**Exercise 7: Packages**

**Step 1: Customer Management Package**

CREATE OR REPLACE PACKAGE CustomerManagement AS

PROCEDURE AddNewCustomer (

p\_customer\_id IN NUMBER,

p\_customer\_name IN VARCHAR2,

p\_age IN NUMBER

);

PROCEDURE UpdateCustomerDetails (

p\_customer\_id IN NUMBER,

p\_customer\_name IN VARCHAR2,

p\_age IN NUMBER

);

FUNCTION GetCustomerBalance (

p\_customer\_id IN NUMBER

) RETURN NUMBER;

END CustomerManagement;

/

**Package Body:**

CREATE OR REPLACE PACKAGE BODY CustomerManagement AS

PROCEDURE AddNewCustomer (

p\_customer\_id IN NUMBER,

p\_customer\_name IN VARCHAR2,

p\_age IN NUMBER

) IS

BEGIN

INSERT INTO customers (customer\_id, customer\_name, age)

VALUES (p\_customer\_id, p\_customer\_name, p\_age);

COMMIT;

END AddNewCustomer;

PROCEDURE UpdateCustomerDetails (

p\_customer\_id IN NUMBER,

p\_customer\_name IN VARCHAR2,

p\_age IN NUMBER

) IS

BEGIN

UPDATE customers

SET customer\_name = p\_customer\_name,

age = p\_age

WHERE customer\_id = p\_customer\_id;

COMMIT;

END UpdateCustomerDetails;

FUNCTION GetCustomerBalance (

p\_customer\_id IN NUMBER

) RETURN NUMBER IS

v\_balance NUMBER;

BEGIN

SELECT SUM(balance) INTO v\_balance

FROM accounts

WHERE account\_id IN (SELECT account\_id FROM transactions WHERE customer\_id = p\_customer\_id);

RETURN v\_balance;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN 0;

END GetCustomerBalance;

END CustomerManagement;

/

**Step 2: Employee Management Package**

CREATE OR REPLACE PACKAGE EmployeeManagement AS

PROCEDURE HireNewEmployee (

p\_employee\_id IN NUMBER,

p\_department\_id IN NUMBER,

p\_salary IN NUMBER

);

PROCEDURE UpdateEmployeeDetails (

p\_employee\_id IN NUMBER,

p\_department\_id IN NUMBER,

p\_salary IN NUMBER

);

FUNCTION CalculateAnnualSalary (

p\_employee\_id IN NUMBER

) RETURN NUMBER;

END EmployeeManagement;

/

**Package Body:**

CREATE OR REPLACE PACKAGE BODY EmployeeManagement AS

PROCEDURE HireNewEmployee (

p\_employee\_id IN NUMBER,

p\_department\_id IN NUMBER,

p\_salary IN NUMBER

) IS

BEGIN

INSERT INTO employees (employee\_id, department\_id, salary)

VALUES (p\_employee\_id, p\_department\_id, p\_salary);

COMMIT;

END HireNewEmployee;

PROCEDURE UpdateEmployeeDetails (

p\_employee\_id IN NUMBER,

p\_department\_id IN NUMBER,

p\_salary IN NUMBER

) IS

BEGIN

UPDATE employees

SET department\_id = p\_department\_id,

salary = p\_salary

WHERE employee\_id = p\_employee\_id;

COMMIT;

END UpdateEmployeeDetails;

FUNCTION CalculateAnnualSalary (

p\_employee\_id IN NUMBER

) RETURN NUMBER IS

v\_salary NUMBER;

BEGIN

SELECT salary INTO v\_salary

FROM employees

WHERE employee\_id = p\_employee\_id;

RETURN v\_salary \* 12; -- Assuming monthly salary

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN 0;

END CalculateAnnualSalary;

END EmployeeManagement;

/

**Step 3: Account Operations Package**

CREATE OR REPLACE PACKAGE AccountOperations AS

PROCEDURE OpenNewAccount (

p\_account\_id IN NUMBER,

p\_initial\_balance IN NUMBER

);

PROCEDURE CloseAccount (

p\_account\_id IN NUMBER

);

FUNCTION GetTotalBalance (

p\_customer\_id IN NUMBER

) RETURN NUMBER;

END AccountOperations;

/

**Package Body:**

CREATE OR REPLACE PACKAGE BODY AccountOperations AS

PROCEDURE OpenNewAccount (

p\_account\_id IN NUMBER,

p\_initial\_balance IN NUMBER

) IS

BEGIN

INSERT INTO accounts (account\_id, balance)

VALUES (p\_account\_id, p\_initial\_balance);

COMMIT;

END OpenNewAccount;

PROCEDURE CloseAccount (

p\_account\_id IN NUMBER

) IS

BEGIN

DELETE FROM accounts

WHERE account\_id = p\_account\_id;

COMMIT;

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 account\_id IN (SELECT account\_id FROM transactions WHERE customer\_id = p\_customer\_id);

RETURN v\_total\_balance;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN 0;

END GetTotalBalance;

END AccountOperations;

/

**Testing the Packages**

1. CustomerManagement:

BEGIN

CustomerManagement.AddNewCustomer(1, 'Alice Johnson', 30);

CustomerManagement.UpdateCustomerDetails(1, 'Alice Smith', 31);

DBMS\_OUTPUT.PUT\_LINE('Balance: ' || CustomerManagement.GetCustomerBalance(1));

END;

/

1. EmployeeManagement:

BEGIN

EmployeeManagement.HireNewEmployee(1, 10, 50000);

EmployeeManagement.UpdateEmployeeDetails(1, 20, 55000);

DBMS\_OUTPUT.PUT\_LINE('Annual Salary: ' || EmployeeManagement.CalculateAnnualSalary(1));

END;

/

1. AccountOperations:

BEGIN

AccountOperations.OpenNewAccount(1, 1000);

AccountOperations.CloseAccount(1);

DBMS\_OUTPUT.PUT\_LINE('Total Balance: ' || AccountOperations.GetTotalBalance(1));

END;

/