**Exercise 5: Triggers**

**Scenario 1:** Automatically update the last modified date when a customer's record is updated.

* + **Question:** Write a trigger **UpdateCustomerLastModified** that updates the LastModified column of the Customers table to the current date whenever a customer's record is updated.

**Scenario 2:** Maintain an audit log for all transactions.

* + **Question:** Write a trigger **LogTransaction** that inserts a record into an AuditLog table whenever a transaction is inserted into the Transactions table.

**Scenario 3:** Enforce business rules on deposits and withdrawals.

* + **Question:** Write a trigger **CheckTransactionRules** that ensures withdrawals do not exceed the balance and deposits are positive before inserting a record into the Transactions table.

**Scenario 1: Automatically update the last modified date**

**🔹 Requirement:**

Update the LastModified column in the Customers table every time a record is **updated**.

**Trigger: UpdateCustomerLastModified**

**CODE:-**

CREATE OR REPLACE TRIGGER UpdateCustomerLastModified

BEFORE UPDATE ON Customers

FOR EACH ROW

BEGIN

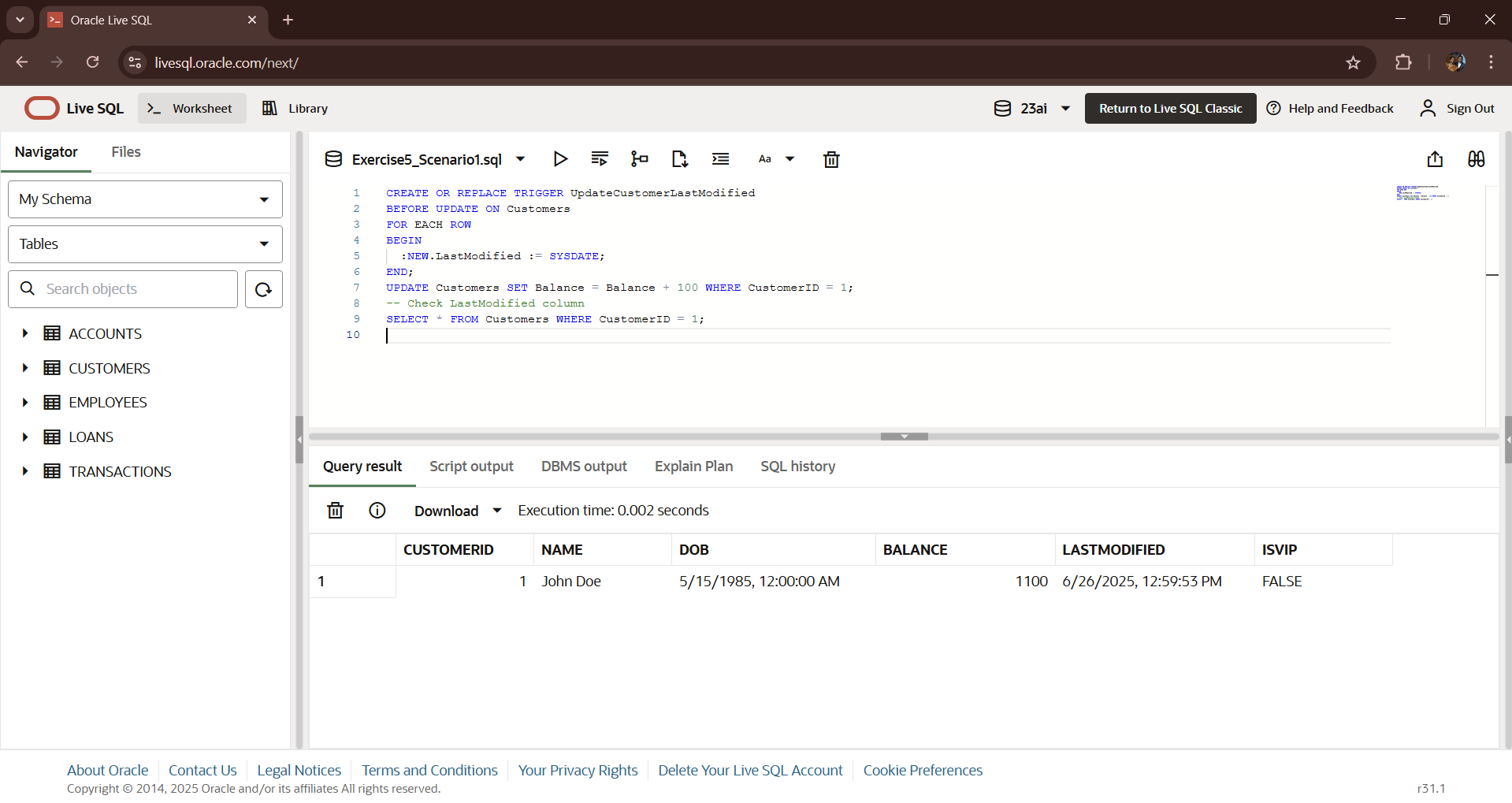
  :NEW.LastModified := SYSDATE;

END;

UPDATE Customers SET Balance = Balance + 100 WHERE CustomerID = 1;

-- Check LastModified column

SELECT \* FROM Customers WHERE CustomerID = 1;

**OUTPUT:-**

**Scenario 2: LogTransaction — Insert into AuditLog on New Transaction**

**CODE:-**

CREATE TABLE AuditLog (

    LogID NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY,

    TransactionID NUMBER,

    AccountID NUMBER,

    TransactionDate DATE,

    Amount NUMBER,

    TransactionType VARCHAR2(10),

    LoggedAt DATE

);

CREATE OR REPLACE TRIGGER LogTransaction

AFTER INSERT ON Transactions

FOR EACH ROW

BEGIN

  INSERT INTO AuditLog (

    TransactionID, AccountID, TransactionDate, Amount, TransactionType, LoggedAt

  ) VALUES (

    :NEW.TransactionID, :NEW.AccountID, :NEW.TransactionDate,

    :NEW.Amount, :NEW.TransactionType, SYSDATE

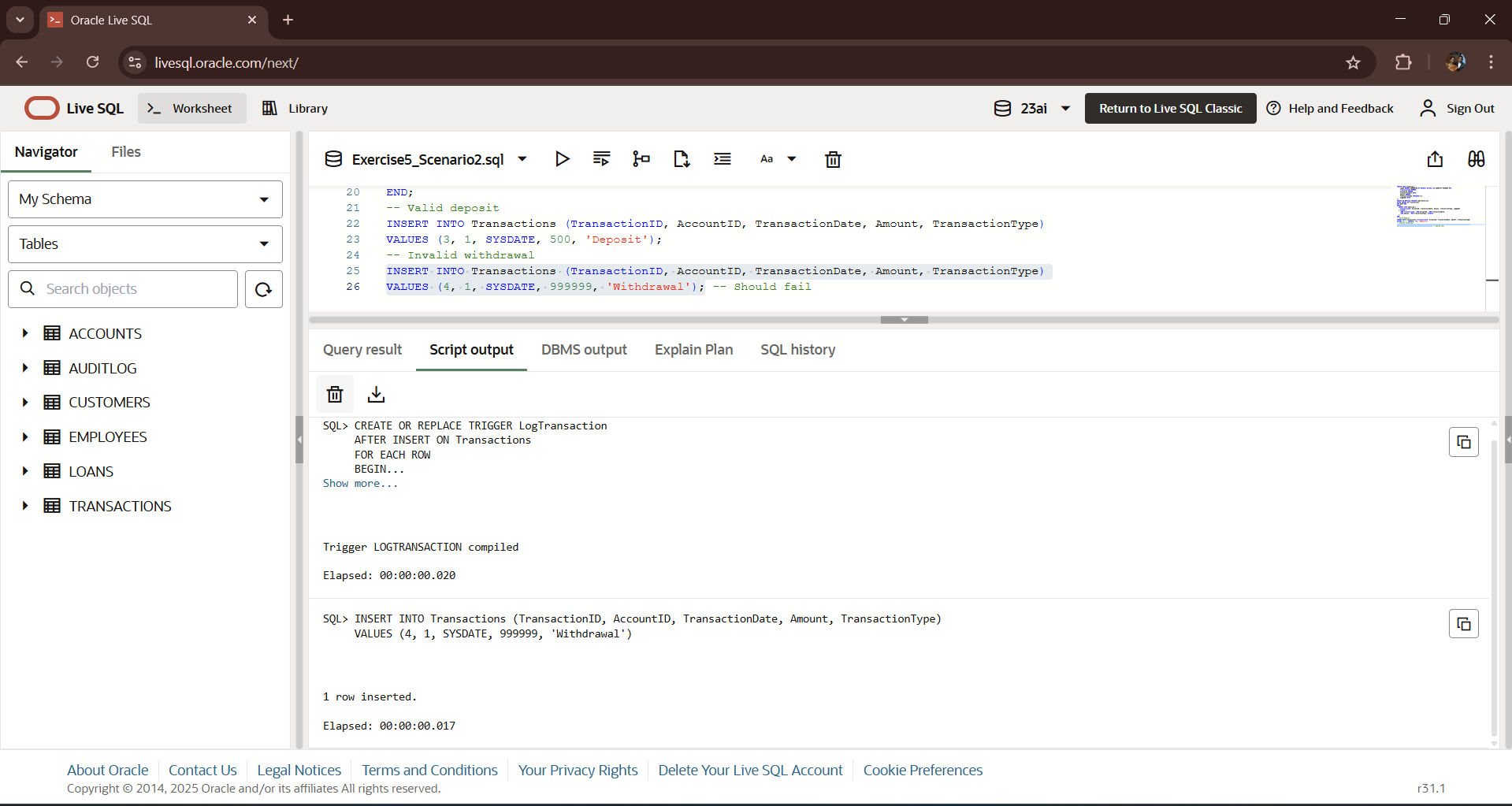
  );

END;

-- Valid deposit

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)

VALUES (3, 1, SYSDATE, 500, 'Deposit');

**OUTPUT:-**

**Scenario 3: CheckTransactionRules — Validate Transactions Before Insert**

Enforce:

* Withdrawal: Cannot exceed current balance.
* Deposit: Amount must be positive.

**CODE:-**

CREATE OR REPLACE TRIGGER CheckTransactionRules

BEFORE INSERT ON Transactions

FOR EACH ROW

DECLARE

  v\_Balance NUMBER;

BEGIN

  -- Get current balance of the account

  SELECT Balance INTO v\_Balance FROM Accounts WHERE AccountID = :NEW.AccountID;

  IF :NEW.TransactionType = 'Withdrawal' AND :NEW.Amount > v\_Balance THEN

    RAISE\_APPLICATION\_ERROR(-20001, 'Withdrawal amount exceeds available balance.');

  ELSIF :NEW.TransactionType = 'Deposit' AND :NEW.Amount <= 0 THEN

    RAISE\_APPLICATION\_ERROR(-20002, 'Deposit amount must be greater than 0.');

  END IF;

END;

-- Valid deposit

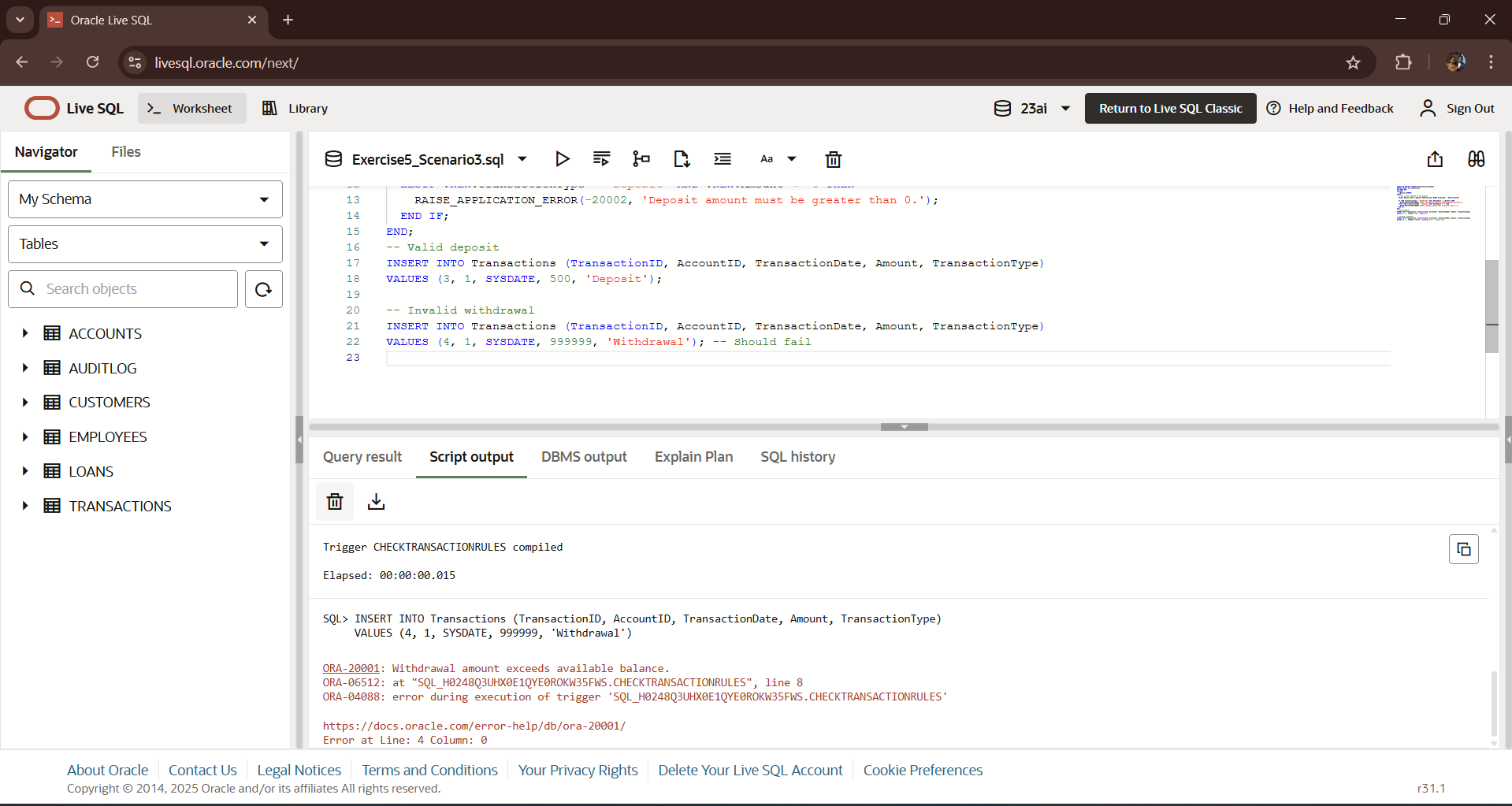
INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)

VALUES (3, 1, SYSDATE, 500, 'Deposit');

-- Invalid withdrawal

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)

VALUES (4, 1, SYSDATE, 999999, 'Withdrawal'); -- Should fail

**OUTPUT:-**