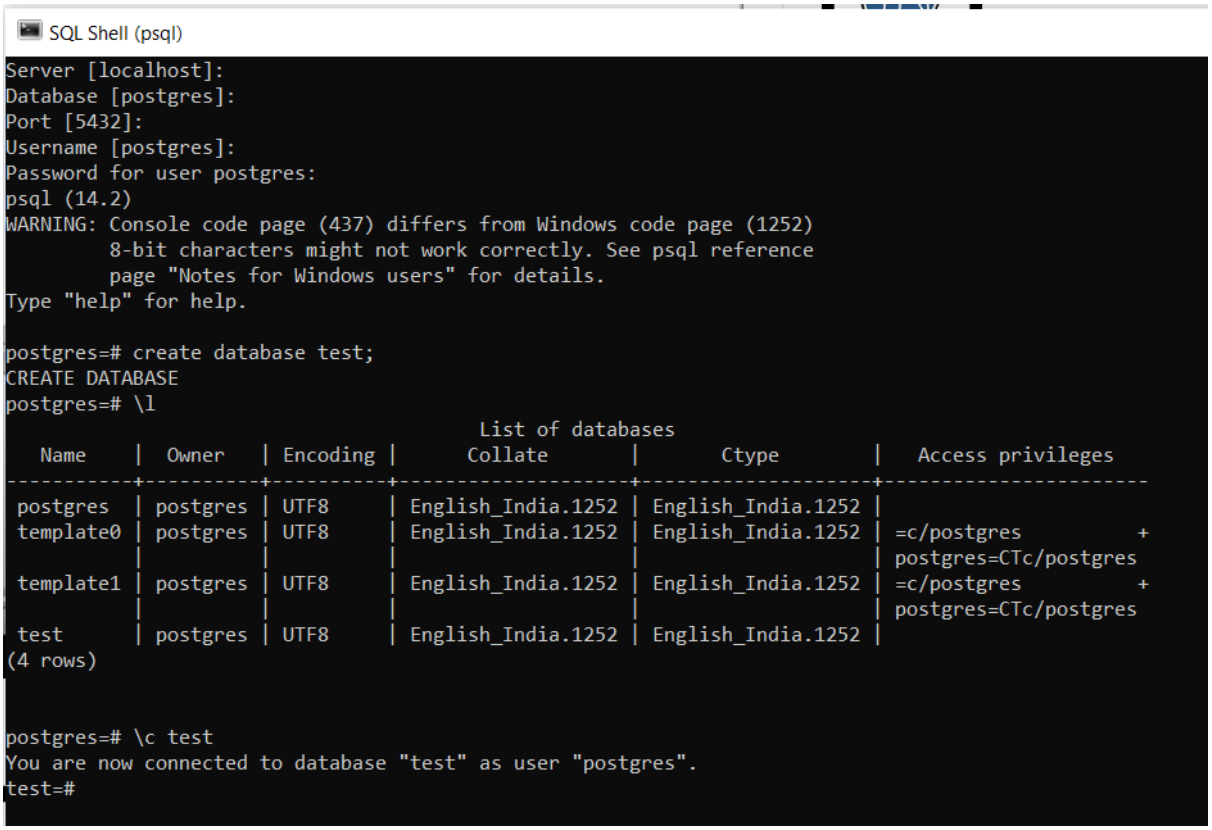
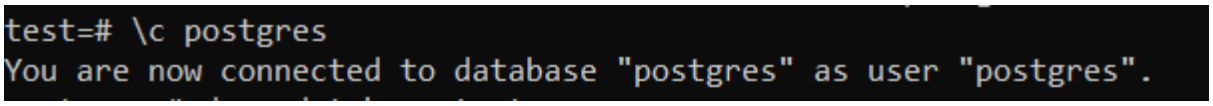


Using sql shell pressing enter till we reach password and type it.

Using commands in pictures we continue.

\l used to list all databases present.

\c test to select test database out of all available databases.

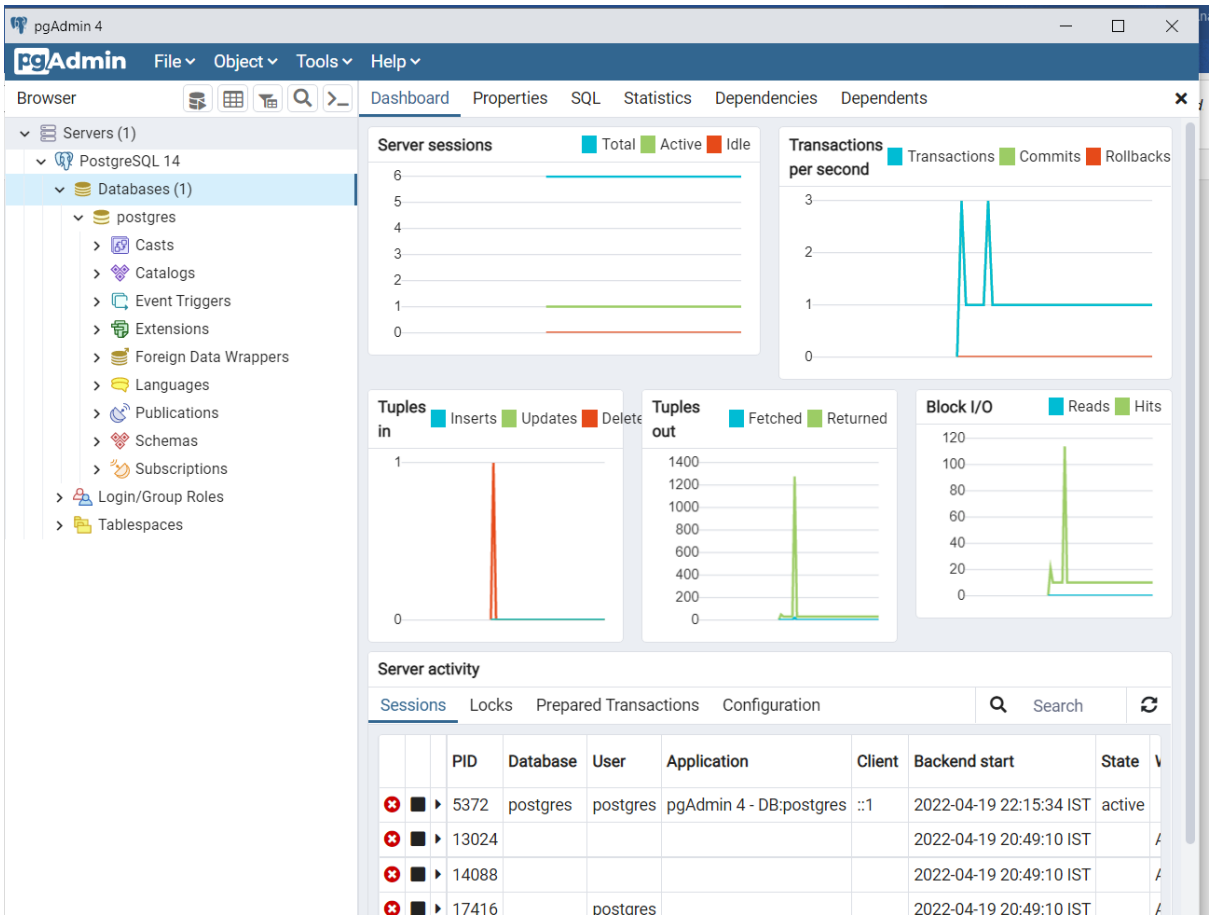


drop to remove table.

\q to disconnect all databases.

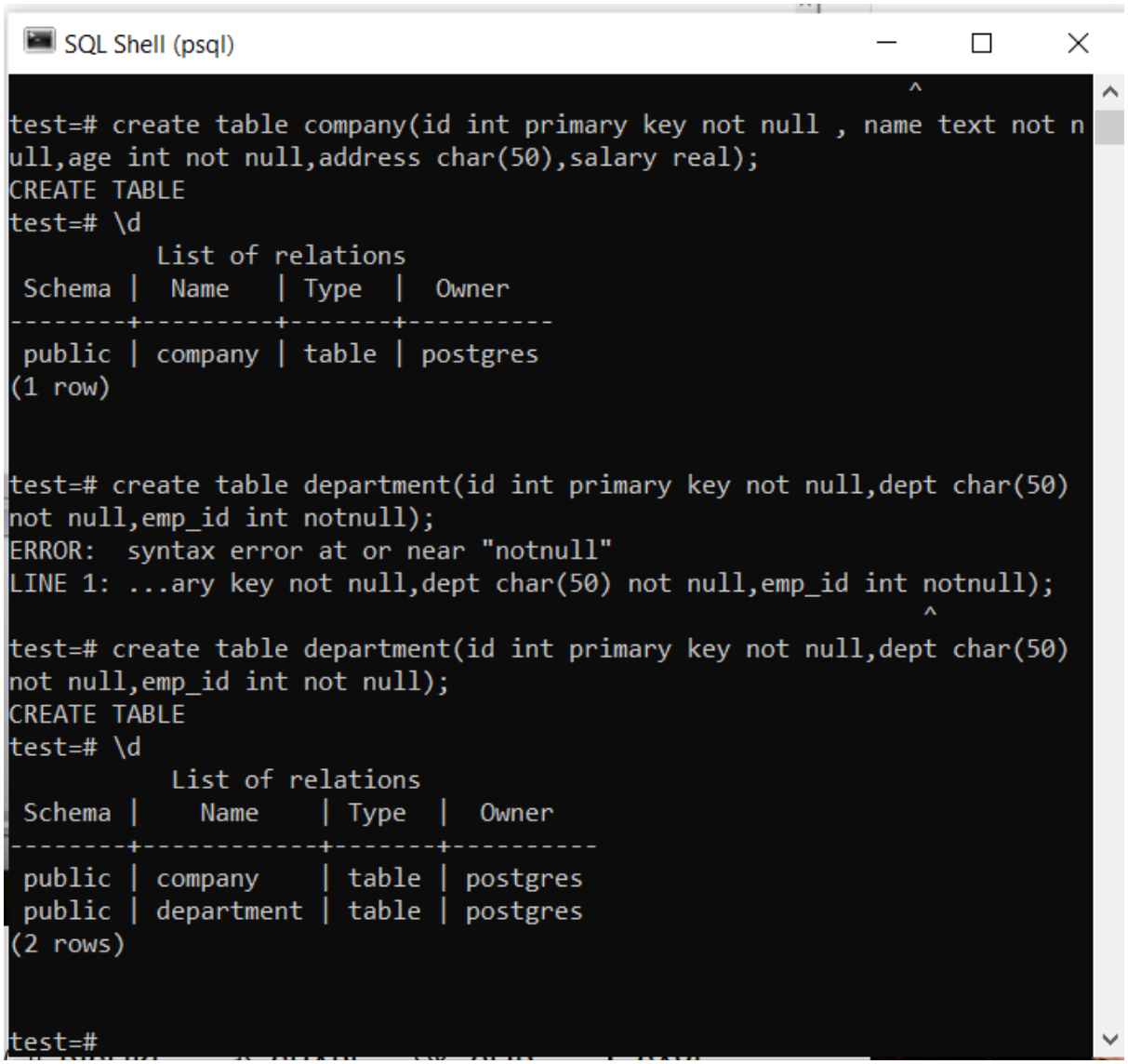
```
postgres=# \q
Press any key to continue . . .
```

Using pgadmin4 also it can be done right click on databases to add / drop new databases.



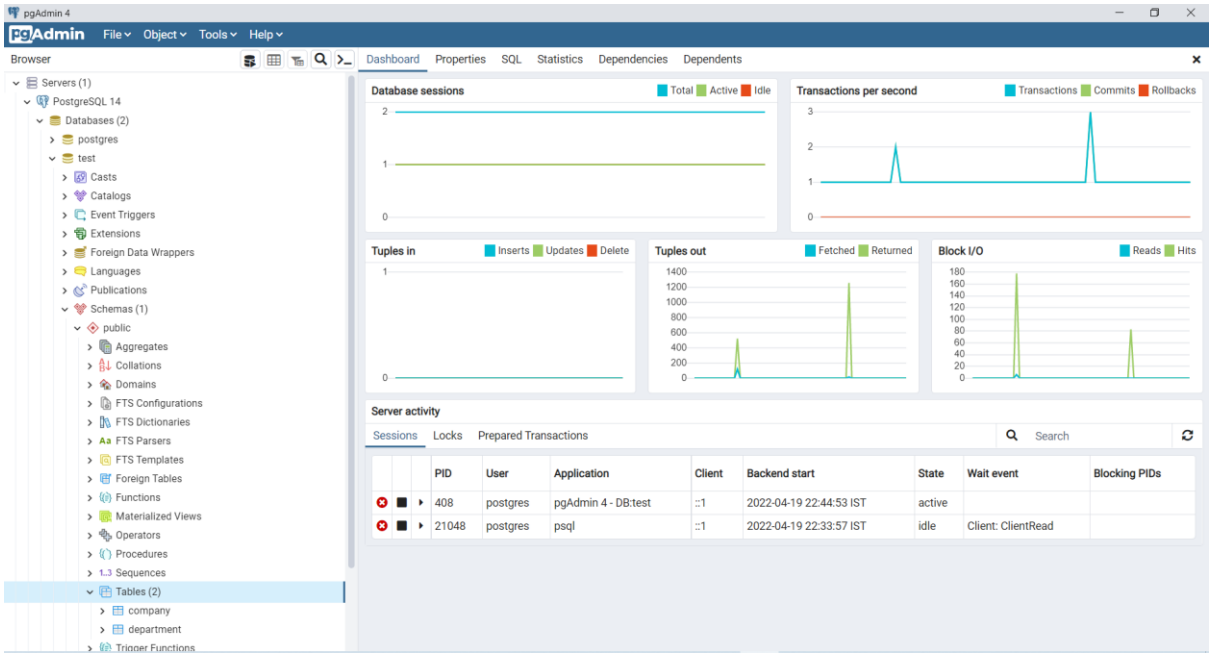

```
test=# create table company(id int primary key not null , name text not null,age int not null,address
 char(50),salary real);
CREATE TABLE
test=# \d
         List of relations
 Schema |  Name   | Type  |  Owner
--------+---------+-------+----------
 public | company | table | postgres
(1 row)
```

\d to describe table.

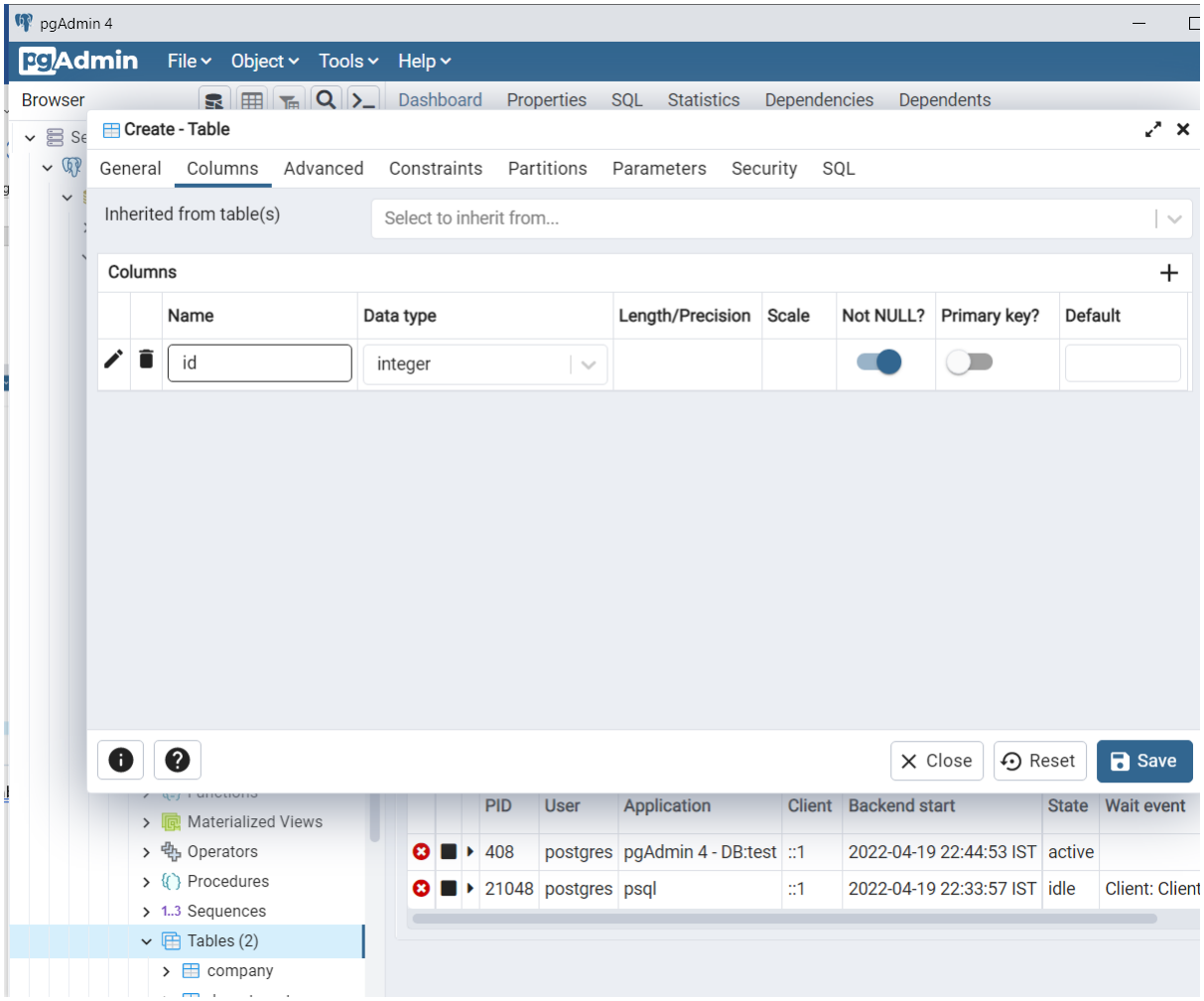


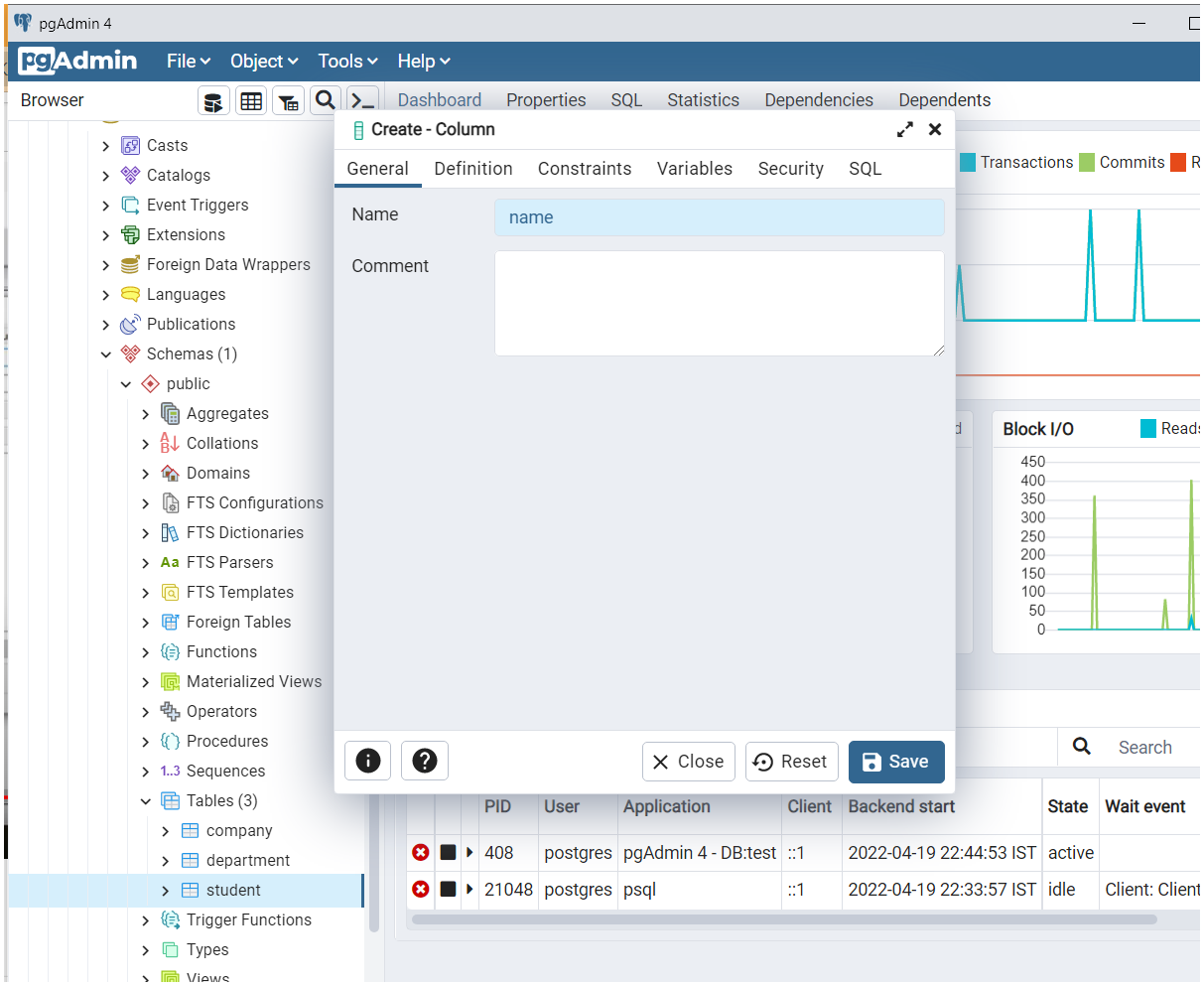
After doing all above steps table test will be available in pgadmin4. If not restart.

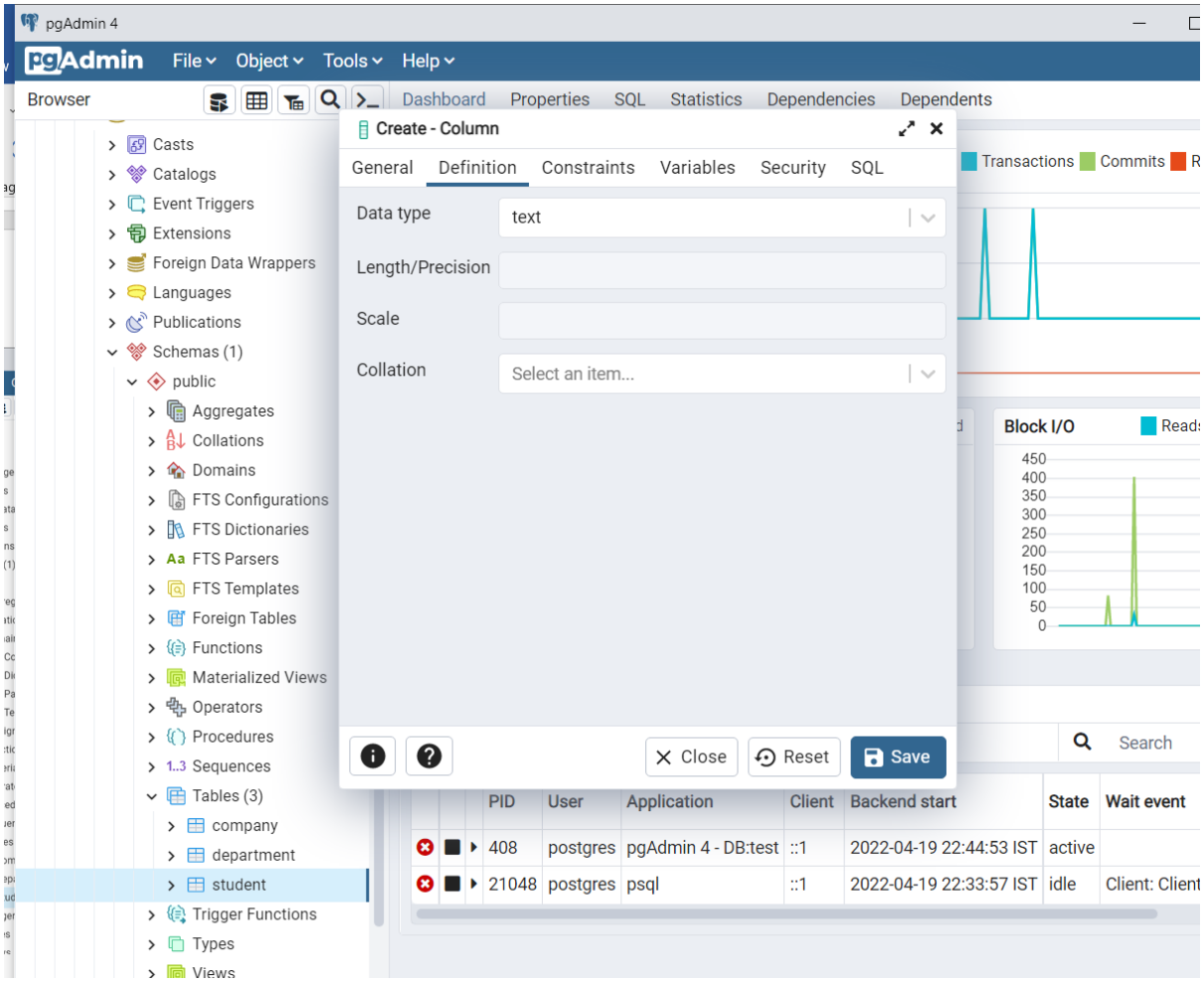
Belo how to reach to tables in pgadmin4.

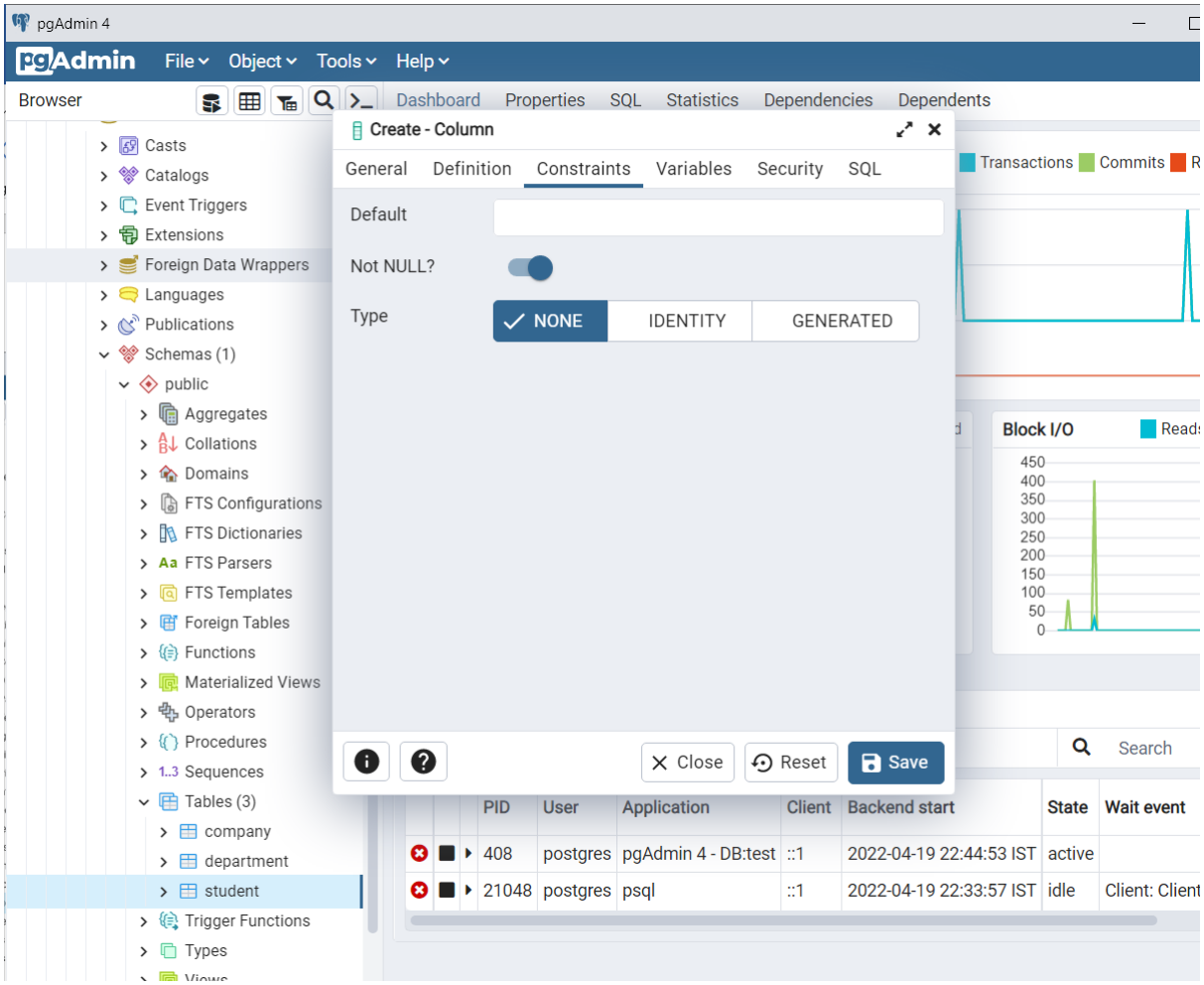


Right clicking on Tables(2) icon to create new table. Filling in general ,column where to fill name of column etc.

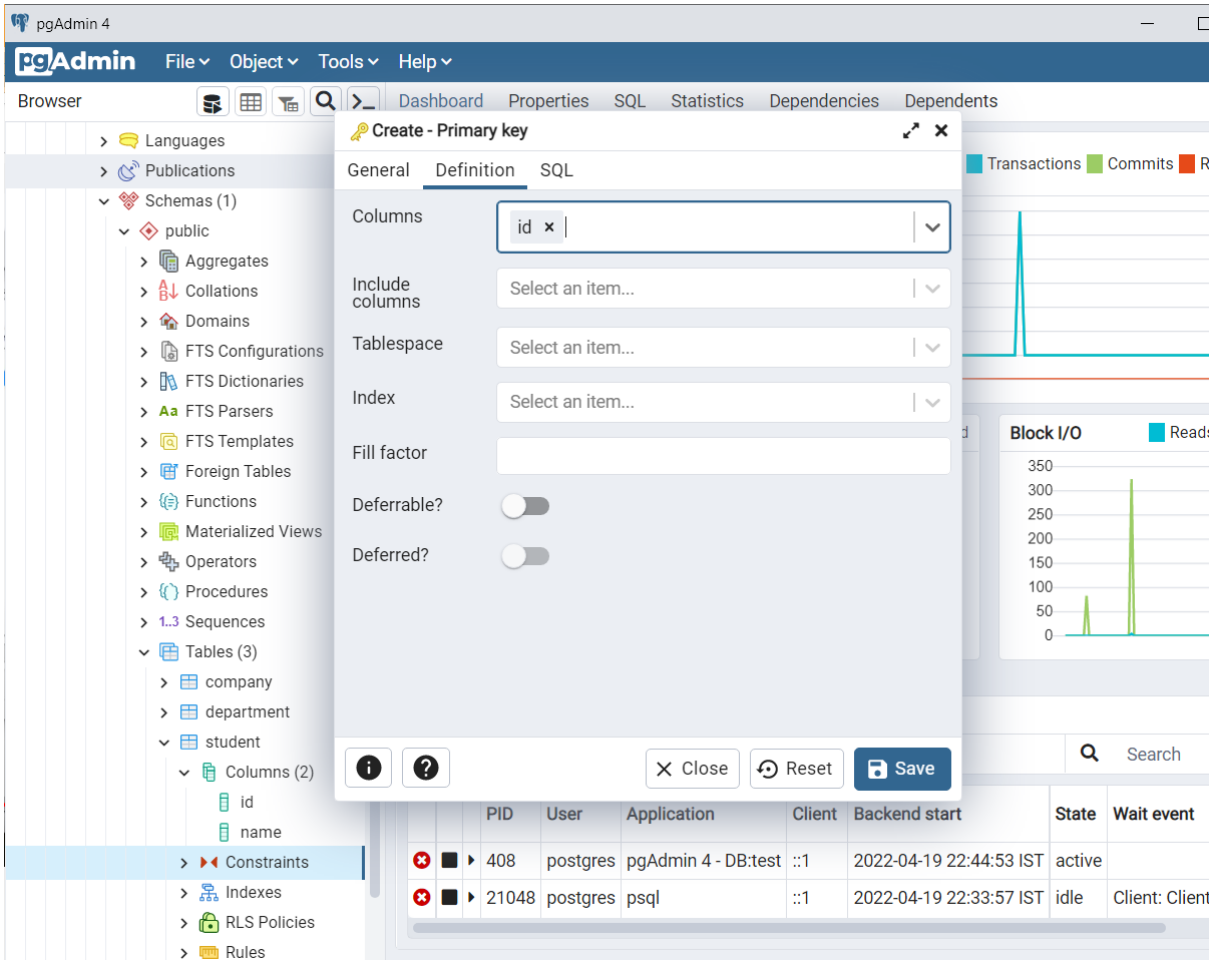



pgAdmin 4
pgAdmin
File
Object
Tools
Help
Browser
Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
Create - Column
General
Definition
Constraints
Variables
Security
SQL
Name
name
Comment
Close
Reset
Save
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas (1)
public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (3)
company
department
student
Trigger Functions
Types
Transactions
Commits
Block I/O
Search
PID
User
Application
Client
Backend start
State
Wait event
408
postgres
pgAdmin 4 - DB:test
::1
2022-04-19 22:44:53 IST
active
21048
postgres
psql
::1
2022-04-19 22:33:57 IST
idle


pgAdmin 4
pgAdmin
File
Object
Tools
Help
Browser
Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
Create - Column
General
Definition
Constraints
Variables
Security
SQL
Data type
text
Length/Precision
Scale
Collation
Select an item...
Close
Reset
Save
Transactions
Commits
Block I/O
Search
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas (1)
public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (3)
company
department
student
Trigger Functions
Types
Views
PID
User
Application
Client
Backend start
State
Wait event
408
postgres
pgAdmin 4 - DB:test
::1
2022-04-19 22:44:53 IST
active
21048
postgres
psql
::1
2022-04-19 22:33:57 IST
idle
Client: Client



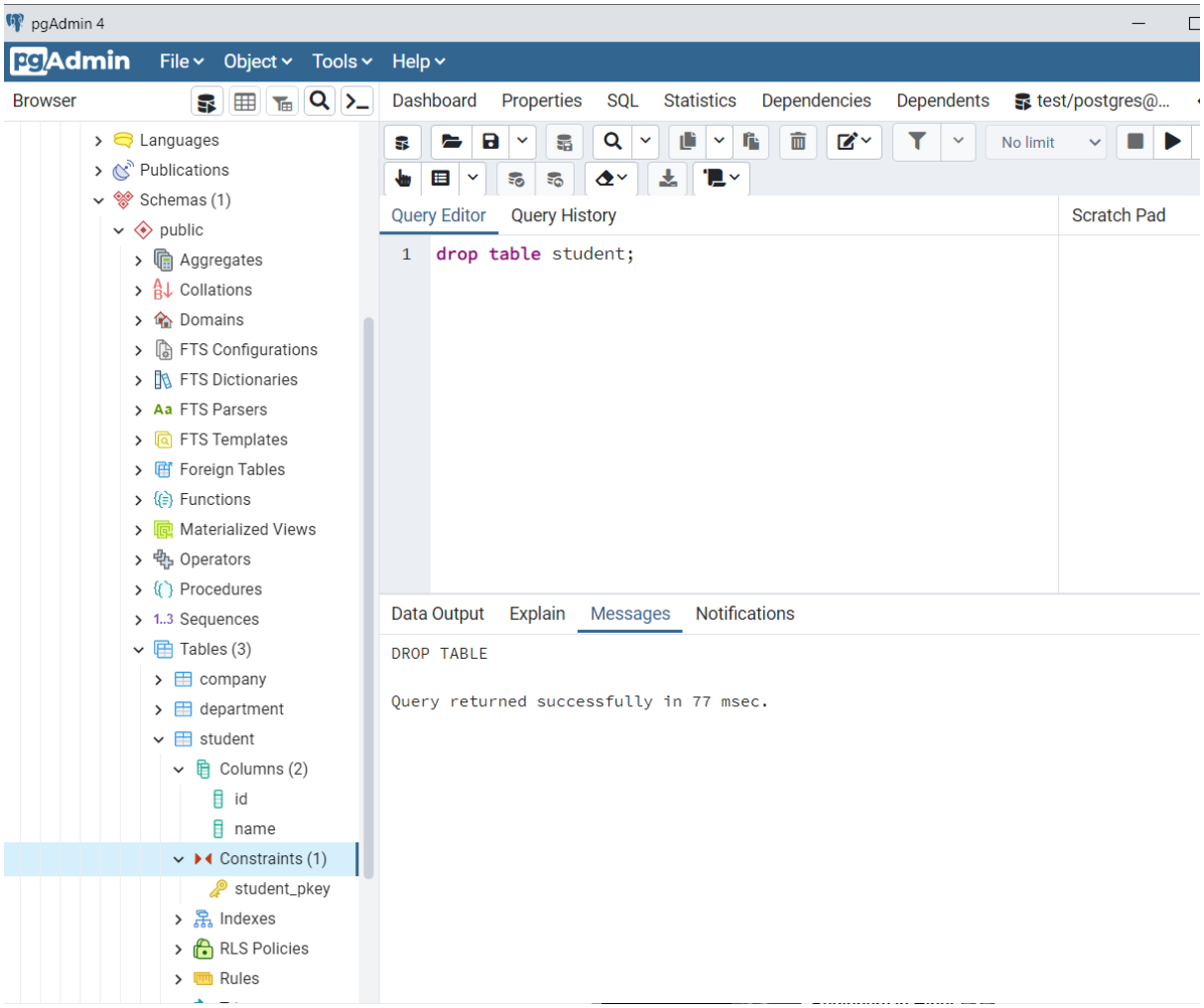
Id as primary key.



To write query use editor from tools.

Right click on tables red. To delete.

To delete a table use – drop table table-name;




```
test=# drop table student;
ERROR:  table "student" does not exist
test=#
```

As done after doing from query editor.

insert into company(id,name,age,address,salary)values(7,'Rahul',20,'Kolkata',50000)

test/postgres@PostgreSQL 14

Query Editor | Query History | Scratch Pad

```
1  insert into company(id,name,age,address,salary)values(7,'Rahul',20,'Kolkata',50000)
```

Data Output | Explain | Messages | Notifications

```
INSERT 0 1

Query returned successfully in 117 msec.
```

test/postgres@PostgreSQL 14

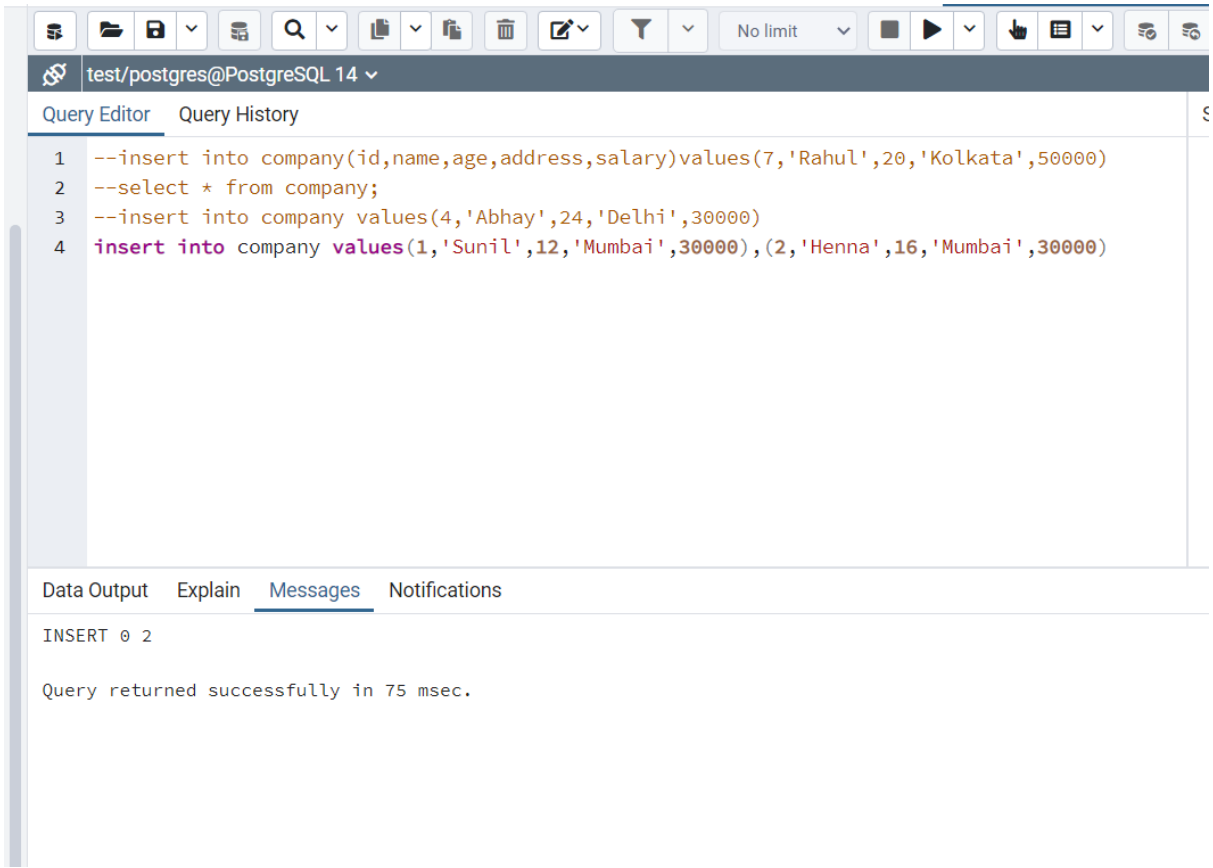
Query Editor | Query History

```
1  --insert into company(id,name,age,address,salary)values(7,'Rahul',20,'Kolkata',50000)
2  select * from company;
```

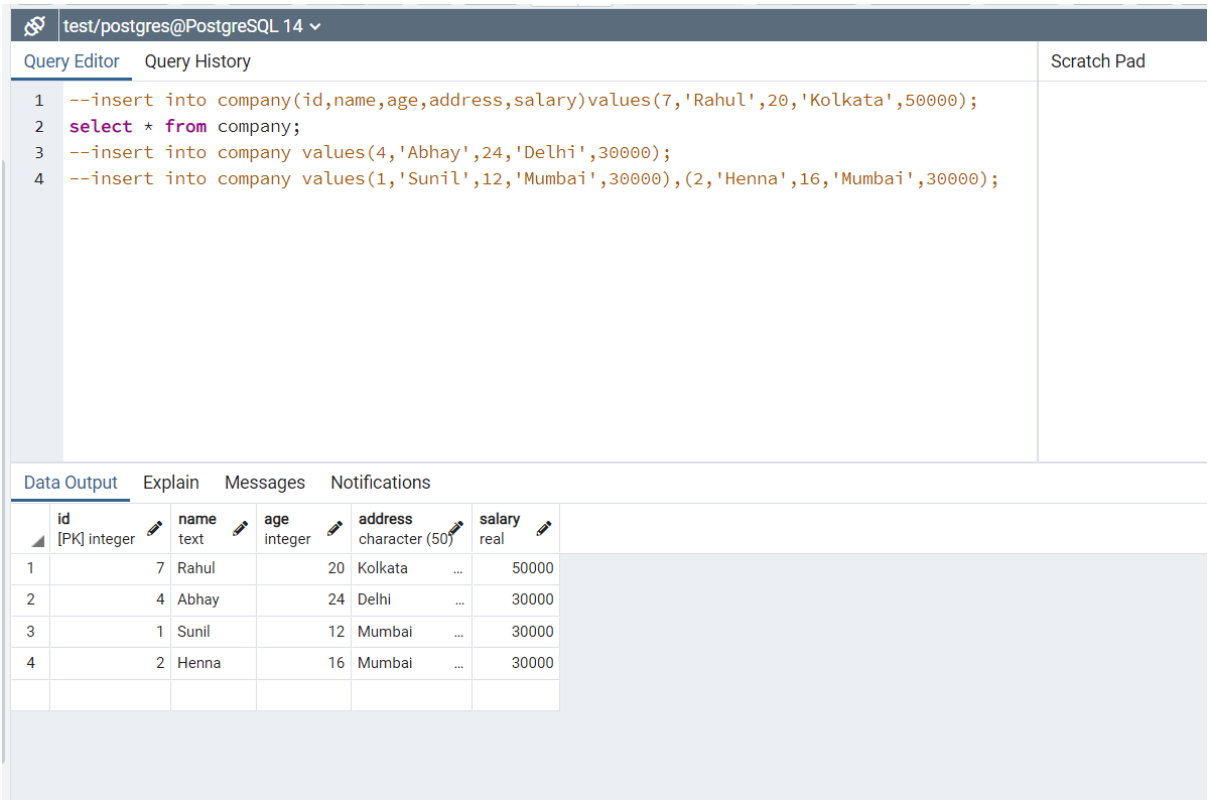
Data Output | Explain | Messages | Notifications

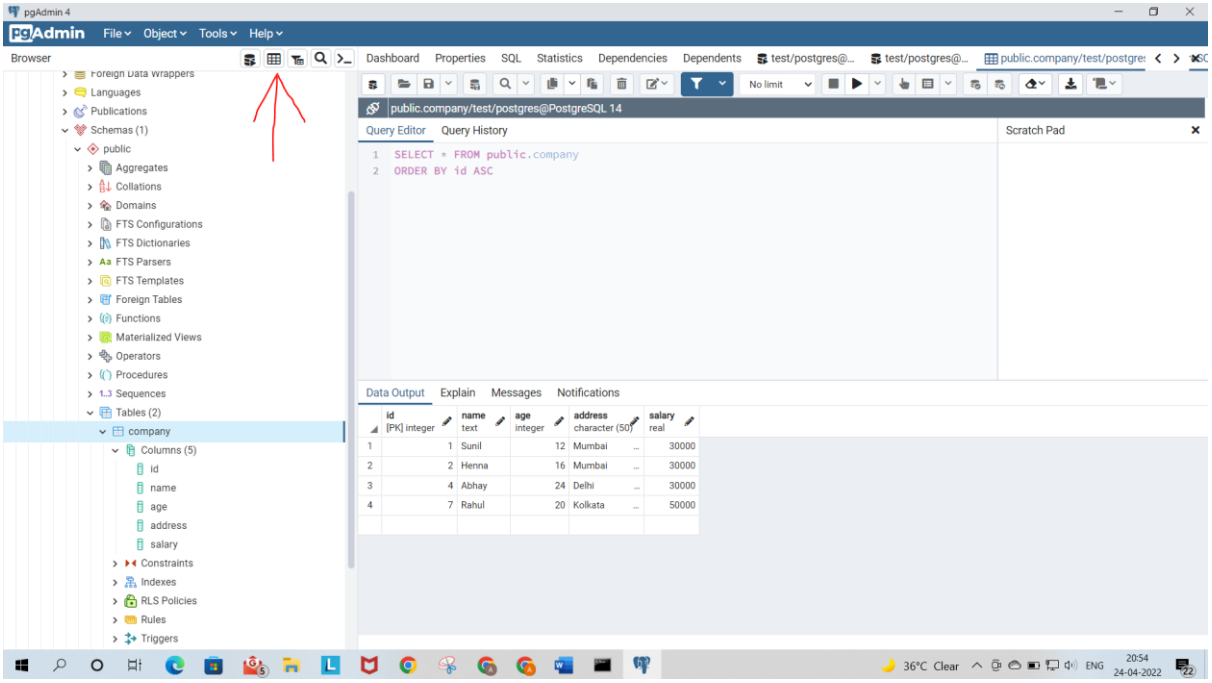
	id [PK] integer	name text	age integer	address character (50)	salary real
1	7	Rahul	20	Kolkata	50000


test/postgres@PostgreSQL 14
Query Editor
Query History
1 --insert into company(id,name,age,address,salary)values(7,'Rahul',20,'Kolkata',50000)
2 --select * from company;
3 insert into company values(4,'Abhay',24,'Delhi',30000)
Data Output
Explain
Messages
Notifications
INSERT 0 1
Query returned successfully in 87 msec.


No limit
test/postgres@PostgreSQL 14
Query Editor
Query History
1 --insert into company(id,name,age,address,salary)values(7,'Rahul',20,'Kolkata',50000)
2 --select * from company;
3 --insert into company values(4,'Abhay',24,'Delhi',30000)
4 insert into company values(1,'Sunil',12,'Mumbai',30000),(2,'Henna',16,'Mumbai',30000)
Data Output
Explain
Messages
Notifications
INSERT 0 2
Query returned successfully in 75 msec.

FORGOT SEMICOLON IN ABOVE STEPS NEED TO PUT SEMICOLON AFTER EACH QUERY.





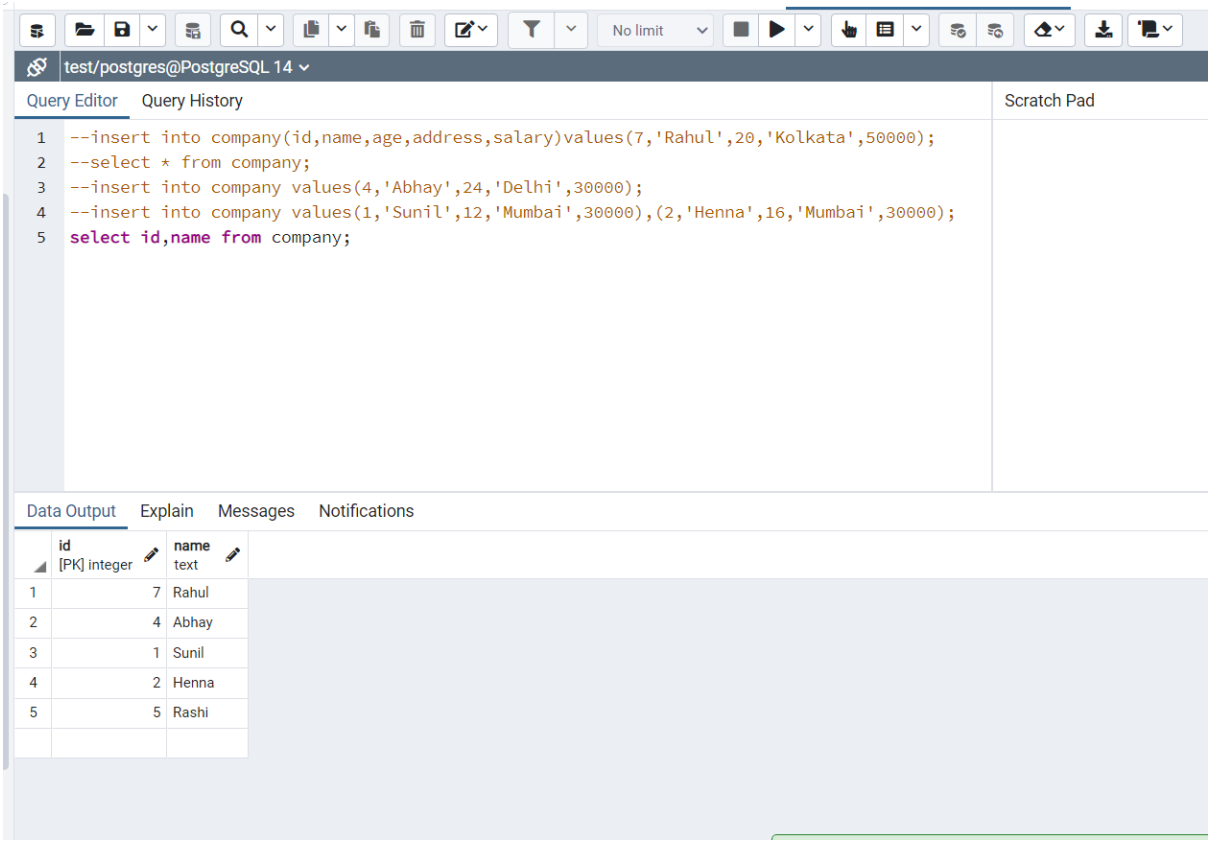
FOR VIEWING DATA IN THE TABLE SELECT COMPANY BY MOUSE AND CLICK ON VIEW DATA BUTTON MARKED BY ARROW. We Can also enter new data by clicking on the cells below.

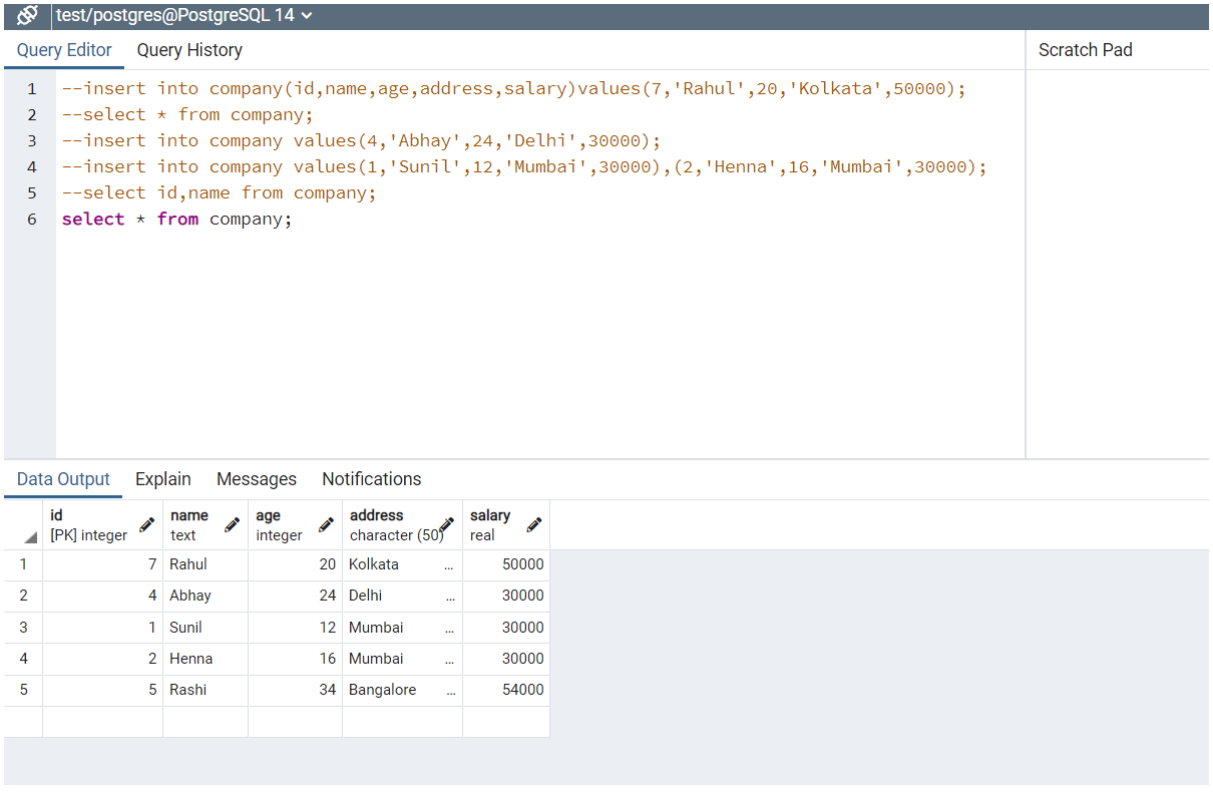
```
1  SELECT * FROM public.company
2  ORDER BY id ASC
```

Data Output | Explain | Messages | Notifications

	id [PK] integer	name text	age integer	address character (50)	salary real
1	1	Sunil	12	Mumbai	30000
2	2	Henna	16	Mumbai	30000
3	4	Abhay	24	Delhi	30000
4	7	Rahul	20	Kolkata	50000
5	5	Rashi	34	Bangalore	[null]

Make sure to save the data.

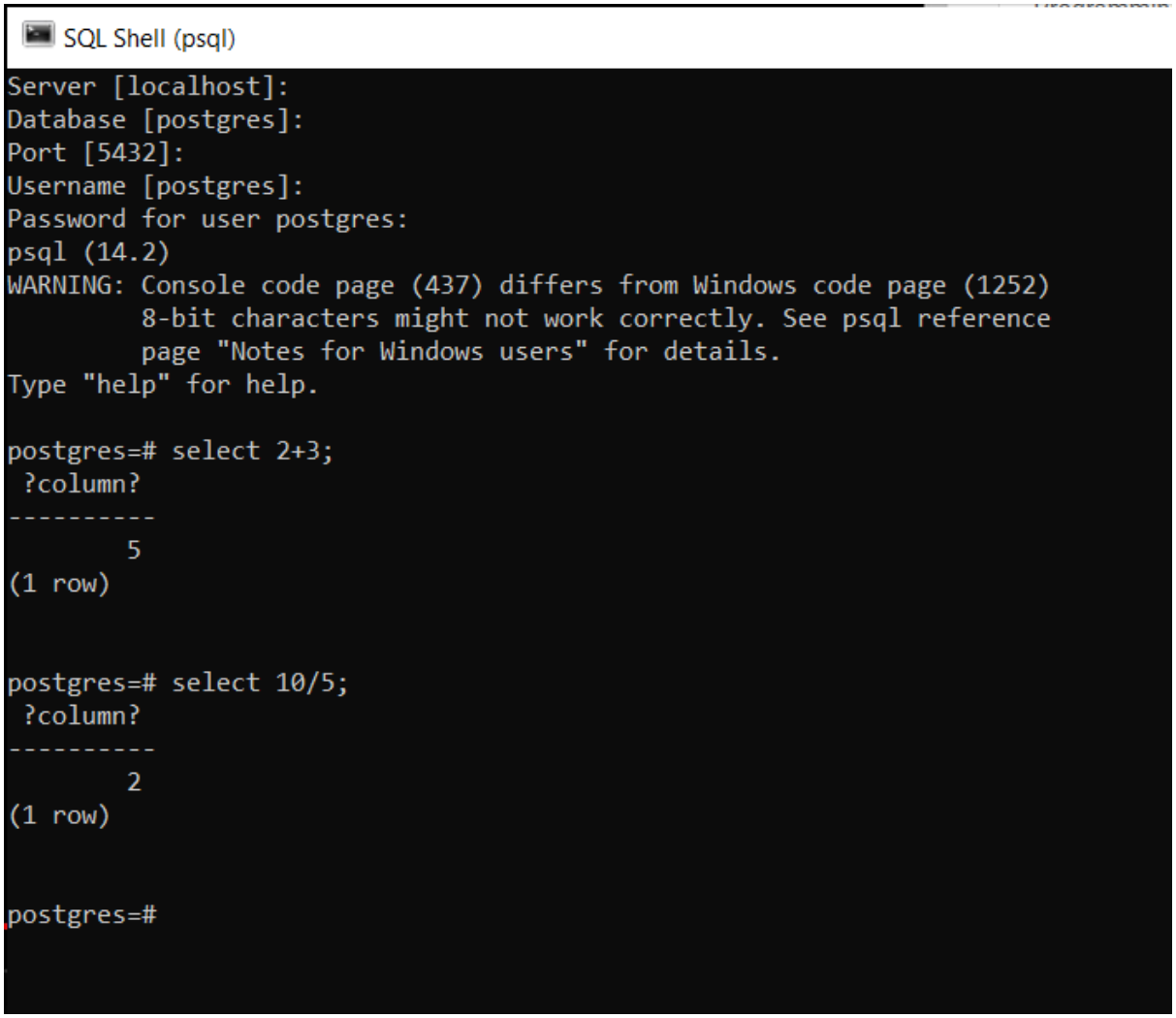

test/postgres@PostgreSQL 14
Query Editor
Query History
Scratch Pad
1 --insert into company(id,name,age,address,salary)values(7,'Rahul',20,'Kolkata',50000);
2 --select * from company;
3 --insert into company values(4,'Abhay',24,'Delhi',30000);
4 --insert into company values(1,'Sunil',12,'Mumbai',30000),(2,'Henna',16,'Mumbai',30000);
5 select id,name from company;
Data Output
Explain
Messages
Notifications
id [PK] integer
name text
1 7 Rahul
2 4 Abhay
3 1 Sunil
4 2 Henna
5 5 Rashi

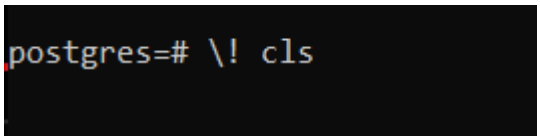

test/postgres@PostgreSQL 14
Query Editor
Query History
Scratch Pad
1 --insert into company(id,name,age,address,salary)values(7,'Rahul',20,'Kolkata',50000);
2 --select * from company;
3 --insert into company values(4,'Abhay',24,'Delhi',30000);
4 --insert into company values(1,'Sunil',12,'Mumbai',30000),(2,'Henna',16,'Mumbai',30000);
5 --select id,name from company;
6 select * from company;
Data Output
Explain
Messages
Notifications
id [PK] integer
name text
age integer
address character (50)
salary real
1 7 Rahul 20 Kolkata 50000
2 4 Abhay 24 Delhi 30000
3 1 Sunil 12 Mumbai 30000
4 2 Henna 16 Mumbai 30000
5 5 Rashi 34 Bangalore 54000

SQL Shell (psql)

```
Server [localhost]:
Database [postgres]:
Port [5432]:
Username [postgres]:
Password for user postgres:
psql (14.2)
WARNING: Console code page (437) differs from Windows code page (1252)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
Type "help" for help.

postgres=# \c test
You are now connected to database "test" as user "postgres".
test=# \d company
                  Table "public.company"
 Column  |     Type      | Collation | Nullable | Default
---------+---------------+-----------+----------+---------
 id      | integer       |           | not null |
 name    | text          |           | not null |
 age     | integer       |           | not null |
 address | character(50) |           |          |
 salary  | real          |           |          |
Indexes:
    "company_pkey" PRIMARY KEY, btree (id)
```

```
test=#  select * from company;
 id | name  | age |                    address                     | salary
----+-------+-----+------------------------------------------------+--------
  7 | Rahul |  20 | Kolkata                                        |  50000
  4 | Abhay |  24 | Delhi                                          |  30000
  1 | Sunil |  12 | Mumbai                                         |  30000
  2 | Henna |  16 | Mumbai                                         |  30000
  5 | Rashi |  34 | Bangalore                                      |  54000
(5 rows)


test=# select * from department;
 id | dept | emp_id
----+------+--------
(0 rows)


test=# select * from company;
 id | name  | age |                    address                     | salary
----+-------+-----+------------------------------------------------+--------
  7 | Rahul |  20 | Kolkata                                        |  50000
  4 | Abhay |  24 | Delhi                                          |  30000
  1 | Sunil |  12 | Mumbai                                         |  30000
  2 | Henna |  16 | Mumbai                                         |  30000
  5 | Rashi |  34 | Bangalore                                      |  54000
(5 rows)


test=#
```





Above command to clear screen.

```
postgres=# select 2^3 ;
 ?column?
----------
        8
(1 row)


postgres=# select 14%4;
 ?column?
----------
        2
(1 row)


postgres=#
```

SQL Shell (psql)

```
postgres=# select |/25;
 ?column?
----------
        5
(1 row)


postgres=#
```

Square root of 25 is 5.

SQL Shell (psql)

```
postgres=# select |/25;
 ?column?
----------
        5
(1 row)


postgres=# select ||/27;
 ?column?
----------
        3
(1 row)


postgres=#
```

```
test=# select name from company where age = 34;
 name
-------
 Rashi
(1 row)
```

```
test=# select name from company where age != 34;
 name
-------
 Rahul
 Abhay
 Sunil
 Henna
(4 rows)
```

```
test=# select * from company where age <> 34;
 id | name  | age |                    address                    | salary
----+-------+-----+-----------------------------------------------+--------
  7 | Rahul |  20 | Kolkata                                       |  50000
  4 | Abhay |  24 | Delhi                                         |  30000
  1 | Sunil |  12 | Mumbai                                        |  30000
  2 | Henna |  16 | Mumbai                                        |  30000
(4 rows)
```

<> is equivalent to (not equal to) or != .

```
test=# select * from company where age > 12;
 id | name  | age |                    address                    | salary
----+-------+-----+-----------------------------------------------+--------
  7 | Rahul |  20 | Kolkata                                       |  50000
  4 | Abhay |  24 | Delhi                                         |  30000
  2 | Henna |  16 | Mumbai                                        |  30000
  5 | Rashi |  34 | Bangalore                                     |  54000
(4 rows)
```

For greater than or equal to use → >= .

```
test=# select * from company where age>13 and salary>2000;
 id | name  | age |                    address                    | salary
----+-------+-----+-----------------------------------------------+--------
  7 | Rahul |  20 | Kolkata                                       |  50000
  4 | Abhay |  24 | Delhi                                         |  30000
  2 | Henna |  16 | Mumbai                                        |  30000
  5 | Rashi |  34 | Bangalore                                     |  54000
(4 rows)
```

```
test=# select * from company where age>13 or  salary>2000;
 id | name  | age |                    address                    | salary
----+-------+-----+-----------------------------------------------+--------
  7 | Rahul |  20 | Kolkata                                       |  50000
  4 | Abhay |  24 | Delhi                                         |  30000
  1 | Sunil |  12 | Mumbai                                        |  30000
  2 | Henna |  16 | Mumbai                                        |  30000
  5 | Rashi |  34 | Bangalore                                     |  54000
(5 rows)
```

```
test=# select * from company where salary is null;
 id | name | age | address | salary
----+------+-----+---------+--------
(0 rows)
```

```
test=# select * from company where name like 'R%';
 id | name  | age |                    address                    | salary
----+-------+-----+-----------------------------------------------+--------
  7 | Rahul |  20 | Kolkata                                       |  50000
  5 | Rashi |  34 | Bangalore                                     |  54000
(2 rows)
```

```
test=# select * from company where name like '%a%';
 id | name  | age |                    address                    | salary
----+-------+-----+-----------------------------------------------+--------
  7 | Rahul |  20 | Kolkata                                       |  50000
  4 | Abhay |  24 | Delhi                                         |  30000
  2 | Henna |  16 | Mumbai                                        |  30000
  5 | Rashi |  34 | Bangalore                                     |  54000
(4 rows)
```

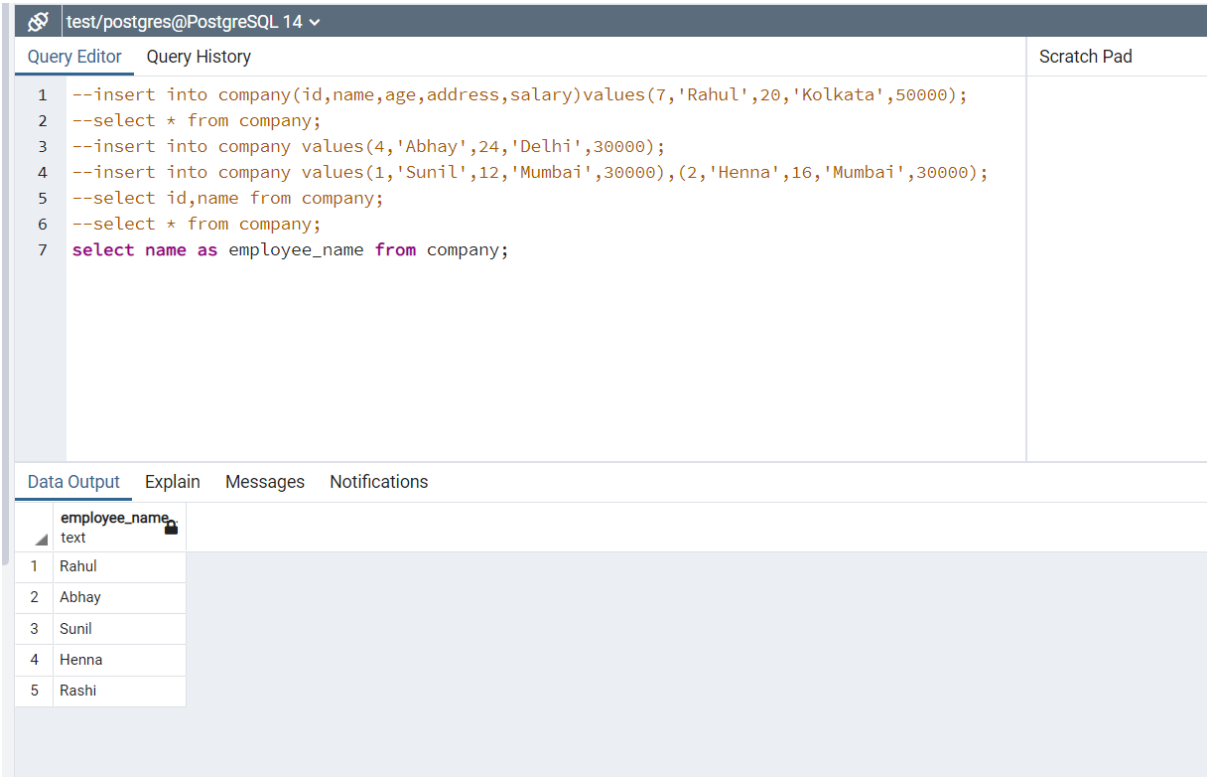
```
test=# select * from company where age in ('20','34','16');
 id | name  | age |                  address                  | salary
----+-------+-----+-------------------------------------------+--------
  7 | Rahul |  20 | Kolkata                                   |  50000
  2 | Henna |  16 | Mumbai                                    |  30000
  5 | Rashi |  34 | Bangalore                                 |  54000
(3 rows)
```

Above query means select row where age is 20 or 34 or 16.

```
test=# select * from company where age between 20 and 34;
 id | name  | age |                  address                  | salary
----+-------+-----+-------------------------------------------+--------
  7 | Rahul |  20 | Kolkata                                   |  50000
  4 | Abhay |  24 | Delhi                                     |  30000
  5 | Rashi |  34 | Bangalore                                 |  54000
(3 rows)
```

--- used for single line comment.

/*----------*/ used for multiline comment.



test/postgres@PostgreSQL 14

Query Editor | Query History

```
--insert into company(id,name,age,address,salary)values(7,'Rahul',20,'Kolkata',50000);
--select * from company;
--insert into company values(4,'Abhay',24,'Delhi',30000);
--insert into company values(1,'Sunil',12,'Mumbai',30000),(2,'Henna',16,'Mumbai',30000);
--select id,name from company;
--select * from company;
--select name as employee_name from company;
select count(*) from company;
```

Data Output | Explain | Messages | Notifications

	count bigint
1	5

test/postgres@PostgreSQL 14

Query Editor | Query History

```
--insert into company(id,name,age,address,salary)values(7,'Rahul',20,'Kolkata',50000);
--select * from company;
--insert into company values(4,'Abhay',24,'Delhi',30000);
--insert into company values(1,'Sunil',12,'Mumbai',30000),(2,'Henna',16,'Mumbai',30000);
--select id,name from company;
--select * from company;
--select name as employee_name from company;
--select count(*) from company;
select max(age) from company;
```

Data Output | Explain | Messages | Notifications

	max integer
1	34

```
--insert into company(id,name,age,address,salary)values(7,'Rahul',20,'Kolkata',50000);
--select * from company;
--insert into company values(4,'Abhay',24,'Delhi',30000);
--insert into company values(1,'Sunil',12,'Mumbai',30000),(2,'Henna',16,'Mumbai',30000);
--select id,name from company;
--select * from company;
--select name as employee_name from company;
--select count(*) from company;
select min(age) from company;
```

Data Output | Explain | Messages | Notifications

	min integer
1	12

test/postgres@PostgreSQL 14

Query Editor | Query History | Scratch P

```
--insert into company(id,name,age,address,salary)values(7,'Rahul',20,'Kolkata',50000);
--select * from company;
--insert into company values(4,'Abhay',24,'Delhi',30000);
--insert into company values(1,'Sunil',12,'Mumbai',30000),(2,'Henna',16,'Mumbai',30000);
--select id,name from company;
--select * from company;
--select name as employee_name from company;
--select count(*) from company;
--select min(age) from company;
select sum(salary) from company;
```

Data Output | Explain | Messages | Notifications

	sum real
1	194000

test/postgres@PostgreSQL 14

Query Editor | Query History

```
1  --insert into company(id,name,age,address,salary)values(7,'Rahul',20,'Kolkata',50000);
2  --select * from company;
3  --insert into company values(4,'Abhay',24,'Delhi',30000);
4  --insert into company values(1,'Sunil',12,'Mumbai',30000),(2,'Henna',16,'Mumbai',30000);
5  --select id,name from company;
6  --select * from company;
7  --select name as employee_name from company;
8  --select count(*) from company;
9  --select min(age) from company;
10 select avg(salary) from company;
11
```

Data Output | Explain | Messages | Notifications

	avg double precision
1	38800

test/postgres@PostgreSQL 14

Query Editor | Query History

```
1  --insert into company(id,name,age,address,salary)values(7,'Rahul',20,'Kolkata',50000);
2  --select * from company;
3  --insert into company values(4,'Abhay',24,'Delhi',30000);
4  --insert into company values(1,'Sunil',12,'Mumbai',30000),(2,'Henna',16,'Mumbai',30000);
5  --select id,name from company;
6  --select * from company;
7  --select name as employee_name from company;
8  --select count(*) from company;
9  --select min(age) from company;
10 --select avg(salary) from company;
11 select current_timestamp;
```

Data Output | Explain | Messages | Notifications

	current_timestamp timestamp with time zone
1	2022-04-24 21:35:28.636283+05:30

Above query to get current time,date etc.

product/postgres@PostgreSQL 14

Query Editor | Query History

```
1  create table products(
2  product_no integer,
3  name text,
4  price numeric CHECK (price>0)
5  );
```

Data Output | Explain | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 134 msec.
```

CHECK constraint to check for price always > 0.

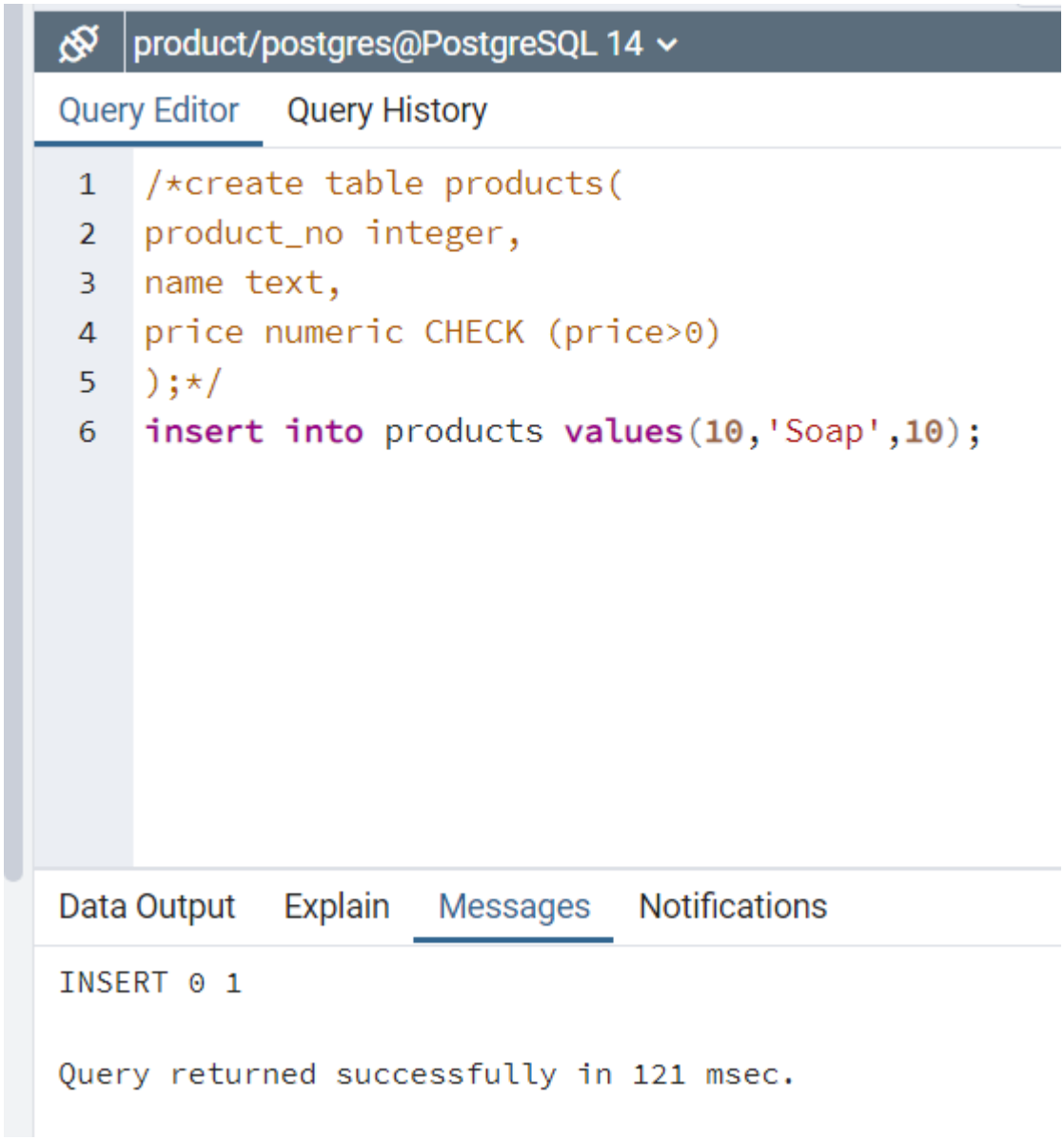
product/postgres@PostgreSQL 14

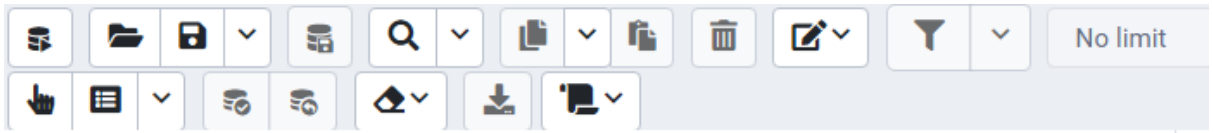
Query Editor | Query History

```
1  /*create table products(
2  product_no integer,
3  name text,
4  price numeric CHECK (price>0)
5  );*/
6  insert into products values(10,'Soap',-10);
```

Data Output | Explain | Messages | Notifications

```
ERROR:  new row for relation "products" violates check constraint "products_price_check"
DETAIL:  Failing row contains (10, Soap, -10).
SQL state: 23514
```


product/postgres@PostgreSQL 14
Query Editor
Query History
1 /*create table products(
2 product_no integer,
3 name text,
4 price numeric CHECK (price>0)
5);*/
6 insert into products values(10,'Soap',10);
Data Output
Explain
Messages
Notifications
INSERT 0 1
Query returned successfully in 121 msec.



Query Editor | Query History

```
1  create table product_2(
2  product_no integer,
3  name text,
4  price numeric CHECK(price>0),
5  discounted_price numeric CHECK(discounted_price>0),
6  CHECK(price>discounted_price)
7  )
```

Data Output | Explain | Messages | Notifications

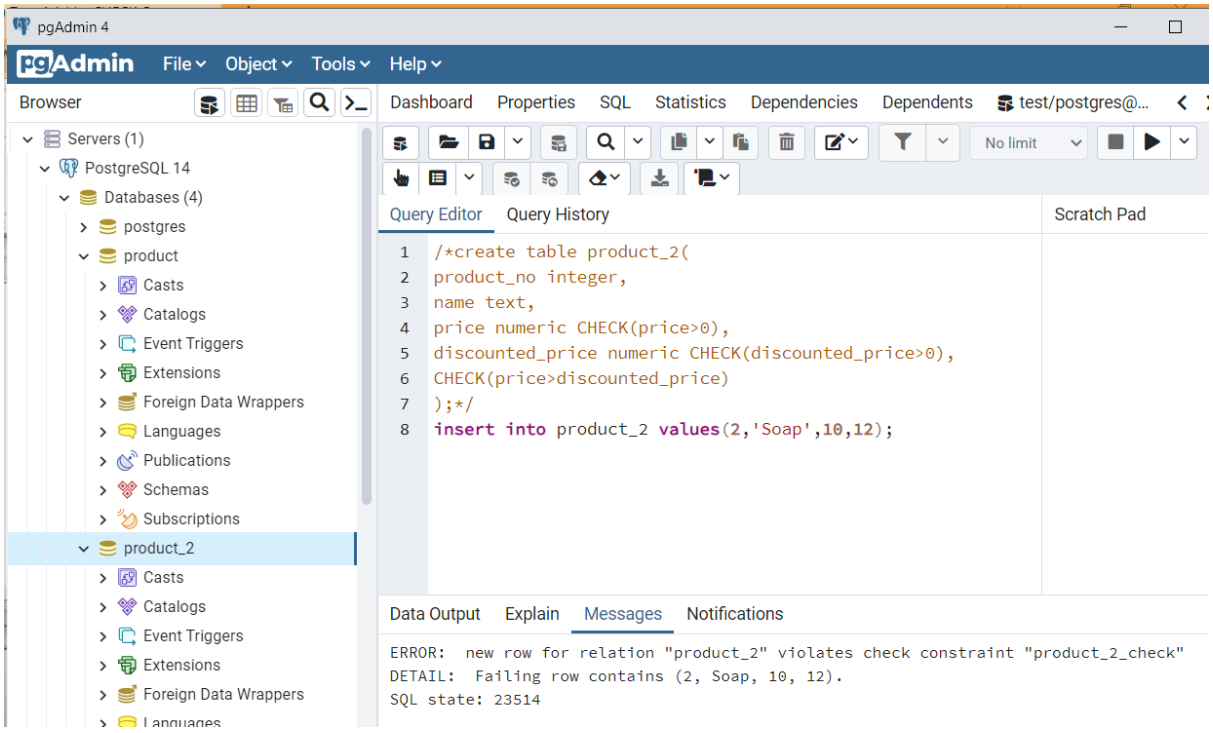
```
CREATE TABLE

Query returned successfully in 179 msec.
```

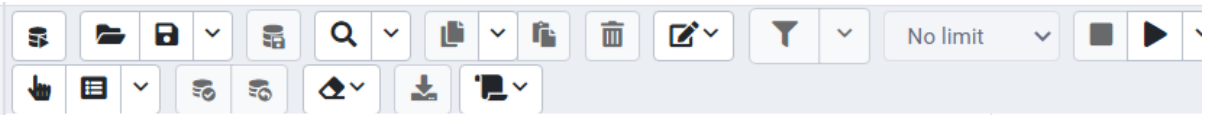
Query Editor | Query History | Scratch

```
/*create table product_2(
product_no integer,
name text,
price numeric CHECK(price>0),
discounted_price numeric CHECK(discounted_price>0),
CHECK(price>discounted_price)
);*/
insert into product_2 values(2,'Soap',10,-4);
```

Data Output | Explain | Messages | Notifications

```
ERROR:  new row for relation "product_2" violates check constraint
"product_2_discounted_price_check"
DETAIL:  Failing row contains (2, Soap, 10, -4).
SQL state: 23514
```





Query Editor | Query History | Scratch Pad

```
1  /*create table product_2(
2  product_no integer,
3  name text,
4  price numeric CHECK(price>0),
5  discounted_price numeric CHECK(discounted_price>0),
6  CHECK(price>discounted_price)
7  );*/
8  insert into product_2 values(2,'Soap',10,8);
```

Data Output | Explain | Messages | Notifications

```
INSERT 0 1

Query returned successfully in 99 msec.
```

Query Editor | Query History | Scratch Pa

```
1  create table product_3(
2  product_no integer NOT NULL,
3      name text NOT NULL,
4      price numeric NOT NULL CHECK(price>0)
5  );
```

Data Output | Explain | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 113 msec.
```

Query Editor | Query History | Scratch Pad

```
1  /*create table product_3(
2  product_no integer NOT NULL,
3      name text NOT NULL,
4      price numeric NOT NULL CHECK(price>0)
5  );*/
6  insert into product_3 values(12,NULL,-5);
```

Data Output | Explain | Messages | Notifications

```
ERROR:  null value in column "name" of relation "product_3" violates not-null constraint
DETAIL:  Failing row contains (12, null, -5).
SQL state: 23502
```

Query Editor | Query History | Scratch Pad

```
1  /*create table product_3(
2  product_no integer NOT NULL,
3      name text NOT NULL,
4      price numeric NOT NULL CHECK(price>0)
5  );*/
6  insert into product_3 values(12,'Soap',67);
```

Data Output | Explain | Messages | Notifications

```
INSERT 0 1

Query returned successfully in 111 msec.
```

Query Editor | Query History | Scratch Pad

```
1  create table product_4(
2  product_no integer,
3  name text,
4  price numeric,
5  UNIQUE(product_no,name)
6  );
```

Data Output | Explain | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 195 msec.
```

Query Editor | Query History | Scratch Pad

```
1  /*create table product_4_1(
2  product_no integer UNIQUE,
3  name text UNIQUE,
4  price numeric,
5  UNIQUE(product_no,name)
6  );  */
7  --insert into product_4_1 values(3,'Plate',100);
8  insert into product_4 values(7,'Cup',100);
```

Data Output | Explain | Messages | Notifications

```
ERROR:  duplicate key value violates unique constraint "product_4_product_no_name_key"
DETAIL:  Key (product_no, name)=(7, Cup) already exists.
SQL state: 23505
```

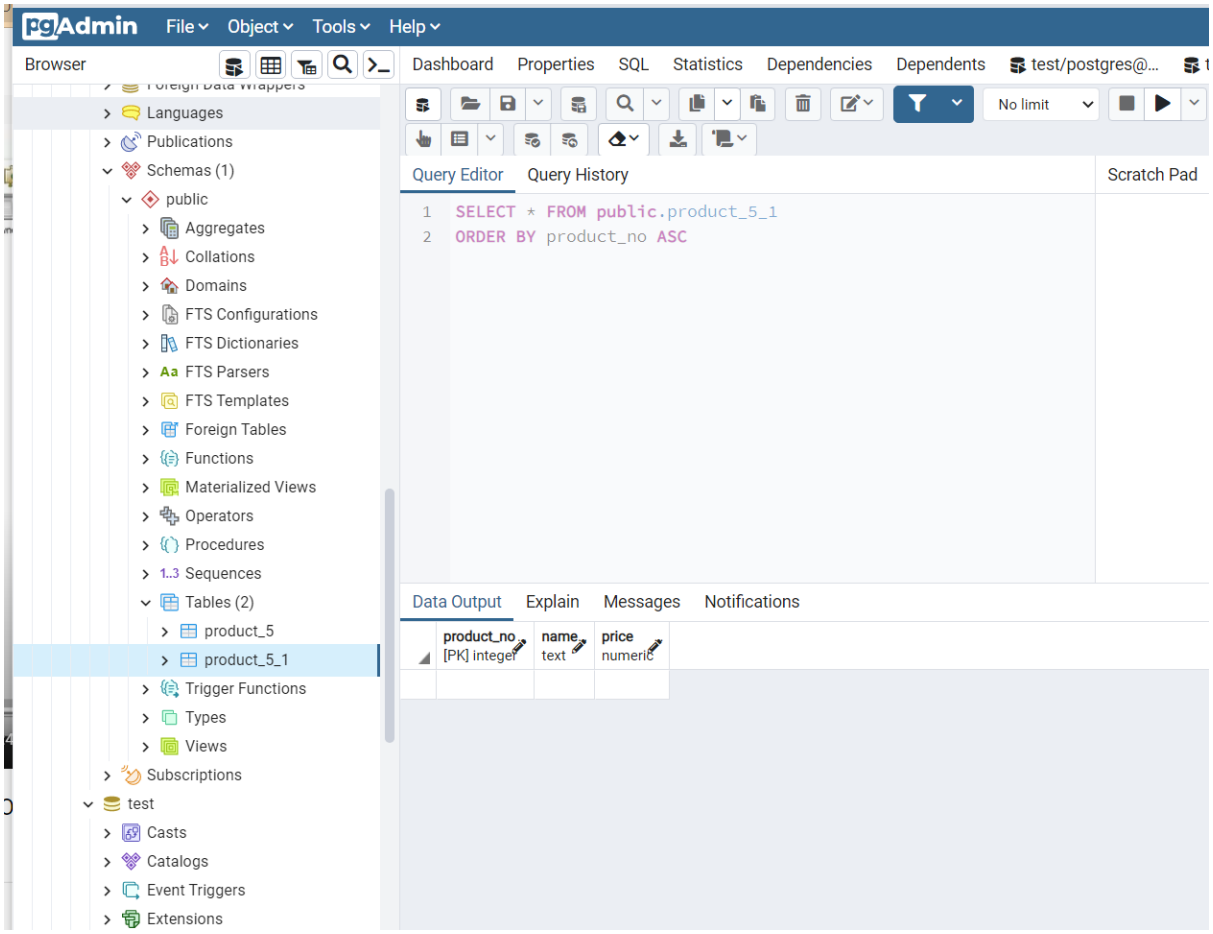
Query Editor | Query History | Scratch Pad

```
/*create table product_5(
product_no integer,
    name text,
    price numeric,
    PRIMARY KEY(name,product_no)
);*/
create table product_5_1(
product_no integer PRIMARY KEY,
    name text,
    price numeric
);
```

Data Output | Explain | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 91 msec.
```

INSSERTING DATA BY→



Error comes when order_id or product_no does not exist.

Query Editor | Query History | Scratch Pad

```
1  create table orders(
2  order_id integer PRIMARY KEY,
3  product_no integer REFERENCES product_6(product_no),
4  quantity integer
5  )
```

	order_id [PK] integer	product_no integer	quantity integer
1	1	1	2
2	2	3	4
*			

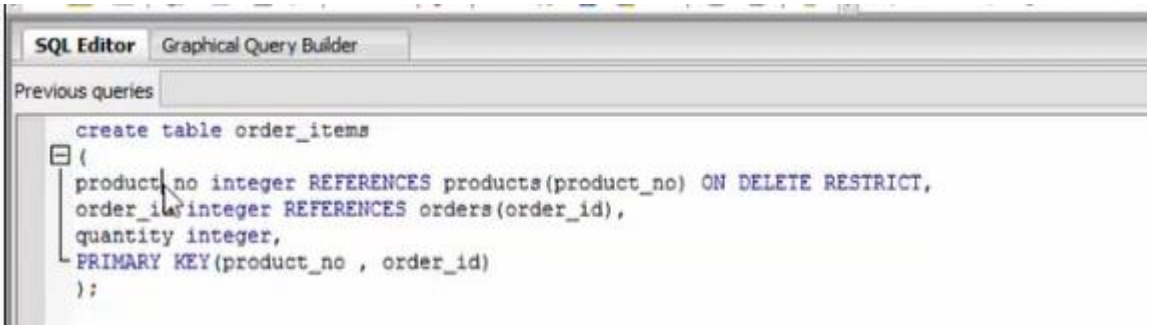
SQL Editor | Graphical Query Builder

Previous queries

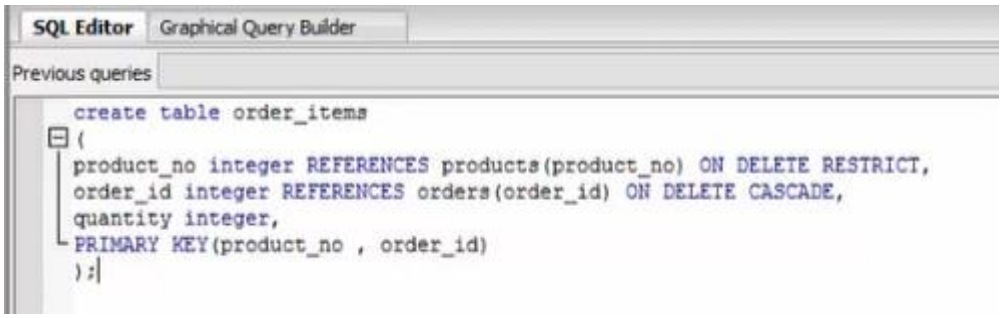
```
create table order_items
(
product_no integer REFERENCES products(product_no),
order_id integer REFERENCES orders(order_id),
quantity integer,
PRIMARY KEY(product_no,order_id)
);
```

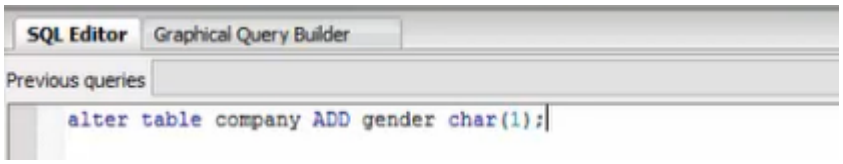
On Delete Restrict: **when data is removed from a parent table, and there is a foreign key associated with child table it gives error**, you can not delete the record.

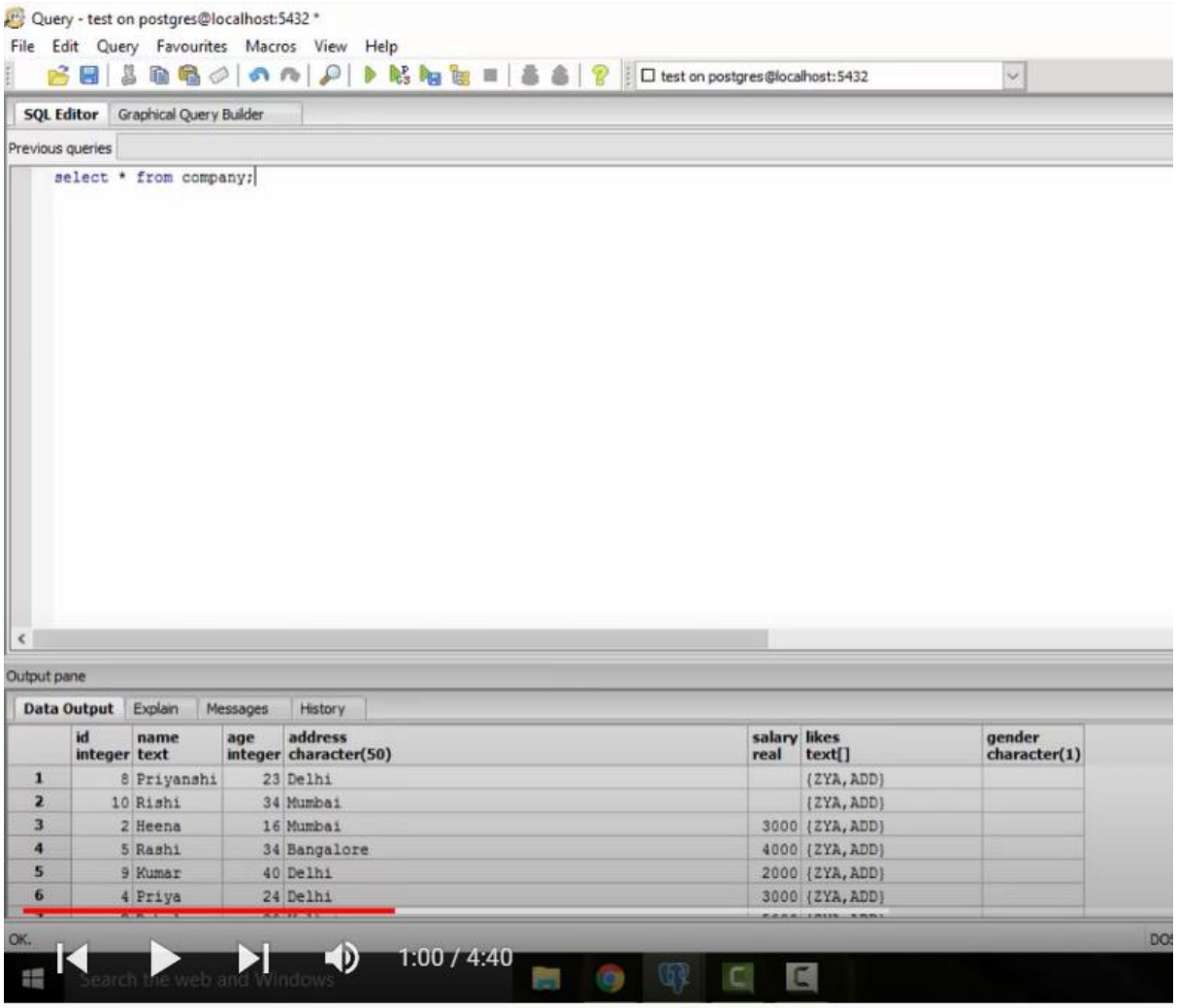
ON DELETE RESTRICT means **you can't delete a given parent row if a child row exists that references the value for that parent row**. If the parent row has no referencing child rows, then you can delete that parent row.

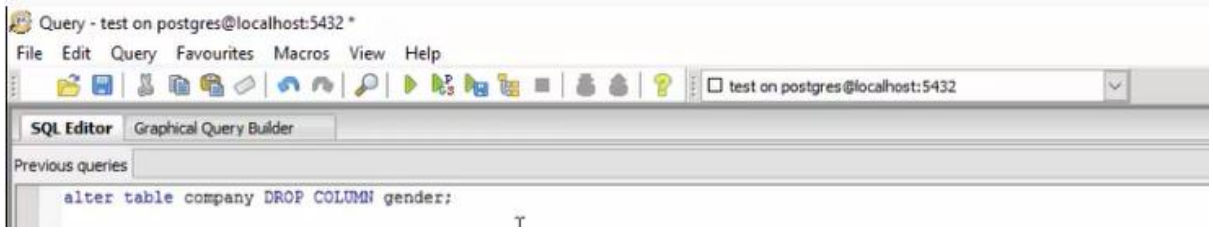


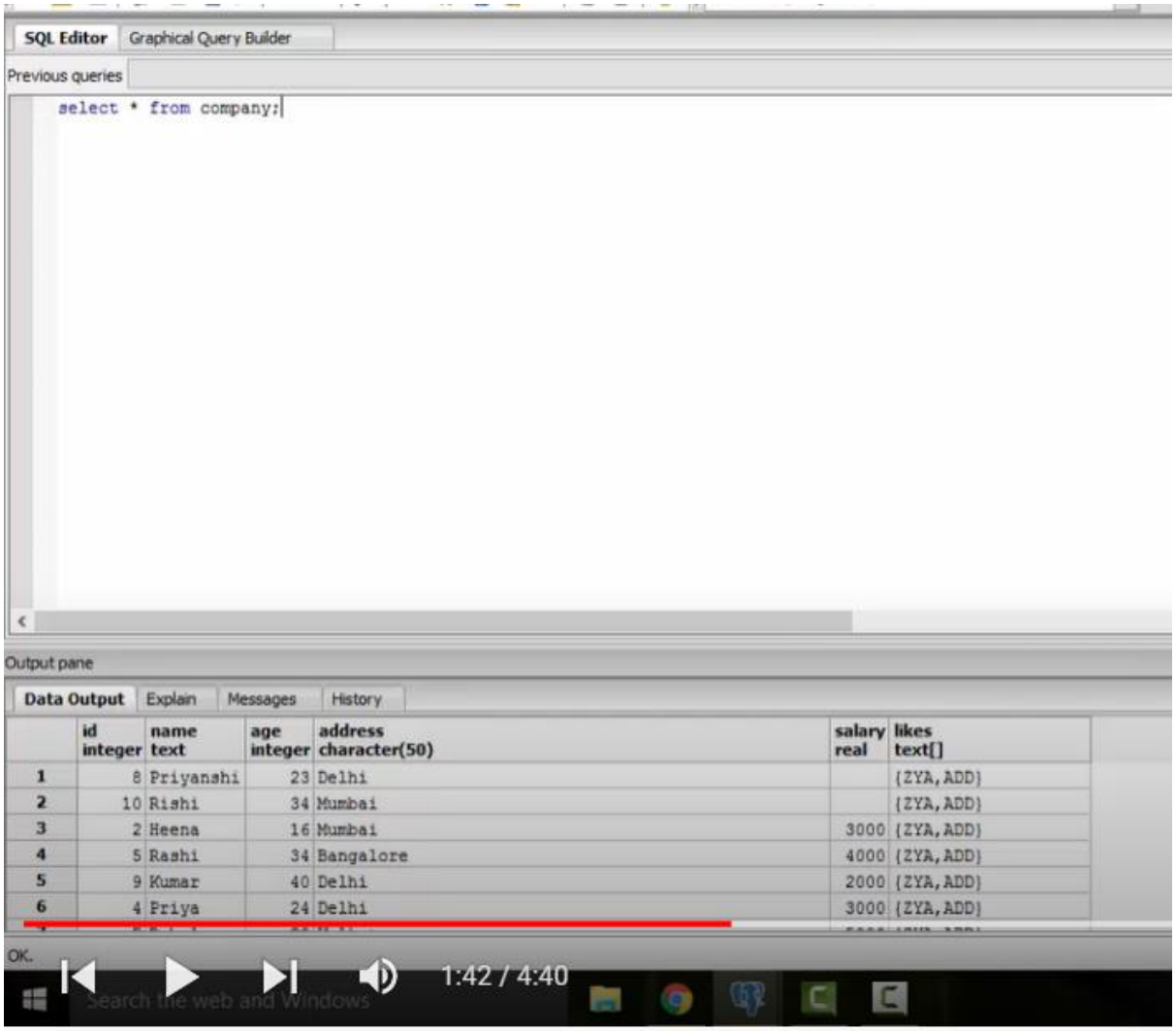
ON DELETE CASCADE clause is **used to automatically remove the matching records from the child table when we delete the rows from the parent table**. It is a kind of referential action related to the foreign key.



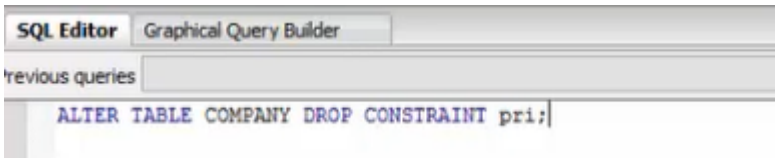



Query - test on postgres@localhost:5432 *
File Edit Query Favourites Macros View Help
test on postgres@localhost:5432
SQL Editor
Graphical Query Builder
Previous queries
select * from company;
Output pane
Data Output
Explain
Messages
History
id integer
name text
age integer
address character(50)
salary real
likes text[]
gender character(1)
1 8 Priyanshi 23 Delhi {ZYA,ADD}
2 10 Rishi 34 Mumbai {ZYA,ADD}
3 2 Heena 16 Mumbai 3000 {ZYA,ADD}
4 5 Rashi 34 Bangalore 4000 {ZYA,ADD}
5 9 Kumar 40 Delhi 2000 {ZYA,ADD}
6 4 Priya 24 Delhi 3000 {ZYA,ADD}
OK.
1:00 / 4:40

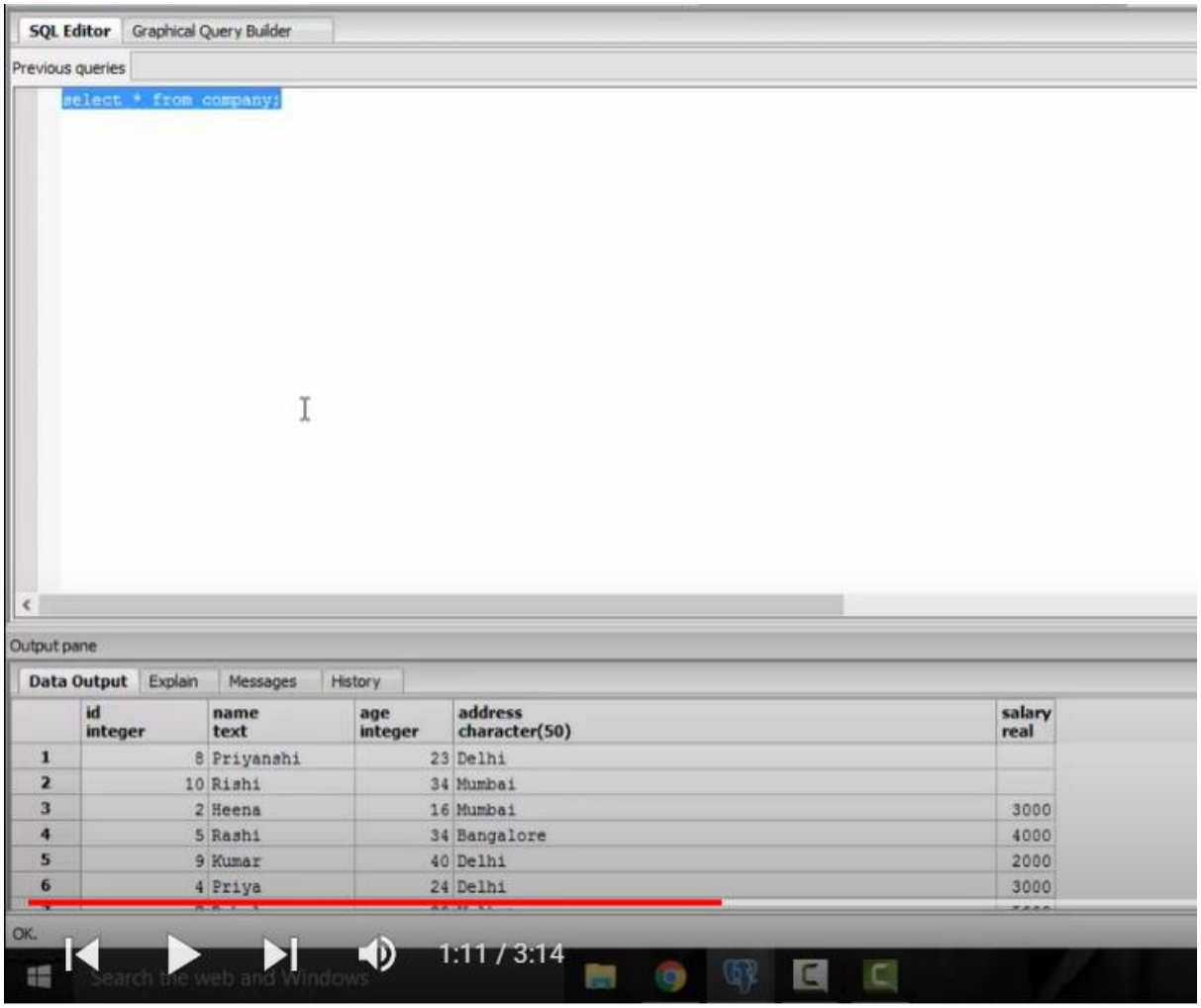

Query - test on postgres@localhost:5432 *
File Edit Query Favourites Macros View Help
test on postgres@localhost:5432
SQL Editor
Graphical Query Builder
Previous queries
alter table company DROP COLUMN gender;

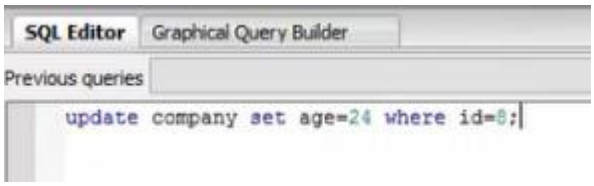

SQL Editor
Graphical Query Builder
Previous queries
select * from company;
Output pane
Data Output
Explain
Messages
History
id integer
name text
age integer
address character(50)
salary real
likes text[]
1 8 Priyanshi 23 Delhi {ZYA,ADD}
2 10 Rishi 34 Mumbai {ZYA,ADD}
3 2 Heena 16 Mumbai 3000 {ZYA,ADD}
4 5 Rashi 34 Bangalore 4000 {ZYA,ADD}
5 9 Kumar 40 Delhi 2000 {ZYA,ADD}
6 4 Priya 24 Delhi 3000 {ZYA,ADD}
OK.
1:42 / 4:40
Search the web and Windows

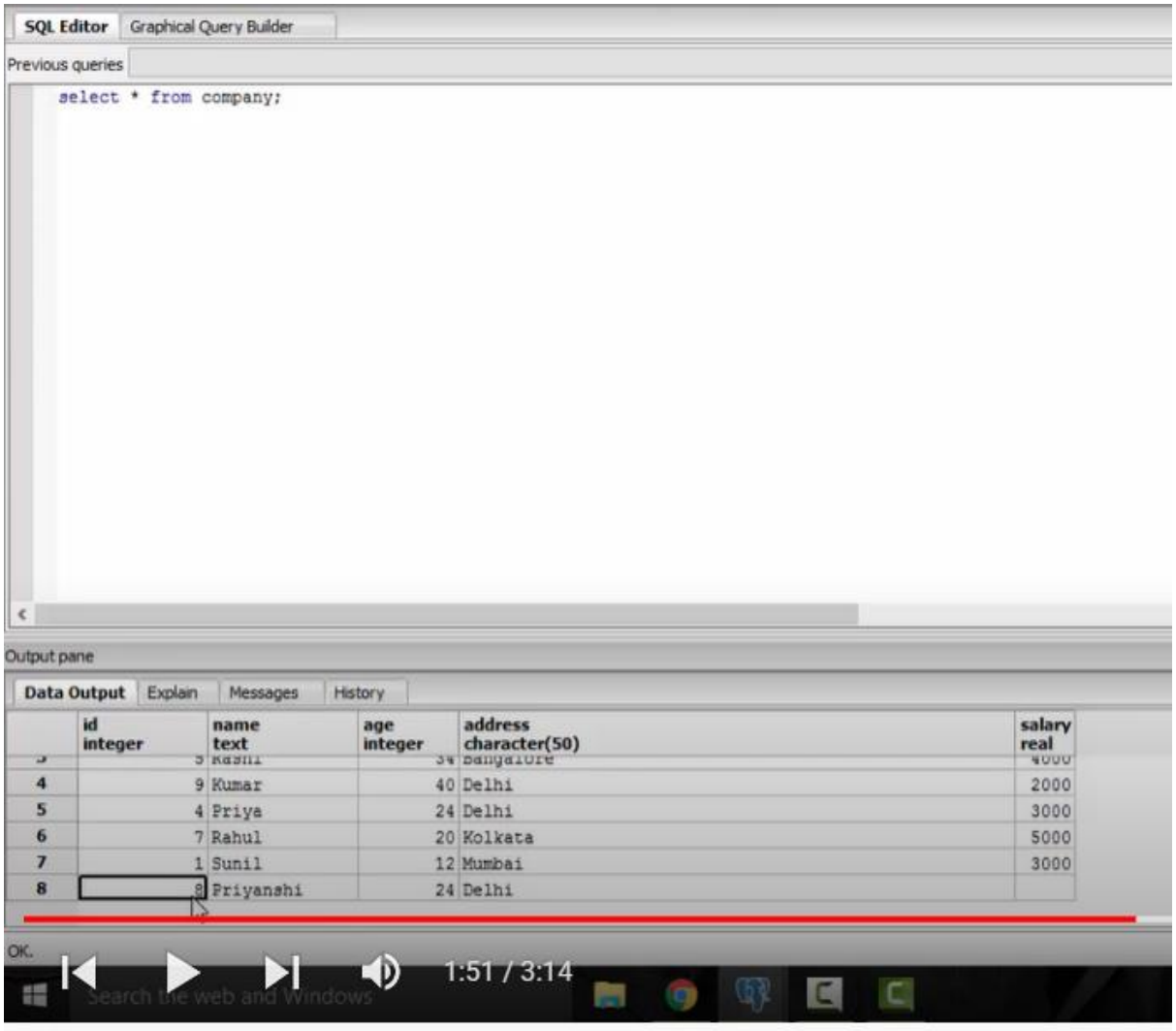
The above query used to add pri named primary key constraint .



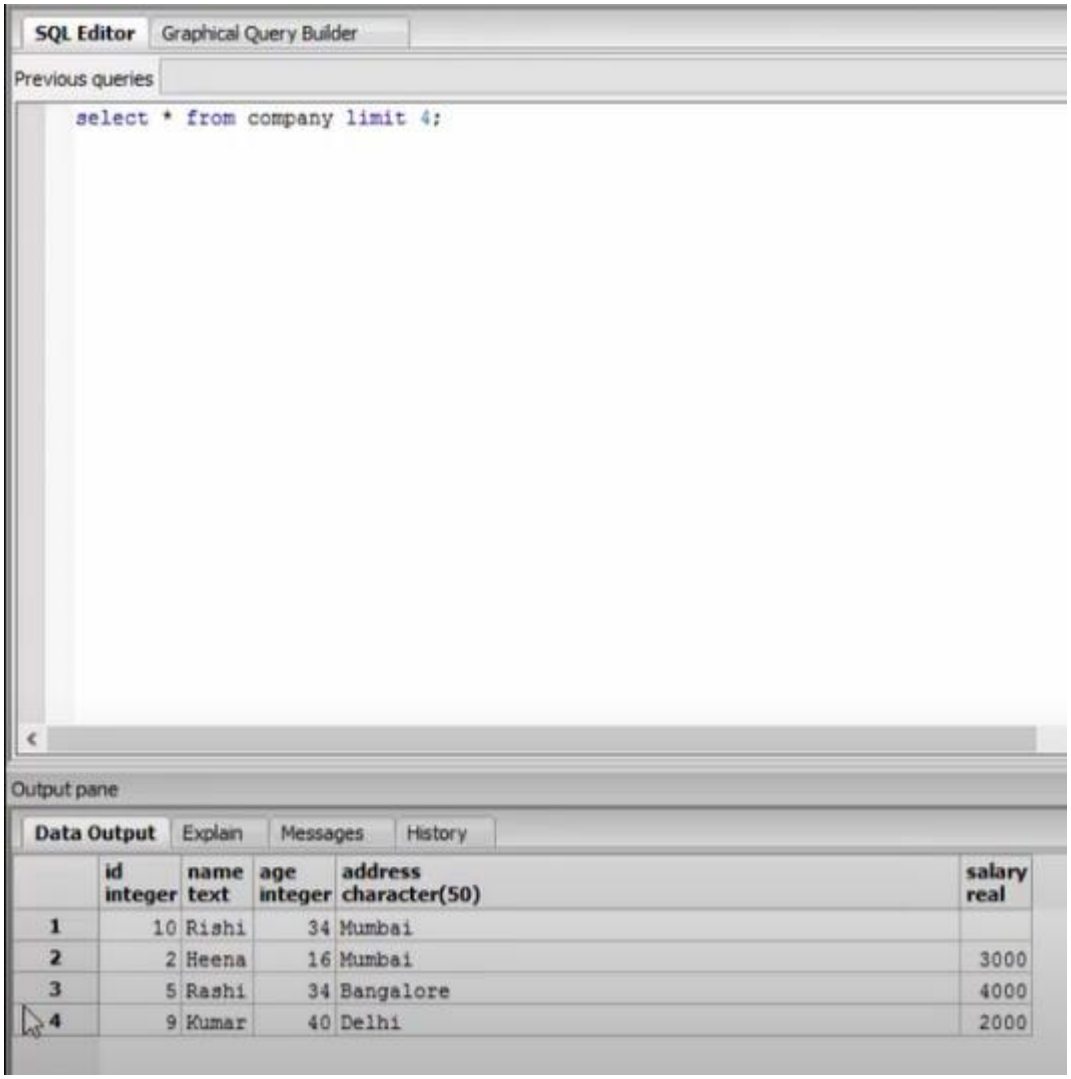
The above command to drop constraint pri.


SQL Editor
Graphical Query Builder
Previous queries
select * from company;
Output pane
Data Output
Explain
Messages
History
id integer
name text
age integer
address character(50)
salary real
1 8 Priyanshi 23 Delhi
2 10 Rishi 34 Mumbai
3 2 Heena 16 Mumbai 3000
4 5 Rashi 34 Bangalore 4000
5 9 Kumar 40 Delhi 2000
6 4 Priya 24 Delhi 3000
OK.
1:11 / 3:14

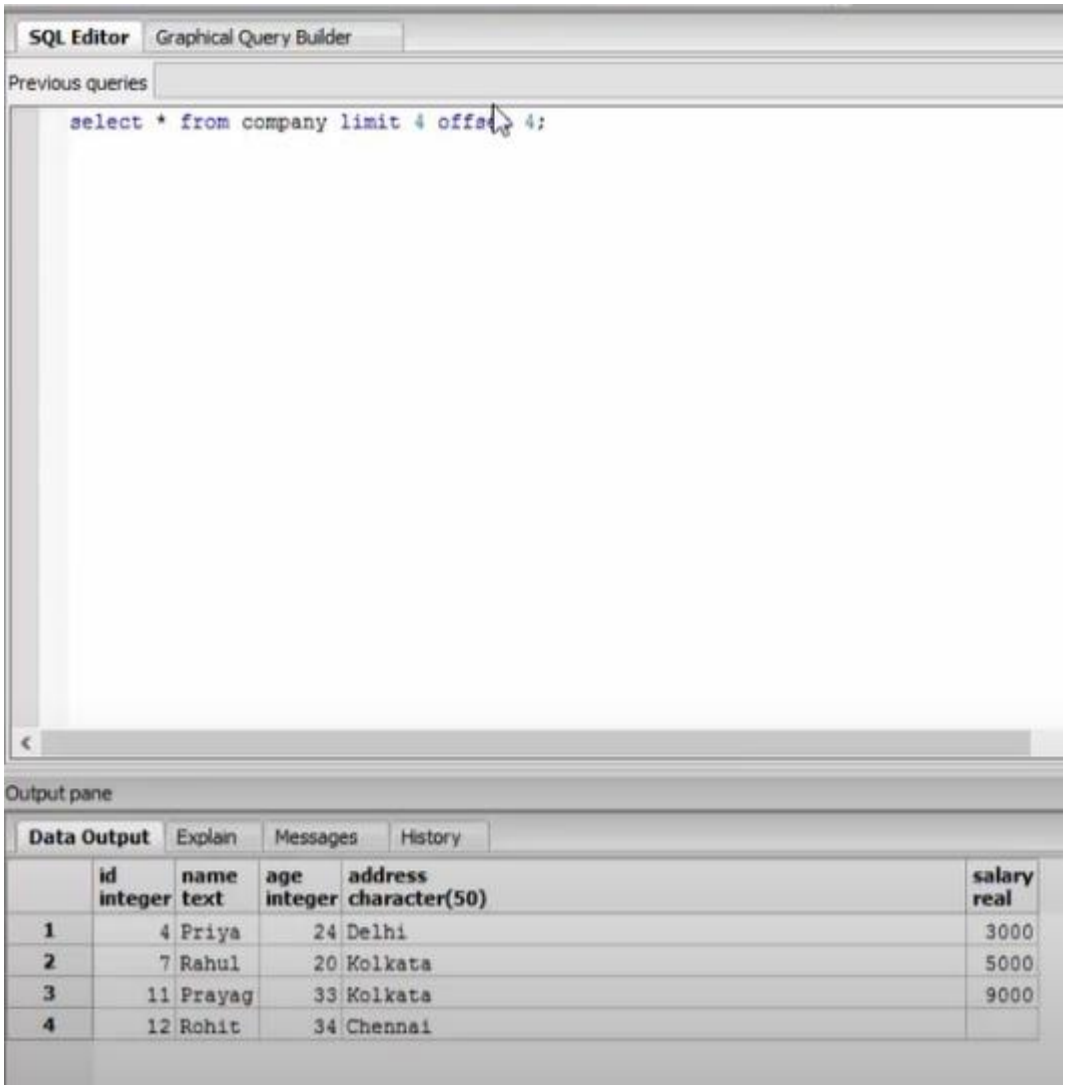

SQL Editor
Graphical Query Builder
Previous queries
update company set age=24 where id=8;


SQL Editor
Graphical Query Builder
Previous queries
select * from company;
Output pane
Data Output
Explain
Messages
History
id integer
name text
age integer
address character(50)
salary real
4 9 Kumar 40 Delhi 2000
5 4 Priya 24 Delhi 3000
6 7 Rahul 20 Kolkata 5000
7 1 Sunil 12 Mumbai 3000
8 8 Priyanshi 24 Delhi
OK.
1:51 / 3:14

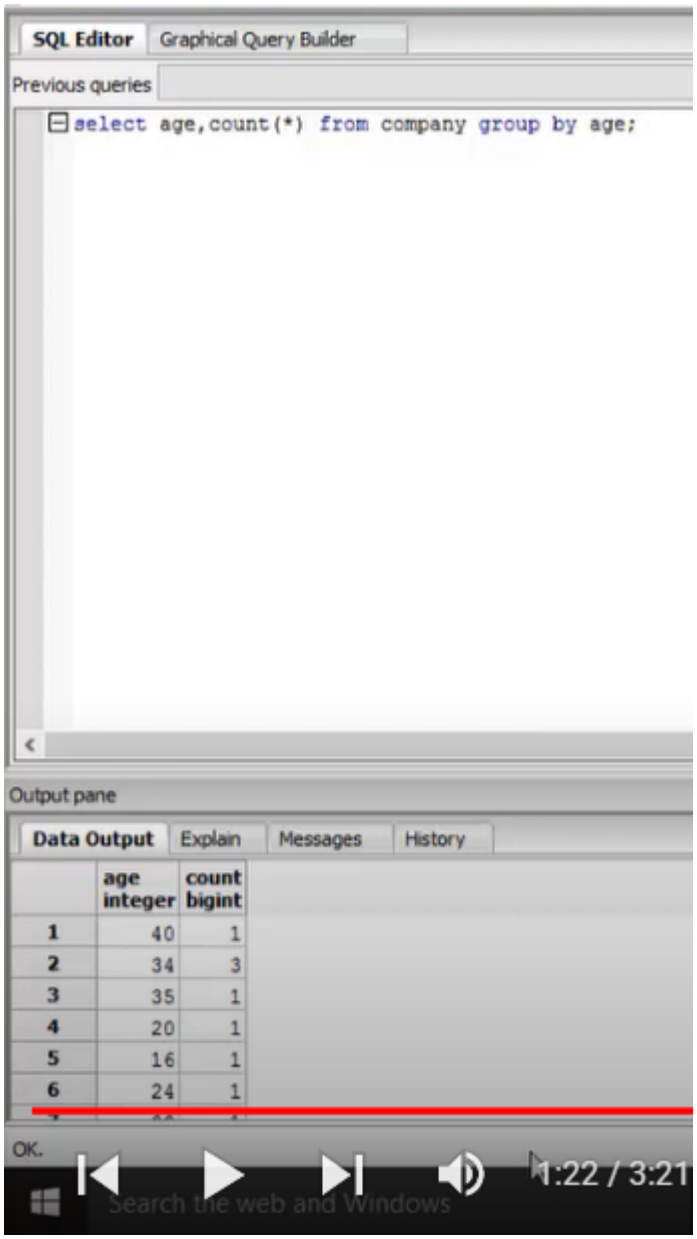

SQL Editor
Graphical Query Builder
Previous queries
delete from company where id = 8;

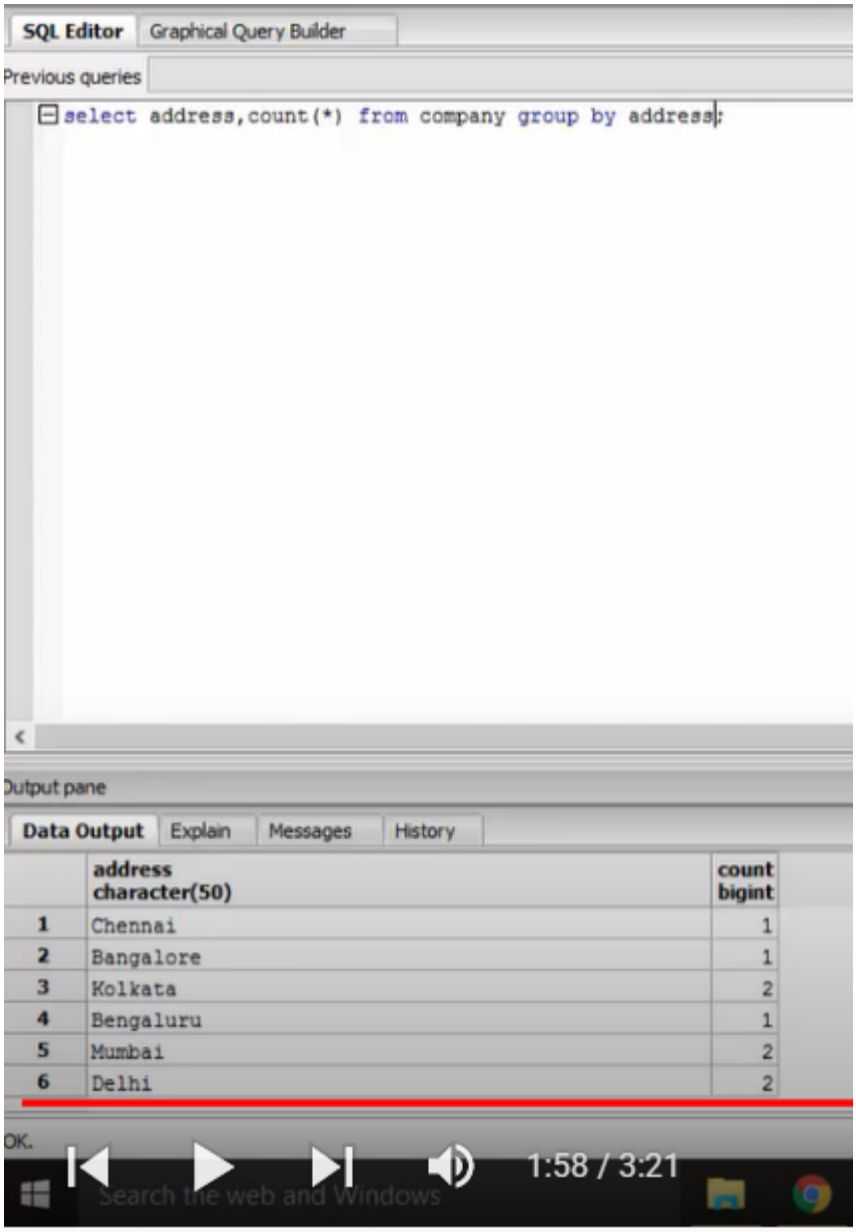


	id integer	name text	age integer	address character(50)	salary real
1	10	Rishi	34	Mumbai	
2	2	Heena	16	Mumbai	3000
3	5	Rashi	34	Bangalore	4000
4	9	Kumar	40	Delhi	2000

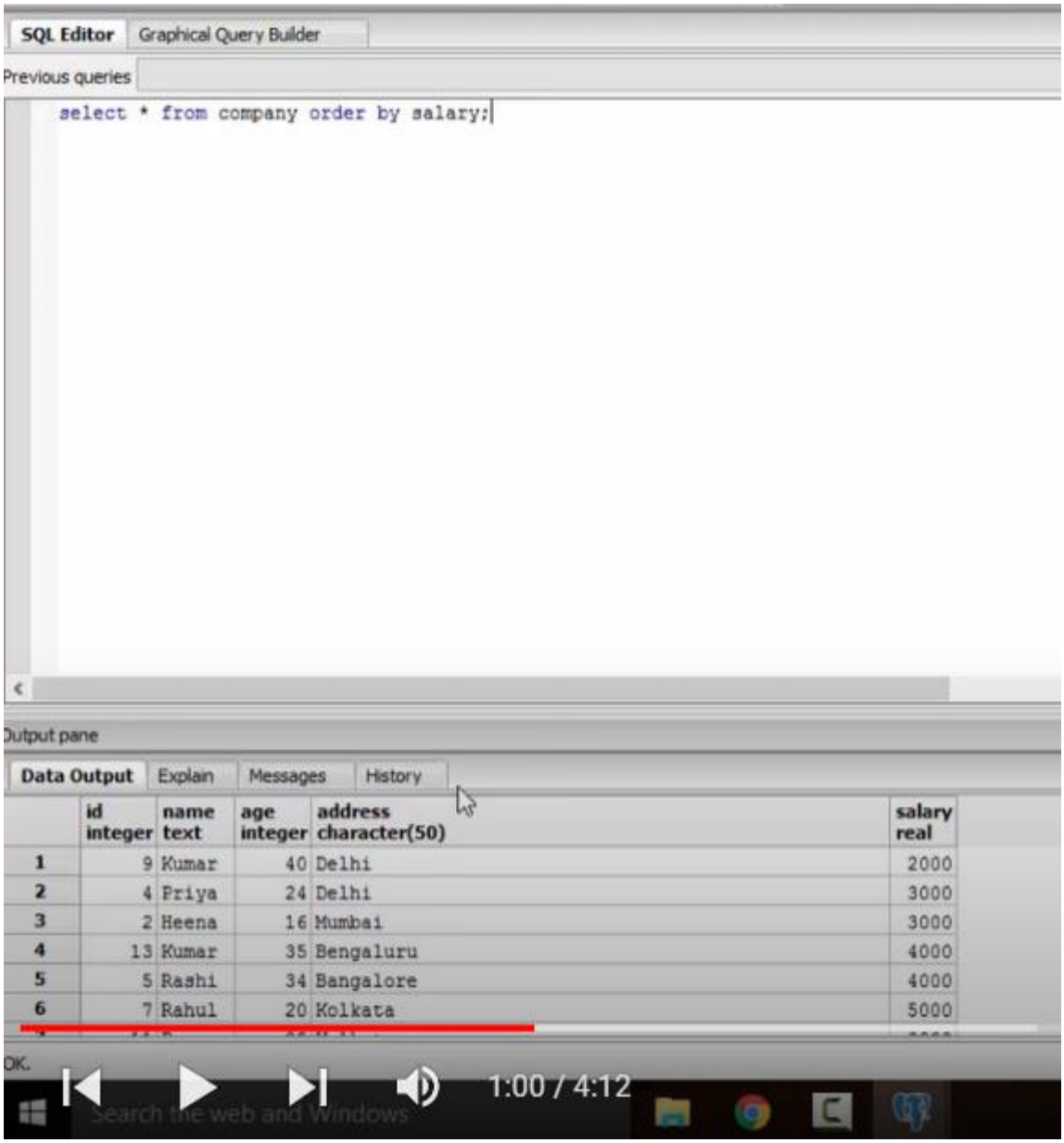


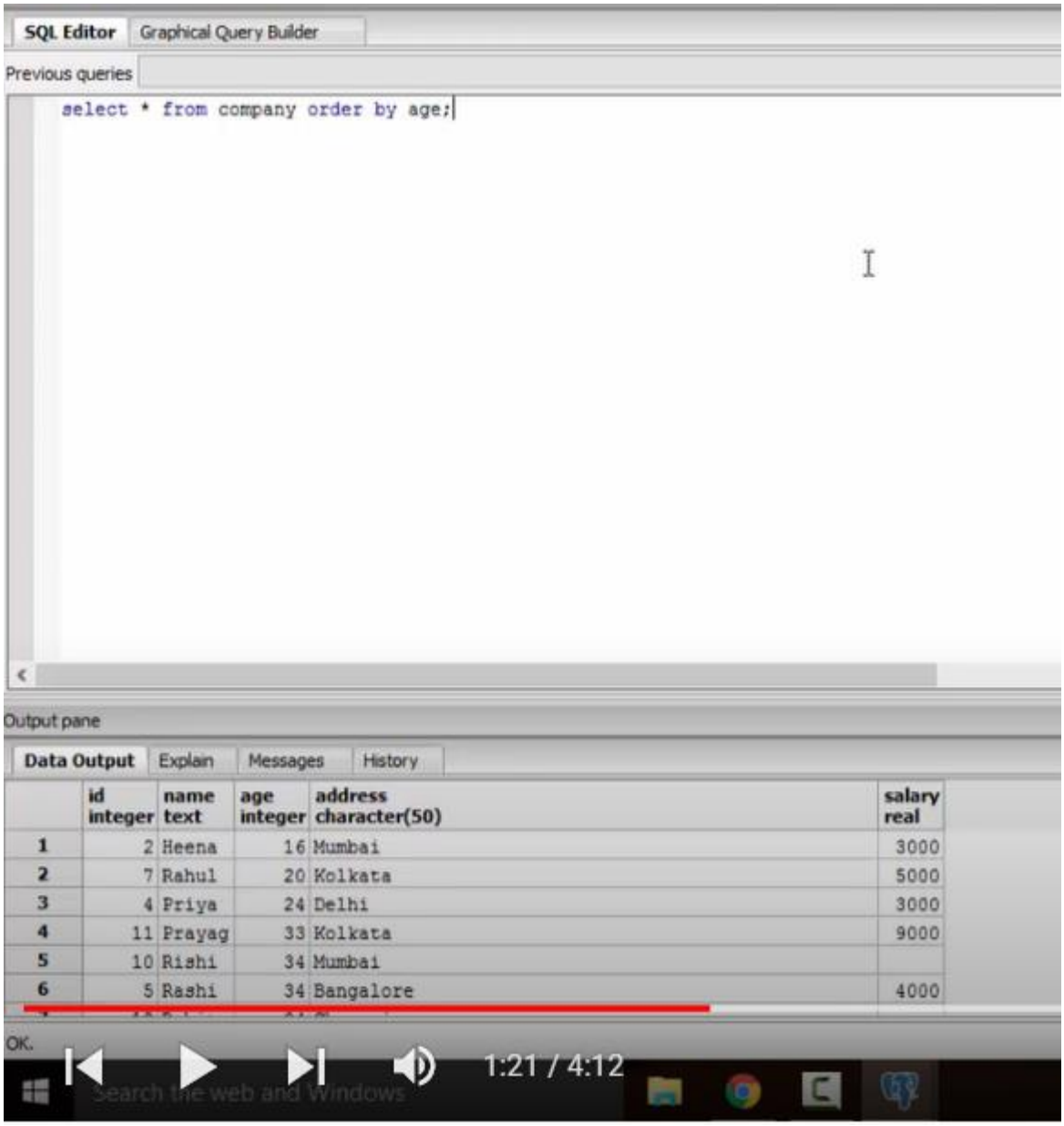
Limit 4 is used to get first 4. Offset 4 is used to get next 4 after first 4.

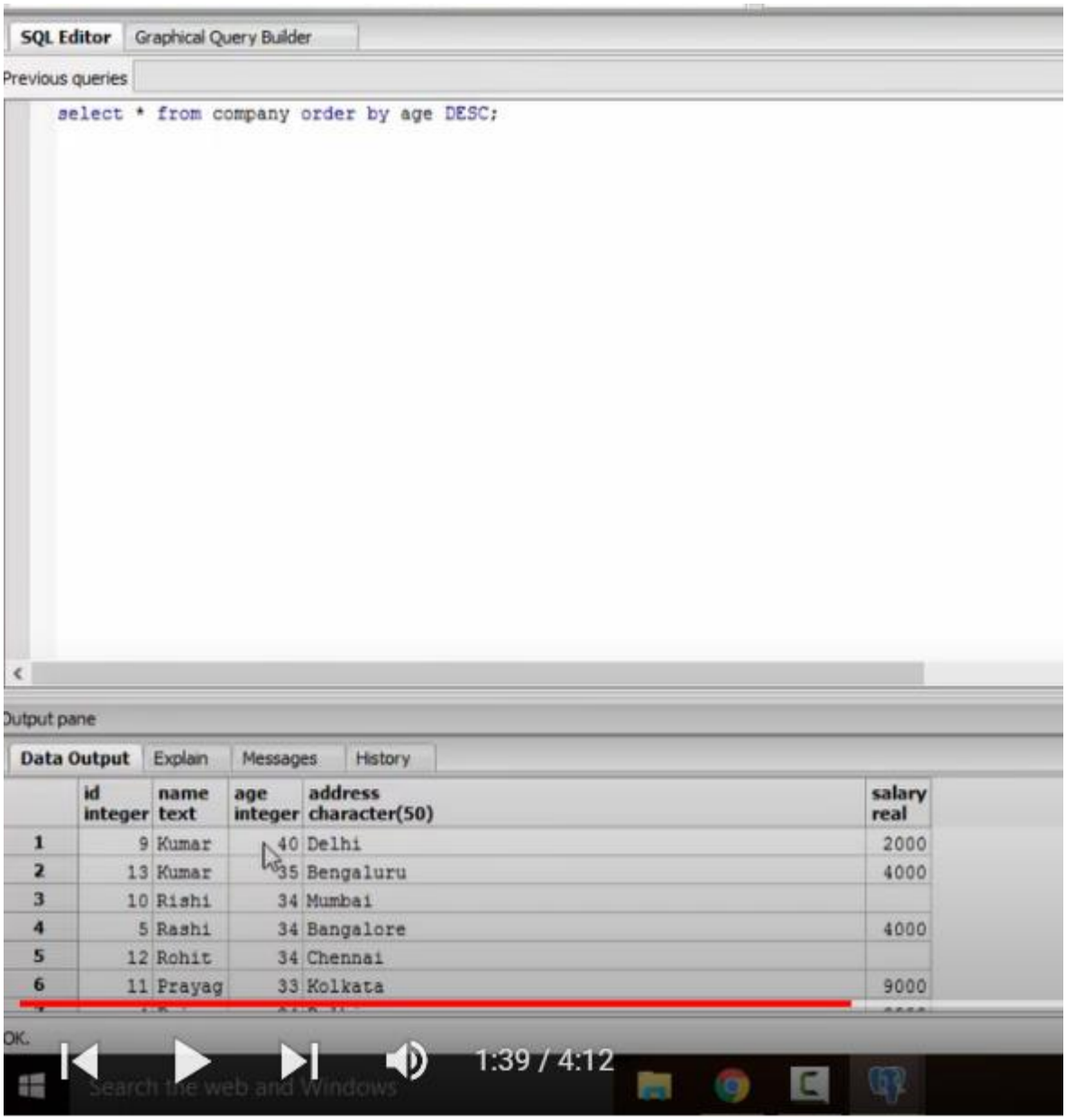

SQL Editor
Graphical Query Builder
Previous queries
select age,count(*) from company group by age;
Output pane
Data Output
Explain
Messages
History
age integer
count bigint
1 40 1
2 34 3
3 35 1
4 20 1
5 16 1
6 24 1
OK.
1:22 / 3:21
Search the web and Windows


SQL Editor
Graphical Query Builder
Previous queries
select address,count(*) from company group by address;
Output pane
Data Output
Explain
Messages
History
address character(50)
count bigint
1 Chennai 1
2 Bangalore 1
3 Kolkata 2
4 Bengaluru 1
5 Mumbai 2
6 Delhi 2
OK.
1:58 / 3:21
Search the web and Windows

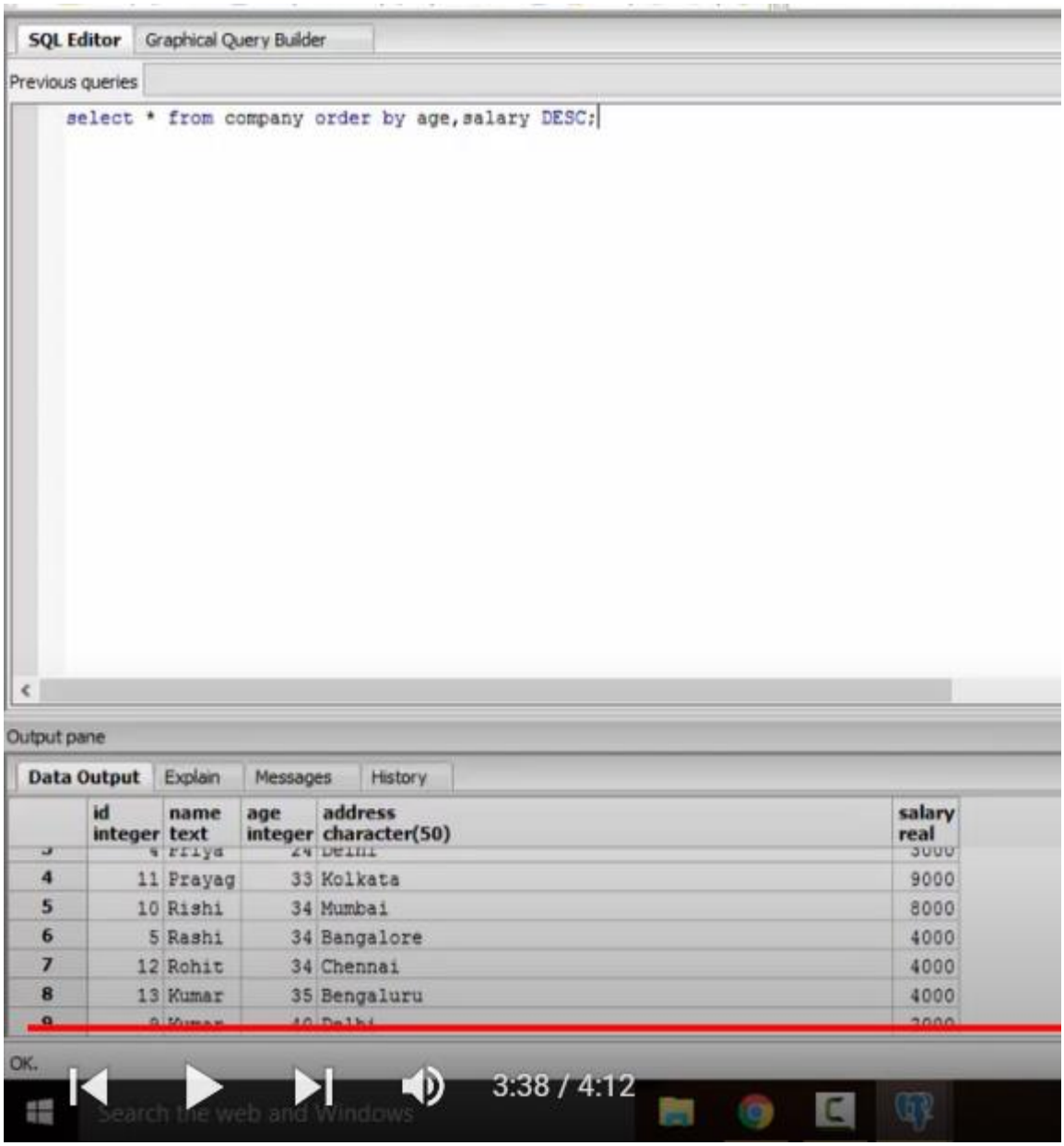
	address character(50)	count bigint
1	Bangalore	1
2	Kolkata	2
3	Bengaluru	1
4	Mumbai	2
5	Delhi	2


SQL Editor
Graphical Query Builder
Previous queries
select * from company order by salary;
Output pane
Data Output
Explain
Messages
History
id integer
name text
age integer
address character(50)
salary real
1 9 Kumar 40 Delhi 2000
2 4 Priya 24 Delhi 3000
3 2 Heena 16 Mumbai 3000
4 13 Kumar 35 Bengaluru 4000
5 5 Rashi 34 Bangalore 4000
6 7 Rahul 20 Kolkata 5000
OK.
1:00 / 4:12

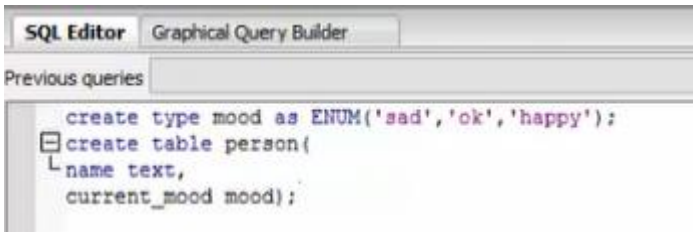

SQL Editor
Graphical Query Builder
Previous queries
select * from company order by age;
Output pane
Data Output
Explain
Messages
History
id integer
name text
age integer
address character(50)
salary real
1 2 Heena 16 Mumbai 3000
2 7 Rahul 20 Kolkata 5000
3 4 Priya 24 Delhi 3000
4 11 Prayag 33 Kolkata 9000
5 10 Rishi 34 Mumbai
6 5 Rashi 34 Bangalore 4000
OK.
1:21 / 4:12

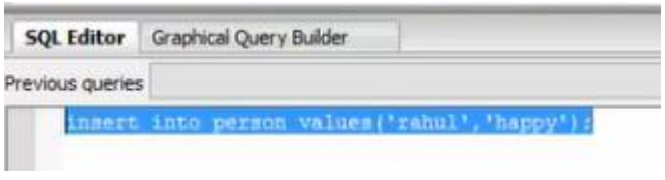


DESC FOR DESCENDING AND ASC FOR ASCENDING.

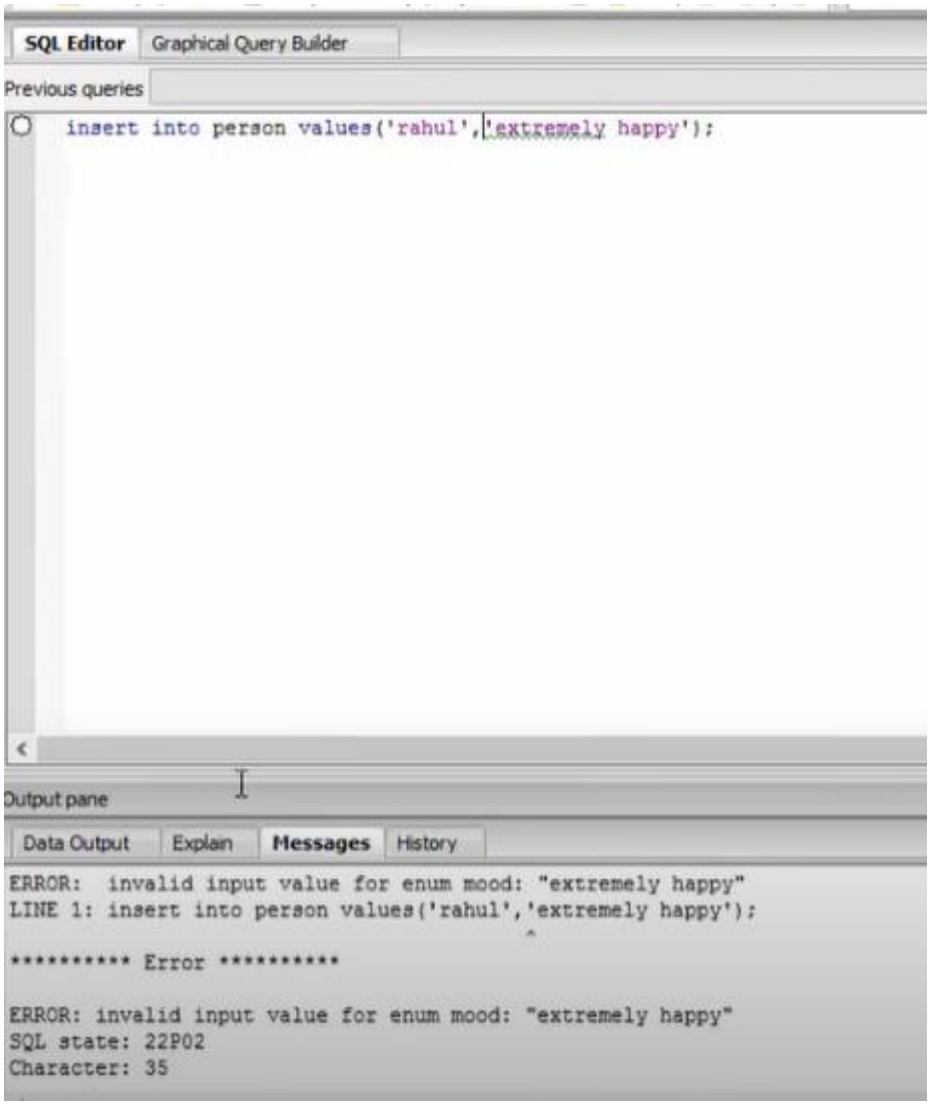


ORDERED BY AGE IN ASCENDING ORDER AND SALARY IN DESCENDING ORDER.

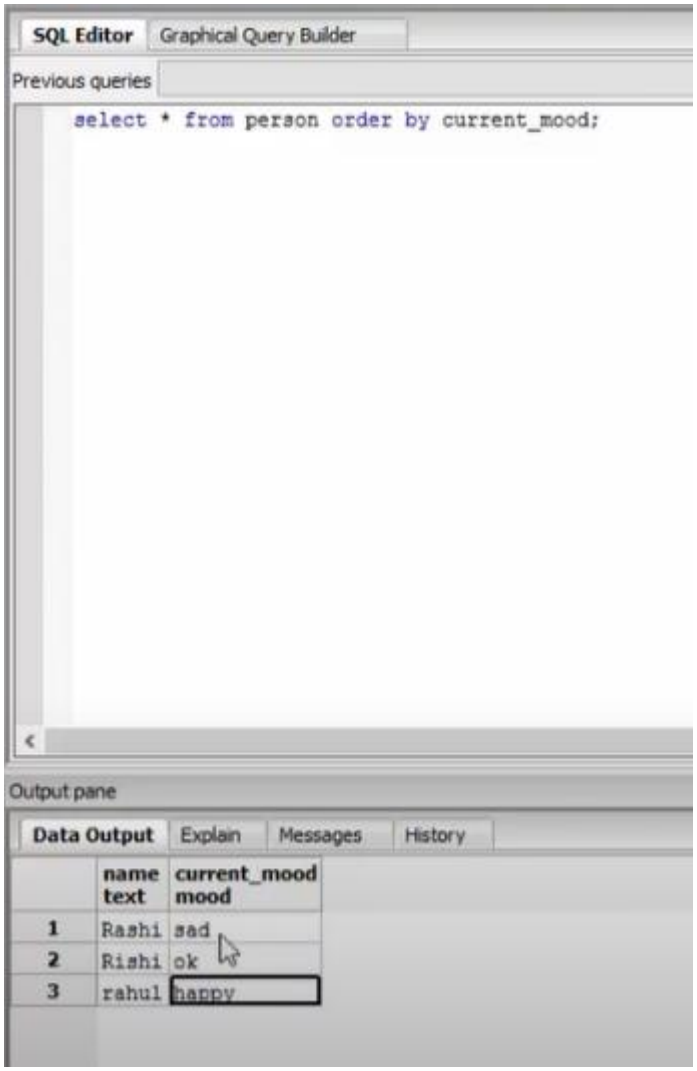


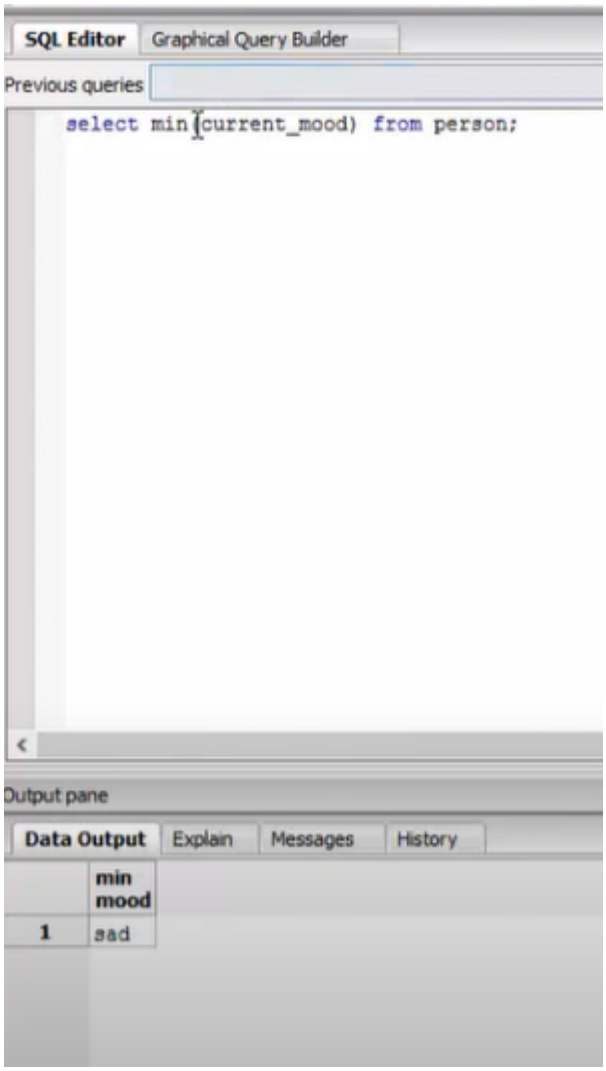


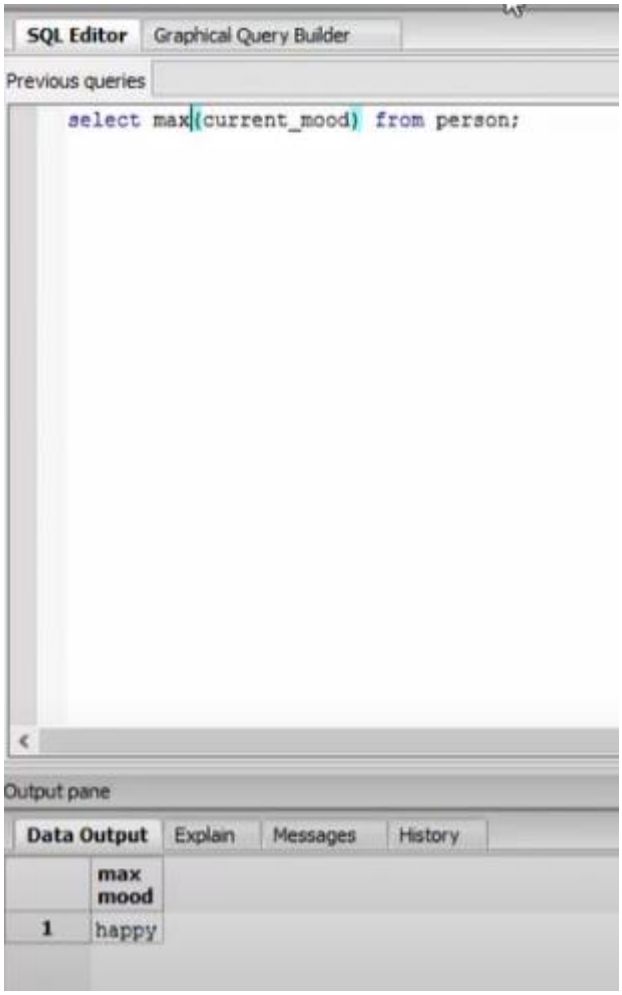

SQL Editor
Graphical Query Builder
Previous queries
select * from person;
Output pane
Data Output
Explain
Messages
History
name text
current_mood mood
1 rahul happy
2 Rishi ok
3 Rashi sad

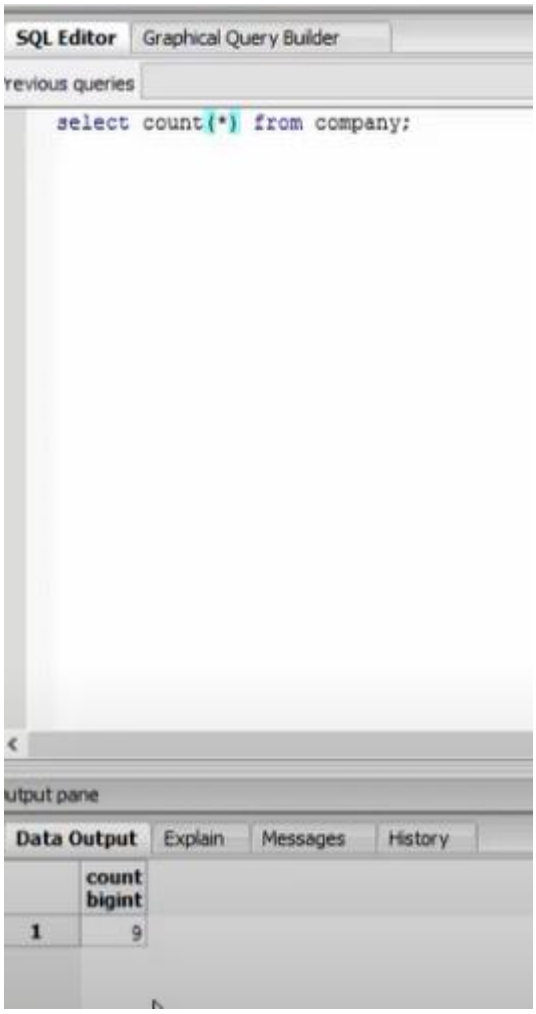


ERROR AS EXTREMELY HAPPY WAS NOT DECLARED IN THE ENUMERATION.


SQL Editor
Graphical Query Builder
Previous queries
select * from person order by current_mood;
Output pane
Data Output
Explain
Messages
History
name text
current_mood mood
1 Rashi sad
2 Rishi ok
3 rahul happy


SQL Editor
Graphical Query Builder
Previous queries
select min(current_mood) from person;
Output pane
Data Output
Explain
Messages
History
min mood
1
sad


SQL Editor
Graphical Query Builder
Previous queries
select max(current_mood) from person;
Output pane
Data Output
Explain
Messages
History
max mood
1
happy

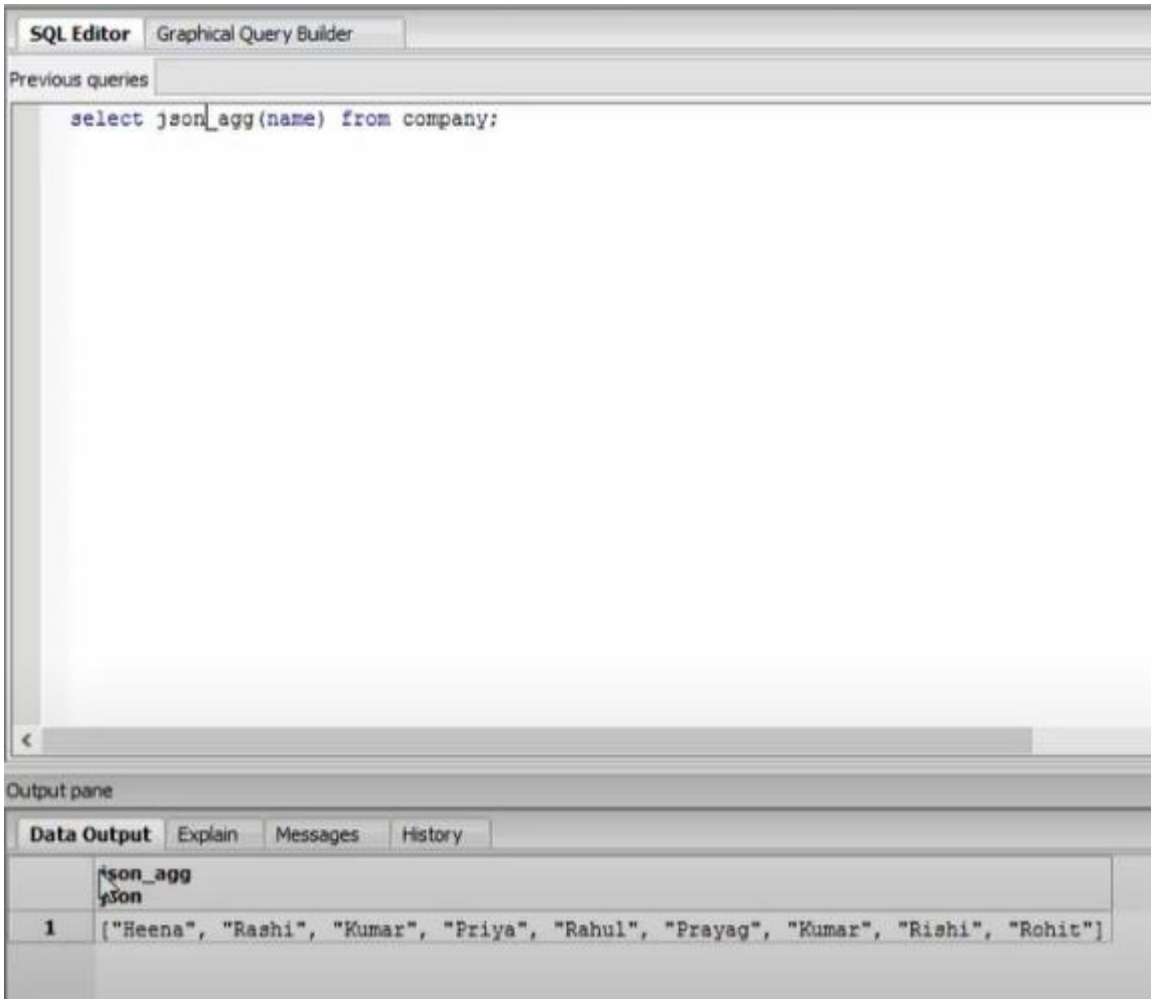

SQL Editor
Graphical Query Builder
revious queries
select count(*) from company;
utput pane
Data Output
Explain
Messages
History
count
bigint
1
9


SQL Editor
Graphical Query Builder
Previous queries
select max(age) from company;
Output pane
Data Output
Explain
Messages
Histor
max
integer
1
40


SQL Editor
Graphical Query Builder
Previous queries
select avg(age) from company;
Output pane
Data Output
Explain
Messages
History
avg
numeric
1
30.0000000000000000

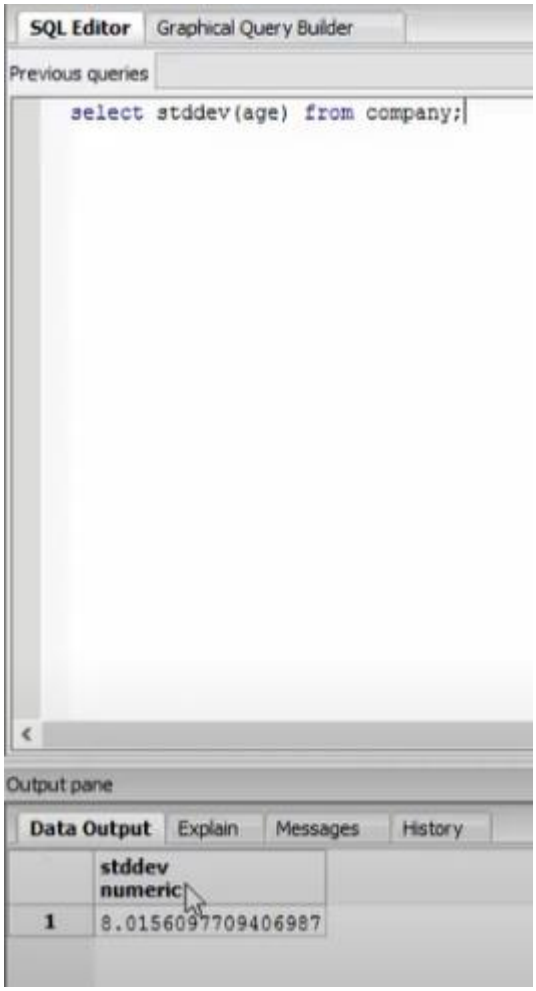

SQL Editor
Graphical Query Builder
Previous queries
select sum(age) from company;
Output pane
Data Output
Explain
Messages
History
sum
bigint
1
270

THE ABOVE QUERY RETURNS AN ARRAY OF NAMES FROM COMPANY TABLE.



THE ABOVE QUERY GIVES JSON FORM OF OUTPUT OF NAMES.

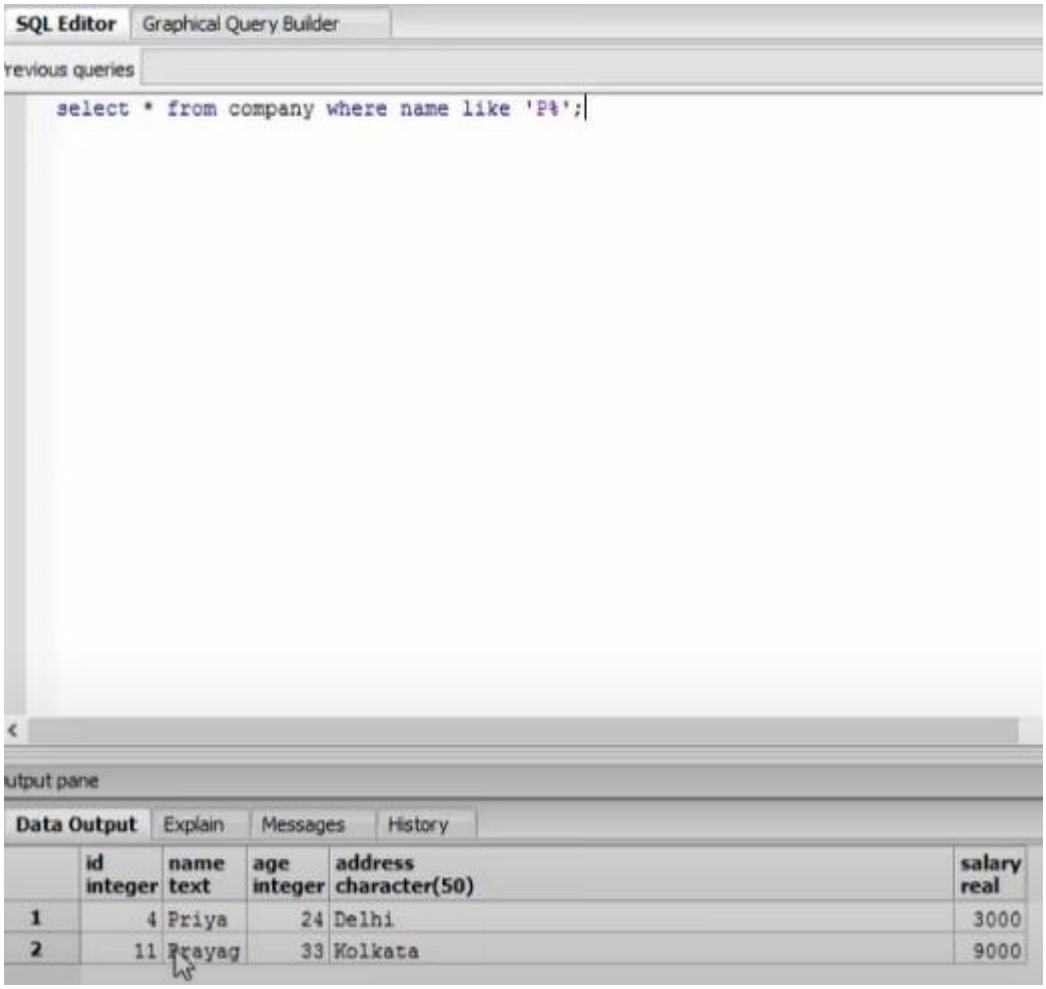
THE ABOVE QUERY GIVES KEY-VALUE FORMAT OF NAME AND AGE IN JSON FORMAT.

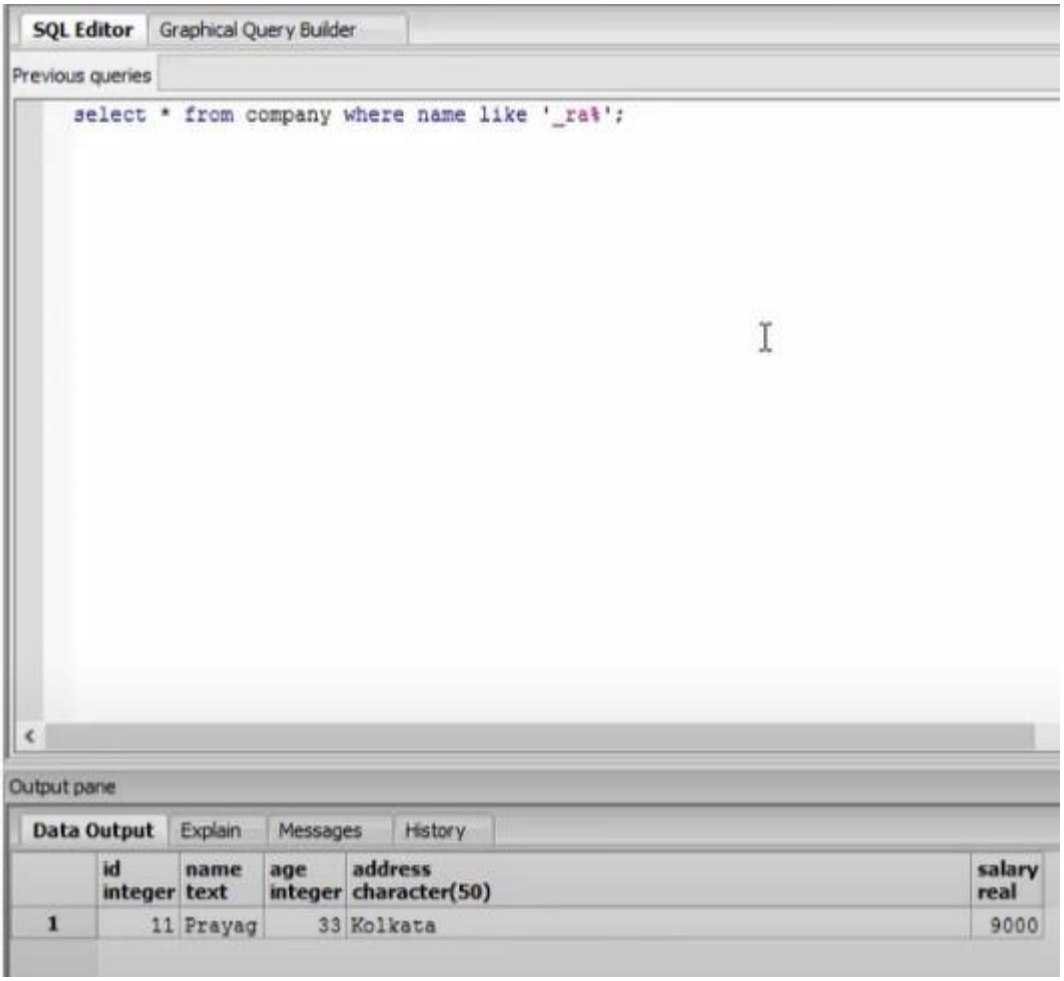

SQL Editor
Graphical Query Builder
Previous queries
select stddev(age) from company;
Output pane
Data Output
Explain
Messages
History
stddev
numeric
1
8.015609770940698

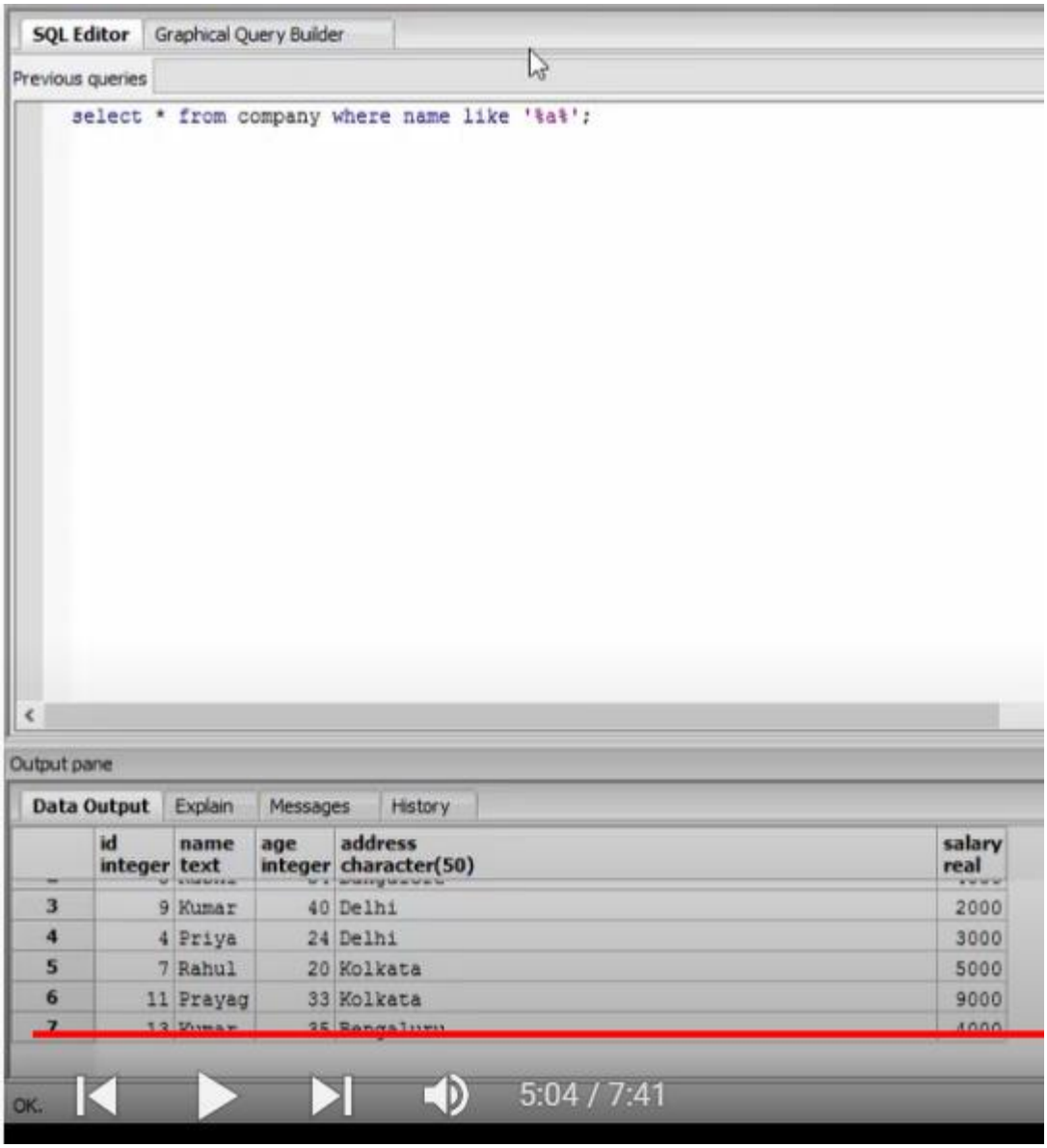

SQL Editor
Graphical Query Builder
Previous queries
select variance(age) from company;
Output pane
Data Output
Explain
Messages
History
variance
numeric
1
64.2500000000000000

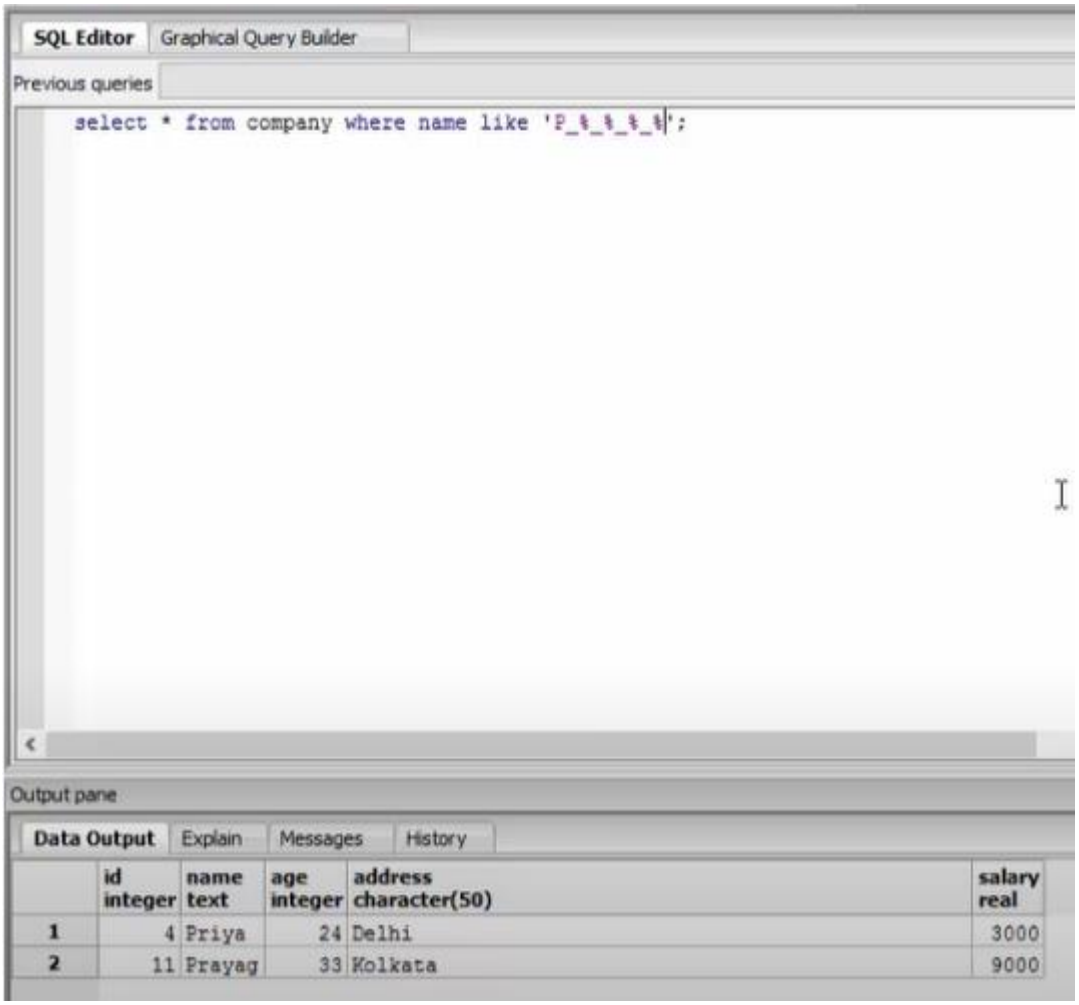
THE ABOVE QUERY RETURNS THE MOST OCCURING AGE IN THE COMPANY TABLE.

RETURNS THE MOST OCCURING SALARY FROM THE TABLE.

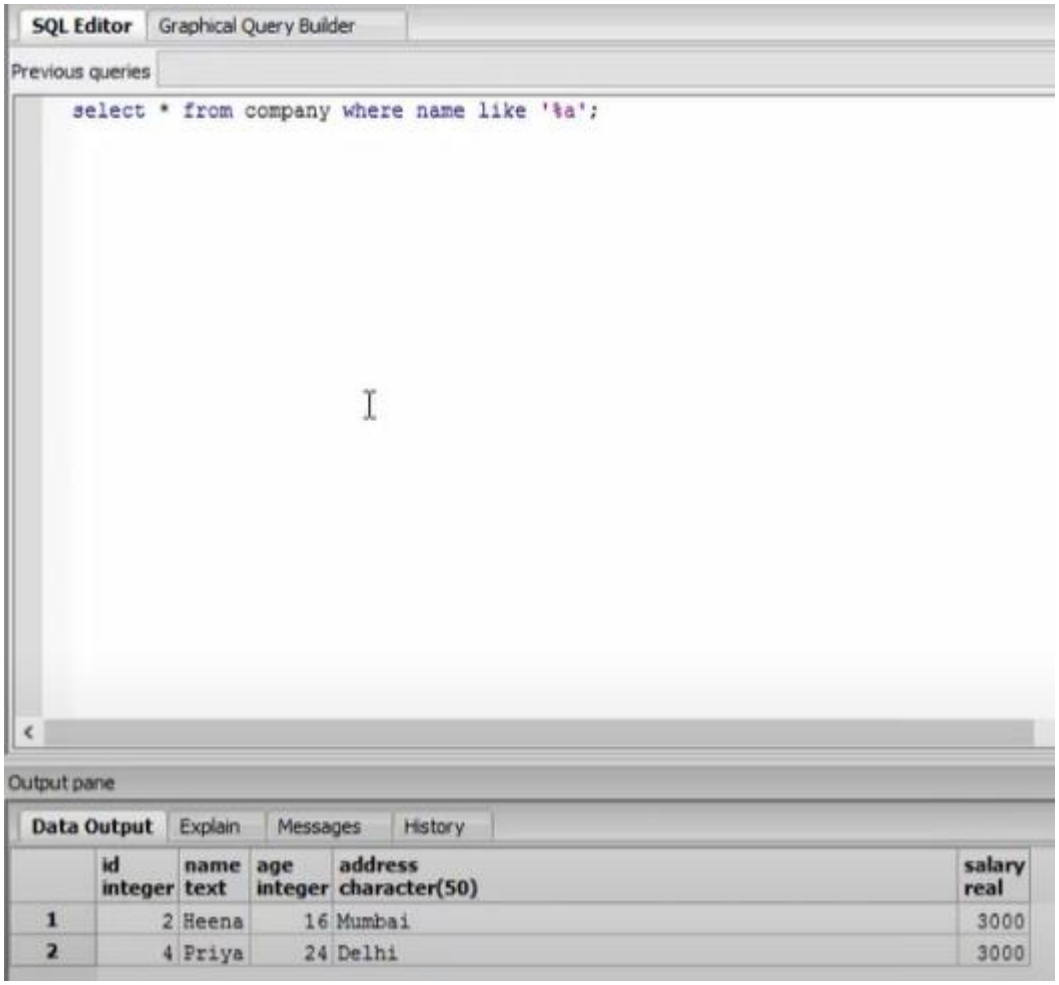

SQL Editor
Graphical Query Builder
revious queries
select * from company where name like 'P%';
utput pane
Data Output
Explain
Messages
History
id integer
name text
age integer
address character(50)
salary real
1
4
Priya
24
Delhi
3000
2
11
Prayag
33
Kolkata
9000


SQL Editor
Graphical Query Builder
Previous queries
select * from company where name like '_ra%';
Output pane
Data Output
Explain
Messages
History
id integer
name text
age integer
address character(50)
salary real
1
11
Prayag
33
Kolkata
9000

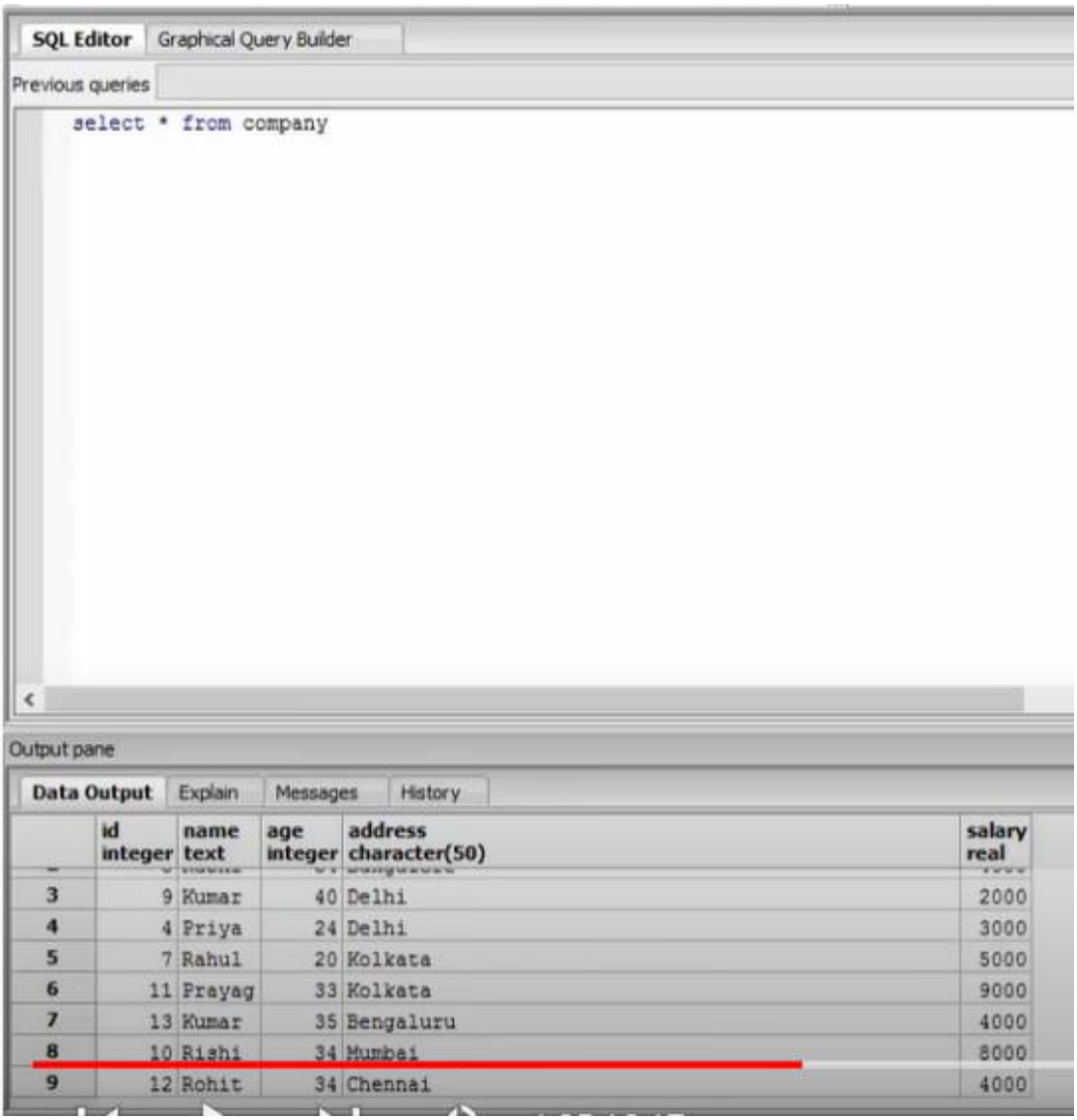

SQL Editor
Graphical Query Builder
Previous queries
select * from company where name like '%a%';
Output pane
Data Output
Explain
Messages
History
id integer
name text
age integer
address character(50)
salary real
3 9 Kumar 40 Delhi 2000
4 4 Priya 24 Delhi 3000
5 7 Rahul 20 Kolkata 5000
6 11 Prayag 33 Kolkata 9000
OK.
5:04 / 7:41

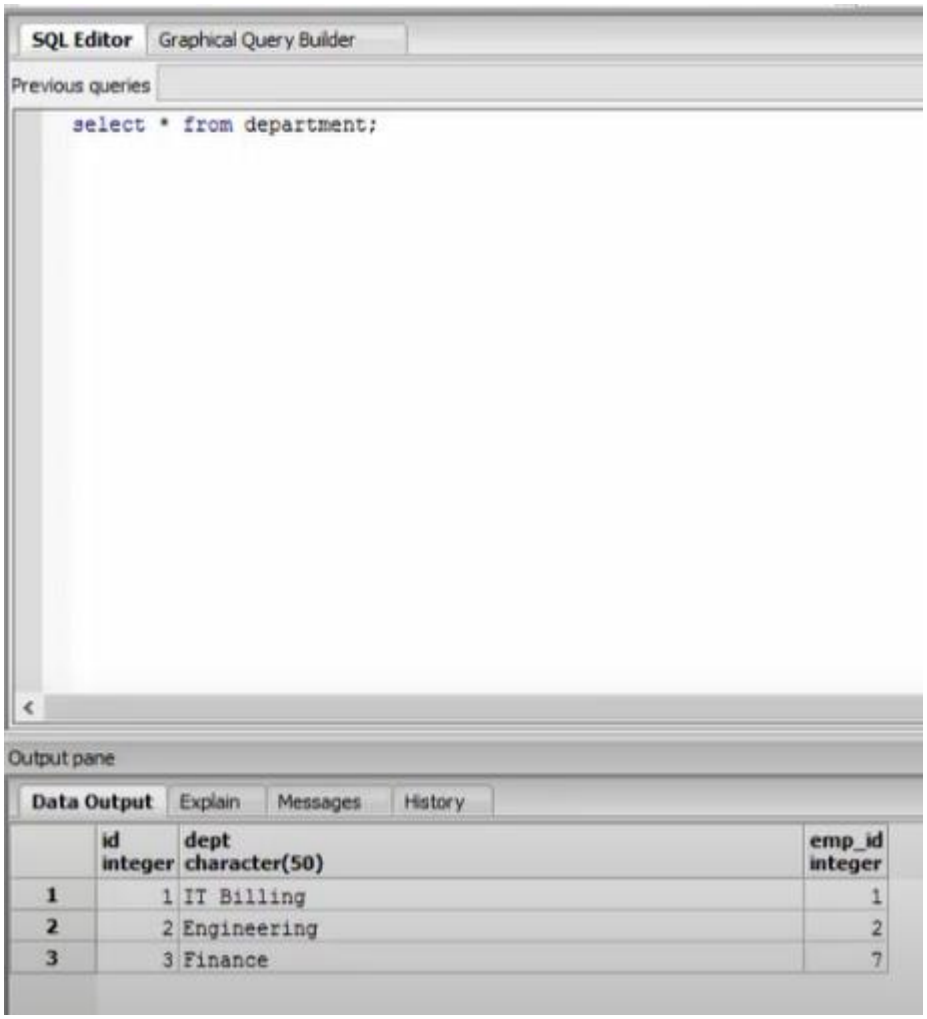


Name to be atleast 5 characters long.

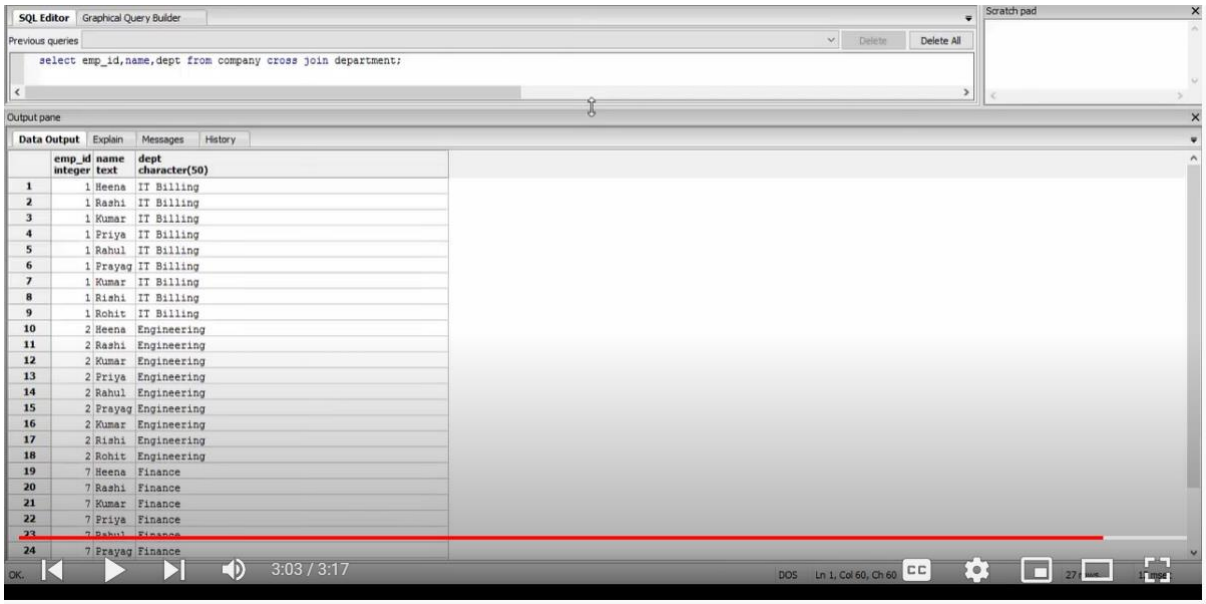


The above query for words ending with ‘a’.

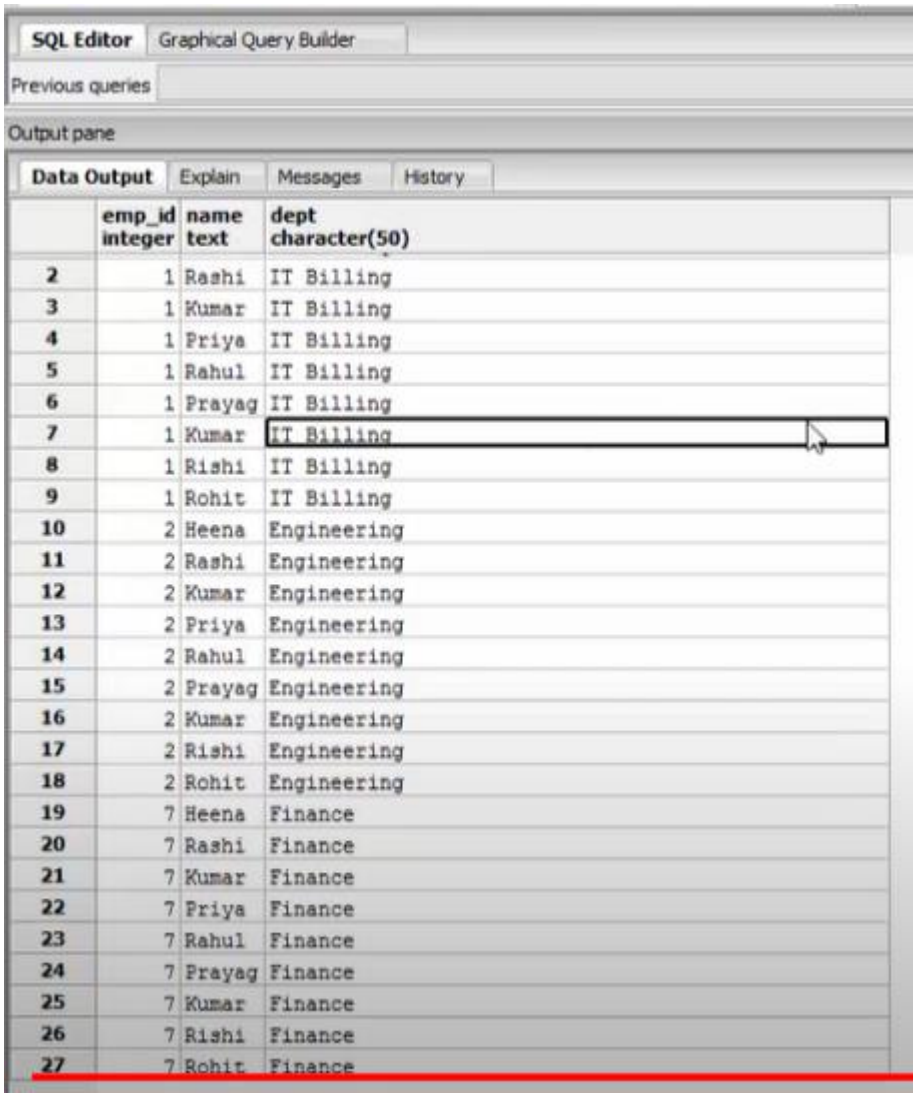

SQL Editor
Graphical Query Builder
Previous queries
select * from company
Output pane
Data Output
Explain
Messages
History
id integer
name text
age integer
address character(50)
salary real
3 9 Kumar 40 Delhi 2000
4 4 Priya 24 Delhi 3000
5 7 Rahul 20 Kolkata 5000
6 11 Prayag 33 Kolkata 9000
7 13 Kumar 35 Bengaluru 4000
8 10 Rishi 34 Mumbai 8000
9 12 Rohit 34 Chennai 4000



IF TABLE X HAS x ROWS AND TABLE Y HAS y ROWS THEN THE TABLE FORMED BY THEIR CROSS JOIN HAS x*y ROWS.



THE ABOVE TABLE FORMED BY CROSS JOIN HAS 9*3=27 ROWS.



	emp_id integer	name text	dept character(50)
2	1	Rashi	IT Billing
3	1	Kumar	IT Billing
4	1	Priya	IT Billing
5	1	Rahul	IT Billing
6	1	Prayag	IT Billing
7	1	Kumar	IT Billing
8	1	Rishi	IT Billing
9	1	Rohit	IT Billing
10	2	Heena	Engineering
11	2	Rashi	Engineering
12	2	Kumar	Engineering
13	2	Priya	Engineering
14	2	Rahul	Engineering
15	2	Prayag	Engineering
16	2	Kumar	Engineering
17	2	Rishi	Engineering
18	2	Rohit	Engineering
19	7	Heena	Finance
20	7	Rashi	Finance
21	7	Kumar	Finance
22	7	Priya	Finance
23	7	Rahul	Finance
24	7	Prayag	Finance
25	7	Kumar	Finance
26	7	Rishi	Finance
27	7	Rohit	Finance

SQL Editor | Graphical Query Builder

Previous queries

```
select name,age,address,dept from company inner join department on company.id=department.id
```

Output pane

Data Output | Explain | Messages | History

	name text	age integer	address character(50)	dept character(50)
1	Heena	16	Mumbai	Engineering

SQL Editor | Graphical Query Builder

Previous queries

```
select name,age,address,dept from company c , department d
where c.id=d.id
```

Output pane

Data Output | Explain | Messages | History

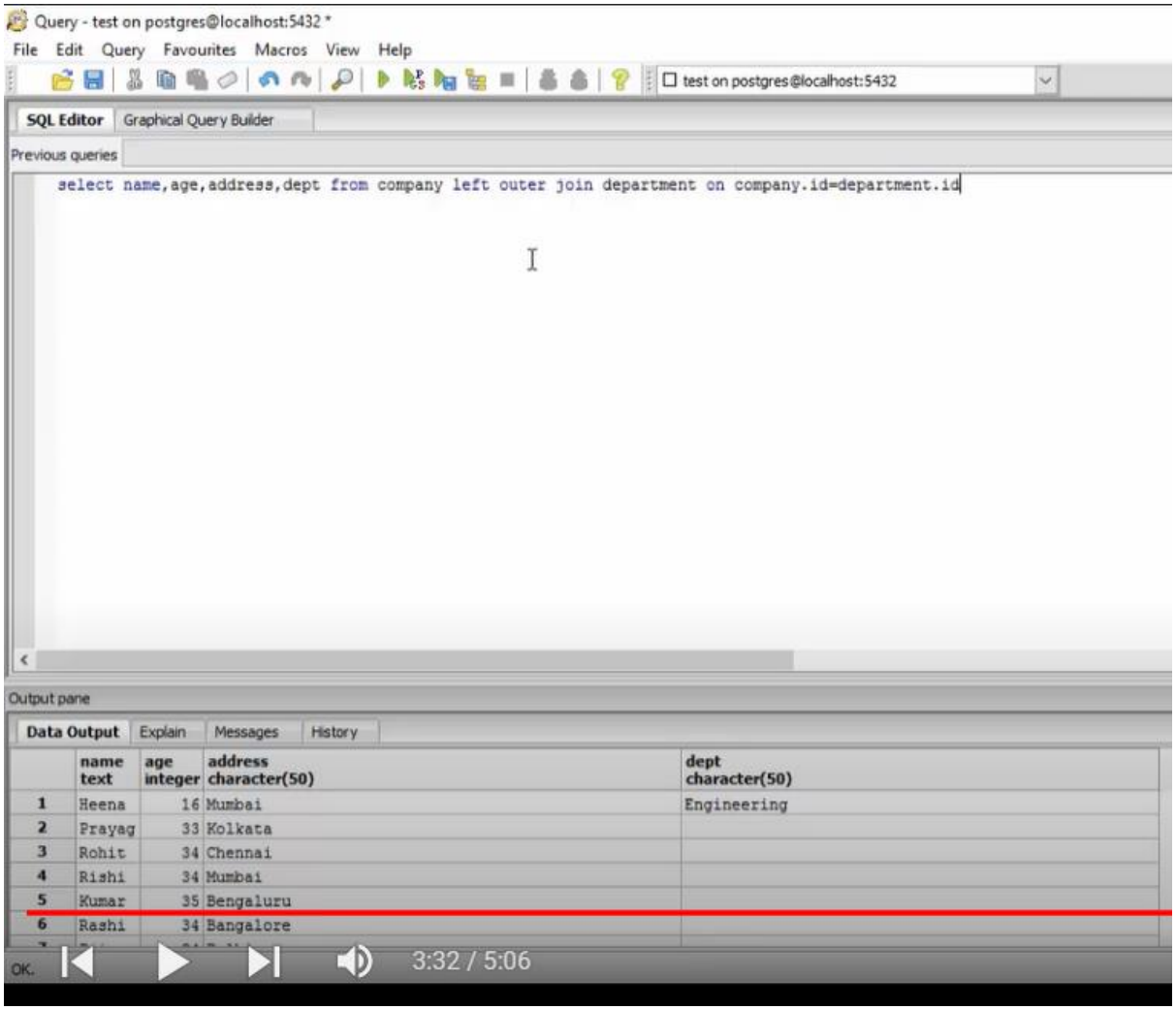
	name text	age integer	address character(50)	dept character(50)
1	Heena	16	Mumbai	Engineering

IN THE ABOVE 2 QUERIES NAME,AGE AND ADDRESS ARE FROM COMPANY TABLE AND DEPT FROM DEPARTMENT TABLE.

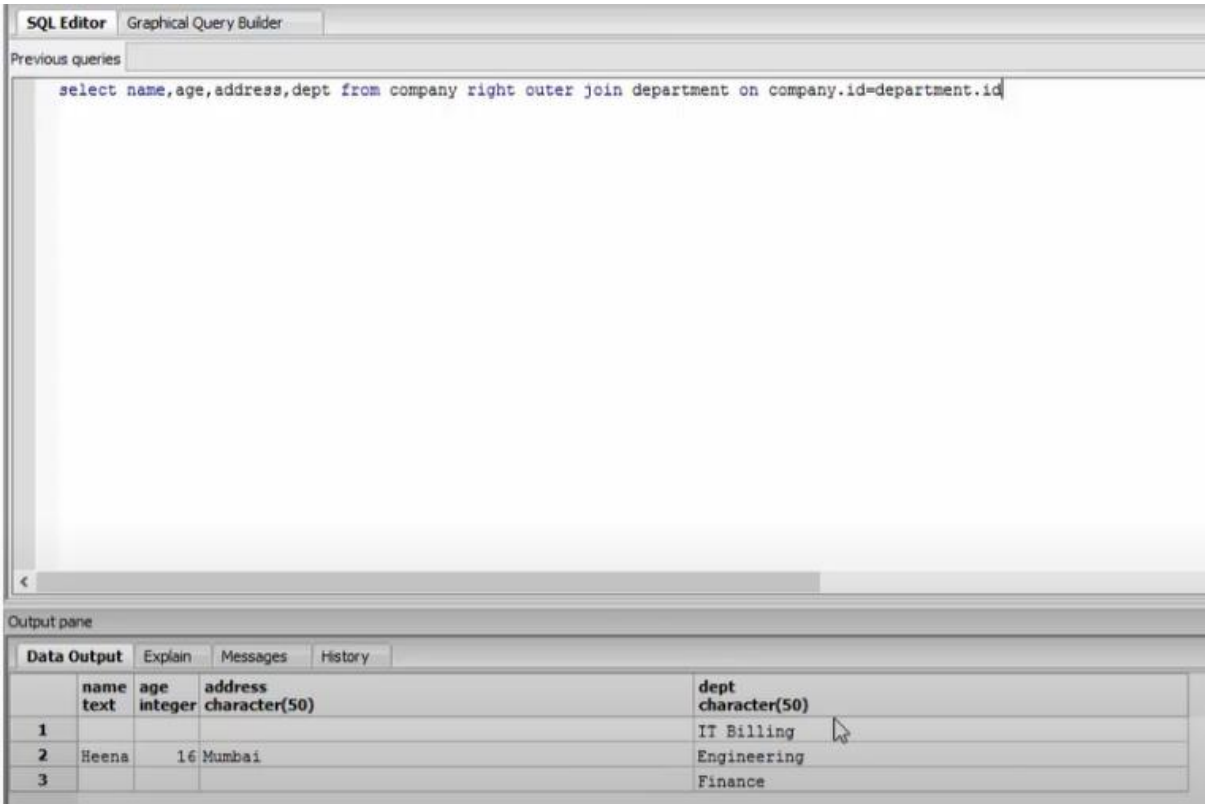
INNER JOIN RETURNS ONLY THE COLUMNS COMMON TO BOTH TABLES.

IN OUTER JOIN FIRST INNER JOIN IS DONE THEN THE VALUES THAT DON'T MATCH ARE REPLACED BY NULL VALUES.

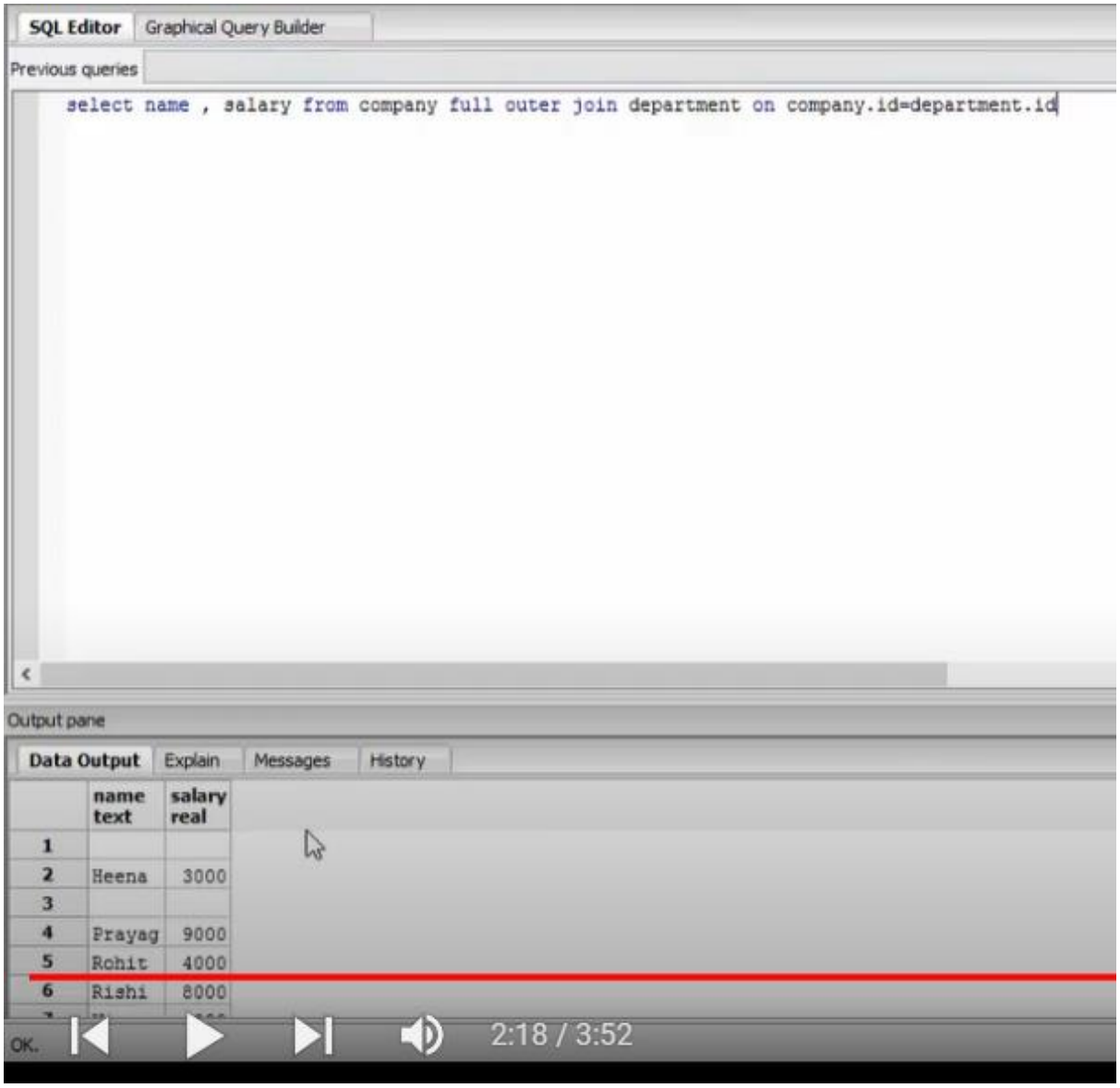
X LEFT OUTER JOIN Y → MEANS ALL COLUMN VALUES OF X IS FULLY DISPLAYED BUT THOSE VALUES IN Y THAT DON'T MATCH WITH ANY ROW ARE KEPT NULL.

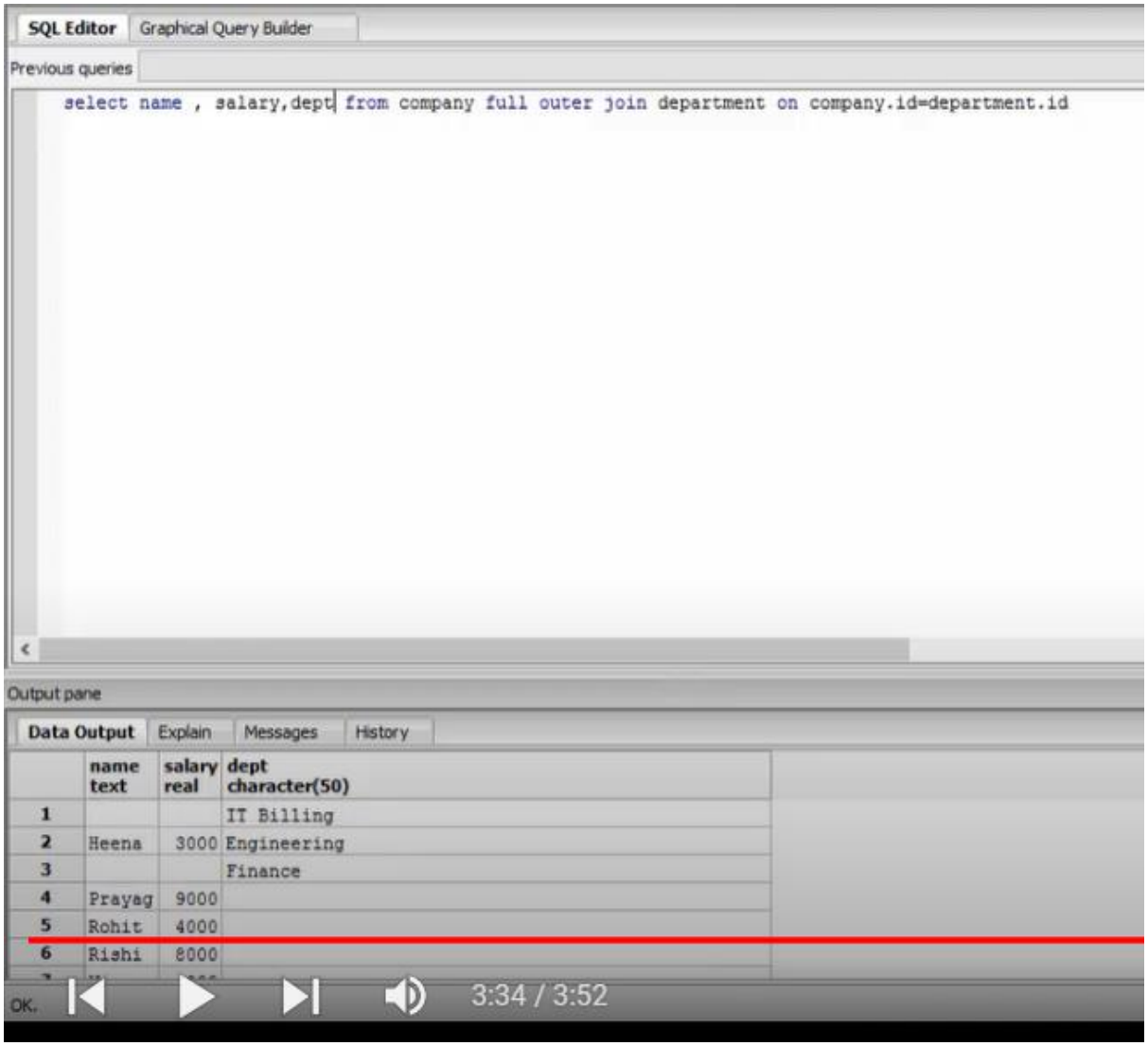


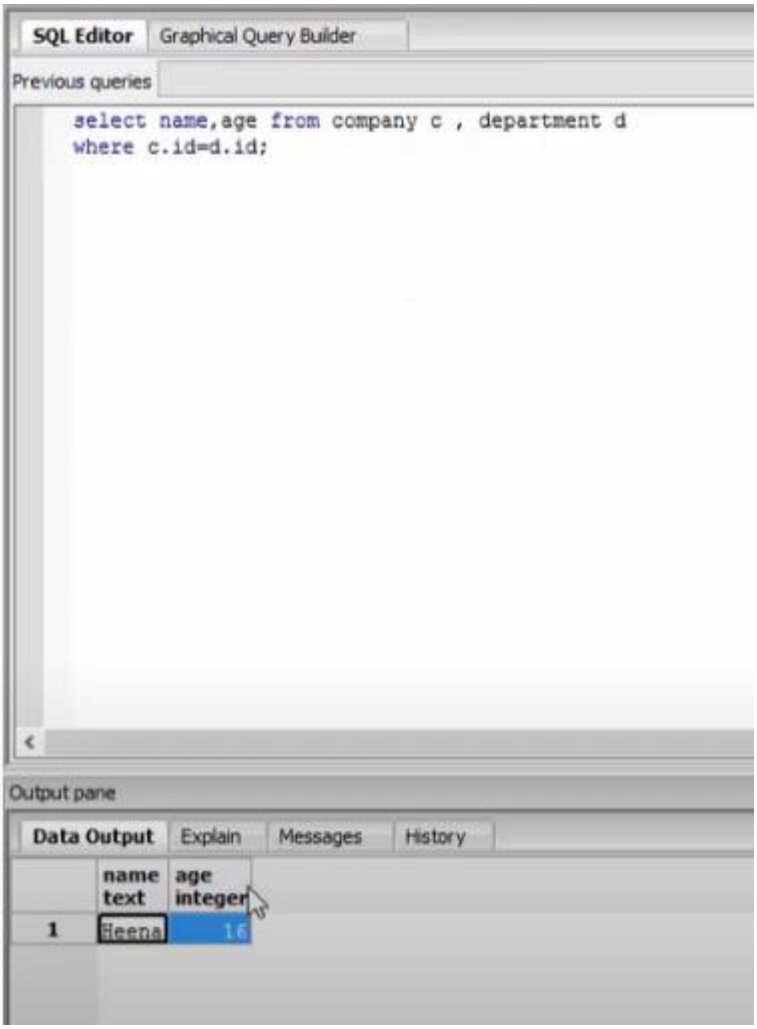
X RIGHT OUTER JOIN Y MEANS Y IS FULLY DISPLAYED BUT X VALUES THAT ARE NOT NOT COMMON TO ANY ROW OF Y IS DISPLAYED AS NULL.



FULL OUTER JOIN HAS ALL VALUES FROM TABLE X AND Y AND EVEN FOR THOSE VALUES WHERE X OR Y IS NULL OUTPUT IS DISPLAYED.

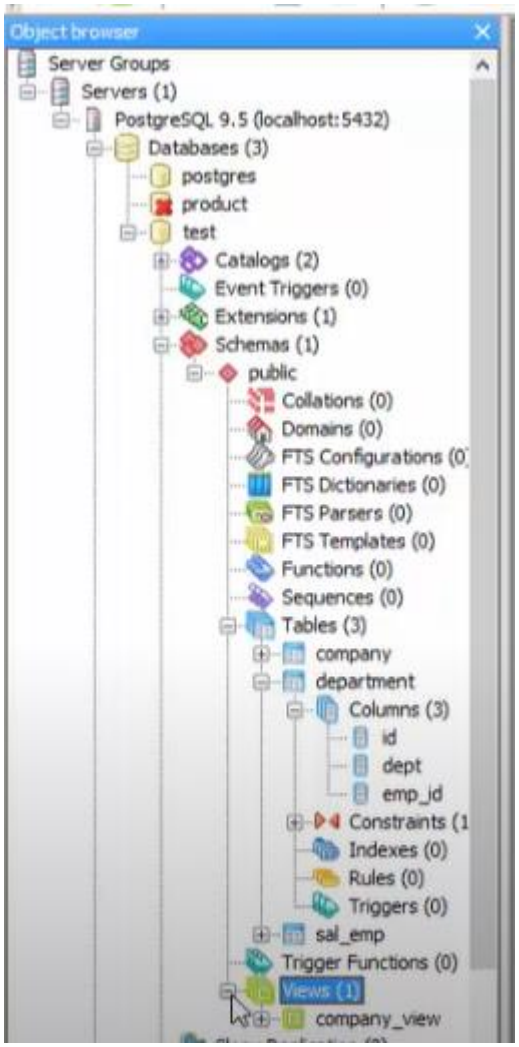

SQL Editor
Graphical Query Builder
Previous queries
select name , salary from company full outer join department on company.id=department.id
Output pane
Data Output
Explain
Messages
History
name text
salary real
1
2 Heena 3000
3
4 Prayag 9000
5 Rohit 4000
6 Rishi 8000
OK.
2:18 / 3:52

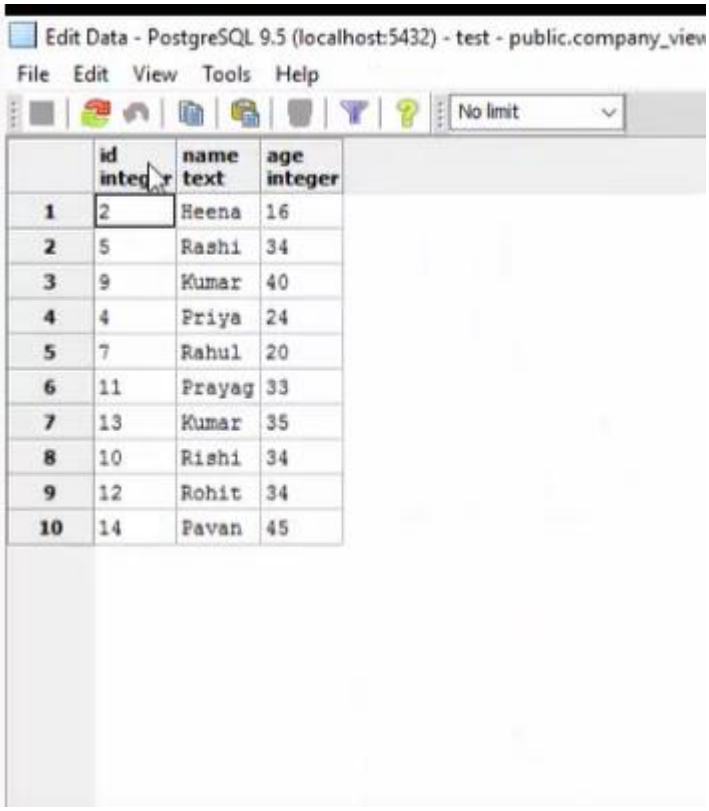

SQL Editor
Graphical Query Builder
Previous queries
select name , salary,dept from company full outer join department on company.id=department.id
Output pane
Data Output
Explain
Messages
History
name text
salary real
dept character(50)
1 IT Billing
2 Heena 3000 Engineering
3 Finance
4 Prayag 9000
5 Rohit 4000
6 Rishi 8000
OK.
3:34 / 3:52


SQL Editor
Graphical Query Builder
Previous queries
select name,age from company c , department d
where c.id=d.id;
Output pane
Data Output
Explain
Messages
History
name
text
age
integer
1
Heena
16

VIEWS ARE PSEUDO TABLES I.E NOT REAL TABLE OR PART OF TABLE . IT IS SUBSET OF TABLE AS IT SELECTS SOME COLUMNS AND ROWS FROM THE TABLE.


SQL Editor
Graphical Query Builder
Previous queries
create view company_view as
select id,name,age from company;
Output pane
Data Output
Explain
Messages
History
Query returned successfully with no result in 313 msec.


Object browser
Server Groups
Servers (1)
PostgreSQL 9.5 (localhost:5432)
Databases (3)
postgres
product
test
Catalogs (2)
Event Triggers (0)
Extensions (1)
Schemas (1)
public
Collations (0)
Domains (0)
FTS Configurations (0)
FTS Dictionaries (0)
FTS Parsers (0)
FTS Templates (0)
Functions (0)
Sequences (0)
Tables (3)
company
department
Columns (3)
id
dept
emp_id
Constraints (1
Indexes (0)
Rules (0)
Triggers (0)
sal_emp
Trigger Functions (0)
Views (1)
company_view



IN VIEWS WE CAN'T UPDATE, DELETE OR MODIFY DATA.

UNION OR UNION ALL IS USED TO COMBINE 2 SELECT STATEMENTS.

IN UNION IF WE HAVE ANY DUPLICATE ROWS THEN IT CAN BE COMBINED INTO SINGLE ROW.

IN UNION ALL WE CAN SEE DUPLICATE ROWS AND THEY ARE PART OF RESULT.

SQL Editor | Graphical Query Builder

Previous queries

```
select company.id,name,department from company inner join department on company.id=department.id
```

Output pane

Data Output | Explain | Messages | History

	id integer	name text	department character(100)
1	2	Heena	IT Billing

SQL Editor | Graphical Query Builder

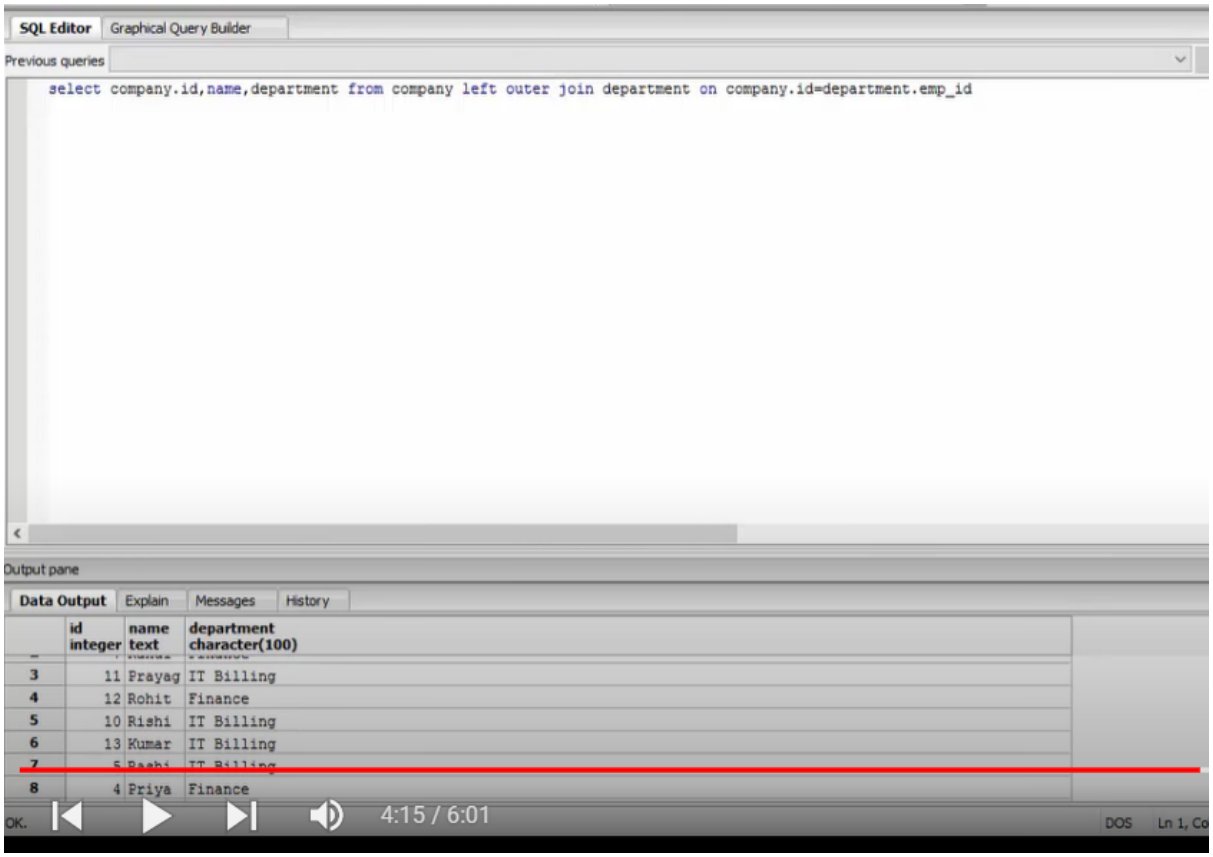
Previous queries

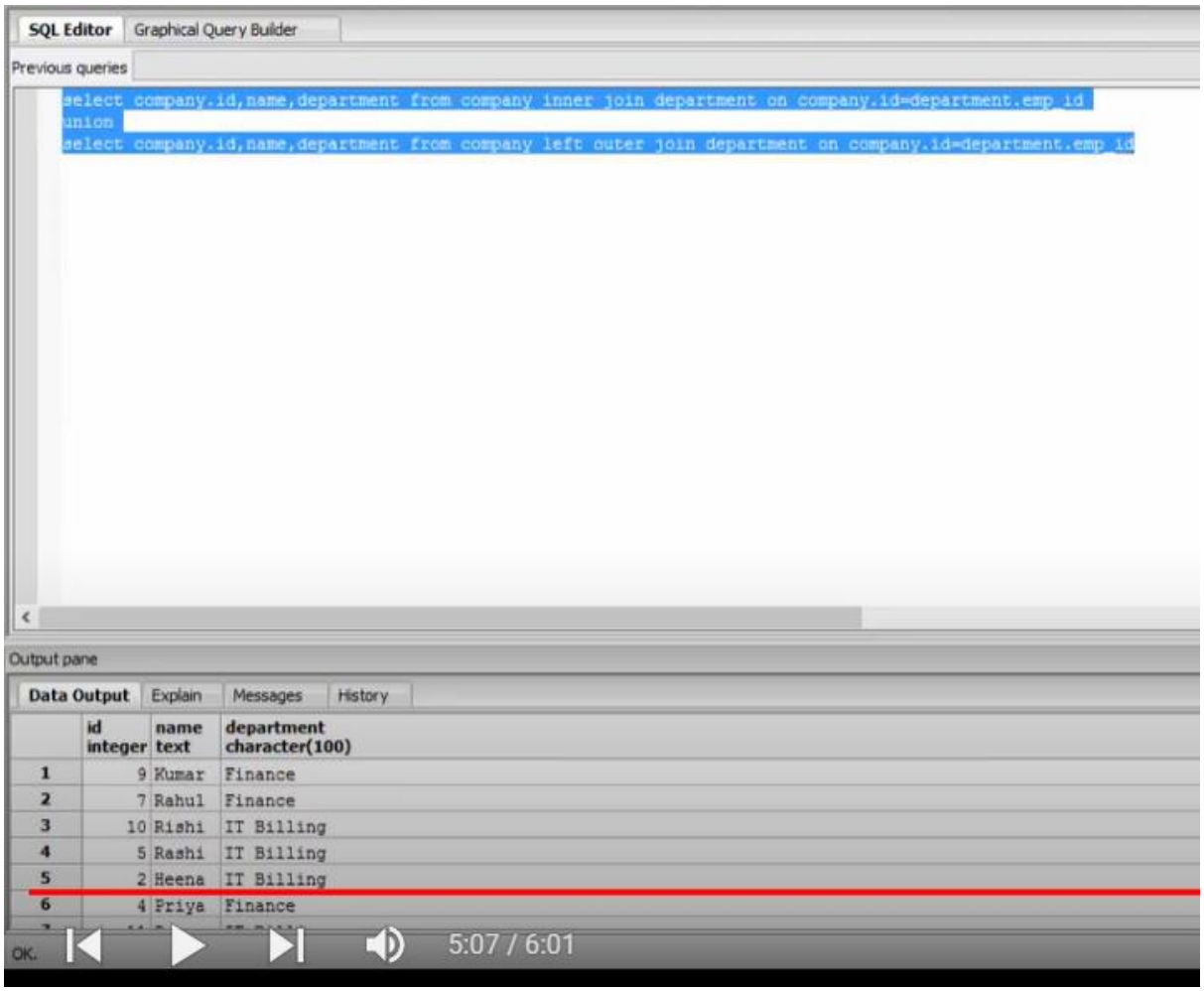
```
select company.id,name,department from company inner join department on company.id=department.emp_id
```

Output pane

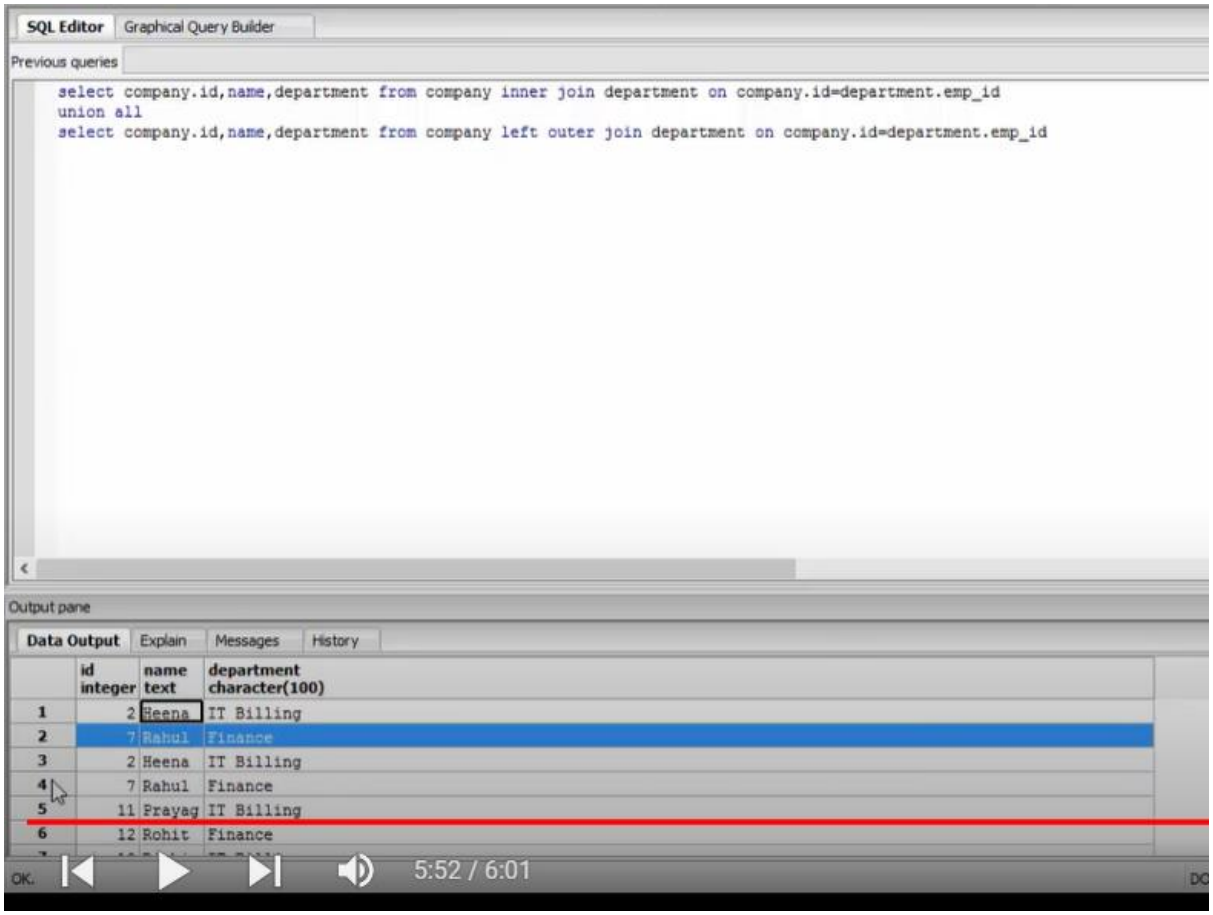
Data Output | Explain | Messages | History

	id integer	name text	department character(100)
1	2	Heena	IT Billing
2	7	Rahul	Finance


SQL Editor
Graphical Query Builder
Previous queries
select company.id,name,department from company left outer join department on company.id=department.emp_id
Output pane
Data Output
Explain
Messages
History
id integer
name text
department character(100)
3 11 Prayag IT Billing
4 12 Rohit Finance
5 10 Rishi IT Billing
6 13 Kumar IT Billing
8 4 Priya Finance
4:15 / 6:01
DOS


SQL Editor
Graphical Query Builder
Previous queries
select company.id,name,department from company inner join department on company.id=department.emp_id
union
select company.id,name,department from company left outer join department on company.id=department.emp_id
Output pane
Data Output
Explain
Messages
History
id integer
name text
department character(100)
1 9 Kumar Finance
2 7 Rahul Finance
3 10 Rishi IT Billing
4 5 Rashi IT Billing
5 2 Heena IT Billing
6 4 Priya Finance
5:07 / 6:01

THE ABOVE QUERY IS FOR COMBINING OUTPUTS DUE TO INNER JOIN AND LEFT OUTER JOIN.

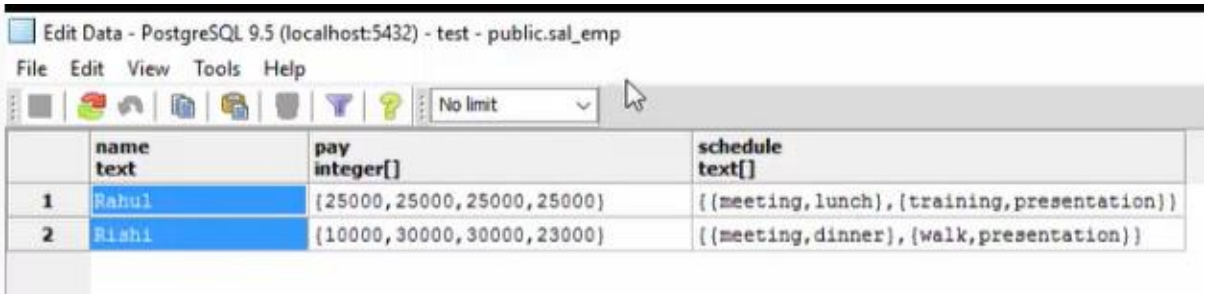


UNION ALL SHOWS DUPLICATE ROWS ALSO. EG IN ABOVE TABLE RAHUL,HEENA ROWS ARE DUPLICATED.

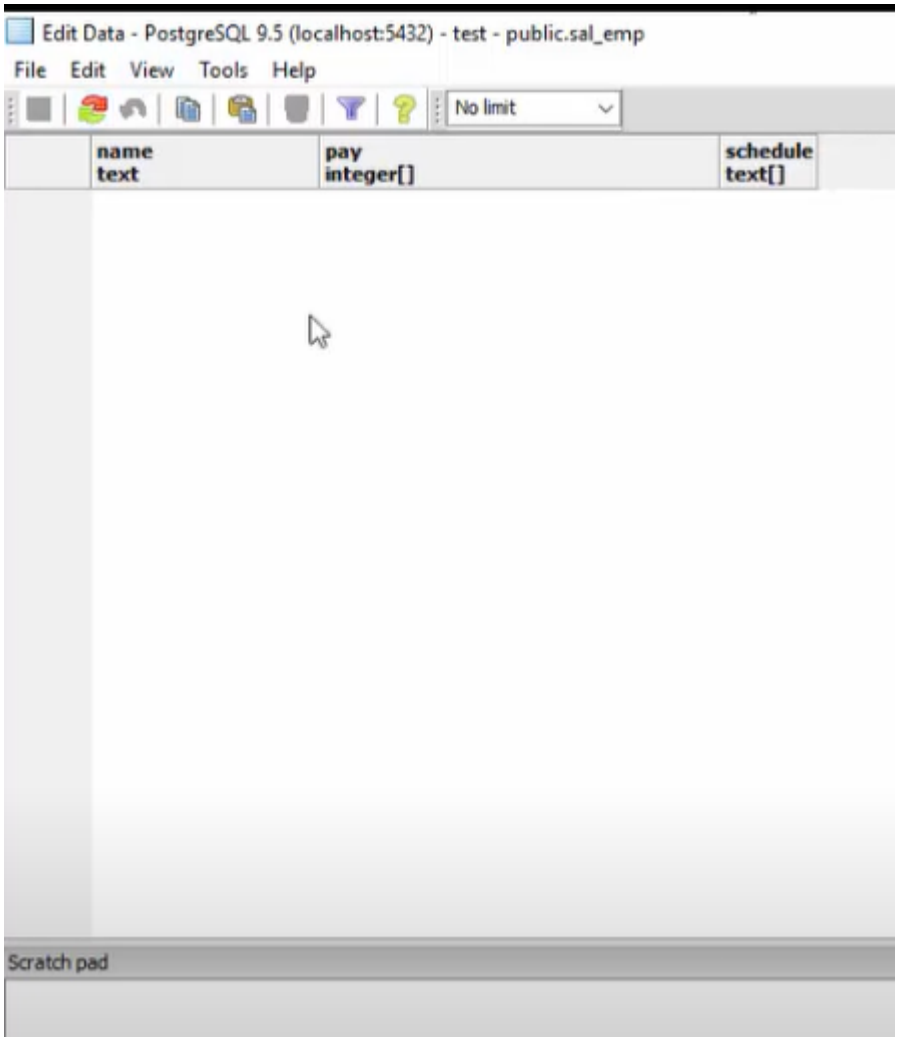
UNION DOES NOT HAVE DUPLICATE ROWS BUT UNION ALL HAS DUPLICATE ROWS.

TRUNCATE IS USED TO DELETE ALL DATA INSIDE ROWS AND COLUMNS OF TABLE BUT STRUCTURE OF TABLE REMAIN AS IT IS.

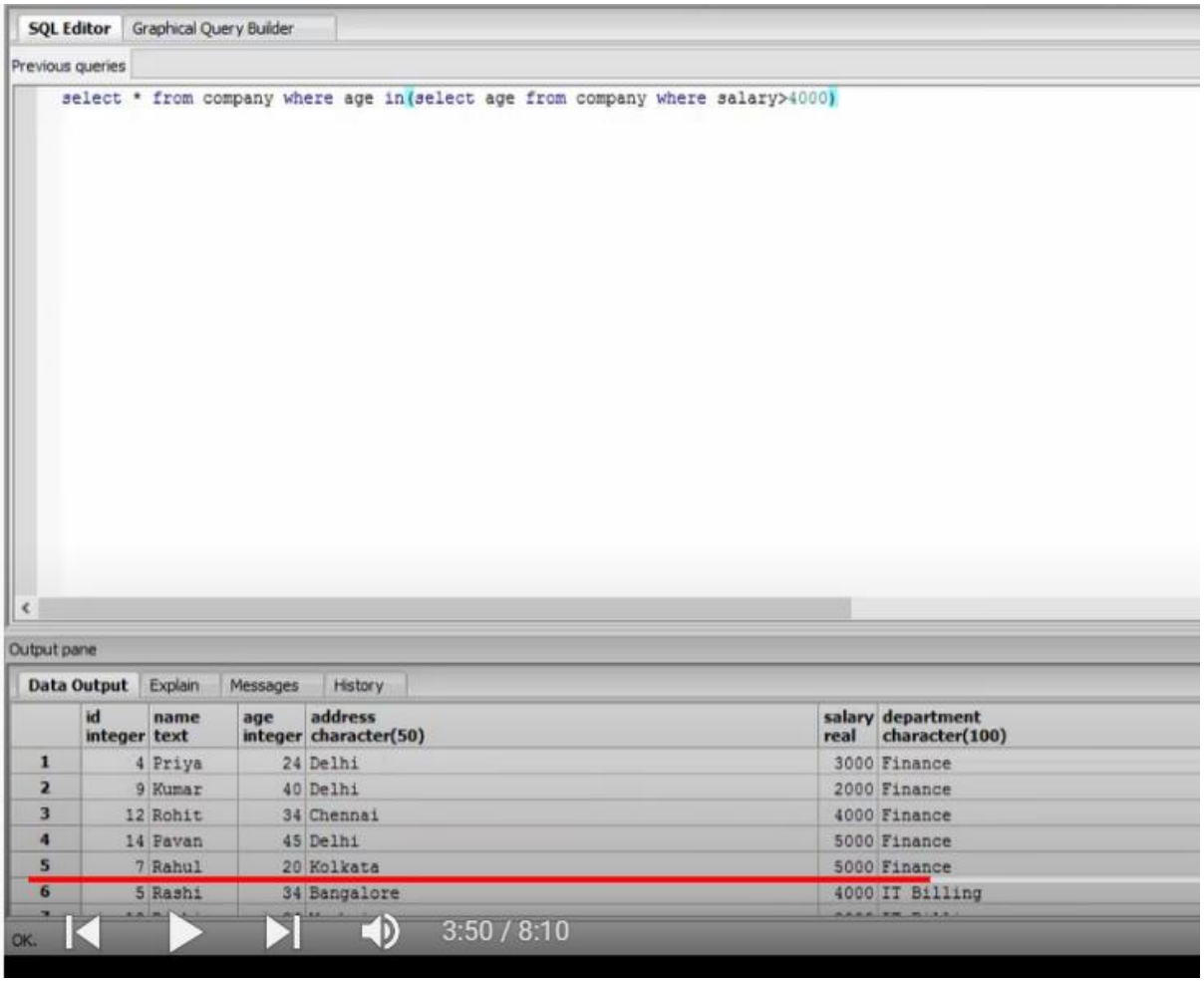
DROP TABLE -> USED TO DELETE TABLE STRUCTURE AS WELL AS DATA.




SQL Editor
Graphical Query Builder
Previous queries
truncate table sal_emp
Output pane
Data Output
Explain
Messages
History
Query returned successfully with no result in 170 msec.

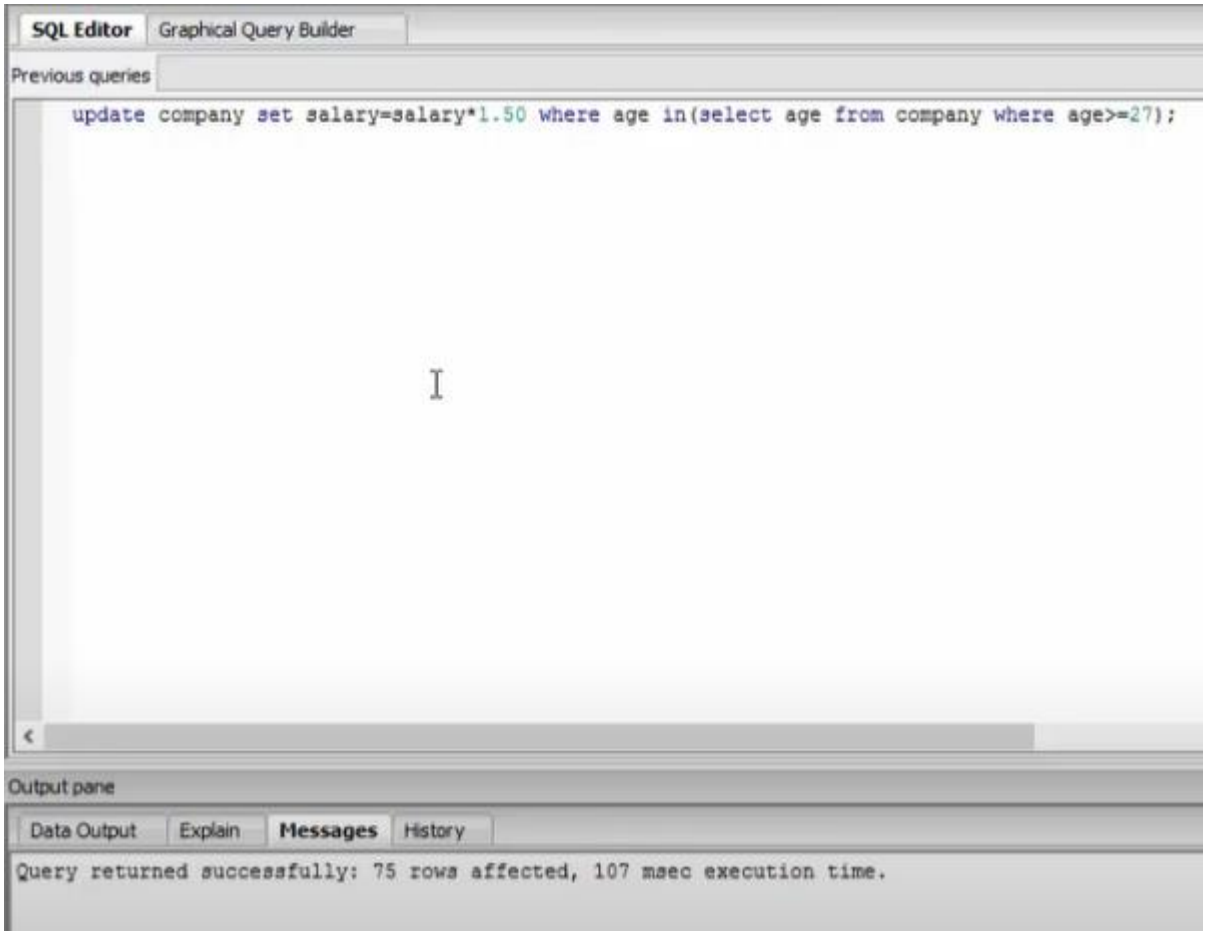


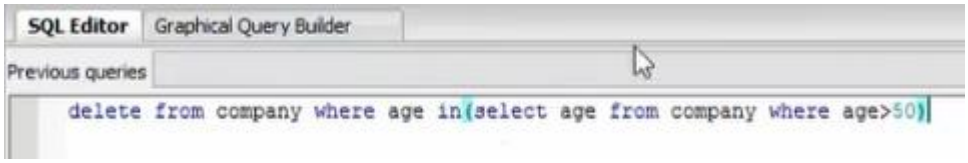
TRUNCATED CASCADED MEANS IF ANY FOREIGN KEY ASSOCIATED WITH PARENT TABLE THEN IF PARENT TABLE IS DELETED THEN FOREIGN KEY ASSOCIATED IS ALSO DELETED.



USED SUBQUERIES IN THE ABOVE.

WE CAN ALSO SEE PEOPLE WITH < 4000 SALARY. this is due to the subquery , and the values of the ages that are being returned , for example in the first query there maybe a person aged 25 who has a salary above 4000 , hence 25 is included in the list , while in the outer query all people are selected whose age is 25. Hence there is no restriction of the salary in the outer query.





SOME POSTGRE SQL IMP COMMANDS REVISED—

SELECT * FROM employees;

SELECT * FROM employees WHERE department = ‘Sports’;

##below finding total salary for each department.

SELECT SUM(salary) as total_salary, department FROM employees GROUP BY department;

##below finding departments having > 50 employees.

SELECT count(*) as total_employee,department FROM employees GROUP BY department HAVING COUNT(*) > 50;

Find total salary by each department and sort **descending by** total salary column.

```
SELECT SUM(salary) as total_salary,department FROM employees
GROUP BY department
ORDER BY total_salary desc
```

Write a query that finds the first 5 employees with their first_name, department and salary and sorted by their first_name.

```
SELECT first_name,department,salary from employees
ORDER BY first_name
LIMIT 5
```

Note: Default Order By clause sorts the result in ASCENDING order.

5. Date Functions

In PostgreSQL, you can easily extract values from date columns. You will see the most used date functions below.

```
SELECT
date_part('year',hire_date) as year,
date_part('month',hire_date) as month,
date_part('day',hire_date) as day,
date_part('dow',hire_date) as dayofweek,
to_char(hire_date, 'Dy') as day_name,
to_char(hire_date,'Month') as month_name,
hire_date
FROM employees
```

year double precision	month double precision	day double precision	dayofweek double precision	day_name text	month_name text	hire_date date
2006	4	20	4	Thu	April	2006-04-20
2009	1	26	1	Mon	January	2009-01-26
2010	5	17	1	Mon	May	2010-05-17
2014	8	2	6	Sat	August	2014-08-02
2003	1	14	2	Tue	January	2003-01-14
2003	6	8	0	Sun	June	2003-06-08

Date Functions Output

Inner, Left or Right Joins

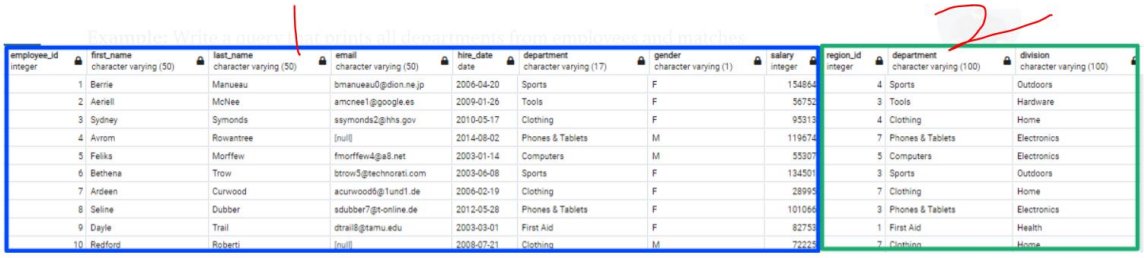
Inner Join clause creates a new table (not physical) by combining rows that have matching values in two or more tables.

Example: Query all employee information and their divisions of the department.

```
SELECT * FROM employees e
INNER JOIN departments d
ON e.department = d.department
```

employee_id integer	first_name character varying (50)	last_name character varying (50)	email character varying (50)	hire_date date	department character varying (17)	gender character varying (1)	salary integer	region_id integer	department character varying (100)	division character varying (100)
1	Berrie	Manueau	bmanueau0@dion.ne.jp	2006-04-20	Sports	F	154864	4	Sports	Outdoors
2	Aeriell	McNee	amcnee1@google.es	2009-01-26	Tools	F	56752	3	Tools	Hardware
3	Sydney	Symonds	ssymonds2@hhs.gov	2010-05-17	Clothing	F	95313	4	Clothing	Home
4	Avrom	Rowantree	[null]	2014-08-02	Phones & Tablets	M	119674	7	Phones & Tablets	Electronics
5	Feliks	Morffew	fmorffew4@a8.net	2003-01-14	Computers	M	55307	5	Computers	Electronics
6	Bethena	Trow	btrow5@technorati.com	2003-06-08	Sports	F	134501	3	Sports	Outdoors
7	Ardeen	Curwood	acurwood6@1und1.de	2006-02-19	Clothing	F	28995	7	Clothing	Home
8	Seline	Dubber	sdubber7@t-online.de	2012-05-28	Phones & Tablets	F	101066	3	Phones & Tablets	Electronics
9	Dayle	Trail	dtrail8@tamu.edu	2003-03-01	First Aid	F	82753	1	First Aid	Health
10	Redford	Roberti	[null]	2008-07-21	Clothing	M	72228	7	Clothing	Home

The **Left Join** returns all rows from the left table and the matching rows from the right table. If no matching rows are found in the right table, **NULL** is used. (vice versa for Right Join)



employee_id integer	first_name character varying (50)	last_name character varying (50)	email character varying (50)	hire_date date	department character varying (17)	gender character varying (1)	salary integer	region_id integer	department character varying (100)	division character varying (100)
1	Berrie	Manueau	bmanueau0@dion.ne.jp	2006-04-20	Sports	F	154864	4	Sports	Outdoors
2	Aeriell	McNee	amcnee1@google.es	2009-01-26	Tools	F	56752	3	Tools	Hardware
3	Sydney	Symonds	ssymonds2@hhs.gov	2010-05-17	Clothing	F	95313	4	Clothing	Home
4	Avrom	Rowantree	[null]	2014-08-02	Phones & Tablets	M	119674	7	Phones & Tablets	Electronics
5	Feliks	Morffew	fmorffew4@a8.net	2003-01-14	Computers	M	55307	5	Computers	Electronics
6	Bethena	Trow	btrow5@technorati.com	2003-06-08	Sports	F	134501	3	Sports	Outdoors
7	Ardeen	Curwood	acurwood6@1und1.de	2006-02-19	Clothing	F	28995	7	Clothing	Home
8	Seline	Dubber	sdubber7@t-online.de	2012-05-28	Phones & Tablets	F	101066	3	Phones & Tablets	Electronics
9	Dayle	Trail	dtrail8@tamu.edu	2003-03-01	First Aid	F	82753	1	First Aid	Health
10	Redford	Roberti	[null]	2008-07-21	Clothing	M	72228	7	Clothing	Home

1 IS EMPLOYEE TABLE AND 2 IS DEPARTMENT TABLE.

Example: Write a query that prints all departments from employees and matches departments from the department table.

```
SELECT e.department,d.department FROM employees e
LEFT JOIN departments  d
ON e.department = d.department
```

department character varying (17)	department character varying (100)
Sports	Sports
Automotive	Automotive
Camping	[null]
Clothing	Clothing
Computers	Computers
Tools	Tools

Example: Query first_name, department, and salary of each employee and also maximum salary given.

```
SELECT first_name,department,salary,(SELECT max(salary)  FROM
employees)
FROM employees
```

first_name character varying (50)	department character varying (17)	salary integer	max integer
Berrie	Sports	154864	166976
Aeriell	Tools	56752	166976
Sydney	Clothing	95313	166976
Avrom	Phones & Tablets	119674	166976
Feliks	Computers	55307	166976
Bethena	Sports	134501	166976

QUE

Example: Write a query that finds the first name, salary, department, and average salary by department.

```
SELECT first_name,salary,department,round((SELECT AVG(salary)
    FROM employees e2
    WHERE e1.department = e2.department
    GROUP BY department )) as avg_salary_by_department
FROM employees e1
WHERE salary > (SELECT AVG(salary)
    FROM employees e2
    WHERE e1.department = e2.department
    GROUP BY department )
ORDER BY salary
```

first_name character varying (50)	salary integer	department character varying (17)	avg_salary_by_department numeric
Jobi	80331	Maintenance	76793
Dexter	80657	Jewelry	80576
Weylin	86194	Clothing	83933
Worth	86252	Tools	83530
Basile	86269	Jewelry	80576
Port	86352	Clothing	83933

Example: Write a query to print the first name, salary, and average salary as well as a new column that shows whether employees' salary is higher than average or not.

```
SELECT first_name,salary,(SELECT ROUND(AVG(salary)) FROM employees) as
average_salary,
(CASE WHEN salary > (SELECT AVG(salary) FROM employees) THEN
'higher_than_average'
ELSE 'lower_than_average' END) as Salary_Case
FROM employees
```

first_name character varying (50)	salary integer	average_salary numeric	salary_case text
Berrie	154864	91572	higher_than_average
Aeriell	56752	91572	lower_than_average
Sydney	95313	91572	higher_than_average
Avrom	119674	91572	higher_than_average
Feliks	55307	91572	lower_than_average
Bethena	134501	91572	higher_than_average

The following query will give you the average salary for each department.

```
SELECT first_name,salary,department,
ROUND(AVG(salary) OVER(PARTITION BY department)) as avg_sales_by_dept
FROM employees
ORDER BY salary DESC
```

first_name character varying (50)	salary integer	department character varying (17)	avg_sales_by_dept numeric
Jacklyn	166976	Clothing	83933
Carissa	166765	Music	92361
Riley	166569	Camping	97302
Lauren	166016	Pharmacy	93372
Lucy	165660	Sports	87832
Barby	164588	Clothing	83933

10.2.Ranking the Values

The Rank() function is a window function that assigns a rank to each row within a partition of a result set.

The following example orders the table by the salary (descending). A rank value of 1 is the highest salary value.

```
SELECT first_name,salary,RANK() OVER(ORDER BY salary DESC)
FROM employees
```

first_name character varying (50)	salary integer	rank bigint
Jacklyn	166976	1
Carissa	166765	2
Riley	166569	3
Lauren	166016	4
Lucy	165660	5
Barby	164588	6

Understanding the Basics

The basics are more than just the basic commands, what matters most when answering difficult questions is the ***order of operations!*** Make sure you know this by heart. If you get in the habit of writing each line of a query in this order rather than from top to bottom I promise you'll be a SQL wizard in no time!

1. ***Order of Operations***

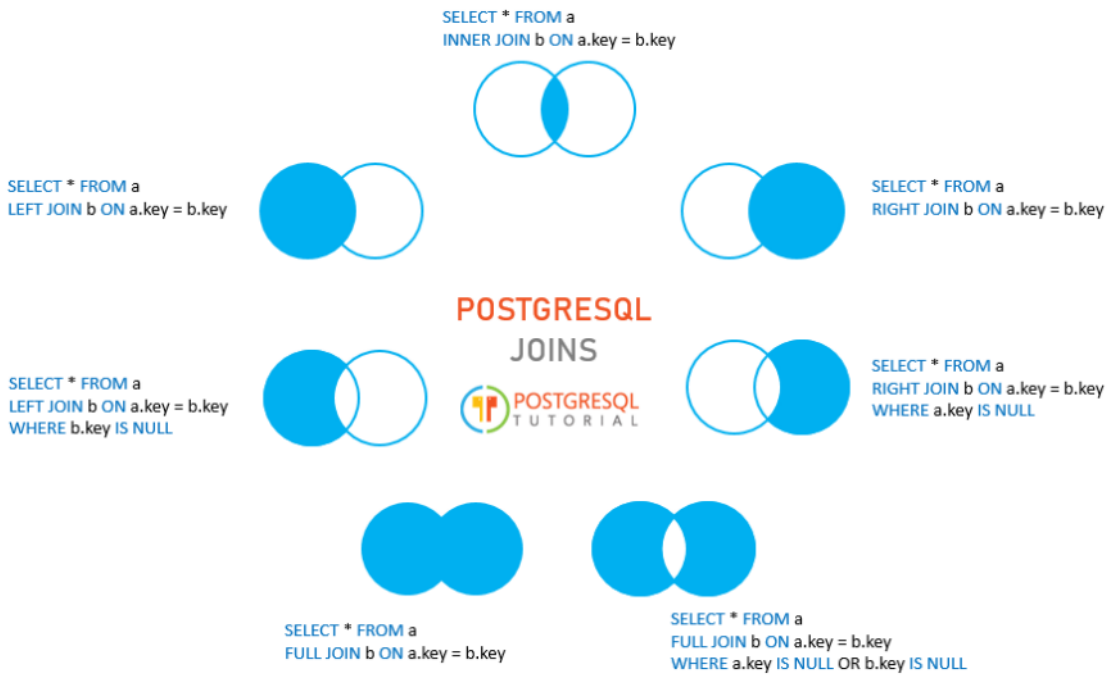
FROM : the table your data is in
WHERE: filters the table down to the contents where a certain condition is met
GROUPBY: clusters the data into groups
SELECT: actually pulls the data
ORDER BY: orders the data
LIMIT: puts a cap on the number of returns

Query Order

SELECT
FROM
WHERE
GROUP BY
ORDER BY
LIMIT

You might wonder why the code executes in one order but is written in another. Unfortunately, this is just the way it's designed.

2. Joins



Almost every SQL query you'll see in entry-level interviews will use inner joins — if you must guess, go for that.

Easy Questions

You have a table called *foods* and it looks like this:

	id	name	restaurant_id	calories	fat	carbs	created_at	updated_at
0	26	Blue Raspberry Slushee	1.0	370.0	0.0	91.0	2017-04-17 19:43:29.627441	2017-04-17 19:43:29.627441
1	27	Blue Raspberry Slushee	1.0	570.0	0.0	142.0	2017-04-17 19:43:29.634462	2017-04-17 19:43:29.634462
2	28	Blue Raspberry Slushee	1.0	740.0	0.0	182.0	2017-04-17 19:43:29.641432	2017-04-17 19:43:29.641432
3	29	Breaded Onion Rings (Large)	1.0	480.0	27.0	62.0	2017-04-17 19:43:29.653546	2017-04-17 19:43:29.653546
4	30	Breaded Onion Rings (Regular)	1.0	350.0	16.0	45.0	2017-04-17 19:43:29.659697	2017-04-17 19:43:29.659697

Q1. Display the entire table.

Q2. Display the `name` (renamed "food") and `calories` from the foods schema. Sort it by calories first and then by name in alphabetical order.

Answers

Q1.

```
SELECT *
FROM foods as f;
```

Q2.

```
SELECT f.name as “food”, f.fat
FROM foods f
ORDER BY f.calories DESC, f.name ASC;
```

Medium Questions

Q3. Display the `name`, `carbs`, and total `calories` of the Top 5 foods where carbs make up between 25% and 75% of the total calories from fat

Q4. You are told there is another table called *restaurants* that has a foreign key which links to the *foods* column named `id`. Name that column appropriately and join the tables.

Q3.

```
SELECT f.name, f.carbs, f.calories
FROM foods as f
WHERE 100*f.carbs/(f.calories+0.001) BETWEEN 25 and 75
ORDER BY 100*f.carbs/(f.calories+0.001) DESC
LIMIT 5;
```

Q4.

```
SELECT f.id
FROM foods as f
INNER JOIN restaurants as r
ON f.id=r.foods_id;
```

This question tests your knowledge of appropriate naming schemas, note that foreign keys should reference the table they are a key for!

Hard Questions

Now you have a table called *salaries* that looks like this:

name	salary	date of hire
Amanda	50,000	05/20/18
Carlos	60,000	02/22/17
Ben	55,000	06/30/18

salaries

Q5. Return the second highest salary. (Hint — — — use a subquery)

Q6 (from CodeWars).

#	Pokémon	HP	Attack	Defense
493	Arceus	120	120	120
646	Kyurem (Black Kyurem)	125	170	100
646	Kyurem (White Kyurem)	125	120	90
249	Lugia	106	90	130
644	Zekrom	100	150	120
483	Dialga	100	120	120

644	 Zekrom	100	150	120
483	 Dialga	100	120	120

You have arrived at the Celadon Gym to battle Erika for the Rainbow Badge.

She will be using Grass-type Pokemon. Any fire pokemon you have will be strong against grass, but your water types will be weakened. The multipliers table within your Pokedex will take care of that.

Using the following tables, return the *pokemon_name, modifiedStrength* and *element* of the Pokemon whose strength, after taking these changes into account, is greater than or equal to 40, ordered from strongest to weakest.

pokemon schema

- id
- pokemon_name
- element_id
- str

multipliers schema

- id
- element
- multiplier

Q5.

SELECT Max(salary)
FROM salaries
WHERE salary NOT IN (SELECT Max(salary) FROM salaries);

Q6.

```
SELECT pkmn.pokemon_name,
pkmn.str*mult.multiplier as modifiedStrength,
mult.element
FROM pokemon as pkmn
INNER JOIN multipliers as mult ON pkm.element_id = mult.id
WHERE pkm.str*mult.multipler ≥ 40
ORDER BY modifiedStrength DESC
```