WEEK 2

PL_SQL Exercise

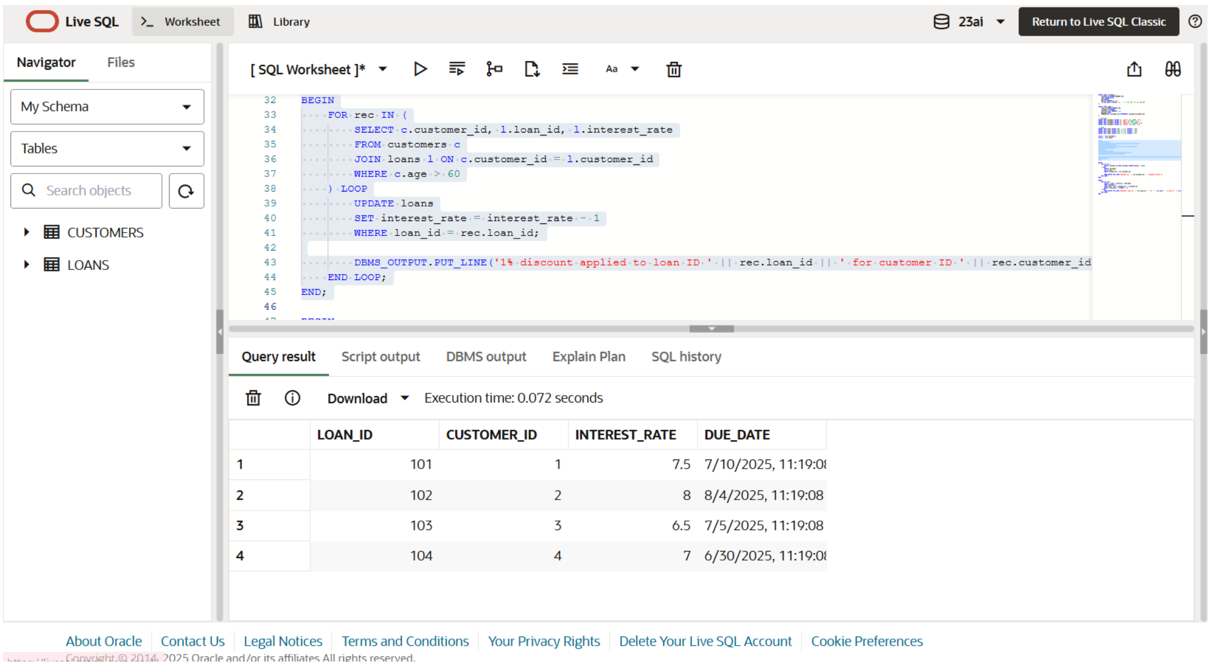
Output snapshot:

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.

Loan table after condition:



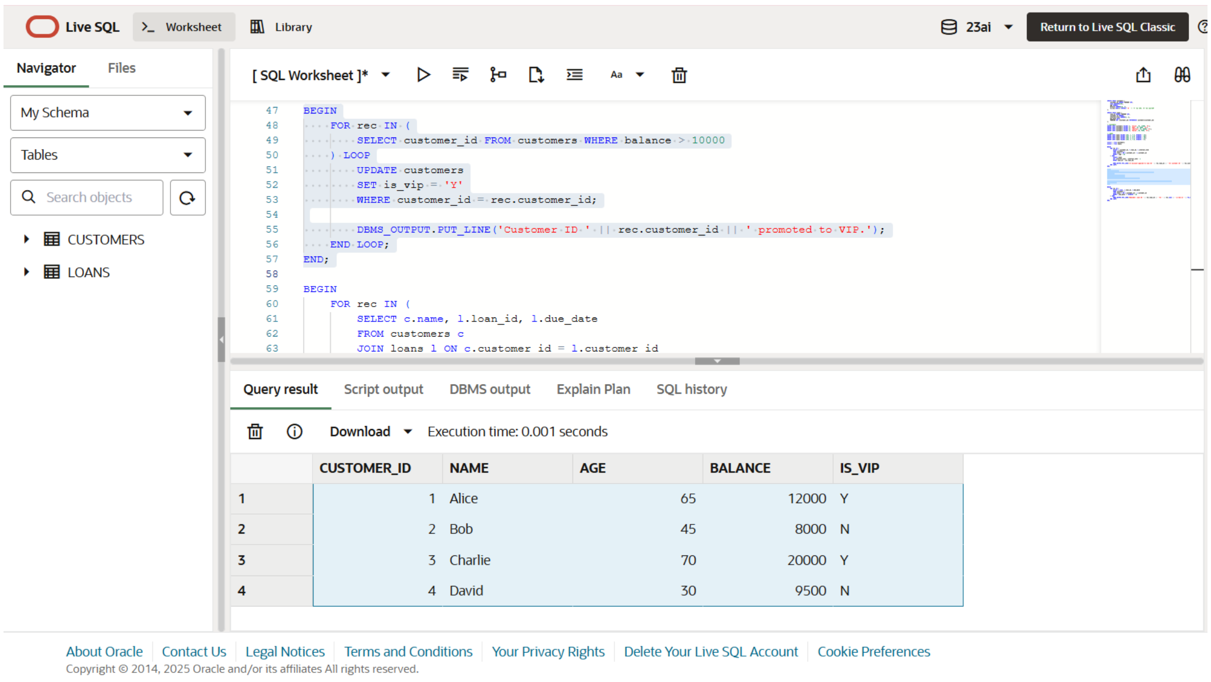
```
BEGIN
    FOR rec IN (
        SELECT c.customer_id, l.loan_id, l.interest_rate
        FROM customers c
        JOIN loans l ON c.customer_id = l.customer_id
        WHERE c.age > 60
    ) LOOP
        UPDATE loans
        SET interest_rate = interest_rate - 1
        WHERE loan_id = rec.loan_id;

        DBMS_OUTPUT.PUT_LINE('1% discount applied to loan ID ' || rec.loan_id || ' for customer ID ' || rec.customer_id
    END LOOP;
END;
```

Execution time: 0.072 seconds

	LOAN_ID	CUSTOMER_ID	INTEREST_RATE	DUE_DATE
1	101	1	7.5	7/10/2025, 11:19:0
2	102	2	8	8/4/2025, 11:19:08
3	103	3	6.5	7/5/2025, 11:19:08
4	104	4	7	6/30/2025, 11:19:0

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

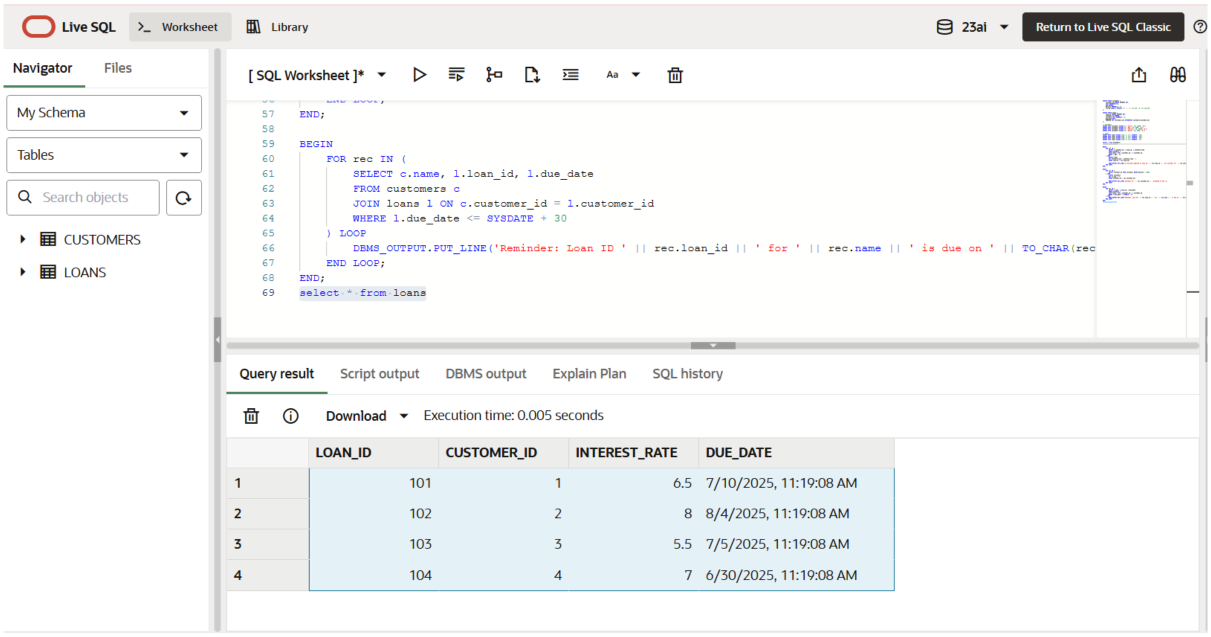


```
BEGIN
    FOR rec IN (
        SELECT customer_id FROM customers WHERE balance > 10000
    ) LOOP
        UPDATE customers
        SET is_vip = 'Y'
        WHERE customer_id = rec.customer_id;

        DBMS_OUTPUT.PUT_LINE('Customer ID ' || rec.customer_id || ' promoted to VIP.');
    END LOOP;
END;

BEGIN
    FOR rec IN (
        SELECT c.name, l.loan_id, l.due_date
        FROM customers c
        JOIN loans l ON c.customer_id = l.customer_id
```

	CUSTOMER_ID	NAME	AGE	BALANCE	IS_VIP
1	1	Alice	65	12000	Y
2	2	Bob	45	8000	N
3	3	Charlie	70	20000	Y
4	4	David	30	9500	N

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

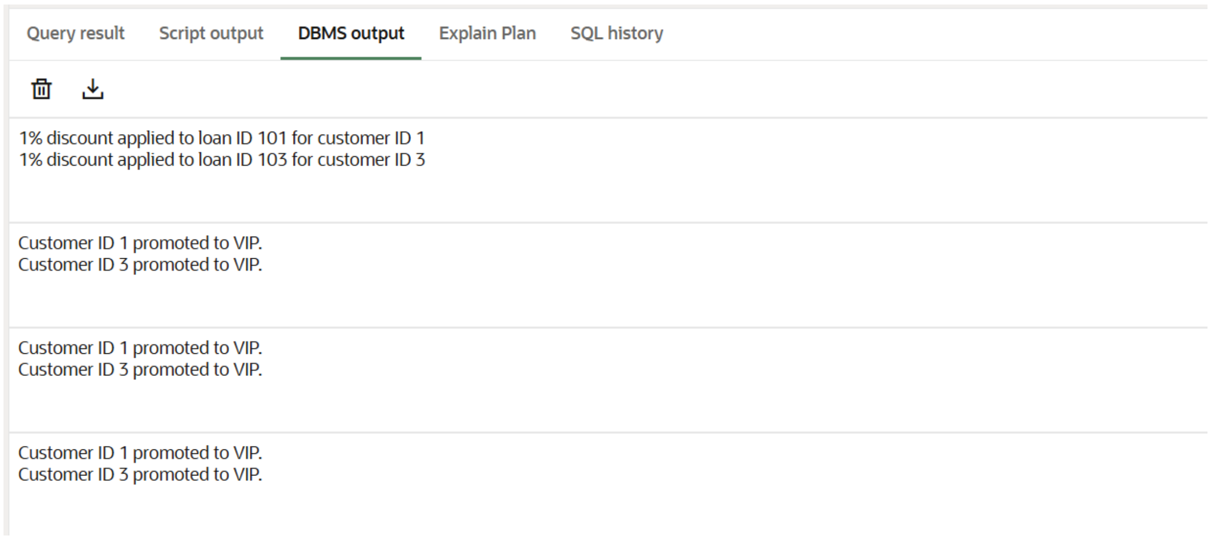
Question: Write a PL/SQL block that fetches all loans due in the next 30 days and prints a reminder message for each customer.



```
END;

BEGIN
    FOR rec IN (
        SELECT c.name, l.loan_id, l.due_date
        FROM customers c
        JOIN loans l ON c.customer_id = l.customer_id
        WHERE l.due_date <= SYSDATE + 30
    ) LOOP
        DBMS_OUTPUT.PUT_LINE('Reminder: Loan ID ' || rec.loan_id || ' for ' || rec.name || ' is due on ' || TO_CHAR(rec
    END LOOP;
END;
select * from loans
```

	LOAN_ID	CUSTOMER_ID	INTEREST_RATE	DUE_DATE
1	101	1	6.5	7/10/2025, 11:19:08 AM
2	102	2	8	8/4/2025, 11:19:08 AM
3	103	3	5.5	7/5/2025, 11:19:08 AM
4	104	4	7	6/30/2025, 11:19:08 AM

DBMS output:

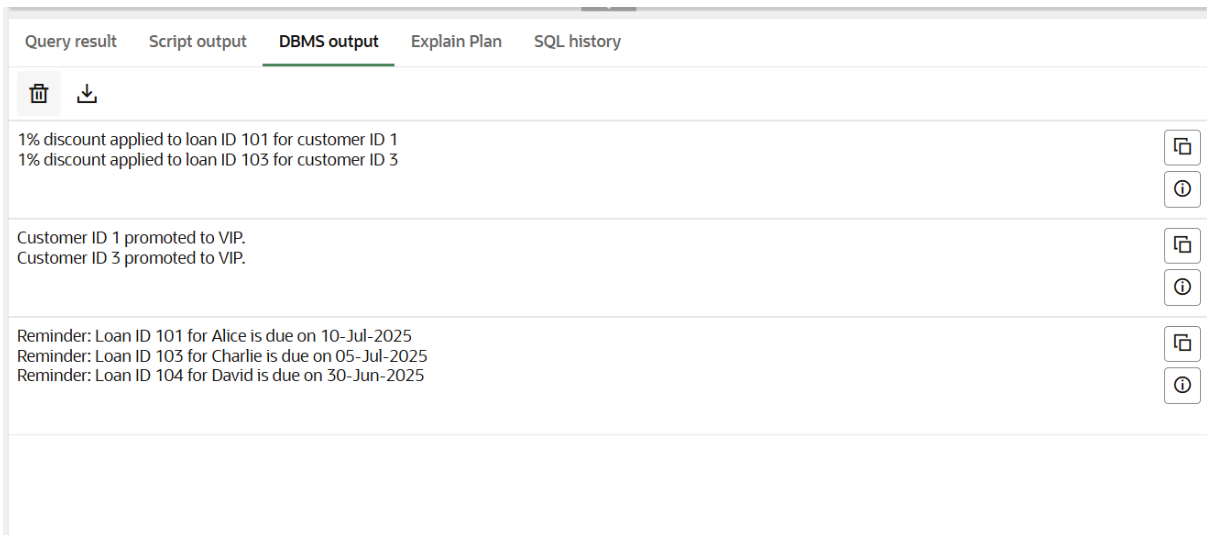


```
1% discount applied to loan ID 101 for customer ID 1
1% discount applied to loan ID 103 for customer ID 3
```

```
Customer ID 1 promoted to VIP.
Customer ID 3 promoted to VIP.
```

```
Customer ID 1 promoted to VIP.
Customer ID 3 promoted to VIP.
```

```
Customer ID 1 promoted to VIP.
Customer ID 3 promoted to VIP.
```



```
1% discount applied to loan ID 101 for customer ID 1
1% discount applied to loan ID 103 for customer ID 3
```

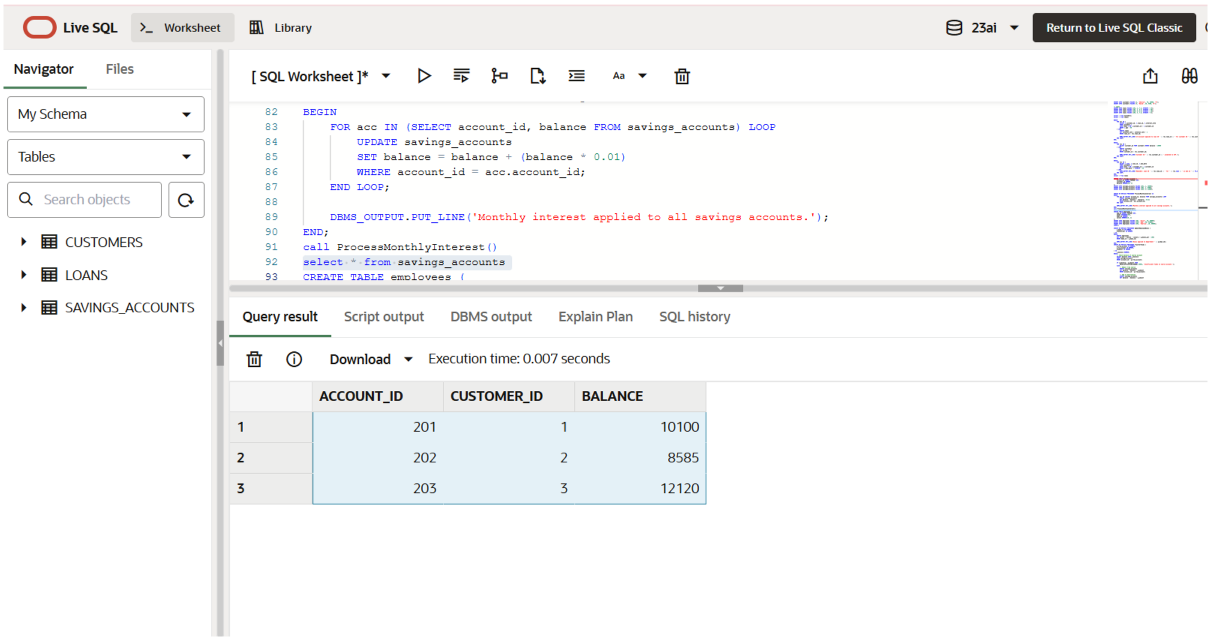
```
Customer ID 1 promoted to VIP.
Customer ID 3 promoted to VIP.
```

```
Reminder: Loan ID 101 for Alice is due on 10-Jul-2025
Reminder: Loan ID 103 for Charlie is due on 05-Jul-2025
Reminder: Loan ID 104 for David is due on 30-Jun-2025
```

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.



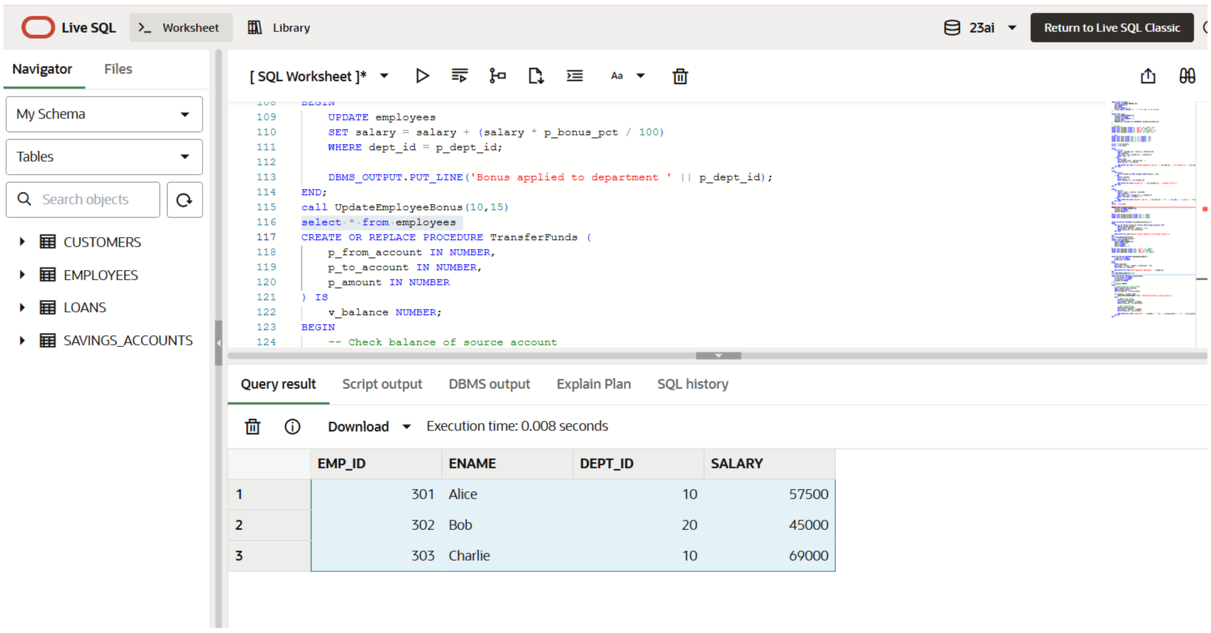
```sql
BEGIN
    FOR acc IN (SELECT account_id, balance FROM savings_accounts) LOOP
        UPDATE savings_accounts
        SET balance = balance + (balance * 0.01)
        WHERE account_id = acc.account_id;
    END LOOP;

    DBMS_OUTPUT.PUT_LINE('Monthly interest applied to all savings accounts.');
END;
call ProcessMonthlyInterest()
select * from savings_accounts
CREATE TABLE employees (
```

Execution time: 0.007 seconds

	ACCOUNT_ID	CUSTOMER_ID	BALANCE
1	201	1	10100
2	202	2	8585
3	203	3	12120

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.



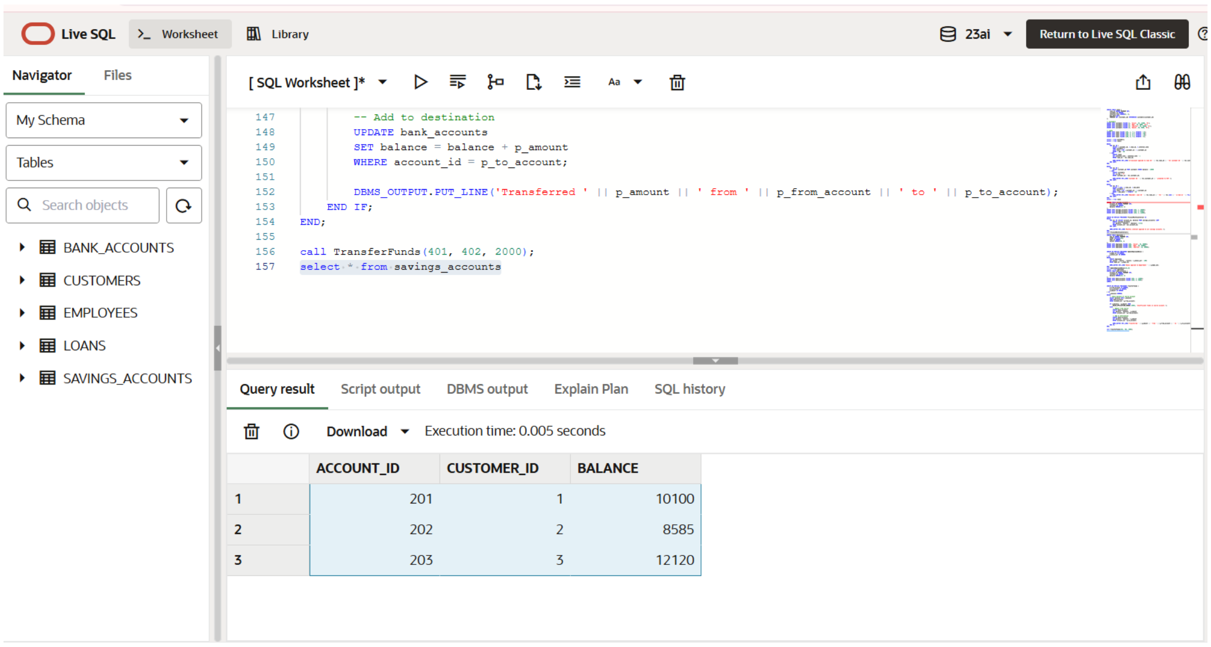
```sql
BEGIN
    UPDATE employees
    SET salary = salary + (salary * p_bonus_pct / 100)
    WHERE dept_id = p_dept_id;

    DBMS_OUTPUT.PUT_LINE('Bonus applied to department ' || p_dept_id);
END;
call UpdateEmployeeBonus(10,15)
select * from employees
CREATE OR REPLACE PROCEDURE TransferFunds (
    p_from_account IN NUMBER,
    p_to_account IN NUMBER,
    p_amount IN NUMBER
) IS
    v_balance NUMBER;
BEGIN
    -- Check balance of source account
```

Execution time: 0.008 seconds

	EMP_ID	ENAME	DEPT_ID	SALARY
1	301	Alice	10	57500
2	302	Bob	20	45000
3	303	Charlie	10	69000

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

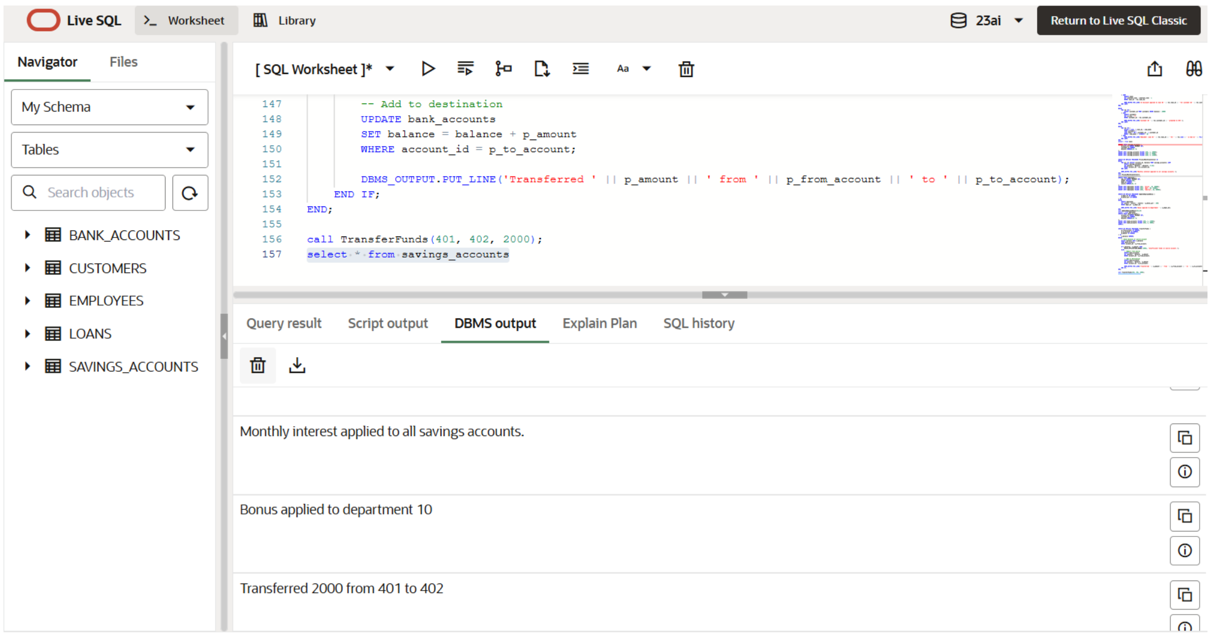
Question: Write a stored procedure TransferFunds that transfers a specified amount from one account to another, checking that the source account has sufficient balance before making the transfer.



```
            -- Add to destination
            UPDATE bank_accounts
            SET balance = balance + p_amount
            WHERE account_id = p_to_account;

            DBMS_OUTPUT.PUT_LINE('Transferred ' || p_amount || ' from ' || p_from_account || ' to ' || p_to_account);
        END IF;
END;

call TransferFunds(401, 402, 2000);
select * from savings_accounts
```

	ACCOUNT_ID	CUSTOMER_ID	BALANCE
1	201	1	10100
2	202	2	8585
3	203	3	12120

DBMS output:



```
            -- Add to destination
            UPDATE bank_accounts
            SET balance = balance + p_amount
            WHERE account_id = p_to_account;

            DBMS_OUTPUT.PUT_LINE('Transferred ' || p_amount || ' from ' || p_from_account || ' to ' || p_to_account);
        END IF;
END;

call TransferFunds(401, 402, 2000);
select * from savings_accounts
```

```
Monthly interest applied to all savings accounts.

Bonus applied to department 10

Transferred 2000 from 401 to 402
```