**Exercise 1: Control Structures**

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

SET SERVEROUTPUT ON;

DECLARE

    v\_age NUMBER;

    v\_discount NUMBER := 1;

BEGIN

    FOR cust\_rec IN (SELECT CustomerID, DOB FROM Customers) LOOP

        v\_age := FLOOR(MONTHS\_BETWEEN(SYSDATE, cust\_rec.DOB)/12)

        IF v\_age > 60 THEN

            UPDATE Loans

            SET InterestRate = InterestRate - v\_discount

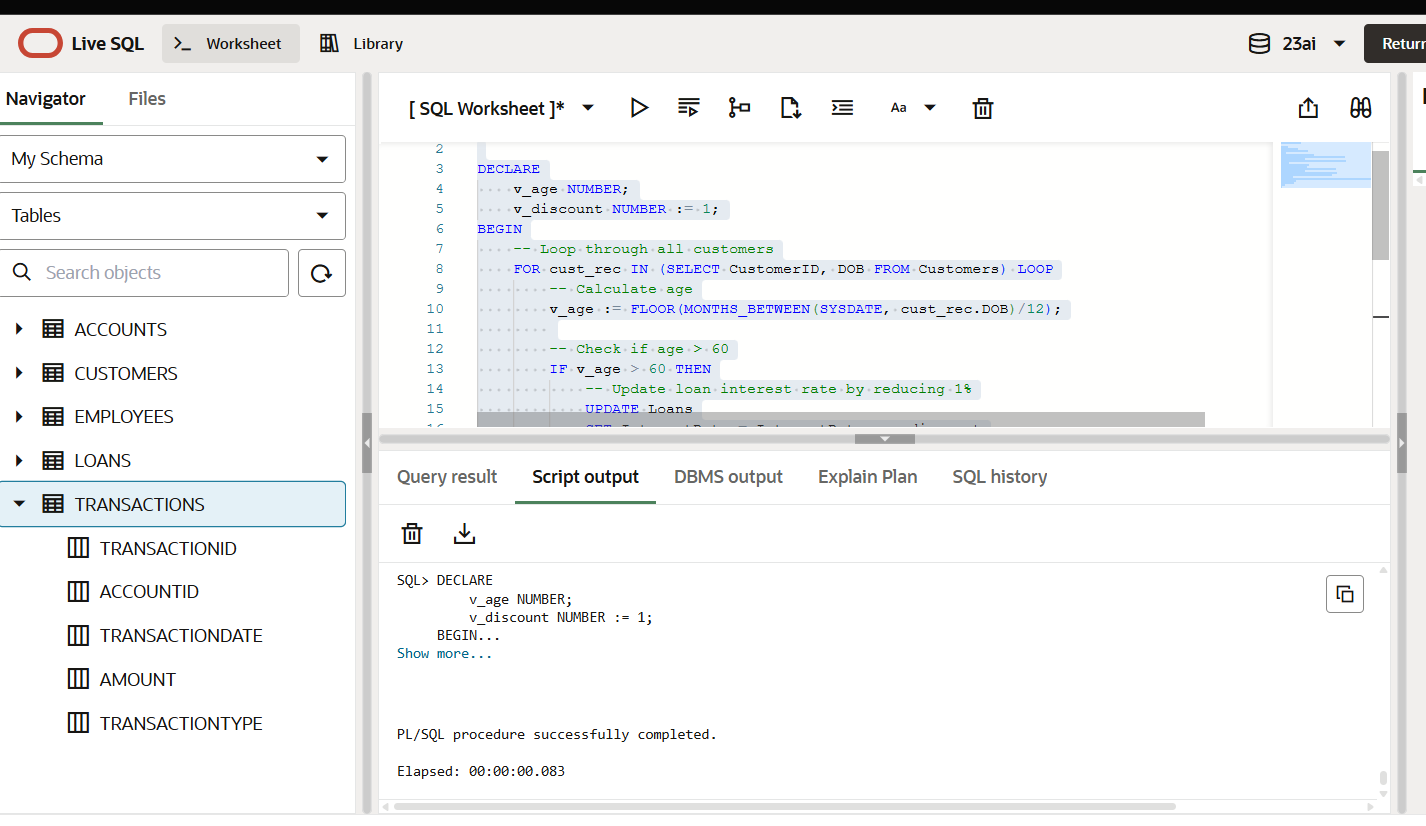
            WHERE CustomerID = cust\_rec.CustomerID;

            DBMS\_OUTPUT.PUT\_LINE('Applied discount for CustomerID: ' || cust\_rec.CustomerID);

        END IF;

    END LOOP;

END;

Output: 

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

ALTER TABLE Customers ADD (IsVIP VARCHAR2(5));

SET SERVEROUTPUT ON;

BEGIN

FOR cust\_rec IN (SELECT CustomerID, Balance FROM Customers) LOOP

IF cust\_rec.Balance > 10000 THEN

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = cust\_rec.CustomerID;

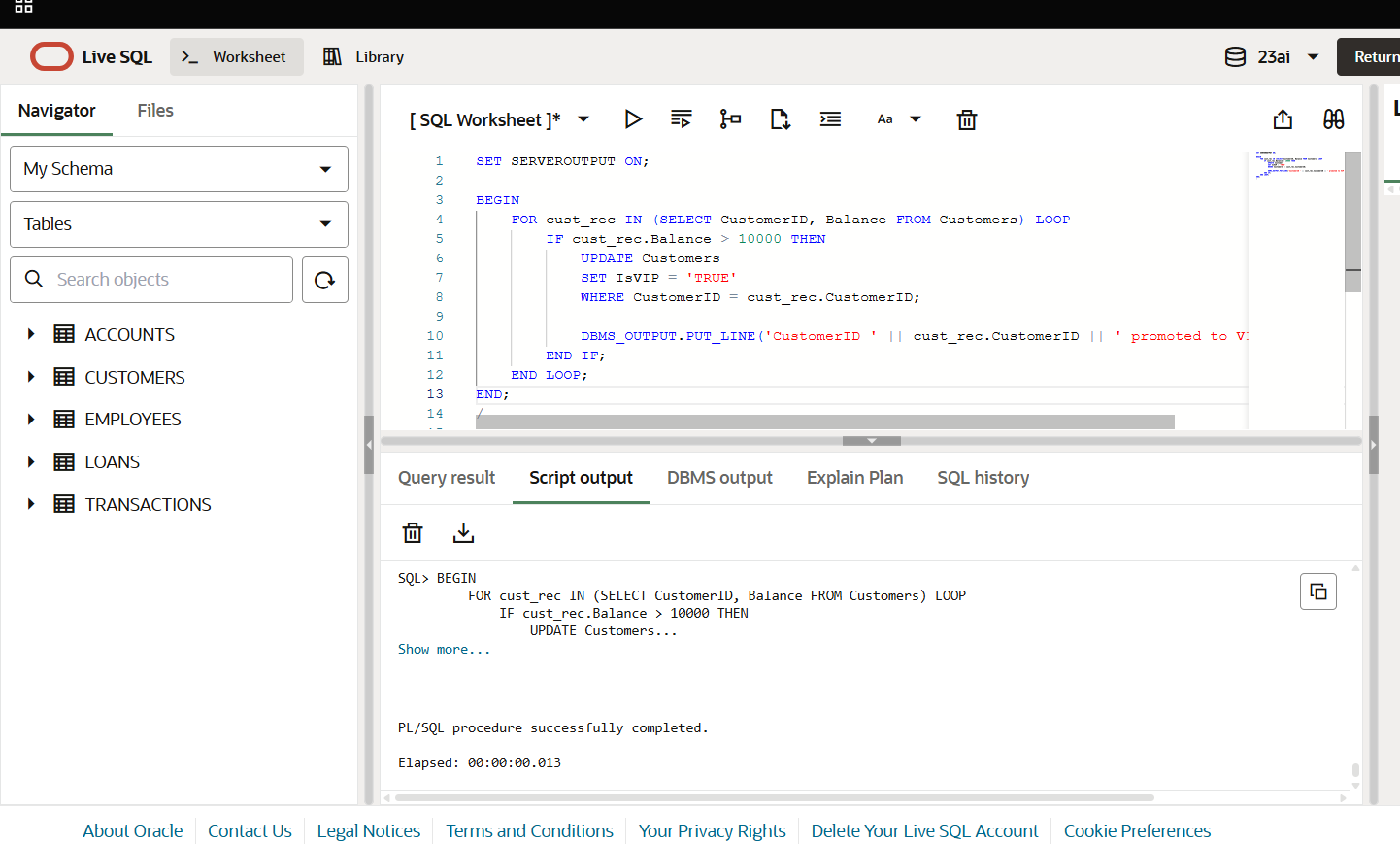
DBMS\_OUTPUT.PUT\_LINE('CustomerID ' || cust\_rec.CustomerID || ' promoted to VIP.');

END IF;

END LOOP;

END;

Output:



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

SET SERVEROUTPUT ON;

BEGIN

FOR loan\_rec IN (

SELECT l.LoanID, l.CustomerID, l.EndDate, c.Name

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID

WHERE l.EndDate BETWEEN SYSDATE AND SYSDATE + 30

) LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: LoanID ' || loan\_rec.LoanID ||

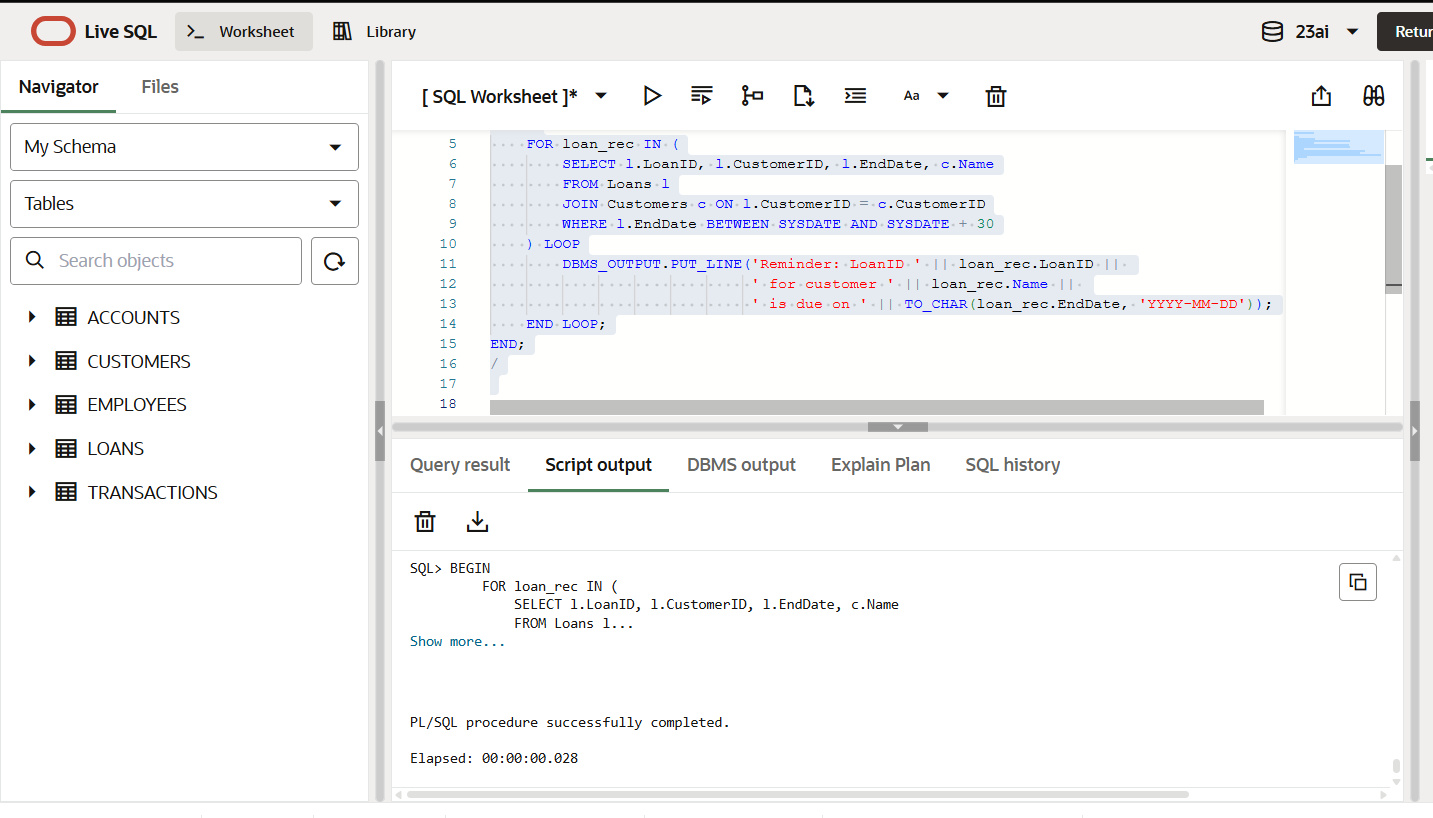
' for customer ' || loan\_rec.Name ||

' is due on ' || TO\_CHAR(loan\_rec.EndDate, 'YYYY-MM-DD'));

END LOOP;

END;

Output:



**Exercise 3: Stored Procedures**

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

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest

IS

v\_interest\_rate NUMBER := 0.01; -- 1% interest

BEGIN

UPDATE Accounts

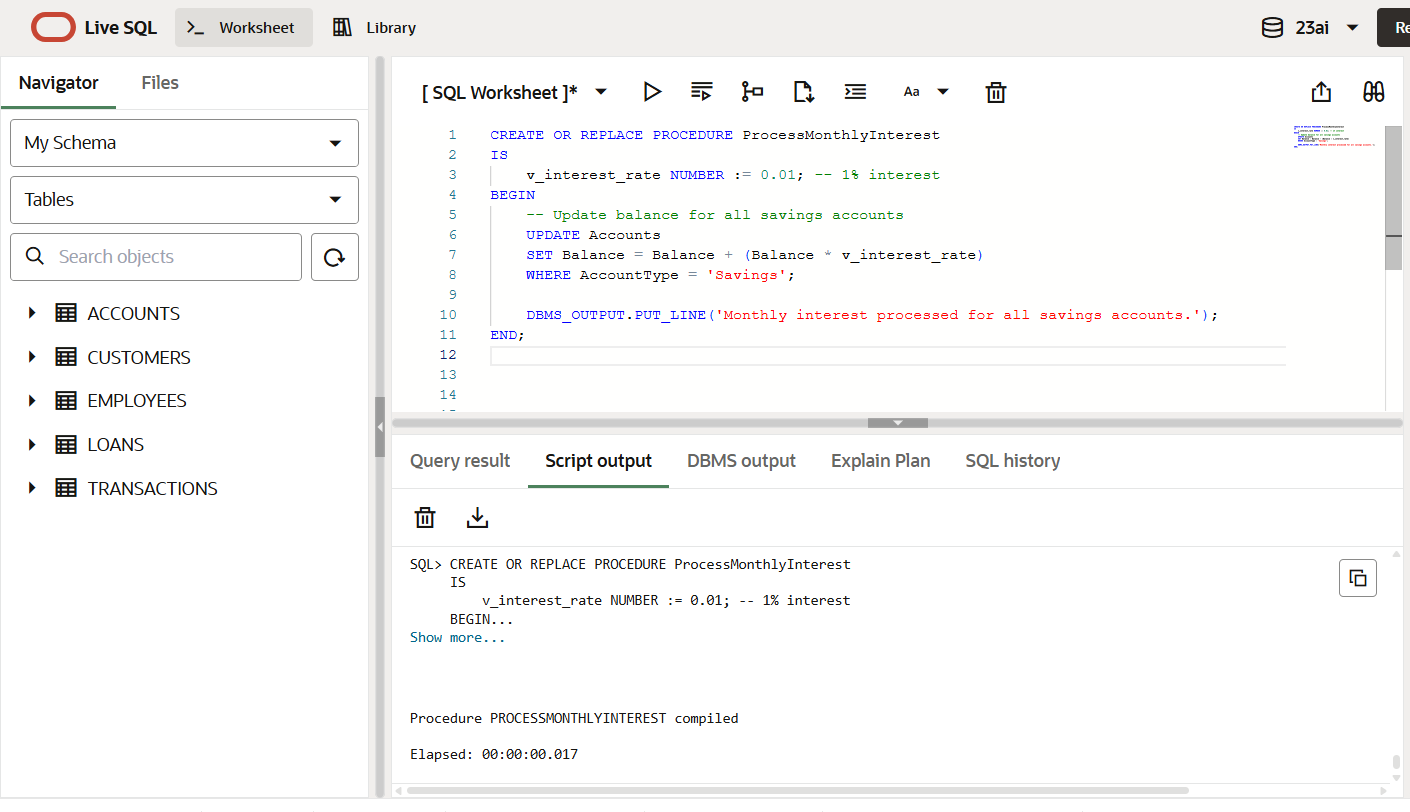
SET Balance = Balance + (Balance \* v\_interest\_rate)

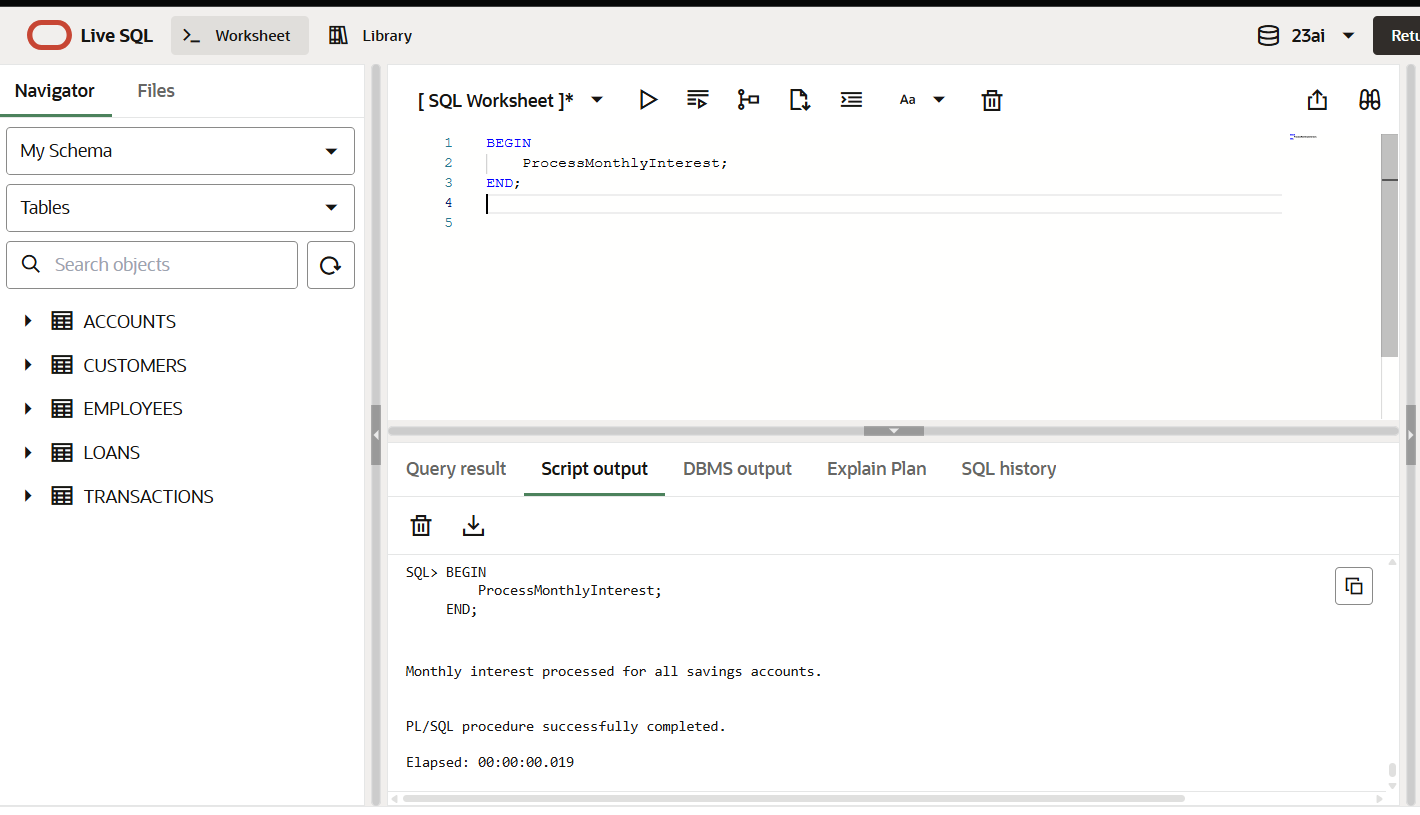
WHERE AccountType = 'Savings';

DBMS\_OUTPUT.PUT\_LINE('Monthly interest processed for all savings accounts.');

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

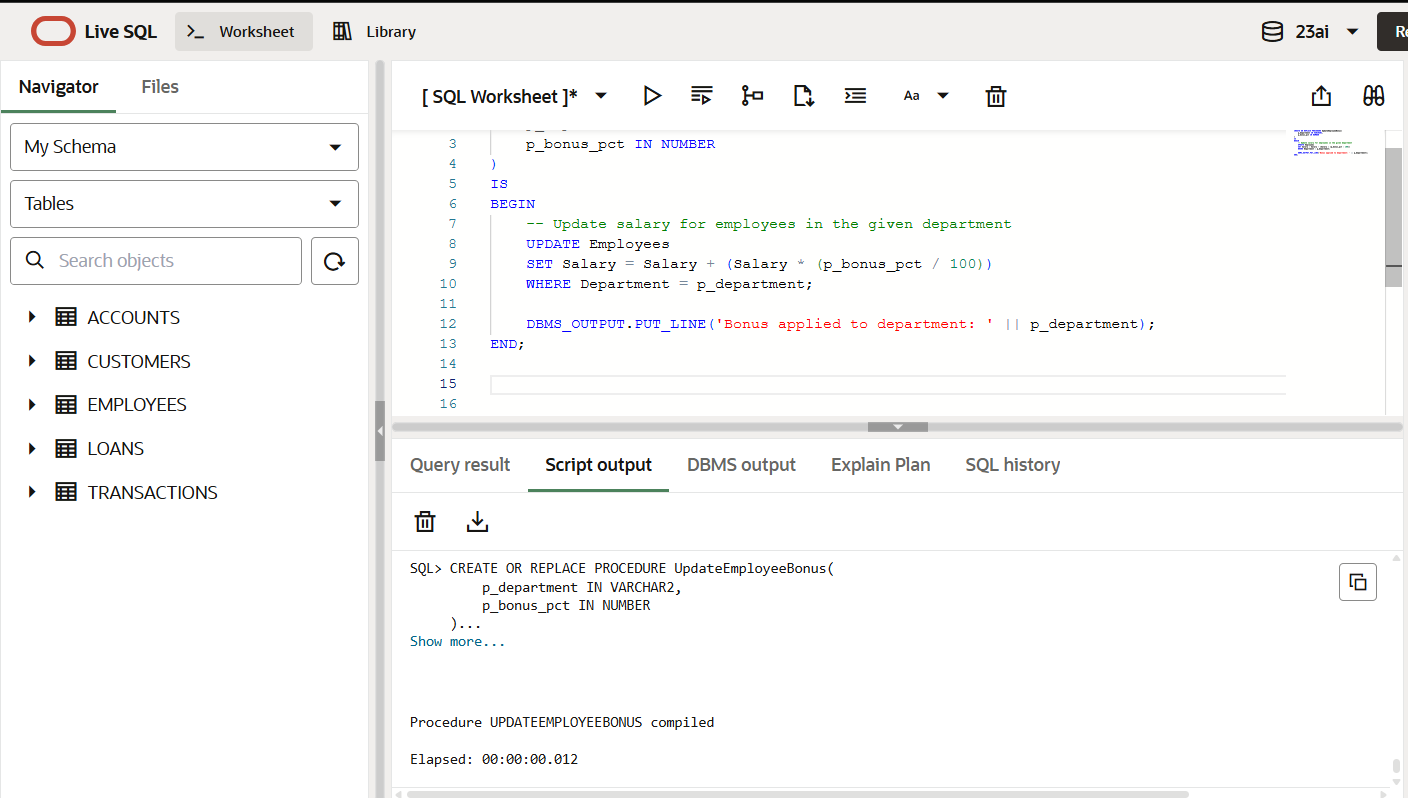
SET Salary = Salary + (Salary \* (p\_bonus\_pct / 100))

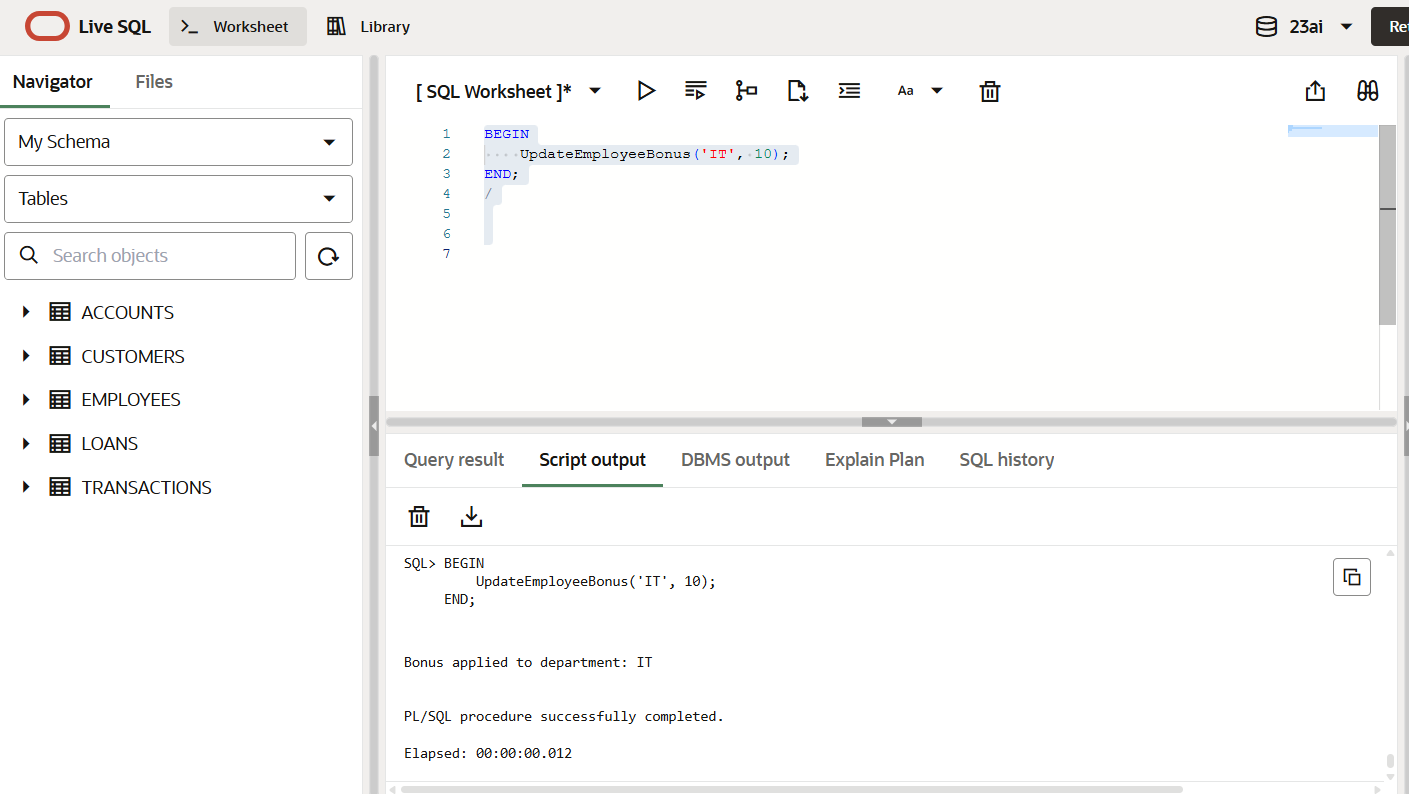
WHERE Department = p\_department;

DBMS\_OUTPUT.PUT\_LINE('Bonus applied to department: ' || p\_department);

END;

Output:





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

CREATE OR REPLACE PROCEDURE TransferFunds(

    p\_source\_account IN NUMBER,

    p\_target\_account IN NUMBER,

    p\_amount IN NUMBER

)

IS

    v\_source\_balance NUMBER;

BEGIN

    SELECT Balance INTO v\_source\_balance

    FROM Accounts

    WHERE AccountID = p\_source\_account;

    IF v\_source\_balance >= p\_amount THEN

        UPDATE Accounts

        SET Balance = Balance - p\_amount

        WHERE AccountID = p\_source\_account;

        UPDATE Accounts

        SET Balance = Balance + p\_amount

        WHERE AccountID = p\_target\_account;

        DBMS\_OUTPUT.PUT\_LINE('Transfer of ' || p\_amount || ' completed from AccountID ' || p\_source\_account || ' to AccountID ' || p\_target\_account);

    ELSE

        DBMS\_OUTPUT.PUT\_LINE('Insufficient balance in AccountID ' || p\_source\_account);

    END IF;

END;

Output:

