

Chapter 4

Using Database Management Systems

This chapter introduces relational database management systems and simple database queries, including how to extract data from a Microsoft Access database and export it into Excel for formatting, presentation, and analysis.

A student who successfully completes this chapter should be able to:

- define the database terms used in the chapter.
- describe what relational database management systems are.
- construct SQL queries to extract specified data from a database table.
- create database queries for a Microsoft Access database.
- export the results of Microsoft Access queries into Microsoft Excel.



Figure 4-1: What do all of the people in these pictures have in common? Databases. Buying something in a store, making a phone call, listening to music, text messaging, checking in for an airline flight, and registering for a class all involve using database management systems.

Section 4.1 – Introduction

In this section we will look at what a relational database management system is, see what software is used for database management systems, and see an overall view of the process for extracting data from Access that will be covered later in the chapter.

4.1a Relational Database Management Systems

A **database** is an organized collection of data, usually one that can be stored and processed using a computer system. It can be thought of as an electronic filing system. A **database management system** is a collection of software that enables users to store, organize, analyze, and extract information using a database.

Most transactions in everyday life involve a database. Buying something at a store, registering for a class, getting a driver's license – all of these are examples of transactions that involve databases. Modern recordkeeping depends heavily on database technology, but even simple things like making a phone call, listening to music, or cooking food can involve the use of a database in a device like a cell phone, a music player, or a microwave oven.

Almost all modern databases are relational databases. A **relational database** is a database with data organized as a set of tables, based on concepts from a branch of mathematics known as relational algebra.

The things we wish to keep track of in a database are called entities. In a relational database, there is a table for each entity. The word **entity** refers to the general object that each table describes. Each specific object is called an **instance** of the entity.

Entities do not need to be physical objects. A teaching assignment is an example of a non-physical entity – it is not a physical object, but it is still something we can record in a database. A bank transaction and a phone call are other examples of entities that are not physical.

Of course, modern databases have many tables, not just one. A college or university student records and registration system might have hundreds of tables – for students, courses, sections of courses, teachers, classrooms, student registration records, teaching assignments, and so on.

In fact, many databases are enterprise-wide databases. An **enterprise-wide database** is a single database with all of the data for an enterprise, such as a business, a government agency, or some other kind of organization. The trend in modern computing is to have all of the major operational software – accounting, sales, scheduling, personnel, and so on – store all of its data in one centralized enterprise-wide database management system, which is administered by database professionals.

snum	last	first	street	city	state	zip	phone	major	gpa
S00188	Smith	John	111 Oak Ave.	Princeton	NJ	08541	555-1111	CIS	3.43
S03154	Jones	Mary	212 Maple St.	Hanover	NH	03755	555-2222	Computer Science	3.62
S10843	Jarvis	Fred	123 Spruce St.	Providence	RI	02901	555-4321	English	3.07
S11013	Ali	Ben	327 Pine St.	Cambridge	MA	02140	555-1212	Accounting	2.91
S11783	Washington	Martha	419 Beech Lane	New York	NY	10027	555-1532	History	3.67
S36912	Williams	Cathy	231 Ash Terrace	New Haven	CT	06520	555-3251	Biology	2.64
S41134	Franklin	Richard	213 Birch Drive	Philadelphia	PA	19104	555-1334	Finance	3.14
S72394	Banner	Blythe	222 Larch St.	Ithaca	NY	14853	555-4422	Agriculture	3.01

Figure 4-2: part of a table of students from a relational database. There is a row for each individual student, and a column for each piece of information about the students. Each row describes an instance of the general entity student. Each column holds an attribute of the students.

Figure 4-2 shows part of a table of students. Student is the entity for this table. Each individual student is an instance of the entity student. There is a row in a database table for each instance of the entity. So, in our student table, there is a row for each student.

An **attribute** is something that provides information about or describes an entity. There is a column in a database table for each attribute of an entity. Our student table has a column for last name, a column for first name, and so on.

The intersection of a row and a column in a database table is called a **field**. Each field holds the value of one attribute for one instance of the entity. In *Access*, the term *field* is often used interchangeably with *attribute*.

In order for a database to work properly, we must be able to uniquely identify each instance of an entity. A **primary key** is an attribute that has a different value for each instance of an entity, so we can identify that instance. Every table in a relational database must have a primary key. In our student example, *student number* is the primary key for students. Two students can have the same name, but no two students can have the same student number, so student number works as a primary key.

Some primary keys require more than one column and are called *composite keys*, but we will not work with composite keys. Many database specialists say that composite keys should be avoided.

This is that nature of a relational database. We have a table for each entity, with a row for each instance of the entity and a column for each attribute of the entity. A special attribute, called the primary key, uniquely identifies each instance of the entity.

The **full database name** for a column includes the table name and the column name, separated by a comma. So, the columns in the student table would have the names *student.snum*, *student.last*, *student.gpa*, and so on.

Information about how a database is organized – in other words, data about the data – is called **metadata**. The metadata for a table tells us the name of each column in the table, the type of data in the column, and sometimes the size of the column. We will see more about data types and metadata later.

4.1b Relational Database Software

Structured Query Language – SQL for short – is a language we can use to query a relational database. It should be pronounced “S-Q-L” and not “sequel”, since *Sequel* is the name of a specific database management software package. SQL is based on relational algebra and is both a **data definition language** (DDL) used to define the structure of a database, and a **data manipulation language**, (DML) used to manipulate data in a database, including extracting data from a database. A standard definition of the SQL language is maintained by the *American National Standards Institute* (ANSI) and the *International Organization for Standardization* (ISO).

Technically, any SQL statement is called a **query**, but people generally use the word *query* to specifically mean statements that extract data from a database. We will learn about queries that extract data from a database, but not about SQL statements to create or modify databases. A more detailed look at database management systems and SQL is included in more advanced courses.

Most of the major relational database management systems in use today, such as *The Oracle Database*, *MySQL*, *Microsoft SQL Server*, *Ingres*, and *Microsoft Access*, use SQL. They each have some added commands, so their versions of SQL are slightly different, but they will understand and accept standard SQL queries. We won't look at all of these software packages, but we will learn how to write simple SQL standard queries that work on all of these systems.



Figure 4-3: logos for some of the most commonly used database management software. All use Structured Query Language – SQL.

4.1c A Practical Approach to using Access

It takes time and skill to create a properly working database. Well-educated professionals design and build the databases used by corporations, non-profit organizations, and governmental agencies. To do so requires years of training in both database management software and the underlying mathematics, such as relational algebra and relational calculus. Even databases for small companies, such as a pizza shop or a musical instrument store, are often purchased as part of specialized software packages written by experienced professionals.

We will not learn to create databases in this chapter. Nor will we learn to modify them, edit the data in a database, or add new data to a database. Most people do not interact directly with a database, but through the use of specialized software, often with a simple Web page as a user interface. For example, when you buy something online, such as books or music from *Amazon.com*, you are using Web pages that are interacting with a database.

So, rather than learning to design and build a database, you will learn to extract data from a database and import it into Microsoft Excel. You will learn to perform some simple queries on a database so that you can, given access to the data, extract the data you need and export it into Excel yourself. Then, you can use your spreadsheet skills to format, print, or analyze the data, or create graphs and charts from the data. This is what happens in the real world. Most business professionals have some skill using Excel. When they need data for a project they are working on, they request it from the database administrators in a form that can be used in Microsoft Excel.

So, why learn to perform database queries? There are several reasons. First, you will develop a better understanding of how data is organized and extracted from a database, which will help you to communicate better with information systems professionals. Second, in some situations, you *will* be allowed to see the data without changing it – granted *read-only* access to the data, and if you know how to perform a query, then you can ask the database to show you the information you want to see. Third, the knowledge you will gain by learning about database queries will give you a feeling for whether or not you would like to explore the topic further, perhaps as a career. Finally, you will gain some experience using Boolean logic, which will help in understanding logic, language, and the world around us a little more deeply and clearly.

In the following sections we will learn about *SQL* queries to extract data from a database, how to use them in *Access*, and how to use the *Access Query Wizard* and *Design View* to extract data without using *SQL*. We will only learn to perform single table queries with *SQL* because multiple table queries take too long to learn for this course. However the *Access Query Wizard* and *Design View* can help us to perform multiple table queries easily and quickly, so we will learn to perform multiple table queries using the *Access Query Wizard* and *Design View*. We will also learn how to easily export the results of a query into *Microsoft Excel*.

Microsoft Access does have a database *Report Wizard* that creates neatly formatted reports, but it often takes a lot of time and effort to get the data just the way we want it in a *Microsoft Access* report. The controls for *Access* reports can be tricky to learn and use. Most people find it more practical to simply move *Access* data into *Excel* and work with it there.

4.1 Section Review

- Describe the nature of a relational database. What are enterprise-wide databases and why are they important? How is data different from metadata?
- What is *Structured Query Language*? Who maintains standardized definitions of *SQL*? What is the technical definition of a *query* and what do many people generally use the word *query* to mean? What are some of the most commonly used database management software packages today?
- What background is needed in order to design and build databases? How do most people interact with large commercial databases? How can data be extracted from a *Microsoft Access* database?

— § —

4.2 Single Table Queries in SQL

In this section we will see how to format a query to extract data from a database table, and how to use comparison operators and Boolean logic in queries with different data types.

4.2a Formatting a Query

It is fairly easy to extract information from a single table in a relational database. The proper format for a basic *SQL* query to extract information from a table is:

```
SELECT (attributes) FROM (table) WHERE (condition);
```

For example, if we wish to get a list of the names and phone numbers from our student table for all of the students who live in New York, the proper *SQL* query would be:

```
SELECT first, last, phone FROM student WHERE state = "NY";
```

This query would return:

First	Last	Phone
Martha	Washington	555-1532
Blythe	Banner	555-4422

The attribute list tells the database what attributes, or columns, we wish to see in our result. The condition tells the database which instances, or rows, we wish to include in our result. The database will look at the rows one at a time and include any row that meets the condition. In this case, the attribute lists tells the database to show us the data in the *first*, *last*, and *phone* columns and the condition tells the database to include every row that has *NY* in the *state* column.

Notice that our result is in the format of a new table. That's what a *SELECT... FROM... WHERE...* query does – it uses the criteria we specify to create and return a new table from existing tables in a database. This *SQL* query returns a table with the *first*, *last* and *phone* columns for the two rows with *NY* as the value in the *state* column.

The SQL language is not case sensitive, so capitalization doesn't matter for SQL commands, table names, and column names. However, capitalization does matter for the data in the database, so be careful. The column name *state* could be *STATE* or *State*, but "ny", "Ny" and "NY" are not all the same data. We must ask for the data the way it was entered into the database.

Usually, SQL programmers and database administrators put SQL commands in UPPERCASE and table and column names in lowercase to make queries easier to read, so that's a good practice for us to adopt also. In our simple extraction queries, *SELECT*, *FROM*, and *WHERE* should be UPPERCASE, but column names like *last* and *first*, and table names like *student*, should be lowercase.

SELECT... FROM... WHERE... queries are often written on three lines to make them easier to read and understand, like this:

```
SELECT first, last, phone
FROM student
WHERE state = "NY";
```

SQL queries end with a semicolon. A computer will use the semicolon to recognize the end of the query, so it doesn't matter if the query is on one line, or spread out over several lines.

Creating SQL *SELECT... FROM... WHERE...* queries for a single table is fairly easy, but multiple table queries can be difficult to create, because factors based on relational algebra need to be considered, such as the data dependencies between tables. So, we will not attempt to write multiple table SQL queries in this course.

4.2b Data Types

The columns in a relational database each have a **data type**, which indicates how the computer should format the data, and which operations it can perform on the data. The way columns are used in database queries depends on their data types. The most common data types are *text*, *numeric*, and *Boolean*.

Text data, also called character data, is simply a string of text stored in a location in the database. It can include any characters that can be typed on a standard keyboard. Things like names and addresses are stored as text data, but so are numbers that will not be used for arithmetic, like phone numbers and zip codes. Text data is used in database queries with quotes around it. The first three examples in Figure 4-4 use text data with quotes.

Numeric data is a number stored in a database in a way that allows computers to perform arithmetic with it. Data that might be used in an arithmetic operation, such as *hours* or *rate* in a payroll database, or *temperature*, *mass*, *length* or *width* in a scientific database, should be stored as numeric data. Numeric data is used in database queries without quotes around it. The last three examples in Figure 4-4 use numeric data without quotes.

Some numbers, such as social security numbers and student numbers, are used as ID numbers, and not to perform arithmetic, so they should be stored as text data, not numeric data.

Boolean data is data that has a *true* or *false* value. We have all probably seen forms that ask us to check a box if something is true, such as if the applicant is a U.S. citizen. These can be thought of as true or false values, and are stored in a computer as Boolean data with a value of *true* or *false*. A *citizen* attribute with a Boolean data type in a database would correspond to a citizen checkbox on a paper form.

In *Microsoft Access*, Boolean data types are often called *Yes/No* data types.

There are many other data types, such as *date* and *time*, and a special data type called *binary large object (BLOB)* for objects such as picture files, videos, and sound clips. There are even specialized number formats for integers, floating point numbers, and different currencies. For the rest of this chapter, we will only use the *text*, *numeric*, and *Boolean* data types so that we can focus on how to extract data and format our results.

4.2c Query Conditions

The conditions in relational database queries are **Boolean conditions** – conditions that are either true or false. They usually involve one of six comparison operators – *equals*, *does not equal*, *is less than*, *is not less than*, *is greater than*, or *is not greater than*, shown in Figure 4-4. The six comparison operators cover all possibilities. Note that “A is less than or equal to B” is the same as “A is not greater than B”. “A is greater than or equal to B” is the same as “A is not less than B”.

Condition	In Math	In SQL	Examples
A equals B	$A = B$	$A = B$	zipcode = "19130"
A is not equal to B	$A \neq B$	$A <> B$	status <> "active"
A is less than B	$A < B$	$A < B$	name < "Miller"
A is greater than B	$A > B$	$A > B$	temperature > 98.6
A is less than or equal to B	$A \leq B, A \nless B$	$A \leq B$	rate <= 12.50
A is greater than or equal to B	$A \geq B, A \ngtr B$	$A \geq B$	hours >= 40

Figure 4-4: the six comparison operators used in Boolean conditions. The values being compared can be column names, or expressions that have the same data type as a column name.

Comparisons are not needed for Boolean data, since it is already *true* or *false*. So, Boolean attributes are used by themselves in logical conditions. The proper SQL command to return a list of all rows where the *citizen* attribute is true would be:

```
SELECT first, last FROM delegate WHERE citizen;
```

The condition is simply **WHERE citizen**, not **WHERE citizen = TRUE**.

Similarly, proper SQL command to return a list of all rows where the *citizen* attribute is not true would be:

```
SELECT first, last FROM delegate WHERE NOT(citizen);
```

The **NOT** operator is discussed in the next section.

4.2d Boolean Functions in Database Queries

Boolean logic is a form of mathematics with only true and false values. It is the basis of all modern computing, and is critical in the creation of database queries. There are three basic functions in Boolean logic: **AND**, **OR**, and **NOT**. They are used to form compound logical conditions from simple conditions.

For example, Assume that we wish to find the names of employees in the Accounting department with at least 40 years of experience. The two conditions we need to use are: **department = "Accounting"** **experience >= 40**.

We use the **AND** operation to join the two simple conditions into one compound condition:

```
SELECT first, last FROM employee WHERE department = "Accounting" AND experience >= 40;
```

The **AND** operation is a binary operation, meaning that it needs two operands. If both operands are true, then the result is true. Otherwise, the result is false. Both operands must be true for the result to be true.

The **OR** operation also has two operands. If either one of the operands is true, then the result is true. Otherwise, the result is false. Both operands must be false for the result to be false.

The **NOT** operation is a unary operation, meaning that it has only one operand. It simply reverses the true or false value of its operand. If the operand is true, then the result is false. If the operand is false, then the result is true.

Figure 4-5 shows truth tables for the primary Boolean operations. They can be read like multiplication tables that define what the result of a function will be based on the value of its operands.

$c = a \text{ AND } b$			
a	b		
	true	false	
true	true	false	
false	false	false	

$c = a \text{ OR } b$			
a	b		
	true	false	
true	true	true	
false	true	false	

$c = \text{NOT } a$		
a	true	false
true	false	true

Figure 4-5: truth tables for the AND, OR and NOT operations.

4.2d SQL Query Examples

Figure 4-6 contains metadata for a table of cars. It will be used for some example queries to illustrate how Boolean logic and comparison operators work in SQL queries.

Table: car			
column name	data type	size	notes about the data
VIN	Text	20	Vehicle Identification Number (primary key)
make	Text	40	manufacturer ("Ford", "Toyota", etc.)
model	Text	20	model name ("Focus", "Camry", etc.)
color	Text	2	
year	text	4	
engine	text	10	
auto	Boolean		true = automatic transmission; false = manual
value	Number		value of the car in US dollars

Figure 4-6: metadata for a table of cars.

The following examples each show a request for data, followed by an SQL command to extract the desired data.

Example 1 – list the VIN, model, year and value for all Fords.

```
SELECT VIN, model, year, value FROM car WHERE make = "FORD";
```

Example 2 – list the make, model, year, and value for all cars worth more than \$20,000.

```
SELECT make, model, year, value FROM car WHERE value > 20000;
```

Example 3 – list the model, year and value for all cars that have an automatic transmissions.

```
SELECT model, year, value FROM car WHERE auto;
```

Example 4 – list the model, year and value for all cars that do not have an automatic transmissions.

```
SELECT model, year, value FROM car WHERE NOT(auto);
```

Example 5 – list the VIN, year and value for all Honda Civics.

```
SELECT VIN, year, value FROM car WHERE make = "Honda" AND model = "Civic";
```

Example 6 – list the make, model, year and value for all Subarus and Volvos.

```
SELECT make, model, year, value FROM car WHERE make = "Subaru" OR make = "Volvo";
```

Example 7 – list the model, year and value for all red Porsches valued at less than \$40,000.

```
SELECT model, year, value FROM car WHERE color = "red" AND make = "Porsche" AND value <= 40000;
```

Example 8 – list the make model, year and value for cars that are not Chevrolets.

```
SELECT make, model, year, value FROM car WHERE make <> "Chevrolet";
```

```
SELECT make, model, year, value FROM car WHERE NOT (make = "Chevrolet");
```

Notice in Example 2 and in Example 7 that no commas or currency symbols are used for numbers in queries.

It is not unusual for people to use language loosely, or even incorrectly when asking for data, but the SQL query needs to be precise. Notice the difference between the request and the SQL query language in example 6. The request asks for all Subarus and Volvos, but each car has only one make. The list will have all Subarus *and* all Volvos, but each row extracted will have a Subaru *or* a Volvo. The condition is checked for each row.

In many cases more there is more than one correct way to word a query. Example 8 shows this.

4.2 Section Review

- Describe the proper format for a basic SQL query to extract information from a relational database table. Give an example of such a query. How is the result set from such a query formatted?
- What is a data type? List and describe three of the most commonly used database data types. How are values with these three data types used in an SQL query?
- Describe the six logical comparison operators used in SQL queries and the symbols used for each.
- Describe the three Boolean operators used for compound conditions in an SQL query.
- Create SQL queries to extract the following from the *house* table, whose metadata is shown in Figure 4-7:

Table: house			
column name	data type	size	notes about the data
ID	Text	5	house ID number (primary key)
street	Text	40	street address
city	Text	20	
state	Text	2	
zip	Text	5	5-digit zip codes only
bedrooms	Number		number of bedrooms; integer
bathrooms	Number		number of bathrooms; could end in .5
value	Number		appraised value or sale price of house
first	Text	20	first name of the listing or selling agent
last	Text	20	last name of the listing or selling agent
sold	Boolean		true = sold ; false = not sold

Figure 4-7: metadata for a table of houses.

- The *House ID*, *street address*, and *value* for houses in the 19140 zip code under \$240,000.
- The *House ID*, *street address*, and *value* for houses in the 19128 zip code with at least three bedrooms and two bathrooms.
- The *House ID*, *street address*, *zip code* and *value* for houses sold by Mark Jones.
- The *street address*, *value*, *number of bedrooms* and *number of bathrooms* for houses in the 19116 and 19154 zip codes.
- The *House ID*, *street address* and *value* for houses in the 19139 zip code that have not been sold.

Section 4.3 – Microsoft Access Queries

In this section we will examine queries in *Microsoft Access*. We will see how to enter *SQL* queries in *Access*, how to use the *Access Query Wizard* and *Design View* to create queries without using *SQL*, and how to export data resulting from *Access* queries into *Excel*.

4.3a Microsoft Access Overview

Before beginning this section, we need to make sure that the *Access* database file *school.accdb*, which comes with the student data files for this chapter, is available and ready to use. Your instructor can help you to locate the file. In this example, we will assume that it is in the *My Documents* folder. You should copy the file to the *My Documents* folder on the computer you are using.

Notice that the file ends with the extension *.accdb*. The file extension for newer *Access* database files is *.accdb*. Versions of *Access* before *Office Access 2007* used database files with the file extension *.mdb*. We can still open *.mdb* files in *Access 2007*.

Your computer may have a shortcut for *Microsoft Access*. If so, then you can use it to open *Access*. Otherwise, follow these steps:

To Open Microsoft Access

1. Click the **Start Button** in the bottom left hand corner of the Windows desktop.
2. Select **All Programs** from the menu that appears.
3. Select **Microsoft Office**, then, select **Microsoft Office Access** as shown in Figure 4-8.

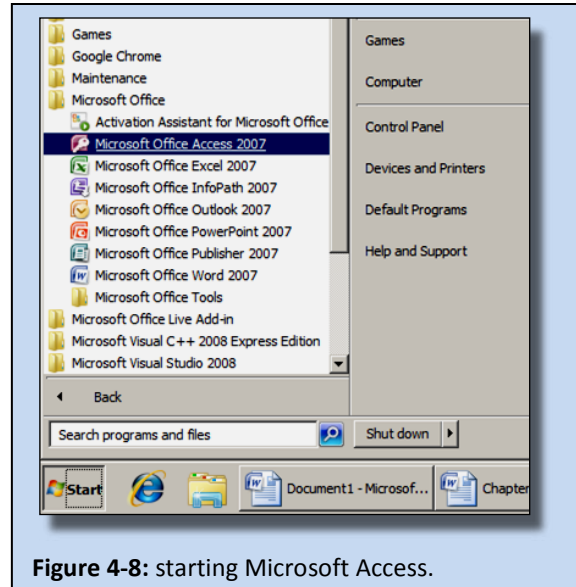


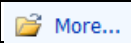
Figure 4-8: starting Microsoft Access.

The opening screen for *Access* should look something like that in Figure 4-9. It may be slightly different, depending on how *Access* was installed on the computer you are using.



Figure 4-9: the opening screen for *Microsoft Access 2007*. The screen may differ slightly from one computer to another, The numbers were added to identify items on the screen: 1 an icon to start a new blank database, 2 menus for new database templates, 3 icons for new database templates, 4 a menu to open existing databases.

To Open the *school.accdb* Access database

1. Click the  icon at the top of the menu to open existing databases.
2. Navigate to **My Documents** and find the file *school.accdb*.
3. Double-click the *school.accdb* entry to open the file.

The screen should now resemble the one shown in Figure 4-10.

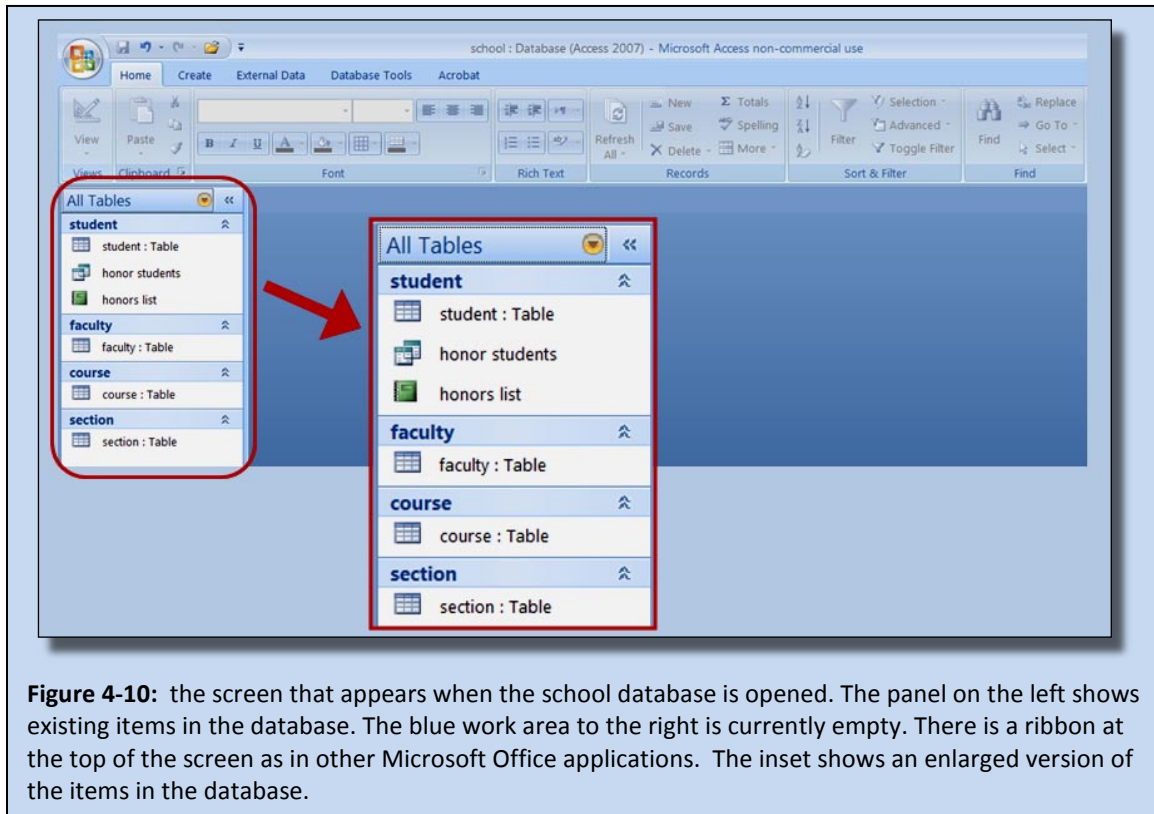


Figure 4-10: the screen that appears when the school database is opened. The panel on the left shows existing items in the database. The blue work area to the right is currently empty. There is a ribbon at the top of the screen as in other Microsoft Office applications. The inset shows an enlarged version of the items in the database.

Let's take a quick look at a table, a query, and a report without making any changes to them.

Examining items in the school database

1. Double-click the ***student: Table*** entry in the list of items to see what a table looks like in Access. The table will open in the *datasheet view*, which allows us to see the data in the form of a table with rows and columns, as shown in Figure 4-11.

student							
snum	last	first	street	city	state	zip	
S00188	Smith	John	111 Oak Ave.	Princeton	NJ	0854	
S03154	Jones	Mary	212 Maple St.	Hanover	NH	0375	
S10843	Jarvis	Fred	123 Spruce St.	Providence	RI	0290	
S11013	Ali	Ben	327 Pine St.	Cambridge	MA	0214	
S11783	Washington	Martha	419 Beech Lane	New York	NY	1002	
S36912	Williams	Cathy	231 Ash Terrace	New Haven	CT	0652	
S41134	Franklin	Richard	213 Birch Drive	Philadelphia	PA	1910	
S72394	Banner	Blythe	222 Larch St.	Ithaca	NY	1485	

Figure 4-11: a datasheet view of the *student* table in Access.

- Right-click the tab for the student table  and select **Design View** from the menu that appears. You should now see the student table in *design view*, which gives us the metadata for the database, as shown in Figure 4-12.

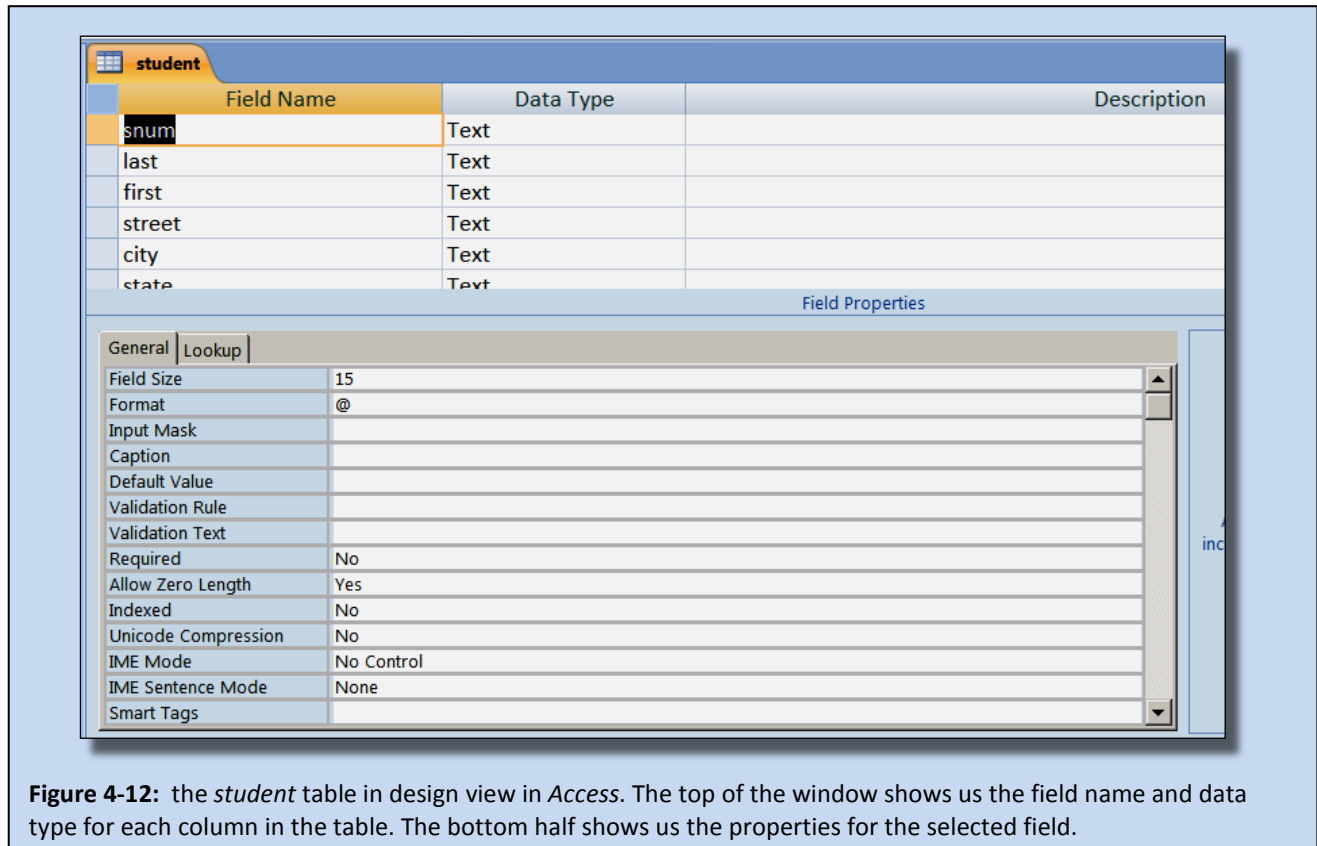
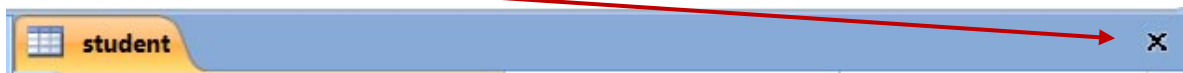


Figure 4-12: the *student* table in design view in Access. The top of the window shows us the field name and data type for each column in the table. The bottom half shows us the properties for the selected field.

Design View for tables can be used to see a table's metadata, but only database administrators should change any of the metadata, otherwise, the database might not work properly. Well managed databases will have both the data and metadata protected so that users can't make changes without permission. This database is not protected.

- Click the **X** on the same line as the *student* tab (not the X at the top of the screen to close the Window) to close the table.



- Double-click the **honors students** entry in the list of items to see what a query looks like in Access.

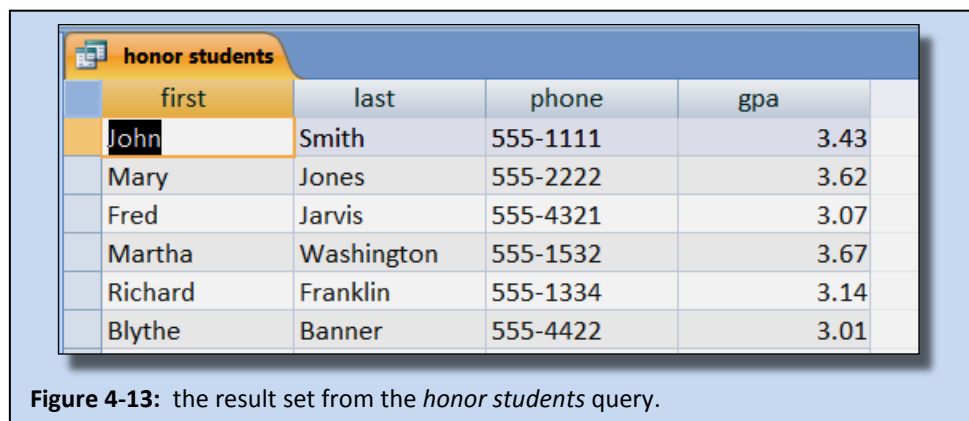



Figure 4-13: the result set from the *honor students* query.

Figure 4-13 shows the result set from the *honors student* query. The result of a query is a new table with a set of data from the database as specified by the query command or in the query wizard. We shall learn more about queries in the following sections.

The *View* button near the top left corner of the screen has a small arrow at the bottom, as shown in Figure 4-14.

5. Click the small arrow below the word *View* on the button to see the list of query views on the right in Figure 4-14.

6. Select the  option from the menu that appears.

You should now see the SQL command for the query, as shown in Figure 4-15. SQL commands can be entered in this window, or created with the query wizard. We will learn to do both later in this chapter.

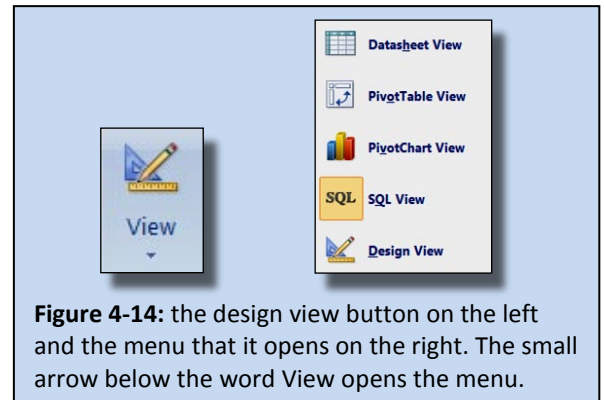


Figure 4-14: the design view button on the left and the menu that it opens on the right. The small arrow below the word *View* opens the menu.

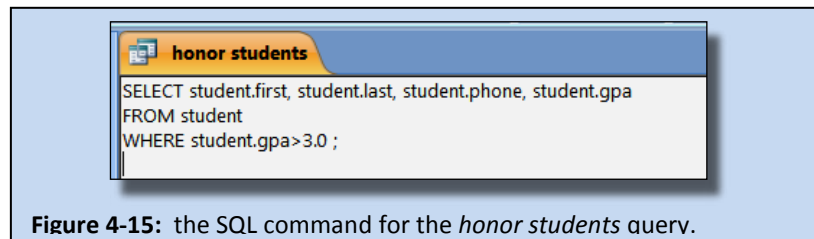


Figure 4-15: the SQL command for the *honor students* query.

7. Click the **X** on the same line as the *honor students* tab to close the query.
8. Double-click the **honors list** entry in the list of items to see what an Access report looks like.

It should resemble the report shown in Figure 4-16.

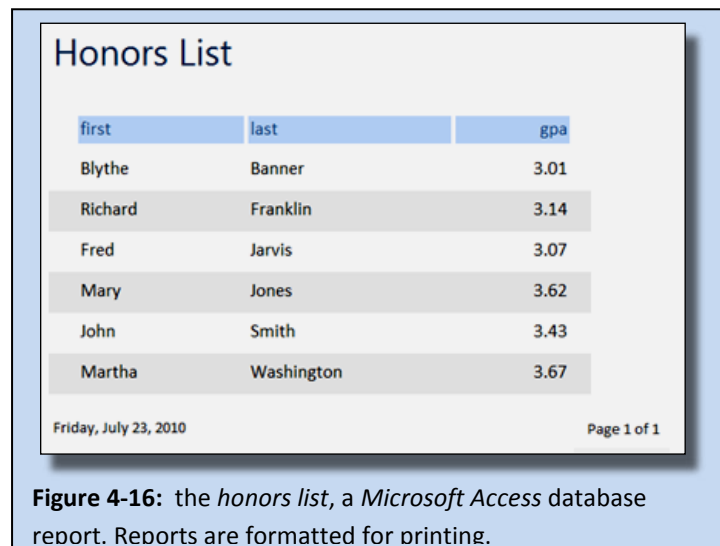



Figure 4-16: the *honors list*, a Microsoft Access database report. Reports are formatted for printing.

Microsoft Access queries extract data from a database. *Access* reports tell the computer how to format the data so that it looks good on the printed page. Reports can be created from a table or from the result set of a query. We will not learn to use the *Report Wizard* in this chapter, because even though it is easy to generate a report, editing a *Microsoft Access* report can be more tedious and time consuming than moving the data to Excel and working with it there, if you already know how to use Excel.

4.3b Entering SQL Queries in Access

Let's go back to our school database and enter an SQL query in *Access*. We will create a single table query extracting data from just the student table. This exercise assumes that you have a copy of the *school.accdb* database file in My Documents

To Open the *school.accdb* Access database

1. Open *Microsoft Access*.
2. Click the  icon at the top of the menu to open existing databases.
3. Navigate to **My Documents** and find the file *school.accdb*.
4. Double-click the *school.accdb* entry to open the file.

The Access screen should be very similar to that shown back in Figure 4-10. We will enter an SQL query to extract the name, major and GPA for all students with a GPA over 3.0:

SELECT first, last, major, gpa FROM student WHERE gpa > 3.0;

To enter an SQL query in Access

1. Click **Create** on the menu bar near the top of the screen to see the *Create* ribbon.

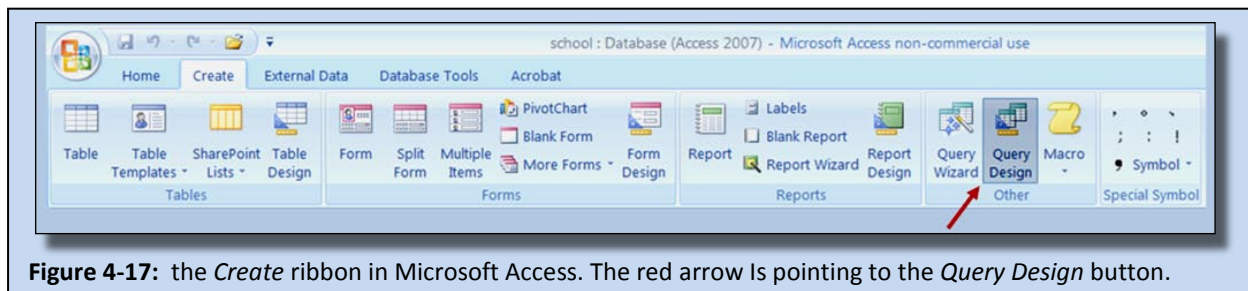


Figure 4-17: the *Create* ribbon in Microsoft Access. The red arrow is pointing to the *Query Design* button.

2. Click the *Query Design* button, highlighted in Figure 4-17. The screen should now resemble that shown in Figure 4-18, with the *Show Table* dialog window open. (The red circle has been added to the image.)

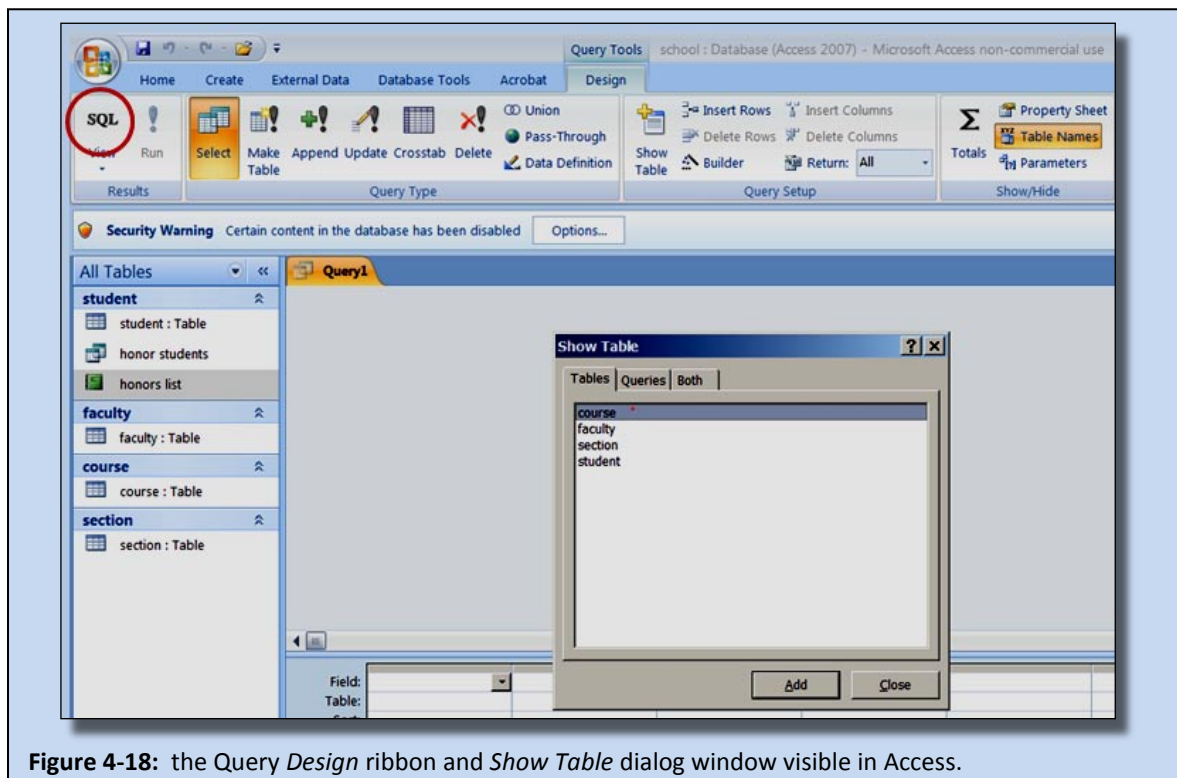
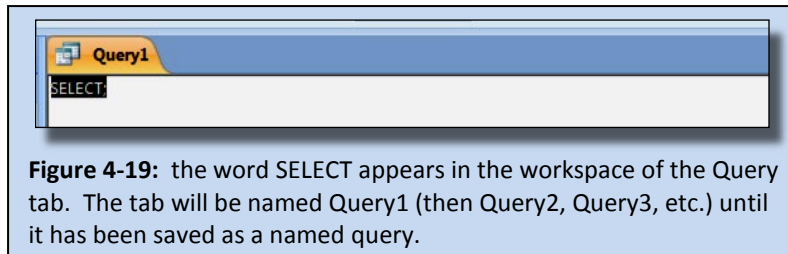


Figure 4-18: the *Query Design* ribbon and *Show Table* dialog window visible in Access.

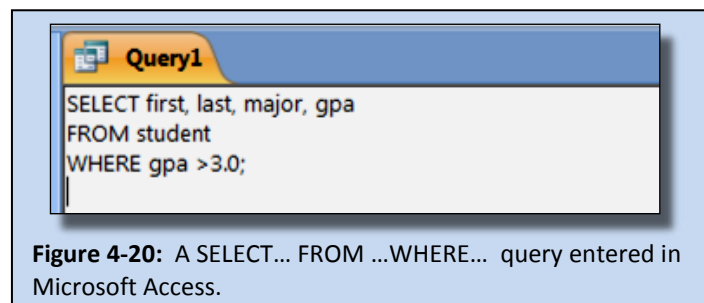
3. Close the *Show Table* dialog window, which we will not use, then click the **SQL** button, which is highlighted by the red circle near the top left corner of the screen in Figure 4-18. If you cannot see the **SQL** button, then click the small arrow below the word *View* in the same location, and select **SQL View** from the menu that appears.
4. The word *SELECT* should appear in the workspace of the *Query1* tab, as shown in Figure 4-19. This is the workspace where SQL queries can be entered directly into access.



5. Enter the new SQL query in the workspace in place of *SELECT*. When *SELECT...FROM... WHERE...* queries are entered in database management system, they are often written on separate lines to make them easier to read, as described earlier. So, enter the query as follows:

```
SELECT first, last, major, gpa
FROM student
WHERE gpa > 3.0;
```

Entering the query like this will make it more readable, and easier to find mistakes. We should expect to make mistakes occasionally when entering queries by hand. Be sure to use the quotes and punctuation marks correctly, including the semicolon at the end of the command. Figure 4-20 shows what this looks like in Access.



6. Proofread the query and edit it if necessary to make sure it's correct. Now we are ready to save and run the query.

To save an SQL query in Access

1. Click the **X** on the same line as the name of the query, which in this case it still *Query1*.
2. Click the **Yes** button in the question window that appears.
3. **Name** the query *honors students with major*, then click **OK**.

The query will close, and a new entry for the saved query will appear in the list of items in the database, as shown in Figure 4-21. In this list, tables have an icon that looks like a table, saved queries have an icon that looks like a double table, and saved reports have an icon that looks like a green report cover. Clicking an icon for a saved query or report will execute that item.

To run an existing query in Access

1. To run an existing query in access, simply double-click the entry for the query in the list of items in the database.
2. If an error message appears, you probably need to correct a syntax error in the query and try again.

A table holding the results of the query should appear in the workspace, as shown in Figure 4-22. This result set is a temporary table that is not stored in the database, but is re-created each time the query is executed. A query can be saved and reused whenever we wish. Each time it is executed, it will generate a new result set

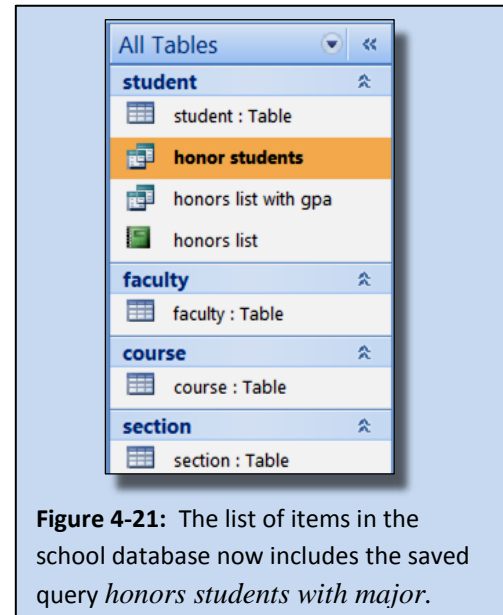


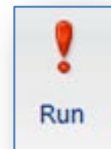
Figure 4-21: The list of items in the school database now includes the saved query *honors students with major*.

from the database at that time.

first	last	major	gpa
John	Smith	CIS	3.43
Mary	Jones	Computer Science	3.62
Fred	Jarvis	English	3.07
Martha	Washington	History	3.67
Richard	Franklin	Finance	3.14
Blythe	Banner	Agriculture	3.01

Figure 4-22: The result set for a query.

NOTE: *Design View* has a run button, with a red exclamation point, near the top left corner of the screen. It can be used to run queries directly from inside *Design View*.



4.3c Using the Query Wizard

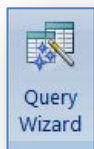
The *Query Wizard* allows us to create queries without entering *SQL* commands. It is often used in conjunction with *Query Design View* to refine the query created by the *Query Wizard*. Let's create the same query with the *Query Wizard* and *Design View*. Remember, our query is:

SELECT first, last, major, gpa FROM student WHERE gpa > 3.0;

We can select the table and columns with the *Query Wizard*, then add the condition in *Design View*.

To create a query in Access using the Query Wizard

1. Click **Create** on the menu bar near the top of the screen to reveal the *Create* ribbon.
2. Click the **Query Wizard** button. A *New Query* dialog window will appear. We wish to use the **Simple Query Wizard**, which is the default selection, so don't change anything, simply click the **OK** button.



The simple query wizard will appear, as shown in Figure 4-23.

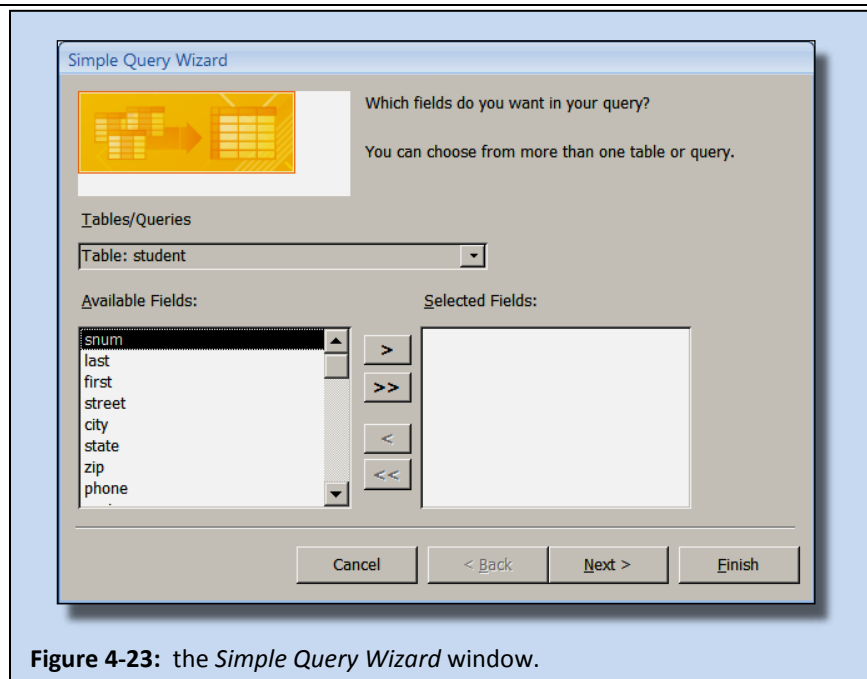


Figure 4-23: the Simple Query Wizard window.

The *Query Wizard* asks us to select the tables or other queries to be used as the basis for the new query, and to select which fields we will use. We will only use the *student* table. Any *field* that is used in the query needs to be selected. In our case, *first*, *last*, *major* and *gpa* are the four fields we will use.

1. Make sure that **Table: student** is selected in the Table/Queries box.
3. One at a time, double-click each of the four fields we will use – **first**, **last**, **major** and **gpa**. The order in which they are chosen will determine the order in which they will appear in the result set.
4. Click the **Next** button.
5. The *Query Wizard* will ask us if we want a *detailed* or *summary* query. **Detail** is the default. It will show us the complete result set for our query. Leave it as is and click the **Next** button.
6. The *Query Wizard* will ask us to name the query. Type the name **honors list from wizard** in the name box. Select **Modify the query design** from the options further down in the window, then click the **Finish** button.

The query *Design View*, shown in Figure 2-24 should appear. Notice that the list of items in the database now has an entry for our new query – *honors list from wizard*.

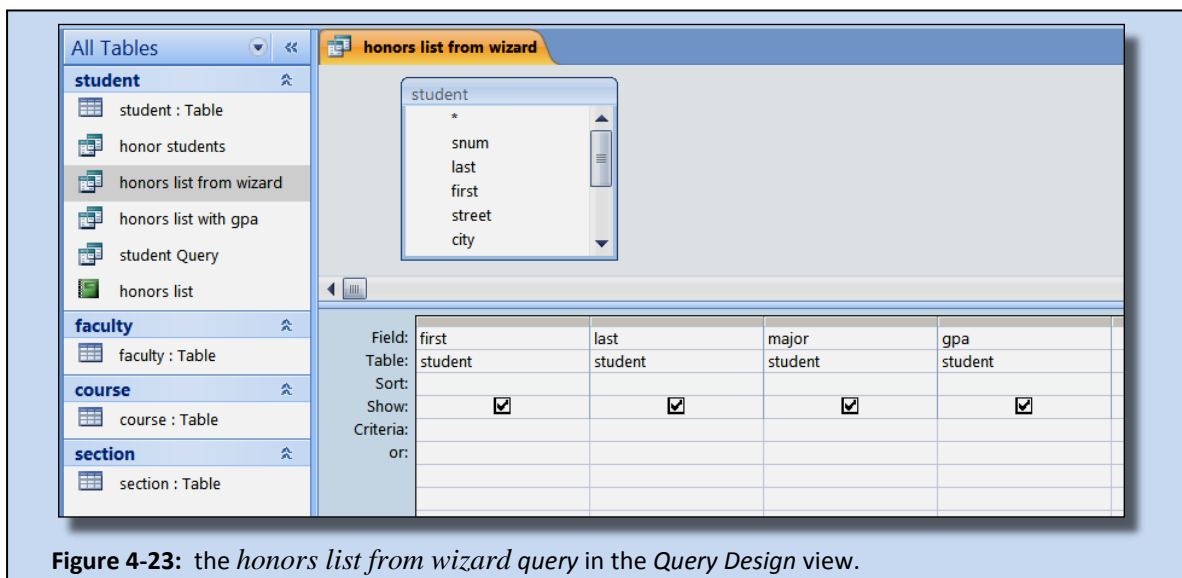


Figure 4-23: the *honors list from wizard* query in the Query Design view.

The boxes checked in the *Show* line will determine which columns appear in the result set. The expressions in the *Criteria* line will determine which rows will be included in the result set, just like the condition after *WHERE* in a *SELECT... FROM... WHERE...* query in SQL. In the following steps, we will leave all four columns checked, and will enter the criteria > 3.0 in the *gpa* column, which is equivalent to the SQL condition $gpa > 3.0$.

To add a condition to an Access query in Design View

1. Make sure the *Show* boxes for *first*, *last*, *major* and *gpa* are checked.
2. Type > 3.0 in the criteria field for *gpa*.
3. Click the **Run** button to see the results of the query.
4. After viewing the result set, click the **X** on the same line as the *honors list from wizard* tab to close the query.
5. Click the **Save** button when the dialog window appears.
6. If you are going to continue with the following exercise, then leave *Microsoft Access* open, otherwise, close it.



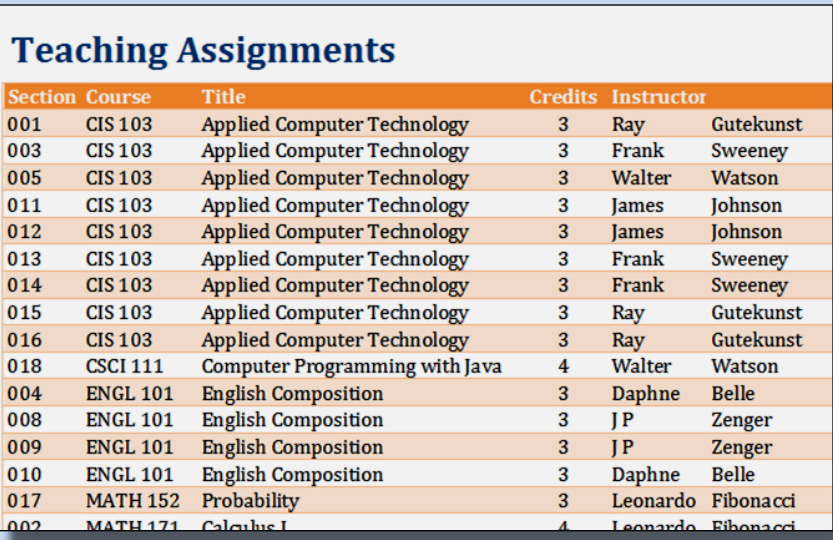
first	last	major	gpa
John	Smith	CIS	3.43
Mary	Jones	Computer Science	3.62
Fred	Jarvis	English	3.07
Martha	Washington	History	3.67
Richard	Franklin	Finance	3.14
Blythe	Banner	Agriculture	3.01

Figure 4-25: The result set for the query generated by the wizard. It matches the results from the similar SQL query.

4.3d Multiple Table Queries in Access

Learning to create *SQL* queries on multiple tables can be difficult and time consuming, because our queries must tell the database how to cross reference the tables in question. The *Access Query Wizard* and *Design View* can be used to execute such queries more quickly and easily, because we simply specify which columns from different tables we wish to extract, and *Access* will cross reference the data for us and give us the correct results.

For example, consider a list of faulty teaching assignments, shown in Figure 4-26. It should include columns for the *section number* and *course number*, the *course name* and *number of credits*, and the instructor's *first name* and *last name*. The *section number* and *course number* will come from the *section* table. The *course name* and *number of credits* will come from the *course* table and the instructor's *first name* and *last name* will come from the *faculty* table. The query involves three tables.



Section	Course	Title	Credits	Instructor
001	CIS 103	Applied Computer Technology	3	Ray Gutekunst
003	CIS 103	Applied Computer Technology	3	Frank Sweeney
005	CIS 103	Applied Computer Technology	3	Walter Watson
011	CIS 103	Applied Computer Technology	3	James Johnson
012	CIS 103	Applied Computer Technology	3	James Johnson
013	CIS 103	Applied Computer Technology	3	Frank Sweeney
014	CIS 103	Applied Computer Technology	3	Frank Sweeney
015	CIS 103	Applied Computer Technology	3	Ray Gutekunst
016	CIS 103	Applied Computer Technology	3	Ray Gutekunst
018	CSCI 111	Computer Programming with Java	4	Walter Watson
004	ENGL 101	English Composition	3	Daphne Belle
008	ENGL 101	English Composition	3	J P Zenger
009	ENGL 101	English Composition	3	J P Zenger
010	ENGL 101	English Composition	3	Daphne Belle
017	MATH 152	Probability	3	Leonardo Fibonacci
002	MATH 171	Calculus I	4	Leonardo Fibonacci


Figure 4-26: A teaching assignment report based on an Access query. The query results were saved as an Excel Spreadsheet and formatted and printed in Excel.

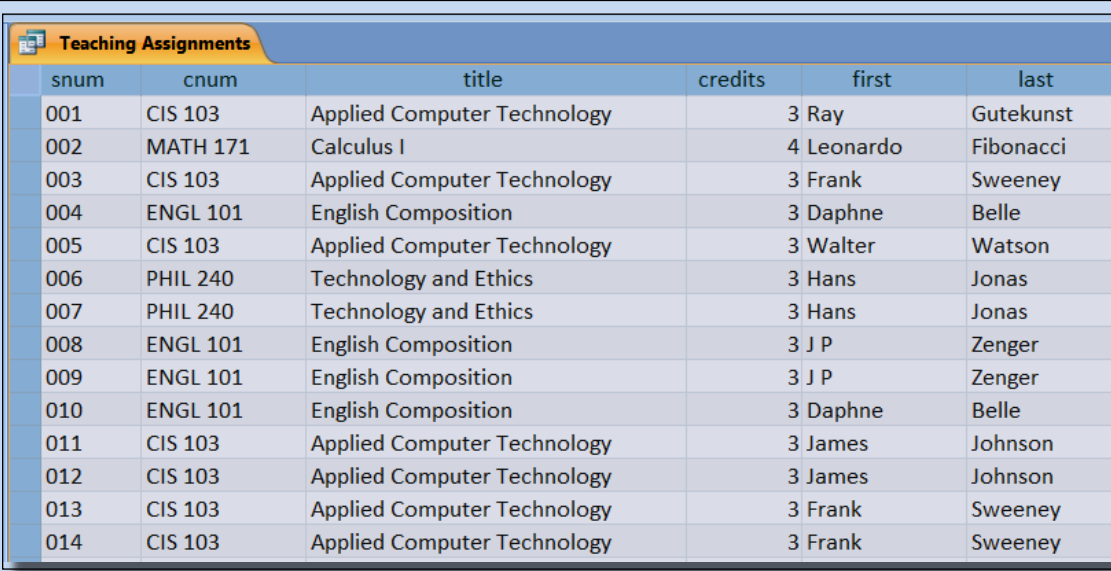
The SQL query to generate the data for this report is:

```
SELECT section.snum, section.cnum, course.title, course.credits, faculty.[First Name], faculty.[Last Name]
FROM faculty RIGHT JOIN (course RIGHT JOIN [section] ON course.cnum = section.cnum) ON faculty.fnum = section.fnum;
```

Creating multiple table queries like this in SQL requires more knowledge than we can deal with in this chapter, but we can generate the same data quickly and easily using the *Access Query Wizard*, as you will see.

To create a multiple table query using the Query Wizard:

1. Start *Microsoft Access*, and open the *school* table.
2. Click **Create** on the menu bar near the top of the screen to reveal the *Create* ribbon.
3. Click the **Query Wizard** button. A *New Query* dialog window will appear. Make sure the Simple Query Wizard option is selected, then click the OK button.
4. The Simple Query Wizard will ask us which tables and columns we want to use, as shown back in Figure 4-23. First we need the *section number* and *course number* from the *section* table. Select **Table: section** from the drop down menu in the **Tables/Queries** box,
5. Select **snum** from the *Available Fields*, then click the arrow button  to move *snum* to **Selected Fields**.
6. Do the same for **cnum**.
7. Select **Table: course** from the drop down menu in the **Tables/Queries** box, then move *title* and *credits* to **Selected Fields**.
8. Select **Table: faculty** from the drop down menu in the **Tables/Queries** box, then move *first* and *last* to **Selected Fields**.
9. Click the **Next** button.
10. The Query Wizard will ask us if we want a *detail* or *summary* query. Make sure **detail** is selected, then click the **Next** button.
11. The Query Wizard will ask us to name the query. Type the name **Teaching Assignments** in the name box. Select **Open the query to view information** from the options further down in the window, then click the **Finish** button.
12. You should now see the result set for the query, as shown in Figure 4-27. **Close the query** when you are finished viewing the data. It's not a good idea leave tables or queries open when they are not being used.



snum	cnum	title	credits	first	last
001	CIS 103	Applied Computer Technology	3	Ray	Gutekunst
002	MATH 171	Calculus I	4	Leonardo	Fibonacci
003	CIS 103	Applied Computer Technology	3	Frank	Sweeney
004	ENGL 101	English Composition	3	Daphne	Belle
005	CIS 103	Applied Computer Technology	3	Walter	Watson
006	PHIL 240	Technology and Ethics	3	Hans	Jonas
007	PHIL 240	Technology and Ethics	3	Hans	Jonas
008	ENGL 101	English Composition	3	J P	Zenger
009	ENGL 101	English Composition	3	J P	Zenger
010	ENGL 101	English Composition	3	Daphne	Belle
011	CIS 103	Applied Computer Technology	3	James	Johnson
012	CIS 103	Applied Computer Technology	3	James	Johnson
013	CIS 103	Applied Computer Technology	3	Frank	Sweeney
014	CIS 103	Applied Computer Technology	3	Frank	Sweeney

Figure 4-27: The result set for the Teaching Assignments multiple table query.

The Query Wizard can be used to create a multiple table query, but it does not allow us to enter a condition to select which rows will appear. What if we wanted the teaching assignments, but only for the CIS Department? In SQL, we would add ... FOR faculty.department = "CIS" to the end of the command. Without using SQL, we need to modify the query in the Query Design view.

Instead of modifying this query, we will copy the query and modify the copy. We will name the copy "CIS Teaching Assignments." Generally, this is a good practice. If you need a query that is similar to an existing query, then you can copy the existing query and modify it.

Notice that there are entries for the *Teaching Assignments* query in the list of items in the database under each of the three tables that are part of the query. Figure 4-28 shows this.

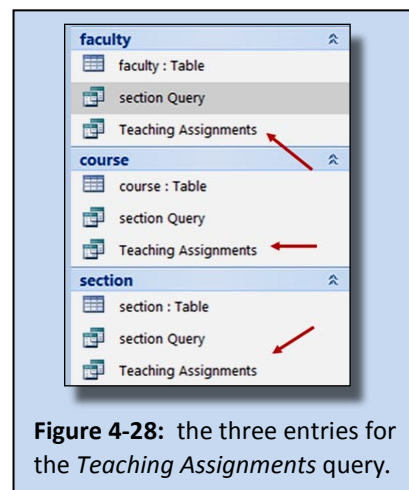


Figure 4-28: the three entries for the *Teaching Assignments* query.

All of these entries are for the same query – there is only one *Teaching Assignments* query, with multiple links to the same query. Next we will make a real copy of the query. Whenever you make a copy of an *Access* query, the new copy of the query must have a new name.

To copy an existing Access query

1. Right-click any one of the entries for the *Teaching Assignments* query, then select **Copy** from the menu that appears.
2. Right-click again any place in the list of items in the database and select **Paste** from the menu that appears.
3. A *Paste As* dialog window will appear, asking you to name the new copy of the query, as shown in Figure 4-29. Change the name to **CIS Teaching Assignments**, then click the **OK** button.

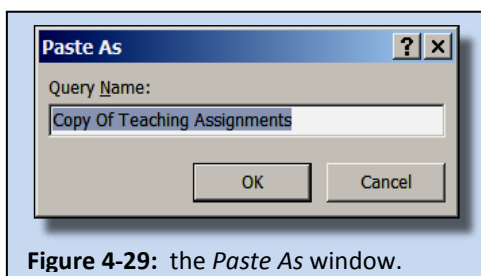


Figure 4-29: the *Paste As* window.

The list of items in the database now includes a *CIS Teaching Assignments* query, which we will edit in Design View.

To edit an Access query

1. Right-click an entry for the *CIS Teaching Assignments* query in the list of items in the database, then select **Design View** from the menu that appears to open in *Design View*.
2. We wish to add criteria equivalent to the condition *department* = "CIS". First, we need to add the *department* column from the *faculty* table to the query specifications. Click and drag the *department* column to an empty field column in the query specifications, as shown in Figure 4-30.

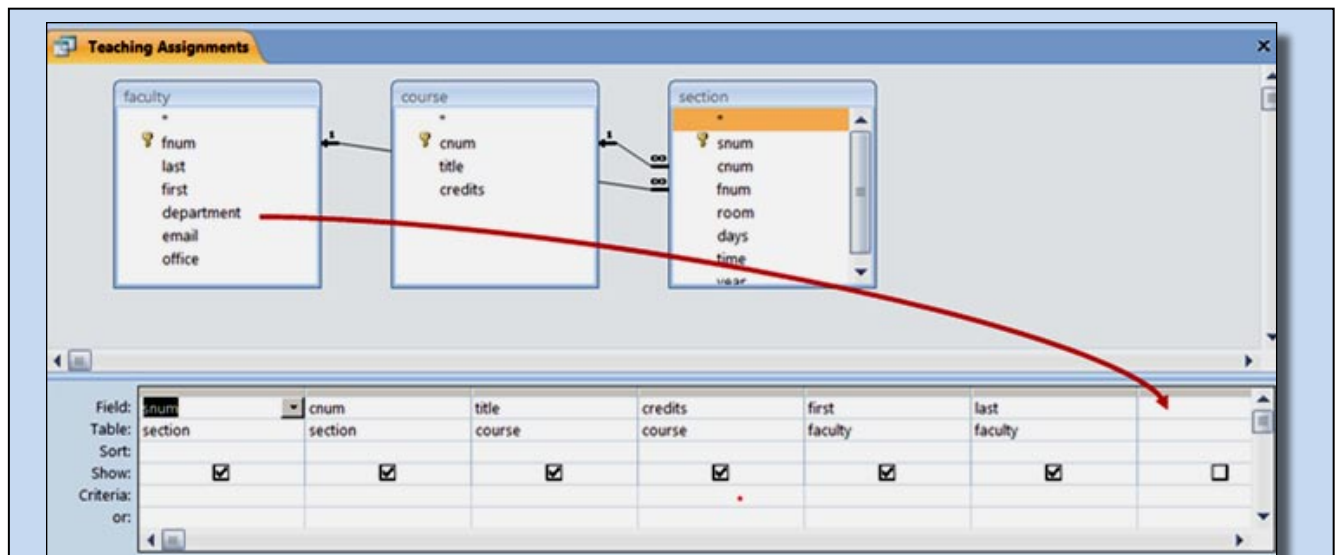


Figure 4-30: the *CIS Teaching Assignments* query in *Design View*. The arrow shows how the *department* column in the *faculty* table should be dragged and dropped in the query field specifications. *Design view* can be used to edit queries, or to create new queries without using the *Query Wizard* or *SQL*.

3. Make sure the **Show** check box is not checked, because we do not want to include the column in the result set, only use it as part of the condition.
4. Type = **"CIS"** as the criteria for *department*. The department column specifications should now look like Figure 4-31.
5. Click the **Run** button to see the result set for the query, shown in Figure 4-32. Close and save the query when you are finished viewing the data.

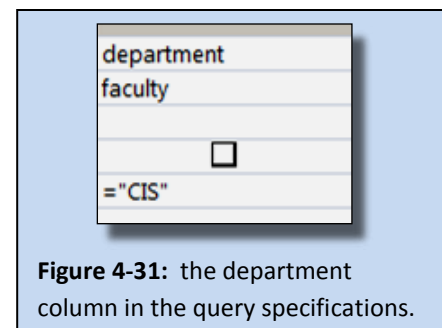


Figure 4-31: the department column in the query specifications.

snum	cnum	title	credits	First	Last
003	CIS 103	Applied Computer Technology	3	Frank	Sweeney
013	CIS 103	Applied Computer Technology	3	Frank	Sweeney
014	CIS 103	Applied Computer Technology	3	Frank	Sweeney
001	CIS 103	Applied Computer Technology	3	Ray	Gutekunst
015	CIS 103	Applied Computer Technology	3	Ray	Gutekunst
016	CIS 103	Applied Computer Technology	3	Ray	Gutekunst
005	CIS 103	Applied Computer Technology	3	Walter	Watson
018	CSCI 111	Computer Programming with Java	4	Walter	Watson
011	CIS 103	Applied Computer Technology	3	James	Johnson
012	CIS 103	Applied Computer Technology	3	James	Johnson

Figure 4-32: a result set for the *CIS Teaching Assignments* query. Notice that the CSCI course is in the CIS department.

This example has a simple condition – *department* = **"CIS"**. We just needed to enter = **"CIS"** as the criteria for the department column. However, editing criteria for queries that use AND and OR can be a little tricky. Here are a few examples:

- department = "CIS" OR department = "ENGL"
 - enter = "CIS" OR = "ENGL" as the department column criteria
- department = "CIS" AND credits = 3
 - enter = "CIS" as the department column criteria and = 3 as the credits column criteria
- department = "CIS" OR credits = 3
 - enter = "CIS" as the department column criteria and = 3 as the credits column criteria, but one line lower than the line that has the = "CIS" criteria.

4.3e Exporting Query Results to Excel

So far we have learned how to extract data from an *Access* database, but what about putting the data into a nice, neat report? You already know how to do this – in *Microsoft Excel*.

The result set from a database query can quickly be exported to *Microsoft Excel*. So, rather than learning to create reports in *Access*, which can be tedious and time consuming, we will learn to export query result sets and tables to *Excel* from *Access*. Then, you can work with the data in *Excel* – format, sort, print, and so on.

Also, knowing how to export *Access* data into an *Excel* spreadsheet will allow you to give data to other people who know how to use *Access* but not *Excel* in a format they can use. This is a very helpful skill to have in the workplace, where many people can use *Excel*, but not *Access*. In fact, If you ask for data from a database administrators in a professional environment with a large enterprise-wide database, it will most likely be given to you as an *Excel* spreadsheet, perhaps as an email attachment.

We will export the results of the *Teaching Assignments* query to *Excel*.

To send a query's result set to Excel from Access

1. Right-click an entry for the *Teaching Assignments* query in the list of items in the database.
2. Select **Export** from the menu that appears, then **Excel**, as shown in Figure 4-33. Notice that there are some other useful options on the menu, such as those for *HTML*, *XML*, *PDF* and *Word files*.

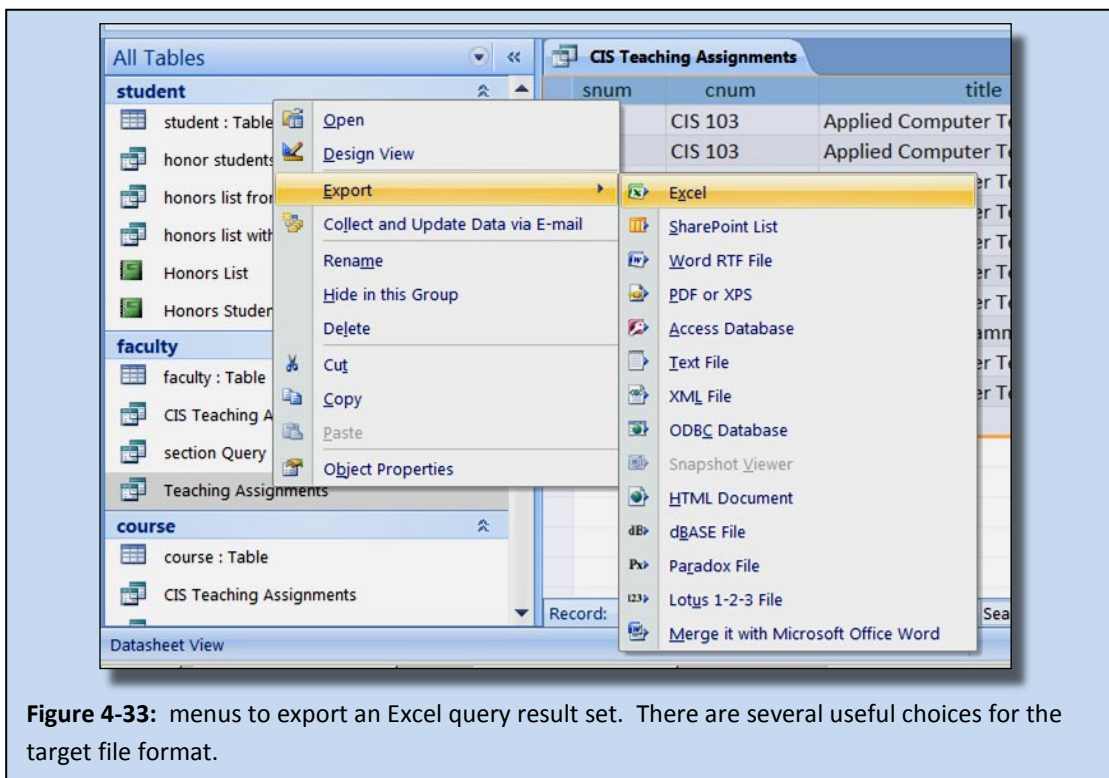


Figure 4-33: menus to export an Excel query result set. There are several useful choices for the target file format.

3. The *Export – Excel Spreadsheet* window will open. **Pay attention to the file location.** Change it as necessary. If you are not sure what to do, then ask your instructor, or, if this is not a test, ask the student sitting next to you. If the student sitting next to you doesn't know, then next time sit someplace else.
4. Leave the file format as **Excel Workbook (*.xlsx)**, unless you have good reason to change it. Make sure that the check boxes for both **Export data with formatting and layout** and **Open the destination after the export operation is complete** are both checked.
5. Click the **OK** button. The query results should open in an *Excel* spreadsheet which you can now format, print, edit, and so on, using your *Excel* skills. Be sure to go back and close *Access* if you are finished using it. A formatted copy of the this result set in *Excel* is shown back in Figure 4-26.

A table can be exported to *Excel* in the same way that a result set can be, by right clicking the table entry in the list of items in the database, then using the menus that appear.

This concludes the chapter on using database management systems. You can learn a lot more about databases by taking a database management course, or using some of the resources listed in the end of the chapter material.

4.3 Section Review

- a. What do the file extensions *accdb* and *mdb* indicate with regard to databases files? What is the difference between the datasheet view and the design view of an *Access* table? What is the difference between an *Access* query and an *Access* report?
- b. How are *SQL* queries entered in *Access*? How are existing queries run in *Access*?
- c. Describe how to create a query in the *Access Query Wizard* equivalent to the *SQL* query:

SELECT first, last, department, hours FROM payroll WHERE hours > 40;

What part of this query cannot be entered using the *Access Query Wizard*? Describe how it can be added to the query.

- d. How can a multiple-table query be created in *Access* using the *Access Query Wizard*? How can a query be copied in *Access*?
- e. Describe how to export the result set from an *Access* query as a *Microsoft Excel* spreadsheet. Why would someone want to know how to do this rather than using the *Access* report generator? What are some other formats that can be used for queries and tables exported from *Access*?

— § —

Chapter 4 – Vocabulary

Define each of the following, as used in this chapter:

.accdb	data type	metadata
.mdb	database	numeric data
attribute	database management system	primary key
Boolean conditions	enterprise-wide database	query
Boolean data	entity	relational database
Boolean logic	field	Structured Query Language
data definition language	full database name	text data
data manipulation language	instance	

Chapter 4 – Questions

End of chapter materials are in the final version. They have been removed from the draft to reduce file size.

Chapter 4 – Exercises

Chapter 4 – Research Topics

Chapter 4 – To Learn More

Books

On The Web

Other Courses