**PL/SQL - Hands-on Solutions**

**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\_customer\_id NUMBER;

v\_dob DATE;

v\_age NUMBER;

BEGIN

FOR cust IN (SELECT CustomerID, DOB FROM Customers) LOOP

v\_customer\_id := cust.CustomerID;

v\_dob := cust.DOB;

v\_age := FLOOR(MONTHS\_BETWEEN(SYSDATE, v\_dob) / 12);

IF v\_age > 60 THEN

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE CustomerID = v\_customer\_id;

DBMS\_OUTPUT.PUT\_LINE('Discount applied for Customer ID: ' || v\_customer\_id);

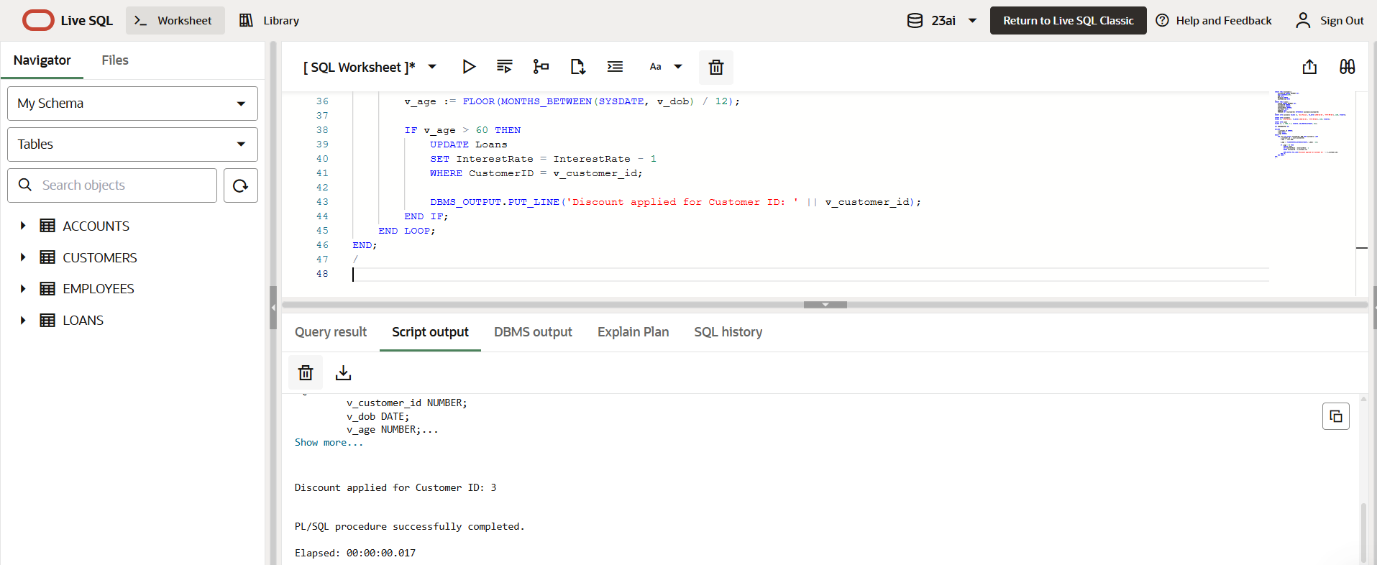
END IF;

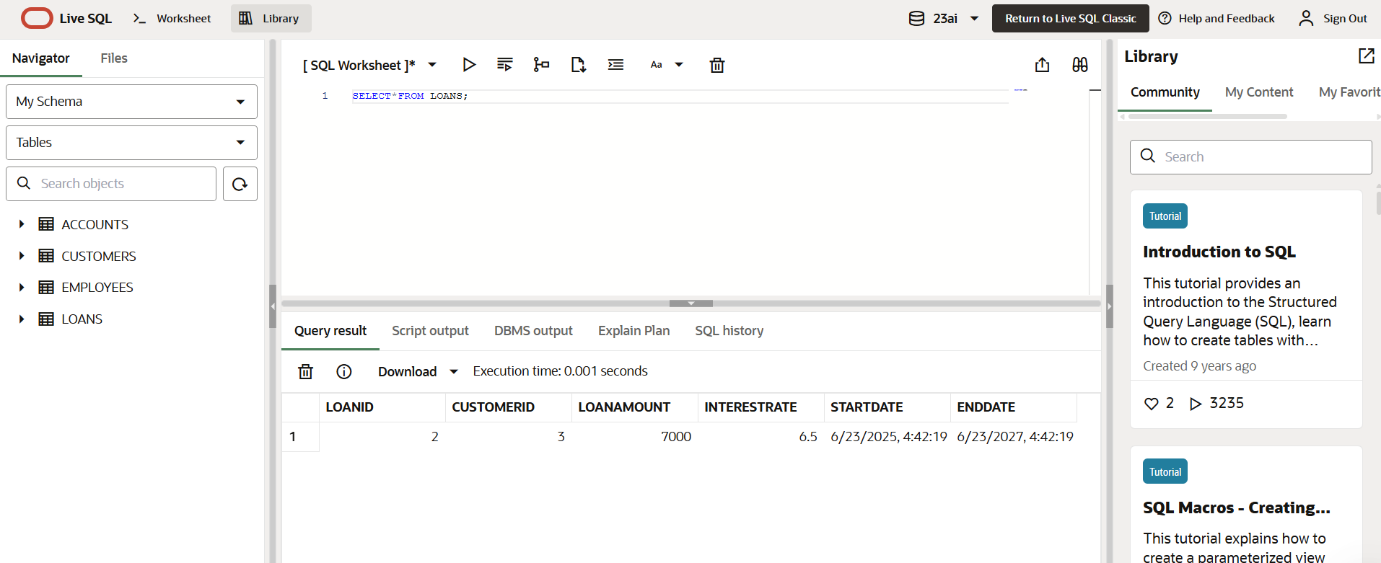
END LOOP;

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.

**Solution:**

SET SERVEROUTPUT ON;

DECLARE

v\_customer\_id Customers.CustomerID%TYPE;

v\_balance Customers.Balance%TYPE;

BEGIN

FOR cust IN (SELECT CustomerID, Balance FROM Customers) LOOP

v\_customer\_id := cust.CustomerID;

v\_balance := cust.Balance;

IF v\_balance > 10000 THEN

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = v\_customer\_id;

DBMS\_OUTPUT.PUT\_LINE('Customer ID ' || v\_customer\_id || ' marked as VIP.');

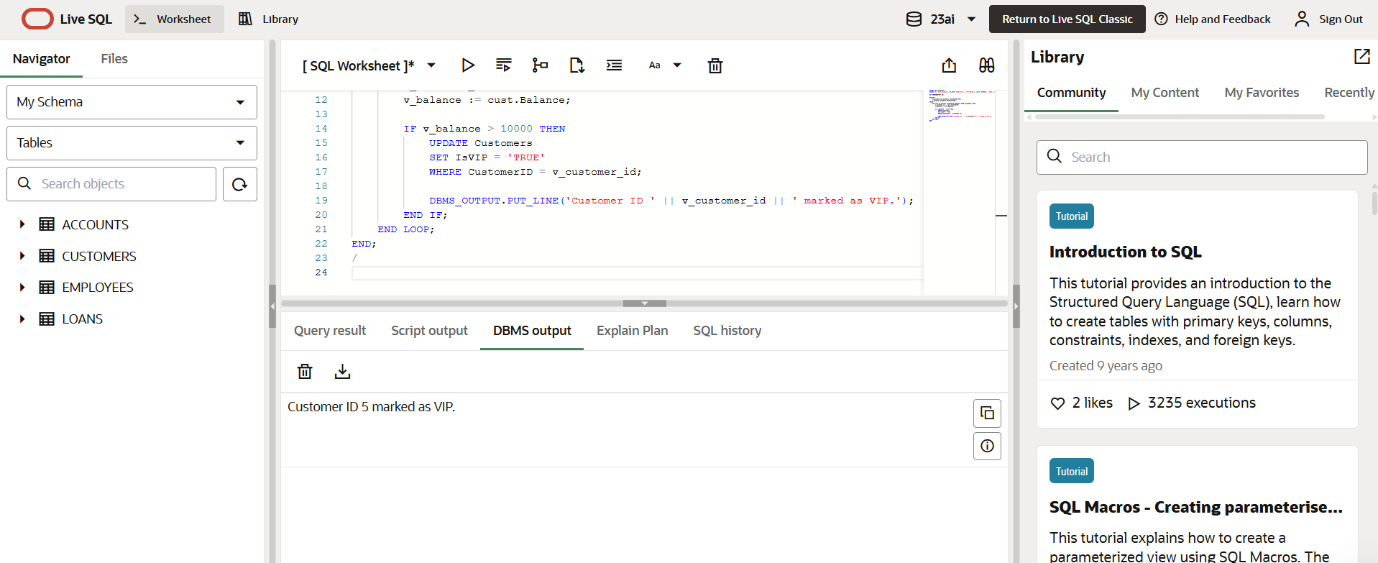
END IF;

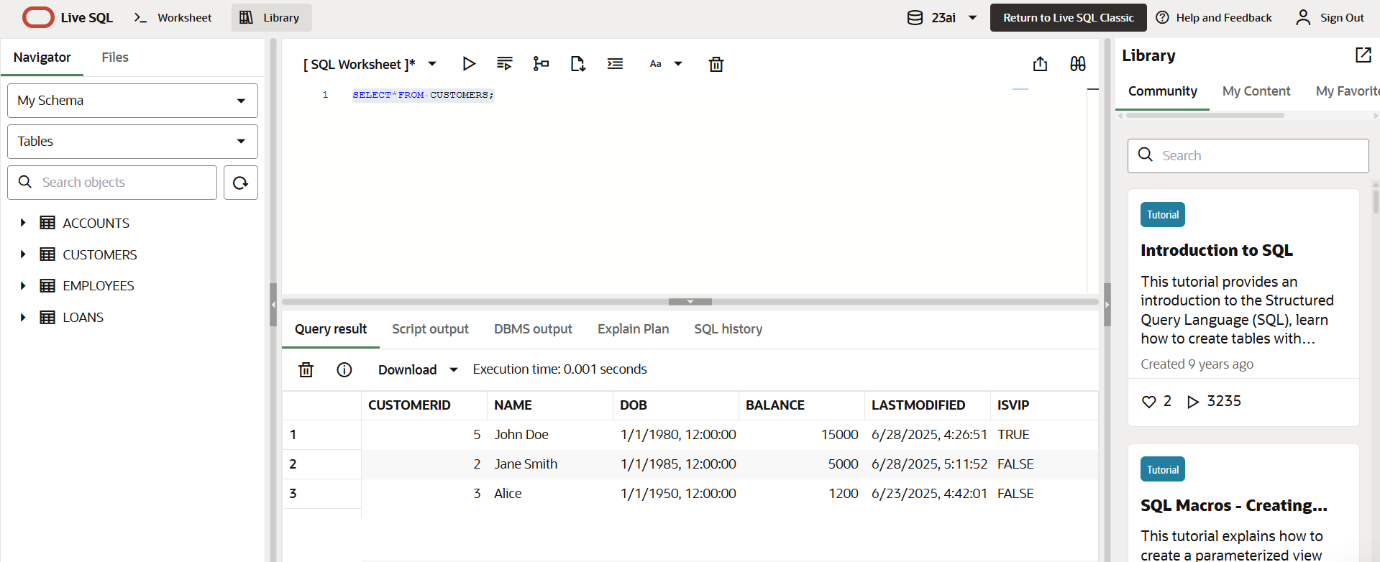
END LOOP;

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.

**SOLUTION:**

SET SERVEROUTPUT ON;

DECLARE

v\_cust\_id Loans.CustomerID%TYPE;

v\_due\_date Loans.EndDate%TYPE;

BEGIN

FOR loan\_rec IN (

SELECT CustomerID, EndDate

FROM Loans

WHERE EndDate BETWEEN SYSDATE AND SYSDATE + 30

) LOOP

v\_cust\_id := loan\_rec.CustomerID;

v\_due\_date := loan\_rec.EndDate;

DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan for Customer ID ' || v\_cust\_id ||

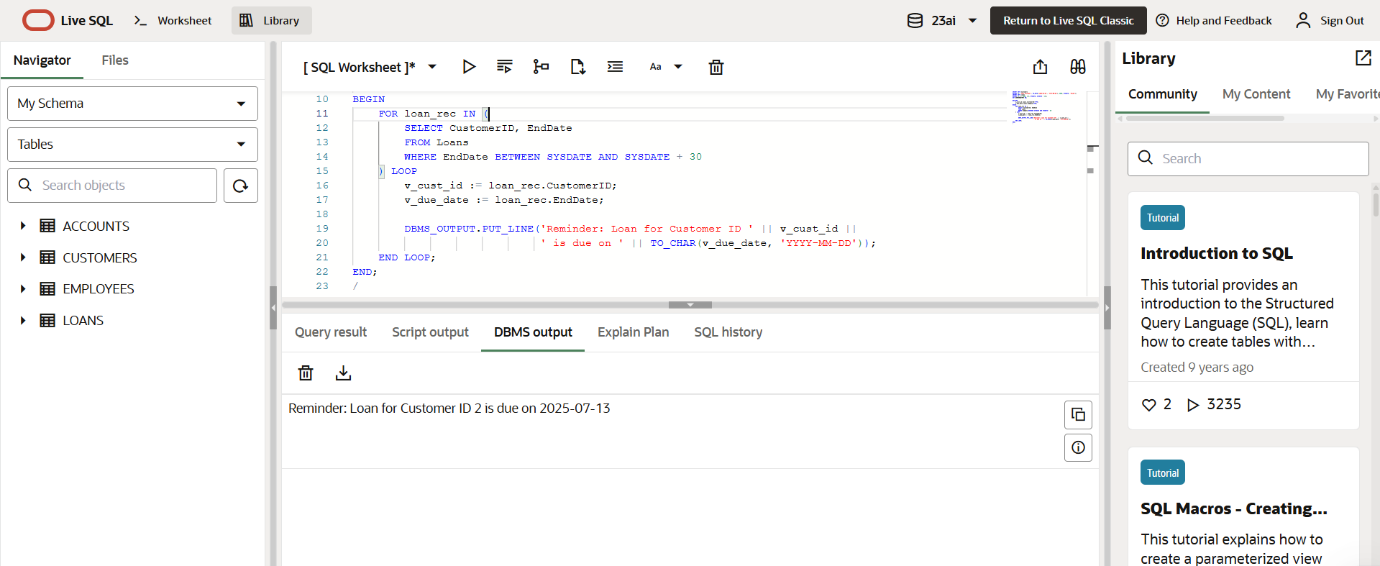
' is due on ' || TO\_CHAR(v\_due\_date, 'YYYY-MM-DD'));

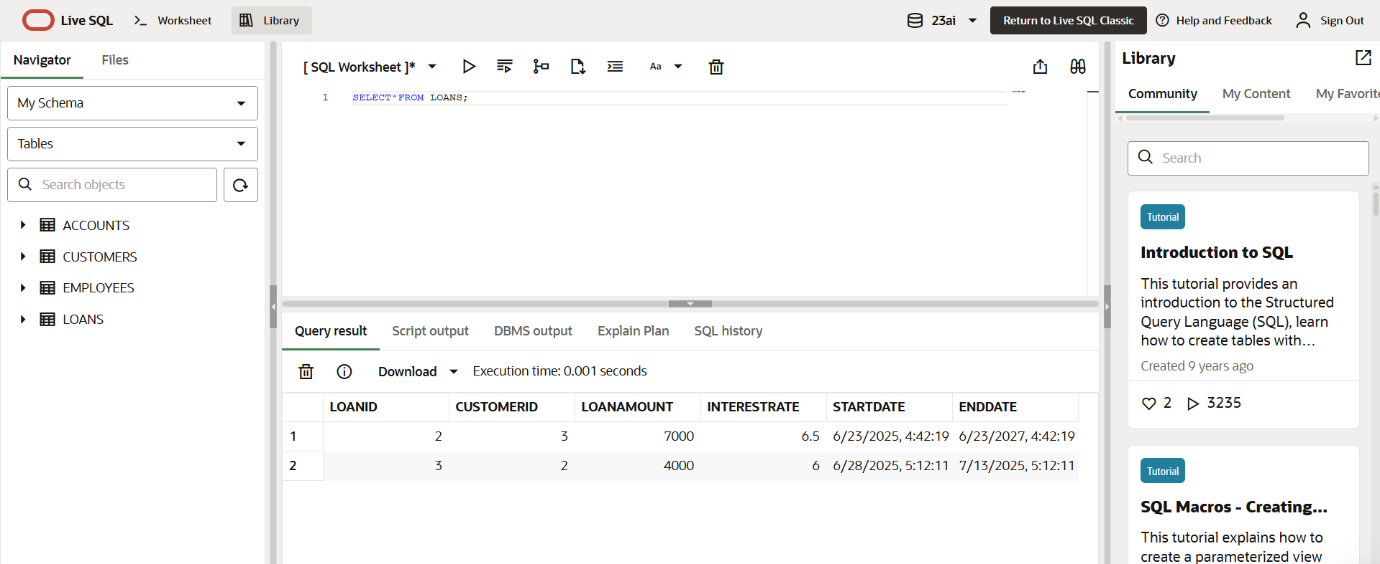
END LOOP;

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.

**SOLUTION :**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

FOR acc IN (

SELECT AccountID, Balance

FROM Accounts

WHERE AccountType = 'Savings'

) LOOP

UPDATE Accounts

SET Balance = Balance + (acc.Balance \* 0.01)

WHERE AccountID = acc.AccountID;

DBMS\_OUTPUT.PUT\_LINE('Interest applied to Account ID: ' || acc.AccountID);

END LOOP;

END;

/

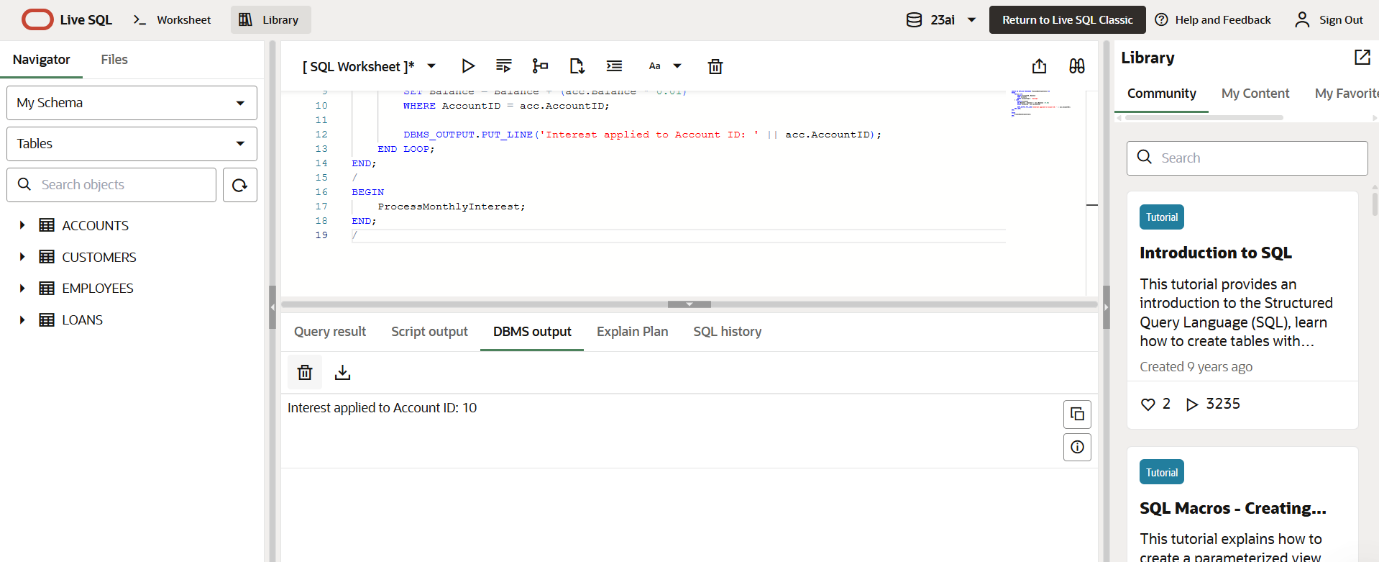
BEGIN

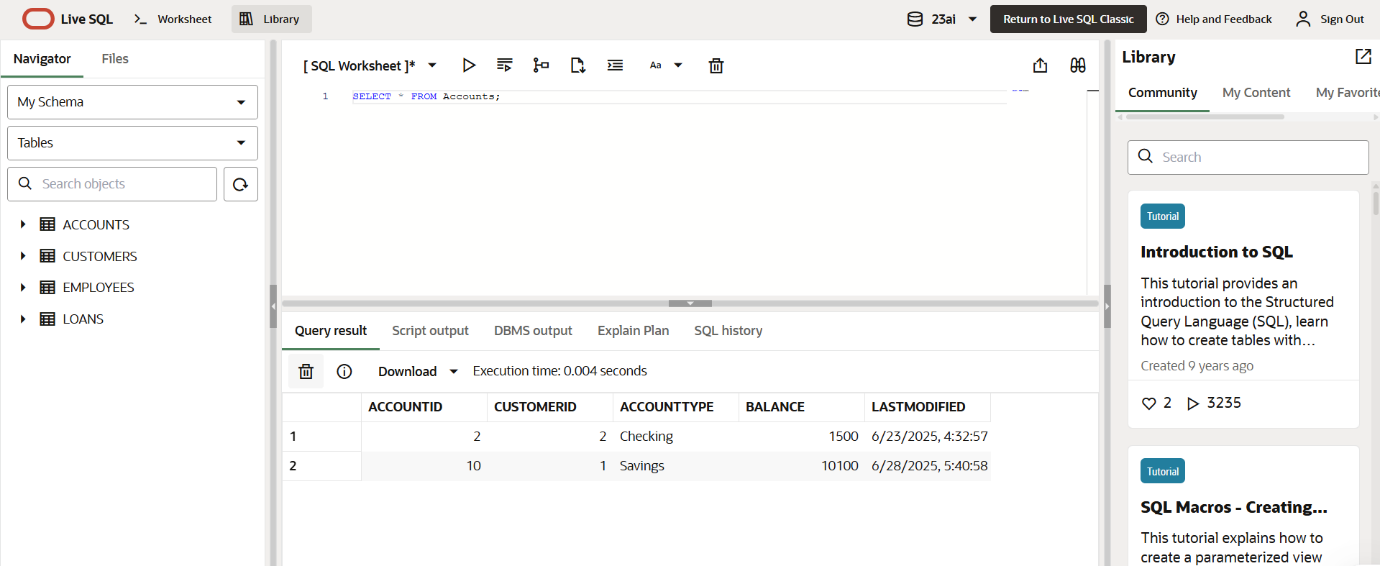
ProcessMonthlyInterest;

END;

/

**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.

**SOLUTION :**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_department IN VARCHAR2,

p\_bonus\_percent IN NUMBER

) AS

BEGIN

UPDATE Employees

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

WHERE Department = p\_department;

DBMS\_OUTPUT.PUT\_LINE('Bonus applied to Department: ' || p\_department);

END;

/

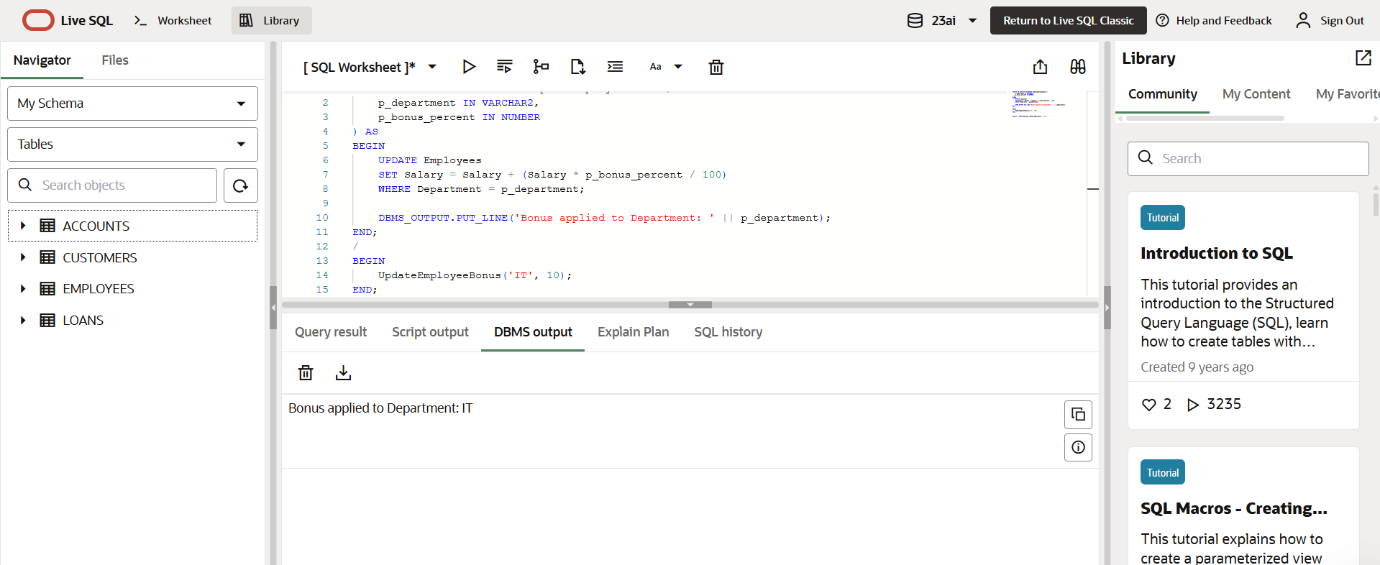
BEGIN

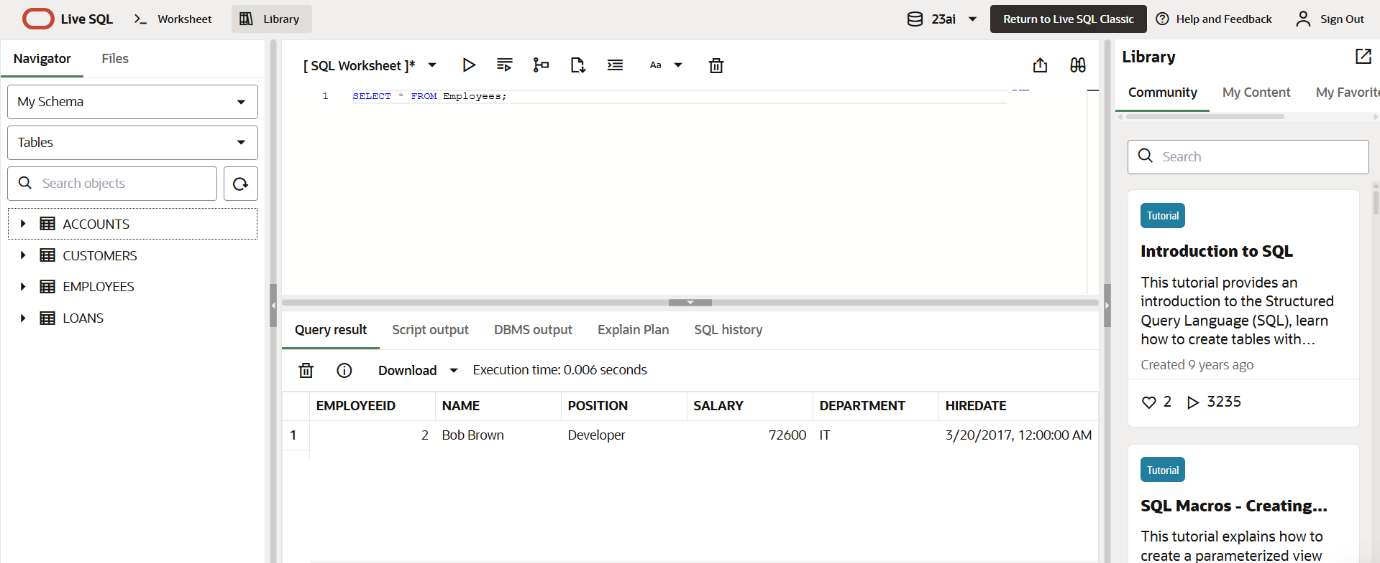
UpdateEmployeeBonus('IT', 10);

END;

/

**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.

**SOLUTION :**

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_from\_account IN NUMBER,

p\_to\_account IN NUMBER,

p\_amount IN NUMBER

) AS

v\_balance NUMBER;

BEGIN

SELECT Balance INTO v\_balance

FROM Accounts

WHERE AccountID = p\_from\_account;

IF v\_balance >= p\_amount THEN

UPDATE Accounts

SET Balance = Balance - p\_amount

WHERE AccountID = p\_from\_account;

UPDATE Accounts

SET Balance = Balance + p\_amount

WHERE AccountID = p\_to\_account;

DBMS\_OUTPUT.PUT\_LINE('₹' || p\_amount || ' transferred from Account ' ||

p\_from\_account || ' to Account ' || p\_to\_account);

ELSE

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: Insufficient balance in Account ' || p\_from\_account);

END IF;

END;

/

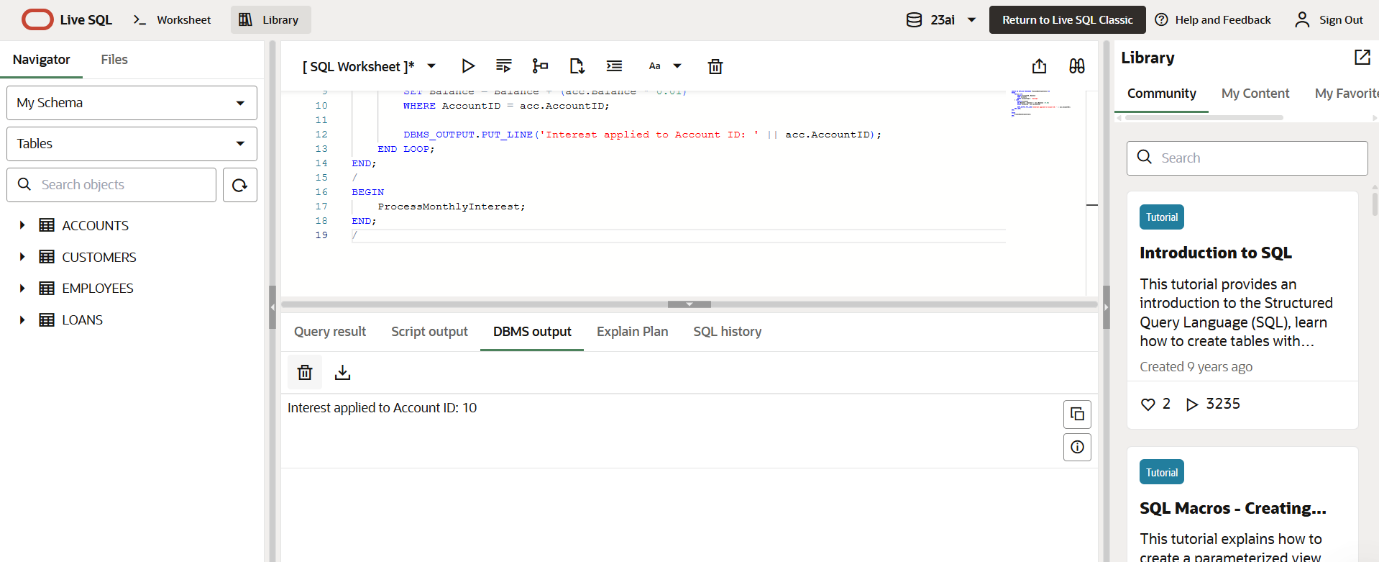
BEGIN

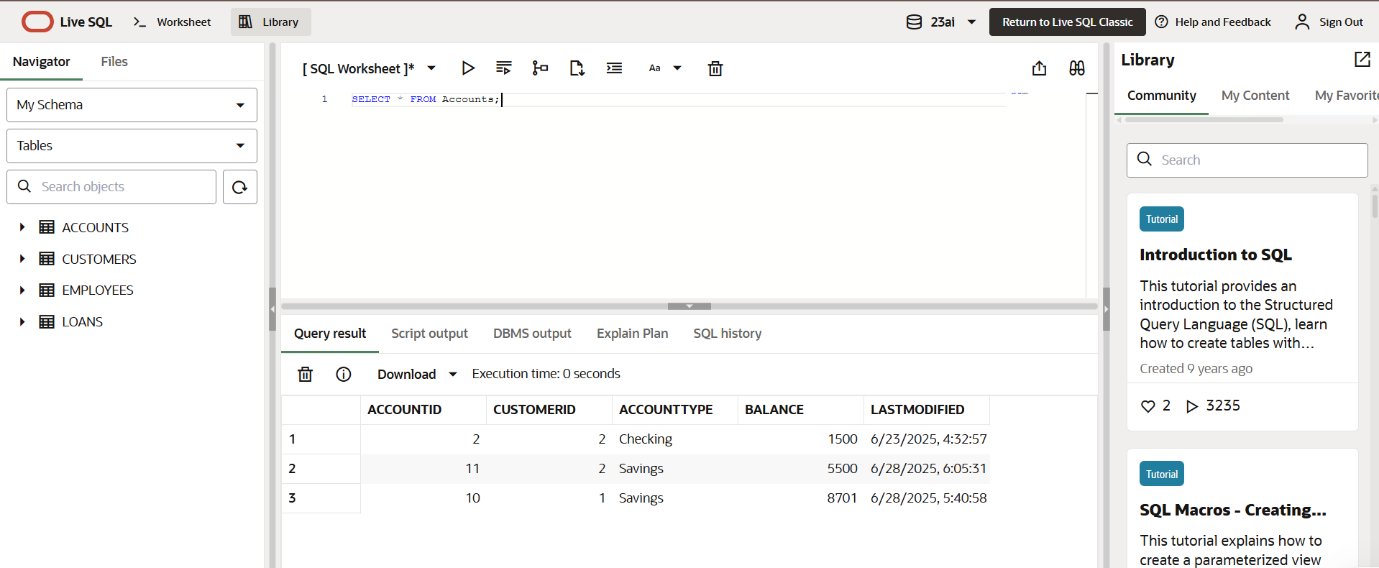
TransferFunds(10, 11, 500);

END;

/

**OUTPUT :**

****

****