II.2314 / II.2414

Advanced Databases and Big Data

Lab 1 & 2 & 3

GUO Xiaofan

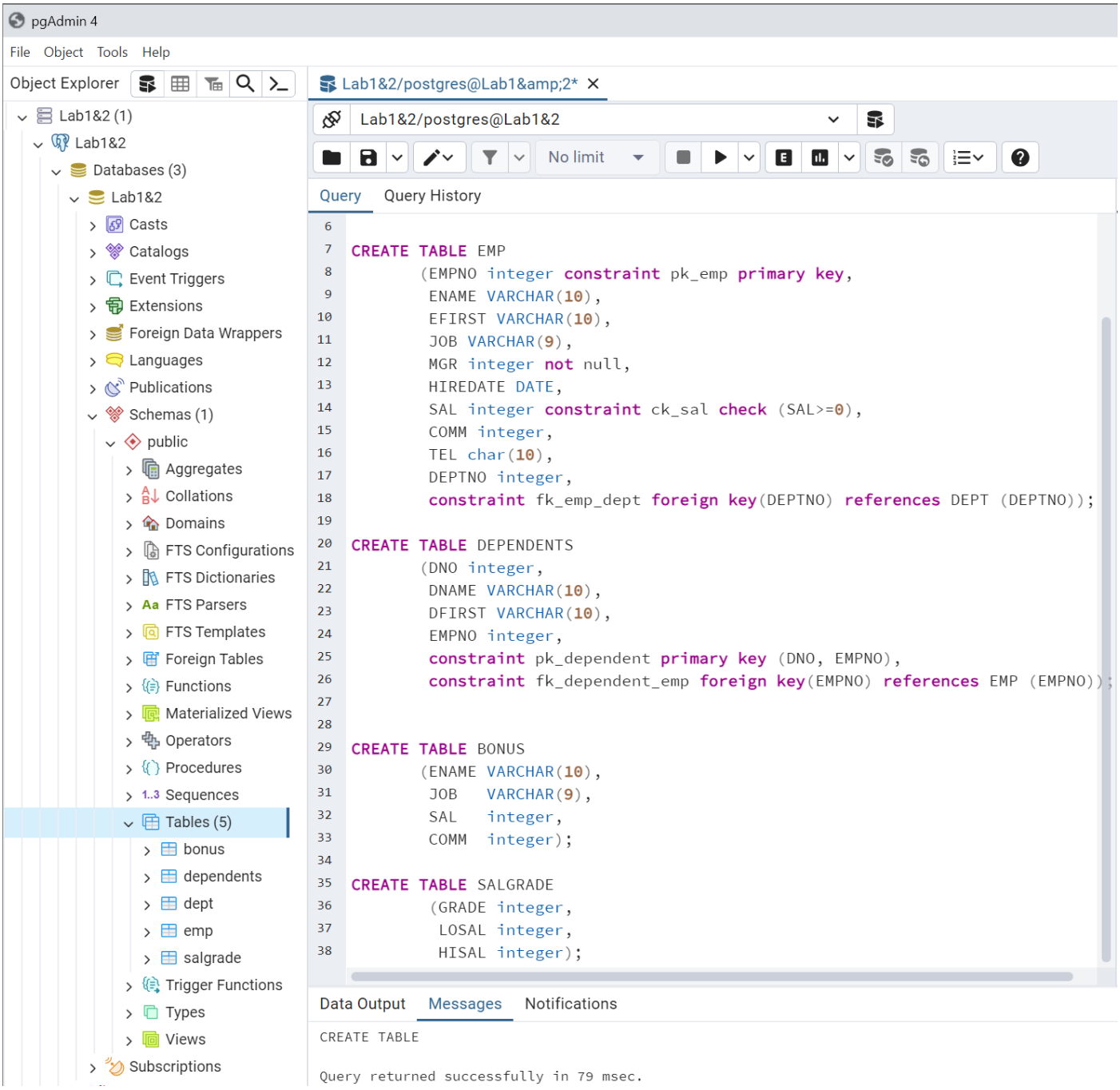
https://github.com/heyPetiteF/ISEP/tree/main/2402-2406/2-BIG%20DATA/LAB/TP3/CODE

CONTENT

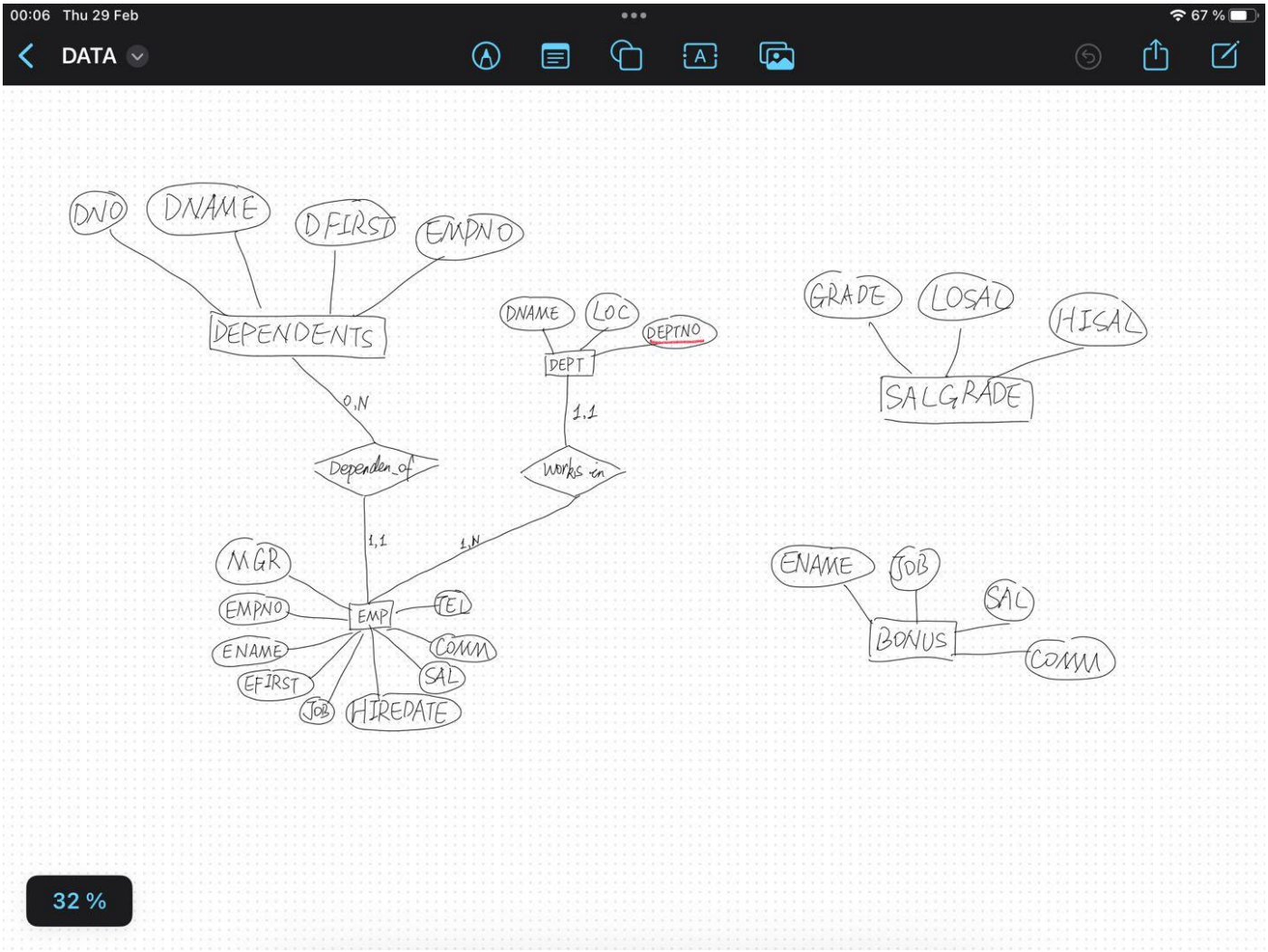
Lab 1

Exercice 1: DDL queries

1. Copy

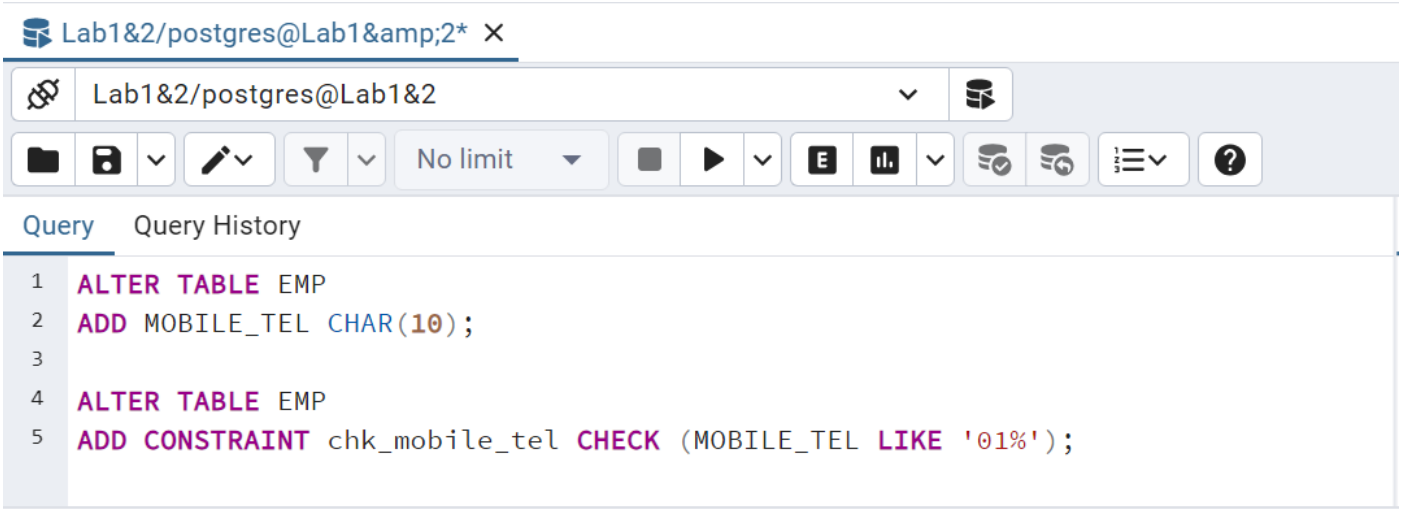


2. E/R diagram:



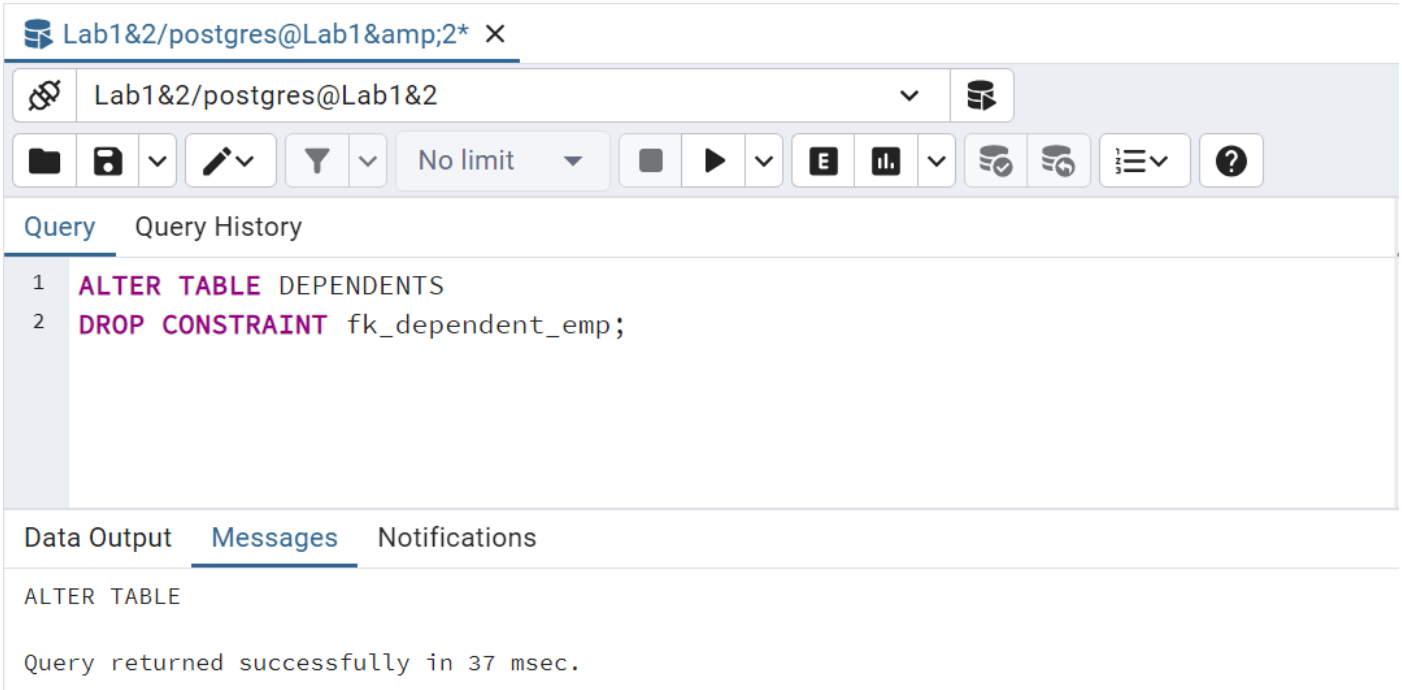
3. First integrity Constraint:

4. Second integrity Constraint

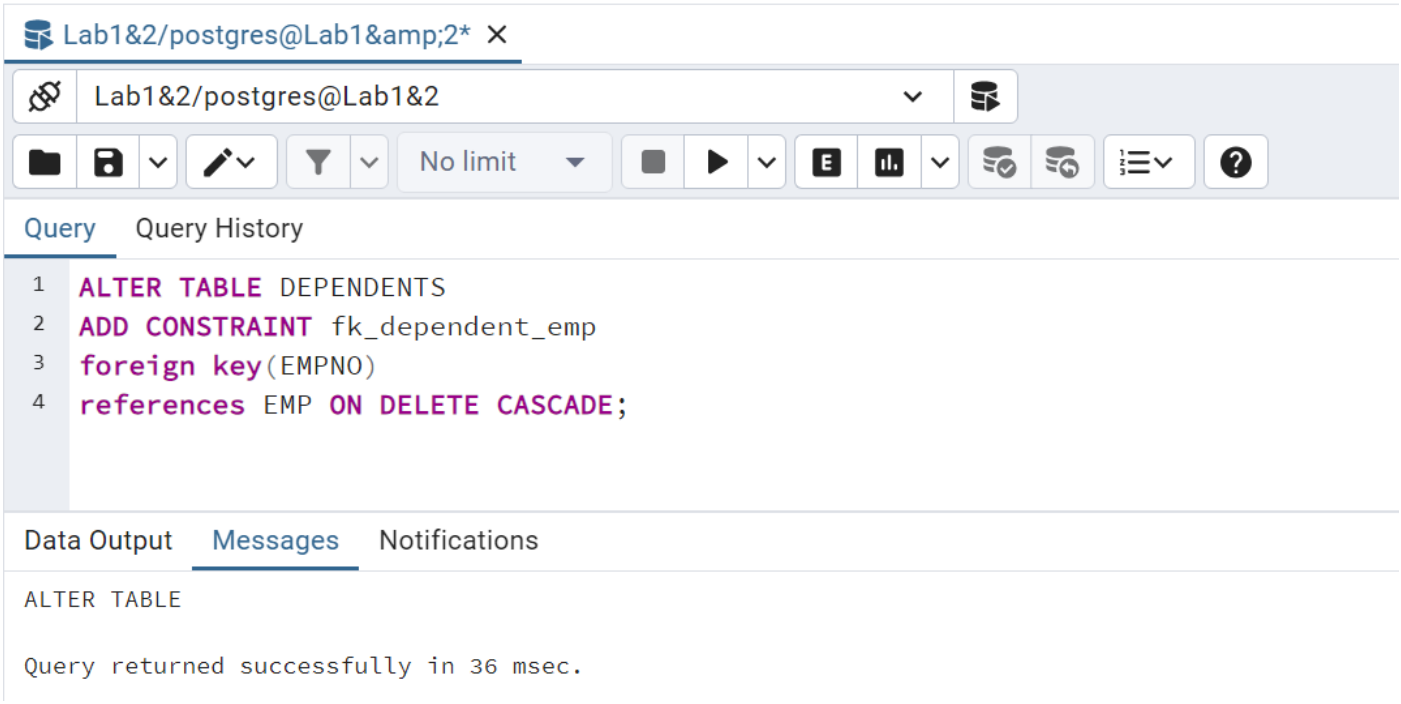


5. Third integrity Constraint on Delete

1) remove the referential constraint fk_ depenent_emp

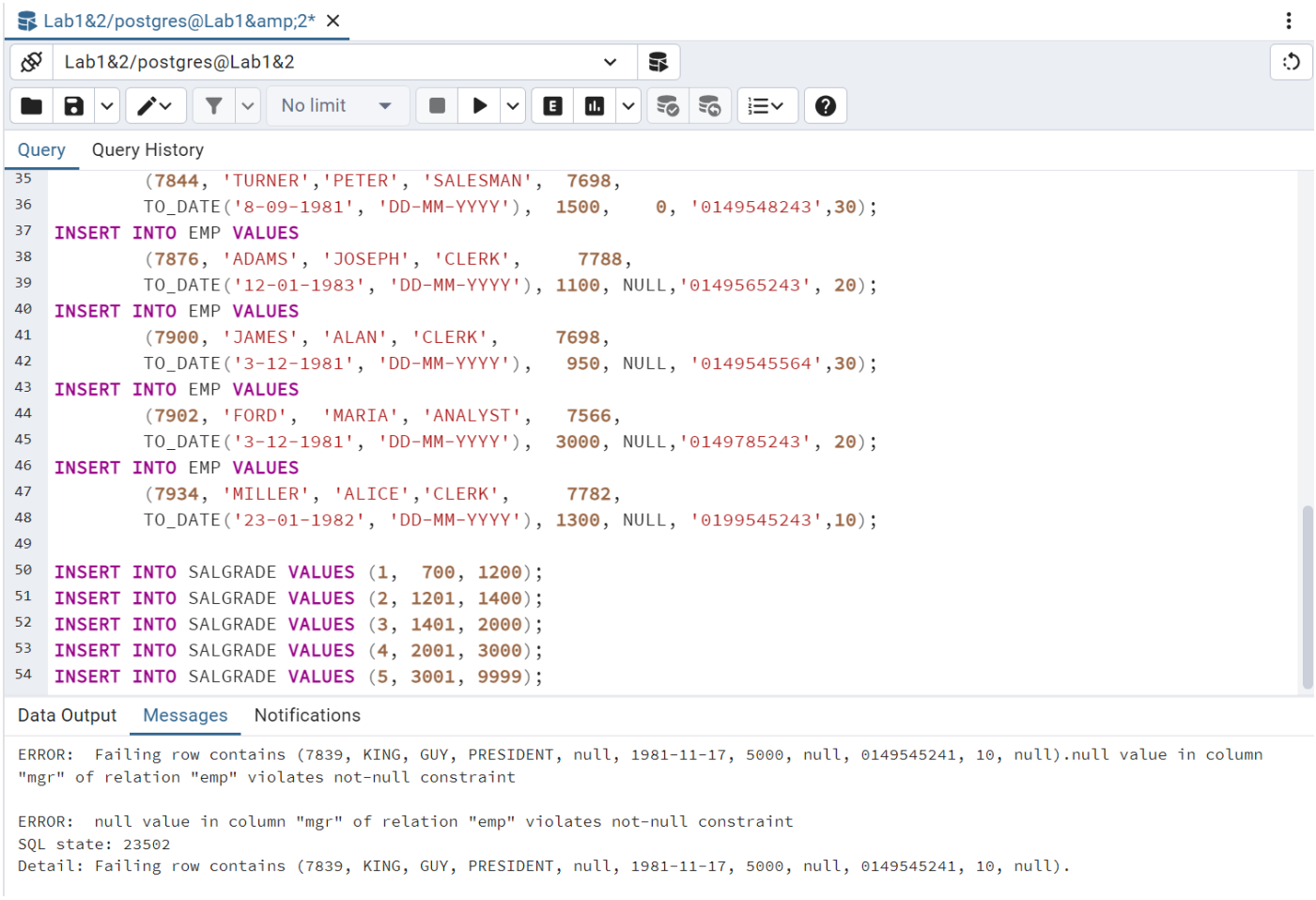


2) delete an employee and remove all his dependents on the same time.

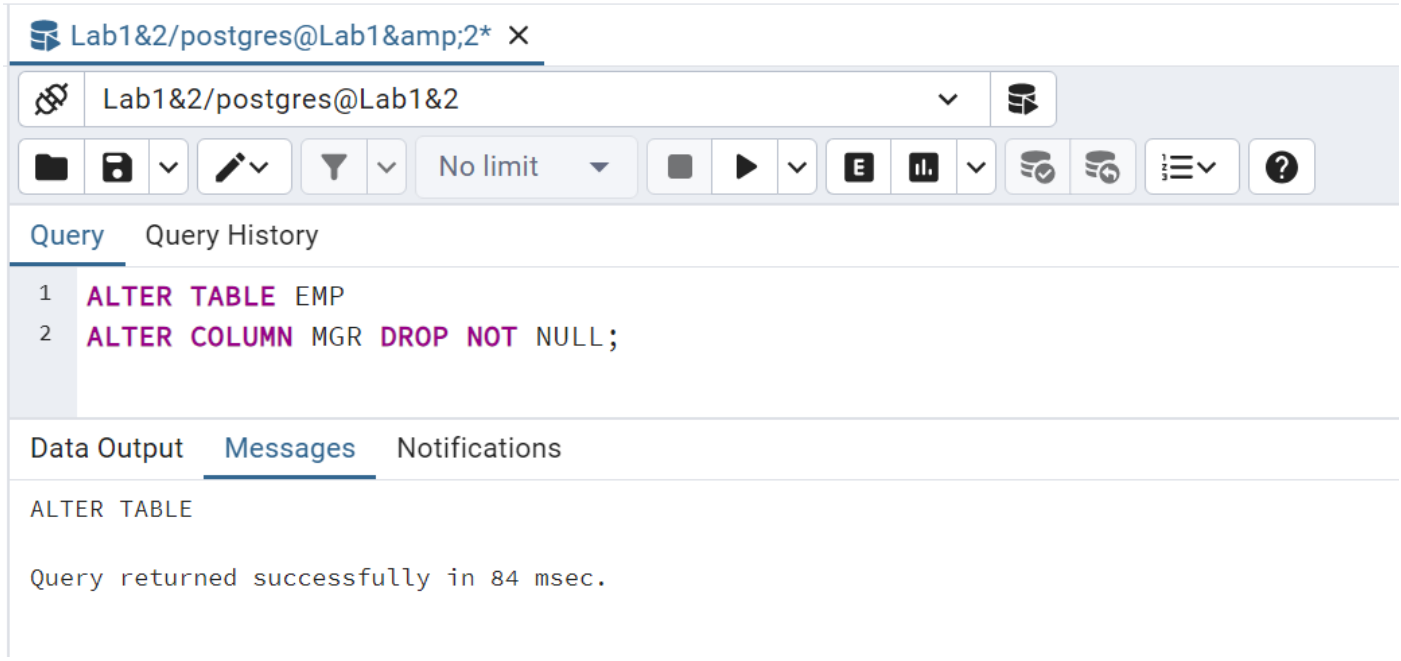


ON DELETE CASCADE is an option used when setting foreign key constraints in the database. It defines that when a record in the referenced table (parent table) is deleted, for the table (child table) containing the referenced key (foreign key) The behavior of how associated records should be processed.

6. Explain Errors for Integrity Constraint



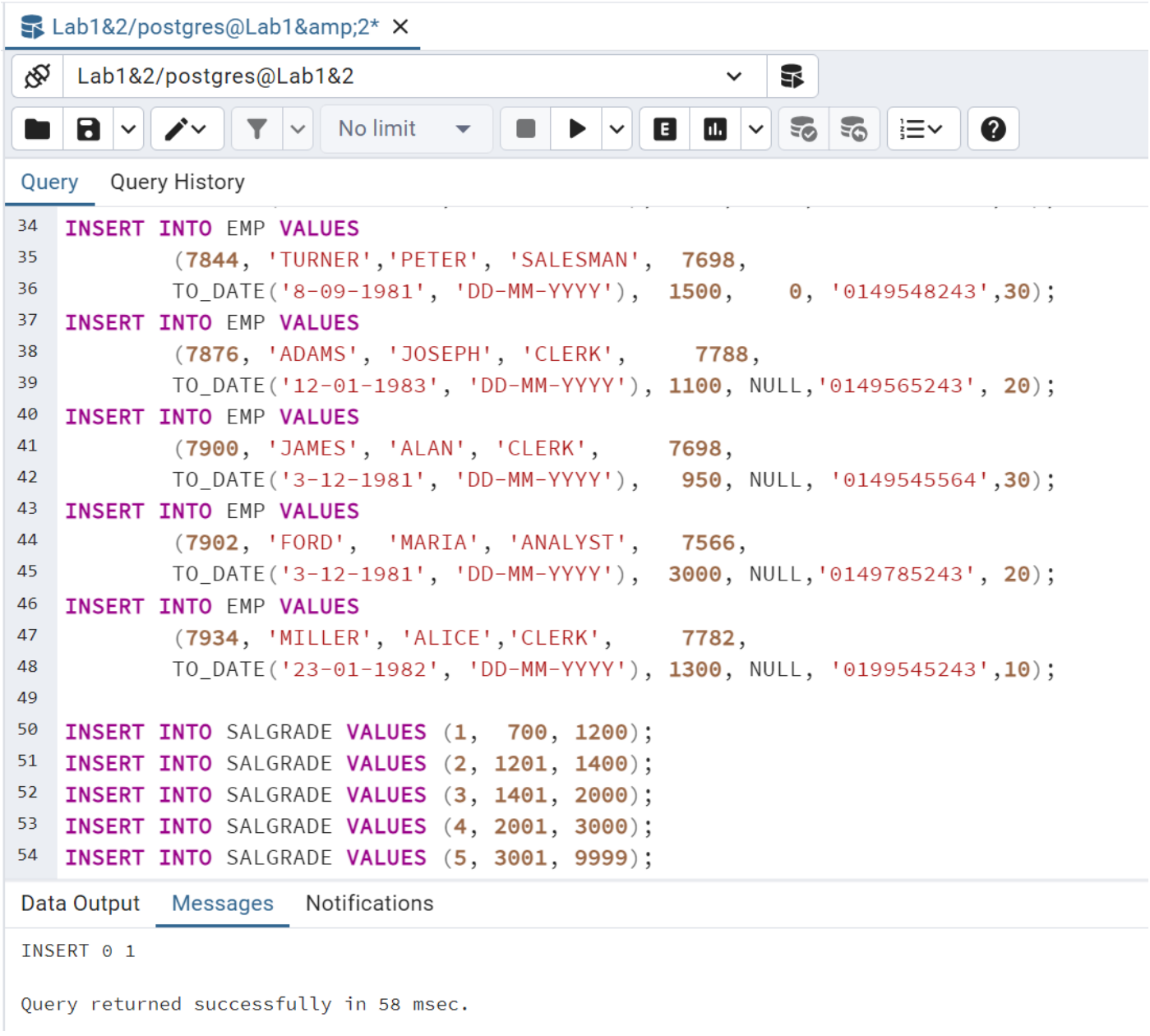
If run the code directly, we will find that due to the existence of null information, then need to do the following:



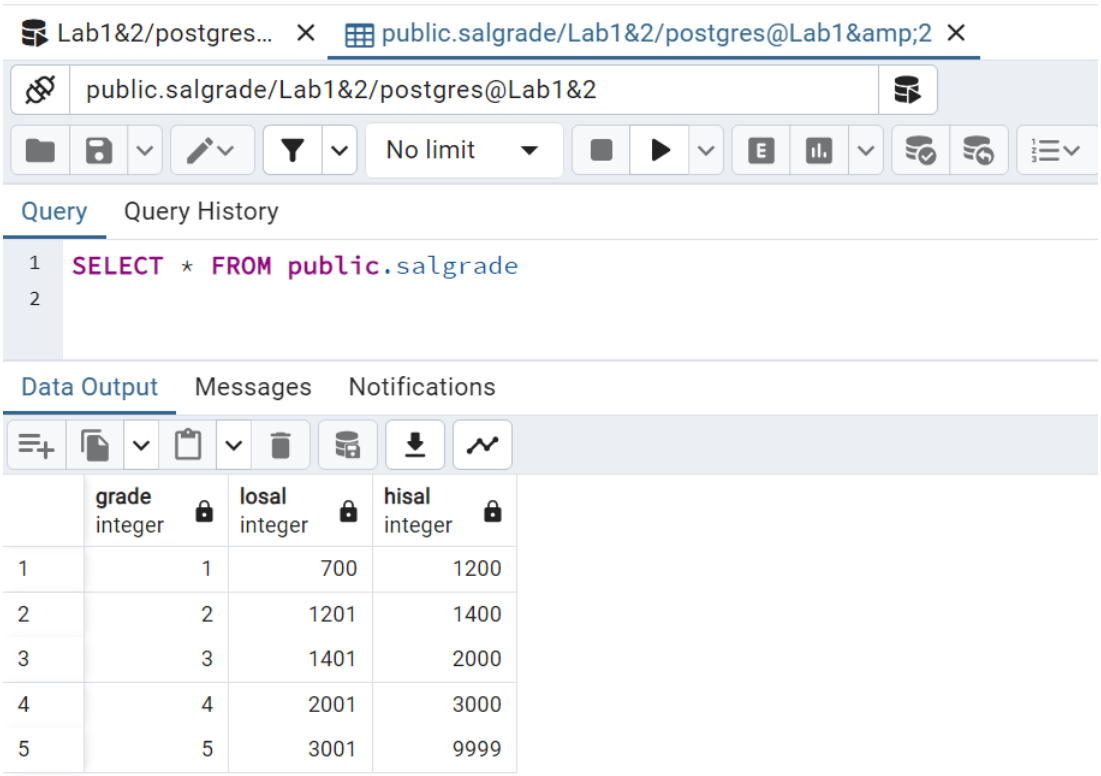
ALTER COLUMN MGR DROP NOT NULL is used to modify column attributes in database tables. This statement is used to remove the NOT NULL constraint on the

MGR column in the EMP table. In SQL, the NOT NULL constraint is used to ensure that a column cannot contain null values (NULL), that is, the column must have a value in each row.

Then run the before code again:

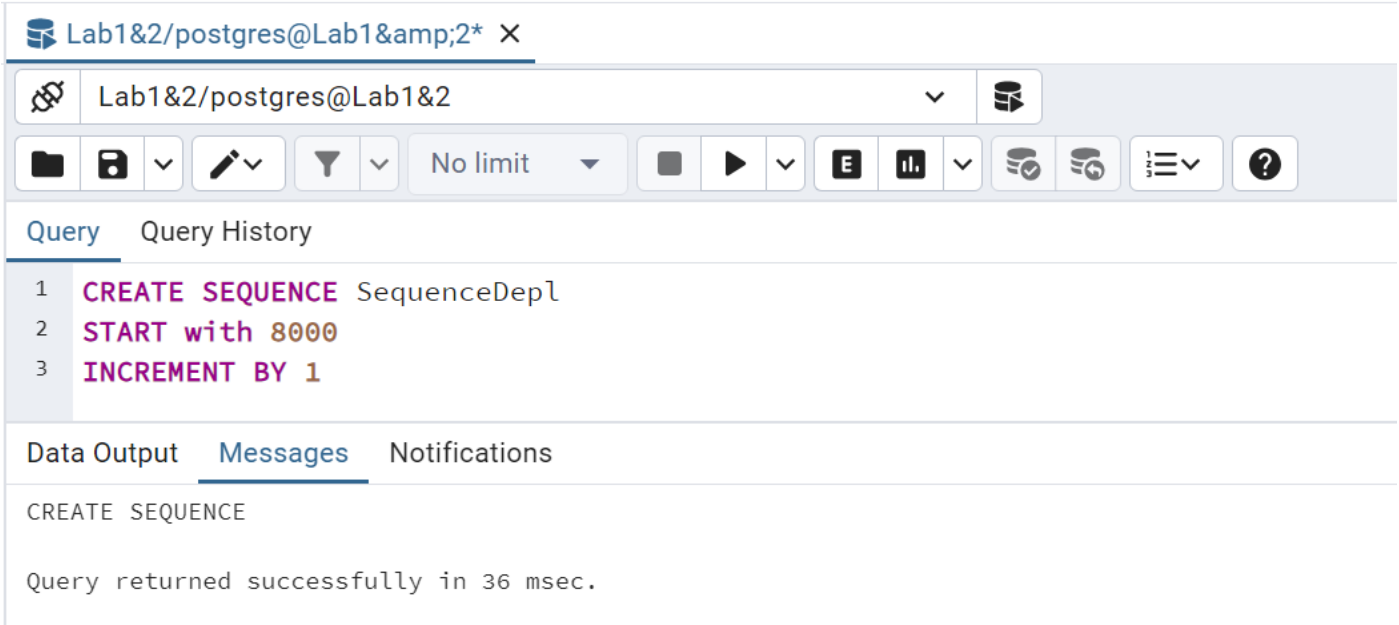


salgrade->view data check results:

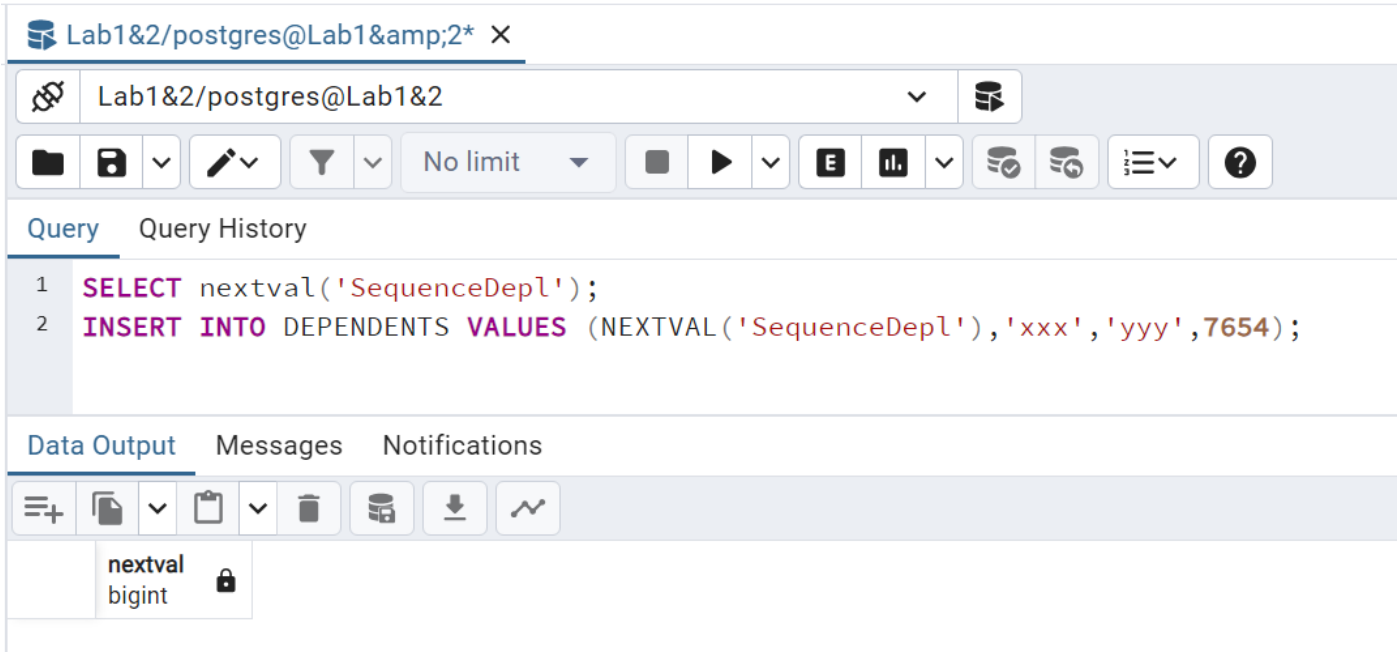


	grade integer	losal integer	hisal integer
1	1	700	1200
2	2	1201	1400
3	3	1401	2000
4	4	2001	3000
5	5	3001	9999

7. Define a sequence

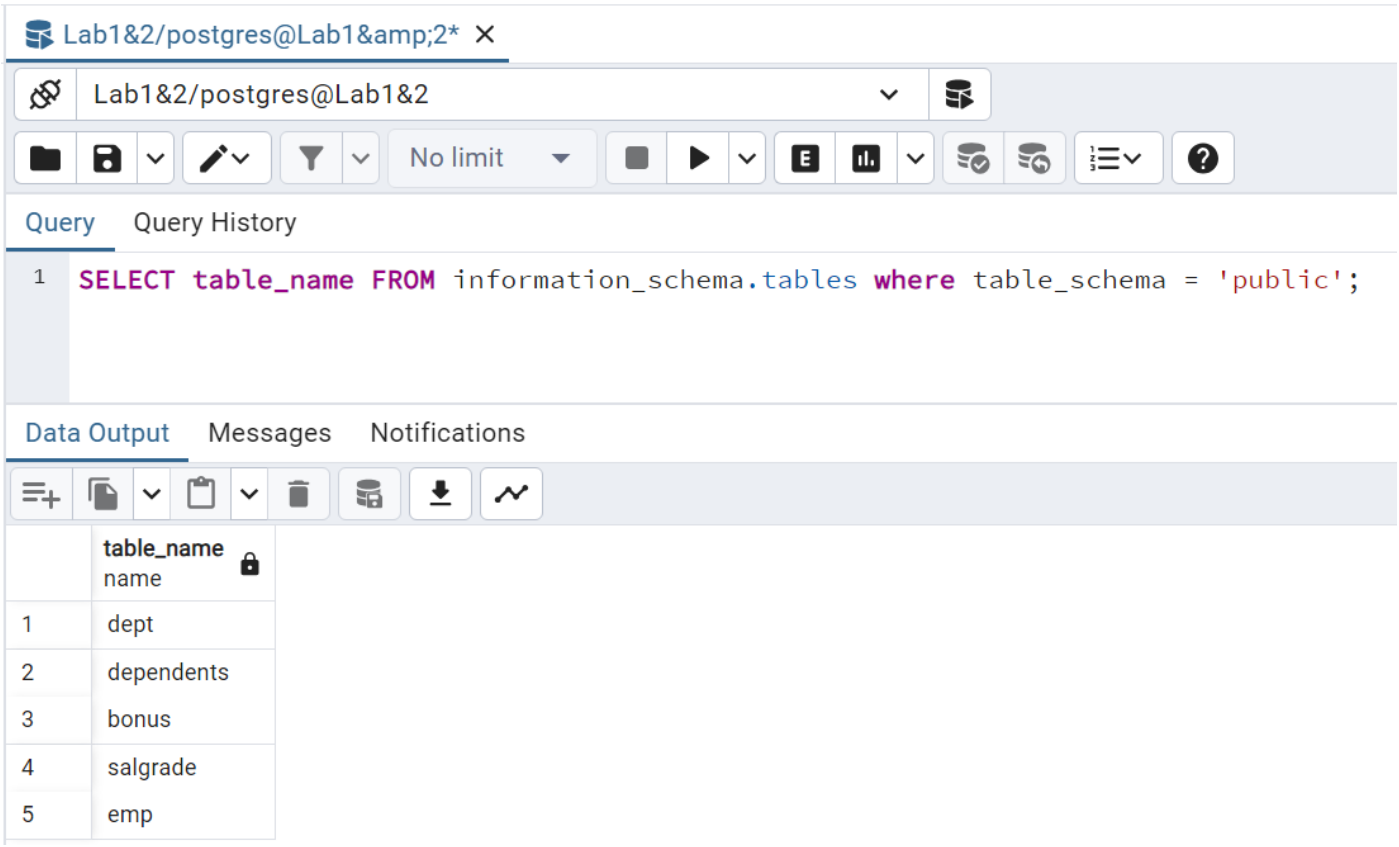


8. Use Sequence:

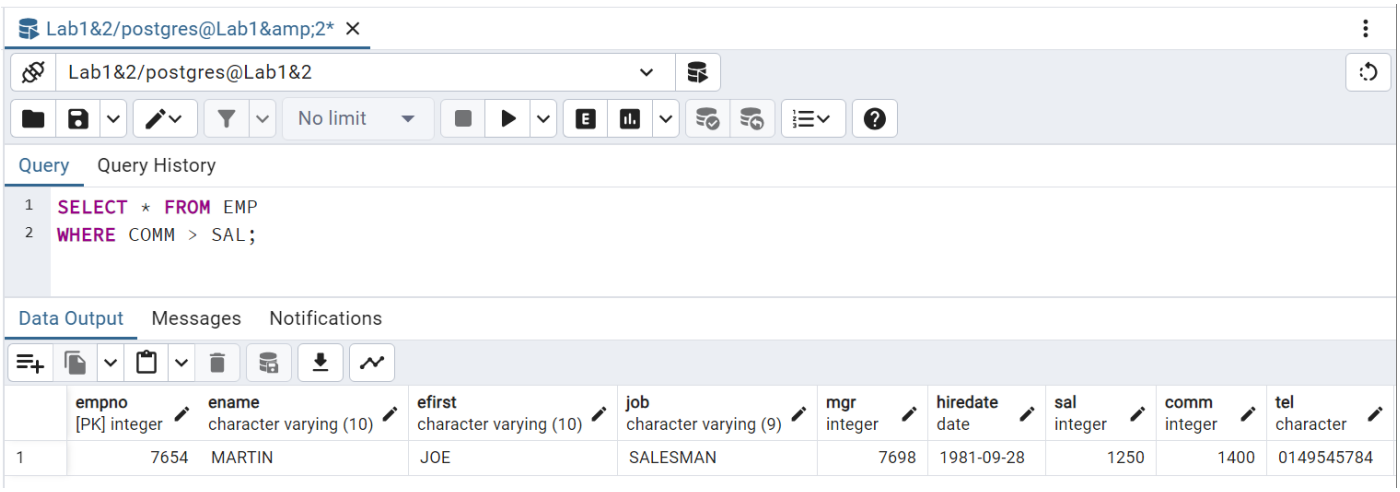


Exercise 2: DML queries Answer the following queries using SQL.

1. List the content of all tables to see the attributes names.

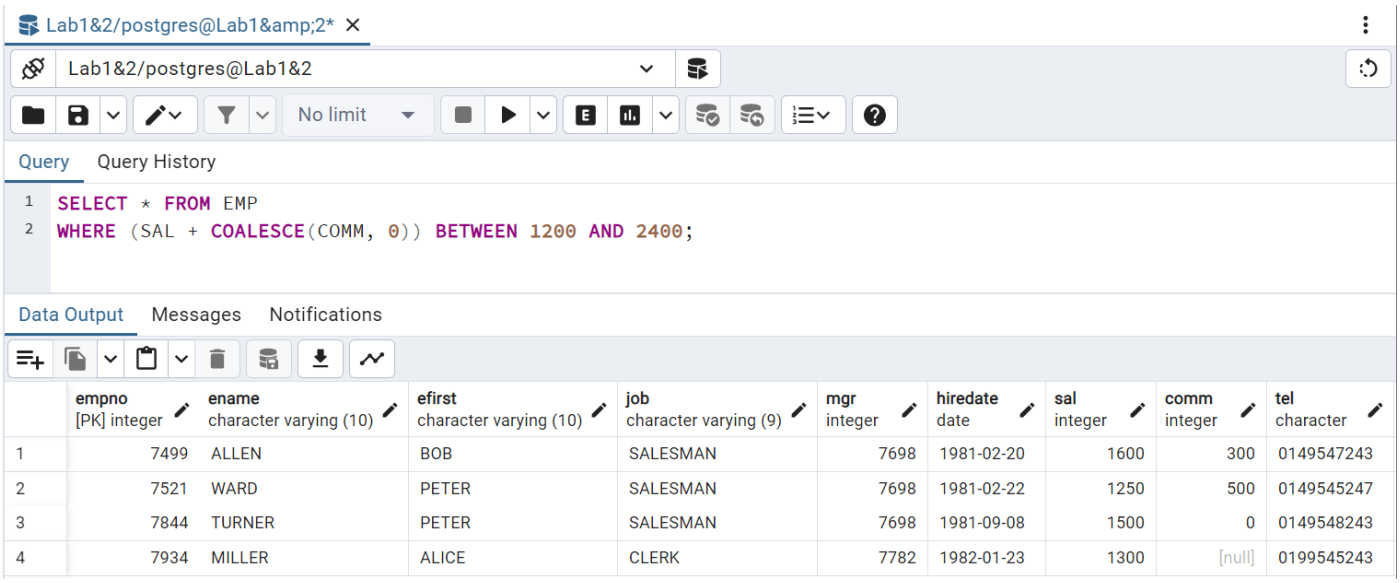


2. Select employees whose commission is higher than their salary.



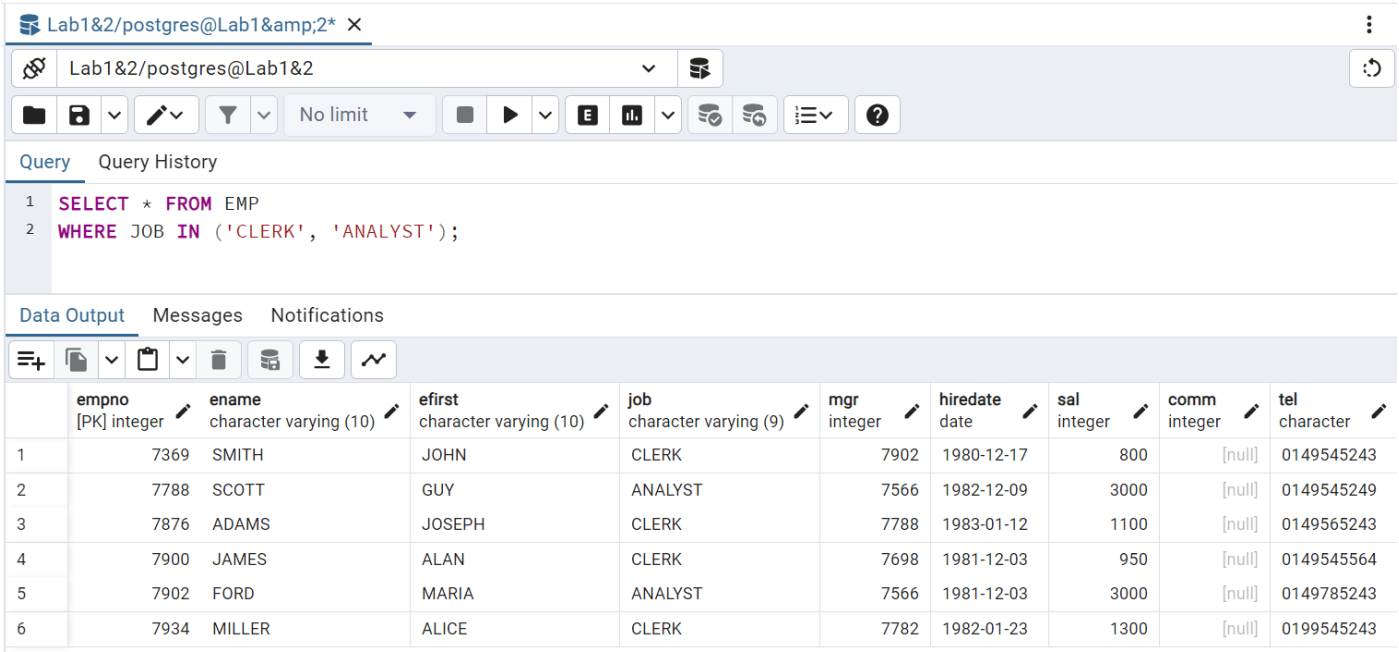
	empno [PK] integer	ename character varying (10)	efirst character varying (10)	job character varying (9)	mgr integer	hiredate date	sal integer	comm integer	tel character
1	7654	MARTIN	JOE	SALESMAN	7698	1981-09-28	1250	1400	0149545784

3. Select employees earning between 1200 and 2400 (earning is sal + commision)



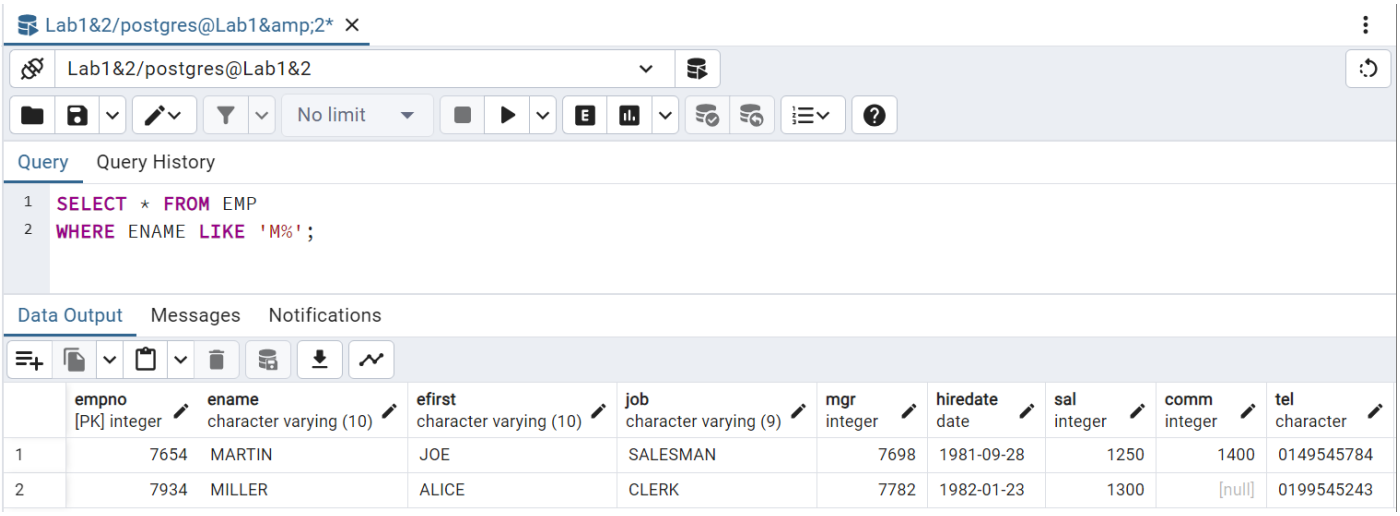
	empno [PK] integer	ename character varying (10)	efirst character varying (10)	job character varying (9)	mgr integer	hiredate date	sal integer	comm integer	tel character
1	7499	ALLEN	BOB	SALESMAN	7698	1981-02-20	1600	300	0149547243
2	7521	WARD	PETER	SALESMAN	7698	1981-02-22	1250	500	0149545247
3	7844	TURNER	PETER	SALESMAN	7698	1981-09-08	1500	0	0149548243
4	7934	MILLER	ALICE	CLERK	7782	1982-01-23	1300	[null]	0199545243

4. Select employees who are CLERK or ANALYST



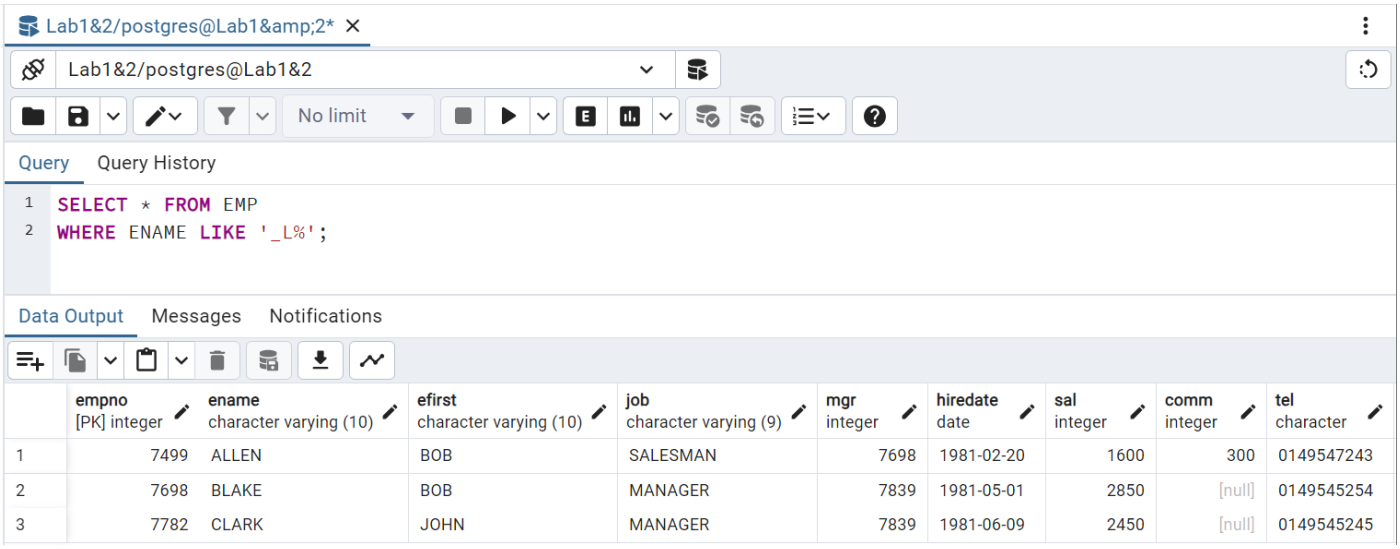
	empno [PK] integer	ename character varying (10)	efirst character varying (10)	job character varying (9)	mgr integer	hiredate date	sal integer	comm integer	tel character
1	7369	SMITH	JOHN	CLERK	7902	1980-12-17	800	[null]	0149545243
2	7788	SCOTT	GUY	ANALYST	7566	1982-12-09	3000	[null]	0149545249
3	7876	ADAMS	JOSEPH	CLERK	7788	1983-01-12	1100	[null]	0149565243
4	7900	JAMES	ALAN	CLERK	7698	1981-12-03	950	[null]	0149545564
5	7902	FORD	MARIA	ANALYST	7566	1981-12-03	3000	[null]	0149785243
6	7934	MILLER	ALICE	CLERK	7782	1982-01-23	1300	[null]	0199545243

5. Select employees whose name begins by M.



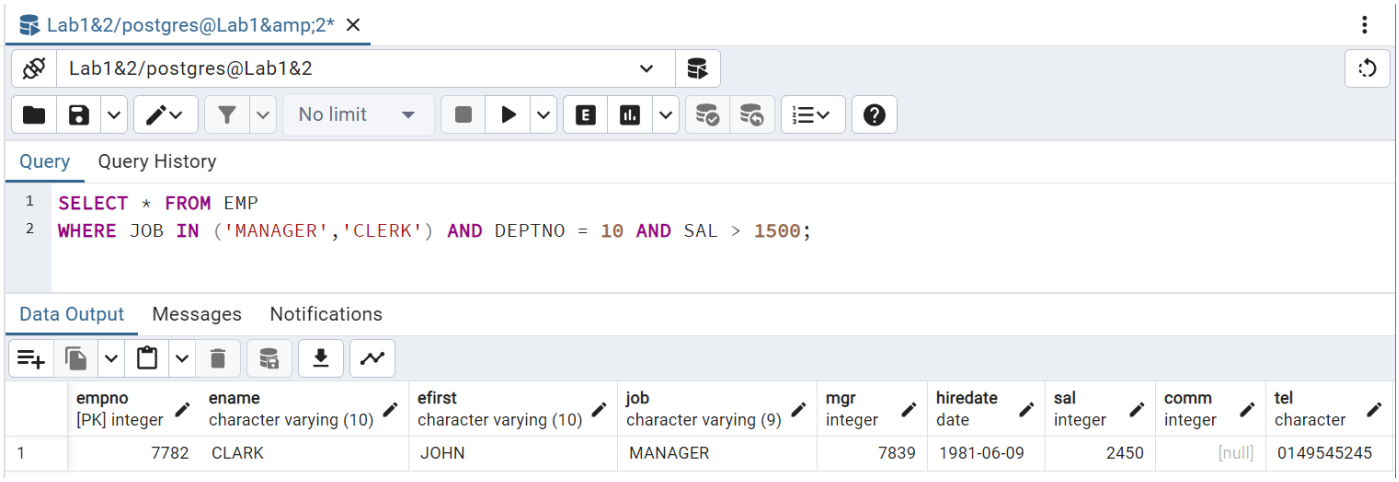
	empno [PK] integer	ename character varying (10)	efirst character varying (10)	job character varying (9)	mgr integer	hiredate date	sal integer	comm integer	tel character
1	7654	MARTIN	JOE	SALESMAN	7698	1981-09-28	1250	1400	0149545784
2	7934	MILLER	ALICE	CLERK	7782	1982-01-23	1300	[null]	0199545243

6. Select employees whose name includes a L in second position.



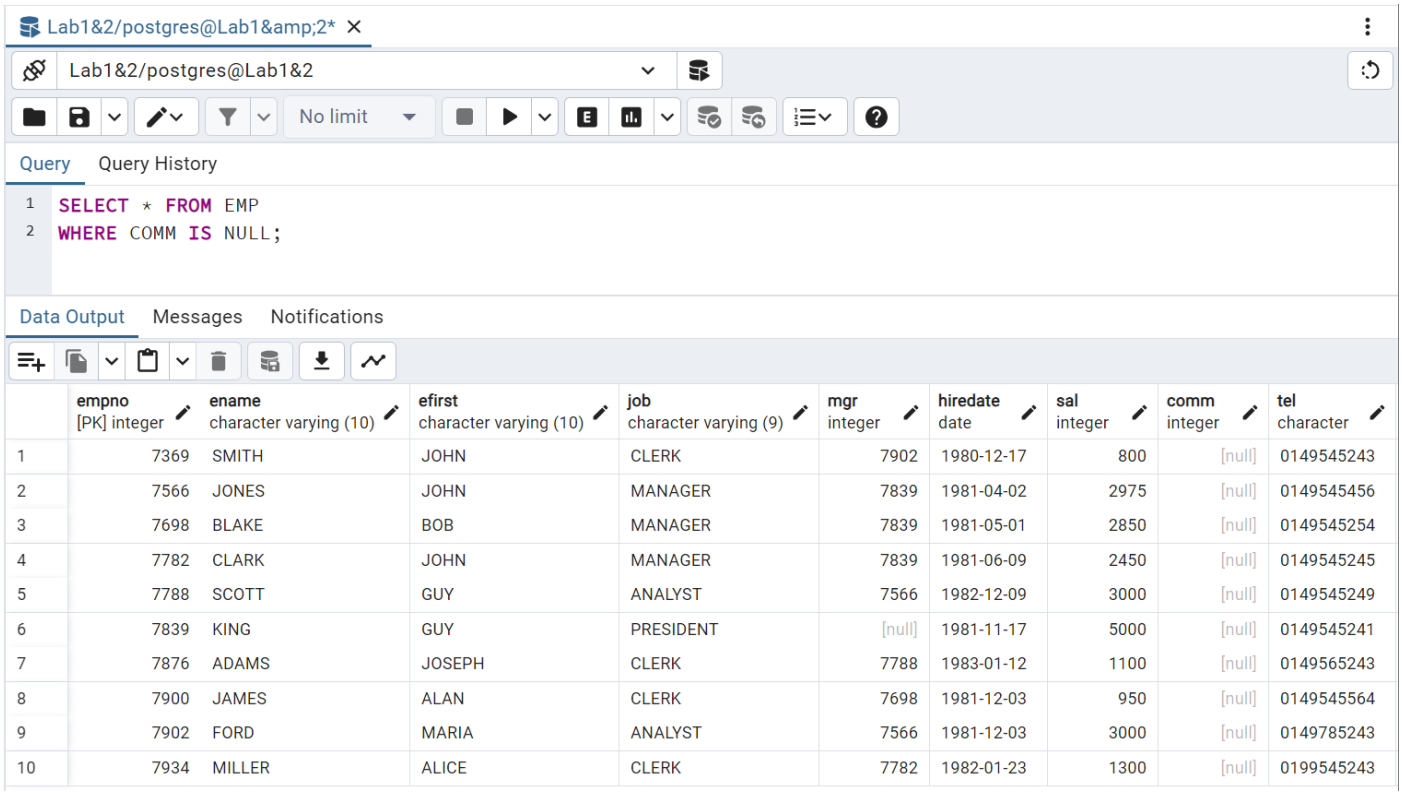
	empno [PK] integer	ename character varying (10)	efirst character varying (10)	job character varying (9)	mgr integer	hiredate date	sal integer	comm integer	tel character
1	7499	ALLEN	BOB	SALESMAN	7698	1981-02-20	1600	300	0149547243
2	7698	BLAKE	BOB	MANAGER	7839	1981-05-01	2850	[null]	0149545254
3	7782	CLARK	JOHN	MANAGER	7839	1981-06-09	2450	[null]	0149545245

7. Select employees who are MANAGER or CLERK in the department 10 and whose salary is greater than 1500.



	empno [PK] integer	ename character varying (10)	efirst character varying (10)	job character varying (9)	mgr integer	hiredate date	sal integer	comm integer	tel character
1	7782	CLARK	JOHN	MANAGER	7839	1981-06-09	2450	[null]	0149545245

8. Select employees whose commission is NULL



Lab1&2/postgres@Lab1&2*

Lab1&2/postgres@Lab1&2

No limit

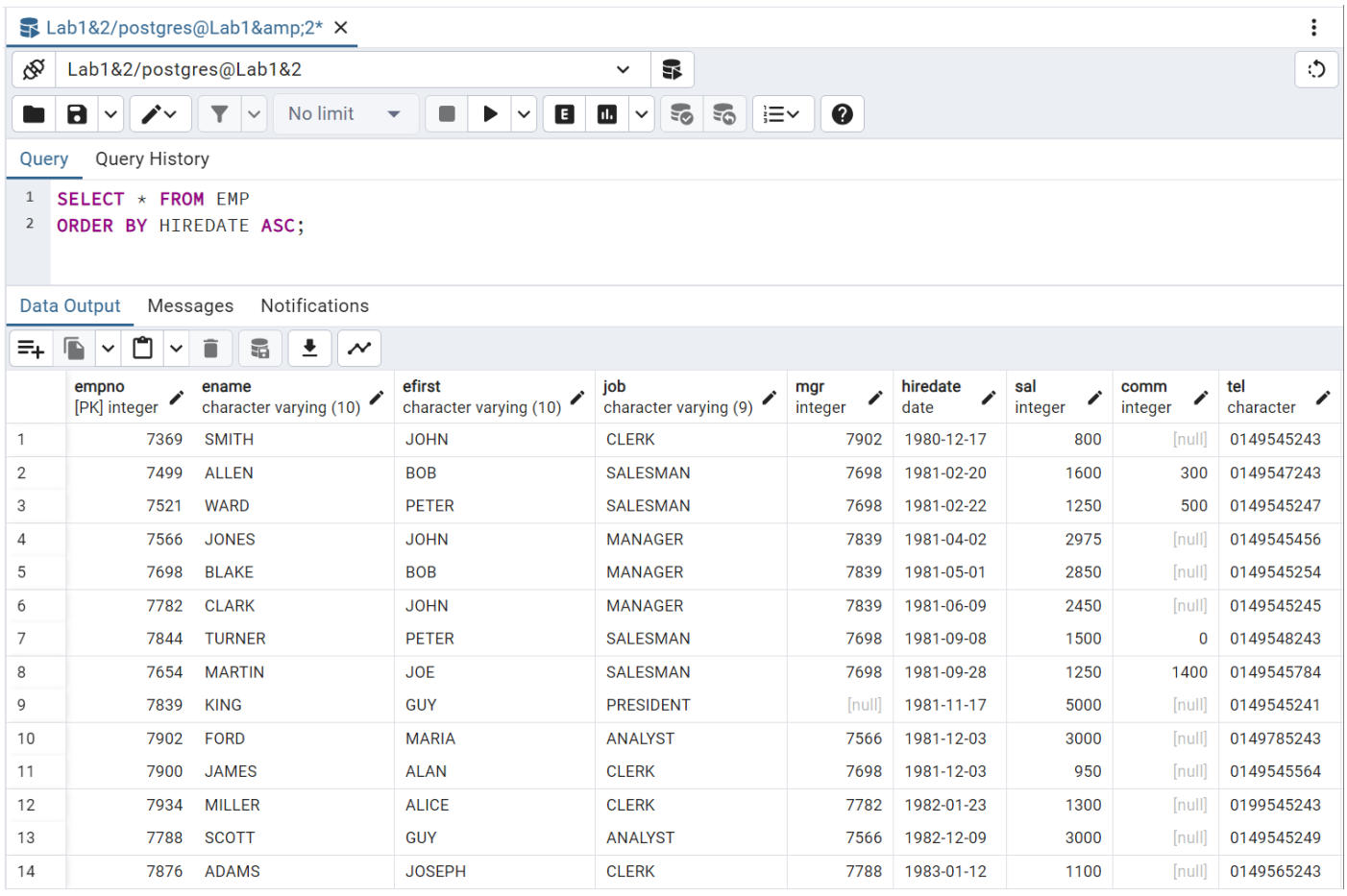
Query | Query History

```
SELECT * FROM EMP
WHERE COMM IS NULL;
```

Data Output | Messages | Notifications

	empno [PK] integer	ename character varying (10)	efirst character varying (10)	job character varying (9)	mgr integer	hiredate date	sal integer	comm integer	tel character
1	7369	SMITH	JOHN	CLERK	7902	1980-12-17	800	[null]	0149545243
2	7566	JONES	JOHN	MANAGER	7839	1981-04-02	2975	[null]	0149545456
3	7698	BLAKE	BOB	MANAGER	7839	1981-05-01	2850	[null]	0149545254
4	7782	CLARK	JOHN	MANAGER	7839	1981-06-09	2450	[null]	0149545245
5	7788	SCOTT	GUY	ANALYST	7566	1982-12-09	3000	[null]	0149545249
6	7839	KING	GUY	PRESIDENT	[null]	1981-11-17	5000	[null]	0149545241
7	7876	ADAMS	JOSEPH	CLERK	7788	1983-01-12	1100	[null]	0149565243
8	7900	JAMES	ALAN	CLERK	7698	1981-12-03	950	[null]	0149545564
9	7902	FORD	MARIA	ANALYST	7566	1981-12-03	3000	[null]	0149785243
10	7934	MILLER	ALICE	CLERK	7782	1982-01-23	1300	[null]	0199545243

9. Select employees by ascending order (by hiredate)



Lab1&2/postgres@Lab1&2*

Lab1&2/postgres@Lab1&2

No limit

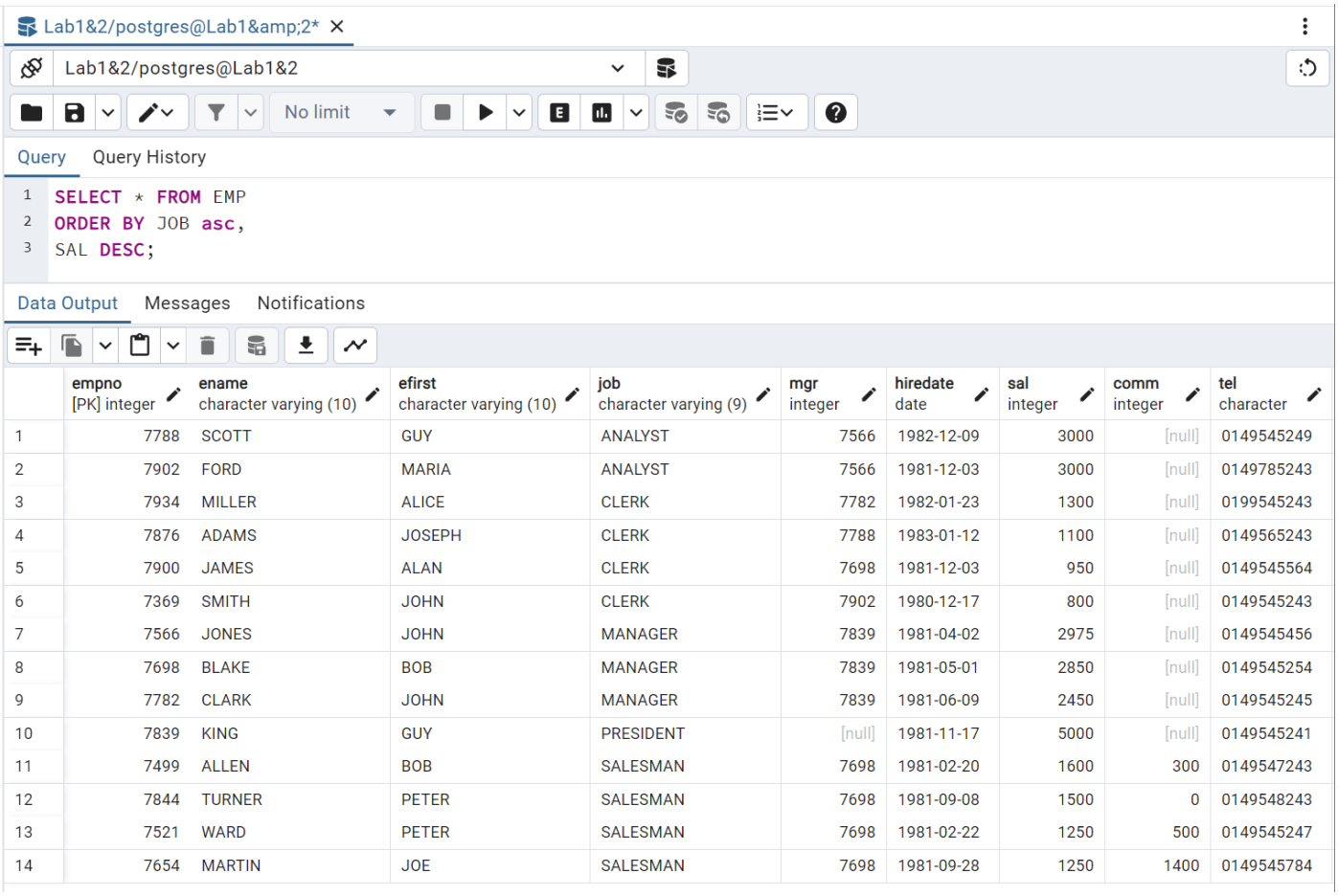
Query | Query History

```
SELECT * FROM EMP
ORDER BY HIREDATE ASC;
```

Data Output | Messages | Notifications

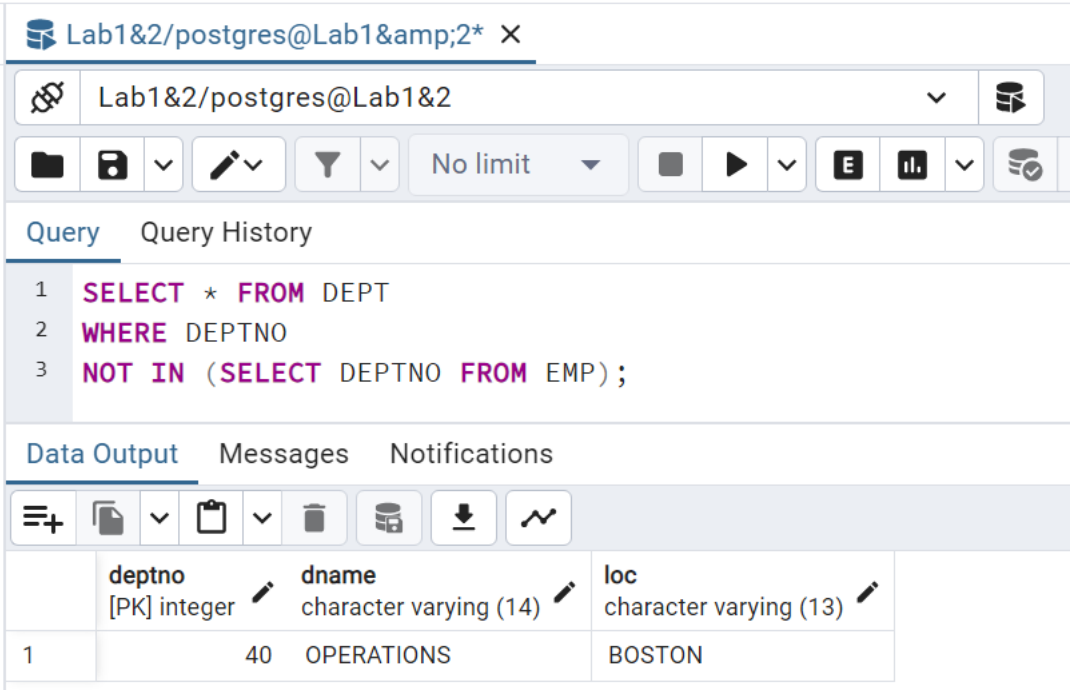
	empno [PK] integer	ename character varying (10)	efirst character varying (10)	job character varying (9)	mgr integer	hiredate date	sal integer	comm integer	tel character
1	7369	SMITH	JOHN	CLERK	7902	1980-12-17	800	[null]	0149545243
2	7499	ALLEN	BOB	SALESMAN	7698	1981-02-20	1600	300	0149547243
3	7521	WARD	PETER	SALESMAN	7698	1981-02-22	1250	500	0149545247
4	7566	JONES	JOHN	MANAGER	7839	1981-04-02	2975	[null]	0149545456
5	7698	BLAKE	BOB	MANAGER	7839	1981-05-01	2850	[null]	0149545254
6	7782	CLARK	JOHN	MANAGER	7839	1981-06-09	2450	[null]	0149545245
7	7844	TURNER	PETER	SALESMAN	7698	1981-09-08	1500	0	0149548243
8	7654	MARTIN	JOE	SALESMAN	7698	1981-09-28	1250	1400	0149545784
9	7839	KING	GUY	PRESIDENT	[null]	1981-11-17	5000	[null]	0149545241
10	7902	FORD	MARIA	ANALYST	7566	1981-12-03	3000	[null]	0149785243
11	7900	JAMES	ALAN	CLERK	7698	1981-12-03	950	[null]	0149545564
12	7934	MILLER	ALICE	CLERK	7782	1982-01-23	1300	[null]	0199545243
13	7788	SCOTT	GUY	ANALYST	7566	1982-12-09	3000	[null]	0149545249
14	7876	ADAMS	JOSEPH	CLERK	7788	1983-01-12	1100	[null]	0149565243

10.Select employees ordered by job, and for each job, by decreasing salary



	empno [PK] integer	ename character varying (10)	efirst character varying (10)	job character varying (9)	mgr integer	hiredate date	sal integer	comm integer	tel character
1	7788	SCOTT	GUY	ANALYST	7566	1982-12-09	3000	[null]	0149545249
2	7902	FORD	MARIA	ANALYST	7566	1981-12-03	3000	[null]	0149785243
3	7934	MILLER	ALICE	CLERK	7782	1982-01-23	1300	[null]	0199545243
4	7876	ADAMS	JOSEPH	CLERK	7788	1983-01-12	1100	[null]	0149565243
5	7900	JAMES	ALAN	CLERK	7698	1981-12-03	950	[null]	0149545564
6	7369	SMITH	JOHN	CLERK	7902	1980-12-17	800	[null]	0149545243
7	7566	JONES	JOHN	MANAGER	7839	1981-04-02	2975	[null]	0149545456
8	7698	BLAKE	BOB	MANAGER	7839	1981-05-01	2850	[null]	0149545254
9	7782	CLARK	JOHN	MANAGER	7839	1981-06-09	2450	[null]	0149545245
10	7839	KING	GUY	PRESIDENT	[null]	1981-11-17	5000	[null]	0149545241
11	7499	ALLEN	BOB	SALESMAN	7698	1981-02-20	1600	300	0149547243
12	7844	TURNER	PETER	SALESMAN	7698	1981-09-08	1500	0	0149548243
13	7521	WARD	PETER	SALESMAN	7698	1981-02-22	1250	500	0149545247
14	7654	MARTIN	JOE	SALESMAN	7698	1981-09-28	1250	1400	0149545784

11.Select departments without employees



	deptno [PK] integer	dname character varying (14)	loc character varying (13)
1	40	OPERATIONS	BOSTON

12. List employees indicating for each the name of his/her manager

Lab1&2/postgres@Lab1&2*

Lab1&2/postgres@Lab1&2

No limit

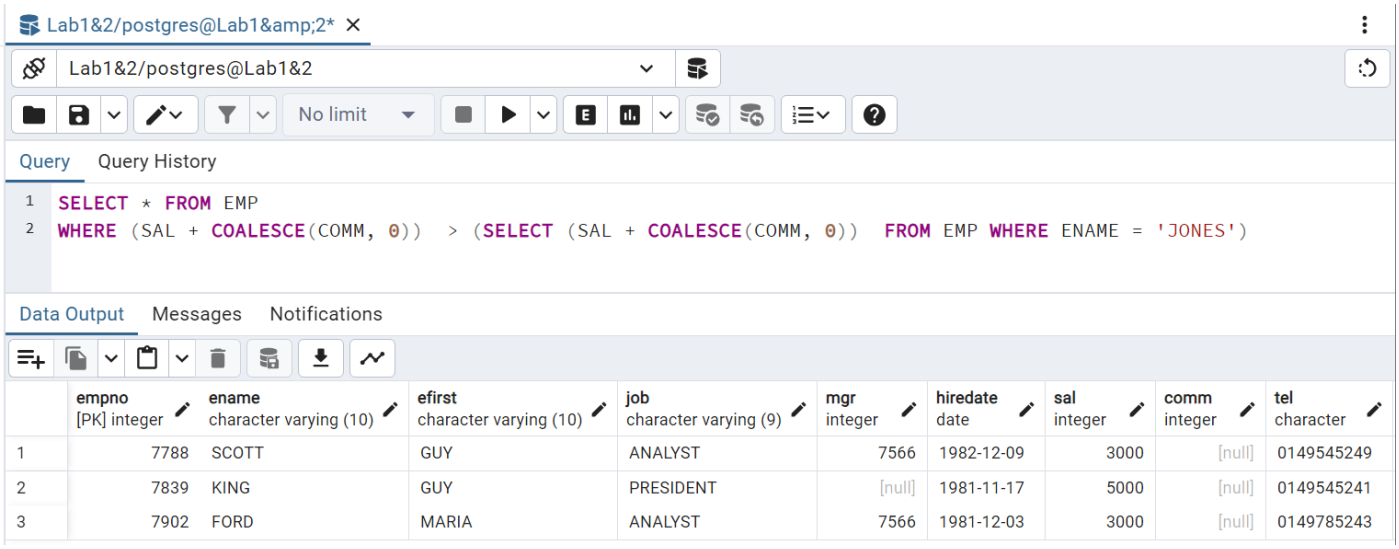
Query | Query History

```
1 SELECT E.EMPNO, E.ENAME, M.ENAME AS MANAGER_NAME
2 FROM EMP E
3 LEFT JOIN EMP M ON E.MGR = M.EMPNO;
```

Data Output | Messages | Notifications

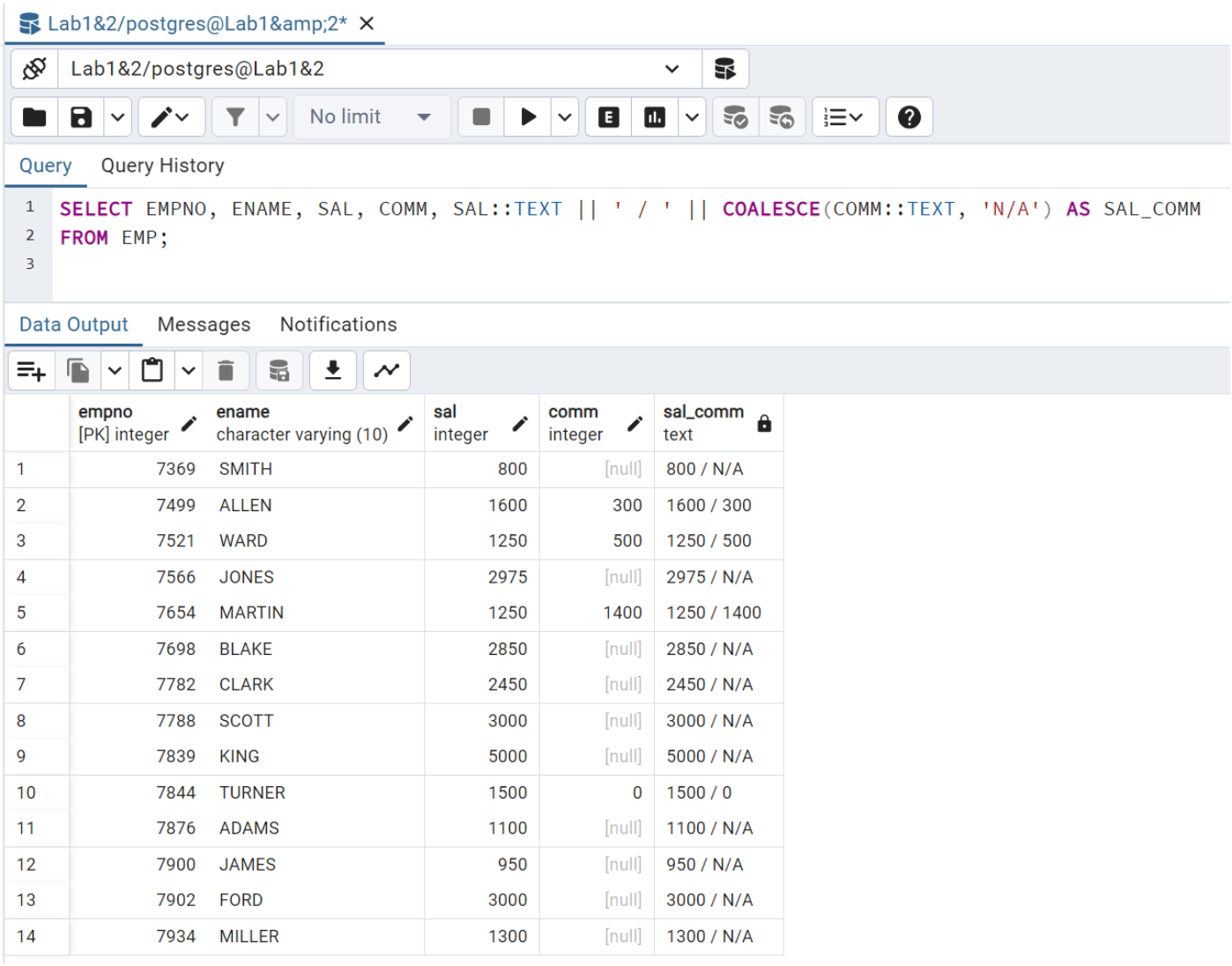
	empno [PK] integer	ename character varying (10)	manager_name character varying (10)
1	7369	SMITH	FORD
2	7499	ALLEN	BLAKE
3	7521	WARD	BLAKE
4	7566	JONES	KING
5	7654	MARTIN	BLAKE
6	7698	BLAKE	KING
7	7782	CLARK	KING
8	7788	SCOTT	JONES
9	7839	KING	[null]
10	7844	TURNER	BLAKE
11	7876	ADAMS	SCOTT
12	7900	JAMES	BLAKE
13	7902	FORD	JONES
14	7934	MILLER	CLARK

13. List employees earning more than JONES



	empno [PK] integer	ename character varying (10)	efirst character varying (10)	job character varying (9)	mgr integer	hiredate date	sal integer	comm integer	tel character
1	7788	SCOTT	GUY	ANALYST	7566	1982-12-09	3000	[null]	0149545249
2	7839	KING	GUY	PRESIDENT	[null]	1981-11-17	5000	[null]	0149545241
3	7902	FORD	MARIA	ANALYST	7566	1981-12-03	3000	[null]	0149785243

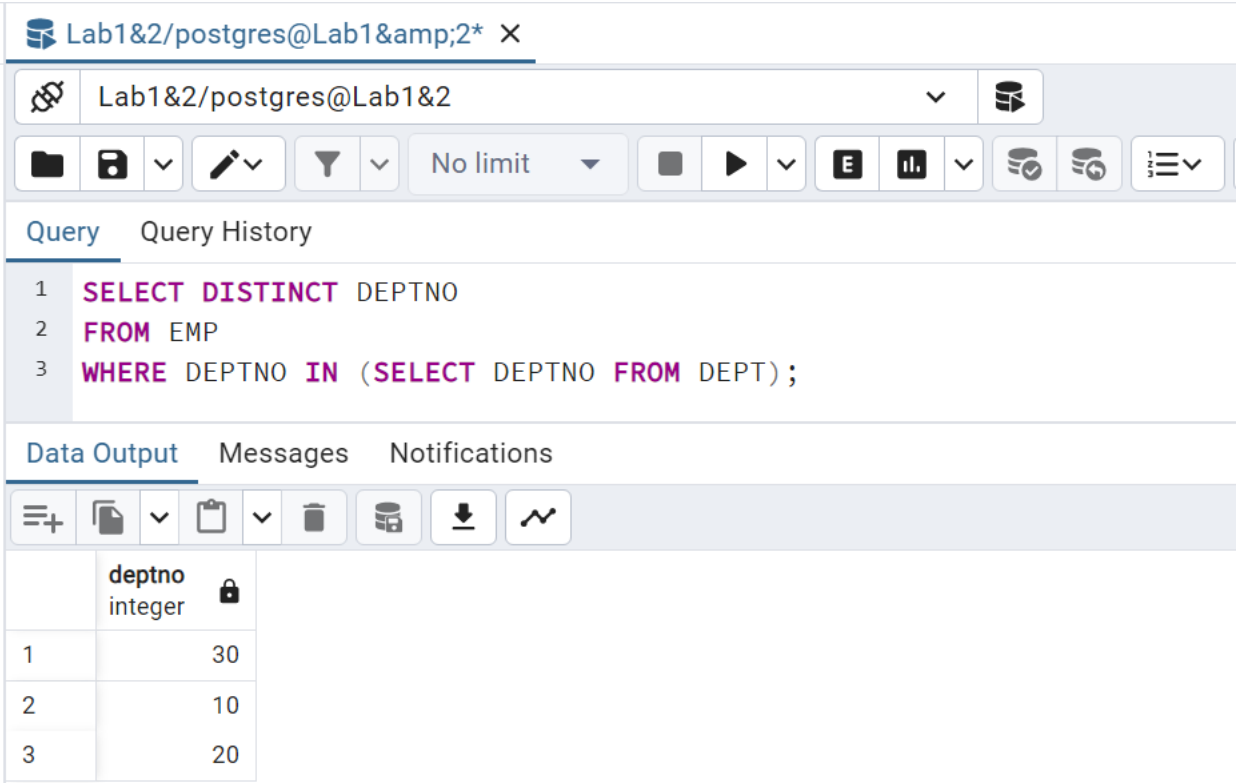
14. List employees displaying in the same column salary and commission



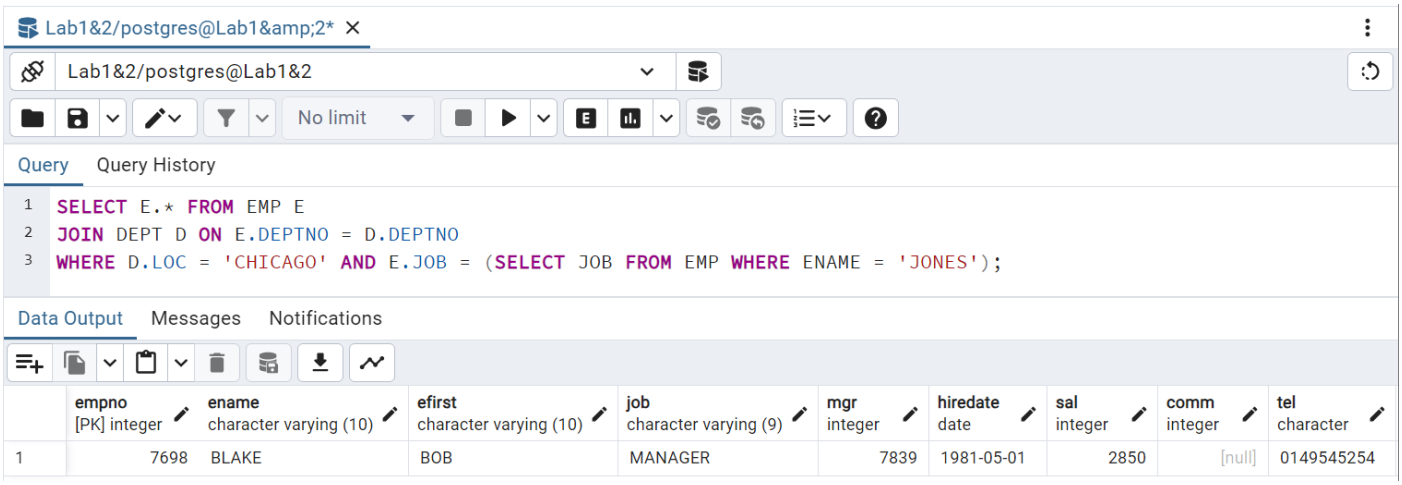
	empno [PK] integer	ename character varying (10)	sal integer	comm integer	sal_comm text
1	7369	SMITH	800	[null]	800 / N/A
2	7499	ALLEN	1600	300	1600 / 300
3	7521	WARD	1250	500	1250 / 500
4	7566	JONES	2975	[null]	2975 / N/A
5	7654	MARTIN	1250	1400	1250 / 1400
6	7698	BLAKE	2850	[null]	2850 / N/A
7	7782	CLARK	2450	[null]	2450 / N/A
8	7788	SCOTT	3000	[null]	3000 / N/A
9	7839	KING	5000	[null]	5000 / N/A
10	7844	TURNER	1500	0	1500 / 0
11	7876	ADAMS	1100	[null]	1100 / N/A
12	7900	JAMES	950	[null]	950 / N/A
13	7902	FORD	3000	[null]	3000 / N/A
14	7934	MILLER	1300	[null]	1300 / N/A

Note: The COALESCE function requires all parameters to be of the same type

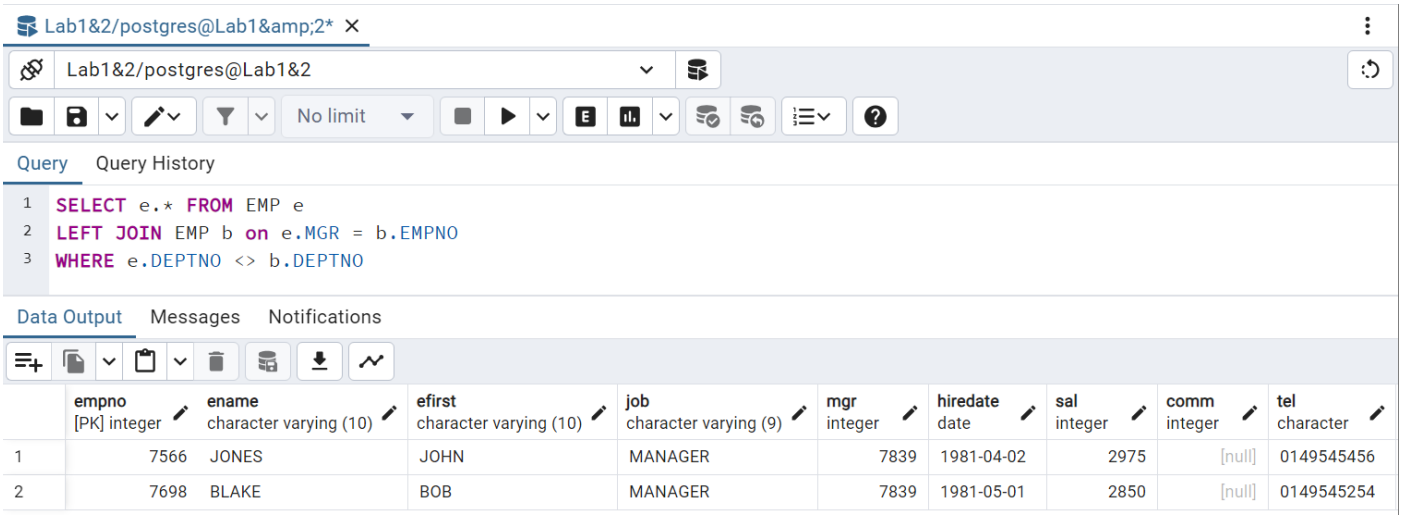
15. List department numbers which are both in table EMP and in table DEPT



16. List employees working in CHICAGO and having the same job than JONES



17. List employees who don't work in the same department than their manager



Lab1&2/postgres@Lab1&2*

Lab1&2/postgres@Lab1&2

No limit

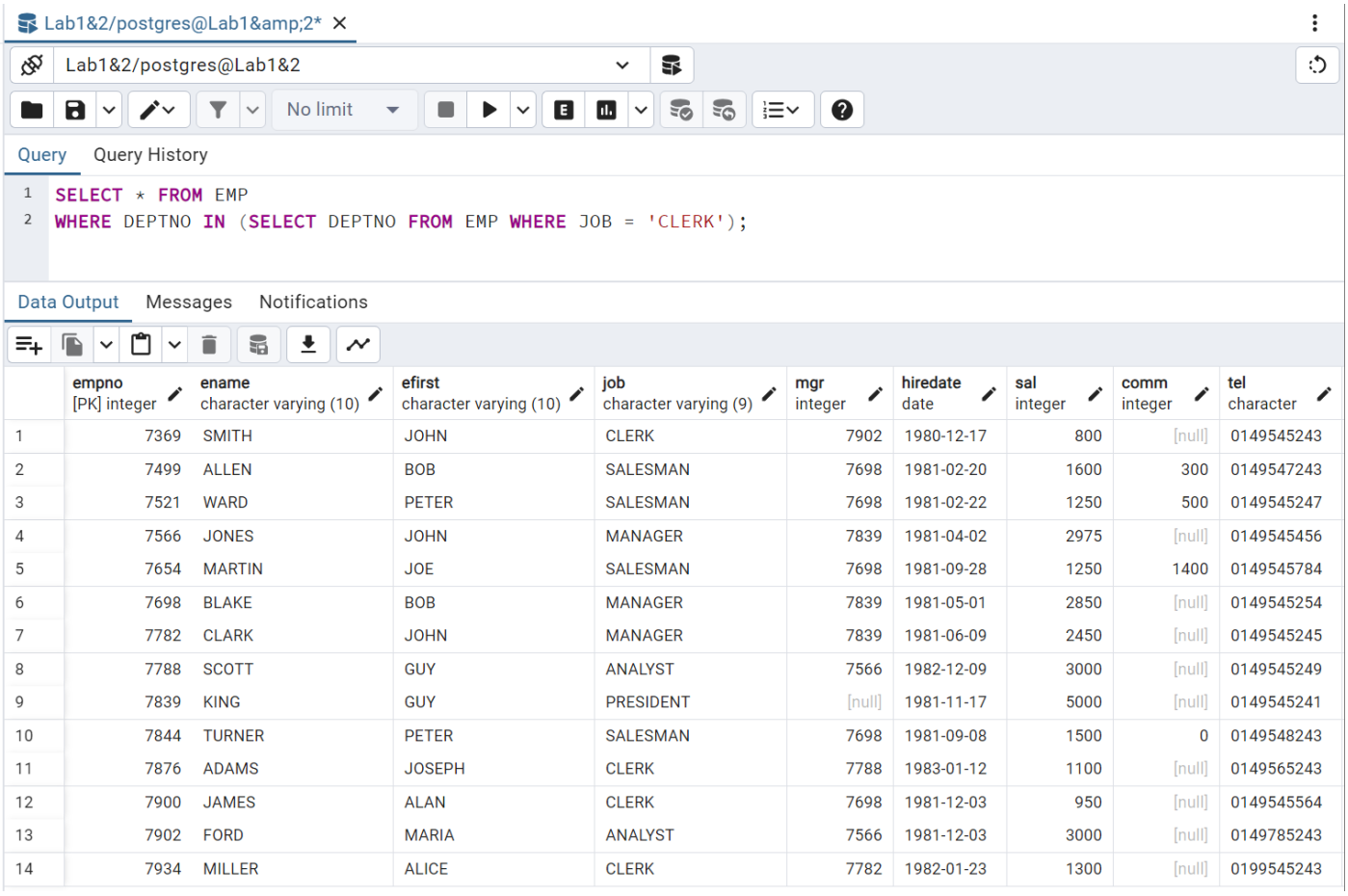
Query | Query History

```
SELECT e.* FROM EMP e
LEFT JOIN EMP b on e.MGR = b.EMPNO
WHERE e.DEPTNO <> b.DEPTNO
```

Data Output | Messages | Notifications

	empno [PK] integer	ename character varying (10)	efirst character varying (10)	job character varying (9)	mgr integer	hiredate date	sal integer	comm integer	tel character
1	7566	JONES	JOHN	MANAGER	7839	1981-04-02	2975	[null]	0149545456
2	7698	BLAKE	BOB	MANAGER	7839	1981-05-01	2850	[null]	0149545254

18. List employees working in a department having at least one CLERK



Lab1&2/postgres@Lab1&2*

Lab1&2/postgres@Lab1&2

No limit

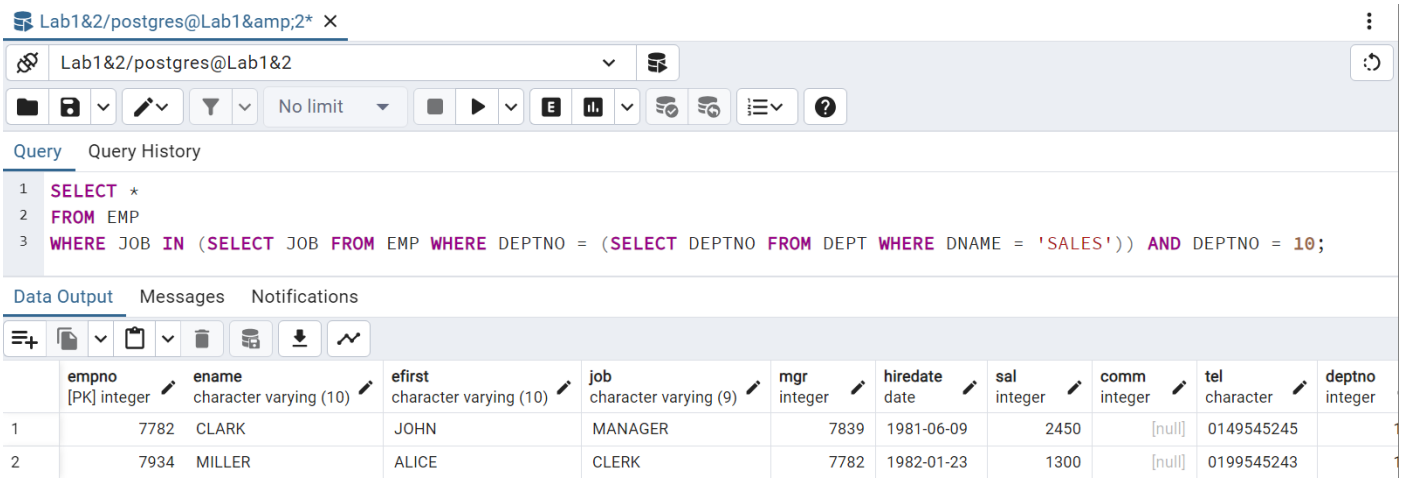
Query | Query History

```
SELECT * FROM EMP
WHERE DEPTNO IN (SELECT DEPTNO FROM EMP WHERE JOB = 'CLERK');
```

Data Output | Messages | Notifications

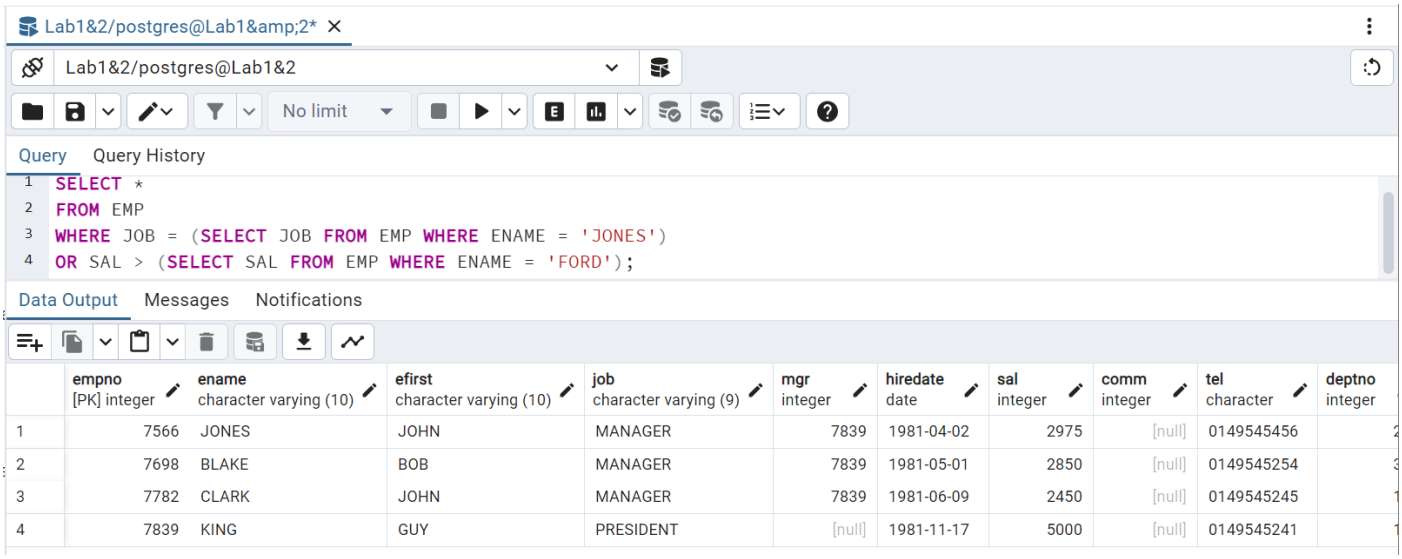
	empno [PK] integer	ename character varying (10)	efirst character varying (10)	job character varying (9)	mgr integer	hiredate date	sal integer	comm integer	tel character
1	7369	SMITH	JOHN	CLERK	7902	1980-12-17	800	[null]	0149545243
2	7499	ALLEN	BOB	SALESMAN	7698	1981-02-20	1600	300	0149547243
3	7521	WARD	PETER	SALESMAN	7698	1981-02-22	1250	500	0149545247
4	7566	JONES	JOHN	MANAGER	7839	1981-04-02	2975	[null]	0149545456
5	7654	MARTIN	JOE	SALESMAN	7698	1981-09-28	1250	1400	0149545784
6	7698	BLAKE	BOB	MANAGER	7839	1981-05-01	2850	[null]	0149545254
7	7782	CLARK	JOHN	MANAGER	7839	1981-06-09	2450	[null]	0149545245
8	7788	SCOTT	GUY	ANALYST	7566	1982-12-09	3000	[null]	0149545249
9	7839	KING	GUY	PRESIDENT	[null]	1981-11-17	5000	[null]	0149545241
10	7844	TURNER	PETER	SALESMAN	7698	1981-09-08	1500	0	0149548243
11	7876	ADAMS	JOSEPH	CLERK	7788	1983-01-12	1100	[null]	0149565243
12	7900	JAMES	ALAN	CLERK	7698	1981-12-03	950	[null]	0149545564
13	7902	FORD	MARIA	ANALYST	7566	1981-12-03	3000	[null]	0149785243
14	7934	MILLER	ALICE	CLERK	7782	1982-01-23	1300	[null]	0199545243

19. List employees of department 10 having the same job than someone from the depart SALES



	empno [PK] integer	ename character varying (10)	efirst character varying (10)	job character varying (9)	mgr integer	hiredate date	sal integer	comm integer	tel character	deptno integer
1	7782	CLARK	JOHN	MANAGER	7839	1981-06-09	2450	[null]	0149545245	
2	7934	MILLER	ALICE	CLERK	7782	1982-01-23	1300	[null]	0199545243	

20. List employees having the same job than JONES or a salary greater than FORD's salary



	empno [PK] integer	ename character varying (10)	efirst character varying (10)	job character varying (9)	mgr integer	hiredate date	sal integer	comm integer	tel character	deptno integer
1	7566	JONES	JOHN	MANAGER	7839	1981-04-02	2975	[null]	0149545456	
2	7698	BLAKE	BOB	MANAGER	7839	1981-05-01	2850	[null]	0149545254	
3	7782	CLARK	JOHN	MANAGER	7839	1981-06-09	2450	[null]	0149545245	
4	7839	KING	GUY	PRESIDENT	[null]	1981-11-17	5000	[null]	0149545241	

21. List employees having a salary greater than all employees of department 20

Lab1&2/postgres@Lab1&2* ×

Lab1&2/postgres@Lab1&2

No limit

Query | Query History

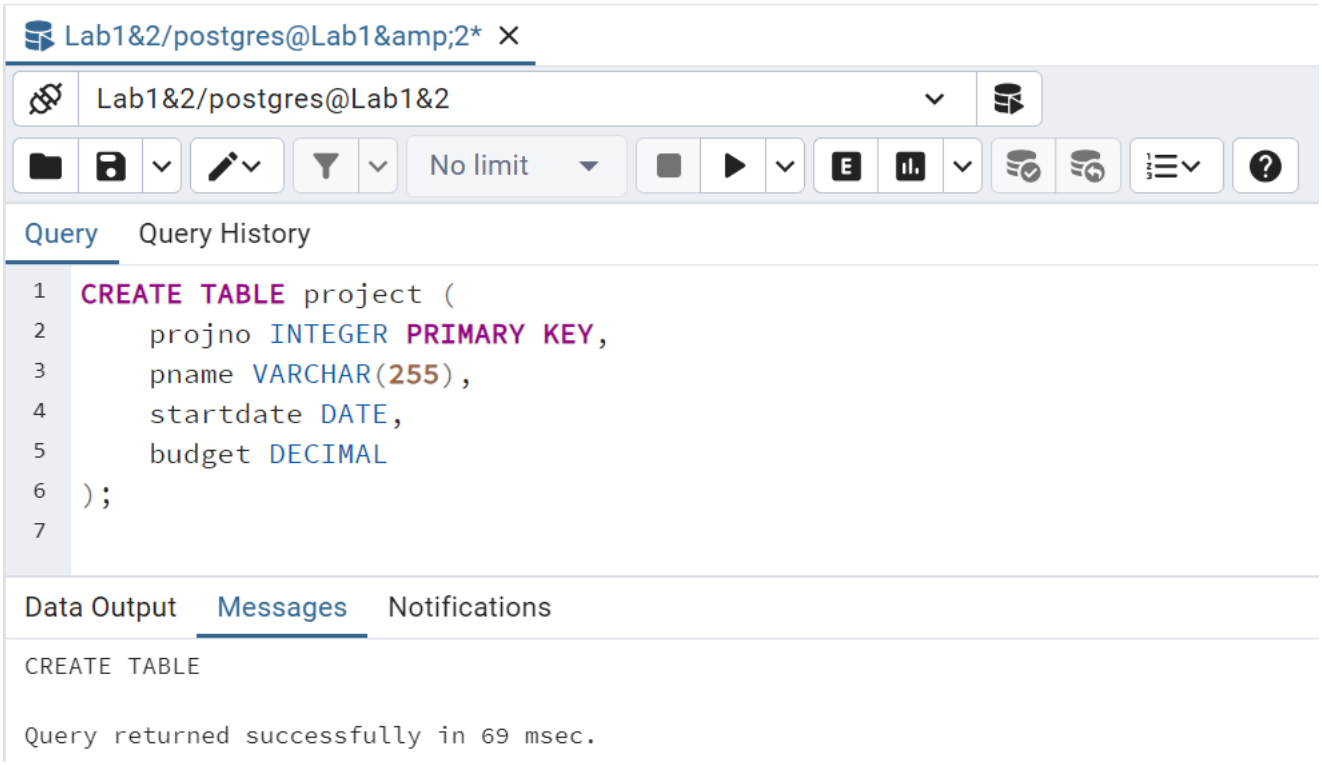
```
1  SELECT E1.ENAME, E1.ENAME, E1.TEL, E2.ENAME AS MANAGER
2  FROM EMP E1 INNER JOIN EMP E2 ON E1.MGR = E2.EMPNO;
```

Data Output | Messages | Notifications

	ename character varying (10)	ename character varying (10)	tel character	manager character varying (10)
1	SMITH	SMITH	0149545243	FORD
2	ALLEN	ALLEN	0149547243	BLAKE
3	WARD	WARD	0149545247	BLAKE
4	JONES	JONES	0149545456	KING
5	MARTIN	MARTIN	0149545784	BLAKE
6	BLAKE	BLAKE	0149545254	KING
7	CLARK	CLARK	0149545245	KING
8	SCOTT	SCOTT	0149545249	JONES
9	TURNER	TURNER	0149548243	BLAKE
10	ADAMS	ADAMS	0149565243	SCOTT
11	JAMES	JAMES	0149545564	BLAKE
12	FORD	FORD	0149785243	JONES
13	MILLER	MILLER	0199545243	CLARK

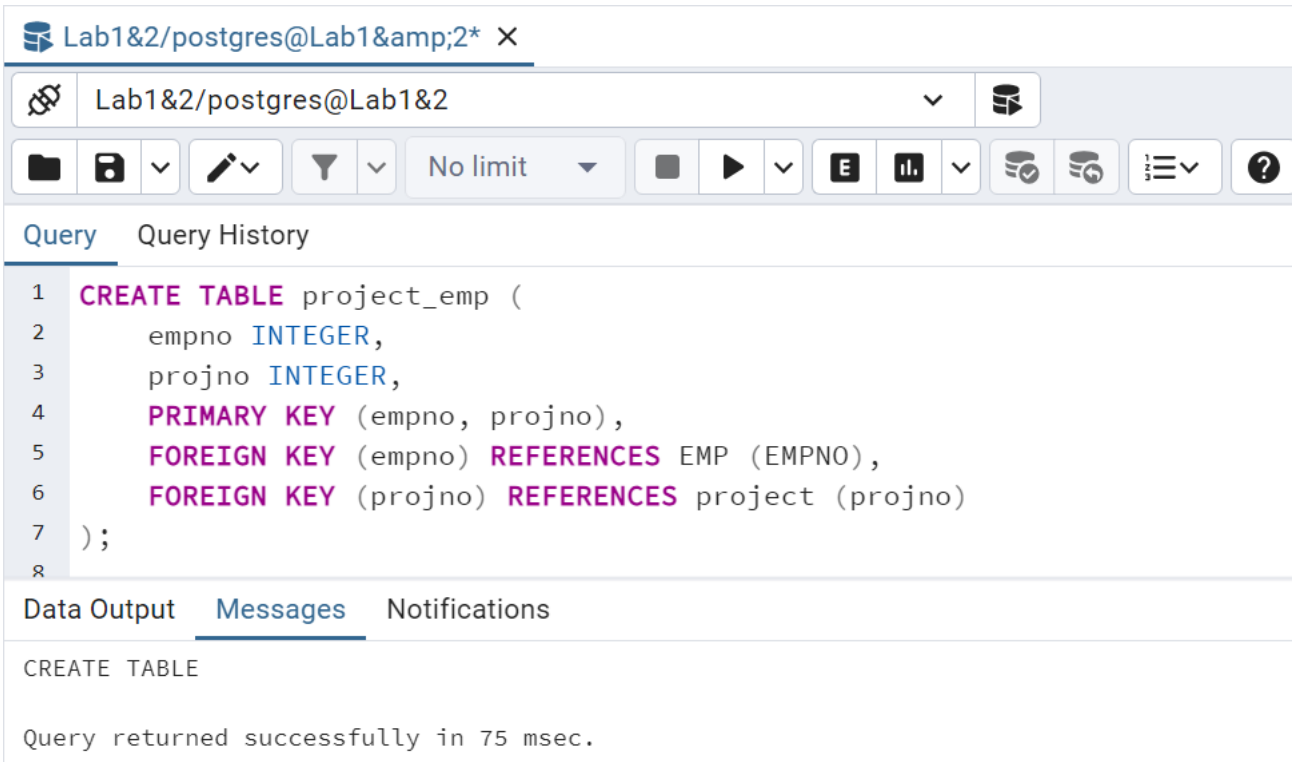
Exercice 3: Join Table

1. Create a Table for employee's Projects:

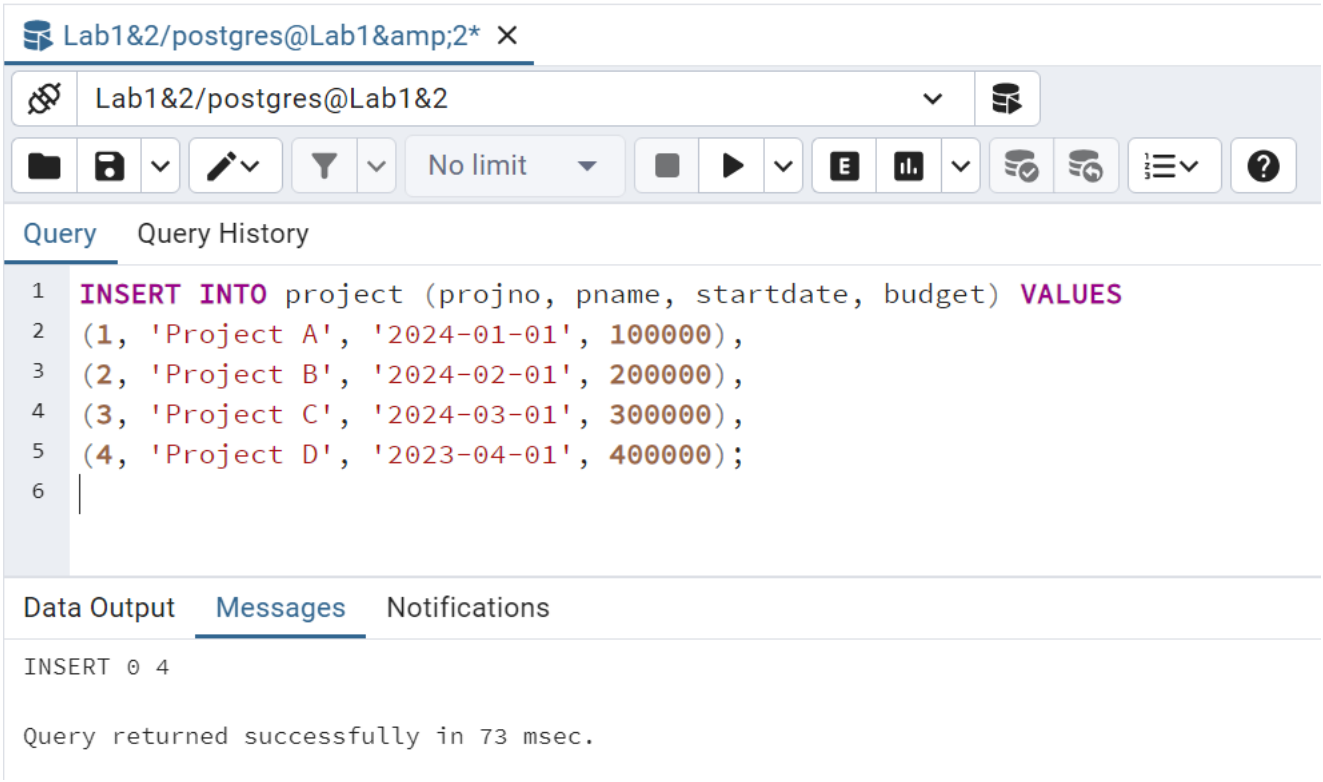


2. Create the Join Table:

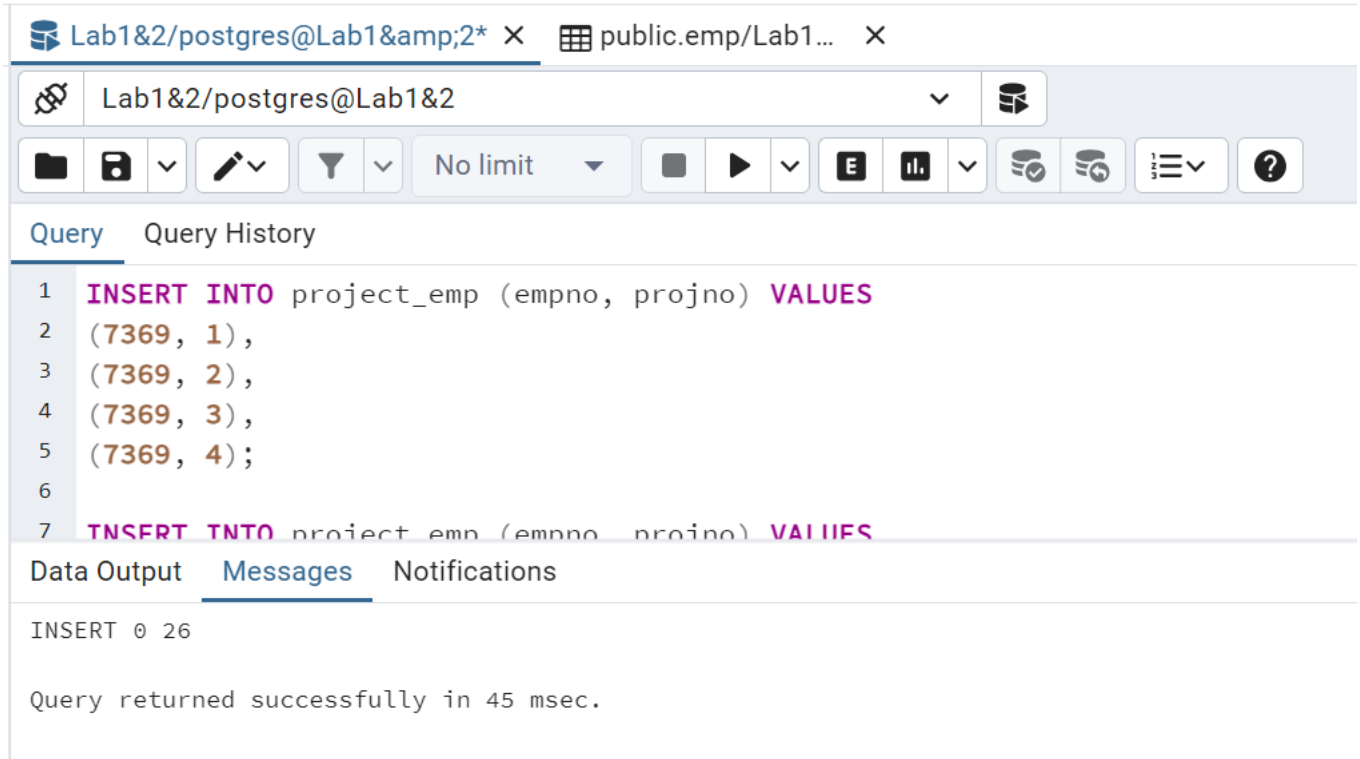
Create table project_emp:



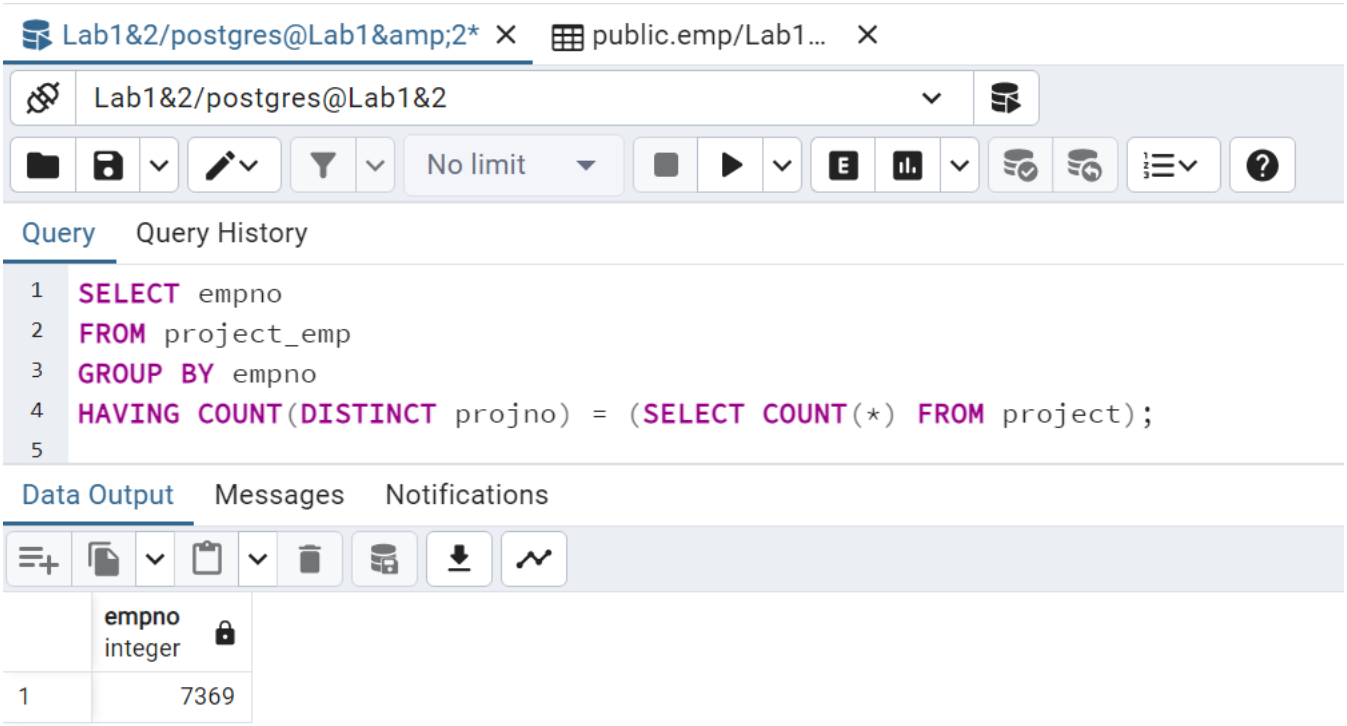
Insert project data into the project table:



Insert project data into the project_emp table:



3. List all employees (by empno) who work on all projects?

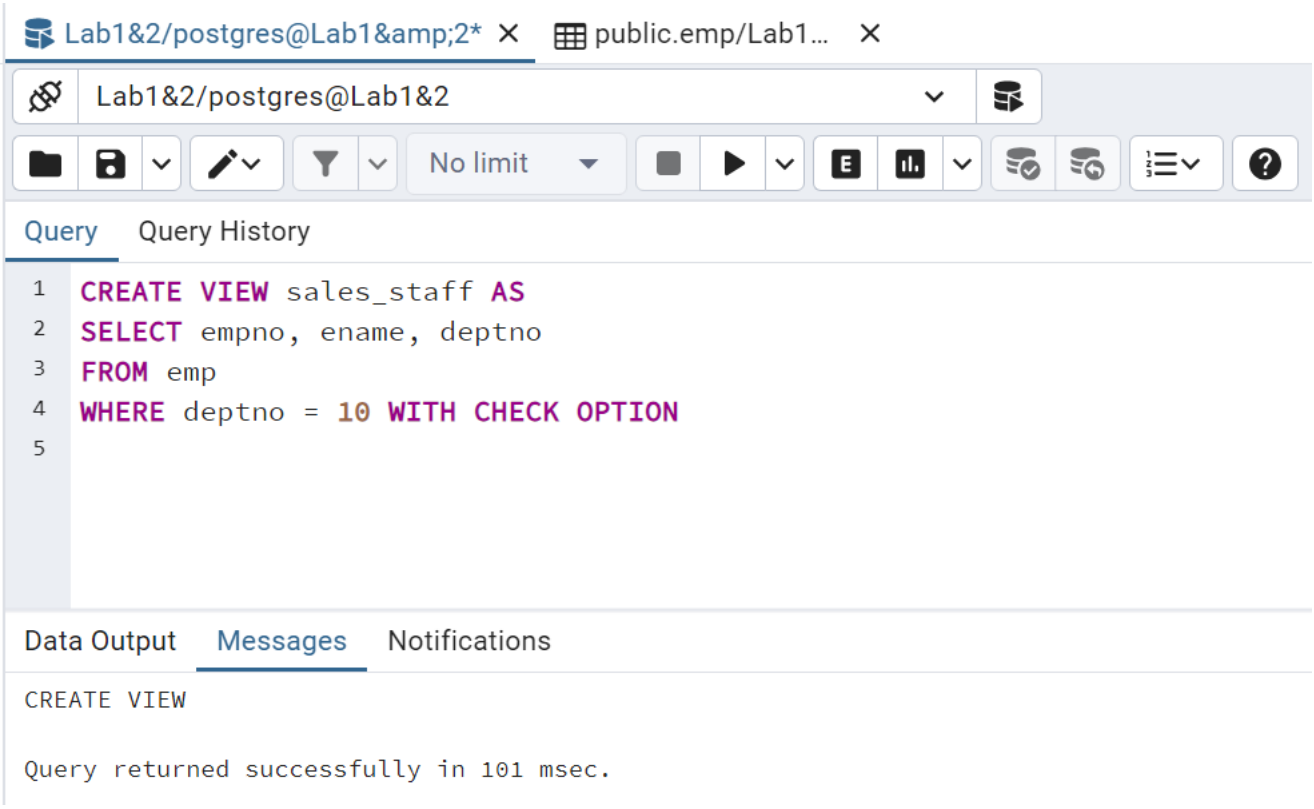


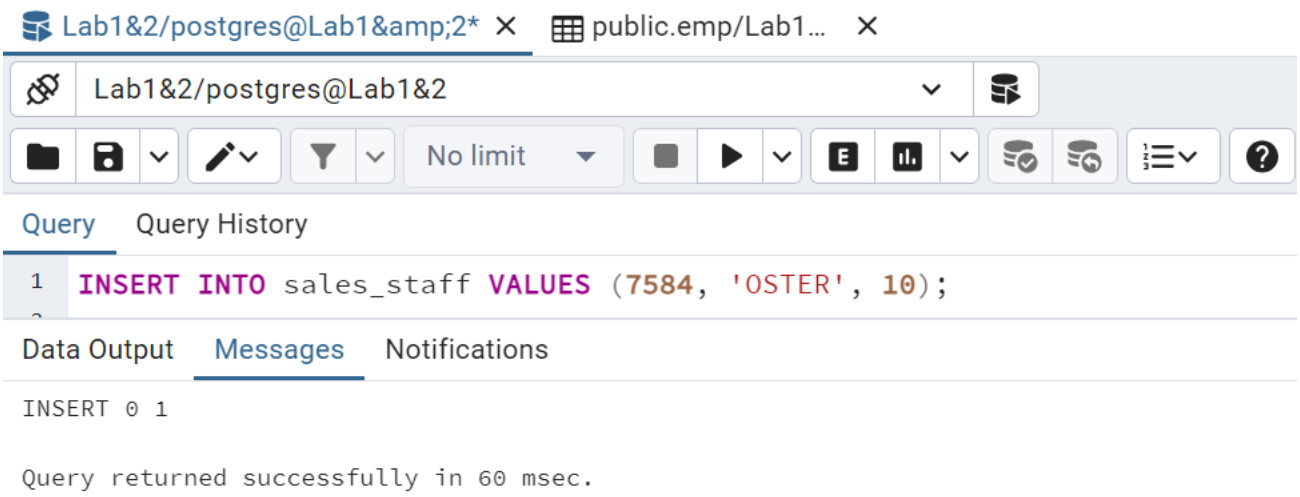
4. Options on View creation:

"CREATE VIEW" statement creates a view named sales_staff, Select the data from the three fields empno, ename and deptno from the emp table, which only contains information about employees with department number 10.

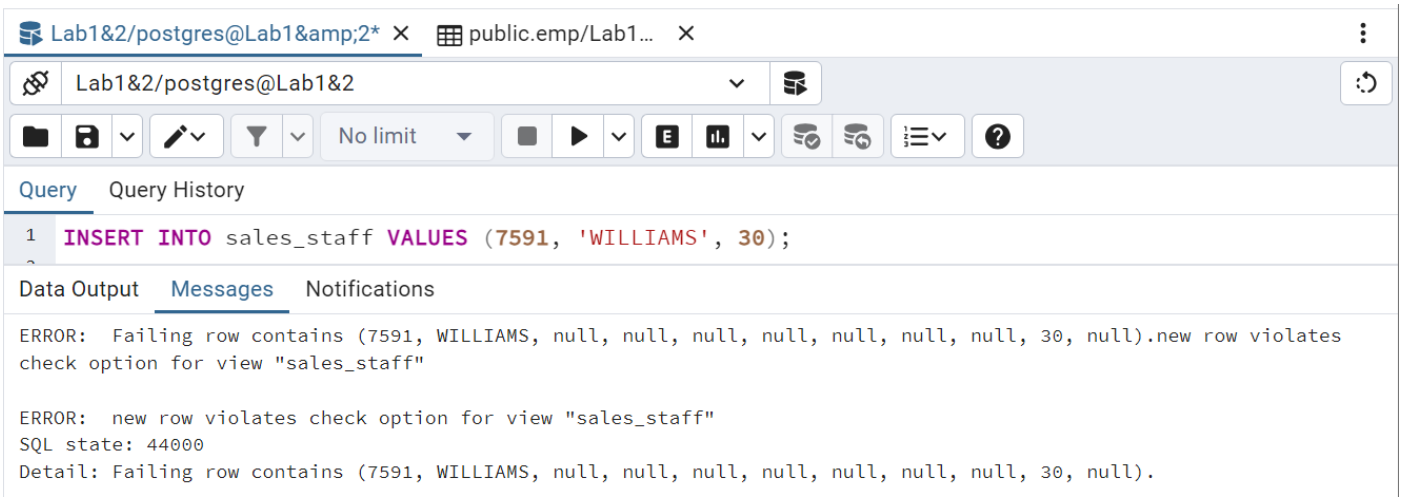
"WITH CHECK OPTION" ensures that all data modifications through the view (including insert and update operations) will comply with the "WHERE "condition in the view definition: all insert or update operations through the sales_staff view must ensure that deptno remains 10.

5. View Creation:



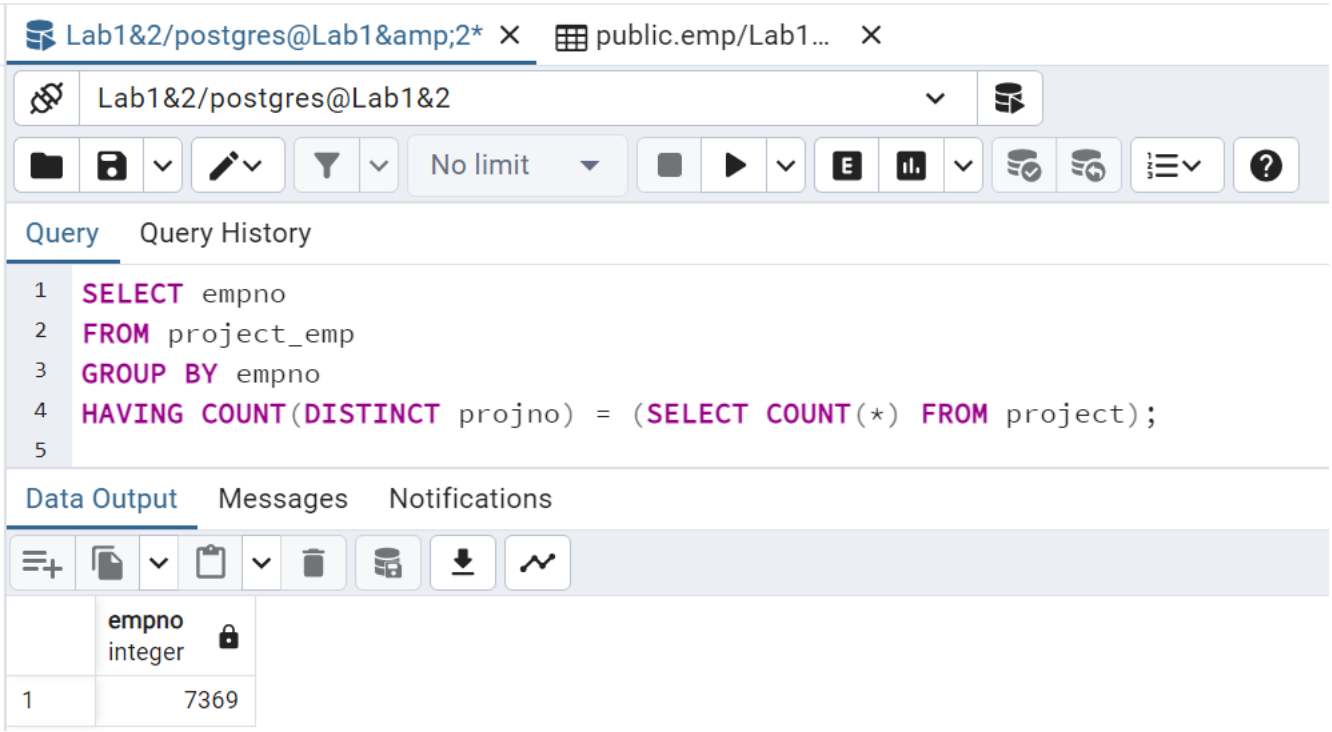


Insert a new record with deptno 10 into the sales_staff view. Insert operations are allowed.

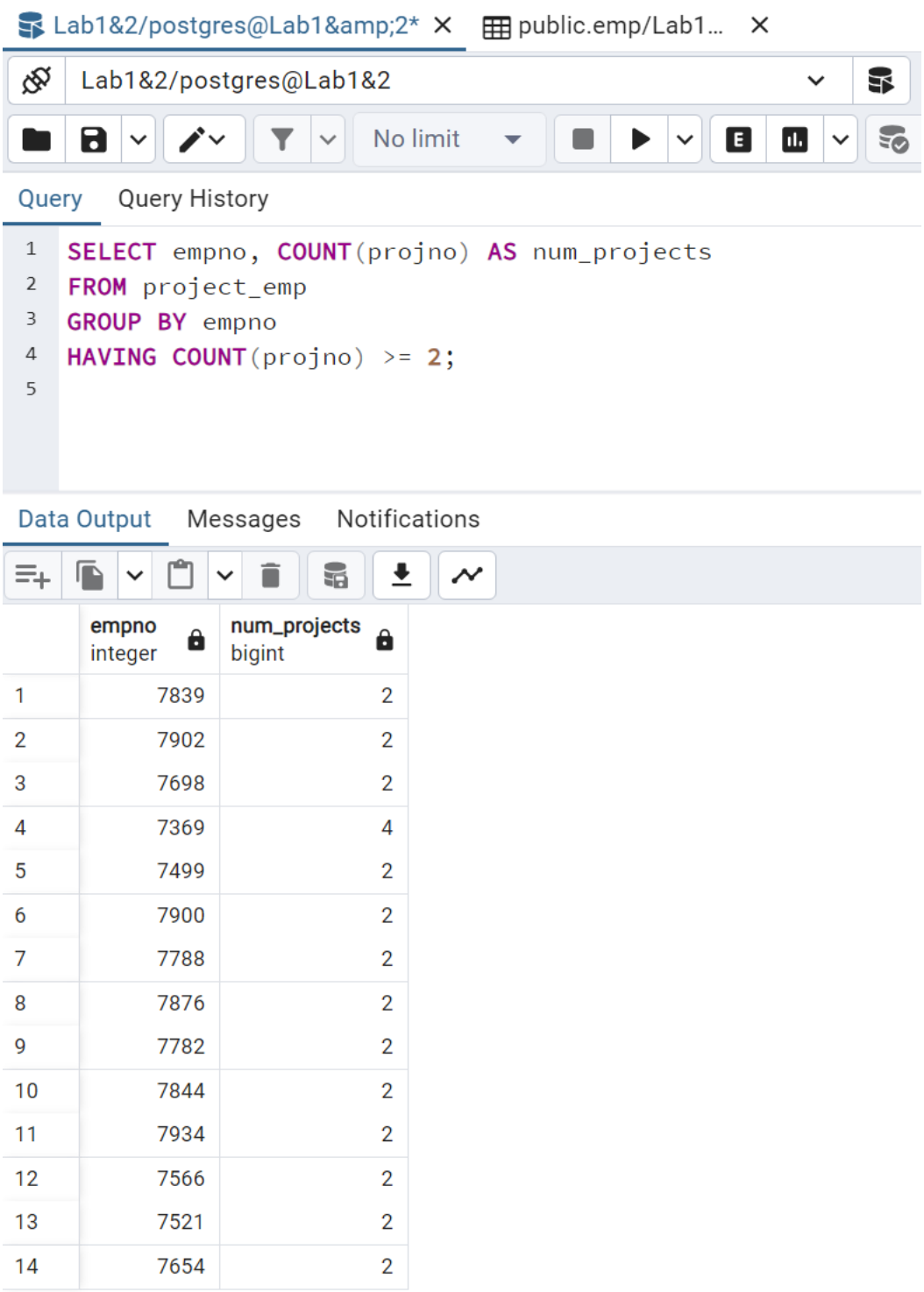


Insert a record with deptno 30 into the sales_staff view. This violates "deptno=10" as well as "WITH CHECK OPTION" . Therefore, this insert operation is rejected.

6. Division Query:



7. Analyze:



	empno integer	num_projects bigint
1	7839	2
2	7902	2
3	7698	2
4	7369	4
5	7499	2
6	7900	2
7	7788	2
8	7876	2
9	7782	2
10	7844	2
11	7934	2
12	7566	2
13	7521	2
14	7654	2

Lab 2-Advanced SQL

Exercice 1: Analytics Queries . Window Queries

1. Gets the 2 persons per department, who have arrived the latest in the company.

```
1  SELECT * FROM (
2  SELECT EMPNO, EFIRST, ENAME, HIREDATE, DEPTNO,
3  RANK() OVER (partition by DEPTNO order by HIREDATE desc) AS RANK
4  FROM EMP
5  )
6  WHERE RANK <= 2;
```

Data Output | Messages | Notifications

	empno [PK] integer	efirst character varying (10)	ename character varying (10)	hiredate date	deptno integer	rank bigint
1	7584	[null]	OSTER	[null]	10	1
2	7934	ALICE	MILLER	1982-01-23	10	2
3	7876	JOSEPH	ADAMS	1983-01-12	20	1
4	7788	GUY	SCOTT	1982-12-09	20	2
5	7900	ALAN	JAMES	1981-12-03	30	1
6	7654	JOE	MARTIN	1981-09-28	30	2

2. Show your analytical Skill and Invents an interesting query using Windows Functions.

Query | Query History

```
1  SELECT EMPNO, EFIRST, ENAME, HIREDATE, DEPTNO, SAL,
2   MIN (SAL) OVER (ORDER BY HIREDATE
3  ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING) AS min
4  FROM EMP
```

Data Output | Messages | Notifications

	empno [PK] integer	efirst character varying (10)	ename character varying (10)	hiredate date	deptno integer	sal integer	min integer
1	7369	JOHN	SMITH	1980-12-17	20	800	800
2	7499	BOB	ALLEN	1981-02-20	30	1600	800
3	7521	PETER	WARD	1981-02-22	30	1250	1250
4	7566	JOHN	JONES	1981-04-02	20	2975	1250
5	7698	BOB	BLAKE	1981-05-01	30	2850	2450
6	7782	JOHN	CLARK	1981-06-09	10	2450	1500
7	7844	PETER	TURNER	1981-09-08	30	1500	1250
8	7654	JOE	MARTIN	1981-09-28	30	1250	1250
9	7839	GUY	KING	1981-11-17	10	5000	950
10	7900	ALAN	JAMES	1981-12-03	30	950	950
11	7902	MARIA	FORD	1981-12-03	20	3000	950
12	7934	ALICE	MILLER	1982-01-23	10	1300	1300
13	7788	GUY	SCOTT	1982-12-09	20	3000	1100
14	7876	JOSEPH	ADAMS	1983-01-12	20	1100	1100
15	7584	[null]	OSTER	[null]	10	[null]	1100

Exercice 2: Index & Explain Plan

1. Execute this script.

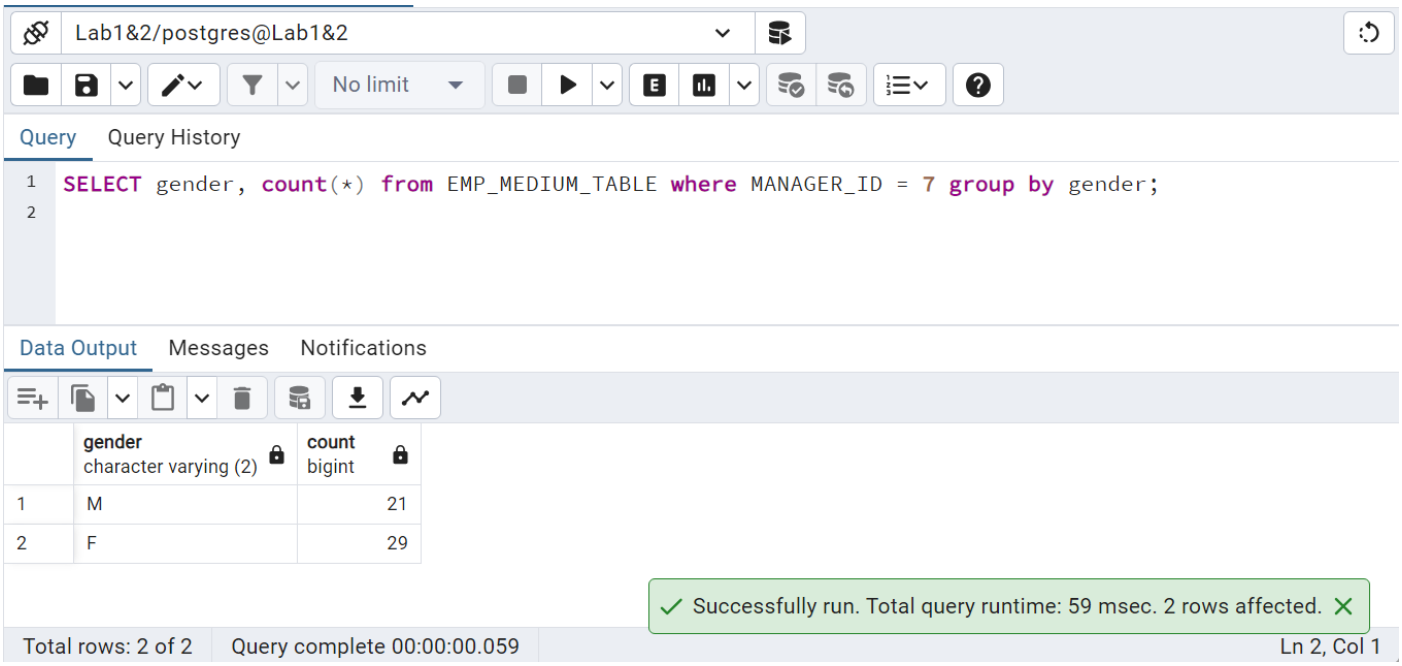
Query | Query History

```
27  FROM generate_series(1,5000) as S;
28  ----------------------------------------
29  ---- Create Join table: Project-Employee -
30  ----------------------------------------
31  CREATE TABLE PROJECT_EMP_MEDIUM_TABLE
32  (PROJECTNO integer,
33  EMPNO integer);
34  INSERT INTO PROJECT_EMP_MEDIUM_TABLE
35  SELECT ROUND(random()*100) AS PROJECTNO,
36  ROUND(random()*100) AS EMPNO
37  FROM generate_series(1,5000) as S;
38
```

Data Output | Messages | Notifications

```
INSERT 0 5000

Query returned successfully in 262 msec.
```

2. The goal of all exercice will be to tune the following query:



3. 1 Use EXPLAIN plan to analyze the query

Query | Query History

```
1 EXPLAIN SELECT gender, count(*) from EMP_MEDIUM_TABLE where MANAGER_ID = 7 group by
2 gender;
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	HashAggregate (cost=114.75..114.77 rows=2 width=10)
2	Group Key: gender
3	-> Seq Scan on emp_medium_table (cost=0.00..114.50 rows=50 width=...
4	Filter: (manager_id = 7)

3. 2 Activate stats.

```
#local preload libraries = ''
#session preload libraries = ''
shared preload libraries = 'pg stat statements' # (change requires restart)
#jit provider = 'llvmjit'              # JIT library to use
```

Run:

Lab1&2/postgres@Lab1&2

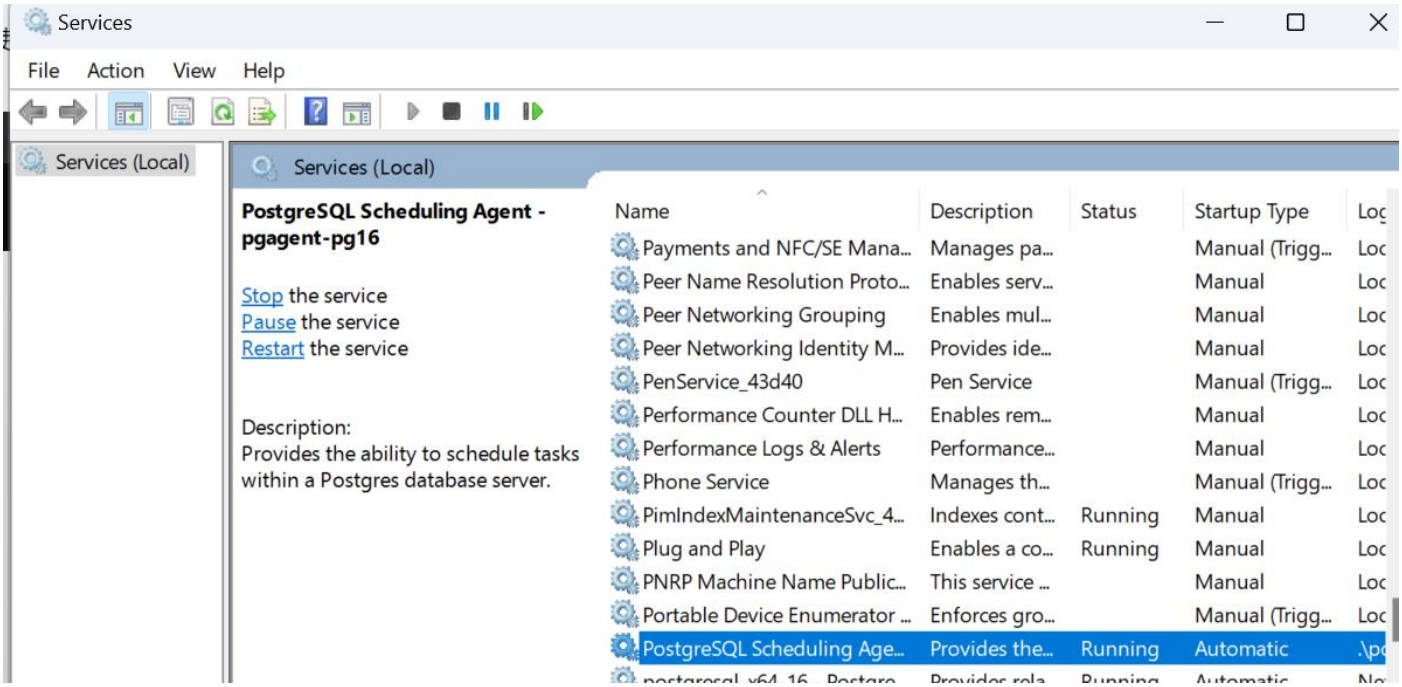
No limit

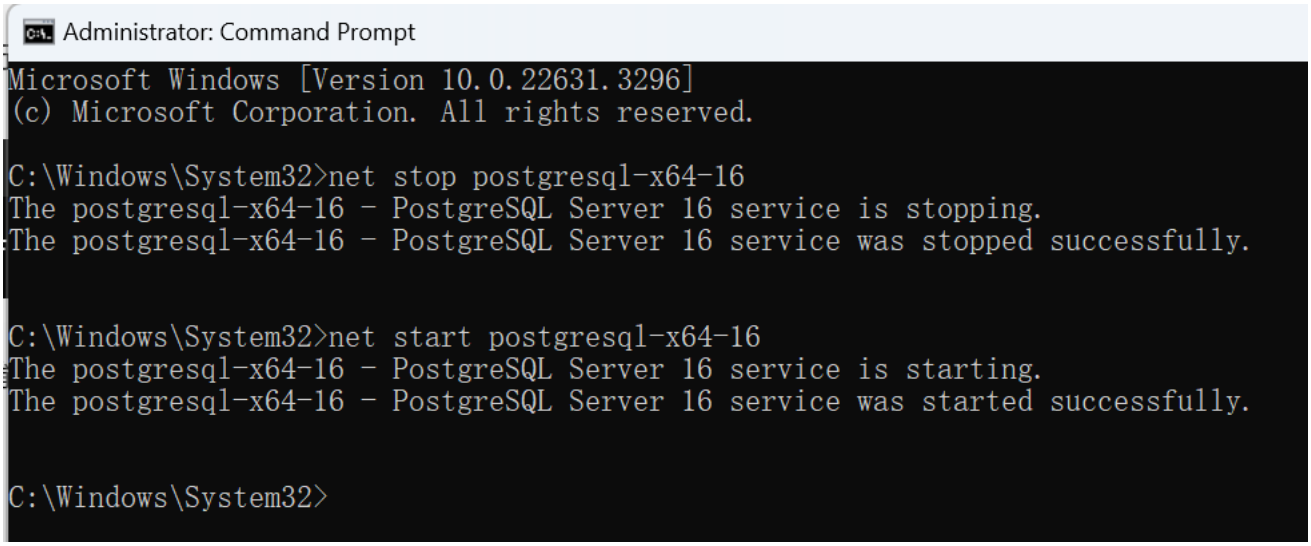
Query | Query History

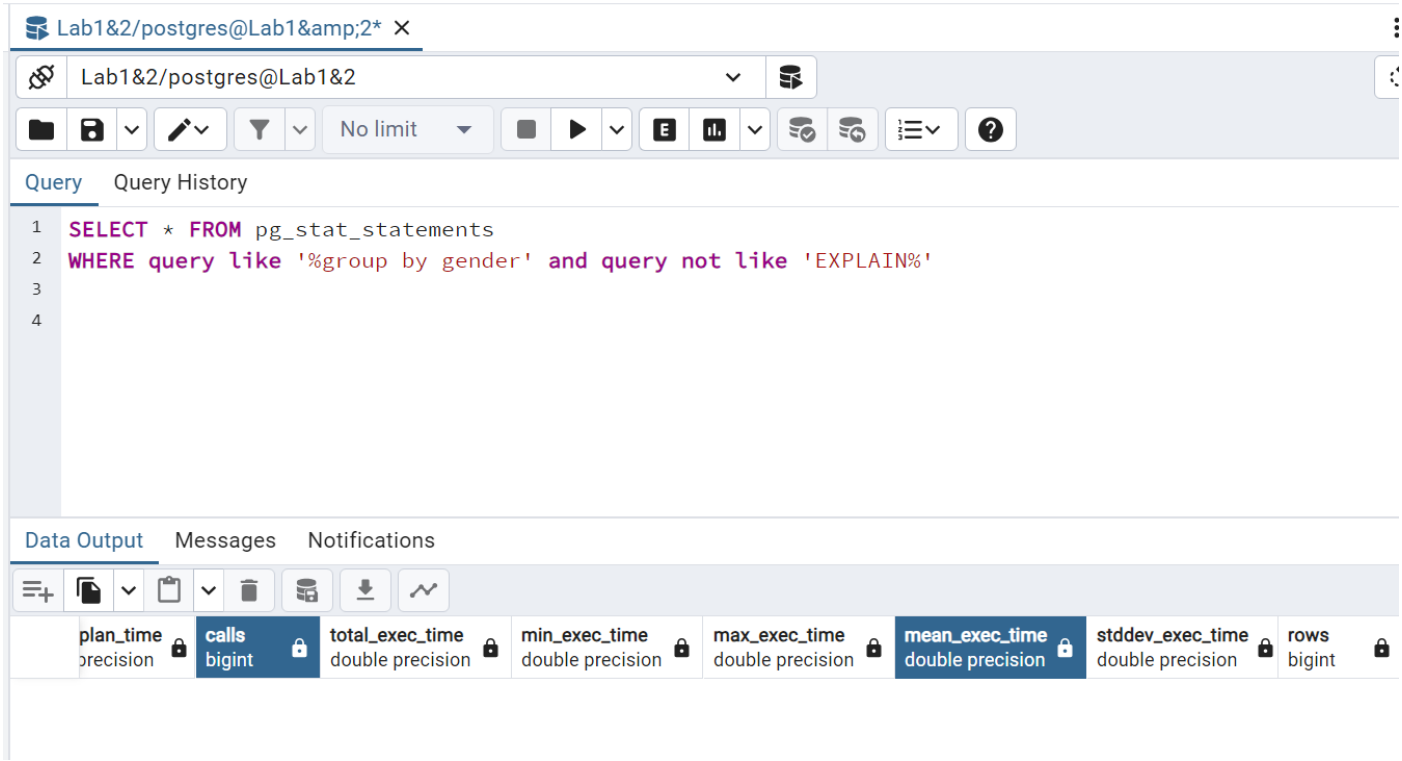
```
1 create extension pg_stat_statements;
2
```

Data Output | Messages | Notifications

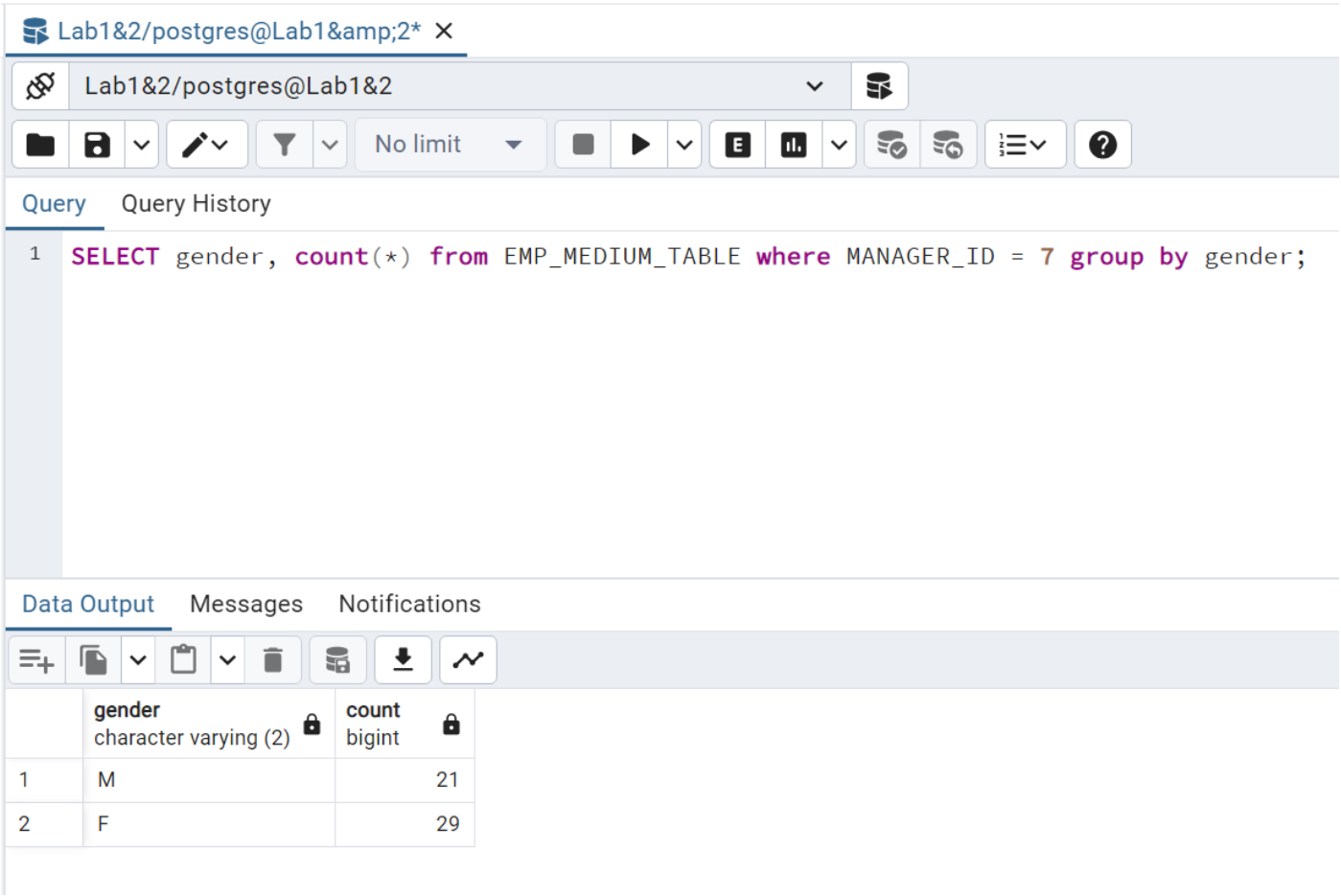
```
CREATE EXTENSION

Query returned successfully in 154 msec.
```

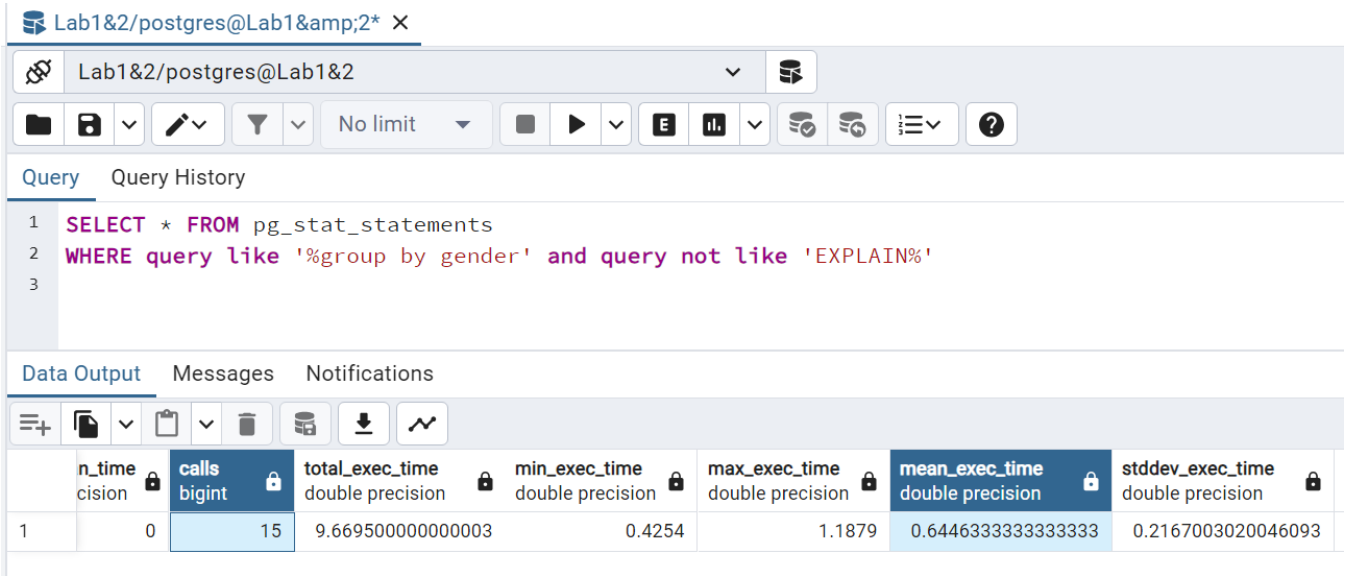

Services
File Action View Help
Services (Local)
Services (Local)
PostgreSQL Scheduling Agent - pgagent-pg16
Stop the service
Pause the service
Restart the service
Description:
Provides the ability to schedule tasks within a Postgres database server.
Name
Description
Status
Startup Type
Payments and NFC/SE Mana...
Manages pa...
Manual (Trigg...
Peer Name Resolution Proto...
Enables serv...
Manual
Peer Networking Grouping
Enables mul...
Manual
Peer Networking Identity M...
Provides ide...
Manual
PenService_43d40
Pen Service
Manual (Trigg...
Performance Counter DLL H...
Enables rem...
Manual
Performance Logs & Alerts
Performance...
Manual
Phone Service
Manages th...
Manual (Trigg...
PimIndexMaintenanceSvc_4...
Indexes cont...
Running
Manual
Plug and Play
Enables a co...
Running
Manual
PNRP Machine Name Public...
This service ...
Manual
Portable Device Enumerator ...
Enforces gro...
Manual (Trigg...
PostgreSQL Scheduling Age...
Provides the...
Running
Automatic


Administrator: Command Prompt
Microsoft Windows [Version 10.0.22631.3296]
(c) Microsoft Corporation. All rights reserved.
C:\Windows\System32>net stop postgresql-x64-16
The postgresql-x64-16 - PostgreSQL Server 16 service is stopping.
The postgresql-x64-16 - PostgreSQL Server 16 service was stopped successfully.
C:\Windows\System32>net start postgresql-x64-16
The postgresql-x64-16 - PostgreSQL Server 16 service is starting.
The postgresql-x64-16 - PostgreSQL Server 16 service was started successfully.
C:\Windows\System32>

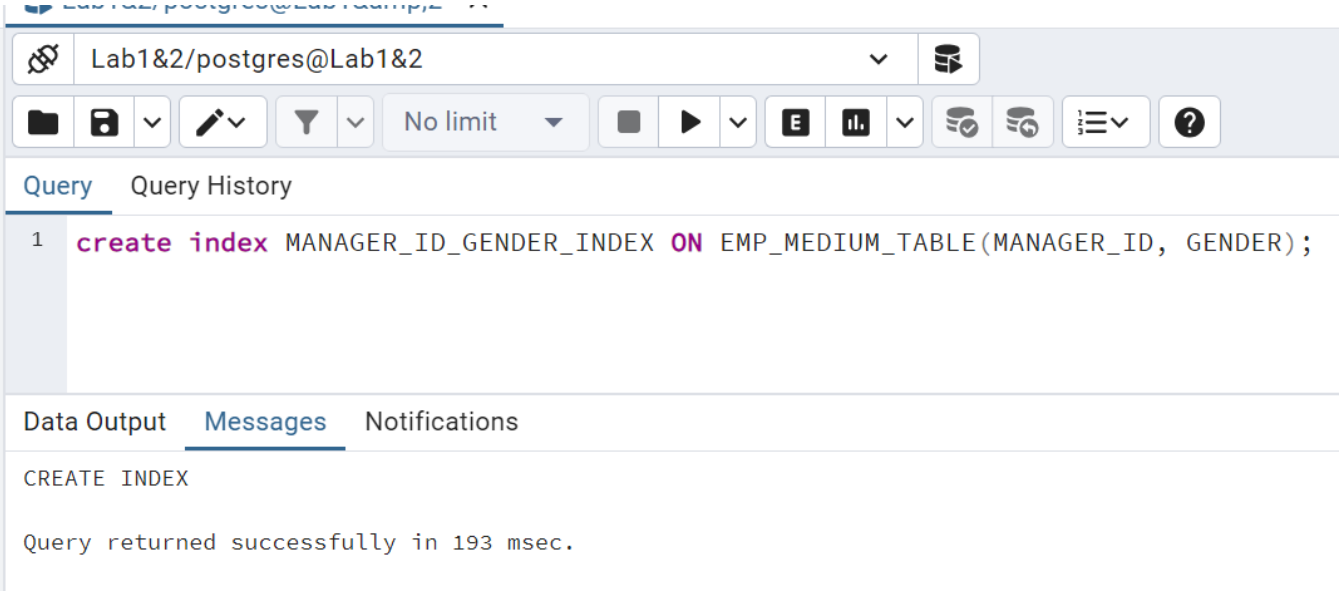

Lab1&2/postgres@Lab1&2*
Lab1&2/postgres@Lab1&2
No limit
Query
Query History
1 SELECT * FROM pg_stat_statements
2 WHERE query like '%group by gender' and query not like 'EXPLAIN%'
3
4
Data Output
Messages
Notifications
plan_time double precision
calls bigint
total_exec_time double precision
min_exec_time double precision
max_exec_time double precision
mean_exec_time double precision
stddev_exec_time double precision
rows bigint

Re-run 10 times:

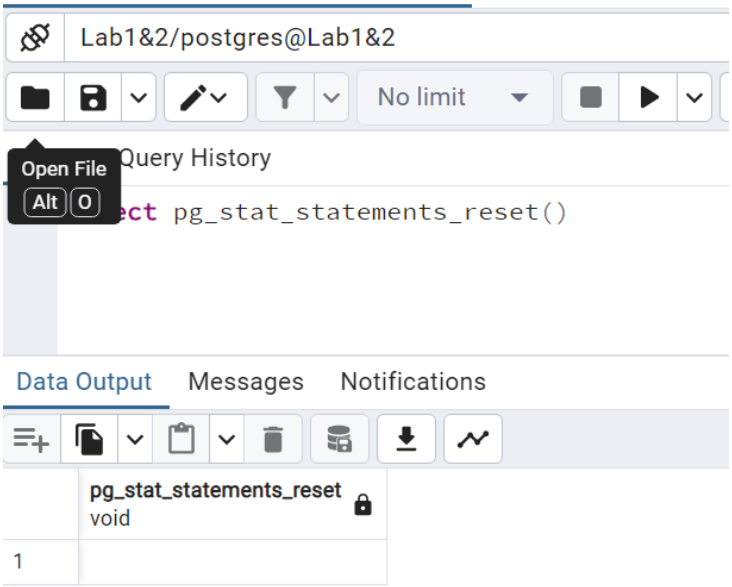


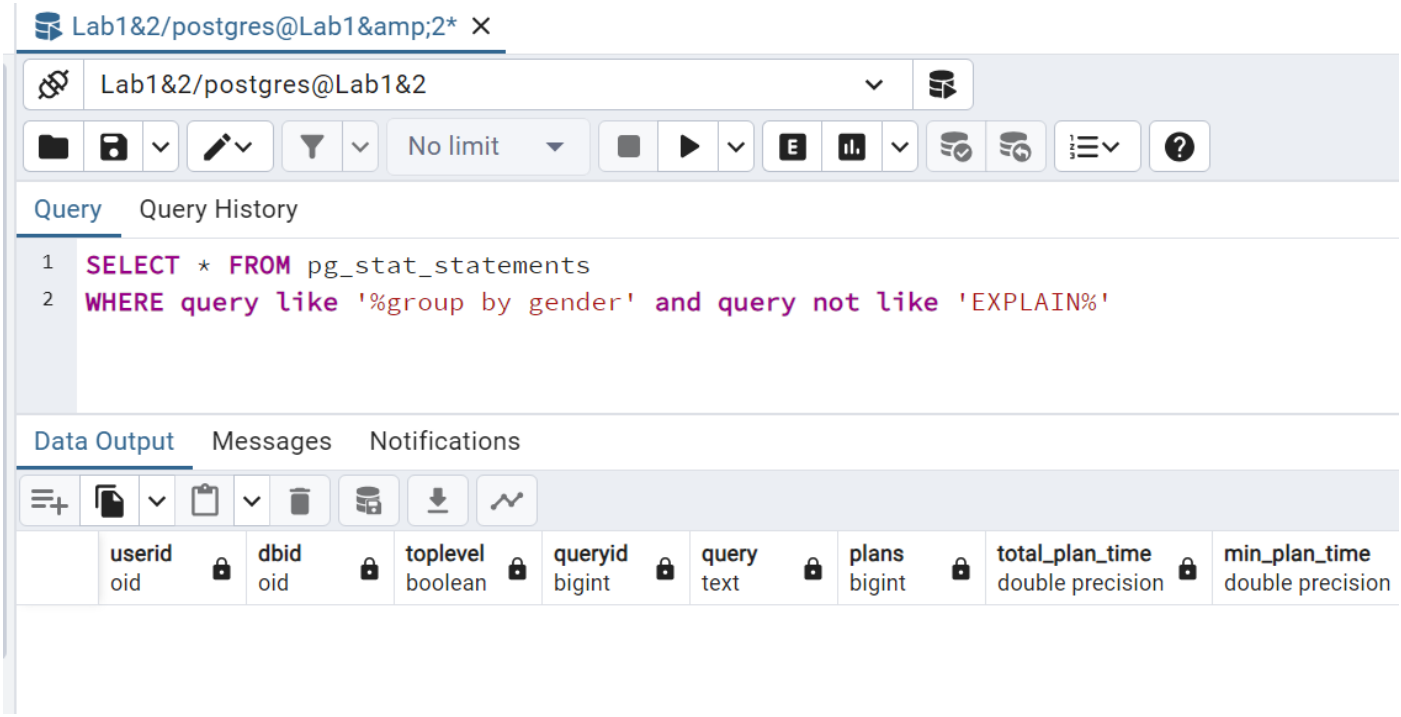


4. Add a covering index on both columns fetched:

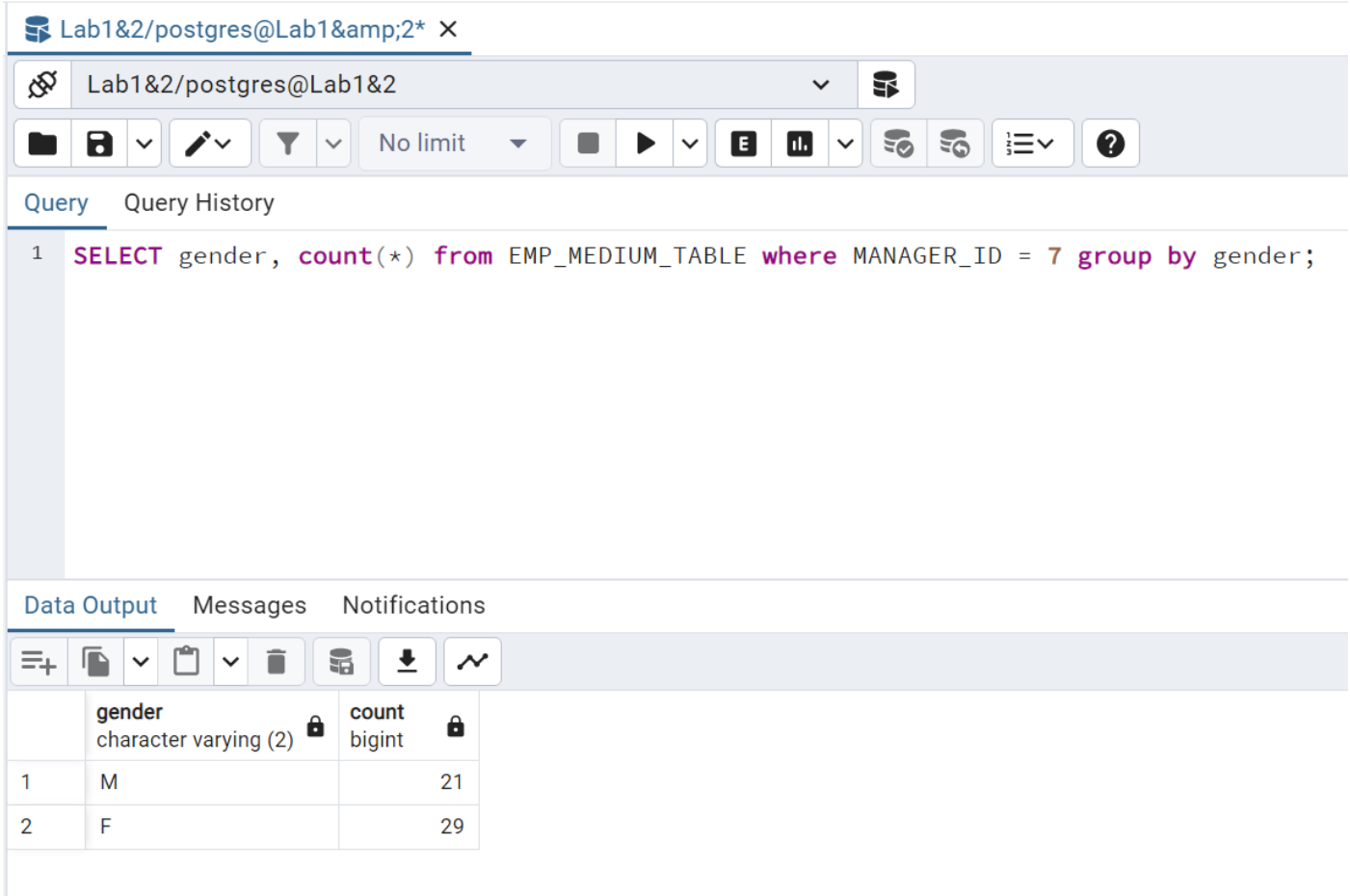


5. Reset the stats:

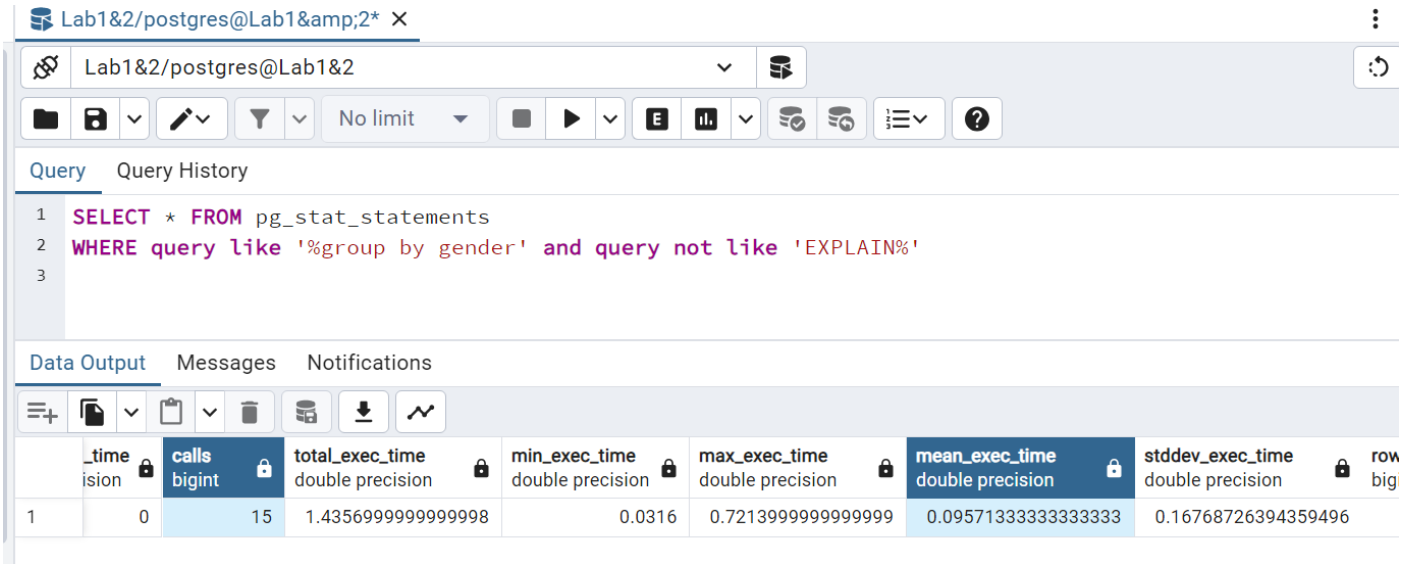




6. Re-run query the same query 15 times:



7. Use the EXPLAIN plan again and compare the plan before the INDEX



For the query performance recorded in the “pg_stat_statements” view before and after the creation of the index “MANAGER_ID_GENDER_INDEX”:

1) the “Calls” remained at 15 times.
2) the “Mean Exec Time” dropped significantly.
3) the value shows that after the creation of the index, the average execution of the query time is reduced, which is direct evidence that indexes optimize query performance.
4) By creating indexes on the MANAGER_ID and GENDER columns, PostgreSQL can locate these specific rows with MANAGER_ID 7 more quickly and group and count more efficiently because the indexes provide This reduces the number of data pages that must be read and the amount of work that queries must perform.

Exercice 3: Data Dictionary

Lab1&2/postgres@Lab1&2

No limit

Query | Query History

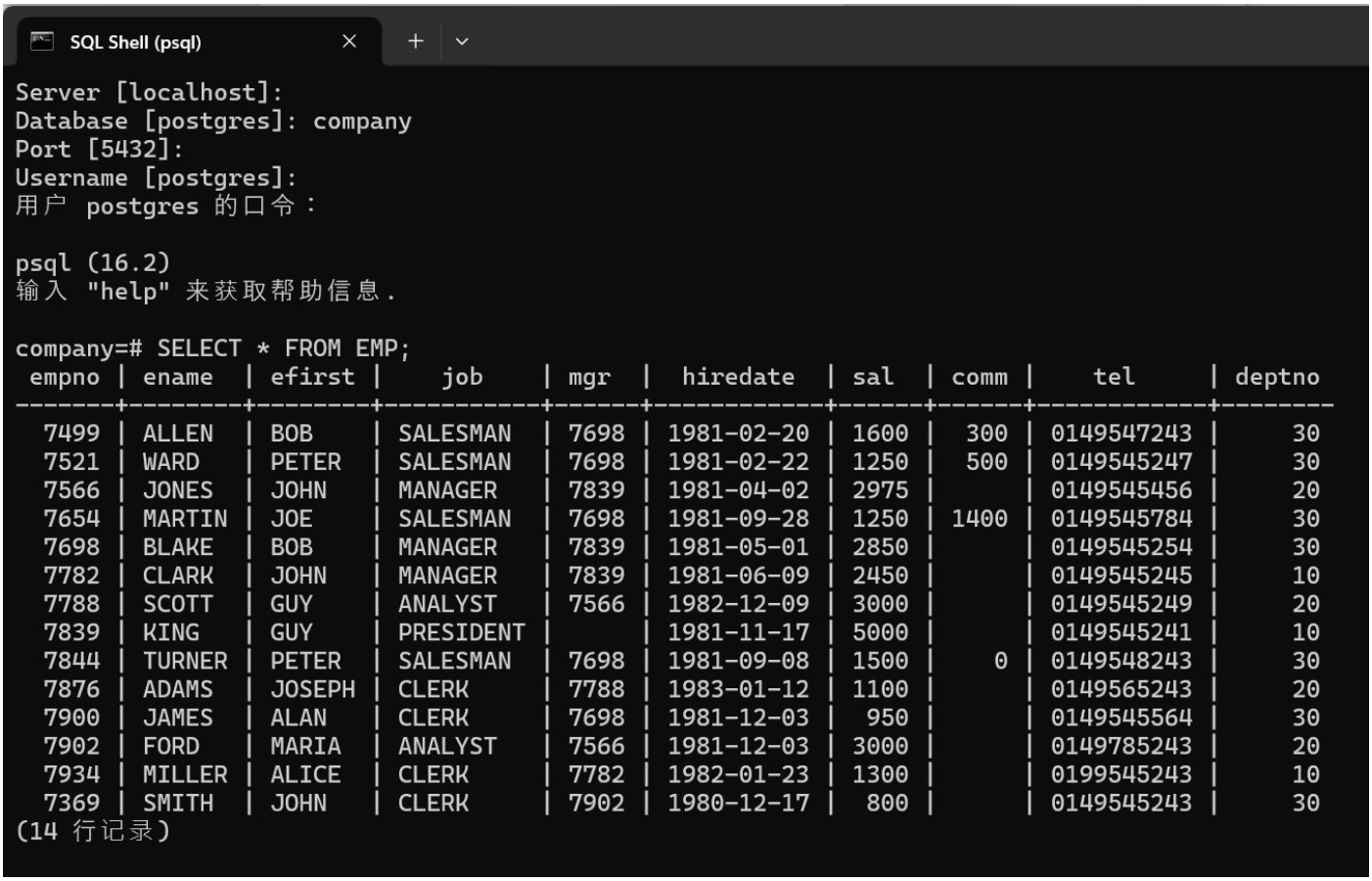
```
1  DROP TABLE IF EXISTS MY_OBJECTS;
2
3  CREATE TABLE MY_OBJECTS (
4      Object_name VARCHAR(100),
5      Type VARCHAR(50)
6  );
7
8  INSERT INTO MY_OBJECTS (Object_name, Type)
9  SELECT table_name, 'Table'
10 FROM information_schema.tables
11 WHERE table_schema = 'public';
12
13 INSERT INTO MY_OBJECTS (Object_name, Type)
14 SELECT column_name, 'Column'
15 FROM information_schema.columns
16 WHERE table_schema = 'public';
17
18 INSERT INTO MY_OBJECTS (Object_name, Type)
19 SELECT conname, 'Constraint'
20 FROM pg_constraint
21 WHERE connamespace = 'public'::regnamespace;
22
23 SELECT * FROM MY_OBJECTS;
```

Data Output | Messages | Notifications

	object_name character varying (100)	type character varying (50)
1	dept	Table
2	dependents	Table
3	bonus	Table
4	salgrade	Table
5	emp	Table
6	project_emp	Table
7	project	Table
8	sales_staff	Table
9	emp_medium_table	Table
10	project_medium_table	Table
11	project_emp_medium_table	Table
12	pg_stat_statements_info	Table
13	pg_stat_statements	Table
14	my_objects	Table
15	grade	Column

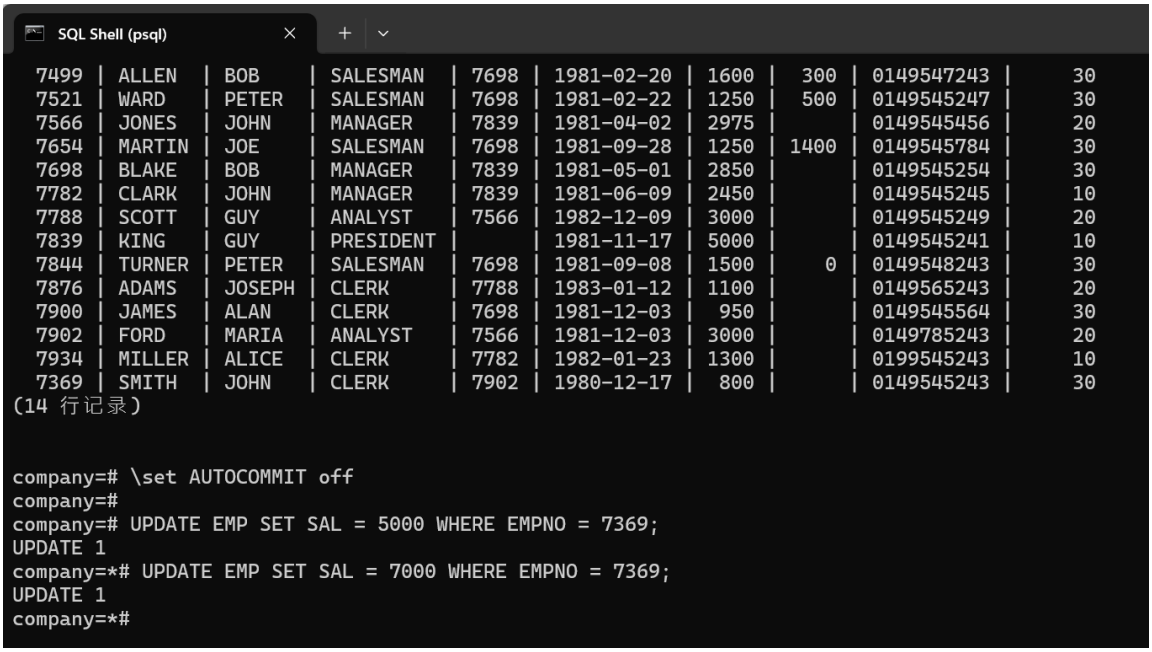
93	efirst	Column
94	jit_functions	Column
95	local_blks_hit	Column
96	ename	Column
97	mean_exec_time	Column
98	name	Column
99	losal	Column
100	tel	Column
101	dname	Column
102	userid	Column
103	sal	Column
104	shared_blks_written	Column
105	pk_dept	Constraint
106	ck_sal	Constraint
107	pk_emp	Constraint
108	fk_emp_dept	Constraint
109	pk_dependent	Constraint
110	emp_unique	Constraint
111	chk_mobile_tel	Constraint
112	fk_dependent_emp	Constraint
113	project_pkey	Constraint
114	project_emp_pkey	Constraint
115	project_emp_empno_fkey	Constraint
116	project_emp_projno_fkey	Constraint

Exercice 4: Use postgres via CLI



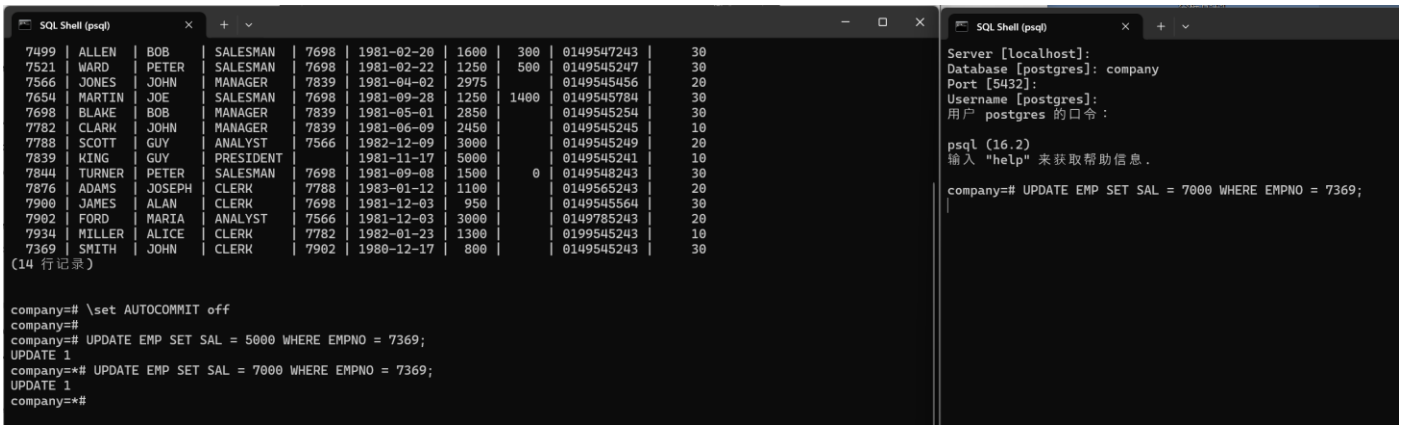
Exercice 5: Transaction Part 1 – Beginner

1.& 2. & 3.



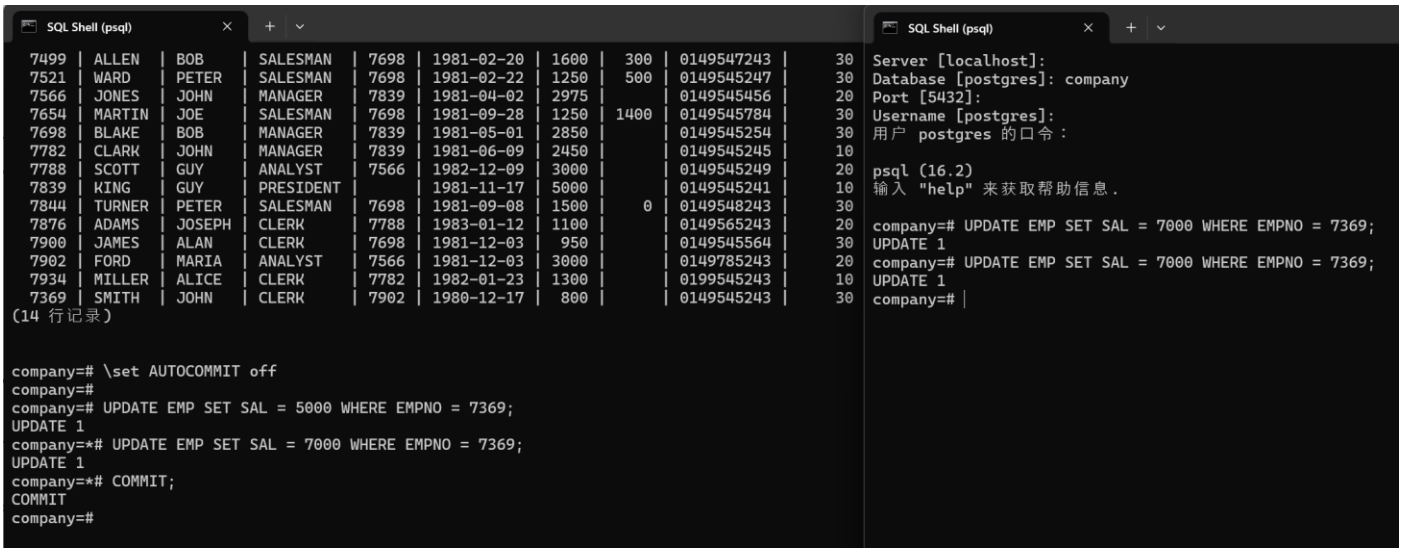
This update will not take effect immediately to the database because the transaction has not yet been committed.

4.-5.



Since the first client's transaction has not yet committed, this command may hang waiting because the first client's transaction has a lock on the record. To make the update available for the second client, the first client should COMMIT.

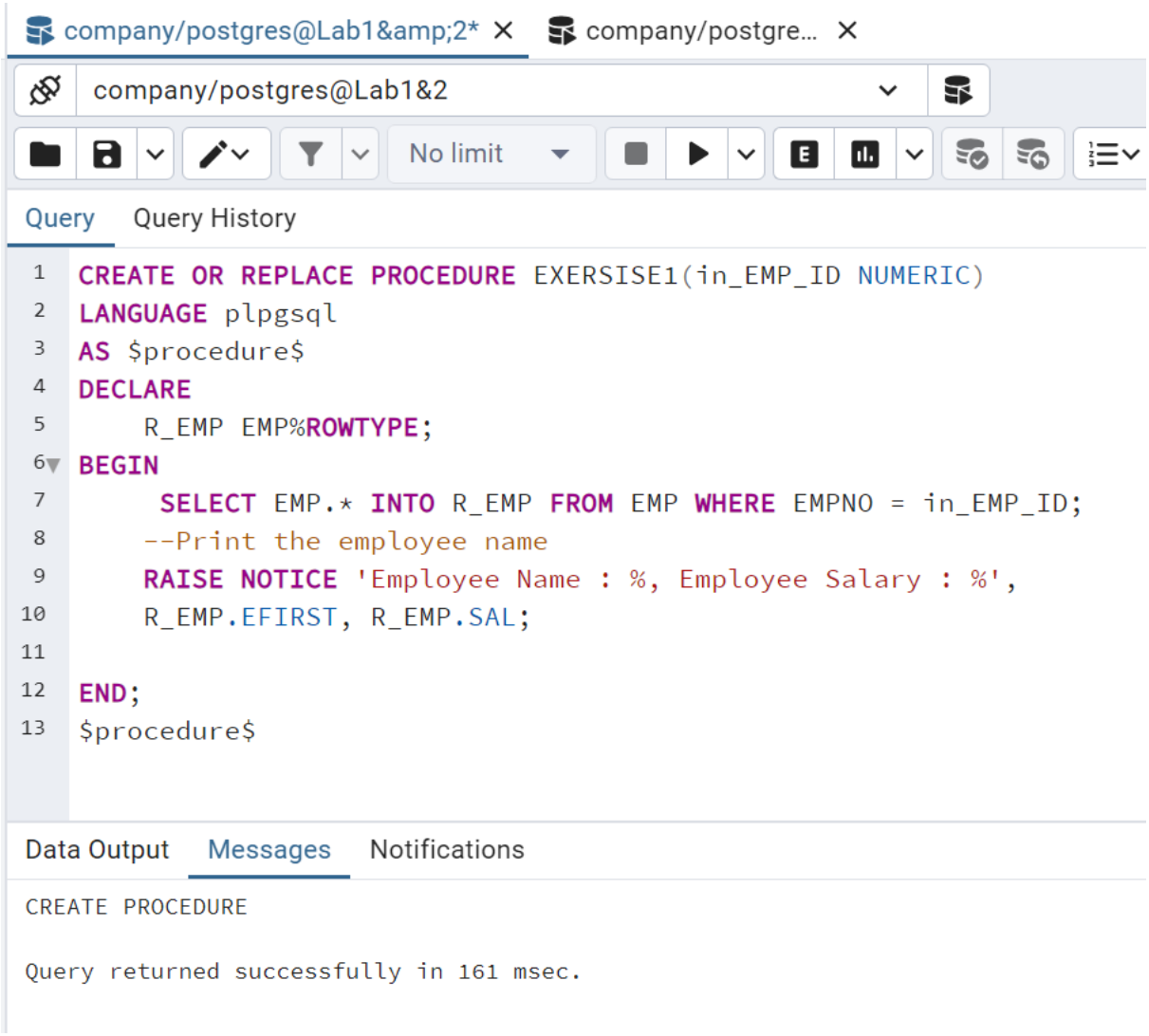
6.-7.

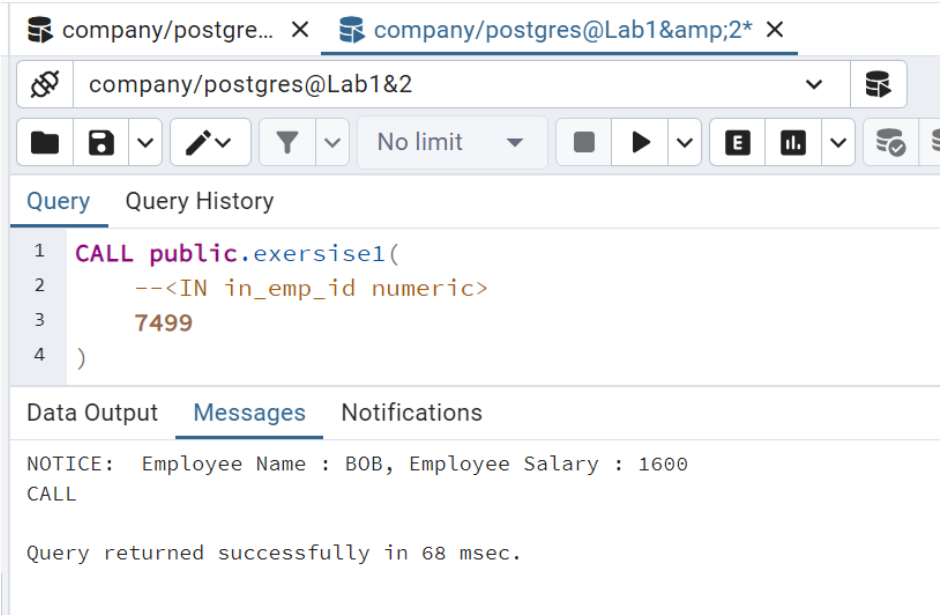


The first client's transaction has been committed, and the second client's update operation has been successfully executed.

Lab 2-Part 2 PL/SQL

Exercice 1: Functions





Exercice 2: Procedure & Display & Cursors

company/postgres@Lab1&2

No limit

Query | Query History

```
1  CREATE OR REPLACE PROCEDURE EXERCISE2(in_EMP_ID NUMERIC)
2  LANGUAGE plpgsql
3  AS $$
4  DECLARE
5      R_EMP RECORD;
6      AVG_SAL NUMERIC;
7
8  BEGIN
9
10     SELECT * FROM EMP EMP WHERE EMPNO = in_emp_ID;
11     SELECT AVG(SAL) into AVG_SAL FROM EMP
12         WHERE JOB IN(SELECT JOB FROM EMP WHERE EMPNO = in_EMP_ID);
13
14     RAISE NOTICE 'Employee Name: %, Employee Salary: %', R_EMP.ENAME, R_EMP.SAL;
15     RAISE NOTICE 'Average Salary for Job: %', AVG_SAL;
16 EXCEPTION
17 WHEN OTHERS THEN
18     RAISE NOTICE 'SQL ERROR OG QUERY : % ', SQLERRM;
19
20 END;
21 $$;
```

Data Output | Messages | Notifications

```
CREATE PROCEDURE

Query returned successfully in 305 msec.
```

company/postgre... | company/postgres@Lab1&2*

company/postgres@Lab1&2

No limit

Query | Query History

```
1  CALL public.exersise2(
2      --<IN in_emp_id numeric>
3      7499
4  )
```

Data Output | Messages | Notifications

```
NOTICE:  SQL ERROR OG QUERY : query has no destination for result data
CALL

Query returned successfully in 115 msec.
```

Exercice 3. Procedure & Update

company/postgres@Lab1&2* company/postgre...

company/postgres@Lab1&2

No limit

Query Query History

```
1  CREATE OR REPLACE PROCEDURE exercise3(in_EMP_ID NUMERIC)
2  LANGUAGE plpgsql
3  AS $procedure$
4  DECLARE
5      R_EMP EMP%ROWTYPE;
6      AVG_SAL NUMERIC;
7  BEGIN
8      SELECT * INTO R_EMP FROM emp WHERE EMPNO = in_EMP_ID;
9
10     SELECT AVG(sal) INTO AVG_SAL FROM emp;
11
12     RAISE NOTICE 'Initial salary: %', R_EMP.sal;
13
14     IF R_EMP.sal >= AVG_SAL THEN
15         UPDATE emp SET sal = sal * 1.10 WHERE EMPNO = in_EMP_ID;
16     ELSE
17         UPDATE emp SET sal = AVG_SAL WHERE EMPNO = in_EMP_ID;
18     END IF;
19
20     SELECT sal INTO R_EMP.sal FROM emp WHERE EMPNO = in_EMP_ID;
21     RAISE NOTICE 'Resulting salary: %', R_EMP.sal;
22 END;
23 $procedure$;
```

Data Output Messages Notifications

```
CREATE PROCEDURE

Query returned successfully in 123 msec.
```

company/postgre... company/postgres@Lab1&2*

company/postgres@Lab1&2

No limit

Query Query History

```
1  CALL public.exercise3(
2      --<IN in_emp_id numeric>
3      7499
4  )
```

Data Output Messages Notifications

```
NOTICE:  Initial salary: 1600
NOTICE:  Resulting salary: 2516
CALL

Query returned successfully in 108 msec.
```

Exercice 4. Procedure

company/postgres@Lab1&2* × company/postgre... × company/postgre... ×

company/postgres@Lab1&2

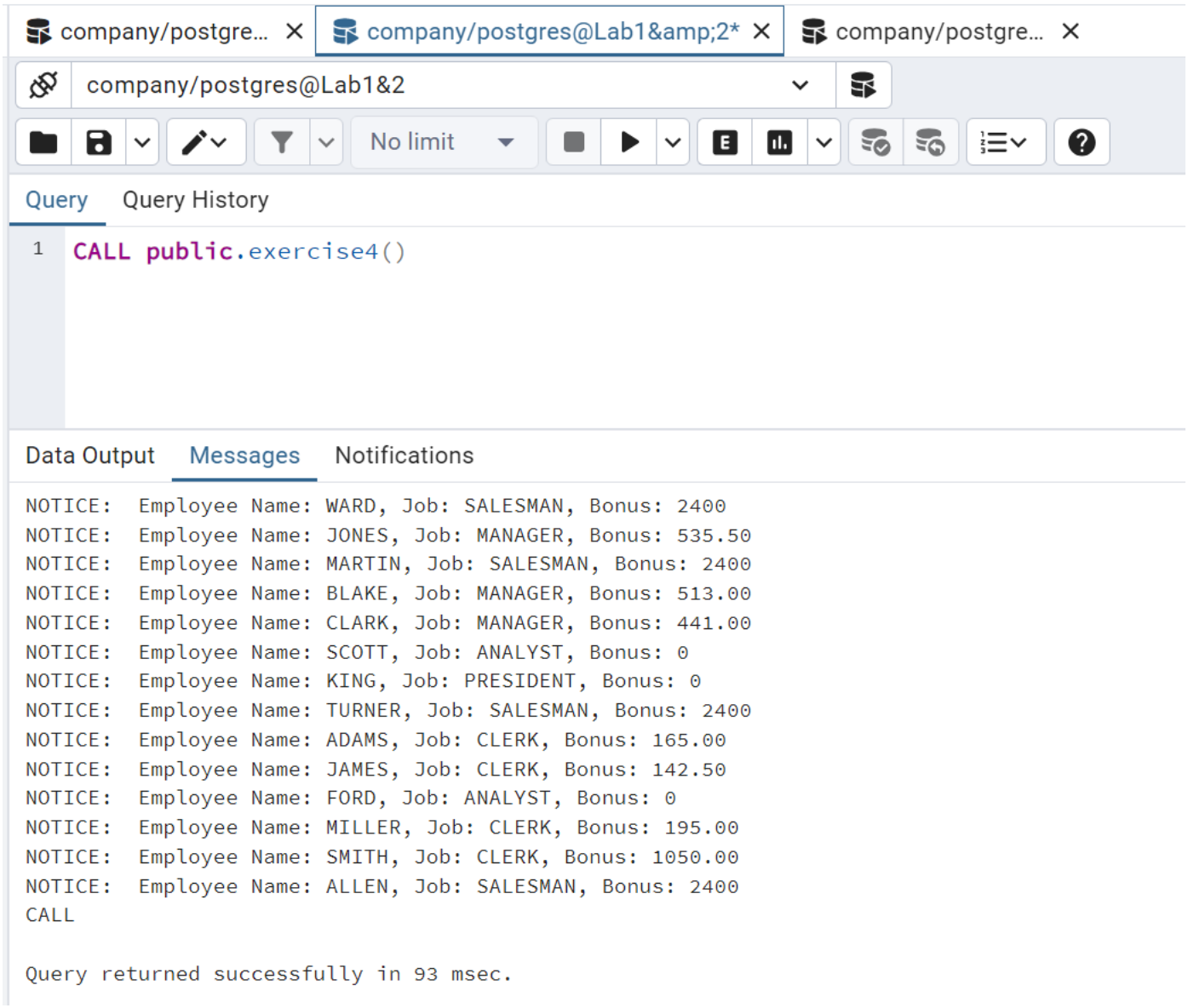
No limit

Query Query History

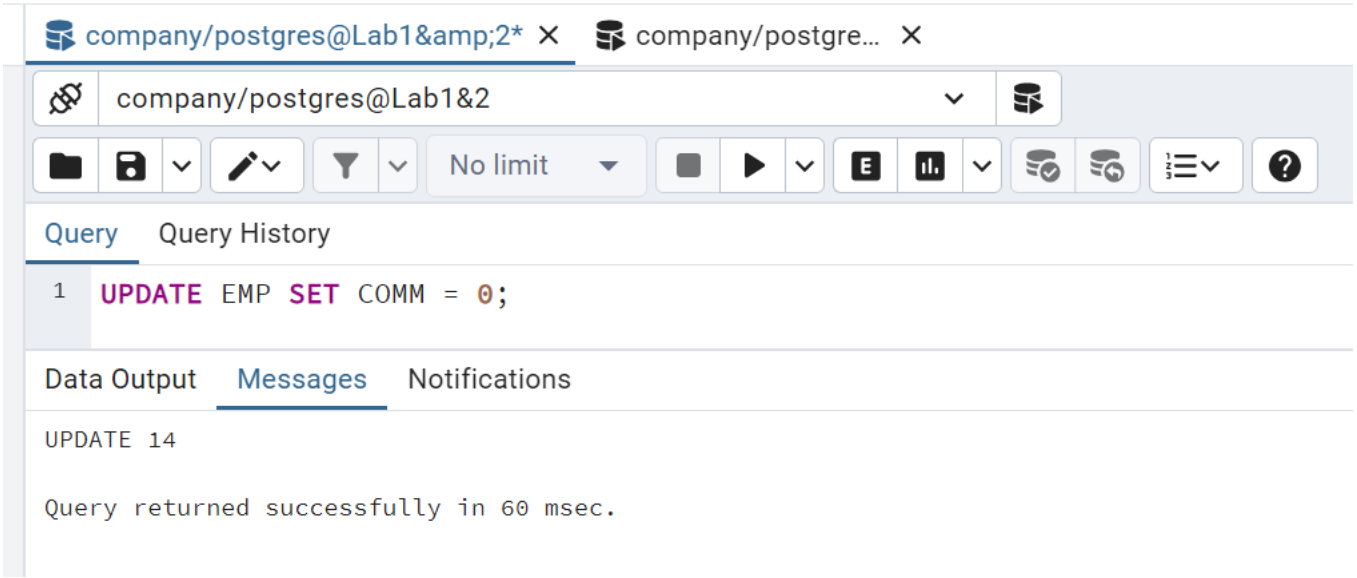
```
1  CREATE OR REPLACE PROCEDURE exercise4()
2  LANGUAGE plpgsql
3  AS $procedure$
4  DECLARE
5      R_EMP RECORD;
6      bonus NUMERIC;
7  BEGIN
8
9      IF NOT EXISTS (SELECT 1 FROM information_schema.columns
10                    WHERE table_name = 'bonus' AND column_name = 'job' AND data_type = 'text')
11                    THEN
12         ALTER TABLE bonus ALTER COLUMN job TYPE TEXT;
13     END IF;
14
15
16     IF NOT EXISTS (SELECT 1 FROM information_schema.columns
17                    WHERE table_name = 'bonus' AND column_name = 'bonus' AND data_type = 'numeric')
18                    THEN
19         ALTER TABLE bonus ADD COLUMN bonus NUMERIC;
20     END IF;
21
22     FOR R_EMP IN SELECT ename, job, comm, sal FROM emp LOOP
23         CASE R_EMP.job
24             WHEN 'SALESMAN' THEN bonus := R_EMP.comm * 2;
25             WHEN 'CLERK' THEN bonus := R_EMP.sal * 0.15;
26             WHEN 'MANAGER' THEN bonus := R_EMP.sal * 0.18;
27             ELSE bonus := 0;
28         END CASE;
29
30         INSERT INTO bonus (ename, job, bonus) VALUES (R_EMP.ename, R_EMP.job, bonus);
31
32
33         RAISE NOTICE 'Employee Name: %, Job: %, Bonus: %', R_EMP.ename, R_EMP.job, bonus;
34     END LOOP;
```

Data Output Messages Notifications

```
CREATE PROCEDURE

Query returned successfully in 78 msec.
```



Exercice 5. SELECT for UPDATE



company/postgres@Lab1&2* | company/postgre... | company/postgre...

company/postgres@Lab1&2

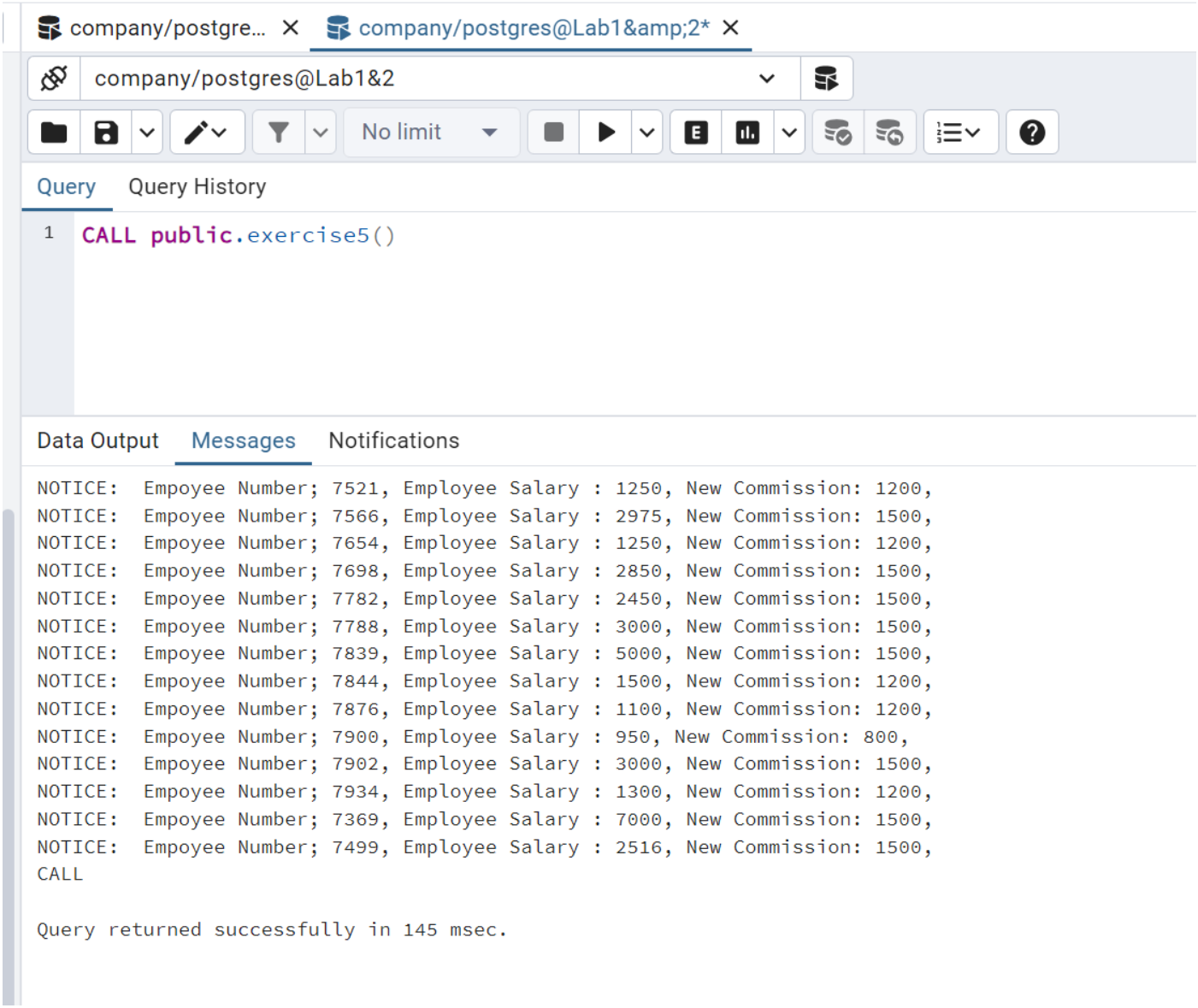
No limit

Query | Query History

```
1  CREATE OR REPLACE PROCEDURE EXERCISE5()
2  language plpgsql
3  AS $procedure$
4
5  DECLARE
6      ONE_EMP EMP%ROWTYPE;
7      NEW_COMM numeric;
8      E_SAL numeric;
9      SAL_EMP CURSOR FOR SELECT * FROM EMP FOR UPDATE;
10
11 BEGIN
12     --Open the cursor
13     OPEN SAL_EMP;
14     --Looping in the cursor
15     LOOP
16         FETCH SAL_EMP into ONE_EMP;
17         EXIT WHEN NOT FOUND;
18         --Cursor usage
19         E_SAL := ONE_EMP.SAL;
20         IF E_SAL <= 1000 THEN
21             NEW_COMM := 800;
22         ELSE IF E_SAL <= 2000 THEN
23             NEW_COMM := 1200;
24         ELSE
25             NEW_COMM := 1500;
26         END IF;
27         END IF;
28
29         UPDATE EMP SET COMM = NEW_COMM WHERE EMPNO = ONE_EMP.EMPNO;
30         raise notice 'Empoyee Number; %, Employee Salary : %, New Commission: %,',
31         ONE_EMP.EMPNO, ONE_EMP.SAL, NEW_COMM;
32
33     END LOOP;
34     --Closing the cursor
35     CLOSE SAL_EMP;
36
37 END
38 $procedure$
```

Data Output | Messages | Notifications

```
CREATE PROCEDURE

Query returned successfully in 74 msec.
```


company/postgre...
company/postgres@Lab1&2*
company/postgres@Lab1&2
No limit
Query
Query History
1
CALL public.exercise5()
Data Output
Messages
Notifications
NOTICE: Empoyee Number; 7521, Employee Salary : 1250, New Commission: 1200,
NOTICE: Empoyee Number; 7566, Employee Salary : 2975, New Commission: 1500,
NOTICE: Empoyee Number; 7654, Employee Salary : 1250, New Commission: 1200,
NOTICE: Empoyee Number; 7698, Employee Salary : 2850, New Commission: 1500,
NOTICE: Empoyee Number; 7782, Employee Salary : 2450, New Commission: 1500,
NOTICE: Empoyee Number; 7788, Employee Salary : 3000, New Commission: 1500,
NOTICE: Empoyee Number; 7839, Employee Salary : 5000, New Commission: 1500,
NOTICE: Empoyee Number; 7844, Employee Salary : 1500, New Commission: 1200,
NOTICE: Empoyee Number; 7876, Employee Salary : 1100, New Commission: 1200,
NOTICE: Empoyee Number; 7900, Employee Salary : 950, New Commission: 800,
NOTICE: Empoyee Number; 7902, Employee Salary : 3000, New Commission: 1500,
NOTICE: Empoyee Number; 7934, Employee Salary : 1300, New Commission: 1200,
NOTICE: Empoyee Number; 7369, Employee Salary : 7000, New Commission: 1500,
NOTICE: Empoyee Number; 7499, Employee Salary : 2516, New Commission: 1500,
CALL
Query returned successfully in 145 msec.

Exercice 6. Condition on EXIT loop

company/postgres@Lab1&2* company/postgre...

company/postgres@Lab1&2

No limit

Query Query History

```
1  CREATE OR REPLACE PROCEDURE exercise6()
2  LANGUAGE plpgsql
3  AS $procedure$
4  DECLARE
5      R_EMP EMP%ROWTYPE;
6      counter INT := 0;
7  BEGIN
8
9      RAISE NOTICE 'Highest salary employees:';
10     FOR R_EMP IN SELECT * FROM EMP ORDER BY sal DESC LIMIT 5 LOOP
11         RAISE NOTICE 'Employee Number: %, Employee Salary: %', R_EMP.empno, R_EMP.sal;
12     END LOOP;
13
14     RAISE NOTICE 'Lowest salary employees:';
15     FOR R_EMP IN SELECT * FROM EMP ORDER BY sal ASC LIMIT 5 LOOP
16         RAISE NOTICE 'Employee Number: %, Employee Salary: %', R_EMP.empno, R_EMP.sal;
17     END LOOP;
18 END;
19 $procedure$;
20
```

Data Output Messages Notifications

```
CREATE PROCEDURE

Query returned successfully in 61 msec.
```

company/postgre... company/postgres@Lab1&2*

company/postgres@Lab1&2

No limit

Query Query History

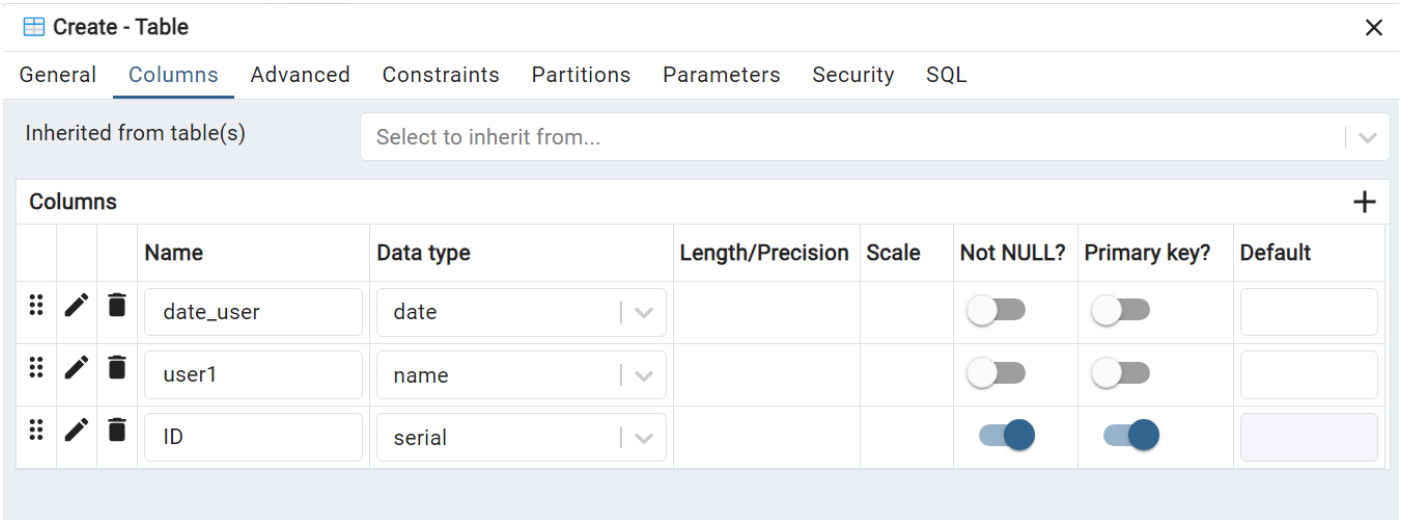
```
1  CALL public.exercise6()
```

Data Output Messages Notifications

```
NOTICE:  Highest salary employees:
NOTICE:  Employee Number: 7369, Employee Salary: 7000
NOTICE:  Employee Number: 7839, Employee Salary: 5000
NOTICE:  Employee Number: 7788, Employee Salary: 3000
NOTICE:  Employee Number: 7902, Employee Salary: 3000
NOTICE:  Employee Number: 7566, Employee Salary: 2975
NOTICE:  Lowest salary employees:
NOTICE:  Employee Number: 7900, Employee Salary: 950
NOTICE:  Employee Number: 7876, Employee Salary: 1100
NOTICE:  Employee Number: 7654, Employee Salary: 1250
NOTICE:  Employee Number: 7521, Employee Salary: 1250
NOTICE:  Employee Number: 7934, Employee Salary: 1300
CALL

Query returned successfully in 62 msec.
```

Exercice 7. Triggers

1. Create the triggers & 2. Function to analyze results



Query | Query History

```
1  CREATE OR REPLACE FUNCTION auto_insert() RETURNS TRIGGER AS $body$
2
3  BEGIN
4      raise notice 'user doing the motification : %', current_user;
5      raise notice 'date of motification : % ', current_date;
6
7      insert into monitor(user1, date_user) VALUES (current_user, current_date);
8  RETURN NULL;
9  END
10 $body$
11 LANGUAGE plpgsql;
12
13 CREATE OR REPLACE TRIGGER MYTRIGGER
14 AFTER INSERT OR UPDATE OR DELETE ON EMP
15 EXECUTE PROCEDURE auto_insert();
16
17 UPDATE EMP
18 SET EFIRST = 'JEAN' WHERE EMPNO = 7369;
19
```

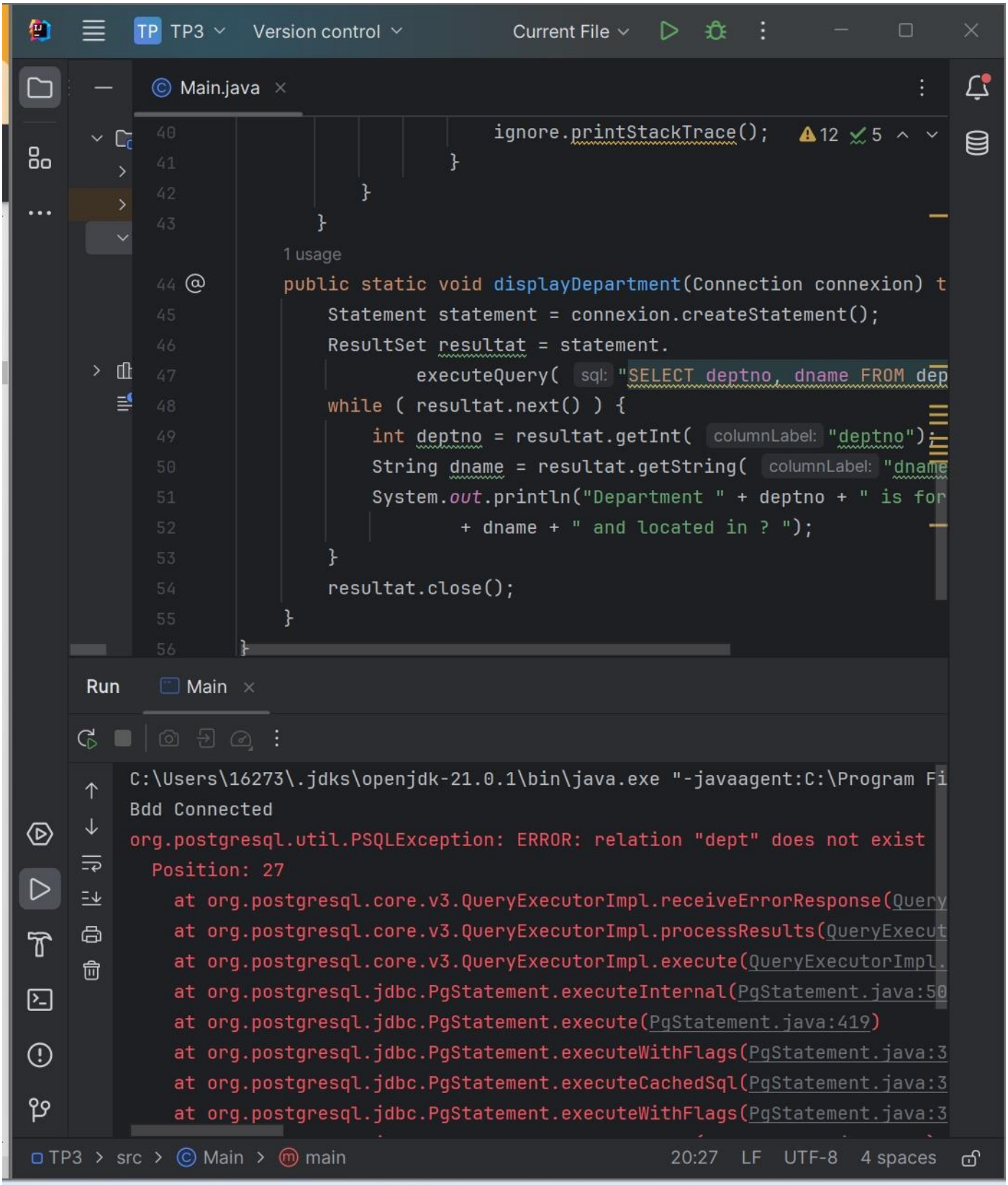
Data Output | Messages | Notifications

```
NOTICE:  user doing the motification : postgres
NOTICE:  date of motification : 2024-04-01
UPDATE 1

Query returned successfully in 75 msec.
```

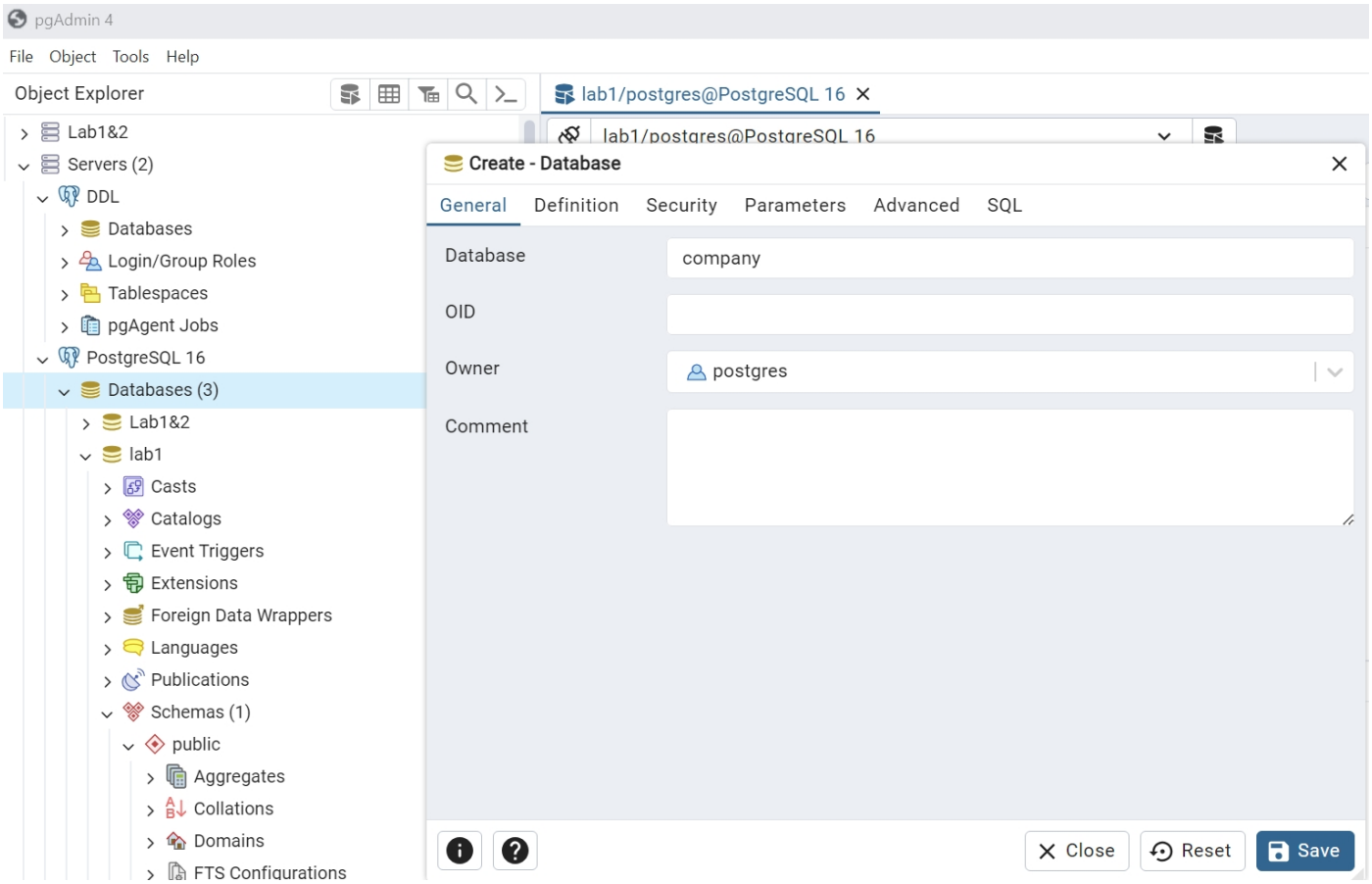
Lab 3

Part I: A Graphic of Table: communication with database

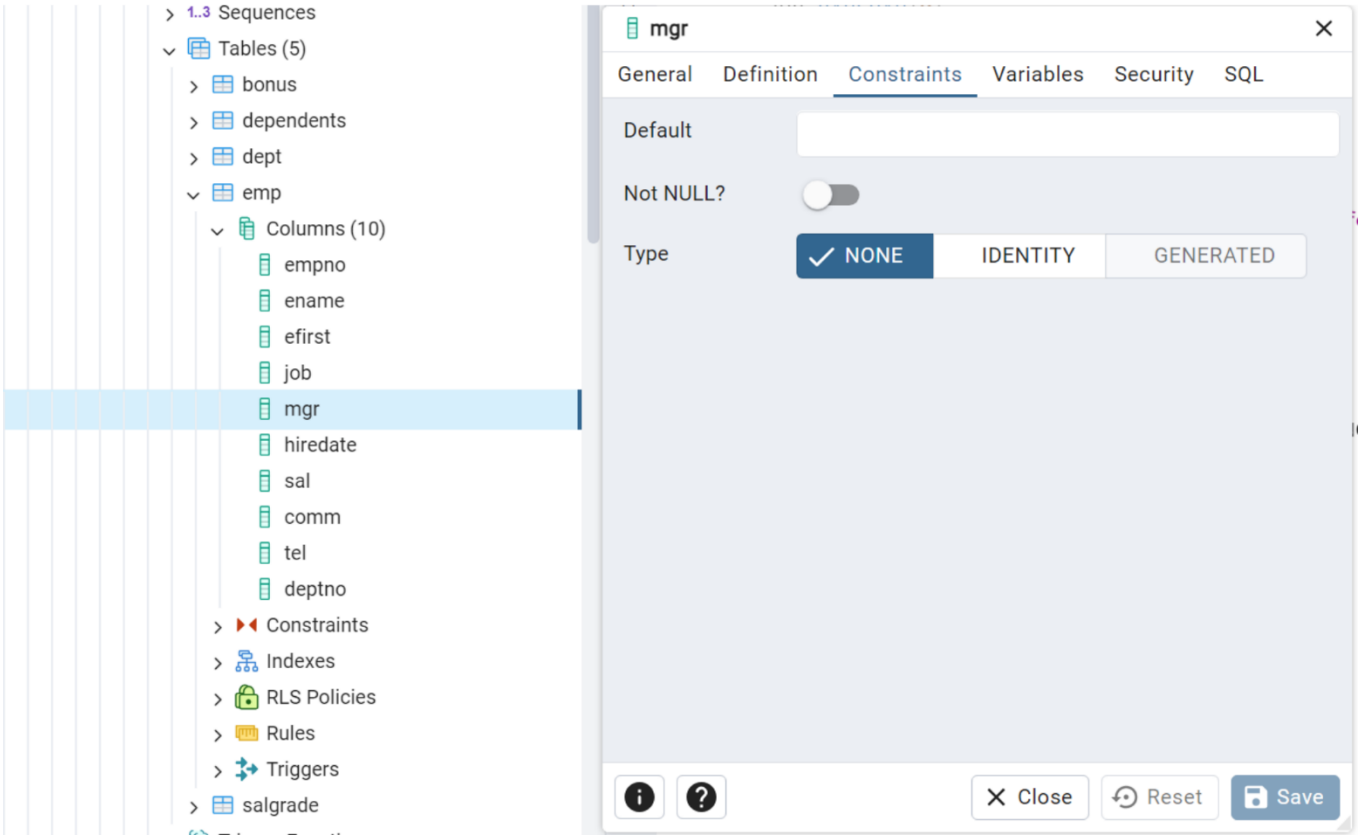
1. Load Driver - 2. Connect to bdd - 3. Make a request.



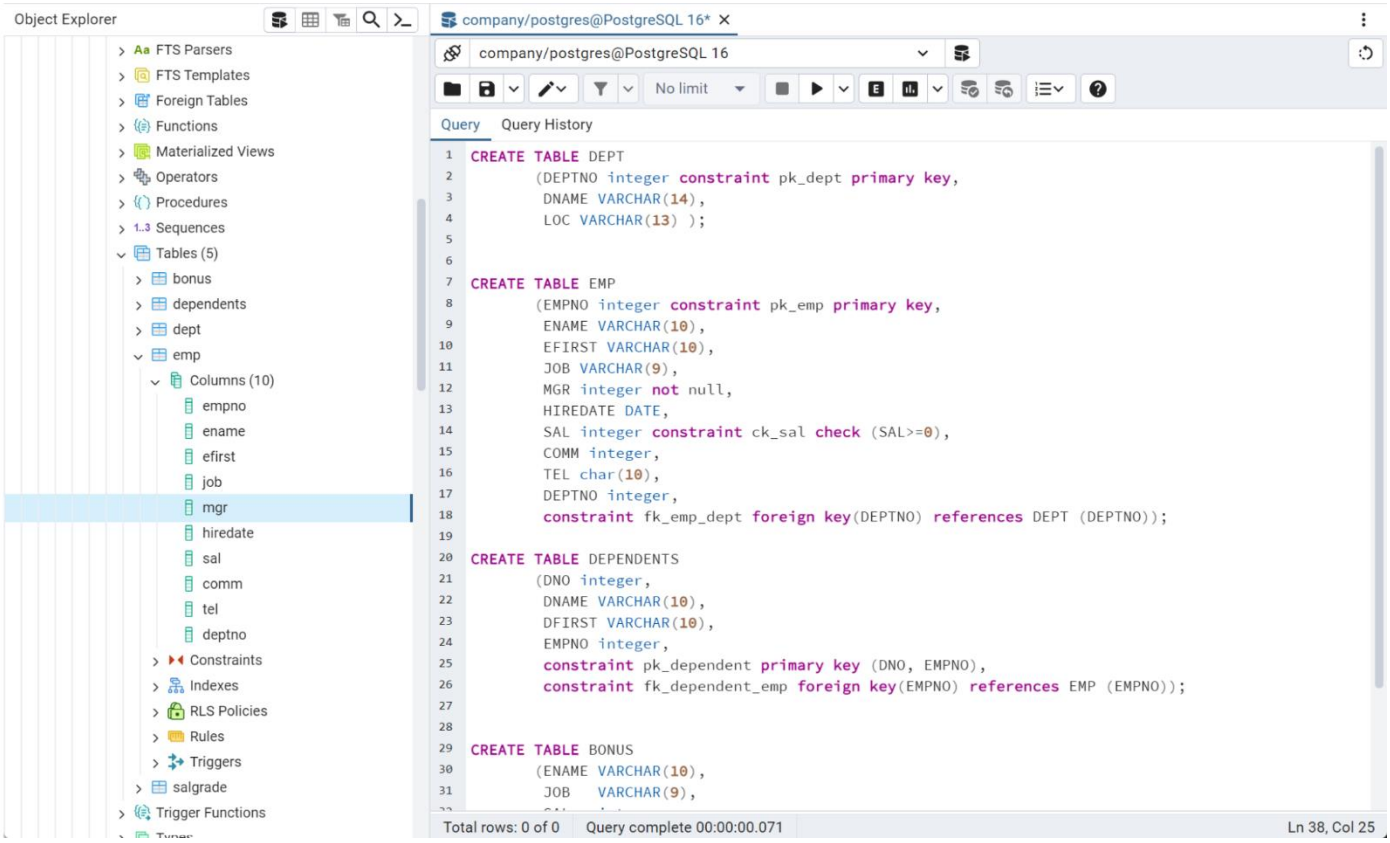
- In order to solve: Create new Database -> "company".


pgAdmin 4
File Object Tools Help
Object Explorer
lab1/postgres@PostgreSQL 16
Lab1&2
Servers (2)
DDL
Databases
Login/Group Roles
Tablespaces
pgAgent Jobs
PostgreSQL 16
Databases (3)
Lab1&2
lab1
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas (1)
public
Aggregates
Collations
Domains
FTS Configurations
Create - Database
General
Definition
Security
Parameters
Advanced
SQL
Database
company
OID
Owner
postgres
Comment
Close
Reset
Save

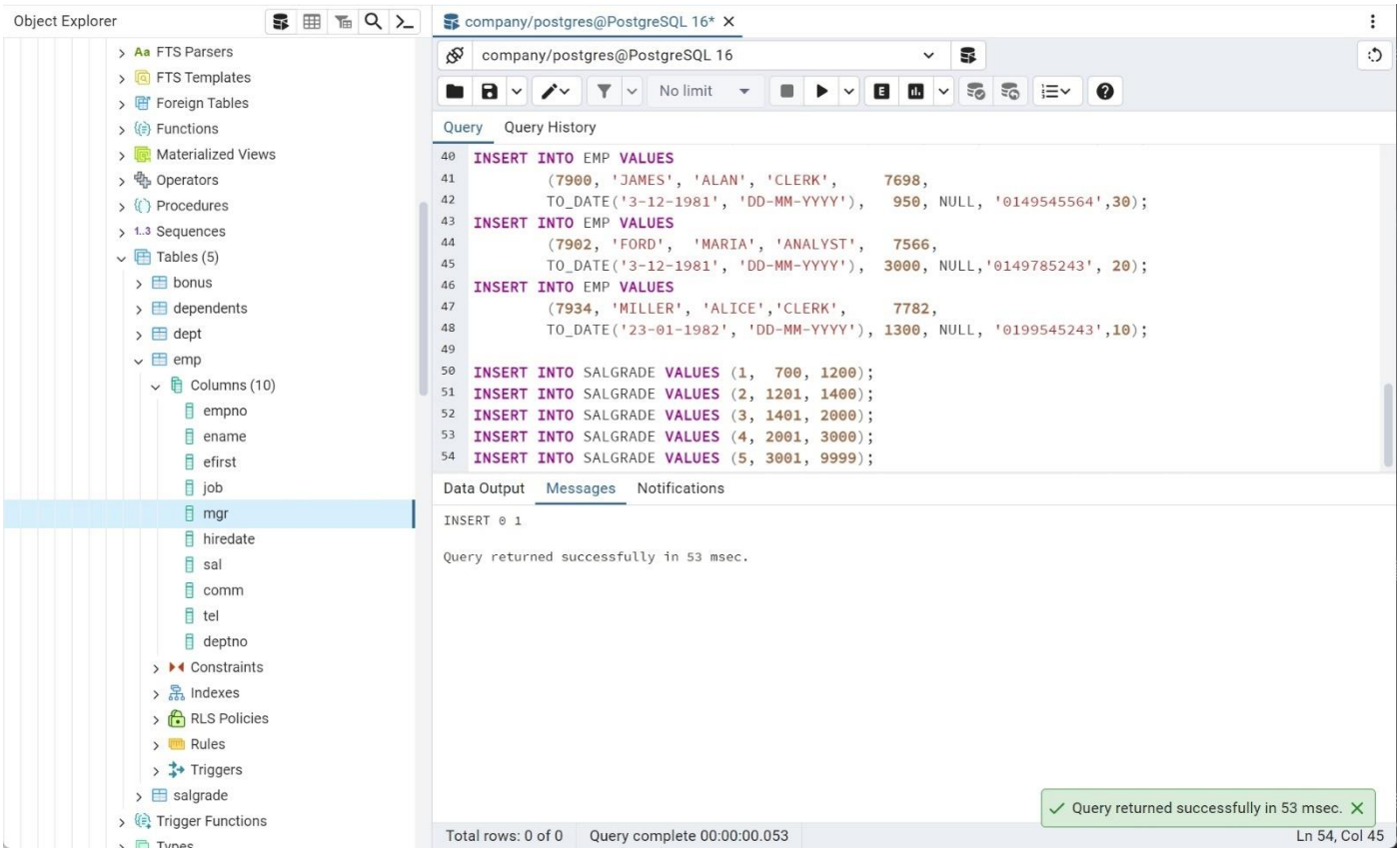
- Close"NULL"


Sequences
Tables (5)
bonus
dependents
dept
emp
Columns (10)
empno
ename
efirst
job
mgr
hiredate
sal
comm
tel
deptno
Constraints
Indexes
RLS Policies
Rules
Triggers
salgrade
mgr
General
Definition
Constraints
Variables
Security
SQL
Default
Not NULL?
Type
NONE
IDENTITY
GENERATED
Close
Reset
Save

- Add table



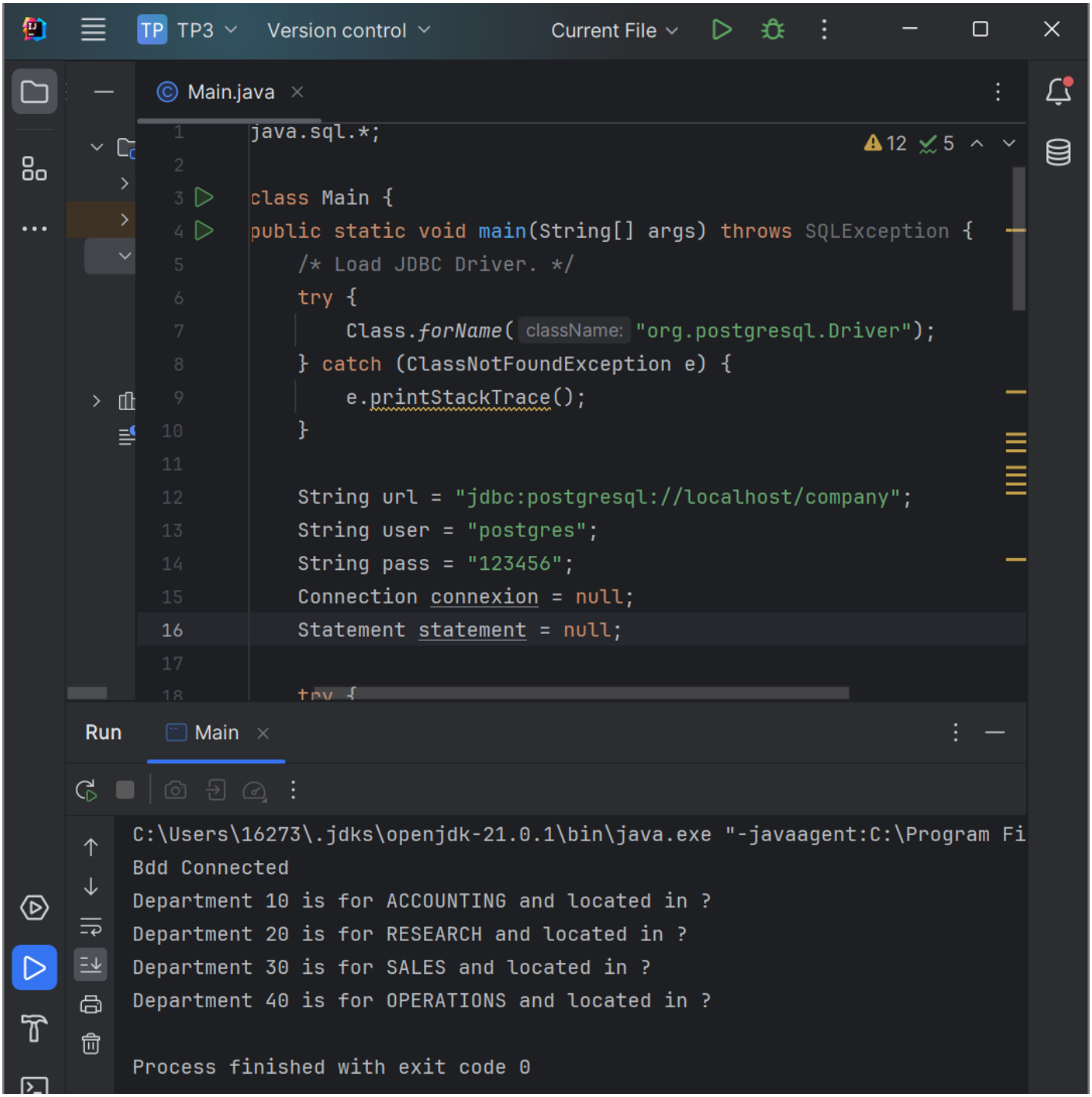
- Add Info



- Modifier the JAVA code to link to “company”

```
String url = "jdbc:postgresql://localhost/company";
String user = "postgres";
String pass = "123456";
```

- Re-run the code



Exercise 1: Modify the query above to add the location of the department.

```java
1 usage
public static void displayDepartment(Connection connexion) throws SQLException {
    Statement statement = connexion.createStatement();
    ResultSet resultat = statement.
            executeQuery( sql: "SELECT deptno, dname, loc FROM dept" );
    while ( resultat.next() ) {
        int deptno = resultat.getInt( columnLabel: "deptno");
        String dname = resultat.getString( columnLabel: "dname" );
        String location = resultat.getString( columnLabel: "loc");
        System.out.println("Department " + deptno + " is for "
                + dname + " and located in " + location);
    }
    resultat.close();
}
```

```
Run  Main
C:\Users\16273\.jdks\openjdk-21.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\Intelli
Bdd Connected
Department 10 is for ACCOUNTING and located in NEW YORK
Department 20 is for RESEARCH and located in DALLAS
Department 30 is for SALES and located in CHICAGO
Department 40 is for OPERATIONS and located in BOSTON

Process finished with exit code 0
```

Exercice 2: Move Department.

- Add the move-function

```java
1 usage
public static void moveDepartment(Connection connection,int Emp, int Deptno) throws SQLException{
    String command = "UPDATE EMP set DEPTNO = ? WHERE EMPNO = ?";
    try(PreparedStatement updateEmp = connection.prepareStatement(command)){
        updateEmp.setInt( parameterIndex: 1,Deptno);
        updateEmp.setInt( parameterIndex: 2,Emp);
        updateEmp.execute();
        System.out.println("\nEmployee No." + Emp + " is in the department " + Deptno);
    }
    catch (Exception exception){
        exception.printStackTrace();
    }
}
```

- Call the function in the Main and run it

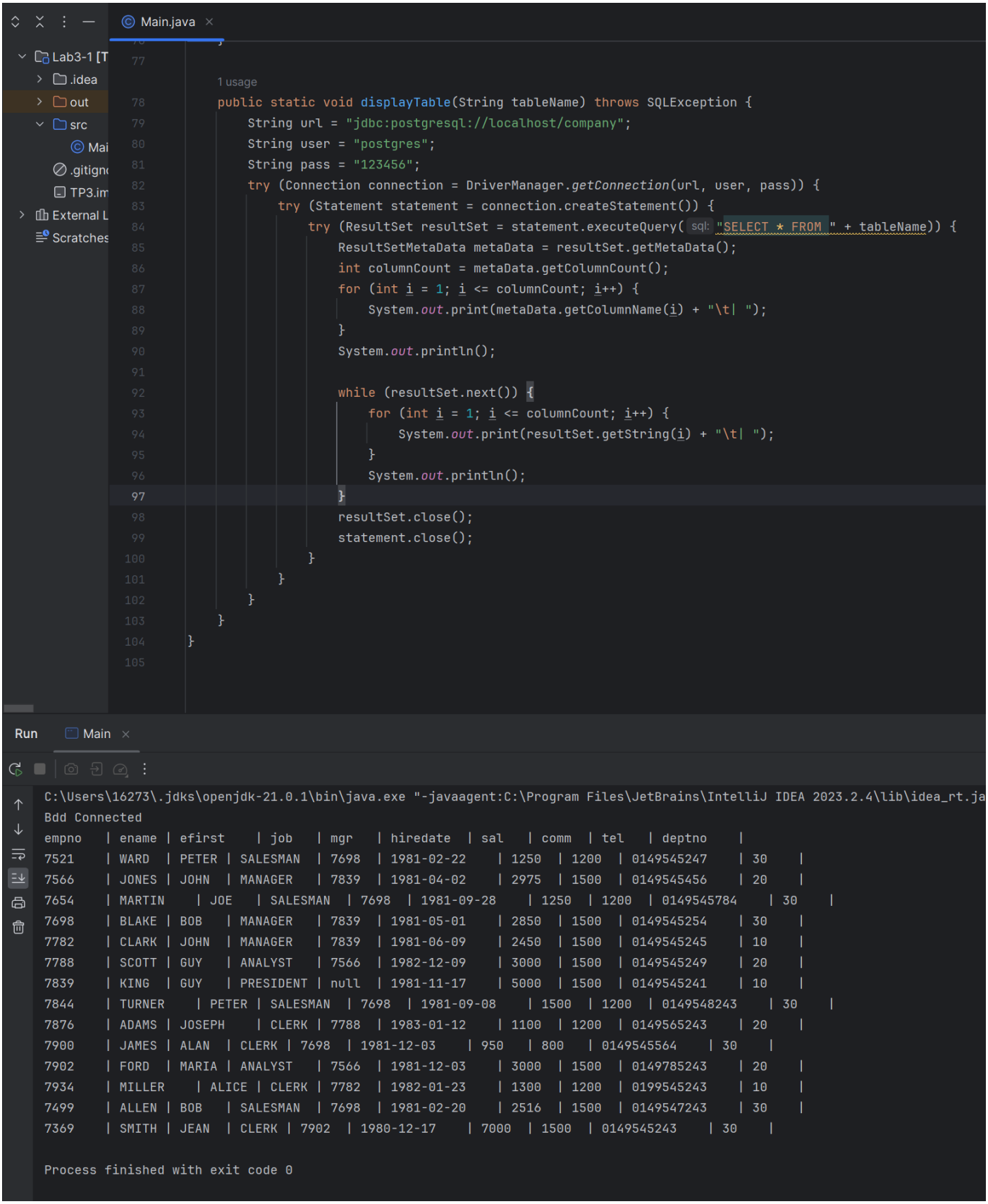
Main.java

```
1 usage
public static void moveDepartment(Connection connection,int Emp, int Deptno) throws SQLException{
    String command = "UPDATE EMP set DEPTNO = ? WHERE EMPNO = ?";
    try(PreparedStatement updateEmp = connection.prepareStatement(command)){
        updateEmp.setInt( parameterIndex: 1,Deptno);
        updateEmp.setInt( parameterIndex: 2,Emp);
        updateEmp.execute();
        System.out.println("\nEmployee No." + Emp + " is in the department " + Deptno);
    }
    catch (Exception exception){
        exception.printStackTrace();
    }
}
}
```

Main

```
C:\Users\16273\.jdks\openjdk-21.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.2.4\lib\i
Bdd Connected

Employee No.7369 is in the department 30

Process finished with exit code 0
```

Exercice 3: Generic display of Tables:



Exercice 4: Security:

- What is the Security flow to the method above?

1) `Statement` objects may lead to SQL injection. If user input is spliced directly into an SQL statement, can modify the behaviour of the SQL statement by constructing special input, for example, to retrieve, modify, or delete data,

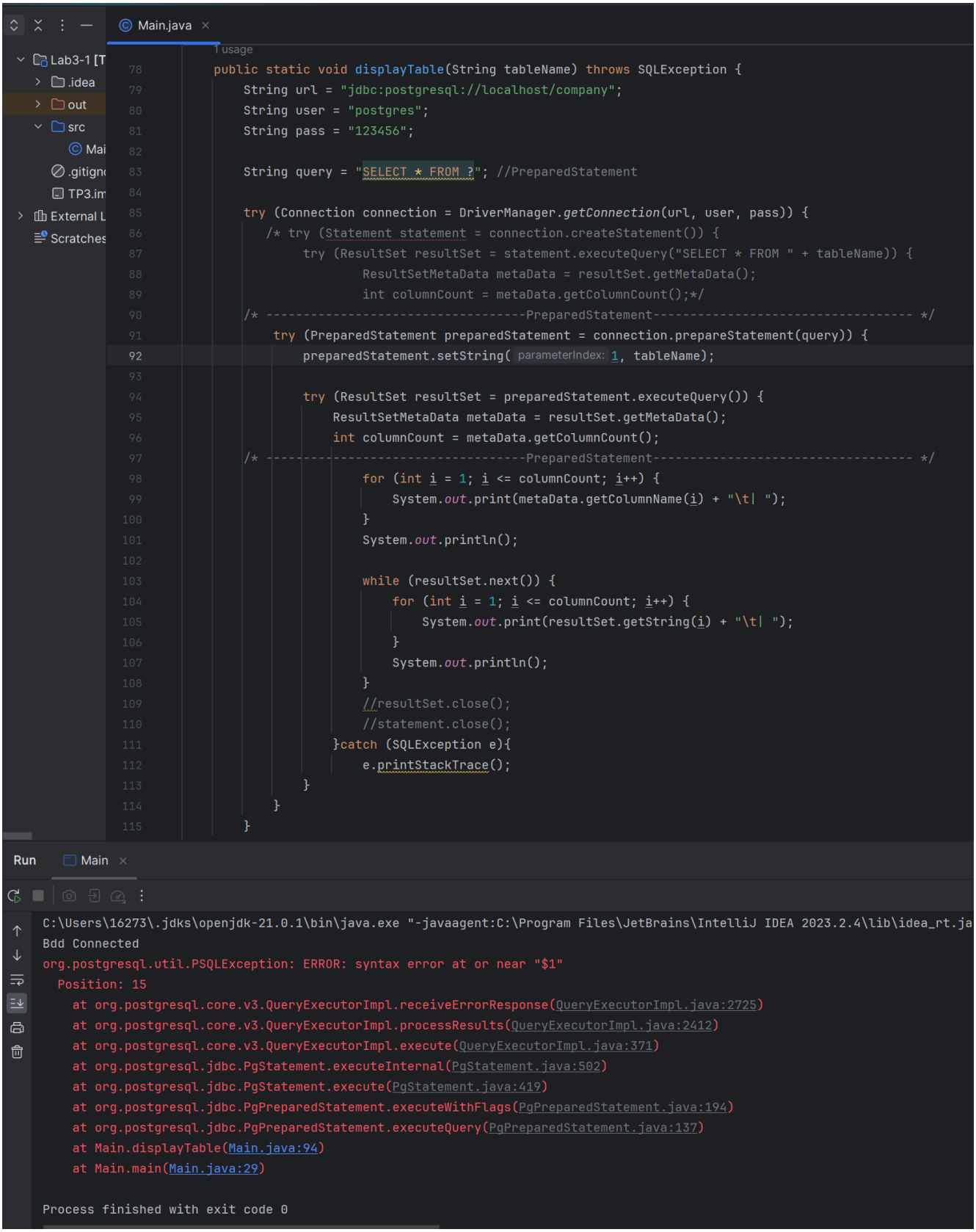
- What are others Cons from this basic native method?
  1) SQL statements are usually hard coded in the code: making modification difficult, and the readability and maintainability of the code are poor.
  2) Developers must ensure that all database-related resources are properly closed after use, otherwise problems such as memory leaks may occur.
  3) SQL injection and potential others wrong arguments passed in the query.

Exercice 5:Modify moveDepartement() to use PrepareStatemen

```
65    public static void moveDepartment(Connection connection,int Emp, int Deptno) throws SQLException{
66        String command = "UPDATE EMP set DEPTNO = ? WHERE EMPNO = ?";
67        try(PreparedStatement updateEmp = connection.prepareStatement(command)){
68            updateEmp.setInt( parameterIndex: 1,Deptno);
69            updateEmp.setInt( parameterIndex: 2,Emp);
70            updateEmp.execute();
71            System.out.println("\nEmployee No. " + Emp + " has been moved to department " + Deptno);
72        }
73        catch (Exception exception){
74            exception.printStackTrace();
75        }
76    }
```

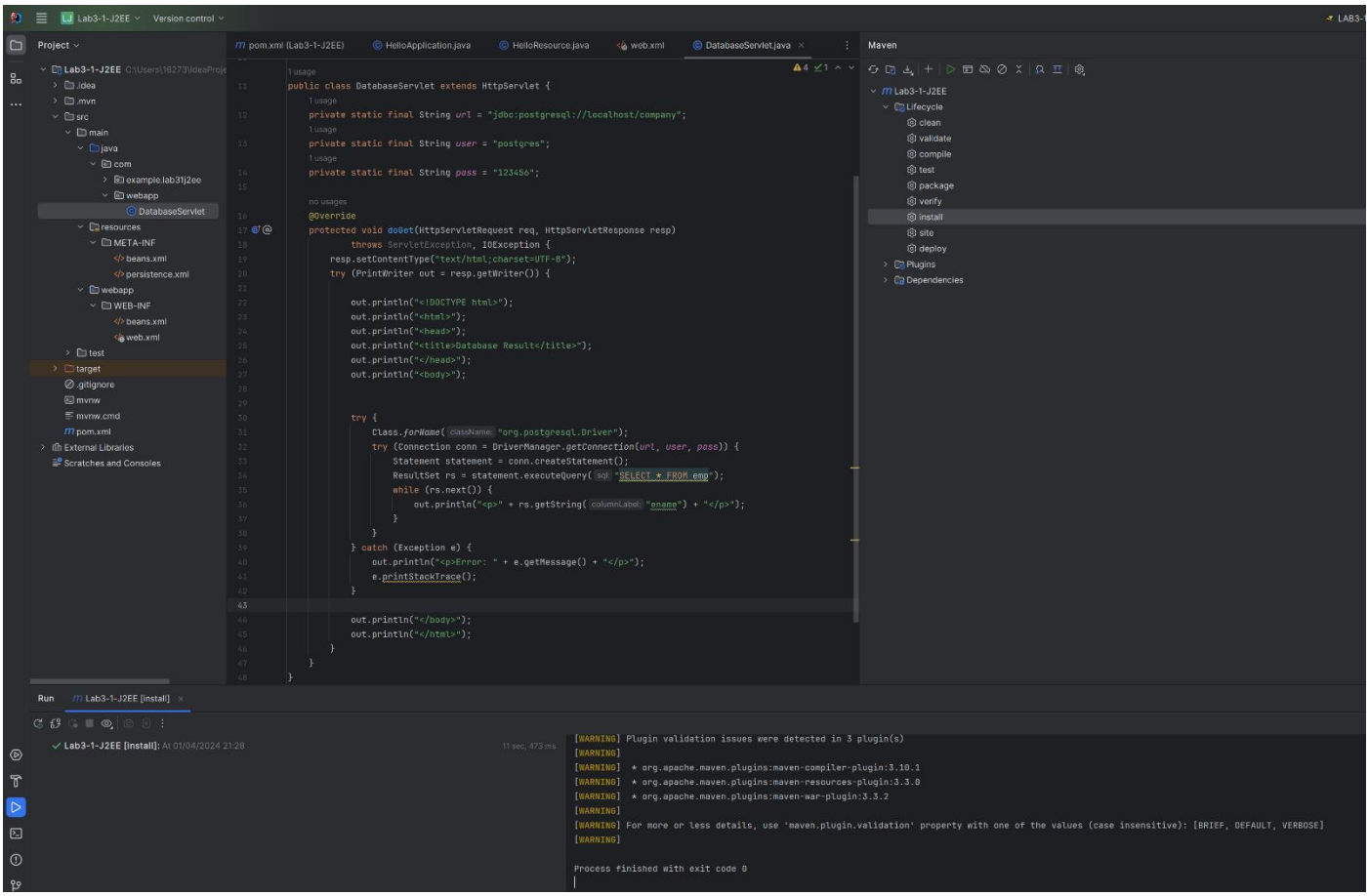
```
Main
C:\Users\16273\.jdks\openjdk-21.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.2.4\lib\idea_rt.jar=64003:C:\Program
Bdd Connected

Employee No. 7369 has been moved to department 30

Process finished with exit code 0
```

Exercice 6: Try with displayTable



JDBC does not allow table names or column names to be passed as parameters to PreparedStatement, preventing SQL injection and ensuring that only values can be dynamically inserted into SQL statements.

Exercice 7: With J2EE

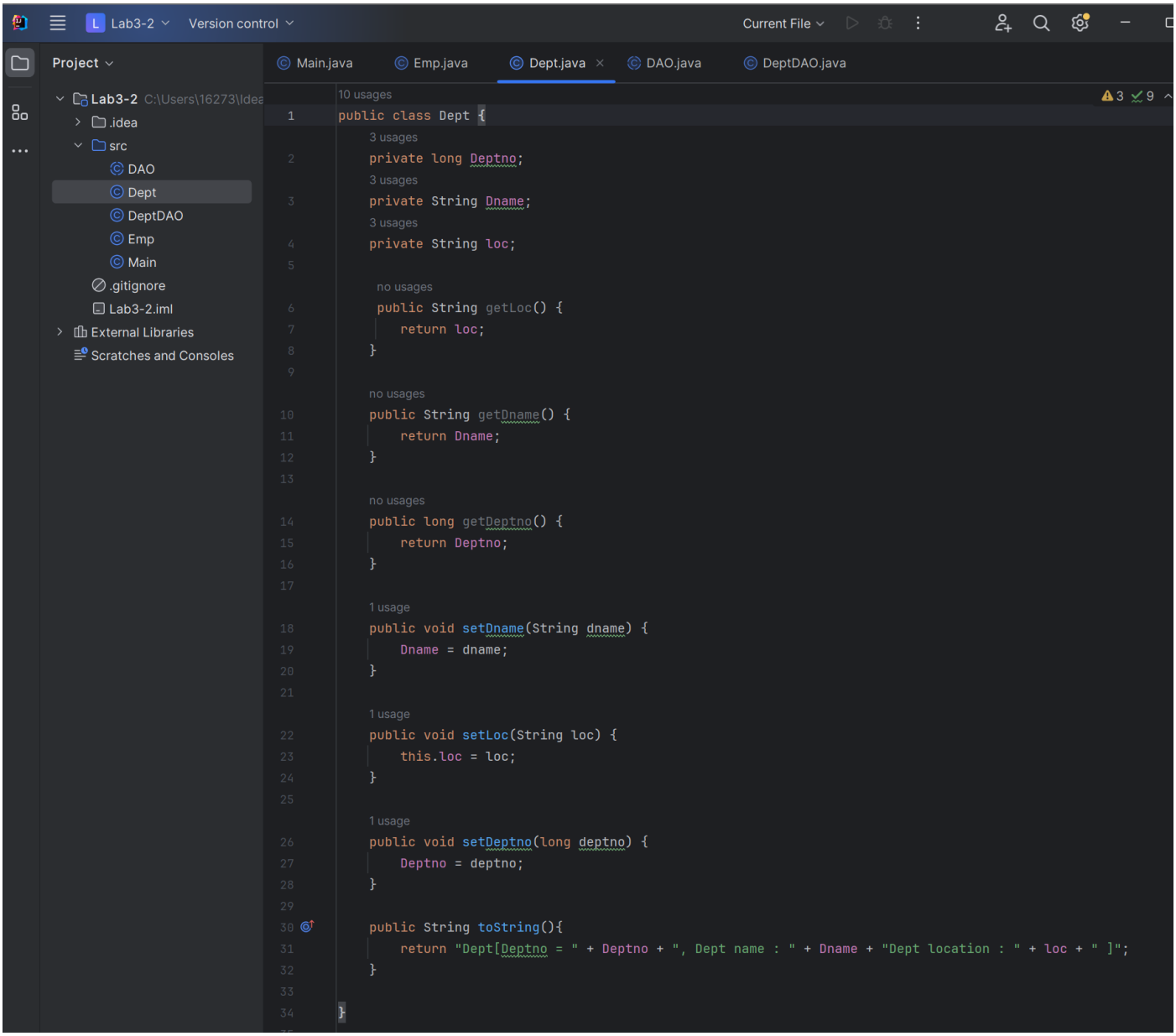


Part II : DAO.

Exercice 8: What are cons of JDBC as used above?

- Boilerplate Code: JDBC requires verbose code for establishing connections, error handling, and resource cleanup.
- Error-Prone: Manually managing resources like connections, statements, and result sets increases the risk of resource leaks.
- Lack of Abstraction: JDBC operates at a low level, resulting in more effort for even simple queries and data manipulations.
- Maintenance Difficulty: Changes in the database schema often require updates in multiple parts of JDBC code, making maintenance challenging.

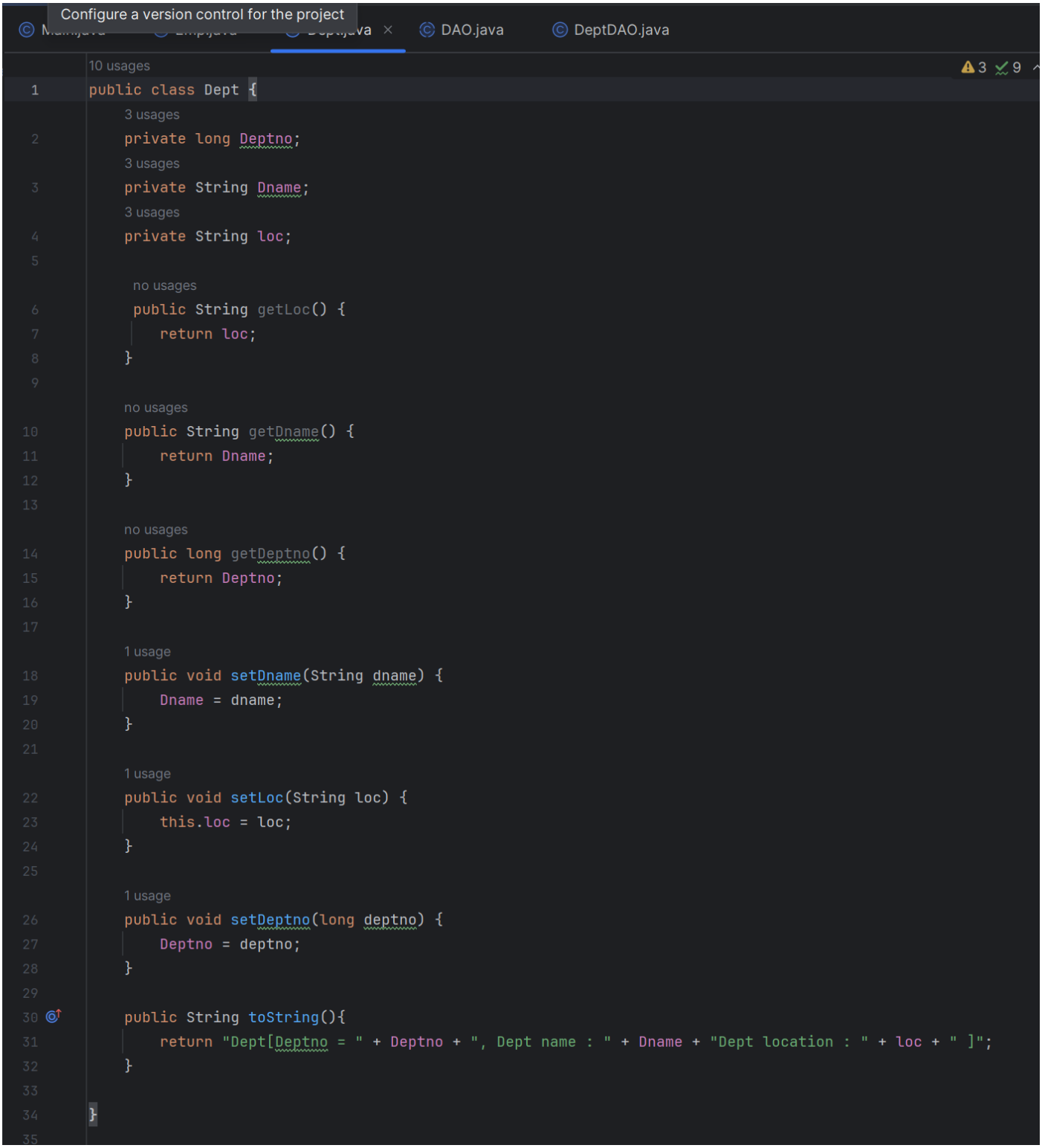
Exercice 9 : Write the Bean for Department

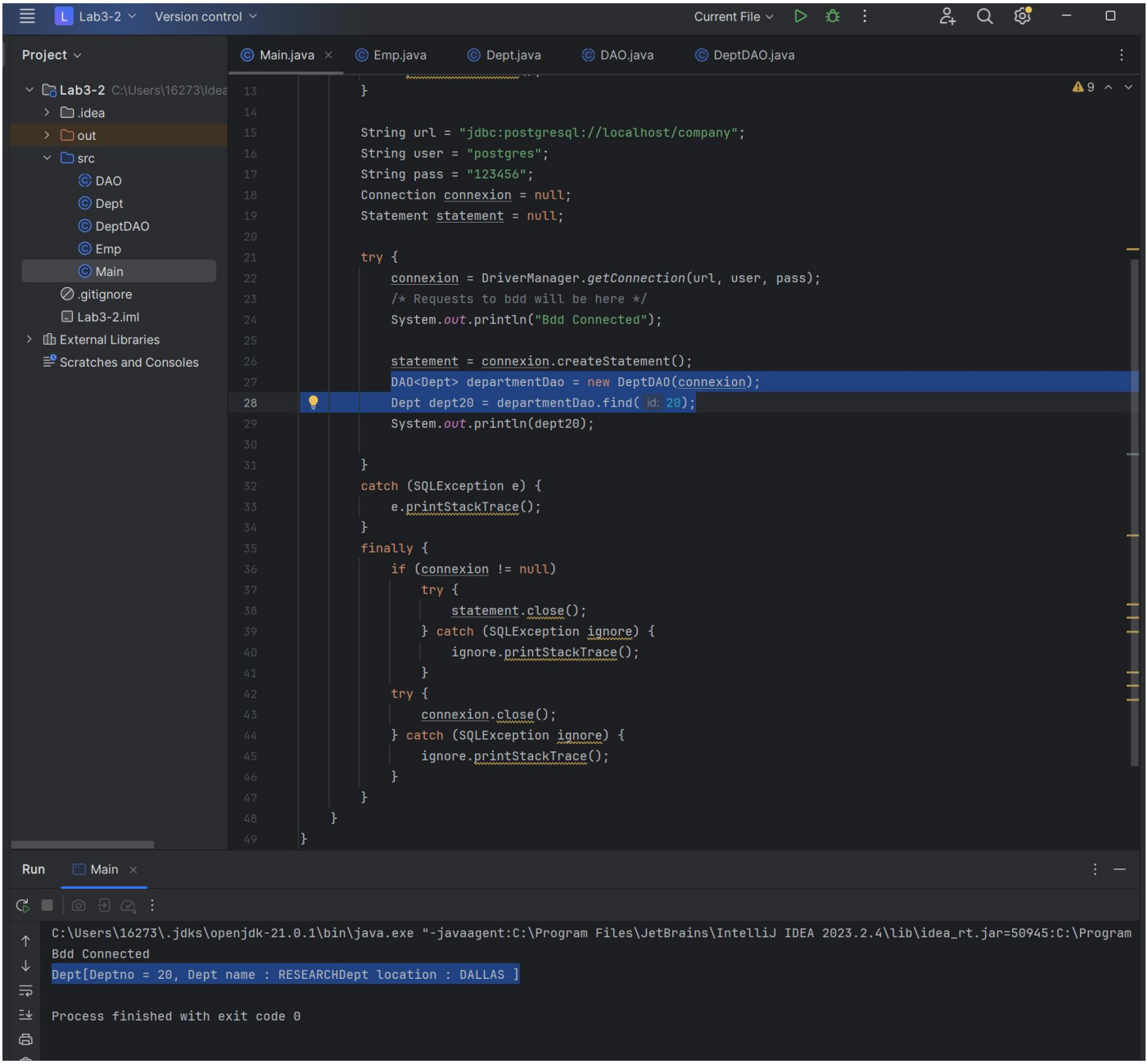


Exercice 10 : Write Implementation for method find for departement DAO

Main.java | Emp.java | Dept.java | DAO.java | DeptDAO.java

```java
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

1 usage
public class DeptDAO extends DAO<Dept> {
1 usage
public DeptDAO(Connection connexion){ super(connexion);}
    1 usage
    public Dept find(int id) {
        Dept dept = new Dept();
        try {
            PreparedStatement finddept = connect.prepareStatement( sql: "SELECT * FROM DEPT where DEPTNO = ?");
            finddept.setInt( parameterIndex: 1, id);
            ResultSet result = finddept.executeQuery();
            while (result.next()){
                dept.setDeptno(result.getInt( columnLabel: "Deptno"));
                dept.setDname(result.getString( columnLabel: "Dname"));
                dept.setLoc(result.getString( columnLabel: "loc"));
            }
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
        return dept;
    }

    no usages
    @Override
    public boolean create(Dept object) {
        return false;
    }

    no usages
    @Override
    public boolean update(Dept object) {
        return false;
    }

    no usages
    @Override
    public boolean delete(Dept object) {
        return false;
    }
}
```

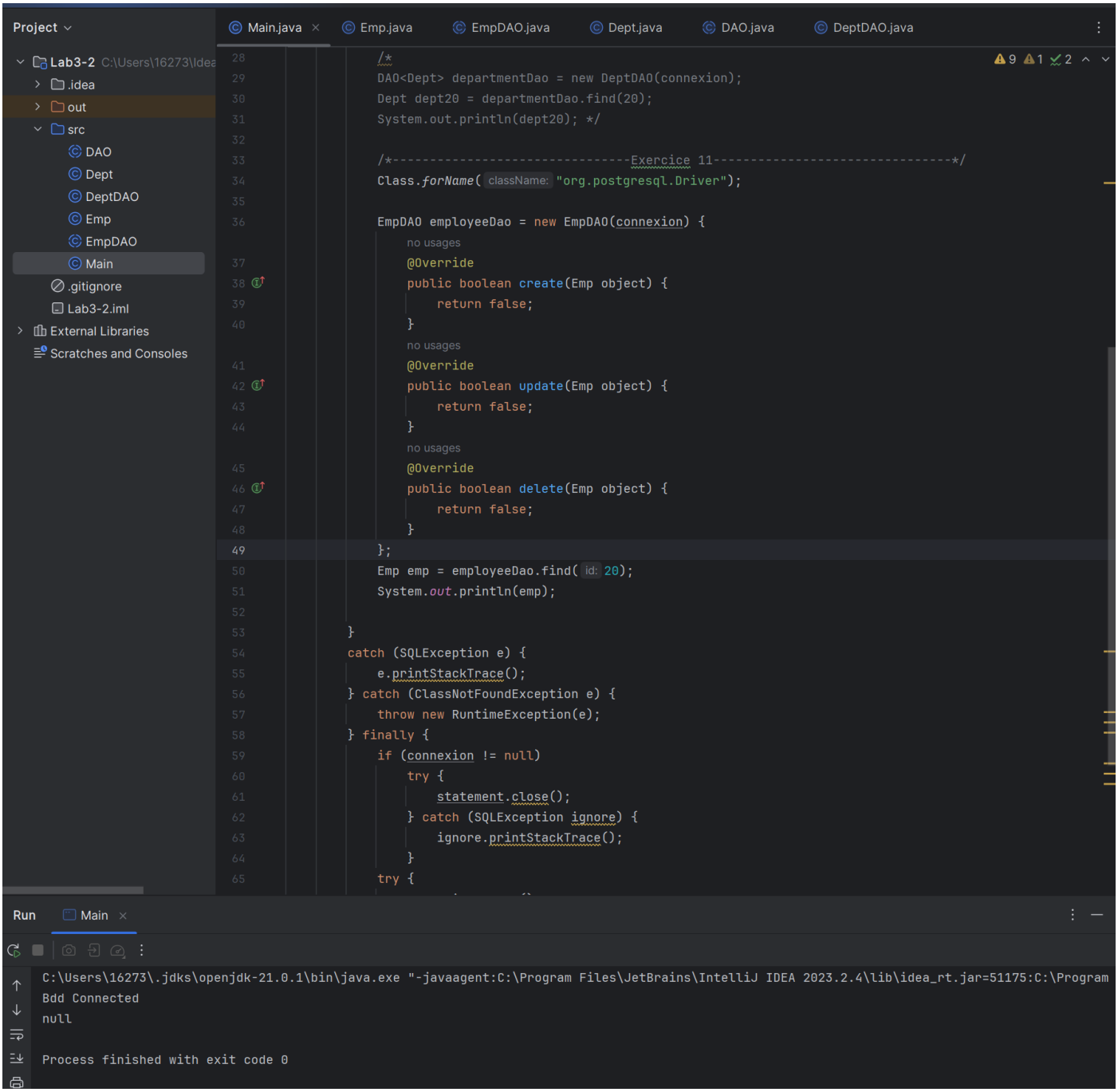

Configure a version control for the project
DAO.java
DeptDAO.java
10 usages
3 9
public class Dept {
3 usages
private long Deptno;
3 usages
private String Dname;
3 usages
private String loc;
no usages
public String getLoc() {
return loc;
}
no usages
public String getDname() {
return Dname;
}
no usages
public long getDeptno() {
return Deptno;
}
1 usage
public void setDname(String dname) {
Dname = dname;
}
1 usage
public void setLoc(String loc) {
this.loc = loc;
}
1 usage
public void setDeptno(long deptno) {
Deptno = deptno;
}
public String toString(){
return "Dept[Deptno = " + Deptno + ", Dept name : " + Dname + "Dept location : " + loc + "]";
}
}


Lab3-2
Version control
Current File
Project
Main.java
Emp.java
Dept.java
DAO.java
DeptDAO.java
Lab3-2 C:\Users\16273\Idea
.idea
out
src
DAO
Dept
DeptDAO
Emp
Main
.gitignore
Lab3-2.iml
External Libraries
Scratches and Consoles
}
String url = "jdbc:postgresql://localhost/company";
String user = "postgres";
String pass = "123456";
Connection connexion = null;
Statement statement = null;
try {
connexion = DriverManager.getConnection(url, user, pass);
/* Requests to bdd will be here */
System.out.println("Bdd Connected");
statement = connexion.createStatement();
DAO<Dept> departmentDao = new DeptDAO(connexion);
Dept dept20 = departmentDao.find(id: 20);
System.out.println(dept20);
}
catch (SQLException e) {
e.printStackTrace();
}
finally {
if (connexion != null)
try {
statement.close();
} catch (SQLException ignore) {
ignore.printStackTrace();
}
try {
connexion.close();
} catch (SQLException ignore) {
ignore.printStackTrace();
}
}
}
}
Run
Main
C:\Users\16273\.jdks\openjdk-21.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.2.4\lib\idea_rt.jar=50945:C:\Program
Bdd Connected
Dept[Deptno = 20, Dept name : RESEARCHDept location : DALLAS]
Process finished with exit code 0

Exercice 11: Write Implementation for method find for employee DAO.

Main.java | Emp.java | EmpDAO.java | Dept.java | DAO.java | DeptDAO.java

```java
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

no usages
public abstract class EmpDAO extends DAO<Emp> {
    no usages
    public EmpDAO(Connection connexion) {
        super(connexion);
    }

    2 usages
    @Override
    public Emp find(int id) {
        Emp emp = null;
        try {
            PreparedStatement findEmp = connect.prepareStatement(
                    sql: "SELECT * FROM EMP WHERE EMPNO = ?"
            );
            findEmp.setInt( parameterIndex: 1, id);
            ResultSet result = findEmp.executeQuery();
            if (result.next()) {
                emp = new Emp();
                emp.setEmpNo((long) result.getInt( columnLabel: "empNo"));
                emp.setEname(result.getString( columnLabel: "ename"));

                int mgrId = result.getInt( columnLabel: "mgr");
                if (mgrId != 0) {
                    emp.setMgr(find(mgrId)); // 递归调用，设置经理
                }
            }
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
        return emp;
    }
}
```


Project
Lab3-2 C:\Users\16273\Idea
.idea
out
src
DAO
Dept
DeptDAO
Emp
EmpDAO
Main
.gitignore
Lab3-2.iml
External Libraries
Scratches and Consoles
Main.java
Emp.java
EmpDAO.java
Dept.java
DAO.java
DeptDAO.java
/*
DAO<Dept> departmentDao = new DeptDAO(connexion);
Dept dept20 = departmentDao.find(20);
System.out.println(dept20); */
/*------------------------------Exercice 11------------------------------*/
Class.forName(className: "org.postgresql.Driver");
EmpDAO employeeDao = new EmpDAO(connexion) {
no usages
@Override
public boolean create(Emp object) {
return false;
}
no usages
@Override
public boolean update(Emp object) {
return false;
}
no usages
@Override
public boolean delete(Emp object) {
return false;
}
};
Emp emp = employeeDao.find(id: 20);
System.out.println(emp);
}
catch (SQLException e) {
e.printStackTrace();
} catch (ClassNotFoundException e) {
throw new RuntimeException(e);
} finally {
if (connexion != null)
try {
statement.close();
} catch (SQLException ignore) {
ignore.printStackTrace();
}
try {
Run
Main
C:\Users\16273\.jdks\openjdk-21.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.2.4\lib\idea_rt.jar=51175:C:\Program
Bdd Connected
null
Process finished with exit code 0

Exercice 12 : Create DAOFactory and use it in previous example

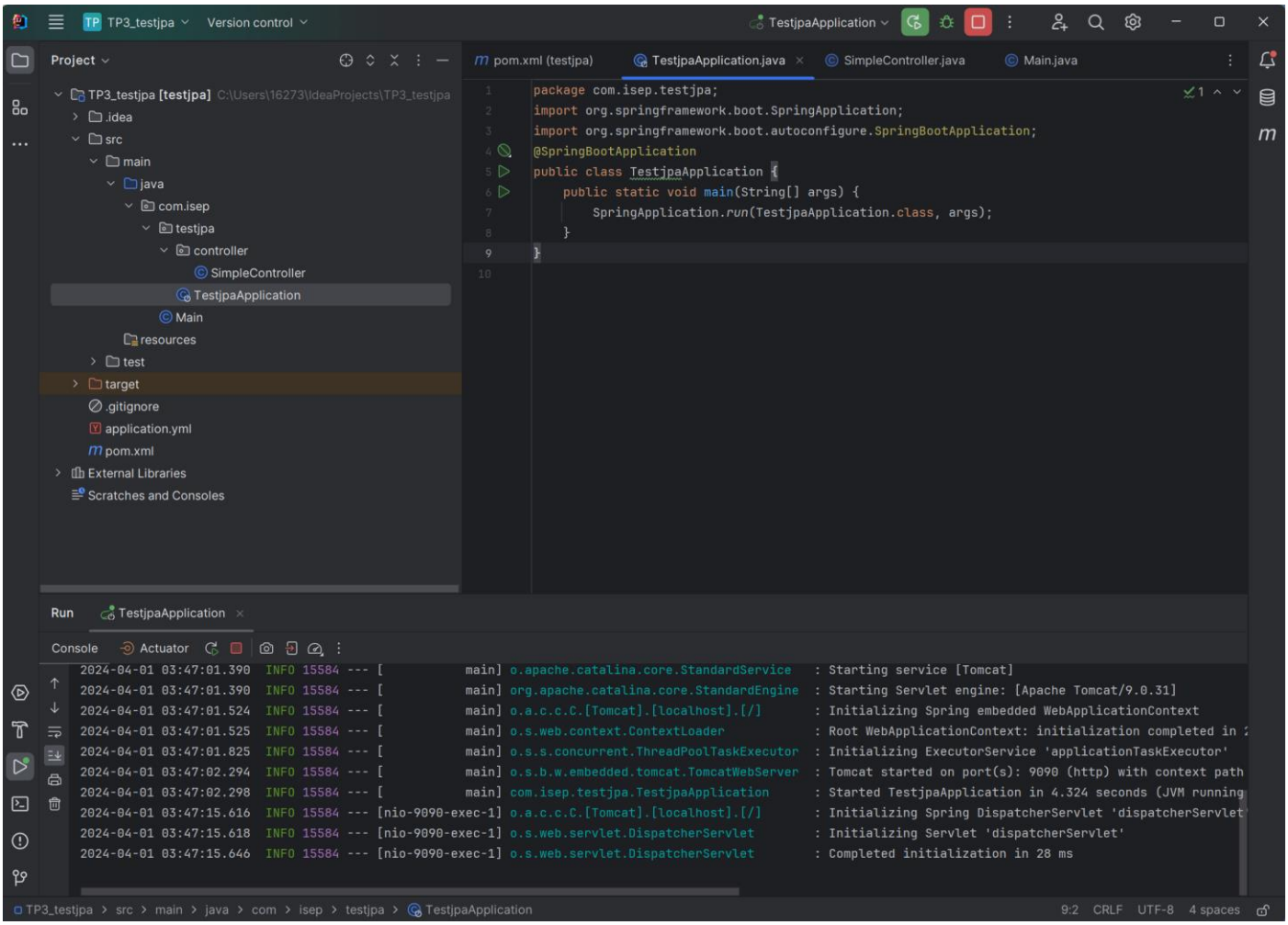
Main.java | DAOFactory.java | Emp.java | EmpDAO.java | Dept.java | DAO.java

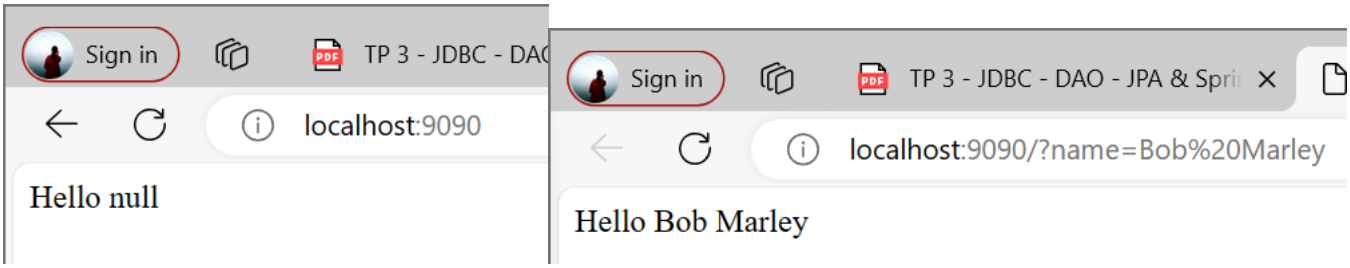
```java
import java.sql.Connection;

no usages
public class DAOFactory {
    5 usages
    private Connection connection;

    no usages
    public DAOFactory(Connection connection) {
        this.connection = connection;
    }

    no usages
    public DeptDAO getDeptDAO() {
        return new DeptDAO(connection);
    }

    no usages
    public EmpDAO getEmpDAO() {
        return new EmpDAO(connection) {
            no usages
            @Override
            public boolean create(Emp object) {
                return false;
            }

            no usages
            @Override
            public boolean update(Emp object) {
                return false;
            }

            no usages
            @Override
            public boolean delete(Emp object) {
                return false;
            }
        };
    }

    no usages
    public DeptDAO getdependentsDAO(){
        return new DeptDAO(connection);
    }

    no usages
    public DAO getBonusDAO(){
        return new DeptDAO(connection);
    }
}
```

```
72              */
73          /*--------------------------------Exercice 12--------------------------------*/
74              DAOFactory daoFactory = new DAOFactory(connexion);
75              DAO<Dept> departmentDao = daoFactory.getDeptDAO();
76              Dept dept20 = departmentDao.find( id: 20);
77              System.out.println(dept20);
78
79              DAO<Emp> employeeDao = daoFactory.getEmpDAO();
80              Emp emp = employeeDao.find( id: 20);
81              System.out.println(emp);
82          }catch (SQLException e) {
83              e.printStackTrace();
84              System.out.println("Connection failure.");
85          }
86      }
87  }
88
```

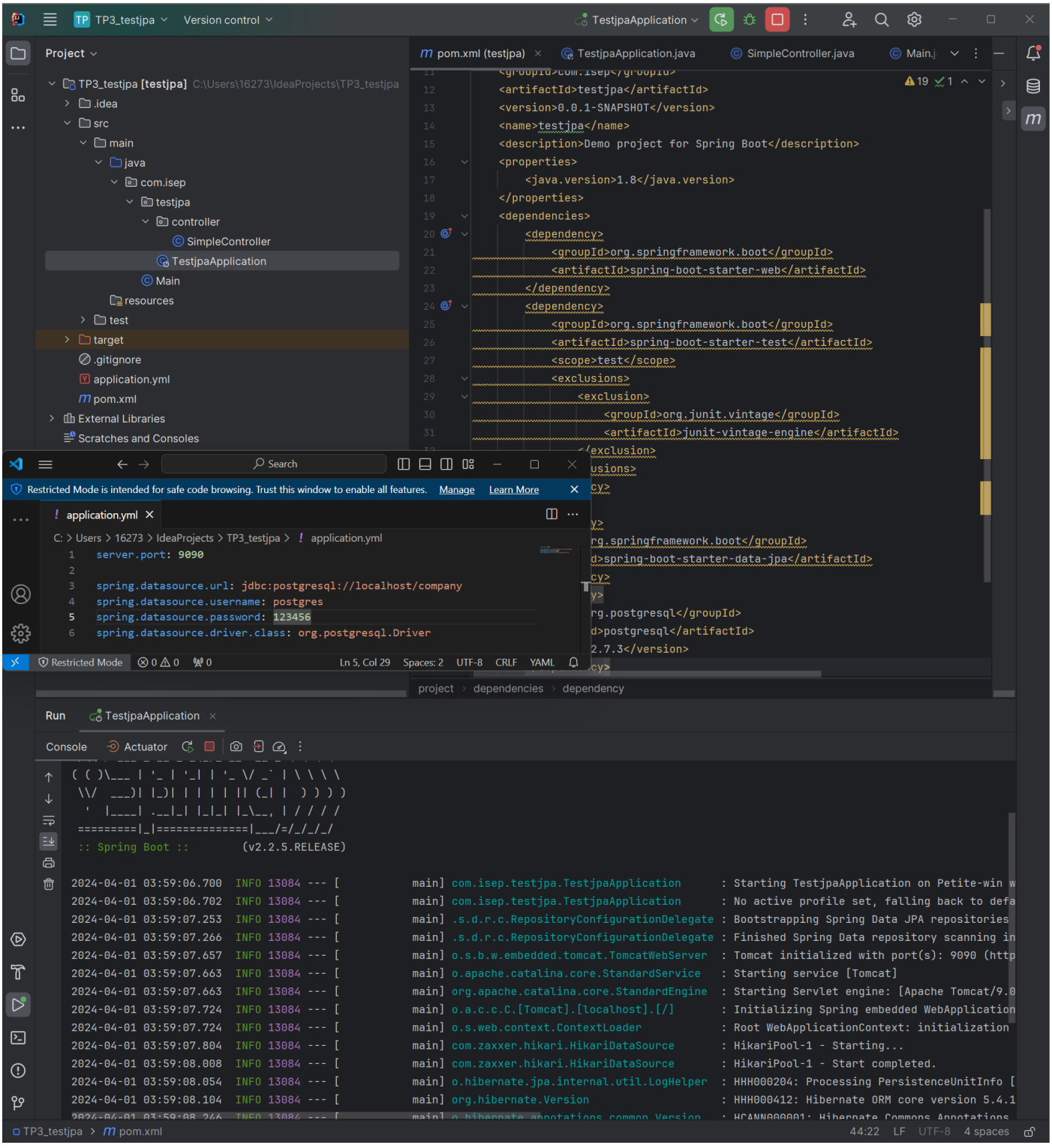
```
un  Main ×

C:\Users\16273\.jdks\openjdk-21.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.2.4\lib\idea_rt.jar=51433:C:\Program
Bdd Connected
Dept[Deptno = 20, Dept name : RESEARCHDept location : DALLAS ]
null

Process finished with exit code 0
```

Part III : Spring Boot & JPA.

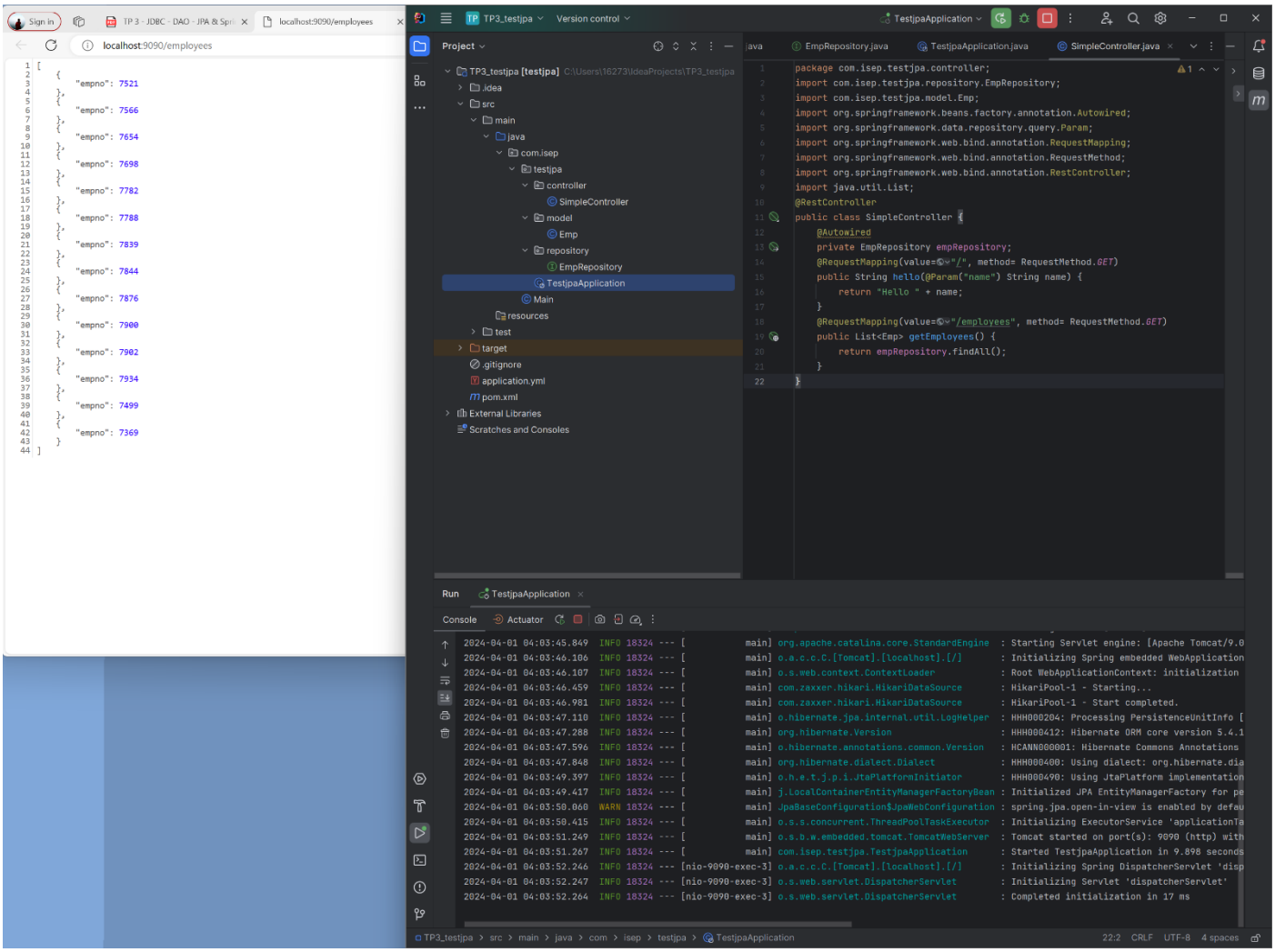


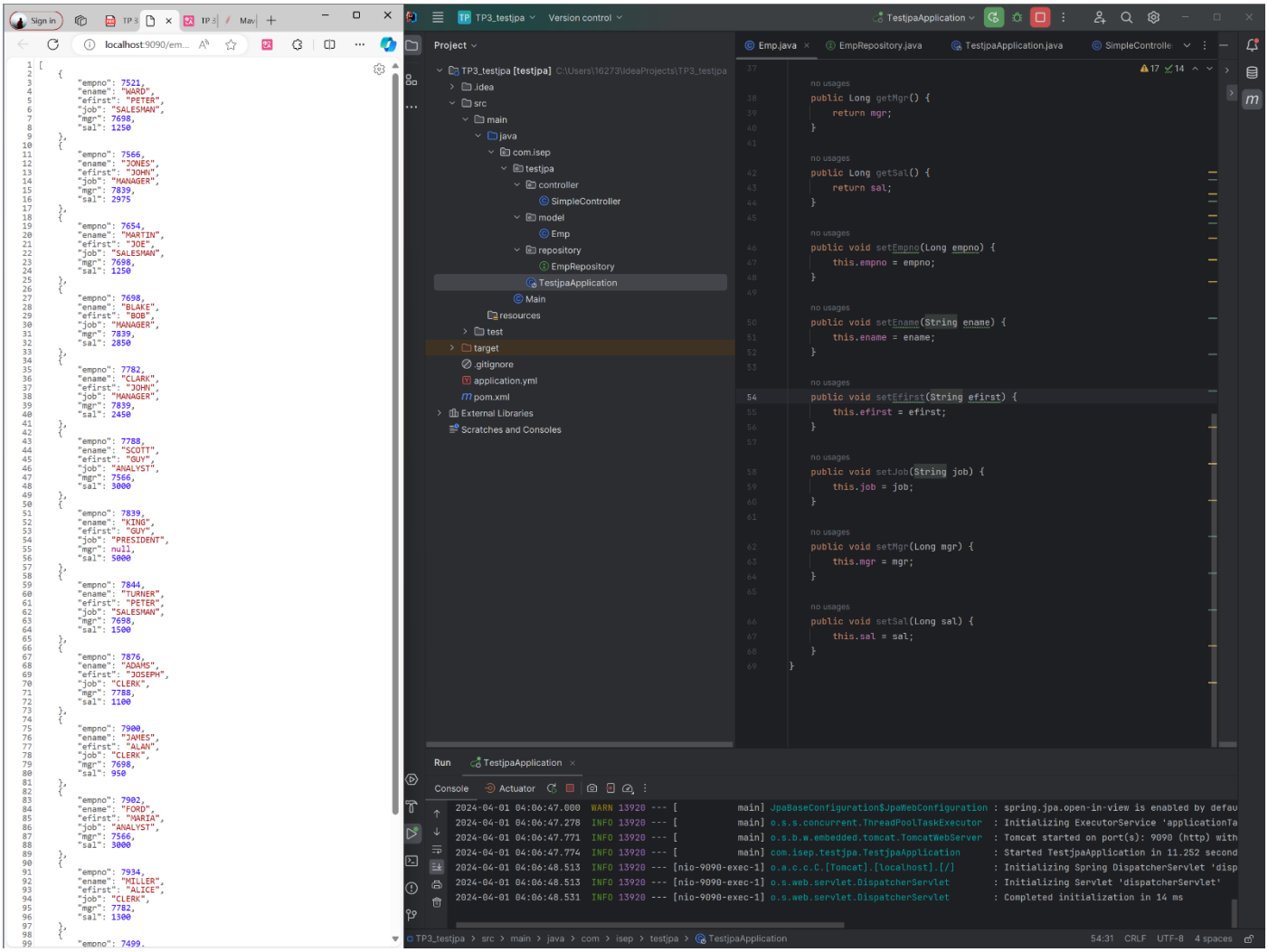


Exercice III.2 : Activate JPA and connect to your Database

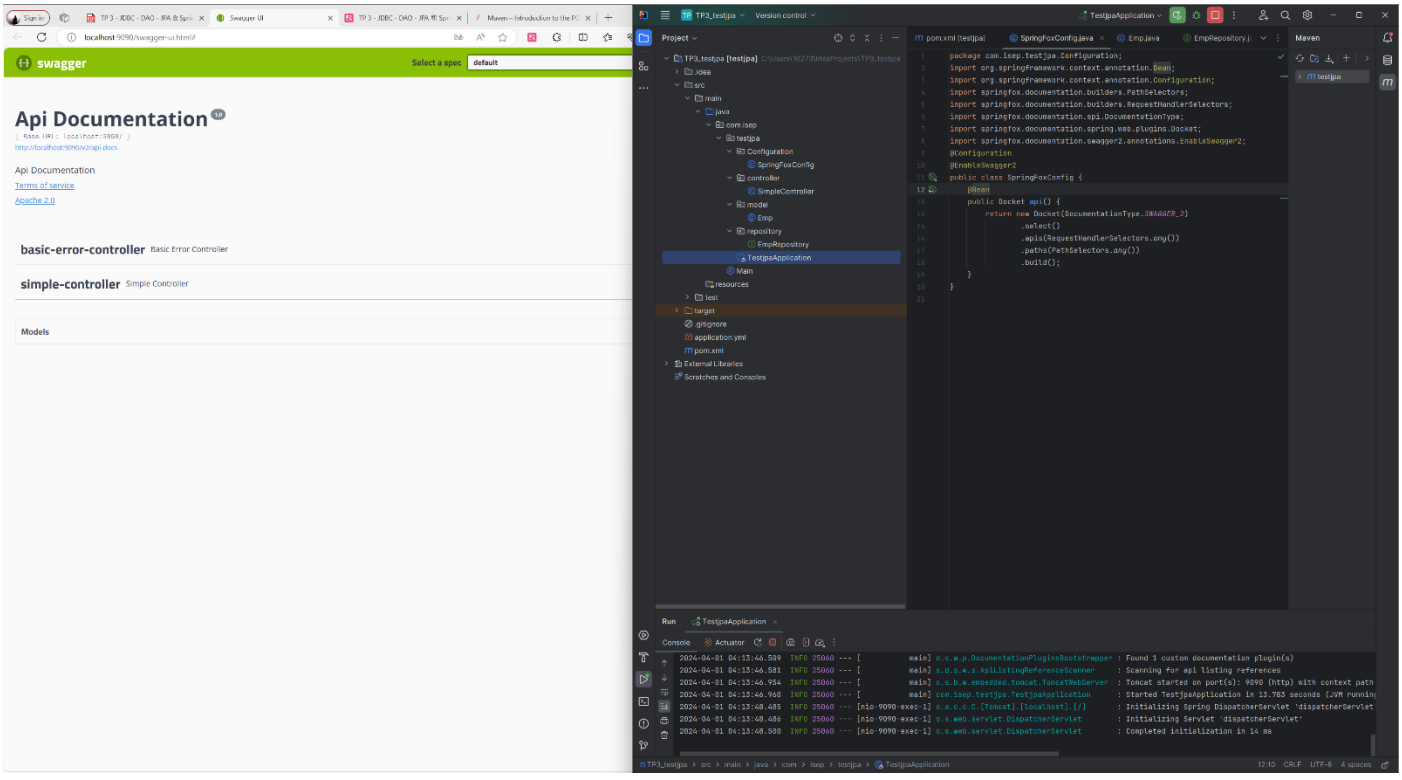


Exercise III.3 : Read Data in your Database





Exercise III.4 : Create an Object via JPA and expose it on a REST endpoint



Exercise III.5 : GET by ID / Update by id / Delete by ID / Get All, via JPA / REST