CSIS 100 – Week 3 Lab 1: Database

**Background:**

A database is an organized collection of data that consists of tables, queries, views, reports, and other objects.

In this assignment, you are a professional working at a helpdesk for Foods, Inc., a grocery store distributor located in Manhattan. It is your responsibility to answer questions from customers, suppliers, and internal managers of your company by querying your Foods, Inc.’s database.

**Database Structure:**

Foods, Inc.’s database consists of eight tables that are already created and populated with data:

Customers

Categories

Employees

OrderDetails

Orders

Products

Shippers

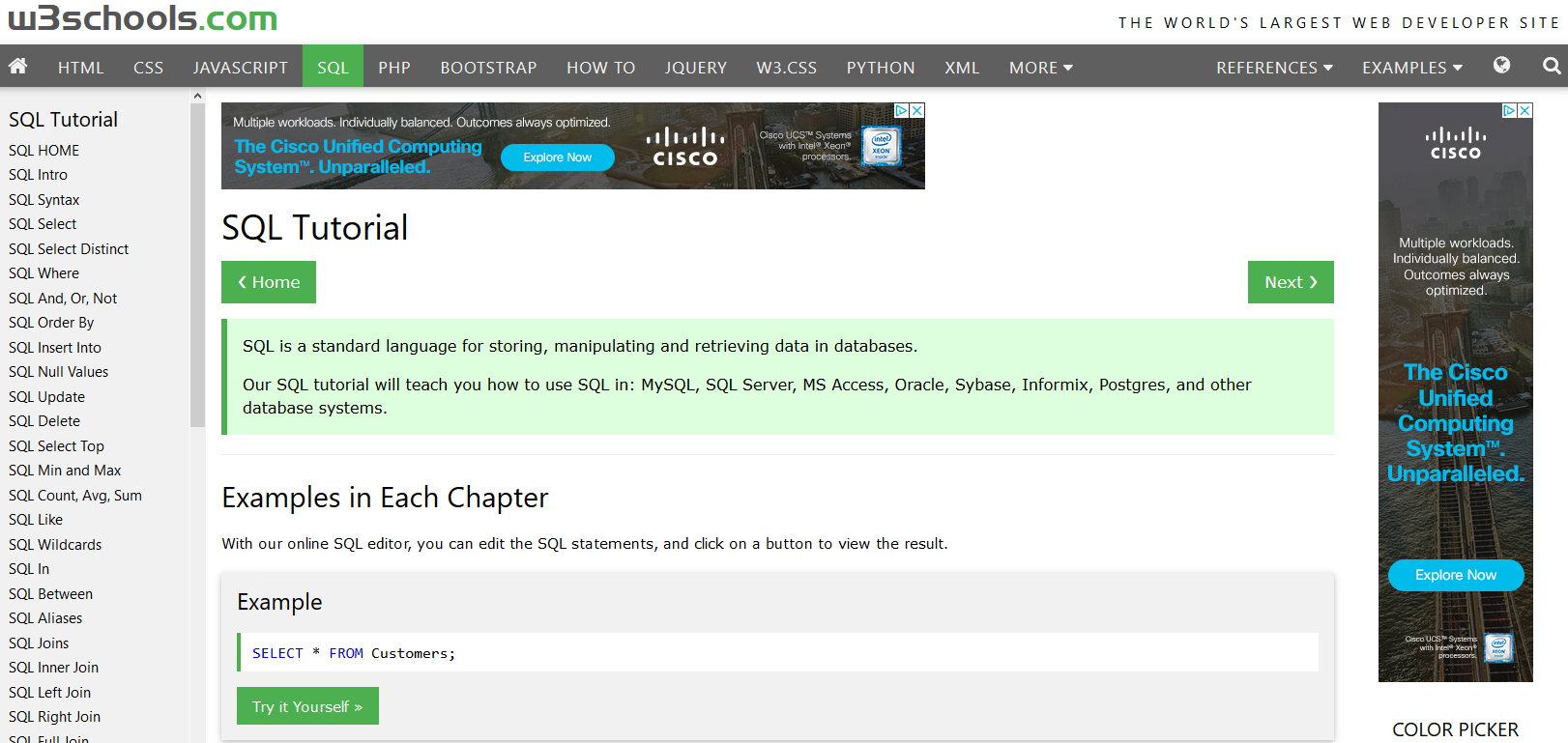
Suppliers

**Querying Basics:**

To access your database, click on the following link: <http://www.w3schools.com/sql/>

W3schools.com is a site for web developers that provides tutorials on everything from creating web pages to database programming in SQL.

To query (i.e. ask a question) from a database, you must type in a command in a SQL editor. W3schools.com provides its own built in SQL editor that you can access by clicking on the “Try it yourself” button as shown below.



The basic format of a query is as follows:

**SELECT** [fieldname]

**FROM** [tablename]

[**WHERE** condition] 🡨 The Where clause is optional.

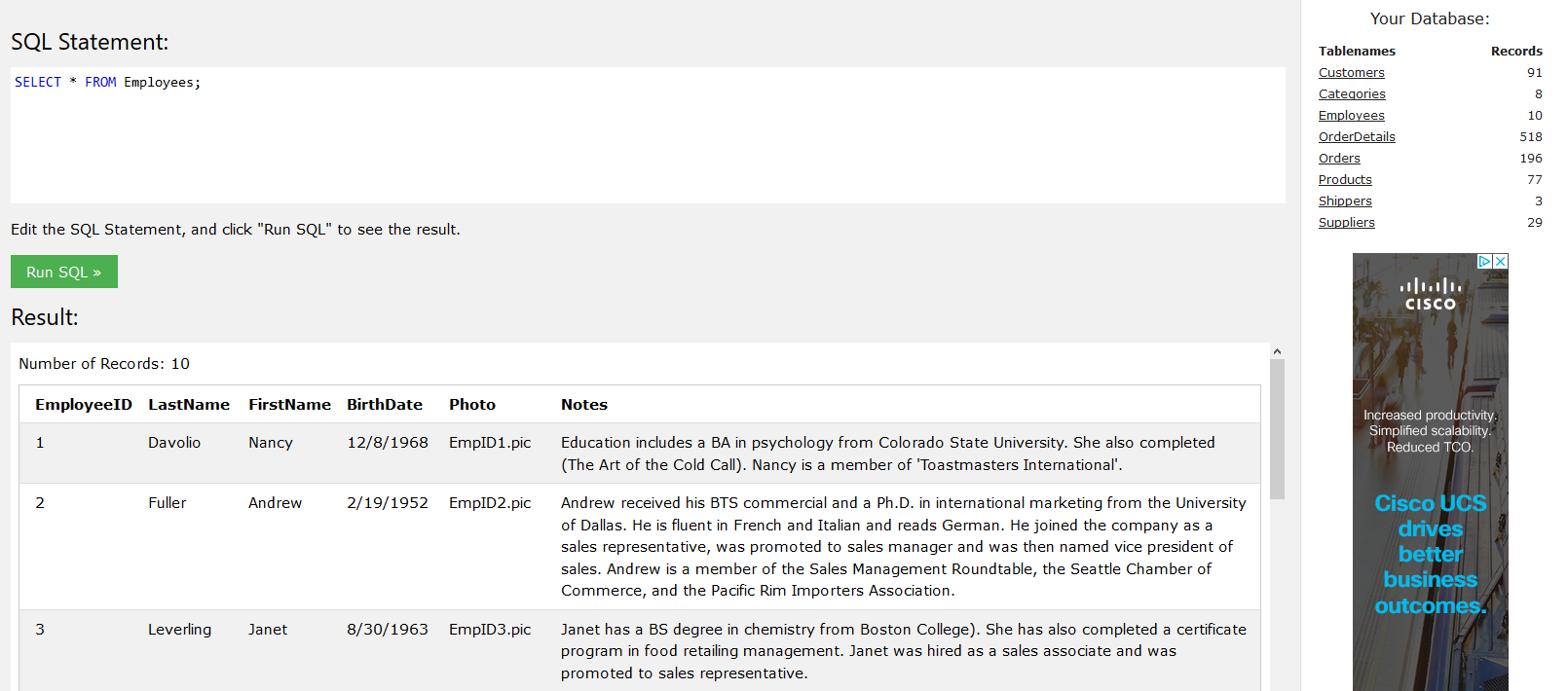
For example, to view every customer’s name in the Customers table, you would enter the command as shown below:

**SELECT** CustomerName

**FROM** Customers;

Clicking the Run SQL>> button causes the SQL editor to execute your query and display the results in the lower half of the screen.

**BIG TIP:** Before beginning this exercise, I encourage you to click on each of the table names in the upper right-hand corner of your screen to familiarize yourself with the structure of each table. Knowing what is in each table is very helpful in building queries. Database administrators have this information clearly displayed in their development area.



**Third, shows results here.**

**Second, places query here**

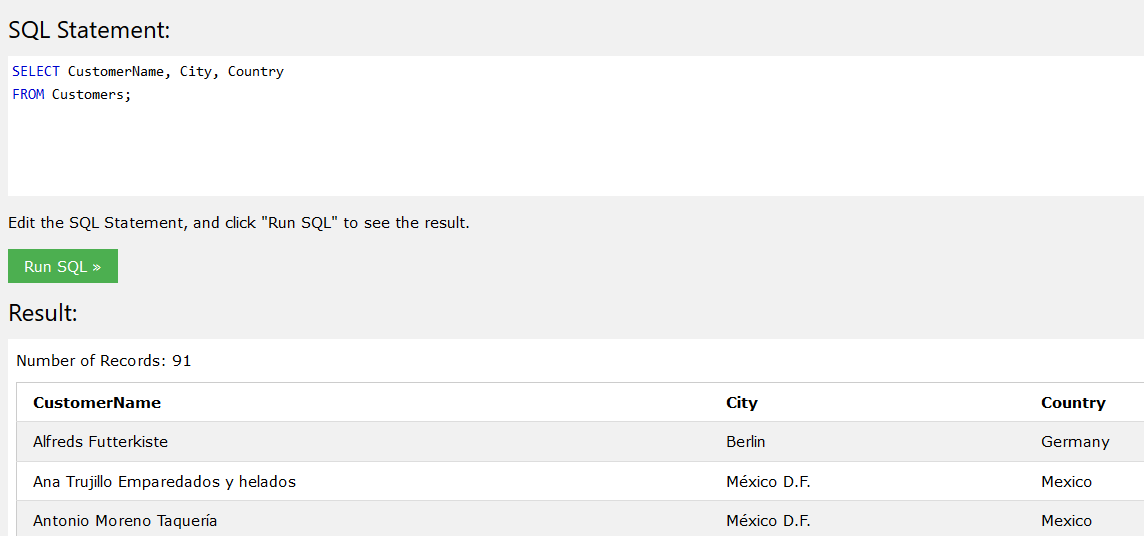
**First, clicking here**

**SELECT clause:**

To view more than one column (field) of data in a table, you can specify the fieldnames in the Select clause, separated by commas:

**SELECT** CustomerName, City, Country

**FROM** Customers;



To view **all** of the columns in a table, you can use the asterisk (\*) as a wildcard character that represents all columns:

**SELECT** \*

**FROM** Customers;

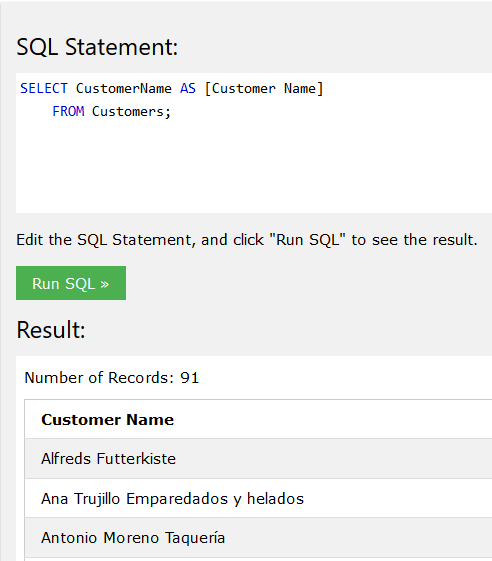
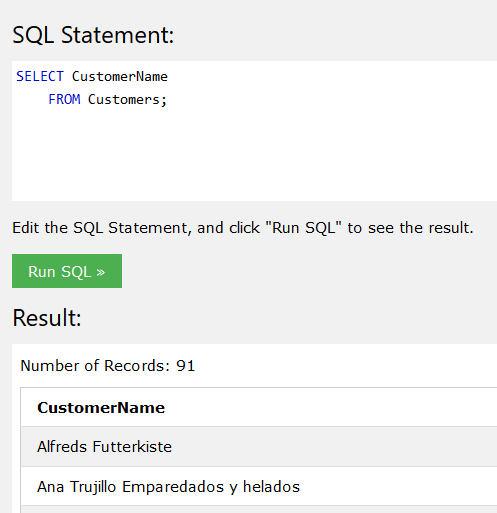
If you were to read the above query out loud you would read, “Select all from customers.”

**Aliases in the SELECT clause:**

Did you notice that CustomerName looks really “techie”? If you were to create a report that displayed CustomerName as the title of a column, the report would not look very professional. Instead, you can make your query display this field as a different name (i.e. an “alias”) using the keyword “AS”:

**SELECT** CustomerName **AS** [Customer Name] 🡨Note that if you have a space in your alias, you

**FROM** Customers; must enclose the alias in square brackets.



**With Alias**

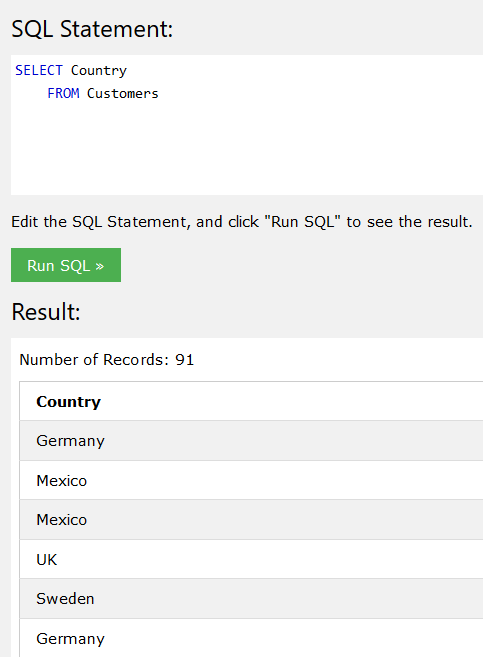
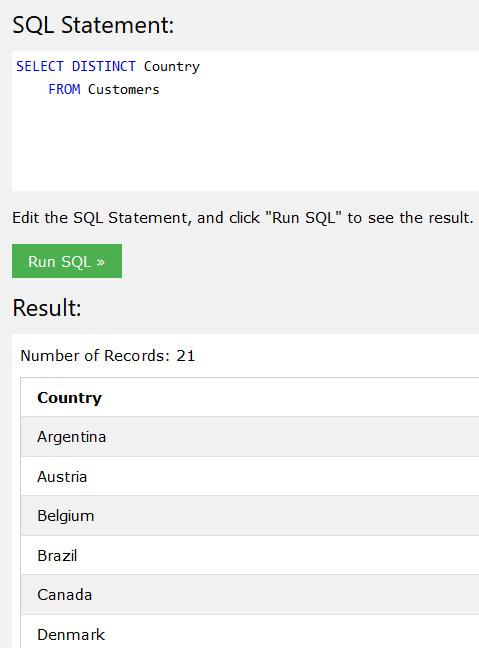
**Without Alias**

**DISTINCT keyword in the Select Clause:**

Sometimes a table includes multiple occurrences of the same value in a field (not a primary key field, of course), and you only want to see each occurrence listed one time in your output. Assume we want to see a unique listing of all of the countries in the Customers table. We can use the DISTINCT keyword to ensure that every country is only listed once:

**SELECT** **DISTINCT** Country

**FROM** Customers



**With Distinct**

**Without Distinct**

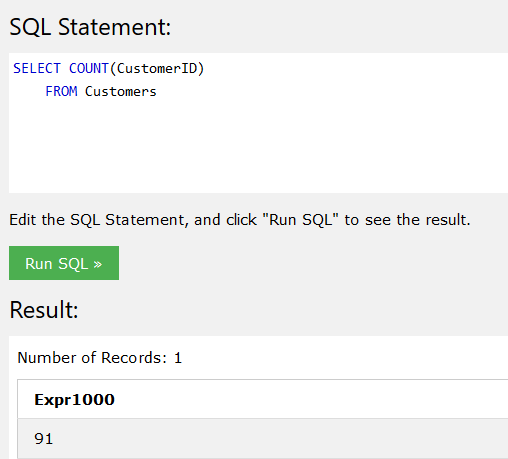
**Aggregate operators in the Select Clause:**

In many queries, you are not interested in the actual values of each individual row in a table, but rather a summary of them. For example, if someone asked you how many customers had ever ordered from you, you could use the COUNT operator in your select clause. For example:

**SELECT** **COUNT**(CustomerID)

**FROM** Customers

This query would return the count (or the total) number of rows in the Customers table.



Note that whenever you specify any other field in the Select clause other than the item that you are aggregating, you must also include a GROUP BY clause at the bottom of your query. For example, assume you want to know how many customers are in each country in your table, you can count the number of customers and group them by Country:

**SELECT** Country, Count(CustomerID) as [Customers by Country]

**FROM** Customers

**GROUP BY** Country



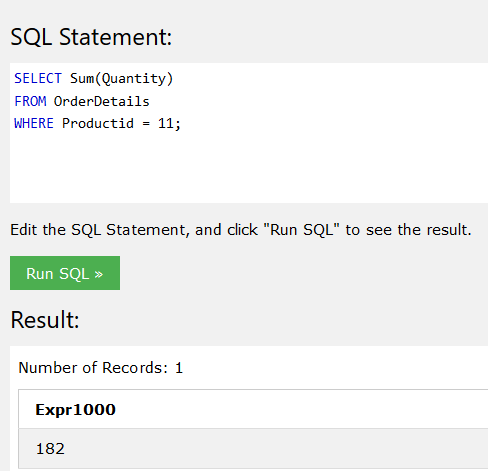
Count is not the only aggregate operator available to you. SUM, AVG, MIN, and MAX are a few examples of others that you can use. Note, however, that statistical functions (e.g. SUM, AVG, etc.) can only be used on fields that contain numbers and are defined with a numeric data type. For instance, it would not make sense to SUM the customer name field. However, it would make sense to SUM the Quantity of a particular product in the OrderDetails table:

**SELECT** Sum(Quantity)

**FROM** OrderDetails

**WHERE** Productid = 11;

This query produces the sum of the quantity field for the product whose ID is 11. Note that this query includes the use of a WHERE clause (discussed next).



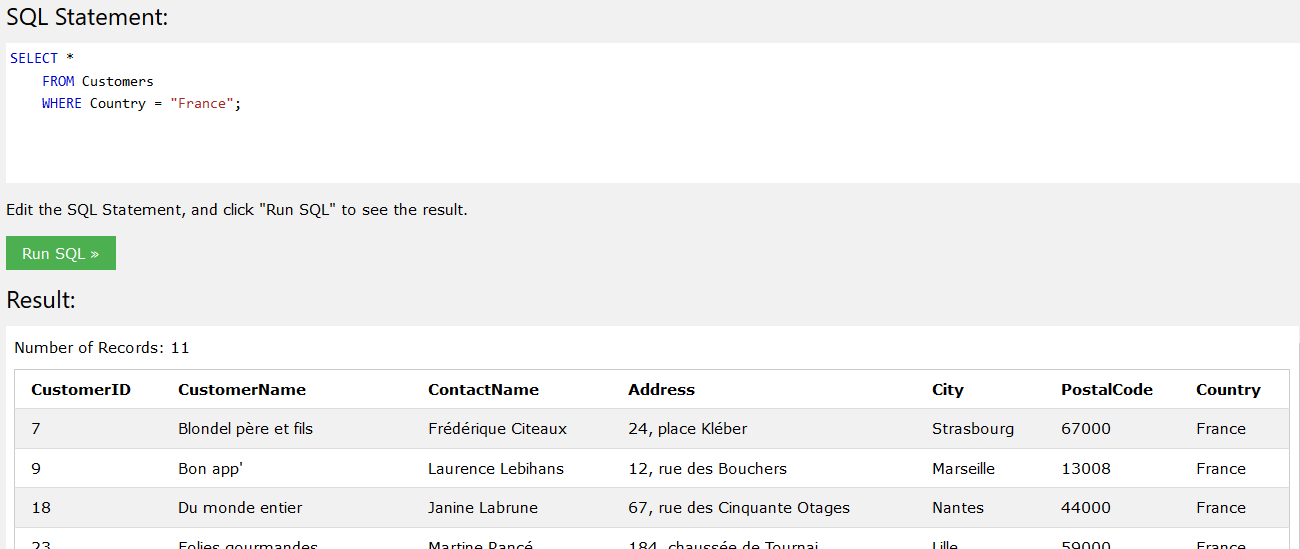
**WHERE Clause:**

What if you want to view only those customers that meet a certain criterion? This is where the “Where” clause comes in. Let’s assume you want to view customers that live in Berlin:

**SELECT** \*

**FROM** Customers

**WHERE** Country = "France";



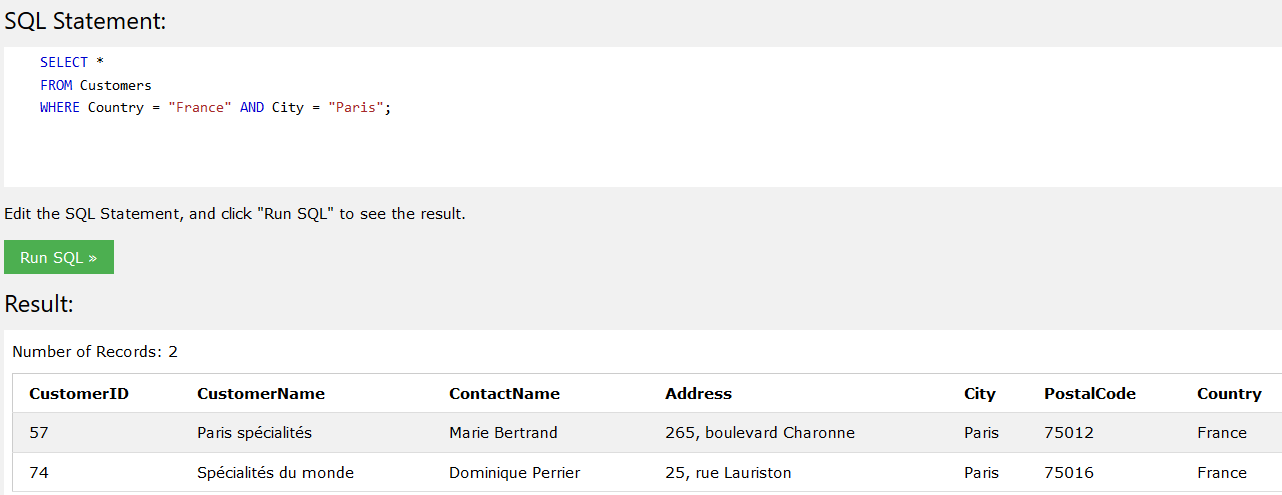
**Multiple conditions in a WHERE clause:**

What if you want to narrow down your results even more? Let’s assume you want to view customers that live in the country of France AND the city of Paris?

**SELECT** \*

**FROM** Customers

**WHERE** Country = "France" **AND** City = "Paris";



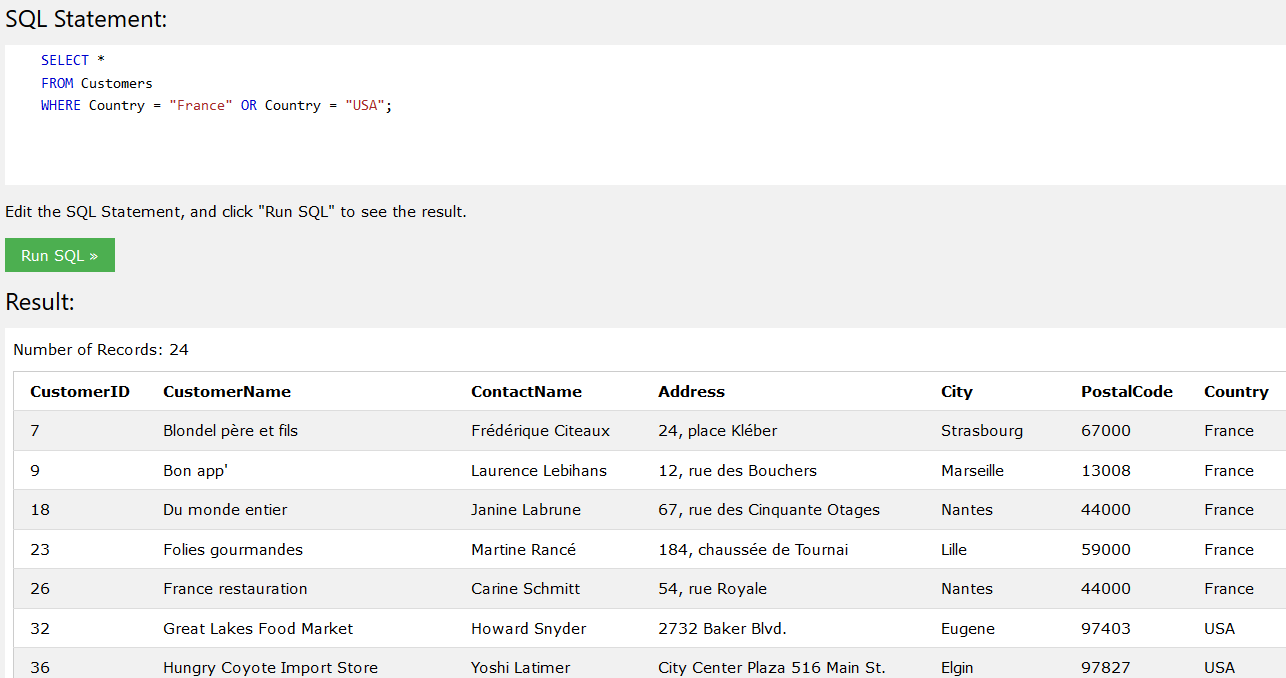
What if you want to expand your results to include everyone in France OR the USA?

**SELECT** \*

**FROM** Customers

**WHERE** Country = “France” **OR** Country = "USA";

Note that you can combine AND’s and OR’s as much as needed to satisfy your question.



**IN keyword:**

Assume you want to specify an elaborate OR condition like this:

**SELECT** \*

**FROM** Customers

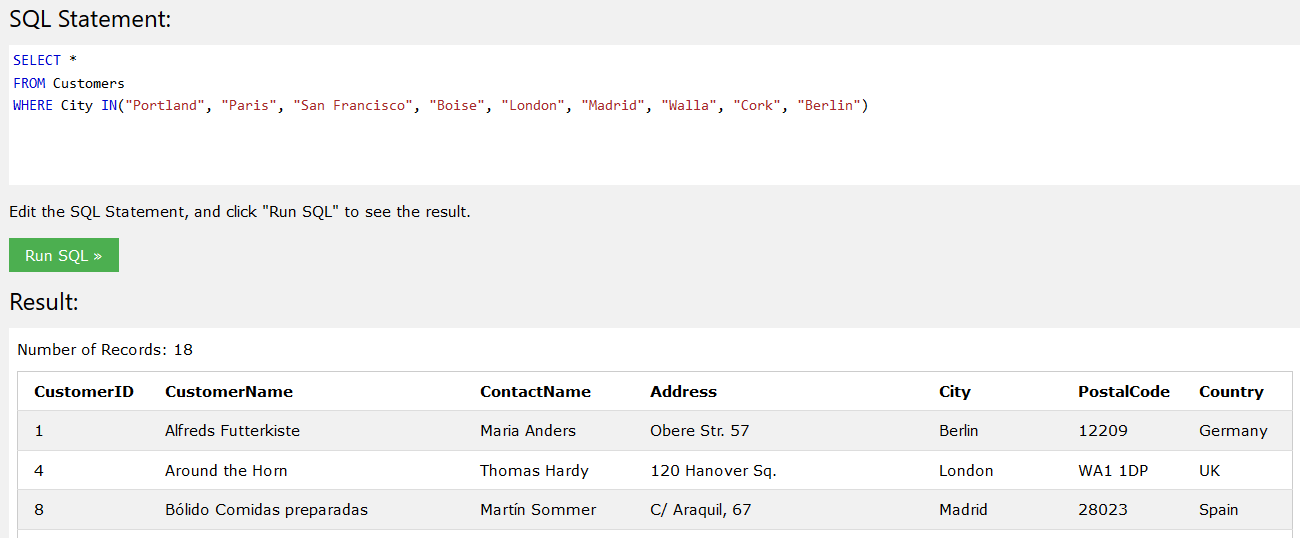
**WHERE** City = "Portland" **OR** City = "Paris" **OR** City = "San Francisco" **OR** City = "Boise" **OR** City = "London" **OR** City = "Madrid" **OR** city = "Walla" **OR** City = "Cork" **OR** City = "Berlin"

SQL gives you a shortcut that eliminates much of the typing. This shortcut uses the keyword “IN”. The list of items following the IN keyword is the list of items for which you are filtering:

**SELECT** \*

**FROM** Customers

**WHERE** City **IN**("Portland", "Paris", "San Francisco", "Boise", "London", "Madrid", "Walla", "Cork", "Berlin")



**Pattern matches:**

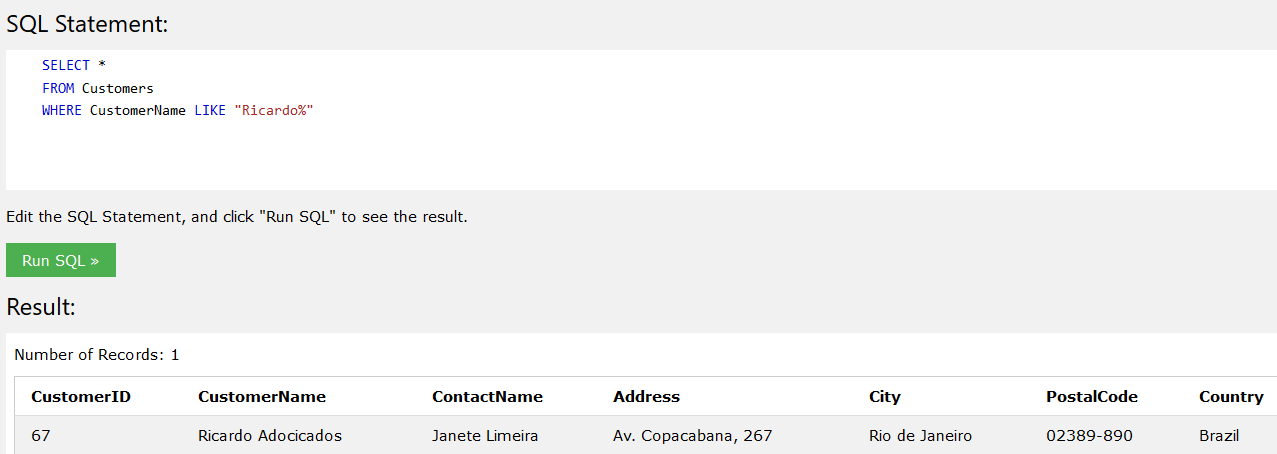
In our previous queries, we have been searching for records that meet a specific criterion, such as City = “Portland” or Country = “France”, etc. What if, however, we do not know exactly how a particular item is spelled? Instead of searching for an “exact” match, we can look for a “pattern”. For example, I want to find a customer whose name is “Ricardo” but I don’t his last name. I can use the following query to extract all of the records whose customer name starts with “Ricardo”:

**SELECT** \*

**FROM** Customers

**WHERE** CustomerName **LIKE** "Ricardo%" 🡨Note the use of the percent (%) sign. This symbol

stands for any character or sequence of characters.



**ORDER BY clause**

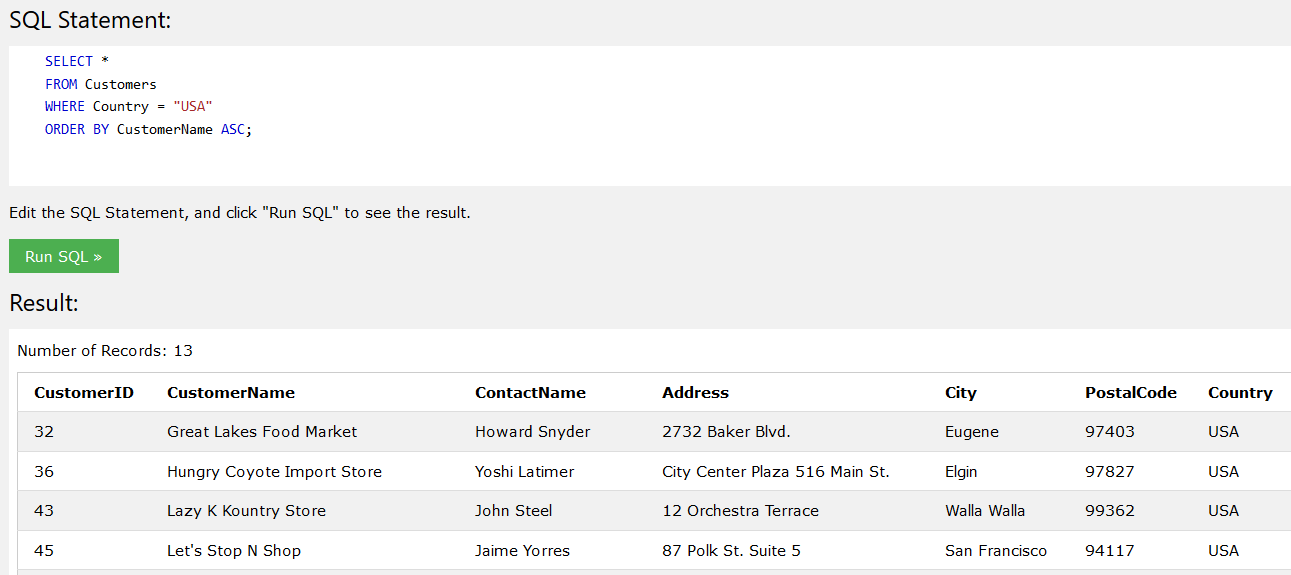
Assume you want to sort your results in a particular order. You can use the ORDER BY clause to do this. You can specify whether you want the results to be in ascending order (A-Z) with the “ASC” argument or in descending order (Z-A) with the “DESC” argument. If you leave out the argument, SQL assumes you mean ascending order.

**SELECT** \*

**FROM** Customers

**WHERE** Country = "USA"

**ORDER BY** CustomerName **ASC**;



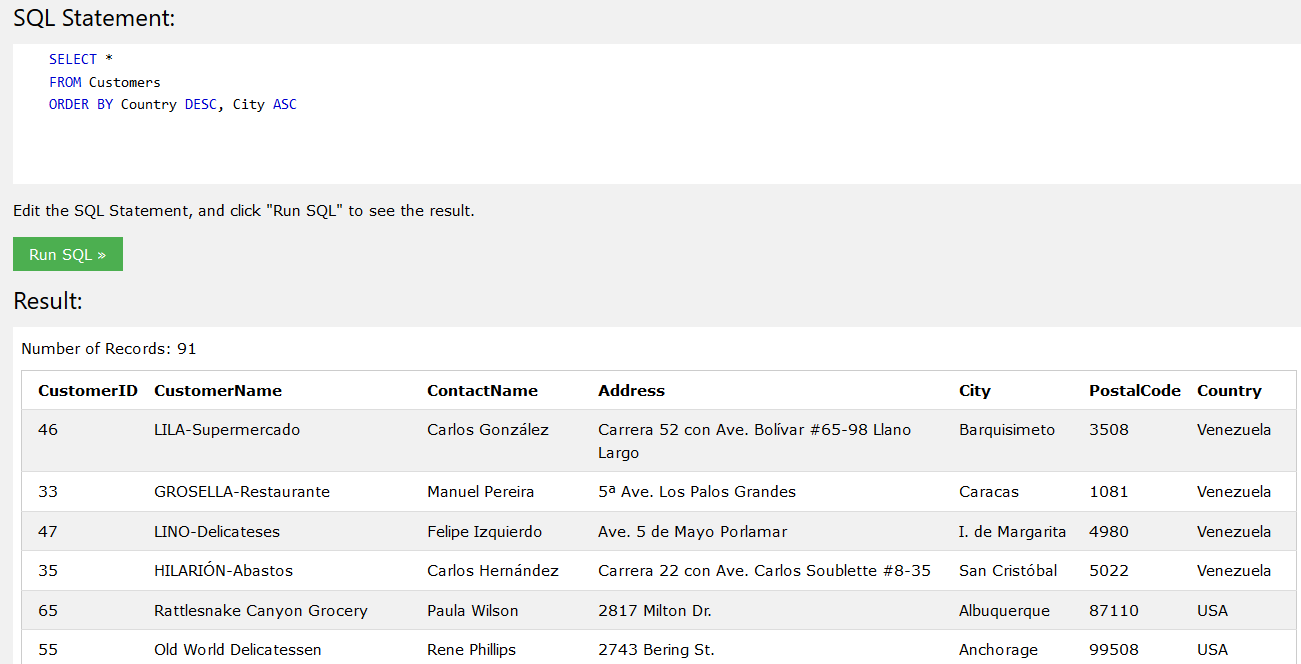
You can even sort by more than one field at a time. Because there are multiple cities and countries listed in the Customers table, you could sort first by Country in descending order and then by City in ascending order with the following query:

**SELECT** \*

**FROM** Customers

**ORDER BY** Country **DESC**, City **ASC**🡨 notice that I don’t have to list Asc here as an argument because SQL assumes that if there is no

argument specified, it will sort that field in ascending order



**JOINING tables**

The preferred way of joining tables in today’s database industry is by using the keywords “inner join”, “outer join”, etc. You will learn these in more advanced database courses. For this lab assignment, I want you to learn an older way of joining tables.

Joins are completer in two steps. Identifying the tables in the FROM clause and identifying the links in the WHERE clause.

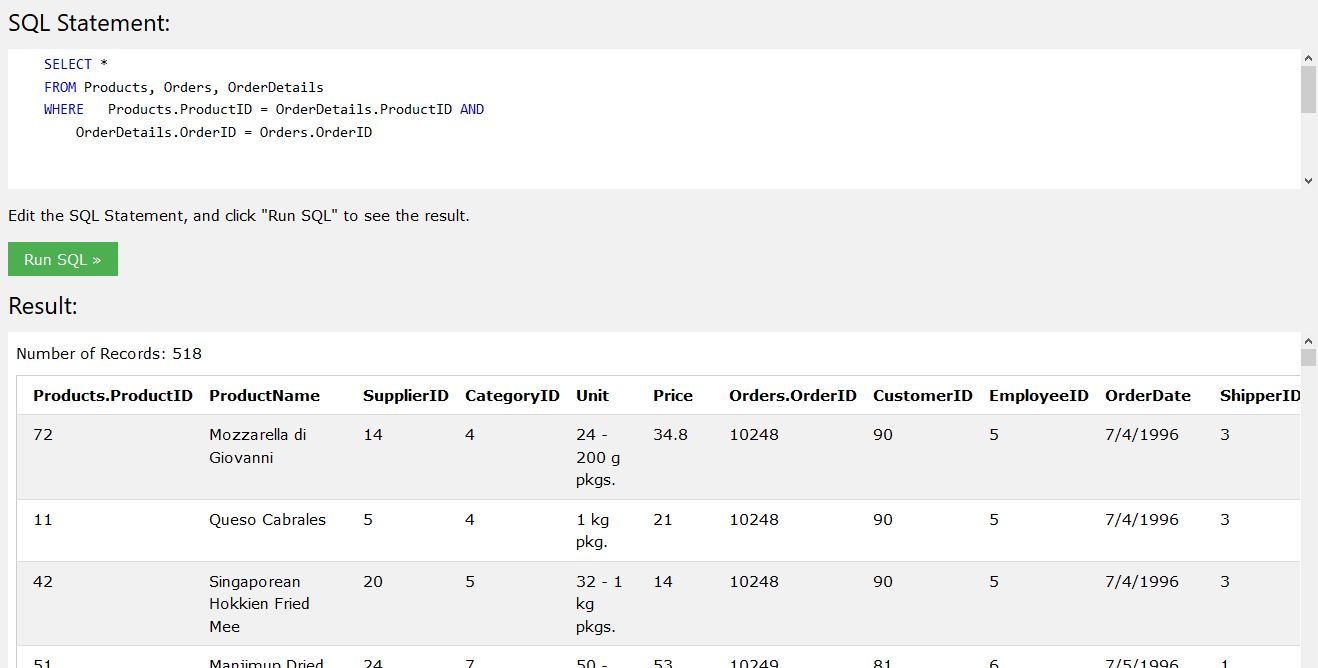
The WHERE clause method allows you to specify the all of the tables to be included in your FROM clause, but the joins are specified in the WHERE clause. For example, assume I want to join the products, orders, and orderdetails tables. I specify all three of them in the FROM clause, but I join them on their common fields in the WHERE clause as follows:

**SELECT** \*

**FROM** Products, Orders, OrderDetails

**WHERE** Products.ProductID = OrderDetails.ProductID **AND**

OrderDetails.OrderID = Orders.OrderID



If you are having difficulty understanding the concept of joins, (or any of the commands presented above), w3schools.com does an excellent job of explaining them. To view the SQL command tutorials, click on any of the SQL Tutorial topics on the left hand side of this screen: <http://www.w3schools.com/sql/>

**Now, it’s your turn…**

Use the following worksheet to work your queries through the w3schools sql labs. Once you think you have the correct quest, cut and paste it into the template below. Additionally, take a screenshot of the results for each answer. Record the query and the number of records in the table following each question. Once you have completed all 25 questions, you can open the Database Lab Quiz page. In this you will see two questions for each of the 25 listed below. The first will ask you to enter the number of records your query revealed. The second will ask you to paste in your query.

It is important that you complete all of these questions before starting the lab quiz as you will nee to fill in all answers in one sitting. There are 50 responses required and you have 2 hours to complete filling it out.

Now, Let’s get started. Remember, if you run into a wall and can’t figure out the correct query, contact your instructor with your questions before giving up. Send them the query as you have it and they will help you see what you are missing. They will not give you the right answer. Their charge is to guide you to figuring it out.

**Scenario**

You are working at the helpdesk of Foods, Inc. and are given the task of answering questions using the database we have been querying during this exercise. The questions you have received each Day are presented below:

As this is your first job working with databases, your manager wants you to familiarize yourself with all of the tables in the database. Therefore, he asks you to produce a query for each table that displays all of the columns (fields) and all of the rows (records) using a separate SELECT query for each table. The tables are listed below.

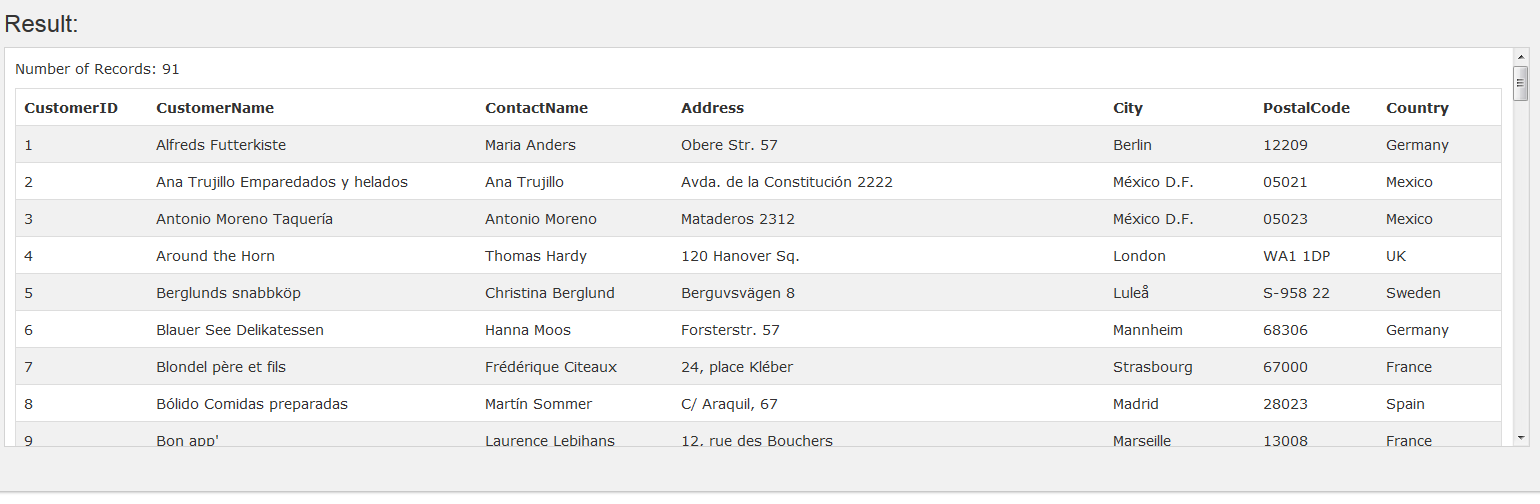
1. Customers

To get you started, the first query is done for you. This first one is a “freebie”.

List all of your queries and their results in the Answer Template that is provided for you using this format:

|  |  |
| --- | --- |
| Query | Number of records returned |
| SELECT \* FROM Customers; | 91 |

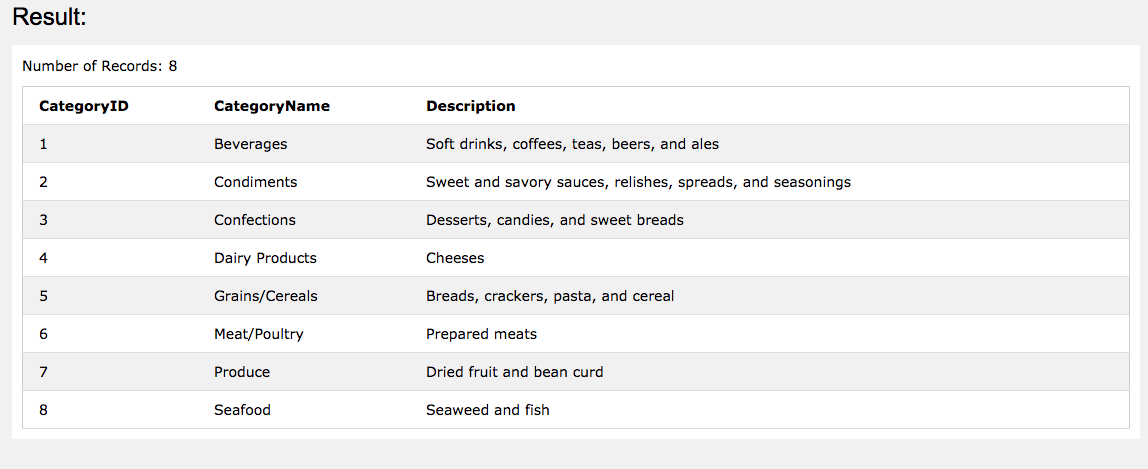
Screenshot of output: [Note that you only need to display the first screen of output for each query.]



1. Categories

|  |  |
| --- | --- |
| Query | Number of records returned |
| SELECT \* FROM Categories; | 8 |

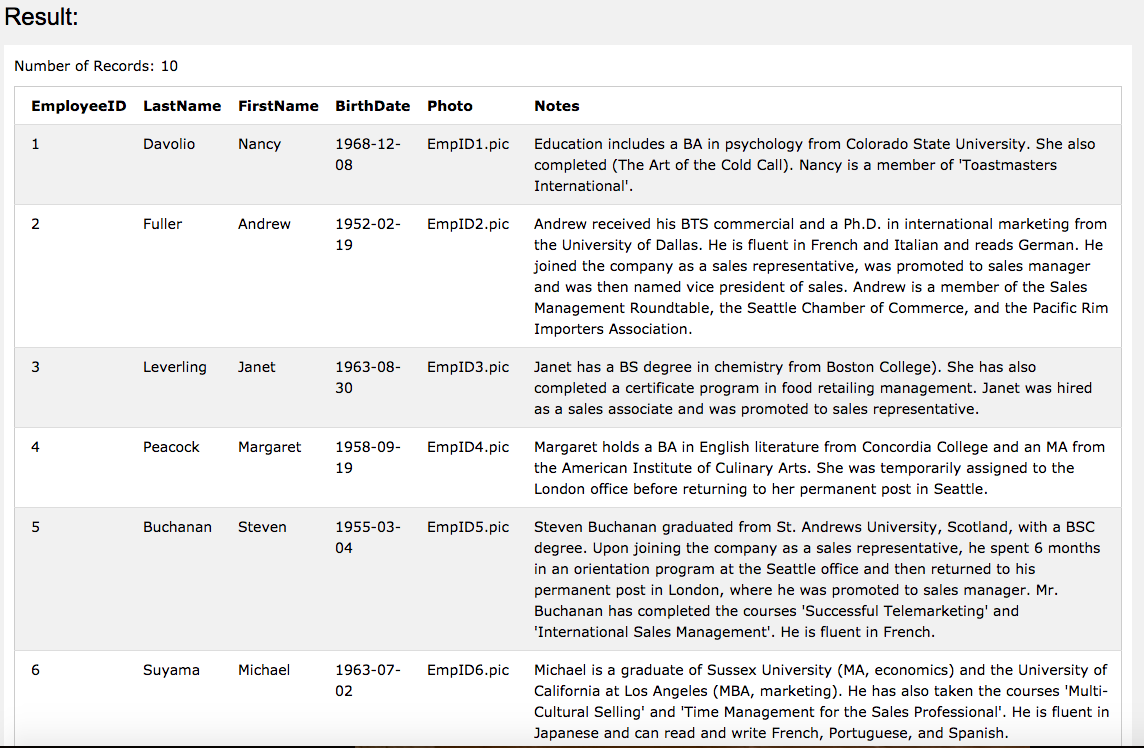
Screenshot of output: [Note that you only need to display the first screen of output for each query.]



1. Employees

|  |  |
| --- | --- |
| Query | Number of records returned |
| SELECT \* FROM Employees; | 10 |

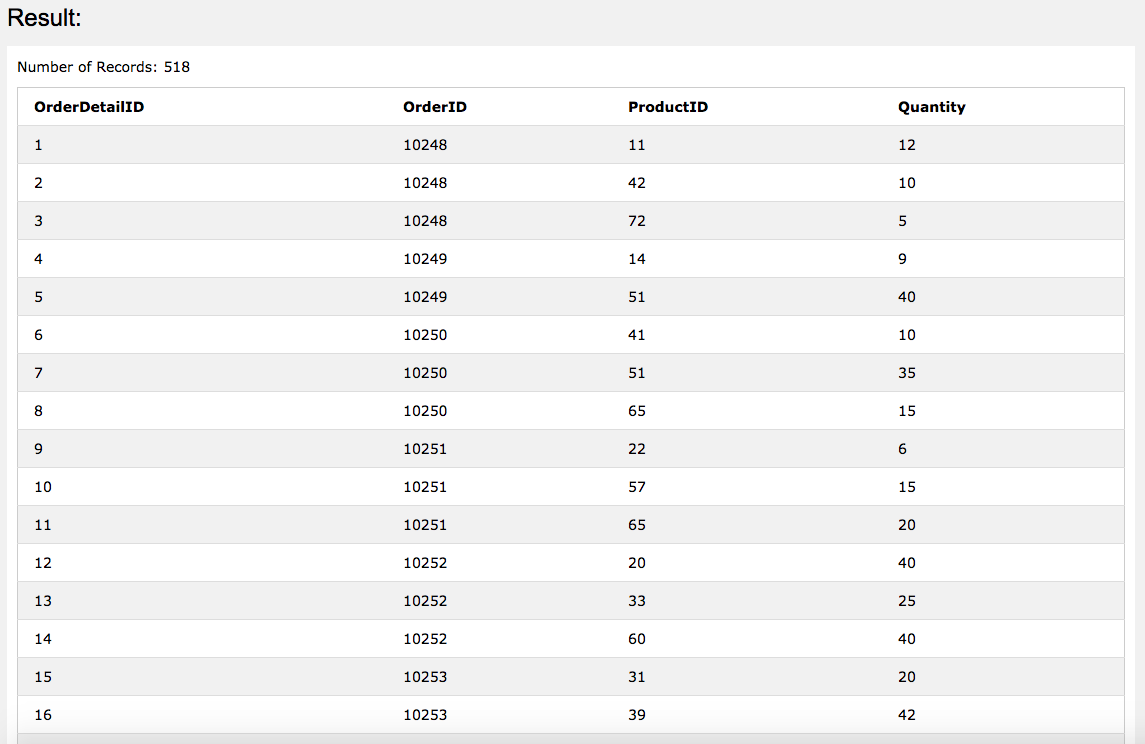
Screenshot of output: [Note that you only need to display the first screen of output for each query.]



1. OrderDetails

|  |  |
| --- | --- |
| Query | Number of records returned |
| SELECT \* FROM OrderDetails; | 518 |

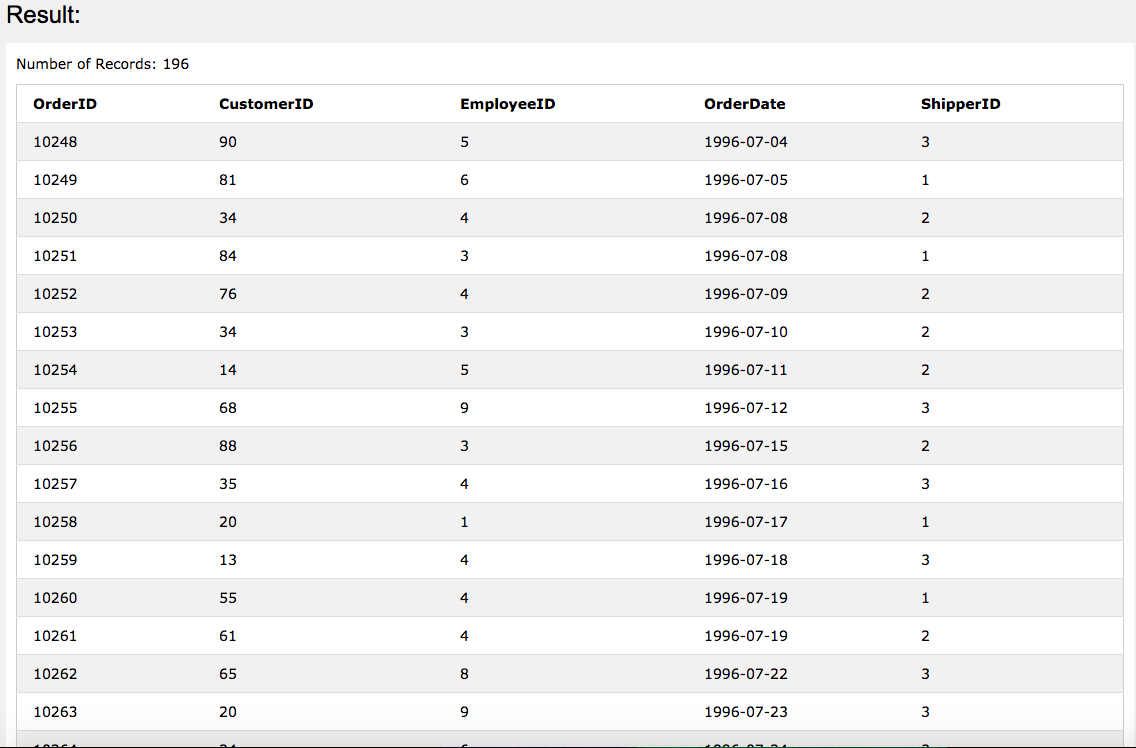
Screenshot of output: [Note that you only need to display the first screen of output for each query.]



1. Orders

|  |  |
| --- | --- |
| Query | Number of records returned |
| SELECT \* FROM Orders; | 196 |

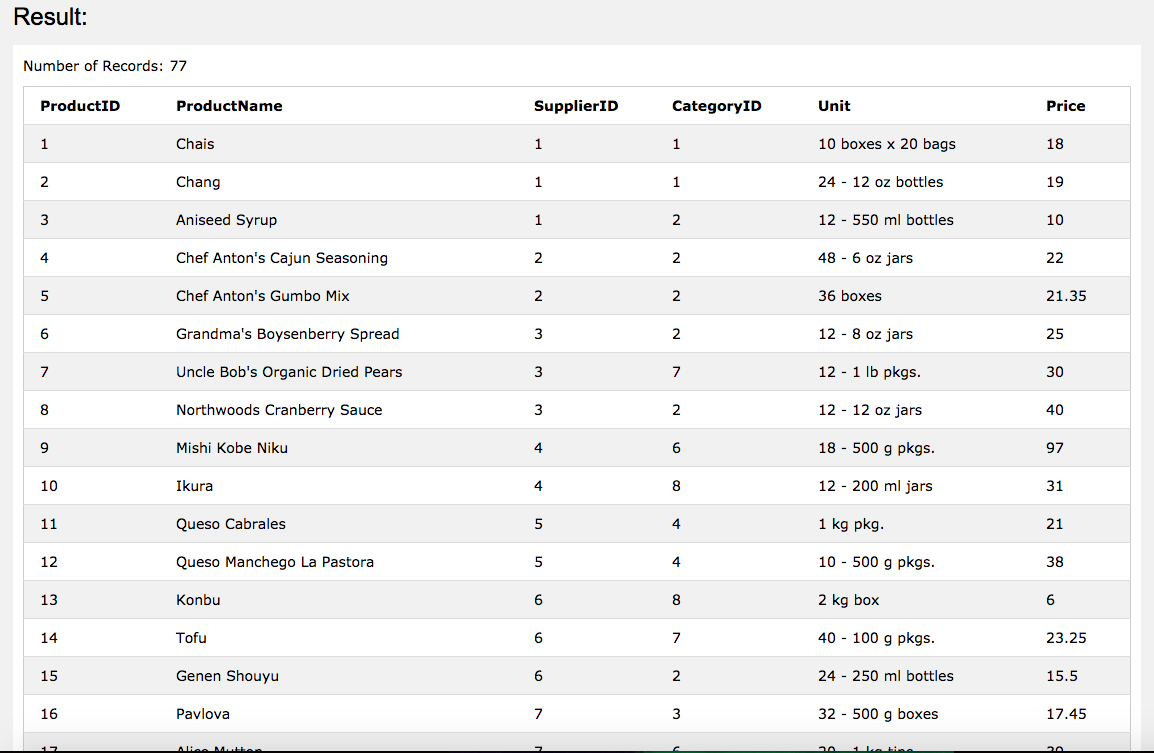
Screenshot of output: [Note that you only need to display the first screen of output for each query.]



1. Products

|  |  |
| --- | --- |
| Query | Number of records returned |
| SELECT \* FROM Products; | 77 |

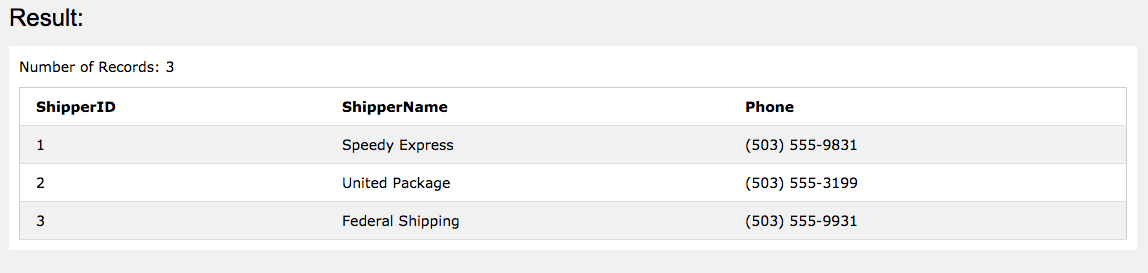
Screenshot of output: [Note that you only need to display the first screen of output for each query.]



1. Shippers

|  |  |
| --- | --- |
| Query | Number of records returned |
| SELECT \* FROM Shippers; | 3 |

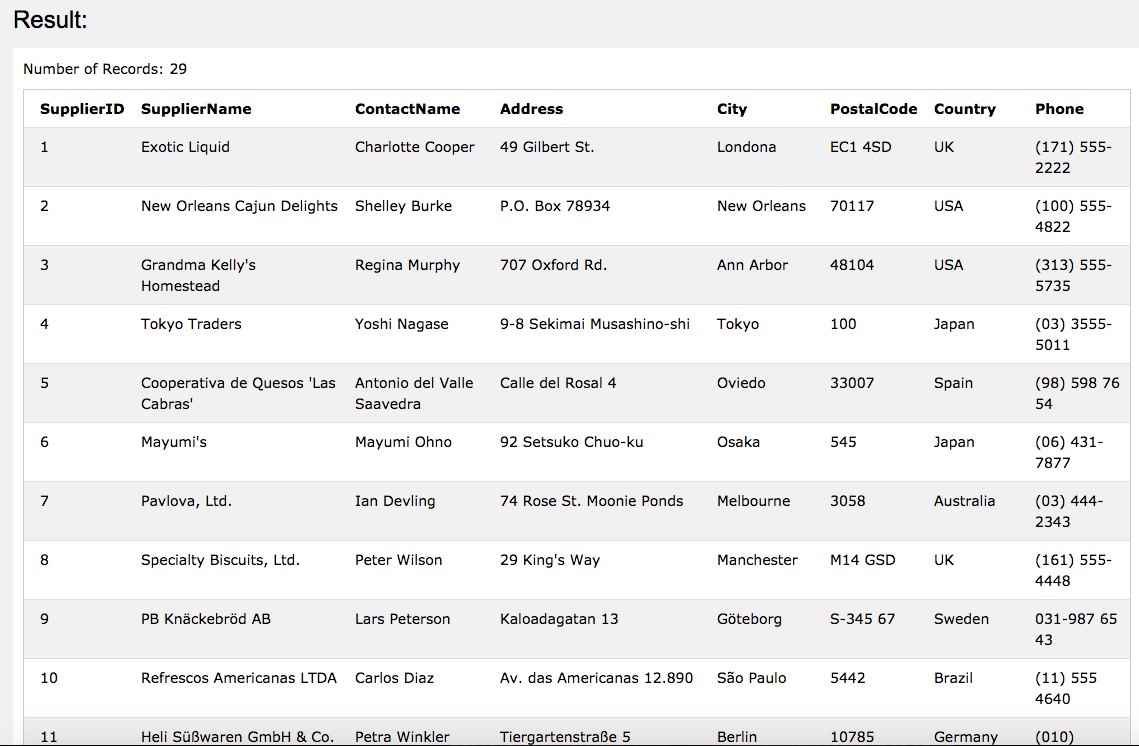
Screenshot of output: [Note that you only need to display the first screen of output for each query.]



1. Suppliers

|  |  |
| --- | --- |
| Query | Number of records returned |
| SELECT \* FROM Suppliers; | 29 |

Screenshot of output: [Note that you only need to display the first screen of output for each query.]

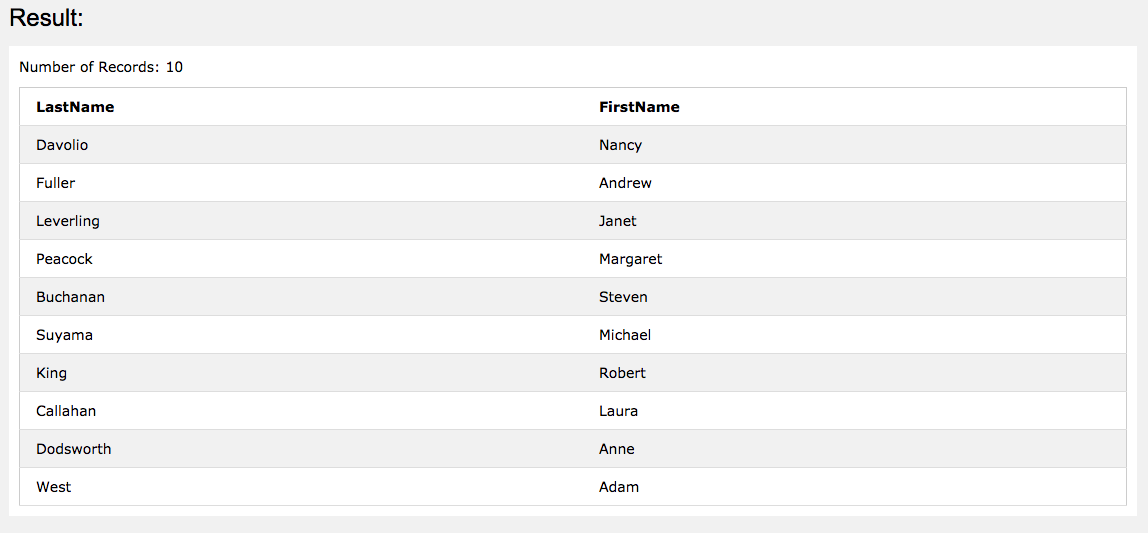


**After familiarizing yourself with the tables, your manager knows that you can handle the remaining requests that come in to the helpdesk.**

1. The manager of Human Resources wants a list of all of the employees who work at the company. He only wants to see the employees’ last name and first name. [You may assume that everyone who works at the company is included in the Employees table].

|  |  |
| --- | --- |
| Query | Number of records returned |
| SELECT LastName, FirstName FROM Employees; | 10 |

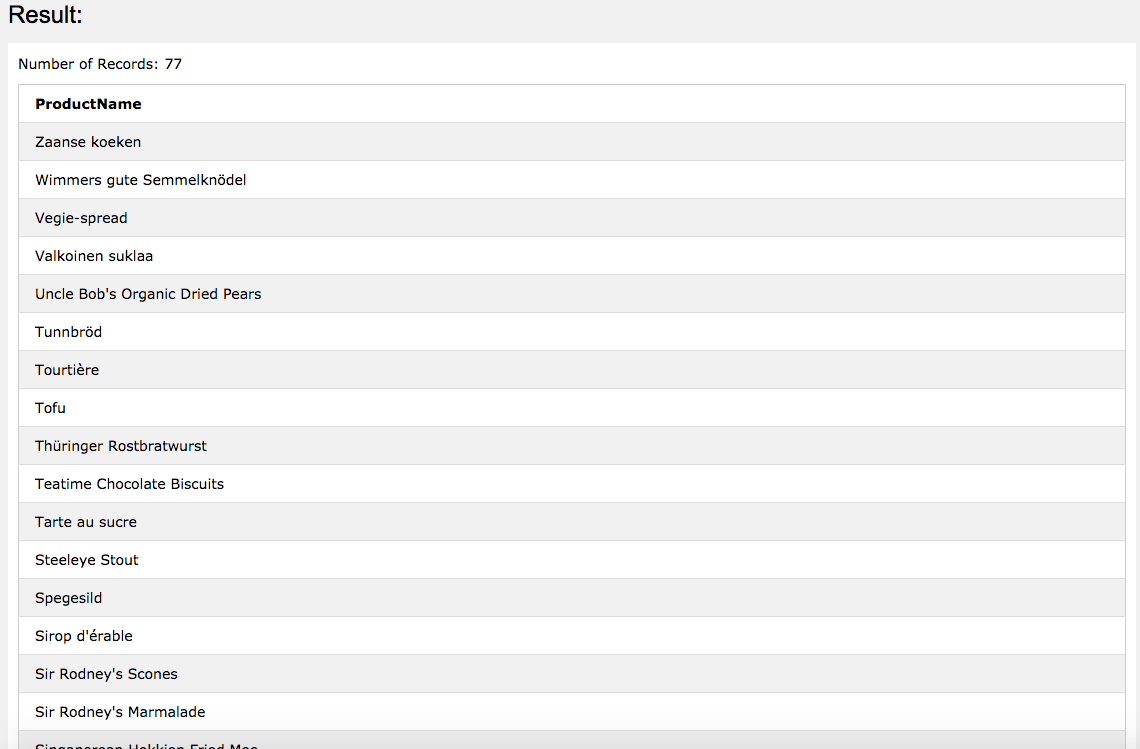
Screenshot of output: [Note that you only need to display the first screen of output for each query.]



1. Your manager asks you for a list of the product names that the company sells. He does not want everything related to each product – only each product’s name. He wants them sorted alphabetically (in descending order from Z-A). [The Products table contains all of the products that are sold by the company.]

|  |  |
| --- | --- |
| Query | Number of records returned |
| SELECT ProductName  FROM Products  ORDER BY ProductName DESC; | 77 |

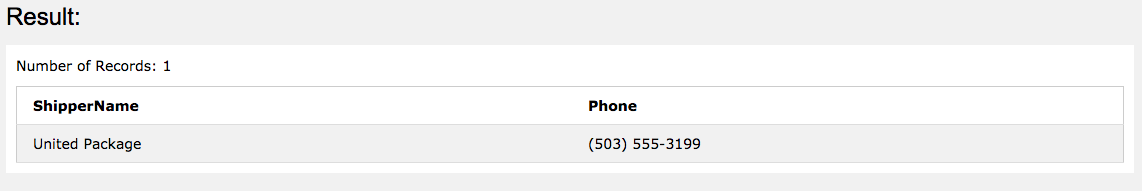
Screenshot of output: [Note that you only need to display the first screen of output for each query.]



1. The manager of the Purchasing department needs the phone number of the United Package shipper.

|  |  |
| --- | --- |
| Query | Number of records returned |
| SELECT ShipperName, Phone  FROM Shippers  WHERE ShipperName=’United Package’; | 1 |

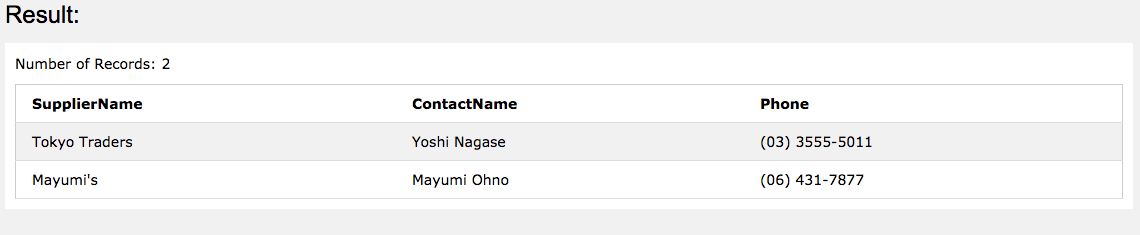
Screenshot of output: [Note that you only need to display the first screen of output for each query.]



1. The Accounting department manager requests a list of all of the suppliers, contacts, and phone numbers of the suppliers in Japan.

|  |  |
| --- | --- |
| Query | Number of records returned |
| SELECT SupplierName, ContactName, Phone  FROM Suppliers  WHERE Country=’Japan’; | 2 |

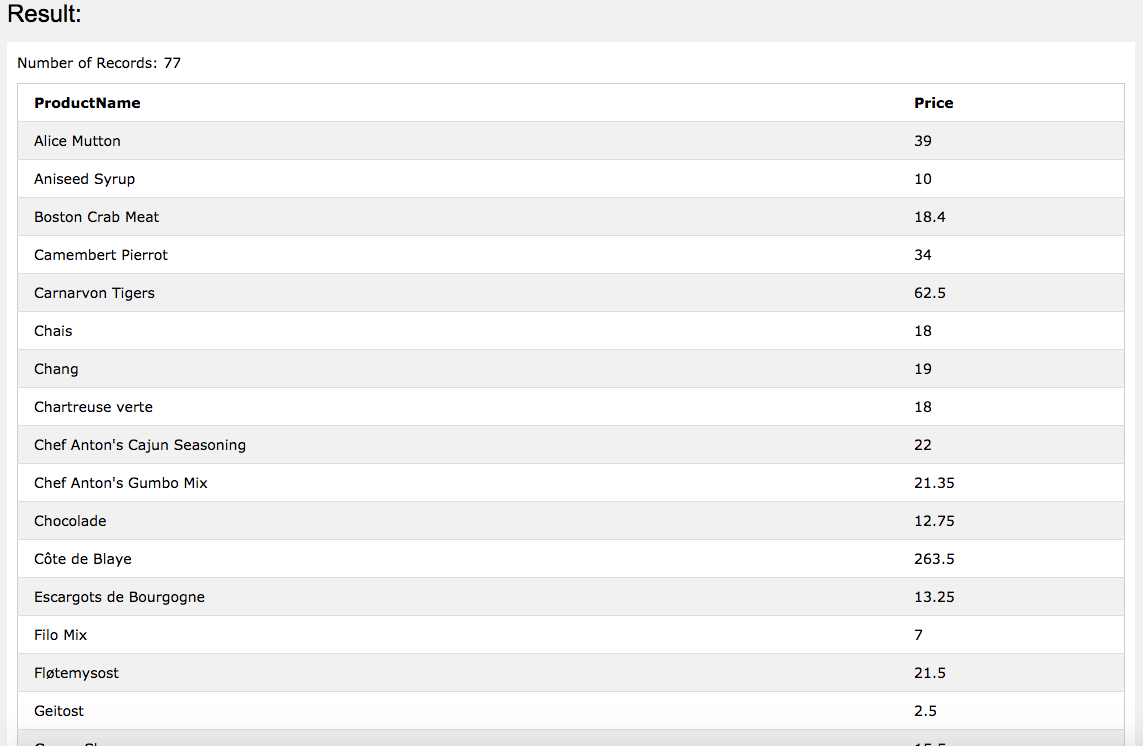
Screenshot of output: [Note that you only need to display the first screen of output for each query.]



1. A customer calls in and wants a list of all of your products and their prices. Your results should be sorted in alphabetical order from A-Z.

|  |  |
| --- | --- |
| Query | Number of records returned |
| SELECT ProductName, Price  FROM Products  ORDER BY ProductName ASC; | 77 |

Screenshot of output: [Note that you only need to display the first screen of output for each query.]



1. Another customer calls in requesting all of your seafood products. Note that he only wants the name of the product in his report – not all of the fields. [Hint: You will need to join the Products table to the Categories table to get this information. Your WHERE clause should include the criterion “WHERE CategoryName = "Seafood".]

|  |  |
| --- | --- |
| Query | Number of records returned |
| Select ProductName  FROM Products, Categories  WHERE Products.CategoryID = Categories.CategoryID AND  CategoryName = "Seafood" | 12 |

Screenshot of output: [Note that you only need to display the first screen of output for each query.]



1. The manager of the accounting department needs a list of all of the suppliers (Supplier name only) in either the USA or the UK. [Hint: You must use the OR operator in your WHERE clause to receive credit for this question.]

|  |  |
| --- | --- |
| Query | Number of records returned |
| SELECT SupplierName  FROM Suppliers  WHERE Country IN ("USA") OR Country IN ("UK"); | 6 |

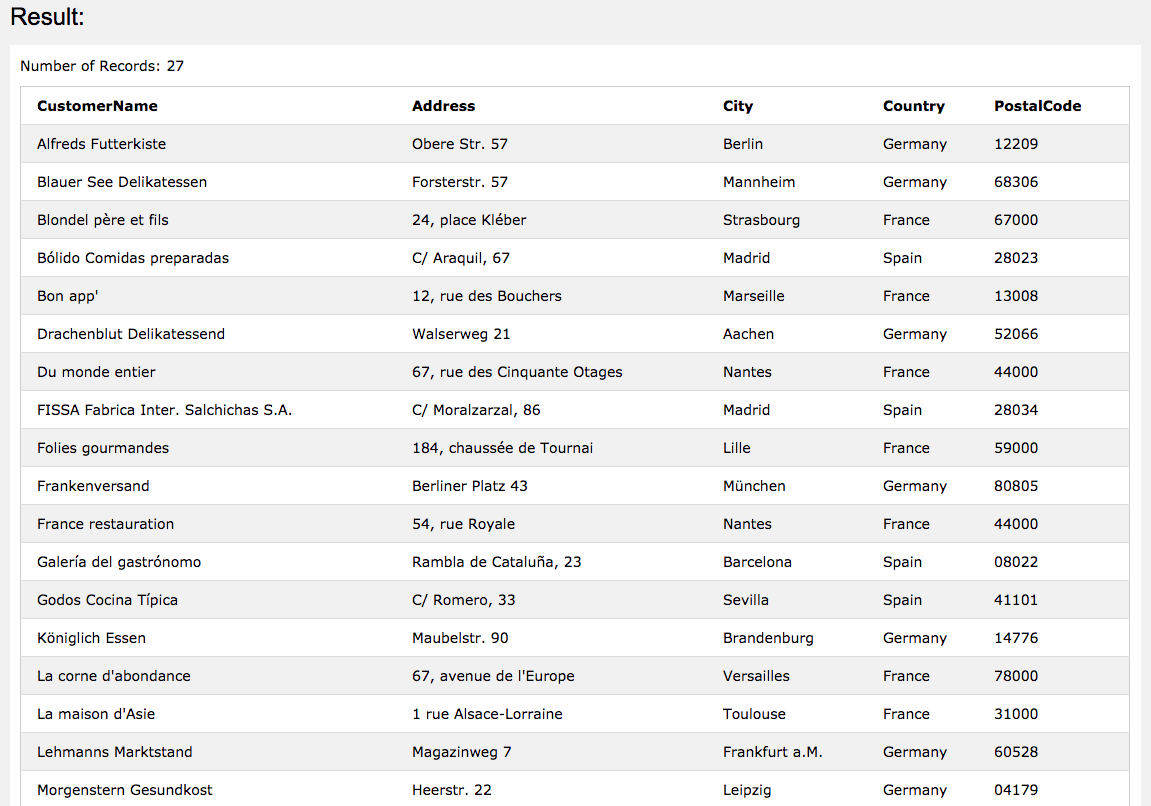
Screenshot of output: [Note that you only need to display the first screen of output for each query.]



1. The Accounts Receivable clerk needs a list of all the customers (including customer names, addresses, city, country, and postal codes) who are located in Germany, France, or Spain. [Hint: You must use the IN operator in your WHERE clause to receive credit for this question.]

|  |  |
| --- | --- |
| Query | Number of records returned |
| SELECT CustomerName, Address, City, Country, PostalCode  FROM Customers  WHERE Country IN ("Germany", "France", "Spain"); | 27 |

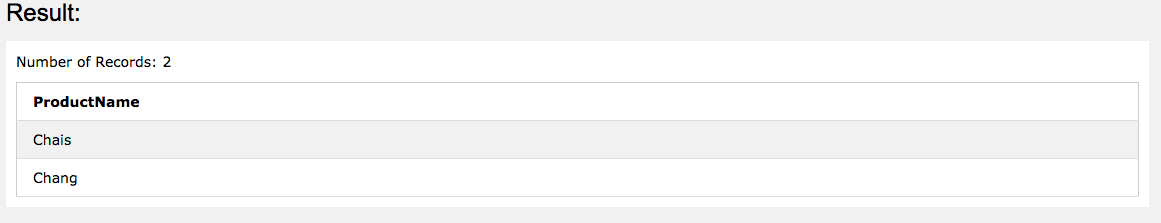
Screenshot of output: [Note that you only need to display the first screen of output for each query.]



1. Just out of curiosity, you want to produce a listing of the products (product name only) that are supplied by SupplierID 1 and are in CategoryID 1.

|  |  |
| --- | --- |
| Query | Number of records returned |
| SELECT ProductName  FROM Products  WHERE CategoryID = 1 AND SupplierID = 1; | 2 |

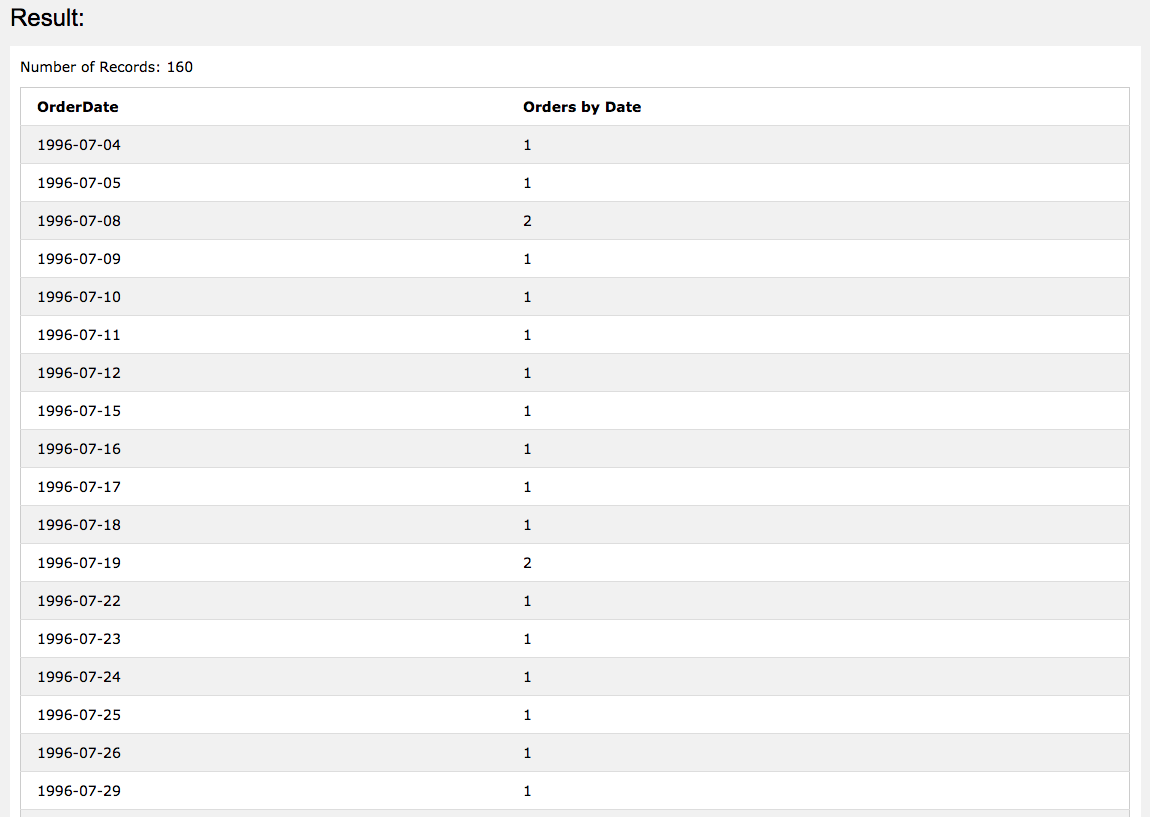
Screenshot of output: [Note that you only need to display the first screen of output for each query.]



1. The Human Resources department requests a list that contains the number (i.e. count) of orders taken on each date, grouped by the order date. [Be sure to use an aggregate operator in your query to receive credit for this question.]

|  |  |
| --- | --- |
| Query | Number of records returned |
| SELECT OrderDate, COUNT(OrderDate) AS [Orders by Date]  FROM Orders  GROUP BY OrderDate  ORDER BY OrderDate; | 160 |

Screenshot of output: [Note that you only need to display the first screen of output for each query.]



1. One of the employees in the IT department asks for the highest order ID that is in the Orders table. [Hint: To receive credit for this question, you must use the MAX function to perform this task.]

|  |  |
| --- | --- |
| Query | Number of records returned |
| SELECT MAX(OrderID) AS [Highest Order ID]  FROM Orders | 1 |

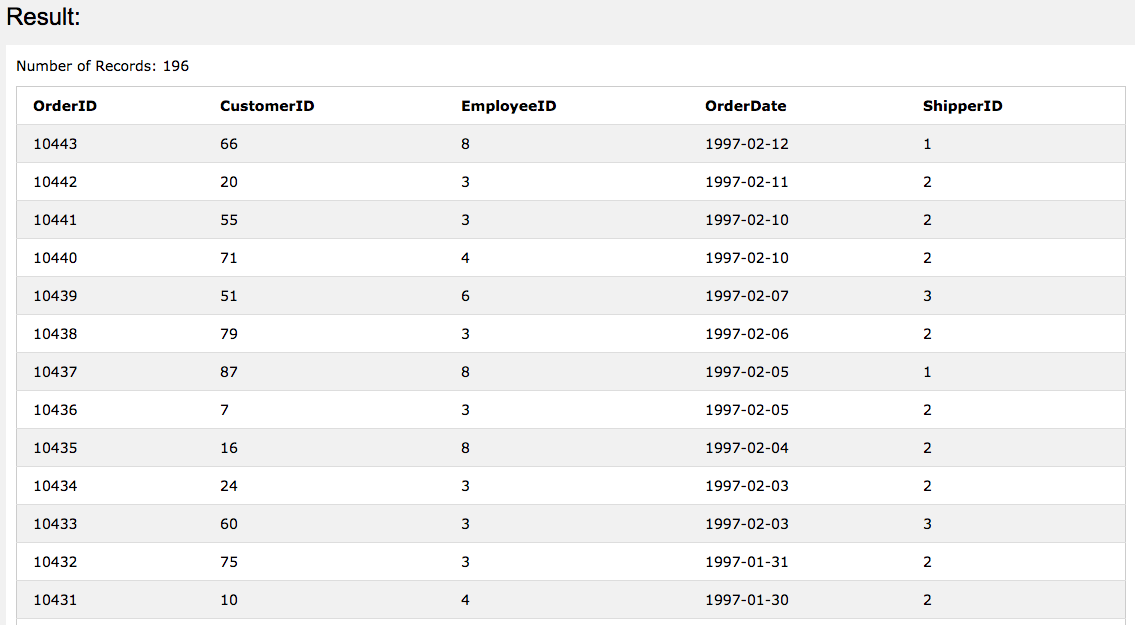
Screenshot of output: [Note that you only need to display the first screen of output for each query.]



1. To double check your results from the previous question, you decide to run a query that returns all of the rows and columns from the Orders table sorted in descending order. If you see that the order ID of the first record in your output equals the answer you obtained in your previous question, you know you were correct.

|  |  |
| --- | --- |
| Query | Number of records returned |
| SELECT \*  FROM Orders  ORDER BY OrderID DESC; | 196 |

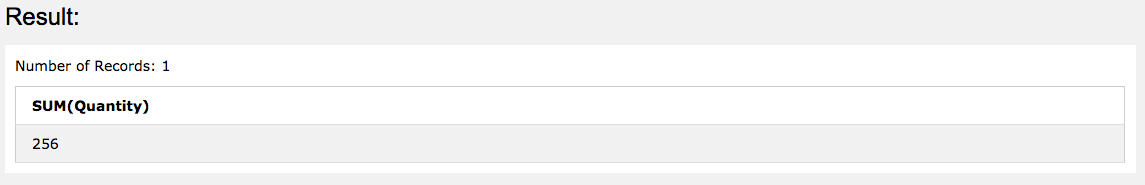
Screenshot of output: [Note that you only need to display the first screen of output for each query.]



1. Your boss wants to know the total quantity of Boston Crab Meat that has ever been ordered. [Hint: You will have to join the OrderDetails and Products tables. Your WHERE clause should contain the criterion ProductName = "Boston Crab Meat", and you will have to use the SUM function in your SELECT clause.]

|  |  |
| --- | --- |
| Query | Number of records returned |
| SELECT SUM(Quantity)  FROM OrderDetails, Products  WHERE OrderDetails.ProductID=Products.ProductID AND ProductName="Boston Crab Meat" | 1 |

Screenshot of output: [Note that you only need to display the first screen of output for each query.]



1. The manager of your Customer Relationship department wants a list of all of the customers whose name begins with the letter “D”.

|  |  |
| --- | --- |
| Query | Number of records returned |
| SELECT CustomerName  FROM Customers  WHERE CustomerName LIKE 'D%' | 3 |

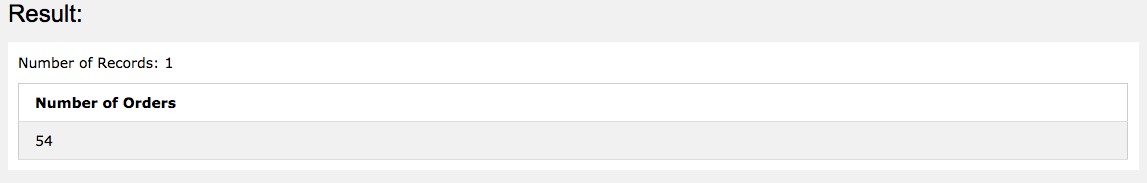
Screenshot of output: [Note that you only need to display the first screen of output for each query.]



1. Speedy Express, one of your shippers, wants to know how many orders it has ever shipped for you. [Hint: You need to join the Shippers and Orders tables and include ShipperName = "Speedy Express" in the WHERE clause. Remember to use an aggregate operator to receive full credit for this question.]

|  |  |
| --- | --- |
| Query | Number of records returned |
| SELECT Count(Orders.ShipperID) AS [Number of Orders]  FROM Orders, Shippers  WHERE Shippers.ShipperID=Orders.ShipperID AND ShipperName="Speedy Express" | 1 |

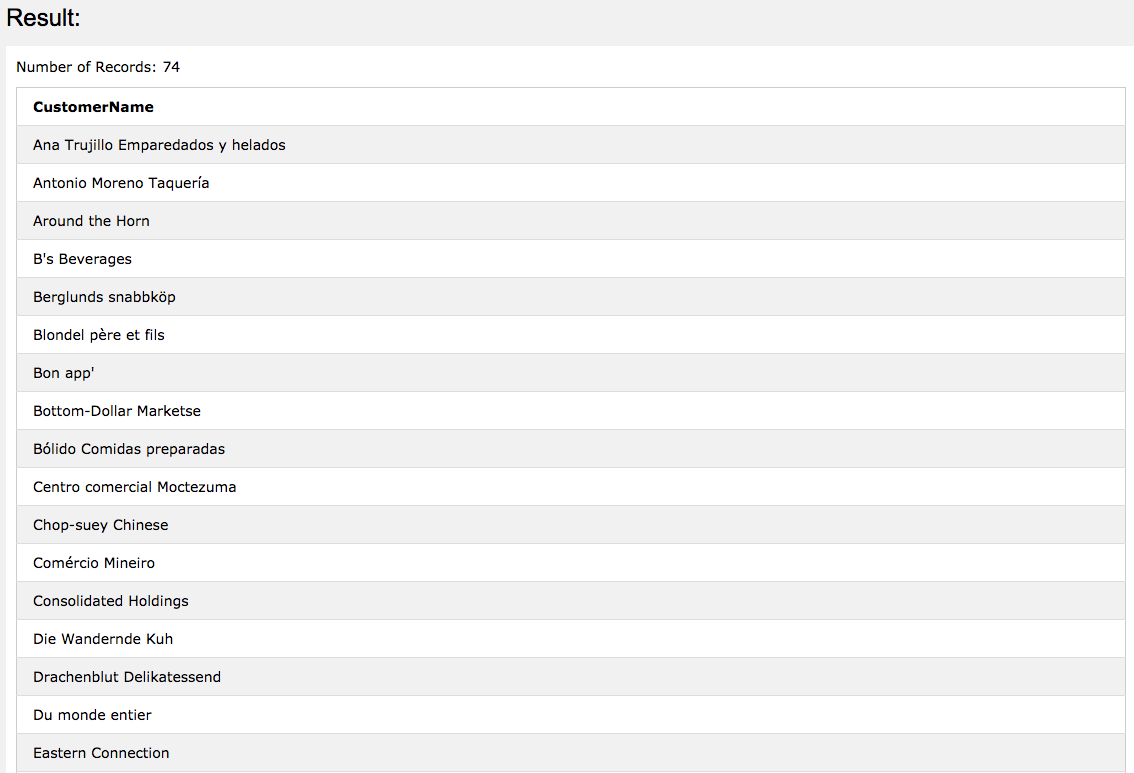
Screenshot of output: [Note that you only need to display the first screen of output for each query.]



1. Someone from the tax department wants a unique list of all of the customers who have ever placed an order with the company. He only needs the customer name, sorted from A-Z. [Hint: You will need to join the Customers table with the Orders table because it is possible that a customer may exist in the customers table who has not yet placed an order with your company.]

|  |  |
| --- | --- |
| Query | Number of records returned |
| SELECT DISTINCT CustomerName  FROM Customers, Orders  WHERE Customers.CustomerID=Orders.CustomerID  ORDER BY CustomerName | 74 |

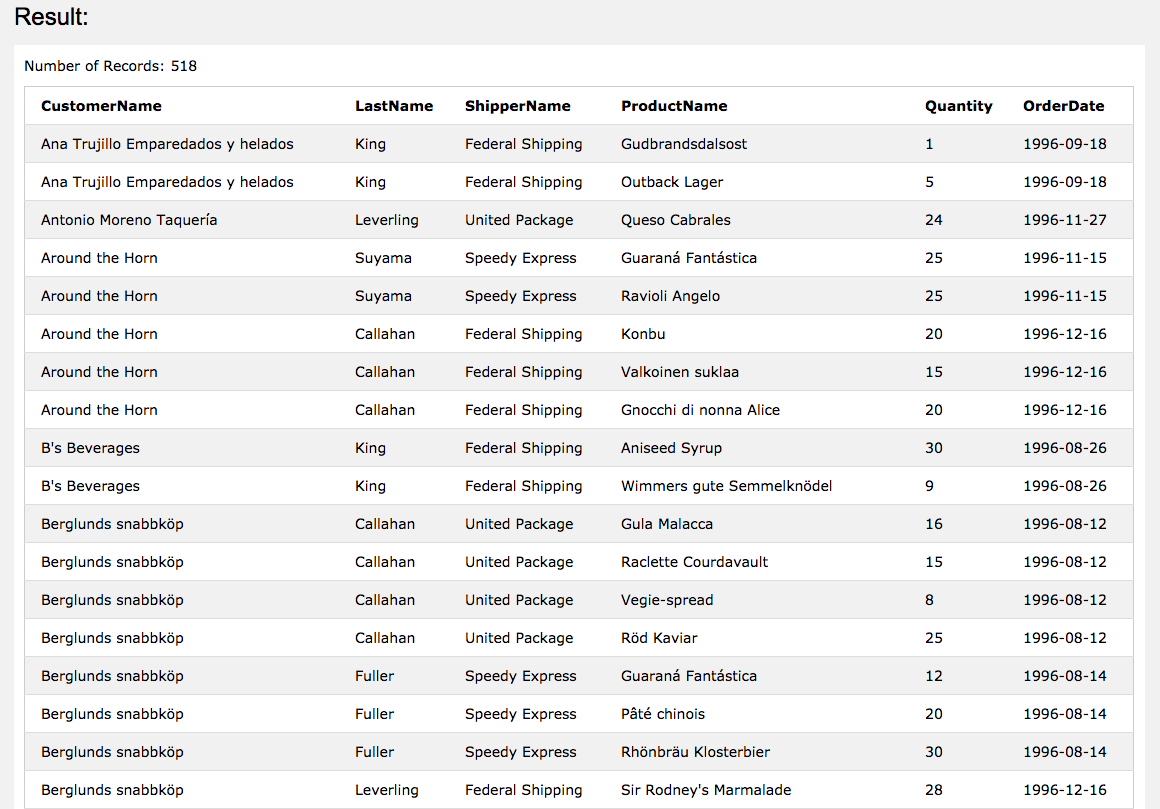
Screenshot of output: [Note that you only need to display the first screen of output for each query.]



1. The president of the company wants a list of all orders ever taken. He wants to see the customer name, the last name of the employee who took the order, the shipper who shipped the order, the product that was ordered, the quantity that was ordered, and the date on which the order was placed. [Hint: You will need to join the following tables: Customers, Employees, Shippers, Orders, OrderDetails, Products, and to get all of the necessary information.]

|  |  |
| --- | --- |
| Query | Number of records returned |
| SELECT CustomerName, LastName, ShipperName, ProductName, Quantity, OrderDate  FROM Customers, Employees, Shippers, Products, Orders, OrderDetails  WHERE Customers.CustomerID = Orders.CustomerID AND Employees.EmployeeID = Orders.EmployeeID AND Shippers.ShipperID = Orders.ShipperID AND Orders.OrderID = OrderDetails.OrderID AND OrderDetails.ProductID = Products.ProductID  ORDER BY CustomerName; | 518 |

Screenshot of output: [Note that you only need to display the first screen of output for each query.]



**Congratulations!**

**You have just helped numerous people inside and outside of your company to solve all kinds of problems! You deserve a raise!**

**Deliverables**:

Now that you have completed these tasks, Open the database lab test and submit your answers. You will be cutting and pasting your queries into the selected answers and entering the number of records found. Be careful when entering your answers.