**JAVA FSE WEEK 2**

**PLSQL**

**Mandatory Hands-On Exercises**

**Exercise 1: Control Structures**

**Solution:**

**Step 1- Creating the Customers and Loans table:**

CREATE TABLE customers (

customer\_id NUMBER PRIMARY KEY,

name VARCHAR2(50),

age NUMBER,

balance NUMBER,

is\_vip VARCHAR2(5)

);

CREATE TABLE loans (

loan\_id NUMBER PRIMARY KEY,

customer\_id NUMBER REFERENCES customers(customer\_id),

interest\_rate NUMBER(5,2),

due\_date DATE

);

**Step 2 : Inserting Rows :**

* Inserting into Customers:

INSERT INTO customers VALUES (1, 'Alice', 65, 15000, 'FALSE');

INSERT INTO customers VALUES (2, 'Bob', 45, 8000, 'FALSE');

INSERT INTO customers VALUES (3, 'Charlie', 70, 12000, 'FALSE');

INSERT INTO customers VALUES (4, 'Diana', 35, 10500, 'FALSE');

* Inserting into Loans:

INSERT INTO loans VALUES (101, 1, 7.5, SYSDATE + 10); -- Due soon

INSERT INTO loans VALUES (102, 2, 8.0, SYSDATE + 40); -- Not due soon

INSERT INTO loans VALUES (103, 3, 6.5, SYSDATE + 25); -- Due soon

INSERT INTO loans VALUES (104, 4, 7.0, SYSDATE + 5); -- Due very soon

**Senario1: Applying 1% Discount to Interest for Customers Over 60 :**

BEGIN

FOR cust\_rec IN (SELECT customer\_id FROM customers WHERE age > 60) LOOP

UPDATE loans

SET interest\_rate = interest\_rate - 1

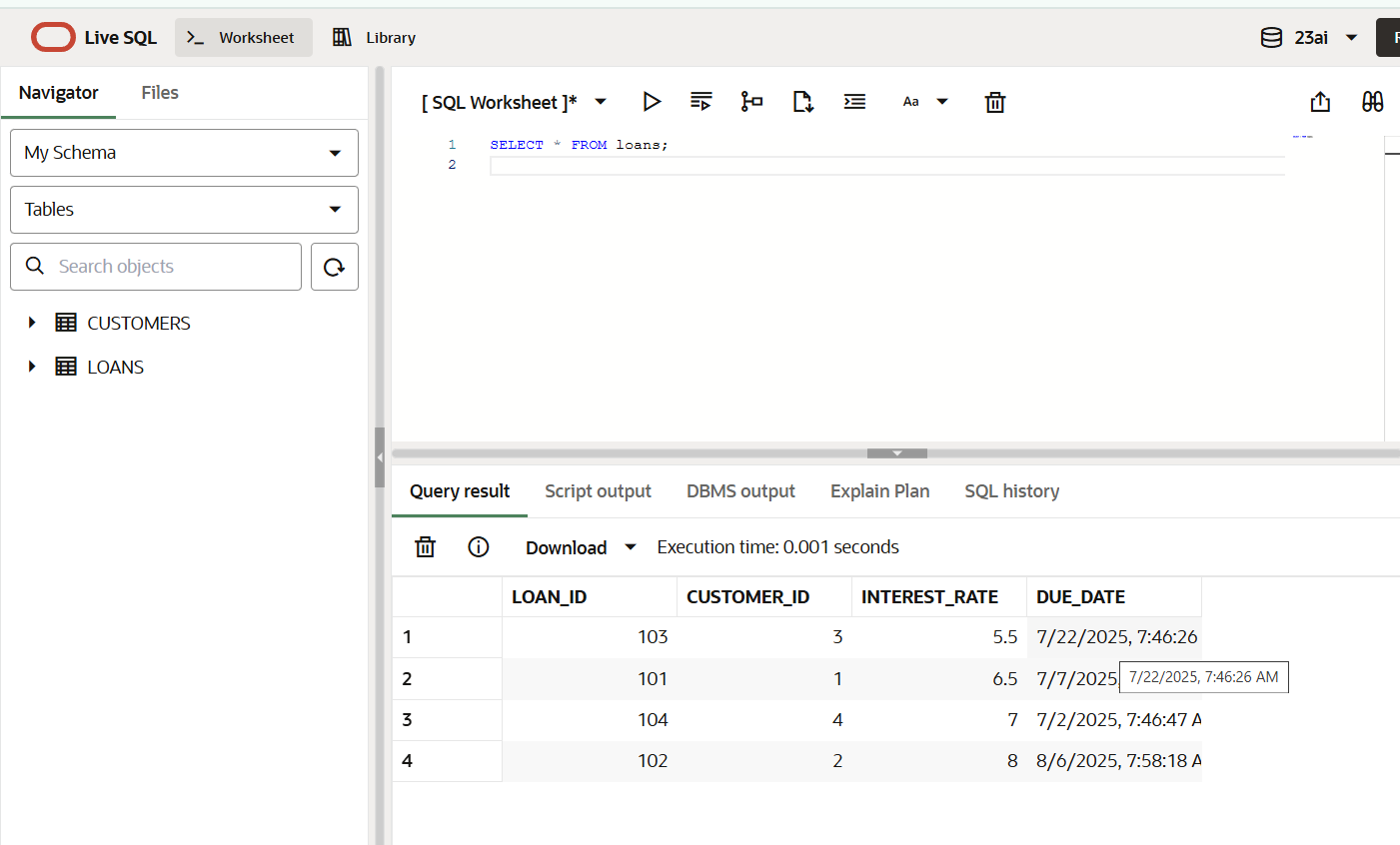
WHERE customer\_id = cust\_rec.customer\_id;

END LOOP;

COMMIT;

END;

**OUTPUT:**

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

BEGIN

FOR cust\_rec IN (

SELECT customer\_id FROM customers WHERE balance > 10000

) LOOP

UPDATE customers

SET is\_vip = 'TRUE'

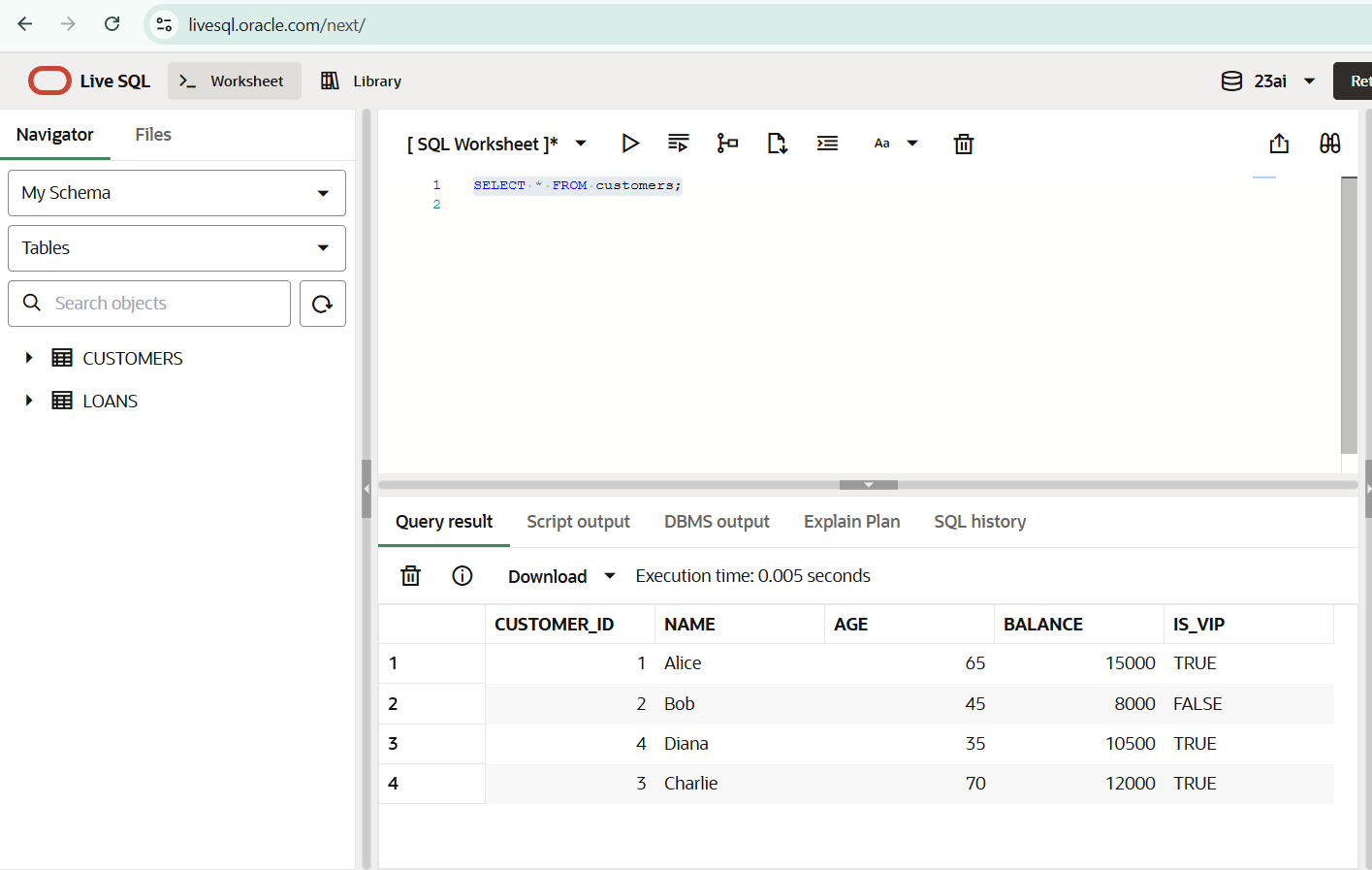
WHERE customer\_id = cust\_rec.customer\_id;

END LOOP;

COMMIT;

END;

**Output:**



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

SET SERVEROUTPUT ON;

BEGIN

FOR loan\_rec IN (

SELECT loan\_id, customer\_id, due\_date

FROM loans

WHERE due\_date <= SYSDATE + 30

) LOOP

DECLARE

v\_name VARCHAR2(50);

BEGIN

SELECT name INTO v\_name

FROM customers

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

DBMS\_OUTPUT.PUT\_LINE('Reminder: Dear ' || v\_name ||

', your loan (ID: ' || loan\_rec.loan\_id ||

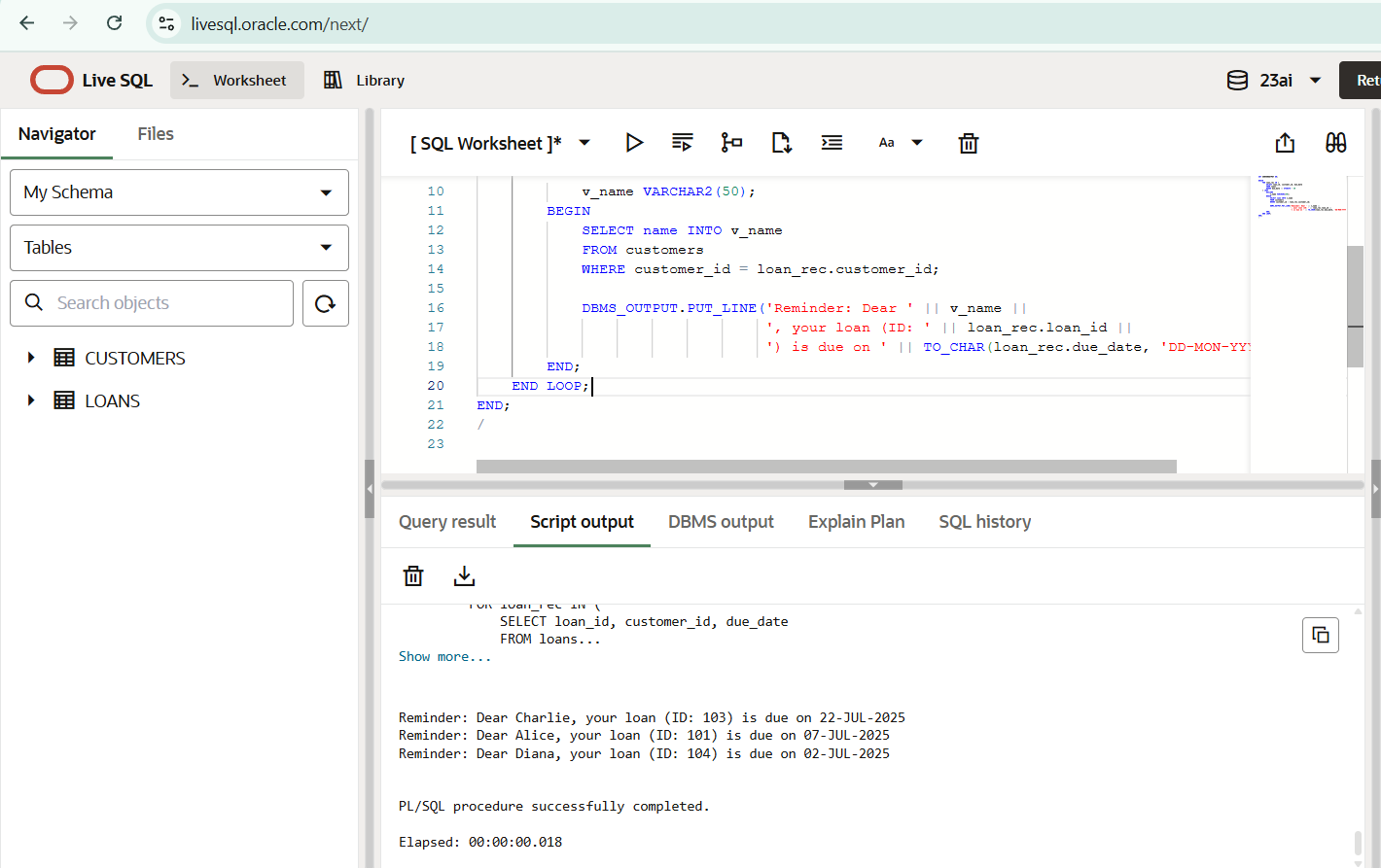
') is due on ' || TO\_CHAR(loan\_rec.due\_date, 'DD-MON-YYYY'));

END;

END LOOP;

END;

**Output:**



**Exercise 3: Stored Procedures**

**Solution :**

**Step 1: Create Tables:**

CREATE TABLE accounts (

account\_id NUMBER PRIMARY KEY,

customer\_id NUMBER,

account\_type VARCHAR2(20),

balance NUMBER

);

CREATE TABLE employees (

emp\_id NUMBER PRIMARY KEY,

name VARCHAR2(50),

department\_id NUMBER,

salary NUMBER

);

**Step 2: Inserting rows :**

-- Insert accounts

INSERT INTO accounts VALUES (101, 1, 'savings', 10000);

INSERT INTO accounts VALUES (102, 2, 'savings', 8000);

INSERT INTO accounts VALUES (103, 3, 'current', 5000);

-- Insert employees

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

INSERT INTO employees VALUES (2, 'Bob', 102, 60000);

INSERT INTO employees VALUES (3, 'Charlie', 101, 55000);

**Senario 1: Applying 1% interest to all Savings accounts:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

UPDATE accounts

SET balance = balance + (balance \* 0.01)

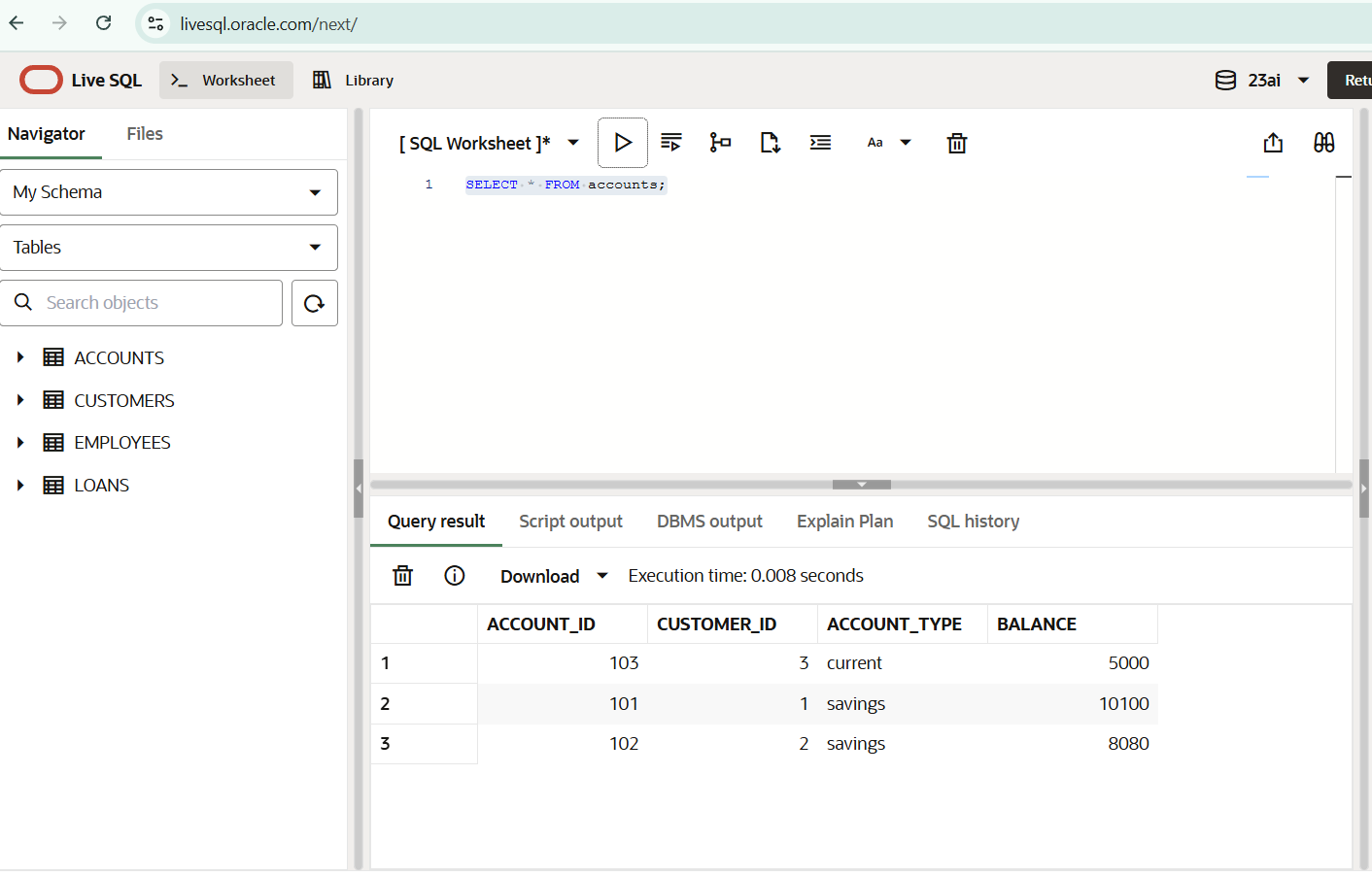
WHERE account\_type = 'savings';

COMMIT;

END;

EXEC ProcessMonthlyInterest;

**Output:**

****

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

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_dept\_id IN NUMBER,

p\_bonus\_pct IN NUMBER -- e.g. pass 0.10 for 10%

) IS

BEGIN

UPDATE employees

SET salary = salary + (salary \* p\_bonus\_pct)

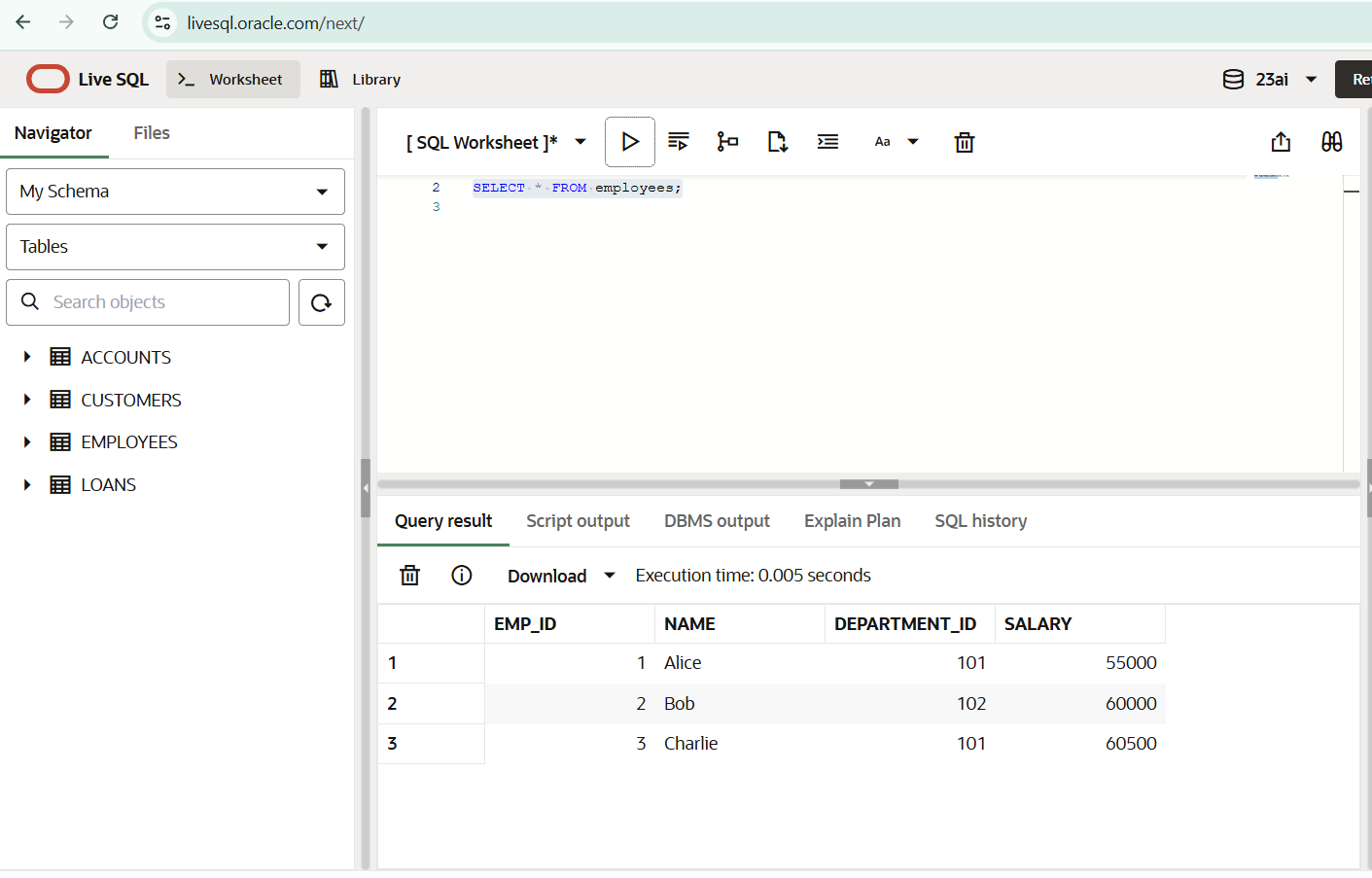
WHERE department\_id = p\_dept\_id;

COMMIT;

END;

EXEC UpdateEmployeeBonus(101, 0.10);

**Output :**

****

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

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 if source account has enough balance

SELECT balance INTO v\_balance

FROM accounts

WHERE account\_id = p\_from\_account;

IF v\_balance < p\_amount THEN

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

END IF;

-- Deduct from source account

UPDATE accounts

SET balance = balance - p\_amount

WHERE account\_id = p\_from\_account;

-- Add to destination account

UPDATE accounts

SET balance = balance + p\_amount

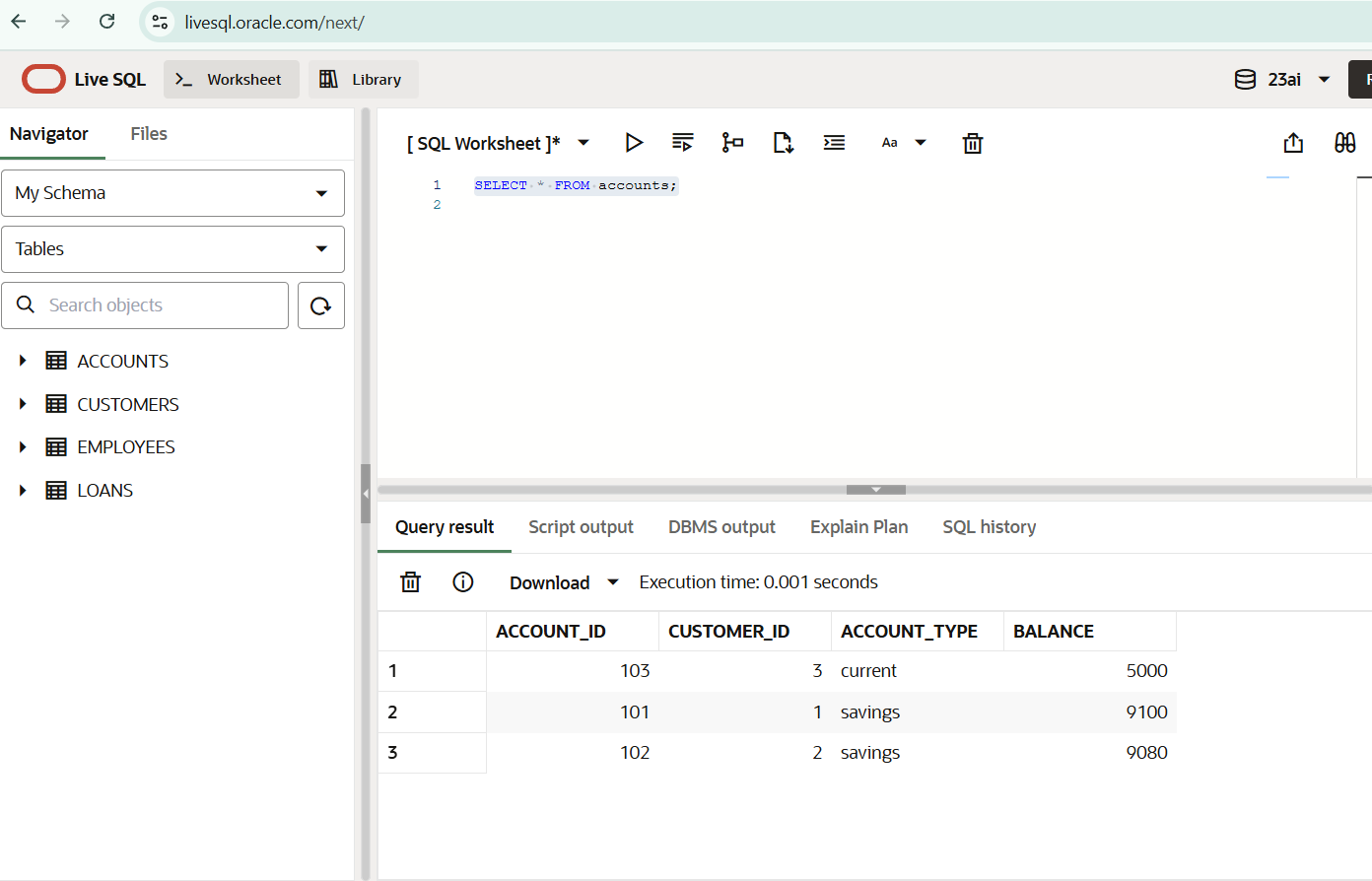
WHERE account\_id = p\_to\_account;

COMMIT;

END;

EXEC TransferFunds(101, 102, 1000);

**Output:**

****