

▼ Python MySQL

- Python can be used in database applications.
- One of the most popular databases is MySQL

Install MySQL Driver

- Python needs a MySQL driver to access the MySQL database.
- In this tutorial we will use the driver "MySQL Connector".
- We recommend that you use PIP to install "MySQL Connector".
- PIP is most likely already installed in your Python environment.

Navigate your command line to the location of PIP, and type the following

```
C:\Users\Your Name\AppData\Local\Programs\Python\Python36-32\Scripts>python -m pip  
install mysql-connector-python
```

Test MySQL Connector

To test if the installation was successful, or if you already have "MySQL Connector" installed, create a Python page with the following content

page name : demo_mysql_test.py

```
import mysql.connector
```

If the above code was executed with no errors, "MySQL Connector" is installed and ready to be used.

▼ Create Connection

- Start by creating a connection to the database.
- Use the username and password from your MySQL database

```
import mysql.connector  
  
mydb = mysql.connector.connect(  
    host="localhost",  
    user="yourusername",  
    password="yourpassword"  
)
```

```
print(mydb)
```

▼ Creating a Database

To create a database in MySQL, use the "CREATE DATABASE" statement

```
# create a database named "mydatabase":

import mysql.connector

mydb = mysql.connector.connect(
    host="localhost",
    user="yourusername",
    password="yourpassword"
)

mycursor = mydb.cursor()

mycursor.execute("CREATE DATABASE mydatabase")
```

▼ Check if Database Exists

You can check if a database exist by listing all databases in your system by using the "SHOW DATABASES" statement

```
# Return a list of your system's databases

import mysql.connector

mydb = mysql.connector.connect(
    host="localhost",
    user="yourusername",
    password="yourpassword"
)

mycursor = mydb.cursor()

mycursor.execute("SHOW DATABASES")

for x in mycursor:
    print(x)
```

```
# Or Try connecting to the database "mydatabase"

import mysql.connector
```

```
mydb = mysql.connector.connect(
    host="localhost",
    user="yourusername",
    password="yourpassword",
    database="mydatabase"
)

# If the database does not exist, you will get an error.
```

▼ Creating a Table

- To create a table in MySQL, use the "CREATE TABLE" statement.
- Make sure you define the name of the database when you create the connection

```
# Create a table named "customers":

import mysql.connector

mydb = mysql.connector.connect(
    host="localhost",
    user="yourusername",
    password="yourpassword",
    database="mydatabase"
)

mycursor = mydb.cursor()

mycursor.execute("CREATE TABLE customers (name VARCHAR(255), address VARCHAR(255))")
```

▼ Check if Table Exists

You can check if a table exist by listing all tables in your database with the "SHOW TABLES" statement

```
# Return a list of your system's databases:

import mysql.connector

mydb = mysql.connector.connect(
    host="localhost",
    user="yourusername",
    password="yourpassword",
    database="mydatabase"
)

mycursor = mydb.cursor()
```

```
mycursor.execute("SHOW TABLES")

for x in mycursor:
    print(x)
```

▼ 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

```
# Create primary key when creating the table:

import mysql.connector

mydb = mysql.connector.connect(
    host="localhost",
    user="yourusername",
    password="yourpassword",
    database="mydatabase"
)

mycursor = mydb.cursor()

mycursor.execute("CREATE TABLE customers (id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(255), address VARCHAR(255), city VARCHAR(255))")
```

If the table already exists, use the ALTER TABLE keyword

```
# Create primary key on an existing table:

import mysql.connector

mydb = mysql.connector.connect(
    host="localhost",
    user="yourusername",
    password="yourpassword",
    database="mydatabase"
)

mycursor = mydb.cursor()

mycursor.execute("ALTER TABLE customers ADD COLUMN id INT AUTO_INCREMENT PRIMARY KEY")
```

▼ Insert Into Table

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

```
# Insert a record in the "customers" table:

import mysql.connector

mydb = mysql.connector.connect(
    host="localhost",
    user="yourusername",
    password="yourpassword",
    database="mydatabase"
)

mycursor = mydb.cursor()

sql = "INSERT INTO customers (name, address) VALUES (%s, %s)"
val = ("John", "Highway 21")
mycursor.execute(sql, val)

mydb.commit()

print(mycursor.rowcount, "record inserted.")
```

▼ 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

```
# Fill the "customers" table with data:

import mysql.connector

mydb = mysql.connector.connect(
    host="localhost",
    user="yourusername",
    password="yourpassword",
    database="mydatabase"
)

mycursor = mydb.cursor()

sql = "INSERT INTO customers (name, address) VALUES (%s, %s)"
val = [
    ('Peter', 'Lowstreet 4'),
    ('Amy', 'Apple st 652'),
```

```

('Hannah', 'Mountain 21'),
('Michael', 'Valley 345'),
('Sandy', 'Ocean blvd 2'),
('Betty', 'Green Grass 1'),
('Richard', 'Sky st 331'),
('Susan', 'One way 98'),
('Vicky', 'Yellow Garden 2'),
('Ben', 'Park Lane 38'),
('William', 'Central st 954'),
('Chuck', 'Main Road 989'),
('Viola', 'Sideway 1633')
]

mycursor.executemany(sql, val)

mydb.commit()

print(mycursor.rowcount, "was inserted.")

```

▼ 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 one row, and return the ID:

import mysql.connector

mydb = mysql.connector.connect(
    host="localhost",
    user="yourusername",
    password="yourpassword",
    database="mydatabase"
)

mycursor = mydb.cursor()

sql = "INSERT INTO customers (name, address) VALUES (%s, %s)"
val = ("Michelle", "Blue Village")
mycursor.execute(sql, val)

mydb.commit()

print("1 record inserted, ID:", mycursor.lastrowid)

```

▼ Select From a Table

To select from a table in MySQL, use the "SELECT" statement.

Note: We use the `fetchall()` method, which fetches all rows from the last executed statement

```
# Select all records from the "customers" table, and display the result:

import mysql.connector

mydb = mysql.connector.connect(
    host="localhost",
    user="yourusername",
    password="yourpassword",
    database="mydatabase"
)

mycursor = mydb.cursor()

mycursor.execute("SELECT * FROM customers")

myresult = mycursor.fetchall()

for x in myresult:
    print(x)
```

▼ Selecting Columns

To select only some of the columns in a table, use the "SELECT" statement followed by the column name(s)

```
# Select only the name and address columns:

import mysql.connector

mydb = mysql.connector.connect(
    host="localhost",
    user="yourusername",
    password="yourpassword",
    database="mydatabase"
)

mycursor = mydb.cursor()

mycursor.execute("SELECT name, address FROM customers")

myresult = mycursor.fetchall()

for x in myresult:
    print(x)
```

▼ Using the fetchone() Method

- If you are only interested in one row, you can use the fetchone() method.
- The fetchone() method will return the first row of the result

```
# Fetch only one row:

import mysql.connector

mydb = mysql.connector.connect(
    host="localhost",
    user="yourusername",
    password="yourpassword",
    database="mydatabase"
)

mycursor = mydb.cursor()

mycursor.execute("SELECT * FROM customers")

myresult = mycursor.fetchone()

print(myresult)
```

▼ Python MySQL Where

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

```
# Select record(s) where the address is "Park Lane 38": result:

import mysql.connector

mydb = mysql.connector.connect(
    host="localhost",
    user="yourusername",
    password="yourpassword",
    database="mydatabase"
)

mycursor = mydb.cursor()

sql = "SELECT * FROM customers WHERE address ='Park Lane 38'"

mycursor.execute(sql)

myresult = mycursor.fetchall()
```



```
for x in myresult:
    print(x)
```

▼ Python MySQL 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.

```
# Sort the result alphabetically by name: result:

import mysql.connector

mydb = mysql.connector.connect(
    host="localhost",
    user="yourusername",
    password="yourpassword",
    database="mydatabase"
)

mycursor = mydb.cursor()

sql = "SELECT * FROM customers ORDER BY name"

mycursor.execute(sql)

myresult = mycursor.fetchall()

for x in myresult:
    print(x)
```

▼ ORDER BY DESC

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

```
# Sort the result reverse alphabetically by name:

import mysql.connector

mydb = mysql.connector.connect(
    host="localhost",
    user="yourusername",
    password="yourpassword",
    database="mydatabase"
```

```

)

mycursor = mydb.cursor()

sql = "SELECT * FROM customers ORDER BY name DESC"

mycursor.execute(sql)

myresult = mycursor.fetchall()

for x in myresult:
    print(x)

```

▼ Python MySQL Delete From By

Delete Record

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

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

Notice the WHERE clause in the DELETE syntax: The WHERE clause specifies which record(s) that should be deleted. If you omit the WHERE clause, all records will be deleted!

```

# Delete any record where the address is "Mountain 21":

import mysql.connector

mydb = mysql.connector.connect(
    host="localhost",
    user="yourusername",
    password="yourpassword",
    database="mydatabase"
)

mycursor = mydb.cursor()

sql = "DELETE FROM customers WHERE address = 'Mountain 21'"

mycursor.execute(sql)

mydb.commit()

print(mycursor.rowcount, "record(s) deleted")

```

▼ Python MySQL Drop Table

Delete a Table

You can delete an existing table by using the "DROP TABLE" statement

```
# Delete the table "customers":

import mysql.connector

mydb = mysql.connector.connect(
    host="localhost",
    user="yourusername",
    password="yourpassword",
    database="mydatabase"
)

mycursor = mydb.cursor()

sql = "DROP TABLE customers"

mycursor.execute(sql)
```

▼ Drop Only if Exist

If the the table you want to delete is already deleted, or for any other reason does not exist, you can use the IF EXISTS keyword to avoid getting an error.

```
# Delete the table "customers" if it exists

import mysql.connector

mydb = mysql.connector.connect(
    host="localhost",
    user="yourusername",
    password="yourpassword",
    database="mydatabase"
)

mycursor = mydb.cursor()

sql = "DROP TABLE IF EXISTS customers"

mycursor.execute(sql)
```

▼ Python MySQL Update Table

Update Table

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

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

Notice the WHERE clause in the UPDATE syntax: The WHERE clause specifies which record or records that should be updated. If you omit the WHERE clause, all records will be updated!

```
# Overwrite the address column from "Valley 345" to "Canyon 123":

import mysql.connector

mydb = mysql.connector.connect(
    host="localhost",
    user="yourusername",
    password="yourpassword",
    database="mydatabase"
)

mycursor = mydb.cursor()

sql = "UPDATE customers SET address = 'Canyon 123' WHERE address = 'Valley 345'"

mycursor.execute(sql)

mydb.commit()

print(mycursor.rowcount, "record(s) affected")
```

▼ Python MySQL Limit

Limit the Result

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

```
# Select the 5 first records in the "customers" table:

import mysql.connector

mydb = mysql.connector.connect(
    host="localhost",
    user="yourusername",
    password="yourpassword",
    database="mydatabase"
)

mycursor = mydb.cursor()

mycursor.execute("SELECT * FROM customers LIMIT 5")
```

```
myresult = mycursor.fetchall()

for x in myresult:
    print(x)
```

▼ Start From Another Position

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

```
# Start from position 3, and return 5 records:

import mysql.connector

mydb = mysql.connector.connect(
    host="localhost",
    user="yourusername",
    password="yourpassword",
    database="mydatabase"
)

mycursor = mydb.cursor()

mycursor.execute("SELECT * FROM customers LIMIT 5 OFFSET 2")

myresult = mycursor.fetchall()

for x in myresult:
    print(x)
```