Introduction

In the previous lesson, you learned how to create a **simple query** with one table. Most queries you design in Access will likely use **multiple tables**, allowing you to answer more complex questions. In this lesson, you'll learn how to design and create a **multi-table query**.

Throughout this tutorial, we will be using a sample database. If you would like to follow along, you'll need to download our " Multi Table Query.accdb". You will need to have Access 2013 installed on your computer in order to open the example.

Designing a multi-table query

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

Planning a query

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?

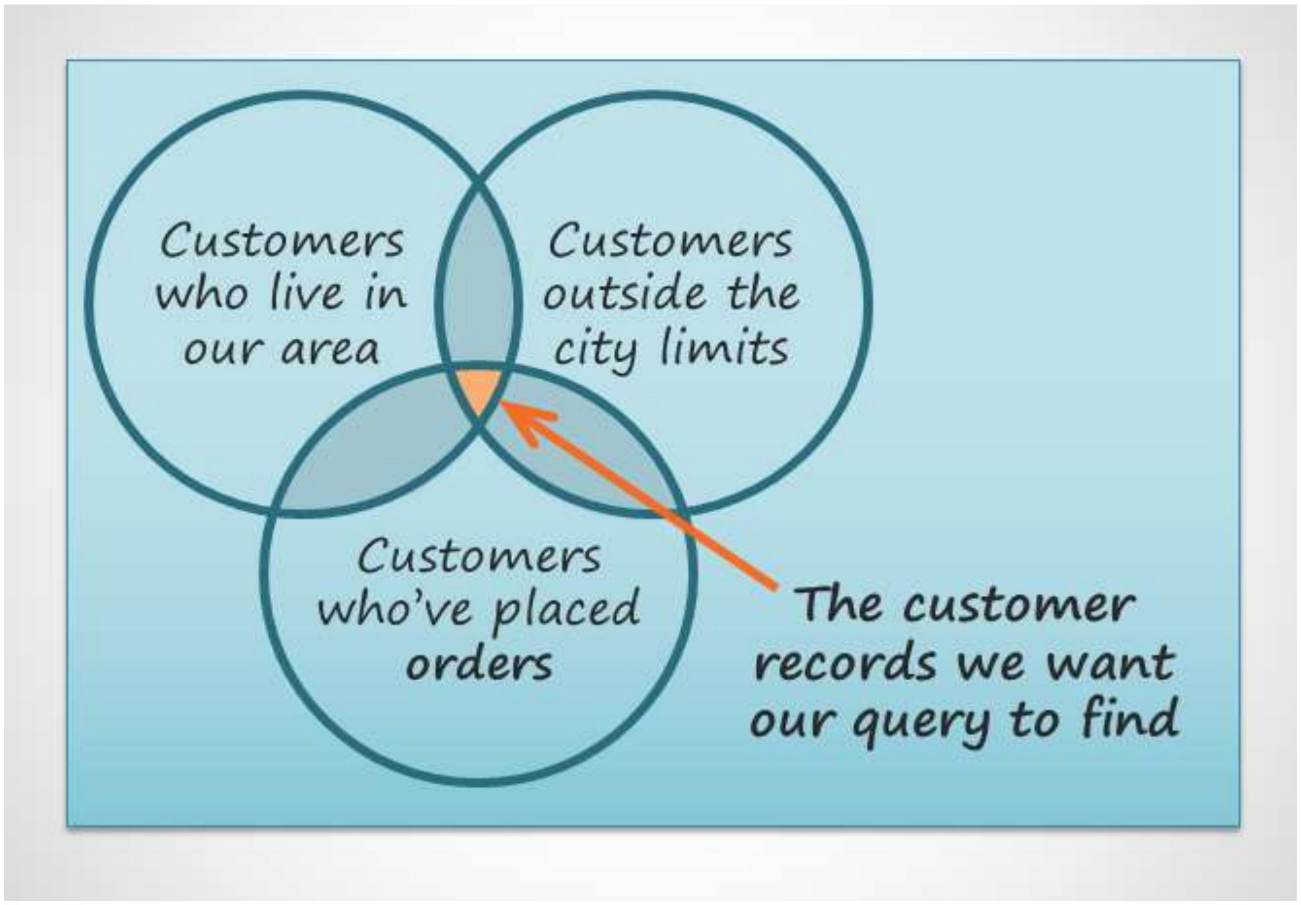
This process might seem abstract at first, but as we go through the process of planning our own multi-table query you should start to understand how planning your queries can make building them a lot easier.

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.

Step 1: Pinpointing the question we want to ask

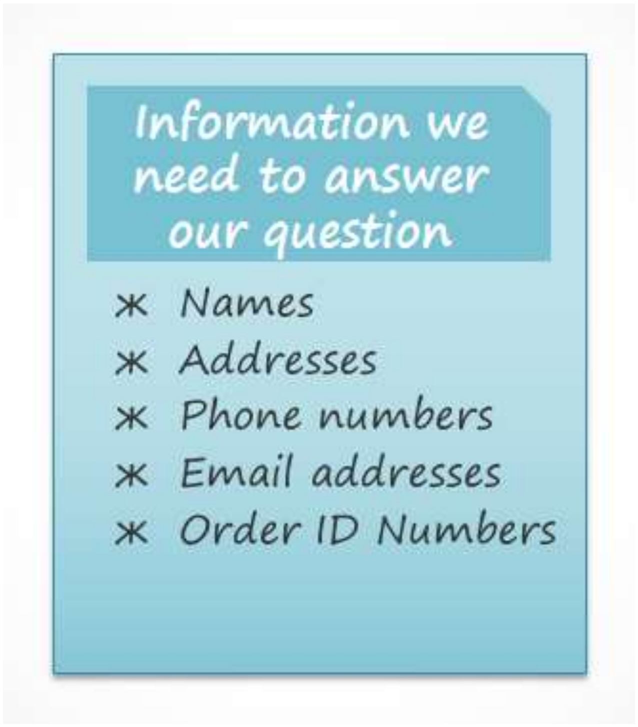
Our bakery database contains many customers, some of whom have never placed an order but who are in our database because they signed up for our mailing list. Most of them live within the city limits, but others live out of town or even out of state. We want to get our out-of-town customers who've placed 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 that person come in. So we just want to find people who don't live in our city but who still live in our area.


Customers who live in our area
Customers outside the city limits
Customers who've placed orders
The customer records we want our 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?**

Step 2: Identifying the information we need

What information might we want to see in a list 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.

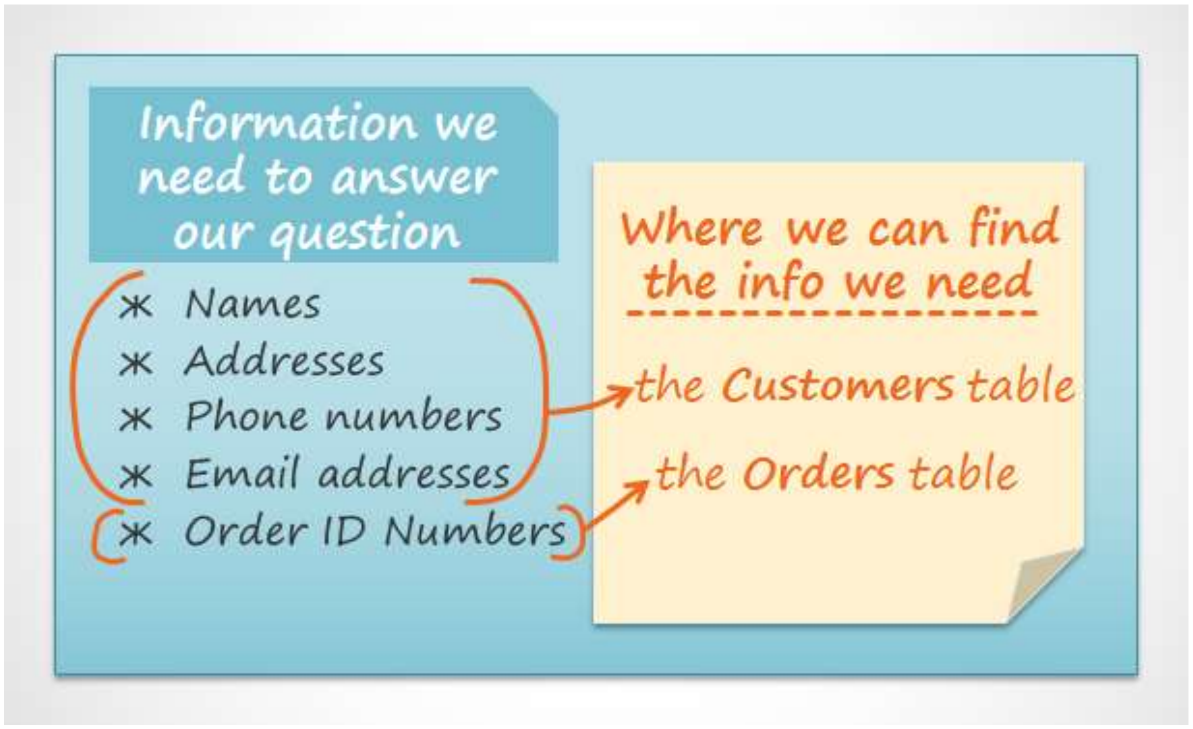


Information we need to answer our question

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

Step 3: Locating the tables containing the information we need

In order to write a query, you need to be 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.

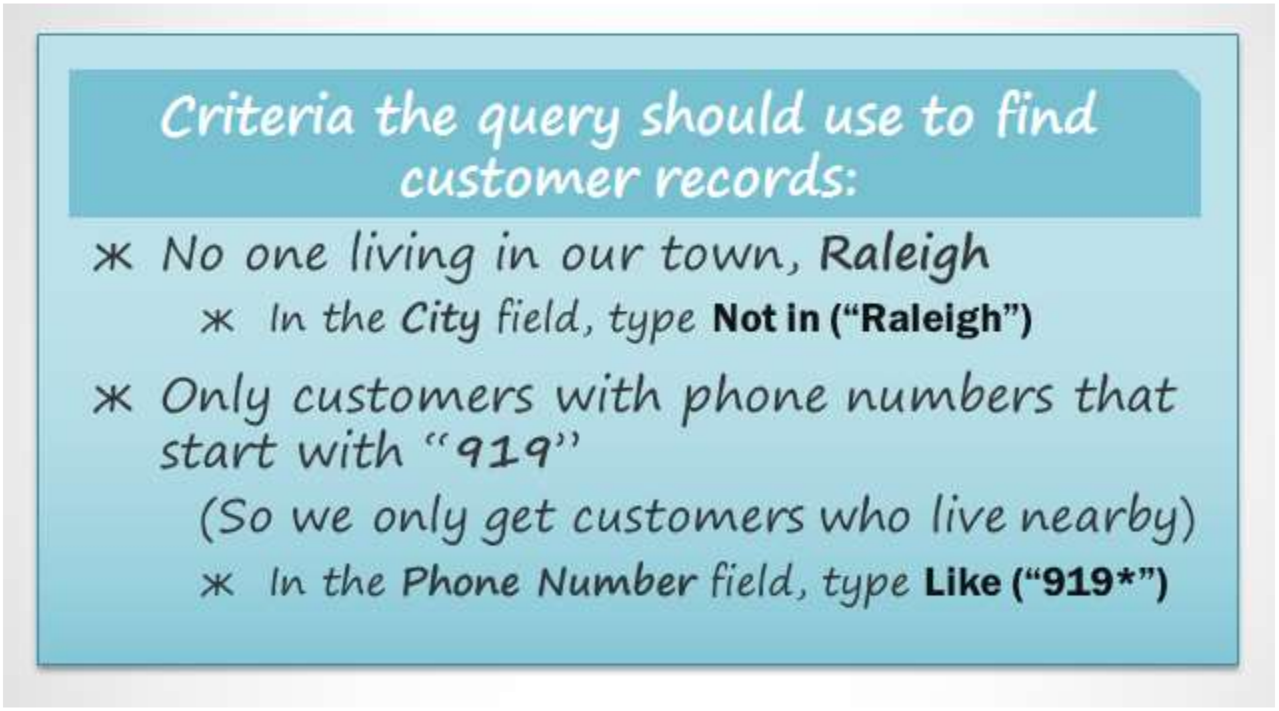


Step 4: Determining the criteria our query should search for

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 for** those with **Raleigh** in the city field. We don't want customers who live too far away, either. All of 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 because we want to see **all** of the orders made by people who meet the two criteria we just set.



Criteria the query should use to find customer records:

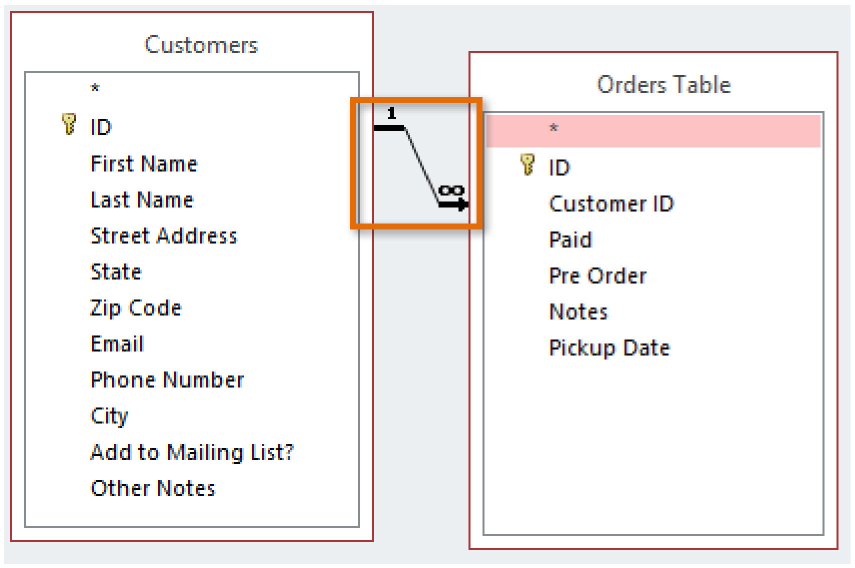
- No one living in our town, Raleigh
  - In the City field, type **Not in ("Raleigh")**
- Only customers with phone numbers that start with "919"

  (So we only get customers who live nearby)
  - In the Phone Number field, type **Like ("919*")**

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**, which includes several of the most common criteria used in Access queries.

Joining tables in queries

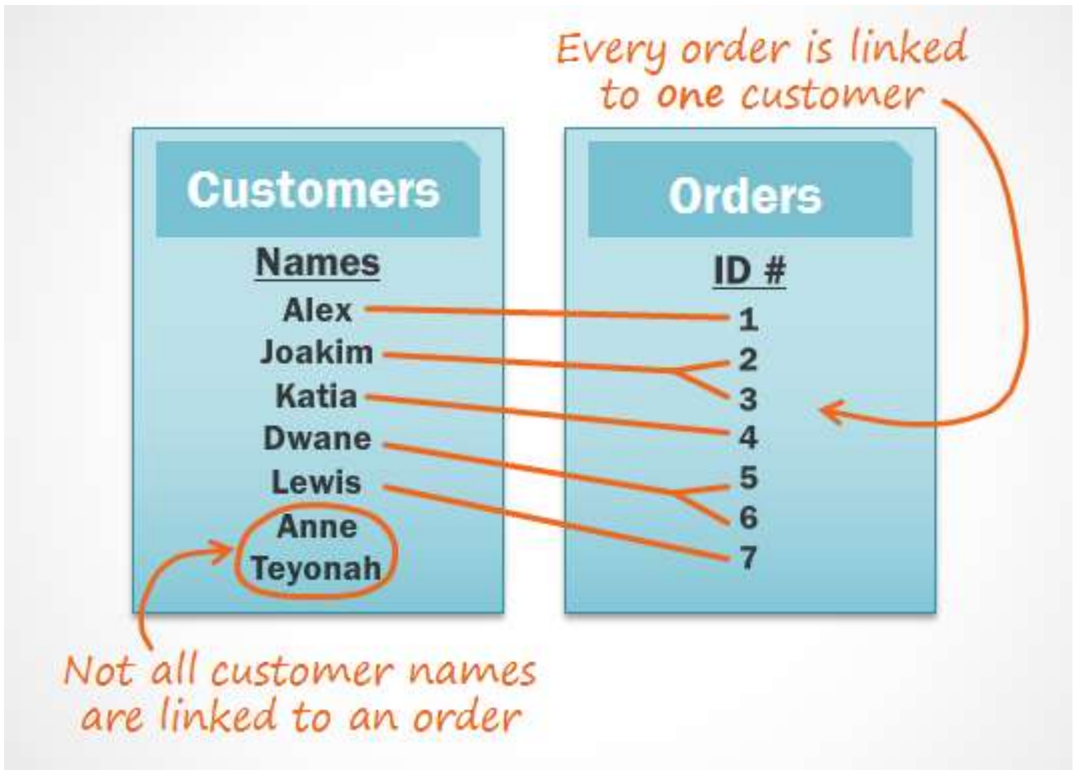
The final thing you need to consider when designing a query is the way you link—or **join**—the tables you're working with. When you add two tables to an Access query, this is what you'll see in the **Object Relationship pane**:



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 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.

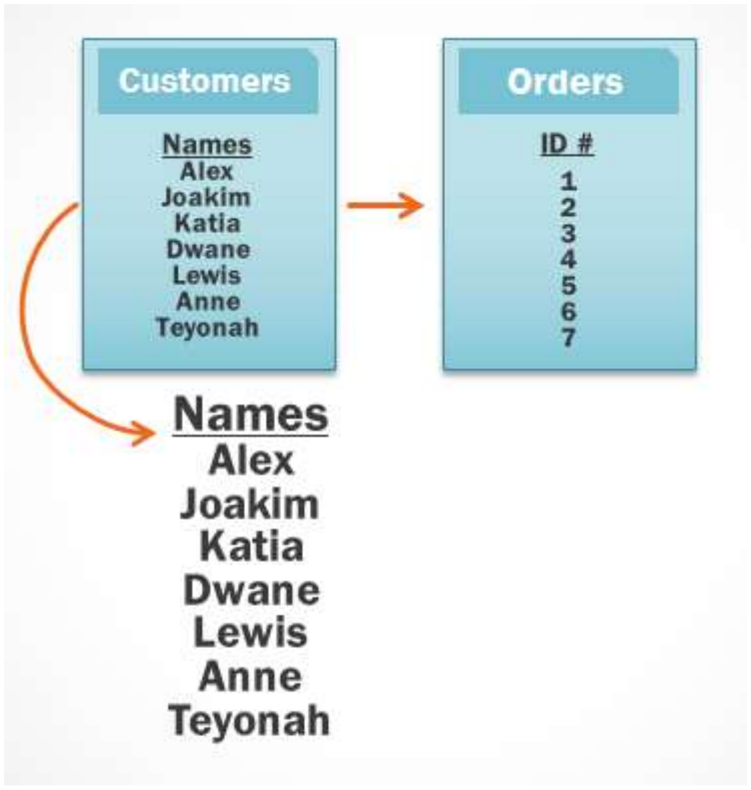
Your tables won't always be joined this way. Sometimes Access will join them **right** to **left**. In either case, you might need to **change the direction** of the join to make sure your query includes the correct information. The join direction can affect **which information** your query **retrieves**.

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 these tables.

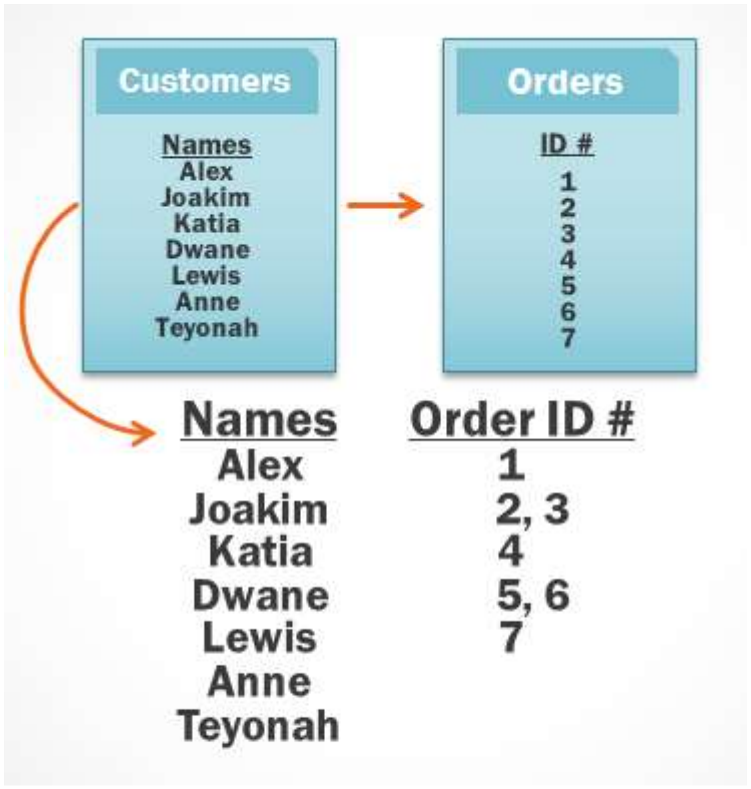

Every order is linked to one customer
Customers
Names
Alex
Joakim
Katia
Dwane
Lewis
Anne
Teyonah
Orders
ID #
1
2
3
4
5
6
7
Not all customer names are linked to an order

What do you notice when you look at these lists? 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. 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**? It pulls every record from the table to the left: our Customers table.

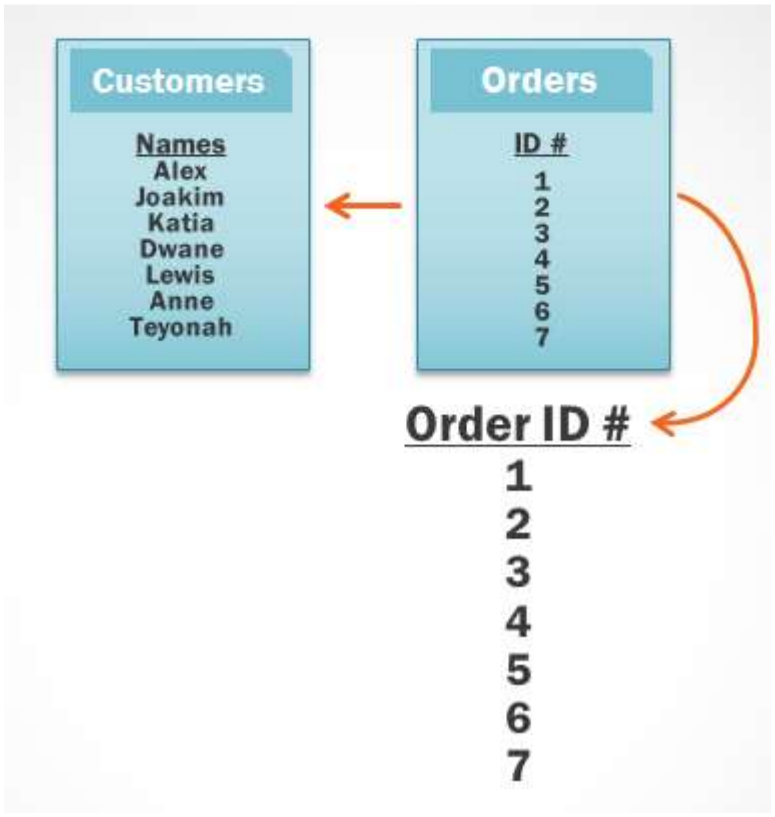


It then retrieves every record from the **right** table that has a relationship with a record Access has already taken from the left table.

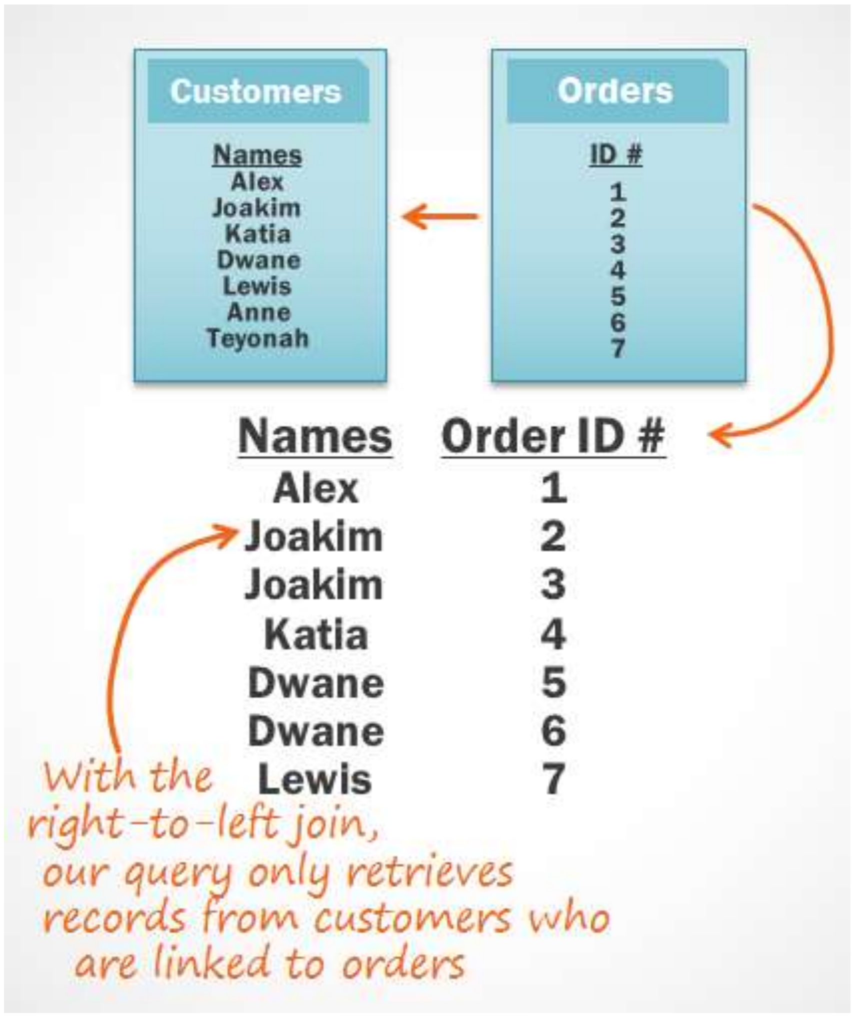


Because our join began with the **Customers** table, our query will include records for **all** of our customers, including those who've never placed orders. This is more information than we need. We **only** want to see records for **customers who have placed orders**.

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 the orders from the **right**table, our **Orders** table:



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



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!

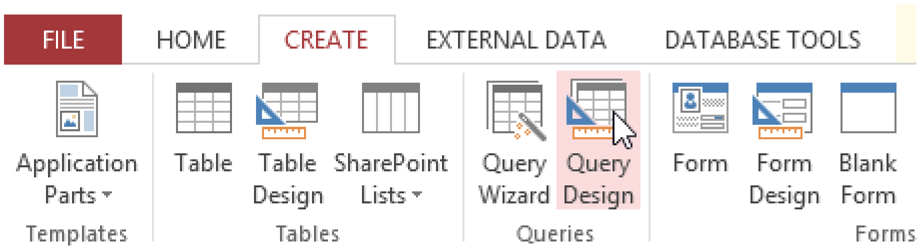
In our query, we needed to use the **right-to-left** join, but the correct join direction for the tables in your queries will depend on **what** information you want to see and **where** that information is stored. When you add tables to a query, Access will automatically join the tables for you, but 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

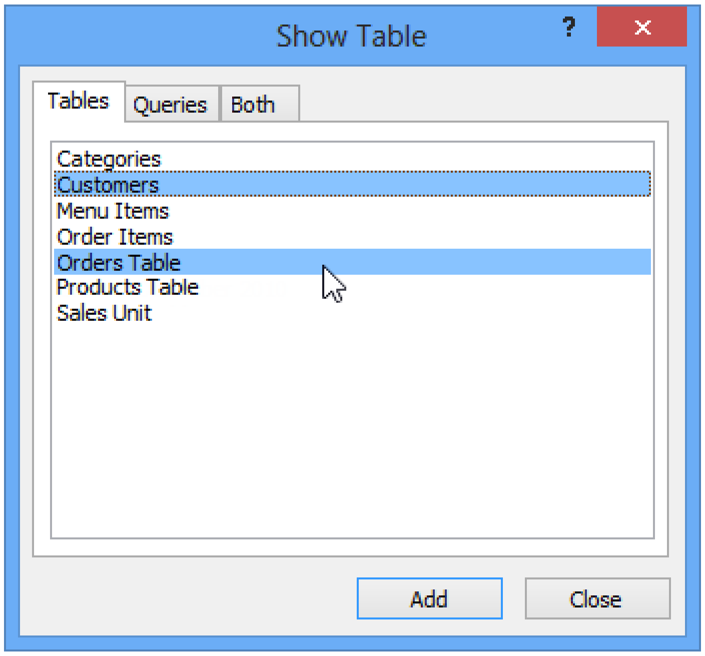
Now that we've planned our query, we're ready to design and run it. If you have created written plans for your query, be sure to reference them often throughout the query design process.

To create a multi-table query:

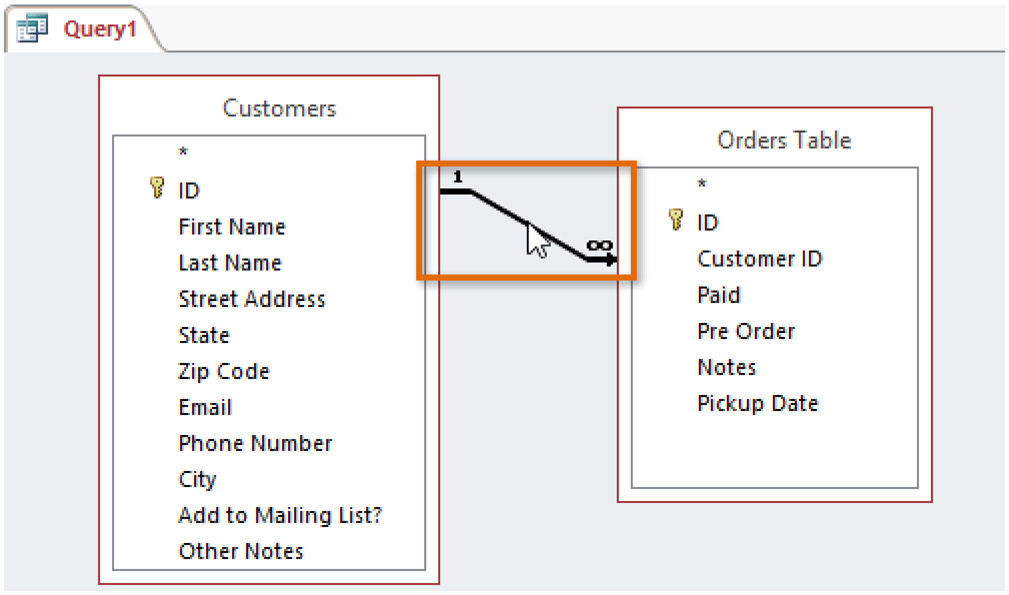
1. Select the **Query Design** command from the **Create** tab on the Ribbon.



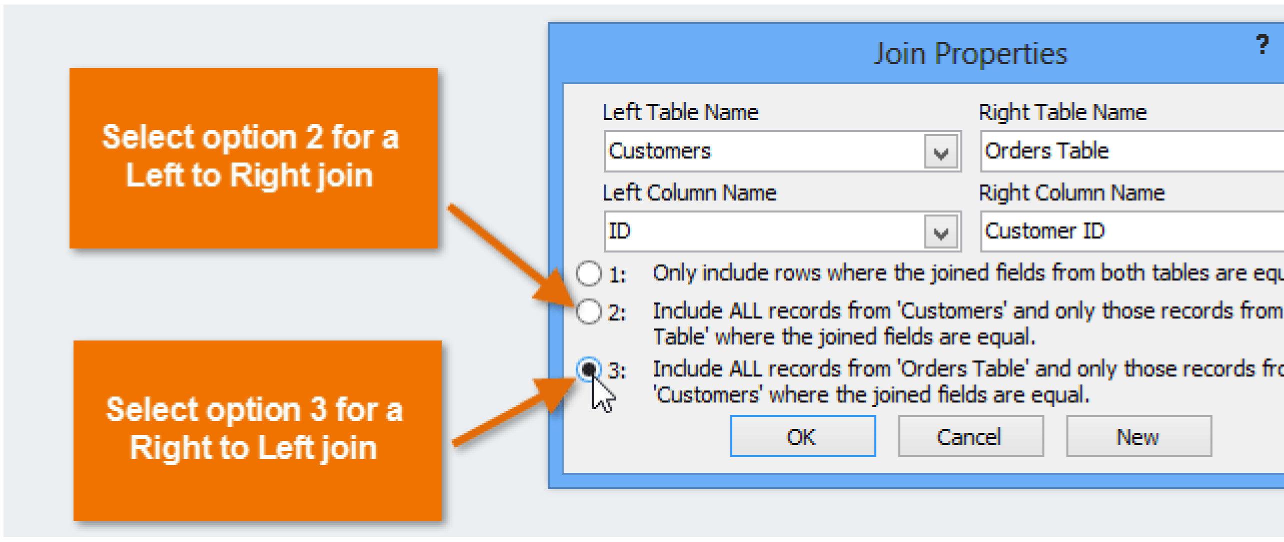
2. In the dialog box that appears, select each table you want to include in your query and click **Add**. You can press and hold the **Ctrl** key on your keyboard to select more than one table. When we planned our query, we decided we needed information from the **Customers** and **Orders** table, so we'll add these.



3. After you have added all of the tables you want, click **Close**.
4. 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**.



5. 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 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** join. Because 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 because this is exactly the data we want to see.


Select option 2 for a Left to Right join
Select option 3 for a Right to Left join

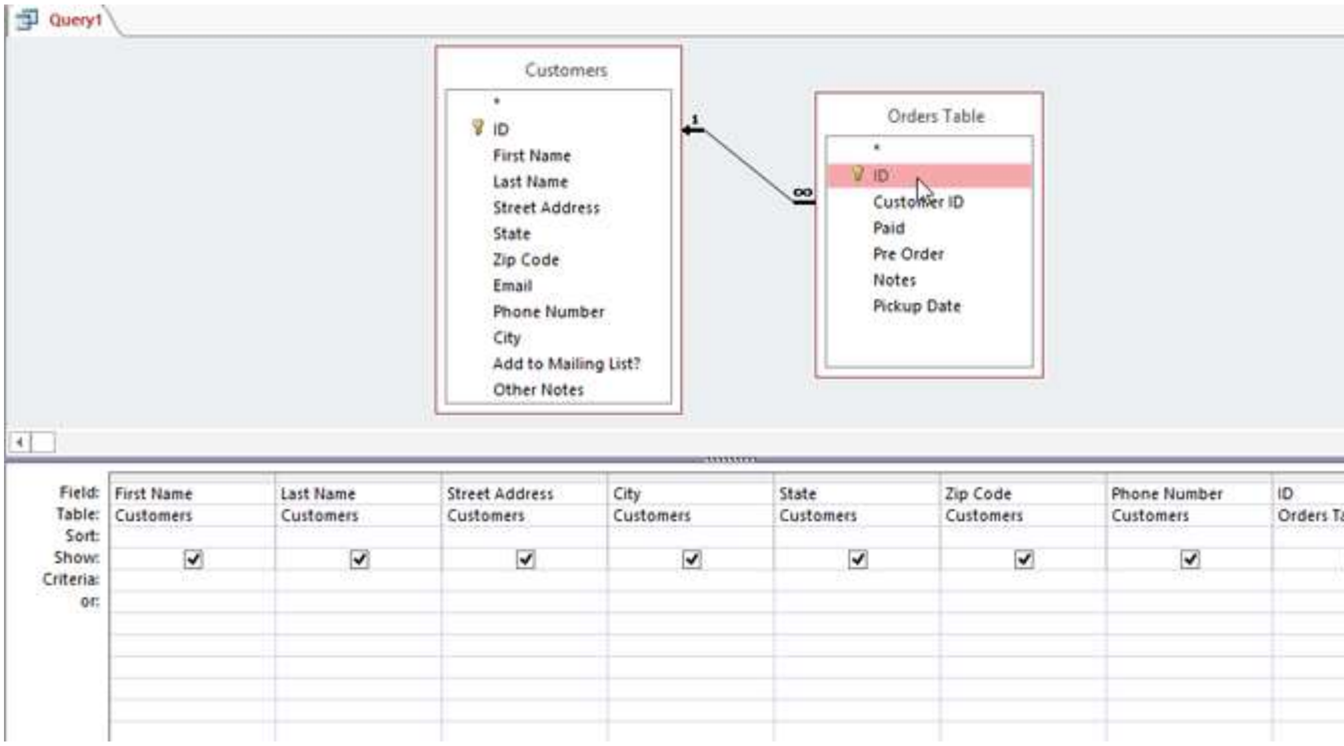
Join Properties

Left Table Name: Customers
Right Table Name: Orders Table
Left Column Name: ID
Right Column Name: Customer ID

1: Only include rows where the joined fields from both tables are equ
2: Include ALL records from 'Customers' and only those records from Table' where the joined fields are equal.
3: Include ALL records from 'Orders Table' and only those records fro 'Customers' where the joined fields are equal.

6. In the table windows, double-click the **field names** you want to include in your query. They will be added to the **design grid** in the bottom part of the screen.

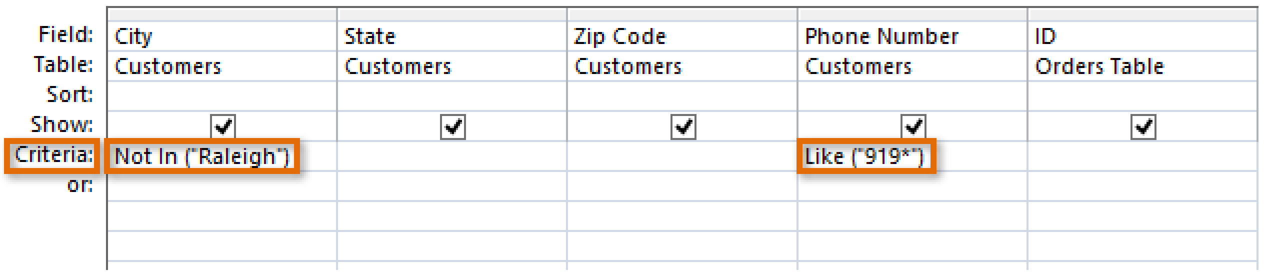
   In our example, we'll include most of the fields from the **Customers** table: **First Name**, **Last Name**, **Street Address**, **City**, **State**, **Zip Code**, and **Phone Number**. We'll also include the **ID number** from the **Orders** table.



Field:	First Name	Last Name	Street Address	City	State	Zip Code	Phone Number	ID
Table:	Customers	Customers	Customers	Customers	Customers	Customers	Customers	Orders T
Sort:								
Show:	✓	✓	✓	✓	✓	✓	✓	
Criteria:								
or:								

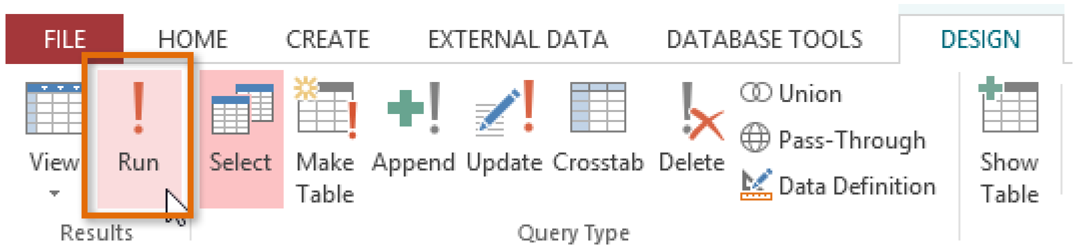
7. Set field **criteria** by entering the desired criteria in the criteria row of each field. We want to set two criteria:

   - First, to find customers who do **not** live in Raleigh, we'll type **Not in ("Raleigh")** in the **City** field.
   - 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.

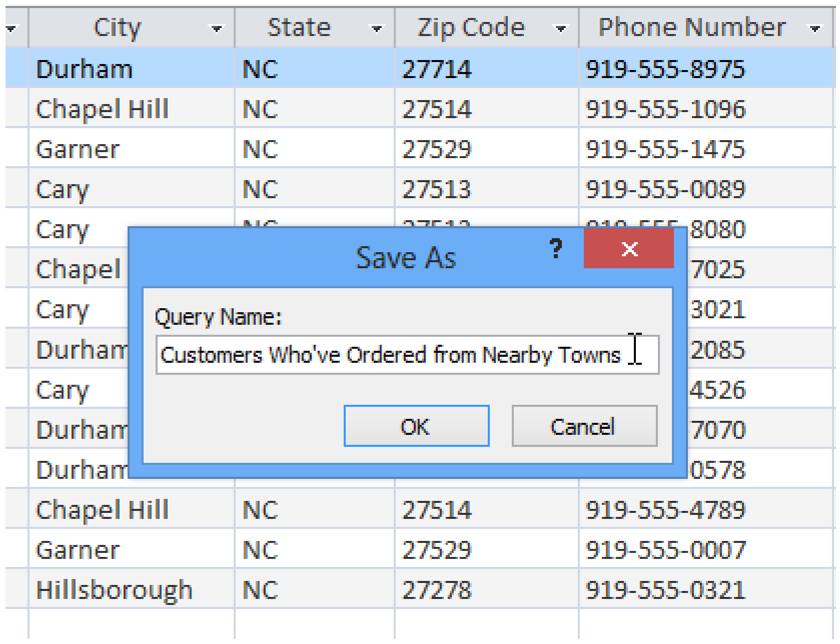


Field:	City	State	Zip Code	Phone Number	ID
Table:	Customers	Customers	Customers	Customers	Orders Table
Sort:					
Show:	✓	✓	✓	✓	✓
Criteria:	Not In ("Raleigh")			Like ("919*")	
or:					

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



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



City	State	Zip Code	Phone Number
Durham	NC	27714	919-555-8975
Chapel Hill	NC	27514	919-555-1096
Garner	NC	27529	919-555-1475
Cary	NC	27513	919-555-0089
Cary			8080
Chapel			7025
Cary			3021
Durham			2085
Cary			4526
Durham			7070
Durham			0578
Chapel Hill	NC	27514	919-555-4789
Garner	NC	27529	919-555-0007
Hillsborough	NC	27278	919-555-0321

Query Name: Customers Who've Ordered from Nearby Towns

Now you know how to create a **multi-table** query. In the next lesson, we'll talk about more query design options that can make your query even more powerful.

Challenge!

1. Open an **existing Access database**. If you want, you can use our “Multi Table Query.accdb”.
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:
    - **First Name**
    - **Last Name**
    - **City**
6. **Add** the following **fields** from the **Orders** table to your query:
    - **Notes**
    - **ID**

7. Set the following criteria:
    - In the **Last Name** field, type **Like "Go*"** to return only records with last names beginning with **Go.**
    - In the **City** field, type **"Raleigh"** to return only records with **Raleigh**in the City field.
    - 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**.