SQL Project Report

Reece Louch

October 6, 2020

Contents

1 Exercise 1: Northwind Queries **2**
1.1 2
1.2 3
1.3 4
1.4 4
1.5 5
1.6 6
1.7 6
1.8 7

2 Exercise 2: Create Spartans Table **8**
2.1 8
2.2 9

3 Exercise 3: Northwind Data Analysis linked to Excel **11**
3.1 11
3.2 12
3.3 13
3.4 14

Chapter 1

Exercise 1: Northwind Queries

1.1

- Question:

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

- Answer:

```
SELECT c.CustomerID, c.ContactName,
CONCAT(c.Address,', ',c.City,', ',c.PostalCode, ', ', c.Country)
AS "Address"
--CONCAT concatenates the desired attributes together,
--as well as concatenating strings together.
FROM Customers c
--FROM dictates which table the data is coming from.
WHERE c.City IN ('Paris', 'London');
--IN here is used to list desired values for the WHERE clause.
--This could also be achieved with an OR operator.
```

- Results:

CustomerID	ContactName	Address
AROUT	Thomas Hardy	120 Hanover Sq., London, WA1 1DP, UK
BSBEV	Victoria Ashworth	Fauntleroy Circus, London, EC2 5NT, UK
CONSH	Elizabeth Brown	Berkeley Gardens 12 Brewery, London, WX1 6LT, UK
EASTC	Ann Devon	35 King George, London, WX3 6FW, UK
NORTS	Simon Crowther	South House 300 Queensbridge, London, SW7 1RZ, UK
PARIS	Marie Bertrand	265, boulevard Charonne, Paris, 75012, France
SEVES	Hari Kumar	90 Wadhurst Rd., London, OX15 4NB, UK
SPECD	Dominique Perrier	25, rue Lauriston, Paris, 75016, France

1.2

- Question:

  List all products stored in bottles.

- Answer:

```
SELECT p.ProductName
FROM Products p
--The table Products is aliased as 'p' here to
--allow for ease of use in the future.
WHERE p.QuantityPerUnit LIKE '%bottle%';
--The LIKE operator accepts wildcards to allow for searching
--the database for similar data. The wildcards can act
--as 1 or n characters each.
```

- Results:

ProductName
Chang
Aniseed Syrup
Genen Shouyu
Sasquatch Ale
Steeleye Stout
Côte de Blaye
Chartreuse verte
Sirop d'érable
Louisiana Fiery Hot Pepper Sauce
Laughing Lumberjack Lager
Outback Lager
Rhönbräu Klosterbier

1.3

- Question:

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

- Answer:

```
SELECT p.ProductName, s.CompanyName, s.Country
FROM Products p, Suppliers s
--Here a Cartesian Join or CROSS JOIN is used to join two tables
--together. This should be avoided, however when used with an
--appropriate WHERE clause, it acts like an INNER JOIN.
WHERE p.QuantityPerUnit LIKE '%bottle%' AND p.SupplierID = s.SupplierID;
--The p.SupplierID = s.SupplierID constraint changes the Cartesian
--Join to an INNER JOIN.
```

- Results:

ProductName	CompanyName	Country
Chang	Exotic Liquids	UK
Aniseed Syrup	Exotic Liquids	UK
Genen Shouyu	Mayumi's	Japan
Sasquatch Ale	Bigfoot Breweries	USA
Steeleye Stout	Bigfoot Breweries	USA
Côte de Blaye	Aux joyeux ecclésiastiques	France
Chartreuse verte	Aux joyeux ecclésiastiques	France
Sirop d'érable	Forêts d'érables	Canada
Louisiana Fiery Hot Pepper Sauce	New Orleans Cajun Delights	USA
Laughing Lumberjack Lager	Bigfoot Breweries	USA
Outback Lager	Pavlova, Ltd.	Australia
Rhönbräu Klosterbier	Plutzer Lebensmittelgroßmärkte AG	Germany

1.4

- Question:

  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.

- Answer:

```
SELECT c.CategoryName AS "Category Name",
COUNT(*) AS "Number of Products"
--Count is an aggregate function that allows for
--the number of rows returned to be selected.
```

```
FROM Products p INNER JOIN Categories c
--INNER JOIN joins two or more tables together on
--their common values. This type of join only returns
--the rows where there are values in both tables.
ON p.CategoryID = c.CategoryID
--ON specifies the joining attribute and the relation.
GROUP BY c.CategoryName
--GROUP BY allows for the elements to be grouped under
--an attribute for the Aggregate Functions. In this
--example, COUNT will instead get the number of rows
--with a particular CategoryName for each Category.
ORDER BY COUNT(*) DESC;
--ORDER BY enforces the results to be returned in a
--particular order dictated by the column passed to
--it, and either DESC or ASC depending on which way
--it needs to be ordered.
```

- Results:

Category Name	Number of Products
Confections	13
Beverages	12
Condiments	12
Seafood	12
Dairy Products	10
Grains/Cereals	7
Meat/Poultry	6
Produce	5

1.5

- Question:

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

- Answer:

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

- Results:

Name	City
Mr. Steven Buchanan	London
Mr. Michael Suyama	London
Mr. Robert King	London
Ms. Anne Dodsworth	London

1.6

- Question:

  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.

- Answer:

```
SELECT t.RegionID, ROUND(SUM(od.UnitPrice*od.Quantity*(1-od.Discount)),2)
FROM [Order Details] od INNER JOIN Orders o
--Square brackets around a table name allows for
--spaces to be utilised. Spaces should still be
--avoided in table names since it is a bad convension.
ON o.OrderID = od.OrderID INNER JOIN Employees e
ON e.EmployeeID = o.EmployeeID INNER JOIN EmployeeTerritories et
ON e.EmployeeID = et.EmployeeID INNER JOIN Territories t
ON et.TerritoryID = t.TerritoryID
GROUP BY t.RegionID
HAVING SUM(od.UnitPrice*od.Quantity*(1-od.Discount)) > 1000000;
--HAVING allows for Aggregate Functions to be used in comparisons.
```

- Results:

RegionID	(No column name)
3	1048605.58
1	2730198.01
2	1615248

1.7

- Question:

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

- Answer:

```
SELECT COUNT(*)
FROM Orders o
WHERE o.ShipCountry IN ('USA', 'UK') AND o.Freight > 100.00;
```

- Results:

(No column name)
49

1.8

- Question:

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

- Answer:

```
SELECT TOP 1 o.OrderID, SUM(o.UnitPrice*o.Quantity*(o.Discount))
FROM [Order Details] o
GROUP BY o.OrderID
ORDER BY SUM(o.UnitPrice*o.Quantity*(o.Discount)) DESC;
--Assumption, Highest amount(value) of discount taken
--to mean the greatest amount off, as opposed to the
--greatest percentage off.
```

- Results:

OrderID	(No column name)
11030	3706.849976

Chapter 2

Exercise 2: Create Spartans Table

2.1

- Question:

  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.

- Answer:

```
DROP TABLE IF EXISTS Spartans;
--Removes the table if it already exists.
CREATE TABLE Spartans (
--Makes a new table with the name Spartans
        id INT IDENTITY PRIMARY KEY,
        --Allows for the Primary Key to be auto-incremented
        --when new data is added.
        title VARCHAR(5),
        --VARCHAR is a variable length character. It changes
        --size depending on the input string with a max size
        --defined on creation. In this case, max size is 5
        first_name VARCHAR(20) NOT NULL,
        --NOT NULL prevents null values from being input to
```

```
        --this column.
        last_name VARCHAR(20) NOT NULL,
        university VARCHAR(50),
        course VARCHAR(50),
        mark VARCHAR(4)
);
```

- Results:

```
Commands completed successfully.
Total execution time: 00:00:00.020
```

2.2

- Question:

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

- Answer:

```
INSERT INTO Spartans VALUES (
--INSERT INTO takes in data to be placed into the
--database.
        'Mr', --title
        'Reece', --first_name
        'Louch', --last_name
        'University Of Warwick', --university
        'Computer Science', --course
        '2:2' --mark
), (
        --Everynew record is separated by a comma.
        --In here goes the next persons information
        --in the same format as above.
)
```

- Results:

```
(11 rows affected)
Total execution time: 00:00:00.019
```

id	title	first_name	last_name	university	course	mark
1	Mr	Reece	Louch	University Of Warwick	Computer Science	2:2
2	title	first_name	last_name	university	course	mark
3	title	first_name	last_name	university	course	mark
4	title	first_name	last_name	university	course	mark
5	title	first_name	last_name	university	course	mark
6	title	first_name	last_name	university	course	mark
7	title	first_name	last_name	university	course	mark
8	title	first_name	last_name	university	course	mark
9	title	first_name	last_name	university	course	mark
10	title	first_name	last_name	university	course	mark
11	title	first_name	last_name	university	course	mark

Chapter 3

Exercise 3: Northwind Data Analysis linked to Excel

3.1

- Question:

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

- Answer:

```
SELECT CONCAT(e.TitleOfCourtesy, ' ', e.FirstName, ' ', e.LastName)
AS "Name",
CONCAT(em.TitleOfCourtesy, ' ', em.FirstName, ' ', em.LastName)
AS "Reports to"
FROM Employees e LEFT JOIN Employees em
--LEFT JOIN returns all the rows in the left table,
--all with the matching rows of the right table.
ON e.ReportsTo = em.EmployeeID;
```

- Results:

Name	Reports to
Ms. Nancy Davolio	Dr. Andrew Fuller
Dr. Andrew Fuller	
Ms. Janet Leverling	Dr. Andrew Fuller
Mrs. Margaret Peacock	Dr. Andrew Fuller
Mr. Steven Buchanan	Dr. Andrew Fuller
Mr. Michael Suyama	Mr. Steven Buchanan
Mr. Robert King	Mr. Steven Buchanan
Ms. Laura Callahan	Dr. Andrew Fuller
Ms. Anne Dodsworth	Mr. Steven Buchanan

Note: There would be a `NULL` value in second column and row if the `CONCAT` function wasn't used. Instead, the `CONCAT` function returns an empty string.

3.2

- Question:

  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:

- Answer:

```
SELECT s.CompanyName, SUM(od.UnitPrice*od.Quantity*(1 - od.Discount)) AS "Total
FROM [Order Details] od INNER JOIN Products p
ON p.ProductID = od.ProductID INNER JOIN Suppliers s
ON p.SupplierID = s.SupplierID
GROUP BY s.CompanyName
HAVING SUM(od.UnitPrice*od.Quantity*(1 - od.Discount)) > 10000;
```

- Results:

CompanyName	Total Value Sold
Pasta Buttini s.r.l.	50254.6101
Plutzer Lebensmittelgroßmärkte AG	145372.3992
G'day, Mate	65626.77011
Bigfoot Breweries	22391.20004
Tokyo Traders	30526.34003
Cooperativa de Quesos 'Las Cabras'	25159.43008
PB Knäckebröd AB	11724.06001
Forêts d'érables	61587.57006
Aux joyeux ecclésiastiques	153691.2752
Specialty Biscuits, Ltd.	46243.97989
Formaggi Fortini s.r.l.	48225.16494
Nord-Ost-Fisch Handelsgesellschaft mbH	13424.1975
Heli Süßwaren GmbH & Co. KG	38653.41949
Mayumi's	14736.75502
Norske Meierier	43141.51003
Exotic Liquids	32188.0601
Gai pâturage	117981.1802
Lyngbysild	10221.17496
Pavlova, Ltd.	106459.7755
Leka Trading	42017.6451
New Orleans Cajun Delights	31167.9899
Ma Maison	22154.63724
Grandma Kelly's Homestead	41953.29999
New England Seafood Cannery	26590.97488
Karkki Oy	28442.72757
Svensk Sjöföda AB	20144.05998

- Resulting Chart in Fig:3.1

3.3

- Question:

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

- Answer:

```
SELECT TOP 10 c.CustomerID, c.CompanyName, SUM(od.UnitPrice*od.Quantity*(1 - od
FROM Orders o INNER JOIN [Order Details] od
ON od.OrderID = o.OrderID INNER JOIN Customers c
ON o.CustomerID = c.CustomerID
```

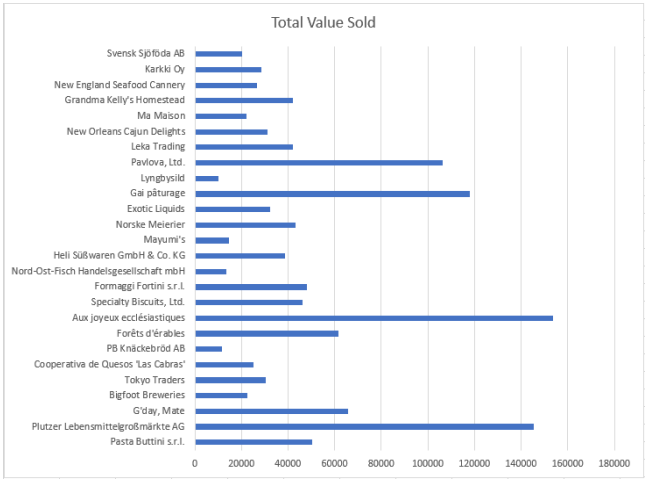

Figure 3.1:

```
WHERE YEAR(o.ShippedDate) IN (
SELECT YEAR(MAX(ord.ShippedDate))
FROM Orders ord
)
GROUP BY c.CustomerID, c.CompanyName
ORDER BY SUM(od.UnitPrice*od.Quantity*(1 - od.Discount)) DESC;
```

- Results:

CustomerID	CompanyName	(No column name)
QUICK	QUICK-Stop	37948.815
SAVEA	Save-a-lot Markets	36310.10978
ERNSH	Ernst Handel	33813.75012
HANAR	Hanari Carnes	23821.19999
KOENE	Königlich Essen	21136.27399
HUNGO	Hungry Owl All-Night Grocers	20402.11993
RATTC	Rattlesnake Canyon Grocery	19982.54993
WHITC	White Clover Markets	15278.89999
FOLKO	Folk och fä HB	13644.0675
SUPRD	Suprêmes délices	11644.6

3.4

- Question:

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

- Answer:

```
SELECT AVG(DATEDIFF(d,o.OrderDate, o.ShippedDate)) AS "Average Shiptime", DATEF
FROM Orders o
WHERE o.ShippedDate IS NOT NULL
GROUP BY DATEFROMPARTS(YEAR(o.OrderDate), MONTH(o.OrderDate), 1)
ORDER BY DATEFROMPARTS(YEAR(o.OrderDate), MONTH(o.OrderDate), 1);
```

- Results:

Average Shiptime	Date
8	1996-07-01
8	1996-08-01
10	1996-09-01
6	1996-10-01
8	1996-11-01
7	1996-12-01
9	1997-01-01
9	1997-02-01
8	1997-03-01
9	1997-04-01
9	1997-05-01
8	1997-06-01
8	1997-07-01
6	1997-08-01
9	1997-09-01
8	1997-10-01
8	1997-11-01
9	1997-12-01
9	1998-01-01
7	1998-02-01
9	1998-03-01
6	1998-04-01
2	1998-05-01

- Resulting Chart is in Fig:3.2

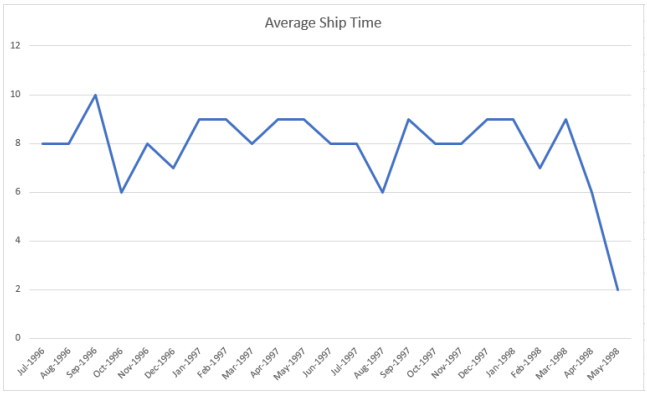
Average Ship Time
12
10
8
6
4
2
0
Jul-1996
Aug-1996
Sep-1996
Oct-1996
Nov-1996
Dec-1996
Jan-1997
Feb-1997
Mar-1997
Apr-1997
May-1997
Jun-1997
Jul-1997
Aug-1997
Sep-1997
Oct-1997
Nov-1997
Dec-1997
Jan-1998
Feb-1998
Mar-1998
Apr-1998
May-1998

Figure 3.2: