# ***WEEK 2 HOME WORK***

USE Northwind

--Q1.1

SELECT

CustomerID AS 'Customer ID',

CompanyName AS 'Company Name', Address + ' ' + City + ' ' + PostalCode AS 'Full Address'

FROM

Customers

WHERE

City = 'Paris' OR City = 'London'

--Q1.2

SELECT \* FROM Products

WHERE QuantityPerUnit LIKE '%bottles%'

USE Northwind

--Q1.3

SELECT

Suppliers.CompanyName AS 'Company Name',

Suppliers.Country

FROM

Products --Join suppliers table onto Products table to get all info

JOIN

Suppliers ON Suppliers.SupplierID = Products.SupplierID

WHERE

QuantityPerUnit LIKE '%bottles%' -- Use % wild card to ensure only bottled products are showed

USE Northwind;

--Q1.4

SELECT

Products.[CategoryID] AS 'Category ID',

CategoryName AS 'Category Name',

SUM(UnitsInStock) AS 'Units In Stock'

FROM

Products

JOIN Categories ON Categories.CategoryID = Products.CategoryID

GROUP BY

Products.CategoryID,

CategoryName

ORDER BY

Products.CategoryID DESC

USE Northwind

-- Q1.5

SELECT

TitleOfCourtesy + ' ' + FirstName + ' ' + LastName AS 'Full Name',

City

FROM

Employees

WHERE

Country = 'UK'

USE Northwind;

--Q1.6

WITH CTE\_Region AS -- A Common Table Experission has been utilised. A subquiery could have also worked

(

SELECT

-- Multiplying the sum totals of the Unit Price and the Quantity will provide to total amount made for each territory

ROUND(SUM(OD.UnitPrice \* OD.Quantity), -4) AS 'Total Sales',

RegionDescription AS 'Region'

-- Joining Territores, Region, EmployeeTerritories, Orders and Order Details should be enuugh to obtain all the information required.

-- Region and Order Details have to be joined or the calculation cant be done

FROM

Territories T

JOIN

Region R ON R.RegionID = T.RegionID

JOIN

EmployeeTerritories ET ON ET.TerritoryID = T.TerritoryID

JOIN

Orders O ON O.EmployeeID = ET.EmployeeID

JOIN

[Order Details] OD on OD.OrderID = O.OrderID

GROUP BY -- GROUP BY used in conjunction with HAVING >= 1000000 to determine the best performing regions

((OD.UnitPrice) \* (OD.Quantity)),

R.RegionDescription

HAVING

(SUM(OD.UnitPrice) \* SUM(OD.Quantity)) >= 1000000

)

SELECT

-- Summing the sales for each region will provide the total Sales for Each Region

SUM(CTE\_Region.[Total Sales]) AS 'Total Sales For Each Region',

CTE\_Region.Region FROM CTE\_Region

GROUP BY

CTE\_Region.Region

USE Northwind;

-- Q1.7

-- Using a common table experrsion allows us to see how many Frieghts we have over 100

WITH CTE\_Freight AS

(

SELECT

COUNT(OrderID) AS 'Number Of Frieghts Over 100.00',

(Freight),

ShipCountry

FROM

Orders

GROUP BY

OrderID, Freight, ShipCountry

HAVING

Freight > 100 AND ShipCountry = 'USA' OR ShipCountry = 'UK'

)

--Once the numbmers of Frights over 100 have been established. We can count the total number for them for the UK and USA

SELECT

SUM(CTE\_Freight.[Number Of Frieghts Over 100.00]) AS 'Number Of Frieghts Over 100.00',

CTE\_Freight.ShipCountry

FROM

CTE\_Freight

GROUP BY

CTE\_Freight.ShipCountry

--Q1.8 Use TOP 1 to obtain the highest discount item.

USE Northwind

-- Multiply Discount by UnitPrice to obtain the amount that was discounted.

SELECT

TOP 1

OrderID,

(Discount \* UnitPrice) AS 'Amount After Discount Applied',

UnitPrice,

Discount

FROM

[Order Details]

GROUP BY

OrderID,

(Discount \* UnitPrice),

UnitPrice,

Discount

ORDER BY

(Discount \* UnitPrice) DESC

QUESTION 2

USE MY\_SQL\_TEST\_FINAL

--Q2.1

-- Each Individual table is to be created. Tables should be Normalised.

-- Set Primary keys and Foreign keys so that tables can be macthed up for accurate data pulling

CREATE TABLE CourseDetails

(

[Course ID] INT NOT NULL IDENTITY(1,1) PRIMARY KEY,

[Course Name] VARCHAR (32),

);

CREATE TABLE StartDate

(

[Start Date] DATE,

[Course ID] INT

NOT NULL FOREIGN KEY ([Course ID])

REFERENCES CourseDetails([Course ID])

);

CREATE TABLE Spartans

(

[Spartan\_ID] INT NOT NULL IDENTITY (1,1) PRIMARY KEY,

[Spartan Name] VARCHAR(32),

[Course ID] INT

FOREIGN KEY ([Course ID])

REFERENCES CourseDetails([Course ID])

)

CREATE TABLE AcademyName

(

[Academy ID] INT NOT NULL IDENTITY (1,1) PRIMARY KEY,

[Academy Name] VARCHAR(24),

[Course ID] INT

FOREIGN KEY ([Course ID])

REFERENCES CourseDetails([Course ID])

)

CREATE TABLE EndDate

(

[End Date] DATE,

[Course ID] INT

FOREIGN KEY ([Course ID])

REFERENCES CourseDetails([Course ID])

)

CREATE TABLE RoomName

(

[Room ID] INT NOT NULL IDENTITY (1,1) PRIMARY KEY,

[Room Name] VARCHAR (24),

[Course ID] INT

FOREIGN KEY ([Course ID])

REFERENCES CourseDetails([Course ID])

)

CREATE TABLE Trainers

(

[Trainer ID] INT NOT NULL IDENTITY (1,1) PRIMARY KEY,

[Trainer Name] VARCHAR (32),

[Course ID] INT

FOREIGN KEY ([Course ID])

REFERENCES CourseDetails([Course ID])

)

USE MY\_SQL\_TEST\_FINAL

--Q2.2

SELECT \* FROM

CourseDetails

JOIN

StartDate ON StartDate.[Course ID] = CourseDetails.[Course ID]

JOIN

EndDate ON EndDate.[Course ID] = CourseDetails.[Course ID]

JOIN

AcademyName ON AcademyName.[Course ID] = CourseDetails.[Course ID]

JOIN

RoomName ON RoomName.[Course ID] = CourseDetails.[Course ID]

JOIN

Trainers ON Trainers.[Course ID] = CourseDetails.[Course ID]

JOIN

Spartans ON Spartans.[Course ID] = CourseDetails.[Course ID]

USE MY\_SQL\_TEST\_FINAL

--Q3.1

SELECT \* FROM

CourseDetails

JOIN

StartDate ON StartDate.[Course ID] = CourseDetails.[Course ID]

JOIN

EndDate ON EndDate.[Course ID] = CourseDetails.[Course ID]

JOIN

AcademyName ON AcademyName.[Course ID] = CourseDetails.[Course ID]

JOIN

RoomName ON RoomName.[Course ID] = CourseDetails.[Course ID]

JOIN

Trainers ON Trainers.[Course ID] = CourseDetails.[Course ID]

JOIN

Spartans ON Spartans.[Course ID] = CourseDetails.[Course ID]

USE MY\_SQL\_TEST\_FINAL

--Q3.3

--EXEC sp\_RENAME 'EndDate.[Check\_Date]' , 'Check Date', 'COLUMN';

-- Used to change the name From Date\_Check to 'Check Date'

UPDATE

EndDate

SET

[Check Date] = DATEADD(dd, (8\*7),'2018-01-15')

WHERE

EndDate.[Course ID] <= 6;

UPDATE

EndDate

SET

[Check Date] = DATEADD(dd, (12\*7),'2018-01-22')

WHERE

EndDate.[Course ID] >= 7;