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
```