Scenario 1: SafeTransferFunds - Handle Exceptions During Fund Transfers Between Accounts

CREATE OR REPLACE PROCEDURE SafeTransferFunds(

p\_FromAccountID IN Accounts.AccountID%TYPE,

p\_ToAccountID IN Accounts.AccountID%TYPE,

p\_Amount IN NUMBER

)

IS

v\_FromBalance Accounts.Balance%TYPE;

v\_ToBalance Accounts.Balance%TYPE;

e\_InsufficientFunds EXCEPTION;

e\_AccountNotFound EXCEPTION;

BEGIN

-- Check if both accounts exist and fetch their balances

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

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

-- Check for sufficient funds

IF v\_FromBalance < p\_Amount THEN

RAISE e\_InsufficientFunds;

END IF;

-- Perform the fund transfer

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

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

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Transfer successful');

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error: One or both accounts not found');

WHEN e\_InsufficientFunds THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error: Insufficient funds in the source account');

WHEN OTHERS THEN

ROLLBACK;

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

END;

/

Scenario 2: UpdateSalary - Manage Errors When Updating Employee Salaries

CREATE OR REPLACE PROCEDURE UpdateSalary(

p\_EmployeeID IN Employees.EmployeeID%TYPE,

p\_Percentage IN NUMBER

)

IS

v\_Salary Employees.Salary%TYPE;

e\_EmployeeNotFound EXCEPTION;

BEGIN

-- Check if the employee exists and fetch their current salary

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

-- Update the salary

UPDATE Employees

SET Salary = Salary + (Salary \* p\_Percentage / 100)

WHERE EmployeeID = p\_EmployeeID;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Salary updated successfully');

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error: Employee not found');

WHEN OTHERS THEN

ROLLBACK;

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

END;

/

Scenario 3: AddNewCustomer - Ensure Data Integrity When Adding a New Customer

CREATE OR REPLACE PROCEDURE AddNewCustomer(

p\_CustomerID IN Customers.CustomerID%TYPE,

p\_Name IN Customers.Name%TYPE,

p\_DOB IN Customers.DOB%TYPE,

p\_Balance IN Customers.Balance%TYPE

)

IS

e\_CustomerExists EXCEPTION;

BEGIN

-- Check if the customer already exists

SELECT 1 INTO v\_exists FROM Customers WHERE CustomerID = p\_CustomerID;

-- If no exception, then customer already exists

RAISE e\_CustomerExists;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

-- Insert the new customer

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

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

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Customer added successfully');

WHEN e\_CustomerExists THEN

ROLLBACK;

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

WHEN OTHERS THEN

ROLLBACK;

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

END;

/