Homework B

Software Testing Methodologies

Assessing a potential software choice

Here, we are using TinyDB which is used to store data. TinyDB is more or the less like a database. We are using TinyDB using the Python packages. TinyDB is responsible for creation of the data base which is in the format of json *(*.json),* and also the TinyDB is used as small database which would be more efficient in storing less data rather storing more data which is not recommendable. It is also called as mini-database, and its targets are small applications with no much data and those which have blown by a SQL-DB or an external database server. The development of TinyDB is made in python purely without any external dependencies.

TinyDB is a query processing system which is used for extracting information from a network of TinyOS sensors. TinyDB will be providing the simple SQL-like interface to specify the data what we want to extract, along with the parameters like the rate at which data should be refreshed – much as we would pose queries against a traditional database.

The reasons why we use TinyDB are as follows:

a) Tiny

b) Document oriented

c) Optimized for our happiness

d) Written in pure python

e) Works on python 2.6 – 3.5 and PyPy

f) Powerfully extensible

g) 100% test coverage

One of the example for the data which can be stored on TinyDB is small store customer database. We have database which will have the customer names including first and last name, phone number (which is unique), number of items. This type of data can be stored on the TinyDB for processing and further analysis.

Is using TinyDB easy than SQLite?

I prefer to answer in this manner, TinyDB is very easy to use, simple to compute too. The main reason behind this answer is that, TinyDB uses json files as the database files. Usage of json is wider than anything because, json files are simple to use and easy to compute. The run time for running any query would be less when compared to run time running a query in SQLite.

INITIAL REPORT

Test Plan

Test plan is the document which haves the detailed description of objectives, target market and processes for a specific beta test for a software or hardware product. The plan typically contains a detailed understanding of eventual workflow.

Test plan will be prepared by the testers itself in-order to verify that all the requirements are mentioned in the test plan clearly. Test plan document formats can be as varied as the products and organizations to which they apply. There are 3 major elements that should be described in the test plan: Test Coverage, Test methods, Test Responsibilities.

Test Coverage: Test coverage in the test plan states what requirements will be verified during what stages of the product life.

Test Methods: Test methods in the test plan state how test coverage will be implemented.

Test Responsibilities: Test responsibilities include what organizations will perform the test methods and at each stage of the product life.

Here, the test plan would be that we have considered a customer database such that it has certain fields like {first-name last-name, phone-number, number-of-items}. We have to create a database file in json file by writing queries to the automation test-cases. We are creating a json file which will have 500 records of the customers with all the 3 fields for every customer. Point which is noted is, phone-number field is unique for every customer. We will be creating a query which does the automation of unique phone numbers, and combination of last-name and first-name. All the records will be generated and saved in to the file *(db.json)*. This file is the database file of the customers for which we've created from the python code by writing the automation test cases. After generating this file, we have to perform operations on the existing json file and the changes made on the existing json file should reflect in the new json.

The operations we may perform the existing json file are as follows:

a) Searching for a particular customer based on one of the fields
b) Inserting a new record of a customer in to existing json file
c) Deleting an existing record of a customer from the json file
d) Updating existing record of a customer in the json file.

Test Methodology

Software testing methodology deals with the practical ideas and proven practices which help in efficient software project management. The software testing methodologies are discussed below:

a) Waterfall model

b) Iterative development

c) Agile methodology

d) Extreme programming

Test methodology is describing the strategy for testing. When planning your methodology, consider:

a) Where will the testing takes place?
b) Who will perform the tests?
c) How will you communicate with and involve participants?
d) How will you schedule the testing?
e) How will you manage application problems?

Prototype of a test

Prototyping tools and testing is one of the way to see and test your website before we spend long nights coding and programming. Although the website design process and mockup tools tends to be relatively fluid, the prototyping phase typically focuses on:

a) Visual layout

b) Interface element design

c) Logical flow

d) Behavior

CONCLUSION REPORT

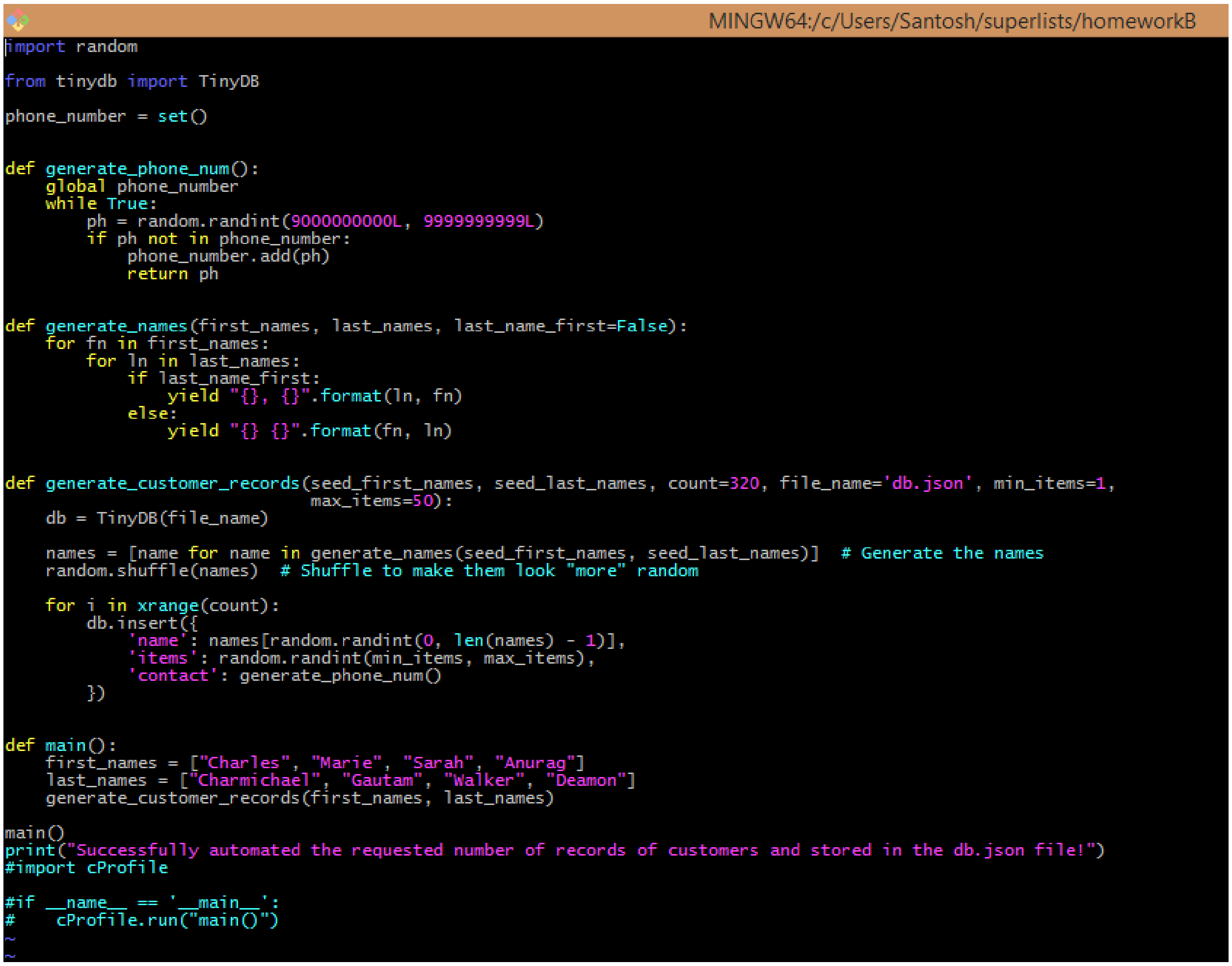
Automation Test Cases:

A) TINYDB

We have written a python code which will generate the db.json with the requested number of records in the test case. The data file will consists of a customer with ***(items; phone-number; first-name last-name).*** Now, we have will consider that data of customers and perform operations on it say like search, update, delete, insert. Any operation performed on the file would reflect in the file itself.

Test cases which are written for the records to generate automatically and get stored in the db.json file.

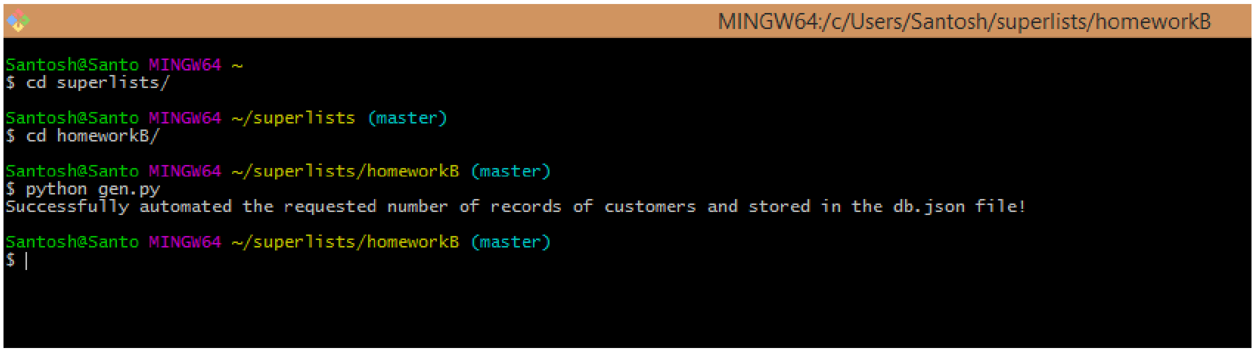
Program:



MINGW64:/c/Users/Santosh/superlists/homeworkB

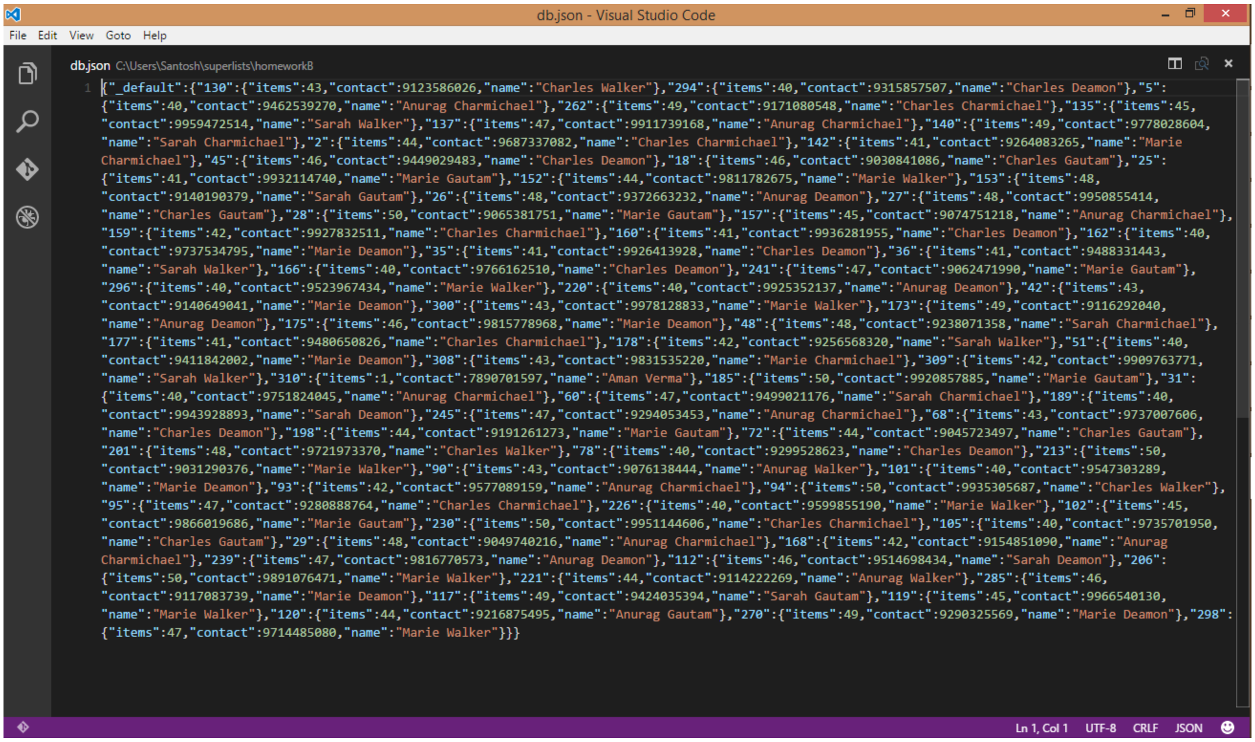
```python
import random

from tinydb import TinyDB

phone_number = set()


def generate_phone_num():
    global phone_number
    while True:
        ph = random.randint(9000000000L, 9999999999L)
        if ph not in phone_number:
            phone_number.add(ph)
            return ph


def generate_names(first_names, last_names, last_name_first=False):
    for fn in first_names:
        for ln in last_names:
            if last_name_first:
                yield "{}, {}".format(ln, fn)
            else:
                yield "{} {}".format(fn, ln)


def generate_customer_records(seed_first_names, seed_last_names, count=320, file_name='db.json', min_items=1,
                              max_items=50):
    db = TinyDB(file_name)

    names = [name for name in generate_names(seed_first_names, seed_last_names)]  # Generate the names
    random.shuffle(names)  # Shuffle to make them look "more" random

    for i in xrange(count):
        db.insert({
            'name': names[random.randint(0, len(names) - 1)],
            'items': random.randint(min_items, max_items),
            'contact': generate_phone_num()
        })


def main():
    first_names = ["Charles", "Marie", "Sarah", "Anurag"]
    last_names = ["Charmichael", "Gautam", "Walker", "Deamon"]
    generate_customer_records(first_names, last_names)

main()
print("Successfully automated the requested number of records of customers and stored in the db.json file!")
#import cProfile

#if __name__ == '__main__':
#    cProfile.run("main()")
~
~
```

Output:



```
Santosh@Santo MINGW64 ~
$ cd superlists/

Santosh@Santo MINGW64 ~/superlists (master)
$ cd homeworkB/

Santosh@Santo MINGW64 ~/superlists/homeworkB (master)
$ python gen.py
Successfully automated the requested number of records of customers and stored in the db.json file!

Santosh@Santo MINGW64 ~/superlists/homeworkB (master)
$ |
```

Db.json

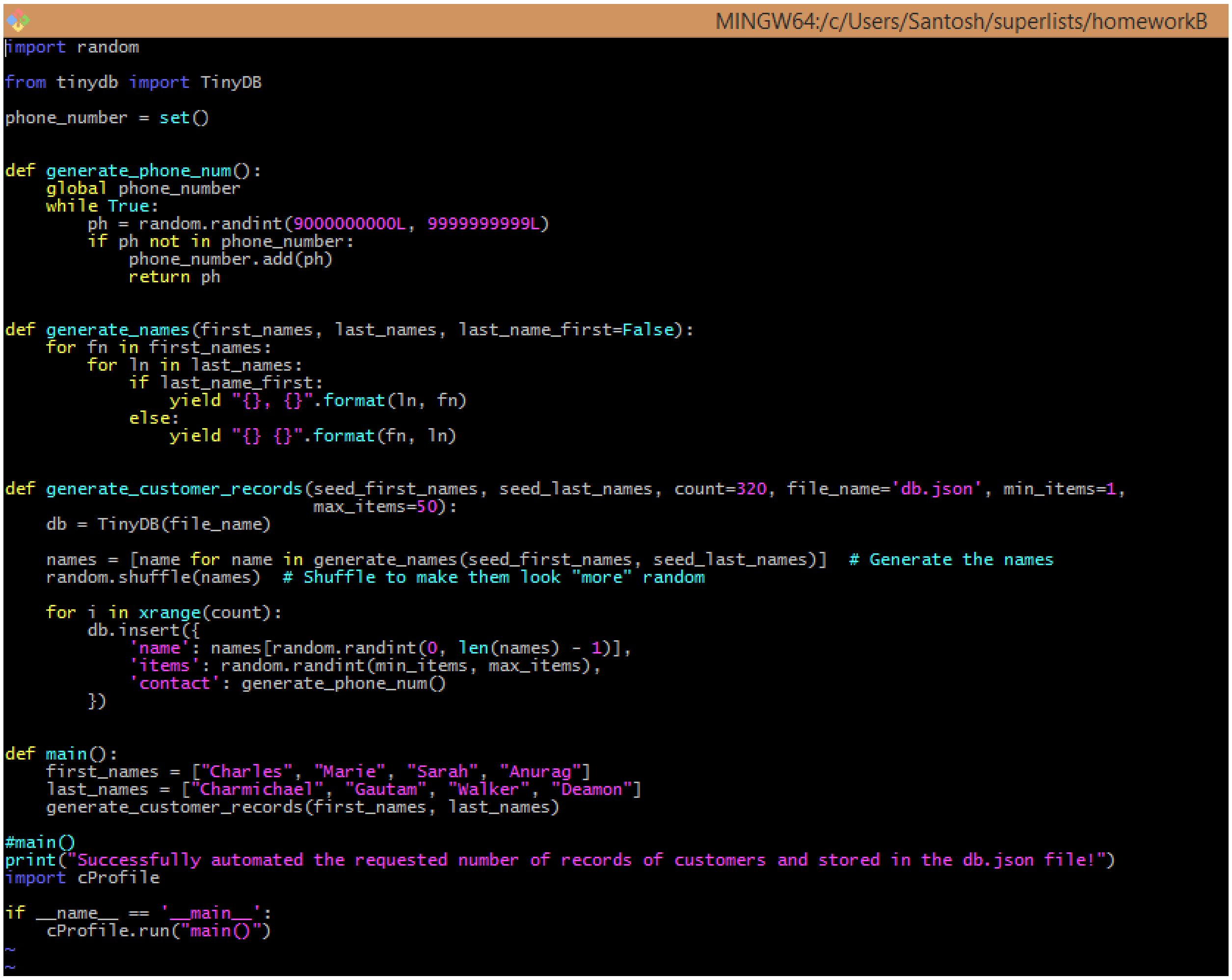

db.json - Visual Studio Code
File Edit View Goto Help
db.json C:\Users\Santosh\superlists\homeworkB
Ln 1, Col 1 UTF-8 CRLF JSON

```
{"_default":{"130":{"items":43,"contact":9123586026,"name":"Charles Walker"},"294":{"items":40,"contact":9315857507,"name":"Charles Deamon"},"5":
{"items":40,"contact":9462539270,"name":"Anurag Charmichael"},"262":{"items":49,"contact":9171080548,"name":"Charles Charmichael"},"135":{"items":45,
"contact":9959472514,"name":"Sarah Walker"},"137":{"items":47,"contact":9911739168,"name":"Anurag Charmichael"},"140":{"items":49,"contact":9778028604,
"name":"Sarah Charmichael"},"2":{"items":44,"contact":9687337082,"name":"Charles Charmichael"},"142":{"items":41,"contact":9264083265,"name":"Marie
Charmichael"},"45":{"items":46,"contact":9449029483,"name":"Charles Deamon"},"18":{"items":46,"contact":9030841086,"name":"Charles Gautam"},"25":
{"items":41,"contact":9932114740,"name":"Marie Gautam"},"152":{"items":44,"contact":9811782675,"name":"Marie Walker"},"153":{"items":48,
"contact":9140190379,"name":"Sarah Gautam"},"26":{"items":48,"contact":9372663232,"name":"Anurag Deamon"},"27":{"items":48,"contact":9950855414,
"name":"Charles Gautam"},"28":{"items":50,"contact":9065381751,"name":"Marie Gautam"},"157":{"items":45,"contact":9074751218,"name":"Anurag Charmichael"},
"159":{"items":42,"contact":9927832511,"name":"Charles Charmichael"},"160":{"items":41,"contact":9936281955,"name":"Charles Deamon"},"162":{"items":40,
"contact":9737534795,"name":"Marie Deamon"},"35":{"items":41,"contact":9926413928,"name":"Charles Deamon"},"36":{"items":41,"contact":9488331443,
"name":"Sarah Walker"},"166":{"items":40,"contact":9766162510,"name":"Charles Deamon"},"241":{"items":47,"contact":9062471990,"name":"Marie Gautam"},
"296":{"items":40,"contact":9523967434,"name":"Marie Walker"},"220":{"items":40,"contact":9925352137,"name":"Anurag Deamon"},"42":{"items":43,
"contact":9140649041,"name":"Marie Deamon"},"300":{"items":43,"contact":9978128833,"name":"Marie Walker"},"173":{"items":49,"contact":9116292040,
"name":"Anurag Deamon"},"175":{"items":46,"contact":9815778968,"name":"Marie Deamon"},"48":{"items":48,"contact":9238071358,"name":"Sarah Charmichael"},
"177":{"items":41,"contact":9480650826,"name":"Charles Charmichael"},"178":{"items":42,"contact":9256568320,"name":"Sarah Walker"},"51":{"items":40,
"contact":9411842002,"name":"Marie Deamon"},"308":{"items":43,"contact":9831535220,"name":"Marie Charmichael"},"309":{"items":42,"contact":9909763771,
"name":"Sarah Walker"},"310":{"items":1,"contact":7890701597,"name":"Aman Verma"},"185":{"items":50,"contact":9920857885,"name":"Marie Gautam"},"31":
{"items":40,"contact":9751824045,"name":"Anurag Charmichael"},"60":{"items":47,"contact":9499021176,"name":"Sarah Charmichael"},"189":{"items":40,
"contact":9943928893,"name":"Sarah Deamon"},"245":{"items":47,"contact":9294053453,"name":"Anurag Charmichael"},"68":{"items":43,"contact":9737007606,
"name":"Charles Deamon"},"198":{"items":44,"contact":9191261273,"name":"Marie Gautam"},"72":{"items":44,"contact":9045723497,"name":"Charles Gautam"},
"201":{"items":48,"contact":9721973370,"name":"Charles Walker"},"78":{"items":40,"contact":9299528623,"name":"Charles Deamon"},"213":{"items":50,
"contact":9031290376,"name":"Marie Walker"},"90":{"items":43,"contact":9076138444,"name":"Anurag Walker"},"101":{"items":40,"contact":9547303289,
"name":"Marie Deamon"},"93":{"items":42,"contact":9577089159,"name":"Anurag Charmichael"},"94":{"items":50,"contact":9935305687,"name":"Charles Walker"},
"95":{"items":47,"contact":9280888764,"name":"Charles Charmichael"},"226":{"items":40,"contact":9599855190,"name":"Marie Walker"},"102":{"items":45,
"contact":9866019686,"name":"Marie Gautam"},"230":{"items":50,"contact":9951144606,"name":"Charles Charmichael"},"105":{"items":40,"contact":9735701950,
"name":"Charles Gautam"},"29":{"items":48,"contact":9049740216,"name":"Anurag Charmichael"},"168":{"items":42,"contact":9154851090,"name":"Anurag
Charmichael"},"239":{"items":47,"contact":9816770573,"name":"Anurag Deamon"},"112":{"items":46,"contact":9514698434,"name":"Sarah Deamon"},"206":
{"items":50,"contact":9891076471,"name":"Marie Walker"},"221":{"items":44,"contact":9114222269,"name":"Anurag Walker"},"285":{"items":46,
"contact":9117083739,"name":"Marie Deamon"},"117":{"items":49,"contact":9424035394,"name":"Sarah Gautam"},"119":{"items":45,"contact":9966540130,
"name":"Marie Walker"},"120":{"items":44,"contact":9216875495,"name":"Anurag Gautam"},"270":{"items":49,"contact":9290325569,"name":"Marie Deamon"},"298":
{"items":47,"contact":9714485080,"name":"Marie Walker"}}}
```

We can also find the time taken by the test cases to run in TinyDB using profiling

We “import cProfile” for this function to be activated.

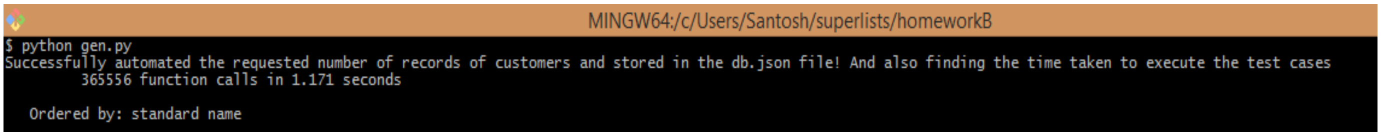
Program:



MINGW64:/c/Users/Santosh/superlists/homeworkB

```
import random

from tinydb import TinyDB

phone_number = set()


def generate_phone_num():
    global phone_number
    while True:
        ph = random.randint(9000000000L, 9999999999L)
        if ph not in phone_number:
            phone_number.add(ph)
            return ph


def generate_names(first_names, last_names, last_name_first=False):
    for fn in first_names:
        for ln in last_names:
            if last_name_first:
                yield "{}, {}".format(ln, fn)
            else:
                yield "{} {}".format(fn, ln)


def generate_customer_records(seed_first_names, seed_last_names, count=320, file_name='db.json', min_items=1,
                              max_items=50):
    db = TinyDB(file_name)

    names = [name for name in generate_names(seed_first_names, seed_last_names)]  # Generate the names
    random.shuffle(names)  # Shuffle to make them look "more" random

    for i in xrange(count):
        db.insert({
            'name': names[random.randint(0, len(names) - 1)],
            'items': random.randint(min_items, max_items),
            'contact': generate_phone_num()
        })


def main():
    first_names = ["Charles", "Marie", "Sarah", "Anurag"]
    last_names = ["Charmichael", "Gautam", "Walker", "Deamon"]
    generate_customer_records(first_names, last_names)

#main()
print("Successfully automated the requested number of records of customers and stored in the db.json file!")
import cProfile

if __name__ == '__main__':
    cProfile.run("main()")
~
~
```

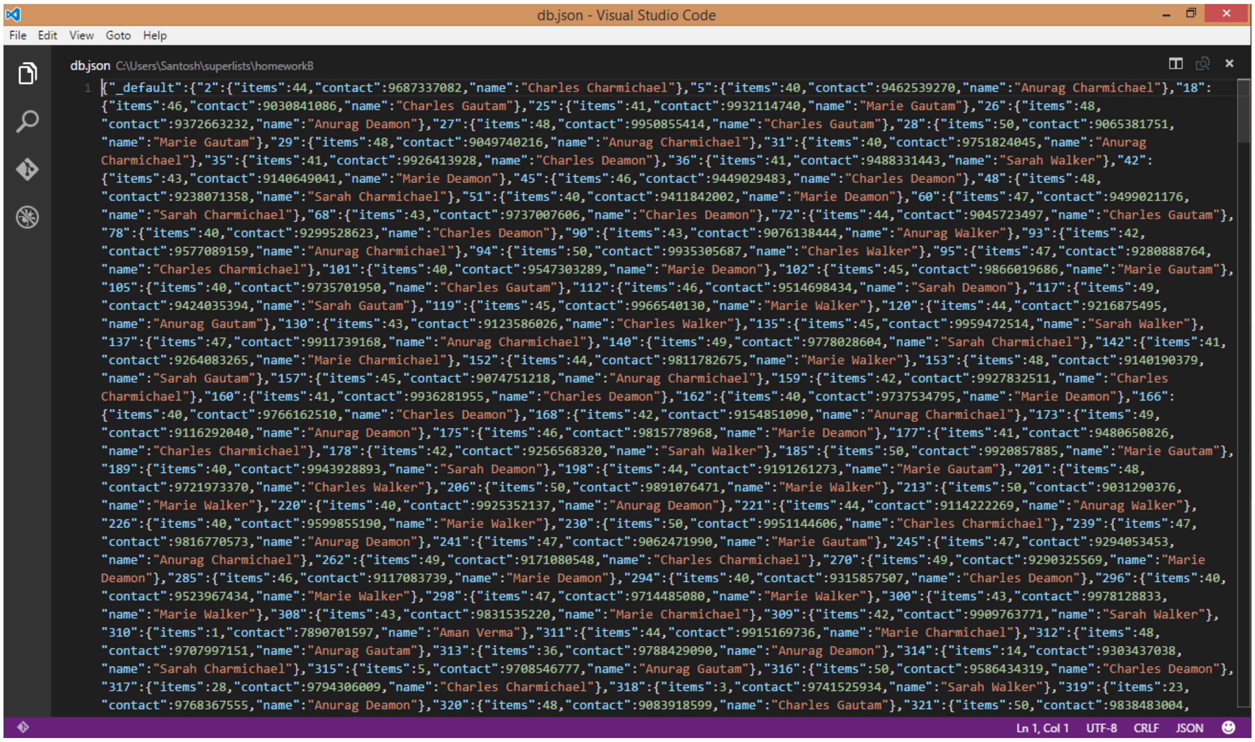
Output:



MINGW64:/c/Users/Santosh/superlists/homeworkB

```
$ python gen.py
Successfully automated the requested number of records of customers and stored in the db.json file! And also finding the time taken to execute the test cases
         365556 function calls in 1.171 seconds

   Ordered by: standard name
```

Db.json


db.json - Visual Studio Code
File
Edit
View
Goto
Help
db.json C:\Users\Santosh\superlists\homeworkB
Ln 1, Col 1
UTF-8
CRLF
JSON

```
{"_default":{"2":{"items":44,"contact":9687337082,"name":"Charles Charmichael"},"5":{"items":40,"contact":9462539270,"name":"Anurag Charmichael"},"18":
{"items":46,"contact":9030841086,"name":"Charles Gautam"},"25":{"items":41,"contact":9932114740,"name":"Marie Gautam"},"26":{"items":48,
"contact":9372663232,"name":"Anurag Deamon"},"27":{"items":48,"contact":9950855414,"name":"Charles Gautam"},"28":{"items":50,"contact":9065381751,
"name":"Marie Gautam"},"29":{"items":48,"contact":9049740216,"name":"Anurag Charmichael"},"31":{"items":40,"contact":9751824045,"name":"Anurag
Charmichael"},"35":{"items":41,"contact":9926413928,"name":"Charles Deamon"},"36":{"items":41,"contact":9488331443,"name":"Sarah Walker"},"42":
{"items":43,"contact":9140649041,"name":"Marie Deamon"},"45":{"items":46,"contact":9449029483,"name":"Charles Deamon"},"48":{"items":48,
"contact":9238071358,"name":"Sarah Charmichael"},"51":{"items":40,"contact":9411842002,"name":"Marie Deamon"},"60":{"items":47,"contact":9499021176,
"name":"Sarah Charmichael"},"68":{"items":43,"contact":9737007606,"name":"Charles Deamon"},"72":{"items":44,"contact":9045723497,"name":"Charles Gautam"},
"78":{"items":40,"contact":9299528623,"name":"Charles Deamon"},"90":{"items":43,"contact":9076138444,"name":"Anurag Walker"},"93":{"items":42,
"contact":9577089159,"name":"Anurag Charmichael"},"94":{"items":50,"contact":9935305687,"name":"Charles Walker"},"95":{"items":47,"contact":9280888764,
"name":"Charles Charmichael"},"101":{"items":40,"contact":9547303289,"name":"Marie Deamon"},"102":{"items":45,"contact":9866019686,"name":"Marie Gautam"},
"105":{"items":40,"contact":9735701950,"name":"Charles Gautam"},"112":{"items":46,"contact":9514698434,"name":"Sarah Deamon"},"117":{"items":49,
"contact":9424035394,"name":"Sarah Gautam"},"119":{"items":45,"contact":9966540130,"name":"Marie Walker"},"120":{"items":44,"contact":9216875495,
"name":"Anurag Gautam"},"130":{"items":43,"contact":9123586026,"name":"Charles Walker"},"135":{"items":45,"contact":9959472514,"name":"Sarah Walker"},
"137":{"items":47,"contact":9911739168,"name":"Anurag Charmichael"},"140":{"items":49,"contact":9778028604,"name":"Sarah Charmichael"},"142":{"items":41,
"contact":9264083265,"name":"Marie Charmichael"},"152":{"items":44,"contact":9811782675,"name":"Marie Walker"},"153":{"items":48,"contact":9140190379,
"name":"Sarah Gautam"},"157":{"items":45,"contact":9074751218,"name":"Anurag Charmichael"},"159":{"items":42,"contact":9927832511,"name":"Charles
Charmichael"},"160":{"items":41,"contact":9936281955,"name":"Charles Deamon"},"162":{"items":40,"contact":9737534795,"name":"Marie Deamon"},"166":
{"items":40,"contact":9766162510,"name":"Charles Deamon"},"168":{"items":42,"contact":9154851090,"name":"Anurag Charmichael"},"173":{"items":49,
"contact":9116292040,"name":"Anurag Deamon"},"175":{"items":46,"contact":9815778968,"name":"Marie Deamon"},"177":{"items":41,"contact":9480650826,
"name":"Charles Charmichael"},"178":{"items":42,"contact":9256568320,"name":"Sarah Walker"},"185":{"items":50,"contact":9920857885,"name":"Marie Gautam"},
"189":{"items":40,"contact":9943928893,"name":"Sarah Deamon"},"198":{"items":44,"contact":9191261273,"name":"Marie Gautam"},"201":{"items":48,
"contact":9721973370,"name":"Charles Walker"},"206":{"items":50,"contact":9891076471,"name":"Marie Walker"},"213":{"items":50,"contact":9031290376,
"name":"Marie Walker"},"220":{"items":40,"contact":9925352137,"name":"Anurag Deamon"},"221":{"items":44,"contact":9114222269,"name":"Anurag Walker"},
"226":{"items":40,"contact":9599855190,"name":"Marie Walker"},"230":{"items":50,"contact":9951144606,"name":"Charles Charmichael"},"239":{"items":47,
"contact":9816770573,"name":"Anurag Deamon"},"241":{"items":47,"contact":9062471990,"name":"Marie Gautam"},"245":{"items":47,"contact":9294053453,
"name":"Anurag Charmichael"},"262":{"items":49,"contact":9171080548,"name":"Charles Charmichael"},"270":{"items":49,"contact":9290325569,"name":"Marie
Deamon"},"285":{"items":46,"contact":9117083739,"name":"Marie Deamon"},"294":{"items":40,"contact":9315857507,"name":"Charles Deamon"},"296":{"items":40,
"contact":9523967434,"name":"Marie Walker"},"298":{"items":47,"contact":9714485080,"name":"Marie Walker"},"300":{"items":43,"contact":9978128833,
"name":"Marie Walker"},"308":{"items":43,"contact":9831535220,"name":"Marie Charmichael"},"309":{"items":42,"contact":9909763771,"name":"Sarah Walker"},
"310":{"items":1,"contact":7890701597,"name":"Aman Verma"},"311":{"items":44,"contact":9915169736,"name":"Marie Charmichael"},"312":{"items":48,
"contact":9707997151,"name":"Anurag Gautam"},"313":{"items":36,"contact":9788429090,"name":"Anurag Deamon"},"314":{"items":14,"contact":9303437038,
"name":"Sarah Charmichael"},"315":{"items":5,"contact":9708546777,"name":"Anurag Gautam"},"316":{"items":50,"contact":9586434319,"name":"Charles Deamon"},
"317":{"items":28,"contact":9794306009,"name":"Charles Charmichael"},"318":{"items":3,"contact":9741525934,"name":"Sarah Walker"},"319":{"items":23,
"contact":9768367555,"name":"Anurag Deamon"},"320":{"items":48,"contact":9083918599,"name":"Charles Gautam"},"321":{"items":50,"contact":9838483004,
```

Operations performing on db.json

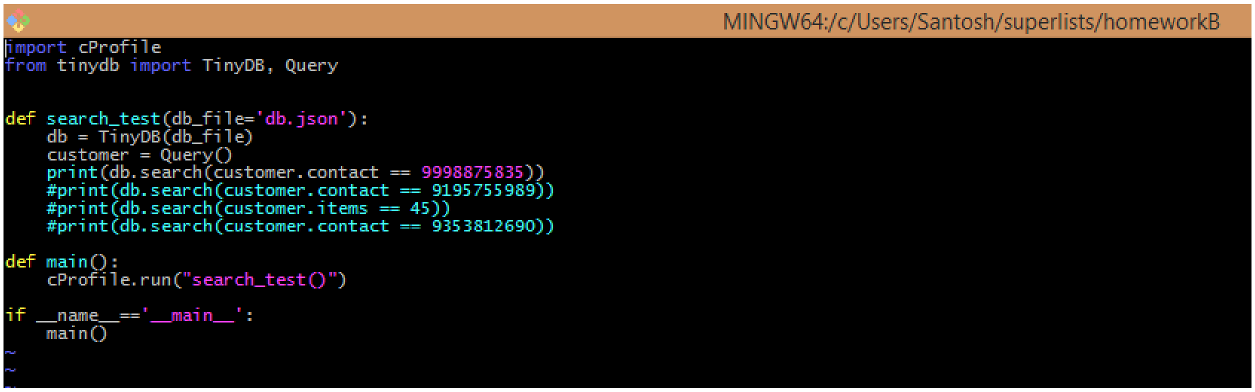
Operations we need to perform on the existing db.json file. The few operations we perform the db.json file are: searching, deleting, inserting, updating.

Searching-

a) Searching by contact-number-

We are searching for a particular customer from the db.json with contact-number of the customer.

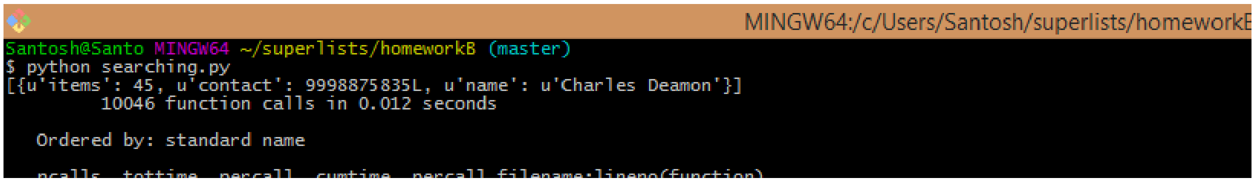
Program:



MINGW64:/c/Users/Santosh/superlists/homeworkB

```
import cProfile
from tinydb import TinyDB, Query


def search_test(db_file='db.json'):
    db = TinyDB(db_file)
    customer = Query()
    print(db.search(customer.contact == 9998875835))
    #print(db.search(customer.contact == 9195755989))
    #print(db.search(customer.items == 45))
    #print(db.search(customer.contact == 9353812690))

def main():
    cProfile.run("search_test()")

if __name__=='__main__':
    main()
~
~
```

Output:



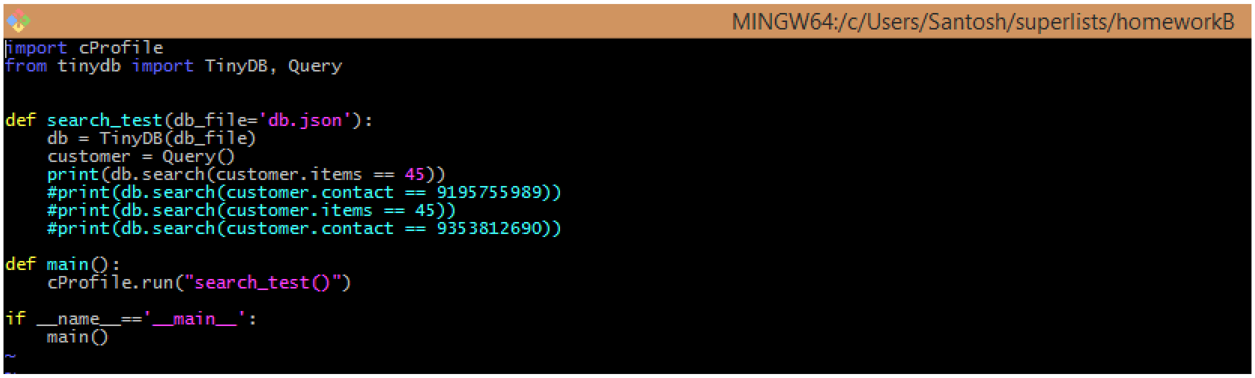
MINGW64:/c/Users/Santosh/superlists/homeworkB

```
Santosh@Santo MINGW64 ~/superlists/homeworkB (master)
$ python searching.py
[{u'items': 45, u'contact': 9998875835L, u'name': u'Charles Deamon'}]
         10046 function calls in 0.012 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
```

b) Searching by number of items-

We are searching for particular customers who have (**number of items = 45**)

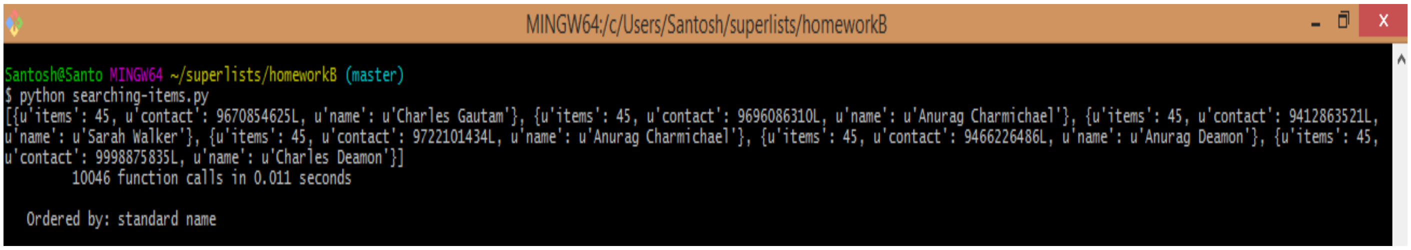
Program:



MINGW64:/c/Users/Santosh/superlists/homeworkB

```
import cProfile
from tinydb import TinyDB, Query


def search_test(db_file='db.json'):
    db = TinyDB(db_file)
    customer = Query()
    print(db.search(customer.items == 45))
    #print(db.search(customer.contact == 9195755989))
    #print(db.search(customer.items == 45))
    #print(db.search(customer.contact == 9353812690))

def main():
    cProfile.run("search_test()")

if __name__=='__main__':
    main()
~
```

Output:



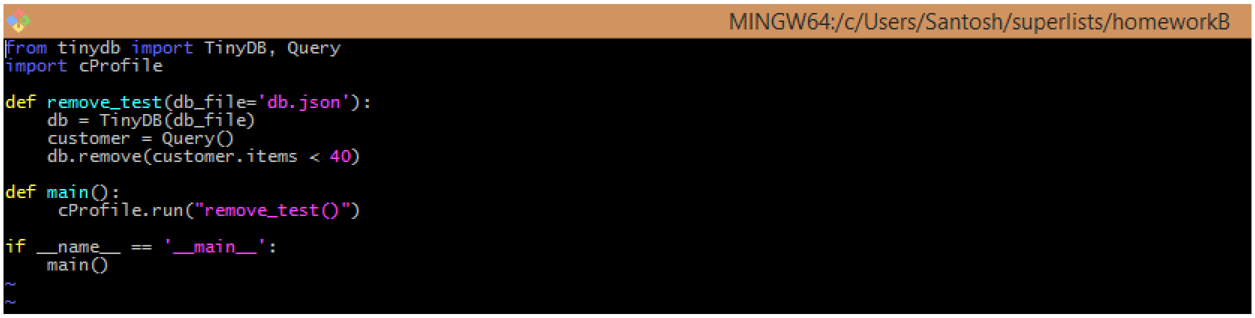
MINGW64:/c/Users/Santosh/superlists/homeworkB

```
Santosh@Santo MINGW64 ~/superlists/homeworkB (master)
$ python searching-items.py
[{u'items': 45, u'contact': 9670854625L, u'name': u'Charles Gautam'}, {u'items': 45, u'contact': 9696086310L, u'name': u'Anurag Charmichael'}, {u'items': 45, u'contact': 9412863521L,
u'name': u'Sarah Walker'}, {u'items': 45, u'contact': 9722101434L, u'name': u'Anurag Charmichael'}, {u'items': 45, u'contact': 9466226486L, u'name': u'Anurag Deamon'}, {u'items': 45,
u'contact': 9998875835L, u'name': u'Charles Deamon'}]
         10046 function calls in 0.011 seconds

   Ordered by: standard name
```

Deleting-

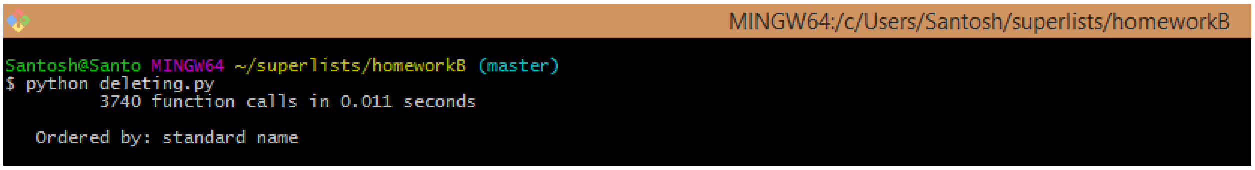
We are deleting the customer records who have less than 40 number of items.

Program:



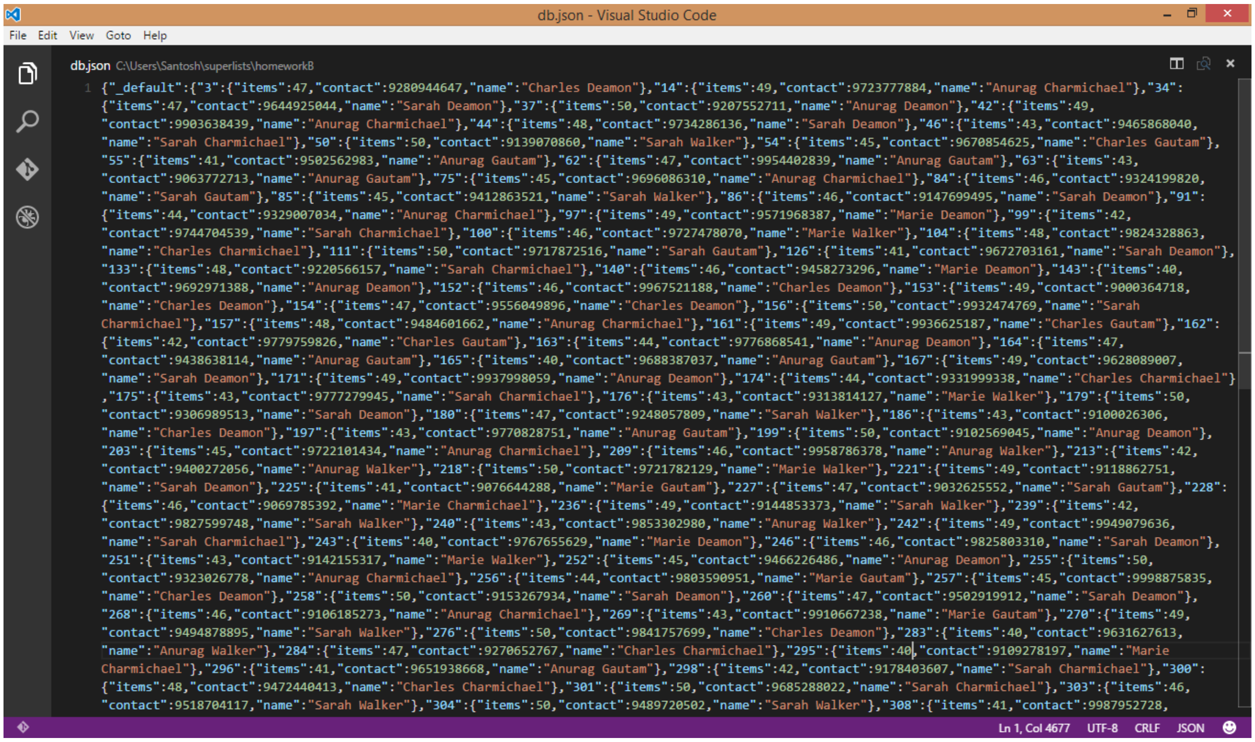
```
from tinydb import TinyDB, Query
import cProfile

def remove_test(db_file='db.json'):
    db = TinyDB(db_file)
    customer = Query()
    db.remove(customer.items < 40)

def main():
     cProfile.run("remove_test()")

if __name__ == '__main__':
    main()
~
~
```

Output:



```
Santosh@Santo MINGW64 ~/superlists/homeworkB (master)
$ python deleting.py
         3740 function calls in 0.011 seconds

   Ordered by: standard name
```

Db.json

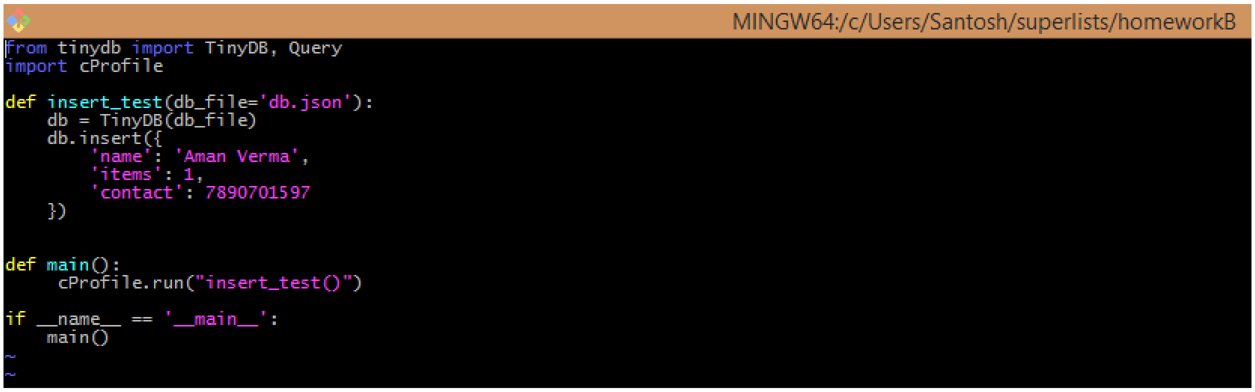

db.json - Visual Studio Code
File
Edit
View
Goto
Help
db.json C:\Users\Santosh\superlists\homeworkB
Ln 1, Col 4677
UTF-8
CRLF
JSON

```
{"_default":{"3":{"items":47,"contact":9280944647,"name":"Charles Deamon"},"14":{"items":49,"contact":9723777884,"name":"Anurag Charmichael"},"34":
{"items":47,"contact":9644925044,"name":"Sarah Deamon"},"37":{"items":50,"contact":9207552711,"name":"Anurag Deamon"},"42":{"items":49,
"contact":9903638439,"name":"Anurag Charmichael"},"44":{"items":48,"contact":9734286136,"name":"Sarah Deamon"},"46":{"items":43,"contact":9465868040,
"name":"Sarah Charmichael"},"50":{"items":50,"contact":9139070860,"name":"Sarah Walker"},"54":{"items":45,"contact":9670854625,"name":"Charles Gautam"},
"55":{"items":41,"contact":9502562983,"name":"Anurag Gautam"},"62":{"items":47,"contact":9954402839,"name":"Anurag Gautam"},"63":{"items":43,
"contact":9063772713,"name":"Anurag Gautam"},"75":{"items":45,"contact":9696086310,"name":"Anurag Charmichael"},"84":{"items":46,"contact":9324199820,
"name":"Sarah Gautam"},"85":{"items":45,"contact":9412863521,"name":"Sarah Walker"},"86":{"items":46,"contact":9147699495,"name":"Sarah Deamon"},"91":
{"items":44,"contact":9329007034,"name":"Anurag Charmichael"},"97":{"items":49,"contact":9571968387,"name":"Marie Deamon"},"99":{"items":42,
"contact":9744704539,"name":"Sarah Charmichael"},"100":{"items":46,"contact":9727478070,"name":"Marie Walker"},"104":{"items":48,"contact":9824328863,
"name":"Charles Charmichael"},"111":{"items":50,"contact":9717872516,"name":"Sarah Gautam"},"126":{"items":41,"contact":9672703161,"name":"Sarah Deamon"},
"133":{"items":48,"contact":9220566157,"name":"Sarah Charmichael"},"140":{"items":46,"contact":9458273296,"name":"Marie Deamon"},"143":{"items":40,
"contact":9692971388,"name":"Anurag Deamon"},"152":{"items":46,"contact":9967521188,"name":"Charles Deamon"},"153":{"items":49,"contact":9000364718,
"name":"Charles Deamon"},"154":{"items":47,"contact":9556049896,"name":"Charles Deamon"},"156":{"items":50,"contact":9932474769,"name":"Sarah
Charmichael"},"157":{"items":48,"contact":9484601662,"name":"Anurag Charmichael"},"161":{"items":49,"contact":9936625187,"name":"Charles Gautam"},"162":
{"items":42,"contact":9779759826,"name":"Charles Gautam"},"163":{"items":44,"contact":9776868541,"name":"Anurag Deamon"},"164":{"items":47,
"contact":9438638114,"name":"Anurag Gautam"},"165":{"items":40,"contact":9688387037,"name":"Anurag Gautam"},"167":{"items":49,"contact":9628089007,
"name":"Sarah Deamon"},"171":{"items":49,"contact":9937998059,"name":"Anurag Deamon"},"174":{"items":44,"contact":9331999338,"name":"Charles Charmichael"}
,"175":{"items":43,"contact":9777279945,"name":"Sarah Charmichael"},"176":{"items":43,"contact":9313814127,"name":"Marie Walker"},"179":{"items":50,
"contact":9306989513,"name":"Sarah Deamon"},"180":{"items":47,"contact":9248057809,"name":"Sarah Walker"},"186":{"items":43,"contact":9100026306,
"name":"Charles Deamon"},"197":{"items":43,"contact":9770828751,"name":"Anurag Gautam"},"199":{"items":50,"contact":9102569045,"name":"Anurag Deamon"},
"203":{"items":45,"contact":9722101434,"name":"Anurag Charmichael"},"209":{"items":46,"contact":9958786378,"name":"Anurag Walker"},"213":{"items":42,
"contact":9400272056,"name":"Anurag Walker"},"218":{"items":50,"contact":9721782129,"name":"Marie Walker"},"221":{"items":49,"contact":9118862751,
"name":"Sarah Deamon"},"225":{"items":41,"contact":9076644288,"name":"Marie Gautam"},"227":{"items":47,"contact":9032625552,"name":"Sarah Gautam"},"228":
{"items":46,"contact":9069785392,"name":"Marie Charmichael"},"236":{"items":49,"contact":9144853373,"name":"Sarah Walker"},"239":{"items":42,
"contact":9827599748,"name":"Sarah Walker"},"240":{"items":43,"contact":9853302980,"name":"Anurag Walker"},"242":{"items":49,"contact":9949079636,
"name":"Sarah Charmichael"},"243":{"items":40,"contact":9767655629,"name":"Marie Deamon"},"246":{"items":46,"contact":9825803310,"name":"Sarah Deamon"},
"251":{"items":43,"contact":9142155317,"name":"Marie Walker"},"252":{"items":45,"contact":9466226486,"name":"Anurag Deamon"},"255":{"items":50,
"contact":9323026778,"name":"Anurag Charmichael"},"256":{"items":44,"contact":9803590951,"name":"Marie Gautam"},"257":{"items":45,"contact":9998875835,
"name":"Charles Deamon"},"258":{"items":50,"contact":9153267934,"name":"Sarah Deamon"},"260":{"items":47,"contact":9502919912,"name":"Sarah Deamon"},
"268":{"items":46,"contact":9106185273,"name":"Anurag Charmichael"},"269":{"items":43,"contact":9910667238,"name":"Marie Gautam"},"270":{"items":49,
"contact":9494878895,"name":"Sarah Walker"},"276":{"items":50,"contact":9841757699,"name":"Charles Deamon"},"283":{"items":40,"contact":9631627613,
"name":"Anurag Walker"},"284":{"items":47,"contact":9270652767,"name":"Charles Charmichael"},"295":{"items":40,"contact":9109278197,"name":"Marie
Charmichael"},"296":{"items":41,"contact":9651938668,"name":"Anurag Gautam"},"298":{"items":42,"contact":9178403607,"name":"Sarah Charmichael"},"300":
{"items":48,"contact":9472440413,"name":"Charles Charmichael"},"301":{"items":50,"contact":9685288022,"name":"Sarah Charmichael"},"303":{"items":46,
"contact":9518704117,"name":"Sarah Walker"},"304":{"items":50,"contact":9489720502,"name":"Sarah Walker"},"308":{"items":41,"contact":9987952728,
```

Inserting-

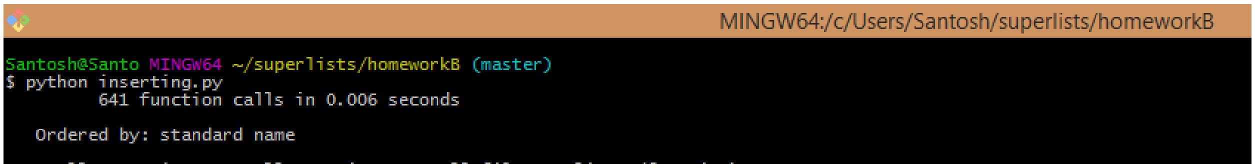
We are inserting new customer records in to the existing json file.

Program:



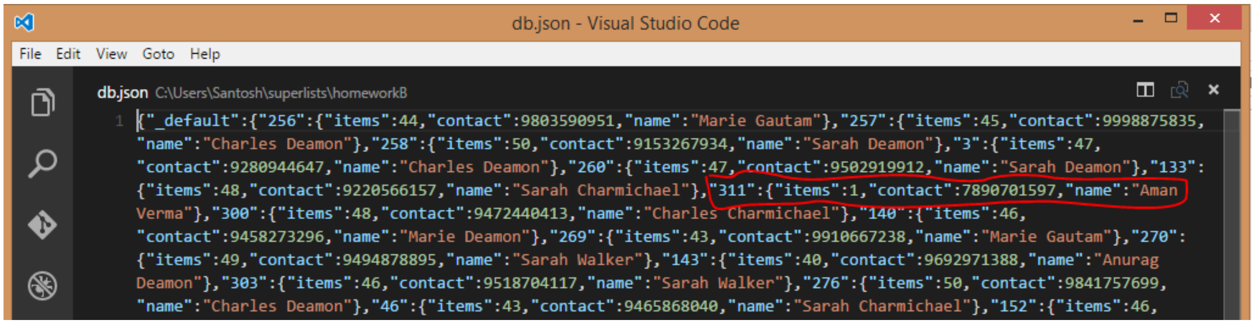
```
from tinydb import TinyDB, Query
import cProfile

def insert_test(db_file='db.json'):
    db = TinyDB(db_file)
    db.insert({
        'name': 'Aman Verma',
        'items': 1,
        'contact': 7890701597
    })

def main():
     cProfile.run("insert_test()")

if __name__ == '__main__':
    main()
~
~
```

Output:



```
Santosh@Santo MINGW64 ~/superlists/homeworkB (master)
$ python inserting.py
         641 function calls in 0.006 seconds

   Ordered by: standard name
```

Db.json

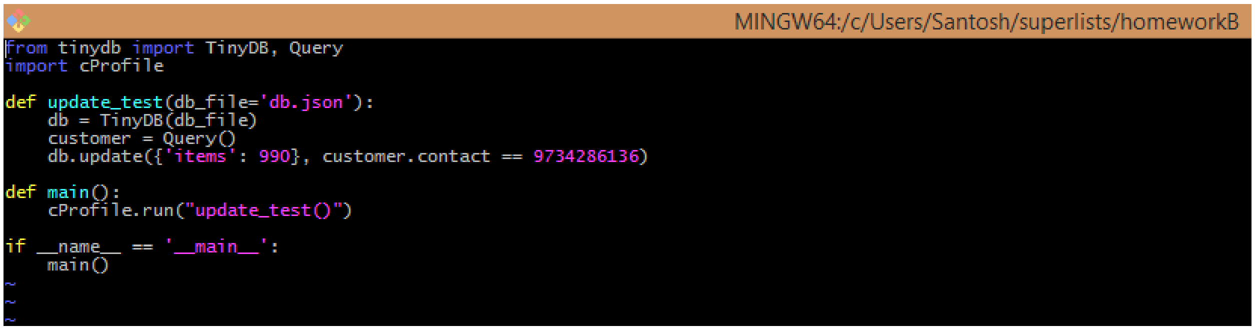

db.json - Visual Studio Code
File Edit View Goto Help
db.json C:\Users\Santosh\superlists\homeworkB

```
{"_default":{"256":{"items":44,"contact":9803590951,"name":"Marie Gautam"},"257":{"items":45,"contact":9998875835,
"name":"Charles Deamon"},"258":{"items":50,"contact":9153267934,"name":"Sarah Deamon"},"3":{"items":47,
"contact":9280944647,"name":"Charles Deamon"},"260":{"items":47,"contact":9502919912,"name":"Sarah Deamon"},"133":
{"items":48,"contact":9220566157,"name":"Sarah Charmichael"},"311":{"items":1,"contact":7890701597,"name":"Aman
Verma"},"300":{"items":48,"contact":9472440413,"name":"Charles Charmichael"},"140":{"items":46,
"contact":9458273296,"name":"Marie Deamon"},"269":{"items":43,"contact":9910667238,"name":"Marie Gautam"},"270":
{"items":49,"contact":9494878895,"name":"Sarah Walker"},"143":{"items":40,"contact":9692971388,"name":"Anurag
Deamon"},"303":{"items":46,"contact":9518704117,"name":"Sarah Walker"},"276":{"items":50,"contact":9841757699,
"name":"Charles Deamon"},"46":{"items":43,"contact":9465868040,"name":"Sarah Charmichael"},"152":{"items":46,
```

Updating-

We are updating the exisiting data in the json file.

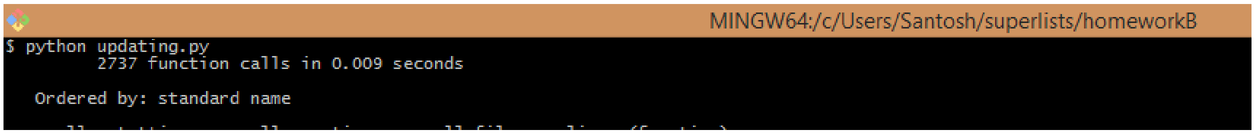
Program:



MINGW64:/c/Users/Santosh/superlists/homeworkB

```
from tinydb import TinyDB, Query
import cProfile

def update_test(db_file='db.json'):
    db = TinyDB(db_file)
    customer = Query()
    db.update({'items': 990}, customer.contact == 9734286136)

def main():
    cProfile.run("update_test()")

if __name__ == '__main__':
    main()
~
~
~
```

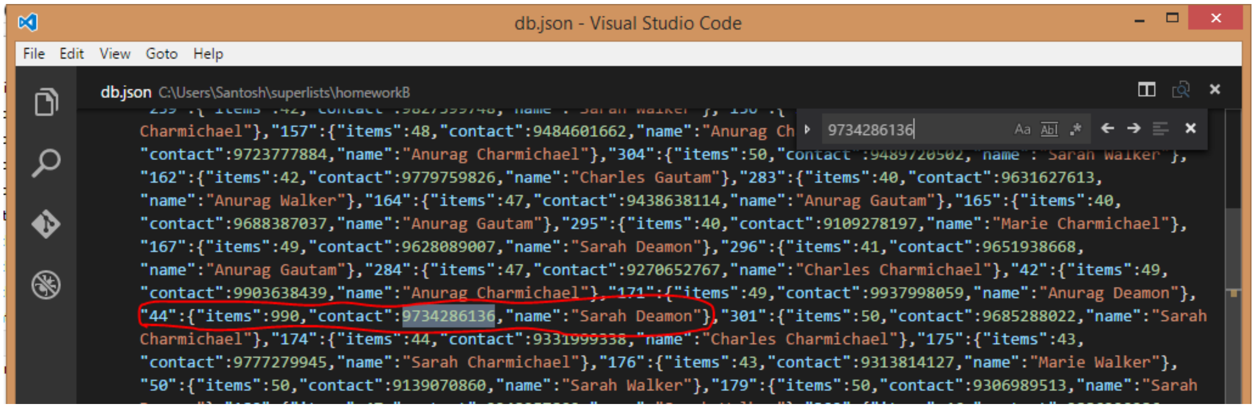
Output:



MINGW64:/c/Users/Santosh/superlists/homeworkB

```
$ python updating.py
         2737 function calls in 0.009 seconds

   Ordered by: standard name
```

Db.json



db.json - Visual Studio Code
File Edit View Goto Help
db.json C:\Users\Santosh\superlists\homeworkB
9734286136

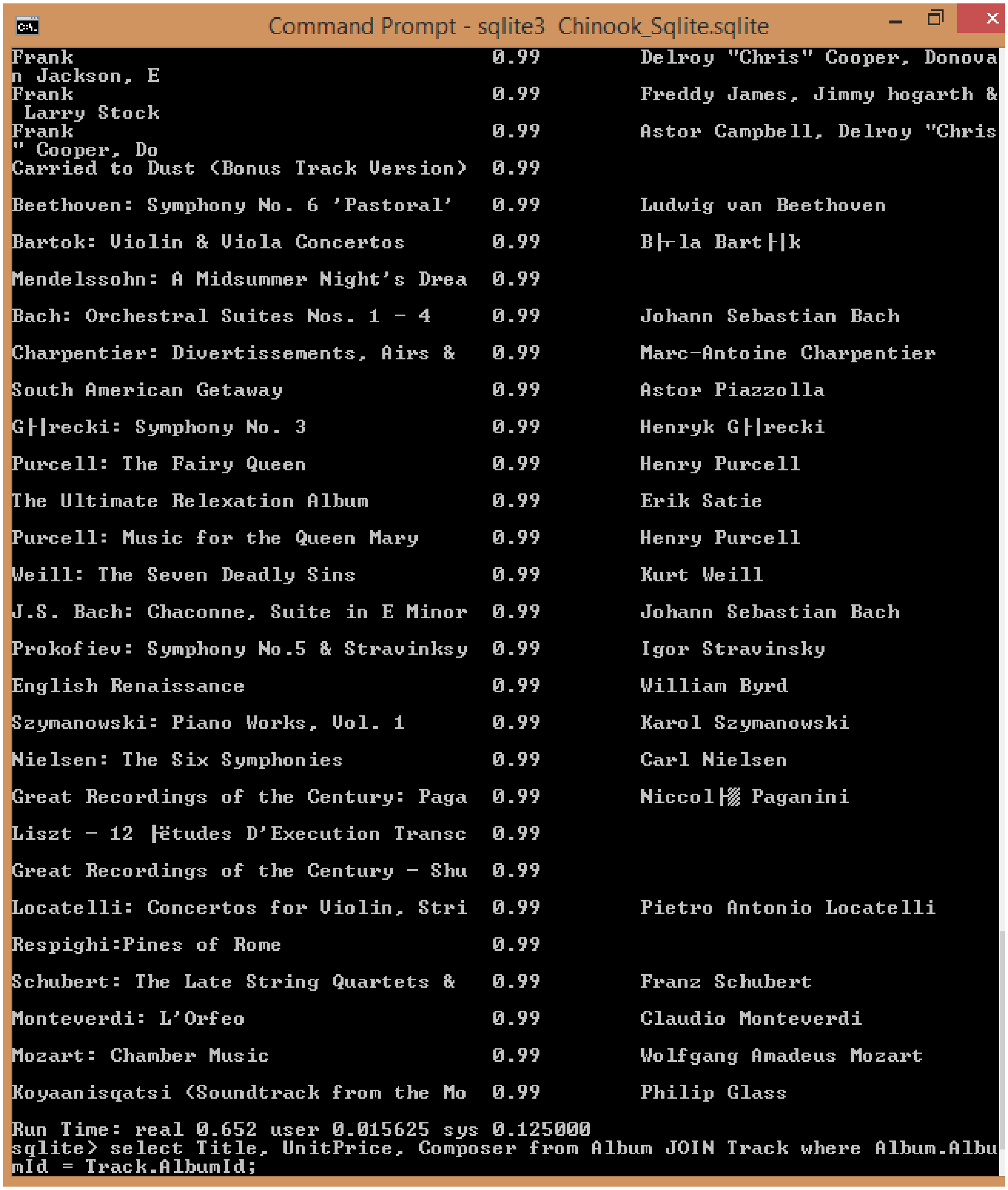
```
Charmichael"},"157":{"items":48,"contact":9484601662,"name":"Anurag Ch
"contact":9723777884,"name":"Anurag Charmichael"},"304":{"items":50,"contact":9489720502,"name":"Sarah Walker"},
"162":{"items":42,"contact":9779759826,"name":"Charles Gautam"},"283":{"items":40,"contact":9631627613,
"name":"Anurag Walker"},"164":{"items":47,"contact":9438638114,"name":"Anurag Gautam"},"165":{"items":40,
"contact":9688387037,"name":"Anurag Gautam"},"295":{"items":40,"contact":9109278197,"name":"Marie Charmichael"},
"167":{"items":49,"contact":9628089007,"name":"Sarah Deamon"},"296":{"items":41,"contact":9651938668,
"name":"Anurag Gautam"},"284":{"items":47,"contact":9270652767,"name":"Charles Charmichael"},"42":{"items":49,
"contact":9903638439,"name":"Anurag Charmichael"},"171":{"items":49,"contact":9937998059,"name":"Anurag Deamon"},
"44":{"items":990,"contact":9734286136,"name":"Sarah Deamon"},"301":{"items":50,"contact":9685288022,"name":"Sarah
Charmichael"},"174":{"items":44,"contact":9331999338,"name":"Charles Charmichael"},"175":{"items":43,
"contact":9777279945,"name":"Sarah Charmichael"},"176":{"items":43,"contact":9313814127,"name":"Marie Walker"},
"50":{"items":50,"contact":9139070860,"name":"Sarah Walker"},"179":{"items":50,"contact":9306989513,"name":"Sarah
```

B) SQLITE

We are running a query in SQLite using JOIN and id's of two different tables. By doing this, we can retrieve the column's on both the tables.

Below is the query with the run time.

Program:



```
Frank                                  0.99        Delroy "Chris" Cooper, Donova
n Jackson, E
Frank                                  0.99        Freddy James, Jimmy hogarth &
 Larry Stock
Frank                                  0.99        Astor Campbell, Delroy "Chris
" Cooper, Do
Carried to Dust (Bonus Track Version)  0.99

Beethoven: Symphony No. 6 'Pastoral'   0.99        Ludwig van Beethoven

Bartok: Violin & Viola Concertos       0.99        B├⌐la Bart├│k

Mendelssohn: A Midsummer Night's Drea  0.99

Bach: Orchestral Suites Nos. 1 - 4     0.99        Johann Sebastian Bach

Charpentier: Divertissements, Airs &   0.99        Marc-Antoine Charpentier

South American Getaway                 0.99        Astor Piazzolla

G├│recki: Symphony No. 3               0.99        Henryk G├│recki

Purcell: The Fairy Queen               0.99        Henry Purcell

The Ultimate Relexation Album          0.99        Erik Satie

Purcell: Music for the Queen Mary      0.99        Henry Purcell

Weill: The Seven Deadly Sins           0.99        Kurt Weill

J.S. Bach: Chaconne, Suite in E Minor  0.99        Johann Sebastian Bach

Prokofiev: Symphony No.5 & Stravinksy  0.99        Igor Stravinsky

English Renaissance                    0.99        William Byrd

Szymanowski: Piano Works, Vol. 1       0.99        Karol Szymanowski

Nielsen: The Six Symphonies            0.99        Carl Nielsen

Great Recordings of the Century: Paga  0.99        Niccol├▓ Paganini

Liszt - 12 ├ëtudes D'Execution Transc  0.99

Great Recordings of the Century - Shu  0.99

Locatelli: Concertos for Violin, Stri  0.99        Pietro Antonio Locatelli

Respighi:Pines of Rome                 0.99

Schubert: The Late String Quartets &   0.99        Franz Schubert

Monteverdi: L'Orfeo                    0.99        Claudio Monteverdi

Mozart: Chamber Music                  0.99        Wolfgang Amadeus Mozart

Koyaanisqatsi (Soundtrack from the Mo  0.99        Philip Glass

Run Time: real 0.652 user 0.015625 sys 0.125000
sqlite> select Title, UnitPrice, Composer from Album JOIN Track where Album.Albu
mId = Track.AlbumId;
```

We performed test cases on SQLite also. We found that SQLite uses the binary format for storing. Whereas, TinyDB uses json which can be used in the web or any application pretty easily. Even JavaScript can parse the json.

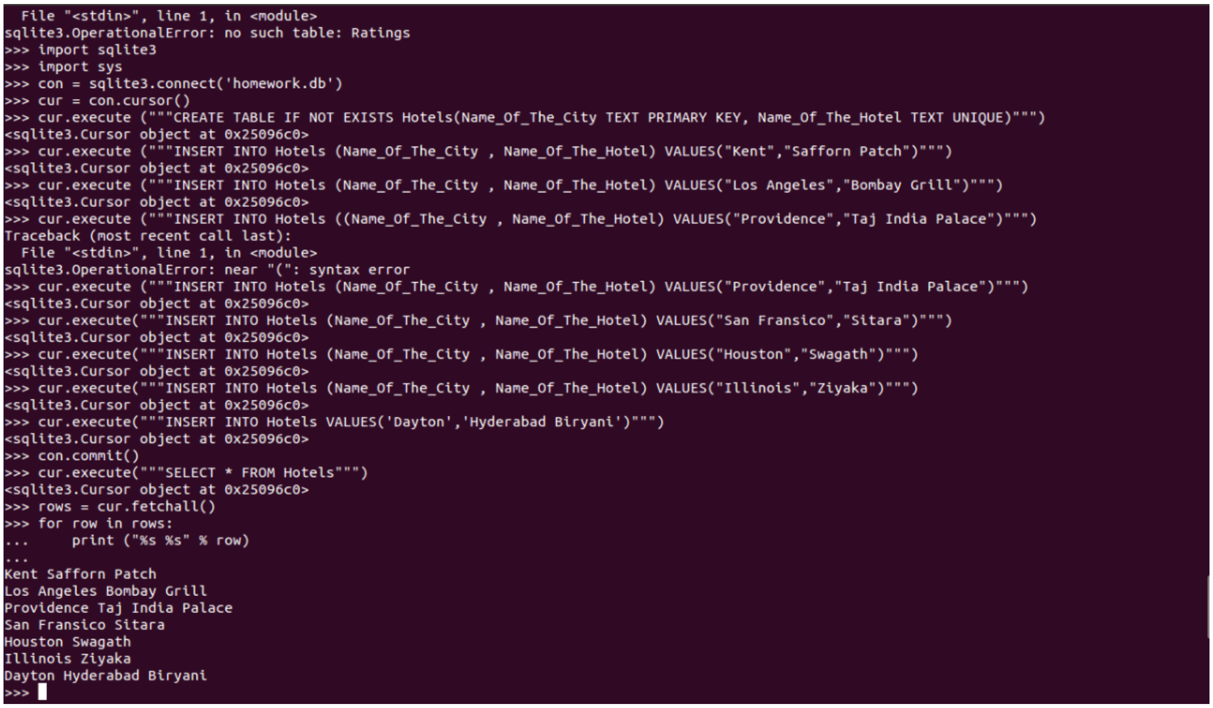
We have many pros of TinyDB over SQLite. This makes TinyDB to be used wider than SQLite. TinyDB is more efficient for tiny-databases, which computes over a tiny-operating systems.

TinyDB would be the choice of databases for storing small data from now onwards.

C) PYTHON

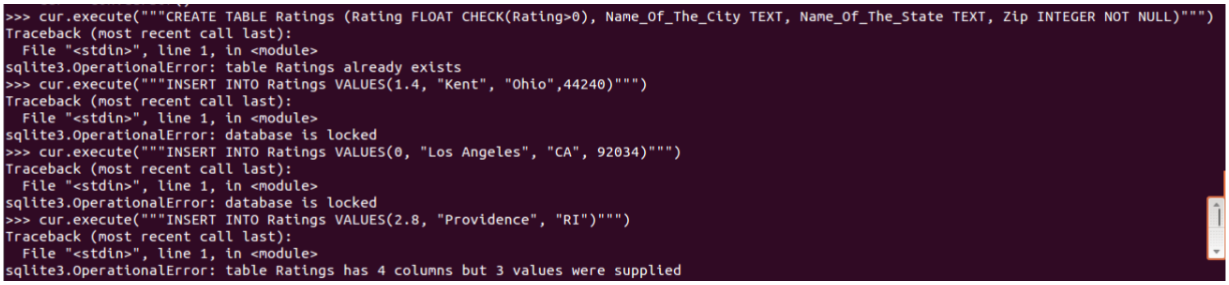
(i) Creating table “Hotels” using the constraints PRIMARY KEY, UNIQUE, IF NOT EXISTS.

In python, sqlite3 can be implemented using the “import sqlite3” statement

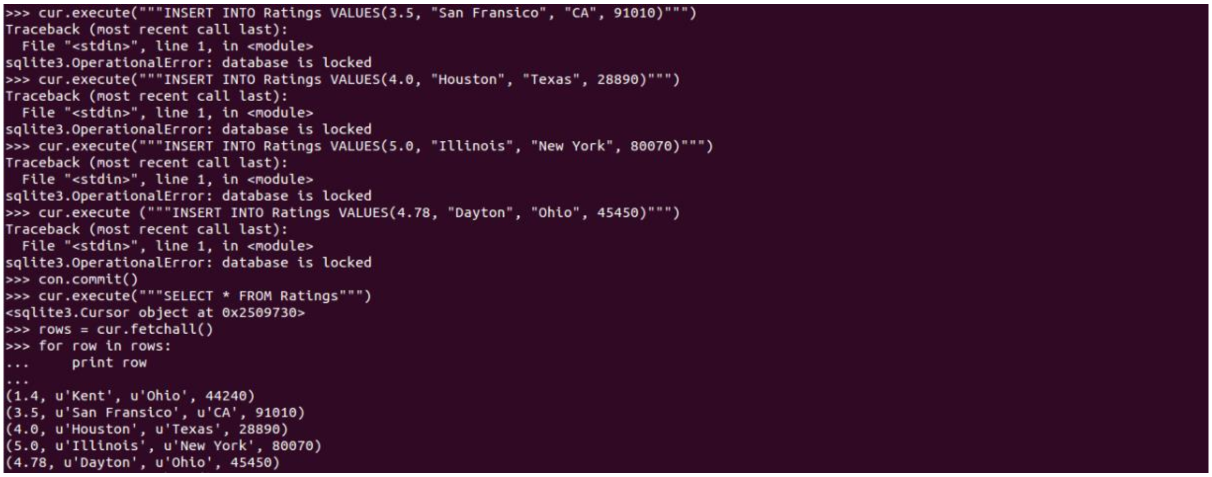


```
  File "<stdin>", line 1, in <module>
sqlite3.OperationalError: no such table: Ratings
>>> import sqlite3
>>> import sys
>>> con = sqlite3.connect('homework.db')
>>> cur = con.cursor()
>>> cur.execute ("""CREATE TABLE IF NOT EXISTS Hotels(Name_Of_The_City TEXT PRIMARY KEY, Name_Of_The_Hotel TEXT UNIQUE)""")
<sqlite3.Cursor object at 0x25096c0>
>>> cur.execute ("""INSERT INTO Hotels (Name_Of_The_City , Name_Of_The_Hotel) VALUES("Kent","Safforn Patch")""")
<sqlite3.Cursor object at 0x25096c0>
>>> cur.execute ("""INSERT INTO Hotels (Name_Of_The_City , Name_Of_The_Hotel) VALUES("Los Angeles","Bombay Grill")""")
<sqlite3.Cursor object at 0x25096c0>
>>> cur.execute ("""INSERT INTO Hotels ((Name_Of_The_City , Name_Of_The_Hotel) VALUES("Providence","Taj India Palace")""")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
sqlite3.OperationalError: near "(": syntax error
>>> cur.execute ("""INSERT INTO Hotels (Name_Of_The_City , Name_Of_The_Hotel) VALUES("Providence","Taj India Palace")""")
<sqlite3.Cursor object at 0x25096c0>
>>> cur.execute("""INSERT INTO Hotels (Name_Of_The_City , Name_Of_The_Hotel) VALUES("San Fransico","Sitara")""")
<sqlite3.Cursor object at 0x25096c0>
>>> cur.execute("""INSERT INTO Hotels (Name_Of_The_City , Name_Of_The_Hotel) VALUES("Houston","Swagath")""")
<sqlite3.Cursor object at 0x25096c0>
>>> cur.execute("""INSERT INTO Hotels (Name_Of_The_City , Name_Of_The_Hotel) VALUES("Illinois","Ziyaka")""")
<sqlite3.Cursor object at 0x25096c0>
>>> cur.execute("""INSERT INTO Hotels VALUES('Dayton','Hyderabad Biryani')""")
<sqlite3.Cursor object at 0x25096c0>
>>> con.commit()
>>> cur.execute("""SELECT * FROM Hotels""")
<sqlite3.Cursor object at 0x25096c0>
>>> rows = cur.fetchall()
>>> for row in rows:
...     print ("%s %s" % row)
...
Kent Safforn Patch
Los Angeles Bombay Grill
Providence Taj India Palace
San Fransico Sitara
Houston Swagath
Illinois Ziyaka
Dayton Hyderabad Biryani
>>> 
```

(ii) Creating table “Ratings” using the constraint NOT NULL, CHECK

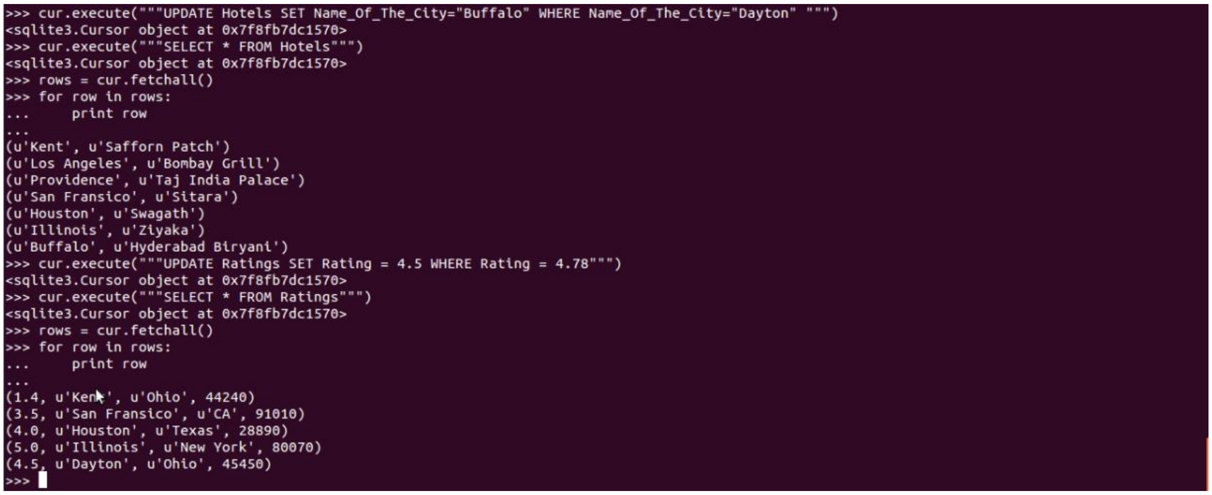


```
>>> cur.execute("""CREATE TABLE Ratings (Rating FLOAT CHECK(Rating>0), Name_Of_The_City TEXT, Name_Of_The_State TEXT, Zip INTEGER NOT NULL)""")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
sqlite3.OperationalError: table Ratings already exists
>>> cur.execute("""INSERT INTO Ratings VALUES(1.4, "Kent", "Ohio",44240)""")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
sqlite3.OperationalError: database is locked
>>> cur.execute("""INSERT INTO Ratings VALUES(0, "Los Angeles", "CA", 92034)""")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
sqlite3.OperationalError: database is locked
>>> cur.execute("""INSERT INTO Ratings VALUES(2.8, "Providence", "RI")""")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
sqlite3.OperationalError: table Ratings has 4 columns but 3 values were supplied
```



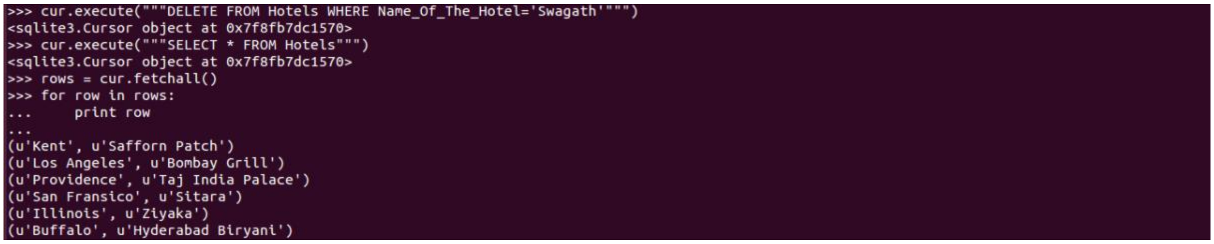
```
>>> cur.execute("""INSERT INTO Ratings VALUES(3.5, "San Fransico", "CA", 91010)""")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
sqlite3.OperationalError: database is locked
>>> cur.execute("""INSERT INTO Ratings VALUES(4.0, "Houston", "Texas", 28890)""")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
sqlite3.OperationalError: database is locked
>>> cur.execute("""INSERT INTO Ratings VALUES(5.0, "Illinois", "New York", 80070)""")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
sqlite3.OperationalError: database is locked
>>> cur.execute ("""INSERT INTO Ratings VALUES(4.78, "Dayton", "Ohio", 45450)""")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
sqlite3.OperationalError: database is locked
>>> con.commit()
>>> cur.execute("""SELECT * FROM Ratings""")
<sqlite3.Cursor object at 0x2509730>
>>> rows = cur.fetchall()
>>> for row in rows:
...     print row
...
(1.4, u'Kent', u'Ohio', 44240)
(3.5, u'San Fransico', u'CA', 91010)
(4.0, u'Houston', u'Texas', 28890)
(5.0, u'Illinois', u'New York', 80070)
(4.78, u'Dayton', u'Ohio', 45450)
```

(iii) Updating the column values in both the tables “Hotels” and “Ratings”



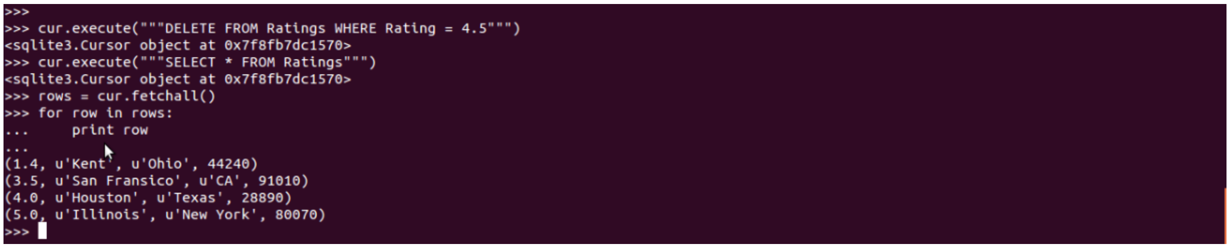
```
>>> cur.execute("""UPDATE Hotels SET Name_Of_The_City="Buffalo" WHERE Name_Of_The_City="Dayton" """)
<sqlite3.Cursor object at 0x7f8fb7dc1570>
>>> cur.execute("""SELECT * FROM Hotels""")
<sqlite3.Cursor object at 0x7f8fb7dc1570>
>>> rows = cur.fetchall()
>>> for row in rows:
...     print row
...
(u'Kent', u'Safforn Patch')
(u'Los Angeles', u'Bombay Grill')
(u'Providence', u'Taj India Palace')
(u'San Fransico', u'Sitara')
(u'Houston', u'Swagath')
(u'Illinois', u'Ziyaka')
(u'Buffalo', u'Hyderabad Biryani')
>>> cur.execute("""UPDATE Ratings SET Rating = 4.5 WHERE Rating = 4.78""")
<sqlite3.Cursor object at 0x7f8fb7dc1570>
>>> cur.execute("""SELECT * FROM Ratings""")
<sqlite3.Cursor object at 0x7f8fb7dc1570>
>>> rows = cur.fetchall()
>>> for row in rows:
...     print row
...
(1.4, u'Kent', u'Ohio', 44240)
(3.5, u'San Fransico', u'CA', 91010)
(4.0, u'Houston', u'Texas', 28890)
(5.0, u'Illinois', u'New York', 80070)
(4.5, u'Dayton', u'Ohio', 45450)
>>> 
```

(iv) Deleting a row from “Hotels” table



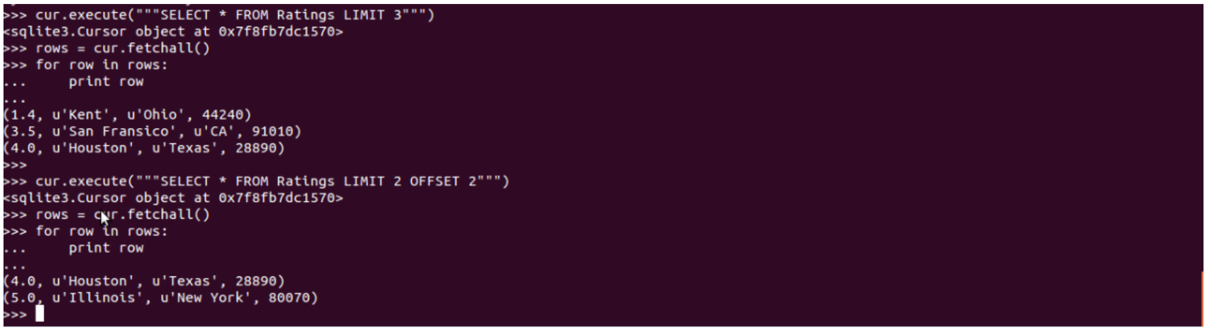
```
>>> cur.execute("""DELETE FROM Hotels WHERE Name_Of_The_Hotel='Swagath'""")
<sqlite3.Cursor object at 0x7f8fb7dc1570>
>>> cur.execute("""SELECT * FROM Hotels""")
<sqlite3.Cursor object at 0x7f8fb7dc1570>
>>> rows = cur.fetchall()
>>> for row in rows:
...     print row
...
(u'Kent', u'Safforn Patch')
(u'Los Angeles', u'Bombay Grill')
(u'Providence', u'Taj India Palace')
(u'San Fransico', u'Sitara')
(u'Illinois', u'Ziyaka')
(u'Buffalo', u'Hyderabad Biryani')
```

(v) Deleting a row from “Ratings” table



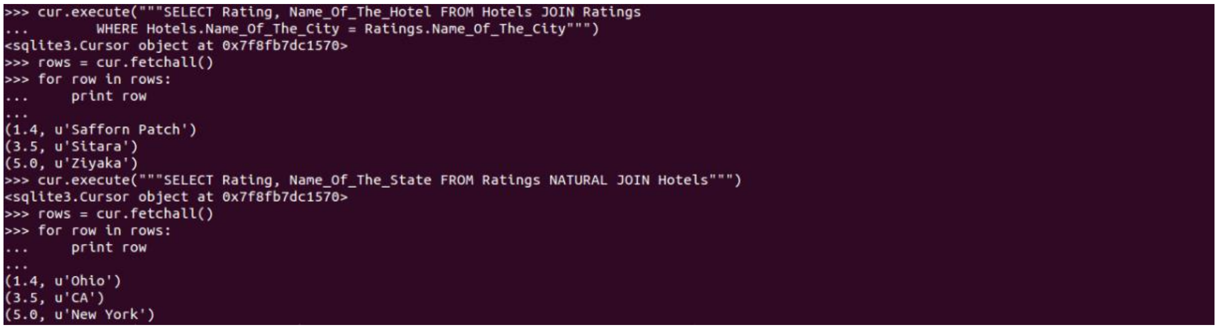
```
>>>
>>> cur.execute("""DELETE FROM Ratings WHERE Rating = 4.5""")
<sqlite3.Cursor object at 0x7f8fb7dc1570>
>>> cur.execute("""SELECT * FROM Ratings""")
<sqlite3.Cursor object at 0x7f8fb7dc1570>
>>> rows = cur.fetchall()
>>> for row in rows:
...     print row
...
(1.4, u'Kent', u'Ohio', 44240)
(3.5, u'San Fransico', u'CA', 91010)
(4.0, u'Houston', u'Texas', 28890)
(5.0, u'Illinois', u'New York', 80070)
>>> 
```

(vi) Implementing LIMIT, OFFSET clauses on the table “Ratings”



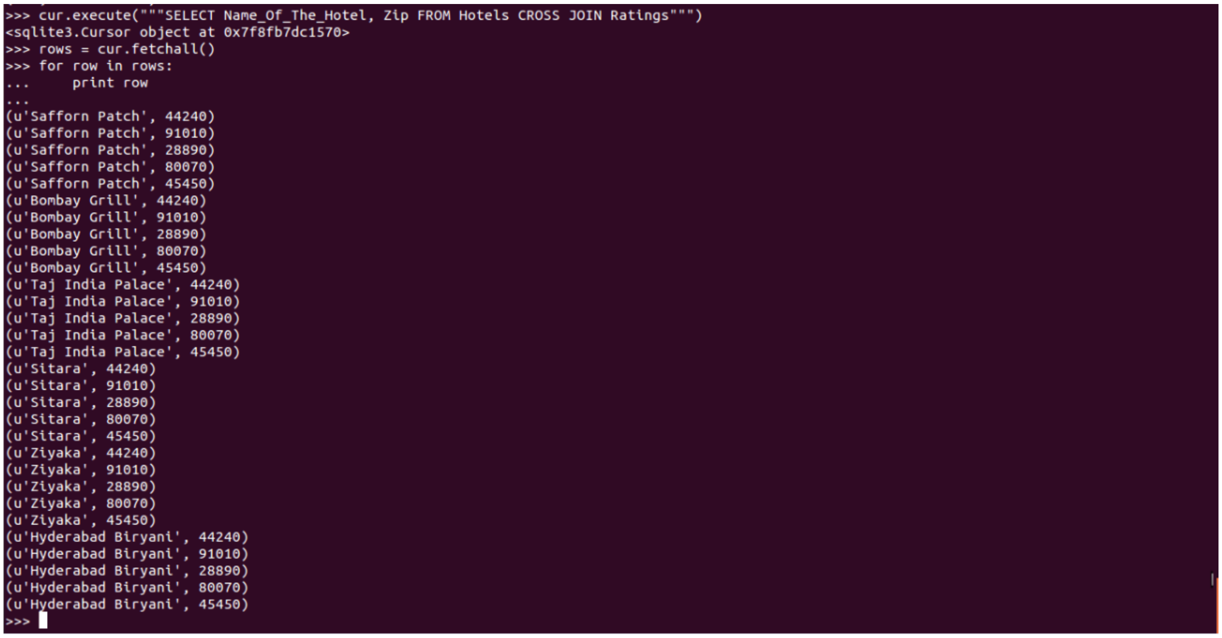
```
>> cur.execute("""SELECT * FROM Ratings LIMIT 3""")
<sqlite3.Cursor object at 0x7f8fb7dc1570>
>> rows = cur.fetchall()
>> for row in rows:
..     print row
..
(1.4, u'Kent', u'Ohio', 44240)
(3.5, u'San Fransico', u'CA', 91010)
(4.0, u'Houston', u'Texas', 28890)
>>
>> cur.execute("""SELECT * FROM Ratings LIMIT 2 OFFSET 2""")
<sqlite3.Cursor object at 0x7f8fb7dc1570>
>> rows = cur.fetchall()
>> for row in rows:
..     print row
..
(4.0, u'Houston', u'Texas', 28890)
(5.0, u'Illinois', u'New York', 80070)
>> 
```

(vii) Performing Inner Joins and Natural Join between “Hotels” and “Ratings” tables



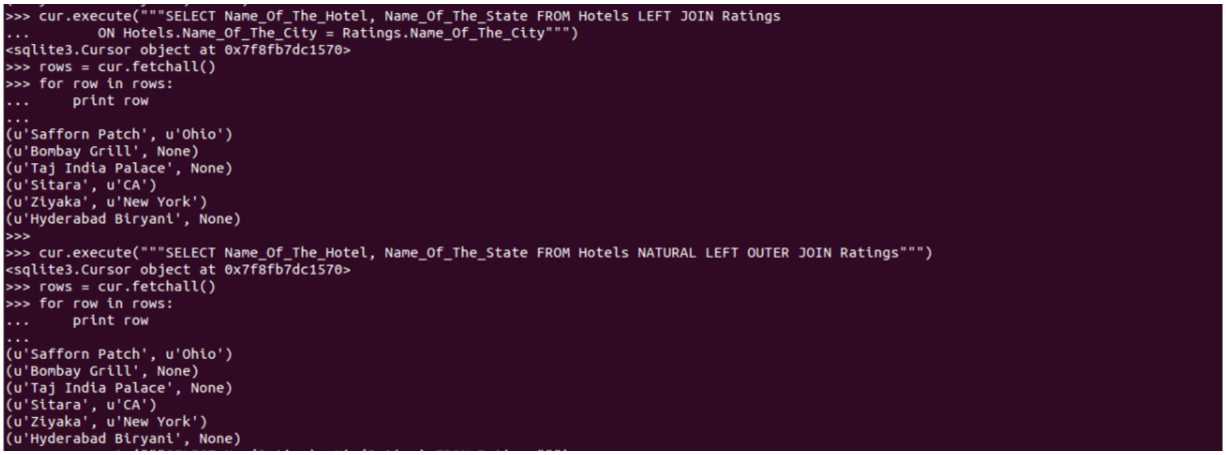
```
>>> cur.execute("""SELECT Rating, Name_Of_The_Hotel FROM Hotels JOIN Ratings
...         WHERE Hotels.Name_Of_The_City = Ratings.Name_Of_The_City""")
<sqlite3.Cursor object at 0x7f8fb7dc1570>
>>> rows = cur.fetchall()
>>> for row in rows:
...     print row
...
(1.4, u'Safforn Patch')
(3.5, u'Sitara')
(5.0, u'Ziyaka')
>>> cur.execute("""SELECT Rating, Name_Of_The_State FROM Ratings NATURAL JOIN Hotels""")
<sqlite3.Cursor object at 0x7f8fb7dc1570>
>>> rows = cur.fetchall()
>>> for row in rows:
...     print row
...
(1.4, u'Ohio')
(3.5, u'CA')
(5.0, u'New York')
```

(viii) Performing Cross Join between “Hotels” and “Ratings” tables



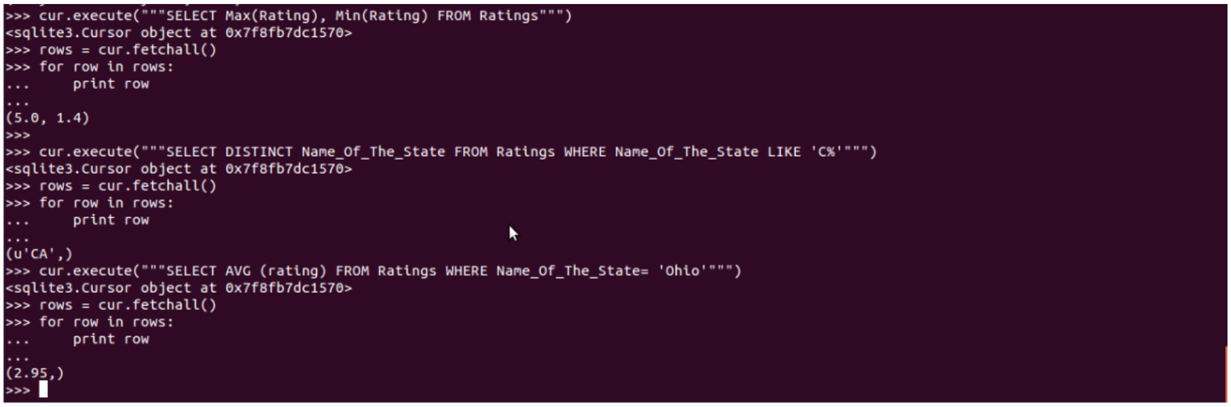
```
>>> cur.execute("""SELECT Name_Of_The_Hotel, Zip FROM Hotels CROSS JOIN Ratings""")
<sqlite3.Cursor object at 0x7f8fb7dc1570>
>>> rows = cur.fetchall()
>>> for row in rows:
...     print row
...
(u'Safforn Patch', 44240)
(u'Safforn Patch', 91010)
(u'Safforn Patch', 28890)
(u'Safforn Patch', 80070)
(u'Safforn Patch', 45450)
(u'Bombay Grill', 44240)
(u'Bombay Grill', 91010)
(u'Bombay Grill', 28890)
(u'Bombay Grill', 80070)
(u'Bombay Grill', 45450)
(u'Taj India Palace', 44240)
(u'Taj India Palace', 91010)
(u'Taj India Palace', 28890)
(u'Taj India Palace', 80070)
(u'Taj India Palace', 45450)
(u'Sitara', 44240)
(u'Sitara', 91010)
(u'Sitara', 28890)
(u'Sitara', 80070)
(u'Sitara', 45450)
(u'Ziyaka', 44240)
(u'Ziyaka', 91010)
(u'Ziyaka', 28890)
(u'Ziyaka', 80070)
(u'Ziyaka', 45450)
(u'Hyderabad Biryani', 44240)
(u'Hyderabad Biryani', 91010)
(u'Hyderabad Biryani', 28890)
(u'Hyderabad Biryani', 80070)
(u'Hyderabad Biryani', 45450)
>>>
```

(ix) Performing Left Outer Join, Natural Left Outer Join between “Hotels” and “Ratings”



```
>>> cur.execute("""SELECT Name_Of_The_Hotel, Name_Of_The_State FROM Hotels LEFT JOIN Ratings
...        ON Hotels.Name_Of_The_City = Ratings.Name_Of_The_City""")
<sqlite3.Cursor object at 0x7f8fb7dc1570>
>>> rows = cur.fetchall()
>>> for row in rows:
...     print row
...
(u'Safforn Patch', u'Ohio')
(u'Bombay Grill', None)
(u'Taj India Palace', None)
(u'Sitara', u'CA')
(u'Ziyaka', u'New York')
(u'Hyderabad Biryani', None)
>>>
>>> cur.execute("""SELECT Name_Of_The_Hotel, Name_Of_The_State FROM Hotels NATURAL LEFT OUTER JOIN Ratings""")
<sqlite3.Cursor object at 0x7f8fb7dc1570>
>>> rows = cur.fetchall()
>>> for row in rows:
...     print row
...
(u'Safforn Patch', u'Ohio')
(u'Bombay Grill', None)
(u'Taj India Palace', None)
(u'Sitara', u'CA')
(u'Ziyaka', u'New York')
(u'Hyderabad Biryani', None)
```

(x) Using functions like Max(), Min(), and Avg() to find the maximum rating, minimum rating, average rating of hotels in ‘Ohio’ state



```
>>> cur.execute("""SELECT Max(Rating), Min(Rating) FROM Ratings""")
<sqlite3.Cursor object at 0x7f8fb7dc1570>
>>> rows = cur.fetchall()
>>> for row in rows:
...     print row
...
(5.0, 1.4)
>>>
>>> cur.execute("""SELECT DISTINCT Name_Of_The_State FROM Ratings WHERE Name_Of_The_State LIKE 'C%'""")
<sqlite3.Cursor object at 0x7f8fb7dc1570>
>>> rows = cur.fetchall()
>>> for row in rows:
...     print row
...
(u'CA',)
>>> cur.execute("""SELECT AVG (rating) FROM Ratings WHERE Name_Of_The_State= 'Ohio'""")
<sqlite3.Cursor object at 0x7f8fb7dc1570>
>>> rows = cur.fetchall()
>>> for row in rows:
...     print row
...
(2.95,)
>>> 
```