## **PostgreSQL Report**

**What is POSTGIS?**

PostGIS  **is an open source software program that adds support for geographic objects to the PostgreSQL object-relational database**

## **What is PostgreSQL?**

PostgreSQL is a powerful, open source object-relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads. The origins of PostgreSQL date back to 1986 as part of the **[POSTGRES](https://www.postgresql.org/docs/current/history.html)** project at the University of California at Berkeley and has more than 35 years of active development on the core platform.

## **Why use PostgreSQL?**

PostgreSQL comes with **[many features](https://www.postgresql.org/about/featurematrix/)** aimed to help developers build applications, administrators to protect data integrity and build fault-tolerant environments, and help you manage your data no matter how big or small the dataset. In addition to being **[free and open source](https://www.postgresql.org/about/license/)**, PostgreSQL is highly extensible. For example, you can define your own data types, build out custom functions, even write code from **[different programming languages](https://www.postgresql.org/docs/current/xplang.html)** without recompiling your database!

Below is an inexhaustive list of various features found in PostgreSQL, with more being added in every **[major release](https://www.postgresql.org/developer/roadmap/)**:

* ****Data Types****
  + Primitives: Integer, Numeric, String, Boolean
  + Structured: Date/Time, Array, Range / Multirange, UUID
  + Document: JSON/JSONB, XML, Key-value (Hstore)
  + Geometry: Point, Line, Circle, Polygon
  + Customizations: Composite, Custom Types
* ****Data Integrity****
  + UNIQUE, NOT NULL
  + Primary Keys
  + Foreign Keys
  + Exclusion Constraints
  + Explicit Locks, Advisory Locks
* ****Concurrency, Performance****
  + Indexing: B-tree, Multicolumn, Expressions, Partial
  + Advanced Indexing: GiST, SP-Gist, KNN Gist, GIN, BRIN, Covering indexes, Bloom filters
  + Sophisticated query planner / optimizer, index-only scans, multicolumn statistics
  + Transactions, Nested Transactions (via savepoints)
  + Multi-Version concurrency Control (MVCC)
  + Parallelization of read queries and building B-tree indexes
  + Table partitioning
  + All transaction isolation levels defined in the SQL standard, including Serializable
  + Just-in-time (JIT) compilation of expressions
* ****Reliability, Disaster Recovery****
  + Write-ahead Logging (WAL)
  + Replication: Asynchronous, Synchronous, Logical
  + Point-in-time-recovery (PITR), active standbys
  + Tablespaces
* ****Security****
  + Authentication: GSSAPI, SSPI, LDAP, SCRAM-SHA-256, Certificate, and more
  + Robust access-control system
  + Column and row-level security
  + Multi-factor authentication with certificates and an additional method
* ****Extensibility****
  + Stored functions and procedures
  + Procedural Languages: PL/pgSQL, Perl, Python, and Tcl. There are other languages available through extensions, e.g. Java, JavaScript (V8), R, Lua, and Rust
  + SQL/JSON path expressions
  + Foreign data wrappers: connect to other databases or streams with a standard SQL interface
  + Customizable storage interface for tables
  + Many extensions that provide additional functionality, including PostGIS
* ****Internationalisation, Text Search****
  + Support for international character sets, e.g. through ICU collations
  + Case-insensitive and accent-insensitive collations
  + Full-text search

There are many more features that you can discover in the PostgreSQL **[documentation](https://www.postgresql.org/docs/)**. Additionally, PostgreSQL is highly extensible: many features, such as indexes, have defined APIs so that you can build out with PostgreSQL to solve your challenges.

PostgreSQL has been proven to be highly scalable both in the sheer quantity of data it can manage and in the number of concurrent users it can accommodate. There are active PostgreSQL clusters in production environments that manage many terabytes of data, and specialized systems that manage petabytes.

Download PostgreSQL :

<https://www.postgresql.org/download/windows/>

Download PgAdmin 4:

<https://www.postgresql.org/ftp/pgadmin/pgadmin4/v7.0/windows/>

## PostgreSQL Features

It is one of the most popular databases supporting JSON (non-relational) queries and SQL for (relational) queries. PostgreSQL is an object-relational database management system (ORDBMS). It contains the various advanced data types and robust feature sets, which increase the extensibility, reliability, and data integrity of the software.

In this section, we are going to discuss the most advanced features, Database administration tools of PostgreSQL, and benefits and drawbacks, which will help us to enhance our knowledge of PostgreSQL.

[PostgreSQL](https://www.javatpoint.com/postgresql-tutorial) includes multiple features that are designed to help the developers in developing the applications, manage our data in the datasets, and managers can keep the data integrity, and create the Risk-tolerant environments. Whenever the new release comes into the market, the PostgreSQL Global group of developers will enhance the previous features and adds some new features based on the user's demand.

The essential features of PostgreSQL are as follows:

* ****Free to download:**** It is open-source, and we can easily download it from the official website of PostgreSQL.
* ****Compatible on several operation systems:**** PostgreSQL runs on all major operating systems such as Microsoft Windows, Linux, MacOS X, UNIX (AIX, BSD, HP-UX, SGI IRIX, Solaris, and Tru64), etc.
* ****Compatible with various programming languages:**** It supports multiple programming interfaces such asC/C++, JAVA, Python, Perl, Ruby, Tcl, and ODBC (Open Database Connectivity).

### pgAdmin

It is one of the most famous and open-source management and development platforms for PostgreSQL. PgAdmin is the most advanced database around the world. It can be used on various operating systems such as UNIX, [Linux](https://www.javatpoint.com/linux-tutorial), Mac OS X, and Windows to achieve PostgreSQL 9.2 and above versions.

## Advantages and disadvantages of PostgreSQL

The ****benefits**** of the PostgreSQL are as follows:

* PostgreSQL is easy to use; that why we do not require much training.
* It requires low maintenance management for enterprise as well as embedded usage.
* PostgreSQL manages data in a relational database as it is very powerful and robust.
* We can quickly get the source code of PostgreSQL as it is freely available in an open-source license, and we can immediately implement, change according to our requirements.
* It can execute dynamic web-application and website as the LAMP stack option.
* PostgreSQL is a highly risk-tolerant database.

The ****drawbacks**** of PostgreSQL are as follows:

* PostgreSQL does not support the various open-source applications as compared to MySQL.
* In this, creating replication is a bit complex.
* It is not maintained by one company.
* PostgreSQL speed performance is not as good as compare to further tools.
* It is a bit slow as compared to MySQL.
* Sometimes, the installation process is not easy for the learner.

# Connect to a PostgreSQL Database Server

In this section, we are going to understand how to connect the ****PostgreSQL database server through**** an interactive terminal software called ****pgAdmin**** and the ****SQL Shell(psql)**** application.

If we installed the PostgreSQL database server, then the PostgreSQL installer will also connect with some beneficial tools to work through the PostgreSQL database server.

Here, we are going to discuss how we can connect the PostgreSQL database server through the below tools:

* ****PgAdmin****
* ****SQL Shell (PSQL)****

****PgAdmin:**** It is a web-based front-end management tool for [PostgreSQL](https://www.javatpoint.com/postgresql-tutorial), which helps us connect the ****PostgreSQL database server.****

****SQL shell(psql):**** It is a terminal-based front-end application, and we can also use this application to connect the ****PostgreSQL database server****.

We will now connecting the PostgreSQL database server with the help of ****pgAdmin****.

* ****Connect to PostgreSQL database server using pgAdmin****

Firstly, we will understand how to connect to a database with the help of ****the pgAdmin**** application.

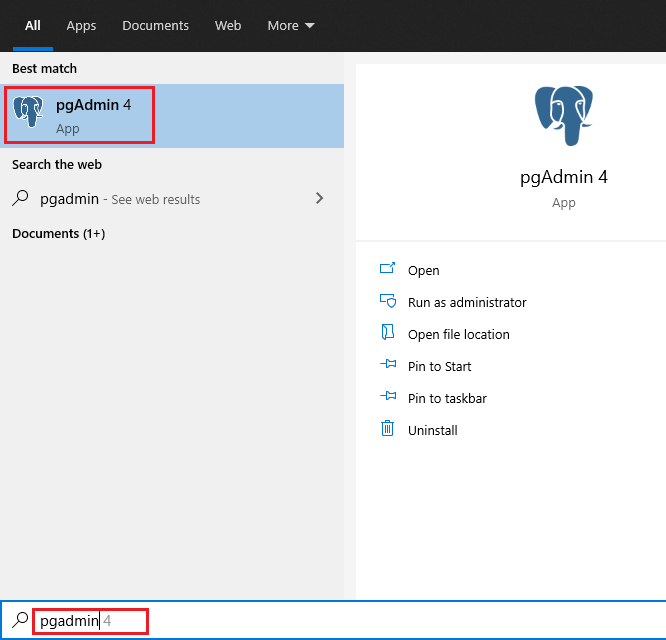
The pgAdmin tool permits us to relate with the PostgreSQL database server through an ****in-built user interface****. And here, we are using the ****pgadmin4 version**** to connected the database.

We will follow the below steps to connect the PostgreSQL database server using the ****pgAdmin GUI tool:****

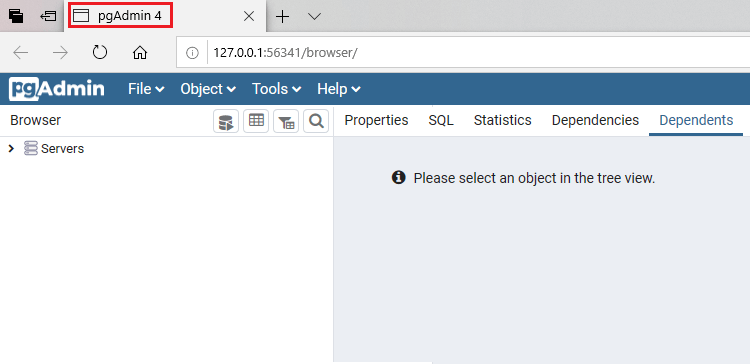
****Step1: Launch the pgAdmin application****

Firstly, we will open the ****pgAdmin**** application. For this, we will enter ****pgAdmin**** into the search bar of our system, as shown in the below screenshot:

of our system, as shown in the below screenshot:

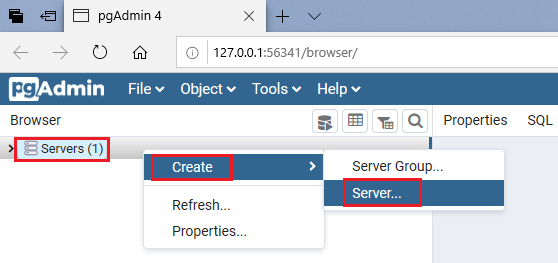


After clicking on the ****pgAdmin**** application, it will launch on the web browser, as we can see in the following screenshot:



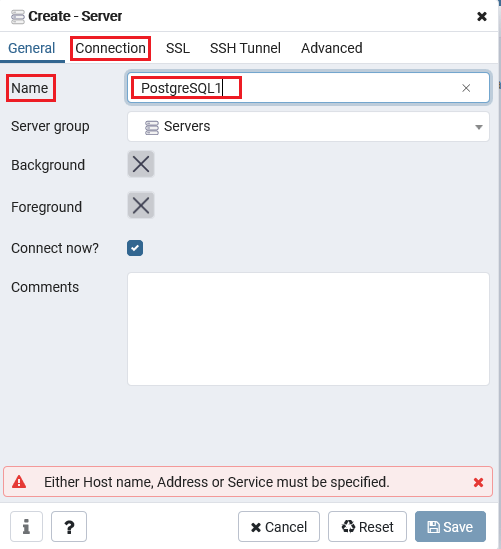
****Step2: Create a server****

After that, we will right-click on the ****Servers**** node and select the ****Create → Server…**** menu to ****create a server****, as we can see in the below image:



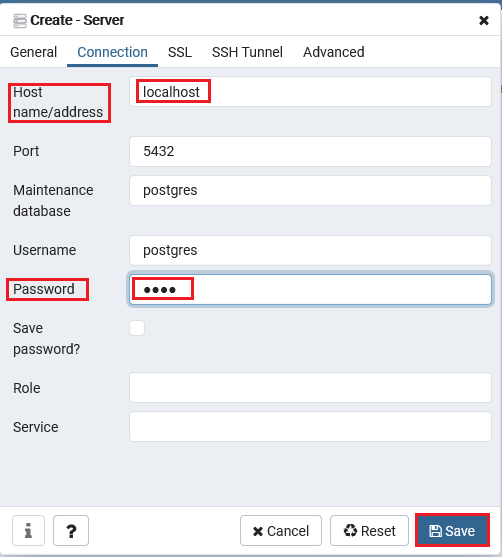
****Step3: Provide the server name****

After selecting the ****Server**** option, the ****Create-Server**** window will be opened, where we enter the ****server name**** in the ****Name column, for example, PostgreSQL1**** and then we will click on the ****Connection**** tab, as shown below:



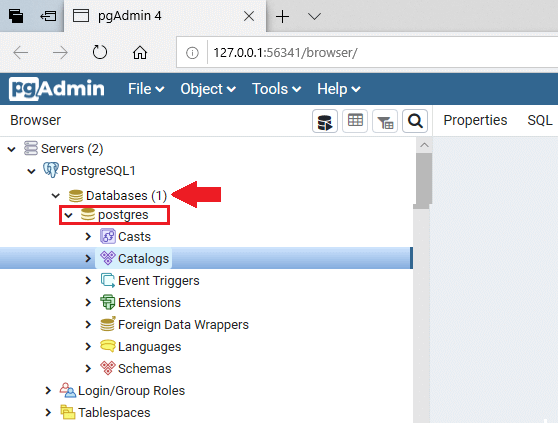
****Step4: Provide the host and password****

After clicking on the ****Connection tab****, we will provide the details of the ****host**** and ****password**** for the ****Postgres**** user, and after that, we will click on the ****Save**** button.



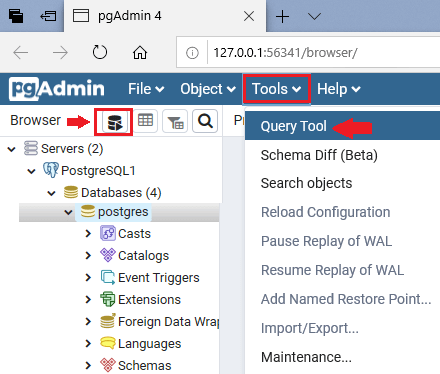
****Step5: Expanding the server****

In the next step, we will click on the ****Servers node**** to ****expand the server****. And PostgreSQL has a database named ****Postgres**** by default, as we can see in the following image:



****Step6: Open the Query tool****

Now, we will open the ****query tool**** by selecting the menu item ****Tool → Query Tool,**** or we can directly click the ****query tool**** icon nearby ****Browser,**** as we can see in the following screenshot:

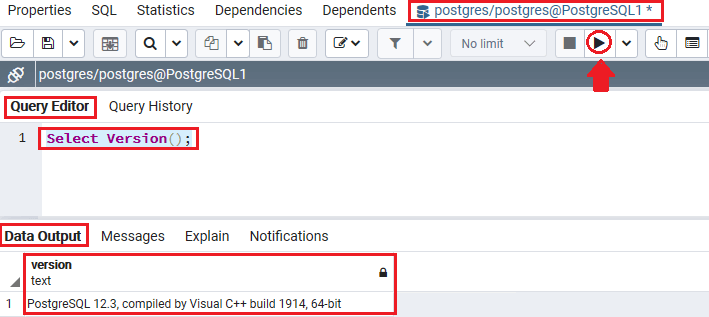


****Step7: Enter the command in the Query editor****

Once the ****query tool**** is opened successfully, we will enter the below command in the ****Query Editor**** and click on the ****Execute**** button.

1. **Select** Version();

And after implementing the above command, we will get the ****Data output**** of the Specified command, which display the complete information of the currently used ****version**** of ****PostgreSQL****, as shown in the below image:



After understanding the ****PostgreSQL database server's connection through pgAdmin4****, we will learn the connection process of the ****PostgreSQL database server using SQL Shell(psql).****

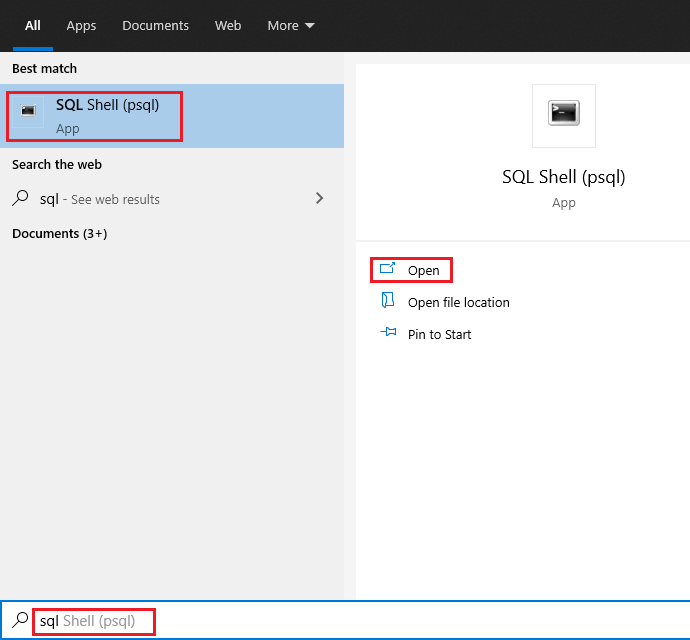
* ****Connect to PostgreSQL database server using psql****

The ****psql**** is an ****interactive shell tool**** provided by PostgreSQL. The psql permits us to interrelate with the PostgreSQL database server, ****for example, working on the database objects**** and ****implementing the**[SQL](https://www.javatpoint.com/sql-tutorial)**commands****.

To connect to the PostgreSQL database server by using the **[SQL shell(psql)](https://www.javatpoint.com/psql-commands)** application, we will follow the below steps:

****Step1: Open the psql****

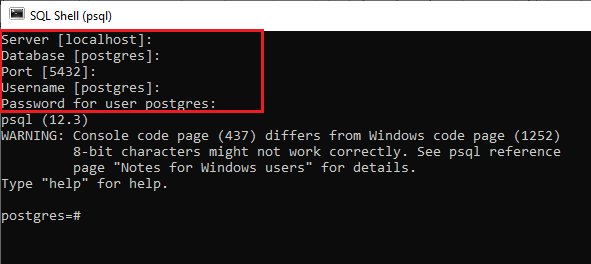
Firstly, we will open the ****psql**** in our local system. For this, we will go to the ****home**** button and search for ****SQL shell(psql)**** and click on the ****Open**** button, as shown in the below screenshot:



****Step2: Connect the Database****

Once the ****SQL shell**** is opened, we will press the enter key ****four to five times.****

And then provide the ****password**** for the user (which we created earlier) to connect the database server using the ****Postgres**** user who is a ****by default**** user in PostgreSQL, as shown in the below screenshot:



#### Note: If we want to provide all the details like Server, Database, Port, Username, and Password manually, we will press Enter key and the application will use the default values, which are defined in the square bracket [] and move the arrow to the new line.

Suppose we have ****localhost**** as the default database server, and we were providing the ****Postgres user password****, we will enter a similar password for the Postgres user-specified throughout the [PostgreSQL installation](https://www.javatpoint.com/install-postgresql).

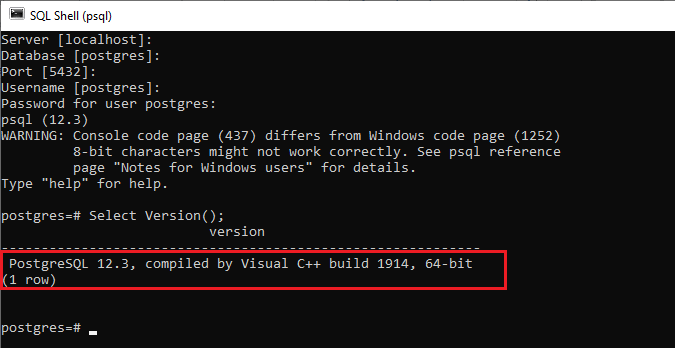
****Step3: Entering a command to relate with the PostgreSQL Database Server****

In this step, we will write one SQL command, which helps us relate with the PostgreSQL Database Server.

In the below command, we are trying to get the complete details about the current version of PostgreSQL:

1. **SELECT** version();

After implementing or pressing the ****Enter key****, we will get the following output, which displays the full information about the existing version of PostgreSQL version of our system:



#### Note: While writing a command in psql, we should make sure that the specified command would end with a semicolon (;).

* ****Connecting the PostgreSQL database from other applications****

If any application supported by ODBC or [JDBC](https://www.javatpoint.com/java-jdbc) also be connected to the PostgreSQL database server.

Furthermore, when we develop an application that uses an explicit driver, the particular software can also connect to the PostgreSQL database server.

For example, we can connect the PostgreSQL from [JAVA](https://www.javatpoint.com/java-tutorial), connect the PostgreSQL from Python and connect to PostgreSQL from [PHP](https://www.javatpoint.com/php-tutorial).

25/05/2023

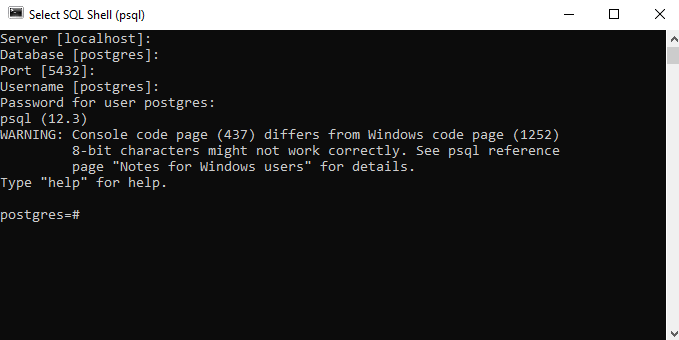
# Connect To a PostgreSQL Database Server:

how to ****connect to the PostgreSQL Database Server**** via an interactive terminal program called ****psql**** and via the ****pgAdmin**** application.

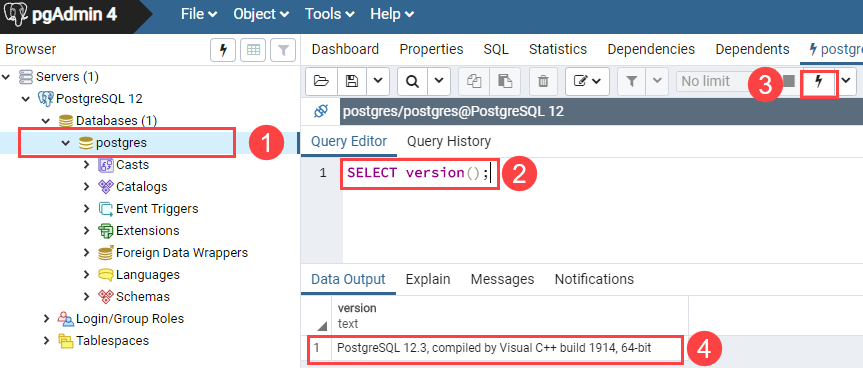
how to connect to the PostgreSQL database server via the following tools:

* psql – a terminal-based front-end to PostgreSQL database server.
* pgAdmin – a web-based front-end to PostgreSQL database server.

Psql-command tool



pgAdmin- user interface tools



PostgreSql Commands:

1. To create a new database:

Create database databse\_name;

1. use database:

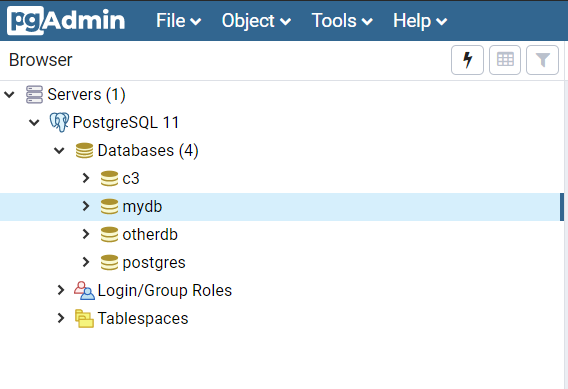
Now, type the following command to connect/select a desired database; here, we will connect to the *testdb*database.

Psql : postgres=# \c testdb;

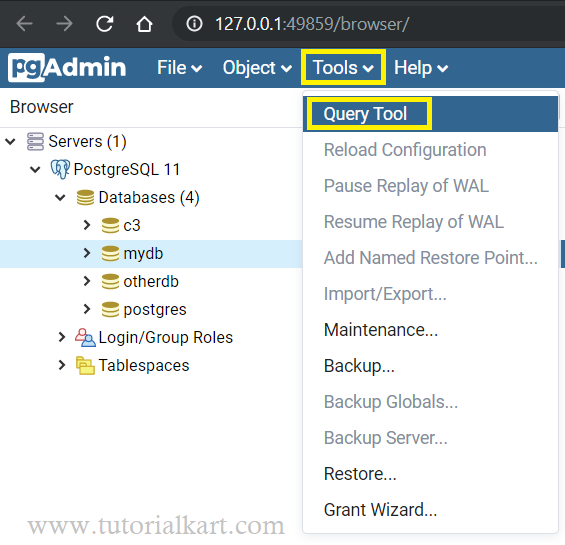
### **Select Database using pgAdmin**

You can also select the database and open SQL Query window in pgAdmin UI.

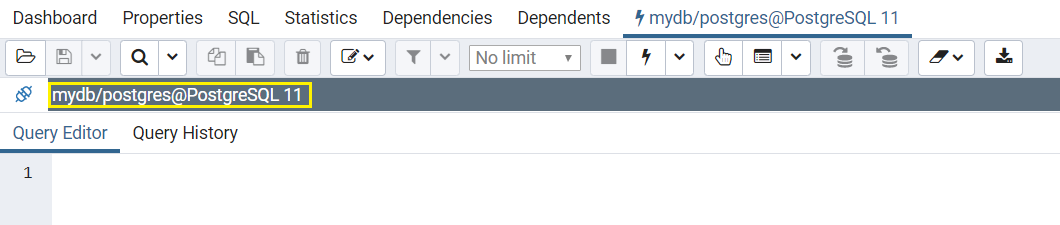
****Step 1****: Select the database, by clicking on it.



Now, click on the Tools drop down menu and click on Query Tool.



Now, a new window appears with a connection make to the database you selected. You can run SQL queries here.



### Using DROP DATABASE

This command drops a database. It removes the catalog entries for the database and deletes the directory containing the data. It can only be executed by the database owner. This command cannot be executed while you or anyone else is connected to the target database (connect to postgres or any other database to issue this command).

Command : Drop database database\_name;

**Create Table:**

The PostgreSQL CREATE TABLE statement is used to create a new table in any of the given database.

Command : create table table\_name(column1 datatype, column datatype2);

**Insert Command:**

The INSERT INTO statement of SQLis used to insert a new row/record in a table. There are two ways of using the SQL INSERT INTO statement for inserting rows.

Ex: INSERT INTO Student (ROLL\_NO, NAME, Age) VALUES ('5','PRATIK','19');

**Update Command:**

The UPDATE statement in [SQL](https://www.geeksforgeeks.org/sql-tutorial/)is used to update the data of an existing table in the database. We can update single columns as well as multiple columns using the UPDATE statement as per our requirement.

In a very simple way, we can say that SQL commands(UPDATE and DELETE) are used to change the data that is already in the database. The SQL DELETE command uses a WHERE clause.

*Syntax: UPDATE table\_name SET column1 = value1, column2 = value2,…*

*Ex:* UPDATE Customer SET CName = 'Nitin' WHERE Age = 22;

**Delete Command:**

Existing records in a table can be deleted using the SQL DELETE Statement. We can delete a single record or multiple records depending on the condition we specify in the WHERE clause.

**Syntax:**

*DELETE FROM table\_name WHERE some\_condition;*

EX: DELETE FROM GFG\_EMPLOyees WHERE NAME = 'Rithvik';

### DROP command:

DROP is used to delete a whole [database](https://www.geeksforgeeks.org/what-is-database/)or just a table.

In this article, we will be learning about the DROP statement which destroys objects like an existing database, table, index, or view. A DROP statement in SQL removes a component from a relational database management system (RDBMS).

**Syntax:** DROP TABLE table\_name;

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

26/05/2023

**Truncate Command:**

It preserves the structure of the table for further use exist but deletes all the data.Truncate is used to eliminate the tuples from the table.

Ex:*TRUNCATE TABLE  table\_name;*

# SQL – ORDER BY:

The ORDER BY statement in [SQL](https://www.geeksforgeeks.org/sql-tutorial/)is used to sort the fetched data in either ascending or descending according to one or more columns.

* By default ORDER BY sorts the data in ascending order.
* We can use the keyword DESC to sort the data in descending order and the keyword ASC to sort in ascending order.

**Syntax:**

*SELECT \* FROM table\_name ORDER BY column\_name ASC | DESC*

***PostgreSQL Like clause:***

The PostgreSQL **LIKE** operator is used to match text values against a pattern using wildcards. If the search expression can be matched to the pattern expression, the LIKE operator will return true, which is **1**.

There are two wildcards used in conjunction with the LIKE operator −

* The percent sign (%)
* The underscore (\_)

Ex: select name from cutomers where name like 'R%';

Output: "Raghav"

"Raja"

## PostgreSQL BETWEEN operator :

## PostgreSQL BETWEEN operator is used to match a value against a range of values The BETWEEN operator is used generally with WHERE clause with association with SELECT, INSERT, UPDATE or DELETE statement.

EX : select \* from cutomers where salary between '5000' and '8000';

# **PostgreSQL DISTINCT:**

we are going to understand the working of ****the PostgreSQL DISTINCT clause****, which is used to delete the matching rows or data from a table and get only the unique records.

EX: select distinct salary from cutomers;

## **PostgreSQL ALTER TABLE command:**

we are going to learn the various commands of **PostgreSQL ALTER TABLE** for changing the structure of a table.

1)Add Column:

Syntax: ALTER TABLE table\_name ADD COLUMN new\_column\_name TYPE;

Ex: alter table cutomers add column address varchar(60);

2)Rename Column:

We use the [ALTER TABLE](https://www.javatpoint.com/postgresql-alter-table) command with the ****Rename Column condition**** to ****rename a column of a table****.

****Syntax**** The syntax of the ****Alter table rename column**** command is given below:

**ALTER** **TABLE** table\_name

RENAME **COLUMN** column\_name **TO** new\_column\_name;

Example:

alter table cutomers rename column dept to department;

1. Drop Column:

We will use the DROP COLUMN condition in the [ALTER TABLE command](https://www.javatpoint.com/postgresql-alter-table) for dropping a column of a table:

Syntax: ALTER TABLE table\_name DROP COLUMN column\_name;

Example: alter table cutomers drop column address;

1. Change column type:

we are going to discuss how we can ****change the columns data type**** with the help of the ****ALTER TABLE**** command.

The basic syntax of changing the column datatype is as follows:

ALTER TABLE table\_name  ALTER COLUMN column\_name [SET DATA] TYPE new\_data\_type;

Example: alter table cutomers alter column id type varchar(90);

# **PostgreSQL LIMIT:**

we are going to understand the working of the ****PostgreSQL**** LIMIT clause, which is used to get a subset of rows produced by a command.

### Syntax of PostgreSQL LIMIT Clause

The basic syntax of the PostgreSQL LIMIT clause is as follows:

SELECT select\_list FROM table\_name ORDER BY sort\_expression LIMIT row\_count

Example: select name,salary from cutomers order by name ,salary limit 5;

**PostgreSQL Operators:**

1. **not equal to : !=**

select \* from cutomers where department!='IT';

1. **Equal to:=**

select \* from cutomers where department!='IT';

1. **< : less than,<=**

select \* from cutomers where salary < '4000';

select \* from cutomers where salary <= '4000';

1. **>: Greate than,>=**

select \* from cutomers where salary >'4000';

select \* from cutomers where salary >= '4000';

**PostgreSQL NOT operator example:**

If we want to display the list of employees with columns empno, emp\_first\_name, designame, salary and deptno from employee table who is not belonging into the deptno 25, the following SQL can be used.

EX1: SELECT empno,emp\_first\_name,designame,salary,deptno FROM employee WHERE NOT deptno=25;

EX2: select \* from cutomers where not department='IT' and not department='Marketing';

**PostgreSQL NOT, AND operator example**

If we want to display the list of employees with columns empno, emp\_first\_name, designame, salary and deptno from employee table who is not belonging into the deptno 25 and 15, the following SQL can be used.

Ex :SELECT eno,ename,salary,deptno FROM employee WHERE NOT deptno=25 AND NOT deptno=15;

# **PostgreSQL JOIN**

In this section, we are going to understand the working of several types of **PostgreSQL joins**, such as **Inner join, Left join, Right join, and Full Outer join** in brief.

PostgreSQL JOINS are used with [SELECT command](https://www.javatpoint.com/postgresql-select), which helps us to retrieve data from various tables. And we can merge the **Select and Joins** statements together into a single command. Whenever we want to get records from two or more tables, we will execute the joins commands.

It is used to merge columns from one or more tables according to the data of the standard columns between connected tables. Usually, the **standard columns** of the **first table** are **primary key columns** and the **second table** columns are **foreign key** columns.

In [PostgreSQL](https://www.javatpoint.com/postgresql-tutorial), we have various types of joins which are as follows:

* [Inner join](https://www.javatpoint.com/postgresql-inner-join)
* [Left join](https://www.javatpoint.com/postgresql-left-join)
* [Right join](https://www.javatpoint.com/postgresql-right-join)
* [Cross join](https://www.javatpoint.com/postgresql-cross-join)
* [Full outer join](https://www.javatpoint.com/postgresql-full-join)
* Natural join
* Self-join.