**Exercise 7: Packages**

**Scenario 1: (Ex7-Scenario1.sql)**

@InitializeData.sql

SET ECHO ON

SET SERVEROUTPUT ON SIZE UNLIMITED

SPOOL output-Ex7-Scenario1.txt

VARIABLE input VARCHAR2(30)

*-- Package Specification*

CREATE OR REPLACE PACKAGE CustomerManagement AS

    PROCEDURE AddCustomer(

        p\_CustomerID IN CUSTOMERS.CUSTOMERID%TYPE,

        p\_Name IN CUSTOMERS.NAME%TYPE,

        p\_DOB IN CUSTOMERS.DOB%TYPE,

        p\_Balance IN CUSTOMERS.BALANCE%TYPE

    );

    PROCEDURE UpdateCustomer(

        p\_CustomerID IN CUSTOMERS.CUSTOMERID%TYPE,

        p\_Name IN CUSTOMERS.NAME%TYPE,

        p\_Balance IN CUSTOMERS.BALANCE%TYPE

    );

    FUNCTION GetCustomerBalance(

        p\_CustomerID IN CUSTOMERS.CUSTOMERID%TYPE

    ) RETURN CUSTOMERS.BALANCE%TYPE;

END CustomerManagement;

/

*-- Package Body*

CREATE OR REPLACE PACKAGE BODY CustomerManagement AS

    PROCEDURE AddCustomer(

        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

    BEGIN

        INSERT INTO CUSTOMERS (

            CUSTOMERID, NAME, DOB, BALANCE, LASTMODIFIED

        ) VALUES (

            p\_CustomerID, p\_Name, p\_DOB, p\_Balance, SYSDATE

        );

    EXCEPTION

        WHEN DUP\_VAL\_ON\_INDEX THEN

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

    END AddCustomer;

    PROCEDURE UpdateCustomer(

        p\_CustomerID IN CUSTOMERS.CUSTOMERID%TYPE,

        p\_Name IN CUSTOMERS.NAME%TYPE,

        p\_Balance IN CUSTOMERS.BALANCE%TYPE

    ) IS

    BEGIN

        UPDATE CUSTOMERS

        SET NAME = p\_Name, BALANCE = p\_Balance, LASTMODIFIED = SYSDATE

        WHERE CUSTOMERID = p\_CustomerID;

        IF SQL%ROWCOUNT = 0 THEN

            DBMS\_OUTPUT.PUT\_LINE('Customer ID not found.');

        END IF;

    END UpdateCustomer;

    FUNCTION GetCustomerBalance(

        p\_CustomerID IN CUSTOMERS.CUSTOMERID%TYPE

    ) RETURN CUSTOMERS.BALANCE%TYPE IS

        v\_Balance CUSTOMERS.BALANCE%TYPE;

    BEGIN

        SELECT BALANCE INTO v\_Balance

        FROM CUSTOMERS

        WHERE CUSTOMERID = p\_CustomerID;

        RETURN v\_Balance;

    EXCEPTION

        WHEN NO\_DATA\_FOUND THEN

            DBMS\_OUTPUT.PUT\_LINE('Customer ID not found.');

            RETURN NULL;

    END GetCustomerBalance;

END CustomerManagement;

/

*-- Before Using the Package*

SELECT \* FROM Customers;

*-- Test Package Procedures and Function*

BEGIN

*-- Add a new customer*

    CustomerManagement.AddCustomer(

        p\_CustomerID => 2001,

        p\_Name => 'Alice Johnson',

        p\_DOB => TO\_DATE('1985-06-15', 'YYYY-MM-DD'),

        p\_Balance => 3000

    );

*-- Update an existing customer*

    CustomerManagement.UpdateCustomer(

        p\_CustomerID => 2001,

        p\_Name => 'Alice Thompson',

        p\_Balance => 3500

    );

*-- Get customer balance*

    DECLARE

        v\_Balance CUSTOMERS.BALANCE%TYPE;

    BEGIN

        v\_Balance := CustomerManagement.GetCustomerBalance(2001);

        DBMS\_OUTPUT.PUT\_LINE('Balance for customer 2001: ' || v\_Balance);

    END;

END;

/

*-- After Using the Package*

SELECT \* FROM Customers;

SPOOL OFF

@DropData.sql

**Scenario 2: (Ex7-Scenario2.sql)**

@InitializeData.sql

SET ECHO ON

SET SERVEROUTPUT ON SIZE UNLIMITED

SPOOL output-Ex7-Scenario2.txt

VARIABLE input VARCHAR2(30)

*-- Package Specification*

CREATE OR REPLACE PACKAGE EmployeeManagement AS

    PROCEDURE HireEmployee(

        p\_EmployeeID IN EMPLOYEES.EMPLOYEEID%TYPE,

        p\_Name IN EMPLOYEES.NAME%TYPE,

        p\_Position IN EMPLOYEES.POSITION%TYPE,

        p\_Salary IN EMPLOYEES.SALARY%TYPE,

        p\_Department IN EMPLOYEES.DEPARTMENT%TYPE

    );

    PROCEDURE UpdateEmployee(

        p\_EmployeeID IN EMPLOYEES.EMPLOYEEID%TYPE,

        p\_Name IN EMPLOYEES.NAME%TYPE,

        p\_Position IN EMPLOYEES.POSITION%TYPE,

        p\_Salary IN EMPLOYEES.SALARY%TYPE,

        p\_Department IN EMPLOYEES.DEPARTMENT%TYPE

    );

    FUNCTION CalculateAnnualSalary(

        p\_EmployeeID IN EMPLOYEES.EMPLOYEEID%TYPE

    ) RETURN NUMBER;

END EmployeeManagement;

/

*-- Package Body*

CREATE OR REPLACE PACKAGE BODY EmployeeManagement AS

    PROCEDURE HireEmployee(

        p\_EmployeeID IN EMPLOYEES.EMPLOYEEID%TYPE,

        p\_Name IN EMPLOYEES.NAME%TYPE,

        p\_Position IN EMPLOYEES.POSITION%TYPE,

        p\_Salary IN EMPLOYEES.SALARY%TYPE,

        p\_Department IN EMPLOYEES.DEPARTMENT%TYPE

    ) IS

    BEGIN

        INSERT INTO EMPLOYEES (

            EMPLOYEEID, NAME, POSITION, SALARY, DEPARTMENT, HIREDATE

        ) VALUES (

            p\_EmployeeID, p\_Name, p\_Position, p\_Salary, p\_Department, SYSDATE

        );

    EXCEPTION

        WHEN DUP\_VAL\_ON\_INDEX THEN

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

    END HireEmployee;

    PROCEDURE UpdateEmployee(

        p\_EmployeeID IN EMPLOYEES.EMPLOYEEID%TYPE,

        p\_Name IN EMPLOYEES.NAME%TYPE,

        p\_Position IN EMPLOYEES.POSITION%TYPE,

        p\_Salary IN EMPLOYEES.SALARY%TYPE,

        p\_Department IN EMPLOYEES.DEPARTMENT%TYPE

    ) IS

    BEGIN

        UPDATE EMPLOYEES

        SET NAME = p\_Name, POSITION = p\_Position, SALARY = p\_Salary, DEPARTMENT = p\_Department

        WHERE EMPLOYEEID = p\_EmployeeID;

        IF SQL%ROWCOUNT = 0 THEN

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

        END IF;

    END UpdateEmployee;

    FUNCTION CalculateAnnualSalary(

        p\_EmployeeID IN EMPLOYEES.EMPLOYEEID%TYPE

    ) RETURN NUMBER IS

        v\_Salary EMPLOYEES.SALARY%TYPE;

        v\_AnnualSalary NUMBER;

    BEGIN

        SELECT SALARY INTO v\_Salary

        FROM EMPLOYEES

        WHERE EMPLOYEEID = p\_EmployeeID;

        v\_AnnualSalary := v\_Salary \* 12;

        RETURN v\_AnnualSalary;

    EXCEPTION

        WHEN NO\_DATA\_FOUND THEN

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

            RETURN NULL;

    END CalculateAnnualSalary;

END EmployeeManagement;

/

*-- Before Using the Package*

SELECT \* FROM Employees;

*-- Test Package Procedures and Function*

BEGIN

*-- Hire a new employee*

    EmployeeManagement.HireEmployee(

        p\_EmployeeID => 5,

        p\_Name => 'Rick Brown',

        p\_Position => 'Developer',

        p\_Salary => 7000,

        p\_Department => 'IT'

    );

*-- Update employee details*

    EmployeeManagement.UpdateEmployee(

        p\_EmployeeID => 2,

        p\_Name => 'Dan Smith',

        p\_Position => 'Senior Developer',

        p\_Salary => 8000,

        p\_Department => 'IT'

    );

*-- Calculate annual salary*

    DECLARE

        v\_AnnualSalary NUMBER;

    BEGIN

        v\_AnnualSalary := EmployeeManagement.CalculateAnnualSalary(3001);

        DBMS\_OUTPUT.PUT\_LINE('Annual salary for employee 3001: ' || v\_AnnualSalary);

    END;

END;

/

*-- After Using the Package*

SELECT \* FROM Employees;

SPOOL OFF

@DropData.sql

**Scenario 3: (Ex7-Scenario3.sql)**

@InitializeData.sql

SET ECHO ON

SET SERVEROUTPUT ON SIZE UNLIMITED

SPOOL output-Ex7-Scenario3.txt

VARIABLE input VARCHAR2(30)

*-- Package Specification*

CREATE OR REPLACE PACKAGE AccountOperations AS

    PROCEDURE OpenAccount(

        p\_AccountID IN ACCOUNTS.ACCOUNTID%TYPE,

        p\_CustomerID IN ACCOUNTS.CUSTOMERID%TYPE,

        p\_AccountType IN ACCOUNTS.ACCOUNTTYPE%TYPE,

        p\_Balance IN ACCOUNTS.BALANCE%TYPE

    );

    PROCEDURE CloseAccount(

        p\_AccountID IN ACCOUNTS.ACCOUNTID%TYPE

    );

    FUNCTION GetTotalBalance(

        p\_CustomerID IN ACCOUNTS.CUSTOMERID%TYPE

    ) RETURN NUMBER;

END AccountOperations;

/

*-- Package Body*

CREATE OR REPLACE PACKAGE BODY AccountOperations AS

    PROCEDURE OpenAccount(

        p\_AccountID IN ACCOUNTS.ACCOUNTID%TYPE,

        p\_CustomerID IN ACCOUNTS.CUSTOMERID%TYPE,

        p\_AccountType IN ACCOUNTS.ACCOUNTTYPE%TYPE,

        p\_Balance IN ACCOUNTS.BALANCE%TYPE

    ) IS

    BEGIN

        INSERT INTO ACCOUNTS (

            ACCOUNTID, CUSTOMERID, ACCOUNTTYPE, BALANCE, LASTMODIFIED

        ) VALUES (

            p\_AccountID, p\_CustomerID, p\_AccountType, p\_Balance, SYSDATE

        );

    EXCEPTION

        WHEN DUP\_VAL\_ON\_INDEX THEN

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

    END OpenAccount;

    PROCEDURE CloseAccount(

        p\_AccountID IN ACCOUNTS.ACCOUNTID%TYPE

    ) IS

    BEGIN

        DELETE FROM ACCOUNTS

        WHERE ACCOUNTID = p\_AccountID;

        IF SQL%ROWCOUNT = 0 THEN

            DBMS\_OUTPUT.PUT\_LINE('Account ID not found.');

        END IF;

    END CloseAccount;

    FUNCTION GetTotalBalance(

        p\_CustomerID IN ACCOUNTS.CUSTOMERID%TYPE

    ) RETURN NUMBER IS

        v\_TotalBalance NUMBER := 0;

    BEGIN

        SELECT SUM(BALANCE) INTO v\_TotalBalance

        FROM ACCOUNTS

        WHERE CUSTOMERID = p\_CustomerID;

        RETURN v\_TotalBalance;

    EXCEPTION

        WHEN NO\_DATA\_FOUND THEN

            DBMS\_OUTPUT.PUT\_LINE('No accounts found for the customer.');

            RETURN NULL;

    END GetTotalBalance;

END AccountOperations;

/

*-- Before Using the Package*

SELECT \* FROM Accounts;

*-- Test Package Procedures and Function*

BEGIN

*-- Open a new account*

    AccountOperations.OpenAccount(

        p\_AccountID => 4001,

        p\_CustomerID => 1,

        p\_AccountType => 'Savings',

        p\_Balance => 5000

    );

*-- Close an account*

    AccountOperations.CloseAccount(4001);

*-- Get total balance for a customer*

    DECLARE

        v\_TotalBalance NUMBER;

    BEGIN

        v\_TotalBalance := AccountOperations.GetTotalBalance(2);

        DBMS\_OUTPUT.PUT\_LINE('Total balance for customer 2: ' || v\_TotalBalance);

    END;

END;

/

*-- After Using the Package*

SELECT \* FROM Accounts;

SPOOL OFF

@DropData.sql