**EXERCISE 3 – STORED PROCEDURES**

**SCENARIO 1 - THE BANK NEEDS TO PROCESS MONTHLY INTEREST FOR ALL SAVINGS ACCOUNTS.**

**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.**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

UPDATE Accounts SET Balance = Balance + (Balance \* 0.01)

WHERE AccountType = 'Savings';

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Monthly interest applied successfully to all savings accounts.');

EXCEPTION

WHEN OTHERS THEN

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

ROLLBACK;

END;

/

-- calling the stored procedure

BEGIN

ProcessMonthlyInterest;

END;

/

**SCENARIO 2 - THE BANK WANTS TO IMPLEMENT A BONUS SCHEME FOR EMPLOYEES BASED ON THEIR PERFORMANCE.**

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

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

dept IN VARCHAR2,

bonus IN NUMBER

) IS

BEGIN

UPDATE Employees

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

WHERE Department = dept;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Salaries updated successfully in department: ' || dept ||

'. The Applied percent of bonus is: ' || bonus || '%');

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

-- Handle the case where no employees exist in the specified department

DBMS\_OUTPUT.PUT\_LINE('No employees found in department: ' || dept);

ROLLBACK;

WHEN OTHERS THEN

-- Handle any other unexpected errors

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

ROLLBACK;

END;

/

-- calling the stored procedure

BEGIN

    UpdateEmployeeBonus( 'HR', 2);

END;

/

**SCENARIO 3 - CUSTOMERS SHOULD BE ABLE TO TRANSFER FUNDS BETWEEN THEIR ACCOUNTS.**

**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.**

CREATE OR REPLACE PROCEDURE TransferFunds(

from\_acc IN NUMBER,

to\_acc IN NUMBER,

transfer\_amount IN NUMBER

) IS

from\_acc\_balance Accounts.Balance%TYPE;

to\_acc\_balance Accounts.Balance%TYPE;

BEGIN

SELECT Balance INTO from\_acc\_balance FROM Accounts

WHERE AccountID = from\_acc FOR UPDATE;

SELECT Balance INTO to\_acc\_balance FROM Accounts WHERE AccountID = to\_acc

FOR UPDATE;

IF from\_acc\_balance < transfer\_amount THEN

DBMS\_OUTPUT.PUT\_LINE('Insufficient funds to make the transaction');

ELSE

UPDATE Accounts SET Balance = Balance - transfer\_amount

WHERE AccountID = from\_acc;

UPDATE Accounts SET Balance = Balance + transfer\_amount

WHERE AccountID = to\_acc;

DBMS\_OUTPUT.PUT\_LINE('Transfer of ' || transfer\_amount || ' from account ' || from\_acc || ' to account ' || to\_acc || ' completed successfully.');

END IF;

COMMIT;

END;

/

-- calling the stored procedure

DECLARE

src\_acc INTEGER;

dest\_acc INTEGER;

amt INTEGER;

BEGIN

TransferFunds(&src\_acc,&dest\_acc,&amt);

END;

/