**Scenario 1: Safe 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

) AS

v\_FromBalance Accounts.Balance%TYPE;

v\_ToBalance Accounts.Balance%TYPE;

BEGIN

-- Retrieve the current balances of both accounts

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 if the from account has sufficient funds

IF v\_FromBalance < p\_Amount THEN

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

ELSE

-- Deduct the amount from the source account

UPDATE Accounts

SET Balance = Balance - p\_Amount

WHERE AccountID = p\_FromAccountID;

-- Add the amount to the destination account

UPDATE Accounts

SET Balance = Balance + p\_Amount

WHERE AccountID = p\_ToAccountID;

-- Log the successful transfer

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)

VALUES (Transactions\_SEQ.NEXTVAL, p\_FromAccountID, SYSDATE, -p\_Amount, 'Transfer');

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)

VALUES (Transactions\_SEQ.NEXTVAL, p\_ToAccountID, SYSDATE, p\_Amount, 'Transfer');

END IF;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

-- Log the error message

DBMS\_OUTPUT.PUT\_LINE(SQLERRM);

ROLLBACK;

END;

/

**Scenario 2: Manage Errors When Updating Employee Salaries**

CREATE OR REPLACE PROCEDURE UpdateSalary (

p\_EmployeeID IN Employees.EmployeeID%TYPE,

p\_Percentage IN NUMBER

) AS

v\_Salary Employees.Salary%TYPE;

BEGIN

-- Retrieve the current salary

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

-- Increase the salary by the given percentage

UPDATE Employees

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

WHERE EmployeeID = p\_EmployeeID;

COMMIT;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

-- Handle the case where the employee ID does not exist

DBMS\_OUTPUT.PUT\_LINE('Error: Employee ID ' || p\_EmployeeID || ' does not exist.');

WHEN OTHERS THEN

-- Log the error message

DBMS\_OUTPUT.PUT\_LINE(SQLERRM);

ROLLBACK;

END;

/

**Scenario 3: 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

) AS

BEGIN

-- Insert the 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

-- Handle the case where a customer with the same ID already exists

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

WHEN OTHERS THEN

-- Log the error message

DBMS\_OUTPUT.PUT\_LINE(SQLERRM);

ROLLBACK;

END;

/