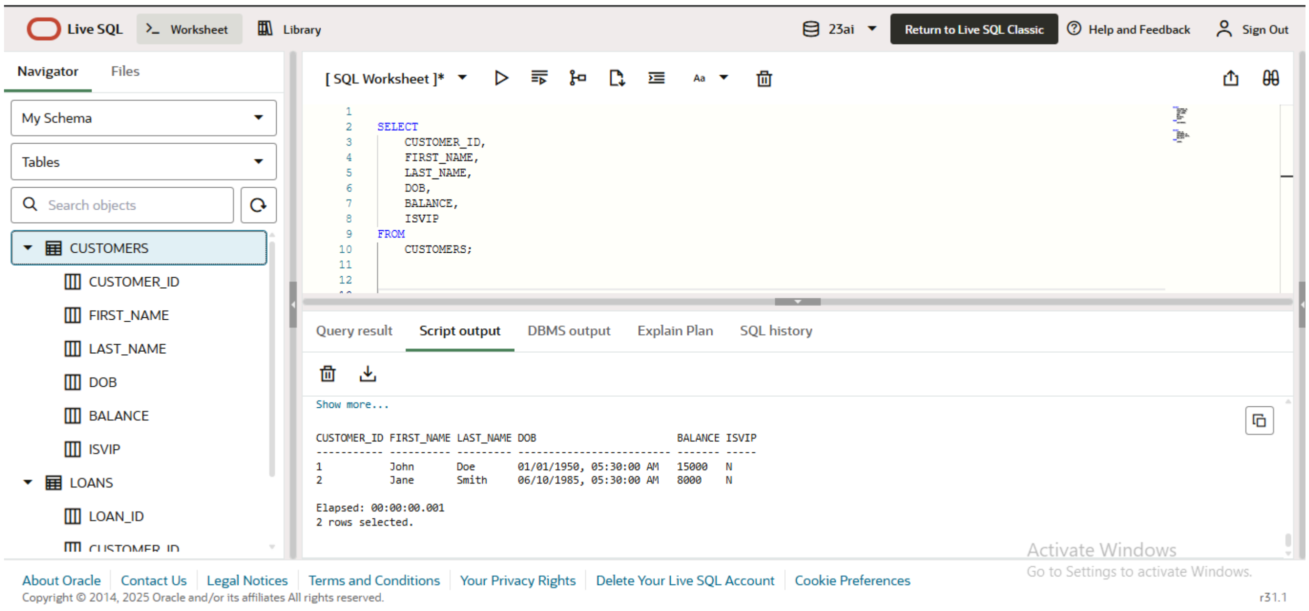
QUESTION 1:-

Creation of **CUSTOMER** Table

```
INSERT INTO customers VALUES (1, 'John', 'Doe', TO_DATE('1950-01-01','YYYY-MM-DD'), 15000, 'N');
INSERT INTO customers VALUES (2, 'Jane', 'Smith', TO_DATE('1985-06-10','YYYY-MM-DD'), 8000, 'N');
```



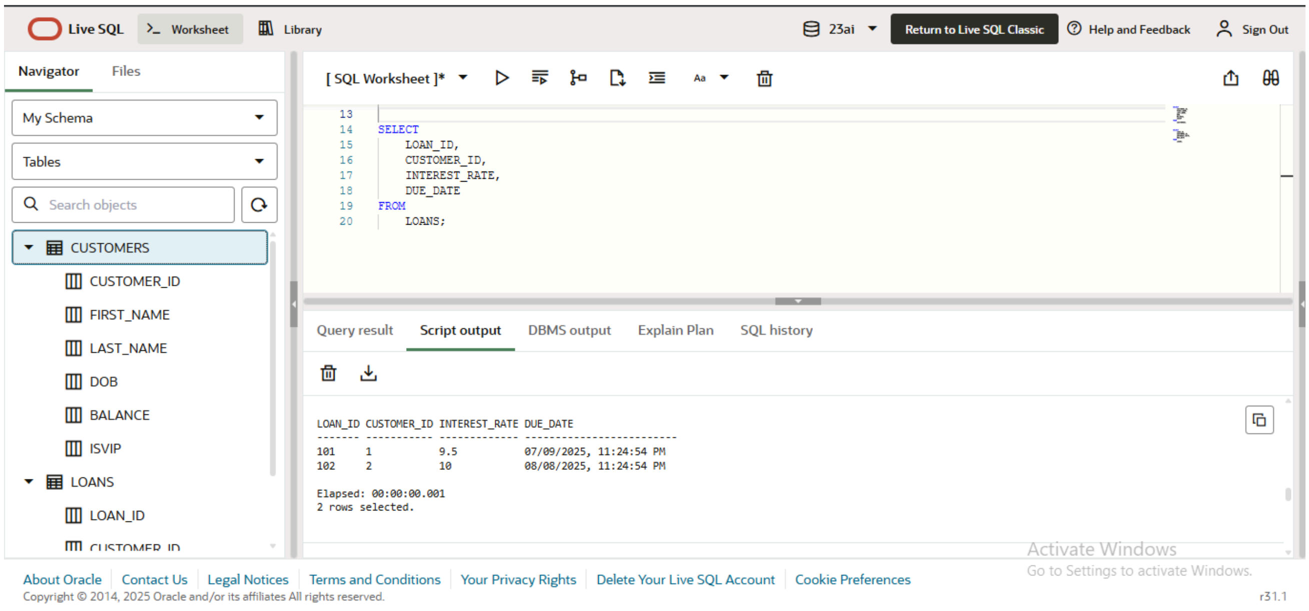
```
SELECT
    CUSTOMER_ID,
    FIRST_NAME,
    LAST_NAME,
    DOB,
    BALANCE,
    ISVIP
FROM
    CUSTOMERS;
```

```
Show more...

CUSTOMER_ID FIRST_NAME LAST_NAME DOB                       BALANCE ISVIP
----------- ---------- --------- ------------------------- ------- -----
1           John       Doe       01/01/1950, 05:30:00 AM   15000   N
2           Jane       Smith     06/10/1985, 05:30:00 AM   8000    N

Elapsed: 00:00:00.001
2 rows selected.
```

About Oracle | Contact Us | Legal Notices | Terms and Conditions | Your Privacy Rights | Delete Your Live SQL Account | Cookie Preferences
Copyright © 2014, 2025 Oracle and/or its affiliates All rights reserved. r31.1

Creation of **LOAN** Table

```
INSERT INTO loans VALUES (101, 1, 9.5, SYSDATE + 10);
INSERT INTO loans VALUES (102, 2, 10.0, SYSDATE + 40);
```



```
SELECT
    LOAN_ID,
    CUSTOMER_ID,
    INTEREST_RATE,
    DUE_DATE
FROM
    LOANS;
```

```
LOAN_ID CUSTOMER_ID INTEREST_RATE DUE_DATE
------- ----------- ------------- -------------------------
101     1           9.5           07/09/2025, 11:24:54 PM
102     2           10            08/08/2025, 11:24:54 PM

Elapsed: 00:00:00.001
2 rows selected.
```

About Oracle | Contact Us | Legal Notices | Terms and Conditions | Your Privacy Rights | Delete Your Live SQL Account | Cookie Preferences
Copyright © 2014, 2025 Oracle and/or its affiliates All rights reserved. r31.1

Exercise 1: Control Structures

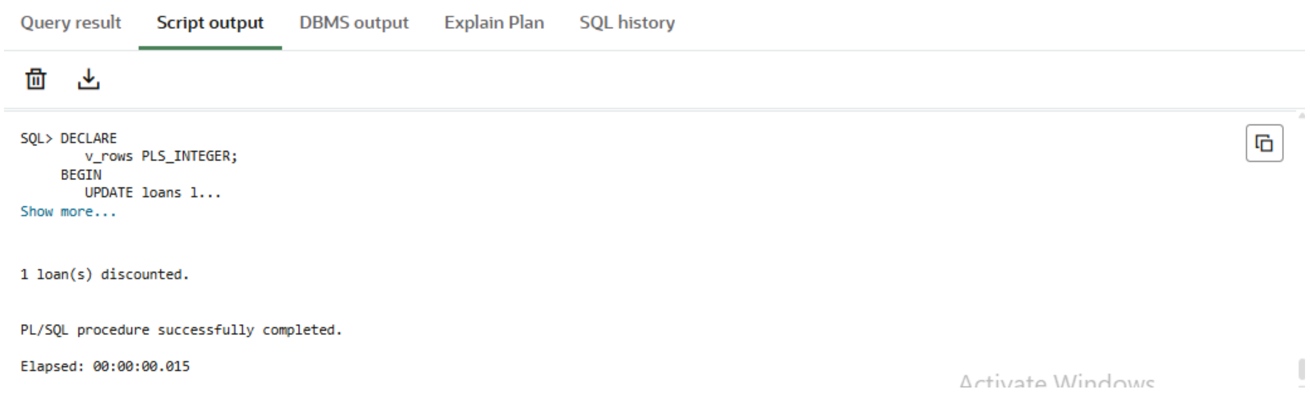
Scenario 1: The bank wants to apply a discount to loan interest rates for customers above 60 years old.

- **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.

Solution:-

```
SET SERVEROUTPUT ON;

DECLARE
   v_rows PLS_INTEGER;
BEGIN
   UPDATE loans l
   SET    interest_rate = interest_rate * 0.99
   WHERE  EXISTS (
            SELECT 1
            FROM   customers c
            WHERE  c.customer_id = l.customer_id
            AND    TRUNC(MONTHS_BETWEEN(SYSDATE, c.dob)/12) > 60
          );

   v_rows := SQL%ROWCOUNT;
   DBMS_OUTPUT.PUT_LINE(v_rows || ' loan(s) discounted.');
   COMMIT;
END;
/
```

OUTPUT:-



```
SQL> DECLARE
        v_rows PLS_INTEGER;
     BEGIN
        UPDATE loans l...

1 loan(s) discounted.

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.015
```

After Senorio 1 the updated table are as follows

```
SELECT * FROM loans;

SELECT * FROM customers;
```

```
LOAN_ID CUSTOMER_ID INTEREST_RATE DUE_DATE
------- ----------- ------------- -------------------------
101     1           9.41          07/09/2025, 11:24:54 PM
102     2           10            08/08/2025, 11:24:54 PM

Elapsed: 00:00:00.003
2 rows selected.

CUSTOMER_ID FIRST_NAME LAST_NAME DOB                       BALANCE ISVIP
----------- ---------- --------- ------------------------- ------- -----
1           John       Doe       01/01/1950, 05:30:00 AM   15000   N
2           Jane       Smith     06/10/1985, 05:30:00 AM   8000    N

Elapsed: 00:00:00.002
2 rows selected.
```

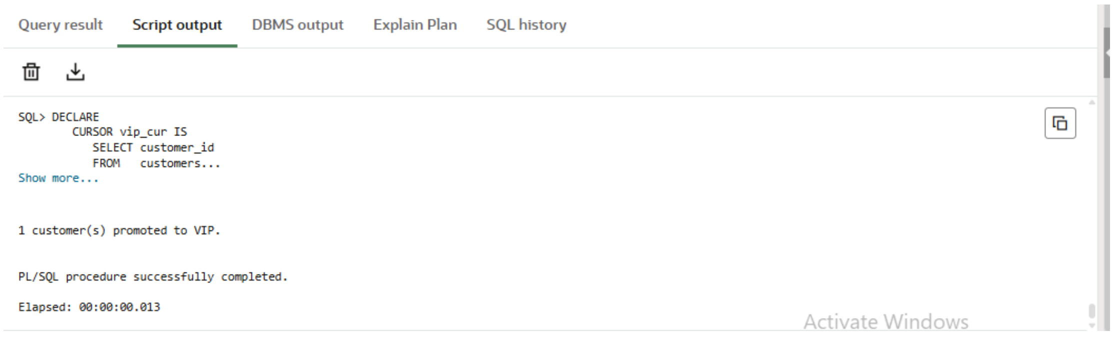
Scenario 2: A customer can be promoted to VIP status based on their balance.

- **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.

Solution:-

```
SET SERVEROUTPUT ON;

DECLARE
   CURSOR vip_cur IS
      SELECT customer_id
      FROM   customers
      WHERE  balance > 10000;

   v_count PLS_INTEGER := 0;
BEGIN
   FOR rec IN vip_cur LOOP
      UPDATE customers
      SET    isvip = 'Y'
      WHERE  customer_id = rec.customer_id;
      v_count := v_count + 1;
   END LOOP;

   DBMS_OUTPUT.PUT_LINE(v_count || ' customer(s) promoted to VIP.');
   COMMIT;
END;
/
```

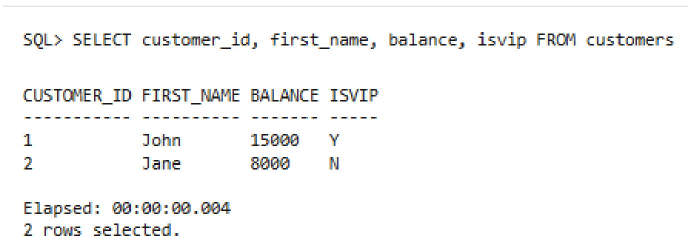
OUTPUT:-



```
SQL> DECLARE
        CURSOR vip_cur IS
           SELECT customer_id
           FROM   customers...
Show more...

1 customer(s) promoted to VIP.

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.013
```

After Senorio 2 the updated table is as follows

```
SELECT customer_id, first_name, balance, isvip FROM customers;
```



```
SQL> SELECT customer_id, first_name, balance, isvip FROM customers

CUSTOMER_ID FIRST_NAME BALANCE ISVIP
----------- ---------- ------- -----
1           John       15000   Y
2           Jane       8000    N

Elapsed: 00:00:00.004
2 rows selected.
```

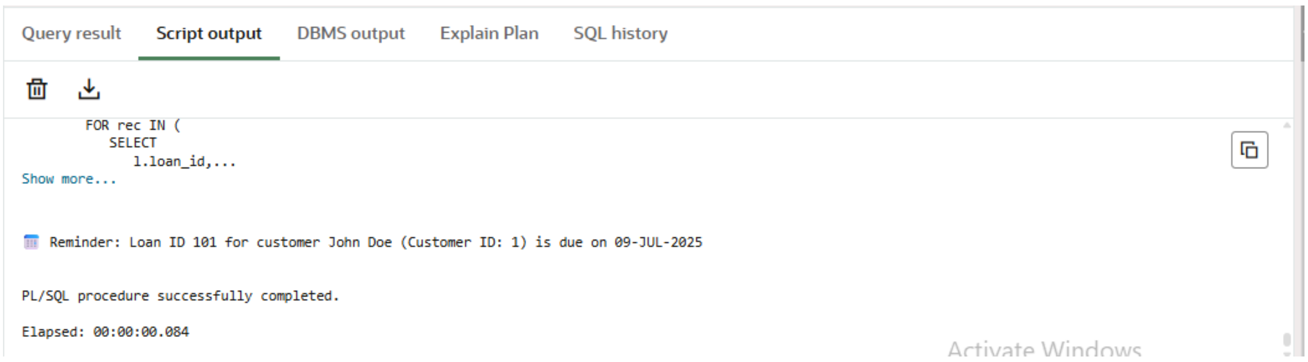
Scenario 3: The bank wants to send reminders to customers whose loans are due within the next 30 days.

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

Solution:-

```
SET SERVEROUTPUT ON;

BEGIN
   FOR rec IN (
      SELECT
         l.loan_id,
         l.due_date,
         c.customer_id,
         c.first_name || ' ' || c.last_name AS full_name
      FROM
         loans l
      JOIN
         customers c ON c.customer_id = l.customer_id
      WHERE
         l.due_date BETWEEN SYSDATE AND SYSDATE + 30
      ORDER BY
         l.due_date
   ) LOOP
      DBMS_OUTPUT.PUT_LINE(
         '📅 Reminder: Loan ID ' || rec.loan_id ||
         ' for customer ' || rec.full_name ||
         ' (Customer ID: ' || rec.customer_id ||
         ') is due on ' || TO_CHAR(rec.due_date, 'DD-MON-YYYY')
      );
   END LOOP;
END;
/
```

OUTPUT:-



```
      FOR rec IN (
         SELECT
            l.loan_id,...
Show more...


📅 Reminder: Loan ID 101 for customer John Doe (Customer ID: 1) is due on 09-JUL-2025

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.084
```

Question 3:-

Build a minimal schema & sample data

```
-------------------------------------------------------------------
-- 1A. Core tables
-------------------------------------------------------------------
CREATE TABLE customers (
  customer_id  NUMBER PRIMARY KEY,
  first_name   VARCHAR2(50),
  last_name    VARCHAR2(50)
);

CREATE TABLE accounts (
  account_id   NUMBER PRIMARY KEY,
  customer_id  NUMBER  REFERENCES customers(customer_id),
  account_type VARCHAR2(15),   -- 'SAVINGS' or 'CHECKING'
  balance      NUMBER(15,2)
);

CREATE TABLE departments (
  department_id NUMBER PRIMARY KEY,
  dept_name     VARCHAR2(50)
);

CREATE TABLE employees (
  employee_id   NUMBER PRIMARY KEY,
  first_name    VARCHAR2(50),
  last_name     VARCHAR2(50),
  department_id NUMBER REFERENCES departments(department_id),
  salary        NUMBER(15,2)
);

-------------------------------------------------------------------
-- 1B. Sample data (tiny but enough to test)
-------------------------------------------------------------------
INSERT INTO customers VALUES (1,'John','Doe');
INSERT INTO customers VALUES (2,'Jane','Smith');

INSERT INTO accounts VALUES (1001,1,'SAVINGS',  5000);
INSERT INTO accounts VALUES (1002,1,'CHECKING', 2000);
INSERT INTO accounts VALUES (1003,2,'SAVINGS', 12000);

INSERT INTO departments VALUES (10,'Operations');
INSERT INTO departments VALUES (20,'IT');

INSERT INTO employees VALUES (101,'Alice','Green',10,60000);
INSERT INTO employees VALUES (102,'Bob','Brown',10,55000);
INSERT INTO employees VALUES (103,'Carol','White',20,70000);

COMMIT;
```

Scenario 1: The bank needs to process monthly interest for all savings accounts.
○ **Question:** Write a stored procedure **ProcessMonthlyInterest** that calculates and updates the balance of all savings accounts by applying an interest rate of 1% to the current balance.

Solution:-

```
CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest
IS
  v_rows PLS_INTEGER;
BEGIN
  UPDATE accounts
  SET    balance = balance * 1.01          -- +1 %
  WHERE  account_type = 'SAVINGS';

  v_rows := SQL%ROWCOUNT;
  DBMS_OUTPUT.PUT_LINE(v_rows || ' savings account(s) credited with monthly
interest.');
  COMMIT;
END;
/
```

Output:-

```
Procedure PROCESSMONTHLYINTEREST compiled

Elapsed: 00:00:00.025
```

Test call:-

```
SET SERVEROUTPUT ON;
EXEC ProcessMonthlyInterest;

-- Verify
SELECT account_id, balance FROM accounts WHERE account_type =
'SAVINGS';
```

Output:-

```
ACCOUNT_ID BALANCE
---------- -------
1001       5050
1003       12120

Elapsed: 00:00:00.005
2 rows selected.
```

Scenario 2: The bank wants to implement a bonus scheme for employees based on their performance.

- **Question:** Write a stored procedure **UpdateEmployeeBonus** that updates the salary of employees in a given department by adding a bonus percentage passed as a parameter.

Solution:-

```
CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (
    p_dept_id   IN  employees.department_id%TYPE,
    p_bonus_pct IN  NUMBER              -- e.g. pass 5 for 5 %
)
IS
  v_rows PLS_INTEGER;
BEGIN
  UPDATE employees
  SET    salary = salary * (1 + p_bonus_pct/100)
  WHERE  department_id = p_dept_id;

  v_rows := SQL%ROWCOUNT;
  DBMS_OUTPUT.PUT_LINE(v_rows || ' employee(s) received a ' || p_bonus_pct
|| '% bonus in department ' || p_dept_id || '.');
  COMMIT;
END;
/
```

Output:-

```
Procedure UPDATEEMPLOYEEBONUS compiled

Elapsed: 00:00:00.022
```

Test call:-

```
SET SERVEROUTPUT ON;
EXEC UpdateEmployeeBonus(p_dept_id => 10, p_bonus_pct => 5);

-- Verify
SELECT employee_id, department_id, salary FROM employees WHERE
department_id = 10;
```

Output:-

```
EMPLOYEE_ID DEPARTMENT_ID SALARY
----------- ------------- ------
101         10            63000
102         10            57750

Elapsed: 00:00:00.005
2 rows selected.
```

Scenario 3: Customers should be able to transfer funds between their accounts.

- **Question:** Write a stored procedure **TransferFunds** that transfers a specified amount from one account to another, checking that the source account has sufficient balance before making the transfer.

Solution:-

```
CREATE OR REPLACE PROCEDURE TransferFunds (
    p_from_acct IN accounts.account_id%TYPE,
    p_to_acct   IN accounts.account_id%TYPE,
    p_amount    IN NUMBER
)
IS
  v_from_bal   NUMBER;
BEGIN
  -- 1. Get current balance of source account
  SELECT balance
  INTO   v_from_bal
  FROM   accounts
  WHERE  account_id = p_from_acct
  FOR UPDATE;

  -- 2. Check sufficient funds
  IF v_from_bal < p_amount THEN
     RAISE_APPLICATION_ERROR(-20001,
        'Insufficient balance in account ' || p_from_acct);
  END IF;

  -- 3. Debit source, credit destination
  UPDATE accounts
  SET    balance = balance - p_amount
  WHERE  account_id = p_from_acct;

  UPDATE accounts
  SET    balance = balance + p_amount
  WHERE  account_id = p_to_acct;

  DBMS_OUTPUT.PUT_LINE('Transferred ' || p_amount ||
                       ' from ' || p_from_acct ||
                       ' to '     || p_to_acct || '.');
  COMMIT;
END;
/
```

Output:-

```
Procedure TRANSFERFUNDS compiled

Elapsed: 00:00:00.021
```

Test call:-

```
SET SERVEROUTPUT ON;

-- Successful transfer
EXEC TransferFunds(1002, 1001, 500);

-- Attempt transfer that will fail (not enough money)
BEGIN
  TransferFunds(1002, 1001, 100000);
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
/

-- Verify balances
SELECT account_id, balance FROM accounts ORDER BY account_id;
```

Output:-

```
 ACCOUNT_ID BALANCE
---------- -------
1001      5550
1002      1500
1003      12120

Elapsed: 00:00:00.003
3 rows selected.
```