**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 NUMBER, p\_To NUMBER, p\_Amount NUMBER) IS

v\_FromBalance NUMBER;

BEGIN

SELECT Balance INTO v\_FromBalance FROM Accounts WHERE AccountID = p\_From;

IF v\_FromBalance < p\_Amount THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient Funds');

END IF;

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

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

COMMIT;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

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.

**CODE:**

CREATE OR REPLACE PROCEDURE UpdateSalary(p\_EID NUMBER, p\_Percent NUMBER) IS

BEGIN

UPDATE Employees SET Salary = Salary + Salary \* p\_Percent / 100 WHERE EmployeeID = p\_EID;

IF SQL%NOTFOUND THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Employee not found');

END IF;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

END;

/

SET SERVEROUTPUT ON SIZE 1000000;

EXEC UpdateSalary(101, 10);

EXEC UpdateSalary(999, 5);

EXEC UpdateSalary(101, -20);

**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\_CID NUMBER, p\_Name VARCHAR2, p\_DOB DATE, p\_Balance NUMBER) IS

BEGIN

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified) VALUES (p\_CID, p\_Name, p\_DOB, p\_Balance, SYSDATE);

COMMIT;

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('Customer with ID ' || p\_CID || ' already exists.');

END;

/

EXEC AddNewCustomer(101, 'John Smith', TO\_DATE('1990-05-15', 'YYYY-MM-DD'), 5000);

EXEC AddNewCustomer(101, 'John Smith', TO\_DATE('1990-05-15', 'YYYY-MM-DD'), 5000);

EXEC AddNewCustomer(102, NULL, TO\_DATE('1985-11-20', 'YYYY-MM-DD'), 10000);