**EXERCISE 2**-**ERROR HANDLING**

**Table creation:**

CREATE TABLE Accounts (

    AccountID NUMBER PRIMARY KEY,

    CustomerID NUMBER,

    AccountType VARCHAR2(20),

    Balance NUMBER,

    LastModified DATE,

    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

**INSERT SAMPLE DATA:**

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

VALUES (101, 1, 'Savings', 5000, SYSDATE);

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

VALUES (102, 2, 'Savings', 2000, SYSDATE);

COMMIT;

**SCENARIO 1:** **Safe fund transfer**

**CREATING PROCEDURE:**

CREATE OR REPLACE PROCEDURE SafeTransferFunds (

p\_from\_acc NUMBER,

p\_to\_acc NUMBER,

p\_amount NUMBER

)

IS

e\_insufficient\_funds EXCEPTION;

BEGIN

-- check sufficient funds

DECLARE

v\_balance NUMBER;

BEGIN

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

IF v\_balance < p\_amount THEN

RAISE e\_insufficient\_funds;

END IF;

-- deduct from source

UPDATE Accounts

SET Balance = Balance - p\_amount

WHERE AccountID = p\_from\_acc;

-- add to destination

UPDATE Accounts

SET Balance = Balance + p\_amount

WHERE AccountID = p\_to\_acc;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Transfer successful.');

EXCEPTION

WHEN e\_insufficient\_funds THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Insufficient funds.');

ROLLBACK;

WHEN OTHERS THEN

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

ROLLBACK;

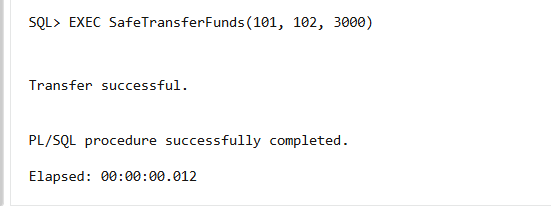
END;/

**ENABLING OUTPUT:**

SET SERVEROUTPUT ON;

EXEC SafeTransferFunds(101, 102, 3000);

**OUTPUT:**

****

**SCENARIO 2:** **Update employee salary**

**TABLE CREATION:**

CREATE TABLE Employees (

EmployeeID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

Position VARCHAR2(50),

Salary NUMBER,

Department VARCHAR2(50),

HireDate DATE

);

**PROCEDURE:**

SELECT \* FROM Employees;

CREATE OR REPLACE PROCEDURE UpdateSalary (

    p\_empid NUMBER,

    p\_percent NUMBER

)

IS

    v\_count NUMBER;

BEGIN

    -- check if employee exists

    SELECT COUNT(\*) INTO v\_count

    FROM Employees

    WHERE EmployeeID = p\_empid;

    IF v\_count = 0 THEN

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

        RETURN;

    END IF;

    UPDATE Employees

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

    WHERE EmployeeID = p\_empid;

COMMIT;

    DBMS\_OUTPUT.PUT\_LINE('Salary updated successfully.');

EXCEPTION

    WHEN OTHERS THEN

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

        ROLLBACK;

END;

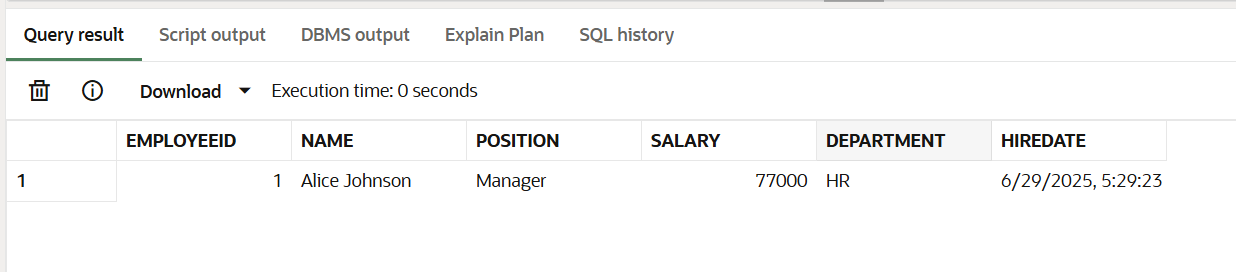
/

SET SERVEROUTPUT ON;

EXEC UpdateSalary(1, 10);

SELECT \* FROM Employees;

**OUTPUT:**

****

**SCENARIO 3:** **Add customer with duplicate check**

CREATE OR REPLACE PROCEDURE AddNewCustomer (

    p\_custid NUMBER,

    p\_name VARCHAR2,

    p\_dob DATE,

    p\_balance NUMBER

)

IS

BEGIN

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

    VALUES (p\_custid, p\_name, p\_dob, p\_balance, SYSDATE);

    COMMIT;

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

EXCEPTION

    WHEN DUP\_VAL\_ON\_INDEX THEN

        DBMS\_OUTPUT.PUT\_LINE('Error: Customer ID already exists.');

        ROLLBACK;

    WHEN OTHERS THEN

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

        ROLLBACK;

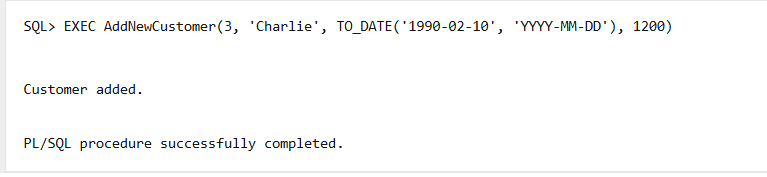
END;

/

EXEC AddNewCustomer(3, 'Charlie', TO\_DATE('1990-02-10', 'YYYY-MM-DD'), 1200);

SELECT \* FROM CUSTOMERS;

**OUTPUT:**

****