**Exercise 3: Stored Procedures**

**Scenario 1: (Ex3-Scenario1.sql)**

@InitializeData.sql

SET ECHO ON

SET SERVEROUTPUT ON SIZE UNLIMITED

SPOOL output-Ex3-Scenario1.txt

VARIABLE input VARCHAR2(30)

*-- Procedure to process monthly interest for savings accounts*

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 has been applied to all savings accounts.');

END PROCESSMONTHLYINTEREST;

/

*-- Test the procedure*

BEGIN

    PROCESSMONTHLYINTEREST;

END;

/

SELECT

    \*

FROM

    ACCOUNTS;

SPOOL OFF

@DropData.sql

**Scenario 2: (Ex3-Scenario2.sql)**

@InitializeData.sql

SET ECHO ON

SET SERVEROUTPUT ON SIZE UNLIMITED

SPOOL output-Ex3-Scenario2.txt

VARIABLE input VARCHAR2(30)

*-- Procedure to update employee bonus*

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

    p\_department IN VARCHAR2,

    p\_bonus\_percentage IN NUMBER

) IS

BEGIN

    UPDATE Employees

    SET Salary = Salary + (Salary \* p\_bonus\_percentage / 100)

    WHERE Department = p\_department;

    COMMIT;

    DBMS\_OUTPUT.PUT\_LINE('Bonus has been updated for employees in department ' || p\_department || '.');

END UpdateEmployeeBonus;

/

*-- Before calling procedure*

SELECT \* FROM Employees;

*-- Test the procedure*

BEGIN

    UpdateEmployeeBonus('IT', 10); *-- Apply 10% bonus to employees in IT department*

END;

/

*-- After calling procedure*

SELECT \* FROM Employees;

SPOOL OFF

@DropData.sql

**Scenario 3: (Ex3-Scenario3.sql)**

@InitializeData.sql

SET ECHO ON

SET SERVEROUTPUT ON SIZE UNLIMITED

SPOOL output-Ex3-Scenario3.txt

VARIABLE input VARCHAR2(30)

*-- Procedure to transfer funds between accounts*

CREATE OR REPLACE PROCEDURE TransferFunds (

    p\_from\_account\_id IN NUMBER,

    p\_to\_account\_id IN NUMBER,

    p\_amount IN NUMBER

) IS

    insufficient\_funds EXCEPTION;

    v\_balance NUMBER;

BEGIN

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

    IF v\_balance < p\_amount THEN

        RAISE insufficient\_funds;

    ELSE

        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;

    END IF;

EXCEPTION

    WHEN insufficient\_funds THEN

        DBMS\_OUTPUT.PUT\_LINE('Error: Insufficient funds in the source account.');

        ROLLBACK;

    WHEN OTHERS THEN

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

        ROLLBACK;

END TransferFunds;

/

*-- Before the procedure call*

SELECT \* FROM Accounts;

*-- Test the procedure*

BEGIN

    TransferFunds(1, 2, 800); *-- Transfer 800 from account 1 to account 2*

END;

/

*-- After the procedure call*

SELECT \* FROM Accounts;

SPOOL OFF

@DropData.sql