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.

```
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 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 (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 with this ID already exists.');
    END AddCustomer;

    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, LastModified =
SYSDATE
        WHERE CustomerID = p_CustomerID;
        IF SQL%ROWCOUNT = 0 THEN
            DBMS_OUTPUT.PUT_LINE('Customer not found.');
        END IF;
    END UpdateCustomer;

    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;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RETURN NULL;
    END GetCustomerBalance;
END CustomerManagement;
```

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.

```
CREATE OR REPLACE PACKAGE EmployeeManagement AS
    PROCEDURE HireEmployee(p_EmployeeID NUMBER, p_Name VARCHAR2, p_Position
VARCHAR2, p_Salary NUMBER, p_Department VARCHAR2, p_HireDate DATE);
    PROCEDURE UpdateEmployee(p_EmployeeID NUMBER, p_Name VARCHAR2, p_Position
VARCHAR2, p_Salary NUMBER, p_Department VARCHAR2);
    FUNCTION CalculateAnnualSalary(p_EmployeeID NUMBER) RETURN NUMBER;
END EmployeeManagement;

CREATE OR REPLACE PACKAGE BODY EmployeeManagement AS
    PROCEDURE HireEmployee(p_EmployeeID NUMBER, p_Name VARCHAR2, p_Position
VARCHAR2, p_Salary NUMBER, p_Department VARCHAR2, p_HireDate DATE) IS
    BEGIN
        INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department,
HireDate)
        VALUES (p_EmployeeID, p_Name, p_Position, p_Salary, p_Department,
p_HireDate);
    EXCEPTION
        WHEN DUP_VAL_ON_INDEX THEN
            DBMS_OUTPUT.PUT_LINE('Employee with this ID already exists.');
    END HireEmployee;

    PROCEDURE UpdateEmployee(p_EmployeeID NUMBER, p_Name VARCHAR2, p_Position
VARCHAR2, p_Salary NUMBER, p_Department VARCHAR2) 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 not found.');
        END IF;
    END UpdateEmployee;

    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;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RETURN NULL;
    END CalculateAnnualSalary;
END EmployeeManagement;
```

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.

```
 CREATE OR REPLACE PACKAGE AccountOperations AS
    PROCEDURE OpenAccount(p_AccountID NUMBER, p_CustomerID NUMBER, p_AccountType
VARCHAR2, p_Balance NUMBER);
    PROCEDURE CloseAccount(p_AccountID NUMBER);
    FUNCTION GetTotalBalance(p_CustomerID NUMBER) RETURN NUMBER;
END AccountOperations;

CREATE OR REPLACE PACKAGE BODY AccountOperations AS
    PROCEDURE OpenAccount(p_AccountID NUMBER, p_CustomerID NUMBER, p_AccountType
VARCHAR2, p_Balance NUMBER) 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 with this ID already exists.');
    END OpenAccount;

    PROCEDURE CloseAccount(p_AccountID NUMBER) IS
    BEGIN
        DELETE FROM Accounts
        WHERE AccountID = p_AccountID;
        IF SQL%ROWCOUNT = 0 THEN
            DBMS_OUTPUT.PUT_LINE('Account not found.');
        END IF;
    END CloseAccount;

    FUNCTION GetTotalBalance(p_CustomerID NUMBER) RETURN NUMBER IS
        v_totalBalance NUMBER;
    BEGIN
        SELECT SUM(Balance) INTO v_totalBalance
        FROM Accounts
        WHERE CustomerID = p_CustomerID;
        RETURN v_totalBalance;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RETURN 0;
    END GetTotalBalance;
END AccountOperations;
```