Class 22 Homework Answers

# Homework

1. Using Sales.SpecialDeals, get the full deal details of the deal with the largest discount percentage.

SELECT \*

FROM Sales.SpecialDeals sd

JOIN ( SELECT MAX(DiscountPercentage) AS MaxDiscount

FROM Sales.SpecialDeals ) AS md

ON sd.DiscountPercentage = md.MaxDiscount;

1. Using Sales.OrderLines, write a query that will get the full order line details for the most expensive item on each order (i.e. highest UnitPrice \* Quantity value). Try doing it both with an inline view and a CTE.

SELECT ol.\*

FROM Sales.OrderLines ol

JOIN ( SELECT MAX(UnitPrice \* Quantity) AS MaxPrice,

OrderID

FROM Sales.OrderLines

GROUP BY OrderID ) AS maxol

ON (ol.UnitPrice \* ol.Quantity) = maxol.MaxPrice

AND ol.OrderId = maxol.OrderID

ORDER BY OrderID;

WITH MaxOL AS (

SELECT MAX(UnitPrice \* Quantity) AS MaxPrice,

OrderID

FROM Sales.OrderLines

GROUP BY OrderID )

SELECT ol.\*

FROM Sales.OrderLines ol

JOIN MaxOL

ON (ol.UnitPrice \* ol.Quantity) = maxol.MaxPrice

AND ol.OrderId = maxol.OrderID

ORDER BY OrderID;

1. Using Sales.Orders and Sales.OrderLines, get the full order line details for the most expensive line on the most recent order (i.e. max order date) for each customer. When multiple orders exist for a customer, get the most expensive order line from any one of them (i.e not one line per order).

WITH MaxOrderDate AS (

SELECT MAX(OrderDate) AS MaxDate,

CustomerID

FROM Sales.Orders

GROUP BY CustomerID ),

MaxOL AS (

SELECT MAX(UnitPrice \* Quantity) AS MaxPrice,

o.OrderID,

o.CustomerID

FROM Sales.OrderLines ol

JOIN Sales.Orders o

ON ol.OrderID = o.OrderID

JOIN MaxOrderDate mo

ON o.OrderDate = mo.MaxDate

AND o.CustomerID = mo.CustomerID

GROUP BY o.OrderID, o.CustomerID )

SELECT o.CustomerID, o.OrderDate, ol.\*

FROM Sales.OrderLines ol

JOIN MaxOL

ON (ol.UnitPrice \* ol.Quantity) = maxol.MaxPrice

AND ol.OrderId = maxol.OrderID

JOIN Sales.Orders o

ON o.OrderID = ol.OrderID

ORDER BY CustomerID;

1. Using the recursive CTE example above, modify the query to track how many levels deep the recursion is (i.e. start with 0 for an order with no backorder, then add 1 for each level above that).

WITH Backorder ( OrderID, OrderChain, OrderLevel, BackorderOrderID ) AS

(

-- Anchor member. Does not reference our CTE

SELECT OrderID,

CAST(OrderID AS VARCHAR(MAX)) AS OrderChain,

1 AS OrderLevel,

BackorderOrderID

FROM Sales.Orders

WHERE BackorderOrderID IS NULL

-- Recursive members. Must be joined in using UNION ALL. References our CTE.

UNION ALL

SELECT o.OrderID,

b.OrderChain + ', ' + CAST(o.OrderID AS VARCHAR(MAX)) AS OrderChain,

b.OrderLevel + 1 AS OrderLevel,

o.BackorderOrderID

FROM Sales.Orders o

JOIN Backorder b

ON o.BackorderOrderID = b.OrderID

)

SELECT \*

FROM Backorder

WHERE BackorderOrderID IS NOT NULL

ORDER BY OrderLevel DESC;