**Week 2 - Advanced SQL - Hands-on**

DEBANJAN KAURI(Superset ID-6362388)

**Exercise 1: Ranking and Window Functions**

Scenario:

Identify the top 3 most expensive products in each category using SQL ranking techniques.

To achieve this:

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

2. Apply RANK() and DENSE\_RANK() to compare handling of ties.

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

### **SQL Code:**

IF OBJECT\_ID('Products') IS NOT NULL

DROP TABLE Products;

CREATE TABLE Products (

ProductID INT PRIMARY KEY,

ProductName VARCHAR(100),

Category VARCHAR(50),

Price DECIMAL(10, 2)

);

INSERT INTO Products VALUES

(1, 'Laptop', 'Electronics', 85000),

(2, 'Smartphone', 'Electronics', 65000),

(3, 'TV', 'Electronics', 75000),

(4, 'Headphones', 'Electronics', 65000),

(5, 'Book A', 'Books', 500),

(6, 'Book B', 'Books', 500),

(7, 'Book C', 'Books', 300),

(8, 'Book D', 'Books', 200),

(9, 'Sofa', 'Furniture', 15000),

(10, 'Chair', 'Furniture', 3000),

(11, 'Table', 'Furniture', 10000),

(12, 'Cupboard', 'Furniture', 15000);

PRINT '--- ROW\_NUMBER() ---';

SELECT

Category, ProductName, Price,

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

FROM Products

ORDER BY Category, Price DESC;

PRINT '--- RANK() ---';

SELECT

Category, ProductName, Price,

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

FROM Products

ORDER BY Category, Price DESC;

PRINT '--- DENSE\_RANK() ---';

SELECT

Category, ProductName, Price,

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

FROM Products

ORDER BY Category, Price DESC;

PRINT '--- Top 3 ROW\_NUMBER ---';

WITH Ranked AS (

SELECT ProductName, Category, Price,

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

FROM Products

)

SELECT \* FROM Ranked WHERE Rnk <= 3

ORDER BY Category, Rnk;

PRINT '--- Top 3 RANK ---';

WITH Ranked AS (

SELECT ProductName, Category, Price,

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

FROM Products

)

SELECT \* FROM Ranked WHERE Rnk <= 3

ORDER BY Category, Rnk;

PRINT '--- Top 3 DENSE\_RANK ---';

WITH Ranked AS (

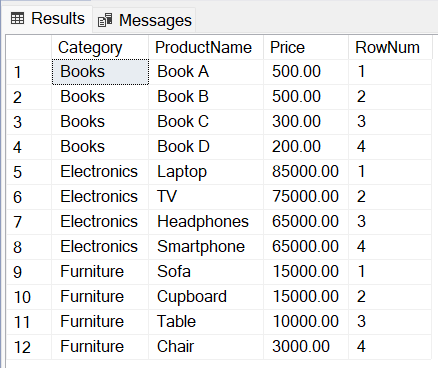
SELECT ProductName, Category, Price,

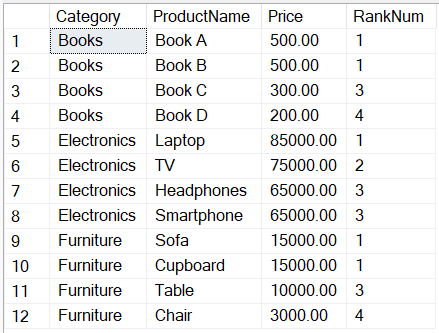
DENSE\_RANK() OVER (PARTITION BY Category ORDER BY Price DESC) AS Rnk

FROM Products

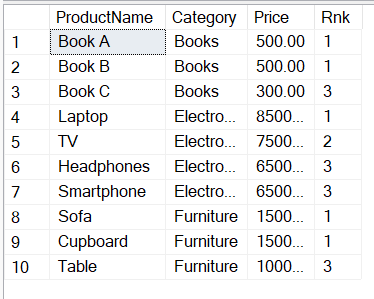
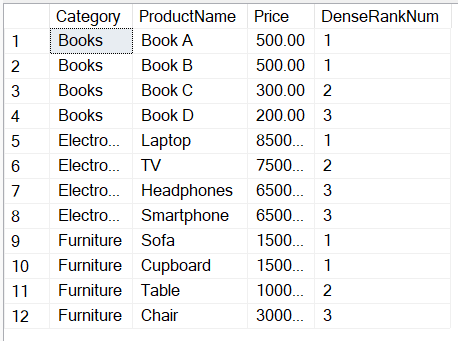
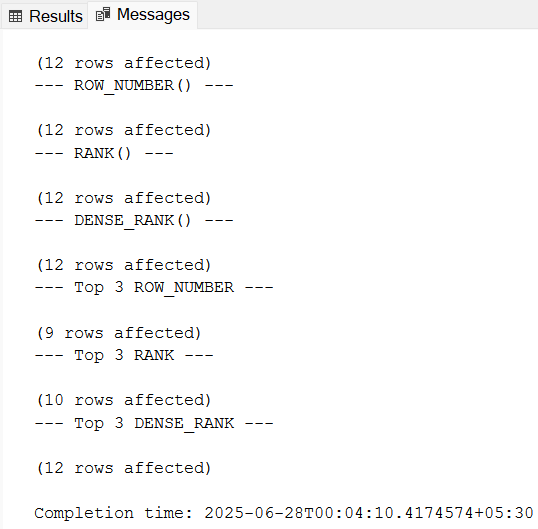
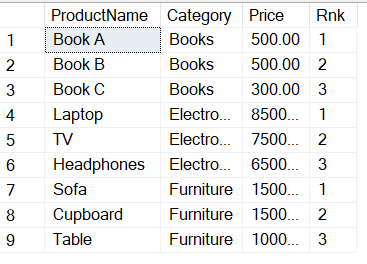
)

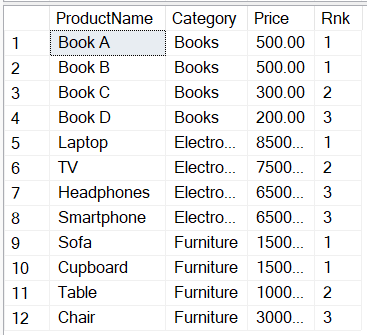
SELECT \* FROM Ranked WHERE Rnk <= 3

ORDER BY Category, Rnk;



**OUTPUT:**

**OUTPUT:**



**Database Schema (For Stored procedure Exercise):**

The following schema defines the structure for an Employee Management System:

Departments Table:

CREATE TABLE Departments

(

DepartmentID INT PRIMARY KEY,

DepartmentName VARCHAR(100)

);

Employees Table:

CREATE TABLE Employees

(

EmployeeID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

DepartmentID INT FOREIGN KEY REFERENCES Departments(DepartmentID),

Salary DECIMAL(10,2),

JoinDate DATE

);

**Sample Data:**

The following sample data can be used for testing:

Departments Sample Data:

INSERT INTO Departments

(DepartmentID, DepartmentName)

VALUES

(1, 'HR'),

(2, 'Finance'),

(3, 'IT'),

(4, 'Marketing');

Employees Sample Data:

INSERT INTO Employees

(EmployeeID, FirstName, LastName, DepartmentID, Salary,

JoinDate)

VALUES

(1, 'John', 'Doe', 1, 5000.00, '2020-01-15'),

(2, 'Jane', 'Smith', 2, 6000.00, '2019-03-22'),

(3, 'Michael', 'Johnson', 3, 7000.00, '2018-07-30'),

(4, 'Emily', 'Davis', 4, 5500.00, '2021-11-05');

**Exercise 1: Create a Stored Procedure**

Scenario:

Create a stored procedure to retrieve employee details by department.

To achieve this:

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

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

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

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;

### **SQL Code:**

IF OBJECT\_ID('Employees') IS NOT NULL DROP TABLE Employees;

IF OBJECT\_ID('Departments') IS NOT NULL DROP TABLE Departments;

CREATE TABLE Departments (

DepartmentID INT PRIMARY KEY,

DepartmentName VARCHAR(100)

);

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY IDENTITY(1,1),

FirstName VARCHAR(50),

LastName VARCHAR(50),

DepartmentID INT FOREIGN KEY REFERENCES Departments(DepartmentID),

Salary DECIMAL(10,2),

JoinDate DATE

);

INSERT INTO Departments (DepartmentID, DepartmentName) VALUES

(1, 'HR'),

(2, 'Finance'),

(3, 'IT'),

(4, 'Marketing');

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

('John', 'Doe', 1, 5000.00, '2020-01-15'),

('Jane', 'Smith', 2, 6000.00, '2019-03-22'),

('Michael', 'Johnson', 3, 7000.00, '2018-07-30'),

('Emily', 'Davis', 4, 5500.00, '2021-11-05');

IF OBJECT\_ID('sp\_GetEmployeesByDepartment') IS NOT NULL

DROP PROCEDURE sp\_GetEmployeesByDepartment;

GO

CREATE PROCEDURE sp\_GetEmployeesByDepartment

@DeptID INT

AS

BEGIN

SELECT

E.EmployeeID,

E.FirstName,

E.LastName,

D.DepartmentName,

E.Salary,

E.JoinDate

FROM Employees E

INNER JOIN Departments D ON E.DepartmentID = D.DepartmentID

WHERE E.DepartmentID = @DeptID;

END;

GO

IF OBJECT\_ID('sp\_InsertEmployee') IS NOT NULL

DROP PROCEDURE 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

EXEC sp\_InsertEmployee

@FirstName = 'Alex',

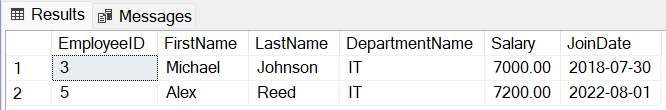
@LastName = 'Reed',

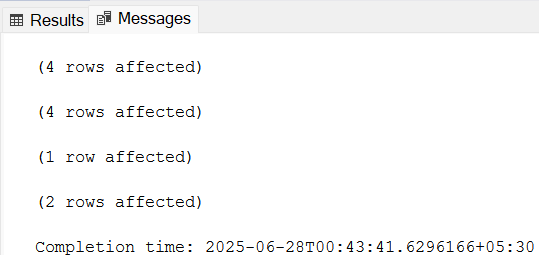
@DepartmentID = 3,

@Salary = 7200.00,

@JoinDate = '2022-08-01';

EXEC sp\_GetEmployeesByDepartment @DeptID = 3;

**OUTPUT:**



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

Scenario:

Create a stored procedure that returns the total number of employees in a department.

To achieve this:

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.

### **SQL Code:**

IF OBJECT\_ID('Employees') IS NOT NULL DROP TABLE Employees;

IF OBJECT\_ID('Departments') IS NOT NULL DROP TABLE Departments;

CREATE TABLE Departments (

DepartmentID INT PRIMARY KEY,

DepartmentName VARCHAR(100)

);

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

DepartmentID INT FOREIGN KEY REFERENCES Departments(DepartmentID),

Salary DECIMAL(10,2),

JoinDate DATE

);

INSERT INTO Departments (DepartmentID, DepartmentName) VALUES

(1, 'HR'),

(2, 'Finance'),

(3, 'IT'),

(4, 'Marketing');

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

(1, 'John', 'Doe', 1, 5000.00, '2020-01-15'),

(2, 'Jane', 'Smith', 2, 6000.00, '2019-03-22'),

(3, 'Michael', 'Johnson', 3, 7000.00, '2018-07-30'),

(4, 'Emily', 'Davis', 4, 5500.00, '2021-11-05');

IF OBJECT\_ID('sp\_CountEmployeesByDepartment') IS NOT NULL

DROP PROCEDURE sp\_CountEmployeesByDepartment;

GO

CREATE PROCEDURE sp\_CountEmployeesByDepartment

@DeptID INT

AS

BEGIN

SELECT COUNT(\*) AS TotalEmployees

FROM Employees

WHERE DepartmentID = @DeptID;

END;

GO

PRINT '--- Total Employees in HR (Dept 1) ---';

EXEC sp\_CountEmployeesByDepartment @DeptID = 1;

PRINT '--- Total Employees in IT (Dept 3) ---';

EXEC sp\_CountEmployeesByDepartment @DeptID = 3;

**OUTPUT:**