**Exercise 4: Functions**

**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.

**PL/SQL Block :-**

CREATE OR REPLACE TRIGGER UpdateCustomerLastModified

BEFORE UPDATE ON Customers

FOR EACH ROW

BEGIN

:NEW.LastModified := SYSDATE;

END;

/

-- Check current data

SELECT CustomerID, Name, LastModified FROM Customers;

-- Update a customer

UPDATE Customers

SET Balance = Balance + 100

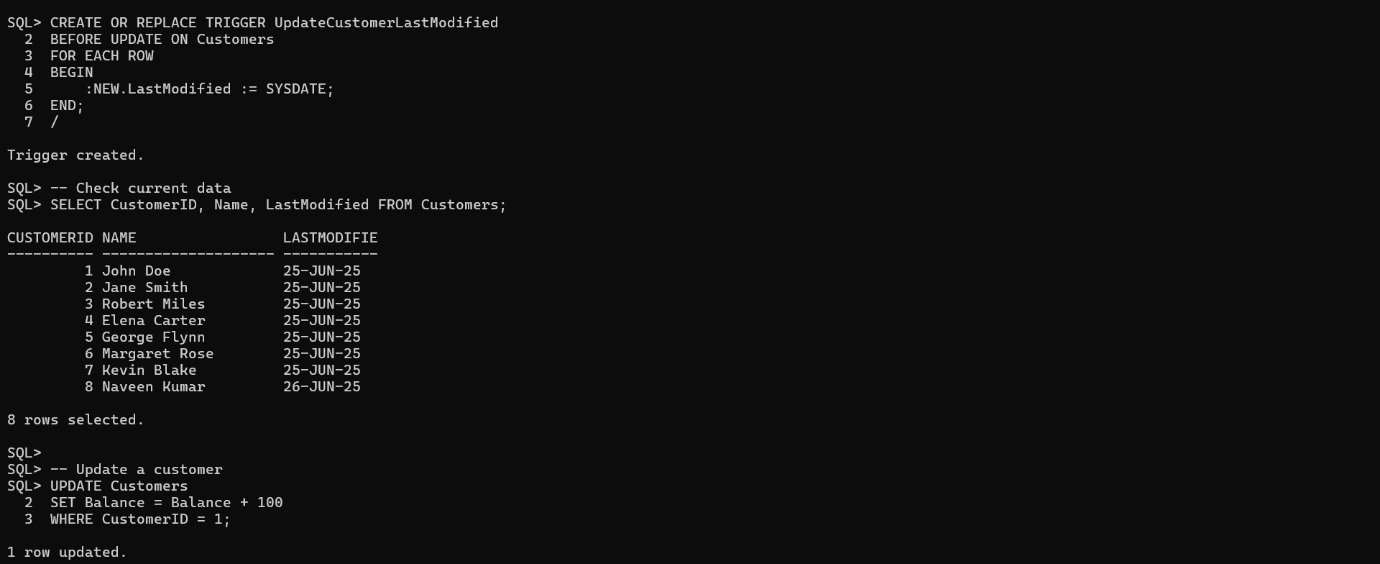
WHERE CustomerID = 1;

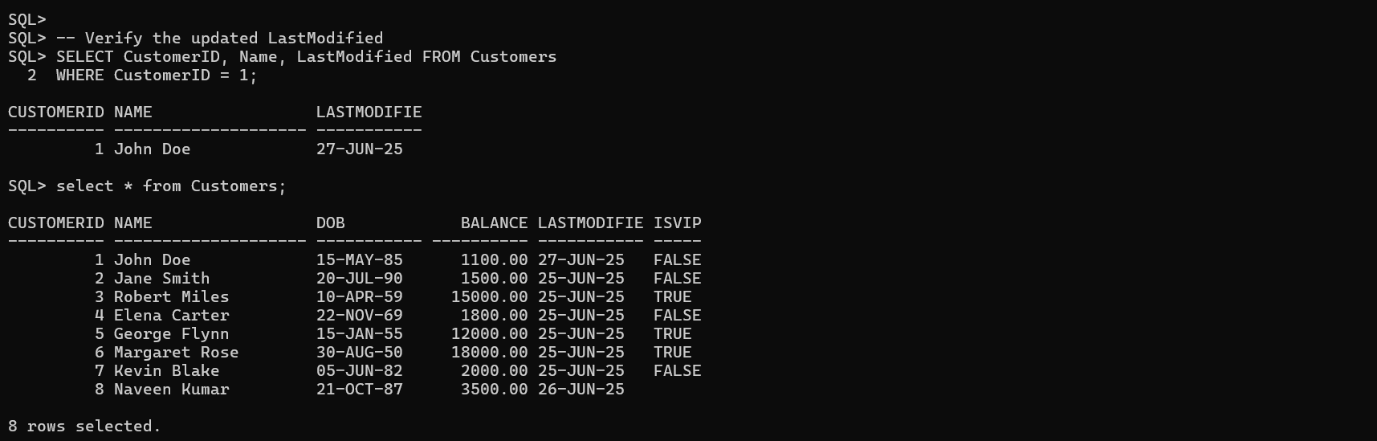
-- Verify the updated LastModified

SELECT CustomerID, Name, LastModified 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.

**PL/SQL Block :-**

CREATE TABLE AuditLog (

LogID NUMBER PRIMARY KEY,

TransactionID NUMBER,

AccountID NUMBER,

Action VARCHAR2(20),

Timestamp DATE

);

CREATE SEQUENCE AuditLog\_SEQ START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER LogTransaction

AFTER INSERT ON Transactions

FOR EACH ROW

BEGIN

INSERT INTO AuditLog (LogID,TransactionID, AccountID, Action, Timestamp)

VALUES (AuditLog\_SEQ.nextval,:NEW.TransactionID, :NEW.AccountID, 'INSERT', SYSDATE);

END;

/

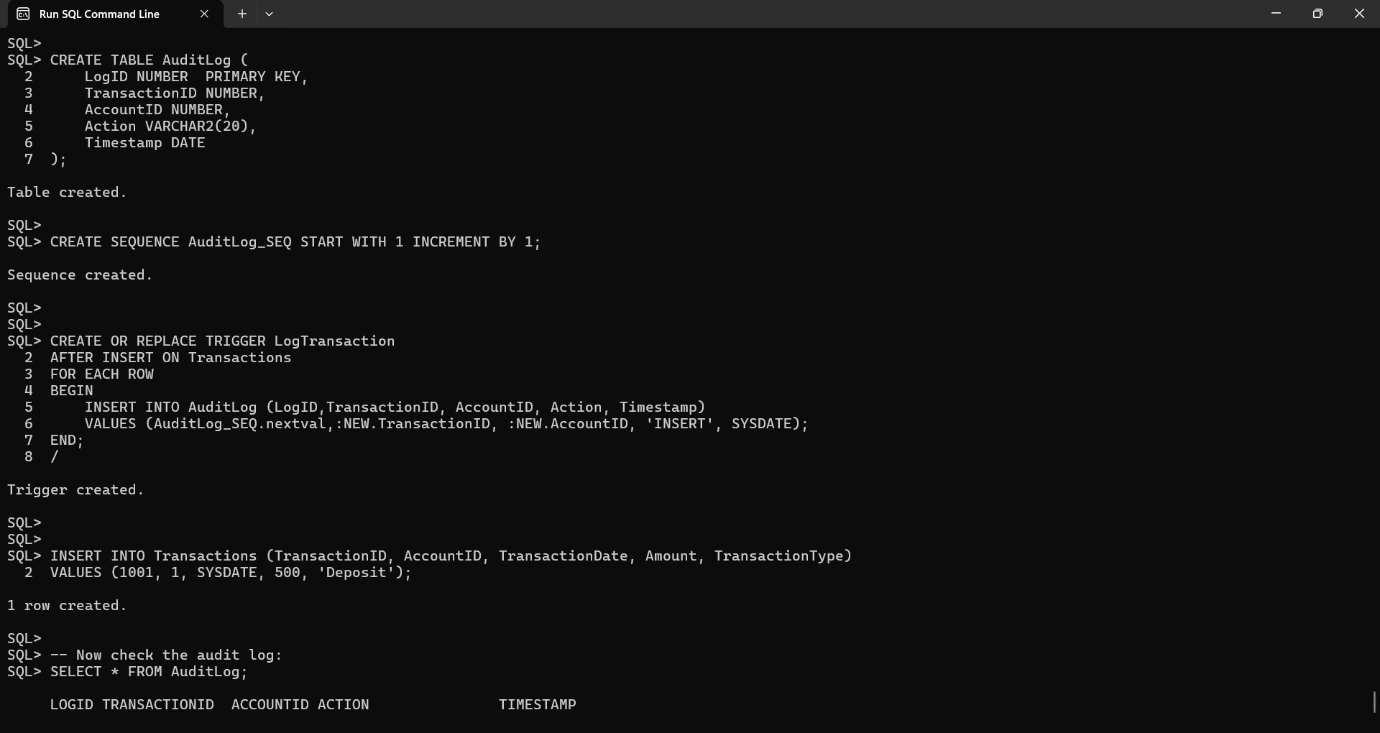
INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)

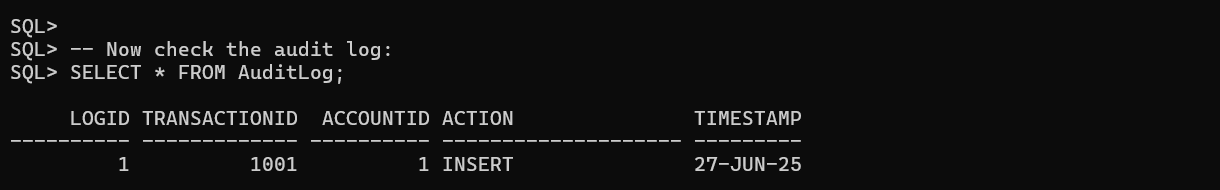
VALUES (1001, 1, SYSDATE, 500, 'Deposit');

-- Now check the audit log:

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.

**PL/SQL Block :-**

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;

-- Check for withdrawal

IF :NEW.TransactionType = 'Withdrawal' THEN

IF :NEW.Amount > v\_balance THEN

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

END IF;

END IF;

-- Check for positive deposit

IF :NEW.TransactionType = 'Deposit' THEN

IF :NEW.Amount <= 0 THEN

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

END IF;

END IF;

END;

/

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

VALUES (1003, 1, SYSDATE, 200, 'Deposit');

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

VALUES (1004, 1, SYSDATE, -100, 'Deposit');

**Output :-**

