

# CHAPTER 5: SELECT QUERIES

“Logic is the art of making truth prevail.”

– L. A. Bruyère, *Characters*, 1688

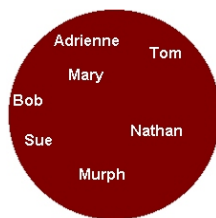
A select query extracts records from the database that meet criteria specified by a question that one poses to a database. If the table is the heart of the database, the query is its brain.

## BOOLEAN LOGIC

Let’s examine some basic concepts that are crucial to query development.

### AN EXAMPLE

A New York firm took a poll of its staff. In that poll, workers could identify which baseball teams they rooted for. Seven admitted being Mets fans. To help us envision this, their names are shown to the right.



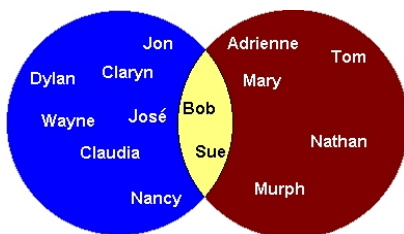
Mets Fans



Yankee Fans

Of course, not everyone was a Mets fan. The Yankees fans are represented on the left.

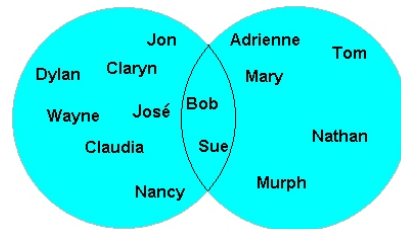
Two—Bob and Sue—inexplicably identified themselves as fans of both. We represent the entire relationship graphically:



Drawings like this, showing membership in various groups or sets, are called “Venn diagrams,” which you may remember from your dreary youth.

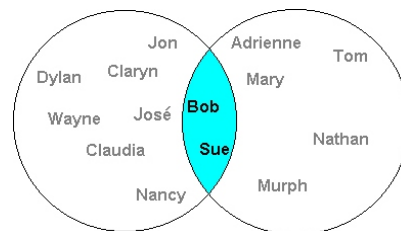
## LOGICAL OR

A logical OR refers to a situation where *one* condition is met *or another* condition is met *or both* conditions are met. For example, I might want a list of everyone in the office who is a fan of the Yankees *or* the Mets:



## LOGICAL AND

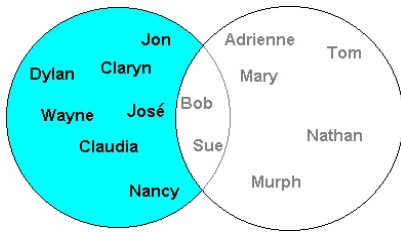
When you have two conditions and you want both met, you need a logical AND. In a logical AND, one condition must be true *and* the other condition must be true. For example, I might want a list of everyone who is a fan of the Yankees *and* the Mets:



As you can see, only Bob and Sue meet both conditions—they are fans of the Yankees *and* they are fans of the Mets—which is the outcome of a logical AND.

## LOGICAL NOT

If I wanted a list of Yankee fans, excluding anyone who harbored Met sympathies, I could use a logical NOT. Yankee fans NOT Met fans would look like this:



### CONFUSION

So far, you might think that these are simple concepts, and, indeed, they are not exactly rocket science. Things can get confusing when we concoct semantic labyrinths. For instance, if I said, “Give me a list of clients who live in New York *and* New Jersey,” what tool do you need? Despite my use of the word “and,” if you think about it in context, I am asking for a logical OR. It may help to use this mnemonic device: “Or gives you more!”

### QUERY BASICS

Queries are the *raison d'être* of a DBMS. Queries extract information from one or more related tables. The results of a query, called a “dynaset,” are displayed in the Datasheet view. A dynaset—short for “dynamic set”—always reflects the current underlying data.

This Query Datasheet view, as with Tables, can show any combination of fields in any order. As with Tables, dynaset records can be sorted in any order, and all of the tools that worked on Tables work on dynasets.

Queries can perform calculations on existing fields, and display the results. Queries can be used as sources for subsequent queries, forms, and reports.

### CREATING A QUERY

From the database window, select the Query tab, and click **New**. This will call up the New Query Dialog, seen in Figure 139.

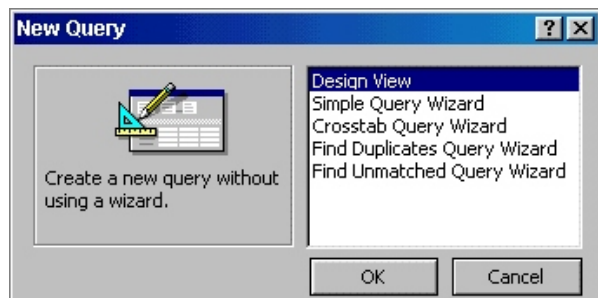


Figure 139

Select Design View and click **OK**. This opens the Show Table dialog seen in Figure 140. In this dialog, you select the Tables and or Queries from which the new Query will take its information.

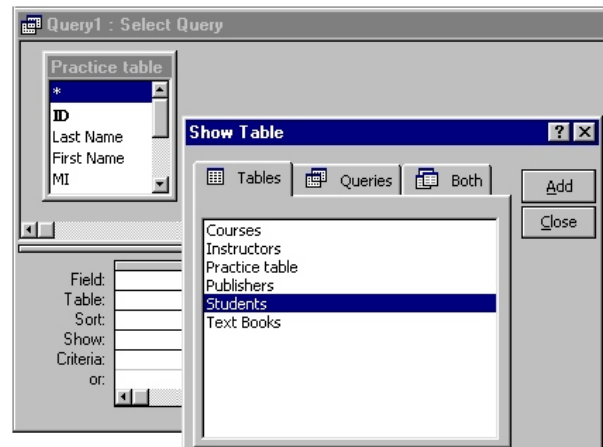


Figure 140

To add a table, select it, and click **Add**. This will place it on the upper frame of the underlying Query Design window. Note that one table has been added in Figure 140. Add additional tables the same way. When you have all the tables you need, click **Close**.

### QBE

To make it easy for you to create queries, Access supplies a Query by Example (QBE) grid. You simply add fields from Field List to the QBE grid in the order you want them to appear. For example, in Figure 141, a query has been created that will display the first names, middle initials, last names, and states of each record in the Patients table.

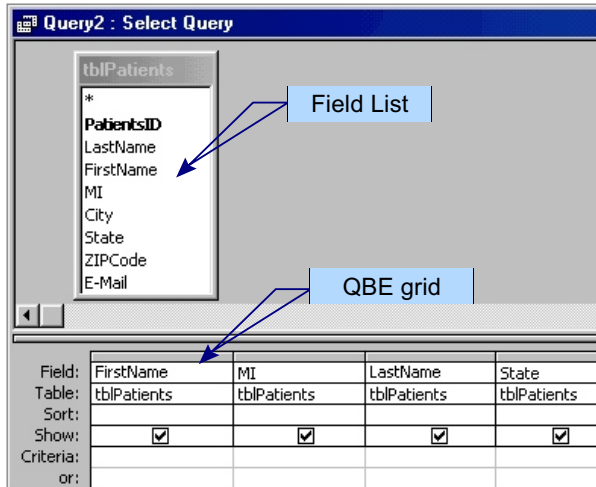


Figure 141

There are several ways to add fields to the QBE grid:

- select the field from a cell’s dropdown list;
- double-click the field in the field list; or,
- click and drag the field from the field list to a cell in the QBE grid.

Of these, the last method—click and drag—is by far the best to use. A query may contain up to 255 fields.

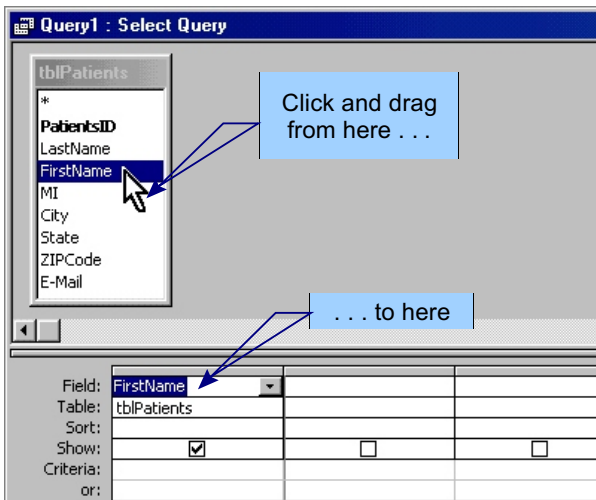


Figure 142

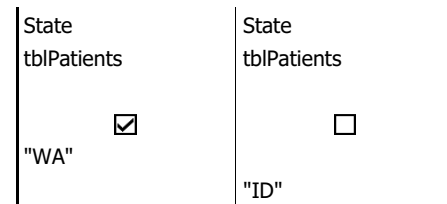
### QBE: LOGICAL OR

There are several ways to construct a query using a logical OR:

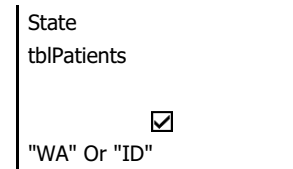
- Place the conditions on different lines in a QBE field:



- Place the conditions on different lines in two QBE fields:



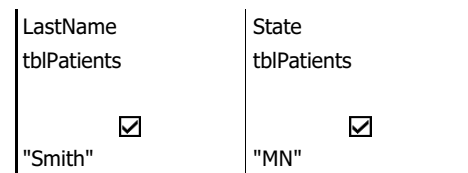
- Type the OR relationship; unlike the other methods, you must type the quotation marks that encase the criteria:



Which method should you use? It doesn’t matter. In each case you will select records whose State field is Washington or Idaho.

### QBE: LOGICAL AND

To construct a query using a logical AND, place the conditions on the same line in the QBE fields. For example, to find all of the Smiths in Minnesota:



There is also a way to type the AND relationship, although it is not as obvious as typing the OR relationship:

State	
tblPatients	<input checked="" type="checkbox"/>

- Sum
- Average
- Count
- First
- Last
- Maximum
- Minimum
- Standard deviation
- Variance

The field name must appear in square brackets:

[FieldName]

If there is more than one table available to this query, we must specify the table whence each field name comes:

[tblDemo]!

or, in its general form:

[tblDemo]![FieldNameA] = "x" And  
[tblDemo]![FieldNameB] = "y"

In plain English, this says, "Return the records for which FieldNameA from tblDemo is equal to x and for which FieldNameB from tblDemo is equal to y."

### QBE: NOT OPERATOR

To find records that don't match specific criteria requires the NOT operator. For instance, to generate a query of customers outside of Connecticut:

State	
tblPatients	
	<input checked="" type="checkbox"/>
Not "CT"	

### TOTALS QUERIES

In addition to selecting records, you can use queries to summarize data. For example, you might want to know the number of clients, the average credit limit extended, or the highest credit limit extended, state-by-state. A totals query returns summary information, not individual records. Here are your options:

You have two other options:

- "Expression" allows you to create calculated fields that use an aggregate function in its expression.
- "Where" allows you to specify criteria for a field other than the one that defines the grouping.

Consider the information in Figure 150. You would like to know how much credit you have extended, broken down by state:

**Step 1:** Start a new query as usual.

**Step 2:** Drag the State field to the first QBE column and CreditLimit to the second, as shown in Figure 150.

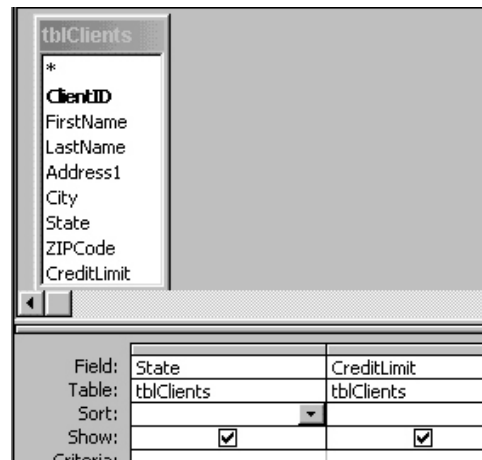



Figure 150

**Step 3:** Click the Totals icon, , on the Query Design toolbar.

- This adds a new line—Total—between Table and Sort in the QBE grid, as seen in Figure 152.

Field:	State	CreditLimit
Table:	tblClients	tblClients
Total:	Group By	Group By
Sort:		

Figure 152

**Step 4:** In the CreditLimit column of the QBE grid, expand the drop-down list in the Total row, as shown in Figure 153.

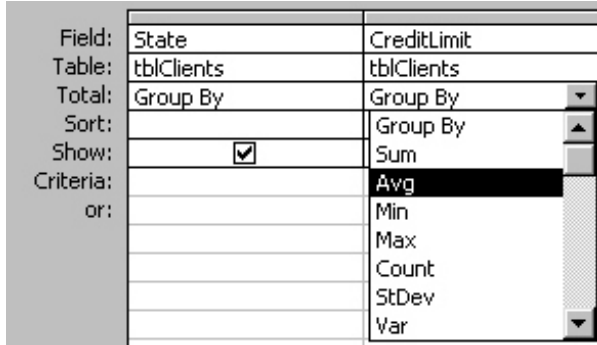


Figure 153

**Step 5:** Select Avg to return the average.

	State	AvgOfCreditLimit
	CT	\$485.71
	MA	\$475.00
▶	RI	\$500.00

Figure 154

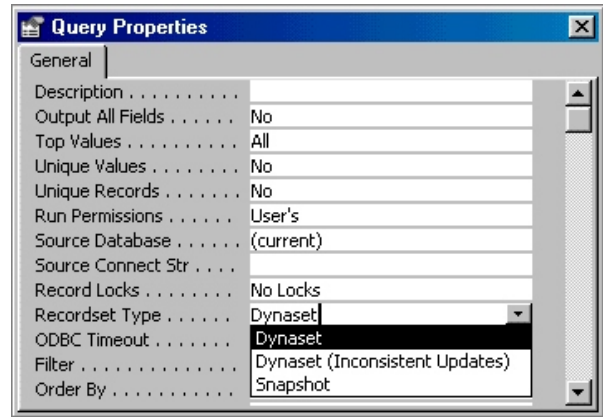


Figure 156

The Snapshot setting doesn't allow updates.

## NOTES


## UPDATING RECORDS

When you change data in a form, it ripples back through the query to the underlying table. If your query doesn't allow updates, check its recordset type:

**Step 1:** From the Database window, select the query you want to edit.

**Step 2:** Press **Ctrl + Enter** to open the selected query in Design view.

**Step 3:** To open the Properties Sheet, select the field whose properties you want to adjust, and either:

- press **Alt + Enter**;
- right-click the object, and select **P**roperties from the pop-up menu; or,
- click the Properties icon, , on the Form Design tool bar.

**Step 4:** Make sure the Recordset Type property is Dynaset or Dynaset (Inconsistent Updates).



## LAB 4: CREATING SELECT QUERIES

In this lab, we will work on creating several simple select queries, using logical ANDs, ORs, and NOTs.

### CREATE A SELECT QUERY

A simple query is a question that is asked of one table. A compound query is one that is asked of more than one table. We will deal with complex queries later. To construct a simple select query:

**Step 1:** From the database window, seen in Figure 157, select Create query in Design view, and press .

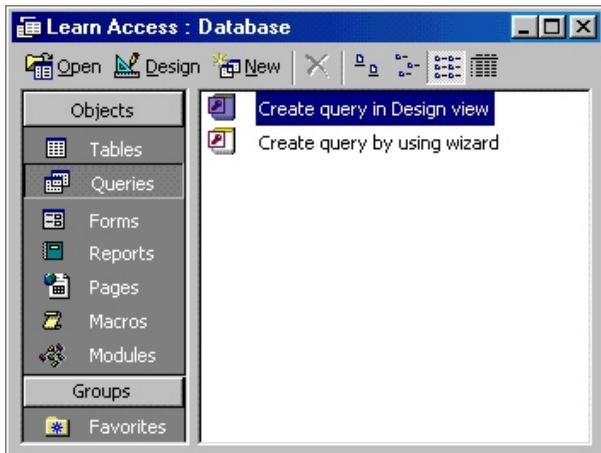


Figure 157

**Step 2:** Select the table that you want to query, as shown in Figure 158, and click .

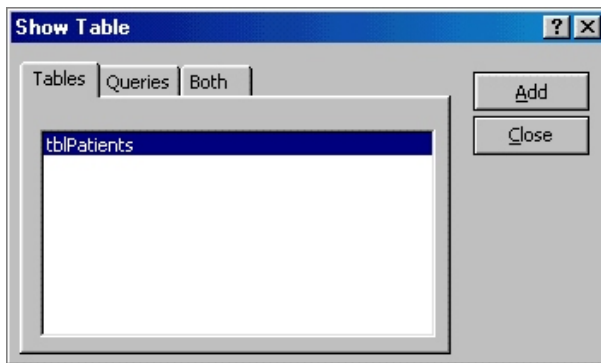


Figure 158

**Step 3:** Click .

**Step 4:** Resize the field list and/or the frame, as shown in Figure 159.

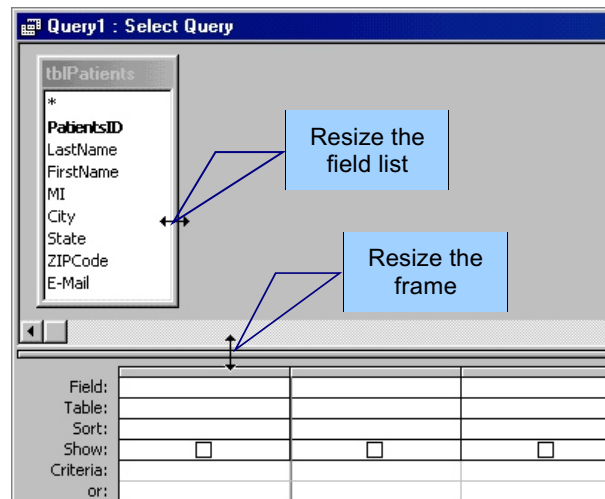


Figure 159

**Step 5:** Click and drag the FirstName field from the Field list to the QBE grid, as shown in Figure 160.

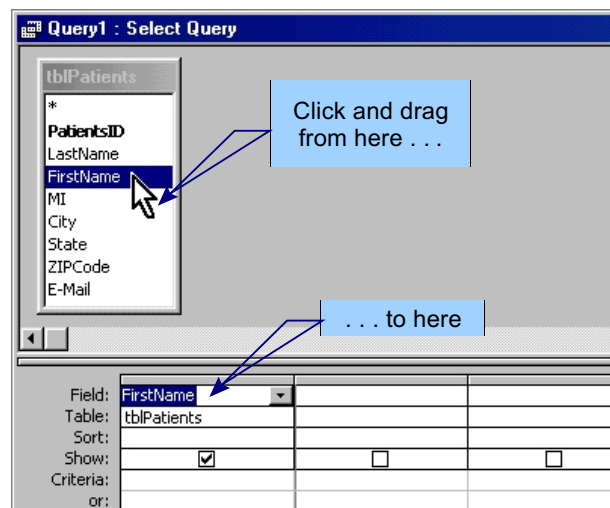


Figure 160

**Step 6:** In similar fashion, add the LastName, MI, and State fields, as shown in Figure 161.

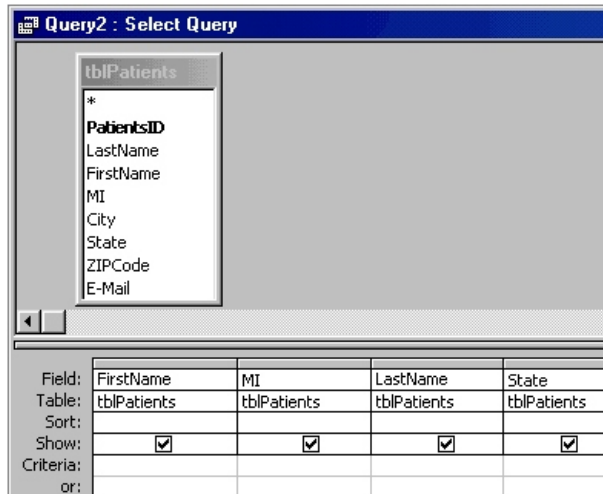



Figure 161

## RUN THE QUERY


The query we have just constructed asks this question, in effect: “Show me the last name, first name, middle initial, and state of each of my patients.” To run this query—that is, to see the results—click the Run icon, , on the Query Design toolbar.

The results look just like the Datasheet view of a table, except that it only displays the fields you dragged to the QBE grid. The same tools that you used on tables—Find, Sort, and so on, as well as the table formatting tools—can be used in the results of a query, as well.



If you edit data in a query’s dynaset, the changes will ripple back and be stored in the underlying table. Furthermore, the properties of the underlying table limit what you can do to a query’s dynaset.

For example, if you set a field’s Required property to Yes, but in a subsequent query’s dynaset, you leave that field blank, you will get an error message. This problem is commonly caused when the required field was not dragged into the QBE grid—so it doesn’t appear in the dynaset, thus preventing you from entering required data.

## RETURN TO DESIGN VIEW

To return to the Query Design view, click the Design view icon, .

## SORTING RECORDS

We could sort the results of the query by clicking the ascending or descending order icons,  , but we can do better than that. We can make the query results sort themselves. To sort the results of a Select Query, specify the sort order in the Sort row under the field by which you are sorting.

For example, to sort the results of our simple query by last name:

**Step 1:** Go to the Sort row, under the LastName field as shown in Figure 165, and either:

- press **A** for “Ascending” (or **D** for “Descending”); or,
- expand the drop-down list and select “Ascending” (or “Descending”).

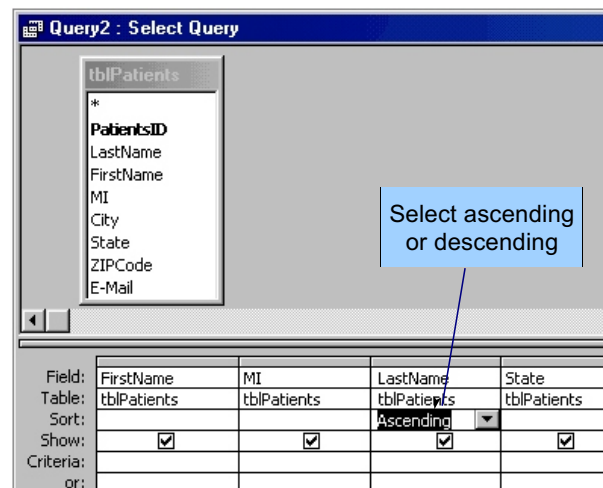




Figure 165

**Step 2:** To see the results of this query, click the Run icon, .

- You should see all of your patients, alphabetized by the LastName field.
- Upon closer inspection, though, it’s clear that we need to refine this. We would prefer names to be listed in “telephone book style”—alphabetized by

last name, and then, if there are any ties, by first name, and then by middle initial.



That way, Dewey Duck would appear before Donald Duck, which would appear before Huey, which would appear before Louis A, which would appear before Louis B, and so on.

**Step 3:** Click  to return to the Query Design view.



### FALSE STEP

Let's try doing it wrong first:

**Step 1:** Go to the Sort row, under the FirstName field as shown in Figure 168, and

-  press **A** for “Ascending”; or,
-  expand the drop-down list and select “Ascending.”

**Step 2:** Go to the Sort row, under the MI field as shown in Figure 168, and

-  press **A** for “Ascending”; or,
-  expand the drop-down list and select “Ascending.”

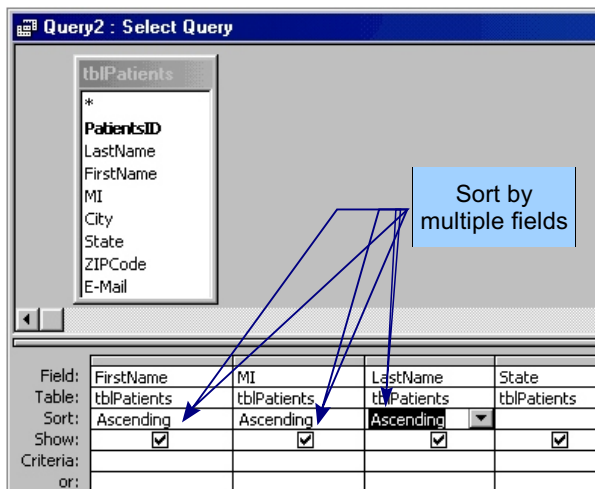



Figure 168

**Step 3:** To see the results of this query, click the Run icon,



- You should see all of your patients, alphabetized —by the FirstName field!

**Step 4:** Click  to return to the Query Design view.

### EXPLANATION

When Access performs a sort, it reads from left to right in the QBE grid. Thus, in Figure 168, it sorts first by the FirstName field. Then if there are any ties, it sorts by the MI field, and, if there are any records with the same FirstName and MI fields, only then will it sort by the LastName field.

### REORDER THE FIELDS


One solution would be to simply reorder the fields:

**Step 1:** In the Design view, click once on the FirstName header in the QBE grid.


**Step 2:** Click it again, and drag it to the right of the LastName header.

**Step 3:** Click once on the MI header.

**Step 4:** Click it again, and drag it to the right of the FirstName header.

**Step 5:** Click the Run icon, .

- You should see all of your patients, alphabetized in telephone book style.



**Step 6:** Click  to return to the Query Design view.

But what if I wanted the results to appear with the first name first?

### SORTING BY MULTIPLE FIELDS

The proper way to sort by multiple fields is:

**Step 1:** In Design View, eliminate the sorting under FirstName and LastName:

-  select the word “Ascending” and press **Delete**; or,
-  expand the drop-down box and select (not sorted).

**Step 2:** Add a second FirstName field to the right of the existing LastName field.

**Step 3:** Add a second MI field to the right of the newly created FirstName field, as shown in Figure 173.

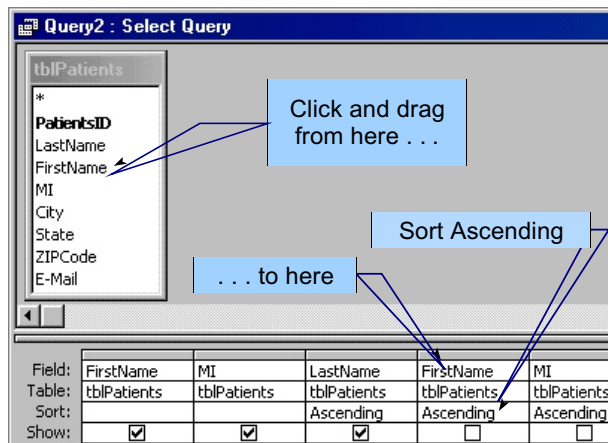


Figure 173

**Step 4:** Sort the second instance of the FirstName field in Ascending order.

**Step 5:** Sort the second instance of the MI field in Ascending order.

**Step 6:** Deselect the check box in the Show line, under the second FirstName and MI fields.

**Step 7:** Click the Run icon,

- You should see all of your patients, alphabetized in telephone book style.

**Step 8:** Click to return to the Query Design view.

Before preceding, make sure that you understand how this works. Reexamine Figure 173 closely. Remember that Access sorts selected fields from left to right, as they appear in the QBE grid. Thus, the LastName field is sorted first, followed—if there is a tie—by the FirstName field, and, finally, by the MI field.

By deselecting the Show check boxes, Access will not display the second instances of the FirstName and MI fields. This is an important idea: you need not display a field in order to sort by it.

## SAVE THE QUERY

When you close the query, Access will prompt you to save it. Remembering the Leszynski Naming Convention, name it:

**qrySortByNames**

## SELECTING RECORDS

If we create a query without specifying selection criteria, it will display the selected fields for all of the records. We usually use queries to select just the records that match certain criteria. For example, to select only those patients from New York:

**Step 1:** Add NY to the criteria line, under the State field, as shown in Figure 176:

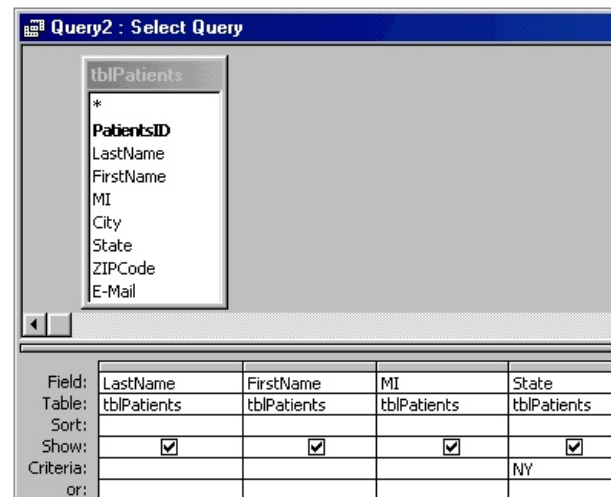


Figure 176

- If you click outside that cell, you will see that Access automatically adds double-quotes on either side of the string, changing NY to "NY".

**Step 2:** To run this query, click the Run icon, , or the View icon,

- You should see only those patients from New York.

**Step 3:** Click to return to the Query Design view.

## USING LOGICAL OR

To locate patients who lived in New York or Connecticut, use a logical OR. As we learned in the last chapter, a logical OR will return those who live in New York, plus those who live in Connecticut, plus those who live in both. To specify a logical OR in Access, place each criteria on a separate line.

To locate only those patients who live in New York or Connecticut:

**Step 1:** In the Query Design view, add CT to the line below "NY" as shown in Figure 180.

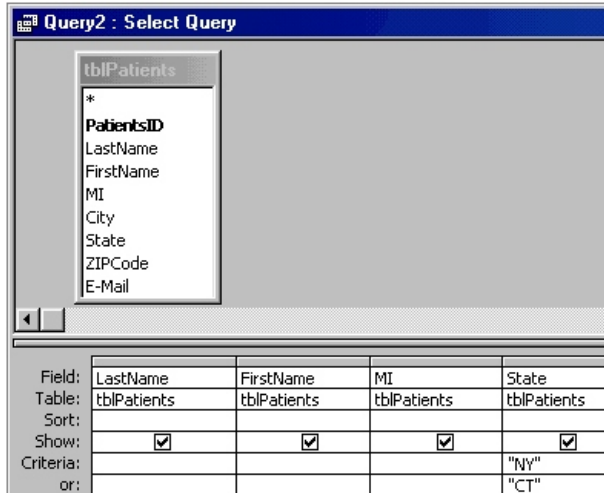





Figure 180

**Step 2:** To run this query, click the Run icon, , or the View icon, .

- You should see only those patients from New York or Connecticut.

**Step 3:** Click  to return to the Query Design view.

**USING LOGICAL AND**

As we learned in the last chapter, a logical AND returns only the values that satisfy *both* conditions. In our example, we allowed people to live in one state, so the set of people who live in Maine AND Iowa would be null.

If I wanted to locate only those patients named Smith who lived in New York, I would use a logical AND. Their last names must be Smith *and* they must live in New York. To specify a logical AND in Access, place each criteria on the same line.

To locate those patients named Smith who live in New York:

**Step 1:** In the Query Design view, add Smith in the LastName field, on the same line as "NY" in the State field, as shown in ?.

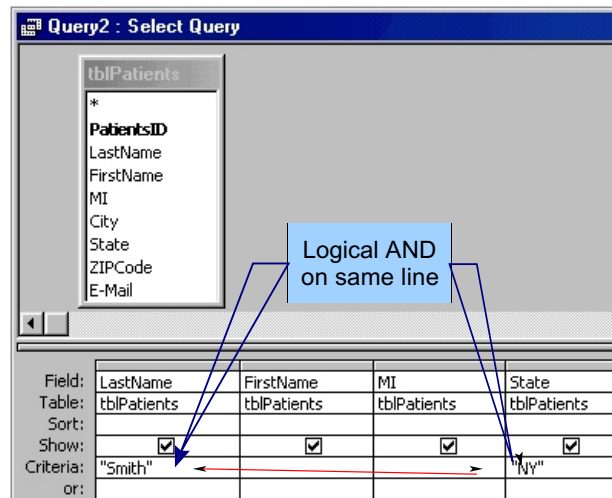





Figure 184

**Step 2:** To run this query, click the Run icon, , or the View icon, .

- You should see only those patients from New York who are named Smith.
- How many records were returned?

**Step 3:** Click  to return to the Query Design view.

Now let's continue this experiment by moving Smith from the same line to its own line. As you know, this produces a logical OR, and, as we learned earlier, OR produces more:

**Step 4:** Delete Smith from the same line, and place it one line lower, as shown in Figure 189.

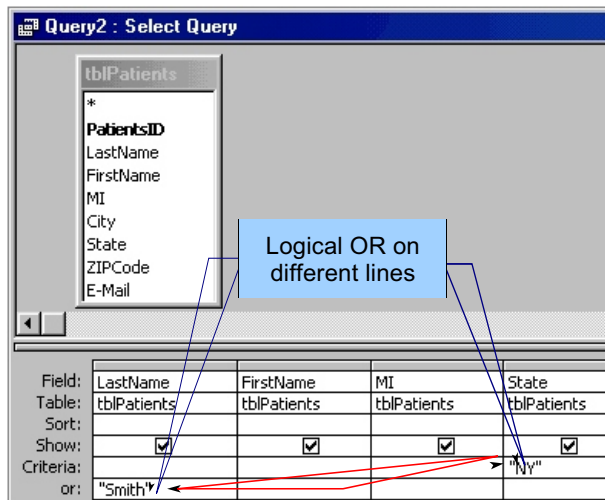


Figure 188

- You should see everyone from New York, regardless of name, in addition to everyone named Smith, regardless of their state.
- As you can see OR returned more than the logical AND did in our last example.

## NOTES