**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.

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.

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.

Here are the PL/SQL packages for the given scenarios:

**Scenario 1: Customer Management Package**

**Package Specification**

CREATE OR REPLACE PACKAGE CustomerManagement IS

PROCEDURE AddNewCustomer (

p\_customer\_id IN NUMBER,

p\_name IN VARCHAR2,

p\_age IN NUMBER,

p\_address IN VARCHAR2

);

PROCEDURE UpdateCustomerDetails (

p\_customer\_id IN NUMBER,

p\_name IN VARCHAR2,

p\_age IN NUMBER,

p\_address IN VARCHAR2

);

FUNCTION GetCustomerBalance (

p\_customer\_id IN NUMBER

) RETURN NUMBER;

END CustomerManagement;

/

**Package Body**

CREATE OR REPLACE PACKAGE BODY CustomerManagement IS

PROCEDURE AddNewCustomer (

p\_customer\_id IN NUMBER,

p\_name IN VARCHAR2,

p\_age IN NUMBER,

p\_address IN VARCHAR2

) IS

BEGIN

INSERT INTO customers (customer\_id, name, age, address)

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

COMMIT;

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Customer ID ' || p\_customer\_id || ' already exists.');

WHEN OTHERS THEN

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

END AddNewCustomer;

PROCEDURE UpdateCustomerDetails (

p\_customer\_id IN NUMBER,

p\_name IN VARCHAR2,

p\_age IN NUMBER,

p\_address IN VARCHAR2

) IS

BEGIN

UPDATE customers

SET name = p\_name, age = p\_age, address = p\_address

WHERE customer\_id = p\_customer\_id;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

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

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 customer\_id = p\_customer\_id;

RETURN v\_balance;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN 0;

WHEN OTHERS THEN

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

RETURN NULL;

END GetCustomerBalance;

END CustomerManagement;

/

**Scenario 2: Employee Management Package**

**Package Specification**

CREATE OR REPLACE PACKAGE EmployeeManagement IS

PROCEDURE HireNewEmployee (

p\_employee\_id IN NUMBER,

p\_name IN VARCHAR2,

p\_department\_id IN NUMBER,

p\_salary IN NUMBER

);

PROCEDURE UpdateEmployeeDetails (

p\_employee\_id IN NUMBER,

p\_name IN VARCHAR2,

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 IS

PROCEDURE HireNewEmployee (

p\_employee\_id IN NUMBER,

p\_name IN VARCHAR2,

p\_department\_id IN NUMBER,

p\_salary IN NUMBER

) IS

BEGIN

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

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

COMMIT;

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Employee ID ' || p\_employee\_id || ' already exists.');

WHEN OTHERS THEN

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

END HireNewEmployee;

PROCEDURE UpdateEmployeeDetails (

p\_employee\_id IN NUMBER,

p\_name IN VARCHAR2,

p\_department\_id IN NUMBER,

p\_salary IN NUMBER

) IS

BEGIN

UPDATE employees

SET name = p\_name, department\_id = p\_department\_id, salary = p\_salary

WHERE employee\_id = p\_employee\_id;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

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

END UpdateEmployeeDetails;

FUNCTION CalculateAnnualSalary (

p\_employee\_id IN NUMBER

) RETURN NUMBER IS

v\_monthly\_salary NUMBER;

v\_annual\_salary NUMBER;

BEGIN

SELECT salary INTO v\_monthly\_salary

FROM employees

WHERE employee\_id = p\_employee\_id;

v\_annual\_salary := v\_monthly\_salary \* 12;

RETURN v\_annual\_salary;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN 0;

WHEN OTHERS THEN

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

RETURN NULL;

END CalculateAnnualSalary;

END EmployeeManagement;

/

**Scenario 3: Account Operations Package**

**Package Specification**

CREATE OR REPLACE PACKAGE AccountOperations IS

PROCEDURE OpenNewAccount (

p\_account\_id IN NUMBER,

p\_customer\_id IN NUMBER,

p\_account\_type IN VARCHAR2,

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 IS

PROCEDURE OpenNewAccount (

p\_account\_id IN NUMBER,

p\_customer\_id IN NUMBER,

p\_account\_type IN VARCHAR2,

p\_initial\_balance IN NUMBER

) IS

BEGIN

INSERT INTO accounts (account\_id, customer\_id, account\_type, balance)

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

COMMIT;

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Account ID ' || p\_account\_id || ' already exists.');

WHEN OTHERS THEN

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

END OpenNewAccount;

PROCEDURE CloseAccount (

p\_account\_id IN NUMBER

) IS

BEGIN

DELETE FROM accounts

WHERE account\_id = p\_account\_id;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error: ' || 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

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

RETURN NULL;

END GetTotalBalance;

END AccountOperations;

/

These packages group related procedures and functions together, making it easier to manage and use them in your PL/SQL programs. Each package handles specific operations related to customers, employees, and accounts, as described in the scenarios.