**Exercise 7: Packages**

**Scenario 1:**

**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\_id NUMBER, p\_name VARCHAR2, p\_dob DATE, p\_balance NUMBER);

  PROCEDURE UpdateCustomer(p\_id NUMBER, p\_name VARCHAR2, p\_balance NUMBER);

  FUNCTION GetCustomerBalance(p\_id NUMBER) RETURN NUMBER;

END CustomerManagement;

/

CREATE OR REPLACE PACKAGE BODY CustomerManagement AS

  PROCEDURE AddCustomer(p\_id NUMBER, p\_name VARCHAR2, p\_dob DATE, p\_balance NUMBER) IS

  BEGIN

    INSERT INTO Customers VALUES (p\_id, p\_name, p\_dob, p\_balance, SYSDATE);

DBMS\_OUTPUT.PUT\_LINE('Customer added.');

EXCEPTION

    WHEN DUP\_VAL\_ON\_INDEX THEN

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

  END;

  PROCEDURE UpdateCustomer(p\_id NUMBER, p\_name VARCHAR2, p\_balance NUMBER) IS

  BEGIN

    UPDATE Customers

    SET Name = p\_name,

        Balance = p\_balance,

        LastModified = SYSDATE

    WHERE CustomerID = p\_id;

    DBMS\_OUTPUT.PUT\_LINE('Customer updated.');

  END;

  FUNCTION GetCustomerBalance(p\_id NUMBER) RETURN NUMBER IS

    v\_balance NUMBER;

  BEGIN

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

    RETURN v\_balance;

  EXCEPTION

    WHEN NO\_DATA\_FOUND THEN

      RETURN -1;

  END;

END CustomerManagement;

/

SET SERVEROUTPUT ON;

BEGIN

  CustomerManagement.AddCustomer(2, 'Mary Jane', TO\_DATE('1990-02-15','YYYY-MM-DD'), 3000);

  DBMS\_OUTPUT.PUT\_LINE('Balance: ' || CustomerManagement.GetCustomerBalance(2));

END

/

**Output**



**Scenario 2:**

**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\_id NUMBER, p\_name VARCHAR2, p\_pos VARCHAR2, p\_salary NUMBER, p\_dept VARCHAR2);

  PROCEDURE UpdateEmployee(p\_id NUMBER, p\_salary NUMBER);

  FUNCTION GetAnnualSalary(p\_id NUMBER) RETURN NUMBER;

END EmployeeManagement;

/

CREATE OR REPLACE PACKAGE BODY EmployeeManagement AS

  PROCEDURE HireEmployee(p\_id NUMBER, p\_name VARCHAR2, p\_pos VARCHAR2, p\_salary NUMBER, p\_dept VARCHAR2) IS

  BEGIN

    INSERT INTO Employees VALUES (p\_id, p\_name, p\_pos, p\_salary, p\_dept, SYSDATE);

    DBMS\_OUTPUT.PUT\_LINE('Employee hired.');

  EXCEPTION

    WHEN DUP\_VAL\_ON\_INDEX THEN

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

  END;

  PROCEDURE UpdateEmployee(p\_id NUMBER, p\_salary NUMBER) IS

  BEGIN

    UPDATE Employees SET Salary = p\_salary WHERE EmployeeID = p\_id;

    DBMS\_OUTPUT.PUT\_LINE('Salary updated.');

  END;

  FUNCTION GetAnnualSalary(p\_id NUMBER) RETURN NUMBER IS

    v\_salary NUMBER;

  BEGIN

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

    RETURN v\_salary \* 12;

  EXCEPTION

    WHEN NO\_DATA\_FOUND THEN

      RETURN -1;

  END;

END EmployeeManagement;

/

SET SERVEROUTPUT ON;

BEGIN

EmployeeManagement.HireEmployee(2, 'Bob', 'Developer', 50000, 'IT');

  DBMS\_OUTPUT.PUT\_LINE('Annual Salary: ' || EmployeeManagement.GetAnnualSalary(2));

END

/

**Output**



**Scenario 3:**

**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\_id NUMBER, p\_cust\_id NUMBER, p\_type VARCHAR2, p\_balance NUMBER);

  PROCEDURE CloseAccount(p\_id NUMBER);

  FUNCTION GetTotalBalance(p\_cust\_id NUMBER) RETURN NUMBER;

END AccountOperations;

/

CREATE OR REPLACE PACKAGE BODY AccountOperations AS

  PROCEDURE OpenAccount(p\_id NUMBER, p\_cust\_id NUMBER, p\_type VARCHAR2, p\_balance NUMBER) IS

  BEGIN

    INSERT INTO Accounts VALUES (p\_id, p\_cust\_id, p\_type, p\_balance, SYSDATE);

    DBMS\_OUTPUT.PUT\_LINE('Account opened.');

  EXCEPTION

    WHEN DUP\_VAL\_ON\_INDEX THEN

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

  END;

  PROCEDURE CloseAccount(p\_id NUMBER) IS

  BEGIN

    DELETE FROM Accounts WHERE AccountID = p\_id;

    DBMS\_OUTPUT.PUT\_LINE('Account closed.');

  END;

  FUNCTION GetTotalBalance(p\_cust\_id NUMBER) RETURN NUMBER IS

    v\_total NUMBER := 0;

  BEGIN

    SELECT SUM(Balance) INTO v\_total FROM Accounts WHERE CustomerID = p\_cust\_id;

    RETURN NVL(v\_total, 0);

  END;

END AccountOperations;

/

SET SERVEROUTPUT ON;

BEGIN

AccountOperations.OpenAccount(2, 2, 'Checking', 2000);

  DBMS\_OUTPUT.PUT\_LINE('Total Balance for Customer 2: ' || AccountOperations.GetTotalBalance(2));

END;

/

**Output**

