**EXERCISE 1 : CONTROL STRUCTURES**

**SCENARIO 1:**

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

**CODE:**

BEGIN

    FOR cust\_rec IN (

        SELECT c.CustomerID, l.LoanID

        FROM Customers c

        JOIN Loans l ON c.CustomerID = l.CustomerID

        WHERE MONTHS\_BETWEEN(SYSDATE, c.DOB) / 12 > 60

    ) LOOP

        UPDATE Loans

        SET InterestRate = InterestRate - 1

        WHERE LoanID = cust\_rec.LoanID;

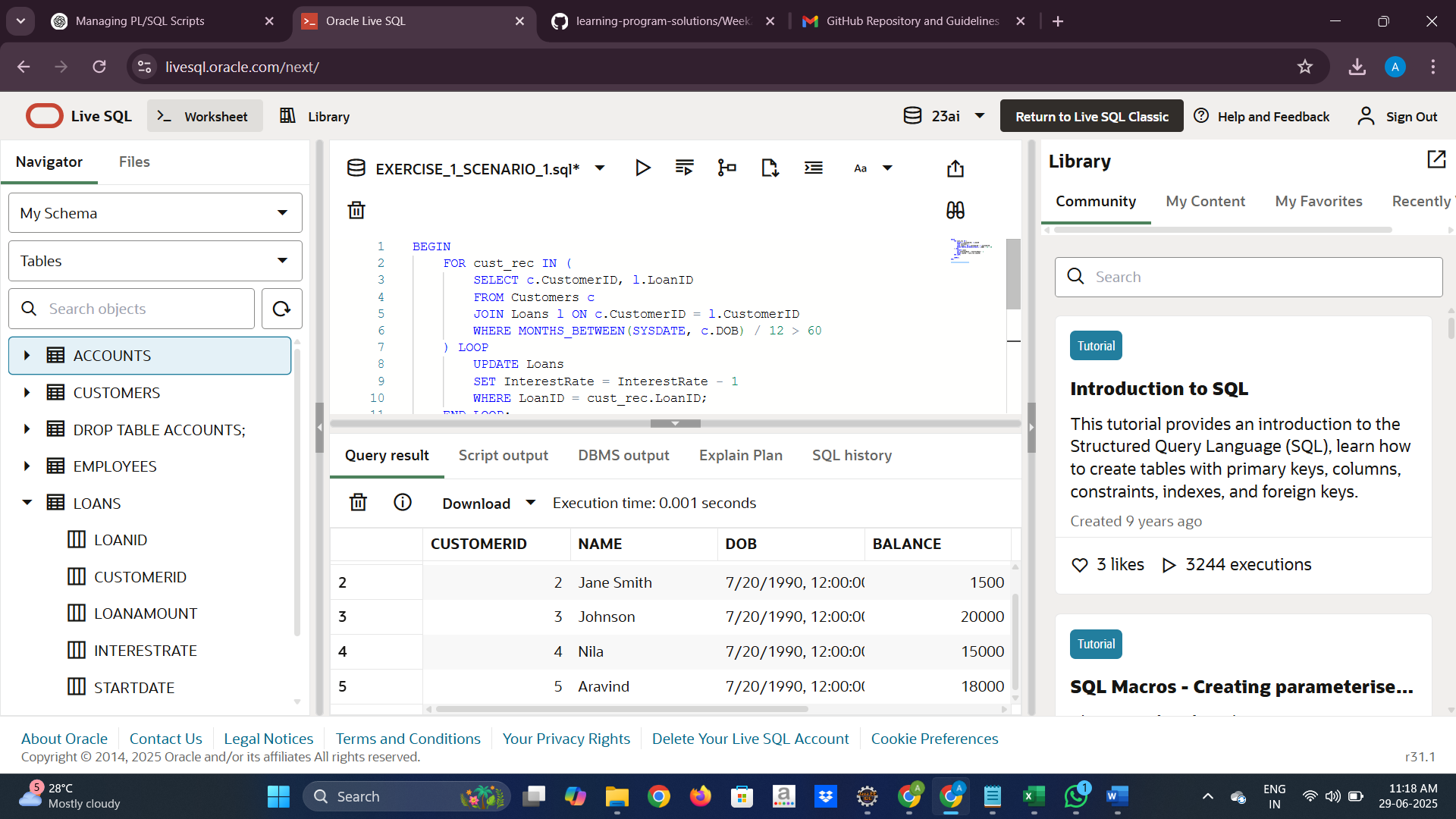
    END LOOP;

COMMIT;

END;

SELECT \* FROM Customers;

**OUTPUT:**



**SCENARIO 2:**

A customer can be promoted to VIP status based on their balance.

**CODE:**

BEGIN

    FOR cust\_rec IN (

        SELECT CustomerID

        FROM Customers

        WHERE Balance > 10000

    ) LOOP

        UPDATE Customers

        SET IsVIP = 'Y'

        WHERE CustomerID = cust\_rec.CustomerID;

    END LOOP;

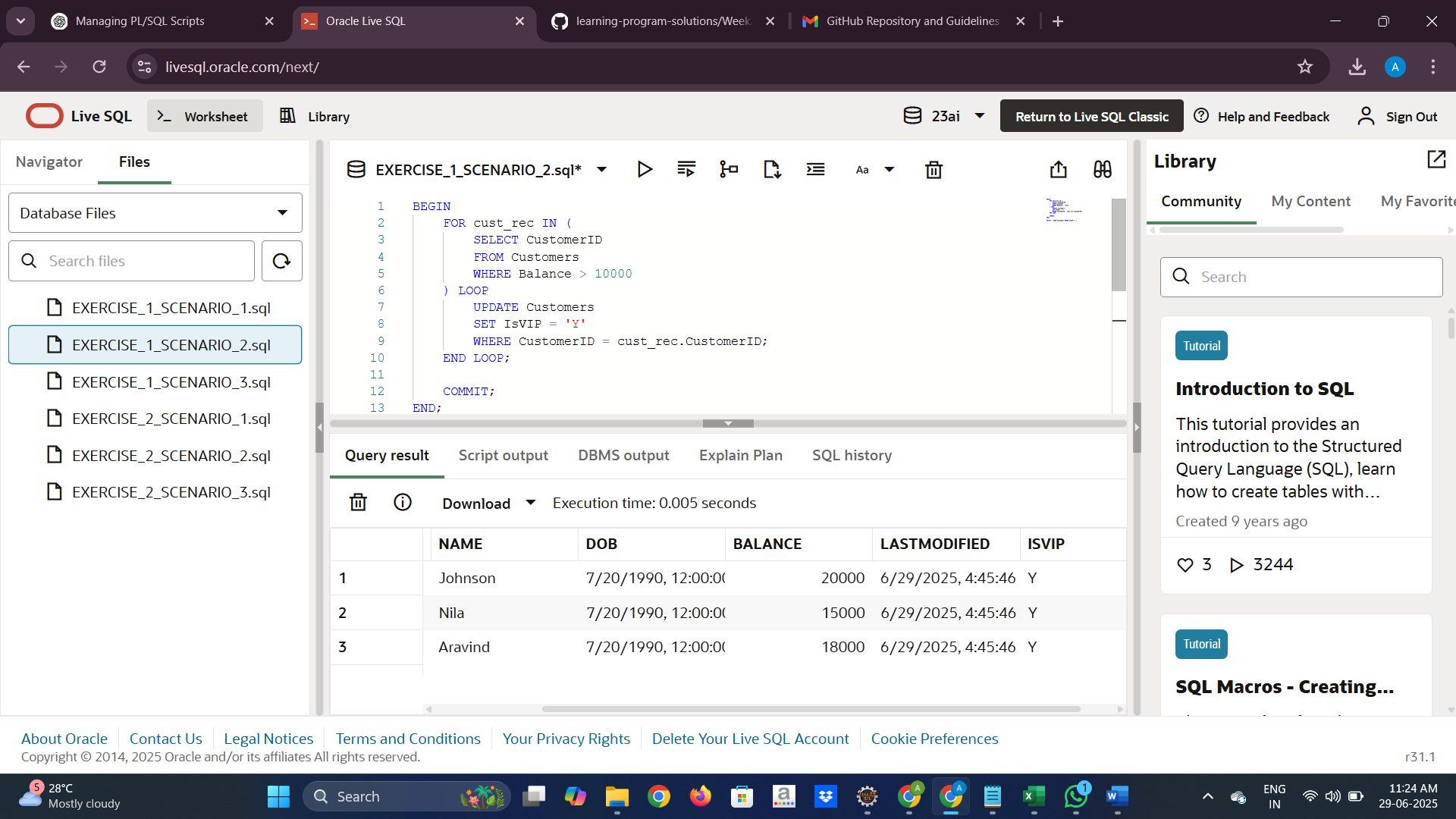
COMMIT;

END;

/

SELECT \* FROM Customers WHERE IsVIP='Y';

**OUTPUT**



**SCENARIO 3:**

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

**CODE:**

BEGIN

    FOR rec IN (

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

        FROM Loans l

        JOIN Customers c ON c.CustomerID = l.CustomerID

        WHERE l.EndDate BETWEEN SYSDATE AND SYSDATE + 30

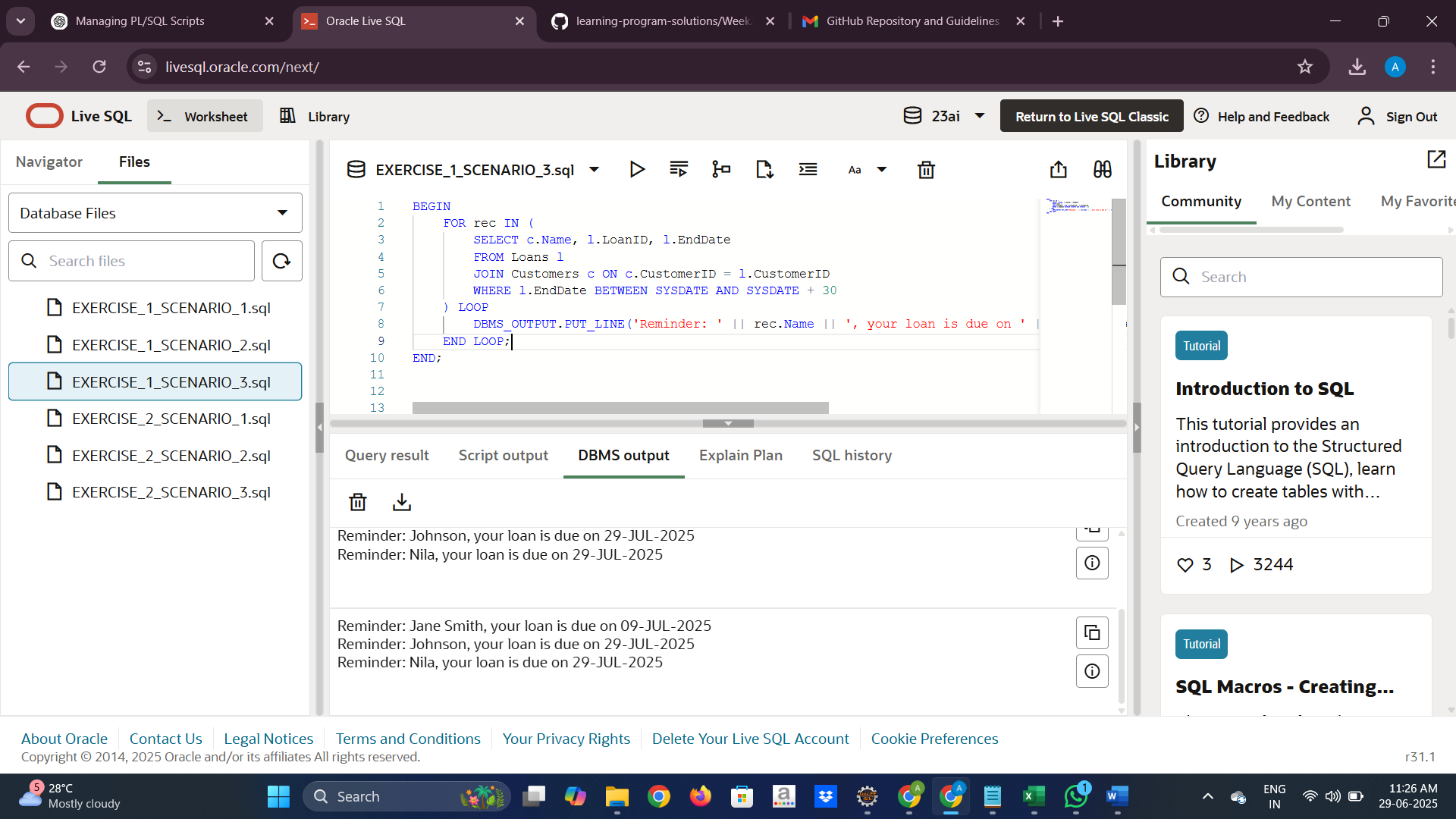
    ) LOOP

        DBMS\_OUTPUT.PUT\_LINE('Reminder: ' || rec.Name || ', your loan is due on ' || TO\_CHAR(rec.EndDate, 'DD-MON-YYYY'));

    END LOOP;

END;

OUTPUT:



**EXERCISE 3: STORED PROCEDURES**

**SCENARIO 1:**

The bank needs to process monthly interest for all savings accounts.

**CODE:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

    UPDATE Accounts

    SET Balance = Balance + (Balance \* 0.01),

        LastModified = SYSDATE

    WHERE AccountType = 'Savings';

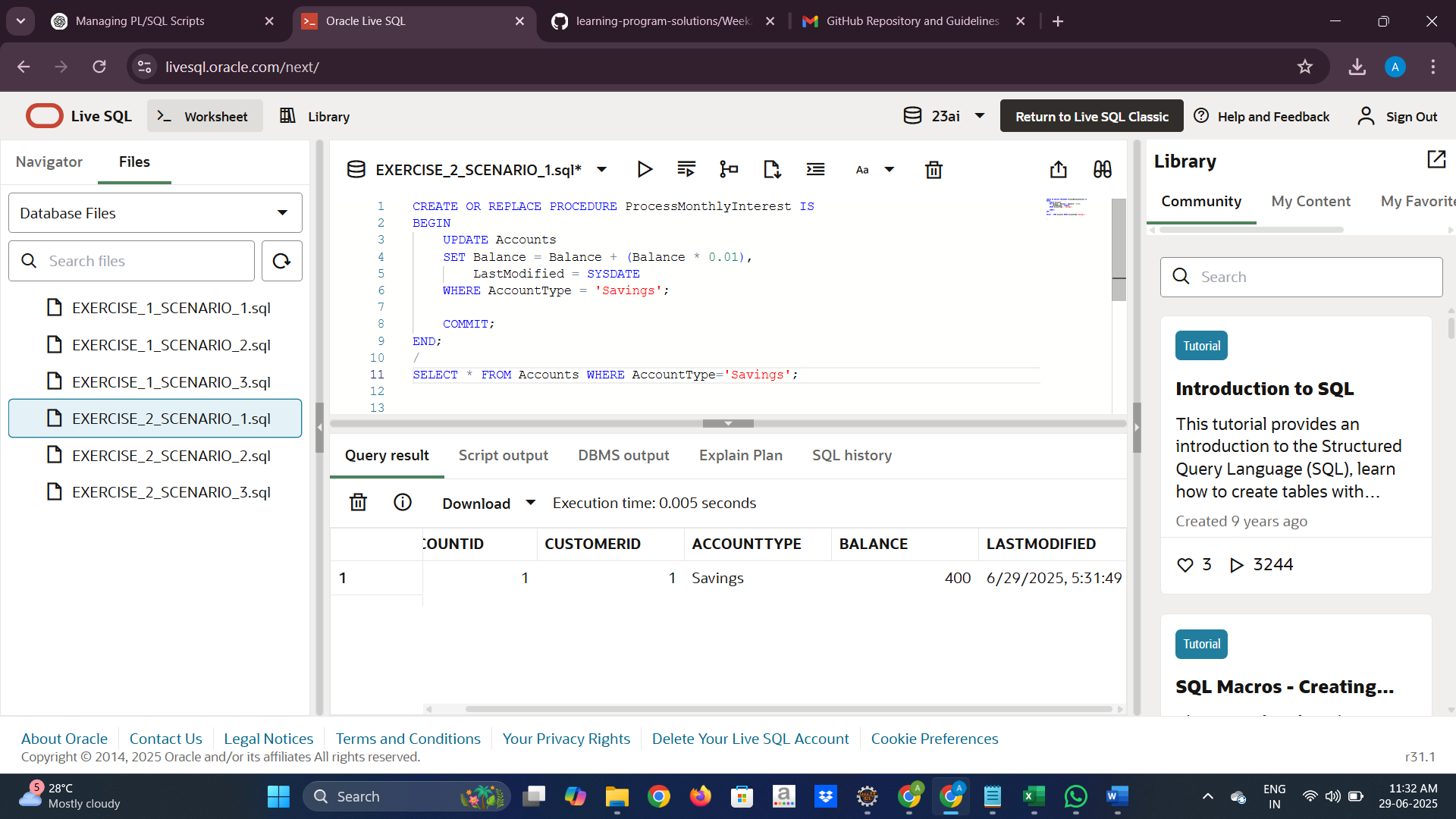
COMMIT;

END;

/

SELECT \* FROM Accounts WHERE AccountType='Savings';

**OUTPUT:**



**SCENARIO 2:**

The bank wants to implement a bonus scheme for employees based on their performance.

**CODE:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus IS

    v\_department     VARCHAR2(50) := 'HR';

    v\_bonus\_percent  NUMBER := 10;

BEGIN

    UPDATE Employees

    SET Salary = Salary + (Salary \* (v\_bonus\_percent / 100))

    WHERE Department = v\_department;

COMMIT;

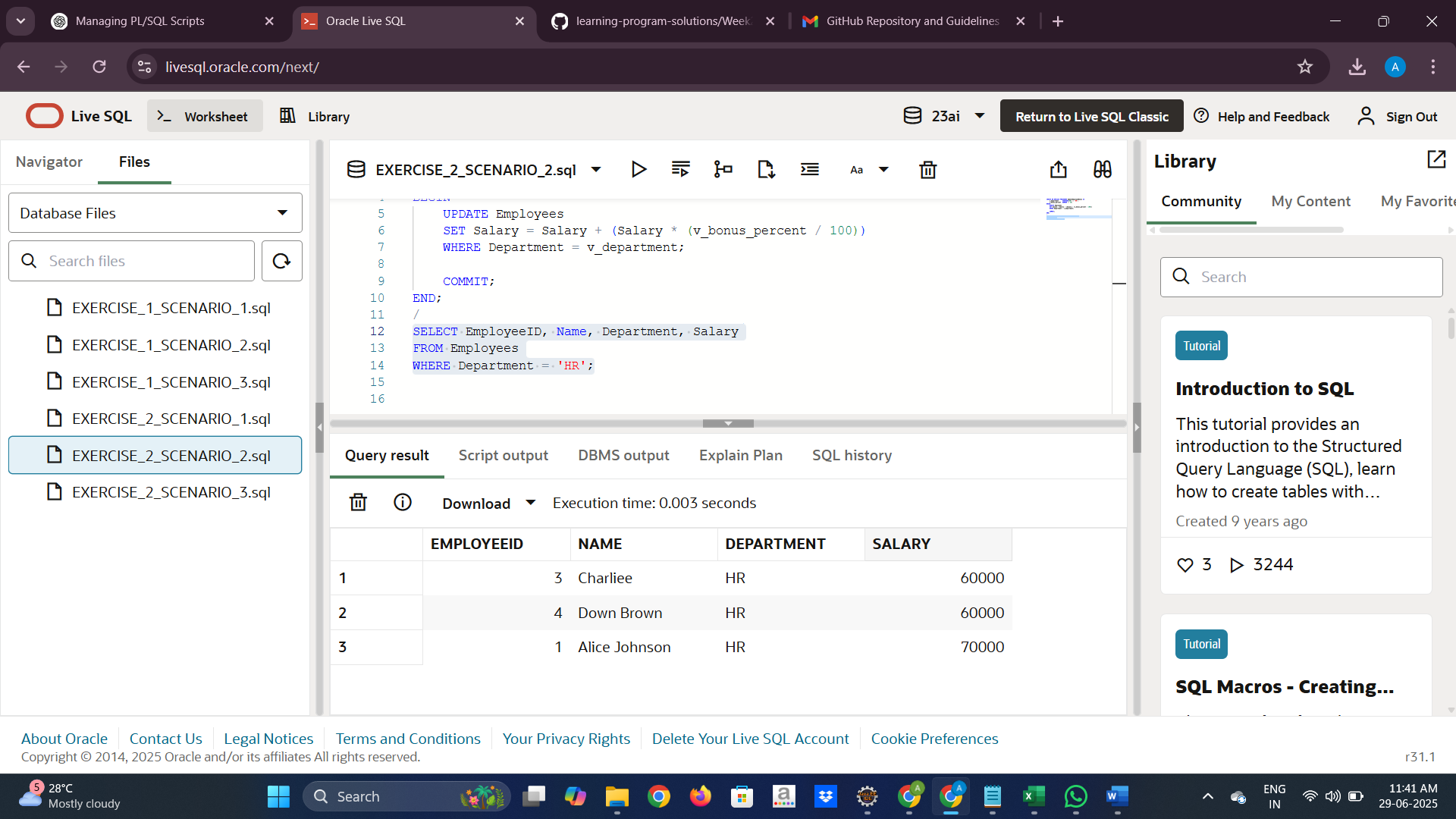
END;

/ SELECT EmployeeID, Name, Department, Salary

FROM Employees

WHERE Department = 'HR';

OUTPUT:



**SCENARIO 3:**

Customers should be able to transfer funds between their accounts.

**CODE:**

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, 'Insufficient balance in source account.');

    END IF;

UPDATE Accounts

    SET Balance = Balance - p\_amount,

        LastModified = SYSDATE

    WHERE AccountID = p\_from\_account;

UPDATE Accounts

    SET Balance = Balance + p\_amount,

        LastModified = SYSDATE

    WHERE AccountID = p\_to\_account;

COMMIT;

END;

/

BEGIN

    TransferFunds(1, 2, 300);

END;

SELECT AccountID, CustomerID, AccountType, Balance, LastModified

FROM Accounts

WHERE AccountID IN (1, 2);

**OUTPUT:**

