**SQL QUERIES**

**1. SQL Exercise - Advanced concepts**

**Exercise 1: Ranking and Window Functions**

CREATE TABLE Products (

ProductID INT PRIMARY KEY,

ProductName VARCHAR(255),

Category VARCHAR(100),

Price DECIMAL(10, 2)

);

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

(1, 'Laptop Pro', 'Electronics', 1500.00),

(2, 'Gaming PC', 'Electronics', 2000.00),

(3, 'Smartphone X', 'Electronics', 1200.00),

(4, 'Tablet Z', 'Electronics', 800.00),

(5, 'Smart TV 4K', 'Electronics', 1200.00), -- Tie with Smartphone X

(6, 'Office Chair Ergo', 'Furniture', 300.00),

(7, 'Executive Desk', 'Furniture', 700.00),

(8, 'Bookshelf Modern', 'Furniture', 250.00),

(9, 'Coffee Table Glass', 'Furniture', 300.00), -- Tie with Office Chair Ergo

(10, 'Sofa Deluxe', 'Furniture', 1000.00),

(11, 'Running Shoes', 'Apparel', 120.00),

(12, 'Sports Jacket', 'Apparel', 150.00),

(13, 'Yoga Pants', 'Apparel', 80.00),

(14, 'Gym Bag', 'Apparel', 70.00),

(15, 'Hiking Boots', 'Apparel', 150.00); -- Tie with Sports Jacket

SELECT

ProductID,

ProductName,

Category,

Price,

RowNumberRank

FROM (

SELECT

ProductID,

ProductName,

Category,

Price,

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

FROM

Products

) AS RankedProducts

WHERE

RowNumberRank <= 3

ORDER BY

Category, RowNumberRank;

-- 2. Using RANK() to find the top 3 most expensive products in each category (handles ties)

SELECT

ProductID,

ProductName,

Category,

Price,

RankValue

FROM (

SELECT

ProductID,

ProductName,

Category,

Price,

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

FROM

Products

) AS RankedProducts

WHERE

RankValue <= 3

ORDER BY

Category, RankValue;

-- 3. Using DENSE\_RANK() to find the top 3 most expensive products in each category (handles ties without gaps)

SELECT

ProductID,

ProductName,

Category,

Price,

DenseRankValue

FROM (

SELECT

ProductID,

ProductName,

Category,

Price,

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

FROM

Products

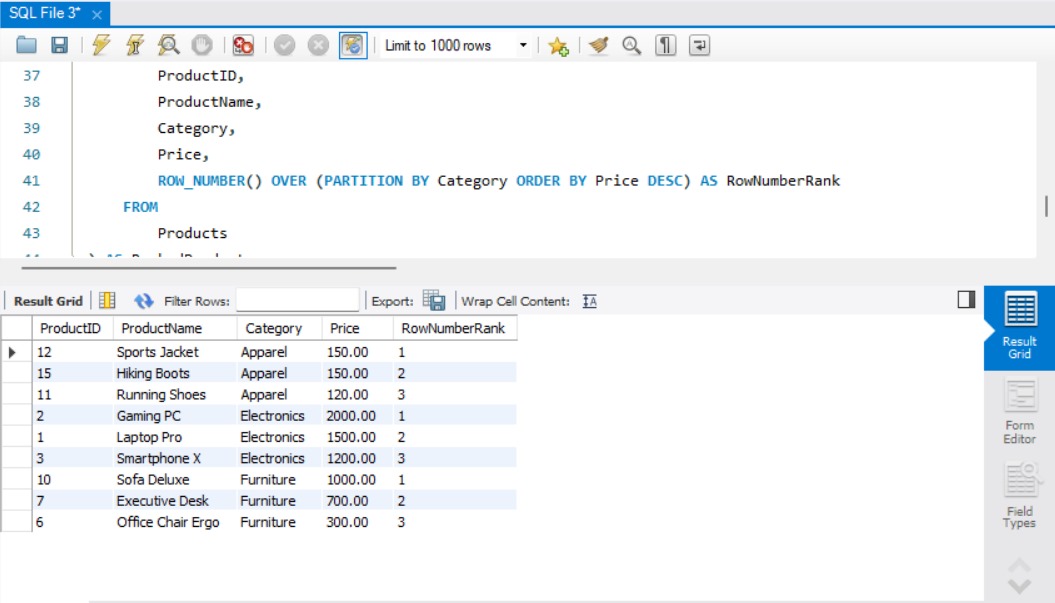
) AS RankedProducts

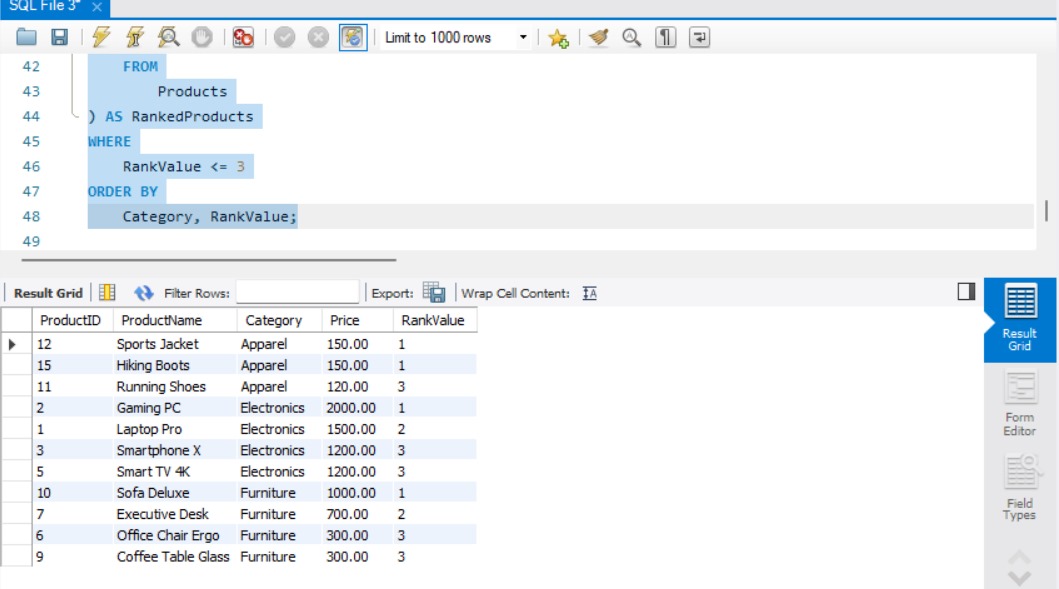
WHERE

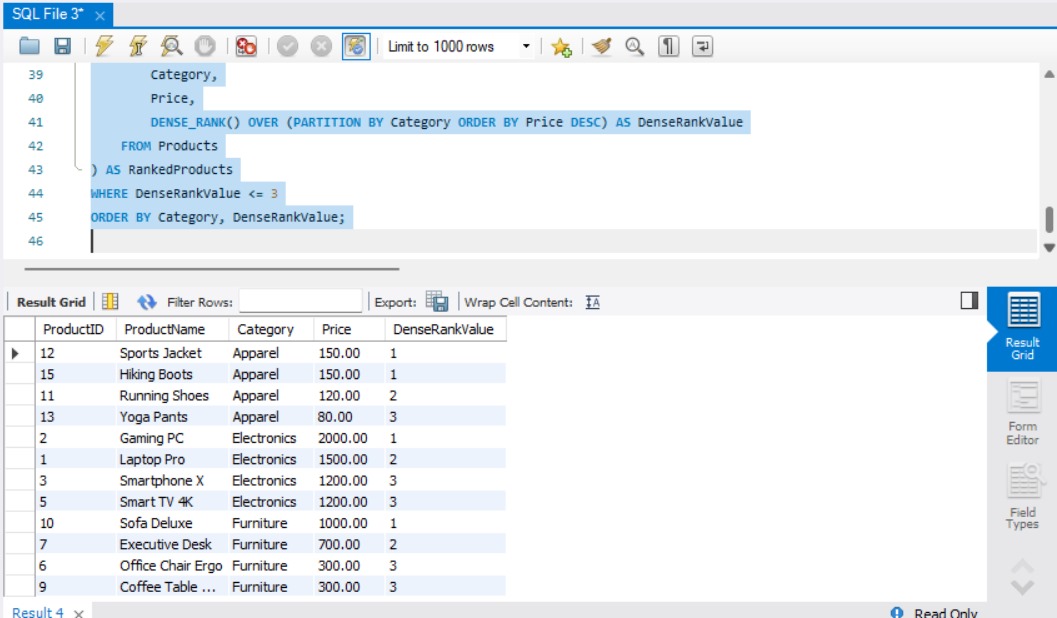
DenseRankValue <= 3

ORDER BY

Category, DenseRankValue;







**4. SQL Exercise - Stored procedure**

**Exercise 1: Create a Stored Procedure**

-- Create Departments table

CREATE TABLE Departments (

DepartmentID INT PRIMARY KEY,

DepartmentName VARCHAR(100)

);

-- Create Employees table

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

INSERT INTO Departments (DepartmentID, DepartmentName) VALUES

(1, 'HR'),

(2, 'Finance'),

(3, 'IT'),

(4, 'Marketing');

-- Insert into Employees

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'),

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

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

select \* from departments;

DELIMITER $$

CREATE PROCEDURE sp\_InsertEmployee(

IN p\_FirstName VARCHAR(50),

IN p\_LastName VARCHAR(50),

IN p\_DepartmentID INT,

IN p\_Salary DECIMAL(10,2),

IN p\_JoinDate DATE

)

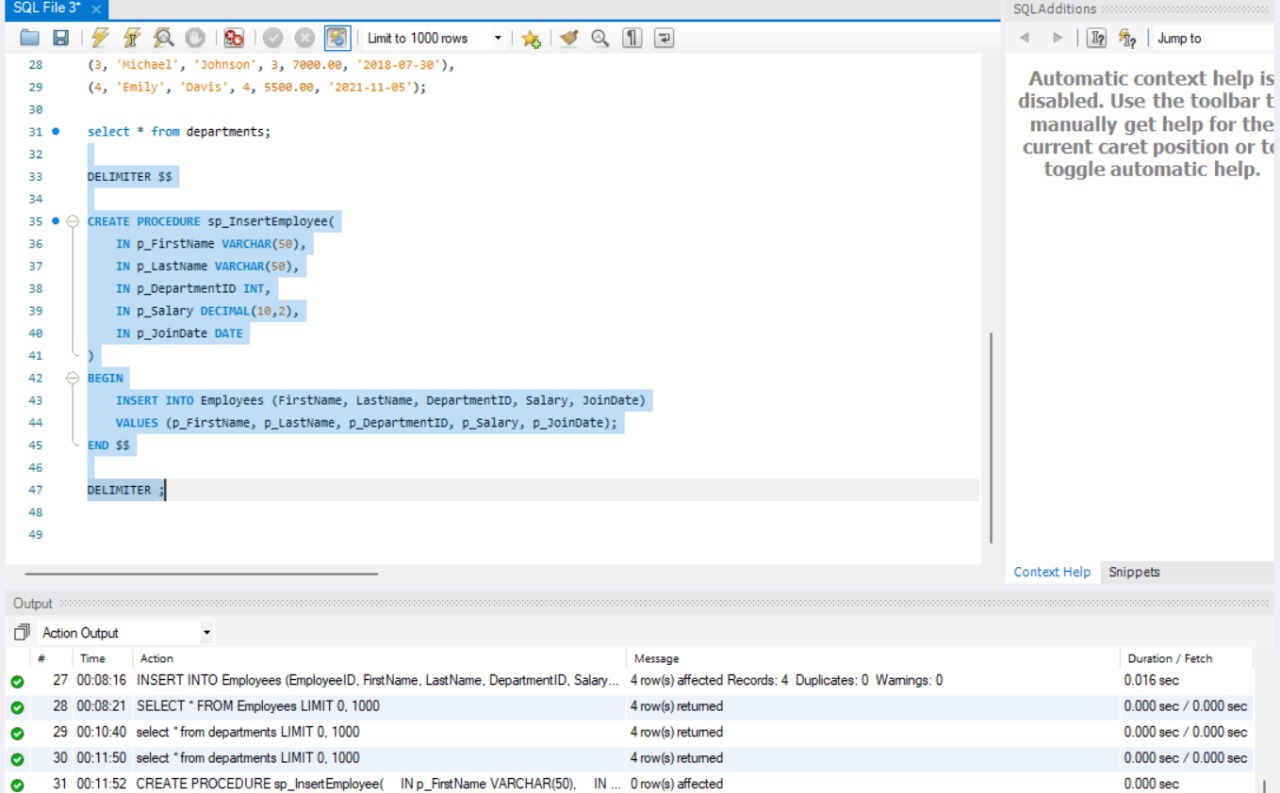
BEGIN

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

VALUES (p\_FirstName, p\_LastName, p\_DepartmentID, p\_Salary, p\_JoinDate);

END $$

DELIMITER ;



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

-- Create Departments table

CREATE TABLE Departments (

DepartmentID INT PRIMARY KEY,

DepartmentName VARCHAR(100)

);

-- Create Employees table

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

INSERT INTO Departments (DepartmentID, DepartmentName) VALUES

(1, 'HR'),

(2, 'Finance'),

(3, 'IT'),

(4, 'Marketing');

-- Insert into Employees

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'),

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

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

select \* from departments;

DELIMITER $$

CREATE PROCEDURE sp\_GetEmployeeCountByDept(

IN p\_DepartmentID INT

)

BEGIN

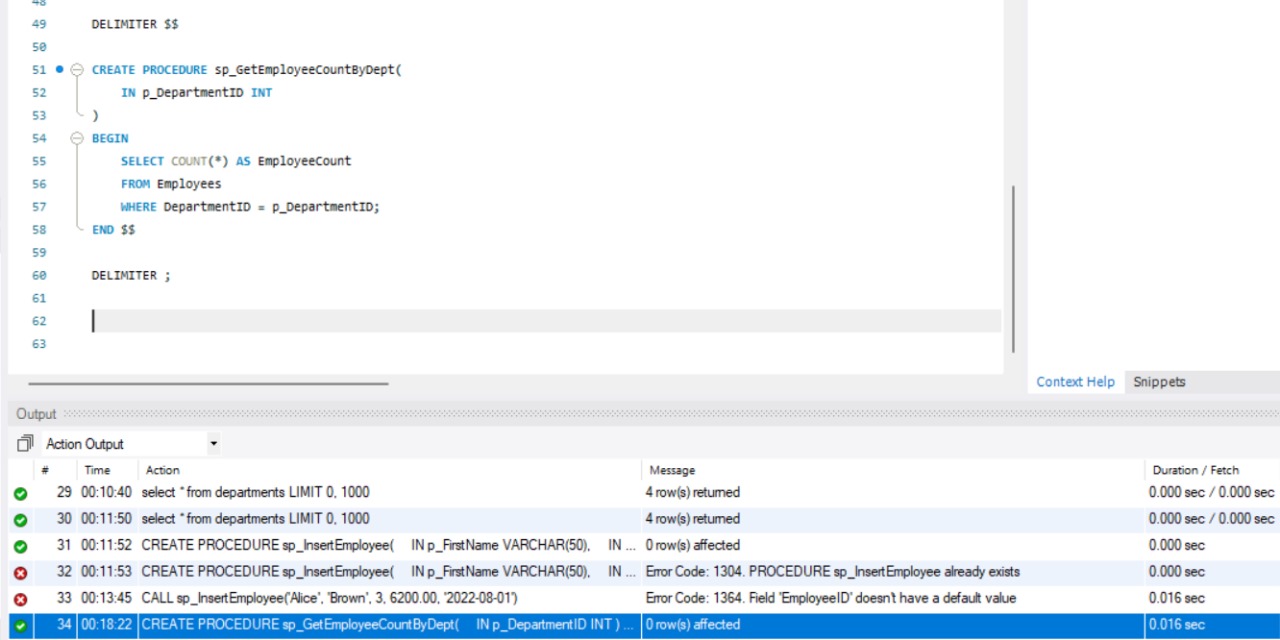
SELECT COUNT(\*) AS EmployeeCount

FROM Employees

WHERE DepartmentID = p\_DepartmentID;

END $$

DELIMITER ;



**5. SQL Exercise - Functions**

-- Step 1: Select or Create Database

CREATE DATABASE IF NOT EXISTS companydb;

USE companydb;

-- Step 2: Drop Tables if They Exist

DROP TABLE IF EXISTS Employees;

DROP TABLE IF EXISTS Departments;

-- Step 3: Create Departments Table

CREATE TABLE Departments (

DepartmentID INT PRIMARY KEY,

DepartmentName VARCHAR(100)

);

-- Step 4: Create Employees Table

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)

);

-- Step 5: Insert Sample Data

INSERT INTO Departments (DepartmentID, DepartmentName) VALUES

(1, 'HR'), (2, 'IT'), (3, 'Finance');

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'),

(3, 'Bob', 'Johnson', 3, 5500.00, '2021-07-01');

-- Step 6: Drop Function if Exists

DROP FUNCTION IF EXISTS fn\_CalculateAnnualSalary;

-- Step 7: Create Scalar Function

DELIMITER //

CREATE FUNCTION fn\_CalculateAnnualSalary(empId INT)

RETURNS DECIMAL(10,2)

DETERMINISTIC

BEGIN

DECLARE annualSalary DECIMAL(10,2);

SELECT Salary \* 12 INTO annualSalary

FROM Employees

WHERE EmployeeID = empId;

RETURN annualSalary;

END;

DELIMITER ;

-- Step 8: Execute Function

SELECT fn\_CalculateAnnualSalary(1) AS AnnualSalary;

