Postgresql:

Links for installation:

1. Installation for windows:
2. Installation for ubuntu:

What is PostgreSQL?

PostgreSQL (pronounced as **post-gress-Q-L**) is an open-source relational database management system (DBMS) developed by a worldwide team of volunteers. PostgreSQL is not controlled by any corporation or other private entity and the source code is available free of charge.

PostgreSQL supports a large part of the SQL standard and offers many modern features including the following −

- Complex SQL queries
- SQL Sub-selects
- Foreign keys
- Trigger
- Views
- Transactions

Now that you have Postgres installed, open the psql as −

Program Files → PostgreSQL 13.0 → SQL Shell(psql).

What is psql

Psql is the interactive terminal for working with Postgres. Theres an abundance of flags available for use when working with psql, but lets focus on some of the most important ones, then how to connect:

- -h the host to connect to
- -U the user to connect with
- -p the port to connect to (default is 5432)

```
psql -h localhost -U username databasename
```

The other option is to use a full string and let psql parse it:

```
psql "dbname=dbhere host=hosthere user=userhere password=pwhere
port=5432 sslmode=require"
```

Once you've connected you can begin querying immediately. In addition to basic queries you can also use certain commands. Running \? will give you a list of all available commands, though a few key ones are called out below.

For example, to connect to dvdrental database under postgres user, you use the following command:

```
C:\Program Files\PostgreSQL\9.5\bin>psql -d dvdrental -U
postgres -W
Password for user postgres:
dvdrental=#
```

If you want to connect to a database that resides on another host, you add the -h option as follows:

```
psql -h host -d database -U user -W
```

In case you want to use SSL mode for the connection, just specify it as shown in the following command:

```
psql -U user -h host "dbname=db sslmode=require"
```

Commonly used commands

Turn query timing on

By default the timing of query results will not be available, but we can turn it on by using the following command.

```
# \timing

Timing is on.
```

This will show query timing in milliseconds.

List tables in database

```
# \d

      List of relations

 Schema |    Name     | Type  | Owner

--------+-------------+-------+-------
```

```
 public | employees | table | craig

(1 row)
```

Describe a table

```
# \d employees

           Table "public.employees"

  Column   |         Type          | Modifiers

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

 id        | integer               |

 last_name | character varying(50) |

 salary    | integer               |

Indexes:

    "idx_emps" btree (salary)
```

List all tables in database along with some additional information

```
# \d+

       List of relations

Schema | Name  | Type  |   Owner   |  Size  | Description

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

public | users | table | jarvis    | 401 MB |

(1 row)
```

Describe a table with additional information

```
#\d+ users

          Table "public.users"

  Column   |  Type   |   Modifiers   | Storage  | Stats target |
Description

-----------+---------+---------------+----------+--------------+-
------------
```

```
userid     | bigint  | not null      | plain    |              |
fullname   | text    | not null      | extended |              |
email      | text    | not null      | extended |              |
phone      | text    | not null      | extended |              |
credits    | money   | default 0.0   | plain    |              |
parked     | boolean | default false | plain    |              |
terminated | boolean | default false | plain    |              |


Indexes:
    "users_pkey" PRIMARY KEY, btree (userid)
```

List all databases

```
# \l
          List of databases
Name     |   Owner   | Encoding | Collate | Ctype |      Access
privileges
---------+-----------+----------+---------+-------+--------------
---------------
learning | jarvis    | UTF8     | C       | UTF-8 |
```

List all databases with additional information

```
# \l+
          List of databases
  Name    |   Owner   | Encoding | Collate | Ctype |      Access
privileges      |  Size   | Tablespace |
Description
----------+-----------+----------+---------+-------+-------------
----------------+---------+------------+------------------------
------------------
learning  | jarvis    | UTF8     | C       | UTF-8 |
| 492 MB  | pg_default |
```

List all schemas

```
# \dn
List of schemas
 Name  | Owner
-------+--------
public | jarvis
(1 row)
```

List all schemas with additional information

```
# \dn+
List of schemas
Name   | Owner  | Access privileges |      Description
-------+--------+-------------------+-----------------------
public | jarvis | jarvis=UC/jarvis +| standard public schema
       |        | =UC/jarvis        |
(1 row)
```

List all functions

```
#\df
```

List all functions with additional information

```
#\df+
```

Connect to another database

```
#\c dbname
```

Quit from postgres shell

```
#\q
```

Text editor inside psql

```
#\e
```

This opens your default text editor inside psql shell

List available views

To list available views in the current database, you use the `\dv` command.

```
\dv
```

List users and their roles

To list all users and their assign roles, you use `\du` command:

```
\du
```

Command history

To display command history, you use the `\s` command.

```
\s
```

If you want to save the command history to a file, you need to specify the file name followed the `\s` command as follows:

```
\s filename
```

Execute psql commands from a file

In case you want to execute psql commands from a file, you use `\i` command as follows:

```
\i filename
```

Turn on query execution time

To turn on query execution time, you use the `\timing` command.

```
dvdrental=# \timing
Timing is on.
```

```
dvdrental=# select count(*) from film;
 count
-------
  1000
(1 row)

Time: 1.495 ms
dvdrental=#
```

You use the same command \timing to turn it off.

```
dvdrental=# \timing
Timing is off.
dvdrental=#
```

SQL Commands for creation and insertion:

The CREATE TABLE statement is used to create a new table in a database.

Syntax

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
   ....
);
```

The column parameters specify the names of the columns of the table.

The datatype parameter specifies the type of data the column can hold (e.g. varchar, integer, date, etc.).

SQL CREATE TABLE Example

The following example creates a table called "Persons" that contains five columns: PersonID, LastName, FirstName, Address, and City:

Example

```
CREATE TABLE Persons (
    PersonID int,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
```

```
    City varchar(255)
);
```

```
CREATE TABLE CUSTOMERS(
   ID   INT              NOT NULL,
   NAME VARCHAR (20)     NOT NULL,
   AGE  INT              NOT NULL,
   ADDRESS  CHAR (25) ,
   SALARY   DECIMAL (18, 2),
   PRIMARY KEY (ID)
);
```

The SQL **CREATE DATABASE** statement is used to create a new SQL database.

Syntax

The basic syntax of this CREATE DATABASE statement is as follows −

```
CREATE DATABASE DatabaseName;
```

Always the database name should be unique within the RDBMS.

Example

If you want to create a new database <testDB>, then the CREATE DATABASE statement would be as shown below −

```
SQL> CREATE DATABASE testDB;
```

INSERT STATEMENT:

The SQL **INSERT INTO** Statement is used to add new rows of data to a table in the database.

Syntax

There are two basic syntaxes of the INSERT INTO statement which are shown below.

```
INSERT INTO TABLE_NAME (column1, column2, column3,...columnN)
VALUES (value1, value2, value3,...valueN);
```

Here, column1, column2, column3,...columnN are the names of the columns in the table into which you want to insert the data.

You may not need to specify the column(s) name in the SQL query if you are adding values for all the columns of the table. But make sure the order of the values is in the same order as the columns in the table.

The **SQL INSERT INTO** syntax will be as follows −

```
INSERT INTO TABLE_NAME VALUES (value1,value2,value3,...valueN);
```

Example

The following statements would create six records in the CUSTOMERS table.

```
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (1, 'Ramesh', 32, 'Ahmedabad', 2000.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (2, 'Khilan', 25, 'Delhi', 1500.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (3, 'kaushik', 23, 'Kota', 2000.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (4, 'Chaitali', 25, 'Mumbai', 6500.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (5, 'Hardik', 27, 'Bhopal', 8500.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (6, 'Komal', 22, 'MP', 4500.00 );
```

You can create a record in the CUSTOMERS table by using the second syntax as shown below.

```
INSERT INTO CUSTOMERS
VALUES (7, 'Muffy', 24, 'Indore', 10000.00 );
```

All the above statements would produce the following records in the CUSTOMERS table as shown below.

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```