**Exercise 7: Packages**

**Scenario 1:** Group all customer-related procedures and functions into a package.

* + **Question:** Create a package **CustomerManagement** with procedures for adding a new customer, updating customer details, and a function to get customer balance.

**Scenario1 Code:**

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE Customers';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

/

-- Create Customers table

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

DOB DATE,

Balance NUMBER

);

-- Create CustomerManagement package specification

CREATE OR REPLACE PACKAGE CustomerManagement AS

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

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

FUNCTION GetCustomerBalance(p\_CustomerID NUMBER) RETURN NUMBER;

END CustomerManagement;

/

-- Create CustomerManagement package body

CREATE OR REPLACE PACKAGE BODY CustomerManagement AS

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

BEGIN

INSERT INTO Customers VALUES (p\_CustomerID, p\_Name, p\_DOB, p\_Balance);

END;

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

BEGIN

UPDATE Customers SET Name = p\_Name, DOB = p\_DOB, Balance = p\_Balance

WHERE CustomerID = p\_CustomerID;

END;

FUNCTION GetCustomerBalance(p\_CustomerID NUMBER) RETURN NUMBER IS

v\_balance NUMBER;

BEGIN

SELECT Balance INTO v\_balance FROM Customers WHERE CustomerID = p\_CustomerID;

RETURN v\_balance;

END;

END CustomerManagement;

/

-- Test CustomerManagement

BEGIN

CustomerManagement.AddCustomer(1, 'John Doe', DATE '1990-01-01', 5000);

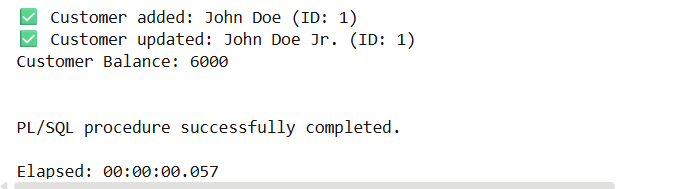
CustomerManagement.UpdateCustomer(1, 'John Doe Jr.', DATE '1990-01-01', 6000);

DBMS\_OUTPUT.PUT\_LINE('Customer Balance: ' || CustomerManagement.GetCustomerBalance(1));

END;

/

**Scenario1 Output:**

****

**Scenario 2:** Create a package to manage employee data.

* **Question:** Write a package **EmployeeManagement** with procedures to hire new employees, update employee details, and a function to calculate annual salary.

**Scenario 2 Code:**

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE Employees';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

/

-- Create Employees table

CREATE TABLE Employees (

EmployeeID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

Salary NUMBER

);

-- Create EmployeeManagement package specification

CREATE OR REPLACE PACKAGE EmployeeManagement AS

PROCEDURE HireEmployee(p\_EmployeeID NUMBER, p\_Name VARCHAR2, p\_Salary NUMBER);

PROCEDURE UpdateEmployee(p\_EmployeeID NUMBER, p\_Name VARCHAR2, p\_Salary NUMBER);

FUNCTION CalculateAnnualSalary(p\_EmployeeID NUMBER) RETURN NUMBER;

END EmployeeManagement;

/

-- Create EmployeeManagement package body

CREATE OR REPLACE PACKAGE BODY EmployeeManagement AS

PROCEDURE HireEmployee(p\_EmployeeID NUMBER, p\_Name VARCHAR2, p\_Salary NUMBER) IS

BEGIN

INSERT INTO Employees VALUES (p\_EmployeeID, p\_Name, p\_Salary);

END;

PROCEDURE UpdateEmployee(p\_EmployeeID NUMBER, p\_Name VARCHAR2, p\_Salary NUMBER) IS

BEGIN

UPDATE Employees SET Name = p\_Name, Salary = p\_Salary

WHERE EmployeeID = p\_EmployeeID;

END;

FUNCTION CalculateAnnualSalary(p\_EmployeeID NUMBER) RETURN NUMBER IS

v\_salary NUMBER;

BEGIN

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

RETURN v\_salary \* 12;

END;

END EmployeeManagement;

/

-- Test EmployeeManagement

BEGIN

EmployeeManagement.HireEmployee(101, 'Alice', 50000);

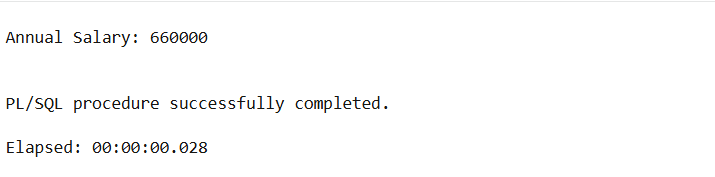
EmployeeManagement.UpdateEmployee(101, 'Alice Smith', 55000);

DBMS\_OUTPUT.PUT\_LINE('AnnualSalary : ' ||EmployeeManagement.CalculateAnnualSalary(101));

END;

/

**Scenario 2 output:**

****

**Scenario 3:** Group all account-related operations into a package.

* **Question:** Create a package **AccountOperations** with procedures for opening a new account, closing an account, and a function to get the total balance of a customer across all accounts.

**Scenario 3 code:**

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE Accounts';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

/

-- Create Accounts table

CREATE TABLE Accounts (

AccountID NUMBER PRIMARY KEY,

CustomerID NUMBER,

Balance NUMBER

);

-- Create AccountOperations package specification

CREATE OR REPLACE PACKAGE AccountOperations AS

PROCEDURE OpenAccount(p\_AccountID NUMBER, p\_CustomerID NUMBER, p\_Balance NUMBER);

PROCEDURE CloseAccount(p\_AccountID NUMBER);

FUNCTION GetTotalBalance(p\_CustomerID NUMBER) RETURN NUMBER;

END AccountOperations;

/

-- Create AccountOperations package body

CREATE OR REPLACE PACKAGE BODY AccountOperations AS

PROCEDURE OpenAccount(p\_AccountID NUMBER, p\_CustomerID NUMBER, p\_Balance NUMBER) IS

BEGIN

INSERT INTO Accounts VALUES (p\_AccountID, p\_CustomerID, p\_Balance);

END;

PROCEDURE CloseAccount(p\_AccountID NUMBER) IS

BEGIN

DELETE FROM Accounts WHERE AccountID = p\_AccountID;

END;

FUNCTION GetTotalBalance(p\_CustomerID NUMBER) RETURN NUMBER IS

v\_total NUMBER := 0;

BEGIN

SELECT NVL(SUM(Balance),0) INTO v\_total FROM Accounts WHERE CustomerID = p\_CustomerID;

RETURN v\_total;

END;

END AccountOperations;

/

-- Test AccountOperations

BEGIN

AccountOperations.OpenAccount(201, 1, 3000);

AccountOperations.OpenAccount(202, 1, 4000);

DBMS\_OUTPUT.PUT\_LINE('Total Balance for Customer 1: ' || AccountOperations.GetTotalBalance(1));

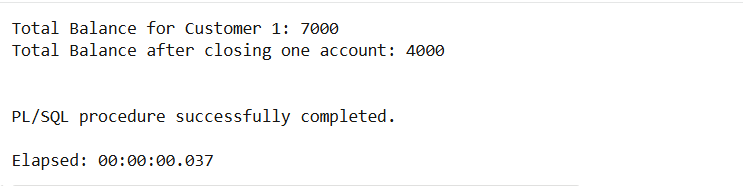
AccountOperations.CloseAccount(201);

DBMS\_OUTPUT.PUT\_LINE('Total Balance after closing one account: ' || AccountOperations.GetTotalBalance(1));

END;

/

**Scenario 3 output:**

****