Teaching Case

SQL: An Introduction to SQL Lesson and Hands-On Lab

Gayle Jesse
West Liberty University
West Liberty, WV, 26074
gayle.jesse@gmail.com

Abstract

We live in a world run by databases. Thus, knowing Structure Query Language (SQL) is vital to extract data from a database. The following teaching case is a hands-on introduction to SQL lab activity based on a scenario of working at a help desk for Foods, Inc using W3 Schools. W3 schools web-based environment is stable, consistent, and allows numerous users at one time without ever altering the database. Students begin the case by learning the background information, database structure, and working through a teaching lesson. The lesson teaches students the basics of writing queries from basic select statements through joins. After students complete the lesson on querying basics they are ready to begin the assignment. The assignment requires students to write 25 queries which, are to be recorded in an answer template designed for easier grading and reduction of cheating.

Keywords: SQL, Introduction Database Course, Applied Lab, W3C Schools

1. BACKGROUND

A database is an organized collection of data that consists of tables, queries, views, reports, and other objects. Structure Query Language (SQL) is a standard language for storing, manipulating and retrieving data in databases (W3 Schools, 2018). As stated in the abstract, W3 schools web-based environment is stable, consistent, and allows numerous users at one time without ever altering the database. The database can easily be restored by clicking the "Restore Database" button. Meaning each user gets a separate instance they can reset at any time.

In this lab, you are a recently hired database professional working at a helpdesk for Foods, Inc., a grocery store distributor located in Manhattan. Foods, Inc, is a specialty store selling a variety of unique upscale imported food. From coffee to tofu as well as deserts to dairy products. Currently, there are twenty-nine (29) suppliers for Food, Inc. located worldwide with several shippers. Foods, Inc. employs 10 people with an average of 100 customers. As a help desk employee, it is your responsibility to answer questions from customers, suppliers, and internal managers of your company by querying your Foods, Inc.'s database using SQL.

2. DATABASE STRUCTURE

Foods, Inc.'s database consists of eight (8) tables that are already created and populated with data.

1. Customers
2. Categories
3. Employees
4. OrderDetails
5. Orders
6. Products
7. Shippers
8. Suppliers

3. LESSON: 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 in the Appendix Figure 1.

The basic format of a query is as follows:

SELECT [fieldname]
FROM [tablename]
[**WHERE** condition]

Note: 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.

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. See Appendix Figure 2.

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;

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]
FROM Customers;

Note: If you have a space in your alias, you must enclose the alias in square brackets.

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;

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;
```

Note: ASC does not have to be listed here as an argument because SQL assumes that if there is no argument specified, it will sort that field in ascending order

JOINING tables
In all of our previous examples, we have been querying a single table, Customers. It is obvious, however, that our one Customers table does not give us the whole picture. In order to get the "bigger picture", we need to be able to combine data from multiple tables.

The way this is accomplished is by "joining" tables based on fields that they have in common. For example, click on the Products table. Notice that it has the following fields in it:

ProductID, ProductName, SupplierID, CategoryID, Unit, Price.

Now, click on the Categories table. Notice that it has the following fields in it:

CategoryID, CategoryName, Description

What field exists in both tables? The CategoryID field, right? Therefore, we can write a query that will allow us to retrieve data out of both tables by joining both tables on that common field:

```
SELECT *
FROM Products
INNER JOIN Categories ON
Categories.CategoryID =
Products.CategoryID;
```

Note the keywords "INNER JOIN" above. These keywords are necessary to tell SQL that the two tables are to be linked based this common field.

There are other types of joins in database terminology (e.g. Left Outer Joins, Right Outer Joins, and Full Joins), but those are beyond the scope of this assignment.

The preferred way of joining tables in today's database industry is by using the keywords "inner join", "outer join", etc. In this assignment, I want you to learn an older way of performing inner joins in case you ever run into them in industry.

An inner join can be accomplished in one of two ways: in the FROM clause (as we illustrated above) or 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/

4. ASSIGNMENT

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 the previous lesson on querying basics. The 25 questions you have received and need to answer are as follows.

This is your first job working with databases, so 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 of the eight (8) tables displaying all of the columns (fields) and all of the rows (records) using a separate SELECT query for each table. The eight (8) tables are listed below.

25 SQL Questions

1. Customers

To get you started, here is the first query.

```
SELECT *
FROM Customers
```

Instructions:

- Download the Answer Template.
- Save as: YourFirstName_YourLastName_IntroSQL Lab.
- List all 25 queries and their results in the Answer Template (Appendix Figure 3) that is provided for you using the format below.

Query	Number of records returned
SELECT * FROM Customers	91

- Paste a Screen Capture of output; Screen Capture must include three items: Sql Statement, Results, and computer date and time (Appendix Figure 4).
  Note: You only need to display the first screen of output for each query.

2. Categories
3. Employees
4. OrderDetails
5. Orders
6. Products
7. Shippers
8. Suppliers

After familiarizing yourself with the eight (8) tables, your manager knows that you can handle the remaining requests that come in to the helpdesk.

For each of the remaining items, use the same format for your queries and results as you did in the previous 8 questions and continue putting your answers in the Answer Template provided.

9. 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].
10. 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.]

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

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

13. 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.

14. 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".]

15. 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.]

16. 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.]

17. 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.

18. 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.]

19. 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.]

20. 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.

21. 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.]

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

23. 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.]

24. 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. Joining the two tables together with an inner join ensures that your list of customers only includes those customerid's that exist in both tables (i.e. customers who have actually placed an order).]

25. 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 get all of the necessary information.]

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

Deliverables
Submit your completed Answer Template to your professor by the due date.

5. REFERENCES

W3 Schools. (2018). SQL Tutorial. Retrieved July 4, 2018, from https://www.w3schools.com/sql/default.asp

Editor Notes: Teaching Notes are available for this teaching case, please contact the author directly.

ISSN: 2473-3857
v4 n4767

Appendices

Figure 1

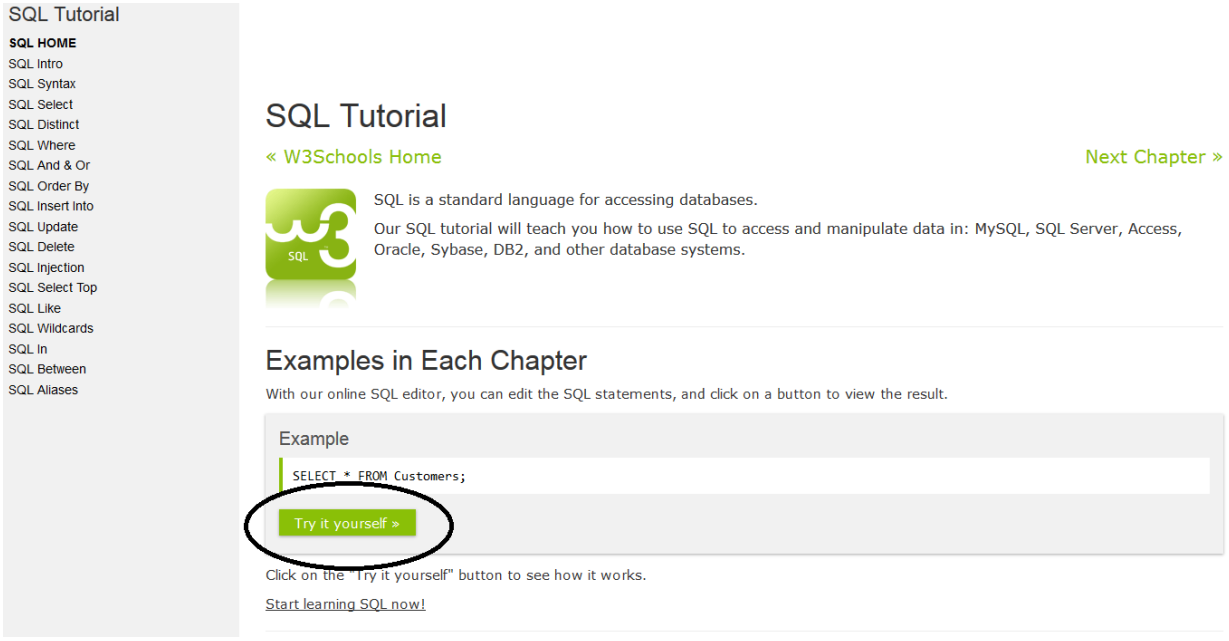
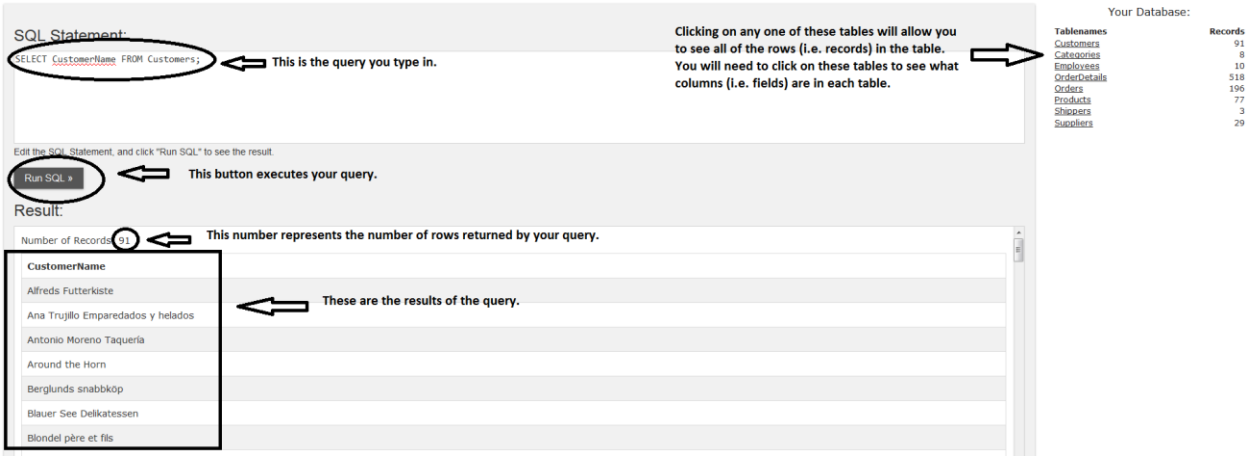


Figure 2



ISSN: 2473-3857
v4 n4767

Figure 3

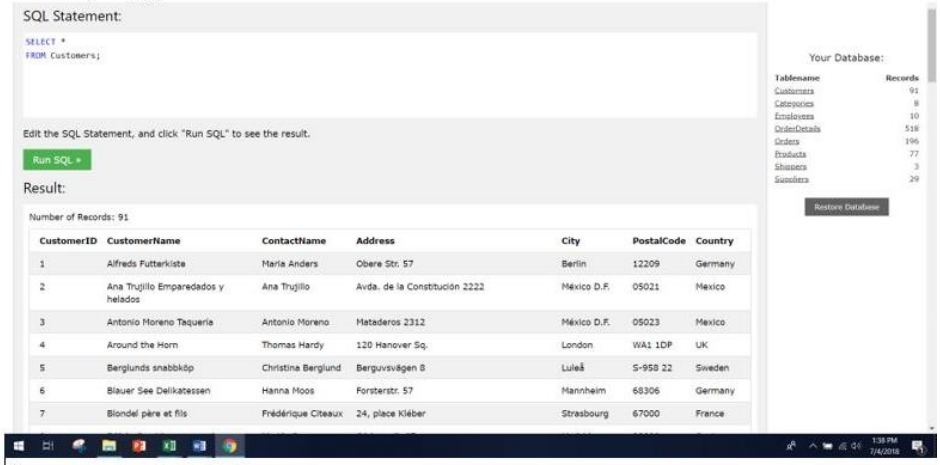
SQL·–·Lab·-·Answer·Template¶

1.→ ¶

Query¤	Number·of·records·returned¤
SELECT·*¶ FROM·Customers;¤	91¤

¶

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



2.→ ¶

¶

Query¤	Number·of·records·returned¤
¤	¤

¶

Insert·Screenshot·of·Output·Here:·[Note·that·you·only·need·to·display·the·first·screen·of·output·

Figure 4