SQL Mini Project

Exercise 1.1

Task: To write a query that lists all Customers in either Paris or London. Include Customer ID, Company Name and all address fields

For this task I choose to select only the specific columns asked for in the task rather than displaying all, as although it only removes a small number of columns it makes the resultant table more presentable

Code:

```
SELECT
  CustomerID,
  CompanyName,
  Address,
  City,
  PostalCode,
  Country
FROM
  Customers
WHERE
  City IN('london', 'paris')
```

CustomerID	CompanyName	Address	City	PostalCode	Country
AROUT	Around the Horn	120 Hanover Sq.	London	WA1 1DP	UK
BSBEV	B's Beverages	Fauntleroy Circus	London	EC2 5NT	UK
CONSH	Consolidated Holdings	Berkeley Gardens 12 Brewery	London	WX1 6LT	UK
EASTC	Eastern Connection	35 King George	London	WX3 6FW	UK
NORTS	North/South	South House 300 Queensbridge	London	SW7 1RZ	UK
PARIS	Paris spécialités	265, boulevard Charonne	Paris	75012	France
SEVES	Seven Seas Imports	90 Wadhurst Rd.	London	OX15 4NB	UK
SPECD	Spécialités du monde	25, rue Lauriston	Paris	75016	France

Exercise 1.2

Task: List all products stored in bottles.

For this task I choose to only show those columns that would convey which item it is and to confirm it is stored in a bottle

Code:

```
SELECT
  ProductID,
  ProductName,
  QuantityPerUnit
FROM
```

```
  Products
WHERE
  QuantityPerUnit LIKE '%bottle%'
```

```
    Products
WHERE
  QuantityPerUnit LIKE '%bottle%'
```

Output:

ProductID	ProductName	QuantityPerUnit
2	Chang	24 - 12 oz bottles
3	Aniseed Syrup	12 - 550 ml bottles
15	Genen Shouyu	24 - 250 ml bottles
34	Sasquatch Ale	24 - 12 oz bottles
35	Steeleye Stout	24 - 12 oz bottles
38	Côte de Blaye	12 - 75 cl bottles
39	Chartreuse verte	750 cc per bottle
61	Sirop d'érable	24 - 500 ml bottles
65	Louisiana Fiery Hot Pepper S…	32 - 8 oz bottles
67	Laughing Lumberjack Lager	24 - 12 oz bottles
70	Outback Lager	24 - 355 ml bottles
75	Rhönbräu Klosterbier	24 - 0.5 l bottles

Exercise 1.3

Task: Repeat 1.2, but add in the Supplier Name and Country.

For this task I took what I did in the previous task and used an inner join to connect the product table with the suppliers table, using the primary/foreign key relationship or the SuppliersID, in order to get all the columns

Code:

```
SELECT
 p.productID,
 p.ProductName,
 p.QuantityPerUnit,
 s.CompanyName,
 s.Country
FROM
  Products p
  INNER JOIN Suppliers s ON p.SupplierID = s.SupplierID
```

```
WHERE
  p.QuantityPerUnit like '%bottle%'
```

Output:

productID	ProductName	QuantityPerUnit	CompanyName	Country
2	Chang	24 - 12 oz bottles	Exotic Liquids	UK
3	Aniseed Syrup	12 - 550 ml bottles	Exotic Liquids	UK
15	Genen Shouyu	24 - 250 ml bottles	Mayumi's	Japan
34	Sasquatch Ale	24 - 12 oz bottles	Bigfoot Breweries	USA
35	Steeleye Stout	24 - 12 oz bottles	Bigfoot Breweries	USA
38	Côte de Blaye	12 - 75 cl bottles	Aux joyeux ecclésiastiques	France
39	Chartreuse verte	750 cc per bottle	Aux joyeux ecclésiastiques	France
61	Sirop d'érable	24 - 500 ml bottles	Forêts d'érables	Canada
65	Louisiana Fiery Hot Pepper S…	32 - 8 oz bottles	New Orleans Cajun Delights	USA
67	Laughing Lumberjack Lager	24 - 12 oz bottles	Bigfoot Breweries	USA
70	Outback Lager	24 - 355 ml bottles	Pavlova, Ltd.	Australia
75	Rhönbräu Klosterbier	24 - 0.5 l bottles	Plutzer Lebensmittelgroßmärk…	Germany

Exercise 1.4

Task: 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.
For this task I used the count keyword to count and inner join with the products table to show how many times the id for a given category appeared in the products table and the group by command so that every row with a category name that is the same is combined into one.

Code:

```
SELECT
  c.CategoryName,
  COUNT(c.CategoryID) AS "Number of products in each Category"
FROM
  Categories c
  INNER JOIN Products p ON c.CategoryID = p.CategoryID
GROUP BY
  c.CategoryName
ORDER BY
  2 DESC
```

Output:

	CategoryName	Number of products in each Category
1	Confections	13
2	Beverages	12
3	Condiments	12
4	Seafood	12
5	Dairy Products	10
6	Grains/Cereals	7
7	Meat/Poultry	6
8	Produce	5

Exercise 1.5

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

For this task I used CONCAT keyword to combine the strings in held in the columns for the different parts of the persons name as well as add a space in between the first and last names and the where command so that only those in the UK are shown.

Code:

```
SELECT
  CONCAT(
    e.TitleOfCourtesy, e.FirstName, ' ',
    e.LastName
  ) AS "Employee",
  e.City
FROM
  Employees e
WHERE
  e.Country = 'UK'
```

Output:

	Employee	City
1	Mr.Steven Buchanan	London
2	Mr.Michael Suyama	London
3	Mr.Robert King	London
4	Ms.Anne Dodsworth	London

Exercise 1.6

Task: 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.

The main decision I took in this task where to round the total sales column to the nearest 1000 to make it more readable without making too big an impact on the data. The other big decision I took was to use a subquery and a join to add a table with Total sales as a column instead of adding the table as normal and having the total sales be calculated at the top select and I did this just to make the top select as simple as possible and to cut down on the amount I would have to calculate total sales as I use it in the having clause to filter out those less than 1 million

Code:

```
SELECT
  r.RegionDescription,
  ROUND(
    SUM(Totalsales),
    -3
  ) AS "Total Sales"
FROM
  region r
  INNER JOIN Territories t ON r.RegionID = t.RegionID
  INNER JOIN EmployeeTerritories et ON t.TerritoryID = et.TerritoryI
D
  INNER JOIN Employees e ON et.EmployeeID = e.EmployeeID
  INNER JOIN Orders o ON e.EmployeeID = o.EmployeeID
  INNER JOIN (
    SELECT
      od.orderID,
      SUM(
        od.UnitPrice * od.Quantity *(1 - od.Discount)
      ) AS TotalSales
    FROM
      [order details] od
    GROUP BY
      orderID
  ) od ON o.orderid = od.orderID
GROUP BY
  RegionDescription
HAVING
  SUM(TotalSales) > 1000000
```

Output:

	RegionDescription	Total Sales
1	Northern	1049000
2	Eastern	2730000
3	Western	1615000

Exercise 1.7

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

Simple code to count the number of times the freight value is greater than 100 and ship country is either USA or UK

Code:

```
SELECT
  COUNT(o.Freight) AS "Number of orders with frieght over 100 and ei
ther USA or UK as ship country"
FROM
  Orders o
WHERE
  o.Freight > 100
  AND o.ShipCountry IN ('USA', 'UK')
```

Output:

	Number of orders with frieght over 100 and either USA or UK as ship country
1	49

Exercise 1.8

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

The code selects only the row at the top of the table and calculates the total value discounted from each order and orders the table so that the order with the highest value discounted at the top.

Code:

```
SELECT
  TOP 1 o.OrderID,
  SUM(
    od.Discount * od.UnitPrice * od.Quantity
```

```
  ) AS DiscountValue
FROM
  Orders o
  INNER JOIN [Order Details] od ON o.OrderID = od.OrderID
GROUP BY
  o.OrderID
ORDER BY
  DiscountValue DESC
```

Output:

	OrderID	DiscountValue
1	11030	3706.8499755859375

Exercise 2.1

Task: To make a 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.

For the table I took into the maximum reasonable length that each column could be and added some room for safety. I was basing the table on my class, at least in name, so I defaulted title as they are only male trainees in the class and defaulted mark to First to be nice.

Code:

```
CREATE TABLE Spartans
(
    Title VARCHAR(5) DEFAULT 'Mr.',
    firstName VARCHAR(15),
    lastName VARCHAR (15),
    University VARCHAR(30),
    Course VARCHAR(40),
    mark VARCHAR (15) DEFAULT 'First'
)
```

Exercise 2.2

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

Using the Insert to add in the field of the table, missing out the title and mark as they will default to the pre-set values

Code:

```
INSERT Spartans (
  firstName, lastName, University, Course
)
VALUES
  (
    'Adrian', 'Wong', 'Oxford University',
    'Bio-mechanical Engineering'
  ),
  (
    'Thomas', 'Canfield', 'Cambridge University',
    ' Software Development'
  ),
  (
    'Karim', 'Wholer', ' University of London',
    'Elecronics System Engineering'
  ),
  (
    'Alexander', 'Legon', 'Edinburgh University',
    'Software Development'
  ),
  (
    'Alex', 'Lynch', 'University of Yorkshire',
    'Digital systems Engineering'
  )
```

Output:

Title	firstName	lastName	University	Course	mark
Mr.	Adrian	Wong	Oxford University	Bio-mechanical Engineering	First
Mr.	Thomas	Canfield	Cambridge University	Software Development	First
Mr.	Karim	Wholer	University of London	Elecronics System Engineering	First
Mr.	Alexander	Legon	Edinburgh University	Software Development	First
Mr.	Alex	Lynch	University of Yorkshire	Digital systems Engineering	First

Exercise 3.1

Task: List all Employees from the Employees table and who they report to. Please mention the Employee Names and the ReportTo names.

I used a left join on this one so that even those that do not have someone in the ReportTo field will still be shown. Joined using the ID in ReportTo from the first table to the employeeID of the second table so that the name fields in the second table would be the names of the person that is reported to by the person of the first table.

Code:

```
SELECT
  CONCAT(e1.FirstName, ' ', e1.LastName) AS "Employee",
  CONCAT (e2.FirstName, ' ', e2.LastName) AS 'Reports to'
FROM
  Employees e1
  LEFT JOIN Employees e2 ON e1.ReportsTo = e2.EmployeeID
```

Output:

Employee	Reports to
Nancy Davolio	Andrew Fuller
Andrew Fuller	
Janet Leverling	Andrew Fuller
Margaret Peacock	Andrew Fuller
Steven Buchanan	Andrew Fuller
Michael Suyama	Steven Buchanan
Robert King	Steven Buchanan
Laura Callahan	Andrew Fuller
Anne Dodsworth	Steven Buchanan

Exercise 3.2

Task: 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.

Similar to a previous task I used a subquery to keep the top SELECT as simple as I could and used a round to round to the nearest 1000 to make the data more readable.

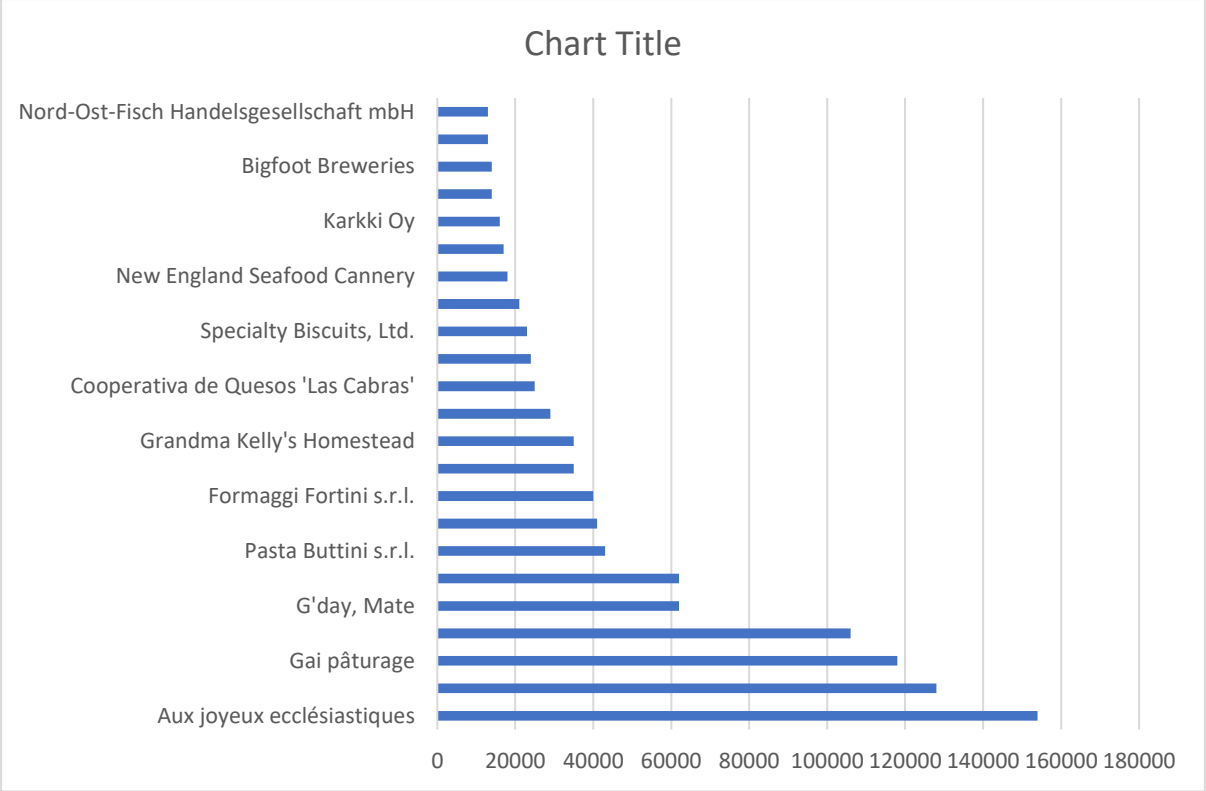
Code:

```
SELECT
  s.CompanyName,
  ROUND(
    SUM(TotalSales),
    -3
  ) AS "Total sales"
FROM
  Suppliers s
  INNER JOIN Products p ON s.SupplierID = p.SupplierID
  INNER JOIN (
    SELECT
      od.ProductID,
```

```
      SUM(
        od.UnitPrice * od.Quantity *(1 - od.Discount)
      ) AS TotalSales
    FROM
      [order details] od
    GROUP BY
      ProductID
  ) od1 ON p.ProductID = od1.productID
WHERE
  TotalSales >= 10000
GROUP BY
  CompanyName
ORDER BY
  SUM(TotalSales) DESC
```

Output:

CompanyName	Total sales
Aux joyeux ecclésiastiques	154000
Plutzer Lebensmittelgroßmärk…	128000
Gai pâturage	118000
Pavlova, Ltd.	106000
G'day, Mate	62000
Forêts d'érables	62000
Pasta Buttini s.r.l.	43000
Norske Meierier	41000
Formaggi Fortini s.r.l.	40000
Heli Süßwaren GmbH & Co. KG	35000
Grandma Kelly's Homestead	35000
Exotic Liquids	29000
Cooperativa de Quesos 'Las C…	25000
Leka Trading	24000



Exercise 3.3

Task: List the Top 10 Customers YTD for the latest year in the Orders file. Based on total value of orders shipped.

Used both top 10 and order by desc to show only the 10 largest values

Code:

```
SELECT
  TOP 10 c.CompanyName,
  ROUND(
    SUM(TotalSales),
    0
  ) AS "Total sales"
FROM
  Orders o
  INNER JOIN Customers c ON o.CustomerID = c.CustomerID
  INNER JOIN (
    (
      SELECT
        od.OrderID,
        SUM(
          od.UnitPrice * od.Quantity *(1 - od.Discount)
        ) AS TotalSales
```

```
      FROM
        [order details] od
      GROUP BY
        OrderID
    )
  ) od1 ON o.OrderID = od1.OrderID
WHERE
  YEAR(ShippedDate) = (
    SELECT
      MAX(
        YEAR(Shippeddate)
      )
    from
      orders
  )
GROUP BY
  CompanyName
ORDER BY
  ROUND(
    SUM(TotalSales),
    0
  ) DESC
```

Output:

	CompanyName	Total sales
1	QUICK-Stop	37949
2	Save-a-lot Markets	36310
3	Ernst Handel	33814
4	Hanari Carnes	23821
5	Königlich Essen	21136
6	Hungry Owl All-Night Grocers	20402
7	Rattlesnake Canyon Grocery	19983
8	White Clover Markets	15279
9	Folk och fä HB	13644
10	Suprêmes délices	11645

Exercise 3.4

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

In this task I used a CONCAT with MONTH and YEAR to separate the month and year of the dates separately to make ordering easier and used cast as float as avg would return only int and i needed more precise values.

Code:

```
SELECT
  AVG(
    CAST(
      DATEDIFF(dd, o.OrderDate, o.ShippedDate) AS FLOAT
    )
  ) AS "Average time to ship",
  CONCAT(
    MONTH(o.OrderDate),
    '/',
    YEAR(o.OrderDate)
  ) AS "Date Orderd"
FROM
  Orders o
GROUP BY
  MONTH(o.OrderDate),
  YEAR(o.OrderDate)
```

Output:

Average time to ship	Date Orderd
8.045454545454545	7/1996
8	8/1996
10.608695652173912	9/1996
6.5	10/1996
8.36	11/1996
7.516129032258065	12/1996
9.969696969696969	1/1997
9.310344827586206	2/1997
8.3	3/1997
9	4/1997
9.15625	5/1997
8.833333333333334	6/1997
8.696969696969697	7/1997
6.787878787878788	8/1997

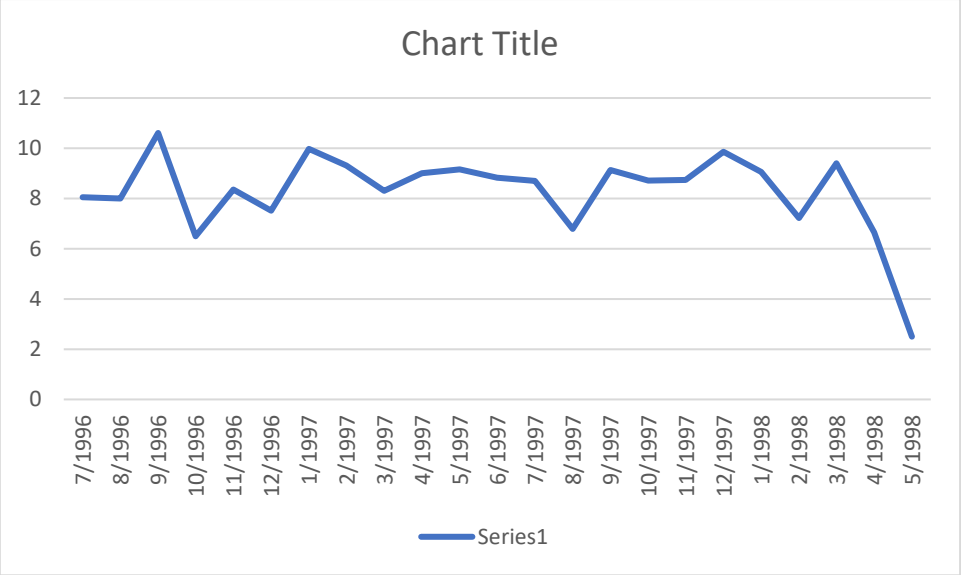


Chart Title
12
10
8
6
4
2
0
7/1996
8/1996
9/1996
10/1996
11/1996
12/1996
1/1997
2/1997
3/1997
4/1997
5/1997
6/1997
7/1997
8/1997
9/1997
10/1997
11/1997
12/1997
1/1998
2/1998
3/1998
4/1998
5/1998
Series1