**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.

**Ans:**

**Creating Package:**

CREATE OR REPLACE PACKAGE CustomerManagement AS

PROCEDURE AddNewCustomer (

custID IN Customers.CustomerID%TYPE,

custname IN Customers.Name%TYPE,

dob IN Customers.DOB%TYPE,

bal IN Customers.Balance%TYPE

);

PROCEDURE UpdateCustomerDetails (

custID IN Customers.CustomerID%TYPE,

custname IN Customers.Name%TYPE,

dob IN Customers.DOB%TYPE,

bal IN Customers.Balance%TYPE

);

FUNCTION GetCustomerBalance (

custID IN Customers.CustomerID%TYPE

) RETURN NUMBER;

END CustomerManagement;

/

**Creating Package Body:**

CREATE OR REPLACE PACKAGE BODY CustomerManagement AS

PROCEDURE AddNewCustomer (

custID IN Customers.CustomerID%TYPE,

custname IN Customers.Name%TYPE,

dob IN Customers.DOB%TYPE,

bal IN Customers.Balance%TYPE

) IS

BEGIN

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

VALUES (custID, custname, dob, bal, SYSDATE);

COMMIT;

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Customer with ID ' || custID || ' already exists in database.');

END AddNewCustomer;

PROCEDURE UpdateCustomerDetails (

custID IN Customers.CustomerID%TYPE,

custname IN Customers.Name%TYPE,

dob IN Customers.DOB%TYPE,

bal IN Customers.Balance%TYPE

) IS

BEGIN

UPDATE Customers

SET Name = custname,

DOB = dob,

Balance = bal,

LastModified = SYSDATE

WHERE CustomerID = custID;

COMMIT;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Customer with ID ' || custID || ' not found in database.');

END UpdateCustomerDetails;

FUNCTION GetCustomerBalance (

custID IN Customers.CustomerID%TYPE

) RETURN NUMBER IS

v\_balance NUMBER;

BEGIN

SELECT Balance INTO v\_balance

FROM Customers

WHERE CustomerID = custID;

RETURN v\_balance;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Customer with ID ' || custID || ' not found in database.');

RETURN NULL;

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.

**Ans:**

**Creating Package:**

CREATE OR REPLACE PACKAGE EmployeeManagement AS

PROCEDURE hiringEmployee (

empID IN Employees.EmployeeID%TYPE,

empName IN Employees.Name%TYPE,

ampPosition IN Employees.Position%TYPE,

empSal IN Employees.Salary%TYPE,

empDept IN Employees.Department%TYPE,

empHiringDate IN Employees.HireDate%TYPE

);

PROCEDURE UpdateEmployeeDetails (

empID IN Employees.EmployeeID%TYPE,

empName IN Employees.Name%TYPE,

ampPosition IN Employees.Position%TYPE,

empSal IN Employees.Salary%TYPE,

empDept IN Employees.Department%TYPE,

empHiringDate IN Employees.HireDate%TYPE

);

FUNCTION CalculateAnnualSal (

empID IN Employees.EmployeeID%TYPE

) RETURN NUMBER;

END EmployeeManagement;

/

**Creating Package Body:**

CREATE OR REPLACE PACKAGE BODY EmployeeManagement AS

PROCEDURE hiringEmployee (

empID IN Employees.EmployeeID%TYPE,

empName IN Employees.Name%TYPE,

ampPosition IN Employees.Position%TYPE,

empSal IN Employees.Salary%TYPE,

empDept IN Employees.Department%TYPE,

empHiringDate IN Employees.HireDate%TYPE

) IS

BEGIN

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

VALUES (empID, empName, ampPosition, empSal, empDept, empHiringDate);

COMMIT;

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Employee with ID ' || empID || ' already exists in database.');

END hiringEmployee;

PROCEDURE UpdateEmployeeDetails (

empID IN Employees.EmployeeID%TYPE,

empName IN Employees.Name%TYPE,

ampPosition IN Employees.Position%TYPE,

empSal IN Employees.Salary%TYPE,

empDept IN Employees.Department%TYPE,

empHiringDate IN Employees.HireDate%TYPE

) IS

BEGIN

UPDATE Employees

SET Name = empName,

Position = ampPosition,

Salary = empSal,

Department = empDept,

HireDate = empHiringDate

WHERE EmployeeID = empID;

COMMIT;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Employee with ID ' || empID || ' not found in database.');

END UpdateEmployeeDetails;

FUNCTION CalculateAnnualSal (

empID IN Employees.EmployeeID%TYPE

) RETURN NUMBER IS

v\_salary Employees.Salary%TYPE;

BEGIN

SELECT Salary INTO v\_salary

FROM Employees

WHERE EmployeeID = empID;

RETURN v\_salary \* 12;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Employee with ID ' || empID || ' not found in database.');

RETURN NULL;

END CalculateAnnualSal;

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.

**Ans:**

**Creating Package:**

CREATE OR REPLACE PACKAGE AccountOperations AS

PROCEDURE OpenNewAccount (

accID IN Accounts.AccountID%TYPE,

custID IN Accounts.CustomerID%TYPE,

accType IN Accounts.AccountType%TYPE,

bal IN Accounts.Balance%TYPE

);

PROCEDURE CloseAccount (

accID IN Accounts.AccountID%TYPE

);

FUNCTION SeeTotalBalance (

custID IN Accounts.CustomerID%TYPE

) RETURN NUMBER;

END AccountOperations;

/

**Creating Package Body:**

CREATE OR REPLACE PACKAGE BODY AccountOperations AS

PROCEDURE OpenNewAccount (

accID IN Accounts.AccountID%TYPE,

custID IN Accounts.CustomerID%TYPE,

accType IN Accounts.AccountType%TYPE,

bal IN Accounts.Balance%TYPE

) IS

BEGIN

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

VALUES (accID, custID, accType, bal, SYSDATE);

COMMIT;

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Account with ID ' || accID || ' already exists in database.');

END OpenNewAccount;

PROCEDURE CloseAccount (

accID IN Accounts.AccountID%TYPE

) IS

BEGIN

DELETE FROM Accounts

WHERE AccountID = accID;

COMMIT;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Account with ID ' || accID || ' not found in database.');

END CloseAccount;

FUNCTION SeeTotalBalance (

custID IN Accounts.CustomerID%TYPE

) RETURN NUMBER IS

v\_totalBalance NUMBER := 0;

BEGIN

SELECT SUM(Balance) INTO v\_totalBalance

FROM Accounts

WHERE CustomerID = custID;

RETURN v\_totalBalance;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Error: No accounts found with Customer ID ' || custID || ' in database.');

RETURN 0;

END SeeTotalBalance;

END AccountOperations;

/