**Skill:-Advanced SQL**

**Filename:- 1. SQL Exercise - Advanced concepts**

**Name:-Exercise 1: Ranking and Window Functions**

**Code:-**

1. **Create your working database**

CREATE DATABASE OnlineRetailStore;

GO

1. **Create the necessary tables**

CREATE TABLE Products (

ProductID INT PRIMARY KEY,

ProductName NVARCHAR(100),

Category NVARCHAR(100),

Price DECIMAL(10, 2)

);

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY,

Name NVARCHAR(100),

Region NVARCHAR(100)

);

CREATE TABLE Orders (

OrderID INT PRIMARY KEY,

CustomerID INT,

OrderDate DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE OrderDetails (

OrderID INT,

ProductID INT,

Quantity INT,

FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),

FOREIGN KEY (ProductID) REFERENCES Products(ProductID)

);

CREATE TABLE StagingProducts (

ProductID INT PRIMARY KEY,

ProductName NVARCHAR(100),

Category NVARCHAR(100),

Price DECIMAL(10, 2)

);

1. **Insert some sample data**

INSERT INTO Products VALUES

(1, 'Laptop', 'Electronics', 800),

(2, 'Smartphone', 'Electronics', 600),

(3, 'Desk Chair', 'Furniture', 150),

(4, 'Table', 'Furniture', 300),

(5, 'Mouse', 'Electronics', 20);

INSERT INTO Customers VALUES

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

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

(3, 'Charlie', 'East');

INSERT INTO Orders VALUES

(101, 1, '2025-06-01'),

(102, 2, '2025-06-02'),

(103, 1, '2025-06-03'),

(104, 3, '2025-06-04');

INSERT INTO OrderDetails VALUES

(101, 1, 2),

(101, 5, 5),

(102, 2, 1),

(103, 3, 1),

(104, 4, 3);

**OUTPUT:-**

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

**Query**

SELECT

Category,

ProductName,

Price,

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

FROM Products;

****

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

**Query**

SELECT

Category,

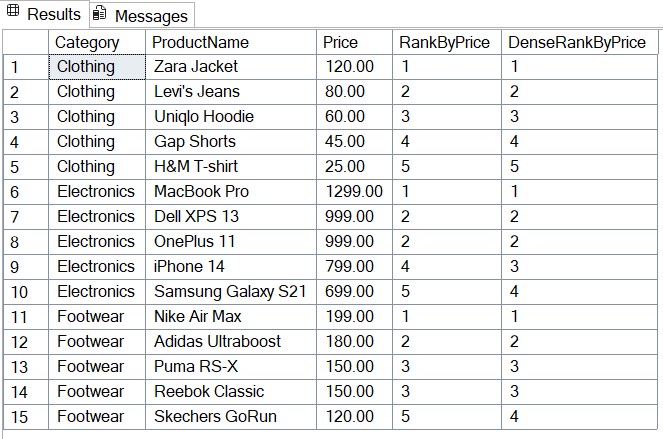
ProductName,

Price,

RANK() OVER (PARTITION BY Category ORDER BY Price DESC) AS RankByPrice,

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

FROM Products;

****

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

**Query**

WITH RankedProducts AS (

SELECT

Category,

ProductName,

Price,

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

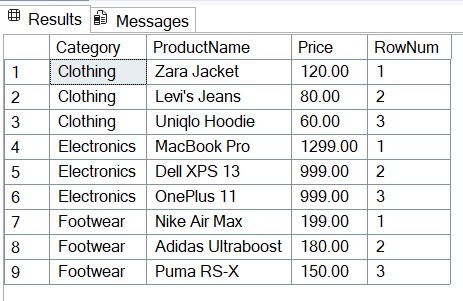
FROM Products

)

SELECT \*

FROM RankedProducts

WHERE RowNum <= 3;

****

**Filename:- 4. SQL Exercise - Stored procedure**

**Name:-Exercise 1: Create a Stored Procedure**

**Code:-**

**1: Create the Tables**

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

);

**2: Insert Sample Data**

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');

**3: Create Stored Procedure to Retrieve Employees by Department**

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;

**4: Create Stored Procedure to Insert a New Employee**

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;

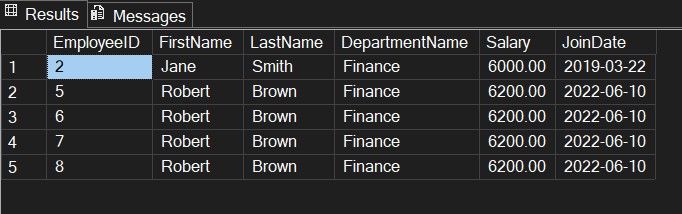
**OUTPUT:-**

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

Answer:-We create a procedure that takes a @DeptID parameter to filter employees by department.

**Query**

EXEC sp\_GetEmployeesByDepartment @DeptID = 2;

****

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

**Query**

DECLARE @DeptID INT;

SET @DeptID = 3;

SELECT

E.EmployeeID,

E.FirstName,

E.LastName,

D.DepartmentName,

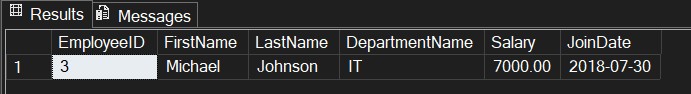
E.Salary,

E.JoinDate

FROM Employees E

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

WHERE E.DepartmentID = @DeptID;

****

**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;

**Query**

EXEC sp\_InsertEmployee

@FirstName = 'Alice',

@LastName = 'Walker',

@DepartmentID = 2,

@Salary = 6200.00,

@JoinDate = '2023-05-10';



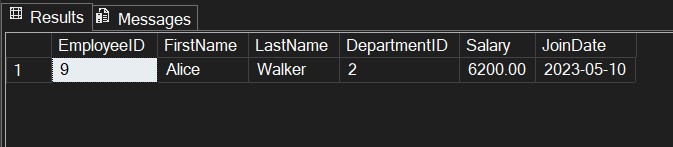
**To Check wheteher we have entered the data or not.**

**Query**

SELECT \*

FROM Employees

WHERE FirstName = 'Alice' AND LastName = 'Walker';



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

Goal: Create a stored procedure that returns the total number of employees in a

department.

Steps:

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.

**Query to Count Employees:-**

CREATE PROCEDURE sp\_GetEmployeeCountByDepartment

@DeptID INT

AS

BEGIN

SELECT

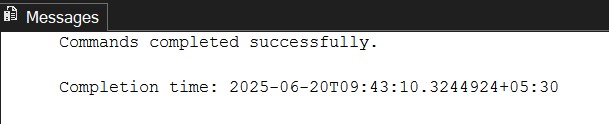
COUNT(\*) AS TotalEmployees

FROM Employees

WHERE DepartmentID = @DeptID;

END;

**Snap:-**

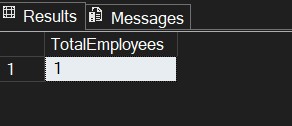


**OUTPUT:-**

**Query**

EXEC sp\_GetEmployeeCountByDepartment @DeptID = 3;

**Snap of Output:-**

****