**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.

**CODE:**

BEGIN

  FOR cust IN (SELECT \* FROM customers) LOOP

    IF cust.age > 60 THEN

      UPDATE loans

      SET interest\_rate = interest\_rate - 1

      WHERE customer\_id = cust.customer\_id;

      DBMS\_OUTPUT.PUT\_LINE('Applied 1% discount to customer ' || cust.name);

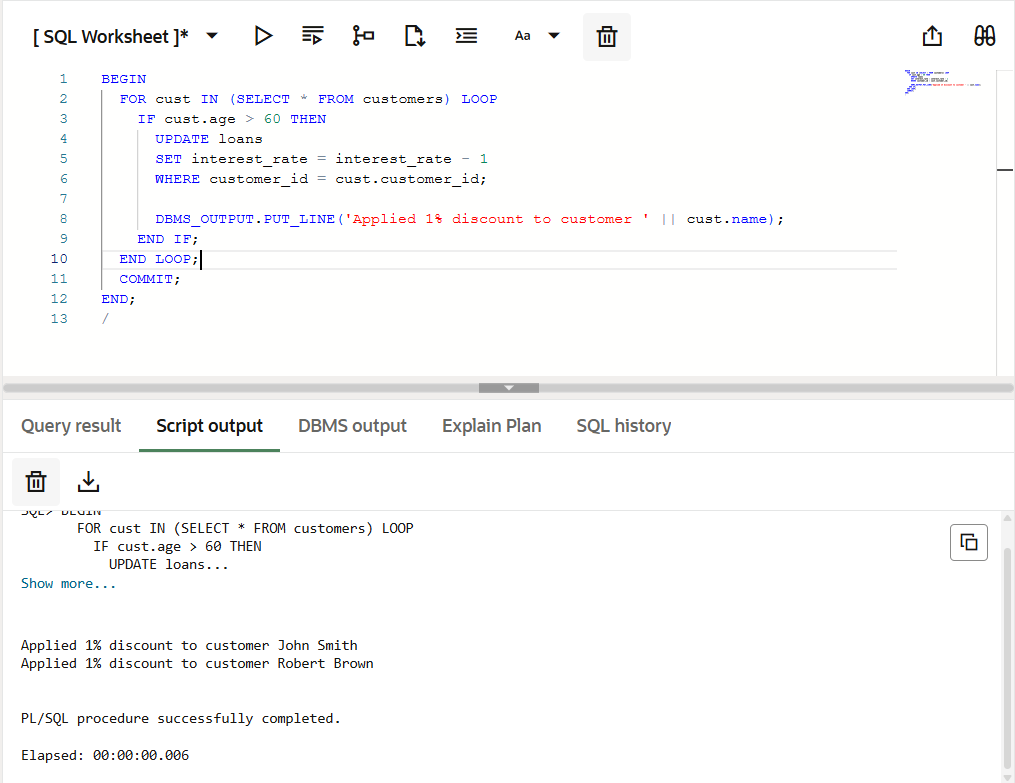
    END IF;

  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.

**CODE:**

BEGIN

FOR cust IN (SELECT \* FROM customers WHERE balance > 10000) LOOP

UPDATE customers

SET is\_vip = 'TRUE'

WHERE customer\_id = cust.customer\_id;

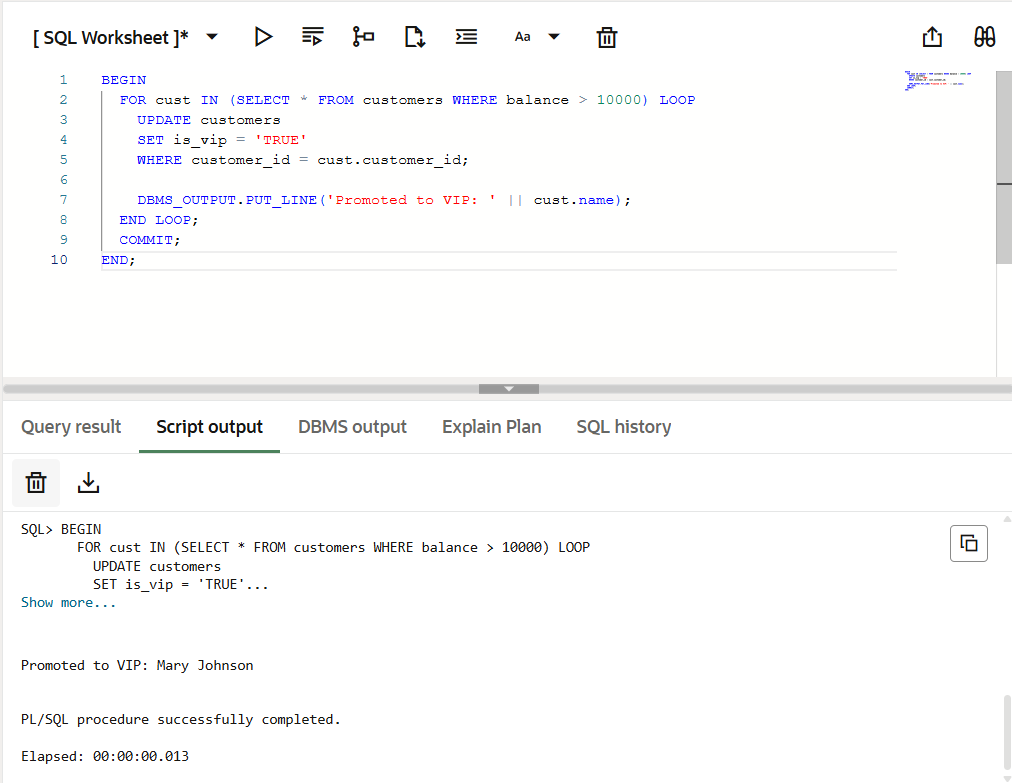
DBMS\_OUTPUT.PUT\_LINE('Promoted to VIP: ' || cust.name);

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.

**CODE:**

BEGIN

FOR loan\_rec IN (

SELECT l.loan\_id, c.name, c.customer\_id, l.due\_date

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

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

DBMS\_OUTPUT.PUT\_LINE('REMINDER: ' || loan\_rec.name ||

', your loan #' || loan\_rec.loan\_id ||

' is due on ' || TO\_CHAR(loan\_rec.due\_date, 'DD-MON-YYYY'));

UPDATE customers

SET last\_reminder\_date = SYSDATE

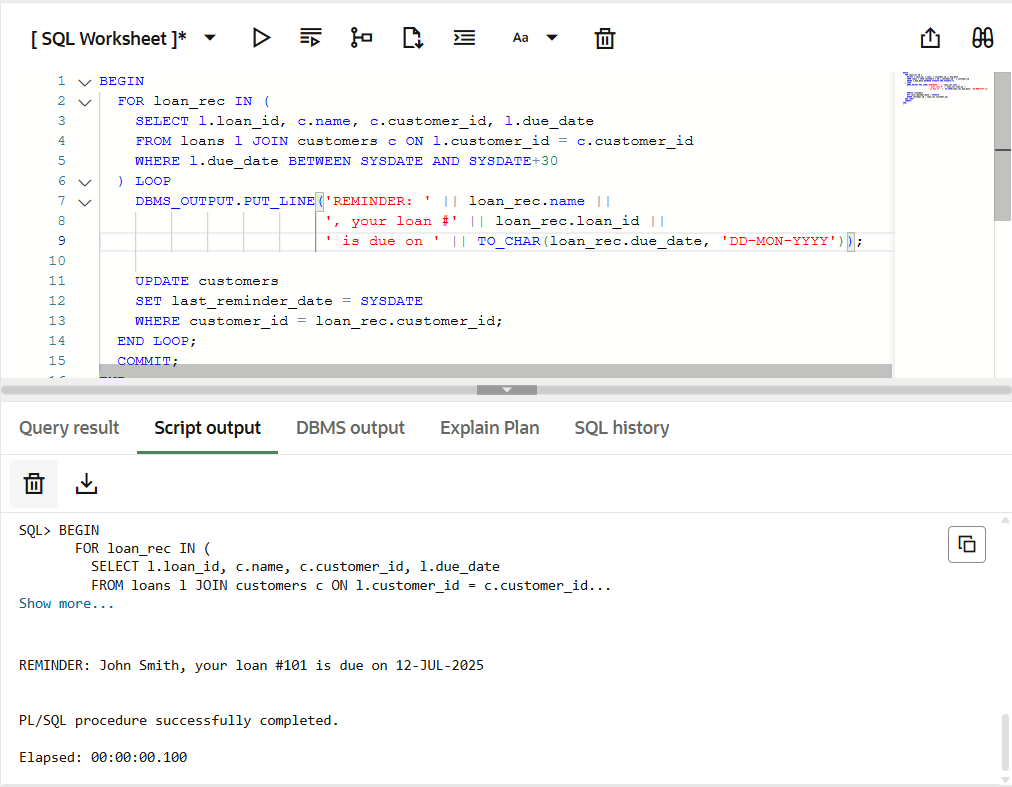
WHERE customer\_id = loan\_rec.customer\_id;

END LOOP;

COMMIT;

END;

**OUTPUT:**

****

**Exercise 3: Stored Procedures**

**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.

**CODE:**

WITH before\_data AS (

  SELECT AccountID, AccountType, Balance, 'Before' AS status

  FROM Accounts

  WHERE AccountType = 'Savings'

)

SELECT \* FROM before\_data

ORDER BY AccountID, status;

EXEC ProcessMonthlyInterest;

WITH before\_data AS (

  SELECT AccountID, AccountType, Balance, 'Before' AS status

  FROM Accounts AS OF TIMESTAMP (SYSTIMESTAMP - INTERVAL '5' MINUTE)

  WHERE AccountType = 'Savings'

)

SELECT \* FROM before\_data

UNION ALL

SELECT AccountID, AccountType, Balance, 'After' AS status

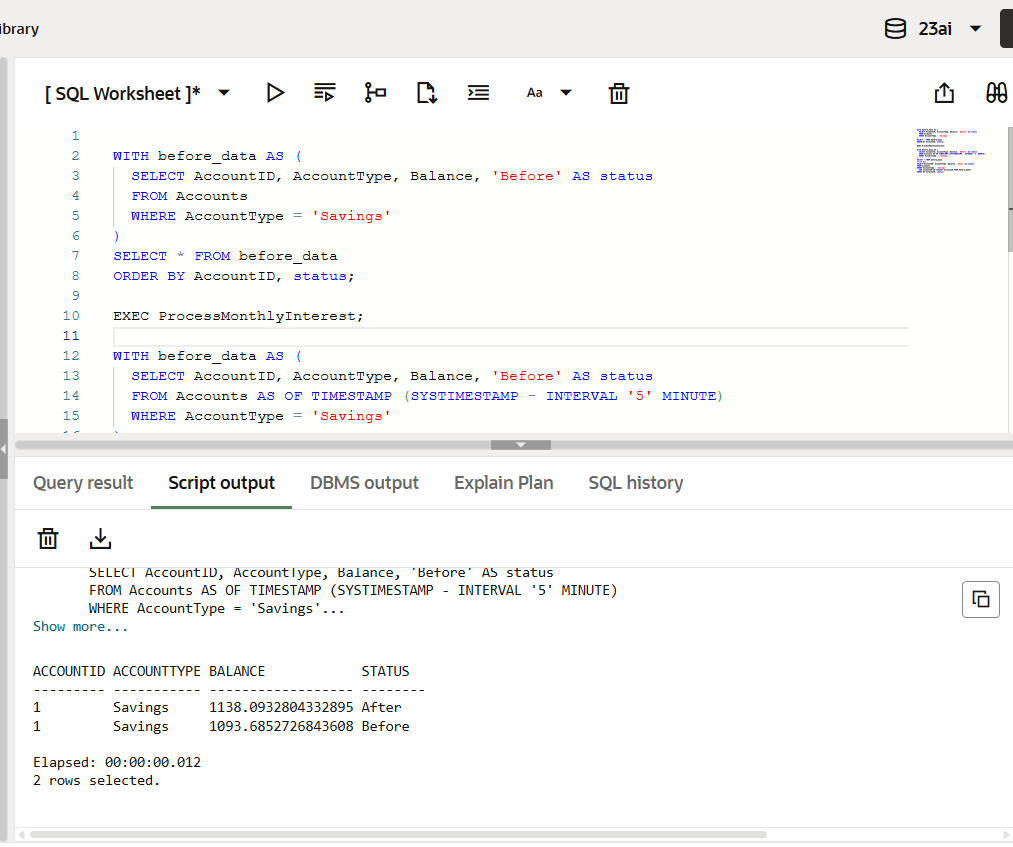
FROM Accounts

WHERE AccountType = 'Savings'

  AND AccountID IN (SELECT AccountID FROM before\_data)

ORDER BY AccountID, status;

**OUTPUT:**



**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.

**CODE:**

WITH before\_data AS (

  SELECT EmployeeID, Name, Department, Salary, 'Before' AS status

  FROM Employees

  WHERE Department = 'IT'

)

SELECT \* FROM before\_data

UNION ALL

SELECT EmployeeID, Name, Department, Salary, 'After' AS status

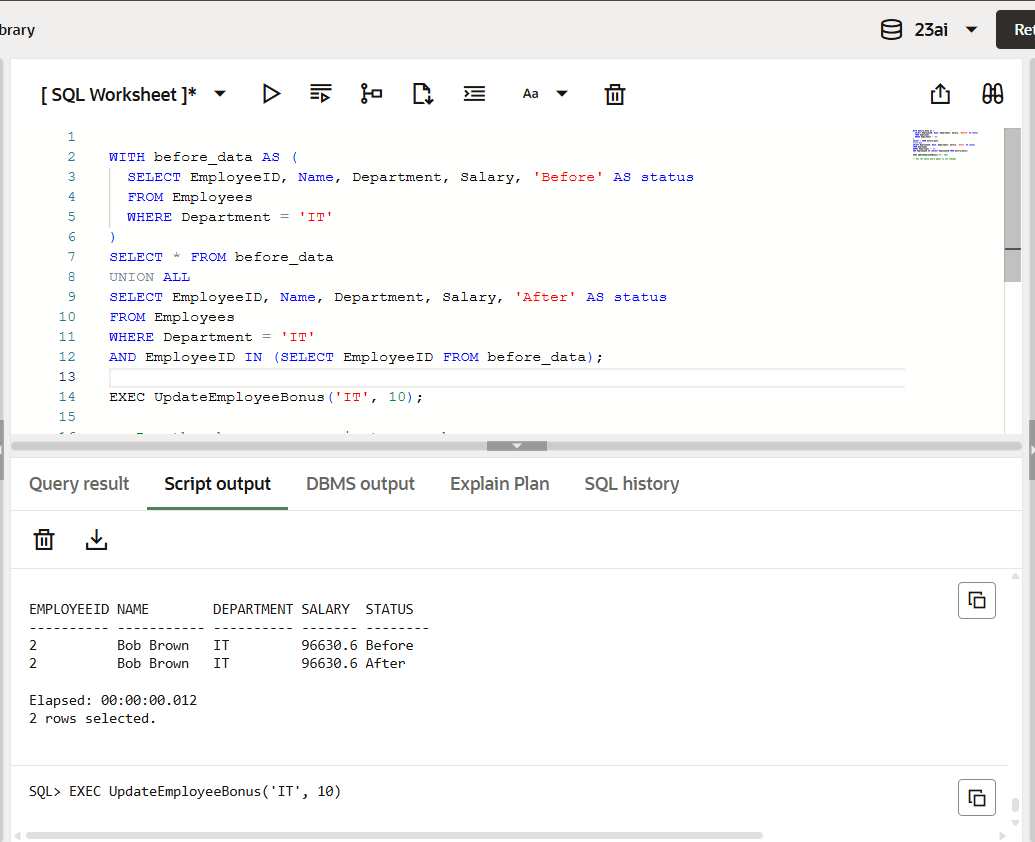
FROM Employees

WHERE Department = 'IT'

AND EmployeeID IN (SELECT EmployeeID FROM before\_data);

EXEC UpdateEmployeeBonus('IT', 10);

**OUTPUT:**

****

**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.

**CODE:**

SELECT AccountID, Balance, 'Before' AS status

FROM Accounts

WHERE AccountID IN (1, 2)

ORDER BY AccountID;

EXEC TransferFunds(1, 2, 200);

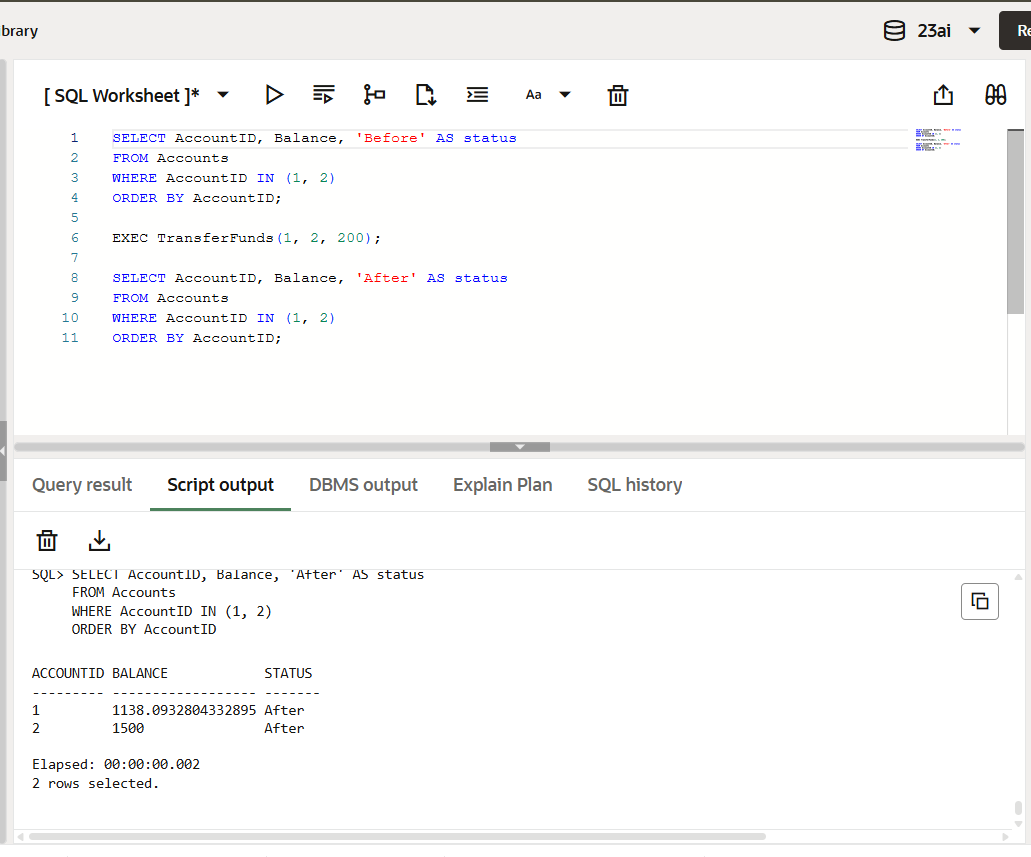
SELECT AccountID, Balance, 'After' AS status

FROM Accounts

WHERE AccountID IN (1, 2)

ORDER BY AccountID;

**OUTPUT:**

****