**EXERCISE 3 : STORED PROCEDURES**

**Scenario 1: ProcessMonthlyInterest - Update Savings Accounts with Monthly Interest**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

v\_InterestRate CONSTANT NUMBER := 0.01; -- 1% interest rate

v\_NewBalance Accounts.Balance%TYPE;

CURSOR cur\_SavingsAccounts IS

SELECT AccountID, Balance

FROM Accounts

WHERE AccountType = 'Savings'

FOR UPDATE OF Balance;

BEGIN

FOR rec IN cur\_SavingsAccounts LOOP

-- Calculate the new balance with interest

v\_NewBalance := rec.Balance \* (1 + v\_InterestRate);

-- Update the balance of the account

UPDATE Accounts

SET Balance = v\_NewBalance

WHERE CURRENT OF cur\_SavingsAccounts;

-- Log the update for verification

DBMS\_OUTPUT.PUT\_LINE('Updated Account ID ' || rec.AccountID || ' with new balance: ' || v\_NewBalance);

END LOOP;

-- Commit the transaction

COMMIT;

END ProcessMonthlyInterest;

/

**Scenario 2: UpdateEmployeeBonus - Implement Bonus Scheme for Employees**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_Department IN Employees.Department%TYPE,

p\_BonusPercentage IN NUMBER

) IS

v\_NewSalary Employees.Salary%TYPE;

CURSOR cur\_Employees IS

SELECT EmployeeID, Salary

FROM Employees

WHERE Department = p\_Department

FOR UPDATE OF Salary;

BEGIN

FOR rec IN cur\_Employees LOOP

-- Calculate the new salary with the bonus

v\_NewSalary := rec.Salary \* (1 + p\_BonusPercentage / 100);

-- Update the employee's salary

UPDATE Employees

SET Salary = v\_NewSalary

WHERE CURRENT OF cur\_Employees;

-- Log the update for verification

DBMS\_OUTPUT.PUT\_LINE('Updated Employee ID ' || rec.EmployeeID || ' with new salary: ' || v\_NewSalary);

END LOOP;

-- Commit the transaction

COMMIT;

END UpdateEmployeeBonus;

/

**Scenario 3: TransferFunds - Enable Fund Transfer Between Customer Accounts**

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_FromAccountID IN Accounts.AccountID%TYPE,

p\_ToAccountID IN Accounts.AccountID%TYPE,

p\_Amount IN NUMBER

) IS

v\_FromBalance Accounts.Balance%TYPE;

v\_ToBalance Accounts.Balance%TYPE;

BEGIN

-- Lock the accounts for update to prevent race conditions

SELECT Balance INTO v\_FromBalance

FROM Accounts

WHERE AccountID = p\_FromAccountID

FOR UPDATE;

SELECT Balance INTO v\_ToBalance

FROM Accounts

WHERE AccountID = p\_ToAccountID

FOR UPDATE;

-- Check if the from account has enough balance

IF v\_FromBalance < p\_Amount THEN

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

END IF;

-- Perform the fund transfer

UPDATE Accounts

SET Balance = Balance - p\_Amount

WHERE AccountID = p\_FromAccountID;

UPDATE Accounts

SET Balance = Balance + p\_Amount

WHERE AccountID = p\_ToAccountID;

-- Commit the transaction

COMMIT;

-- Log success message

DBMS\_OUTPUT.PUT\_LINE('Transfer of ' || p\_Amount || ' from account ' || p\_FromAccountID || ' to account ' || p\_ToAccountID || ' completed successfully.');

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error: One of the account IDs does not exist.');

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

END TransferFunds;

/