**EXERCISE 7 – PACKAGES**

**SCENARIO 1 - GROUP ALL CUSTOMER-RELATED PROCEDURES AND FUNCTIONS INTO A PACKAGE.**

**Create a package CustomerManagement with procedures for adding a new customer, updating customer details, and a function to get customer balance**

**Package Specification**

CREATE OR REPLACE PACKAGE CustomerManagement AS

-- add a new customer

PROCEDURE AddNewCustomer(

p\_CustomerID IN NUMBER,

p\_Name IN VARCHAR2,

p\_DOB IN DATE,

p\_Balance IN NUMBER

);

-- update customer details

PROCEDURE UpdateCustomerDetails(

p\_CustomerID IN NUMBER,

p\_Name IN VARCHAR2,

p\_DOB IN DATE,

p\_Balance IN NUMBER

);

-- get the balance of a customer

FUNCTION GetCustomerBalance(

p\_CustomerID IN NUMBER

) RETURN NUMBER;

END CustomerManagement;

/

**Package Body**

CREATE OR REPLACE PACKAGE BODY CustomerManagement AS

PROCEDURE AddNewCustomer(

p\_CustomerID IN NUMBER,

p\_Name IN VARCHAR2,

p\_DOB IN DATE,

p\_Balance IN NUMBER

) IS

BEGIN

BEGIN

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

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

COMMIT;

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

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

WHEN OTHERS THEN

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

ROLLBACK;

END;

END AddNewCustomer;

PROCEDURE UpdateCustomerDetails(

p\_CustomerID IN NUMBER,

p\_Name IN VARCHAR2,

p\_DOB IN DATE,

p\_Balance IN NUMBER

) IS

BEGIN

BEGIN

UPDATE Customers

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

WHERE CustomerID = p\_CustomerID;

IF SQL%ROWCOUNT = 0 THEN

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

ELSE

COMMIT;

END IF;

EXCEPTION

WHEN OTHERS THEN

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

ROLLBACK;

END;

END UpdateCustomerDetails;

FUNCTION GetCustomerBalance(

p\_CustomerID IN NUMBER

) RETURN NUMBER IS

v\_Balance NUMBER;

BEGIN

BEGIN

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 with ID ' || p\_CustomerID || ' does not exist.');

RETURN NULL;

WHEN OTHERS THEN

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

RETURN NULL;

END;

END GetCustomerBalance;

END CustomerManagement;

/

**Using Package**

BEGIN

-- Add a new customer

CustomerManagement.AddNewCustomer(8,'Emily', TO\_DATE('2001-11-22', 'YYYY-MM-DD'),12000 );

-- Update customer details

CustomerManagement.UpdateCustomerDetails( 6,'Emily',TO\_DATE('2003-05-11', 'YYYY-MM-DD'),14000 );

-- Get customer balance

DECLARE

v\_Balance NUMBER;

BEGIN

v\_Balance := CustomerManagement.GetCustomerBalance(6);

DBMS\_OUTPUT.PUT\_LINE('Customer balance after Updation: ' || v\_Balance);

END;

END;

/

**SCENARIO 2 - CREATE A PACKAGE TO MANAGE EMPLOYEE DATA.**

**Write a package EmployeeManagement with procedures to hire new employees, update employee details, and a function to calculate annual salary.**

**Package Specification**

CREATE OR REPLACE PACKAGE EmployeeManagement AS

-- Procedure to hire a new employee

PROCEDURE HireEmployee(

p\_EmployeeID IN NUMBER,

p\_Name IN VARCHAR2,

p\_Position IN VARCHAR2,

p\_Salary IN NUMBER,

p\_Department IN VARCHAR2,

p\_HireDate IN DATE

);

-- Procedure to update employee details

PROCEDURE UpdateEmployeeDetails(

p\_EmployeeID IN NUMBER,

p\_Name IN VARCHAR2,

p\_Position IN VARCHAR2,

p\_Salary IN NUMBER,

p\_Department IN VARCHAR2

);

-- Function to calculate annual salary

FUNCTION CalculateAnnualSalary(

p\_EmployeeID IN NUMBER

) RETURN NUMBER;

END EmployeeManagement;

/

**Package Body**

CREATE OR REPLACE PACKAGE BODY EmployeeManagement AS

PROCEDURE HireEmployee(

p\_EmployeeID IN NUMBER,

p\_Name IN VARCHAR2,

p\_Position IN VARCHAR2,

p\_Salary IN NUMBER,

p\_Department IN VARCHAR2,

p\_HireDate IN DATE

) IS

BEGIN

BEGIN

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

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

COMMIT;

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

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

WHEN OTHERS THEN

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

ROLLBACK;

END;

END HireEmployee;

-- Procedure to update employee details

PROCEDURE UpdateEmployeeDetails(

p\_EmployeeID IN NUMBER,

p\_Name IN VARCHAR2,

p\_Position IN VARCHAR2,

p\_Salary IN NUMBER,

p\_Department IN VARCHAR2

) IS

BEGIN

BEGIN

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 with ID ' || p\_EmployeeID || ' does not exist.');

ELSE

COMMIT;

END IF;

EXCEPTION

WHEN OTHERS THEN

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

ROLLBACK;

END;

END UpdateEmployeeDetails;

FUNCTION CalculateAnnualSalary(

p\_EmployeeID IN NUMBER

) RETURN NUMBER IS

v\_AnnualSalary NUMBER;

BEGIN

BEGIN

SELECT Salary \* 12

INTO v\_AnnualSalary

FROM Employees

WHERE EmployeeID = p\_EmployeeID;

RETURN v\_AnnualSalary;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

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

RETURN NULL;

WHEN OTHERS THEN

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

RETURN NULL;

END;

END CalculateAnnualSalary;

END EmployeeManagement;

/

**Using Package**

BEGIN

-- Hire a new employee

EmployeeManagement.HireEmployee( 6,'Fresco','Analyst', 50000,'Finance',SYSDATE );

-- Update employee details

EmployeeManagement.UpdateEmployeeDetails(2,'Bob Brown','Senior Developer', 70000,'IT' );

-- Calculate annual salary

DECLARE

v\_AnnualSalary NUMBER;

emp\_id NUMBER;

BEGIN

emp\_id := 3;

v\_AnnualSalary := EmployeeManagement.CalculateAnnualSalary(emp\_id);

DBMS\_OUTPUT.PUT\_LINE('Annual Salary of Employee with ID '|| emp\_id || ' is :' || v\_AnnualSalary);

END;

END;

/

**SCENARIO 3 - GROUP ALL ACCOUNT-RELATED OPERATIONS INTO A PACKAGE.**

**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**

**Package Specification**

CREATE OR REPLACE PACKAGE AccountOperations AS

-- Procedure to open a new account

PROCEDURE OpenAccount(

p\_AccountID IN NUMBER,

p\_CustomerID IN NUMBER,

p\_AccountType IN VARCHAR2,

p\_Balance IN NUMBER

);

-- Procedure to close an account

PROCEDURE CloseAccount(

p\_AccountID IN NUMBER

);

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

FUNCTION GetTotalBalance(

p\_CustomerID IN NUMBER

) RETURN NUMBER;

END AccountOperations;

/

**Package Body**

CREATE OR REPLACE PACKAGE BODY AccountOperations AS

-- Procedure to open a new account

PROCEDURE OpenAccount(

p\_AccountID IN NUMBER,

p\_CustomerID IN NUMBER,

p\_AccountType IN VARCHAR2,

p\_Balance IN NUMBER

) IS

BEGIN

BEGIN

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

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

COMMIT;

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

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

WHEN OTHERS THEN

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

ROLLBACK;

END;

END OpenAccount;

-- Procedure to close an account

PROCEDURE CloseAccount(

p\_AccountID IN NUMBER

) IS

BEGIN

BEGIN

-- Delete related transactions first

DELETE FROM Transactions

WHERE AccountID = p\_AccountID;

-- Delete the account

DELETE FROM Accounts

WHERE AccountID = p\_AccountID;

IF SQL%ROWCOUNT = 0 THEN

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

ELSE

COMMIT;

END IF;

EXCEPTION

WHEN OTHERS THEN

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

ROLLBACK;

END;

END CloseAccount;

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

FUNCTION GetTotalBalance(

p\_CustomerID IN NUMBER

) RETURN NUMBER IS

v\_TotalBalance NUMBER;

BEGIN

BEGIN

SELECT NVL(SUM(Balance), 0)

INTO v\_TotalBalance

FROM Accounts

WHERE CustomerID = p\_CustomerID;

RETURN v\_TotalBalance;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Error: No accounts found for Customer ID ' || p\_CustomerID);

RETURN NULL;

WHEN OTHERS THEN

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

RETURN NULL;

END;

END GetTotalBalance;

END AccountOperations;

/

**Using Package**

BEGIN

-- Open a new account

AccountOperations.OpenAccount(6, 5, 'Checking', 5000);

-- Close an account

AccountOperations.CloseAccount( 2 );

DECLARE

v\_TotalBalance NUMBER;

acc\_id NUMBER;

BEGIN

acc\_id := 6;

v\_TotalBalance := AccountOperations.GetTotalBalance( acc\_id );

DBMS\_OUTPUT.PUT\_LINE('Total Balance for Account id '|| acc\_id || v\_TotalBalance);

END;

END;

/