--List products that have an order quantity greatern than 20

SELECT Name from SalesLT.Product

WHERE ProductID IN(SELECT ProductID FROM SalesLT.SalesOrderDetail WHERE OrderQty>20);

--Self Contained

SELECT CustomerID,SalesOrderID,Orderdate FROM SalesLT.SalesOrderHeader AS SO1

WHERE OrderDate =

(SELECT MAX(OrderDate)

FROM SalesLT.SalesOrderHeader AS SO2 WHERE SO2.CustomerID = SO1.CustomerID)

ORDER BY CustomerID;

--CROSSS APPLY

CREATE FUNCTION SalesLT.udfMaxUnitPrice (@SalesOrderID int)

RETURNS TABLE

AS

RETURN

SELECT SalesOrderID,Max(UnitPrice) As MaxUnitPrice FROM

SalesLT.SalesOrderDetail

WHERE SalesOrderID=@SalesOrderID

GROUP BY SalesOrderID;

--Displaying the sales order detail for items that are equal to

-- the maximum unit price for a particular sales order

SELECT \* FROM SalesLT.SalesOrderDetail AS SOH

CROSS APPLY

SalesLT.udfMaxUnitPrice(SalesOrderID) AS MUP

WHERE SOH.UnitPrice=MUP.MaxUnitPrice

ORDER BY SOH.SalesOrderID;

--CROSS APPLY

SELECT\*FROM SalesLT.SalesOrderDetail as SOD

CROSS APPLY

SalesLT.udfMaxUnitPrice(SOH.SalesOrderID) AS MUP

WHERE SOD.UnitPrice=MUP.MaxUnitPrice

Order by SOD.SalesOrderID

SELECT\*FROM SalesLT.SalesOrderDetail

--VIEWS

CREATE VIEW SalesLT. vProductColoredYellow

AS

SELECT \* FROM SalesLT.product

WHERE Color='yellow';

--USING A VIEW

SELECT Name,Weight, color FROM SalesLT.vProductsColoredYellow;

SELECT C.customerID,c.firstName,c.LastName,A.AddressLine1,A.city,A.StateProvince

FROM SalesLT.Customer AS c

JOIN SalesLT.CustomerAddress AS ca

ON c.CustomerID =ca.CustomerID

JOIN SalesLT.Address As A

ON ca.AddressID= A.AddressID

--MAKING THE TOP ONE A VIEW

CREATE VIEW SalesLT.vCustomerAddress

AS

SELECT C.customerID,c.firstName,c.LastName,A.AddressLine1,A.city,A.StateProvince

FROM SalesLT.Customer AS c

JOIN SalesLT.CustomerAddress AS ca

ON c.CustomerID =ca.CustomerID

JOIN SalesLT.Address As A

ON ca.AddressID= A.AddressID;

SELECT StateProvince,City,ISNULL(SUM(soh.TotalDue),0.00) AS totalRevenue

FROM SalesLT.vCustomerAddress

JOIN SalesLT.SalesOrderHeader AS soh

ON vCustomerAddress.CustomerID=soh.CustomerID

GROUP BY StateProvince,city

ORDER BY StateProvince,totalRevenue;

--Temporary Table

CREATE TABLE #Color

(

color VARCHAR(15)

);

INSERT INTO #Color SELECT DISTINCT (Color) FROM SalesLT.Product;

SELECT color FROM #Color;

--Table Variables queries - stores immediately and disappear or derived table

DECLARE @Colors

AS TABLE

(

Color VARCHAR(15)

);

INSERT INTO @Colors SELECT DISTINCT(Color)FROM

SalesLT.Product;

SELECT color FROM @Colors;

--one derived statement query, find a productid, name as product name and product category as category.

--the derived table should have an alias

SELECT Category, COUNT(ProductID)

FROM

(SELECT p.ProductID, p.Name AS ProductName, pc.Name AS Category

FROM SalesLT.Product AS p

JOIN SalesLT.ProductCategory AS pc

ON p.ProductCategoryID = pc.ProductCategoryID)

AS ProdCats

GROUP BY Category;

--you cannot count the productID without categorising them. from the above preamble. this sums all the values up into one answer

SELECT COUNT(ProductID)

FROM

(SELECT p.ProductID, p.Name AS ProductName, c.Name AS Category

FROM SalesLT.Product AS p

JOIN SalesLT.ProductCategory AS c

ON p.ProductCategoryID = c.ProductCategoryID)

AS ProdCats

--Common table expressions CTEs ie name table expressions

--for all products in the CTE form create using WITH. use the preamble above to write a CTE

-- Once you use COUNT or Aggregate function you need to GROUP BY

-- for a CTE it does not belong to one category hence final answer dsiplays all comapred to derived tables

--MAX RECURSION shows how many time it recurrs its self

WITH ProductsByCategory(ProductID, ProductName, Category)

AS

(SELECT p.ProductID, p.Name AS ProductName, pc.Name AS Category

FROM SalesLT.Product AS p

JOIN SalesLT.ProductCategory AS pc

ON p.ProductCategoryID = pc.ProductCategoryID

)

SELECT Category, COUNT(ProductID)

FROM ProductsByCategory

GROUP BY Category

OPTION(MAXRECURSION 3)

--GROUPING AND PIVOTING TABLES

--GROUPING SET

--PIVOTING

--converting row values into columns values