EXERCISE 2:

**Scenario 1: Handle Exceptions During Fund Transfers Between Accounts**

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;

PRAGMA EXCEPTION\_INIT(insufficient\_funds, -20001);

BEGIN

-- Start the transaction

SAVEPOINT start\_transfer;

-- Check if the from account has sufficient funds

DECLARE

v\_balance NUMBER;

BEGIN

SELECT balance INTO v\_balance

FROM accounts

WHERE account\_id = p\_from\_account\_id;

IF v\_balance < p\_amount THEN

RAISE insufficient\_funds;

END IF;

END;

-- Deduct amount from the from account

UPDATE accounts

SET balance = balance - p\_amount

WHERE account\_id = p\_from\_account\_id;

-- Add amount to the to account

UPDATE accounts

SET balance = balance + p\_amount

WHERE account\_id = p\_to\_account\_id;

-- Commit the transaction

COMMIT;

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

EXCEPTION

WHEN insufficient\_funds THEN

ROLLBACK TO start\_transfer;

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

INSERT INTO error\_log (error\_message, error\_time)

VALUES ('Insufficient funds during transfer', SYSDATE);

WHEN OTHERS THEN

ROLLBACK TO start\_transfer;

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

INSERT INTO error\_log (error\_message, error\_time)

VALUES (SQLERRM, SYSDATE);

END;

/

**Scenario 2: Manage Errors When Updating Employee Salaries**

CREATE OR REPLACE PROCEDURE UpdateSalary(

p\_employee\_id IN NUMBER,

p\_percentage IN NUMBER

) IS

no\_employee\_found EXCEPTION;

PRAGMA EXCEPTION\_INIT(no\_employee\_found, -20002);

BEGIN

-- Check if the employee exists and update the salary

UPDATE employees

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

WHERE employee\_id = p\_employee\_id;

IF SQL%ROWCOUNT = 0 THEN

RAISE no\_employee\_found;

END IF;

COMMIT;

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

EXCEPTION

WHEN no\_employee\_found THEN

ROLLBACK;

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

INSERT INTO error\_log (error\_message, error\_time)

VALUES ('Employee ID not found: ' || p\_employee\_id, SYSDATE);

WHEN OTHERS THEN

ROLLBACK;

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

INSERT INTO error\_log (error\_message, error\_time)

VALUES (SQLERRM, SYSDATE);

END;

/

**Scenario 3: Ensure Data Integrity When Adding a New Customer**

CREATE OR REPLACE PROCEDURE AddNewCustomer(

p\_customer\_id IN NUMBER,

p\_name IN VARCHAR2,

p\_address IN VARCHAR2

) IS

duplicate\_customer EXCEPTION;

PRAGMA EXCEPTION\_INIT(duplicate\_customer, -20003);

BEGIN

-- Insert the new customer

INSERT INTO customers (customer\_id, name, address)

VALUES (p\_customer\_id, p\_name, p\_address);

COMMIT;

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

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

ROLLBACK;

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

INSERT INTO error\_log (error\_message, error\_time)

VALUES ('Duplicate customer ID: ' || p\_customer\_id, SYSDATE);

WHEN OTHERS THEN

ROLLBACK;

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

INSERT INTO error\_log (error\_message, error\_time)

VALUES (SQLERRM, SYSDATE);

END;

/