**EXERCISE 7: PACKAGES**

**Scenario 1: Group All Customer-Related Procedures and Functions into a Package**

CREATE OR REPLACE PACKAGE CustomerManagement AS

PROCEDURE AddNewCustomer(

p\_CustomerID NUMBER,

p\_Name VARCHAR2,

p\_DOB DATE,

p\_Balance NUMBER

);

PROCEDURE UpdateCustomerDetails(

p\_CustomerID NUMBER,

p\_Name VARCHAR2,

p\_DOB DATE,

p\_Balance NUMBER

);

FUNCTION GetCustomerBalance(

p\_CustomerID NUMBER

) RETURN NUMBER;

END CustomerManagement;

/

CREATE OR REPLACE PACKAGE BODY CustomerManagement AS

PROCEDURE AddNewCustomer(

p\_CustomerID NUMBER,

p\_Name VARCHAR2,

p\_DOB DATE,

p\_Balance NUMBER

) IS

BEGIN

-- Insert a new customer into the Customers table

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)

VALUES (p\_CustomerID, p\_Name, p\_DOB, p\_Balance, SYSDATE);

DBMS\_OUTPUT.PUT\_LINE('Customer added: ' || p\_Name);

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

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

END AddNewCustomer;

PROCEDURE UpdateCustomerDetails(

p\_CustomerID NUMBER,

p\_Name VARCHAR2,

p\_DOB DATE,

p\_Balance NUMBER

) IS

BEGIN

-- Update customer details

UPDATE Customers

SET Name = p\_Name, DOB = p\_DOB, Balance = p\_Balance, LastModified = SYSDATE

WHERE CustomerID = p\_CustomerID;

IF SQL%ROWCOUNT = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Customer ID ' || p\_CustomerID || ' not found.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Customer updated: ' || p\_Name);

END IF;

END UpdateCustomerDetails;

FUNCTION GetCustomerBalance(

p\_CustomerID NUMBER

) RETURN NUMBER IS

v\_Balance NUMBER;

BEGIN

-- Retrieve the customer's balance

SELECT Balance INTO v\_Balance

FROM Customers

WHERE CustomerID = p\_CustomerID;

RETURN v\_Balance;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Customer ID ' || p\_CustomerID || ' not found.');

RETURN NULL;

END GetCustomerBalance;

END CustomerManagement;

/

**Scenario 2: Create a Package to Manage Employee Data**

CREATE OR REPLACE PACKAGE EmployeeManagement AS

PROCEDURE HireEmployee(

p\_EmployeeID NUMBER,

p\_Name VARCHAR2,

p\_Position VARCHAR2,

p\_Salary NUMBER,

p\_Department VARCHAR2,

p\_HireDate DATE

);

PROCEDURE UpdateEmployeeDetails(

p\_EmployeeID NUMBER,

p\_Name VARCHAR2,

p\_Position VARCHAR2,

p\_Salary NUMBER,

p\_Department VARCHAR2

);

FUNCTION CalculateAnnualSalary(

p\_EmployeeID NUMBER

) RETURN NUMBER;

END EmployeeManagement;

/

CREATE OR REPLACE PACKAGE BODY EmployeeManagement AS

PROCEDURE HireEmployee(

p\_EmployeeID NUMBER,

p\_Name VARCHAR2,

p\_Position VARCHAR2,

p\_Salary NUMBER,

p\_Department VARCHAR2,

p\_HireDate DATE

) IS

BEGIN

-- Insert a new employee into the Employees table

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)

VALUES (p\_EmployeeID, p\_Name, p\_Position, p\_Salary, p\_Department, p\_HireDate);

DBMS\_OUTPUT.PUT\_LINE('Employee hired: ' || p\_Name);

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

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

END HireEmployee;

PROCEDURE UpdateEmployeeDetails(

p\_EmployeeID NUMBER,

p\_Name VARCHAR2,

p\_Position VARCHAR2,

p\_Salary NUMBER,

p\_Department VARCHAR2

) IS

BEGIN

-- Update employee details

UPDATE Employees

SET Name = p\_Name, Position = p\_Position, Salary = p\_Salary, Department = p\_Department

WHERE EmployeeID = p\_EmployeeID;

IF SQL%ROWCOUNT = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Employee ID ' || p\_EmployeeID || ' not found.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Employee updated: ' || p\_Name);

END IF;

END UpdateEmployeeDetails;

FUNCTION CalculateAnnualSalary(

p\_EmployeeID NUMBER

) RETURN NUMBER IS

v\_Salary NUMBER;

BEGIN

-- Calculate the annual salary

SELECT Salary INTO v\_Salary

FROM Employees

WHERE EmployeeID = p\_EmployeeID;

RETURN v\_Salary \* 12;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Employee ID ' || p\_EmployeeID || ' not found.');

RETURN NULL;

END CalculateAnnualSalary;

END EmployeeManagement;

/

**Scenario 3: Group All Account-Related Operations into a Package**

CREATE OR REPLACE PACKAGE AccountOperations AS

PROCEDURE OpenAccount(

p\_AccountID NUMBER,

p\_CustomerID NUMBER,

p\_AccountType VARCHAR2,

p\_Balance NUMBER

);

PROCEDURE CloseAccount(

p\_AccountID NUMBER

);

FUNCTION GetTotalBalance(

p\_CustomerID NUMBER

) RETURN NUMBER;

END AccountOperations;

/

CREATE OR REPLACE PACKAGE BODY AccountOperations AS

PROCEDURE OpenAccount(

p\_AccountID NUMBER,

p\_CustomerID NUMBER,

p\_AccountType VARCHAR2,

p\_Balance NUMBER

) IS

BEGIN

-- Insert a new account into the Accounts table

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)

VALUES (p\_AccountID, p\_CustomerID, p\_AccountType, p\_Balance, SYSDATE);

DBMS\_OUTPUT.PUT\_LINE('Account opened: Account ID ' || p\_AccountID || ', Customer ID ' || p\_CustomerID);

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

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

END OpenAccount;

PROCEDURE CloseAccount(

p\_AccountID NUMBER

) IS

BEGIN

-- Delete the account from the Accounts table

DELETE FROM Accounts

WHERE AccountID = p\_AccountID;

IF SQL%ROWCOUNT = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Account ID ' || p\_AccountID || ' not found.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Account closed: Account ID ' || p\_AccountID);

END IF;

END CloseAccount;

FUNCTION GetTotalBalance(

p\_CustomerID NUMBER

) RETURN NUMBER IS

v\_TotalBalance NUMBER;

BEGIN

-- Calculate the total balance across all accounts for the customer

SELECT SUM(Balance) INTO v\_TotalBalance

FROM Accounts

WHERE CustomerID