**TRIGGERS.**

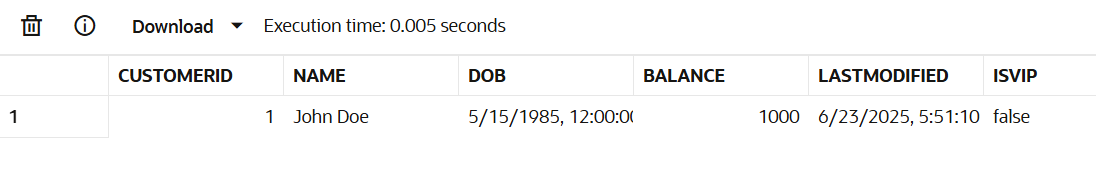
**Scenario 1:** Automatically update the last modified date when a customer's record is updated.

* + **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.

**SOLUTION:**

**Query:** select \* from customers where customerid=1;

**Output:**



**Query:**

create or replace trigger UpdateCustomerLastModified

before update on customers

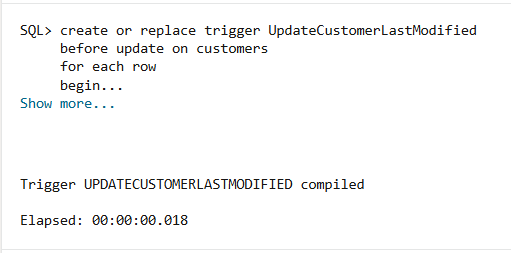
for each row

begin

:new.lastmodified:=sysdate;

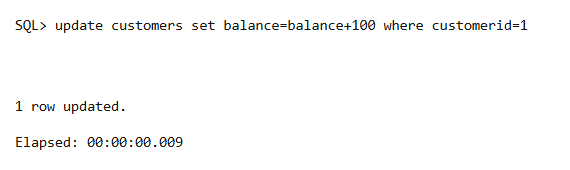
end;

**Output:**



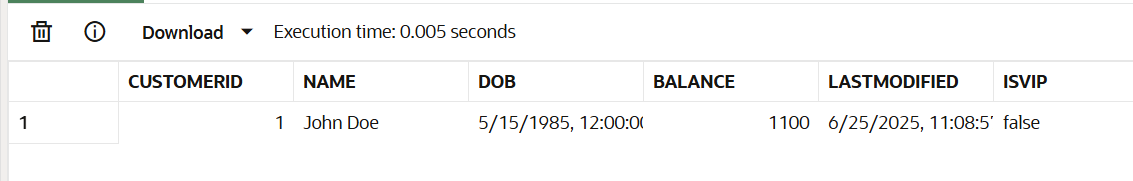
**Query:** update customers set balance=balance+100 where customerid=1;

**Output:**



**Query:** select \* from customers where customerid=1;

**Output:**



**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.

**SOLUTION:**

**Query:**

CREATE TABLE AuditLog (

    logid number generated always as identity primary key,

    TransactionID NUMBER,

    TransactionDate DATE,

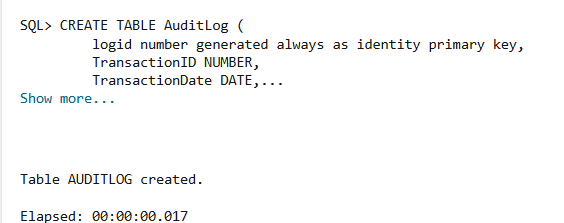
    Amount NUMBER,

    TransactionType VARCHAR2(10),

    changedby varchar2(20)

);

**Output:**



**Query:**

create or replace trigger LogTransaction

after insert on transactions

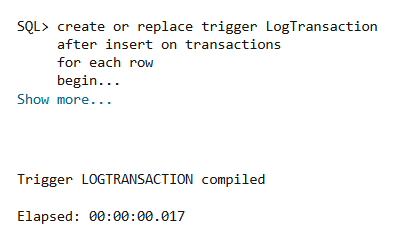
for each row

begin

insert into auditlog(transactionid,transactiondate,amount,transactiontype,changedby) values (:new.transactionid,:new.transactionDate,:new.amount,:new.Transactiontype,'USER');

end;

**Output:**

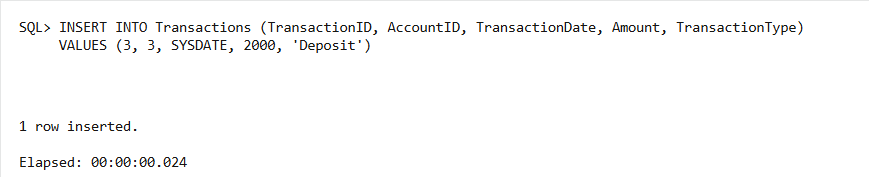


**Query:**

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

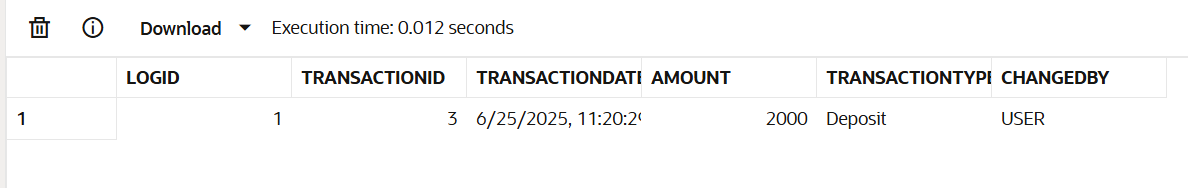
VALUES (3, 3, SYSDATE, 2000, 'Deposit');

**Output:**



**Query:** select \* from auditlog;

**Output:**



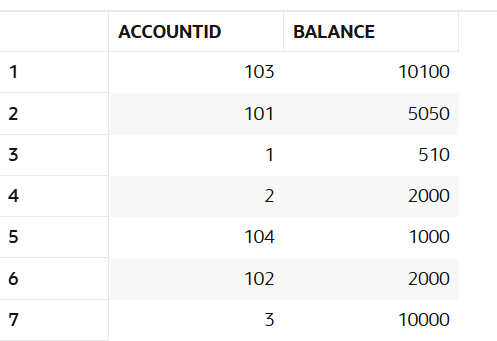
**Scenario 3:** Enforce business rules on deposits and withdrawals.

* + **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.

**SOLUTION:**

**Query**: select accountid,balance from accounts;

**Output:**



**Query:**

CREATE OR REPLACE TRIGGER CheckTransactionRules

BEFORE INSERT ON Transactions

FOR EACH ROW

DECLARE

    acc\_balance NUMBER;

BEGIN

    SELECT balance INTO acc\_balance FROM Accounts WHERE accountid = :NEW.accountid;

    IF :NEW.transactiontype = 'WITHDRAW' THEN

        IF :NEW.amount > acc\_balance THEN

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

        END IF;

    ELSIF :NEW.transactiontype = 'DEPOSIT' THEN

        IF :NEW.amount <= 0 THEN

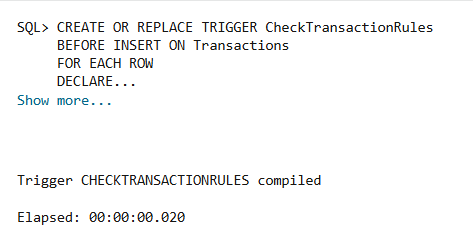
            RAISE\_APPLICATION\_ERROR(-20002, 'Deposit amount must be positive.');

        END IF;

    END IF;

END;

**Output:**

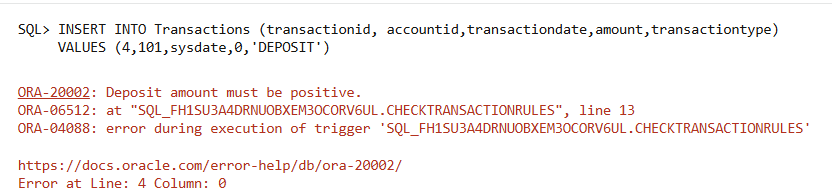


**Query:**

INSERT INTO Transactions (transactionid, accountid,transactiondate,amount,transactiontype)

VALUES (4,101,sysdate,0,'DEPOSIT');

**Output:**

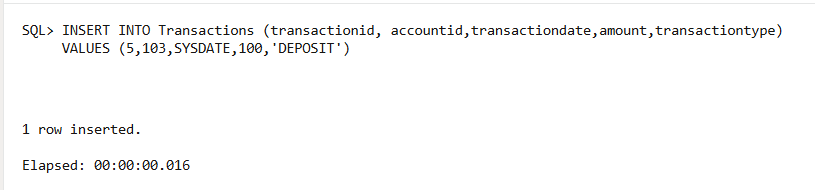


**Query:**

INSERT INTO Transactions (transactionid, accountid,transactiondate,amount,transactiontype)

VALUES (5,103,SYSDATE,100,'DEPOSIT');

**Output:**



**Query:**

INSERT INTO Transactions (transactionid,accountid,transactiondate,amount,transactiontype)

VALUES (7, 1,sysdate,999999,'WITHDRAW');

**Output:**

