**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.  
CREATE TABLE customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

DOB DATE,

Balance NUMBER,

LastModified DATE

);

CREATE OR REPLACE PACKAGE CustomerManagement AS

PROCEDURE AddCustomer(p\_id NUMBER, p\_name VARCHAR2, p\_dob DATE, p\_balance NUMBER);

PROCEDURE UpdateCustomerBalance(p\_id NUMBER, p\_new\_balance NUMBER);

FUNCTION GetCustomerBalance(p\_id NUMBER) RETURN NUMBER;

END CustomerManagement;

/

CREATE OR REPLACE PACKAGE BODY CustomerManagement AS

PROCEDURE AddCustomer(p\_id NUMBER, p\_name VARCHAR2, p\_dob DATE, p\_balance NUMBER) IS

BEGIN

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

VALUES (p\_id, p\_name, p\_dob, p\_balance, SYSDATE);

END;

PROCEDURE UpdateCustomerBalance(p\_id NUMBER, p\_new\_balance NUMBER) IS

BEGIN

UPDATE customers

SET Balance = p\_new\_balance, LastModified = SYSDATE

WHERE CustomerID = p\_id;

END;

FUNCTION GetCustomerBalance(p\_id NUMBER) RETURN NUMBER IS

v\_balance NUMBER;

BEGIN

SELECT Balance INTO v\_balance FROM customers WHERE CustomerID = p\_id;

RETURN v\_balance;

END;

END CustomerManagement;

/

--test

BEGIN

CustomerManagement.AddCustomer(3, 'Koushika', TO\_DATE('1999-01-01','YYYY-MM-DD'), 2000);

END;

/

BEGIN

CustomerManagement.UpdateCustomerBalance(3, 2500);

END;

/

DECLARE

v\_bal NUMBER;

BEGIN

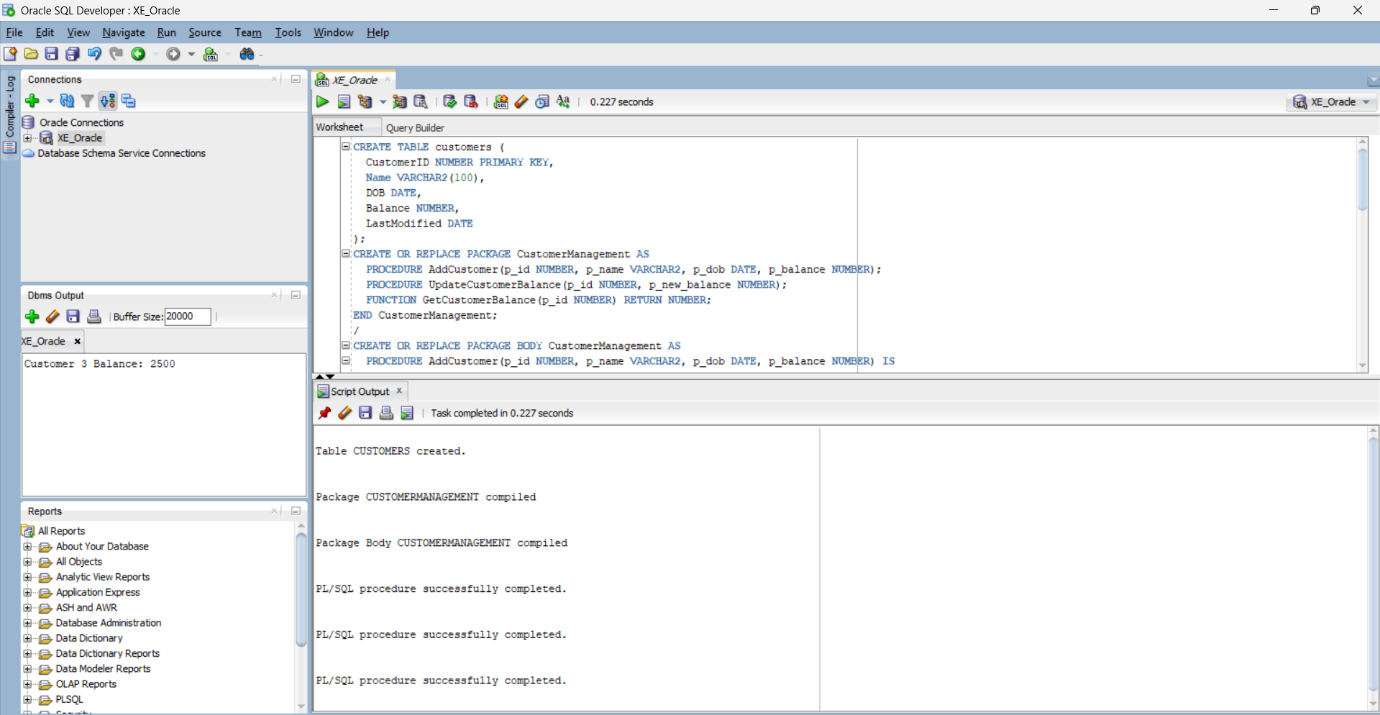
v\_bal := CustomerManagement.GetCustomerBalance(3);

DBMS\_OUTPUT.PUT\_LINE('Customer 3 Balance: ' || v\_bal);

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.

SET SERVEROUTPUT ON;

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE employees CASCADE CONSTRAINTS';

EXCEPTION

WHEN OTHERS THEN

IF SQLCODE != -942 THEN

RAISE;

END IF;

END;

/

CREATE TABLE employees (

EmployeeID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

Position VARCHAR2(50),

Salary NUMBER,

Department VARCHAR2(50),

HireDate DATE

);

/

CREATE OR REPLACE PACKAGE EmployeeManagement AS

PROCEDURE HireEmployee(

p\_id NUMBER, p\_name VARCHAR2, p\_position VARCHAR2,

p\_salary NUMBER, p\_department VARCHAR2, p\_hiredate DATE

);

PROCEDURE UpdateEmployeeSalary(p\_id NUMBER, p\_new\_salary NUMBER);

FUNCTION GetAnnualSalary(p\_id NUMBER) RETURN NUMBER;

END EmployeeManagement;

/

CREATE OR REPLACE PACKAGE BODY EmployeeManagement AS

PROCEDURE HireEmployee(

p\_id NUMBER, p\_name VARCHAR2, p\_position VARCHAR2,

p\_salary NUMBER, p\_department VARCHAR2, p\_hiredate DATE

) IS

BEGIN

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

VALUES (p\_id, p\_name, p\_position, p\_salary, p\_department, p\_hiredate);

END;

PROCEDURE UpdateEmployeeSalary(p\_id NUMBER, p\_new\_salary NUMBER) IS

BEGIN

UPDATE employees

SET Salary = p\_new\_salary

WHERE EmployeeID = p\_id;

END;

FUNCTION GetAnnualSalary(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;

/

BEGIN

EmployeeManagement.HireEmployee(

3, 'Siva', 'Analyst', 55000, 'Finance', SYSDATE

);

END;

/

DECLARE

v\_annual NUMBER;

BEGIN

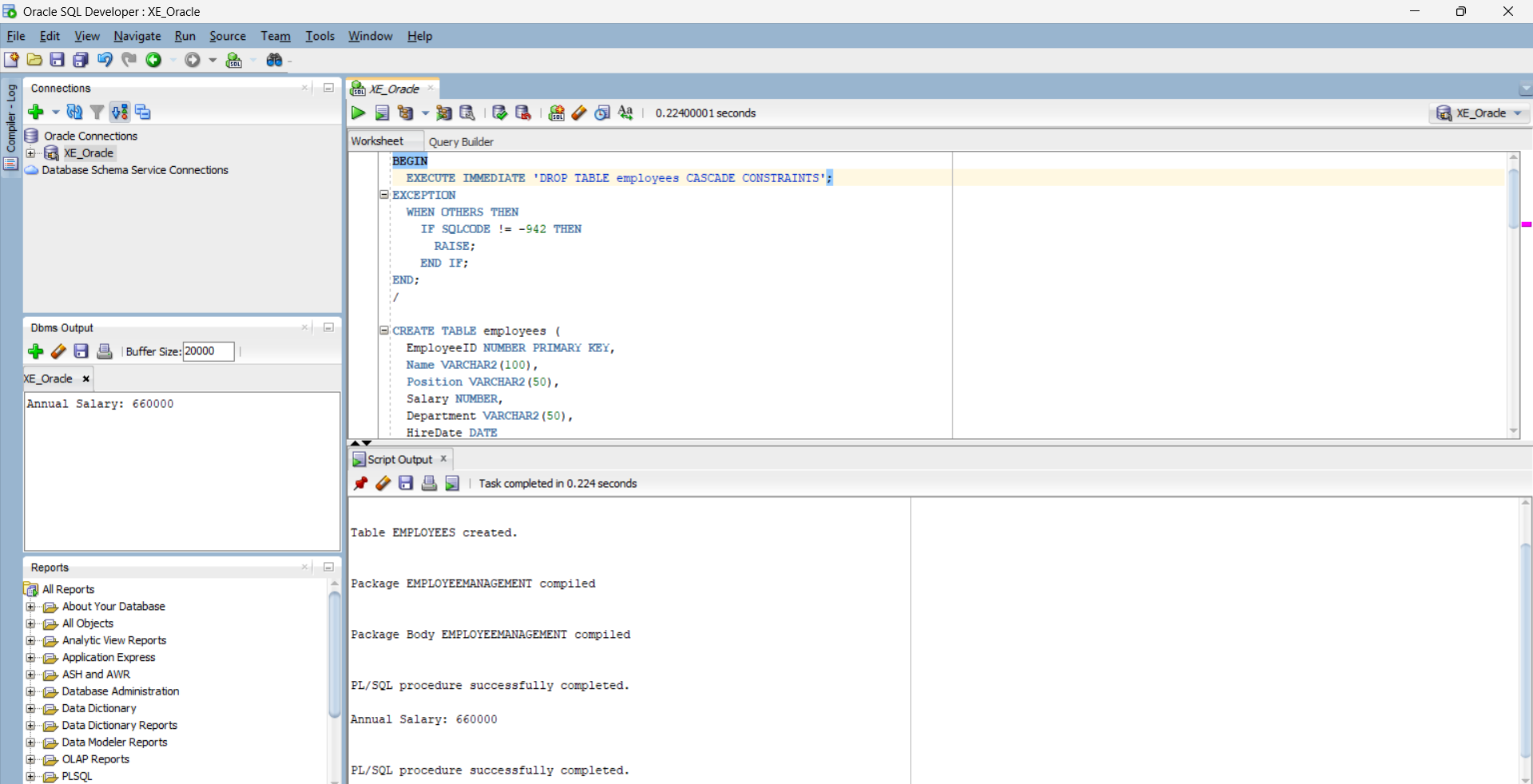
v\_annual := EmployeeManagement.GetAnnualSalary(3);

DBMS\_OUTPUT.PUT\_LINE('Annual Salary: ' || v\_annual);

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.  
  
SET SERVEROUTPUT ON;

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE accounts CASCADE CONSTRAINTS';

EXCEPTION

WHEN OTHERS THEN

IF SQLCODE != -942 THEN

RAISE;

END IF;

END;

/

CREATE TABLE accounts (

AccountID NUMBER PRIMARY KEY,

CustomerID NUMBER,

AccountType VARCHAR2(20),

Balance NUMBER,

LastModified DATE

);

/

CREATE OR REPLACE PACKAGE AccountOperations AS

PROCEDURE OpenAccount(p\_accID NUMBER, p\_custID NUMBER, p\_type VARCHAR2, p\_balance NUMBER);

PROCEDURE CloseAccount(p\_accID NUMBER);

FUNCTION GetTotalBalance(p\_custID NUMBER) RETURN NUMBER;

END AccountOperations;

/

CREATE OR REPLACE PACKAGE BODY AccountOperations AS

PROCEDURE OpenAccount(p\_accID NUMBER, p\_custID NUMBER, p\_type VARCHAR2, p\_balance NUMBER) IS

BEGIN

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

VALUES (p\_accID, p\_custID, p\_type, p\_balance, SYSDATE);

END;

PROCEDURE CloseAccount(p\_accID NUMBER) IS

BEGIN

DELETE FROM accounts WHERE AccountID = p\_accID;

END;

FUNCTION GetTotalBalance(p\_custID NUMBER) RETURN NUMBER IS

v\_total NUMBER := 0;

BEGIN

SELECT NVL(SUM(Balance), 0) INTO v\_total

FROM accounts

WHERE CustomerID = p\_custID;

RETURN v\_total;

END;

END AccountOperations;

/

BEGIN

AccountOperations.OpenAccount(101, 1, 'Savings', 2500);

AccountOperations.OpenAccount(102, 1, 'Checking', 1500);

END;

/

DECLARE

v\_balance NUMBER;

BEGIN

v\_balance := AccountOperations.GetTotalBalance(1);

DBMS\_OUTPUT.PUT\_LINE('Total Balance for Customer 1: ' || v\_balance);

END;

/

BEGIN

AccountOperations.CloseAccount(102);

END;

/

DECLARE

v\_balance NUMBER;

BEGIN

v\_balance := AccountOperations.GetTotalBalance(1);

DBMS\_OUTPUT.PUT\_LINE('Balance after closing one account: ' || v\_balance);

END;

/

