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) Code-:**

CREATE DATABASE DeepSkillings;

GO

USE DeepSkillings;

CREATE TABLE Productss(

ProductID INT PRIMARY KEY,

ProductName VARCHAR(50),

Category VARCHAR(50),

Price INT

);

INSERT INTO Productss VALUES

(1, 'iPhone 13', 'Electronics', 80000),

(2, 'Samsung Galaxy', 'Electronics', 60000),

(3, 'MacBook Air', 'Electronics', 120000),

(4, 'Dell XPS', 'Electronics', 100000),

(5, 'Shirt', 'Clothing', 1500),

(6, 'Jacket', 'Clothing', 3000),

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

(8, 'Hoodie', 'Clothing', 3000);

SELECT

ProductName,

Category,

Price,

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

FROM Productss;

SELECT

ProductName,

Category,

Price,

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

FROM Productss;

SELECT

ProductName,

Category,

Price,

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

FROM Productss;

WITH RankedProducts AS (

SELECT

\*,

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

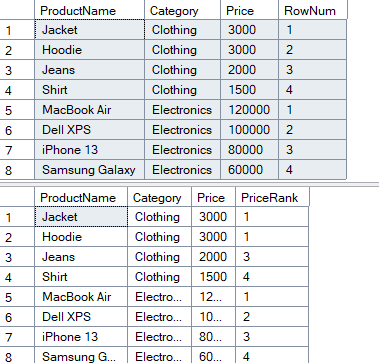
FROM Productss

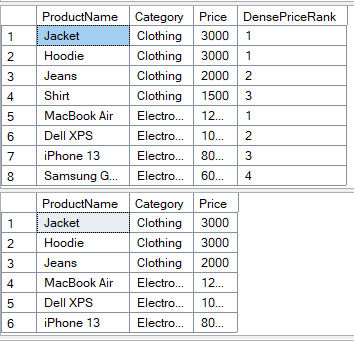
)

SELECT ProductName, Category, Price

FROM RankedProducts

WHERE RowNum <= 3;





**Exercise 2: Create a Stored Procedure**

Goal: Create a stored procedure to retrieve employee details by department.

Steps:

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

**ANS) Code:-**

USE EmployeeDB;

GO

-- Drop stored procedures

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

DROP PROCEDURE sp\_InsertEmployee;

GO

IF OBJECT\_ID('sp\_CountEmployeesByDept', 'P') IS NOT NULL

DROP PROCEDURE sp\_CountEmployeesByDept;

GO

-- Drop tables in the correct order (Employees first due to FK)

IF OBJECT\_ID('Employees', 'U') IS NOT NULL

DROP TABLE Employees;

GO

IF OBJECT\_ID('Departments', 'U') IS NOT NULL

DROP TABLE Departments;

GO

-- Departments Table

CREATE TABLE Departments (

DepartmentID INT PRIMARY KEY,

DepartmentName VARCHAR(100)

);

GO

-- Employees Table

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

);

GO

-- Insert 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');

GO

-- Insert procedure

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

-- Count procedure

CREATE PROCEDURE sp\_CountEmployeesByDept

@DepartmentID INT

AS

BEGIN

SELECT COUNT(\*) AS TotalEmployees

FROM Employees

WHERE DepartmentID = @DepartmentID;

END;

GO

-- Insert a new employee

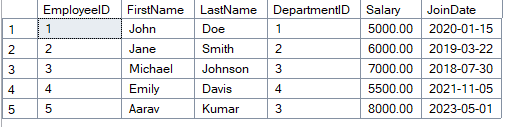
EXEC sp\_InsertEmployee 'Aarav', 'Kumar', 3, 8000.00, '2023-05-01';

-- View all employees

SELECT \* FROM Employees;

-- Count employees in IT (Department 3)

EXEC sp\_CountEmployeesByDept 3;



**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.

**Ans) Code:-**

CREATE PROCEDURE sp\_CountEmployeesByDept

@DepartmentID INT

AS

BEGIN

SELECT COUNT(\*) AS TotalEmployees

FROM Employees

WHERE DepartmentID = @DepartmentID;

END;

GO

-- Insert a new employee

EXEC sp\_InsertEmployee 'Aarav', 'Kumar', 3, 8000.00, '2023-05-01';

-- View all employees

SELECT \* FROM Employees;

-- Count employees in IT (Department 3)

EXEC sp\_CountEmployeesByDept 3;

