**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.

**Package Specification :-**

CREATE OR REPLACE PACKAGE CustomerManagement AS

PROCEDURE AddCustomer(p\_name VARCHAR2, p\_email VARCHAR2);

PROCEDURE UpdateCustomer(p\_id NUMBER, p\_name VARCHAR2, p\_email VARCHAR2);

FUNCTION GetCustomerBalance(p\_id NUMBER) RETURN NUMBER;

END CustomerManagement;

/

CREATE OR REPLACE PACKAGE BODY CustomerManagement AS

PROCEDURE AddCustomer(p\_name VARCHAR2, p\_email VARCHAR2) IS

BEGIN

INSERT INTO Customers (id, name, email)

VALUES (Customers\_seq.NEXTVAL, p\_name, p\_email);

END;

PROCEDURE UpdateCustomer(p\_id NUMBER, p\_name VARCHAR2, p\_email VARCHAR2) IS

BEGIN

UPDATE Customers

SET name = p\_name,

email = p\_email

WHERE id = p\_id;

END;

FUNCTION GetCustomerBalance(p\_id NUMBER) RETURN NUMBER IS

v\_balance NUMBER := 0;

BEGIN

SELECT NVL(SUM(balance), 0)

INTO v\_balance

FROM Accounts

WHERE customer\_id = p\_id;

RETURN v\_balance;

END;

END CustomerManagement;

/

BEGIN

CustomerManagement.AddCustomer('Naresh');

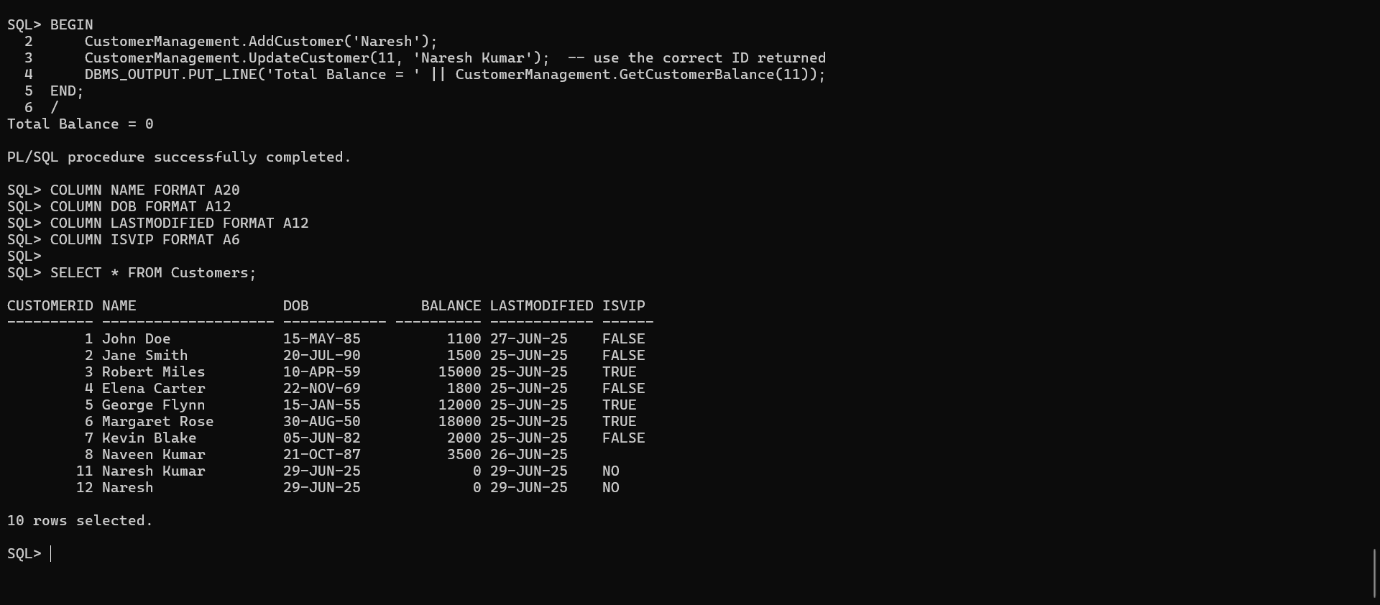
CustomerManagement.UpdateCustomer(11, 'Naresh Kumar'); -- use the correct ID returned

DBMS\_OUTPUT.PUT\_LINE('Total Balance = ' || CustomerManagement.GetCustomerBalance(11));

END;

/

Output :-



**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.

-- Step 1: Create sequence for employee IDs

CREATE SEQUENCE Employees\_seq START WITH 1 INCREMENT BY 1;

-- Step 2: Create package specification

CREATE OR REPLACE PACKAGE EmployeeManagement AS

PROCEDURE HireEmployee(p\_name VARCHAR2, p\_designation VARCHAR2, p\_salary NUMBER);

PROCEDURE UpdateEmployee(p\_id NUMBER, p\_name VARCHAR2, p\_designation VARCHAR2, p\_salary NUMBER);

FUNCTION CalculateAnnualSalary(p\_id NUMBER) RETURN NUMBER;

END EmployeeManagement;

/

-- Step 3: Create package body

CREATE OR REPLACE PACKAGE BODY EmployeeManagement AS

PROCEDURE HireEmployee(p\_name VARCHAR2, p\_designation VARCHAR2, p\_salary NUMBER) IS

BEGIN

INSERT INTO Employees (employeeid, name, designation, salary, hiredate)

VALUES (Employees\_seq.NEXTVAL, p\_name, p\_designation, p\_salary, SYSDATE);

END;

PROCEDURE UpdateEmployee(p\_id NUMBER, p\_name VARCHAR2, p\_designation VARCHAR2, p\_salary NUMBER) IS

BEGIN

UPDATE Employees

SET name = p\_name,

designation = p\_designation,

salary = p\_salary

WHERE employeeid = p\_id;

END;

FUNCTION CalculateAnnualSalary(p\_id NUMBER) RETURN NUMBER IS

v\_salary NUMBER;

BEGIN

SELECT salary INTO v\_salary FROM Employees WHERE employeeid = p\_id;

RETURN v\_salary \* 12;

END;

END EmployeeManagement;

/

-- Step 4: Test block

SET SERVEROUTPUT ON;

BEGIN

-- Hire a new employee

EmployeeManagement.HireEmployee('Naresh', 'Developer', 30000);

-- Update the employee (assuming employeeid = 1)

EmployeeManagement.UpdateEmployee(1, 'Naresh Kumar', 'Senior Developer', 40000);

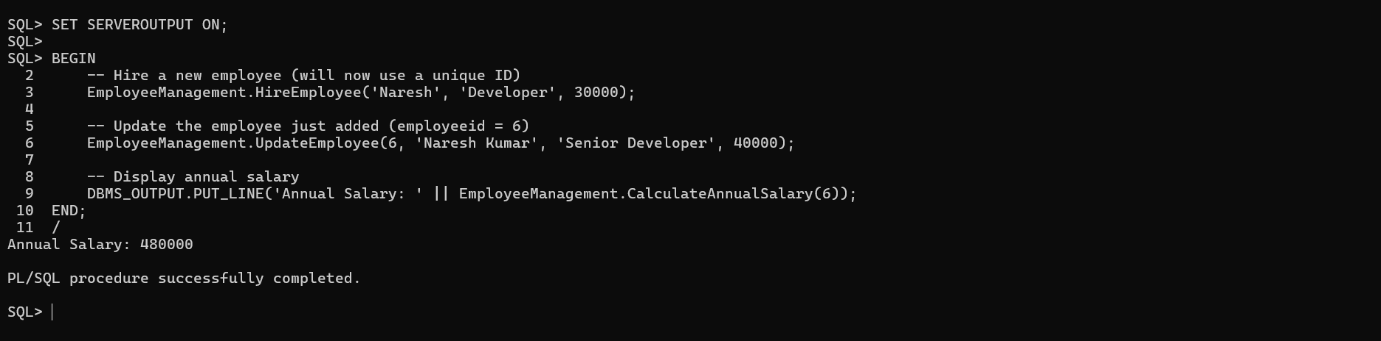
-- Display annual salary

DBMS\_OUTPUT.PUT\_LINE('Annual Salary: ' || EmployeeManagement.CalculateAnnualSalary(1));

END;

/

Output :-



**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.

-- Create the sequence

CREATE SEQUENCE Accounts\_seq START WITH 1 INCREMENT BY 1;

-- Create the package specification

CREATE OR REPLACE PACKAGE AccountOperations AS

PROCEDURE OpenAccount(p\_customerid NUMBER, p\_accounttype VARCHAR2, p\_balance NUMBER);

PROCEDURE CloseAccount(p\_accountid NUMBER);

FUNCTION GetTotalBalance(p\_customerid NUMBER) RETURN NUMBER;

END AccountOperations;

/

-- Create the package body

CREATE OR REPLACE PACKAGE BODY AccountOperations AS

PROCEDURE OpenAccount(p\_customerid NUMBER, p\_accounttype VARCHAR2, p\_balance NUMBER) IS

BEGIN

INSERT INTO Accounts (accountid, customerid, accounttype, balance, lastmodified)

VALUES (Accounts\_seq.NEXTVAL, p\_customerid, p\_accounttype, p\_balance, SYSDATE);

END;

PROCEDURE CloseAccount(p\_accountid NUMBER) IS

BEGIN

DELETE FROM Accounts

WHERE accountid = p\_accountid;

END;

FUNCTION GetTotalBalance(p\_customerid NUMBER) RETURN NUMBER IS

v\_total NUMBER := 0;

BEGIN

SELECT NVL(SUM(balance), 0)

INTO v\_total

FROM Accounts

WHERE customerid = p\_customerid;

RETURN v\_total;

END;

END AccountOperations;

/

-- Test the package

SET SERVEROUTPUT ON;

BEGIN

-- Open two accounts for customer ID 1

AccountOperations.OpenAccount(1, 'Savings', 5000);

AccountOperations.OpenAccount(1, 'Current', 8000);

-- Show total balance for customer ID 1

DBMS\_OUTPUT.PUT\_LINE('Total Balance: ' || AccountOperations.GetTotalBalance(1));

-- Close one account (assume accountid = 1, verify manually if needed)

AccountOperations.CloseAccount(1);

-- Show updated balance

DBMS\_OUTPUT.PUT\_LINE('Updated Total Balance: ' || AccountOperations.GetTotalBalance(1));

END;

/

Output :-

