SQL Mini-Project I

By Golam Choudhury

Exercise 1

Task 1.1

Write a query that lists all customers in either Paris or London. Include customer ID, company name, and all address fields.

```
/* First selecting the relevant information within the SELECT statement.
Filtering which cities the information is related to using a WHERE statement */
SELECT
    c.CustomerID,
    c.CompanyName,
    c.ContactName,
    c.Address,
    c.City,
    c.Region,
    c.PostalCode,
    c.Country
FROM Customers c
WHERE c.City IN ('London','Paris')
;
```

Task 1.2

List all products stored in bottles.

```
/* To find which products are contained in bottles, filter the Quantity per Unit
column to contain 'bottle' */
SELECT p.ProductName
FROM Products p
WHERE p.QuantityPerUnit LIKE '%bottle%'
;
```

Task 1.3

Repeat question above, but add in the Supplier Name and Country.

```
/* Company Name and Country are from the Suppliers table, therefore we must
attatch these results using a join */
SELECT
    p.ProductID,
    s.CompanyName,
    s.Country
FROM Products p
INNER JOIN Suppliers s ON p.SupplierID = s.SupplierID
;
```

Task 1.4

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.

```
/* We ned to know how many products are in each category, therefore we select
to view the Category Name and the Count of the total rows, The GROUP BY statement
ensures that the counts are split into various categories */
SELECT
    c.CategoryName,
    COUNT(*) AS "Number of Products"
FROM Products p
/* Again since we're querying data from two different tables within the DB, we
need to join the Products and Categories tables */
INNER JOIN Categories c ON c.CategoryID = p.CategoryID
GROUP BY c.CategoryName
ORDER BY "Number of Products" DESC;
```

Task 1.5

List all UK employees using concatenation to join their title of courtesy, first name and last name together. Also include their city of residence.

```
/* Here the concatenation function CONCAT is utilised in order to present the
Employee name as one column, aliased as "Name" */
SELECT CONCAT(e.TitleofCourtesy,e.FirstName,' ',e.LastName) AS "Name",
    e.City AS "City of resisdence"
FROM Employees e
;
```

Task 1.6

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.

```
SELECT
    -- The total price considers the discount
    ROUND(SUM(od.Quantity*od.UnitPrice*(1 - od.Discount)), 2) AS "Total",
    t.RegionID
/* Using the ERD diagram, I mapped out the path of traversal using 5 Join statements
in order to group total order values by the regions */
FROM Products p
INNER JOIN [Order Details] od ON p.ProductID = od.ProductID
INNER JOIN Orders o ON od.OrderID = o.OrderID
INNER JOIN Employees e ON o.EmployeeID = e.EmployeeID
INNER JOIN EmployeeTerritories et ON e.EmployeeID = et.EmployeeID
INNER JOIN Territories t ON et.TerritoryID = t.TerritoryID
GROUP BY t.RegionID
/* This HAVING statement ensures only regions which have a total order value of
£10000+ are considered*/
HAVING SUM(od.Quantity*od.UnitPrice*(1 - od.Discount)) >= 1000000
ORDER BY SUM(od.Quantity*od.UnitPrice*(1 - od.Discount)) DESC
;
```

Task 1.7

Count how many Orders have a Freight amount greater than 100.00 and either USA or UK as Ship Country.

```
SELECT
    o.ShipCountry,
    -- Count alised to give the returned data context
    COUNT(*) AS "Freight amounts greater than 100"
FROM Orders o
-- Filter results to only include UK or USA
WHERE o.ShipCountry IN ('UK', 'USA')
    AND o.Freight > 100
GROUP BY o.ShipCountry
;
```

Task 1.8

Write an SQL Statement to identify the Order Number of the Order with the highest amount(value) of discount applied to that order.

```
/* We only need the top result, so we use TOP 1*/
SELECT TOP 1
    od.OrderID,
    -- Caclculating the discount & column aliasing
    SUM(od.Discount*od.UnitPrice*od.Quantity) AS "Total Discount"
FROM [Order Details] od
GROUP BY od.orderID
-- Making sure the first result holds the highest value
ORDER BY "Total Discount" DESC
;
```

Exercise 2

Initialising the Environment

```
-- Creating a new database
CREATE DATABASE golam_db;

-- Switiching the desired database
USE golam_db;
```

Task 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.

```
-- Creating an empty table named 'spartan_table'
CREATE TABLE spartan_table (
    -- defining the spartan ID as a primary key
    /* IDENTITY(1,1) means that the values start at one and
     auto-increments by 1 with each new entry */
    spartan_id INT IDENTITY(1,1) PRIMARY KEY,
    title VARCHAR(5),
    first_name VARCHAR(10),
    last_name VARCHAR(15),
    university_attended VARCHAR(50),
    course_taken VARCHAR(50),
    mark_achieved CHAR(3),
);
```

Task 2.2

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

```
-- Example of adding an entry into the table
/* All column names declared with the exception of the Spartan ID
since this is a primary key which auto-increments, therefore it is not required to
define when adding new rows*/
INSERT INTO spartan_table (
    title,
    first_name,
    last_name,
    university_attended,
    course_taken,
    mark_achieved
) VALUES (
    'Mr.',
    'Golam',
    'Choudhury',
    'City University of London',
    'BEng Aeronautical Engineering',
    '2:1'
);
```

Exercise 3

Task 3.1

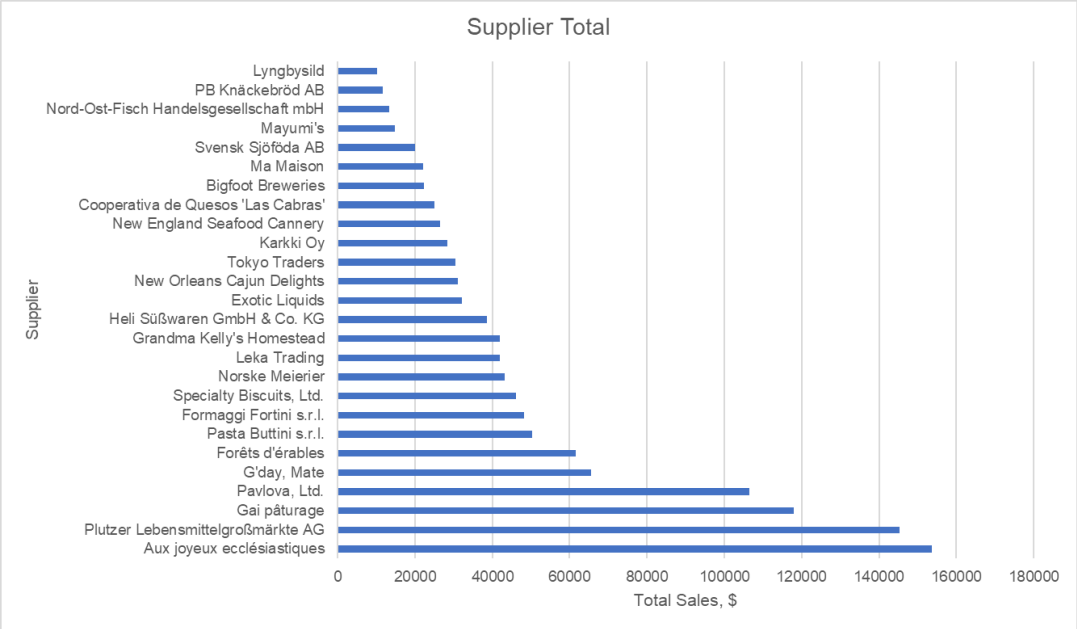
List all Employees from the Employees table and who they report to. No Excel required. Please mention the Employee Names and the ReportTo names.

```
-- This requires a table to join itself
SELECT
    -- Concatenation of the employee names
    CONCAT(e.FirstName,' ',e.LastName) AS "Employee Name",
    CONCAT(emp.FirstName,' ',emp.LastName) AS "Reports To"
FROM Employees e
/* Left join used to see all the employees and the employees they
report to by matching the Report To number in the left table with
the Employee ID in the right table*/
LEFT JOIN Employees emp ON e.ReportsTo = emp.EmployeeID
;
```

Task 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.

```
SELECT
    -- Selecting supplier name
    s.CompanyName AS "Supplier",
    -- Selecting a rounded value of the TOTAL order value
    ROUND(SUM(od.Quantity*od.UnitPrice*(1 - od.Discount)), 2) AS "Total Sales, $"
FROM [Order Details] od
INNER JOIN Products p ON od.ProductID = p.ProductID
INNER JOIN Suppliers s ON p.SupplierID = s.SupplierID
-- Categorising by each company/supplier
GROUP BY s.CompanyName
-- Filtering results to only consider values above 10000
HAVING SUM(od.Quantity*od.UnitPrice*(1 - od.Discount)) > 10000
-- Sorting from highest to lowest
ORDER BY SUM(od.Quantity*od.UnitPrice*(1 - od.Discount)) DESC
;
```



Task 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.

```
-- We only require the top ten results, hence, TOP 10
SELECT TOP 10
    c.CompanyName,
    -- I personally interpreted value as product/order worth without discount
    SUM(od.UnitPrice * od.Quantity) AS "Total value of orders"
FROM Orders o
    -- Traversing from Orders to Customers table to obtain the Customer Name
    -- By matching the Order IDs from [Order Deatils] with Order IDs in Orders
    -- and matching the Customer IDs from Orders with Customer IDs in Customers
    INNER JOIN [Order Details] od ON o.OrderID = od.OrderID
    INNER JOIN Customers c ON o.CustomerID = c.CustomerID
    -- Using a subquery within a WHERE statement
    -- The WHERE ensures the year of the data matches that obtained from the subquery
    WHERE FORMAT(o.OrderDate, 'yyyy') LIKE (
        -- Obtains the latest year from the result set
        SELECT TOP 1 FORMAT(ord.OrderDate, 'yyyy') as "Years"
        FROM Orders ord
        -- This ORDER BY makes sure latest year is at the top
        ORDER BY "Years" DESC
    )
    AND o.ShippedDate IS NOT NULL
GROUP BY c.CompanyName
-- This ORDER BY ensures the results go from highest to lowest value
ORDER BY "Total value of Orders" DESC
;
```

Task 3.4

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

```
SELECT
    -- Selecting the average ship time
    -- Calculated as the difference in days between shipped date and order date
    AVG(DATEDIFF(d,o.OrderDate,o.ShippedDate)) AS "Average ship time",
    -- Selecting the order date in a custom format
    FORMAT(o.OrderDate, 'MMM-yy') AS "Month"
FROM Orders o
-- GROUP BY o.OrderID
GROUP BY FORMAT(o.OrderDate, 'MMM-yy')
-- Ordering by the earliest to the latest date
ORDER BY MAX(FORMAT(o.OrderDate, 'yyyy-MM'))
;
```