Advanced SQL, Transactions, and Stored Procedures

Activity: Implementing Advanced SQL Techniques with Stored Procedures and Functions

Objective: by the end of this activity, you will have created and executed stored procedures and functions to automate SQL operations. You will set up a database, populate it with sample data, and manage SQL tasks using stored procedures and functions. This activity will reinforce your ability to work with advanced SQL techniques, manage transactions, and optimize database operations.

Step 1: Prepare for the Activity

You will create a new SQL database and set up the environment using MySQL in Visual Studio Code. Follow these steps to create a database named EmployeeDB and populate it with sample data.

Instructions:

- Open MySQL in Visual Studio Code.
- Create a new database using the following command: **CREATE DATABASE EmployeeDB;**
- Switch to the newly created database: **USE EmployeeDB;**
- Create a table **Employees** to store employee data:

```
CREATE TABLE Employees (
    EmployeeID INT AUTO_INCREMENT PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    Department VARCHAR(50),
    Salary DECIMAL(10, 2),
    HireDate DATE
);
```

- Populate the table with diverse and inclusive sample data:

```
INSERT INTO Employees (FirstName, LastName, Department, Salary,
HireDate)
VALUES
    ('Aisha', 'Khan', 'Finance', 85000.00, '2019-03-15'),
    ('Luis', 'Garcia', 'IT', 95000.00, '2020-07-22'),
    ('Chloe', 'Nguyen', 'Marketing', 72000.00, '2018-10-05'),
    ('Amara', 'Smith', 'HR', 67000.00, '2021-01-18'),
    ('Ravi', 'Patel', 'Finance', 88000.00, '2017-11-03');
```

- Verify the data by running a query to display all records in the Employees table:

```
SELECT * FROM Employees;
```

Step 2: Creating a Stored Procedure

Now, create a stored procedure to increase the salary of employees in a specific department.

Instructions:

1. Define a stored procedure named IncreaseSalary that takes two parameters:
    - deptName (department name)
    - increment (amount to increase salaries).
2. Write the SQL logic to update salaries for employees in the specified department.
3. Execute the procedure and verify the results.

Step 3: Creating a Scalar Function

Create a scalar function to calculate the annual bonus for an employee based on their salary.

Instructions:

1. Define a function named CalculateBonus that takes one parameter:
    - salary (employee's salary).
2. Write the function logic to calculate the bonus as 10% of the salary.
3. Use the function in a query to display each employee's name and bonus.

Step 4: Best Practices

Incorporate error handling and validation in your procedures and functions.

Instructions:

1. Modify the IncreaseSalary procedure to:
    - Validate that the increment is a positive number.
    - Use error handling to return a message if the department doesn't exist.
2. Modify the CalculateBonus function to:
    - Validate that the salary is greater than zero.
    - Use error handling to manage invalid inputs.

lab_advanced_sql.sql:

```
-- ============================================
-- Lab: Advanced SQL, Transactions, and Stored Procedures
-- Database: EmployeeDB
-- ============================================

CREATE DATABASE IF NOT EXISTS EmployeeDB;
USE EmployeeDB;

-- Clean start
DROP TABLE IF EXISTS Employees;

CREATE TABLE Employees (
    EmployeeID INT AUTO_INCREMENT PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    Department VARCHAR(50),
    Salary DECIMAL(10, 2),
    HireDate DATE
);

INSERT INTO Employees (FirstName, LastName, Department, Salary,
HireDate)
VALUES
    ('Aisha', 'Khan', 'Finance', 85000.00, '2019-03-15'),
    ('Luis', 'Garcia', 'IT', 95000.00, '2020-07-22'),
    ('Chloe', 'Nguyen', 'Marketing', 72000.00, '2018-10-05'),
    ('Amara', 'Smith', 'HR', 67000.00, '2021-01-18'),
    ('Ravi', 'Patel', 'Finance', 88000.00, '2017-11-03');

-- Verify seed
SELECT * FROM Employees;

-- ============================================
-- Step 2: Stored Procedure (fixed for MySQL 8.x)
-- ============================================

DROP PROCEDURE IF EXISTS IncreaseSalary;
DELIMITER $$

CREATE PROCEDURE IncreaseSalary(
    IN deptName VARCHAR(50),
    IN p_increment DECIMAL(10,2)
)
BEGIN
    DECLARE deptCount INT;
    DECLARE errMsg VARCHAR(255);

    -- Validate increment
    IF p_increment <= 0 THEN
        SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Error: Increment must be greater than
zero.';
    END IF;

    -- Check department existence
    SELECT COUNT(*) INTO deptCount
    FROM Employees
```

```
    WHERE Department = deptName;

    IF deptCount = 0 THEN
        SET errMsg = CONCAT('Error: Department ', deptName, ' not
found.');
        SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = errMsg;
    ELSE
        START TRANSACTION;
            UPDATE Employees
            SET Salary = Salary + p_increment
            WHERE Department = deptName;
        COMMIT;
    END IF;
END$$

DELIMITER ;

-- Test procedure
CALL IncreaseSalary('Finance', 2000.00);
SELECT * FROM Employees WHERE Department = 'Finance';

-- ============================================
-- Step 3: Scalar Function (fixed for MySQL 8.x)
-- ============================================
DROP FUNCTION IF EXISTS CalculateBonus;
DELIMITER $$

CREATE FUNCTION CalculateBonus(salary DECIMAL(10,2))
RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
    DECLARE bonus DECIMAL(10,2);

    IF salary <= 0 THEN
        SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Error: Salary must be greater than
zero.';
    END IF;

    SET bonus = salary * 0.10;
    RETURN bonus;
END$$

DELIMITER ;

-- Test function
SELECT FirstName, LastName, Salary, CalculateBonus(Salary) AS Bonus
FROM Employees;

-- ============================================
-- Step 4: Error handling tests (uncomment to try)
-- ============================================
-- CALL IncreaseSalary('Finance', -100); -- invalid increment
-- CALL IncreaseSalary('Legal', 2000);   -- nonexistent department
-- SELECT CalculateBonus(-5000);         -- invalid salary

-- Final state
SELECT * FROM Employees;
```