**Exercise 1: Control Structures**

**Initial Schema Given**

CREATE TABLE Customers (

    CustomerID NUMBER PRIMARY KEY,

    Name VARCHAR2(100),

    DOB DATE,

    Balance NUMBER,

    LastModified DATE

);

CREATE TABLE Accounts (

    AccountID NUMBER PRIMARY KEY,

    CustomerID NUMBER,

    AccountType VARCHAR2(20),

    Balance NUMBER,

    LastModified DATE,

    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE Transactions (

    TransactionID NUMBER PRIMARY KEY,

    AccountID NUMBER,

    TransactionDate DATE,

    Amount NUMBER,

    TransactionType VARCHAR2(10),

    FOREIGN KEY (AccountID) REFERENCES Accounts(AccountID)

);

CREATE TABLE Loans (

    LoanID NUMBER PRIMARY KEY,

    CustomerID NUMBER,

    LoanAmount NUMBER,

    InterestRate NUMBER,

    StartDate DATE,

    EndDate DATE,

    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE Employees (

    EmployeeID NUMBER PRIMARY KEY,

    Name VARCHAR2(100),

    Position VARCHAR2(50),

    Salary NUMBER,

    Department VARCHAR2(50),

    HireDate DATE

);

**Values inserted Initially**

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)

VALUES (1, 'John Doe', TO\_DATE('1985-05-15', 'YYYY-MM-DD'), 1000, SYSDATE);

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)

VALUES (2, 'Jane Smith', TO\_DATE('1990-07-20', 'YYYY-MM-DD'), 1500, SYSDATE);

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)

VALUES (3, 'Nakka Girish', TO\_DATE('1960-12-20', 'YYYY-MM-DD'), 25000, SYSDATE);

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)

VALUES (1, 1, 'Savings', 1000, SYSDATE);

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)

VALUES (2, 2, 'Checking', 15000, SYSDATE);

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)

VALUES (3, 3, 'Savings', 25000, SYSDATE);

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

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

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

VALUES (2, 2, SYSDATE, 300, 'Withdrawal');

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

VALUES (3, 3, SYSDATE, 500, 'Withdrawal');

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate)

VALUES (1, 1, 5000, 5, SYSDATE, ADD\_MONTHS(SYSDATE, 60));

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate)

VALUES (2, 3, 6000, 5, SYSDATE, ADD\_MONTHS(SYSDATE, 90));

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)

VALUES (1, 'Alice Johnson', 'Manager', 70000, 'HR', TO\_DATE('2015-06-15', 'YYYY-MM-DD'));

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)

VALUES (2, 'Bob Brown', 'Developer', 60000, 'IT', TO\_DATE('2017-03-20', 'YYYY-MM-DD'));

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)

VALUES (3, 'Nakka Girish', 'Java\_FSE', 90000, 'IT', TO\_DATE('2024-08-20', 'YYYY-MM-DD'));

Table values Initially

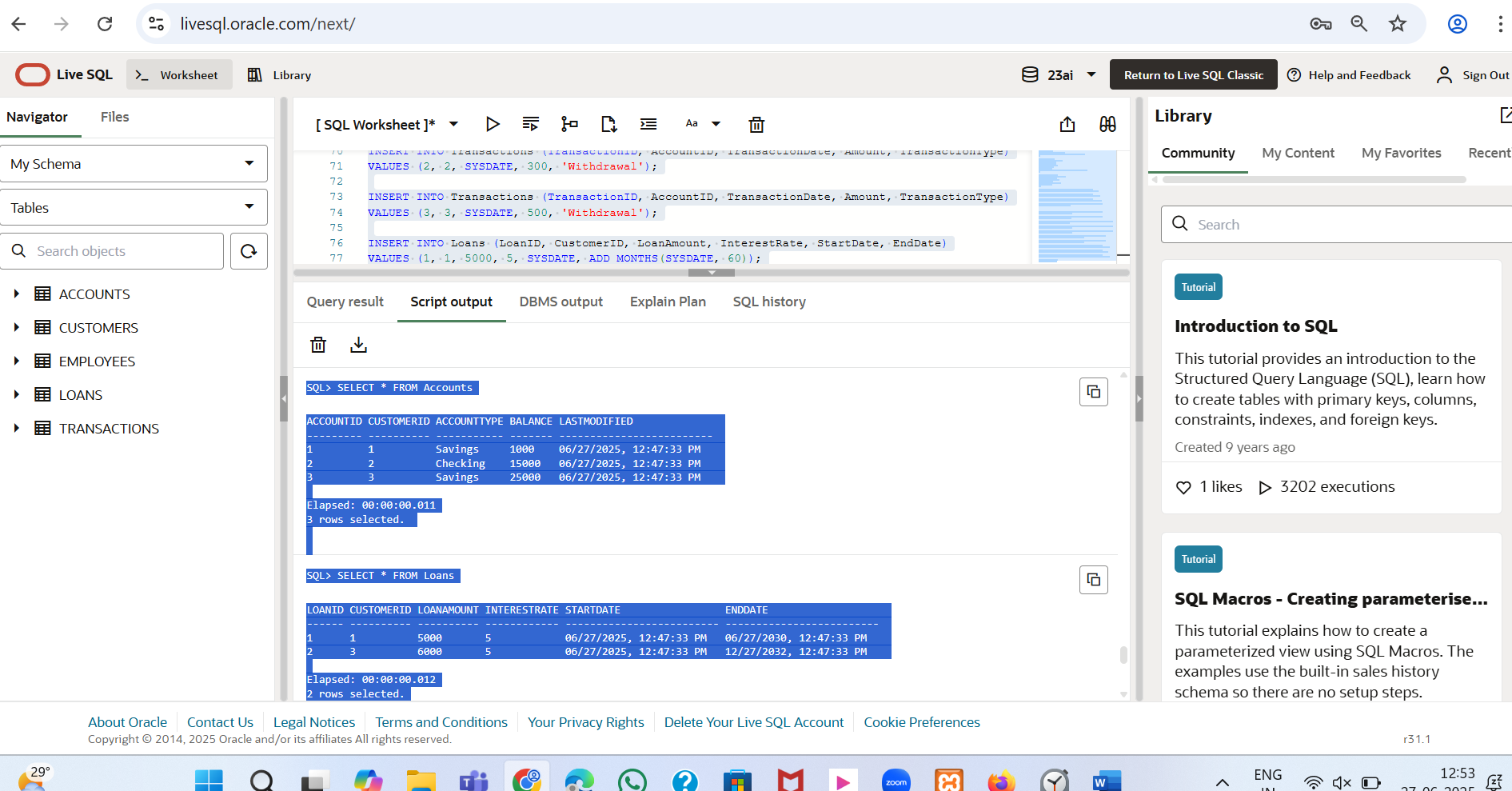
SELECT \* FROM Customers;

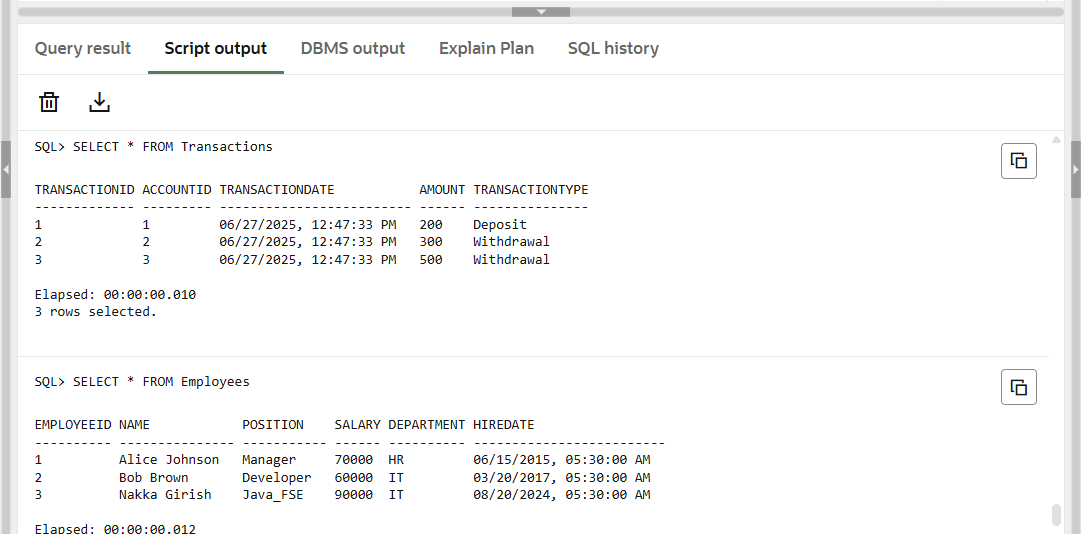
SELECT \* FROM Accounts;

SELECT \* FROM Loans;

SELECT \* FROM Transactions;

SELECT \* FROM Employees;

****

****

**Scenario 1:** The bank wants to apply a discount to loan interest rates for customers above 60 years old.

* + **Question:** Write a PL/SQL block that loops through all customers, checks their age, and if they are above 60, apply a 1% discount to their current loan interest rates.

BEGIN

  FOR c IN (SELECT CustomerID, Name, DOB FROM Customers) LOOP

    IF MONTHS\_BETWEEN(SYSDATE, c.DOB)/12 > 60 THEN

      UPDATE Loans

      SET InterestRate = InterestRate - (InterestRate \* 0.01)

      WHERE CustomerID = c.CustomerID;

      DBMS\_OUTPUT.PUT\_LINE(

        'Interest rate discounted for Customer ID: ' || c.CustomerID ||

        ', Name: ' || c.Name || ', DOB: ' || TO\_CHAR(c.DOB, 'YYYY-MM-DD')

      );

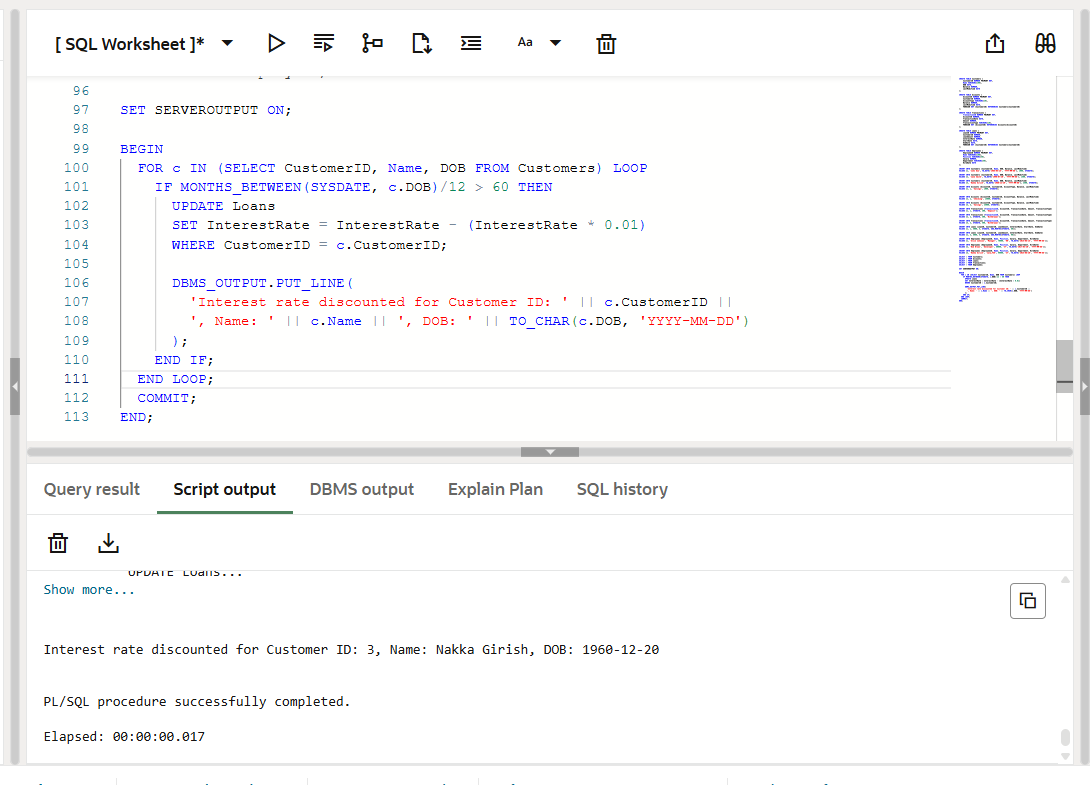
    END IF;

  END LOOP;

  COMMIT;

END;

**Output:**

****

**Scenario 2:** A customer can be promoted to VIP status based on their balance.

* + **Question:** Write a PL/SQL block that iterates through all customers and sets a flag IsVIP to TRUE for those with a balance over $10,000.

ALTER TABLE Customers ADD IsVIP CHAR(1);

BEGIN

  FOR c IN (SELECT CustomerID,Name ,Balance FROM Customers) LOOP

    IF c.Balance > 10000 THEN

      UPDATE Customers

      SET IsVIP = 'T'

      WHERE CustomerID = c.CustomerID;

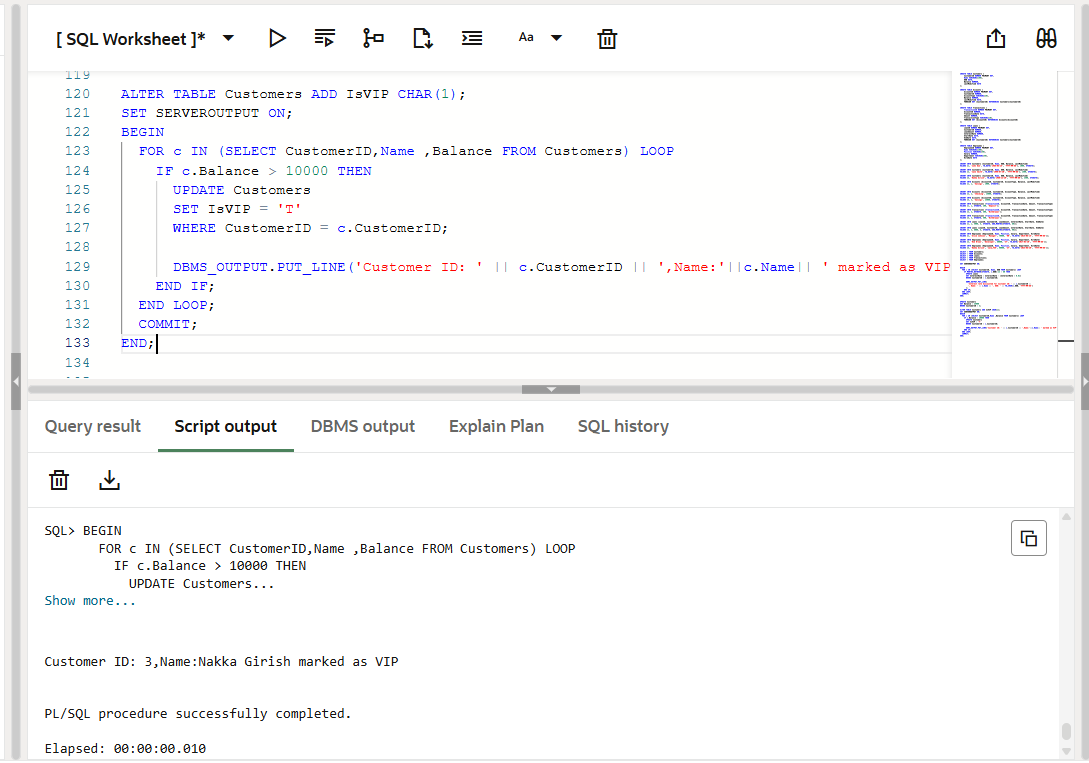
      DBMS\_OUTPUT.PUT\_LINE('Customer ID: ' || c.CustomerID ||',Name:'||c.Name|| ' marked as VIP');

    END IF;

  END LOOP;

  COMMIT;

END;

Output:

**Scenario 3:** The bank wants to send reminders to customers whose loans are due within the next 30 days.

* + **Question:** Write a PL/SQL block that fetches all loans due in the next 30 days and prints a reminder message for each customer.

BEGIN

  FOR g IN (

    SELECT l.LoanID, l.CustomerID, c.Name, l.EndDate

    FROM Loans l

    JOIN Customers c ON l.CustomerID = c.CustomerID

    WHERE l.EndDate BETWEEN SYSDATE AND SYSDATE + 30

  ) LOOP

    DBMS\_OUTPUT.PUT\_LINE(

      'Reminder: Dear ' || g.Name ||

      ', your loan (ID: ' || g.LoanID ||

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

    );

  END LOOP;

END;

**Output:**

