**Psql shell**

* Server [localhost]: **---just enter---**
* Database [postgres]: **---just enter---**
* Port [5432]: **---just enter---**
* Username [postgres]: **---just enter---**
* Password for user postgres: **---type password---**

**Response :**

psql (16.2)

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=# select version(); **---to check version---**

**Response :**

version

------------------------------------------------------------

PostgreSQL 16.2, compiled by Visual C++ build 1937, 64-bit

(1 row)

* postgres=# \l **---shows the database that already there---**

List of databases

Name | Owner | Encoding | Locale Provider | Collate | Ctype | ICU Locale | ICU Rules | Access privileges

-----------+----------+----------+-----------------+--------------------+--------------------+------------+-----------+-----------------------

* postgres=# create database sql\_demo; **---create database---**

**Response :**

CREATE DATABASE

* postgres=# \c sql\_demo; **---connect to database---**

**Response :**

You are now connected to database "sql\_demo" as user "postgres".

**Connect PostgresSql using pg-admin**

In pg-admin go to your database 🡪right click 🡪select query tool 🡪 type query 🡪 type query & select query & execute or f5 press it.

* select version(); **---to check version---**
* select 5\*3; **---basic operation---**

Same query can be run on psql

**Now in psql shell** :

* sql\_demo=# create table movies(movie\_id int, movie\_name varchar(40), movie\_gener varchar(30), imdb\_rating real); **---create table---**

**Response :**

CREATE TABLE  
  
You can cross-verify in pg-admin , refresh database 🡪 select schemas 🡪 tables.

**Table creation in pg-admin :**

* select schemas 🡪tables 🡪 right click 🡪 select tables 🡪 in general just table name rest default 🡪 columns 🡪 plus sign add column name , data type, to not to have any null values check ‘not null’, check primary key for unique values 🡪 save
* Query :

select \* from movies **---Display whole table---**

**To Drop/Delete any tables :**

Query :

drop table movies; **---delete the table---**

* sql\_demo=# \dt **---display all tables in the database---**
* sql\_demo=# insert into movies(movie\_id, movie\_name, movie\_gener, imdb\_rating)

values (101,'a', 'action', 4); **---intsert into tables---**

**Response :**

INSERT 0 1

sql\_demo=# select \* from movies;

movie\_id | movie\_name | movie\_gener | imdb\_rating

----------+------------+-------------+-------------

101 | a | action | 4

(1 row)

* sql\_demo=# \d movies **---describe the table---**

Table "public.movies"

Column | Type | Collation | Nullable | Default

-------------+-----------------------+-----------+----------+---------

movie\_id | integer | | |

movie\_name | character varying(40) | | |

movie\_gener | character varying(30) | | |

imdb\_rating | real | | |

* sql\_demo=# update movies set movie\_gener = 'drama, kdaram' where movie\_id =

104; **---update the existing record---**

**Response :**

UPDATE 1

sql\_demo=# select \* from movies;

movie\_id | movie\_name | movie\_gener | imdb\_rating

----------+------------+---------------+-------------

101 | a | action | 4

102 | b | boxing | 3

103 | c | comedy | 3

104 | d | drama, kdaram | 5

(4 rows)

* sql\_demo=# delete from movies where movie\_id = 105; **--- delete a record---**

**Response :**

DELETE 1

* sql\_demo=# select \* from movies where imdb\_rating > 3; **---sorting using where clause---**

**Response :**

movie\_id | movie\_name | movie\_gener | imdb\_rating

----------+------------+---------------+-------------

101 | a | action | 4

104 | d | drama, kdaram | 5

(2 rows)

* sql\_demo=# select \* from movies where imdb\_rating between 3 and 4; **---between operator---**

**Response :**

movie\_id | movie\_name | movie\_gener | imdb\_rating

----------+------------+-------------+-------------

101 | a | action | 4

102 | b | boxing | 3

103 | c | comedy | 3(3 rows)

**Note : \* represent all the column in the table**

* sql\_demo=# select movie\_name from movies; **---using column names---**

**Response :**

movie\_name

------------

a

b

c

d

(4 rows)

* sql\_demo=# select movie\_name, movie\_gener from movies where imdb\_rating < 4;

**---using column names---**

**Response :**

movie\_name | movie\_gener

------------+-------------

b | boxing

c | comedy

(2 rows)

* **IN Clause works like a or operator**
* sql\_demo=# select \* from movies where imdb\_rating in (4, 3); **---in operator---**

**Response :**

movie\_id | movie\_name | movie\_gener | imdb\_rating

----------+------------+-------------+-------------

101 | a | action | 4

102 | b | boxing | 3

103 | c | comedy | 3(3 rows)

**Now Importing csv file into pgadmin :**

* sql\_demo=# create table employee(employee\_id int not null primary key, employee\_name varchar(40), employee\_email varchar(40), employee\_salary int, employee\_department int);

**---first create table with the same column name---**

**Response :**

CREATE TABLE

* sql\_demo=# \copy employee(employee\_id, employee\_name, employee\_email, employee\_salary, employee\_department) from 'C:\Users\Bharath Kumar S\Downloads\employees.csv' delimiter ',' csv header;

**---in pg-admin you can select table 🡪 import🡪path🡪csv🡪select headers🡪save. ---**

**---in psql the above command fine to copy csv---**

**Response :**

COPY 50

* sql\_demo=# select distinct employee\_department from employee; **---to display only unique---**

**Response :**

employee\_department

---------------------

70

60

40

30

90

10

50

100

20

110

(10 rows)

* sql\_demo=# select \* from employee where employee\_email is null; **---is null---**

**Response :**

employee\_id | employee\_name | employee\_email | employee\_salary | employee\_department

-------------+---------------+----------------+-----------------+---------------------

(0 rows)

* sql\_demo=# select \* from employee order by employee\_salary; **---order by by default ascending---**

**Response :**

employee\_id | employee\_name | employee\_email | employee\_salary | employee\_department

-------------+---------------+----------------+-----------------+---------------------

132 | TJ | TJOLSON | 2100 | 50

128 | Steven | SMARKLE | 2200 | 50

* sql\_demo=# select \* from employee order by employee\_salary desc; **---order by desc---**

**Response :**

employee\_id | employee\_name | employee\_email | employee\_salary |employee\_department

-------------+---------------+----------------+-----------------+---------------------

100 | Steven | SKING | 24000 | 90

102 | Lex | LDEHAAN | 17000 | 90

* sql\_demo=# alter table employee rename column employee\_first\_name to employee\_name;

**---Rename column---**

**Response:**

ALTER TABLE

* sql\_demo=# select \* from employee where employee\_department = 50 and employee\_salary < 11000; **--- and operator---**

**Response:**

employee\_id | employee\_name | employee\_email | employee\_salary | employee\_department

-------------+---------------+----------------+-----------------+---------------------

198 | Donald | DOCONNEL | 2600 | 50

199 | Douglas | DGRANT | 2600 | 50

120 | Matthew | MWEISS | 8000 | 50

**Note:  
and operator = if both the condition is satisfied then it return value,**

**or operator = from both the condition if any 1 satisfy the condition it return the value**

* sql\_demo=# select \* from employee where employee\_salary = 2600 or employee\_department = 10; **---or operator---**

**Response:**

employee\_id | employee\_name | employee\_email | employee\_salary |employee\_department

-------------+---------------+----------------+-----------------+---------------------

198 | Donald | DOCONNEL | 2600 | 50

199 | Douglas | DGRANT | 2600 | 50

200 | Jennifer | JWHALEN | 4400 | 10

118 | Guy | GHIMURO | 2600 | 30

(4 rows)

* sql\_demo=# select \* from employee order by employee\_salary desc limit 5;

**---limit clause which prints top specified values---**

**Respone:**

employee\_id | employee\_name | employee\_email | employee\_salary | employee\_department

-------------+---------------+----------------+-----------------+---------------------

100 | Steven | SKING | 24000 | 90

101 | Neena | NKOCHHAR | 17000 | 90

102 | Lex | LDEHAAN | 17000 | 90

201 | Michael | MHARTSTE | 13000 | 20

205 | Shelley | SHIGGINS | 12008 | 110

(5 rows)

* sql\_demo=# select \* from employee order by employee\_salary desc limit 5 offset 3;

**---limit clause prints top specified values and offset skips top specified rows---**

**Response:**

employee\_id | employee\_name | employee\_email | employee\_salary | employee\_department

-------------+---------------+----------------+-----------------+---------------------

201 | Michael | MHARTSTE | 13000 | 20

205 | Shelley | SHIGGINS | 12008 | 110

108 | Nancy | NGREENBE | 12008 | 100

114 | Den | DRAPHEAL | 11000 | 30

204 | Hermann | HBAER | 10000 | 70

(5 rows)

* sql\_demo=# select \* from employee order by employee\_salary desc fetch first 3 rows only;

**---fetch first 3 rows only---**

**Response:**

employee\_id | employee\_name | employee\_email | employee\_salary | employee\_department

-------------+---------------+----------------+-----------------+---------------------

100 | Steven | SKING | 24000 | 90

101 | Neena | NKOCHHAR | 17000 | 90

102 | Lex | LDEHAAN | 17000 | 90

(3 rows)

* sql\_demo=# select \* from employee order by employee\_salary desc offset 3 fetch first 3 rows only; **---works same as of limit and offset---**

**Response:**

employee\_id | employee\_name | employee\_email | employee\_salary | employee\_department

-------------+---------------+----------------+-----------------+---------------------

201 | Michael | MHARTSTE | 13000 | 20

205 | Shelley | SHIGGINS | 12008 | 110

108 | Nancy | NGREENBE | 12008 | 100

(3 rows)

**Note: Like Operator 🡪 it is a pattern matching.**

* sql\_demo=# select employee\_name, employee\_email from employee where employee\_name like 'B%'; **---name pattern matching where starting letter must be B rest any letters---**

**Response:**

employee\_name | employee\_email

---------------+----------------

Bruce | BERNST

(1 row)

* sql\_demo=# select employee\_name, employee\_email from employee where employee\_name like '%y'; **---name pattern matching where starting letter must be any but ending letter is y---**

**Response:**

employee\_name | employee\_email

---------------+----------------

Shelley | SHIGGINS

Nancy | NGREENBE

Guy | GHIMURO

(3 rows)

* sql\_demo=# select employee\_name, employee\_email from employee where employee\_name like '%ll%'; **---beginning and end can be any but in middle there must be specified string---**

**Response:**

employee\_name | employee\_email

---------------+----------------

Shelley | SHIGGINS

William | WGIETZ

Valli | VPATABAL

Shelli | SBAIDA

(4 rows)

* sql\_demo=# select employee\_name, employee\_email from employee where employee\_name like '\_h%';**--- apart from starting first character second character should match and rest any character---**

**Response:**

employee\_name | employee\_email

---------------+----------------

Shelley | SHIGGINS

Shelli | SBAIDA

Shanta | SVOLLMAN

(3 rows)

* sql\_demo=# select sum(employee\_salary) from employee; **---to use sql functions---**

**Response:**

sum

--------

309116

(1 row)

* sql\_demo=# select sum(employee\_salary) as total\_salary from employee;

**---change the column name with the alias name---**

**Response:**

total\_salary

--------------

309116

(1 row)

* sql\_demo=# select avg(employee\_salary) as total\_avg\_salary from employee;

**---avg sql function---**

**Response:**

total\_avg\_salary

-----------------------

6182.3200000000000000

(1 row)

* sql\_demo=# select max(employee\_department) as max\_department from employee;

**---max sql function---**

**Response:**

max\_department

----------------

110

(1 row)

* sql\_demo=# select min(employee\_department) as min\_department from employee;

**---min sql function---**

**Response:**

min\_department

----------------

10

(1 row)

* sql\_demo=# select count(distinct employee\_department) as total\_department from employee; **---count sql function---**

**Response:**

total\_department

------------------

10

(1 row)

* update employee set employee\_department = ‘99’ where employee\_department is null;

**--- update null values---**

* sql\_demo=# select employee\_department, avg(employee\_salary) as grp\_avg\_salary from employee group by employee\_department; **---group by sql function---**

**Response:**

employee\_department | grp\_avg\_salary

---------------------+------------------------

70 | 10000.0000000000000000

60 | 5760.0000000000000000

40 | 6500.0000000000000000

30 | 4150.0000000000000000

90 | 19333.333333333333

10 | 4400.0000000000000000

50 | 3721.7391304347826087

100 | 8601.3333333333333333

20 | 9500.0000000000000000

(10 rows)

* sql\_demo=# select employee\_department, avg(employee\_salary) as avg\_emp\_salary from employee group by employee\_department order by avg\_emp\_salary desc;

**---group by and order by sql function---**

**Response:**

employee\_department | avg\_emp\_salary

---------------------+------------------------

90 | 19333.333333333333

110 | 10154.0000000000000000

70 | 10000.0000000000000000

20 | 9500.0000000000000000

100 | 8601.3333333333333333

40 | 6500.0000000000000000

60 | 5760.0000000000000000

10 | 4400.0000000000000000

30 | 4150.0000000000000000

50 | 3721.7391304347826087

(10 rows)

* sql\_demo=# select employee\_name, avg(employee\_salary) as avg\_sal from employee group by employee\_name having avg(employee\_salary)>5000; **---having clause---**

**Response:**

employee\_name | avg\_sal

---------------+------------------------

Pat | 6000.0000000000000000

Nancy | 12008.0000000000000000

Susan | 6500.0000000000000000

* sql\_demo=# select employee\_name, employee\_salary, case when employee\_salary >15000 and employee\_salary < 18000 then 'medium salary' when employee\_salary >18000 then 'high\_salary' when employee\_salary <15000 then 'low\_salary' end as emp\_sal from employee order by employee\_name; **--- case if else conditions in sql---**

**Response:**

employee\_name | employee\_salary | emp\_sal

---------------+-----------------+---------------

Adam | 8200 | low\_salary

Alexander | 9000 | low\_salary

* sql\_demo=# select \* from employee where employee\_salary > (select avg(employee\_salary) from employee); **---sub query---**

**Response:**

employee\_id | employee\_name | employee\_email | employee\_salary | employee\_department

-------------+---------------+----------------+-----------------+---------------------

201 | Michael | MHARTSTE | 13000 | 20

* sql\_demo=# select abs(-1); **---abs sql function---**

**Response:**

abs

-----

1

(1 row)

* sql\_demo=# select greatest(1, 5, 9); **---greatest sql function---**

**Response:**

greatest

----------

9

(1 row)

* sql\_demo=# select least(0.0, -1, 6); **---least sql function---**

**Response:**

least

-------

-1

(1 row)

* sql\_demo=# select mod(5, 10); **---mod sql function---**

**Response:**

mod

-----

5

(1 row)

* sql\_demo=# select power(2, 3); **---2 power 3 , power sql function---**

**Response:**

power

-------

8

(1 row)

* sql\_demo=# select char\_length('welcome to sql\_demo');**---char\_length sql function---**

**Response:**

char\_length

-------------

19

(1 row)

* sql\_demo=# select concat('Postgres', 'is', 'intresting'); **---concat sql function---**

**Response:**

concat

----------------------

Postgresisintresting

(1 row)

* sql\_demo=# select repeat('top\_g', 8); **---repeat sql function---**

**Response:**

repeat

------------------------------------------

top\_gtop\_gtop\_gtop\_gtop\_gtop\_gtop\_gtop\_g

(1 row)