# Using Database in Python

4.1. Database Management Systems

4.2. MySQL Database

4.3. Installing and Configuring MySQL Server

4.4. Connecting to MySQL Server from Python

4.6. Inserting Data into Database from Python GUI

4.7. Selecting Data from Database and Populating

Python GUI

4.8. Updating and Deleting Data from Database

## Database Management Systems

The name indicates what the database is. A database is one of the essential components for many applications and is used for storing a series of data in a single set. In other words, it is a group/package of information that is put in order so that it can be easily accessed, manage, and update

A database management system (DBMS) refers to the technology for creating and managing databases. DBMS is a software tool to organize (create, retrieve, update, and manage) data in a database.

In a database, even the smallest portion of information becomes the data. For example, a Student is a data, a roll number is a data, and the address is data, height, weight, marks everything is data. In brief, all the living and non-living objects in this world are data.

One of the primary aims of a database is to supply users with an abstract view of data, hiding a certain element of how data is stored and manipulated. Therefore, the starting point for the design of a database should be an abstract and general description of the information needs of the organization that is to be represented in the database. And hence you will require an environment to store data and make it work as a database.

## MySQL Database

Python can be used in database applications.

One of the most popular databases is MySQL.

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/>.

## Installing and Configuring MySQL Server

### 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-python

Now you have downloaded and installed a MySQL driver.

### 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:

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.

## Connecting to MySQL Server from Python

### Create Connection

Start by creating a connection to the database.

Use the username and password from your MySQL database:

demo\_mysql\_connection.py:

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:

### Example

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")  
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:

### Example

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)

### Example

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

### Example

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))")

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:

### Example

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.

### Example

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, name VARCHAR(255), address VARCHAR(255))")

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

### Example

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.

### Example

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:

### Example

Fill the "customers" table with data:

import mysql.connector  
  
mydb = mysql.connector.connect(  
  host="localhost",  
  user="yourusername",  
  password="yourpassword",  
  database="mydatabase"  
)  
val = (‘kef’,’WG’  
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.

### Example

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)

4.7. Selecting Data from Database and Populating Python GUI

### Select From a Table

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

### Example

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):

### Example

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:

### Example

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)

### Select With a Filter

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

### Example

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)

### Wildcard Characters

You can also select the records that starts, includes, or ends with a given letter or phrase.

Use the %  to represent wildcard characters:

### Example

Select records where the address contains the word "way":

import mysql.connector  
  
mydb = mysql.connector.connect(  
  host="localhost",  
  user="yourusername",  
  password="yourpassword",  
  database="mydatabase"  
)  
  
mycursor = mydb.cursor()  
  
sql = "SELECT \* FROM customers WHERE address LIKE '%way%'"  
  
mycursor.execute(sql)  
  
myresult = mycursor.fetchall()  
  
for x in myresult:  
  print(x)

## Updating and Deleting Data from Database

### Update Table

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

#### Example

Overwrite the address column from "Valley 345" to "Canyoun 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")

### Delete Record

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

#### Example

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")

## Practical example in PostgreSQL

### Install

pip3 install psycopg2

python -m pip install psycopg2

Or

tkinter

pip3 download psycopg2 -d"C:\repo"

pip install -r requirements.txt --find-links=C:\repo --no-index

pip install --no-index --find-links /path/to/download/dir/ -r requirements.txt

import psycopg2

conn = psycopg2.connect(

dbname='ethhotel2021v2',

user='postgres',

password='postgres',

host='localhost')

cursor = conn.cursor()

cursor.execute('SELECT \* FROM city LIMIT 10')

records = cursor.fetchall()

for row in records:

print(row[1])

for x in records:

print(x[2:6])

cursor.close()

conn.close()