Microsoft Access – SQL
(Structured Query Language)

Recommended Reading

The best resources for learning SQL are:

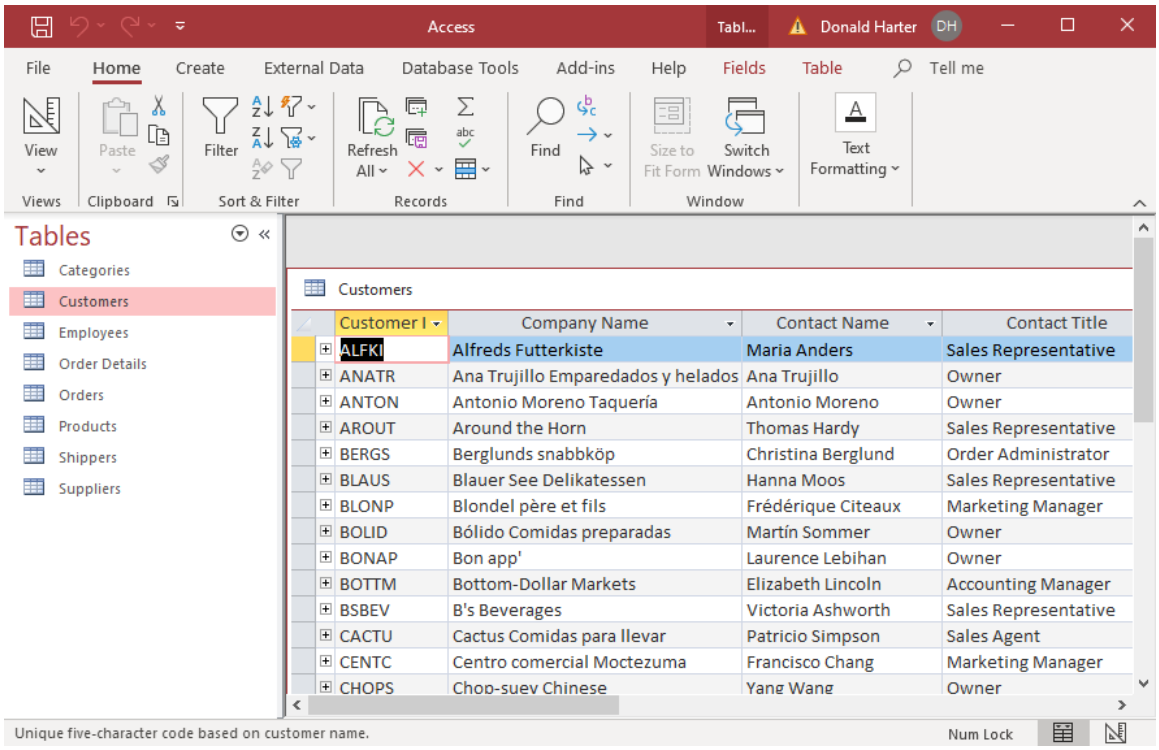
1. Sams Teach Yourself SQL in 10 Minutes by Ben Forta, 2004
2. SQL Guide (Quickstudy: Computer) by Inc. Barcharts, 2005
3. Lynda.com

Download files from BlackBoard

Download the Orders database and the Recruiting database from BlackBoard by right clicking and saving to your desktop.

Viewing the contents of the data base

1. Open the database Orders.accdb.
2. Click Enable, then Yes for trust.
3. Once the list of data base tables appears (customers, orders, order details, etc.), double click on customers to open the Customers table.
4. After viewing the contents of the Customers table, close it by clicking on the X in the upper right corner of the customers table screen.
5. Similarly, open the Orders and Order Details tables.

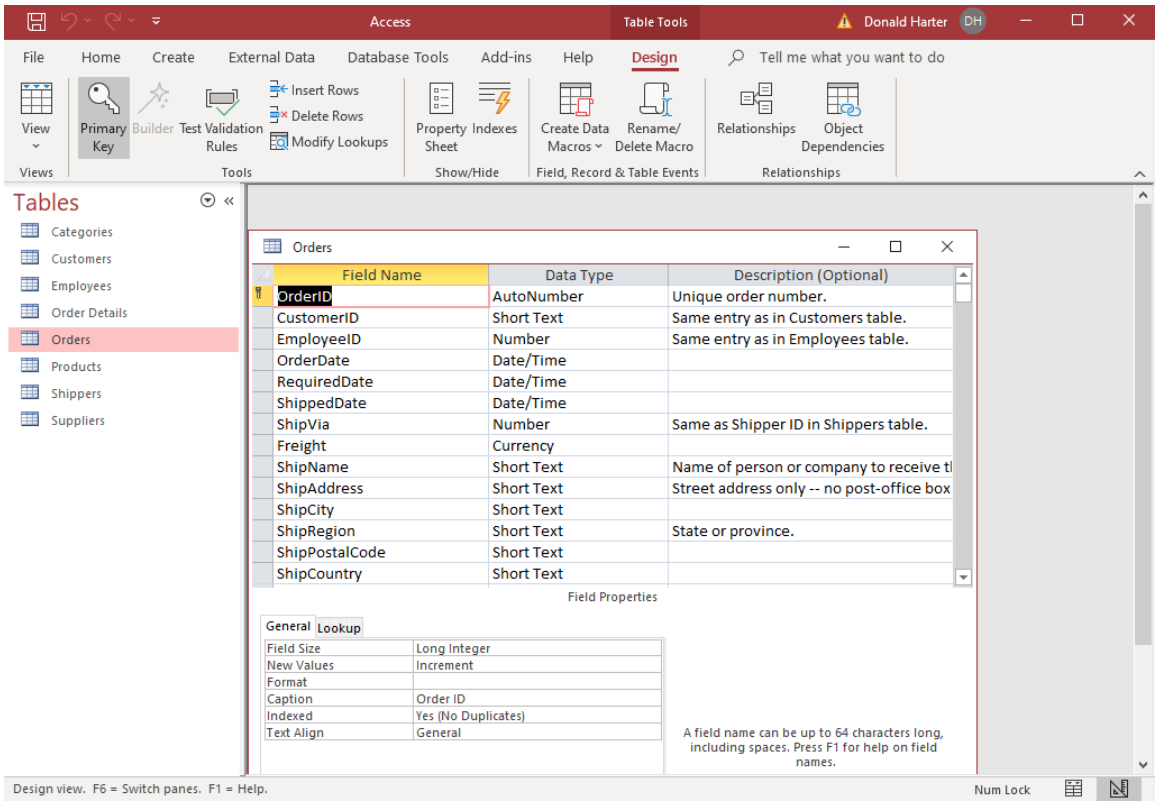


6. Terminology
    a. Table: set of data in columns and rows
    b. Field: column
    c. Record: row

Viewing the design of a table

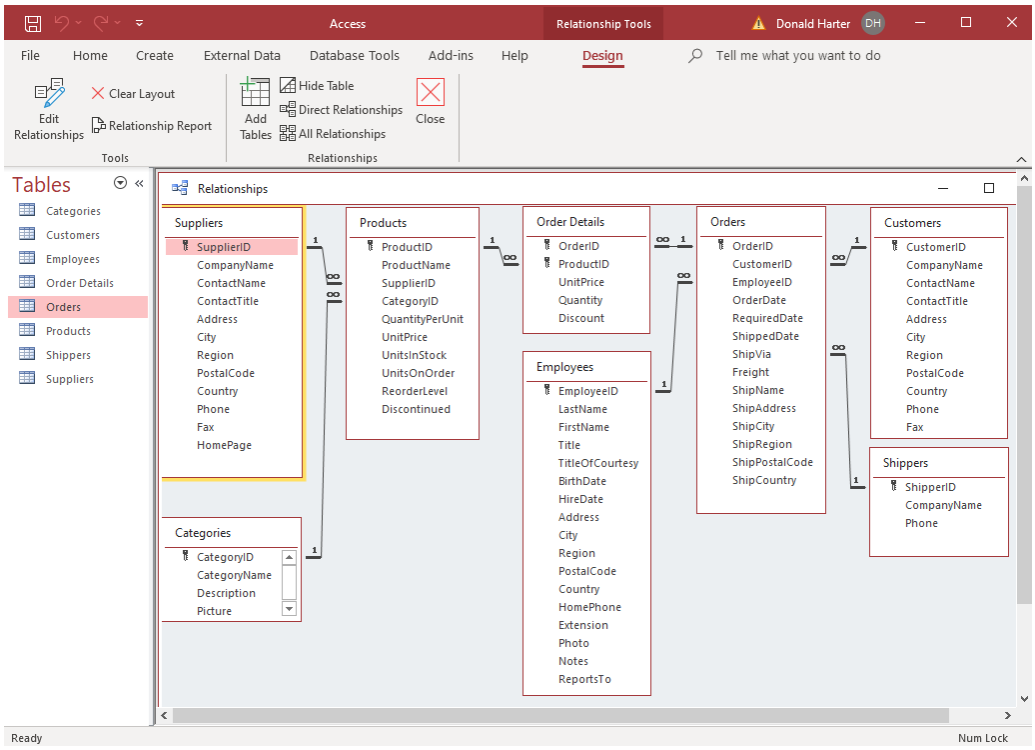
It is important to understand how data is store in the data base. Data can be stored as text or numbers, in a variety of formats. To view the design of the Orders table:

1. Double click on the Orders table to open it.
2. Click on View > Design View.
3. A data field can have one of several data types:
    a. Autonumber: automatically increments when you add new data
    b. Text: alphanumeric data
    c. Number: numeric data
    d. Date/Time: Year, month, day, time
4. Note that the Field Name “OrderDate” has no space between the words, but the Caption is “Order Date”. The field name is used in SQL, but the caption will appear as the heading in a table
5. To return to the Datasheet View, click on View > Datasheet View
6. Close the table after viewing the Data Types

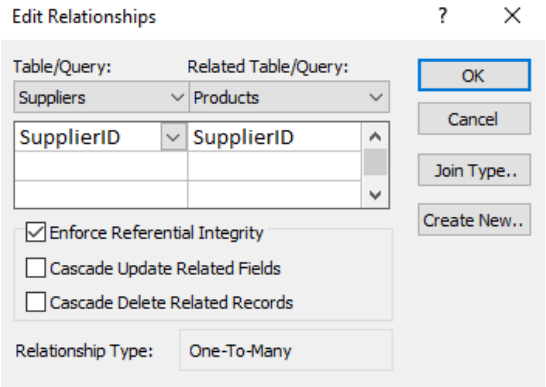


Viewing the structure of the data base

1. Now click on Database Tools, then Relationships
2. You should see all of the tables, data elements in the tables, key fields, and links between tables. You will need to refer to this during this lab.
3. A key field uniquely identifies each record in the table. Key fields have “keys” next to them



4. Double click on the relationship line between Suppliers and Products
    a. Click on Join Type to confirm that this is an inner join
    b. What other types of join are there? Click cancel to return.
    c. What is the Relationship Type? Click Cancel to close the Edit Relationship screen, and cancel again



5. If you want close the relationships, click on X to close the relationships screen. In this case, keep the relationship screen open for reference.

SQL commands for Queries

The essential SQL commands for queries are:

SELECT	identify fields to be reported
AS	creates an alias for reporting labels
FROM	identify tables to be used
WHERE	filter criteria
GROUP BY	clustering criteria
HAVING	additional criteria based on a calculation
ORDER BY	sorting criteria
IN	part of a list
BETWEEN	range of numbers
IS NULL	empty values
LIKE	matches string of characters

Order of SQL commands

When building SQL commands, the order of words is important. In English, the order is generally:

Noun => Verb => Object

In SQL the order is:

SELECT
FROM
WHERE
GROUP BY
HAVING
ORDER BY

Failure to follow this order will result in a syntax error.

In the following examples, we will enter the commands in uppercase letters, only to make them stand out. You do not need to use uppercase, but it makes the SQL easier to read. Also, each command will be entered on a separate line, only to make it easier to read.

All SQL commands end with a semicolon (;).

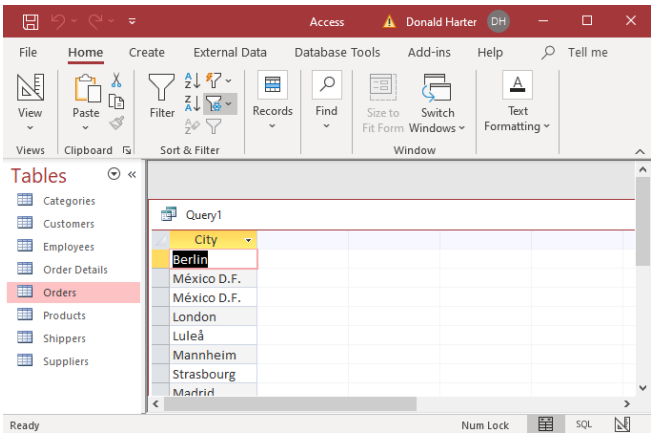
Simple query from table

First retrieve all the cities where we have customers. Returning to the Relationships view, which table has customer information? The table is Customers. What is the name of the field (column) that holds city information? Field is City.

1. Returning to the data base screen (you should see the list of tables again); click on Create, Query Design.
2. At the top left of the screen, click on SQL View, then SQL View again. Notice that you can toggle between SQL and Design views.
3. You should see “SELECT;”. Delete this so we can start from scratch.
4. Type in the following SQL command. Notice that it ends with a semi-colon.

```
SELECT City
FROM Customers;
```

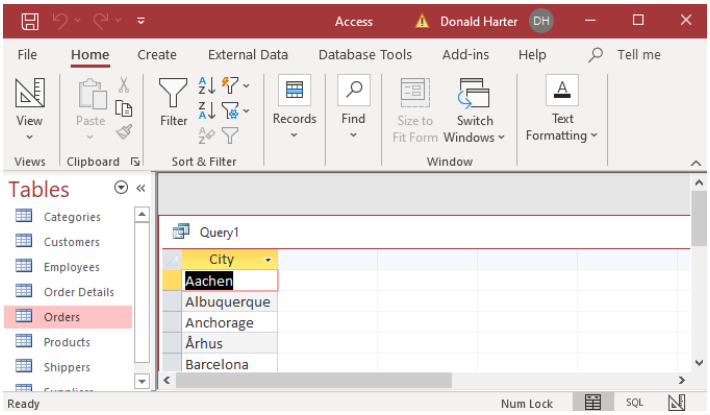
5. At the top of the screen, click on View, Datasheet View. (You could also click on Run!)
6. This query will select all city names from the customer table. Notice that Mexico D.F. and London appear multiple times.



7. To eliminate duplicates, click on View, SQL, now use the following query.

```
SELECT DISTINCT City
FROM Customers;
```

8. Click Run!
9. Duplicates have now been eliminated.

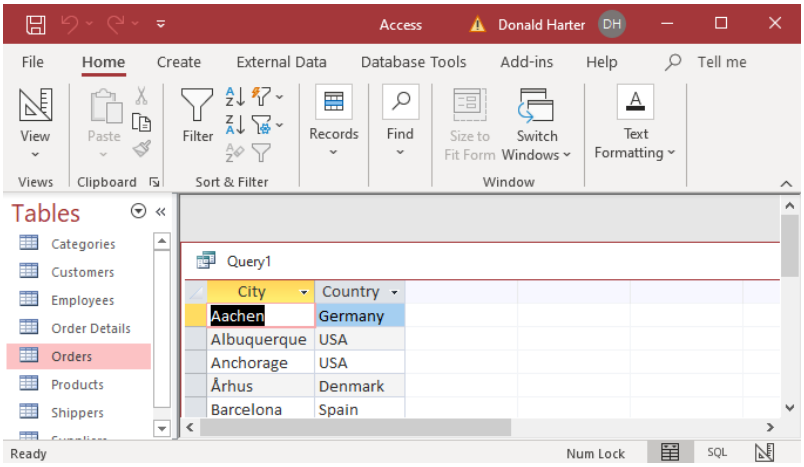


Query retrieving two or more columns

1. To retrieve more than one field of data, put commas between the data field names
2. Retrieve City and Country from the Customers table

```
SELECT DISTINCT City, Country
FROM Customers;
```

3. Click Run!



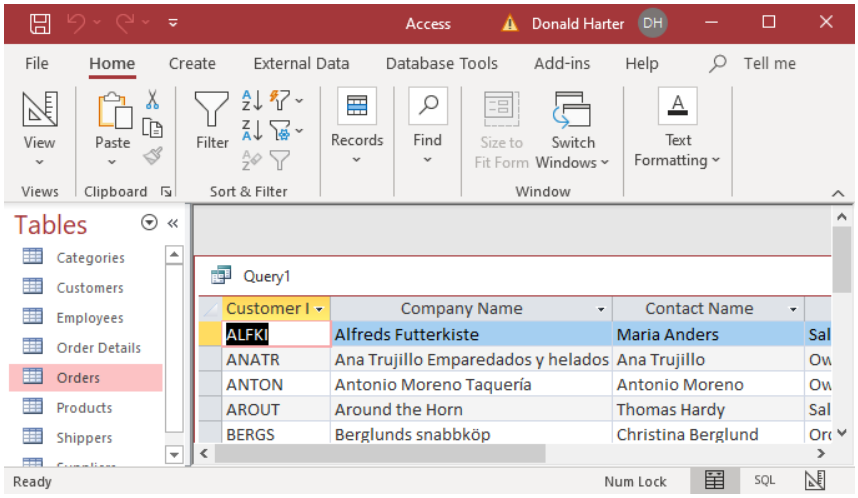
Retrieving All Columns from a Table

Sometimes you might want to display everything in a table. Use the wildcard character * for this.

1. To retrieve all columns in the Customers table, enter the SQL:

```
SELECT *
FROM Customers
```

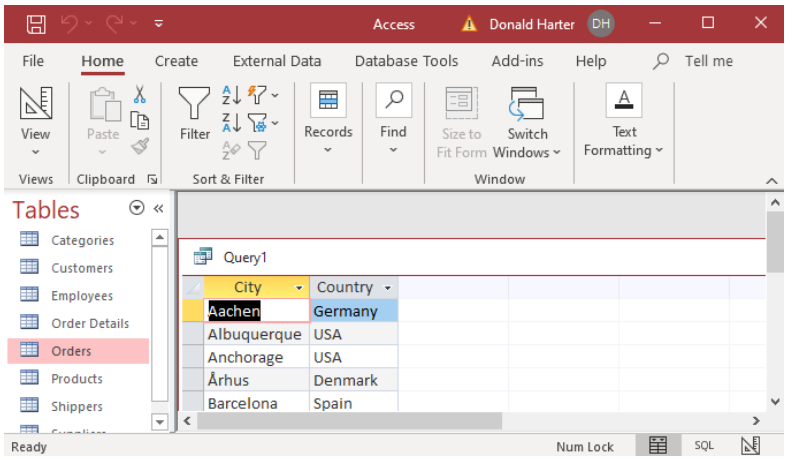
2. Click Run!



Sorting Records in a Retrieval

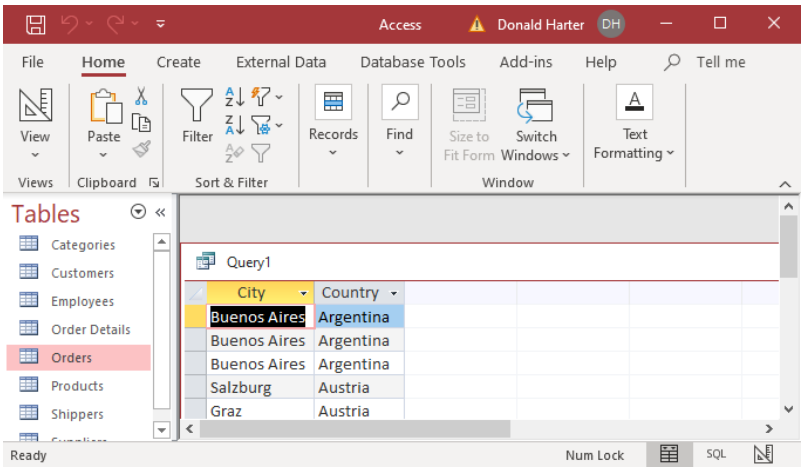
1. To sort the output, use the ORDER BY command. In this first example, display City and Country from the Customers table, sorting by City

```
SELECT City, Country
FROM Customers
ORDER BY City;
```



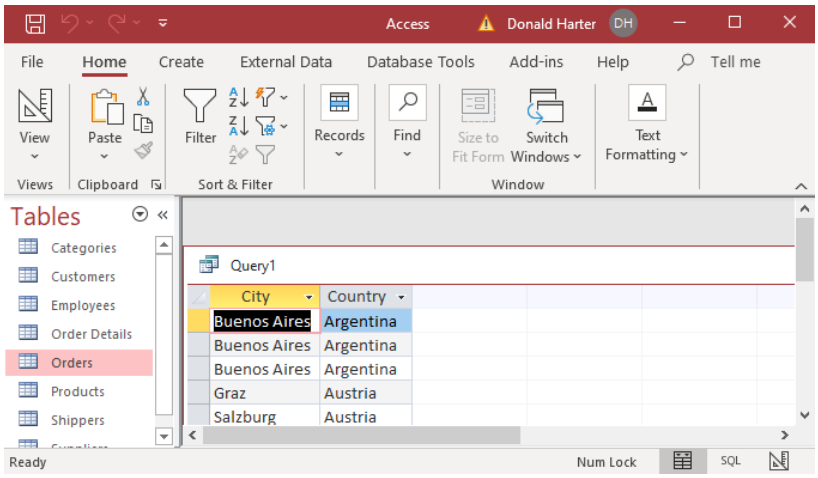
2. Now, order by Country
3. Click on View, SQL to return to the SQL screen

```
SELECT City, Country
FROM Customers
ORDER BY Country;
```



4. Click on View, SQL to return to the SQL screen
5. You can have multiple sort fields, where the first field is sorted first, the second field is second, etc. In this example, sort by Country first, then City.

```
SELECT City, Country
FROM Customers
ORDER BY Country, City;
```

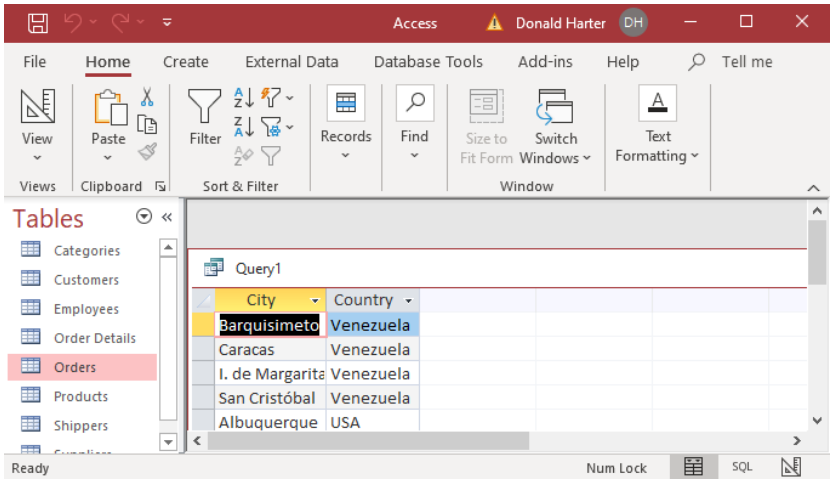


6. Alternatively, you can use the position order to specify the sort order.

```
SELECT City, Country
FROM Customers
ORDER BY 2, 1;
```

7. You can reverse the order of the sort by using the DESC (descending) command after the appropriate sort term. In this example, use a descending sort on Country, but keep the default ascending sort for City (no command necessary).

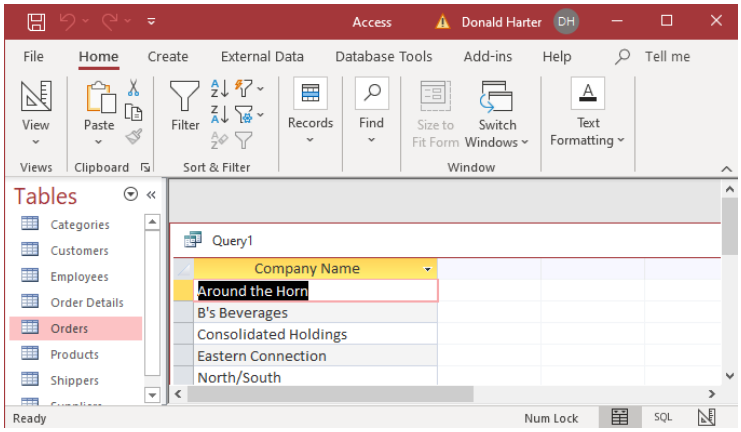
```
SELECT City, Country
FROM Customers
ORDER BY Country DESC, City;
```



Filters using Where Clauses

1. Now identify all customer companies in London. Use the following command. Notice that the column names (e.g., CompanyName) has no space in it, and that London is in double quotes. In a Where clause, number filters do not use quotes, text fields use quotes.

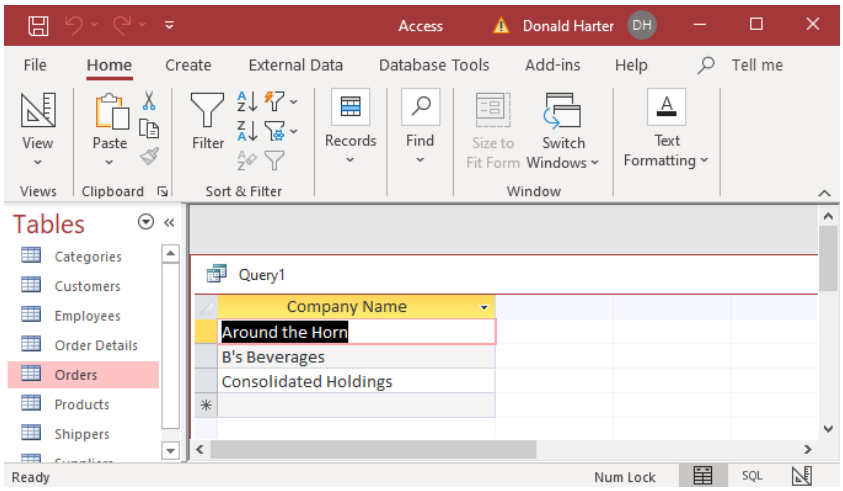
   SELECT CompanyName
   FROM Customers
   WHERE City="London";



Filters using Where Clauses with Boolean Operators

1. Now select CompanyName where the City is London and the ContactTitle is Sales Representative. Add quotation marks around Sales Representative. Enter the Boolean operator AND between the criteria.
2. Why is there no space in CompanyName and ContactTitle, but there is a space in Sales Representative?

   SELECT CompanyName
   FROM Customers
   WHERE City="London" AND ContactTitle="Sales Representative";

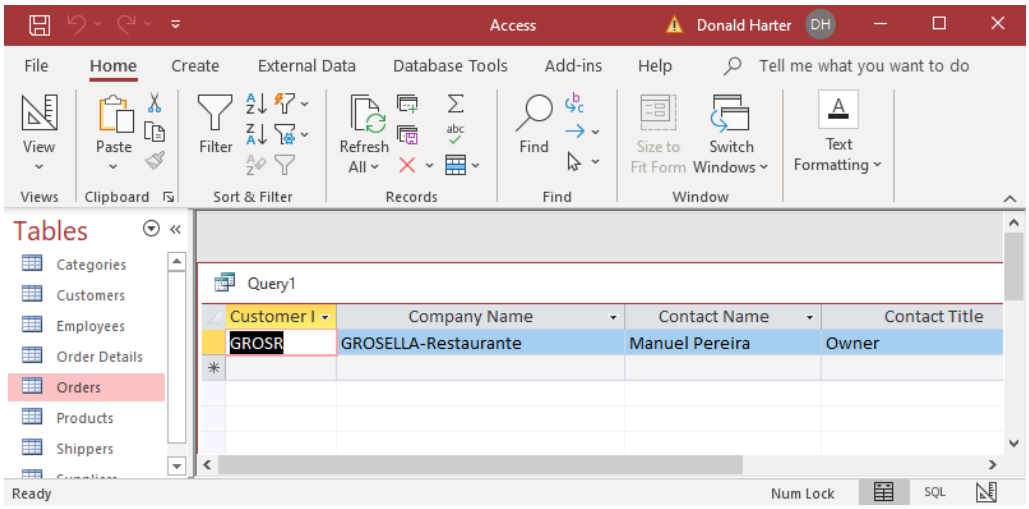


Selecting all data using Where Clauses with Boolean Operators

What if you wanted to find out everything about each company where the company contact was the owner and the company was in Caracas?

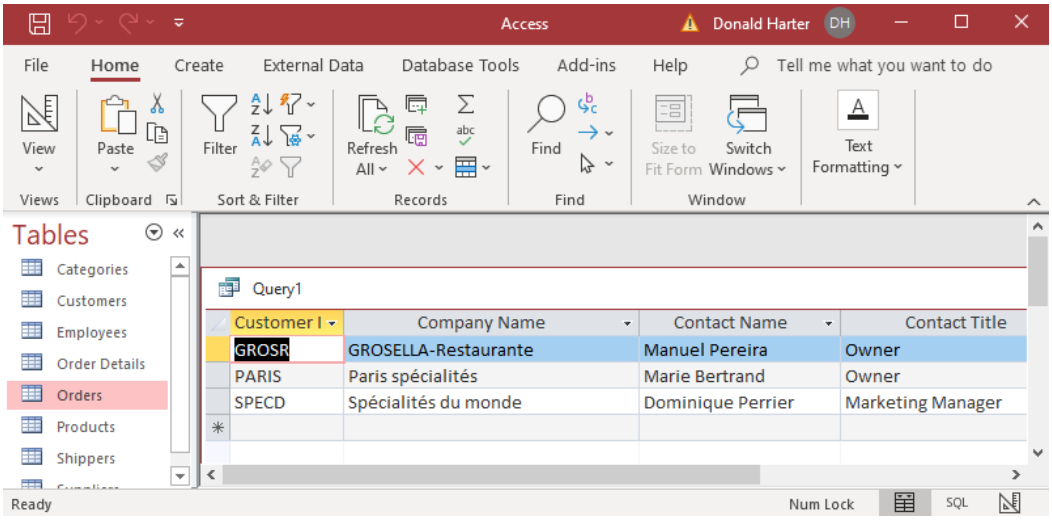
1. Recall that SELECT * will select all fields from a table
2. You can use Boolean operators to filter on several columns at once:

```
SELECT *
FROM Customers
WHERE ContactTitle="Owner" AND City="Caracas";
```



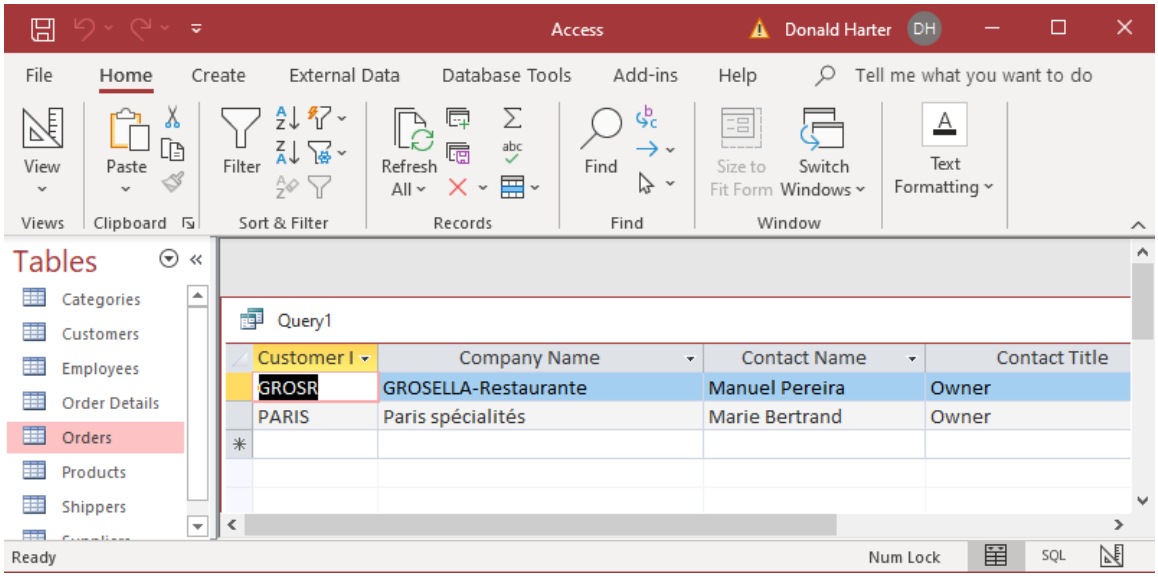
3. Now find Owners in Caracas or Paris

```
SELECT *
FROM Customers
WHERE ContactTitle="Owner" AND City="Caracas" OR City="Paris";
```



4. What went wrong? The order of operation of AND and OR conditions executes ANDs before ORs. To group the conditions properly, you must put parentheses around the conditions. SQL executes within the parentheses first, then works outward.

```
SELECT *
FROM Customers
WHERE ContactTitle="Owner" AND (City="Caracas" OR City="Paris");
```



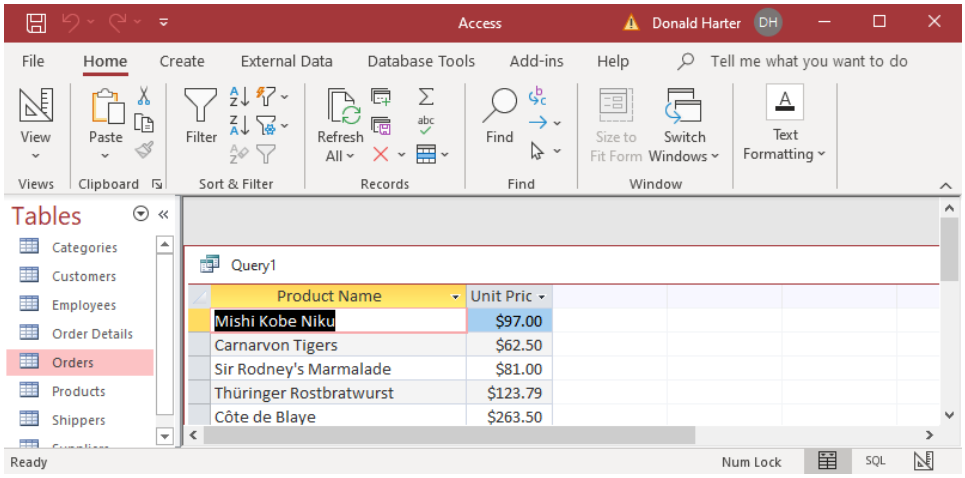
Filters Using Numbers

1. To retrieve records using a filter with conditions on numbers, use the following operators:

=	equal
<>	not equal
!=	not equal (*not available in Access*)
<	less than
!<	not less than (*not available in Access*)
<=	less than or equal
>	greater than
!>	not greater than (*not available in Access*)
>=	greater than or equal
BETWEEN	between two numbers (e.g., BETWEEN 10 AND 20)
IS NULL	is a null or no value

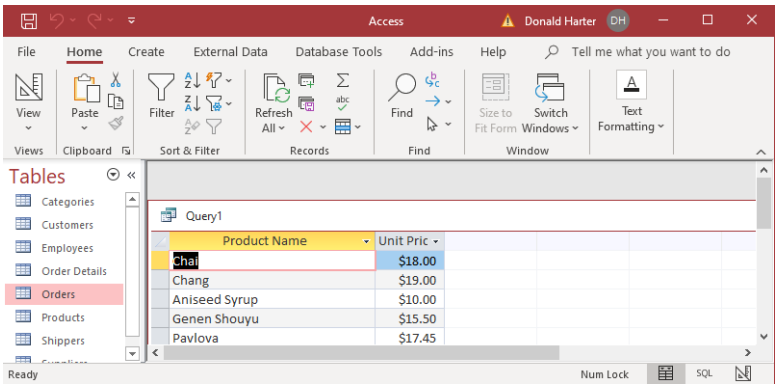
2. For this example, retrieve ProductName and UnitPrice from the Products table where a UnitPrice greater than 50.

```
SELECT ProductName, UnitPrice
FROM Products
WHERE UnitPrice>50;
```



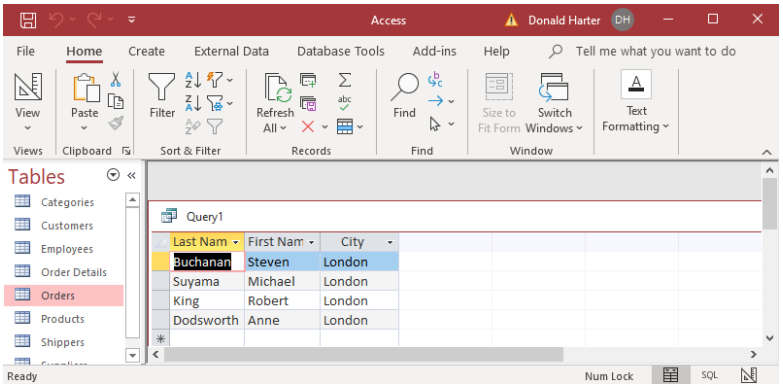
3. Now retrieve ProductName with UnitPrice BETWEEN 10 AND 20

```
SELECT ProductName, UnitPrice
FROM Products
WHERE UnitPrice BETWEEN 10 AND 20;
```



4. The Employees table has missing data in the Region field. Why? Retrieve the LastName and FirstName of Employees, and the City of Employees, where the Region is null

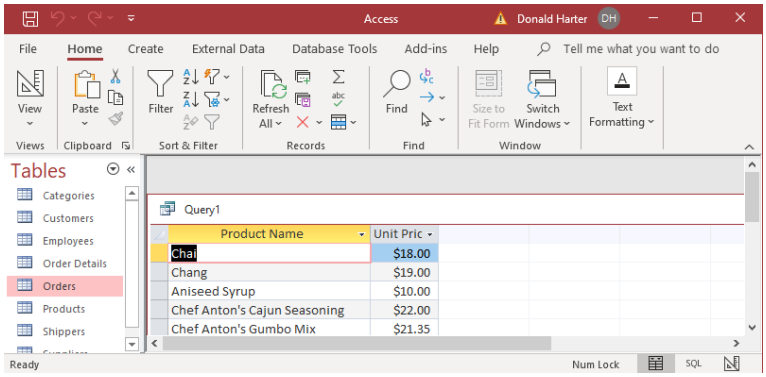
```
SELECT LastName, FirstName, City
FROM Employees
WHERE Region IS NULL;
```



NOT Operator

1. The NOT operator can be placed in a where clause to negate the condition; in this case NOT > 50 means <= 50.

```
SELECT ProductName, UnitPrice
FROM Products
WHERE NOT UnitPrice>50;
```



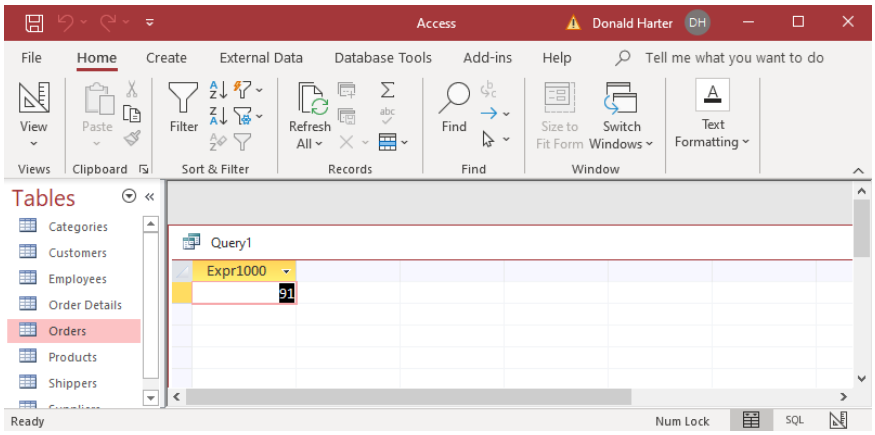
Mathematical Operations

Next we will use some mathematical operations. These include count, sum, and avg (average). To use a mathematical operation, type the math expression, then (field). If you want to count the number of rows in a table, use count(*).

1. First let's count the number of records in the Customers table.

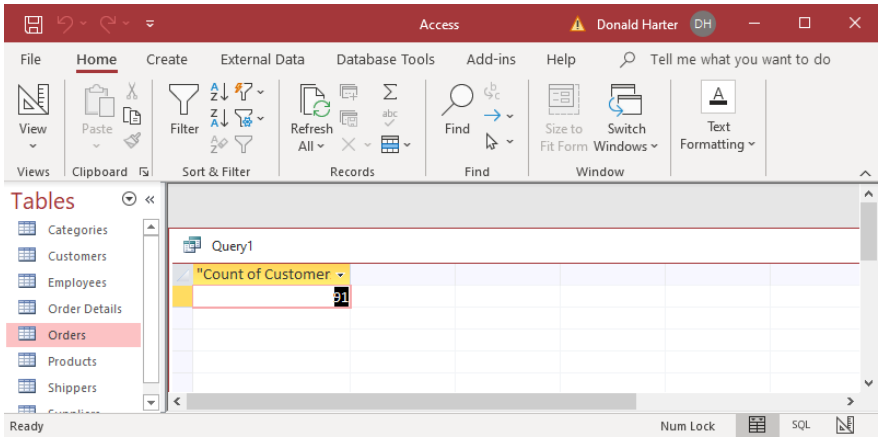
   SELECT count(*)
   FROM Customers;

2. You should have counted 91. You can check this by looking at the contents of the table again. At the bottom of the screen, it tells you the number of records.



3. The "Expr1000" is not very informative. Let's create a new label using the AS clause.

   SELECT count(*) AS "Count of Customers"
   FROM Customers;



Grouping and Summarizing Data

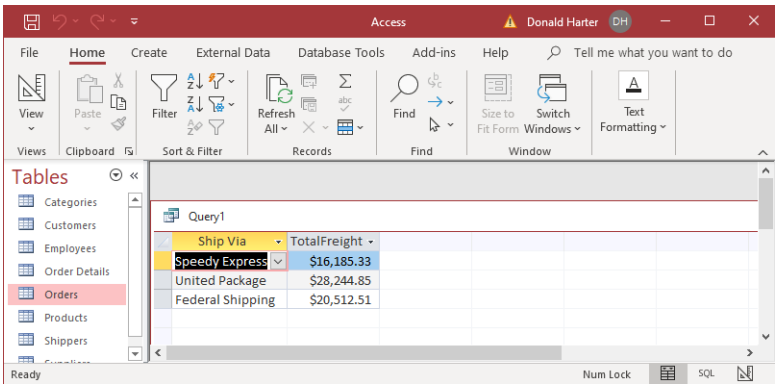
You can also perform math operations on subsets of the table and report them separately. Summarization functions include:

AVG	average of column values
COUNT	number of records
MIN	minimum of column values
MAX	maximum of column values
SUM	sum of column values

1. For example, next look at the orders table and calculate how much each shipper collected in shipping charges.

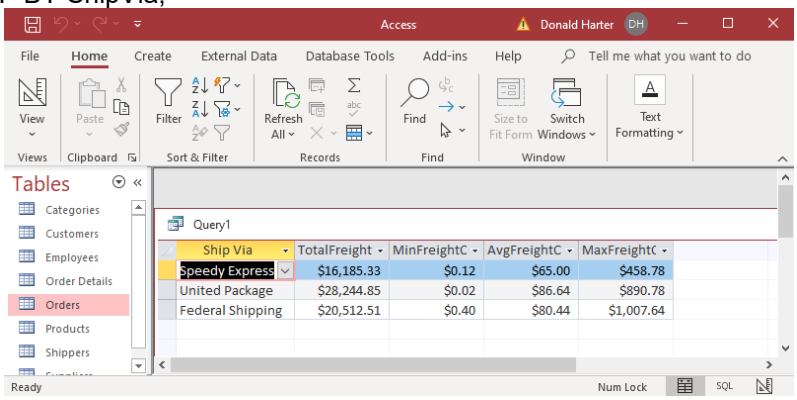
   ```
   SELECT ShipVia, SUM(freight) AS TotalFreightCost
   FROM Orders
   GROUP by ShipVia;
   ```

2. You should have seen three shippers and the total freight charges. This one used the group by command, which basically organizes the data by "ShipVia" then calculates the sum. Make sure that you understand this one before you continue.



3. Now include the minimum, average, and maximum as separate columns. Note that the commands can be on multiple lines.

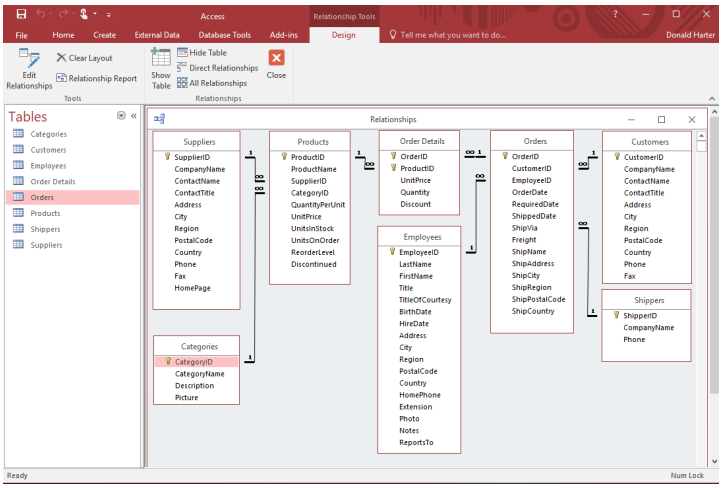
   ```
   SELECT ShipVia, SUM(freight) AS TotalFreightCost, MIN(freight) AS MinFreightCost,
          AVG(freight) AS AvgFreightCost, MAX(freight) AS MaxFreightCost
   FROM Orders
   GROUP BY ShipVia;
   ```



Retrieving from Multiple Tables: Table Joins

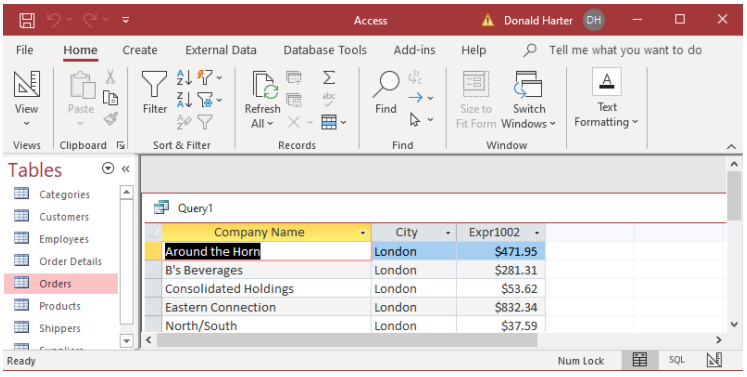
Now the tricky one. What if you want information that is contained in more than one table? To accomplish that you need to use a join.

1. When using more than one table, each field (data column) must be identified by the table name and column name. To do this, use Table.Field. For example, to select CompanyName from the Customers table, type Customers.CompanyName
2. To retrieve data from two tables, you need to use a join command. This is just a Where clause that includes the two fields that must match to bring the tables together. Refer to the Relationships chart below.



3. Create a query to find out how much each customer in London paid in freight charges. Note that CompanyName and City are in the Customers table, but freight is in the Orders table.
4. Since we are using two tables, both table names must be in the From line, separated by commas.
5. The WHERE clause must identify the City as London, but also link the two tables using the common key (Customers.customerID=Orders.customerID)
6. The GROUP BY simply says that we should calculate the sum of freight charges after we have grouped the companies together.

```
SELECT Customers.CompanyName, Customers.City, sum(Orders.freight)
FROM Customers, Orders
WHERE Customers.City="London"
AND Customers.customerID=Orders.customerID
GROUP BY Customers.City, Customers.CompanyName;
```

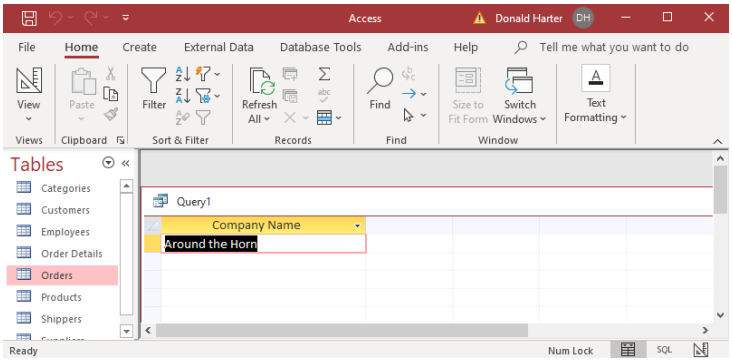


WHERE clauses with date functions

Let's retrieve names of companies with a specific order date (company name is in the Customers table, orders in the Orders table)

1. Retrieve all the names of companies with an order on July 5, 1995 using a join (this only works with the United States date format).
2. Since we are retrieving from two tables (Customers and Orders), we need to include them in the FROM clause
3. The WHERE clause must include the join for the two tables and the date value that we want

```
SELECT CompanyName
FROM Customers, ORDERS
WHERE Customers.CustomerID=Orders.CustomerID
AND OrderDate=#7/5/95#;
```

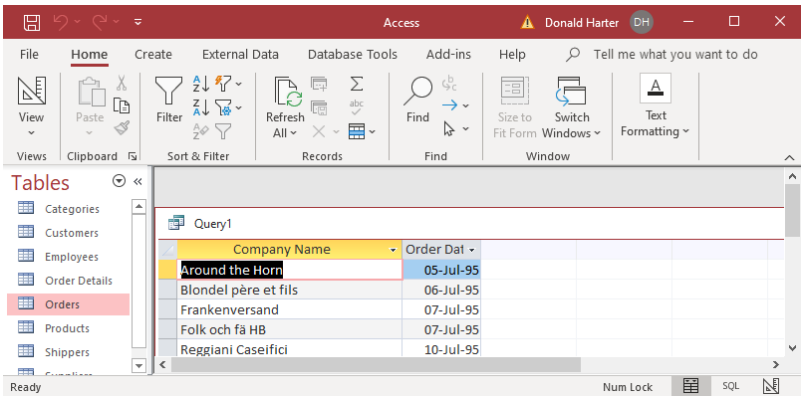


Data Ranges using BETWEEN

When you want to retrieve data that is in a range, such as dates, you can use the BETWEEN command.

1. Retrieve all the names of companies with an order between July 5 and July 12, 1995

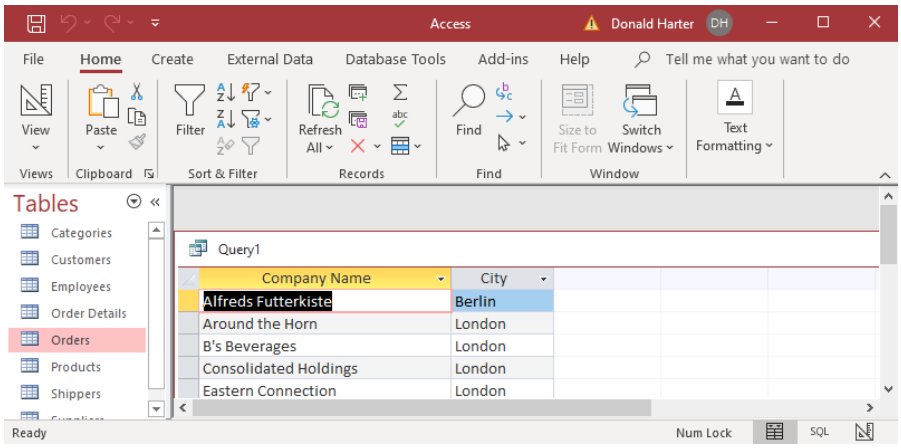
```
SELECT CompanyName, OrderDate
FROM Customers, Orders
WHERE Customers.CustomerID=Orders.CustomerID
AND OrderDate BETWEEN #7/5/95# AND #7/12/95#;
```



Additional Boolean Operators and uses of the IN list operator

1. Retrieve all the names of companies in London or Berlin using the OR operator

   ```
   SELECT CompanyName, City
   FROM Customers
   WHERE City="London" OR City="Berlin";
   ```

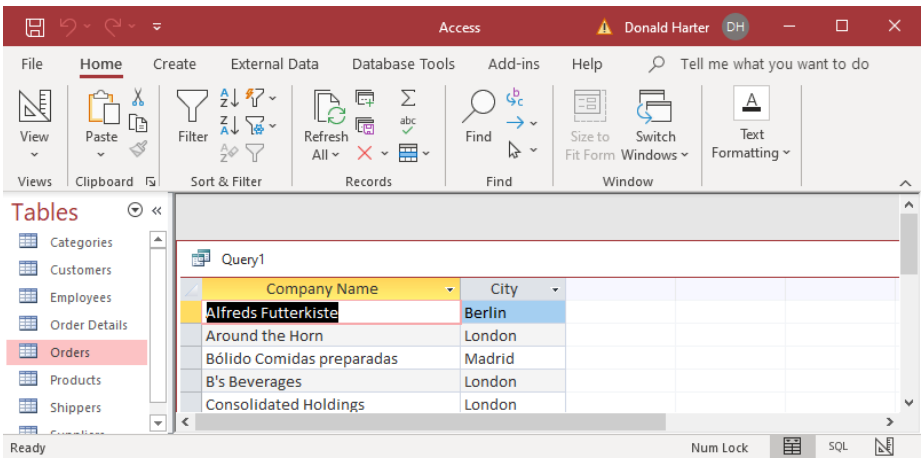


2. Now do the same query using the IN operator. The list of cities is enclosed in parentheses, separated by commas). You should see the same result as above.

   ```
   SELECT CompanyName, City
   FROM Customers
   WHERE City IN ("London","Berlin");
   ```

3. Notice it's much easier to add to a list when using the IN list option.

   ```
   SELECT CompanyName, City
   FROM Customers
   WHERE City IN ("London","Berlin","Madrid");
   ```

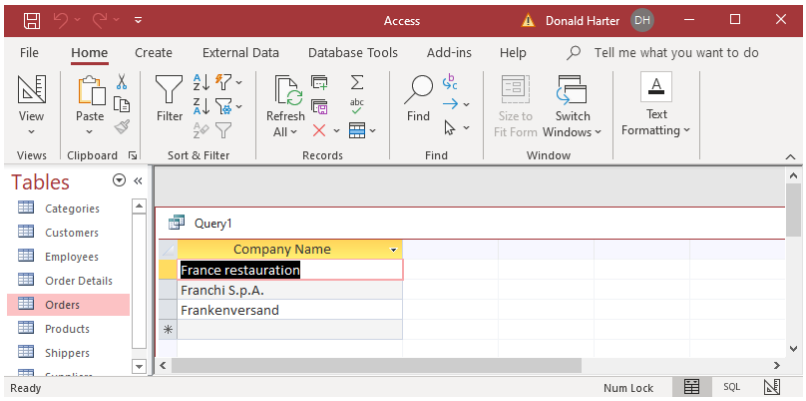


LIKE Command: Matching part of a string of characters

The LIKE command in a where clause matches a group of characters to the data in a field.

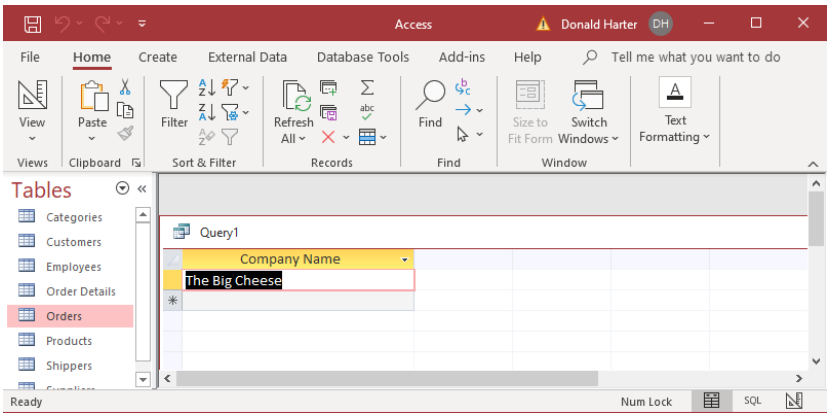
1. Retrieve all the names of companies that start with the letters “Fr”; the wildcard character * means that any character can follow Fr. (Note: other DBMS software use % as the wildcard character.)

   ```
   SELECT CompanyName
   FROM Customers
   WHERE CompanyName LIKE "Fr*";
   ```



2. Now retrieve the names of companies that have Big in the name. Since we don’t know if Big is at the beginning, middle or end, add a wildcard character * before and after Big

   ```
   SELECT CompanyName
   FROM Customers
   WHERE CompanyName LIKE "*Big*";
   ```

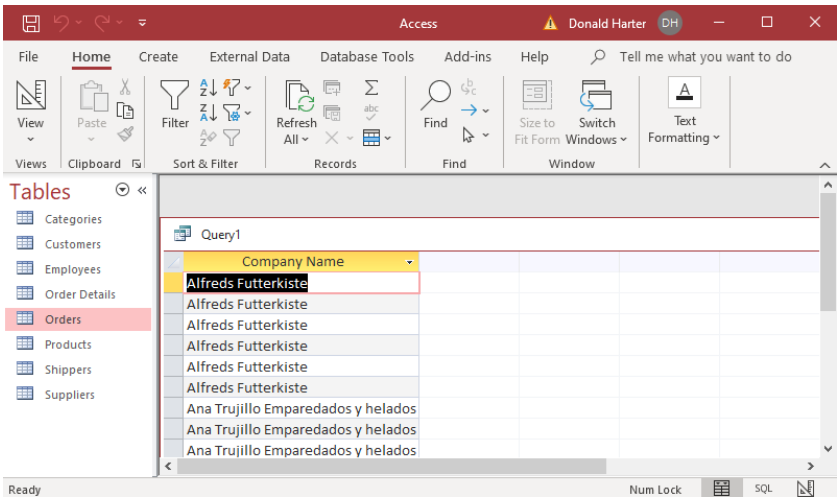


HAVING command and Table Aliases

Pretty easy, isn't it? Now try a tough one. Retrieve all company names with more than twenty (20) orders. Use the option of a table alias to shorten the command.

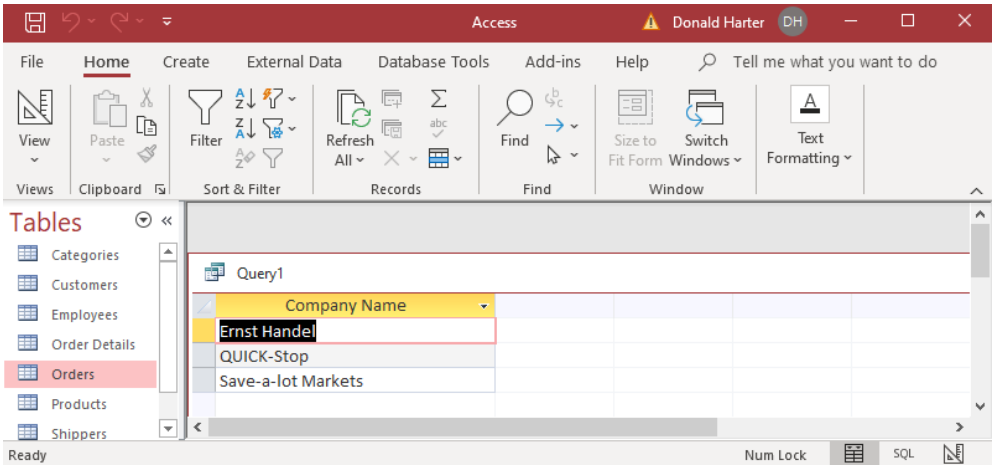
1. A table alias is another name for a table that allows you to shorten your commands; in the FROM clause, entering Customers C means that to refer to table Customers, you can now abbreviate it to C

   SELECT CompanyName
   FROM Customers C, Orders O
   WHERE C.CustomerID=O.CustomerID;



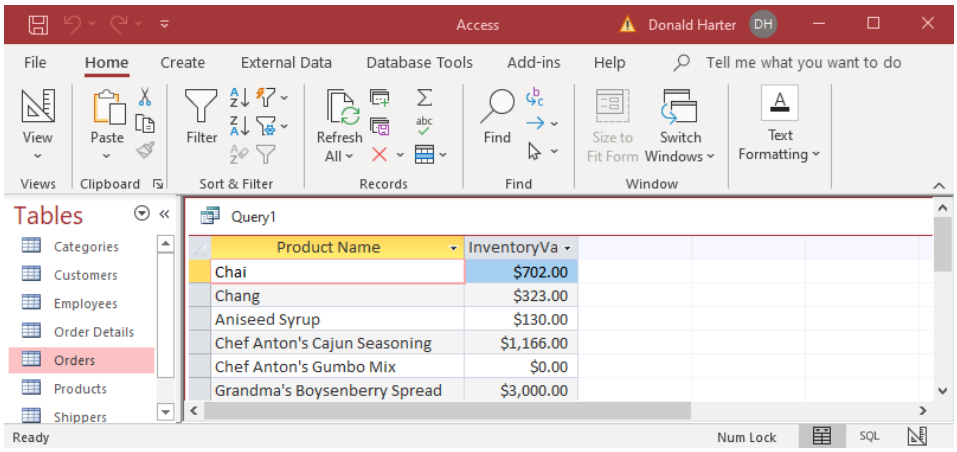
2. The HAVING clause combines a WHERE clause and a calculation. Use the HAVING clause to restrict the data displayed to only companies with more than 20 orders. We use the same SQL as the above example, adding a GROUP BY and HAVING clause.

   SELECT CompanyName
   FROM Customers C, Orders O
   WHERE C.CustomerID=O.CustomerID
   GROUP BY C.CompanyName
   HAVING count(*)>20;



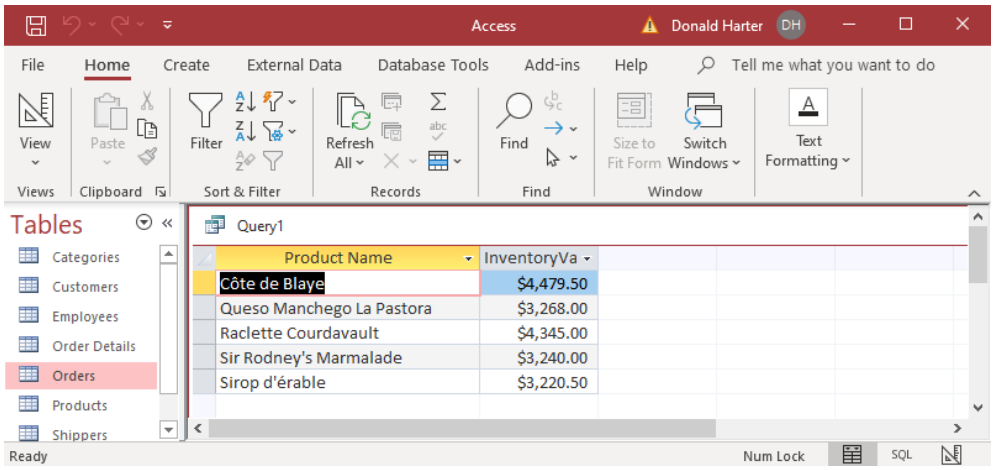
3. The HAVING clause can also allow you to find a product inventory value over a given threshold. In this example, find ProductName with total inventory value over 3000.
4. Inventory value is not in the database, but UnitPrice and UnitsInStock is in the Products table; we can multiply these together
5. First, list Product Name (Products table) and their InventoryValue (UnitPrice*UnitsInStock)

   SELECT ProductName, UnitPrice*UnitsInStock AS InventoryValue
   FROM Products;



6. In order to use HAVING, we first need to GROUP BY. The GROUP BY will list a line for each GROUP BY field. The InventoryValue will need to be represented as a sum.

   SELECT ProductName, SUM(UnitPrice*UnitsInStock) AS InventoryValue
   FROM Products
   GROUP BY ProductName
   HAVING SUM(UnitPrice*UnitsInStock)>3000;



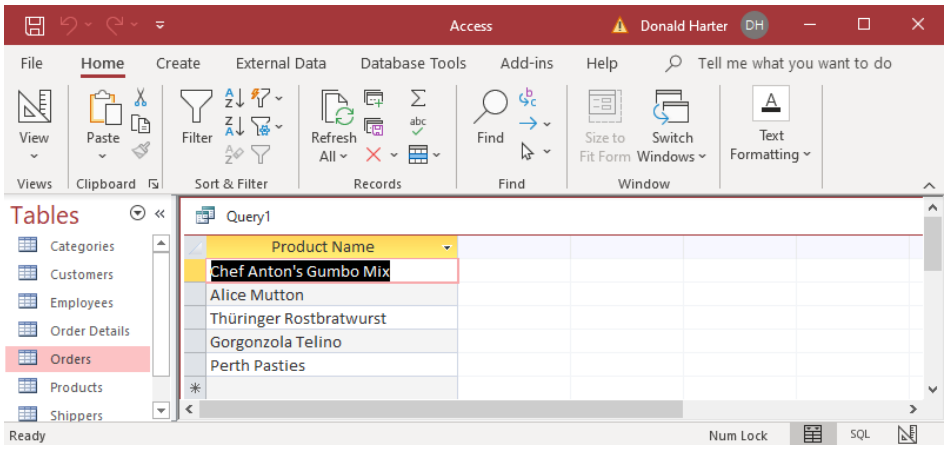
Subqueries with lists and the IN command

A subquery is a query embedded within a query. The IN command looks for items in a list. The SELECT within the parentheses is executed first, then work outwards toward the outer SELECT. The IN option matches items in a list

Now, retrieve companies that supply products which are out of stock, i.e., UnitsInStock=0.

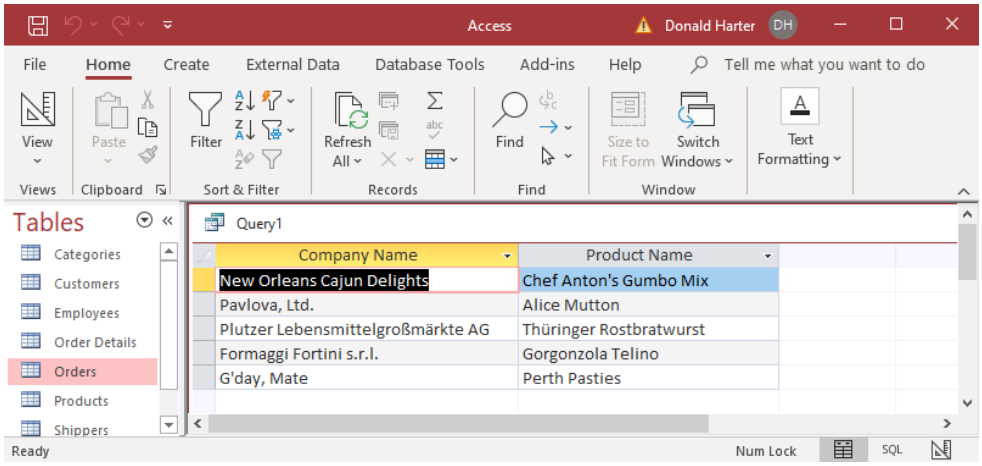
1. First create a query to find all Products within the Beverages Category.

```
SELECT ProductName
FROM Products
WHERE UnitsInStock=0;
```



2. Then find all Company Names from the Suppliers table that have a Product in this list.

```
SELECT CompanyName, ProductName
FROM  Suppliers, Products
WHERE Suppliers.SupplierID=Products.SupplierID AND ProductName IN
(SELECT ProductName
FROM Products
WHERE UnitsInStock=0);
```

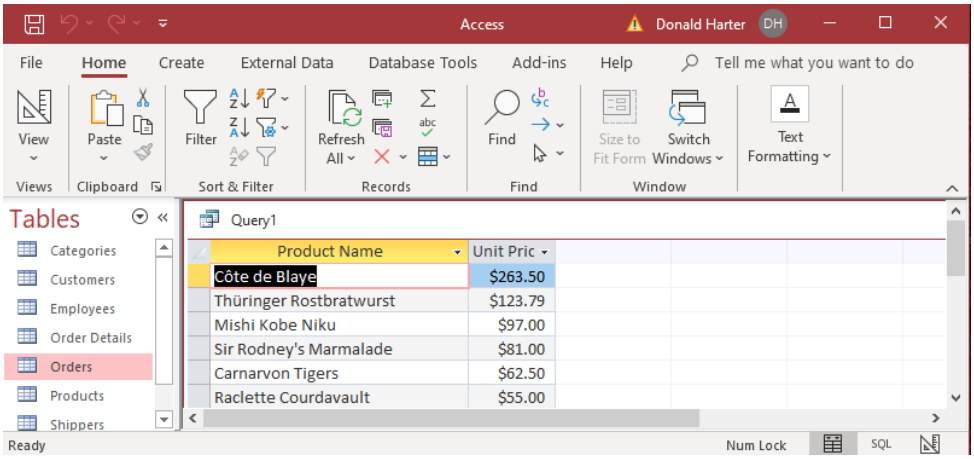


Limiting Results and List Top/Bottom

You sometimes will want to list the top items of a list. For example, you might want to know the most expensive products, without listing all products.

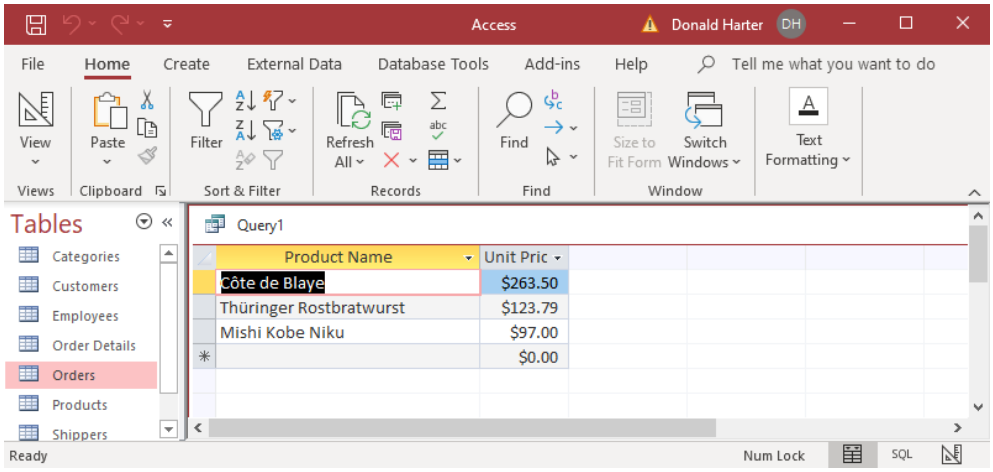
1. First, list all products by ProductName and UnitPrice, sorting on UnitPrice from high to low (DESC).

   ```
   SELECT ProductName, UnitPrice
   FROM Products
   ORDER BY UnitPrice DESC;
   ```



2. To select only the top 3, use the TOP option

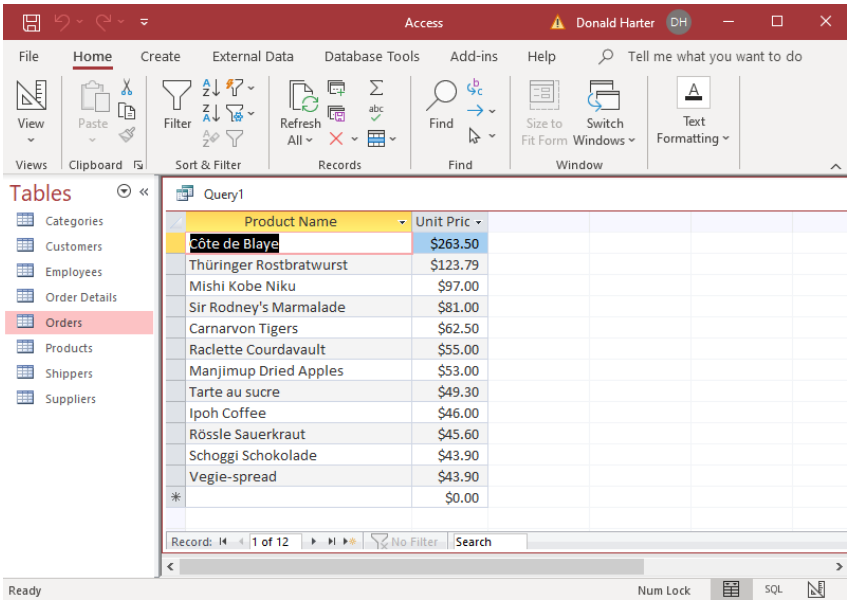
   ```
   SELECT TOP 3 ProductName, UnitPrice
   FROM Products
   ORDER BY UnitPrice DESC;
   ```



3. How would you select the bottom 3? Order by Ascending, then TOP 3.

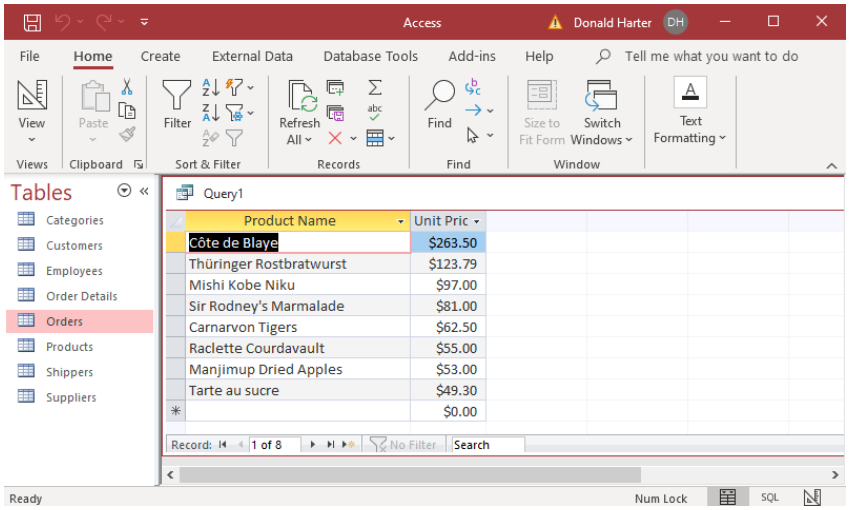
4. Change TOP 3 to TOP 11 and rerun

```
SELECT TOP 11 ProductName, UnitPrice
FROM Products
ORDER BY UnitPrice DESC;
```



5. Note that it returned 12 records, not 11. When there is a tie in Access, it returns the tied values, even if it exceeds the TOP limit.
6. You can also retrieve by a Percent. In this example, retrieve the TOP 10 PERCENT.

```
SELECT TOP 10 PERCENT ProductName, UnitPrice
FROM Products
ORDER BY UnitPrice DESC;
```

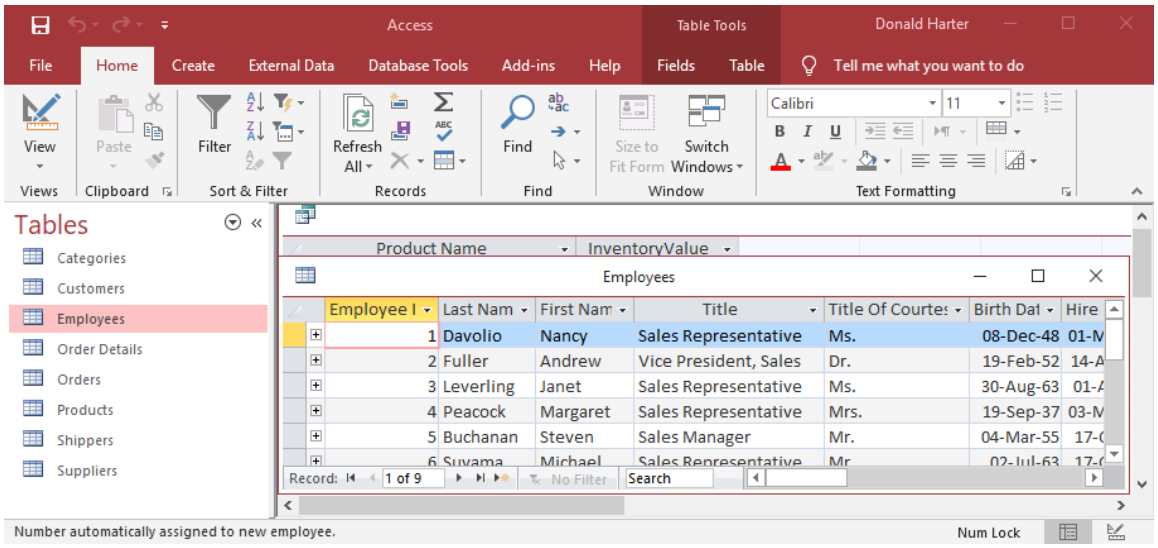


7. There are 77 records, so 10 Percent of 77 is approximately 8 records.

Concatenation

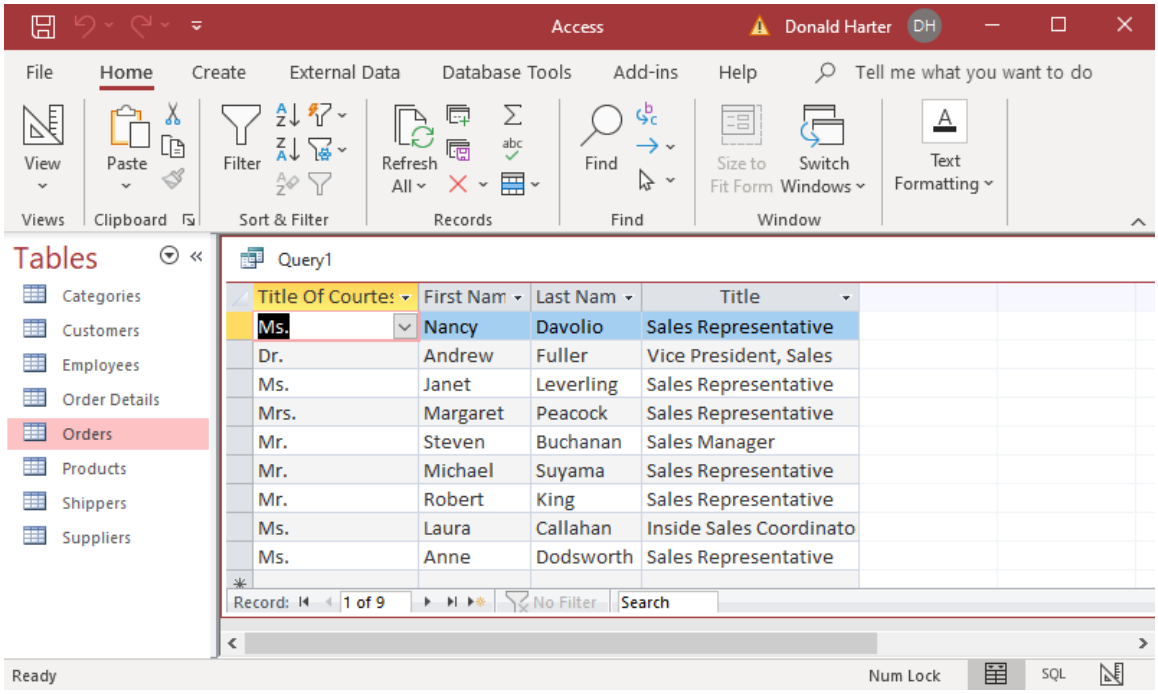
Occasionally, you will want to combine fields. This is called concatenation.

1. The Employees table has an employee's last name, first name, title, and title of courtesy:



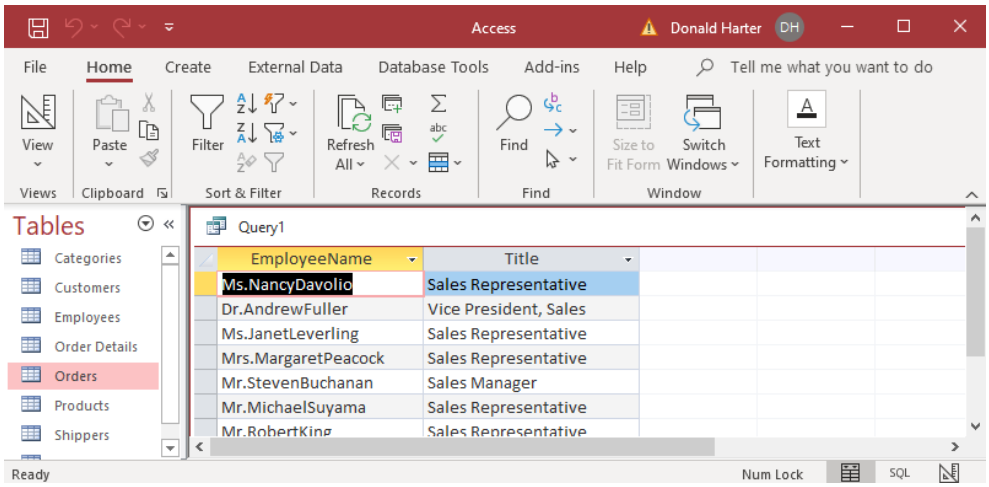
2. To create a query that listed title of courtesy, then first name, then last name, the title, we would write:

```
SELECT TitleofCourtesy, FirstName, LastName, Title
FROM Employees;
```



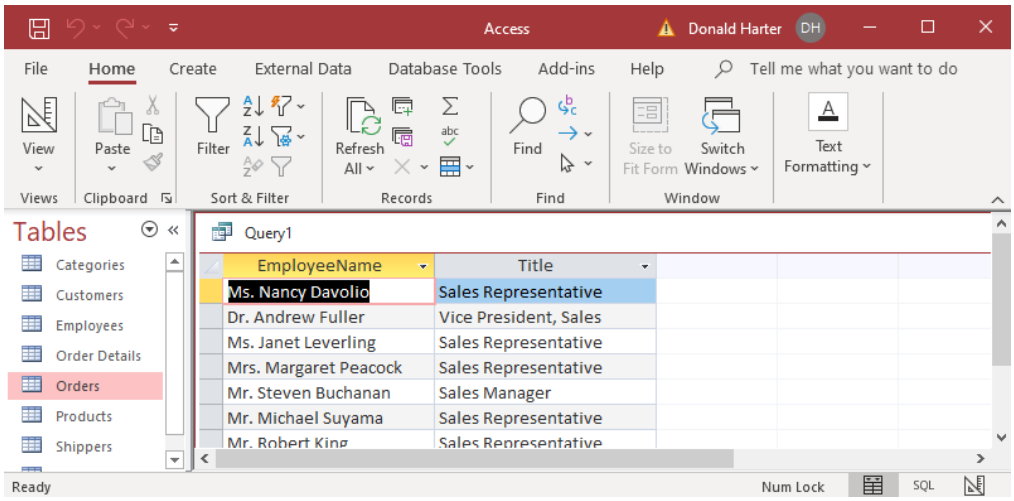
3. Next, concatenate (merge) the Title of Courtesy, First Name, and Last Name, labelling as EmployeeName, by inserting + between the fields.

   SELECT TitleofCourtesy + FirstName + LastName AS EmployeeName, Title
   FROM Employees;



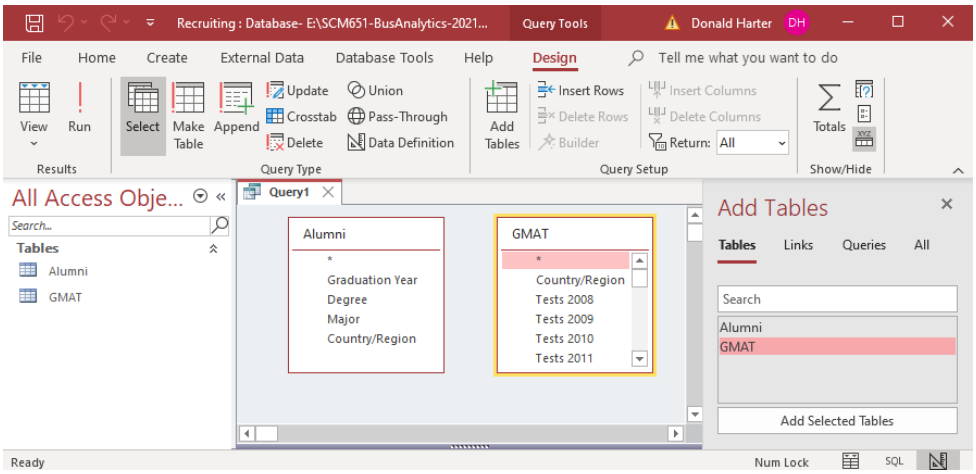
4. Note that we lose some spacing. That can be fixed by inserting spaces.

   SELECT TitleofCourtesy + ' ' + FirstName + ' ' + LastName AS EmployeeName, Title
   FROM Employees;

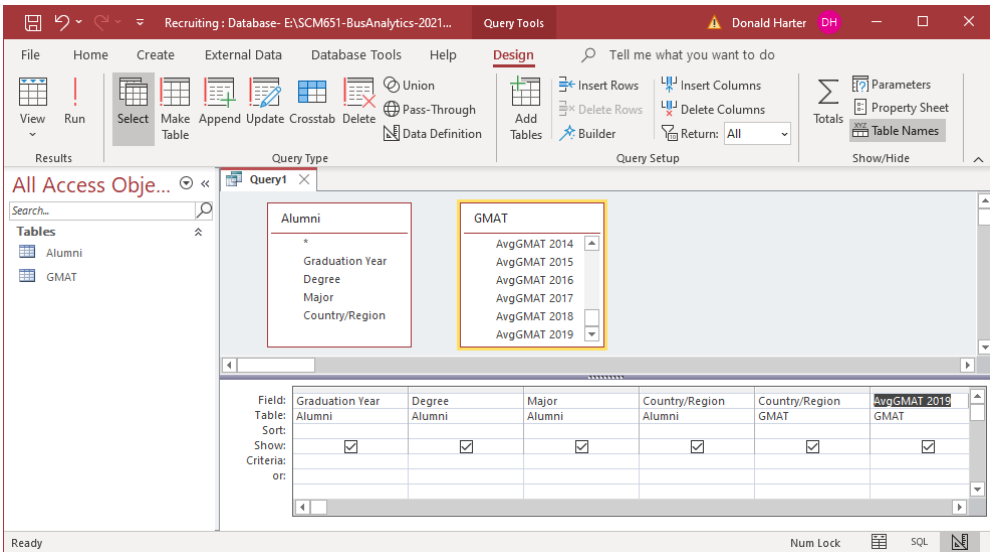


Joins: Inner, Left, Right, and Outer

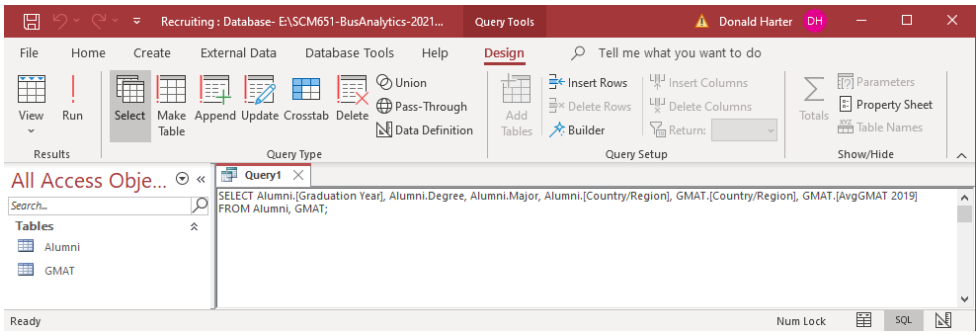
1. For this exercise, use the Recruiting data base.
2. Click Enable, then Yes for trust.
3. Create a query by clicking on Create, Query Design
4. In Add Tables, double click on Alumni
5. In Add Tables, double click on GMAT
6. Click on the X in Add Tables to close the right window



7. Notice that there is no relationship between Alumni and GMAT. We will create the join in SQL
8. Double click on Graduation Year, Degree, Major, and Country/Region in the Alumni table
9. Double click on Country/Region and AvgGMAT 2019 in the GMAT table



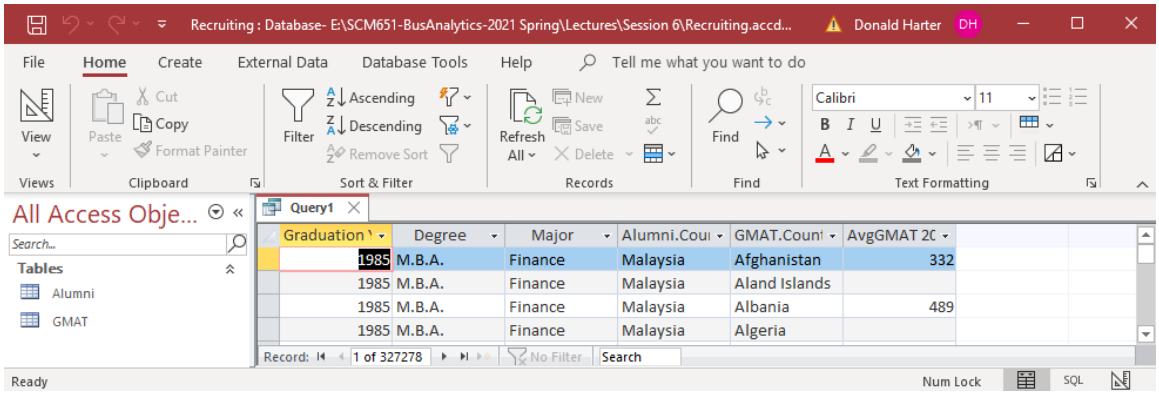
10. Click on View, SQL



11. You should see the SQL:

    SELECT Alumni.[Graduation Year], Alumni.Degree, Alumni.Major,
    Alumni.[Country/Region], GMAT.[Country/Region], GMAT.[AvgGMAT 2019]
    FROM Alumni, GMAT;

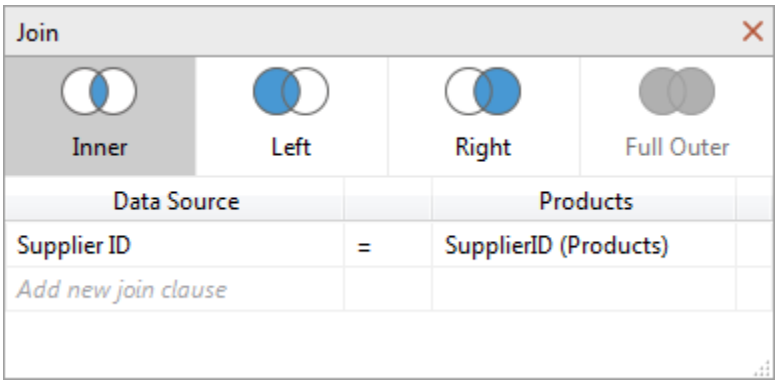
12. Note that Graduation Year has a space between Graduation and Year. To accommodate this space, Access add square brackets [ ] to the field name. Similarly Country/Region has a / between the words; we need to add brackets.
13. Also note that when Country/Region is used twice, Access adds the table name in front of Country, e.g., Alumni.Country/Region and GMAT.Country/Region
14. Why are there square brackets around AvgGMAT 2019?
15. Click Run !
16. How many records are there?
17. This is the Cartesian product: 1358 records in Alumni x 241 records in GMAT = 327,278 combinations.



Join Types

There are four types of database joins:

- Inner Join – only include rows where the joined fields from both tables are equal
- Left Join – Include all records from the left table and only those records from the right table where the joined fields are equal
- Right Join – Include all records from the right table and only those records from the left table where the joined fields are equal
- Outer Join – Include all records from the left and right table where the key field appears in either table

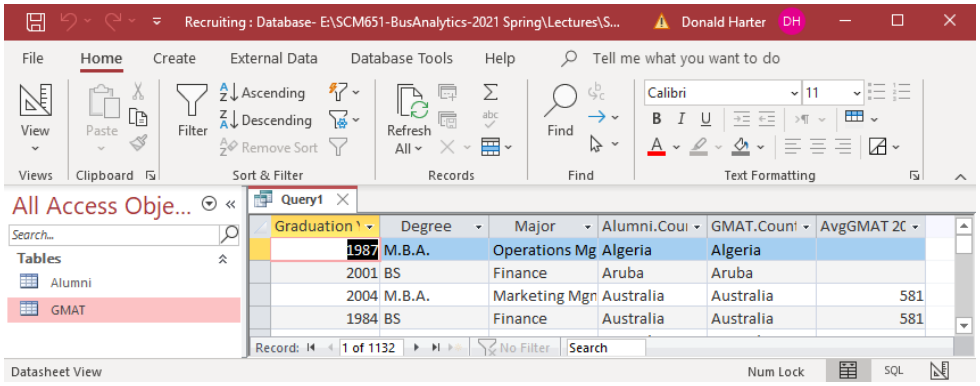


INNER JOIN

1. Click View, Design View to return to the SQL
2. Create an INNER JOIN with the following changes:

   SELECT Alumni.[Graduation Year], Alumni.Degree, Alumni.Major,
   Alumni.[Country/Region], GMAT.[Country/Region], GMAT.[AvgGMAT 2019]
   FROM Alumni INNER JOIN GMAT
   ON Alumni.[Country/Region] = GMAT.[Country/Region];

3. Click Run !
4. How many records are there?

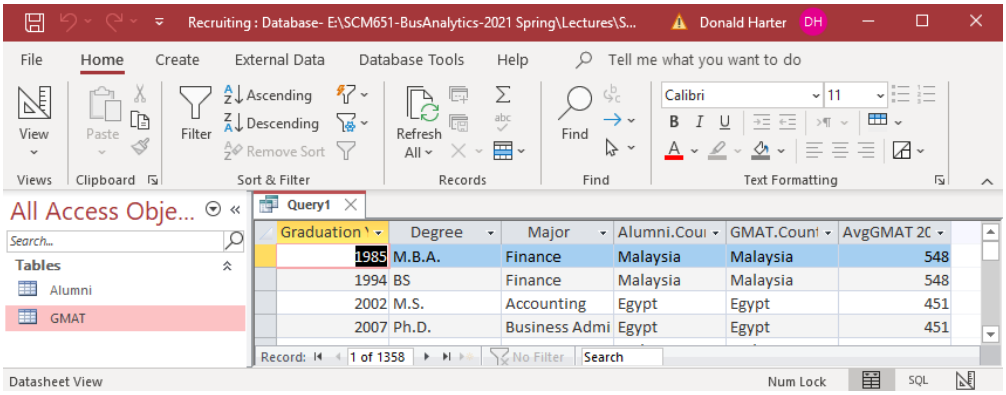


LEFT JOIN

1. Click View, Design View to return to the SQL
2. Create a LEFT JOIN with the following changes:

   SELECT Alumni.[Graduation Year], Alumni.Degree, Alumni.Major,
   Alumni.[Country/Region], GMAT.[Country/Region], GMAT.[AvgGMAT 2019]
   FROM Alumni LEFT JOIN GMAT
   ON Alumni.[Country/Region] = GMAT.[Country/Region];

3. Click Run !
4. How many records are there?

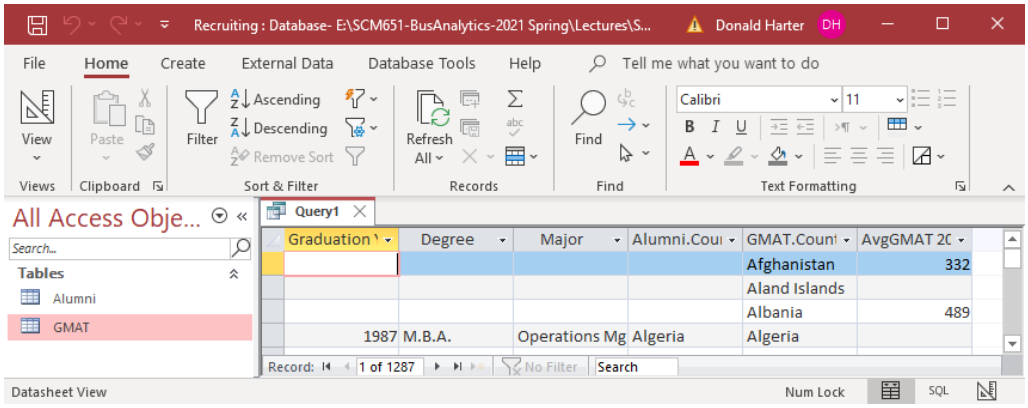


RIGHT JOIN

1. Click View, Design View to return to the SQL
2. Create a LEFT JOIN with the following changes:

   SELECT Alumni.[Graduation Year], Alumni.Degree, Alumni.Major,
   Alumni.[Country/Region], GMAT.[Country/Region], GMAT.[AvgGMAT 2019]
   FROM Alumni RIGHT JOIN GMAT
   ON Alumni.[Country/Region] = GMAT.[Country/Region];

3. Click Run !
4. How many records are there?



OUTER JOIN

1. An OUTER JOIN can be created by performing a UNION on a LEFT JOIN and a RIGHT JOIN
2. Click View, Design View to return to the SQL
3. Create an OUTER JOIN with the following changes:

```
SELECT Alumni.[Graduation Year], Alumni.Degree, Alumni.Major,
Alumni.[Country/Region], GMAT.[Country/Region], GMAT.[AvgGMAT 2019]
FROM Alumni LEFT JOIN GMAT
ON Alumni.[Country/Region] = GMAT.[Country/Region]

UNION

SELECT Alumni.[Graduation Year], Alumni.Degree, Alumni.Major,
Alumni.[Country/Region], GMAT.[Country/Region], GMAT.[AvgGMAT 2019]
FROM Alumni RIGHT JOIN GMAT
ON Alumni.[Country/Region] = GMAT.[Country/Region];
```

4. Click Run !
5. How many records are there?