Exercise 1: Ranking and Window Functions

-- 1: Create a new database

CREATE DATABASE RetailStore;

GO

-- 2: Use the new database

USE RetailStore;

GO

-- 3: Create the Products table

CREATE TABLE Products (

ProductID INT PRIMARY KEY,

ProductName VARCHAR(100),

Category VARCHAR(50),

Price DECIMAL(10,2)

);

GO

-- 4: Insert sample data

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

VALUES

(1, 'Laptop', 'Electronics', 75000),

(2, 'Headphones', 'Electronics', 2500),

(3, 'Smartphone', 'Electronics', 60000),

(4, 'TV', 'Electronics', 40000),

(5, 'Refrigerator', 'Appliances', 30000),

(6, 'Microwave', 'Appliances', 12000),

(7, 'Washing Machine', 'Appliances', 35000),

(8, 'Blender', 'Appliances', 4000),

(9, 'Shirt', 'Clothing', 1500),

(10, 'Jacket', 'Clothing', 4000),

(11, 'Jeans', 'Clothing', 2500),

(12, 'Shoes', 'Clothing', 3000);

GO

-- 5: Use ROW\_NUMBER to get top 3 expensive products per category

SELECT 'ROW\_NUMBER' AS RankType, \*

FROM (

SELECT \*,

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

FROM Products

) AS Ranked

WHERE RankPos <= 3;

GO

-- 6: Use RANK to get top 3 expensive products per category (allows ties)

SELECT 'RANK' AS RankType, \*

FROM (

SELECT \*,

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

FROM Products

) AS Ranked

WHERE RankPos <= 3;

GO

-- Step 7: Use DENSE\_RANK to get top 3 expensive products per category (no gaps)

SELECT 'DENSE\_RANK' AS RankType, \*

FROM (

SELECT \*,

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

FROM Products

) AS Ranked

WHERE RankPos <= 3;

GO

Exercise 1: Create a Stored Procedure

-- 1: Create the database

CREATE DATABASE EmployeeDB;

GO

-- 2: Use the database

USE EmployeeDB;

GO

-- 3: Create Departments table

CREATE TABLE Departments (

DepartmentID INT PRIMARY KEY,

DepartmentName VARCHAR(100)

);

GO

-- 4: Create 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

-- 5: Insert sample data into Departments

INSERT INTO Departments (DepartmentID, DepartmentName) VALUES

(1, 'HR'),

(2, 'Finance'),

(3, 'IT'),

(4, 'Marketing');

GO

-- 6: Insert sample data into 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');

GO

-- 7: Create stored procedure to get 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;

GO

-- 8: Create stored procedure to insert an 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;

GO  
  
Exercise 5: Return Data from a Stored Procedure  
  
USE EmployeeDB;

GO

-- 1: Create stored procedure to count employees in a department

CREATE PROCEDURE sp\_CountEmployeesByDepartment

@DeptID 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 = @DeptID

GROUP BY D.DepartmentName;

END;

GO