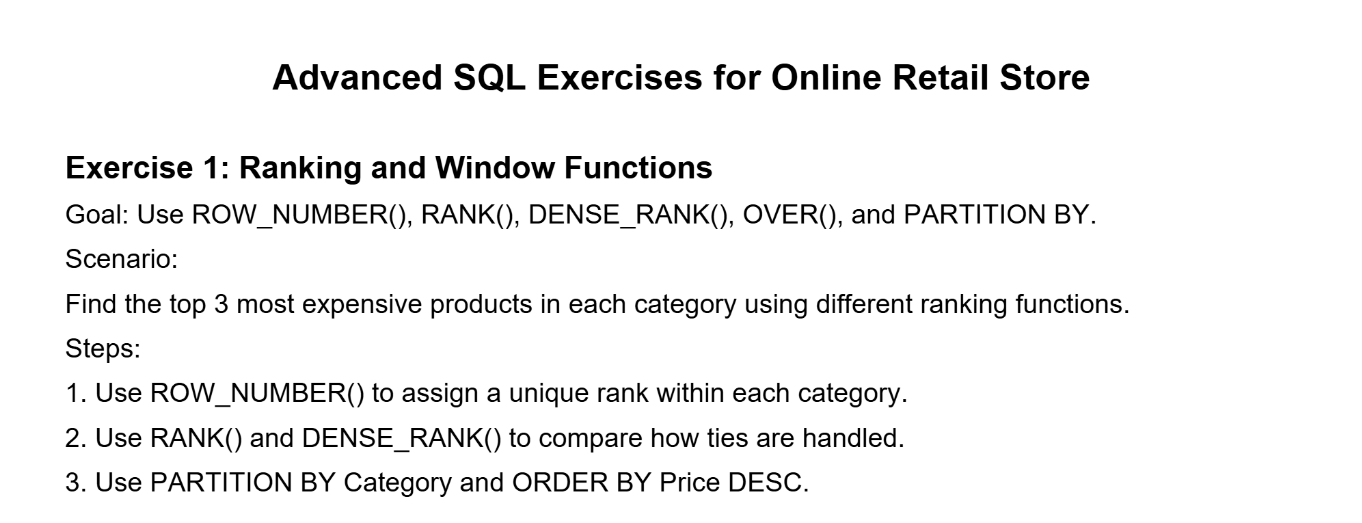
**Cognizant - DN 4.0 I Deep Skilling**

**WEEK-2**

**Advanced SQL**



**SOLUTION :**

**CODE -:**

Use RetaildetailsDB;

CREATE TABLE Products (

ProductID INT PRIMARY KEY,

ProductName VARCHAR(100),

Category VARCHAR(50),

Price DECIMAL(10, 2)

);

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

(1, 'Wireless Mouse', 'Electronics', 25.99),

(2, 'Gaming Keyboard', 'Electronics', 89.99),

(3, 'Monitor 24 inch', 'Electronics', 199.99),

(4, 'Bluetooth Speaker', 'Electronics', 49.99),

(5, 'Smartphone', 'Electronics', 599.99),

(6, 'Notebook A5', 'Stationery', 2.99),

(7, 'Marker Pen', 'Stationery', 1.49),

(8, 'Stapler', 'Stationery', 5.49),

(9, 'Desk Organizer', 'Stationery', 12.99),

(10, 'Whiteboard', 'Stationery', 25.00),

(11, 'T-Shirt', 'Apparel', 15.00),

(12, 'Jeans', 'Apparel', 39.99),

(13, 'Jacket', 'Apparel', 79.99),

(14, 'Sneakers', 'Apparel', 59.99),

(15, 'Cap', 'Apparel', 9.99);

SELECT \* from Products;

-- Get top 3 most expensive products in each category

-- and compare different ranking functions

SELECT \*

FROM (

SELECT

ProductID,

ProductName,

Category,

Price,

-- Step 1: Assign unique row number within each category

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

-- Step 2: Rank with gaps for ties

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

-- Step 2: Rank without gaps for ties

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

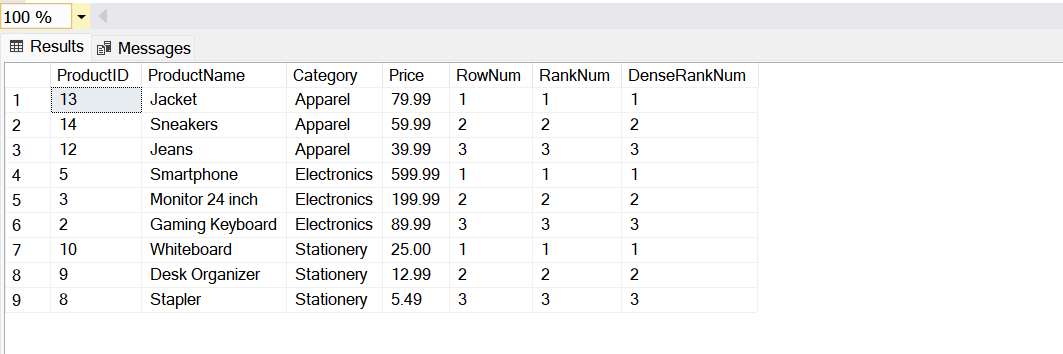
FROM Products

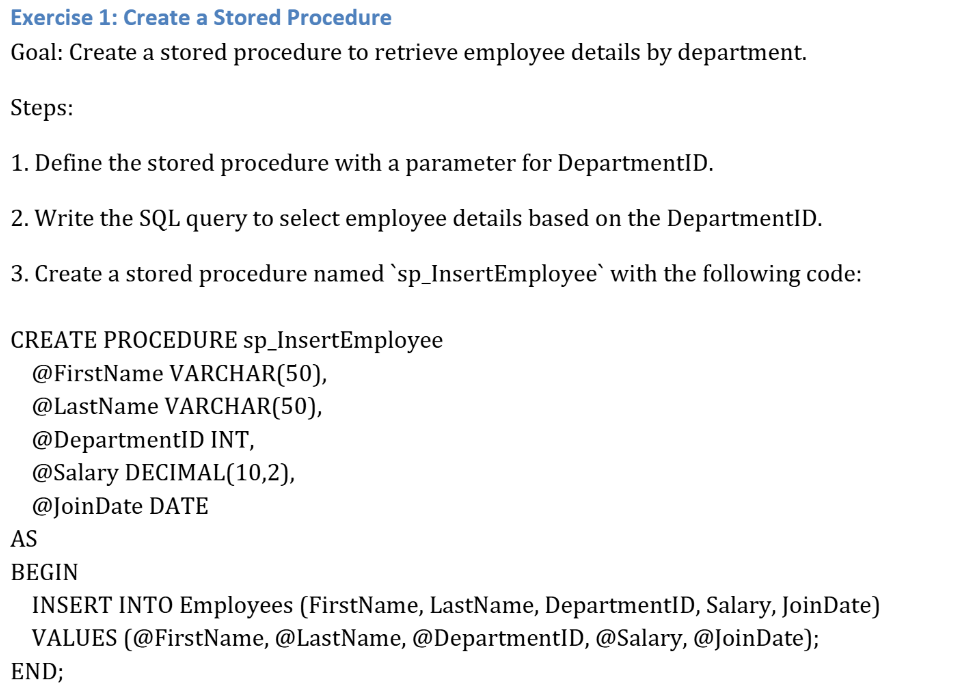
) AS RankedProducts

-- Step 3: Only show top 3 per category using RowNum (strict cutoff)

WHERE RowNum <= 3;

**OUTPUT -:**





**SOLUTION :**

**CODE -:**

Use EmployeesDB;

CREATE TABLE Departments(

DepartmentID int primary key,

DepartmentName varchar(20)

);

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

DepartmentID INT,

Salary DECIMAL(10, 2),

JoinDate DATE,

FOREIGN KEY (DepartmentID) REFERENCES Departments(DepartmentID)

);

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');

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

VALUES

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

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

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

DROP PROCEDURE sp\_InsertEmployee;

CREATE PROCEDURE sp\_InsertEmployee

@DepartmentID 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

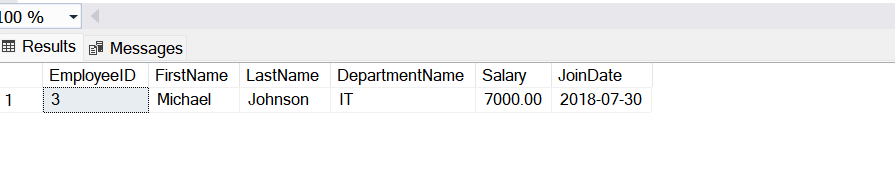
WHERE E.DepartmentID = @DepartmentID;

END;

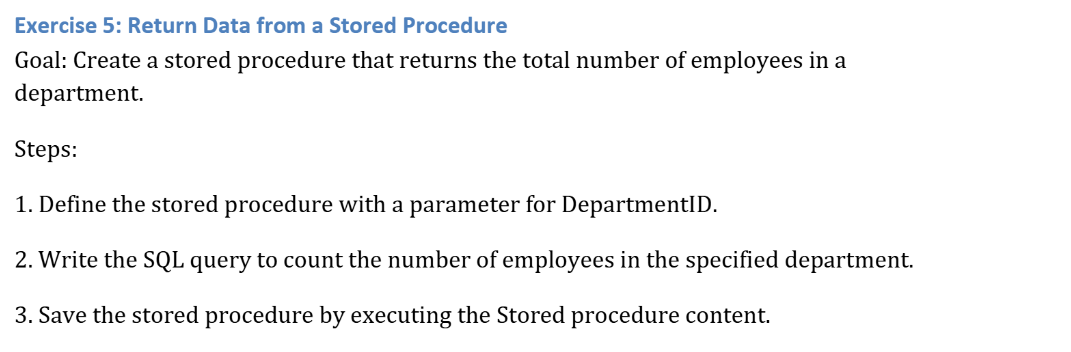
-- Example: Get all employees from DepartmentID = 3 (IT)

EXEC sp\_InsertEmployee @DepartmentID = 3;

**OUTPUT -:**



--------------------------------------------------------------------------------------------------------------------------------------



**SOLUTION :**

**CODE -:**

-- Step 1: Drop the procedure if it already exists

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

DROP PROCEDURE sp\_CountEmployeesByDepartment;

-- Step 2: Create the stored procedure

CREATE PROCEDURE sp\_CountEmployeesByDepartment

@DepartmentID INT

AS

BEGIN

SELECT

D.DepartmentName,

COUNT(E.EmployeeID) AS TotalEmployees

FROM Departments D

LEFT JOIN Employees E ON D.DepartmentID = E.DepartmentID

WHERE D.DepartmentID = @DepartmentID

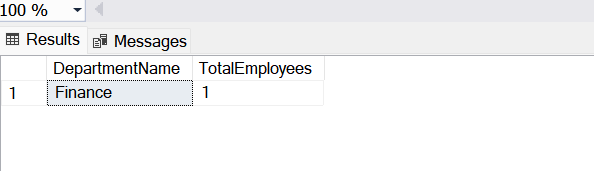
GROUP BY D.DepartmentName;

END;

-- Example: Get employee count in DepartmentID = 2 (Finance)

EXEC sp\_CountEmployeesByDepartment @DepartmentID = 2;

**OUTPUT -:**



------------------------------------------------------------------------------------------------------------------------------------