**Week 2 - PL/SQL**

**Exercise 1: Control Structures**

-- Customers Table

CREATE TABLE Customers (

    CustomerID     NUMBER PRIMARY KEY,

    CustomerName   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)

);

-- Insert Customers

INSERT INTO Customers VALUES (1, 'Alice Smith', 65, 12000, 'FALSE');

INSERT INTO Customers VALUES (2, 'Bob Johnson', 45, 8000,  'FALSE');

INSERT INTO Customers VALUES (3, 'Carol White', 70, 15000, 'FALSE');

INSERT INTO Customers VALUES (4, 'David Brown', 30, 9500,  'FALSE');

-- Insert Loans

INSERT INTO Loans VALUES (101, 1, 7.5, SYSDATE + 15);  -- Alice, due soon

INSERT INTO Loans VALUES (102, 2, 8.0, SYSDATE + 40);  -- Bob, not due soon

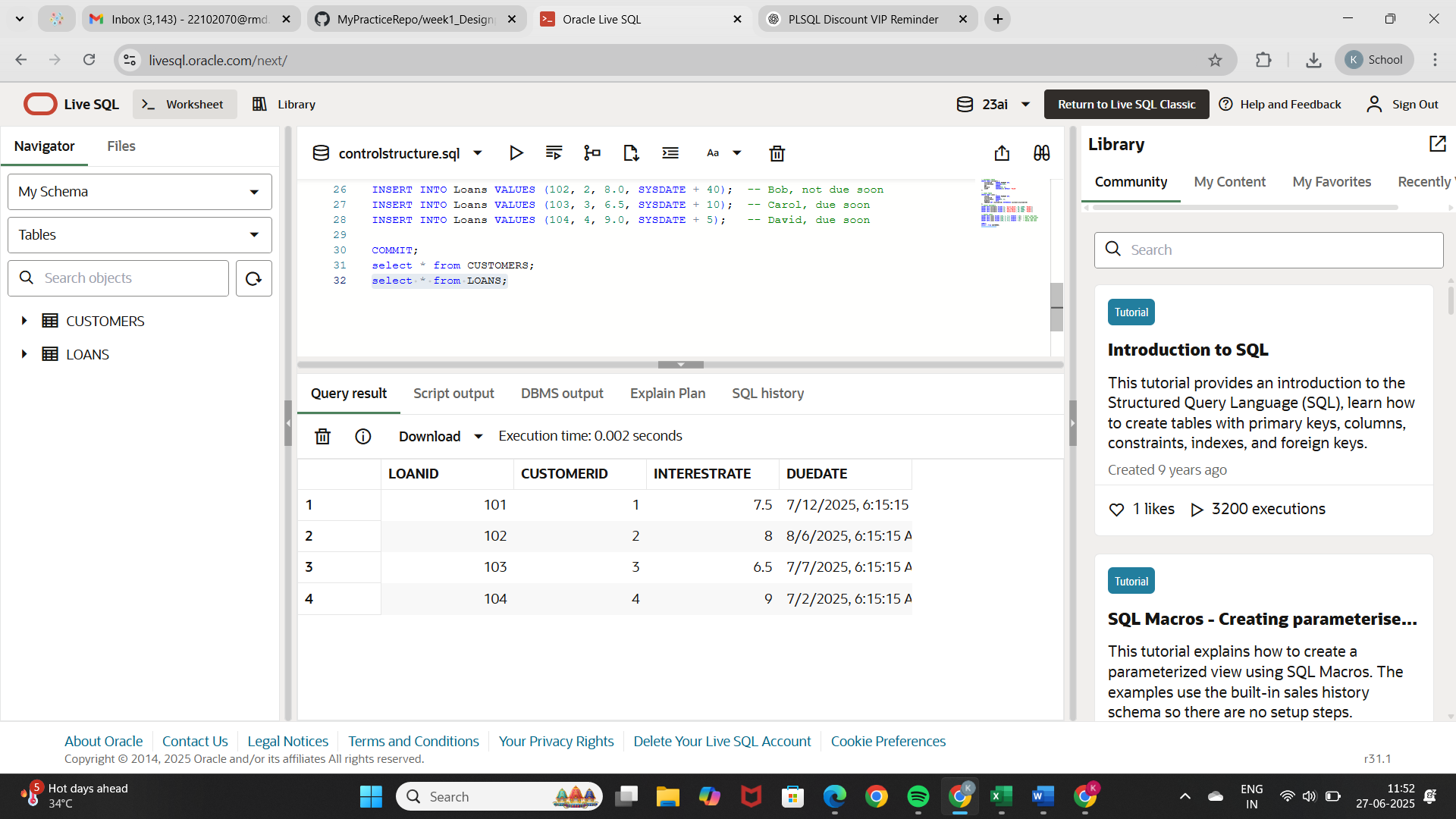
INSERT INTO Loans VALUES (103, 3, 6.5, SYSDATE + 10);  -- Carol, due soon

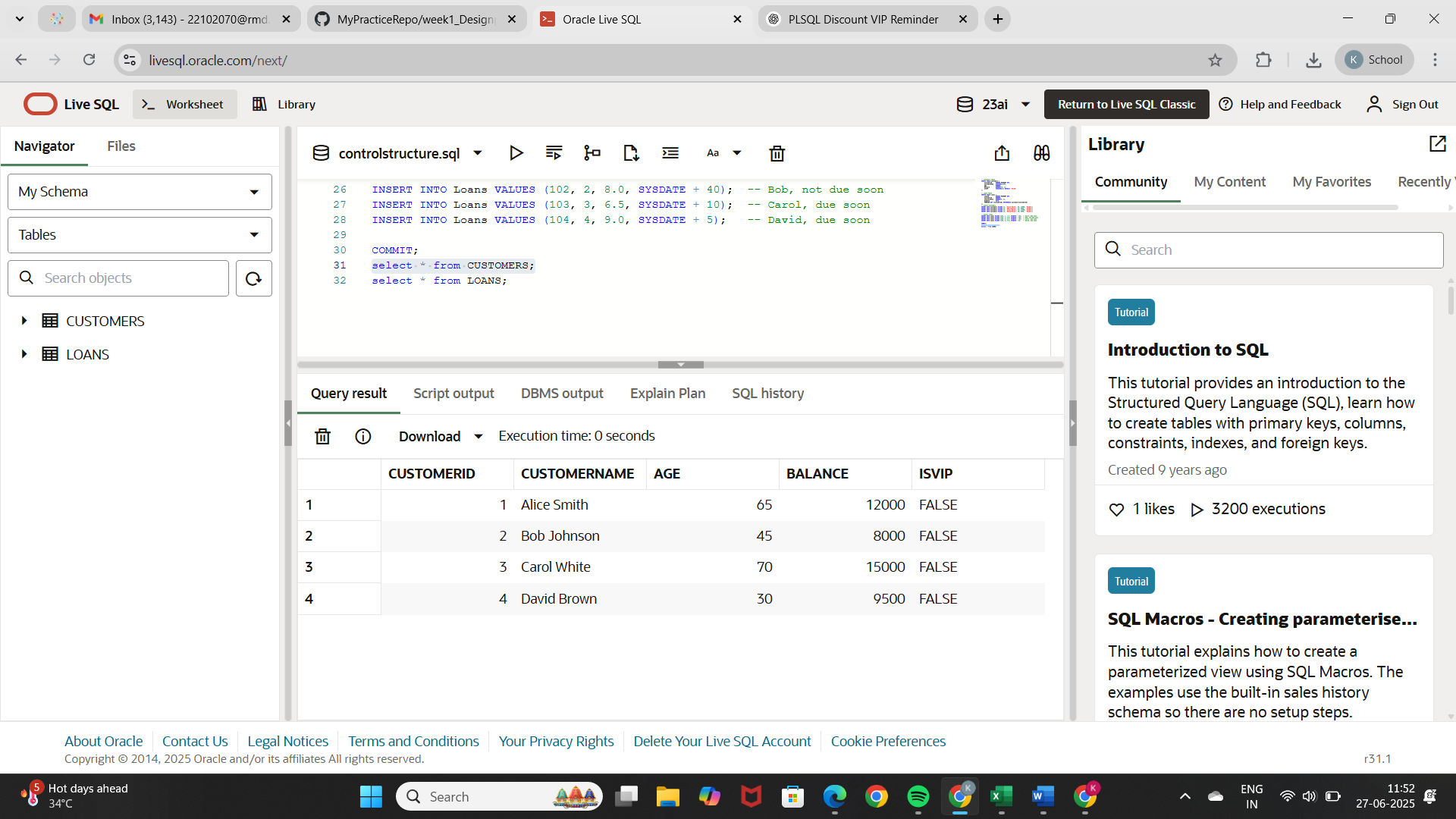
INSERT INTO Loans VALUES (104, 4, 9.0, SYSDATE + 5);   -- David, due soon

COMMIT;

select \* from CUSTOMERS;

select \* from LOANS;





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

o Question: Write a PL/SQL block that loops through all customers, checks their age, and if they are above 60, apply a 1% discount to their current loan interest rates.

BEGIN

  FOR cust IN (SELECT CustomerID FROM Customers WHERE Age > 60) LOOP

    UPDATE Loans

    SET InterestRate = InterestRate - 1

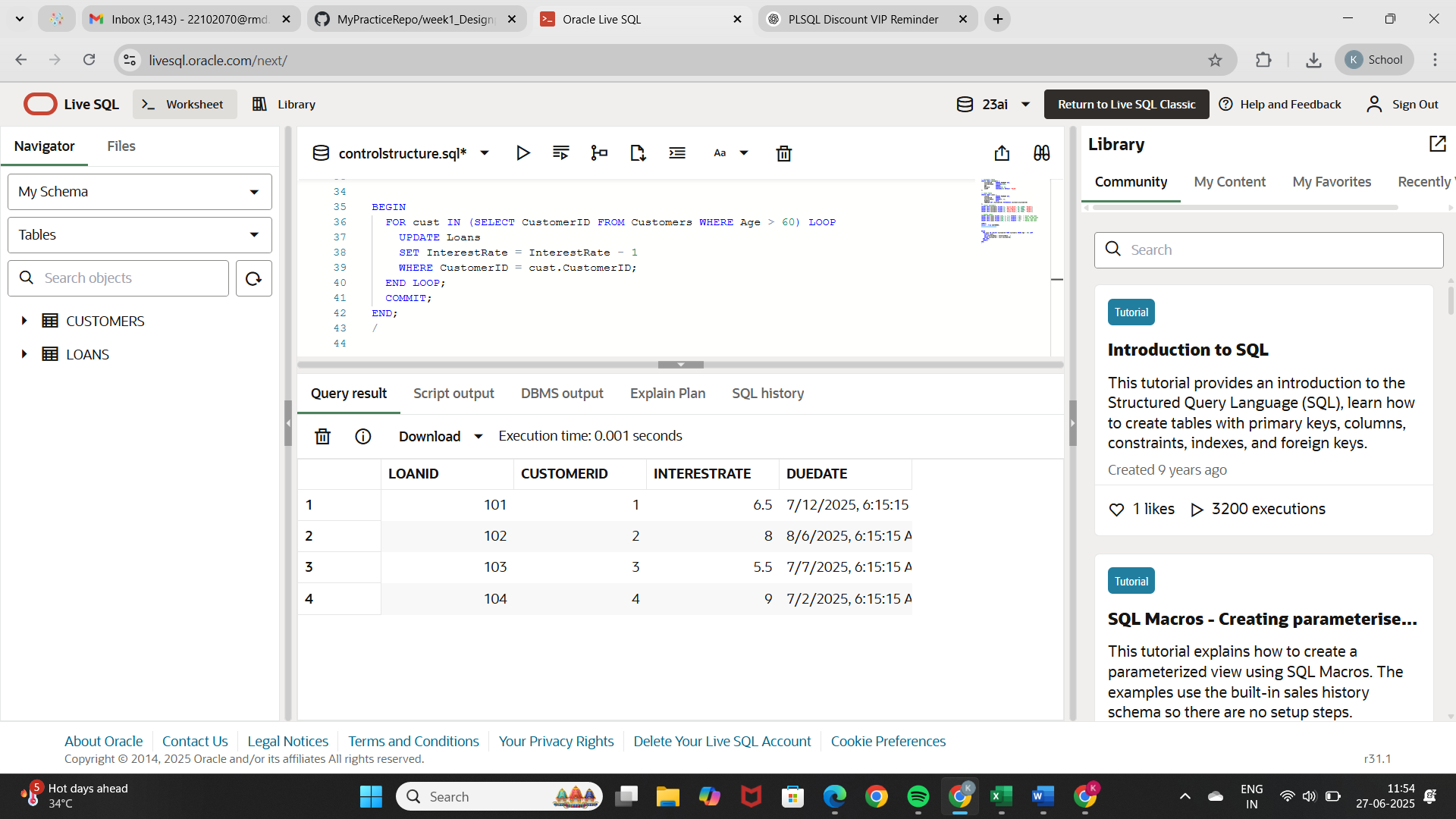
    WHERE CustomerID = cust.CustomerID;

  END LOOP;

  COMMIT;

END;

/

**Output:**

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

o Question: Write a PL/SQL block that iterates through all customers and sets a flag IsVIP to TRUE for those with a balance over $10,000.

BEGIN

  FOR cust IN (SELECT CustomerID FROM Customers WHERE Balance > 10000) LOOP

    UPDATE Customers

    SET IsVIP = 'TRUE'

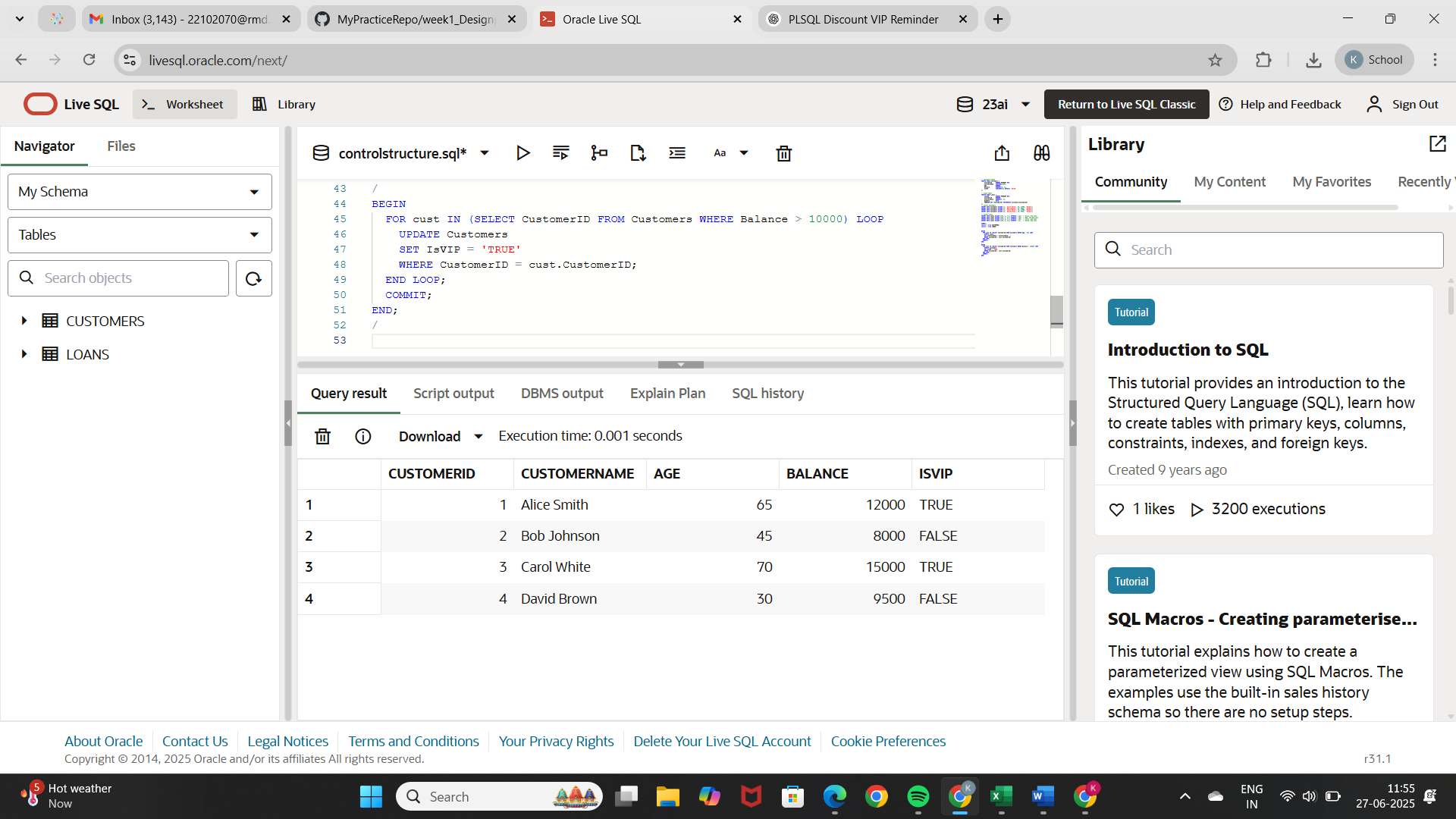
    WHERE CustomerID = cust.CustomerID;

  END LOOP;

  COMMIT;

END;

/

**Output:**

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

o Question: Write a PL/SQL block that fetches all loans due in the next 30 days and prints a reminder message for each customer.

BEGIN

FOR loan IN (

SELECT l.LoanID, c.CustomerName, l.DueDate

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID

WHERE l.DueDate BETWEEN SYSDATE AND SYSDATE + 30

) LOOP

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

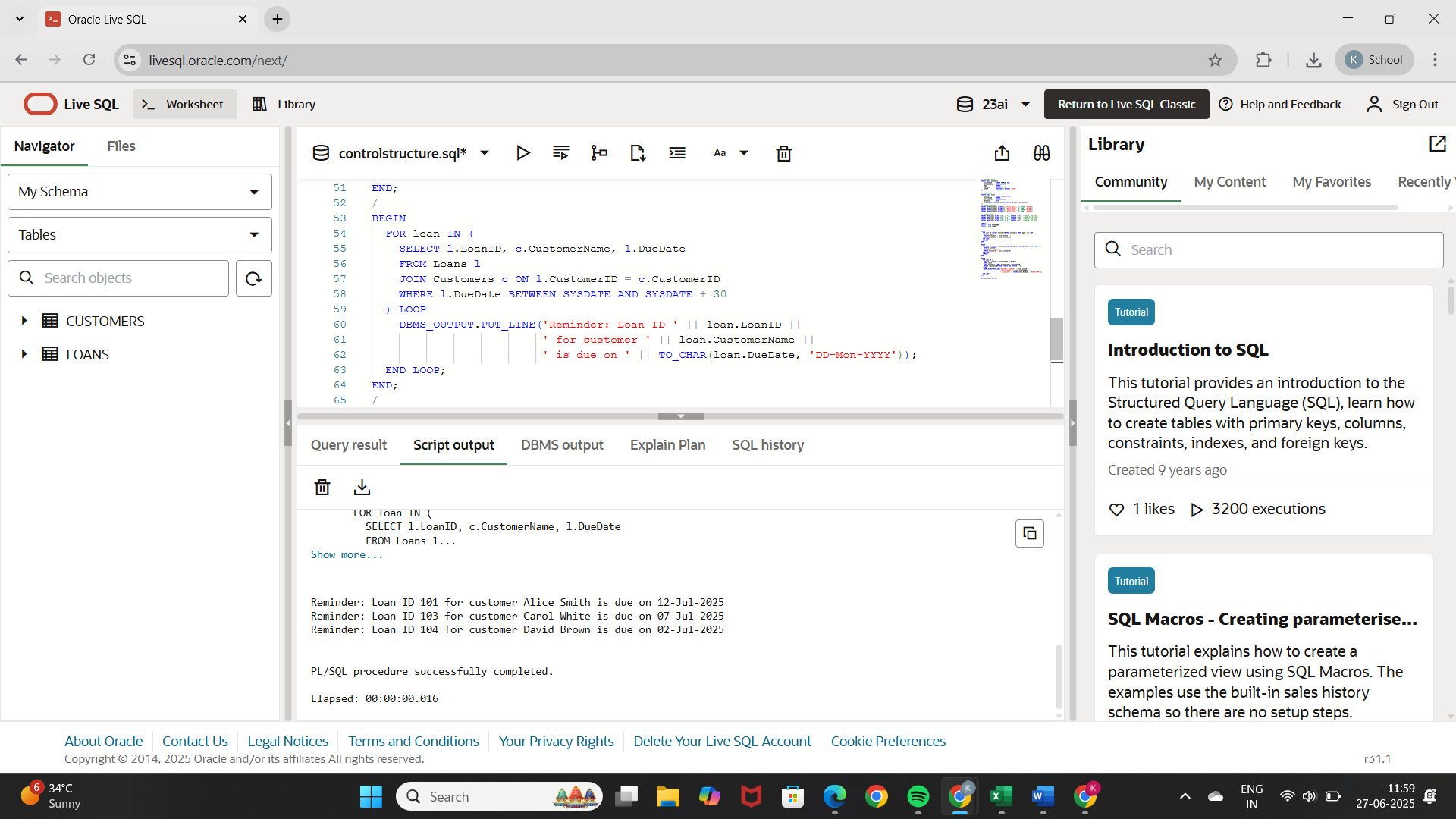
' for customer ' || loan.CustomerName ||

' is due on ' || TO\_CHAR(loan.DueDate, 'DD-Mon-YYYY'));

END LOOP;

END;

/

**Output:**

**Exercise 3: Stored Procedures**

-- Create Customers table

CREATE TABLE Customers (

    CustomerID     NUMBER PRIMARY KEY,

    CustomerName   VARCHAR2(100),

    Age            NUMBER,

    Balance        NUMBER(10, 2),

    IsVIP          VARCHAR2(5) DEFAULT 'FALSE'

);

-- Create Accounts table

CREATE TABLE Accounts (

    AccountID     NUMBER PRIMARY KEY,

    CustomerName  VARCHAR2(100),

    AccountType   VARCHAR2(20),

    Balance       NUMBER(10, 2)

);

-- Create Employees table

CREATE TABLE Employees (

    EmployeeID     NUMBER PRIMARY KEY,

    EmployeeName   VARCHAR2(100),

    Department     VARCHAR2(50),

    Salary         NUMBER(10, 2)

);

-- Insert Customers

INSERT INTO Customers VALUES (1, 'Alice Smith', 65, 12000, 'FALSE');

INSERT INTO Customers VALUES (2, 'Bob Johnson', 45, 8000,  'FALSE');

INSERT INTO Customers VALUES (3, 'Carol White', 70, 15000, 'FALSE');

INSERT INTO Customers VALUES (4, 'David Brown', 30, 9500,  'FALSE');

-- Insert Accounts

INSERT INTO Accounts VALUES (1, 'Alice Smith', 'SAVINGS', 10000);

INSERT INTO Accounts VALUES (2, 'Bob Johnson', 'SAVINGS', 15000);

INSERT INTO Accounts VALUES (3, 'Carol White', 'CURRENT', 8000);

INSERT INTO Accounts VALUES (4, 'David Brown', 'SAVINGS', 20000);

-- Insert Employees

INSERT INTO Employees VALUES (101, 'John Doe', 'HR', 50000);

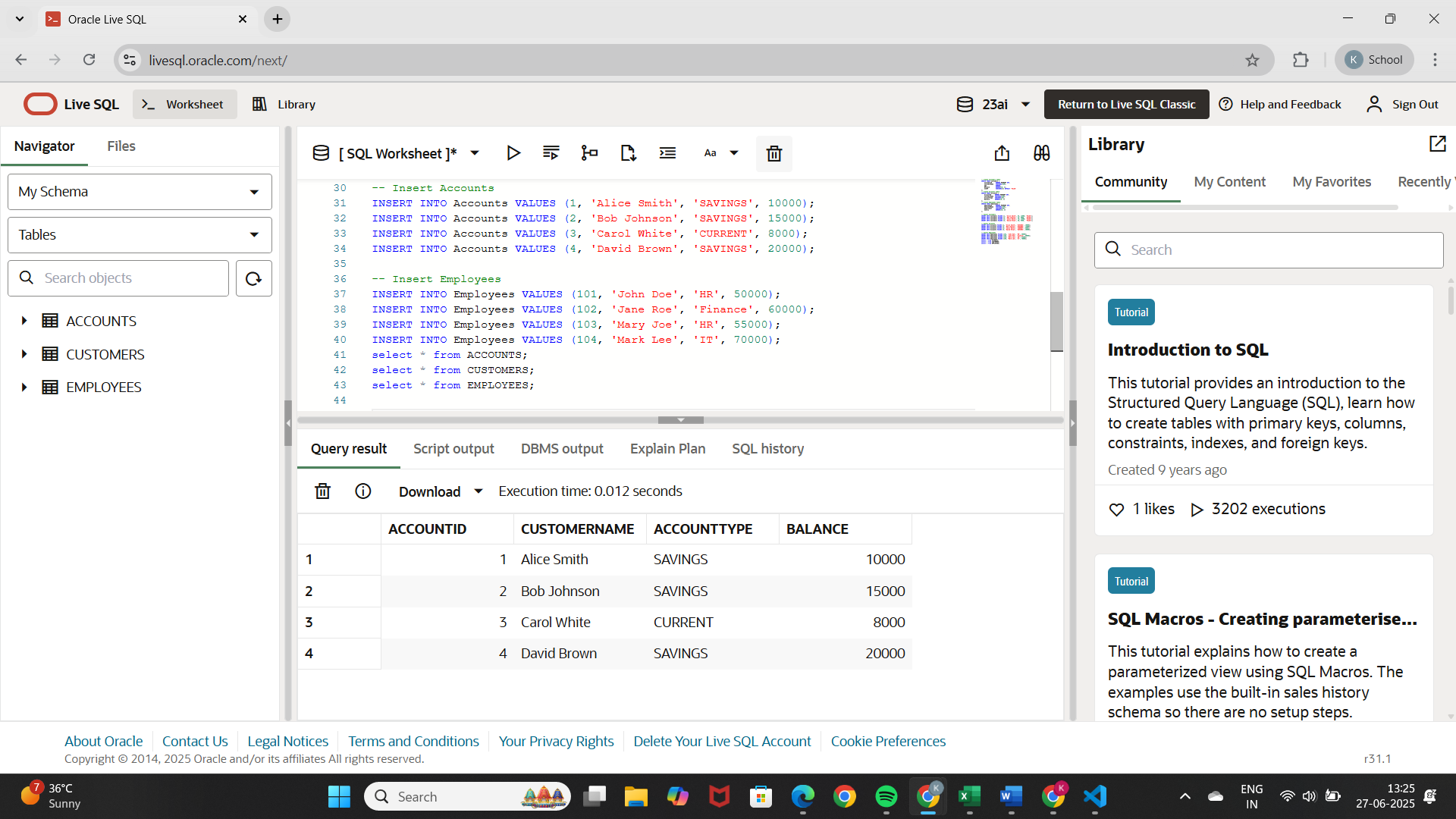
INSERT INTO Employees VALUES (102, 'Jane Roe', 'Finance', 60000);

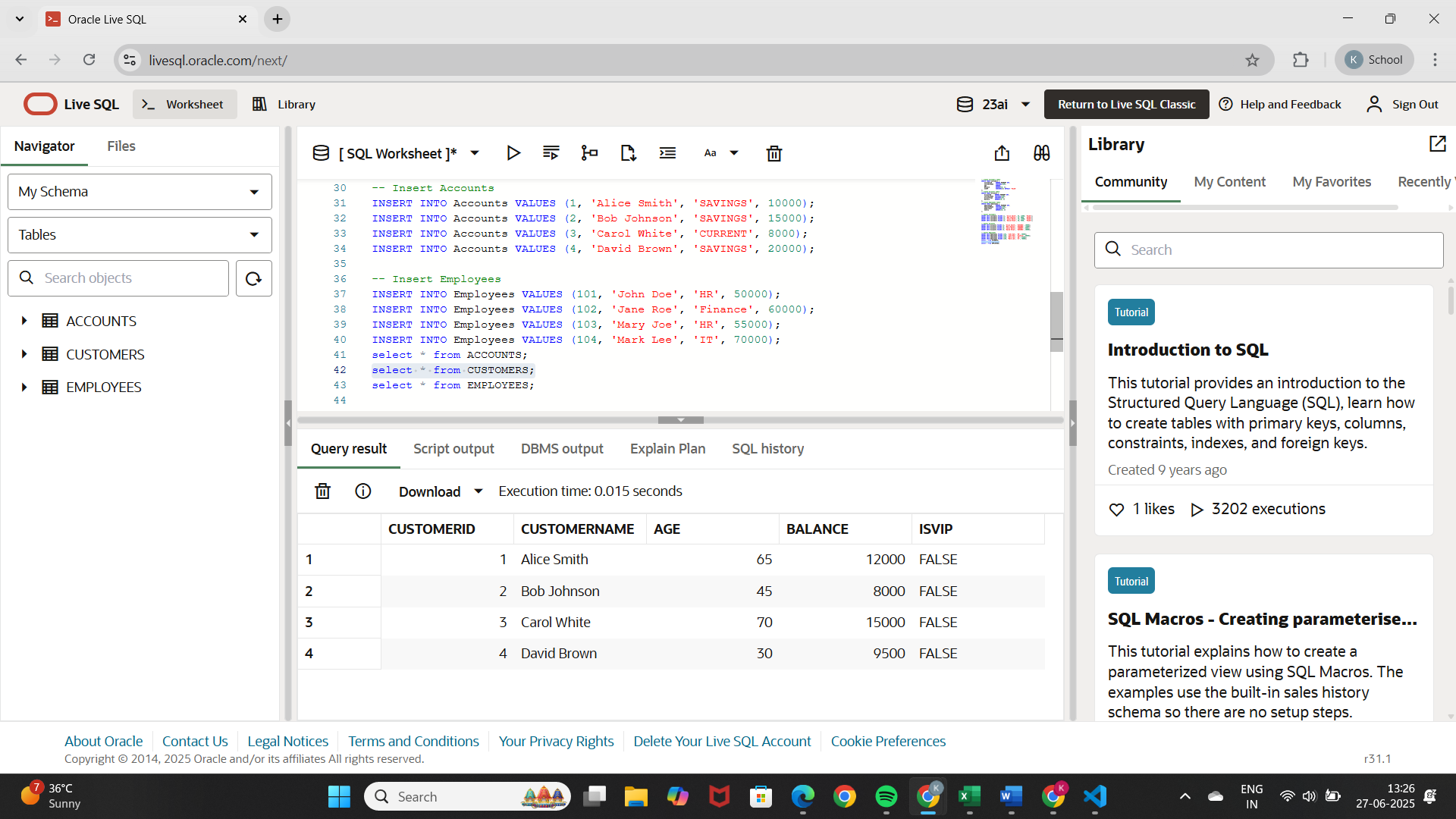
INSERT INTO Employees VALUES (103, 'Mary Joe', 'HR', 55000);

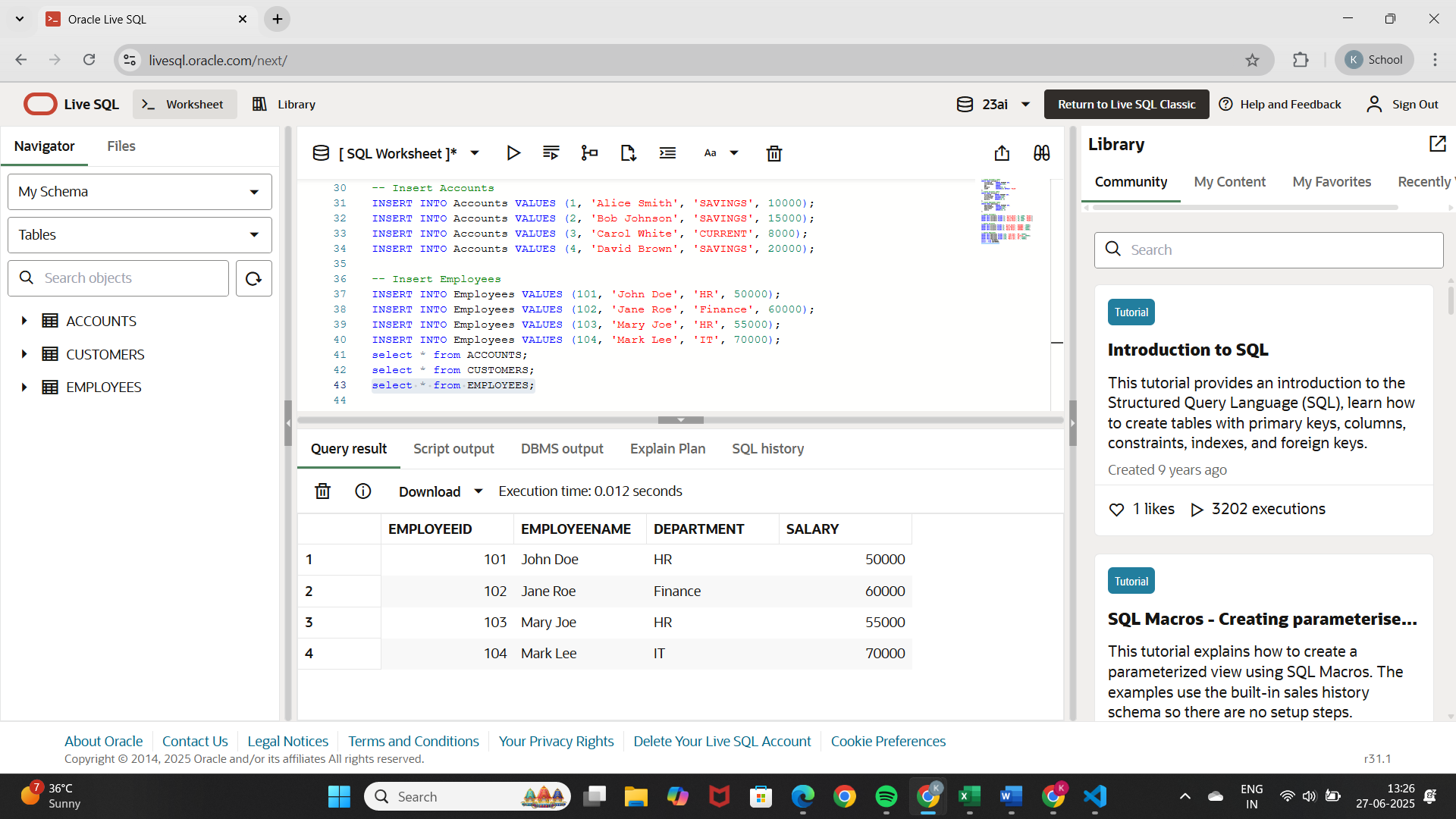
INSERT INTO Employees VALUES (104, 'Mark Lee', 'IT', 70000);

select \* from ACCOUNTS;

select \* from CUSTOMERS;

****select \* from EMPLOYEES;

****

****

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

o 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 OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

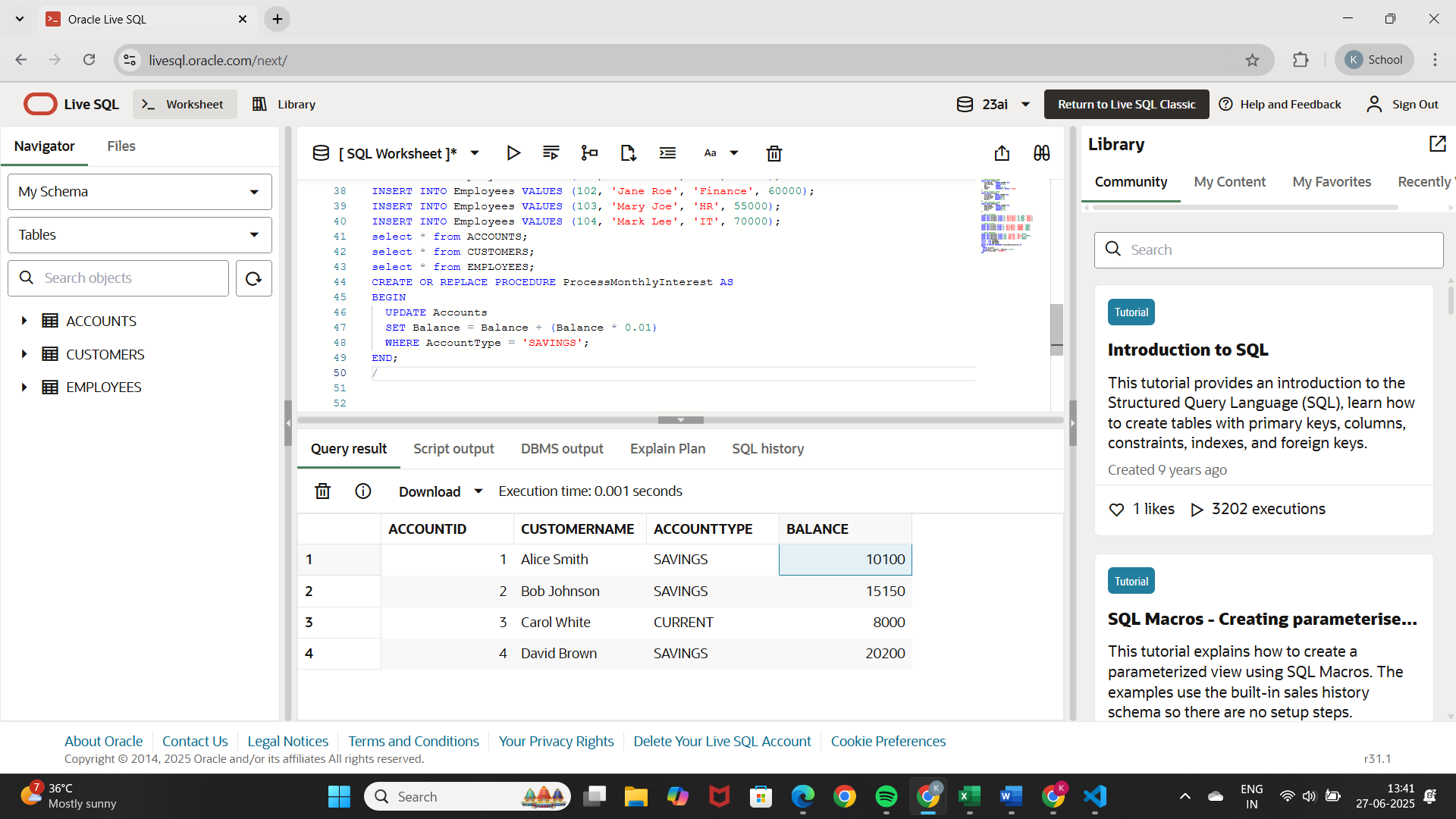
  UPDATE Accounts

  SET Balance = Balance + (Balance \* 0.01)

  WHERE AccountType = 'SAVINGS';

END;

/

****

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

o Question: Write a stored procedure UpdateEmployeeBonus that updates the salary of employees in a given department by adding a bonus percentage passed as a parameter.

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

  p\_Department IN VARCHAR2,

  p\_BonusPercent IN NUMBER

) AS

BEGIN

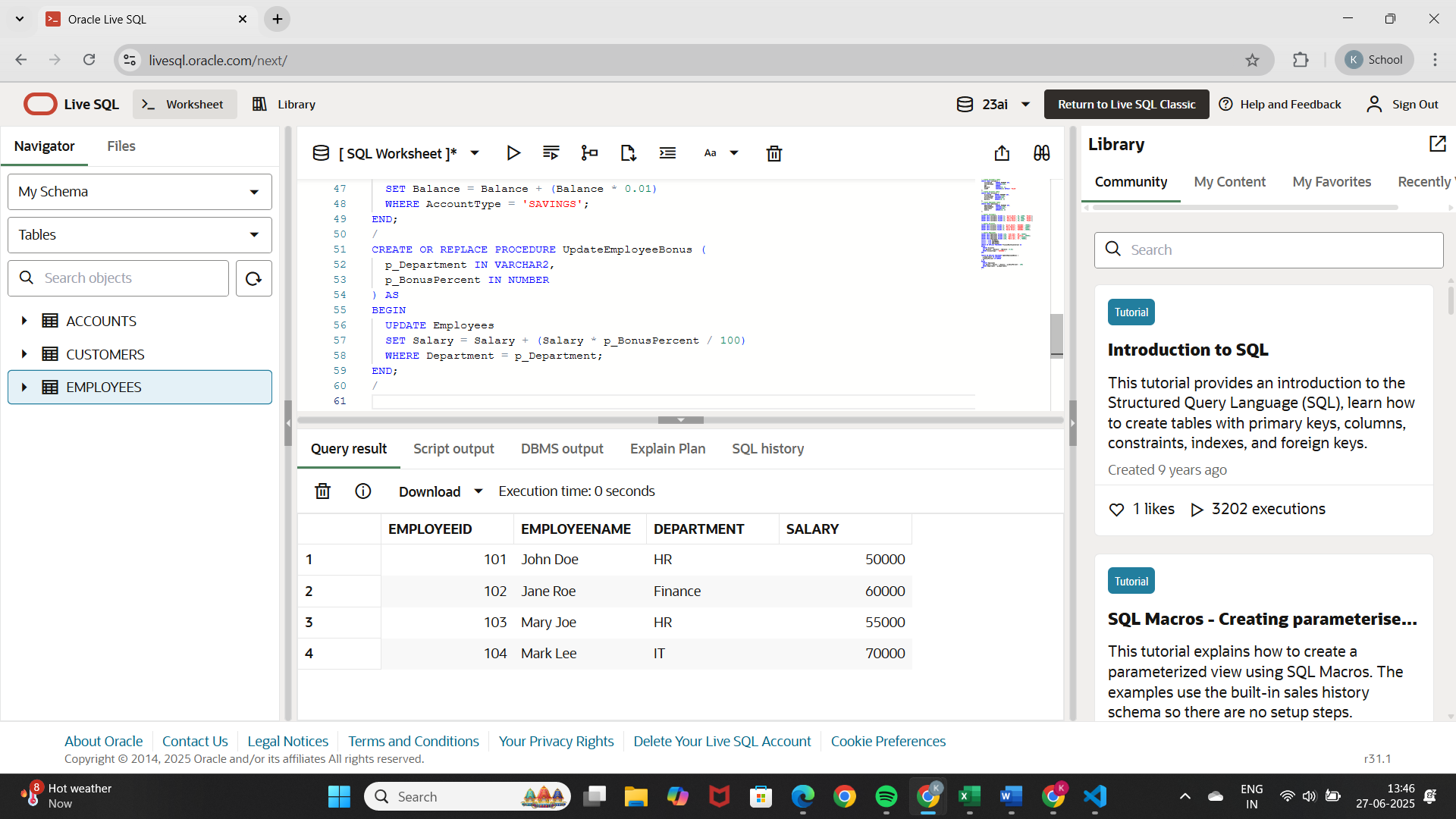
  UPDATE Employees

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

  WHERE Department = p\_Department;

END;

/

**Output:**

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

o Question: Write a stored procedure TransferFunds that transfers a specified amount from one account to another, checking that the source account has sufficient balance before making the transfer.

CREATE OR REPLACE PROCEDURE TransferFunds (

  p\_FromAccountID IN NUMBER,

  p\_ToAccountID   IN NUMBER,

  p\_Amount        IN NUMBER

) AS

  v\_FromBalance NUMBER;

BEGIN

  -- Get balance of source account

  SELECT Balance INTO v\_FromBalance

  FROM Accounts

  WHERE AccountID = p\_FromAccountID;

  -- Check if sufficient balance exists

  IF v\_FromBalance < p\_Amount THEN

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

  END IF;

  -- Deduct from source

  UPDATE Accounts

  SET Balance = Balance - p\_Amount

  WHERE AccountID = p\_FromAccountID;

  -- Add to target

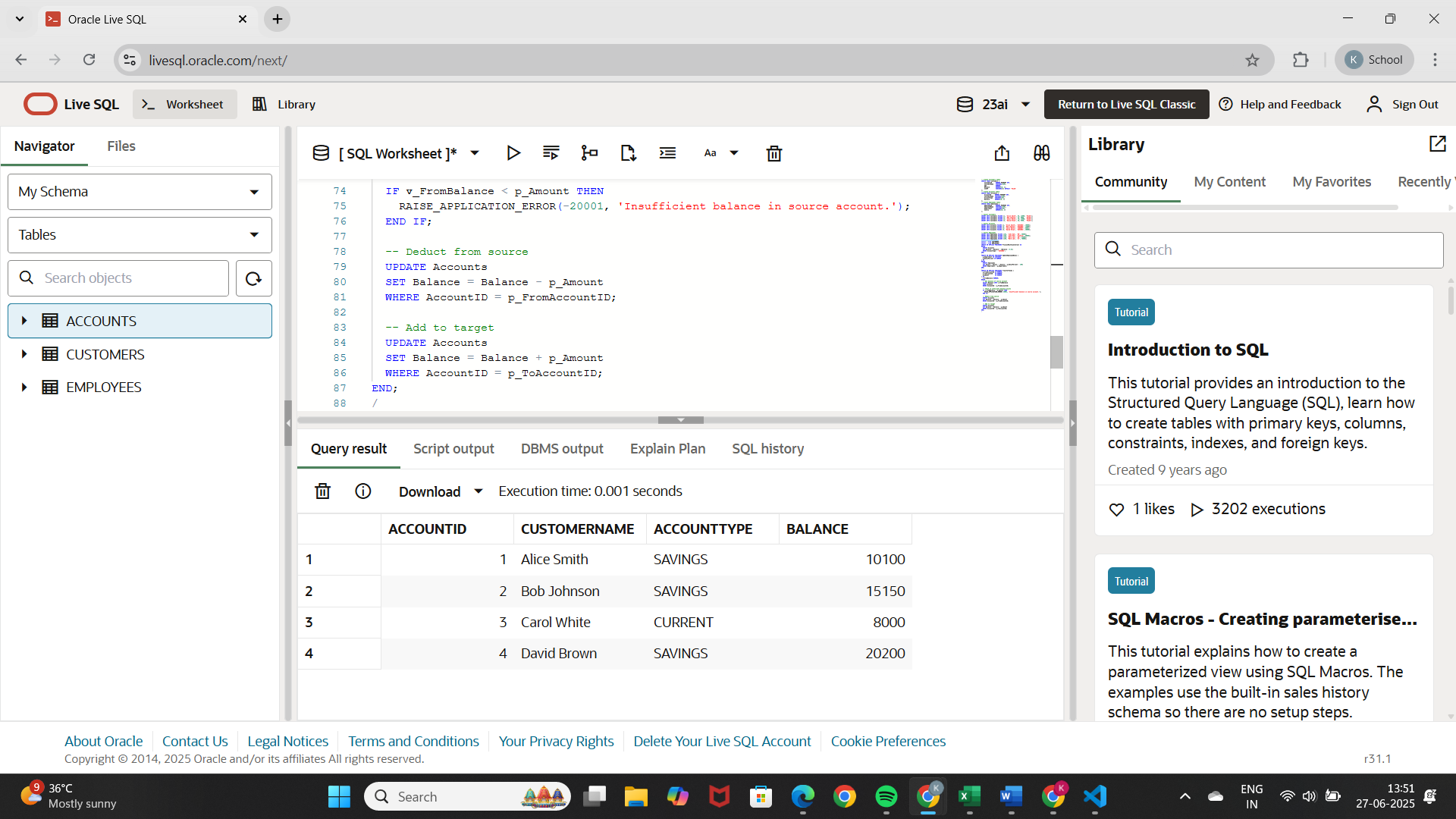
  UPDATE Accounts

  SET Balance = Balance + p\_Amount

  WHERE AccountID = p\_ToAccountID;

END;

/

**output:**