**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)