**Exercise 2: Error Handling**

**CREATE TABLE ErrorLog (**

**ErrorMessage VARCHAR2(4000),**

**ErrorDate DATE**

**);**

**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.

**Ans:**

CREATE OR REPLACE PROCEDURE SafeTransferFunds (

  p\_from\_account\_id IN NUMBER,

  p\_to\_account\_id IN NUMBER,

  p\_amount IN NUMBER) AS

  v\_from\_balance NUMBER;

  v\_to\_balance NUMBER;

  v\_error\_message VARCHAR2(4000);

BEGIN

  SELECT Balance INTO v\_from\_balance FROM Accounts WHERE AccountID = p\_from\_account\_id FOR UPDATE;

  IF v\_from\_balance < p\_amount THEN

    RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient funds in the source account.');

  END IF;

  SELECT Balance INTO v\_to\_balance FROM Accounts WHERE AccountID = p\_to\_account\_id FOR UPDATE;

  UPDATE Accounts SET Balance = Balance - p\_amount WHERE AccountID = p\_from\_account\_id;

  UPDATE Accounts SET Balance = Balance + p\_amount WHERE AccountID = p\_to\_account\_id;

  COMMIT;

EXCEPTION

  WHEN OTHERS THEN

    v\_error\_message := SQLERRM;

    ROLLBACK;

    INSERT INTO ErrorLog (ErrorMessage, ErrorDate) VALUES (v\_error\_message, SYSDATE);

    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.

Ans:

CREATE OR REPLACE PROCEDURE UpdateSalary (

  p\_employee\_id IN NUMBER,

  p\_percentage IN NUMBER

) AS

  v\_error\_message VARCHAR2(4000);

BEGIN

  UPDATE Employees

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

  WHERE EmployeeID = p\_employee\_id;

  IF SQL%ROWCOUNT = 0 THEN

    RAISE\_APPLICATION\_ERROR(-20002, 'Employee ID does not exist.');

  END IF;

  COMMIT;

EXCEPTION

  WHEN OTHERS THEN

    ROLLBACK;

    v\_error\_message := SQLERRM;

    INSERT INTO ErrorLog (ErrorMessage, ErrorDate) VALUES (v\_error\_message, 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.

**Ans:**

CREATE OR REPLACE PROCEDURE AddNewCustomer (

  p\_customer\_id IN NUMBER,

  p\_name IN VARCHAR2,

  p\_dob IN DATE,

  p\_balance IN NUMBER,

  p\_last\_modified IN DATE

) AS

  v\_error\_message VARCHAR2(4000);

BEGIN

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

  VALUES (p\_customer\_id, p\_name, p\_dob, p\_balance, p\_last\_modified);

  COMMIT;

EXCEPTION

  WHEN DUP\_VAL\_ON\_INDEX THEN

    v\_error\_message := 'Customer ID already exists.';

    INSERT INTO ErrorLog (ErrorMessage, ErrorDate) VALUES (v\_error\_message, SYSDATE);

    ROLLBACK;

  WHEN OTHERS THEN

    v\_error\_message := SQLERRM;

    INSERT INTO ErrorLog (ErrorMessage, ErrorDate) VALUES (v\_error\_message, SYSDATE);

    ROLLBACK;

END AddNewCustomer;