**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.

**Ans:**

CREATE OR REPLACE PACKAGE CustomerManagement AS

  PROCEDURE AddCustomer(

    p\_customerid IN NUMBER,

    p\_name IN VARCHAR2,

    p\_dob IN DATE,

    p\_balance IN NUMBER,

    p\_lastmodified IN DATE,

    p\_isvip IN CHAR

  );

  PROCEDURE UpdateCustomer(

    p\_customerid IN NUMBER,

    p\_name IN VARCHAR2,

    p\_dob IN DATE,

    p\_balance IN NUMBER,

    p\_lastmodified IN DATE,

    p\_isvip IN CHAR

  );

  FUNCTION GetCustomerBalance(p\_customerid IN NUMBER) RETURN NUMBER;

END CustomerManagement;

/

SELECT COLUMN\_NAME

FROM ALL\_TAB\_COLUMNS

WHERE TABLE\_NAME = 'CUSTOMERS';

CREATE OR REPLACE PACKAGE BODY CustomerManagement AS

  PROCEDURE AddCustomer(

    p\_customerid IN NUMBER,

    p\_name IN VARCHAR2,

    p\_dob IN DATE,

    p\_balance IN NUMBER,

    p\_lastmodified IN DATE,

    p\_isvip IN CHAR

  ) IS

  BEGIN

    INSERT INTO Customers (

      CUSTOMERID,

      NAME,

      DOB,

      BALANCE,

      LASTMODIFIED,

      ISVIP

    ) VALUES (

      p\_customerid,

      p\_name,

      p\_dob,

      p\_balance,

      p\_lastmodified,

      p\_isvip

    );

  EXCEPTION

    WHEN OTHERS THEN

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

  END;

  PROCEDURE UpdateCustomer(

    p\_customerid IN NUMBER,

    p\_name IN VARCHAR2,

    p\_dob IN DATE,

    p\_balance IN NUMBER,

    p\_lastmodified IN DATE,

    p\_isvip IN CHAR

  ) IS

  BEGIN

    UPDATE Customers

    SET

      NAME = p\_name,

      DOB = p\_dob,

      BALANCE = p\_balance,

      LASTMODIFIED = p\_lastmodified,

      ISVIP = p\_isvip

    WHERE CUSTOMERID = p\_customerid;

  EXCEPTION

    WHEN OTHERS THEN

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

  END;

  FUNCTION GetCustomerBalance(p\_customerid IN 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;

    WHEN OTHERS THEN

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

      RETURN NULL;

  END;

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.

**Ans:**

CREATE OR REPLACE PACKAGE EmployeeManagement AS

  PROCEDURE HireEmployee(p\_employee\_id IN NUMBER, p\_name IN VARCHAR2, p\_position IN VARCHAR2,

                         p\_salary IN NUMBER, p\_department IN VARCHAR2, p\_hire\_date IN DATE);

  PROCEDURE UpdateEmployee(p\_employee\_id IN NUMBER, p\_name IN VARCHAR2, p\_position IN VARCHAR2,

                           p\_salary IN NUMBER, p\_department IN VARCHAR2);

  FUNCTION CalculateAnnualSalary(p\_employee\_id IN NUMBER) RETURN NUMBER;

END EmployeeManagement;

/

CREATE OR REPLACE PACKAGE BODY EmployeeManagement AS

  PROCEDURE HireEmployee(p\_employee\_id IN NUMBER, p\_name IN VARCHAR2, p\_position IN VARCHAR2,

                         p\_salary IN NUMBER, p\_department IN VARCHAR2, p\_hire\_date IN DATE) IS

  BEGIN

    INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)

    VALUES (p\_employee\_id, p\_name, p\_position, p\_salary, p\_department, p\_hire\_date);

  EXCEPTION

    WHEN OTHERS THEN

      DBMS\_OUTPUT.PUT\_LINE('An error occurred while hiring employee: ' || SQLERRM);

  END HireEmployee;

  PROCEDURE UpdateEmployee(p\_employee\_id IN NUMBER, p\_name IN VARCHAR2, p\_position IN VARCHAR2,

                           p\_salary IN NUMBER, p\_department IN VARCHAR2) IS

  BEGIN

    UPDATE Employees

    SET Name = p\_name,

        Position = p\_position,

        Salary = p\_salary,

        Department = p\_department

    WHERE EmployeeID = p\_employee\_id;

  EXCEPTION

    WHEN OTHERS THEN

      DBMS\_OUTPUT.PUT\_LINE('An error occurred while updating employee: ' || SQLERRM);

  END UpdateEmployee;

  FUNCTION CalculateAnnualSalary(p\_employee\_id IN NUMBER) RETURN NUMBER IS

    v\_salary NUMBER;

  BEGIN

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

    RETURN v\_salary \* 12;

  EXCEPTION

    WHEN NO\_DATA\_FOUND THEN

      RETURN 0; -- Handle case where the employee is not found

    WHEN OTHERS THEN

      DBMS\_OUTPUT.PUT\_LINE('An error occurred while calculating annual salary: ' || SQLERRM);

      RETURN 0;

  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.

**Ans:**

CREATE OR REPLACE PACKAGE AccountOperations AS

  PROCEDURE OpenAccount(p\_customer\_id IN NUMBER, p\_initial\_balance IN NUMBER, p\_account\_type IN VARCHAR2);

  PROCEDURE CloseAccount(p\_account\_id IN NUMBER);

  FUNCTION GetTotalBalance(p\_customer\_id IN NUMBER) RETURN NUMBER;

END AccountOperations;

/

CREATE OR REPLACE PACKAGE BODY AccountOperations AS

  PROCEDURE OpenAccount(p\_customer\_id IN NUMBER, p\_initial\_balance IN NUMBER, p\_account\_type IN VARCHAR2) IS

  BEGIN

    INSERT INTO Accounts (CustomerID, Balance, AccountType, DateOpened)

    VALUES (p\_customer\_id, p\_initial\_balance, p\_account\_type, SYSDATE);

    COMMIT;

  EXCEPTION

    WHEN OTHERS THEN

      ROLLBACK;

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

  END OpenAccount;

  PROCEDURE CloseAccount(p\_account\_id IN NUMBER) IS

  BEGIN

    DELETE FROM Accounts

    WHERE AccountID = p\_account\_id;

    COMMIT;

  EXCEPTION

    WHEN OTHERS THEN

      ROLLBACK;

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

  END CloseAccount;

  FUNCTION GetTotalBalance(p\_customer\_id IN NUMBER) RETURN NUMBER IS

    v\_total\_balance NUMBER;

  BEGIN

    SELECT SUM(Balance)

    INTO v\_total\_balance

    FROM Accounts

    WHERE CustomerID = p\_customer\_id;

    RETURN v\_total\_balance;

  EXCEPTION

    WHEN NO\_DATA\_FOUND THEN

      RETURN 0;

    WHEN OTHERS THEN

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

      RETURN NULL;

  END GetTotalBalance;

END AccountOperations;