**Exercise 2: Error Handling**

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

CREATE OR REPLACE PROCEDURE SafeTransferFunds (

p\_from\_account\_id IN accounts.account\_id%TYPE,

p\_to\_account\_id IN accounts.account\_id%TYPE,

p\_amount IN NUMBER

) IS

insufficient\_funds EXCEPTION;

v\_balance\_from accounts.balance%TYPE;

BEGIN

-- Check if the from account has sufficient funds

SELECT balance INTO v\_balance\_from FROM accounts WHERE account\_id = p\_from\_account\_id;

IF v\_balance\_from < p\_amount THEN

RAISE insufficient\_funds;

END IF;

-- Perform the transfer

UPDATE accounts

SET balance = balance - p\_amount

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

UPDATE accounts

SET balance = balance + p\_amount

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

COMMIT;

EXCEPTION

WHEN insufficient\_funds THEN

ROLLBACK;

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

VALUES ('Insufficient funds for account ' || p\_from\_account\_id, SYSDATE);

WHEN OTHERS THEN

ROLLBACK;

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

VALUES (SQLERRM, SYSDATE);

END SafeTransferFunds;

/

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

CREATE OR REPLACE PROCEDURE UpdateSalary (

p\_employee\_id IN employees.employee\_id%TYPE,

p\_percentage IN NUMBER

) IS

employee\_not\_found EXCEPTION;

v\_salary employees.salary%TYPE;

BEGIN

-- Check if the employee exists

SELECT salary INTO v\_salary FROM employees WHERE employee\_id = p\_employee\_id;

-- Update the salary

UPDATE employees

SET salary = salary \* (1 + p\_percentage / 100)

WHERE employee\_id = p\_employee\_id;

COMMIT;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

ROLLBACK;

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

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

WHEN OTHERS THEN

ROLLBACK;

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

VALUES (SQLERRM, SYSDATE);

END UpdateSalary;

/

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

CREATE OR REPLACE PROCEDURE AddNewCustomer (

p\_customer\_id IN customers.customer\_id%TYPE,

p\_customer\_name IN customers.customer\_name%TYPE,

p\_balance IN customers.balance%TYPE

) IS

duplicate\_customer EXCEPTION;

BEGIN

-- Insert the new customer

INSERT INTO customers (customer\_id, customer\_name, balance)

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

COMMIT;

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

ROLLBACK;

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

VALUES ('Customer ID ' || p\_customer\_id || ' already exists.', SYSDATE);

WHEN OTHERS THEN

ROLLBACK;

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

VALUES (SQLERRM, SYSDATE);

END AddNewCustomer;

/