How To Use SQLite3 in Python

How to install SQLite3 for Ubuntu:

```
sudo apt-get install sqlite3 libsqlite3-dev
```

How to Install sqlitebrowser for Ubunt:

```
sudo apt-get install sqlitebrowser
```

How to use:

- To start: ubuntu@ubuntu:~$ sqlite3
- To quit: sqlite> .quit
- To creat: ubuntu@ubuntu:~$ sqlite3 database_name.db
- To check: sqlite> .databases

Check SQLITE3 Module & Version

In [1]:

```
import sqlite3
dir(sqlite3)
```

Out[1]:

```
['Binary',
 'Cache',
 'Connection',
 'Cursor',
 'DataError',
 'DatabaseError',
 'Date',
 'DateFromTicks',
 'Error',
 'IntegrityError',
 'InterfaceError',
 'InternalError',
 'NotSupportedError',
 'OperationalError',
 'OptimizedUnicode',
 'PARSE_COLNAMES',
 'PARSE_DECLTYPES',
 'PrepareProtocol',
 'ProgrammingError',
 'Row',
 'SQLITE_ALTER_TABLE',
 'SQLITE_ANALYZE',
 'SQLITE_ATTACH',
 'SQLITE_CREATE_INDEX',
 'SQLITE_CREATE_TABLE',
 'SQLITE_CREATE_TEMP_INDEX',
 'SQLITE_CREATE_TEMP_TABLE',
 'SQLITE_CREATE_TEMP_TRIGGER',
 'SQLITE_CREATE_TEMP_VIEW',
 'SQLITE_CREATE_TRIGGER',
 'SQLITE_CREATE_VIEW',
 'SQLITE_DELETE',
 'SQLITE_DENY',
 'SQLITE_DETACH',
 'SQLITE_DROP_INDEX',
 'SQLITE_DROP_TABLE',
 'SQLITE_DROP_TEMP_INDEX',
 'SQLITE_DROP_TEMP_TABLE',
 'SQLITE_DROP_TEMP_TRIGGER',
 'SQLITE_DROP_TEMP_VIEW',
 'SQLITE_DROP_TRIGGER',
 'SQLITE_DROP_VIEW',
 'SQLITE_IGNORE',
 'SQLITE_INSERT',
 'SQLITE_OK',
 'SQLITE_PRAGMA',
 'SQLITE_READ',
 'SQLITE_REINDEX',
 'SQLITE_SELECT',
 'SQLITE_TRANSACTION',
 'SQLITE_UPDATE',
 'Statement',
 'Time',
 'TimeFromTicks',
 'Timestamp',
 'TimestampFromTicks',
 'Warning',
 '__builtins__',
 '__doc__',
 '__file__',
 '__name__',
 '__package__',
 '__path__',
 'adapt',
 'adapters',
 'apilevel',
 'complete_statement',
 'connect',
 'converters',
 'datetime',
 'dbapi2',
 'enable_callback_tracebacks',
 'enable_shared_cache',
 'paramstyle',
 'register_adapter',
 'register_converter',
 'sqlite_version',
 'sqlite_version_info',
 'threadsafety',
 'time',
 'version',
 'version_info',
 'x']
```

In [2]:

```
import sqlite3
# sqlite3 module version
print(sqlite3.version)
print(sqlite3.version_info)
# sqlite3 db version
print(sqlite3.sqlite_version)
print(sqlite3.sqlite_version_info)
```

```
2.6.0
(2, 6, 0)
3.8.2
(3, 8, 2)
```

How To Use the sqlite_version function For Checking Version in sqlites

```
In [3]:
```

```python
import sqlite3
def sqlite_version() :
    try :
        conn = sqlite3.connect('sql_test.db')
        cur = conn.cursor()
        sql = "select sqlite_version() AS 'SQLite Version';"

        cur.execute(sql)
        print(cur.fetchone())

        conn.close()
        print("sqlite_version sucess")
    except Exception as err :
        print('error', err)

sqlite_version()
```

```
(u'3.8.2',)
sqlite_version sucess
```

Running DB Server

DB Connection

```
In [4]:
```

```python
import sqlite3
conn = sqlite3.connect('sql_test.db')

#####
print(type(conn))
print(dir(conn))
```

```
<type 'sqlite3.Connection'>
['DataError', 'DatabaseError', 'Error', 'IntegrityError', 'InterfaceEr
ror', 'InternalError', 'NotSupportedError', 'OperationalError', 'Progr
ammingError', 'Warning', '__call__', '__class__', '__delattr__', '__do
c__', '__enter__', '__exit__', '__format__', '__getattribute__', '__ha
sh__', '__init__', '__new__', '__reduce__', '__reduce_ex__', '__repr_
_', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', 'clos
e', 'commit', 'create_aggregate', 'create_collation', 'create_functio
n', 'cursor', 'enable_load_extension', 'execute', 'executemany', 'exec
utescript', 'interrupt', 'isolation_level', 'iterdump', 'load_extensio
n', 'rollback', 'row_factory', 'set_authorizer', 'set_progress_handle
r', 'text_factory', 'total_changes']
```

Sqlite3.cursor

In [5]:

```
import sqlite3
conn = sqlite3.connect('sql_test.db')
cur = conn.cursor()
print(type(conn))
for i in dir(cur) :
    if not i.startswith("__") :
        print(i)
cur.close()
```

```
<type 'sqlite3.Connection'>
arraysize
close
connection
description
execute
executemany
executescript
fetchall
fetchmany
fetchone
lastrowid
next
row_factory
rowcount
setinputsizes
setoutputsize
```

cursor.excute

In [6]:

```
# Making student table
import sqlite3
conn = sqlite3.connect('sql_test.db')
cur = conn.cursor()
sql = "create table student(name text, age int)"
cur.execute(sql)
conn.commit()
conn.close()
```

In [7]:

```
# END
```

SQLite - Python Quick Guide

Cotents of Table

```
(0) Connect to Database
(1) Create a Table
(2) Delete a Table
(3) INSERT Operation
(4) SELECT Operation
(5) UPDATE Operation
(6) DELETE Operation
```

(0) Connect To Database

Following Python code shows how to connect to an existing database. If the database does not exist, then it will be created and finally a database object will be returned.

In [8]:

```
import sqlite3

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

print "Opened database successfully";
```

```
Opened database successfully
```

(1) Create a Table

Following Python program will be used to create a table in the previously created database.

Syntax

```
CREATE TABLE database_name.table_name(
    column1 datatype PRIMARY KEY(one or more columns),
    column2 datatype,
    column3 datatype,
    .....
    columnN datatype
);
```

In [9]:

```
import sqlite3

conn = sqlite3.connect('test.db')
print "Opened database successfully";

conn.execute('''CREATE TABLE COMPANY
         (ID INT PRIMARY KEY     NOT NULL,
         NAME           TEXT    NOT NULL,
         AGE            INT     NOT NULL,
         ADDRESS        CHAR(50),
         SALARY         REAL);''')
print "Table created successfully";

conn.close()
```

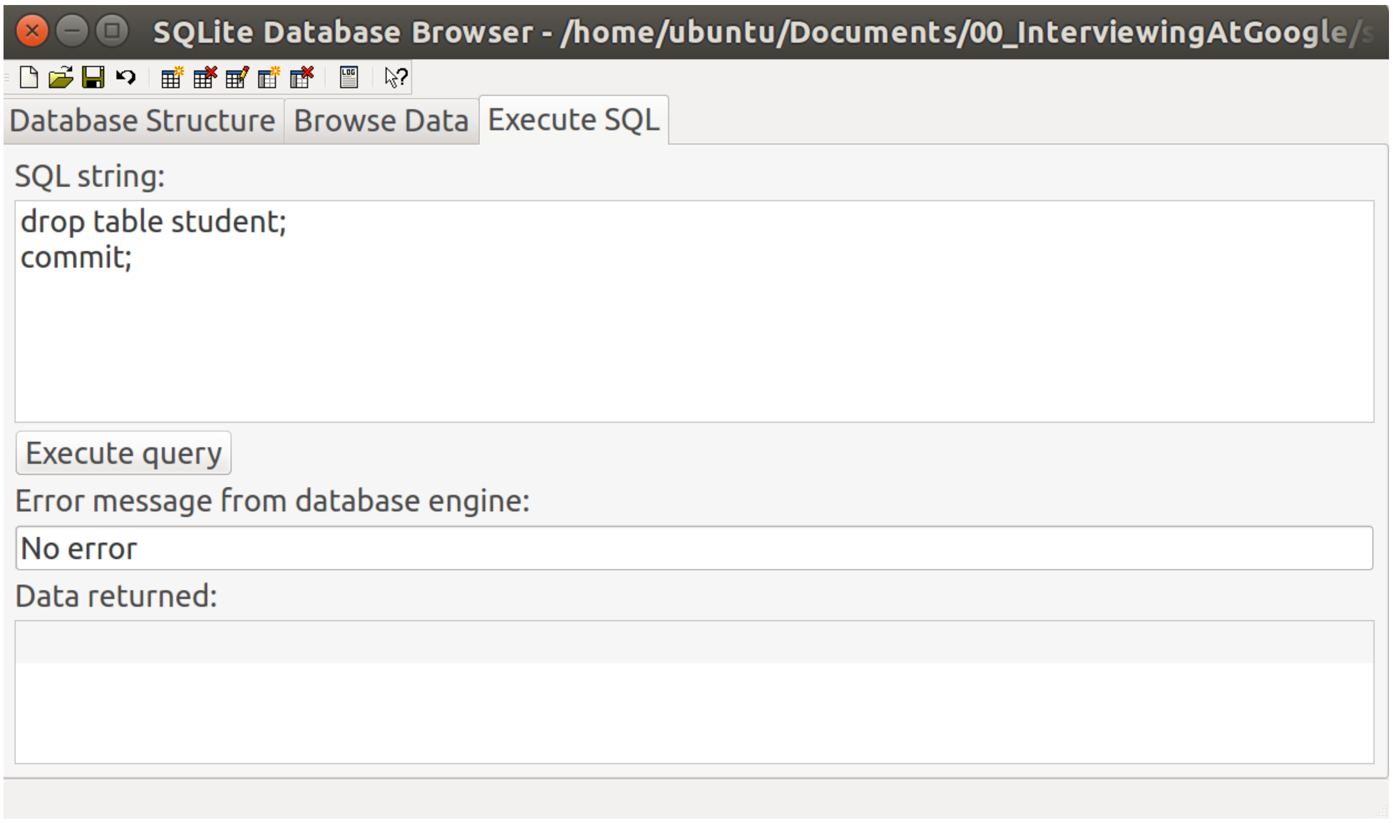
```
Opened database successfully
Table created successfully
```

(2) Delete a Table

Syntax

```
DROP TABLE database_name.table_name;
```

- (Or) We can also use at SQLiteBrowser.


SQLite Database Browser - /home/ubuntu/Documents/00_InterviewingAtGoogle/s
Database Structure
Browse Data
Execute SQL
SQL string:
Execute query
Error message from database engine:
Data returned:

```
drop table student;
commit;
```

No error

In [10]:

```
# Deleting the student table in Python
import sqlite3
conn = sqlite3.connect('sql_test.db')
cur = conn.cursor()
sql = "DROP TABLE student;"
cur.execute(sql)
conn.commit()
conn.close()
```

(3) INSERT Operation

Following Python program shows how to create records in the COMPANY table created in the above example.

Syntax

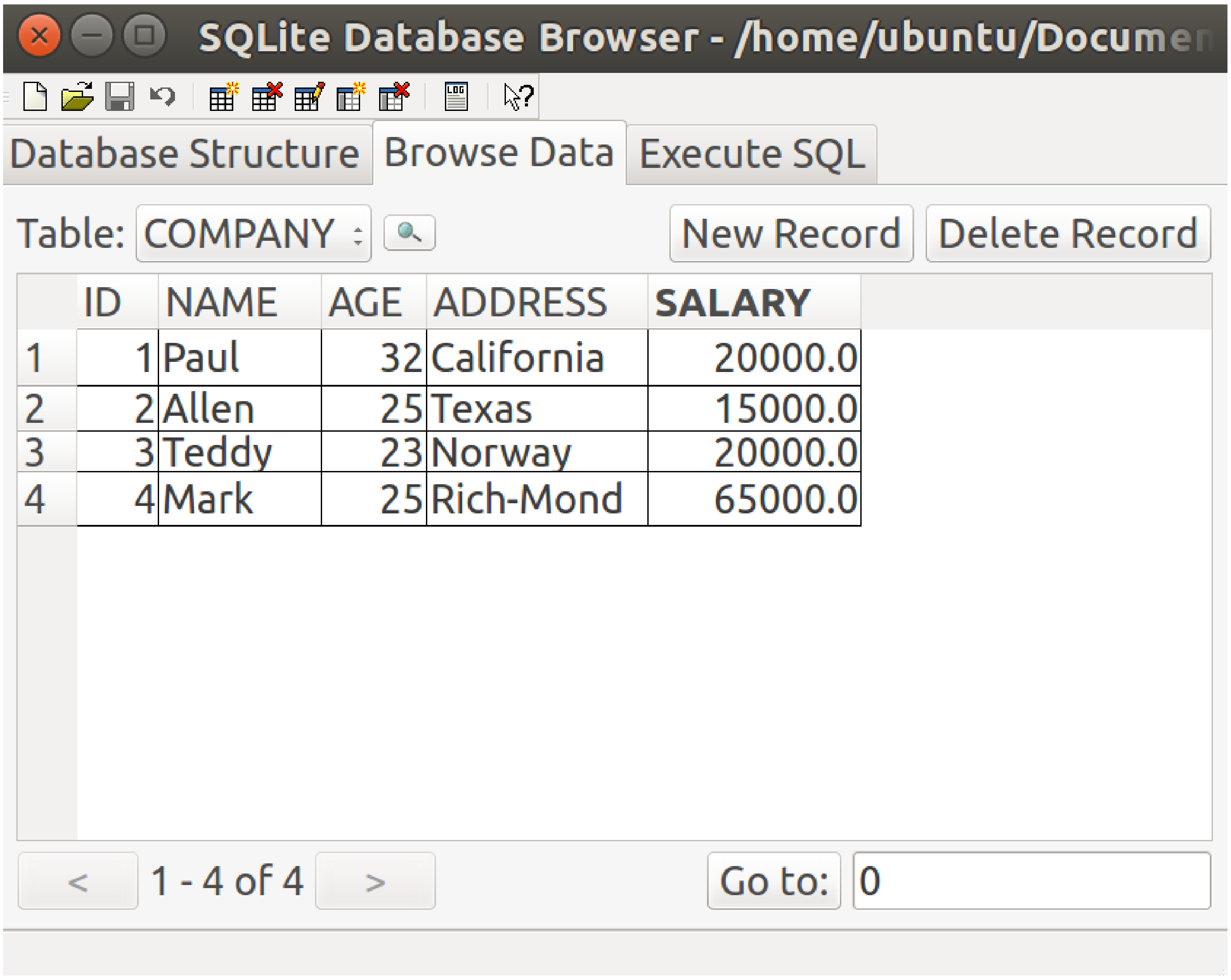
- INSERT INTO TABLE_NAME [(column1, column2, column3,...columnN)]
- VALUES (value1, value2, value3,...valueN);

In [11]:

```
import sqlite3

conn = sqlite3.connect('test.db')
print "Opened database successfully";

conn.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \
      VALUES (1, 'Paul', 32, 'California', 20000.00 )");

conn.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \
      VALUES (2, 'Allen', 25, 'Texas', 15000.00 )");

conn.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \
      VALUES (3, 'Teddy', 23, 'Norway', 20000.00 )");

conn.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \
      VALUES (4, 'Mark', 25, 'Rich-Mond ', 65000.00 )");

conn.commit()
print "Records created successfully";
conn.close()
```

```
Opened database successfully
Records created successfully
```

You can also check at DBbrowser.



	ID	NAME	AGE	ADDRESS	SALARY
1	1	Paul	32	California	20000.0
2	2	Allen	25	Texas	15000.0
3	3	Teddy	23	Norway	20000.0
4	4	Mark	25	Rich-Mond	65000.0

(4) SELECT Operation

Following Python program shows how to fetch and display records from the COMPANY table created in the above example.

Syntax

- SELECT column1, column2, columnN FROM table_name;
- SELECT * FROM table_name;

In [12]:

```
import sqlite3

conn = sqlite3.connect('test.db')
print "Opened database successfully";

cursor = conn.execute("SELECT id, name, address, salary from COMPANY")
for row in cursor:
   print "ID = ", row[0]
   print "NAME = ", row[1]
   print "ADDRESS = ", row[2]
   print "SALARY = ", row[3], "\n"

print "Operation done successfully";
conn.close()
```

```
Opened database successfully
ID =  1
NAME =  Paul
ADDRESS =  California
SALARY =  20000.0

ID =  2
NAME =  Allen
ADDRESS =  Texas
SALARY =  15000.0

ID =  3
NAME =  Teddy
ADDRESS =  Norway
SALARY =  20000.0

ID =  4
NAME =  Mark
ADDRESS =  Rich-Mond
SALARY =  65000.0

Operation done successfully
```

(5) UPDATE Operation

Following Python code shows how to use UPDATE statement to update any record and then fetch and display the updated records from the COMPANY table.

Syntax

```
UPDATE table_name
SET column1 = value1, column2 = value2...., columnN = valueN
WHERE [condition];
```

In [13]:

```
import sqlite3

conn = sqlite3.connect('test.db')
print "Opened database successfully";

conn.execute("UPDATE COMPANY set SALARY = 40000.00 where ID = 1")
conn.commit
print "Total number of rows updated :", conn.total_changes

cursor = conn.execute("SELECT id, name, address, salary from COMPANY")
for row in cursor:
   print "ID = ", row[0]
   print "NAME = ", row[1]
   print "ADDRESS = ", row[2]
   print "SALARY = ", row[3], "\n"

print "Operation done successfully";
conn.close()
```

```
Opened database successfully
Total number of rows updated : 1
ID =  1
NAME =  Paul
ADDRESS =  California
SALARY =  40000.0

ID =  2
NAME =  Allen
ADDRESS =  Texas
SALARY =  15000.0

ID =  3
NAME =  Teddy
ADDRESS =  Norway
SALARY =  20000.0

ID =  4
NAME =  Mark
ADDRESS =  Rich-Mond
SALARY =  65000.0

Operation done successfully
```

(6) DELETE Operation

Following Python code shows how to use DELETE statement to delete any record and then fetch and display the remaining records from the COMPANY table.

Syntax

```
DELETE FROM table_name
WHERE [condition];
```

In [14]:

```python
import sqlite3

conn = sqlite3.connect('test.db')
print "Opened database successfully";

conn.execute("DELETE from COMPANY where ID = 2;")
conn.commit()
print "Total number of rows deleted :", conn.total_changes

cursor = conn.execute("SELECT id, name, address, salary from COMPANY")
for row in cursor:
   print "ID = ", row[0]
   print "NAME = ", row[1]
   print "ADDRESS = ", row[2]
   print "SALARY = ", row[3], "\n"

print "Operation done successfully";
conn.close()
```

```
Opened database successfully
Total number of rows deleted : 1
ID =  1
NAME =  Paul
ADDRESS =  California
SALARY =  20000.0

ID =  3
NAME =  Teddy
ADDRESS =  Norway
SALARY =  20000.0

ID =  4
NAME =  Mark
ADDRESS =  Rich-Mond
SALARY =  65000.0

Operation done successfully
```

You can also check at DBbrowser.