Superset ID- 6363144

Name-Harshita Sinha

WEEK 2 ADVANCED SQL CONCEPTS

**Exercise 1: Ranking and Window Functions**

CREATE TABLE Products (

ProductID INT PRIMARY KEY,

ProductName VARCHAR(100),

Category VARCHAR(50),

Price DECIMAL(10, 2)

);

INSERT INTO Products (ProductID, ProductName, Category, Price)

VALUES

(1, 'Laptop', 'Electronics', 1200.00),

(2, 'Smartphone', 'Electronics', 900.00),

(3, 'Headphones', 'Electronics', 150.00),

(4, 'TV', 'Electronics', 1500.00),

(5, 'Monitor', 'Electronics', 300.00),

(6, 'T-shirt', 'Clothing', 25.00),

(7, 'Jeans', 'Clothing', 50.00),

(8, 'Jacket', 'Clothing', 80.00),

(9, 'Sneakers', 'Clothing', 70.00),

(10, 'Coat', 'Clothing', 80.00);

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;

WITH RankedProducts AS (

SELECT

\*,

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

FROM Products

)

SELECT

ProductID,

ProductName,

Category,

Price

FROM RankedProducts

WHERE RowNum <= 3;

OUTPUT:-



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

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY,

Name VARCHAR(100),

Region VARCHAR(50)

);

CREATE TABLE Orders (

OrderID INT PRIMARY KEY,

CustomerID INT,

OrderDate DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE Products (

ProductID INT PRIMARY KEY,

ProductName VARCHAR(100),

Category VARCHAR(50)

);

CREATE TABLE OrderDetails (

OrderDetailID INT PRIMARY KEY,

OrderID INT,

ProductID INT,

Quantity INT,

FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),

FOREIGN KEY (ProductID) REFERENCES Products(ProductID)

);

INSERT INTO Customers VALUES

(1, 'Alice', 'North'),

(2, 'Bob', 'South'),

(3, 'Charlie', 'North'),

(4, 'David', 'East');

INSERT INTO Orders VALUES

(1, 1, '2025-01-01'),

(2, 2, '2025-01-02'),

(3, 3, '2025-01-03'),

(4, 4, '2025-01-04');

INSERT INTO Products VALUES

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

(2, 'TV', 'Electronics'),

(3, 'Jeans', 'Clothing'),

(4, 'Sneakers', 'Clothing');

INSERT INTO OrderDetails VALUES

(1, 1, 1, 2),

(2, 1, 2, 1),

(3, 2, 3, 4),

(4, 3, 4, 1),

(5, 4, 2, 3),

(6, 4, 3, 2);

SELECT

c.Region,

p.Category,

SUM(od.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 (

(c.Region),

(p.Category),

(c.Region, p.Category)

);

SELECT

c.Region,

p.Category,

SUM(od.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 (c.Region, p.Category);

SELECT

c.Region,

p.Category,

SUM(od.Quantity) AS TotalQuantity

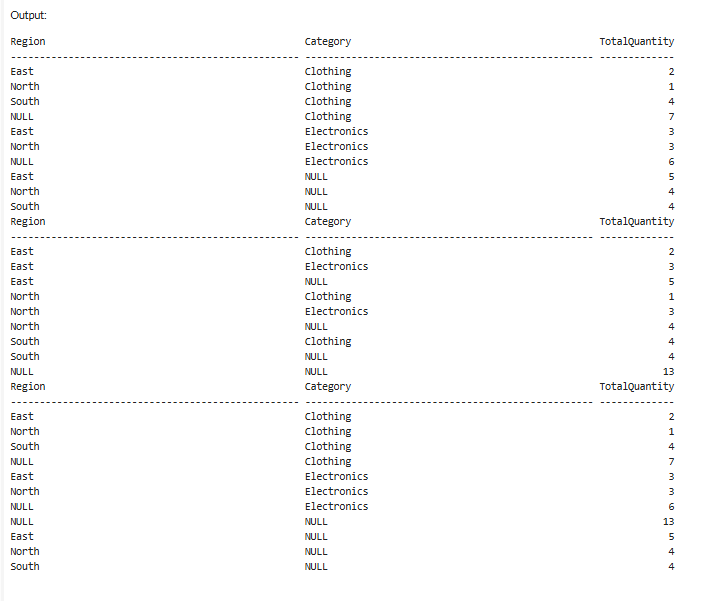
FROM Orders o

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 (c.Region, p.Category);



Exercise 3: CTEs and MERGE

-- Generate calendar dates

WITH Calendar AS (

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

UNION ALL

SELECT DATEADD(DAY, 1, CalendarDate)

FROM Calendar

WHERE CalendarDate < '2025-01-31'

)

SELECT \* FROM Calendar

OPTION (MAXRECURSION 0);

-- Products table

CREATE TABLE Products (

ProductID INT PRIMARY KEY,

ProductName VARCHAR(100),

Price DECIMAL(10, 2),

Category VARCHAR(50)

);

-- Sample data in Products

INSERT INTO Products VALUES

(1, 'Laptop', 1000.00, 'Electronics'),

(2, 'TV', 1500.00, 'Electronics'),

(3, 'Jeans', 40.00, 'Clothing');

-- StagingProducts with new prices

CREATE TABLE StagingProducts (

ProductID INT,

ProductName VARCHAR(100),

Price DECIMAL(10, 2),

Category VARCHAR(50)

);

-- Sample data in staging

INSERT INTO StagingProducts VALUES

(2, 'TV', 1600.00, 'Electronics'), -- price updated

(4, 'T-shirt', 25.00, 'Clothing'); -- new product

MERGE Products AS target

USING StagingProducts AS source

ON target.ProductID = source.ProductID

WHEN MATCHED THEN

UPDATE SET

target.ProductName = source.ProductName,

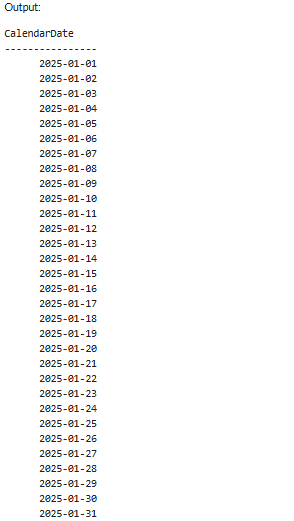
target.Price = source.Price,

target.Category = source.Category

WHEN NOT MATCHED THEN

INSERT (ProductID, ProductName, Price, Category)

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



**Exercise 4: PIVOT and UNPIVOT**

-- Products table (if not already created)

IF NOT EXISTS (SELECT \* FROM sysobjects WHERE name='Products')

BEGIN

CREATE TABLE Products (

ProductID INT PRIMARY KEY,

ProductName VARCHAR(100),

Category VARCHAR(50)

);

INSERT INTO Products VALUES

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

(2, 'Jeans', 'Clothing');

END

-- Orders table

CREATE TABLE Orders (

OrderID INT PRIMARY KEY,

OrderDate DATE

);

INSERT INTO Orders VALUES

(1, '2025-01-10'),

(2, '2025-02-15'),

(3, '2025-01-20'),

(4, '2025-03-05');

-- OrderDetails table

CREATE TABLE OrderDetails (

OrderDetailID INT PRIMARY KEY,

OrderID INT,

ProductID INT,

Quantity INT,

FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),

FOREIGN KEY (ProductID) REFERENCES Products(ProductID)

);

INSERT INTO OrderDetails VALUES

(1, 1, 1, 2), -- Jan

(2, 2, 1, 1), -- Feb

(3, 3, 2, 3), -- Jan

(4, 4, 2, 2); -- Mar

SELECT \*

FROM (

SELECT

p.ProductName,

FORMAT(o.OrderDate, 'MMM') AS SalesMonth,

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 SalesMonth IN ([Jan], [Feb], [Mar])

) AS PivotTable;

-- Assuming previous query output is saved as a table or CTE named PivotTable

-- Here's the unpivot form directly from PivotTable result:

SELECT

ProductName,

SalesMonth,

Quantity

FROM (

SELECT

p.ProductName,

FORMAT(o.OrderDate, 'MMM') AS SalesMonth,

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)

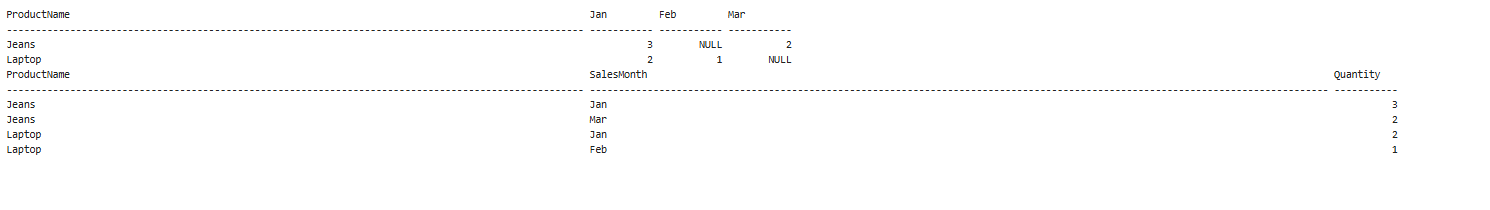
FOR SalesMonth IN ([Jan], [Feb], [Mar])

) AS Pivoted

UNPIVOT (

Quantity FOR SalesMonth IN ([Jan], [Feb], [Mar])

) AS Unpivoted;



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

IF OBJECT\_ID('OrderDetails', 'U') IS NOT NULL DROP TABLE OrderDetails;

IF OBJECT\_ID('Orders', 'U') IS NOT NULL DROP TABLE Orders;

IF OBJECT\_ID('Customers', 'U') IS NOT NULL DROP TABLE Customers;

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY,

Name VARCHAR(100),

Region VARCHAR(50)

);

CREATE TABLE Orders (

OrderID INT PRIMARY KEY,

CustomerID INT,

OrderDate DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

INSERT INTO Customers (CustomerID, Name, Region) VALUES

(1, 'Harshita', 'North'),

(2, 'Babli', 'South'),

(3, 'Simran', 'North'),

(4, 'Soumya', 'East');

INSERT INTO Orders (OrderID, CustomerID, OrderDate) VALUES

(1, 1, '2025-01-01'),

(2, 2, '2025-01-02'),

(3, 3, '2025-01-03'),

(4, 4, '2025-01-04');

INSERT INTO Orders (OrderID, CustomerID, OrderDate) VALUES

(5, 1, '2025-02-01'),

(6, 1, '2025-02-02'),

(7, 1, '2025-02-03'),

(8, 2, '2025-02-04'),

(9, 2, '2025-02-05');

WITH CustomerOrderCounts AS (

SELECT

CustomerID,

COUNT(OrderID) AS OrderCount

FROM Orders

GROUP BY CustomerID

)

SELECT

c.CustomerID,

c.Name,

coc.OrderCount

FROM CustomerOrderCounts coc

JOIN Customers c ON c.CustomerID = coc.CustomerID

WHERE coc.OrderCount > 3;

