**Exercise 7: Packages**

**Scenario 1: Customer Management Package**

**Package Specification: CustomerManagement**

CREATE OR REPLACE PACKAGE CustomerManagement AS

PROCEDURE AddNewCustomer (

p\_customerid IN NUMBER,

p\_name IN VARCHAR2,

p\_dob IN DATE,

p\_balance IN NUMBER

);

PROCEDURE UpdateCustomerDetails (

p\_customerid IN NUMBER,

p\_name IN VARCHAR2,

p\_dob IN DATE,

p\_balance IN NUMBER

);

FUNCTION GetCustomerBalance (

p\_customerid IN NUMBER

) RETURN NUMBER;

END CustomerManagement;

/

**Package Body: CustomerManagement**

CREATE OR REPLACE PACKAGE BODY CustomerManagement AS

PROCEDURE AddNewCustomer (

p\_customerid IN NUMBER,

p\_name IN VARCHAR2,

p\_dob IN DATE,

p\_balance IN NUMBER

) IS

BEGIN

INSERT INTO customers (customerid, name, dob, balance, lastmodified)

VALUES (p\_customerid, p\_name, p\_dob, p\_balance, SYSDATE);

COMMIT;

END AddNewCustomer;

PROCEDURE UpdateCustomerDetails (

p\_customerid IN NUMBER,

p\_name IN VARCHAR2,

p\_dob IN DATE,

p\_balance IN NUMBER

) IS

BEGIN

UPDATE customers

SET name = p\_name,

dob = p\_dob,

balance = p\_balance,

lastmodified = SYSDATE

WHERE customerid = p\_customerid;

COMMIT;

END UpdateCustomerDetails;

FUNCTION GetCustomerBalance (

p\_customerid IN NUMBER

) RETURN NUMBER 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

RETURN 0;

END GetCustomerBalance;

END CustomerManagement;

/

**Scenario 2: Employee Management Package**

**Package Specification: EmployeeManagement**

CREATE OR REPLACE PACKAGE EmployeeManagement AS

PROCEDURE HireNewEmployee (

p\_employeeid IN NUMBER,

p\_name IN VARCHAR2,

p\_position IN VARCHAR2,

p\_salary IN NUMBER,

p\_department IN VARCHAR2,

p\_hiredate IN DATE

);

PROCEDURE UpdateEmployeeDetails (

p\_employeeid IN NUMBER,

p\_name IN VARCHAR2,

p\_position IN VARCHAR2,

p\_salary IN NUMBER,

p\_department IN VARCHAR2,

p\_hiredate IN DATE

);

FUNCTION CalculateAnnualSalary (

p\_employeeid IN NUMBER

) RETURN NUMBER;

END EmployeeManagement;

/

**Package Body: EmployeeManagement**

CREATE OR REPLACE PACKAGE BODY EmployeeManagement AS

PROCEDURE HireNewEmployee (

p\_employeeid IN NUMBER,

p\_name IN VARCHAR2,

p\_position IN VARCHAR2,

p\_salary IN NUMBER,

p\_department IN VARCHAR2,

p\_hiredate IN 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);

COMMIT;

END HireNewEmployee;

PROCEDURE UpdateEmployeeDetails (

p\_employeeid IN NUMBER,

p\_name IN VARCHAR2,

p\_position IN VARCHAR2,

p\_salary IN NUMBER,

p\_department IN VARCHAR2,

p\_hiredate IN DATE

) IS

BEGIN

UPDATE employees

SET name = p\_name,

position = p\_position,

salary = p\_salary,

department = p\_department,

hiredate = p\_hiredate

WHERE employeeid = p\_employeeid;

COMMIT;

END UpdateEmployeeDetails;

FUNCTION CalculateAnnualSalary (

p\_employeeid IN NUMBER

) RETURN NUMBER IS

v\_salary employees.salary%TYPE;

BEGIN

SELECT salary INTO v\_salary

FROM employees

WHERE employeeid = p\_employeeid;

RETURN v\_salary \* 12; -- Annual salary calculation

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN 0;

END CalculateAnnualSalary;

END EmployeeManagement;

/

**Scenario 3: Account Operations Package**

**Package Specification: AccountOperations**

CREATE OR REPLACE PACKAGE AccountOperations AS

PROCEDURE OpenNewAccount (

p\_accountid IN NUMBER,

p\_customerid IN NUMBER,

p\_accounttype IN VARCHAR2,

p\_balance IN NUMBER

);

PROCEDURE CloseAccount (

p\_accountid IN NUMBER

);

FUNCTION GetTotalCustomerBalance (

p\_customerid IN NUMBER

) RETURN NUMBER;

END AccountOperations;

/

**Package Body: AccountOperations**

CREATE OR REPLACE PACKAGE BODY AccountOperations AS

PROCEDURE OpenNewAccount (

p\_accountid IN NUMBER,

p\_customerid IN NUMBER,

p\_accounttype IN VARCHAR2,

p\_balance IN NUMBER

) IS

BEGIN

INSERT INTO accounts (accountid, customerid, accounttype, balance, lastmodified)

VALUES (p\_accountid, p\_customerid, p\_accounttype, p\_balance, SYSDATE);

COMMIT;

END OpenNewAccount;

PROCEDURE CloseAccount (

p\_accountid IN NUMBER

) IS

BEGIN

DELETE FROM accounts

WHERE accountid = p\_accountid;

COMMIT;

END CloseAccount;

FUNCTION GetTotalCustomerBalance (

p\_customerid IN NUMBER

) RETURN NUMBER IS

v\_total\_balance NUMBER := 0;

BEGIN

SELECT SUM(balance) INTO v\_total\_balance

FROM accounts

WHERE customerid = p\_customerid;

RETURN v\_total\_balance;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN 0;

END GetTotalCustomerBalance;

END AccountOperations;

/