Python MySQL Database

To be able to experiment with the code examples in this tutorial, you should have MySQL installed on your computer.

You can download a free MySQL database at https://www.mysql.com/downloads/.

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:

Download and install "MySQL Connector":

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

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:

```
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",
  passwd="yourpassword"
)
print(mydb)
```

Output: `<mysql.connector.connection.MySQLConnection object ar 0x016645F0>`

Now you can start querying the database using SQL statements.

Python MySQL Create Database

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

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  passwd="yourpassword"
)
mycursor = mydb.cursor()
mycursor.execute("CREATE DATABASE mydatabase")
```

If the above code was executed with no errors, you have successfully created a database.

Check if Database Exists

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

```
import mysql.connector
mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  passwd="yourpassword"
)
mycursor = mydb.cursor()
mycursor.execute("SHOW DATABASES")
for x in mycursor:
  print(x)
```

Output:

```
('information_scheme',)
('mydatabase',)
('performance_schema',)
('sys',)
```

Or you can try to access the database when making the connection:

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

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

Python MySQL Create 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",
  passwd="yourpassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

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

If the above code was executed with no errors, you have now successfully created a table.

Check if Table Exists

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

```
import mysql.connector

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

mycursor = mydb.cursor()

mycursor.execute("SHOW TABLES")

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

Output: ('customers',)

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.

```
import mysql.connector
mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  passwd="yourpassword",
  database="mydatabase"
)
mycursor = mydb.cursor()
mycursor.execute("CREATE TABLE customers (id INT
AUTO_INCREMENT PRIMARY KEY, name VARCHAR(255), address
VARCHAR(255))")
```

If the table already exists, use the ALTER TABLE keyword:

```
import mysql.connector
mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  passwd="yourpassword",
  database="mydatabase"
)
mycursor = mydb.cursor()
mycursor.execute("ALTER TABLE customers ADD COLUMN id INT
AUTO_INCREMENT PRIMARY KEY")
```

Python MySQL Insert Into Table

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  passwd="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.")
```

```
Output: 1 record inserted.
```

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

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:

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

Output: 13 record 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 that one row, the id of the last inserted row is returned.

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  passwd="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)
```

Output: 1 record inserted, ID: 15

Select From a Table

```
import mysql.connector
mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  passwd="yourpassword",
  database="mydatabase"
)
mycursor = mydb.cursor()
mycursor.execute("SELECT * FROM customers")
myresult = mycursor.fetchall()
for x in myresult:
  print(x)
```

```
Output:
(1, 'John', 'Highway 21')
(2, 'Peter', 'Lowstreet 27')
(3, 'Amy', 'Apple st 652')
(4, 'Hannah', 'Mountain 21')
(5, 'Michael', 'Valley 345')
(6, 'Sandy', 'Ocean blvd 2')
(7, 'Betty', 'Green Grass 1')
(8, 'Richard', 'Sky st 331')
(9, 'Susan', 'One way 98')
(10, 'Vicky', 'Yellow Garden 2')
(11, 'Ben', 'Park Lane 38')
(12, 'William', 'Central st 954')
(13, 'Chuck', 'Main Road 989')
(14, 'Viola', 'Sideway 1633')
(15, 'Michelle', 'Blue Village')
```

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

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:

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

myresult = mycursor.fetchone()

print(myresult)
```

Output: `(1, 'John', 'Highway 21')`

Update Table

Overwrite the address column from "Valley 345" to "Canyoun 123":

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  passwd="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")
```

```
Output: 1 record(s) affected
```

Python MySQL Limit

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",
  passwd="yourpassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

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

myresult = mycursor.fetchall()

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

Output:

```
(1, 'John', 'Highway 21')
(2, 'Peter', 'Lowstreet 27')
(3, 'Amy', 'Apple st 652')
(4, 'Hannah', 'Mountain 21')
(5, 'Michael', 'Valley 345')
```

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",
  passwd="yourpassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

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

myresult = mycursor.fetchall()

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

Output:

```
(3, 'Amy', 'Apple st 652')
(4, 'Hannah', 'Mountain 21')
(5, 'Michael', 'Valley 345')
(6, 'Sandy', 'Ocean blvd 2')
(7, 'Betty', 'Green Grass 1')
```

Insert / Update / Alter / Delete Commands

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

Note: After Drop command, commit() cannot execute on table.

For More on Python MySQL connectivity and projects visit the Python Projects in the blog.