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
```