First install mysql_connector in the virtual env.
Write a config file dbConfig_Medocare = {

```
   'user': 'root',
   'password':'shahreen',
   'host': 'localhost',
   'database': 'Medicine_List'

}

#this file contains the configuration settings for the db
```

```
from mysql_configa_medocare import dbConfig_Medocare
import _mysql_connector as mycon

sql_cnt = mycon.connect (**dbConfig_Medocare)
#the connect , function basically is used to create connection to the server and
python
#the double asterisks basically means its going to unpack the content of that
dictonary key to the function
cursor_cnt =sql_cnt.cursor()
# A cursor is an object which helps to execute the query and fetch the records from
the database
```

Mysql_configa_medocare is the config.py file name where the dictionary of the db config settings is located .

Cursor object:

-> The cursor object in Python is used to interact with a database after a connection has been established. The cursor allows you to execute SQL commands and queries against the database and fetch the results of those queries.

When you create a cursor object, it's associated with a specific connection to the database. You can then use the cursor object's methods to execute SQL commands and queries against that database connection.

Here are some of the most common methods of the cursor object:

- `execute(sql[, parameters])`: This method is used to execute an SQL statement. The `sql` parameter is the SQL statement you want to execute, and the `parameters` parameter is an optional sequence of values to substitute into the SQL statement. If the SQL statement returns any rows, you can retrieve them using one of the cursor's fetch methods.
- `fetchone()`: This method is used to retrieve the next row of a query result set. It returns a tuple of values representing the columns of the row, or `None` if there are no more rows.
- `fetchmany(size=None)`: This method is used to retrieve multiple rows of a query result set. The `size` parameter is an optional integer that specifies the maximum number of rows to retrieve. If `size` is not specified, the cursor's arraysize attribute is used instead.
- `fetchall()`: This method is used to retrieve all remaining rows of a query result set. It returns a list of tuples, where each tuple represents a row of the result set.
- `rowcount`: This property returns the number of rows affected by the last SQL statement executed by the cursor.
- `close()`: This method is used to close the cursor.

These methods allow you to execute SQL commands and queries against the database and retrieve the results of those queries. The cursor object is an essential part of interacting with databases in Python.

#commit 👍

```
def insert_list (self,medi_name ,pharma_name ,n_of_sa):

      inset_list=(" INSERT INTO medicine(medi_name ,pharma_name,n_of_sa)Values
(%s,%s,%s) ") #%s , placeholder

      values =[medi_name,pharma_name,n_of_sa]

      self.cursor.execute(inset_list,values)

      self.sql_cnt.commit ()

      #This method sends a COMMIT statement to the MySQL server, committing the
current transaction.Since by default Connector/Python does not autocommit, it is
important to call this method after every transaction that modifies data for tables
that use transactional storage engines

      messagebox.showinfo(title="Medicine_List Databse", message= "New medicine is
added to the database")
```

In most relational database systems, such as PostgreSQL, MySQL, and SQLite, changes made to the database are not permanent until they are committed. The commit operation makes the changes permanent and visible to other users and transactions.

how to run your python file from the environment using mysql connector?

To run a Python file that uses the MySQL Connector library to connect to a MySQL database, you need to perform the following steps:

1. Install the MySQL Connector library: If you haven't already installed the MySQL Connector library, you can install it using the pip package manager. Open a terminal or command prompt and enter the following command:

```
pip install mysql-connector-python
```

2.Import the MySQL Connector library: In your Python file, you need to import the `mysql.connector` module to use the MySQL Connector library. You can do this with the following line:

```
import mysql.connector
```

3.Connect to the MySQL database: Next, you need to establish a connection to the MySQL database using the `connect()` method of the `mysql.connector` module. This method takes several parameters, including the host, database name, user name, and password. Here's an example:

```
cnx = mysql.connector.connect(
```

```
    host='localhost',
    database='mydatabase',
    user='myuser',
    password='mypassword'
)
```

4.Create a cursor object: Once you have established a connection to the database, you can create a cursor object using the `cursor()` method of the connection object. The cursor object allows you to execute SQL statements against the database. Here's an example:

```
cursor = cnx.cursor()
```

5.Execute SQL statements: Using the cursor object, you can execute SQL statements against the database. Here's an example:

```
cursor.execute('SELECT * FROM mytable')
```