-- Advanced SQL Exercises for Online Retail Store with Outputs

-- Exercise 1: Ranking and Window Functions

-- Goal: Use ROW\_NUMBER(), RANK(), DENSE\_RANK(), OVER(), and PARTITION BY to find top 3 most expensive products in each category

SELECT \* FROM (

SELECT

ProductID,

ProductName,

Category,

Price,

ROW\_NUMBER() OVER(PARTITION BY Category ORDER BY Price DESC) AS RowNum,

RANK() OVER(PARTITION BY Category ORDER BY Price DESC) AS RankNum,

DENSE\_RANK() OVER(PARTITION BY Category ORDER BY Price DESC) AS DenseRankNum

FROM Products

) AS RankedProducts

WHERE RowNum <= 3;

-- Output: Top 3 products per category with:

-- - RowNum: unique rank (no ties)

-- - RankNum: same rank for ties, with gaps

-- - DenseRankNum: same rank for ties, no gaps

-- Exercise 2: Aggregation with GROUPING SETS, CUBE, and ROLLUP

-- Goal: Total quantity sold by Region and Category

SELECT Region, Category, SUM(Quantity) AS TotalQuantity

FROM Orders o

JOIN OrderDetails od ON o.OrderID = od.OrderID

JOIN Customers c ON o.CustomerID = c.CustomerID

JOIN Products p ON od.ProductID = p.ProductID

GROUP BY GROUPING SETS ((Region), (Category), (Region, Category));

-- Output: Results with totals for:

-- - Region only

-- - Category only

-- - Region and Category combined

SELECT Region, Category, SUM(Quantity) AS TotalQuantity

FROM Orders o

JOIN OrderDetails od ON o.OrderID = od.OrderID

JOIN Customers c ON o.CustomerID = c.CustomerID

JOIN Products p ON od.ProductID = p.ProductID

GROUP BY ROLLUP (Region, Category);

-- Output: Results with subtotals for each Region and grand total

SELECT Region, Category, SUM(Quantity) AS TotalQuantity

FROM Orders o

JOIN OrderDetails od ON o.OrderID = od.OrderID

JOIN Customers c ON o.CustomerID = c.CustomerID

JOIN Products p ON od.ProductID = p.ProductID

GROUP BY CUBE (Region, Category);

-- Output: Results for all combinations of Region and Category, including totals

-- Exercise 3: CTEs and MERGE

-- (a) Recursive CTE to generate calendar

WITH DateCTE AS (

SELECT CAST('2025-01-01' AS DATE) AS CalendarDate

UNION ALL

SELECT DATEADD(DAY, 1, CalendarDate)

FROM DateCTE

WHERE CalendarDate < '2025-01-31'

)

SELECT \* FROM DateCTE;

-- Output: 31 rows - one for each day in January 2025

-- (b) MERGE updated product prices

MERGE INTO Products AS target

USING StagingProducts AS source

ON target.ProductID = source.ProductID

WHEN MATCHED THEN

UPDATE SET target.Price = source.Price

WHEN NOT MATCHED THEN

INSERT (ProductID, ProductName, Category, Price)

VALUES (source.ProductID, source.ProductName, source.Category, source.Price);

-- Output: Updated prices for matched products, inserted new products if not present

-- Exercise 4: PIVOT and UNPIVOT

-- (a) Pivot monthly sales quantity per product

SELECT \* FROM (

SELECT ProductID, MONTH(OrderDate) AS SalesMonth, Quantity

FROM Orders o

JOIN OrderDetails od ON o.OrderID = od.OrderID

) AS SalesData

PIVOT (

SUM(Quantity) FOR SalesMonth IN ([1], [2], [3], [4], [5], [6])

) AS PivotTable;

-- Output: Each row represents a ProductID, columns show quantities sold in Jan-Jun

-- (b) Unpivot it back

SELECT ProductID, SalesMonth, Quantity

FROM (

SELECT ProductID, [1], [2], [3], [4], [5], [6]

FROM PivotTable

) p

UNPIVOT (

Quantity FOR SalesMonth IN ([1], [2], [3], [4], [5], [6])

) AS Unpivoted;

-- Output: Flat table showing sales quantities by product and month (1–6)

-- Exercise 5: CTE for Customers with > 3 orders

WITH CustomerOrderCounts AS (

SELECT o.CustomerID, COUNT(o.OrderID) AS OrderCount

FROM Orders o

GROUP BY o.CustomerID

)

SELECT c.CustomerID, c.Name, coc.OrderCount

FROM CustomerOrderCounts coc

JOIN Customers c ON c.CustomerID = coc.CustomerID

WHERE coc.OrderCount > 3;

-- Output: List of customers who have made more than 3 orders, with names and order counts