**WEEK 2 : Advanced SQL**

**Exercise 1: Ranking and Window Functions**

**SOURCE CODE**

**Table.sql**

CREATE TABLE Products (

    ProductID INT PRIMARY KEY,

    ProductName VARCHAR(100),

    Category VARCHAR(50),

    Price DECIMAL(10,2)

);

**Rownum.sql**

SELECT \*

FROM (

    SELECT

        ProductID,

        ProductName,

        Category,

        Price,

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

    FROM Products

) AS RankedProducts

WHERE RowNum <= 3;

**Rank.sql**

SELECT \*

FROM (

    SELECT

        ProductID,

        ProductName,

        Category,

        Price,

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

    FROM Products

) AS RankedProducts

WHERE RankNum <= 3;

**Denserank.sql**

SELECT \*

FROM (

    SELECT

        ProductID,

        ProductName,

        Category,

        Price,

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

    FROM Products

) AS RankedProducts

WHERE DenseRankNum <= 3;

**Insert.sql**

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

VALUES

(1, 'iPhone 13', 'Electronics', 999.99),

(2, 'Samsung Galaxy', 'Electronics', 899.99),

(3, 'Sony Headphones', 'Electronics', 199.99),

(4, 'LG TV', 'Electronics', 999.99),

(5, 'Nike Shoes', 'Clothing', 149.99),

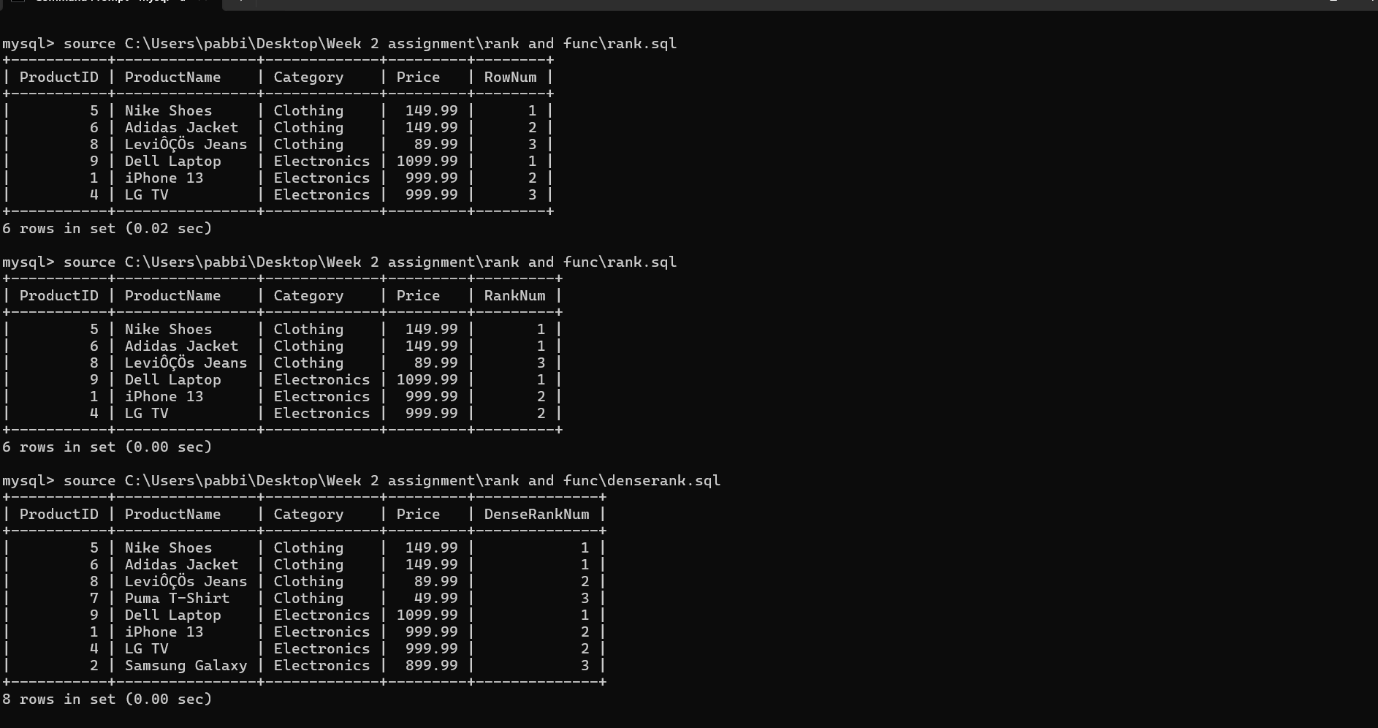
(6, 'Adidas Jacket', 'Clothing', 149.99),

(7, 'Puma T-Shirt', 'Clothing', 49.99),

(8, 'Levi’s Jeans', 'Clothing', 89.99),

(9, 'Dell Laptop', 'Electronics', 1099.99),

(10, 'Boat Earbuds', 'Electronics', 59.99);



**Exercise 2: Create a Stored Procedure**

**Source code**

**Table.sql**

CREATE TABLE Departments (

    DepartmentID INT PRIMARY KEY,

    DepartmentName VARCHAR(100)

);

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)

);

**sp\_InsertEmployee.sql**

DELIMITER $$

CREATE PROCEDURE sp\_InsertEmployee (

    IN FirstName VARCHAR(50),

    IN LastName VARCHAR(50),

    IN DepartmentID INT,

    IN Salary DECIMAL(10,2),

    IN JoinDate DATE

)

BEGIN

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

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

END$$

DELIMITER ;

**Inset.sql**

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'),

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

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

**Employees.sql**

CREATE TABLE Employees (

    EmployeeID INT AUTO\_INCREMENT PRIMARY KEY,

    FirstName VARCHAR(50),

    LastName VARCHAR(50),

    DepartmentID INT,

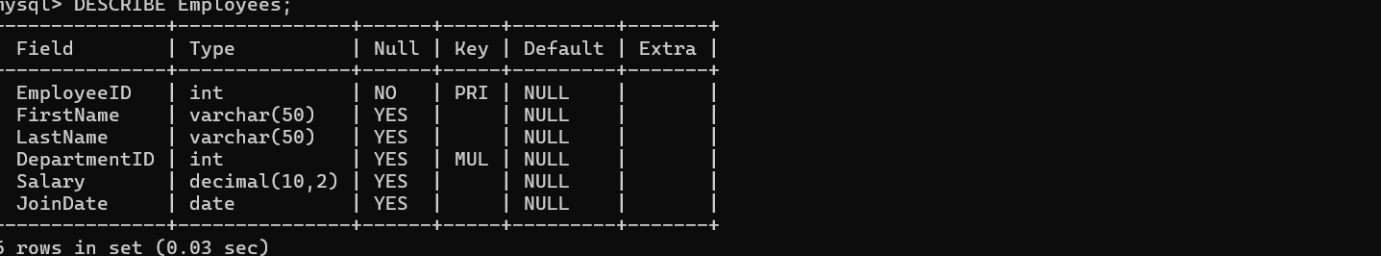
    Salary DECIMAL(10,2),

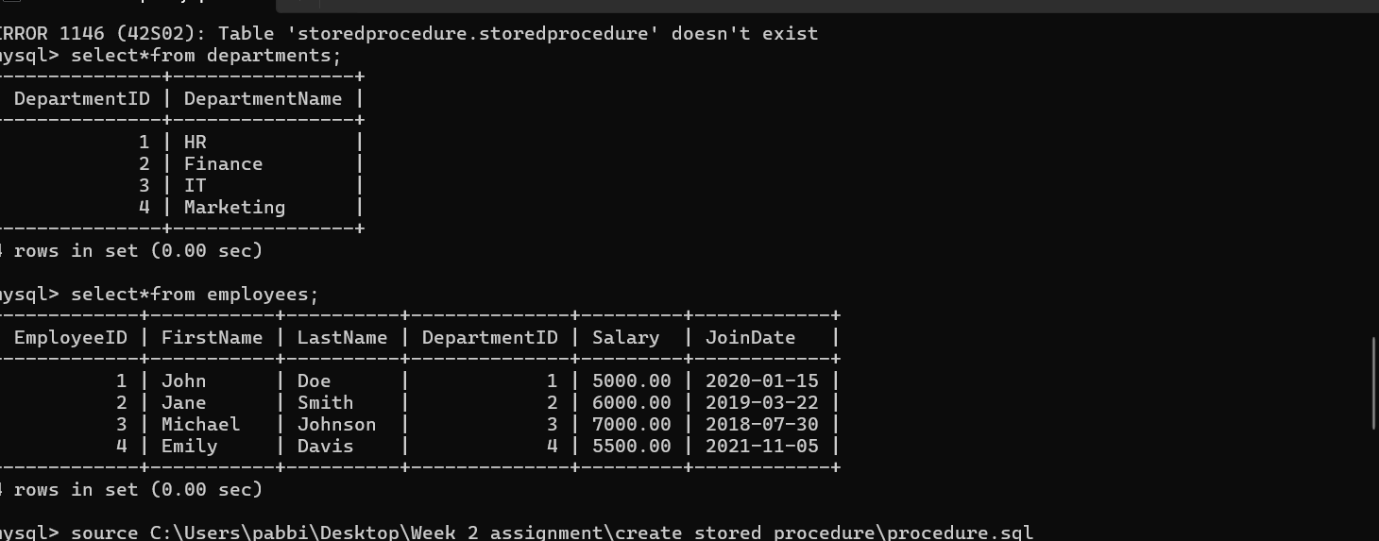
    JoinDate DATE

);

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

VALUES ('John', 'Doe', 2, 50000.00, '2025-06-27');





**Exercise 3: Return Data from a Scalar Function**

**Source code**

**Index.sql**

-- Step 1: Create Employees Table

CREATE TABLE Employees (

    EmployeeID INT PRIMARY KEY,

    EmployeeName VARCHAR(100),

    MonthlySalary DECIMAL(10, 2)

);

-- Step 2: Insert Sample Data

INSERT INTO Employees (EmployeeID, EmployeeName, MonthlySalary) VALUES

(1, 'John Doe', 5000.00),

(2, 'Jane Smith', 6500.00),

(3, 'Alice Johnson', 7000.00);

-- Step 3: Create Scalar Function (MySQL version)

DELIMITER $$

CREATE FUNCTION fn\_CalculateAnnualSalary(EmpID INT)

RETURNS DECIMAL(10,2)

DETERMINISTIC

BEGIN

    DECLARE AnnualSalary DECIMAL(10,2);

    SELECT MonthlySalary \* 12 INTO AnnualSalary

    FROM Employees

    WHERE EmployeeID = EmpID;

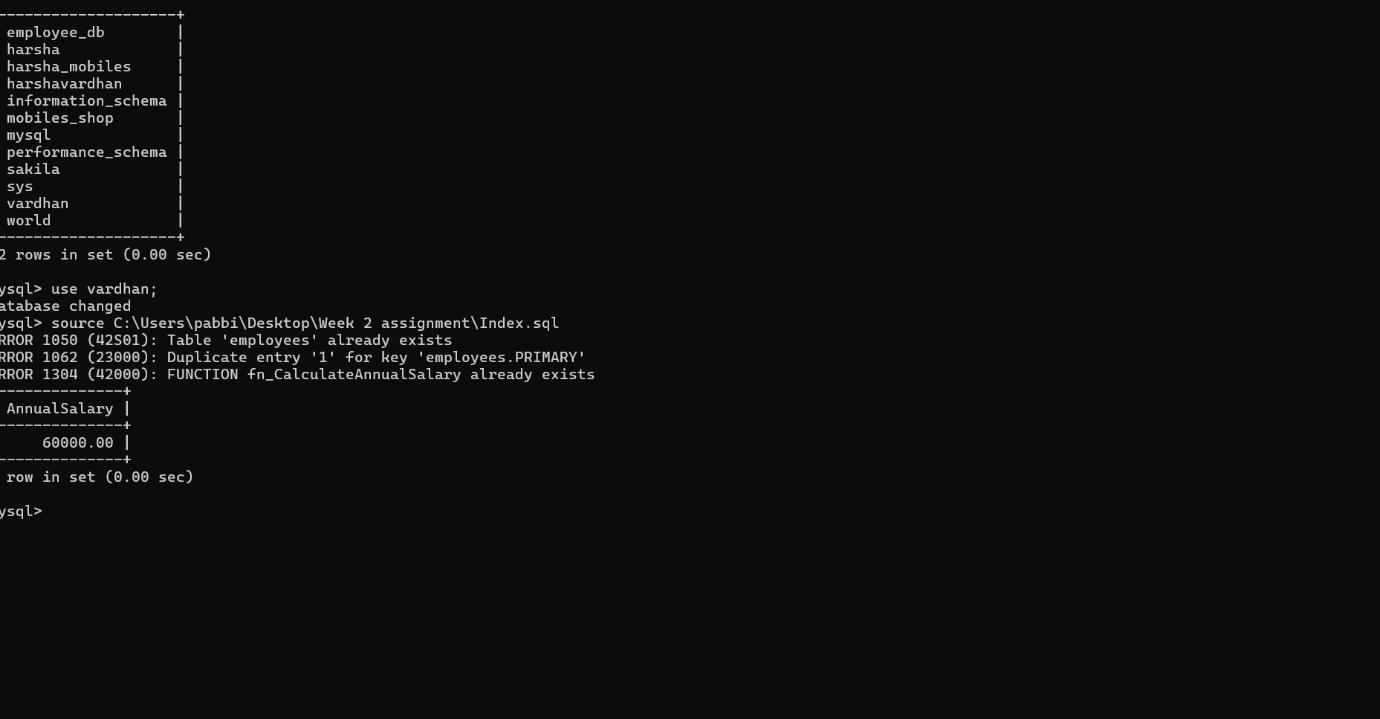
    RETURN AnnualSalary;

END$$

DELIMITER ;

-- Step 4: Execute the Function

SELECT fn\_CalculateAnnualSalary(1) AS AnnualSalary;



**Exercise 4: Return Data from a Stored Procedure**

**Source code**

**Procedure.sql**

-- Step 1: Set a custom delimiter for stored procedure

DELIMITER $$

CREATE PROCEDURE GetEmployeeCountByDepartment (

    IN dept\_id INT,

    OUT total\_employees INT

)

BEGIN

    SELECT COUNT(\*)

    INTO total\_employees

    FROM employees

    WHERE DepartmentID = dept\_id;

END $$

-- Reset the delimiter back to ;

DELIMITER ;

-- Step 2: Declare a variable

SET @total = 0;

-- Step 3: Call the procedure

CALL GetEmployeeCountByDepartment(2, @total);

-- Step 4: Display the result

SELECT @total AS TotalEmployees;

