1. **SQL Exercise - Advanced concepts**

**Advanced SQL Exercises for Online Retail Store**

**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.

**ANSWER :**

--Create the Products table

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, 'OnePlus 10R', 'Electronics', 999.99),

(4, 'Dell XPS', 'Electronics', 1899.99),

(5, 'Bluetooth Speaker', 'Electronics', 199.99),

(6, 'Sofa Set', 'Furniture', 1499.99),

(9, 'Bed', 'Furniture', 1599.99),

(10, 'Office Chair', 'Furniture', 399.99),

(11, 'T-Shirt', 'Clothing', 19.99),

(12, 'Jeans', 'Clothing', 49.99),

--Ranking using ROW\_NUMBER

SELECT

Category,

ProductName,

Price,

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

FROM Products;

--Ranking using RANK

SELECT

Category,

ProductName,

Price,

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

FROM Products;

-- Ranking using DENSE\_RANK

SELECT

Category,

ProductName,

Price,

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

FROM Products;

--Top 3 using ROW\_NUMBER

SELECT \*

FROM (

SELECT

Category,

ProductName,

Price,

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

FROM Products

) AS Ranked

WHERE RowNum <= 3;

-- Top 3 using RANK

SELECT \*

FROM (

SELECT

Category,

ProductName,

Price,

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

FROM Products

) AS Ranked

WHERE RankNum <= 3;

-- Top 3 using DENSE\_RANK

SELECT \*

FROM (

SELECT

Category,

ProductName,

Price,

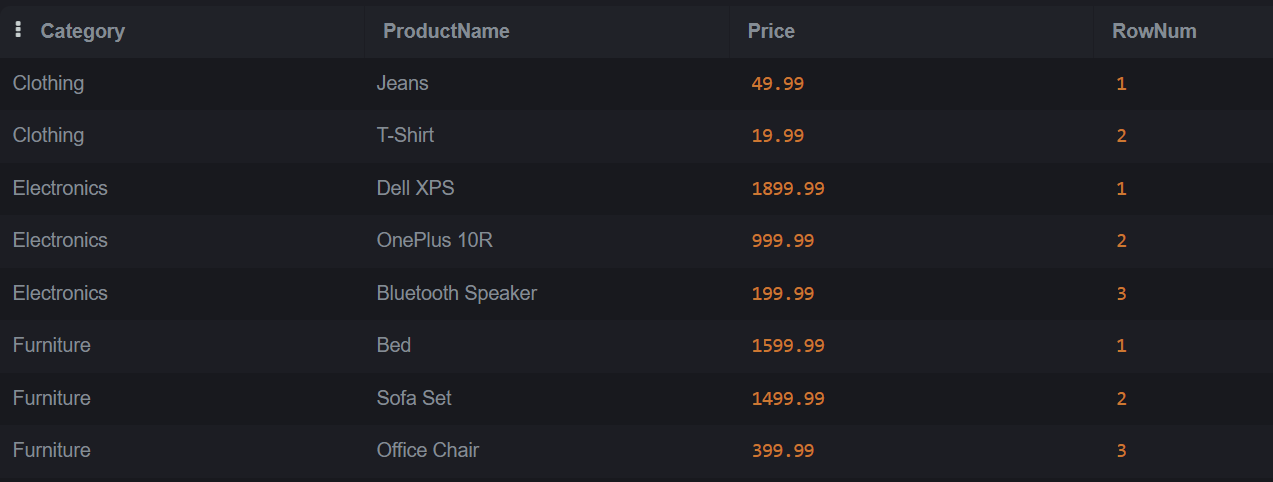
DENSE\_RANK() OVER (PARTITION BY Category ORDER BY Price DESC) AS DenseRankNum

FROM Products

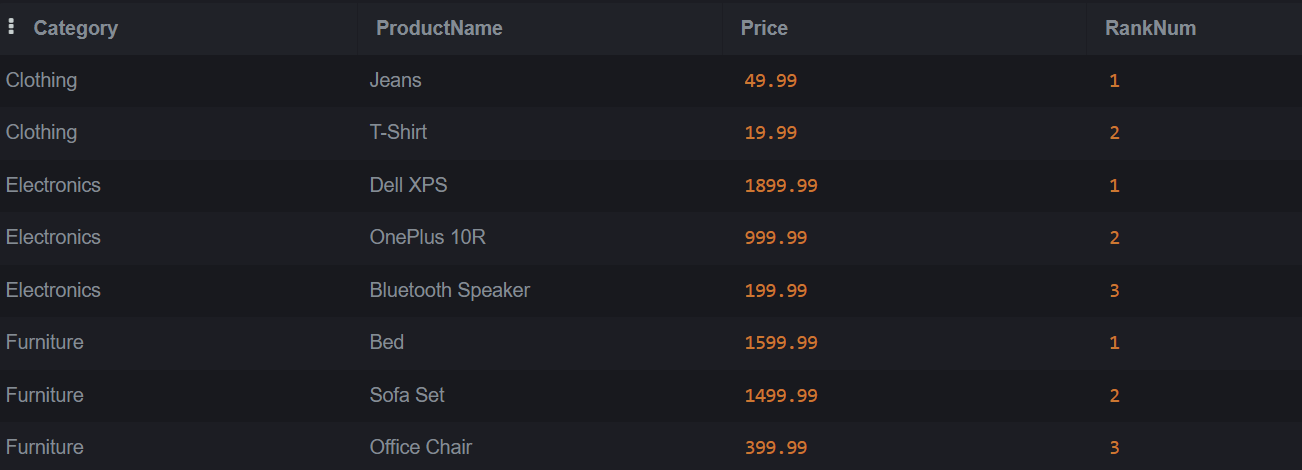
) AS Ranked

WHERE DenseRankNum <= 3;

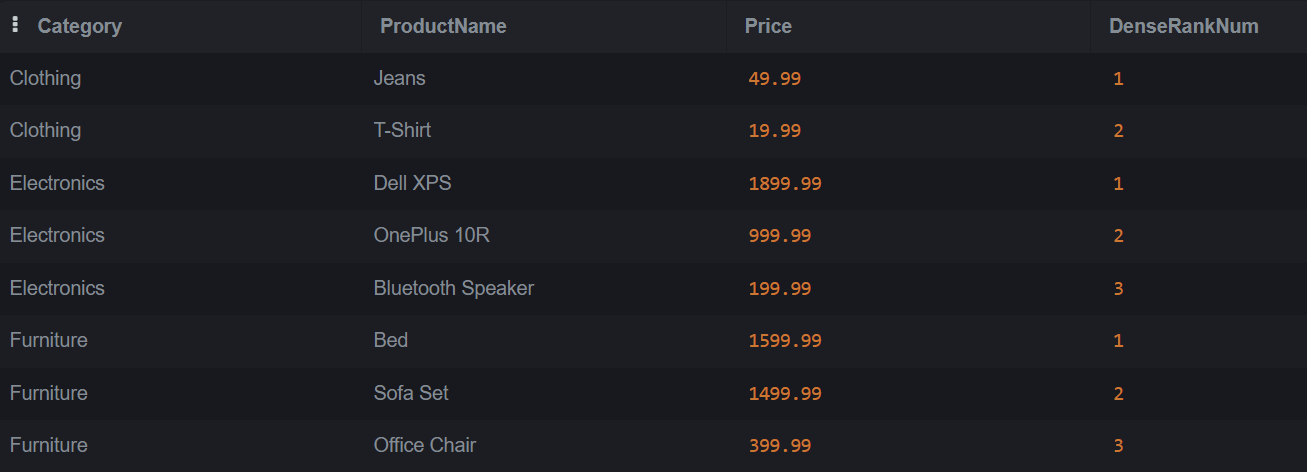
**OUTPUT 1 :**

****

**OUTPUT 2 :**

****

**OUTPUT 3 :**

****

1. **SQL Exercise - Stored Procedure**

**Exercise 1: 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;

**ANSWER :**

-- Create Departments table

CREATE TABLE Departments (

DepartmentID INT PRIMARY KEY,

DepartmentName VARCHAR(100)

);

-- Create Employees table

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY IDENTITY(1,1), -- IDENTITY for auto-increment (SQL Server)

FirstName VARCHAR(50),

LastName VARCHAR(50),

DepartmentID INT FOREIGN KEY REFERENCES Departments(DepartmentID),

Salary DECIMAL(10,2),

JoinDate DATE

);

-- Insert departments

INSERT INTO Departments (DepartmentID, DepartmentName) VALUES

(1, 'HR'),

(2, 'Finance'),

(3, 'IT'),

(4, 'Marketing');

-- Insert employees

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');

-- Procedure to get employees of a specific 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;

-- 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

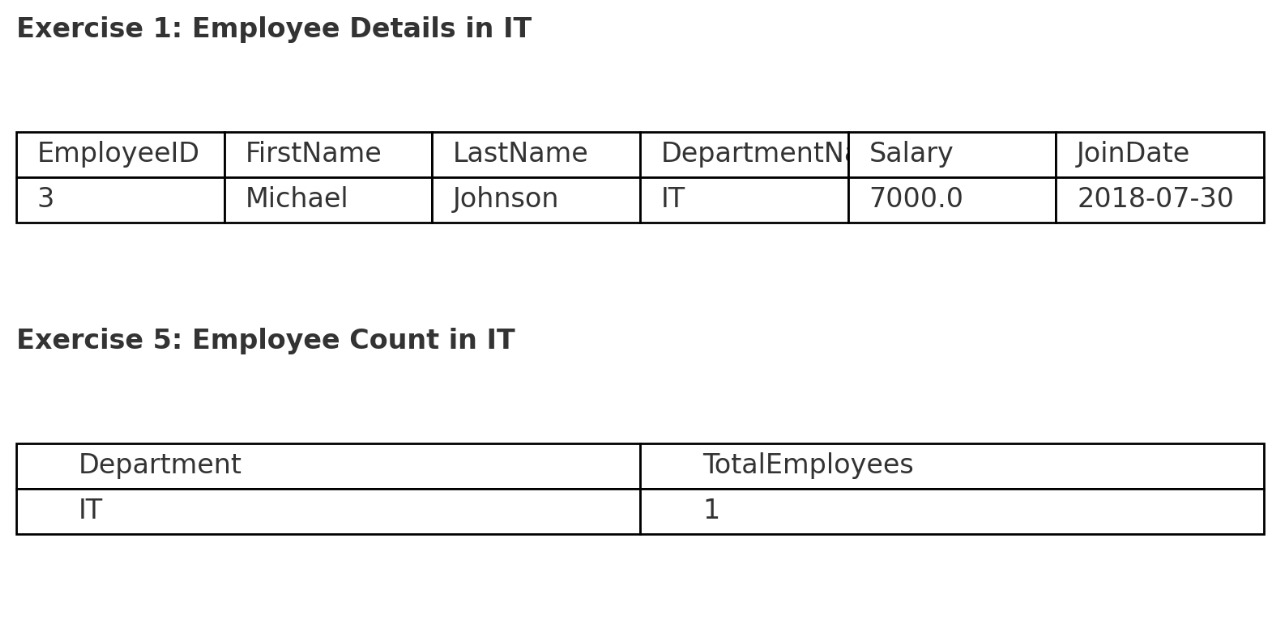
BEGIN

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

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

END;

**OUTPUT :**

****

**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.

**ANSWER :**

-- Procedure to count employees in a department

CREATE PROCEDURE sp\_CountEmployeesByDepartment

@DeptID INT

AS

BEGIN

SELECT

COUNT(\*) AS TotalEmployees

FROM Employees

WHERE DepartmentID = @DeptID;

END;

**OUTPUT :**

