TUTORIAL **B**

MICROSOFT ACCESS

Microsoft Access is a relational database package that runs on the Microsoft Windows operating system. There are many different versions of Access; this tutorial was prepared using Access 2013.

Before using this tutorial, you should know the fundamentals of Access and know how to use Windows. This tutorial explains advanced Access skills you will need to complete database case studies. The tutorial concludes with a discussion of common Access problems and how to solve them.

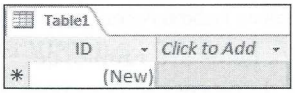
To prevent losing your work, always observe proper file-saving and closing procedures. To exit Access, click the File tab and select Close, then click the Close button in the upper-right corner. Always end your work with these steps. If you remove your USB key or other portable storage device when database forms and tables are shown on the screen, you will lose your work.

To begin this tutorial, you will create a new database called Employee.

AT THE KEYBOARD

Open Access. Click the Blank desktop database icon from the templates list. Name the database Employee. Click the file folder next to the filename to browse for the folder where you want to save the file. Click the new folder, click Open, and then click OK. Otherwise, your file will be saved automatically in the Documents folder. Click the Create button.

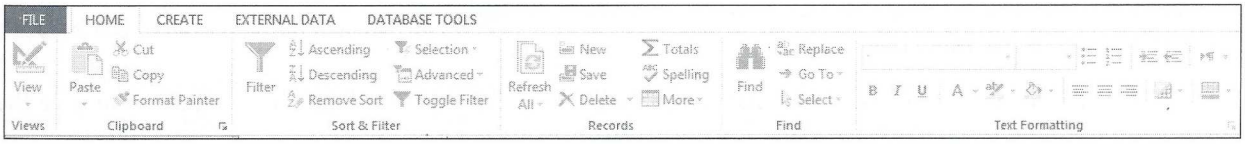
A portion of your opening screen should resemble the screen shown in Figure B-1.



Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-1 Entering data in Datasheet view

When you create a table, Access opens it in Datasheet view by default. Because you will use Design view to build your tables, close the new table by clicking the *X* in the upper-right corner of the table window that corresponds to Close "Table1." You are now on the Home tab in the Database window of Access, as shown in Figure B-2. From this screen, you can create or change objects.



Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-2 The Database window Home tab in Access

CREATING TABLES

Your database will contain data about employees, their wage rates, and the hours they worked.

Defining Tables

In the Database window, build three new tables using the following instructions.

AT THE KEYBOARD

Defining the Employee Table

This table contains permanent data about employees. To create the table, click the Create tab and then click Table Design in the Tables group. The table's fields are Last Name, First Name, Employee ID, Street Address, City, State, Zip, Date Hired, and US Citizen. The Employee ID field is the primary key field. Change the lengths of Short Text fields from the default 255 spaces to more appropriate lengths; for example, the Last Name field might be 30 spaces, and the Zip field might be 10 spaces. Your completed definition should resemble the one shown in Figure B-3.

Table1

Field Name	Data Type	Description (Optional)
Last Name	Short Text	
First Name	Short Text	
Employee ID	Short Text	
Street Address	Short Text	
City	Short Text	
State	Short Text	
Zip	Short Text	
Date Hired	Date/Time	
US Citizen	Yes/No	

Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-3 Fields in the Employee table

When you finish, click the File tab, select Save As, select Save Object As, click the Save As button, and then enter a name for the table. In this example, the table is named Employee. (It is a coincidence that the Employee table has the same name as its database file.) After entering the name, click OK in the Save As window. Close the table by clicking the Close button (X) that corresponds to the Employee table.

Defining the Wage Data Table

This table contains permanent data about employees and their wage rates. The table's fields are Employee ID, Wage Rate, and Salaried. The Employee ID field is the primary key field. Use the data types shown in Figure B-4. Your definition should resemble the one shown in Figure B-4.

Table1

Field Name	Data Type	Description (Optional)
Employee ID	Short Text	
Wage Rate	Currency	
Salaried	Yes/No	

Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-4 Fields in the Wage Data table

Click the File tab and then select Save As, select Save Object As, and click the Save As button to save the table definition. Name the table Wage Data.

Defining the Hours Worked Table

The purpose of this table is to record the number of hours that employees work each week during the year. The table's three fields are Employee ID (which has a Short Text data type), Week # (number–long integer), and Hours (number–double). The Employee ID and Week # are the compound keys.

In the following example, the employee with ID number 08965 worked 40 hours in Week 1 of the year and 52 hours in Week 2.

Employee ID	Week #	Hours
08965	1	40
08965	2	52

Note that no single field can be the primary key field because 08965 is an entry for each week. In other words, if this employee works each week of the year, 52 records will have the same Employee ID value at the end of the year. Thus, Employee ID values will not distinguish records. No other single field can distinguish these records either, because other employees will have worked during the same week number and some employees will have worked the same number of hours. For example, 40 hours—which corresponds to a full-time workweek—would be a common entry for many weeks.

All of this presents a problem because a table must have a primary key field in Access. The solution is to use a compound primary key; that is, use values from more than one field to create a combined field that will distinguish records. The best compound key to use for the current example consists of the Employee ID field and the Week # field, because as each person works each week, the week number changes. In other words, there is only *one* combination of Employee ID 08965 and Week # 1. Because those values *can occur in only one record*, the combination distinguishes that record from all others.

The first step of setting a compound key is to highlight the fields in the key. Those fields must appear one after the other in the table definition screen. (Plan ahead for that format.) As an alternative, you can highlight one field, hold down the Control key, and highlight the next field.

AT THE KEYBOARD

In the Hours Worked table, click the first field's left prefix area (known as the row selector), hold down the mouse button, and drag down to highlight the names of all fields in the compound primary key. Your screen should resemble the one shown in Figure B-5.

Table1

Field Name	Data Type	Description (Optional)
Employee ID	Short Text	
Week #	Number	
Hours	Number	

Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-5 Selecting fields for the compound primary key for the Hours Worked table

Now click the Key icon. Your screen should resemble the one shown in Figure B-6.

Table1

Field Name	Data Type	Description (Optional)
Employee ID	Short Text	
Week #	Number	
Hours	Number	

Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-6 The compound primary key for the Hours Worked table

You have created the compound primary key and finished defining the table. Click the File tab and then select Save As, select Save Object As, and click the Save As button to save the table as Hours Worked.

Adding Records to a Table

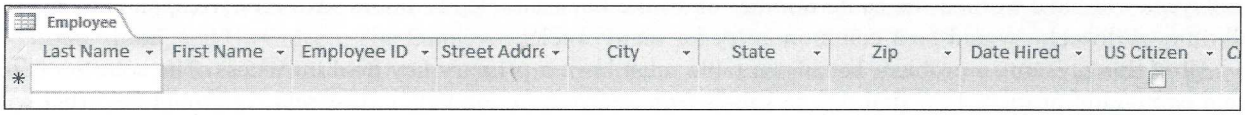
At this point, you have set up the skeletons of three tables. The tables have no data records yet. If you printed the tables now, you would only see column headings (the field names). The most direct way to enter data into a table is to double-click the table's name in the navigation pane at the left side of the screen and then type the data directly into the cells.

NOTE

To display and open the database objects, Access 2013 uses a navigation pane, which is on the left side of the Access window.

AT THE KEYBOARD

On the Home tab of the Database window, double-click the Employee table. Your data entry screen should resemble the one shown in Figure B-7.



Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-7 The data entry screen for the Employee table

The Employee table has many fields, some of which may be off the screen to the right. Scroll to see obscured fields. (Scrolling happens automatically as you enter data.) Figure B-7 shows all of the fields on the screen.

Enter your data one field value at a time. Note that the first row is empty when you begin. Each time you finish entering a value, press Enter to move the cursor to the next cell. After you enter data in the last cell in a row, the cursor moves to the first cell of the next row *and* Access automatically saves the record. Thus, you do not need to click the File tab and then select Save after entering data into a table.

When entering data in your table, you should enter dates in the following format: 6/15/10. Access automatically expands the entry to the proper format in output.

Also note that Yes/No variables are clicked (checked) for Yes; otherwise, the box is left blank for No. You can change the box from Yes to No by clicking it.

Enter the data shown in Figure B-8 into the Employee table. If you make errors in data entry, click the cell, backspace over the error, and type the correction.

Employee

Last Name	First Name	Employee ID	Street Addre	City	State	Zip	Date Hired	US Citizen	Click to Add
Howard	Jane	11411	28 Sally Dr	Glasgow	DE	19702	6/1/2016	☑	
Johnson	John	12345	30 Elm St	Newark	DE	19711	6/1/1996	☑	
Smith	Albert	14890	44 Duce St	Odessa	DE	19722	7/15/1987	☑	
Jones	Sue	22282	18 Spruce St	Newark	DE	19716	7/15/2004	☐	
Ruth	Billy	71460	1 Tater Dr	Baltimore	MD	20111	8/15/1999	☐	
Add	Your	Data	Here	Elkton	MD	21921		☑	
								☐	

Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-8 Data for the Employee table

Note that the sixth record is *your* data record. Assume that you live in Elkton, Maryland, were hired on today's date (enter the date), and are a U.S. citizen. Make up a fictitious Employee ID number. For purposes of this tutorial, the sixth record has been created using the name of one of this text's authors and the employee ID 09911.

After adding records to the Employee table, open the Wage Data table and enter the data shown in Figure B-9.

Wage Data

Employee ID	Wage Rate	Salaried
11411	$10.00	☐
12345	$0.00	☑
14890	$12.00	☐
22282	$0.00	☑
71460	$0.00	☑
Your Employee ID	$8.00	☐
	$0.00	☐

Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-9 Data for the Wage Data table

In this table, you are again asked to create a new entry. For this record, enter your own employee ID. Also assume that you earn $8 an hour and are not salaried. Note that when an employee's Salaried box is not checked (in other words, Salaried = No), the implication is that the employee is paid by the hour. Because salaried employees are not paid by the hour, their hourly rate is 0.00.

When you finish creating the Wage Data table, open the Hours Worked table and enter the data shown in Figure B-10.

Hours Worked

Employee ID	Week #	Hours
11411	1	40
11411	2	50
12345	1	40
12345	2	40
14890	1	38
14890	2	40
22282	1	40
22282	2	40
71460	1	40
71460	2	40
Your Employee ID	1	60
Your Employee ID	2	55
	0	0

Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-10 Data for the Hours Worked table

Notice that salaried employees are always given 40 hours. Nonsalaried employees (including you) might work any number of hours. For your record, enter your fictitious employee ID, 60 hours worked for Week 1, and 55 hours worked for Week 2.

CREATING QUERIES

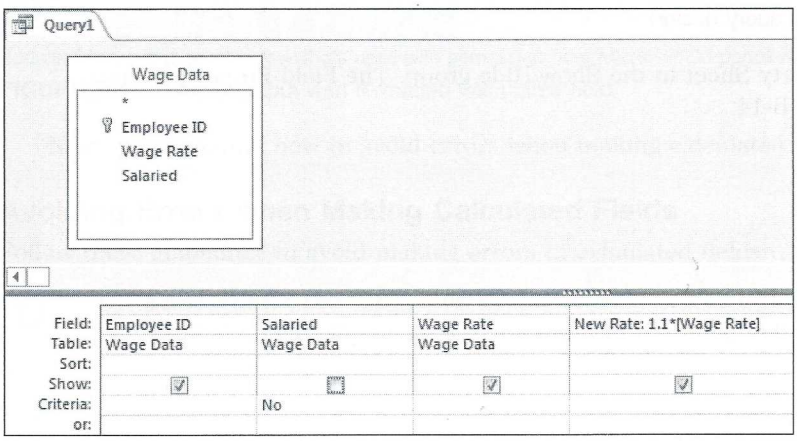
Because you know how to create basic queries, this section explains the advanced queries you will create in the cases in this book.

Using Calculated Fields in Queries

A **calculated field** is an output field made up of *other* field values. A calculated field can be a field in a table; here it is created in the query generator. The calculated field here does not become part of the table—it is just part of the query output. The best way to understand this process is to work through an example.

AT THE KEYBOARD

Suppose you want to see the employee IDs and wage rates of hourly workers, and the new wage rates if all employees were given a 10 percent raise. To view that information, show the employee ID, the current wage rate, and the higher rate, which should be titled New Rate in the output. Figure B-11 shows how to set up the query.



Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-11 Query setup for the calculated field

To set up this query, you need to select hourly workers by using the Salaried field with Criteria = No. Note in Figure B-11 that the Show box for the field is not checked, so the Salaried field values will not appear in the query output.

Note the expression for the calculated field, which you can see in the far-right field cell:

New Rate: 1.1 * [Wage Rate]

The term *New Rate:* merely specifies the desired output heading. (Don't forget the colon.) The rest of the expression, 1.1 * [Wage Rate], multiplies the old wage rate by 110 percent, which results in the 10 percent raise.

In the expression, the field name Wage Rate must be enclosed in square brackets. Remember this rule: *Any time an Access expression refers to a field name, the field name must be enclosed in square brackets.*

If you run this query, your output should resemble the one in Figure B-12.

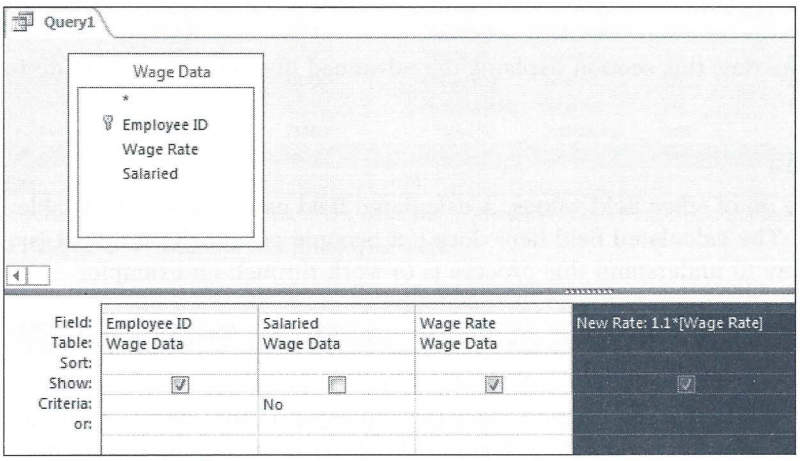
Query1

Employee ID	Wage Rate	New Rate
11411	$10.00	11
14890	$12.00	13.2
09911	$8.00	8.8
*	$0.00	

Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-12 Output for a query with calculated field

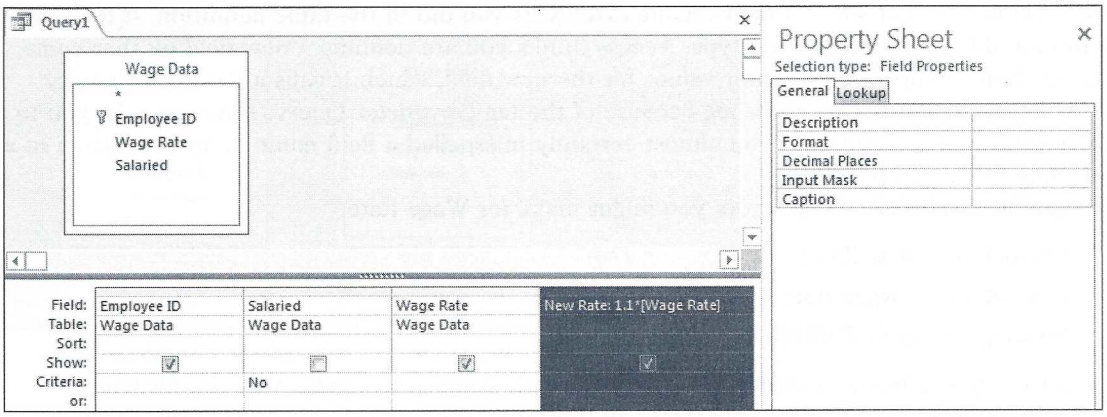
Notice that the calculated field output is not shown in Currency format, but as a Double—a number with digits after the decimal point. To convert the output to Currency format, select the output column by clicking the line above the calculated field expression. The column darkens to indicate its selection. Your data entry screen should resemble the one shown in Figure B-13.



Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-13 Activating a calculated field in query design

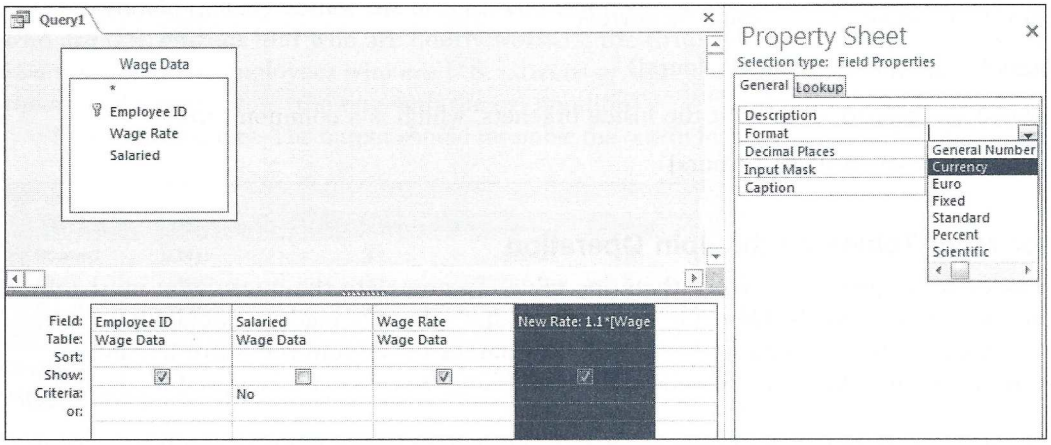
Then, on the Design tab, click Property Sheet in the Show/Hide group. The Field Properties sheet appears, as shown on the right in Figure B-14.



Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-14 Field properties of a calculated field

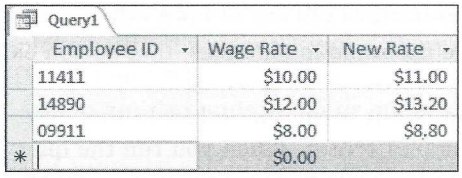
Click Format and choose Currency, as shown in Figure B-15. Then click the *X* in the upper-right corner of the window to close it.



Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-15 Currency format of a calculated field

When you run the query, the output should resemble the one in Figure B-16.



Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-16 Query output with formatted calculated field

Next, you examine how to avoid errors when making calculated fields.

Avoiding Errors when Making Calculated Fields

Follow these guidelines to avoid making errors in calculated fields:

- Do not enter the expression in the *Criteria* cell as if the field definition were a filter. You are making a field, so enter the expression in the *Field* cell.

- Spell, capitalize, and space a field's name *exactly* as you did in the table definition. If the table definition differs from what you type, Access thinks you are defining a new field by that name. Access then prompts you to enter values for the new field, which it calls a Parameter Query field. This problem is easy to debug because of the tag *Parameter Query*. If Access asks you to enter values for a parameter, you almost certainly misspelled a field name in an expression in a calculated field or criterion.
  For example, here are some errors you might make for Wage Rate:

  Misspelling: (Wag Rate)

  Case change: (wage Rate / WAGE RATE)

  Spacing change: (WageRate / Wage Rate)

- Do not use parentheses or curly braces instead of the square brackets. Also, do not put parentheses inside square brackets. You *can*, however, use parentheses outside the square brackets in the normal algebraic manner.
  For example, suppose that you want to multiply Hours by Wage Rate to get a field called Wages Owed. This is the correct expression:

  Wages Owed: [Wage Rate] * [Hours]

  The following expression also would be correct:

  Wages Owed: ([Wage Rate] * [Hours])

  But it would *not* be correct to omit the inside brackets, which is a common error:

  Wages Owed: [Wage Rate * Hours]

"Relating" Two or More Tables by the Join Operation

Often, the data you need for a query is in more than one table. To complete the query, you must **join** the tables by linking the common fields. One rule of thumb is that joins are made on fields that have common *values*, and those fields often can be key fields. The names of the join fields are irrelevant; also, the names of the tables or fields to be joined may be the same, but it is not required for an effective join.

Make a join by bringing in (adding) the tables needed. Next, decide which fields you will join. Then click one field name and hold down the left mouse button while you drag the cursor over to the other field's name in its window. Release the button. Access inserts a line to signify the join. (If a relationship between two tables has been formed elsewhere, Access inserts the line automatically, and you do not have to perform the click-and-drag operation. Access often inserts join lines without the user forming relationships.)

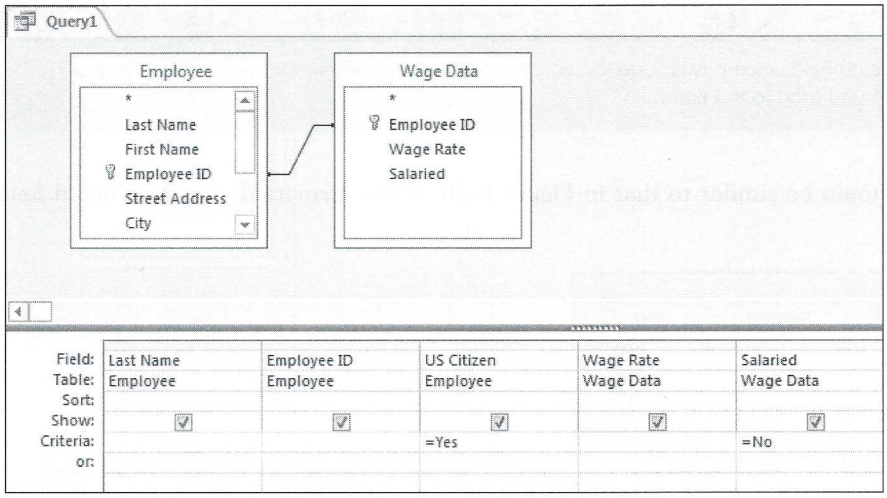
You can join more than two tables. The common fields *need not* be the same in all tables; that is, you can daisy-chain them together.

A common join error is to add a table to the query and then fail to link it to another table. In that case, you will have a table floating in the top part of the QBE (query by example) screen. When you run the query, your output will show the same records over and over. The error is unmistakable because there is *so much* redundant output. The two rules are to add only the tables you need and to link all tables.

Next, you will work through an example of a query that needs a join.

AT THE KEYBOARD

Suppose you want to see the last names, employee IDs, wage rates, salary status, and citizenship only for U.S. citizens and hourly workers. Because the data is spread across two tables, Employee and Wage Data, you should add both tables and pull down the five fields you need. Then you should add the Criteria expressions. Set up your work to resemble the one in Figure B-17. Make sure the tables are joined on the common field, Employee ID.

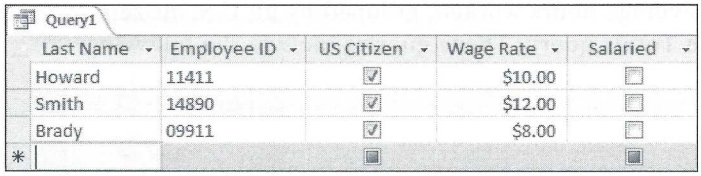


Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-17 A query based on two joined tables

You should quickly review the criteria you will need to set up this join: If you want data for employees who are U.S. citizens *and* who are hourly workers, the Criteria expressions go in the *same* Criteria row. If you want data for employees who are U.S. citizens *or* who are hourly workers, one of the expressions goes in the second Criteria row (the one with the or: notation).

Now run the query. The output should resemble the one in Figure B-18, with the exception of the name "Brady."



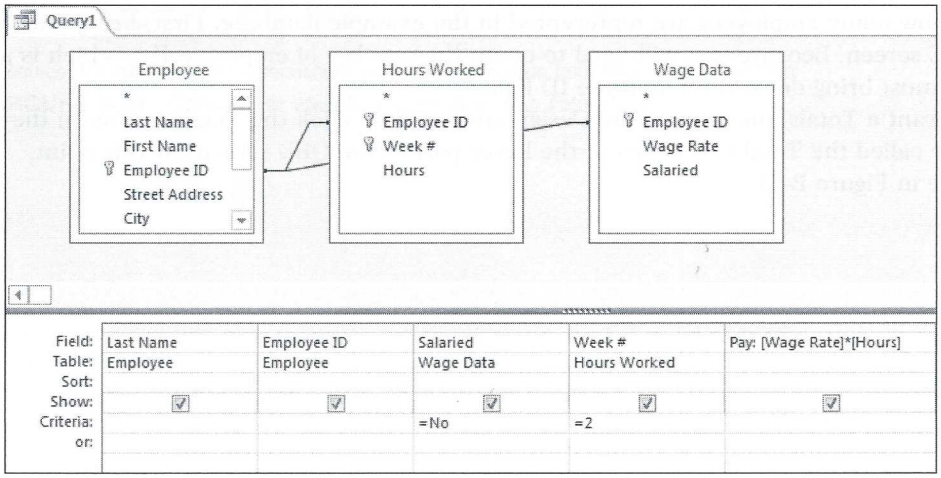
Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-18 Output of a query based on two joined tables

You do not need to print or save the query output, so return to Design view and close the query. Another practice query follows.

AT THE KEYBOARD

Suppose you want to see the wages owed to hourly employees for Week 2. You should show the last name, the employee ID, the salaried status, the week #, and the wages owed. Wages will have to be a calculated field ([Wage Rate] * [Hours]). The criteria are No for Salaried and 2 for the Week #. (This means that another "And" query is required.) Your query should be set up like the one in Figure B-19.



Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-19 Query setup for wages owed to hourly employees for Week 2

NOTE

In the query in Figure B-19, the calculated field column was widened so you could see the whole expression. To widen a column, click the column boundary line and drag to the right.

Run the query. The output should be similar to that in Figure B-20, if you formatted your calculated field to Currency.

Query1

Last Name	Employee ID	Salaried	Week #	Pay
Howard	11411	☐	2	$500.00
Smith	14890	☐	2	$480.00
Brady	09911	☐	2	$440.00
*		☐		

Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-20 Query output for wages owed to hourly employees for Week 2

Notice that it was not necessary to pull down the Wage Rate and Hours fields to make the query work. You do not need to save or print the query output, so return to Design view and close the query.

Summarizing Data from Multiple Records (Totals Queries)

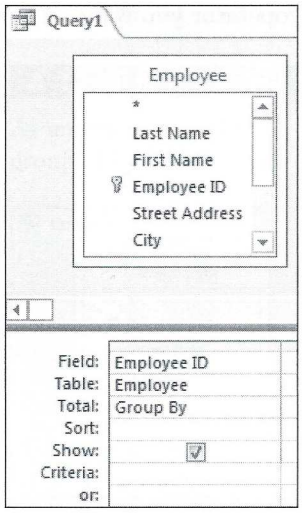
You may want data that summarizes values from a field for several records (or possibly all records) in a table. For example, you might want to know the average hours that all employees worked in a week or the total (sum) of all of the hours worked. Furthermore, you might want data grouped or stratified in some way. For example, you might want to know the average hours worked, grouped by all U.S. citizens versus all non-U.S. citizens. Access calls such a query a **Totals query**. These queries include the following operations:

Sum	The total of a given field's values
Count	A count of the number of instances in a field—that is, the number of records. In the current example, you would count the number of employee IDs to get the number of employees.
Average	The average of a given field's values
Min	The minimum of a given field's values
Var	The variance of a given field's values
StDev	The standard deviation of a given field's values
Where	The field has criteria for the query output

AT THE KEYBOARD

Suppose you want to know how many employees are represented in the example database. First, bring the Employee table into the QBE screen. Because you will need to count the number of employee IDs, which is a Totals query operation, you must bring down the Employee ID field.

To tell Access that you want a Totals query, click the Design tab and then click the Totals button in the Show/Hide group. A new row called the Total row opens in the lower part of the QBE screen. At this point, the screen resembles the one in Figure B-21.

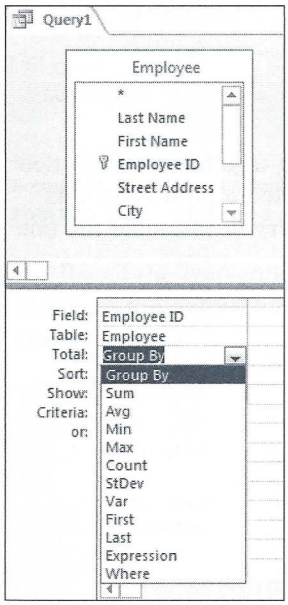


Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-21 Totals query setup

Note that the Total cell contains the words *Group By*. Until you specify a statistical operation, Access assumes that a field will be used for grouping (stratifying) data.

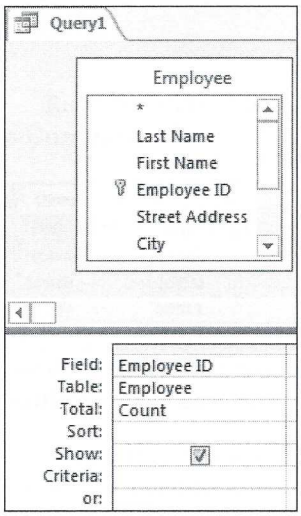
To count the number of employee IDs, click next to Group By to display an arrow. Click the arrow to reveal a drop-down menu, as shown in Figure B-22.



Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-22 Choices for statistical operation in a Totals query

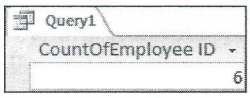
Select the Count operator. (You might need to scroll down the menu to see the operator you want.) Your screen should resemble the one shown in Figure B-23.



Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-23 Count in a Totals query

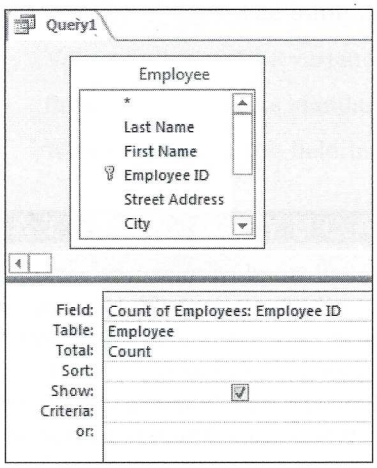
Run the query. Your output should resemble the one in Figure B-24.



Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-24 Output of Count in a Totals query

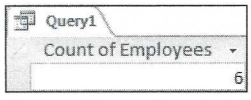
Notice that Access created a pseudo-heading, "CountOfEmployee ID," by splicing together the statistical operation (Count), the word Of, and the name of the field (Employee ID). If you wanted a phrase such as "Count of Employees" as a heading, you would go to Design view and change the query to resemble the one shown in Figure B-25.



Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-25 Heading change in a Totals query

When you run the query, the output should resemble the one in Figure B-26.



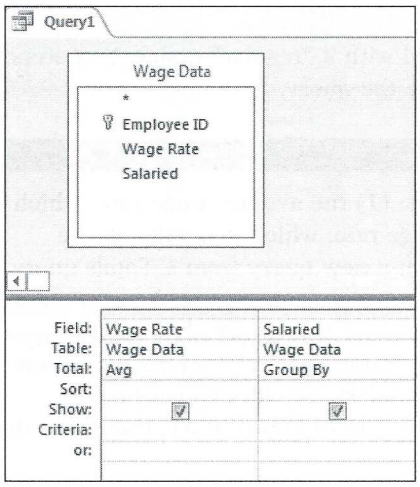
Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-26 Output of heading change in a Totals query

You do not need to print or save the query output, so return to Design view and close the query.

AT THE KEYBOARD

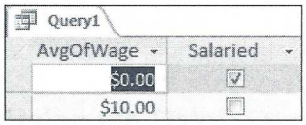
As another example of a Totals query, suppose you want to know the average wage rate of employees, grouped by whether the employees are salaried. Figure B-27 shows how to set up your query.



Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-27 Query setup for average wage rate of employees

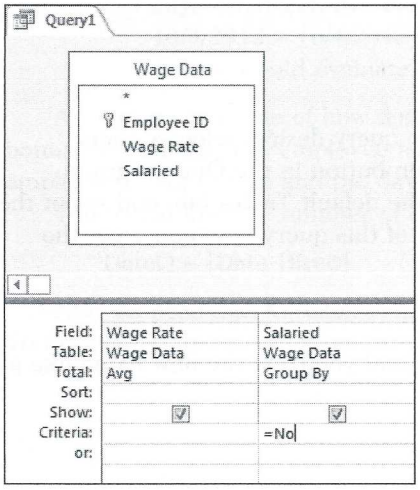
When you run the query, your output should resemble the one in Figure B-28.



Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-28 Output of query for average wage rate of employees

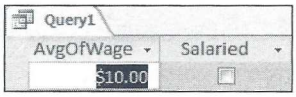
Recall the convention that salaried workers are assigned zero dollars an hour. Suppose you want to eliminate the output line for zero dollars an hour because only hourly-rate workers matter for the query. The query setup is shown in Figure B-29.



Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-29 Query setup for nonsalaried workers only

When you run the query, you will get output for nonsalaried employees only, as shown in Figure B-30.



Source: Microsoft product screenshots used with permission from Microsoft Corporation.

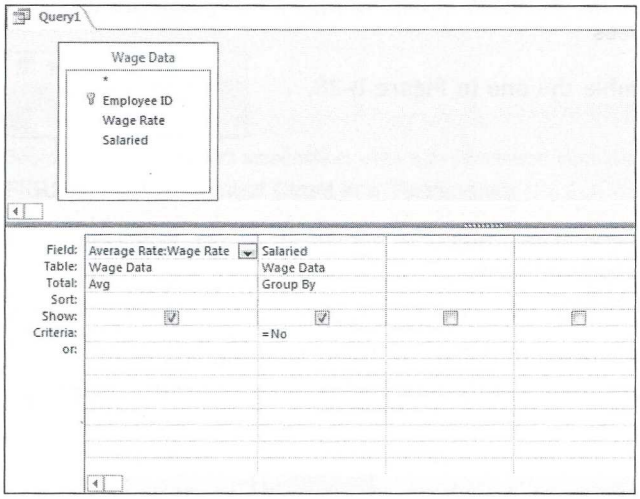
FIGURE B-30 Query output for nonsalaried workers only

Thus, it is possible to use Criteria in a Totals query, just as you would with a "regular" query. You do not need to print or save the query output, so return to Design view and close the query.

AT THE KEYBOARD

Assume that you want to see two pieces of information for hourly workers: (1) the average wage rate, which you will call Average Rate in the output, and (2) 110 percent of the average rate, which you will call the Increased Rate. To get this information, you can make a calculated field in a new query from a Totals query. In other words, you use one query as a basis for another query.

Create the first query; you already know how to perform certain tasks for this query. The revised heading for the average rate will be Average Rate, so type *Average Rate: Wage Rate* in the Field cell. Note that you want the average of this field. Also, the grouping will be by the Salaried field. (To get hourly workers only, enter *Criteria: No*.) Confirm that your query resembles the one in Figure B-31, then save the query and close it.



Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-31 A totals query with average

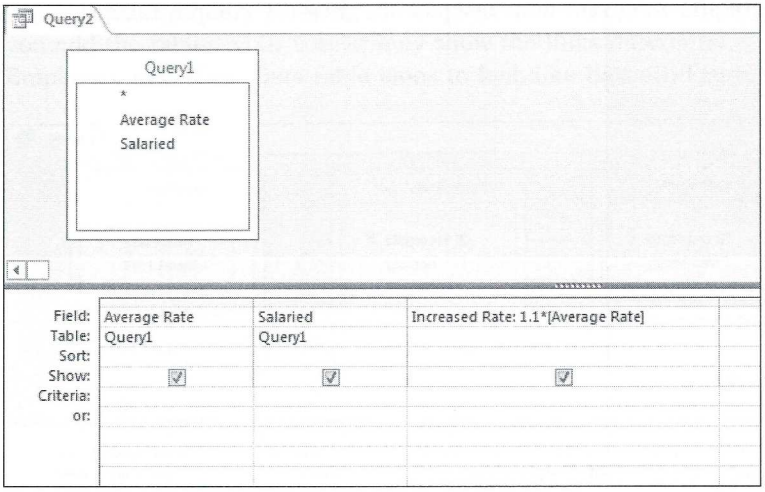
Now begin a new query. However, instead of bringing in a table to the query design, select a query. To start a new query, click the Create tab and then click the Query Design button in the Queries group. The Show Table window appears. Click the Queries tab instead of using the default Tables tab, and select the query you just saved as a basis for the new query. The most difficult part of this query is to construct the expression for the calculated field. Conceptually, it is as follows:

Increased Rate: 1.1 * [The current average]

You use the new field name in the new query as the current average, and you treat the new name like a new field:

Increased Rate: 1.1 * [Average Rate]

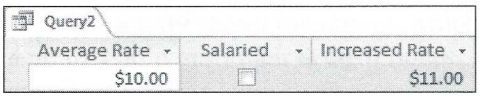
The query within a query is shown in Figure B-32.



Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-32 A query within a query

Figure B-33 shows the output of the new query. Note that the calculated field is formatted.



Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-33 Output of an Expression in a Totals query

You do not need to print or save the query output, so return to Design view and close the query.

Using the Date() Function in Queries

Access has two important date function features:

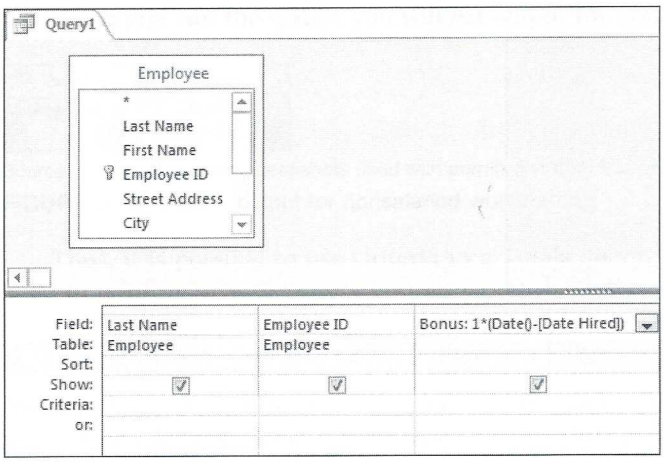
- The built-in Date() function gives you today's date. You can use the function in query criteria or in a calculated field. The function "returns" the day on which the query is run; in other words, it inserts the value where the Date() function appears in an expression.
- Date arithmetic lets you subtract one date from another to obtain the difference—in number of days—between two calendar dates. For example, suppose you create the following expression:
  10/9/2012 – 10/4/2012
  Access would evaluate the expression as the integer 5 (9 minus 4 is 5).

As another example of how date arithmetic works, suppose you want to give each employee a one-dollar bonus for each day the employee has worked. You would need to calculate the number of days between the employee's date of hire and the day the query is run, and then multiply that number by $1.

You would find the number of elapsed days by using the following equation:

Date() – [Date Hired]

Also suppose that for each employee, you want to see the last name, employee ID, and bonus amount. You would set up the query as shown in Figure B-34.



Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-34 Date arithmetic in a query

Assume that you set the format of the Bonus field to Currency. The output will be similar to that in Figure B-35, although your Bonus data will be different because you used a different date.

Query1

Last Name	Employee ID	Bonus
Brady	09911	$0.00
Howard	11411	$42.00
Johnson	12345	$6,981.00
Smith	14890	$10,225.00
Jones	22282	$4,015.00
Ruth	71460	$5,811.00

Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-35 Output of query with date arithmetic

Using Time Arithmetic in Queries

Access also allows you to subtract the values of time fields to get an elapsed time. Assume that your database has a Job Assignments table showing the times that nonsalaried employees were at work during a day. The definition is shown in Figure B-36.

Table1

Field Name	Data Type
Employee ID	Short Text
ClockIn	Date/Time
ClockOut	Date/Time
DateWorked	Date/Time

Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-36 Date/Time data definition in the Job Assignments table

Assume that the DateWorked field is formatted for Long Date and that the ClockIn and ClockOut fields are formatted for Medium Time. Also assume that for a particular day, nonsalaried workers were scheduled as shown in Figure B-37.

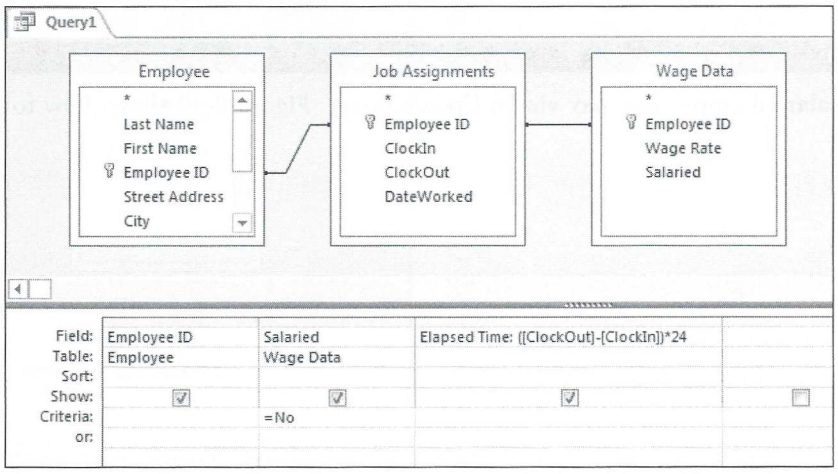
Job Assignments

Employee ID	ClockIn	ClockOut	DateWorked	Click to Add
09911	8:30 AM	4:30 PM	Thursday, September 29, 2016	
11411	9:00 AM	3:00 PM	Thursday, September 29, 2016	
14890	7:00 AM	5:00 PM	Thursday, September 29, 2016	

Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-37 Display of date and time in a table

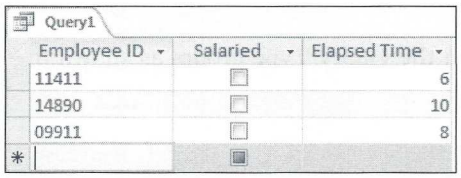
You want a query showing the elapsed time that your employees were on the premises for the day. When you add the tables, your screen may show the links differently. Click and drag the Job Assignments, Employee, and Wage Data table icons to look like those in Figure B-38.



Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-38 Query setup for time arithmetic

Figure B-39 shows the output, which looks correct. For example, employee 09911 was at work from 8:30 a.m. to 4:30 p.m., which is eight hours. But how does the odd expression that follows yield the correct answers?



Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-39 Query output for time arithmetic

([ClockOut] – [ClockIn]) * 24

Why wouldn't the following expression work?

[ClockOut] – [ClockIn]

Here is the answer: In Access, subtracting one time from the other yields the *decimal* portion of a 24-hour day. Returning to the example, you can see that employee 09911 worked eight hours, which is one-third of a day, so the time arithmetic function yields .3333. That is why you must multiply by 24—to convert from decimals to an hourly basis. Hence, for employee 09911, the expression performs the following calculation: 1/3 × 24 = 8.

Note that parentheses are needed to force Access to do the subtraction *first*, before the multiplication. Without parentheses, multiplication takes precedence over subtraction. For example, consider the following expression:

[ClockOut] – [ClockIn] * 24

In this example, ClockIn would be multiplied by 24, the resulting value would be subtracted from ClockOut, and the output would be a nonsensical decimal number.

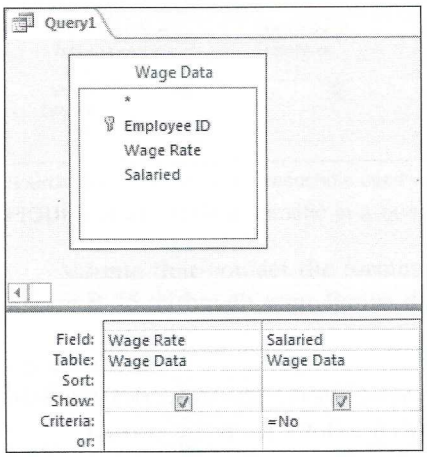
Deleting and Updating Queries

The queries presented in this tutorial so far have been Select queries. They select certain data from specific tables based on a given criterion. You also can create queries to update the original data in a database. Businesses use such queries often, and in real time. For example, when you order an item from a Web site, the company's database is updated to reflect your purchase through the deletion of that item from the company's inventory.

Consider an example. Suppose you want to give all nonsalaried workers a $0.50 per hour pay raise. Because you have only three nonsalaried workers, it would be easy to change the Wage Rate data in the table. However, if you had 3,000 nonsalaried employees, it would be much faster and more accurate to change the Wage Rate data by using an Update query that adds $0.50 to each nonsalaried employee's wage rate.

AT THE KEYBOARD

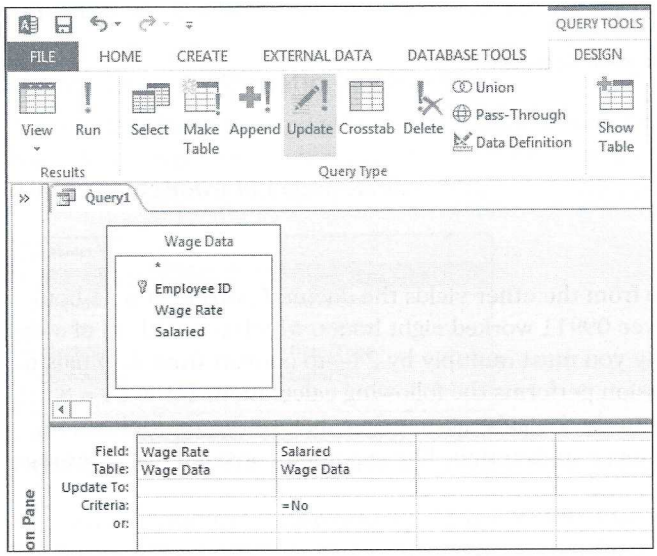
Now you will change each of the nonsalaried employees' pay via an Update query. Figure B-40 shows how to set up the query.



Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-40 Query setup for an Update query

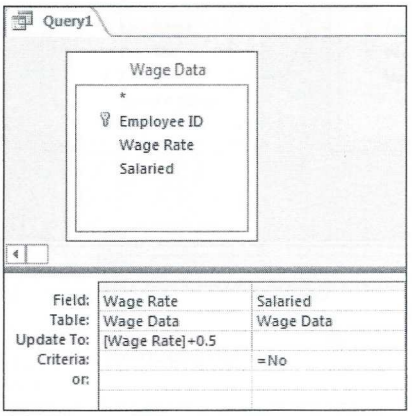
So far, this query is just a Select query. Click the Update button in the Query Type group, as shown in Figure B-41.



Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-41 Selecting a query type

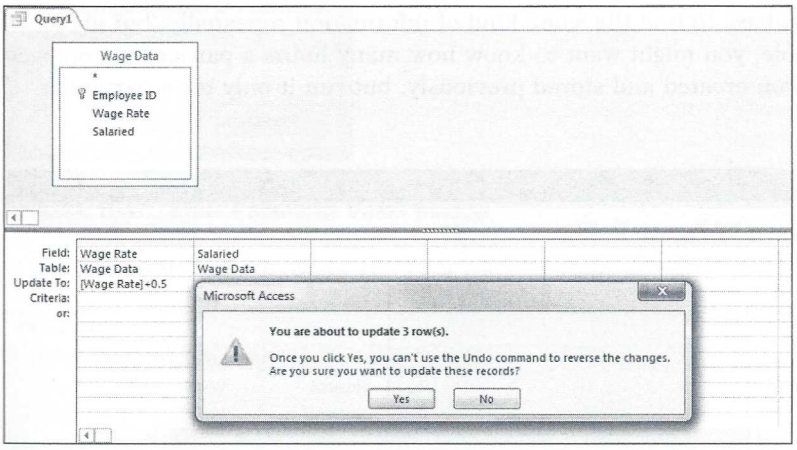
Notice that you now have another line on the QBE grid called Update To:, which is where you specify the change or update the data. Notice that you will update only the nonsalaried workers by using a filter under the Salaried field. Update the Wage Rate data to Wage Rate plus $0.50, as shown in Figure B-42. Note that the update involves the use of brackets [ ], as in a calculated field.



Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-42 Updating the wage rate for nonsalaried workers

Now run the query by clicking the Run button in the Results group. If you cannot run the query because it is blocked by Disabled Mode, click the Enable Content button on the Security Warning message bar. When you successfully run the query, the warning message in Figure B-43 appears.



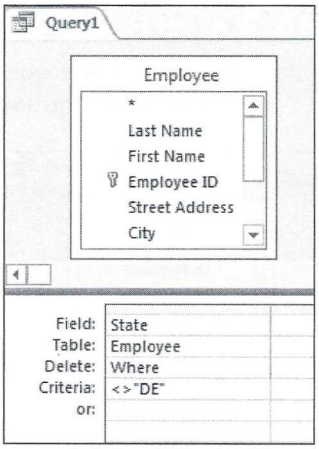
Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-43 Update query warning

When you click Yes, the records are updated. Check the updated records by viewing the Wage Data table. Each nonsalaried wage rate should be increased by $0.50. You could add or subtract data from another table as well. If you do, remember to put the field name in square brackets.

Another type of query is the Delete query, which works like Update queries. For example, assume that your company has been purchased by the state of Delaware, which has a policy of employing only state residents. Thus, you must delete (or fire) all employees who are not exclusively Delaware residents.

To do that, you would create a Select query. Using the Employee table, you would click the Delete button in the Query Type group, then bring down the State field and filter only those records that were not in Delaware (DE). Do not perform the operation, but note that if you did, the setup would look like the one in Figure B-44.



Source: Microsoft product screenshots used with permission from Microsoft Corporation.

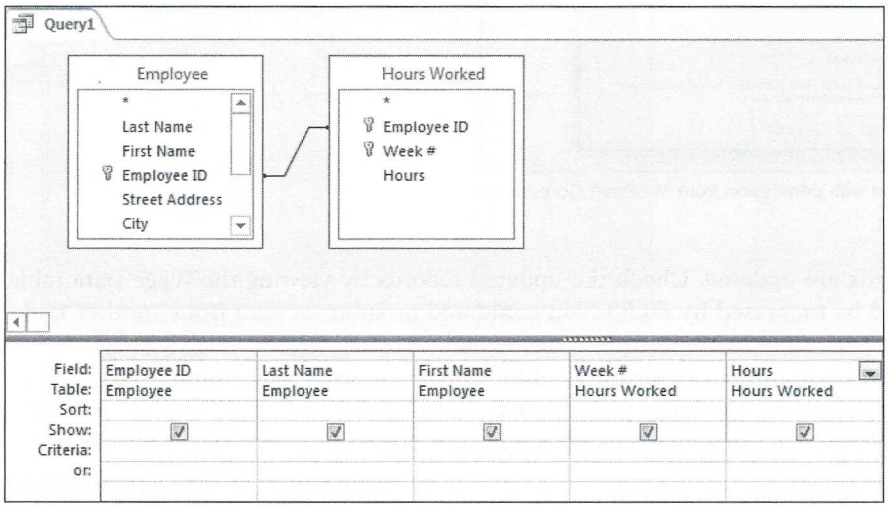
FIGURE B-44 Deleting all employees who are not Delaware residents

Using Parameter Queries

A **Parameter query** is actually a type of Select query. For example, suppose your company has 5,000 employees and you want to query the database to find the same kind of information repeatedly, but about different employees each time. For example, you might want to know how many hours a particular employee has worked. You could run a query that you created and stored previously, but run it only for a particular employee.

AT THE KEYBOARD

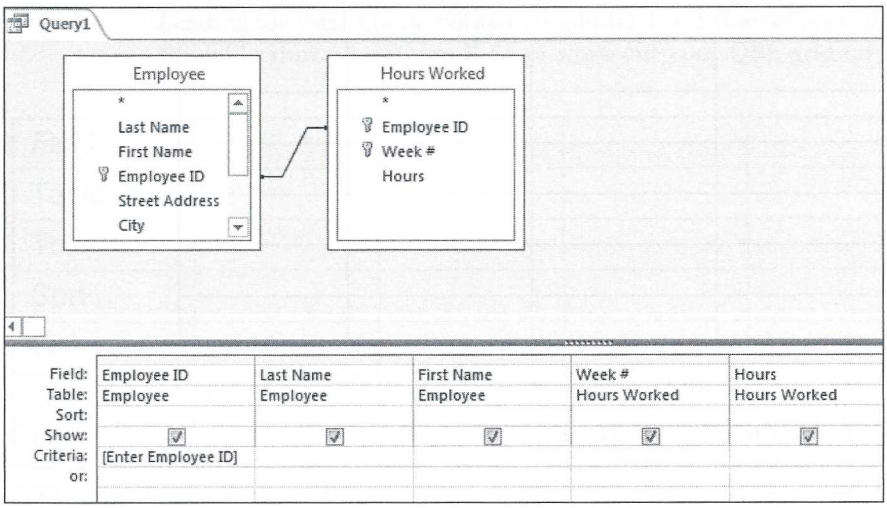
Create a Select query with the format shown in Figure B-45.



Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-45 Design of a Parameter query beginning as a Select query

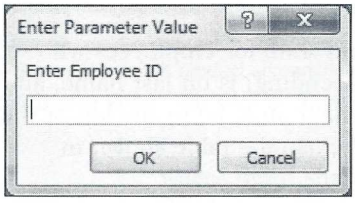
In the Criteria line of the QBE grid for the Employee ID field, type what is shown in Figure B-46.



Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-46 Design of a Parameter query, continued

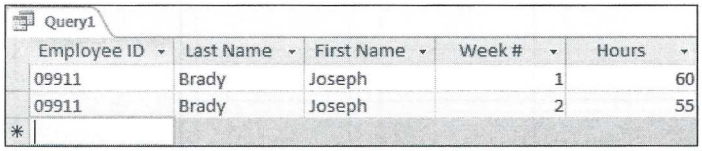
Note that the Criteria line uses square brackets, as you would expect to see in a calculated field. Now run the query. You will be prompted for the employee's ID number, as shown in Figure B-47.



Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-47 Enter Parameter Value window

Enter your own employee ID. Your query output should resemble the one in Figure B-48.



Source: Microsoft product screenshots used with permission from Microsoft Corporation.

FIGURE B-48 Output of a Parameter query

MAKING SEVEN PRACTICE QUERIES

This portion of the tutorial gives you additional practice in creating queries. Before making these queries, you must create the specified tables and enter the records shown in the "Creating Tables" section of this tutorial. The output shown for the practice queries is based on those inputs.

AT THE KEYBOARD

For each query that follows, you are given a problem statement and a "scratch area." You also are shown what the query output should look like. Set up each query in Access and then run the query. When you are satisfied with the results, save the query and continue with the next one. Note that you will work with the Employee, Hours Worked, and Wage Data tables.

1. Create a query that shows the employee ID, last name, state, and date hired for employees who live in Delaware *and* were hired after 12/31/99. Perform an ascending sort by employee ID. First