Hands-on 2:

PL/SQL programming

Exercise 1: Control Structures

Table creating:

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(50),

Age NUMBER,

Balance NUMBER,

LoanInterestRate NUMBER,

IsVIP CHAR(1)

);

CREATE TABLE Loans (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER,

LoanAmount NUMBER,

DueDate DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

INSERT INTO Customers (CustomerID, Name, Age, Balance, LoanInterestRate, IsVIP) VALUES (1, 'John Doe', 65, 12000, 10.5, 'N');

INSERT INTO Customers (CustomerID, Name, Age, Balance, LoanInterestRate, IsVIP) VALUES (2, 'Alice Smith', 45, 9500, 11.0, 'N');

INSERT INTO Customers (CustomerID, Name, Age, Balance, LoanInterestRate, IsVIP) VALUES (3, 'Bob Lee', 70, 15000, 9.5, 'N');

INSERT INTO Customers (CustomerID, Name, Age, Balance, LoanInterestRate, IsVIP) VALUES (4, 'Maria Davis', 58, 5000, 12.0, 'N');

INSERT INTO Customers (CustomerID, Name, Age, Balance, LoanInterestRate, IsVIP) VALUES (5, 'Tom Clark', 61, 20000, 10.0, 'N');

COMMIT;

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, DueDate) VALUES (101, 1, 50000, SYSDATE + 15);

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, DueDate) VALUES (102, 2, 75000, SYSDATE + 35);

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, DueDate) VALUES (103, 3, 30000, SYSDATE + 5);

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, DueDate) VALUES (104, 4, 90000, SYSDATE - 10);

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, DueDate) VALUES (105, 5, 100000, SYSDATE + 29);

-- Scenario 1: The bank wants to apply a discount to loan interest rates for customers above 60 years old.

BEGIN

FOR rec IN (SELECT CustomerID, Age FROM Customers) LOOP

IF rec.Age > 60 THEN

UPDATE Customers

SET LoanInterestRate = LoanInterestRate - 1

WHERE CustomerID = rec.CustomerID;

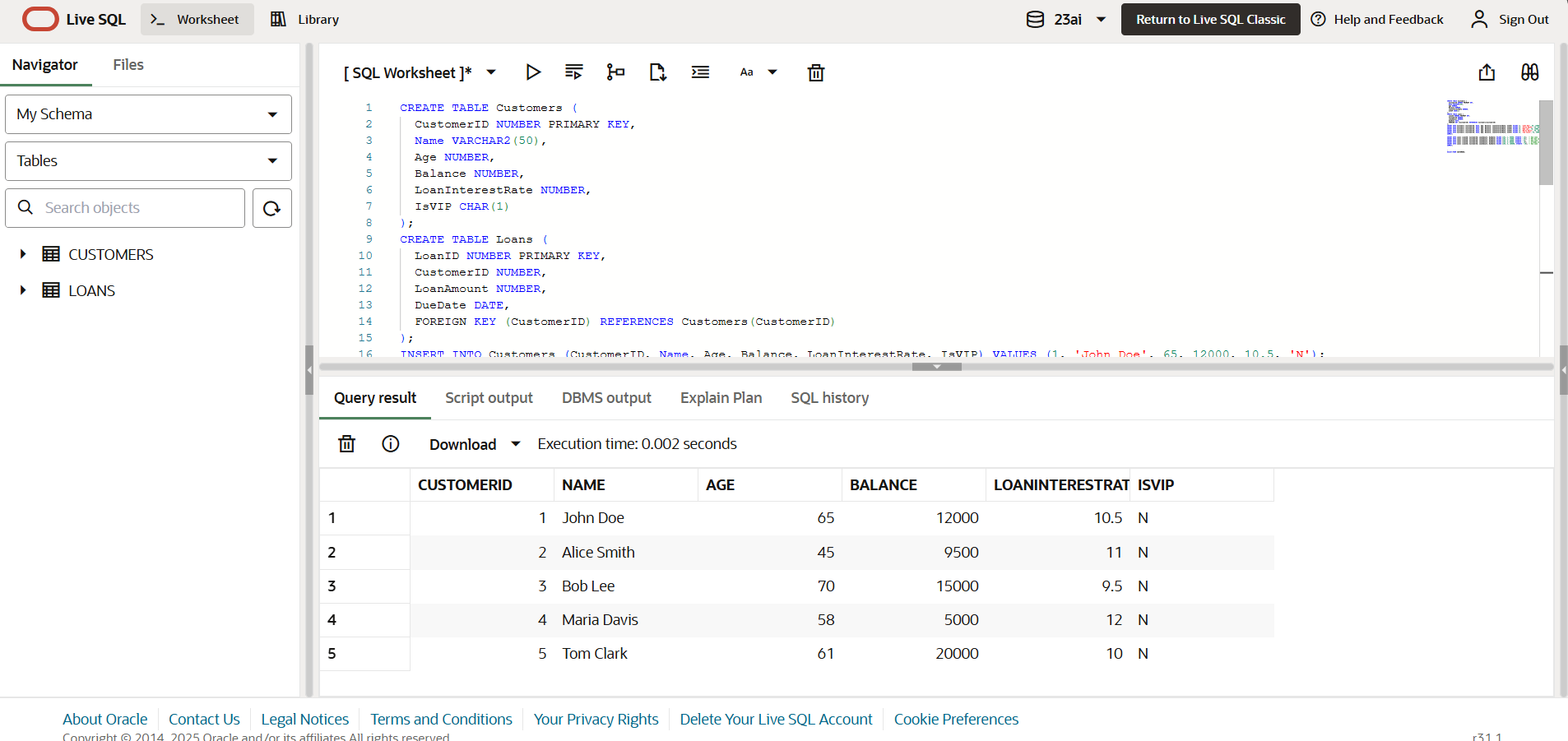
END IF;

END LOOP;

COMMIT;

END;

OUTPUT:



-- Scenario 2: A customer can be promoted to VIP status based on their balance.

BEGIN

FOR rec IN (SELECT CustomerID, Balance FROM Customers) LOOP

IF rec.Balance > 10000 THEN

UPDATE Customers

SET IsVIP = 'Y'

WHERE CustomerID = rec.CustomerID;

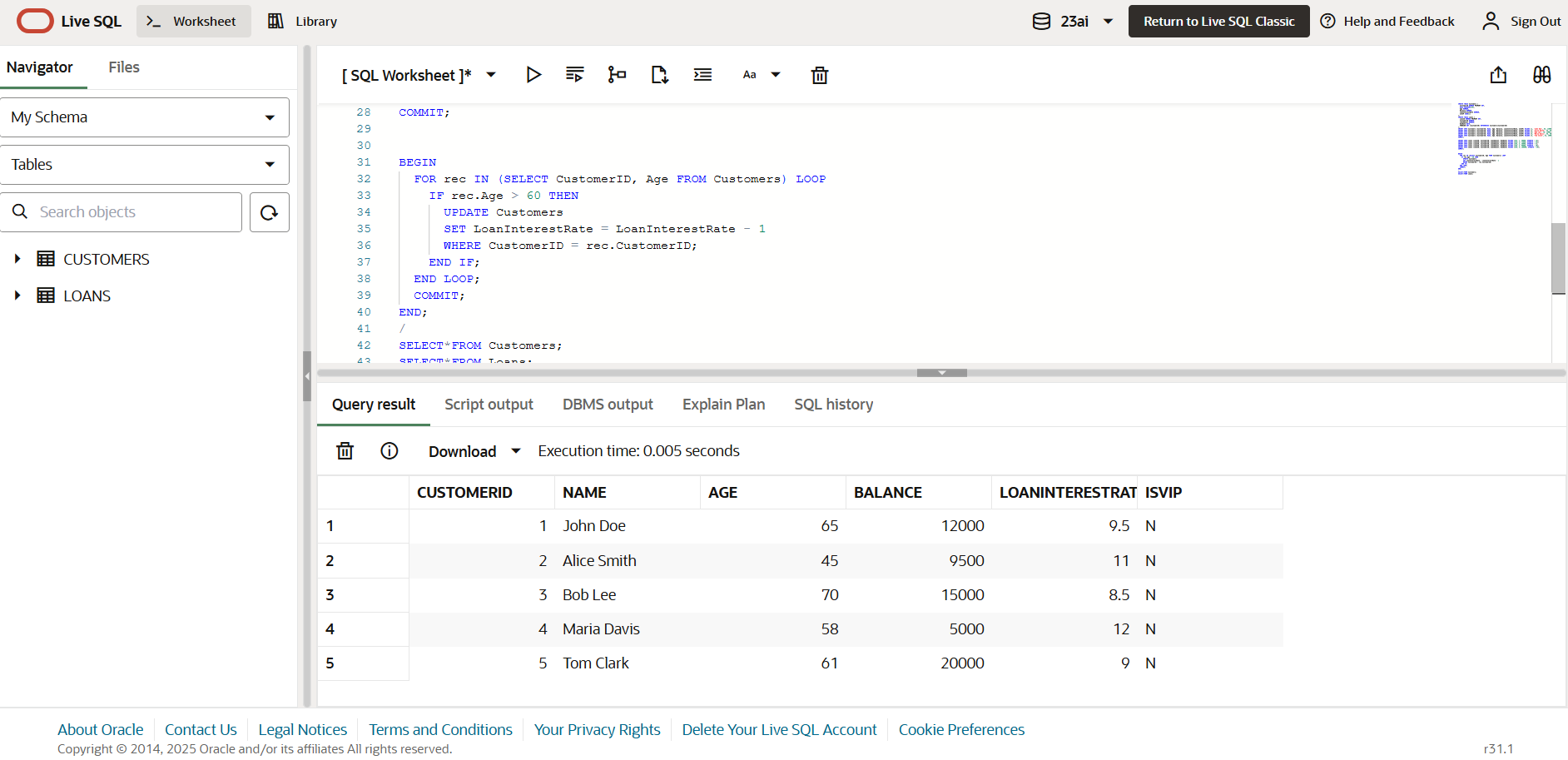
END IF;

END LOOP;

COMMIT;

END;

OUTPUT:



-- Scenario 3: The bank wants to send reminders to customers whose loans are due within the next 30 days.

BEGIN

FOR rec IN (

SELECT LoanID, CustomerID, DueDate

FROM Loans

WHERE DueDate BETWEEN SYSDATE AND SYSDATE + 30

) LOOP

DBMS\_OUTPUT.PUT\_LINE(

'Reminder: Loan ID ' || rec.LoanID ||

' for Customer ID ' || rec.CustomerID ||

' is due on ' || TO\_CHAR(rec.DueDate, 'DD-MON-YYYY')

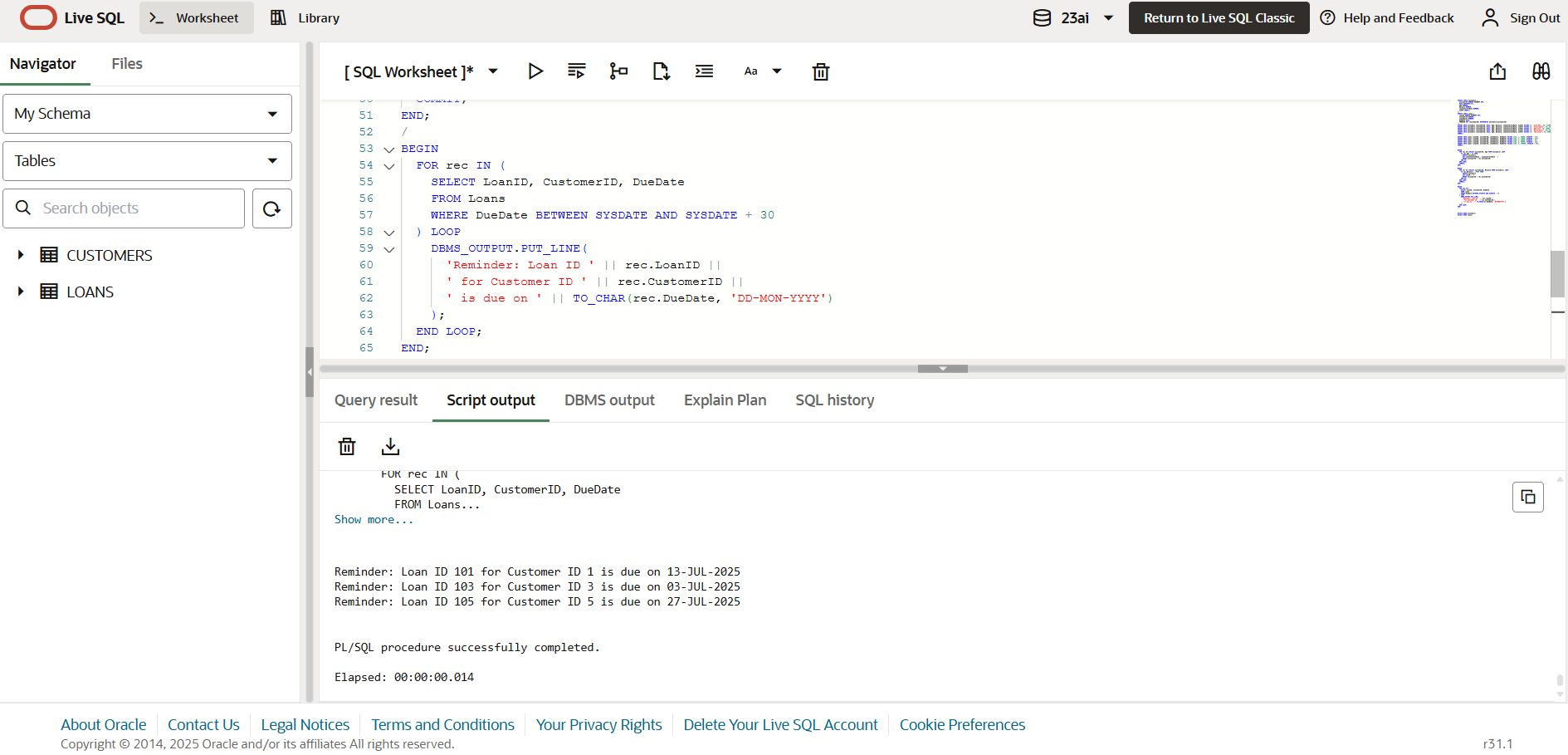
);

END LOOP;

END;

/

OUTPUT:



Exercise 3: Stored Procedures:

--Scenario 1: The bank needs to process monthly interest for all savings accounts.

Question: Write a stored procedure ProcessMonthlyInterest that calculates and updates the balance of all savings accounts by applying an interest rate of 1% to the current balance.

CREATE TABLE Employees (

EmployeeID NUMBER PRIMARY KEY,

Name VARCHAR2(50),

Department VARCHAR2(30),

Salary NUMBER

);

INSERT INTO Employees VALUES (1, 'Anita Sharma', 'IT', 50000);

INSERT INTO Employees VALUES (2, 'Ravi Kumar', 'HR', 40000);

INSERT INTO Employees VALUES (3, 'Priya Desai', 'IT', 60000);

INSERT INTO Employees VALUES (4, 'Sunil Mehta', 'Finance', 55000);

INSERT INTO Employees VALUES (5, 'Neha Singh', 'HR', 42000);

COMMIT;

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

FOR rec IN (SELECT CustomerID, Balance FROM Customers) LOOP

UPDATE Customers

SET Balance = Balance + (Balance \* 0.01)

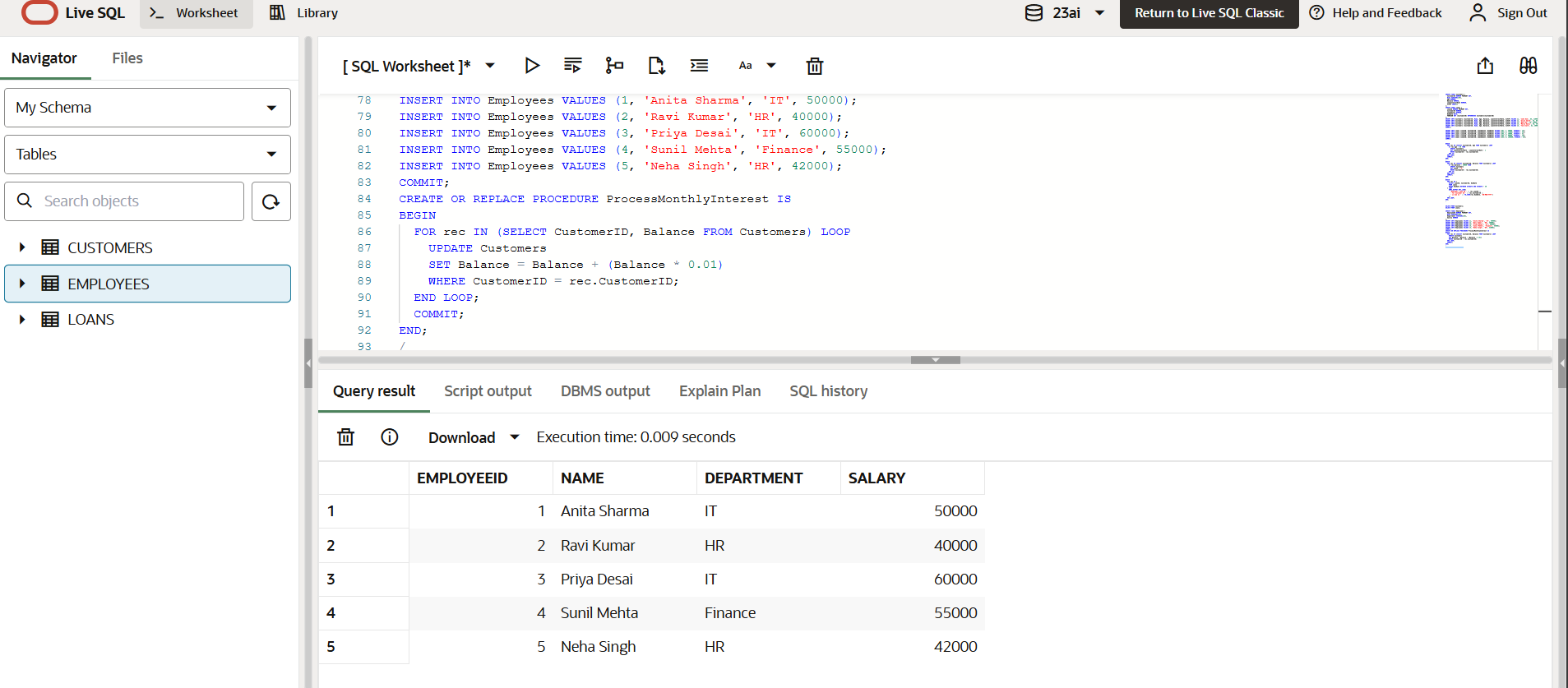
WHERE CustomerID = rec.CustomerID;

END LOOP;

COMMIT;

END;

OUTPUT:



--Scenario 2: The bank wants to implement a bonus scheme for employees based on their performance.

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_department IN VARCHAR2,

p\_bonus\_pct IN NUMBER

) IS

BEGIN

UPDATE Employees

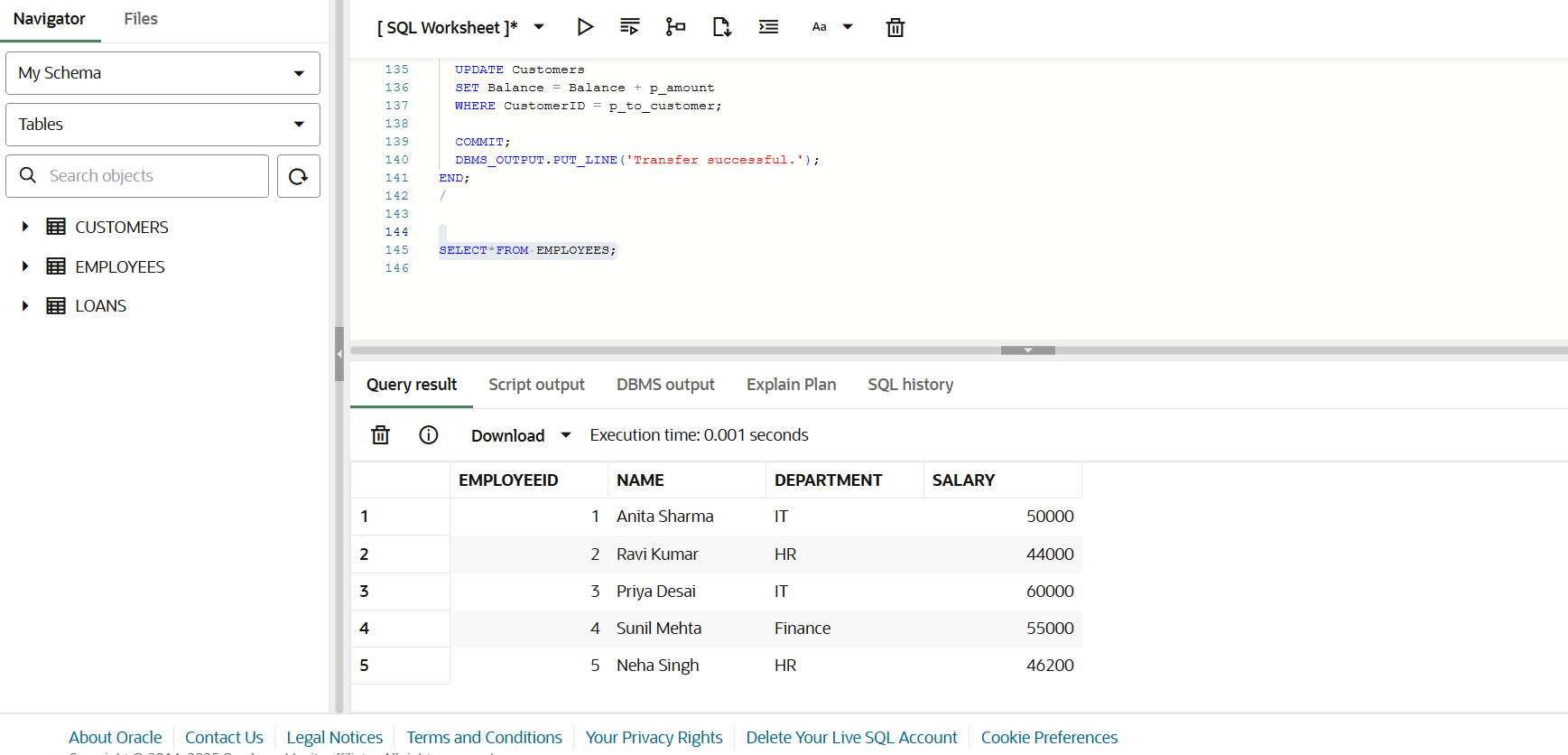
SET Salary = Salary + (Salary \* p\_bonus\_pct / 100)

WHERE Department = p\_department;

COMMIT;

END;

OUTPUT:



--Scenario 3: Customers should be able to transfer funds between their accounts.

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_department IN VARCHAR2,

p\_bonus\_pct IN NUMBER

) IS

BEGIN

UPDATE Employees

SET Salary = Salary + (Salary \* p\_bonus\_pct / 100)

WHERE Department = p\_department;

COMMIT;

END;

/

EXEC UpdateEmployeeBonus('HR', 10);

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_from\_customer IN NUMBER,

p\_to\_customer IN NUMBER,

p\_amount IN NUMBER

) IS

v\_balance NUMBER;

BEGIN

-- Step 1: Check source balance

SELECT Balance INTO v\_balance

FROM Customers

WHERE CustomerID = p\_from\_customer;

IF v\_balance < p\_amount THEN

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: Insufficient balance.');

RETURN;

END IF;

-- Step 2: Deduct from sender

UPDATE Customers

SET Balance = Balance - p\_amount

WHERE CustomerID = p\_from\_customer;

-- Step 3: Add to receiver

UPDATE Customers

SET Balance = Balance + p\_amount

WHERE CustomerID = p\_to\_customer;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Transfer successful.');

END;

/

OUTPUT: