**PL/SQL Exercises**

**Exercise-1**

Exercise 1:

* Scenario 1:

**Solution:**

**//** For adding column isVIP in Customers table **//**

ALTER TABLE Customers ADD IsVIP VARCHAR2(5);

**//** Decrementing interest rate by 1 for customers above 60 years of age **//**

BEGIN

FOR c IN (

SELECT CustomerID

FROM Customers

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

) LOOP

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE CustomerID = c.CustomerID;

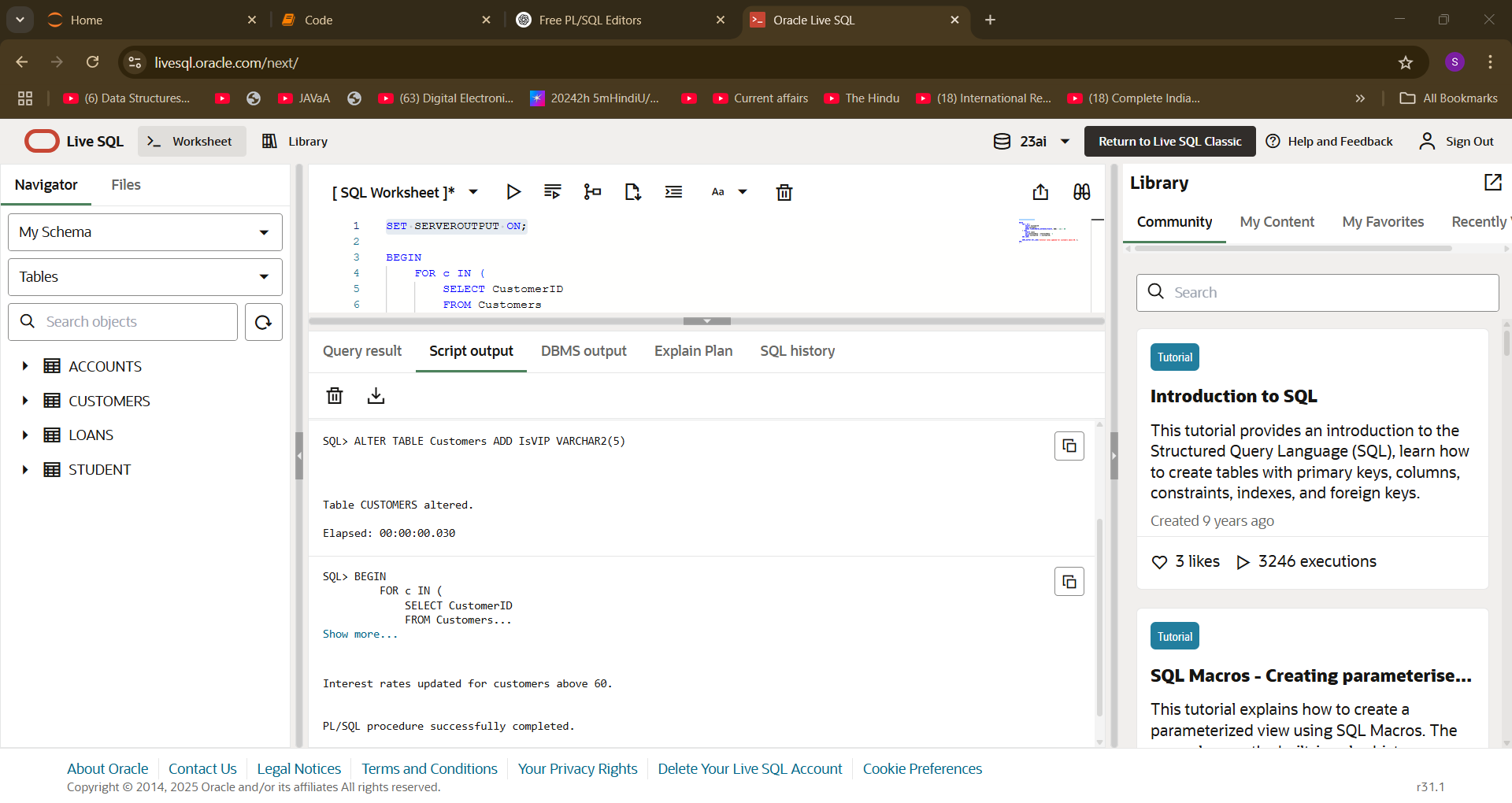
END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Interest rates updated for customers above 60.');

END;

/

**OUTPUT:**

****

Exercise 1:

* Scenario 2:

**Solution:**

**//** Setting flag isVIP as per customer balance with threshold at 10000 **//**

BEGIN

FOR c IN (

SELECT CustomerID, Balance FROM Customers

) LOOP

IF c.Balance > 10000 THEN

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = c.CustomerID;

ELSE

UPDATE Customers

SET IsVIP = 'FALSE'

WHERE CustomerID = c.CustomerID;

END IF;

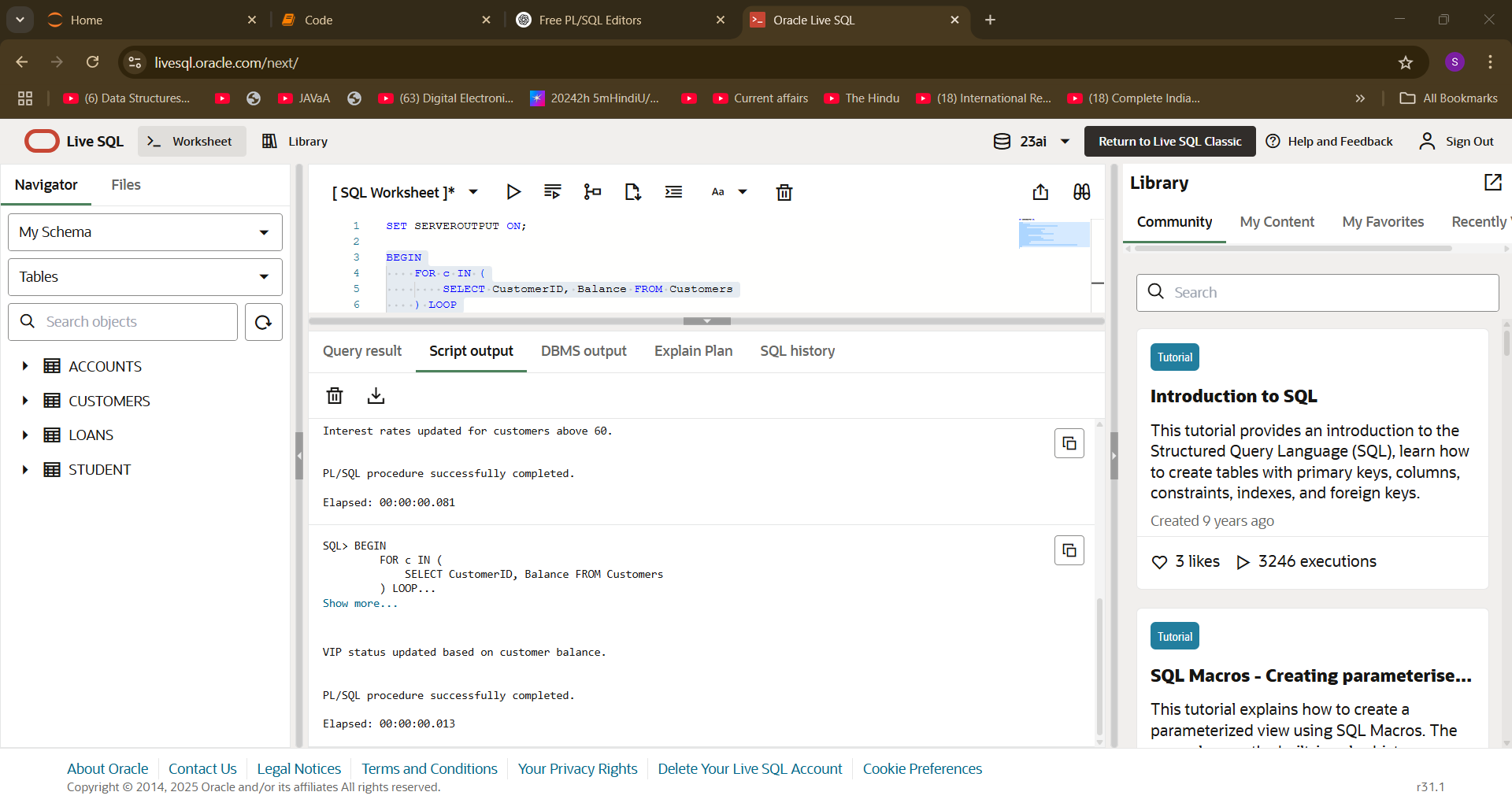
END LOOP;

DBMS\_OUTPUT.PUT\_LINE('VIP status updated based on customer balance.');

END;

/

**OUTPUT:**

****

Exercise 1:

* Scenario 3:

**Solution:**

**//** Printing reminder message for customers with loan due in next 30 days **//**

BEGIN

FOR l IN (

SELECT L.LoanID, L.CustomerID, C.Name, 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: Loan ID ' || l.LoanID ||

' for customer ' || l.Name ||

' (Customer ID ' || l.CustomerID ||

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

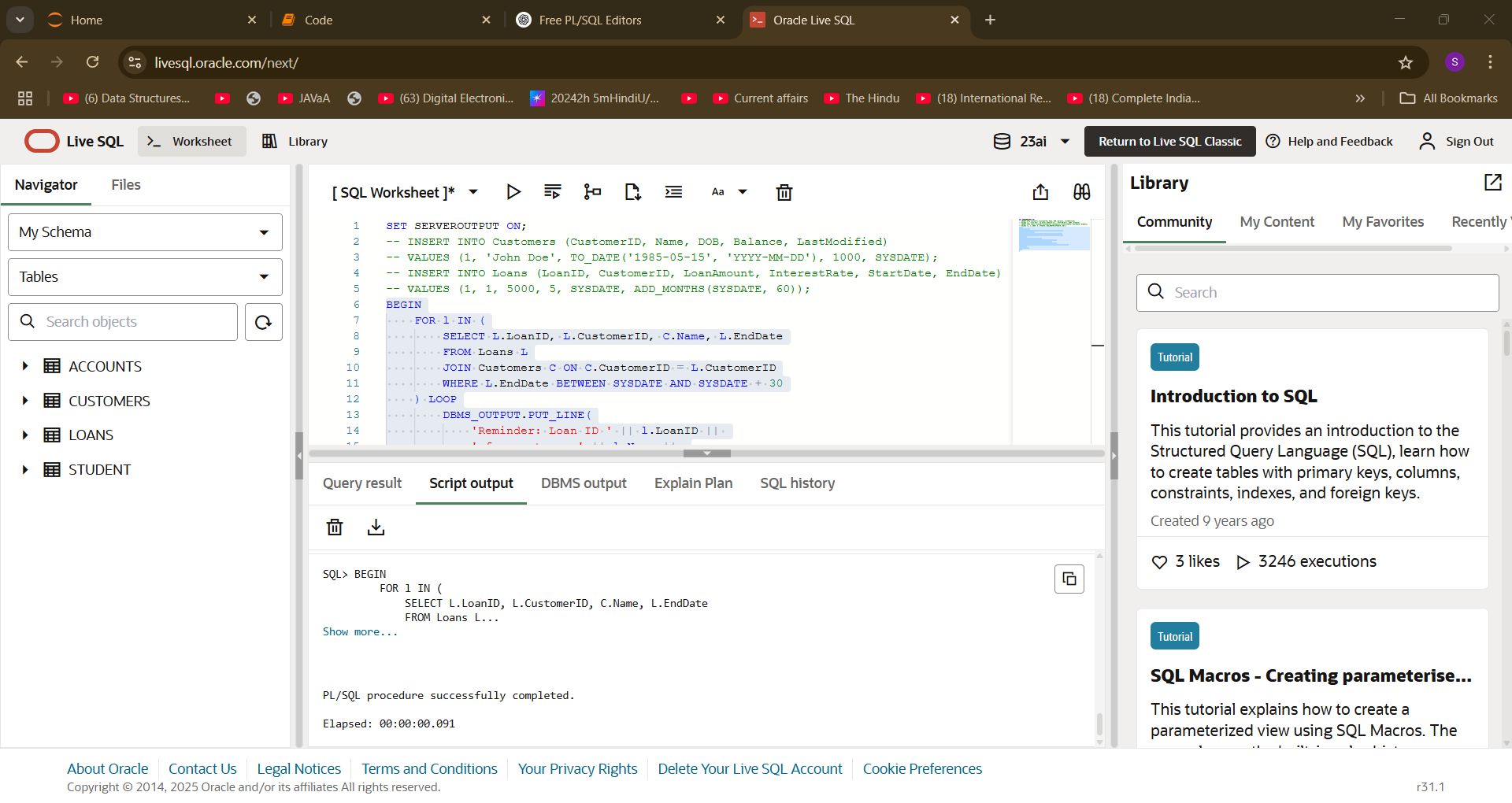
);

END LOOP;

END;

/

**OUTPUT:**

****

**Exercise-3**

Exercise 3:

* Scenario 1:

**Solution:**

**//** Monthly interest for all savings accounts **//**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

FOR acc IN (

SELECT AccountID, Balance

FROM Accounts

WHERE AccountType = 'Savings'

) LOOP

UPDATE Accounts

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

LastModified = SYSDATE

WHERE AccountID = acc.AccountID;

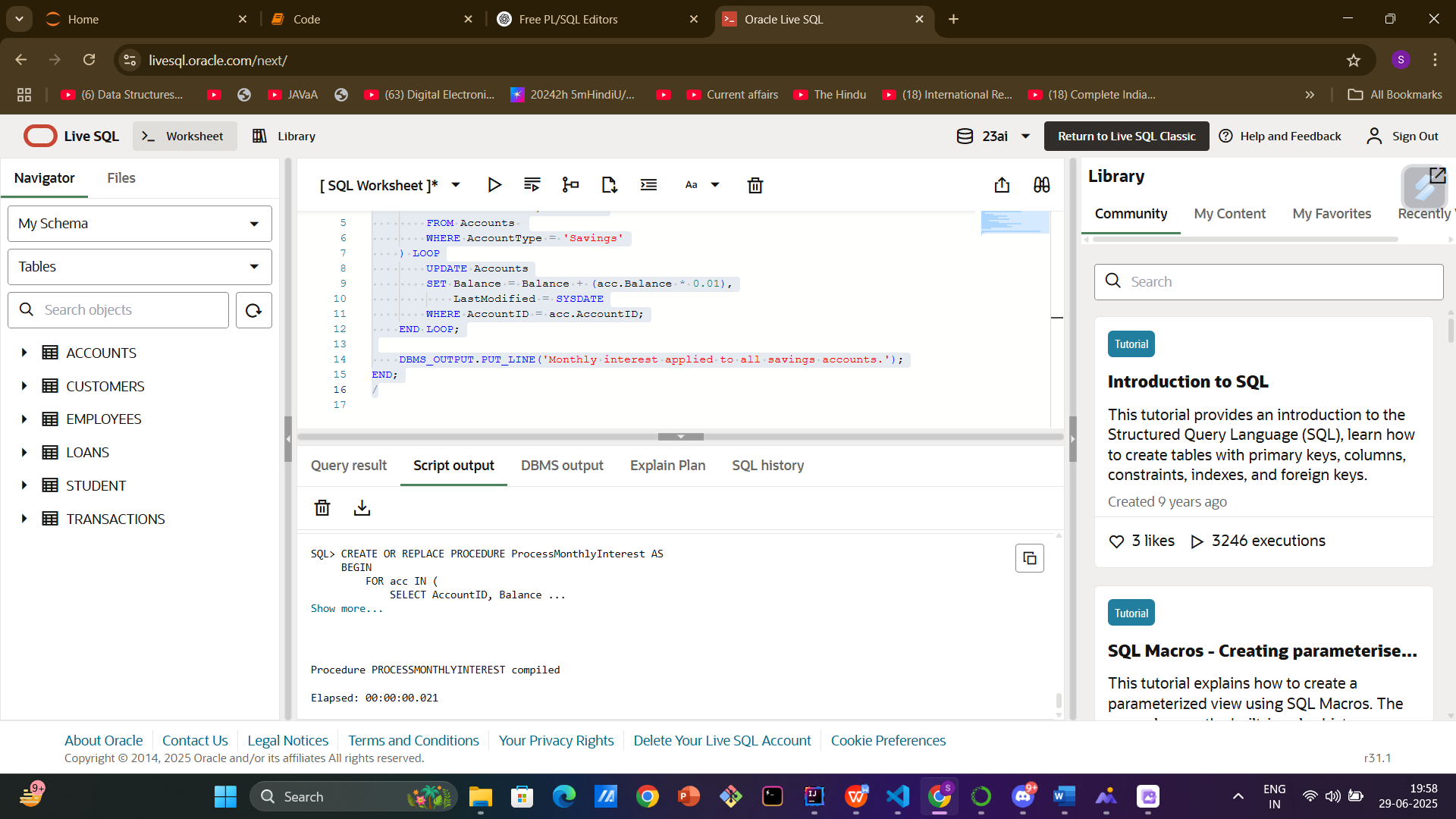
END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Monthly interest applied to all savings accounts.');

END;

/

**OUTPUT:**

****

Exercise 3:

* Scenario 2:

**Solution:**

**//** Employee Bonus by Department**//**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_department IN VARCHAR2,

p\_bonus\_percent IN NUMBER

) AS

BEGIN

UPDATE Employees

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

WHERE Department = p\_department;

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

EXCEPTION

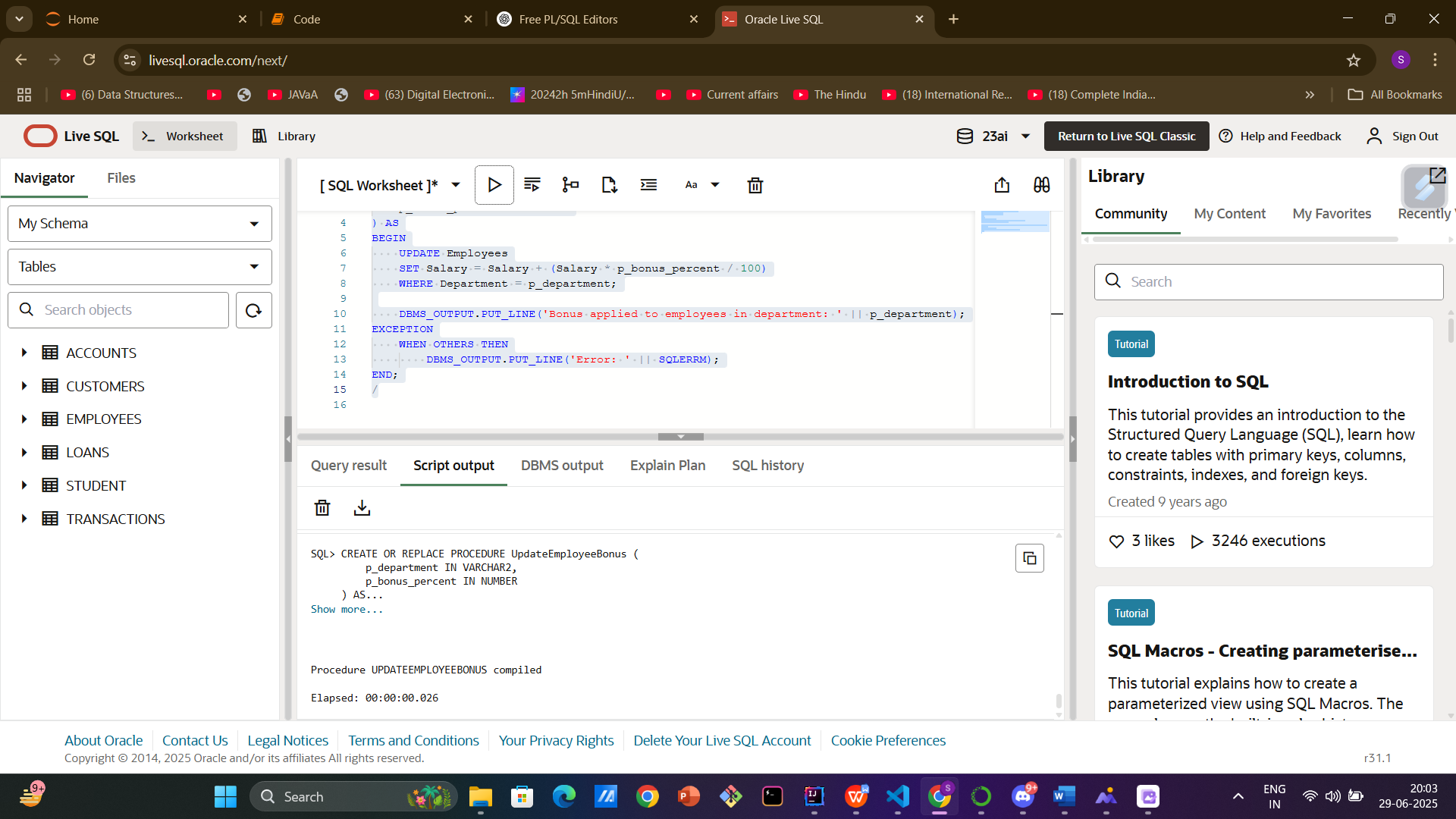
WHEN OTHERS THEN

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

END;

/

**OUTPUT:**



Exercise 3:

* Scenario 3:

**Solution:**

**//** Transfer funds between accounts**//**

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_from\_account IN NUMBER,

p\_to\_account IN NUMBER,

p\_amount IN NUMBER

) AS

v\_balance NUMBER;

BEGIN

SELECT Balance INTO v\_balance

FROM Accounts

WHERE AccountID = p\_from\_account

FOR UPDATE;

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;

DBMS\_OUTPUT.PUT\_LINE('Transferred ' || p\_amount || ' from Account ' || p\_from\_account || ' to Account ' || p\_to\_account);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

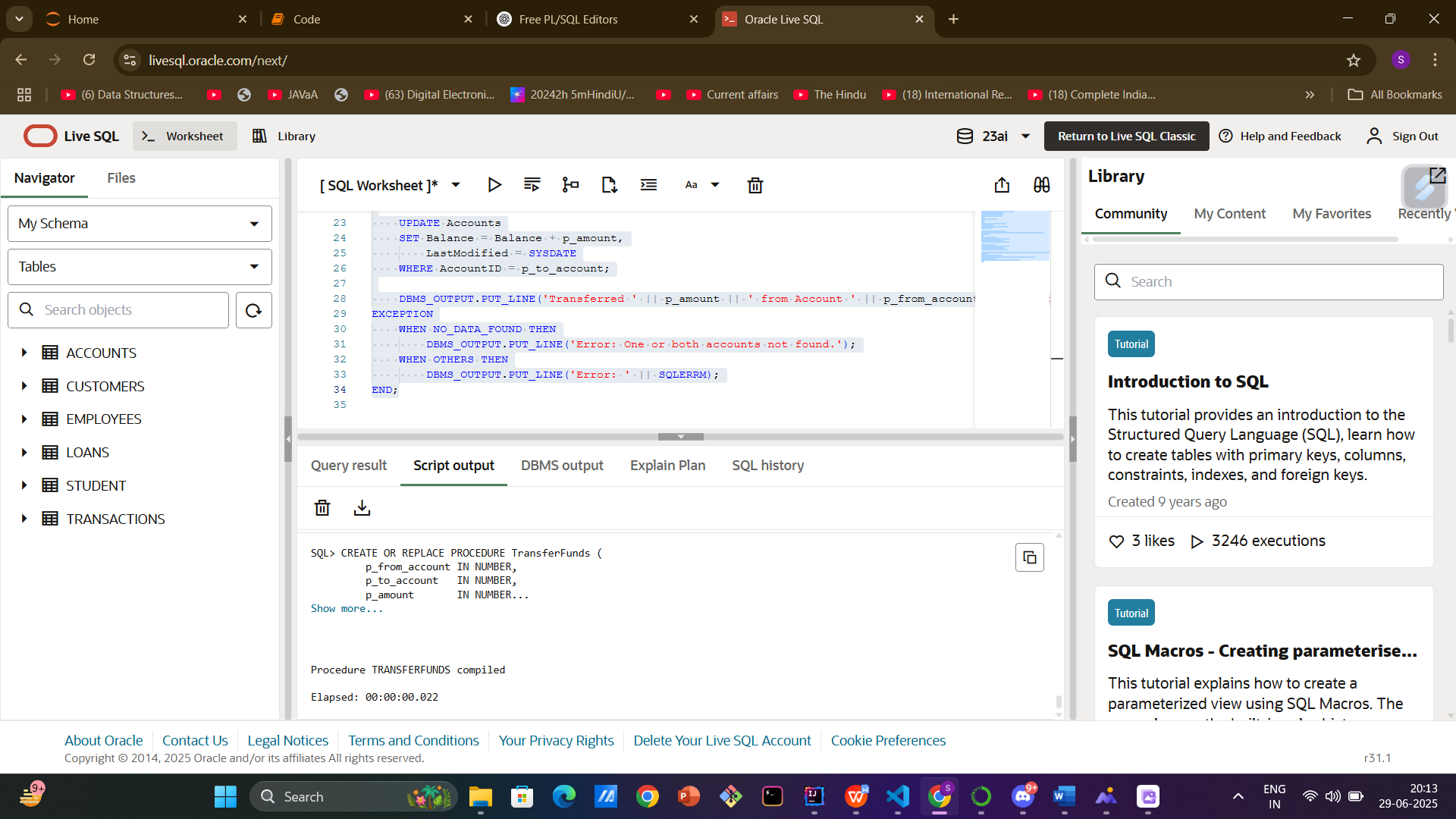
DBMS\_OUTPUT.PUT\_LINE('Error: One or both accounts not found.');

WHEN OTHERS THEN

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

END;

/**OUTPUT:**

****