How to connect and use PostgreSQL in Python

If you are writing a Python program that needs to access data from a Postgres database, you will need to know how to connect the two. Once you establish a connection, you can use it to run queries and fetch or save data.

If you are writing a Python program that needs to access data from a Postgres database, you will need to know how to connect the two. Once you establish a connection, you can use it to run queries and fetch or save data.

Download and install PostgreSQL

PostgreSQL is a great choice for your programming projects. You can download and install the required version of PostgreSQL depending on your operating system. Postgres is available for download on standard operating systems such as Windows, macOS, and Linux Ubuntu.

The installation process will vary between operating systems, so you should follow the installation steps to ensure a smooth setup experience.

Install required libraries

You can use the psycopg2 library to connect to a PostgreSQL database from Python. Run this command in the Python interpreter to check if the library is installed:

```
import psycopg2
```

If you receive an error message (e.g. "No module named 'psycopg2'"), install the library with this command:

```
pip install psycopg2
```

PIP is a Python package manager that you can install on Windows, Mac, or Linux. It helps reduce the complexity of installing Python packages.

Fetch credentials using pgAdmin4

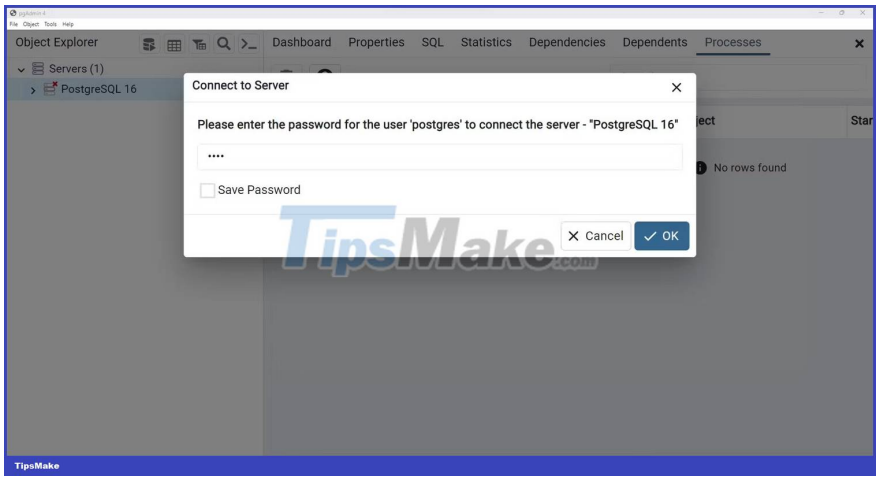
You can use the pgAdmin4 application to manage your Postgres database in a GUI environment. You may have installed it at the same time as you installed Postgres, but you can download pgAdmin4 and install it separately if needed.

Here's how you can use pgAdmin4 to get your server credentials:

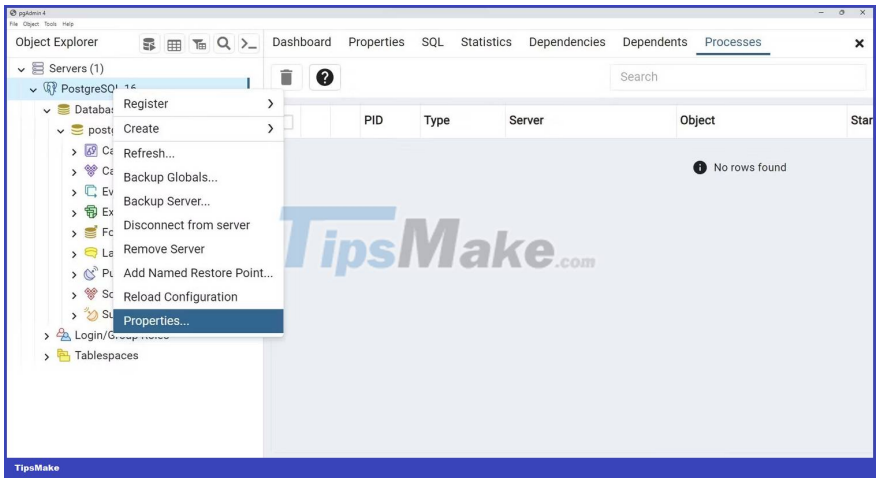
1. From the applications menu, open pgAdmin4.

2. Click the Servers menu on the left side of the application screen.

3. Enter the Postgres password you entered during setup.

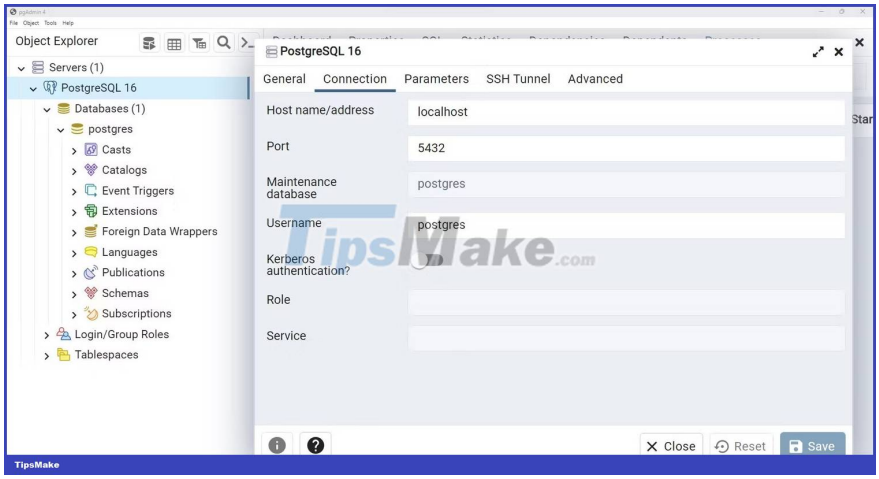


4. After you connect to the server, right-click the PostgreSQL 16 entry, then select Properties.



5. In the Properties dialog box, click Connection.

6. Record Host name, Port number and Username.



Connect to the Postgres server

With your credentials in hand, you can use the psycopg2 library to establish a connection to your Postgres server. To do so, you need to use the connect function as follows:

```
conn = psycopg2.connect(host='localhost', port= '5432', user='postgres', passwor
```

Next, you must use the cursor function to execute Postgres commands in the Python environment:

```
cur = conn.cursor()
```

Finally, you can set the auto-commit flag to ensure Python executes and commits each code statement. This way, you don't need to pass separate commit statements after each line of code.

```
conn.set_session(autocommit = True)
```

You can run these commands at the same time to connect to a local instance of the Postgres server.

How to create a Postgres database

The Postgres database plays a key role in storing a collection of related tables. Create a new command using the CREATE DATABASE SQL command that you can pass to the cursor object's execute method:

```
try: cur.execute('''CREATE DATABASE DB_NAME''') except psycopg2.Error as e: print
```

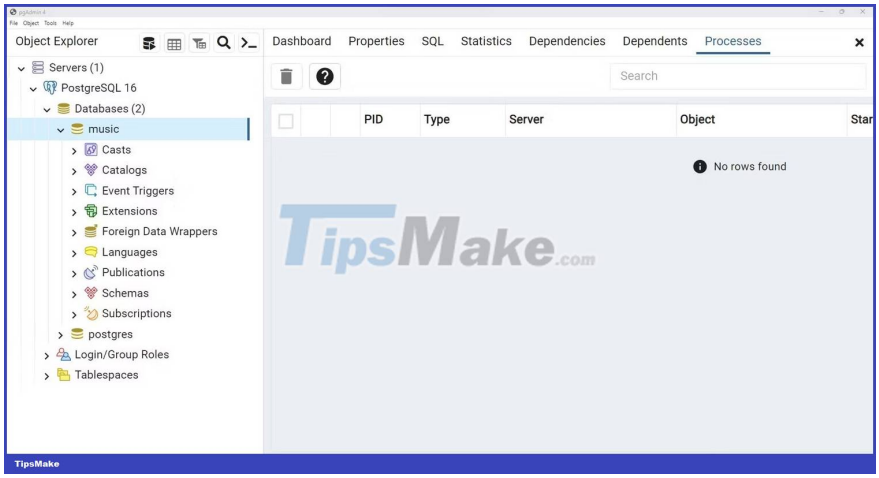
You must always be careful to handle any exceptions that may arise. This example simply prints any errors that occur but in production code you will want to take appropriate action.

```
In [9]: # import the library to run Postgres instance

        import psycopg2

        #establish a new connection
        #enter the host, port, username and password in the connection string as shown

        conn = psycopg2.connect(host='localhost', port= '5432', user='postgres', password='s123')

        # use the cursor function to execute Postgres's commands in a Python environment
        cur = conn.cursor()

        #commit your queries so that Python remembers every execution step

        conn.set_session(autocommit = True)

        # create a new database

        try:
            cur.execute ('''CREATE DATABASE MUSIC''')
        except psycopg2.Error as e:
```


Check the database in pgAdmin4

When you run the above query, you can check if the database using pgAdmin4 has been created successfully. Go to the application interface, refresh the list of existing databases and find the new database.

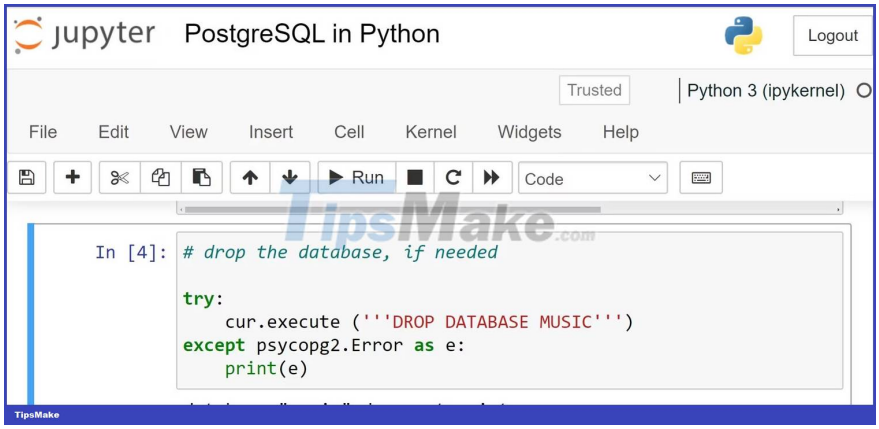
For example, if you create a sample database named music using the above query, it will show up in the list of databases under the Postgres16 > Databases category.



How to delete Postgres database

If you don't want to keep a specific database, you can use the following command to delete it:

```
try: cur.execute('''DROP DATABASE MUSIC''') except psycopg2.Error as e: print(e)
```



Instead of the create command, you need to use the drop command. Once executed, you will not see the database in question.

You finished reading the article "**How to connect and use PostgreSQL in Python**" edited by the **TipsMake** team. We hope this article has provided you with many useful tech tips and tricks. You can search for similar articles on tips and guides. Thank you for reading and for following us regularly.
