**MODULE : PL/SQL PROGRAMMING**

**Task 1: Control Structures**

**Control Structures.sql (File Name )**

**Query:**

-- Create customers table

CREATE TABLE customers (

  customer\_id   NUMBER PRIMARY KEY,

  name          VARCHAR2(50),

  age           NUMBER,

  balance       NUMBER,

  IsVIP         VARCHAR2(5)

);

-- Create loans table

CREATE TABLE loans (

  loan\_id        NUMBER PRIMARY KEY,

  customer\_id    NUMBER,

  interest\_rate  NUMBER,

  due\_date       DATE,

  FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id)

);

-- Insert sample customers

INSERT INTO customers VALUES (1, 'John Doe', 65, 8000, 'FALSE');

INSERT INTO customers VALUES (2, 'Jane Smith', 45, 12000, 'FALSE');

INSERT INTO customers VALUES (3, 'Emily Green', 70, 15000, 'FALSE');

-- Insert sample loans

INSERT INTO loans VALUES (101, 1, 10.0, SYSDATE + 15); -- due soon

INSERT INTO loans VALUES (102, 2, 9.5, SYSDATE + 40);  -- not due soon

INSERT INTO loans VALUES (103, 3, 11.0, SYSDATE + 20); -- due soon

COMMIT;

BEGIN

  FOR rec IN (

    SELECT c.customer\_id, l.loan\_id

    FROM customers c

    JOIN loans l ON c.customer\_id = l.customer\_id

    WHERE c.age > 60

  ) LOOP

    UPDATE loans

    SET interest\_rate = interest\_rate - 1

    WHERE loan\_id = rec.loan\_id;

  END LOOP;

  COMMIT;

END;

BEGIN

  FOR rec IN (

    SELECT customer\_id

    FROM customers

    WHERE balance > 10000

  ) LOOP

    UPDATE customers

    SET IsVIP = 'TRUE'

    WHERE customer\_id = rec.customer\_id;

  END LOOP;

  COMMIT;

END;

SET SERVEROUTPUT ON;

BEGIN

  FOR rec IN (

    SELECT c.name, l.due\_date

    FROM customers c

    JOIN loans l ON c.customer\_id = l.customer\_id

    WHERE l.due\_date BETWEEN SYSDATE AND SYSDATE + 30

  ) LOOP

    DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan for ' || rec.name || ' is due on ' || TO\_CHAR(rec.due\_date, 'DD-MON-YYYY'));

  END LOOP;

END;

**OUTPUT:**

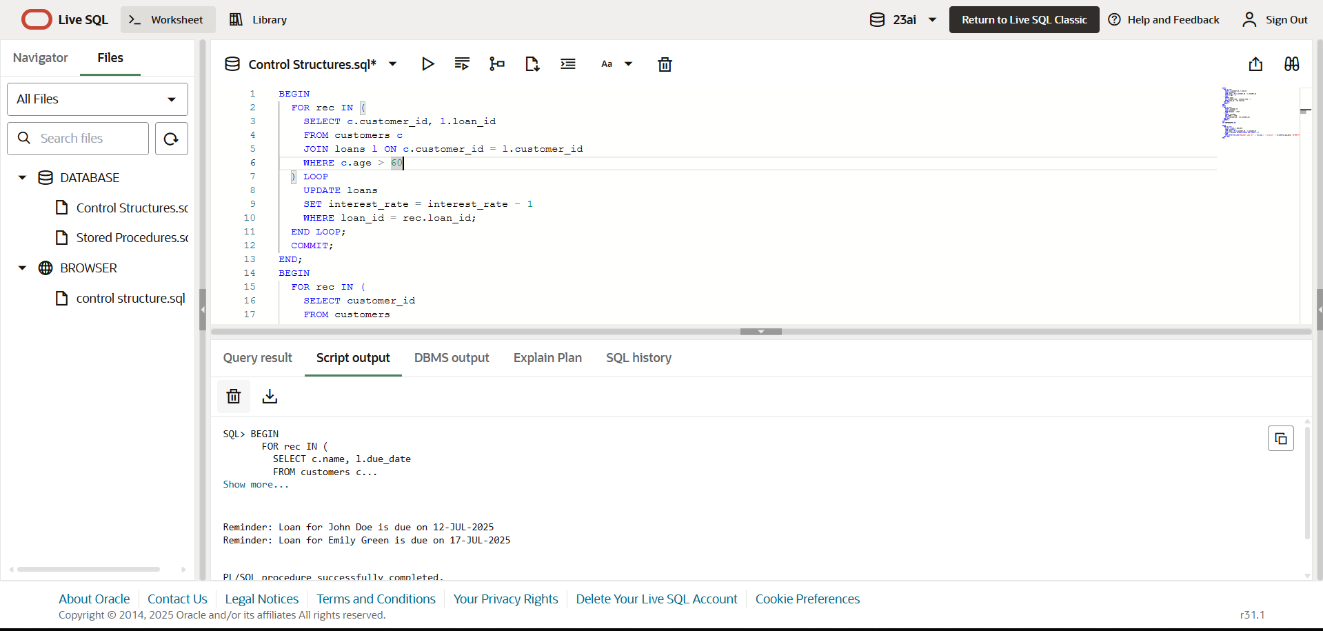
****

Fig 1.Table Creation.

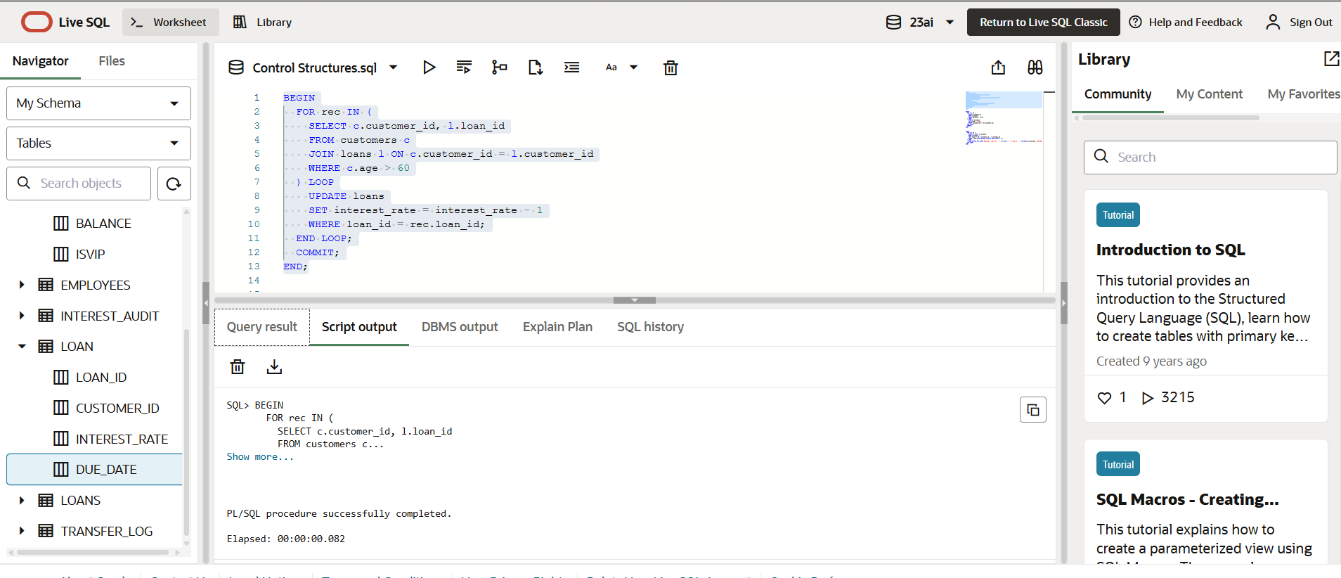


Fig 2. The bank wants to apply a discount to loan interest rates for customers above 60 years old.

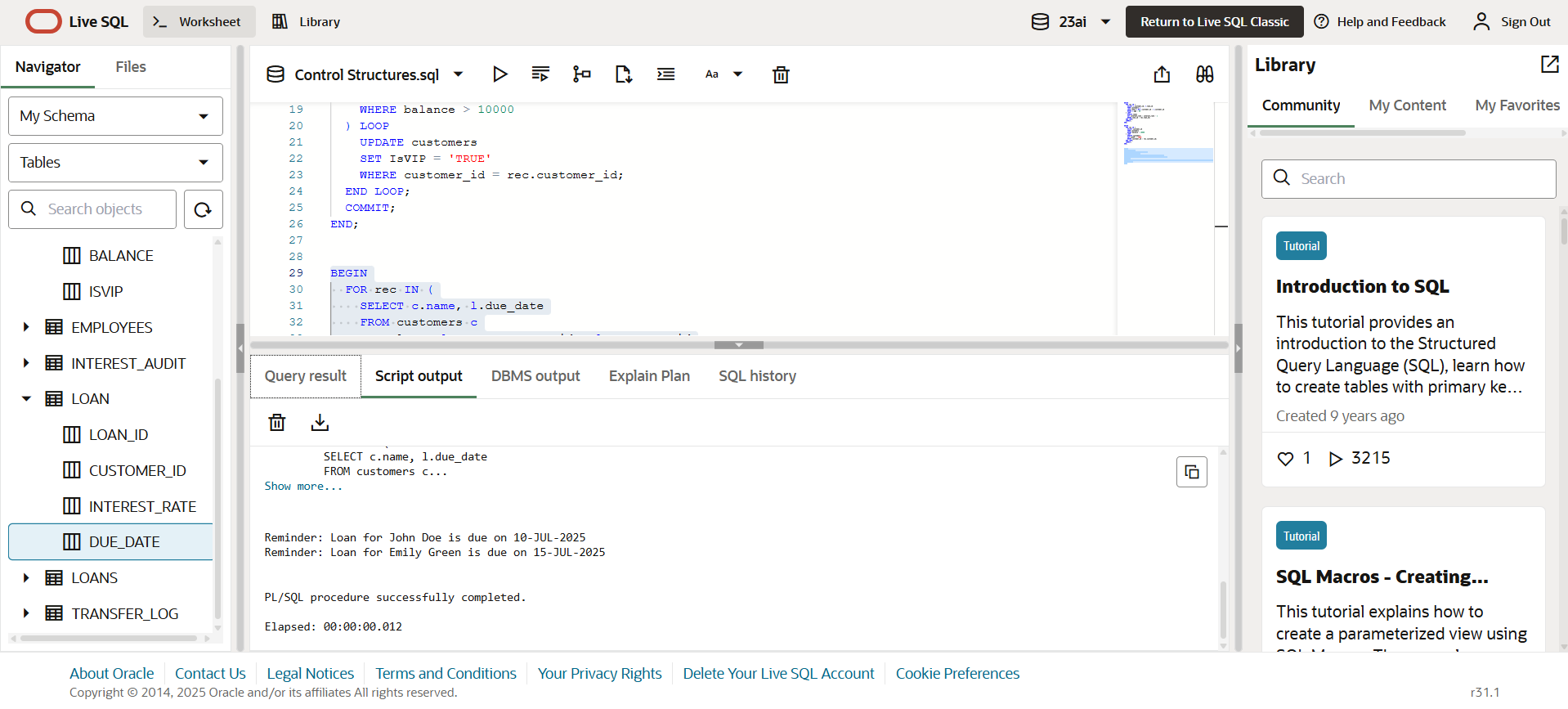


Fig . The bank wants to send reminders to customers whose loans are due within the next 30 days.

------END-----

**Task 3: Stored Procedures**

**Stored Procedures.sql (File Name )**

**Query:**

-- Drop old tables if they exist

BEGIN EXECUTE IMMEDIATE 'DROP TABLE accounts'; EXCEPTION WHEN OTHERS THEN NULL; END;

/

BEGIN EXECUTE IMMEDIATE 'DROP TABLE employees'; EXCEPTION WHEN OTHERS THEN NULL; END;

/

-- Create accounts table

CREATE TABLE accounts (

  account\_id   NUMBER PRIMARY KEY,

  customer\_id  NUMBER,

  account\_type VARCHAR2(20),

  balance      NUMBER

);

-- Create employees table

CREATE TABLE employees (

  emp\_id        NUMBER PRIMARY KEY,

  name          VARCHAR2(50),

  department\_id NUMBER,

  salary        NUMBER

);

-- Insert sample accounts

INSERT INTO accounts VALUES (1001, 1, 'savings', 5000);

INSERT INTO accounts VALUES (1002, 1, 'savings', 7000);

INSERT INTO accounts VALUES (1003, 2, 'current', 12000);

-- Insert sample employees with new names

INSERT INTO employees VALUES (1, 'Riya', 101, 50000);

INSERT INTO employees VALUES (2, 'Karan', 101, 45000);

INSERT INTO employees VALUES (3, 'Sneha', 102, 60000);

COMMIT;

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

  UPDATE accounts

  SET balance = balance + (balance \* 0.01)

  WHERE account\_type = 'savings';

  COMMIT;

END;

/

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

  p\_department\_id IN NUMBER,

  p\_bonus\_percent IN NUMBER

) AS

BEGIN

  UPDATE employees

  SET salary = salary + (salary \* p\_bonus\_percent / 100)

  WHERE department\_id = p\_department\_id;

  COMMIT;

END;

/

CREATE OR REPLACE PROCEDURE TransferFunds (

  p\_from\_account IN NUMBER,

  p\_to\_account   IN NUMBER,

  p\_amount       IN NUMBER

) AS

  v\_balance NUMBER;

BEGIN

  -- Get current balance

  SELECT balance INTO v\_balance

  FROM accounts

  WHERE account\_id = p\_from\_account;

  -- Check if sufficient

  IF v\_balance < p\_amount THEN

    RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient balance.');

  END IF;

  -- Deduct from sender

  UPDATE accounts

  SET balance = balance - p\_amount

  WHERE account\_id = p\_from\_account;

  -- Credit to receiver

  UPDATE accounts

  SET balance = balance + p\_amount

  WHERE account\_id = p\_to\_account;

  COMMIT;

END;

/

**OUTPUT:**

