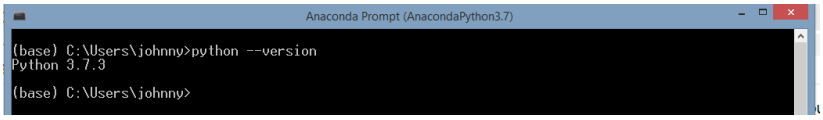
Lab Environment Setup:

1. We use Jupiter notebook installed on our PC. We used Python 3.7.3 version.



I. Python Code

We use SQLite database, where data are stored in table format. For example in our created 'students' SQLite table we work on following data:

first_name	last_name	email	year_born	grade
'Tim'	'Eshton'	'timmy@gmail.com'	1999	8.56
'John'	'Westly'	'john123@gmail.com'	1998	7.5
'Eshley'	'Tangton'	'eshleyyy@yahoo.com'	1995	8.14

Our Python script allows us to enter, modify, delete entries into this SQLite table. Python code is presented and explained bellow:

```
###############################################################################
#                                  PART I                                     #
#            We create database, and database table named 'students'          #
###############################################################################
```

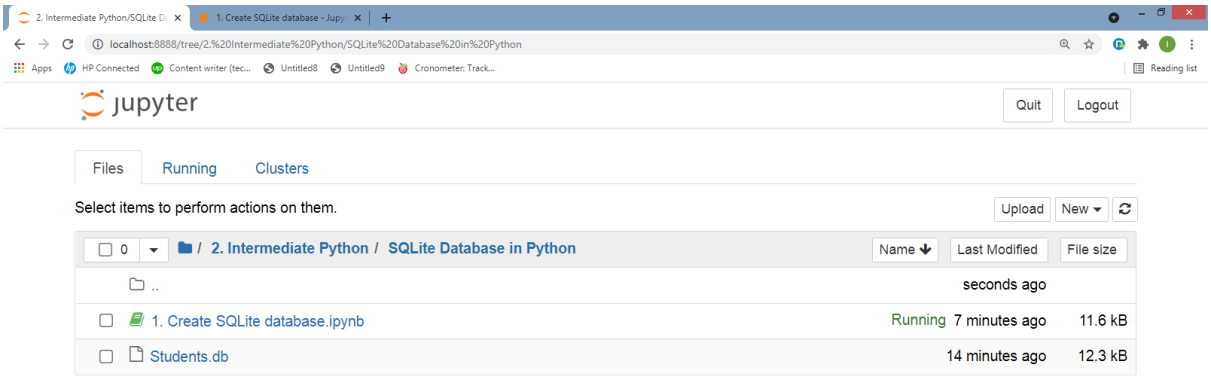
```
import sqlite3

# Create a database named 'Students.db' or connect to already existing database
conn = sqlite3.connect('Students.db')

# Create cursor
curs = conn.cursor()

# Create table named 'students' inside the database named 'Students.db'
curs.execute('''CREATE TABLE IF NOT EXISTS students
      (id integer PRIMARY KEY,
      first_name text NOT NULL,
      last_name text NOT NULL,
      email text NOT NULL,
      year_born integer NOT NULL,
      grade real NOT NULL);''')
```

We create SQLite database named 'Students.db'. We can see this database file bellow.



CREATE TABLE IF NOT EXISTS students - statement is used to create SQLite table if alreadt do not exists.
Inside this SQLite table we have columns with headers id, first_name, last_name, email, year_born, grade. For any of them we define if entries into the table will be from data format: text, integer, real number.
NOT NULL - we want all entries to be with data, not empty
PRIMARY KEY - used for SQLite database relationships or other things, which in this Python code are not used.

```
###############################################################################################
#                                        PART II                                              #
#   We create support functions to work with SQLite database                                  #
#                                                                                             #
#   - to enter data into the table  (Function 1)                                              #
#   - to see what data table 'students' have (Function 2)                                     #
#   - to delete row from the table 'students' (Function 3)                                    #
#   - to edit row data from the table 'students' (Function 4)                                 #
#   - to delete duplicate row data stored in the table 'students' (Function 5)                #
#   - to delete whole SQLite table permanently (Function 6)                                   #
#                                                                                             #
###############################################################################################
```

```
# (Function 1) - For populating database with data
def Insert(first_name,last_name,email,year_born,grade):

    conn = sqlite3.connect('Students.db')
    curs = conn.cursor()

    # SQLite statement for - Insert data Into Table
    Insert_Parameters="""INSERT INTO students
       (first_name,last_name,email,year_born,grade) VALUES
       (?, ?, ?, ?, ?);"""

    data_tuple = (first_name,last_name,email,year_born,grade)

    curs.execute(Insert_Parameters,data_tuple)

    conn.commit()
    conn.close()
```

```
# (Function 2) - For seeing database data
def StudentsData():

    conn = sqlite3.connect('Students.db')
    curs = conn.cursor()

    # Find what data in the rows we want to present on the output
    data_values = curs.execute("SELECT first_name,last_name,email,year_born,grade FROM students").fetchall()

    #We print data - row by row
    for data_value in data_values:
        print(data_value)

    conn.commit()
    conn.close()


# (Function 3)
def DeleteTableData():
    conn = sqlite3.connect('Students.db')
    curs = conn.cursor()

    # SQLite statement for - Deleting row records where students are born 1999
    DeleteData = """DELETE FROM students WHERE year_born = 1999"""
    curs.execute(DeleteData)

    conn.commit()
    conn.close()


# (Function 4)
def UpdateRowData():

    conn = sqlite3.connect('Students.db')
    curs = conn.cursor()

    # SQLite statement for - Updating row 'grade' record on the base of student email data value
    UpdateData = """UPDATE students SET grade = 8.65 WHERE email = 'timmy@gmail.com'"""
    curs.execute(UpdateData)

    conn.commit()
    conn.close()


# (Function 5)
def UniqueRowData():

    conn = sqlite3.connect('Students.db')
    curs = conn.cursor()

    # Find duplicate rows data
    curs.execute("SELECT * FROM students \
            GROUP BY first_name,last_name,email \
            HAVING COUNT(*) > 1;")

    # Find id's of duplicate rows and remove duplicate rows data
    for row in curs.fetchall():
        print(row[0])
        sql_query = "DELETE FROM students WHERE id = {}".format(row[0])
        curs.execute(sql_query)
```

```
    conn.commit()
    conn.close()


# (Function 6)
def DeleteTable():
    conn = sqlite3.connect('Students.db')
    curs = conn.cursor()

    # SQLite statement for - deleting table 'students'
    TableDelete = "DROP TABLE IF EXISTS students"
    curs.execute(TableDelete)

    conn.commit()
    conn.close()
```

Any of these functions are separate code if we look them from a side. So, because of this we need to do the process of opening, access and close the database in any of these functions independently.

- In function 3 SQLite statement: DELETE FROM students WHERE year_born = 1999 is used to delete row entries in the students SQLite table which have item 1999 in the position of the column with header year_born. We can have any other examples like: DELETE FROM students WHERE first_name = 'Tim' or DELETE FROM students WHERE email = 'eshleyyy@yahoo.com' etc.

- In function 4 SQLite statement: UPDATE students SET grade = 8.65 WHERE email = 'timmy@gmail.com' we are updating value of the variable grade on the positions in SQLite table where email value 'timmy@gmail.com' is present. We can have any other examples like: UPDATE students SET email = 'tim2021@gmail.com' WHERE last_name= ' Eshton' etc.

- In function 5 for cycle variable row will present all data in a SQLite table row. As an example:

(21, 'Eshley', 'Tangton', 'eshleyyy@yahoo.com', 1995, 8.14)

But if instead in print function we use row[0], then it will be printed just 21, because this element is on the position [0].

curs.fetchall() will give duplicate rows, and with SQLite statement "DELETE FROM students WHERE id = {}".format(row[0]) we will delete one by one all these entries. "Some text data plus {} place for a variable value ".format(variable) is concept to insert variable value into the text. In this case this SQLite statement is repeatably used inside the ***for*** cycle, with different values, as variable values change in any independent loop.

```
##############################################################################
#                                 PART III                                   #
#   Use functions to enter, edit, delete data, delete tables, on the base of needs  #
##############################################################################
```

```
Insert('Tim','Eshton','timmy@gmail.com',1999,8.56)
Insert('John','Westly','john123@gmail.com',1998,7.50)
Insert('Eshley','Tangton','eshleyyy@yahoo.com',1995,8.14)

#We can see data records of table 'students'
#StudentsData()

#Used to delete row in table 'students' for students born 1999
#DeleteTableData()

#We permanently delete whole table 'students'
#DeleteTable()

#Identify row id's and remove duplicate row data
UniqueRowData()

#We can edit row table entry on the base of identifying the row with specific email value
UpdateRowData()

#We can see data records of table 'students'
StudentsData()
```

In this part we are executing all the actions in this Python code, activating one, or multiple functions, on the basre what we want to achieve. As an example we have 3 entries inserted in our students SQLite table, with Insert(...) function.

```
#############################################################################################
#                                        PART IV                                            #
#    We have continuous access to the database, and finally we close the database          #
#############################################################################################
```

```
#Save work changes related to the database
conn.commit()

# Close Connection to the database
conn.close()
```

This part must be at the end, otherwise when we call our functions we need open connection to the database, and at the end we need to close this connection.

II. How this code works

Example 1: If we want to see what is on the output if we insert just data:

```
142
143 Insert('Tim','Eshton','timmy@gmail.com',1999,8.56)
144 Insert('John','Westly','john123@gmail.com',1998,7.50)
145 Insert('Eshley','Tangton','eshleyyy@yahoo.com',1995,8.14)
146
147 #We can see data records of table 'students'
148 #StudentsData()
149
150 #Used to delete row in table 'students' for students born 1999
151 #DeleteTableData()
152
153 #We permanently delete whole table 'students'
154 #DeleteTable()
155
156 #Identify row id's and remove duplicate row data
157 #UniqueRowData()
158
159 #We can edit row table entry on the base of identifying the row with specific email value
160 #UpdateRowData()
161
162 #We can see data records of table 'students'
163 StudentsData()
```

output will be:

```
('Tim', 'Eshton', 'timmy@gmail.com', 1999, 8.56)
('John', 'Westly', 'john123@gmail.com', 1998, 7.5)
('Eshley', 'Tangton', 'eshleyyy@yahoo.com', 1995, 8.14)
```

But, what if we need this Python script to be repeatably executed in automation manner? Let say if we run this code again, output will be:

```
('Tim', 'Eshton', 'timmy@gmail.com', 1999, 8.56)
('John', 'Westly', 'john123@gmail.com', 1998, 7.5)
('Eshley', 'Tangton', 'eshleyyy@yahoo.com', 1995, 8.14)
('Tim', 'Eshton', 'timmy@gmail.com', 1999, 8.56)
('John', 'Westly', 'john123@gmail.com', 1998, 7.5)
('Eshley', 'Tangton', 'eshleyyy@yahoo.com', 1995, 8.14)
```

We can see from second output that now we have duplicate entries inside this SQLite table. If we run this script 100 times, then we will have 100 duplicate entries, which will grow this table HDD space occupance. That is not good !

Example 2: Now because our database has duplicated entries we will delete it permanently using DeleteTable() function and we will start all over again:

```
Insert('Tim','Eshton','timmy@gmail.com',1999,8.56)
Insert('John','Westly','john123@gmail.com',1998,7.50)
Insert('Eshley','Tangton','eshleyyy@yahoo.com',1995,8.14)

#We can see data records of table 'students'
#StudentsData()

#Used to delete row in table 'students' for students born 1999
#DeleteTableData()

#We permanently delete whole table 'students'
DeleteTable()

#Identify row id's and remove duplicate row data
#UniqueRowData()

#We can edit row table entry on the base of identifying the row with specific email value
#UpdateRowData()

#We can see data records of table 'students'
StudentsData()
```

Now we will execute the script again and we can see that error was given and that table 'students' do not exist, which was our intention. We have deleted all entries inside including SQLite table as well. *NOTE:* Be careful when use this type of DeleteTable() functions because it deletes everyting and all data entered in the table !!!

```
---------------------------------------------------------------------------
OperationalError                          Traceback (most recent call last)
<ipython-input-49-67b03a2cca1b> in <module>
    160
    161 #We can see data records of table 'students'
--> 162 StudentsData()
    163
    164

<ipython-input-49-67b03a2cca1b> in StudentsData()
     63
     64     # Find what data in the rows we want to present on the output
---> 65     data_values = curs.execute("SELECT first_name,last_name,email,year_born,grade FROM studen
ts").fetchall()
     66
     67     #We print data - row by row

OperationalError: no such table: students
```

Example 3: Now we will use UniqueRowData() function to see if there will be duplicate entries:

```
142
143 Insert('Tim','Eshton','timmy@gmail.com',1999,8.56)
144 Insert('John','Westly','john123@gmail.com',1998,7.50)
145 Insert('Eshley','Tangton','eshleyyy@yahoo.com',1995,8.14)
146
147 #We can see data records of table 'students'
148 #StudentsData()
149
150 #Used to delete row in table 'students' for students born 1999
151 #DeleteTableData()
152
153 #We permanently delete whole table 'students'
154 #DeleteTable()
155
156 #Identify row id's and remove duplicate row data
157 UniqueRowData()
158
159 #We can edit row table entry on the base of identifying the row with specific email value
160 #UpdateRowData()
161
162 #We can see data records of table 'students'
163 StudentsData()
164
```

If we execute the code first time we will get just 3 entry as we have done with Insert(...) function:

```
('Tim', 'Eshton', 'timmy@gmail.com', 1999, 8.56)
('John', 'Westly', 'john123@gmail.com', 1998, 7.5)
('Eshley', 'Tangton', 'eshleyyy@yahoo.com', 1995, 8.14)
```

If we execute this code for a second time we will get following output:

```
3
2
1
('Tim', 'Eshton', 'timmy@gmail.com', 1999, 8.56)
('John', 'Westly', 'john123@gmail.com', 1998, 7.5)
('Eshley', 'Tangton', 'eshleyyy@yahoo.com', 1995, 8.14)
```

We can see UniqueRowData() function has found that duplicate rows are with id's 3, 2 and 1 and all of them were removed from the SQL table. This means we can run this script multiple times, but we are entering inside the SQLite database just unique entries, not the duplicates.

Example 4: If we want to update SQLite table specific data we can do that using UpdateRowData() function. In this case we use twice StudentsData() function to be able to see what was previous value of the row entry and lates value after we make changes:

```
142
143 Insert('Tim','Eshton','timmy@gmail.com',1999,8.56)
144 Insert('John','Westly','john123@gmail.com',1998,7.50)
145 Insert('Eshley','Tangton','eshleyyy@yahoo.com',1995,8.14)
146
147 #We can see data records of table 'students'
148 StudentsData()
149
150 #Used to delete row in table 'students' for students born 1999
151 #DeleteTableData()
152
153 #We permanently delete whole table 'students'
154 #DeleteTable()
155
156 #Identify row id's and remove duplicate row data
157 UniqueRowData()
158
159 #We can edit row table entry on the base of identifying the row with specific email value
160 UpdateRowData()
161
162 #We can see data records of table 'students'
163 StudentsData()
```

In our cadse output will be:

```
('Tim', 'Eshton', 'timmy@gmail.com', 1999, 8.56)
('John', 'Westly', 'john123@gmail.com', 1998, 7.5)
('Eshley', 'Tangton', 'eshleyyy@yahoo.com', 1995, 8.14)
('Tim', 'Eshton', 'timmy@gmail.com', 1999, 8.56)
('John', 'Westly', 'john123@gmail.com', 1998, 7.5)
('Eshley', 'Tangton', 'eshleyyy@yahoo.com', 1995, 8.14)
6
5
4
('Tim', 'Eshton', 'timmy@gmail.com', 1999, 8.65)
('John', 'Westly', 'john123@gmail.com', 1998, 7.5)
('Eshley', 'Tangton', 'eshleyyy@yahoo.com', 1995, 8.14)
```

In the first row entry grade value has been changed from 8.56 to 8.65 as we can see.

NOTE: This output also shows that because we have executed again this script at the input of the UniqueRowData() function we have duplicate data (each row has one additional duplicate)

Example 5: We can also on the base of identification to delete some row entry from the SQLite table using DeleteTableData() function.

```
142
143 Insert('Tim','Eshton','timmy@gmail.com',1999,8.56)
144 Insert('John','Westly','john123@gmail.com',1998,7.50)
145 Insert('Eshley','Tangton','eshleyyy@yahoo.com',1995,8.14)
146
147 #We can see data records of table 'students'
148 #StudentsData()
149
150 #Used to delete row in table 'students' for students born 1999
151 DeleteTableData()
152
153 #We permanently delete whole table 'students'
154 #DeleteTable()
155
156 #Identify row id's and remove duplicate row data
157 UniqueRowData()
158
159 #We can edit row table entry on the base of identifying the row with specific email value
160 #UpdateRowData()
161
162 #We can see data records of table 'students'
163 StudentsData()
```

Output from this action will be deleting the first from the 3 rows inside our 'students' SQLite table:

```
9
8
('John', 'Westly', 'john123@gmail.com', 1998, 7.5)
('Eshley', 'Tangton', 'eshleyyy@yahoo.com', 1995, 8.14)
```