

ASSIGNMENT-09

NAME-ANUBHAV ANAND

ENROLLMENT NUMBER-2020CSB102

SUBJECT-ASSIGNMENT-9 OF DBMS

G-Suite [id -](#)

2020CSB102.anubhav@students.iiests.ac.in

Table which I have used in this assignment-

Dept-

```
SQL> select * from dept;
```

DEPTNO	DNAME	LOC
20	Research	Dallas
30	Sales	Chicago
40	Operations	Boston
10	Accounting	New York

```
SQL> █
```

Emp-

```
mysql> select * from emp;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	Smith	Clerk	7902	1980-12-17	800	NULL	20
7499	Allen	Salesman	7698	1981-02-20	1600	300	30
7521	Ward	Salesman	7698	1981-02-22	1250	500	30
7566	Jones	Manager	7839	1981-04-02	2975	NULL	20
7654	Martin	Salesman	7698	1981-09-28	1250	1400	30
7698	Blake	Manager	7839	1981-05-01	2850	NULL	30
7782	Clark	Manager	7839	1981-06-09	2450	NULL	10
7788	Scott	Analyst	7566	1982-12-09	3000	NULL	20
7839	King	President	NULL	1981-11-17	5000	NULL	10
7844	Turner	Salesman	7698	1981-09-08	1500	0	30
7876	Adams	Clerk	7788	1983-01-12	1100	NULL	20
7900	James	Clerk	7698	1981-12-03	950	NULL	30
7902	Ford	Analyst	7566	1981-12-04	3000	NULL	20
7934	Miller	Clerk	7782	1982-01-23	1300	NULL	10

```
14 rows in set (0.07 sec)
```

Department-

```
SQL> select * from department;
```

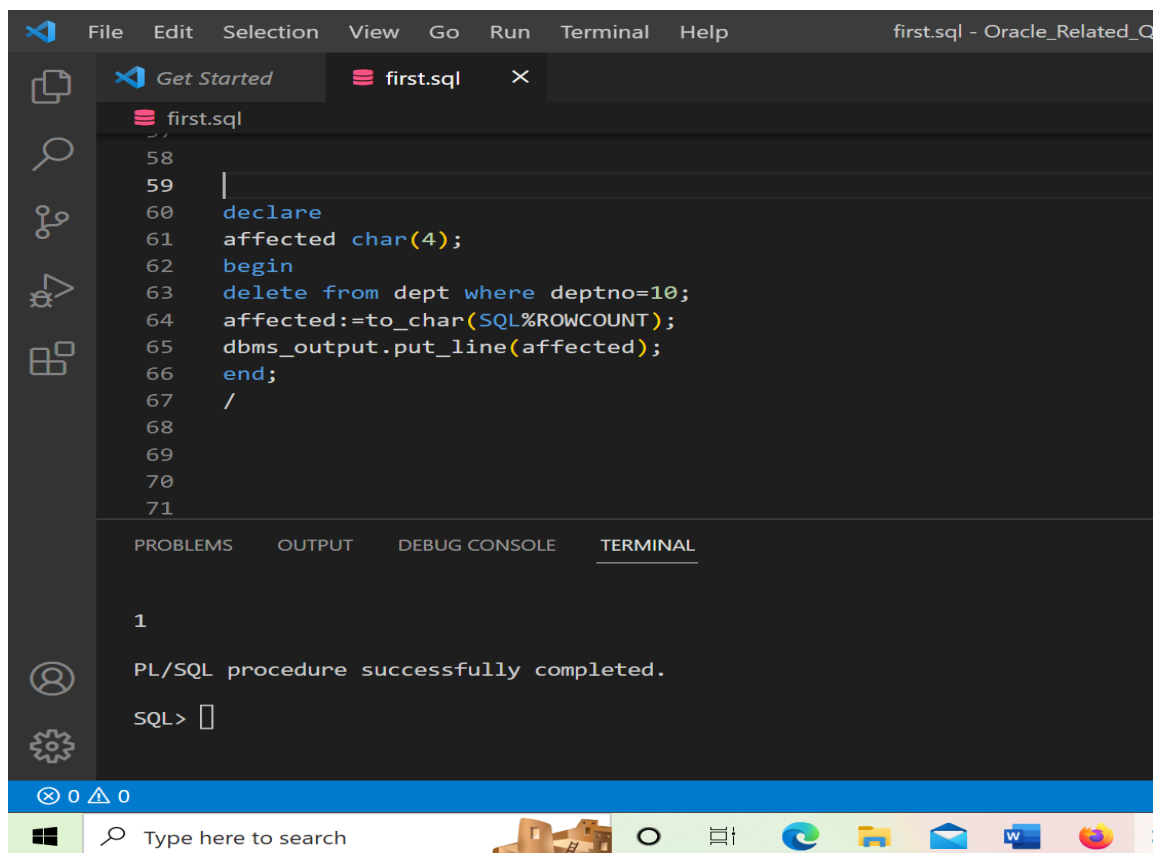
DEPTNO	DNAME	LOC
20	Research	Dallas
30	Sales	Chicago
40	Operations	Boston
10	Accounting	New York

- Department and Dept table are storing same values

Part(A)-ASSIGNMENT ON CURSOR:-

Q1. Take the help of implicit cursor to print the total number of rows deleted after a deletion operation performed on a table.

Ans-Code-

A screenshot of a code editor window titled 'first.sql - Oracle_Related_Q'. The editor shows a PL/SQL script in a file named 'first.sql'. The script is as follows:

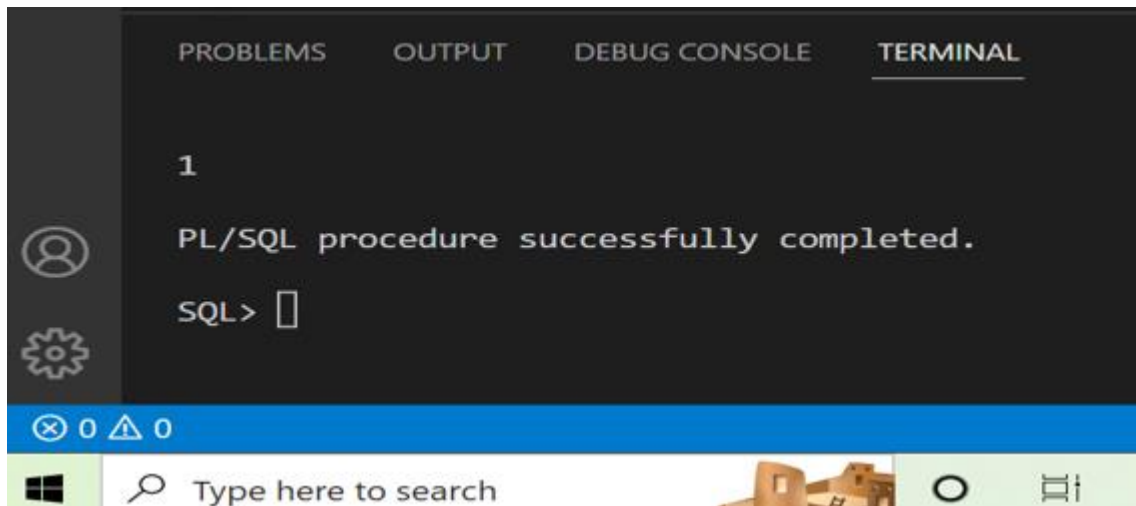
```
58  
59  
60 declare  
61 affected char(4);  
62 begin  
63 delete from dept where deptno=10;  
64 affected:=to_char(SQL%ROWCOUNT);  
65 dbms_output.put_line(affected);  
66 end;  
67 /  
68  
69  
70  
71
```

The bottom panel of the IDE is divided into 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', and 'TERMINAL'. The 'TERMINAL' tab is active, showing the output of the script execution:

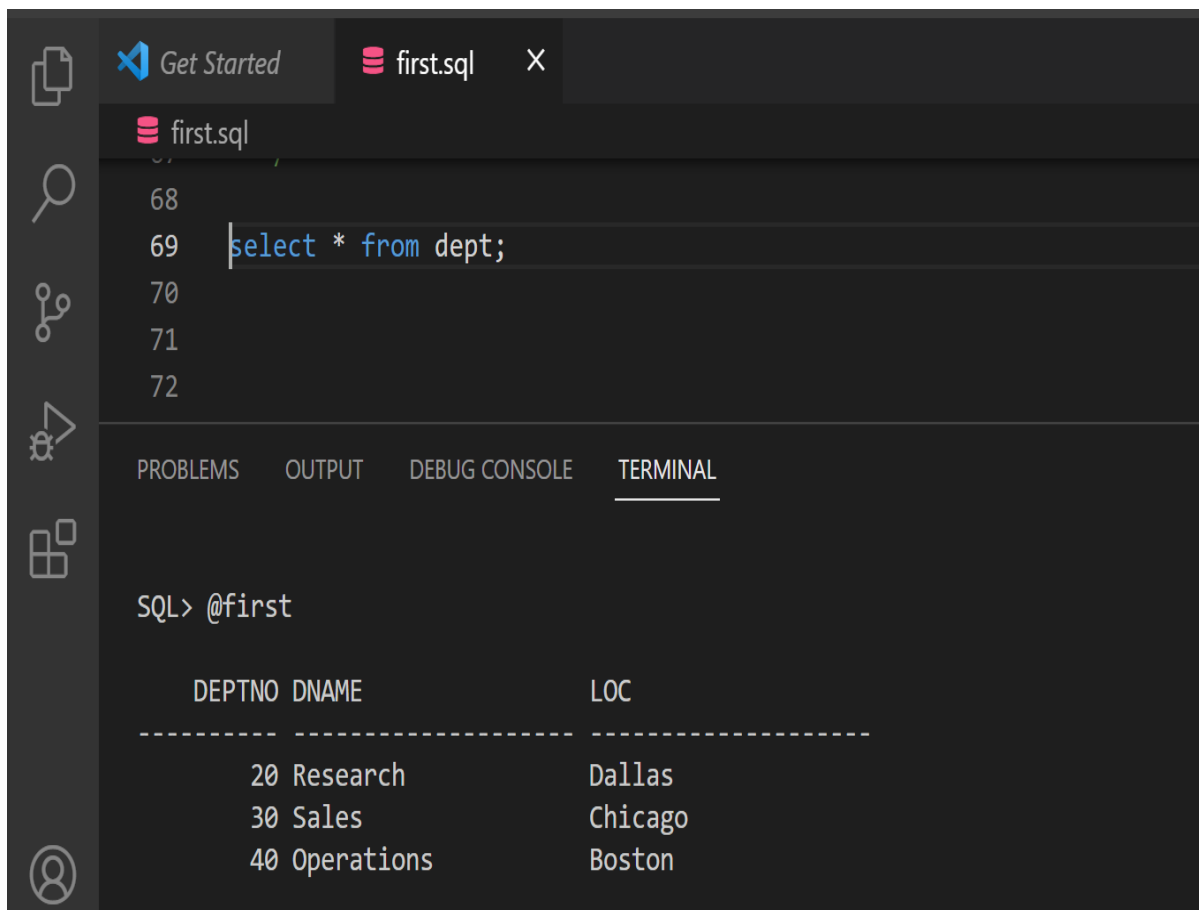
```
1  
  
PL/SQL procedure successfully completed.  
  
SQL> 
```

The Windows taskbar is visible at the bottom of the screen.

Output-



After Deleting from dept table using implicit cursor-



Q2. Create a table X that contains two attributes A(number(2)) and B(varchar2(25)). Now,create an explicit cursor that will help you to populate the table X with the value from DEPT table (Ref. Assignment No. - 2) where A is same as deptno and b is same as dname.

Ans-Code-

```
second.sql
1  create table X(A number(2),B varchar2(25));
2
3  declare
4  cursor cur2 is
5  select deptno,dname
6  from dept;
7  a_m dept.deptno%type;
8  b_m dept.dname%type;
9  begin
10 open cur2;
11 if cur2%isopen then
12 loop
13 fetch cur2 into a_m,b_m;
14 exit when cur2%notfound;
15 if cur2%found then
16 insert into x values(a_m,b_m);
17 end if;
18 end loop;
19 commit;
20 else
21 dbms_output.put_line('Unable to open cursor');
22 end if;
23 close cur2;
24 end;
25 /
```

Table X after creation-

```
26  select * from X;  
27
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

```
SQL> @second
```

```
      A B
```

```
-----
```

```
20 Research
```

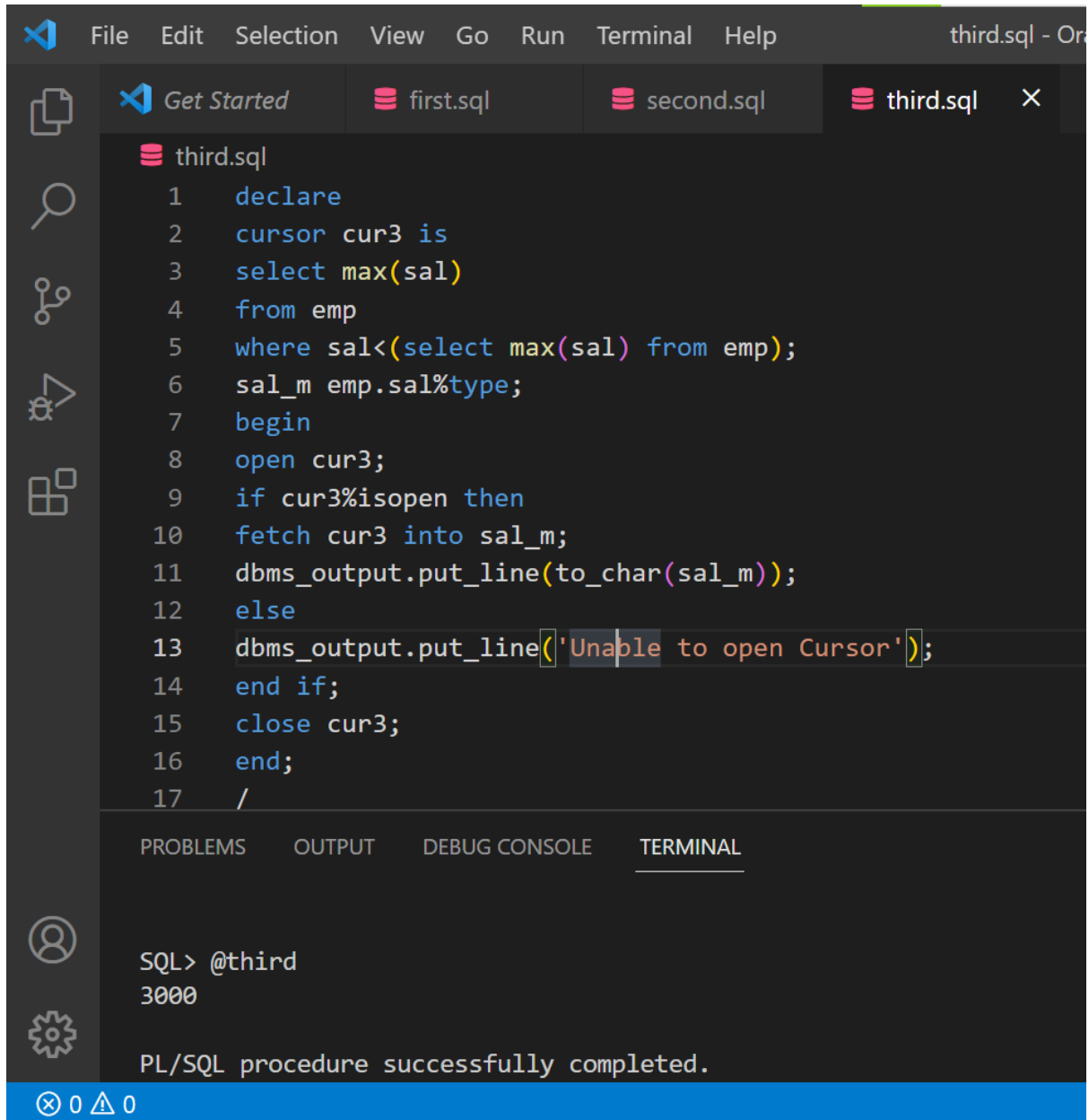
```
30 Sales
```

```
40 Operations
```

```
10 Accounting
```

Q3. Print the second highest salary of the EMP table (Ref. Assignment No. - 2) using an explicit cursor.

Ans-



```
third.sql - Or
File Edit Selection View Go Run Terminal Help
Get Started first.sql second.sql third.sql X
third.sql
1 declare
2 cursor cur3 is
3 select max(sal)
4 from emp
5 where sal < (select max(sal) from emp);
6 sal_m emp.sal%type;
7 begin
8 open cur3;
9 if cur3%isopen then
10 fetch cur3 into sal_m;
11 dbms_output.put_line(to_char(sal_m));
12 else
13 dbms_output.put_line('Unable to open Cursor');
14 end if;
15 close cur3;
16 end;
17 /

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
SQL> @third
3000
PL/SQL procedure successfully completed.
0 0 0
```


Output-

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
SQL> @third  
3000
```

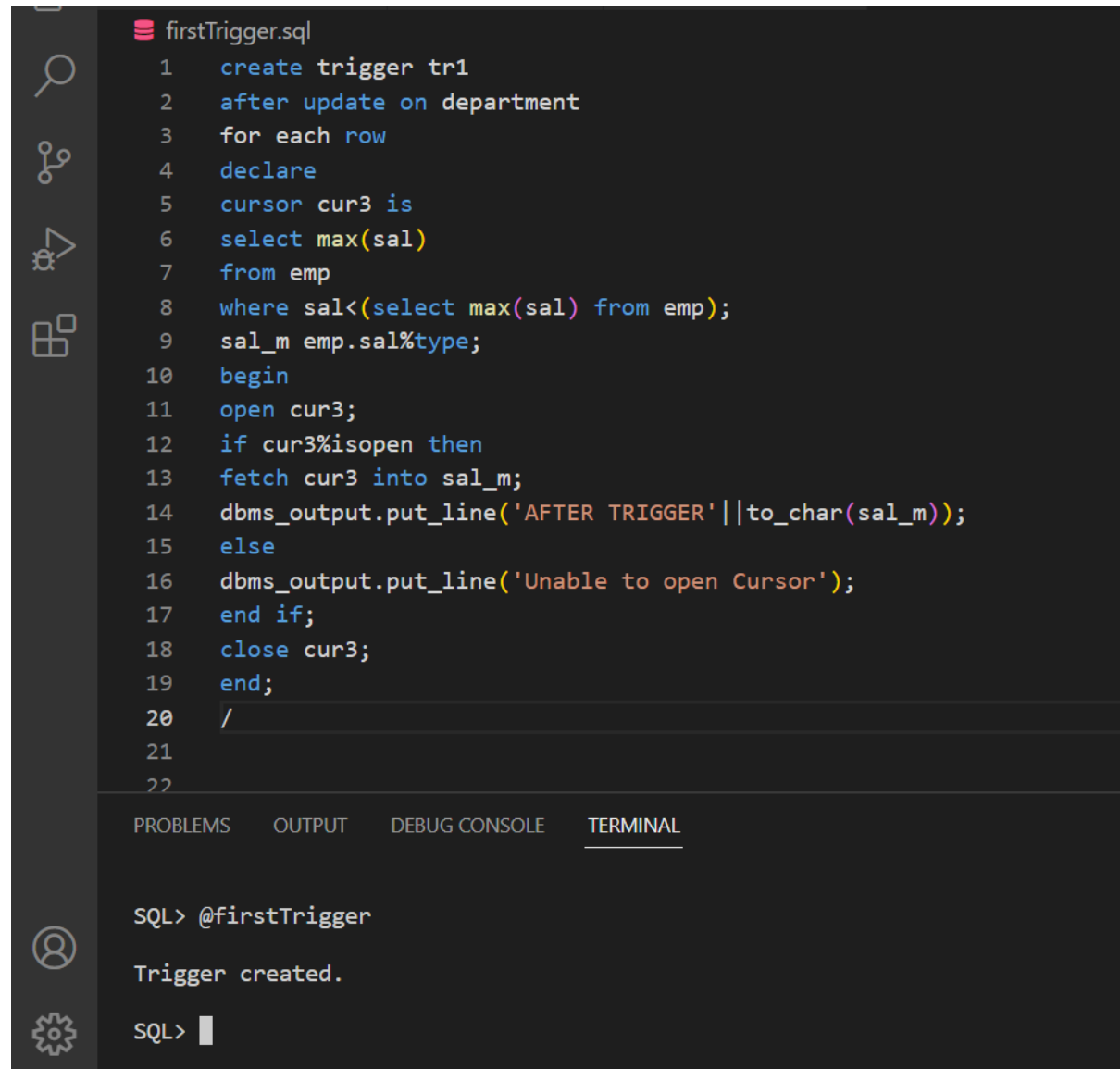
```
PL/SQL procedure successfully completed.
```

```
SQL> █
```

Part(B)-ASSIGNMENT ON TRIGGERS-

Q1. Create a row trigger that will display the second highest salary of EMP table (or perform any other valid action) after any updation performed on the DEPT table.

Ans-Code-After Creating Trigger-



```
firstTrigger.sql
1  create trigger tr1
2  after update on department
3  for each row
4  declare
5  cursor cur3 is
6  select max(sal)
7  from emp
8  where sal<(select max(sal) from emp);
9  sal_m emp.sal%type;
10 begin
11 open cur3;
12 if cur3%isopen then
13 fetch cur3 into sal_m;
14 dbms_output.put_line('AFTER TRIGGER'||to_char(sal_m));
15 else
16 dbms_output.put_line('Unable to open Cursor');
17 end if;
18 close cur3;
19 end;
20 /
21
22
```

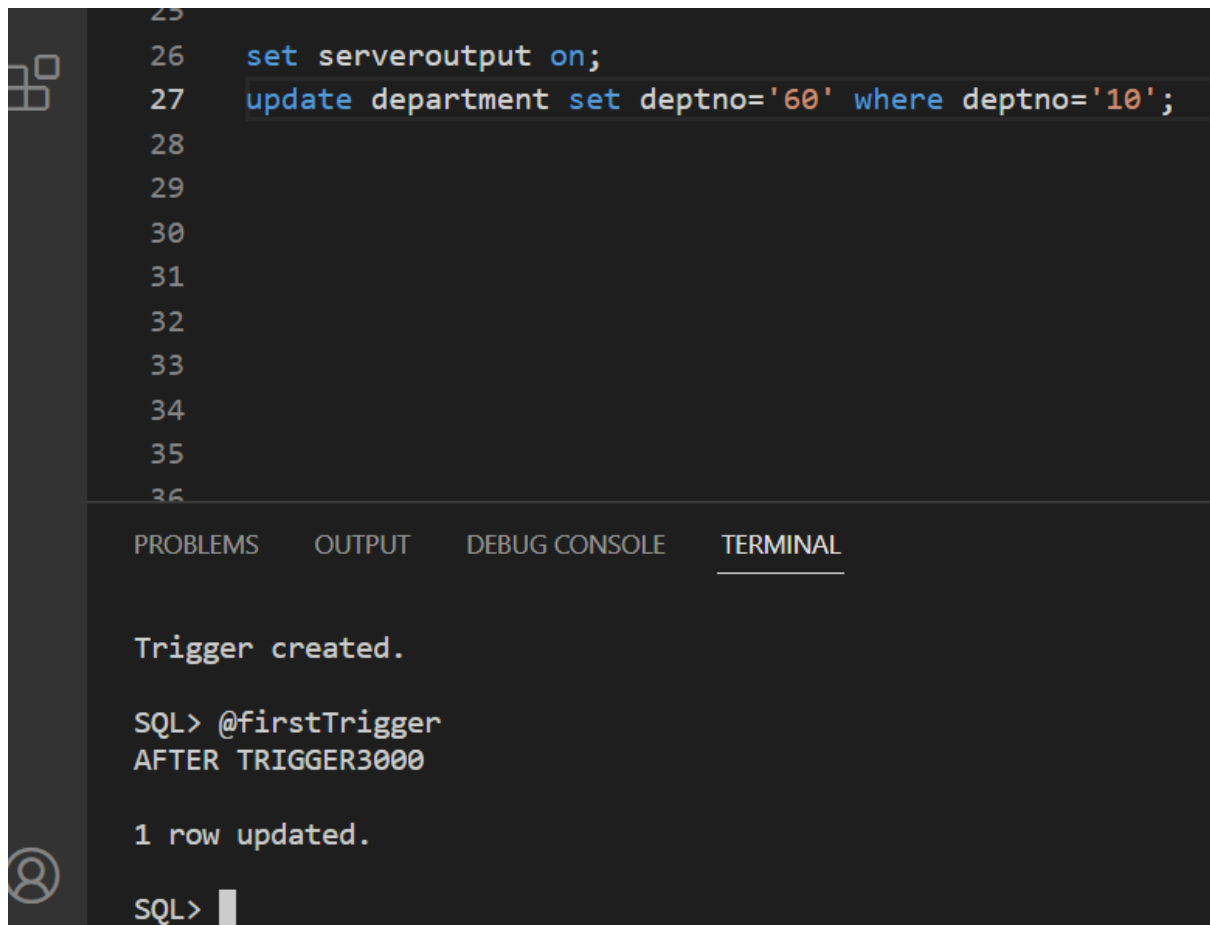
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
SQL> @firstTrigger

Trigger created.

SQL> 
```

After updating any value of dept table-



```
25
26  set serveroutput on;
27  update department set deptno='60' where deptno='10';
28
29
30
31
32
33
34
35
36
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

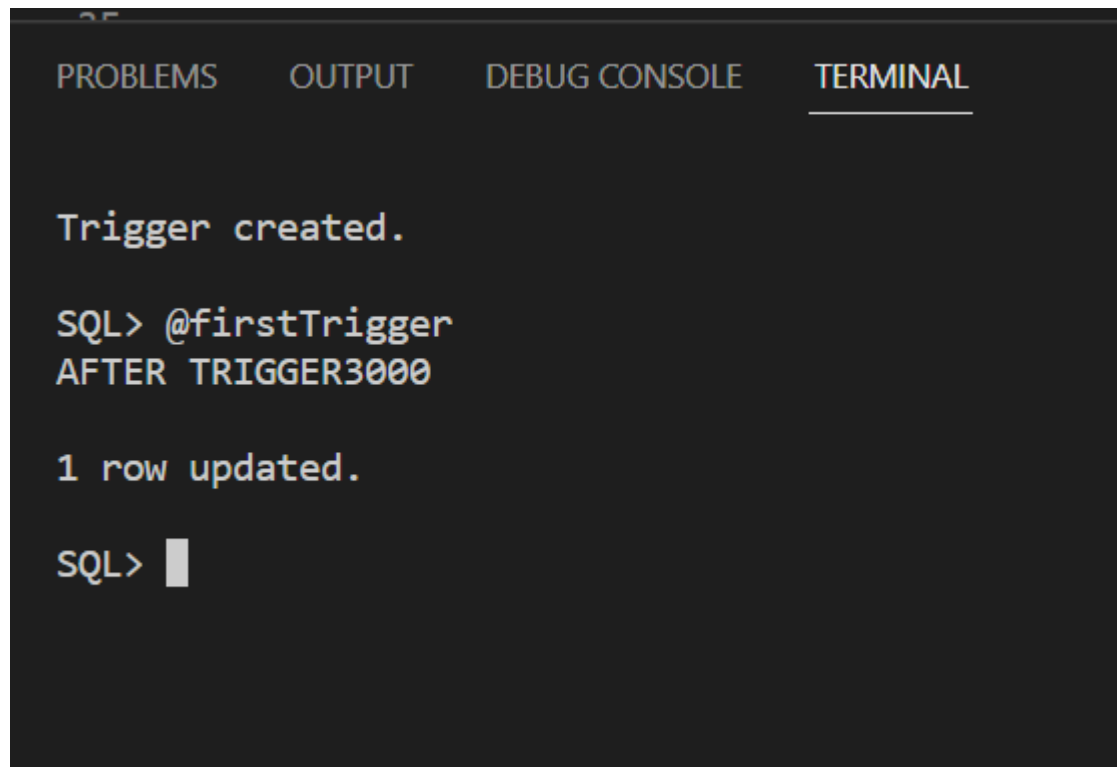
Trigger created.

SQL> @firstTrigger
AFTER TRIGGER3000

1 row updated.

SQL> █

Output-

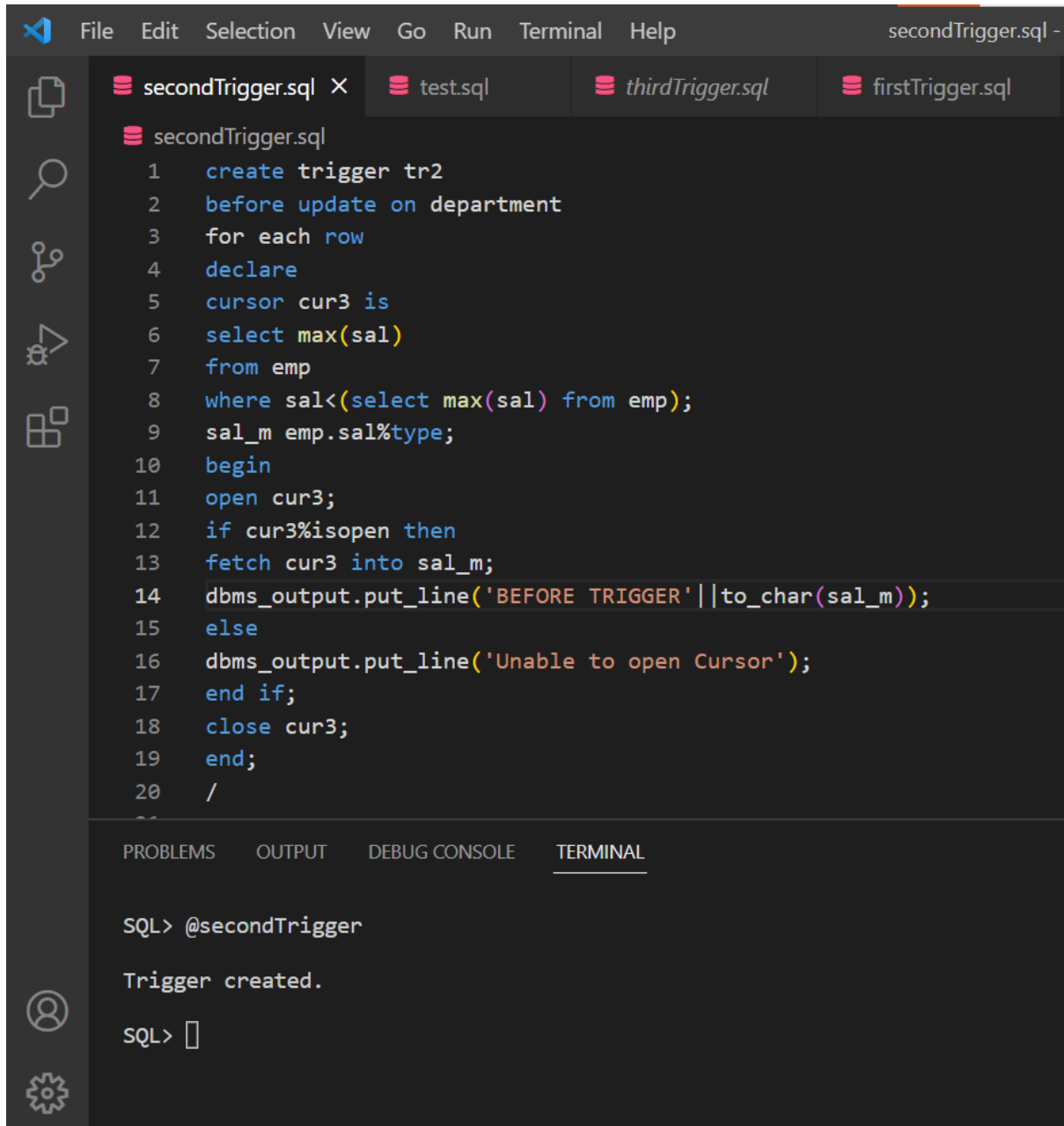


The screenshot shows the 'OUTPUT' tab of a SQL Server Enterprise Manager interface. The tab is selected and underlined. The output text is as follows:

```
Trigger created.  
  
SQL> @firstTrigger  
AFTER TRIGGER3000  
  
1 row updated.  
  
SQL> █
```

Q2. Repeat problem no. 1 for before trigger.

Ans-Code-



The screenshot shows an IDE with a dark theme. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The title bar on the right says 'secondTrigger.sql -'. There are four tabs open: 'secondTrigger.sql' (active), 'test.sql', 'thirdTrigger.sql', and 'firstTrigger.sql'. The main editor area contains the following SQL code:

```
1 create trigger tr2
2 before update on department
3 for each row
4 declare
5 cursor cur3 is
6 select max(sal)
7 from emp
8 where sal < (select max(sal) from emp);
9 sal_m emp.sal%type;
10 begin
11 open cur3;
12 if cur3%isopen then
13 fetch cur3 into sal_m;
14 dbms_output.put_line('BEFORE TRIGGER' || to_char(sal_m));
15 else
16 dbms_output.put_line('Unable to open Cursor');
17 end if;
18 close cur3;
19 end;
20 /
```

Below the editor is a panel with four tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL. The TERMINAL tab is active, showing the following text:

```
SQL> @secondTrigger

Trigger created.

SQL> 
```

After Updating the dept table-

```
22  update department set deptno='10' where deptno='60';
23
24
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

trigger created.

SQL> @secondTrigger
BEFORE TRIGGER3000
AFTER TRIGGER3000

1 row updated.

Due to updating output will be-

```
SQL> @secondTrigger  
BEFORE TRIGGER3000  
AFTER TRIGGER3000
```

1 row updated.

SQL>

⊗ 0 ⚠ 0

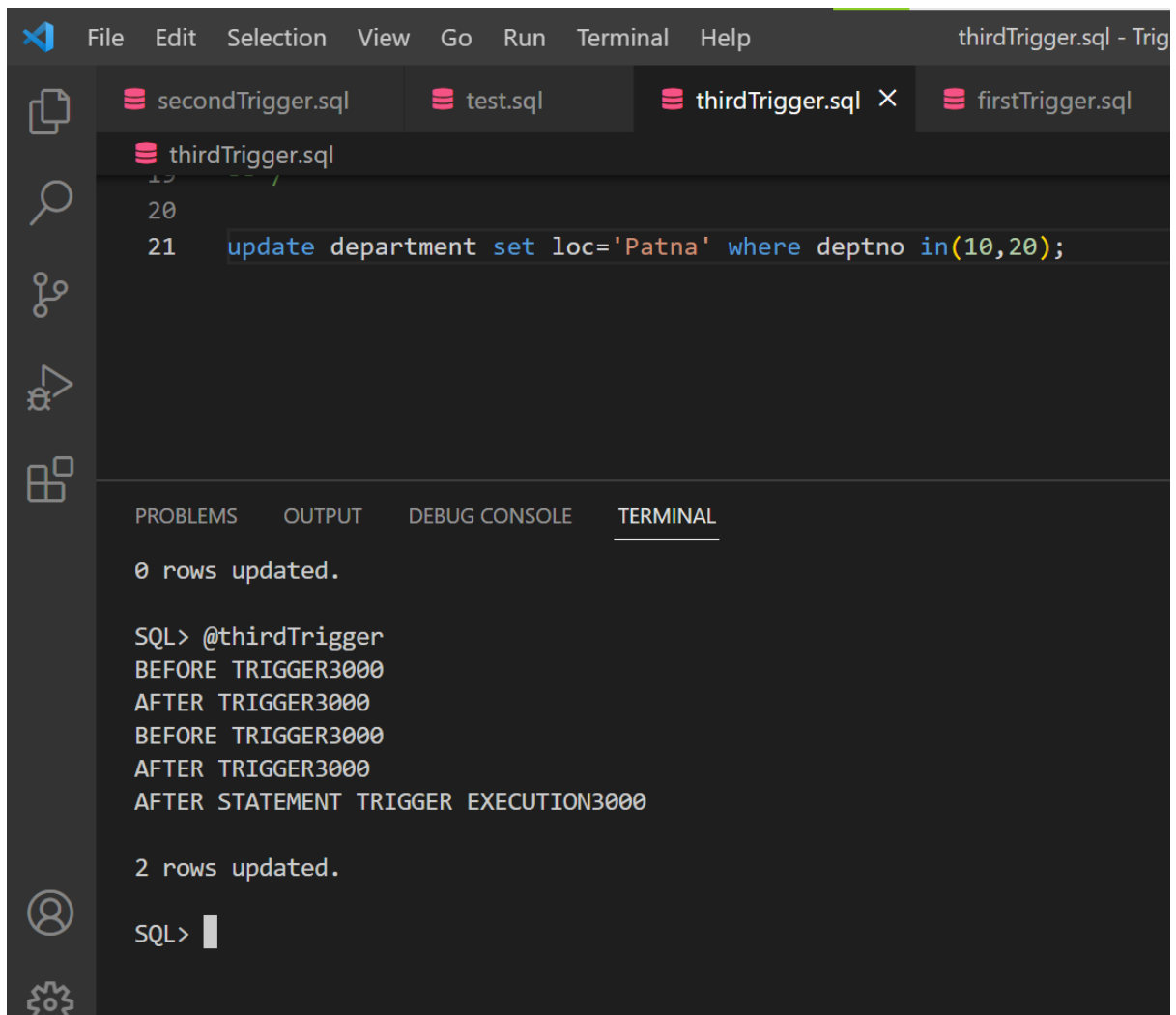
Q3. Modify the triggers of problem nos. 1 and 2 to statement triggers and observe the changes in the outputs.

Ans-Code-

To create trigger tr3-

```
thirdTrigger.sql - Trigger - Visual Studio
secondTrigger.sql test.sql thirdTrigger.sql X firstTrigger.sql
thirdTrigger.sql
1 create trigger tr3
2 after update on department
3 declare
4 cursor cur3 is
5 select max(sal)
6 from emp
7 where sal < (select max(sal) from emp);
8 sal_m emp.sal%type;
9 begin
10 open cur3;
11 if cur3%isopen then
12 fetch cur3 into sal_m;
13 dbms_output.put_line('AFTER STATEMENT TRIGGER EXECUTION' || to_char(sal_m));
14 else
15 dbms_output.put_line('Unable to open Cursor');
16 end if;
17 close cur3;
18 end;
19 /
20
```

After Updating Department table-



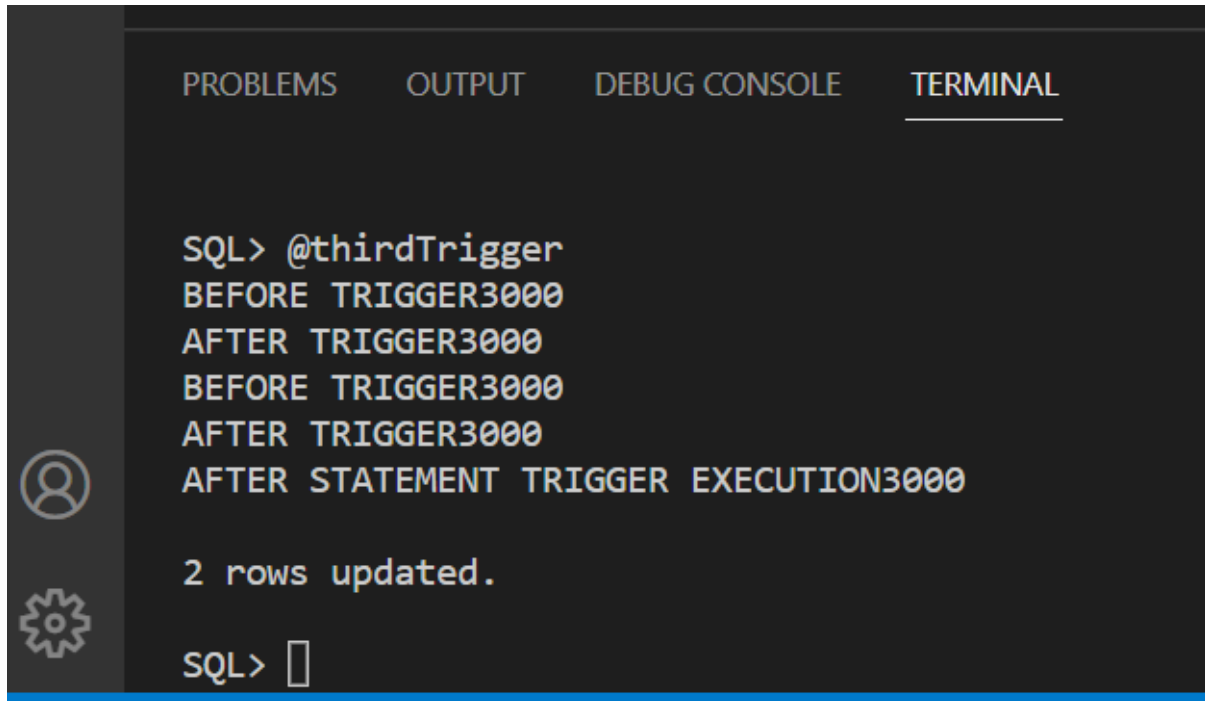
The screenshot shows an IDE with a dark theme. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The title bar on the right says 'thirdTrigger.sql - Trig'. The editor has four tabs: secondTrigger.sql, test.sql, thirdTrigger.sql (active), and firstTrigger.sql. The active tab contains the following SQL code:

```
19  -- 3000  
20  
21  update department set loc='Patna' where deptno in(10,20);
```

Below the editor is a panel with four tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL (active). The terminal shows the output of the SQL execution:

```
0 rows updated.  
  
SQL> @thirdTrigger  
BEFORE TRIGGER3000  
AFTER TRIGGER3000  
BEFORE TRIGGER3000  
AFTER TRIGGER3000  
AFTER STATEMENT TRIGGER EXECUTION3000  
  
2 rows updated.  
  
SQL> |
```


Output-



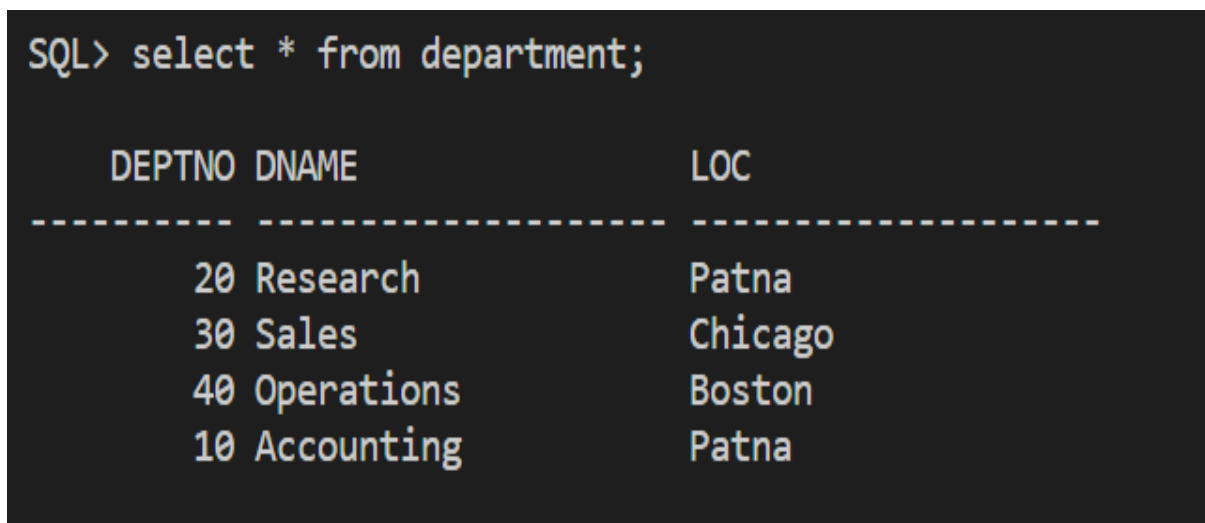
The screenshot shows a SQL IDE interface with a dark theme. At the top, there are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', and 'TERMINAL'. The 'TERMINAL' tab is selected and underlined. On the left side of the terminal, there are icons for a user profile and a settings gear. The terminal text shows the execution of a trigger named '@thirdTrigger', which is a BEFORE TRIGGER3000, AFTER TRIGGER3000, BEFORE TRIGGER3000, AFTER TRIGGER3000, and AFTER STATEMENT TRIGGER EXECUTION3000. The output indicates that 2 rows were updated. The prompt 'SQL>' is followed by a cursor.

```
SQL> @thirdTrigger
BEFORE TRIGGER3000
AFTER TRIGGER3000
BEFORE TRIGGER3000
AFTER TRIGGER3000
AFTER STATEMENT TRIGGER EXECUTION3000

2 rows updated.

SQL> 
```

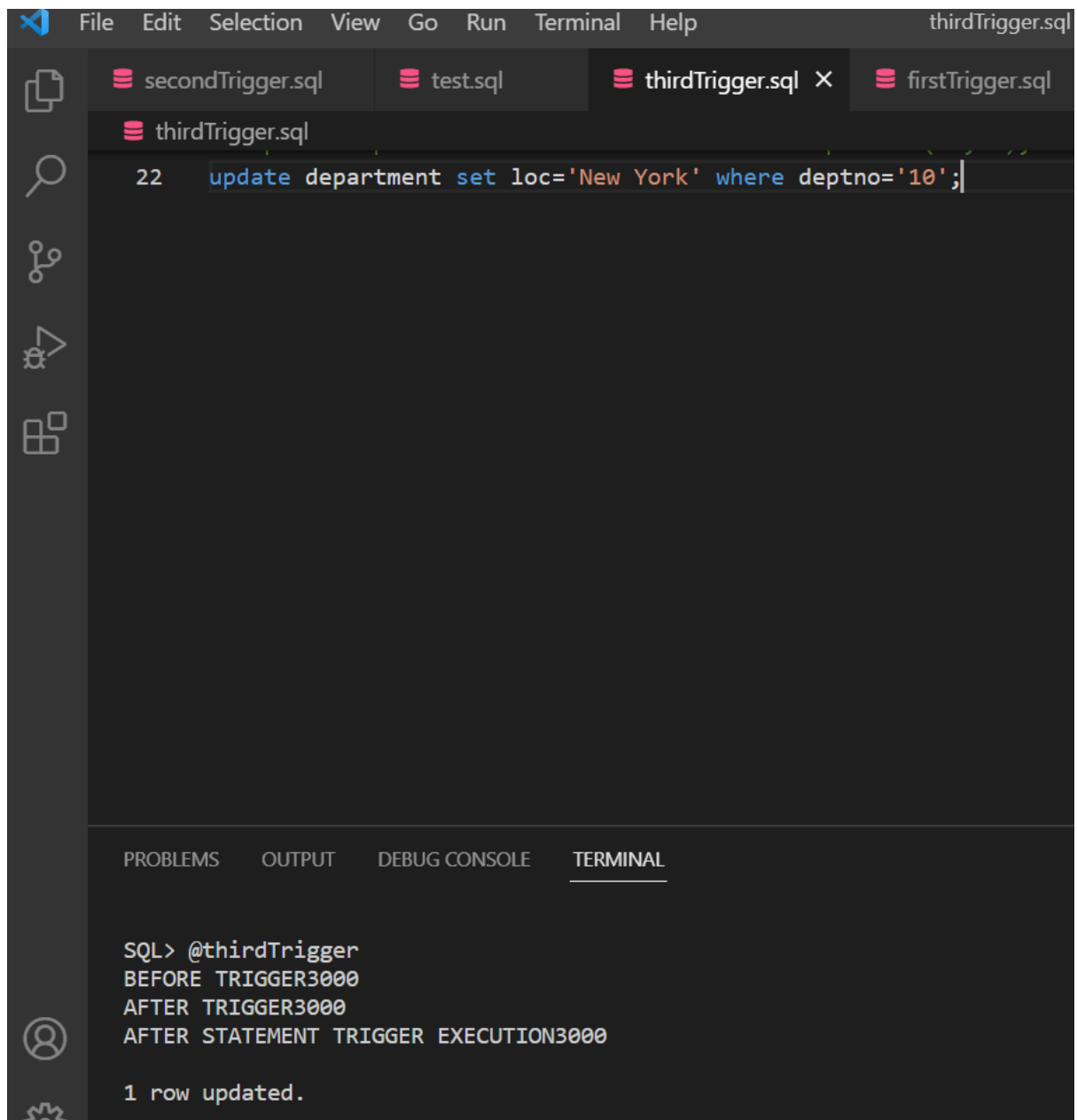
After updation my department table will look like-



The screenshot shows a SQL IDE interface with a dark theme. The terminal text shows the execution of the query 'select * from department;'. The output is a table with three columns: DEPTNO, DNAME, and LOC. The data is as follows:

DEPTNO	DNAME	LOC
20	Research	Patna
30	Sales	Chicago
40	Operations	Boston
10	Accounting	Patna

If we change only one row-



The screenshot shows an IDE with a dark theme. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The title bar on the right says 'thirdTrigger.sql'. The editor has four tabs: 'secondTrigger.sql', 'test.sql', 'thirdTrigger.sql' (active), and 'firstTrigger.sql'. The active tab shows a SQL statement at line 22: `update department set loc='New York' where deptno='10';`. The bottom panel has four tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL (active). The terminal shows the following output:

```
SQL> @thirdTrigger
BEFORE TRIGGER3000
AFTER TRIGGER3000
AFTER STATEMENT TRIGGER EXECUTION3000

1 row updated.
```