**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 TABLE ErrorLog (

ErrorID NUMBER PRIMARY KEY,

ErrorMessage VARCHAR2(4000),

ErrorDate DATE

);

CREATE OR REPLACE PROCEDURE SafeTransferFunds(

p\_FromAccountID IN NUMBER,

p\_ToAccountID IN NUMBER,

p\_Amount IN NUMBER

) IS

v\_FromBalance Accounts.Balance%TYPE;

v\_ToBalance Accounts.Balance%TYPE;

BEGIN

-- Fetch balances

SELECT Balance INTO v\_FromBalance FROM Accounts WHERE AccountID = p\_FromAccountID FOR UPDATE;

SELECT Balance INTO v\_ToBalance FROM Accounts WHERE AccountID = p\_ToAccountID FOR UPDATE;

-- Check sufficient funds

IF v\_FromBalance < p\_Amount THEN

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

END IF;

-- Perform transfer

UPDATE Accounts SET Balance = Balance - p\_Amount WHERE AccountID = p\_FromAccountID;

UPDATE Accounts SET Balance = Balance + p\_Amount WHERE AccountID = p\_ToAccountID;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

INSERT INTO ErrorLog (ErrorID, ErrorMessage, ErrorDate)

VALUES (ErrorLog\_SEQ.NEXTVAL, SQLERRM, SYSDATE);

RAISE;

END;

/

**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 SEQUENCE ErrorLog\_SEQ START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE PROCEDURE UpdateSalary(

p\_EmployeeID IN NUMBER,

p\_Percentage IN NUMBER

) IS

v\_Salary Employees.Salary%TYPE;

BEGIN

-- Check if employee exists

SELECT Salary INTO v\_Salary FROM Employees WHERE EmployeeID = p\_EmployeeID FOR UPDATE;

-- Update salary

UPDATE Employees SET Salary = Salary + (Salary \* p\_Percentage / 100) WHERE EmployeeID = p\_EmployeeID;

COMMIT;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

INSERT INTO ErrorLog (ErrorID, ErrorMessage, ErrorDate)

VALUES (ErrorLog\_SEQ.NEXTVAL, 'Employee ID ' || p\_EmployeeID || ' does not exist.', SYSDATE);

WHEN OTHERS THEN

ROLLBACK;

INSERT INTO ErrorLog (ErrorID, ErrorMessage, ErrorDate)

VALUES (ErrorLog\_SEQ.NEXTVAL, SQLERRM, SYSDATE);

RAISE;

END;

/

**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\_CustomerID IN NUMBER,

p\_Name IN VARCHAR2,

p\_DOB IN DATE,

p\_Balance IN NUMBER

) IS

BEGIN

-- Insert new customer

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

VALUES (p\_CustomerID, p\_Name, p\_DOB, p\_Balance, SYSDATE);

COMMIT;

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

INSERT INTO ErrorLog (ErrorID, ErrorMessage, ErrorDate)

VALUES (ErrorLog\_SEQ.NEXTVAL, 'Customer ID ' || p\_CustomerID || ' already exists.', SYSDATE);

WHEN OTHERS THEN

ROLLBACK;

INSERT INTO ErrorLog (ErrorID, ErrorMessage, ErrorDate)

VALUES (ErrorLog\_SEQ.NEXTVAL, SQLERRM, SYSDATE);

RAISE;

END;

/