**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.

**QUERY:**

CREATE OR REPLACE PACKAGE CustomerManagement IS

-- Procedure to add a new customer

PROCEDURE AddCustomer(customer\_id IN NUMBER, customer\_name IN VARCHAR2, customer\_email IN VARCHAR2);

-- Procedure to update customer details

PROCEDURE UpdateCustomerDetails(customer\_id IN NUMBER, new\_name IN VARCHAR2, new\_email IN VARCHAR2);

-- Function to get customer balance

FUNCTION GetCustomerBalance(customer\_id IN NUMBER) RETURN NUMBER;

END CustomerManagement;

/

CREATE OR REPLACE PACKAGE BODY CustomerManagement IS

-- Procedure to add a new customer

PROCEDURE AddCustomer(customer\_id IN NUMBER, customer\_name IN VARCHAR2, customer\_email IN VARCHAR2) IS

BEGIN

-- Add logic to insert a new customer into the database

-- Example: INSERT INTO Customers VALUES (customer\_id, customer\_name, customer\_email);

DBMS\_OUTPUT.PUT\_LINE('New customer added successfully.');

END AddCustomer;

-- Procedure to update customer details

PROCEDURE UpdateCustomerDetails(customer\_id IN NUMBER, new\_name IN VARCHAR2, new\_email IN VARCHAR2) IS

BEGIN

-- Add logic to update customer details in the database

-- Example: UPDATE Customers SET customer\_name = new\_name, customer\_email = new\_email WHERE customer\_id = customer\_id;

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

END UpdateCustomerDetails;

-- Function to get customer balance

FUNCTION GetCustomerBalance(customer\_id IN NUMBER) RETURN NUMBER IS

v\_balance NUMBER;

BEGIN

-- Add logic to retrieve customer balance from the database

-- Example: SELECT balance INTO v\_balance FROM CustomerBalances WHERE customer\_id = customer\_id;

v\_balance := 1000; -- Example balance value

RETURN v\_balance;

END GetCustomerBalance;

END CustomerManagement;

**Explanation:**

The `CustomerManagement` package declares procedures for adding a new customer and updating customer details, along with a function to get the customer balance. The package body defines the implementation of each procedure and function. You can customize the SQL statements within each procedure to interact with your database tables accordingly.

**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.

**QUERY:**

CREATE OR REPLACE PACKAGE EmployeeManagement IS

-- Procedure to hire a new employee

PROCEDURE HireEmployee(employee\_id IN NUMBER, employee\_name IN VARCHAR2, salary IN NUMBER);

-- Procedure to update employee details

PROCEDURE UpdateEmployeeDetails(employee\_id IN NUMBER, new\_name IN VARCHAR2, new\_salary IN NUMBER);

-- Function to calculate annual salary

FUNCTION CalculateAnnualSalary(employee\_id IN NUMBER) RETURN NUMBER;

END EmployeeManagement;

/

CREATE OR REPLACE PACKAGE BODY EmployeeManagement IS

-- Procedure to hire a new employee

PROCEDURE HireEmployee(employee\_id IN NUMBER, employee\_name IN VARCHAR2, salary IN NUMBER) IS

BEGIN

-- Add logic to insert a new employee into the database

-- Example: INSERT INTO Employees VALUES (employee\_id, employee\_name, salary);

DBMS\_OUTPUT.PUT\_LINE('New employee hired successfully.');

END HireEmployee;

-- Procedure to update employee details

PROCEDURE UpdateEmployeeDetails(employee\_id IN NUMBER, new\_name IN VARCHAR2, new\_salary IN NUMBER) IS

BEGIN

-- Add logic to update employee details in the database

-- Example: UPDATE Employees SET employee\_name = new\_name, salary = new\_salary WHERE employee\_id = employee\_id;

DBMS\_OUTPUT.PUT\_LINE('Employee details updated successfully.');

END UpdateEmployeeDetails;

-- Function to calculate annual salary

FUNCTION CalculateAnnualSalary(employee\_id IN NUMBER) RETURN NUMBER IS

v\_monthly\_salary NUMBER;

v\_annual\_salary NUMBER;

BEGIN

-- Add logic to retrieve employee's monthly salary from the database

-- Example: SELECT salary INTO v\_monthly\_salary FROM Employees WHERE employee\_id = employee\_id;

v\_monthly\_salary := 5000; -- Example monthly salary value

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

RETURN v\_annual\_salary;

END CalculateAnnualSalary;

END EmployeeManagement;

**Explanation:**

The `EmployeeManagement` package is defined with procedures for hiring new employees and updating employee details, along with a function to calculate the annual salary. The package body contains the implementation details for each procedure and function, including sample logic to interact with the database tables.

**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.

**QUERY:**

CREATE OR REPLACE PACKAGE AccountOperations IS

-- Procedure to open a new account

PROCEDURE OpenAccount(customer\_id IN NUMBER, account\_number IN VARCHAR2, initial\_balance IN NUMBER);

-- Procedure to close an account

PROCEDURE CloseAccount(account\_number IN VARCHAR2);

-- Function to get the total balance of a customer across all accounts

FUNCTION GetTotalBalance(customer\_id IN NUMBER) RETURN NUMBER;

END AccountOperations;

/

CREATE OR REPLACE PACKAGE BODY AccountOperations IS

-- Procedure to open a new account

PROCEDURE OpenAccount(customer\_id IN NUMBER, account\_number IN VARCHAR2, initial\_balance IN NUMBER) IS

BEGIN

-- Add logic to insert a new account into the database

-- Example: INSERT INTO Accounts VALUES (customer\_id, account\_number, initial\_balance);

DBMS\_OUTPUT.PUT\_LINE('New account opened successfully.');

END OpenAccount;

-- Procedure to close an account

PROCEDURE CloseAccount(account\_number IN VARCHAR2) IS

BEGIN

-- Add logic to close an account in the database

-- Example: DELETE FROM Accounts WHERE account\_number = account\_number;

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

END CloseAccount;

-- Function to get the total balance of a customer across all accounts

FUNCTION GetTotalBalance(customer\_id IN NUMBER) RETURN NUMBER IS

v\_total\_balance NUMBER := 0;

BEGIN

-- Add logic to calculate the total balance of a customer across all accounts

-- Example: SELECT SUM(balance) INTO v\_total\_balance FROM Accounts WHERE customer\_id = customer\_id;

v\_total\_balance := 10000; -- Example total balance value

RETURN v\_total\_balance;

END GetTotalBalance;

END AccountOperations;

**Explanation:**

The `AccountOperations` package is defined with procedures for opening a new account and closing an account, along with a function to get the total balance of a customer across all accounts. The package body contains the implementation details for each procedure and function, including sample logic to interact with the database tables.