**Exercise 2: Error Handling**

**Scenario 1:** Handle exceptions during fund transfers between accounts.

**Question:** Write a stored procedure SafeTransferFunds that transfers funds between two accounts. Ensure that if any error occurs (e.g., insufficient funds), an appropriate error message is logged and the transaction is rolled back.

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 account\_id = p\_from\_account\_id

FOR UPDATE;

IF v\_balance < p\_amount THEN

RAISE insufficient\_funds;

END IF;

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\_time)

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

WHEN OTHERS THEN

ROLLBACK;

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

VALUES (SQLERRM, SYSDATE);

END SafeTransferFunds;

**Scenario 2:** Manage errors when updating employee salaries.

**Question:** Write a stored procedure UpdateSalary that increases the salary of an employee by a given percentage. If the employee ID does not exist, handle the exception and log an error message.

CREATE OR REPLACE PROCEDURE UpdateSalary (

p\_employee\_id IN NUMBER,

p\_percentage IN NUMBER

) IS

employee\_not\_found EXCEPTION;

v\_salary employees.salary%TYPE;

BEGIN

BEGIN

SELECT salary INTO v\_salary

FROM employees

WHERE employee\_id = p\_employee\_id

FOR UPDATE;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RAISE employee\_not\_found;

END;

UPDATE employees

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

WHERE employee\_id = p\_employee\_id;

COMMIT;

EXCEPTION

WHEN employee\_not\_found THEN

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

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

WHEN OTHERS THEN

ROLLBACK;

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

VALUES (SQLERRM, SYSDATE);

END UpdateSalary;

**Scenario 3:** Ensure data integrity when adding a new customer.

**Question:** Write a stored procedure AddNewCustomer that inserts a new customer into the Customers table. If a customer with the same ID already exists, handle the exception by logging an error and preventing the insertion.

CREATE OR REPLACE PROCEDURE AddNewCustomer (

p\_customer\_id IN NUMBER,

p\_name IN VARCHAR2,

p\_age IN NUMBER,

p\_address IN VARCHAR2

) IS

customer\_exists EXCEPTION;

BEGIN

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

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

COMMIT;

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

ROLLBACK;

RAISE customer\_exists;

WHEN customer\_exists THEN

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

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

WHEN OTHERS THEN

ROLLBACK;

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

VALUES (SQLERRM, SYSDATE);

END AddNewCustomer;