Python MySQL

Python can be used in database applications.

One of the most popular databases is MySQL.

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 MySQL on Mac by

```
~ % brew install mysql
```

To start the MySQL server, you would run

```
~ % brew services start mysql
~ % mysql -u root
```

This will start a MySQL server on your machine with username root and no password. You can then run SQL queries on your database.

To stop the MySQL server, you would run

```
        ~ % brew services stop 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:

```
        ~ % pip install mysql-connector-python
```

Test MySQL Connector

Now you can test if the installation was successful.

In [1]:

```
# Import the mysql connector
```

```
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:

In [2]:
```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root"
)

print(mydb)
```

```
<mysql.connector.connection_cext.CMySQLConnection object at 0x103d2e970>
```

Now you can start querying the database using Python.

Arsalan

In [ ]:
```

```

Python MySQL Create Database

Creating a Database

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

In [1]:

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

import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root"
)

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:

In [2]:

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

import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root"
)

mycursor = mydb.cursor()

mycursor.execute("SHOW DATABASES")

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

```
('information_schema',)
('mydatabase',)
('mysql',)
('performance_schema',)
('sys',)
```

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

In [3]:
```
# Try connecting to the database "mydatabase":

import mysql.connector

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

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

Arsalan

In [ ]:
```
```

Python MySQL Create Table

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

In [1]:

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

import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
    database="mydatabase"
)

mycursor = mydb.cursor()

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

If the above code was executed with no errors, you have 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:

In [2]:

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

import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
    database="mydatabase"
)

mycursor = mydb.cursor()

mycursor.execute("SHOW TABLES")

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

```
('customers',)
```

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

In [3]:

```
# Check if "customers" table exists:

import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
    database="mydatabase"
)

mycursor = mydb.cursor()

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

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

In [4]:

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

import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
    database="mydatabase"
)

mycursor = mydb.cursor()

mycursor.execute("CREATE TABLE customers (id INT AUTO_INCREMENT PRIMARY K
```

```
---------------------------------------------------------------------------
-
MySQLInterfaceError                       Traceback (most recent call las
t)
File ~/Developer/Pycharm/LearnPython/venv/lib/python3.9/site-packages/mysq
l/connector/connection_cext.py:705, in CMySQLConnection.cmd_query(self, qu
ery, raw, buffered, raw_as_string)
    704         query = query.encode("utf-8")
--> 705     self. cmysql.query(
    706         query,
    707         raw=raw,
    708         buffered=buffered,
    709         raw as string=raw as string,
    710         query_attrs=self.query_attrs,
    711     )
    712 except MySQLInterfaceError as err:

MySQLInterfaceError: Table 'customers' already exists

The above exception was the direct cause of the following exception:

ProgrammingError                          Traceback (most recent call las
t)
Cell In[4], line 13
      5 mydb = mysql.connector.connect(
      6   host="localhost",
      7   user="root",
      8     database="mydatabase"
      9 )
     11 mycursor = mydb.cursor()
---> 13 mycursor.execute("CREATE TABLE customers (id INT AUTO_INCREMENT PR
IMARY KEY, name VARCHAR(255), address VARCHAR(255))")

File ~/Developer/Pycharm/LearnPython/venv/lib/python3.9/site-packages/mysq
l/connector/cursor_cext.py:357, in CMySQLCursor.execute(self, operation, p
arams, multi)
    352             raise ProgrammingError(
    353                 "Not all parameters were used in the SQL statemen
t"
    354             )
    356 try:
--> 357     result = self._connection.cmd_query(
    358         stmt,
    359         raw=self._raw,
    360         buffered=self._buffered,
    361         raw_as_string=self._raw_as_string,
    362     )
    363 except MySQLInterfaceError as err:
    364     raise get_mysql_exception(
    365         msg=err.msg, errno=err.errno, sqlstate=err.sqlstate
    366     ) from err

File ~/Developer/Pycharm/LearnPython/venv/lib/python3.9/site-packages/mysq
l/connector/opentelemetry/context_propagation.py:97, in with_context_propa
gation.<locals>.wrapper(cnx, *args, **kwargs)
     95 # pylint: disable=possibly-used-before-assignment
     96 if not OTEL_ENABLED or not cnx.otel_context_propagation:
---> 97     return method(cnx, *args, **kwargs)
     99 current_span = trace.get_current_span()
    100 tp_header = None

File ~/Developer/Pycharm/LearnPython/venv/lib/python3.9/site-packages/mysq
l/connector/connection_cext.py:713, in CMySQLConnection.cmd_query(self, qu
ery, raw, buffered, raw_as_string)
    705     self._cmysql.query(
    706         query,
    707         raw=raw,
   (...)
    710         query_attrs=self.query_attrs,
    711     )
    712 except MySQLInterfaceError as err:
--> 713     raise get_mysql_exception(
    714         err.errno, msg=err.msg, sqlstate=err.sqlstate
    715     ) from err
    716 except AttributeError as err:
    717     addr = (
    718         self._unix_socket if self._unix_socket else f"{self._hos
t}:{self._port}"
    719     )

ProgrammingError: 1050 (42S01): Table 'customers' already exists
```

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

In [2]:

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

import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
    database="mydatabase"
)

mycursor = mydb.cursor()

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

if the above code was executed with no errors, you have successfully created a primary key.

Arsalan

In [ ]:

```
```

Python MySQL Insert Into Table

Insert Into Table

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

In [1]:
```
# Insert a record in the "customers" table:

import mysql.connector

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

```
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:

In [2]:
```
# Insert multiple rows in the "customers" table:

import mysql.connector

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

```
13 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.

In [6]:

```
# Insert one row, and return the ID:

import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
    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(mycursor.rowcount, "record inserted.\nID:", mycursor.lastrowid)
```

```
1 record inserted.
ID: 16
```

Arsalan

In [ ]:

```
```

Python MySQL Select From

Select From a Table

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

In [1]:

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

import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
    database="mydatabase"
)

mycursor = mydb.cursor()

mycursor.execute("SELECT * FROM customers")

myresult = mycursor.fetchall()

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

```
('John', 'Highway 21', 1)
('Peter', 'Lowstreet 4', 2)
('Amy', 'Apple st 652', 3)
('Hannah', 'Mountain 21', 4)
('Michael', 'Valley 345', 5)
('Sandy', 'Ocean blvd 2', 6)
('Betty', 'Green Grass 1', 7)
('Richard', 'Sky st 331', 8)
('Susan', 'One way 98', 9)
('Vicky', 'Yellow Garden 2', 10)
('Ben', 'Park Lane 38', 11)
('William', 'Central st 954', 12)
('Chuck', 'Main Road 989', 13)
('Viola', 'Sideway 1633', 14)
('Michelle', 'Blue Village', 15)
('Michelle', 'Blue Village', 16)
```

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

Selecting Columns

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

In [2]:

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

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
    database="mydatabase"
)

mycursor = mydb.cursor()

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

myresult = mycursor.fetchall()

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

```
('John', 'Highway 21')
('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')
('Michelle', 'Blue Village')
('Michelle', 'Blue Village')
```

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:

In [3]:

```
# Fetch only one row:

import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
    database="mydatabase"
)

mycursor = mydb.cursor()

mycursor.execute("SELECT * FROM customers")

myresult = mycursor.fetchone()

print(myresult)
```

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

Arsalan

```
In [ ]:
```

Python MySQL Where

Select With a Filter

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

In [4]:
```
# Select record(s) where the address is "Park Lane 38"

import mysql.connector

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

```
('Ben', 'Park Lane 38', 11)
```

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:

In [5]:
```
# Select records where the address contains the word "way"

import mysql.connector

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

```
('John', 'Highway 21', 1)
('Susan', 'One way 98', 9)
('Viola', 'Sideway 1633', 14)
```

Prevent SQL Injection

When query values are provided by the user, you should escape the values.

This is to prevent SQL injections, which is a common web hacking technique to destroy or misuse your database.

The **mysql.connector** module has methods to escape query values:

In [6]:

```
# Escape query values by using the placholder %s method:

import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
    database="mydatabase"
)

mycursor = mydb.cursor()

sql = "SELECT * FROM customers WHERE address = %s"
adr = ("Yellow Garden 2", )

mycursor.execute(sql, adr)

myresult = mycursor.fetchall()

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

```
('Vicky', 'Yellow Garden 2', 10)
```

Arsalan

In [ ]:

```
```

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.

In [1]:

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

import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
    database="mydatabase"
)

mycursor = mydb.cursor()

sql = "SELECT * FROM customers ORDER BY name"

mycursor.execute(sql)

myresult = mycursor.fetchall()

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

```
('Amy', 'Apple st 652', 3)
('Ben', 'Park Lane 38', 11)
('Betty', 'Green Grass 1', 7)
('Chuck', 'Main Road 989', 13)
('Hannah', 'Mountain 21', 4)
('John', 'Highway 21', 1)
('Michael', 'Valley 345', 5)
('Michelle', 'Blue Village', 15)
('Michelle', 'Blue Village', 16)
('Peter', 'Lowstreet 4', 2)
('Richard', 'Sky st 331', 8)
('Sandy', 'Ocean blvd 2', 6)
('Susan', 'One way 98', 9)
('Vicky', 'Yellow Garden 2', 10)
('Viola', 'Sideway 1633', 14)
('William', 'Central st 954', 12)
```

Order By Descending

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

In [2]:

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

```
import mysql.connector

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

```
('William', 'Central st 954', 12)
('Viola', 'Sideway 1633', 14)
('Vicky', 'Yellow Garden 2', 10)
('Susan', 'One way 98', 9)
('Sandy', 'Ocean blvd 2', 6)
('Richard', 'Sky st 331', 8)
('Peter', 'Lowstreet 4', 2)
('Michelle', 'Blue Village', 15)
('Michelle', 'Blue Village', 16)
('Michael', 'Valley 345', 5)
('John', 'Highway 21', 1)
('Hannah', 'Mountain 21', 4)
('Chuck', 'Main Road 989', 13)
('Betty', 'Green Grass 1', 7)
('Ben', 'Park Lane 38', 11)
('Amy', 'Apple st 652', 3)
```

Arsalan

```
In [ ]:
```

Python MySQL Delete From By

Delete Record

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

In [1]:
```
# Delete any record where the address is "Mountain 21"

import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
    database="mydatabase"
)

mycursor = mydb.cursor()

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

mycursor.execute(sql)

mydb.commit()

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

```
1 record(s) deleted
```

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

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

Prevent SQL Injection

It is considered a good practice to escape the values of any query, also in delete statements.

This is to prevent SQL injections, which is a common web hacking technique to destroy or misuse your database.

The **mysql.connector** module uses the placeholder **%s** to escape values in the delete statement:

In [2]:
```
# Escape values by using the placeholder %s method:

import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
    database="mydatabase"
)

mycursor = mydb.cursor()

sql = "DELETE FROM customers WHERE address = %s"
adr = ("Yellow Garden 2", )

mycursor.execute(sql, adr)

mydb.commit()

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

```
1 record(s) deleted
```

Arsalan

In [ ]:
```
```

Python MySQL Drop Table

Delete a Table

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

In [1]:

```
# Delete the table "customers"

import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
    database="mydatabase"
)

mycursor = mydb.cursor()

sql = "DROP TABLE customers"

mycursor.execute(sql)
```

```
---------------------------------------------------------------------------
-
MySQLInterfaceError                       Traceback (most recent call las
t)
File ~/Developer/Pycharm/LearnPython/venv/lib/python3.9/site-packages/mysq
l/connector/connection_cext.py:705, in CMySQLConnection.cmd_query(self, qu
ery, raw, buffered, raw_as_string)
    704         query = query.encode("utf-8")
--> 705     self. cmysql.query(
    706         query,
    707         raw=raw,
    708         buffered=buffered,
    709         raw as string=raw as string,
    710         query_attrs=self.query_attrs,
    711     )
    712 except MySQLInterfaceError as err:

MySQLInterfaceError: Unknown table 'mydatabase.customers'

The above exception was the direct cause of the following exception:

ProgrammingError                          Traceback (most recent call las
t)
Cell In[1], line 15
     11 mycursor = mydb.cursor()
     13 sql = "DROP TABLE customers"
---> 15 mycursor.execute(sql)

File ~/Developer/Pycharm/LearnPython/venv/lib/python3.9/site-packages/mysq
l/connector/cursor_cext.py:357, in CMySQLCursor.execute(self, operation, p
arams, multi)
    352             raise ProgrammingError(
    353                 "Not all parameters were used in the SQL statemen
t"
    354                 )
    356 try:
--> 357     result = self._connection.cmd_query(
    358         stmt,
    359         raw=self._raw,
    360         buffered=self._buffered,
    361         raw_as_string=self._raw_as_string,
    362     )
    363 except MySQLInterfaceError as err:
    364     raise get_mysql_exception(
    365         msg=err.msg, errno=err.errno, sqlstate=err.sqlstate
    366     ) from err

File ~/Developer/Pycharm/LearnPython/venv/lib/python3.9/site-packages/mysq
l/connector/opentelemetry/context_propagation.py:97, in with_context_propa
gation.<locals>.wrapper(cnx, *args, **kwargs)
     95 # pylint: disable=possibly-used-before-assignment
     96 if not OTEL_ENABLED or not cnx.otel_context_propagation:
---> 97     return method(cnx, *args, **kwargs)
     99 current_span = trace.get_current_span()
    100 tp_header = None

File ~/Developer/Pycharm/LearnPython/venv/lib/python3.9/site-packages/mysq
l/connector/connection_cext.py:713, in CMySQLConnection.cmd_query(self, qu
ery, raw, buffered, raw_as_string)
    705     self._cmysql.query(
    706         query,
    707         raw=raw,
   (...)
    710         query_attrs=self.query_attrs,
    711     )
    712 except MySQLInterfaceError as err:
--> 713     raise get_mysql_exception(
    714         err.errno, msg=err.msg, sqlstate=err.sqlstate
    715     ) from err
    716 except AttributeError as err:
    717     addr = (
    718         self._unix_socket if self._unix_socket else f"{self._hos
t}:{self._port}"
    719     )

ProgrammingError: 1051 (42S02): Unknown table 'mydatabase.customers'
```

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.

In [2]:
```
# Drop the table "customers" if it exists:

import mysql.connector

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

```
mycursor = mydb.cursor()

sql = "DROP TABLE IF EXISTS customers"

mycursor.execute(sql)
```

Arsalan

```
In [ ]:
```

Python MySQL Update Table

Update Table

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

In [2]:

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

import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
    database="mydatabase"
)

mycursor = mydb.cursor()

sql = "UPDATE customers SET address = 'Canyoun 123' WHERE address = 'Val

mycursor.execute(sql)

mydb.commit()

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

```
1 record(s) affected
```

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

> **Notice:** 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!

Prevent SQL Injection

It is considered a good practice to escape the values of any query, also in update statements.

This is to prevent SQL injections, which is a common web hacking technique to destroy or misuse your database.

The **mysql.connector** module uses the placeholder **%s** to escape values in the delete statement:

In [3]:

```
# Escape values by using the placeholder %s method:
```

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
    database="mydatabase"
)

mycursor = mydb.cursor()

sql = "UPDATE customers SET address = %s WHERE address = %s"
val = ("Valley 345", "Canyoun 123")

mycursor.execute(sql, val)

mydb.commit()

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

```
1 record(s) affected
```

Arsalan

```
In [ ]:
```

Python MySQL Limit

Limit the Result

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

In [2]:

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

import mysql.connector

mydb = mysql.connector.connect(
    host="localhost",
    user="root",
      database="mydatabase"
)

mycursor = mydb.cursor()

sql = "SELECT * from customers LIMIT 5"

mycursor.execute(sql)

myresult = mycursor.fetchall()

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

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

Start From Another Position

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

In [3]:

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

import mysql.connector

mydb = mysql.connector.connect(
    host="localhost",
    user="root",
      database="mydatabase"
)

mycursor = mydb.cursor()

sql = "SELECT * from customers LIMIT 5 OFFSET 2"
```

```
mycursor.execute(sql)

myresult = mycursor.fetchall()

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

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

Arsalan

```
In [ ]:
```

Python MySQL Join

Join Two or More Tables

You can combine rows from two or more tables, based on a related column between them, by using a JOIN statement.

Consider you have a "users" table and a "products" table:

In [1]:

```
# Users Table

import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
    database="mydatabase"
)

mycursor = mydb.cursor()

mycursor.execute("CREATE TABLE users (id INT AUTO_INCREMENT PRIMARY KEY,

sql = "INSERT INTO users (name, fav) VALUES (%s, %s)"
val = [
  ('John', 154),
  ('Peter', 154),
  ('Amy', 155),
  ('Hannah', 0),
  ('Michael', 0)
]
mycursor.executemany(sql, val)

mydb.commit()

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

sql = "SELECT * FROM users"
mycursor.execute(sql)
myresult = mycursor.fetchall()
for x in myresult:
  print(x)
```

```
5 record was inserted.
(1, 'John', 154)
(2, 'Peter', 154)
(3, 'Amy', 155)
(4, 'Hannah', 0)
(5, 'Michael', 0)
```

In [2]:

```
# Products Table

import mysql.connector
```

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

mycursor = mydb.cursor()

mycursor.execute("CREATE TABLE products (id INT AUTO_INCREMENT PRIMARY KE

sql = "INSERT INTO products (id, name) VALUES (%s, %s)"

val = [
  (154, 'Chocolate Heaven'),
  (155, 'Tasty Lemons'),
  (156, 'Vanilla Dreams')
]

mycursor.executemany(sql, val)

mydb.commit()

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

sql = "SELECT * FROM products"
mycursor.execute(sql)
myresult = mycursor.fetchall()
for x in myresult:
  print(x)
```

```
3 record was inserted.
(154, 'Chocolate Heaven')
(155, 'Tasty Lemons')
(156, 'Vanilla Dreams')
```

These two tables can be combined by using users' **fav** field and products' **id** field.

In [3]:

```
# Join users and products to see the name of the users favorite product:

import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
    database="mydatabase"
)

mycursor = mydb.cursor()

sql = "SELECT users.name AS user, products.name AS favorite FROM users JO

mycursor.execute(sql)

myresult = mycursor.fetchall()

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

```
('John', 'Chocolate Heaven')
('Peter', 'Chocolate Heaven')
('Amy', 'Tasty Lemons')
```

> **Note:** You can use JOIN instead of INNER JOIN. They will both give you the same result.

LEFT JOIN

In the example above, Hannah and Michael were excluded from the result, that is because **INNER JOIN** only shows the records where there is a match.

If you want to show all users, even if they do not have a favorite product, use the **LEFT JOIN** statement:

In [4]:
```python
# Select all users and their favorite product:

import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
    database="mydatabase"
)

mycursor = mydb.cursor()

sql = "SELECT users.name AS user, products.name AS favorite FROM users LI

mycursor.execute(sql)

myresult = mycursor.fetchall()

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

```
('John', 'Chocolate Heaven')
('Peter', 'Chocolate Heaven')
('Amy', 'Tasty Lemons')
('Hannah', None)
('Michael', None)
```

RIGHT JOIN

If you want to return all products, and the users who have them as their favorite, even if no user have them as their favorite, use the **RIGHT JOIN** statement:

In [5]:
```python
# Select all products, and the user(s) who have them as their favorite:

import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
    database="mydatabase"
)

mycursor = mydb.cursor()

sql = "SELECT users.name AS user, products.name AS favorite FROM users R

mycursor.execute(sql)

myresult = mycursor.fetchall()

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

```
('Peter', 'Chocolate Heaven')
('John', 'Chocolate Heaven')
('Amy', 'Tasty Lemons')
(None, 'Vanilla Dreams')
```

> **Note:** Hannah and Michael are excluded from the result, because there is no match for them in the "users" table.

> **Note:** You can use LEFT JOIN or RIGHT JOIN depending on which table you want to return all records from.

Arsalan

```
In [ ]:
```