**Python & MySQL Notes**

# Connect Database

Python MySQLdb provides **MySQLdb.connect()** function to open a database connection. This function takes multiple parameters and returns a connection object to do database operations.

## **Syntax**

db = mysql.connector.connect(host, username, passwd, dbName, port, socket);

**or**

db = MySQLdb.connect(host, username, passwd, dbName, port, socket);

|  |  |
| --- | --- |
| **Sr.No.** | **Parameter & Description** |
| 1 | **host**  Optional − The host name running the database server. If not specified, then the default value will be **localhost:3306**. |
| 2 | **username**  Optional − The username accessing the database. If not specified, then the default will be the name of the user that owns the server process. |
| 3 | **passwd**  Optional − The password of the user accessing the database. If not specified, then the default will be an empty password. |
| 4 | **dbName**  Optional − database name on which query is to be performed. |
| 5 | **port**  Optional − the port number to attempt to connect to the MySQL server.. |
| 6 | **socket**  Optional − socket or named pipe that should be used. |

There are several other properties as well. Refer [MySQLdb](https://mysqlclient.readthedocs.io/user_guide.html" \l "functions-and-attributes) for complete reference.

You can disconnect from the MySQL database anytime using another connection object function **close()**.

1. **Database Operations**

**1.1 Connection db**

import mysql.connector

#Create connections

conn=mysql.connector.connect(

    host="localhost",

    user="Shahriyar",

    password="shary786",

    db='mydatabase'

)

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Checking Connection \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

if conn.is\_connected:

   print("Connection is Done!")

else:

    print("Wrong Connection")

* 1. **Creation db or Exists db**

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Creating Database \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

#Creating a Database  use the "CREATE DATABASE" statement

cr=conn.cursor()

#db name

db\_name="mydatabase"

try:

    cr.execute("CREATE DATABASE {}".format(db\_name))

    print("Database {} created ".format(db\_name))

except:

    print(f"{db\_name} already exists")

* 1. **Show databases**

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Show Database \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

cr.execute("Show databases")

for dbs in cr:

    print(dbs)

* 1. **Db version**

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Database Version \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

cr.execute("select version()")

for ver in cr:

    print("Version available",ver)

* 1. **Droping db**

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Drop Database \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

#Example Drop database (Drop database dbname)

db\_name1='my\_database'

try:

    cr.execute(f"Drop database {db\_name1}")

    print(f"Droped {db\_name1}")

except:

    print(f"{db\_name1} not exists")

* 1. **Renaming db**

1. **Table Operations**

## **2.1 Creating a Table**

To create a table in MySQL, use the "CREATE TABLE" statement.

#Create table with fields

tb=(""" Create table if not exists Customer\_tb(

           cust\_id int not null auto\_increment,

           cust\_name varchar(255) not null,

           cust\_address varchar(255),

           Cust\_Post varchar(255),

           primary key(cust\_id))

           """)

#Execute Sql query using Execute method

cr.execute(tb)

print("Customer\_tb is created")

cr.execute("Create table if not exists sharytb(id int)")

* 1. **Show tables**

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Show tables \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

cr.execute("show tables")

for tbs in cr:

    print("Tables:\n",tbs)

* 1. **Drop table**

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*    Drop tables  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

try:

    cr.execute("Drop table sharytb")

    print("sharytb is droped")

except:

    print("sharytb is not exist")

* 1. **Alter table**

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Alter table  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print("\n\*\*\*\*\*\*\*\*\*\*\* Add column \*\*\*\*\*\*\*\*\*\*\*\*\*")

# Add cust\_salary column

cr.execute("Alter table Customer\_tb add cust\_salary varchar(255)")

print("Added the Cust\_salary column" )

print("\n\*\*\*\*\*\*\*\*\*\*\* modify column \*\*\*\*\*\*\*\*\*\*\*\*\*")

# cr.execute("ALTER TABLE Customer\_tb drop column cust\_name")

# print("Droped column cust\_salary")

**Important Points:**

## **Primary Key**

* + When creating a table, you should also create a column with a unique key for each record.
  + This can be done by defining a PRIMARY KEY.
  + We use the statement "INT AUTO\_INCREMENT PRIMARY KEY" which will insert a unique number for each record. Starting at 1, and increased by one for each record.

1. **Cursor object:**
   * In database programming, a cursor is a database object used to traverse and manipulate data returned by a query.
   * Cursors provide a way to navigate through the result set of a query one row at a time,
   * enabling operations such as fetching individual rows,
   * iterating over the result set, updating data, and deleting data.
2. **Commit()**

 Notice the statement: **mydb.commit().** It is required to make the changes, otherwise no changes are made to the table.

## **Get Inserted ID**

You can get the id of the row you just inserted by asking the cursor object.

**Note:** If you insert more than one row, the id of the last inserted row is returned.

# Insert Into Table

# To fill a table in MySQL, use the "INSERT INTO" statement.

# 4.1 Insert One Row

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Insert single row into table  \*\*\*\*\*\*\*\*\*\*")

cr.execute("""Insert into Customer\_tb (cust\_name,cust\_address,cust\_Post)

           values ("Shary","Charsadda","HR")

           """)

#Save the record

conn.commit()

#get the row id for inserted record

print("Id:",cr.lastrowid)

#print the number of records inserted

print(cr.rowcount,"Record inserted")

# 4.2 Insert Multiple Rows

* To insert multiple rows into a table, use the **executemany()** method.
* The second parameter of the **executemany()** method is a list of tuples, containing the data you want to insert:

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Insert multiple row into table  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

cr.execute("""Insert into Customer\_tb (cust\_name,cust\_address,cust\_Post)

           values ("Shary","Peshawer","MR"),

                  ("Umair","Charsadda","GR"),

                  ("Hamad","Charsadda","HR"),

                  ("Ali","Mardan","SR")

           """)

#Save the record

conn.commit()

#get the row id for inserted record

print("Id:",cr.lastrowid)

#print the number of records inserted

print(cr.rowcount,"Record inserted")

# Select Records Example

[Previous](https://www.tutorialspoint.com/python_mysql/python_mysql_insert_records.htm)

[Next](https://www.tutorialspoint.com/python_mysql/python_mysql_update_records.htm)

Select/Read Operation on any database means to fetch some useful information from the database.

Once our database connection is established, you are ready to make a query into this database. You can use either **fetchone()** method to fetch single record or **fetchall()** method to fetch multiple values from a database table.

* **fetchone()** − It fetches the next row of a query result set. A result set is an object that is returned when a cursor object is used to query a table.
* **fetchall()** − It fetches all the rows in a result set. If some rows have already been extracted from the result set, then it retrieves the remaining rows from the result set.
* **rowcount** − This is a read-only attribute and returns the number of rows that were affected by an execute() method.

## **Syntax**

# execute SQL query using execute() method.

cursor.execute(sql)

result = cursor.fetchall()

for record in result:

print(record)

1. **Retrieve all data (Using Fetchall()**

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Select from table   \*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Fetchall()   \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

cr.execute("Select \* from Customer\_tb")

rows=cr.fetchall()

print("Cust\_Id\t\tCust\_name\tCust\_Address\t\tCust\_Post")

print("------------------------------------------------------------")

for row in rows:

    print(f"{row[0]}\t\t{row[1]}\t\t{row[2]}\t\t{row[3]}")

1. **Retrieve one first row (Using Fetchone()**

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Fetchone()   \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

cr.execute("Select \* from Customer\_tb ")

row=cr.fetchone()

print(row)

## **6.1 Where Claus**

When selecting records from a table, you can filter the selection by using the "WHERE" statement:

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Select with filter    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

cr.execute("""SELECT \* FROM Customer\_tb WHERE cust\_name = 'Shary'""")

result = cr.fetchall()

for row in result:

    print(row)

# 6.2 Like Claus (Whild card)

print("\n\*\*\*\*\*\*\*\*\*\*\*\* Wildcard char \*\*\*\*\*\*\*\*\*\*")

cr.execute("""SELECT \* FROM Customer\_tb WHERE Cust\_address like '%adda%'""")

result = cr.fetchall()

for row in result:

    print(row)

## **6.3 Prevent SQL Injection**

When query values are provided by the user, you should escape the values.

This is to prevent SQL injections, which is a common web hacking technique to destroy or misuse your database.

The mysql.connector module has methods to escape query values:

print("\n\*\*\*\*\*\*\*\*\*\*\*\* %s Escap char \*\*\*\*\*\*\*\*\*\*")

sql="Select \* from Customer\_tb where Cust\_address=%s"

adr=("Peshawer",)

cr.execute(sql,adr)

result=cr.fetchall()

for rows in result:

    print(rows)

# Order By

## **Sort the Result**

Use the ORDER BY statement to sort the result in ascending or descending order.

The ORDER BY keyword sorts the result ascending by default. To sort the result in descending order, use the DESC keyword.

cr.execute("Select \* from Customer\_tb order by Cust\_name")

result=cr.fetchall()

## **ORDER BY DESC**

Use the DESC keyword to sort the result in a descending order.

print("\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Order by desc \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n")

cr.execute("Select \* from Customer\_tb order by Cust\_name desc")

result=cr.fetchall()

print("Cust\_ID\t Cust\_name\t Cust\_Address\t Cust\_Post")

print("-------------------------------------------------------")

for rows in result:

    print(f"{rows[0]}\t{rows[1]}\t\t{rows[2]}\t{rows[3]}")

# Delete From By

## **8.1 Delete Record**

You can delete records from an existing table by using the "DELETE FROM" statement:

print("\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Delete Records  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n")

cr.execute("Delete from Customer\_tb where Cust\_address='Peshawer'")

print(cr.rowcount,"Record(s) Deleted")

## **8.2 Prevent SQL Injection**

It is considered a good practice to escape the values of any query, also in delete statements.

This is to prevent SQL injections, which is a common web hacking technique to destroy or misuse your database.

The mysql.connector module uses the placeholder %s to escape values in the delete statement:

print("\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Delete Records using Escap char  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n")

sql=("Delete from Customer\_tb where Cust\_address=%s")

adr=("Charsadda",)

cr.execute(sql,adr)

print(cr.rowcount,"Record(s) Deleted")

## **Update Table**

You can update existing records in a table by using the "UPDATE" statement:

print("\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Update Records  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n")

cr.execute("Update Customer\_tb set Cust\_address='Swaat' where Cust\_address='Charsadda'")

print(cr.rowcount,"Record(s) Updated")

## **Limit the Result**

You can limit the number of records returned from the query, by using the "LIMIT" statement:

print("\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Limit Result 4 rows  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n")

cr.execute("Select \* from Customer\_tb limit 4")

rows=cr.fetchall()

print("Cust\_Id\t\tCust\_name\tCust\_Address\t\tCust\_Post")

print("------------------------------------------------------------")

for row in rows:

    print(f"{row[0]}\t\t{row[1]}\t\t{row[2]}\t\t{row[3]}")

## **10.2 Start From Another Position**

If you want to return five records, starting from the third record, you can use the "OFFSET" keyword: