**WEEK 2**

**PL/SQL EXERCISES**

**Exercise 1: CONTROL STRUCTURES**

SET SERVEROUTPUT ON;

BEGIN

    EXECUTE IMMEDIATE 'DROP TABLE Loans';

    EXECUTE IMMEDIATE 'DROP TABLE Customers';

EXCEPTION

    WHEN OTHERS THEN

        NULL;

END;

/

**-- Customers Table**

CREATE TABLE Customers (

    CustomerID     NUMBER PRIMARY KEY,

    Name           VARCHAR2(100),

    Age            NUMBER,

    Balance        NUMBER(10,2),

    IsVIP          VARCHAR2(5) DEFAULT 'FALSE'

);

**-- Loans Table**

CREATE TABLE Loans (

    LoanID         NUMBER PRIMARY KEY,

    CustomerID     NUMBER,

    InterestRate   NUMBER(5,2),

    DueDate        DATE,

    FOREIGN KEY (CustomerID) REFERENCES Customers (CustomerID)

);

**-- Customers Data**

BEGIN

    INSERT INTO Customers VALUES (1, 'Sara', 34, 100000, 'FALSE');

    DBMS\_OUTPUT.PUT\_LINE('Inserted Customer: Sara');

    INSERT INTO Customers VALUES (2, 'Radha', 40, 80000, 'FALSE');

    DBMS\_OUTPUT.PUT\_LINE('Inserted Customer: Radha');

    INSERT INTO Customers VALUES (3, 'Eva', 30, 67000, 'FALSE');

    DBMS\_OUTPUT.PUT\_LINE('Inserted Customer: Eva');

    INSERT INTO Customers VALUES (4, 'Rita', 65, 120000, 'FALSE');

    DBMS\_OUTPUT.PUT\_LINE('Inserted Customer: Rita');

    COMMIT;

    DBMS\_OUTPUT.PUT\_LINE('Customer records committed.');

END;

/

**-- Loan Data**

BEGIN

    INSERT INTO Loans VALUES (101, 1, 5.5, SYSDATE + 10);

    DBMS\_OUTPUT.PUT\_LINE('Inserted Loan for Customer 1');

    INSERT INTO Loans VALUES (102, 2, 6.0, SYSDATE + 40);

    DBMS\_OUTPUT.PUT\_LINE('Inserted Loan for Customer 2');

    INSERT INTO Loans VALUES (103, 3, 5.0, SYSDATE + 20);

    DBMS\_OUTPUT.PUT\_LINE('Inserted Loan for Customer 3');

    INSERT INTO Loans VALUES (104, 4, 5.0, SYSDATE + 30); -- loan for senior customer

    DBMS\_OUTPUT.PUT\_LINE('Inserted Loan for Customer 4');

    COMMIT;

    DBMS\_OUTPUT.PUT\_LINE('Loan records committed.');

END;

/

**Scenario 1**  
BEGIN

    DECLARE

        v\_found BOOLEAN := FALSE;

    BEGIN

        FOR cust\_rec IN (

            SELECT c.CustomerID, l.LoanID, l.InterestRate

            FROM Customers c

            JOIN Loans l ON c.CustomerID = l.CustomerID

            WHERE c.Age > 60

        ) LOOP

            v\_found := TRUE;

            UPDATE Loans

            SET InterestRate = InterestRate - 1

            WHERE LoanID = cust\_rec.LoanID;

            DBMS\_OUTPUT.PUT\_LINE('The discount applied for customer\_ID: ' || cust\_rec.CustomerID);

        END LOOP;

        IF NOT v\_found THEN

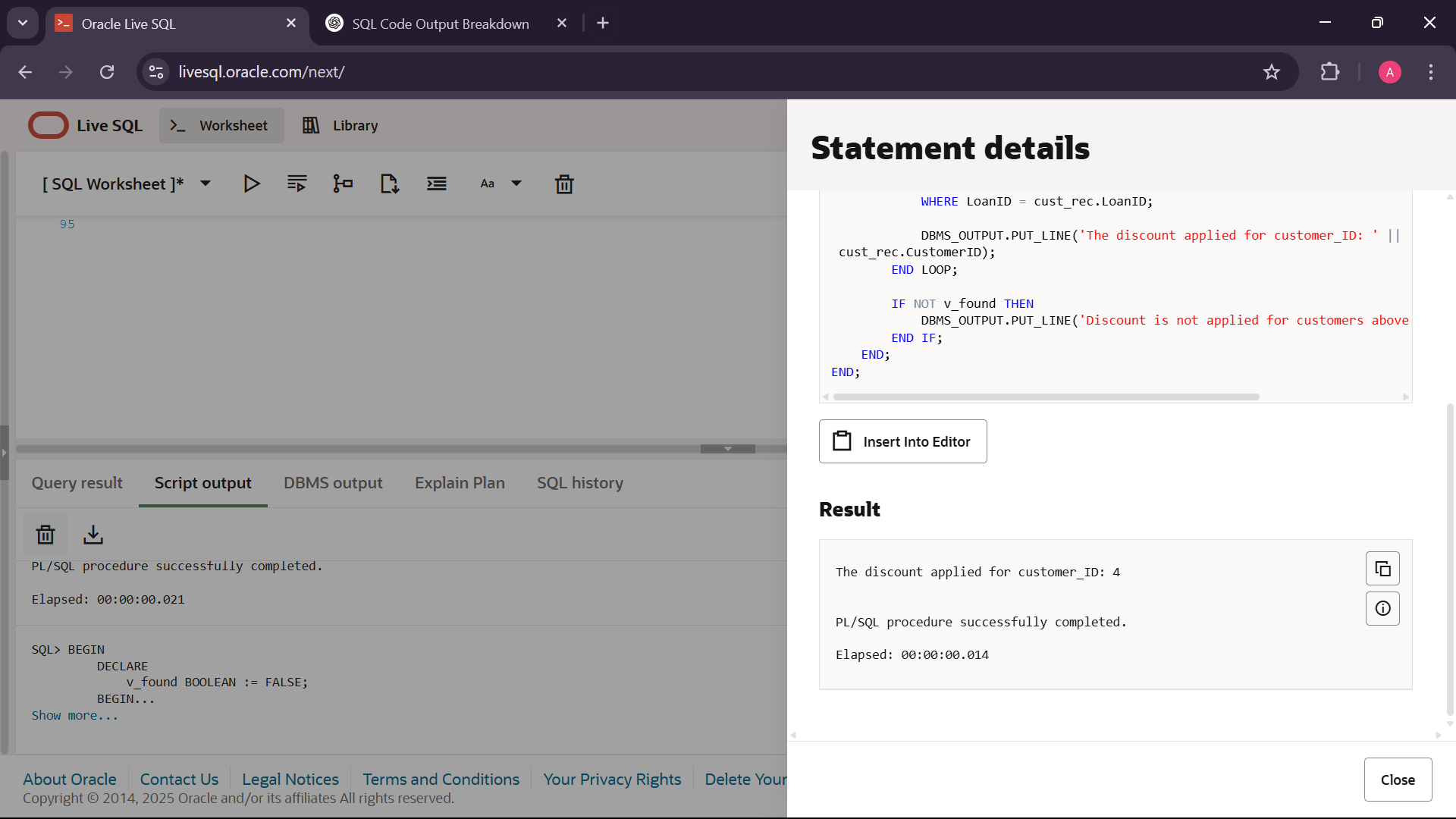
            DBMS\_OUTPUT.PUT\_LINE('Discount is not applied for customers above 60 years old.');

        END IF;

    END;

END;

/



**Scenario 2**

BEGIN

    FOR cust\_rec IN (

        SELECT CustomerID, Name, Balance

        FROM Customers

        WHERE Balance > 100000 AND IsVIP = 'FALSE'

    ) LOOP

        UPDATE Customers

        SET IsVIP = 'TRUE'

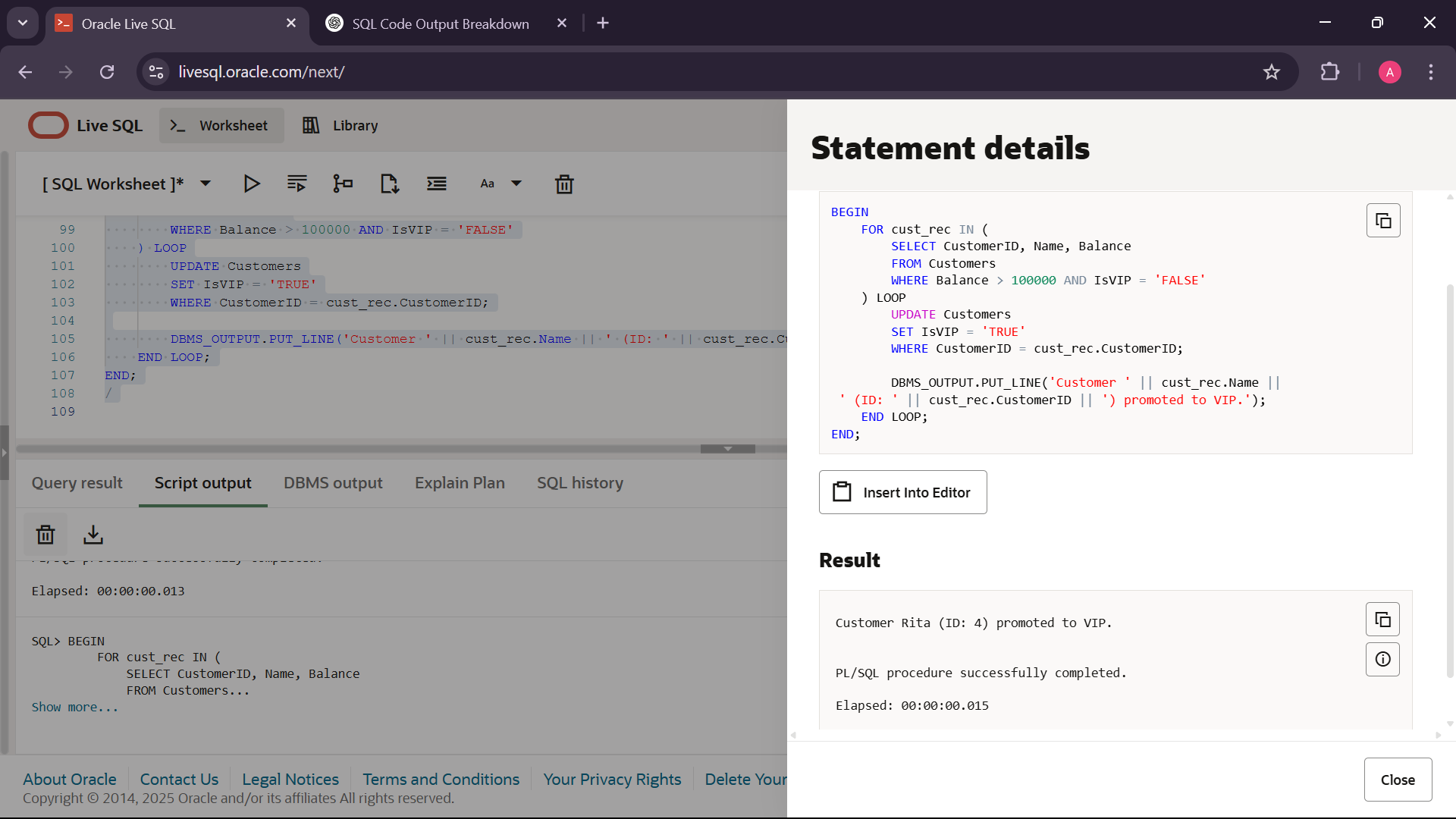
        WHERE CustomerID = cust\_rec.CustomerID;

        DBMS\_OUTPUT.PUT\_LINE('Customer ' || cust\_rec.Name || ' (ID: ' || cust\_rec.CustomerID || ') promoted to VIP.');

    END LOOP;

END;

/



**Scenario 3**

BEGIN

    FOR loan\_rec IN (

        SELECT LoanID, CustomerID, InterestRate, DueDate

        FROM Loans

        WHERE DueDate BETWEEN SYSDATE AND SYSDATE + 30

    ) LOOP

        DBMS\_OUTPUT.PUT\_LINE(

            'Loan ID: ' || loan\_rec.LoanID ||

            ', Customer ID: ' || loan\_rec.CustomerID ||

            ', Interest Rate: ' || loan\_rec.InterestRate ||

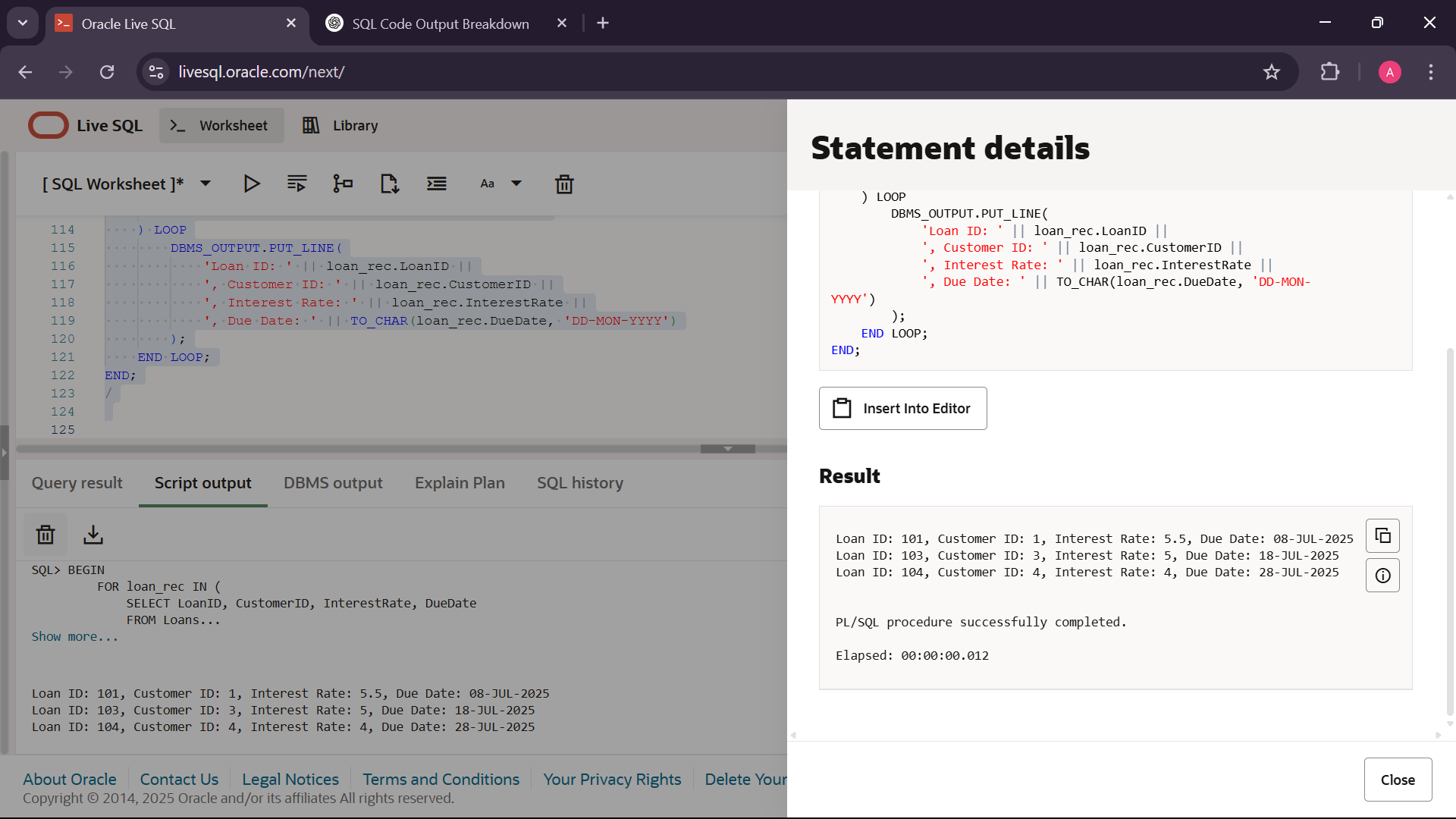
            ', Due Date: ' || TO\_CHAR(loan\_rec.DueDate, 'DD-MON-YYYY')

        );

    END LOOP;

END;

/



**Exercise 3: STORED PROCEDURES**

**-- Accounts table**

DROP TABLE Accounts CASCADE CONSTRAINTS;

CREATE TABLE Accounts (

    AccountID     NUMBER PRIMARY KEY,

    CustomerID    NUMBER,

    AccountType   VARCHAR2(20),

    Balance       NUMBER(15, 2)

);

**-- Accounts Data**

INSERT INTO Accounts VALUES (101, 1, 'Savings', 10000);

INSERT INTO Accounts VALUES (102, 2, 'Checking', 8000);

INSERT INTO Accounts VALUES (103, 3, 'Savings', 15000);

COMMIT;

**-- Employees table**

DROP TABLE Employees CASCADE CONSTRAINTS;

CREATE TABLE Employees (

    EmployeeID     NUMBER PRIMARY KEY,

    Name           VARCHAR2(100),

    DepartmentID   NUMBER,

    Salary         NUMBER(10, 2)

);

**-- Employees Data**

INSERT INTO Employees VALUES (01, 'Aliya', 10, 45000);

INSERT INTO Employees VALUES (02, 'Bhanu', 30, 55000);

INSERT INTO Employees VALUES (03, 'Charmi', 20, 70000);

COMMIT;

**-- Monthly Interest**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

    UPDATE Accounts

    SET Balance = Balance + (Balance \* 0.01)

    WHERE AccountType = 'Savings';

    DBMS\_OUTPUT.PUT\_LINE('Monthly interest applied to all savings accounts.');

END;

/

**-- Employee Bonus**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

    p\_DepartmentID IN NUMBER,

    p\_BonusPercent IN NUMBER

) IS

BEGIN

    UPDATE Employees

    SET Salary = Salary + (Salary \* p\_BonusPercent / 100)

    WHERE DepartmentID = p\_DepartmentID;

    DBMS\_OUTPUT.PUT\_LINE('Bonus applied to Department ' || p\_DepartmentID || '.');

END;

/

**-- Transfer Funds**

CREATE OR REPLACE PROCEDURE TransferFunds(

    p\_SourceAccountID IN NUMBER,

    p\_DestAccountID   IN NUMBER,

    p\_Amount          IN NUMBER

) IS

    v\_SourceBalance NUMBER;

BEGIN

    SELECT Balance INTO v\_SourceBalance

    FROM Accounts

    WHERE AccountID = p\_SourceAccountID

    FOR UPDATE;

    IF v\_SourceBalance < p\_Amount THEN

        RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient balance in source account.');

    END IF;

    UPDATE Accounts

    SET Balance = Balance - p\_Amount

    WHERE AccountID = p\_SourceAccountID;

    UPDATE Accounts

    SET Balance = Balance + p\_Amount

    WHERE AccountID = p\_DestAccountID;

    DBMS\_OUTPUT.PUT\_LINE('Transferred ' || p\_Amount || ' from account ' || p\_SourceAccountID || ' to account ' || p\_DestAccountID);

END;

SET SERVEROUTPUT ON;

**-- Scenario 1: Apply Interest**

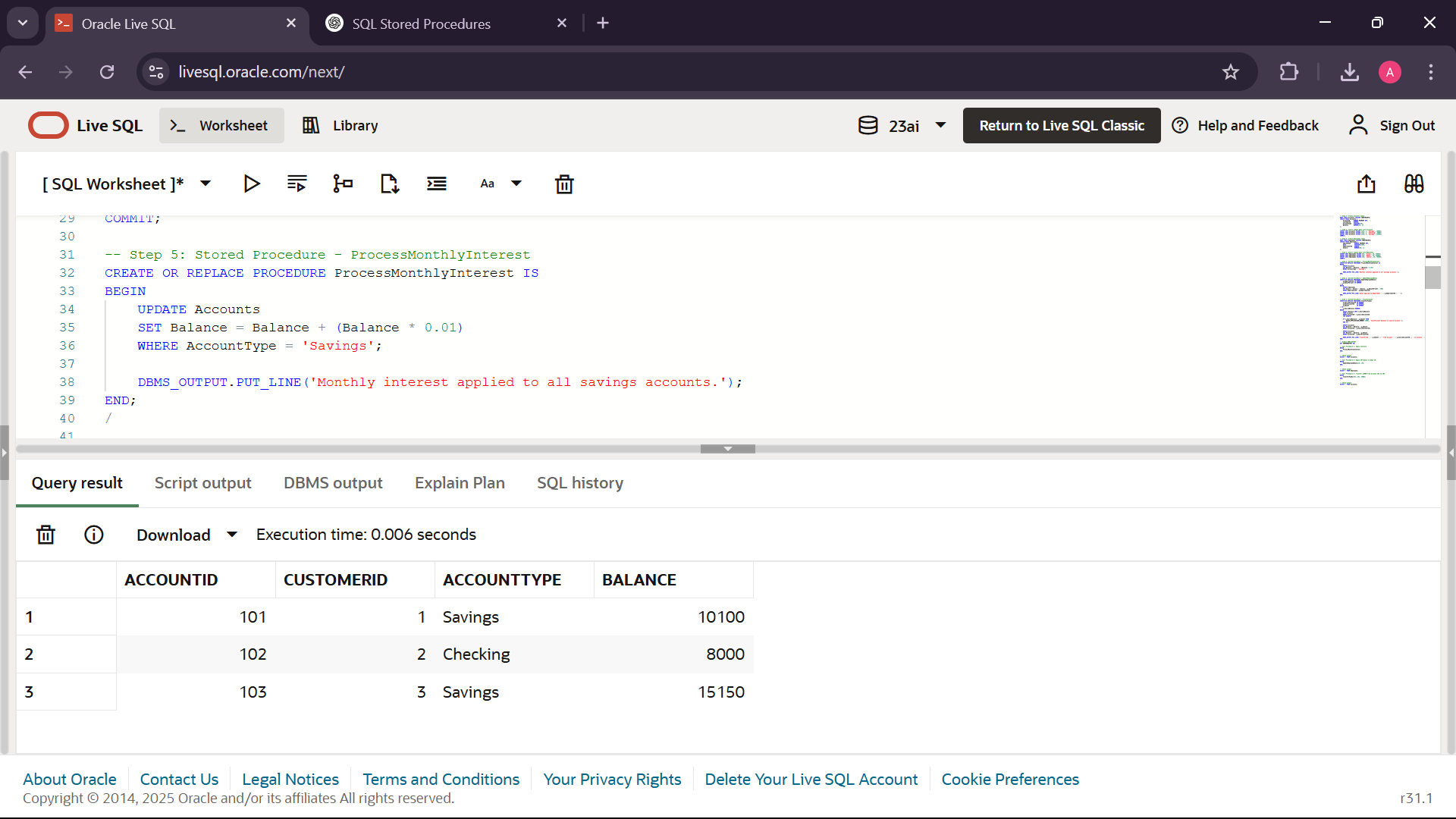
BEGIN

    ProcessMonthlyInterest;

END;

/

SELECT \* FROM Accounts;



**-- Scenario 2: Apply 10% Bonus to Dept 10**

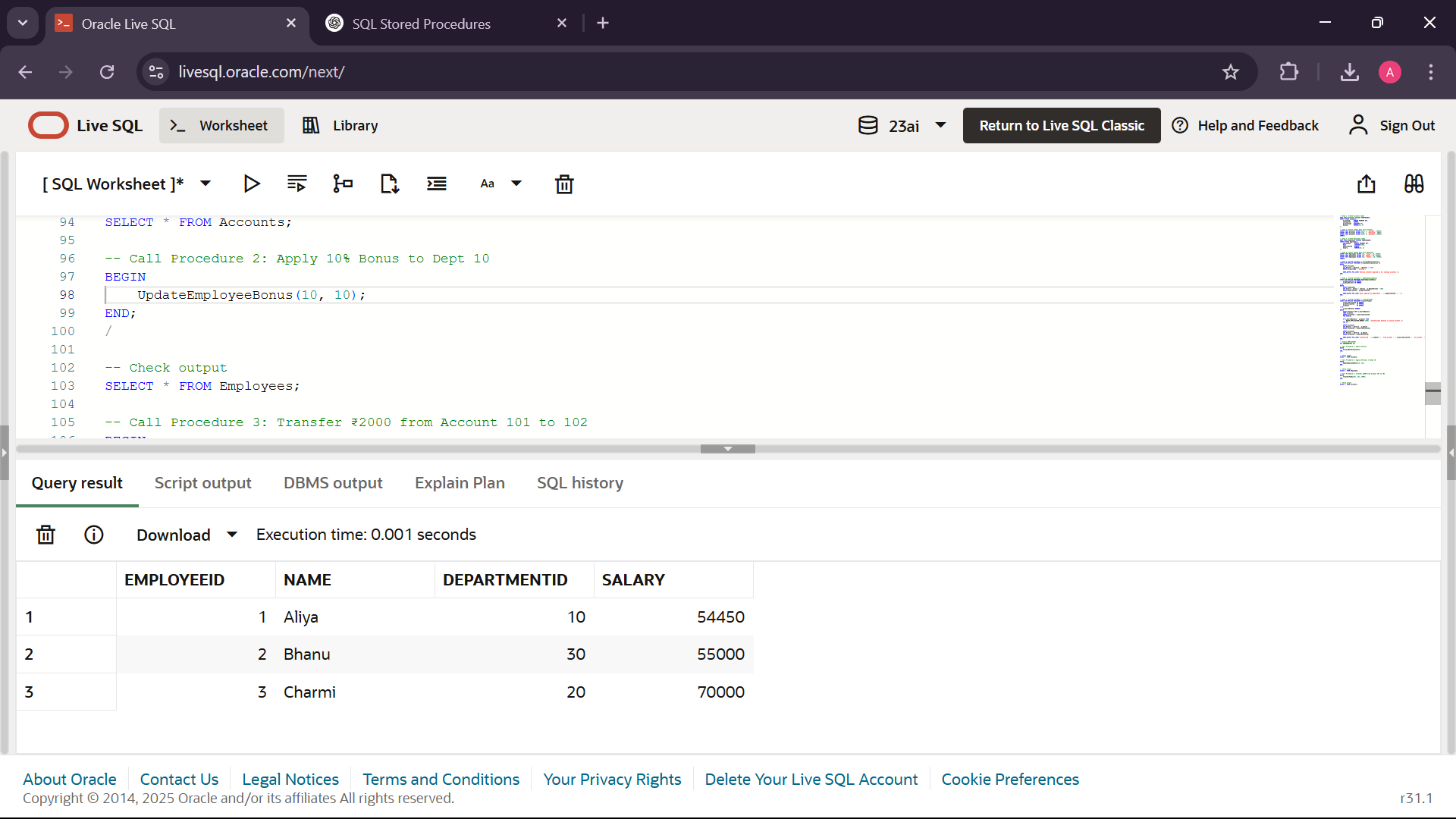
BEGIN

    UpdateEmployeeBonus(10, 10);

END;

/

SELECT \* FROM Employees;



**-- Scenario 3: Transfer ₹2000 from Account 101 to 102**

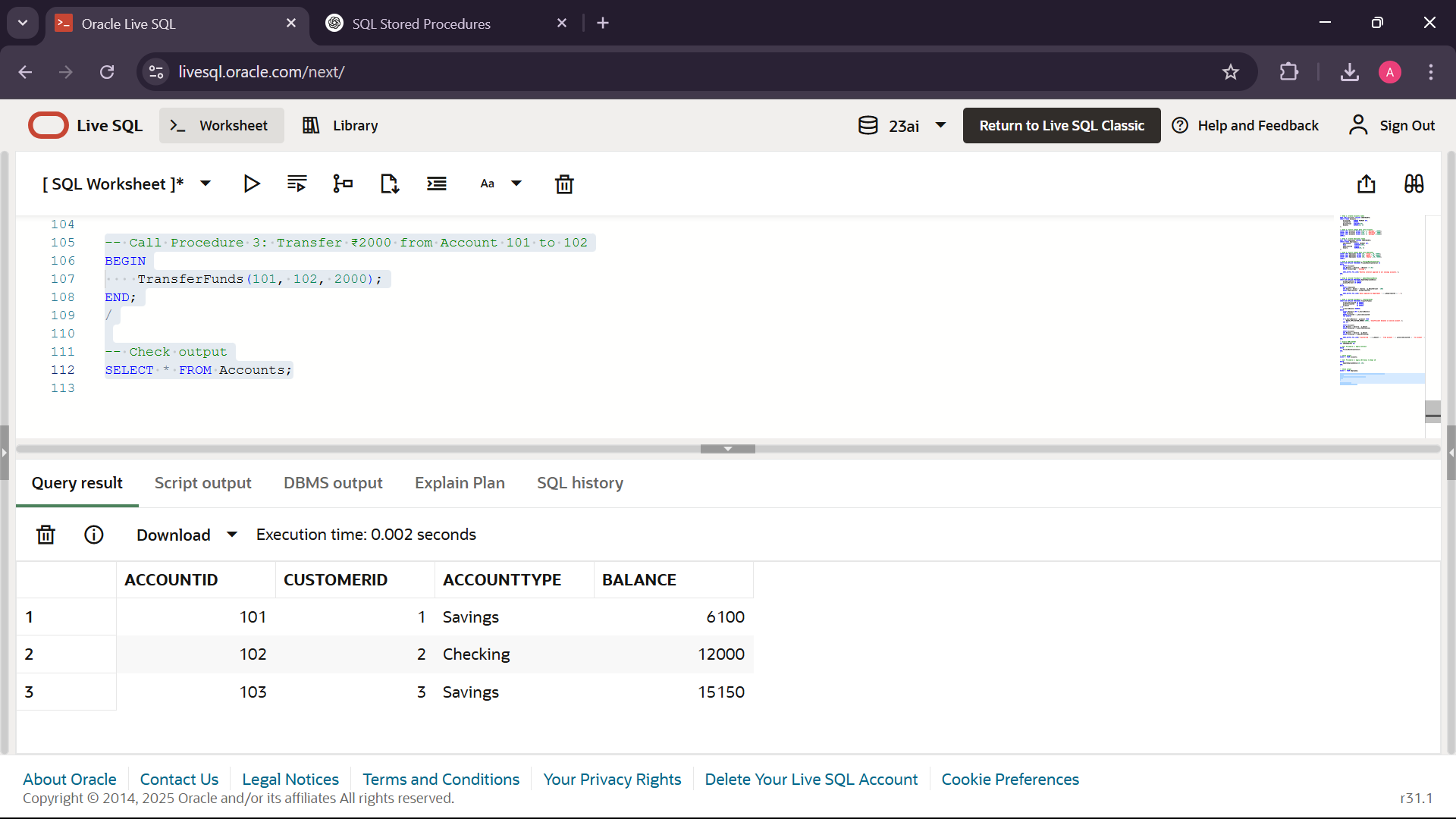
BEGIN

    TransferFunds(101, 102, 2000);

END;

/

SELECT \* FROM Accounts;

****