**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.

**Code :**

CREATE OR REPLACE TRIGGER UpdateCustomerLastModified

BEFORE UPDATE ON Customers

FOR EACH ROW

BEGIN

  :NEW.LastModified := SYSDATE;

END;

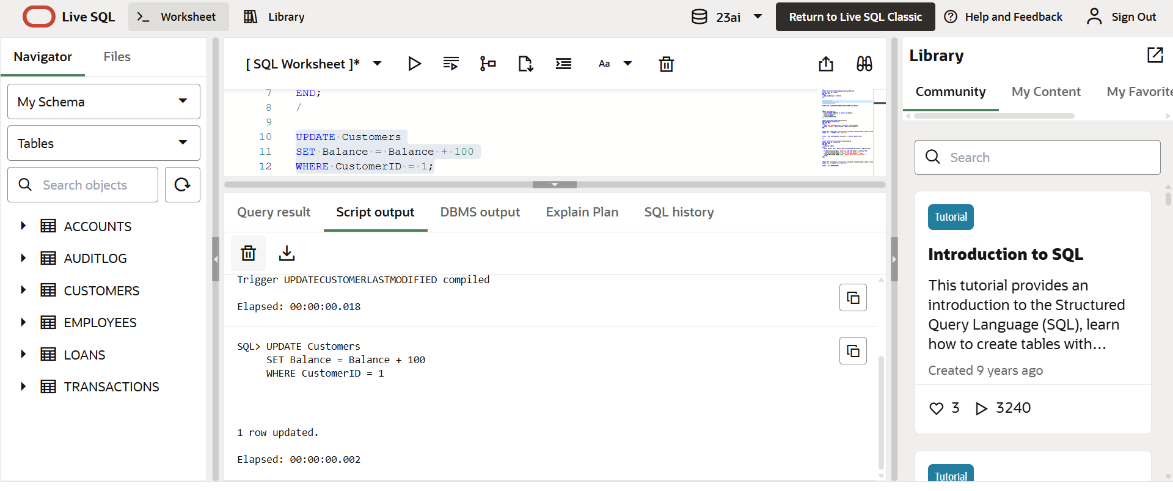
/

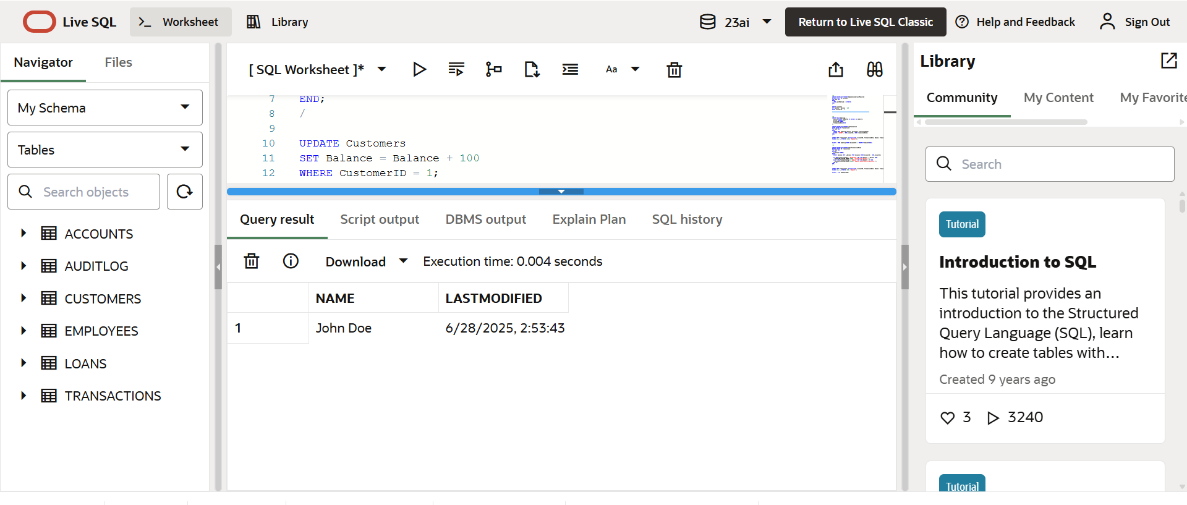
UPDATE Customers

SET Balance = Balance + 100

WHERE CustomerID = 1;

SELECT Name, LastModified FROM Customers WHERE CustomerID=1;

****

****

**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.

**Code :**

CREATE TABLE AuditLog (

  AuditID NUMBER GENERATED BY DEFAULT AS IDENTITY,

  Action VARCHAR2(50),

  AccountID NUMBER,

  TransactionDate DATE

);

CREATE OR REPLACE TRIGGER LogTransaction

AFTER INSERT ON Transactions

FOR EACH ROW

BEGIN

  INSERT INTO AuditLog (Action, AccountID, TransactionDate)

  VALUES ('INSERT', :NEW.AccountID, :NEW.TransactionDate);

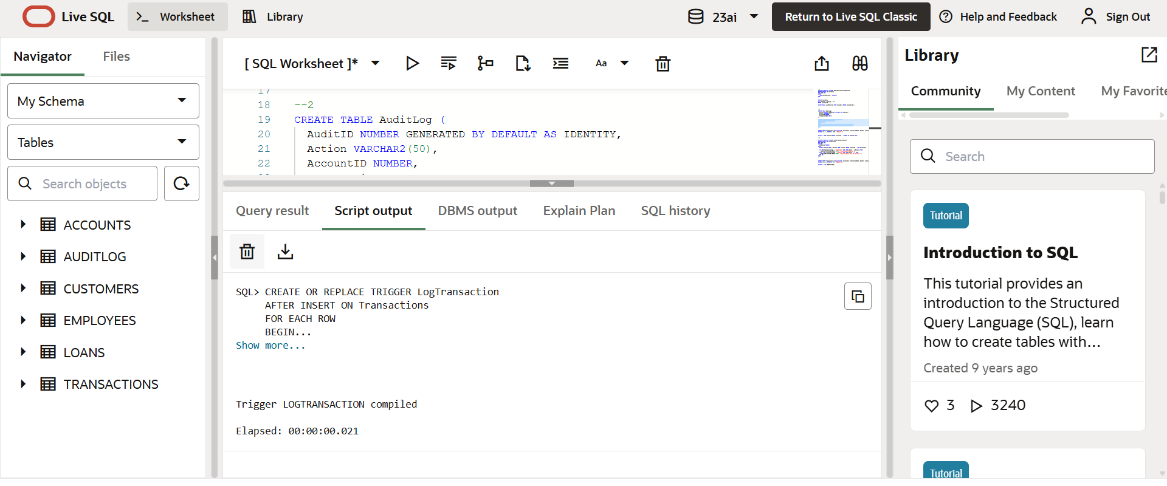
END;

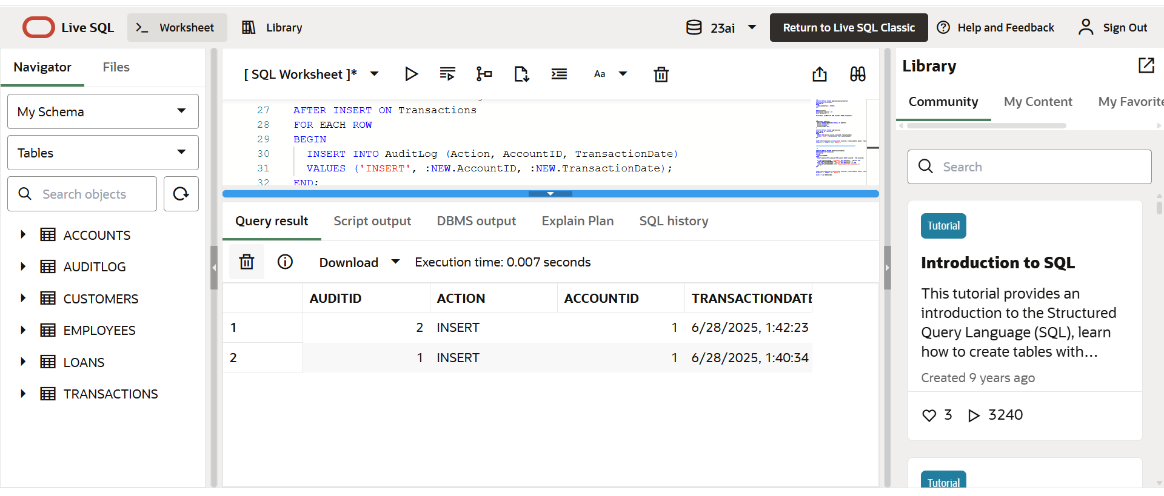
/

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

VALUES (21, 1, SYSDATE, 1000, 'Deposit');

SELECT \* FROM AuditLog WHERE AccountID = 1 ORDER BY AuditID DESC;



****

**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.

**Code :**

CREATE OR REPLACE TRIGGER CheckTransactionRules

BEFORE INSERT ON Transactions

FOR EACH ROW

DECLARE

  v\_Balance NUMBER;

BEGIN

  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, 'Insufficient balance.');

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

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

  END IF;

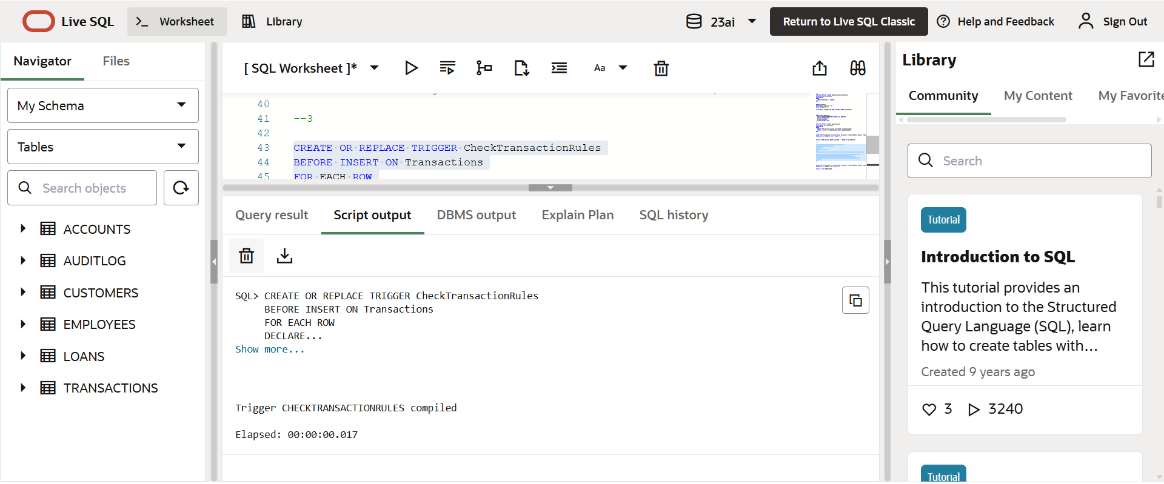
END;

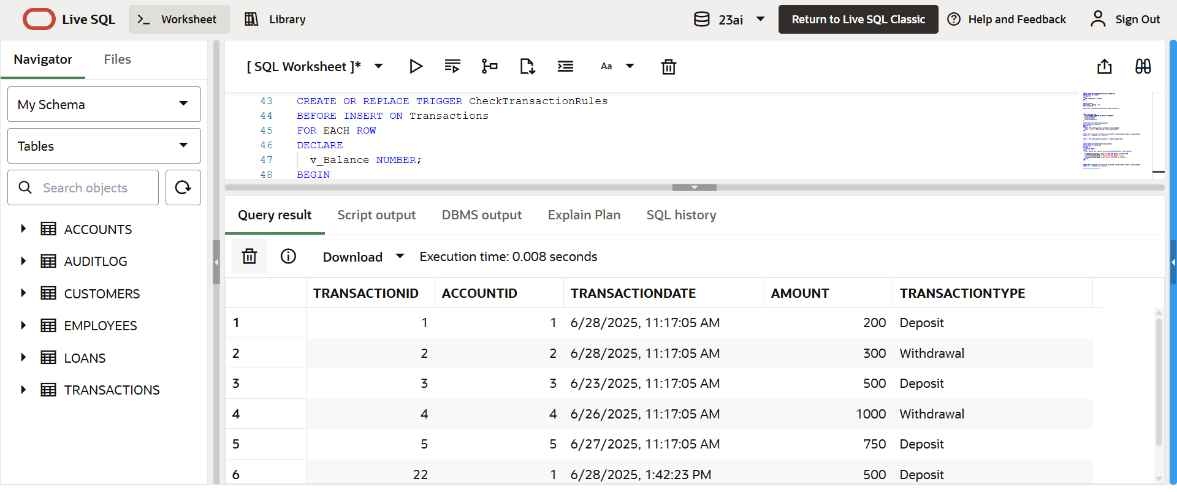
/

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

VALUES (22, 1, SYSDATE, 500, 'Deposit');

SELECT \* from TRANSACTIONS;

****

****