SQL with Python

- To use SQL, we must import sqlite3.
    - Then create a connection using connect() method and pass the name of the database you want to access if there is a file with that name, it will open that file. Otherwise Python will create a file with the given name.
- After this, a cursor object is called to be capable to send commands to the SQL.
- Cursor is a control structure used to traverse and fetch the records of the database.
- Cursor has a major role in working with Python.
    - All the commands will be executed using cursor object only.
    - To create a table in the database, create an object and write the SQL command in it with being commented.
    - Example:- sql_comm = ”SQL statement”
- And executing the command is very easy. Call the cursor method execute and pass the name of the sql command as a parameter in it. Save a number of commands as the sql_comm and execute them.
- After you perform all your activities, save the changes in the file by committing those changes and then lose the connection.

In [1]:

```
# Python code to demonstrate table creation and
# insertions with SQL

# importing module
import sqlite3

# connecting to the database
connection = sqlite3.connect("myTable.db")

# cursor
crsr = connection.cursor()

# SQL command to create a table in the database
sql_command = """CREATE TABLE emp (
staff_number INTEGER PRIMARY KEY,
fname VARCHAR(20),
lname VARCHAR(30),
gender CHAR(1),
joining DATE);"""

# execute the statement
crsr.execute(sql_command)

# SQL command to insert the data in the table
sql_command = """INSERT INTO emp VALUES (23, "Rishabh", "Bansal", "M", "2014-03-28");"""
crsr.execute(sql_command)

# another SQL command to insert the data in the table
sql_command = """INSERT INTO emp VALUES (1, "Bill", "Gates", "M", "1980-10-28");"""
crsr.execute(sql_command)

# To save the changes in the files. Never skip this.
# If we skip this, nothing will be saved in the database.
connection.commit()

# close the connection
connection.close()
```

Fetching the data from record is simple as the inserting them

In [2]:

```
# connect withe the myTable database
connection = sqlite3.connect("myTable.db")

# cursor object
crsr = connection.cursor()

# execute the command to fetch all the data from the table emp
crsr.execute("SELECT * FROM emp")

# store all the fetched data in the ans variable
ans= crsr.fetchall()

# loop to print all the data
for i in ans:
    print(i)
```

```
(1, 'Bill', 'Gates', 'M', '1980-10-28')
(23, 'Rishabh', 'Bansal', 'M', '2014-03-28')
```

In [3]:

```
# Updation and Deletion Operation

conn = sqlite3.connect('myTable.db')

# update the student record
conn.execute("UPDATE emp SET fname = 'Sam' where fname='Rishabh'")
conn.commit()

print ("Total number of rows updated :", conn.total_changes )

cursor = conn.execute("SELECT * FROM emp")
for row in cursor:
   print (row)

conn.close()
```

```
Total number of rows updated : 1
(1, 'Bill', 'Gates', 'M', '1980-10-28')
(23, 'Sam', 'Bansal', 'M', '2014-03-28')
```

In [4]:

```
# database name to be passed as parameter
conn = sqlite3.connect('myTable.db')

# delete student record from database
conn.execute("DELETE from emp where fname='Sam'")
conn.commit()
print ("Total number of rows deleted :", conn.total_changes)

cursor = conn.execute("SELECT * FROM emp")
for row in cursor:
   print (row)

conn.close()
```

```
Total number of rows deleted : 1
(1, 'Bill', 'Gates', 'M', '1980-10-28')
```

Data input by User

In [6]:

```
# creates a database in RAM
con = sqlite3.connect(":memory:")
cur = con.cursor()
cur.execute("create table person (name, age, id)")

print ("Enter 5 students names:")
who = [input() for i in range(5)]
print ("Enter their ages respectively:")
age = [int(input()) for i in range(5)]
print ("Enter their ids respectively:")
p_id = [int(input()) for i in range(5)]
n = len(who)

for i in range(n):

    # This is the q-mark style:
    cur.execute("insert into person values (?, ?, ?)", (who[i], age[i], p_id[i]))

    # And this is the named style:
    cur.execute("select * from person")

    # Fetches all entries from table
    print (cur.fetchall())
```

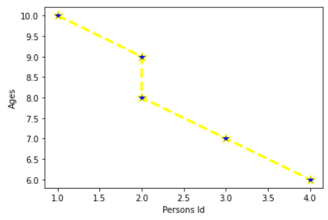
```
Enter 5 students names:
gsaidheeraj
saidheeraj
dheeraj
sai
myself
Enter their ages respectively:
19
2021
22
23
18
Enter their ids respectively:
1
2
3
4
5
[('gsaidheeraj', 19, 1)]
[('gsaidheeraj', 19, 1), ('saidheeraj', 2021, 2)]
[('gsaidheeraj', 19, 1), ('saidheeraj', 2021, 2), ('dheeraj', 22, 3)]
[('gsaidheeraj', 19, 1), ('saidheeraj', 2021, 2), ('dheeraj', 22, 3), ('sai', 23, 4)]
[('gsaidheeraj', 19, 1), ('saidheeraj', 2021, 2), ('dheeraj', 22, 3), ('sai', 23, 4), ('myself', 18, 5)]
```

Graphing with SQLite

In [7]:

```
import matplotlib.pyplot as plt

def graph_data(p_id,age):

    # plotting the points
    plt.plot(p_id, age, color='yellow', linestyle='dashed', linewidth = 3,
    marker='*', markerfacecolor='blue', markersize=12)

    # naming the x axis
    plt.xlabel('Persons Id')

    # naming the y axis
    plt.ylabel('Ages')

    # plt.plot(p_id,age)
    plt.show()

print ("Enter 5 students names:")
who = [input() for i in range(5)]
print ("Enter their ages respectively:")
age = [int(input()) for i in range(5)]
print ("Enter their ids respectively:")
p_id = [int(input()) for i in range(5)]

# calling graph function
graph_data(p_id,age)
```

```
Enter 5 students names:

saidheeraj
gsaidheeraj
gsaidheeraj
gsaidheeraj
Enter their ages respectively:
10
9
8
7
6
Enter their ids respectively:
1
2
2
3
4
```



executescript()

- This is a convenience method for executing multiple SQL statements at once. It executes the SQL script it gets as a parameter.

In [8]:

```
# Connection with the DataBase
# 'library.db'
connection = sqlite3.connect("library.db")
cursor = connection.cursor()
```

```
# SQL piece of code Executed
# SQL piece of code Executed
cursor.executescript("""
    CREATE TABLE people(
        firstname,
        lastname,
        age
    );

    CREATE TABLE book(
        title,
        author,
        published
    );

    INSERT INTO
    book(title, author, published)
    VALUES (
        'Dan Clarke''s GFG Detective Agency',
        'Sean Simpsons',
        1987
    );
    """)

sql = """
SELECT COUNT(*) FROM book;"""

cursor.execute(sql)

# The output in fetched and returned
# as a List by fetchall()
result = cursor.fetchall()
print(result)

sql = """
SELECT * FROM book;"""

cursor.execute(sql)

result = cursor.fetchall()
print(result)

# Changes saved into database
connection.commit()

# Connection closed(broken)
# with DataBase
connection.close()
```

```
[(1,)]
[("Dan Clarke's GFG Detective Agency", 'Sean Simpsons', 1987)]
```

executemany()

- It is often the case when, large amount of data has to be inserted into database from Data Files(for simpler case take Lists, arrays).
- It would be simple to iterate the code many a times than write every time, each line into database.

In [9]:

```
# Connection with the DataBase
# 'library.db'
connection = sqlite3.connect("library.db")
cursor = connection.cursor()

# SQL piece of code Executed
cursor.execute("""

    CREATE TABLE books(
        title,
        author,
        published);""")

List = [('A', 'B', 2008), ('C', 'D', 2008),
                          ('E', 'F', 2010)]

connection. executemany("""

    INSERT INTO
    books(title, author, published)
    VALUES (?, ?, ?)""", List)

sql = """
      SELECT * FROM books;"""
cursor.execute(sql)
result = cursor.fetchall()
for x in result:
    print(x)

# Changes saved into database
connection.commit()

# Connection closed(broken)
# with DataBase
connection.close()
```

```
('A', 'B', 2008)
('C', 'D', 2008)
('E', 'F', 2010)
```

In [10]:

```
# Connection created with the
# database using sqlite3.connect()
connection = sqlite3.connect("company.db")
cursor = connection.cursor()

# Create Table command executed
sql = """
        CREATE TABLE employee (
        ID INTEGER PRIMARY KEY,
        fname VARCHAR(20),
        lname VARCHAR(30),
        gender CHAR(1),
        dob DATE);"""
cursor.execute(sql)

# Single Tuple inserted
sql = """
        INSERT INTO employee
        VALUES (1007, "Will", "Olsen", "M", "24-SEP-1865");"""
cursor.execute(sql)

# Multiple Rows inserted
List = [(1008, 'Rkb', 'Boss', 'M', "27-NOV-1864"),
        (1098, 'Sak', 'Rose', 'F', "27-DEC-1864"),
        (1908, 'Royal', 'Bassen', "F", "17-NOV-1894")]

connection.executemany(
        "INSERT INTO employee VALUES (?, ?, ?, ?, ?)", List)

print("Method-1\n")

# Multiple Rows fetched from
# the Database
for row in connection.execute('SELECT * FROM employee ORDER BY ID'):
        print (row)

print("\nMethod-2\n")

# Method-2 to fetch multiple
# rows
sql = """
        SELECT * FROM employee ORDER BY ID;"""

cursor.execute(sql)
result = cursor.fetchall()

for x in result:
    print(x)

connection.commit()
connection.close()
```

```
Method-1

(1007, 'Will', 'Olsen', 'M', '24-SEP-1865')
(1008, 'Rkb', 'Boss', 'M', '27-NOV-1864')
(1098, 'Sak', 'Rose', 'F', '27-DEC-1864')
(1908, 'Royal', 'Bassen', 'F', '17-NOV-1894')

Method-2

(1007, 'Will', 'Olsen', 'M', '24-SEP-1865')
(1008, 'Rkb', 'Boss', 'M', '27-NOV-1864')
(1098, 'Sak', 'Rose', 'F', '27-DEC-1864')
(1908, 'Royal', 'Bassen', 'F', '17-NOV-1894')
```

In [ ]:

```
#How to Connect Python and SQL Server
import pyodbc
conn = pyodbc.connect("Driver={SQL Server Native Client 11.0};"
                      "Server=LAPTOP-LFBVI1JS\SQLEXPRESS;"
                      "Database=master;"
                      "Trusted_Connection=yes;")

cursor = conn.cursor()
cursor.execute('SELECT * FROM blood_pressure')

for row in cursor:
    print('row = %r' % (row,))
```

In [ ]:

```
# Python SQL Create DB Example
cursor = conn.cursor()
cursor.execute("SELECT name FROM master.dbo.sysdatabases")

for x in cursor:
    print('Database = %r' %x)
```

In [ ]:

```
conn.autocommit = True
cursor = conn.cursor()
cursor.execute("CREATE DATABASE [PythonDatabase1]")
```

In [ ]:

```
cursor = conn.cursor()
cursor.execute("SELECT name FROM master.dbo.sysdatabases")

for x in cursor:
    print('Database = %r' %x)
```

In [ ]:

```
# Using the Select Statement
cursor = conn.cursor()
cursor.execute('SELECT patient, sex, agegrp, bp_before from blood_pressure')

for row in cursor:
    print('row = %r' % (row,))
```

In [ ]:

```
#Another Select Statement
cursor = conn.cursor()
cursor.execute('SELECT * FROM blood_pressure')
result = cursor.fetchone()
print(result)
```

In [ ]:

```
# Order By Statement
cursor = conn.cursor()
cursor.execute('SELECT * FROM blood_pressure ORDER BY agegrp')

for row in cursor:
    print('row = %r' % (row,))
```

In [ ]:

```
# Order By Statement - Descending Order
cursor = conn.cursor()
cursor.execute('SELECT * FROM blood_pressure ORDER BY agegrp desc')

for row in cursor:
    print('row = %r' % (row,))
```

In [ ]:

```
#Selection Top N items
cursor = conn.cursor()
cursor.execute('SELECT Top 1 * FROM blood_pressure')
for row in cursor:
    print('row = %r' % (row,))
```

In [ ]:

```
#Selection Top N items and Ordering by
cursor = conn.cursor()
cursor.execute('SELECT Top 10 * FROM blood_pressure ORDER BY agegrp desc')
for row in cursor:
    print('row = %r' % (row,))
```

In [ ]:

```
#Selection Top N items by Percentage and Ordering by
cursor = conn.cursor()
cursor.execute('SELECT Top 40 Percent * FROM blood_pressure ORDER BY agegrp desc')
for row in cursor:
    print('row = %r' % (row,))
```

In [ ]:

```
#Selecting Data by Where Clause
cursor = conn.cursor()
cursor.execute("SELECT * FROM blood_pressure where sex = 'Male' ORDER BY agegrp desc")
for row in cursor:
    print('row = %r' % (row,))
```

In [ ]:

```
#Selecting Data by Where Clause
cursor = conn.cursor()
cursor.execute("SELECT * FROM blood_pressure where bp_before >= 130 ORDER BY agegrp desc")
for row in cursor:
    print('row = %r' % (row,))
```