Table of Contents

1. Python MySQL — Introduction

The Python standard for database interfaces is the Python DB-API. Most Python database interfaces adhere to this standard.

You can choose the right database for your application. Python Database API supports a wide range of database servers such as:

- GadFly
- mSQL
- MySQL
- PostgreSQL
- Microsoft SQL Server 2000
- Informix
- Interbase
- Oracle
- Sybase

Here is the list of available Python database interfaces: Python Database Interfaces and APIs .You must download a separate DB API module for each database you need to access. For example, if you need to access an Oracle database as well as a MySQL database, you must download both the Oracle and the MySQL database modules.

What is mysql-connector-python?

MySQL Python/Connector is an interface for connecting to a MySQL database server from Python. It implements the Python Database API and is built on top of the MySQL.

Howdo I Installmysql-connector-python?

First of all, you need to make sure you have already installed python in your machine. To do so, open command prompt and type python in it and press *Enter*. If python is already installed in your system, this command will display its version as shown below:

```
C:\Users\Tutorialspoint>python
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Now press *ctrl+z* and then *Enter* to get out of the python shell and create a folder (in which you intended to install Python-MySQL connector) named Python_MySQL as:

```
>>> ^Z
```

```
C:\Users\Tutorialspoint>d:
D:\>mkdir Python_MySQL
```

Verify PIP

PIP is a package manager in python using which you can install various modules/packages in Python. Therefore, to install Mysql-python mysql-connector-python you need to make sure that you have PIP installed in your computer and have its location added to path.

You can do so, by executing the pip command. If you didn't have PIP in your system or, if you haven't added its location in the ***Path*** environment variable, you will get an error message as:

```
D:\Python_MySQL>pip
'pip' is not recognized as an internal or external command,
operable program or batch file.
```

To install PIP, download the get-pip.py to the above created folder and, from command navigate it and install pip as follows:

```
D:\>cd Python_MySQL
D:\Python_MySQL>python get-pip.py
Collecting pip
  Downloading
https://files.pythonhosted.org/packages/8d/07/f7d7ced2f97ca3098c16565efbe6b15fa
fcba53e8d9bdb431e09140514b0/pip-19.2.2-py2.py3-none-any.whl (1.4MB)
     |████████████████████████████████| 1.4MB 1.3MB/s
Collecting wheel
  Downloading
https://files.pythonhosted.org/packages/00/83/b4a77d044e78ad1a45610eb88f745be2f
d2c6d658f9798a15e384b7d57c9/wheel-0.33.6-py2.py3-none-any.whl
Installing collected packages: pip, wheel
  Consider adding this directory to PATH or, if you prefer to suppress this
warning, use --no-warn-script-location.
Successfully installed pip-19.2.2 wheel-0.33.6
```

Installing mysql-connector-python

Once you have Python and PIP installed, open command prompt and upgrade pip (optional) as shown below:

```
C:\Users\Tutorialspoint>python -m pip install --upgrade pip
Collecting pip Using cached


https://files.pythonhosted.org/packages/8d/07/f7d7ced2f97ca3098c16565efbe6b15fa
fcba53e8d9bdb431e09140514b0/pip-19.2.2-py2.py3-none-any.whl
```

```
Installing collected packages: pip
  Found existing installation: pip 19.0.3
    Uninstalling pip-19.0.3:
      Successfully uninstalled pip-19.0.3
Successfully installed pip-19.2.2
```

Then open command prompt in admin mode and install python MySQL connect as:

```
C:\WINDOWS\system32>pip install mysql-connector-python
Collecting mysql-connector-python
  Using cached
https://files.pythonhosted.org/packages/99/74/f41182e6b7aadc62b038b6939dce784b7
f9ec4f89e2ae14f9ba8190dc9ab/mysql_connector_python-8.0.17-py2.py3-none-any.whl
Collecting protobuf>=3.0.0 (from mysql-connector-python)
  Using cached
https://files.pythonhosted.org/packages/09/0e/614766ea191e649216b87d331a4179338
c623e08c0cca291bcf8638730ce/protobuf-3.9.1-cp37-cp37m-win32.whl
Collecting six>=1.9 (from protobuf>=3.0.0->mysql-connector-python)
  Using cached
https://files.pythonhosted.org/packages/73/fb/00a976f728d0d1fecfe898238ce23f502
a721c0ac0ecfedb80e0d88c64e9/six-1.12.0-py2.py3-none-any.whl Requirement already
satisfied: setuptools in c:\program files (x86)\python37-
32\lib\site-packages (from protobuf>=3.0.0->mysql-connector-python) (40.8.0)
Installing collected packages: six, protobuf, mysql-connector-python
Successfully installed mysql-connector-python-8.0.17 protobuf-3.9.1 six-1.12.0
```

Verification

To verify the installation of the create a sample python script with the following line in it.

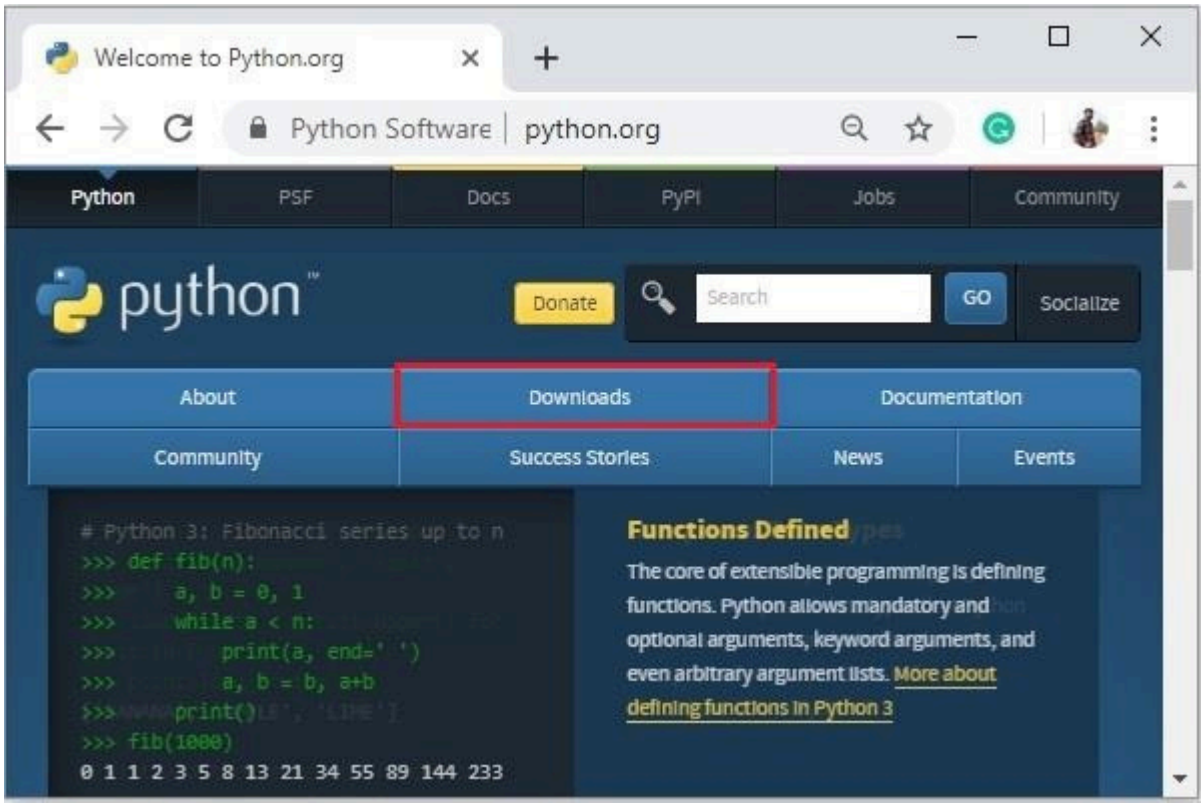
```
import mysql.connector
```

If the installation is successful, when you execute it, you should not get any errors:

```
D:\Python_MySQL>python test.py
D:\Python_MySQL>
```

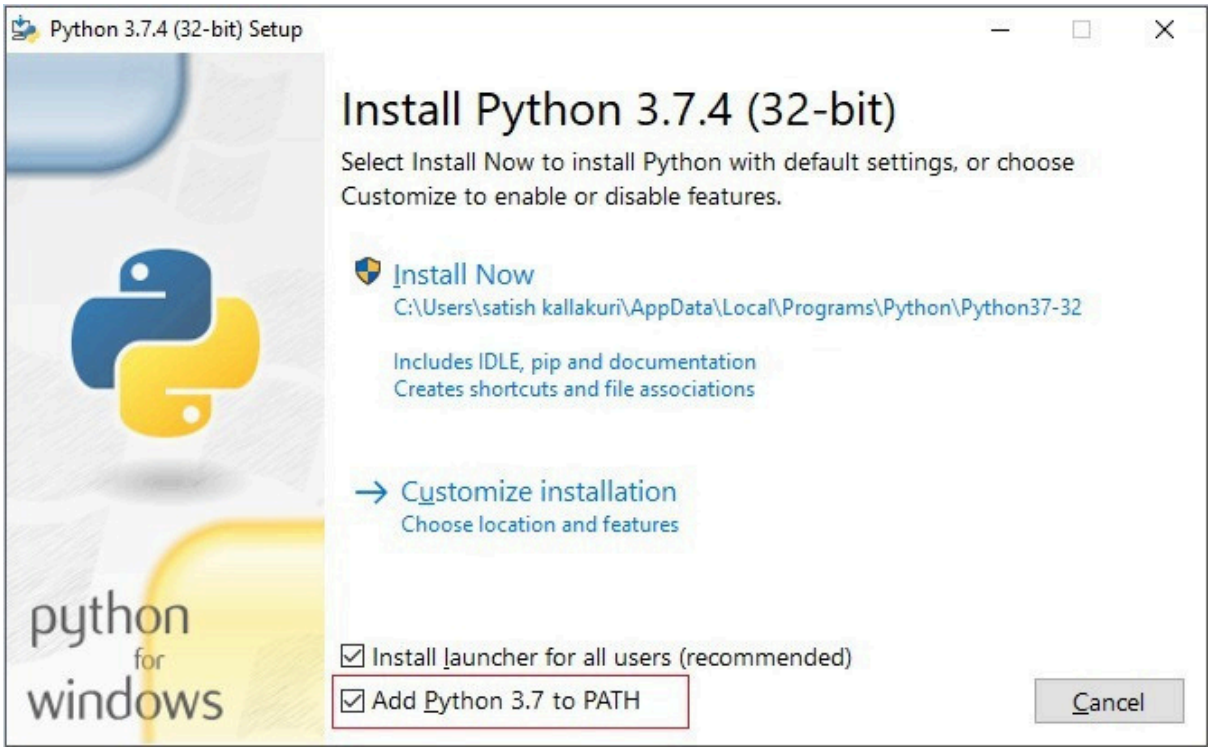
Installing python from scratch

Simply, if you need to install Python from scratch. Visit the Python Home Page.



Click on the ***Downloads*** button, you will be redirected to the downloads page which provides links for latest version of python for various platforms choose one and download it.

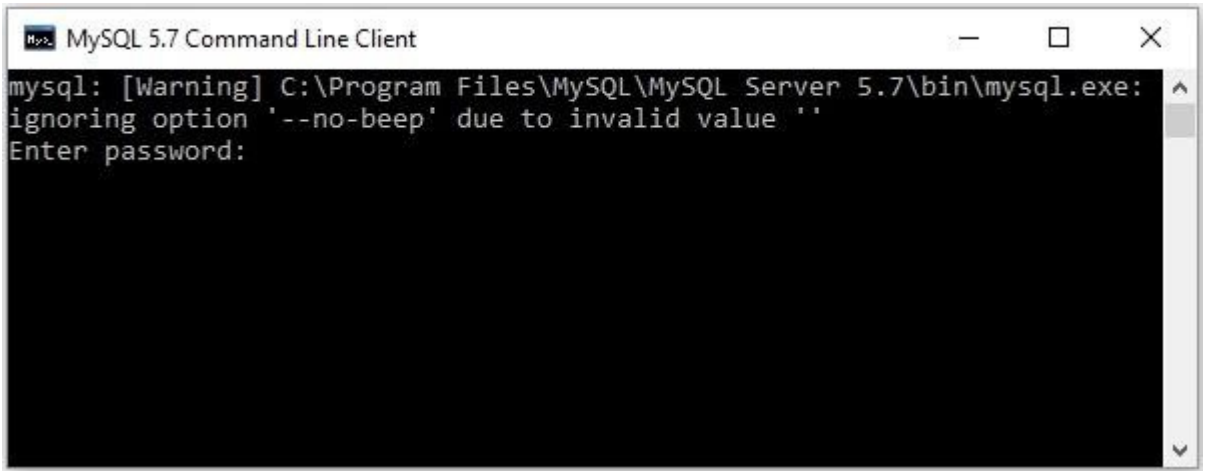
For instance, we have downloaded python-3.7.4.exe (for windows). Start the installation process by double-clicking the downloaded *.exe* file.



Check the Add Python 3.7 to Path option and proceed with the installation. After completion of this process, python will be installed in your system.

2. Python MySQL — Database ConnPeythcont iN

To connect with MySQL, (one way is to) open the MySQL command prompt in your system as shown below:



It asks for password here; you need to type the password you have set to the default user (root) at the time of installation.

Then a connection is established with MySQL displaying the following message:

```
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.7.12-log MySQL Community Server (GPL)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

You can disconnect from the MySQL database any time using the **exit** command at *mysql>* prompt.

```
mysql> exit
Bye
```

Establishing connection with MySQL using python

Before establishing connection to MySQL database using python, assume:

That we have created a database with name mydb.

We have created a table EMPLOYEE with columns FIRST_NAME, LAST_NAME, AGE, SEX and INCOME.

The credentials we are using to connect with MySQL are username: ***root***, password: ***password.***

You can establish a connection using the ***connect()*** constructor. This accepts username, password, host and, name of the database you need to connect with (optional) and, returns an object of the MySQLConnection class.

Example

Following is the example of connecting with MySQL database "mydb".

```
import mysql.connector


#establishing the connection
conn = mysql.connector.connect(user='root', password='password',
host='127.0.0.1', database='mydb')


#Creating a cursor object using the cursor() method
cursor = conn.cursor()


#Executing an MYSQL function using the execute() method
cursor.execute("SELECT DATABASE()")


# Fetch a single row using fetchone() method.
data = cursor.fetchone()


print("Connection established to: ",data)


#Closing the connection
conn.close()
```

On executing, this script produces the following output:

```
D:\Python_MySQL>python EstablishCon.py
Connection established to: ('mydb',)
```

You can also establish connection to MySQL by passing credentials (user name, password, hostname, and database name) to *connection.MySQLConnection()* as shown below:

```
from mysql.connector import (connection)


#establishing the connection
conn = connection.MySQLConnection(user='root', password='password',
host='127.0.0.1', database='mydb')


#Closing the connection
conn.close()
```

3. Python MySQL — Create Database

You can create a database in MYSQL using the CREATE DATABASE query.

Syntax
Following is the syntax of the CREATE DATABASE query:

```
CREATE DATABASE name_of_the_database
```

Example
Following statement creates a database with name mydb in MySQL:

```
mysql> CREATE DATABASE  mydb;
Query OK, 1 row affected (0.04 sec)
```

If you observe the list of databases using the SHOW DATABASES statement, you can observe the newly created database in it as shown below:

```
mysql> SHOW DATABASES; +--
------------------+      |
Database | +--------------
------+                   |
information_schema    |   |
logging | | mydatabase | |
mydb           |          |
performance_schema    |   |
students | | sys | +------
--------------+ 26 rows in
set (0.15 sec)
```

Creating a database in MySQL using python

After establishing connection with MySQL, to manipulate data in it you need to connect to a database. You can connect to an existing database or, create your own.

You would need special privileges to create or to delete a MySQL database. So if you have access to the root user, you can create any database.

Example

Following example establishes connection with MYSQL and creates a database in it.

```
import mysql.connector


#establishing the connection
conn = mysql.connector.connect(user='root', password='password',
host='127.0.0.1')


#Creating a cursor object using the cursor() method
cursor = conn.cursor()


#Doping database MYDATABASE if already exists.
cursor.execute("DROP database IF EXISTS MyDatabase")


#Preparing query to create a database
sql = "CREATE database MYDATABASE";


#Creating a database
cursor.execute(sql)


#Retrieving the list of databases
print("List of databases: ")
cursor.execute("SHOW DATABASES")
print(cursor.fetchall())


#Closing the connection
conn.close()
```

Output

```
List of databases:
[('information_schema',), ('dbbug61332',), ('details',), ('exampledatabase',),
('mydatabase',), ('mydb',), ('mysql',), ('performance_schema',)]
```

4. Python MySQL — Create Table

The CREATE TABLE statement is used to create tables in MYSQL database. Here, you need to specify the name of the table and, definition (name and datatype) of each column.

Syntax

Following is the syntax to create a table in MySQL:

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

Example

Following query creates a table named EMPLOYEE in MySQL with five columns namely, FIRST_NAME, LAST_NAME, AGE, SEX and, INCOME.

```
mysql> CREATE TABLE EMPLOYEE(
    FIRST_NAME CHAR(20) NOT NULL,
    LAST_NAME CHAR(20),
    AGE INT,
    SEX CHAR(1),
    INCOME FLOAT);
Query OK, 0 rows affected (0.42 sec)
```

The DESC statement gives you the description of the specified table. Using this you can verify if the table has been created or not as shown below:

```
                                    mysql> Desc Employee;
  +------------+----------+------+-----+---------+-------+
| Field      | Type     | Null | Key | Default | Extra   |
  +------------+----------+------+-----+---------+-------+
    | FIRST_NAME | char(20) | |N O   | NULL    |         |
    | LAST_NAME | char(20) | YES |   | NULL    |         |
|  AGE       |  int(11)    | YES |   | NULL    |         |
| SEX        | char(1)    | YES |    | NULL    |         |
```

```
| INCOME | float | YES | | NULL|| +------------+---------
-+------+-----+---------+-------+ 5 rows in set (0.07 sec)
```

Creating a table in MySQL using python

The method named ***execute()*** (invoked on the cursor object) accepts two variables:

A String value representing the query to be executed.

An optional args parameter which can be a tuple or, list or, dictionary, representing the parameters of the query (values of the place holders).

It returns an integer value representing the number of rows effected by the query.

Once a database connection is established, you can create tables by passing the CREATE TABLE query to the **execute()** method.

In short, to create a table using python:

Import ***mysql.connector*** package.

Create a connection object using the ***mysql.connector.connect()*** method, by passing the user name, password, host (optional default: localhost) and, database (optional) as parameters to it.

Create a cursor object by invoking the ***cursor()*** method on the connection object created above.

Then, execute the *CREATE TABLE* statement by passing it as a parameter to the **execute()** method.

Example

Following example creates a table named **Employee** in the database mydb.

```
import mysql.connector


#establishing the connection
conn = mysql.connector.connect(user='root', password='password',
host='127.0.0.1', database='mydb')


#Creating a cursor object using the cursor() method
cursor = conn.cursor()
#Dropping EMPLOYEE table if already exists.
cursor.execute("DROP TABLE IF EXISTS EMPLOYEE")


#Creating table as per requirement
```

```
sql ='''CREATE TABLE EMPLOYEE(
         FIRST_NAME  CHAR(20) NOT NULL,
         LAST_NAME CHAR(20),
         AGE INT,
         SEX CHAR(1),
         INCOME FLOAT)'''
cursor.execute(sql)

#Closing the connection
conn.close()
```

5. Python MySQL — Insert Data

You can add new rows to an existing table of MySQL using the **INSERT INTO** statement. In this, you need to specify the name of the table, column names, and values (in the same order as column names).

Syntax

Following is the syntax of the INSERT INTO statement of MySQL.

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

Example

Following query inserts a record into the table named EMPLOYEE.

```
INSERT INTO EMPLOYEE(FIRST_NAME, LAST_NAME, AGE, SEX, INCOME) VALUES ('Mac',
'Mohan', 20, 'M', 2000);
```

You can verify the records of the table after insert operation using the SELECT statement as:

```
mysql> select * from Employee;
+------------+-----------+------+------+--------+
| FIRST_NAME | LAST_NAME | AGE  | SEX  | INCOME |
+------------+-----------+------+------+--------+
| Mac        | Mohan     |   20 | M    |   2000 |
+------------+-----------+------+------+--------+
1 row in set (0.00 sec)
```

It is not mandatory to specify the names of the columns always, if you pass values of a record in the same order of the columns of the table you can execute the SELECT statement without the column names as follows:

```
INSERT INTO EMPLOYEE VALUES ('Mac', 'Mohan', 20, 'M', 2000);
```

Inserting data in MySQL table using python

The **execute()** method (invoked on the cursor object) accepts a query as parameter and executes the given query. To insert data, you need to pass the MySQL INSERT statement as a parameter to it.

```
cursor.execute("""INSERT INTO EMPLOYEE(FIRST_NAME, LAST_NAME, AGE, SEX, INCOME)
VALUES ('Mac', 'Mohan', 20, 'M', 2000)""")
```

To insert data into a table in MySQL using python:

import ***mysql.connector*** package.

Create a connection object using the ***mysql.connector.connect()*** method, by passing the user name, password, host (optional default: localhost) and, database (optional) as parameters to it.

Create a cursor object by invoking the ***cursor()*** method on the connection object created above.
Then, execute the ***INSERT*** statement by passing it as a parameter to the **execute()** method.

Example

The following example executes SQL *INSERT* statement to insert a record into the EMPLOYEE table:

```
import mysql.connector


#establishing the connection
conn = mysql.connector.connect(user='root', password='password',
host='127.0.0.1', database='mydb')


#Creating a cursorobject  using thecursor() method
cursor =conn.cursor()


# Preparing SQLqueryto   INSERT a record into the database.
sql = """INSERTINTO  EMPLOYEE(FIRST_NAME,
         LAST_NAME,AGE, SEX,INCOME)
         VALUES ('Mac', 'Mohan', 20, 'M', 2000)"""
try:
   # Executing theSQLcommand
   cursor.execute(sql)
   # Commit  yourchanges  in the database
   conn.commit()
except:
   # Rollingback in case of error
   conn.rollback()
```

```
# Closing the connection
conn.close()
```

Inserting values dynamically

You can also use "*%s*" instead of values in the **INSERT** query of MySQL and pass values to them as lists as shown below:

```
cursor.execute("""INSERT INTO EMPLOYEE VALUES ('Mac', 'Mohan', 20, 'M',
2000)""", ('Ramya', 'Ramapriya', 25, 'F', 5000))
```

Example

Following example inserts a record into the Employee table dynamically.

```
import mysql.connector


#establishing the connection
conn = mysql.connector.connect(user='root', password='password',
host='127.0.0.1', database='mydb')


#Creating a cursor object using the cursor() method
cursor = conn.cursor()


# Preparing SQL query to INSERT a record into the database.
insert_stmt = (
  "INSERT INTO EMPLOYEE(FIRST_NAME, LAST_NAME, AGE, SEX, INCOME)"
  "VALUES (%s, %s, %s, %s, %s)"
)
data = ('Ramya', 'Ramapriya', 25, 'F', 5000)


try:
   # Executing the SQL command
   cursor.execute(insert_stmt, data)
   # Commit your changes in the database
   conn.commit()
except:
   # Rolling back in case of error
   conn.rollback()


print("Data inserted")
```

```
# Closing the connection
conn.close()
```

Output

```
Data inserted
```

6. Python MySQL — Select Data

You can retrieve/fetch data from a table in MySQL using the SELECT query. This query/statement returns contents of the specified table in tabular form and it is called as result-set.

Syntax

Following is the syntax of the SELECT query:

```
SELECT column1, column2, columnN FROM table_name;
```

Example

Assume we have created a table in MySQL with name *cricketers_data* as:

```
CREATE TABLE cricketers_data(
First_Name VARCHAR(255),
Last_Name VARCHAR(255),
Date_Of_Birth date,
Place_Of_Birth VARCHAR(255),
Country VARCHAR(255)
);
```

And if we have inserted 5 records in to it using INSERT statements as:

```
insert into cricketers_data values('Shikhar', 'Dhawan', DATE('1981-12-05'),
'Delhi', 'India');
insert into cricketers_data values('Jonathan', 'Trott', DATE('1981-04-22'),
'CapeTown', 'SouthAfrica');
insert into cricketers_data values('Kumara', 'Sangakkara', DATE('1977-10-27'),
'Matale', 'Srilanka');
insert into cricketers_data values('Virat', 'Kohli', DATE('1988-11-05'),
'Delhi', 'India');
insert into cricketers_data values('Rohit', 'Sharma', DATE('1987-04-30'),
'Nagpur', 'India');
```

Following query retrieves the FIRST_NAME and Country values from the table.

```
mysql> select FIRST_NAME, Country  from cricketers_data;
+------------+-------------+ |
| FIRST_NAME | Country
```

```
+------------+-------------+
| Shikhar | India| | Jonathan
|  SouthAfrica|  |  Kumara  |
Srilanka| | Virat | India| |
Rohit | India| +--- ---------
+-------------+ 5 rows in set
(0.00 sec)
```

You can also retrieve all the values of each record using * instated of the name of the columns as:

```
                                    mysql> SELECT * from cricketers_data;
   +------------+------------+---------------+----------------+-------------+
| First_Name | Last_Name  | Date_Of_Birth | Place_Of_Birth | Country        |
   +------------+------------+---------------+----------------+-------------+
 | Shikhar |  | Dhawan       | 1981-12-05   | Delhi          |   India    |    |
Jonathan   |  | Trott       | 1981-04-22    | CapeTown        SouthAfrica |    |
Kumara     |  | Sangakkara | 1977-10-27    | Matale         Srilanka     |    |
Virat      |  || 1K9o8h8l-i11-05           | Delhi          India | India    |
Rohit        || 1S9h8a7r-m0a4-30           | Nagpur                          |
   +------------+------------+---------------+----------------+-------------+
                                                  5 rows in set (0.00 sec)
```

Reading data from a MYSQL table using Python

READ Operation on any database means to fetch some useful information from the database. You can fetch data from MYSQL using the **fetch()** method provided by the mysql-connector-python.

The *cursor.MySQLCursor* class provides three methods namely **fetchall()**, **fetchmany()** and, **fetchone()** where,

The **fetchall()** method retrieves all the rows in the result set of a query and returns them as list of tuples. (If we execute this after retrieving few rows it returns the remaining ones).

The **fetchone()** method fetches the next row in the result of a query and returns it as a tuple.

The **fetchmany()** method is similar to the fetchone() but, it retrieves the next set of rows in the result set of a query, instead of a single row.

Note: A result set is an object that is returned when a cursor object is used to query a table.

rowcount: This is a read-only attribute and returns the number of rows that were affected by an execute() method.

Example

Following example fetches all the rows of the EMPLOYEE table using the SELECT query and from the obtained result set initially, we are retrieving the first row using the fetchone() method and then fetching the remaining rows using the fetchall() method.

```
import mysql.connector


#establishing the connection
conn = mysql.connector.connect(user='root', password='password',
host='127.0.0.1', database='mydb')


#Creating a cursor object using the cursor() method
cursor = conn.cursor()


#Retrieving single row
sql = '''SELECT * from EMPLOYEE'''


#Executing the query
cursor.execute(sql)


#Fetching 1st row from the table
result = cursor.fetchone();
print(result)


#Fetching 1st row from the table
result = cursor.fetchall();
print(result)


#Closing the connection
conn.close()
```

Output

```
('Krishna', 'Sharma', 19, 'M', 2000.0)
[('Raj', 'Kandukuri', 20, 'M', 7000.0), ('Ramya', 'Ramapriya', 25, 'M',
5000.0)]
```

Following example retrieves first two rows of the EMPLOYEE table using the fetchmany() method.

Example

```
import mysql.connector


#establishing the connection
conn = mysql.connector.connect(user='root', password='password',
host='127.0.0.1', database='mydb')


#Creating a cursor object using the cursor() method
cursor = conn.cursor()


#Retrieving single row
sql = '''SELECT * from EMPLOYEE'''


#Executing the query
cursor.execute(sql)


#Fetching 1st row from the table
result = cursor.fetchmany(size =2);
print(result)


#Closing the connection
conn.close()
```

Output

```
[('Krishna', 'Sharma', 19, 'M', 2000.0), ('Raj', 'Kandukuri', 20, 'M', 7000.0)]
```

7. Python MySQL — Where Clause

If you want to fetch, delete or, update particular rows of a table in MySQL, you need to use the where clause to specify condition to filter the rows of the table for the operation.

For example, if you have a SELECT statement with where clause, only the rows which satisfies the specified condition will be retrieved.

Syntax

Following is the syntax of the WHERE clause:

```
SELECT column1, column2, columnN
FROM table_name
WHERE [condition]
```

Example

Assume we have created a table in MySQL with name EMPLOYEES as:

```
mysql> CREATE TABLE EMPLOYEE(
FIRST_NAME CHAR(20) NOT NULL,
LAST_NAME CHAR(20),
AGE INT,
SEX CHAR(1),
INCOME FLOAT);
Query OK, 0 rows affected (0.36 sec)
```

And if we have inserted 4 records in to it using INSERT statements as:

```
mysql> INSERT INTO EMPLOYEE VALUES
     ('Krishna', 'Sharma', 19, 'M', 2000),
     ('Raj', 'Kandukuri', 20, 'M', 7000),
     ('Ramya', 'Ramapriya', 25, 'F', 5000),
     ('Mac', 'Mohan', 26, 'M', 2000);
```

Following MySQL statement retrieves the records of the employees whose income is greater than 4000.

```
mysql> SELECT * FROM EMPLOYEE WHERE INCOME > 4000;
+------------+-----------+------+------+--------+ |
| FIRST_NAME | LAST_NAME | AGE  | SEX  | INCOME
```

```
+------------+-----------+------+------+--------+ |
Raj  |  Kandukuri  |  20  |  M  |  7000|  |  Ramya  |
Ramapriya | 25 | F | 5000| +------------+---------
--+------+------+--------+ 2 rows in set (0.00 sec)
```

WHERE clause using python

To fetch specific records from a table using the python program:

import ***mysql.connector*** package.

Create a connection object using the ***mysql.connector.connect()*** method, by passing the user name, password, host (optional default: localhost) and, database (optional) as parameters to it.

Create a cursor object by invoking the ***cursor()*** method on the connection object created above.
Then, execute the *SELECT* statement with *WHERE* clause, by passing it as a parameter to the **execute()** method.

Example

Following example creates a table named Employee and populates it. Then using the where clause it retrieves the records with age value less than 23.

```
import mysql.connector


#establishing the connection
conn = mysql.connector.connect(user='root', password='password',
host='127.0.0.1', database='mydb')


#Creating a cursor object using the cursor() method
cursor = conn.cursor()


#Doping EMPLOYEE table if already exists.
cursor.execute("DROP TABLE IF EXISTS EMPLOYEE")
sql = '''CREATE TABLE EMPLOYEE(
         FIRST_NAME  CHAR(20) NOT NULL,
         LAST_NAME CHAR(20),
         AGE INT,
         SEX CHAR(1),
```

```
         INCOME FLOAT)'''
cursor.execute(sql)


#Populating the table
insert_stmt = "INSERT INTO EMPLOYEE (FIRST_NAME, LAST_NAME, AGE, SEX, INCOME)
VALUES (%s, %s, %s, %s, %s)"
data = [('Krishna','Sharma', 19, 'M', 2000),('Raj', 'Kandukuri', 20, 'M',7000),
      ('Ramya', 'Ramapriya', 25, 'F', 5000),('Mac', 'Mohan', 26, 'M', 2000)]
cursor.executemany(insert_stmt, data)
conn.commit()


#Retrieving specific records using the where clause
cursor.execute("SELECT * from EMPLOYEE WHERE AGE <23")
print(cursor.fetchall())


#Closing the connection
conn.close()
```

Output

```
 [('Krishna', 'Sharma', 19, 'M', 2000.0), ('Raj', 'Kandukuri', 20, 'M',
7000.0)]
```

8. Python MySQL — Order By

While fetching data using SELECT query, you can sort the results in desired order (ascending or descending) using the OrderBy clause. By default, this clause sorts results in ascending order, if you need to arrange them in descending order you need to use “DESC” explicitly.

Syntax

Following is the syntax SELECT column-list

```
FROM table_name
[WHERE condition]
[ORDER BY column1, column2,.. columnN] [ASC | DESC]; of the ORDER BY clause:
```

Example

Assume we have created a table in MySQL with name EMPLOYEES as:

```
mysql> CREATE TABLE EMPLOYEE(
FIRST_NAME CHAR(20) NOT NULL,
LAST_NAME CHAR(20),
AGE INT,
SEX CHAR(1),
INCOME FLOAT);
Query OK, 0 rows affected (0.36 sec)
```

And if we have inserted 4 records in to it using INSERT statements as:

```
mysql> INSERT INTO EMPLOYEE VALUES
     ('Krishna', 'Sharma', 19, 'M', 2000),
     ('Raj', 'Kandukuri', 20, 'M', 7000),
     ('Ramya', 'Ramapriya', 25, 'F', 5000),
     ('Mac', 'Mohan', 26, 'M', 2000);
```

Following statement retrieves the contents of the EMPLOYEE table in ascending order of the age.

```
mysql> SELECT * FROM EMPLOYEE ORDER BY AGE; +-----
-------+-----------+------+------+--------+ |
| FIRST_NAME | LAST_NAME | AGE   | SEX   | INCOME
```

```
+------------+-----------+------+------+--------+ |
Krishna | Sharma | 19 | M | 2000| | Raj | Kandukuri
| 20 | M | 7000| | Ramya | Ramapriya | 25 | F |
5000| | Mac | Mohan | 26 | M | 2000| +--- 4
---------+-----------+------+------+--------+
rows in set (0.04 sec)
```

You can also retrieve data in descending order using DESC as:

```
    mysql> SELECT * FROM EMPLOYEE ORDER BY FIRST_NAME, INCOME DESC;
                  +------------+-----------+------+------+--------+
| FIRST_NAME | LAST_NAME | AGE   | SEX                    | INCOME |
                  +------------+-----------+------+------+--------+
|   Krishna   |   Sharma   |  19 | M    |                     2000 |
|   Mac   |   Mohan       | | 26 | M    |                     2000 |
Raj       |   Kandukuri | |   20 | M    |                     7000 |
Ramya        Ramapriya |     25 | F    |                     5000 |
                  +------------+-----------+------+------+--------+
                                          4 rows in set (0.00 sec)
```

ORDER BY clause using python

To retrieve contents of a table in specific order, invoke the **execute()** method on the cursor object and, pass the SELECT statement along with ORDER BY clause, as a parameter to it.

Example

In the following example we are creating a table with name and Employee, populating it, and retrieving its records back in the (ascending) order of their age, using the ORDER BY clause.

```
import mysql.connector


#establishing the connection
conn = mysql.connector.connect(user='root', password='password',
host='127.0.0.1', database='mydb')


#Creating a cursor object using the cursor() method
cursor = conn.cursor()
```

```
#Doping EMPLOYEE table if already exists.
cursor.execute("DROP TABLE IF EXISTS EMPLOYEE")
sql = '''CREATE TABLE EMPLOYEE(
         FIRST_NAME  CHAR(20) NOT NULL,
         LAST_NAME CHAR(20),
         AGE INT,
         SEX CHAR(1),
         INCOME FLOAT)'''
cursor.execute(sql)


#Populating the table
insert_stmt = "INSERT INTO EMPLOYEE (FIRST_NAME, LAST_NAME, AGE, SEX, INCOME)
VALUES (%s, %s, %s, %s, %s)"
data = [('Krishna', 'Sharma', 26, 'M', 2000), ('Raj', 'Kandukuri', 20, 'M',
7000),
       ('Ramya', 'Ramapriya', 29, 'F', 5000),('Mac', 'Mohan', 26, 'M', 2000)]
cursor.executemany(insert_stmt, data)
conn.commit()


#Retrieving specific records using the ORDER BY clause
cursor.execute("SELECT * from EMPLOYEE ORDER BY AGE")


print(cursor.fetchall())


#Closing the connection
conn.close()
```

Output

```
[('Raj', 'Kandukuri', 20, 'M', 7000.0), ('Krishna', 'Sharma', 26, 'M', 2000.0),
('Mac', 'Mohan', 26, 'M', 2000.0), ('Ramya', 'Ramapriya', 29, 'F', 5000.0)]
```

In the same way you can retrieve data from a table in descending order using the ORDER BY clause.

Example

```
import mysql.connector


#establishing the connection
```

```
conn = mysql.connector.connect(user='root', password='password',
host='127.0.0.1', database='mydb')


#Creating a cursor object using the cursor() method
cursor = conn.cursor()


#Retrieving specific records using the ORDERBY clause
cursor.execute("SELECT * from EMPLOYEE ORDER BY INCOME DESC")


print(cursor.fetchall())


#Closing the connection
conn.close()
```

Output

```
[('Raj', 'Kandukuri', 20, 'M', 7000.0), ('Ramya', 'Ramapriya', 29, 'F',
5000.0), ('Krishna', 'Sharma', 26, 'M', 2000.0), ('Mac', 'Mohan', 26, 'M',
2000.0)]
```

9. Python MySQL — Update Table

UPDATE Operation on any database updates one or more records, which are already available in the database. You can update the values of existing records in MySQL using the UPDATE statement. To update specific rows, you need to use the WHERE clause along with it.

Syntax

Following is the syntax of the UPDATE statement in MySQL:

```
UPDATE table_name
SET column1 = value1, column2 = value2...., columnN = valueN
WHERE [condition];
```

You can combine N number of conditions using the AND or the OR operators.

Example

Assume we have created a table in MySQL with name EMPLOYEES as:

```
mysql> CREATE TABLE EMPLOYEE(
FIRST_NAME CHAR(20) NOT NULL,
LAST_NAME CHAR(20),
AGE INT,
SEX CHAR(1),
INCOME FLOAT);
Query OK, 0 rows affected (0.36 sec)
```

And if we have inserted 4 records in to it using INSERT statements as:

```
mysql> INSERT INTO EMPLOYEE VALUES
     ('Krishna', 'Sharma', 19, 'M', 2000),
     ('Raj', 'Kandukuri', 20, 'M', 7000),
     ('Ramya', 'Ramapriya', 25, 'F', 5000),
     ('Mac', 'Mohan', 26, 'M', 2000);
```

Following MySQL statement increases the age of all male employees by one year:

```
mysql> UPDATE EMPLOYEE SET AGE = AGE + 1 WHERE SEX = 'M';
Query OK, 3 rows affected (0.06 sec)
```

```
Rows matched:3   Changed: 3  Warnings: 0
```

If you retrieve the contents of the table, you can see the updated values as:

```
                    mysql> select * from EMPLOYEE;
  +------------+-----------+------+------+--------+
| FIRST_NAME | LAST_NAME | AGE  | SEX  | INCOME  |
  +------------+-----------+------+------+--------+
| Krishna    |   Sharma | |  20 | M    |   2000  |
| Raj        Kandukuri | |   21 | M    |   7000  |
| Ramya      Ramapriya | |   25 | F    |   5000  |
| Mac        | Mohan         27 | M    |   2000  |
  +------------+-----------+------+------+--------+
                          4 rows in set (0.00 sec)
```

Updating the contents of a table using Python

To update the records in a table in MySQL using python:

import *mysql.connector* package. Create a connection object using the *mysql.connector.connect()* method, by passing the user name, password, host (optional default: localhost) and, database (optional) as parameters to it. Create a cursor object by invoking the ***cursor()*** method on the connection object created above. Then, execute the *UPDATE* statement by passing it as a parameter to the **execute()** method.

Example

The following example increases age of all the males by one year.

```
import mysql.connector


#establishing the connection
conn = mysql.connector.connect(user='root', password='password',
host='127.0.0.1', database='mydb')


#Creating a cursor object using the cursor() method
cursor = conn.cursor()
```

```
#Preparing the query to update the records
sql = '''UPDATE EMPLOYEE SET AGE = AGE + 1 WHERE SEX = 'M' '''

try:
   # Execute the SQL command
   cursor.execute(sql)
   # Commit your changes in the database
   conn.commit()
except:
   # Rollback in case there is any error
   conn.rollback()

#Retrieving data
sql = '''SELECT * from EMPLOYEE'''

#Executing the query
cursor.execute(sql)

#Displaying the result
print(cursor.fetchall())

#Closing the connection
conn.close()
```

Output

```
[('Krishna', 'Sharma', 22, 'M', 2000.0), ('Raj', 'Kandukuri', 23, 'M', 7000.0),
('Ramya', 'Ramapriya', 26, 'F', 5000.0)]
```

10. Python MySQL — Delete Data

To delete records from a MySQL table, you need to use the **DELETE FROM** statement. To remove specific records, you need to use WHERE clause along with it.

Syntax

Following is the syntax of the DELETE query in MYSQL:

```
DELETE FROM table_name [WHERE Clause]
```

Example

Assume we have created a table in MySQL with name EMPLOYEES as:

```
mysql> CREATE TABLE EMPLOYEE(
FIRST_NAME CHAR(20) NOT NULL,
LAST_NAME CHAR(20),
AGE INT,
SEX CHAR(1),
INCOME FLOAT);
Query OK, 0 rows affected (0.36 sec)
```

And if we have inserted 4 records in to it using INSERT statements as:

```
mysql> INSERT INTO EMPLOYEE VALUES
     ('Krishna', 'Sharma', 19, 'M', 2000),
     ('Raj', 'Kandukuri', 20, 'M', 7000),
     ('Ramya', 'Ramapriya', 25, 'F', 5000),
     ('Mac', 'Mohan', 26, 'M', 2000);
```

Following MySQL statement deletes the record of the employee with *FIRST_NAME* "*Mac*".

```
mysql> DELETE FROM EMPLOYEE WHERE FIRST_NAME = 'Mac';
Query OK, 1 row affected (0.12 sec)
```

If you retrieve the contents of the table, you can see only 3 records since we have deleted one.

```
mysql> select * from EMPLOYEE;
+------------+-----------+------+------+--------+


| FIRST_NAME | LAST_NAME | AGE  |SEX   |INCOME |
```

```
+------------+----------+------+------+--------+ |
Krishna | Sharma | 20 | M | 2000| | Raj | Kandukuri
| 21 | M | 7000| | Ramya | Ramapriya | 25 | F |
5000| +------------+----------+------+-- ----+----
----+ 3 rows in set (0.00 sec)
```

If you execute the DELETE statement without the WHERE clause all the records from the specified table will be deleted.

```
mysql> DELETE FROM EMPLOYEE;
Query OK, 3 rows affected (0.09 sec)
```

If you retrieve the contents of the table, you will get an empty set as shown below:

```
mysql> select * from EMPLOYEE;
Empty set (0.00 sec)
```

Removing records of a table using python

DELETE operation is required when you want to delete some records from your database.

To delete the records in a table:

import *mysql.connector* package.

Create a connection object using the *mysql.connector.connect()* method, by passing the user name, password, host (optional default: localhost) and, database (optional) as parameters to it.

Create a cursor object by invoking the *cursor()* method on the connection object created above.

Then, execute the *DELETE* statement by passing it as a parameter to the **execute()** method.

Example

Following program deletes all the records from EMPLOYEE whose AGE is more than 20:

```
import mysql.connector


#establishing the connection
conn = mysql.connector.connect(user='root', password='password',
host='127.0.0.1', database='mydb')


#Creating a cursor object using the cursor() method
```

```
cursor = conn.cursor()

#Retrieving single row
print("Contents of the table: ")
cursor.execute("SELECT * from EMPLOYEE")
print(cursor.fetchall())

#Preparing the query to delete records
sql = "DELETE FROM EMPLOYEE WHERE AGE > '%d'" % (25)
try:
   # Execute the SQL command
   cursor.execute(sql)
   # Commit your changes in the database
   conn.commit()
except:
   # Roll back in case there is any error
   conn.rollback()

#Retrieving data
print("Contents of the table after delete operation ")
cursor.execute("SELECT * from EMPLOYEE")
print(cursor.fetchall())

#Closing the connection
conn.close()
```

Output

```
Contents of the table:
[('Krishna', 'Sharma', 22, 'M', 2000.0), ('Raj', 'Kandukuri', 23, 'M', 7000.0),
('Ramya', 'Ramapriya', 26, 'F', 5000.0), ('Mac', 'Mohan', 20, 'M', 2000.0),
('Ramya', 'Rama priya', 27, 'F', 9000.0)]
Contents of the table after delete operation:
[('Krishna', 'Sharma', 22, 'M', 2000.0), ('Raj', 'Kandukuri', 23, 'M', 7000.0),
('Mac', 'Mohan', 20, 'M', 2000.0)]
```

11. Python MySQL — Drop Table

You can remove an entire table using the **DROP TABLE** statement. You just need to specify the name of the table you need to delete.

Syntax

Following is the syntax of the DROP TABLE statement in MySQL:

```
DROP TABLE table_name;
```

Example

Before deleting a table get the list of tables using the SHOW TABLES statement as follows:

```
mysql> SHOW TABLES; +----
-------------+           |
Tables_in_mydb | +-------
----------+ | contact | |
cricketers_data    |     |
employee  |  | sample  |  |
tutorials | +------------
-----+   5   rows   in   set
(0.00 sec)
```

Following statement removes the table named sample from the database completely:

```
mysql> DROP TABLE sample;
Query OK, 0 rows affected (0.29 sec)
```

Since we have deleted the table named sample from MySQL, if you get the list of tables again you will not find the table name sample in it.

```
mysql> SHOW TABLES;
+-----------------+
| Tables_in_mydb|
+-----------------+
| contact| |
cricketers_data|
```

```
| employee | | tutorials |
+-----------------+ 4 rows
in set (0.00 sec)
```

Removing a table using python

You can drop a table whenever you need to, using the DROP statement of MYSQL, but you need to be very careful while deleting any existing table because the data lost will not be recovered after deleting a table.

To drop a table from a MYSQL database using python invoke the **execute()** method on the cursor object and pass the drop statement as a parameter to it.

Example

Following table drops a table named EMPLOYEE from the database.

```
import mysql.connector


#establishing the connection
conn = mysql.connector.connect(user='root', password='password',
host='127.0.0.1', database='mydb')


#Creating a cursor object using the cursor() method
cursor = conn.cursor()


#Retrieving the list of tables
print("List of tables in the database: ")
cursor.execute("SHOW Tables")
print(cursor.fetchall())


#Doping EMPLOYEE table if already exists
cursor.execute("DROP TABLE EMPLOYEE")
print("Table dropped... ")


#Retrieving the list of tables
print("List of tables after dropping the EMPLOYEE table: ")
cursor.execute("SHOW Tables")
print(cursor.fetchall())
```

```
#Closing the connection
conn.close()
```

Output

```
List of tables in the database: [('employee',),
('employeedata',), ('sample',), ('tutorials',)] Table dropped...
List of tables after dropping the EMPLOYEE table:
[('employeedata',), ('sample',), ('tutorials',)]
```

Drop table only if exists

If you try to drop a table which does not exist in the database, an error occurs as:

```
mysql.connector.errors.ProgrammingError: 1051 (42S02): Unknown table
'mydb.employee'
```

You can prevent this error by verifying whether the table exists before deleting, by adding the IF EXISTS to the DELETE statement.

```
import mysql.connector


#establishing the connection
conn = mysql.connector.connect(user='root', password='password',
host='127.0.0.1', database='mydb')


#Creating a cursor object using the cursor() method
cursor = conn.cursor()


#Retrieving the list of tables
print("List of tables in the database: ")
cursor.execute("SHOW Tables")
print(cursor.fetchall())


#Doping EMPLOYEE table if already exists
cursor.execute("DROP TABLE IF EXISTS EMPLOYEE")
print("Table dropped... ")


#Retrieving the list of tables
print("List of tables after dropping the EMPLOYEE table: ")
```

```
cursor.execute("SHOW Tables")
print(cursor.fetchall())

#Closing the connection
conn.close()
```

Output

```
List of tables in the database:
[('employeedata',), ('sample',), ('tutorials',)]
Table dropped...
List of tables after dropping the EMPLOYEE table:
[('employeedata',), ('sample',),
 ('tutorials',)]
```

12. Python MySQL — Limit

While fetching records if you want to limit them by a particular number, you can do so, using the LIMIT clause of MYSQL.

Example

Assume we have created a table in MySQL with name EMPLOYEES as:

```
mysql> CREATE TABLE EMPLOYEE(
FIRST_NAME CHAR(20) NOT NULL,
LAST_NAME CHAR(20),
AGE INT,
SEX CHAR(1),
INCOME FLOAT);
Query OK, 0 rows affected (0.36 sec)
```

And if we have inserted 4 records in to it using INSERT statements as:

```
mysql> INSERT INTO EMPLOYEE VALUES
     ('Krishna', 'Sharma', 19, 'M', 2000),
     ('Raj', 'Kandukuri', 20, 'M', 7000),
     ('Ramya', 'Ramapriya', 25, 'F', 5000),
     ('Mac', 'Mohan', 26, 'M', 2000);
```

Following SQL statement retrieves first two records of the Employee table using the LIMIT clause.

```
                    SELECT * FROM EMPLOYEE LIMIT 2;
  +------------+-----------+------+------+--------+
| FIRST_NAME | LAST_NAME | AGE   | SEX  | INCOME   |
  +------------+-----------+------+------+--------+
| Krishna    | Sharma    |   19 | M    |   2000   |
| Raj        | Kandukuri |   20 | M    |   7000   |
  +------------+-----------+------+------+--------+
                          2 rows in set (0.00 sec)
```

Limit clause using python

If you invoke the **execute()** method on the cursor object by passing the SELECT query along with the LIMIT clause, you can retrieve required number of records.

Example

Following python example creates and populates a table with name EMPLOYEE and, using the LIMIT clause it fetches the first two records of it.

```
import mysql.connector


#establishing the connection
conn = mysql.connector.connect(user='root', password='password',
host='127.0.0.1', database='mydb')


#Creating a cursor object using the cursor() method
cursor = conn.cursor()


#Retrieving single row
sql = '''SELECT * from EMPLOYEE LIMIT 2'''


#Executing the query
cursor.execute(sql)


#Fetching the data
result = cursor.fetchall();
print(result)


#Closing the connection
conn.close()
```

Output

```
[('Krishna', 'Sharma', 26, 'M', 2000.0), ('Raj', 'Kandukuri', 20, 'M', 7000.0)]
```

LIMIT with OFFSET

If you need to limit the records starting from nth record (not 1st), you can do so, using OFFSET along with LIMIT.

```
import mysql.connector
```

```
#establishing the connection
conn = mysql.connector.connect(user='root', password='password',
host='127.0.0.1', database='mydb')


#Creating a cursor object using the cursor() method
cursor = conn.cursor()


#Retrieving single row
sql = '''SELECT * from EMPLOYEE LIMIT 2 OFFSET 2'''


#Executing the query
cursor.execute(sql)


#Fetching the data
result = cursor.fetchall();
print(result)


#Closing the connection
conn.close()
```

Output

```
[('Ramya', 'Ramapriya', 29, 'F', 5000.0), ('Mac', 'Mohan', 26, 'M', 2000.0)]
```

13. Python MySQL — Join

When you have divided the data in two tables you can fetch combined records from these two tables using Joins.

Example

Suppose we have created a table with name EMPLOYEE and populated data into it as shown below:

```
mysql> CREATE TABLE EMPLOYEE(
         FIRST_NAME CHAR(20) NOT NULL,
         LAST_NAME CHAR(20),
         AGE INT,
         SEX CHAR(1),
         INCOME FLOAT,
         CONTACT INT
);
Query OK, 0 rows affected (0.36 sec)
INSERT INTO Employee VALUES ('Ramya', 'Rama Priya', 27, 'F', 9000, 101),
('Vinay', 'Bhattacharya', 20, 'M', 6000, 102), ('Sharukh', 'Sheik', 25, 'M',
8300, 103), ('Sarmista', 'Sharma', 26, 'F', 10000, 104), ('Trupthi', 'Mishra',
24, 'F', 6000, 105);
Query OK, 5 rows affected (0.08 sec)

Records: 5  Duplicates: 0  Warnings: 0
```

Then, if we have created another table and populated it as:

```
CREATE TABLE CONTACT(
        ID INT NOT NULL,
        EMAIL CHAR(20) NOT NULL,
        PHONE LONG,
        CITY CHAR(20));
Query OK, 0 rows affected (0.49 sec)
```

```
INSERT INTO CONTACT (ID, EMAIL, CITY) VALUES (101, 'Krishna@mymail.com',
'Hyderabad'), (102, 'Raja@mymail.com', 'Vishakhapatnam'), (103,
'Krishna@mymail.com', 'Pune'), (104, 'Raja@mymail.com', 'Mumbai');
```

```
 Query OK, 4 rows affected (0.10 sec)
Records: 4  Duplicates: 0   Warnings:0
```

Following statement retrieves data combining the values in these two tables:

```
mysql> SELECT * from EMPLOYEE INNER JOIN CONTACT ON EMPLOYEE.CONTACT =
CONTACT.ID;
+------------+--------------+------+------+--------+---------+-----+-----------
---------+-------+----------------+
| FIRST_NAME | LAST_NAME    | AGE | SEX | INCOME | CONTACT | ID | EMAIL
| PHONE | CITY           |
+------------+--------------+------+------+--------+---------+-----+-----------
---------+-------+----------------+
| Ramya      | Rama Priya   |   27 | F    |   9000 |     101 | 101 |
Krishna@mymail.com | NULL   | Hyderabad      |
| Vinay      | Bhattacharya |   20 | M    |   6000 |     102 | 102 |
Raja@mymail.com    | NULL   | Vishakhapatnam |
| Sharukh    | Sheik        |   25 | M    |   8300 |     103 | 103 |
Krishna@mymail.com | NULL   | Pune           |
| Sarmista   | Sharma       |   26 | F    | 10000 |     104 | 104 |
Raja@mymail.com    | NULL   | Mumbai         |
+------------+--------------+------+------+--------+---------+-----+-----------
---------+-------+----------------+
4 rows in set (0.00 sec)
```

MYSQL JOIN using python

Following example retrieves data from the above two tables combined by contact column of the EMPLOYEE table and ID column of the CONTACT table.

```
import mysql.connector


#establishing the connection
conn = mysql.connector.connect(user='root', password='password',
host='127.0.0.1', database='mydb')


#Creating a cursor object using the cursor() method
cursor = conn.cursor()


#Retrieving single row
sql = '''SELECT * from EMPLOYEE INNER JOIN CONTACT ON EMPLOYEE.CONTACT =
CONTACT.ID'''
```

```
#Executing the query
cursor.execute(sql)

#Fetching 1st row from the table
result = cursor.fetchall();

print(result)

#Closing the connection
conn.close()
```

Output

```
[('Krishna', 'Sharma', 26, 'M', 2000, 101, 101, 'Krishna@mymail.com',
9848022338, 'Hyderabad'), ('Raj', 'Kandukuri', 20, 'M', 7000, 102, 102,
'Raja@mymail.com', 9848022339, 'Vishakhapatnam'), ('Ramya', 'Ramapriya', 29,
'F', 5000, 103, 103, 'Krishna@mymail.com', 9848022337, 'Pune'), ('Mac',
'Mohan', 26, 'M', 2000, 104, 104, 'Raja@mymail.com', 9848022330, 'Mumbai')]
```

14. Python MySQL — Cursor Object

The MySQLCursor of mysql-connector-python (and similar libraries) is used to execute statements to communicate with the MySQL database.

Using the methods of it you can execute SQL statements, fetch data from the result sets, call procedures.
You can create **Cursor** object using the cursor() method of the Connection object/class.

Example

```
import mysql.connector


#establishing the connection
conn = mysql.connector.connect(user='root', password='password',
host='127.0.0.1', database='mydb')


#Creating a cursor object using the cursor() method
cursor = conn.cursor()
```

Methods

Following are the various methods provided by the Cursor class/object.

Method	Description
callproc()	This method is used to call existing procedures MySQL database.
close()	This method is used to close the current cursor object.
Info()	This method gives information about the last query.
executemany()	This method accepts a list series of parameters list. Prepares an MySQL query and executes it with all the parameters.
execute()	This method accepts a MySQL query as a parameter and executes the given query.
fetchall()	This method retrieves all the rows in the result set of a query and returns them as list of tuples. (If we execute this after retrieving few rows it returns the remaining ones)

fetchone()	This method fetches the next row in the result of a query and returns it as a tuple.
fetchmany()	This method is similar to the fetchone() but, it retrieves the next set of rows in the result set of a query, instead of a single row.
fetchwarnings()	This method returns the warnings generated by the last executed query.

Properties

Following are the properties of the Cursor class:

Property	Description
column_names	This is a read only property which returns the list containing the column names of a result-set.
description	This is a read only property which returns the list containing the description of columns in a result-set.
l astrowi d	This is a read only property, if there are any auto-incremented columns in the table, this returns the value generated for that column in the last INSERT or, UPDATE operation.
rowcount	This returns the number of rows returned/updated in case of SELECT and UPDATE operations.
statement	This property returns the last executed statement.