MICROSOFT ACCESS TUTORIAL

Microsoft Access is a relational database package that runs on the Microsoft Windows operating system. This tutorial was prepared using Access 2007.

Before using this tutorial, you should be familiar with the fundamentals of Microsoft Access and know how to use Windows. This tutorial teaches you some of the advanced Access skills you’ll need to do database case studies. The tutorial concludes with a discussion of common Access problems and explains how to solve them.

A preliminary word of caution: always observe proper file-saving and closing procedures. Use these steps to exit Access: (1) Office Button—Close Database, then (2) X-Exit Access Button. Or you may simply click the X-Exit Access Button, which gets you back to Windows. Always end your work with those file-closing steps. Do not remove your disk, CD, or other portable storage device such as a USB flash drive when database forms, tables, etc., appear on the screen; you will lose your work.

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

AT THE KEYBOARD

Open a new database. On the Getting Started with Microsoft Office Access page, under the heading New Blank Database, click Blank Database. Call the database Employee. Click the file folder next to the filename to browse for the desired folder. Otherwise, your file will be saved automatically in My Documents.

Your opening screen should resemble the screen shown in Figure B-1.

![Figure B-1](image)

FIGURE B-1 Entering data in the Datasheet view

When you create a table, Access opens the table in the Datasheet view by default. Because you will be using the 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 Table I. 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.

![Figure B-2](image)

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 instructions that follow.
(1) Define a table called EMPLOYEE.

This table contains permanent data about employees. To create it, choose the Create tab and in the Tables group, click **Table design**. The table's fields are Last Name, First Name, SSN (Social Security Number), Street Address, City, State, Zip, Date Hired, and US Citizen. The field SSN is the primary key field. Change the length of text fields from the default 255 spaces to more appropriate lengths; for example, the field Last Name might be 30 spaces and the Zip field might be 10 spaces. Your completed definition should resemble the one shown in Figure B-3.

![Table design](image)

**FIGURE B-3** Fields in the EMPLOYEE table

When you're finished, choose Office Button—Save; then enter the name desired for the table (here, EMPLOYEE). Make sure you specify the name of the table, not the database itself. (Here, it is a coincidence that the EMPLOYEE table has the same name as its database file.) Close the table by clicking the x-Close corresponding to the table EMPLOYEE.

(2) Define a table called WAGE DATA.

This table contains permanent data about employees and their wage rates. The table’s fields are SSN, Wage Rate, and Salaried. The field SSN 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.

![Table design](image)

**FIGURE B-4** Fields in the WAGE DATA table

Use Office Button—Save to save the table definition. Name the table WAGE DATA.

(3) Define a table called HOURS WORKED.

The purpose of this table is to record the number of hours employees work each week during the year. The table's fields are SSN (text), Week # (number—long integer), and Hours (number—double). The SSN and Week # are the compound keys.

In the following example, the employee with SSN 089-65-9000 worked 40 hours in Week 1 of the year and 52 hours in Week 2.

<table>
<thead>
<tr>
<th>SSN</th>
<th>Week</th>
<th>#Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>089-65-9000</td>
<td>1</td>
<td>40</td>
</tr>
<tr>
<td>089-65-9000</td>
<td>2</td>
<td>52</td>
</tr>
</tbody>
</table>

Note that no single field can be the primary key field. Why? The reason is that 089-65-9000 is an entry for each week. In other words, if this employee works each week of the year, at the end of the year, 52 records
will have the SSN value. Thus, SSN values will not distinguish records. In addition, 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—corresponding to a full-time workweek—would be a common entry for many weeks.

However, that presents a problem because in Access, a table must have a primary key field. What is the solution? 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 field SSN and the Week # field. Why? Because as each person works each week, the week passes. That means, for example, that there is only one combination of SSN 089-65-9000 and Week # 1. Because those values can occur in only one record, the combination distinguishes that record from all others.

How do you set a compound key? The first step 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**

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

![Figure B-5](image)

**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.

![Figure B-6](image)

**FIGURE B-6** The compound primary key for the HOURS WORKED table

That completes the creation of the compound primary key and the table definition. Use Office Button—Save 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 were to print the tables, all you would see would be 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 along the left side of the screen and type the data directly into the cells. (Note: Access 2007 uses a Navigation Pane to display and open the database objects. The Navigation Pane is on the left side of the Access window.)

**AT THE KEYBOARD**

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

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 data is entered.) Figure B-7 shows all of the fields on the screen.

Tutorial B
FIGURE B-7  The data entry screen for the EMPLOYEE table

Type in your data one field value at a time. Note that the first row is empty when you begin. Each time you finish a value, press Enter; the cursor will move to the next cell. After data has been entered 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, there is no need to perform the Office Button—Save step after entering data into a table.)

When entering data in your table, note that dates (for example, in the Date Hired field) should be entered as follows: 6/15/07. 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, as if you were using a toggle switch.

If you make errors in data entry, click the cell, backspace over the error, and type the correction.

Enter the data shown in Figure B-8 into the EMPLOYEE table.

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 Social Security number. (For purposes of this tutorial, this sixth record, as you’ll discover as you continue reading, has been created using the name of one of this text’s authors and the SSN 099-11-3344.)

After adding records to the EMPLOYEE table, open the WAGE DATA table and enter the data shown in Figure B-9.

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 SSN. Also assume that you earn $8 an hour and are not salaried. (Note that when an employee’s Salaried box is not checked (i.e., Salaried = No), the implication is that the employee is paid by the hour. Because employees who are salaried do not get paid by the hour, their hourly rate is shown as 0.00.)

Once you have finished creating the WAGE DATA table, open the HOURS WORKED table and enter the data shown in Figure B-10.

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 SSN, 60 hours worked for Week 1, and 55 hours worked for Week 2.
FIGURE B-10  Data for the HOURS WORKED table

CREATING QUERIES

Because you know how to create basic queries, this section teaches you the kinds of advanced queries you will create in the Case Studies.

Using Calculated Fields in Queries

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

AT THE KEYBOARD

Suppose you want to see the SSNs and wage rates of hourly workers and you want to see what the wage rates would be if all employees were given a 10 percent raise. To view that information, show the SSN, 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.

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 the Criteria = No. Note in Figure B-11 that the Show box for that field is not checked, so the Salaried field values will not show in the query output.

Note the expression for the calculated field, which you can see in the rightmost field cell:
New Rate: 1.1 \* [Wage Rate]
The term New Rate merely specifies the desired output heading. (Don't forget the colon.) The 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. That is a rule. Any time an Access expression refers to a field name, the expression must be enclosed in square brackets.

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

<table>
<thead>
<tr>
<th>SSN</th>
<th>Wage Rate</th>
<th>New Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>114-11-3333</td>
<td>$10.00</td>
<td>11.10</td>
</tr>
<tr>
<td>148-90-1234</td>
<td>$12.00</td>
<td>13.20</td>
</tr>
<tr>
<td>089-11-3344</td>
<td>$8.00</td>
<td>8.80</td>
</tr>
</tbody>
</table>

**FIGURE B-12** Output for a query with calculated field

Notice that the calculated field output is not shown in Currency format; it's shown as a Double—a number with digits after the decimal point. To convert the output to Currency format, you should select the output column by clicking the line above the calculated field expression, thus activating the column, which subsequently darkens. Your data entry screen should resemble the one shown in Figure B-13.

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

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

Now when you run the query, the output should resemble that shown in Figure B-16.

Next, you'll look at how to avoid errors when making calculated fields.
Avoiding Errors When Making Calculated Fields

Follow these guidelines to avoid making errors in calculated fields:

- Don’t put the expression in the Criteria cell as if the field definition were a filter. You are making a field, so put 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’re defining a new field by that name. Access then prompts you to enter values for the new field, which calls a Parameter Query field. That 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.

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)

- Don’t use parentheses or curly braces instead of the square brackets. Also, don’t put parentheses inside square brackets. You are, however, allowed to use parentheses outside the square brackets, in the normal algebraic manner.
Example: Suppose you want to multiply Hours times Wage Rate to get a field called Wages Owed. This is the correct expression:

Wages Owed: \([\text{Wage Rate}] \times [\text{Hours}]\)

The following also would be correct:

Wages Owed: \([\text{Wage Rate}] \times ([\text{Hours}])\)

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

Wages Owed: \([\text{Wage Rate} \times \text{Hours}]\)

"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, which is known as a join. 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 that is not a requirement 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 puts in a line, signifying the join. (Note: If a relationship between two tables has been formed elsewhere, Access will put in the line automatically, in which case you do not have to perform the click-and-drag operation. Often Access puts in 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. That error is unmistakable because there is so much redundant output. The rules are (1) add only the tables you need and (2) link all tables.

Next, you'll work through an example of a query needing a join.

**AT THE KEYBOARD**

Suppose you want to see the last names, SSNs, 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. Set up your work to resemble that shown in Figure B-17. Make sure the tables are joined on the common field, SSN.

![Figure B-17](image-url) A query based on two joined tables
Here is a quick review of 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 that has the or: notation).

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

![Figure B-18](output of a query based on two joined tables)

As there is no need to print the query output or save it, you should go back to the 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. To do that, you should show the last name, the SSN, 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 #. (Note: This means another “And” query is required.) Your query should be set up like the query displayed in Figure B-19.

![Figure B-19](query setup for wages owed to hourly employees for Week 2)

### NOTE

Note that in the query in Figure B-19, the calculated field column was widened so you can 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 shown in Figure B-20 (if you formatted your calculated field to Currency).

![Table showing query output for wages owed to hourly employees for Week 2](image)

**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. As there is no need to print the query output or save it, you should go back to the 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 perhaps the total (sum) of all of the hours worked. Furthermore, you might want data grouped (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 that kind of query a **Totals query**. Those operations include the following:

- **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, to get the number of employees, you’d count the number of SSNs.)
- **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. The first step is to bring the EMPLOYEE table into the QBE screen. Do that now. Because you will need to count the number of SSNs, which is a Totals query operation, you must bring down the SSN field.

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

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

To count the number of SSNs, click next to Group By, which reveals a little arrow. Click the arrow to reveal a drop-down menu, as shown in Figure B-22.

Select the Count operator. (With this menu, you may need to scroll to see the operator you want.) Your screen should now resemble the one shown in Figure B-23.

Run the query. Your output should resemble that shown in Figure B-24.
FIGURE B-21  Totals query setup

FIGURE B-22  Choices for statistical operation in a Totals query
FIGURE B-23  Count in a Totals query

FIGURE B-24  Output of Count in a Totals query

Notice that Access made a pseudo-heading “CountOfSSN.” To do that, Access spliced together the statistical operation (Count), the word Of, and the name of the field (SSN). What if you wanted a phrase such as Count of Employees as a heading? In the Design View, you’d change the query to resemble the one shown in Figure B-25.

FIGURE B-25  Heading change in a Totals query

Now when you run the query, the output should resemble that shown in Figure B-26.

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

As there is no need to print the query output or save it, you should go back to the Design View and close the query.
Here is 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 your query should be set up.

![Query setup for average wage rate of employees](image)

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

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

![Output of query for average wage rate of employees](image)

**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 that query. The query setup is shown in Figure B-29.

![Query setup for nonsalaried workers only](image)

**FIGURE B-29** Query setup for nonsalaried workers only

Now when you run the query, you'll get output for nonsalaried employees only, as shown in Figure B-30.
Thus, it’s possible to use Criteria in a Totals query, just as you would with a “regular” query. As there is no need to print the query output or save it, you should go back to the Design View and close the query.

**AT THE KEYBOARD**

Assume you want to see two pieces of information for hourly workers: (1) the average wage rate—call it Average Rate in the output and (2) 110 percent of the average rate—call it the Increased Rate. To do that, you can make a calculated field in a Totals query.

You already know how to do certain things for this query. The revised heading for the average rate will be Average Rate (type **Average Rate: Wage Rate** in the Field cell). Note that you want the average of this field. Also, the grouping would be by the Salaried field. (To get hourly workers only, enter **Criteria: No.**)

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, however that is denoted]**

The question is how to represent [The current average]. You cannot use Wage Rate, because that heading denotes the wages before they are averaged. Surprisingly, you can use the new heading (Average Rate) to denote the averaged amount as follows:

**Increased Rate: 1.1 * [Average Rate]**

Although it may seem counterintuitive, you can treat **Average Rate as if it were an actual field name**. Note, however, that if you use a calculated field such as **Average Rate** in another calculated field, as shown in Figure B-31, you must show that original calculated field in the query output. If you don’t, the query will ask you to enter parameter value, which is incorrect. Use the setup shown in Figure B-31.

![Figure B-31](image)

**FIGURE B-31** Using a calculated field in another calculated field

**NOTE**

If you run the query now shown in Figure B-31, you’d get some sort of error message because there is no statistical operator applied to the calculated field’s total cell, instead, the words **Group By** appear there. To correct that, you must change the **Group By** operator to **Expression**. You may have to scroll down to get to **Expression** in the list.

Figure B-32 shows how the screen looks before the query is run. Figure B-33 shows the output of the query.
As there is no need to print the query output or save it, you should go back to the Design View and close the query.

**Using the Date() Function in Queries**

Access has two date function features that are important for you to know. A description of each follows.

1. The following built-in function gives you today's date:
   
   ```
   Date()
   ```

   You can use that function in query criteria or in a calculated field. The function “returns” the day on which the query is run; that is, it puts the value into the place where the date() function appears in an expression.

2. *Date arithmetic* lets you subtract one date from another to obtain the difference—in terms of number of days—between two calendar dates. For example, suppose you create the following expression:

   ```
   ```

   Access would evaluate that as the integer 5 (9 less 4 is 5).

   Here is another example of how date arithmetic works. Suppose you want to give each employee a bonus equaling a dollar for each day the employee has worked for you. You’d need to calculate the number of days between the employee’s date of hire and the day the query is run, 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, SSN, and bonus amount. You’d set up the query as shown in Figure B-34.

   Assume you had set the format of the Bonus field to Currency. The output will be similar to that shown in Figure B-35. (Your Bonus data will be different because you are working on a date that is different from the date this tutorial was written.)

**Using Time Arithmetic in Queries**

Access also allows you to subtract the values of time fields to get an elapsed time. Assume your database has a JOB ASSIGNMENTS table that shows the times that nonsalaried employees were at work during a day. The definition is shown in Figure B-36.
FIGURE B-34 Date arithmetic in a query

FIGURE B-35 Output of query with date arithmetic

FIGURE B-36 Date/Time data definition in the JOB ASSIGNMENTS table

Assume the Date field is formatted for Long Date and 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.

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

You want a query that will show 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.

Figure B-39 shows the output, which looks correct. For example, employee 099-11-3344 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?

\((\text{ClockOut} - \text{ClockIn}) \times 24\)
Why wouldn't the following expression work?

\[[\text{ClockOut}] - [\text{ClockIn}]\]

Here is the answer: In Access, subtracting one time from the other yields the \textit{decimal} portion of a 24-hour day. Returning to the example, you can see that employee 099-11-3344 worked eight hours, which is one-third of a day, so the time arithmetic function yields \(0.3333\). That is why you must multiply by 24—to convert from decimals to an hourly basis. Hence, for employee 099-11-3344, the expression performs the following calculation: \(1/3 \times 24 = 8\).

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

\[[\text{ClockOut}] - [\text{ClockIn}] ^ 24\]

In that case, 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 thus 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 do that often, and they do it in real time. For example, when you order an item from a Web site, the company's database is updated to reflect the purchase of the item through the deletion of that item from the company's inventory.

Consider an example. Suppose you want to give all of the nonsalaried workers a \$0.50 per hour pay raise. With the three nonsalaried workers you have now, it would be easy to go into the table and simply change the \textit{Wage Rate} data. But assume you have 3,000 nonsalaried employees. Now it would be much faster—not to mention more accurate—to change the \textit{Wage Rate} data for each of the 3,000 nonsalaried employees by using an Update query that adds \$0.50 to each employee's wage rate.
Now you will change each of the nonsalaried employees' pay via an Update query. Figure B-40 shows how to set up the query.

![Query setup for an Update query](image)

**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.

![Selecting a query type](image)

**FIGURE B-41** Selecting a query type

Notice that you now have another line on the QBE grid called *Update to*: That is where you specify the change or update to 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.)

Now run the query by clicking the Run button in the Results group. If you are unable to run the query because it's blocked by Disabled Mode, you need to choose Database Tools tab, Message Bar in the Show/Hide group. Click the Options button and choose “enable this content”, then OK. When you successfully run the query, you will get a warning message as shown in Figure B-43.
FIGURE B-42  Updating the wage rate for nonsalaried workers

FIGURE B-43  Update Query warning

Once you click Yes, the records will be updated. Check those updated records now by viewing the WAGE DATA table. Each salaried wage rate should be increased by $0.50. Note that in this example, you are simply adding $0.50 to each salaried wage rate. You could add or subtract data from another table as well. If you do, remember to put the field name in square brackets.

Another kind of query is the Delete query. Delete queries work like Update queries. For example, assume your company has been taken over by the state of Delaware. The state has a policy of employing only Delaware residents. Thus, you must delete (or fire) all employees who are not exclusively Delaware residents. To do that, you need to create a Select query. Using the EMPLOYEE table, you would choose the Delete icon from the Query Type group, then bring down the State field and filter only those records 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.

Using Parameter Queries
Another kind of query, which is actually a type of Select query, is a Parameter query. Here is an example: Suppose your company has 5,000 employees and you want to query the database to find the same kind of information again and again, but about different employees each time. For example, you might want to query the database to find out how many hours a particular employee has worked. To do that, you could run a query that you created and stored previously, but run it only for a particular employee.
FIGURE B-44 Dealing all employees who are not Delaware residents

**AT THE KEYBOARD**

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

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

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

FIGURE B-46 Design of a Parameter query, continued
Note that the Criteria line involves the use of square brackets, as you would expect to see in a calculated field.

Now run the query. You will be prompted for the specific employee's SSN, as shown in Figure B-47.

![FIGURE B-47 Enter Parameter Value dialog box](image)

Type in your own (fictitious) SSN. Your query output should resemble that shown in Figure B-48.

![FIGURE B-48 Output of a Parameter query](image)

**MAKING SEVEN PRACTICE QUERIES**

This portion of the tutorial is designed to give you additional practice in making 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. Follow this procedure: Set up a query in Access. Run the query. When you are satisfied with the results, save the query and continue with the next query. Note: You will be working with the EMPLOYEE, HOURS WORKED, and WAGE DATA tables.

1. Create a query that shows the SSN, last name, state, and date hired for those employees who are currently living in Delaware and were hired after 12/31/99. Sort (ascending) by SSN. (A recap of sorting procedures: Click the Sort cell of the field. Choose Ascending or Descending.) Before creating your query, use the table shown in Figure B-49 to work out your QBE grid on paper.

<table>
<thead>
<tr>
<th>Field</th>
<th>Table</th>
<th>Sort</th>
<th>Show</th>
<th>Criteria</th>
<th>Or:</th>
</tr>
</thead>
</table>

![FIGURE B-49 QBE grid template](image)

Your output should resemble that shown in Figure B-50.
2. Create a query that shows the last name, first name, date hired, and state for those employees who are currently living in Delaware or were hired after 12/31/99. The primary sort (ascending) is on last name, and the secondary sort (ascending) is on first name. (Sorting recap: The Primary Sort field must be to the left of the Secondary Sort field in the query setup.) Before creating your query, use the table shown in Figure B-51 to work out your QBE grid on paper.

<table>
<thead>
<tr>
<th>Field</th>
<th>Table</th>
<th>Sort</th>
<th>Show</th>
<th>Criteria</th>
<th>Or:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

FIGURE B-51 QBE grid template

If your name was Joe Brady, your output would look like that shown in Figure B-52.

<table>
<thead>
<tr>
<th>Last Name</th>
<th>First Name</th>
<th>Date Hired</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brady</td>
<td>Joe</td>
<td>9/14/2007</td>
<td>MD</td>
</tr>
<tr>
<td>Howard</td>
<td>Jane</td>
<td>8/1/2008</td>
<td>DE</td>
</tr>
<tr>
<td>Jones</td>
<td>Sue</td>
<td>7/15/2004</td>
<td>DE</td>
</tr>
<tr>
<td>Smith</td>
<td>Albert</td>
<td>7/15/1987</td>
<td>DE</td>
</tr>
<tr>
<td>Smith</td>
<td>John</td>
<td>6/1/1996</td>
<td>DE</td>
</tr>
</tbody>
</table>

FIGURE B-52 Number 2 query output

3. Create a query that shows the sum of hours that U.S. citizens worked and the same sum for non-U.S. citizens (that is, group on citizenship). The heading for total hours worked should be Total Hours Worked. Before creating your query, use the table shown in Figure B-53 to work out your QBE grid on paper.
4. Create a query that shows the wages owed to hourly workers for Week 1. The heading for the wages owed should be Total Owed. The output headings should be Last Name, SSN, Week #, and Total Owed. Before creating your query, use the table shown in Figure B-55 to work out your QBE grid on paper.

<table>
<thead>
<tr>
<th>Field</th>
<th>Table</th>
<th>Sort</th>
<th>Show</th>
<th>Criteria</th>
<th>Or:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**FIGURE B-54** Number 3 query output

If your name was Joe Brady, your output would look like that shown in Figure B-56.

<table>
<thead>
<tr>
<th>Last Name</th>
<th>SSN</th>
<th>Week #</th>
<th>Total Owed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Howard</td>
<td>114-11-2333</td>
<td>1</td>
<td>$420.00</td>
</tr>
<tr>
<td>Smith</td>
<td>148-90-1234</td>
<td>1</td>
<td>$475.00</td>
</tr>
<tr>
<td>Brady</td>
<td>099-11-3344</td>
<td>1</td>
<td>$510.00</td>
</tr>
</tbody>
</table>

**FIGURE B-56** Number 4 query output

5. Create a query that shows the last name, SSN, hours worked, and overtime amount owed for hourly employees who earned overtime during Week 2. Overtime is paid at 1.5 times the normal hourly rate for all of the hours worked over 40. Note that the amount shown in the query should be just the overtime portion of the wages paid. Also, this is not a Totals query—amounts should be
shown for individual workers. Before creating your query, use the table shown in
Figure B-57 to work out your QBE grid on paper.

<table>
<thead>
<tr>
<th>Field</th>
<th>Table</th>
<th>Sort</th>
<th>Show</th>
<th>Criteria</th>
<th>Or:</th>
</tr>
</thead>
</table>

**FIGURE B-57** QBE grid template

If your name was Joe Brady, your output would look like that shown in Figure B-58.

<table>
<thead>
<tr>
<th>Last Name</th>
<th>SSN</th>
<th>Hours</th>
<th>OT Pay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Howard</td>
<td>114-11-2333</td>
<td>50</td>
<td>$157.50</td>
</tr>
<tr>
<td>Brady</td>
<td>699-11-3344</td>
<td>55</td>
<td>$191.25</td>
</tr>
</tbody>
</table>

**FIGURE B-58** Number 5 query output

6. Create a Parameter query that shows the hours employees have worked. Have the Parameter query
prompt for the week number. The output headings should be Last Name, First Name, Week #, and
Hours. Do this only for the nonsalaried workers. Before creating your query, use the table shown
in Figure B-59 to work out your QBE grid on paper.

<table>
<thead>
<tr>
<th>Field</th>
<th>Table</th>
<th>Sort</th>
<th>Show</th>
<th>Criteria</th>
<th>Or:</th>
</tr>
</thead>
</table>

**FIGURE B-59** QBE grid template

Run the query using 2 when prompted for the Week #. Your output should look like that shown
in Figure B-60.

<table>
<thead>
<tr>
<th>Last Name</th>
<th>First Name</th>
<th>Week #</th>
<th>Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Howard</td>
<td>Jane</td>
<td>2</td>
<td>50</td>
</tr>
<tr>
<td>Smith</td>
<td>Albert</td>
<td>2</td>
<td>40</td>
</tr>
<tr>
<td>Brady</td>
<td>Joe</td>
<td>2</td>
<td>55</td>
</tr>
</tbody>
</table>

**FIGURE B-60** Number 6 query output
7. Create an update query that gives certain workers a merit raise. First, you must create an additional table as shown in Figure B-61.

<table>
<thead>
<tr>
<th>SSN</th>
<th>Merit Raise</th>
<th>Add New Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>114-11-2833</td>
<td>50.25</td>
<td></td>
</tr>
<tr>
<td>148-90-1234</td>
<td>50.15</td>
<td></td>
</tr>
</tbody>
</table>

**FIGURE B-61** MERIT RAISES table

Now make a query that adds the Merit Raise to the current Wage Rate for those employees who will receive a raise. When you run the query, you should be prompted with You are about to update two rows. Check the original WAGE DATA table to confirm the update. Before creating your query, use the table shown in Figure B-62 to work out your QBE grid on paper.

<table>
<thead>
<tr>
<th>Field</th>
<th>Table</th>
<th>Update to</th>
<th>Criteria</th>
<th>Or:</th>
</tr>
</thead>
</table>

**FIGURE B-62** QBE grid template

**CREATING REPORTS**

Database packages let you make attractive management reports from a table's records or from a query's output. If you are making a report from a table, the Access report generator looks up the data in the table and puts it into report format. If you are making a report from a query's output, Access runs the query in the background (you do not control it or see it happen) and then puts the output in report format.

There are three ways to make a report. One is to hand-craft the report from scratch in the Design View. Because that is a tedious process, it is not shown in this tutorial. The second way is to use the Report Wizard, during which Access leads you through a menu-driven construction. That method is shown in this tutorial. The third way is to start in the Wizard and then use the Design View to tailor what the Wizard produces. That method also is shown in this tutorial.

**Creating a Grouped Report**

This tutorial assumes that you already know how to use the Wizard to make a basic ungrouped report. This section of the tutorial teaches you how to make a grouped report. (If you don’t know how to make an ungrouped report, you can learn by following the first example in the upcoming section.)

**AT THE KEYBOARD**

Suppose you want to make a report out of the HOURS WORKED table. Choose the Create tab, Report Wizard in the Reports group. Select the HOURS WORKED table from the drop-down menu as the report basis. Select all of the fields (using the >> button), as shown in Figure B-63.

Click Next. Then tell Access you want to group on Week # by double-clicking that field name. This grouping step is shown in Figure B-64.

Click Next. You'll see a screen, similar to the one in Figure B-65, for Sorting and Summary Options.

Because you chose a grouping field, Access will let you decide whether you want to see group subtotals and/or report grand totals. If you choose that option, all numeric fields could be added. In this example, group subtotals are for total hours in each week. Assume you do want the total of hours by week. Click **Summary Options**. You'll get a screen similar to the one in Figure B-66.
FIGURE B-63  Field selection step in the Report Wizard

FIGURE B-64  Grouping step in the Report Wizard

FIGURE B-65  Sorting and Summary Options step in the Report Wizard
Next, follow these steps:

1. Click the Sum box for Hours (to sum the hours in the group).
2. Click Detail and Summary. (Detail equates with "group"; Summary, with "grand total for the report.")
3. Click OK. This takes you back to the Sorting screen, where you can choose an ordering within the group if desired. (In this case, you don’t choose one.)
4. Click Next to continue.
5. In the Layout screen (not shown here), choose Stepped and Portrait.
6. Make sure the “Adjust the field width so all fields fit on a page” check box is unchecked.
7. Click Next.
8. In the Style screen (not shown), accept Office.
9. Click Next.
10. Provide a title—Hours Worked by Week would be appropriate.
11. Select the Preview the report button to view the report.
12. Click Finish.

Your report will look like that shown in Figure B-67.

Notice that data is grouped by weeks, with Week 1 on top, then a subtotal for that week. Week 2 data is next, then a grand total (which you can scroll down to see). The subtotal is labeled “Sum,” which is not very descriptive. That can be changed later in the Design View. Also, there is the apparently useless “Summary for Week . . .” line, which can be deleted later in the Design View as well. Click the x in the upper-right corner of
your report window corresponding to "Close Hours Worked by Week." Note that the Navigation Pane on the left side of your screen contains the report shown in Figure B-68.

![Figure B-68 Navigation Pane with report](image)

To edit the report in the Design View, right-click the report title and choose Design view. You will see a complex (and intimidating) screen, similar to the one shown in Figure B-69.

![Figure B-69 Report design screen](image)

The organization of the screen is hierarchical. At the top is the Report level. The next level down (within a report) is the Page level. The next levels down (within a page) are for any data groupings you have specified.

If you told Access to make group (summary) totals, your report would have a Report Header area and end with a Grand Total in the Report Footer. The report header is usually just the title you specified.

A page also has a header, which is usually just the names of the fields you told Access to put in the report (here, Week #, SSN, and Hours fields). Sometimes the page number is put in by default.

Groupings of data are more complex. There is a header for the group. In this case, the value of the Week # will be the header, so the values shown will be 1 and 2. Those headers indicate that there is a group of data.
for the first week, then a group of data for the second week. Within each data grouping is the other “detail” that you’ve requested. In this case, there will be data for each SSN and the related hours.

Each Week # gets a footer, which is a labeled sum. Recall that you asked for the footer to be shown (i.e., Detail and Summary were requested). The Week # Footer is composed of three elements:

1. The line that starts =Summary for...
2. The Sum label
3. The adjacent expression =Sum(Hours)

The line beneath the Week # Footer will be printed unless you eliminate it. Similarly, the word Sum will be printed as the subtotal label unless you eliminate it. The =Sum(Hours) is an expression that tells Access to add the quantity for the header in question and to put that number into the report as the subtotal. (In this example, that number would be the sum of hours by Week #.)

Each report also gets a footer—the grand total (in this case, of hours) for the report.

If you look closely, each of the detail items appears to be doubly inserted in the design. For example, in Figure B-69, you will see the notation for SSN twice, once in the Page Header band and then again in the Detail band. Hours are treated similarly.

Those items will not, however, be printed twice, because each detail item in the report is an object. In Access, an object is denoted by both a label and its value. In other words, there is a representation of the name of the object, which is the boldfaced name itself (in this example, SSN in the Page Header band), and there is a representation in less bold type of the value of the object (corresponding to SSN in the Detail band).

Sometimes the Report Wizard is arbitrary about where it puts labels and data. If you do not like where the Wizard puts data, you can move the objects containing the data around in the Design View. Also, you can click and drag within the band or across bands. Often a box will be too small to allow the full values for that object to show. When that happens, select the box and click one of the sides to stretch it. Doing so will allow full values to show. At other times, an object’s box will be very long. When that happens, you can click the box, resize it, then drag right or left in its panel to reposition the output.

Suppose you do not want the Summary for... line to appear in the report. Also suppose you would like different subtotal and grand total labels. The Summary for... line is an object you can activate by clicking. Do that now. “Handles” (little squares) should appear around its edges, as shown in Figure B-70.

![FIGURE B-70 Selecting an object in the Report Design View](image)

Press the Delete key to get rid of the selected object.
To change the subtotal heading, click the Sum object, as shown in Figure B-71.

![FIGURE B-71 Selecting the Sum object in the Report Design View](image)

Click again. That gives you an insertion point from which you can type, as shown in Figure B-72.

![FIGURE B-72 Typing an object in the Report Design View](image)

Change the label to Sum of Hours for Week and press Enter or click somewhere else in the report to deactivate the object. Your screen should resemble the one shown in Figure B-73.

You can change the Grand Total in the same way.
In the Design tab, click the View button on the Views group. You should see a report similar to that in Figure B-74 (the top part is shown).
FIGURE B-73  Changing a label in the Report Design View

**Hours Worked by Week**

<table>
<thead>
<tr>
<th>Week</th>
<th>Hours Worked</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>60</td>
</tr>
<tr>
<td>724-60-1927</td>
<td>40</td>
</tr>
<tr>
<td>222-92-1122</td>
<td>40</td>
</tr>
<tr>
<td>148-90-1234</td>
<td>38</td>
</tr>
<tr>
<td>123-45-6789</td>
<td>40</td>
</tr>
<tr>
<td>114-11-2333</td>
<td>40</td>
</tr>
</tbody>
</table>

Sum of Hours for Week 258

FIGURE B-74  Hours Worked by Week report

Notice that the data is grouped by week number (data for Week 1 is shown) and subtotaled for that week. The report also has a grand total at the bottom.

**Moving Fields in the Design View**

If you want to group on more than one field in the Report Wizard, the report will have an odd “staircase” look, display repeated data, or it will have both features. Next, you will learn how to overcome that effect in the Design View.

Suppose you make a query showing an employee’s last name, first name, street address, zip code, week #, and hours worked. Then you make a report from that query, grouping on last name and first name only. See Figure B-75.

FIGURE B-75  Grouping in the Report Wizard

Then follow these steps:

1. Click Next.
2. Do not sum anything in Summary Options. Click Next.
3. Select Stepped.
4. Select Landscape.
5. Click the check mark by Adjust the field width so all fields fit on a page. Click Next.
7. Type a title (Hours Worked by Employees). Click Finish.

As you preview the report, you will notice repeating data. In the report displayed in Figure B-76, notice that the Zip data and Street Address data are repeating and that both are shown below the First Name data, which is below the Last Name data—hence, the staircase effect. (The fields Week # and Hours are shown subordinate to Last Name and First Name, as desired.)

![Hours Worked by Employees](image)

**FIGURE B-76**  Hours Worked by Employees grouped report

Suppose you want the last name, first name, street address, and zip code to appear on the same line. If so, take the report into the Design View for editing. From the Navigation Pane on the left side of your screen, right-click the Hours Worked by Employees report and choose Design View. At this point, the headers look like those shown in Figure B-77.

![Hours Worked by Employees report displayed in Design View](image)

**FIGURE B-77**  Hours Worked by Employees report displayed in Design View

Your goal is to get the First Name, Street Address, and Zip fields into the Last Name header band (not into the Page Header band) so they will print on the same line. The first step is to click the First Name object in the First Name Header band, as shown in Figure B-78.

![Selecting First Name object in the First Name header](image)

**FIGURE B-78**  Selecting First Name object in the First Name header

Right-click, cut, place your cursor in the Last Name header, and paste into the Last Name header, as shown in Figure B-79. *(Note: When you paste the First Name object, it will overlay the Last Name object. To correct that, simply move the First Name object to the right side of the Last Name object.)* Alternatively, you can right-click the First Name object, choose Layout, and then choose Move Up a Section.

Now cut the Street Address object and paste it into the Last Name header, as shown in Figure B-80.
Do the same for the Zip object. Now correct the spacing and the Page Header by choosing the Layout View from the View group on the Design tab. Click **Hours** and note that the entire column is selected (your heading and data). Drag the column to the right side of the report. At this point, your screen should look like that shown in Figure B-81.

Drag the left side of the Hours column to the right; the Week # heading and data will follow your cursor to the right. Your report should now resemble the one shown in Figure B-82.

Finally, adjust the headings by dragging the left edge of the Week # to the right and adding more descriptions such as First Name, Address, and Zip to the Last Name header. This final adjustment will result in a report similar to that shown in Figure B-83. (Note: This is the Report View.)
FIGURE B-83  Hours Worked by Employees report

IMPORTING DATA

Text or spreadsheet data is easily imported into Access. In business, it is often necessary to import data because companies use disparate systems. Assume your healthcare coverage data is on the human resources manager's computer in an Excel spreadsheet.

Open the software application Microsoft Excel. Create a spreadsheet in Excel using the data shown in Figure B-84.

FIGURE B-84  Excel data

Save the file; then close it. Now you can easily import that spreadsheet data into a new table in Access. With your Employee database open, go to the External Data tab, Import group and click Excel. Browse to find the Excel file you just created and make sure the first radio button (Import the source data into a new table in the current database) is selected, as shown in Figure B-85. Click OK.

Choose the correct worksheet. Assuming you have just one worksheet in your Excel file, your next screen should look like that shown in Figure B-86.

Choose Next and make sure you select the box that says First Row Contains Column Headings, as shown in Figure B-87.

Choose Next. Accept the default for each field you are importing on this screen. Each field is assigned a text data type, which, in this case, is correct for this table. Your screen should look like that shown in Figure B-88.

Choose Next. In the next screen of the Wizard, you'll be prompted to create an index—that is, define a primary key. Because you will be storing your data in a new table, choose your own primary key (SSN), as shown in Figure B-89.

Continue through the Wizard, giving your table an appropriate name. After importing the table, take a look at it and its design. (Highlight the Table option and use the Design button.) Note the width of each field (very large). Adjust the field properties as needed.