**PL/SQL PROGRAMMING**

**Exercise 1: Control Structures**

**Code:**

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY AUTO\_INCREMENT,

Name VARCHAR(100) NOT NULL,

Age INT,

Balance DECIMAL(15, 2),

IsVIP BOOLEAN DEFAULT FALSE

);

CREATE TABLE Loans (

LoanID INT PRIMARY KEY AUTO\_INCREMENT,

CustomerID INT,

LoanAmount DECIMAL(15, 2) NOT NULL,

InterestRate DECIMAL(5, 2) NOT NULL,

DueDate DATE NOT NULL,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

INSERT INTO Customers (Name, Age, Balance, IsVIP) VALUES

('Alice Smith', 35, 5000.00, FALSE),

('Bob Johnson', 62, 12000.00, FALSE),

('Charlie Brown', 58, 8000.00, FALSE),

('Diana Prince', 65, 15000.00, FALSE),

('Eve Adams', 40, 3000.00, FALSE),

('Frank White', 70, 9000.00, FALSE);

INSERT INTO Loans (CustomerID, LoanAmount, InterestRate, DueDate) VALUES

(1, 25000.00, 5.00, '2025-09-15'),

(2, 50000.00, 4.50, '2025-07-01'),

(3, 10000.00, 6.00, '2025-10-20'),

(4, 75000.00, 4.00, '2025-07-25'),

(5, 5000.00, 7.00, '2025-08-05'),

(6, 30000.00, 4.75, '2025-11-01');

DELIMITER //

CREATE PROCEDURE ApplyInterestDiscount()

BEGIN

DECLARE v\_customerID INT;

DECLARE v\_age INT;

DECLARE done INT DEFAULT FALSE;

DECLARE cur\_old\_customers CURSOR FOR

SELECT CustomerID, Age

FROM Customers

WHERE Age > 60;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur\_old\_customers;

read\_loop: LOOP

FETCH cur\_old\_customers INTO v\_customerID, v\_age;

IF done THEN

LEAVE read\_loop;

END IF;

UPDATE Loans

SET InterestRate = GREATEST(0.00, InterestRate - 1.00)

WHERE CustomerID = v\_customerID;

SELECT CONCAT('Applied 1% discount for CustomerID: ', v\_customerID, ', Age: ', v\_age) AS Status;

END LOOP;

CLOSE cur\_old\_customers;

END //

DELIMITER ;

CALL ApplyInterestDiscount();

SELECT \* FROM Customers;

DELIMITER //

CREATE PROCEDURE PromoteToVIP()

BEGIN

DECLARE v\_customerID INT;

DECLARE v\_balance DECIMAL(15, 2);

DECLARE done INT DEFAULT FALSE;

DECLARE cur\_high\_balance\_customers CURSOR FOR

SELECT CustomerID, Balance

FROM Customers

WHERE Balance > 10000.00;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur\_high\_balance\_customers;

read\_loop: LOOP

FETCH cur\_high\_balance\_customers INTO v\_customerID, v\_balance;

IF done THEN

LEAVE read\_loop;

END IF;

UPDATE Customers

SET IsVIP = TRUE

WHERE CustomerID = v\_customerID;

SELECT CONCAT('Promoted CustomerID: ', v\_customerID, ' to VIP status. Balance: $', v\_balance) AS Status;

END LOOP;

CLOSE cur\_high\_balance\_customers;

END //

DELIMITER ;

CALL PromoteToVIP();

DROP PROCEDURE IF EXISTS SendLoanReminders;

DELIMITER //

CREATE PROCEDURE SendLoanReminders()

BEGIN

-- Declare variables

DECLARE v\_loanID INT;

DECLARE v\_customerID INT;

DECLARE v\_name VARCHAR(100);

DECLARE v\_dueDate DATE;

DECLARE done INT DEFAULT FALSE;

DECLARE future\_date DATE;

DECLARE cur\_due\_loans CURSOR FOR

SELECT

L.LoanID,

C.CustomerID,

C.Name,

L.DueDate

FROM

Loans L

JOIN

Customers C ON L.CustomerID = C.CustomerID

WHERE

L.DueDate BETWEEN CURDATE() AND DATE\_ADD(CURDATE(), INTERVAL 30 DAY);

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

SET future\_date = DATE\_ADD(CURDATE(), INTERVAL 30 DAY);

OPEN cur\_due\_loans;

read\_loop: LOOP

FETCH cur\_due\_loans INTO v\_loanID, v\_customerID, v\_name, v\_dueDate;

IF done THEN

LEAVE read\_loop;

END IF;

SELECT CONCAT(

'Reminder: Dear ', v\_name, ' (CustomerID: ', v\_customerID,

'), your loan (ID: ', v\_loanID,

') is due on ', DATE\_FORMAT(v\_dueDate, '%Y-%m-%d'),

'. Please ensure timely payment.'

) AS ReminderMessage;

END LOOP;

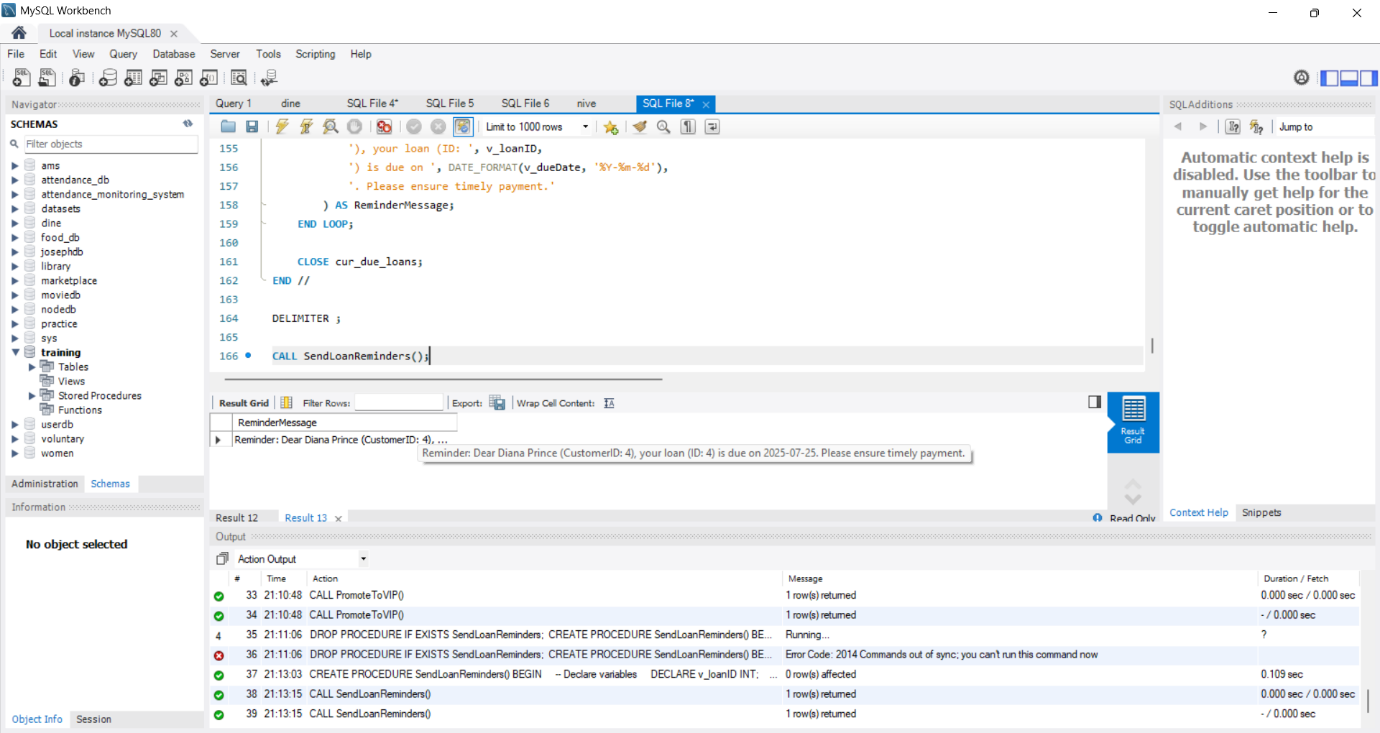
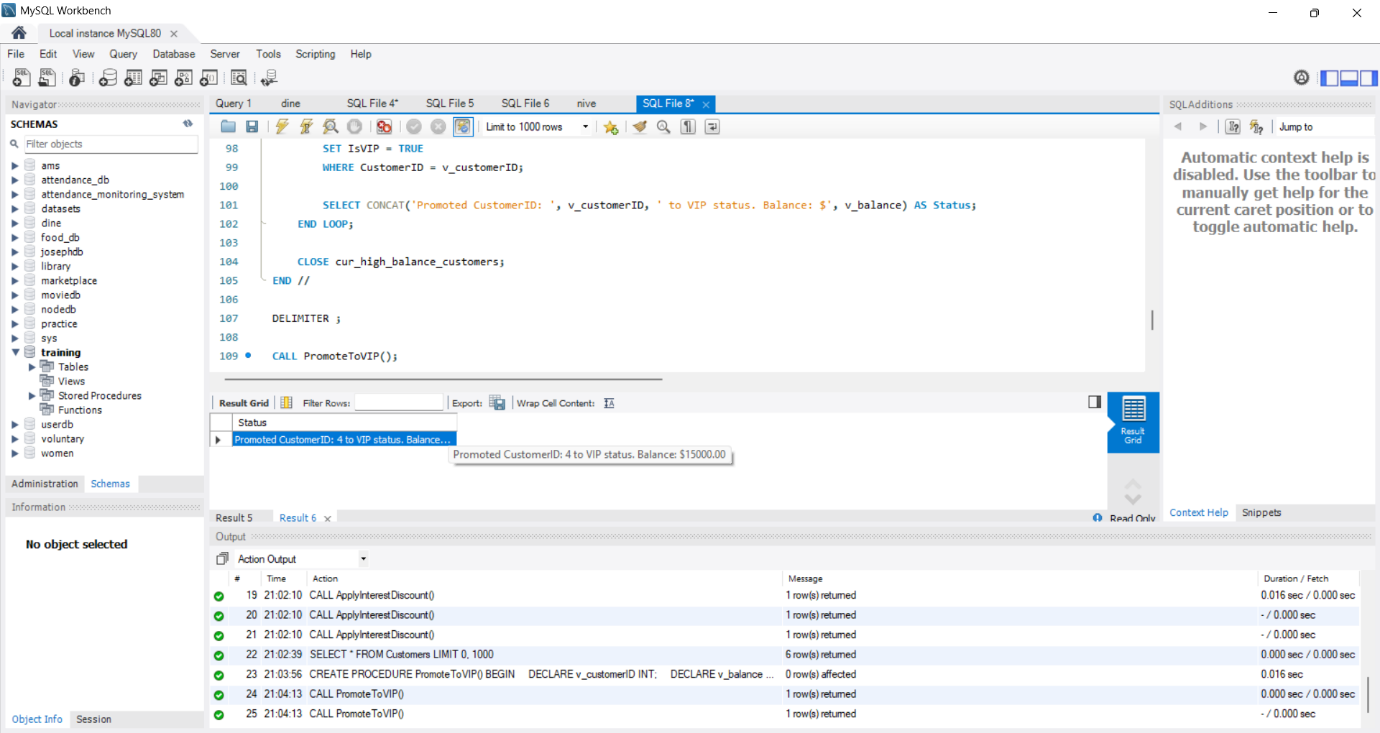
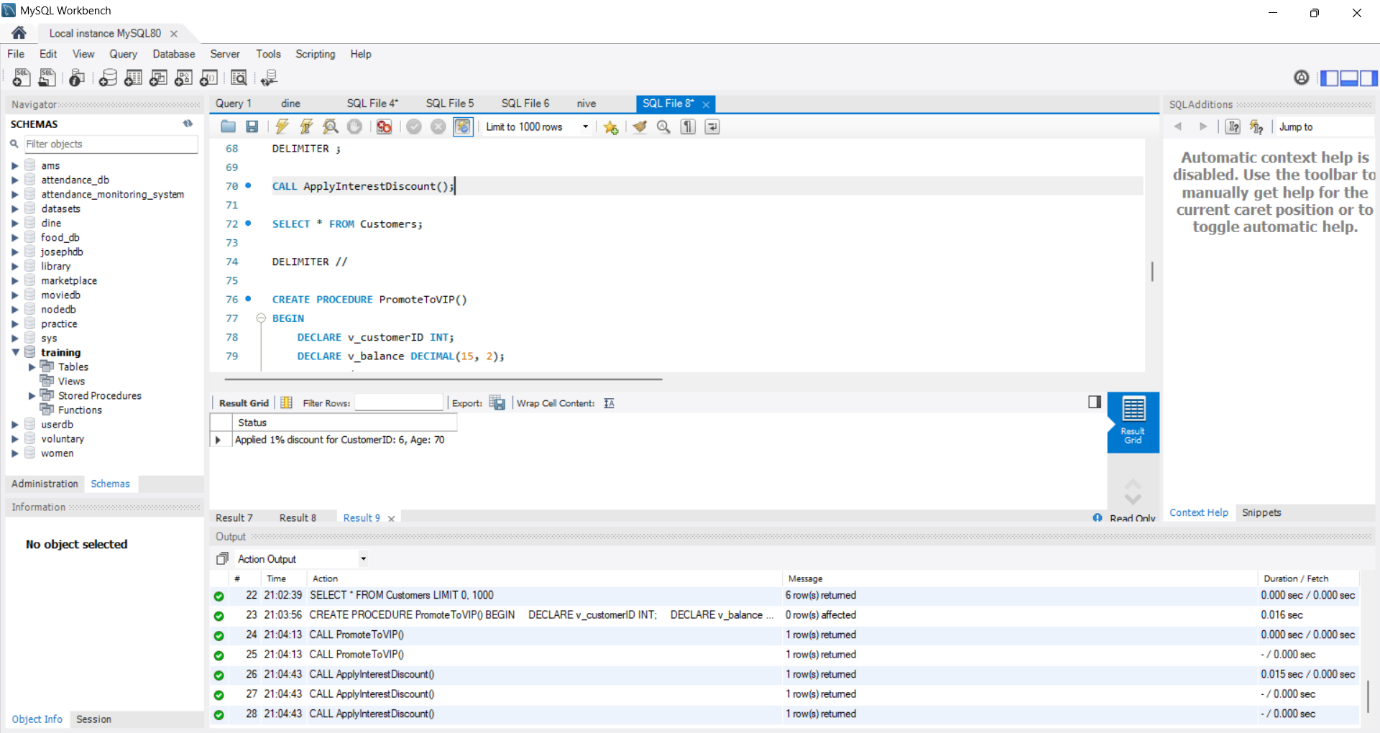
CLOSE cur\_due\_loans;

END //

DELIMITER ;

CALL SendLoanReminders();

**Output:**



**Exercise 3: Stored Procedures**

**Code:**

DROP TABLE IF EXISTS Accounts;

CREATE TABLE Accounts (

AccountID INT PRIMARY KEY AUTO\_INCREMENT,

CustomerID INT,

AccountType VARCHAR(50),

Balance DECIMAL(15,2)

);

-- Dummy Values

INSERT INTO Accounts (CustomerID, AccountType, Balance) VALUES

(1, 'Savings', 10000.00),

(2, 'Savings', 20000.00),

(3, 'Checking', 15000.00),

(4, 'Savings', 5000.00),

(5, 'Checking', 8000.00);

DROP TABLE IF EXISTS Employees;

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY AUTO\_INCREMENT,

Name VARCHAR(100),

Department VARCHAR(100),

Salary DECIMAL(15,2)

);

-- Dummy Values

INSERT INTO Employees (Name, Department, Salary) VALUES

('Alice', 'IT', 50000.00),

('Bob', 'HR', 45000.00),

('Charlie', 'IT', 55000.00),

('Diana', 'Finance', 60000.00),

('Eve', 'HR', 47000.00);

DELIMITER //

CREATE PROCEDURE ProcessMonthlyInterest()

BEGIN

UPDATE Accounts

SET Balance = Balance \* 1.01

WHERE AccountType = 'Savings';

SELECT 'Interest applied to all savings accounts.' AS Status;

END //

DELIMITER ;

-- Execute

SET SQL\_SAFE\_UPDATES = 0;

CALL ProcessMonthlyInterest();

SET SQL\_SAFE\_UPDATES = 1;

DELIMITER //

CREATE PROCEDURE UpdateEmployeeBonus(

IN dept\_name VARCHAR(100),

IN bonus\_percent DECIMAL(5,2)

)

BEGIN

UPDATE Employees

SET Salary = Salary + (Salary \* bonus\_percent / 100)

WHERE Department = dept\_name;

SELECT CONCAT('Bonus of ', bonus\_percent, '% applied to ', dept\_name, ' department.') AS Status;

END //

DELIMITER ;

SET SQL\_SAFE\_UPDATES = 0;

CALL UpdateEmployeeBonus('IT', 10);

SET SQL\_SAFE\_UPDATES = 1;

DELIMITER //

CREATE PROCEDURE TransferFunds(

IN sourceAccountID INT,

IN targetAccountID INT,

IN amount DECIMAL(15,2)

)

BEGIN

DECLARE source\_balance DECIMAL(15,2);

-- Get source account balance

SELECT Balance INTO source\_balance

FROM Accounts

WHERE AccountID = sourceAccountID;

-- Check for sufficient balance

IF source\_balance IS NULL THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Source account does not exist.';

ELSEIF source\_balance < amount THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Insufficient funds in source account.';

ELSE

-- Deduct from source

UPDATE Accounts

SET Balance = Balance - amount

WHERE AccountID = sourceAccountID;

-- Add to target

UPDATE Accounts

SET Balance = Balance + amount

WHERE AccountID = targetAccountID;

SELECT CONCAT('Transferred ₹', amount, ' from Account ', sourceAccountID, ' to Account ', targetAccountID) AS Status;

END IF;

END //

DELIMITER ;

CALL TransferFunds(1, 2, 2000.00);

**Output:**

