Sparta Global SQL Report

By Emmanuel Obafemi Buraimo

The purpose of this report is to demonstrate my current proficiency with the Structured Query Language, commonly known as SQL.

This report will display the queries I created to solve the questions asked, screenshots of the queries along with a brief explanation of what it executes.

Exercise 1 Questions

1. Write a query that lists all Customers in either Paris or London. Include Customer ID, Company Name and all address fields.
2. List all products stored in bottles.
3. Repeat question above, but add in the Supplier Name and Country.
4. Write an SQL Statement that shows how many products there are in each category. Include Category Name in result set and list the highest number first.
5. List all UK employees using concatenation to join their title of courtesy, first name and last name together. Also include their city of residence.
6. List Sales Totals for all Sales Regions (via the Territories table using 4 joins) with a Sales Total greater than 1,000,000. Use rounding or FORMAT to present the numbers.
7. Count how many Orders have a Freight amount greater than 100.00 and either USA or UK as Ship Country.
8. Write an SQL Statement to identify the Order Number of the Order with the highest amount(value) of discount applied to that order.MAX

Solutions section 1

1. Write a query that lists all Customers in either Paris or London. Include Customer ID, Company Name and all address fields.
2. **SELECT** c.CustomerID , c.CompanyName , c.Address , c.PostalCode , c.City , c.Country
4. **FROM** Customers c
6. **WHERE** c.City IN ('Paris' , 'London')

This straightforward query lists out all the customers from the customers table who have their city as either in Paris or in London. The query also lists their customer ID, their company name, their address, their postal code, their city and their country. The resulting table can be viewed below.

Table

Description automatically generated

1. List all products stored in bottles.
2. **SELECT** \*
4. **FROM** Products p
6. **WHERE** p.QuantityPerUnit LIKE '%Bottle%'

The LIKE operator in SQL searches for patterns in columns. When text is placed between the percentage sign (%) it searches for any values that have that text in it, in this case it navigates throughout the QuantityPerUnit column and displays all the information of any product that is a bottle. The resulting table can be viewed below.

Table

Description automatically generated

1. Repeat question above but add in the Supplier Name and Country.
2. **SELECT** p.ProductName , s.CompanyName , s.Country
4. **FROM** Products p
6. **INNER** JOIN Suppliers s **ON** p.SupplierID = s.SupplierID
8. **WHERE** p.QuantityPerUnit LIKE '%Bottle%'

This query requires additional functionality of an inner join as we need information from two tables with a unique column as a similarity, this resulting query will show the product name along with the bottle’s supplier’s company name and their country. The resulting table can be viewed below.

Table

Description automatically generated

1. Write an SQL Statement that shows how many products there are in each category. Include Category Name in result set and list the highest number first.
2. **SELECT** c.CategoryName , COUNT(c.CategoryID) **AS** "Category Count"
4. **FROM** Products p
6. **INNER** JOIN Categories c **ON** p.CategoryID = c.CategoryID
8. **GROUP** **BY** c.CategoryName , c.CategoryID
10. **ORDER** **BY** COUNT(c.CategoryID) **DESC**

This query makes use of the inner joins to link the product table with the category table, the GROUP BY clause which groups repeating items in a column together in other to remove repeated values and also the ORDER BY clause which can sort based on the specified command in ascending or descending order.

The resulting table can be viewed below.

Table

Description automatically generated

1. List all UK employees using concatenation to join their title of courtesy, first name and last name together. Also include their city of residence.
2. **SELECT** CONCAT(e.TitleOfCourtesy, + ' ' + e.FirstName + ' ' + e.LastName) **AS** "Employee Description"
3. , e.City

6. **FROM** Employees e
8. **WHERE** e.Country IN ('UK')

This query makes use of the IN clause which specifies the columns that should be displayed, and also concatenates the names together and uses an aliases on them. The resulting table can be viewed below.

Table

Description automatically generated

1. List Sales Totals for all Sales Regions (via the Territories table using 4 joins) with a Sales Total greater than 1,000,000. Use rounding or FORMAT to present the numbers.
2. **SELECT** rr.RegionID, rr.RegionDescription **AS** Region,
4. FORMAT(SUM((UnitPrice \* Quantity) \* (1-Discount)),'C')
6. **AS** "Regional Sales"
7. **FROM** Orders **AS** o
8. **INNER** JOIN [**Order** Details] **AS** od **ON** od.OrderID = o.OrderID
9. **INNER** JOIN EmployeeTerritories **AS** et **ON** o.EmployeeID = et.EmployeeID
10. **INNER** JOIN Territories **AS** t **ON** et.TerritoryID = t.TerritoryID
11. **INNER** JOIN Region **AS** rr **ON** t.RegionID = rr.RegionID
12. **GROUP** **BY** rr.RegionDescription, rr.RegionID
13. **HAVING** SUM((UnitPrice \* Quantity) \* (1-Discount)) > 1000000
14. **ORDER** **BY** "Regional Sales" **DESC**;

This query uses multiple inner joins as the required questions needs data from multiple tables, joining them together with their foreign keys. It also makes use of the HAVING clause which is used in the place of the WHERE clause as it wouldn’t work with aggregate operations. The resulting table can be viewed below. The southern row is no included as their final total sales came in as $811,251.37

Table

Description automatically generated

1. Count how many Orders have a Freight amount greater than 100.00 and either USA or UK as Ship Country.
2. **SELECT** COUNT(oo.OrderID) **AS** "Order count"
4. **FROM** Orders oo
6. **WHERE** oo.Freight > 100 AND oo.ShipCountry IN ('USA' , 'UK')

This query counts the total amount of orders with a freight over 100 that has it’s shipping country located either in the USA or in the UK. The resulting table can be viewed below

Table

Description automatically generated

1. Write an SQL Statement to identify the Order Number of the Order with the highest amount(value) of discount applied to that order.MAX
2. **SELECT** oo.Orderid **AS** "Order ID's",
4. FORMAT((oo.UnitPrice \* oo.Quantity) \* oo.Discount,'C') **AS** "Total Discount added"
6. **FROM** [**Order** Details] oo
8. **WHERE** (oo.UnitPrice \* oo.Quantity \* oo.Discount) =
10. (**SELECT** **MAX**((oo.UnitPrice \* oo.Quantity) \* oo.Discount)
12. **FROM** [**Order** Details] oo)

The final query from Section 1 uses a sub-query to find the highest amount of discount added on orders from the order details table. The resulting table can be viewed below.

Table

Description automatically generated

Exercise 2 Questions

2.1 Write the correct SQL statement to create the following table:

Spartans Table – include details about all the Spartans on this course. Separate Title, First Name and Last Name into separate columns, and include University attended, course taken and mark achieved. Add any other columns you feel would be appropriate.

IMPORTANT NOTE: For data protection reasons do NOT include date of birth in this exercise.

2.2 Write SQL statements to add the details of the Spartans in your course to the table you have created.

Solutions section 2

1. **CREATE** **DATABASE** emmanuel\_db
3. USE emmanuel\_db
5. **DROP** **TABLE** IF EXISTS Spartans
7. **CREATE** **TABLE** Spartans(
8. s\_id  **int** IDENTITY(1,1) NOT NULL **PRIMARY** **KEY** ,
9. Title  **VARCHAR**(6),
10. First\_Name **VARCHAR**(25),
11. Last\_Name **VARCHAR** (25),
12. University **VARCHAR** (70),
13. Course\_Taken **VARCHAR**(50),
14. Mark\_attained **VARCHAR** (60)

INSERT INTO Spartans VALUES

('Mr', 'Svilen', 'Petrov', 'London Metropolitan University', 'BSc Computing', 'First'),

('Ms', 'Janja', 'Kovacevic', 'University of Massachusetts Amherst', 'Computer Science' , '3.9'),

('Mr','Shahid','Enayat','Brunel University', 'Electronic and Electrical Engineering','2:2'),

( 'Mr', 'Dami', 'Oshidele', 'Kings College London', 'Electronic Engineering with Management' , '2:1'),

( 'Mr', 'Chris', 'Cunnigham', 'Loughborough University', 'Computer Science' , '2:1'),

( 'Mr', 'Ben', 'Swift', 'Nottingham Trent University', 'Computer Science' , '2:1'),

( 'Mr', 'Toyin', 'Ajani', 'University of Bath', 'Chemical Engineering' , 'First'),

( 'Mr', 'Saleh', 'Sandhu', 'University of Westminster', 'Computer Science' , '2:1'),

( 'Mr', 'Reece', 'Louch', 'University of Warwick', 'Computer Science' , '2:2');

SELECT \* FROM Spartans

ORDER BY First\_Name ASC

The solution to the questions above involve the creation of a new database to store all the values, a statement which deletes the table created if it already exists as SQL doesn’t allow changing the variable parameters once created, and then inserting the values into table. When all the information about the Spartan’s has been added, all of it is printed out and the table is sorted in an ascending order based on the Spartan’s first name.

This is the output of the above query.

Table

Description automatically generated

Exercise 3 Questions

Write SQL statements to extract the data required for the following charts (create these in Excel):

3.1 List all Employees from the Employees table and who they report to. No Excel required.

3.2 List all Suppliers with total sales over $10,000 in the Order Details table. Include the Company Name from the Suppliers Table and present as a bar chart as below:

3.3 List the Top 10 Customers YTD for the latest year in the Orders file. Based on total value of orders shipped. No Excel required.

3.4 Plot the Average Ship Time by month for all data in the Orders Table using a line chart as below.

Solution sections 3

3.1 List all Employees from the Employees table and who they report to. No Excel required.

1. **SELECT** CONCAT (ee.FirstName , + ' '  + ee.LastName) **AS** " Employee " , ee.ReportsTo , CONCAT (eee.FirstName + ' ' , + eee.LastName) **AS** " Reports to "
3. **FROM** Employees ee
5. **INNER** JOIN Employees eee **ON** ee.ReportsTo = eee.EmployeeID
7. **ORDER** **BY** ee.ReportsTo

This query makes use of a self join in order to get the name of the employee that another employee reports too along with the ID. The table is also sorted in order of their employee ID. The resulting table can be viewed below.

Table

Description automatically generated

3.2 List all Suppliers with total sales over $10,000 in the Order Details table. Include the Company Name from the Suppliers Table and present as a bar chart as below:

1. **SELECT** ss.CompanyName , FORMAT (SUM( od.UnitPrice \* od.Quantity \* (1 - od.Discount)) , 'C') **AS** " Total sum of sales "
3. **FROM** [**Order** Details] od
5. **INNER** JOIN Products pp **ON** od.ProductID=pp.ProductID
6. **INNER** JOIN Suppliers ss **ON** pp.SupplierID=ss.SupplierID
7. **GROUP** **BY** ss.CompanyName
8. **HAVING** SUM( od.UnitPrice \* od.Quantity \* (1 - od.Discount))  > 10000
9. **ORDER** **BY** SUM( od.UnitPrice \* od.Quantity \* (1 - od.Discount))  **DESC**

This query is slightly more complex than the previous ones as it makes use of two aggregate functions along with 2 inner joins and also the HAVING clause as the WHERE clause can’t be used with aggregate functions, the subsequent result is formatted into currency and sorted by their respective amounts. The resulting table can be viewed below.

Table

Description automatically generated

3.3 List the Top 10 Customers YTD for the latest year in the Orders file. Based on total value of orders shipped. No Excel required.

1. **SELECT** **TOP** 10 cc.CustomerID **AS** "Customer ID", cc.CompanyName **As** "Company Name",
3. FORMAT(SUM(od.UnitPrice \* od.Quantity \* (1-od.Discount)),'C')
5. **AS** "Year to date sales"
7. **FROM** Customers cc
9. **INNER** JOIN Orders o **ON** o.CustomerID=cc.CustomerID
11. **INNER** JOIN [**Order** Details] od **ON** od.OrderID=o.OrderID
13. **WHERE** YEAR(OrderDate)=(**SELECT** **MAX**(YEAR(OrderDate)) **From** Orders)
15. AND o.ShippedDate **IS** NOT NULL
17. **GROUP** **BY** cc.CustomerID, cc.CompanyName
19. **ORDER** **BY** SUM(od.UnitPrice \* od.Quantity \* (1-od.Discount)) **DESC**;

The penultimate query makes use of Top to get top 10 items in the table after connecting them with two inner joins to get the order details information along with information from the orders table, it then groups by the company’s name and the customer’s ID. The resulting table can be viewed below.

Table

Description automatically generated

3.4 Plot the Average Ship Time by month for all data in the Orders Table using a line chart as below.

1. **SELECT** MONTH(oo.OrderDate) Month, YEAR(oo.OrderDate) Year, AVG(CAST(DATEDIFF(d, oo.OrderDate, oo.ShippedDate) **As** **DECIMAL**(10,2))) **AS** "Shipping Time"
3. **FROM** orders oo
5. **WHERE** oo.ShippedDate **IS** NOT NULL
7. **GROUP** **BY** YEAR(oo.OrderDate),MONTH(oo.OrderDate)
9. **ORDER** **BY** Year , Month

The final query displays the months of the order dates based on their average ship time, the resulting table would then be used to plot a chart. Below are the results. The chart has been plotted already.

Chart, line chart

Description automatically generated

Table

Description automatically generated