**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(

from\_account IN NUMBER,

to\_account IN NUMBER,

transfer\_amt IN NUMBER ) IS

from\_acc\_balance Accounts.Balance%TYPE;

to\_acc\_balance Accounts.Balance%TYPE;

BEGIN

SELECT Balance INTO from\_acc\_balance FROM Accounts

WHERE AccountID = from\_account FOR UPDATE;

SELECT Balance INTO to\_acc\_balance FROM Accounts

WHERE AccountID = to\_account FOR UPDATE;

IF from\_acc\_balance < transfer\_amt THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient funds for transfer.');

END IF;

UPDATE Accounts SET Balance = Balance - transfer\_amt

WHERE AccountID = from\_account;

UPDATE Accounts SET Balance = Balance + transfer\_amt

WHERE AccountID = to\_account;

COMMIT;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Error: One or both account IDs do not exist.');

ROLLBACK;

WHEN OTHERS THEN

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

ROLLBACK;

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 OR REPLACE PROCEDURE UpdateSalary(

emp\_id IN NUMBER,

percent IN NUMBER ) IS

curr\_salary Employees.Salary%TYPE;

BEGIN

SELECT Salary INTO curr\_salary FROM Employees WHERE EmployeeID = emp\_id;

UPDATE Employees SET Salary = Salary + (Salary \* percent / 100)

WHERE EmployeeID = emp\_id;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Salary updated successfully for employee ID ' || emp\_id ||

'. New salary: ' || (curr\_salary + (curr\_salary \* percent / 100)));

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Employee with ID ' || emp\_id || ' does not exist.');

ROLLBACK;

WHEN OTHERS THEN

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

ROLLBACK;

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(

cust\_id IN NUMBER,

cust\_name IN VARCHAR2,

dob IN DATE,

cust\_balance IN NUMBER,

last\_modified IN DATE

) IS

BEGIN

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

VALUES (cust\_id, cust\_name, dob, cust\_balance, last\_modified);

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Customer added successfully. Customer ID: ' || cust\_id);

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('Error: A customer with ID ' || cust\_id || ' already exists.');

ROLLBACK;

WHEN OTHERS THEN

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

ROLLBACK;

END;