**Exercise 3: Stored Procedures**

**Scenario 1:** The bank needs to process monthly interest for all savings accounts.

**Question:** Write a stored procedure **ProcessMonthlyInterest** that calculates and updates the balance of all savings accounts by applying an interest rate of 1% to the current balance.

**PL/SQL Procedure:**

DELIMITER //

CREATE PROCEDURE ProcessMonthlyInterest()

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE account\_id INT;

DECLARE account\_balance DECIMAL(10, 2);

DECLARE account\_cursor CURSOR FOR

SELECT AccountID, Balance FROM Accounts WHERE AccountType = 'Savings';

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN account\_cursor;

read\_loop: LOOP

FETCH account\_cursor INTO account\_id, account\_balance;

IF done THEN

LEAVE read\_loop;

END IF;

-- Calculate and update the balance

UPDATE Accounts

SET Balance = Balance \* 1.01

WHERE AccountID = account\_id;

END LOOP;

CLOSE account\_cursor;

END;

//

DELIMITER ;

**Sample Execution:**

CALL ProcessMonthlyInterest();

**Scenario 2:** The bank wants to implement a bonus scheme for employees based on their performance.

**Question:** Write a stored procedure **UpdateEmployeeBonus** that updates the salary of employees in a given department by adding a bonus percentage passed as a parameter.

**PL/SQL Procedure:**

DELIMITER //

CREATE PROCEDURE UpdateEmployeeBonus(IN p\_Department VARCHAR(50), IN p\_BonusPercentage DECIMAL(5, 2))

BEGIN

UPDATE Employees

SET Salary = Salary \* (1 + p\_BonusPercentage / 100)

WHERE Department = p\_Department;

END;

//

DELIMITER ;

**Sample Execution:**

CALL UpdateEmployeeBonus('HR', 10);

**Viewing Outputs:**

SELECT \* FROM Employees;

**Scenario 3:** Customers should be able to transfer funds between their accounts.

**Question:** Write a stored procedure **TransferFunds** that transfers a specified amount from one account to another, checking that the source account has sufficient balance before making the transfer.

**PL/SQL Procedure:**

DELIMITER //

CREATE PROCEDURE TransferFunds(

IN p\_FromAccountID INT,

IN p\_ToAccountID INT,

IN p\_Amount DECIMAL(10, 2)

)

BEGIN

DECLARE v\_FromBalance DECIMAL(10, 2);

-- Check the balance of the source account

SELECT Balance INTO v\_FromBalance FROM Accounts WHERE AccountID = p\_FromAccountID;

IF v\_FromBalance < p\_Amount THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Insufficient funds in the source account.';

ELSE

-- Deduct the amount from the source account

UPDATE Accounts SET Balance = Balance - p\_Amount WHERE AccountID = p\_FromAccountID;

-- Add the amount to the destination account

UPDATE Accounts SET Balance = Balance + p\_Amount WHERE AccountID = p\_ToAccountID;

END IF;

END;

//

DELIMITER ;

**Sample Execution:**

CALL TransferFunds(1, 2, 200);

**Viewing Outputs:**

SELECT \* FROM Accounts;