**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.

**CODE:**

**Package specification**

CREATE OR REPLACE PACKAGE CustomerManagement AS

PROCEDURE AddCustomer (

p\_customer\_id IN NUMBER,

p\_name IN VARCHAR2,

p\_address IN VARCHAR2

);

PROCEDURE UpdateCustomer (

p\_customer\_id IN NUMBER,

p\_name IN VARCHAR2,

p\_address IN VARCHAR2

);

FUNCTION GetCustomerBalance (

p\_customer\_id IN NUMBER

) RETURN NUMBER;

END CustomerManagement;

/

**Package Body**

CREATE OR REPLACE PACKAGE BODY CustomerManagement AS

PROCEDURE AddCustomer (

p\_customer\_id IN NUMBER,

p\_name IN VARCHAR2,

p\_address IN VARCHAR2

) IS

BEGIN

INSERT INTO CUSTOMERS (CUSTOMER\_ID, NAME, ADDRESS)

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

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Customer ID already exists.');

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Error adding customer: ' || SQLERRM);

END AddCustomer;

PROCEDURE UpdateCustomer (

p\_customer\_id IN NUMBER,

p\_name IN VARCHAR2,

p\_address IN VARCHAR2

) IS

BEGIN

UPDATE CUSTOMERS

SET NAME = p\_name,

ADDRESS = p\_address

WHERE CUSTOMER\_ID = p\_customer\_id;

IF SQL%ROWCOUNT = 0 THEN

RAISE\_APPLICATION\_ERROR(-20003, 'Customer ID not found.');

END IF;

EXCEPTION

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20004, 'Error updating customer: ' || SQLERRM);

END UpdateCustomer;

FUNCTION GetCustomerBalance (

p\_customer\_id IN NUMBER

) RETURN NUMBER IS

v\_balance NUMBER;

BEGIN

SELECT SUM(BALANCE) INTO v\_balance

FROM ACCOUNTS

WHERE CUSTOMER\_ID = p\_customer\_id;

RETURN v\_balance;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN 0;

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20005, 'Error retrieving customer balance: ' || SQLERRM);

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.

**CODE:**

**Package Specification**

CREATE OR REPLACE PACKAGE EmployeeManagement AS

PROCEDURE HireEmployee (

p\_employee\_id IN NUMBER,

p\_name IN VARCHAR2,

p\_salary IN NUMBER,

p\_department IN VARCHAR2

);

PROCEDURE UpdateEmployee (

p\_employee\_id IN NUMBER,

p\_name IN VARCHAR2,

p\_salary IN NUMBER,

p\_department IN VARCHAR2

);

FUNCTION CalculateAnnualSalary (

p\_employee\_id IN NUMBER

) RETURN NUMBER;

END EmployeeManagement;

/

**Package Body**

CREATE OR REPLACE PACKAGE BODY EmployeeManagement AS

PROCEDURE HireEmployee (

p\_employee\_id IN NUMBER,

p\_name IN VARCHAR2,

p\_salary IN NUMBER,

p\_department IN VARCHAR2

) IS

BEGIN

INSERT INTO EMPLOYEES (EMPLOYEE\_ID, NAME, SALARY, DEPARTMENT)

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

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Employee ID already exists.');

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Error hiring employee: ' || SQLERRM);

END HireEmployee;

PROCEDURE UpdateEmployee (

p\_employee\_id IN NUMBER,

p\_name IN VARCHAR2,

p\_salary IN NUMBER,

p\_department IN VARCHAR2

) IS

BEGIN

UPDATE EMPLOYEES

SET NAME = p\_name,

SALARY = p\_salary,

DEPARTMENT = p\_department

WHERE EMPLOYEE\_ID = p\_employee\_id;

IF SQL%ROWCOUNT = 0 THEN

RAISE\_APPLICATION\_ERROR(-20003, 'Employee ID not found.');

END IF;

EXCEPTION

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20004, 'Error 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 EMPLOYEE\_ID = p\_employee\_id;

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

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN 0;

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20005, 'Error calculating annual salary: ' || SQLERRM);

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.

**CODE:**

**Package specification**

CREATE OR REPLACE PACKAGE AccountOperations AS

PROCEDURE OpenAccount (

p\_account\_id IN NUMBER,

p\_customer\_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 OpenAccount (

p\_account\_id IN NUMBER,

p\_customer\_id IN NUMBER,

p\_initial\_balance IN NUMBER

) IS

BEGIN

INSERT INTO ACCOUNTS (ACCOUNT\_ID, CUSTOMER\_ID, BALANCE)

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

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Account ID already exists.');

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Error opening account: ' || SQLERRM);

END OpenAccount;

PROCEDURE CloseAccount (

p\_account\_id IN NUMBER

) IS

BEGIN

DELETE FROM ACCOUNTS

WHERE ACCOUNT\_ID = p\_account\_id;

IF SQL%ROWCOUNT = 0 THEN

RAISE\_APPLICATION\_ERROR(-20003, 'Account ID not found.');

END IF;

EXCEPTION

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20004, '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 CUSTOMER\_ID = p\_customer\_id;

RETURN v\_total\_balance;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN 0;

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20005, 'Error retrieving total balance: ' || SQLERRM);

END GetTotalBalance;

END AccountOperations;

/