**Exercise 1: Ranking and Window Functions**

**Code:-**

**Creating Table Products:-**

**CREATE TABLE Products (**

**ProductID INT PRIMARY KEY,**

**ProductName VARCHAR(100),**

**Category VARCHAR(50),**

**Price DECIMAL(10, 2)**

**);**

**Inserting Values In Table Products:-**

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

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

**(2, 'Smartphone', 'Electronics', 800.00),**

**(3, 'Tablet', 'Electronics', 800.00), -- Same price as Smartphone (tie)**

**(4, 'Headphones', 'Accessories', 150.00),**

**(5, 'Mouse', 'Accessories', 150.00), -- Same price as Headphones (tie)**

**(6, 'Keyboard', 'Accessories', 100.00),**

**(7, 'Monitor', 'Electronics', 600.00);**

**Using ROW\_NUMBER():-**

**SELECT**

**ProductID,**

**ProductName,**

**Category,**

**Price,**

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

**FROM**

**Products**

**ORDER BY**

**Category, Price DESC;**

**Using RANK():-**

**SELECT**

**ProductID,**

**ProductName,**

**Category,**

**Price,**

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

**FROM**

**Products**

**ORDER BY**

**Category, Price DESC;**

**Using DENSE\_RANK():-**

**SELECT**

**ProductID,**

**ProductName,**

**Category,**

**Price,**

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

**FROM**

**Products**

**ORDER BY**

**Category, Price DESC;**

**TOP THREE RANKS:-**

**WITH Ranked AS (**

**SELECT**

**ProductID,**

**ProductName,**

**Category,**

**Price,**

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

**FROM**

**Products**

**)**

**SELECT \***

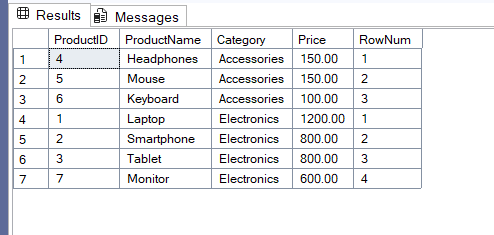
**FROM Ranked**

**WHERE RankNum <= 3**

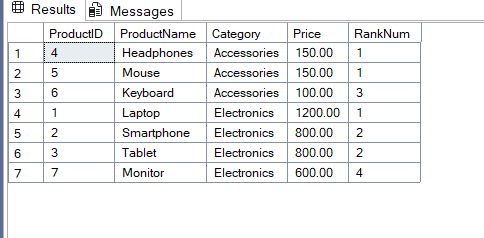
**ORDER BY Category, Price DESC;**

**OUTPUTS:-**

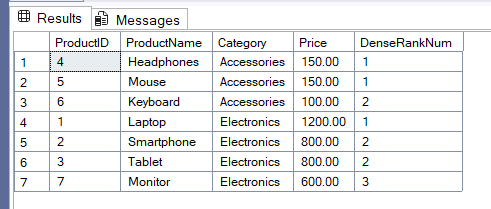
**ROW\_NUMBER():-**



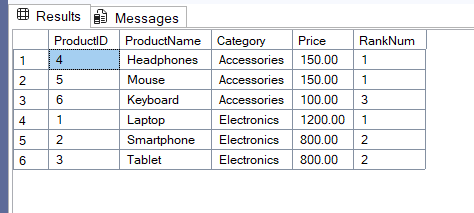
**RANK():-**



**DENSE\_RANK():-**



**TOP THREE RANKS:-**



**Stored Procedure:-**

**Exercise 1: Create a Stored Procedure**

1. **Define the stored procedure with a parameter for DepartmentID.**

**2. Write the SQL query to select employee details based on the DepartmentID.**

**Code :-**

CREATE PROCEDURE sp\_GetEmployeesByDepartment

@DepartmentID INT

AS

BEGIN

SELECT

EmployeeID,

FirstName,

LastName,

DepartmentID,

Salary,

JoinDate

FROM

Employees

WHERE

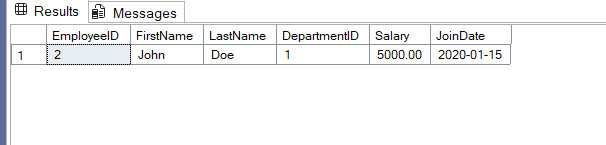
DepartmentID = @DepartmentID;

END;

GO

EXEC sp\_GetEmployeesByDepartment @DepartmentID = 1;

**OUTPUT:-**



1. **Create a stored procedure named `sp\_InsertEmployee` with the following code:**

DROP PROCEDURE IF EXISTS sp\_InsertEmployee;

GO

CREATE PROCEDURE sp\_InsertEmployee

@FirstName VARCHAR(50),

@LastName VARCHAR(50),

@DepartmentID INT,

@Salary DECIMAL(10,2),

@JoinDate DATE

AS

BEGIN

INSERT INTO Employees (FirstName, LastName, DepartmentID, Salary, JoinDate)

VALUES (@FirstName, @LastName, @DepartmentID, @Salary, @JoinDate);

END;

GO

**EXECUTING:-**

EXEC sp\_InsertEmployee

@FirstName = 'Alice',

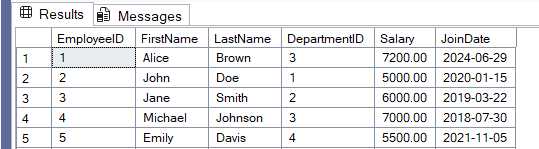
@LastName = 'Brown',

@DepartmentID = 3,

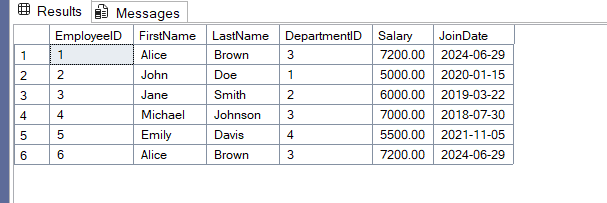
@Salary = 7200.00,

@JoinDate = '2024-06-29';

**OUTPUT (BEFORE):-**



**OUTPUT (AFTER):-**



**Exercise 5: Return Data from a Stored Procedure**

1. **Define the stored procedure with a parameter for DepartmentID.**
2. **Write the SQL query to count the number of employees in the specified department.**

**3. Save the stored procedure by executing the Stored procedure content.**

**CODE:-**

DROP PROCEDURE IF EXISTS sp\_GetEmployeeCountByDepartment;

GO

CREATE PROCEDURE sp\_GetEmployeeCountByDepartment

@DepartmentID INT

AS

BEGIN

SELECT

COUNT(\*) AS TotalEmployees

FROM

Employees

WHERE

DepartmentID = @DepartmentID;

END;

GO

EXEC sp\_GetEmployeeCountByDepartment @DepartmentID = 3;

**OUTPUT:-**

