PES UNIVERSITY
ELECTRONIC CITY CAMPUS

(Established under Karnataka Act No. 16 of 2013)
100 Feet Ring Road, BSK III Stage, Bengaluru-560 085
Department of Computer Science and Engineering
Session: Aug – Dec, 2021
SEMESTER – 5

Database Management System(UE19CS301)

Assignment - 3

Team Details

Team Number-5

	NAME	SRN
1	Prachi Sengar	PES2UG19CS285
2	Raeesa Tanseen	PES2UG19CS310
3	Ria Singh	PES2UG19CS326

Problem Statement

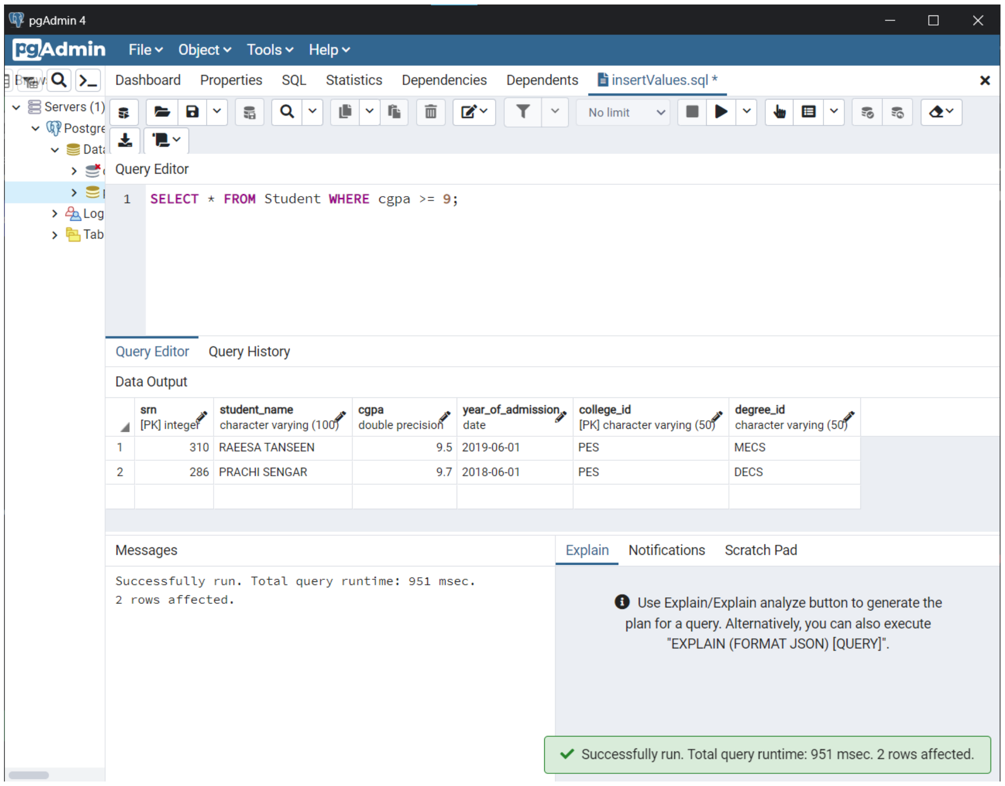
Our aim is to use a database management software and build a relational database for efficiently organizing, storing, managing, and using the data kept by a placement cell with details about students, colleges, and companies in order to make the placement process hassle-free.

Queries

Simple, Complex, Nested Queries And Concurrency Control

<u>Simple Queries:</u>

1. Display all the details of students who have a CGPA greater than or equal to 9.



```sql
SELECT * FROM Student WHERE cgpa >= 9;
```

Data Output

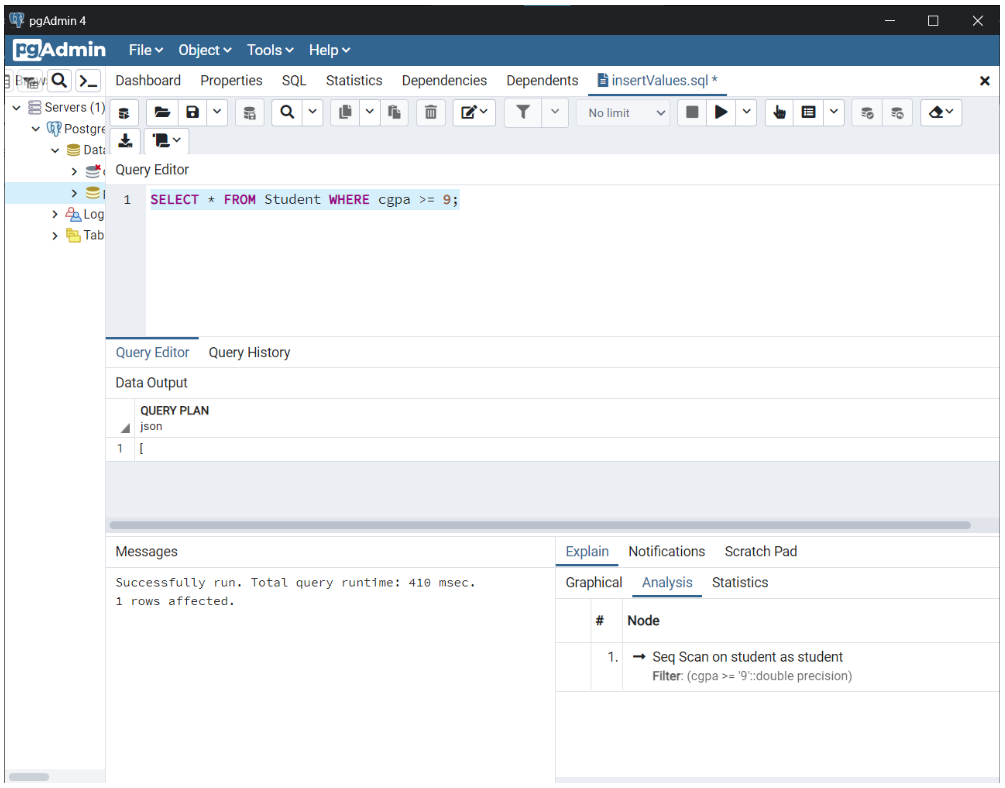
srn [PK] integer	student_name character varying (100)	cgpa double precision	year_of_admission date	college_id [PK] character varying (50)	degree_id character varying (50)
310	RAEESA TANSEEN	9.5	2019-06-01	PES	MECS
286	PRACHI SENGAR	9.7	2018-06-01	PES	DECS

Messages

```
Successfully run. Total query runtime: 951 msec.
2 rows affected.
```

Use Explain/Explain analyze button to generate the plan for a query. Alternatively, you can also execute "EXPLAIN (FORMAT JSON) [QUERY]".

Successfully run. Total query runtime: 951 msec. 2 rows affected.



```sql
SELECT * FROM Student WHERE cgpa >= 9;
```

Data Output

QUERY PLAN json
[

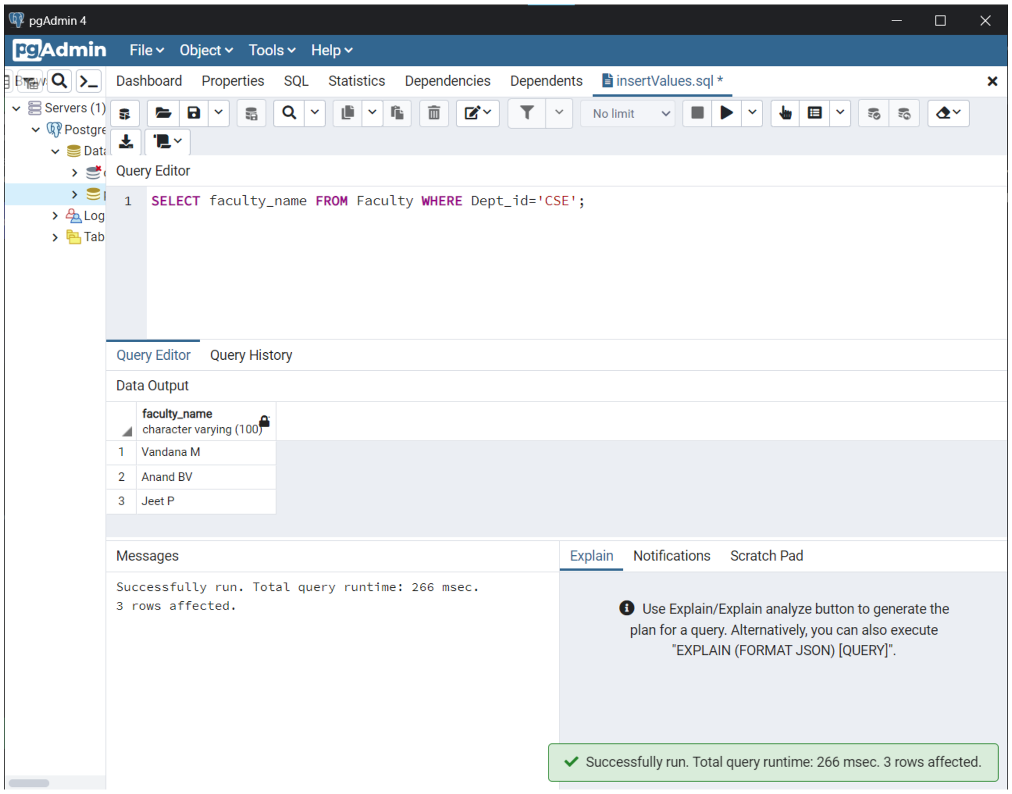
Messages

```
Successfully run. Total query runtime: 410 msec.
1 rows affected.
```

Explain — Analysis

#	Node
1.	→ Seq Scan on student as student Filter: (cgpa >= '9'::double precision)

2. Select the names of all the faculty members in the CSE department.



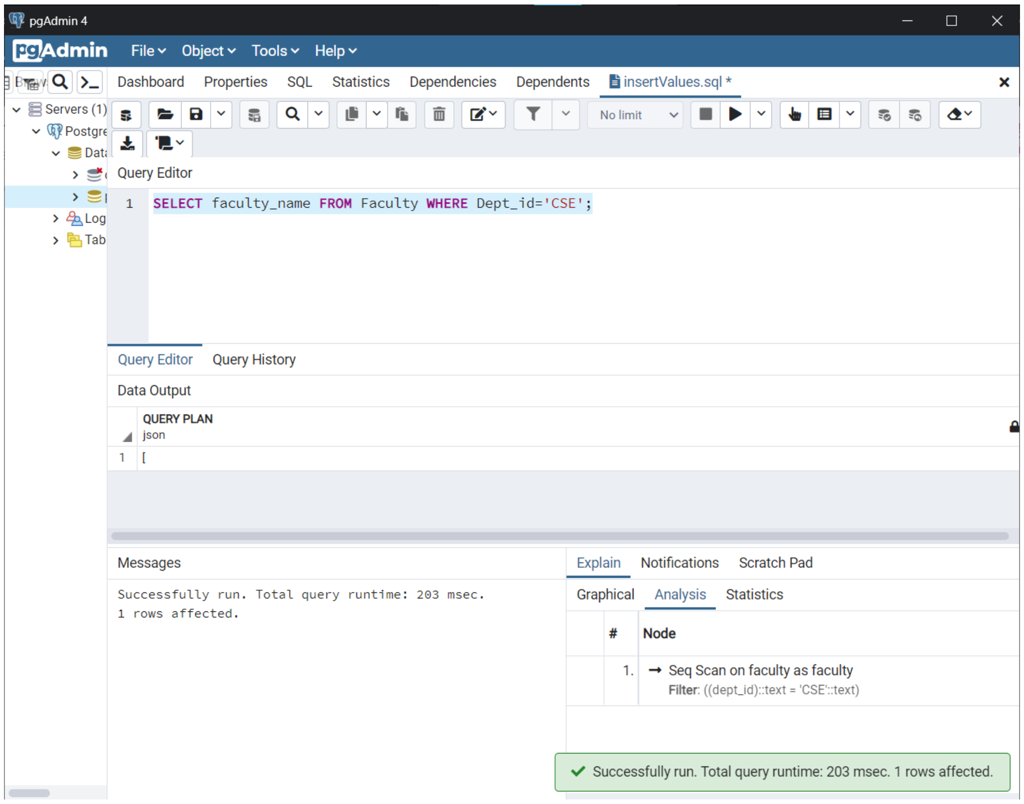
```
SELECT faculty_name FROM Faculty WHERE Dept_id='CSE';
```

	faculty_name character varying (100)
1	Vandana M
2	Anand BV
3	Jeet P

```
Successfully run. Total query runtime: 266 msec.
3 rows affected.
```

Use Explain/Explain analyze button to generate the plan for a query. Alternatively, you can also execute "EXPLAIN (FORMAT JSON) [QUERY]".

Successfully run. Total query runtime: 266 msec. 3 rows affected.



```
SELECT faculty_name FROM Faculty WHERE Dept_id='CSE';
```

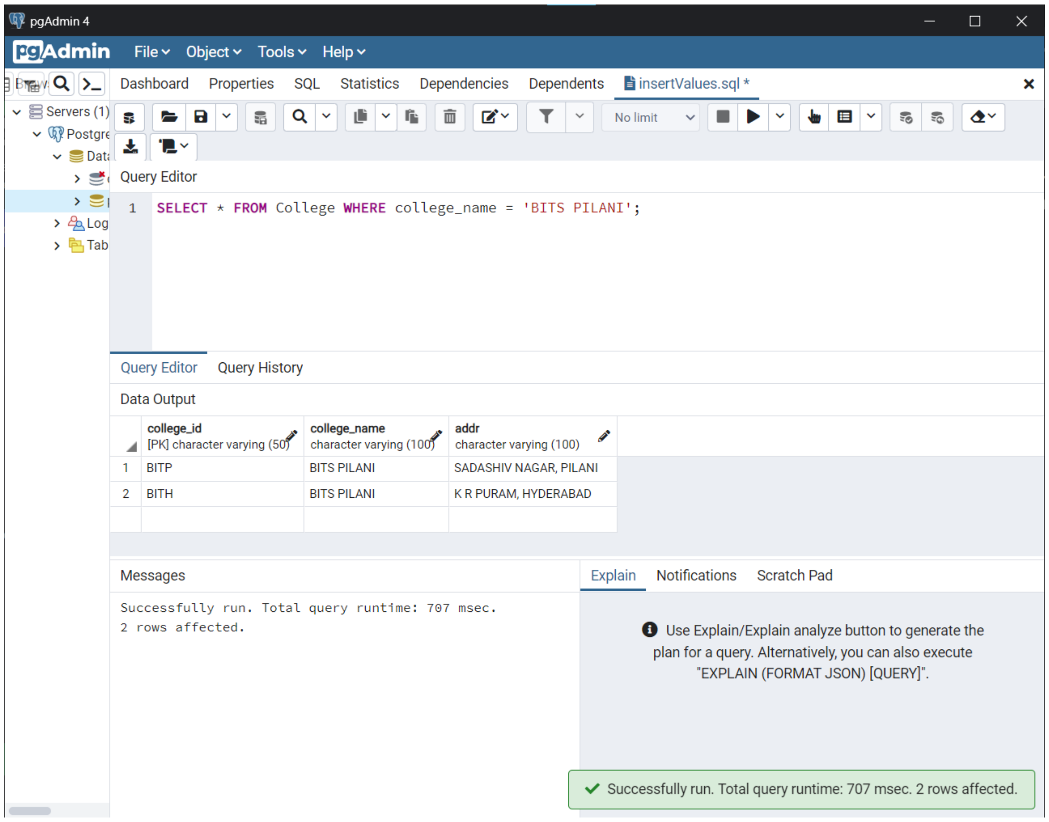
	QUERY PLAN json
1	[

```
Successfully run. Total query runtime: 203 msec.
1 rows affected.
```

#	Node
1.	→ Seq Scan on faculty as faculty Filter: ((dept_id)::text = 'CSE'::text)

Successfully run. Total query runtime: 203 msec. 1 rows affected.

3. Select all details of colleges with the college name as ‘BITS PILANI’.



```
SELECT * FROM College WHERE college_name = 'BITS PILANI';
```

Data Output

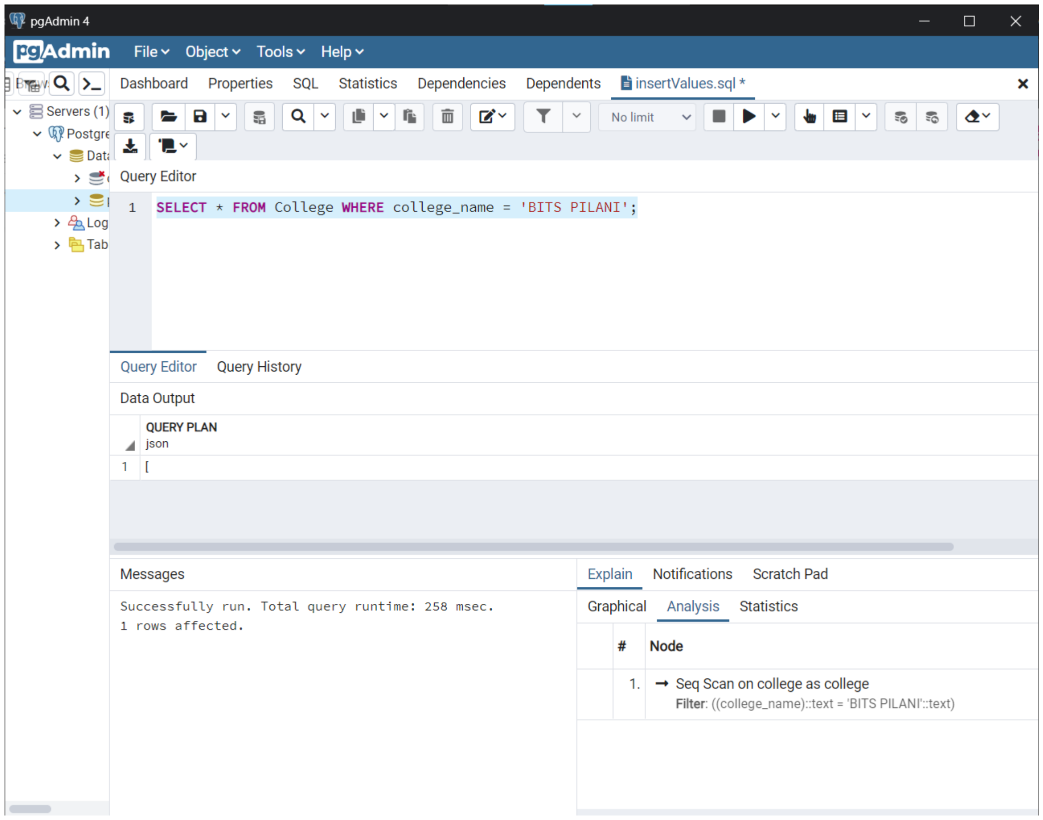
	college_id [PK] character varying (50)	college_name character varying (100)	addr character varying (100)
1	BITP	BITS PILANI	SADASHIV NAGAR, PILANI
2	BITH	BITS PILANI	K R PURAM, HYDERABAD

Messages

```
Successfully run. Total query runtime: 707 msec.
2 rows affected.
```

Use Explain/Explain analyze button to generate the plan for a query. Alternatively, you can also execute "EXPLAIN (FORMAT JSON) [QUERY]".

Successfully run. Total query runtime: 707 msec. 2 rows affected.



```
SELECT * FROM College WHERE college_name = 'BITS PILANI';
```

Data Output

	QUERY PLAN json
1	[

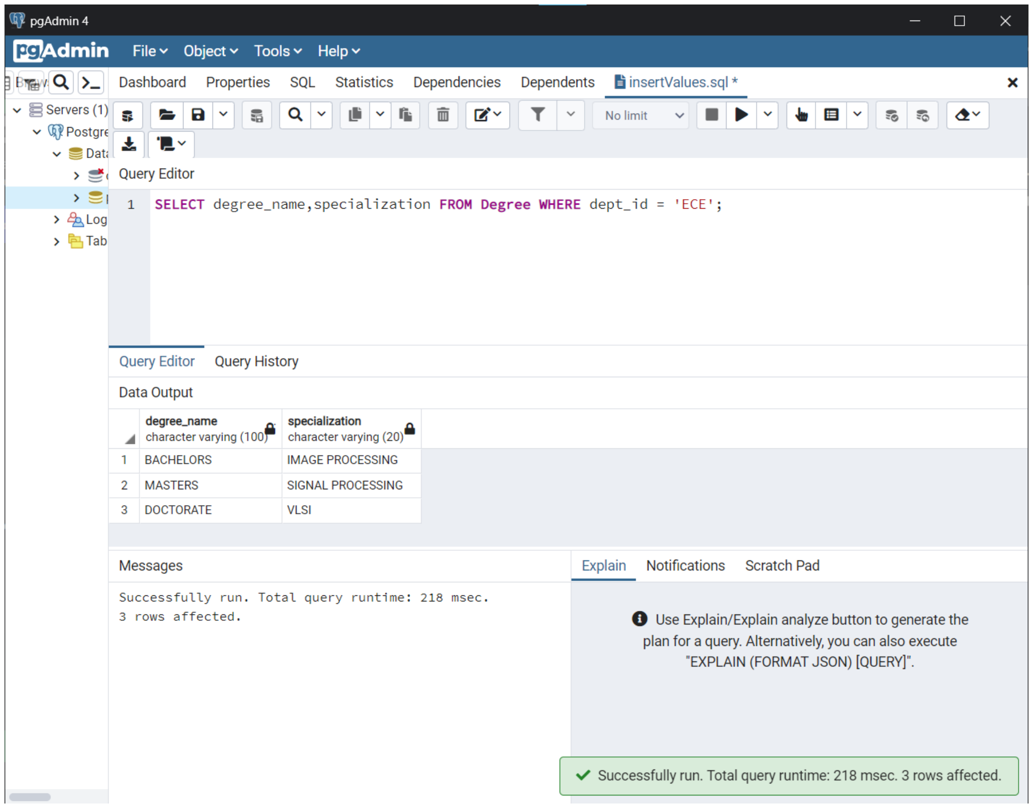
Messages

```
Successfully run. Total query runtime: 258 msec.
1 rows affected.
```

Explain — Analysis

#	Node
1.	→ Seq Scan on college as college Filter: ((college_name)::text = 'BITS PILANI'::text)

4. Display the degree names and specializations available in the ECE department.



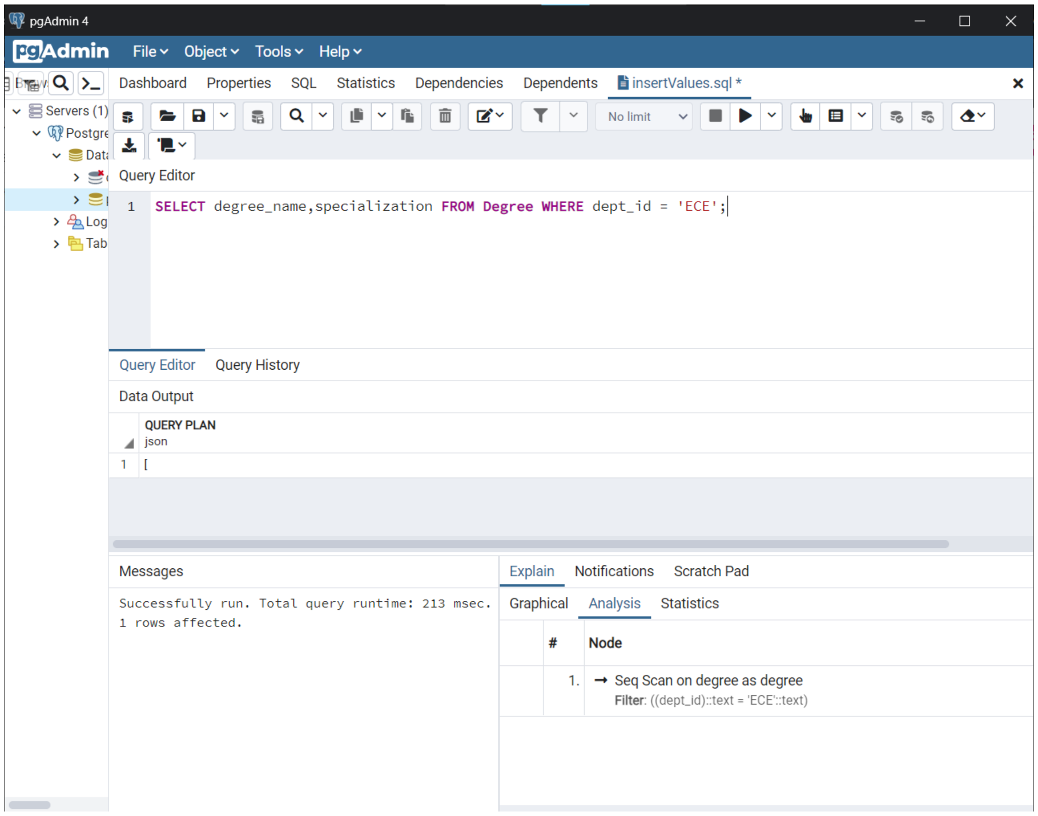
```
SELECT degree_name,specialization FROM Degree WHERE dept_id = 'ECE';
```

	degree_name character varying (100)	specialization character varying (20)
1	BACHELORS	IMAGE PROCESSING
2	MASTERS	SIGNAL PROCESSING
3	DOCTORATE	VLSI

```
Successfully run. Total query runtime: 218 msec.
3 rows affected.
```

Use Explain/Explain analyze button to generate the plan for a query. Alternatively, you can also execute "EXPLAIN (FORMAT JSON) [QUERY]".

Successfully run. Total query runtime: 218 msec. 3 rows affected.



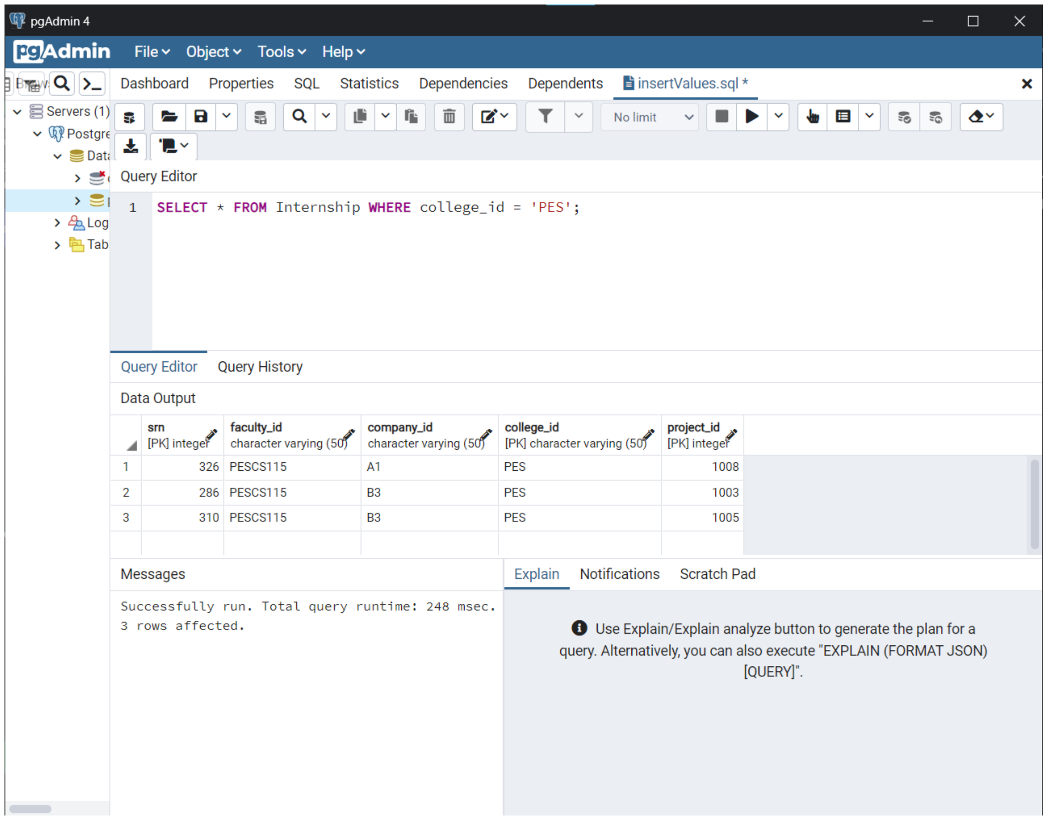
```
SELECT degree_name,specialization FROM Degree WHERE dept_id = 'ECE';
```

	QUERY PLAN json
1	[

```
Successfully run. Total query runtime: 213 msec.
1 rows affected.
```

#	Node
1.	→ Seq Scan on degree as degree Filter: ((dept_id)::text = 'ECE'::text)

5. Display details about internships offered to students in college with college id PES.

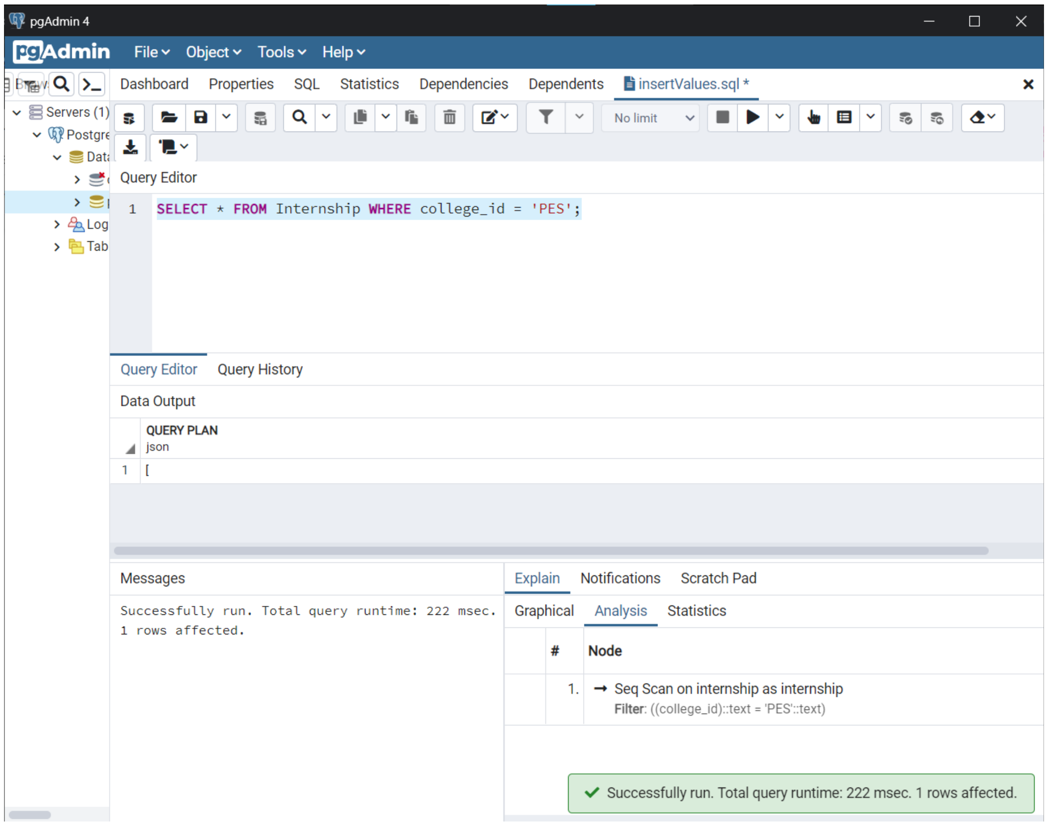


```
SELECT * FROM Internship WHERE college_id = 'PES';
```

	srn [PK] integer	faculty_id character varying (50)	company_id character varying (50)	college_id [PK] character varying (50)	project_id [PK] integer
1	326	PESCS115	A1	PES	1008
2	286	PESCS115	B3	PES	1003
3	310	PESCS115	B3	PES	1005

```
Successfully run. Total query runtime: 248 msec.
3 rows affected.
```

Use Explain/Explain analyze button to generate the plan for a query. Alternatively, you can also execute "EXPLAIN (FORMAT JSON) [QUERY]".



```
SELECT * FROM Internship WHERE college_id = 'PES';
```

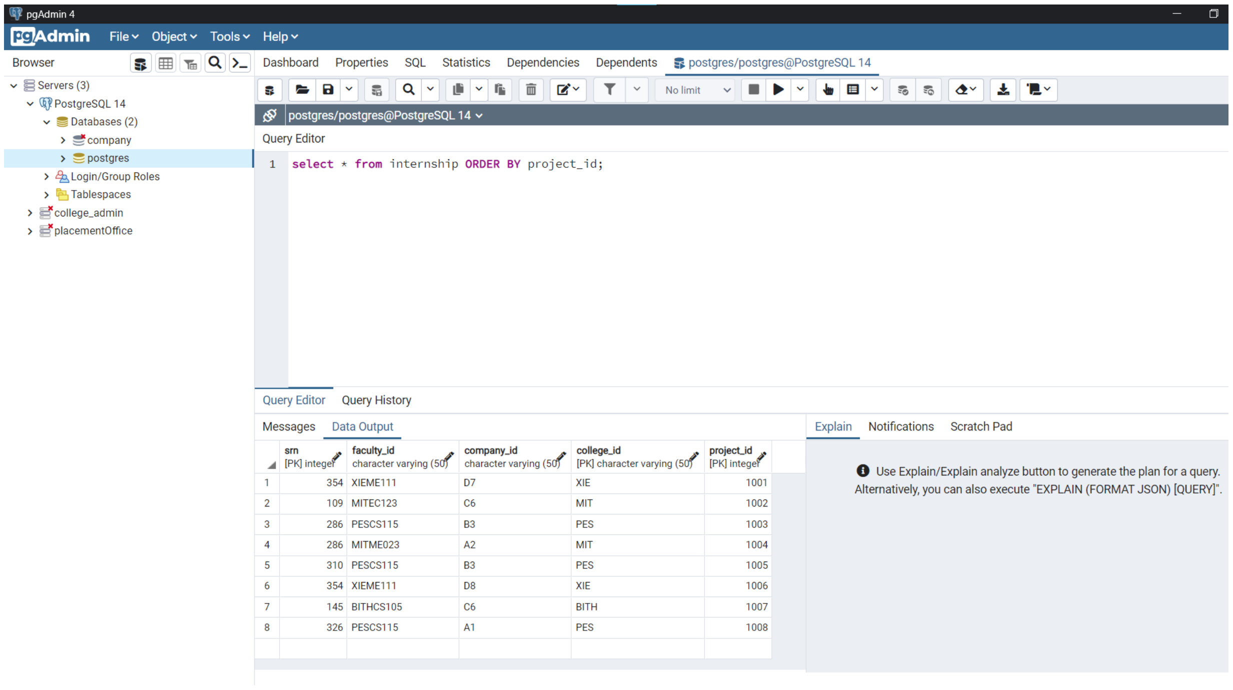
	QUERY PLAN json
1	[

```
Successfully run. Total query runtime: 222 msec.
1 rows affected.
```

#	Node
1.	→ Seq Scan on internship as internship Filter: ((college_id)::text = 'PES'::text)

Successfully run. Total query runtime: 222 msec. 1 rows affected.

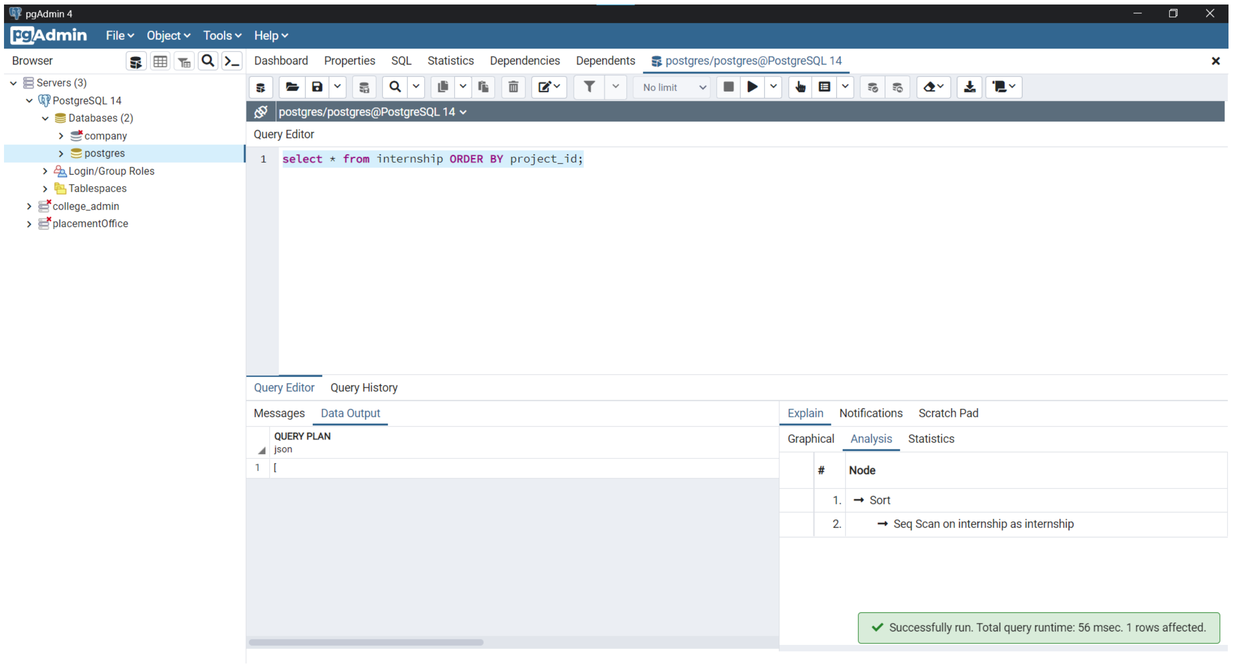
6. Display all the internships given, sorted according to project ID.



```
select * from internship ORDER BY project_id;
```

	srn [PK] integer	faculty_id character varying (50)	company_id character varying (50)	college_id [PK] character varying (50)	project_id [PK] integer
1	354	XIEME111	D7	XIE	1001
2	109	MITEC123	C6	MIT	1002
3	286	PESCS115	B3	PES	1003
4	286	MITME023	A2	MIT	1004
5	310	PESCS115	B3	PES	1005
6	354	XIEME111	D8	XIE	1006
7	145	BITHCS105	C6	BITH	1007
8	326	PESCS115	A1	PES	1008

Use Explain/Explain analyze button to generate the plan for a query. Alternatively, you can also execute "EXPLAIN (FORMAT JSON) [QUERY]".



```
select * from internship ORDER BY project_id;
```

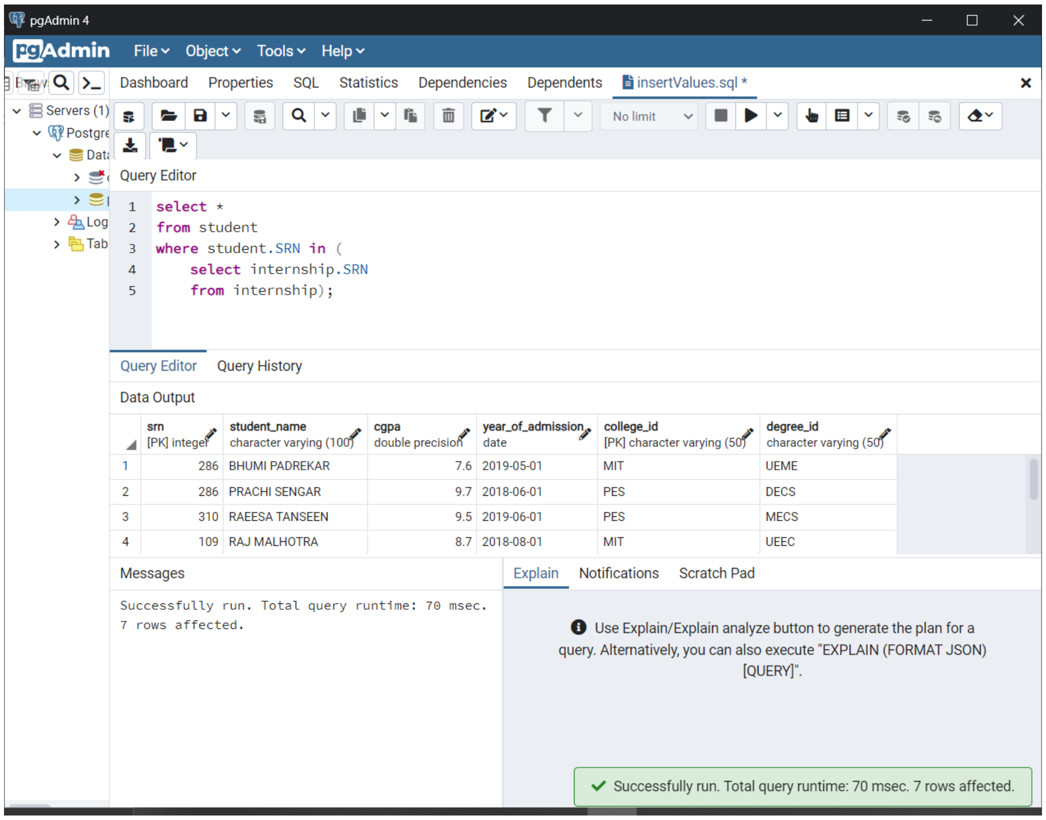
```
QUERY PLAN
json
1 [
```

#	Node
1.	→ Sort
2.	→ Seq Scan on internship as internship

Successfully run. Total query runtime: 56 msec. 1 rows affected.

Complex Queries:

7. Display all the details of students who have internships.



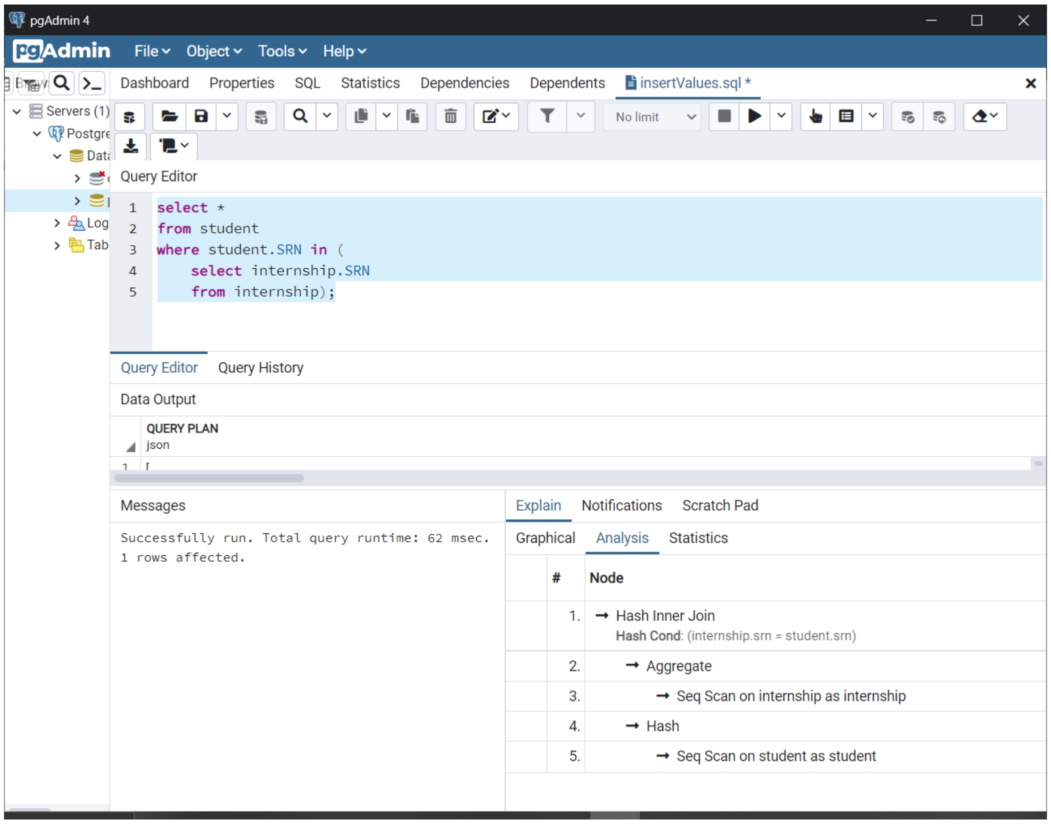
```
select *
from student
where student.SRN in (
    select internship.SRN
    from internship);
```

srn [PK] integer	student_name character varying (100)	cgpa double precision	year_of_admission date	college_id [PK] character varying (50)	degree_id character varying (50)
286	BHUMI PADREKAR	7.6	2019-05-01	MIT	UEME
286	PRACHI SENGAR	9.7	2018-06-01	PES	DECS
310	RAEESA TANSEEN	9.5	2019-06-01	PES	MECS
109	RAJ MALHOTRA	8.7	2018-08-01	MIT	UEEC

```
Successfully run. Total query runtime: 70 msec.
7 rows affected.
```

Use Explain/Explain analyze button to generate the plan for a query. Alternatively, you can also execute "EXPLAIN (FORMAT JSON) [QUERY]".

Successfully run. Total query runtime: 70 msec. 7 rows affected.



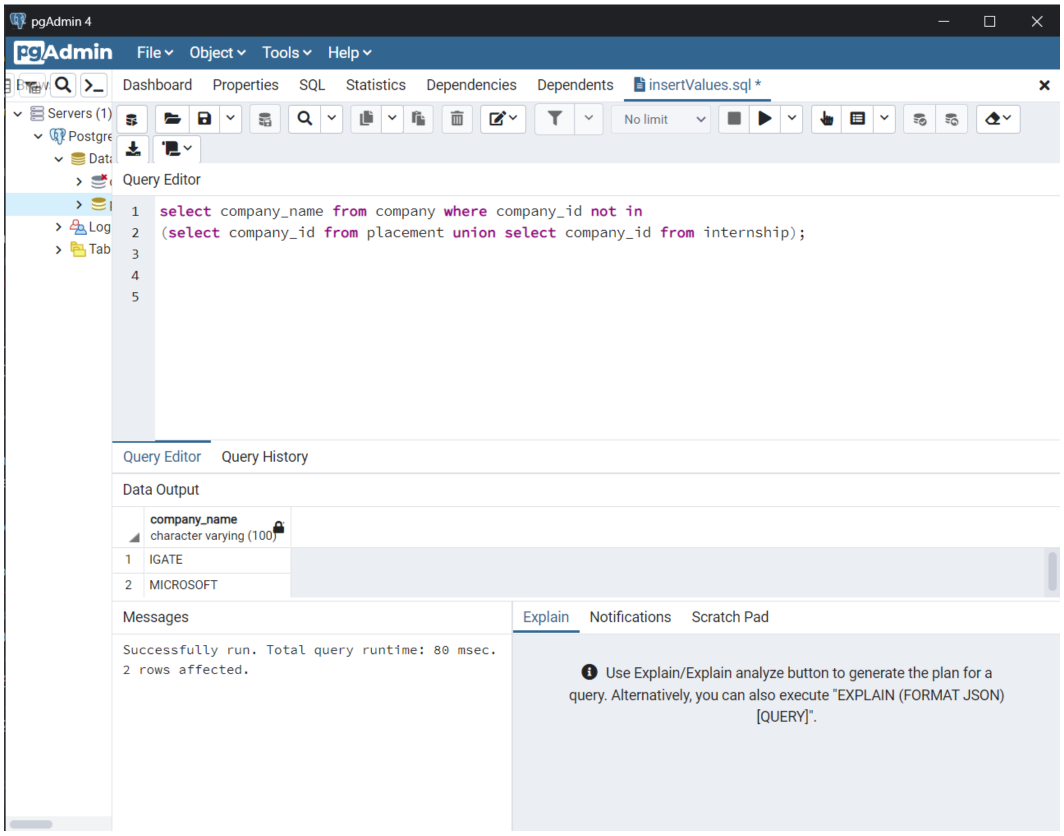
```
select *
from student
where student.SRN in (
    select internship.SRN
    from internship);
```

QUERY PLAN
json

```
Successfully run. Total query runtime: 62 msec.
1 rows affected.
```

#	Node
1.	→ Hash Inner Join Hash Cond: (internship.srn = student.srn)
2.	→ Aggregate
3.	→ Seq Scan on internship as internship
4.	→ Hash
5.	→ Seq Scan on student as student

8. Display names of companies which have not provided any internship or placement.



```
select company_name from company where company_id not in
(select company_id from placement union select company_id from internship);
```

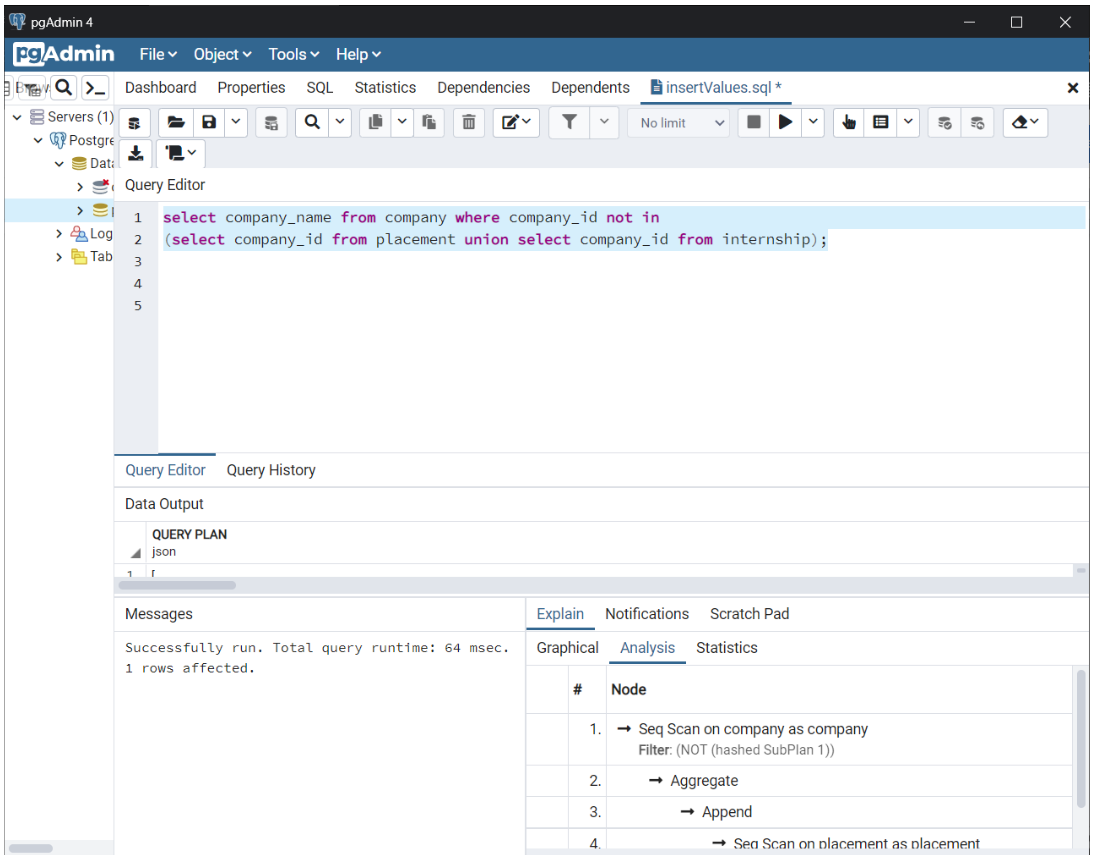
Data Output

	company_name character varying (100)
1	IGATE
2	MICROSOFT

Messages

```
Successfully run. Total query runtime: 80 msec.
2 rows affected.
```

Use Explain/Explain analyze button to generate the plan for a query. Alternatively, you can also execute "EXPLAIN (FORMAT JSON) [QUERY]".



```
select company_name from company where company_id not in
(select company_id from placement union select company_id from internship);
```

Data Output

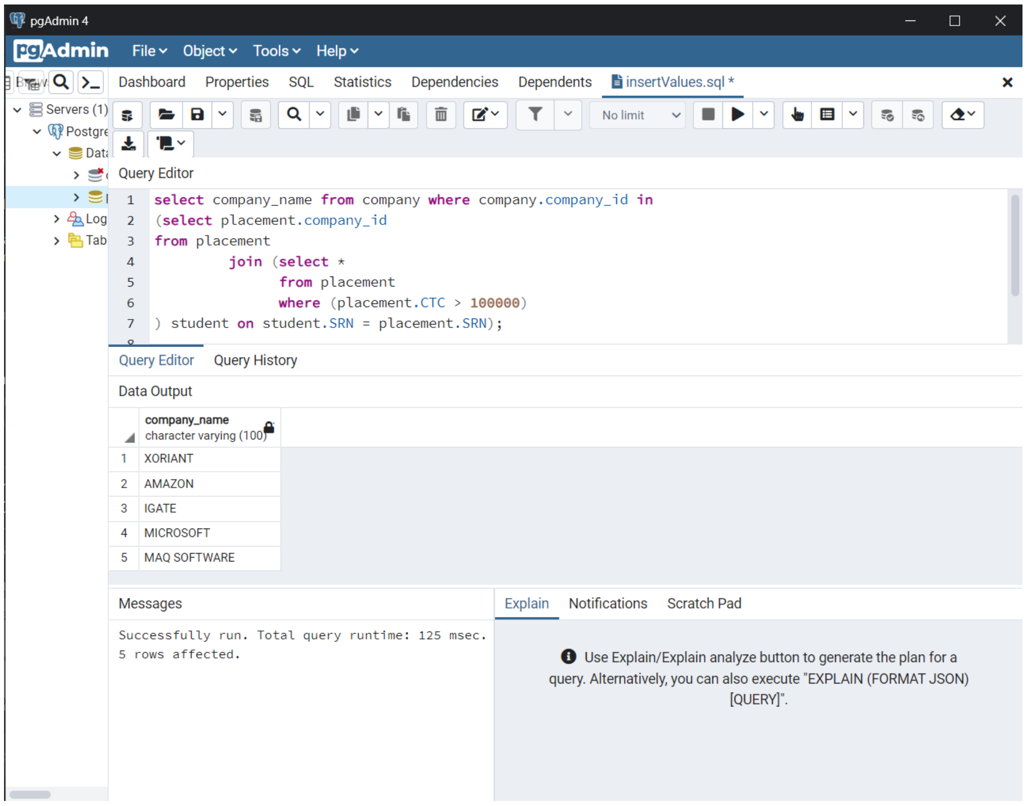
QUERY PLAN
json

Messages

```
Successfully run. Total query runtime: 64 msec.
1 rows affected.
```

#	Node
1.	→ Seq Scan on company as company Filter: (NOT (hashed SubPlan 1))
2.	→ Aggregate
3.	→ Append
4.	→ Seq Scan on placement as placement

9. Display names of companies who have offered placements to students with a CTC > 100000.

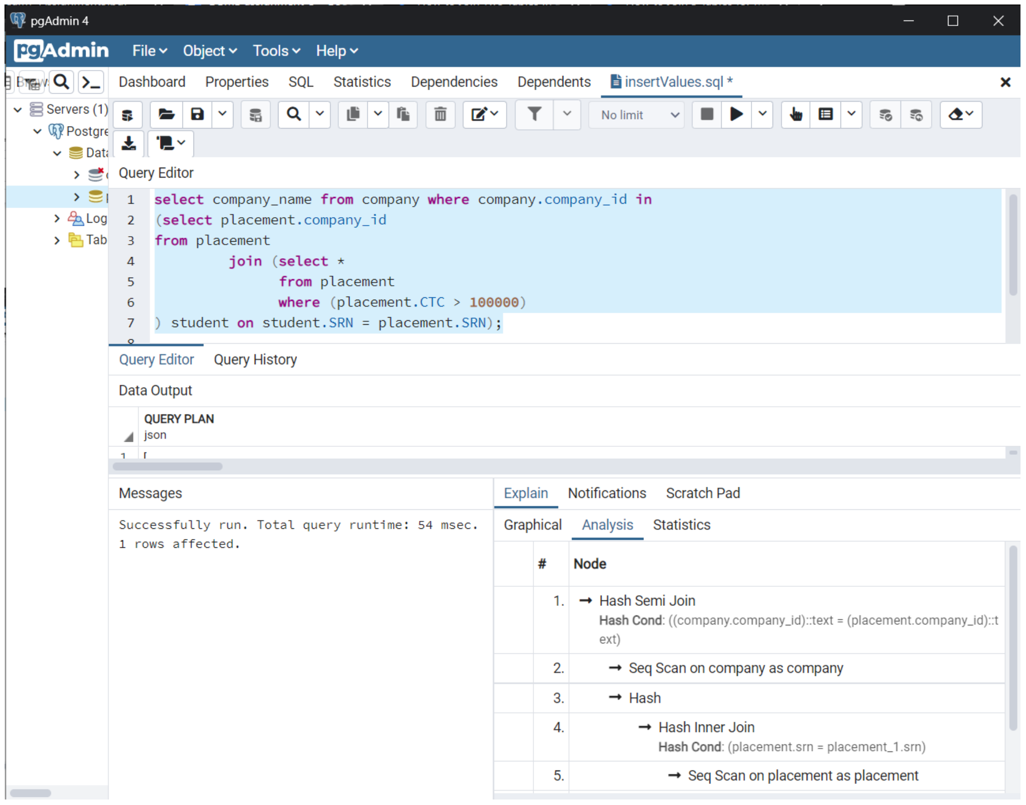

pgAdmin 4
File Object Tools Help
Dashboard Properties SQL Statistics Dependencies Dependents insertValues.sql *
No limit
Query Editor
Query Editor Query History
Data Output
Messages Explain Notifications Scratch Pad

```
select company_name from company where company.company_id in
(select placement.company_id
from placement
        join (select *
              from placement
              where (placement.CTC > 100000)
) student on student.SRN = placement.SRN);
```

	company_name character varying (100)
1	XORIANT
2	AMAZON
3	IGATE
4	MICROSOFT
5	MAQ SOFTWARE

```
Successfully run. Total query runtime: 125 msec.
5 rows affected.
```

Use Explain/Explain analyze button to generate the plan for a query. Alternatively, you can also execute "EXPLAIN (FORMAT JSON) [QUERY]".

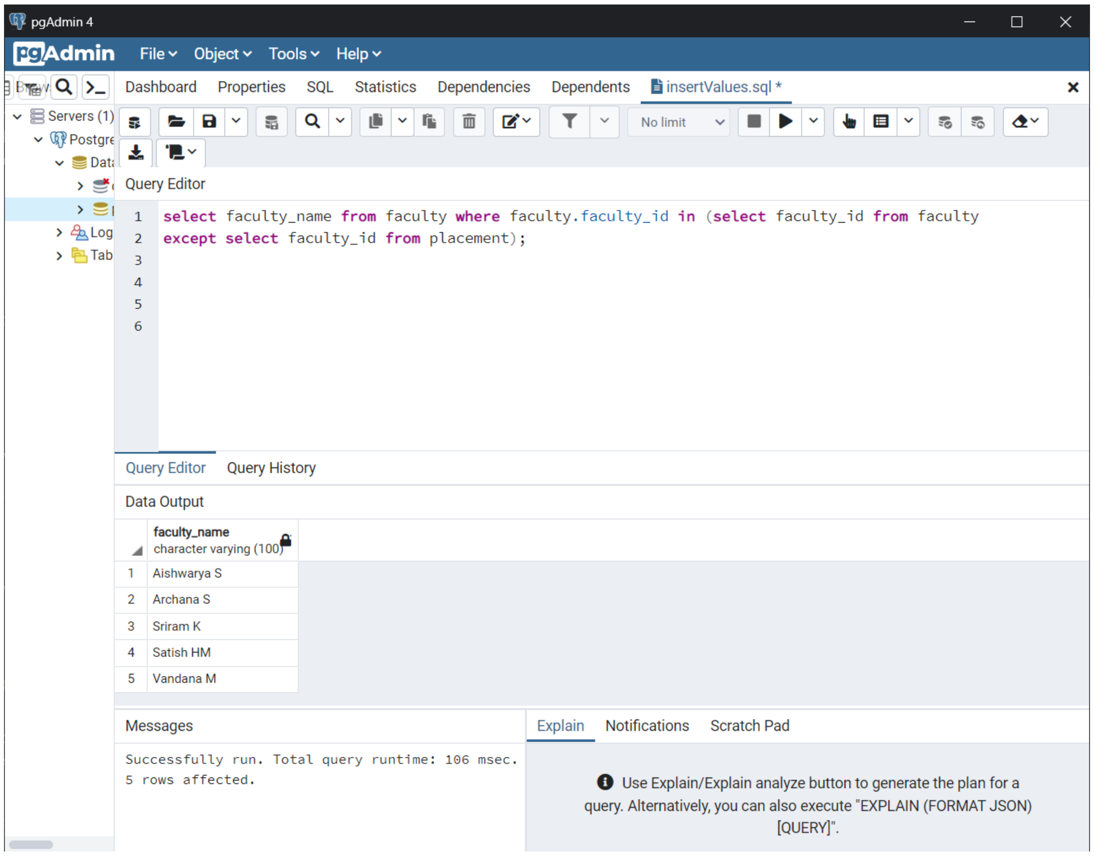

pgAdmin 4
File Object Tools Help
Dashboard Properties SQL Statistics Dependencies Dependents insertValues.sql *
No limit
Query Editor
Query Editor Query History
Data Output
QUERY PLAN json
Messages Explain Notifications Scratch Pad
Graphical Analysis Statistics

```
select company_name from company where company.company_id in
(select placement.company_id
from placement
        join (select *
              from placement
              where (placement.CTC > 100000)
) student on student.SRN = placement.SRN);
```

```
Successfully run. Total query runtime: 54 msec.
1 rows affected.
```

#	Node
1.	→ Hash Semi Join Hash Cond: ((company.company_id)::text = (placement.company_id)::text)
2.	→ Seq Scan on company as company
3.	→ Hash
4.	→ Hash Inner Join Hash Cond: (placement.srn = placement_1.srn)
5.	→ Seq Scan on placement as placement

10. Display names of faculty who are not incharge of any placements.



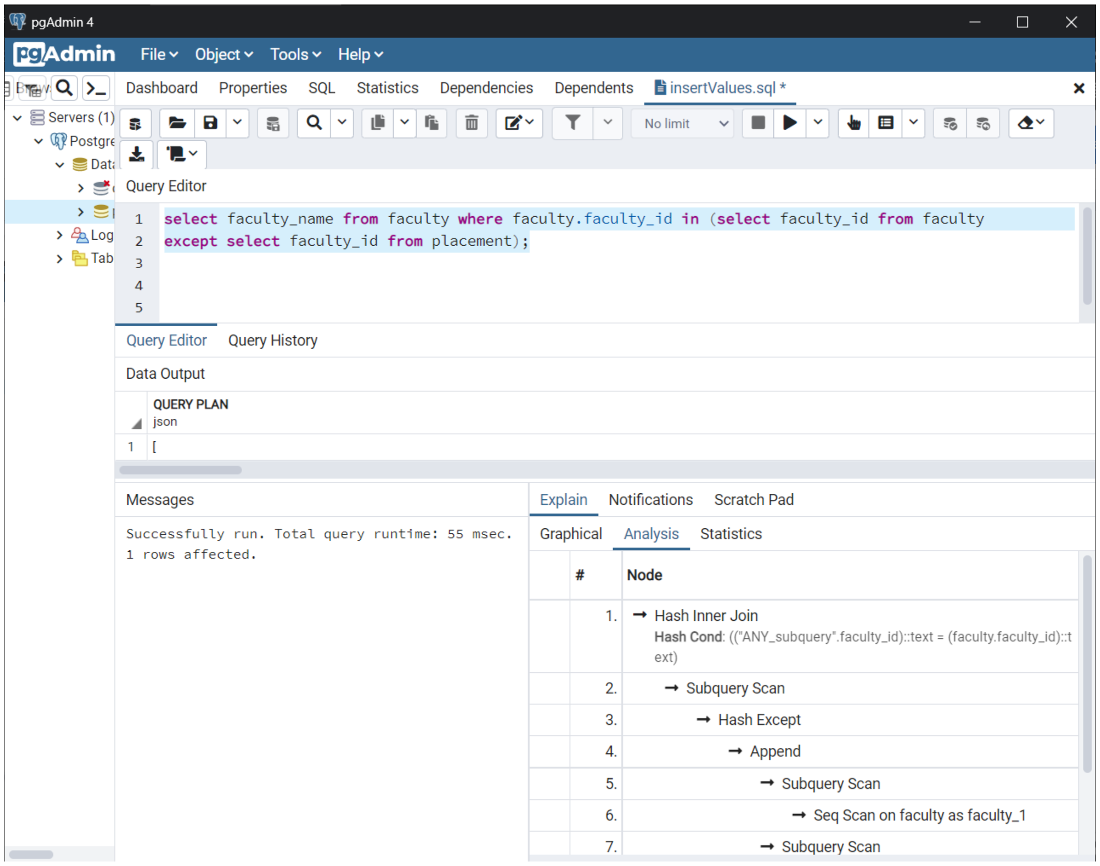
pgAdmin 4
File Object Tools Help
Dashboard Properties SQL Statistics Dependencies Dependents insertValues.sql *
Query Editor
Query Editor Query History
Data Output
Messages
Explain Notifications Scratch Pad

```
select faculty_name from faculty where faculty.faculty_id in (select faculty_id from faculty
except select faculty_id from placement);
```

	faculty_name character varying (100)
1	Aishwarya S
2	Archana S
3	Sriram K
4	Satish HM
5	Vandana M

```
Successfully run. Total query runtime: 106 msec.
5 rows affected.
```

Use Explain/Explain analyze button to generate the plan for a query. Alternatively, you can also execute "EXPLAIN (FORMAT JSON) [QUERY]".



pgAdmin 4
File Object Tools Help
Dashboard Properties SQL Statistics Dependencies Dependents insertValues.sql *
Query Editor
Query Editor Query History
Data Output
Messages
Explain Notifications Scratch Pad
Graphical Analysis Statistics

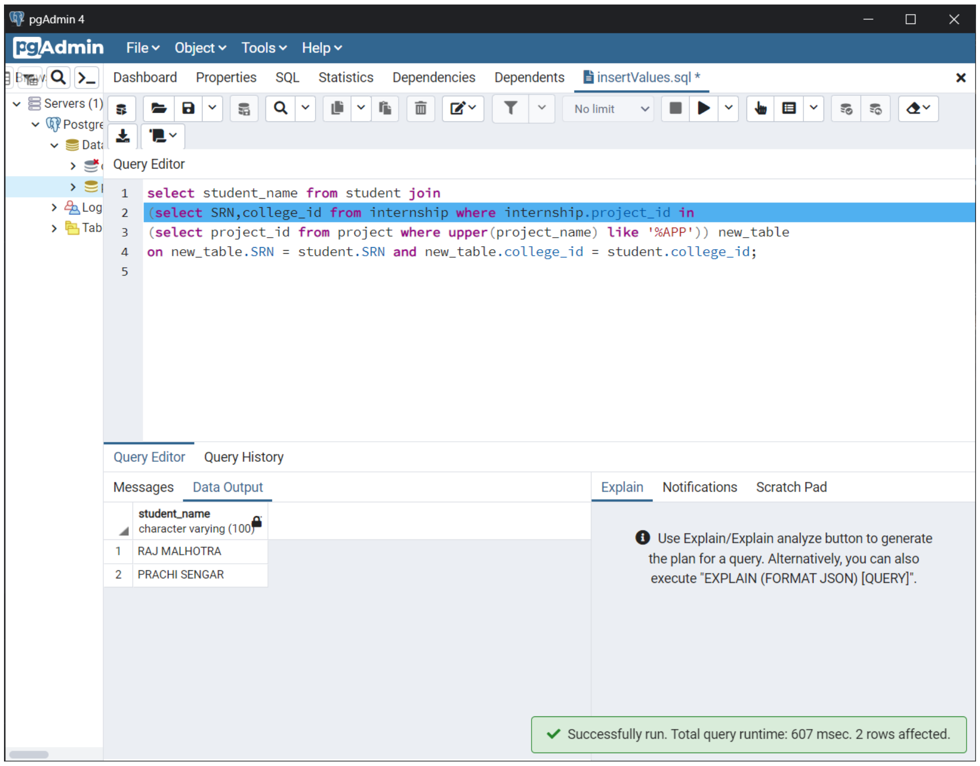
```
select faculty_name from faculty where faculty.faculty_id in (select faculty_id from faculty
except select faculty_id from placement);
```

	QUERY PLAN json
1	[

```
Successfully run. Total query runtime: 55 msec.
1 rows affected.
```

#	Node
1.	→ Hash Inner Join Hash Cond: (("ANY_subquery".faculty_id)::text = (faculty.faculty_id)::text)
2.	→ Subquery Scan
3.	→ Hash Except
4.	→ Append
5.	→ Subquery Scan
6.	→ Seq Scan on faculty as faculty_1
7.	→ Subquery Scan

11. Display names of the students who are working on an app as their project for their internship.

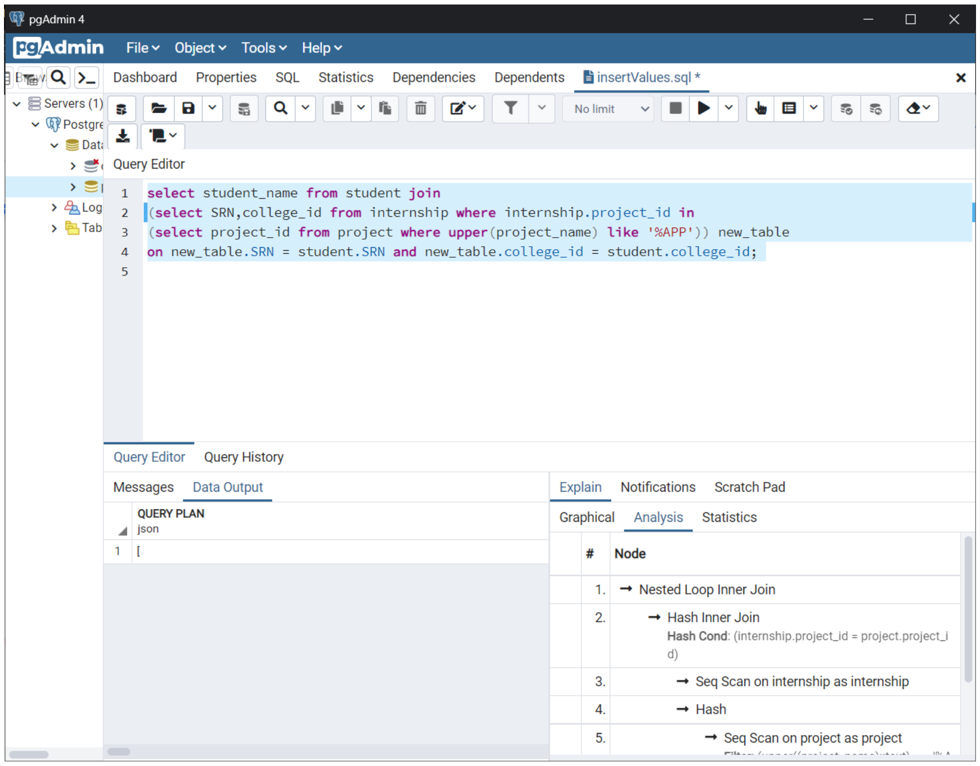

pgAdmin 4
File Object Tools Help
Dashboard Properties SQL Statistics Dependencies Dependents insertValues.sql *
Servers (1)
Postgre
Data
Log
Tab
No limit
Query Editor
Query Editor Query History
Messages Data Output
Explain Notifications Scratch Pad

```sql
select student_name from student join
(select SRN,college_id from internship where internship.project_id in
(select project_id from project where upper(project_name) like '%APP')) new_table
on new_table.SRN = student.SRN and new_table.college_id = student.college_id;
```

	student_name character varying (100)
1	RAJ MALHOTRA
2	PRACHI SENGAR

Use Explain/Explain analyze button to generate the plan for a query. Alternatively, you can also execute "EXPLAIN (FORMAT JSON) [QUERY]".

✓ Successfully run. Total query runtime: 607 msec. 2 rows affected.

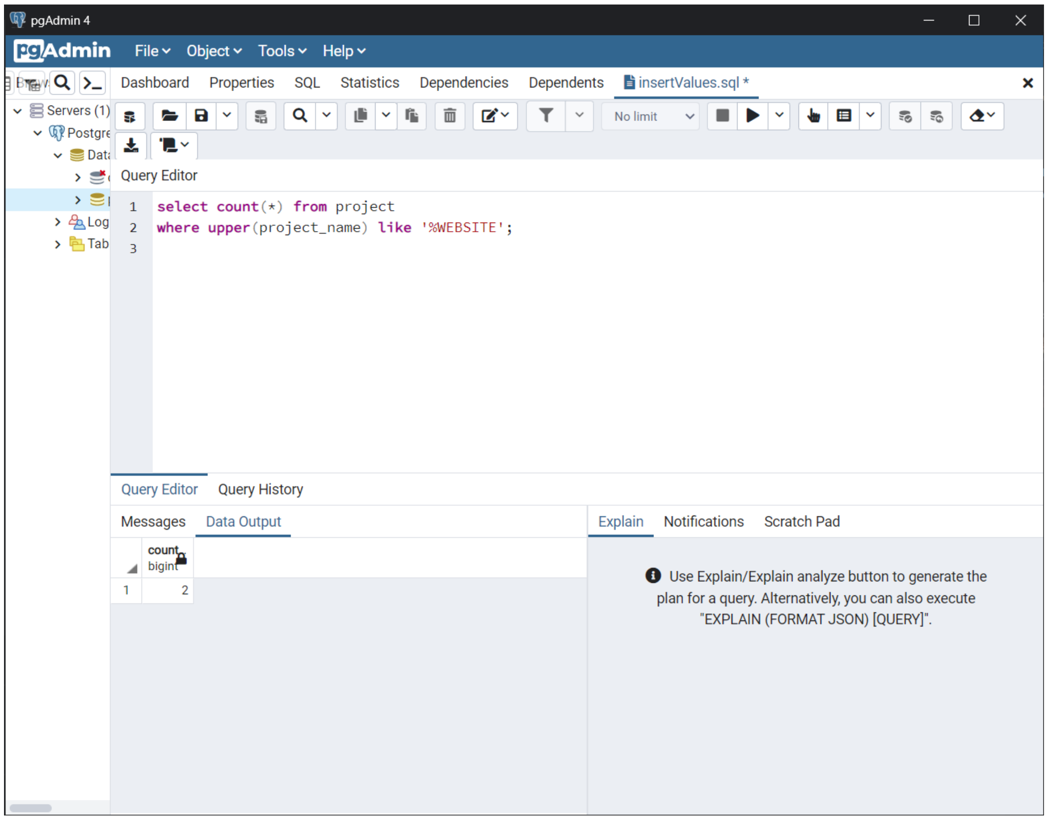

pgAdmin 4
File Object Tools Help
Dashboard Properties SQL Statistics Dependencies Dependents insertValues.sql *
Servers (1)
Postgre
Data
Log
Tab
No limit
Query Editor
Query Editor Query History
Messages Data Output
Explain Notifications Scratch Pad
Graphical Analysis Statistics

```sql
select student_name from student join
(select SRN,college_id from internship where internship.project_id in
(select project_id from project where upper(project_name) like '%APP')) new_table
on new_table.SRN = student.SRN and new_table.college_id = student.college_id;
```

	QUERY PLAN json
1	[

#	Node
1.	→ Nested Loop Inner Join
2.	→ Hash Inner Join Hash Cond: (internship.project_id = project.project_id)
3.	→ Seq Scan on internship as internship
4.	→ Hash
5.	→ Seq Scan on project as project

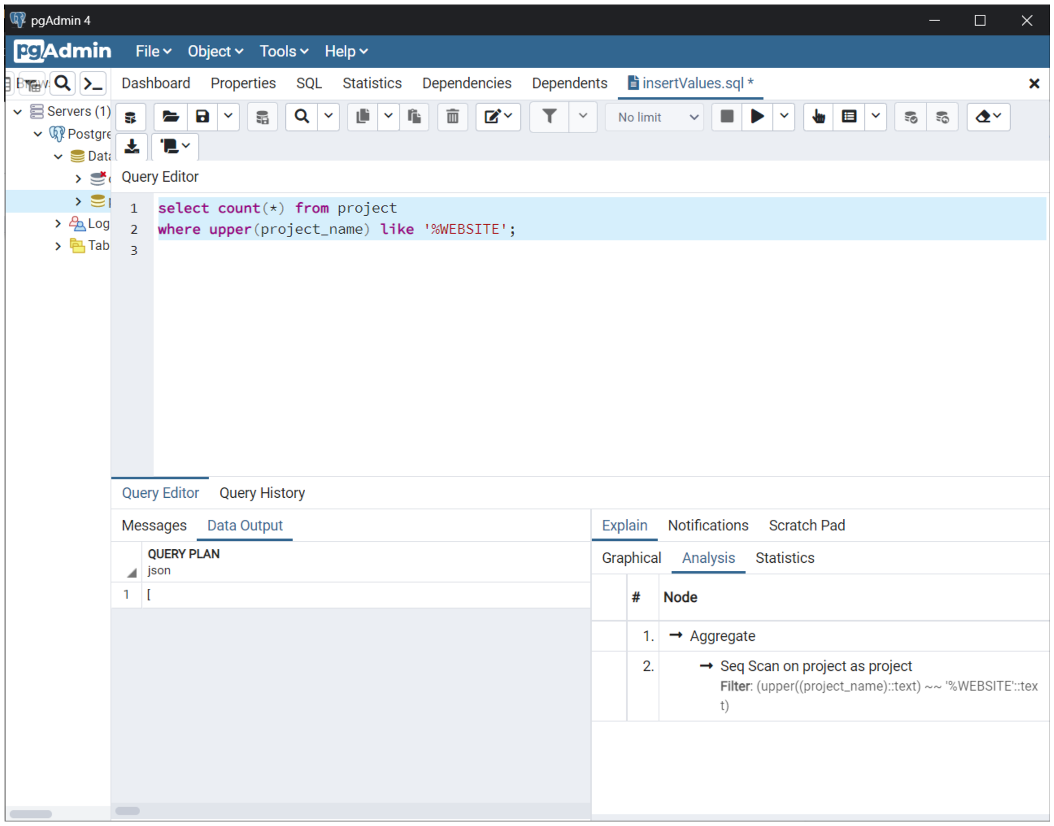
12. Display the number of students who are working on an internship project which involves making a website.



```
select count(*) from project
where upper(project_name) like '%WEBSITE';
```

	count bigint
1	2

Use Explain/Explain analyze button to generate the plan for a query. Alternatively, you can also execute "EXPLAIN (FORMAT JSON) [QUERY]".

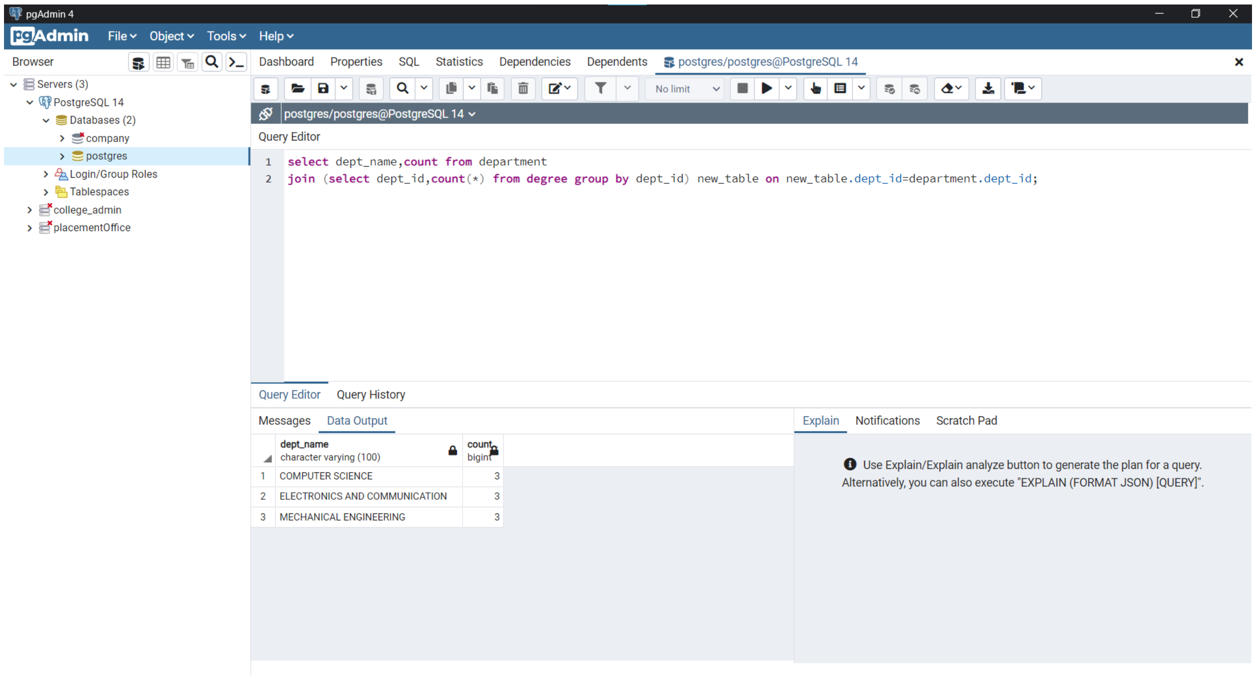


```
select count(*) from project
where upper(project_name) like '%WEBSITE';
```

	QUERY PLAN json
1	[

#	Node
1.	→ Aggregate
2.	→ Seq Scan on project as project Filter: (upper((project_name)::text) ~~ '%WEBSITE'::text)

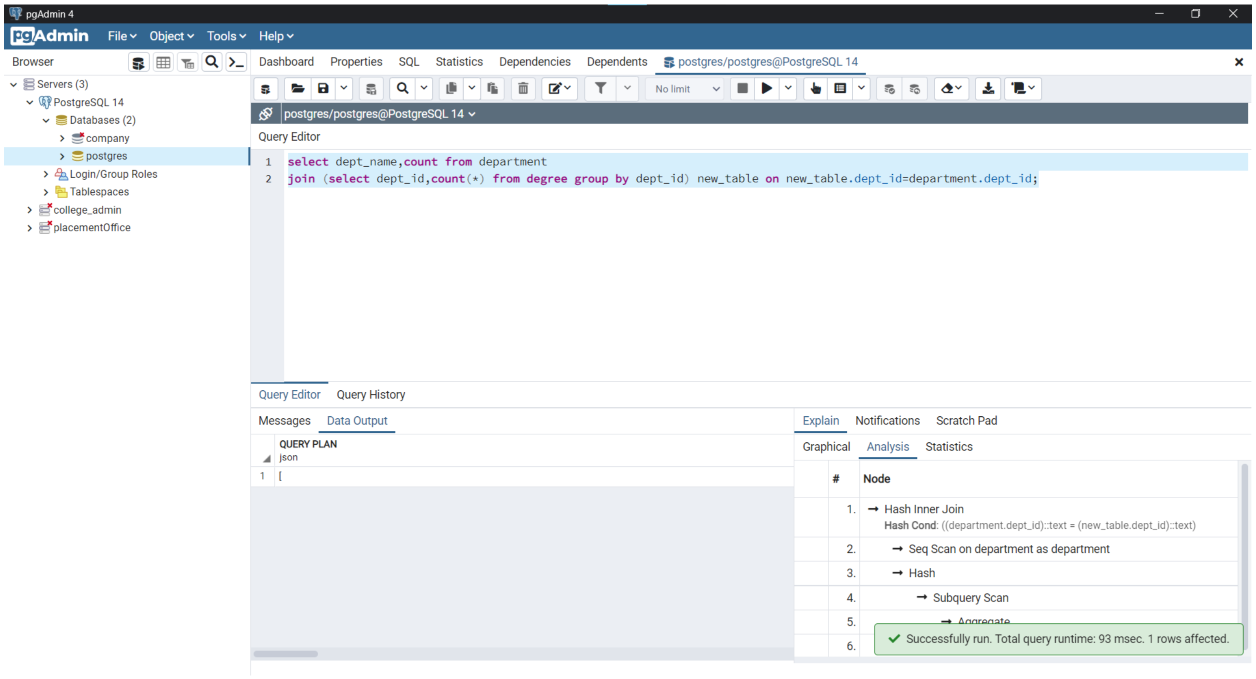
13. Display the number of degrees offered by each department.



```
select dept_name,count from department
join (select dept_id,count(*) from degree group by dept_id) new_table on new_table.dept_id=department.dept_id;
```

	dept_name character varying (100)	count bigint
1	COMPUTER SCIENCE	3
2	ELECTRONICS AND COMMUNICATION	3
3	MECHANICAL ENGINEERING	3

Use Explain/Explain analyze button to generate the plan for a query. Alternatively, you can also execute "EXPLAIN (FORMAT JSON) [QUERY]".

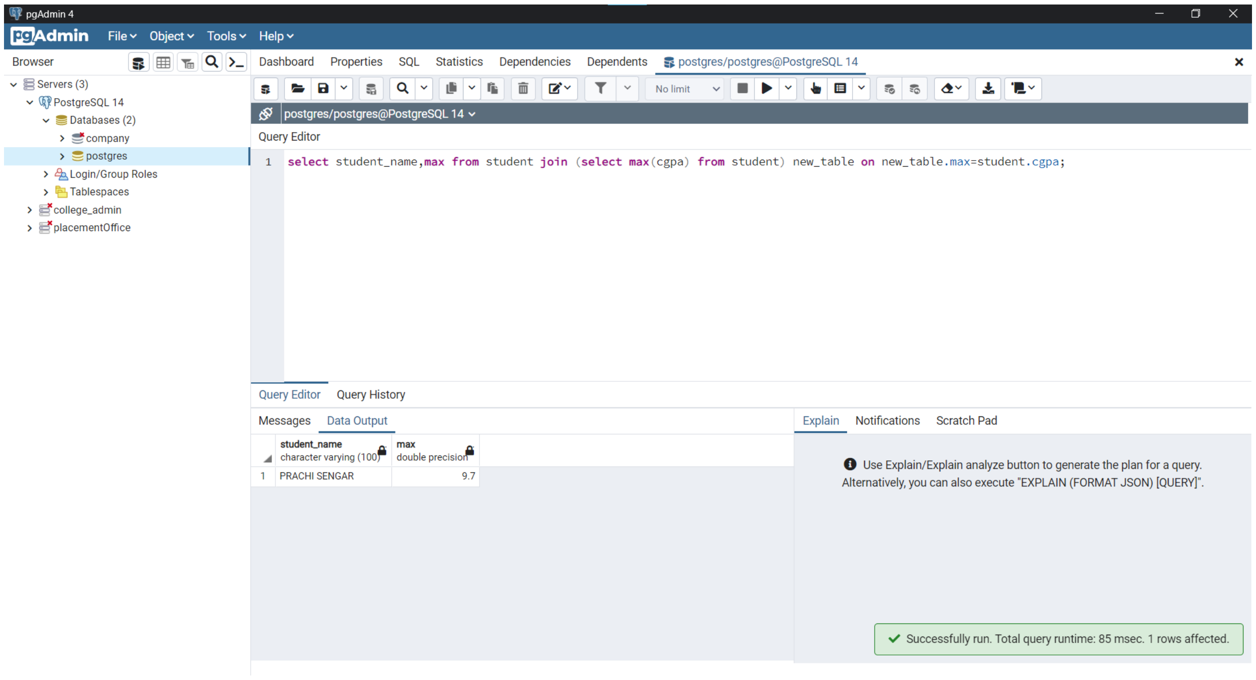


```
select dept_name,count from department
join (select dept_id,count(*) from degree group by dept_id) new_table on new_table.dept_id=department.dept_id;
```

#	Node
1.	→ Hash Inner Join Hash Cond: ((department.dept_id)::text = (new_table.dept_id)::text)
2.	→ Seq Scan on department as department
3.	→ Hash
4.	→ Subquery Scan
5.	→ Aggregate
6.	

Successfully run. Total query runtime: 93 msec. 1 rows affected.

14. Find the name of the student with the highest CGPA.

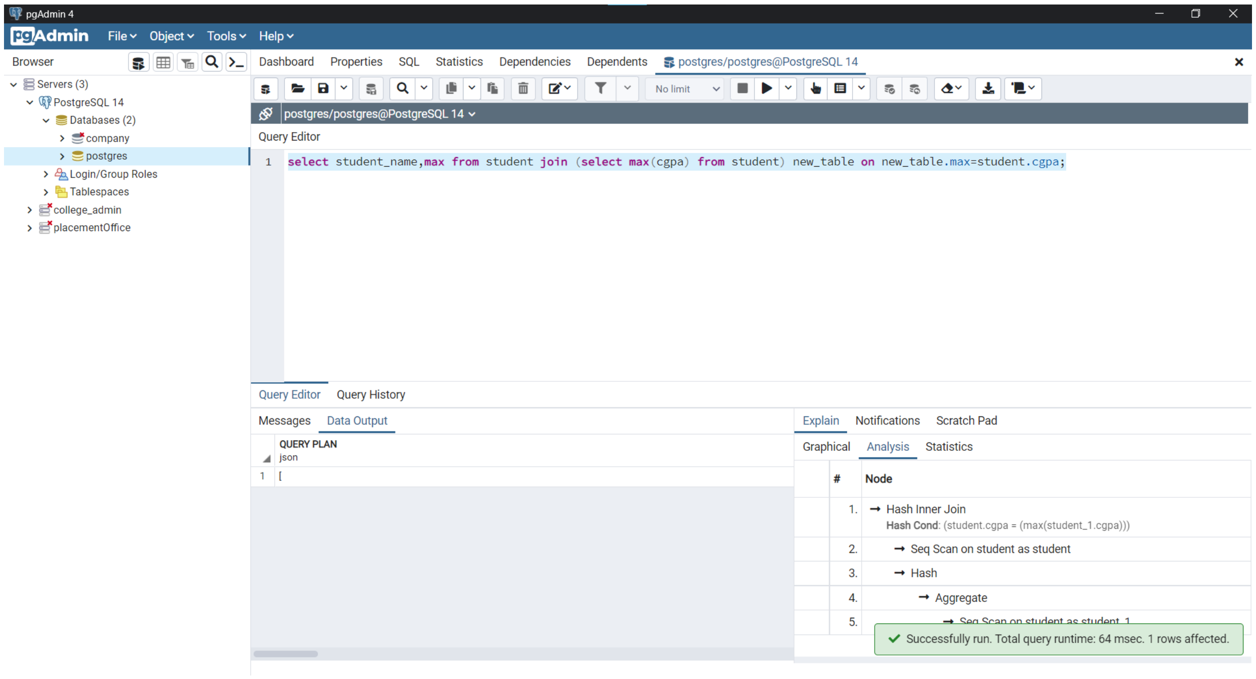


```
select student_name,max from student join (select max(cgpa) from student) new_table on new_table.max=student.cgpa;
```

	student_name character varying (100)	max double precision
1	PRACHI SENGAR	9.7

Use Explain/Explain analyze button to generate the plan for a query. Alternatively, you can also execute "EXPLAIN (FORMAT JSON) [QUERY]".

Successfully run. Total query runtime: 85 msec. 1 rows affected.



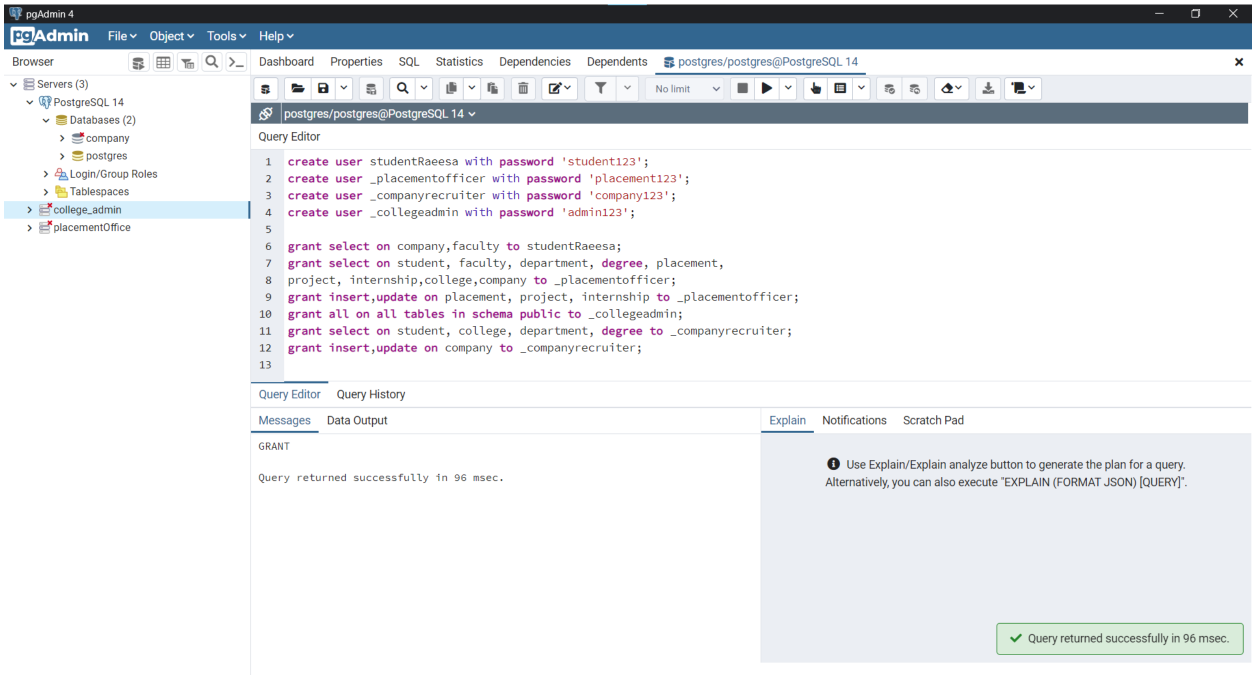
```
select student_name,max from student join (select max(cgpa) from student) new_table on new_table.max=student.cgpa;
```

	QUERY PLAN json
1	[

#	Node
1.	→ Hash Inner Join Hash Cond: (student.cgpa = (max(student_1.cgpa)))
2.	→ Seq Scan on student as student
3.	→ Hash
4.	→ Aggregate
5.	→ Seq Scan on student as student_1

Successfully run. Total query runtime: 64 msec. 1 rows affected.

Creating multiple users with different access privilege levels for different parts of the database.

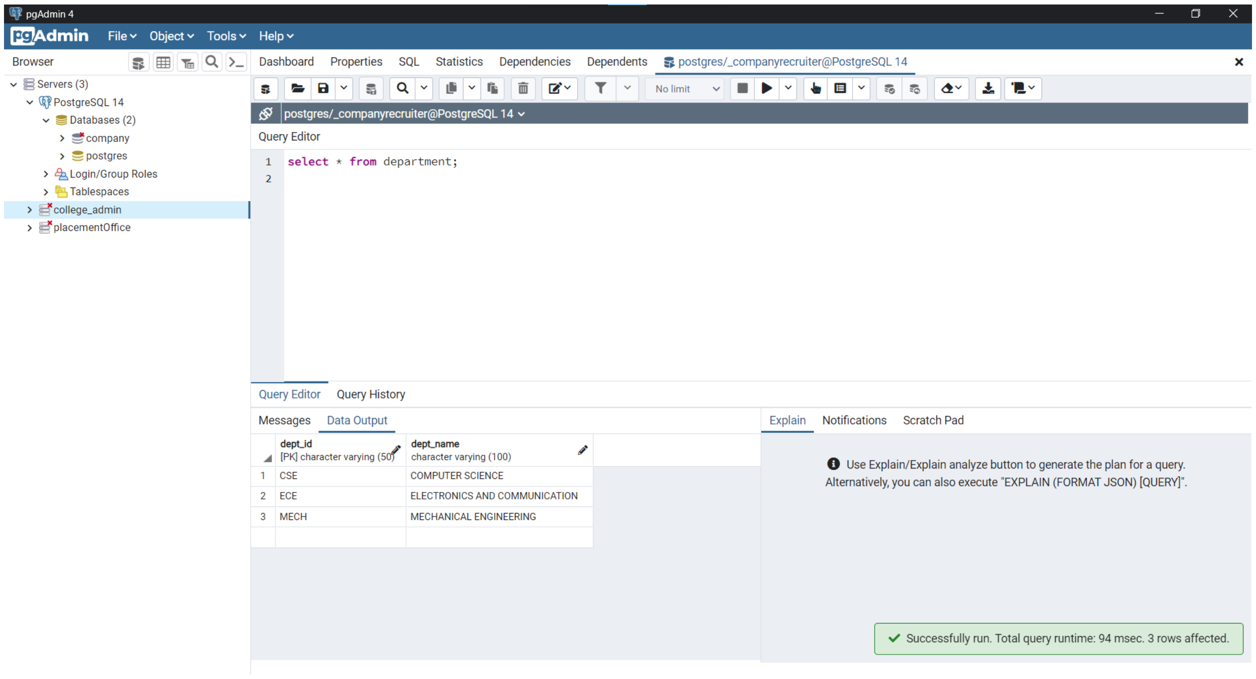


```
create user studentRaeesa with password 'student123';
create user _placementofficer with password 'placement123';
create user _companyrecruiter with password 'company123';
create user _collegeadmin with password 'admin123';

grant select on company,faculty to studentRaeesa;
grant select on student, faculty, department, degree, placement,
project, internship,college,company to _placementofficer;
grant insert,update on placement, project, internship to _placementofficer;
grant all on all tables in schema public to _collegeadmin;
grant select on student, college, department, degree to _companyrecruiter;
grant insert,update on company to _companyrecruiter;
```

```
GRANT

Query returned successfully in 96 msec.
```

Query returned successfully in 96 msec.

Initiating concurrent Transactions and demonstrating the concurrency control for the conflicting actions.

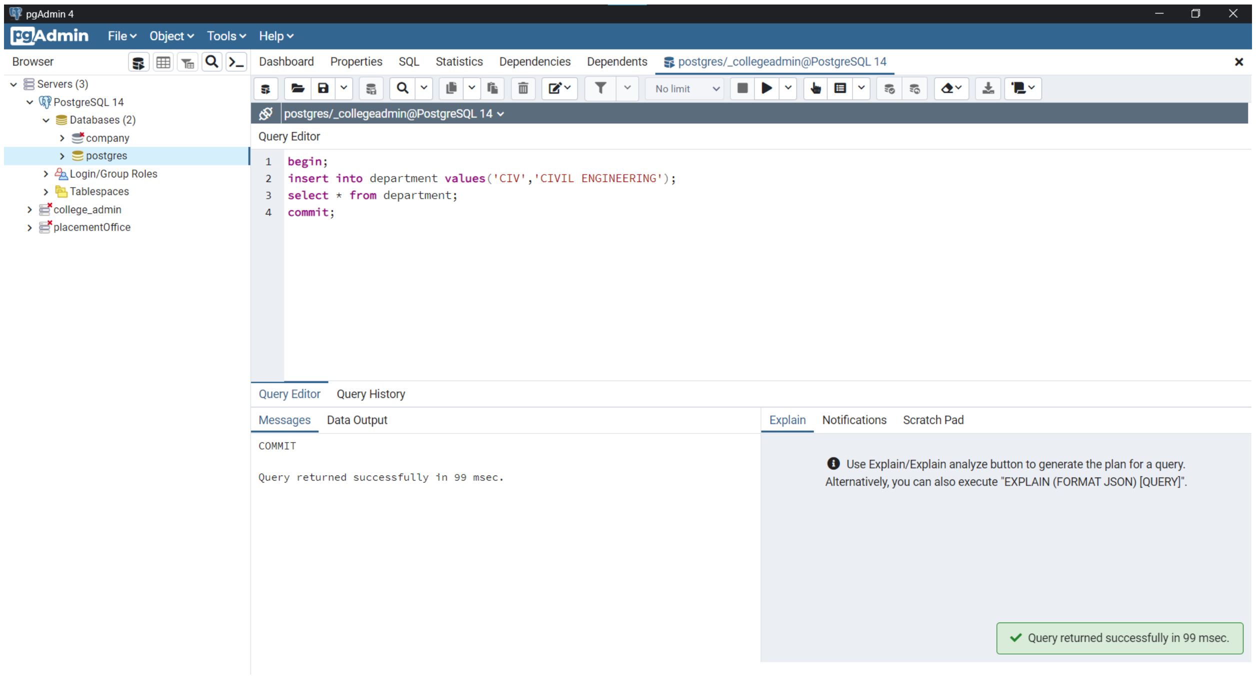


```
select * from department;
```

	dept_id [PK] character varying (50)	dept_name character varying (100)
1	CSE	COMPUTER SCIENCE
2	ECE	ELECTRONICS AND COMMUNICATION
3	MECH	MECHANICAL ENGINEERING

Successfully run. Total query runtime: 94 msec. 3 rows affected.

Viewing Department table by user_ companyrecruiter.



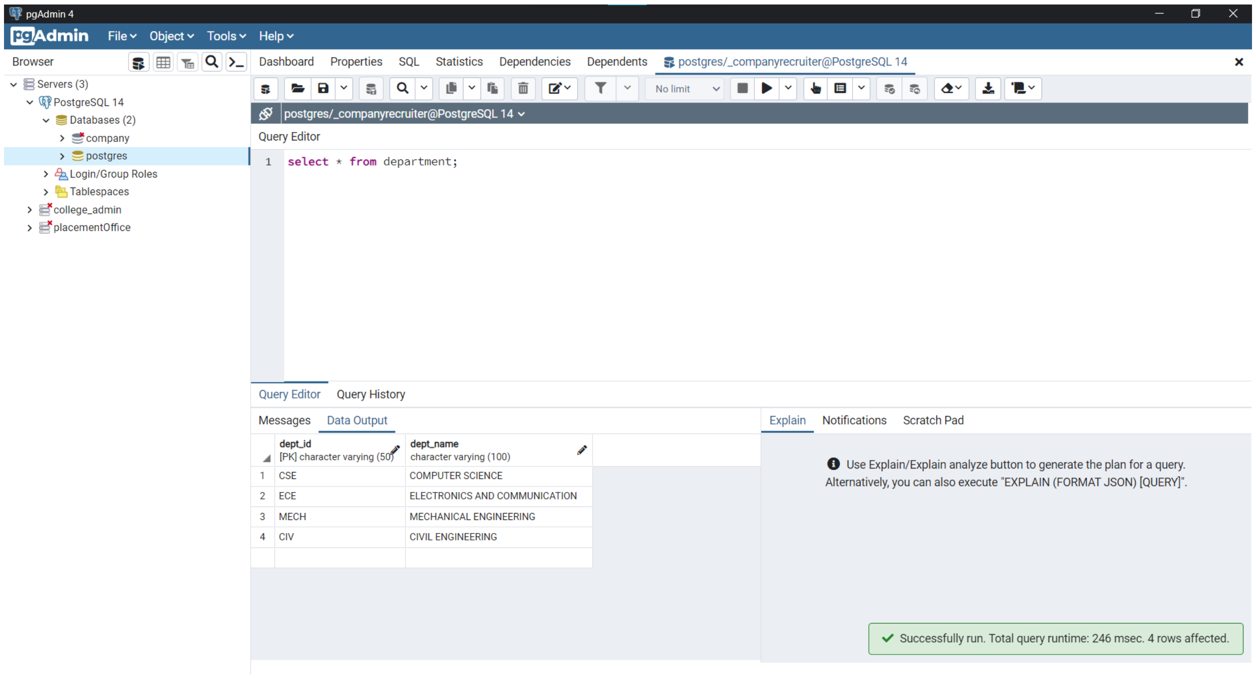
```
begin;
insert into department values('CIV','CIVIL ENGINEERING');
select * from department;
commit;
```

```
COMMIT

Query returned successfully in 99 msec.
```

Use Explain/Explain analyze button to generate the plan for a query. Alternatively, you can also execute "EXPLAIN (FORMAT JSON) [QUERY]".

Query returned successfully in 99 msec.

Inserting a record into Department table by user _collegeadmin.



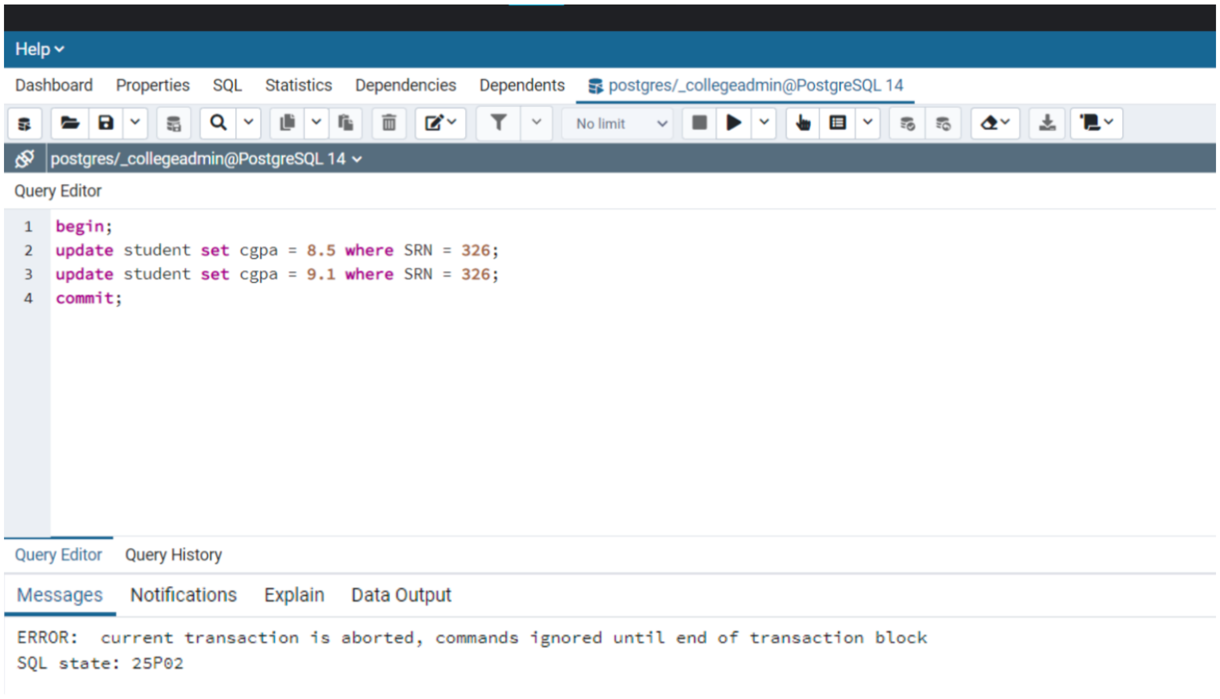
```
select * from department;
```

	dept_id [PK] character varying (50)	dept_name character varying (100)
1	CSE	COMPUTER SCIENCE
2	ECE	ELECTRONICS AND COMMUNICATION
3	MECH	MECHANICAL ENGINEERING
4	CIV	CIVIL ENGINEERING

Use Explain/Explain analyze button to generate the plan for a query. Alternatively, you can also execute "EXPLAIN (FORMAT JSON) [QUERY]".

Successfully run. Total query runtime: 246 msec. 4 rows affected.

Viewing Department table by user _companyrecruiter after updation. Change is only reflected after the second select query.



```sql
begin;
update student set cgpa = 8.5 where SRN = 326;
update student set cgpa = 9.1 where SRN = 326;
commit;
```

```
ERROR:  current transaction is aborted, commands ignored until end of transaction block
SQL state: 25P02
```

Not allowed – Because each update command is affecting the same record during the same transaction. Leads to inconsistency.

Contributions

Prachi Sengar - Simple queries and complex queries
Raeesa Tanseen - Transaction control and complex queries
Ria Singh - Transaction control and simple queries

Time Spent- 6 hrs