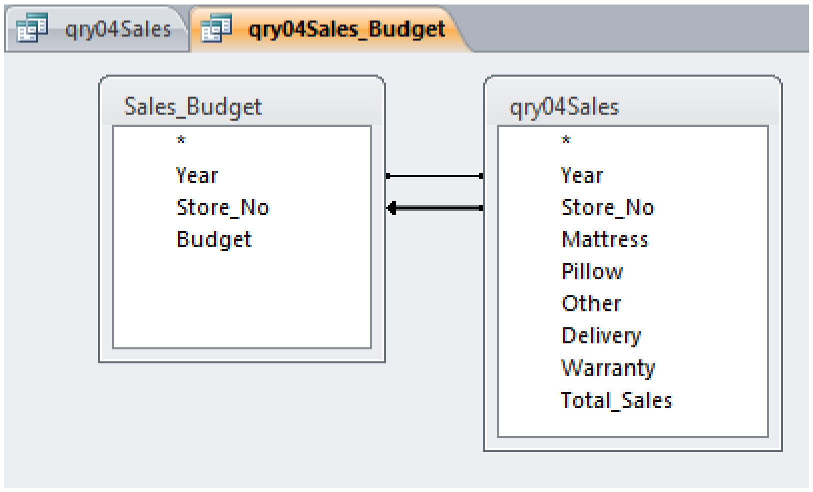
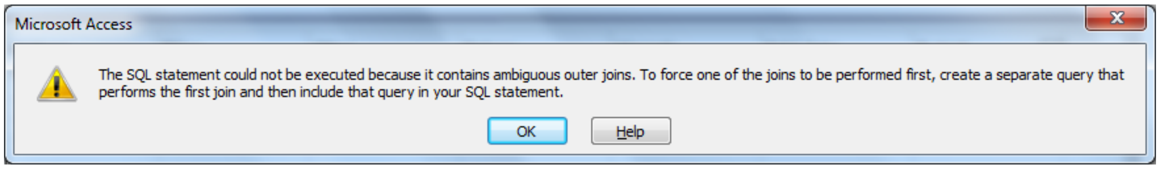


Figure 4.34

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.

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 same tables. 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.*

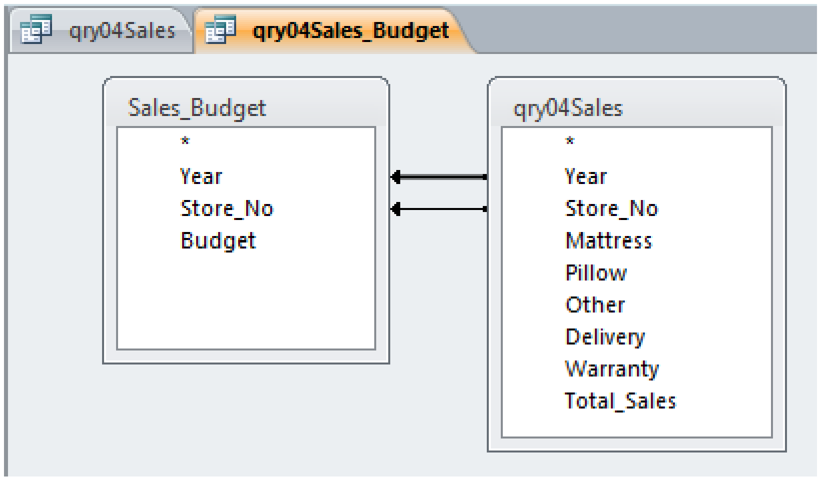
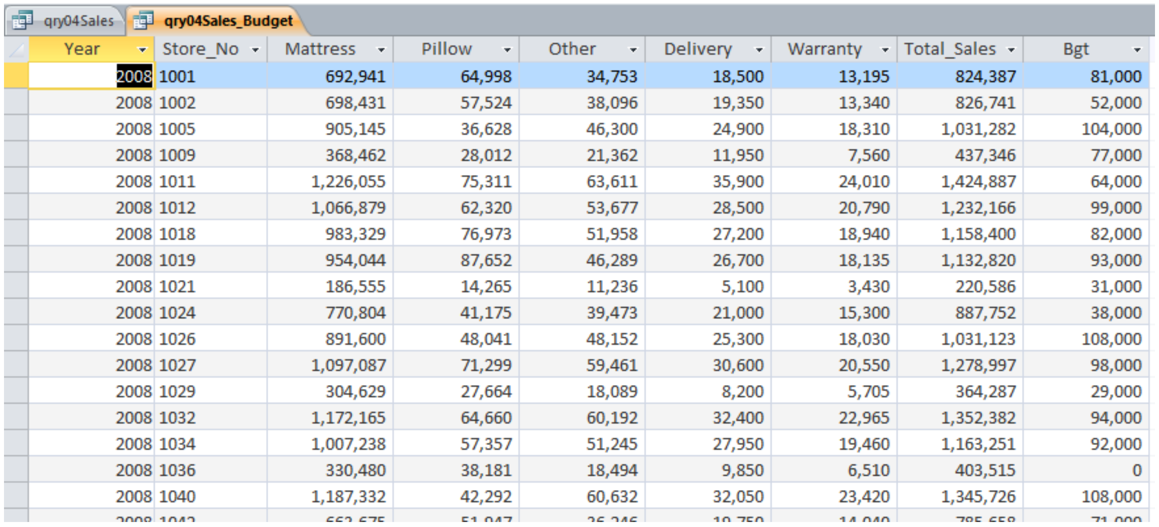


Figure 4.36

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.*



qry04Sales | qry04Sales_Budget

Year	Store_No	Mattress	Pillow	Other	Delivery	Warranty	Total_Sales	Bgt
2008	1001	692,941	64,998	34,753	18,500	13,195	824,387	81,000
2008	1002	698,431	57,524	38,096	19,350	13,340	826,741	52,000
2008	1005	905,145	36,628	46,300	24,900	18,310	1,031,282	104,000
2008	1009	368,462	28,012	21,362	11,950	7,560	437,346	77,000
2008	1011	1,226,055	75,311	63,611	35,900	24,010	1,424,887	64,000
2008	1012	1,066,879	62,320	53,677	28,500	20,790	1,232,166	99,000
2008	1018	983,329	76,973	51,958	27,200	18,940	1,158,400	82,000
2008	1019	954,044	87,652	46,289	26,700	18,135	1,132,820	93,000
2008	1021	186,555	14,265	11,236	5,100	3,430	220,586	31,000
2008	1024	770,804	41,175	39,473	21,000	15,300	887,752	38,000
2008	1026	891,600	48,041	48,152	25,300	18,030	1,031,123	108,000
2008	1027	1,097,087	71,299	59,461	30,600	20,550	1,278,997	98,000
2008	1029	304,629	27,664	18,089	8,200	5,705	364,287	29,000
2008	1032	1,172,165	64,660	60,192	32,400	22,965	1,352,382	94,000
2008	1034	1,007,238	57,357	51,245	27,950	19,460	1,163,251	92,000
2008	1036	330,480	38,181	18,494	9,850	6,510	403,515	0
2008	1040	1,187,332	42,292	60,632	32,050	23,420	1,345,726	108,000

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 all 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 2010 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, log in and take the exam. Make sure that you take the exam on the same computer that you completed the sample files on, as some of the questions on the exam may refer to some of the completed examples.

ExcelCEO

Chief Excel Officer

Access 2010 and SQL

Complete Self-study Course

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.

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 we had 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:

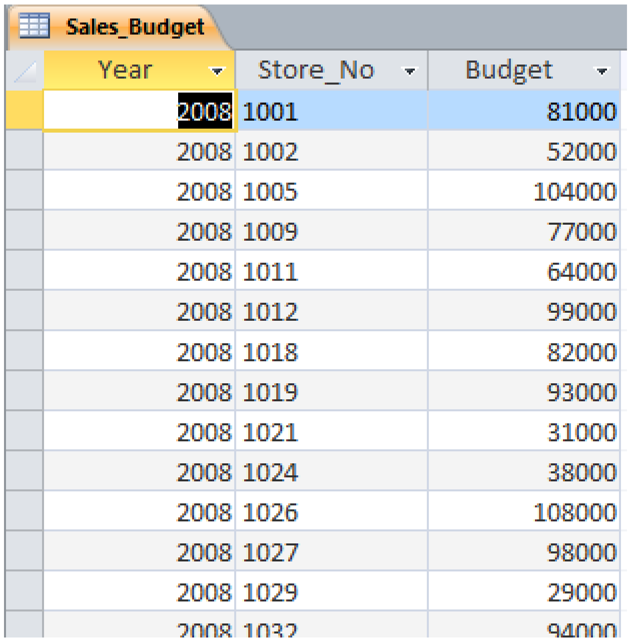
- **Select Query**: The most common type and the only type we've used up to this point.
- **Crosstab Query**: Allows you to pivot data from columns into rows or vice versa.
- **Make Table Query**: Creates a table when executed.
- **Update Query**: Modifies data in a table based on criteria you specify.
- **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 2010** *database, open the* **Sales_Budget** *table. Notice how the years and store numbers are displayed.*

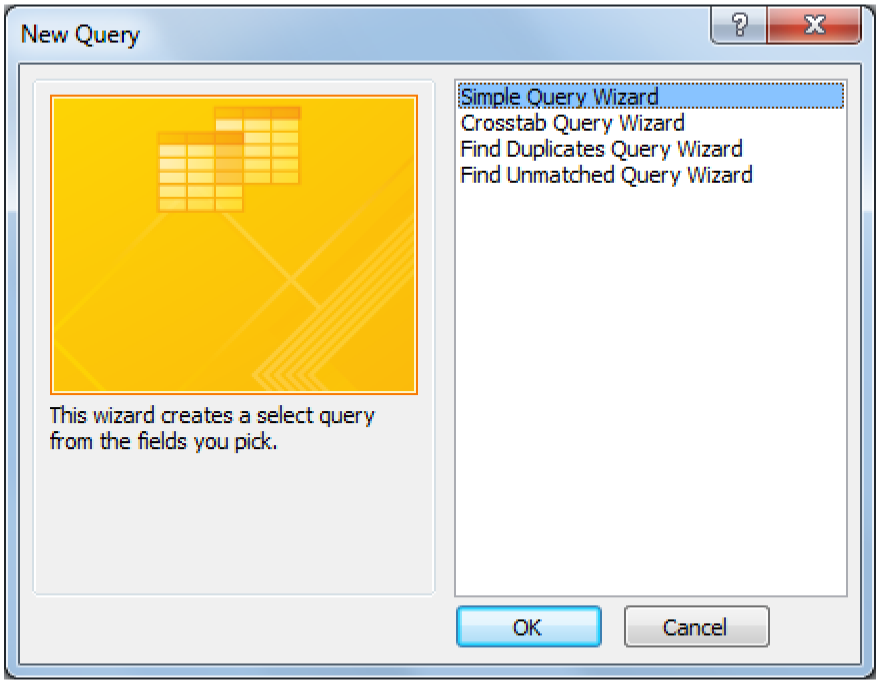


Sales_Budget

Year	Store_No	Budget
2008	1001	81000
2008	1002	52000
2008	1005	104000
2008	1009	77000
2008	1011	64000
2008	1012	99000
2008	1018	82000
2008	1019	93000
2008	1021	31000
2008	1024	38000
2008	1026	108000
2008	1027	98000
2008	1029	29000
2008	1032	94000

Figure 5.1

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



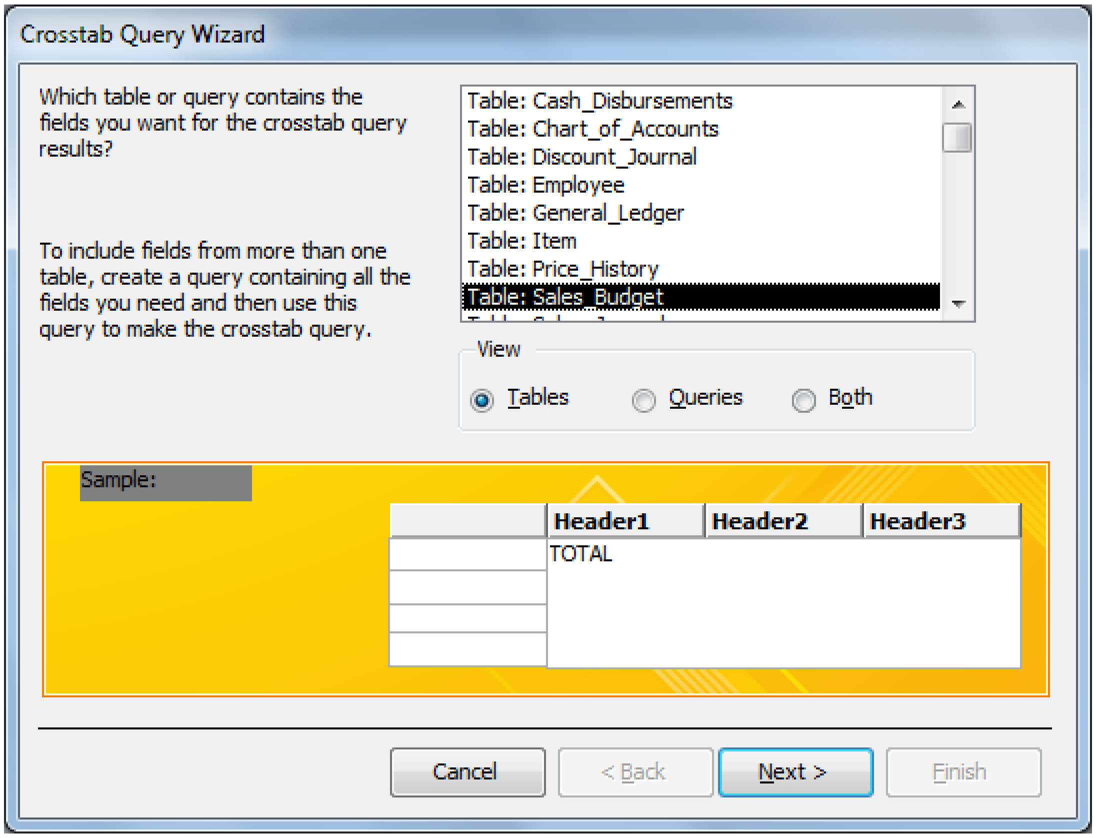
New Query

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

- Simple Query Wizard
- Crosstab Query Wizard
- Find Duplicates Query Wizard
- Find Unmatched Query Wizard

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

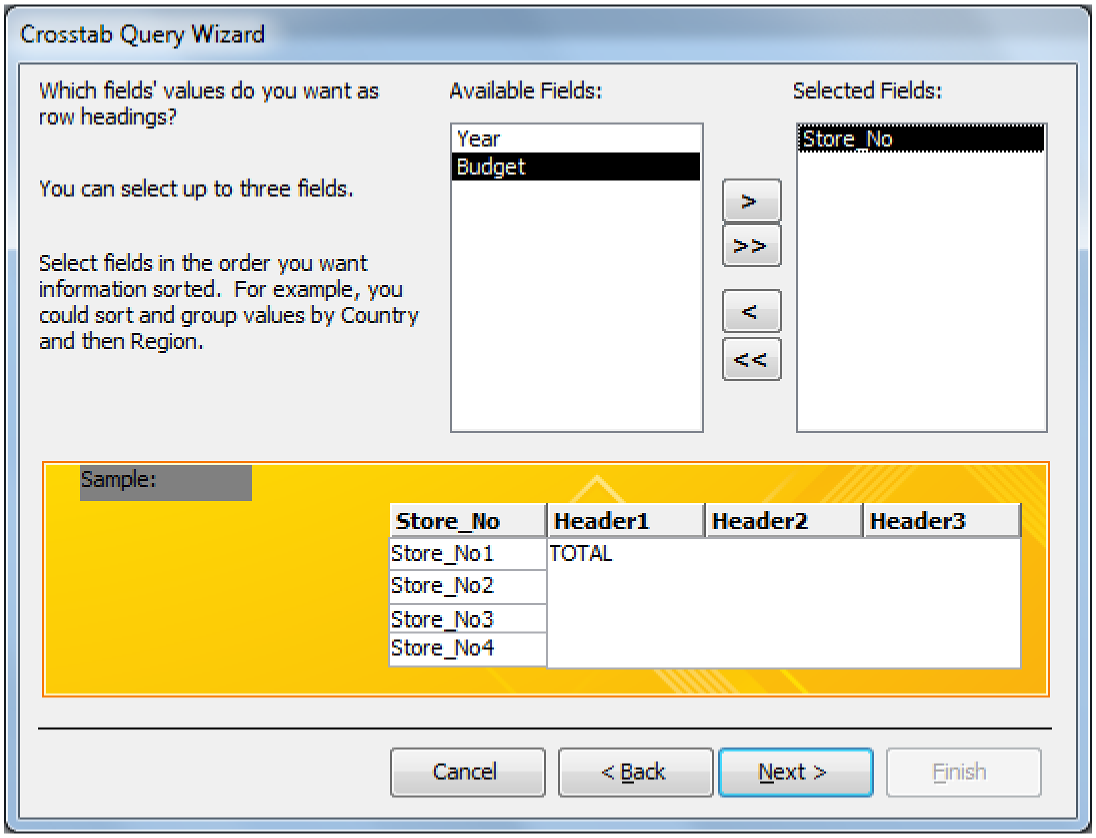
Sample:

	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).*



Which fields' values do you want as row headings?

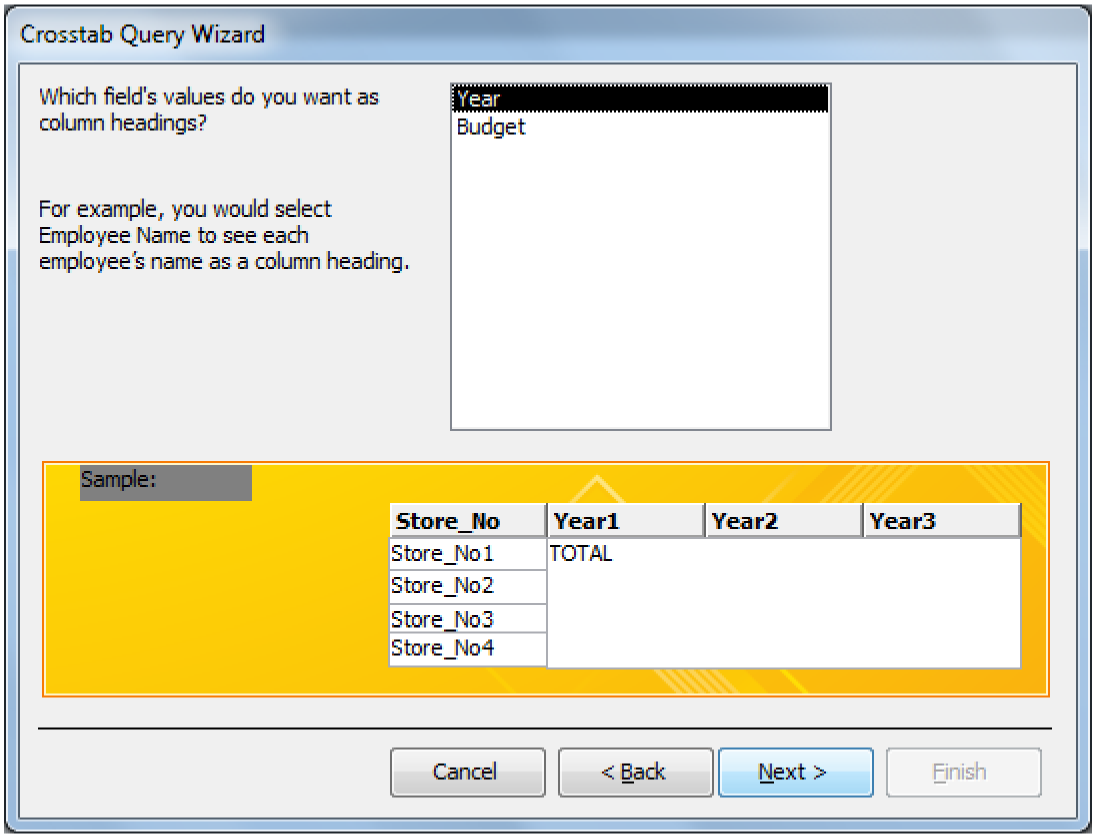
You can select up to three fields.

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

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

Which field's values do you want as column headings?

For example, you would select Employee Name to see each employee's name as a column heading.

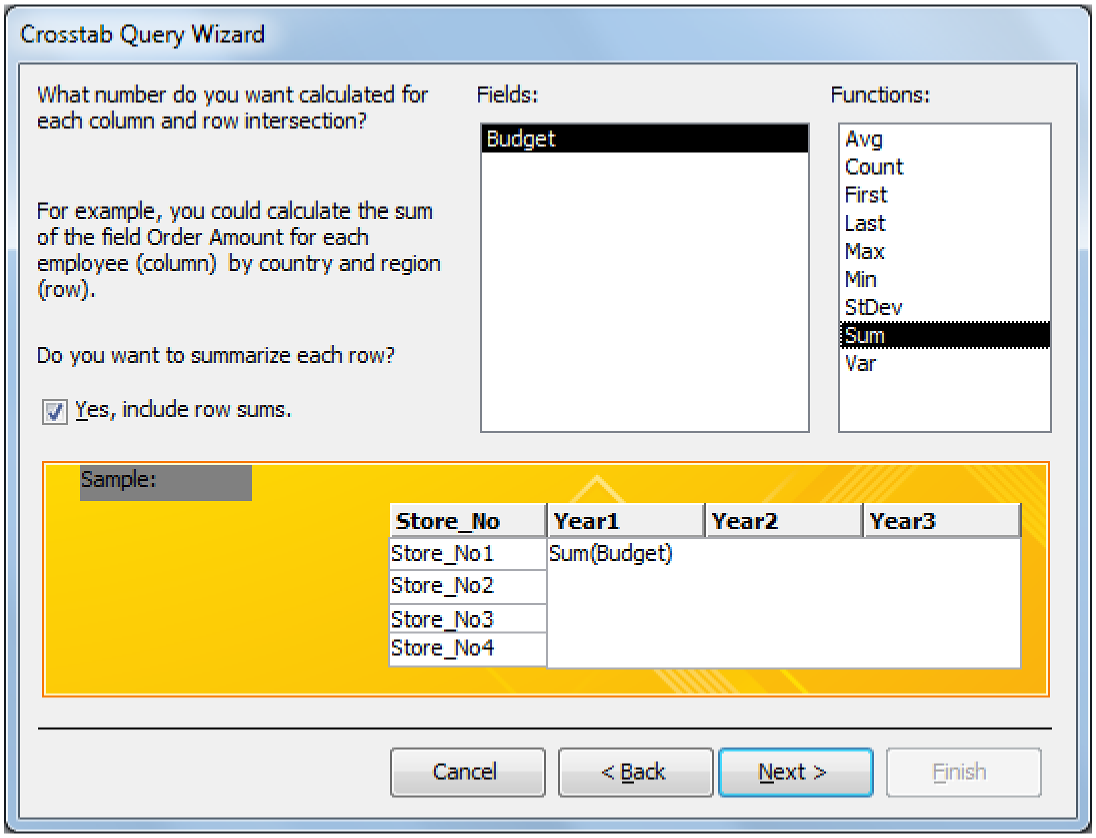
Year
Budget

Sample:

Store_No	Year1	Year2	Year3
Store_No1	TOTAL		
Store_No2			
Store_No3			
Store_No4			

Figure 5.5

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



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:

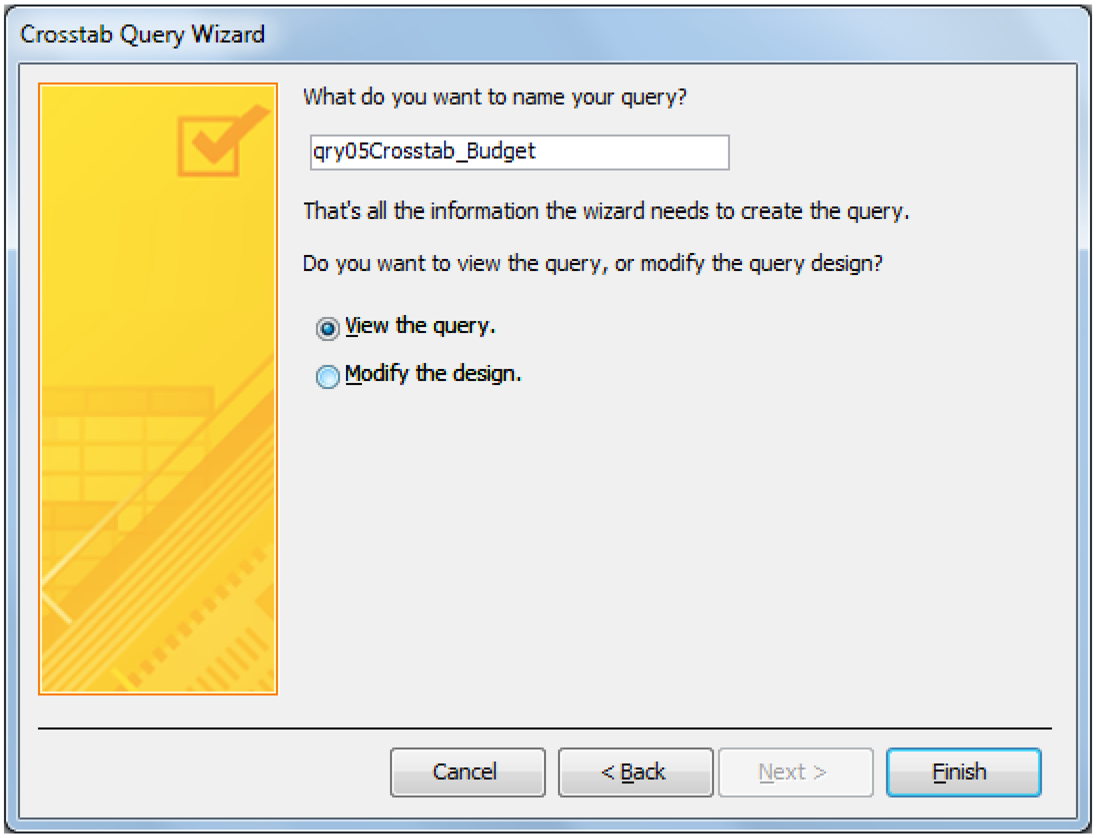
Avg
Count
First
Last
Max
Min
StDev
Sum
Var

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?

qry05Crosstab_Budget

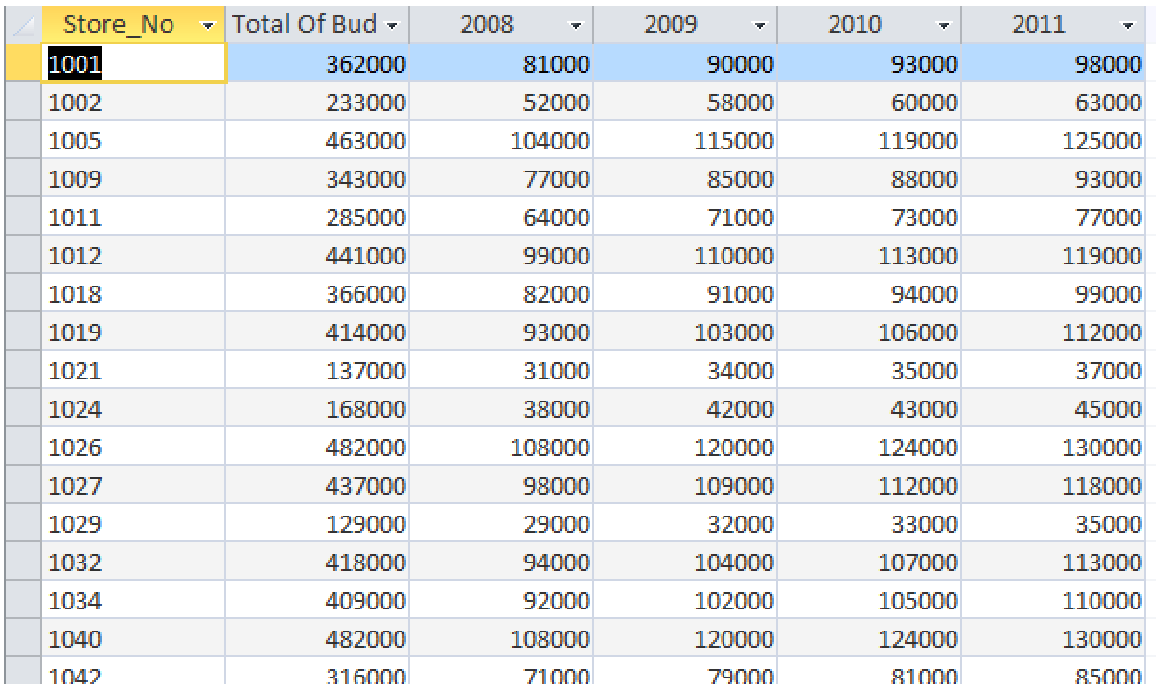
That's all the information the wizard needs to create the query.

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

[✓ View the query.]
Modify the design.

Figure 5.7

11. Click **Finish**.



Store_No	Total Of Bud	2008	2009	2010	2011
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

Figure 5.8

The years have moved from the values in rows to being column 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.

A Make Table Query can cut the run time 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 for some reason does not run, 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**.

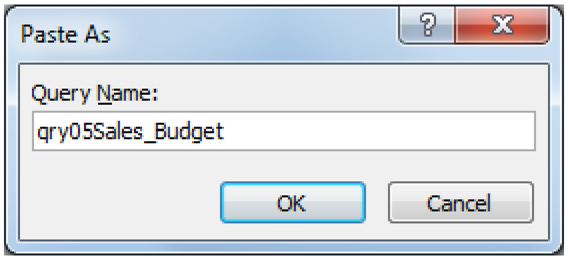


Figure 5.9

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**.

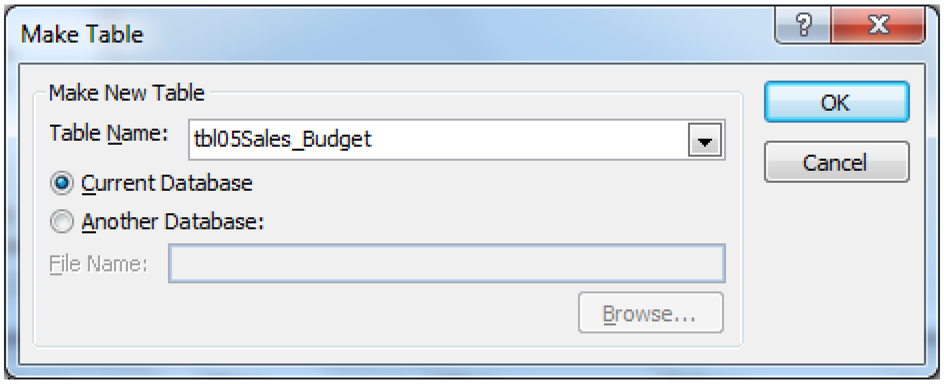
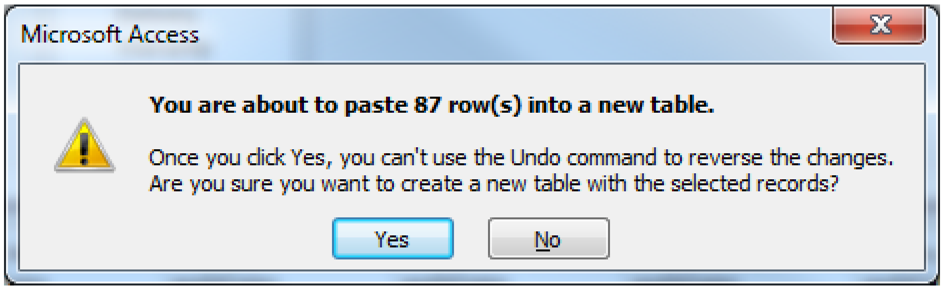


Figure 5.10

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?

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. **IMPORTANT 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.

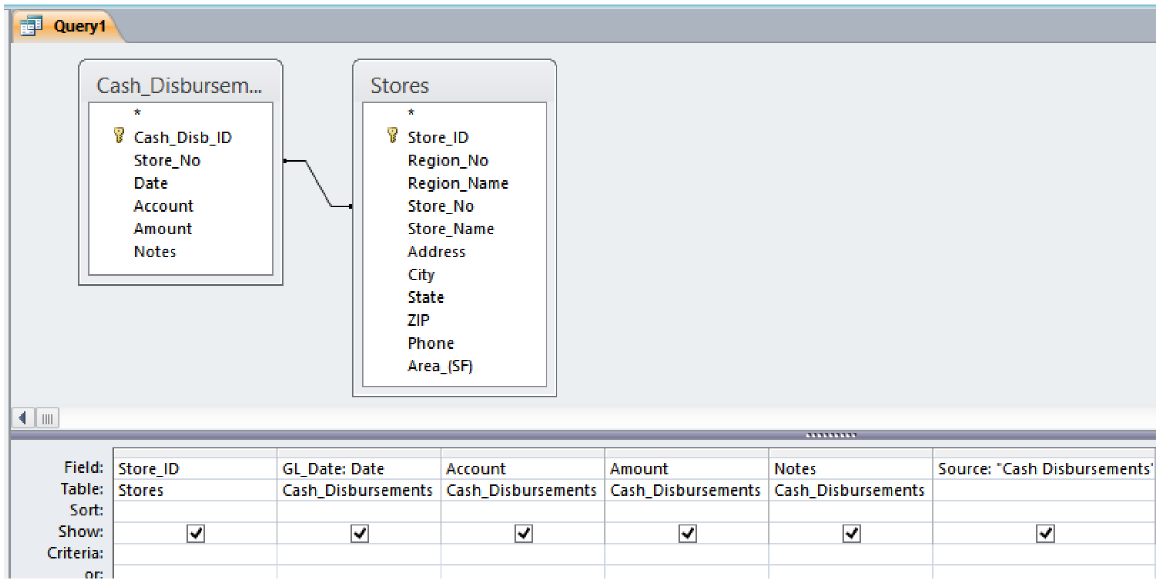
9. *Save and close* **qry05Sales_Budget**.

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_2010 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")



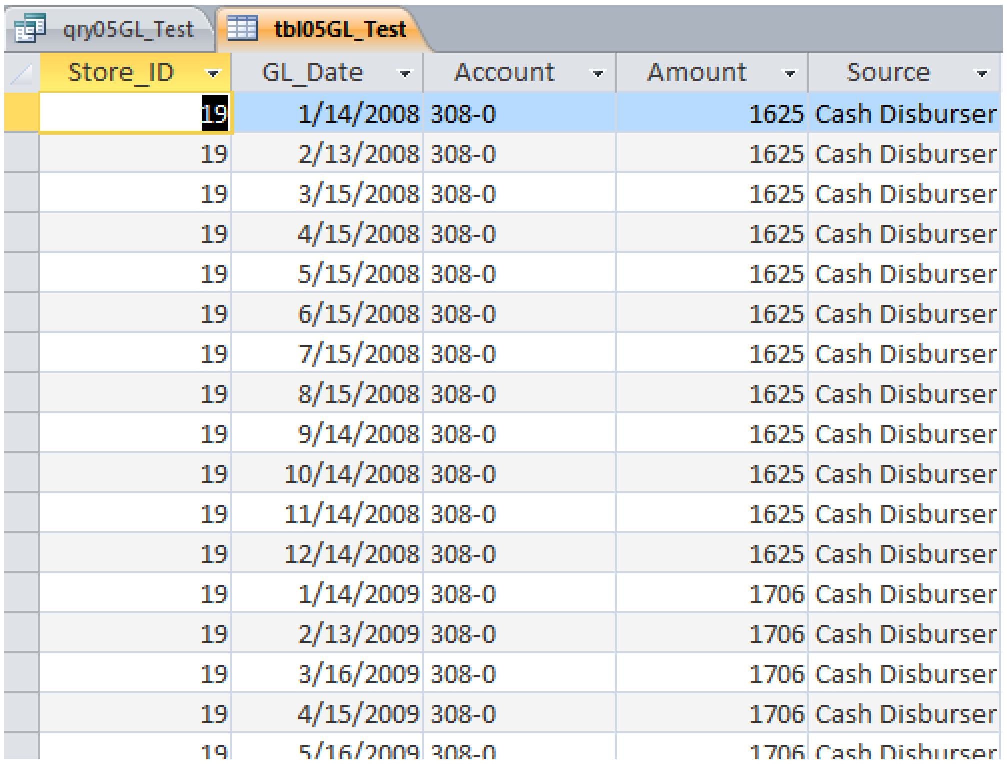
Field:	Store_ID	GL_Date: Date	Account	Amount	Notes	Source: "Cash Disbursements"
Table:	Stores	Cash_Disbursements	Cash_Disbursements	Cash_Disbursements	Cash_Disbursements	
Sort:						
Show:	✓	✓	✓	✓	✓	✓
Criteria:						
or:						

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.*



Store_ID	GL_Date	Account	Amount	Source
19	1/14/2008	308-0	1625	Cash Disburser
19	2/13/2008	308-0	1625	Cash Disburser
19	3/15/2008	308-0	1625	Cash Disburser
19	4/15/2008	308-0	1625	Cash Disburser
19	5/15/2008	308-0	1625	Cash Disburser
19	6/15/2008	308-0	1625	Cash Disburser
19	7/15/2008	308-0	1625	Cash Disburser
19	8/15/2008	308-0	1625	Cash Disburser
19	9/14/2008	308-0	1625	Cash Disburser
19	10/14/2008	308-0	1625	Cash Disburser
19	11/14/2008	308-0	1625	Cash Disburser
19	12/14/2008	308-0	1625	Cash Disburser
19	1/14/2009	308-0	1706	Cash Disburser
19	2/13/2009	308-0	1706	Cash Disburser
19	3/16/2009	308-0	1706	Cash Disburser
19	4/15/2009	308-0	1706	Cash Disburser
19	5/16/2009	308-0	1706	Cash Disburser

Figure 5.13

6. *Save and close the 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 2010 Review Questions Chapter 5, Section 1 of 2*** *option and complete the review questions.*

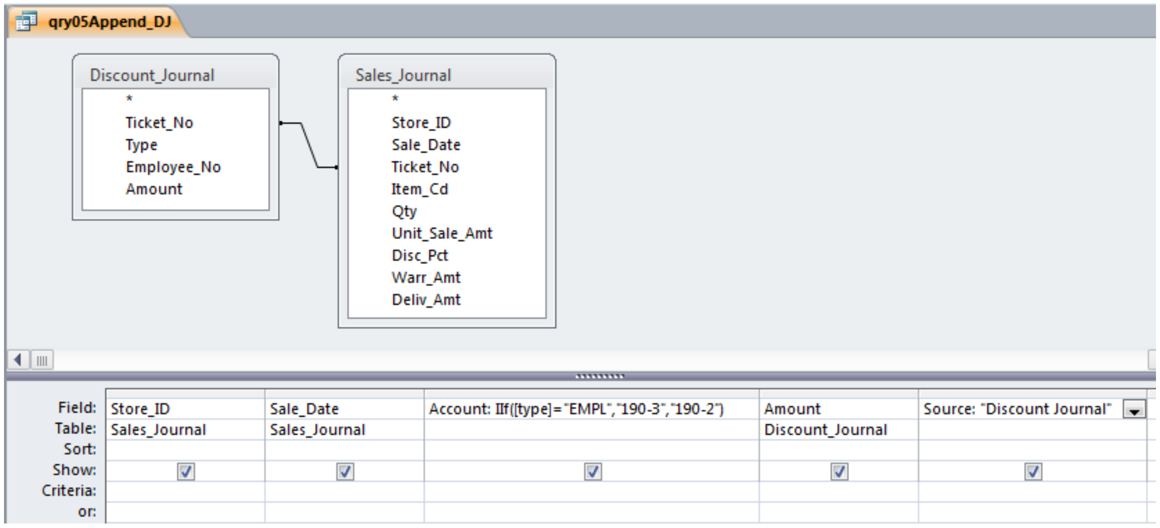
Append Query

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.

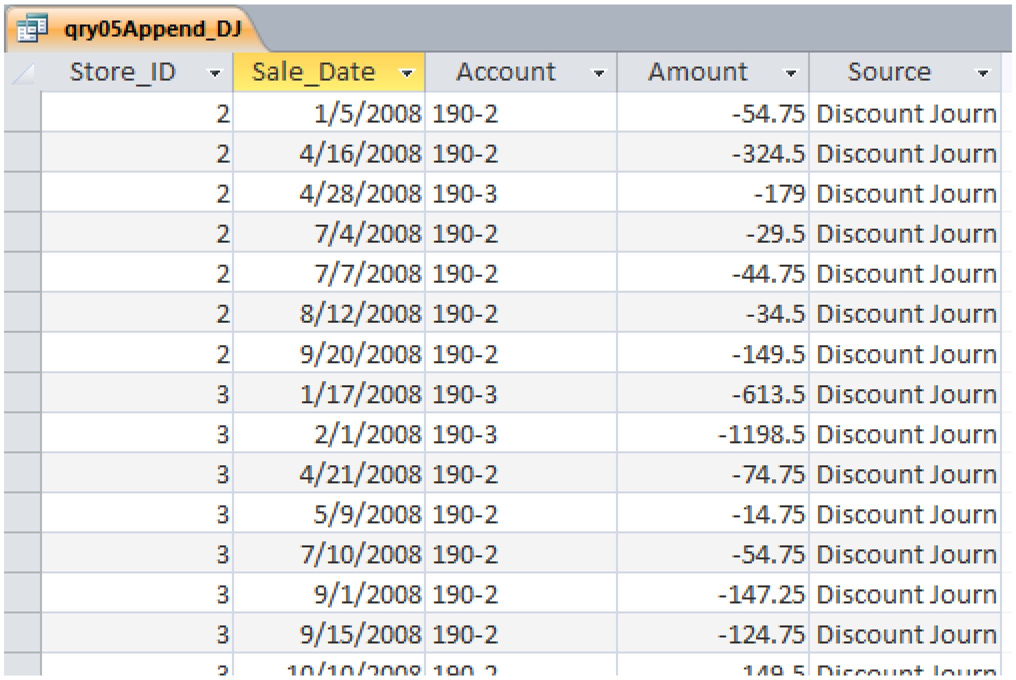
1. *Create a new query based on the* **Discount_Journal** *and* **Sales_Journal** *tables.*
2. *Create a join between the* **Discount_Journal** *and* **Sales_Journal** *using the* **Ticket_No** *field.*
3. *Bring the* **Store_ID** *and* **Sale_Date** *fields from the* **Sales_Journal** *into the* **Design** *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 that 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**.



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 5.14

8. *Run the query.*


qry05Append_DJ

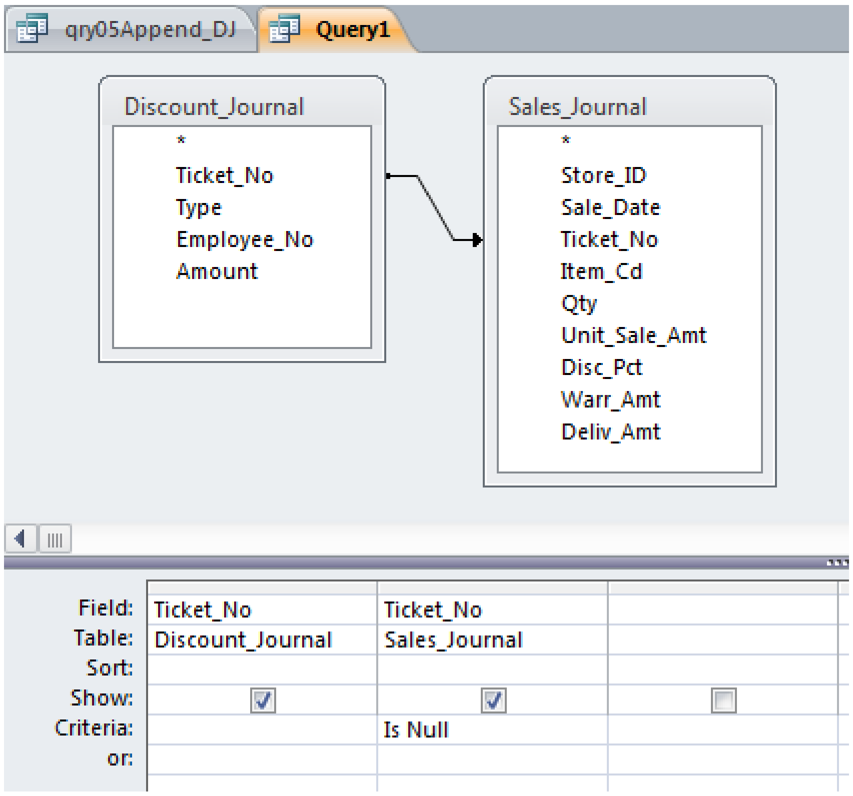
Store_ID	Sale_Date	Account	Amount	Source
2	1/5/2008	190-2	-54.75	Discount Journ
2	4/16/2008	190-2	-324.5	Discount Journ
2	4/28/2008	190-3	-179	Discount Journ
2	7/4/2008	190-2	-29.5	Discount Journ
2	7/7/2008	190-2	-44.75	Discount Journ
2	8/12/2008	190-2	-34.5	Discount Journ
2	9/20/2008	190-2	-149.5	Discount Journ
3	1/17/2008	190-3	-613.5	Discount Journ
3	2/1/2008	190-3	-1198.5	Discount Journ
3	4/21/2008	190-2	-74.75	Discount Journ
3	5/9/2008	190-2	-14.75	Discount Journ
3	7/10/2008	190-2	-54.75	Discount Journ
3	9/1/2008	190-2	-147.25	Discount Journ
3	9/15/2008	190-2	-124.75	Discount Journ

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, that should be no problem. We'll just use the one-way join trick you learned in Chapter 4.

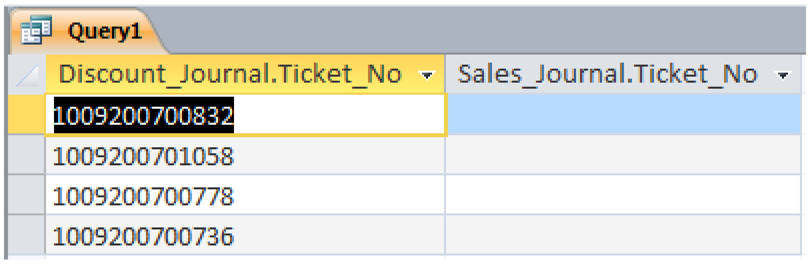
1. *Create a new query using the* **Discount_Journal** *and* **Sales_Journal** *tables.*
2. *Do a* **one-way join** *to include all records from the* **Discount_Journal** *table and the corresponding records from the* **Sales_Journal** *table on the* **Ticket_No** *field.*
3. *Bring in the* **Ticket_No** *fields from both tables into the* **Design** *grid.*
4. *In the* **Criteria** *section for the* **Sales_Journal.Ticket_No** *field, type* **Is Null**



Field:	Ticket_No	Ticket_No	
Table:	Discount_Journal	Sales_Journal	
Sort:			
Show:	✓	✓	☐
Criteria:		Is Null	
or:			

Figure 5.16

5. *Run the query.*



Discount_Journal.Ticket_No	Sales_Journal.Ticket_No
1009200700832	
1009200701058	
1009200700778	
1009200700736	

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

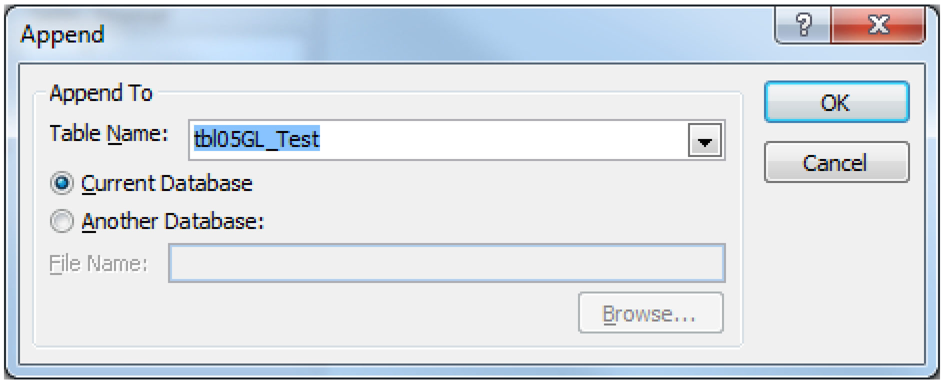
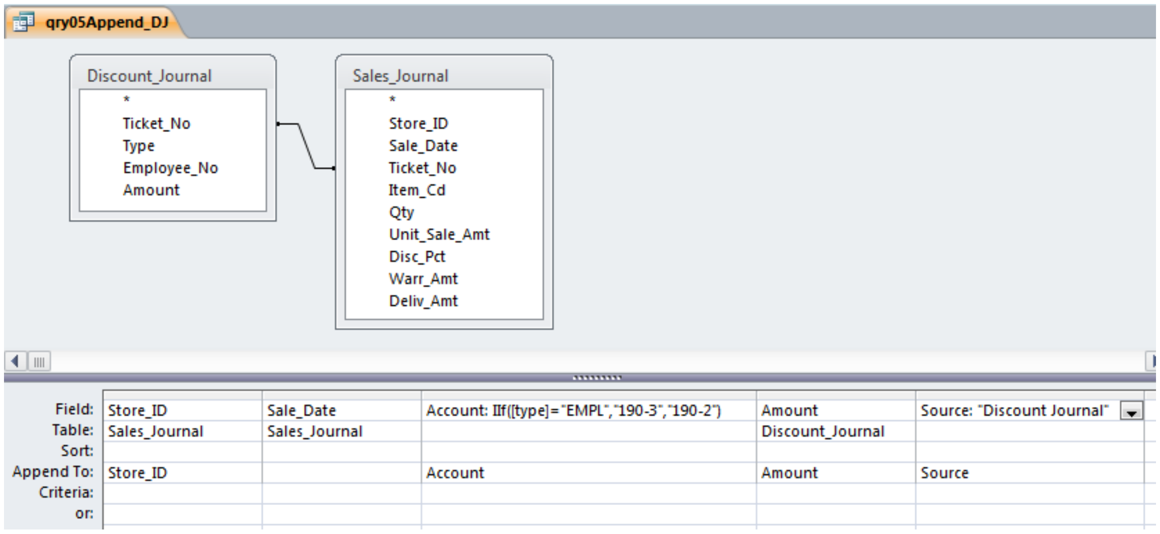


Figure 5.18

9. *Click* **OK**.

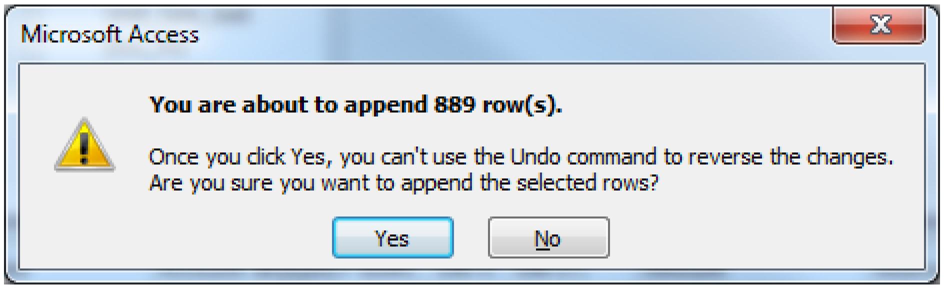


Field:	Store_ID	Sale_Date	Account: IIf([type]="EMPL","190-3","190-2")	Amount	Source: "Discount Journal"
Table:	Sales_Journal	Sales_Journal		Discount_Journal	
Sort:					
Append To:	Store_ID		Account	Amount	Source
Criteria:					
or:					

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.*


Microsoft Access

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?

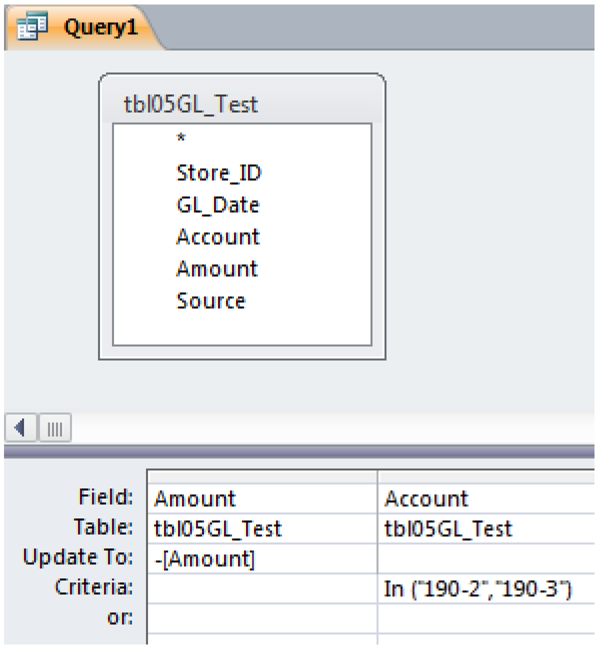
Figure 5.20

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]**

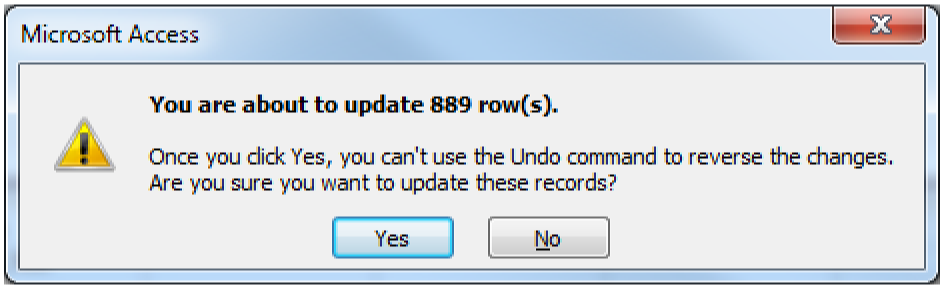


Field:	Amount	Account
Table:	tbl05GL_Test	tbl05GL_Test
Update To:	-[Amount]	
Criteria:		In ("190-2","190-3")
or:		

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 by clicking the* **Run** *icon.*



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.

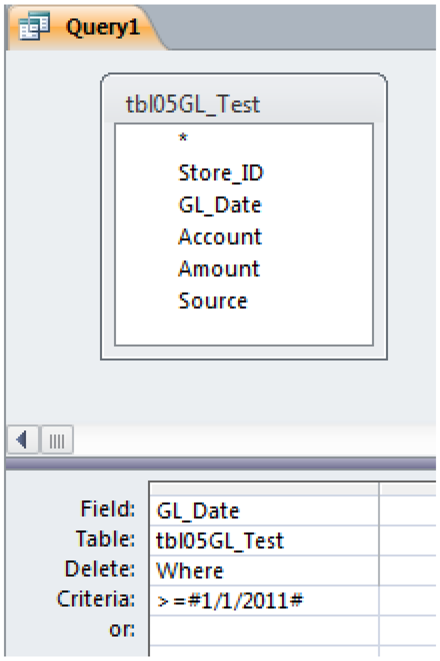
8. *Save the query as* **qry05Update_GL** *and close it.*
9. *To make sure you don't repeat the same mistake again, edit the* **Amount** *field in the* **qry05Append_DJ** *to* **Amt: -[Amount]**
10. *Save and close* **qry05Append_DJ.**

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 2011, so we'll write a Delete Query which will delete all records with GL dates on or after 1/1/2011.

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* **>=1/1/2011** *(Access will change the code to >=#1/1/2011#)*



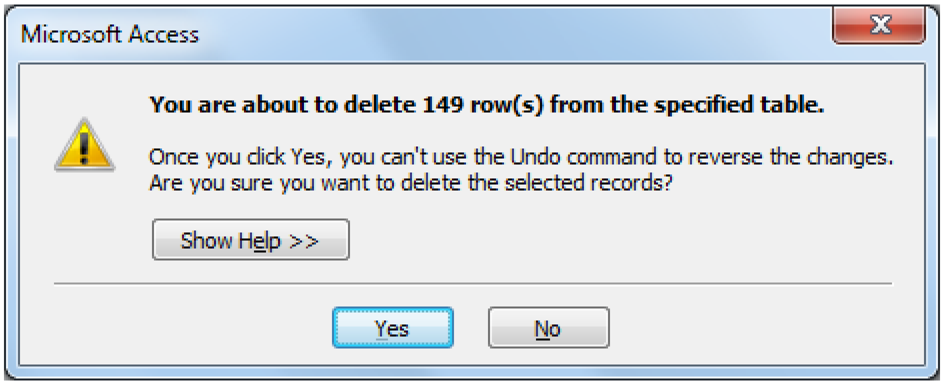
Field:	GL_Date
Table:	tbl05GL_Test
Delete:	Where
Criteria:	>=#1/1/2011#
or:	

Figure 5.23

5. *View the record set in* **View** *mode, but don't run the query yet.*

You should get 149 records with dates greater than or equal to 1/1/2011. 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?

Figure 5.24

8. *Click* **Yes.**
9. *Save the query as* **qry05Delete_2011.**
10. *Close the 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.*

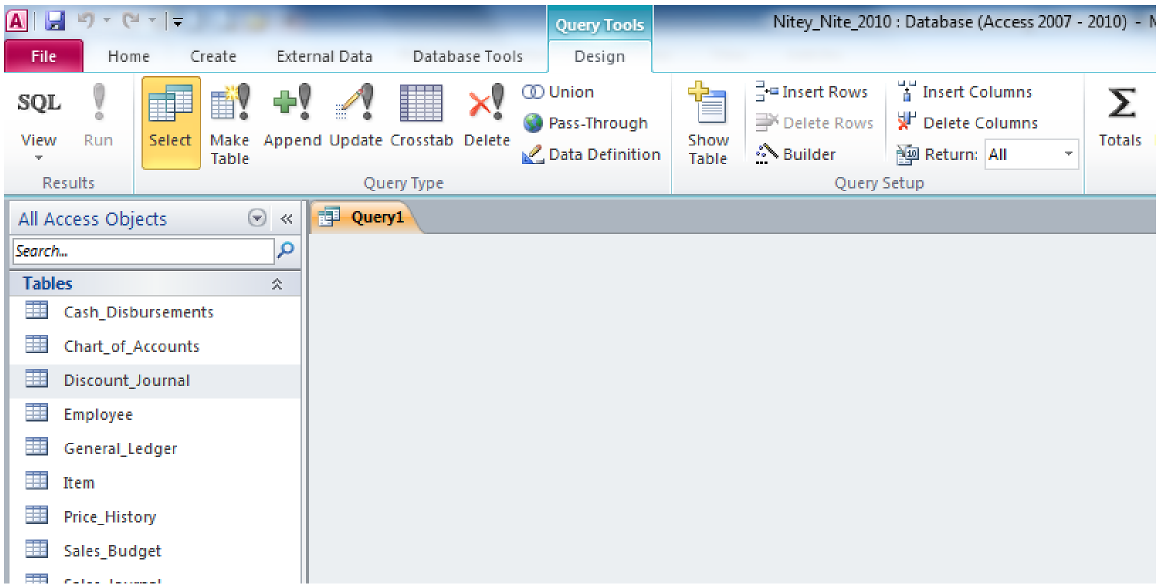
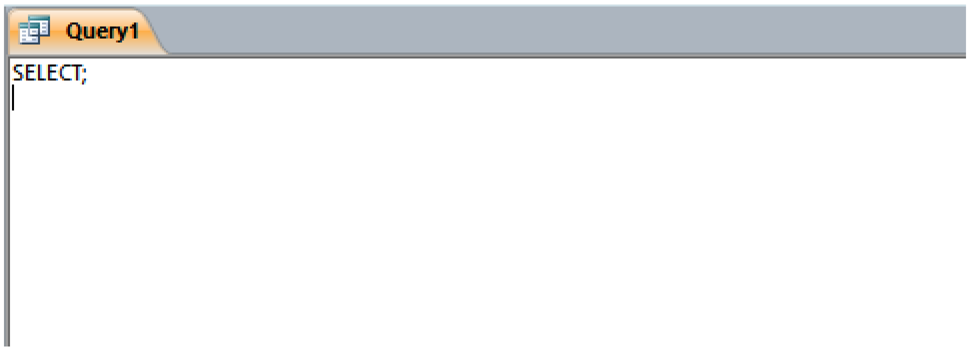


Figure 5.25

If you look on the left side of the 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** *icon.*

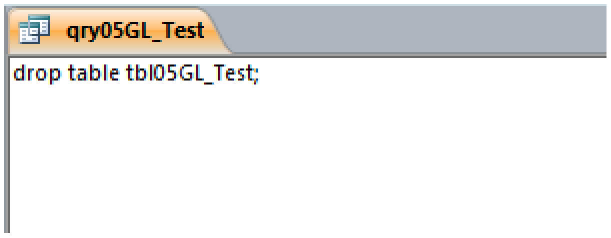


```
SELECT;
```

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** *icon and save the query as* **qry05Drop_Table.**



```
drop table tbl05GL_Test;
```

Figure 5.27

6. *Close the 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 someone opens the database. 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 2010** *folder (or whichever folder on which you stored this course) and click* **OK***.*
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 that 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** *from the* **Macros & Code** *group.*

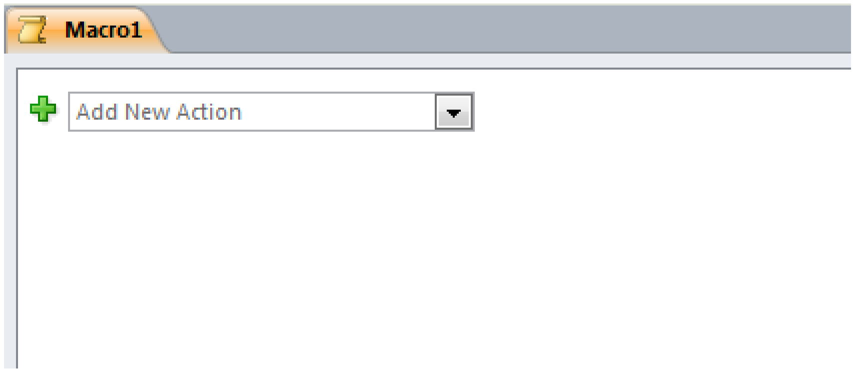
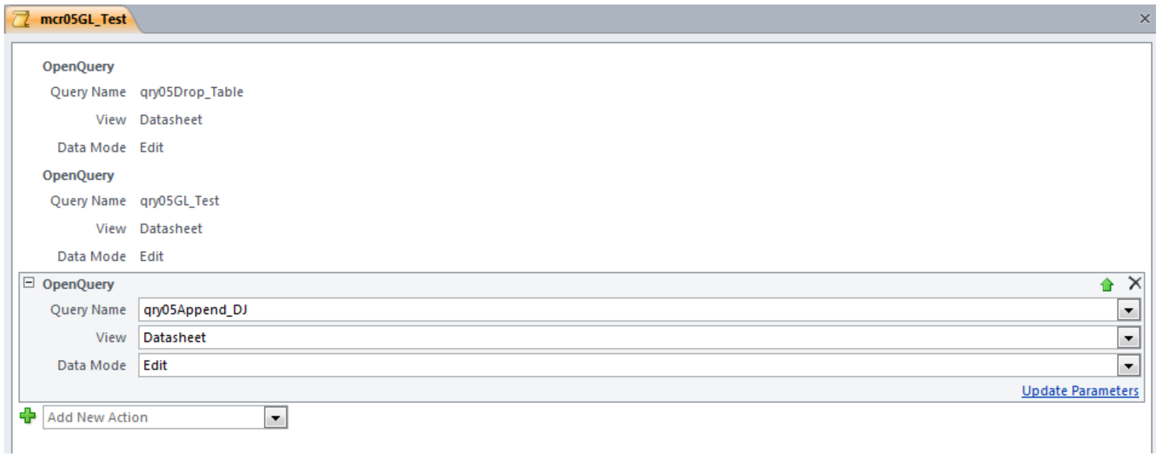


Figure 5.28

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** *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

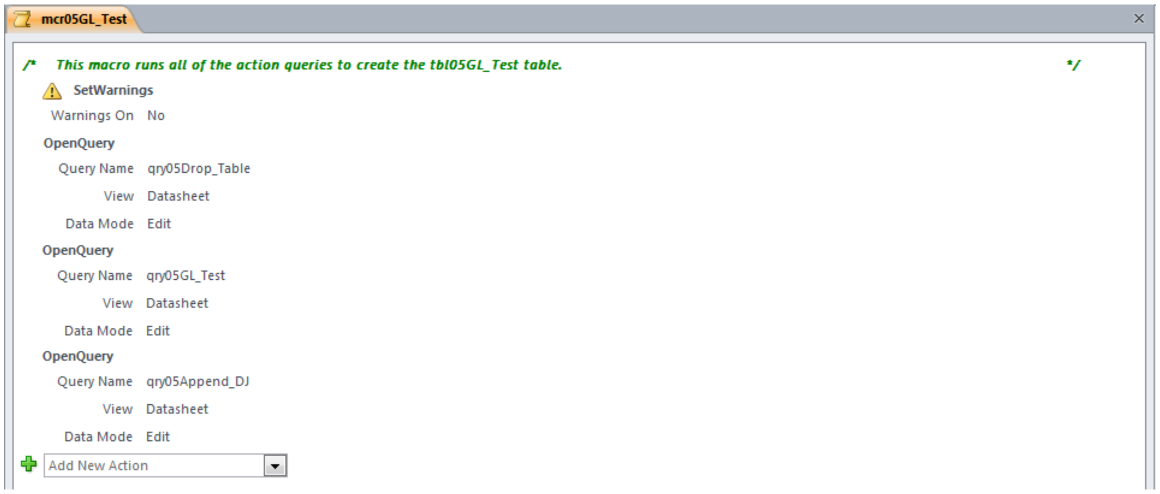
12. *Close the macro.*
13. *In the newly created* **Macros** *section (located below the* **Forms** *section) in the Navigation pane, double-click* **mcr05GL_Test.**

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 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.*


mcr05GL_Test

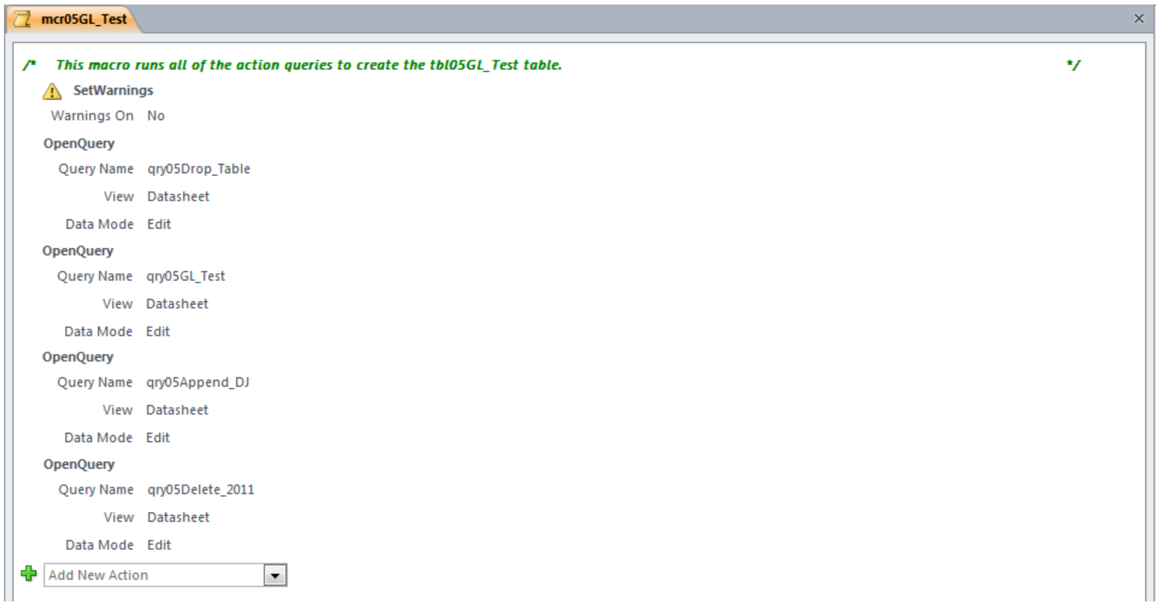
```
/* This macro runs all of the action queries to create the tbl05GL_Test table.  */
SetWarnings
  Warnings On  No
OpenQuery
   Query Name  qry05Drop_Table
         View  Datasheet
    Data Mode  Edit
OpenQuery
   Query Name  qry05GL_Test
         View  Datasheet
    Data Mode  Edit
OpenQuery
   Query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

Figure 5.30

22. *Save and run the macro.*

If you open tbl05GL_Test, it should contain 9,298 records. We forgot to run the query that takes out the 2011 data, but you can easily edit the macro to include that query.

23. *Edit the macro to include* **qry05Delete_2011** *as the last action.*
24. *Save, close and test the macro.*



```
mcr05GL_Test
/*  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_2011
          View  Datasheet
     Data Mode  Edit
  Add New Action
```

Figure 5.31

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 2010 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, log in and take the exam. Make sure that you take the exam on the same computer that you completed the sample files on, as some of the questions on the exam may refer to some of the completed examples.

ExcelCEO
Chief Excel Officer

Access 2010 and SQL

Complete Self-study Course

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.

Introduction to Forms

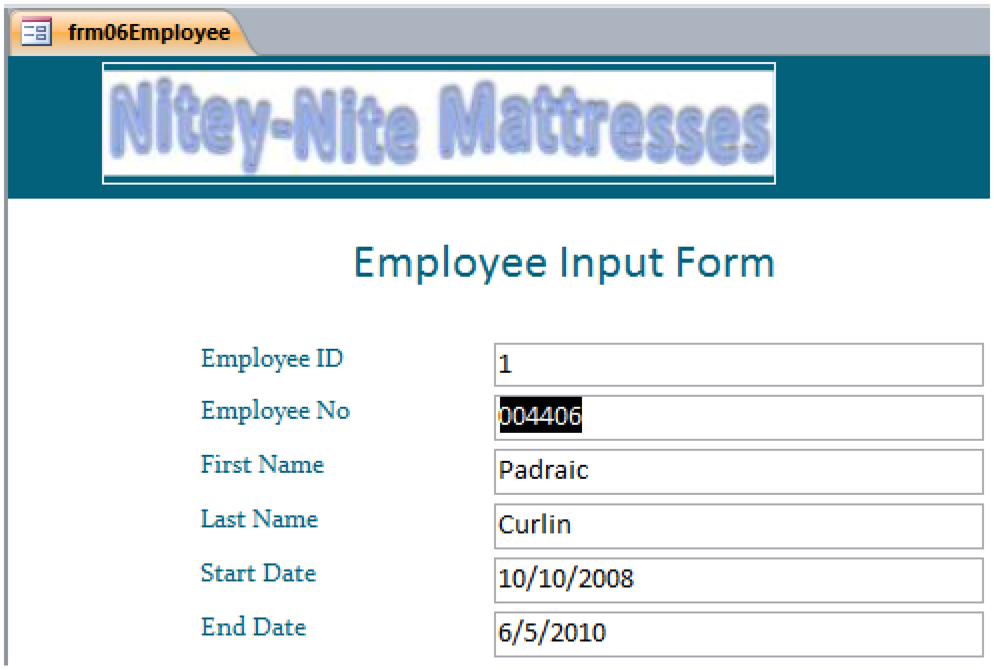
As long as you are the only person who will ever use your database (which is most likely *not* the case), you will probably be comfortable inputting data directly into tables and navigating around the different tabs. However, if you need to get someone to help you with entering data or running reports, 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_2010** *database under the* **Forms** *tab, double-click on* **frmEmployee** *to open it.*


frm06Employee

Nitey-Nite Mattresses

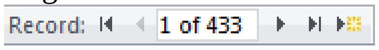
Employee Input Form

Employee ID	1
Employee No	004406
First Name	Padraic
Last Name	Curlin
Start Date	10/10/2008
End Date	6/5/2010

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, 2008, and she ended her employment with Nitey-Nite on June 5, 2010.

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** 

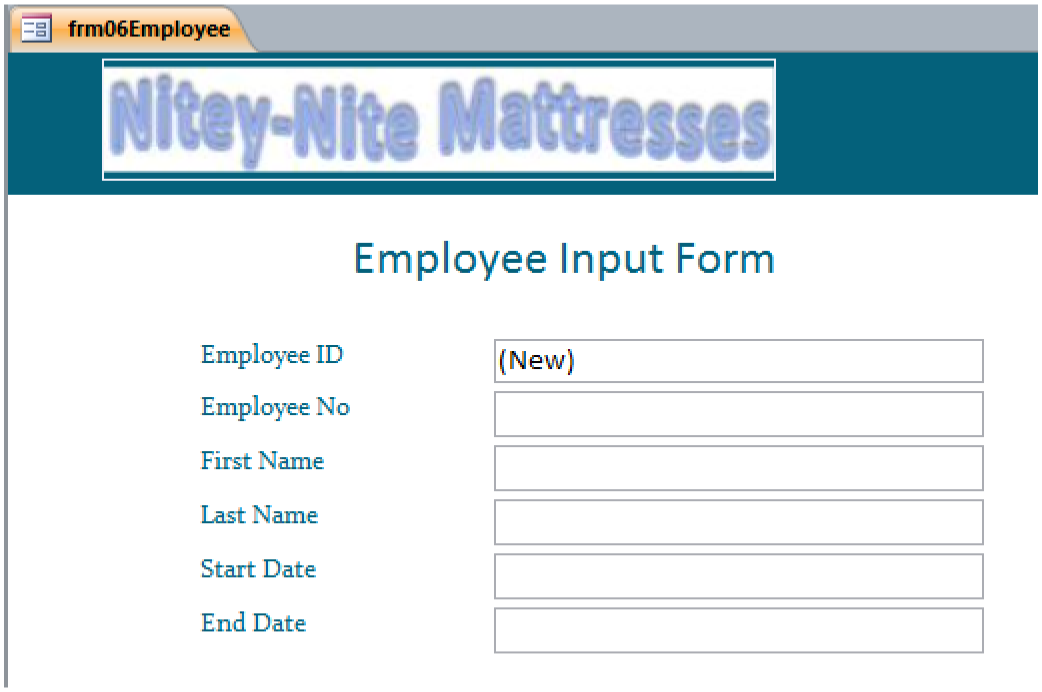
are located at the bottom left of the form. It is similar to the tab selectors in Excel, but it displays (or sets focus on) one individual record at a time. You can click the first record, previous record, next record, last record, or new record selectors to scroll through each record in the table. Lastly, the designer 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.*



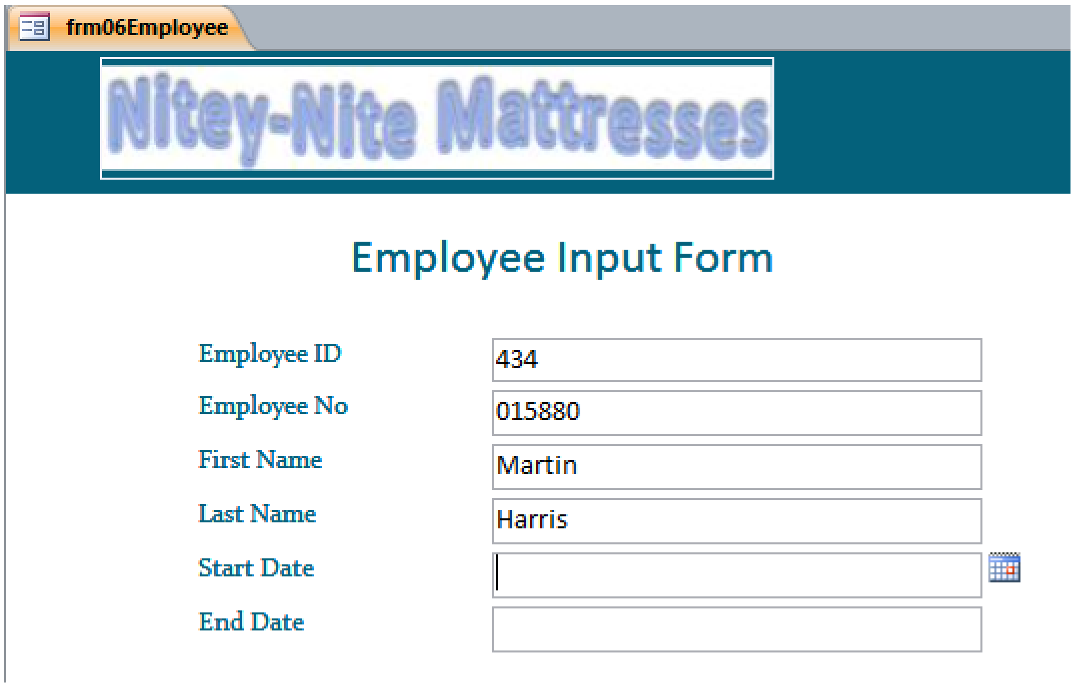
Employee Input Form

Employee ID	(New)
Employee No	
First Name	
Last Name	
Start Date	
End Date	

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*


frm06Employee

Nitey-Nite Mattresses

Employee Input Form

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 is a new feature in Access 2010.

5. *Click on the* **Calendar control** *next to the* **Start Date box** *and choose* **January 12, 2010.**
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	David	Oliver
Last Name	Whitmer	Cowdery
Start Date	1/15/2010	1/20/2010
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. 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 on the drop-down arrow under the* **View** *icon and choose* **Design View**.

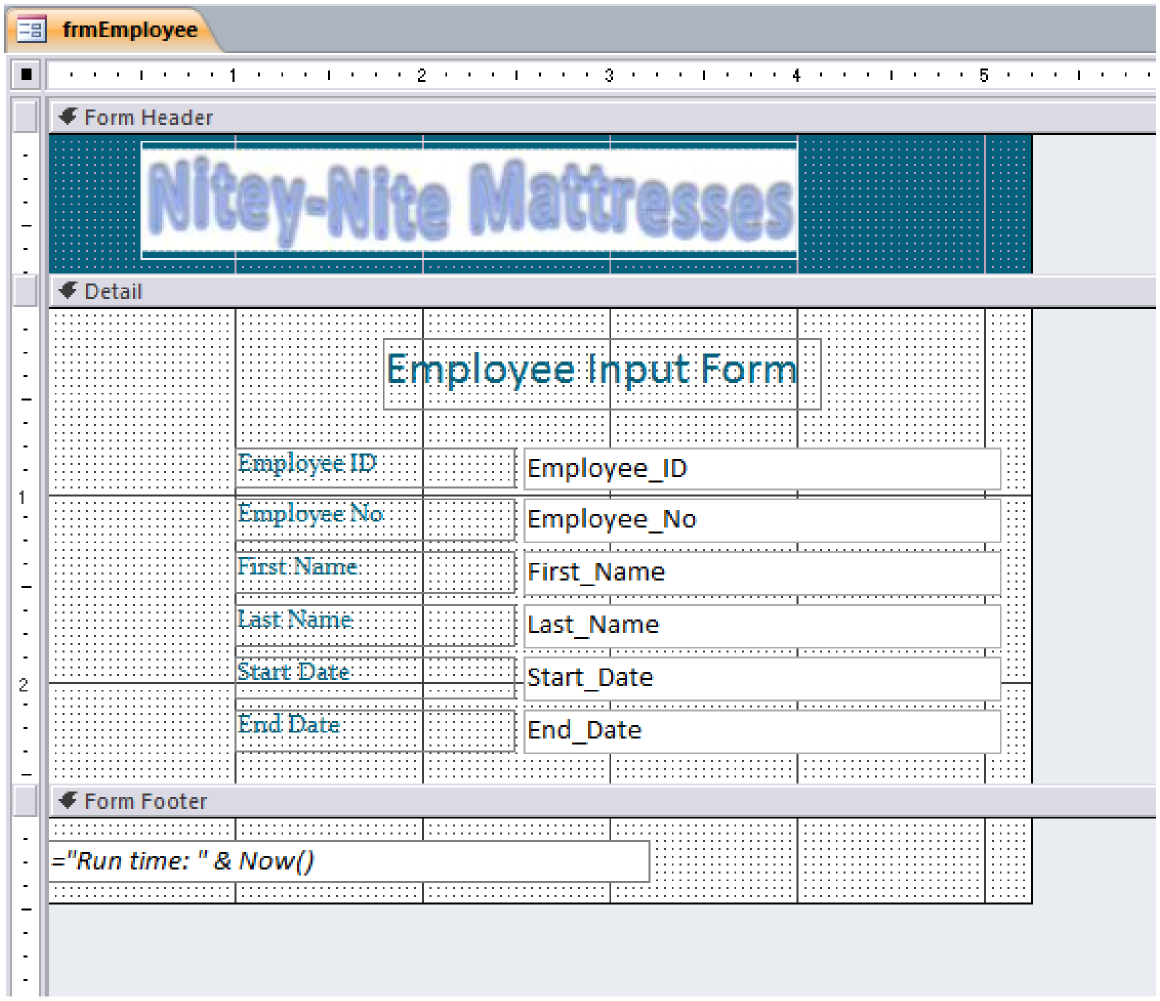


Figure 6.4

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 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 jpeg file that can be imported into a variety of applications, including Access forms and reports. Here the form designer imported the logo.jpg 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 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

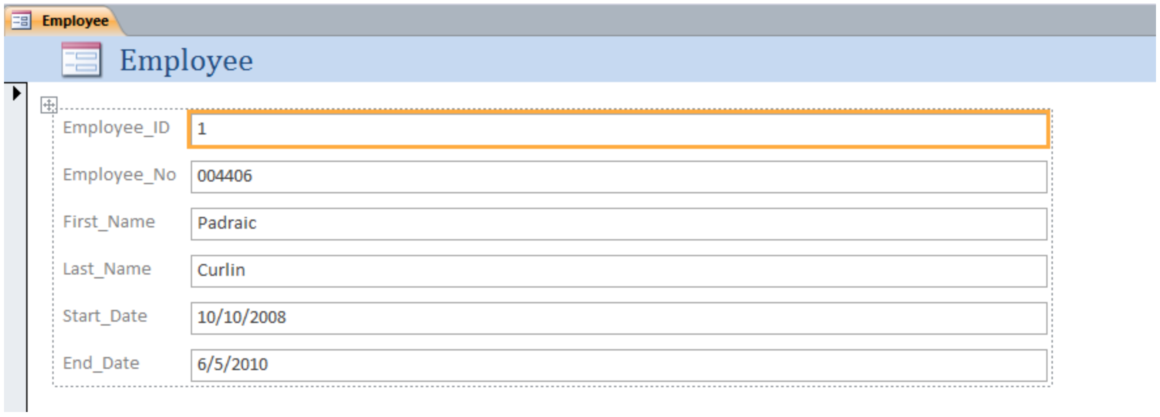
As with Excel, Access 2010 contains a ribbon that houses the various icons available for use within the form design view. You are already very familiar with the ribbon and using icons, so I won't go into an in-depth explanation. Most of the icons you should already know, and it will not be a difficult 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 2010, 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 2010, 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.*



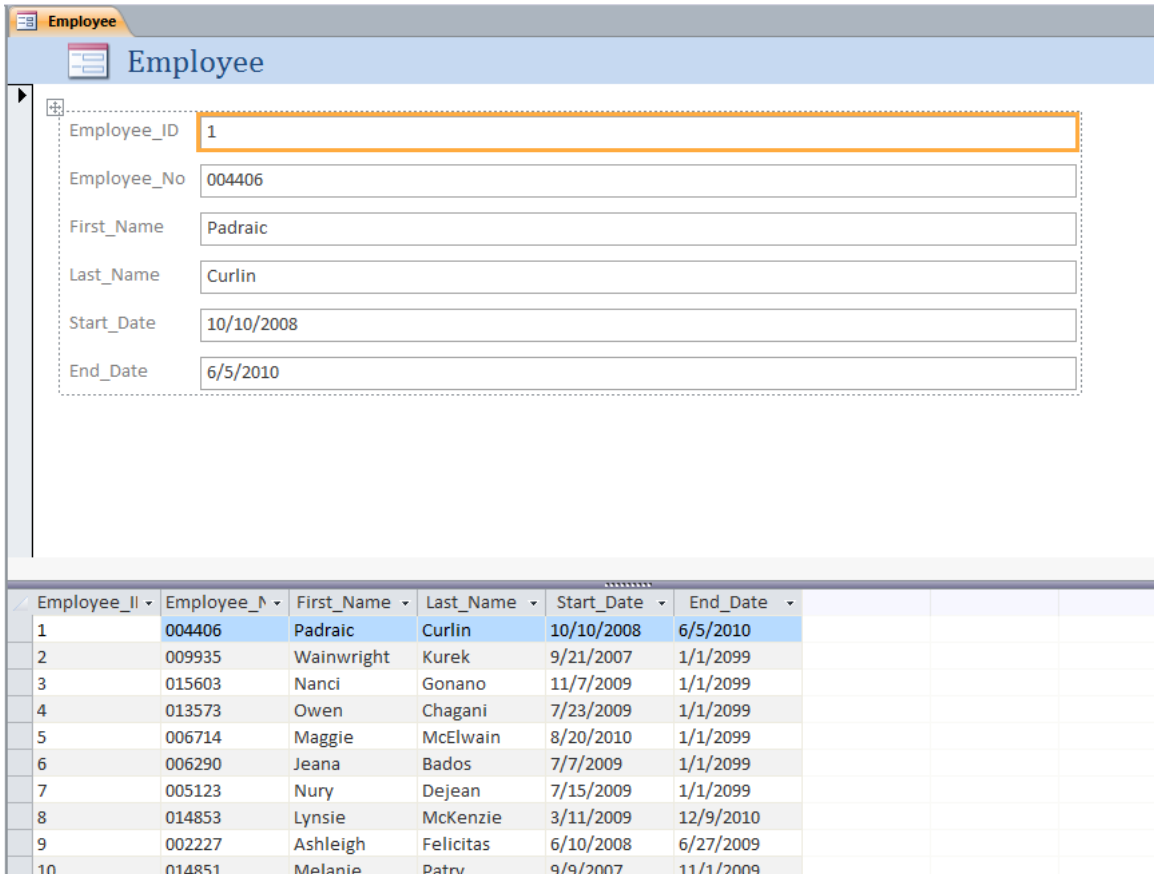
Employee_ID	1
Employee_No	004406
First_Name	Padraic
Last_Name	Curlin
Start_Date	10/10/2008
End_Date	6/5/2010

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 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* **form** *without saving it.*
5. *In the* **Create** *tab, click on the* **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	10/10/2008
End_Date	6/5/2010

Employee_II	Employee_N	First_Name	Last_Name	Start_Date	End_Date
1	004406	Padraic	Curlin	10/10/2008	6/5/2010
2	009935	Wainwright	Kurek	9/21/2007	1/1/2099
3	015603	Nanci	Gonano	11/7/2009	1/1/2099
4	013573	Owen	Chagani	7/23/2009	1/1/2099
5	006714	Maggie	McElwain	8/20/2010	1/1/2099
6	006290	Jeana	Bados	7/7/2009	1/1/2099
7	005123	Nury	Dejean	7/15/2009	1/1/2099
8	014853	Lynsie	McKenzie	3/11/2009	12/9/2010
9	002227	Ashleigh	Felicitas	6/10/2008	6/27/2009
10	014851	Melanie	Patry	9/9/2007	11/1/2009

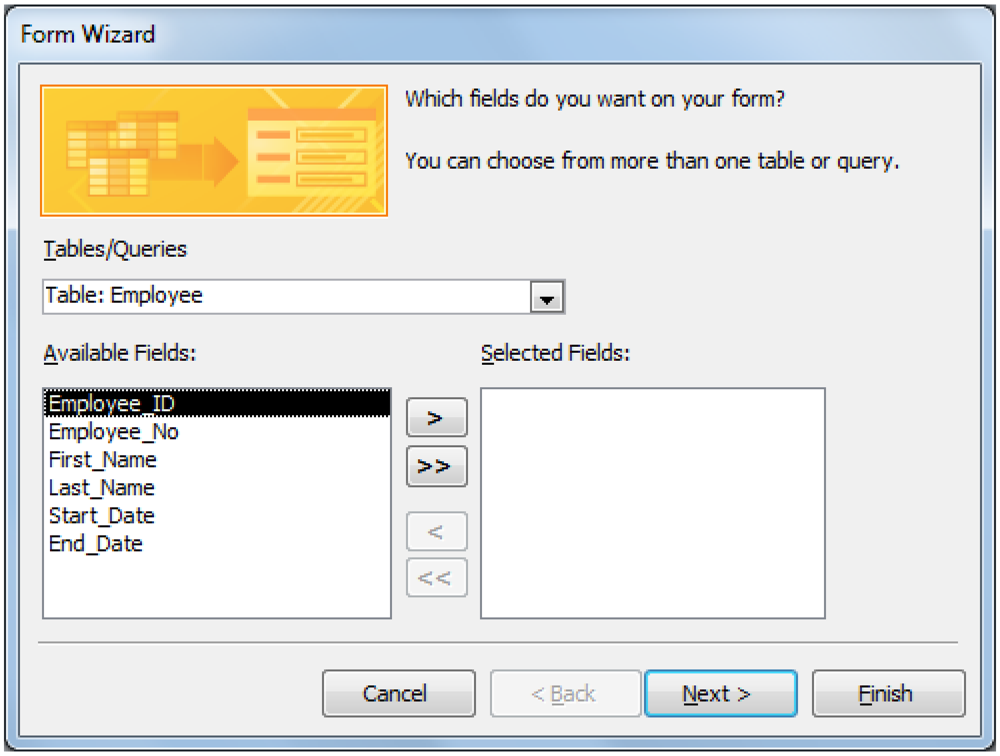
Figure 6.6

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* **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 2010 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.

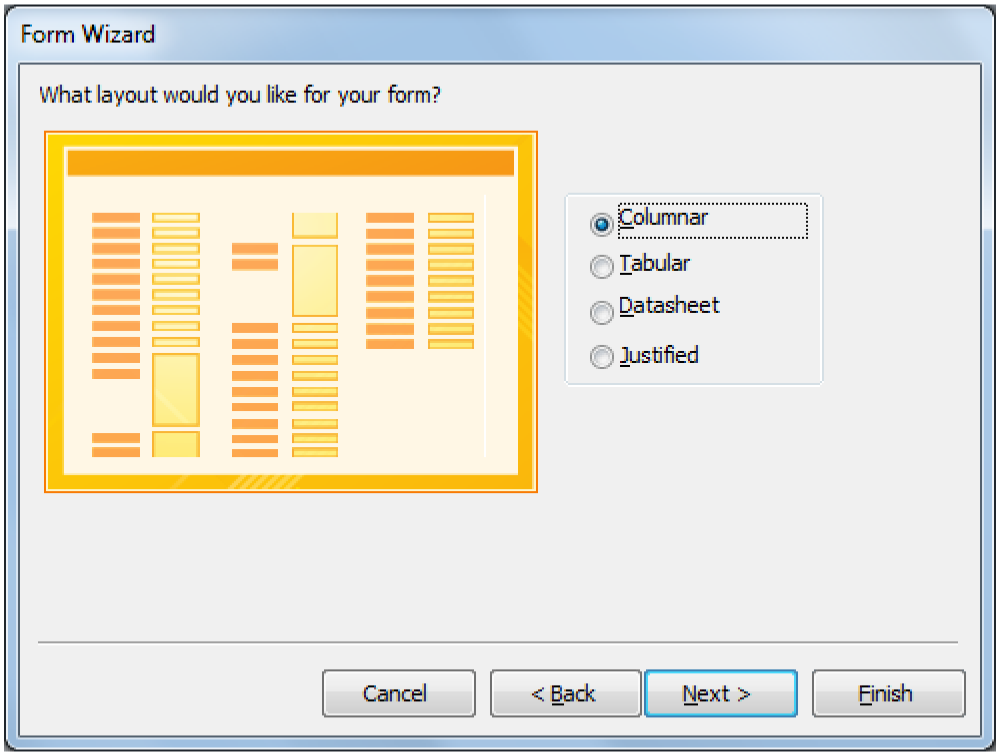
Tables/Queries: Table: Employee

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 >.**

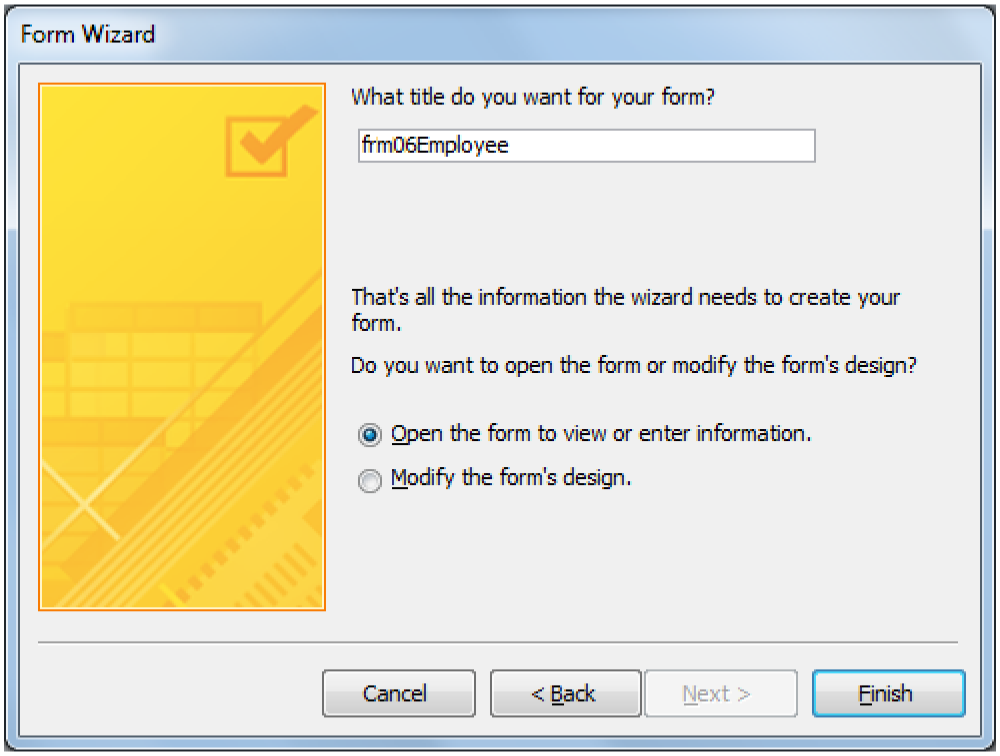


What layout would you like for your form?

[✓ Columnar] Tabular Datasheet Justified

Figure 6.8

3. *Make sure* **Columnar** *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.

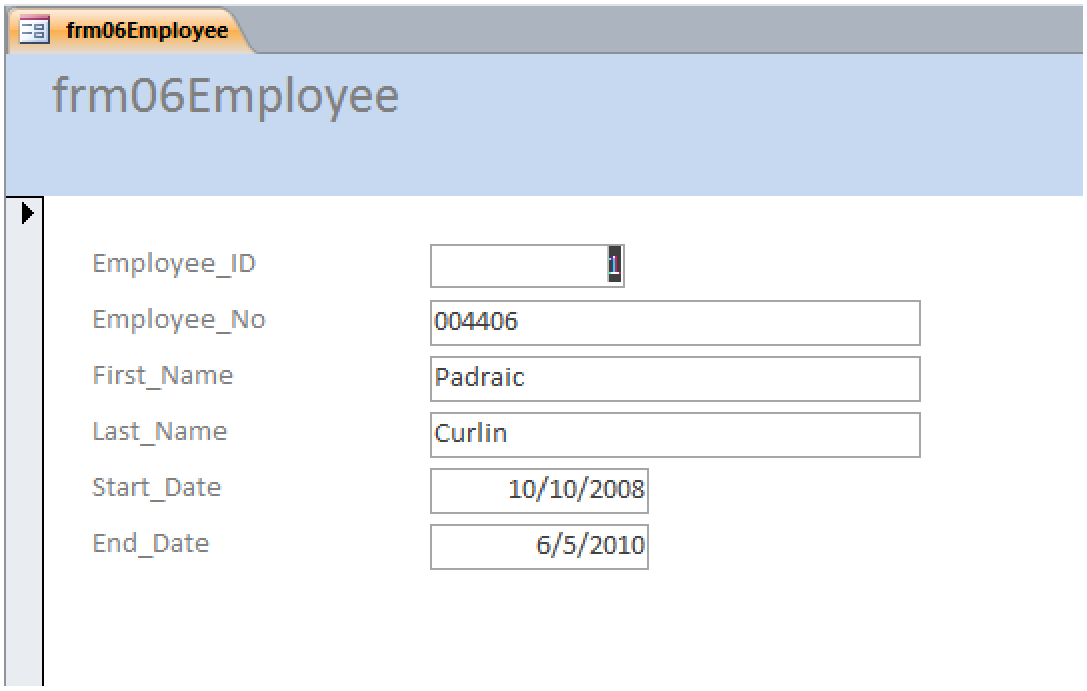
Do you want to open the form or modify the form's design?

[✓ 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	10/10/2008
End_Date	6/5/2010

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 2010 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 the* **form**.

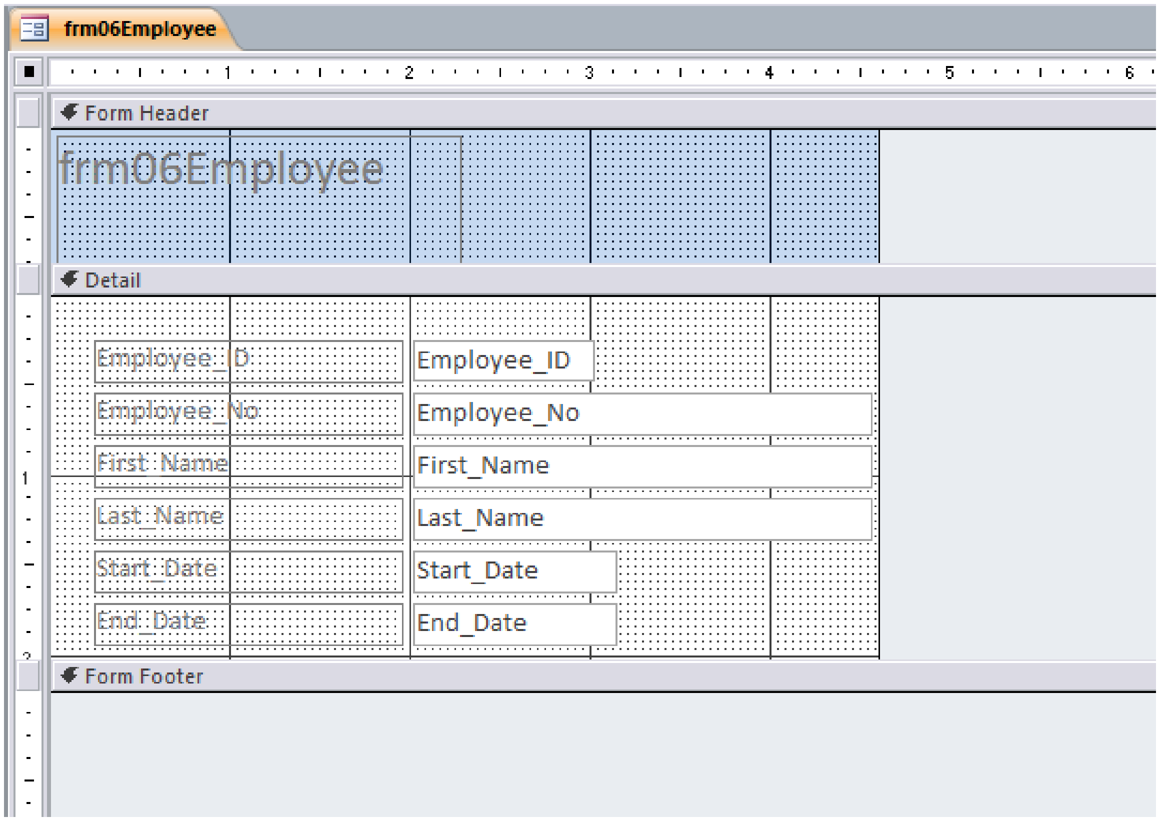


Figure 6.11

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 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]** *key on your keyboard (you can also right-click on the label and choose* **Delete***).*
8. *Click on the* **Form Header** *bar.*

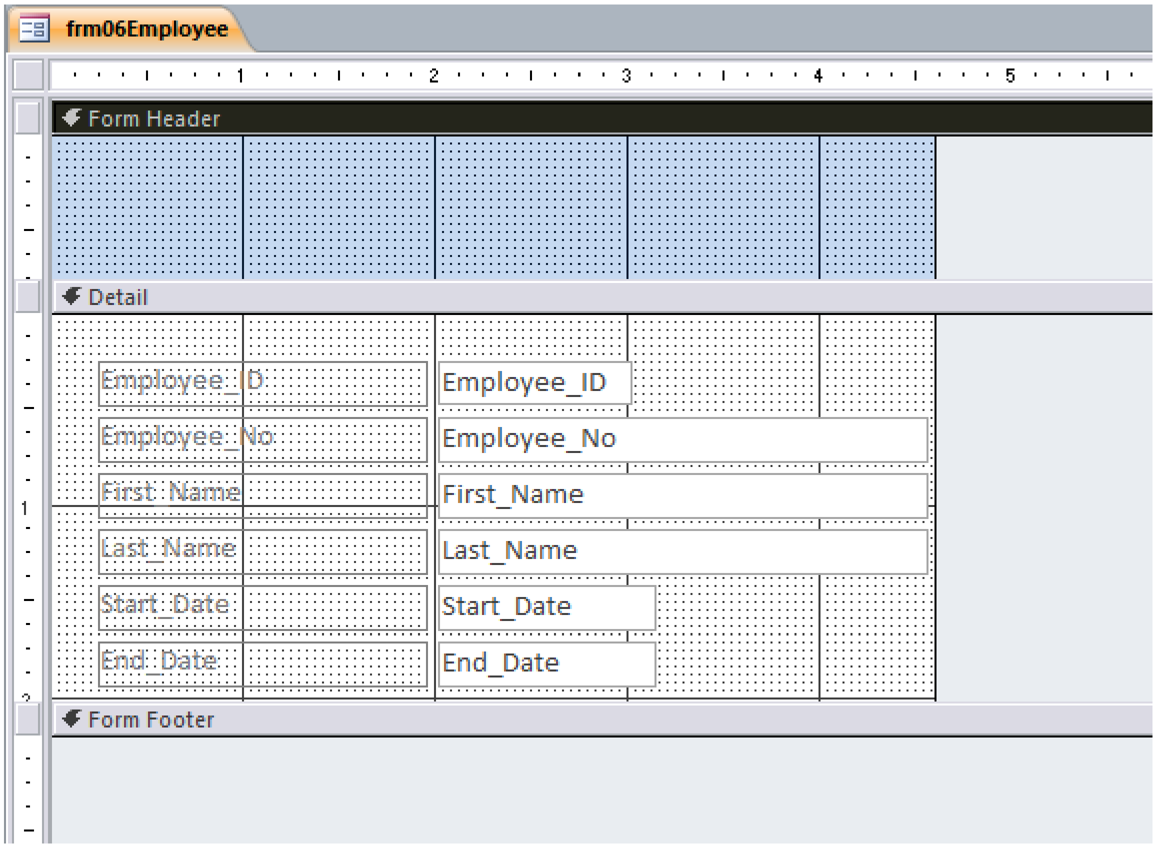


Figure 6.12

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

9. *Under the* **Form Design Tools Design** *contextual tab, click on the* **Logo** *icon in the* **Header / Footer** *group.*
10. *Navigate to the* **C:\ExcelCEO\Access 2010** *folder and double-click on the* **logo.jpg** *file.*
11. *Click on the bottom right handle on the graphic to resize it to look like* **Figure 6.13**.

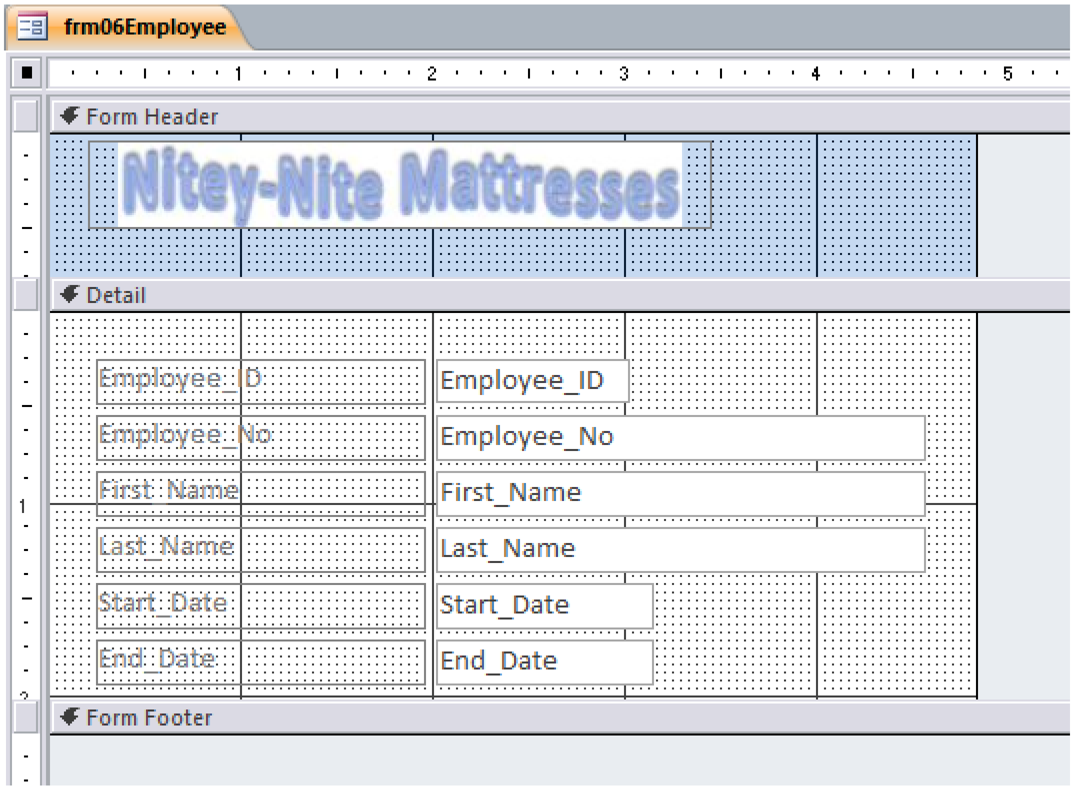


Figure 6.13

> ***Note**: Depending on the version of Access you have, you may have imported a label in addition to the logo. If you did, you will notice the size of the form increased to the right, and next to the logo there will be a faint dotted rectangular box. The label text is hidden by the logo. Click in the area of the dotted rectangular box and press [Delete] to get rid of the imported label.*

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* **5¼”** *mark on the top ruler.*
14. *Expand the* **Detail** *section of the report 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.

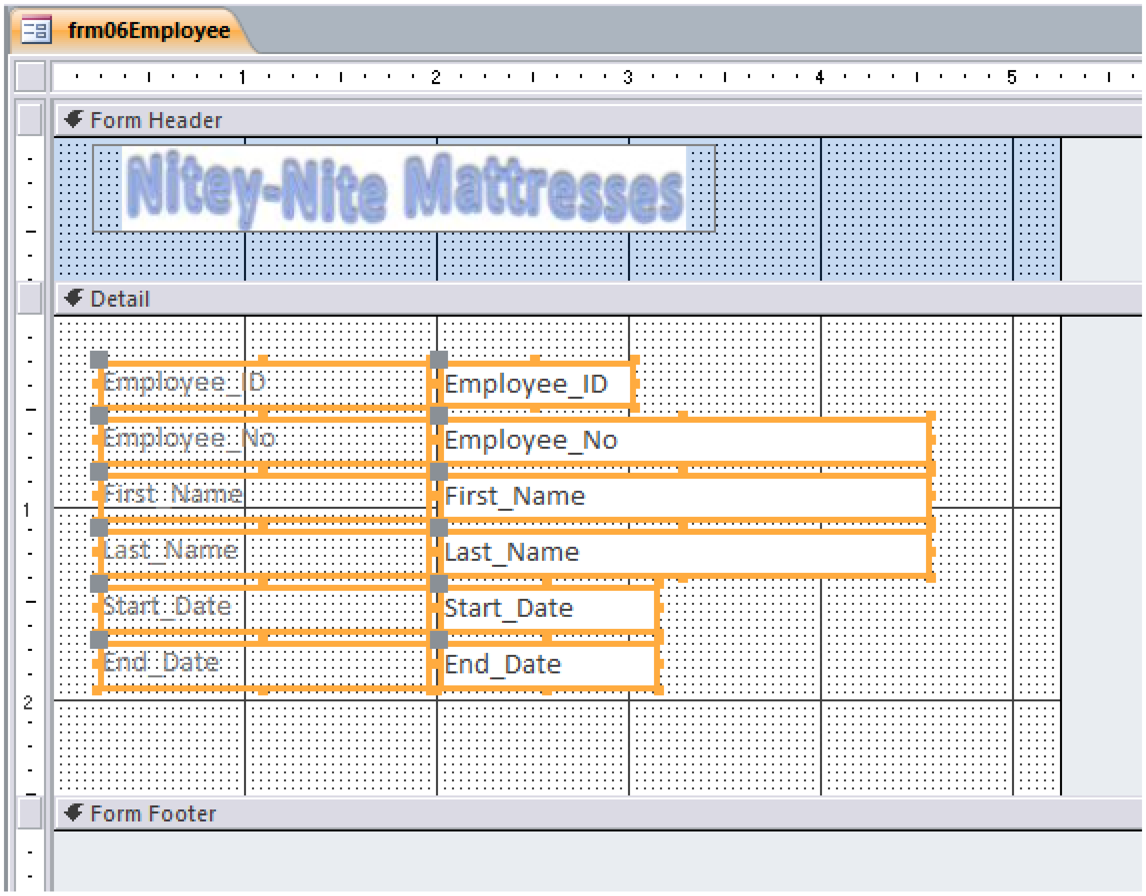


Figure 6.14

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 under 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.*

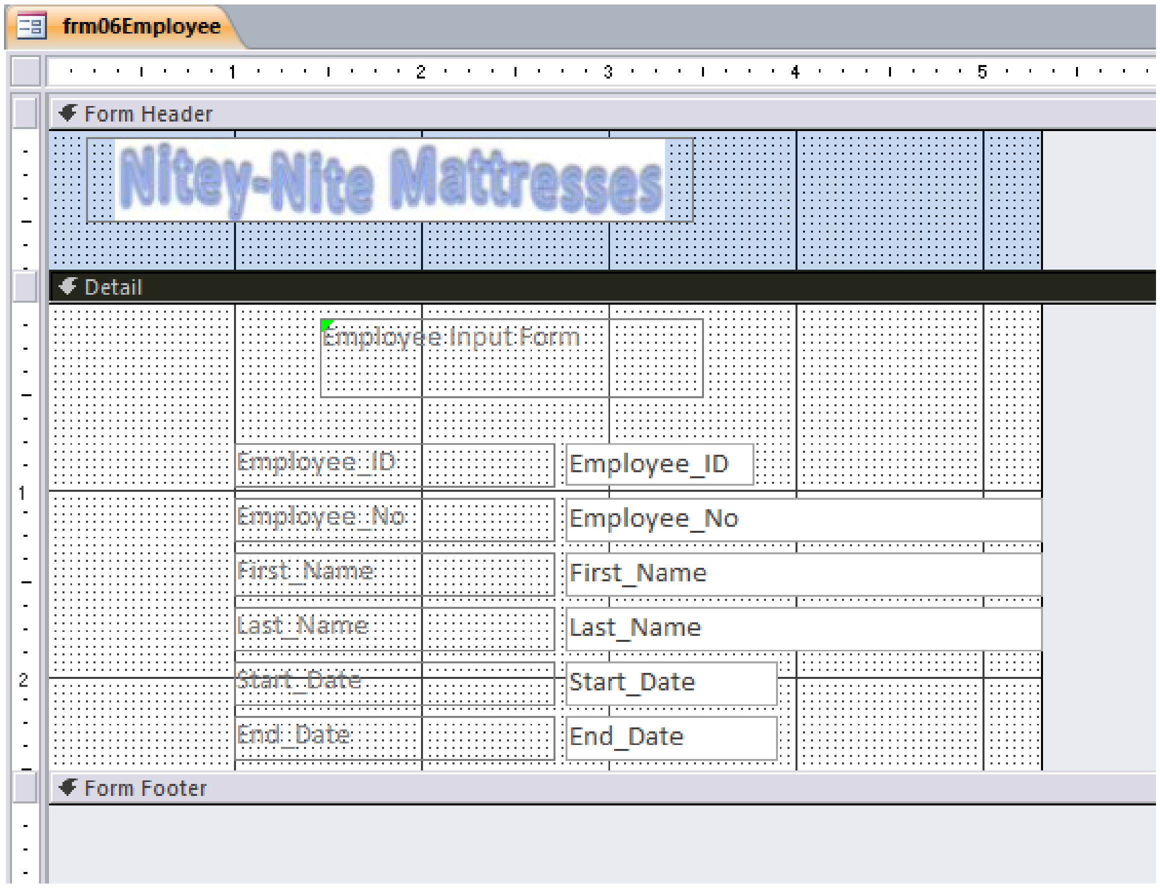


Figure 6.15

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** *and change the* **Font Name** *to* **Calibri**, *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.*

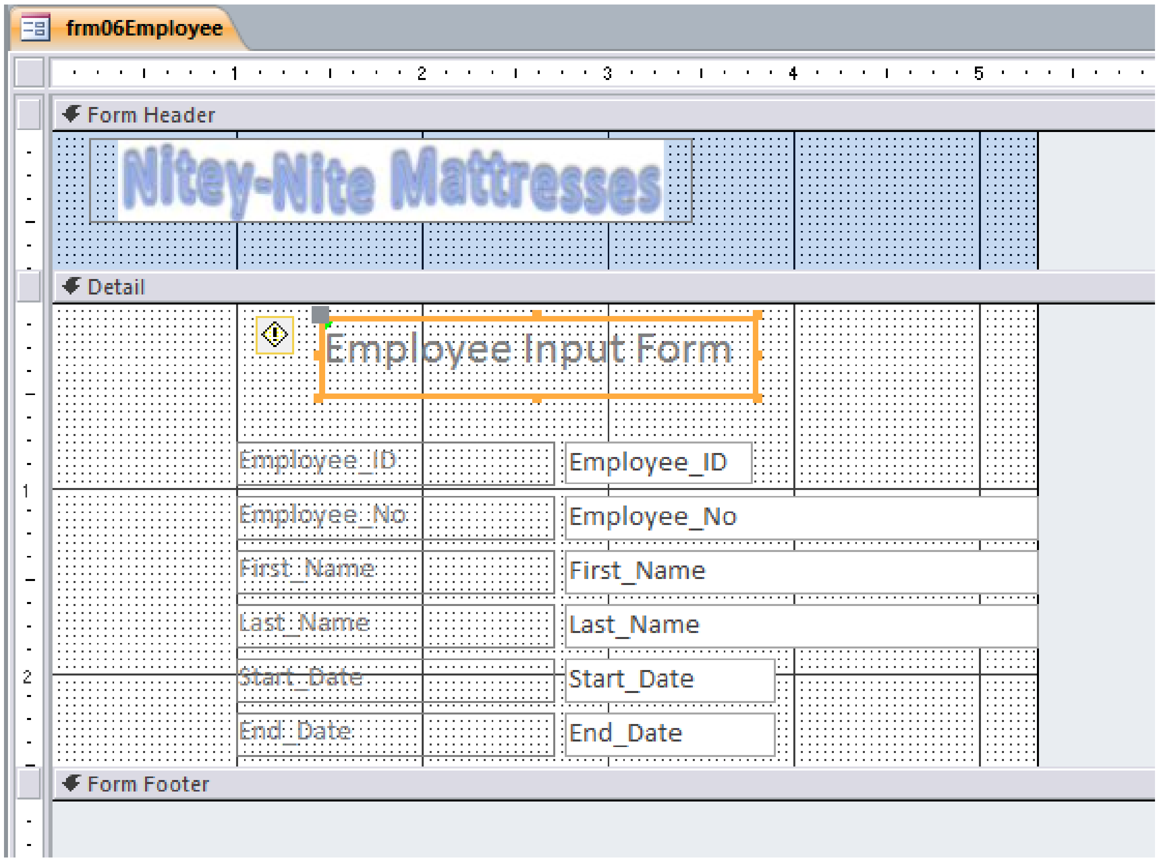


Figure 6.16

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 have 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** *<u>label</u> (not the text box) and change the text to "***Employee ID***".*
27. *Change all of the other labels as shown in* **Figure 6.17**.

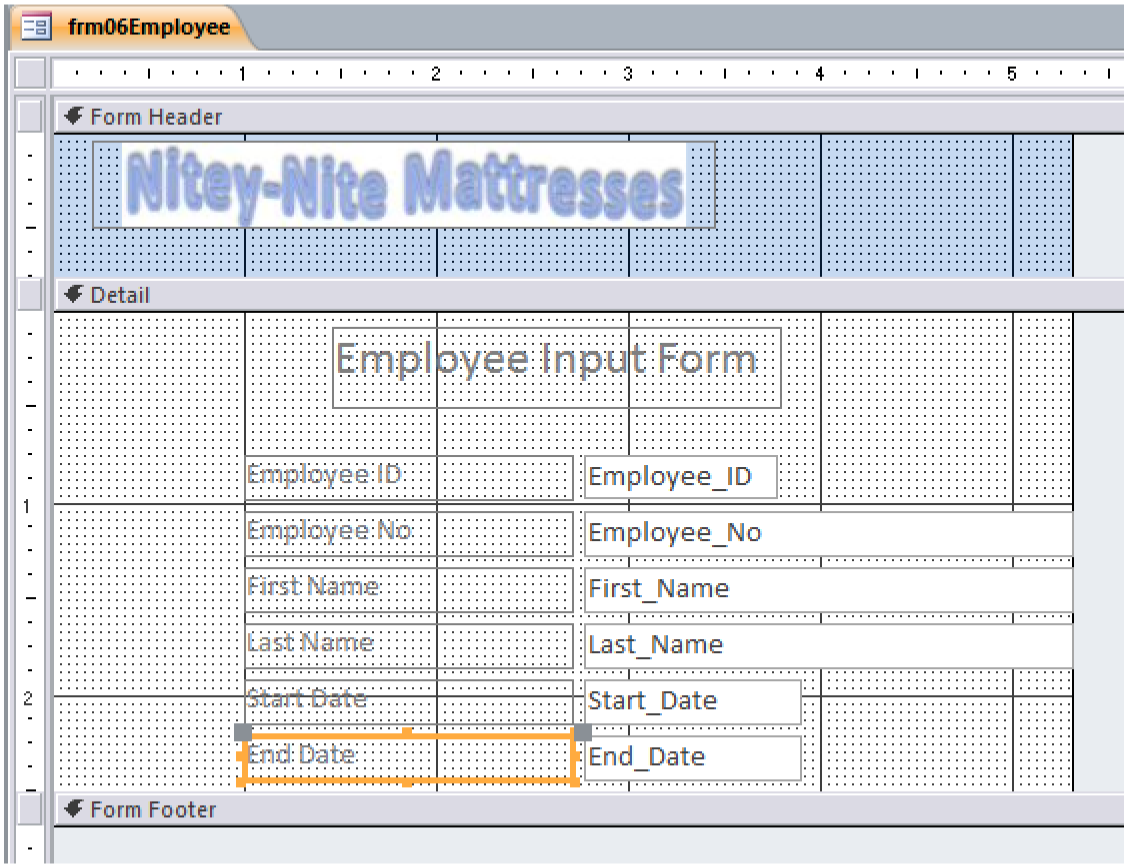
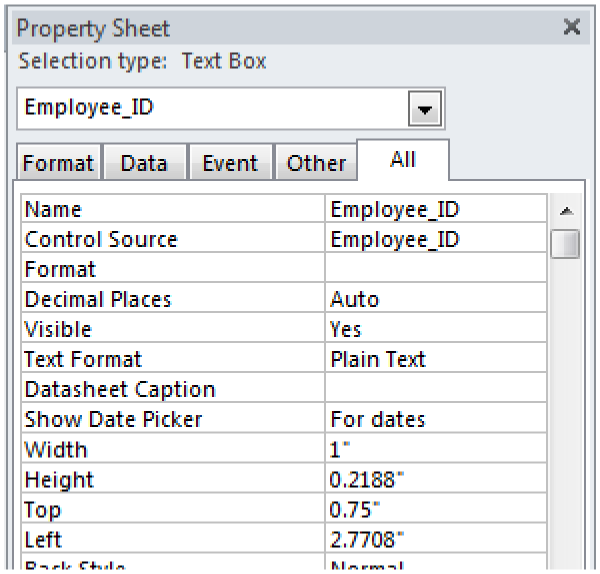


Figure 6.17

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**.

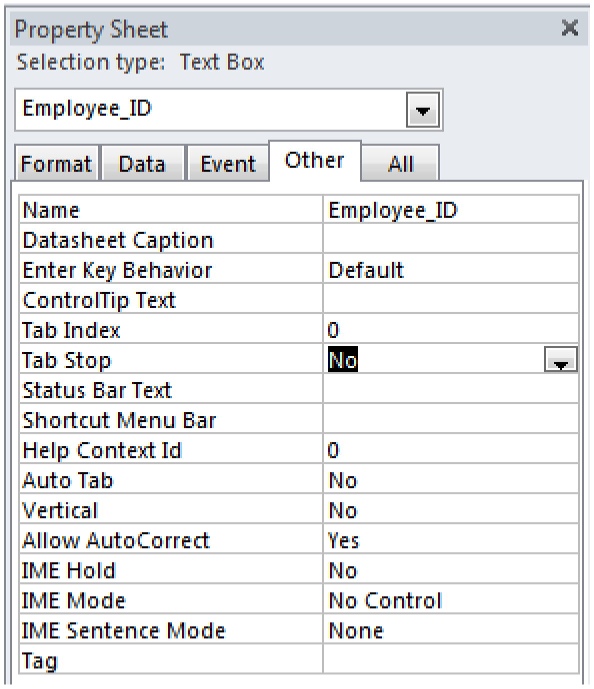
28. *Click on the* **Employee_ID** *text box and make sure the* **Property Sheet** *is displayed.*


Property Sheet
Selection type: Text Box
Employee_ID
Format | Data | Event | Other | All

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"

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**.


Property Sheet
Selection type: Text Box
Employee_ID
Format | Data | Event | Other | All

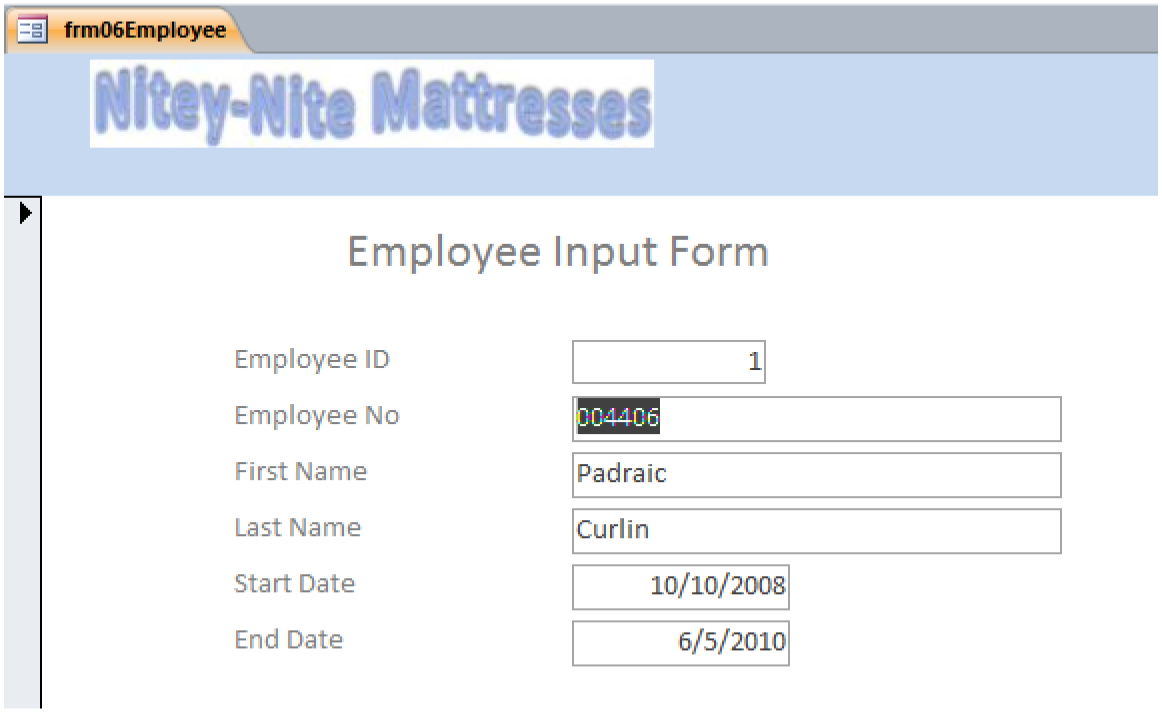
Name	Employee_ID
Datasheet Caption	
Enter Key Behavior	Default
ControlTip Text	
Tab Index	0
Tab Stop	No
Status Bar Text	
Shortcut Menu Bar	
Help Context Id	0
Auto Tab	No
Vertical	No
Allow AutoCorrect	Yes
IME Hold	No
IME Mode	No Control
IME Sentence Mode	None
Tag	

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. *Close the* **Property Sheet** *dialog box and click the* **View** *icon to display the form in* **Form View**.



frm06Employee

Niteey-Nite Mattresses

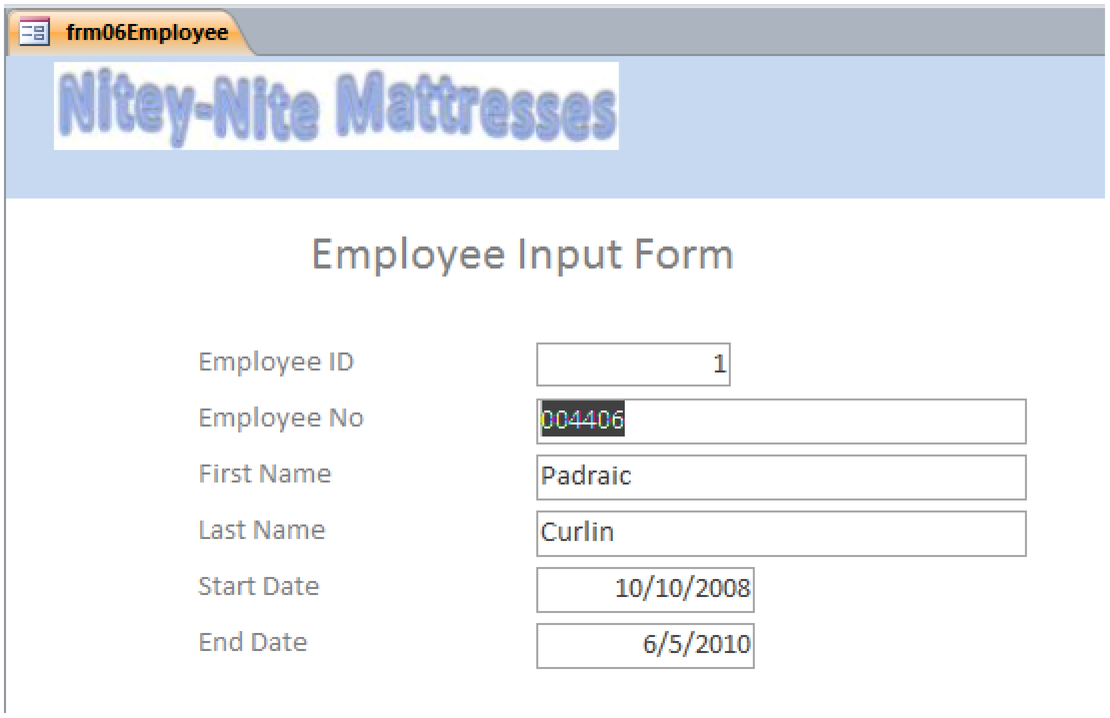
Employee Input Form

Employee ID	1
Employee No	004406
First Name	Padraic
Last Name	Curlin
Start Date	10/10/2008
End Date	6/5/2010

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**.


frm06Employee

Niteey-Nite Mattresses

Employee Input Form

Employee ID	1
Employee No	004406
First Name	Padraic
Last Name	Curlin
Start Date	10/10/2008
End Date	6/5/2010

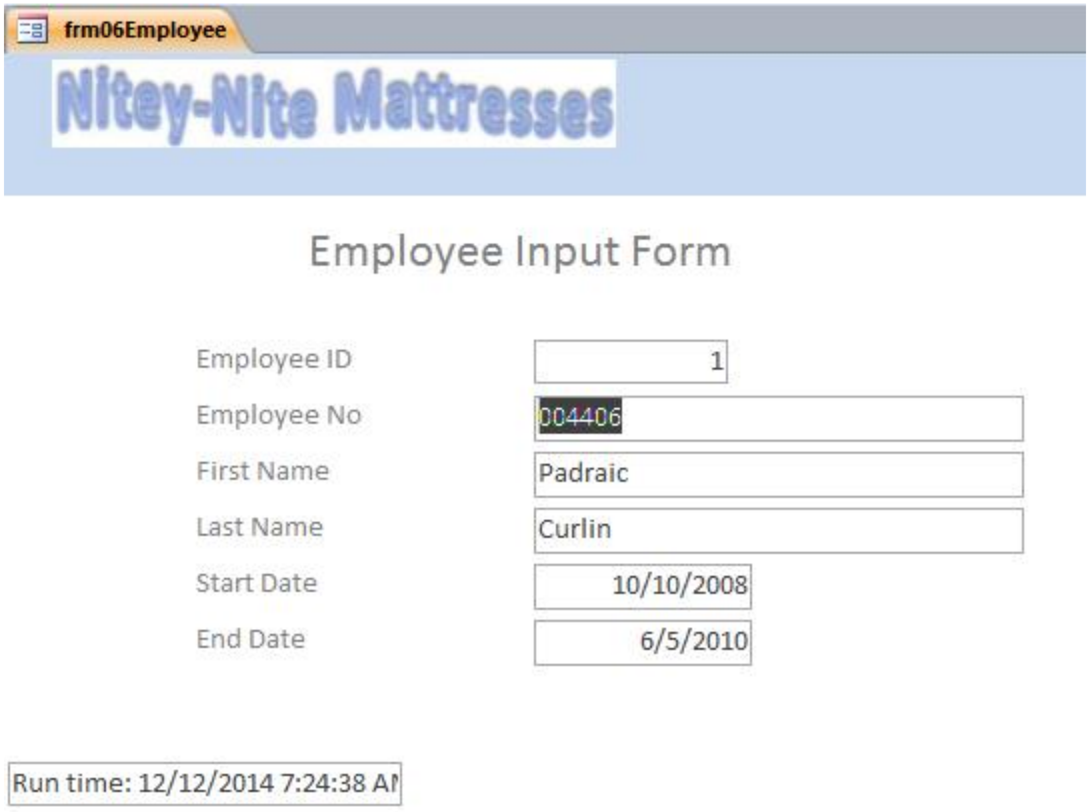
Figure 6.21

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 on 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 that expands 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 **Text16**. You don't need that label, so you can delete it.

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


frm06Employee

Niteey-Nite Mattresses

Employee Input Form

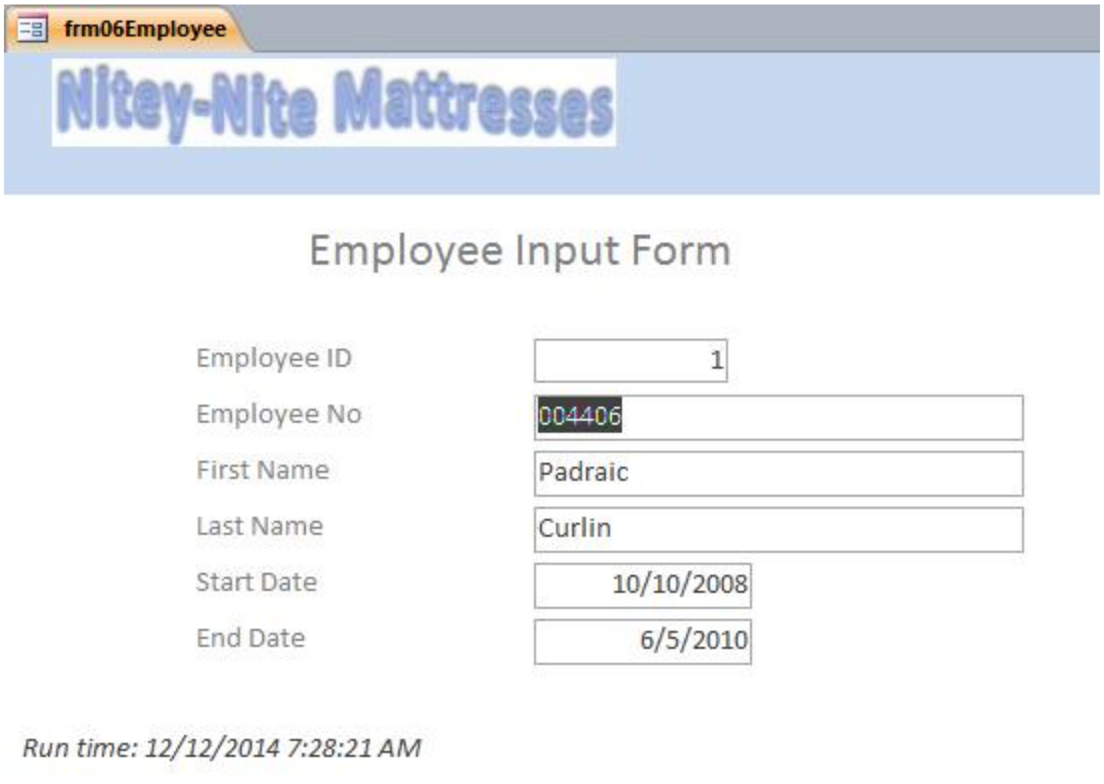
Employee ID	1
Employee No	004406
First Name	Padraic
Last Name	Curlin
Start Date	10/10/2008
End Date	6/5/2010

Run time: 12/12/2014 7:24:38 AI

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.

42. *Return to the* **Design View** *of the form.*
43. *With the* **text box** *in the* **Form Footer** *selected, click on the* **italics** *icon in the* **Text Formatting** *group of the* **Home** *tab.*
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. *View the form in* **Form View**.


frm06Employee

Niteey-Nite Mattresses

Employee Input Form

Employee ID	1
Employee No	004406
First Name	Padraic
Last Name	Curlin
Start Date	10/10/2008
End Date	6/5/2010

Run time: 12/12/2014 7:28:21 AM

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?*"

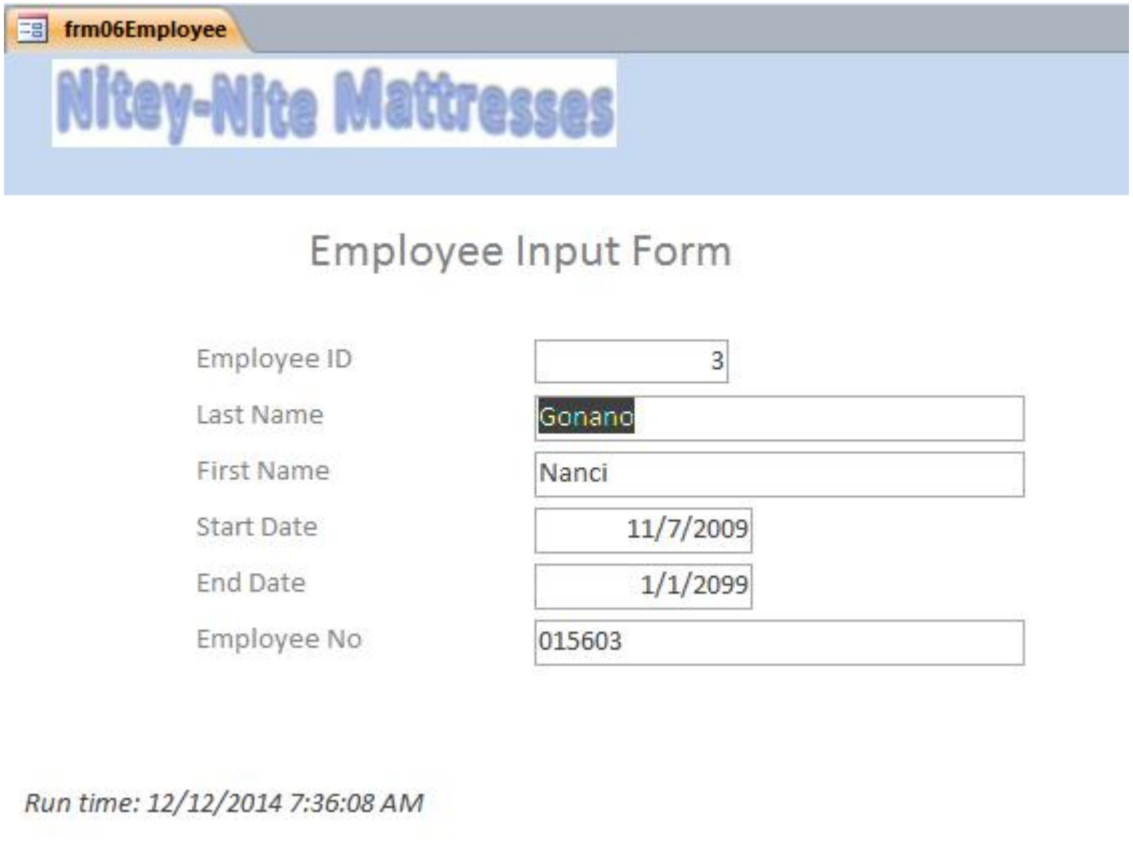
I really like to get users involved in the design phase of my projects because sometimes they can come up with things like this which will help them out in the long run, and many times the changes are easy to do. Don't ever discount input by users.

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

47. *Return to the* **Design View** *of the form.*
48. *Move the text boxes around in the order the clerk wanted.*
49. *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 a* **Tab Stop** *of* **No**.
50. *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.

51. *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***.*
52. *Change the* **Tab Index** *to* **1** *and make sure the* **Tab Stop** *is set to* **Yes***.*
53. *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.*
54. *Change the* **Tab Stop** *for the* **Run time** *text box to* **No***.*
55. *View the form in* **Form View** *and make sure everything is working properly.*
56. *Save and close the form.*



Employee ID	3
Last Name	Gonano
First Name	Nanci
Start Date	11/7/2009
End Date	1/1/2099
Employee No	015603

Run time: 12/12/2014 7:36:08 AM

Figure 6.24

> ***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 2010 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 toolbars. 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, log in and take the exam. Make sure that you take the exam on the same computer that you completed the sample files on, as some of the questions on the exam may refer to some of the completed examples.

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.

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. 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_2010** *database.*
2. *In the* **Create** *tab, click on the* **Form Design** *icon in the* **Forms** *group.*

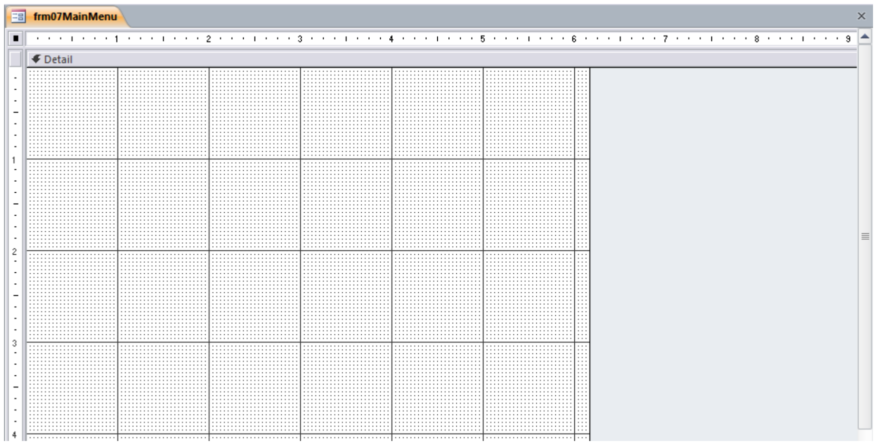


Figure 7.1

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

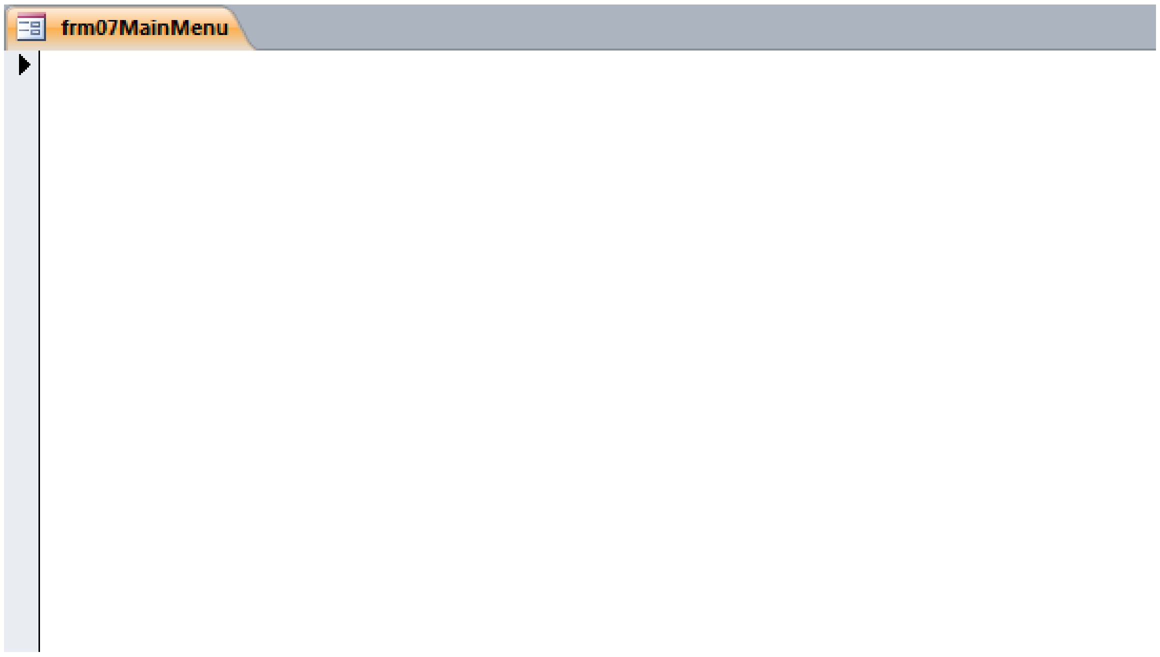


Figure 7.2

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 on the gray box at the intersection where the top and left rulers meet).*
7. *On the* **All** *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***.*

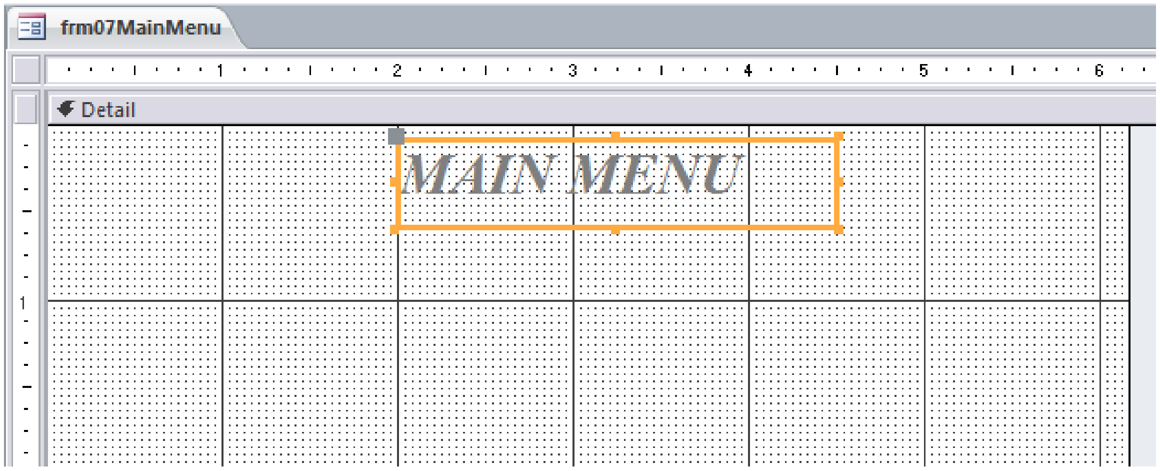
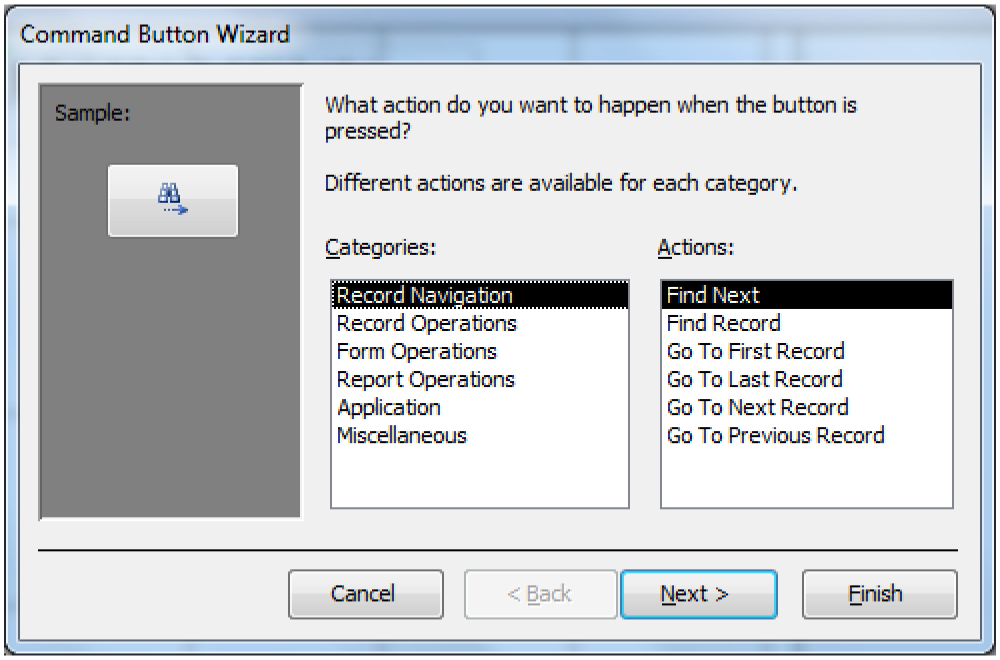


Figure 7.3

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** *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?

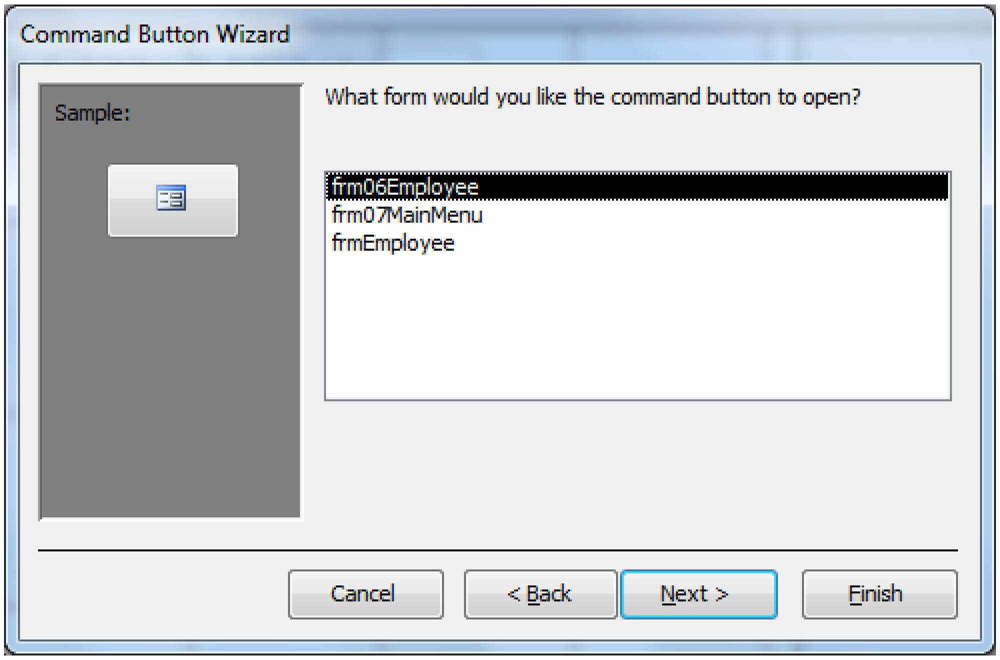
Different actions are available for each category.

Categories:	Actions:
Record Navigation	Find Next
Record Operations	Find Record
Form Operations	Go To First Record
Report Operations	Go To Last Record
Application	Go To Next Record
Miscellaneous	Go To Previous Record

Figure 7.5

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 >.**

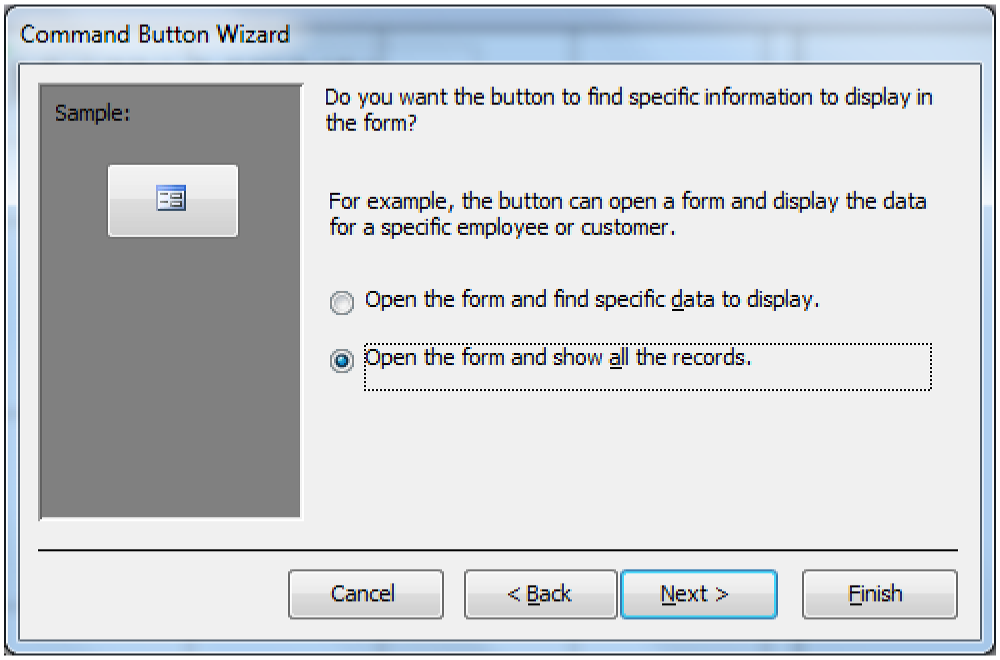


What form would you like the command button to open?

frm06Employee
frm07MainMenu
frmEmployee

Figure 7.6

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



Command Button Wizard

Sample:

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

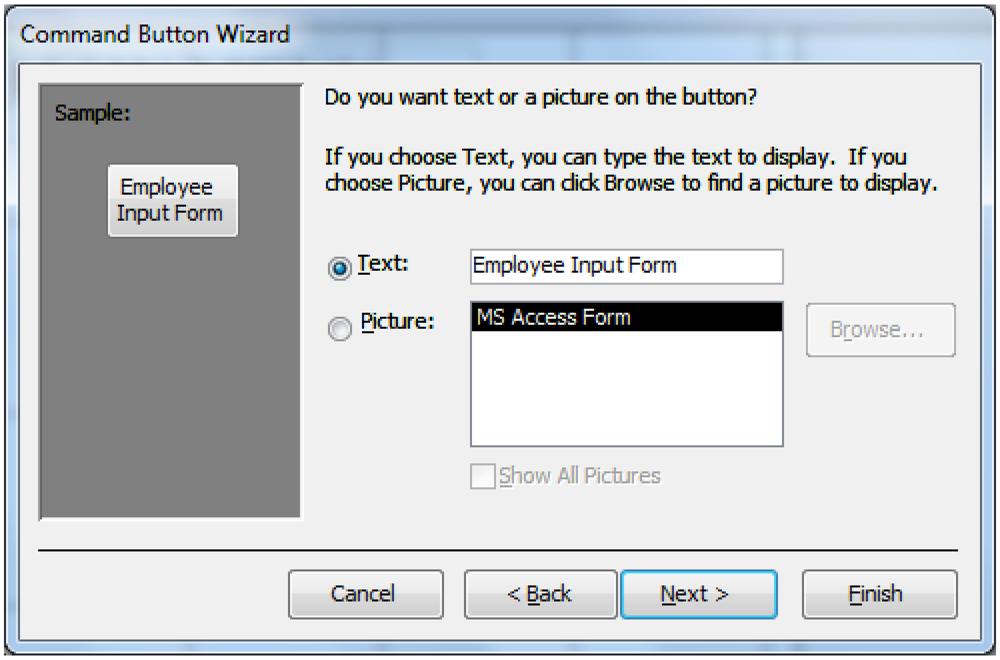
For example, the button can open a form and display the data for a specific employee or customer.

○ Open the form and find specific data to display.

● Open the form and show all the records.

Figure 7.7

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

Sample:

Employee Input Form

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

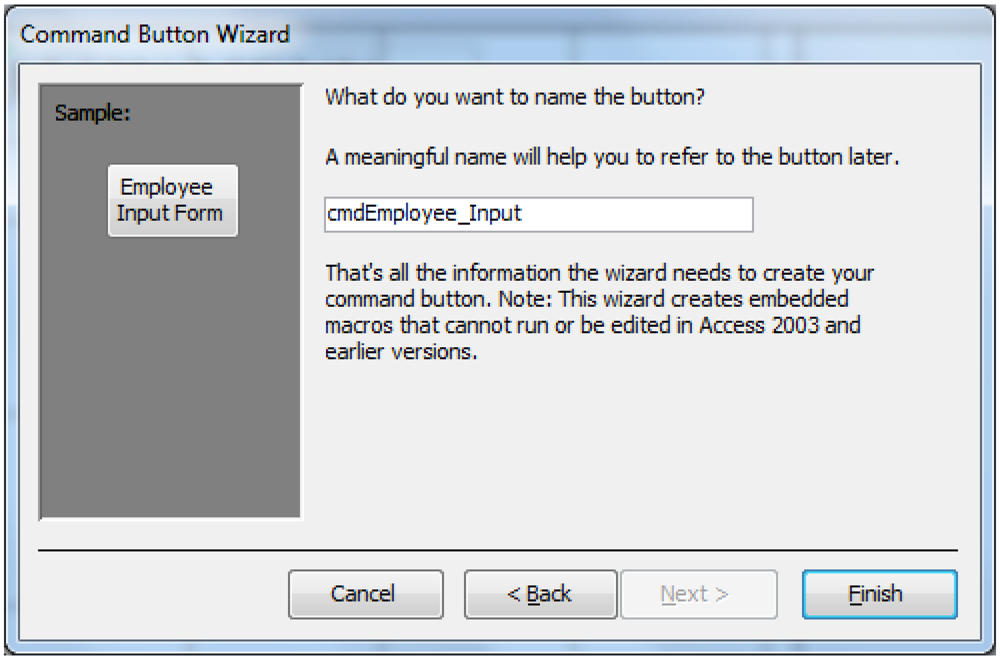
○ Picture: MS Access Form

☐ Show All Pictures

Figure 7.8

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 it was a cemetery I was creating the database for, we decided against that idea. I could just see the picture if a client family was in the room and someone decided to delete a record. We'll just use the text option in our example.

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



What do you want to name the button?

A meaningful name will help you to refer to the button later.

cmdEmployee_Input

That's all the information the wizard needs to create your command button. Note: This wizard creates embedded macros that cannot run or be edited in Access 2003 and earlier versions.

Figure 7.9

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**.

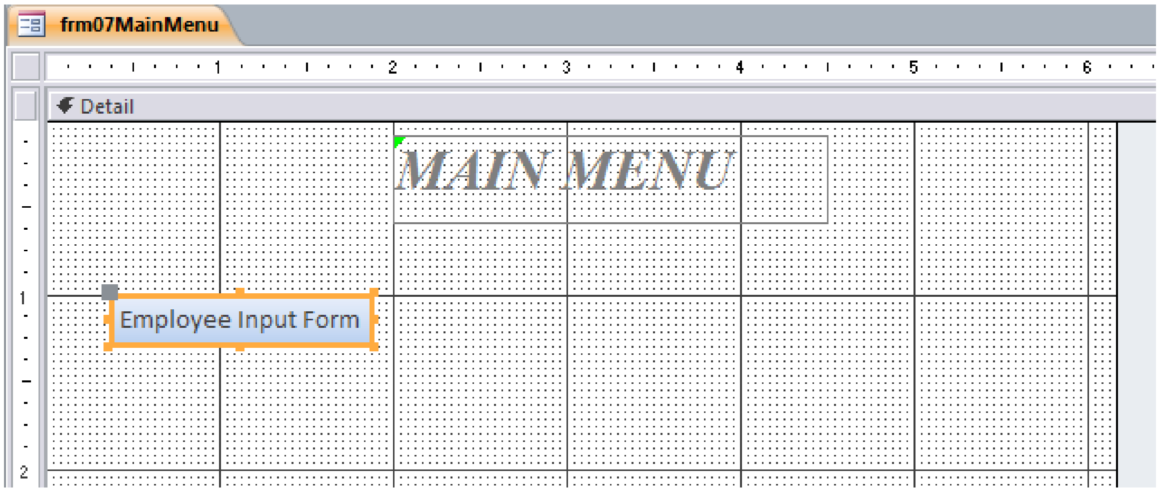


Figure 7.9

17. *Open the form in* **display** *mode and click the* **Employee Input Form command button**.

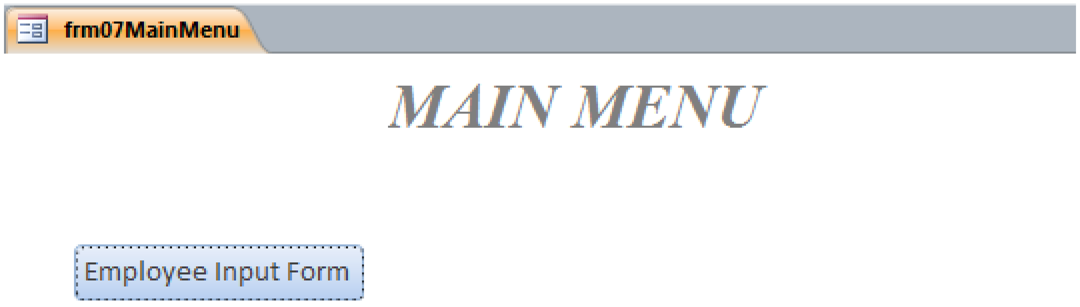
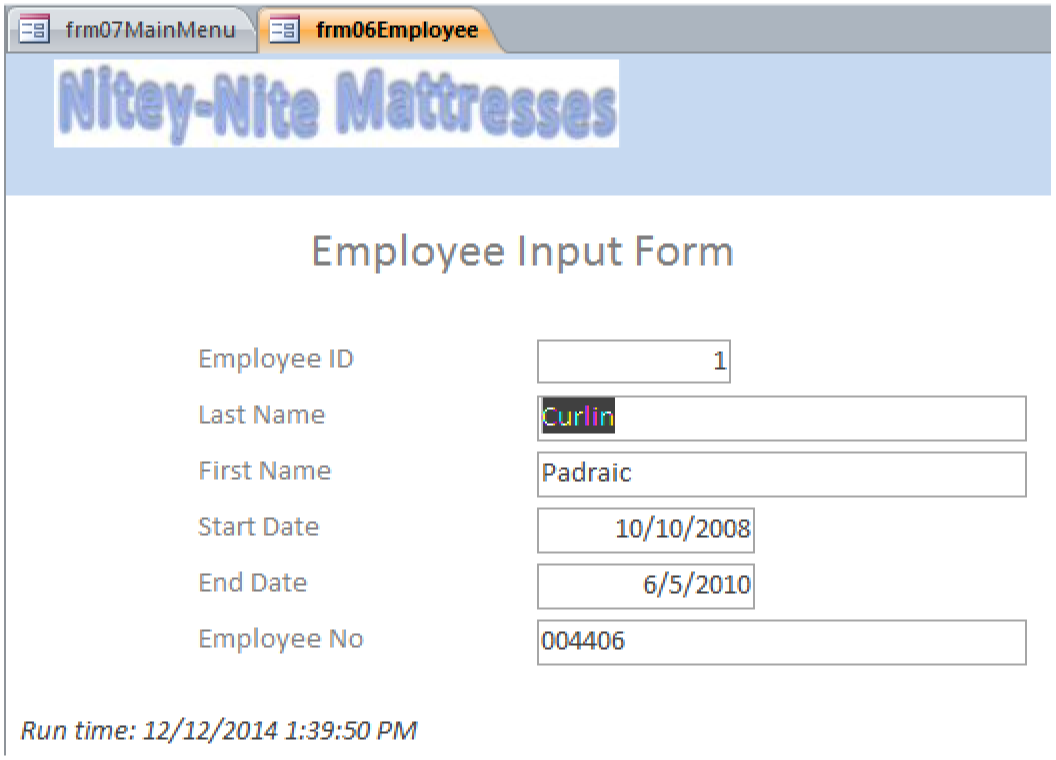


Figure 7.10



Employee Input Form

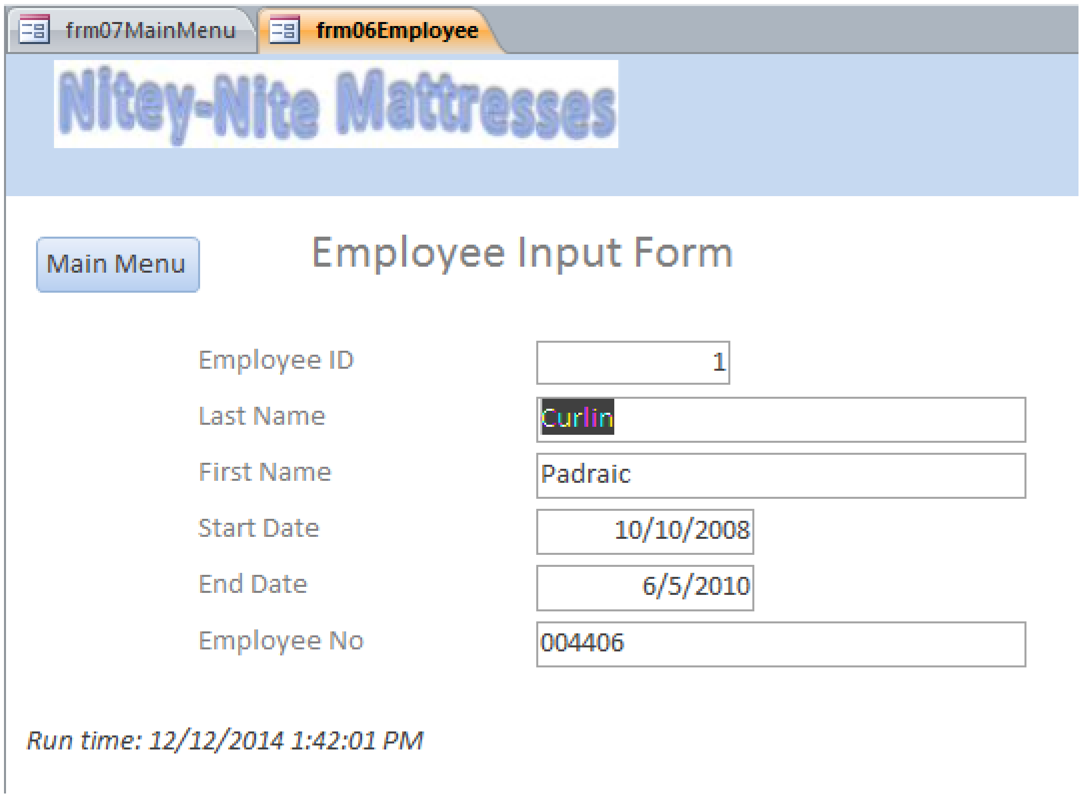
Employee ID	1
Last Name	Curlin
First Name	Padraic
Start Date	10/10/2008
End Date	6/5/2010
Employee No	004406

Run time: 12/12/2014 1:39:50 PM

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. *View the* **form** *in* **Form View**.



Employee Input Form

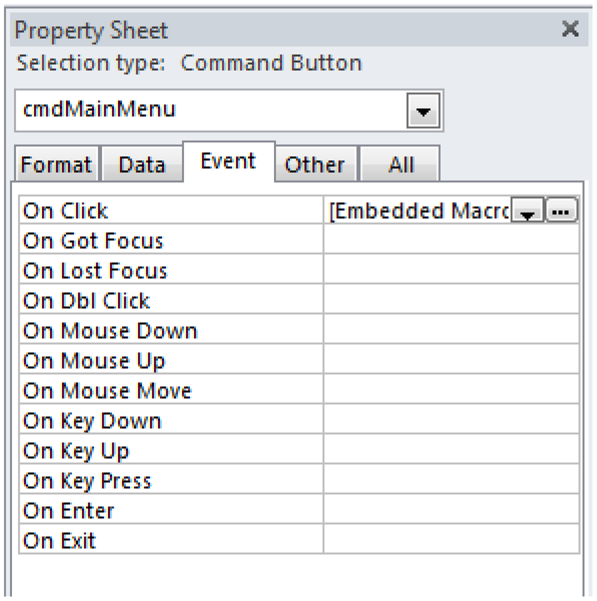
Employee ID	1
Last Name	Curlin
First Name	Padraic
Start Date	10/10/2008
End Date	6/5/2010
Employee No	004406

Run time: 12/12/2014 1:42:01 PM

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 in orange, 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**.



Property Sheet

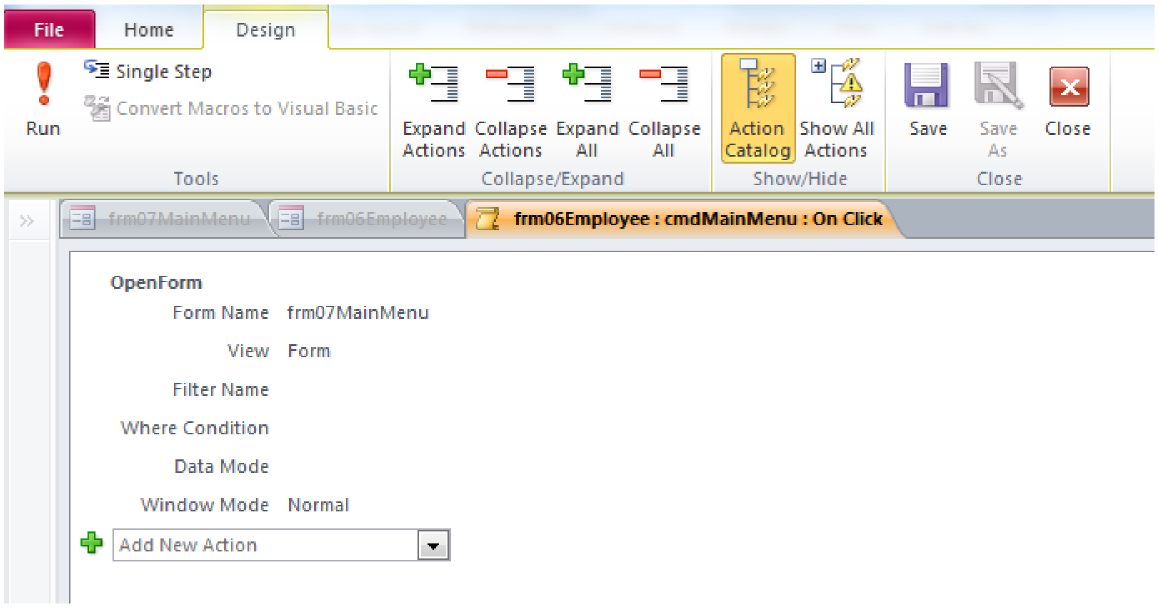
Selection type: Command Button

cmdMainMenu

On Click	[Embedded Macro
On Got Focus	
On Lost Focus	
On Dbl Click	
On Mouse Down	
On Mouse Up	
On Mouse Move	
On Key Down	
On Key Up	
On Key Press	
On Enter	
On Exit	

Figure 7.12

23. *Click on the* **ellipses (…)** *button (also called a* **Build** *button) to the right of the* **On Click** *property.*



OpenForm

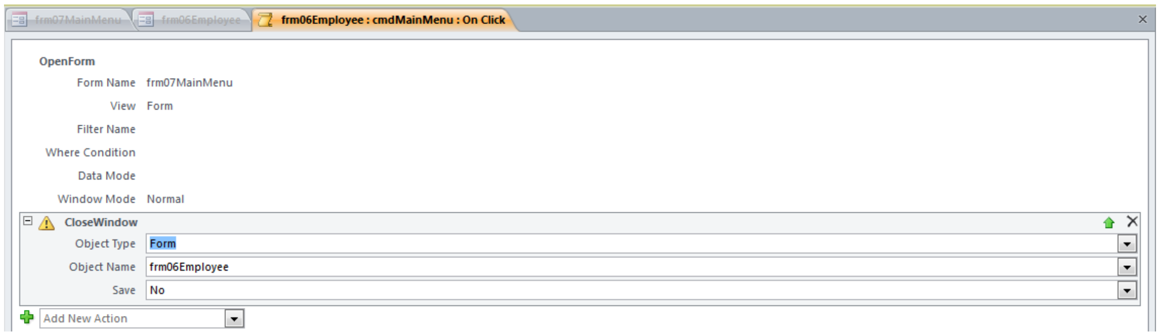
Form Name	frm07MainMenu
View	Form
Filter Name	
Where Condition	
Data Mode	
Window Mode	Normal

Add New Action

Figure 7.13

After you click on the build button, you should get a screen that looks like Figure 7.13. 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 7.14

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 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.*

The form frm06Employee opens up in Design View. Now you'll test the button to see if it works.

30. *Go to the* **Form View** *of the* **frm06Employee** *form and click on the* **Main Menu** *button.*
31. *When 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.

32. *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.

33. *Modify the* **On Click** *property of the* **Employee Input Form** *command button in* **frm07MainMenu** *to close the form before* **frm06Employee** *opens up.*
34. *Test the functionality of both command buttons to make sure they work as expected.*
35. *Save and close both forms.*

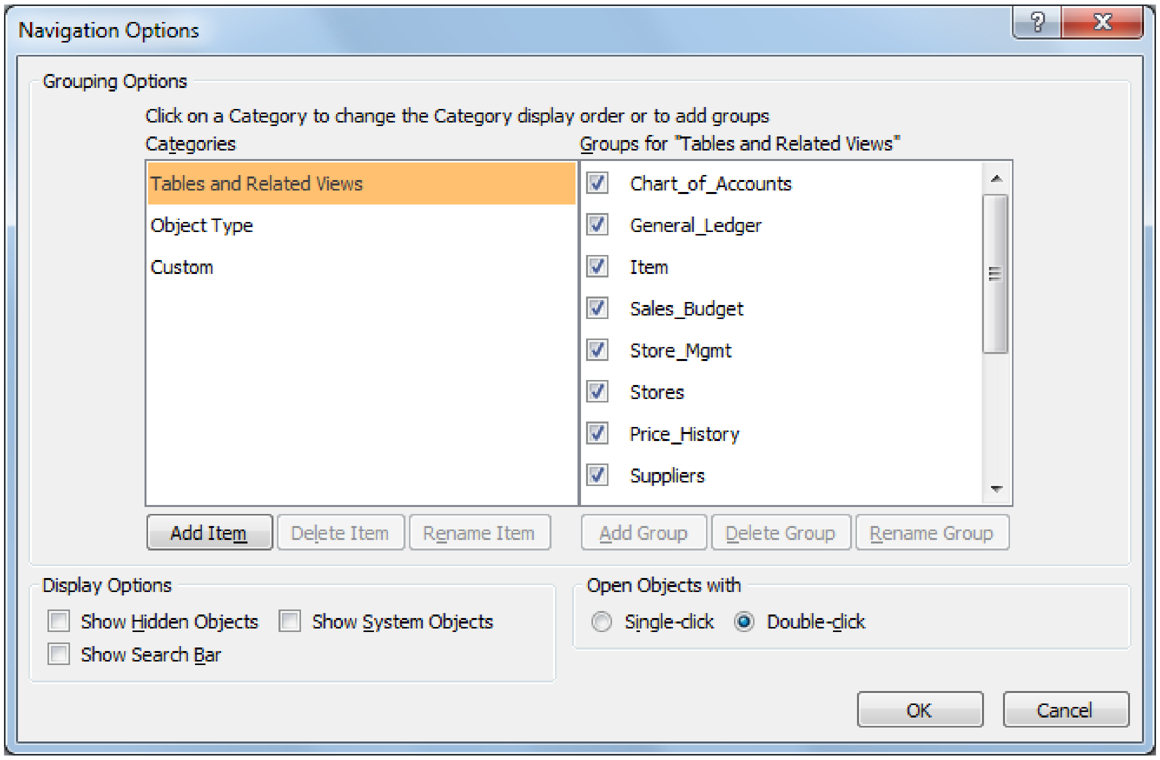
***Review Questions**: It is now time to complete the hands-on Review Questions. Log on to www.ExcelCEO.com with your Email and Password, click on the **Access 2010 Review Questions Chapter 7, Section 1 of 2** option and complete the review questions.*

Creating Custom Categories

In the past, Access has used the concept of a switchboard to navigate around a database. The switchboard is being phased out in Access 2010 and may be totally eliminated in 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. As Access switchboards are quickly becoming a thing of the past, this course will not explore them, but instead will 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
- Tables and Related Views
- Object Type
- Custom

Groups for "Tables and Related Views"
- ✓ Chart_of_Accounts
- ✓ General_Ledger
- ✓ Item
- ✓ Sales_Budget
- ✓ Store_Mgmt
- ✓ Stores
- ✓ Price_History
- ✓ Suppliers

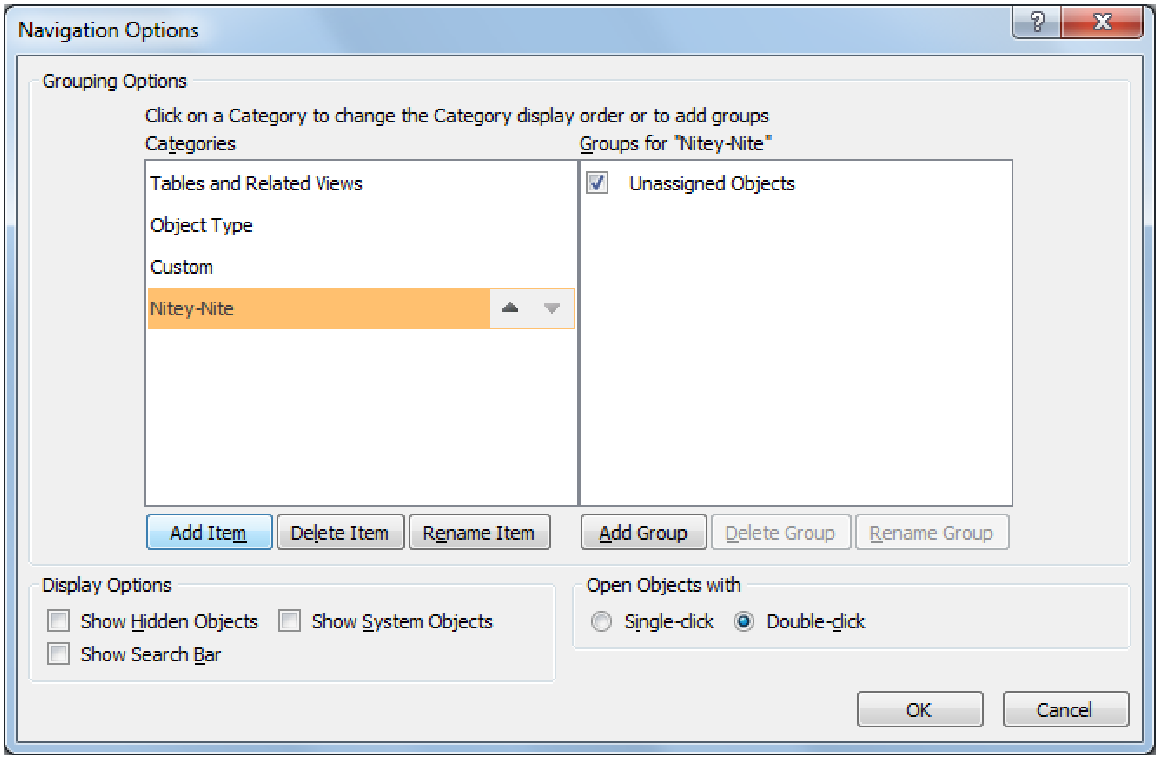
Display Options
- ☐ Show Hidden Objects
- ☐ Show System Objects
- ☐ Show Search Bar

Open Objects with
Single-click [✓ Double-click]

Figure 7.15

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* **Categories**, *click the* **Add Item** *button.*
3. *Replace* **Custom Category 1** *with* **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

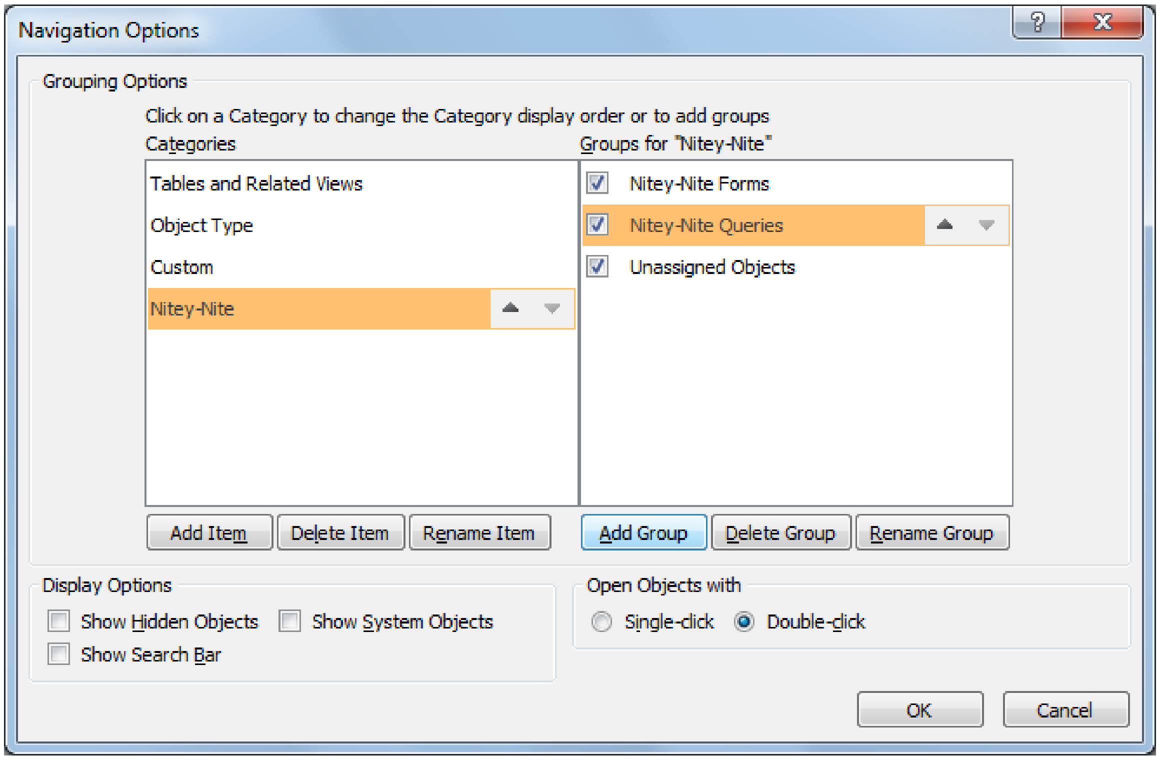
Open Objects with

○ Single-click ● Double-click

Figure 7.16

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

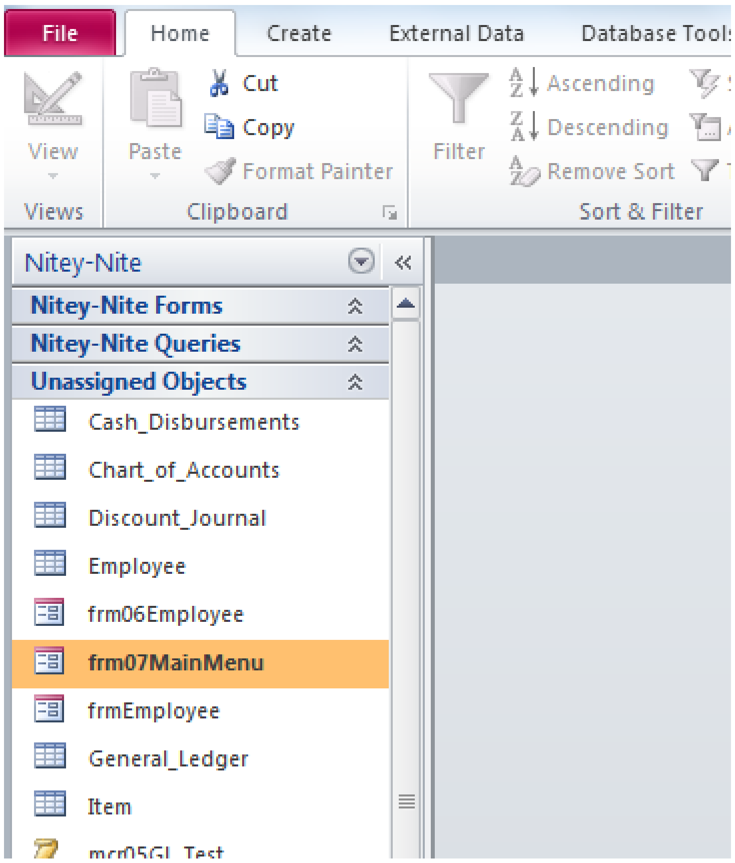
Single-click [✓ Double-click]

Figure 7.17

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**.



Nitey-Nite

Nitey-Nite Forms

Nitey-Nite Queries

Unassigned Objects

Cash_Disbursements

Chart_of_Accounts

Discount_Journal

Employee

frm06Employee

frm07MainMenu

frmEmployee

General_Ledger

Item

mcr05GL_Test

Figure 7.18

You see that 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.*

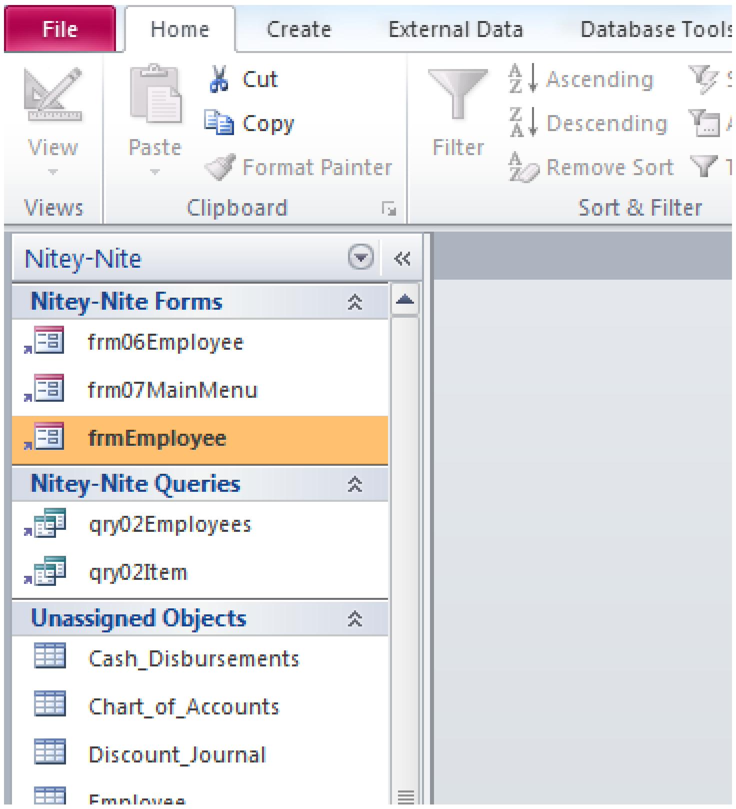


Figure 7.19

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.

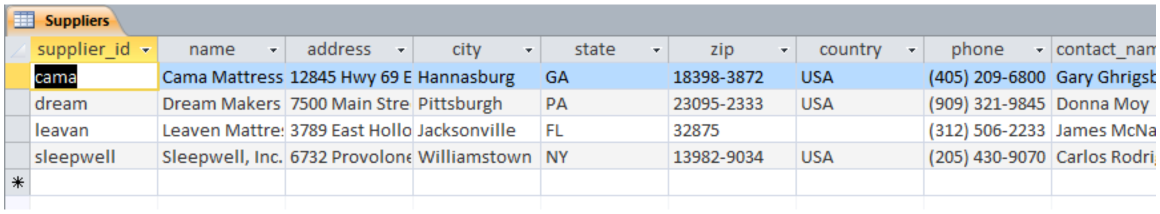
12. *Right-click on the* **Nitey-Nite** *category, point to* **Category,** *and click on* **Object Type** *to have the* **Navigation Pane** *return to* **All Access Objects**.

SubForms

Sometimes it is necessary to view data from multiple tables in a 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.*

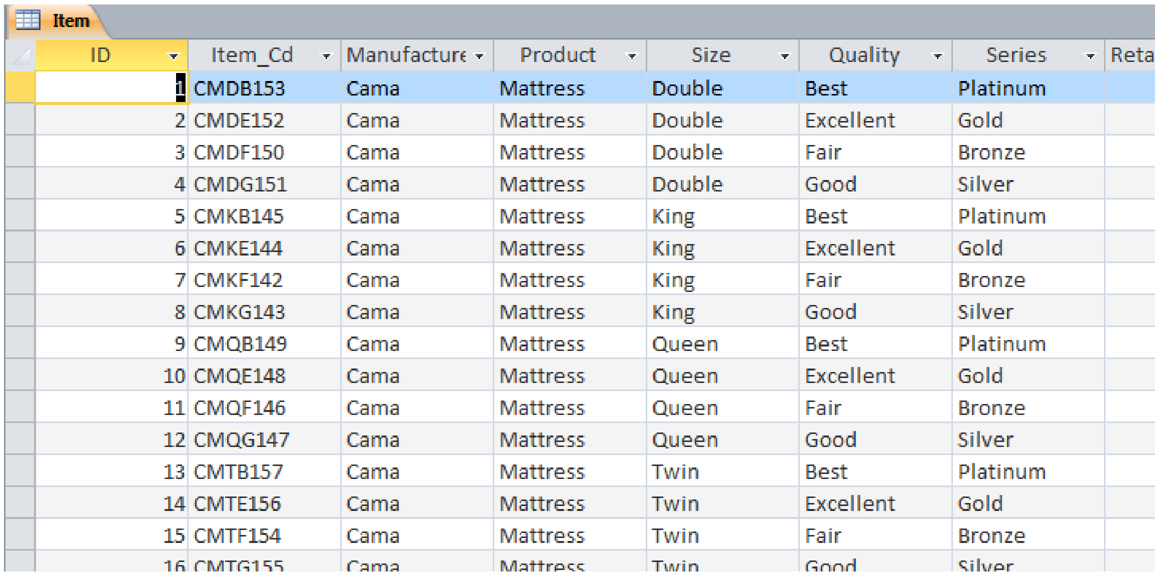

Suppliers

supplier_id	name	address	city	state	zip	country	phone	contact_nam
cama	Cama Mattress	12845 Hwy 69 E	Hannasburg	GA	18398-3872	USA	(405) 209-6800	Gary Ghrigsb
dream	Dream Makers	7500 Main Stre	Pittsburgh	PA	23095-2333	USA	(909) 321-9845	Donna Moy
leavan	Leaven Mattre:	3789 East Hollo	Jacksonville	FL	32875		(312) 506-2233	James McNa
sleepwell	Sleepwell, Inc.	6732 Provolone	Williamstown	NY	13982-9034	USA	(205) 430-9070	Carlos Rodri

Figure 7.20

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	Reta
1	CMDB153	Cama	Mattress	Double	Best	Platinum	
2	CMDE152	Cama	Mattress	Double	Excellent	Gold	
3	CMDF150	Cama	Mattress	Double	Fair	Bronze	
4	CMDG151	Cama	Mattress	Double	Good	Silver	
5	CMKB145	Cama	Mattress	King	Best	Platinum	
6	CMKE144	Cama	Mattress	King	Excellent	Gold	
7	CMKF142	Cama	Mattress	King	Fair	Bronze	
8	CMKG143	Cama	Mattress	King	Good	Silver	
9	CMQB149	Cama	Mattress	Queen	Best	Platinum	
10	CMQE148	Cama	Mattress	Queen	Excellent	Gold	
11	CMQF146	Cama	Mattress	Queen	Fair	Bronze	
12	CMQG147	Cama	Mattress	Queen	Good	Silver	
13	CMTB157	Cama	Mattress	Twin	Best	Platinum	
14	CMTE156	Cama	Mattress	Twin	Excellent	Gold	
15	CMTF154	Cama	Mattress	Twin	Fair	Bronze	
16	CMTG155	Cama	Mattress	Twin	Good	Silver	

Figure 7.21

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 table 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.*

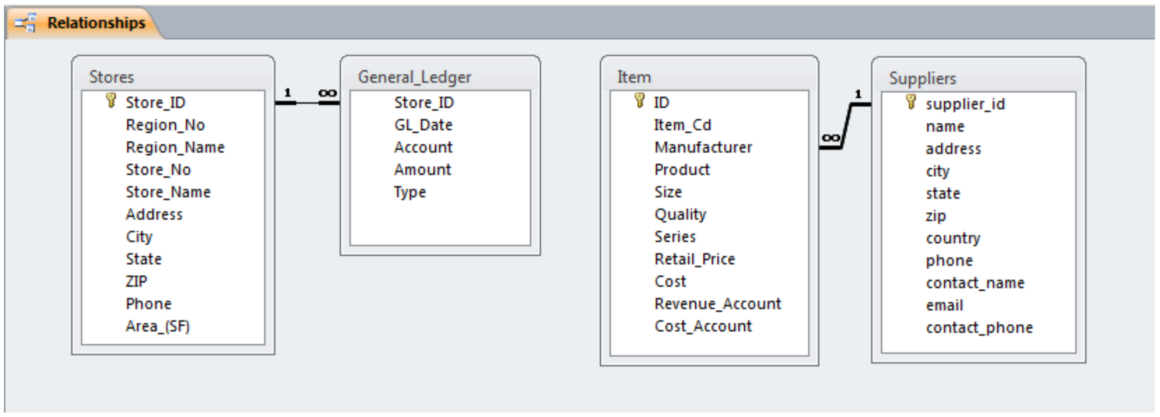


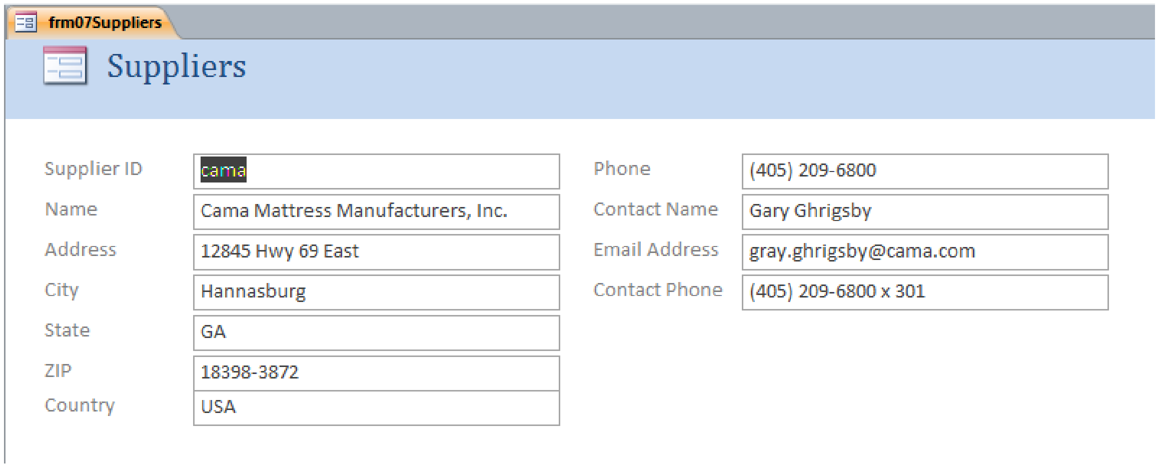
Figure 7.22

6. *Close the* **Relationships** *view. Choose* **Yes** *when asked if you want to save it.*

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 7.23**.
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 reposition the text boxes and labels within the form.*


frm07Suppliers

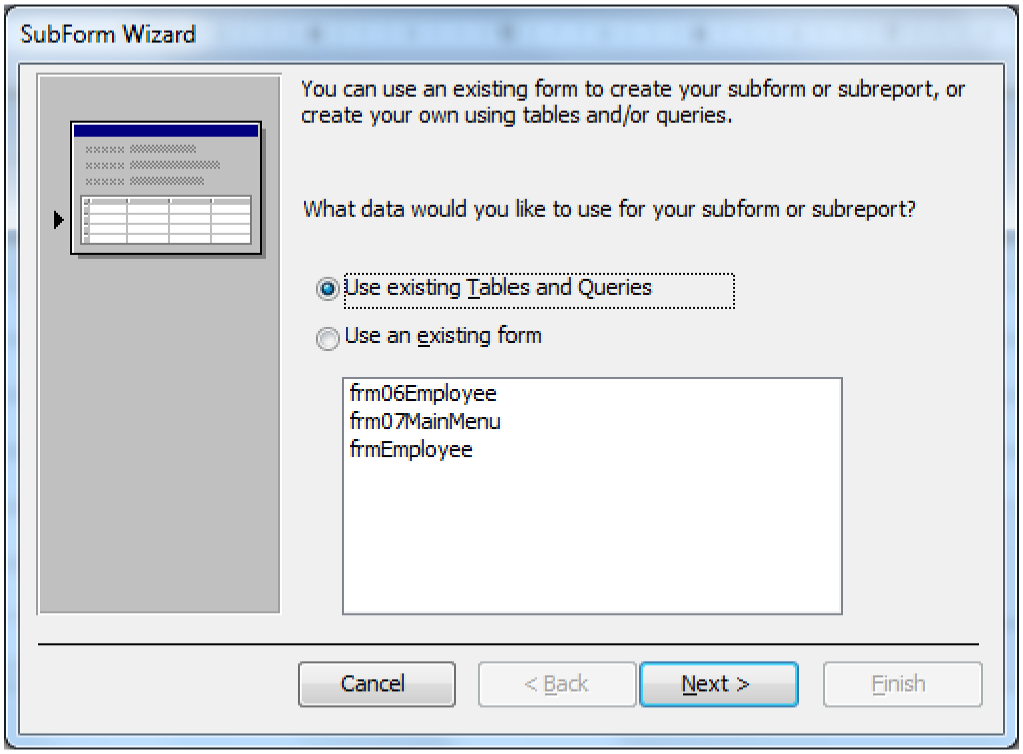
Suppliers

Supplier ID	cama	Phone	(405) 209-6800
Name	Cama Mattress Manufacturers, Inc.	Contact Name	Gary Ghrigsby
Address	12845 Hwy 69 East	Email Address	gray.ghrigsby@cama.com
City	Hannasburg	Contact Phone	(405) 209-6800 x 301
State	GA		
ZIP	18398-3872		
Country	USA		

Figure 7.23

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. *Go to the* **Design View** *of the form and make sure you have about* **3"** *of space below the last text box to work with.*
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 7.24

13. *Make sure the* **Use existing Tables and Queries** *radio button is selected and click* **Next >**.

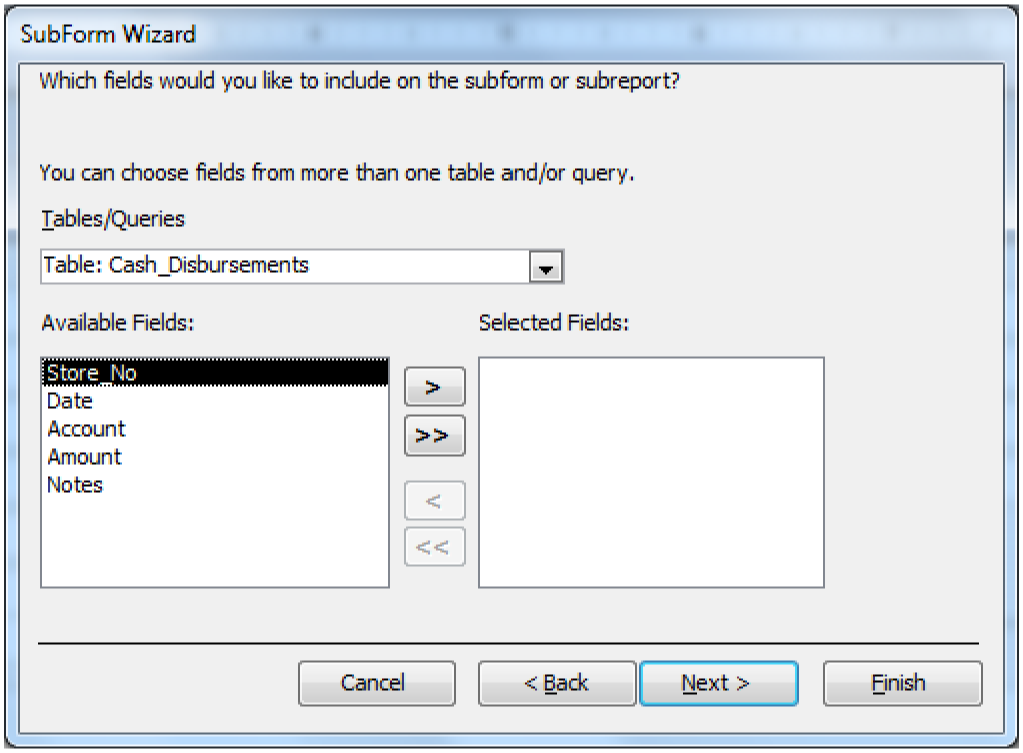
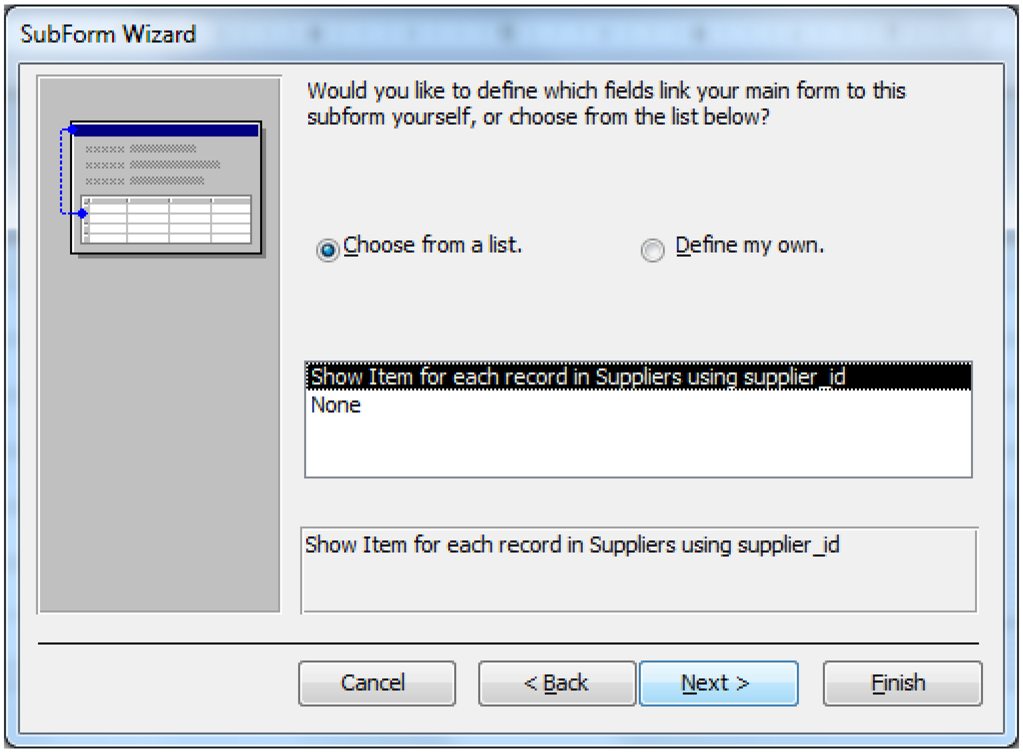

SubForm Wizard
Which fields would you like to include on the subform or subreport?
You can choose fields from more than one table and/or query.
Tables/Queries
Table: Cash_Disbursements
Available Fields:
Selected Fields:
Store_No
Date
Account
Amount
Notes
>
>>
<
<<
Cancel
< Back
Next >
Finish

Figure 7.25

14. In the **Tables/Queries** *drop-down menu, choose* **Table: Item.**

15. Move all of the fields over to the **Selected Fields** *area, except for* **Revenue_Account** *and* **Cost_Account** *and click* **Next >**.


SubForm Wizard

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.

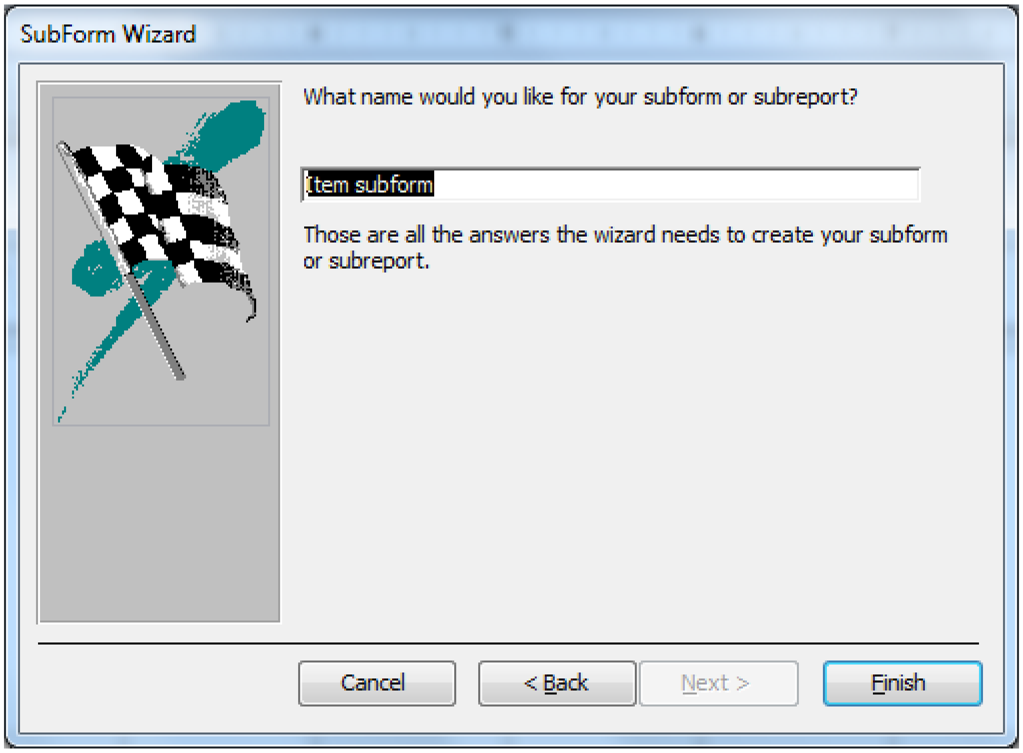
Show Item for each record in Suppliers using supplier_id
None

Show Item for each record in Suppliers using supplier_id

Figure 7.26

The next screen is a really cool thing about Access 2010. 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 >**.


SubForm Wizard

What name would you like for your subform or subreport?

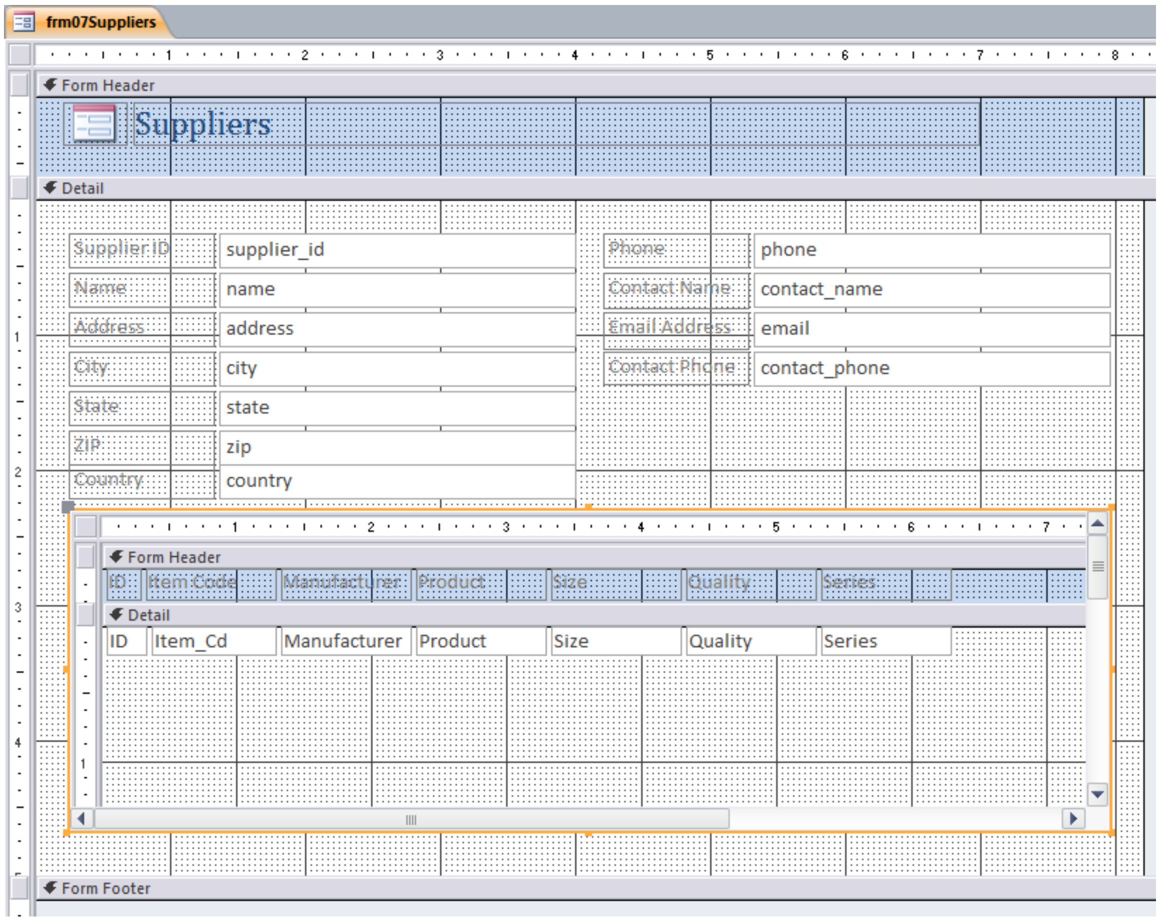
Item subform

Those are all the answers the wizard needs to create your subform or subreport.

Figure 7.27

17. Keep the default subform name and click **Finish**.

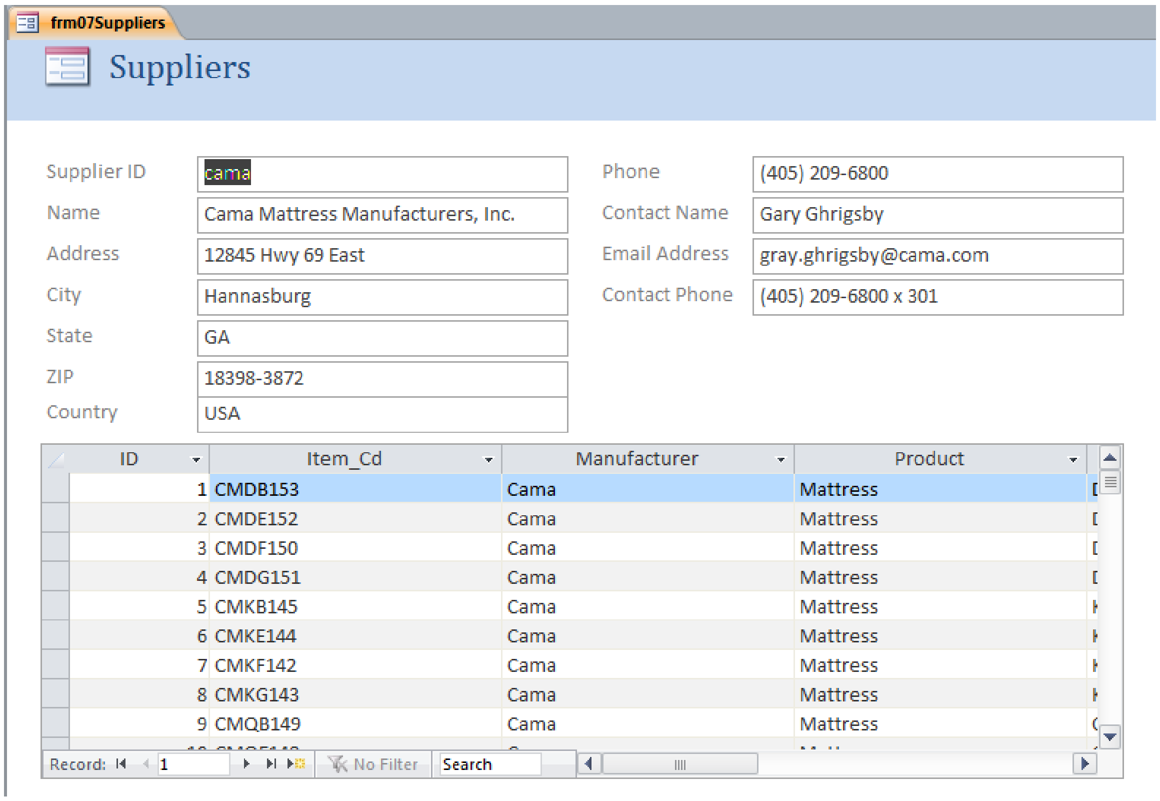
18. Delete the label **Item subform** *and rearrange the subform to make it look like* **Figures 7.28 and 7.29**.



Supplier ID	supplier_id	Phone	phone
Name	name	Contact Name	contact_name
Address	address	Email Address	email
City	city	Contact Phone	contact_phone
State	state		
ZIP	zip		
Country	country		

ID	Item Code	Manufacturer	Product	Size	Quality	Series
ID	Item_Cd	Manufacturer	Product	Size	Quality	Series

Figure 7.28


frm07Suppliers

Suppliers

Supplier ID	cama	Phone	(405) 209-6800
Name	Cama Mattress Manufacturers, Inc.	Contact Name	Gary Ghrigsby
Address	12845 Hwy 69 East	Email Address	gray.ghrigsby@cama.com
City	Hannasburg	Contact Phone	(405) 209-6800 x 301
State	GA		
ZIP	18398-3872		
Country	USA		

ID	Item_Cd	Manufacturer	Product
1	CMDB153	Cama	Mattress
2	CMDE152	Cama	Mattress
3	CMDF150	Cama	Mattress
4	CMDG151	Cama	Mattress
5	CMKB145	Cama	Mattress
6	CMKE144	Cama	Mattress
7	CMKF142	Cama	Mattress
8	CMKG143	Cama	Mattress
9	CMQB149	Cama	Mattress

Figure 7.29

It may take a little work to get it to look just right, but such a form can be very powerful. 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.

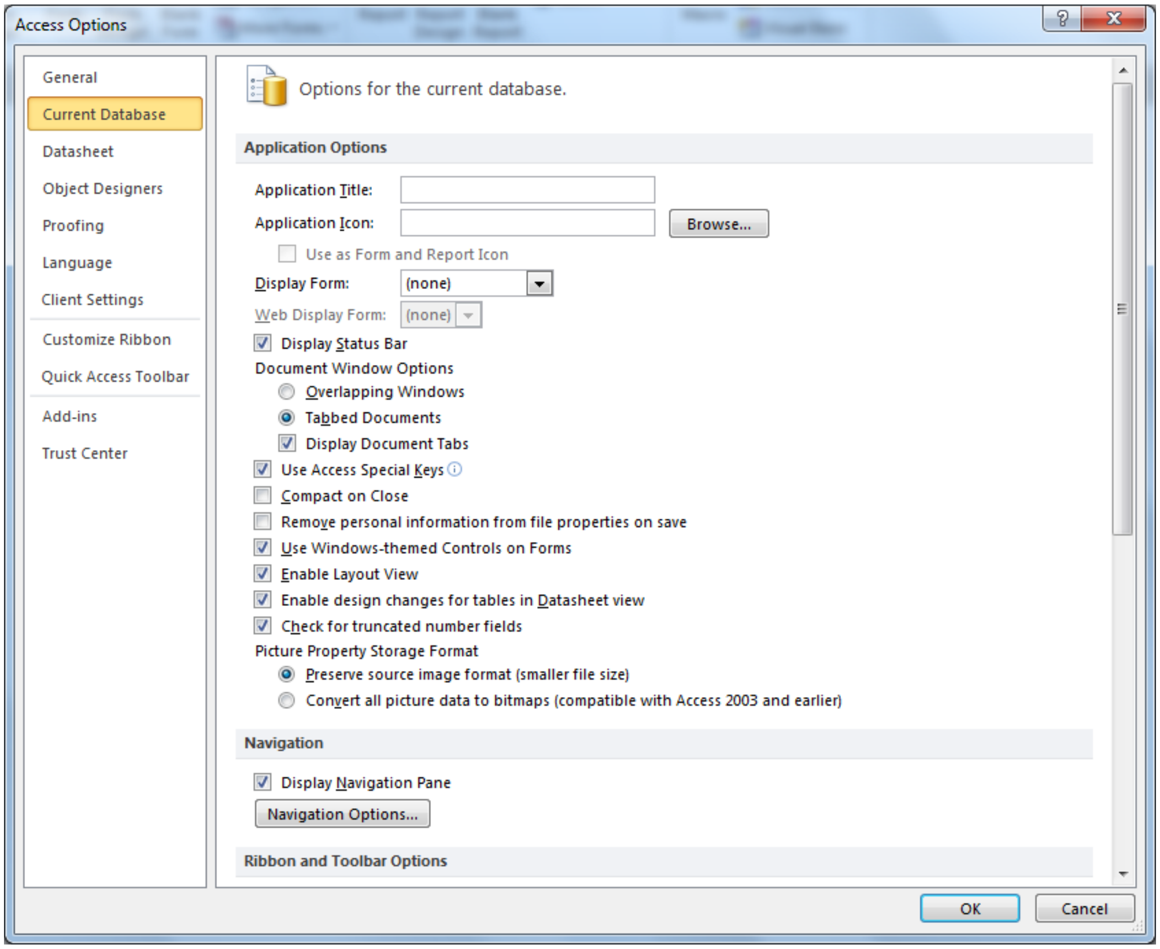
19. *Save and close* **frm07Suppliers**.

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 objects you may have open.*
2. *Click on the* **File** *button, 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

☑ Check for truncated number fields

Picture Property Storage Format

● Preserve source image format (smaller file size)

○ Convert all picture data to bitmaps (compatible with Access 2003 and earlier)

Navigation

☑ Display Navigation Pane

Ribbon and Toolbar Options

Figure 7.30

4. *In the* **Display Form:** *drop-down menu, choose* **frm07MainMenu**.
5. *Make sure the* **Display Navigation Pane** *check box is cleared.*
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_2010** *database.*

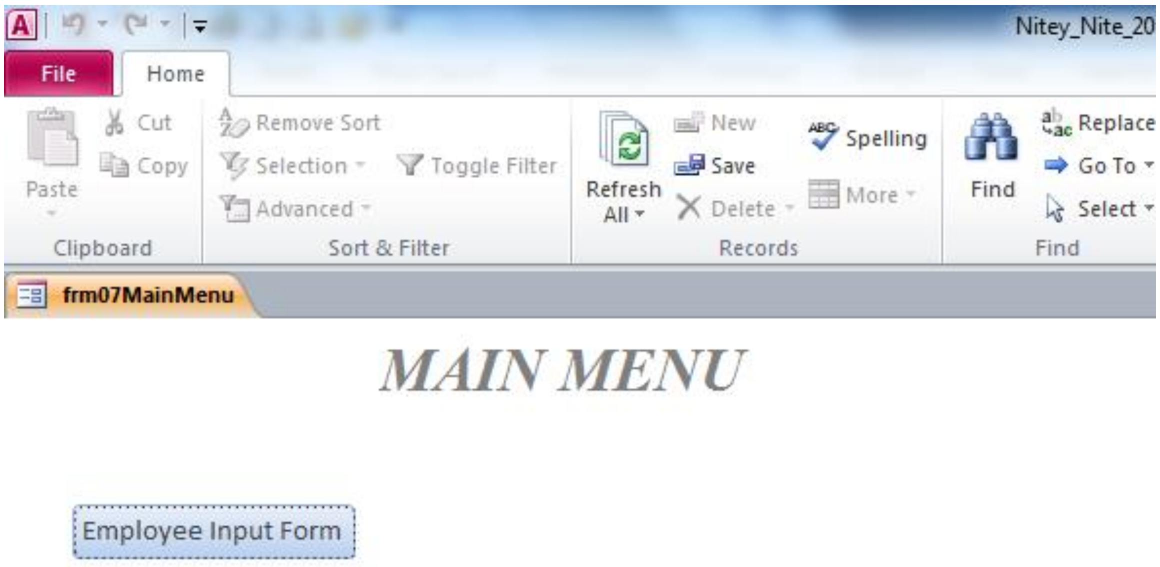


Figure 7.31

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_2010** *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 made.

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 2010 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, log in and take the exam. Make sure that you take the exam on the same computer that you completed the sample files on, as some of the questions on the exam may refer to some of the completed examples.

ExcelCEO

Chief Excel Officer

Access 2010 and SQL

Complete Self-study Course

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.

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.

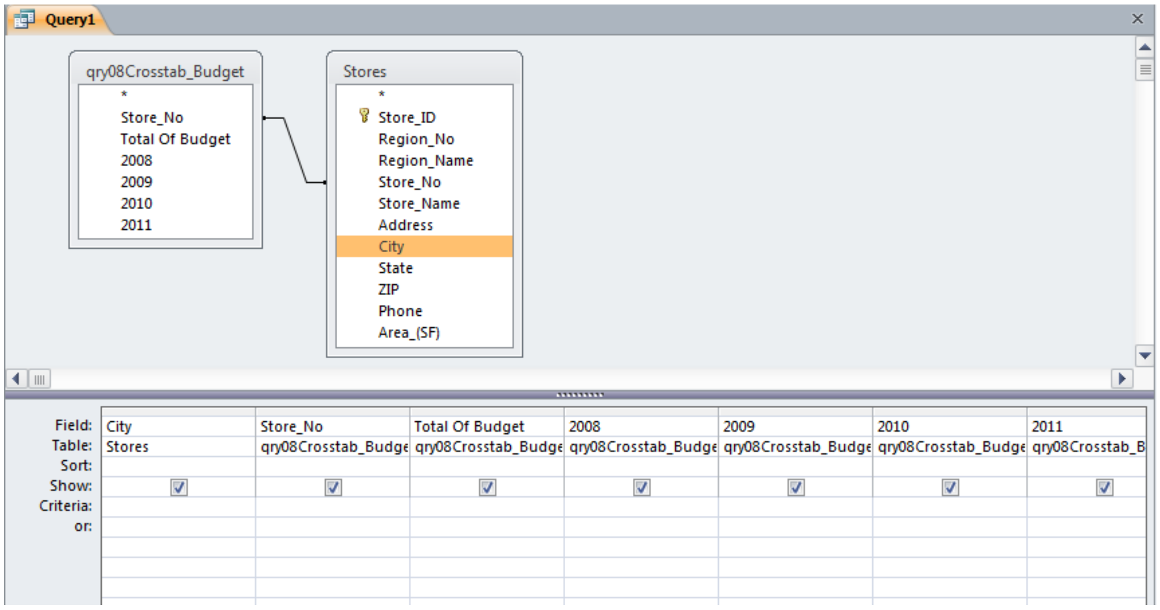
In this chapter, you will develop a very simple report, so you can learn the basics of creating reports. In subsequent chapters, we'll get into the cool stuff. This chapter will concentrate on reporting basics.

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 it is 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 2008, 2009, 2010, and 2011 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.

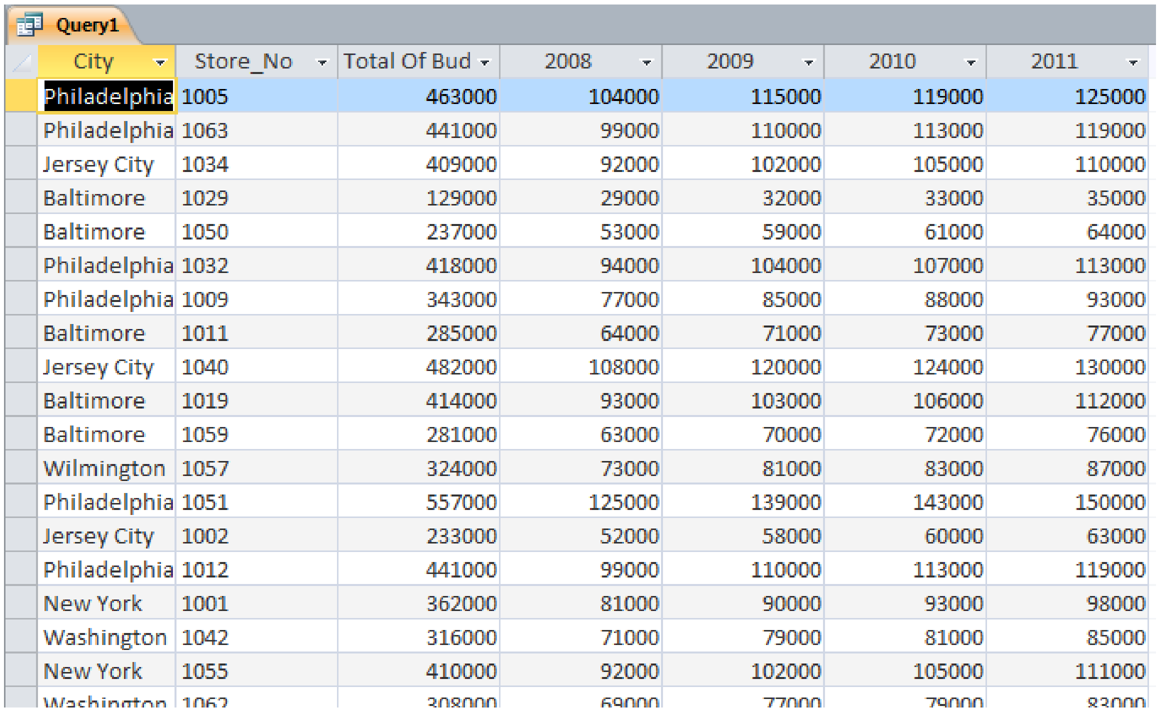
1. *Open the* **Nitey_Nite_2010** *database and close the* **Main Menu** *form.*
2. *Copy* **qry05Crosstab_Budget** *and name the new query* **qry08Crosstab_Budget**.
3. *Create a new query that joins* **qry08Crosstab_Budget** *with the* **Stores** *table on* **Store_No.**
4. *Bring the* **City** *field from the* **Stores** *table, and all fields from* **qry08Crosstab_Budget**.



Field:	City	Store_No	Total Of Budget	2008	2009	2010	2011
Table:	Stores	qry08Crosstab_Budge	qry08Crosstab_Budge	qry08Crosstab_Budge	qry08Crosstab_Budge	qry08Crosstab_Budge	qry08Crosstab_B
Sort:							
Show:	✓	✓	✓	✓	✓	✓	✓
Criteria:							
or:							

Figure 8.1

5. *Run the query.*



City	Store_No	Total Of Bud	2008	2009	2010	2011
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
Washington	1062	308000	69000	77000	79000	83000

Figure 8.2

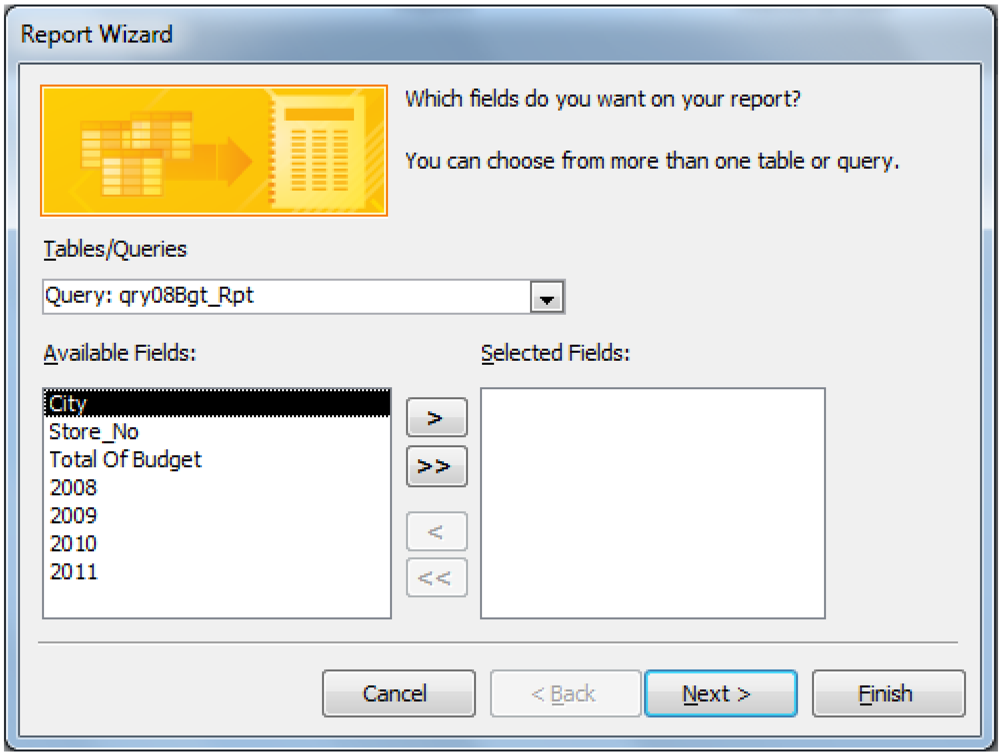
It has all of the correct information in it, but it's not sorted or formatted, and it just doesn't look pretty. Using this query as our data source, we will create a report.

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 little too much of a 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

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, 2008, 2009, 2010, 2011

Selected Fields:

Figure 8.3

The first screen of the reporting 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** >.

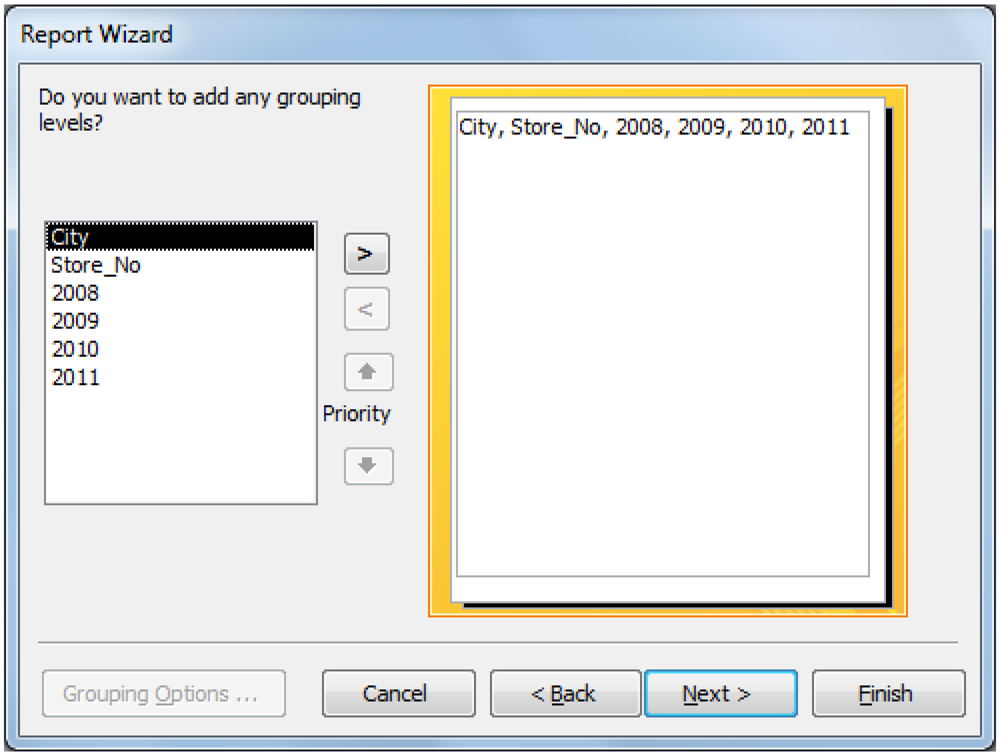
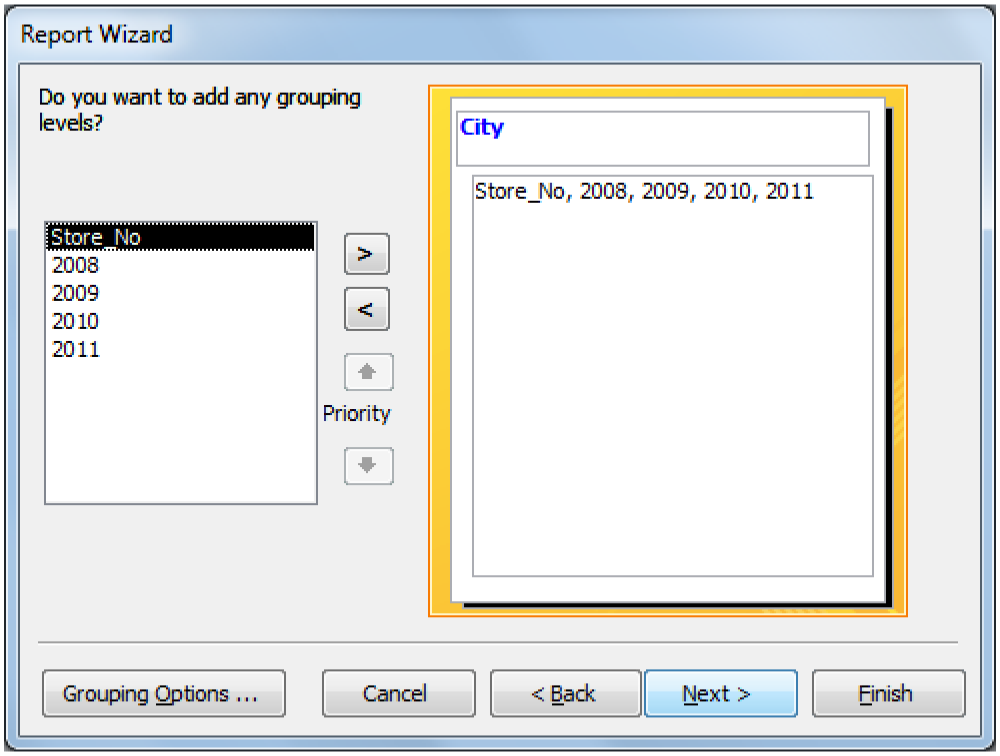


Figure 8.4

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.*



Do you want to add any grouping levels?

Store_No
2008
2009
2010
2011

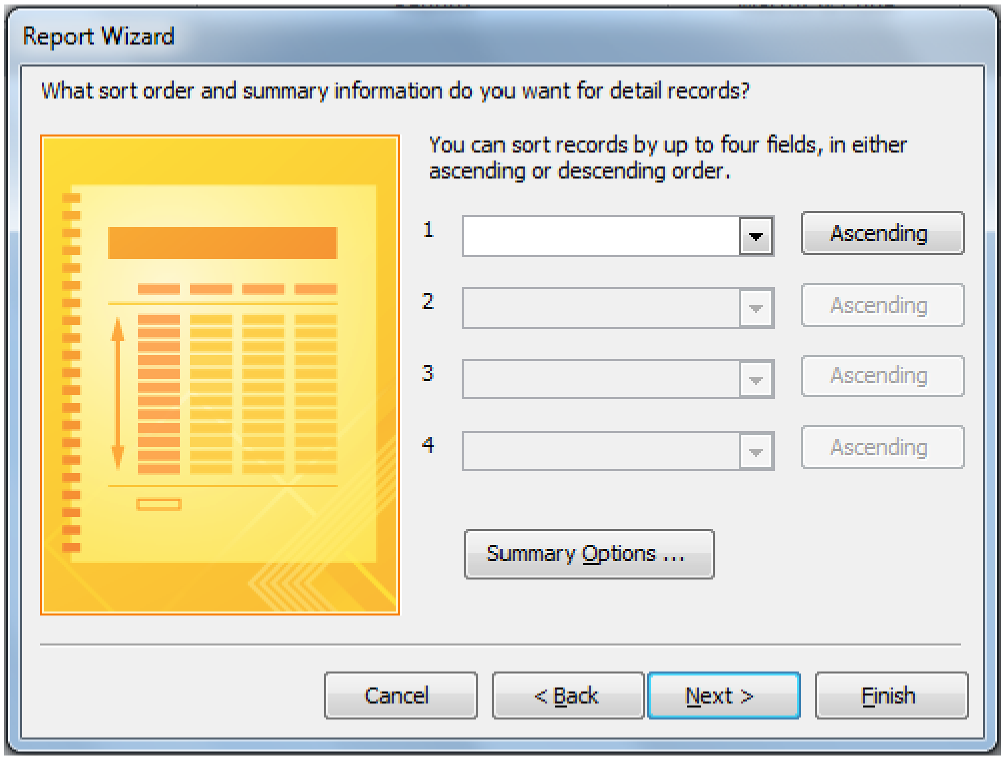
City

Store_No, 2008, 2009, 2010, 2011

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.

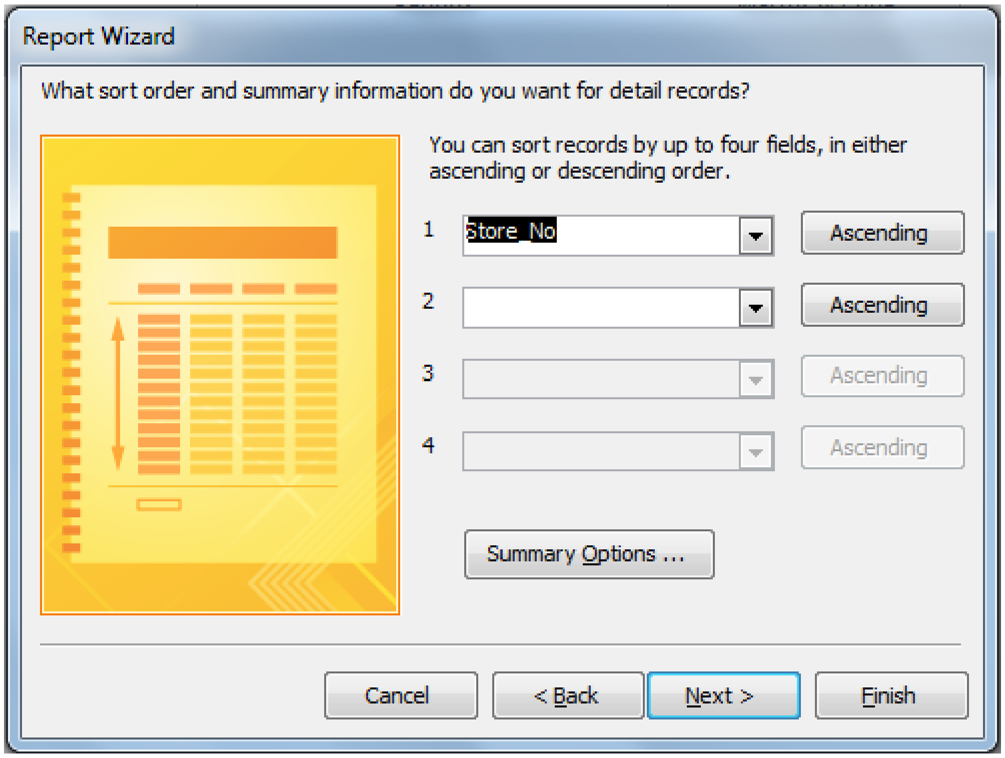


What sort order and summary information do you want for detail records?

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**.



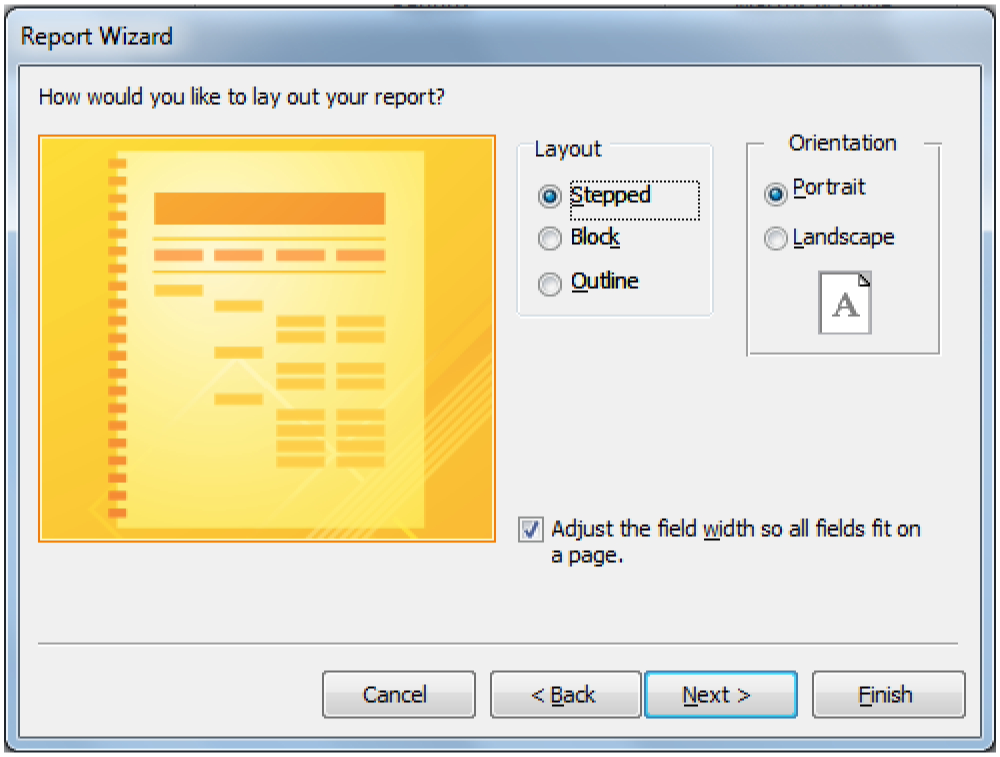
What sort order and summary information do you want for detail records?

You can sort records by up to four fields, in either ascending or descending order.

1 Store_No

Figure 8.7

14. Click **Next** >.



How would you like to lay out your report?

Layout: [✓ Stepped] Block Outline

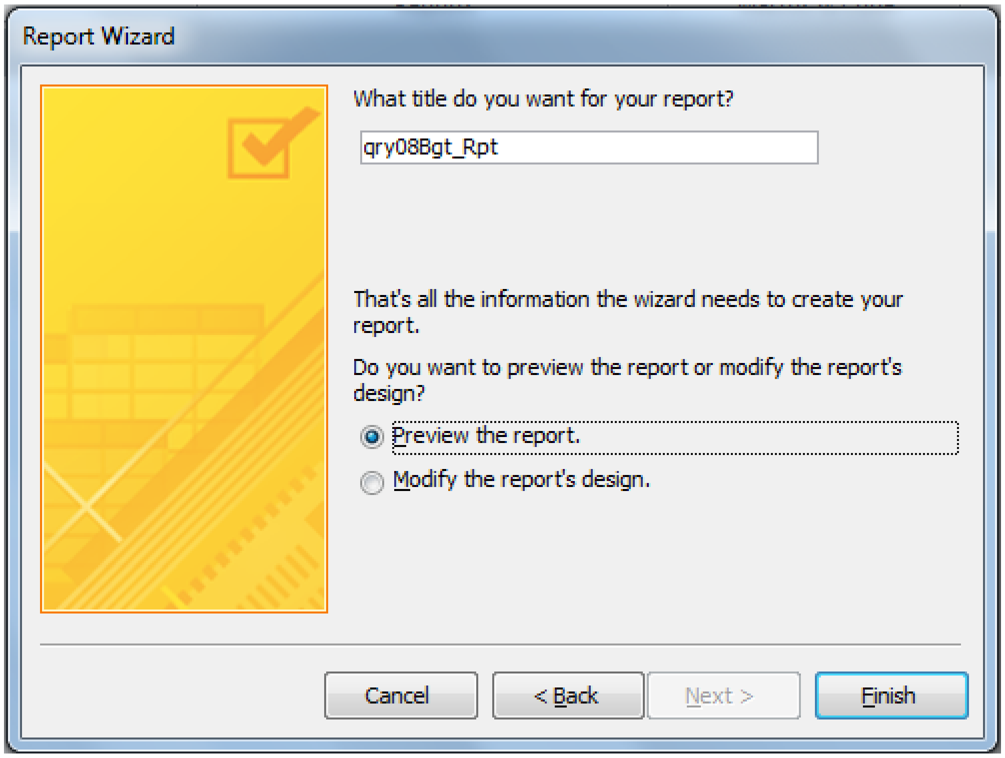
Orientation: [✓ Portrait] Landscape

[✓] 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** >.



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?

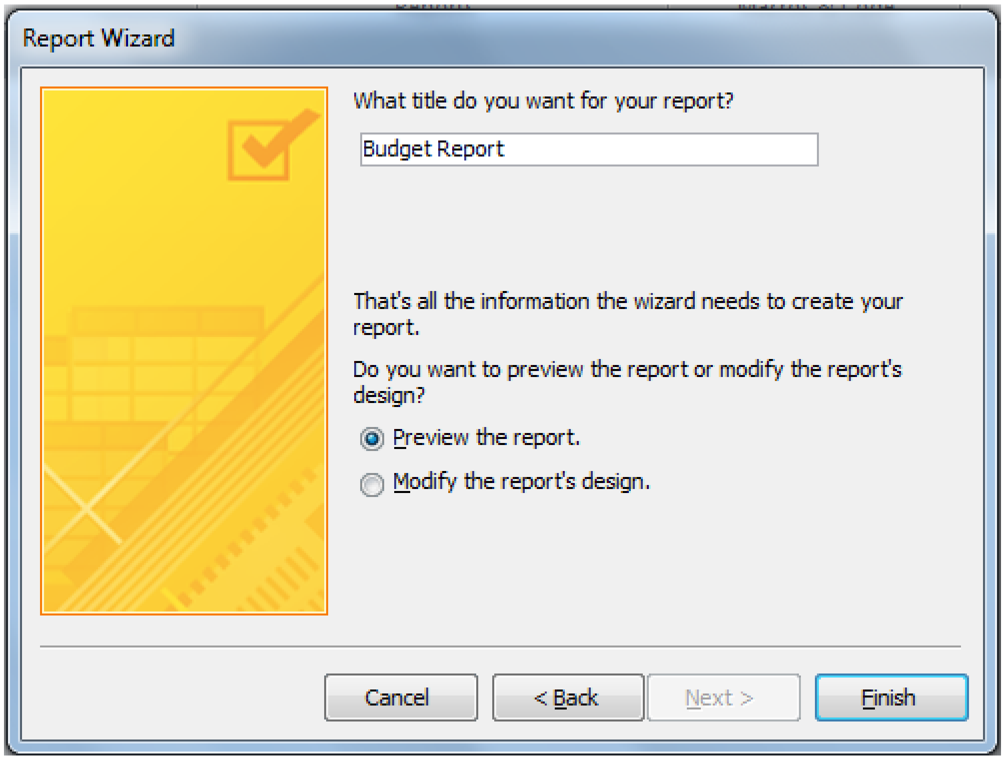
[✓ Preview the report.]

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.

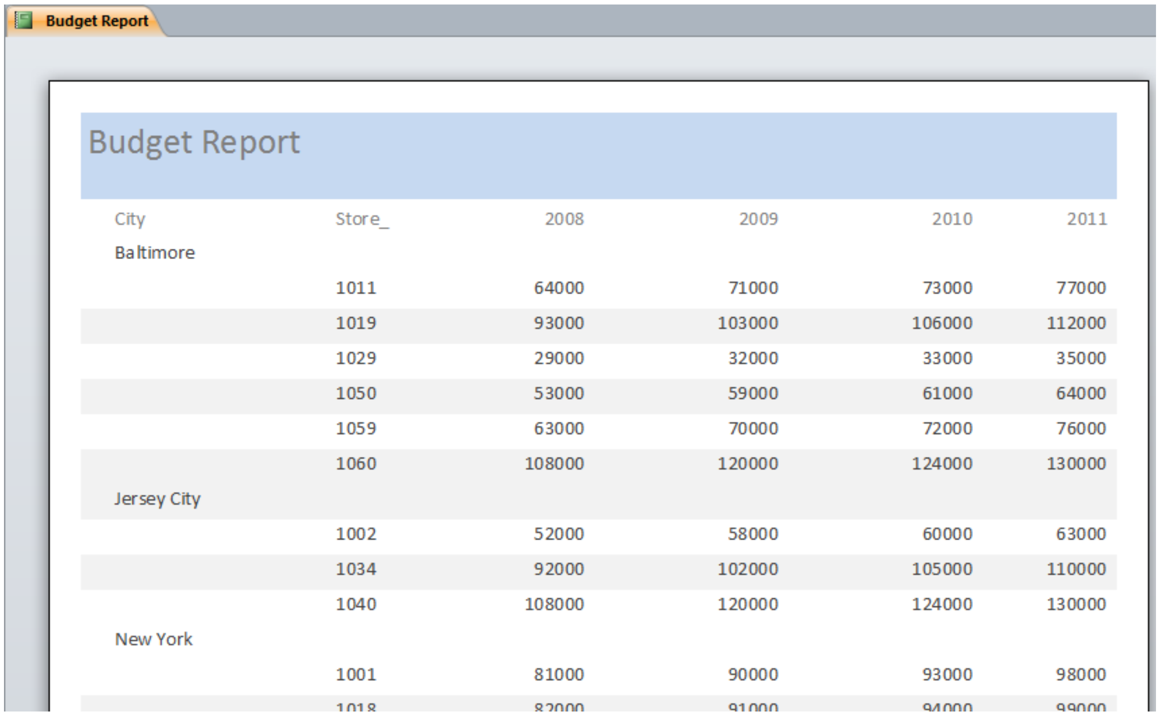
Do you want to preview the report or modify the report's design?

[✓ Preview the report.]

Modify the report's design.

Figure 8.10

17. Click **Finish.**


Budget Report

City	Store_	2008	2009	2010	2011
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
	1018	82000	91000	94000	99000

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 there is one page in the report. 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 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 of the report and clean it up.

18. Right-click anywhere in the report and choose **Design View**.

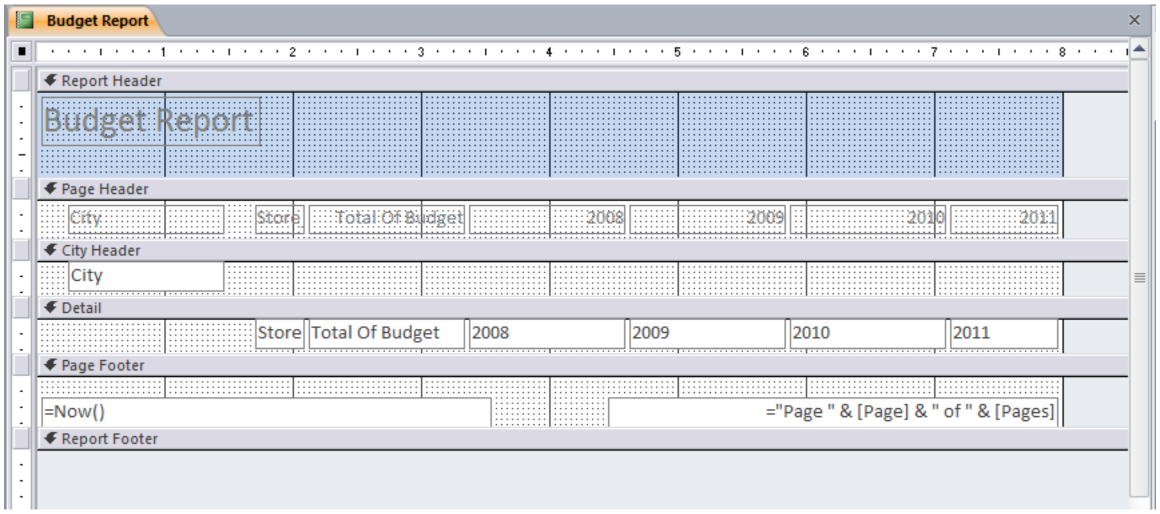


Figure 8.12

Once you click on Design View, the design grid of the report opens. You will probably see the Field List 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 soon 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, 2008, 2009, 2010, and 2011 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**.
20. *Click and drag that section down to approximately the bottom of the* **Detail** *gray bar and release.*

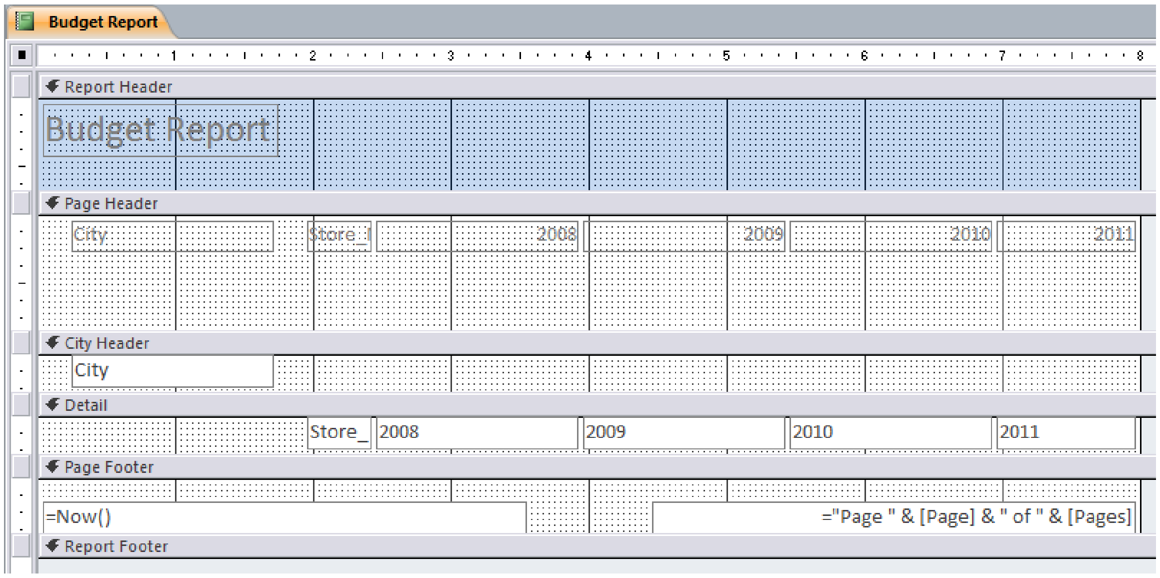


Figure 8.13

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.*

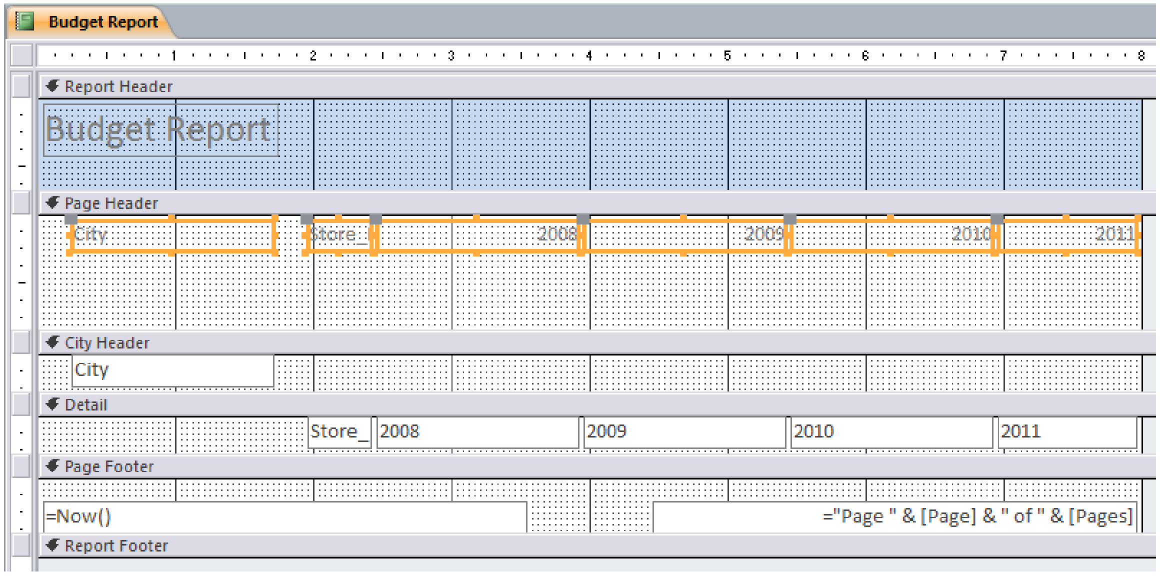


Figure 8.14

All of the label boxes are selected.

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 2010 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**.

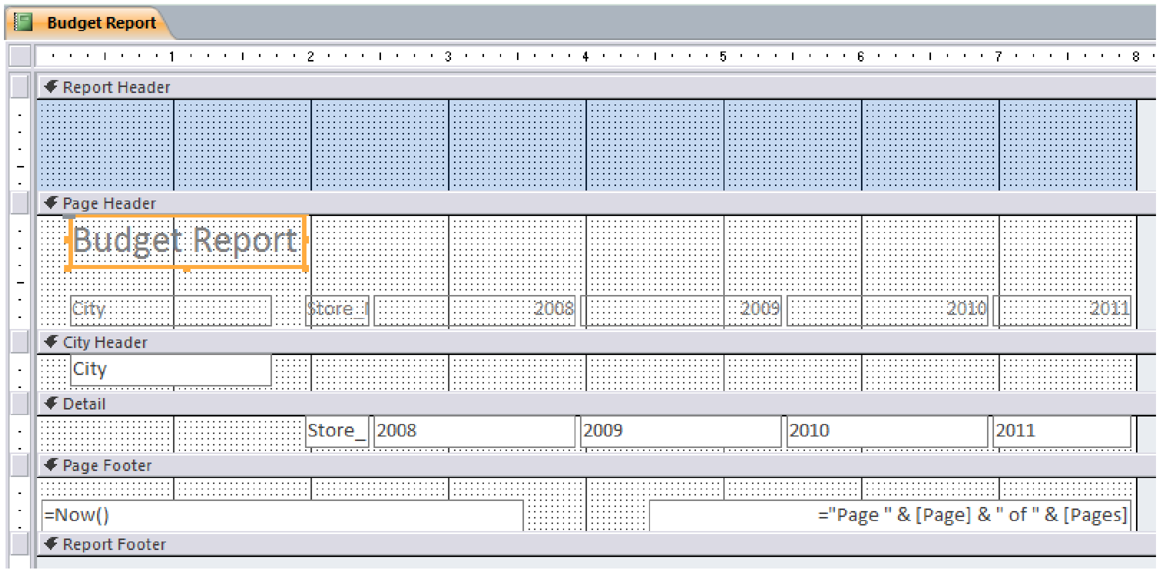
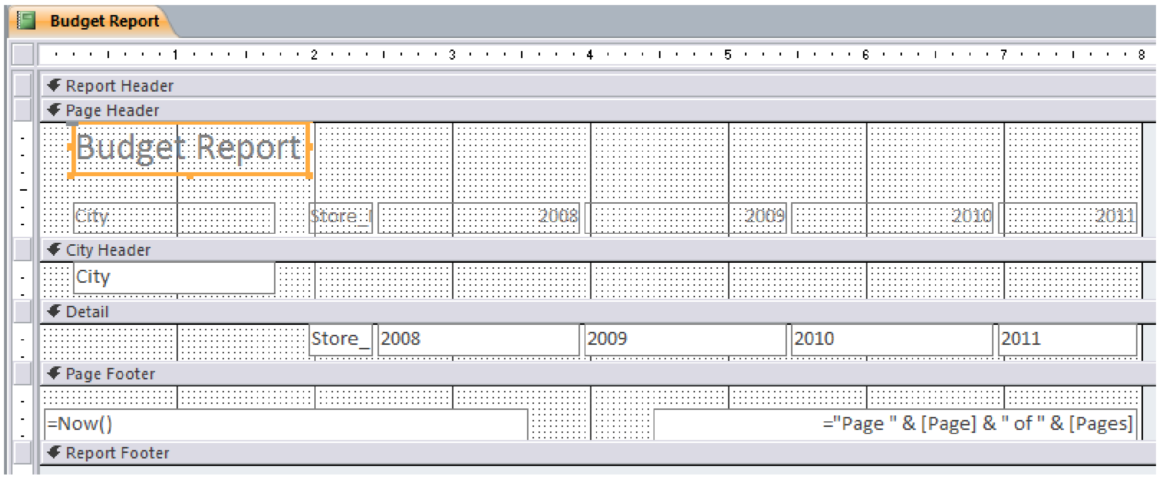


Figure 8.15

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.*



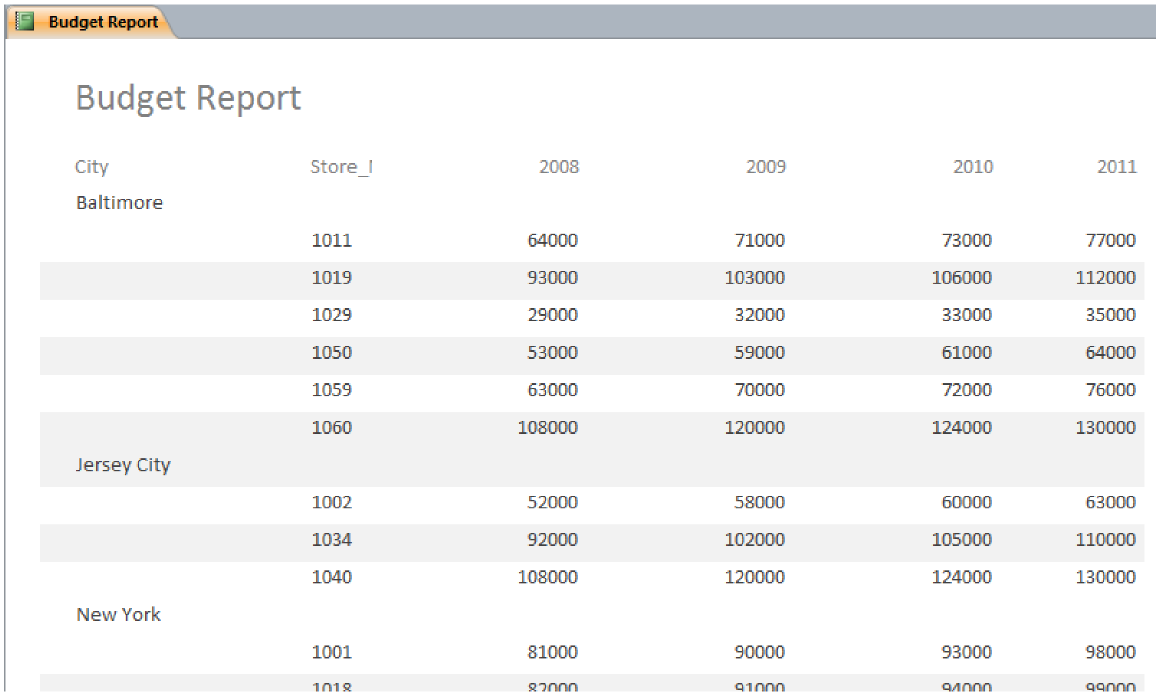
```
=Now()
="Page " & [Page] & " of " & [Pages]
```

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 **View** icon to view the report.*



Budget Report

City	Store_I	2008	2009	2010	2011
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
	1018	82000	91000	94000	99000

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) most of the label text is cut off and 2) most reports don't show an underscore character for a space. You can edit that out easily enough.

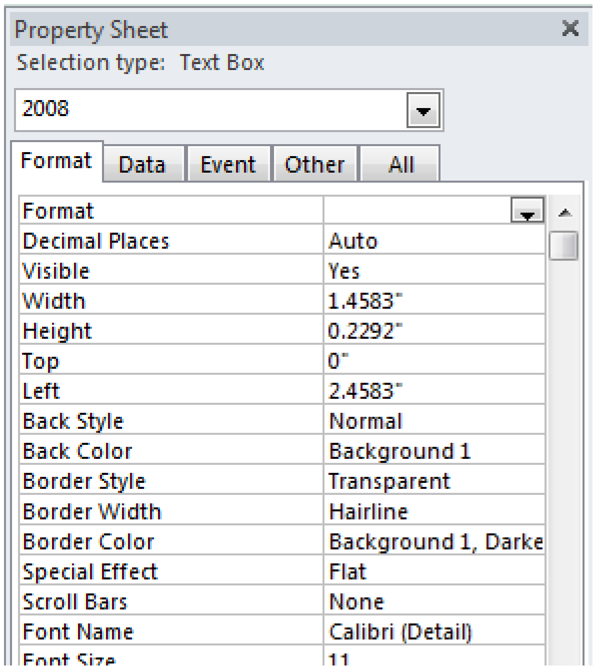
27. *In* **Design View***, click on the* **Store_No** *label, take out the* **underscore** *and replace it with a* **space***.*
28. *Expand the* **Store No** *label so that the entire phrase is visible then click somewhere outside of the label to deselect it.*
29. *Resize the* **2008** *label to where all of the* **Store No** *label appears.*

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 2010 Review Questions Chapter 8, Section 1 of 2*** *option and complete the review questions.*

Object Properties

Up to now 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* **2008 text box** *(not the label box).*
2. *Click on the* **Property Sheet** *icon (if the Property Sheet is not already displayed).*

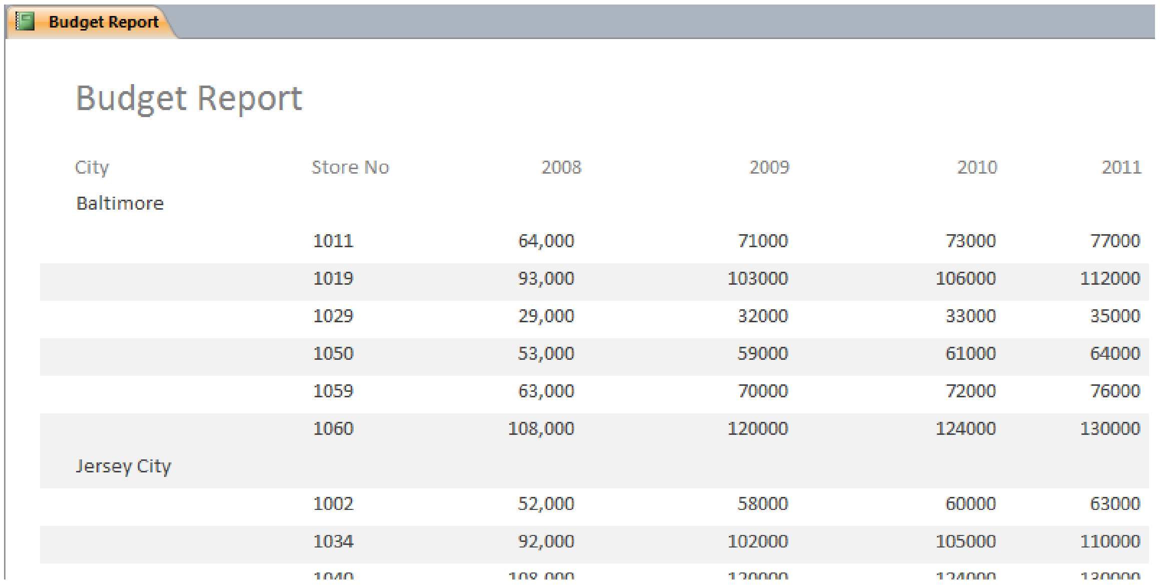

Property Sheet
Selection type: Text Box
2008
Format | Data | Event | Other | All

Format	
Decimal Places	Auto
Visible	Yes
Width	1.4583"
Height	0.2292"
Top	0"
Left	2.4583"
Back Style	Normal
Back Color	Background 1
Border Style	Transparent
Border Width	Hairline
Border Color	Background 1, Darke
Special Effect	Flat
Scroll Bars	None
Font Name	Calibri (Detail)
Font Size	11

Figure 8.18

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.*
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

Budget Report

City	Store No	2008	2009	2010	2011
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

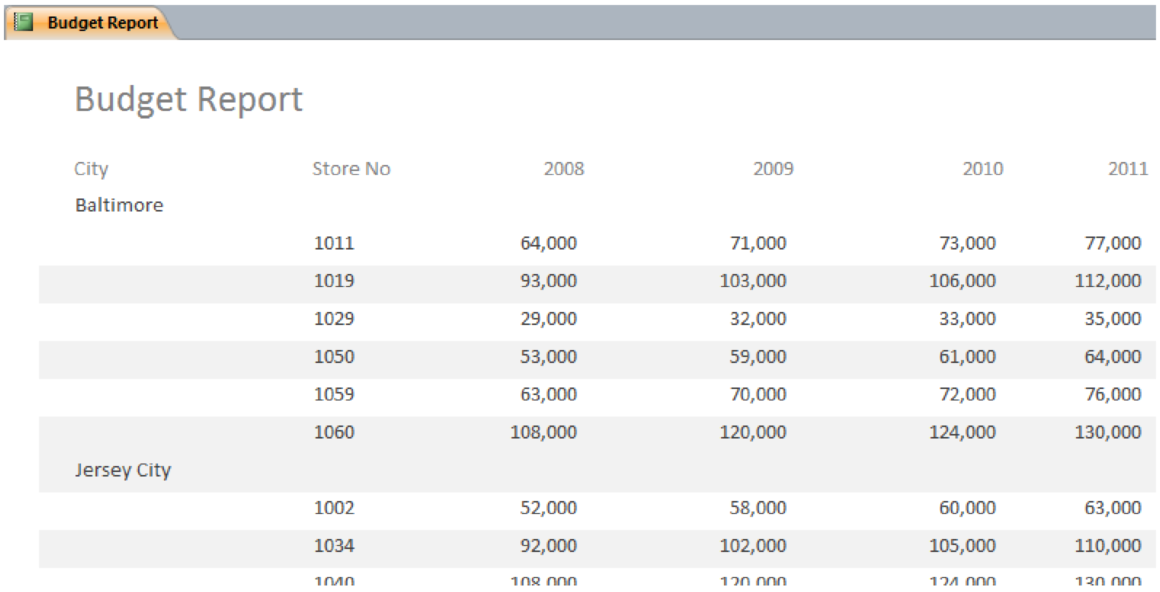
Figure 8.19

See how the budget numbers under the year 2008 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* **2009** *text box, hold down the* **[Shift]** *key, and click on the* **2010** *and* **2011** *text boxes.*

Tip: *When you click on the 2010 and 2011 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 that really fast.*

8. *With the* **2009**, **2010** *and* **2011** *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	2008	2009	2010	2011
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

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 over 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.*

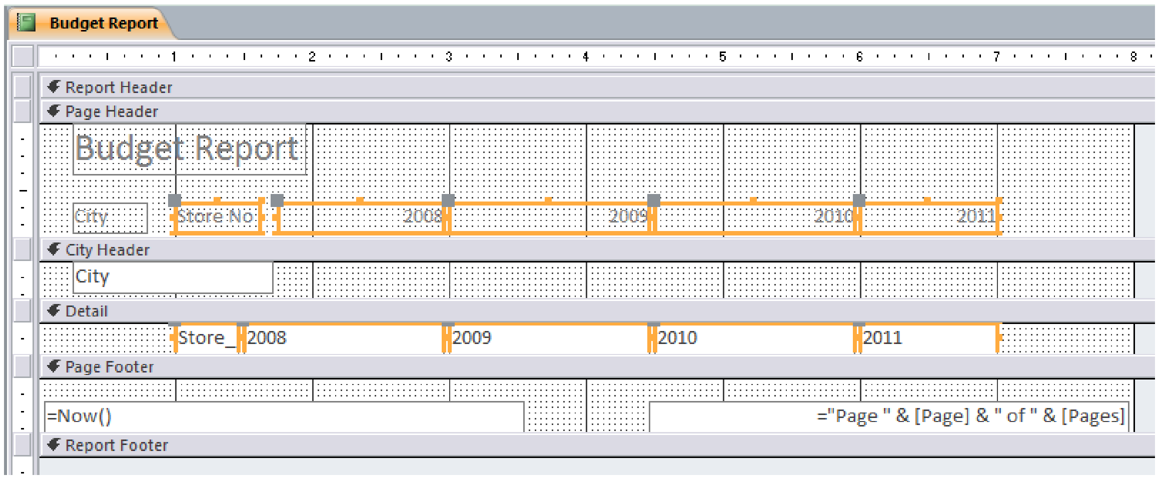
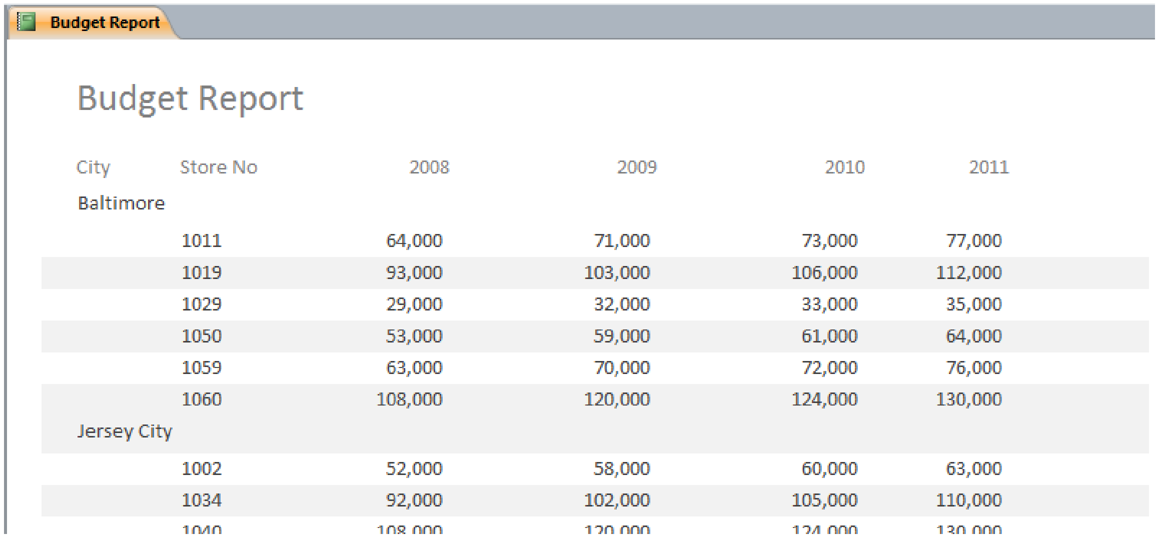


Figure 8.21

13. View the report.



Budget Report

City	Store No	2008	2009	2010	2011
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

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.*

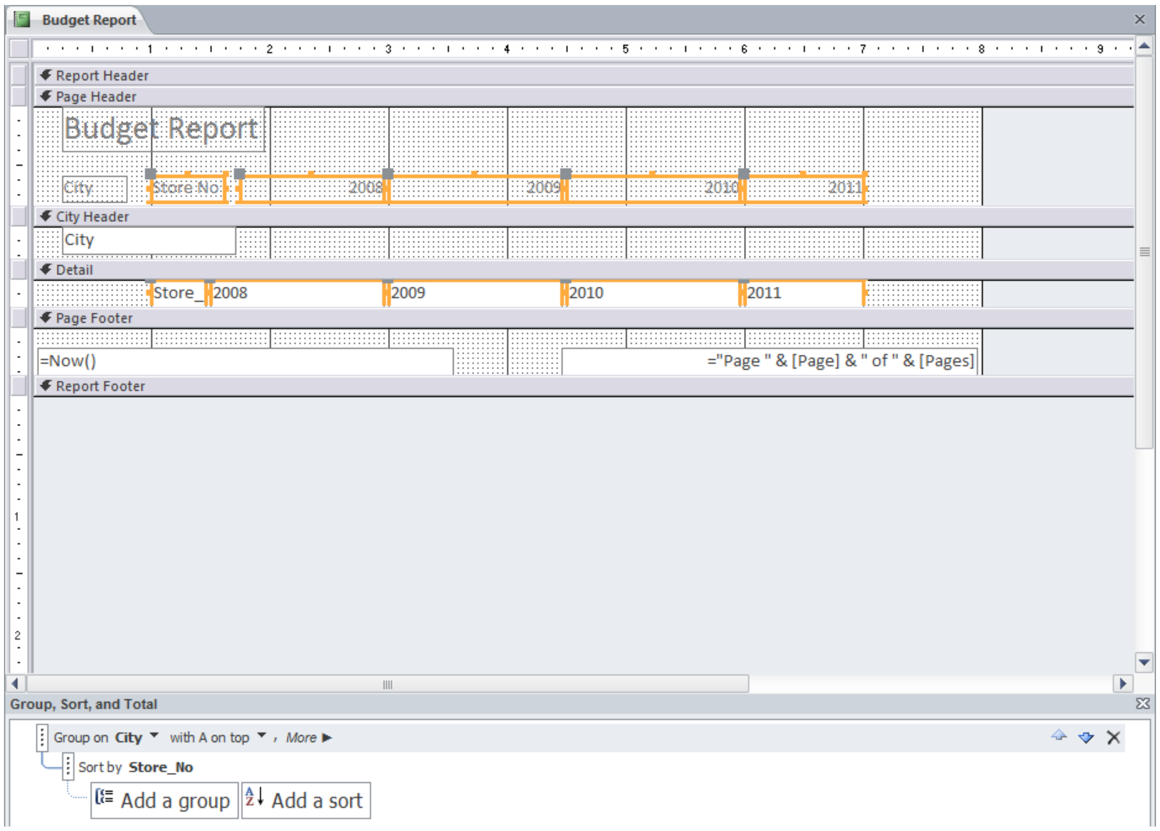
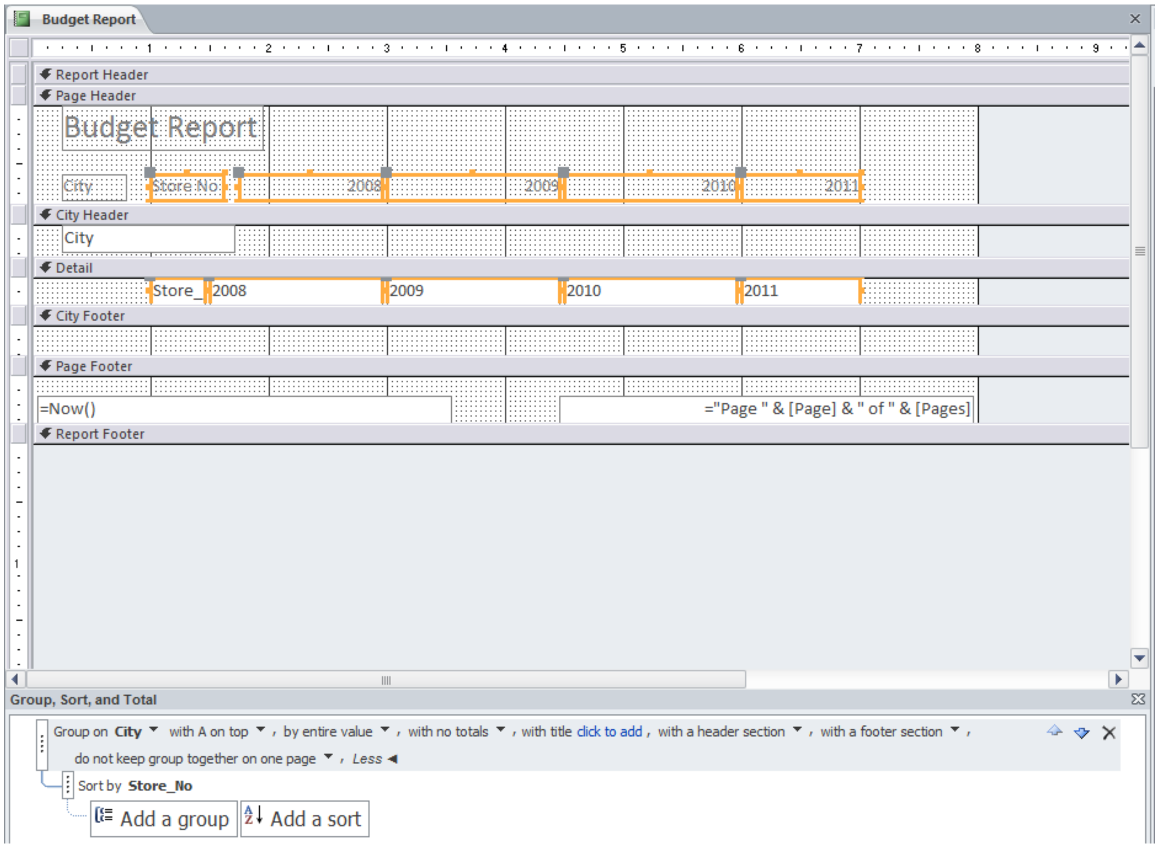


Figure 8.23

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*
Sort by **Store_No**

Figure 8.24

4. *Close the* **Group, Sort, and Total** *section.*

Creating Subtotals

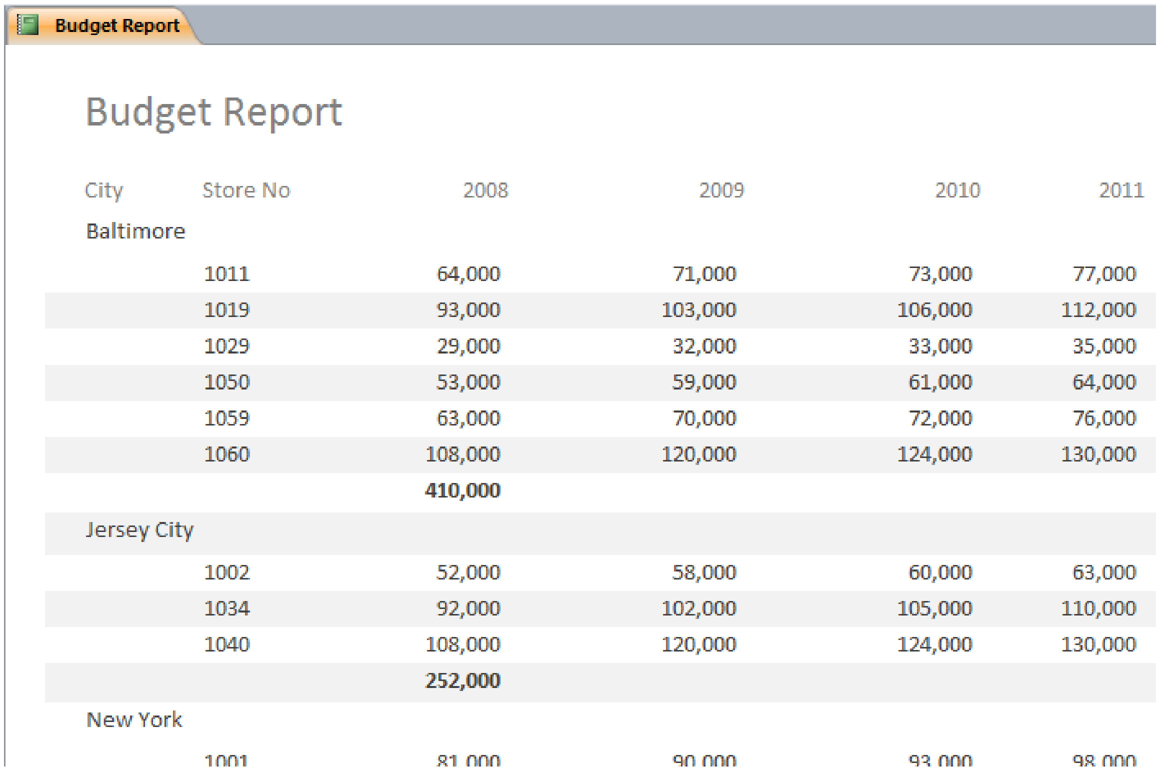
Now, you need to pay really close attention to the next part. I want to make sure you understand this very well. You will now create budget subtotals by city. In the following exercise, we will copy the text boxes 2008, 2009, 2010, and 2011 from the Detail section to the City Footer section. Then you will go into each text box and write a SUM() function around it.

5. *Click on the* **2008 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 2008 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* **2008** *text box in the* **City Footer section** *to read:* **=Sum([2008])** *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([2008])** *text box to be aligned under the* **2008** *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

City	Store No	2008	2009	2010	2011
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					
	1001	81,000	90,000	93,000	98,000

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 2008 in your formula so Access knows 2008 is the field and not the number 2008. If you add up the six values under the 2008 column for Baltimore, you will see that they add up to $410,000. Now you just have to copy that 2008 formula over to change 2009, 2010, and 2011.

12. *In* **Design view**, *make sure the* **=Sum([2008])** *text box is selected.*
13. *Copy and paste the text box in the* **City Footer** *section.*
14. *Change the new text box to read* **2009**.
15. *Do the same for* **2010** *and* **2011**.
16. *Resize all of the text boxes 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* **2011** *label box be the same width as the* **2010** *label.*

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.*

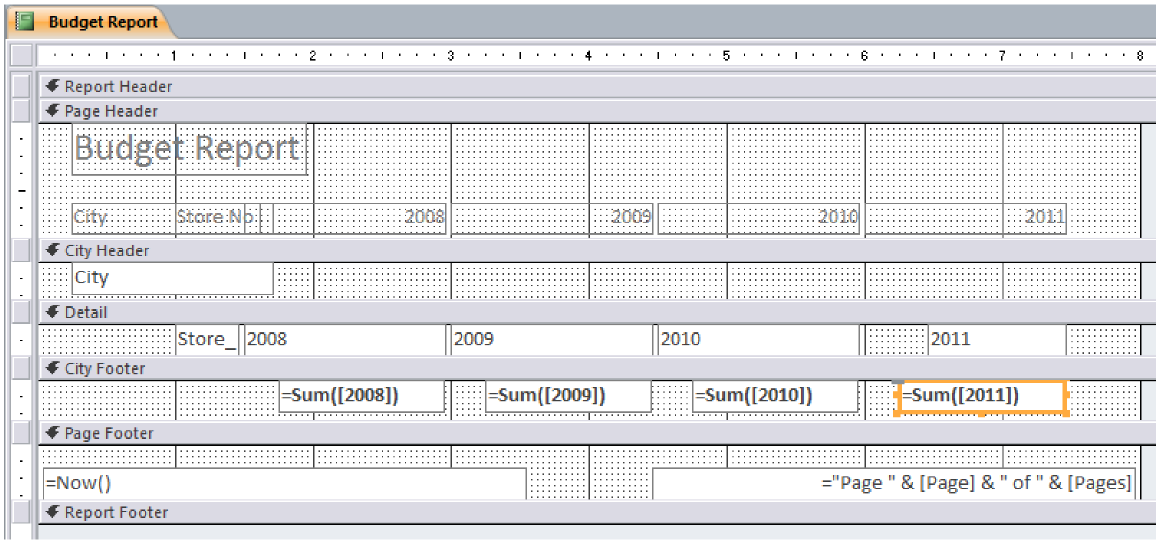
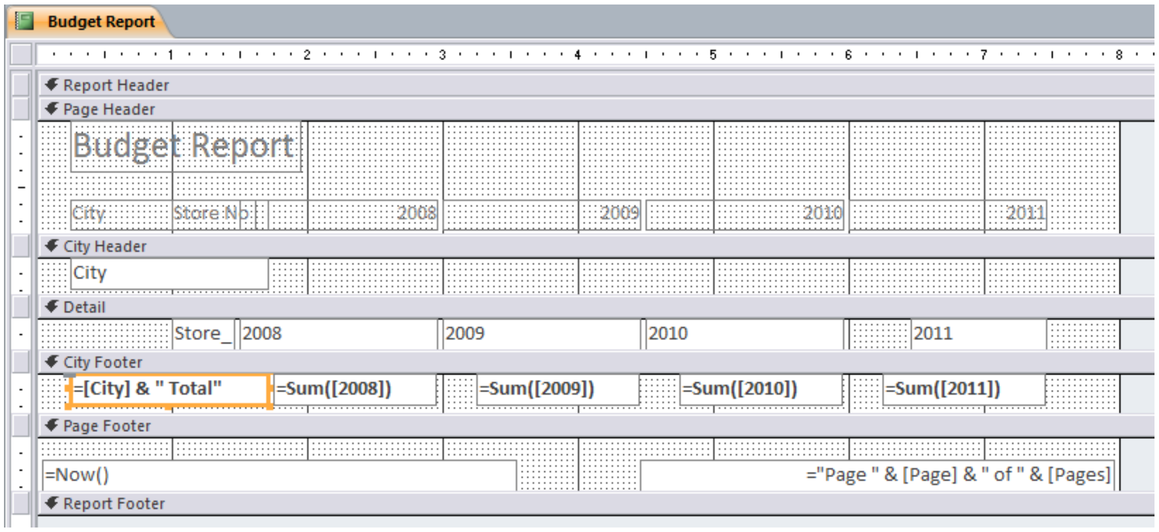


Figure 8.26

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([2008])** *text box (you could actually copy ANY text box or create a new one).*
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.*


Budget Report

Report Header

Page Header

Budget Report

City | Store No | 2008 | 2009 | 2010 | 2011

City Header

City

Detail

Store_ | 2008 | 2009 | 2010 | 2011

City Footer

=[City] & " Total" | =Sum([2008]) | =Sum([2009]) | =Sum([2010]) | =Sum([2011])

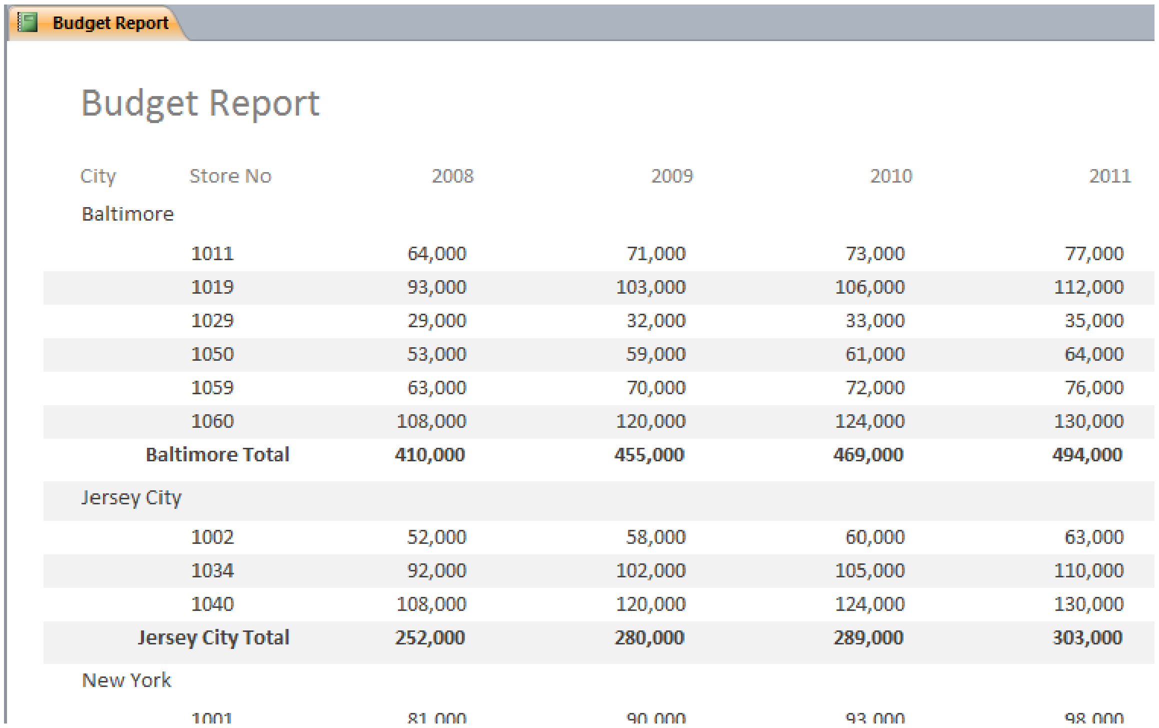
Page Footer

=Now() | ="Page " & [Page] & " of " & [Pages]

Report Footer

Figure 8.27

24. View the report.


Budget Report

Budget Report

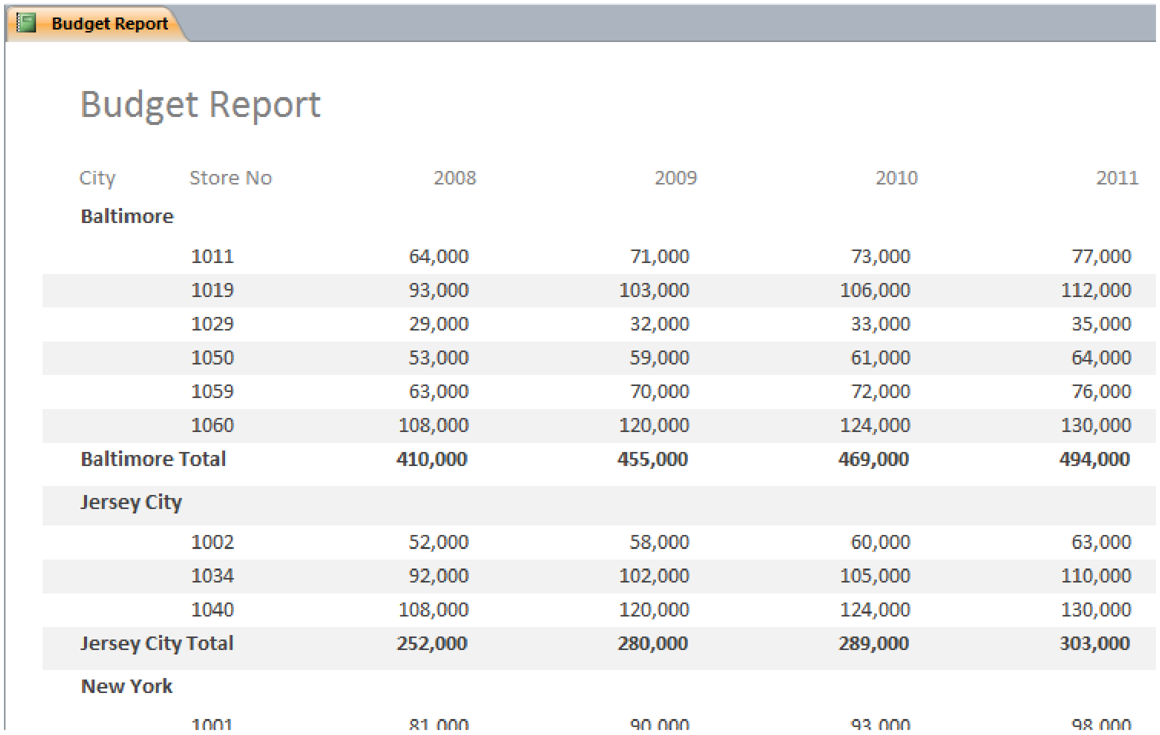
City	Store No	2008	2009	2010	2011
Baltimore					
	1011	64,000	71,000	73,000	77,000
	1019	93,000	103,000	106,000	112,000
	1029	29,000	32,000	33,000	35,000
	1050	53,000	59,000	61,000	64,000
	1059	63,000	70,000	72,000	76,000
	1060	108,000	120,000	124,000	130,000
	Baltimore Total	**410,000**	**455,000**	**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

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.*
27. View the report.



Budget Report

Budget Report

City	Store No	2008	2009	2010	2011
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

Figure 8.29

If you go to the bottom of the report in Print Preview mode, you will see that it ends with 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. 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 2008**.

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([2008])** *and press* **[Enter]**.
37. *Create other text boxes for the years* **2009**, **2010**, *and* **2011**.

The Report Footer section should now look something like this:

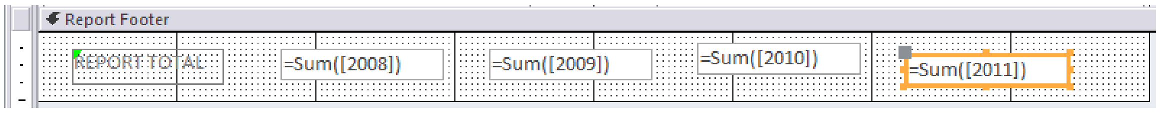


Figure 8.30

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.

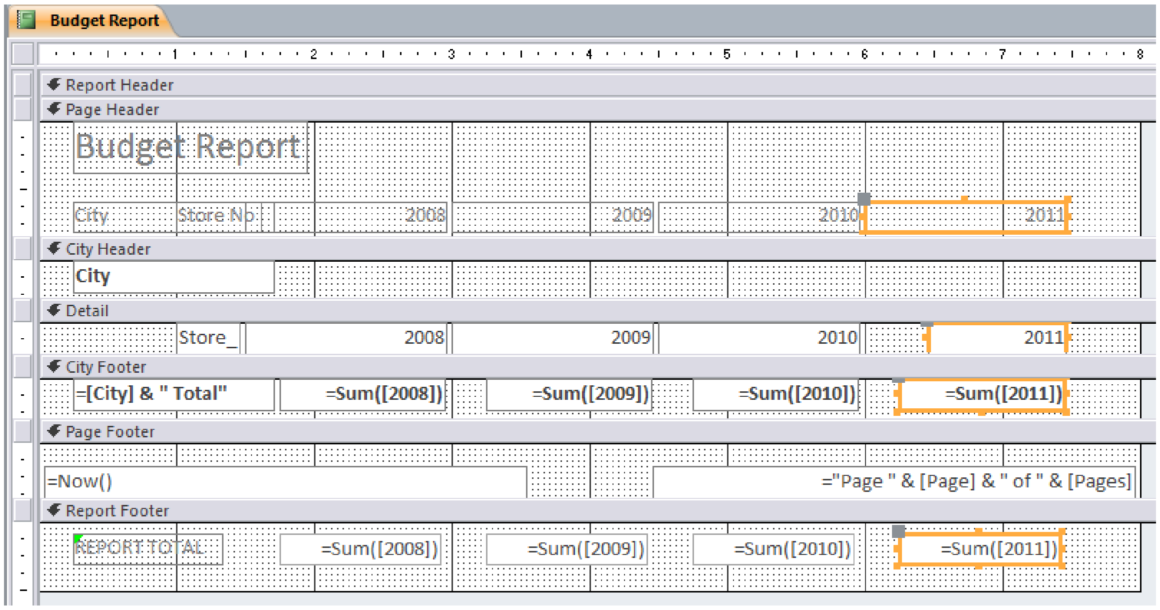
38. *Click in the* **Left Ruler** *and select all of the text boxes and labels in the* **Report Footer** *section.*
39. *Click on the* **Arrange** *tab of the* **Report Design Tools** *contextual tab, and click on the* **Size/Space** *icon in the* **Sizing and Ordering** *group.*
40. *Click on the* **To Shortest** *icon under* **Size**.
41. *Then click on the* **Align** *icon in the same group and click on* **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* **2008 label** *in the* **Page Header**, *hold down the* **[Shift]** *key and click on the* **2008 text box** *in the* **Detail** *section, the* **Sum([2008]) text box** *in the* **City Footer** *section, and the* **Sum([2008]) 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 **2009**, **2010,** *and* **2011***.*

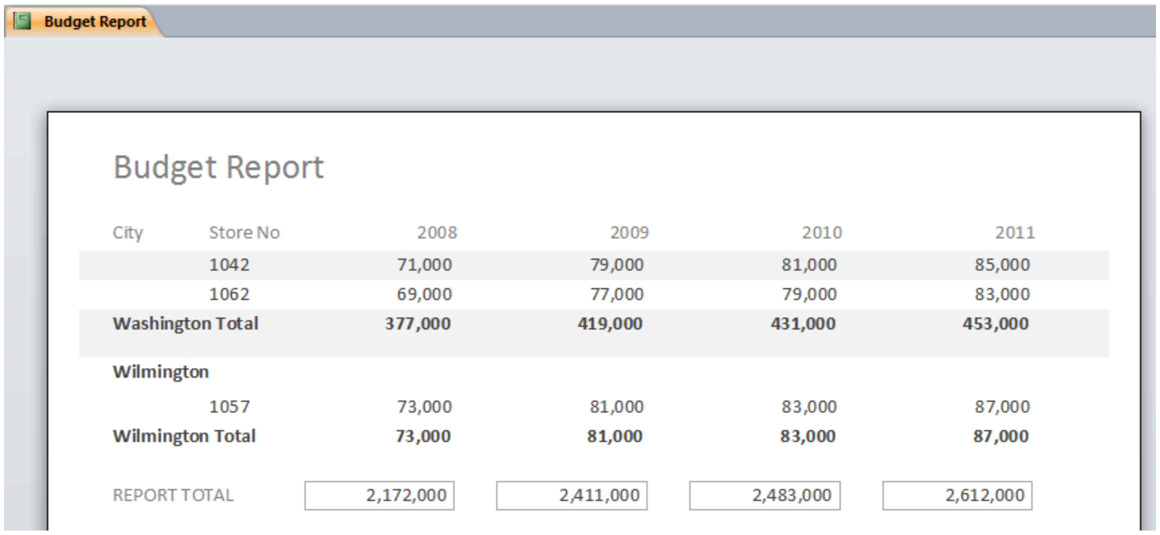
The Design grid of the report should look like the figure below.



```
Page Header:   Budget Report
               City   Store No   2008   2009   2010   2011
City Header:   City
Detail:        Store_   2008   2009   2010   2011
City Footer:   =[City] & " Total"   =Sum([2008])   =Sum([2009])   =Sum([2010])   =Sum([2011])
Page Footer:   =Now()   ="Page " & [Page] & " of " & [Pages]
Report Footer: REPORT TOTAL   =Sum([2008])   =Sum([2009])   =Sum([2010])   =Sum([2011])
```

Figure 8.31

47. Run the report in **Print Preview** *and go to the second page.*



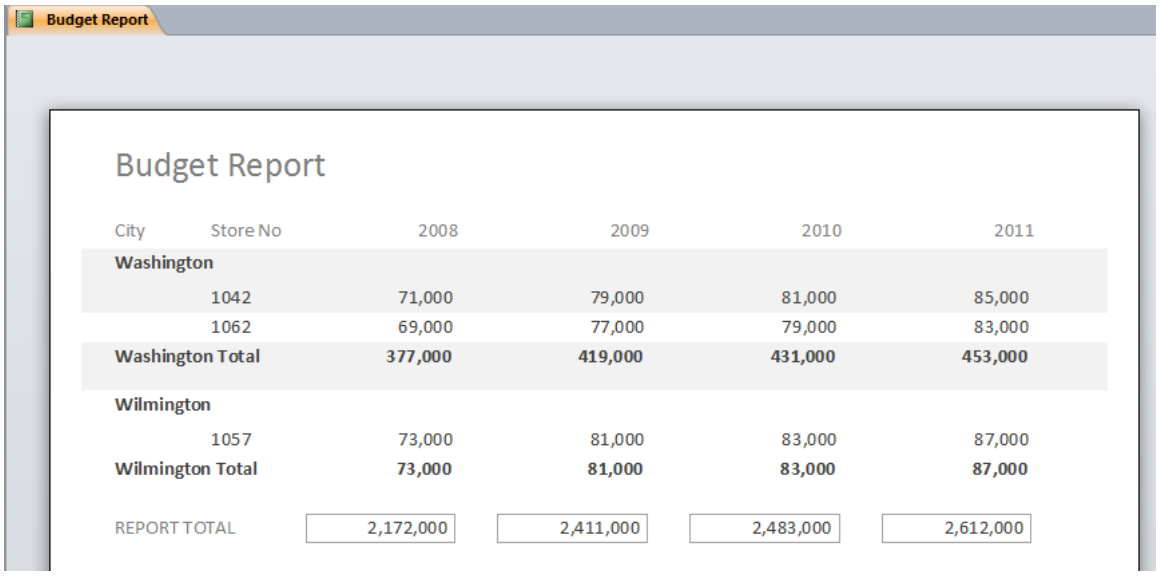
Budget Report

City	Store No	2008	2009	2010	2011
	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 go to the second page.*



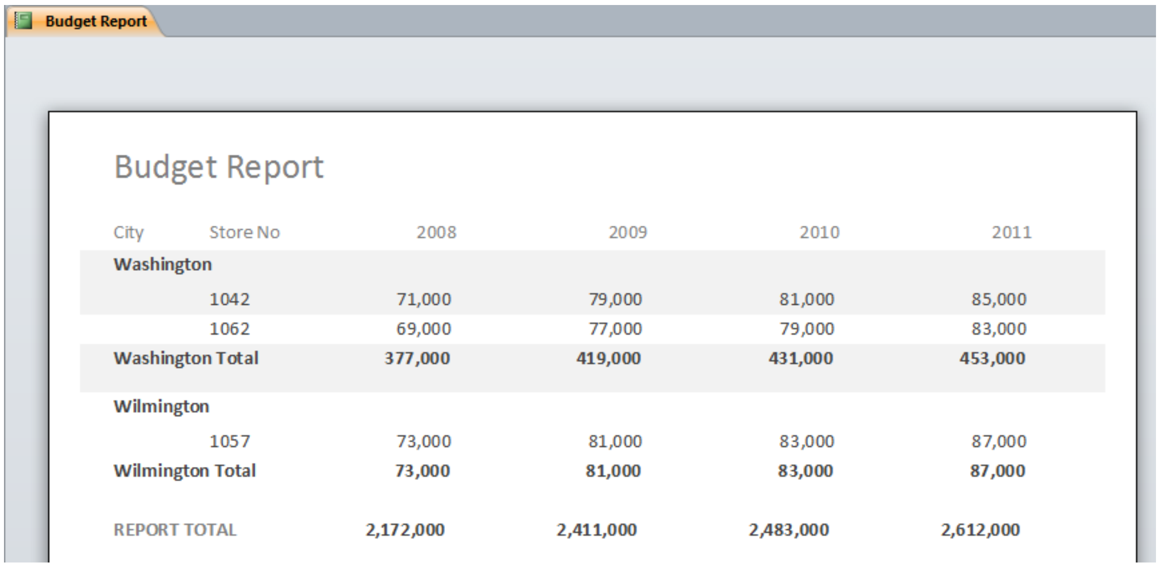
Budget Report

City	Store No	2008	2009	2010	2011
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 TOTALS 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.*



Budget Report

City	Store No	2008	2009	2010	2011
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 8.34

54. *Save and close the 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 2010 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, log in and take the exam. Make sure that you take the exam on the same computer that you completed the sample files on, as some of the questions on the exam may refer to some of the completed examples.

ExcelCEO

Chief Excel Officer

Access 2010 and SQL

Complete Self-study Course

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.

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.

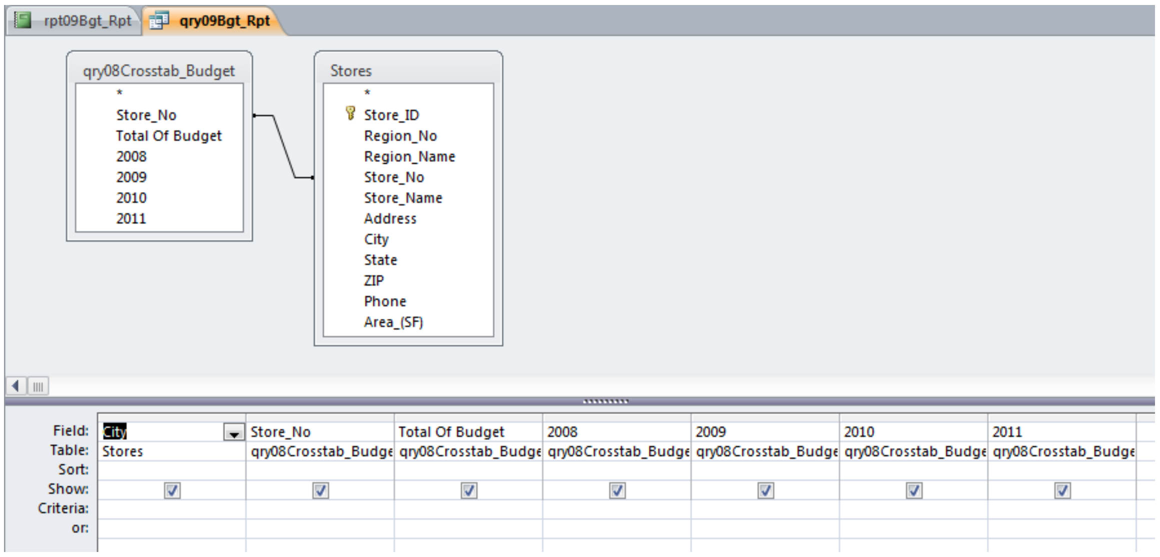
1. *Open the* **Nitey_Nite_2010** *database, close the* **Main Menu** *form, copy* **qry08Bgt_Rpt** *and name the copy* **qry09Bgt_Rpt**.
2. *Copy* **rpt08Bgt_Rpt** *and name it* **rpt09Bgt_Rpt**.

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**.



Field:	City	Store_No	Total Of Budget	2008	2009	2010	2011
Table:	Stores	qry08Crosstab_Budge	qry08Crosstab_Budge	qry08Crosstab_Budge	qry08Crosstab_Budge	qry08Crosstab_Budge	qry08Crosstab_Budge
Sort:							
Show:	✓	✓	✓	✓	✓	✓	✓
Criteria:							
or:							

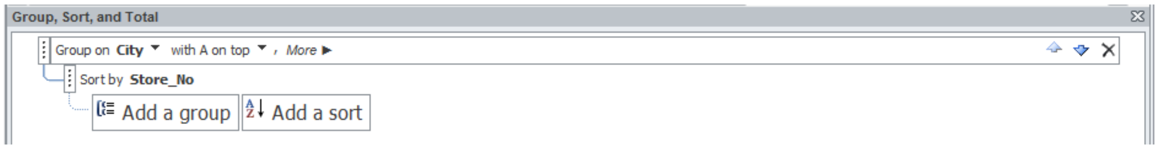
Figure 9.1

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 the query.*

Now that the field is in the query, you need to create a report header and footer for that level.

9. *In the* **Design** *tab of* **rpt09Bgt_Rpt**, *click on the* **Group & Sort** *icon.*


Group, Sort, and Total

Group on **City** with A on top , More ▶

Sort by **Store_No**

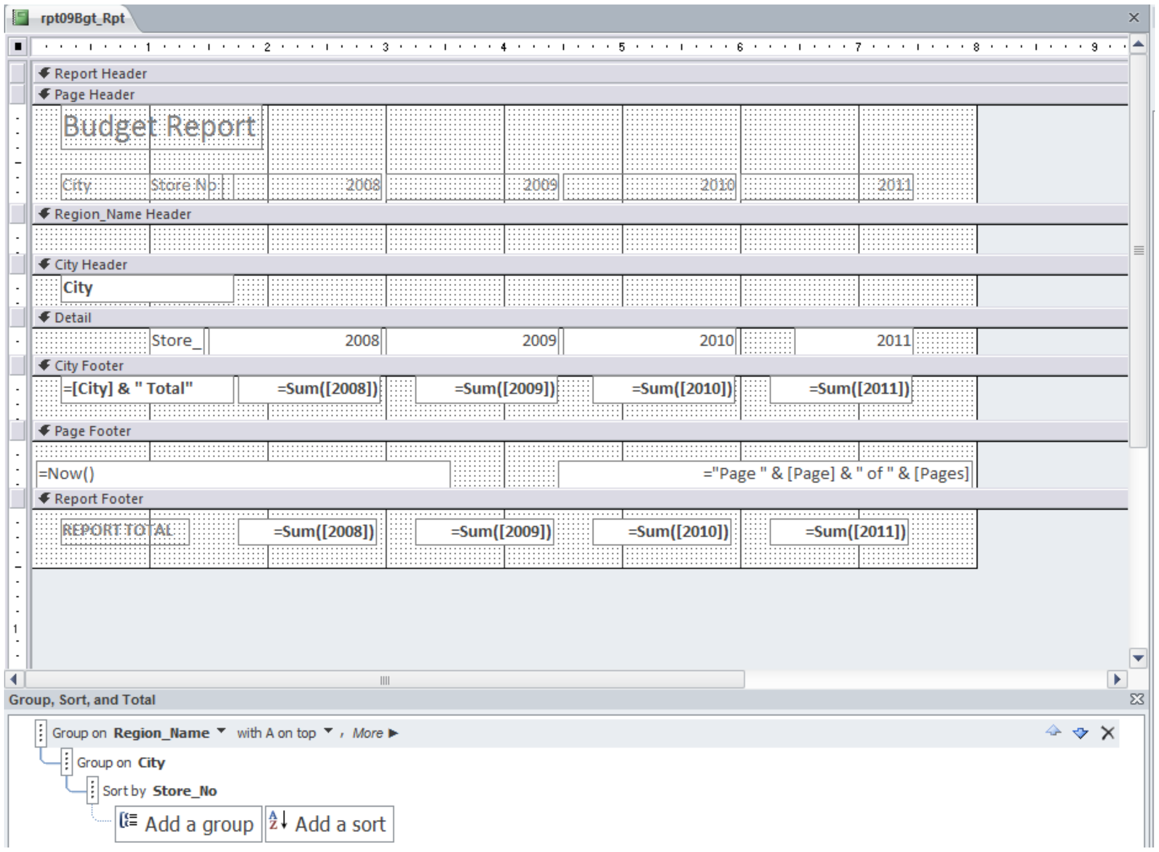
Figure 9.2

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 Group on Store_No** *and choose* **Region_Name**.

11. *When the* **Region_Name** *group appears below* **Group on Store_No**, *click and drag it above the* **Group on City** *group (or use the up arrow) (Note: You may have to click and drag on the icon to the left of the group.)*

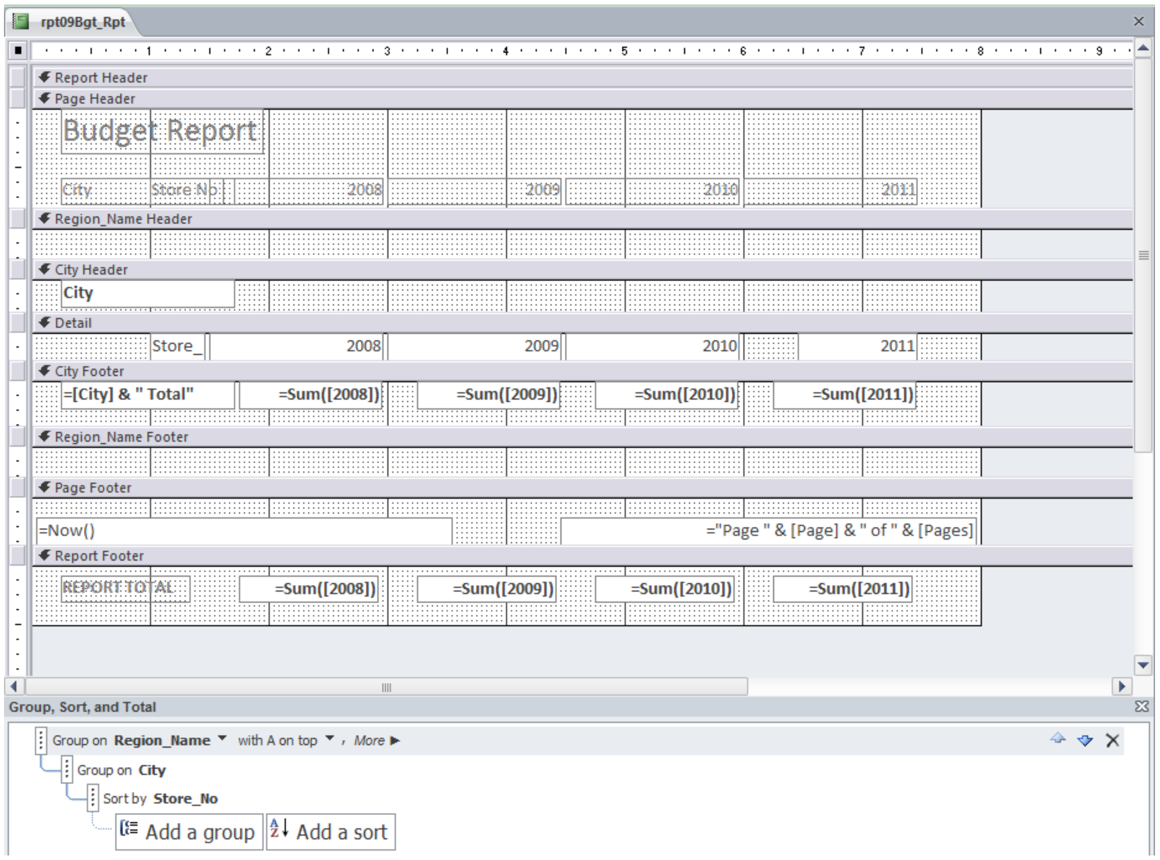


Group on **Region_Name** with A on top , More ►
Group on **City**
Sort by **Store_No**

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.*


rpt09Bgt_Rpt

Report Header

Page Header

Budget Report

City | Store No | 2008 | 2009 | 2010 | 2011

Region_Name Header

City Header

City

Detail

Store_ | 2008 | 2009 | 2010 | 2011

City Footer

```
=[City] & " Total"   =Sum([2008])   =Sum([2009])   =Sum([2010])   =Sum([2011])
```

Region_Name Footer

Page Footer

```
=Now()   ="Page " & [Page] & " of " & [Pages]
```

Report Footer

```
REPORT TOTAL   =Sum([2008])   =Sum([2009])   =Sum([2010])   =Sum([2011])
```

Group, Sort, and Total

Group on **Region_Name** with A on top , More ▶

Group on **City**

Sort by **Store_No**

Add a group | Add a sort

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** *label 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:*

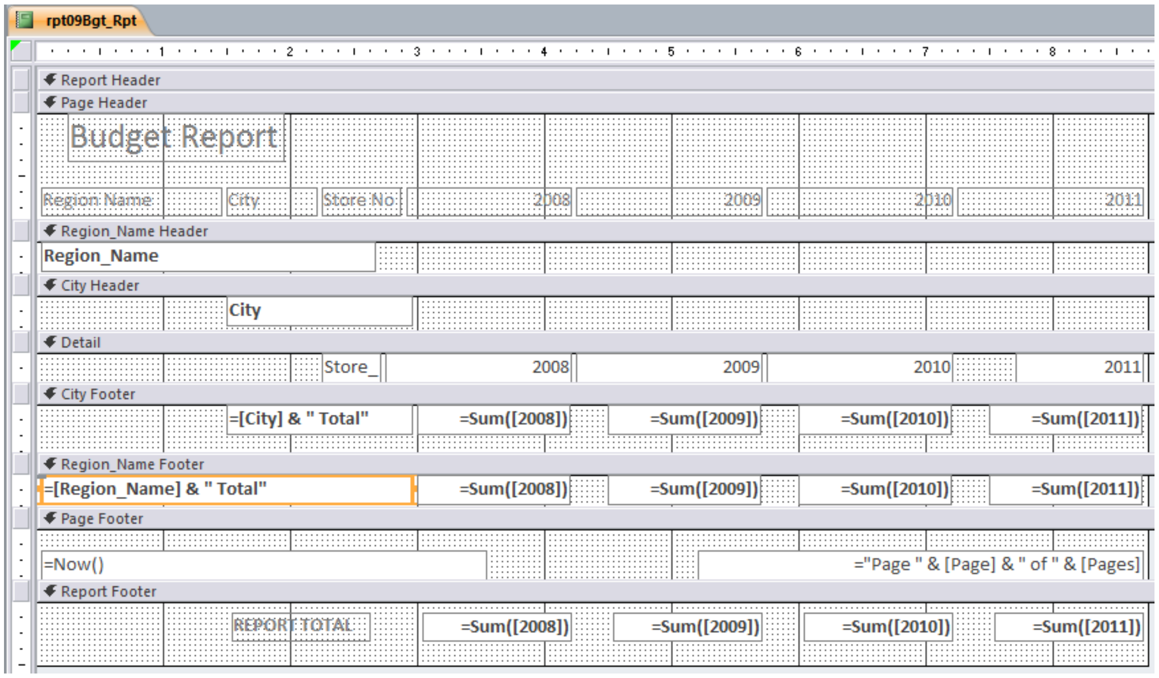
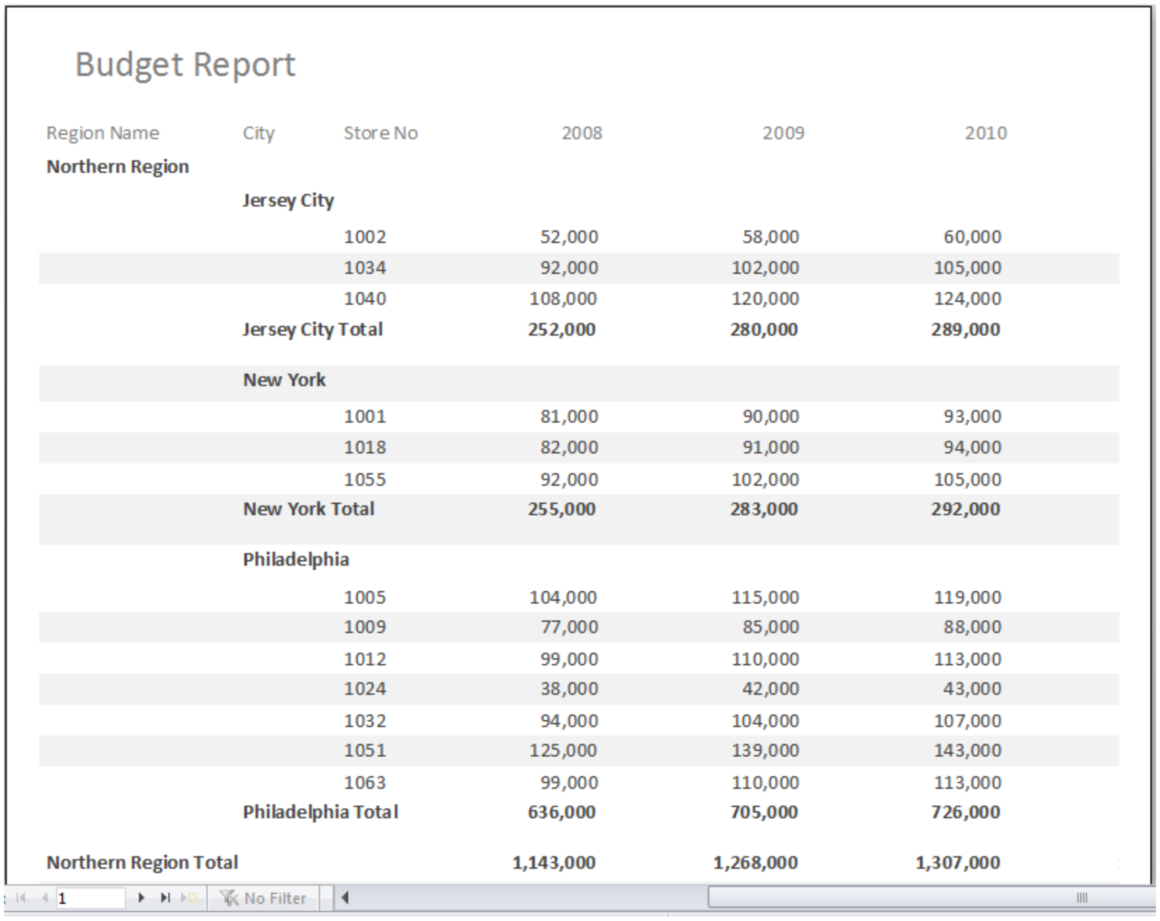


Figure 9.5

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	2008	2009	2010
Northern Region					
	Jersey City				
		1002	52,000	58,000	60,000
		1034	92,000	102,000	105,000
		1040	108,000	120,000	124,000
	Jersey City Total		**252,000**	**280,000**	**289,000**
	New York				
		1001	81,000	90,000	93,000
		1018	82,000	91,000	94,000
		1055	92,000	102,000	105,000
	New York Total		**255,000**	**283,000**	**292,000**
	Philadelphia				
		1005	104,000	115,000	119,000
		1009	77,000	85,000	88,000
		1012	99,000	110,000	113,000
		1024	38,000	42,000	43,000
		1032	94,000	104,000	107,000
		1051	125,000	139,000	143,000
		1063	99,000	110,000	113,000
	Philadelphia Total		**636,000**	**705,000**	**726,000**
Northern Region Total			**1,143,000**	**1,268,000**	**1,307,000**

Figure 9.6

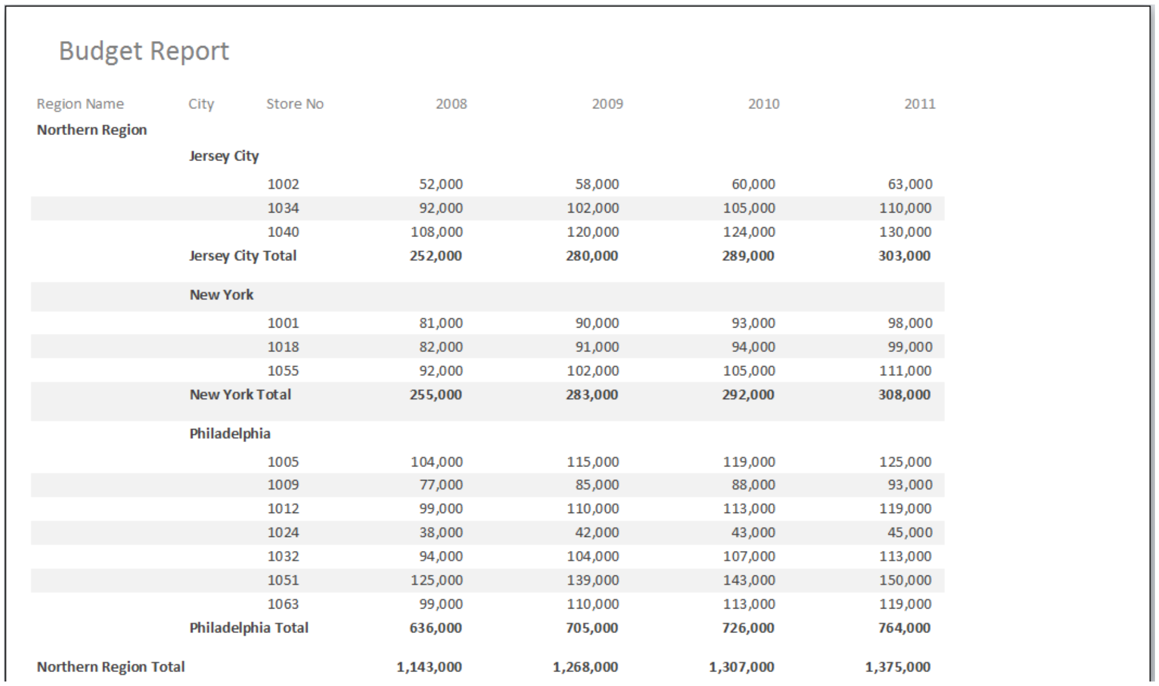
Changing Page Orientation

Whoa there! Where did the data for the year 2011 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 mode. Not to worry, we can easily change the Page Setup settings.

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.*

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.

3. *Click* **OK** *to view the report in* **Print Preview**.



Budget Report

Region Name	City	Store No	2008	2009	2010	2011
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
		1032	94,000	104,000	107,000	113,000
		1051	125,000	139,000	143,000	150,000
		1063	99,000	110,000	113,000	119,000
	Philadelphia Total		**636,000**	**705,000**	**726,000**	**764,000**
Northern Region Total			**1,143,000**	**1,268,000**	**1,307,000**	**1,375,000**

Figure 9.7

The report should now appear on two pages, with all columns appearing on each page.

> ***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 2010 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 2008 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 2008 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 size of the design grid to* **10** *inches.*
2. *Select all text boxes under and including the headings* **2008 – 2011**.
3. *In the* **Property Sheet**, *make all 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 like they do in the following figure.*

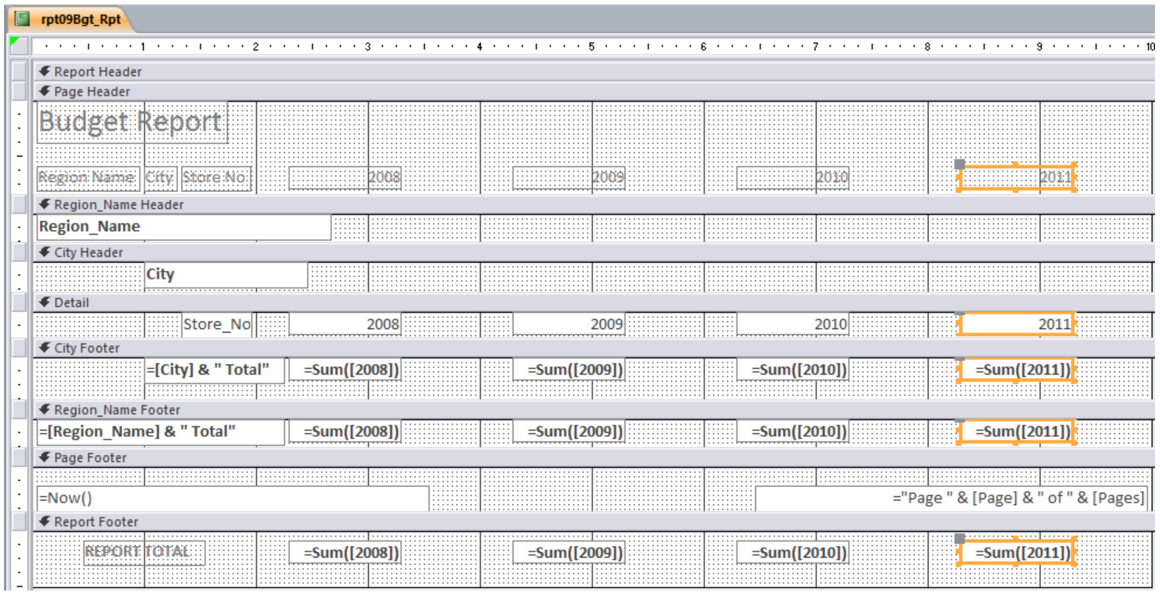
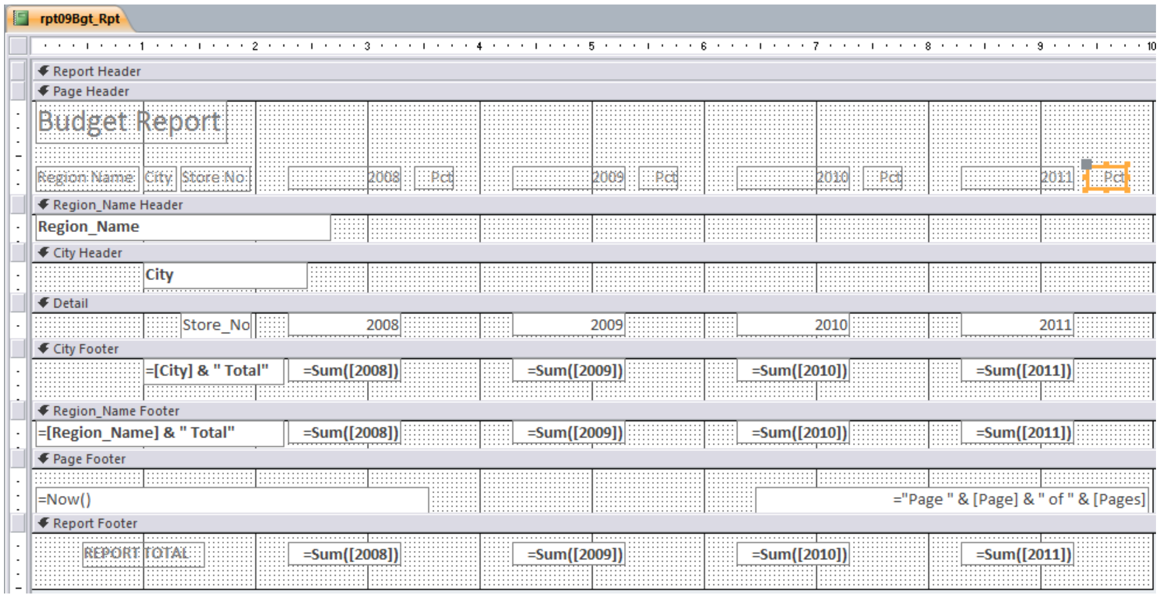


Figure 9.8

Setting Report Margins

When you resize text boxes that contain data, make sure you go back and forth between Design View and the view of the report 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**.



rpt09Bgt_Rpt

Report Header

Page Header

Budget Report

Region Name | City | Store No | 2008 | Pct | 2009 | Pct | 2010 | Pct | 2011 | Pct

Region_Name Header

Region_Name

City Header

City

Detail

Store_No | 2008 | 2009 | 2010 | 2011

City Footer

=[City] & " Total" | =Sum([2008]) | =Sum([2009]) | =Sum([2010]) | =Sum([2011])

Region_Name Footer

=[Region_Name] & " Total" | =Sum([2008]) | =Sum([2009]) | =Sum([2010]) | =Sum([2011])

Page Footer

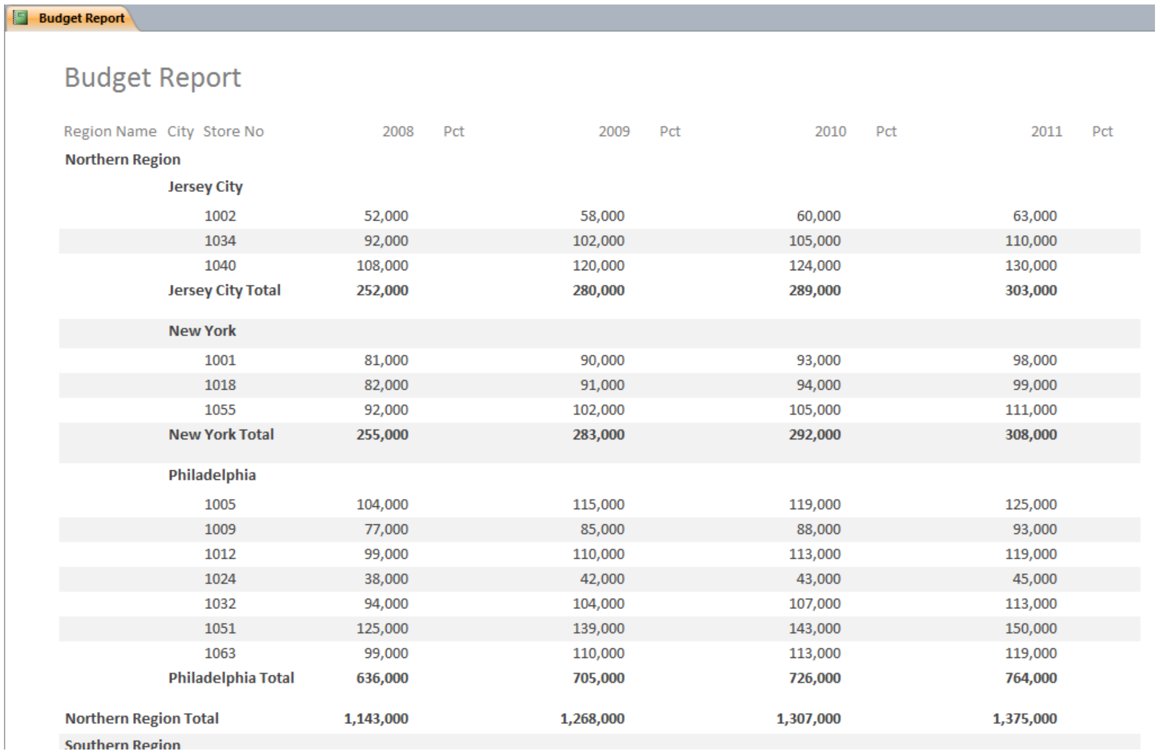
=Now() | ="Page " & [Page] & " of " & [Pages]

Report Footer

REPORT TOTAL | =Sum([2008]) | =Sum([2009]) | =Sum([2010]) | =Sum([2011])

Figure 9.9

8. *Run the report.*



Budget Report

Budget Report

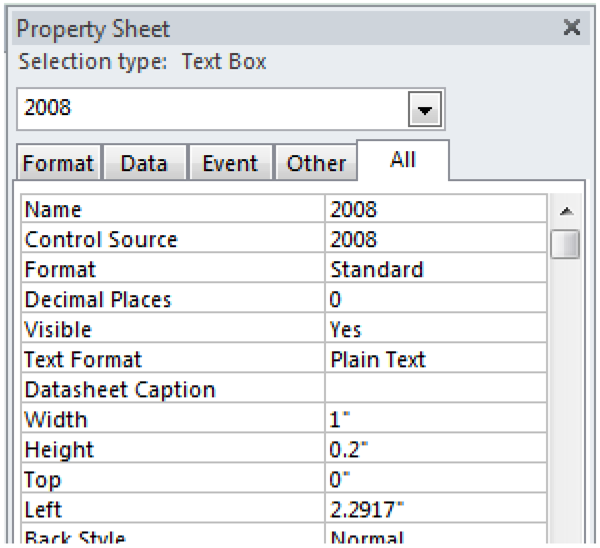
Region Name	City	Store No	2008	Pct	2009	Pct	2010	Pct	2011	Pct
Northern Region										
	Jersey City									
		1002	52,000		58,000		60,000		63,000	
		1034	92,000		102,000		105,000		110,000	
		1040	108,000		120,000		124,000		130,000	
	Jersey City Total		252,000		280,000		289,000		303,000	
	New York									
		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	
		1032	94,000		104,000		107,000		113,000	
		1051	125,000		139,000		143,000		150,000	
		1063	99,000		110,000		113,000		119,000	
	Philadelphia Total		636,000		705,000		726,000		764,000	
Northern Region Total			1,143,000		1,268,000		1,307,000		1,375,000	
Southern Region										

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 that 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 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 all you have to do is 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 2008 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* **2008** *text box in the* **Detail** *section and view the properties of that box.*

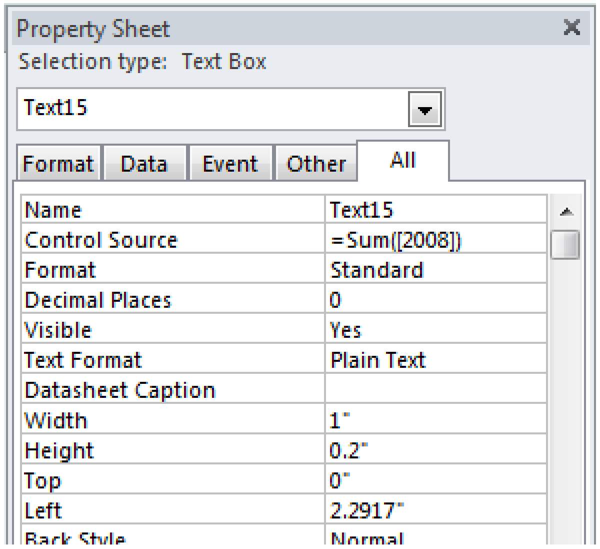


Name	2008
Control Source	2008
Format	Standard
Decimal Places	0
Visible	Yes
Text Format	Plain Text
Datasheet Caption	
Width	1"
Height	0.2"
Top	0"
Left	2.2917"
Back Style	Normal

Figure 9.11

The name of the text box is "2008". 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([2008])** *text box in the* **City Footer** *section and look at its properties.*


Property Sheet
Selection type: Text Box

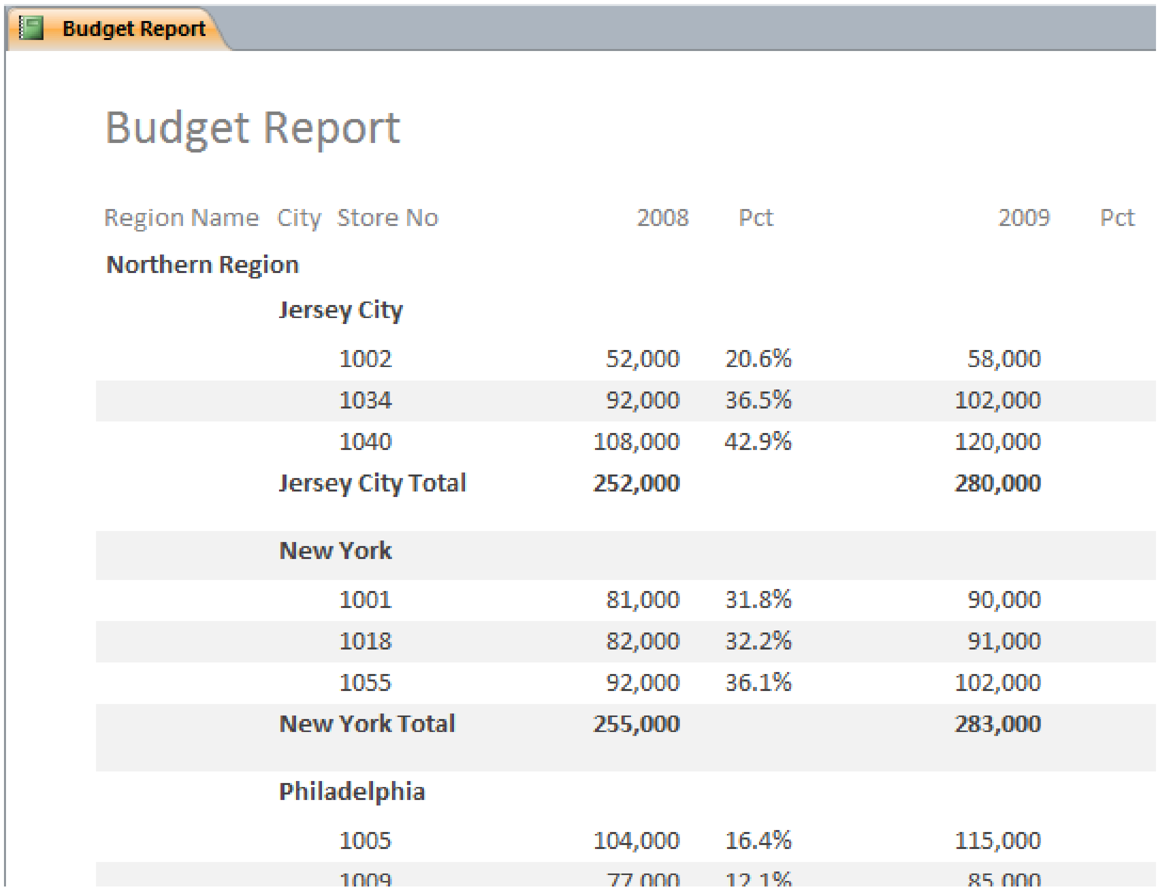
Text15

Name	Text15
Control Source	=Sum([2008])
Format	Standard
Decimal Places	0
Visible	Yes
Text Format	Plain Text
Datasheet Caption	
Width	1"
Height	0.2"
Top	0"
Left	2.2917"
Back Style	Normal

Figure 9.12

This text box in my report was named Text15. **Please note that 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.** This text box will be the denominator in the calculation.

11. *In* **Design View**, *create a text box and place it under the* **2008 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* **=[2008]/[Text15]** *(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 run the report.*



Budget Report

Budget Report

Region Name	City	Store No	2008	Pct	2009	Pct
Northern Region						
	Jersey City					
		1002	52,000	20.6%	58,000	
		1034	92,000	36.5%	102,000	
		1040	108,000	42.9%	120,000	
	Jersey City Total		252,000		280,000	
	New York					
		1001	81,000	31.8%	90,000	
		1018	82,000	32.2%	91,000	
		1055	92,000	36.1%	102,000	
	New York Total		255,000		283,000	
	Philadelphia					
		1005	104,000	16.4%	115,000	
		1009	77,000	12.1%	85,000	

Figure 9.13

You should see these formatted numbers appearing to the right of the 2008 column as in the figure above. If not, you did something wrong and need to correct it.

16. *Make sure the font size and style of the new* **text box** *is the same as the* **2008** *text box.*
17. *Repeat the same process to create the calculations for the* **City**, **Region** *and* **Company** *contributions (the* **Company** *contribution will be* **100%***).*

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.

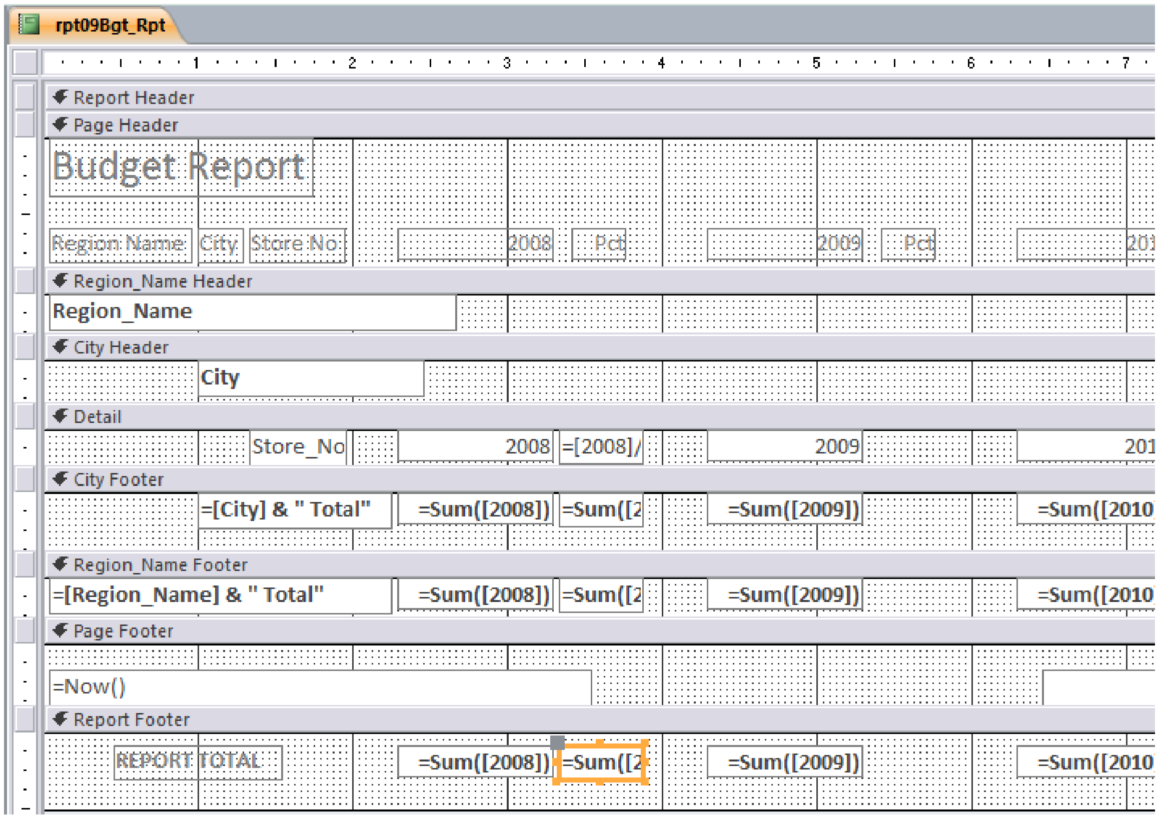
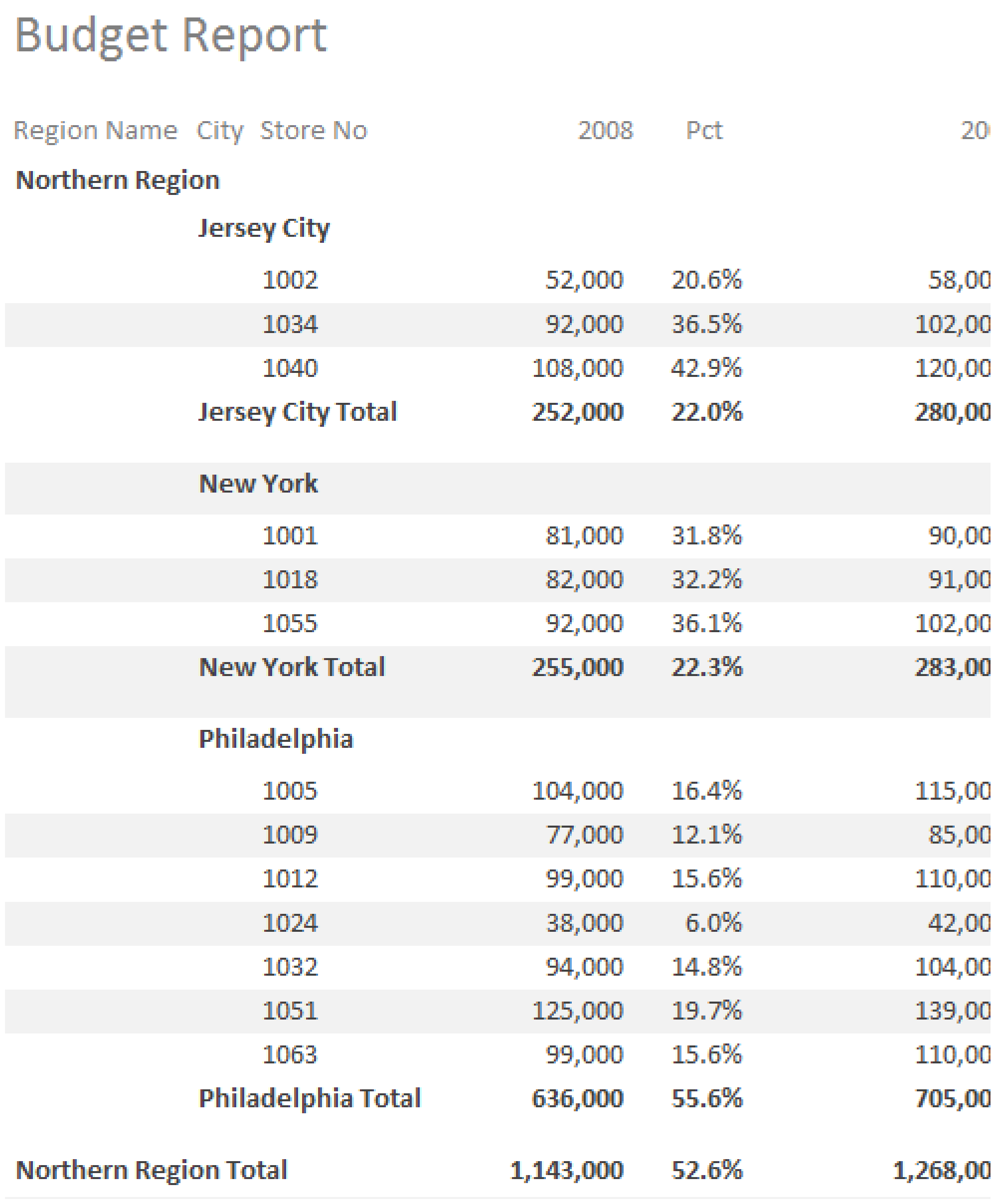


Figure 9.14

In my report, the calculation for the 2008 City Pct was =Sum([2008])/[Text28], the Region Pct was =Sum([2008])/[Text21], and for the Report Total was =Sum([2008])/[Text21]. 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 2008 in your report with these figures:

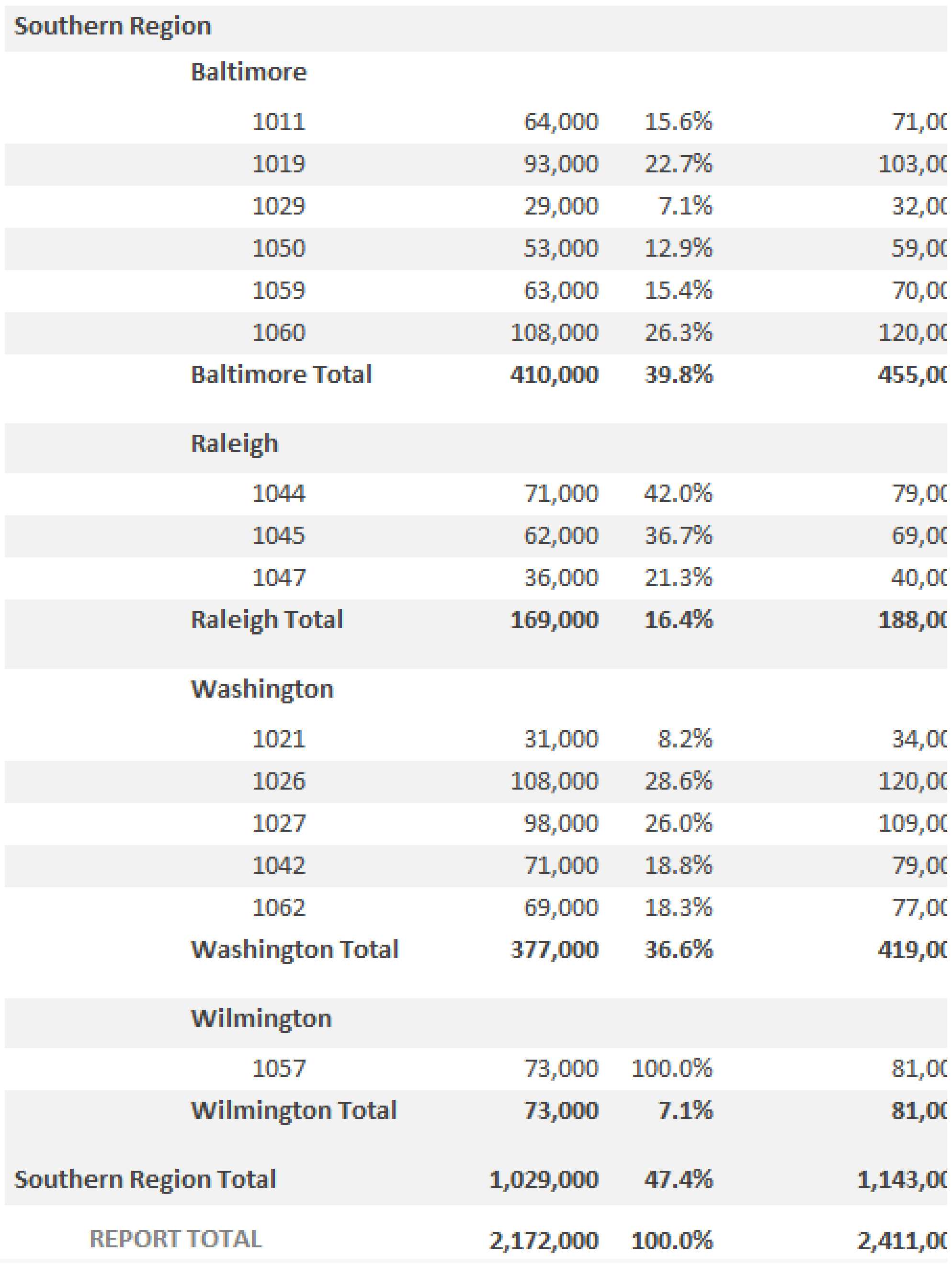


Budget Report

Region Name	City	Store No	2008	Pct	20
Northern Region					
	Jersey City				
		1002	52,000	20.6%	58,00
		1034	92,000	36.5%	102,00
		1040	108,000	42.9%	120,00
	Jersey City Total		**252,000**	**22.0%**	**280,00**
	New York				
		1001	81,000	31.8%	90,00
		1018	82,000	32.2%	91,00
		1055	92,000	36.1%	102,00
	New York Total		**255,000**	**22.3%**	**283,00**
	Philadelphia				
		1005	104,000	16.4%	115,00
		1009	77,000	12.1%	85,00
		1012	99,000	15.6%	110,00
		1024	38,000	6.0%	42,00
		1032	94,000	14.8%	104,00
		1051	125,000	19.7%	139,00
		1063	99,000	15.6%	110,00
	Philadelphia Total		**636,000**	**55.6%**	**705,00**
Northern Region Total			**1,143,000**	**52.6%**	**1,268,00**

Figure 9.15

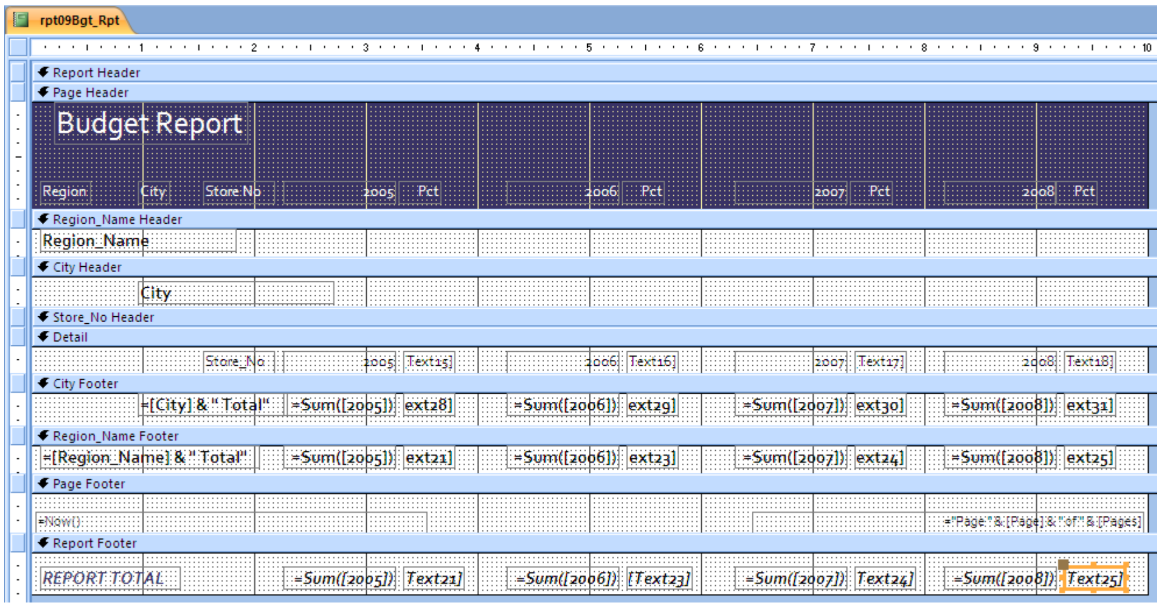
Here are some numbers on the second page of the report.



Southern Region				
	Baltimore			
	1011	64,000	15.6%	71,0
	1019	93,000	22.7%	103,0
	1029	29,000	7.1%	32,0
	1050	53,000	12.9%	59,0
	1059	63,000	15.4%	70,0
	1060	108,000	26.3%	120,0
	Baltimore Total	**410,000**	**39.8%**	**455,0**
	Raleigh			
	1044	71,000	42.0%	79,0
	1045	62,000	36.7%	69,0
	1047	36,000	21.3%	40,0
	Raleigh Total	**169,000**	**16.4%**	**188,0**
	Washington			
	1021	31,000	8.2%	34,0
	1026	108,000	28.6%	120,0
	1027	98,000	26.0%	109,0
	1042	71,000	18.8%	79,0
	1062	69,000	18.3%	77,0
	Washington Total	**377,000**	**36.6%**	**419,0**
	Wilmington			
	1057	73,000	100.0%	81,0
	Wilmington Total	**73,000**	**7.1%**	**81,0**
Southern Region Total		**1,029,000**	**47.4%**	**1,143,0**
REPORT TOTAL		**2,172,000**	**100.0%**	**2,411,0**

Figure 9.16

18. *Repeat the same process to create the calculations for the Years* **2009**, **2010** *and* **2011**.


rpt09Bgt_Rpt

Report Header

Page Header

Budget Report

Region City Store No 2005 Pct 2006 Pct 2007 Pct 2008 Pct

Region_Name Header

Region_Name

City Header

City

Store_No Header

Detail

Store_No 2005 Text15] 2006 Text16] 2007 Text17] 2008 Text18]

City Footer

=[City] & " Total" =Sum([2005]) ext28] =Sum([2006]) ext29] =Sum([2007]) ext30] =Sum([2008]) ext31]

Region_Name Footer

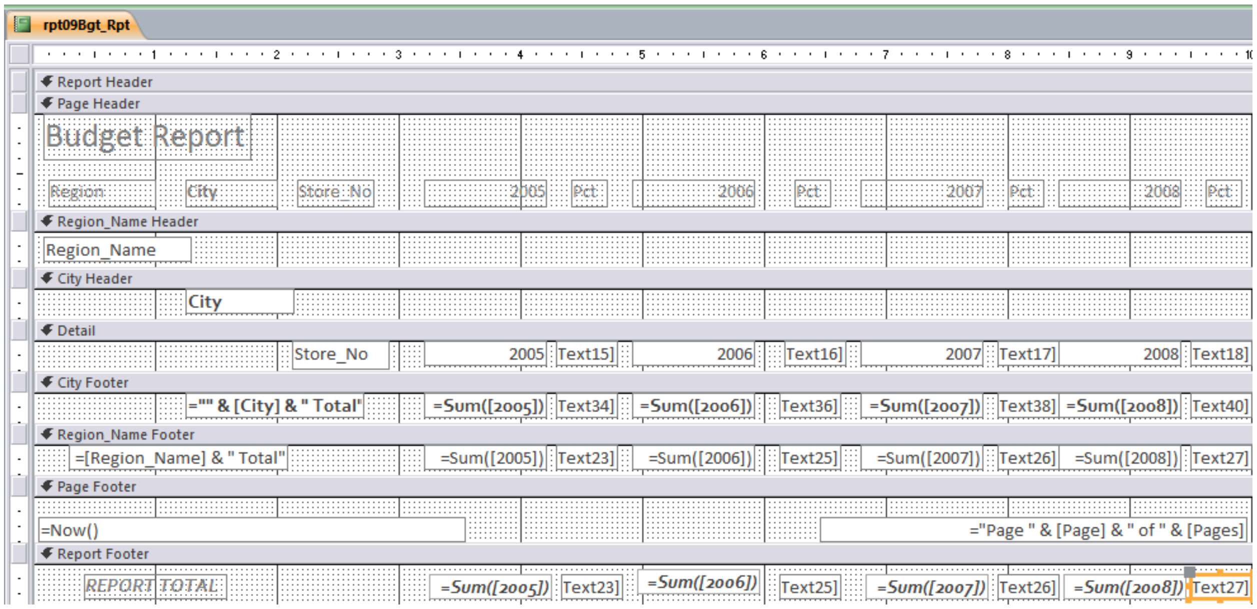
=[Region_Name] & " Total" =Sum([2005]) ext21] =Sum([2006]) ext23] =Sum([2007]) ext24] =Sum([2008]) ext25]

Page Footer

=Now() ="Page " & [Page] & " of " & [Pages]

Report Footer

REPORT TOTAL =Sum([2005]) Text21] =Sum([2006]) [Text23] =Sum([2007]) Text24] =Sum([2008]) Text25]


rpt09Bgt_Rpt

Report Header

Page Header

Budget Report

Region City Store_No 2005 Pct 2006 Pct 2007 Pct 2008 Pct

Region_Name Header

Region_Name

City Header

City

Detail

Store_No 2005 Text15] 2006 Text16] 2007 Text17] 2008 Text18]

City Footer

="" & [City] & " Total" =Sum([2005]) Text34] =Sum([2006]) Text36] =Sum([2007]) Text38] =Sum([2008]) Text40]

Region_Name Footer

=[Region_Name] & " Total" =Sum([2005]) Text23] =Sum([2006]) Text25] =Sum([2007]) Text26] =Sum([2008]) Text27]

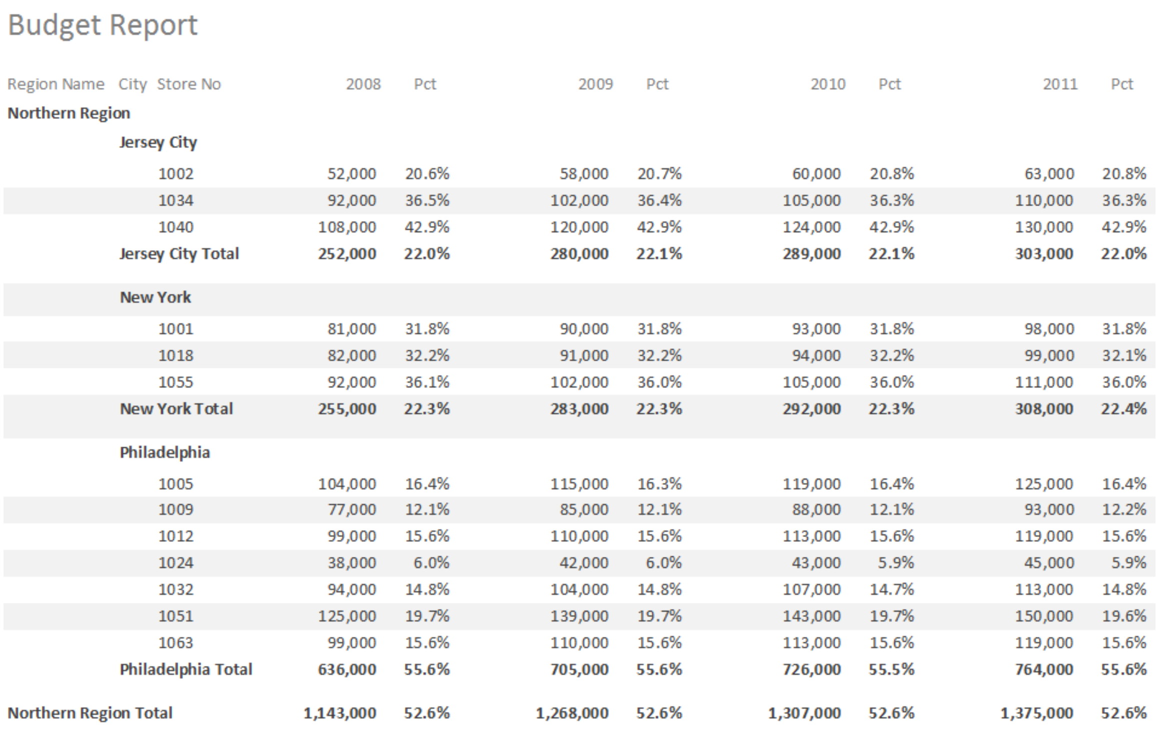
Page Footer

=Now() ="Page " & [Page] & " of " & [Pages]

Report Footer

REPORT TOTAL =Sum([2005]) Text23] =Sum([2006]) Text25] =Sum([2007]) Text26] =Sum([2008]) Text27]

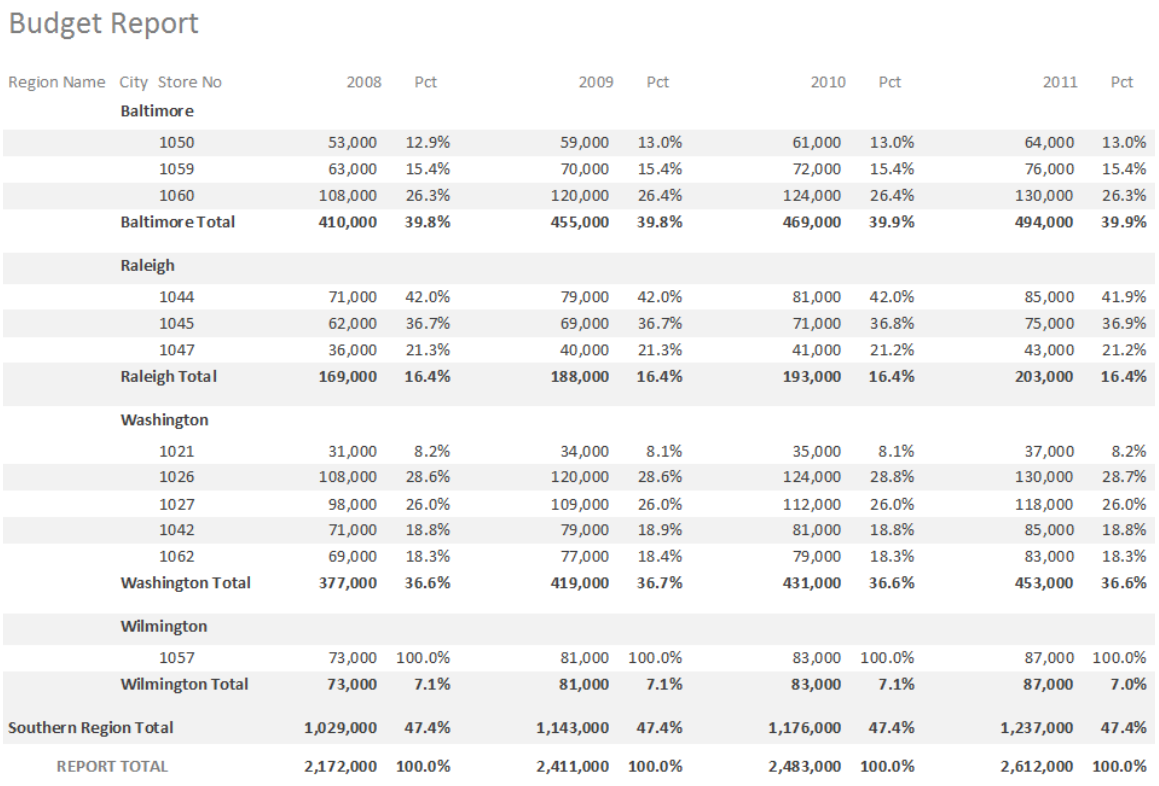
Figure 9.17



Budget Report

Region Name	City	Store No	2008	Pct	2009	Pct	2010	Pct	2011	Pct
Northern Region										
	Jersey City									
		1002	52,000	20.6%	58,000	20.7%	60,000	20.8%	63,000	20.8%
		1034	92,000	36.5%	102,000	36.4%	105,000	36.3%	110,000	36.3%
		1040	108,000	42.9%	120,000	42.9%	124,000	42.9%	130,000	42.9%
	Jersey City Total		**252,000**	**22.0%**	**280,000**	**22.1%**	**289,000**	**22.1%**	**303,000**	**22.0%**
	New York									
		1001	81,000	31.8%	90,000	31.8%	93,000	31.8%	98,000	31.8%
		1018	82,000	32.2%	91,000	32.2%	94,000	32.2%	99,000	32.1%
		1055	92,000	36.1%	102,000	36.0%	105,000	36.0%	111,000	36.0%
	New York Total		**255,000**	**22.3%**	**283,000**	**22.3%**	**292,000**	**22.3%**	**308,000**	**22.4%**
	Philadelphia									
		1005	104,000	16.4%	115,000	16.3%	119,000	16.4%	125,000	16.4%
		1009	77,000	12.1%	85,000	12.1%	88,000	12.1%	93,000	12.2%
		1012	99,000	15.6%	110,000	15.6%	113,000	15.6%	119,000	15.6%
		1024	38,000	6.0%	42,000	6.0%	43,000	5.9%	45,000	5.9%
		1032	94,000	14.8%	104,000	14.8%	107,000	14.7%	113,000	14.8%
		1051	125,000	19.7%	139,000	19.7%	143,000	19.7%	150,000	19.6%
		1063	99,000	15.6%	110,000	15.6%	113,000	15.6%	119,000	15.6%
	Philadelphia Total		**636,000**	**55.6%**	**705,000**	**55.6%**	**726,000**	**55.5%**	**764,000**	**55.6%**
Northern Region Total			**1,143,000**	**52.6%**	**1,268,000**	**52.6%**	**1,307,000**	**52.6%**	**1,375,000**	**52.6%**

Figure 9.18



Budget Report

Region Name	City	Store No	2008	Pct	2009	Pct	2010	Pct	2011	Pct
	Baltimore									
		1050	53,000	12.9%	59,000	13.0%	61,000	13.0%	64,000	13.0%
		1059	63,000	15.4%	70,000	15.4%	72,000	15.4%	76,000	15.4%
		1060	108,000	26.3%	120,000	26.4%	124,000	26.4%	130,000	26.3%
	Baltimore Total		**410,000**	**39.8%**	**455,000**	**39.8%**	**469,000**	**39.9%**	**494,000**	**39.9%**
	Raleigh									
		1044	71,000	42.0%	79,000	42.0%	81,000	42.0%	85,000	41.9%
		1045	62,000	36.7%	69,000	36.7%	71,000	36.8%	75,000	36.9%
		1047	36,000	21.3%	40,000	21.3%	41,000	21.2%	43,000	21.2%
	Raleigh Total		**169,000**	**16.4%**	**188,000**	**16.4%**	**193,000**	**16.4%**	**203,000**	**16.4%**
	Washington									
		1021	31,000	8.2%	34,000	8.1%	35,000	8.1%	37,000	8.2%
		1026	108,000	28.6%	120,000	28.6%	124,000	28.8%	130,000	28.7%
		1027	98,000	26.0%	109,000	26.0%	112,000	26.0%	118,000	26.0%
		1042	71,000	18.8%	79,000	18.9%	81,000	18.8%	85,000	18.8%
		1062	69,000	18.3%	77,000	18.4%	79,000	18.3%	83,000	18.3%
	Washington Total		**377,000**	**36.6%**	**419,000**	**36.7%**	**431,000**	**36.6%**	**453,000**	**36.6%**
	Wilmington									
		1057	73,000	100.0%	81,000	100.0%	83,000	100.0%	87,000	100.0%
	Wilmington Total		**73,000**	**7.1%**	**81,000**	**7.1%**	**83,000**	**7.1%**	**87,000**	**7.0%**
Southern Region Total			**1,029,000**	**47.4%**	**1,143,000**	**47.4%**	**1,176,000**	**47.4%**	**1,237,000**	**47.4%**
REPORT TOTAL			**2,172,000**	**100.0%**	**2,411,000**	**100.0%**	**2,483,000**	**100.0%**	**2,612,000**	**100.0%**

Figure 9.19

19. Save and close the report.

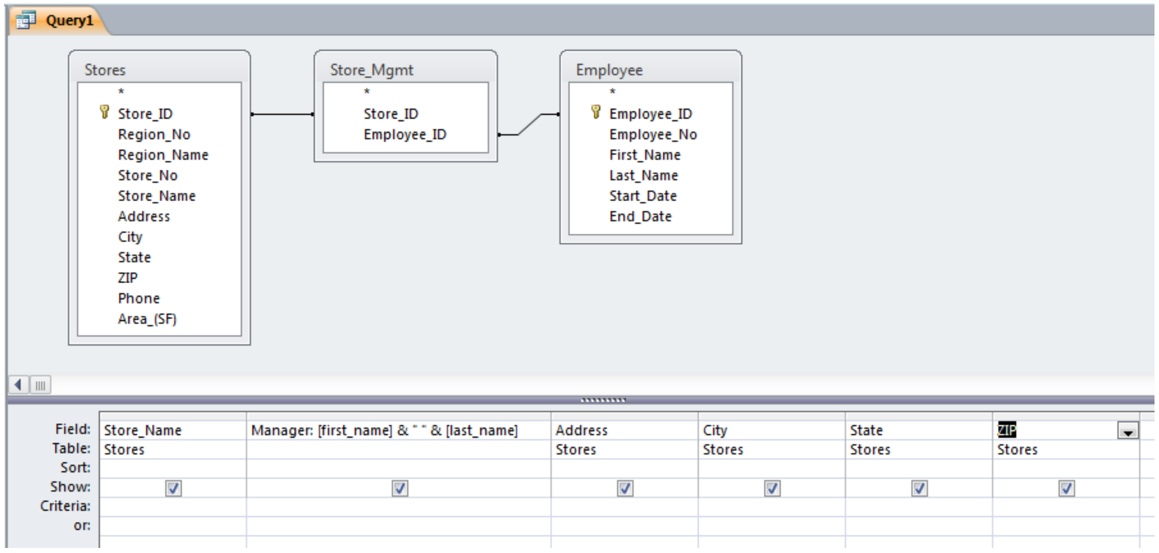
Wow! You did it! That wasn't too hard, was it? I guess it was, but you're a better person for going through all of that. I promise that if you made a few mistakes, all the better. I assume you had to play around with some of the formatting or placing of the text boxes and labels to make it come out just right. In the reporting world, there are a billion and one ways to do these kinds of report. 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**.


Query1

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

Store_Mgmt: *, Store_ID, Employee_ID

Employee: *, Employee_ID, Employee_No, First_Name, Last_Name, Start_Date, End_Date

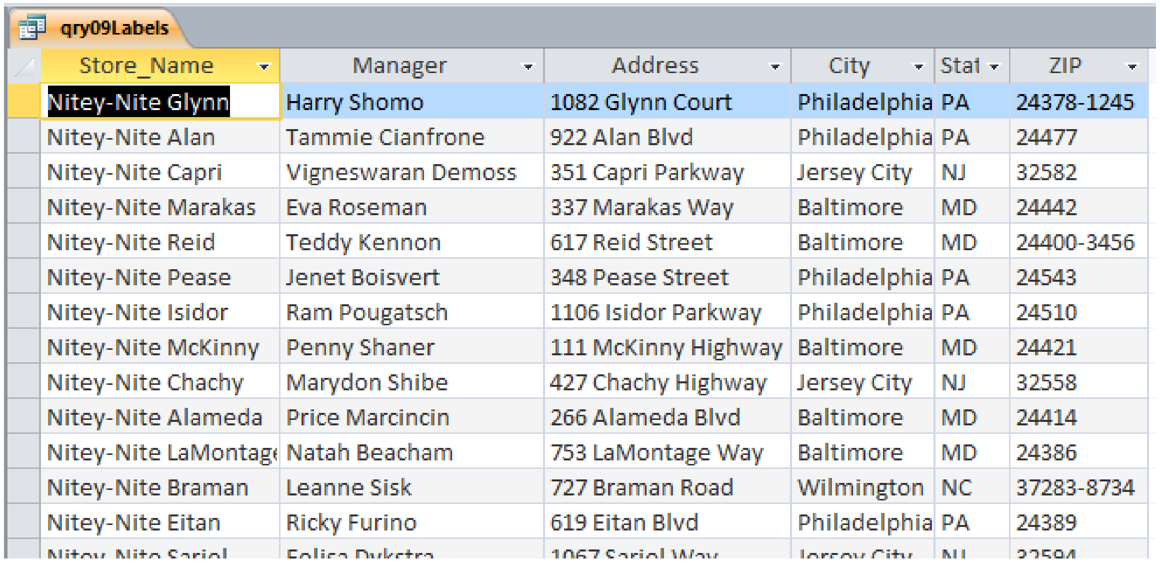
Field:	Store_Name	Manager: [first_name] & " " & [last_name]	Address	City	State	ZIP
Table:	Stores		Stores	Stores	Stores	Stores
Sort:						
Show:	✓	✓	✓	✓	✓	✓
Criteria:						
or:						

Figure 9.20

4. *Save the query as* **qry09Labels** *and run it.*

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 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

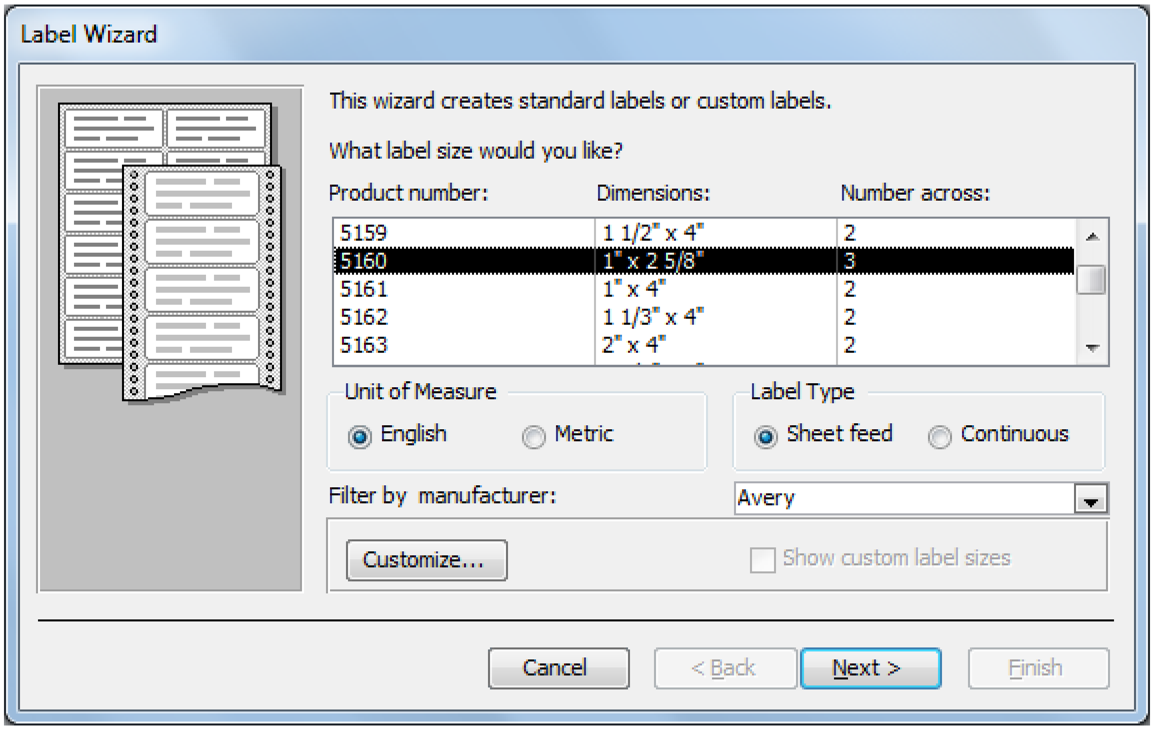
Store_Name	Manager	Address	City	Stat	ZIP
Nitey-Nite Glynn	Harry Shomo	1082 Glynn Court	Philadelphia	PA	24378-1245
Nitey-Nite Alan	Tammie Cianfrone	922 Alan Blvd	Philadelphia	PA	24477
Nitey-Nite Capri	Vigneswaran Demoss	351 Capri Parkway	Jersey City	NJ	32582
Nitey-Nite Marakas	Eva Roseman	337 Marakas 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
Nitey-Nite LaMontag	Natah Beacham	753 LaMontage Way	Baltimore	MD	24386
Nitey-Nite Braman	Leanne Sisk	727 Braman Road	Wilmington	NC	37283-8734
Nitey-Nite Eitan	Ricky Furino	619 Eitan Blvd	Philadelphia	PA	24389
Nitey-Nite Sariol	Felica Dykstra	1067 Sariol Way	Jersey City	NJ	32594

Figure 9.21

5. *Close the query.*
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 Labels Wizard opens. You can use just about any type of label manufacturer and the Product number will show up here.

1. *In the first screen of the* **Label Wizard**, *make sure* **Filter by manufacturer** *is set to* **Avery** *(to use* **Avery** *labels), scroll to* **5160**, *select it from the list of products and click* **Next >**.


Label Wizard

This wizard creates standard labels or custom labels.

What label size would you like?

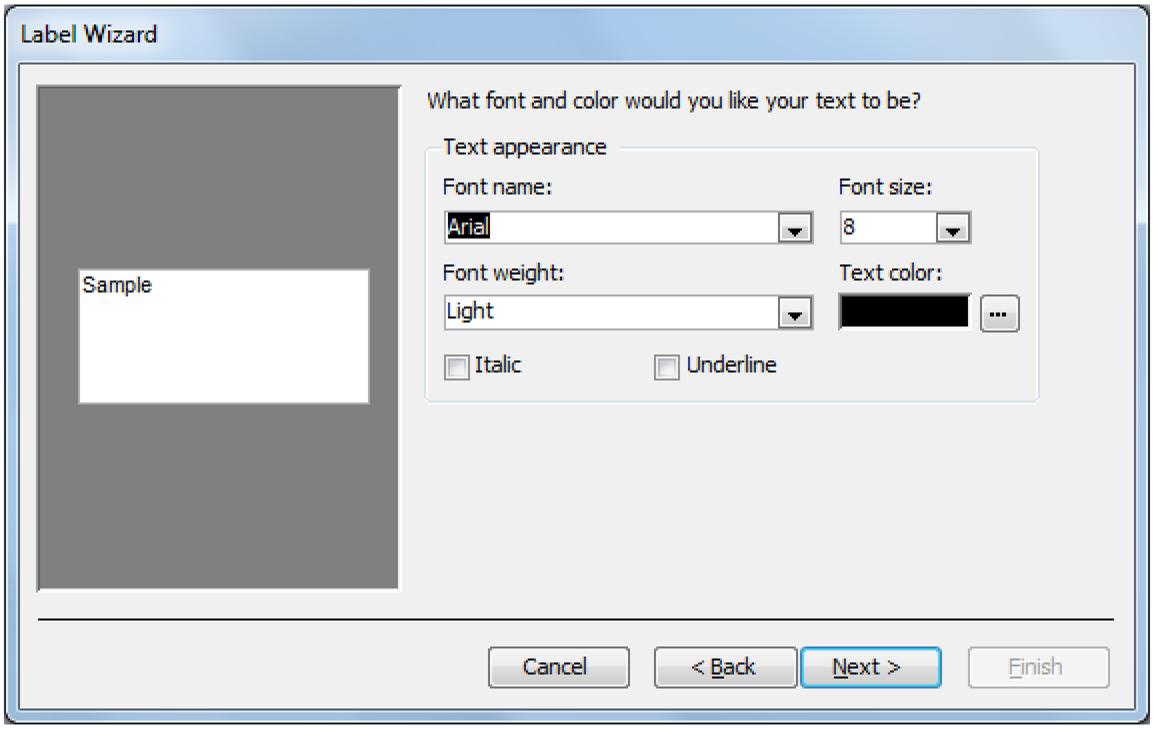
Product number:	Dimensions:	Number across:
5159	1 1/2" x 4"	2
5160	1" x 2 5/8"	3
5161	1" x 4"	2
5162	1 1/3" x 4"	2
5163	2" x 4"	2

Unit of Measure: [✓ English] Metric

Label Type: [✓ Sheet feed] Continuous

Filter by manufacturer: Avery

Figure 9.22


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:

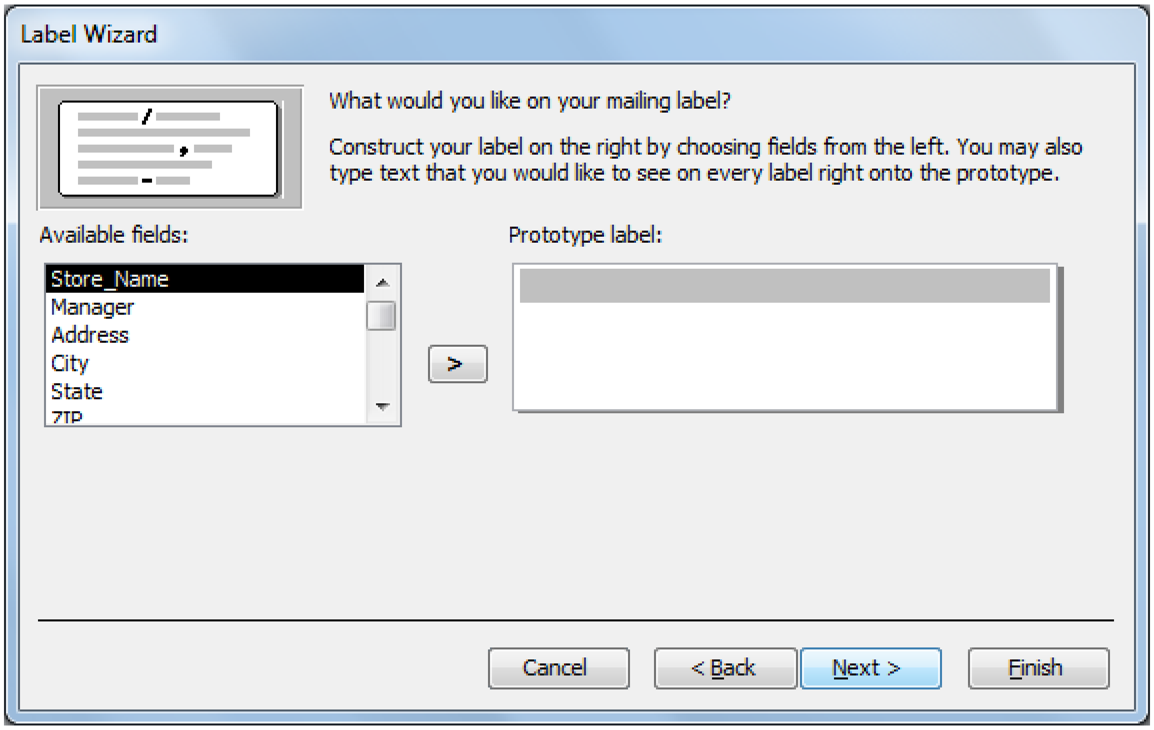
Italic [ ] Underline [ ]

Sample

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.

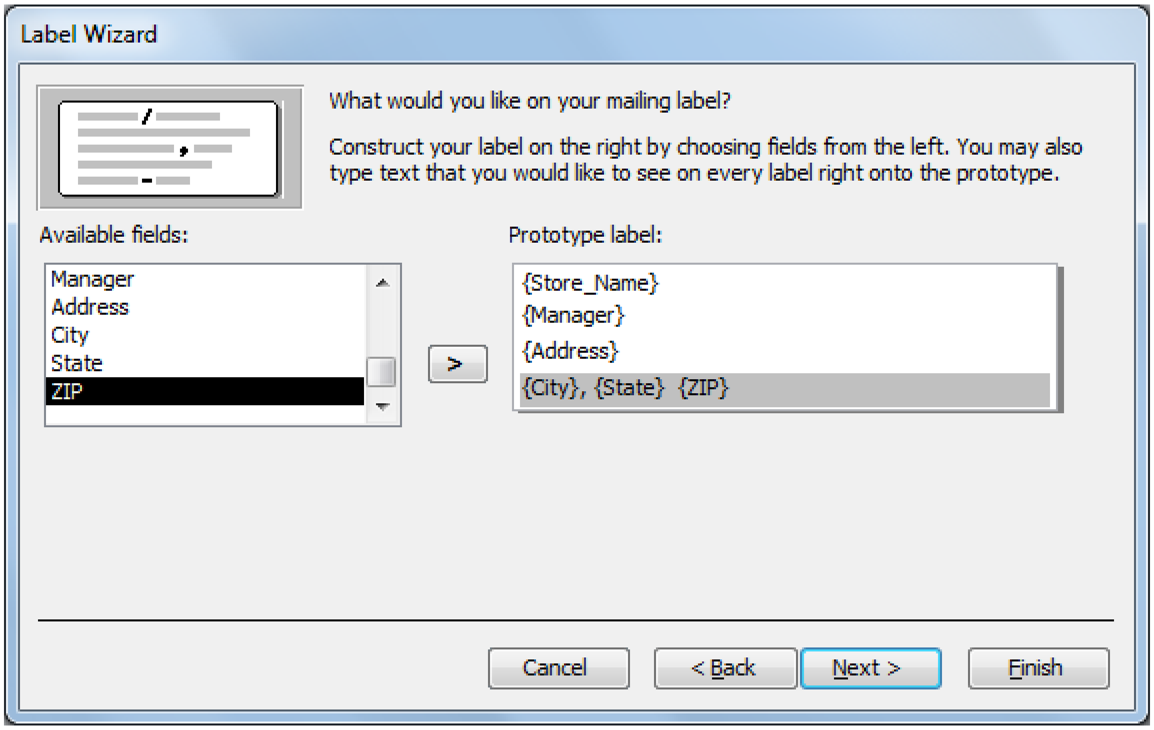
Available fields:

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:

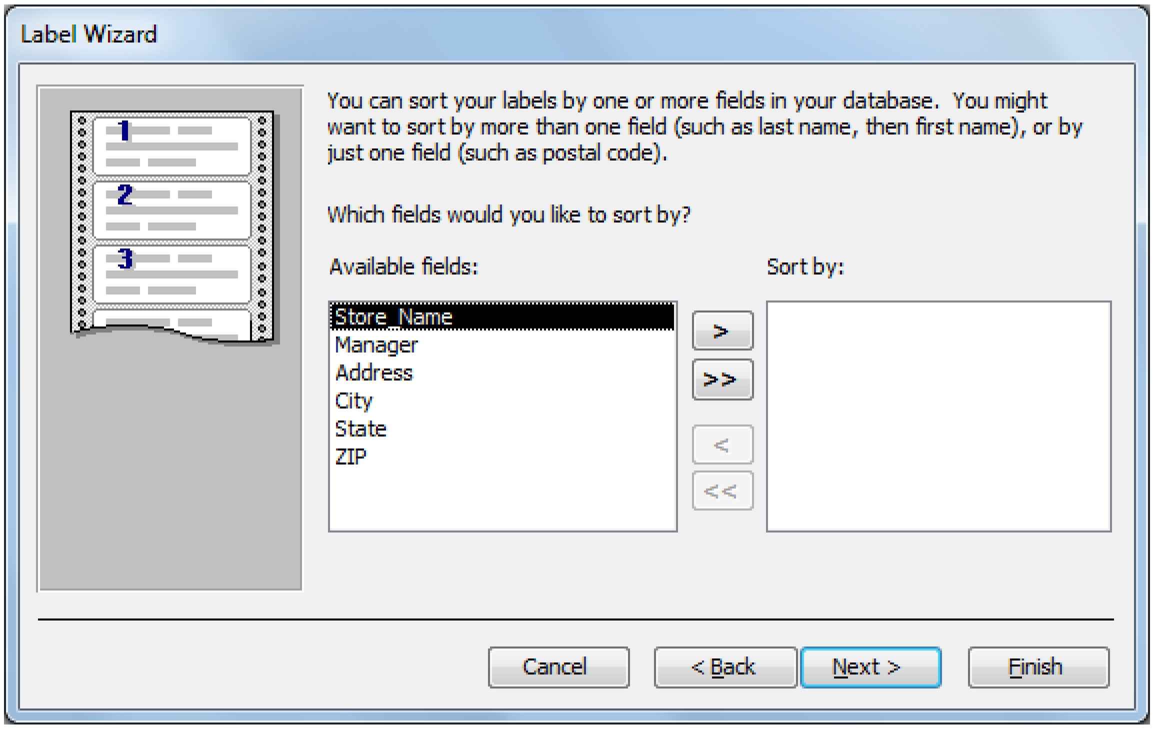
Manager
Address
City
State
ZIP

Prototype label:

{Store_Name}
{Manager}
{Address}
{City}, {State} {ZIP}

Figure 9.25

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



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?

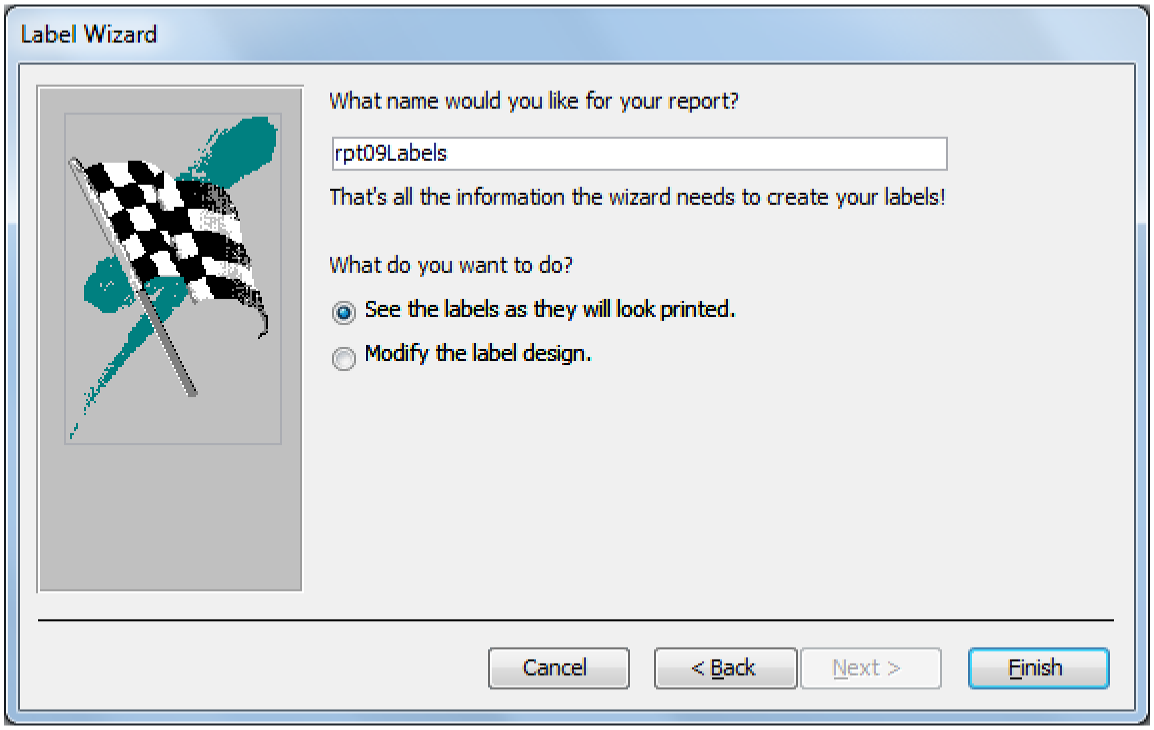
Available fields:

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**.



What name would you like for your report?

rpt09Labels

That's all the information the wizard needs to create your labels!

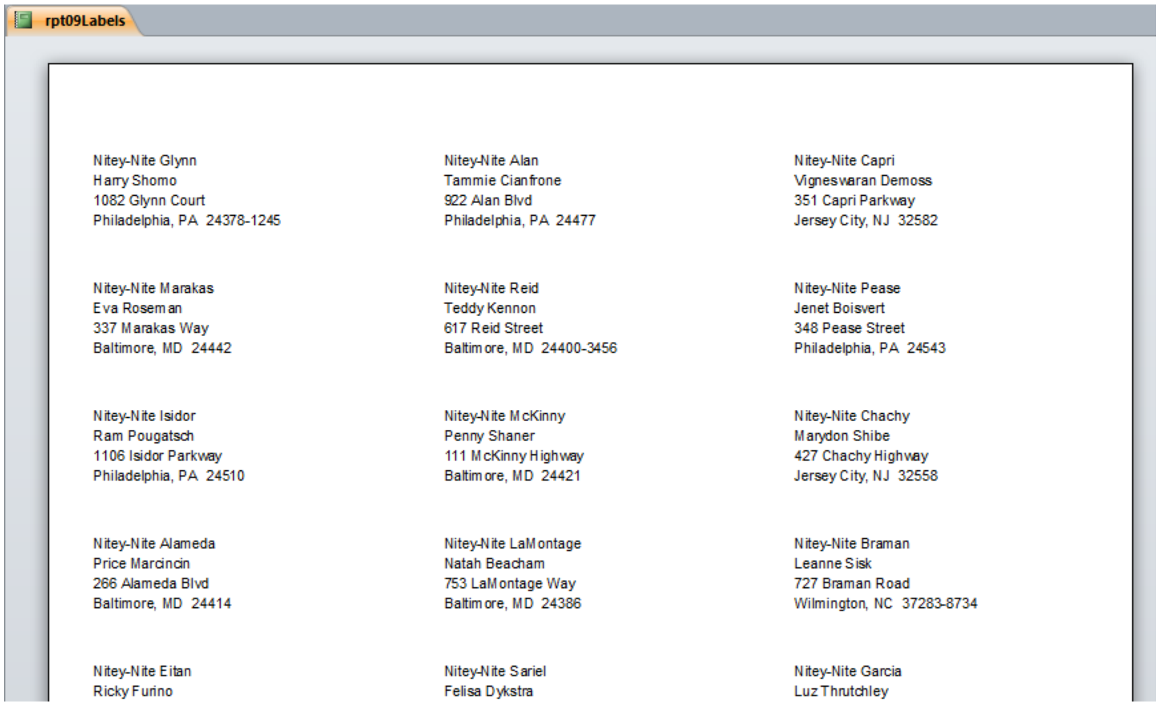
What do you want to do?

[✓ 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                 Nitey-Nite Alan                  Nitey-Nite Capri
Harry Shomo                      Tammie Cianfrone                 Vigneswaran Demoss
1082 Glynn Court                 922 Alan Blvd                    351 Capri Parkway
Philadelphia, PA  24378-1245     Philadelphia, PA  24477          Jersey City, NJ  32582

Nitey-Nite Marakas               Nitey-Nite Reid                  Nitey-Nite Pease
Eva Roseman                      Teddy Kennon                     Jenet Boisvert
337 Marakas Way                  617 Reid Street                  348 Pease Street
Baltimore, MD  24442             Baltimore, MD  24400-3456        Philadelphia, PA  24543

Nitey-Nite Isidor                Nitey-Nite McKinny               Nitey-Nite Chachy
Ram Pougatsch                    Penny Shaner                     Marydon Shibe
1106 Isidor Parkway              111 McKinny Highway              427 Chachy Highway
Philadelphia, PA  24510          Baltimore, MD  24421             Jersey City, NJ  32558

Nitey-Nite Alameda               Nitey-Nite LaMontage             Nitey-Nite Braman
Price Marcincin                  Natah Beacham                    Leanne Sisk
266 Alameda Blvd                 753 LaMontage Way                727 Braman Road
Baltimore, MD  24414             Baltimore, MD  24386             Wilmington, NC  37283-8734

Nitey-Nite Eitan                 Nitey-Nite Sariel                Nitey-Nite Garcia
Ricky Furino                     Felisa Dykstra                   Luz Thrutchley
```

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 the report.

> ***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 2010 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, the query that generated it, and use 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. You learned how to work with the Repeat Section to make section headers repeat at page breaks. You calculated 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, log in and take the exam. Make sure that you take the exam on the same computer that you completed the sample files on, as some of the questions on the exam may refer to some of the completed examples.

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.

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, and 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** are just that – data to test with. 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. 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 development 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 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 the 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_2010 I have provided when you download the Nitey_Nite_2010 database. You will find that table in the C:\ExcelCEO\Access2010 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 host your SQL Server 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* **Start**, *then click on* **Control Panel** *(or just navigate to* **Control Panel**).

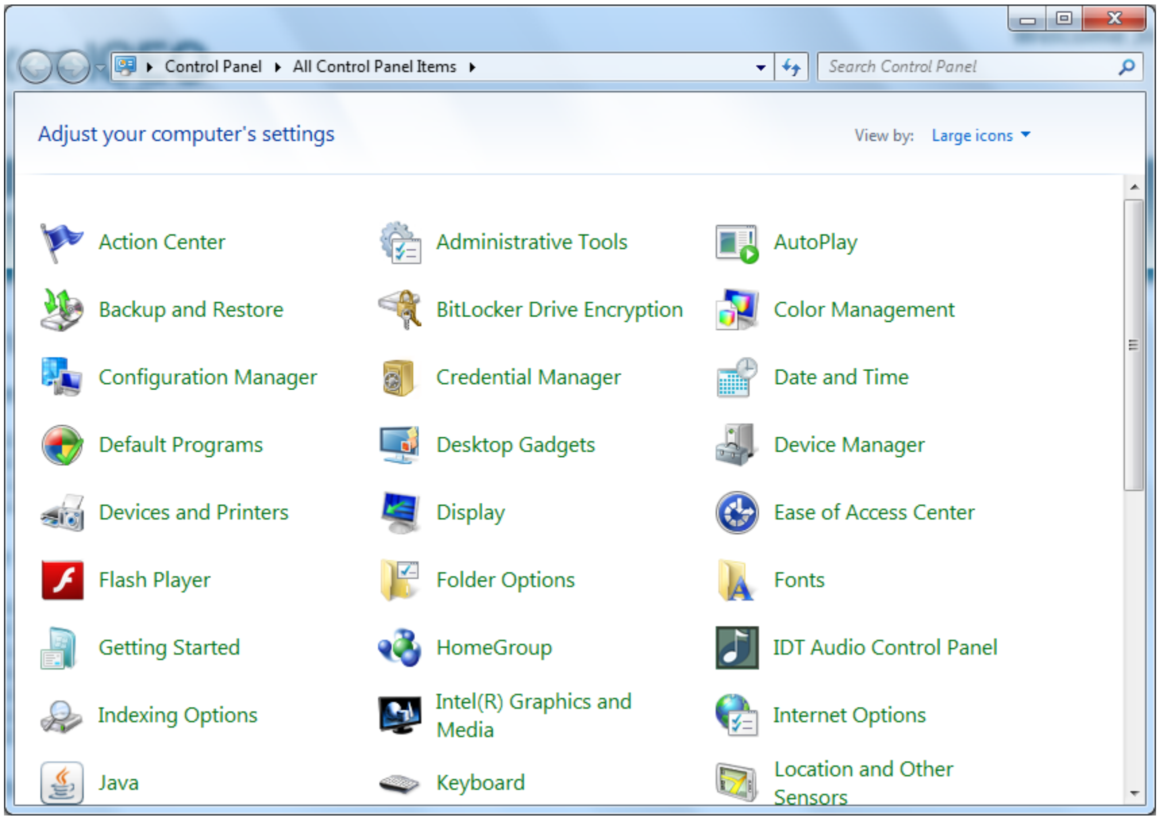


Figure 10.1

Your screen should look similar to Figure 10.1. If not, you may have to click on "*Switch to Classic view*" and/or click on View, Icons from the Main Menu Bar. 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**.

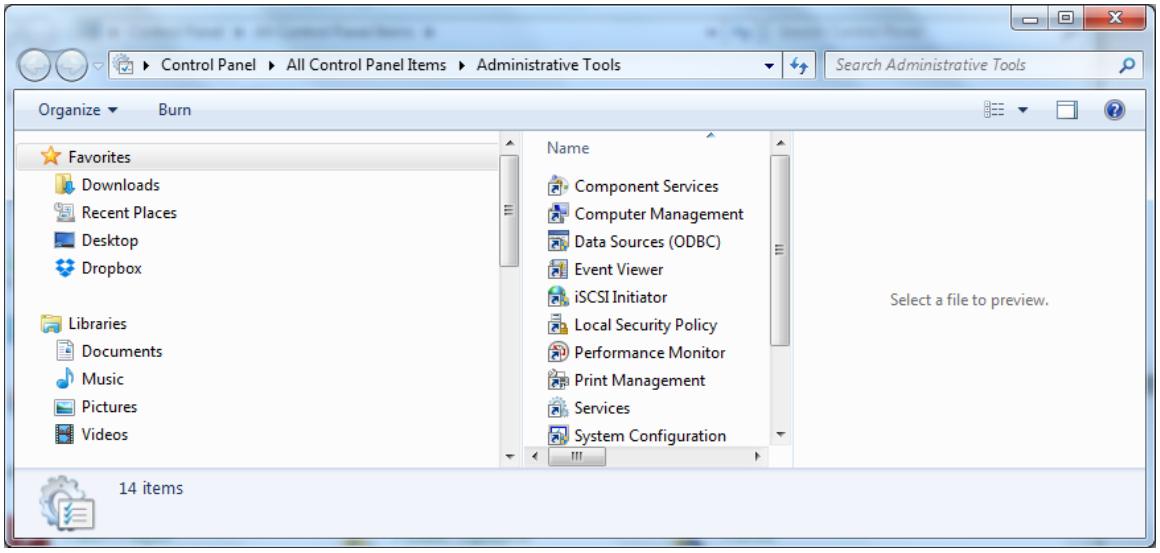
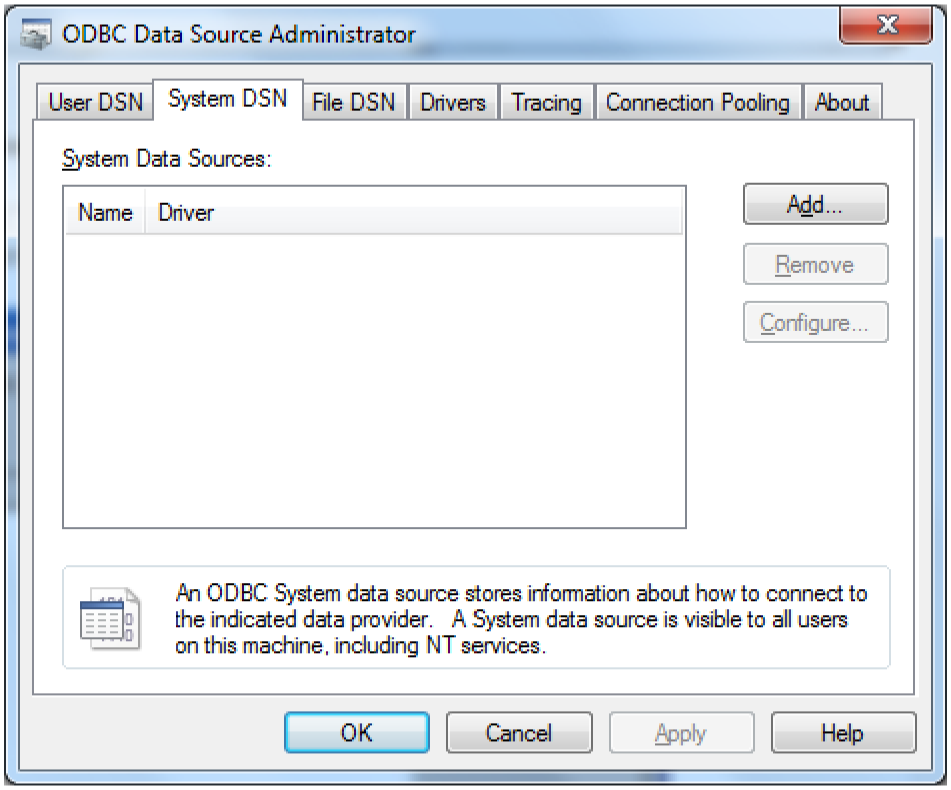


Figure 10.2

3. *Open* **Data Sources (ODBC)** *(this may also display as (ODBC) Data Sources).*

ODBC stands for **Open Database Connectivity**. It is the tool that allows you to connect to a host of different database types.

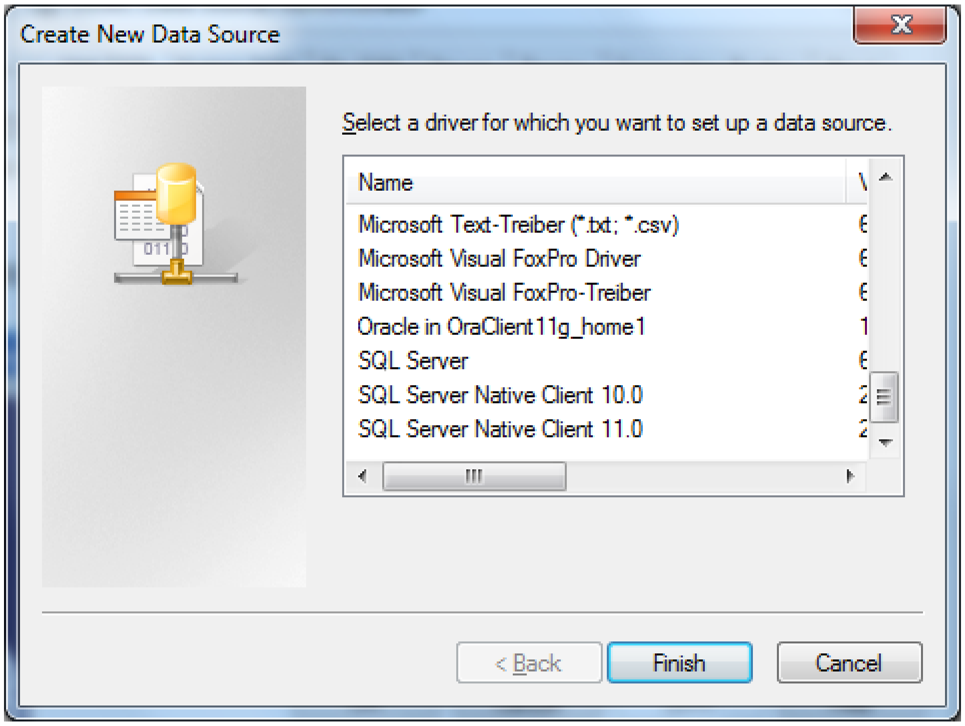


An ODBC System data source stores information about how to connect to the indicated data provider. A System data source is visible to all users on this machine, including NT services.

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.*

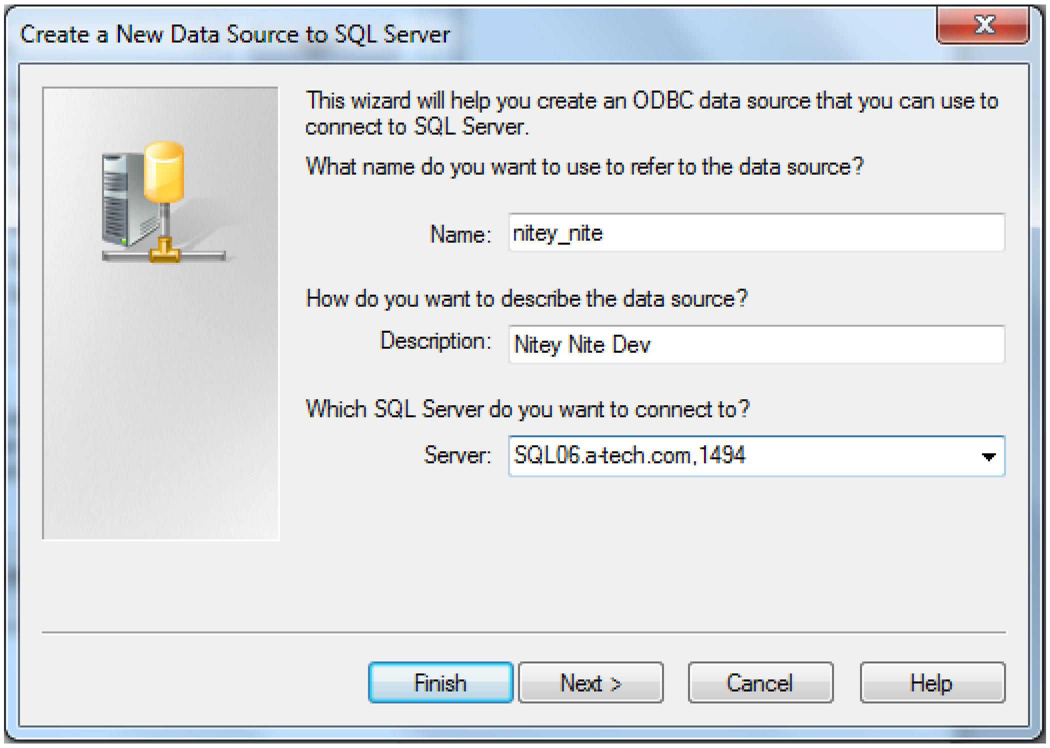


Select a driver for which you want to set up a data source.

Name	V
Microsoft Text-Treiber (*.txt; *.csv)	6
Microsoft Visual FoxPro Driver	6
Microsoft Visual FoxPro-Treiber	6
Oracle in OraClient11g_home1	1
SQL Server	6
SQL Server Native Client 10.0	2
SQL Server Native Client 11.0	2

Figure 10.4

5. *In the* **Create New Data Source** *dialog box, scroll down and click on the* **SQL Server** *option and click* **Finish**.


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: SQL06.a-tech.com,1494

Figure 10.5

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**.

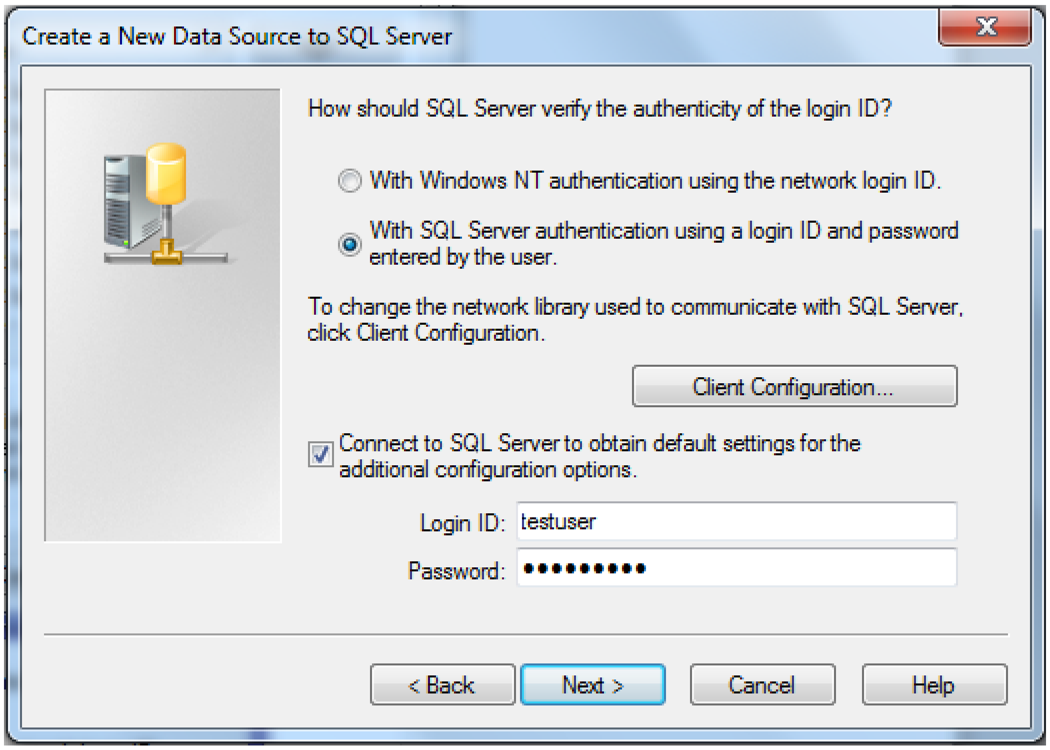
6. *Type in the* **name**, **description** *and* **server** *for the* **DSN** *exactly as it appears in* **Figure 10.5**.

   ***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 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.

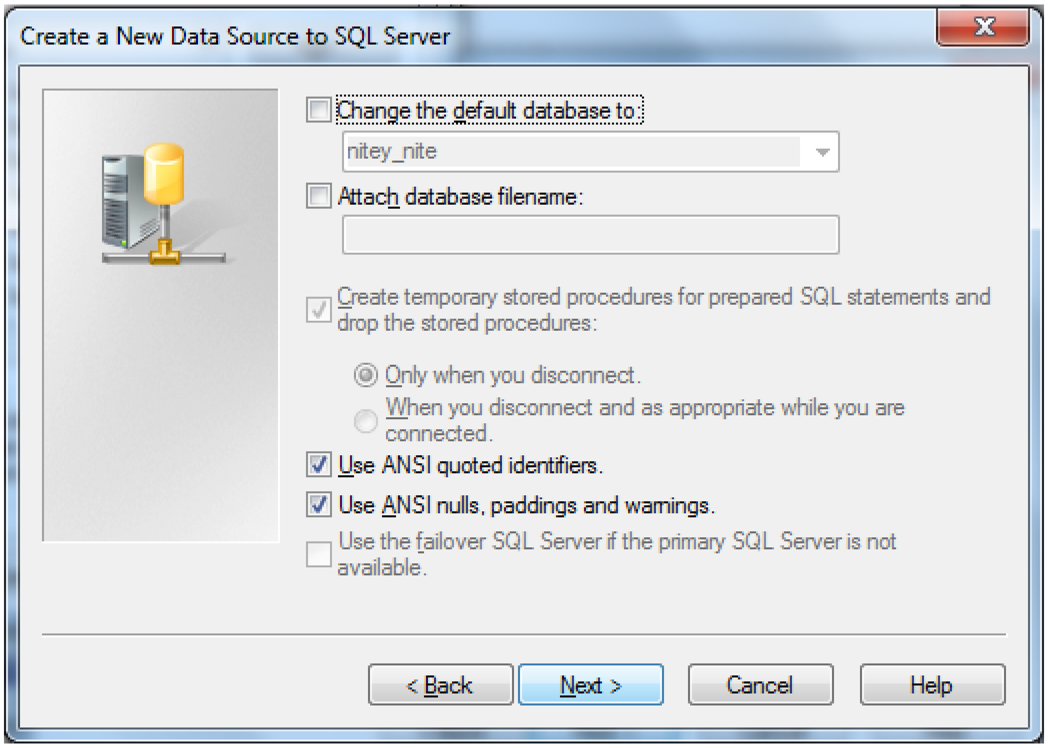
☑ Connect to SQL Server to obtain default settings for the additional configuration options.

Login ID: testuser

Password: ••••••••••

Figure 10.6

11. *Click* **Next >**.

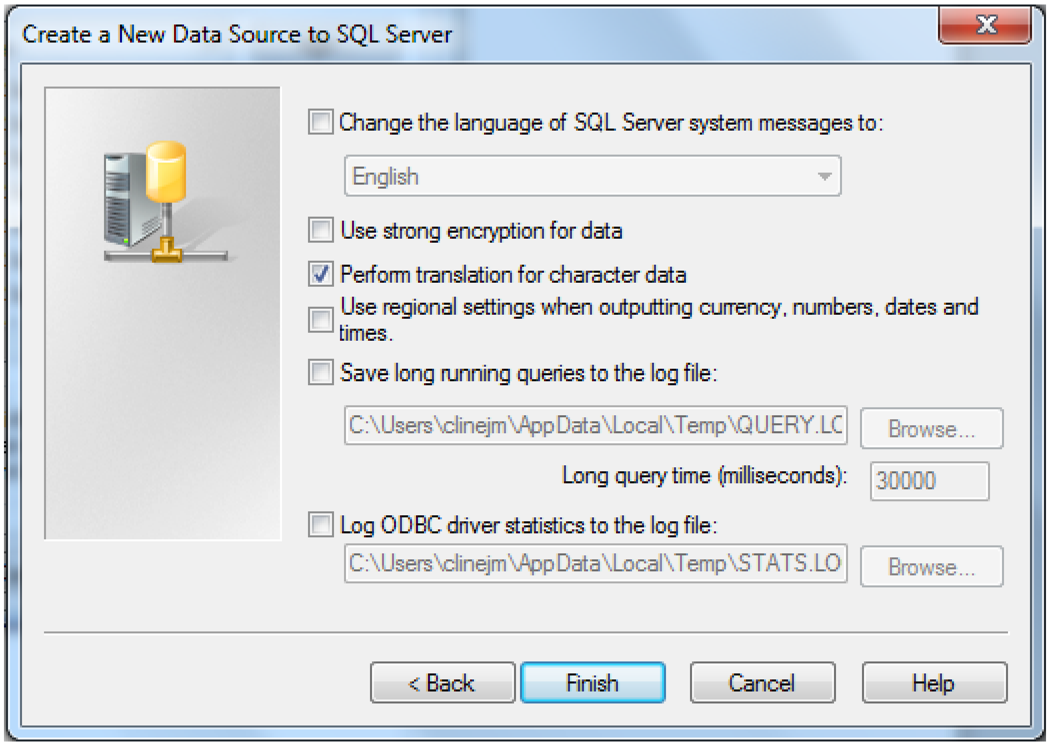

Create a New Data Source to SQL Server

[ ] Change the default database to:
nitey_nite
[ ] Attach database filename:

[✓] Create temporary stored procedures for prepared SQL statements and drop the stored procedures:
(●) Only when you disconnect.
( ) When you disconnect and as appropriate while you are connected.
[✓] 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

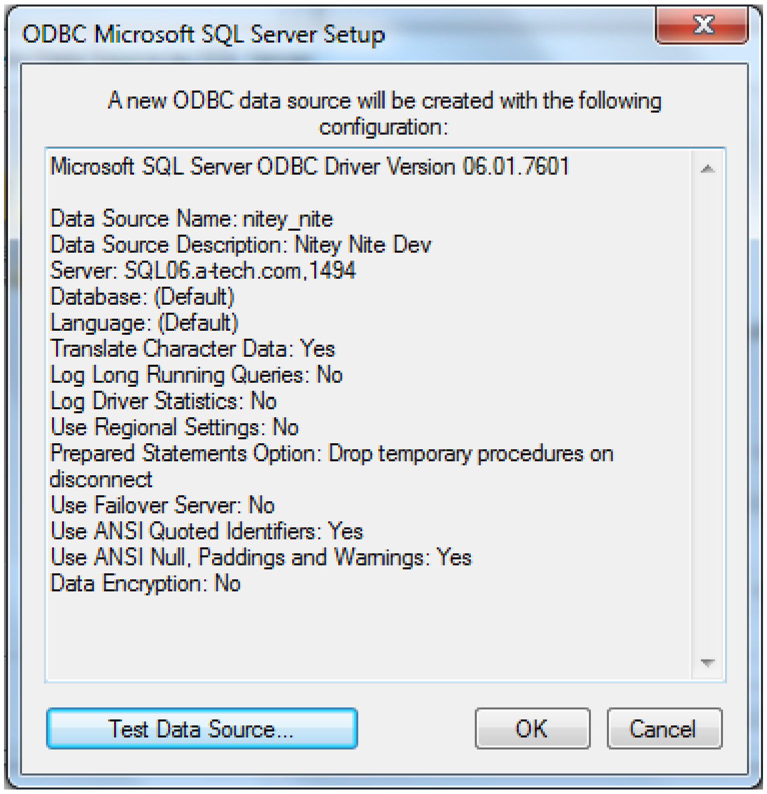
12. Make sure the default database is set to **nitey_nite** *and click* **Next >***.*


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\clinejm\AppData\Local\Temp\QUERY.LC
Long query time (milliseconds): 30000
[ ] Log ODBC driver statistics to the log file:
C:\Users\clinejm\AppData\Local\Temp\STATS.LO

Figure 10.8

13. Click **Finish***.*



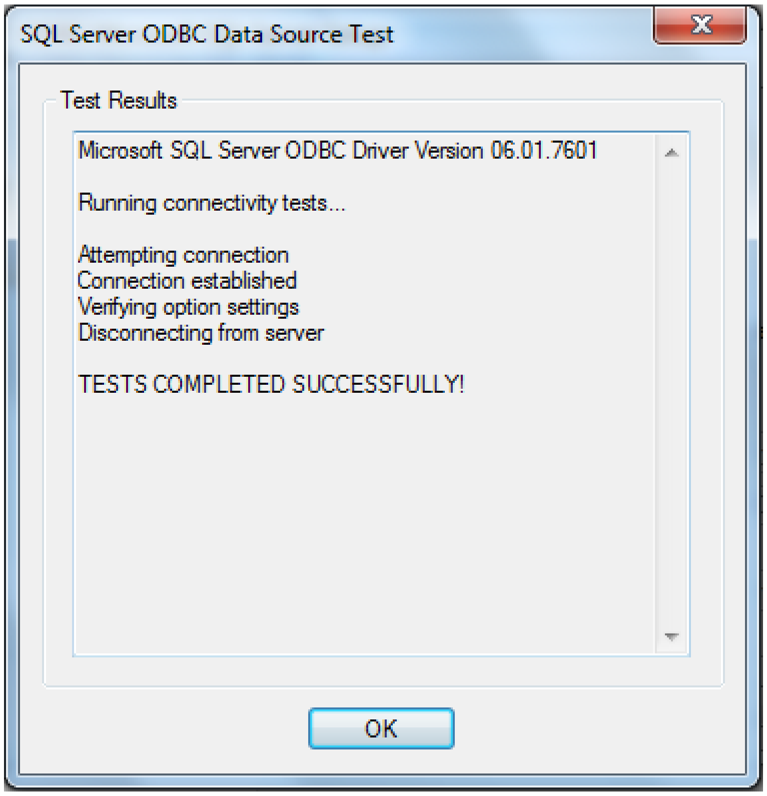
A new ODBC data source will be created with the following configuration:

```
Microsoft SQL Server ODBC Driver Version 06.01.7601

Data Source Name: nitey_nite
Data Source Description: Nitey Nite Dev
Server: SQL06.a-tech.com,1494
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 06.01.7601

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**.

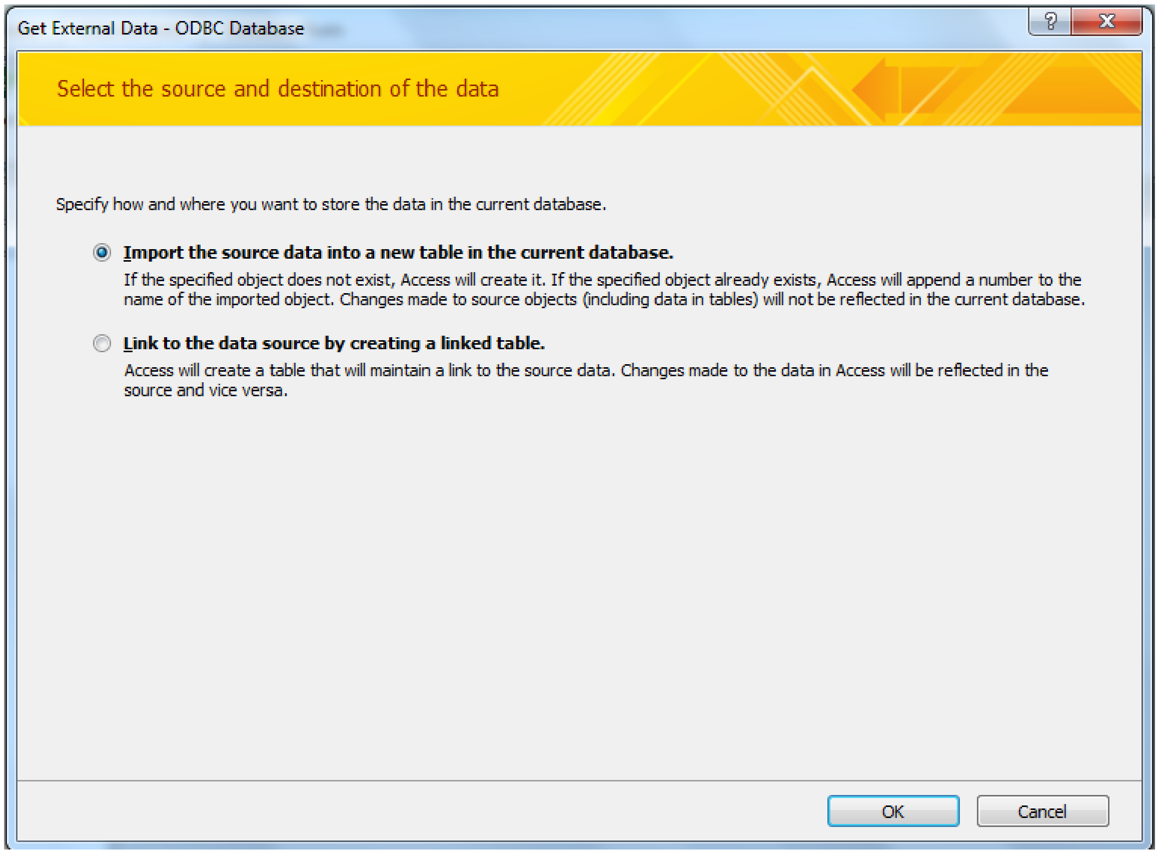
Now that you have the DSN created, you can connect to the Nitey-Nite SQL Server database. This server is physically located somewhere along the East Coast, but when you create the connection, it connects your computer to the server and allows you to manipulate the data and perform calculations as if 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 new* **Access database** *and save it as* **C:\ExcelCEO\Access 2010\Inc_Stmt.accdb**.
2. *Close out of the new table that automatically opens when you create a new* **Access 2010** *database.*

3. *Click on the* **External Data** *tab and click on the* **ODBC Database** *icon in the* **Import & Link** *group.*


Get External Data - ODBC Database
Select the source and destination of the data
OK
Cancel

Specify how and where you want to store the data in the current database.

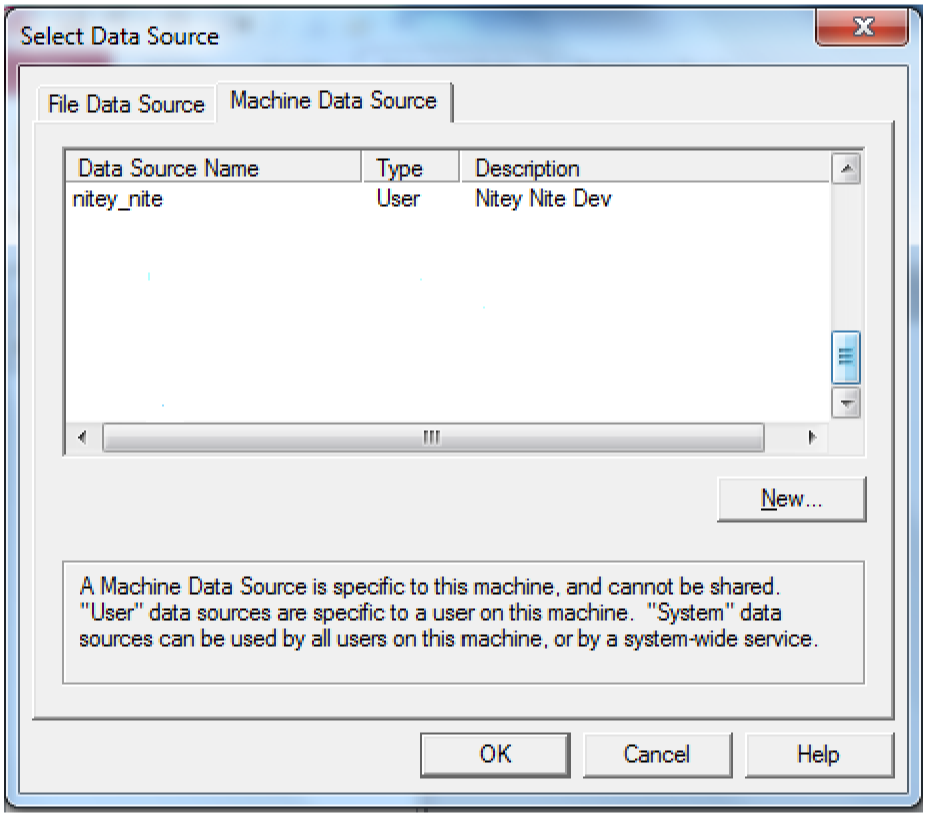
(●) **Import the source data into a new table in the current database.**
If the specified object does not exist, Access will create it. If the specified object already exists, Access will append a number to the name of the imported object. Changes made to source objects (including data in tables) will 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. Changes made to the data in Access will be reflected in the source and vice versa.

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* **Get External Data – ODBC Database** *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.*



Select Data Source

File Data Source | Machine Data Source

Data Source Name	Type	Description
nitey_nite	User	Nitey Nite Dev

New...

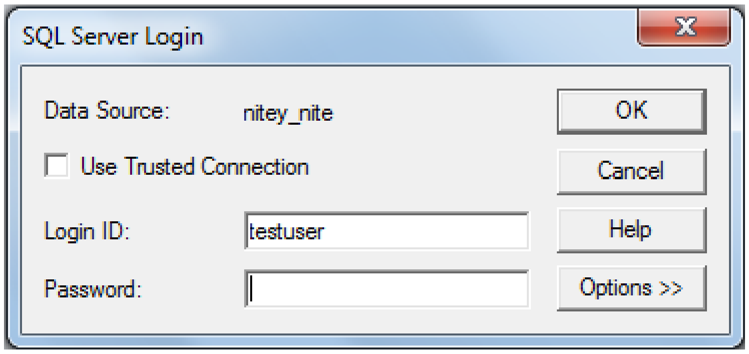
A Machine Data Source is specific to this machine, and cannot be shared. "User" data sources are specific to a user on this machine. "System" data sources can be used by all users on this machine, or by a system-wide service.

OK | Cancel | Help

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**.



SQL Server Login

Data Source: nitey_nite

☐ Use Trusted Connection

Login ID: testuser

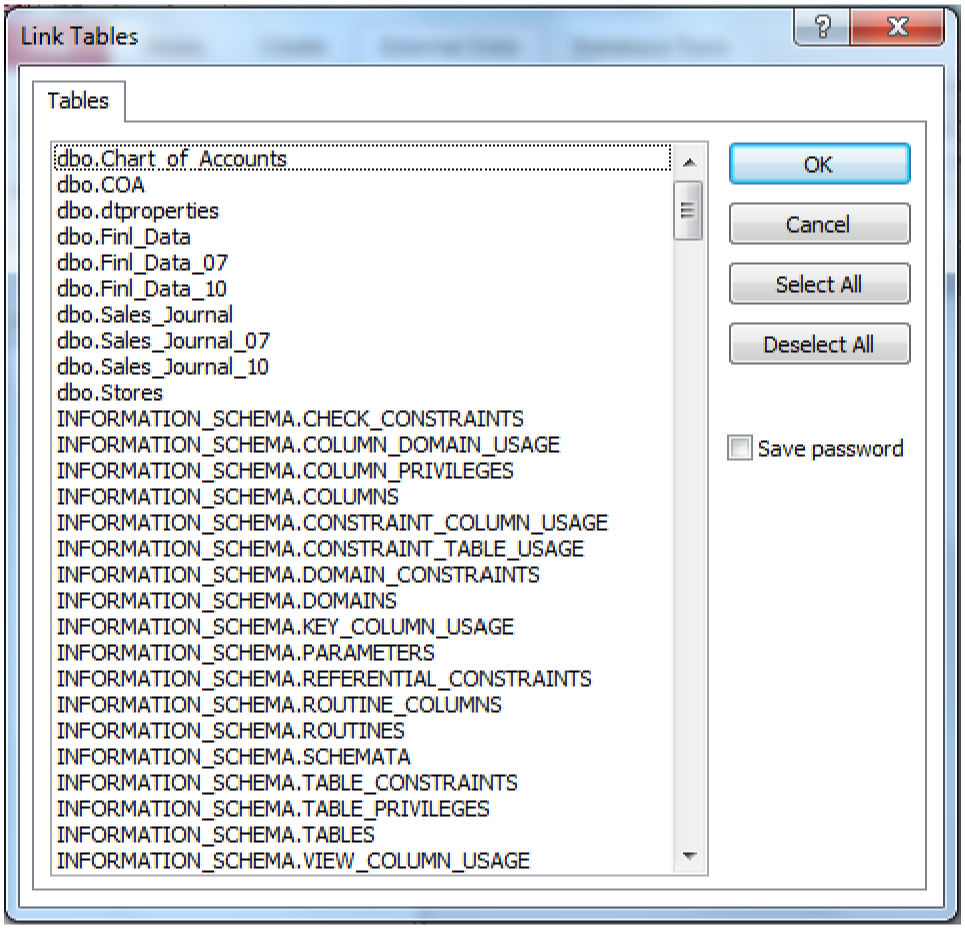
Password:

OK | Cancel | Help | Options >>

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**.

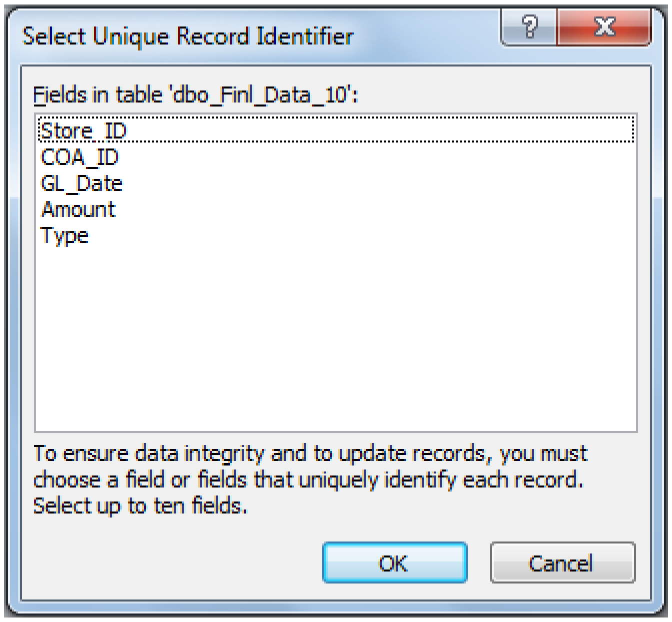


dbo.Chart_of_Accounts
dbo.COA
dbo.dtproperties
dbo.Finl_Data
dbo.Finl_Data_07
dbo.Finl_Data_10
dbo.Sales_Journal
dbo.Sales_Journal_07
dbo.Sales_Journal_10
dbo.Stores
INFORMATION_SCHEMA.CHECK_CONSTRAINTS
INFORMATION_SCHEMA.COLUMN_DOMAIN_USAGE
INFORMATION_SCHEMA.COLUMN_PRIVILEGES
INFORMATION_SCHEMA.COLUMNS
INFORMATION_SCHEMA.CONSTRAINT_COLUMN_USAGE
INFORMATION_SCHEMA.CONSTRAINT_TABLE_USAGE
INFORMATION_SCHEMA.DOMAIN_CONSTRAINTS
INFORMATION_SCHEMA.DOMAINS
INFORMATION_SCHEMA.KEY_COLUMN_USAGE
INFORMATION_SCHEMA.PARAMETERS
INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS
INFORMATION_SCHEMA.ROUTINE_COLUMNS
INFORMATION_SCHEMA.ROUTINES
INFORMATION_SCHEMA.SCHEMATA
INFORMATION_SCHEMA.TABLE_CONSTRAINTS
INFORMATION_SCHEMA.TABLE_PRIVILEGES
INFORMATION_SCHEMA.TABLES
INFORMATION_SCHEMA.VIEW_COLUMN_USAGE

Figure 10.14

The three tables to which you want to link are Chart_of_Accounts, Finl_Data_10, 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_10** *and* **dbo.Stores** *and click* **OK**.


Select Unique Record Identifier

Fields in table 'dbo_Finl_Data_10':

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_10 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**.

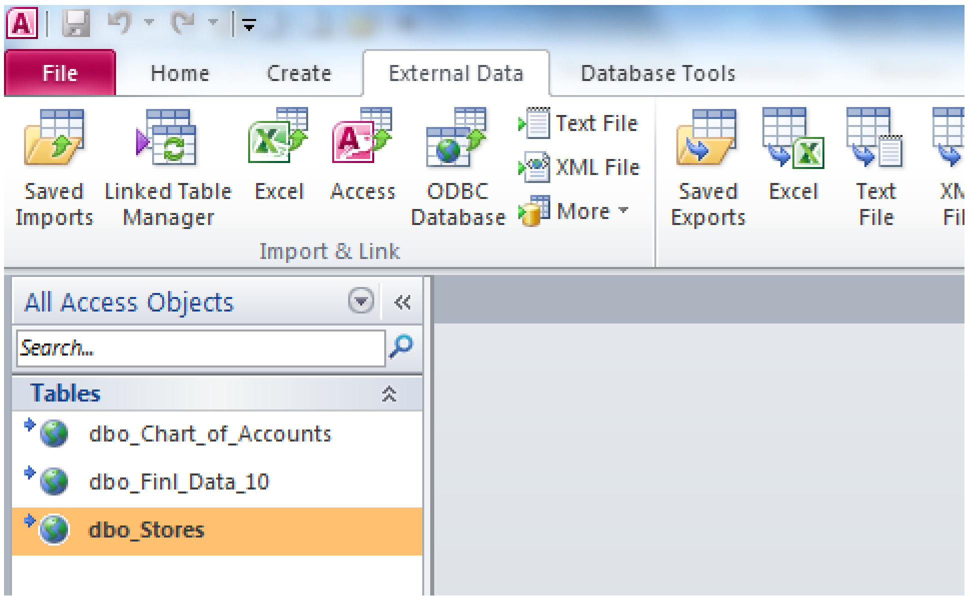


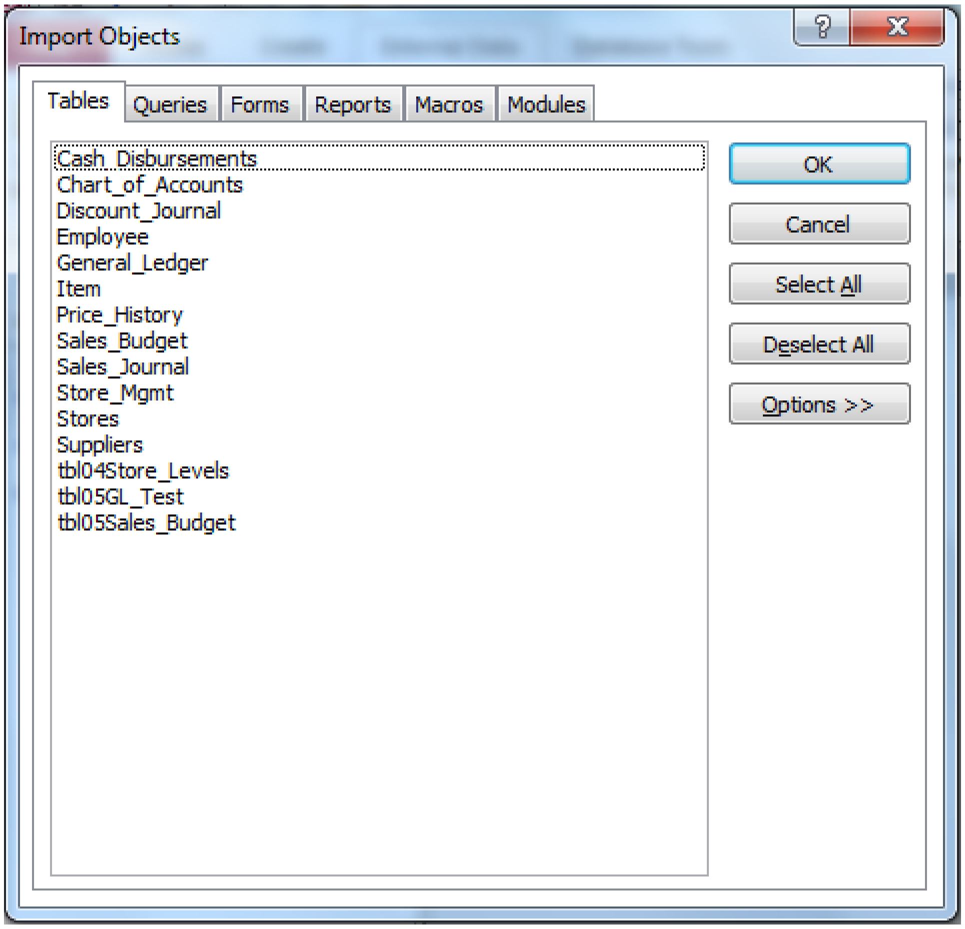
Figure 10.16

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 2010\Nitey_Nite_2010.accdb,** *click* **Open** *then click on* **OK**.



Cash_Disbursements
Chart_of_Accounts
Discount_Journal
Employee
General_Ledger
Item
Price_History
Sales_Budget
Sales_Journal
Store_Mgmt
Stores
Suppliers
tbl04Store_Levels
tbl05GL_Test
tbl05Sales_Budget

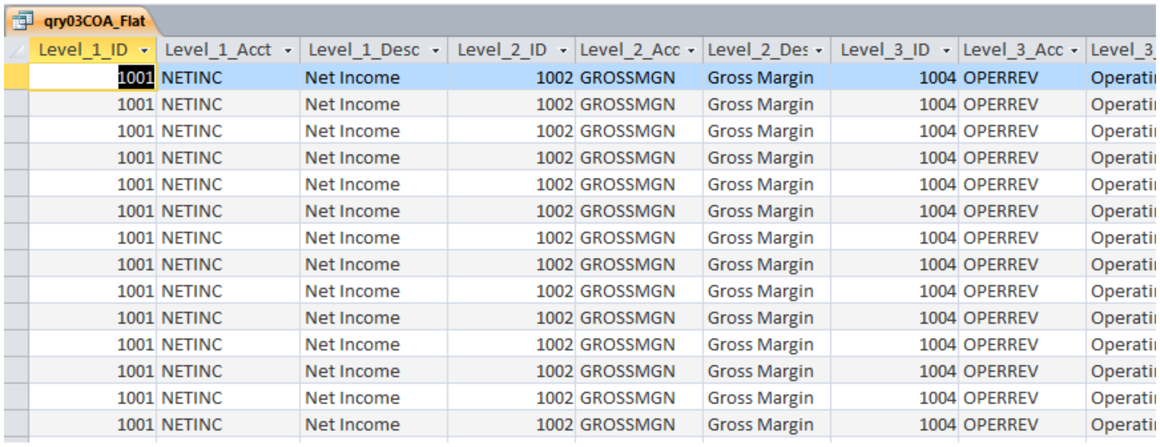
Figure 10.17

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* **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

8. *Open* **qry03COA_Flat**.



qry03COA_Flat

Level_1_ID	Level_1_Acct	Level_1_Desc	Level_2_ID	Level_2_Acc	Level_2_Des	Level_3_ID	Level_3_Acc	Level_3
1001	NETINC	Net Income	1002	GROSSMGN	Gross Margin	1004	OPERREV	Operati
1001	NETINC	Net Income	1002	GROSSMGN	Gross Margin	1004	OPERREV	Operati
1001	NETINC	Net Income	1002	GROSSMGN	Gross Margin	1004	OPERREV	Operati
1001	NETINC	Net Income	1002	GROSSMGN	Gross Margin	1004	OPERREV	Operati
1001	NETINC	Net Income	1002	GROSSMGN	Gross Margin	1004	OPERREV	Operati
1001	NETINC	Net Income	1002	GROSSMGN	Gross Margin	1004	OPERREV	Operati
1001	NETINC	Net Income	1002	GROSSMGN	Gross Margin	1004	OPERREV	Operati
1001	NETINC	Net Income	1002	GROSSMGN	Gross Margin	1004	OPERREV	Operati
1001	NETINC	Net Income	1002	GROSSMGN	Gross Margin	1004	OPERREV	Operati
1001	NETINC	Net Income	1002	GROSSMGN	Gross Margin	1004	OPERREV	Operati
1001	NETINC	Net Income	1002	GROSSMGN	Gross Margin	1004	OPERREV	Operati
1001	NETINC	Net Income	1002	GROSSMGN	Gross Margin	1004	OPERREV	Operati
1001	NETINC	Net Income	1002	GROSSMGN	Gross Margin	1004	OPERREV	Operati
1001	NETINC	Net Income	1002	GROSSMGN	Gross Margin	1004	OPERREV	Operati

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 2010 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 files that can be imported into an Access database.

Chapter Exam

You can now go to www.ExcelCEO.com, log in and take the exam. Make sure that you take the exam on the same computer that you completed the sample files on, as some of the questions on the exam may refer to some of the completed examples.

ExcelCEO

Chief Excel Officer

Access 2010 and SQL

Complete Self-study Course

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.

Advanced Reporting (Part I)

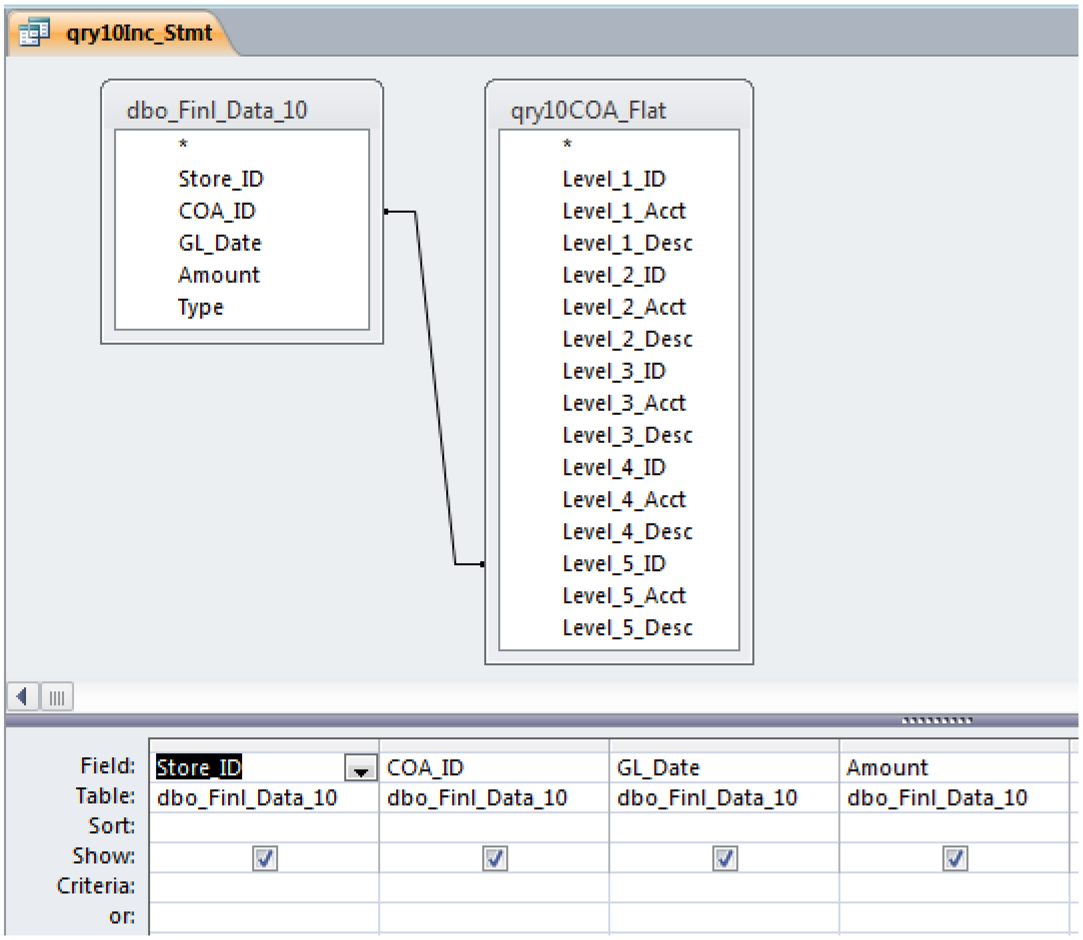
There are so many things I want to cover in Advanced Reporting that I had to break them up into two chapters. Up to this point, you've learned the basics and some intermediate Access reporting skills. You can create a somewhat complex report, and I hope you have mastered queries and forms. Chapters 11 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 following 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.

In this exercise, you will tie the dbo_Finl_Data_10 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* **Inc_Stmt** *database.*
2. *Rename* **qry03COA_Flat** *as* **qry10COA_Flat**.
3. *Create a new query and pull in* **dbo_Finl_Data_10** *and* **qry10COA_Flat**.
4. *Link the* **COA_ID** *field in* **dbo_Finl_Data_10** *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_10	dbo_Finl_Data_10	dbo_Finl_Data_10	dbo_Finl_Data_10
Sort:				
Show:	✓	✓	✓	✓
Criteria:				
or:				

Figure 11.1

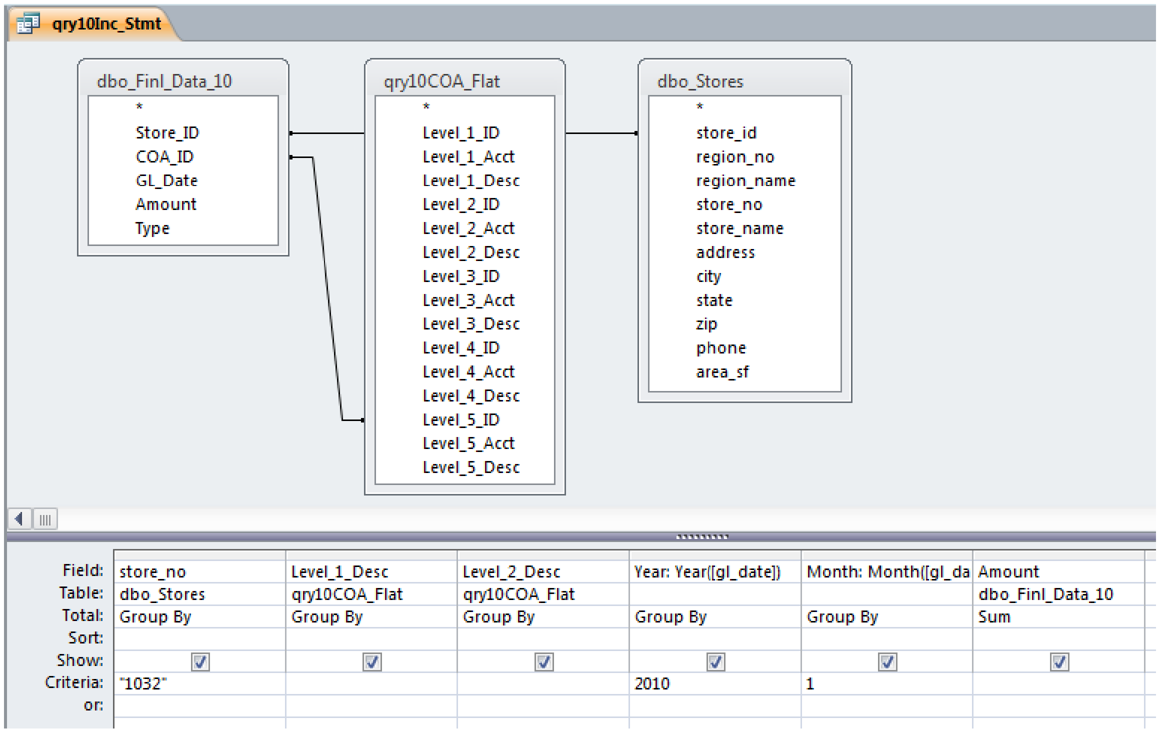
If you closed the Inc_Stmt database after you completed Chapter Ten, and if you are using the external database connections, you will be prompted to input the user name and password provided in Chapter 10 when you run the query. This re-establishes the connection to the hosted server. It may take a minute 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_10** *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)**

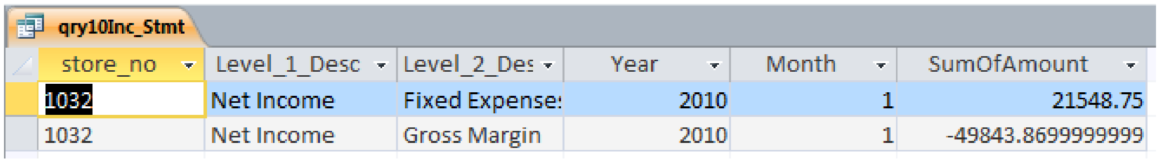
10. *Group all fields (by clicking on the* **Totals** *icon) and change the* **Group By** *on the* **Amount** *field to* **Sum***.*
11. *Filter* **Year** *for* **2010, Month** *for* **1** *and* **Store_No 1032.**
12. *Take out the* **Store_ID, COA_ID** *and* **GL_Date** *fields, and move the* **Amount** *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_da	Amount
Table:	dbo_Stores	qry10COA_Flat	qry10COA_Flat			dbo_Finl_Data_10
Total:	Group By	Group By	Group By	Group By	Group By	Sum
Sort:						
Show:	✓	✓	✓	✓	✓	✓
Criteria:	"1032"			2010	1	
or:						

Figure 11.2

13. *Run the query.*



store_no	Level_1_Desc	Level_2_Des	Year	Month	SumOfAmount
1032	Net Income	Fixed Expense:	2010	1	21548.75
1032	Net Income	Gross Margin	2010	1	-49843.8699999999

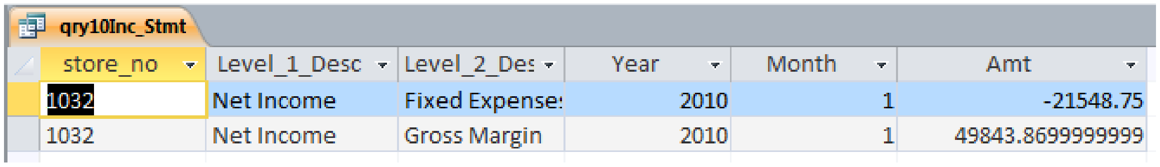
Figure 11.3

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_10 table). It is currently filtered for Store 1032, January 2010. 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 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_10 table, which carries GAAP signs (i.e., revenues are 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

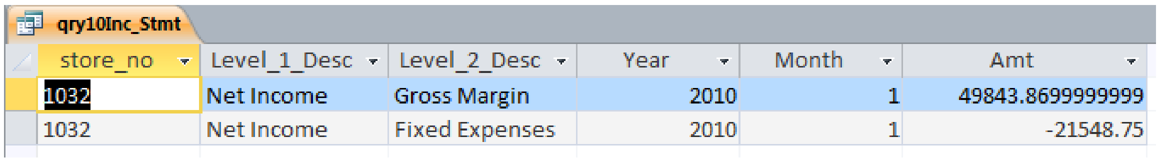
store_no	Level_1_Desc	Level_2_Des	Year	Month	Amt
1032	Net Income	Fixed Expense:	2010	1	-21548.75
1032	Net Income	Gross Margin	2010	1	49843.8699999999

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.*



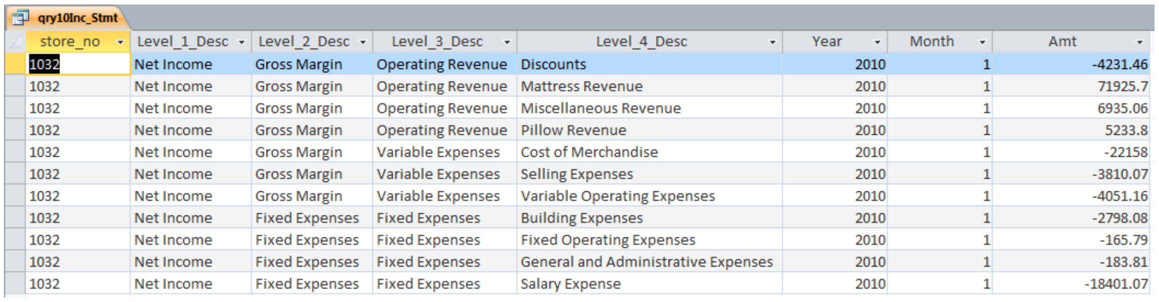
qry10Inc_Stmt

store_no	Level_1_Desc	Level_2_Desc	Year	Month	Amt
1032	Net Income	Gross Margin	2010	1	49843.8699999999
1032	Net Income	Fixed Expenses	2010	1	-21548.75

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.*



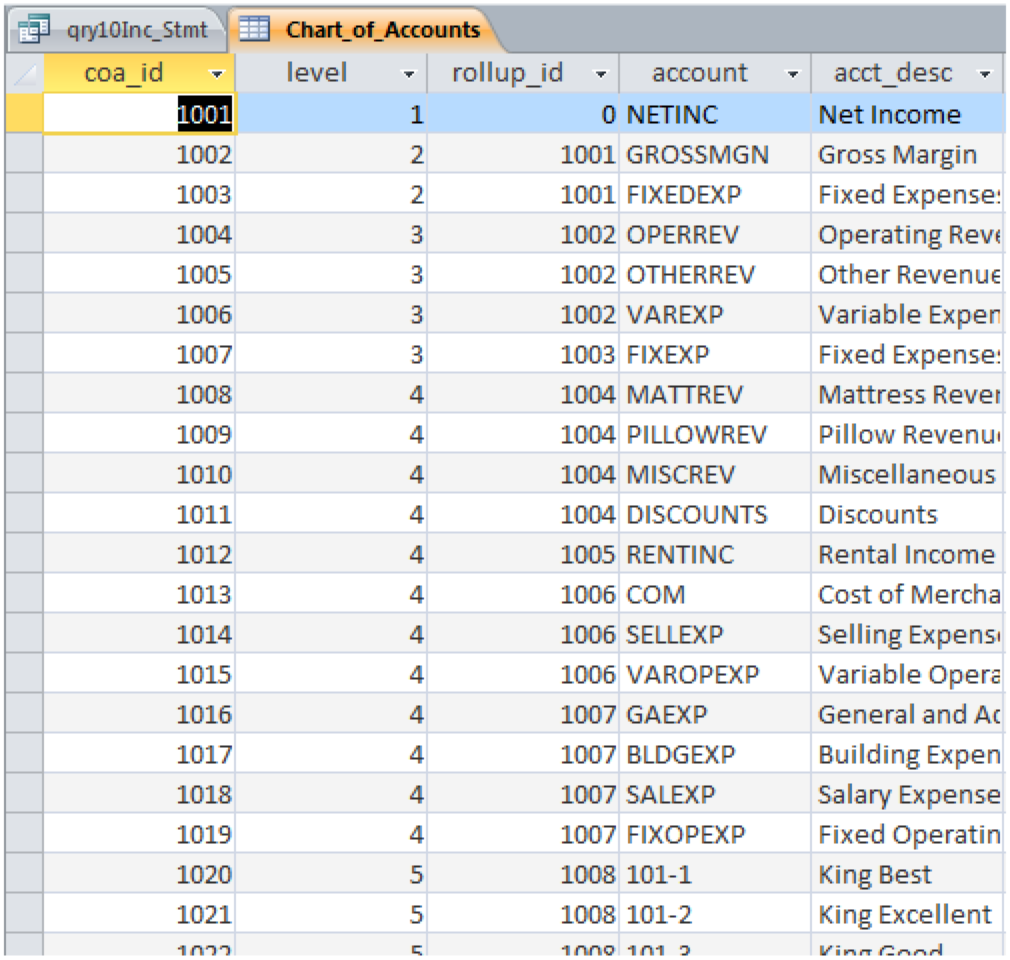
qry10Inc_Stmt

store_no	Level_1_Desc	Level_2_Desc	Level_3_Desc	Level_4_Desc	Year	Month	Amt
1032	Net Income	Gross Margin	Operating Revenue	Discounts	2010	1	-4231.46
1032	Net Income	Gross Margin	Operating Revenue	Mattress Revenue	2010	1	71925.7
1032	Net Income	Gross Margin	Operating Revenue	Miscellaneous Revenue	2010	1	6935.06
1032	Net Income	Gross Margin	Operating Revenue	Pillow Revenue	2010	1	5233.8
1032	Net Income	Gross Margin	Variable Expenses	Cost of Merchandise	2010	1	-22158
1032	Net Income	Gross Margin	Variable Expenses	Selling Expenses	2010	1	-3810.07
1032	Net Income	Gross Margin	Variable Expenses	Variable Operating Expenses	2010	1	-4051.16
1032	Net Income	Fixed Expenses	Fixed Expenses	Building Expenses	2010	1	-2798.08
1032	Net Income	Fixed Expenses	Fixed Expenses	Fixed Operating Expenses	2010	1	-165.79
1032	Net Income	Fixed Expenses	Fixed Expenses	General and Administrative Expenses	2010	1	-183.81
1032	Net Income	Fixed Expenses	Fixed Expenses	Salary Expense	2010	1	-18401.07

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 another way to do it.

20. *Open the* **Chart_of_Accounts** *table.*



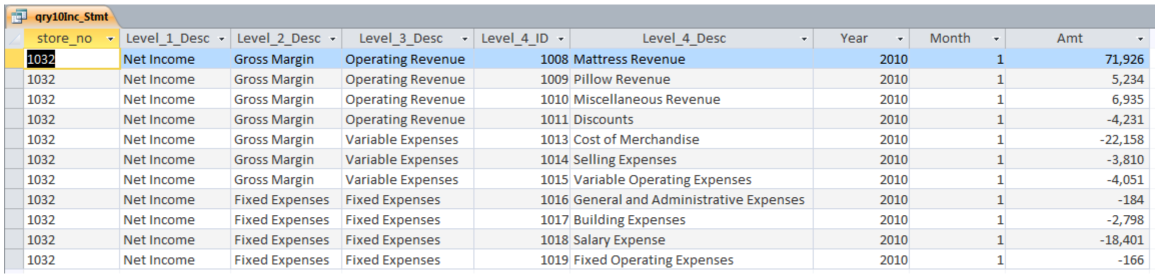
qry10Inc_Stmt | Chart_of_Accounts

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 Expen
1007	3	1003	FIXEXP	Fixed Expense:
1008	4	1004	MATTREV	Mattress Rever
1009	4	1004	PILLOWREV	Pillow Revenu
1010	4	1004	MISCREV	Miscellaneous
1011	4	1004	DISCOUNTS	Discounts
1012	4	1005	RENTINC	Rental Income
1013	4	1006	COM	Cost of Mercha
1014	4	1006	SELLEXP	Selling Expens
1015	4	1006	VAROPEXP	Variable Opera
1016	4	1007	GAEXP	General and Ac
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

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. Also, I'm getting kind of annoyed of seeing the unformatted Amt numbers, so we'll format those as well.

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**, **no decimal places**.
26. *Run the query.*

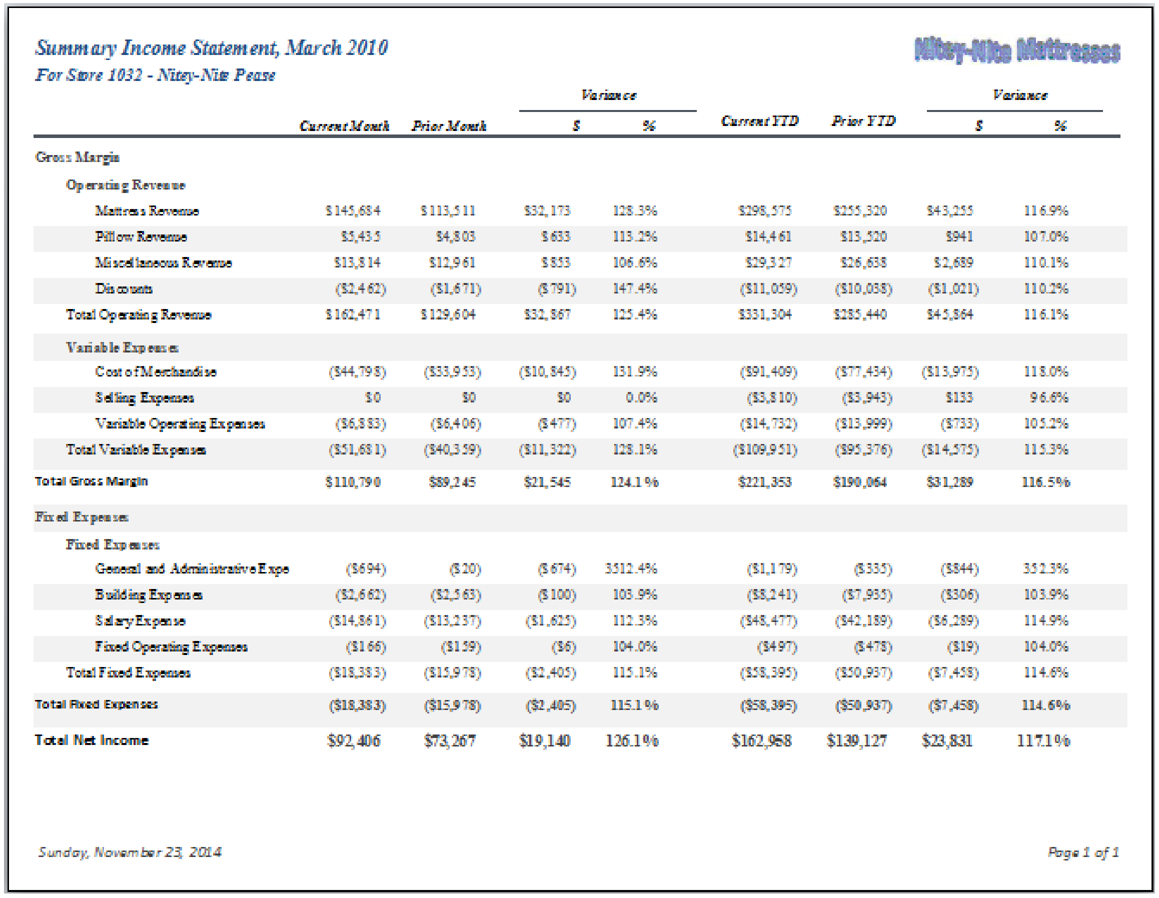


qry10Inc_Stmt

store_no	Level_1_Desc	Level_2_Desc	Level_3_Desc	Level_4_ID	Level_4_Desc	Year	Month	Amt
1032	Net Income	Gross Margin	Operating Revenue	1008	Mattress Revenue	2010	1	71,926
1032	Net Income	Gross Margin	Operating Revenue	1009	Pillow Revenue	2010	1	5,234
1032	Net Income	Gross Margin	Operating Revenue	1010	Miscellaneous Revenue	2010	1	6,935
1032	Net Income	Gross Margin	Operating Revenue	1011	Discounts	2010	1	-4,231
1032	Net Income	Gross Margin	Variable Expenses	1013	Cost of Merchandise	2010	1	-22,158
1032	Net Income	Gross Margin	Variable Expenses	1014	Selling Expenses	2010	1	-3,810
1032	Net Income	Gross Margin	Variable Expenses	1015	Variable Operating Expenses	2010	1	-4,051
1032	Net Income	Fixed Expenses	Fixed Expenses	1016	General and Administrative Expenses	2010	1	-184
1032	Net Income	Fixed Expenses	Fixed Expenses	1017	Building Expenses	2010	1	-2,798
1032	Net Income	Fixed Expenses	Fixed Expenses	1018	Salary Expense	2010	1	-18,401
1032	Net Income	Fixed Expenses	Fixed Expenses	1019	Fixed Operating Expenses	2010	1	-166

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"* (giving credit to 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, March 2010

For Store 1032 - Nitey-Nite Pease

Nitey-Nite Mattresses

	Current Month	Prior Month	Variance $	Variance %	Current YTD	Prior YTD	Variance $	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	$633	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 Expe	($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%

Sunday, November 23, 2014 *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 2010. 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 2010 and the prior month means the prior year's month, which is March 2009. 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 2010 and in 2009. 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 2010 numbers, a column for March 2009 numbers, then two columns containing January through March numbers for 2010 and 2009. 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 2010 numbers and another column with 2009 numbers. Once one year goes by, we'll need 2011 numbers to compare with 2010 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**.

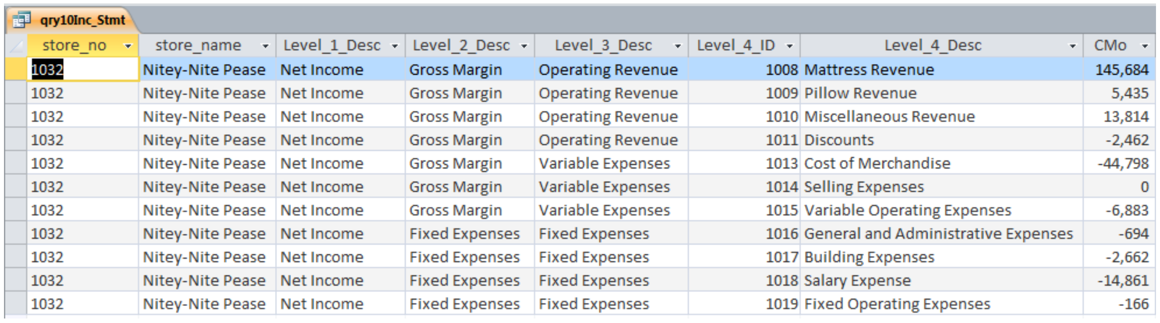
30. *Change the* **Month** *and* **Year** *fields to show* **Where** *on the* **Total** *line (this can also make the query run a little faster since you're not returning data for those fields).*

A **Where** clause filters the query for a specified criteria. 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])=2010,-[amount],0).**

This formula basically runs the query to return the amount field only when the Month is 3 and the Year is 2010. 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.*

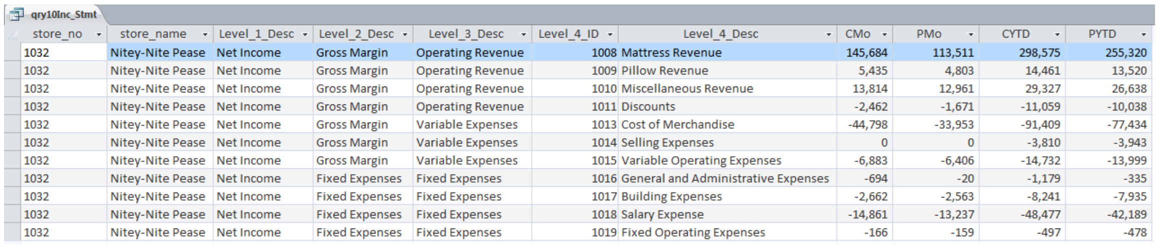

qry10Inc_Stmt

store_no	store_name	Level_1_Desc	Level_2_Desc	Level_3_Desc	Level_4_ID	Level_4_Desc	CMo
1032	Nitey-Nite Pease	Net Income	Gross Margin	Operating Revenue	1008	Mattress Revenue	145,684
1032	Nitey-Nite Pease	Net Income	Gross Margin	Operating Revenue	1009	Pillow Revenue	5,435
1032	Nitey-Nite Pease	Net Income	Gross Margin	Operating Revenue	1010	Miscellaneous Revenue	13,814
1032	Nitey-Nite Pease	Net Income	Gross Margin	Operating Revenue	1011	Discounts	-2,462
1032	Nitey-Nite Pease	Net Income	Gross Margin	Variable Expenses	1013	Cost of Merchandise	-44,798
1032	Nitey-Nite Pease	Net Income	Gross Margin	Variable Expenses	1014	Selling Expenses	0
1032	Nitey-Nite Pease	Net Income	Gross Margin	Variable Expenses	1015	Variable Operating Expenses	-6,883
1032	Nitey-Nite Pease	Net Income	Fixed Expenses	Fixed Expenses	1016	General and Administrative Expenses	-694
1032	Nitey-Nite Pease	Net Income	Fixed Expenses	Fixed Expenses	1017	Building Expenses	-2,662
1032	Nitey-Nite Pease	Net Income	Fixed Expenses	Fixed Expenses	1018	Salary Expense	-14,861
1032	Nitey-Nite Pease	Net Income	Fixed Expenses	Fixed Expenses	1019	Fixed Operating Expenses	-166

Figure 11.10

Now that we 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])=2009,-amount,0)**
    - **CYTD: IIf(Month([gl_date]) Between 1 And 3 And Year([gl_date])=2010,-amount,0)**
    - **PYTD: IIf(Month([gl_date]) Between 1 And 3 And Year([gl_date])=2009,-amount,0)**
34. *Change the* **Total** *line on all the new formula fields to* **Sum***.*
35. *Modify the criteria in the* **Year** *field to be* **2010 or 2009***.*
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

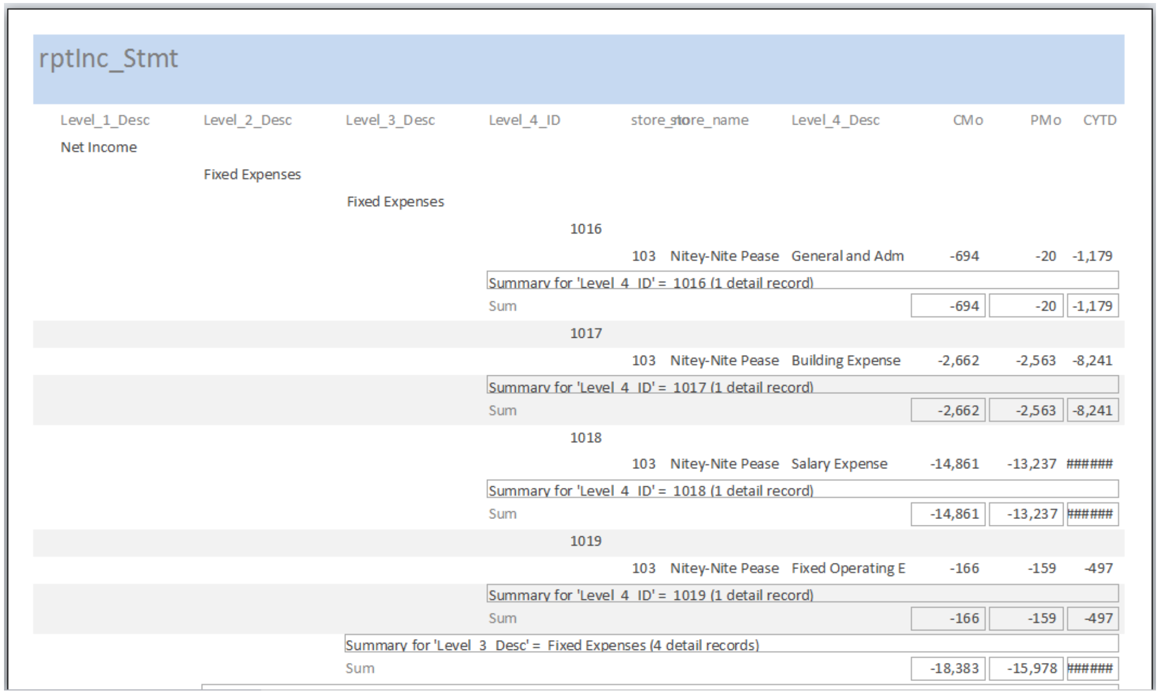
store_no	store_name	Level_1_Desc	Level_2_Desc	Level_3_Desc	Level_4_ID	Level_4_Desc	CMo	PMo	CYTD	PYTD
1032	Nitey-Nite Pease	Net Income	Gross Margin	Operating Revenue	1008	Mattress Revenue	145,684	113,511	298,575	255,320
1032	Nitey-Nite Pease	Net Income	Gross Margin	Operating Revenue	1009	Pillow Revenue	5,435	4,803	14,461	13,520
1032	Nitey-Nite Pease	Net Income	Gross Margin	Operating Revenue	1010	Miscellaneous Revenue	13,814	12,961	29,327	26,638
1032	Nitey-Nite Pease	Net Income	Gross Margin	Operating Revenue	1011	Discounts	-2,462	-1,671	-11,059	-10,038
1032	Nitey-Nite Pease	Net Income	Gross Margin	Variable Expenses	1013	Cost of Merchandise	-44,798	-33,953	-91,409	-77,434
1032	Nitey-Nite Pease	Net Income	Gross Margin	Variable Expenses	1014	Selling Expenses	0	0	-3,810	-3,943
1032	Nitey-Nite Pease	Net Income	Gross Margin	Variable Expenses	1015	Variable Operating Expenses	-6,883	-6,406	-14,732	-13,999
1032	Nitey-Nite Pease	Net Income	Fixed Expenses	Fixed Expenses	1016	General and Administrative Expenses	-694	-20	-1,179	-335
1032	Nitey-Nite Pease	Net Income	Fixed Expenses	Fixed Expenses	1017	Building Expenses	-2,662	-2,563	-8,241	-7,935
1032	Nitey-Nite Pease	Net Income	Fixed Expenses	Fixed Expenses	1018	Salary Expense	-14,861	-13,237	-48,477	-42,189
1032	Nitey-Nite Pease	Net Income	Fixed Expenses	Fixed Expenses	1019	Fixed Operating Expenses	-166	-159	-497	-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 the query.*
2. *In the navigation pane, click on* **qry10Inc_Stmt***.*
3. *Click on the* **Create** *tab then click on the* **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. *Don't choose a* **Sorting** *field, but click on the* **Summary Options** *button and choose* **Sum** *for all of the amount fields.*
7. *Choose a* **Stepped Layout** *and* **Landscape** *orientation options.*
8. *Name the report* **rptInc_Stmt***.*



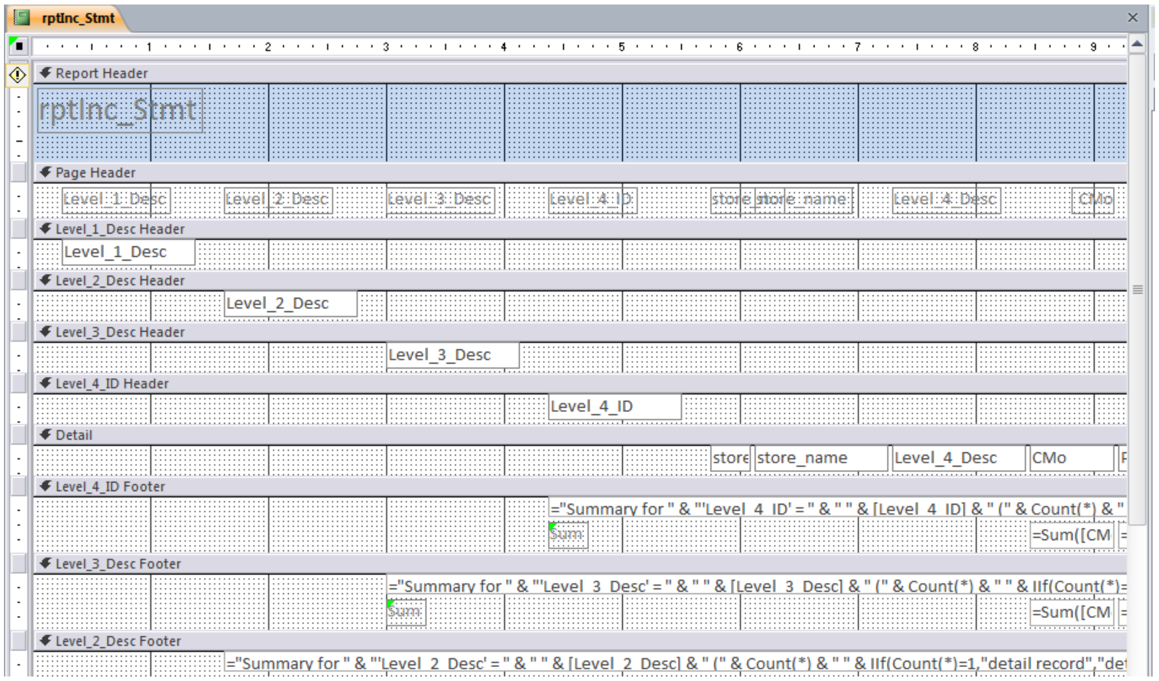
```
rptInc_Stmt

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     -20    -1,179
                                             Summary for 'Level_4_ID' = 1016 (1 detail record)
                                             Sum                                                    -694     -20    -1,179
                                                  1017
                                                         103  Nitey-Nite Pease  Building Expense  -2,662  -2,563    -8,241
                                             Summary for 'Level_4_ID' = 1017 (1 detail record)
                                             Sum                                                  -2,662  -2,563    -8,241
                                                  1018
                                                         103  Nitey-Nite Pease  Salary Expense   -14,861 -13,237    ######
                                             Summary for 'Level_4_ID' = 1018 (1 detail record)
                                             Sum                                                 -14,861 -13,237    ######
                                                  1019
                                                         103  Nitey-Nite Pease  Fixed Operating E   -166    -159      -497
                                             Summary for 'Level_4_ID' = 1019 (1 detail record)
                                             Sum                                                    -166    -159      -497
                              Summary for 'Level_3_Desc' = Fixed Expenses (4 detail records)
                              Sum                                                                -18,383 -15,978    ######
```

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.

9. *Go to the* **Design View** *of the report.*



```
="Summary for " & "'Level_4_ID' = " & " " & [Level_4_ID] & " (" & Count(*) & "
="Summary for " & "'Level_3_Desc' = " & " " & [Level_3_Desc] & " (" & Count(*) & " " & IIf(Count(*)=
="Summary for " & "'Level_2_Desc' = " & " " & [Level_2_Desc] & " (" & Count(*) & " " & IIf(Count(*)=1,"detail record","de
```

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, I personally don't use the report header much, so let's take it out. We'll move the report header label down to the Page Header section. We'll 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 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.

10. *In* **Design View** *of the report, close any dialog boxes that may appear.*
11. *Expand the* **Page Header** *section to be* **1"** *tall***.**
12. *Move all labels in the* **Page Header** *section down to the bottom part of that section.*
13. *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 2010***.*
14. *Format it to be* **Times New Roman, 14 pt, bold** *and* **italicized***.*
15. *Rearrange the borders of the label to fit over its text, if necessary.*
16. *Collapse the* **Report Header** *section.*
17. *Import the* **Nitey-Nite logo** *from the* **Access 2010** *folder. (***Hint***: use the* **Logo** *icon in the* **Header / Footer** *group of the* **Design** *tab).*

Layout Grouping

The image should import into the Report Header section of the report. When you 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. Therefore, you need to remove the layout before continuing.

18. *To remove the* **Layout** *property of the image, right-click on the image, hover over* **Layout***, and click on* **Remove Layout**.

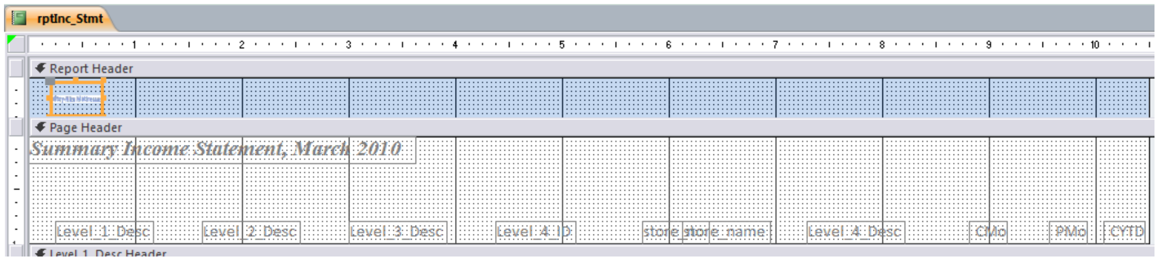


Figure 11.14

The layout should be removed, evidenced by the disappearance of the dotted lines. You can now continue.

19. *Move the logo image into the* **Page Header** *section of the report and align it to the top of the section all the way to the right.*
20. *Collapse the* **Report Header** *section.*
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* **0.33 "** *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**.
24. *Delete all of the labels in the bottom part of the* **Page Header** *section.*
25. *Create the labels that are shown in* **Figure 11.15**.
26. *Format all of those labels to be* **Times New Roman, 10 pt, bold** *and* **italicized, no background color, black font color**, *and position the labels as in* **Figure 11.15**.
27. *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 Dark**.
28. *Make the "***Summary Income Statement, March 2010***" and the "***For Store 1032 – Nitey-Nite Pease***" labels a font color of* **Dark Blue**.
29. *Draw a* **horizontal line** *at the bottom of the* **Page Header** *section that extends most of the way to the right. Make the line the same color as the ones under the* **Variance** *labels with a* **3 pt** *width.*

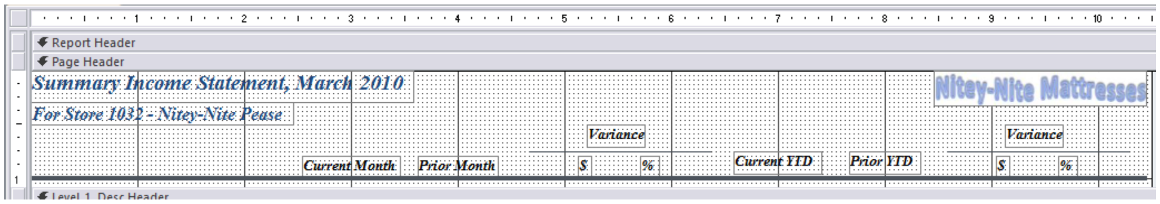


Figure 11.15

This is 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 Levels 2 and 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, click on the* **More** *icon, and change the* **with A on top** *to* **with Z on top** *(to make the group sort 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**.

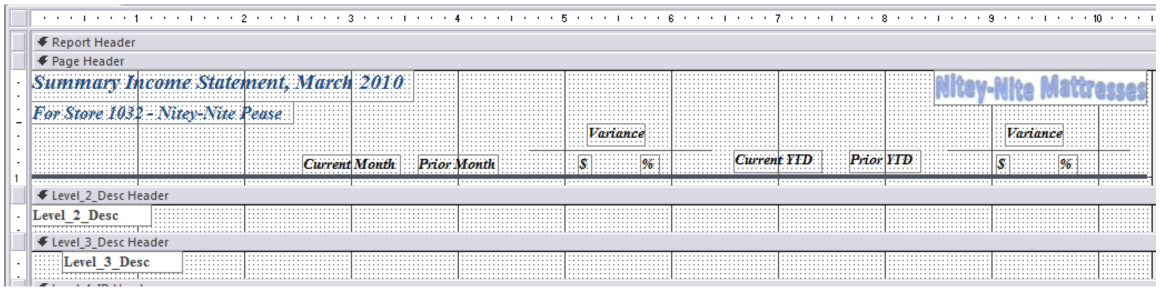


Figure 11.16

Next we'll modify the lowest level of account activity, Level 4. You need to pay close attention to this one, 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 color 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.*

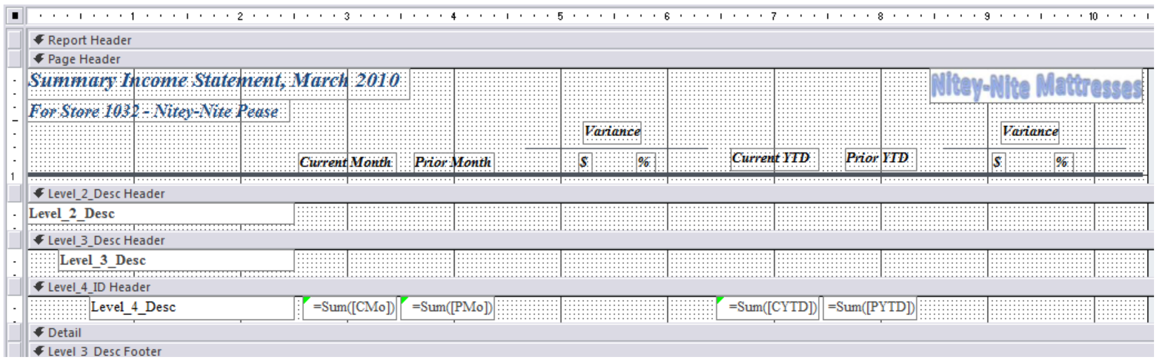


Figure 11.17

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.*
21. *Format the* **dollar** *variances as* **Currency**, **zero decimal places**, *and the* **percent** *variances as* **Percent**, **one decimal place**.

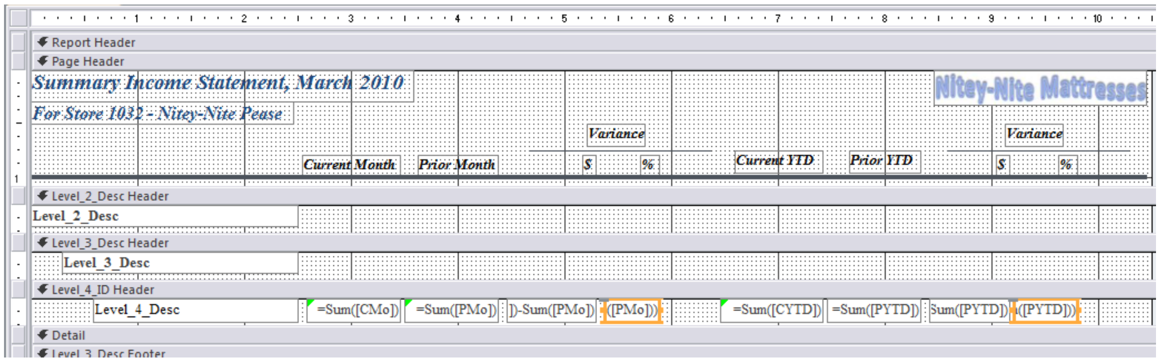


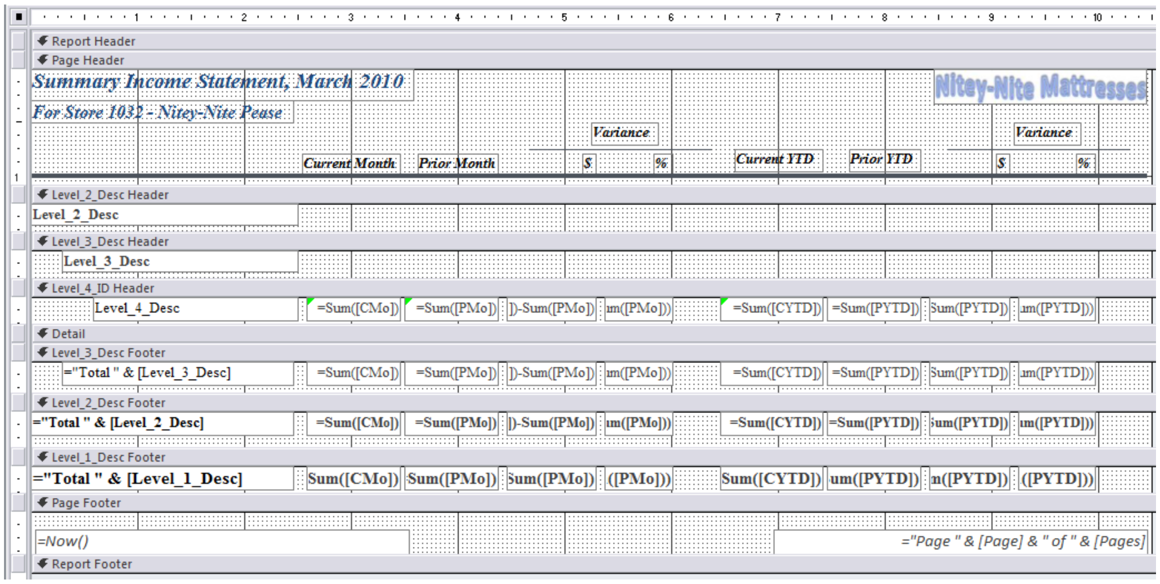
Figure 11.18

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**, **no decimal places**. **Percentage** *amounts should be formatted as* **Percent**, **one decimal place**.
32. *Italicize the text boxes in the* **Page Footer** *section.*
33. *Close the* **Group, Sort, and Total** *section.*
34. *Go back and forth between* **Design View** *and* **Report View** *to make sure that all numbers and descriptions are displayed appropriately.*

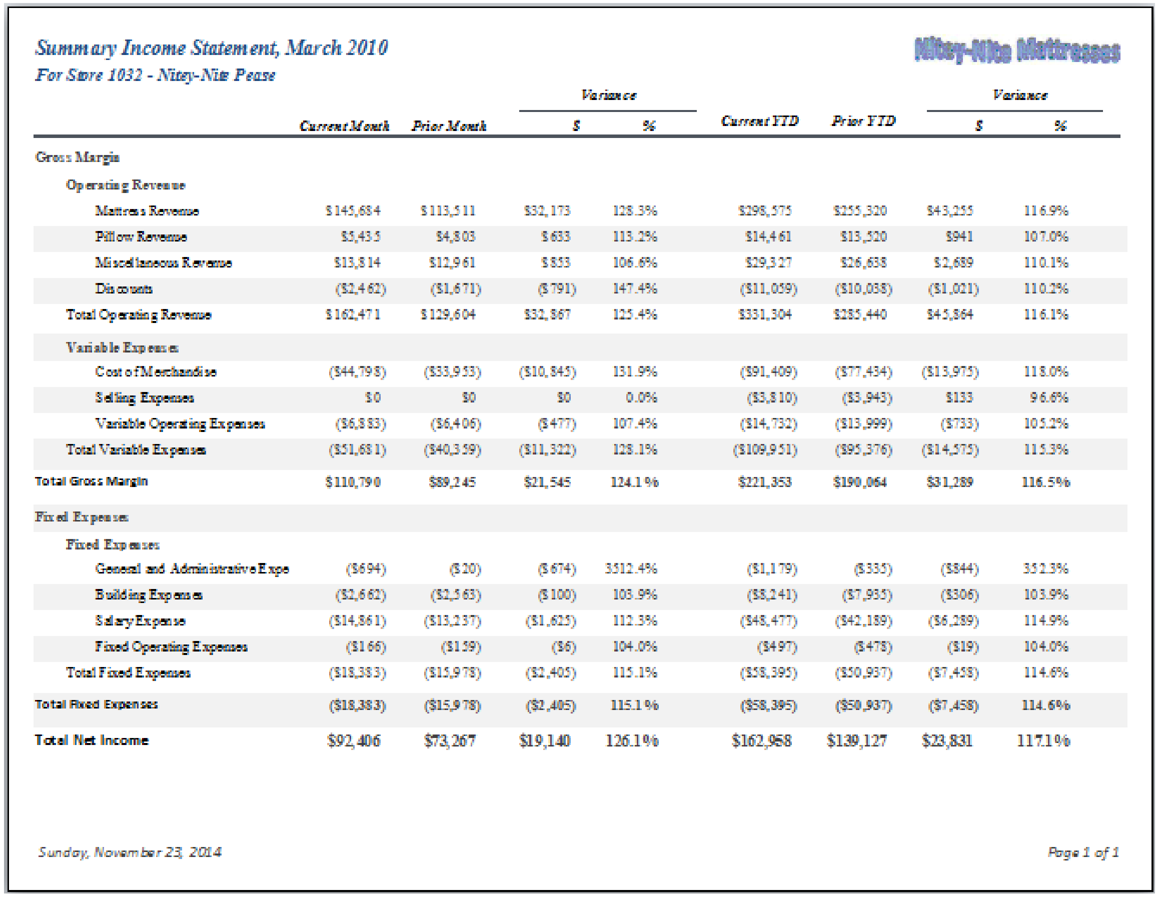


```
Report Header
Page Header
Summary Income Statement, March 2010                                                       Nitey-Nite Mattresses
For Store 1032 - Nitey-Nite Pease
                                                  Variance                                          Variance
                  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([PMo])  ım([PMo]))   =Sum([CYTD])  =Sum([PYTD])  Sum([PYTD])  ım([PYTD]))
Detail
Level_3_Desc Footer
  ="Total " & [Level_3_Desc]   =Sum([CMo])  =Sum([PMo])  ])-Sum([PMo])  ım([PMo]))   =Sum([CYTD])  =Sum([PYTD])  Sum([PYTD])  ım([PYTD]))
Level_2_Desc Footer
="Total " & [Level_2_Desc]     =Sum([CMo])  =Sum([PMo])  ])-Sum([PMo])  ım([PMo]))   =Sum([CYTD])  =Sum([PYTD])  Sum([PYTD])  ım([PYTD]))
Level_1_Desc Footer
="Total " & [Level_1_Desc]     Sum([CMo])  Sum([PMo])  Sum([PMo])  ([PMo]))   Sum([CYTD])  um([PYTD])  n([PYTD])  ([PYTD]))
Page Footer
=Now()                                                              ="Page " & [Page] & " of " & [Pages]
Report Footer
```

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 2010 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.

35. *View the report in* **Print Preview** *mode.*
36. *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.*
37. *Save the report and run it again in* **Print Preview** *mode.*



Summary Income Statement, March 2010
For Store 1032 - Nitey-Nite Pease

Nitey-Nite Mattresses

	Current Month	Prior Month	Variance $	Variance %	Current YTD	Prior YTD	Variance $	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	$633	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 Expe	($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%

Sunday, November 23, 2014 *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 2010 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 2010 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 the report.*
2. *Create a* **blank unbound form** *(using the* **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 point***,* **black text***,* **bold** *and* **italicized** *and position the label as shown in* **Figure 11.21***.*

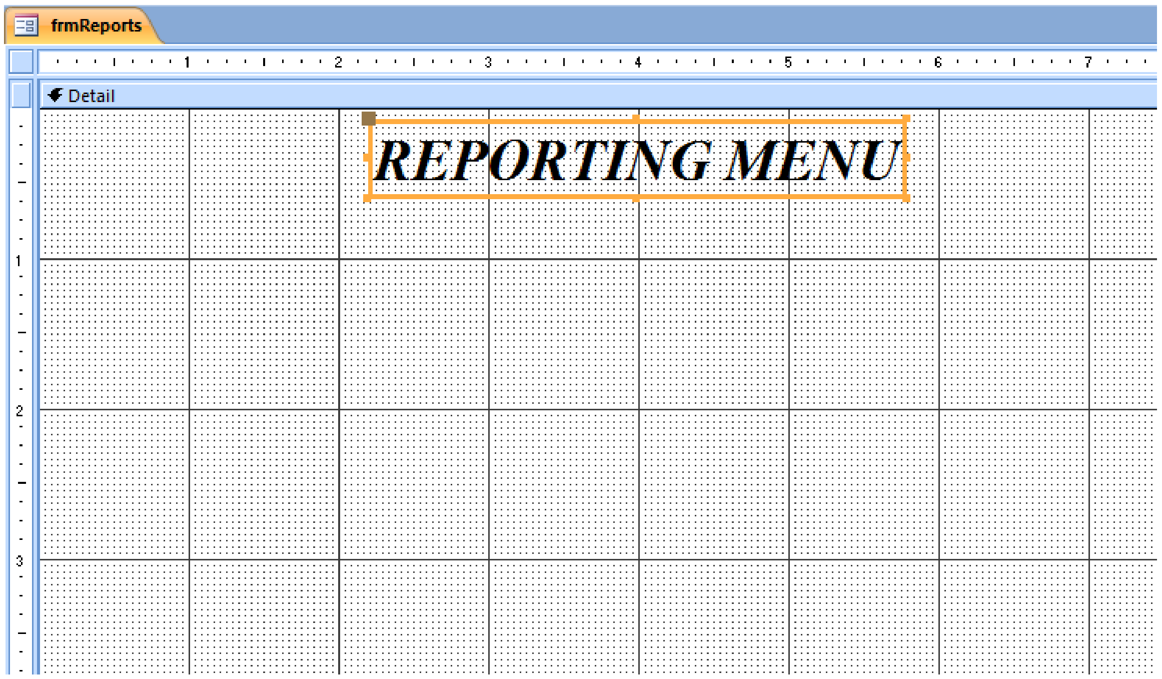


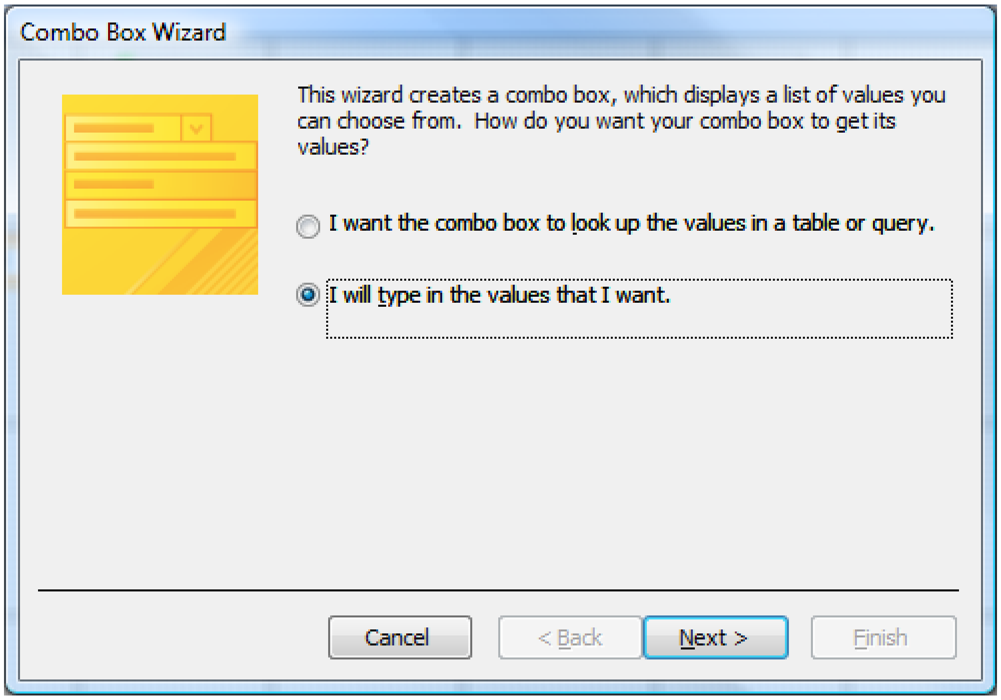
Figure 11.21

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.

7. *Choose the* **I will type in the values that I want** *option and click* **Next >**.



Combo Box Wizard

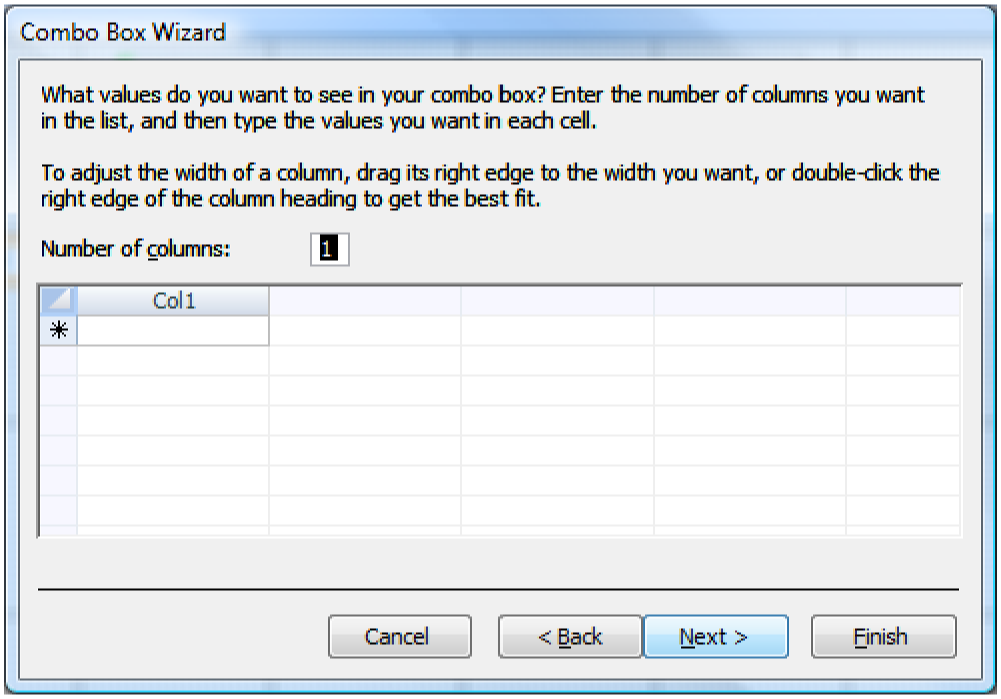
This wizard creates a combo box, which displays a list of values you can choose from. How do you want your combo box to get its values?

I want the combo box to look up the values in a table or query.

[✓ I will type in the values that I want.]

Figure 11.22

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.

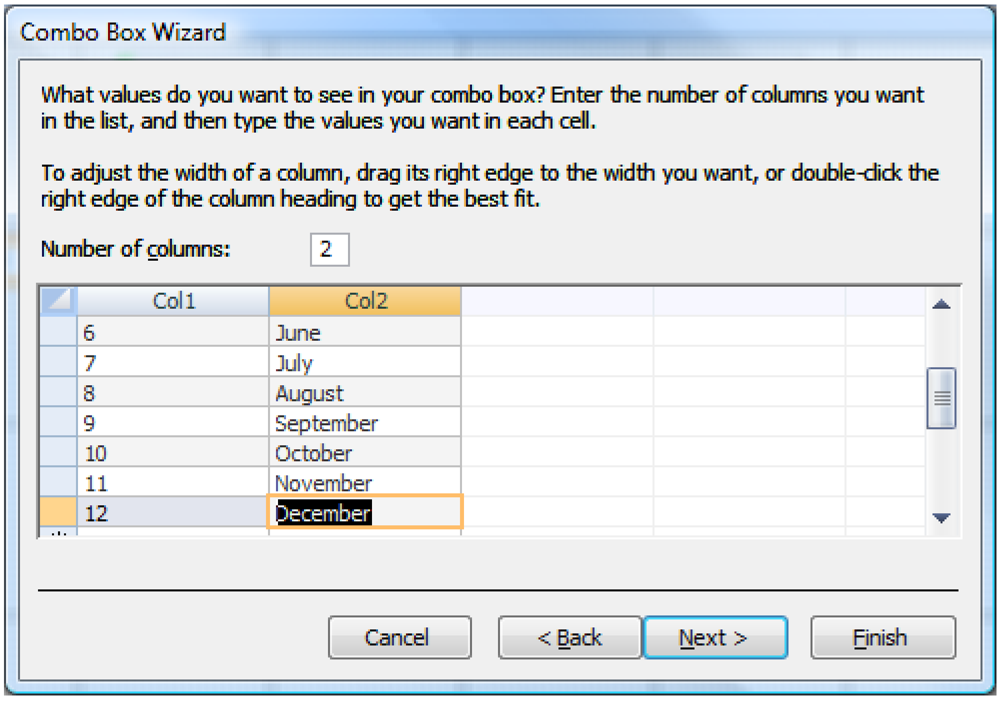
To adjust the width of a column, drag its right edge to the width you want, or double-click the right edge of the column heading to get the best fit.

Number of columns: 1

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 10.22**.



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

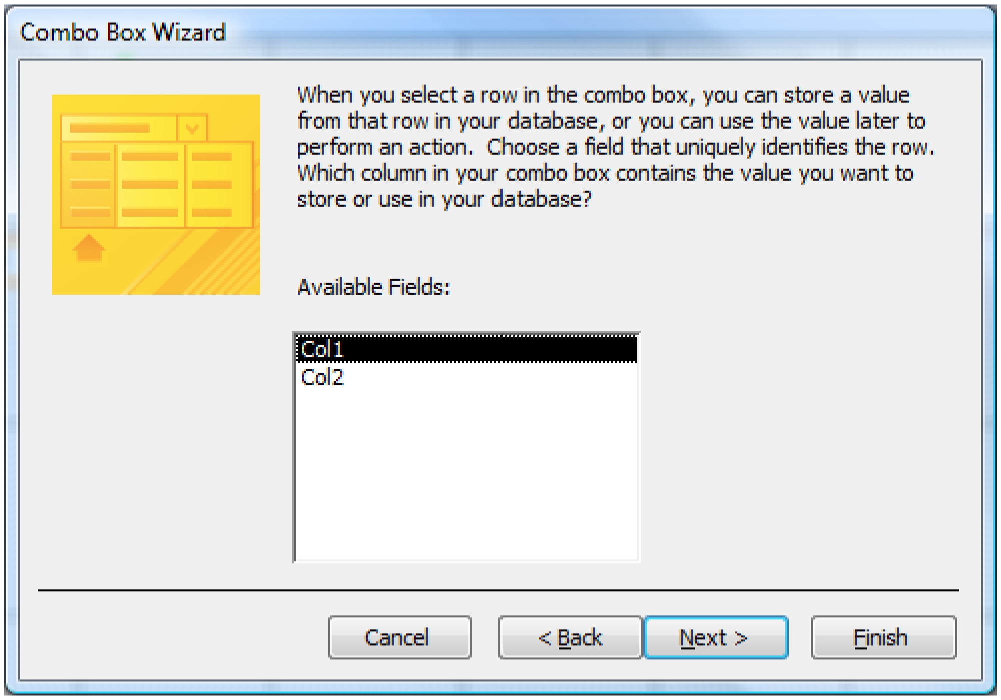
Col1	Col2
6	June
7	July
8	August
9	September
10	October
11	November
12	December

Figure 11.24

11. Click **Next >**.

The next step of the wizard asks you to pick a value that will be used. 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

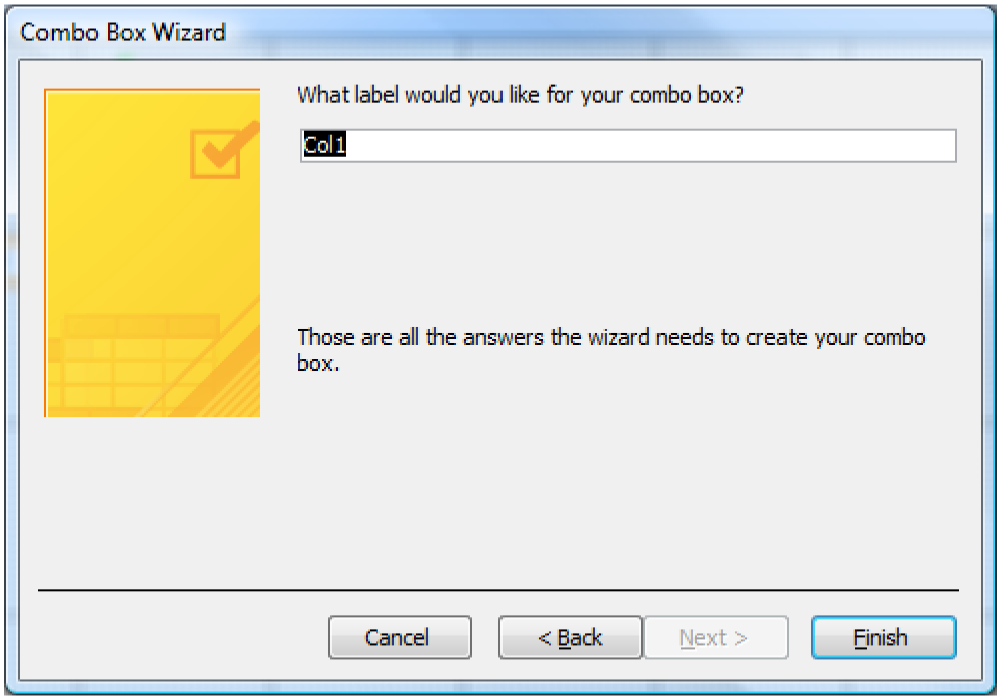
When you select a row in the combo box, you can store a value from that row in your database, or you can use the value later to perform an action. Choose a field that uniquely identifies the row. Which column in your combo box contains the value you want to store or use in your database?

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***.*

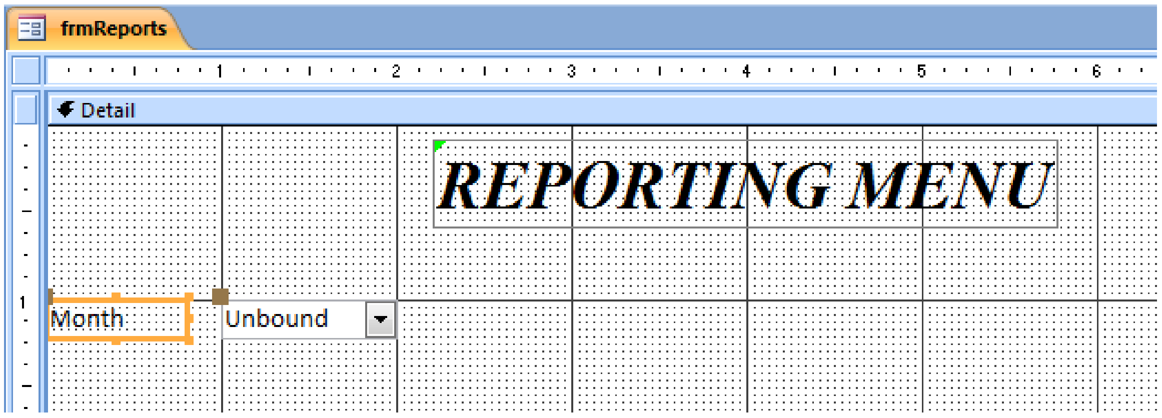
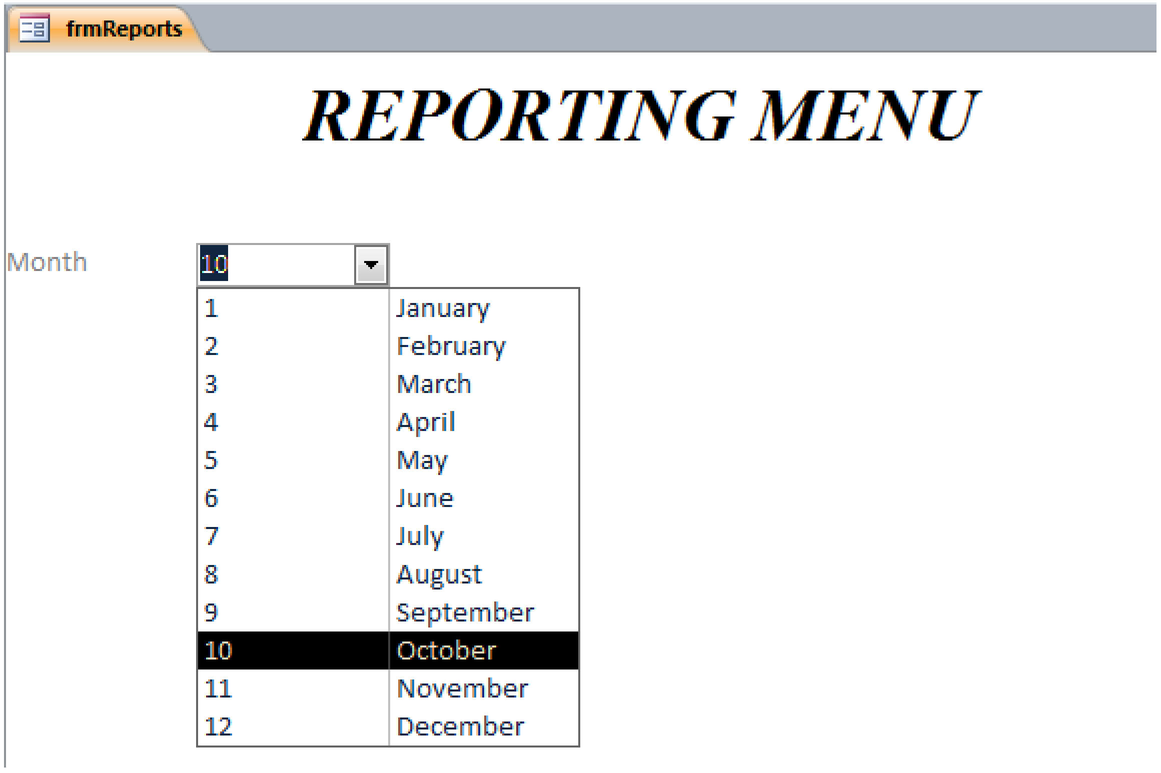


Figure 11.27

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 properties**, *set* **Record Selectors** *and* **Navigation Buttons** *to* **No**.
17. *Save the form.*
18. *Run the* **Form** *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 2009, 2010 and 2011. Even though we have 2008 data in the table behind it, 2009 is the first year where we can do a year-over-year comparison, so it's not necessary to have 2008 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 2009**, **2010**, *and* **2011**.
20. *Name it* **cmbYear**, *save the form, and view it.*

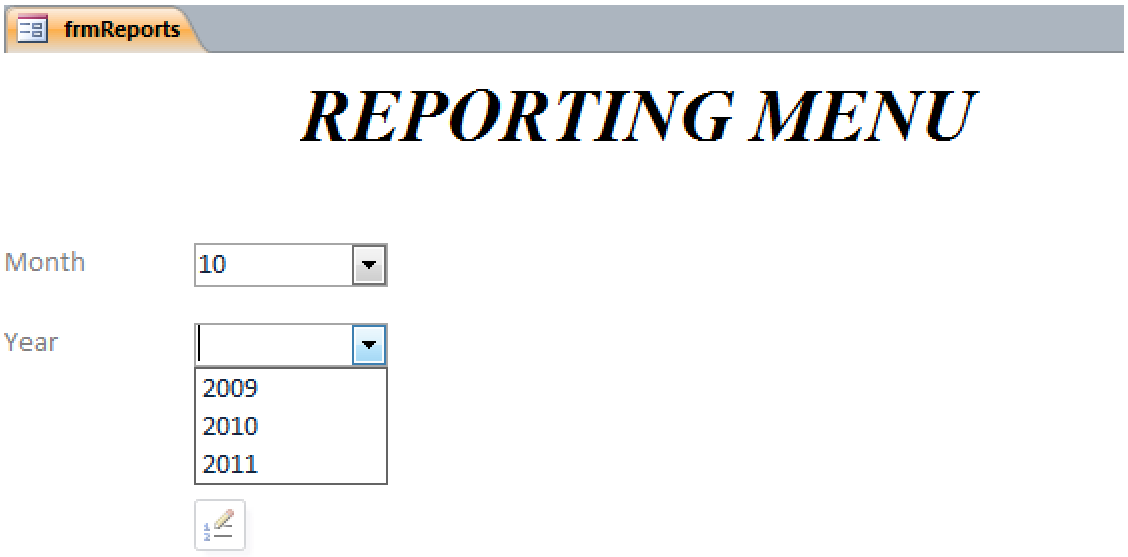
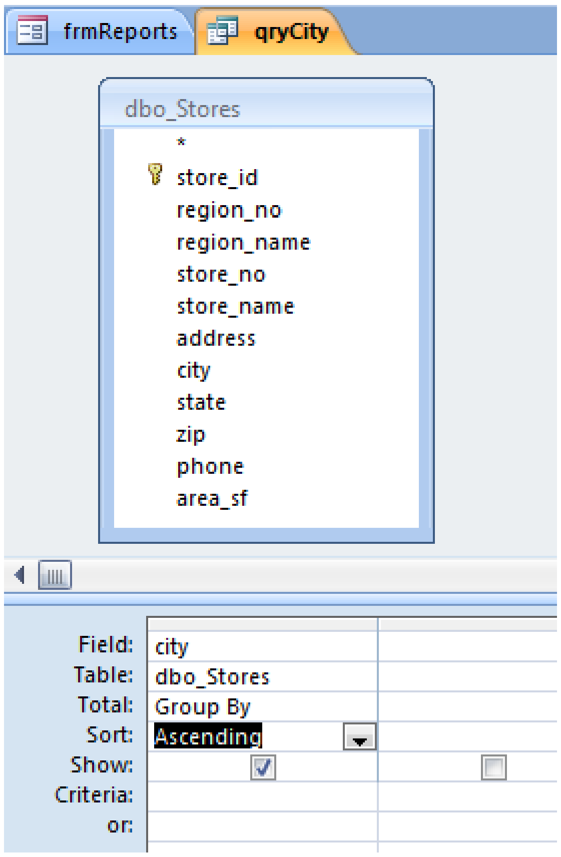


Figure 11.29

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**.



Field:	city
Table:	dbo_Stores
Total:	Group By
Sort:	Ascending
Show:	✓
Criteria:	
or:	

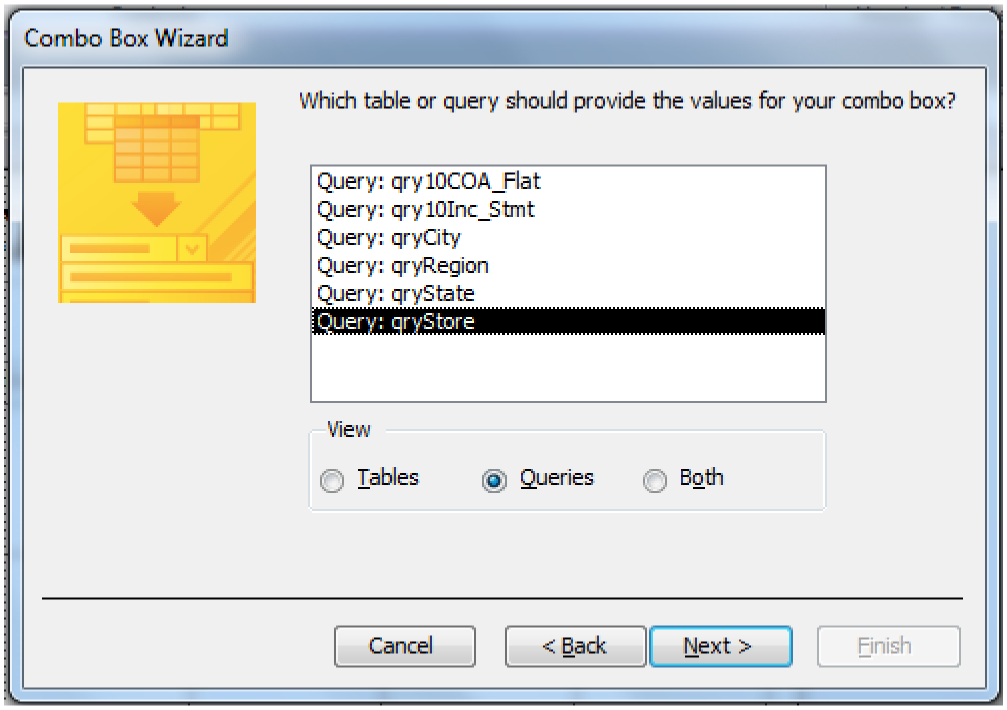
Figure 11.30

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* **dash**. *Call that field* **Store**.
6. *Name that query* **qryStore**.

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

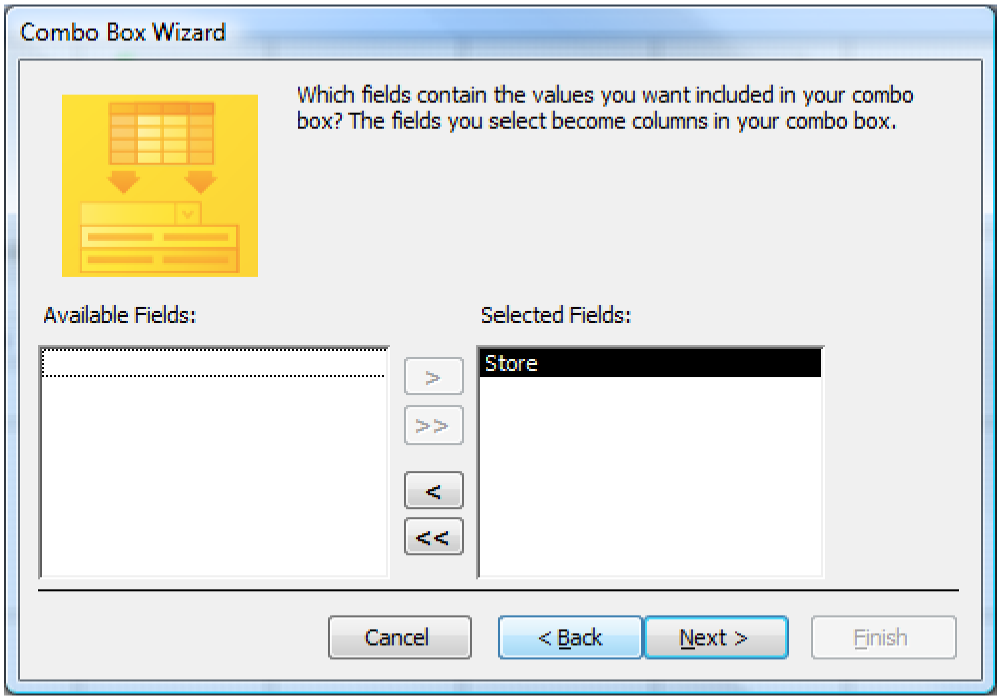
Which table or query should provide the values for your combo box?

Query: qry10COA_Flat
Query: qry10Inc_Stmt
Query: qryCity
Query: qryRegion
Query: qryState
Query: qryStore

View: Tables [✓ Queries] Both

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 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. *Next, adjust the* **margins** *of the column 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**.

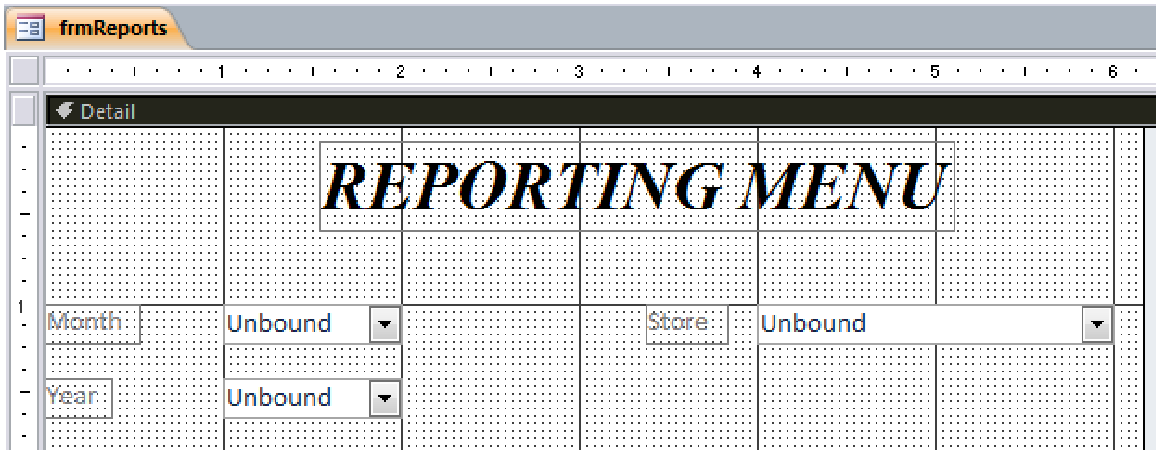


Figure 11.33

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.**

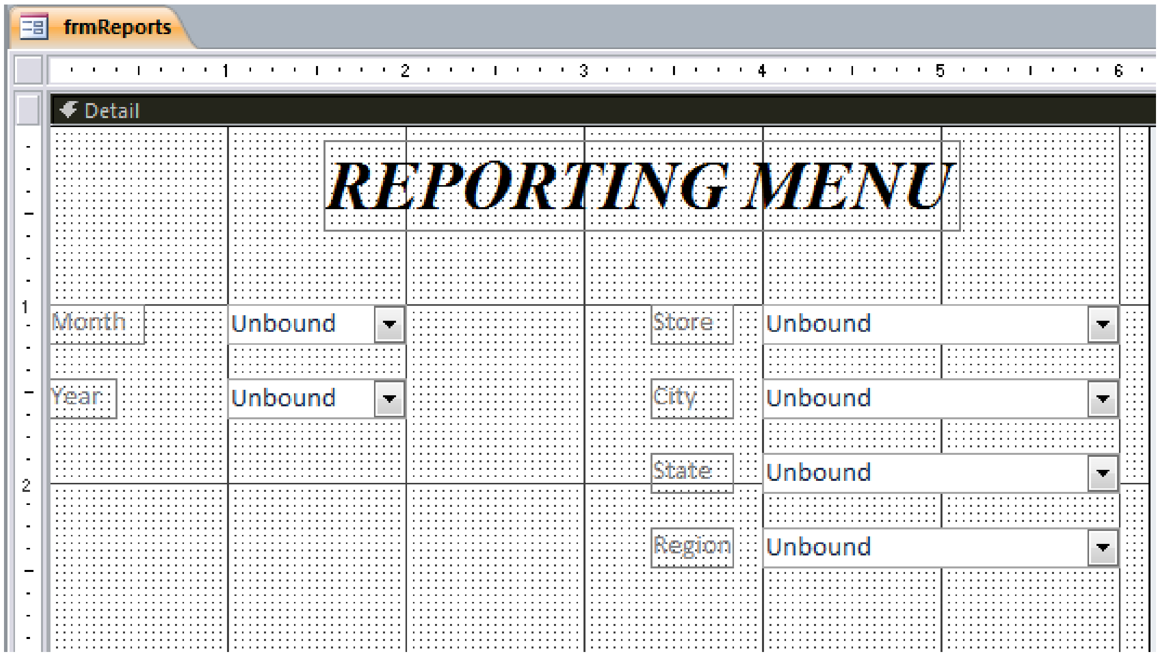


Figure 11.34

14. *Display in* **Form View**.

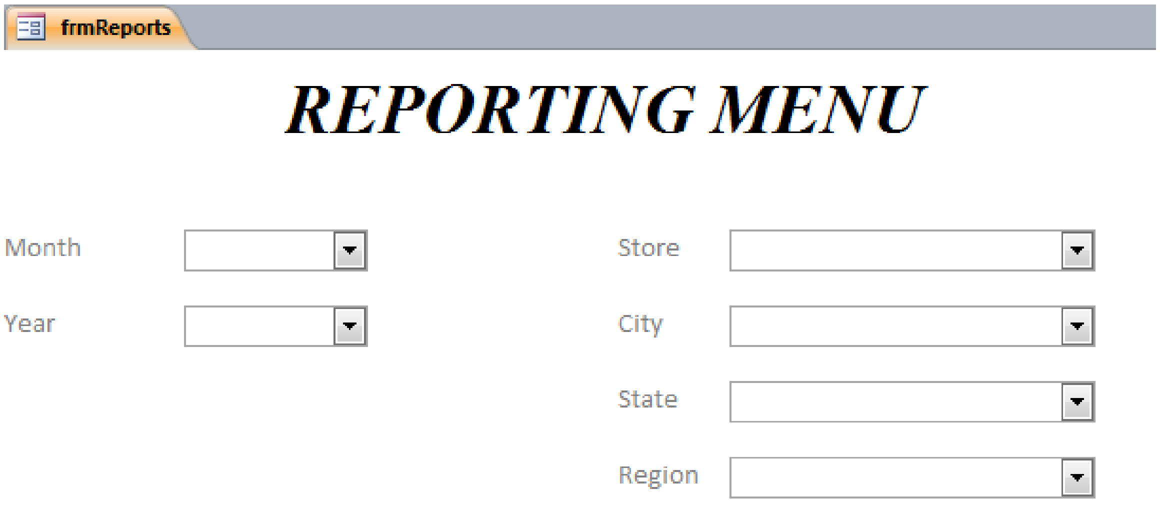


Figure 11.35

15. Save the form.

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 2010 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 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, log in and take the exam. Make sure that you take the exam on the same computer that you completed the sample files on, as some of the questions on the exam may refer to some of the completed examples.

ExcelCEO

Chief Excel Officer

Access 2010 and SQL

Complete Self-study Course

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 reports.

Advanced Reporting (Part II)

In Chapter 11, you built a very impressive income statement that you could be proud to show to any one 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 the* **frm_Reports** *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. *Save the form.*

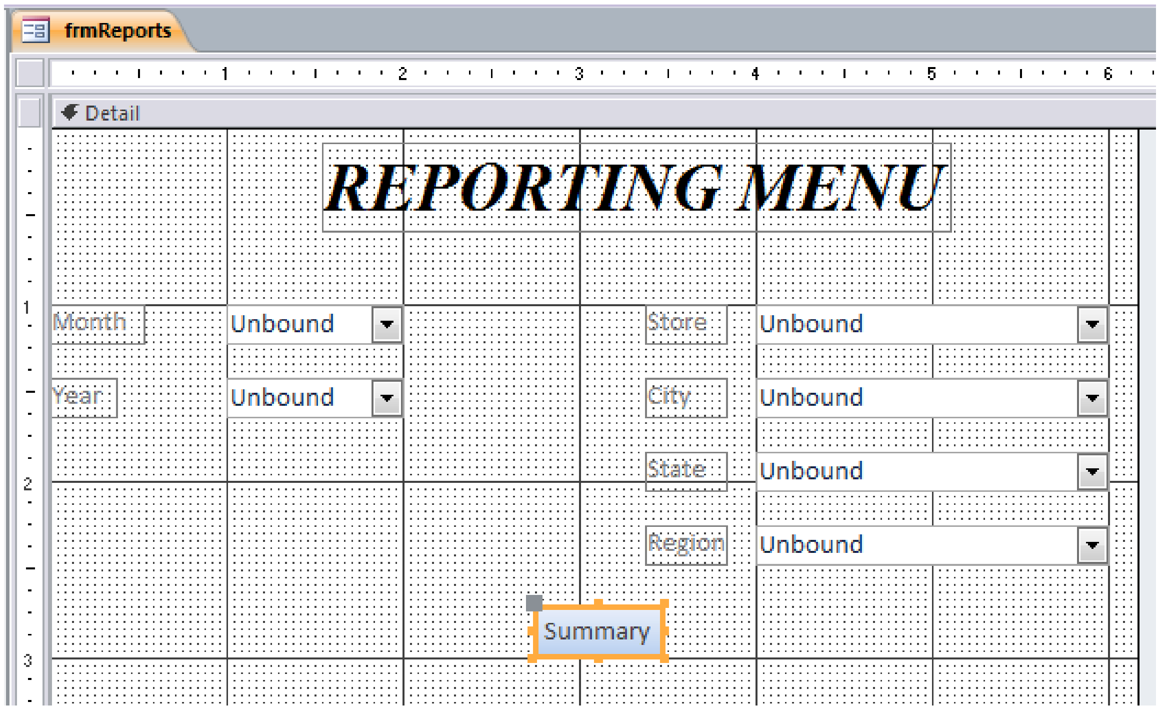
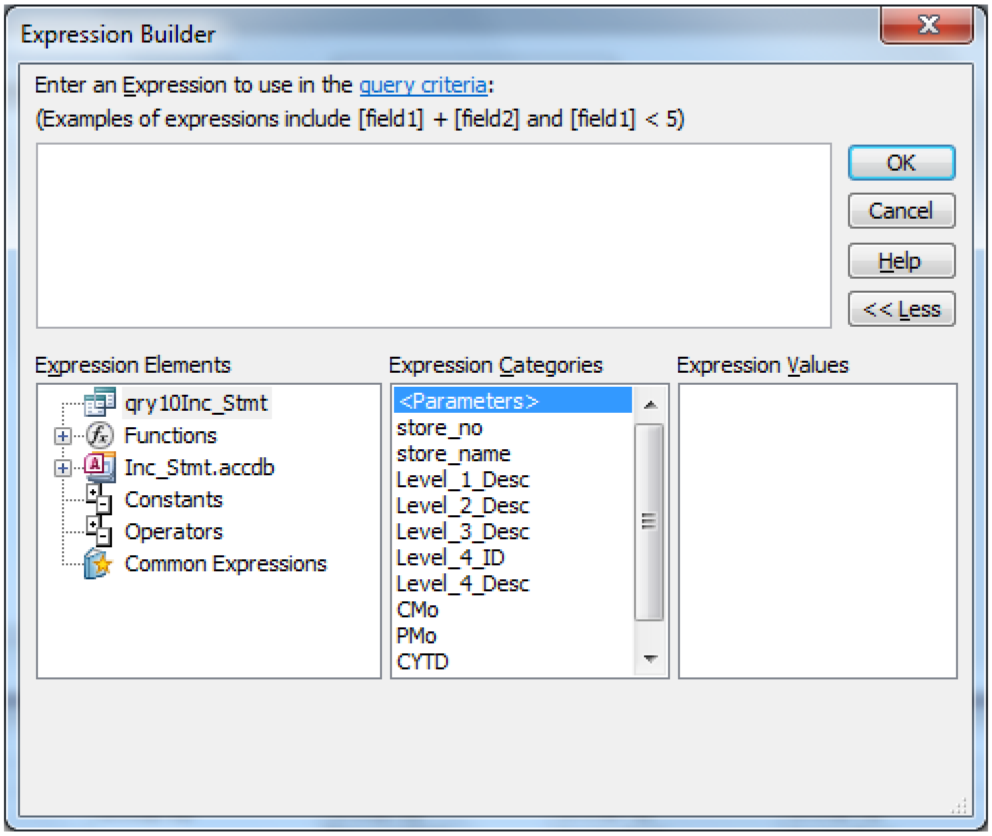


Figure 12.1

5. *In* **Form View**, *choose* **January 2010** *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.*

The Summary command button should run the report just fine, but with the variables changed to January 2010 and Store 1005, the report will still run only for March 2010, 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.

6. *Close the* **Print Preview** *of the report.*
7. *Open* **qry10Inc_Stmt** *in* **Design View**.
8. *Delete the* **"1032"** *criteria under* **Store_No**.
9. *While on the* **Store_No** *criteria, click the* **Builder** *icon in the* **Query Setup** *group.*

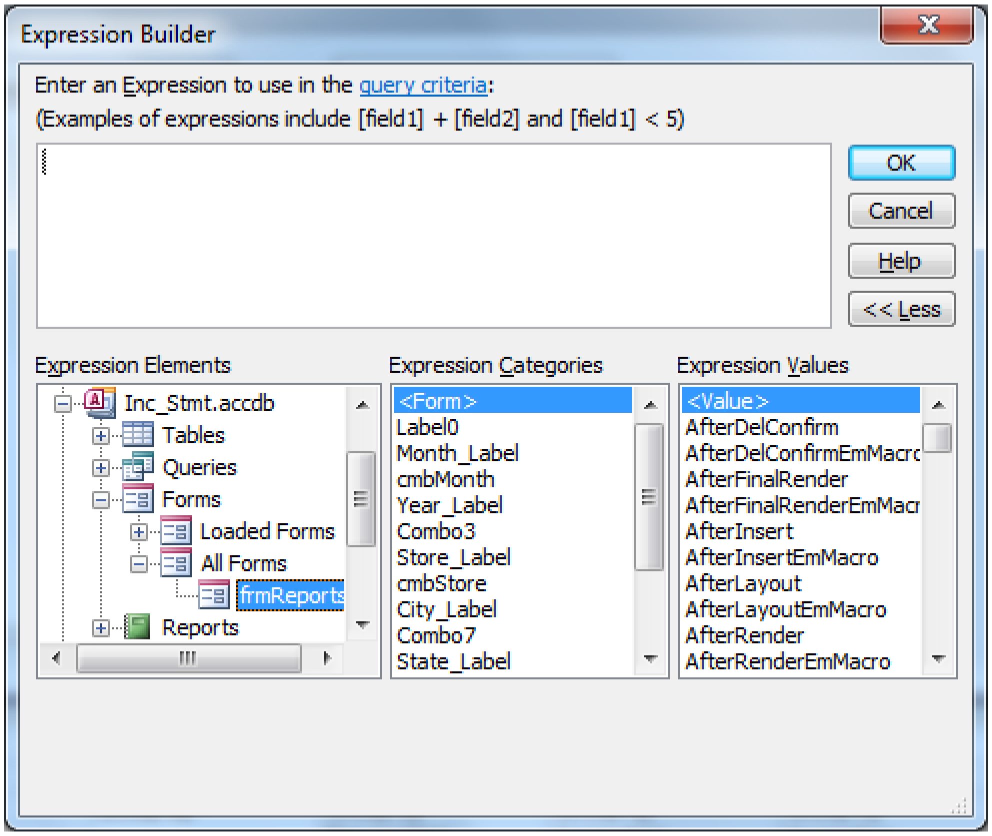

Expression Builder

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.

10. *Expand* **Inc_Stmt.accdb**, *then double-click on* **Forms**, *then double-click on* **All Forms**, *then click on* **frmReports**.



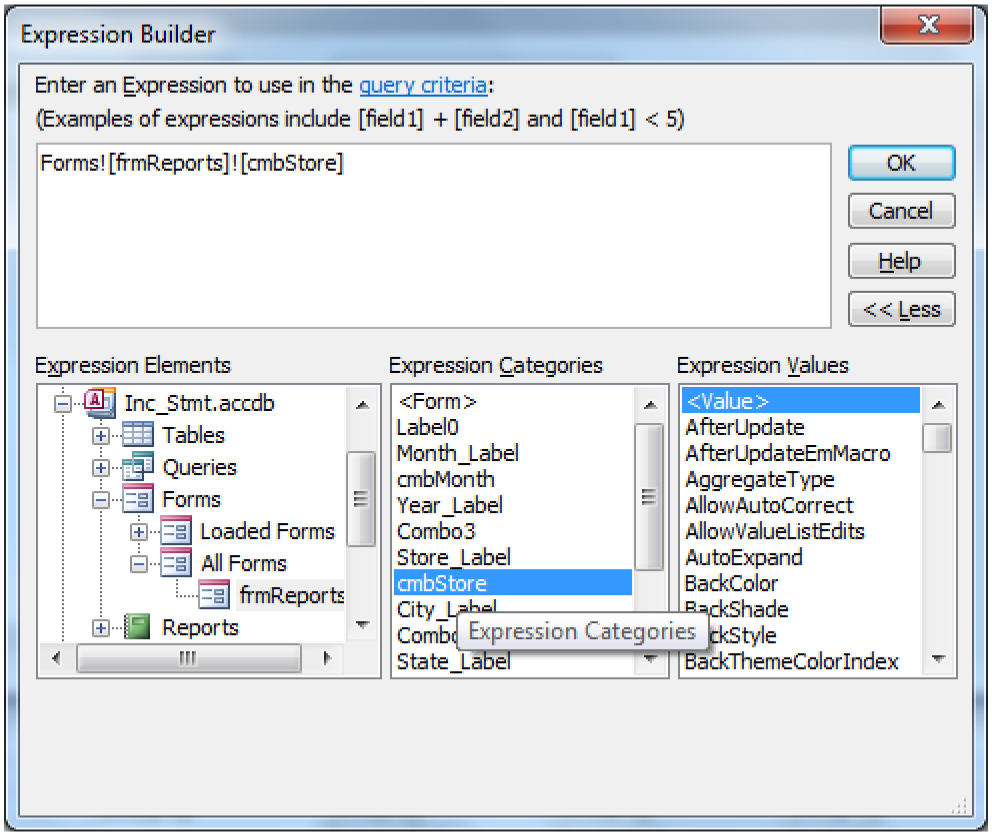
Enter an Expression to use in the query criteria:
(Examples of expressions include [field1] + [field2] and [field1] < 5)

Expression Elements	Expression Categories	Expression Values
Inc_Stmt.accdb	<Form>	<Value>
Tables	Label0	AfterDelConfirm
Queries	Month_Label	AfterDelConfirmEmMacro
Forms	cmbMonth	AfterFinalRender
Loaded Forms	Year_Label	AfterFinalRenderEmMacr
All Forms	Combo3	AfterInsert
frmReports	Store_Label	AfterInsertEmMacro
Reports	cmbStore	AfterLayout
	City_Label	AfterLayoutEmMacro
	Combo7	AfterRender
	State_Label	AfterRenderEmMacro

Figure 12.3

All of the objects in frmReports appear in the middle section under <Form>.

11. Double-click on **cmbStore** *in the* **Expression Categories** *section.*

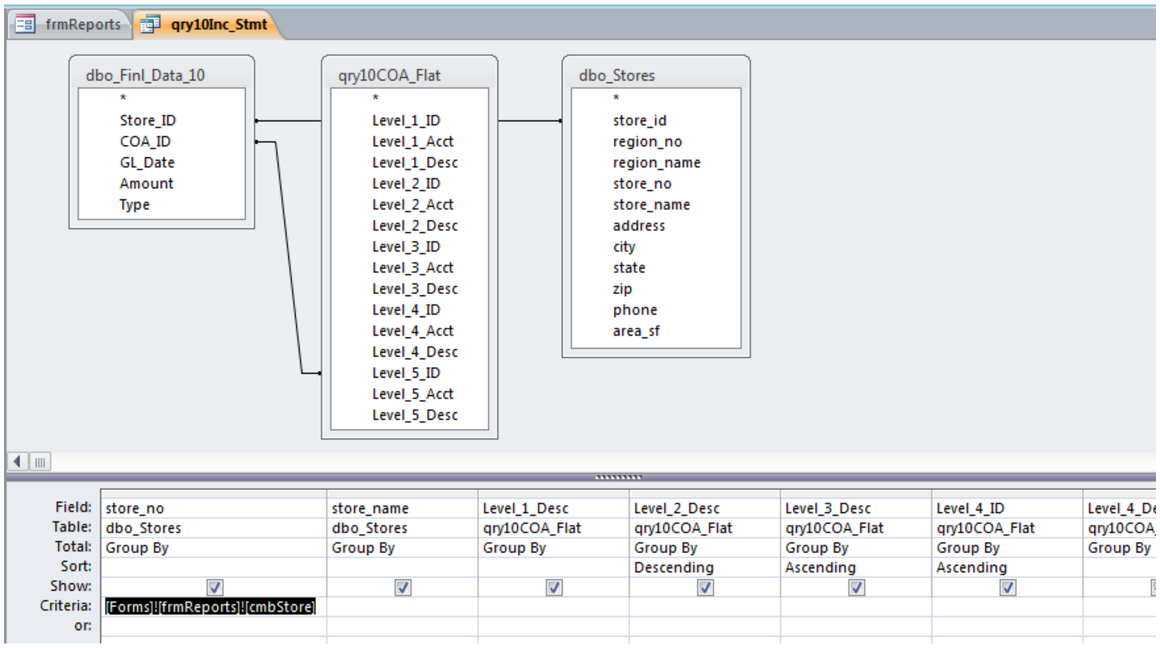


Forms![frmReports]![cmbStore]

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.

12. *Click* **OK**.

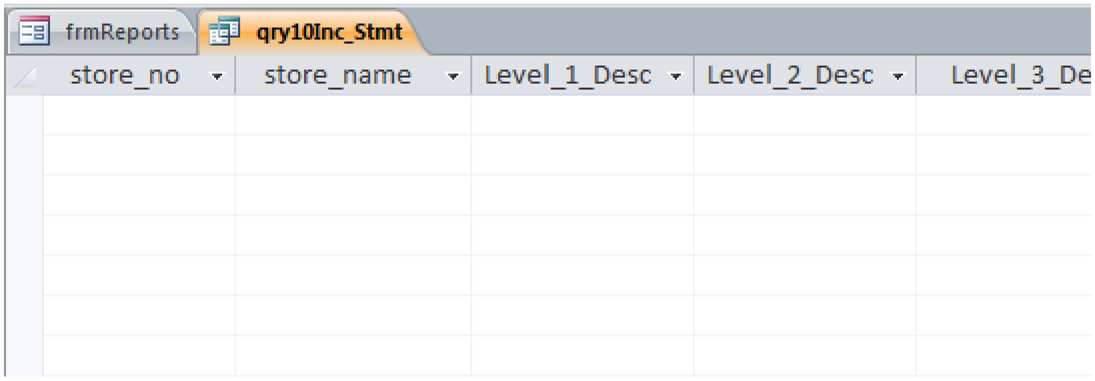


Field:	store_no	store_name	Level_1_Desc	Level_2_Desc	Level_3_Desc	Level_4_ID	Level_4_De
Table:	dbo_Stores	dbo_Stores	qry10COA_Flat	qry10COA_Flat	qry10COA_Flat	qry10COA_Flat	qry10COA
Total:	Group By	Group By	Group By	Group By	Group By	Group By	Group By
Sort:				Descending	Ascending	Ascending	
Show:	✓	✓	✓	✓	✓	✓	✓
Criteria:	[Forms]![frmReports]![cmbStore]						
or:							

Figure 12.5

The builder inserts the formula into the Criteria section of the Store_No field.

13. Run the query.



store_no	store_name	Level_1_Desc	Level_2_Desc	Level_3_De

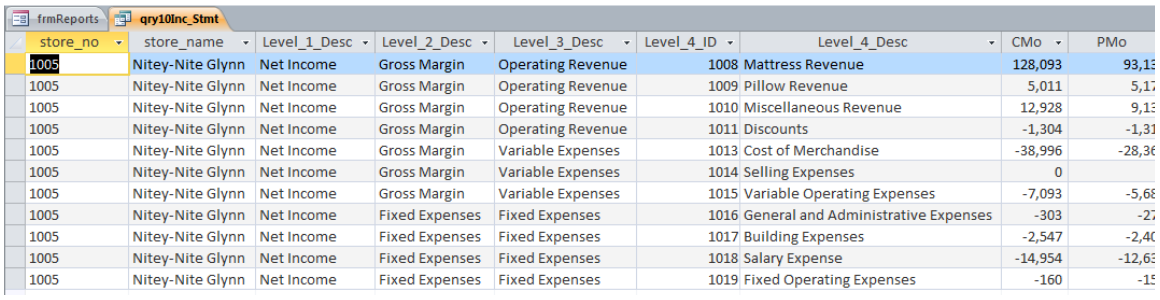
Figure 12.6

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.

14. 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)**

15. 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	CMo	PMo
1005	Nitey-Nite Glynn	Net Income	Gross Margin	Operating Revenue	1008	Mattress Revenue	128,093	93,13
1005	Nitey-Nite Glynn	Net Income	Gross Margin	Operating Revenue	1009	Pillow Revenue	5,011	5,17
1005	Nitey-Nite Glynn	Net Income	Gross Margin	Operating Revenue	1010	Miscellaneous Revenue	12,928	9,13
1005	Nitey-Nite Glynn	Net Income	Gross Margin	Operating Revenue	1011	Discounts	-1,304	-1,31
1005	Nitey-Nite Glynn	Net Income	Gross Margin	Variable Expenses	1013	Cost of Merchandise	-38,996	-28,36
1005	Nitey-Nite Glynn	Net Income	Gross Margin	Variable Expenses	1014	Selling Expenses	0	
1005	Nitey-Nite Glynn	Net Income	Gross Margin	Variable Expenses	1015	Variable Operating Expenses	-7,093	-5,68
1005	Nitey-Nite Glynn	Net Income	Fixed Expenses	Fixed Expenses	1016	General and Administrative Expenses	-303	-27
1005	Nitey-Nite Glynn	Net Income	Fixed Expenses	Fixed Expenses	1017	Building Expenses	-2,547	-2,40
1005	Nitey-Nite Glynn	Net Income	Fixed Expenses	Fixed Expenses	1018	Salary Expense	-14,954	-12,63
1005	Nitey-Nite Glynn	Net Income	Fixed Expenses	Fixed Expenses	1019	Fixed Operating Expenses	-160	-15

Figure 12.7

NOW it's working. If you look to the amount fields, you will see that it is returning the hard-coded years and months. Let's fix that. We'll work with the Year first.

16. *In the* **Design View** *of the* **qry10Inc_Stmt** *query, use the builder to choose the* **Year** *from the* **cmbYear** *combo box, replacing the* **2010** *reference in the* **Year** *criteria.*
17. *Replace* **2009** *with the same expression for* **year**, **minus one**.
18. *Change the* **Year** *combo box in the form to be* **2010**, *and save the form.*

The formula you have in the Year criteria should look like the following:

[Forms]![frmReports]![cmbYear] Or [Forms]![frmReports]![cmbYear]-1

Let's modify the Month criteria now.

19. *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]

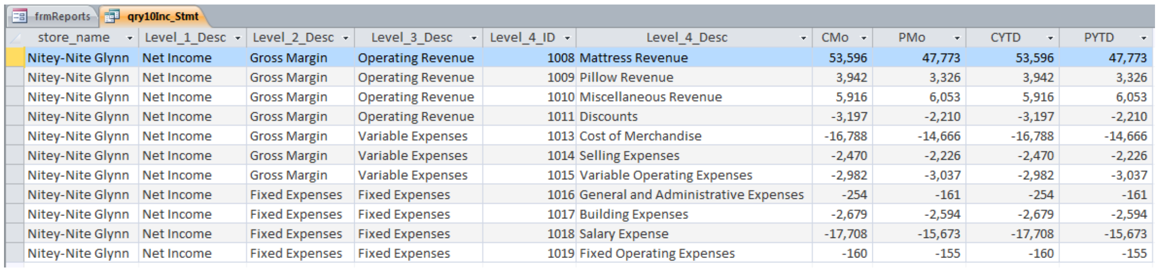
We have to treat the month field a little different, because in the YTD fields we will always choose from January (Month 1) through the month chosen.

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))**

20. Run the query.



frmReports | qry10Inc_Stmt

store_name	Level_1_Desc	Level_2_Desc	Level_3_Desc	Level_4_ID	Level_4_Desc	CMo	PMo	CYTD	PYTD
Nitey-Nite Glynn	Net Income	Gross Margin	Operating Revenue	1008	Mattress Revenue	53,596	47,773	53,596	47,773
Nitey-Nite Glynn	Net Income	Gross Margin	Operating Revenue	1009	Pillow Revenue	3,942	3,326	3,942	3,326
Nitey-Nite Glynn	Net Income	Gross Margin	Operating Revenue	1010	Miscellaneous Revenue	5,916	6,053	5,916	6,053
Nitey-Nite Glynn	Net Income	Gross Margin	Operating Revenue	1011	Discounts	-3,197	-2,210	-3,197	-2,210
Nitey-Nite Glynn	Net Income	Gross Margin	Variable Expenses	1013	Cost of Merchandise	-16,788	-14,666	-16,788	-14,666
Nitey-Nite Glynn	Net Income	Gross Margin	Variable Expenses	1014	Selling Expenses	-2,470	-2,226	-2,470	-2,226
Nitey-Nite Glynn	Net Income	Gross Margin	Variable Expenses	1015	Variable Operating Expenses	-2,982	-3,037	-2,982	-3,037
Nitey-Nite Glynn	Net Income	Fixed Expenses	Fixed Expenses	1016	General and Administrative Expenses	-254	-161	-254	-161
Nitey-Nite Glynn	Net Income	Fixed Expenses	Fixed Expenses	1017	Building Expenses	-2,679	-2,594	-2,679	-2,594
Nitey-Nite Glynn	Net Income	Fixed Expenses	Fixed Expenses	1018	Salary Expense	-17,708	-15,673	-17,708	-15,673
Nitey-Nite Glynn	Net Income	Fixed 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.

21. Save and close the query.

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 2010, 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 2010** *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 point**, **bold** *and* **italicized.**
4. *Make sure it is wide enough to display the results.*

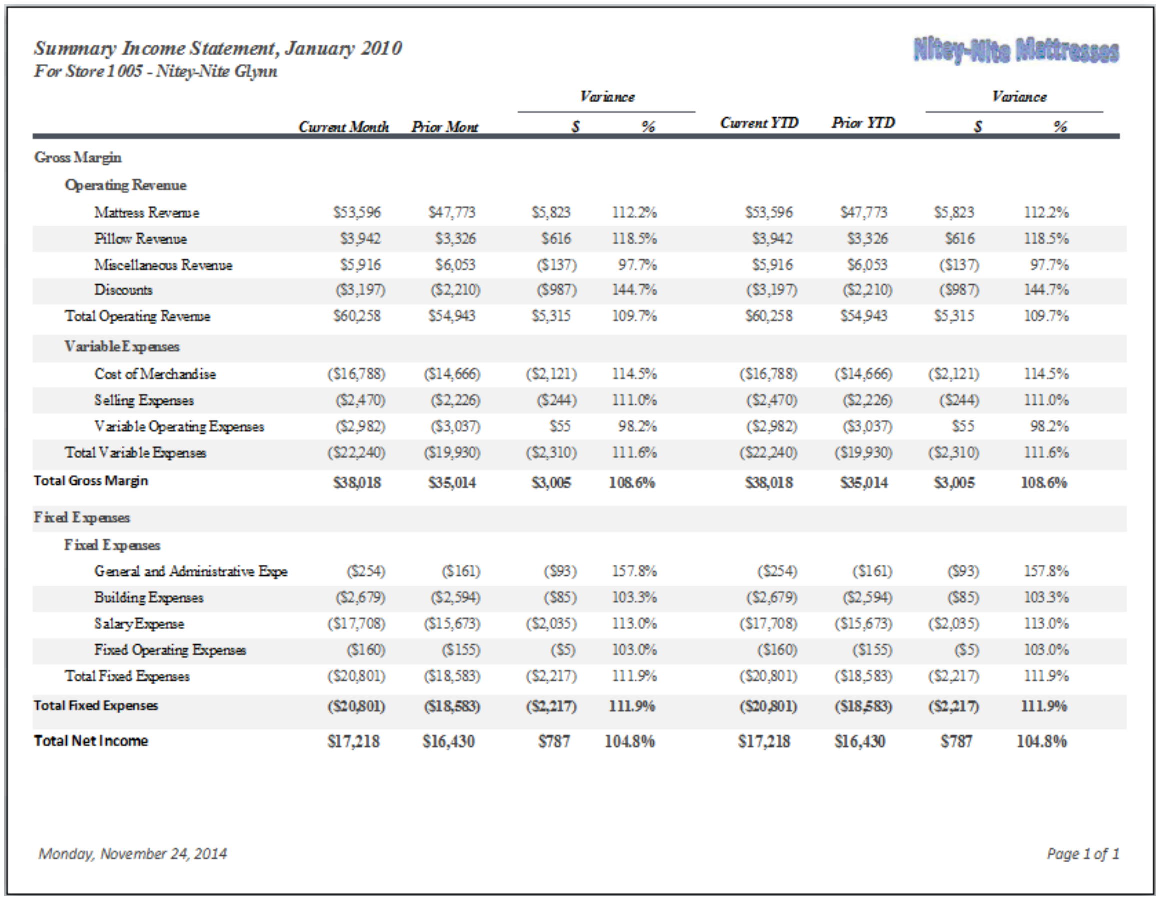
This formula will display the text "Summary Income Statement" followed by the name of the month and the year as chosen in the form.

5. *Copy the text box you just created and place the copy below the first text box.*

6. *Modify the copied text box's formula to be:* **="For Store " &[Forms]![frmReports]![cmbStore]**
7. *Format it as* **Times New Roman**, **12 point**, **bold** *and* **italicized**.

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**.

8. *Run the report from the form to make sure everything is working correctly.*



Summary Income Statement, January 2010
For Store 1005 - Nitey-Nite Glynn

Nitey-Nite Mattresses

	Current Month	Prior Mont	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	$3,942	$3,326	$616	118.5%	$3,942	$3,326	$616	118.5%
Miscellaneous Revenue	$5,916	$6,053	($137)	97.7%	$5,916	$6,053	($137)	97.7%
Discounts	($3,197)	($2,210)	($987)	144.7%	($3,197)	($2,210)	($987)	144.7%
Total Operating Revenue	$60,258	$54,943	$5,315	109.7%	$60,258	$54,943	$5,315	109.7%
Variable Expenses								
Cost of Merchandise	($16,788)	($14,666)	($2,121)	114.5%	($16,788)	($14,666)	($2,121)	114.5%
Selling Expenses	($2,470)	($2,226)	($244)	111.0%	($2,470)	($2,226)	($244)	111.0%
Variable Operating Expenses	($2,982)	($3,037)	$55	98.2%	($2,982)	($3,037)	$55	98.2%
Total Variable Expenses	($22,240)	($19,930)	($2,310)	111.6%	($22,240)	($19,930)	($2,310)	111.6%
Total Gross Margin	**$38,018**	**$35,014**	**$3,005**	**108.6%**	**$38,018**	**$35,014**	**$3,005**	**108.6%**
Fixed Expenses								
Fixed Expenses								
General and Administrative Expe	($254)	($161)	($93)	157.8%	($254)	($161)	($93)	157.8%
Building Expenses	($2,679)	($2,594)	($85)	103.3%	($2,679)	($2,594)	($85)	103.3%
Salary Expense	($17,708)	($15,673)	($2,035)	113.0%	($17,708)	($15,673)	($2,035)	113.0%
Fixed Operating Expenses	($160)	($155)	($5)	103.0%	($160)	($155)	($5)	103.0%
Total Fixed Expenses	($20,801)	($18,583)	($2,217)	111.9%	($20,801)	($18,583)	($2,217)	111.9%
Total Fixed Expenses	**($20,801)**	**($18,583)**	**($2,217)**	**111.9%**	**($20,801)**	**($18,583)**	**($2,217)**	**111.9%**
Total Net Income	**$17,218**	**$16,430**	**$787**	**104.8%**	**$17,218**	**$16,430**	**$787**	**104.8%**

Monday, November 24, 2014 *Page 1 of 1*

Figure 12.9

9. *Save and close the report.*

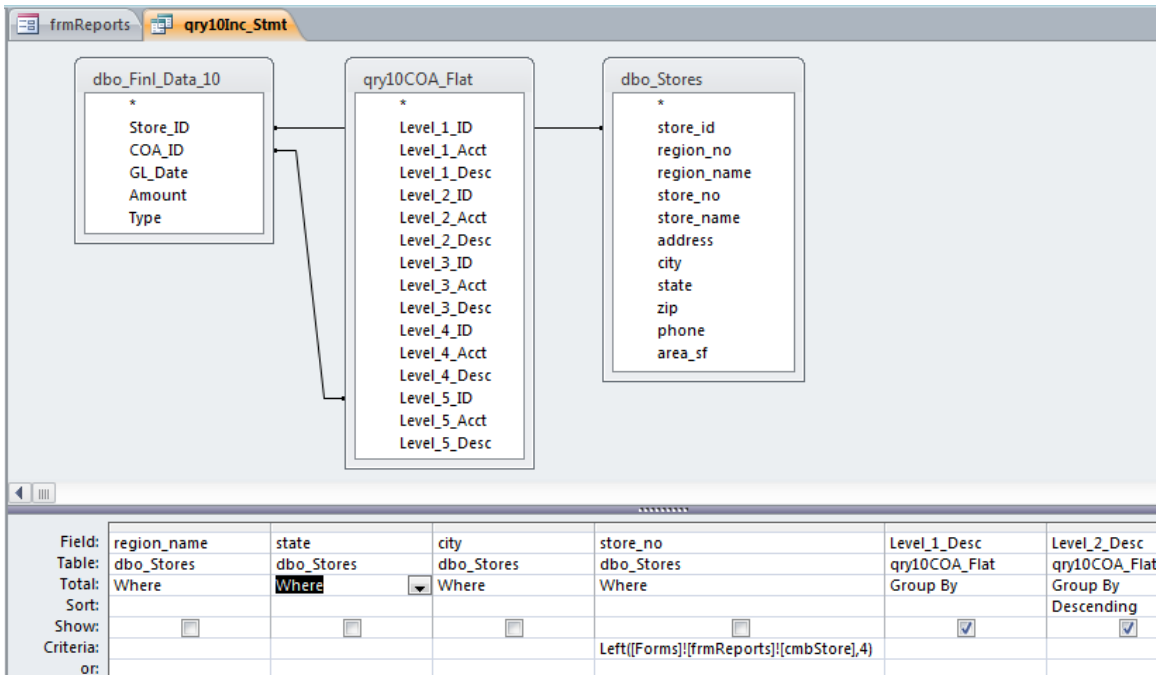
You show this database to your manager who absolutely loves it. He shows it to the Regional President of the Northern Region says, "*What if I want to see a rollup of all of the stores in my region? Also, I need to see the detail statement, so I can look at all of the individual accounts. Can you do that?*" You 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 area 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 try to 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.*



Field:	region_name	state	city	store_no	Level_1_Desc	Level_2_Desc
Table:	dbo_Stores	dbo_Stores	dbo_Stores	dbo_Stores	qry10COA_Flat	qry10COA_Flat
Total:	Where	Where	Where	Where	Group By	Group By
Sort:						Descending
Show:	☐	☐	☐	☐	☑	☑
Criteria:				Left([Forms]![frmReports]![cmbStore],4)		
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. *Do 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: " & [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.

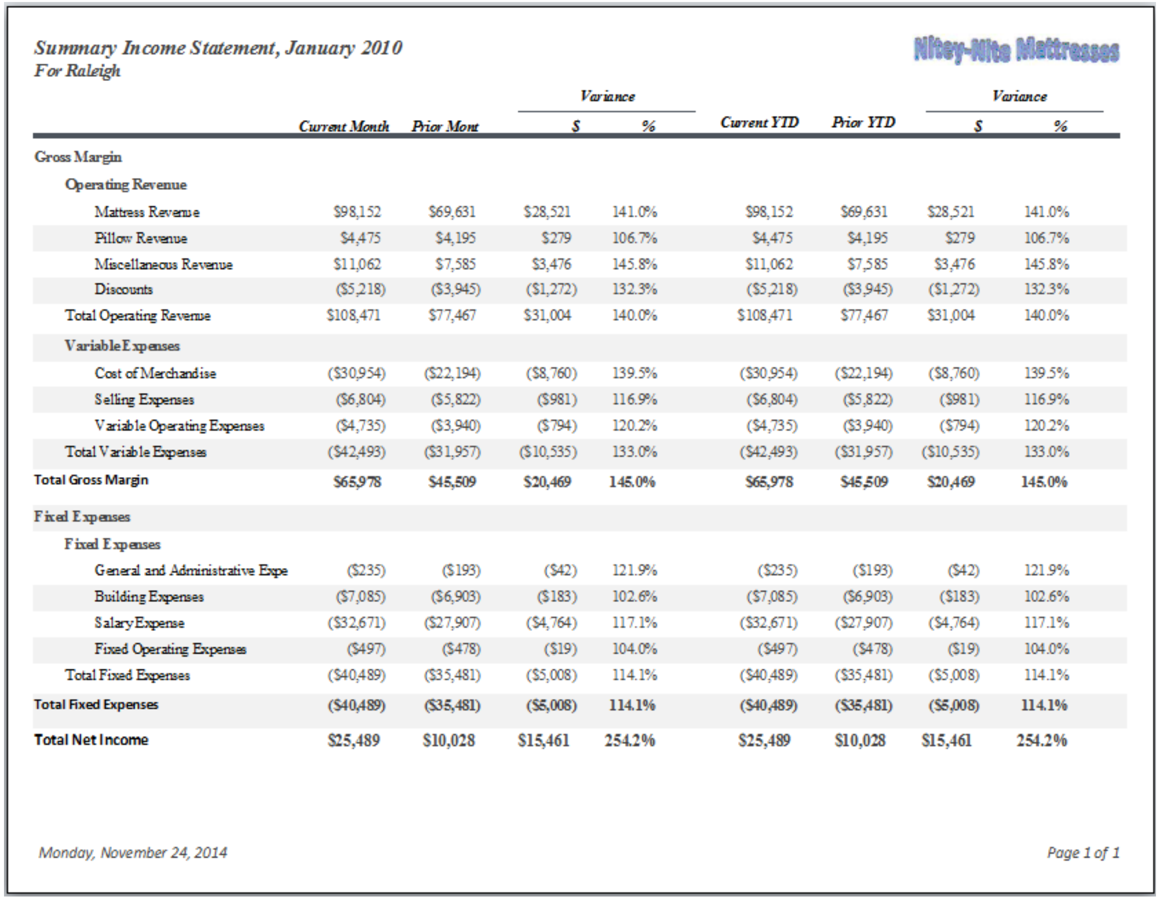
Next, you need to replace the formula in the text box in the report that contains the Store_No and Store_Name with a reference to the Report_Name field in the query.

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 report and the query.*

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 2010**.

    Note*: To delete a value in a combo box, simply select the value and press* **[Del].**



Summary Income Statement, January 2010
For Raleigh

Nitey-Nite Mattresses

	Current Month	Prior Mont	Variance $	Variance %	Current YTD	Prior YTD	Variance $	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 Expe	($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%**

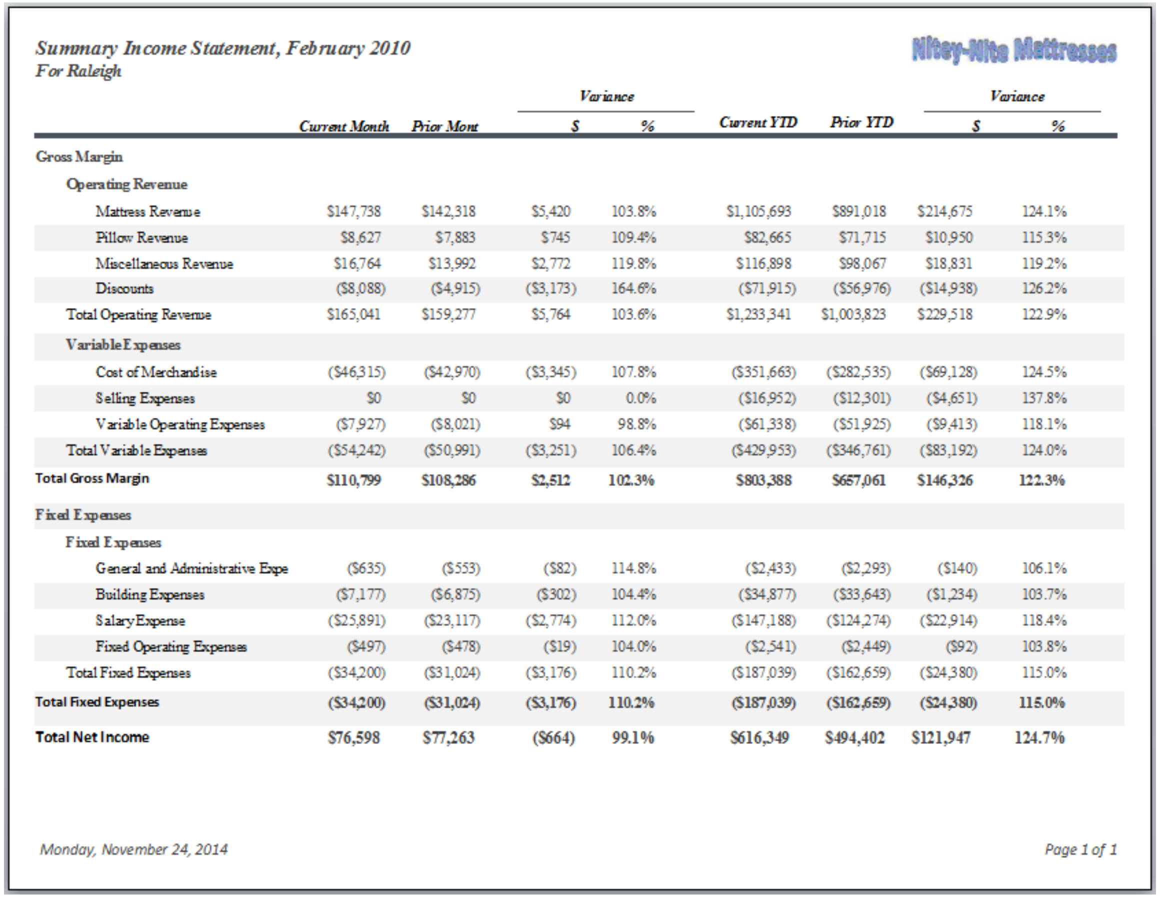
Monday, November 24, 2014 *Page 1 of 1*

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 2010** *for the city of* **Raleigh**.



Summary Income Statement, February 2010
For Raleigh

Nitey-Nite Mattresses

	Current Month	Prior Mont	Variance $	Variance %	Current YTD	Prior YTD	Variance $	Variance %
Gross Margin								
Operating Revenue								
Mattress Revenue	$147,738	$142,318	$5,420	103.8%	$1,105,693	$891,018	$214,675	124.1%
Pillow Revenue	$8,627	$7,883	$745	109.4%	$82,665	$71,715	$10,950	115.3%
Miscellaneous Revenue	$16,764	$13,992	$2,772	119.8%	$116,898	$98,067	$18,831	119.2%
Discounts	($8,088)	($4,915)	($3,173)	164.6%	($71,915)	($56,976)	($14,938)	126.2%
Total Operating Revenue	$165,041	$159,277	$5,764	103.6%	$1,233,341	$1,003,823	$229,518	122.9%
Variable Expenses								
Cost of Merchandise	($46,315)	($42,970)	($3,345)	107.8%	($351,663)	($282,535)	($69,128)	124.5%
Selling Expenses	$0	$0	$0	0.0%	($16,952)	($12,301)	($4,651)	137.8%
Variable Operating Expenses	($7,927)	($8,021)	$94	98.8%	($61,338)	($51,925)	($9,413)	118.1%
Total Variable Expenses	($54,242)	($50,991)	($3,251)	106.4%	($429,953)	($346,761)	($83,192)	124.0%
Total Gross Margin	**$110,799**	**$108,286**	**$2,512**	**102.3%**	**$803,388**	**$657,061**	**$146,326**	**122.3%**
Fixed Expenses								
Fixed Expenses								
General and Administrative Expe	($635)	($553)	($82)	114.8%	($2,433)	($2,293)	($140)	106.1%
Building Expenses	($7,177)	($6,875)	($302)	104.4%	($34,877)	($33,643)	($1,234)	103.7%
Salary Expense	($25,891)	($23,117)	($2,774)	112.0%	($147,188)	($124,274)	($22,914)	118.4%
Fixed Operating Expenses	($497)	($478)	($19)	104.0%	($2,541)	($2,449)	($92)	103.8%
Total Fixed Expenses	($34,200)	($31,024)	($3,176)	110.2%	($187,039)	($162,659)	($24,380)	115.0%
Total Fixed Expenses	**($34,200)**	**($31,024)**	**($3,176)**	**110.2%**	**($187,039)**	**($162,659)**	**($24,380)**	**115.0%**
Total Net Income	**$76,598**	**$77,263**	**($664)**	**99.1%**	**$616,349**	**$494,402**	**$121,947**	**124.7%**

Monday, November 24, 2014 *Page 1 of 1*

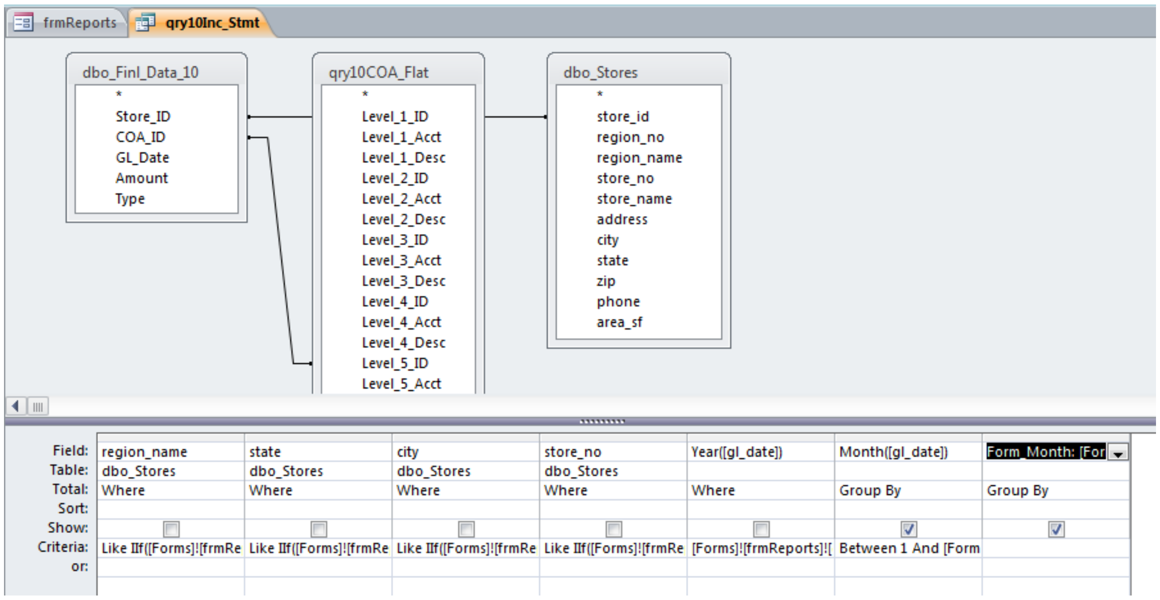
Figure 12.12

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 $616,349. 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 $102,087. Obviously something is wrong. It looks like the YTD numbers are adding in a lot more numbers than necessary. Let's audit our query to see if anything is amiss there.

14. Close the report 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.

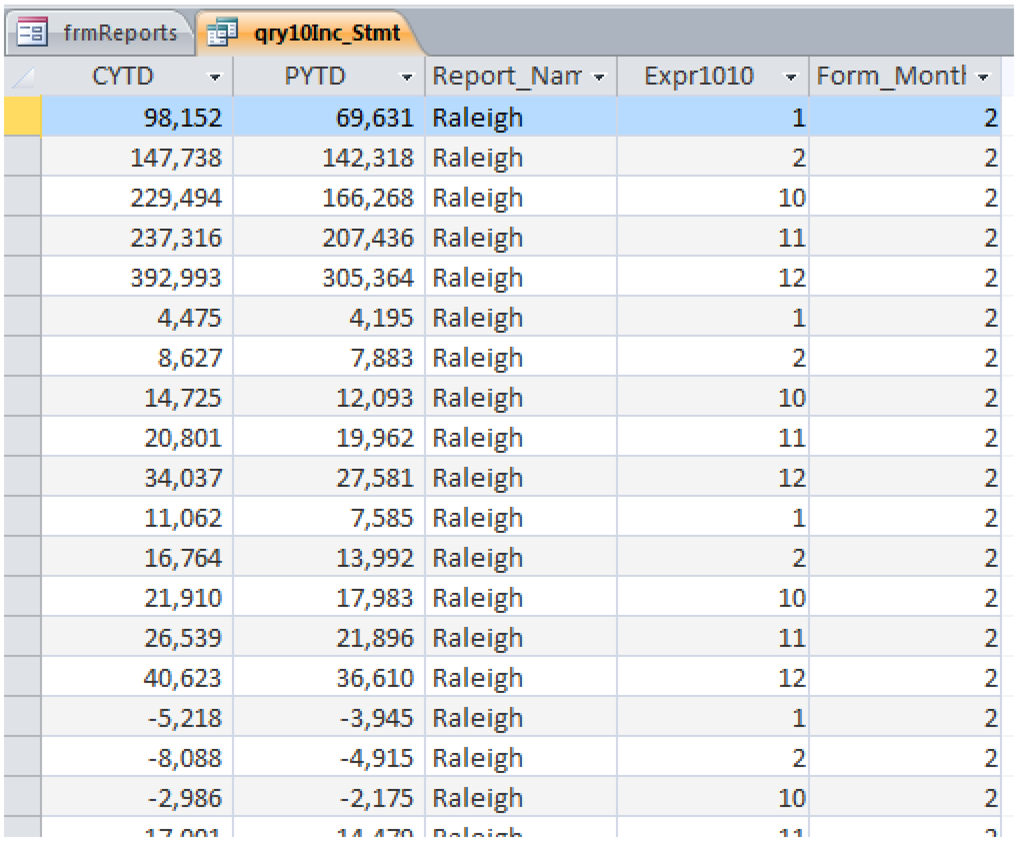
15. *In the* **Design View** *of the query, change the* **Month([gl_date])** *field* **Total** *line to* **Group By**, *check the* **Show** *box.*
16. *To the right of the* **Month([gl_date])** *field, insert a column (use the* **Insert Columns** *icon in the* **Query Setup** *group of the* **Design** *tab) and write the following formula:* **Form_Month: [Forms]![frmReports]![cmbMonth]**
17. *Run the query without saving it.*



Field:	region_name	state	city	store_no	Year([gl_date])	Month([gl_date])	Form_Month: [For
Table:	dbo_Stores	dbo_Stores	dbo_Stores	dbo_Stores			
Total:	Where	Where	Where	Where	Where	Group By	Group By
Sort:							
Show:	☐	☐	☐	☐	☐	✓	✓
Criteria:	Like IIf([Forms]![frmRe	Like IIf([Forms]![frmRe	Like IIf([Forms]![frmRe	Like IIf([Forms]![frmRe	[Forms]![frmReports]![	Between 1 And [Form	
or:							

Figure 12.13

You need to have the form still open with the variables set to February 2010 for the city of Raleigh for the query to run properly.



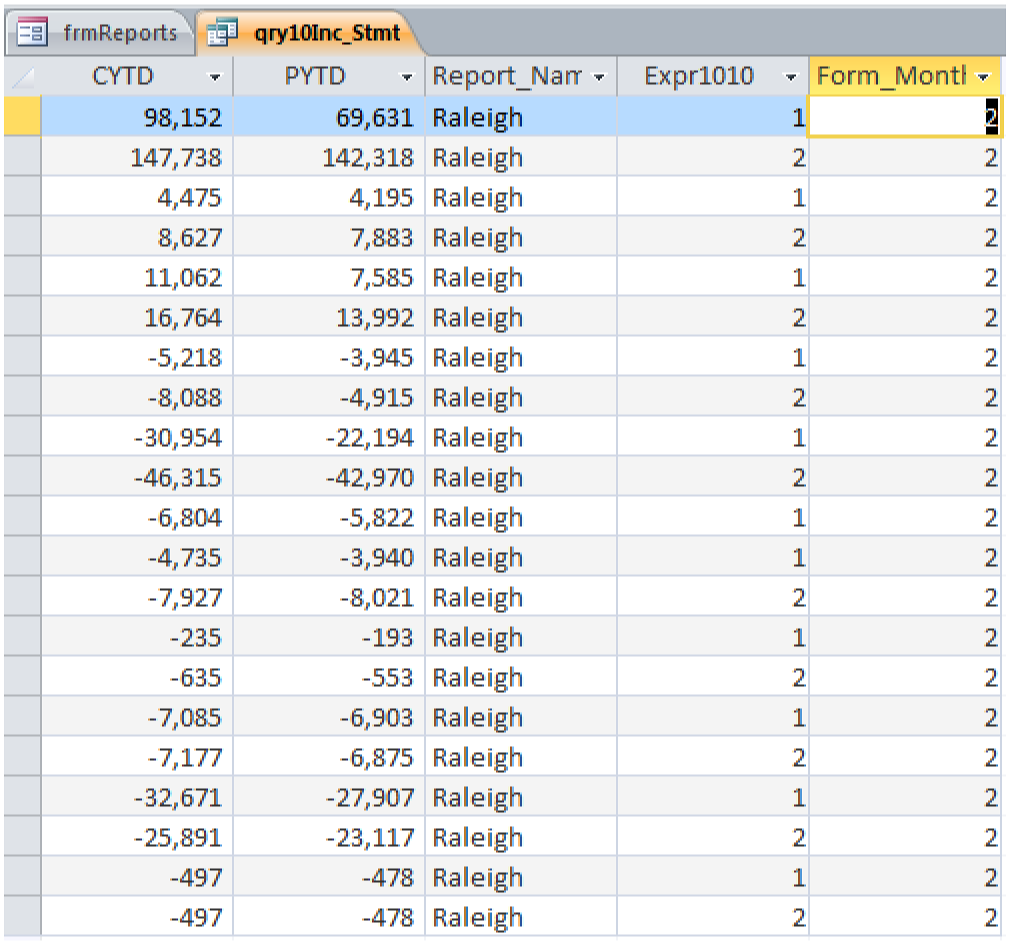
CYTD	PYTD	Report_Nar	Expr1010	Form_Montl
98,152	69,631	Raleigh	1	2
147,738	142,318	Raleigh	2	2
229,494	166,268	Raleigh	10	2
237,316	207,436	Raleigh	11	2
392,993	305,364	Raleigh	12	2
4,475	4,195	Raleigh	1	2
8,627	7,883	Raleigh	2	2
14,725	12,093	Raleigh	10	2
20,801	19,962	Raleigh	11	2
34,037	27,581	Raleigh	12	2
11,062	7,585	Raleigh	1	2
16,764	13,992	Raleigh	2	2
21,910	17,983	Raleigh	10	2
26,539	21,896	Raleigh	11	2
40,623	36,610	Raleigh	12	2
-5,218	-3,945	Raleigh	1	2
-8,088	-4,915	Raleigh	2	2
-2,986	-2,175	Raleigh	10	2

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? 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 of 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.*

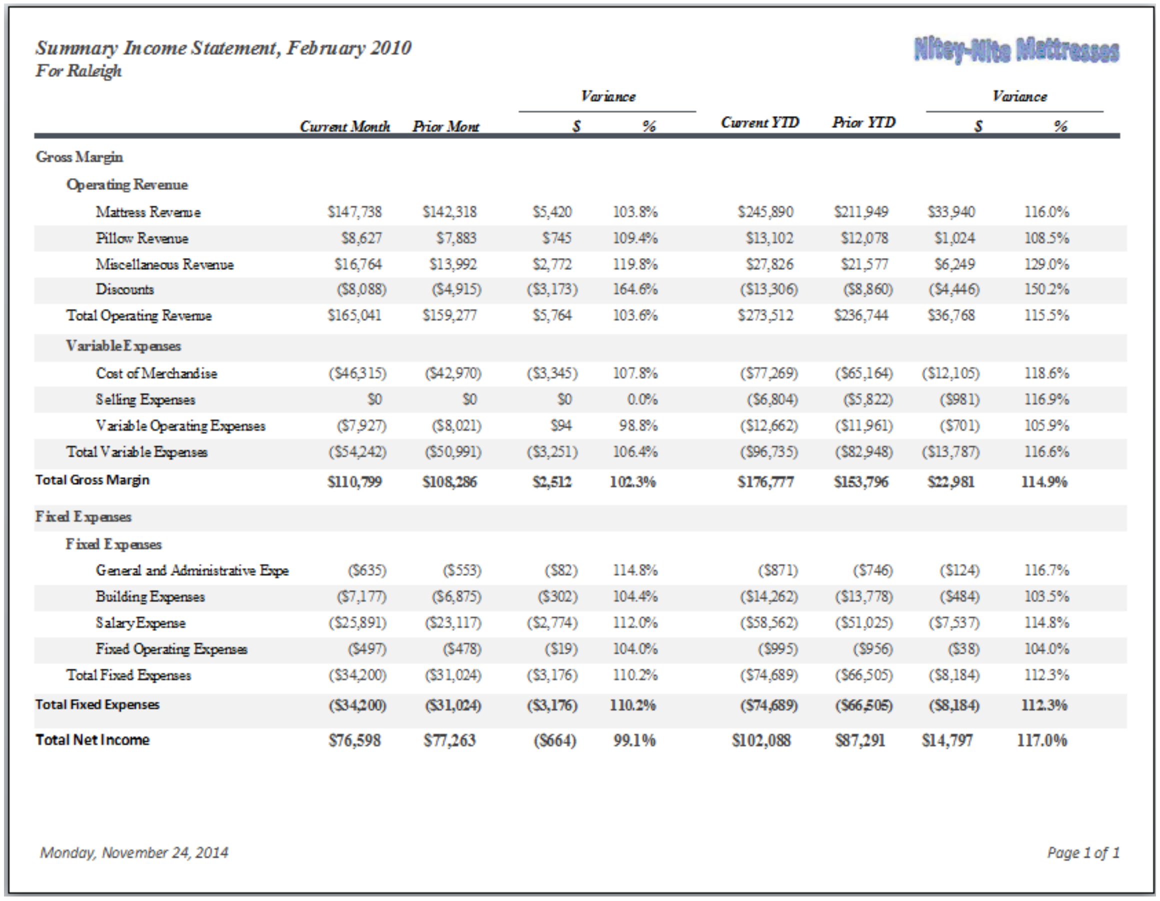


CYTD	PYTD	Report_Nam	Expr1010	Form_Montl
98,152	69,631	Raleigh	1	2
147,738	142,318	Raleigh	2	2
4,475	4,195	Raleigh	1	2
8,627	7,883	Raleigh	2	2
11,062	7,585	Raleigh	1	2
16,764	13,992	Raleigh	2	2
-5,218	-3,945	Raleigh	1	2
-8,088	-4,915	Raleigh	2	2
-30,954	-22,194	Raleigh	1	2
-46,315	-42,970	Raleigh	2	2
-6,804	-5,822	Raleigh	1	2
-4,735	-3,940	Raleigh	1	2
-7,927	-8,021	Raleigh	2	2
-235	-193	Raleigh	1	2
-635	-553	Raleigh	2	2
-7,085	-6,903	Raleigh	1	2
-7,177	-6,875	Raleigh	2	2
-32,671	-27,907	Raleigh	1	2
-25,891	-23,117	Raleigh	2	2
-497	-478	Raleigh	1	2
-497	-478	Raleigh	2	2

Figure 12.15

Woo hoo!!! 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 the query and run the report from the form again.*



Summary Income Statement, February 2010
For Raleigh

Nitey-Nite Mattresses

	Current Month	Prior Mont	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 Expe	($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%**

Monday, November 24, 2014 *Page 1 of 1*

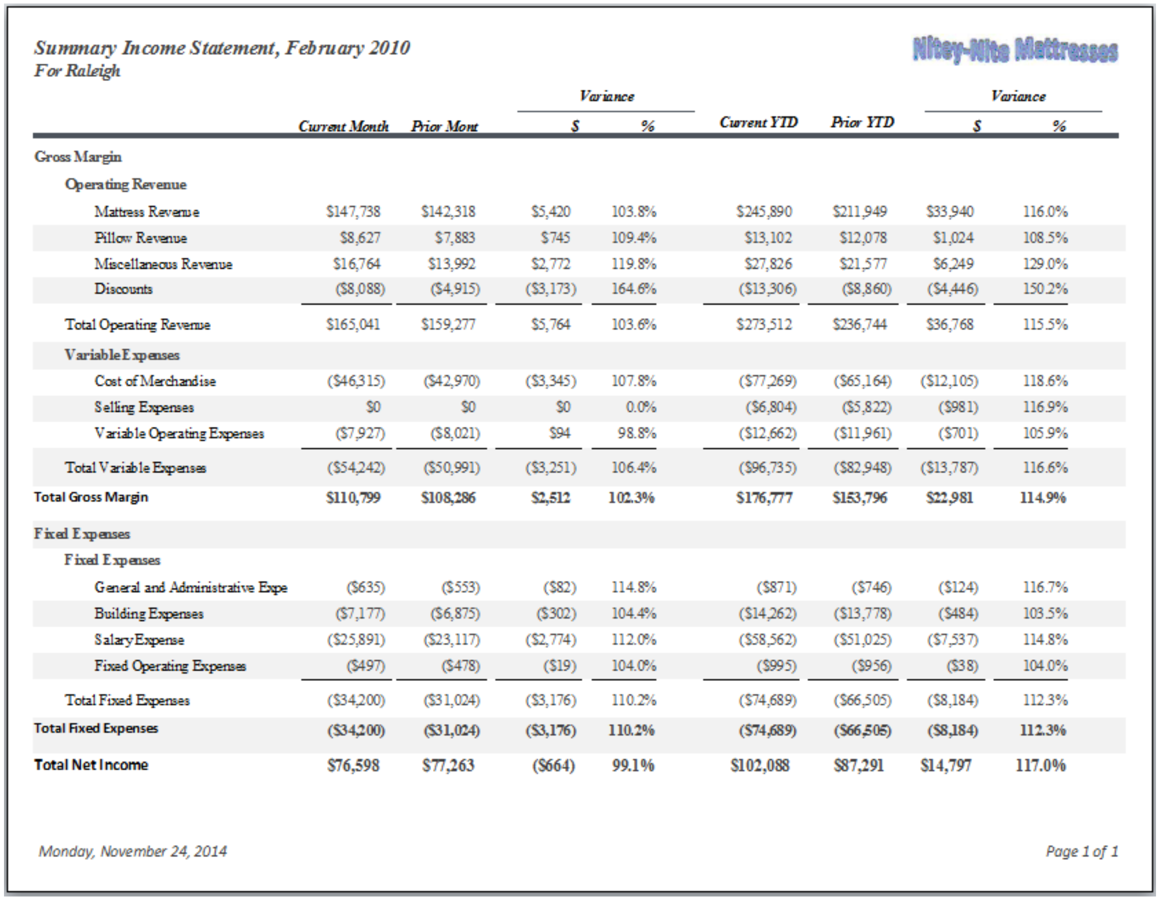
Figure 12.16

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 2010
For Raleigh

Nitey-Nite Mattresses

	Current Month	Prior Mont	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 Expe	($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%**

Monday, November 24, 2014 *Page 1 of 1*

Figure 12.17

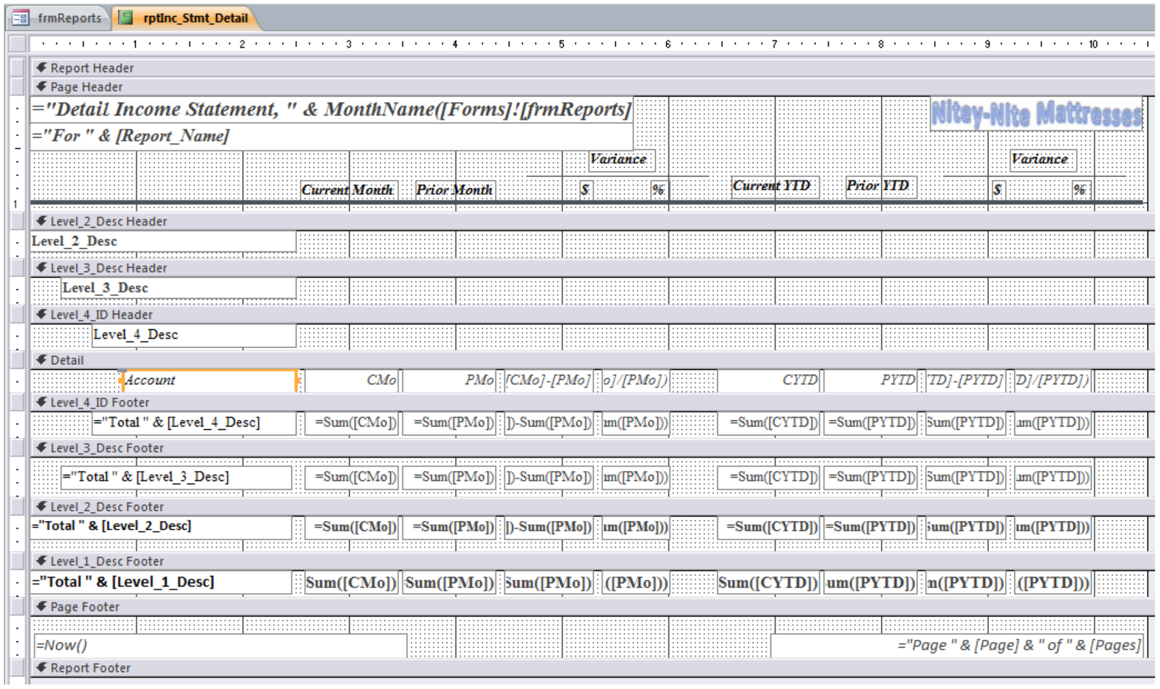
Create the Detail Statement

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 the report (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. *Makes sure the query works and save and close it.*
5. *Copy* **rptInc_Stmt** *to* **rptInc_Stmt_Detail**.
6. *In the report design of* **rptInc_Stmt_Detail,** *change record source to* **qry10Inc_Stmt_Detail** *(done in the* **Property Sheet***).*
7. *Expand the* **Detail** *section.*
8. *Expand the footer section for* **Level_4_ID**.
9. *Copy everything in the* **Level_4_ID Header** *section to be in the* **Level_4_ID Footer** *section and align appropriately.*
10. *Modify the* **Level_4_Desc** *text box in the* **Level_4_ID footer** *section to read* **=“Total ” &[Level_4_Desc]**
11. *Delete all text boxes in* **Level_4_ID Header** *section except for the* **Level_4_Desc** *text box.*
12. *Copy all objects in the* **Level_4_ID Footer** *section up to the* **Detail** *section and align appropriately.*
13. *Modify the* **Level_4_Desc** *text box in the* **Detail** *section to pull in the field called* **Account**.
14. *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.*
15. *Make everything in the* **Detail** *section italicized and indent the* **Account** *text box seven grid dots.*
16. *Change the formula that begins with “***Summary Income Statement***…”* *to “***Detail Income Statement***…”*
17. *Test, save, and close the report.*

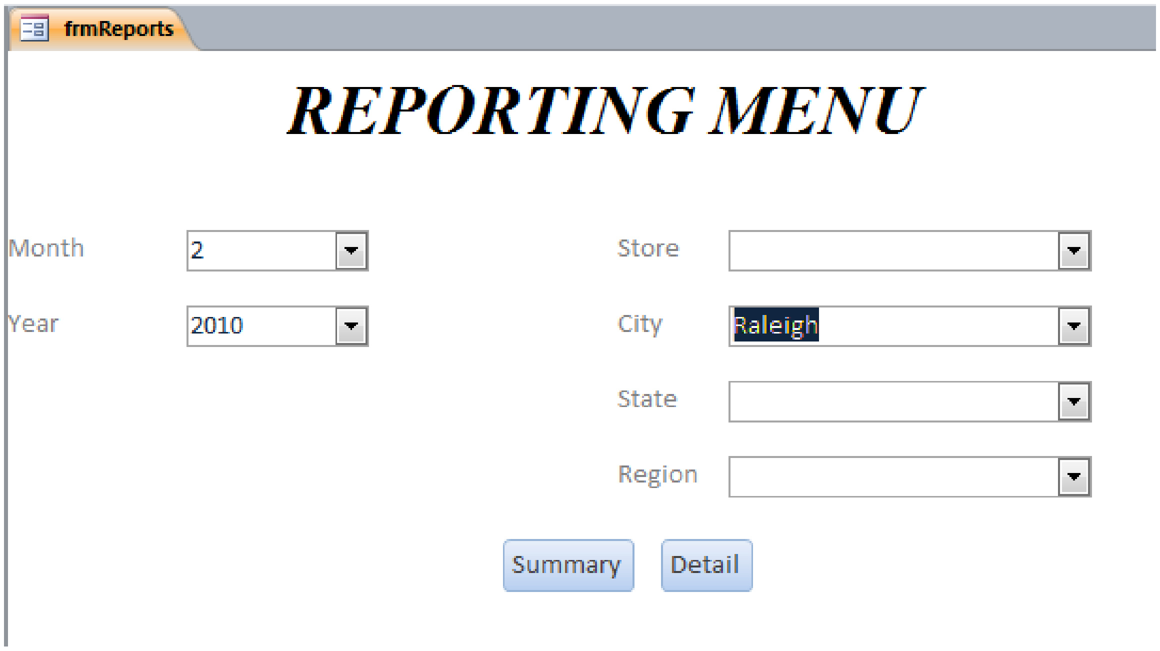


```
Page Header:
="Detail Income Statement, " & MonthName([Forms]![frmReports]
="For " & [Report_Name]
                          Variance                            Variance
Current Month  Prior Month  $  %    Current YTD  Prior YTD    $  %

Level_2_Desc Header:  Level_2_Desc
Level_3_Desc Header:  Level_3_Desc
Level_4_ID Header:    Level_4_Desc
Detail:  Account  CMo  PMo  [CMo]-[PMo]  o]/[PMo])  CYTD  PYTD  TD]-[PYTD]  D]/[PYTD])
Level_4_ID Footer:  ="Total " & [Level_4_Desc]  =Sum([CMo])  =Sum([PMo])  ])-Sum([PMo])  um([PMo]))  =Sum([CYTD])  =Sum([PYTD])  Sum([PYTD])  um([PYTD]))
Level_3_Desc Footer:  ="Total " & [Level_3_Desc]  =Sum([CMo])  =Sum([PMo])  ])-Sum([PMo])  um([PMo]))  =Sum([CYTD])  =Sum([PYTD])  Sum([PYTD])  um([PYTD]))
Level_2_Desc Footer:  ="Total " & [Level_2_Desc]  =Sum([CMo])  =Sum([PMo])  ])-Sum([PMo])  um([PMo]))  =Sum([CYTD])  =Sum([PYTD])  Sum([PYTD])  um([PYTD]))
Level_1_Desc Footer:  ="Total " & [Level_1_Desc]  Sum([CMo])  Sum([PMo])  Sum([PMo])  ([PMo]))  Sum([CYTD])  um([PYTD])  n([PYTD])  ([PYTD]))
Page Footer:  =Now()   ="Page " & [Page] & " of " & [Pages]
```

Figure 12.18

Now all that's left to do is to create a command button on the form to run the report.

18. *Open* **frmReports** *in* **Design View**.
19. *Create a command button called* **Detail** *that previews* **rptInc_Stmt_Detail**.
20. *Position both command buttons where they are in the middle of the form.*
21. *Test the* **Detail** *button's functionality.*
22. *Save and close the form.*


frmReports

REPORTING MENU

Month	2	Store	
Year	2010	City	Raleigh
		State	
		Region	

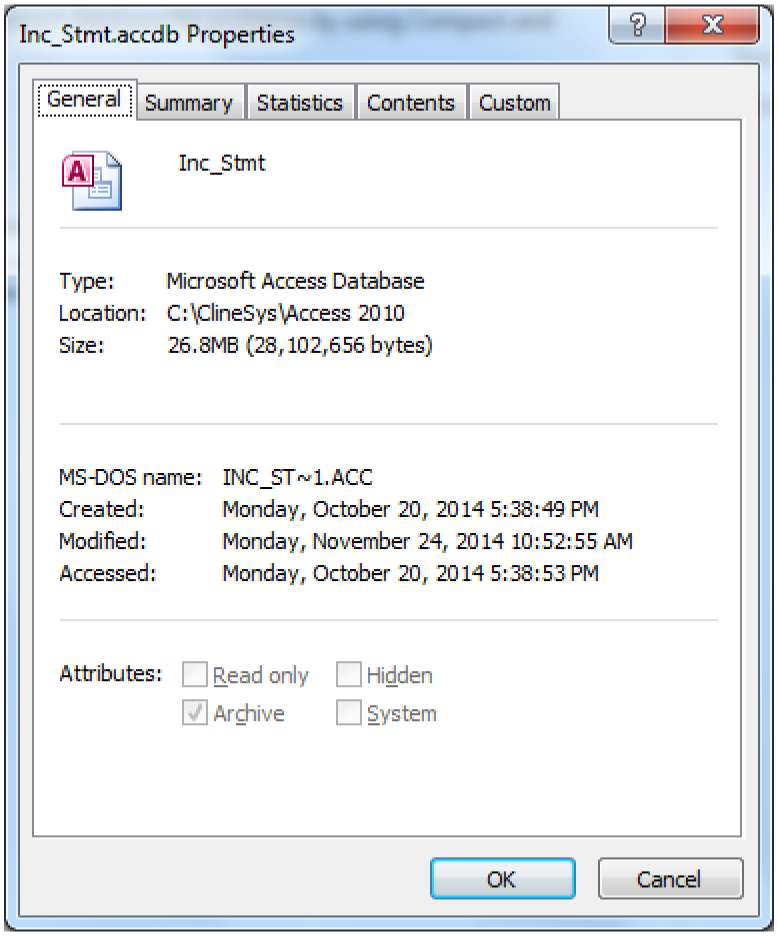
Figure 12.19

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.*



Inc_Stmt

Type: Microsoft Access Database
Location: C:\ClineSys\Access 2010
Size: 26.8MB (28,102,656 bytes)

MS-DOS name: INC_ST~1.ACC
Created: Monday, October 20, 2014 5:38:49 PM
Modified: Monday, November 24, 2014 10:52:55 AM
Accessed: Monday, October 20, 2014 5:38:53 PM

Attributes: ☐ Read only ☐ Hidden ✓ Archive ☐ System

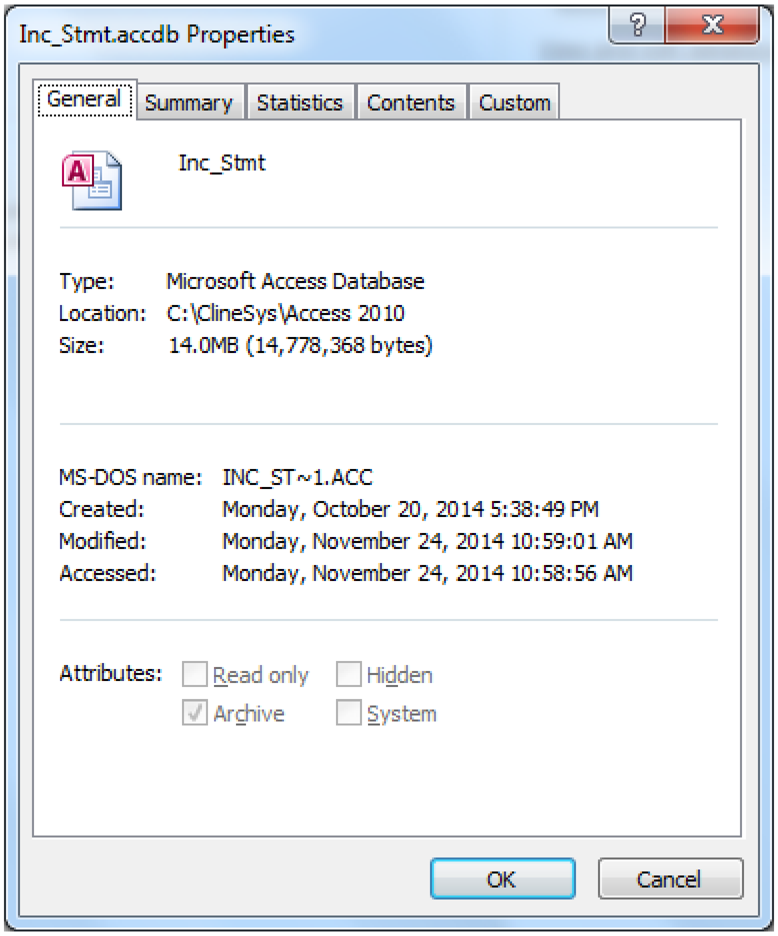
Figure 12.20

The Inc_Stmt Properties dialog box shows that my database contains 26.8 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.*



Inc_Stmt

Type: Microsoft Access Database
Location: C:\ClineSys\Access 2010
Size: 14.0MB (14,778,368 bytes)

MS-DOS name: INC_ST~1.ACC
Created: Monday, October 20, 2014 5:38:49 PM
Modified: Monday, November 24, 2014 10:59:01 AM
Accessed: Monday, November 24, 2014 10:58:56 AM

Attributes: Read only [ ] Hidden [ ] Archive [✓] System [ ]

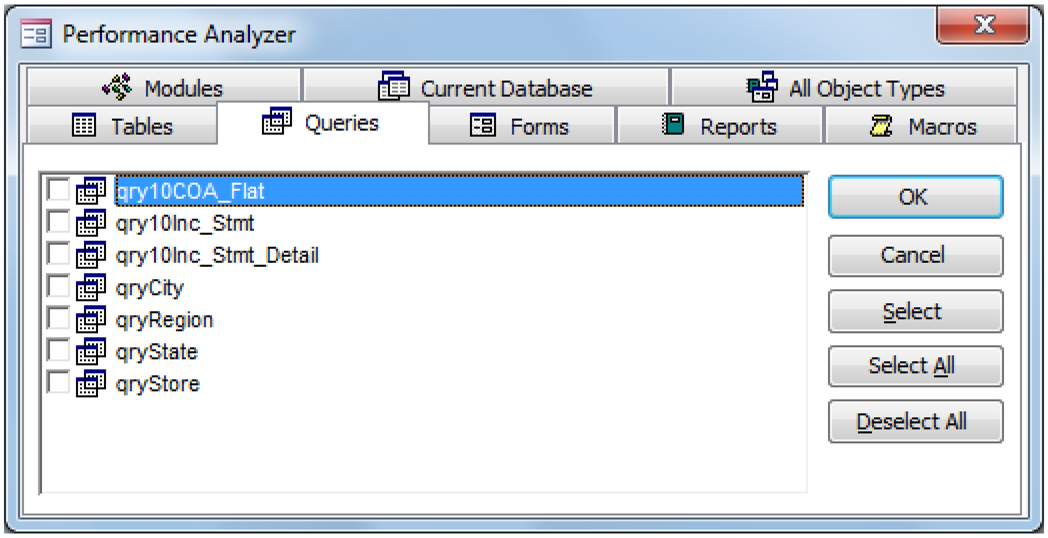
Figure 12.21

The database now has about 14MB instead of 28MB. 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.

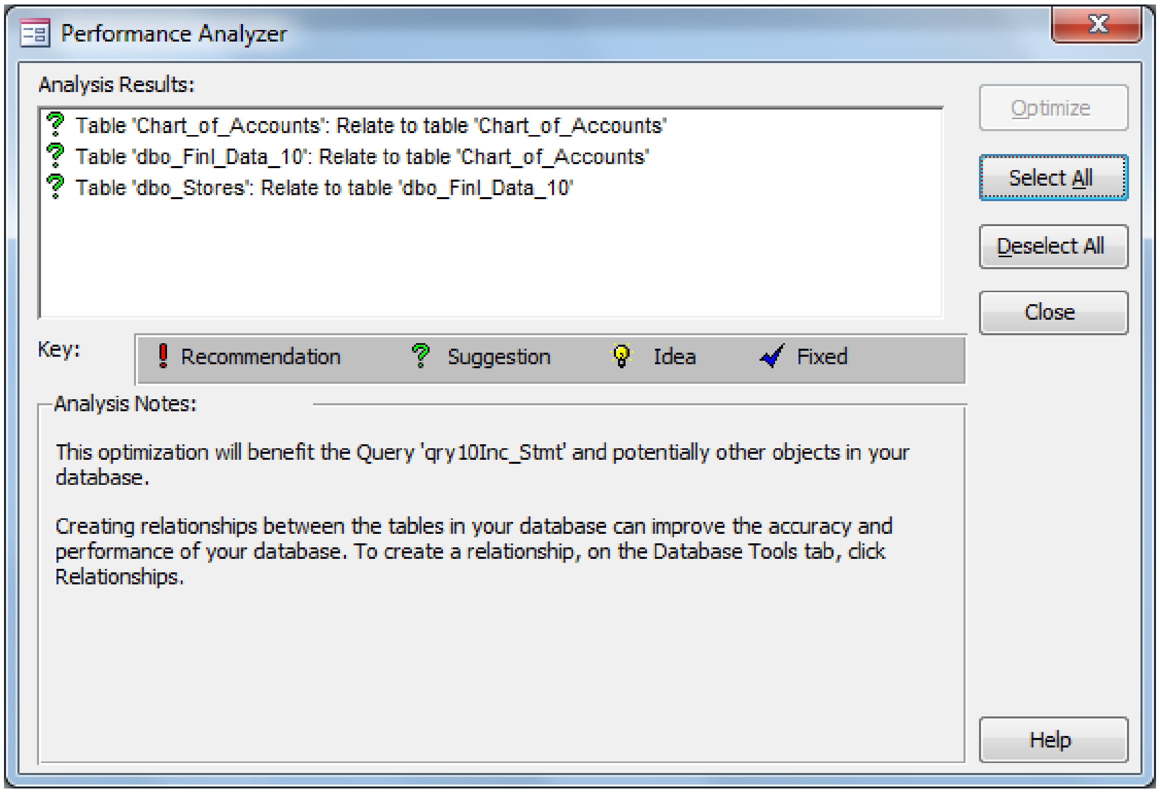
1. *Close the* **Inc_Stmt.accdb Properties** *dialog box.*
2. *Click on the* **Database Tools** *tab then on* **qry10Inc_Stmt**.
3. *Click on the* **Analyze Performance** *icon in the* **Analyze** *group.*



- ☐ qry10COA_Flat
- ☐ qry10Inc_Stmt
- ☐ qry10Inc_Stmt_Detail
- ☐ qryCity
- ☐ qryRegion
- ☐ qryState
- ☐ qryStore

Figure 12.22

4. *Check the box next to* **qry10Inc_Stmt** *and click* **OK**.



Analysis Results:

- ? Table 'Chart_of_Accounts': Relate to table 'Chart_of_Accounts'
- ? Table 'dbo_Finl_Data_10': Relate to table 'Chart_of_Accounts'
- ? Table 'dbo_Stores': Relate to table 'dbo_Finl_Data_10'

Analysis Notes:

This optimization will benefit the Query 'qry10Inc_Stmt' and potentially other objects in your database.

Creating relationships between the tables in your database can improve the accuracy and performance of your database. To create a relationship, on the Database Tools tab, click Relationships.

Figure 12.23

The only suggestions 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.

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** *icon in the* **Analyze** *group of the* **Database Tools** *tab.*

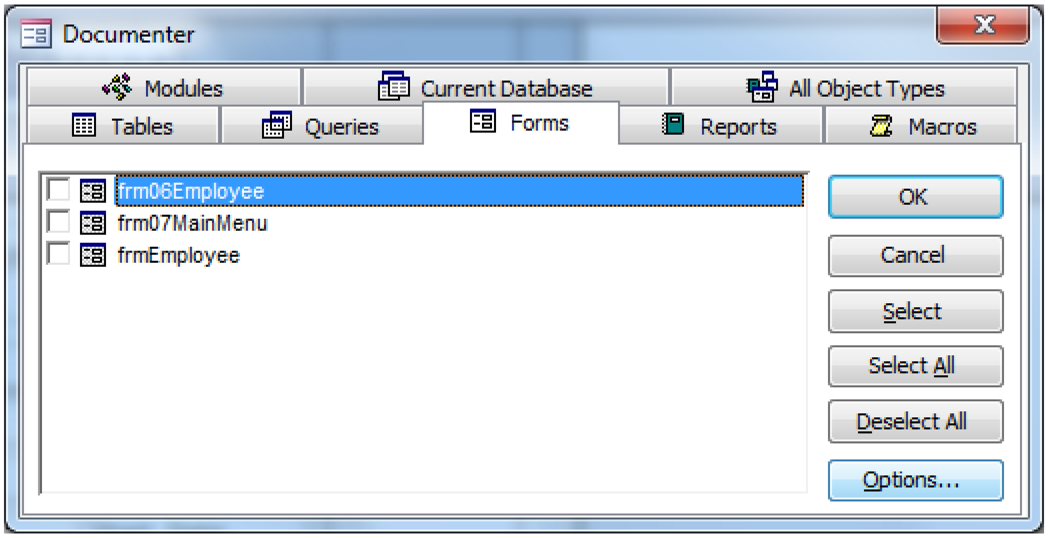
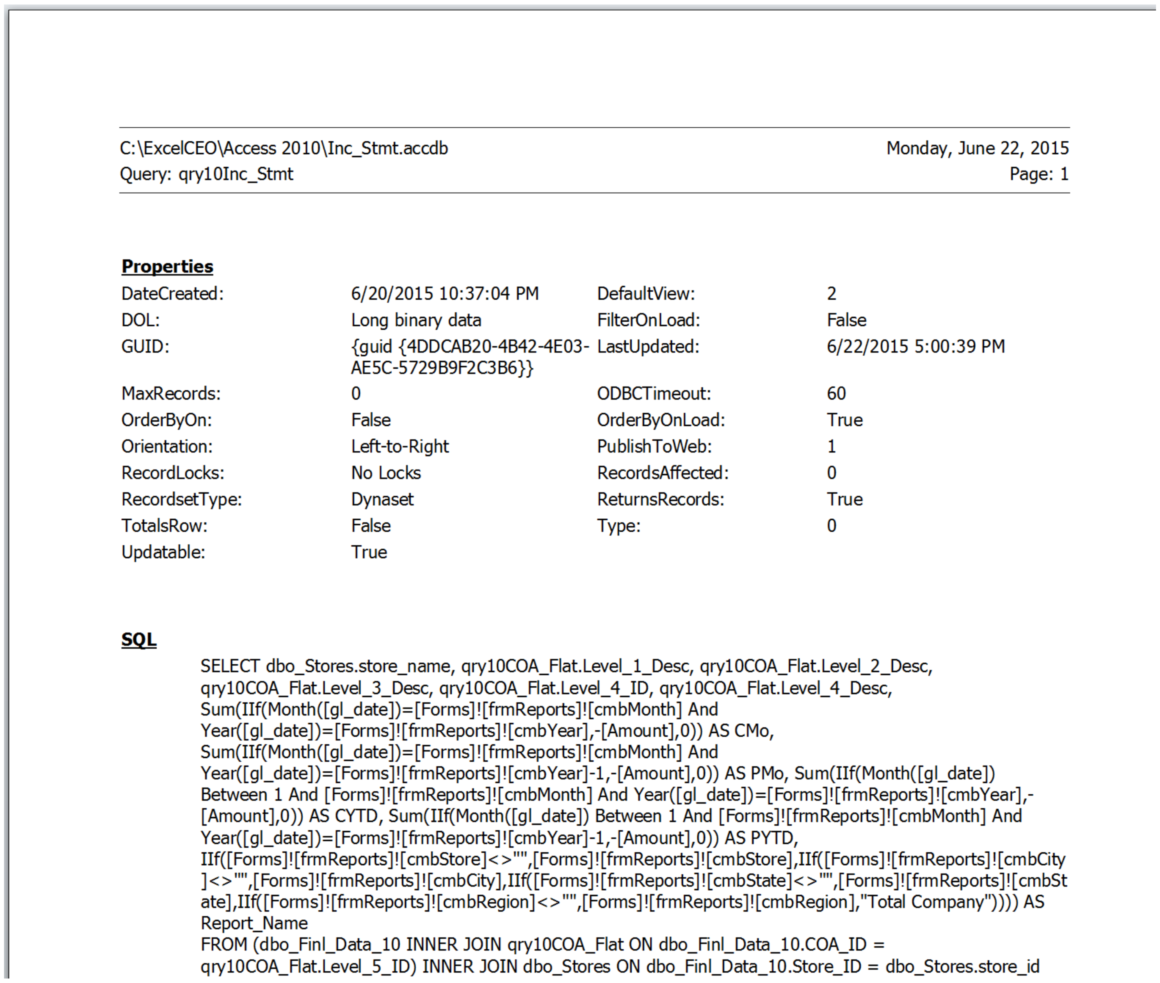


Figure 12.24

2. *Click on* **qry10Inc_Stmt** *and click* **OK**.



C:\ExcelCEO\Access 2010\Inc_Stmt.accdb — Monday, June 22, 2015
Query: qry10Inc_Stmt — Page: 1

Properties

DateCreated:	6/20/2015 10:37:04 PM	DefaultView:	2
DOL:	Long binary data	FilterOnLoad:	False
GUID:	{guid {4DDCAB20-4B42-4E03-AE5C-5729B9F2C3B6}}	LastUpdated:	6/22/2015 5:00:39 PM
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 dbo_Stores.store_name, 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
Year([gl_date])=[Forms]![frmReports]![cmbYear]-1,-[Amount],0)) AS PMo, Sum(IIf(Month([gl_date])
Between 1 And [Forms]![frmReports]![cmbMonth] And Year([gl_date])=[Forms]![frmReports]![cmbYear],-
[Amount],0)) AS CYTD, Sum(IIf(Month([gl_date]) Between 1 And [Forms]![frmReports]![cmbMonth] And
Year([gl_date])=[Forms]![frmReports]![cmbYear]-1,-[Amount],0)) AS PYTD,
IIf([Forms]![frmReports]![cmbStore]<>"",[Forms]![frmReports]![cmbStore],IIf([Forms]![frmReports]![cmbCity
]<>"",[Forms]![frmReports]![cmbCity],IIf([Forms]![frmReports]![cmbState]<>"",[Forms]![frmReports]![cmbSt
ate],IIf([Forms]![frmReports]![cmbRegion]<>"",[Forms]![frmReports]![cmbRegion],"Total Company")))) AS
Report_Name
FROM (dbo_Finl_Data_10 INNER JOIN qry10COA_Flat ON dbo_Finl_Data_10.COA_ID =
qry10COA_Flat.Level_5_ID) INNER JOIN dbo_Stores ON dbo_Finl_Data_10.Store_ID = dbo_Stores.store_id
```

Figure 12.25

It returns a six page report 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 2010 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.

Chapter Exam

You can now go to www.ExcelCEO.com, log in and take the exam. Make sure that you take the exam on the same computer that you completed the sample files on, as some of the questions on the exam may refer to some of the completed examples.

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 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.

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*", which is 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 in 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 system.

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. 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, but 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 software. 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_2010 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.*

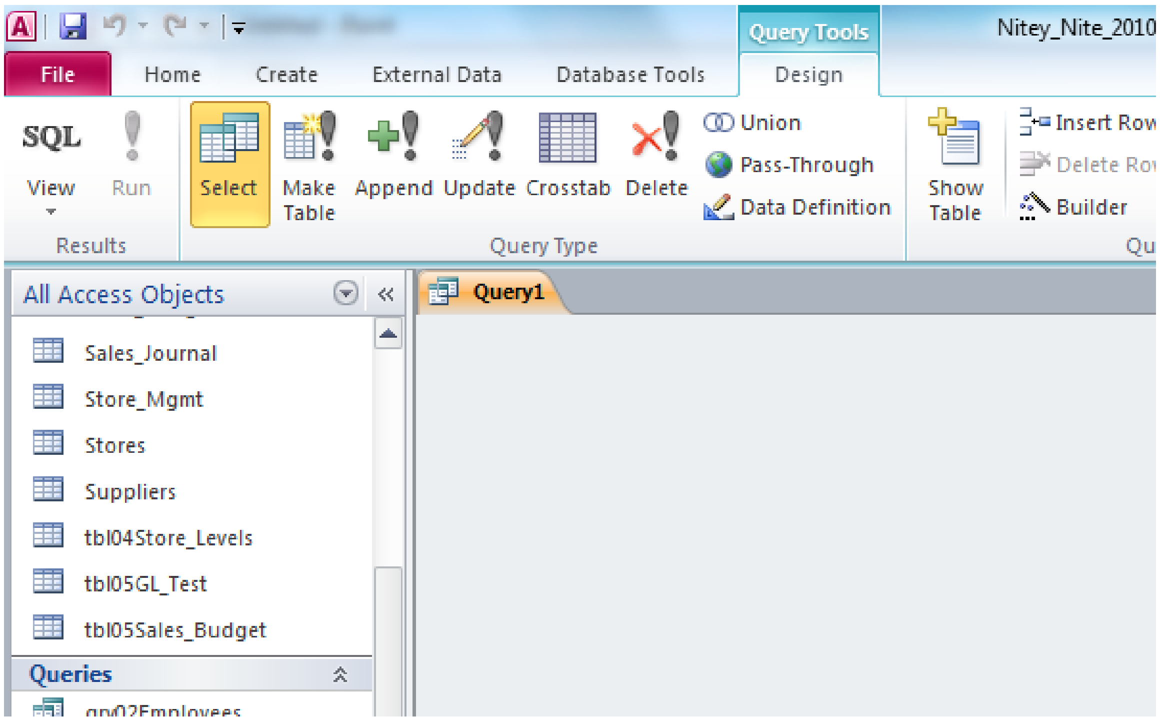
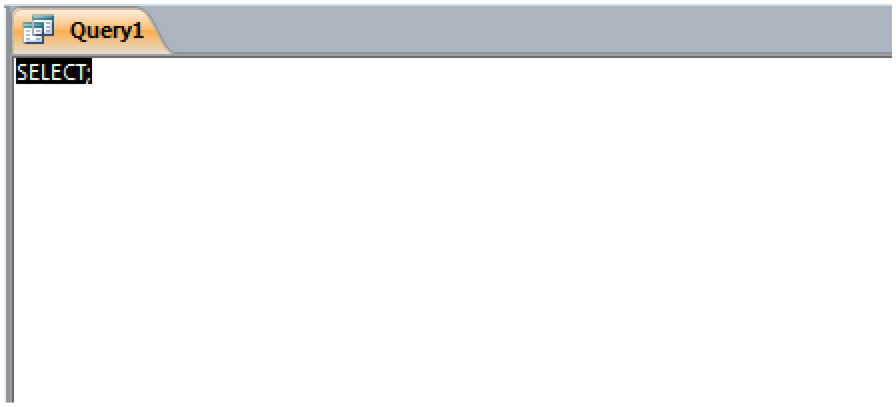


Figure 13.1

Notice on the left side of the Office ribbon the View icon reads SQL, instead of the normal View icon. It shows 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;
```

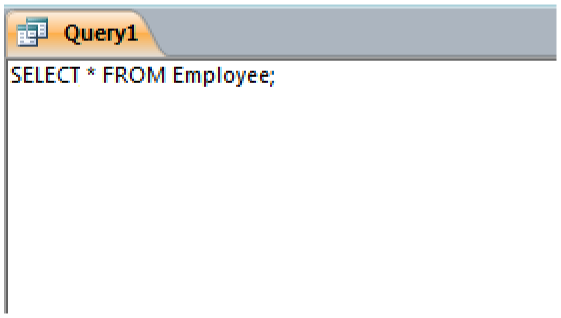
Figure 13.2

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.

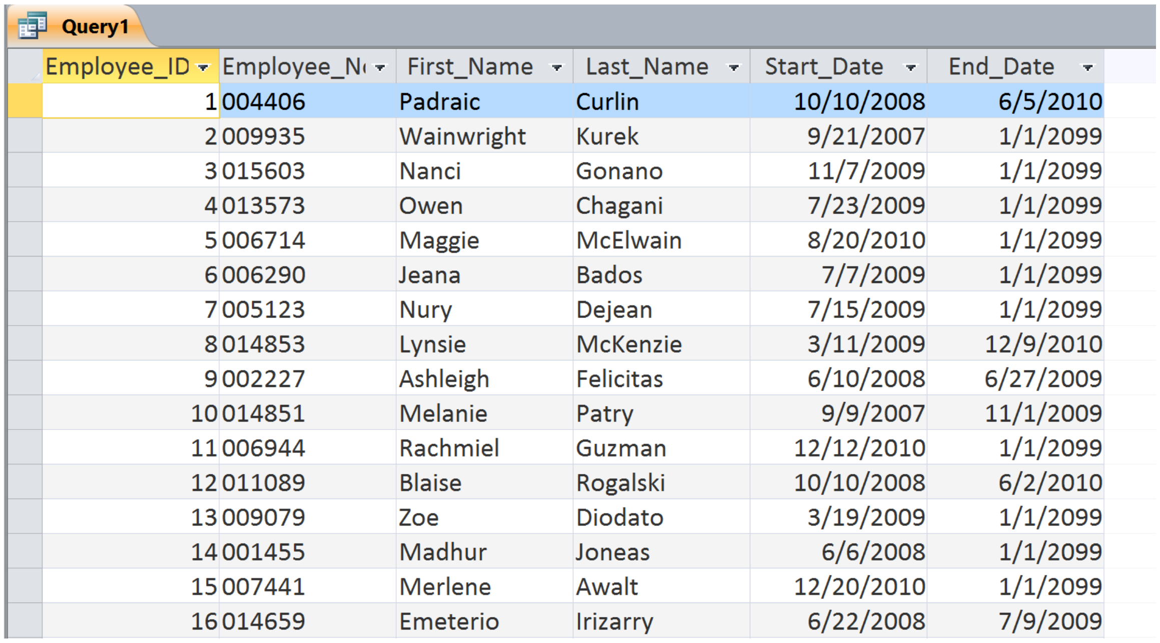
5. *In* **SQL View**, *type* **SELECT * FROM Employee;**



```
SELECT * FROM Employee;
```

Figure 13.3

6. *Click the* **View** *icon (to see the results in* **Datasheet** *view).*



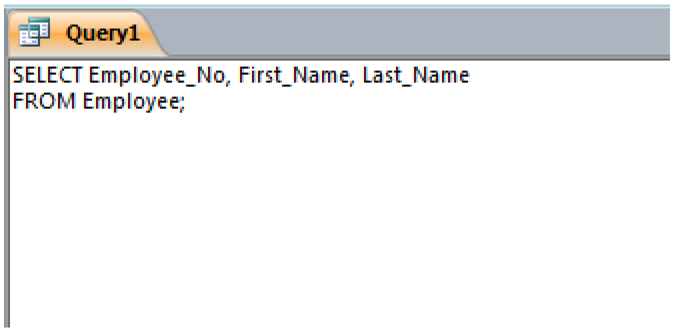
Employee_ID	Employee_N	First_Name	Last_Name	Start_Date	End_Date
1	004406	Padraic	Curlin	10/10/2008	6/5/2010
2	009935	Wainwright	Kurek	9/21/2007	1/1/2099
3	015603	Nanci	Gonano	11/7/2009	1/1/2099
4	013573	Owen	Chagani	7/23/2009	1/1/2099
5	006714	Maggie	McElwain	8/20/2010	1/1/2099
6	006290	Jeana	Bados	7/7/2009	1/1/2099
7	005123	Nury	Dejean	7/15/2009	1/1/2099
8	014853	Lynsie	McKenzie	3/11/2009	12/9/2010
9	002227	Ashleigh	Felicitas	6/10/2008	6/27/2009
10	014851	Melanie	Patry	9/9/2007	11/1/2009
11	006944	Rachmiel	Guzman	12/12/2010	1/1/2099
12	011089	Blaise	Rogalski	10/10/2008	6/2/2010
13	009079	Zoe	Diodato	3/19/2009	1/1/2099
14	001455	Madhur	Joneas	6/6/2008	1/1/2099
15	007441	Merlene	Awalt	12/20/2010	1/1/2099
16	014659	Emeterio	Irizarry	6/22/2008	7/9/2009

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. 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 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 for 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.*

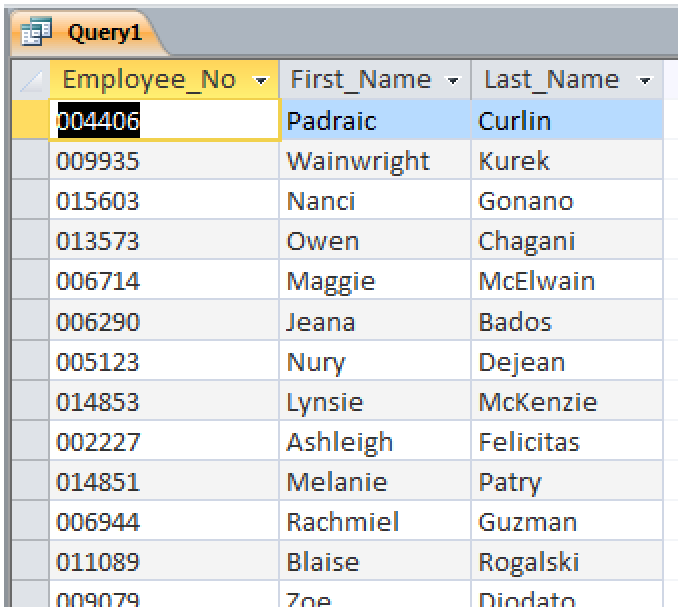


```
SELECT Employee_No, First_Name, Last_Name
FROM Employee;
```

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
002227	Ashleigh	Felicitas
014851	Melanie	Patry
006944	Rachmiel	Guzman
011089	Blaise	Rogalski
009079	Zoe	Diodato

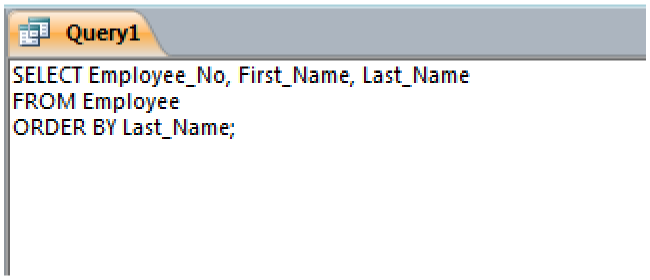
Figure 13.6

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.

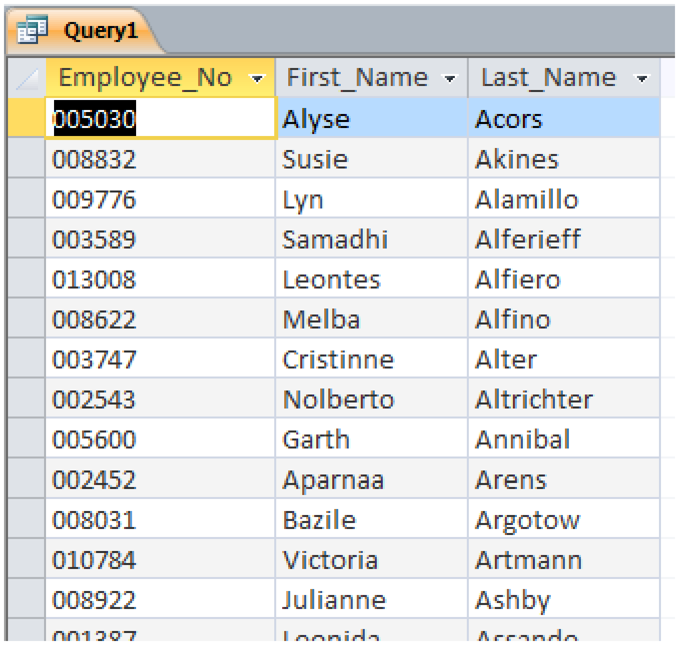
11. Return to **SQL View**.
12. Edit the code as in **Figure 13.7:**



```
SELECT Employee_No, First_Name, Last_Name
FROM Employee
ORDER BY Last_Name;
```

Figure 13.7

13. Run the query.


Query1

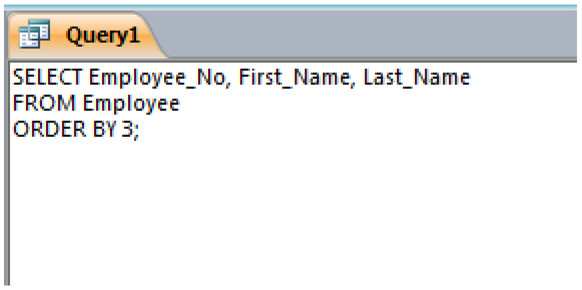
Employee_No	First_Name	Last_Name
005030	Alyse	Acors
008832	Susie	Akines
009776	Lyn	Alamillo
003589	Samadhi	Alferieff
013008	Leontes	Alfiero
008622	Melba	Alfino
003747	Cristinne	Alter
002543	Nolberto	Altrichter
005600	Garth	Annibal
002452	Aparnaa	Arens
008031	Bazile	Argotow
010784	Victoria	Artmann
008922	Julianne	Ashby
001387	Leonida	Assando

Figure 13.8

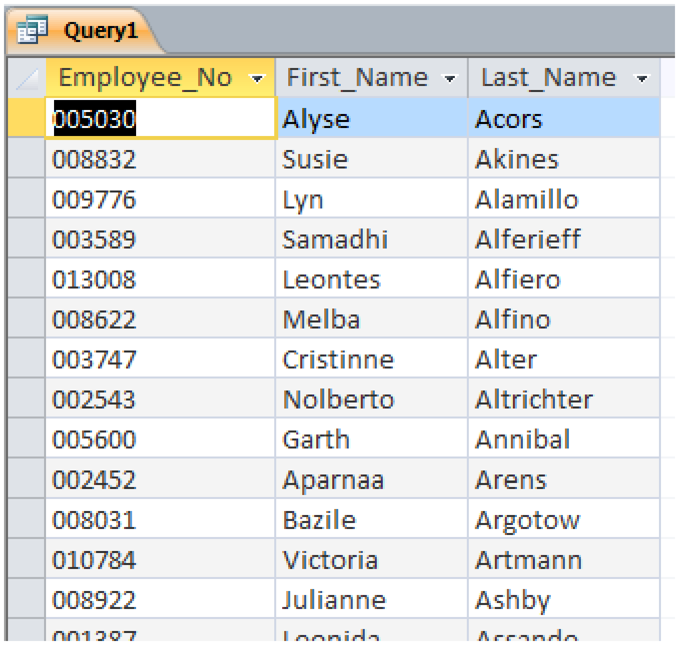
From this point on, I won't always ask you to type in the code, or show a picture of the code then tell you to run it, then go back to the SQL view. Every time you change your code, you should run it to make sure it works, and that you're getting the expected results.

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, 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.


Query1

```
SELECT Employee_No, First_Name, Last_Name
FROM Employee
ORDER BY 3;
```



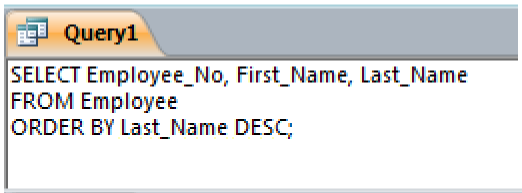
Query1

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	Accando

Figure 13.9

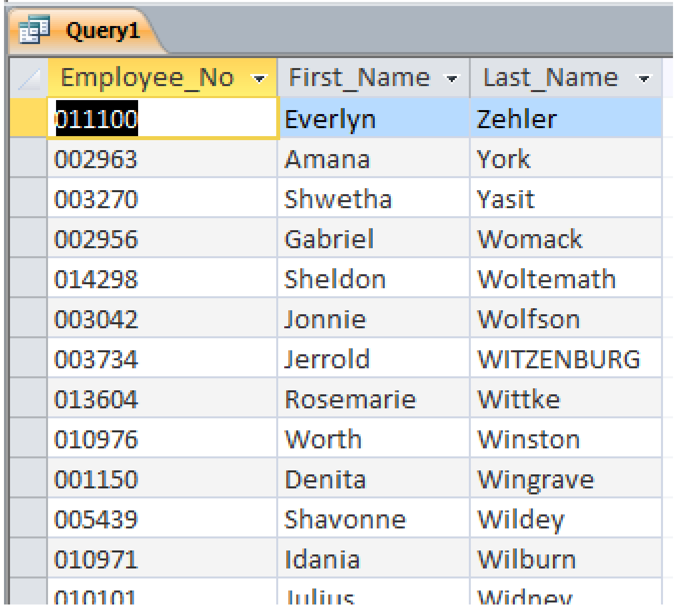
By default, SQL assumes you want to sort in ascending order. If you want to sort in descending order, just type "desc" after the field or position, like this:

15. Edit the code as in **Figure 13.10** *and run the query.*



Query1

```
SELECT Employee_No, First_Name, Last_Name
FROM Employee
ORDER BY Last_Name DESC;
```



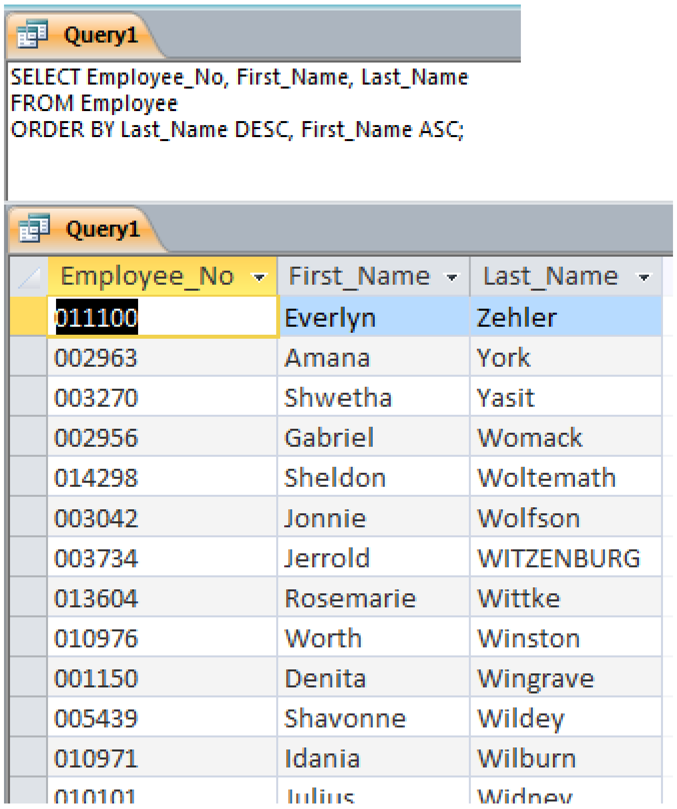
Query1

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
001150	Denita	Wingrave
005439	Shavonne	Wildey
010971	Idania	Wilburn
010101	Julius	Widney

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 ascending order and then by first name in descending 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
003042	Jonnie	Wolfson
003734	Jerrold	WITZENBURG
013604	Rosemarie	Wittke
010976	Worth	Winston
001150	Denita	Wingrave
005439	Shavonne	Wildey
010971	Idania	Wilburn
010101	Julius	Widney

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.

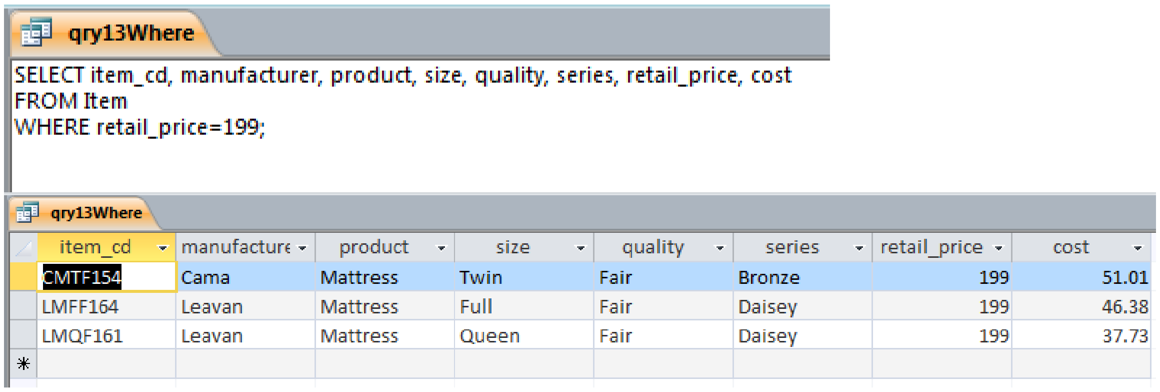
17. Save the query as **qry13Select_From_Order** *and close the query.*

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 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;
```

item_cd	manufacture	product	size	quality	series	retail_price	cost
CMTF154	Cama	Mattress	Twin	Fair	Bronze	199	51.01
LMFF164	Leavan	Mattress	Full	Fair	Daisey	199	46.38
LMQF161	Leavan	Mattress	Queen	Fair	Daisey	199	37.73
*							

Figure 13.12

Out of the 68 items in the Item table, only three have a retail price of $199.00.

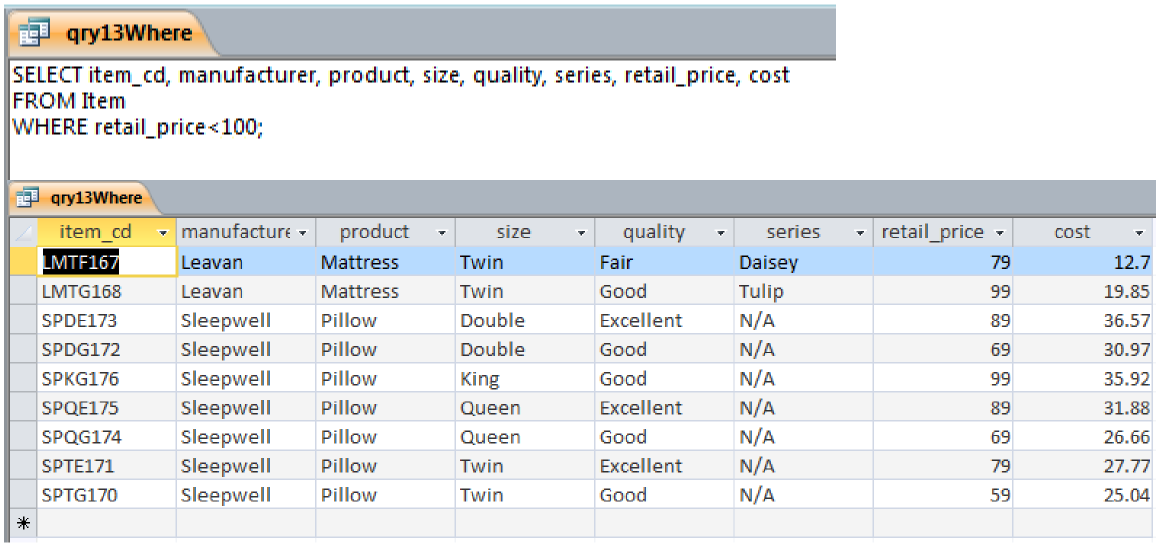
This exercise is a simple equality test (i.e., using the "=" sign), but SQL can do much more than equality. There are numerous operators you can use in a WHERE clause in SQL. The table below lists those operators.

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.*



qry13Where

```
SELECT item_cd, manufacturer, product, size, quality, series, retail_price, cost
FROM Item
WHERE retail_price<100;
```

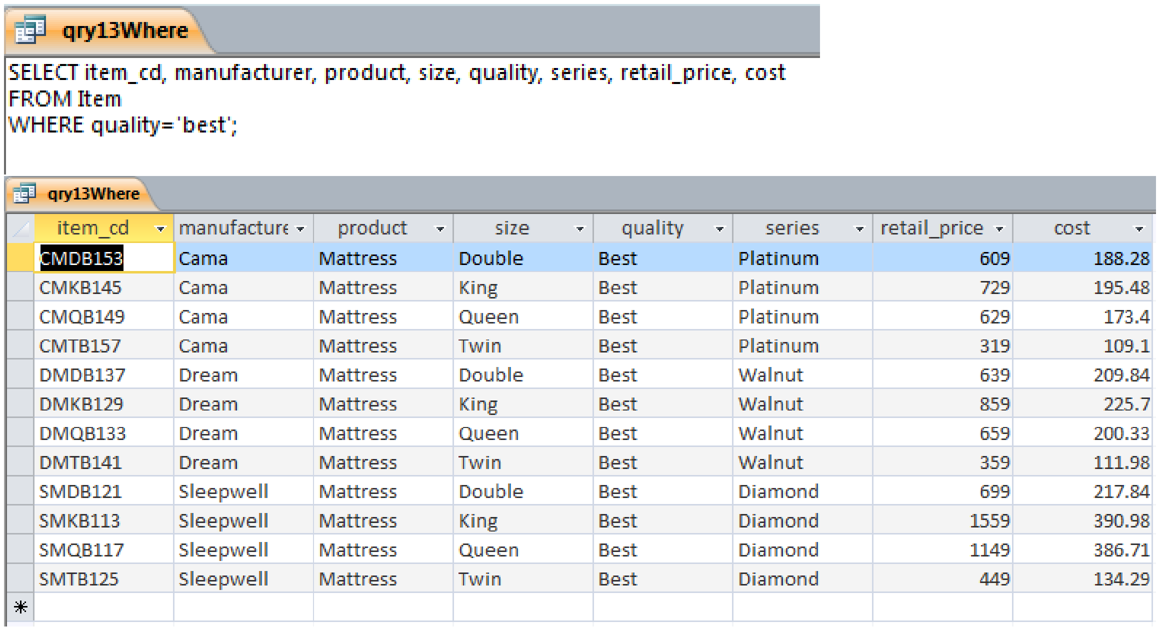
qry13Where

item_cd	manufacture	product	size	quality	series	retail_price	cost
LMTF167	Leavan	Mattress	Twin	Fair	Daisey	79	12.7
LMTG168	Leavan	Mattress	Twin	Good	Tulip	99	19.85
SPDE173	Sleepwell	Pillow	Double	Excellent	N/A	89	36.57
SPDG172	Sleepwell	Pillow	Double	Good	N/A	69	30.97
SPKG176	Sleepwell	Pillow	King	Good	N/A	99	35.92
SPQE175	Sleepwell	Pillow	Queen	Excellent	N/A	89	31.88
SPQG174	Sleepwell	Pillow	Queen	Good	N/A	69	26.66
SPTE171	Sleepwell	Pillow	Twin	Excellent	N/A	79	27.77
SPTG170	Sleepwell	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.*



qry13Where

```
SELECT item_cd, manufacturer, product, size, quality, series, retail_price, cost
FROM Item
WHERE quality='best';
```

qry13Where

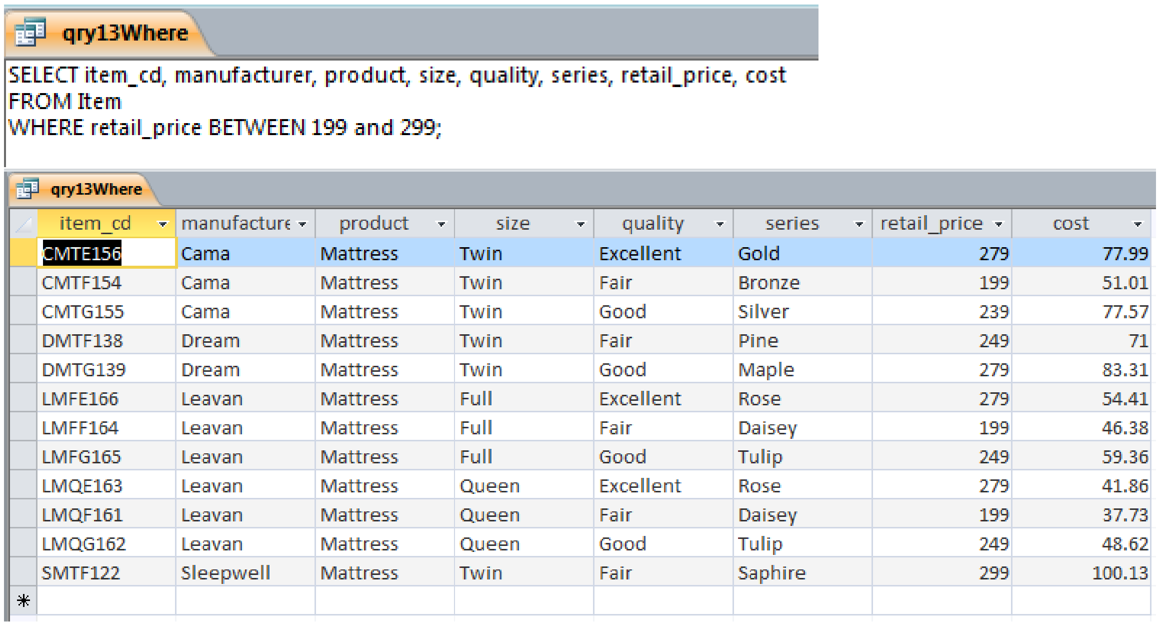
item_cd	manufacture	product	size	quality	series	retail_price	cost
CMDB153	Cama	Mattress	Double	Best	Platinum	609	188.28
CMKB145	Cama	Mattress	King	Best	Platinum	729	195.48
CMQB149	Cama	Mattress	Queen	Best	Platinum	629	173.4
CMTB157	Cama	Mattress	Twin	Best	Platinum	319	109.1
DMDB137	Dream	Mattress	Double	Best	Walnut	639	209.84
DMKB129	Dream	Mattress	King	Best	Walnut	859	225.7
DMQB133	Dream	Mattress	Queen	Best	Walnut	659	200.33
DMTB141	Dream	Mattress	Twin	Best	Walnut	359	111.98
SMDB121	Sleepwell	Mattress	Double	Best	Diamond	699	217.84
SMKB113	Sleepwell	Mattress	King	Best	Diamond	1559	390.98
SMQB117	Sleepwell	Mattress	Queen	Best	Diamond	1149	386.71
SMTB125	Sleepwell	Mattress	Twin	Best	Diamond	449	134.29

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. 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.*



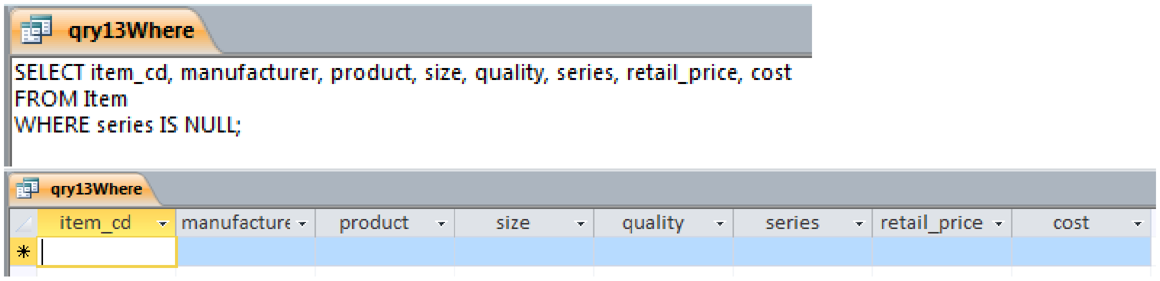
```
SELECT item_cd, manufacturer, product, size, quality, series, retail_price, cost
FROM Item
WHERE retail_price BETWEEN 199 and 299;
```

item_cd	manufacture	product	size	quality	series	retail_price	cost
CMTE156	Cama	Mattress	Twin	Excellent	Gold	279	77.99
CMTF154	Cama	Mattress	Twin	Fair	Bronze	199	51.01
CMTG155	Cama	Mattress	Twin	Good	Silver	239	77.57
DMTF138	Dream	Mattress	Twin	Fair	Pine	249	71
DMTG139	Dream	Mattress	Twin	Good	Maple	279	83.31
LMFE166	Leavan	Mattress	Full	Excellent	Rose	279	54.41
LMFF164	Leavan	Mattress	Full	Fair	Daisey	199	46.38
LMFG165	Leavan	Mattress	Full	Good	Tulip	249	59.36
LMQE163	Leavan	Mattress	Queen	Excellent	Rose	279	41.86
LMQF161	Leavan	Mattress	Queen	Fair	Daisey	199	37.73
LMQG162	Leavan	Mattress	Queen	Good	Tulip	249	48.62
SMTF122	Sleepwell	Mattress	Twin	Fair	Saphire	299	100.13

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;
```

item_cd	manufacture	product	size	quality	series	retail_price	cost

Figure 13.16

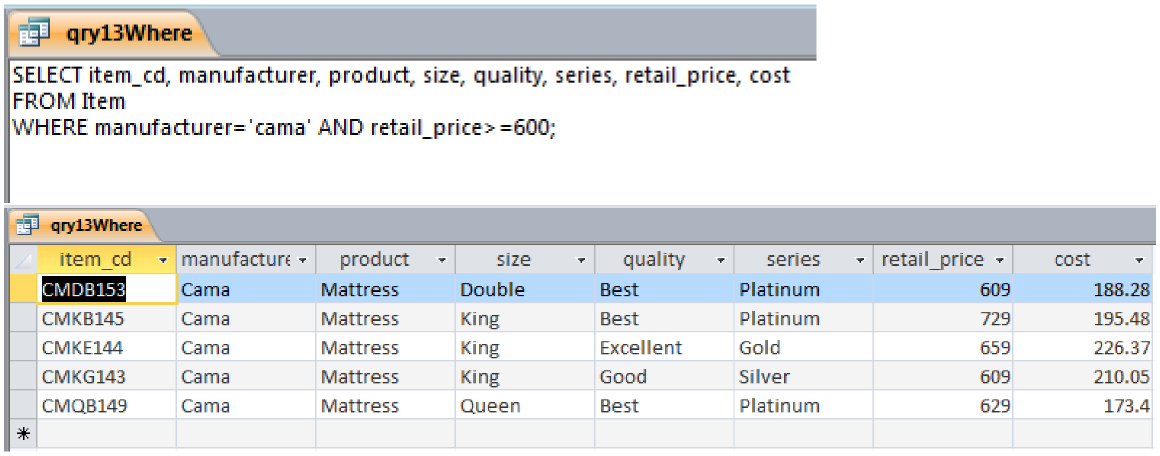
No records were returned, meaning there are no records with NULL values in the Series field.

> ***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 2010 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. *Edit the code as shown in* **Figure 13.17** *and run the query.*



```
SELECT item_cd, manufacturer, product, size, quality, series, retail_price, cost
FROM Item
WHERE manufacturer='cama' AND retail_price>=600;
```

item_cd	manufacture	product	size	quality	series	retail_price	cost
CMDB153	Cama	Mattress	Double	Best	Platinum	609	188.28
CMKB145	Cama	Mattress	King	Best	Platinum	729	195.48
CMKE144	Cama	Mattress	King	Excellent	Gold	659	226.37
CMKG143	Cama	Mattress	King	Good	Silver	609	210.05
CMQB149	Cama	Mattress	Queen	Best	Platinum	629	173.4

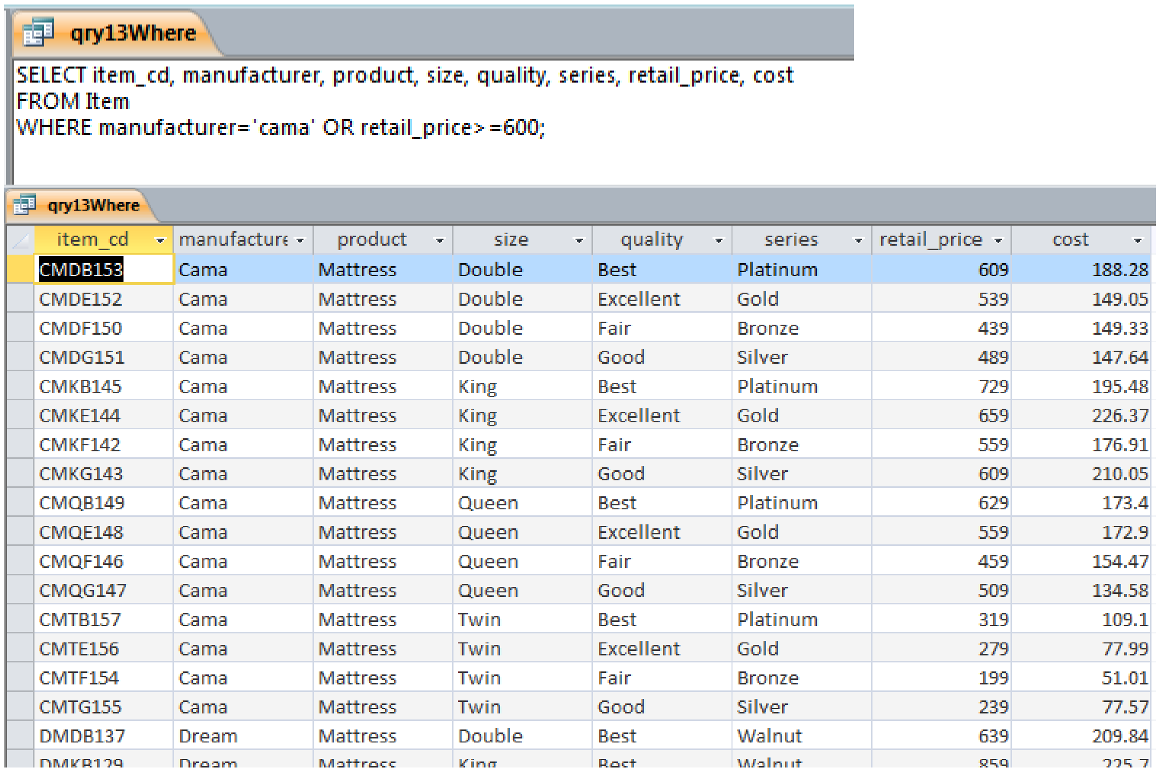
Figure 13.17

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.*


qry13Where

```
SELECT item_cd, manufacturer, product, size, quality, series, retail_price, cost
FROM Item
WHERE manufacturer='cama' OR retail_price>=600;
```

qry13Where

item_cd	manufacture	product	size	quality	series	retail_price	cost
CMDB153	Cama	Mattress	Double	Best	Platinum	609	188.28
CMDE152	Cama	Mattress	Double	Excellent	Gold	539	149.05
CMDF150	Cama	Mattress	Double	Fair	Bronze	439	149.33
CMDG151	Cama	Mattress	Double	Good	Silver	489	147.64
CMKB145	Cama	Mattress	King	Best	Platinum	729	195.48
CMKE144	Cama	Mattress	King	Excellent	Gold	659	226.37
CMKF142	Cama	Mattress	King	Fair	Bronze	559	176.91
CMKG143	Cama	Mattress	King	Good	Silver	609	210.05
CMQB149	Cama	Mattress	Queen	Best	Platinum	629	173.4
CMQE148	Cama	Mattress	Queen	Excellent	Gold	559	172.9
CMQF146	Cama	Mattress	Queen	Fair	Bronze	459	154.47
CMQG147	Cama	Mattress	Queen	Good	Silver	509	134.58
CMTB157	Cama	Mattress	Twin	Best	Platinum	319	109.1
CMTE156	Cama	Mattress	Twin	Excellent	Gold	279	77.99
CMTF154	Cama	Mattress	Twin	Fair	Bronze	199	51.01
CMTG155	Cama	Mattress	Twin	Good	Silver	239	77.57
DMDB137	Dream	Mattress	Double	Best	Walnut	639	209.84
DMKB129	Dream	Mattress	King	Best	Walnut	859	225.7

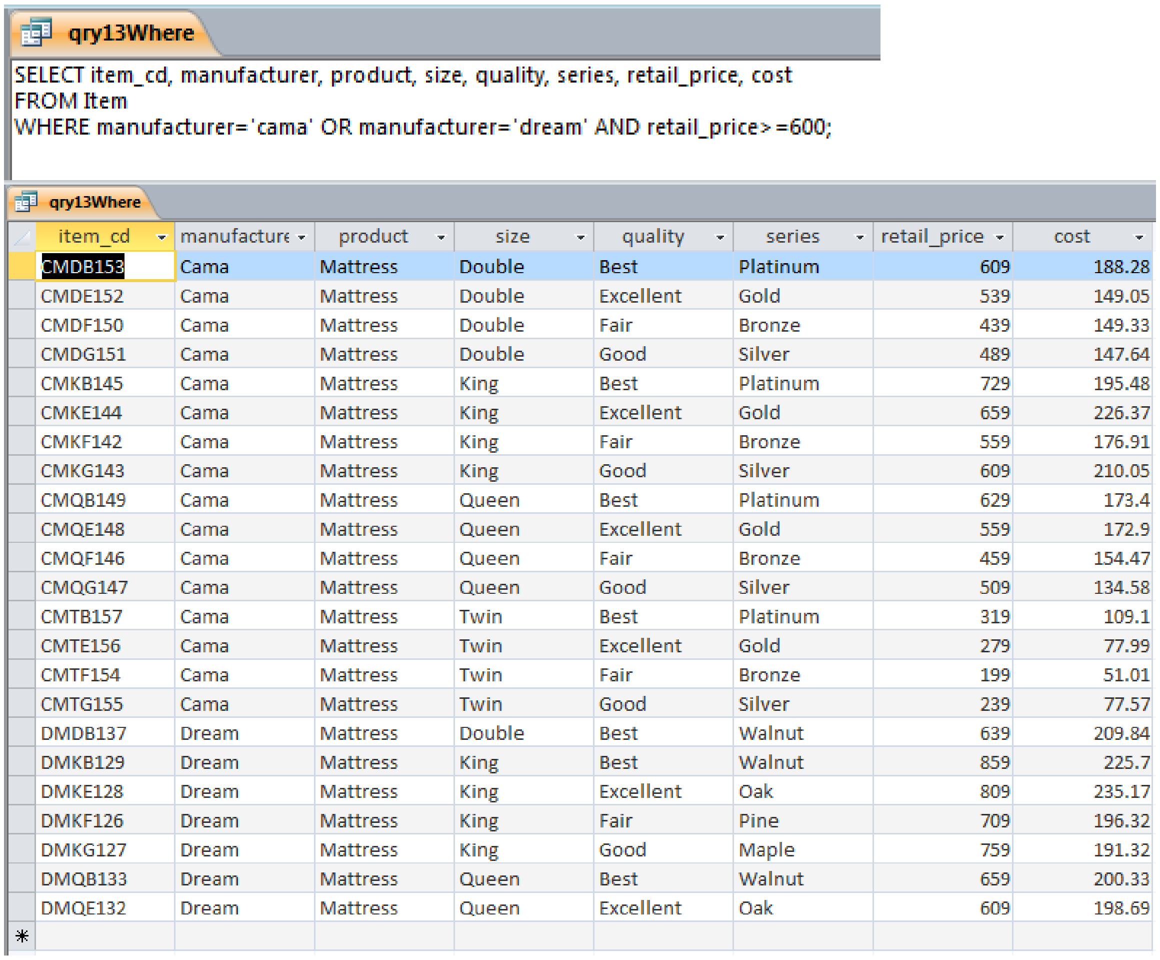
Figure 13.18

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.*



qry13Where

```
SELECT item_cd, manufacturer, product, size, quality, series, retail_price, cost
FROM Item
WHERE manufacturer='cama' OR manufacturer='dream' AND retail_price>=600;
```

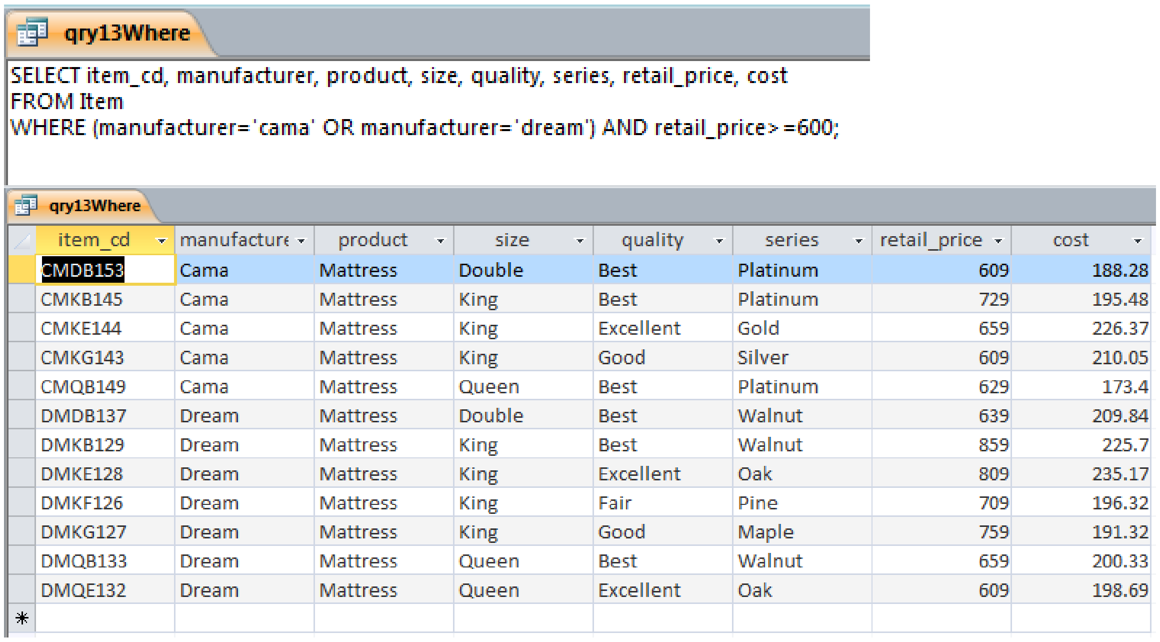
qry13Where

item_cd	manufacture	product	size	quality	series	retail_price	cost
CMDB153	Cama	Mattress	Double	Best	Platinum	609	188.28
CMDE152	Cama	Mattress	Double	Excellent	Gold	539	149.05
CMDF150	Cama	Mattress	Double	Fair	Bronze	439	149.33
CMDG151	Cama	Mattress	Double	Good	Silver	489	147.64
CMKB145	Cama	Mattress	King	Best	Platinum	729	195.48
CMKE144	Cama	Mattress	King	Excellent	Gold	659	226.37
CMKF142	Cama	Mattress	King	Fair	Bronze	559	176.91
CMKG143	Cama	Mattress	King	Good	Silver	609	210.05
CMQB149	Cama	Mattress	Queen	Best	Platinum	629	173.4
CMQE148	Cama	Mattress	Queen	Excellent	Gold	559	172.9
CMQF146	Cama	Mattress	Queen	Fair	Bronze	459	154.47
CMQG147	Cama	Mattress	Queen	Good	Silver	509	134.58
CMTB157	Cama	Mattress	Twin	Best	Platinum	319	109.1
CMTE156	Cama	Mattress	Twin	Excellent	Gold	279	77.99
CMTF154	Cama	Mattress	Twin	Fair	Bronze	199	51.01
CMTG155	Cama	Mattress	Twin	Good	Silver	239	77.57
DMDB137	Dream	Mattress	Double	Best	Walnut	639	209.84
DMKB129	Dream	Mattress	King	Best	Walnut	859	225.7
DMKE128	Dream	Mattress	King	Excellent	Oak	809	235.17
DMKF126	Dream	Mattress	King	Fair	Pine	709	196.32
DMKG127	Dream	Mattress	King	Good	Maple	759	191.32
DMQB133	Dream	Mattress	Queen	Best	Walnut	659	200.33
DMQE132	Dream	Mattress	Queen	Excellent	Oak	609	198.69
*							

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.*


qry13Where

```
SELECT item_cd, manufacturer, product, size, quality, series, retail_price, cost
FROM Item
WHERE (manufacturer='cama' OR manufacturer='dream') AND retail_price>=600;
```

qry13Where

item_cd	manufacture	product	size	quality	series	retail_price	cost
CMDB153	Cama	Mattress	Double	Best	Platinum	609	188.28
CMKB145	Cama	Mattress	King	Best	Platinum	729	195.48
CMKE144	Cama	Mattress	King	Excellent	Gold	659	226.37
CMKG143	Cama	Mattress	King	Good	Silver	609	210.05
CMQB149	Cama	Mattress	Queen	Best	Platinum	629	173.4
DMDB137	Dream	Mattress	Double	Best	Walnut	639	209.84
DMKB129	Dream	Mattress	King	Best	Walnut	859	225.7
DMKE128	Dream	Mattress	King	Excellent	Oak	809	235.17
DMKF126	Dream	Mattress	King	Fair	Pine	709	196.32
DMKG127	Dream	Mattress	King	Good	Maple	759	191.32
DMQB133	Dream	Mattress	Queen	Best	Walnut	659	200.33
DMQE132	Dream	Mattress	Queen	Excellent	Oak	609	198.69
*							

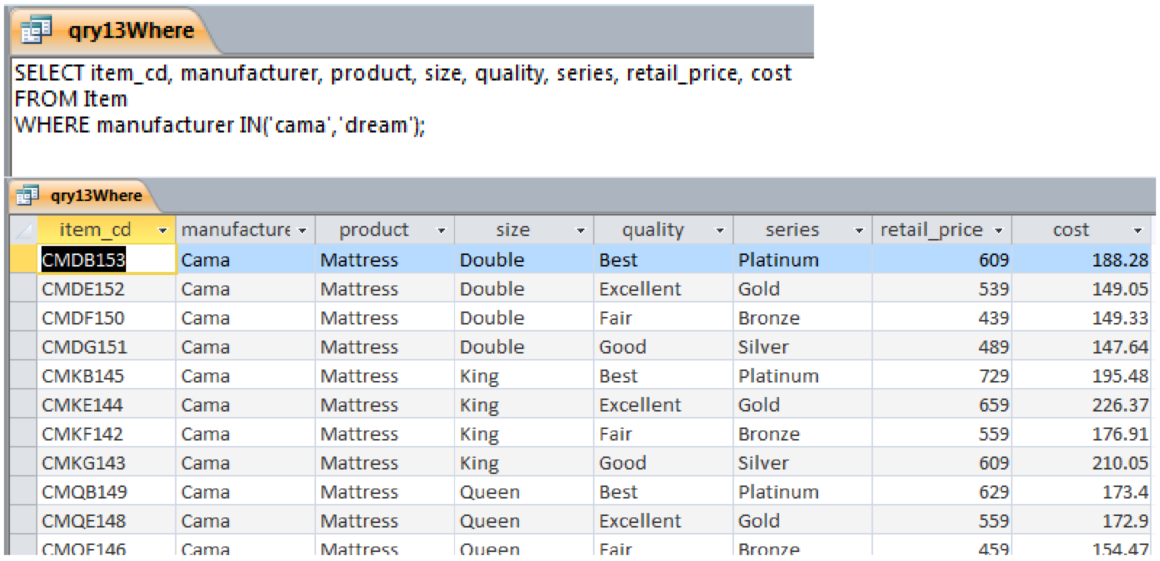
Figure 13.20

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.*



qry13Where

```
SELECT item_cd, manufacturer, product, size, quality, series, retail_price, cost
FROM Item
WHERE manufacturer IN('cama','dream');
```

qry13Where

item_cd	manufacturer	product	size	quality	series	retail_price	cost
CMDB153	Cama	Mattress	Double	Best	Platinum	609	188.28
CMDE152	Cama	Mattress	Double	Excellent	Gold	539	149.05
CMDF150	Cama	Mattress	Double	Fair	Bronze	439	149.33
CMDG151	Cama	Mattress	Double	Good	Silver	489	147.64
CMKB145	Cama	Mattress	King	Best	Platinum	729	195.48
CMKE144	Cama	Mattress	King	Excellent	Gold	659	226.37
CMKF142	Cama	Mattress	King	Fair	Bronze	559	176.91
CMKG143	Cama	Mattress	King	Good	Silver	609	210.05
CMQB149	Cama	Mattress	Queen	Best	Platinum	629	173.4
CMQE148	Cama	Mattress	Queen	Excellent	Gold	559	172.9
CMQF146	Cama	Mattress	Queen	Fair	Bronze	459	154.47

Figure 13.21

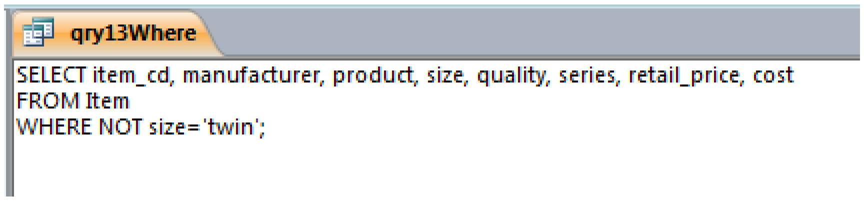
You now get a list of 32 items, which is ALL of the items that have a manufacturer of either Cama or Dream. Other advantages of using the IN operator as opposed to an OR operator include:

- 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

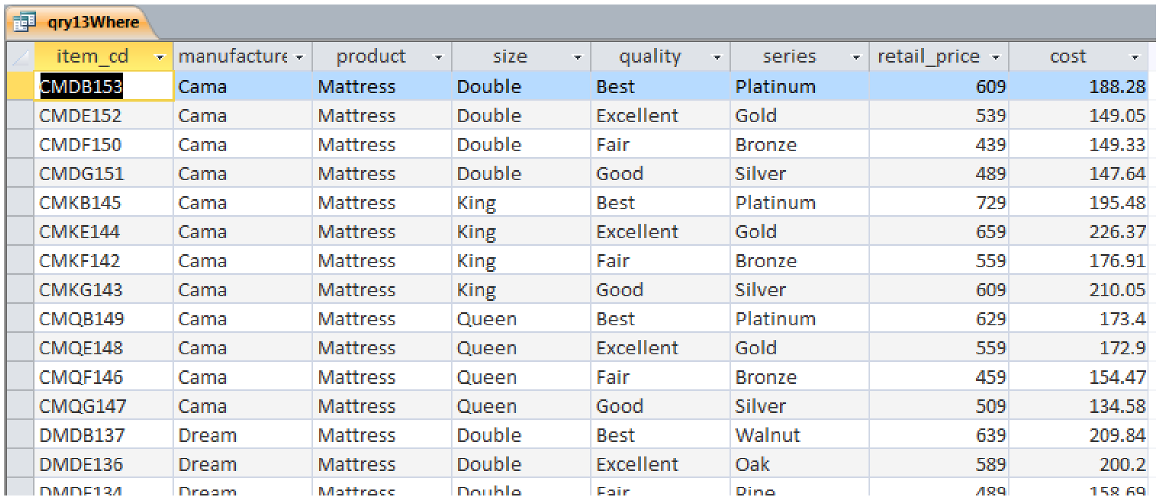
There is one purpose and one purpose only in the NOT operator's life – to negate a condition. NOT is used only when you don't want to include something. Normally, NOT is used with other operators in a WHERE statement, but it can be used by itself as well. Let's suppose you want to see a list of records 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	manufacture	product	size	quality	series	retail_price	cost
CMDB153	Cama	Mattress	Double	Best	Platinum	609	188.28
CMDE152	Cama	Mattress	Double	Excellent	Gold	539	149.05
CMDF150	Cama	Mattress	Double	Fair	Bronze	439	149.33
CMDG151	Cama	Mattress	Double	Good	Silver	489	147.64
CMKB145	Cama	Mattress	King	Best	Platinum	729	195.48
CMKE144	Cama	Mattress	King	Excellent	Gold	659	226.37
CMKF142	Cama	Mattress	King	Fair	Bronze	559	176.91
CMKG143	Cama	Mattress	King	Good	Silver	609	210.05
CMQB149	Cama	Mattress	Queen	Best	Platinum	629	173.4
CMQE148	Cama	Mattress	Queen	Excellent	Gold	559	172.9
CMQF146	Cama	Mattress	Queen	Fair	Bronze	459	154.47
CMQG147	Cama	Mattress	Queen	Good	Silver	509	134.58
DMDB137	Dream	Mattress	Double	Best	Walnut	639	209.84
DMDE136	Dream	Mattress	Double	Excellent	Oak	589	200.2
DMDF134	Dream	Mattress	Double	Fair	Pine	489	158.69

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 a 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

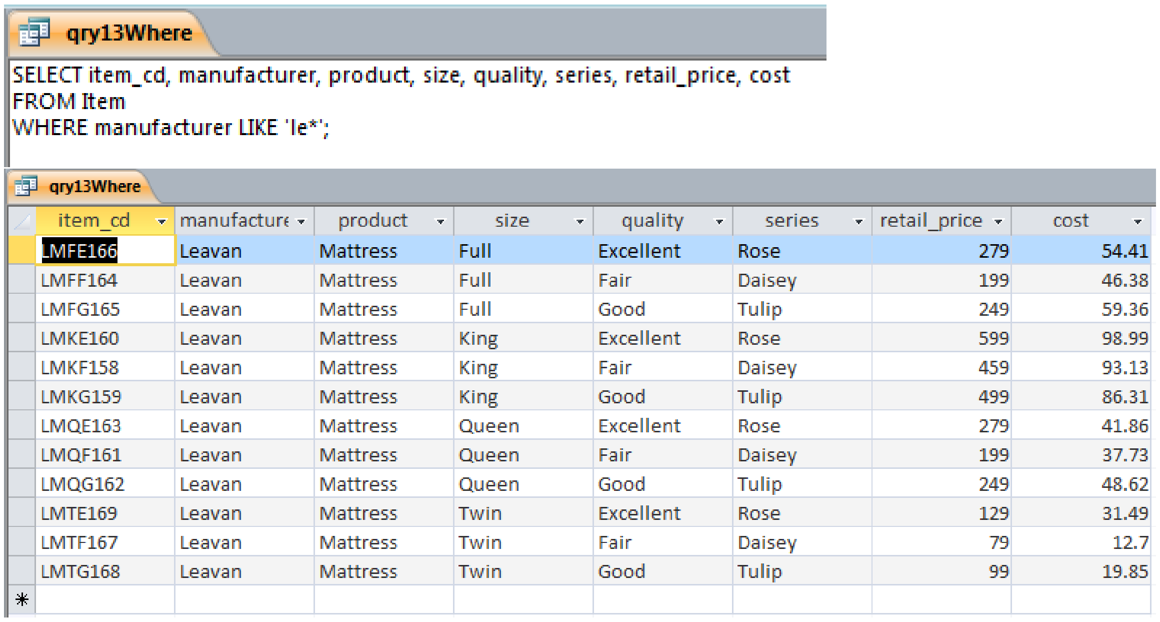
Whenever you use a wildcard character to search for a value in a WHERE clause, you must use the LIKE operator. The LIKE operator tells SQL to search for the pattern that follows where the wildcard character could stand for any character in the record. It comes in very handy for people who can't spell 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.*



qry13Where

```
SELECT item_cd, manufacturer, product, size, quality, series, retail_price, cost
FROM Item
WHERE manufacturer LIKE 'le*';
```

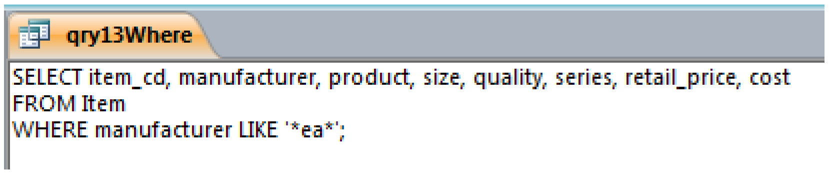
qry13Where

item_cd	manufacture	product	size	quality	series	retail_price	cost
LMFE166	Leavan	Mattress	Full	Excellent	Rose	279	54.41
LMFF164	Leavan	Mattress	Full	Fair	Daisey	199	46.38
LMFG165	Leavan	Mattress	Full	Good	Tulip	249	59.36
LMKE160	Leavan	Mattress	King	Excellent	Rose	599	98.99
LMKF158	Leavan	Mattress	King	Fair	Daisey	459	93.13
LMKG159	Leavan	Mattress	King	Good	Tulip	499	86.31
LMQE163	Leavan	Mattress	Queen	Excellent	Rose	279	41.86
LMQF161	Leavan	Mattress	Queen	Fair	Daisey	199	37.73
LMQG162	Leavan	Mattress	Queen	Good	Tulip	249	48.62
LMTE169	Leavan	Mattress	Twin	Excellent	Rose	129	31.49
LMTF167	Leavan	Mattress	Twin	Fair	Daisey	79	12.7
LMTG168	Leavan	Mattress	Twin	Good	Tulip	99	19.85
*							

Figure 13.23

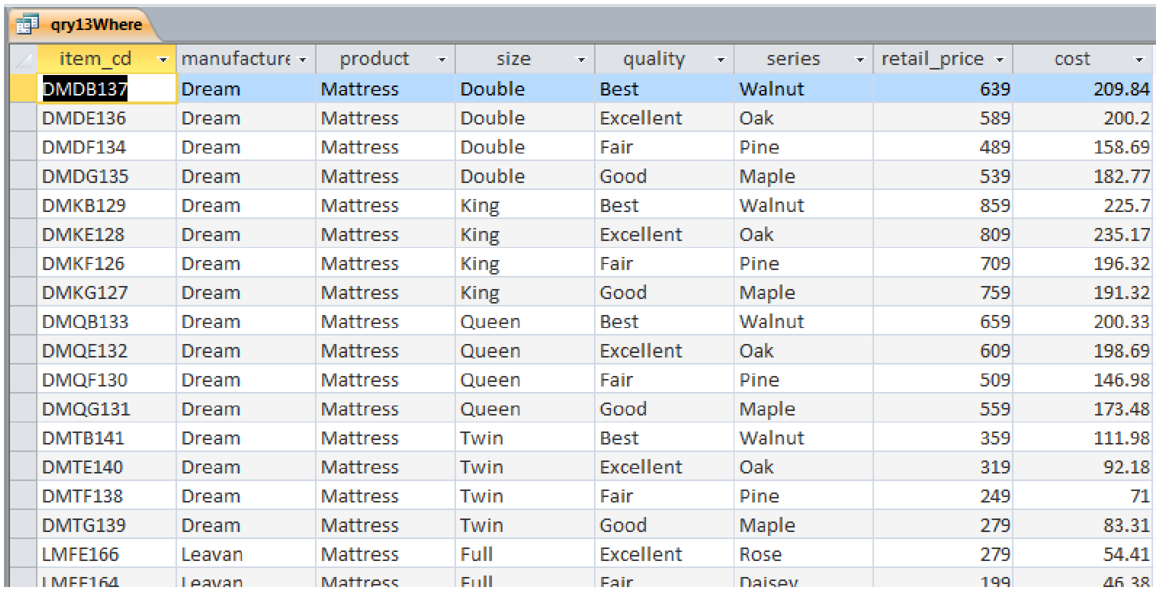
You should get a record set of 12 records, which represents all of the items that are manufactured by Leaven. You can use the asterisk (or percent sign in ANSI SQL) at any place in the search string. Let's suppose that you remember that the name of the manufacturer has an "ea" somewhere in the name, but that's all you know. 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.*



qry13Where

```
SELECT item_cd, manufacturer, product, size, quality, series, retail_price, cost
FROM Item
WHERE manufacturer LIKE '*ea*';
```



qry13Where

item_cd	manufacture	product	size	quality	series	retail_price	cost
DMDB137	Dream	Mattress	Double	Best	Walnut	639	209.84
DMDE136	Dream	Mattress	Double	Excellent	Oak	589	200.2
DMDF134	Dream	Mattress	Double	Fair	Pine	489	158.69
DMDG135	Dream	Mattress	Double	Good	Maple	539	182.77
DMKB129	Dream	Mattress	King	Best	Walnut	859	225.7
DMKE128	Dream	Mattress	King	Excellent	Oak	809	235.17
DMKF126	Dream	Mattress	King	Fair	Pine	709	196.32
DMKG127	Dream	Mattress	King	Good	Maple	759	191.32
DMQB133	Dream	Mattress	Queen	Best	Walnut	659	200.33
DMQE132	Dream	Mattress	Queen	Excellent	Oak	609	198.69
DMQF130	Dream	Mattress	Queen	Fair	Pine	509	146.98
DMQG131	Dream	Mattress	Queen	Good	Maple	559	173.48
DMTB141	Dream	Mattress	Twin	Best	Walnut	359	111.98
DMTE140	Dream	Mattress	Twin	Excellent	Oak	319	92.18
DMTF138	Dream	Mattress	Twin	Fair	Pine	249	71
DMTG139	Dream	Mattress	Twin	Good	Maple	279	83.31
LMFE166	Leavan	Mattress	Full	Excellent	Rose	279	54.41
LMFF164	Leavan	Mattress	Full	Fair	Daisey	199	46.38

Figure 13.24

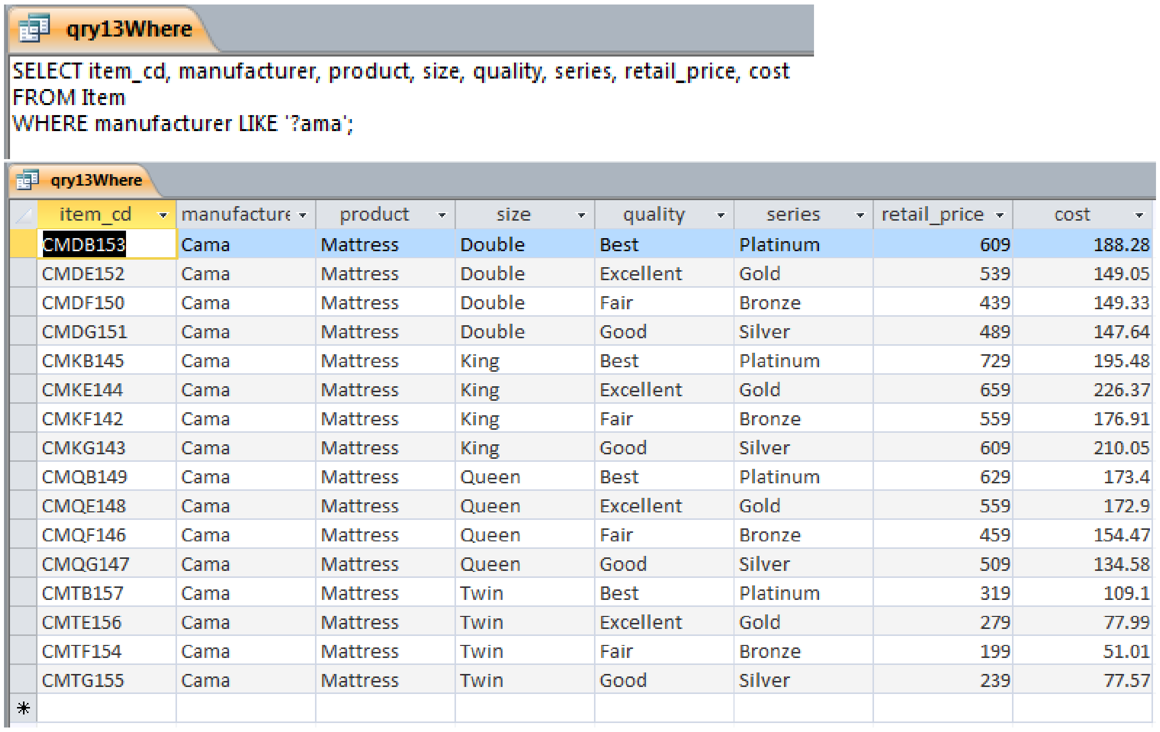
Oops. Now you have all of the manufacturers that have the string "ea" somewhere in their name, which includes Dream and Leaven. I've heard that if God wanted to destroy the world, all He would have to do is give everyone everything they prayed for. This is the text string you wanted, so this is what 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.*



qry13Where

```
SELECT item_cd, manufacturer, product, size, quality, series, retail_price, cost
FROM Item
WHERE manufacturer LIKE '?ama';
```

qry13Where

item_cd	manufacture	product	size	quality	series	retail_price	cost
CMDB153	Cama	Mattress	Double	Best	Platinum	609	188.28
CMDE152	Cama	Mattress	Double	Excellent	Gold	539	149.05
CMDF150	Cama	Mattress	Double	Fair	Bronze	439	149.33
CMDG151	Cama	Mattress	Double	Good	Silver	489	147.64
CMKB145	Cama	Mattress	King	Best	Platinum	729	195.48
CMKE144	Cama	Mattress	King	Excellent	Gold	659	226.37
CMKF142	Cama	Mattress	King	Fair	Bronze	559	176.91
CMKG143	Cama	Mattress	King	Good	Silver	609	210.05
CMQB149	Cama	Mattress	Queen	Best	Platinum	629	173.4
CMQE148	Cama	Mattress	Queen	Excellent	Gold	559	172.9
CMQF146	Cama	Mattress	Queen	Fair	Bronze	459	154.47
CMQG147	Cama	Mattress	Queen	Good	Silver	509	134.58
CMTB157	Cama	Mattress	Twin	Best	Platinum	319	109.1
CMTE156	Cama	Mattress	Twin	Excellent	Gold	279	77.99
CMTF154	Cama	Mattress	Twin	Fair	Bronze	199	51.01
CMTG155	Cama	Mattress	Twin	Good	Silver	239	77.57

Figure 13.25

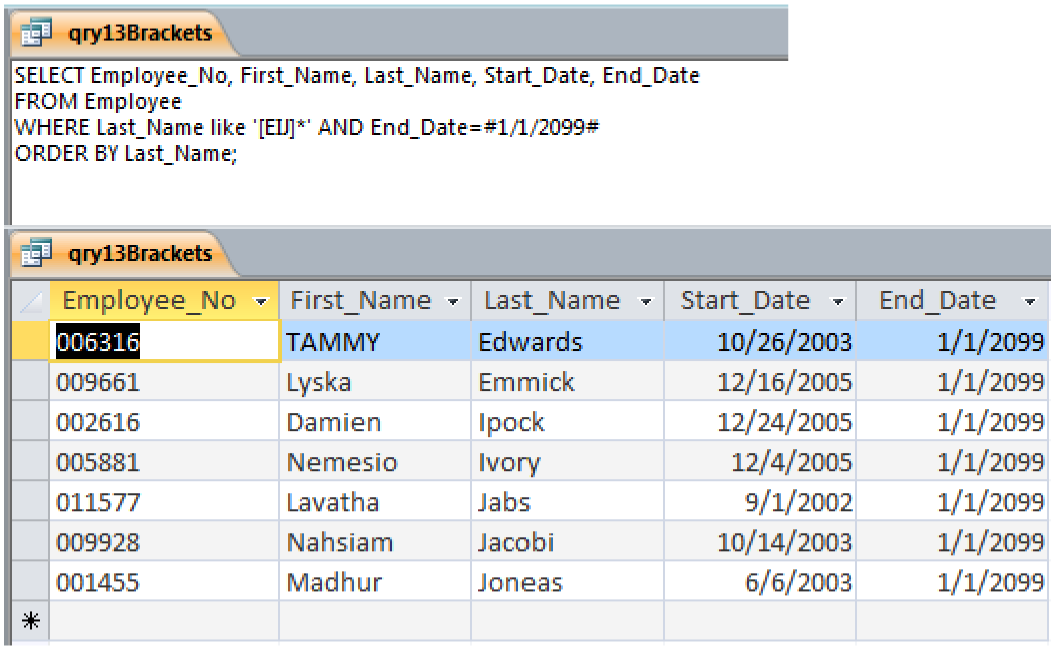
Here you get a record set of all of the items that are manufactured by Cama. Just remember that the asterisk (or percent sign in ANSI SQL) is used for any number of characters whereas the question mark (or underscore in ANSI SQL) searches for a single character.

16. Save and close the query.

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.*



qry13Brackets

```
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;
```

qry13Brackets

Employee_No	First_Name	Last_Name	Start_Date	End_Date
006316	TAMMY	Edwards	10/26/2003	1/1/2099
009661	Lyska	Emmick	12/16/2005	1/1/2099
002616	Damien	Ipock	12/24/2005	1/1/2099
005881	Nemesio	Ivory	12/4/2005	1/1/2099
011577	Lavatha	Jabs	9/1/2002	1/1/2099
009928	Nahsiam	Jacobi	10/14/2003	1/1/2099
001455	Madhur	Joneas	6/6/2003	1/1/2099
*				

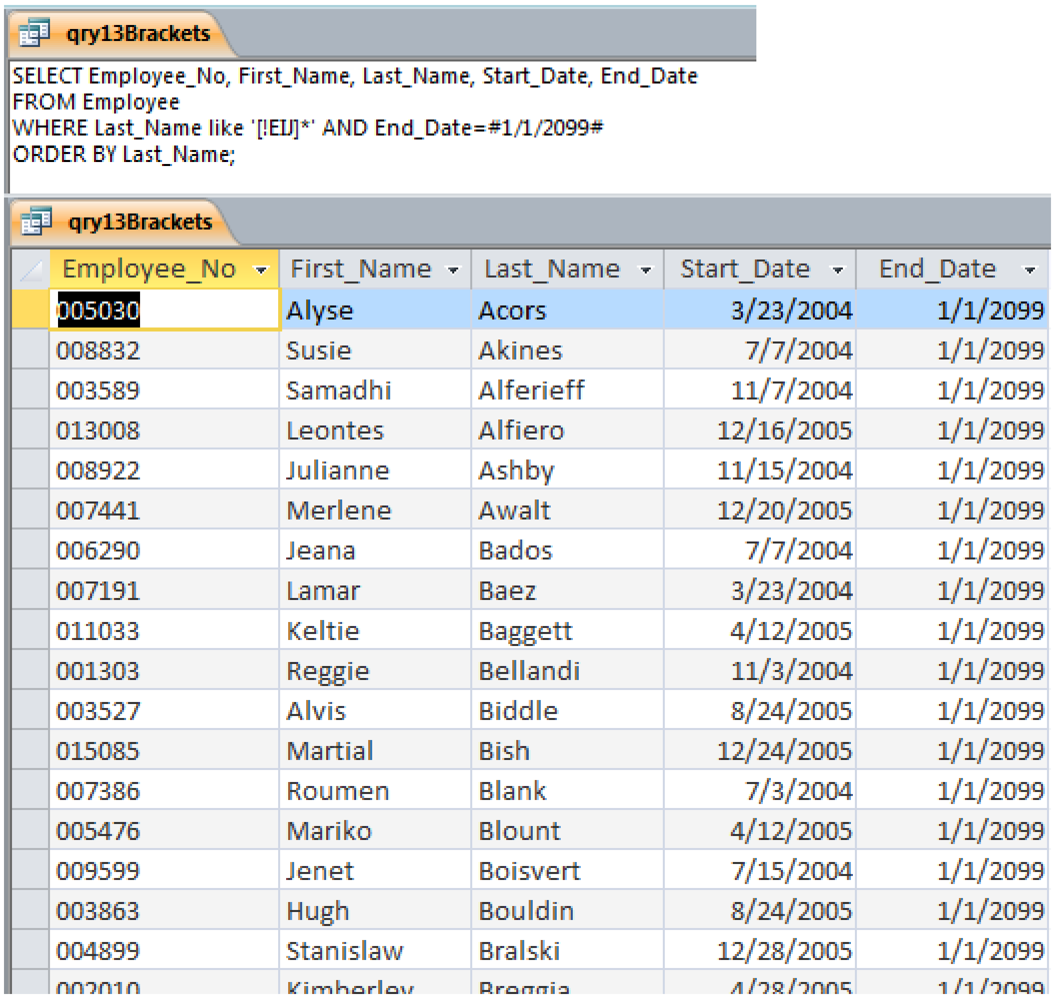
Figure 13.26

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.*


qry13Brackets

```
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;
```

qry13Brackets

Employee_No	First_Name	Last_Name	Start_Date	End_Date
005030	Alyse	Acors	3/23/2004	1/1/2099
008832	Susie	Akines	7/7/2004	1/1/2099
003589	Samadhi	Alferieff	11/7/2004	1/1/2099
013008	Leontes	Alfiero	12/16/2005	1/1/2099
008922	Julianne	Ashby	11/15/2004	1/1/2099
007441	Merlene	Awalt	12/20/2005	1/1/2099
006290	Jeana	Bados	7/7/2004	1/1/2099
007191	Lamar	Baez	3/23/2004	1/1/2099
011033	Keltie	Baggett	4/12/2005	1/1/2099
001303	Reggie	Bellandi	11/3/2004	1/1/2099
003527	Alvis	Biddle	8/24/2005	1/1/2099
015085	Martial	Bish	12/24/2005	1/1/2099
007386	Roumen	Blank	7/3/2004	1/1/2099
005476	Mariko	Blount	4/12/2005	1/1/2099
009599	Jenet	Boisvert	7/15/2004	1/1/2099
003863	Hugh	Bouldin	8/24/2005	1/1/2099
004899	Stanislaw	Bralski	12/28/2005	1/1/2099
002010	Kimberley	Breggia	4/28/2005	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 code 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 the query.*

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.

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 2010 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, log in and take the exam. Make sure that you take the exam on the same computer that you completed the sample files on, as some of the questions on the exam may refer to some of the completed examples.

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.

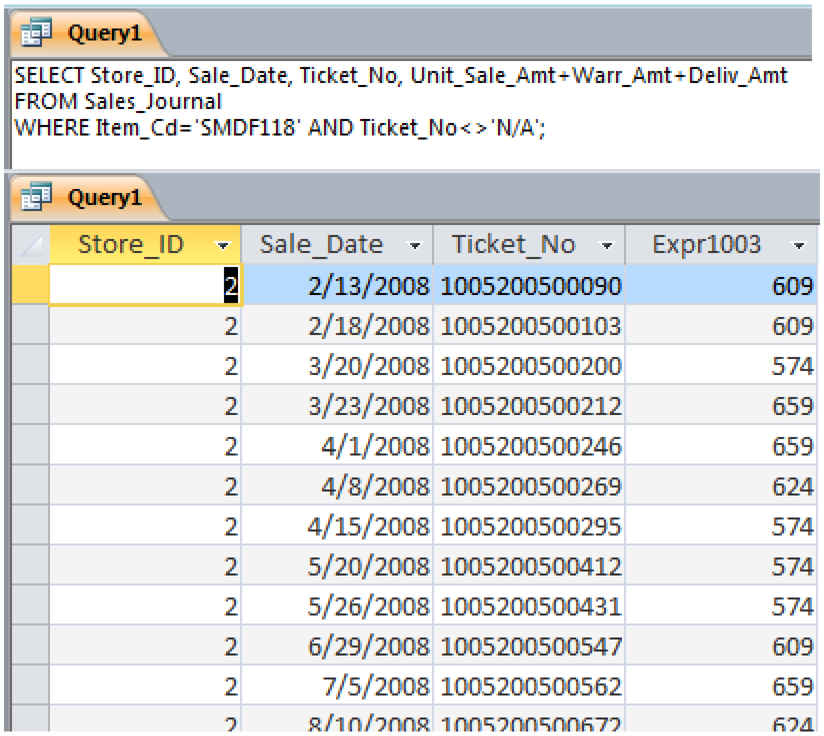
Calculated Fields

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 hard drive or on a network drive), 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 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_2010** *Access database.*
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 not* **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';
```

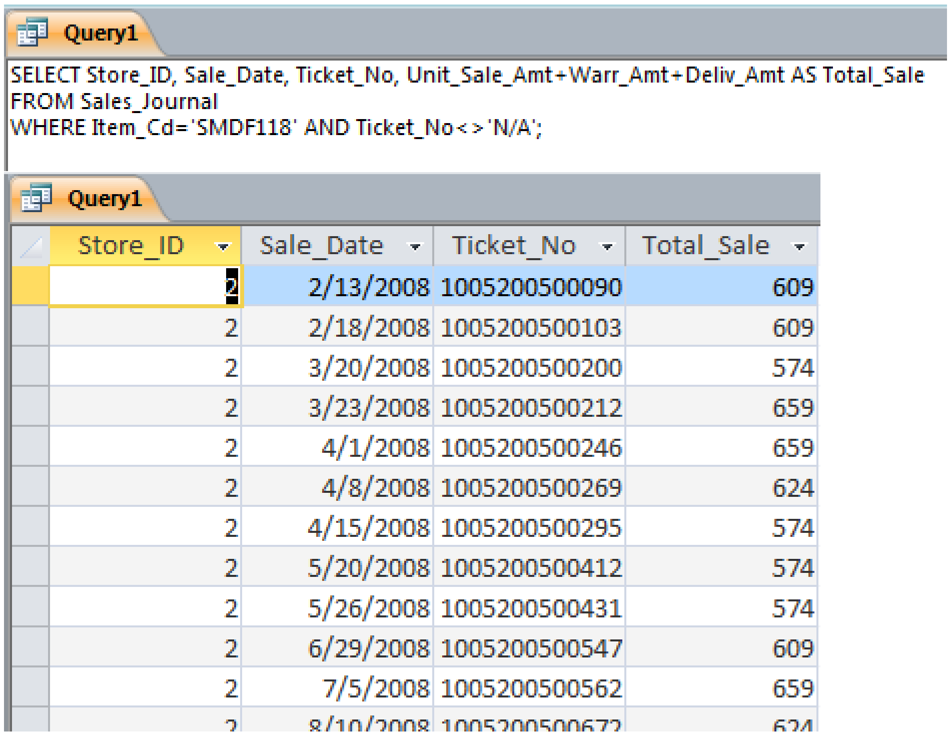
Store_ID	Sale_Date	Ticket_No	Expr1003
2	2/13/2008	1005200500090	609
2	2/18/2008	1005200500103	609
2	3/20/2008	1005200500200	574
2	3/23/2008	1005200500212	659
2	4/1/2008	1005200500246	659
2	4/8/2008	1005200500269	624
2	4/15/2008	1005200500295	574
2	5/20/2008	1005200500412	574
2	5/26/2008	1005200500431	574
2	6/29/2008	1005200500547	609
2	7/5/2008	1005200500562	659
2	8/10/2008	1005200500672	624

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. As this field is a calculation, SQL doesn't have 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';
```

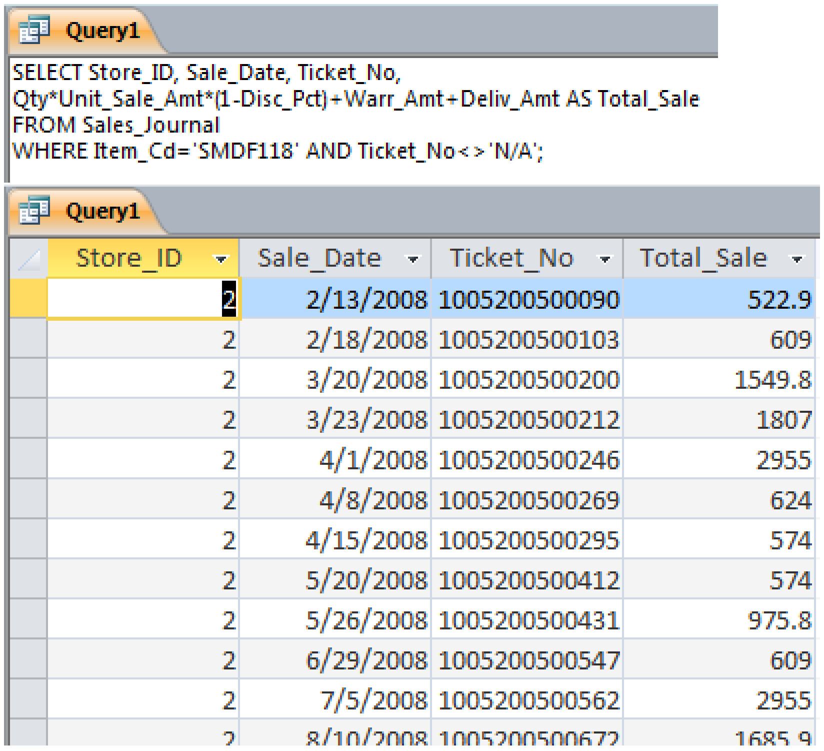
Store_ID	Sale_Date	Ticket_No	Total_Sale
2	2/13/2008	1005200500090	609
2	2/18/2008	1005200500103	609
2	3/20/2008	1005200500200	574
2	3/23/2008	1005200500212	659
2	4/1/2008	1005200500246	659
2	4/8/2008	1005200500269	624
2	4/15/2008	1005200500295	574
2	5/20/2008	1005200500412	574
2	5/26/2008	1005200500431	574
2	6/29/2008	1005200500547	609
2	7/5/2008	1005200500562	659
2	8/10/2008	1005200500672	624

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**.
>
> *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. It is essential that you place the parentheses in the correct places.

4. *Edit the code as shown in* **Figure 14.3** *and run the query.*



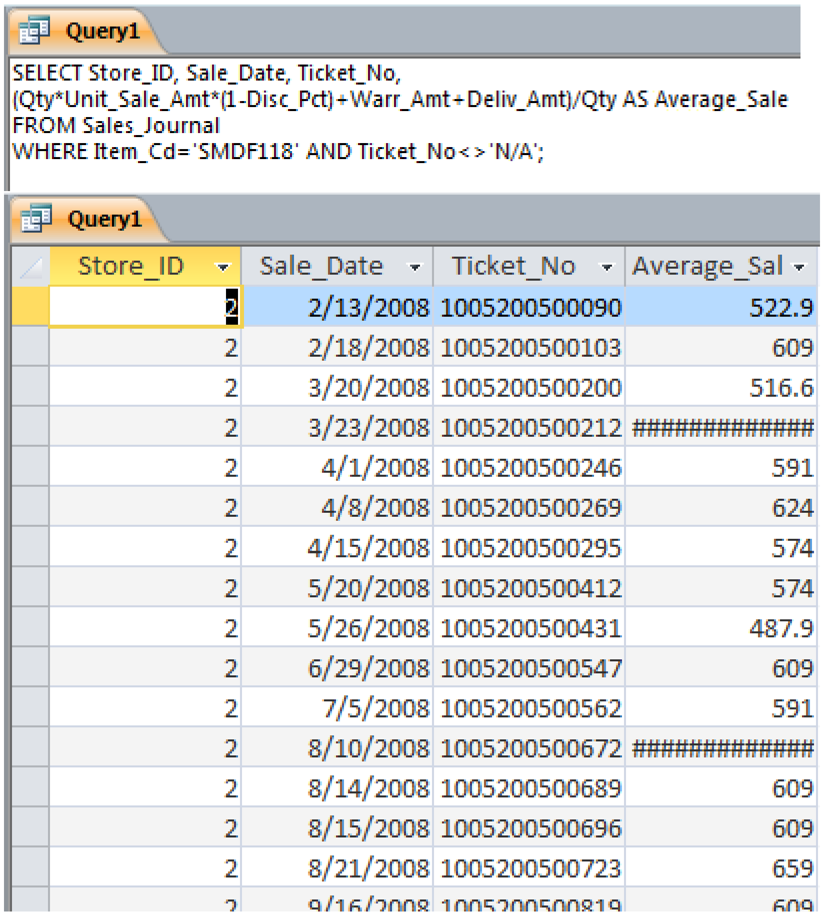
```
SELECT Store_ID, Sale_Date, Ticket_No,
Qty*Unit_Sale_Amt*(1-Disc_Pct)+Warr_Amt+Deliv_Amt AS Total_Sale
FROM Sales_Journal
WHERE Item_Cd='SMDF118' AND Ticket_No<>'N/A';
```

Store_ID	Sale_Date	Ticket_No	Total_Sale
2	2/13/2008	1005200500090	522.9
2	2/18/2008	1005200500103	609
2	3/20/2008	1005200500200	1549.8
2	3/23/2008	1005200500212	1807
2	4/1/2008	1005200500246	2955
2	4/8/2008	1005200500269	624
2	4/15/2008	1005200500295	574
2	5/20/2008	1005200500412	574
2	5/26/2008	1005200500431	975.8
2	6/29/2008	1005200500547	609
2	7/5/2008	1005200500562	2955
2	8/10/2008	1005200500672	1685.9

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.*


Query1

```
SELECT Store_ID, Sale_Date, Ticket_No,
(Qty*Unit_Sale_Amt*(1-Disc_Pct)+Warr_Amt+Deliv_Amt)/Qty AS Average_Sale
FROM Sales_Journal
WHERE Item_Cd='SMDF118' AND Ticket_No<>'N/A';
```

Query1

Store_ID	Sale_Date	Ticket_No	Average_Sal
2	2/13/2008	1005200500090	522.9
2	2/18/2008	1005200500103	609
2	3/20/2008	1005200500200	516.6
2	3/23/2008	1005200500212	##############
2	4/1/2008	1005200500246	591
2	4/8/2008	1005200500269	624
2	4/15/2008	1005200500295	574
2	5/20/2008	1005200500412	574
2	5/26/2008	1005200500431	487.9
2	6/29/2008	1005200500547	609
2	7/5/2008	1005200500562	591
2	8/10/2008	1005200500672	##############
2	8/14/2008	1005200500689	609
2	8/15/2008	1005200500696	609
2	8/21/2008	1005200500723	659
2	9/16/2008	1005200500819	609

Figure 14.4

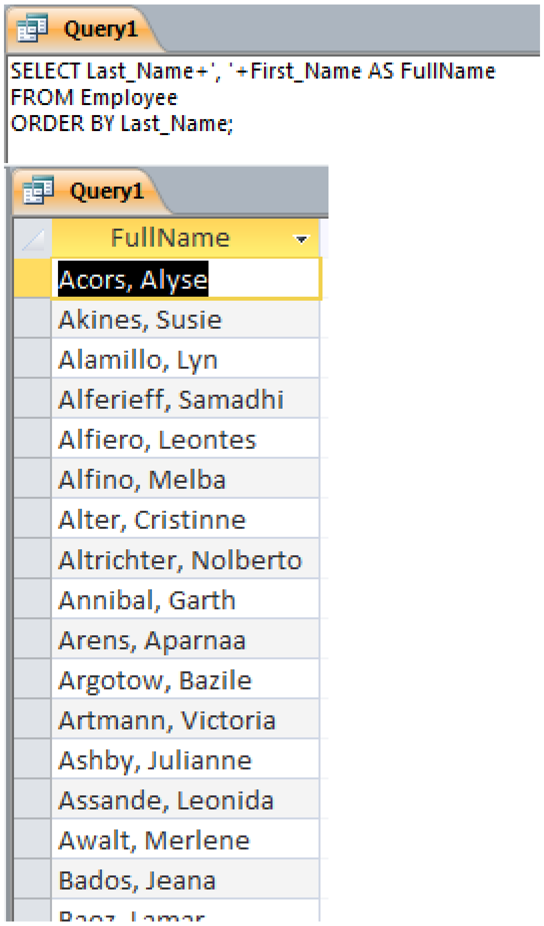
Since the values in the Avg_Sale field are not formatted, the numbers that are too big to fit inside the width of the field are represented as ##############. To fix it, simply adjust the width of the column or apply formatting to the field. Again, with your experience in Excel and Access, you should have no problem understanding mathematical operators and how to use them. In SQL, they work just like they do in Access and Excel.

6. *Save the query as* **qry14Calculations** *and close the query.*

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 14.5** *and run the query.*



Query1

```
SELECT Last_Name+', '+First_Name AS FullName
FROM Employee
ORDER BY Last_Name;
```

Query1

FullName
Acors, Alyse
Akines, Susie
Alamillo, Lyn
Alferieff, Samadhi
Alfiero, Leontes
Alfino, Melba
Alter, Cristinne
Altrichter, Nolberto
Annibal, Garth
Arens, Aparnaa
Argotow, Bazile
Artmann, Victoria
Ashby, Julianne
Assande, Leonida
Awalt, Merlene
Bados, Jeana

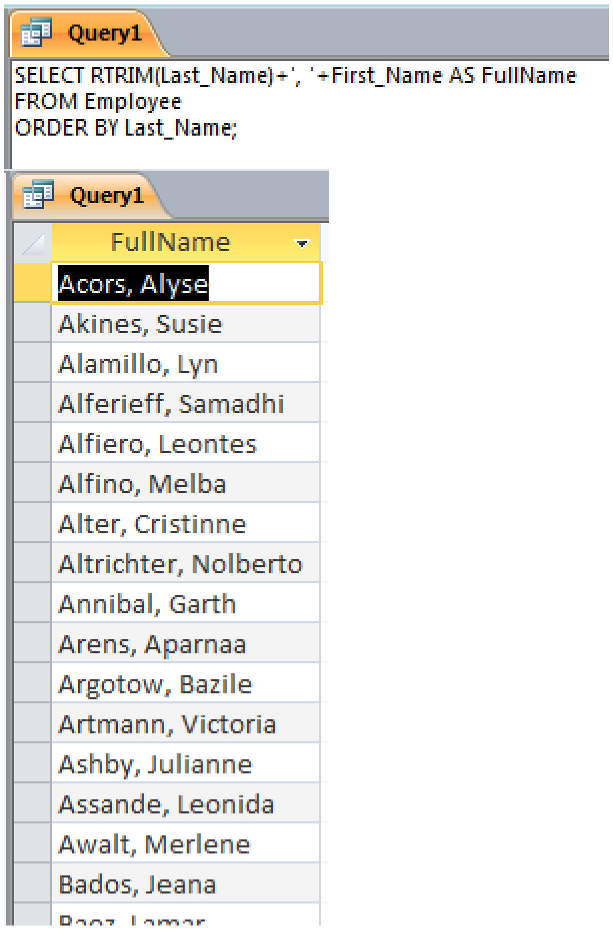
Figure 14.5

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 14.6** *and run it.*



Query1

```
SELECT RTRIM(Last_Name)+', '+First_Name AS FullName
FROM Employee
ORDER BY Last_Name;
```

Query1

FullName
Acors, Alyse
Akines, Susie
Alamillo, Lyn
Alferieff, Samadhi
Alfiero, Leontes
Alfino, Melba
Alter, Cristinne
Altrichter, Nolberto
Annibal, Garth
Arens, Aparnaa
Argotow, Bazile
Artmann, Victoria
Ashby, Julianne
Assande, Leonida
Awalt, Merlene
Bados, Jeana

Figure 14.6

The results should be the same as in Figure 12.5, as there were no spaces after any of the last names.

3. *Save the query as* **qry14Concatenation** *and close.*

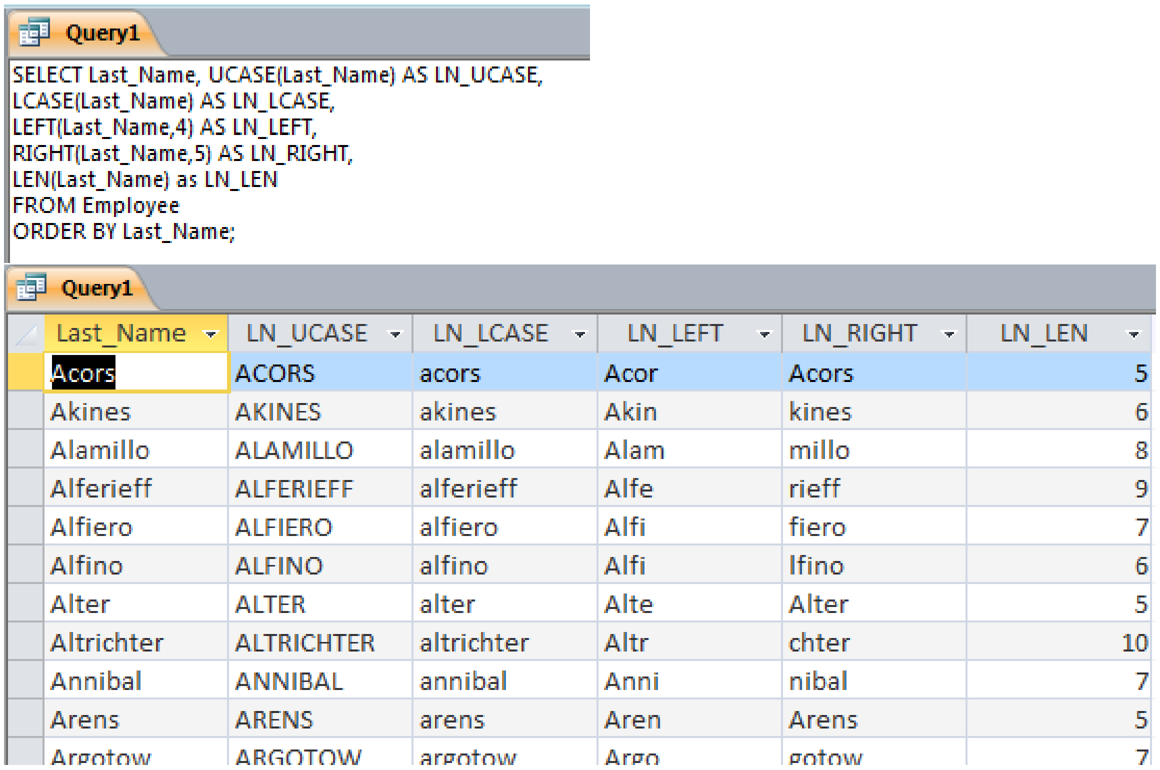
Text Functions

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 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 14.7** *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
Alfiero	ALFIERO	alfiero	Alfi	fiero	7
Alfino	ALFINO	alfino	Alfi	lfino	6
Alter	ALTER	alter	Alte	Alter	5
Altrichter	ALTRICHTER	altrichter	Altr	chter	10
Annibal	ANNIBAL	annibal	Anni	nibal	7
Arens	ARENS	arens	Aren	Arens	5
Argotow	ARGOTOW	argotow	Argo	gotow	7

Figure 14.7

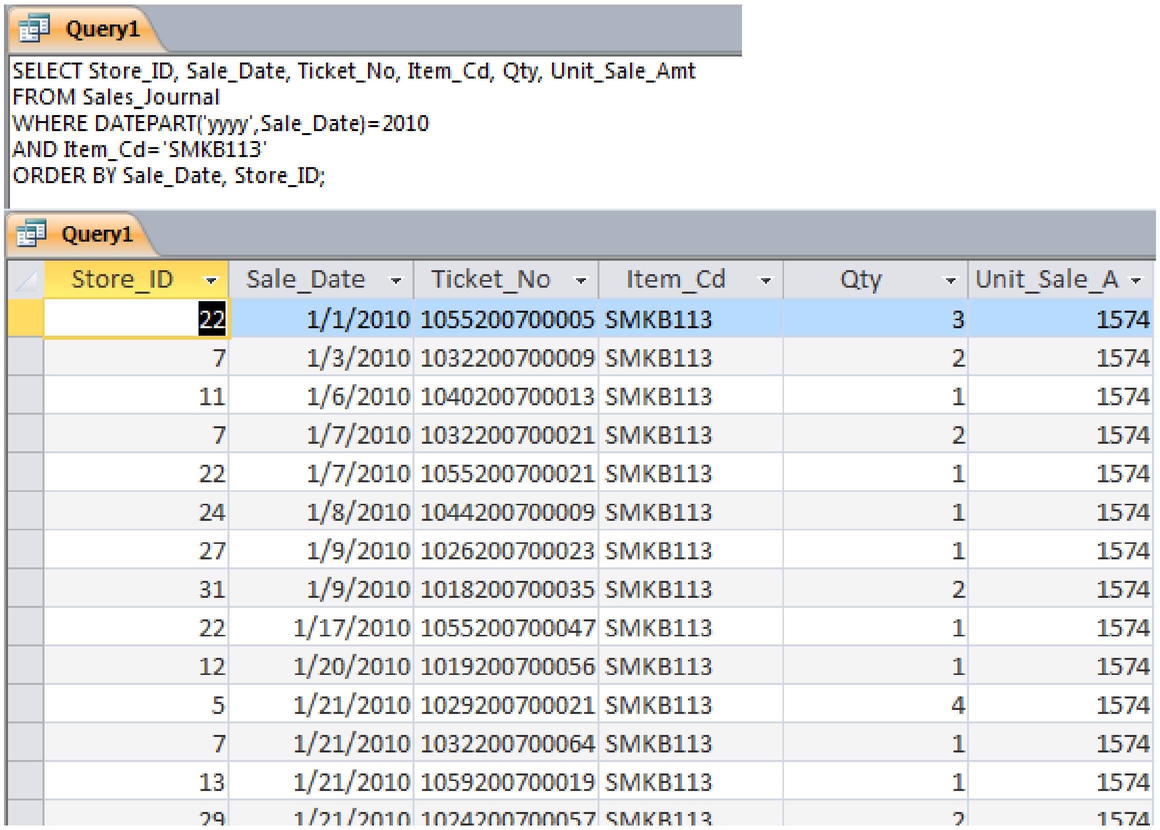
5. *Save the query as* **qry14Text** *and close.*

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 2010 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 14.8** *and run it.*


Query1

```
SELECT Store_ID, Sale_Date, Ticket_No, Item_Cd, Qty, Unit_Sale_Amt
FROM Sales_Journal
WHERE DATEPART('yyyy',Sale_Date)=2010
AND Item_Cd='SMKB113'
ORDER BY Sale_Date, Store_ID;
```

Query1

Store_ID	Sale_Date	Ticket_No	Item_Cd	Qty	Unit_Sale_A
22	1/1/2010	1055200700005	SMKB113	3	1574
7	1/3/2010	1032200700009	SMKB113	2	1574
11	1/6/2010	1040200700013	SMKB113	1	1574
7	1/7/2010	1032200700021	SMKB113	2	1574
22	1/7/2010	1055200700021	SMKB113	1	1574
24	1/8/2010	1044200700009	SMKB113	1	1574
27	1/9/2010	1026200700023	SMKB113	1	1574
31	1/9/2010	1018200700035	SMKB113	2	1574
22	1/17/2010	1055200700047	SMKB113	1	1574
12	1/20/2010	1019200700056	SMKB113	1	1574
5	1/21/2010	1029200700021	SMKB113	4	1574
7	1/21/2010	1032200700064	SMKB113	1	1574
13	1/21/2010	1059200700019	SMKB113	1	1574
29	1/21/2010	1024200700057	SMKB113	2	1574

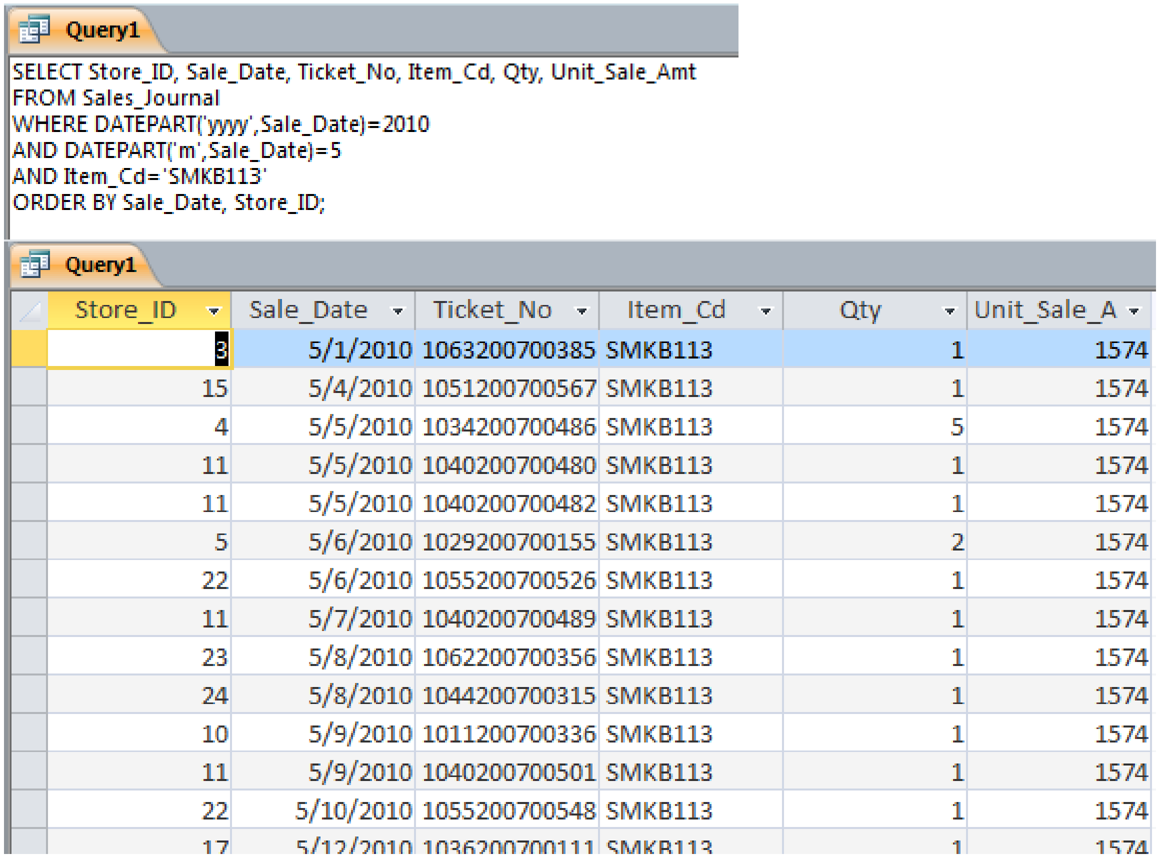
Figure 14.8

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)=2010

To return the record set of all records in May 2010, 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 14.9** *and run it.*



Query1

```
SELECT Store_ID, Sale_Date, Ticket_No, Item_Cd, Qty, Unit_Sale_Amt
FROM Sales_Journal
WHERE DATEPART('yyyy',Sale_Date)=2010
AND DATEPART('m',Sale_Date)=5
AND Item_Cd='SMKB113'
ORDER BY Sale_Date, Store_ID;
```

Query1

Store_ID	Sale_Date	Ticket_No	Item_Cd	Qty	Unit_Sale_A
3	5/1/2010	1063200700385	SMKB113	1	1574
15	5/4/2010	1051200700567	SMKB113	1	1574
4	5/5/2010	1034200700486	SMKB113	5	1574
11	5/5/2010	1040200700480	SMKB113	1	1574
11	5/5/2010	1040200700482	SMKB113	1	1574
5	5/6/2010	1029200700155	SMKB113	2	1574
22	5/6/2010	1055200700526	SMKB113	1	1574
11	5/7/2010	1040200700489	SMKB113	1	1574
23	5/8/2010	1062200700356	SMKB113	1	1574
24	5/8/2010	1044200700315	SMKB113	1	1574
10	5/9/2010	1011200700336	SMKB113	1	1574
11	5/9/2010	1040200700501	SMKB113	1	1574
22	5/10/2010	1055200700548	SMKB113	1	1574
17	5/12/2010	1036200700111	SMKB113	1	1574

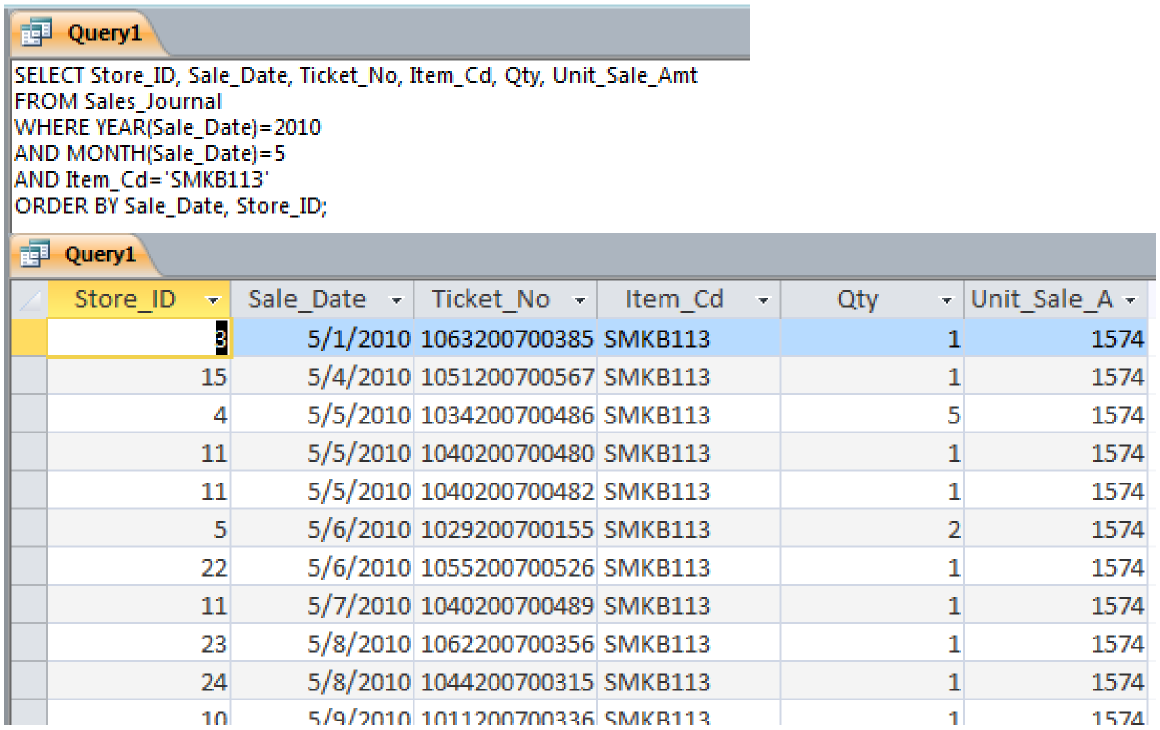
Figure 14.9

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 14.10** *and run it.*



Query1

```
SELECT Store_ID, Sale_Date, Ticket_No, Item_Cd, Qty, Unit_Sale_Amt
FROM Sales_Journal
WHERE YEAR(Sale_Date)=2010
AND MONTH(Sale_Date)=5
AND Item_Cd='SMKB113'
ORDER BY Sale_Date, Store_ID;
```

Query1

Store_ID	Sale_Date	Ticket_No	Item_Cd	Qty	Unit_Sale_A
3	5/1/2010	1063200700385	SMKB113	1	1574
15	5/4/2010	1051200700567	SMKB113	1	1574
4	5/5/2010	1034200700486	SMKB113	5	1574
11	5/5/2010	1040200700480	SMKB113	1	1574
11	5/5/2010	1040200700482	SMKB113	1	1574
5	5/6/2010	1029200700155	SMKB113	2	1574
22	5/6/2010	1055200700526	SMKB113	1	1574
11	5/7/2010	1040200700489	SMKB113	1	1574
23	5/8/2010	1062200700356	SMKB113	1	1574
24	5/8/2010	1044200700315	SMKB113	1	1574
10	5/9/2010	1011200700336	SMKB113	1	1574

Figure 14.10

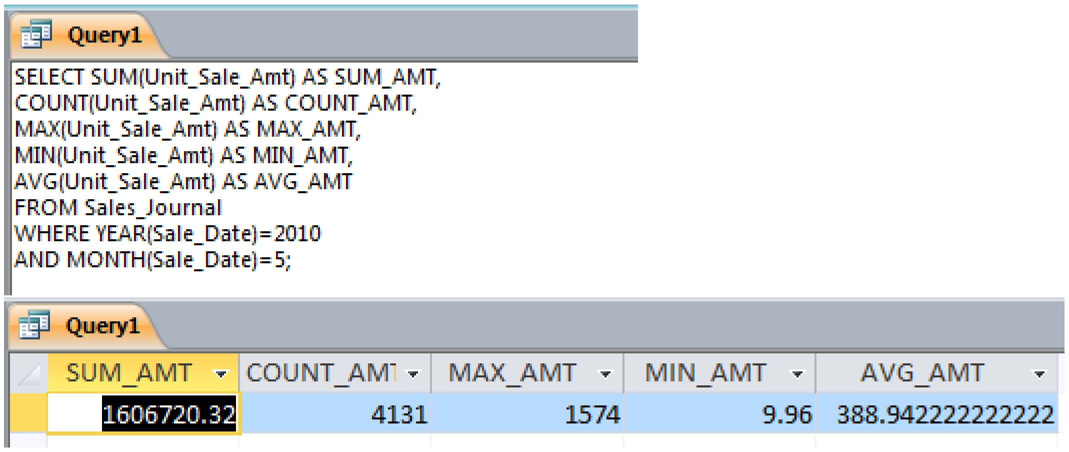
4. *Save the query as* **qry14Dates** *and close it.*

   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 2010 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 2010.

5. *Create a new* **SQL** *query, type in the code as in* **Figure 14.11** *and run it.*



Query1

```
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)=2010
AND MONTH(Sale_Date)=5;
```

Query1

SUM_AMT	COUNT_AMT	MAX_AMT	MIN_AMT	AVG_AMT
1606720.32	4131	1574	9.96	388.942222222222

Figure 14.11

You should get one record that looks like Figure 14.11. 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** *and close it.*

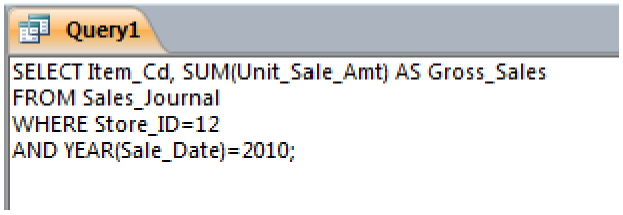
Grouping

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 2010.

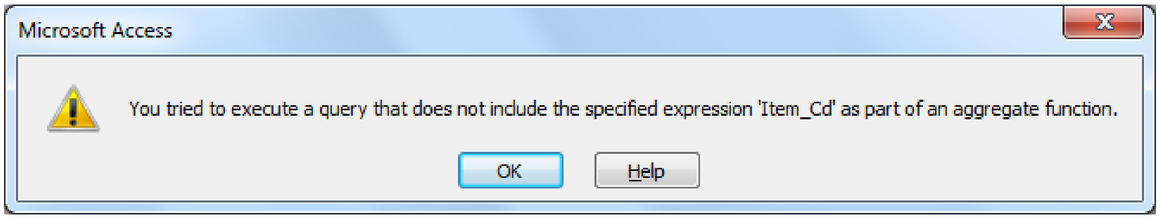
1. *Create a new* **SQL** *query, type in the code as in* **Figure 14.12** *and run it.*



```
SELECT Item_Cd, SUM(Unit_Sale_Amt) AS Gross_Sales
FROM Sales_Journal
WHERE Store_ID=12
AND YEAR(Sale_Date)=2010;
```

Figure 14.12

When you run this code, you will get an error message that looks like this:

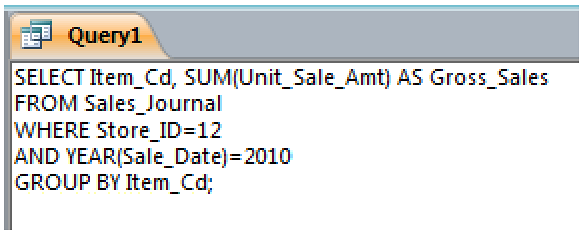


You tried to execute a query that does not include the specified expression 'Item_Cd' as part of an aggregate function.

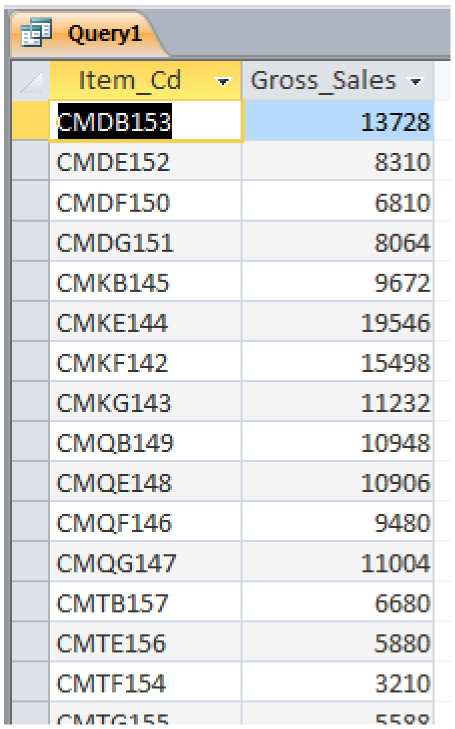
Figure 14.13

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 14.14** *and run it.*



```
SELECT Item_Cd, SUM(Unit_Sale_Amt) AS Gross_Sales
FROM Sales_Journal
WHERE Store_ID=12
AND YEAR(Sale_Date)=2010
GROUP BY Item_Cd;
```



Item_Cd	Gross_Sales
CMDB153	13728
CMDE152	8310
CMDF150	6810
CMDG151	8064
CMKB145	9672
CMKE144	19546
CMKF142	15498
CMKG143	11232
CMQB149	10948
CMQE148	10906
CMQF146	9480
CMQG147	11004
CMTB157	6680
CMTE156	5880
CMTF154	3210
CMTG155	5588

Figure 14.14

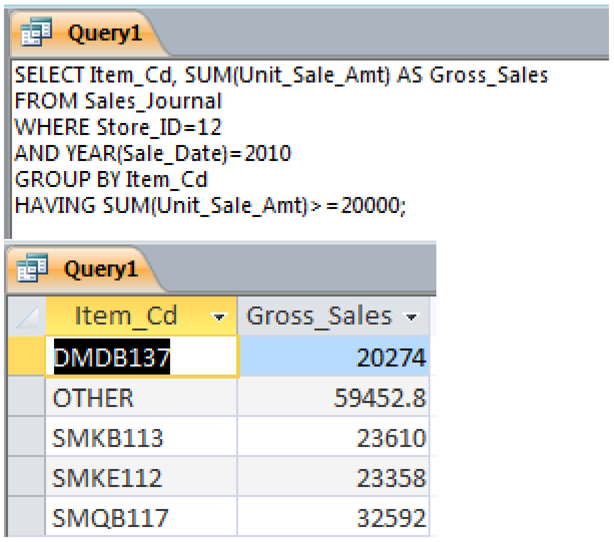
There should have been 69 records returned by this query. Because you used the GROUP BY clause, the query automatically included all items. You could have used an IN() function to include specific items, or if you wanted certain items excluded from the analysis. Here are a few rules that you need to 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 14.15** *and run it.*



Query1

```
SELECT Item_Cd, SUM(Unit_Sale_Amt) AS Gross_Sales
FROM Sales_Journal
WHERE Store_ID=12
AND YEAR(Sale_Date)=2010
GROUP BY Item_Cd
HAVING SUM(Unit_Sale_Amt)>=20000;
```

Query1

Item_Cd	Gross_Sales
DMDB137	20274
OTHER	59452.8
SMKB113	23610
SMKE112	23358
SMQB117	32592

Figure 14.15

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.

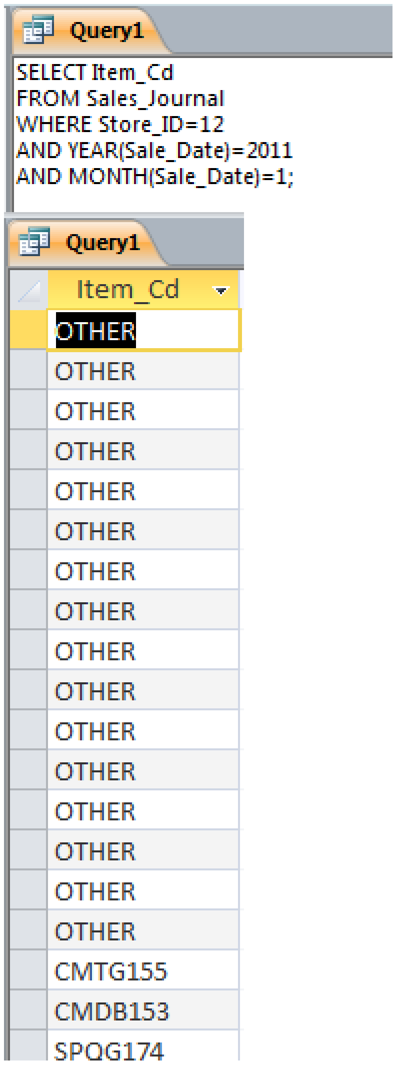
Additionally, don't forget the ORDER BY clause. Whenever you group data in a GROUP BY clause with a WHERE and/or HAVING clauses, it is probably a good idea to include an ORDER BY clause. Don't depend on the GROUP BY clause to sort your data for you. Sometimes a GROUP BY statement will order it correctly, 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 2011.

6. *Create a new* **SQL** *query, type in the code as in* **Figure 14.16** *and run it.*



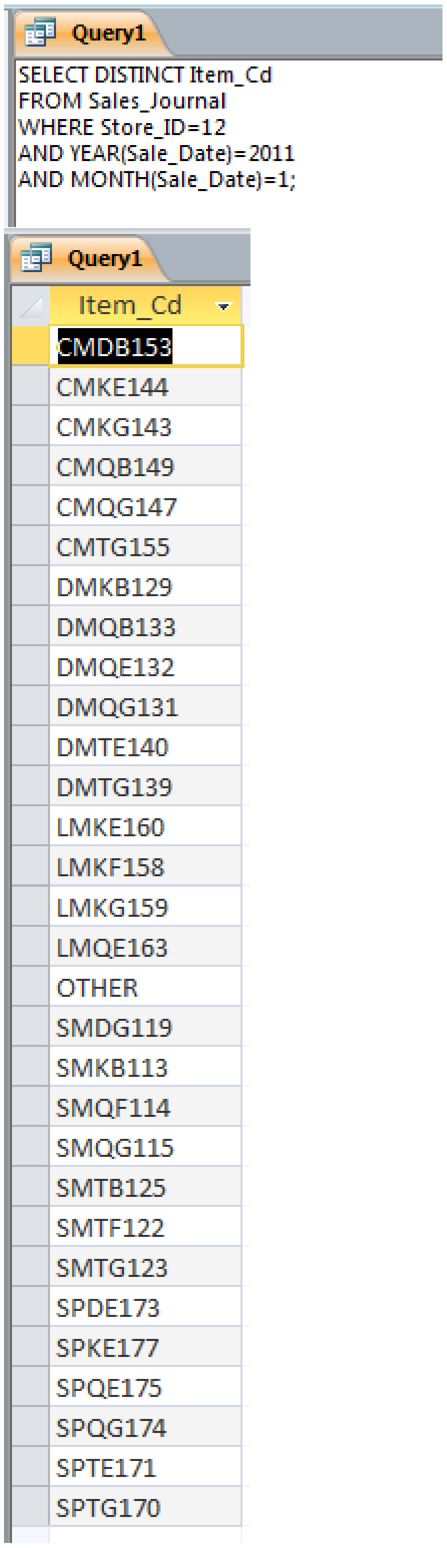
```
SELECT Item_Cd
FROM Sales_Journal
WHERE Store_ID=12
AND YEAR(Sale_Date)=2011
AND MONTH(Sale_Date)=1;
```

Item_Cd
OTHER
OTHER
OTHER
OTHER
OTHER
OTHER
OTHER
OTHER
OTHER
OTHER
OTHER
OTHER
OTHER
OTHER
OTHER
OTHER
CMTG155
CMDB153
SPQG174

Figure 14.16

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 14.17** *and run it.*



```
SELECT DISTINCT Item_Cd
FROM Sales_Journal
WHERE Store_ID=12
AND YEAR(Sale_Date)=2011
AND MONTH(Sale_Date)=1;
```

Item_Cd
CMDB153
CMKE144
CMKG143
CMQB149
CMQG147
CMTG155
DMKB129
DMQB133
DMQE132
DMQG131
DMTE140
DMTG139
LMKE160
LMKF158
LMKG159
LMQE163
OTHER
SMDG119
SMKB113
SMQF114
SMQG115
SMTB125
SMTF122
SMTG123
SPDE173
SPKE177
SPQE175
SPQG174
SPTE171
SPTG170

Figure 14.17

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 grouping.

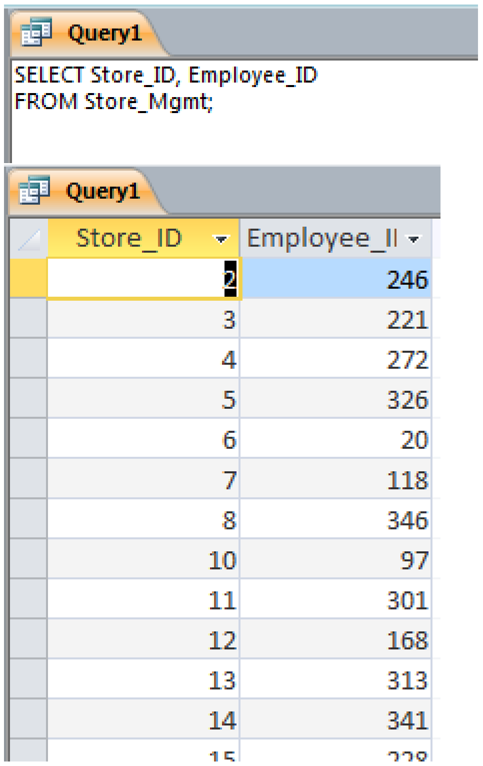
8. *Save the query as* **qry14Distinct** *and close it.*

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 14.18** *and run it.*



```
SELECT Store_ID, Employee_ID
FROM Store_Mgmt;
```

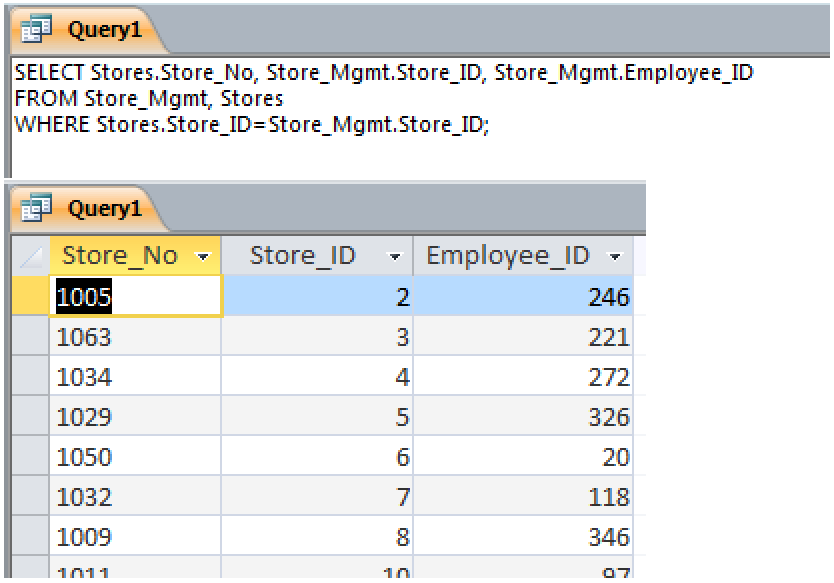
Store_ID	Employee_II
2	246
3	221
4	272
5	326
6	20
7	118
8	346
10	97
11	301
12	168
13	313
14	341
15	228

Figure 14.18

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 typically all you have to type is the DATABASE.TABLE, 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 14.19** *and run it.*



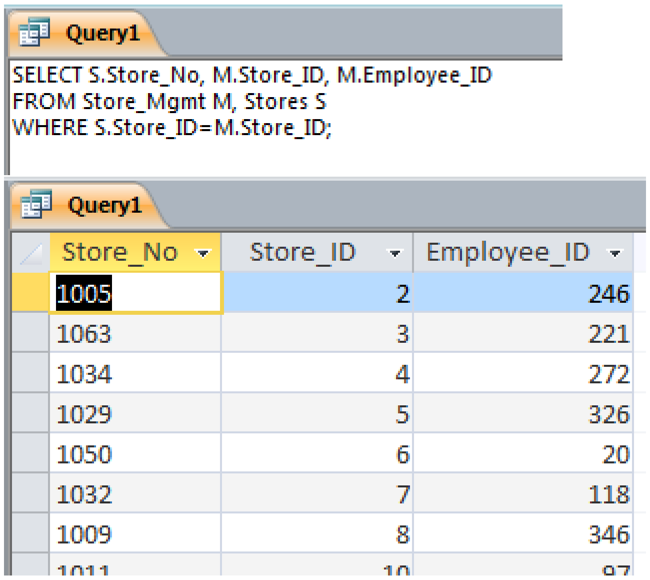
```
SELECT Stores.Store_No, Store_Mgmt.Store_ID, Store_Mgmt.Employee_ID
FROM Store_Mgmt, Stores
WHERE Stores.Store_ID=Store_Mgmt.Store_ID;
```

Store_No	Store_ID	Employee_ID
1005	2	246
1063	3	221
1034	4	272
1029	5	326
1050	6	20
1032	7	118
1009	8	346
1011	10	97

Figure 14.19

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 14.20** *and run it.*


Query1

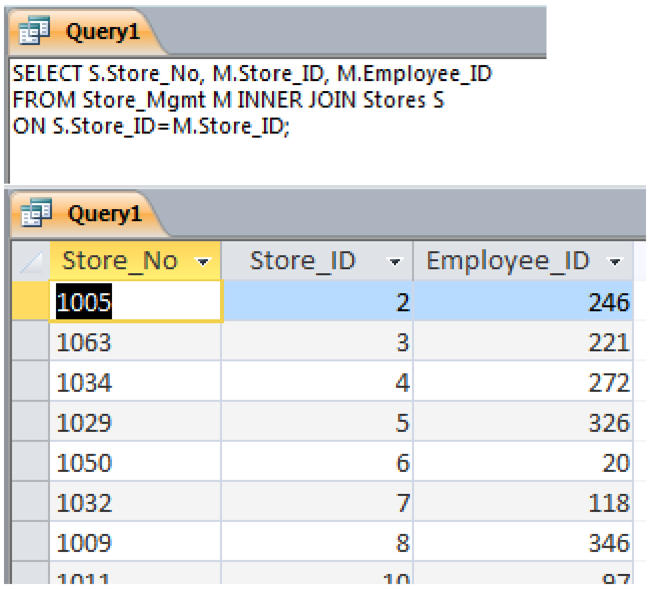
```
SELECT S.Store_No, M.Store_ID, M.Employee_ID
FROM Store_Mgmt M, Stores S
WHERE S.Store_ID=M.Store_ID;
```

Store_No	Store_ID	Employee_ID
1005	2	246
1063	3	221
1034	4	272
1029	5	326
1050	6	20
1032	7	118
1009	8	346
1011	10	97

Figure 14.20

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 14.21** *and run it.*


Query1

```
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_No	Store_ID	Employee_ID
1005	2	246
1063	3	221
1034	4	272
1029	5	326
1050	6	20
1032	7	118
1009	8	346
1011	10	97

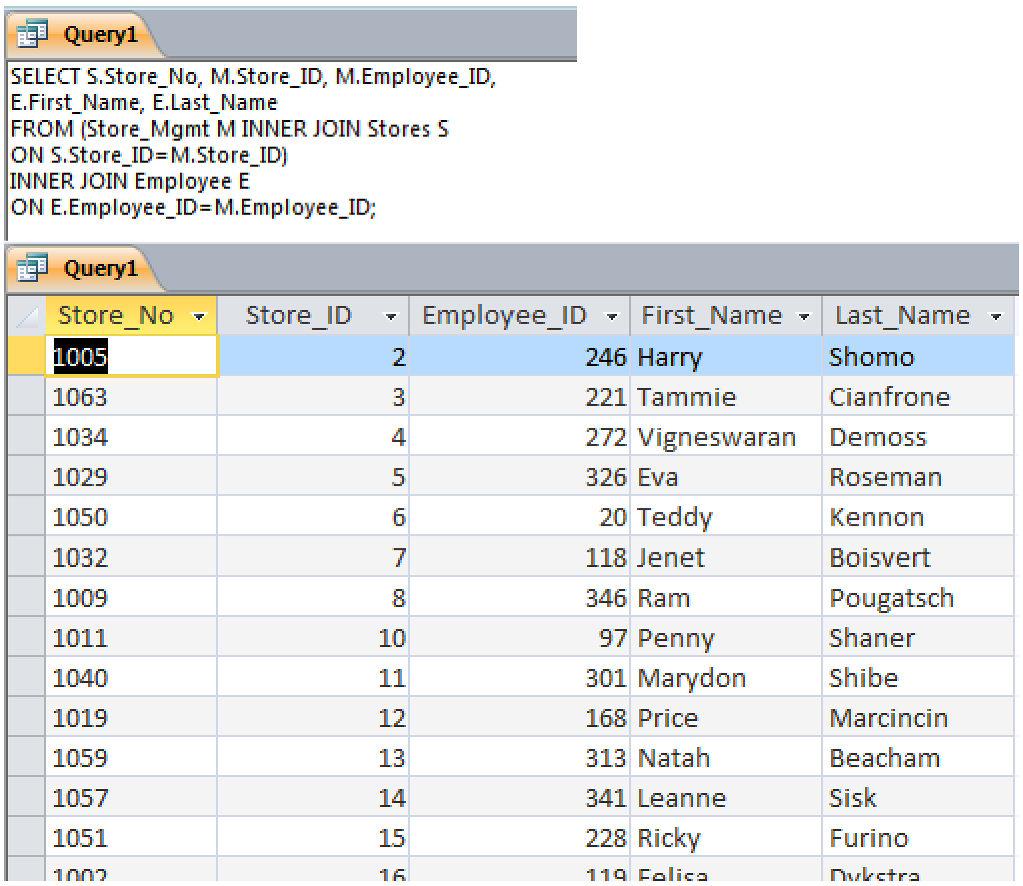
Figure 14.21

You should get exactly the same results as in Figure 14.20 and 14.21.

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 14.22** *and run it.*



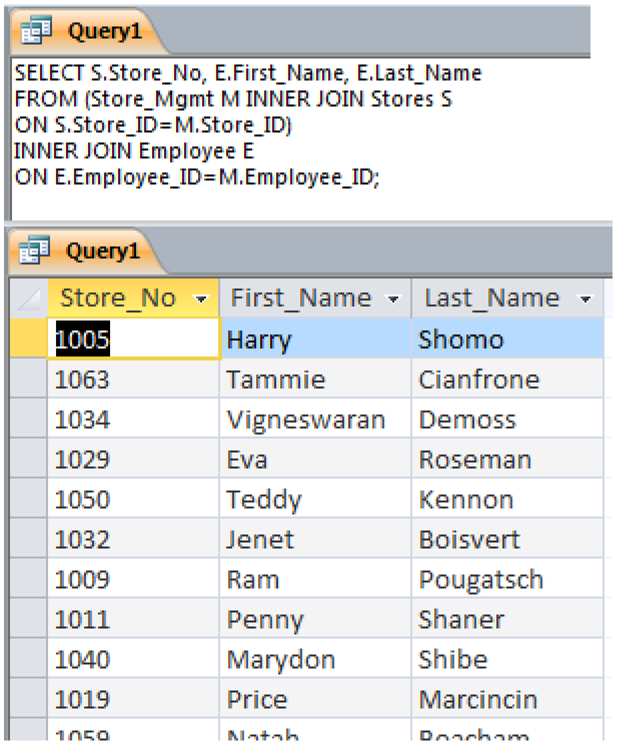
```
SELECT S.Store_No, M.Store_ID, M.Employee_ID,
E.First_Name, E.Last_Name
FROM (Store_Mgmt M INNER JOIN Stores S
ON S.Store_ID=M.Store_ID)
INNER JOIN Employee E
ON E.Employee_ID=M.Employee_ID;
```

Store_No	Store_ID	Employee_ID	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
1032	7	118	Jenet	Boisvert
1009	8	346	Ram	Pougatsch
1011	10	97	Penny	Shaner
1040	11	301	Marydon	Shibe
1019	12	168	Price	Marcincin
1059	13	313	Natah	Beacham
1057	14	341	Leanne	Sisk
1051	15	228	Ricky	Furino
1002	16	119	Felisa	Dykstra

Figure 14.22

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 14.23** *and run it.*


Query1

```
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;
```

Query1

Store_No	First_Name	Last_Name
1005	Harry	Shomo
1063	Tammie	Cianfrone
1034	Vigneswaran	Demoss
1029	Eva	Roseman
1050	Teddy	Kennon
1032	Jenet	Boisvert
1009	Ram	Pougatsch
1011	Penny	Shaner
1040	Marydon	Shibe
1019	Price	Marcincin
1059	Natah	Beacham

Figure 14.23

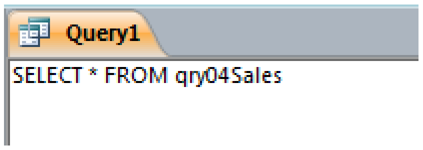
Now notice in this query that the primary table is the Store_Mgmt table, as that is the table that contains all the necessary information, yet we're not bringing in ANY of the fields from that table. This kind of table is referred to as a **Union table**, or a table that brings the necessary information together in one query.

7. *Save the query as* **qry14Joins** *and close it.*

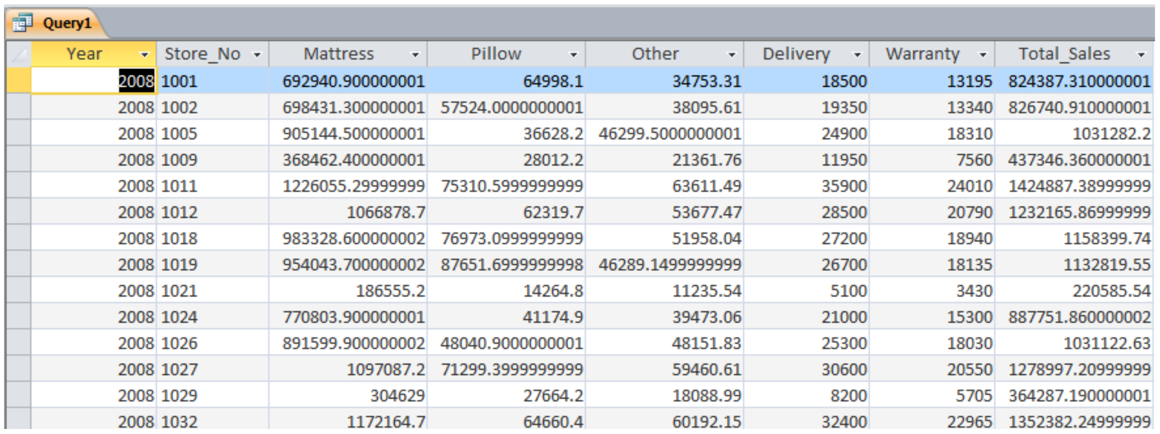
One Way (or Outer) Joins

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 14.24** *and run it.*


Query1

```
SELECT * FROM qry04Sales
```



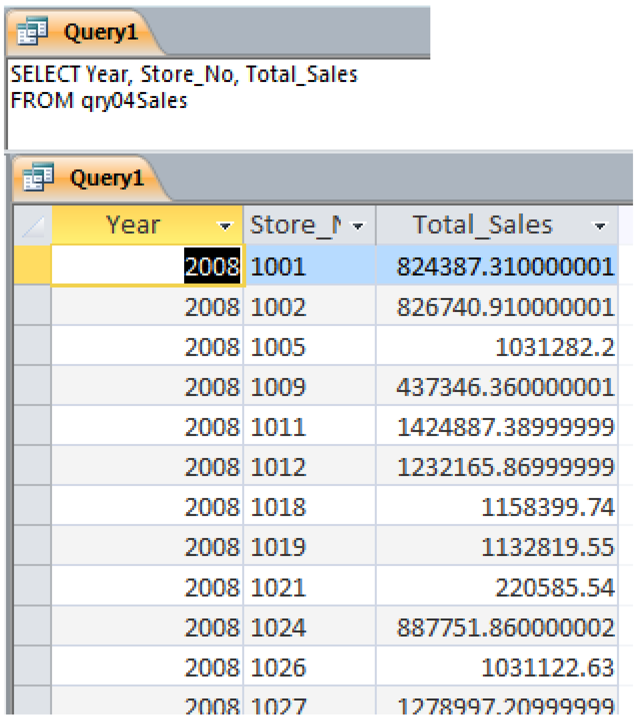
Query1

Year	Store_No	Mattress	Pillow	Other	Delivery	Warranty	Total_Sales
2008	1001	692940.900000001	64998.1	34753.31	18500	13195	824387.310000001
2008	1002	698431.300000001	57524.0000000001	38095.61	19350	13340	826740.910000001
2008	1005	905144.500000001	36628.2	46299.5000000001	24900	18310	1031282.2
2008	1009	368462.400000001	28012.2	21361.76	11950	7560	437346.360000001
2008	1011	1226055.29999999	75310.5999999999	63611.49	35900	24010	1424887.38999999
2008	1012	1066878.7	62319.7	53677.47	28500	20790	1232165.86999999
2008	1018	983328.600000002	76973.0999999999	51958.04	27200	18940	1158399.74
2008	1019	954043.700000002	87651.6999999998	46289.1499999999	26700	18135	1132819.55
2008	1021	186555.2	14264.8	11235.54	5100	3430	220585.54
2008	1024	770803.900000001	41174.9	39473.06	21000	15300	887751.860000002
2008	1026	891599.900000002	48040.9000000001	48151.83	25300	18030	1031122.63
2008	1027	1097087.2	71299.3999999999	59460.61	30600	20550	1278997.20999999
2008	1029	304629	27664.2	18088.99	8200	5705	364287.190000001
2008	1032	1172164.7	64660.4	60192.15	32400	22965	1352382.24999999

Figure 14.24

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 14.25** *and run it.*



Query1

```
SELECT Year, Store_No, Total_Sales
FROM qry04Sales
```

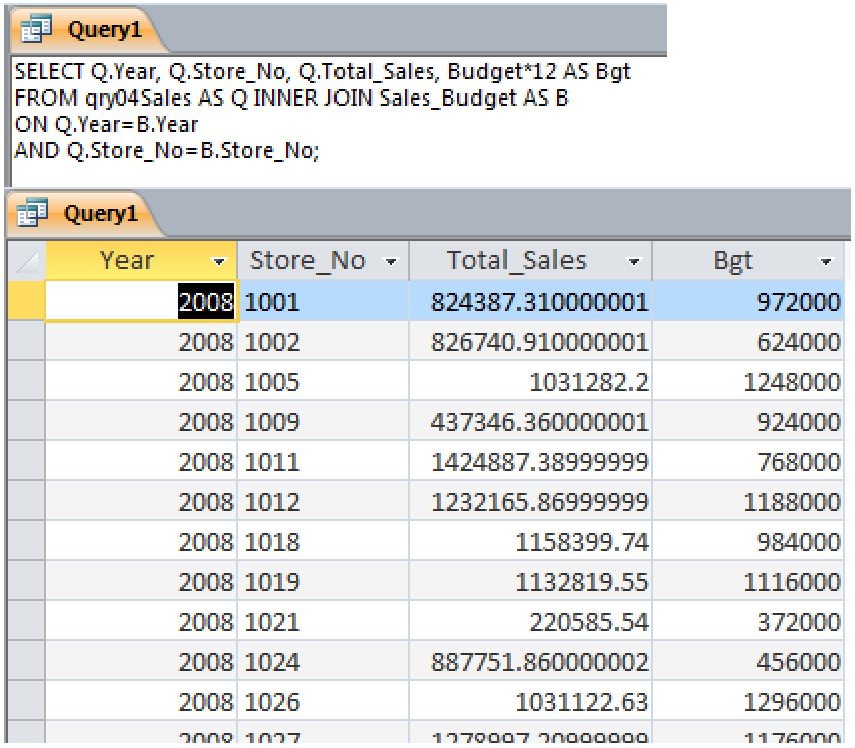
Query1

Year	Store_N	Total_Sales
2008	1001	824387.310000001
2008	1002	826740.910000001
2008	1005	1031282.2
2008	1009	437346.360000001
2008	1011	1424887.38999999
2008	1012	1232165.86999999
2008	1018	1158399.74
2008	1019	1132819.55
2008	1021	220585.54
2008	1024	887751.860000002
2008	1026	1031122.63
2008	1027	1278997.20999999

Figure 14.25

The dataset still contains 87 records. Now you want to 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 14.26** *and run it.*


Query1

```
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;
```

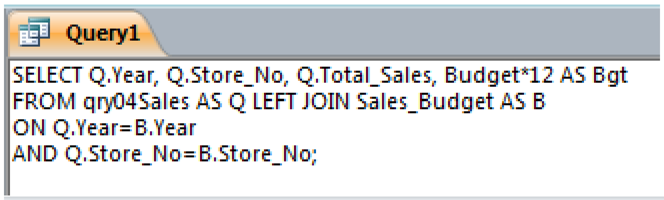
Query1

Year	Store_No	Total_Sales	Bgt
2008	1001	824387.310000001	972000
2008	1002	826740.910000001	624000
2008	1005	1031282.2	1248000
2008	1009	437346.360000001	924000
2008	1011	1424887.38999999	768000
2008	1012	1232165.86999999	1188000
2008	1018	1158399.74	984000
2008	1019	1132819.55	1116000
2008	1021	220585.54	372000
2008	1024	887751.860000002	456000
2008	1026	1031122.63	1296000
2008	1027	1278997.20999999	1176000

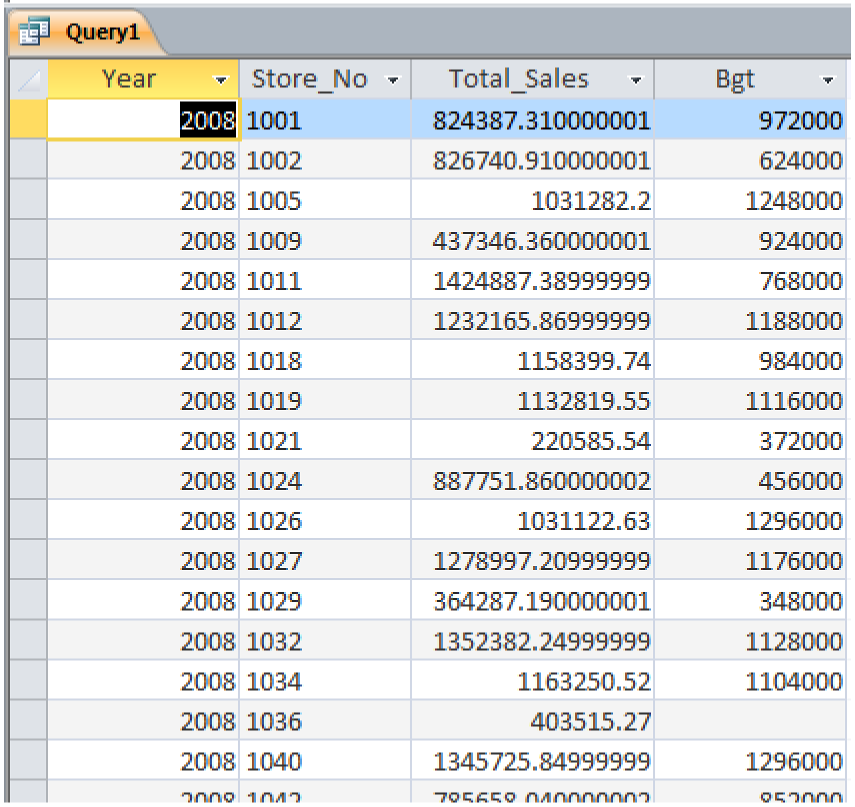
Figure 14.26

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 14.27** *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
2008	1001	824387.310000001	972000
2008	1002	826740.910000001	624000
2008	1005	1031282.2	1248000
2008	1009	437346.360000001	924000
2008	1011	1424887.38999999	768000
2008	1012	1232165.86999999	1188000
2008	1018	1158399.74	984000
2008	1019	1132819.55	1116000
2008	1021	220585.54	372000
2008	1024	887751.860000002	456000
2008	1026	1031122.63	1296000
2008	1027	1278997.20999999	1176000
2008	1029	364287.190000001	348000
2008	1032	1352382.24999999	1128000
2008	1034	1163250.52	1104000
2008	1036	403515.27	
2008	1040	1345725.84999999	1296000
2008	1042	785658.040000002	852000

Figure 14.27

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.

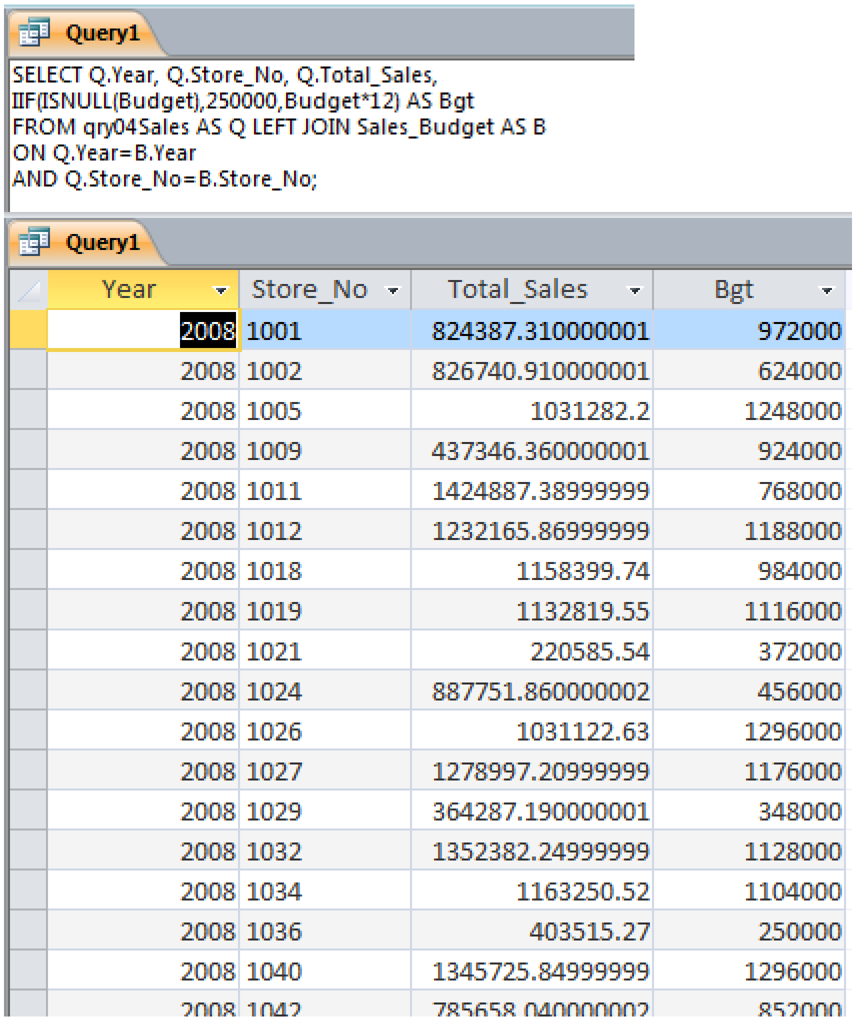
Some older versions of SQL allow you to do the one-way join in the WHERE clause where the syntax is as follows:

```
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 14.28** *and run it.*



Query1

```
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;
```

Query1

Year	Store_No	Total_Sales	Bgt
2008	1001	824387.310000001	972000
2008	1002	826740.910000001	624000
2008	1005	1031282.2	1248000
2008	1009	437346.360000001	924000
2008	1011	1424887.38999999	768000
2008	1012	1232165.86999999	1188000
2008	1018	1158399.74	984000
2008	1019	1132819.55	1116000
2008	1021	220585.54	372000
2008	1024	887751.860000002	456000
2008	1026	1031122.63	1296000
2008	1027	1278997.20999999	1176000
2008	1029	364287.190000001	348000
2008	1032	1352382.24999999	1128000
2008	1034	1163250.52	1104000
2008	1036	403515.27	250000
2008	1040	1345725.84999999	1296000
2008	1042	785658.040000002	852000

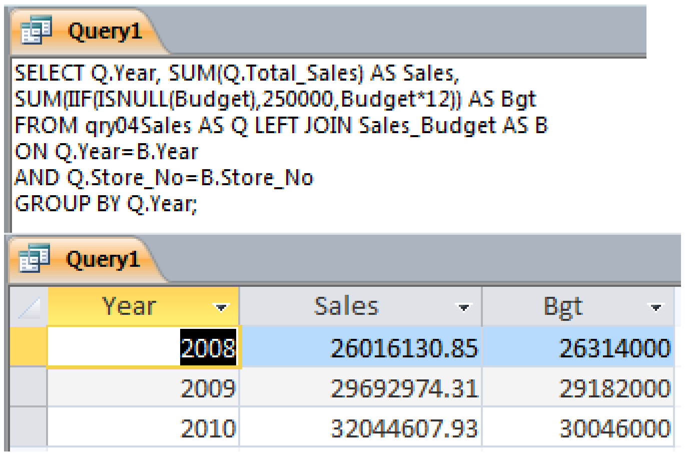
Figure 14.28

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 14.29** *and run it.*



Query1

```
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;
```

Query1

Year	Sales	Bgt
2008	26016130.85	26314000
2009	29692974.31	29182000
2010	32044607.93	30046000

Figure 14.29

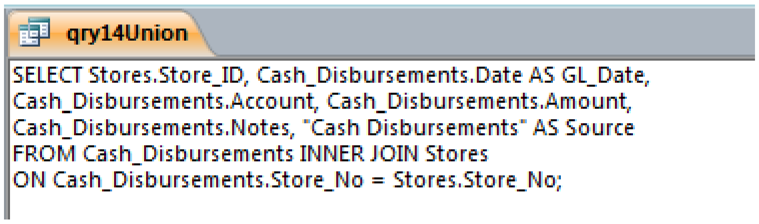
14. *Save the query as* **qry14OneWay_IIF** *and close.*

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.*



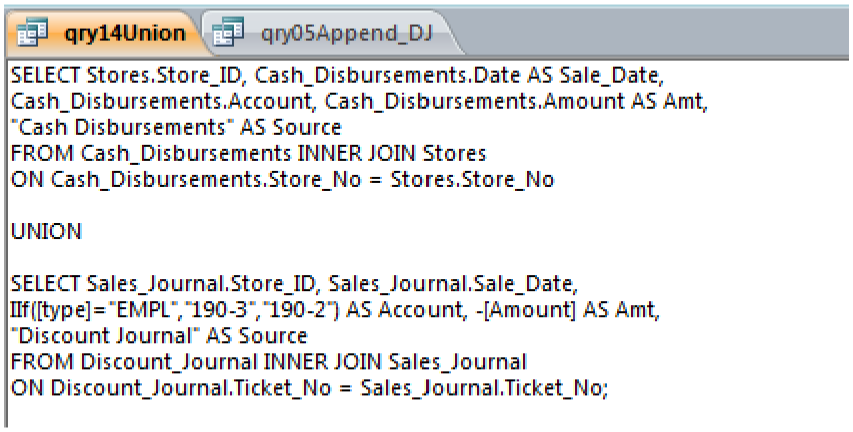
```
SELECT Stores.Store_ID, Cash_Disbursements.Date AS GL_Date,
Cash_Disbursements.Account, Cash_Disbursements.Amount,
Cash_Disbursements.Notes, "Cash Disbursements" AS Source
FROM Cash_Disbursements INNER JOIN Stores
ON Cash_Disbursements.Store_No = Stores.Store_No;
```

Figure 14.30

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 14.31**.

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** *to* **Sale_Date**.
8. *Make the* **Cash_Disbursements.Amount** *field have an alias of "***Amt***".*
9. *Delete the* **Cash_Disbursements.Notes** *field.*

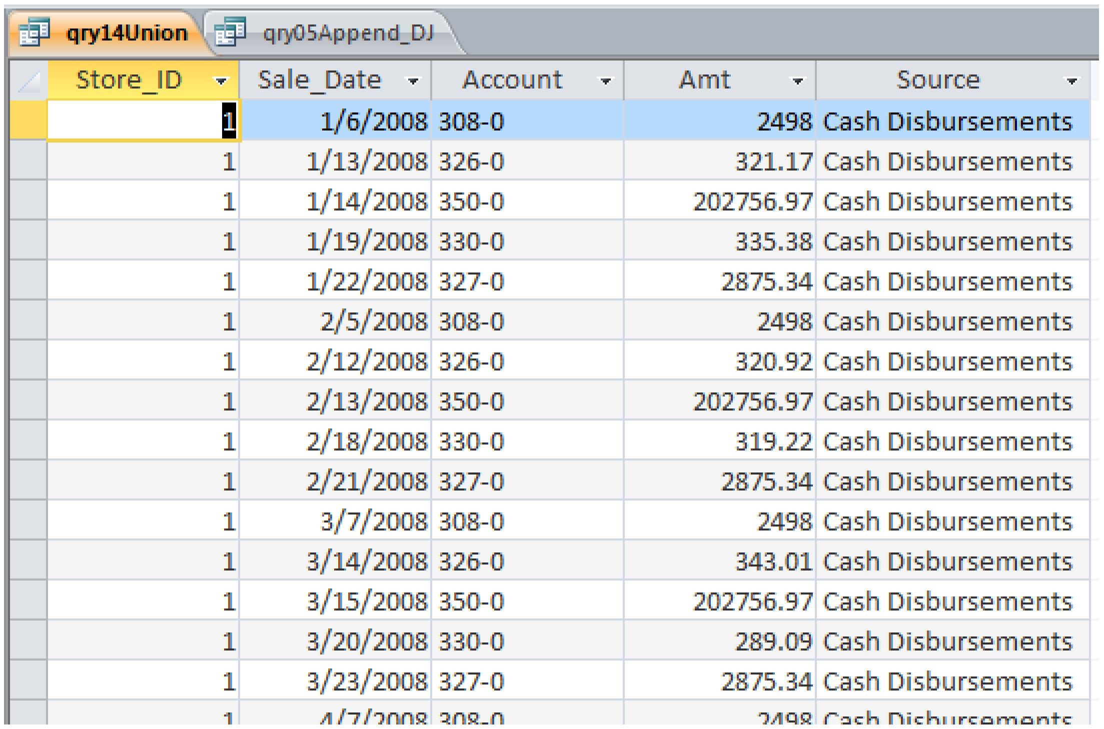


```
SELECT Stores.Store_ID, Cash_Disbursements.Date AS Sale_Date,
Cash_Disbursements.Account, Cash_Disbursements.Amount AS Amt,
"Cash Disbursements" AS Source
FROM Cash_Disbursements INNER JOIN Stores
ON Cash_Disbursements.Store_No = Stores.Store_No

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 14.31

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.**



qry14Union | qry05Append_DJ

Store_ID	Sale_Date	Account	Amt	Source
1	1/6/2008	308-0	2498	Cash Disbursements
1	1/13/2008	326-0	321.17	Cash Disbursements
1	1/14/2008	350-0	202756.97	Cash Disbursements
1	1/19/2008	330-0	335.38	Cash Disbursements
1	1/22/2008	327-0	2875.34	Cash Disbursements
1	2/5/2008	308-0	2498	Cash Disbursements
1	2/12/2008	326-0	320.92	Cash Disbursements
1	2/13/2008	350-0	202756.97	Cash Disbursements
1	2/18/2008	330-0	319.22	Cash Disbursements
1	2/21/2008	327-0	2875.34	Cash Disbursements
1	3/7/2008	308-0	2498	Cash Disbursements
1	3/14/2008	326-0	343.01	Cash Disbursements
1	3/15/2008	350-0	202756.97	Cash Disbursements
1	3/20/2008	330-0	289.09	Cash Disbursements
1	3/23/2008	327-0	2875.34	Cash Disbursements
1	4/7/2008	308-0	2498	Cash Disbursements

Figure 14.32

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 2010 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, an accountant will not perform advanced procedures such as triggers and inserting, editing, and deleting data from a SQL Server database. That is typically performed by the Database Administrators (DBAs). However, there is a great need for an accountant 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, log in and take the exam. Make sure that you take the exam on the same computer that you completed the sample files on, as some of the questions on the exam may refer to some of the completed examples.

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.

Excel and Microsoft Query

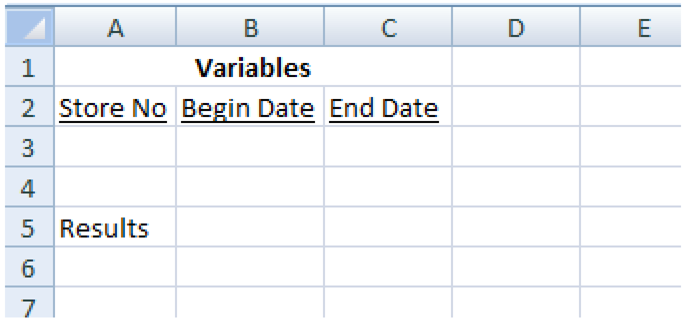
In the last couple of chapters, you've seen how to write SQL code in an Access database, and you may have thought to yourself, "*I don't see a whole lot of need for SQL in Access since you can do most of the query design using the Design tool.*" If you thought that, you are mostly right. The Access query 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 spreadsheet.*
2. *Make the spreadsheet look like* **Figure 15.1**.



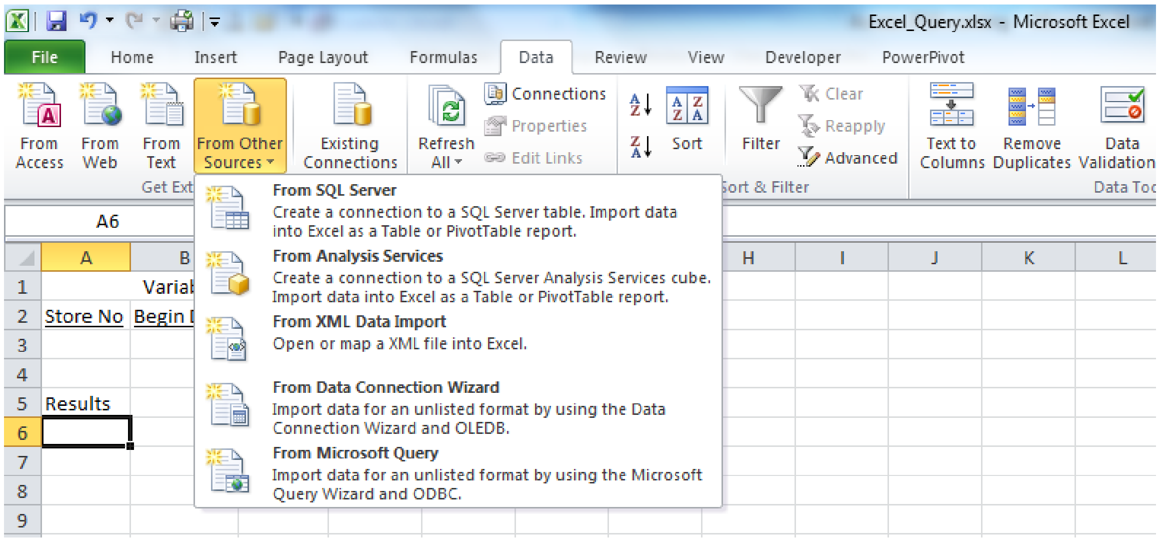
	A	B	C	D	E
1		Variables			
2	Store No	Begin Date	End Date		
3					
4					
5	Results				
6					
7					

Figure 15.1

3. *Save the spreadsheet as* **C:\ExcelCEO\Access 2010\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** *icon in the* **Get External Data** *group.*



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 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.

Figure 15.2

6. *Click on the* **From Microsoft Query** *option.*

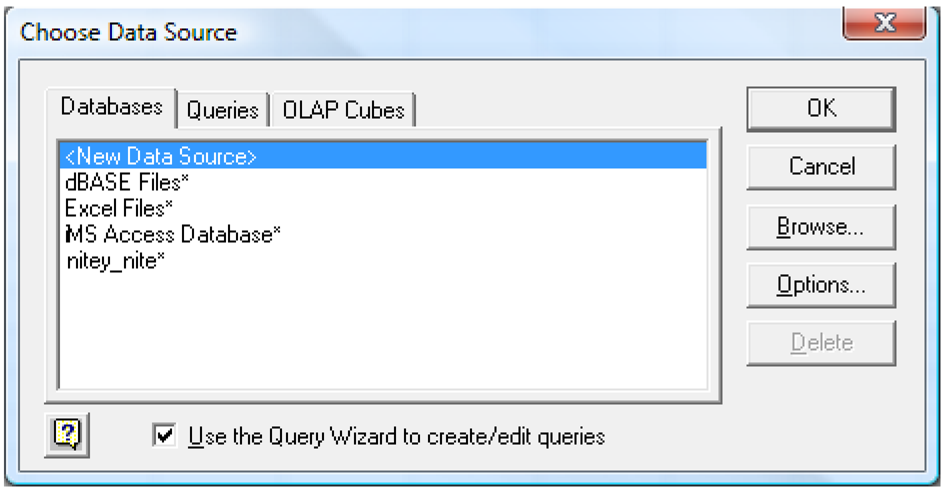


Figure 15.3

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 2010** *and choose the* **Static_2010.accdb** *database.*

8. *In the* **SQL Server Login** *dialog box, type* **testuser** *as the* **Login ID** *and* **clinesys1** *as the* **password** *and click* **OK**.
9. *In the* **Query Wizard**, *scroll down and click on the* **Sales_Journal_10** *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 >**.

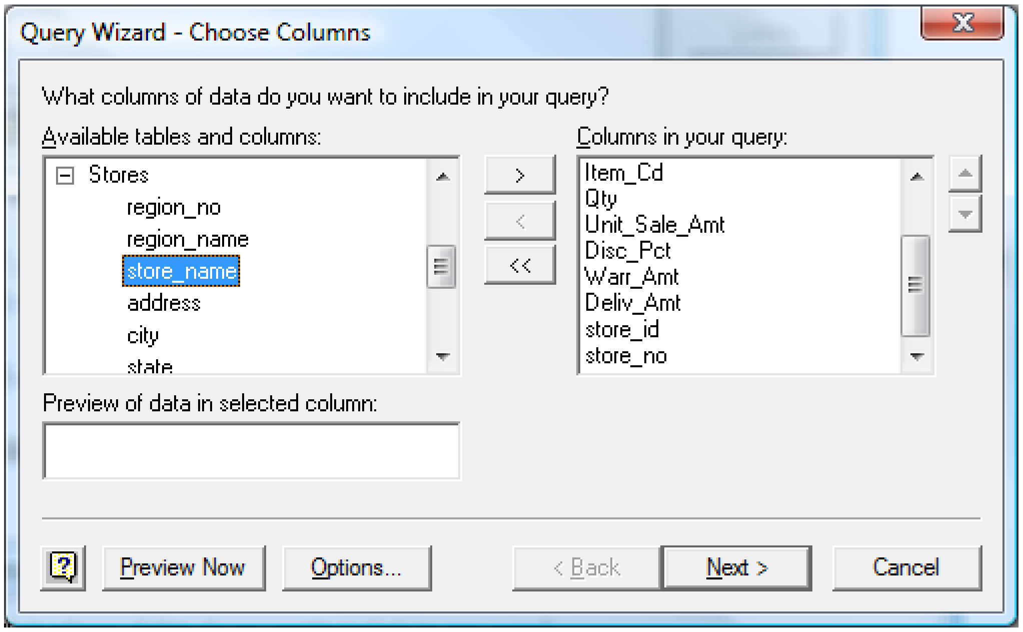
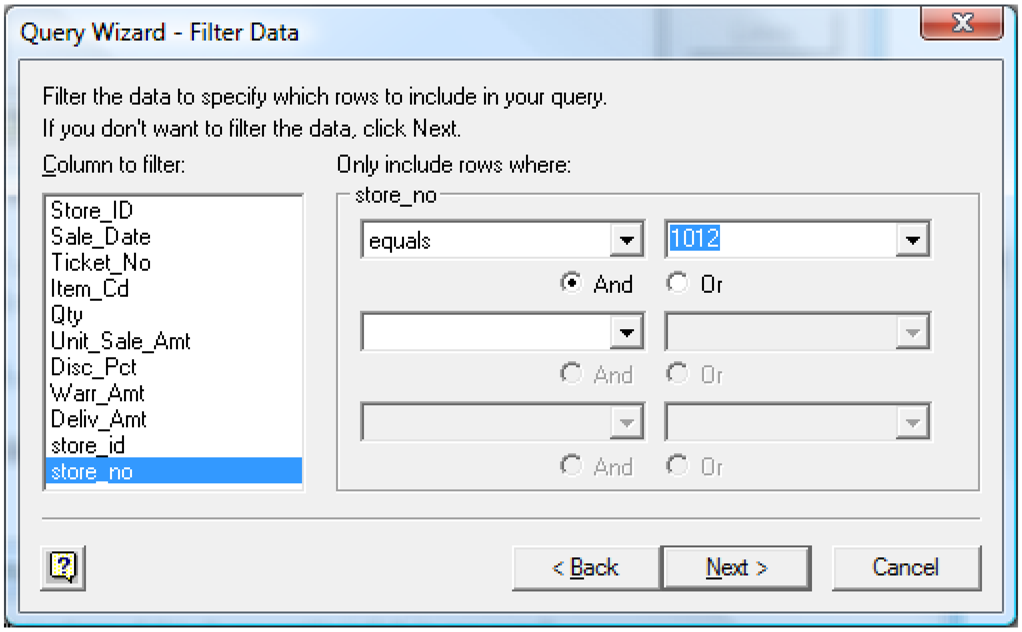


Figure 15.4

11. *In the next 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**).



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

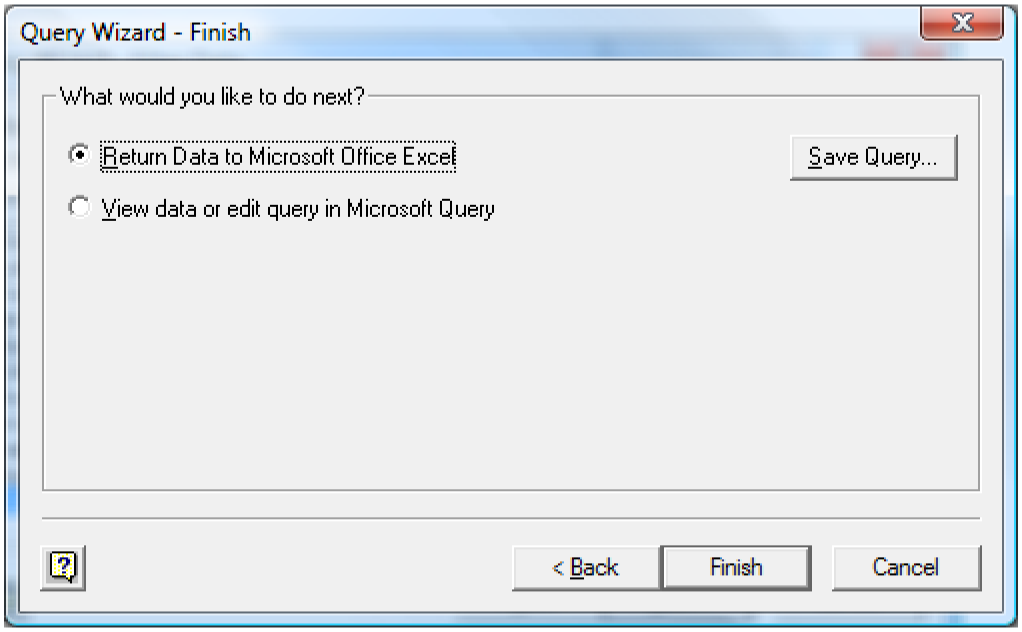
[✓ And] Or

And Or

And Or

Figure 15.4

13. *Click* **Next** > *two times to get to the last screen of the* **Query Wizard**.



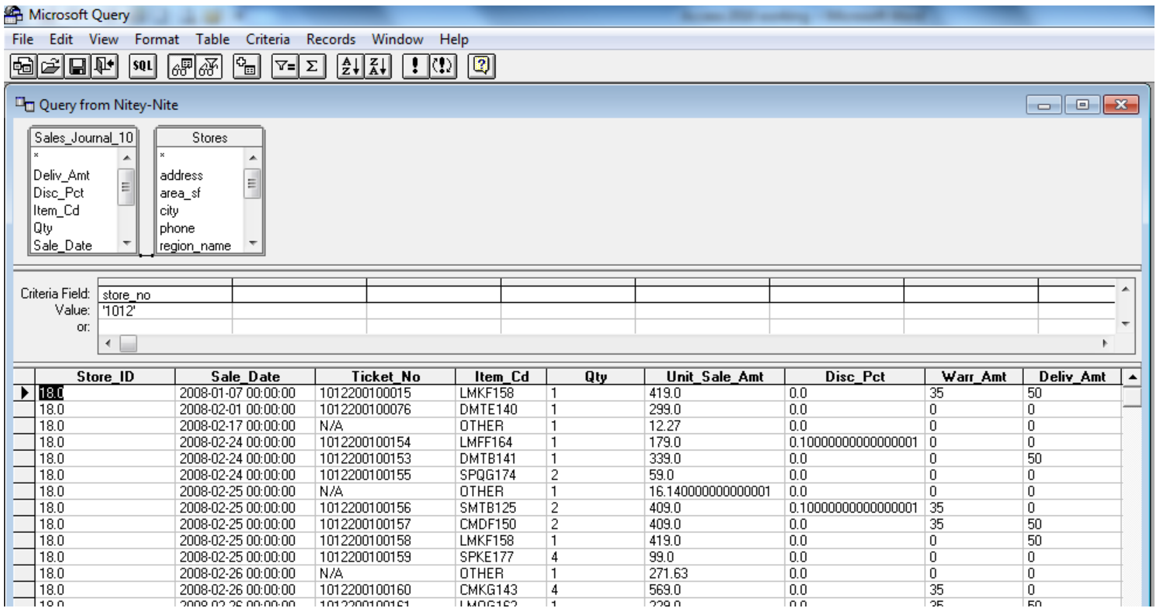
What would you like to do next?

[✓ Return Data to Microsoft Office Excel]

View data or edit query in Microsoft Query

Figure 15.5

14. *In the last screen of the* **Query Wizard**, *click on the* **View data or edit query in Microsoft Query** *radio button and click* **Finish**.



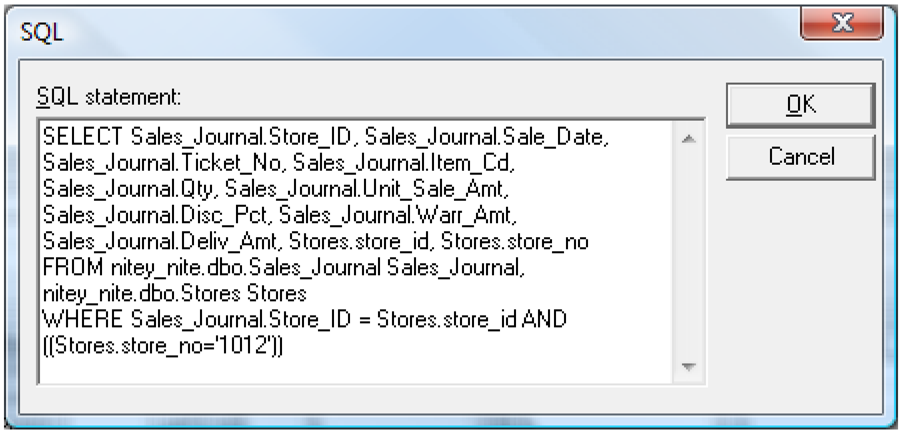
Criteria Field: store_no
Value: '1012'
or:

Store_ID	Sale_Date	Ticket_No	Item_Cd	Qty	Unit_Sale_Amt	Disc_Pct	Warr_Amt	Deliv_Amt
18.0	2008-01-07 00:00:00	1012200100015	LMKF158	1	419.0	0.0	35	50
18.0	2008-02-01 00:00:00	1012200100076	DMTE140	1	299.0	0.0	0	0
18.0	2008-02-17 00:00:00	N/A	OTHER	1	12.27	0.0	0	0
18.0	2008-02-24 00:00:00	1012200100154	LMFF164	1	179.0	0.10000000000000001	0	0
18.0	2008-02-24 00:00:00	1012200100153	DMTB141	1	339.0	0.0	0	50
18.0	2008-02-24 00:00:00	1012200100155	SPQG174	2	59.0	0.0	0	0
18.0	2008-02-25 00:00:00	N/A	OTHER	1	16.140000000000001	0.0	0	0
18.0	2008-02-25 00:00:00	1012200100156	SMTB125	2	409.0	0.10000000000000001	35	0
18.0	2008-02-25 00:00:00	1012200100157	CMDF150	2	409.0	0.0	35	50
18.0	2008-02-25 00:00:00	1012200100158	LMKF158	1	419.0	0.0	0	50
18.0	2008-02-25 00:00:00	1012200100159	SPKE177	4	99.0	0.0	0	0
18.0	2008-02-26 00:00:00	N/A	OTHER	1	271.63	0.0	0	0
18.0	2008-02-26 00:00:00	1012200100160	CMKG143	4	569.0	0.0	35	0

Figure 15.6

You should get a screen similar to the one in Figure 15.6. 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** *icon.*



```
SELECT Sales_Journal.Store_ID, Sales_Journal.Sale_Date,
Sales_Journal.Ticket_No, Sales_Journal.Item_Cd,
Sales_Journal.Qty, Sales_Journal.Unit_Sale_Amt,
Sales_Journal.Disc_Pct, Sales_Journal.Warr_Amt,
Sales_Journal.Deliv_Amt, Stores.store_id, Stores.store_no
FROM nitey_nite.dbo.Sales_Journal Sales_Journal,
nitey_nite.dbo.Stores Stores
WHERE Sales_Journal.Store_ID = Stores.store_id AND
((Stores.store_no='1012'))
```

Figure 15.7

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_2010.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* **1/1/2010 AND 1/31/2010**.

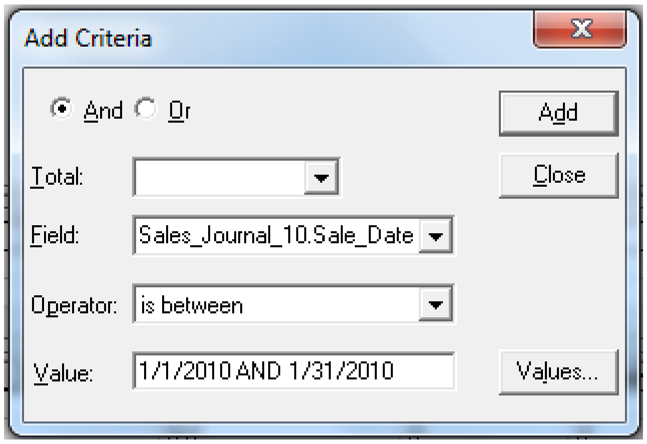
16. Click **OK** *on the SQL dialog box.*

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 ever use Microsoft Query, and need to create another join, just 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 1/1/2010 and 1/31/2010, as well as make a couple of changes in Microsoft Query.

1. *Maximize the interior window of the* **Design** *grid.*
2. **Microsoft Query** *should automatically establish a relationship on* **Store_ID**. *If it doesn't, you do it.*
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* **Criteria, Add Criteria…** *from the* **Microsoft Query Bar** *and input the criteria as shown in* **Figure 15.8**.



[✓ And] Or

Total:

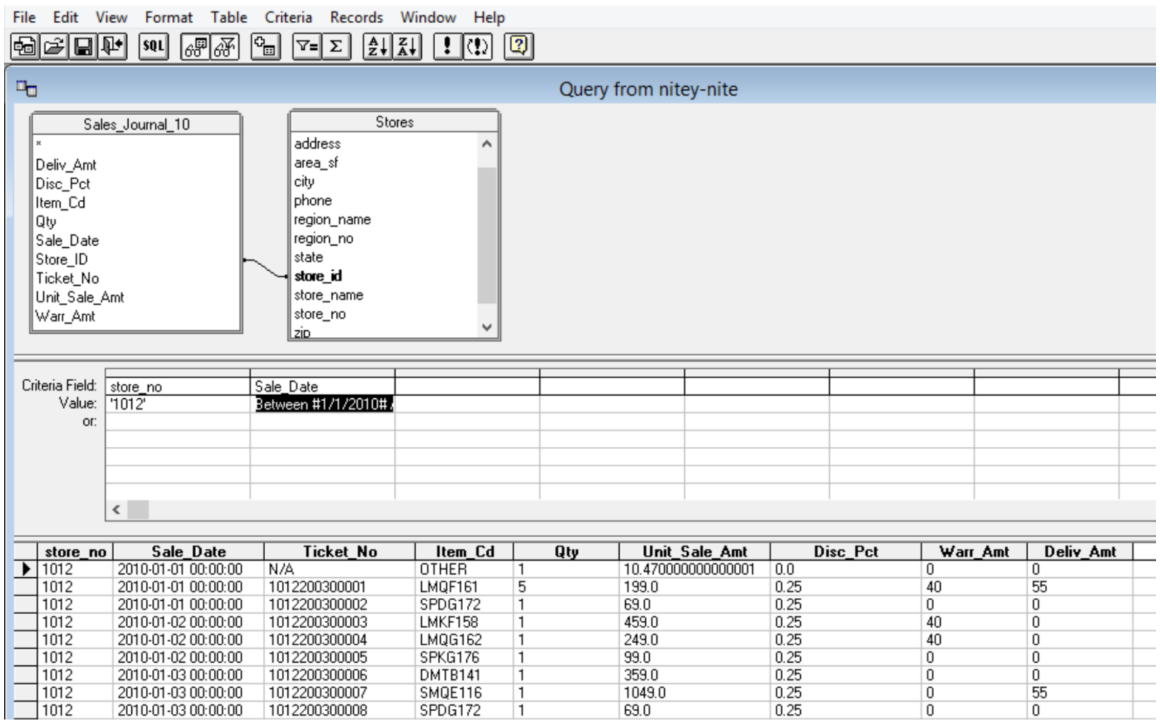
Field: Sales_Journal_10.Sale_Date

Operator: is between

Value: 1/1/2010 AND 1/31/2010

Figure 15.8

6. *Click the* **Add** *button then click the* **Close** *button.*



Criteria Field:	store_no	Sale_Date
Value:	'1012'	Between #1/1/2010#
or:		

store_no	Sale_Date	Ticket_No	Item_Cd	Qty	Unit_Sale_Amt	Disc_Pct	Warr_Amt	Deliv_Amt
1012	2010-01-01 00:00:00	N/A	OTHER	1	10.470000000000001	0.0	0	0
1012	2010-01-01 00:00:00	1012200300001	LMQF161	5	199.0	0.25	40	55
1012	2010-01-01 00:00:00	1012200300002	SPDG172	1	69.0	0.25	0	0
1012	2010-01-02 00:00:00	1012200300003	LMKF158	1	459.0	0.25	40	0
1012	2010-01-02 00:00:00	1012200300004	LMQG162	1	249.0	0.25	40	0
1012	2010-01-02 00:00:00	1012200300005	SPKG176	1	99.0	0.25	0	0
1012	2010-01-03 00:00:00	1012200300006	DMTB141	1	359.0	0.25	0	0
1012	2010-01-03 00:00:00	1012200300007	SMQE116	1	1049.0	0.25	0	55
1012	2010-01-03 00:00:00	1012200300008	SPDG172	1	69.0	0.25	0	0

Figure 15.9

The Microsoft Query design view screen should look like the Figure 15.9.

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…**

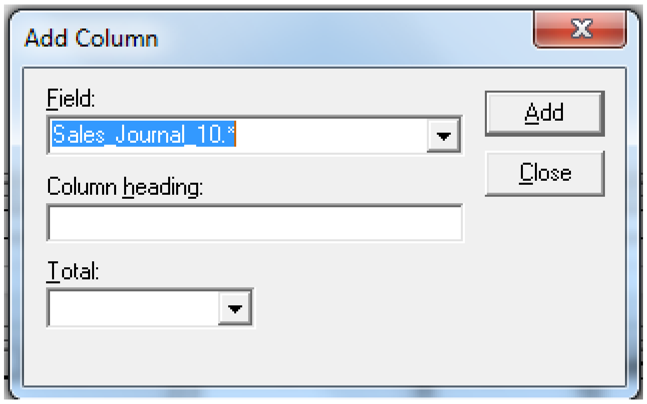


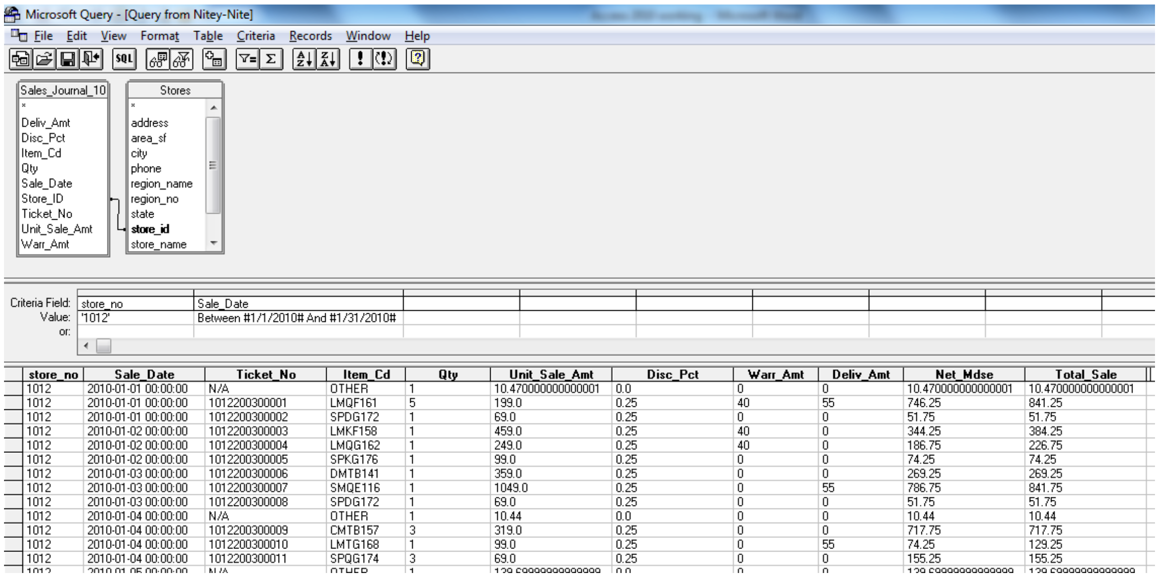
Figure 15.10

> ***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_10.Unit_Sale_Amt*Sales_Journal_10.Qty*(1-Sales_Journal_10.Disc_Pct)
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_10.Unit_Sale_Amt*Sales_Journal_10.Qty*(1-Sales_Journal_10.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.*



Criteria Field:	store_no	Sale_Date
Value:	'1012'	Between #1/1/2010# And #1/31/2010#
or:		

store_no	Sale_Date	Ticket_No	Item_Cd	Qty	Unit_Sale_Amt	Disc_Pct	Warr_Amt	Deliv_Amt	Net_Mdse	Total_Sale
1012	2010-01-01 00:00:00	N/A	OTHER	1	10.470000000000001	0.0	0	0	10.470000000000001	10.470000000000001
1012	2010-01-01 00:00:00	1012200300001	LMQF161	5	199.0	0.25	40	55	746.25	841.25
1012	2010-01-01 00:00:00	1012200300002	SPDG172	1	69.0	0.25	0	0	51.75	51.75
1012	2010-01-02 00:00:00	1012200300003	LMKF158	1	459.0	0.25	40	0	344.25	384.25
1012	2010-01-02 00:00:00	1012200300004	LMQG162	1	249.0	0.25	40	0	186.75	226.75
1012	2010-01-02 00:00:00	1012200300005	SPKG176	1	99.0	0.25	0	0	74.25	74.25
1012	2010-01-03 00:00:00	1012200300006	DMTB141	1	359.0	0.25	0	0	269.25	269.25
1012	2010-01-03 00:00:00	1012200300007	SMQE116	1	1049.0	0.25	0	55	786.75	841.75
1012	2010-01-03 00:00:00	1012200300008	SPDG172	1	69.0	0.25	0	0	51.75	51.75
1012	2010-01-04 00:00:00	N/A	OTHER	1	10.44	0.0	0	0	10.44	10.44
1012	2010-01-04 00:00:00	1012200300009	CMTB157	3	319.0	0.25	0	0	717.75	717.75
1012	2010-01-04 00:00:00	1012200300010	LMTG168	1	99.0	0.25	0	55	74.25	129.25
1012	2010-01-04 00:00:00	1012200300011	SPQG174	3	69.0	0.25	0	0	155.25	155.25

Figure 15.11

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 red* **X** *at the top-right of the* **Microsoft Query** *window.*

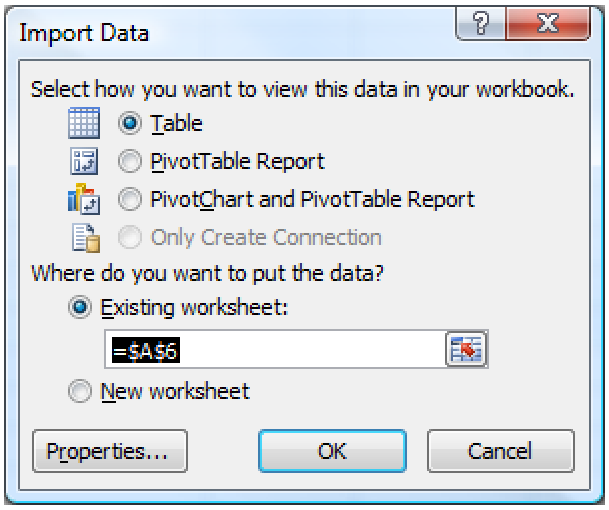
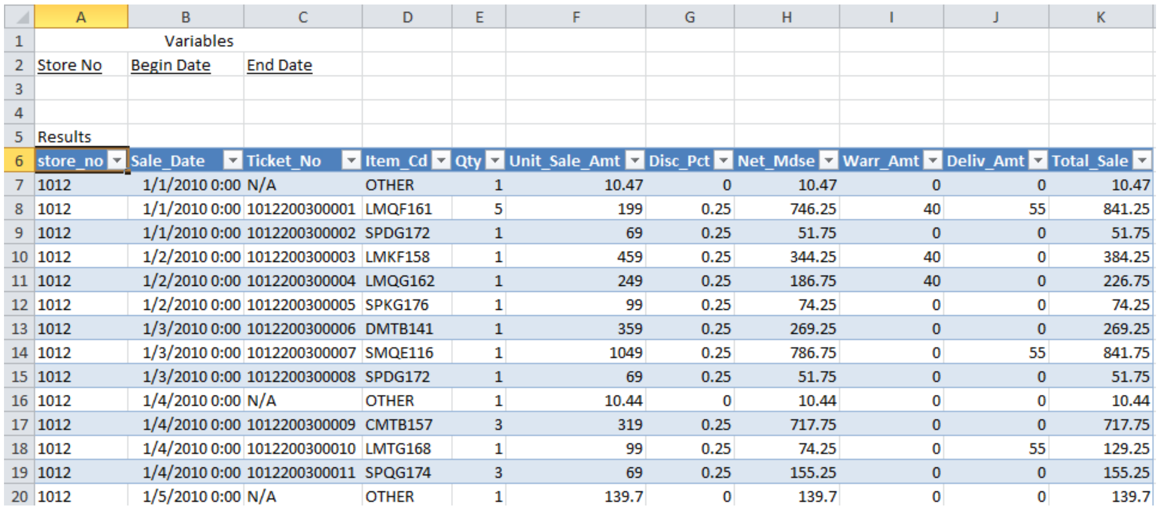

Import Data
Select how you want to view this data in your workbook.
[✓ Table]
PivotTable Report
PivotChart and PivotTable Report
Only Create Connection
Where do you want to put the data?
[✓ Existing worksheet:]
=$A$6
New worksheet
Properties...
OK
Cancel

Figure 15.12

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** *of the* **existing worksheet** *and click* **OK***.*



	A	B	C	D	E	F	G	H	I	J	K
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_Mdse	Warr_Amt	Deliv_Amt	Total_Sale
7	1012	1/1/2010 0:00	N/A	OTHER	1	10.47	0	10.47	0	0	10.47
8	1012	1/1/2010 0:00	1012200300001	LMQF161	5	199	0.25	746.25	40	55	841.25
9	1012	1/1/2010 0:00	1012200300002	SPDG172	1	69	0.25	51.75	0	0	51.75
10	1012	1/2/2010 0:00	1012200300003	LMKF158	1	459	0.25	344.25	40	0	384.25
11	1012	1/2/2010 0:00	1012200300004	LMQG162	1	249	0.25	186.75	40	0	226.75
12	1012	1/2/2010 0:00	1012200300005	SPKG176	1	99	0.25	74.25	0	0	74.25
13	1012	1/3/2010 0:00	1012200300006	DMTB141	1	359	0.25	269.25	0	0	269.25
14	1012	1/3/2010 0:00	1012200300007	SMQE116	1	1049	0.25	786.75	0	55	841.75
15	1012	1/3/2010 0:00	1012200300008	SPDG172	1	69	0.25	51.75	0	0	51.75
16	1012	1/4/2010 0:00	N/A	OTHER	1	10.44	0	10.44	0	0	10.44
17	1012	1/4/2010 0:00	1012200300009	CMTB157	3	319	0.25	717.75	0	0	717.75
18	1012	1/4/2010 0:00	1012200300010	LMTG168	1	99	0.25	74.25	0	55	129.25
19	1012	1/4/2010 0:00	1012200300011	SPQG174	3	69	0.25	155.25	0	0	155.25
20	1012	1/5/2010 0:00	N/A	OTHER	1	139.7	0	139.7	0	0	139.7

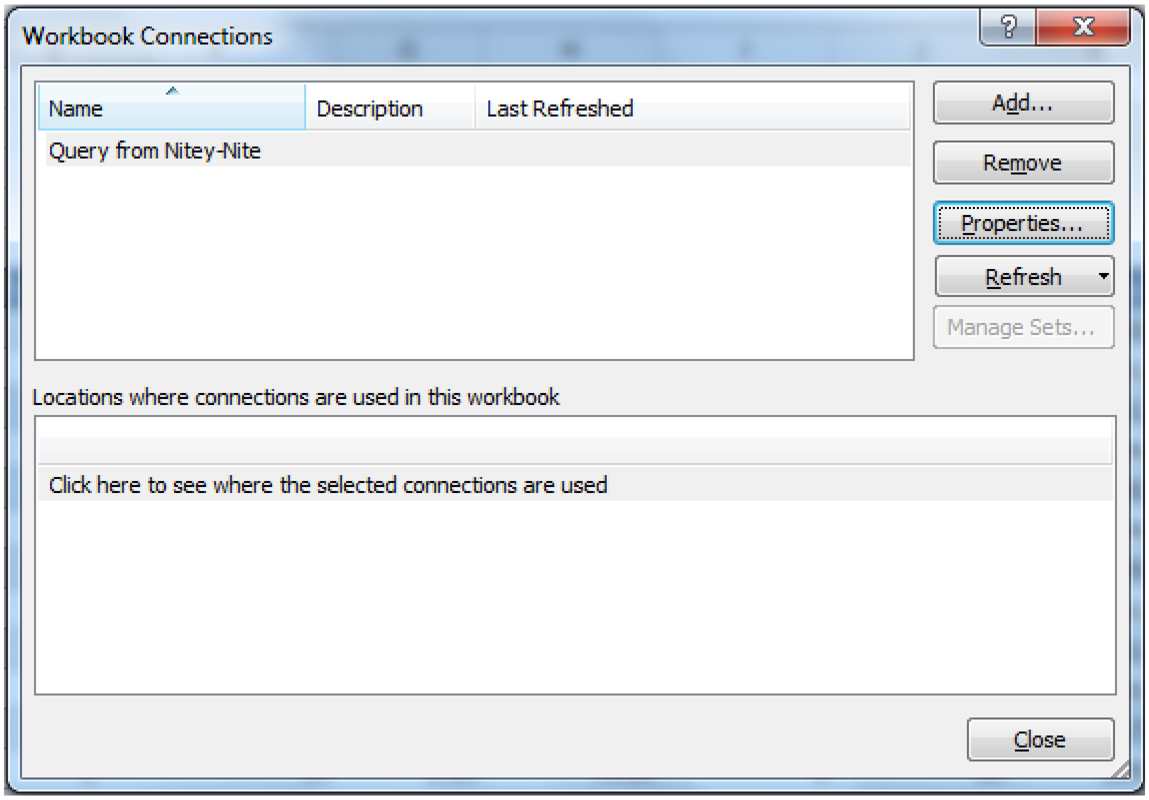
Figure 15.13

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** *icon in the* **Data** *tab.*



Name	Description	Last Refreshed
Query from Nitey-Nite		

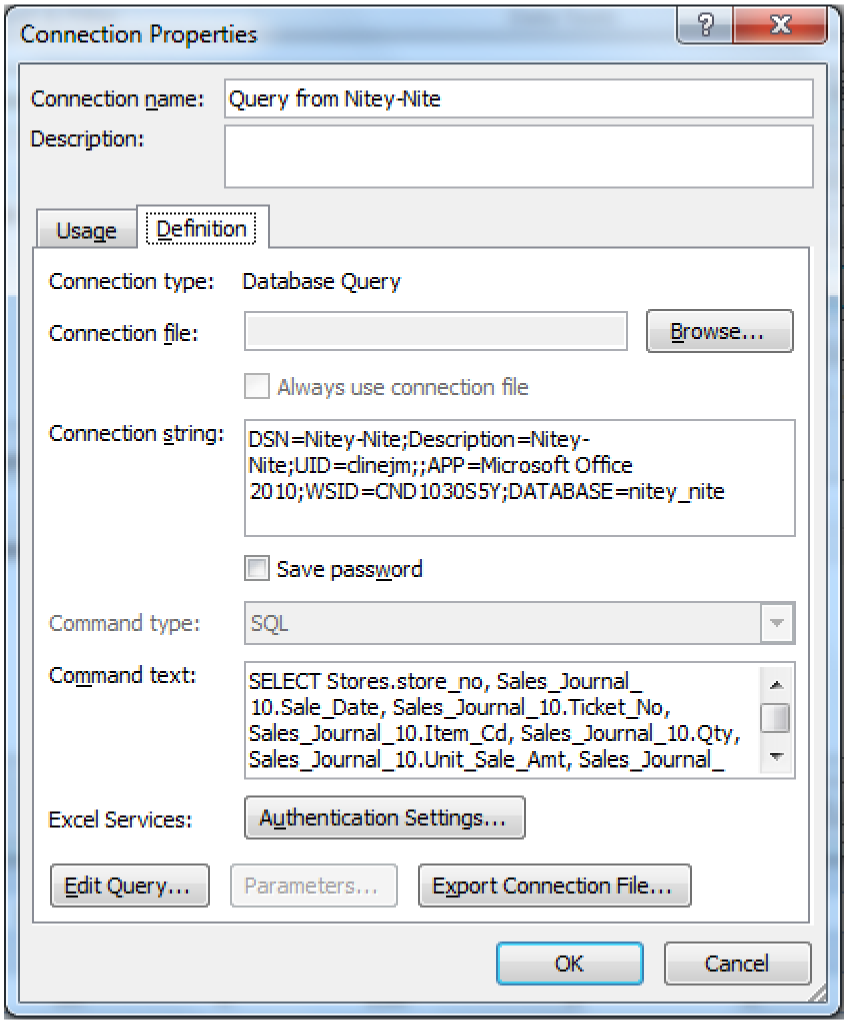
Locations where connections are used in this workbook

Click here to see where the selected connections are used

Figure 15.14

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;UID=clinejm;;APP=Microsoft Office 2010;WSID=CND1030S5Y;DATABASE=nitey_nite

Save password

Command type: SQL

Command text:

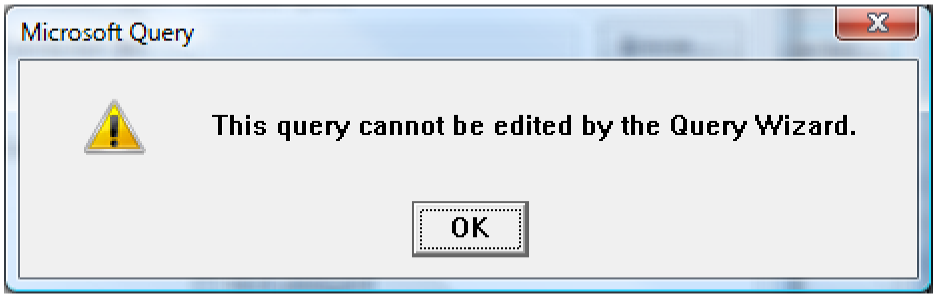
```
SELECT Stores.store_no, Sales_Journal_
10.Sale_Date, Sales_Journal_10.Ticket_No,
Sales_Journal_10.Item_Cd, Sales_Journal_10.Qty,
Sales_Journal_10.Unit_Sale_Amt, Sales_Journal_
```

Excel Services: Authentication Settings...

Figure 15.15

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.*


Microsoft Query

This query cannot be edited by the Query Wizard.

Figure 15.16

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.*

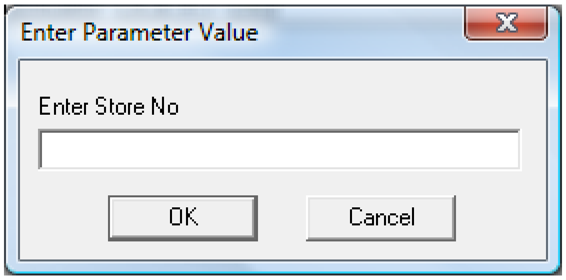


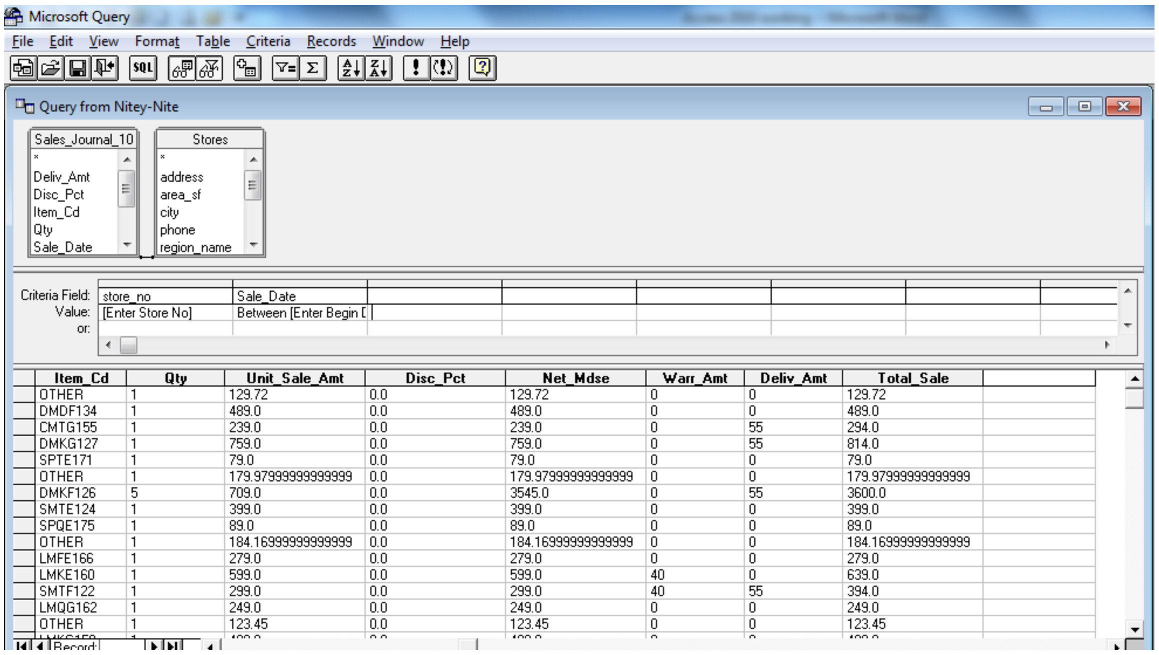
Figure 12.17

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 #1/1/2010# And #1/31/2010#** *with* **Between [Enter Begin Date] And [Enter End Date].**
8. *Tab out of the* **Sale_Date** *criteria field and input* **4/1/2010** *and* **6/30/2010** *in the* **Enter Begin Date** *and* **Enter End Date** *parameters when prompted.*

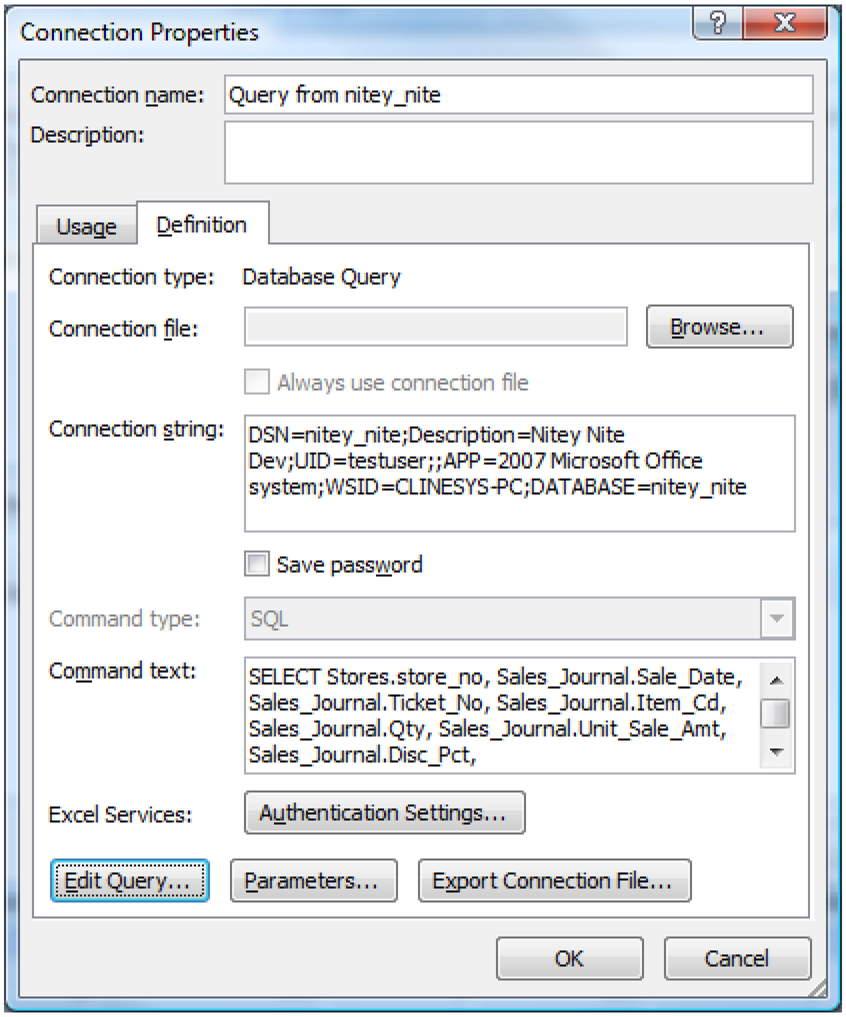


Criteria Field:	store_no	Sale_Date
Value:	[Enter Store No]	Between [Enter Begin [
or:		

Item_Cd	Qty	Unit_Sale_Amt	Disc_Pct	Net_Mdse	Warr_Amt	Deliv_Amt	Total_Sale
OTHER	1	129.72	0.0	129.72	0	0	129.72
DMDF134	1	489.0	0.0	489.0	0	0	489.0
CMTG155	1	239.0	0.0	239.0	0	55	294.0
DMKG127	1	759.0	0.0	759.0	0	55	814.0
SPTE171	1	79.0	0.0	79.0	0	0	79.0
OTHER	1	179.97999999999999	0.0	179.97999999999999	0	0	179.97999999999999
DMKF126	5	709.0	0.0	3545.0	0	55	3600.0
SMTE124	1	399.0	0.0	399.0	0	0	399.0
SPQE175	1	89.0	0.0	89.0	0	0	89.0
OTHER	1	184.16999999999999	0.0	184.16999999999999	0	0	184.16999999999999
LMFE166	1	279.0	0.0	279.0	0	0	279.0
LMKE160	1	599.0	0.0	599.0	40	0	639.0
SMTF122	1	299.0	0.0	299.0	40	55	394.0
LMQG162	1	249.0	0.0	249.0	0	0	249.0
OTHER	1	123.45	0.0	123.45	0	0	123.45

Figure 15.18

9. *Exit* **Microsoft Query**.



Connection name: Query from nitey_nite

Description:

Connection type: Database Query

Connection file:

Connection string: DSN=nitey_nite;Description=Nitey Nite Dev;UID=testuser;;APP=2007 Microsoft Office system;WSID=CLINESYS-PC;DATABASE=nitey_nite

Command type: SQL

Command text:

```
SELECT Stores.store_no, Sales_Journal.Sale_Date,
Sales_Journal.Ticket_No, Sales_Journal.Item_Cd,
Sales_Journal.Qty, Sales_Journal.Unit_Sale_Amt,
Sales_Journal.Disc_Pct,
```

Excel Services:

Figure 15.19

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* **1/1/2009**.
13. *In* **Cell C3**, *type* **1/31/2009**.
14. *Click on the* **Data** *tab,* **Connections** *icon, then on the* **Properties…** *button and the* **Definition** *tab, and lastly on the* **Parameters…** *button.*

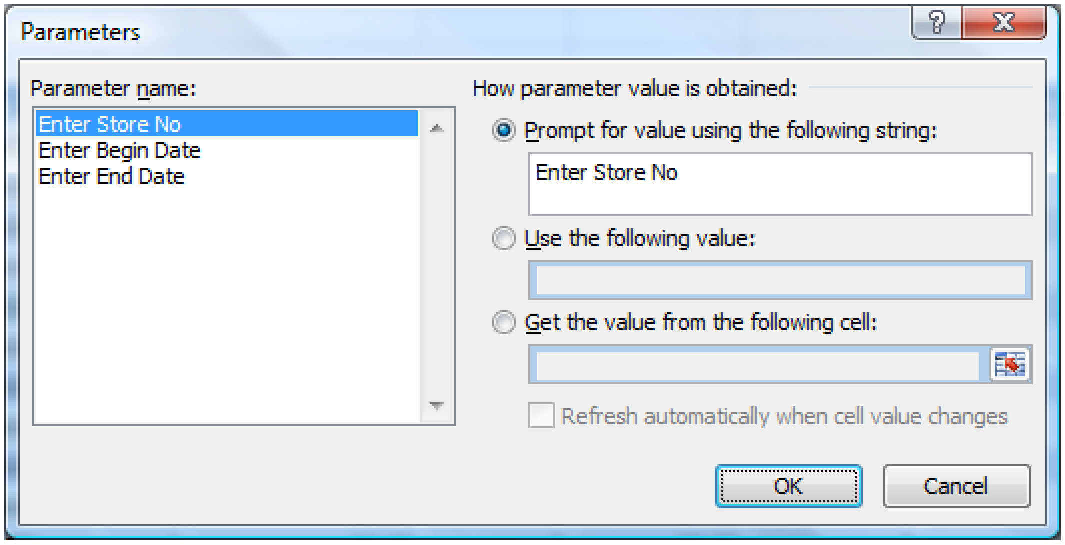

Parameters
Parameter name:
Enter Store No
Enter Begin Date
Enter End Date
How parameter value is obtained:
Prompt for value using the following string:
Enter Store No
Use the following value:
Get the value from the following cell:
Refresh automatically when cell value changes
OK
Cancel

Figure 15.20

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**.

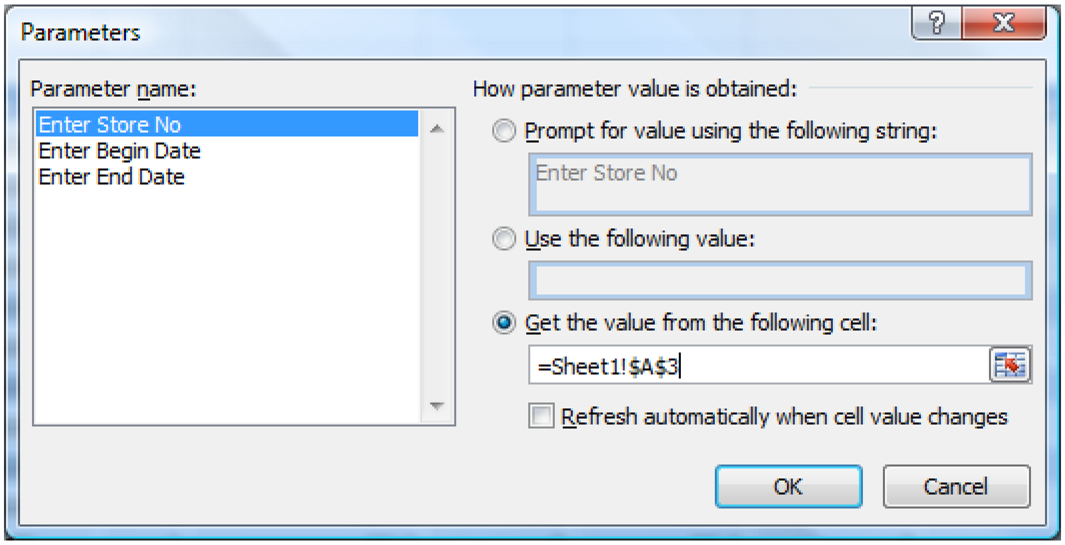

Parameters
Parameter name:
Enter Store No
Enter Begin Date
Enter End Date
How parameter value is obtained:
Prompt for value using the following string:
Enter Store No
Use the following value:
Get the value from the following cell:
=Sheet1!$A$3
Refresh automatically when cell value changes
OK
Cancel

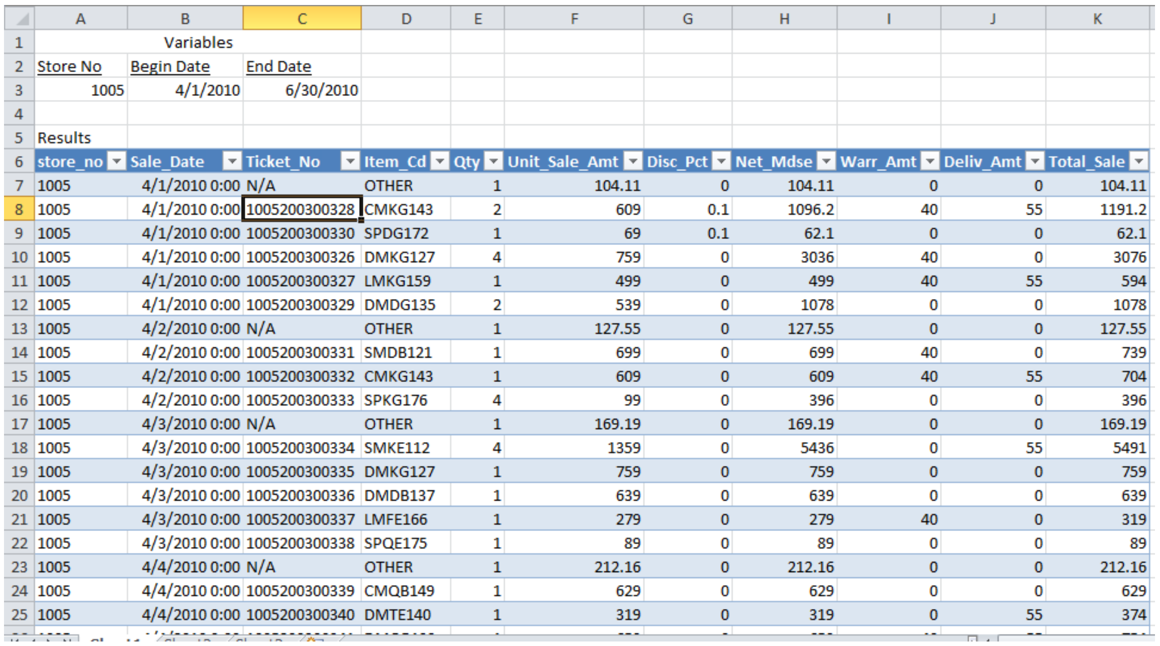
Figure 15.21

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 (testuser) 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* **4/1/2010** *and the* **End Date** *to* **6/30/2010***.*
21. *Right-click anywhere in the table and choose* **Refresh** *icon.*



	A	B	C	D	E	F	G	H	I	J	K
1		Variables									
2	Store No	Begin Date	End Date								
3	1005	4/1/2010	6/30/2010								
4											
5	Results										
6	store_no	Sale_Date	Ticket_No	Item_Cd	Qty	Unit_Sale_Amt	Disc_Pct	Net_Mdse	Warr_Amt	Deliv_Amt	Total_Sale
7	1005	4/1/2010 0:00	N/A	OTHER	1	104.11	0	104.11	0	0	104.11
8	1005	4/1/2010 0:00	1005200300328	CMKG143	2	609	0.1	1096.2	40	55	1191.2
9	1005	4/1/2010 0:00	1005200300330	SPDG172	1	69	0.1	62.1	0	0	62.1
10	1005	4/1/2010 0:00	1005200300326	DMKG127	4	759	0	3036	40	0	3076
11	1005	4/1/2010 0:00	1005200300327	LMKG159	1	499	0	499	40	55	594
12	1005	4/1/2010 0:00	1005200300329	DMDG135	2	539	0	1078	0	0	1078
13	1005	4/2/2010 0:00	N/A	OTHER	1	127.55	0	127.55	0	0	127.55
14	1005	4/2/2010 0:00	1005200300331	SMDB121	1	699	0	699	40	0	739
15	1005	4/2/2010 0:00	1005200300332	CMKG143	1	609	0	609	40	55	704
16	1005	4/2/2010 0:00	1005200300333	SPKG176	4	99	0	396	0	0	396
17	1005	4/3/2010 0:00	N/A	OTHER	1	169.19	0	169.19	0	0	169.19
18	1005	4/3/2010 0:00	1005200300334	SMKE112	4	1359	0	5436	0	55	5491
19	1005	4/3/2010 0:00	1005200300335	DMKG127	1	759	0	759	0	0	759
20	1005	4/3/2010 0:00	1005200300336	DMDB137	1	639	0	639	0	0	639
21	1005	4/3/2010 0:00	1005200300337	LMFE166	1	279	0	279	40	0	319
22	1005	4/3/2010 0:00	1005200300338	SPQE175	1	89	0	89	0	0	89
23	1005	4/4/2010 0:00	N/A	OTHER	1	212.16	0	212.16	0	0	212.16
24	1005	4/4/2010 0:00	1005200300339	CMQB149	1	629	0	629	0	0	629
25	1005	4/4/2010 0:00	1005200300340	DMTE140	1	319	0	319	0	55	374

Figure 15.22

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* **Controls** *group and click on the first image* (**Button** (**Form Control**)).
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**.

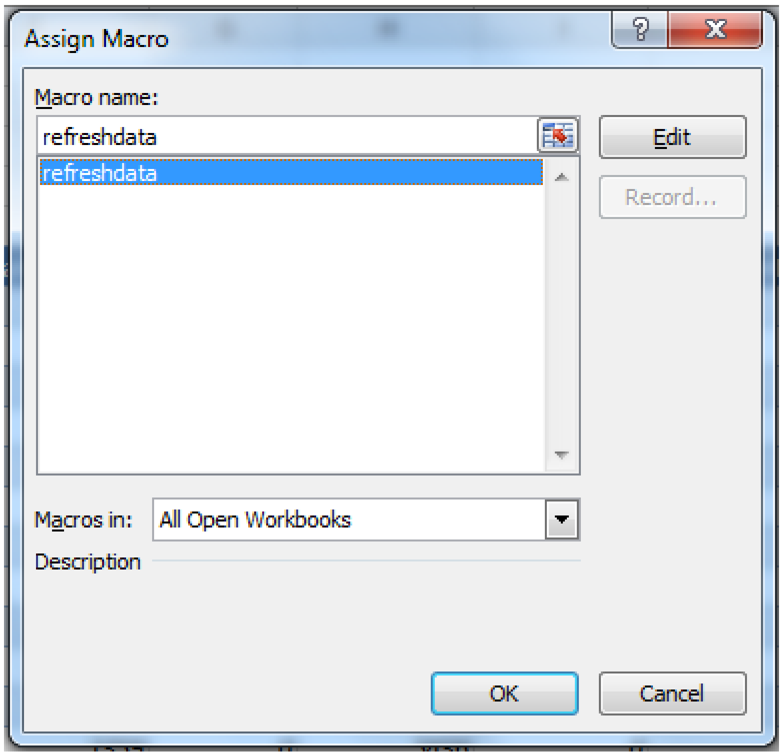
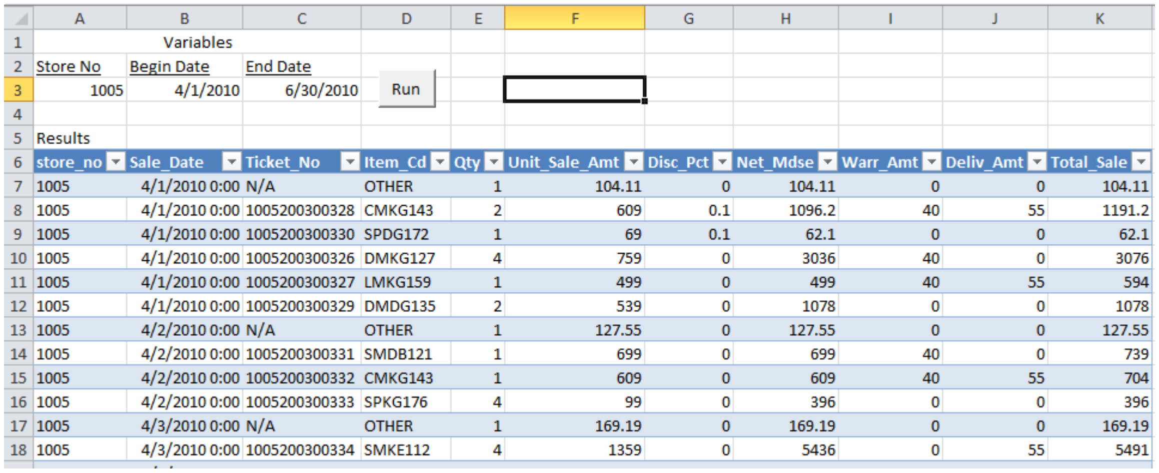


Figure 15.23

31. *While the button is in* **Edit** *mode, replace* **Button1** *with* **Run**, *and click outside of the* **Command Button**.



	A	B	C	D	E	F	G	H	I	J	K
1		Variables									
2	Store No	Begin Date	End Date								
3	1005	4/1/2010	6/30/2010	Run							
4											
5	Results										
6	store_no	Sale_Date	Ticket_No	Item_Cd	Qty	Unit_Sale_Amt	Disc_Pct	Net_Mdse	Warr_Amt	Deliv_Amt	Total_Sale
7	1005	4/1/2010 0:00	N/A	OTHER	1	104.11	0	104.11	0	0	104.11
8	1005	4/1/2010 0:00	1005200300328	CMKG143	2	609	0.1	1096.2	40	55	1191.2
9	1005	4/1/2010 0:00	1005200300330	SPDG172	1	69	0.1	62.1	0	0	62.1
10	1005	4/1/2010 0:00	1005200300326	DMKG127	4	759	0	3036	40	0	3076
11	1005	4/1/2010 0:00	1005200300327	LMKG159	1	499	0	499	40	55	594
12	1005	4/1/2010 0:00	1005200300329	DMDG135	2	539	0	1078	0	0	1078
13	1005	4/2/2010 0:00	N/A	OTHER	1	127.55	0	127.55	0	0	127.55
14	1005	4/2/2010 0:00	1005200300331	SMDB121	1	699	0	699	40	0	739
15	1005	4/2/2010 0:00	1005200300332	CMKG143	1	609	0	609	40	55	704
16	1005	4/2/2010 0:00	1005200300333	SPKG176	4	99	0	396	0	0	396
17	1005	4/3/2010 0:00	N/A	OTHER	1	169.19	0	169.19	0	0	169.19
18	1005	4/3/2010 0:00	1005200300334	SMKE112	4	1359	0	5436	0	55	5491

Figure 15.24

You now have a fully functional data query tool in Excel! The data behind this record set is stored on a server somewhere on the East Coast, 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 the file (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.

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. Does 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 2010** *called* **Mattress_Mix.accdb**.
2. *In the database, import the* **Item**, **Sales_Journal**, *and* **Stores** *tables from the* **Nitey_Nite_2010 Access** *database.*
3. *Change the* **navigation pane** *to view the* **Access objects** *by* **object type** *(if necessary).*

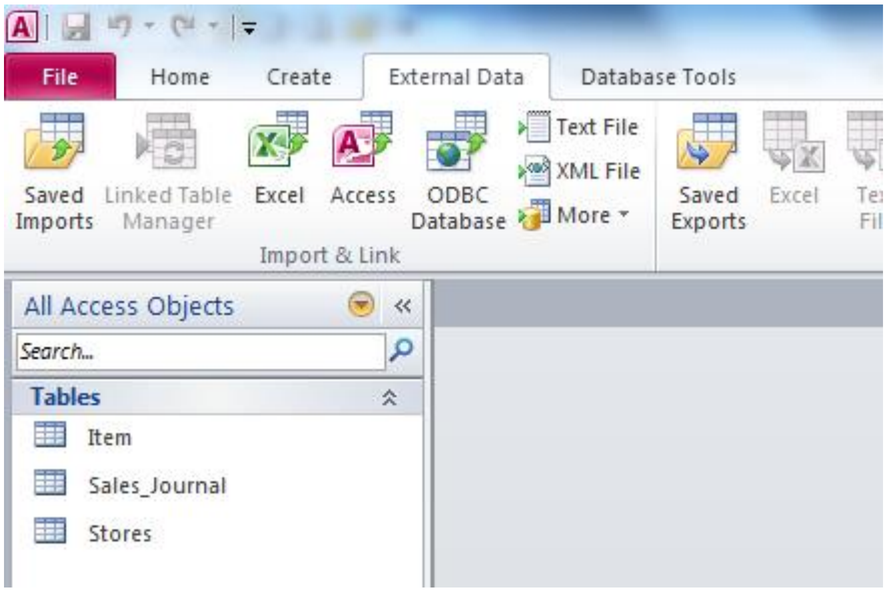
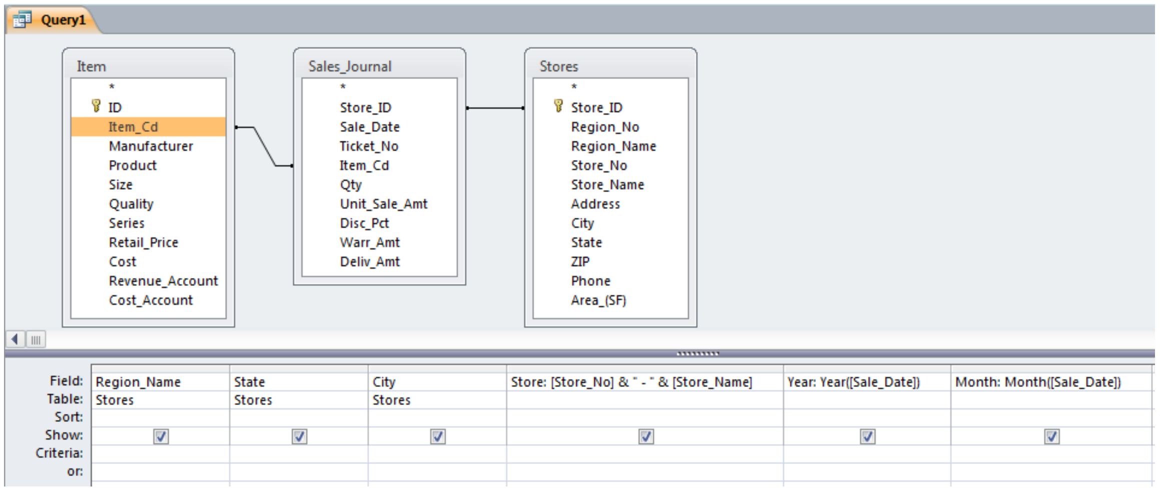


Figure 15.25

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.

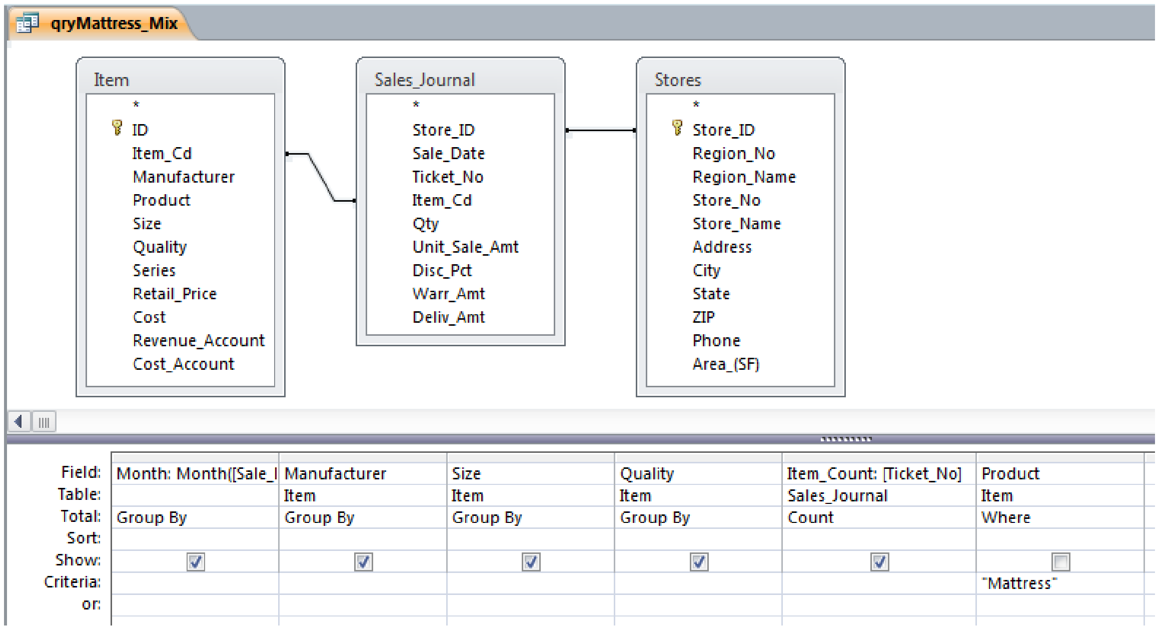
4. *In a new* **Access** *query, bring in the* **Item**, **Sales_Journal**, *and* **Stores** *tables.*
5. *Join the* **Item** *and* **Sales_Journal** *tables on* **Item_Cd** *and make sure the* **Sales_Journal** *and* **Stores** *tables are still joined on the* **Store_ID**.
6. *Bring the* **Region_Name**, **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**.



Field:	Region_Name	State	City	Store: [Store_No] & " - " & [Store_Name]	Year: Year([Sale_Date])	Month: Month([Sale_Date])
Table:	Stores	Stores	Stores			
Sort:						
Show:	✓	✓	✓	✓	✓	✓
Criteria:						
or:						

Figure 15.26

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**.



qryMattress_Mix

Item	Sales_Journal	Stores
*	*	*
ID	Store_ID	Store_ID
Item_Cd	Sale_Date	Region_No
Manufacturer	Ticket_No	Region_Name
Product	Item_Cd	Store_No
Size	Qty	Store_Name
Quality	Unit_Sale_Amt	Address
Series	Disc_Pct	City
Retail_Price	Warr_Amt	State
Cost	Deliv_Amt	ZIP
Revenue_Account		Phone
Cost_Account		Area_(SF)

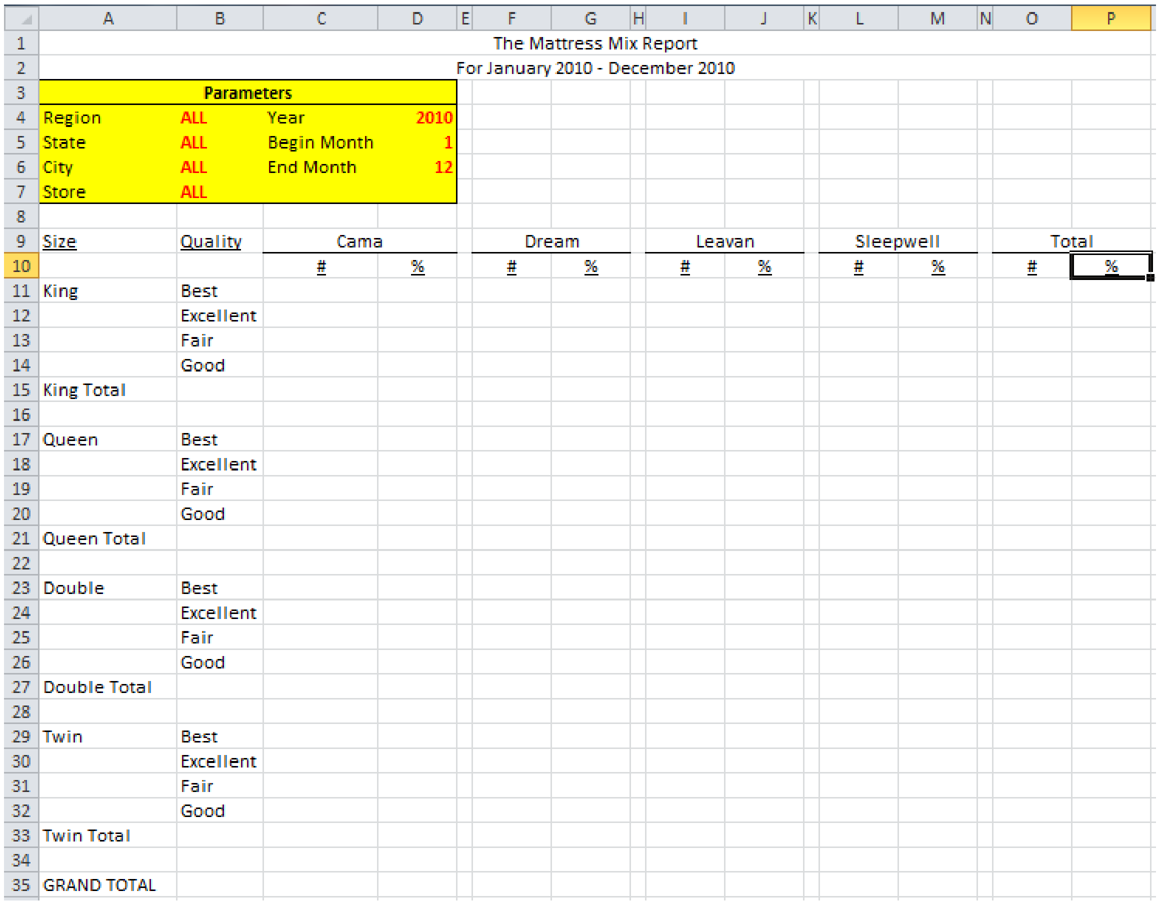
Field:	Month: Month([Sale_I	Manufacturer	Size	Quality	Item_Count: [Ticket_No]	Product
Table:		Item	Item	Item	Sales_Journal	Item
Total:	Group By	Group By	Group By	Group By	Count	Where
Sort:						
Show:	✓	✓	✓	✓	✓	☐
Criteria:						"Mattress"
or:						

Figure 15.27

15. *Test the query to make sure it is working (you should have* **42,662** *rows of data).*
16. *Close the query and close* **Access***.*

Now that you have your data source, you'll create the shell of the Excel file.

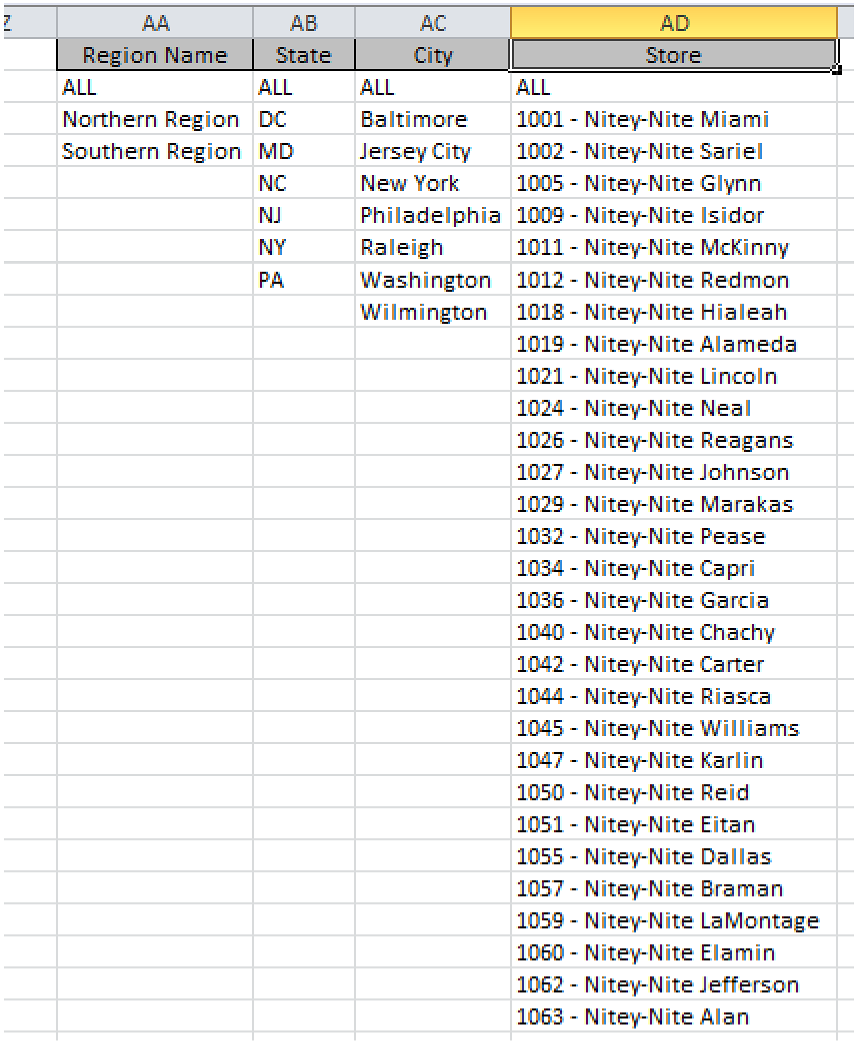
1. *Open a new* **Excel** *file and save it as* **C:\ExcelCEO\Access 2010\Mattress_Mix.xlsx***.*
2. *Rename* **Sheet1** *as* **Report***,* **Sheet2** *as* **Data** *and delete* **Sheet3***.*
3. *On the* **Report** *tab, create a spreadsheet as in* **Figure 15.26***.*



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1						The Mattress Mix Report										
2						For January 2010 - December 2010										
3	Parameters															
4	Region	ALL	Year	2010												
5	State	ALL	Begin Month	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															

Figure 15.28

4. *On the* **Report** *tab, go to* **Cell AA1** *and input the values according to* **Figure 15.27**.

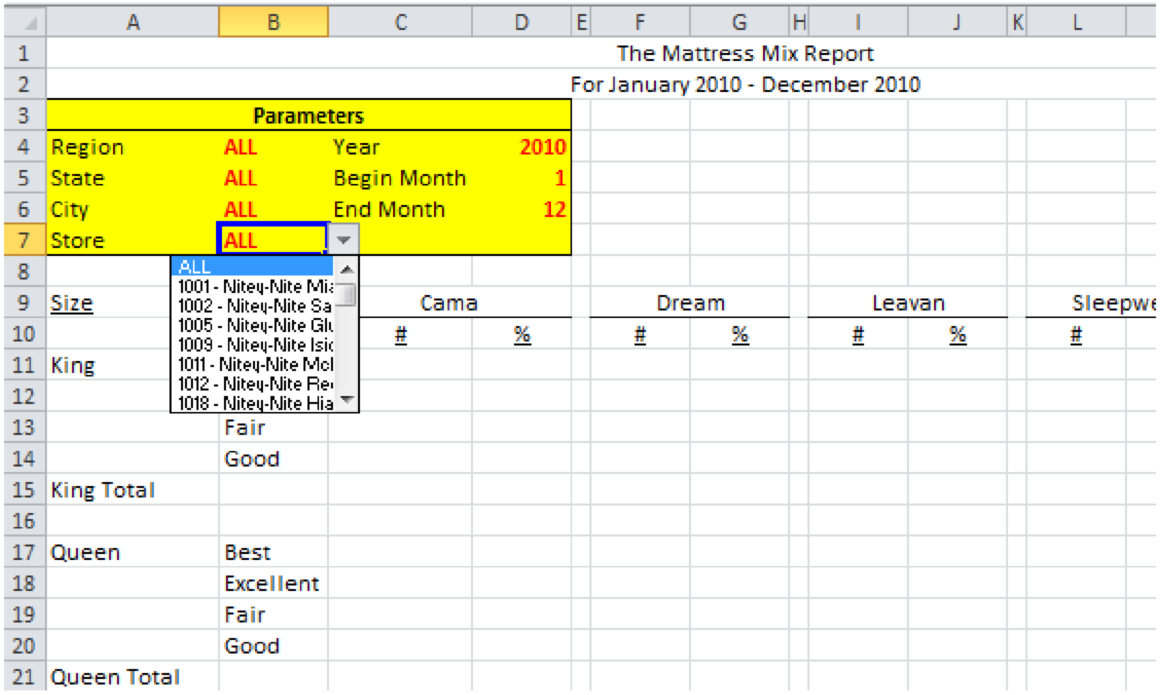


Region Name	State	City	Store
ALL	ALL	ALL	ALL
Northern Region	DC	Baltimore	1001 - Nitey-Nite Miami
Southern Region	MD	Jersey City	1002 - Nitey-Nite Sariel
	NC	New York	1005 - Nitey-Nite Glynn
	NJ	Philadelphia	1009 - Nitey-Nite Isidor
	NY	Raleigh	1011 - Nitey-Nite McKinny
	PA	Washington	1012 - Nitey-Nite 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 15.29

These columns contain the values we will use in a list for Cells B4 through B7. To create these ranges, all I did was to create a query in Access to group each of the Region, State and City fields. For the Store No. and Name, I created a concatenated field in Access and copied it to the indicated column. It's probably easier to do in Access than to type in all the values, especially for the Store field. 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	K	L
1							The Mattress Mix Report					
2							For January 2010 - December 2010					
3		Parameters										
4	Region	ALL	Year	2010								
5	State	ALL	Begin Month	1								
6	City	ALL	End Month	12								
7	Store	ALL										
8												
9	Size		Cama			Dream			Leavan			Sleepwe
10			#	%		#	%		#	%		#
11	King											
12												
13		Fair										
14		Good										
15	King Total											
16												
17	Queen	Best										
18		Excellent										
19		Fair										
20		Good										
21	Queen Total											

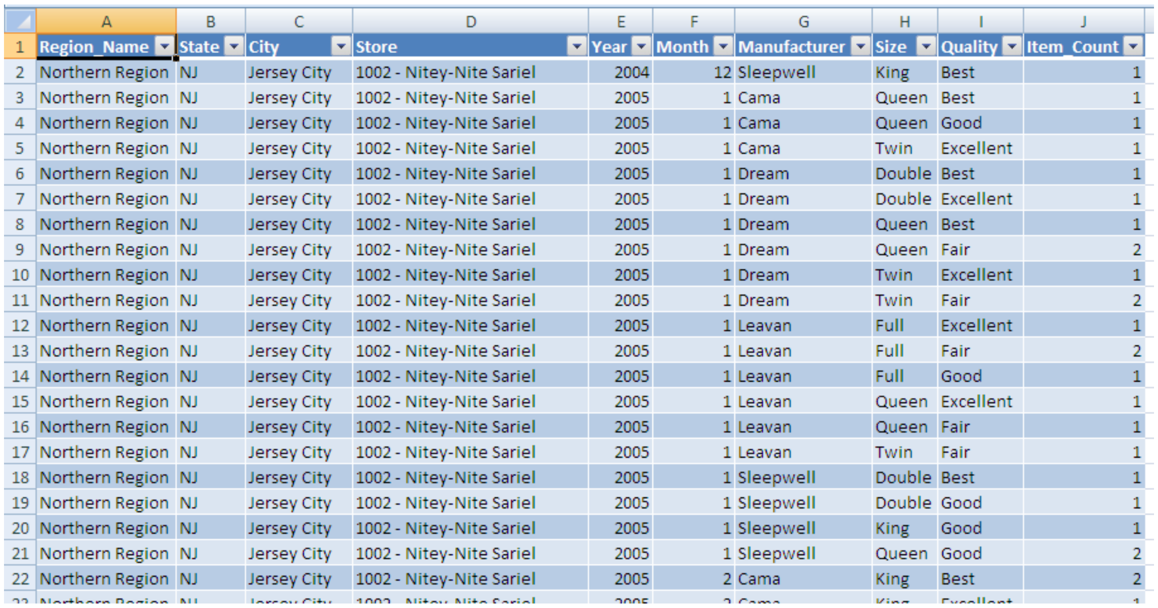
ALL
1001 - Nitey-Nite Mi
1002 - Nitey-Nite Sa
1005 - Nitey-Nite Gl
1009 - Nitey-Nite Isi
1011 - Nitey-Nite Mcl
1012 - Nitey-Nite Re
1018 - Nitey-Nite Hia

Figure 15.30

6. *In* **Cell D4,** *create a* **data validation list** *that contains the values* **2008**, **2009**, **2010**, *and* **2011**.
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 2010** *folder and click on the* **Mattress_Mix.accdb** *file and click* **OK**.
5. *In the first screen of the* **Query Wizard**, *bring in all fields from* **qryMattress_Mix**, *click* **Next >** *three times, then click* **Finish**.
6. *After a few seconds, the* **Import Data** *dialog box should respond to put the data in* **Cell A1**. *Click* **OK**.

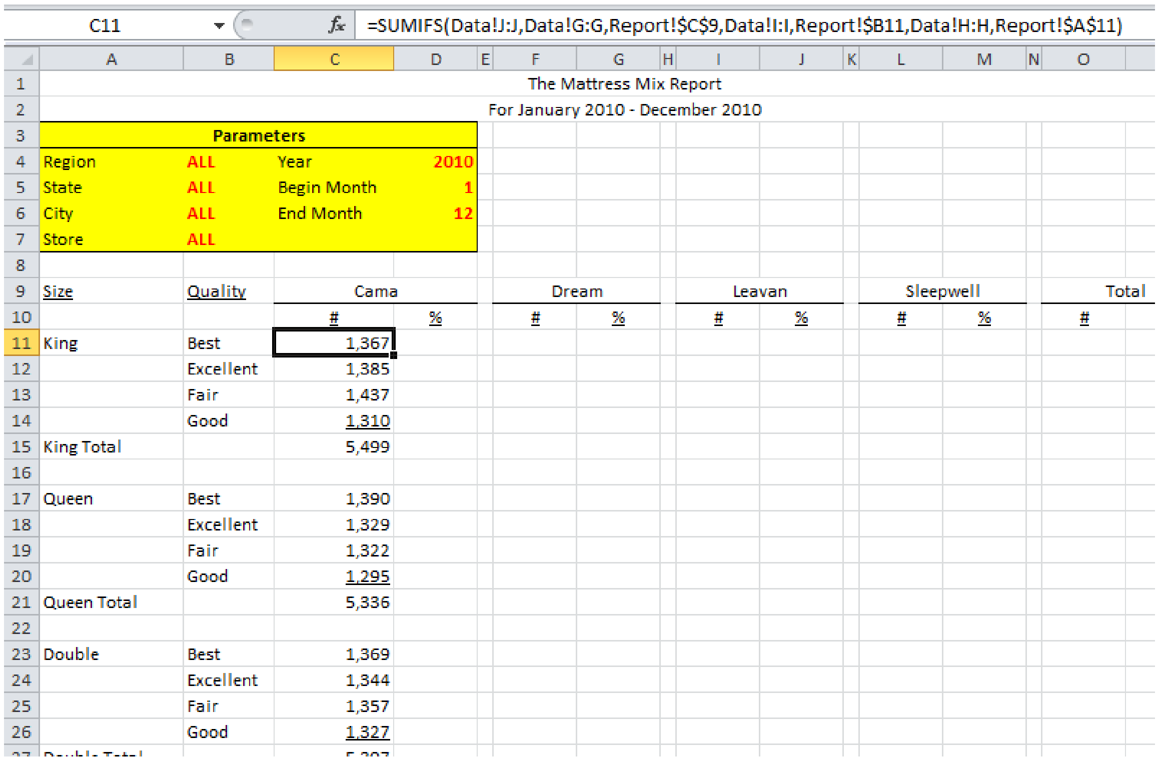


	A	B	C	D	E	F	G	H	I	J
1	Region_Name	State	City	Store	Year	Month	Manufacturer	Size	Quality	Item_Count
2	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2004	12	Sleepwell	King	Best	1
3	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2005	1	Cama	Queen	Best	1
4	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2005	1	Cama	Queen	Good	1
5	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2005	1	Cama	Twin	Excellent	1
6	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2005	1	Dream	Double	Best	1
7	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2005	1	Dream	Double	Excellent	1
8	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2005	1	Dream	Queen	Best	1
9	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2005	1	Dream	Queen	Fair	2
10	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2005	1	Dream	Twin	Excellent	1
11	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2005	1	Dream	Twin	Fair	2
12	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2005	1	Leavan	Full	Excellent	1
13	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2005	1	Leavan	Full	Fair	2
14	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2005	1	Leavan	Full	Good	1
15	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2005	1	Leavan	Queen	Excellent	1
16	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2005	1	Leavan	Queen	Fair	1
17	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2005	1	Leavan	Twin	Fair	1
18	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2005	1	Sleepwell	Double	Best	1
19	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2005	1	Sleepwell	Double	Good	1
20	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2005	1	Sleepwell	King	Good	1
21	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2005	1	Sleepwell	Queen	Good	2
22	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2005	2	Cama	King	Best	2
23	Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2005	2	Cama	King	Excellent	1

Figure 15.31

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 in 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.*

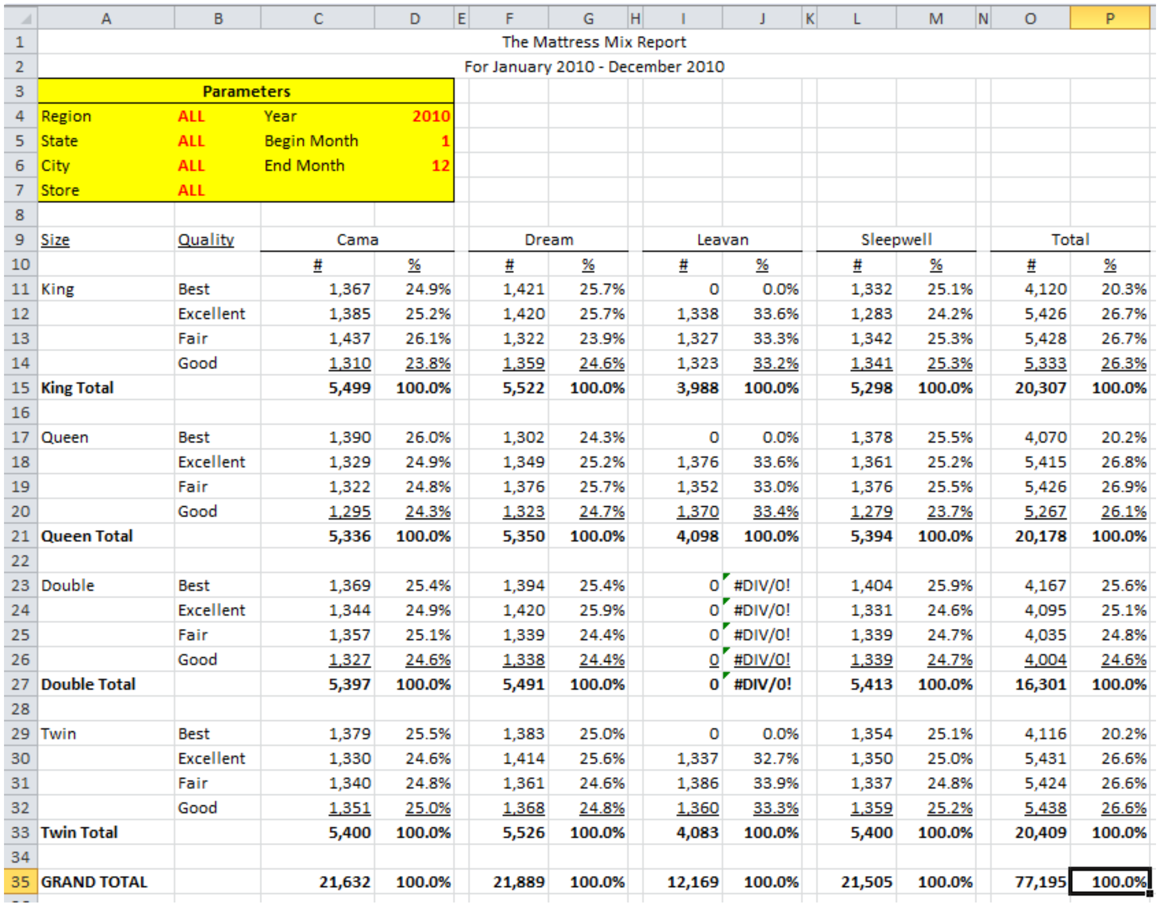


C11 =SUMIFS(Data!J:J,Data!G:G,Report!$C$9,Data!I:I,Report!$B11,Data!H:H,Report!$A$11)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1						The Mattress Mix Report									
2						For January 2010 - December 2010									
3	Parameters														
4	Region	ALL	Year	2010											
5	State	ALL	Begin Month	1											
6	City	ALL	End Month	12											
7	Store	ALL													
8															
9	Size	Quality	Cama			Dream			Leavan			Sleepwell			Total
10			#	%		#	%		#	%		#	%		#
11	King	Best	1,367												
12		Excellent	1,385												
13		Fair	1,437												
14		Good	1,310												
15	King Total		5,499												
16															
17	Queen	Best	1,390												
18		Excellent	1,329												
19		Fair	1,322												
20		Good	1,295												
21	Queen Total		5,336												
22															
23	Double	Best	1,369												
24		Excellent	1,344												
25		Fair	1,357												
26		Good	1,327												

Figure 15.32

3. *Create the appropriate* **% of Total** *formulas in* **Column D**.
4. *Bold the* **Total** *lines.*
5. *Copy of the formulas to the remaining cells to tie with the figures in* **Figure 15.34**.



The Mattress Mix Report

For January 2010 - December 2010

Parameters			
Region	ALL	Year	2010
State	ALL	Begin Month	1
City	ALL	End Month	12
Store	ALL		

Size	Quality	Cama		Dream		Leavan		Sleepwell		Total	
		#	%	#	%	#	%	#	%	#	%
King	Best	1,367	24.9%	1,421	25.7%	0	0.0%	1,332	25.1%	4,120	20.3%
	Excellent	1,385	25.2%	1,420	25.7%	1,338	33.6%	1,283	24.2%	5,426	26.7%
	Fair	1,437	26.1%	1,322	23.9%	1,327	33.3%	1,342	25.3%	5,428	26.7%
	Good	1,310	23.8%	1,359	24.6%	1,323	33.2%	1,341	25.3%	5,333	26.3%
King Total		**5,499**	**100.0%**	**5,522**	**100.0%**	**3,988**	**100.0%**	**5,298**	**100.0%**	**20,307**	**100.0%**
Queen	Best	1,390	26.0%	1,302	24.3%	0	0.0%	1,378	25.5%	4,070	20.2%
	Excellent	1,329	24.9%	1,349	25.2%	1,376	33.6%	1,361	25.2%	5,415	26.8%
	Fair	1,322	24.8%	1,376	25.7%	1,352	33.0%	1,376	25.5%	5,426	26.9%
	Good	1,295	24.3%	1,323	24.7%	1,370	33.4%	1,279	23.7%	5,267	26.1%
Queen Total		**5,336**	**100.0%**	**5,350**	**100.0%**	**4,098**	**100.0%**	**5,394**	**100.0%**	**20,178**	**100.0%**
Double	Best	1,369	25.4%	1,394	25.4%	0	#DIV/0!	1,404	25.9%	4,167	25.6%
	Excellent	1,344	24.9%	1,420	25.9%	0	#DIV/0!	1,331	24.6%	4,095	25.1%
	Fair	1,357	25.1%	1,339	24.4%	0	#DIV/0!	1,339	24.7%	4,035	24.8%
	Good	1,327	24.6%	1,338	24.4%	0	#DIV/0!	1,339	24.7%	4,004	24.6%
Double Total		**5,397**	**100.0%**	**5,491**	**100.0%**	**0**	**#DIV/0!**	**5,413**	**100.0%**	**16,301**	**100.0%**
Twin	Best	1,379	25.5%	1,383	25.0%	0	0.0%	1,354	25.1%	4,116	20.2%
	Excellent	1,330	24.6%	1,414	25.6%	1,337	32.7%	1,350	25.0%	5,431	26.6%
	Fair	1,340	24.8%	1,361	24.6%	1,386	33.9%	1,337	24.8%	5,424	26.6%
	Good	1,351	25.0%	1,368	24.8%	1,360	33.3%	1,359	25.2%	5,438	26.6%
Twin Total		**5,400**	**100.0%**	**5,526**	**100.0%**	**4,083**	**100.0%**	**5,400**	**100.0%**	**20,409**	**100.0%**
GRAND TOTAL		**21,632**	**100.0%**	**21,889**	**100.0%**	**12,169**	**100.0%**	**21,505**	**100.0%**	**77,195**	**100.0%**

Figure 15.33

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 in the Data tab.

7. *In the* **Data** *tab, filter the table for* **Leavan***.*
8. *With the* **Leavan** *filter on, click on the* **Quality** *filter.*

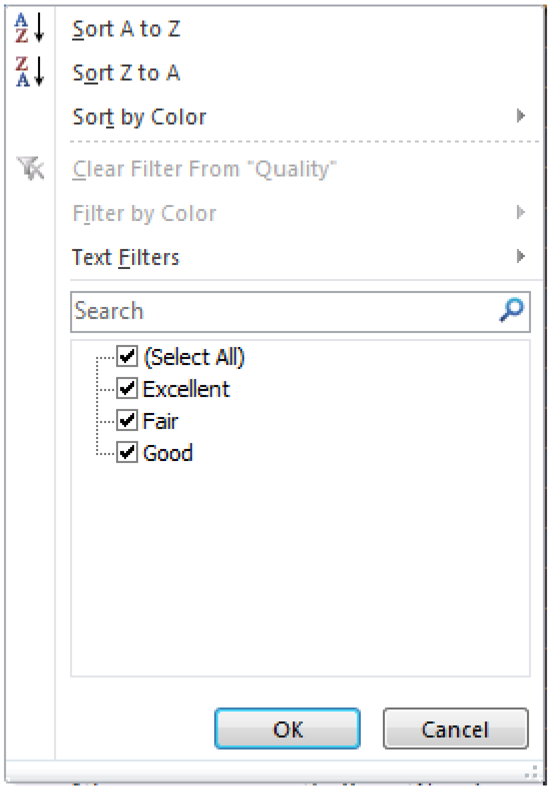


Figure 15.34

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.*

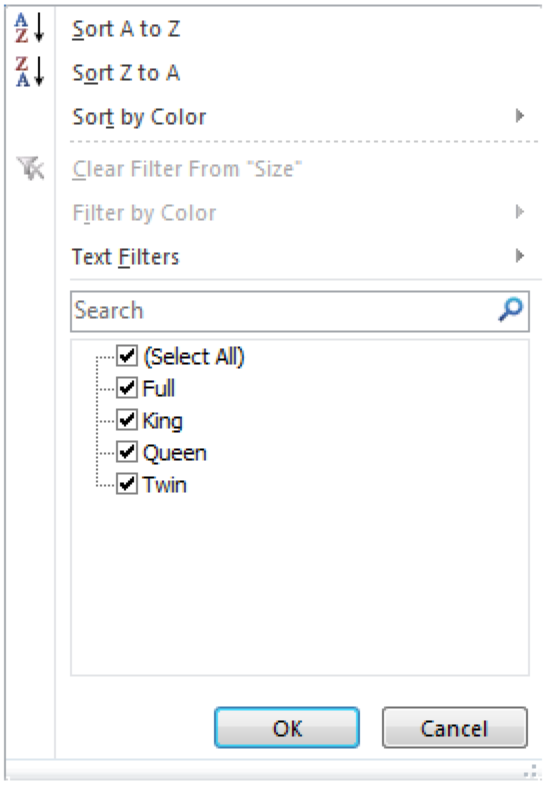
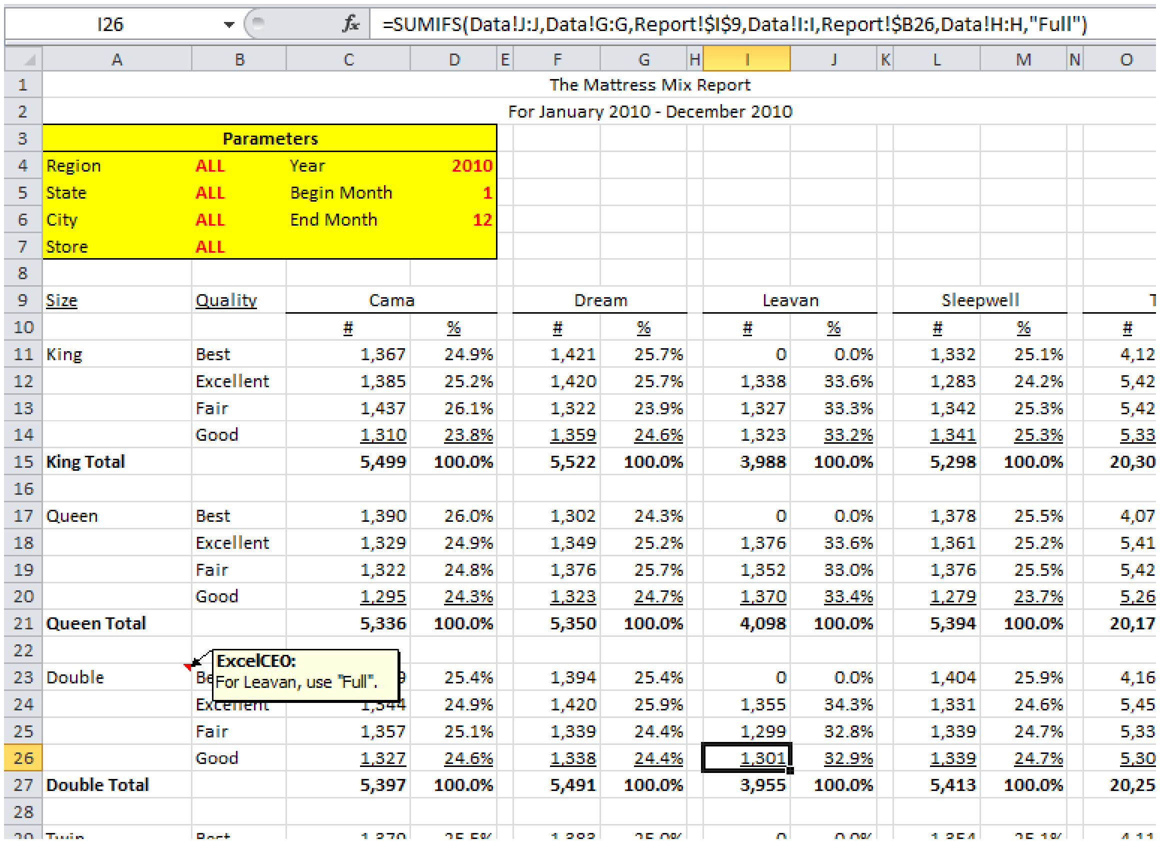


Figure 15.35

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.*
11. *Change the formula in* **Cell I23** *to reflect a* **Full** *size instead of* **Double** *and copy to the appropriate cells below.*
12. *Create a comment in* **Cell A23** *to let the users know that for* **Leavan** *you used* **Full** *instead of* **Double**.

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 2010 Review Questions Chapter 15, Section 1 of 2*** *option and complete the review questions.*



I26 =SUMIFS(Data!J:J,Data!G:G,Report!$I$9,Data!I:I,Report!$B26,Data!H:H,"Full")

	A	B	C	D	F	G	I	J	L	M	O
1						The Mattress Mix Report					
2						For January 2010 - December 2010					
3		Parameters									
4	Region	ALL	Year	2010							
5	State	ALL	Begin Month	1							
6	City	ALL	End Month	12							
7	Store	ALL									
8											
9	Size	Quality	Cama		Dream		Leavan		Sleepwell		T
10			#	%	#	%	#	%	#	%	#
11	King	Best	1,367	24.9%	1,421	25.7%	0	0.0%	1,332	25.1%	4,12
12		Excellent	1,385	25.2%	1,420	25.7%	1,338	33.6%	1,283	24.2%	5,42
13		Fair	1,437	26.1%	1,322	23.9%	1,327	33.3%	1,342	25.3%	5,42
14		Good	1,310	23.8%	1,359	24.6%	1,323	33.2%	1,341	25.3%	5,33
15	**King Total**		**5,499**	**100.0%**	**5,522**	**100.0%**	**3,988**	**100.0%**	**5,298**	**100.0%**	**20,30**
16											
17	Queen	Best	1,390	26.0%	1,302	24.3%	0	0.0%	1,378	25.5%	4,07
18		Excellent	1,329	24.9%	1,349	25.2%	1,376	33.6%	1,361	25.2%	5,41
19		Fair	1,322	24.8%	1,376	25.7%	1,352	33.0%	1,376	25.5%	5,42
20		Good	1,295	24.3%	1,323	24.7%	1,370	33.4%	1,279	23.7%	5,26
21	**Queen Total**		**5,336**	**100.0%**	**5,350**	**100.0%**	**4,098**	**100.0%**	**5,394**	**100.0%**	**20,17**
22											
23	Double	Be	9	25.4%	1,394	25.4%	0	0.0%	1,404	25.9%	4,16
24		Excellent	1,344	24.9%	1,420	25.9%	1,355	34.3%	1,331	24.6%	5,45
25		Fair	1,357	25.1%	1,339	24.4%	1,299	32.8%	1,339	24.7%	5,33
26		Good	1,327	24.6%	1,338	24.4%	1,301	32.9%	1,339	24.7%	5,30
27	**Double Total**		**5,397**	**100.0%**	**5,491**	**100.0%**	**3,955**	**100.0%**	**5,413**	**100.0%**	**20,25**
28											

ExcelCEO:
For Leavan, use "Full".

Figure 15.36

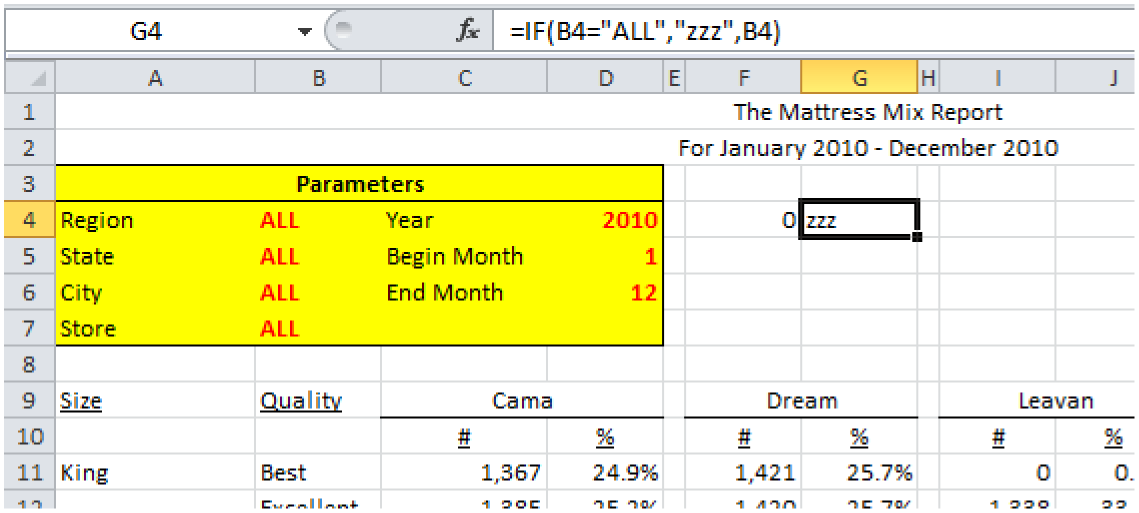
Now your totals should match the total on the Data tab, 81,150.

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)**

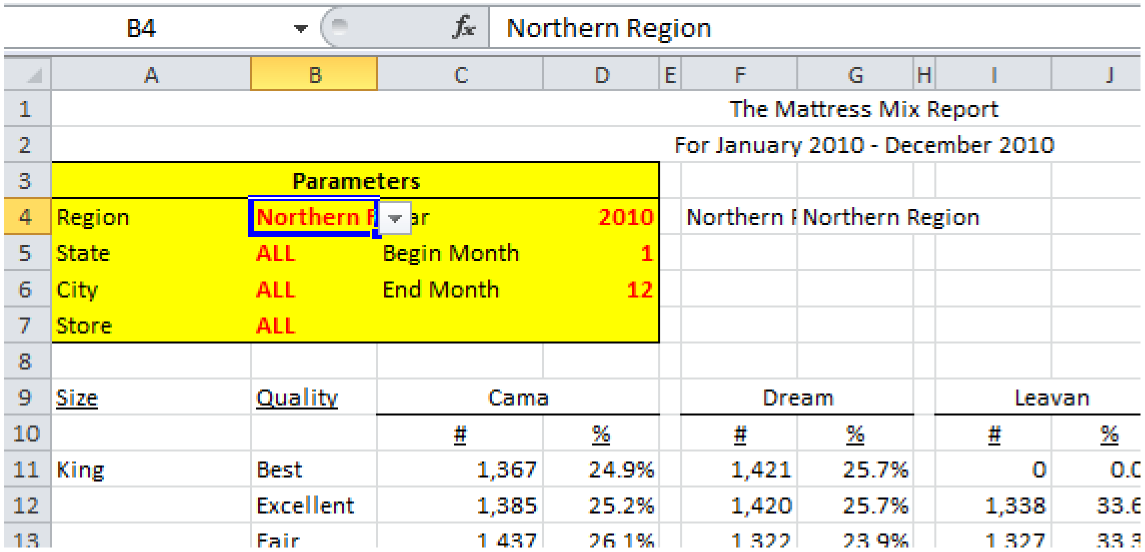


G4 =IF(B4="ALL","zzz",B4)

	A	B	C	D	E	F	G	H	I	J
1						The Mattress Mix Report				
2						For January 2010 - December 2010				
3	Parameters									
4	Region	ALL	Year	2010		0	zzz			
5	State	ALL	Begin Month	1						
6	City	ALL	End Month	12						
7	Store	ALL								
8										
9	Size	Quality	Cama			Dream			Leavan	
10			#	%		#	%		#	%
11	King	Best	1,367	24.9%		1,421	25.7%		0	0.

Figure 15.37

3. *Change* **Cell B4** *to be* **Northern Region**.

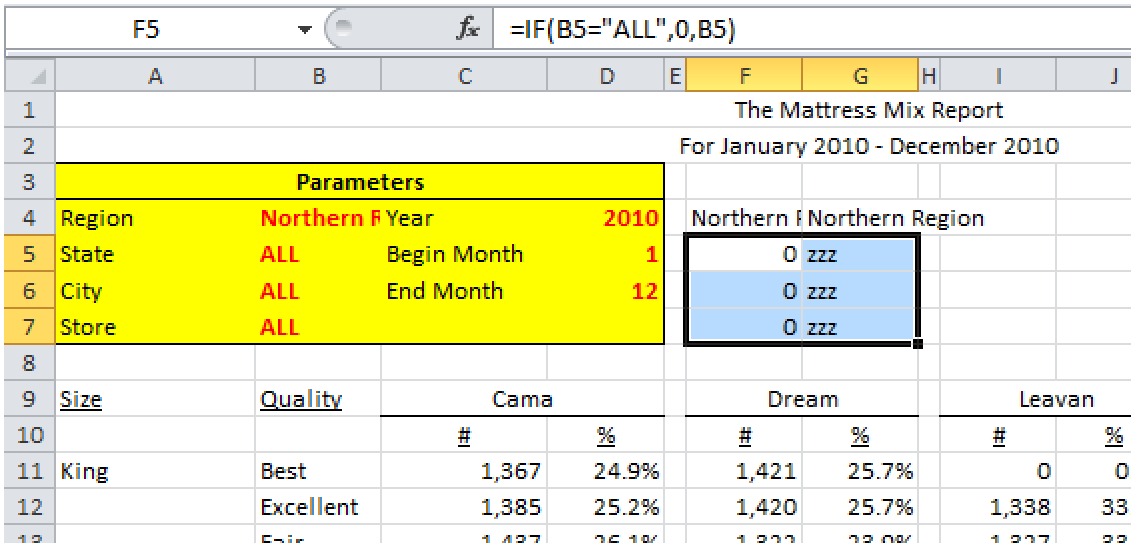


B4 Northern Region

	A	B	C	D	E	F	G	H	I	J
1						The Mattress Mix Report				
2						For January 2010 - December 2010				
3	Parameters									
4	Region	Northern R	ar	2010		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	
10			#	%		#	%		#	%
11	King	Best	1,367	24.9%		1,421	25.7%		0	0.0
12		Excellent	1,385	25.2%		1,420	25.7%		1,338	33.6

Figure 15.38

4. *Copy* **Cells F4** *and* **G4** *down to* **Cells F5 – G7**.



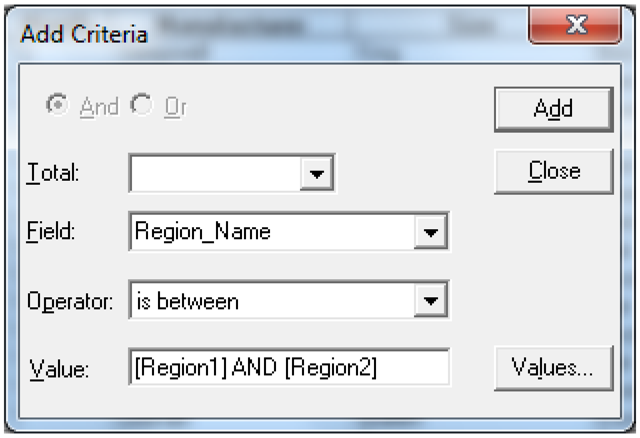
F5 =IF(B5="ALL",0,B5)

	A	B	C	D	E	F	G	H	I	J
1						The Mattress Mix Report				
2						For January 2010 - December 2010				
3	Parameters									
4	Region	Northern F	Year	2010		Northern F	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	
10			#	%		#	%		#	%
11	King	Best	1,367	24.9%		1,421	25.7%		0	0
12		Excellent	1,385	25.2%		1,420	25.7%		1,338	33

Figure 15.39

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]**

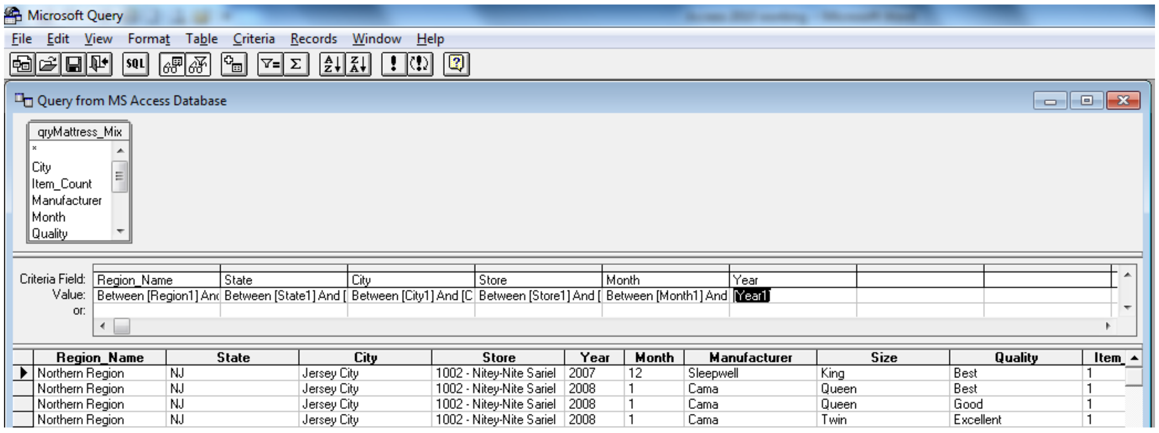


Total:
Field: Region_Name
Operator: is between
Value: [Region1] AND [Region2]

Figure 15.40

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)*


Microsoft Query
File Edit View Format Table Criteria Records Window Help
Query from MS Access Database
qryMattress_Mix
*
City
Item_Count
Manufacturer
Month
Quality

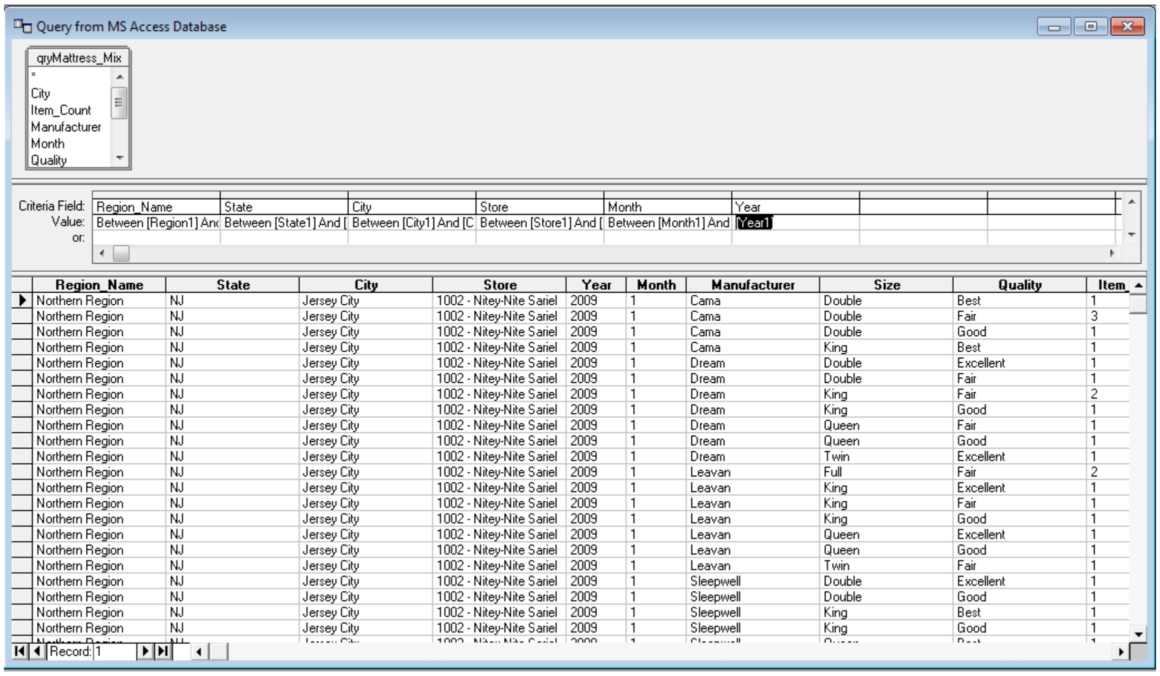
Criteria Field:	Region_Name	State	City	Store	Month	Year
Value:	Between [Region1] Anc	Between [State1] And [	Between [City1] And [C	Between [Store1] And [	Between [Month1] And	[Year1]
or:						

Region_Name	State	City	Store	Year	Month	Manufacturer	Size	Quality	Item_
Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2007	12	Sleepwell	King	Best	1
Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2008	1	Cama	Queen	Best	1
Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2008	1	Cama	Queen	Good	1
Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2008	1	Cama	Twin	Excellent	1

Figure 15.41

11. Run the query by clicking on the **exclamation point (Query Now)** *icon.*
12. Input the following values when prompted:

Region1: 0
Region2: zzz
State1: 0
State2: zzz
City1: 0
City2: zzz
Store1: 0
Store2: zzz
Month1: 1
Month2: 6
Year: 2009


Query from MS Access Database

qryMattress_Mix
*
City
Item_Count
Manufacturer
Month
Quality

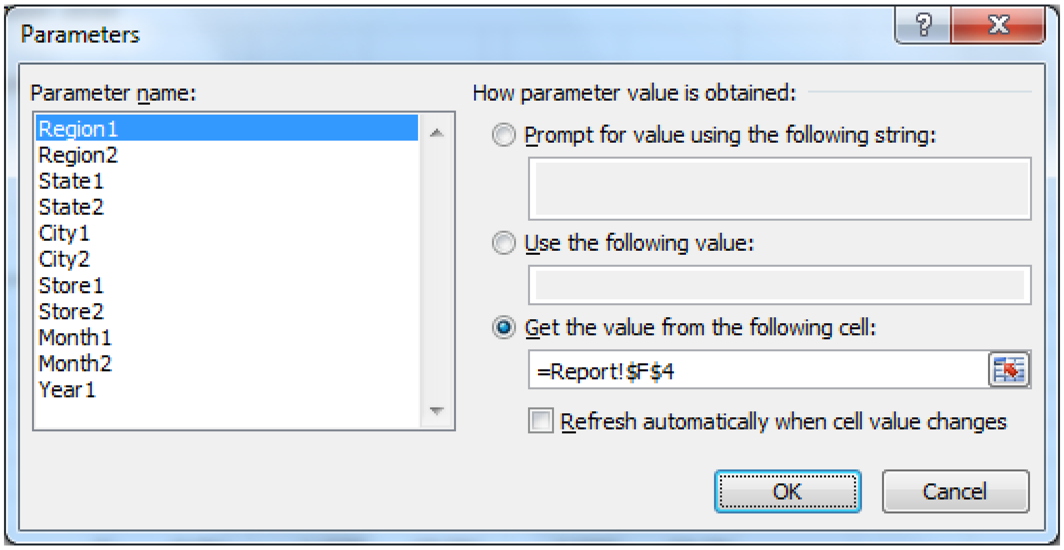
Criteria Field:	Region_Name	State	City	Store	Month	Year
Value:	Between [Region1] An	Between [State1] And [	Between [City1] And [C	Between [Store1] And [	Between [Month1] And	[Year1]
or:						

Region_Name	State	City	Store	Year	Month	Manufacturer	Size	Quality	Item_
Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2009	1	Cama	Double	Best	1
Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2009	1	Cama	Double	Fair	3
Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2009	1	Cama	Double	Good	1
Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2009	1	Cama	King	Best	1
Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2009	1	Dream	Double	Excellent	1
Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2009	1	Dream	Double	Fair	1
Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2009	1	Dream	King	Fair	2
Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2009	1	Dream	King	Good	1
Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2009	1	Dream	Queen	Fair	1
Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2009	1	Dream	Queen	Good	1
Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2009	1	Dream	Twin	Excellent	1
Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2009	1	Leavan	Full	Fair	2
Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2009	1	Leavan	King	Excellent	1
Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2009	1	Leavan	King	Fair	1
Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2009	1	Leavan	King	Good	1
Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2009	1	Leavan	Queen	Excellent	1
Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2009	1	Leavan	Queen	Good	1
Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2009	1	Leavan	Twin	Fair	1
Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2009	1	Sleepwell	Double	Excellent	1
Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2009	1	Sleepwell	Double	Good	1
Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2009	1	Sleepwell	King	Best	1
Northern Region	NJ	Jersey City	1002 - Nitey-Nite Sariel	2009	1	Sleepwell	King	Good	1

Record: 1

Figure 15.42

13. *Exit out of* **Microsoft Query***.*
14. *In the* **Connection Properties** *dialog box, click on the* **Parameters…** *button.*
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.*


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:
(●) Get the value from the following cell:
=Report!$F$4
☐ Refresh automatically when cell value changes

Figure 15.43

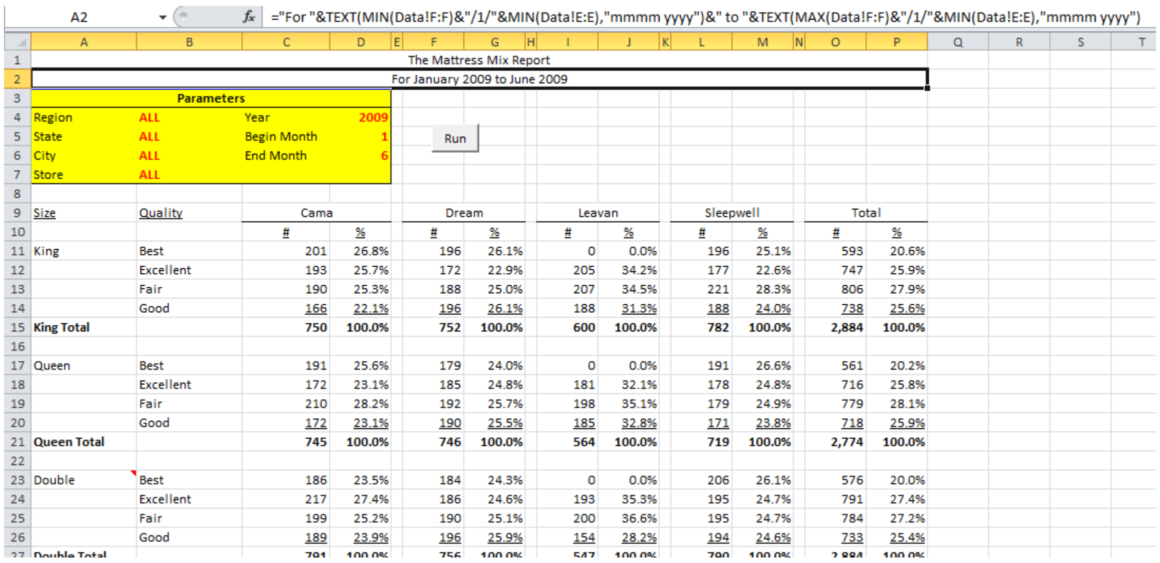
17. *Click* **OK** *and exit out of* **Microsoft Query** *and close the* **Workbook Connections** *dialog box.*

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.*



A2 `="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")`

The Mattress Mix Report

For January 2009 to June 2009

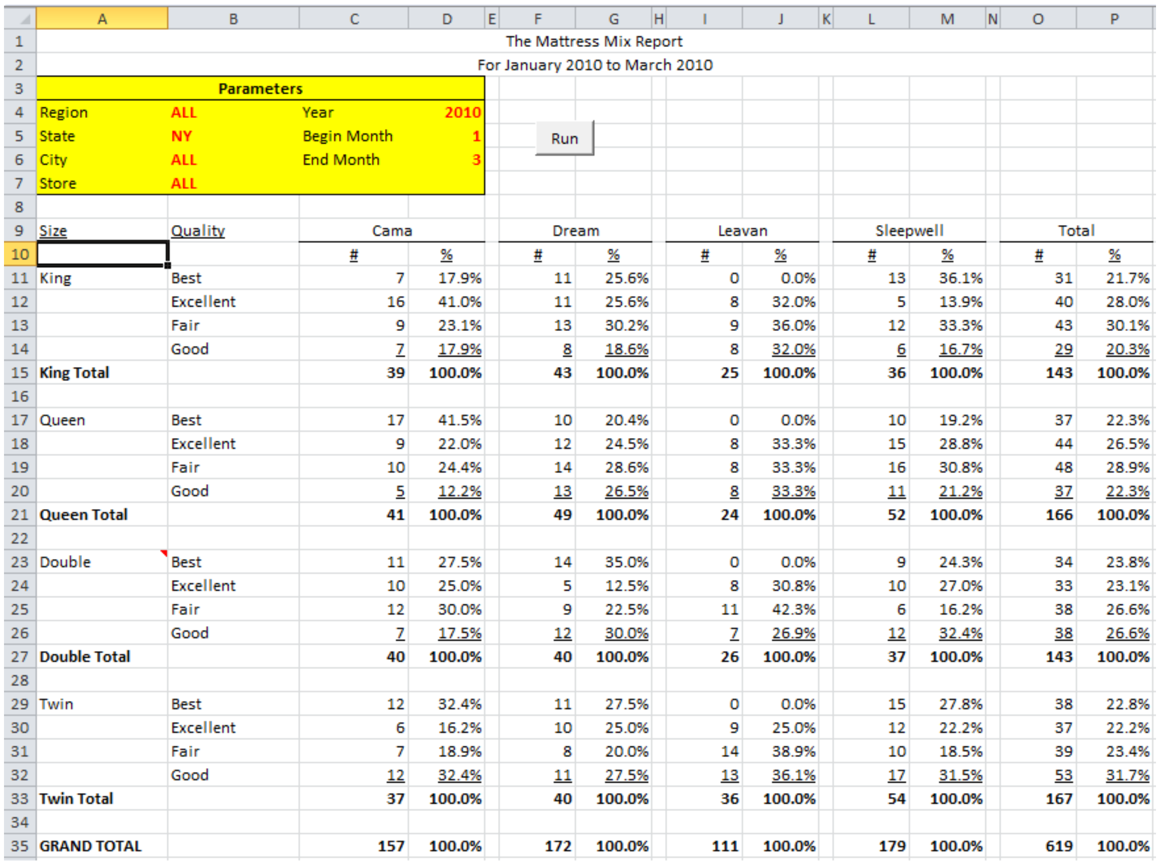
Parameters			
Region	ALL	Year	2009
State	ALL	Begin Month	1
City	ALL	End Month	6
Store	ALL		

Size	Quality	Cama		Dream		Leavan		Sleepwell		Total	
		#	%	#	%	#	%	#	%	#	%
King	Best	201	26.8%	196	26.1%	0	0.0%	196	25.1%	593	20.6%
	Excellent	193	25.7%	172	22.9%	205	34.2%	177	22.6%	747	25.9%
	Fair	190	25.3%	188	25.0%	207	34.5%	221	28.3%	806	27.9%
	Good	166	22.1%	196	26.1%	188	31.3%	188	24.0%	738	25.6%
King Total		**750**	**100.0%**	**752**	**100.0%**	**600**	**100.0%**	**782**	**100.0%**	**2,884**	**100.0%**
Queen	Best	191	25.6%	179	24.0%	0	0.0%	191	26.6%	561	20.2%
	Excellent	172	23.1%	185	24.8%	181	32.1%	178	24.8%	716	25.8%
	Fair	210	28.2%	192	25.7%	198	35.1%	179	24.9%	779	28.1%
	Good	172	23.1%	190	25.5%	185	32.8%	171	23.8%	718	25.9%
Queen Total		**745**	**100.0%**	**746**	**100.0%**	**564**	**100.0%**	**719**	**100.0%**	**2,774**	**100.0%**
Double	Best	186	23.5%	184	24.3%	0	0.0%	206	26.1%	576	20.0%
	Excellent	217	27.4%	186	24.6%	193	35.3%	195	24.7%	791	27.4%
	Fair	199	25.2%	190	25.1%	200	36.6%	195	24.7%	784	27.2%
	Good	189	23.9%	196	25.9%	154	28.2%	194	24.6%	733	25.4%
Double Total		**791**	**100.0%**	**756**	**100.0%**	**547**	**100.0%**	**790**	**100.0%**	**2,884**	**100.0%**

Figure 15.44

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 2010**.



The Mattress Mix Report

For January 2010 to March 2010

Parameters			
Region	ALL	Year	2010
State	NY	Begin Month	1
City	ALL	End Month	3
Store	ALL		

Run

Size	Quality	Cama		Dream		Leavan		Sleepwell		Total	
		#	%	#	%	#	%	#	%	#	%
King	Best	7	17.9%	11	25.6%	0	0.0%	13	36.1%	31	21.7%
	Excellent	16	41.0%	11	25.6%	8	32.0%	5	13.9%	40	28.0%
	Fair	9	23.1%	13	30.2%	9	36.0%	12	33.3%	43	30.1%
	Good	7	17.9%	8	18.6%	8	32.0%	6	16.7%	29	20.3%
King Total		**39**	**100.0%**	**43**	**100.0%**	**25**	**100.0%**	**36**	**100.0%**	**143**	**100.0%**
Queen	Best	17	41.5%	10	20.4%	0	0.0%	10	19.2%	37	22.3%
	Excellent	9	22.0%	12	24.5%	8	33.3%	15	28.8%	44	26.5%
	Fair	10	24.4%	14	28.6%	8	33.3%	16	30.8%	48	28.9%
	Good	5	12.2%	13	26.5%	8	33.3%	11	21.2%	37	22.3%
Queen Total		**41**	**100.0%**	**49**	**100.0%**	**24**	**100.0%**	**52**	**100.0%**	**166**	**100.0%**
Double	Best	11	27.5%	14	35.0%	0	0.0%	9	24.3%	34	23.8%
	Excellent	10	25.0%	5	12.5%	8	30.8%	10	27.0%	33	23.1%
	Fair	12	30.0%	9	22.5%	11	42.3%	6	16.2%	38	26.6%
	Good	7	17.5%	12	30.0%	7	26.9%	12	32.4%	38	26.6%
Double Total		**40**	**100.0%**	**40**	**100.0%**	**26**	**100.0%**	**37**	**100.0%**	**143**	**100.0%**
Twin	Best	12	32.4%	11	27.5%	0	0.0%	15	27.8%	38	22.8%
	Excellent	6	16.2%	10	25.0%	9	25.0%	12	22.2%	37	22.2%
	Fair	7	18.9%	8	20.0%	14	38.9%	10	18.5%	39	23.4%
	Good	12	32.4%	11	27.5%	13	36.1%	17	31.5%	53	31.7%
Twin Total		**37**	**100.0%**	**40**	**100.0%**	**36**	**100.0%**	**54**	**100.0%**	**167**	**100.0%**
GRAND TOTAL		**157**	**100.0%**	**172**	**100.0%**	**111**	**100.0%**	**179**	**100.0%**	**619**	**100.0%**

Figure 15.45

YOU DID IT!!! At this point, you have a fully functioning reporting tool that ANYONE would be proud of.

22. Save the file 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 2010 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!!!

Chapter Exam

You can now go to www.ExcelCEO.com, log in and take the exam. Make sure that you take the exam on the same computer that you completed the sample files on, as some of the questions on the exam may refer to some of the completed examples.

GLOSSARY

Term or Acronym	Definition
Chapter Examination	The exam that is administered at the end of each chapter to test whether or not the participant has learned the material. A passing grade of 70% or above must be obtained before continuing to the next chapter.
CPE	Continuing Professional Education. A course by which people maintain their knowledge and skills for a particular profession.
Data Source	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.
Dialog Box	A small window on a computer screen that communicates information to the user and prompts them for a response.
Formula	Equations that perform calculations on values, dates or text string in an Excel worksheet or Access database.
Function	A preset formula in Microsoft Excel or Access.
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.
Hyperlink	A formatted text string on a browser that when clicked opens a bookmark, Internet page, or file.
Icon	A pictorial representation of an object. When clicked, the icon performs a predesigned task.
Keyboard Shortcut	A series of keystrokes that executes a certain predefined (by Microsoft Office or programmed by the user) functionality without having to choose the options from the Office Ribbon with a mouse.
Nesting Functions	Using multiple functions in one formula.
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 (<>).
Quick Access Toolbar	A predefined set of icons located at the upper left corner of the Excel spreadsheet or Access database that contains a number of shortcuts to various functionalities, like Open, Save, Undo and quick print.
Review Questions	A set of non-graded questions the participant must answer before taking the Chapter Examination.

User ID	Used when logging in to the ExcelCEO training program. Most often, it is the same as the user’s email address.

INDEX