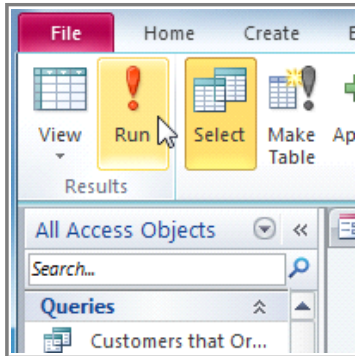


Introduction



The real power of a relational database is in the ability to quickly **retrieve** and **analyze** your data by running a query. **Queries** allow you to **pull information** from one or more tables based on a set of search conditions you define.

In this lesson, you will learn how to create a simple **one-table query**. Then you will learn how to plan and run a slightly more complex **multi-table** query.

As we show you how to create queries, we'll be using our sample database. If you would like to follow along, [download our example](#) and use it to follow the procedures demonstrated in this lesson.

What are Queries?

Queries are a way of **searching** for and **compiling** data from one or more tables. Running a query is like asking a detailed **question** of your database. When you build a query in Access, you are **defining specific search conditions** to find exactly the data you want.

Video: Creating a Simple Query in Access 2010



Watch the video (4:32). [Need help?](#)

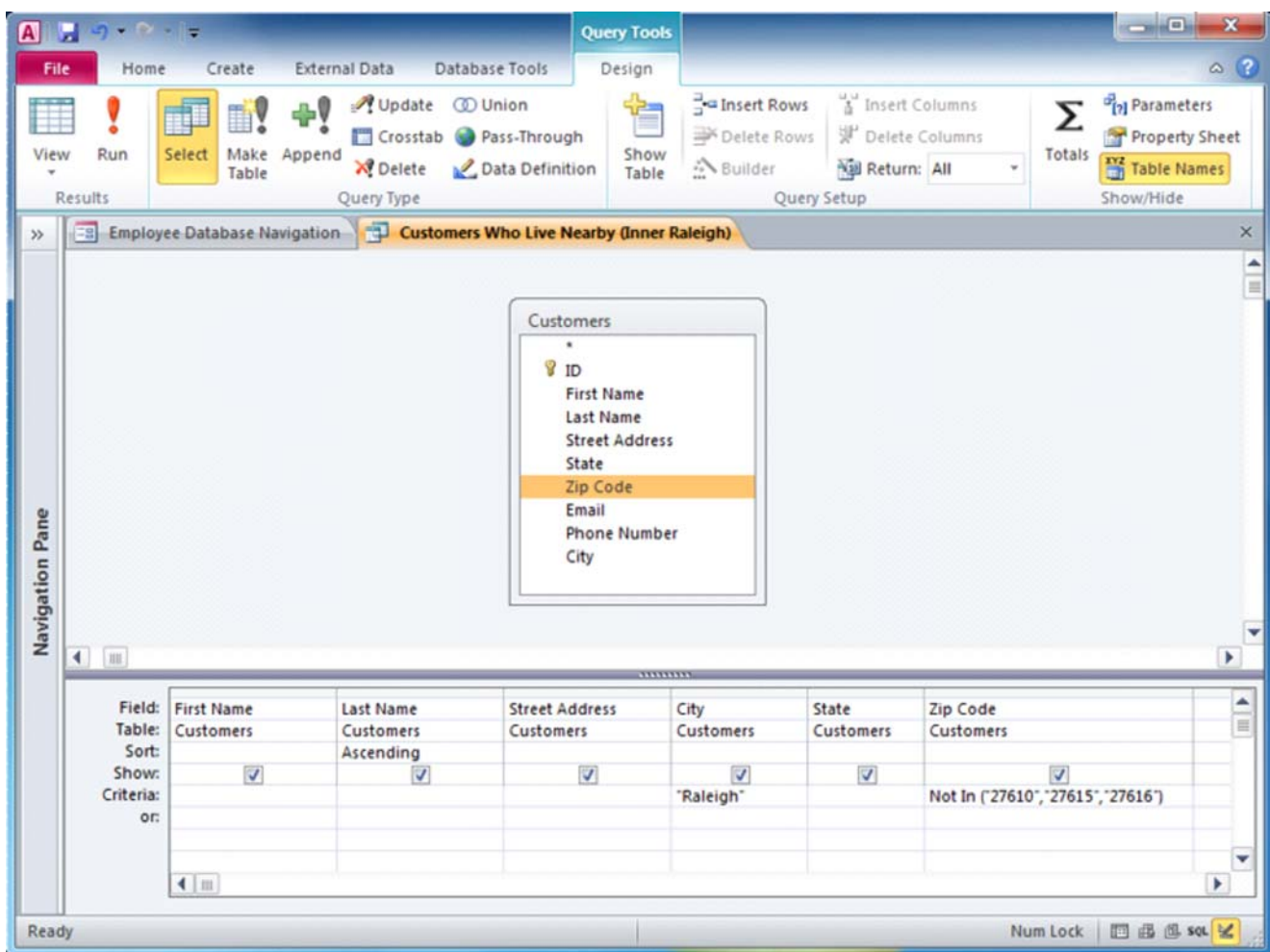
How are Queries Used?

Queries are far more powerful than the simple searches or filters you might use to find data within a table. This is because queries can draw their information from **multiple** tables. For example, while you could use a **search** in

the customers table to find the name of one customer at your business or a **filter** on the orders table to view only orders placed within the past week, neither of those would let you view both customers and orders at once. However, you could easily run a **query** to find the name and phone number of every customer who's made a purchase within the past week. A well-designed query can give information that you might not be able to find out just by examining the data in your tables.

When you run a query, the results are presented to you in a table, but when you design one, you use a very different view. This is called **Query Design view**, and it lets you see how your query is put together.

➡➡➡ Click the buttons in the interactive below to learn how to navigate the **Query Design view**.



One-Table Queries

Let's familiarize ourselves with the query-building process by building the **simplest** query possible: a one-table

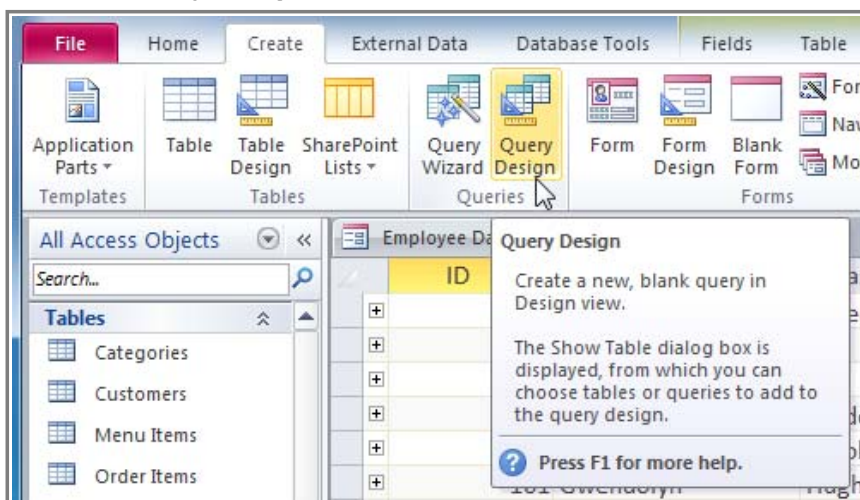
query.

We will run a query on the **Customers** table of our bakery database. Imagine that our bakery is having a special event, and we want to invite our customers who live nearby, since they are the most likely to come. This means we need to see a list of all the customers who live close by, and **only** those customers.

If you think this sounds a little like applying a filter, you're right. A one-table query is actually just an **advanced filter** applied to a table.

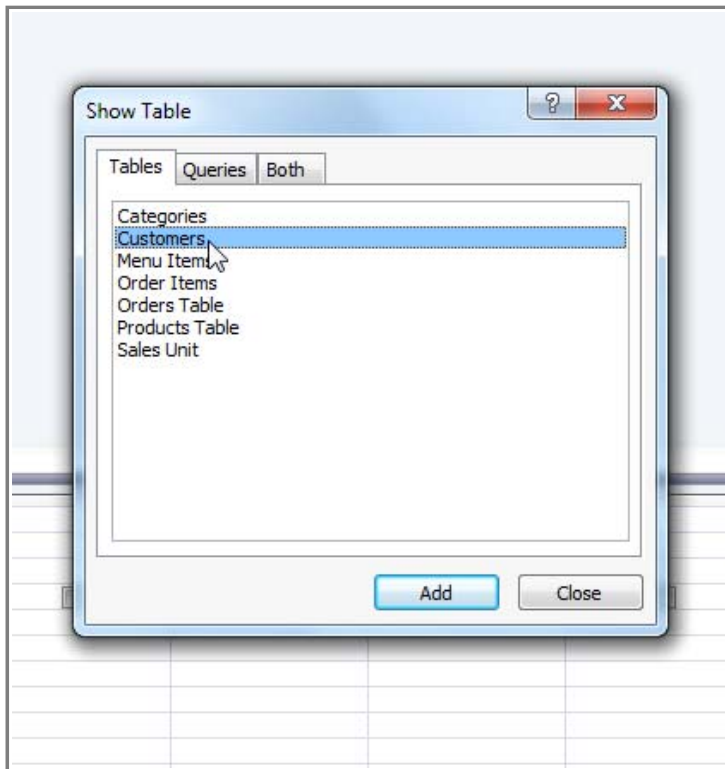
To Apply a Simple One-Table Query:

1. Select the **Create** tab on the Ribbon and locate the **Queries** group.
2. Select the **Query Design** command.



The Query Design Command

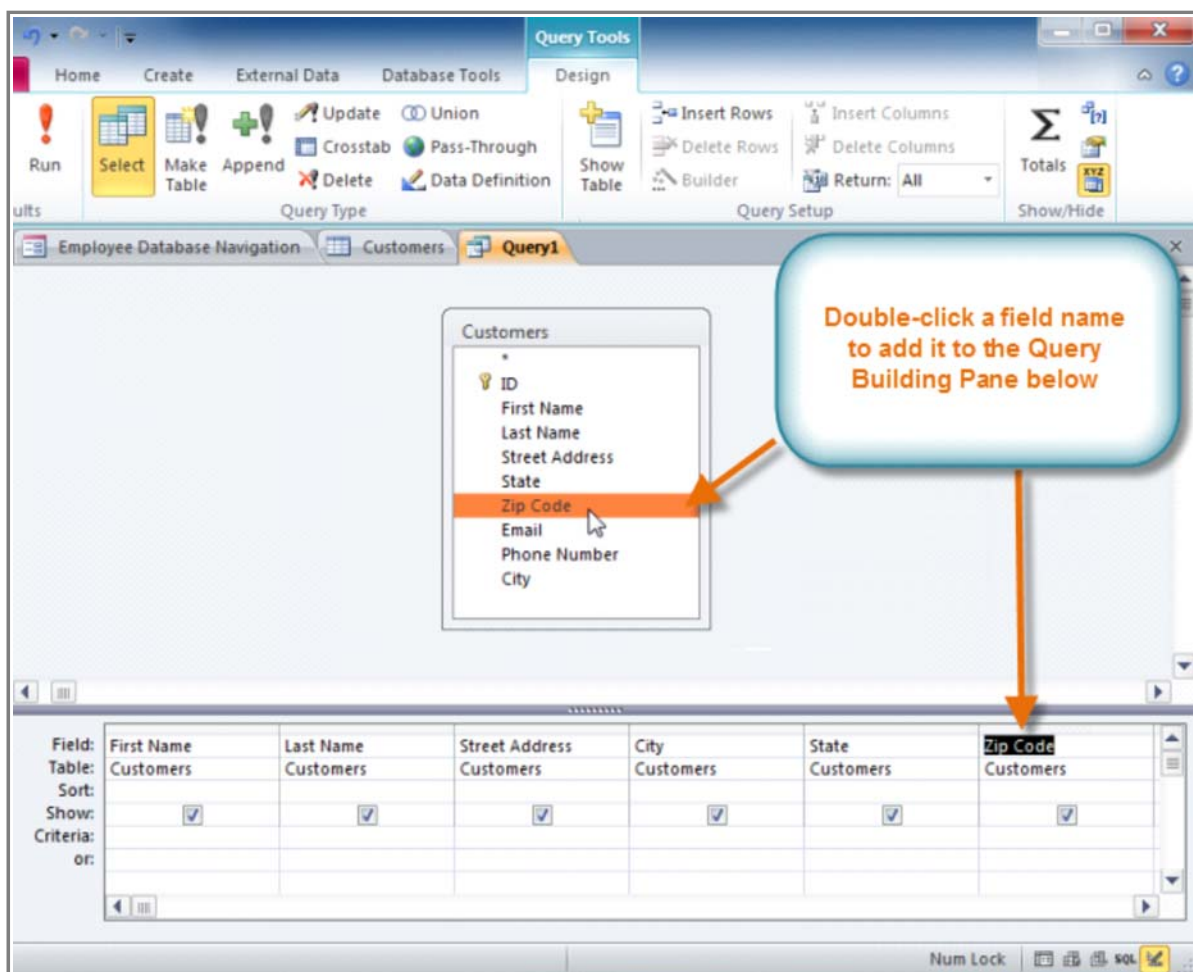
3. Access will switch to **Query Design view**. In the **Show Table** dialog box that appears, select the table you would like to run a query on. Click **Add**, then click **Close**. We are running a query about our customers, so we will add the **Customers** table.



Selecting a table to use in the query

4. The selected table will appear as a small window in the **Object Relationship Pane**. In the table window, double-click the **field names** you would like to include in your query. They will be added to the **Design Grid** in the bottom part of the screen.

In our example, we want to mail invitations to customers who live in a certain area, so we'll include the **first** and **last name**, **street address**, **city**, **state**, and **zip code** fields. We aren't planning on calling or emailing our customers, so we don't have to include the **telephone** or **email** fields.



Selecting fields to add to the query

5. Set the **search criteria** by clicking on the cell in the **Criteria: row** of each **field** you would like to filter. Typing criteria into more than one field in the Criteria: row will set your query to include only results that meet all the criteria. If you want to set multiple criteria, but don't need the records shown in your results to meet them all, type the first criteria in the Criteria: row and additional criteria in the **Or: row** and the rows beneath it.

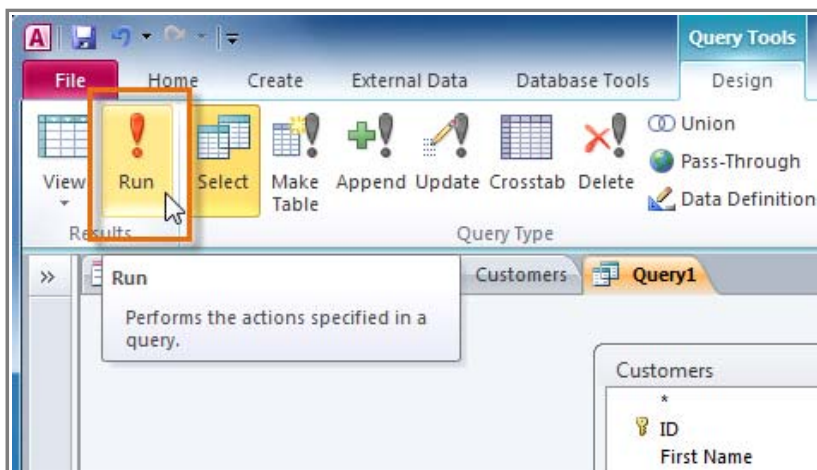
For this one-table query, we'll use very simple search criteria.

- We want to find our customers who live in a city called **Raleigh**, so in our **City** field, we'll type "**Raleigh**." Typing "Raleigh" in **quotation marks** will retrieve all records with an **exact match** for "Raleigh" in the City field.
- Some customers who live in the suburbs live fairly close by, and we'd like to invite them as well. We'll add their **zip code, 27513** as another criteria. Since we want to find customers who either live in Raleigh **or** the 27513 zip code, we'll type "27513" in the **or: row** of the **Zip Code** field.

Field:	City	State	Zip Code
Table:	Customers	Customers	Customers
Sort:			
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Criteria:	"Raleigh"		"27513"
or:			

Setting the search criteria so that the query will find records with either "Raleigh" in the City field or "27513" in the Zip Code field.

- After you have set your criteria, **run** the query by clicking the **Run** command on the **Query Tools Design** tab.



The Run Query command

- The query results will be displayed in the query's **Datasheet View**, which looks like a table. If desired, **save** your query by clicking the **Save** command in the Quick Access Toolbar. When prompted to name it, type in the desired name and click **OK**.

Sort & Filter		Records	Find
Navigation Query1			
Last Name	Street Address	City	State
Beckham	7 East Walker Dr.	Raleigh	NC
Newkirk	47 Hillsborough St.	Raleigh	NC
Jones	23 Solo Ln	Raleigh	NC
Hall			NC
Clayton			NC
Joslin			NC
Allen			NC
Hill			NC
Hayes			NC
Gibson	5 West St.	Raleigh	NC
Love	7825 Venice Ct.	Raleigh	NC
Freeman	78-A Meadowview Ln.	Raleigh	NC
Jameson	29 North Luke Ct.	Raleigh	NC
Williams	123 Garden Plow Way	Raleigh	NC
Thomas	127 South Pejulup Ln.	Raleigh	NC
Rinder	124 Heuristic Way	Raleigh	NC

Save As

Query Name:

Nearby Customers

OK Cancel

Naming the new query to save it

Designing a Multi-Table Query

Queries can be hard to understand and build if you don't have a good idea of what you're trying to find and how to find it. A one-table query can be simple enough to make up as you go along,

Video: Multi-Table Queries in Access 2010 Part 1



Watch the video (4:58). [Need help?](#)

but to build anything more powerful, you'll need to plan the query in advance.


Planning a Query

Video: Multi-Table Queries in

When planning a query that uses more than one table, you should go through these four steps:

1. **Pinpoint** exactly what you want to know. If you could ask your database any question, what would it be? Building a query is more complicated than just asking a question, but knowing precisely what question you want to answer is essential to building a useful query.
2. **Identify** every type of information you want included in your query results. Which fields contain this information?
3. **Locate** the fields you want to include in your query. Which tables are they contained in?
4. **Determine** the criteria the information in each field needs to meet. Think about the question you asked in the first step. Which fields do you need to search for specific information? What information are you looking for? How will you search for it?

Access 2010 Part 2



Every order is linked to one customer

Not all customer names are linked to an order

Watch the video (3:16). [Need help?](#)

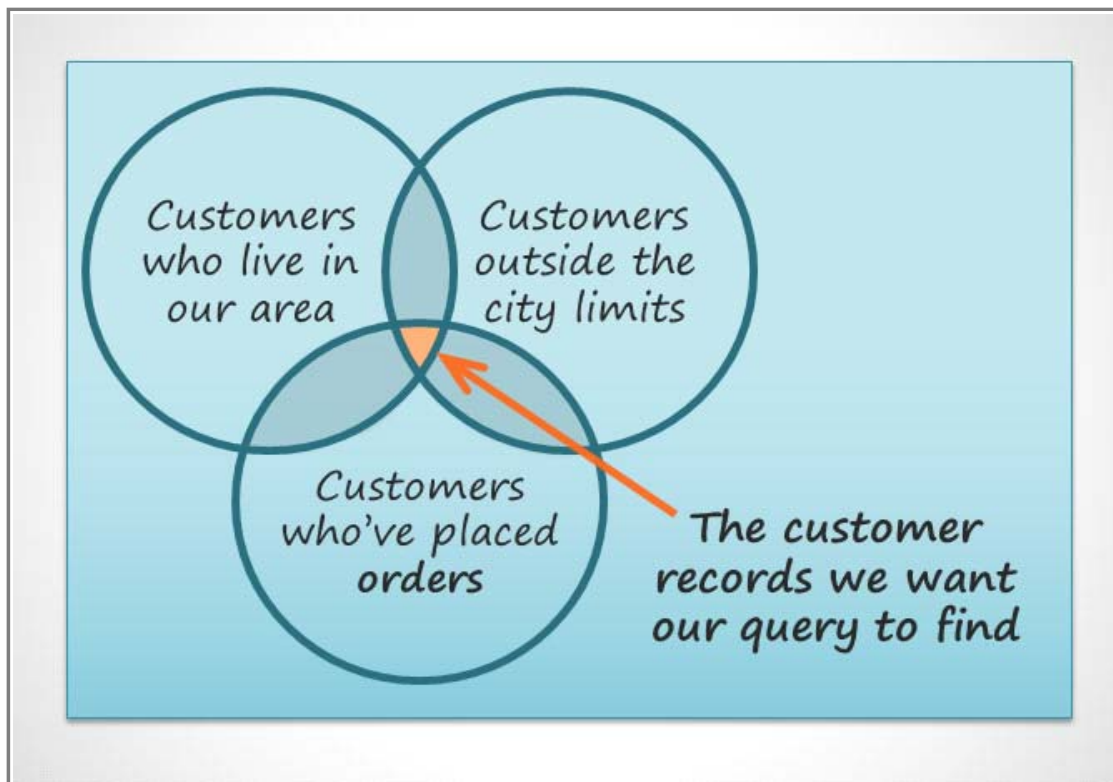
This process might seem abstract at first, but as we go through the process of planning our own multi-table query, you should start to understand how planning your queries can make building them a lot easier.

Planning Our Query

Let's go through this planning process with a query we'll run on our bakery database. As you read through the planning process step-by-step, think about how each part of the planning process could apply to other queries you might run.

Pinpointing the Question We Want to Ask

Our bakery database contains many customers, some of whom have never placed an order, but who are in our database because they signed up for our mailing list. Most of them live within the city limits, but others live out of town or even out of state! We want to get our out-of-town customers who've placed orders in the past to come back and give us another try, so we're going to mail them some coupons. We don't actually want our list to include customers who live *too* far away—sending a coupon to someone who doesn't live in our area probably won't make them come in. So really, we just want to find people who don't live in our city, but still live in our area.



Identifying the data we want the query to find

In short, the question we want our query to answer is this: **Which customers live in our area, are outside the city limits, and have placed an order at our bakery?**

Identifying the Information We Need

What information might we want to see in a list of information about these customers? Obviously, we'll need the **customers' names** and their **contact information**—their **addresses**, **phone numbers**, and **email addresses**. But how are we going to know if they've placed orders? Each record of an order identifies the customer who placed that order. If we include the **order ID numbers**, we should be able to narrow our list down to only customers who have previously placed orders.



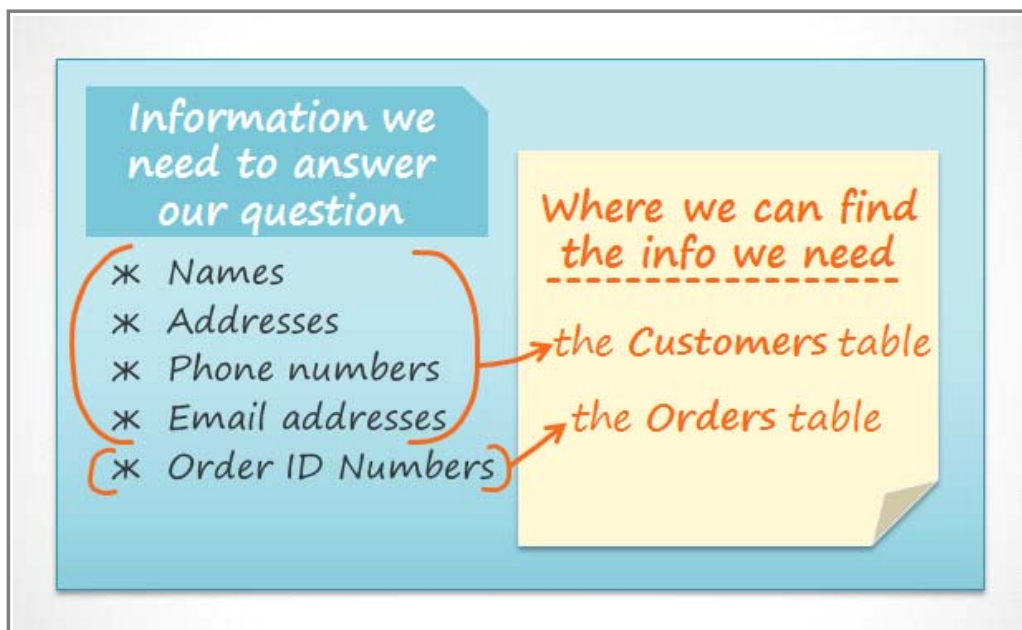
Information we need to answer our question

- * Names
- * Addresses
- * Phone numbers
- * Email addresses
- * Order ID Numbers

Making a list of the information we want the query to find

Locating the Tables that Contain the Information We Need

In order to write a query, you need to be pretty familiar with the different tables in your database. From working extensively with our own database, we know that the customer information we need is located in fields in the **Customers** table. Our **Order ID numbers** are in a field in the **Orders** table. We only need to include these two tables to find all of the information we need.



Information we need to answer our question

- * Names
- * Addresses
- * Phone numbers
- * Email addresses
- * Order ID Numbers

Where we can find the info we need

- the Customers table
- the Orders table

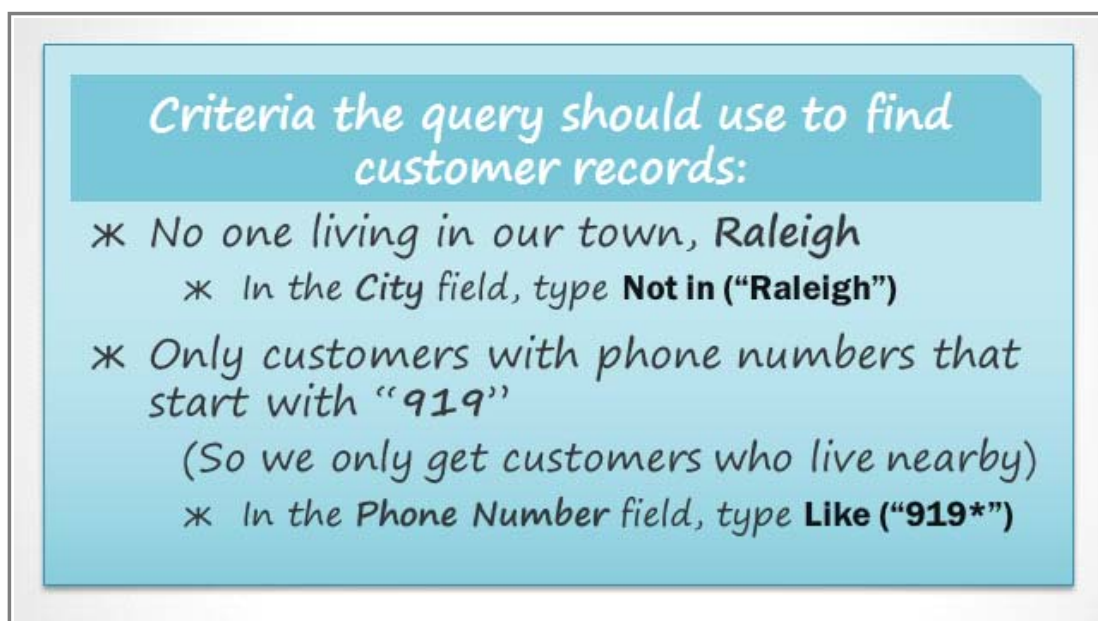
Making a list of the tables where we can find the information we need for our query

Determining the Criteria our Query Should Search For

When you set criteria for a field in a query, you are basically applying a filter to it that tells the query to retrieve only information that matches your criteria. Review the list of fields we are including in this query. How and where can we set criteria that will best help us answer our question?

We don't want customers who live in our town, Raleigh, so we want a criteria that will return all records **except** those with "Raleigh" in the city field. We don't want customers who live too far away, either. All the phone numbers in the area start with the area code "919," so we'll also include a criteria that will only return records whose entries from the **phone number field** begin with "919." This should guarantee that we'll only send coupons to customers who live close enough to actually come back and use them.

We won't set a criteria for the order ID field or any other fields, since we want to see **all** the orders made by people who meet the two criteria we just set.

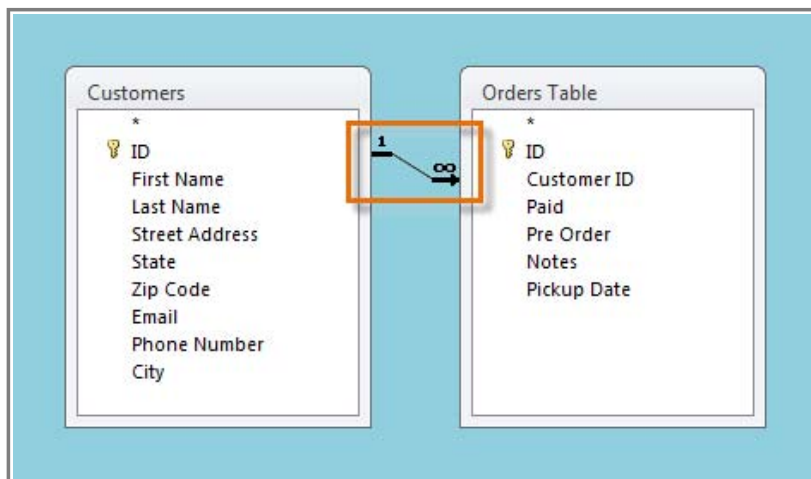


Figuring out the criteria we will use to build our query

To write queries, you'll need to be able to set criteria in a language that Access **understands**. As you can see in the image above, our criteria requiring phone numbers to begin with "919" must be typed like this: **Like ("919*")**. To learn how to write additional criteria, consult our printable [Query Criteria Quick Reference Guide](#) in the **Extras** section of this tutorial. The guide includes a number of the most common criteria used in Access queries.

Joining Tables in Queries

The final thing you need to consider when designing a query is the way you link, or **join**, the tables you're working with. When you add two tables to an Access query, this is what you'll see in the **Object Relationship Pane**:

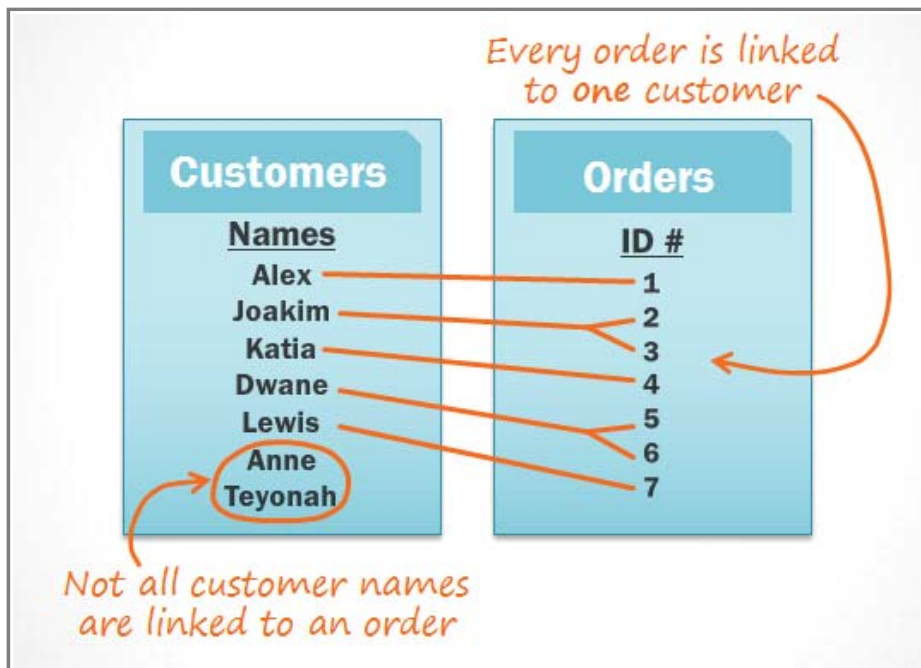


Joined tables in the Object Relationship Pane

The line connecting the two tables is called the **join line**. See how the join line is actually an arrow? This is because it indicates the order in which the query looks at data from the two tables. In the image above, the arrow is pointing from **left** to **right**, which means that the query will look at data in the **left** table first, then look at only the data in the **right** table that **relates** to the records it's already seen in the left table.

Your tables won't always be joined this way—sometimes Access will join them **right** to **left**. In either case, you might need to **change the direction** of the join to make sure your query includes the correct information. The join direction can affect **which information** your query **retrieves**.

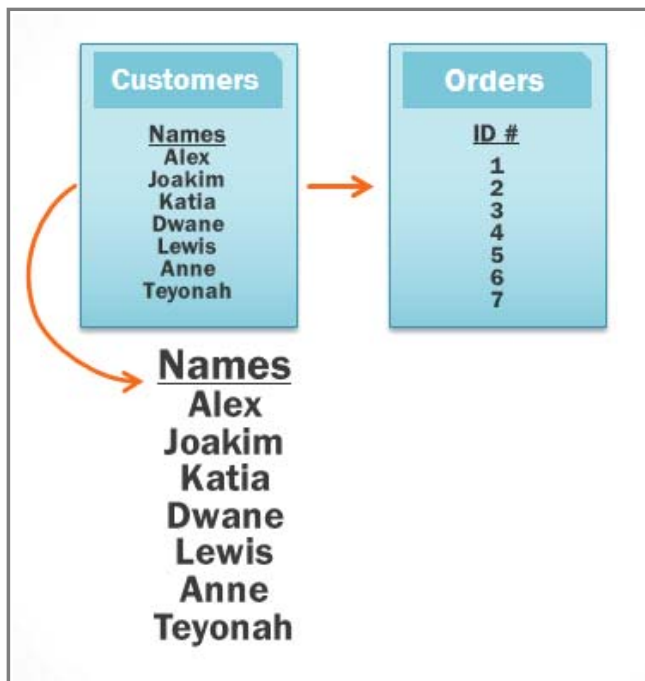
To understand what this means, consider the query we're designing. For our query, we need to see customers who have placed orders, so we've included the **Customers** table and the **Orders** table. Let's take a look at some of the data contained in those tables.



Related data stored in the Customers and Orders tables

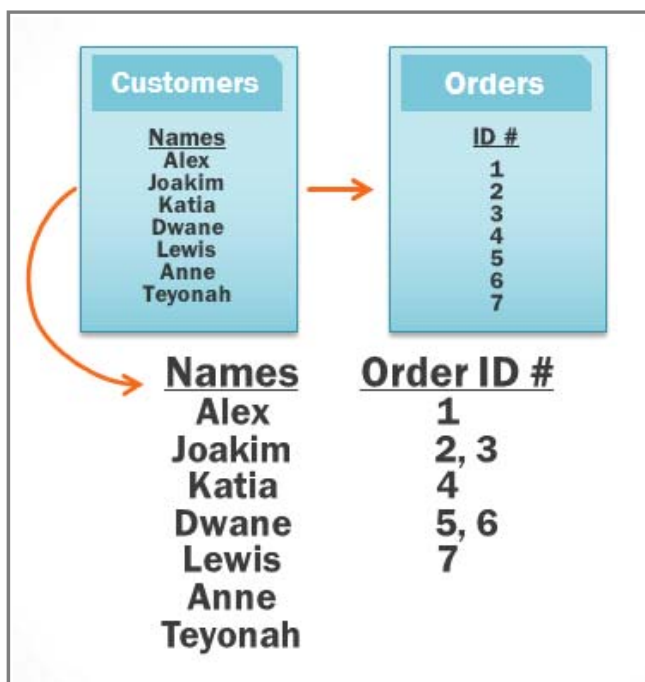
What do you notice when you look at these lists? Well, first of all, every single order in the **Orders** table is linked to someone in the **Customers** table—the customer who placed that order. However, when you look at the Customers table, you'll see that the customers who've placed multiple orders are linked to more than one order, and those who've never placed an order are linked to no orders at all. As you can see, even when two tables are linked, it's possible to have records in one table that have no relationship to any record in the other table.

So what happens when Access tries to run our query with the current join, **left to right**? Well, first it pulls every record from the table to the left, our Customers table.



The Left to Right join retrieves all the records from the table on the left first.

It then retrieves every record from the **right** table that has a relationship with a record Access has already taken from the left table.

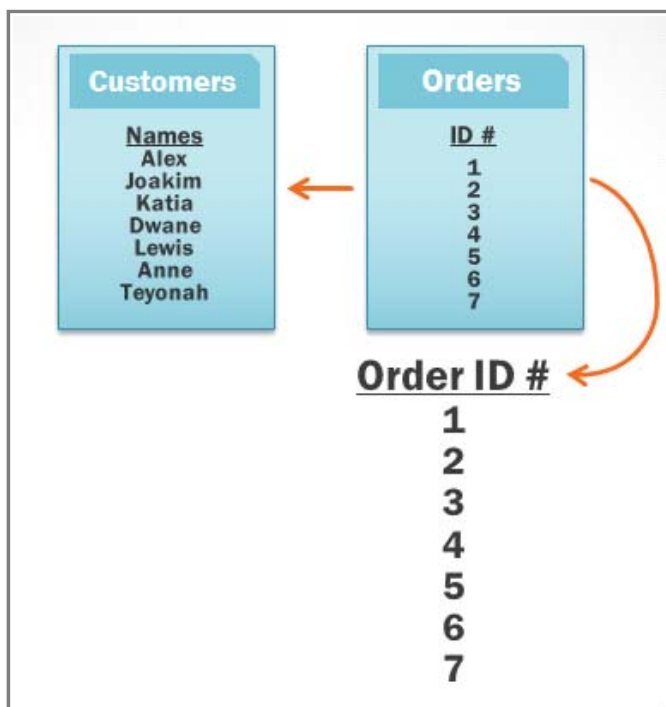


The query then retrieves the orders linked to the customer records it already pulled. These are the records the query

will draw its information from.

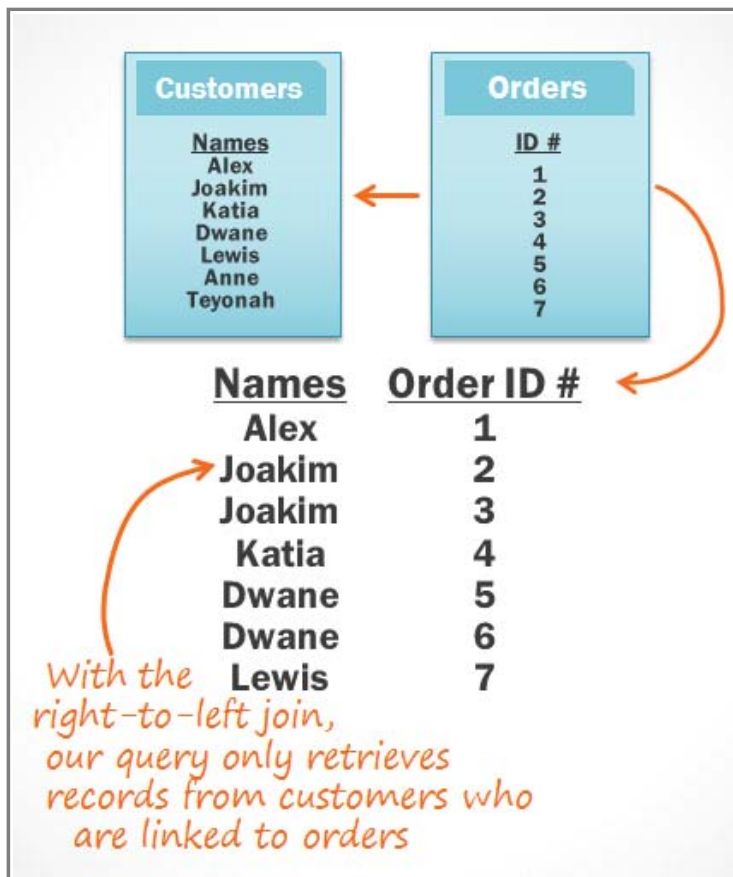
Since our join began with the **Customers** table, our query will include records for **all** of our customers, including those who never placed orders. This is way more information than we want! We **only** want to see records for **customers who have placed orders**.

Fortunately, we can fix this problem by changing the direction of the join line. If we join the tables from **right to left** instead, Access will first retrieve all the orders from the **right** table, our **Orders** table:



The Right to Left join retrieves all the records from the table on the right first.

Then, Access will look at the left table and retrieve **only** the records of customers who are linked to an order on the right.



Next, Access retrieves only the records from the left table that are linked to existing orders from the right. These are the records the query will draw its information from.

We now have exactly the information we want: **all** of the customers who have placed an order, and **only** those customers. As you can see, we had to join our tables in the **correct direction** to obtain the information we wanted.

Now that we understand which join direction we need to use, we're ready to build our query!

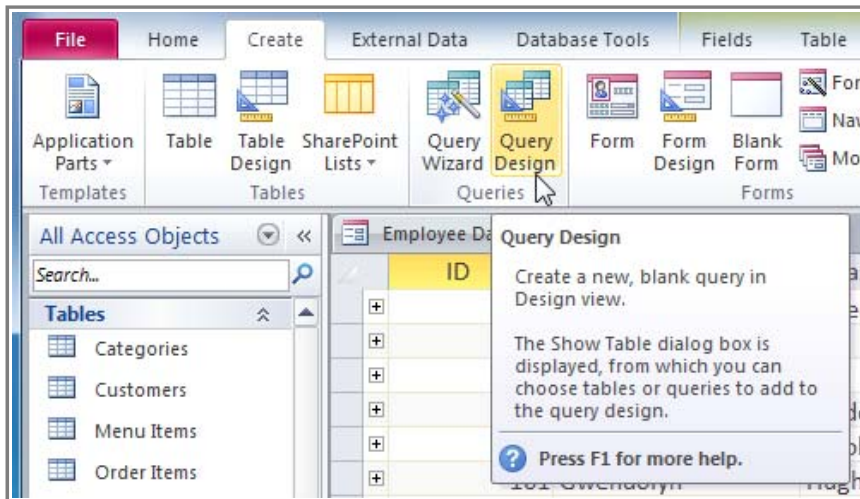
In our query, we needed to use the **right to left** join, but the correct join direction for the tables in your queries will depend on **what** information you want to see and **where** that information is stored. When you add tables to a query, Access will automatically join the tables for you, but it often doesn't join them in the correct direction. This is why it's important to **always review the joins** between your tables before you build a query.

Creating a Multi-Table Query

Now that we've planned our query, we're ready to design and run it. If you have created written plans for your query, be sure to reference them often throughout the query design process.

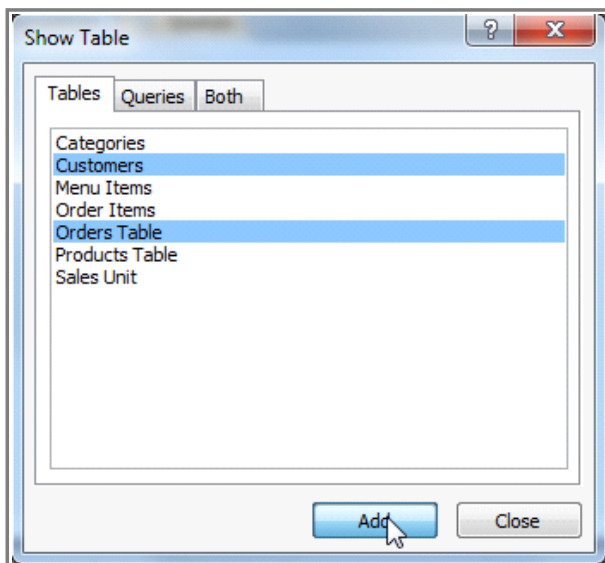
To Create a Multi-Table Query:

1. Select the **Query Design Command** from the **Create** tab on the Ribbon.



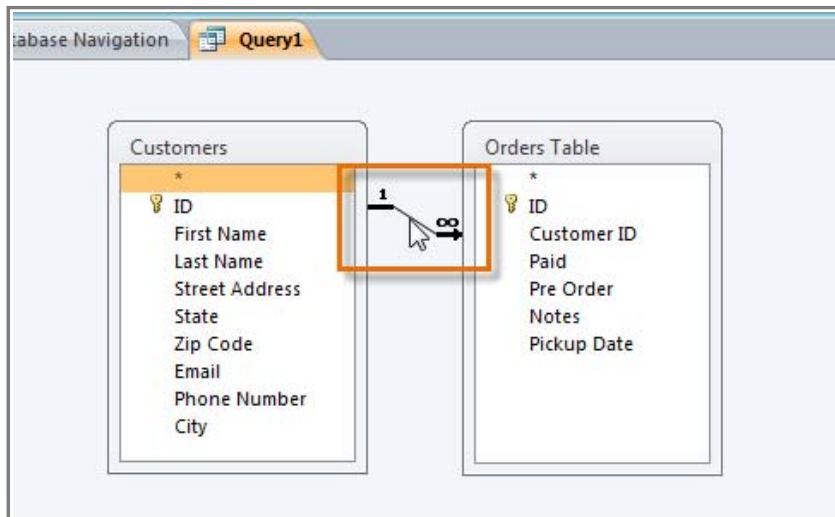
The Query Design Command

2. In the **Show Table** dialog box that appears, select each table you would like to include in your query and click **Add**. After you have added all of the tables you wish, click **Close**. When we planned our query, we decided we needed information from the **Customers** and **Orders** table, so we'll add those.



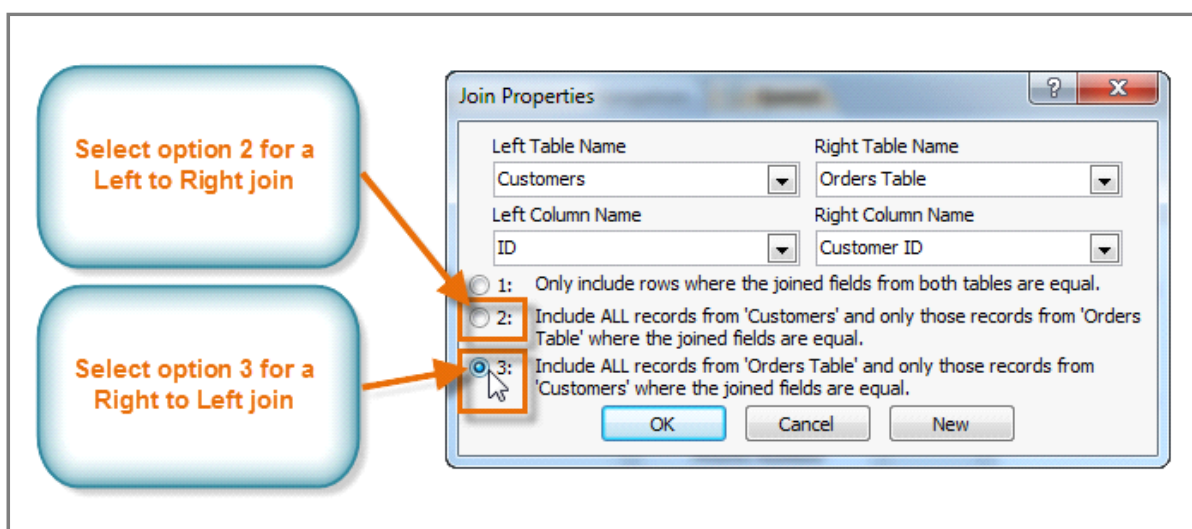
Selecting tables to use in the query

- The tables will appear in the **Object Relationship Pane**, linked by a **join line**. Double-click the thin section of the join line between two tables to edit its **join direction**.



Clicking the join line to edit its direction

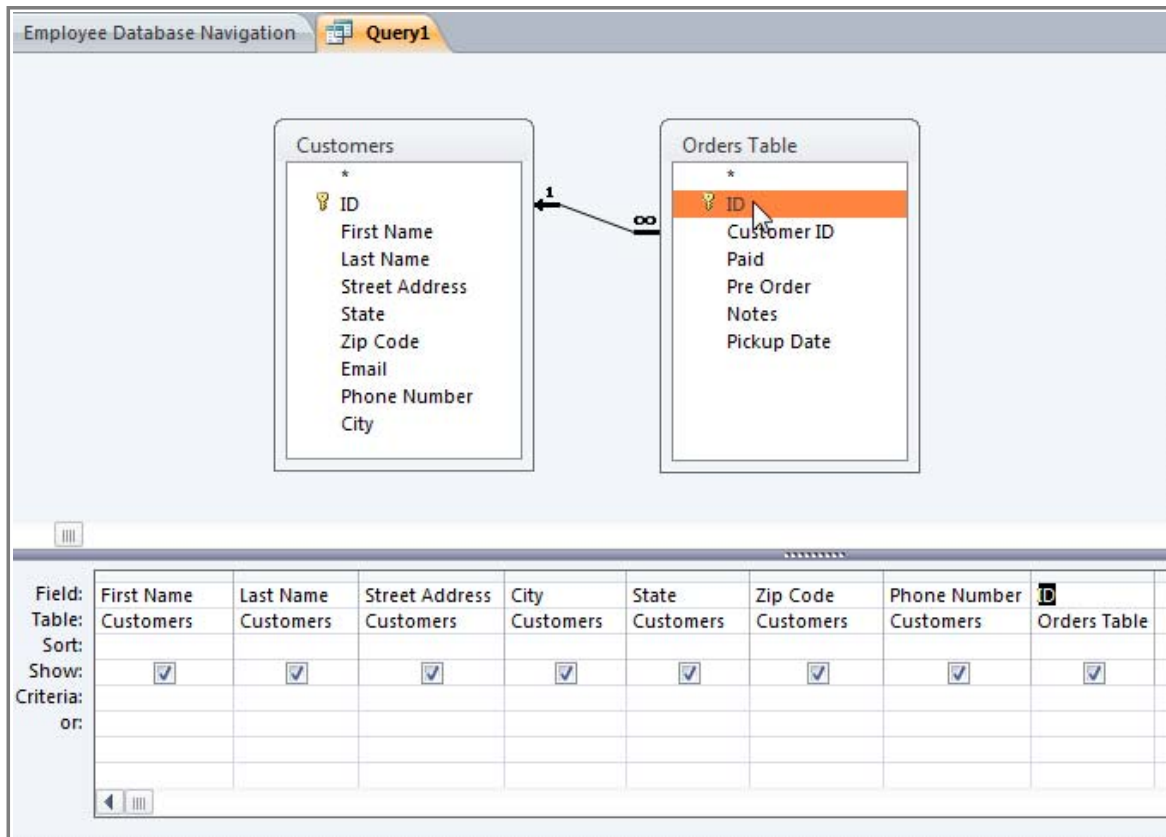
- The **Join Properties** dialog box will appear. Select an option to choose the direction of your join.
 - Choose option **2**: for a **Left to Right** join. In our query, the **left** table is the **Customers** table, so choosing this would mean that all of the customers who met our location criteria, whether or not they had placed an order, would be included in our results. We don't want to choose this option for our query.
 - Choose option **3**: for a **Right to Left** query. Since our **right** table is our **Orders** table, selecting this option will let us work with records for **all** of the orders and **only** the customers who've placed orders. We'll choose this option for our query, since this is exactly the data we want to see.



Changing the join direction to Right to Left

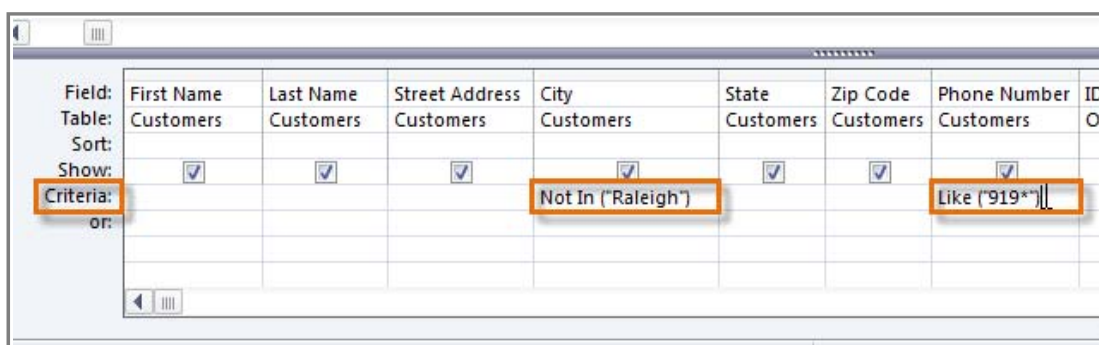
- In the table windows, double-click the **field names** you would like to include in your query. They will be added to the **Design Grid** in the bottom part of the screen.

In our example, we'll include most of the fields from the **Customers** table: **First Name**, **Last Name**, **Address**, **City**, **State**, **Zip Code**, and **Phone Number**. We'll also include the **ID** number from the **Orders** table.



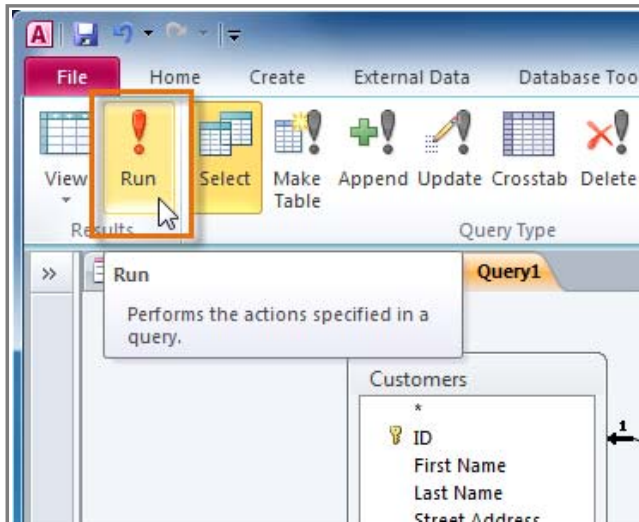
Adding table fields to the query

6. Set field **criteria** by entering the desired criteria in the criteria row of each field. We want to set two criteria:
 - o First, to find customers who do **not** live in Raleigh, we'll type **Not like ("Raleigh")** in the **City** field.
 - o Second, to find customers who have a phone number beginning with the area code **919**, we'll type **Like ("919**")** in the **Phone Number** field.



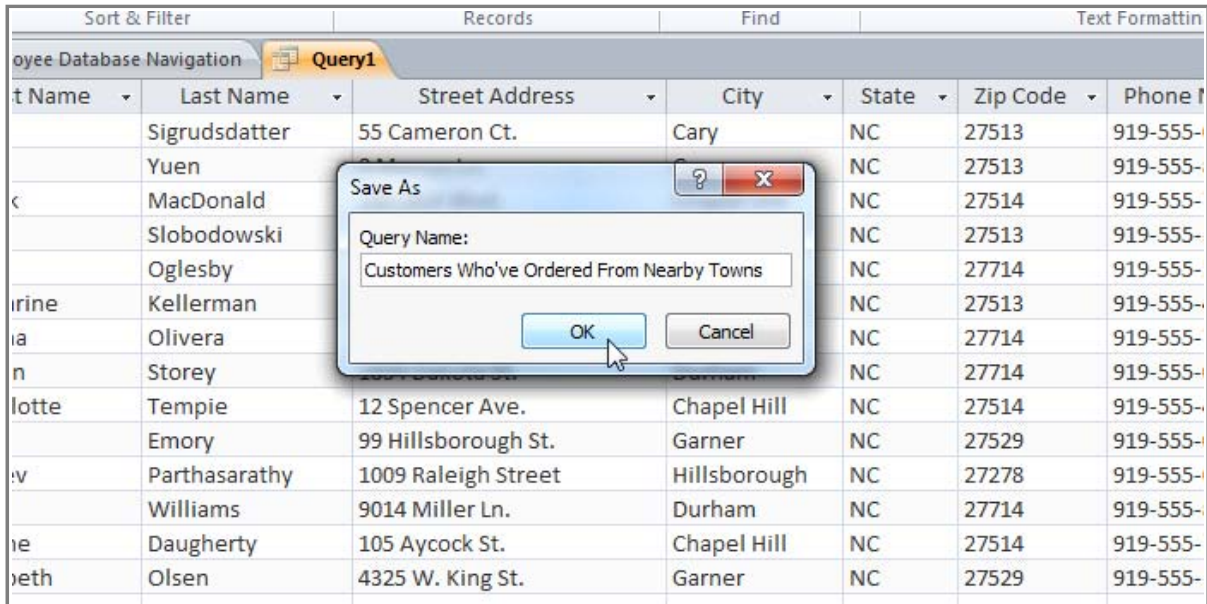
Setting field criteria

7. After you have set your criteria, **run** the query by clicking the **Run** command on the **Query Tools Design** tab.



Selecting the Run command

8. The query results will be displayed in the query's **Datasheet View**, which looks like a table. If desired, **save** your query by clicking the **Save** command in the Quick Access Toolbar. When prompted to name it, type in the desired name and click **OK**.



Naming the new query to save it

Challenge!

1. If you haven't already, [download our sample database](#) and **open** it.
2. **Create** a new query.
3. Select the **Customers** and **Orders** tables to include in your query.
4. Change the **join direction** to **Right to Left**.
5. **Add** the following **fields** from the **Customers** table to your query :
 - o **First Name**
 - o **Last Name**
 - o **City**
6. **Add** the following **fields** from the **Orders** table to your query:
 - o **Notes**
 - o **ID**
7. Set the following criteria:
 - o In the **Last Name** field, type **Like "Go*"** to return only records with last names beginning with "Go."
 - o In the **City** field, type **"Raleigh"** to return only records with "Raleigh" in the City field.
 - o In the **ID** field, type **>=60** to return only records with an ID number greater than or equal to 60.
8. **Run** the query. If you entered the query correctly, your results will include one record for a customer named "Will Good." If not, click the **View** drop-down arrow on the ribbon to return to Design View and check your work.
9. **Save** the query with the name **Will Query**.

