**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.

**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.

**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.

**Scenario 1: Package CustomerManagement**

**Step 1: Package Specification**

**CODE:-**

CREATE OR REPLACE PACKAGE CustomerManagement AS

  PROCEDURE AddCustomer(

    p\_CustomerID IN NUMBER,

    p\_Name IN VARCHAR2,

    p\_DOB IN DATE,

    p\_Balance IN NUMBER

  );

  PROCEDURE UpdateCustomer(

    p\_CustomerID IN NUMBER,

    p\_Name IN VARCHAR2,

    p\_Balance IN NUMBER

  );

  FUNCTION GetCustomerBalance(p\_CustomerID IN NUMBER) RETURN NUMBER;

END CustomerManagement;

CREATE OR REPLACE PACKAGE BODY CustomerManagement AS

  PROCEDURE AddCustomer(

    p\_CustomerID IN NUMBER,

    p\_Name IN VARCHAR2,

    p\_DOB IN DATE,

    p\_Balance IN NUMBER

  ) IS

  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('Customer ID already exists.');

  END;

  PROCEDURE UpdateCustomer(

    p\_CustomerID IN NUMBER,

    p\_Name IN VARCHAR2,

    p\_Balance IN NUMBER

  ) IS

  BEGIN

    UPDATE Customers

    SET Name = p\_Name,

        Balance = p\_Balance,

        LastModified = SYSDATE

    WHERE CustomerID = p\_CustomerID;

    IF SQL%ROWCOUNT = 0 THEN

      DBMS\_OUTPUT.PUT\_LINE('Customer not found.');

    ELSE

      COMMIT;

    END IF;

  END;

  FUNCTION GetCustomerBalance(p\_CustomerID IN NUMBER) RETURN NUMBER IS

    v\_Balance NUMBER;

  BEGIN

    SELECT Balance INTO v\_Balance FROM Customers WHERE CustomerID = p\_CustomerID;

    RETURN v\_Balance;

  EXCEPTION

    WHEN NO\_DATA\_FOUND THEN

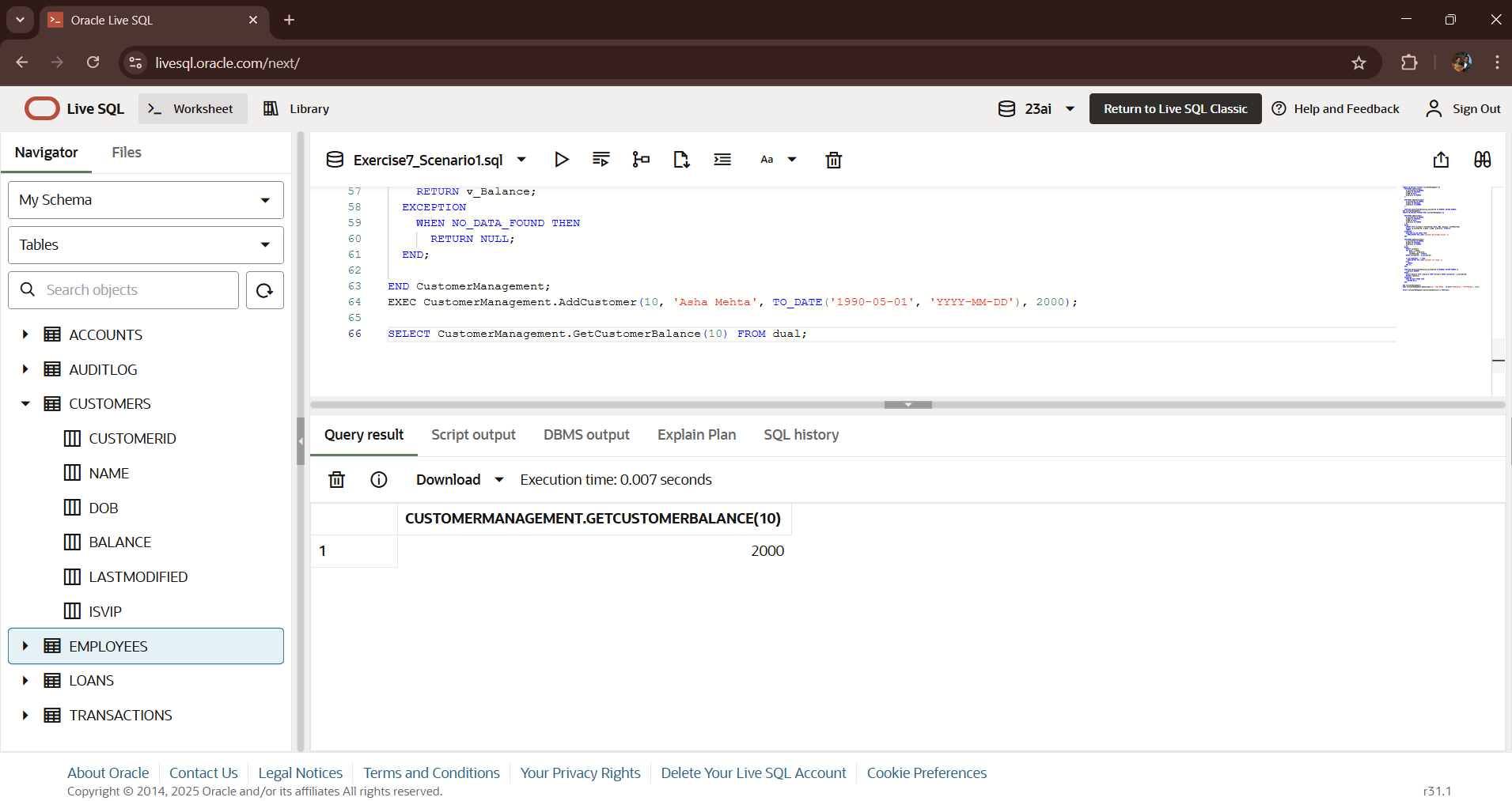
      RETURN NULL;

  END;

END CustomerManagement;

EXEC CustomerManagement.AddCustomer(10, 'Asha Mehta', TO\_DATE('1990-05-01', 'YYYY-MM-DD'), 2000);

SELECT CustomerManagement.GetCustomerBalance(10) FROM dual;

**OUTPUT:**

**Scenario 2: Package EmployeeManagement**

**CODE:-**

CREATE OR REPLACE PACKAGE 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

  );

  PROCEDURE UpdateEmployee(

    p\_EmployeeID IN NUMBER,

    p\_Position IN VARCHAR2,

    p\_Salary IN NUMBER

  );

  FUNCTION GetAnnualSalary(p\_EmployeeID IN NUMBER) RETURN NUMBER;

END EmployeeManagement;

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

    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('Employee ID already exists.');

  END;

  PROCEDURE UpdateEmployee(

    p\_EmployeeID IN NUMBER,

    p\_Position IN VARCHAR2,

    p\_Salary IN NUMBER

  ) IS

  BEGIN

    UPDATE Employees

    SET Position = p\_Position,

        Salary = p\_Salary

    WHERE EmployeeID = p\_EmployeeID;

    IF SQL%ROWCOUNT = 0 THEN

      DBMS\_OUTPUT.PUT\_LINE('Employee not found.');

    ELSE

      COMMIT;

    END IF;

  END;

  FUNCTION GetAnnualSalary(p\_EmployeeID IN NUMBER) RETURN NUMBER IS

    v\_Salary NUMBER;

  BEGIN

    SELECT Salary INTO v\_Salary FROM Employees WHERE EmployeeID = p\_EmployeeID;

    RETURN v\_Salary \* 12;

  EXCEPTION

    WHEN NO\_DATA\_FOUND THEN

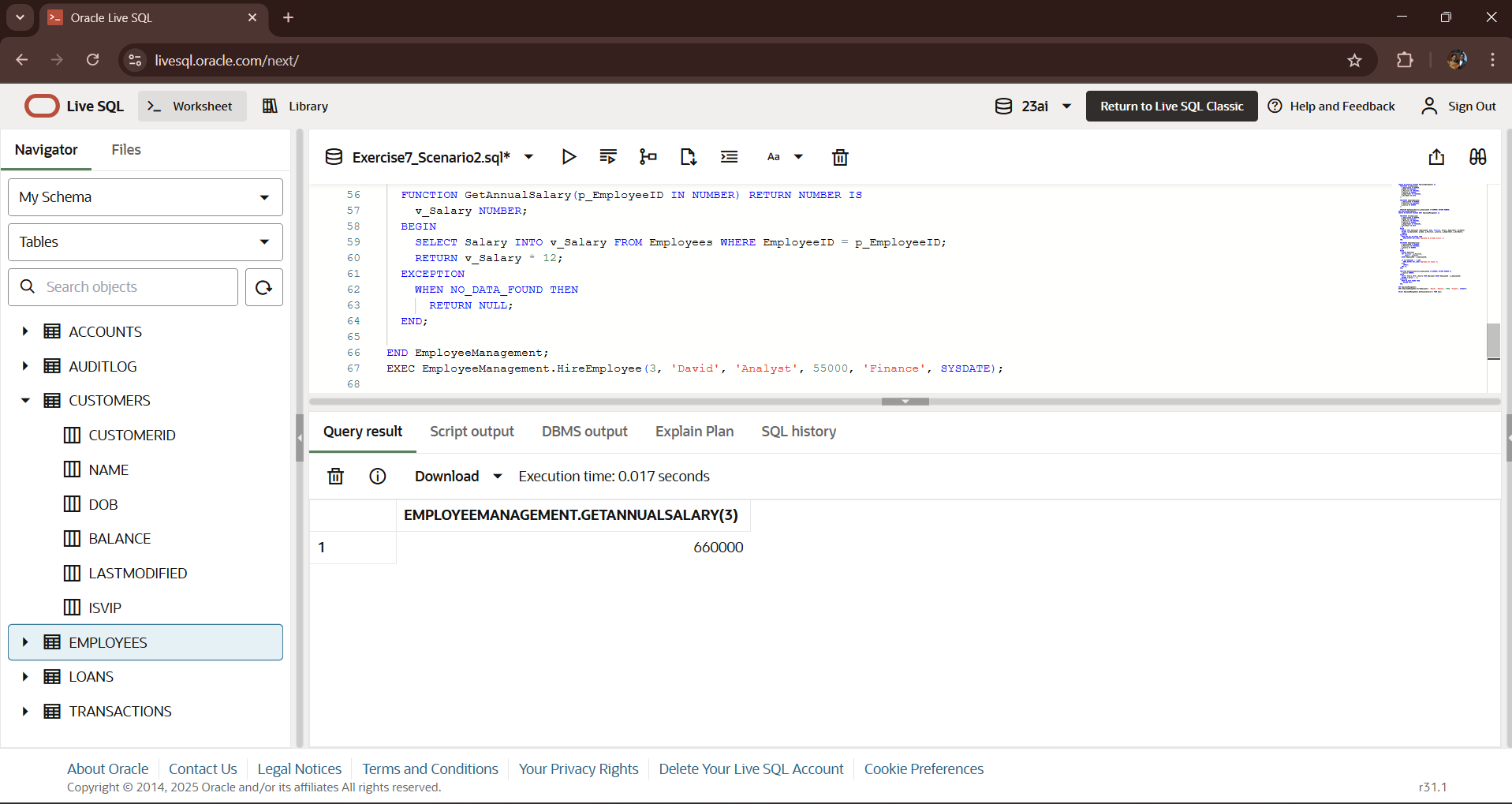
      RETURN NULL;

  END;

END EmployeeManagement;

EXEC EmployeeManagement.HireEmployee(3, 'David', 'Analyst', 55000, 'Finance', SYSDATE);

SELECT EmployeeManagement.GetAnnualSalary(3) FROM dual;

**OUTPUT:-**

**Scenario 3: Package AccountOperations**

**CODE:-**

CREATE OR REPLACE PACKAGE AccountOperations AS

  PROCEDURE OpenAccount(

    p\_AccountID IN NUMBER,

    p\_CustomerID IN NUMBER,

    p\_AccountType IN VARCHAR2,

    p\_Balance IN NUMBER

  );

  PROCEDURE CloseAccount(p\_AccountID IN NUMBER);

  FUNCTION GetTotalBalance(p\_CustomerID IN NUMBER) RETURN NUMBER;

END AccountOperations;

CREATE OR REPLACE PACKAGE BODY AccountOperations AS

  PROCEDURE OpenAccount(

    p\_AccountID IN NUMBER,

    p\_CustomerID IN NUMBER,

    p\_AccountType IN VARCHAR2,

    p\_Balance IN NUMBER

  ) IS

  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('Account ID already exists.');

  END;

  PROCEDURE CloseAccount(p\_AccountID IN NUMBER) IS

  BEGIN

    DELETE FROM Accounts WHERE AccountID = p\_AccountID;

    IF SQL%ROWCOUNT = 0 THEN

      DBMS\_OUTPUT.PUT\_LINE('Account not found.');

    ELSE

      COMMIT;

    END IF;

  END;

  FUNCTION GetTotalBalance(p\_CustomerID IN NUMBER) RETURN NUMBER IS

    v\_Total NUMBER;

  BEGIN

    SELECT SUM(Balance) INTO v\_Total FROM Accounts WHERE CustomerID = p\_CustomerID;

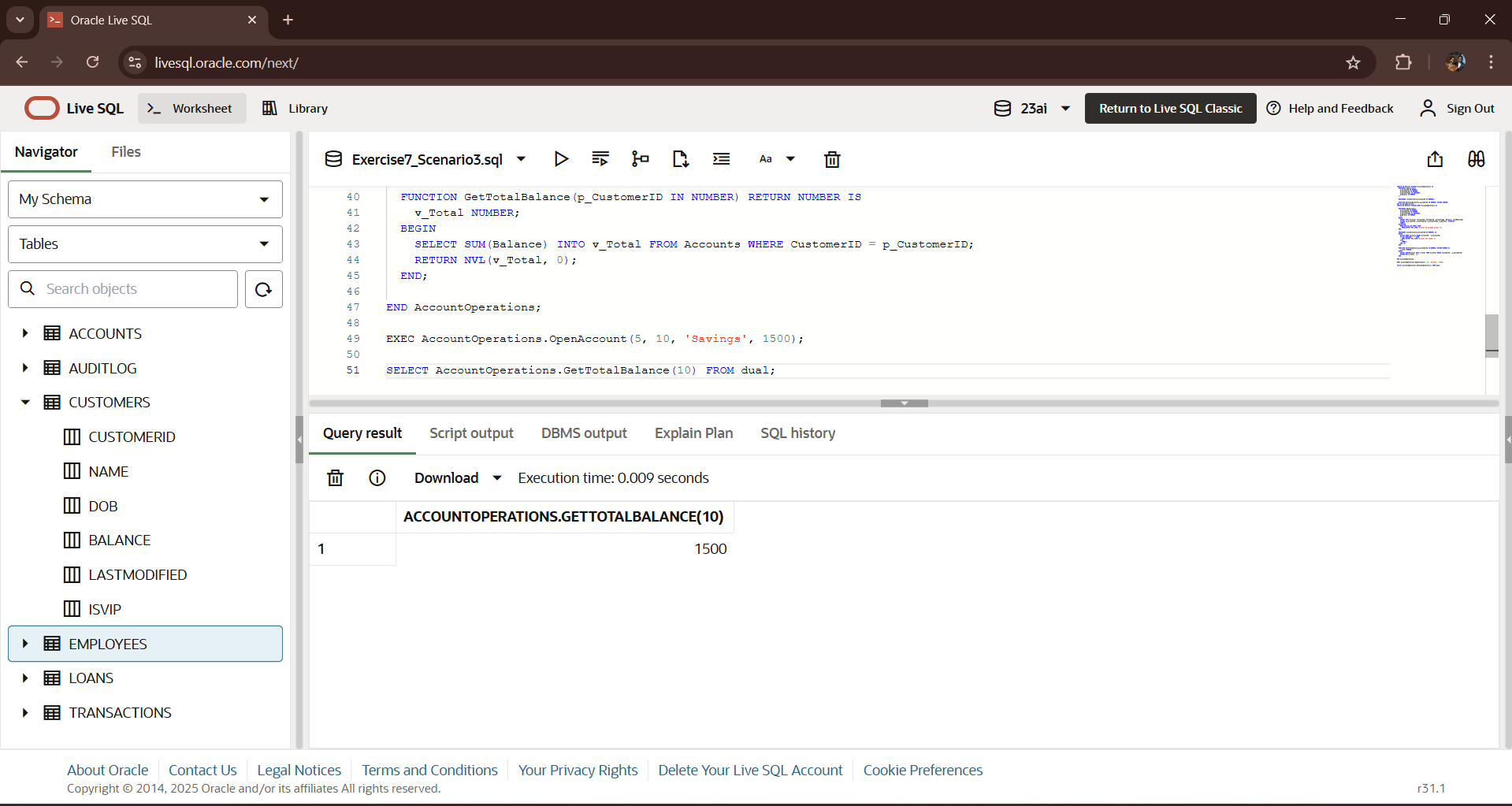
    RETURN NVL(v\_Total, 0);

  END;

END AccountOperations;

EXEC AccountOperations.OpenAccount(5, 10, 'Savings', 1500);

SELECT AccountOperations.GetTotalBalance(10) FROM dual;

**OUTPUT:-**