MICROSOFT ACCESS

More Query Design Options

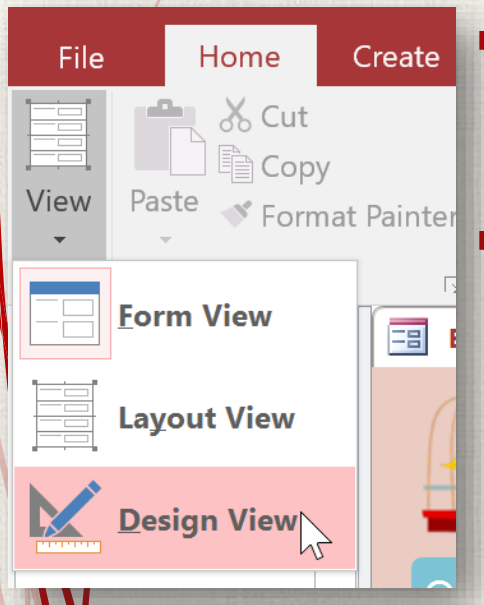
Contents

Introduction

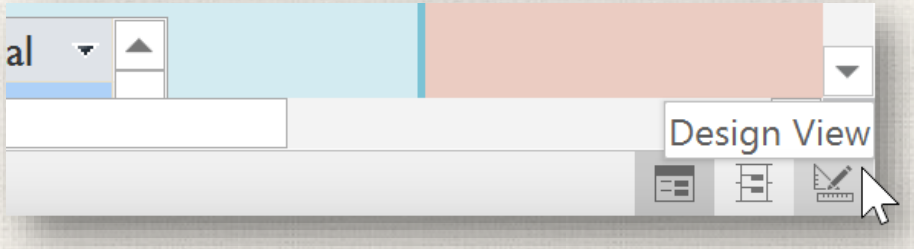
- Access offers several options that let you design and run queries that return exactly the information you're looking for.
- For instance, what if you need to find out **how many** of something exists within your database? Or what if you would like your query results to automatically be sorted a certain way? If you know how to use query options in Access, you can design almost any query you want.

Modifying queries

When you open an existing query in Access, it is displayed in **Datasheet view**, meaning you will see your query results in a table. To modify your query, you must enter **Design view**, the view you used when creating it. There are two ways to switch to Design view:



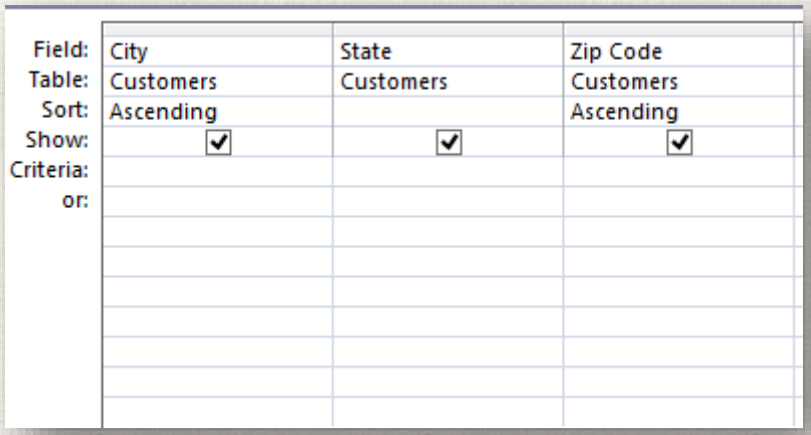
- On the **Home** tab of the Ribbon, click the **View** command. Select **Design View** from the drop-down menu that appears.
- In the bottom-right corner of your Access window, locate the small **view icons**. Click the **Design View** icon, which is the icon farthest to the right.



Once in Design view, make the desired changes, then select the Run command to view your updated results.

Sorting queries

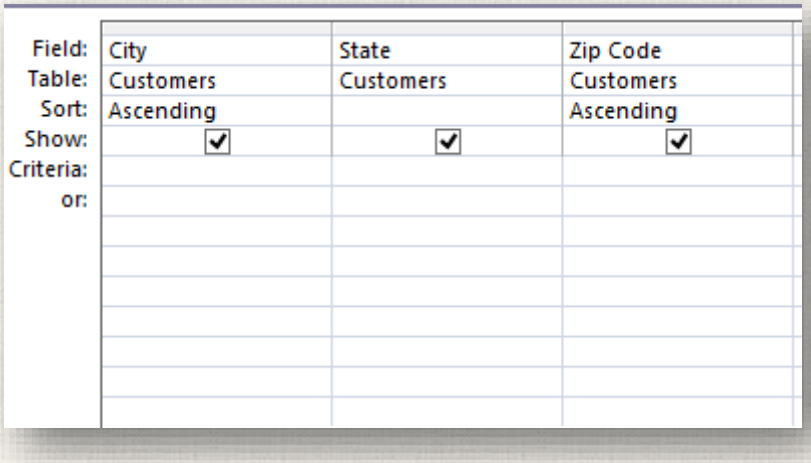
- Access allows you to apply multiple sorts at once while you're designing your query. This allows you to view your data exactly the way you want.
- A sort that includes more than one sorted field is called a **multilevel sort**. A multilevel sort allows you to apply an initial sort, then further organize data with additional sorts. For instance, if you had a table filled with customers and their addresses, you might choose to first sort the records by city, then alphabetically by last name.



Field:	City	State	Zip Code
Table:	Customers	Customers	Customers
Sort:	Ascending		Ascending
Show:	☑	☑	☑
Criteria:			
or:			

Sorting queries

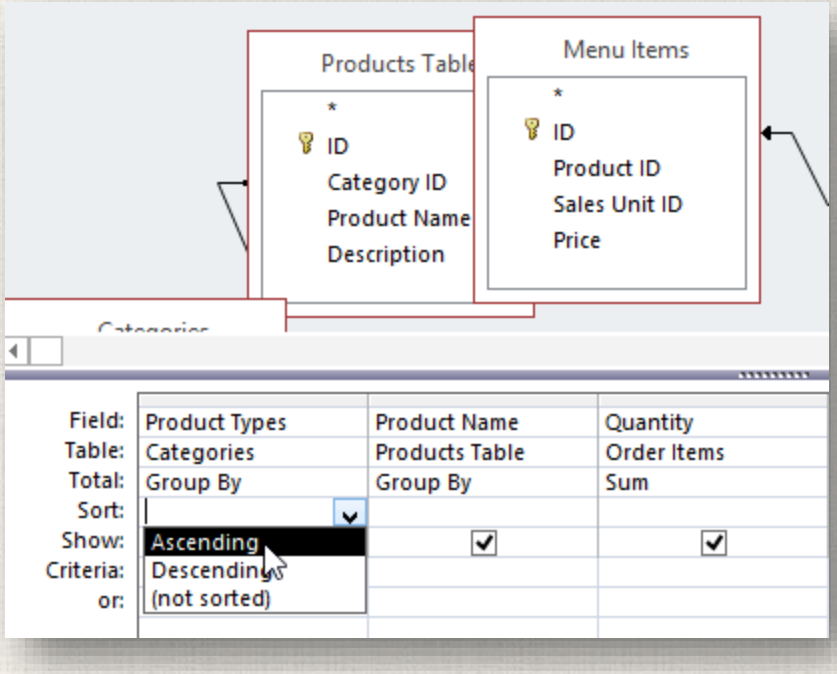
- When more than one sort is included in a query, Access reads the sorts from **left to right**. This means the leftmost sort will be applied first. In the example below, customers will be sorted first by the **City** where they live and then by the **Zip Code** within that city.



Field:	City	State	Zip Code
Table:	Customers	Customers	Customers
Sort:	Ascending		Ascending
Show:	☑	☑	☑
Criteria:			
or:			

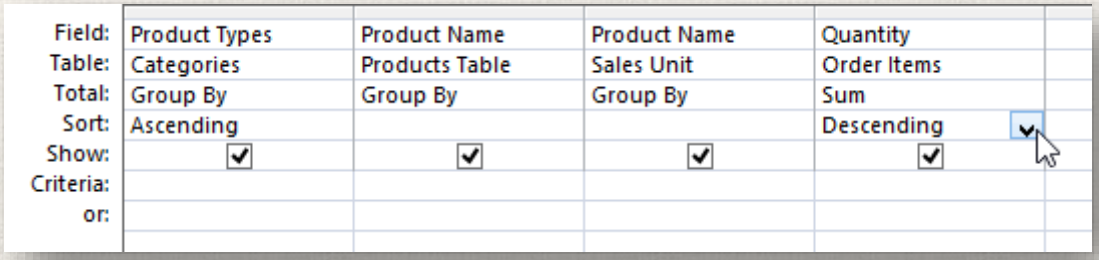
Apply a multilevel sort

1. Open the query and switch to **Design view**.
2. Locate the field you want to sort first. In the **Sort:** row, click the drop-down arrow to select either an **Ascending** or **Descending** sort.

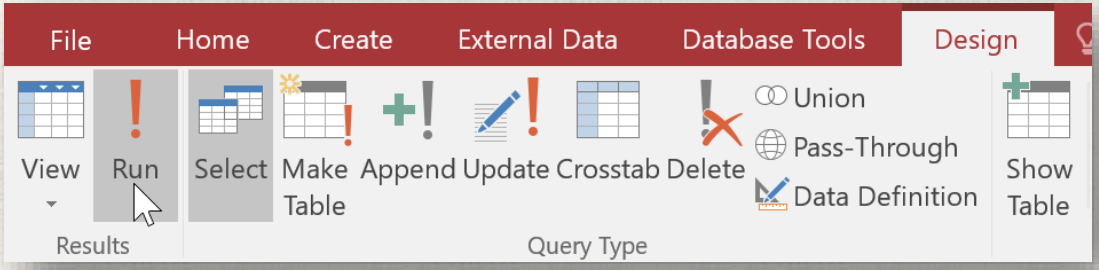


Apply a multilevel sort

3. Repeat the process in the other fields to add additional sorts. Remember, the sorts are applied from left to right, so any additional sorts must be applied to fields located **to the right** of your primary sort. If necessary, you can **rearrange** the fields by clicking the top of a field and dragging it to a new location.

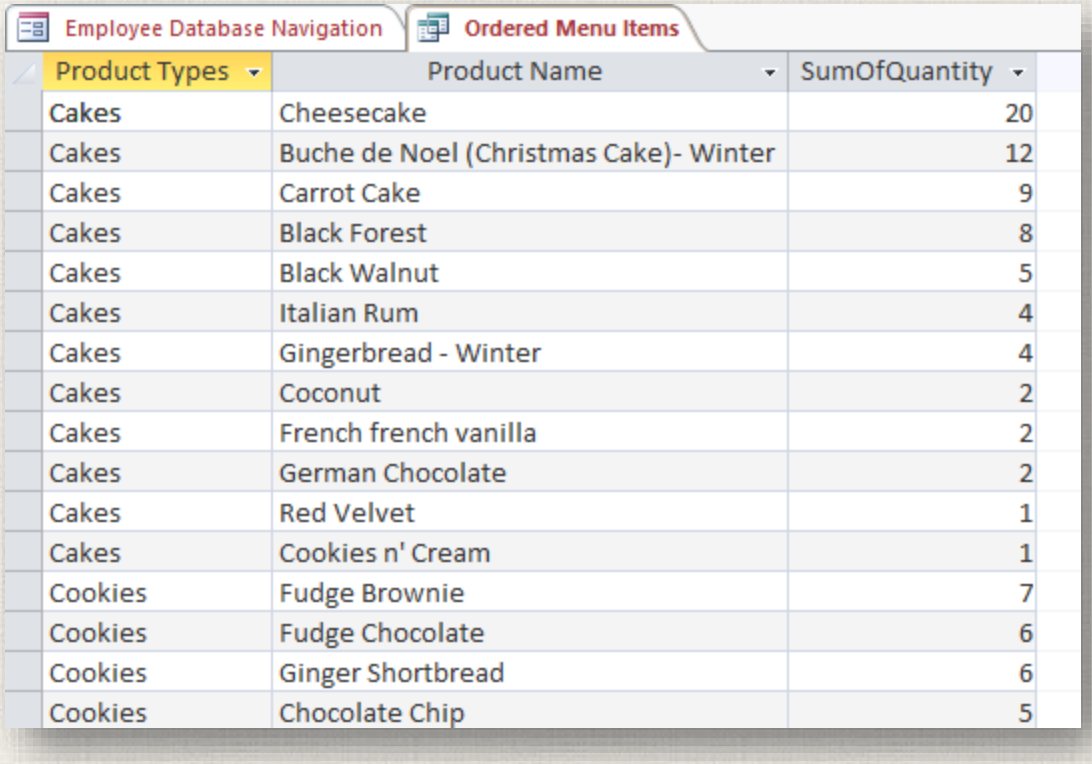


4. To apply the sort, click the **Run** command.



Apply a multilevel sort

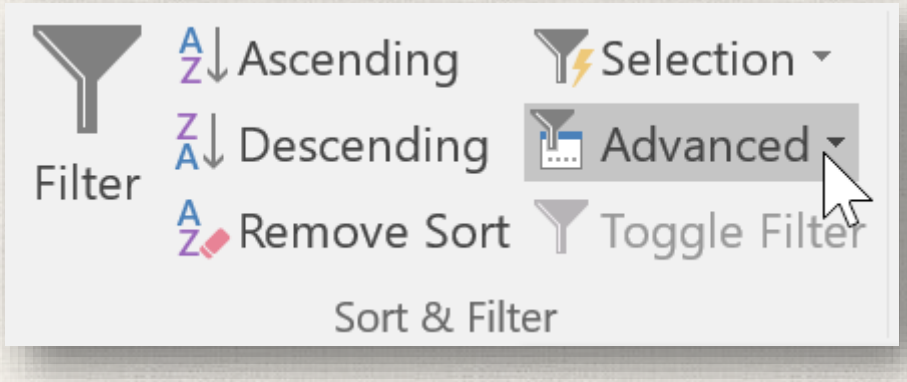
5. Your query results will appear with the desired sort.



Product Types	Product Name	SumOfQuantity
Cakes	Cheesecake	20
Cakes	Buche de Noel (Christmas Cake)- Winter	12
Cakes	Carrot Cake	9
Cakes	Black Forest	8
Cakes	Black Walnut	5
Cakes	Italian Rum	4
Cakes	Gingerbread - Winter	4
Cakes	Coconut	2
Cakes	French french vanilla	2
Cakes	German Chocolate	2
Cakes	Red Velvet	1
Cakes	Cookies n' Cream	1
Cookies	Fudge Brownie	7
Cookies	Fudge Chocolate	6
Cookies	Ginger Shortbread	6
Cookies	Chocolate Chip	5

Apply a multilevel sort

- You can also apply multilevel sorts to tables that don't have queries applied to them. On the **Home** tab on the Ribbon, select the **Advanced** drop-down command in the **Sort & Filter** group. From the menu that appears, select **Advanced Filter/Sort** and create the multilevel sort as you normally would. When you're finished, click the **Toggle Filter** command to apply your sort.

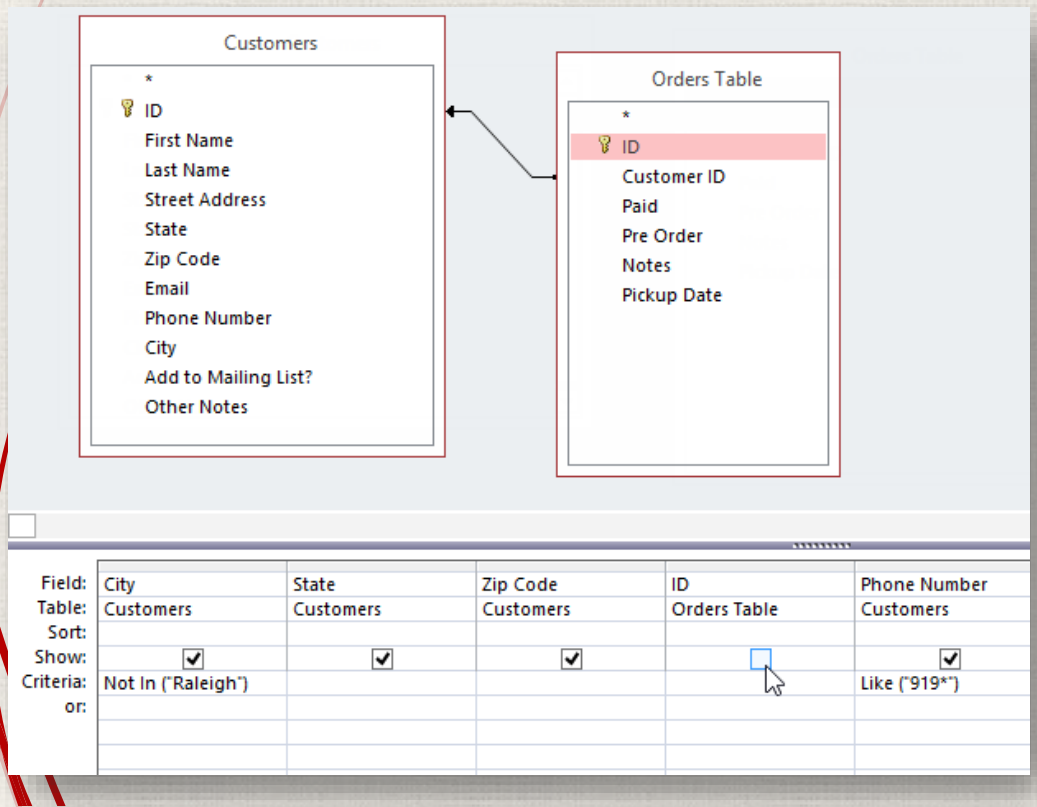


Hiding fields within queries

- Sometimes you might have fields that contain important criteria, but you might not need to actually see the information from that field in the final results.
- For example, take one of the queries we built in our last lesson: a query to find the names and contact information of customers who had placed orders. We included Order ID numbers in our query because we wanted to make sure we only pulled customers who had placed orders.
- However, we really didn't need to see this information in our final query results. In fact, if we were just looking for customer names and addresses, seeing the order number mixed in there might have been distracting. Fortunately, Access makes it easy to **hide** fields while still including any criteria they contain.

Hiding fields within queries

1. Open the query and switch to **Design view**.
2. Locate the field you want to hide.
3. Click the **checkbox** in the **Show:** row to uncheck it.
4. To see the updated query, select the **Run** command. The field will be hidden.



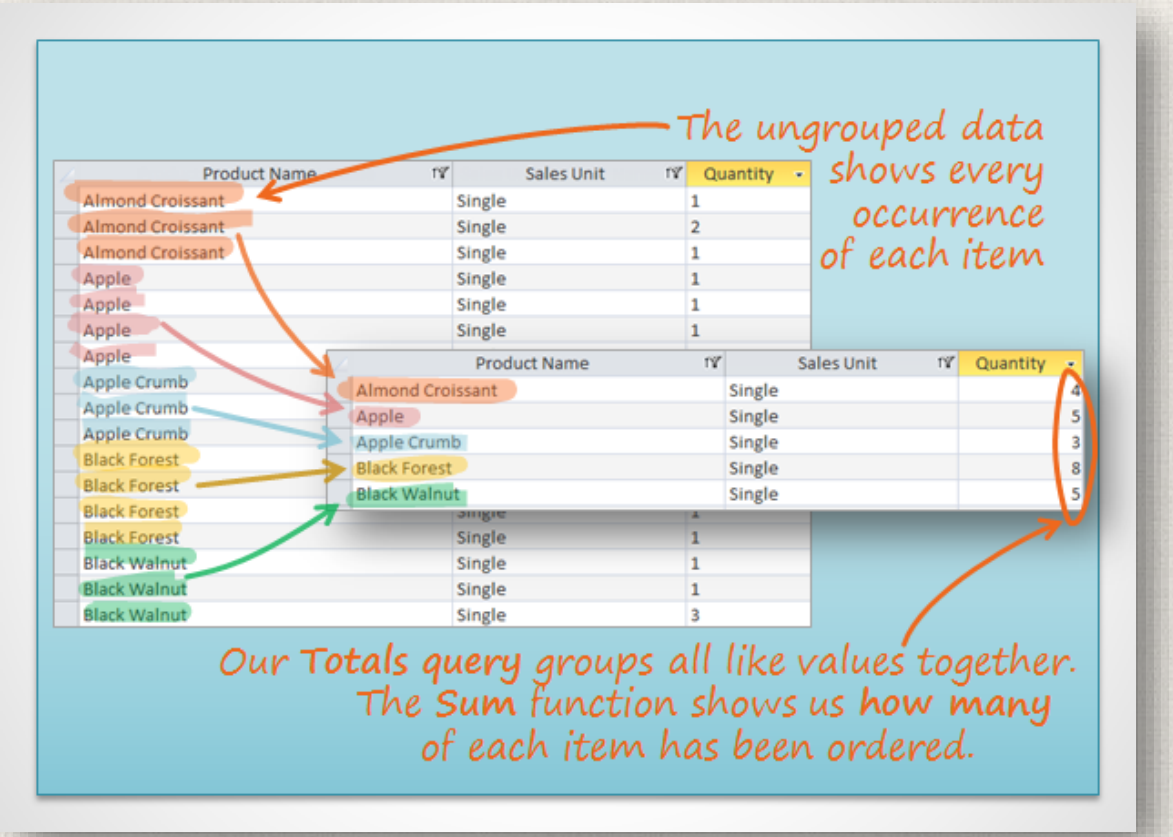
To **unhide** a hidden field, simply return to Design view and click the checkbox in the field's **Show:** row again.

Totals queries

- Sometimes setting simple criteria won't give you the results you need, especially when you're working with numerical values. You may want to see your query results grouped or counted in some way.
- For example, let's say we want to find out **how many of each menu item at our bakery has been ordered**—how many Almond Croissants, Apple Pies, and so on. To do this, we could create a totals query to find the **sum** of the **quantities** for each item.

Totals queries

- First, the totals query will group all similar menu items from separate orders (for example, Almond Croissants). Then, the Sum function will add the values in the Quantity field to calculate the total number sold for that item.



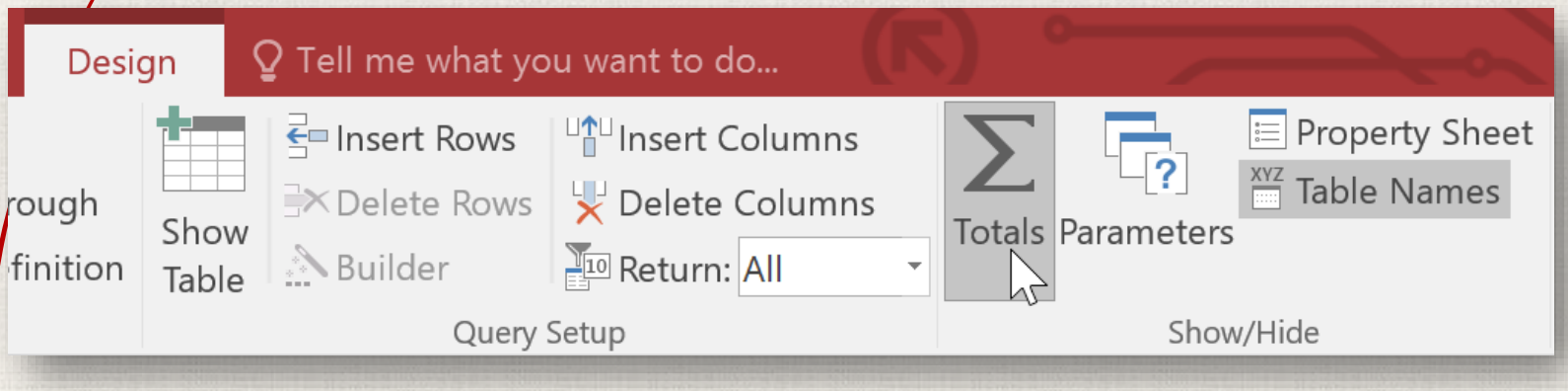
Totals queries

- The **Sum** function helped us find the desired information in this example, but in other situations you may need to use a different function to find the answer you need. There are several functions you can choose from:
  - **Count**: Counts the total number of each item
  - **Sum**: Adds the values together
  - **Average**: Finds the average of the values
  - **Maximum**: Returns the highest value
  - **Minimum**: Returns the lowest value
  - **First**: Returns the first—or earliest—value
  - **Last**: Returns the last—or most recent—value
- In our example, we created a **subtotal** for each menu item in our query. If you wanted to create a **grand total** for all of the items, you would need to add a **totals row**.

Creating a totals query

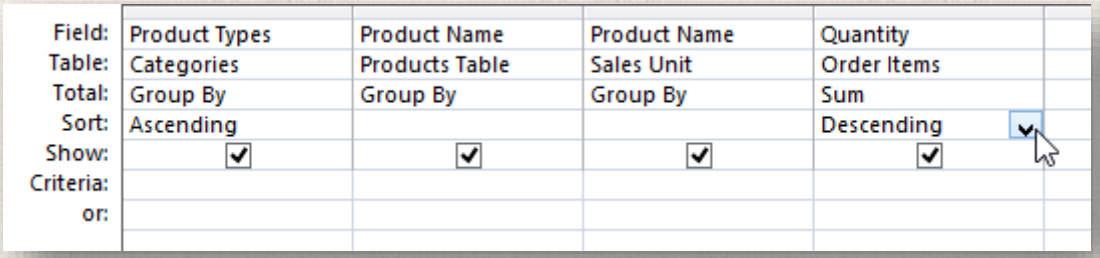
For our example, we want to find the total number we've sold of each of our menu items, so we'll use a query showing us all of the menu items we've sold. If you want to follow along in our database, open the **Menu Items Ordered** query.

1. Create or open a query you want to use as a **totals query**.
2. From the **Design** tab, locate the **Show/Hide** group, then select the **Totals** command.



Creating a totals query

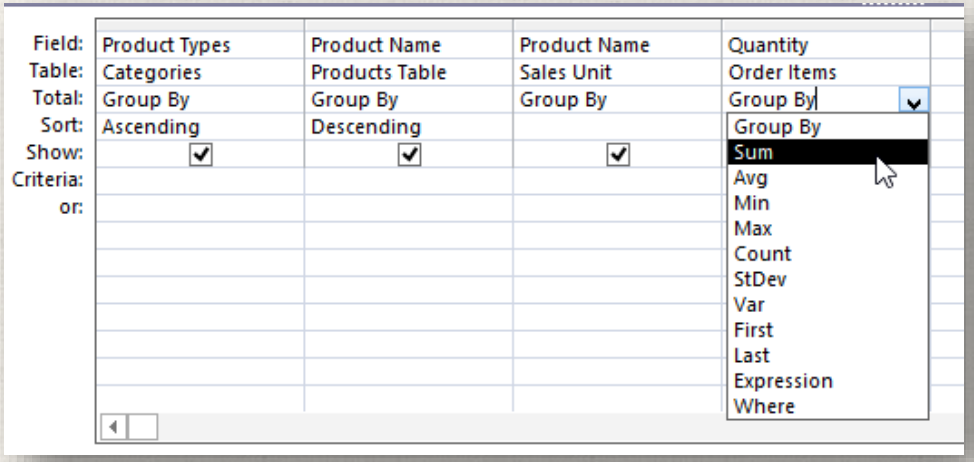
3. A row will be added to the table in the **design grid**, with all values in that row set to **Group By**. Select the cell in the **Total:** row of the field you want to perform a calculation on, then click the drop-down arrow that appears.



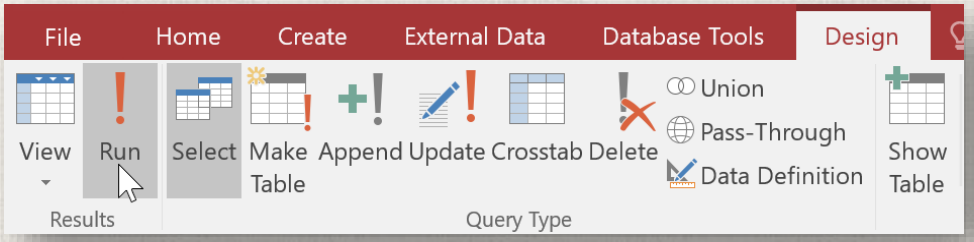
Field:	Product Types	Product Name	Product Name	Quantity
Table:	Categories	Products Table	Sales Unit	Order Items
Total:	Group By	Group By	Group By	Sum
Sort:	Ascending			Descending
Show:	☑	☑	☑	☑
Criteria:				
or:				

Creating a totals query

4. Select the calculation you want to be performed in that field. In our example, we want to **add** the quantities of products we've sold, so we'll select the **Sum** option.



5. When you are satisfied with your query design, select the **Run** command on the **Query Tools Design** tab to **run** the query.



Creating a totals query

6. 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 on the Quick Access Toolbar.

Menu Items Ordered

Product Types	Products Table.Product Name	Sales Unit.Product Name	SumOfQuantity
Cakes	Cheesecake	Single	20
Cakes	Buche de Noel (Christmas Cake)- Winter	Single	12
Cakes	Carrot Cake	Single	9
Cakes	Black Forest	Single	8
Cakes	Black Walnut	Single	5
Cakes	Italian Rum	Single	4
Cakes	Gingerbread - Winter	Single	4
Cakes	Coconut	Single	2
Cakes	French french vanilla	Single	2
Cakes	German Chocolate	Single	2
Cakes	Red Velvet	Single	1
Cakes	Cookies n' Cream	Single	1
Cookies	Fudge Brownie	One Dozen	7
Cookies	Fudge Chocolate	Single	6
Cookies	Ginger Shortbread	One Dozen	6
Cookies	Chocolate Chip	Single	5

More query options

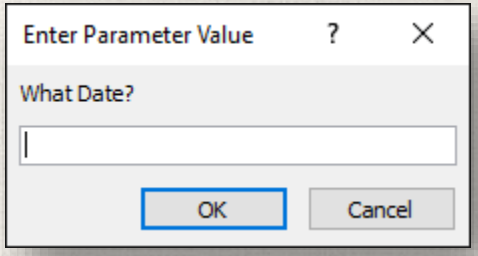
- A **parameter query** allows you to create a query that can be updated easily to reflect a new criterion, or **search term**. When you open a parameter query, Access will prompt you for a search term and then show you query results that reflect that search.
- A **find duplicates query** lets you find all **duplicate records** in your database so you can **delete** them. Duplicate records can negatively affect the **integrity** of your database.

Creating a parameter query

- A **parameter query** is one of the simplest and most useful queries you can create. Because parameter queries are so simple, they can be easily updated to reflect a new **search term**. When you open a parameter query, Access will prompt you for a search term and show you query results that reflect your search.

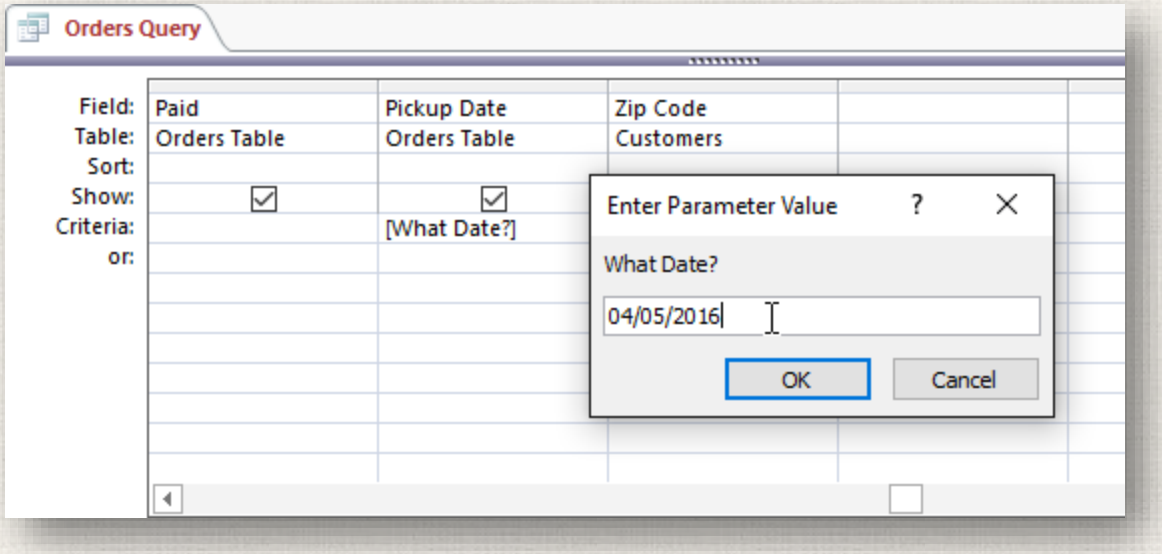
Creating a parameter query

- When you're running parameter queries, search terms act as **variable criteria**, which are query criteria that **change** each time you run the query. For instance, let's say we own a bakery and want to create a query that will quickly look up orders that were placed on a certain date. We could create a parameter query with variable criteria in the **Date** field. This way, each time we run the query a dialog box will appear prompting us to enter the date we'd like our query to search for.
- We'll enter the date we want, then Access will run the query using the date we entered as a search term.



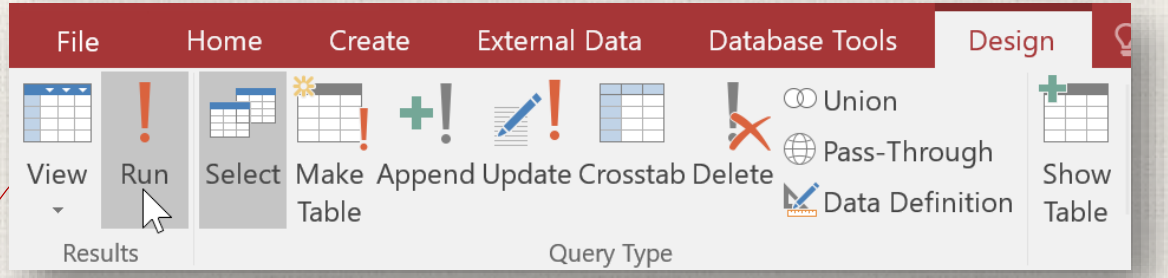
Creating a parameter query

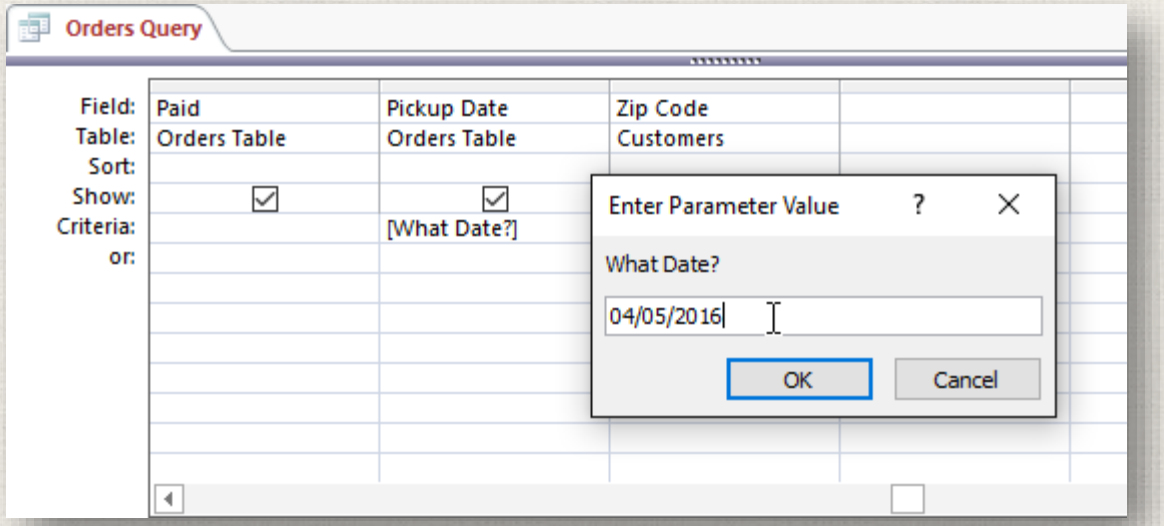
1. Locate the field or fields where you want the variable criteria to appear, then select the **Criteria:** row.
2. Type the phrase you want to appear in the prompt that will pop up each time you run your query. Make sure to enclose the phrase in brackets **[ ]**. For example, in our parameter query that searches for orders placed on a certain date, we might type our criteria like this: **[What Date?]**.



Creating a parameter query

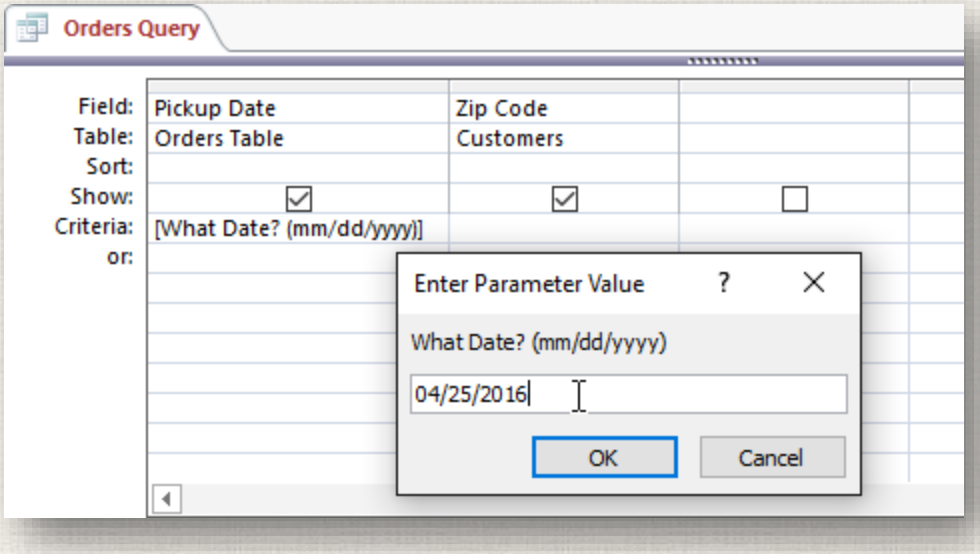
3. On the **Design** tab, click the **Run** command to **run** your query. A dialog box will appear with the specified prompt. Enter your search term and click **OK** to view your query results.





Tips for writing parameter queries

- Ideally, the prompt you create for your query should make it clear what **type** of information the search term should be, as well as the desired **format**. For example, to guarantee users enter a search for a date in the format used in our database, we could write the following in the **Criteria:** row of the **Pickup Date** field like this: **[What Date? (mm/dd/yyyy)]** .

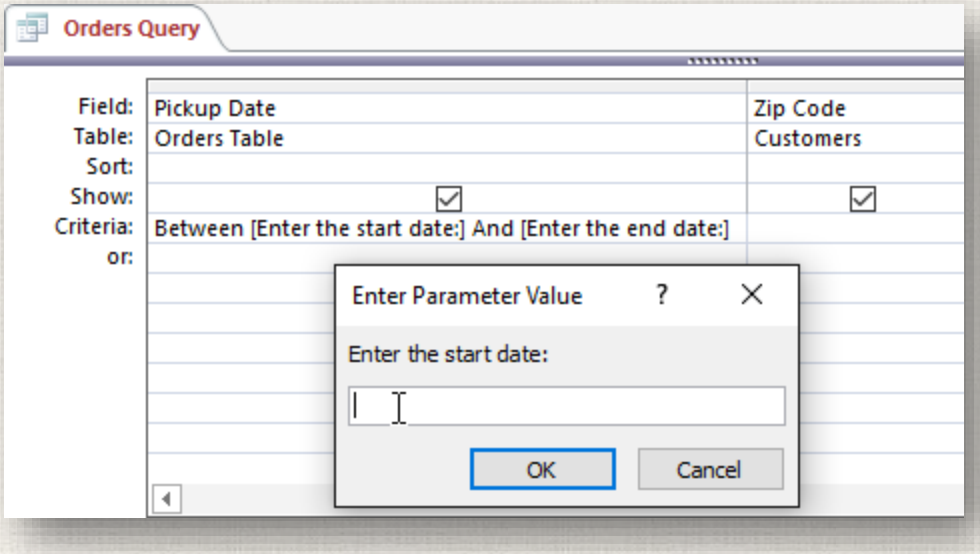


Tips for writing parameter queries

- The simplest parameter query will give you an exact-match criteria, meaning the query will search for the **exact text** you enter in the prompt. However, you can turn any type of criteria into a variable criteria. Simply type your prompt text in brackets in the part of the criteria where you would normally put a search term.

Tips for writing parameter queries

- For example, in a normal query we could find orders that were placed **between** two dates by using the criteria **Between x AND y** and replacing **x** and **y** with the first and second dates, respectively. To turn this into a parameter criteria, we would simply replace the **x** and **y** with the text we want to appear in the prompt. Our variable criteria might look like this: **Between [Enter the start date:] And [Enter the end date:]**. These two prompts will appear when you run the query.



Creating a Find Duplicates query

- A **find duplicates query** allows you to search for and identify **duplicate records** within a table or tables. A duplicate record is a record that refers to the **same thing** or **person** as another record.
- Not all records containing similar information are duplicates.
  - For instance, records of two orders that were placed on different dates but that contained identical items would **not** be duplicate records. Likewise, not all duplicate records contain completely identical information.
  - For example, two customer records could refer to the same person but include different addresses. The record with the out-of-date address would be the duplicate record.

Creating a Find Duplicates query

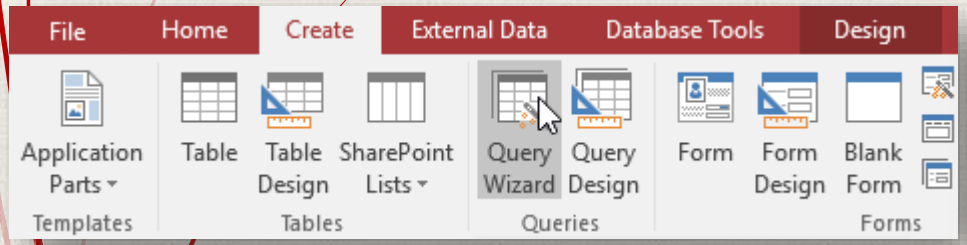
- Why is getting rid of duplicate records so important? Consider the example above. If we had multiple records for one customer, it would be difficult to view an order history for him because the information would be spread across several unlinked records. We might even deliver his order to the wrong address if the person entering the order information selects an outdated record. It's easy to see how having duplicate records can undermine the integrity and usefulness of your database.

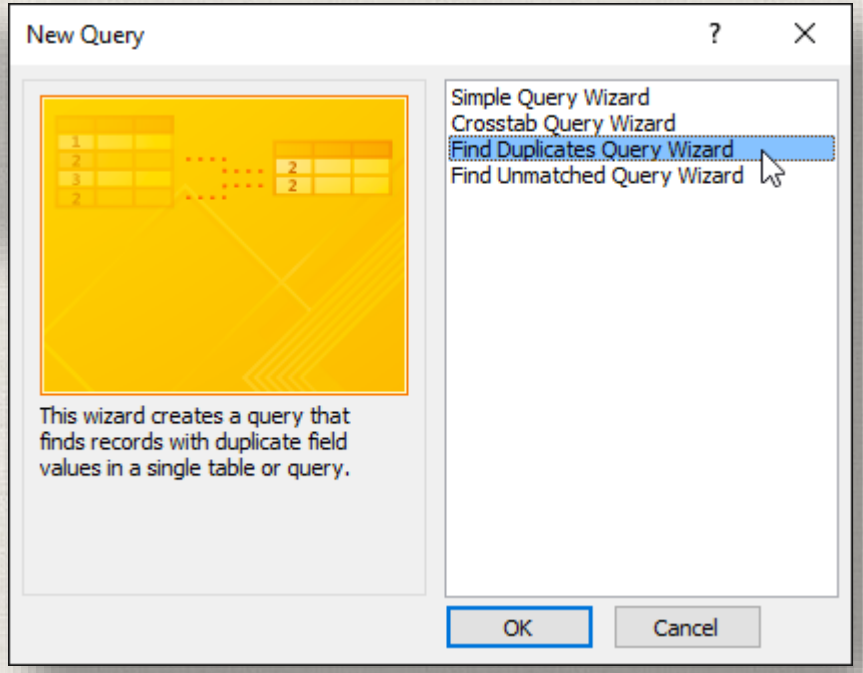
Creating a Find Duplicates query

- Fortunately, Access makes it easy to search for and locate potential duplicate records. Note that Access won't delete the records for you or help you figure out which one is current—you'll have to do those things for yourself. If you're familiar with the data in your database, though, getting rid of duplicate records will be a manageable task.

Creating a Find Duplicates query

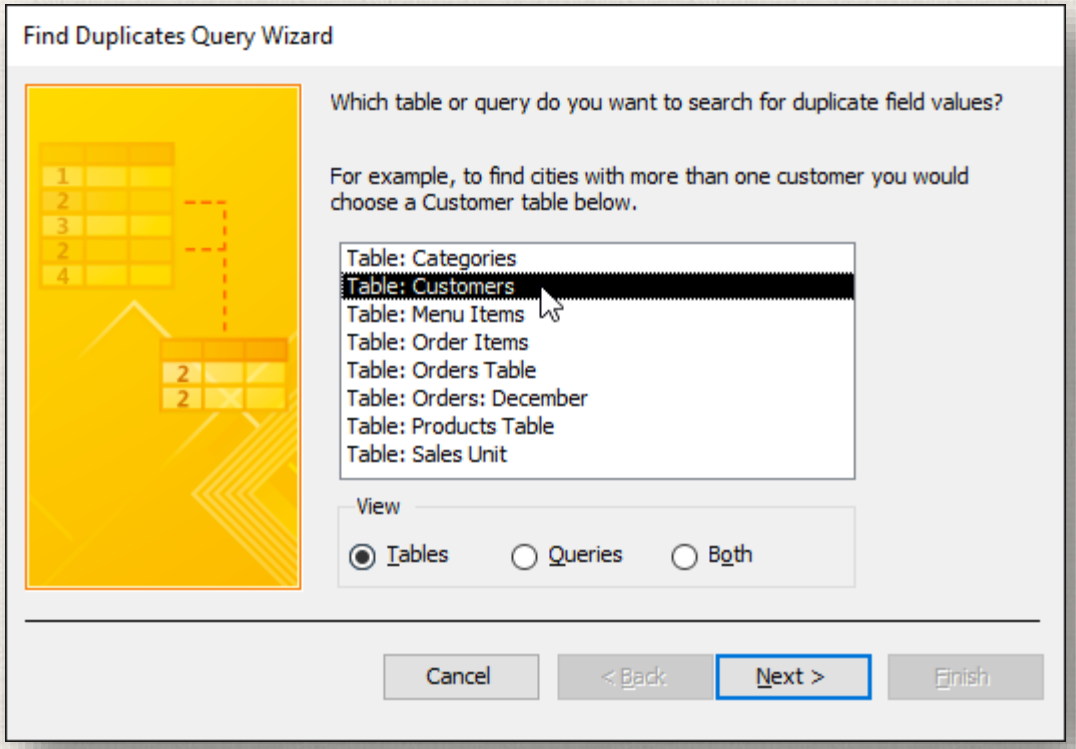
1. Select the **Create** tab on the Ribbon, locate the **Queries** group, and click the **Query Wizard** command.
2. The **New Query** dialog box will appear. Select **Find Duplicates Query Wizard** from the list of queries, then click **OK**.





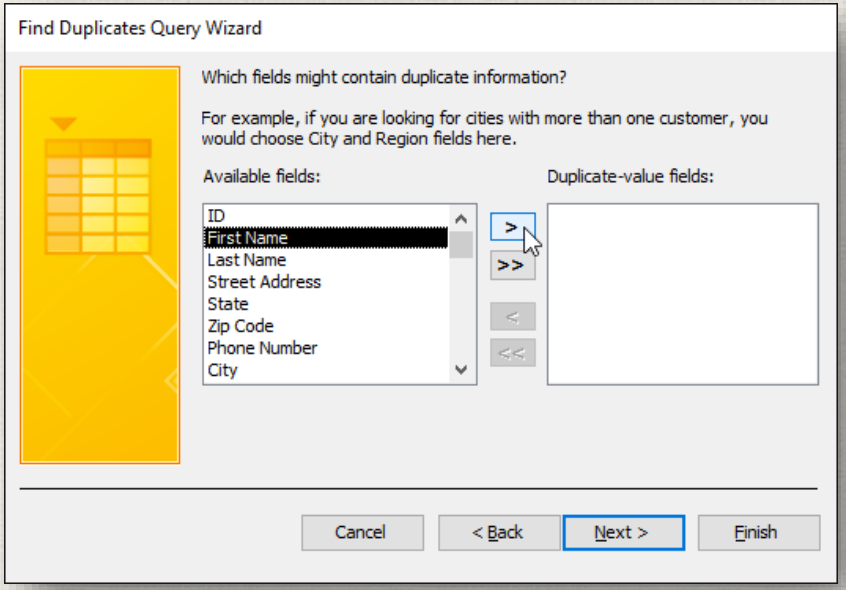
Creating a Find Duplicates query

3. Select the table you want to search for duplicate records, then click **Next**. We're searching for duplicate customer records, so we'll select the **Customers** table.



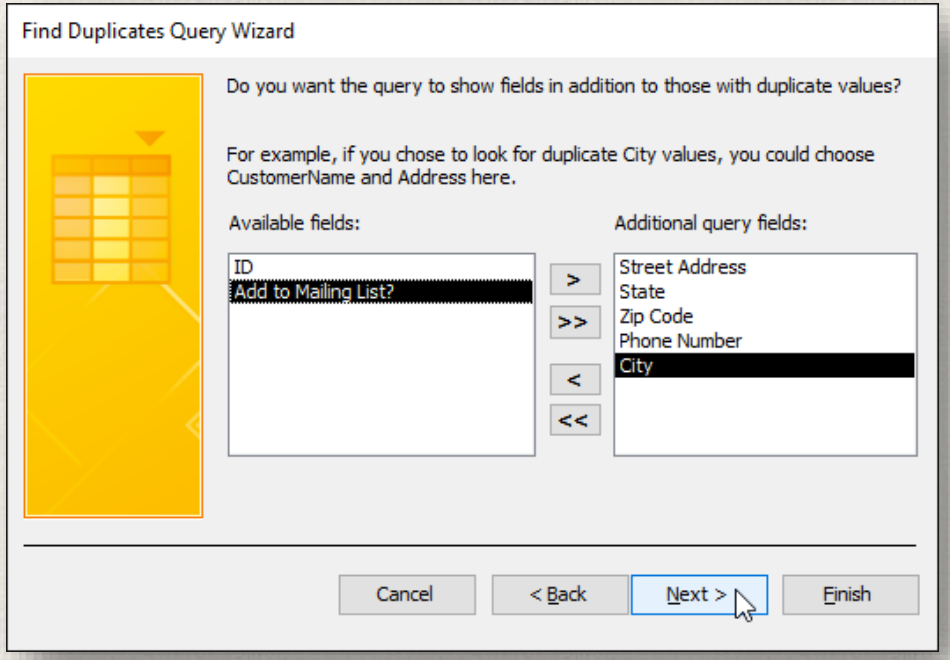
Creating a Find Duplicates query

4. Choose the fields you want to search for duplicate information by selecting them and clicking the **right arrow button**. Only select fields that should not be identical in nonduplicate records. For instance, because we're searching for duplicate customers we'll only select the **First Name** and **Last Name** fields because it's unlikely that multiple people with the exact same first and last names would place orders at our bakery.
5. When you've added the desired fields, click **Next**.



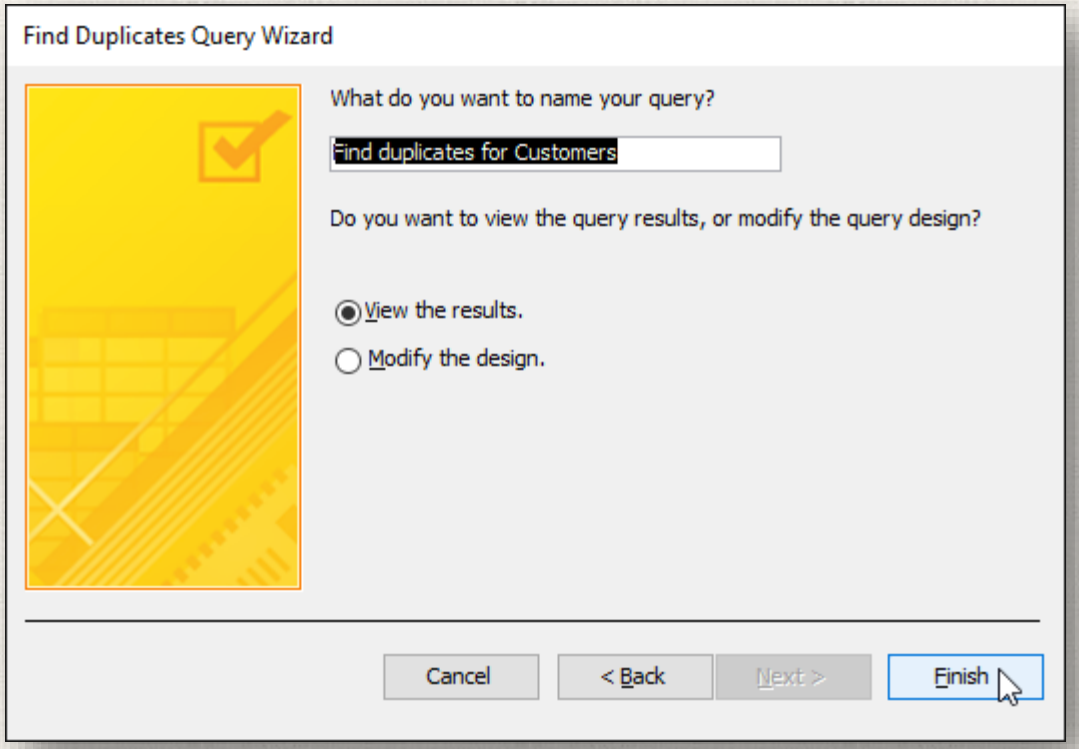
Creating a Find Duplicates query

6. Select additional fields to view in the query results. Choose fields that will help you distinguish between the duplicate records, and choose which one you want to keep. In our example, we'll add all of the fields relating to customer **addresses**, plus the **Phone Number** field because records with identical customer names might contain nonidentical information in this field. When you're satisfied, click **Next**.



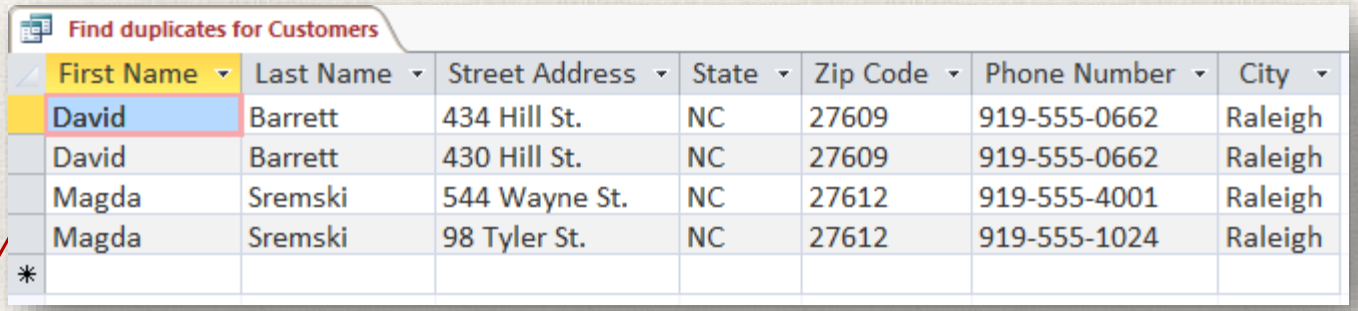
Creating a Find Duplicates query

7. Access will suggest a name for your query, but you can type a different name if you want. When you're satisfied with the query name, click **Finish** to run your query.



Creating a Find Duplicates query

8. If Access found any duplicate records in your query, they will be displayed in the **query results**. Review the records and **delete** any outdated or incorrect records as needed.



Find duplicates for Customers

First Name	Last Name	Street Address	State	Zip Code	Phone Number	City
David	Barrett	434 Hill St.	NC	27609	919-555-0662	Raleigh
David	Barrett	430 Hill St.	NC	27609	919-555-0662	Raleigh
Magda	Sremski	544 Wayne St.	NC	27612	919-555-4001	Raleigh
Magda	Sremski	98 Tyler St.	NC	27612	919-555-1024	Raleigh
*						

Tips for resolving duplicate records

- **Save** your duplicate records queries, and run them often.
- **Investigate** potential duplicate records by looking at linked data in other tables. You can do this by searching for these records' **record ID numbers** in related tables. Is one record linked to mostly old orders while another contains recent ones? The latter is likely to be the current one.
- Once you decide which record to delete, make sure you won't be losing any information you might need. In our example, before we deleted our duplicate record we found all of the orders linked to that record's **ID number** and replaced them with the ID number of the record we decided to keep.

Practice

- Open **practice database**.
- Open the **Customers Who've Ordered from Nearby Towns** query, and switch to **Design view**.
- Add a **Totals** row to the query.
- Set the Totals row in the **Orders Table ID** field to **Count**. This will let us count how many orders each customer has placed.
- In the **Customers** table in the **Object Relationship pane**, double-click the word **City** to add another City field to the design grid below.
- **Click** and **drag** the City field you just added so it is to the **left** of the **First Name** field. It should now be the **leftmost field** in the design grid.
- Apply the following multilevel sort:
  In the leftmost **City** field, apply an **ascending** sort.
  In the **Last Name** field, apply an **ascending** sort.
- **Hide** the leftmost **City** field.
- **Run** the query. If you did it correctly, there should be 14 records in the query results. The first record should look like this:

First Name	Last Name	Street Address	State	Zip Code	CountOfID	Phone Number
Katharine	Kellerman	76 Murphy Ave.	NC	27513	1	919-555-4526

THE END