Exercise 1 – Northwind Queries (40 marks: 5 for each question)

1. Write a query that lists all Customers in either Paris or London. Include Customer ID, Company Name and all address fields.

```
SELECT c.CustomerID, c.CompanyName, c.Address, c.City, c.Region, c.Country
FROM Customers c
WHERE c.City IN ('London', 'Paris');  --
Using IN allows for a range of values rather than chaining ORs
```

2. List all products stored in bottles.

```
SELECT *
FROM Products p
WHERE p.QuantityPerUnit
LIKE '%bottle%'; -
- % wildcard allows any number of characters on either side to catch all mentions of bottle
```

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

```
SELECT p.ProductID, p.ProductName, s.CompanyName, s.Country
FROM Products p
INNER JOIN Suppliers s ON p.SupplierID = s.SupplierID --Joins the products and suppliers
WHERE p.QuantityPerUnit LIKE '%bottle%';
```

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.

```
-SELECT c.CategoryName, COUNT(c.CategoryID) AS "Number of products" --
COUNT counts the records for each category name

FROM Products p LEFT JOIN Categories c ON c.CategoryID = p.CategoryID --
Joins all the categories to product IDs, even the categories with no products
GROUP BY c.CategoryName
ORDER BY "Number of products" DESC;
```

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.

```
SELECT CONCAT(e.TitleOfCourtesy,' ',e.FirstName,' ',e.LastName) AS "Title and full name", e
.City  --CONCAT is more concise for joining multiple items
FROM Employees e
WHERE e.Country = 'UK';
```

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. SELECT t.RegionID, ROUND(SUM(od.UnitPrice*od.Quantity*(1-
   od.Discount)),2) AS "Total sales" --
   Rounds to 2dp as currency, and calculates price with discount given that discount i
   s a decimal
8. FROM Territories t
```

```
INNER JOIN EmployeeTerritories et ON et.TerritoryID = t.TerritoryID
INNER JOIN Employees e ON et.EmployeeID = e.EmployeeID
INNER JOIN Orders o ON e.EmployeeID = o.EmployeeID
INNER JOIN [Order Details] od ON o.OrderID = od.OrderID  --
Long join between all tables
GROUP BY t.RegionID
HAVING ROUND(SUM(od.UnitPrice*od.Quantity*(1-od.Discount)),2) >= 1000000;  --
Having clause as where cannot be used after a group by
```

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

```
SELECT COUNT(o.OrderID) AS "Number of orders"  -
-  this only asks for a count so just produces 1 record with 1 column
FROM Orders o
WHERE o.Freight > 100 AND o.ShipCountry IN ('USA','UK'); --
Uses AND to satisfy both conditions
```

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

```
SELECT TOP 1 od.OrderID, ROUND(SUM(od.Discount*od.UnitPrice*od.Quantity),2) AS "Actual Disc
ount"  -- Handles discount being given as decimal and calculates actual discount
FROM [Order Details] od
GROUP BY OrderID
ORDER BY "Actual Discount" DESC; --
Alias given earlier can be used due to ORDER BY coming after SELECT in the processing order
```

Exercise 2 – Create Spartans Table (20 marks – 10 each)

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.

```
CREATE TABLE spartan_table (
    spartan_id int IDENTITY(1,1) PRIMARY KEY, -
- IDENTITY(1,1) sets an auto increment of 1 starting from 1, used for generating the primar
y key
    title VARCHAR(5), --Allows up to 5 characters
    first_name VARCHAR(20),
    last_name VARCHAR(10),
    university VARCHAR(50),
    course_taken VARCHAR(20),
    mark CHAR(3) -- Allows strictly 3 characters, and fills blanks with whitespace
)
```

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

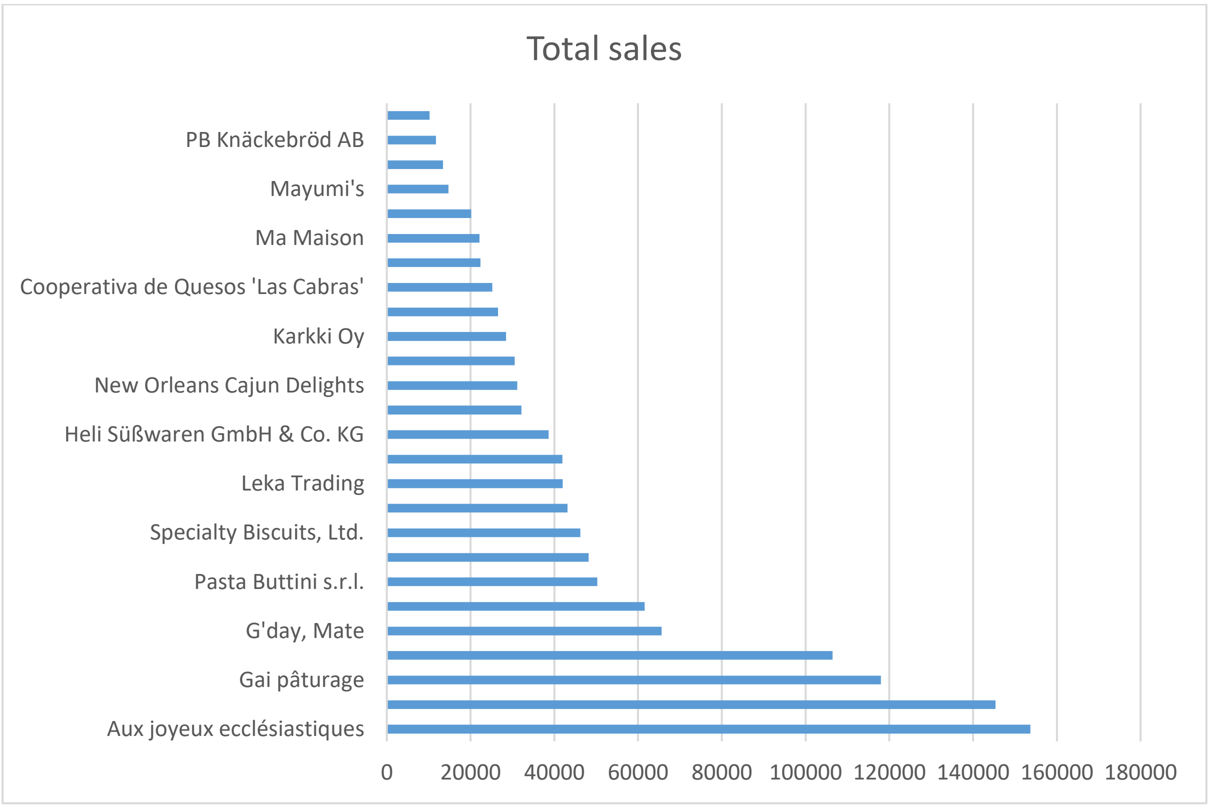
```
INSERT INTO spartan_table (  --
specifying the columns isn't necessary if they're in order, but can make the statement more
 readable
    title,
    first_name,
    last_name,
    university,
    course_taken,
    mark
) VALUES (
    'Mr.',
    'Golam',
    'Choudhury',
    'City University of London',
    'BEng Aeronautical Engineering',
    '2:1'
),  (  --
different entries can be separated by a , to make adding multiple entries at once more conc
ise
    'Mr.',
    'Jakub',
    'Matyjewicz',
    'Poznan University of Technology',
    'Technical Physics',
    NULL
), (
    'Mr.',
    'Alasdair',
    'Malcolm',
    'University of Exeter',
    'Electronic Engineering',
    '2:2'
),(
    'Mr.',
    'Matthew',
    'Holmes',
    'University of Bath',
    'Computer Science and Mathematics',
    '2:2'
);
```

3.1 List all Employees from the Employees table and who they report to. No Excel required. Please mention the Employee Names and the Report To names. (5 Marks)

```
SELECT CONCAT(e.TitleOfCourtesy,' ',e.FirstName,' ',e.LastName) AS "Full name", CONCAT(m.Ti
tleOfCourtesy,' ',m.FirstName,' ',m.LastName) AS "Manager"  --
Uses 2 different aliases for the self table for the self lookup
FROM Employees e LEFT JOIN Employees m ON e.ReportsTo = m.EmployeeID;  --
LEFT JOIN ensures employees who do not report to anyone are not omitted
```

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: (5 Marks)

```
SELECT s.CompanyName, SUM((1-od.Discount)*od.UnitPrice*od.Quantity) 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
GROUP BY s.CompanyName
HAVING SUM(od.Quantity * od.UnitPrice *(1 - od.Discount)) > 10000
ORDER BY "Total Sales" DESC;
```

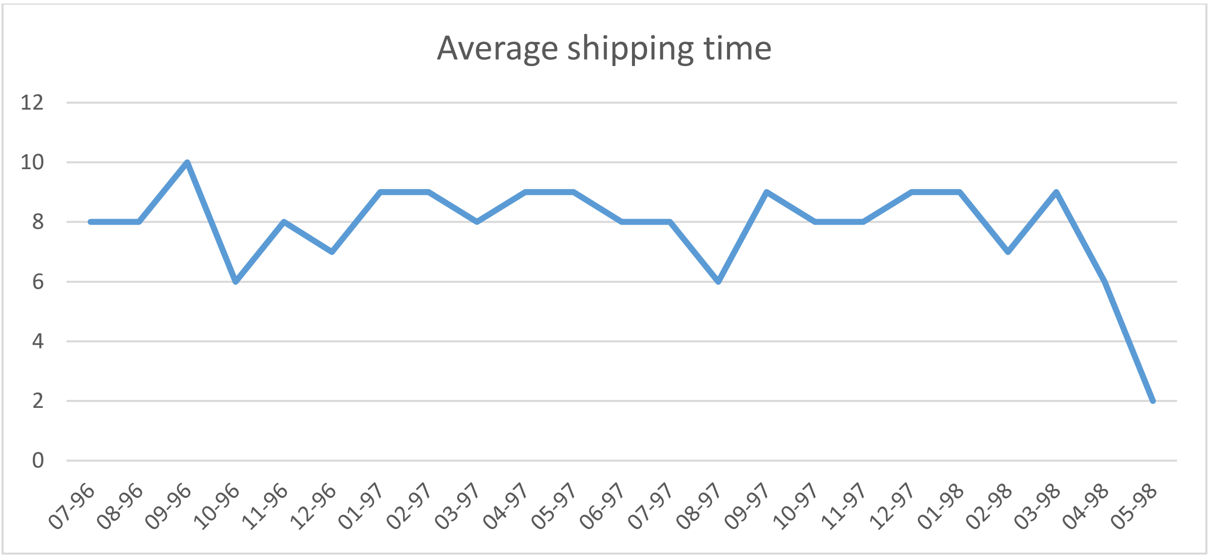


3.3 List the Top 10 Customers YTD for the latest year in the Orders file. Based on total value of orders shipped. No Excel required. (10 Marks)

```
SELECT TOP 10 c.CompanyName AS 'Company name', ROUND(SUM(od.UnitPrice * od.Quantity
*(1 - od.Discount)),2) AS "Total value of orders"   --
FROM Customers c
INNER JOIN Orders o ON c.CustomerID = o.CustomerID
INNER JOIN [Order Details] od on o.OrderID = od.OrderID
WHERE FORMAT(o.OrderDate, 'yyyy') = (                                  --
I tried using YEAR() but this doesn't compare properly - I guess this returns varchar?
        SELECT MAX(FORMAT(ord.OrderDate, 'yyyy'))           --
subquery pulls the most recent year to use in the main query
        FROM Orders ord
    )
    AND o.ShippedDate IS NOT NULL    -- prevents unshipped orders from being included
GROUP BY c.CompanyName
ORDER BY "Total value of Orders" DESC;
```

3.4 Plot the Average Ship Time by month for all data in the Orders Table using a line chart as below. (10 Marks)

```
SELECT  MAX(FORMAT(o.OrderDate, 'MM-
yy')) AS "Month",  AVG(DATEDIFF(day,o.OrderDate,o.ShippedDate)) AS "Average shipping time"
--Gets the month and average ship time by calculating the date difference in days
FROM Orders o
WHERE o.ShippedDate IS NOT NULL -
- prevents unshipped orders from being included and breaking the maths
GROUP BY FORMAT(o.OrderDate, 'yyyy-MM')  --yyyy-
MM is a more functional way od displaying the date, MM-yy is more human readable
ORDER BY MAX(FORMAT(o.OrderDate, 'yyyy-MM'));
```


Average shipping time