**Exercise – 1:**

**Senario-1:**

SET SERVEROUTPUT ON;

DECLARE

    CURSOR cur\_loans IS

        SELECT l.LoanID, l.InterestRate, c.DOB

        FROM Customers c

        JOIN Loans l ON c.CustomerID = l.CustomerID;

    v\_age NUMBER;

BEGIN

    FOR rec IN cur\_loans LOOP

        v\_age := TRUNC(MONTHS\_BETWEEN(SYSDATE, rec.DOB) / 12);

        IF v\_age > 60 THEN

            UPDATE Loans

            SET InterestRate = rec.InterestRate - 1

            WHERE LoanID = rec.LoanID;

            DBMS\_OUTPUT.PUT\_LINE('Customer with id: ' || rec.LoanID || ' interest reduced. Age: ' || v\_age);

        END IF;

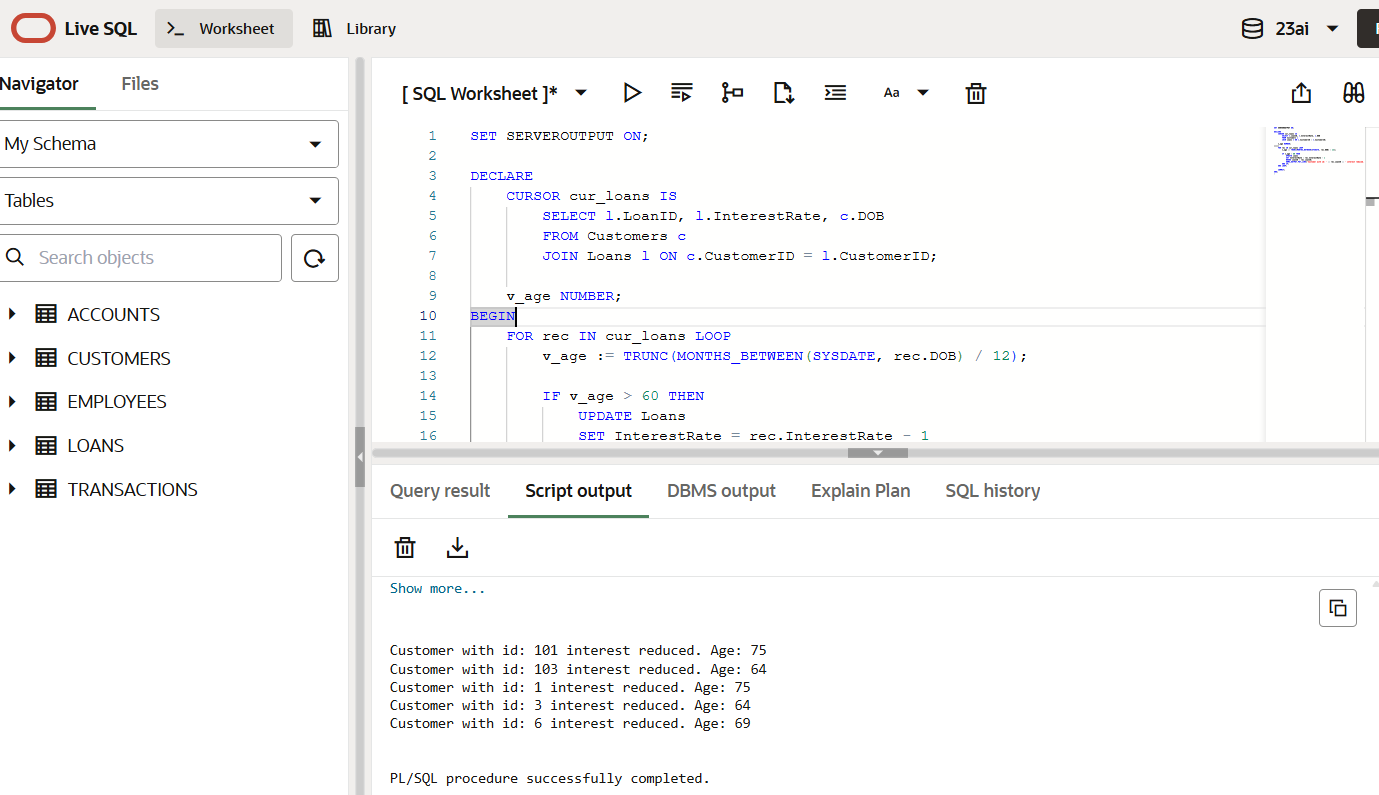
    END LOOP;

    COMMIT;

END;

/

**Output:**



**Senario-2:**

As there is no column stating VIP status we can add it to the table.

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

SET SERVEROUTPUT ON;

BEGIN

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

IF rec.Balance > 10000 THEN

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = rec.CustomerID;

DBMS\_OUTPUT.PUT\_LINE('Customer ' || rec.CustomerID || ' marked as VIP.');

ELSE

UPDATE Customers

SET IsVIP = 'FALSE'

WHERE CustomerID = rec.CustomerID;

DBMS\_OUTPUT.PUT\_LINE('Customer ' || rec.CustomerID || ' is not a VIP.');

END IF;

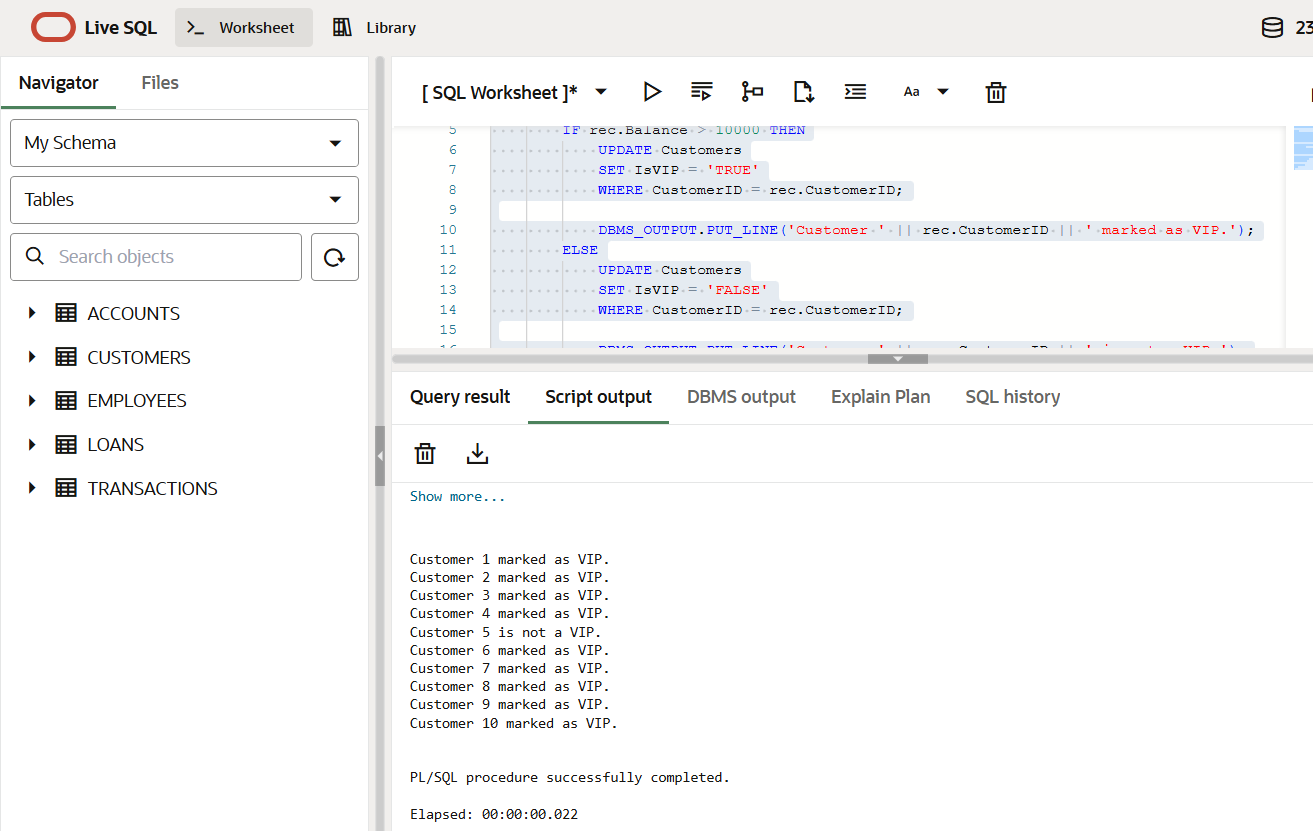
END LOOP;

COMMIT;

END;

/

**Output:**



**Scenario-3:**

SET SERVEROUTPUT ON;

BEGIN

FOR rec IN (

SELECT l.LoanID, 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: Loan ID ' || rec.LoanID ||

' for customer ' || rec.Name ||

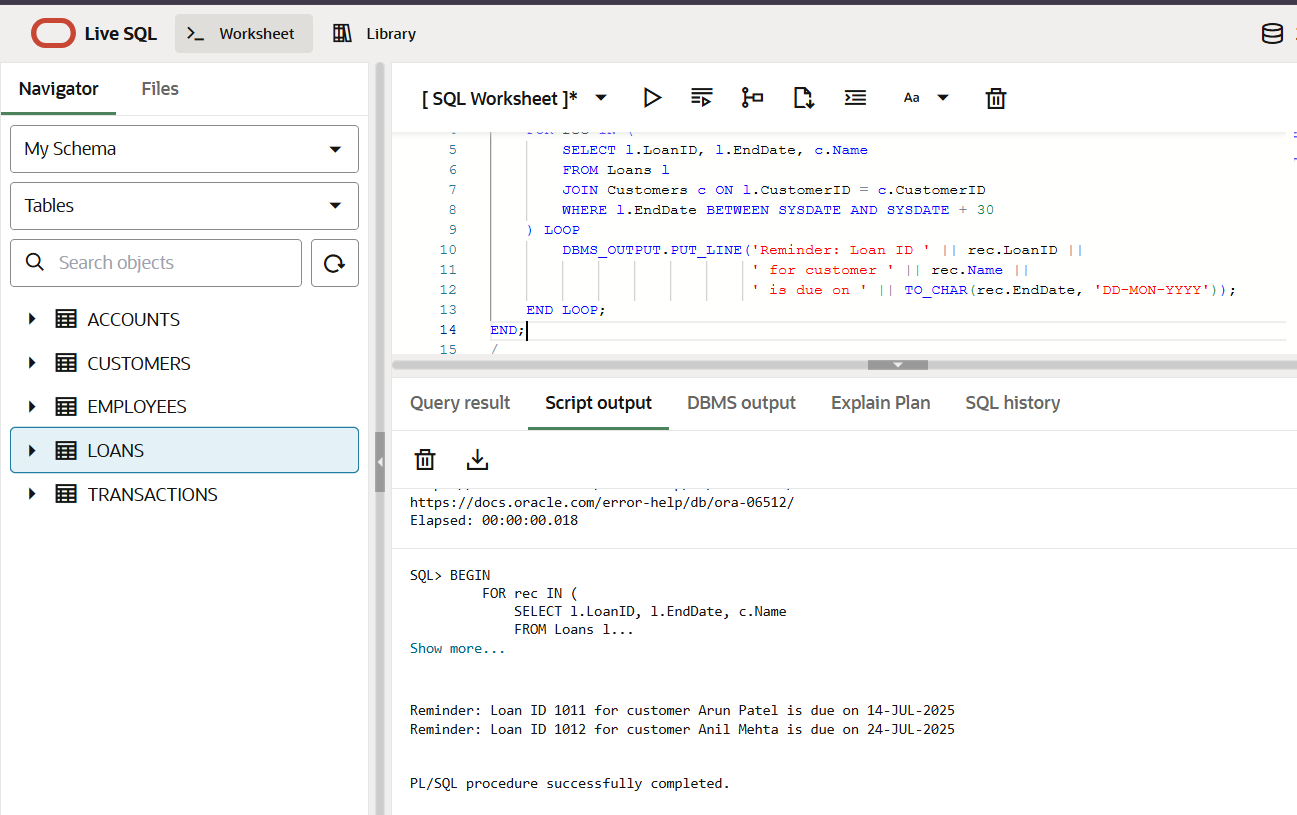
' is due on ' || TO\_CHAR(rec.EndDate, 'DD-MON-YYYY'));

END LOOP;

END;

/

**Output:**



**Exercise 2: Error Handling**

**Scenario-01:** **Safe Fund Transfer Between Accounts**

CREATE OR REPLACE PROCEDURE SafeTransferFunds (

    p\_from\_account IN NUMBER,

    p\_to\_account   IN NUMBER,

    p\_amount       IN NUMBER

) IS

    v\_balance NUMBER;

BEGIN

    SELECT Balance INTO v\_balance FROM Accounts

    WHERE AccountID = p\_from\_account;

    IF v\_balance < p\_amount THEN

        RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient money');

    END IF;

    UPDATE Accounts

    SET Balance = Balance - p\_amount

    WHERE AccountID = p\_from\_account;

    UPDATE Accounts

    SET Balance = Balance + p\_amount

    WHERE AccountID = p\_to\_account;

    DBMS\_OUTPUT.PUT\_LINE('₹' || p\_amount || ' successfully transferred from ' ||

                         p\_from\_account || ' to ' || p\_to\_account);

    COMMIT;

EXCEPTION

    WHEN NO\_DATA\_FOUND THEN

        ROLLBACK;

        DBMS\_OUTPUT.PUT\_LINE('Error: One or both accounts not found.');

    WHEN OTHERS THEN

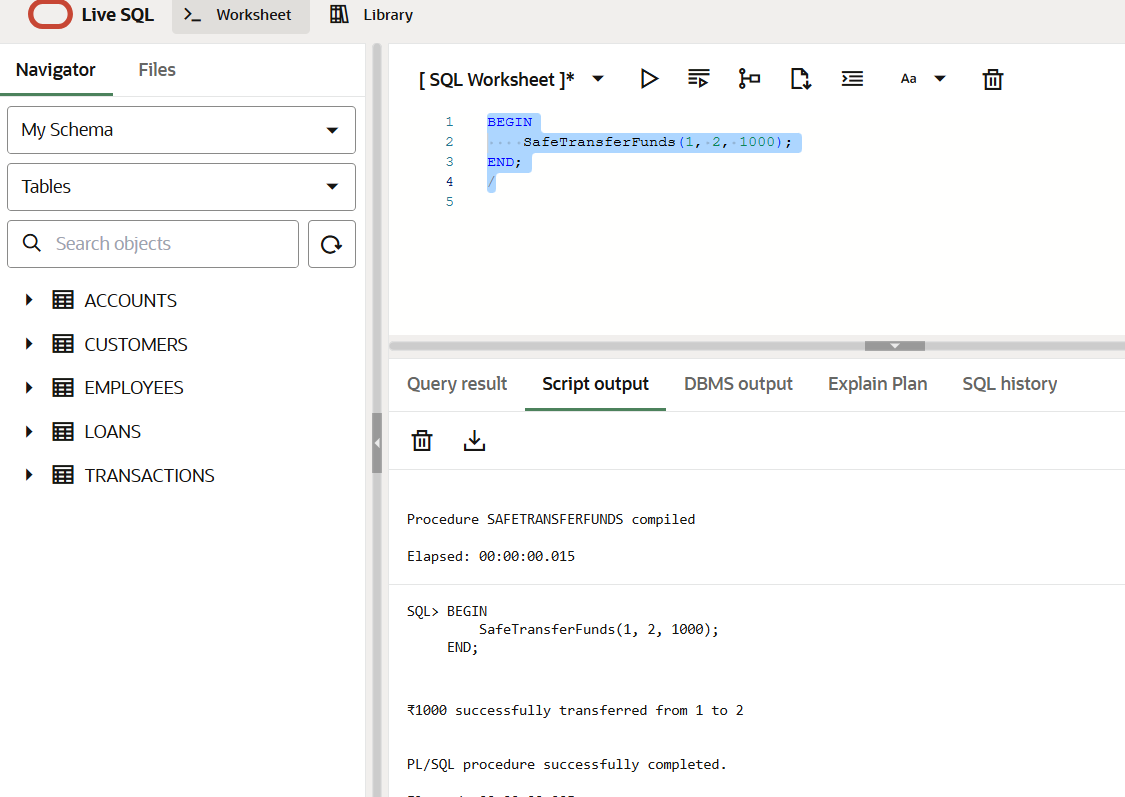
        ROLLBACK;

        DBMS\_OUTPUT.PUT\_LINE('Transfer failed: ' || SQLERRM);

END;

/

**Output:**

****

**Scenario-02: Update Employee Salary with Exception Handling**

CREATE OR REPLACE PROCEDURE UpdateSalary (

    p\_emp\_id        IN NUMBER,

    p\_increase\_pct  IN NUMBER

) IS

BEGIN

    UPDATE Employees

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

    WHERE EmployeeID = p\_emp\_id;

    IF SQL%ROWCOUNT = 0 THEN

        RAISE\_APPLICATION\_ERROR(-20002, 'Employee ID not found.');

    END IF;

    DBMS\_OUTPUT.PUT\_LINE('Salary updated for Employee ID ' || p\_emp\_id);

    COMMIT;

EXCEPTION

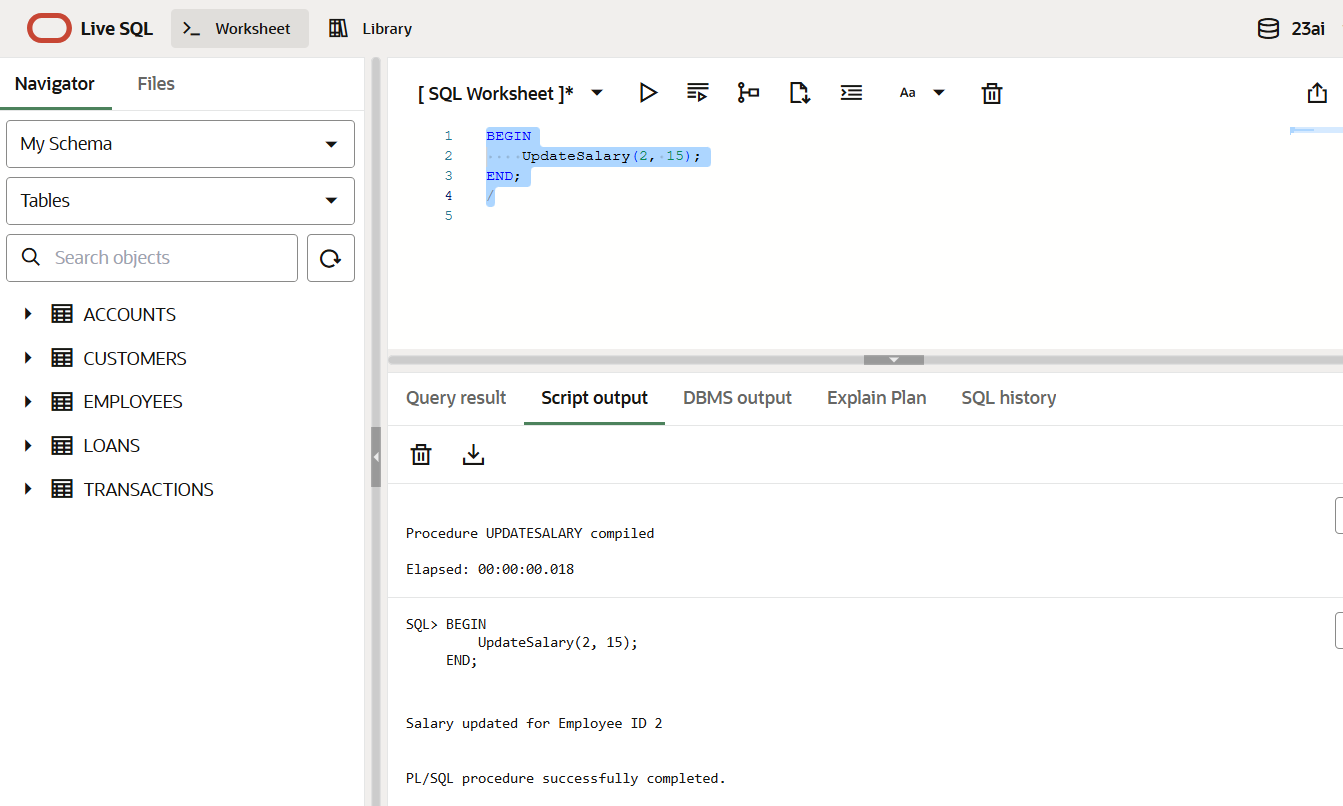
    WHEN OTHERS THEN

        ROLLBACK;

        DBMS\_OUTPUT.PUT\_LINE('Update failed: ' || SQLERRM);

END;

/

**Output:  
**

**Scenario-03:** **Add New Customer with Duplicate Checking while inserting**

CREATE OR REPLACE PROCEDURE AddNewCustomer (

    p\_customer\_id   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\_customer\_id, p\_name, p\_dob, p\_balance, SYSDATE);

    DBMS\_OUTPUT.PUT\_LINE('Customer ' || p\_name || ' added successfully.');

    COMMIT;

EXCEPTION

    WHEN DUP\_VAL\_ON\_INDEX THEN

        ROLLBACK;

        DBMS\_OUTPUT.PUT\_LINE('Error: Customer ID ' || p\_customer\_id || ' already exists.');

    WHEN OTHERS THEN

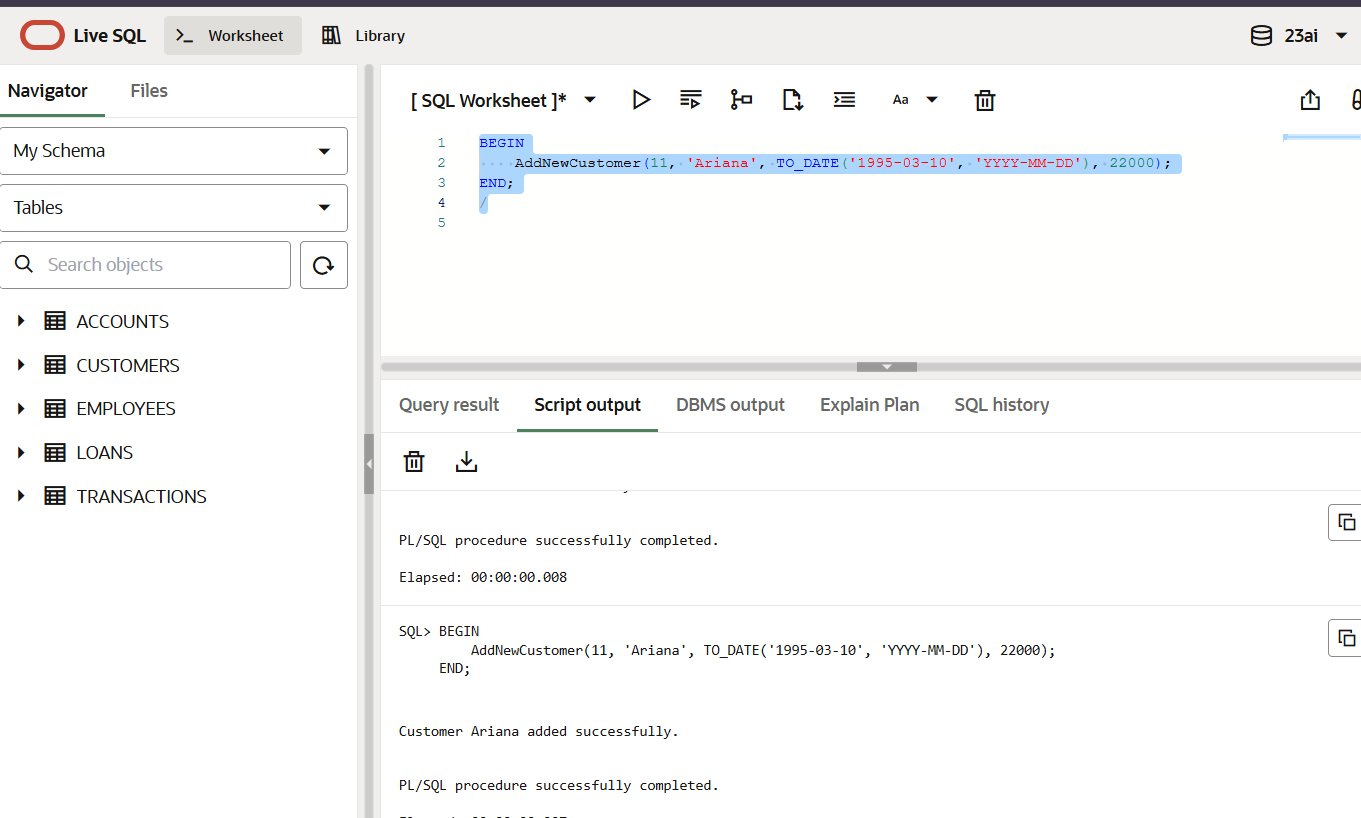
        ROLLBACK;

        DBMS\_OUTPUT.PUT\_LINE('Insertion failed: ' || SQLERRM);

END;

/

**Output:**

****

**Exercise-03: STORED PROCEDURES**

**Scenario-1: Process Monthly Interest for Savings Accounts**

**Procedure Creation**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

    FOR acc IN (

        SELECT AccountID, Balance

        FROM Accounts

        WHERE AccountType = 'Savings'

    ) LOOP

        UPDATE Accounts

        SET Balance = Balance + (acc.Balance \* 0.01)

        WHERE AccountID = acc.AccountID;

        DBMS\_OUTPUT.PUT\_LINE(' 1% Interest added to Account ID ' || acc.AccountID);

    END LOOP;

    COMMIT;

END;

/

**Excecution:**

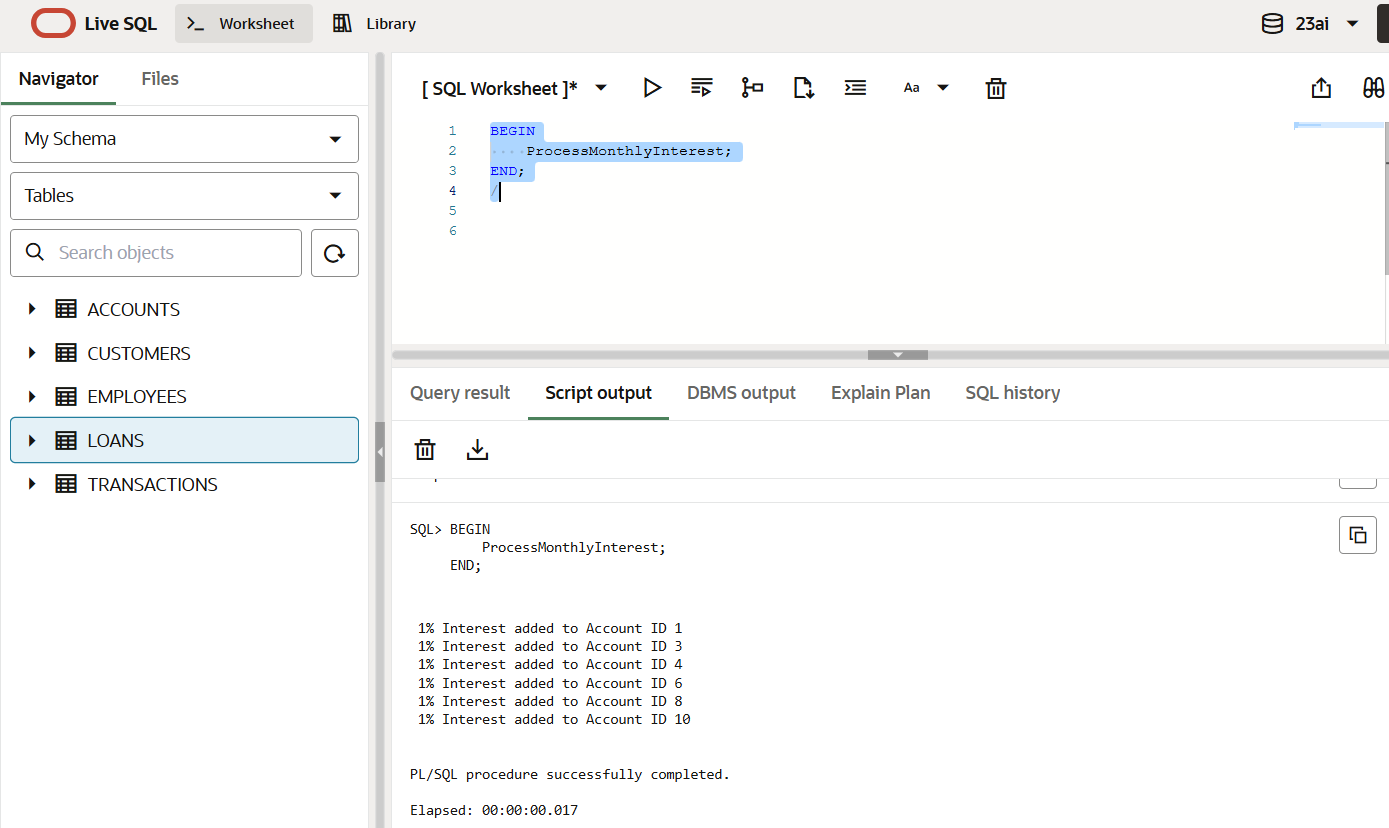
BEGIN

ProcessMonthlyInterest;

END;

/

**OUTPUT:**

****

**Scenario-02: Update Employee Bonus based on Department**CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

dept\_name IN VARCHAR2,

bonus\_percent IN NUMBER

) IS

BEGIN

UPDATE Employees

SET Salary = Salary + (Salary \* bonus\_percent / 100)

WHERE Department = dept\_name;

DBMS\_OUTPUT.PUT\_LINE('Bonus applied based on the department name : ' || dept\_name);

COMMIT;

END;

/

**Exceute:**

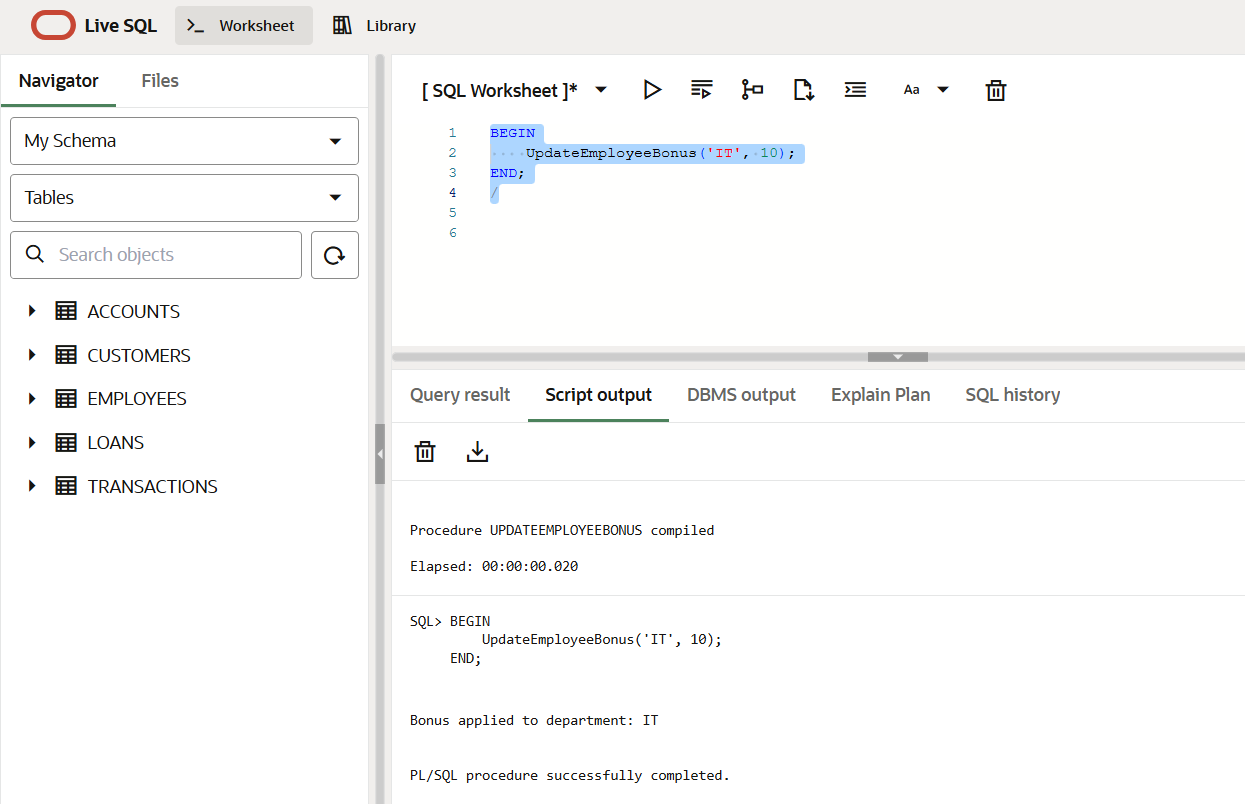
*BEGIN*

*UpdateEmployeeBonus('IT', 10);*

*END;*

*/*

**OUTPUT:**



**Scenario-03: Transfering Funds Between Accounts**

CREATE OR REPLACE PROCEDURE TransferFunds (

    p\_from\_account IN NUMBER,

    p\_to\_account   IN NUMBER,

    p\_amount       IN NUMBER

) IS

    v\_balance NUMBER;

BEGIN

    SELECT Balance INTO v\_balance FROM Accounts

    WHERE AccountID = p\_from\_account;

    IF v\_balance < p\_amount THEN

        RAISE\_APPLICATION\_ERROR(-20001, 'Balance insufficient.');

    END IF;

    UPDATE Accounts

    SET Balance = Balance - p\_amount

    WHERE AccountID = p\_from\_account;

    UPDATE Accounts

    SET Balance = Balance + p\_amount

    WHERE AccountID = p\_to\_account;

    DBMS\_OUTPUT.PUT\_LINE('₹' || p\_amount || ' transferred from Account ' ||

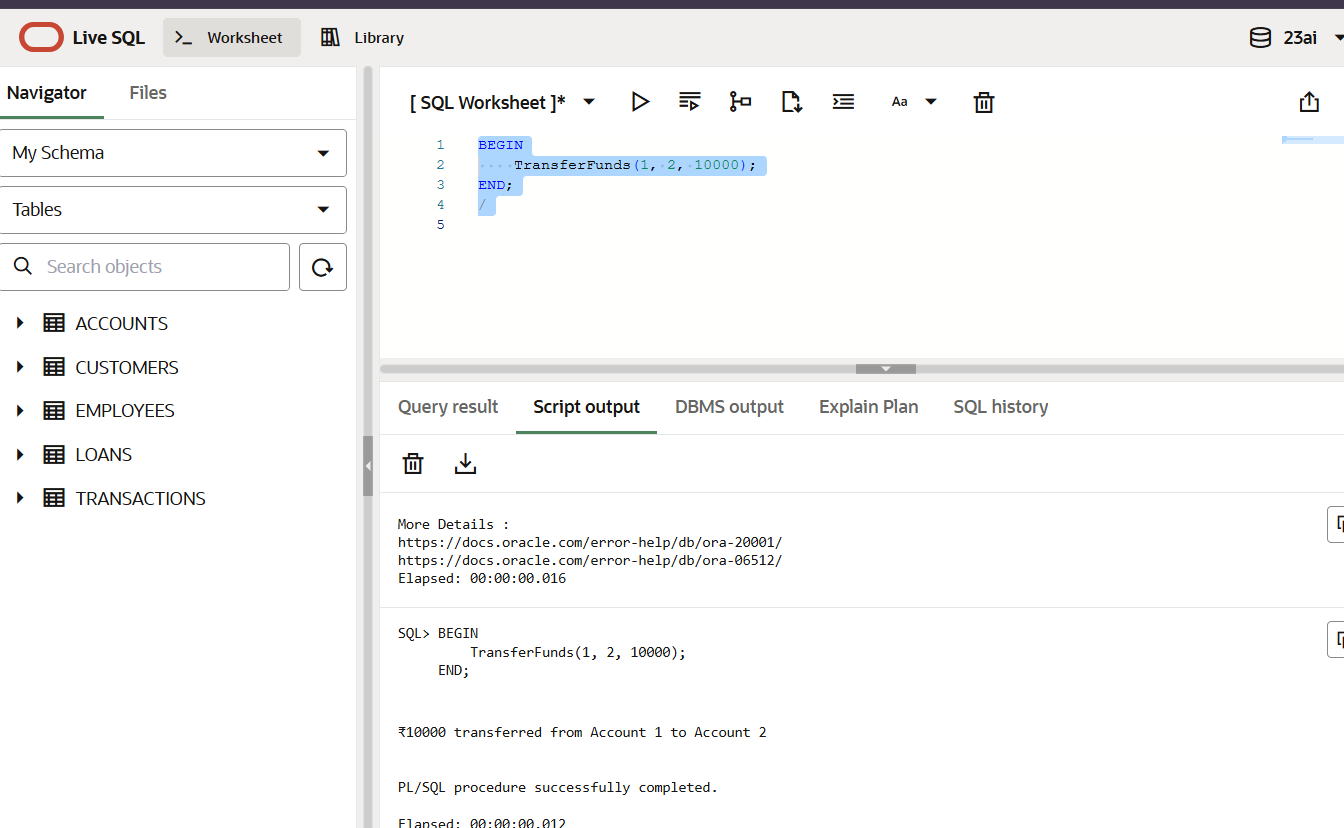
                         p\_from\_account || ' to Account ' || p\_to\_account);

    COMMIT;

END;

/

**Excecute:**

****