**Exercise 1: Ranking and Window Functions**

**Using ROW\_Number()**

SELECT \*

FROM (

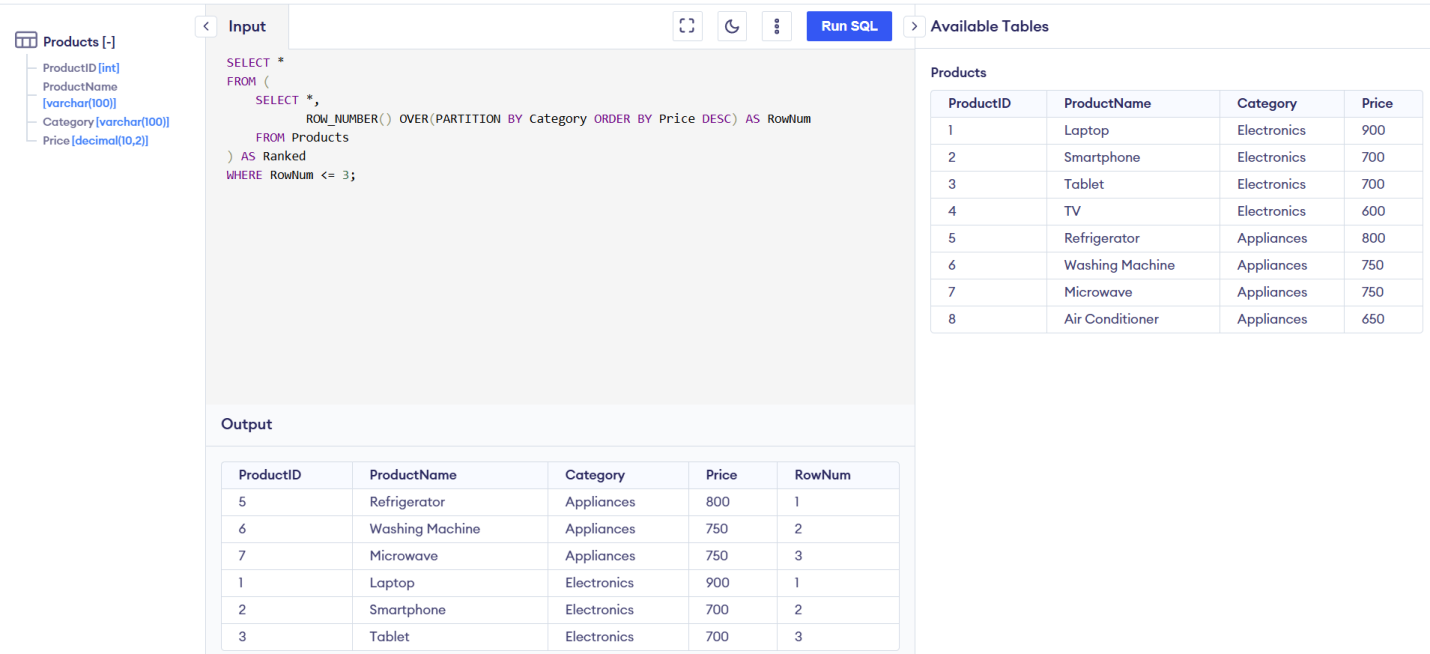
SELECT \*,

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

FROM Products

) AS Ranked

WHERE RowNum <= 3;



2. Using RANK()

SELECT \*

FROM (

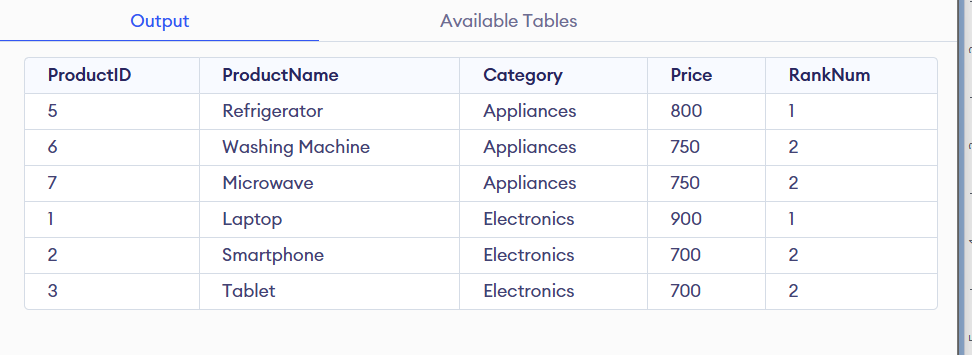
SELECT \*,

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

FROM Products

) AS Ranked

WHERE RankNum <= 3;



**3. Using DENSE\_RANK()**

SELECT \*

FROM (

SELECT \*,

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

FROM Products

) AS Ranked

WHERE DenseRankNum <= 3;

****

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

* **Query with GROUPING SETS**

SELECT

c.Region,

p.Category,

SUM(od.Quantity) AS TotalQuantity

FROM Customers c

JOIN Orders o ON c.CustomerID = o.CustomerID

JOIN OrderDetails od ON o.OrderID = od.OrderID

JOIN Products p ON od.ProductID = p.ProductID

GROUP BY GROUPING SETS (

(c.Region),

(p.Category),

(c.Region, p.Category)

);



🡪CUBE()

SELECT

c.Region,

p.Category,

SUM(od.Quantity) AS TotalQuantity

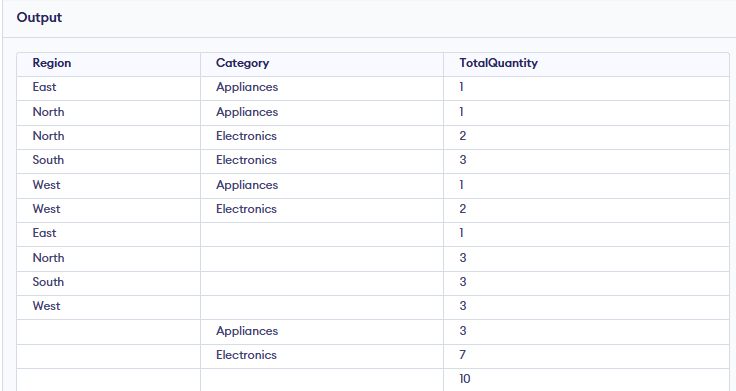
FROM Customers c

JOIN Orders o ON o.CustomerID = c.CustomerID

JOIN OrderDetails od ON o.OrderID = od.OrderID

JOIN Products p ON od.ProductID = p.ProductID

GROUP BY CUBE (c.Region, p.Category);



**Rollup()**

SELECT

c.Region,

p.Category,

SUM(od.Quantity) AS TotalQuantity

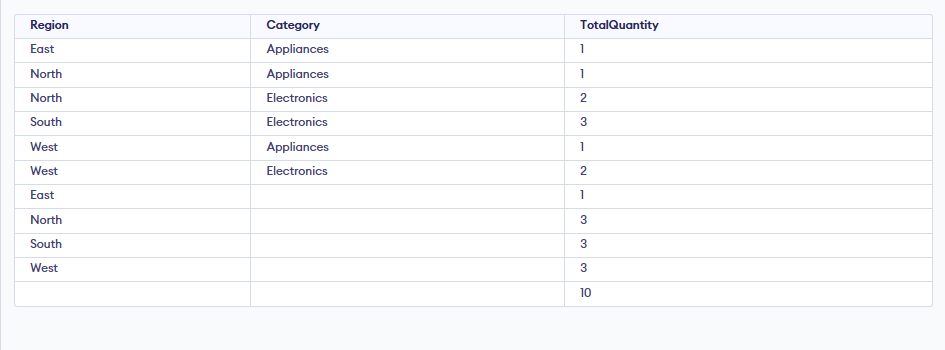
FROM Customers c

JOIN Orders o ON o.CustomerID = c.CustomerID

JOIN OrderDetails od ON o.OrderID = od.OrderID

JOIN Products p ON od.ProductID = p.ProductID

GROUP BY ROLLUP (c.Region, p.Category);



**Exercise 3: CTEs and MERGE**

**🡪Create a Recursive CTE to Generate Dates (2025-01-01 to 2025-01-31)**

WITH RECURSIVE Calendar AS (

SELECT DATE '2025-01-01' AS CalendarDate

UNION ALL

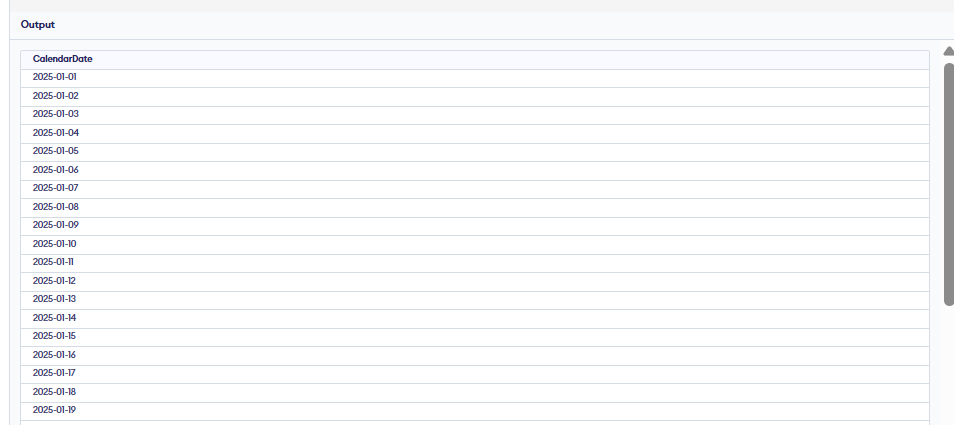
SELECT CalendarDate + INTERVAL '1 day'

FROM Calendar

WHERE CalendarDate < DATE '2025-01-31'

)

SELECT \* FROM Calendar;



🡪Create a StagingProducts Table with Updated/New Products

CREATE TABLE StagingProducts (

ProductID INT,

ProductName TEXT,

Category TEXT,

Price DECIMAL(10,2)

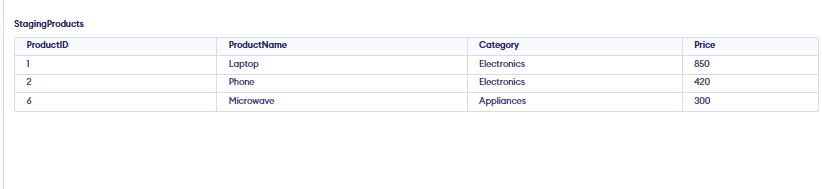
);

INSERT INTO StagingProducts VALUES

(1, 'Laptop', 'Electronics', 850),

(2, 'Phone', 'Electronics', 420),

(6, 'Microwave', 'Appliances', 300);



**🡪Use MERGE to Update/Insert**

MERGE INTO Products AS Target

USING StagingProducts AS Source

ON Target.ProductID = Source.ProductID

WHEN MATCHED THEN

UPDATE SET

Target.ProductName = Source.ProductName,

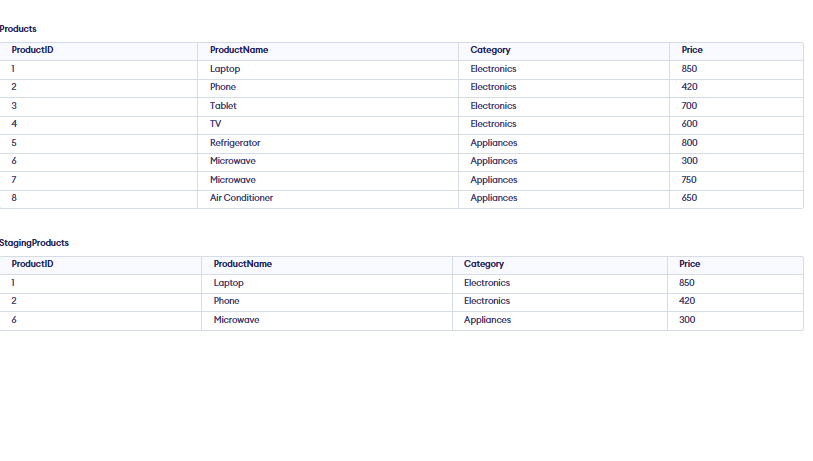
Target.Category = Source.Category,

Target.Price = Source.Price

WHEN NOT MATCHED THEN

INSERT (ProductID, ProductName, Category, Price)

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

****

Exercise 4: PIVOT and UNPIVOT

🡪 Aggregate sales by Product and Month

SELECT

p.ProductName,

FORMAT(o.OrderDate, 'MM') AS Month,

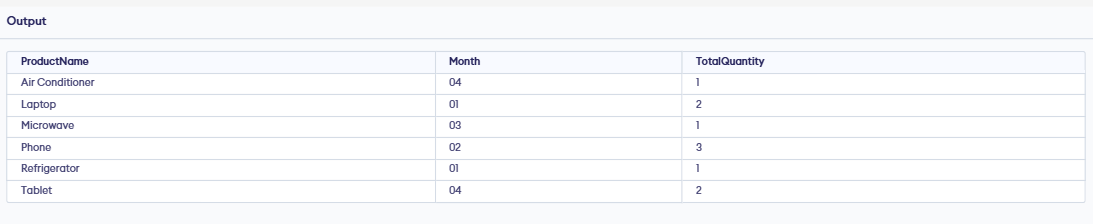
SUM(od.Quantity) AS TotalQuantity

FROM Orders o

JOIN OrderDetails od ON o.OrderID = od.OrderID

JOIN Products p ON od.ProductID = p.ProductID

GROUP BY p.ProductName, FORMAT(o.OrderDate, 'MM');



🡪Pivot

SELECT \*

FROM (

SELECT

p.ProductName,

FORMAT(o.OrderDate, 'MM') AS Month,

od.Quantity

FROM Orders o

JOIN OrderDetails od ON o.OrderID = od.OrderID

JOIN Products p ON od.ProductID = p.ProductID

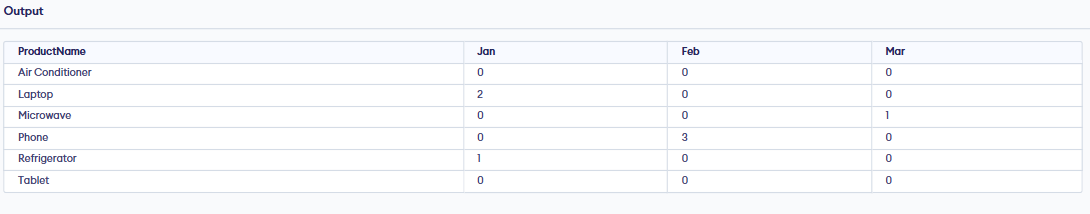
) AS SourceTable

PIVOT (

SUM(Quantity)

FOR Month IN ([01], [02], [03])

) AS PivotTable;



🡪 UNPIVOT

SELECT ProductName, Month, Quantity

FROM (

SELECT ProductName, [01], [02], [03]

FROM (

SELECT

p.ProductName,

FORMAT(o.OrderDate, 'MM') AS Month,

od.Quantity

FROM Orders o

JOIN OrderDetails od ON o.OrderID = od.OrderID

JOIN Products p ON od.ProductID = p.ProductID

) AS InnerData

PIVOT (

SUM(Quantity) FOR Month IN ([01], [02], [03])

) AS Pivoted

) AS Final

UNPIVOT (

Quantity FOR Month IN ([01], [02], [03])

) AS Unpivoted;



**Exercise 5: Using CTE to Simplify a Query**

WITH CustomerOrderCounts AS (

SELECT

CustomerID,

COUNT(\*) AS TotalOrders

FROM Orders

GROUP BY CustomerID

)

WITH CustomerOrderCounts AS (

SELECT

CustomerID,

COUNT(\*) AS TotalOrders

FROM Orders

GROUP BY CustomerID

)

SELECT

c.CustomerID,

c.CustomerName,

c.Region,

coc.TotalOrders

FROM CustomerOrderCounts coc

JOIN Customers c ON c.CustomerID = coc.CustomerID

WHERE coc.TotalOrders > 3

ORDER BY coc.TotalOrders DESC;

