**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.

**Code :**

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

v\_from\_balance ACCOUNTS.BALANCE%TYPE;

v\_to\_balance ACCOUNTS.BALANCE%TYPE;

-- Custom exception for insufficient funds

insufficient\_funds EXCEPTION;

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

-- Procedure to log errors

PROCEDURE LogError(p\_error\_message IN VARCHAR2) IS

BEGIN

INSERT INTO ERROR\_LOG (LOG\_ID, ERROR\_MESSAGE, LOG\_DATE)

VALUES (ERROR\_LOG\_SEQ.NEXTVAL, p\_error\_message, SYSDATE);

END LogError;

BEGIN

-- Lock the rows for the two accounts to prevent concurrent updates

SELECT BALANCE INTO v\_from\_balance

FROM ACCOUNTS

WHERE ACCOUNT\_ID = p\_from\_account\_id

FOR UPDATE;

SELECT BALANCE INTO v\_to\_balance

FROM ACCOUNTS

WHERE ACCOUNT\_ID = p\_to\_account\_id

FOR UPDATE;

-- Check if the from account has sufficient funds

IF v\_from\_balance < p\_amount THEN

RAISE insufficient\_funds;

END IF;

-- Perform the fund 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 the transaction

COMMIT;

EXCEPTION

WHEN insufficient\_funds THEN

-- Log insufficient funds error

LogError('Insufficient funds in account ' || p\_from\_account\_id);

ROLLBACK;

RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient funds in account ' || p\_from\_account\_id);

WHEN OTHERS THEN

-- Log any other errors

LogError(SQLERRM);

ROLLBACK;

RAISE;

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

**CODE :**

CREATE OR REPLACE PROCEDURE UpdateSalary (

p\_employee\_id IN EMPLOYEES.EMPLOYEE\_ID%TYPE,

p\_percentage IN NUMBER

) IS

v\_salary EMPLOYEES.SALARY%TYPE;

-- Procedure to log errors

PROCEDURE LogError(p\_error\_message IN VARCHAR2) IS

BEGIN

INSERT INTO ERROR\_LOG (LOG\_ID, ERROR\_MESSAGE, LOG\_DATE)

VALUES (ERROR\_LOG\_SEQ.NEXTVAL, p\_error\_message, SYSDATE);

END LogError;

BEGIN

-- Select the current salary of the employee

SELECT SALARY INTO v\_salary

FROM EMPLOYEES

WHERE EMPLOYEE\_ID = p\_employee\_id;

-- Update the salary by the given percentage

UPDATE EMPLOYEES

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

WHERE EMPLOYEE\_ID = p\_employee\_id;

-- Commit the transaction

COMMIT;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

-- Log error if employee ID does not exist

LogError('Employee ID ' || p\_employee\_id || ' does not exist.');

RAISE\_APPLICATION\_ERROR(-20001, 'Employee ID ' || p\_employee\_id || ' does not exist.');

WHEN OTHERS THEN

-- Log any other errors

LogError(SQLERRM);

ROLLBACK;

RAISE;

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.

**CODE :**

CREATE OR REPLACE PROCEDURE AddNewCustomer (

p\_customer\_id IN CUSTOMERS.CUSTOMER\_ID%TYPE,

p\_customer\_name IN CUSTOMERS.CUSTOMER\_NAME%TYPE,

p\_date\_of\_birth IN CUSTOMERS.DATE\_OF\_BIRTH%TYPE

) IS

-- Procedure to log errors

PROCEDURE LogError(p\_error\_message IN VARCHAR2) IS

BEGIN

INSERT INTO ERROR\_LOG (LOG\_ID, ERROR\_MESSAGE, LOG\_DATE)

VALUES (ERROR\_LOG\_SEQ.NEXTVAL, p\_error\_message, SYSDATE);

END LogError;

BEGIN

-- Attempt to insert the new customer

INSERT INTO CUSTOMERS (CUSTOMER\_ID, CUSTOMER\_NAME, DATE\_OF\_BIRTH)

VALUES (p\_customer\_id, p\_customer\_name, p\_date\_of\_birth);

-- Commit the transaction

COMMIT;

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

-- Log error if customer with the same ID already exists

LogError('Customer ID ' || p\_customer\_id || ' already exists.');

RAISE\_APPLICATION\_ERROR(-20001, 'Customer ID ' || p\_customer\_id || ' already exists.');

WHEN OTHERS THEN

-- Log any other errors

LogError(SQLERRM);

ROLLBACK;

RAISE;

END AddNewCustomer;