https://github.com/Desteferi11/edvai.git

-- 1

SELECT

c.CategoryName,

p.ProductName,

p.UnitPrice,

ROUND(AVG(p.UnitPrice) OVER(PARTITION BY c.CategoryName), 2) AS AvgPrice

FROM

Products p

JOIN

Categories c ON p.CategoryID = c.CategoryID;

--2

-- Consulta para obtener el promedio de venta de cada cliente

--consulta: pide el promedio de ventas por clientes y en este caso trae el cliente, el total de ordenes y el promedio de ventas

SELECT

c.CustomerID,

COUNT(DISTINCT o.OrderID) AS TotalOrders,

AVG(od.UnitPrice \* od.Quantity) AS AvgSale

FROM

Customers c

INNER JOIN

Orders o ON c.CustomerID = o.CustomerID

INNER JOIN

[Order Details] od ON o.OrderID = od.OrderID

GROUP BY

c.CustomerID;

-- imagen

SELECT

AVG(od.UnitPrice \* od.Quantity) AS AverageOrderAmount,

o.OrderID,

o.CustomerID,

o.EmployeeID,

o.OrderDate,

o.RequiredDate,

DATEDIFF(day, o.OrderDate, o.RequiredDate) AS DaysBetweenOrderAndRequired

FROM

Orders o

INNER JOIN

[Order Details] od ON o.OrderID = od.OrderID

GROUP BY

o.OrderID,

o.CustomerID,

o.EmployeeID,

o.OrderDate,

o.RequiredDate,

o.ShippedDate;

---wf/CategoryID

WITH OrderDetailsWithDays AS (

SELECT

o.OrderID,

o.CustomerID,

o.EmployeeID,

o.OrderDate,

o.RequiredDate,

DATEDIFF(day, o.OrderDate, o.RequiredDate) AS DaysBetweenOrderAndRequired,

od.UnitPrice \* od.Quantity AS OrderAmount,

p.CategoryID

FROM

Orders o

INNER JOIN

[Order Details] od ON o.OrderID = od.OrderID

INNER JOIN

Products p ON od.ProductID = p.ProductID

)

SELECT

AVG(OrderAmount) OVER(PARTITION BY CategoryID) AS AverageOrderAmount,

OrderID,

CustomerID,

EmployeeID,

OrderDate,

RequiredDate,

DaysBetweenOrderAndRequired

FROM

OrderDetailsWithDays;

--3

--Obtener el promedio de cantidad de productos vendidos por categoría (product\_name,

--quantity\_per\_unit, unit\_price, quantity, avgquantity) y ordenarlo por nombre de la

--categoría y nombre del producto

SELECT

c.CategoryName,

p.ProductName,

p.QuantityPerUnit,

p.UnitPrice,

od.Quantity,

AVG(od.Quantity) OVER(PARTITION BY c.CategoryName) AS AvgQuantity

FROM

Products p

INNER JOIN

Categories c ON p.CategoryID = c.CategoryID

INNER JOIN

[Order Details] od ON p.ProductID = od.ProductID

ORDER BY

c.CategoryName,

p.ProductName;

--4

-- Consulta para seleccionar el ID del cliente, la fecha de la orden y la fecha más antigua de la orden para cada cliente

SELECT

CustomerID,

OrderDate,

MIN(OrderDate) OVER(PARTITION BY CustomerID) AS EarliestOrderDate

FROM

Orders;

--5

-- Consulta para seleccionar el ID de producto, el nombre de producto, el precio unitario, el ID de categoría y el precio unitario máximo para cada categoría

SELECT

ProductID,

ProductName,

UnitPrice,

CategoryID,

MAX(UnitPrice) OVER(PARTITION BY CategoryID) AS MaxUnitPrice

FROM

Products;

--6

--

WITH ProductSalesRank AS (

SELECT

p.ProductID,

p.ProductName,

SUM(od.Quantity) AS TotalQuantitySold,

ROW\_NUMBER() OVER (ORDER BY SUM(od.Quantity) DESC) AS SalesRank

FROM

Products p

INNER JOIN

[Order Details] od ON p.ProductID = od.ProductID

GROUP BY

p.ProductID, p.ProductName

)

SELECT

ProductID,

ProductName,

TotalQuantitySold,

SalesRank

FROM

ProductSalesRank;

--

SELECT

ROW\_NUMBER() OVER(ORDER BY SUM(od.Quantity) DESC) AS SalesRank,

p.ProductName,

SUM(od.Quantity) AS TotalQuantity

FROM

Products p

INNER JOIN

[Order Details] od ON p.ProductID = od.ProductID

GROUP BY

p.ProductName

ORDER BY

SalesRank;

--7

--Asignar numeros de fila para cada cliente, ordenados por customer\_id

SELECT

ROW\_NUMBER() OVER(ORDER BY CustomerID) AS RowNumber,

CustomerID,

CompanyName,

ContactName,

Address

FROM

Customers;

--8

-- Consulta para obtener el ranking de los empleados más jóvenes

SELECT

ROW\_NUMBER() OVER(ORDER BY BirthDate) AS Ranking,

FirstName + ' ' + LastName AS EmployeeName,

BirthDate

FROM

Employees

ORDER BY

Ranking ASC;

--9

--

SELECT

c.CustomerID,

c.CompanyName,

SUM(od.UnitPrice \* od.Quantity) AS TotalSales

FROM

Customers c

INNER JOIN

Orders o ON c.CustomerID = o.CustomerID

INNER JOIN

[Order Details] od ON o.OrderID = od.OrderID

GROUP BY

c.CustomerID,

c.CompanyName;

--10

--Obtener la suma total de ventas por categoría de producto

SELECT

cat.CategoryID,

cat.CategoryName,

p.ProductName,

od.UnitPrice,

od.Quantity,

SUM(od.UnitPrice \* od.Quantity) AS TotalSales

FROM

Categories cat

INNER JOIN

Products p ON cat.CategoryID = p.CategoryID

INNER JOIN

[Order Details] od ON p.ProductID = od.ProductID

GROUP BY

cat.CategoryID,

cat.CategoryName,

p.ProductName,

od.UnitPrice,

od.Quantity;

--11

-- Consulta para calcular la suma total de gastos de envío por país de destino y ordenarlos por país en orden ascendente

SELECT

ShipCountry,

SUM(Freight) AS TotalFreight

FROM

Orders

GROUP BY

ShipCountry

ORDER BY

ShipCountry ASC;

--12

--RANK////Ranking de ventas por cliente

SELECT

c.CustomerID,

c.CompanyName,

SUM(od.UnitPrice \* od.Quantity) AS TotalSales,

RANK() OVER(ORDER BY SUM(od.UnitPrice \* od.Quantity) DESC) AS SalesRank

FROM

Customers c

INNER JOIN

Orders o ON c.CustomerID = o.CustomerID

INNER JOIN

[Order Details] od ON o.OrderID = od.OrderID

GROUP BY

c.CustomerID,

c.CompanyName;

--13

-- Consulta para obtener el ranking de empleados por fecha de contratación

SELECT

EmployeeID,

FirstName,

LastName,

HireDate,

ROW\_NUMBER() OVER(ORDER BY HireDate) AS HireDateRank

FROM

Employees;

--14

-- Consulta para obtener el ranking de productos por precio unitario (desde el más barato al más caro)

SELECT

ProductID,

ProductName,

UnitPrice,

ROW\_NUMBER() OVER(ORDER BY UnitPrice) AS PriceRank

FROM

Products;

--15

--Mostrar por cada producto de una orden, la cantidad vendida y la cantidad

--vendida del producto previo.

SELECT

OrderID,

ProductID,

Quantity AS QuantitySold,

LAG(Quantity) OVER(PARTITION BY ProductID ORDER BY OrderID) AS PreviousQuantitySold

FROM

[Order Details];

--16

WITH LastOrderDates AS (

SELECT

OrderID,

CustomerID,

OrderDate,

LAG(OrderDate) OVER(PARTITION BY CustomerID ORDER BY OrderDate) AS LastOrderDate

FROM

Orders

)

SELECT

OrderID,

OrderDate,

CustomerID,

MAX(LastOrderDate) OVER(PARTITION BY CustomerID) AS LastOrderDate

FROM

LastOrderDates;

--otra forma 16

SELECT

OrderID,

OrderDate,

CustomerID,

MAX(LastOrderDate) OVER(PARTITION BY CustomerID) AS LastOrderDate

FROM (

SELECT

OrderID,

CustomerID,

OrderDate,

LAG(OrderDate) OVER(PARTITION BY CustomerID ORDER BY OrderDate) AS LastOrderDate

FROM

Orders

) AS LastOrderDates;

--17

WITH ProductPrices AS (

SELECT

ProductID,

ProductName,

UnitPrice,

LAG(UnitPrice) OVER(ORDER BY ProductID) AS PreviousUnitPrice

FROM

Products

)

SELECT

ProductID,

ProductName,

UnitPrice,

PreviousUnitPrice,

(UnitPrice - PreviousUnitPrice) AS PriceDifference

FROM

ProductPrices;

--18

WITH ProductPrices AS (

SELECT

ProductID,

UnitPrice,

LEAD(UnitPrice) OVER(ORDER BY ProductID) AS NextUnitPrice

FROM

Products

)

SELECT

ProductID,

UnitPrice AS CurrentPrice,

NextUnitPrice AS NextPrice

FROM

ProductPrices;

-- otraforma

SELECT

ProductID,

UnitPrice AS CurrentPrice,

LEAD(UnitPrice) OVER(ORDER BY ProductID) AS NextPrice

FROM

Products;

---- otras windfunc

---me dejo estas notas de otras funciones

--dense rank

SELECT

CustomerID,

CompanyName,

TotalSales,

DENSE\_RANK() OVER(ORDER BY TotalSales DESC) AS DenseSalesRank

FROM (

SELECT

c.CustomerID,

c.CompanyName,

SUM(od.UnitPrice \* od.Quantity) AS TotalSales

FROM

Customers c

INNER JOIN

Orders o ON c.CustomerID = o.CustomerID

INNER JOIN

[Order Details] od ON o.OrderID = od.OrderID

GROUP BY

c.CustomerID,

c.CompanyName

) AS CustomerSales;

--ntile (dvido en grupos)

SELECT

ProductID,

ProductName,

UnitPrice,

NTILE(3) OVER(ORDER BY UnitPrice) AS PriceGroup

FROM

Products;

--ntile

SELECT

PriceGroup,

AVG(UnitPrice) AS AveragePrice

FROM (

SELECT

ProductID,

ProductName,

UnitPrice,

NTILE(3) OVER(ORDER BY UnitPrice) AS PriceGroup

FROM

Products

) AS ProductsWithGroups

GROUP BY

PriceGroup;

--- first y last value

SELECT

CategoryID,

FIRST\_VALUE(ProductName) OVER(PARTITION BY CategoryID ORDER BY ProductName) AS FirstProduct,

LAST\_VALUE(ProductName) OVER(PARTITION BY CategoryID ORDER BY ProductName ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS LastProduct

FROM

Products;

--percentrank

SELECT

ProductID,

ProductName,

UnitPrice,

ROUND(PERCENT\_RANK() OVER(ORDER BY UnitPrice), 2) AS PricePercentile

FROM

Products;

--distri acumulativa ///cume\_dist

SELECT

ProductID,

ProductName,

UnitPrice,

CUME\_DIST() OVER(ORDER BY UnitPrice) AS CumulativeDistribution

FROM

Products;