**NAME - SAUMY RAJ**

**SUPERSET ID - 6363348**

**Exercise 1: Ranking and Window Functions**

Goal: Use ROW\_NUMBER(), RANK(), DENSE\_RANK(), OVER(), and PARTITION BY.

Scenario:

Find the top 3 most expensive products in each category using different ranking functions.

Steps:

1. Use ROW\_NUMBER() to assign a unique rank within each category.

2. Use RANK() and DENSE\_RANK() to compare how ties are handled.

3. Use PARTITION BY Category and ORDER BY Price DESC.

ANS - QUERY -

CREATE DATABASE RetailDB;

USE RetailDB;

CREATE TABLE Products (

ProductID INT PRIMARY KEY,

ProductName VARCHAR(100),

Category VARCHAR(50),

Price DECIMAL(10,2)

);

INSERT INTO Products VALUES

(1, 'Wireless Mouse', 'Electronics', 500),

(2, 'USB-C Adapter', 'Electronics', 450),

(3, 'Gaming Laptop', 'Electronics', 85000),

(4, 'Hoodie', 'Clothing', 2000),

(5, 'Leather Jacket', 'Clothing', 8000),

(6, 'T-Shirt', 'Clothing', 800),

(7, 'Denim Jeans', 'Clothing', 2000),

(8, 'Tablet', 'Electronics', 15000);

WITH RankedProducts AS (

SELECT

ProductID,

ProductName,

Category,

Price,

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

FROM Products

)

SELECT \* FROM RankedProducts

WHERE RowNum <= 3;

WITH RankedProducts AS (

SELECT

ProductID,

ProductName,

Category,

Price,

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

FROM Products

)

SELECT \* FROM RankedProducts

WHERE PriceRank <= 3;

WITH RankedProducts AS (

SELECT

ProductID,

ProductName,

Category,

Price,

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

FROM Products

)

SELECT \* FROM RankedProducts

WHERE DensePriceRank <= 3;

SELECT

ProductID,

ProductName,

Category,

Price,

ROW\_NUMBER() OVER (

PARTITION BY Category -- Restart numbering for each category

ORDER BY Price DESC -- Sort within each category

) AS RowNum

FROM Products;

OUTPUTS-







