# 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.

# Answer:

CREATE OR REPLACE PROCEDURE SafeTransferFunds( p\_from\_account IN NUMBER,

p\_to\_account IN NUMBER, p\_amount IN NUMBER

## ) IS

v\_from\_balance NUMBER; v\_to\_balance NUMBER;

## BEGIN

*-- Check current balance of the from account* SELECT Balance INTO v\_from\_balance FROM Accounts

WHERE AccountID = p\_from\_account;

*-- Check current balance of the to account* SELECT Balance INTO v\_to\_balance FROM Accounts

WHERE AccountID = p\_to\_account;

*-- Ensure sufficient funds*

*IF* v\_from\_balance < p\_amount *THEN*

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

*END IF*;

*-- Perform the transfer*

UPDATE Accounts

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

UPDATE Accounts

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

*-- Commit the transaction*

## COMMIT;

EXCEPTION

*WHEN* NO\_DATA\_FOUND *THEN*

DBMS\_OUTPUT.PUT\_LINE('One of the account IDs does not exist.');

ROLLBACK;

*WHEN* OTHERS *THEN* DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM); ROLLBACK;

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.

# Answer:

CREATE OR REPLACE PROCEDURE UpdateSalary(

p\_employee\_id IN NUMBER, p\_percentage IN NUMBER

## ) IS

v\_current\_salary NUMBER; BEGIN

-- Fetch current salary of the employee SELECT Salary INTO v\_current\_salary FROM Employees

WHERE EmployeeID = p\_employee\_id;

-- Update the salary UPDATE Employees

SET Salary = Salary \* (1 + p\_percentage / 100) WHERE EmployeeID = p\_employee\_id;

-- Commit the transaction COMMIT;

## EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Employee ID does not exist.'); ROLLBACK;

WHEN OTHERS THEN DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM); ROLLBACK;

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.

# Answer:

CREATE OR REPLACE PROCEDURE AddNewCustomer(

p\_customer\_id IN NUMBER, p\_name IN VARCHAR2, p\_dob IN DATE,

p\_balance IN NUMBER

## ) IS BEGIN

-- Attempt to insert a new customer

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified) VALUES (p\_customer\_id, p\_name, p\_dob, p\_balance, SYSDATE);

-- Commit the transaction COMMIT;

## EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('Customer ID already exists.'); ROLLBACK;

WHEN OTHERS THEN DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM); ROLLBACK;

END AddNewCustomer;

/