DROP TABLE film\_table

CREATE TABLE film\_table

(

   film\_name VARCHAR(20),

   film\_type VARCHAR(10),

   Date\_of\_release DATE,

   Director VARCHAR(30),

   Writer VARCHAR(30),

   STAR VARCHAR(30),

   Film\_language VARCHAR(30),

   Official\_website VARCHAR(30),

   Plot\_summary VARCHAR(30)

);

INSERT INTO film\_table(

    film\_name,

    film\_type,

    Date\_of\_release,

    Director,

    Writer,

    STAR,

    Film\_language,

    Official\_website,

    Plot\_summary

)VALUES(

    'The grand budapest',

    'Comedy',

    '10/15/2015',

    'Wes Anderson',

    'Who knows',

    'Jude Law',

    'English',

    'rfrfieurfiue',

    'blah blah blah'

);

SELECT \*

FROM film\_table

--CREATE DATABASE spartan\_table\_db;

-- DROP TABLE table\_people;

-- CREATE TABLE table\_people

-- (

--     people\_ID INT PRIMARY KEY,

--     Name VARCHAR(30)

-- );

-- DROP TABLE table\_course;

-- CREATE TABLE table\_course

-- (

--     Course\_ID INT PRIMARY KEY,

--     Course\_steam VARCHAR(30),

--     Modules VARCHAR(30)

-- );

-- DROP TABLE table\_trainer;

-- CREATE TABLE table\_trainer

-- (

--     Trainer\_ID INT PRIMARY KEY,

--     Trainer\_name VARCHAR(30),

-- );

-- --DROP TABLE table\_Facts;

-- CREATE TABLE table\_Facts

-- (

--     People\_ID INT FOREIGN KEY REFERENCES table\_people(people\_ID),

--     Course\_ID INT FOREIGN KEY REFERENCES table\_course(course\_ID),

--     Trainer\_ID INT FOREIGN KEY REFERENCES table\_trainer(trainer\_ID)

-- );

-- ALTER TABLE table\_people

--     ALTER COLUMN people\_ID INT IDENTITY(1,1) PRIMARY KEY;

-- INSERT INTO table\_people

-- (

--     NAME

-- )VALUES

-- (

--     'MICHAEL'--,MATTHEW,ELIZABETH,ADAM

-- )

SELECT \*

FROM table\_Facts;

SELECT count(\*)

FROM Employees

WHERE City = 'London';

SELECT \*

FROM Employees

WHERE TitleOfCourtesy = 'Dr.';

SELECT Count(\*)

FROM Products

WHERE Discontinued = 'True';

18/11/21

SELECT UnitPrice, Quantity, Discount,

    UnitPrice\*Quantity AS "Gross Total",

    UnitPrice\*Quantity\*(1-Discount) AS "Net Total"

FROM[Order Details]

SELECT UnitPrice, Quantity, Discount, OrderID,

    UnitPrice\*Quantity AS "Gross Total",

    UnitPrice\*Quantity\*(1-Discount) AS "Net Total"

FROM[Order Details]

ORDER BY "Net Total" DESC

SELECT PostalCode "Post Code",

LEFT(PostalCode, CHARINDEX(' ',PostalCode)-1) AS "Post Code Region",

CHARINDEX(' ',PostalCode) AS "Space Found", Country

FROM Customers

WHERE Country = 'UK'

SET QUOTED\_IDENTIFIER OFF

SELECT ProductName

FROM Products

WHERE CHARINDEX ("'",ProductName) > 0

SELECT FirstName + ',' + LastName AS 'EmpName',

    DATEDIFF(year,BirthDate,GETDATE()) AS 'Age'

FROM Employees

ORDER BY 'Age' DESC

SELECT CASE

WHEN DATEDIFF(d,OrderDate,ShippedDate) <10 THEN 'On Time'

ELSE 'Overdue'

END AS "Status"

FROM Orders

SELECT FirstName + ',' + LastName AS 'EmpName',

    DATEDIFF(DAY,BirthDate,GETDATE())/365.25 AS 'Age',

    CASE

    WHEN DATEDIFF(DAY,BirthDate,GETDATE())/365.25 >65 THEN 'Retired'

    WHEN DATEDIFF(DAY,BirthDate,GETDATE())/365.25 >60 THEN 'Retirement due'

    WHEN DATEDIFF(DAY,BirthDate,GETDATE())/365.25 =60 THEN 'More than 5yrs to go'

    ELSE 'Get back to work'

    END AS "Start working"

FROM Employees

ORDER BY 'Age' DESC

SELECT SupplierID,

SUM(unitsOnOrder) AS "Total On Order",

    AVG(UnitsOnOrder) AS 'Average',

    MIN(UnitsOnOrder) AS 'Min',

    MAX(UnitsOnOrder) AS 'Max',

SupplierID

FROM Products

GROUP BY SupplierID

SELECT CategoryID,

    AVG(ReorderLevel) AS 'Avg'

FROM Products

GROUP BY CategoryID

ORDER BY 'Avg' DESC

SELECT SupplierID,

SUM(UnitsOnOrder) AS "Total On Order",

    AVG(UnitsOnOrder) AS 'Average',

SupplierID

FROM Products

GROUP BY SupplierID

HAVING AVG(UnitsOnOrder) >5

SELECT \*

FROM Orders INNER JOIN Customers

ON Customers.CustomerID = Orders.CustomerID

WHERE Country = 'Brazil'

SELECT CompanyName,

    AVG(UnitsOnOrder) AS 'Avg'

    FROM Products INNER JOIN Suppliers

    ON Suppliers.SupplierID = Products.SupplierID

    GROUP BY CompanyName

Better Version than before – Dannys, you were close

SELECT p.SupplierID,

s.CompanyName,

AVG(UnitsOnOrder) AS 'Avg units on order'

FROM Products p INNER JOIN Suppliers s

ON p.SupplierID = s.SupplierID

GROUP BY p.SupplierID, s.CompanyName

SELECT OrderID,OrderDate,Freight,CompanyName, FirstName + ',' + LastName AS 'Employee Name'

-- FROM Orders o

-- INNER JOIN Employees e ON o.EmployeeID = e.EmployeeID

-- INNER JOIN Customers c ON o.CustomerID = c.CustomerID

SELECT o.OrderID, CompanyName AS 'Customers'

FROM Customers c

LEFT JOIN Orders o ON c.CustomerID = o.CustomerID

WHERE OrderID IS NULL

SELECT count(\*) FROM [Order Details] od

WHERE ProductID IN (SELECT ProductID FROM Products WHERE Discontinued=1)

**The mini practice questions**

**19/11/2021**

-- 1.1

-- SELECT CustomerID, CompanyName, Address, City, PostalCode

-- FROM Customers

-- WHERE City IN ('Paris','London'); -- OR City = 'London'

--1.2

-- SELECT QuantityPerUnit

-- FROM Products

-- WHERE QuantityPerUnit LIKE '%tles'

--1.3

-- SELECT ProductName, QuantityPerUnit, CompanyName, Country

-- FROM Products p INNER JOIN Suppliers s

-- ON p.SupplierID = s.SupplierID

-- WHERE QuantityPerUnit LIKE '%tles'

--1.4

-- SELECT c.CategoryName, COUNT(\*) AS "No of prods"

-- FROM Categories c INNER JOIN Products p

-- ON c.CategoryID = p.CategoryID

-- GROUP BY c.CategoryName

-- ORDER BY [No of prods] DESC

SELECT name,

    price,

    MIN(average\_stock) --AS "Minimum Average Category Stock"

FROM product p

JOIN (

    SELECT product\_category\_id, AVG(available\_stock) AS "average\_stock"

    FROM product

    GROUP BY product\_category\_id

) av ON p.product\_category\_id = av.product\_category\_id

**ADVANCED SQL 22/11/21**

Subquery and a join

SELECT ord.OrderID, ord.OrderDate, ord.CustomerID,o.EmployeeID, o.[Employee Order Count]

FROM Orders ord INNER JOIN (SELECT EmployeeID, COUNT(\*) AS "Employee Order Count"

FROM Orders GROUP BY EmployeeID) o ON ord.EmployeeID = o.EmployeeID

SELECT OrderID, EmployeeID, OrderDate,

COUNT(\*) OVER (PARTITION BY EmployeeID) AS 'Employee Order Count' --() AS 'Total Count'

FROM Orders

ORDER BY OrderID

-- NEED TO FINISH / CAPTURE 2 - DONE

-- SELECT p.ProductID, p.ProductName, p.CategoryID, c.CategoryName,

--     COUNT(\*) OVER (PARTITION BY p.CategoryID) -1 AS "No prods, each category"

-- FROM Products p INNER JOIN Categories c ON p.CategoryID = c.CategoryID--INNER JOIN Categories c ON p.CategoryID = c.CategoryID

SELECT \*,

COUNT(\*) OVER () AS 'Total Count'

FROM Products

SELECT p.ProductID,p.ProductName, c.CategoryName,

SUM(UnitsOnOrder) OVER (PARTITION BY c.categoryName) AS 'Total Units on order for category'

FROM Products p INNER JOIN Categories c ON p.CategoryID=c.CategoryID

DANIEL Solution, I used SUM he used Count

SELECT p.ProductID,p.ProductName, c.CategoryName,

COUNT(UnitsOnOrder) OVER (PARTITION BY c.categoryID) AS 'Total Units on order for category'

FROM Products p INNER JOIN Categories c ON p.CategoryID=c.CategoryID

-- SELECT ProductName,

-- UnitPrice,

-- DENSE\_RANK() OVER (ORDER BY UnitPrice DESC) AS 'Row Number'

-- FROM Products

SELECT ProductName,

UnitPrice, CategoryID,

ROW\_NUMBER() OVER (PARTITION BY CategoryID ORDER BY UnitPrice DESC) AS 'Row Number'

FROM Products

SELECT OrderID,CustomerID,OrderDate,

RANK() OVER (PARTITION BY CustomerID ORDER BY OrderID DESC) AS "test"

FROM Orders

Model underneath

SELECT OrderID,CustomerID,OrderDate,

RANK() OVER (PARTITION BY CustomerID ORDER BY OrderDate) AS "Ordered by order date per cus"

FROM Orders

ORDER BY OrderID

SELECT ProductName, SupplierID,

FIRST\_VALUE(UnitsOnOrder) OVER (PARTITION BY SupplierID ORDER BY UnitsOnOrder DESC) AS 'Most Units on Order from Supplier'

FROM Products

SELECT OrderID, CustomerID, EmployeeID, OrderDate,

DATEDIFF(DAY,FIRST\_VALUE(OrderDate) OVER (PARTITION BY CustomerID ORDER BY OrderDATE), GETDATE()) AS "Time since first order"

FROM Orders

ORDER BY OrderID

--Inital Statement

-- SELECT OrderID, EmployeeID,

-- LAST\_VALUE(OrderID) Over (PARTITION BY EmployeeID ORDER BY OrderID)

-- AS 'Last order for Employee'

-- FROM Orders

-- Same as above

-- SELECT OrderID, EmployeeID,

-- LAST\_VALUE(OrderID) Over (PARTITION BY EmployeeID ORDER BY OrderID ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)

-- AS 'Last order for Employee'

-- FROM Orders

-- SELECT OrderID, EmployeeID,

-- LAST\_VALUE(OrderID) Over (PARTITION BY EmployeeID ORDER BY OrderID ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING)

-- AS 'Last order for Employee'

-- FROM Orders

--Michael Work - Go and make sense of it dummy

SELECT o.OrderID, o.OrderDate, [Total Price],

AVG([Total Price]) OVER (ORDER BY OrderDate ROWS BETWEEN 2 PRECEDING AND 2 FOLLOWING) AS '5pt Moving AVG'

FROM Orders o INNER JOIN [Order Details] od

ON o.OrderID = od.OrderID INNER JOIN (SELECT orderID, SUM((UnitPrice \* Quantity \* (1 - Discount))) AS 'Total Price' FROM [Order Details] GROUP BY OrderID) sq1

ON o.orderID = sq1.orderID

--GROUP BY o.OrderID, o.OrderDate, [Total Price]

ORDER BY o.orderID;

**23/11/2021 – Further SQL**

SELECT ProductName

FROM products p

INNER JOIN (SELECT CategoryID

FROM Products

GROUP BY CategoryID

HAVING AVG(UnitPrice) > 30) m ON p.CategoryID = m.CategoryID