EXERCISE 7

**Scenario 1: Encapsulate customer-related operations in a package**

CREATE OR REPLACE PACKAGE CustomerOperations AS

PROCEDURE AddCustomer(p\_CustomerID IN NUMBER, p\_Name IN VARCHAR2, p\_DOB IN DATE, p\_Balance IN NUMBER);

PROCEDURE UpdateCustomerBalance(p\_CustomerID IN NUMBER, p\_NewBalance IN NUMBER);

FUNCTION GetCustomerAge(p\_CustomerID IN NUMBER) RETURN NUMBER;

END CustomerOperations;

/

PACKAGE BODY:

CREATE OR REPLACE PACKAGE BODY CustomerOperations AS

PROCEDURE AddCustomer(p\_CustomerID IN NUMBER, p\_Name IN VARCHAR2, p\_DOB IN DATE, p\_Balance IN NUMBER) IS

BEGIN

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

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

EXCEPTION

WHEN OTHERS THEN

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

END AddCustomer;

PROCEDURE UpdateCustomerBalance(p\_CustomerID IN NUMBER, p\_NewBalance IN NUMBER) IS

BEGIN

UPDATE Customers

SET Balance = p\_NewBalance,

LastModified = SYSDATE

WHERE CustomerID = p\_CustomerID;

EXCEPTION

WHEN OTHERS THEN

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

END UpdateCustomerBalance;

FUNCTION GetCustomerAge(p\_CustomerID IN NUMBER) RETURN NUMBER IS

v\_DOB DATE;

v\_Age NUMBER;

BEGIN

SELECT DOB INTO v\_DOB FROM Customers WHERE CustomerID = p\_CustomerID;

v\_Age := TRUNC((SYSDATE - v\_DOB) / 365.25);

RETURN v\_Age;

EXCEPTION

WHEN OTHERS THEN

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

RETURN NULL;

END GetCustomerAge;

END CustomerOperations;

/

**Scenario 2: Create a package for handling employee bonuses**

First, create the package specification:

CREATE OR REPLACE PACKAGE EmployeeBonus AS

PROCEDURE AddBonus(p\_EmployeeID IN NUMBER, p\_BonusAmount IN NUMBER);

PROCEDURE UpdateBonus(p\_EmployeeID IN NUMBER, p\_BonusPercentage IN NUMBER);

FUNCTION GetTotalBonus(p\_EmployeeID IN NUMBER) RETURN NUMBER;

END EmployeeBonus;

/

Then, create the package body:

CREATE OR REPLACE PACKAGE BODY EmployeeBonus AS

PROCEDURE AddBonus(p\_EmployeeID IN NUMBER, p\_BonusAmount IN NUMBER) IS

BEGIN

UPDATE Employees

SET Salary = Salary + p\_BonusAmount

WHERE EmployeeID = p\_EmployeeID;

EXCEPTION

WHEN OTHERS THEN

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

END AddBonus;

PROCEDURE UpdateBonus(p\_EmployeeID IN NUMBER, p\_BonusPercentage IN NUMBER) IS

BEGIN

UPDATE Employees

SET Salary = Salary \* (1 + p\_BonusPercentage / 100)

WHERE EmployeeID = p\_EmployeeID;

EXCEPTION

WHEN OTHERS THEN

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

END UpdateBonus;

FUNCTION GetTotalBonus(p\_EmployeeID IN NUMBER) RETURN NUMBER IS

v\_TotalBonus NUMBER;

BEGIN

SELECT SUM(Bonus) INTO v\_TotalBonus

FROM EmployeeBonuses

WHERE EmployeeID = p\_EmployeeID;

RETURN v\_TotalBonus;

EXCEPTION

WHEN OTHERS THEN

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

RETURN NULL;

END GetTotalBonus;

END EmployeeBonus;

/

**Scenario 3: Develop a package for processing transactions**

First, create the package specification:

CREATE OR REPLACE PACKAGE TransactionProcessing AS

PROCEDURE TransferFunds(p\_FromAccountID IN NUMBER, p\_ToAccountID IN NUMBER, p\_Amount IN NUMBER);

PROCEDURE RecordTransaction(p\_AccountID IN NUMBER, p\_Amount IN NUMBER, p\_TransactionType IN VARCHAR2);

FUNCTION GetAccountBalance(p\_AccountID IN NUMBER) RETURN NUMBER;

END TransactionProcessing;

/

Then, create the package body:

CREATE OR REPLACE PACKAGE BODY TransactionProcessing AS

PROCEDURE TransferFunds(p\_FromAccountID IN NUMBER, p\_ToAccountID IN NUMBER, p\_Amount IN NUMBER) IS

insufficient\_funds EXCEPTION;

BEGIN

DECLARE

v\_FromBalance NUMBER;

BEGIN

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

IF v\_FromBalance < p\_Amount THEN

RAISE insufficient\_funds;

END IF;

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

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

RecordTransaction(p\_FromAccountID, p\_Amount, 'Debit');

RecordTransaction(p\_ToAccountID, p\_Amount, 'Credit');

COMMIT;

EXCEPTION

WHEN insufficient\_funds THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error: Insufficient funds.');

WHEN OTHERS THEN

ROLLBACK;

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

END;

END TransferFunds;

PROCEDURE RecordTransaction(p\_AccountID IN NUMBER, p\_Amount IN NUMBER, p\_TransactionType IN VARCHAR2) IS

BEGIN

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

VALUES (p\_AccountID, p\_Amount, p\_TransactionType, SYSDATE);

EXCEPTION

WHEN OTHERS THEN

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

END RecordTransaction;

FUNCTION GetAccountBalance(p\_AccountID IN NUMBER) RETURN NUMBER IS

v\_Balance NUMBER;

BEGIN

SELECT Balance INTO v\_Balance FROM Accounts WHERE AccountID = p\_AccountID;

RETURN v\_Balance;

EXCEPTION

WHEN OTHERS THEN

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

RETURN NULL;

END GetAccountBalance;

END TransactionProcessing;

/