**MySQL in Python Tutorial: Getting Started**

[](https://www.datacamp.com/profile/hafeezulkareem)

[**Hafeezul Kareem Shaik**](https://www.datacamp.com/profile/hafeezulkareem) • December 9, 2019

In this tutorial, you'll learn how to install MySQL, create databases, and create tables, along with much more.

**Prerequisites**

* Python Data Types, Control Structures, Loops, etc., If you're new to **Python** learn **Datacamp's** free [**Intro to Python for Data Science**](https://www.datacamp.com/courses/intro-to-python-for-data-science) course.
* SQL Basics. If you don't know SQL, take the **Datacamp's** free [**SQL**](https://www.datacamp.com/courses/intro-to-sql-for-data-science) course.

**Table Of Contents**

1. [**Installing MySQL**](https://www.datacamp.com/tutorial/mysql-python#IMySQL)
2. [**Connecting And Creating**](https://www.datacamp.com/tutorial/mysql-python#CAC)  
   2.1. [**Creating Databases**](https://www.datacamp.com/tutorial/mysql-python#CD)  
   2.2. [**Creating Tables**](https://www.datacamp.com/tutorial/mysql-python#CT)  
   2.3. [**Primary Key**](https://www.datacamp.com/tutorial/mysql-python#PK)
3. [**Inserting Data**](https://www.datacamp.com/tutorial/mysql-python#ID)
4. [**Select Data**](https://www.datacamp.com/tutorial/mysql-python#SD)
5. [**Where**](https://www.datacamp.com/tutorial/mysql-python#W)
6. [**Order By**](https://www.datacamp.com/tutorial/mysql-python#OB)
7. [**Delete**](https://www.datacamp.com/tutorial/mysql-python#D)
8. [**Update**](https://www.datacamp.com/tutorial/mysql-python#U)

**Note:**

* This tutorial is not meant to teach you complete **SQL** syntax or language.
* This tutorial will teach you how you can work with **MySQL** in **Python**.

**Procedure To Follow In Python To Work With MySQL**

1. Connect to the database.
2. Create an object for your database.
3. Execute the **SQL** query.
4. Fetch records from the result.
5. Informing the Database if you make any changes in the table.

**1. Installing MySQL**

**MySQL** is one of the most popular databases.

Download and install **MySQL** from the **MySQL's** official [**website**](https://www.mysql.com/downloads/). You need to install the **MySQL** server to follow this tutorial.

Next, you have to install **mysql.connector** for **Python**. We need **mysql.connector** to connect **Python Script** to the **MySQL** database. Download the **mysql.connector** from [**here**](https://dev.mysql.com/downloads/connector/python/) and install it on your computer.

Now, check whether you have installed the **mysql.connector** correctly or not using the following code.

import mysql.connector

If the above code runs without any errors, then you have successfully installed **mysql.connector**, and it is ready to use.

**2. Connecting And Creating**

Now, we will connect to the database using **username** and **password** of **MySQL**. If you don't remember your **username** or **password**, create a new **user** with a password.

To create a new **user** refer to **MySQL** official [**documentation**](https://dev.mysql.com/doc/refman/8.0/en/create-user.html).

Now, connect to the database using your **username** and **password**.

## Connecting to the database

## importing 'mysql.connector' as mysql for convenient

import mysql.connector as mysql

## connecting to the database using 'connect()' method

## it takes 3 required parameters 'host', 'user', 'passwd'

db = mysql.connect(

host = "localhost",

user = "root",

passwd = "dbms"

)

print(db) # it will print a connection object if everything is fine

<mysql.connector.connection\_cext.CMySQLConnection object at 0x0000020C26A84C50>

That's it now you have connected to the **MySQL** database.

**2.1. Creating Databases**

Now, we will create a database with the name **datacamp**.

To create a database in **MySQL**, we use **CREATE DATABASE database\_name** statement.

import mysql.connector as mysql

db = mysql.connect(

host = "localhost",

user = "root",

passwd = "dbms"

)

## creating an instance of 'cursor' class which is used to execute the 'SQL' statements in 'Python'

cursor = db.cursor()

## creating a databse called 'datacamp'

## 'execute()' method is used to compile a 'SQL' statement

## below statement is used to create tha 'datacamp' database

cursor.execute("CREATE DATABASE datacamp")

If the database already exists you will get an error. Make sure that the database does not exist.

See all the databases present in **MySQL** using the following code.

To see all the databases we use **SHOW DATABASES** statement.

import mysql.connector as mysql

db = mysql.connect(

host = "localhost",

user = "root",

passwd = "dbms"

)

cursor = db.cursor()

## executing the statement using 'execute()' method

cursor.execute("SHOW DATABASES")

## 'fetchall()' method fetches all the rows from the last executed statement

databases = cursor.fetchall() ## it returns a list of all databases present

## printing the list of databases

print(databases)

## showing one by one database

for database in databases:

print(database)

[('datacamp',), ('information\_schema',), ('mysql',), ('performance\_schema',), ('sakila',), ('sys',), ('world',)]

('datacamp',)

('information\_schema',)

('mysql',)

('performance\_schema',)

('sakila',)

('sys',)

('world',)

**2.2. Creating Tables**

Creating tables in the database to store the information. Before creating tables, we have to select a database first.

Run the following code, to select **datacamp** database which we have created a minute before.

import mysql.connector as mysql

db = mysql.connect(

host = "localhost",

user = "root",

passwd = "dbms",

database = "datacamp"

)

The above code will execute with no errors if the database exists. Now, you have connected to the database called **datacamp**.

Use the **CREATE TABLE table\_name** to create a table in the selected database.

import mysql.connector as mysql

db = mysql.connect(

host = "localhost",

user = "root",

passwd = "dbms",

database = "datacamp"

)

cursor = db.cursor()

## creating a table called 'users' in the 'datacamp' database

cursor.execute("CREATE TABLE users (name VARCHAR(255), user\_name VARCHAR(255))")

You have successfully created the table **users** in the **datacamp** database. See all the tables present in the database using the **SHOW TABLES** statement.

import mysql.connector as mysql

db = mysql.connect(

host = "localhost",

user = "root",

passwd = "dbms",

database = "datacamp"

)

cursor = db.cursor()

## getting all the tables which are present in 'datacamp' database

cursor.execute("SHOW TABLES")

tables = cursor.fetchall() ## it returns list of tables present in the database

## showing all the tables one by one

for table in tables:

print(table)

('users',)

**2.3. Primary Key**

**Primary Key:-** It is a unique value in the table. It helps to find each row uniquely in the table.

To create a **Primary Key**, we use the **PRIMARY KEY** statement while creating the table.

The statement **INT AUTO\_INCREMENT PRIMARY KEY** is used to identify each row uniquely with a number starting from 1.

Let's see how to create **Primary Key** for a table.

import mysql.connector as mysql

db = mysql.connect(

host = "localhost",

user = "root",

passwd = "dbms",

database = "datacamp"

)

cursor = db.cursor()

## first we have to 'drop' the table which has already created to create it again with the 'PRIMARY KEY'

## 'DROP TABLE table\_name' statement will drop the table from a database

cursor.execute("DROP TABLE users")

## creating the 'users' table again with the 'PRIMARY KEY'

cursor.execute("CREATE TABLE users (id INT(11) NOT NULL AUTO\_INCREMENT PRIMARY KEY, name VARCHAR(255), user\_name VARCHAR(255))")

To see the table run the following code.

import mysql.connector as mysql

db = mysql.connect(

host = "localhost",

user = "root",

passwd = "dbms",

database = "datacamp"

)

cursor = db.cursor()

## 'DESC table\_name' is used to get all columns information

cursor.execute("DESC users")

## it will print all the columns as 'tuples' in a list

print(cursor.fetchall())

[('id', 'int(11)', 'NO', 'PRI', None, 'auto\_increment'), ('name', 'varchar(255)', 'YES', '', None, ''), ('user\_name', 'varchar(255)', 'YES', '', None, '')]

**Dropping Primary Key**

We use **ALTER TABLE table\_name DROP column\_name** statement to drop the column with **Primary Key**.

import mysql.connector as mysql

db = mysql.connect(

host = "localhost",

user = "root",

passwd = "dbms",

database = "datacamp"

)

cursor = db.cursor()

## dropping the 'id' column

cursor.execute("ALTER TABLE users DROP id")

cursor.execute("DESC users")

print(cursor.fetchall())

[('name', 'varchar(255)', 'YES', '', None, ''), ('user\_name', 'varchar(255)', 'YES', '', None, '')]

Now, we have dropped the **Primary Key** column. Let's see how we can add a column with **Primary Key** to the existing table.

**Adding Primary Key**

Adding **Primary Key** to the existing table. We use **ALTER TABLE table\_name ADD PRIMARY KEY(column\_name)** statement to add a **Primary Key** to a table.

import mysql.connector as mysql

db = mysql.connect(

host = "localhost",

user = "root",

passwd = "dbms",

database = "datacamp"

)

cursor = db.cursor()

## adding 'id' column to the 'users' table

## 'FIRST' keyword in the statement will add a column in the starting of the table

cursor.execute("ALTER TABLE users ADD COLUMN id INT(11) NOT NULL AUTO\_INCREMENT PRIMARY KEY FIRST")

cursor.execute("DESC users")

print(cursor.fetchall())

[('id', 'int(11)', 'NO', 'PRI', None, 'auto\_increment'), ('name', 'varchar(255)', 'YES', '', None, ''), ('user\_name', 'varchar(255)', 'YES', '', None, '')]

We have added the column **id** to the **users** table.

**3. Inserting Data**

Inserting data into table to store it. Use **INSERT INTO table\_name (column\_names) VALUES (data)** statement to insert into the table.

**Inserting A Single Row**

Let's see how to insert a single row into the table.

import mysql.connector as mysql

db = mysql.connect(

host = "localhost",

user = "root",

passwd = "dbms",

database = "datacamp"

)

cursor = db.cursor()

## defining the Query

query = "INSERT INTO users (name, user\_name) VALUES (%s, %s)"

## storing values in a variable

values = ("Hafeez", "hafeez")

## executing the query with values

cursor.execute(query, values)

## to make final output we have to run the 'commit()' method of the database object

db.commit()

print(cursor.rowcount, "record inserted")

1 record inserted

The above code will insert one row into the **users** table.

**Inserting Multiple Rows**

Let's see how to insert multiple rows into the table.

To insert multiple rows into the table, we use the **executemany()** method. It takes a list of **tuples** containing the data as a second parameter and a **query** as the first argument.

import mysql.connector as mysql

db = mysql.connect(

host = "localhost",

user = "root",

passwd = "dbms",

database = "datacamp"

)

cursor = db.cursor()

## defining the Query

query = "INSERT INTO users (name, user\_name) VALUES (%s, %s)"

## storing values in a variable

values = [

("Peter", "peter"),

("Amy", "amy"),

("Michael", "michael"),

("Hennah", "hennah")

]

## executing the query with values

cursor.executemany(query, values)

## to make final output we have to run the 'commit()' method of the database object

db.commit()

print(cursor.rowcount, "records inserted")

4 records inserted

The above code inserted four records into the **users** table.

**4. Select Data**

To retrieve the data from a table we use, **SELECT column\_names FROM table\_name** statement.

**Getting All Records From Table**

To get all records from a table, we use **\*** in place of column names. Let's get all the data from the **users** table which we inserted before.

import mysql.connector as mysql

db = mysql.connect(

host = "localhost",

user = "root",

passwd = "dbms",

database = "datacamp"

)

cursor = db.cursor()

## defining the Query

query = "SELECT \* FROM users"

## getting records from the table

cursor.execute(query)

## fetching all records from the 'cursor' object

records = cursor.fetchall()

## Showing the data

for record in records:

print(record)

(1, 'Hafeez', 'hafeez')

(2, 'Peter', 'peter')

(3, 'Amy', 'amy')

(4, 'Michael', 'michael')

(5, 'Hennah', 'hennah')

**Getting Some Columns**

To select some columns from the table mention column name after the **SELECT** in the statement. Let's retrieve the username column from the **users** table.

import mysql.connector as mysql

db = mysql.connect(

host = "localhost",

user = "root",

passwd = "dbms",

database = "datacamp"

)

cursor = db.cursor()

## defining the Query

query = "SELECT user\_name FROM users"

## getting 'user\_name' column from the table

cursor.execute(query)

## fetching all usernames from the 'cursor' object

usernames = cursor.fetchall()

## Showing the data

for username in usernames:

print(username)

('hafeez',)

('peter',)

('amy',)

('michael',)

('hennah',)

You can also retrieve more than one column at a time as follows.

import mysql.connector as mysql

db = mysql.connect(

host = "localhost",

user = "root",

passwd = "dbms",

database = "datacamp"

)

cursor = db.cursor()

## defining the Query

query = "SELECT name, user\_name FROM users"

## getting 'name', 'user\_name' columns from the table

cursor.execute(query)

## fetching all records from the 'cursor' object

data = cursor.fetchall()

## Showing the data

for pair in data:

print(pair)

('Hafeez', 'hafeez')

('Peter', 'peter')

('Amy', 'amy')

('Michael', 'michael')

('Hennah', 'hennah')

**5. Where**

**WHERE** is used to select data on some condition. Now, we will select a record with id 5.

**SELECT column\_name FROM table\_name WHERE condition** statement will be used to retrieve the data on some condition.

import mysql.connector as mysql

db = mysql.connect(

host = "localhost",

user = "root",

passwd = "dbms",

database = "datacamp"

)

cursor = db.cursor()

## defining the Query

query = "SELECT \* FROM users WHERE id = 5"

## getting records from the table

cursor.execute(query)

## fetching all records from the 'cursor' object

records = cursor.fetchall()

## Showing the data

for record in records:

print(record)

(5, 'Hennah', 'hennah')

You can specify any condition based on your data.

**6. Order By**

Use the **ORDER BY** to sort the result in ascending or descending order. It sorts the result in ascending order by default, to sort the result in descending order use the keyword **DESC**.

**SELECT column\_names FROM table\_name ORDER BY column\_name** statement will be used to sort the result in **ascending order** by a **column**.

**SELECT column\_names FROM table\_name ORDER BY column\_name DESC** statement will be used to sort the result in **descending order** by a **column**.

Sorting the data in **ascending order** using the **name** column. Let's see the code.

import mysql.connector as mysql

db = mysql.connect(

host = "localhost",

user = "root",

passwd = "dbms",

database = "datacamp"

)

cursor = db.cursor()

## defining the Query

query = "SELECT \* FROM users ORDER BY name"

## getting records from the table

cursor.execute(query)

## fetching all records from the 'cursor' object

records = cursor.fetchall()

## Showing the data

for record in records:

print(record)

(3, 'Amy', 'amy')

(1, 'Hafeez', 'hafeez')

(5, 'Hennah', 'hennah')

(4, 'Michael', 'michael')

(2, 'Peter', 'peter')

Sorting the data in **descending order** by **name** column. Let's see the code.

import mysql.connector as mysql

db = mysql.connect(

host = "localhost",

user = "root",

passwd = "dbms",

database = "datacamp"

)

cursor = db.cursor()

## defining the Query

query = "SELECT \* FROM users ORDER BY name DESC"

## getting records from the table

cursor.execute(query)

## fetching all records from the 'cursor' object

records = cursor.fetchall()

## Showing the data

for record in records:

print(record)

(2, 'Peter', 'peter')

(4, 'Michael', 'michael')

(5, 'Hennah', 'hennah')

(1, 'Hafeez', 'hafeez')

(3, 'Amy', 'amy')

**7. Delete**

**DELETE** keyword is used to delete the records from the table.

**DELETE FROM table\_name WHERE condition** statement is used to delete records. If you don't specify the condition, then all of the records will be deleted.

Let's delete a record from the **users** table with id 5.

import mysql.connector as mysql

db = mysql.connect(

host = "localhost",

user = "root",

passwd = "dbms",

database = "datacamp"

)

cursor = db.cursor()

## defining the Query

query = "DELETE FROM users WHERE id = 5"

## executing the query

cursor.execute(query)

## final step to tell the database that we have changed the table data

db.commit()

Checking Whether it is deleted or not by querying all the records from the table.

import mysql.connector as mysql

db = mysql.connect(

host = "localhost",

user = "root",

passwd = "dbms",

database = "datacamp"

)

cursor = db.cursor()

## defining the Query

query = "SELECT \* FROM users"

## getting records from the table

cursor.execute(query)

## fetching all records from the 'cursor' object

records = cursor.fetchall()

## Showing the data

for record in records:

print(record)

(1, 'Hafeez', 'hafeez')

(2, 'Peter', 'peter')

(3, 'Amy', 'amy')

(4, 'Michael', 'michael')

5th record is deleted.

**8. Update**

**UPDATE** keyword is used to update the data of a record or records.

**UPDATE table\_name SET column\_name = new\_value WHERE condition** statement is used to update the value of a specific row.

Let's update the name of the 1st record from **Hafeez** to **Kareem**.

import mysql.connector as mysql

db = mysql.connect(

host = "localhost",

user = "root",

passwd = "dbms",

database = "datacamp"

)

cursor = db.cursor()

## defining the Query

query = "UPDATE users SET name = 'Kareem' WHERE id = 1"

## executing the query

cursor.execute(query)

## final step to tell the database that we have changed the table data

db.commit()

Checking the data whether it's updated or not by retrieving all records from the data.

import mysql.connector as mysql

db = mysql.connect(

host = "localhost",

user = "root",

passwd = "dbms",

database = "datacamp"

)

cursor = db.cursor()

## defining the Query

query = "SELECT \* FROM users"

## getting records from the table

cursor.execute(query)

## fetching all records from the 'cursor' object

records = cursor.fetchall()

## Showing the data

for record in records:

print(record)

(1, 'Kareem', 'hafeez')

(2, 'Peter', 'peter')

(3, 'Amy', 'amy')

(4, 'Michael', 'michael')

See, the name of the 1st record was changed.

I didn't talk about all the methods of the **database** object. You can inspect all the methods using **dir()** method.