

## SQL Exercise: Advanced Ranking and Window Functions

- Objective:  
Use ROW\_NUMBER(), RANK(), and DENSE\_RANK() with OVER(PARTITION BY ...) to retrieve the top 3 priced products per category and explore how ranking functions behave with ties.

### Step 1: Create and Use Database

```
CREATE DATABASE RetailStore;  
GO
```

```
USE RetailStore;  
GO
```

### Step 2: Create Products Table

```
CREATE TABLE Products (  
    ProductID INT PRIMARY KEY,  
    ProductName VARCHAR(100),  
    Category VARCHAR(50),  
    Price DECIMAL(10, 2)  
);
```

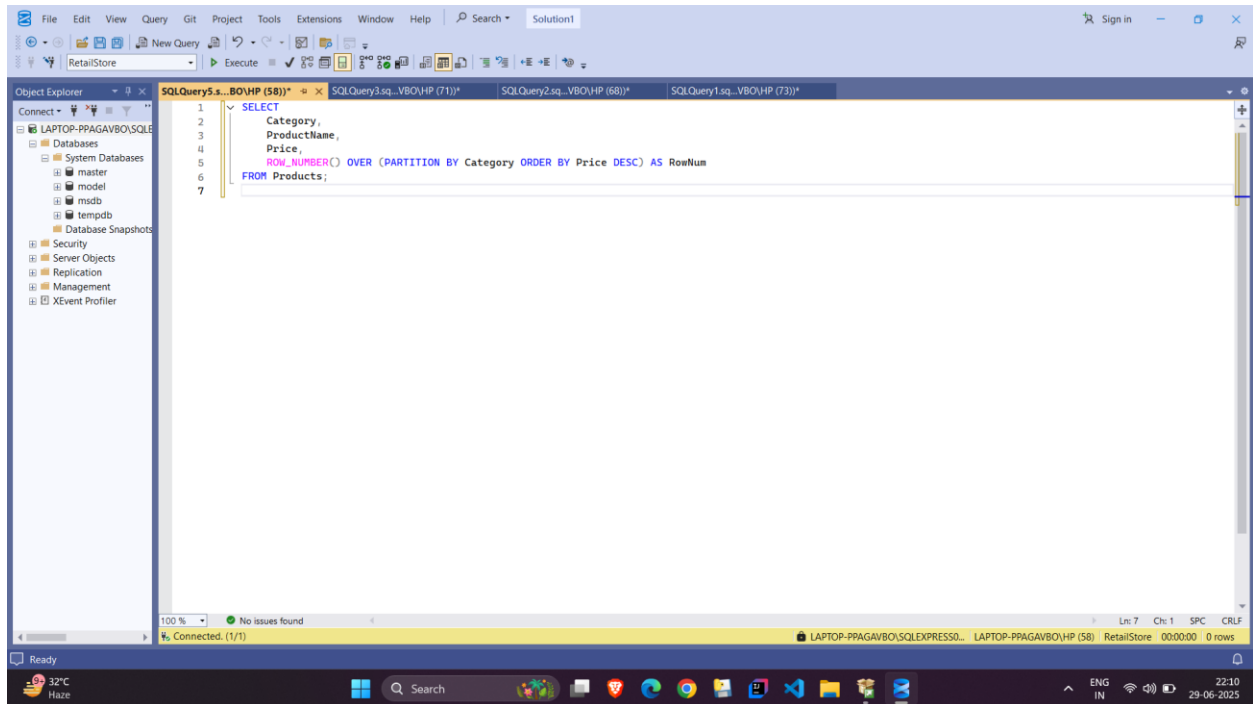
### Step 3: Insert Sample Data

```
INSERT INTO Products VALUES  
(1, 'Laptop', 'Electronics', 1000),  
(2, 'Smartphone', 'Electronics', 700),  
(3, 'Tablet', 'Electronics', 700),  
(4, 'Headphones', 'Electronics', 150),  
(5, 'Jeans', 'Clothing', 50),  
(6, 'Jacket', 'Clothing', 120),  
(7, 'Shoes', 'Clothing', 80),  
(8, 'T-Shirt', 'Clothing', 20),  
(9, 'Refrigerator', 'Appliances', 900),  
(10, 'Microwave', 'Appliances', 300),  
(11, 'Blender', 'Appliances', 300),  
(12, 'Toaster', 'Appliances', 80);
```

### Step 4: Ranking with ROW\_NUMBER()

```
SELECT  
    Category,
```

```
        ProductName,  
        Price,  
        ROW_NUMBER() OVER (PARTITION BY Category ORDER BY Price DESC) AS RowNum  
FROM Products;
```



## Step 5: Ranking with RANK() and DENSE\_RANK()

```
SELECT  
    Category,  
    ProductName,  
    Price,  
    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;
```

The screenshot shows the SQL Server Enterprise Manager interface. The Object Explorer on the left shows the server structure. The central query window displays the following SQL query:

```

1 SELECT
2   Category,
3   ProductName,
4   Price,
5   ROW_NUMBER() OVER (PARTITION BY Category ORDER BY Price DESC) AS RowNum,
6   RANK() OVER (PARTITION BY Category ORDER BY Price DESC) AS RankNum,
7   DENSE_RANK() OVER (PARTITION BY Category ORDER BY Price DESC) AS DenseRankNum
8 FROM Products;
9

```

The Results pane at the bottom shows the output of the query, which is a table with 12 rows and 6 columns: Category, ProductName, Price, RowNum, RankNum, and DenseRankNum. The data is as follows:

	Category	ProductName	Price	RowNum	RankNum	DenseRankNum
1	Appliances	Refrigerator	900.00	1	1	1
2	Appliances	Microwave	300.00	2	2	2
3	Appliances	Blender	300.00	3	2	2
4	Appliances	Toaster	80.00	4	4	3
5	Clothing	Jacket	120.00	1	1	1
6	Clothing	Shoes	80.00	2	2	2
7	Clothing	Jeans	50.00	3	3	3
8	Clothing	T-Shirt	20.00	4	4	4
9	Electronics	Laptop	1000.00	1	1	1
10	Electronics	Smartphone	700.00	2	2	2
11	Electronics	Tablet	700.00	3	2	2
12	Electronics	Headphones	150.00	4	4	3

The status bar at the bottom indicates that the query was executed successfully and returned 12 rows.

## Step 6: Get Top 3 Products Using ROW\_NUMBER()

```

WITH RankedProducts AS (
    SELECT *,
           ROW_NUMBER() OVER (PARTITION BY Category ORDER BY Price DESC) AS
RowNum
    FROM Products
)
SELECT * FROM RankedProducts
WHERE RowNum <= 3;

```

The screenshot shows the SQL Server Enterprise Manager interface. The Object Explorer on the left displays the 'RetailStore' database structure. The central query window contains the following T-SQL query:

```

1  WITH Ranked AS (
2  SELECT *,
3      RANK() OVER (PARTITION BY Category ORDER BY Price DESC) AS RankNum,
4      DENSE_RANK() OVER (PARTITION BY Category ORDER BY Price DESC) AS DenseRankNum
5  FROM Products
6  )
7  SELECT * FROM Ranked
8  WHERE RankNum <= 3 OR DenseRankNum <= 3;
9

```

The Results pane at the bottom displays the output of the query as a table with 11 rows. The status bar at the bottom indicates 'Query executed successfully.' and '11 rows'.

	ProductID	ProductName	Category	Price	RankNum	DenseRankNum
1	9	Refrigerator	Appliances	900.00	1	1
2	10	Microwave	Appliances	300.00	2	2
3	11	Blender	Appliances	300.00	2	2
4	12	Toaster	Appliances	80.00	4	3
5	6	Jacket	Clothing	120.00	1	1
6	7	Shoes	Clothing	80.00	2	2
7	5	Jeans	Clothing	50.00	3	3
8	1	Laptop	Electronics	1000.00	1	1
9	2	Smartphone	Electronics	700.00	2	2
10	3	Tablet	Electronics	700.00	2	2
11	4	Headphones	Electronics	150.00	4	3

## Step 7: Get Top 3 Products Using RANK() and DENSE\_RANK()

```

WITH Ranked AS (
    SELECT *,
        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
)
SELECT * FROM Ranked

```

WHERE RankNum <= 3 OR DenseRankNum <= 3;

The screenshot shows the SQL Server Enterprise Manager interface. The query window displays the following SQL code:

```
1 WITH Ranked AS (  
2     SELECT *,  
3     RANK() OVER (PARTITION BY Category ORDER BY Price DESC) AS RankNum,  
4     DENSE_RANK() OVER (PARTITION BY Category ORDER BY Price DESC) AS DenseRankNum  
5     FROM Products  
6 )  
7 SELECT * FROM Ranked  
8 WHERE RankNum <= 3 OR DenseRankNum <= 3;  
9
```

The results grid shows the following data:

	ProductID	ProductName	Category	Price	RankNum	DenseRankNum
1	9	Refrigerator	Appliances	900.00	1	1
2	10	Microwave	Appliances	300.00	2	2
3	11	Blender	Appliances	300.00	2	2
4	12	Toaster	Appliances	80.00	4	3
5	6	Jacket	Clothing	120.00	1	1
6	7	Shoes	Clothing	80.00	2	2
7	5	Jeans	Clothing	50.00	3	3
8	1	Laptop	Electronics	1000.00	1	1
9	2	Smartphone	Electronics	700.00	2	2
10	3	Tablet	Electronics	700.00	2	2
11	4	Headphones	Electronics	150.00	4	3

The status bar at the bottom indicates "Query executed successfully." and "11 rows".

## Summary

- ROW\_NUMBER() always gives unique ranks, no matter if prices are tied.
- RANK() skips rank numbers when there are ties.
- DENSE\_RANK() does not skip rank numbers when there are ties.

The queries return top 3 items per category based on these differences.