**Exercise 2: Error Handling**

**Scenario 1: (Ex2-Scenario1.sql)**

@InitializeData.sql

SET ECHO ON

SET SERVEROUTPUT ON SIZE UNLIMITED

SPOOL output-Ex2-Scenario1.txt

VARIABLE input VARCHAR2(30)

*-- Insert sample customers*

INSERT INTO CUSTOMERS (CustomerID, Name, DOB, Balance, LastModified)

VALUES (1001, 'John Doe', TO\_DATE('1950-01-01', 'YYYY-MM-DD'), 5000, SYSDATE);

INSERT INTO CUSTOMERS (CustomerID, Name, DOB, Balance, LastModified)

VALUES (1002, 'Jane Smith', TO\_DATE('1955-01-01', 'YYYY-MM-DD'), 6000, SYSDATE);

*-- Insert sample accounts with balances*

INSERT INTO ACCOUNTS (AccountID, CustomerID, AccountType, Balance, LastModified)

VALUES (3001, 1001, 'Checking', 5000, SYSDATE);

INSERT INTO ACCOUNTS (AccountID, CustomerID, AccountType, Balance, LastModified)

VALUES (3002, 1002, 'Savings', 1000, SYSDATE);

*-- Procedure for safe fund transfer*

CREATE OR REPLACE PROCEDURE SafeTransferFunds (

    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;

        DBMS\_OUTPUT.PUT\_LINE('Fund Transfer Successful');

        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 SafeTransferFunds;

/

*-- Test the procedure*

BEGIN

    SafeTransferFunds(3001, 3002, 1000); *-- This should be successful*

    SafeTransferFunds(3001, 3002, 6000); *-- This should cause insufficient funds error*

END;

/

SPOOL OFF

@DropData.sql

**Scenario 2: (Ex2-Scenario2.sql)**

@InitializeData.sql

SET ECHO ON

SET SERVEROUTPUT ON SIZE UNLIMITED

SPOOL output-Ex2-Scenario2.txt

VARIABLE input VARCHAR2(30)

*-- Insert sample employees*

INSERT INTO EMPLOYEES (EMPLOYEEID, NAME, POSITION, SALARY, DEPARTMENT, HIREDATE)

VALUES (4001, 'Alice Johnson', 'Manager', 70000, 'HR', TO\_DATE('2015-06-15', 'YYYY-MM-DD'));

INSERT INTO EMPLOYEES (EMPLOYEEID, NAME, POSITION, SALARY, DEPARTMENT, HIREDATE)

VALUES (4002, 'Bob Brown', 'Developer', 60000, 'IT', TO\_DATE('2017-03-20', 'YYYY-MM-DD'));

*-- Procedure to update salary*

CREATE OR REPLACE PROCEDURE UpdateSalary (

    p\_employee\_id IN NUMBER,

    p\_percentage IN NUMBER

) IS

    employee\_not\_found EXCEPTION;

BEGIN

    UPDATE Employees

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

    WHERE EmployeeID = p\_employee\_id;

    IF SQL%ROWCOUNT = 0 THEN

        RAISE employee\_not\_found;

    END IF;

    COMMIT;

        DBMS\_OUTPUT.PUT\_LINE('Salary Update Successfull');

EXCEPTION

    WHEN employee\_not\_found THEN

        DBMS\_OUTPUT.PUT\_LINE('Error: Employee not found.');

        ROLLBACK;

    WHEN OTHERS THEN

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

        ROLLBACK;

END UpdateSalary;

/

*-- Test the procedure*

BEGIN

    UpdateSalary(4001, 10); *-- This should be successful*

    UpdateSalary(9999, 10); *-- This should cause employee not found error*

END;

/

SPOOL OFF

@DropData.sql

**Scenario 3: (Ex2-Scenario3.sql)**

@InitializeData.sql

SET ECHO ON

SET SERVEROUTPUT ON SIZE UNLIMITED

SPOOL output-Ex2-Scenario3.txt

VARIABLE input VARCHAR2(30)

*-- Procedure to add a new customer*

CREATE OR REPLACE PROCEDURE AddNewCustomer (

    p\_customer\_id IN NUMBER,

    p\_name IN VARCHAR2,

    p\_dob IN DATE,

    p\_balance IN NUMBER

) IS

    customer\_exists EXCEPTION;

BEGIN

    INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)

    VALUES (p\_customer\_id, p\_name, p\_dob, p\_balance, SYSDATE);

    COMMIT;

    DBMS\_OUTPUT.PUT\_LINE('Customer successfully added');

EXCEPTION

    WHEN DUP\_VAL\_ON\_INDEX THEN

        DBMS\_OUTPUT.PUT\_LINE('Error: Customer with ID ' || p\_customer\_id || ' already exists.');

        ROLLBACK;

    WHEN OTHERS THEN

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

        ROLLBACK;

END AddNewCustomer;

/

*-- Test the procedure*

BEGIN

    AddNewCustomer(1006, 'David Wright', TO\_DATE('1980-02-15', 'YYYY-MM-DD'), 8000); *-- This should be successful*

    AddNewCustomer(1006, 'Eve Adams', TO\_DATE('1985-03-10', 'YYYY-MM-DD'), 9000); *-- This should cause duplicate customer error*

END;

/

SPOOL OFF

@DropData.sql