**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.

**Ans:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

  v\_interest\_rate CONSTANT NUMBER := 0.01;

  v\_new\_balance NUMBER;

  v\_account\_id Accounts.AccountID%TYPE;

  v\_error\_message VARCHAR2(4000);

BEGIN

  FOR account\_rec IN (SELECT AccountID, Balance FROM Accounts WHERE AccountType = 'Savings') LOOP

    BEGIN

      v\_new\_balance := account\_rec.Balance + (account\_rec.Balance \* v\_interest\_rate);

      UPDATE Accounts

      SET Balance = v\_new\_balance,

          LastModified = SYSDATE

      WHERE AccountID = account\_rec.AccountID;

      COMMIT;

    EXCEPTION

      WHEN OTHERS THEN

        v\_error\_message := SQLERRM;

        INSERT INTO ErrorLog (ErrorMessage, ErrorDate) VALUES (v\_error\_message, SYSDATE);

        ROLLBACK;

    END;

  END LOOP;

END 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.

**Ans:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

  p\_department IN VARCHAR2,

  p\_bonus\_percentage IN NUMBER

) AS

  v\_new\_salary NUMBER;

  v\_error\_message VARCHAR2(4000);

BEGIN

  FOR emp\_rec IN (SELECT EmployeeID, Salary FROM Employees WHERE Department = p\_department) LOOP

    BEGIN

      v\_new\_salary := emp\_rec.Salary \* (1 + p\_bonus\_percentage / 100);

      UPDATE Employees

      SET Salary = v\_new\_salary

      WHERE EmployeeID = emp\_rec.EmployeeID;

      COMMIT;

    EXCEPTION

      WHEN OTHERS THEN

        v\_error\_message := SQLERRM;

        INSERT INTO ErrorLog (ErrorMessage, ErrorDate) VALUES (v\_error\_message, SYSDATE);

        ROLLBACK;

    END;

  END LOOP;

END UpdateEmployeeBonus;

**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.

**Ans:**

CREATE OR REPLACE PROCEDURE TransferFunds (

  p\_from\_account\_id IN NUMBER,

  p\_to\_account\_id IN NUMBER,

  p\_amount IN NUMBER

) AS

  v\_from\_balance NUMBER;

  v\_error\_message VARCHAR2(4000);

BEGIN

  SELECT Balance INTO v\_from\_balance FROM Accounts WHERE AccountID = p\_from\_account\_id FOR UPDATE;

  IF v\_from\_balance < p\_amount THEN

    RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient funds in the source account.');

  END IF;

  UPDATE Accounts

  SET Balance = Balance - p\_amount

  WHERE AccountID = p\_from\_account\_id;

  UPDATE Accounts

  SET Balance = Balance + p\_amount

  WHERE AccountID = p\_to\_account\_id;

  COMMIT;

EXCEPTION

  WHEN OTHERS THEN

    v\_error\_message := SQLERRM;

    ROLLBACK;

    INSERT INTO ErrorLog (ErrorMessage, ErrorDate) VALUES (v\_error\_message, SYSDATE);

END TransferFunds;