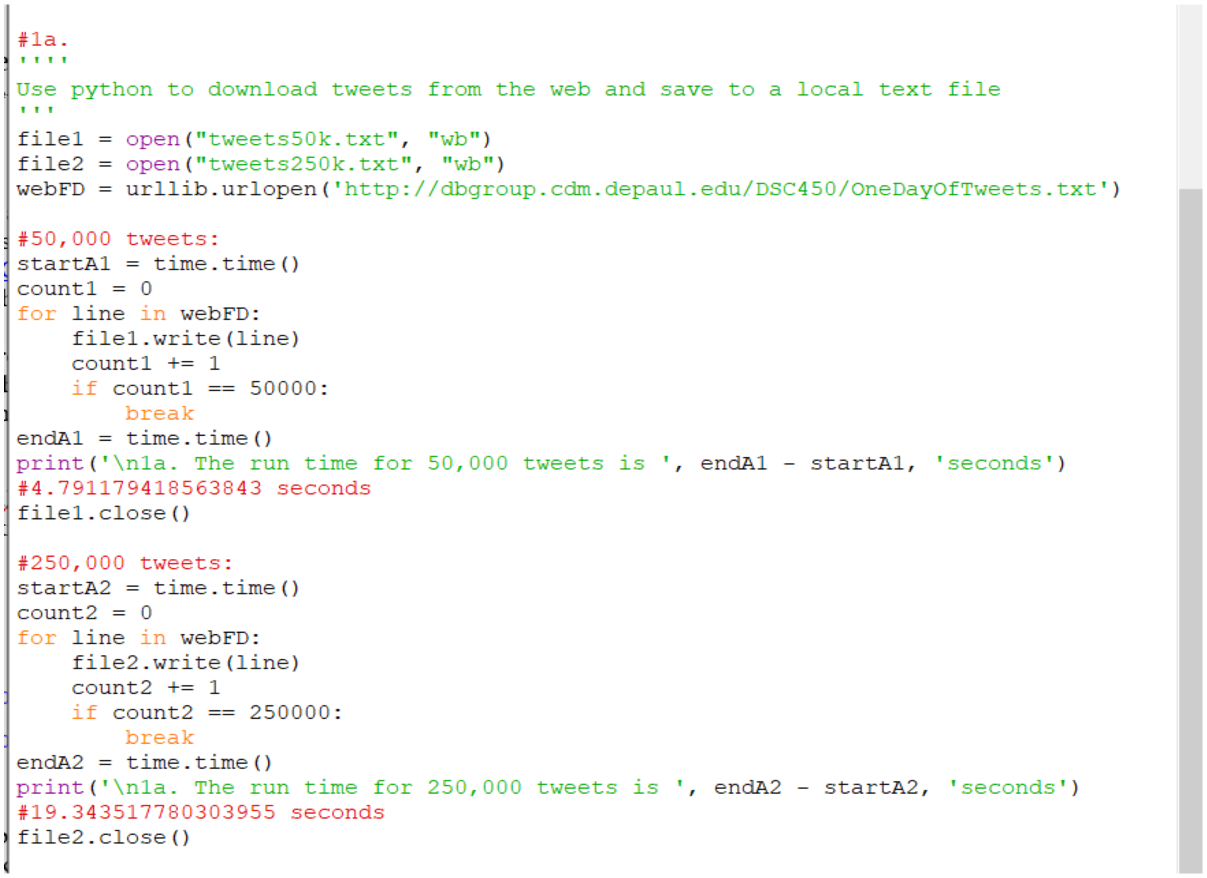
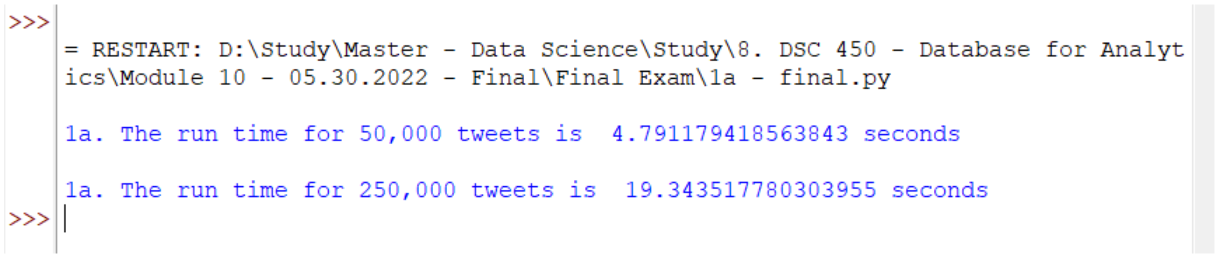
Database Processing for Large-Scale Analytics
Final Report
Student name: Ha Hai Vu

Part 1

a. Use python to download tweets from the web and save to a local text file (not into a database yet, just to a text file). This is as simple as it sounds, all you need is a for-loop that reads lines and writes them into a file, just don't forget to add '\n' at the end so they are, in fact, on separate lines.



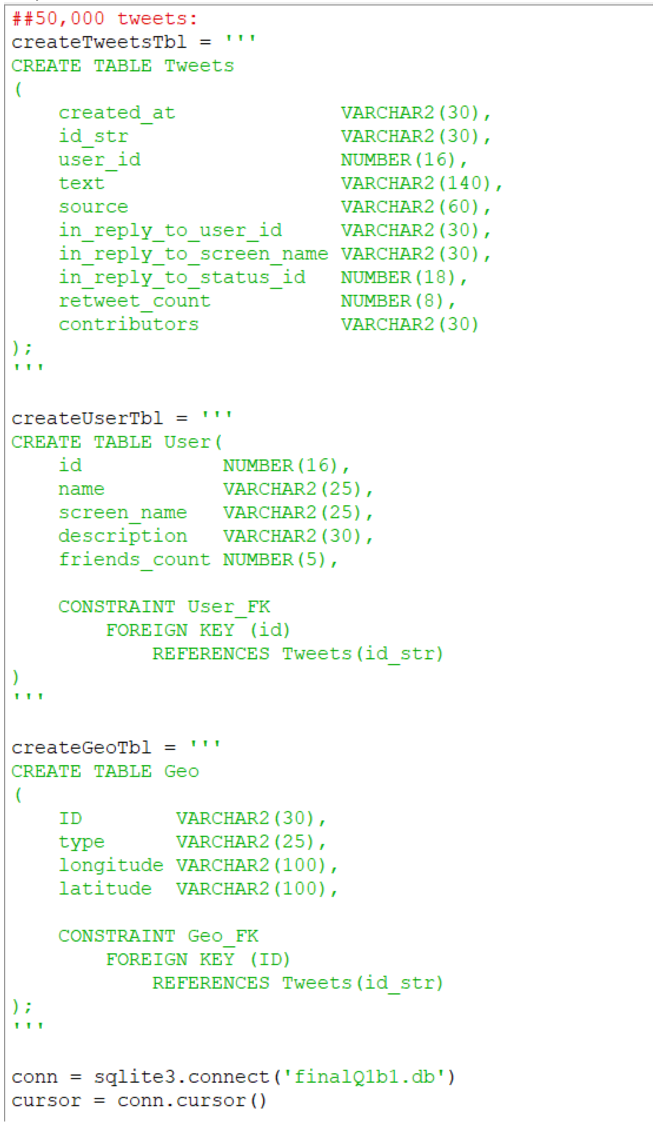
```
#1a.
''''
Use python to download tweets from the web and save to a local text file
'''
file1 = open("tweets50k.txt", "wb")
file2 = open("tweets250k.txt", "wb")
webFD = urllib.urlopen('http://dbgroup.cdm.depaul.edu/DSC450/OneDayOfTweets.txt')

#50,000 tweets:
startA1 = time.time()
count1 = 0
for line in webFD:
    file1.write(line)
    count1 += 1
    if count1 == 50000:
        break
endA1 = time.time()
print('\n1a. The run time for 50,000 tweets is ', endA1 - startA1, 'seconds')
#4.791179418563843 seconds
file1.close()

#250,000 tweets:
startA2 = time.time()
count2 = 0
for line in webFD:
    file2.write(line)
    count2 += 1
    if count2 == 250000:
        break
endA2 = time.time()
print('\n1a. The run time for 250,000 tweets is ', endA2 - startA2, 'seconds')
#19.343517780303955 seconds
file2.close()
```



```
>>>
= RESTART: D:\Study\Master - Data Science\Study\8. DSC 450 - Database for Analyt
ics\Module 10 - 05.30.2022 - Final\Final Exam\1a - final.py

1a. The run time for 50,000 tweets is  4.791179418563843 seconds

1a. The run time for 250,000 tweets is  19.343517780303955 seconds
>>>
```

b. Repeat what you did in part 1-a, but instead of saving tweets to the file, populate the 3-table schema that you previously created in SQLite. Be sure to execute commit and verify that the data has been successfully loaded. Report loaded row counts for each of the 3 tables.

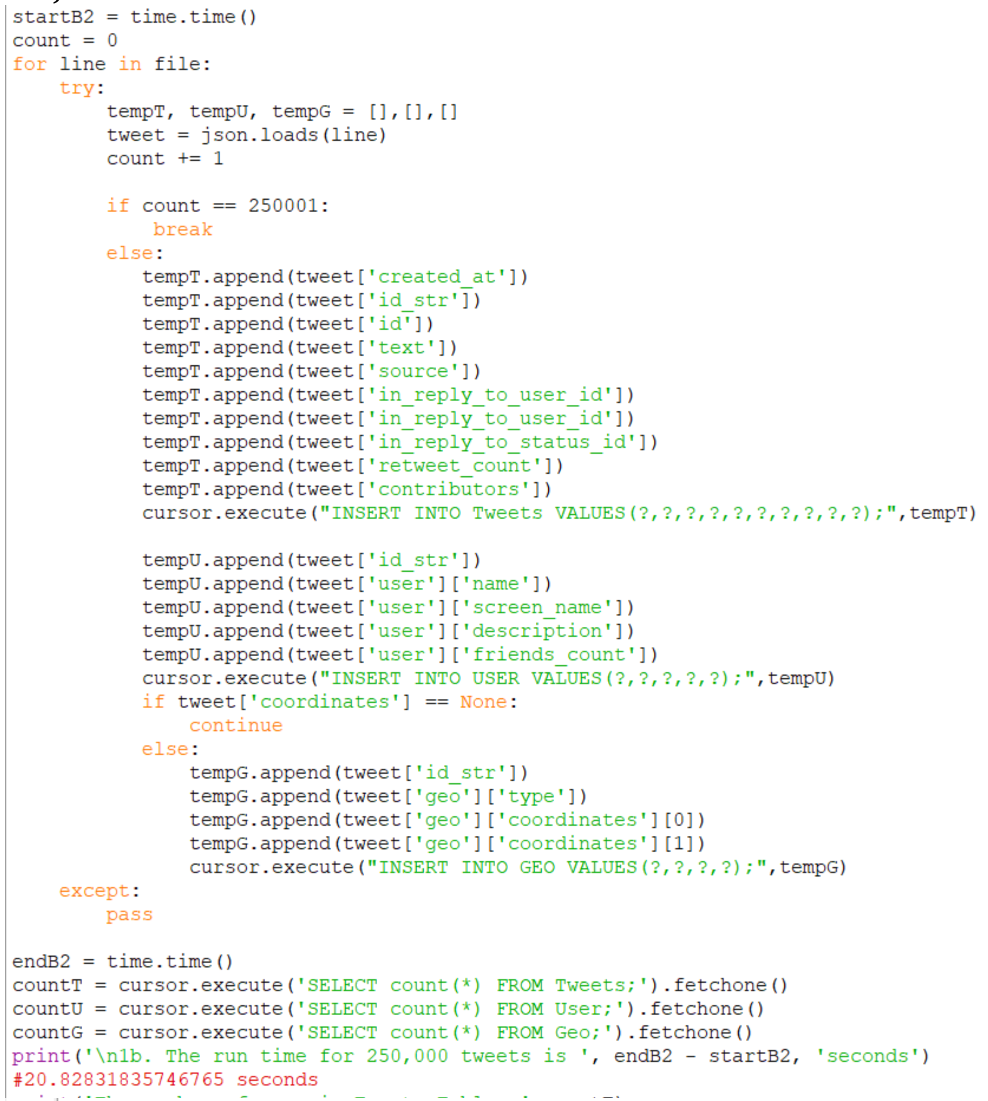
50,000 tweets



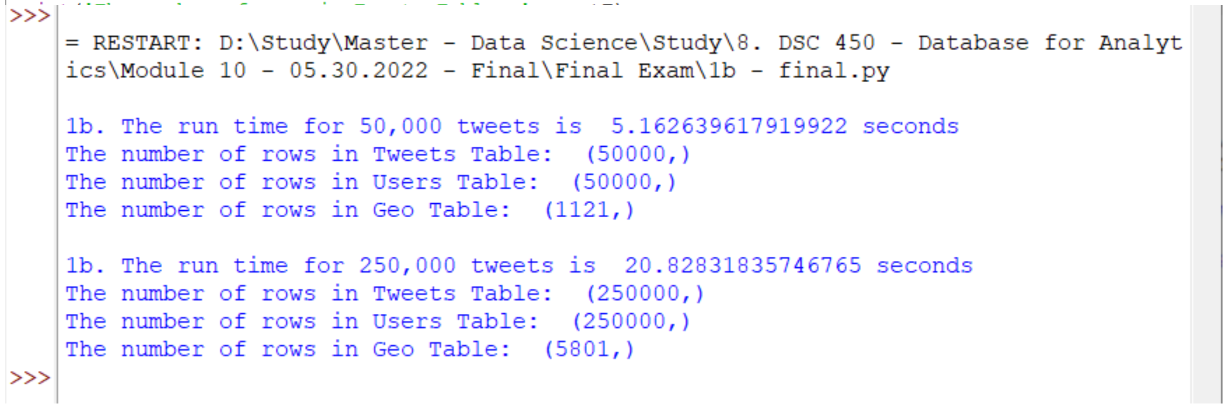
```
##50,000 tweets:
createTweetsTbl = '''
CREATE TABLE Tweets
(
    created_at              VARCHAR2(30),
    id_str                  VARCHAR2(30),
    user_id                 NUMBER(16),
    text                    VARCHAR2(140),
    source                  VARCHAR2(60),
    in_reply_to_user_id     VARCHAR2(30),
    in_reply_to_screen_name VARCHAR2(30),
    in_reply_to_status_id   NUMBER(18),
    retweet_count           NUMBER(8),
    contributors            VARCHAR2(30)
);
'''

createUserTbl = '''
CREATE TABLE User(
    id            NUMBER(16),
    name          VARCHAR2(25),
    screen_name   VARCHAR2(25),
    description   VARCHAR2(30),
    friends_count NUMBER(5),

    CONSTRAINT User_FK
        FOREIGN KEY (id)
            REFERENCES Tweets(id_str)
)
'''

createGeoTbl = '''
CREATE TABLE Geo
(
    ID        VARCHAR2(30),
    type      VARCHAR2(25),
    longitude VARCHAR2(100),
    latitude  VARCHAR2(100),

    CONSTRAINT Geo_FK
        FOREIGN KEY (ID)
            REFERENCES Tweets(id_str)
);
'''

conn = sqlite3.connect('finalQ1b1.db')
cursor = conn.cursor()
```

```
            if tweet['coordinates'] == None:
                continue
            else:
                tempG.append(tweet['id_str'])
                tempG.append(tweet['geo']['type'])
                tempG.append(tweet['geo']['coordinates'][0])
                tempG.append(tweet['geo']['coordinates'][1])
                cursor.execute("INSERT INTO GEO VALUES(?,?,?,?);",tempG)
    except:
        pass

endB1 = time.time()
countT = cursor.execute('SELECT count(*) FROM Tweets;').fetchone()
countU = cursor.execute('SELECT count(*) FROM User;').fetchone()
countG = cursor.execute('SELECT count(*) FROM Geo;').fetchone()
print('\n1b. The run time for 50,000 tweets is ', endB1 - startB1, 'seconds')
#5.162639617919922 seconds
print('The number of rows in Tweets Table: ',countT)
print('The number of rows in Users Table: ',countU)
print('The number of rows in Geo Table: ',countG)
file.close()


cursor.execute('DROP TABLE Tweets;')
cursor.execute('DROP TABLE User;')
cursor.execute('DROP TABLE Geo;')

cursor.execute(createTweetsTbl)
cursor.execute(createUserTbl)
cursor.execute(createGeoTbl)

file = urllib.urlopen('http://dbgroup.cdm.depaul.edu/DSC450/OneDayOfTweets.txt')

startB1 = time.time()
count = 0
for line in file:
    try:
        tempT, tempU, tempG = [],[],[]
        tweet = json.loads(line)
        count += 1

        if count == 50001:
            break
        else:
            tempT.append(tweet['created_at'])
            tempT.append(tweet['id_str'])
            tempT.append(tweet['id'])
            tempT.append(tweet['text'])
            tempT.append(tweet['source'])
            tempT.append(tweet['in_reply_to_user_id'])
            tempT.append(tweet['in_reply_to_user_id'])
            tempT.append(tweet['in_reply_to_status_id'])
            tempT.append(tweet['retweet_count'])
            tempT.append(tweet['contributors'])
            cursor.execute("INSERT INTO Tweets VALUES(?,?,?,?,?,?,?,?,?,?);",tempT)

            if tweet['coordinates'] == None:
                continue
            else:
                tempG.append(tweet['id_str'])
                tempG.append(tweet['geo']['type'])
                tempG.append(tweet['geo']['coordinates'][0])
                tempG.append(tweet['geo']['coordinates'][1])
                cursor.execute("INSERT INTO GEO VALUES(?,?,?,?);",tempG)
    except:
        pass

endB1 = time.time()
countT = cursor.execute('SELECT count(*) FROM Tweets;').fetchone()
countU = cursor.execute('SELECT count(*) FROM User;').fetchone()
countG = cursor.execute('SELECT count(*) FROM Geo;').fetchone()
print('\n1b. The run time for 50,000 tweets is ', endB1 - startB1, 'seconds')
#5.162639617919922 seconds
print('The number of rows in Tweets Table: ',countT)
print('The number of rows in Users Table: ',countU)
print('The number of rows in Geo Table: ',countG)
file.close()
```

250,000 tweets



```
startB2 = time.time()
count = 0
for line in file:
    try:
        tempT, tempU, tempG = [],[],[]
        tweet = json.loads(line)
        count += 1

        if count == 250001:
            break
        else:
           tempT.append(tweet['created_at'])
           tempT.append(tweet['id_str'])
           tempT.append(tweet['id'])
           tempT.append(tweet['text'])
           tempT.append(tweet['source'])
           tempT.append(tweet['in_reply_to_user_id'])
           tempT.append(tweet['in_reply_to_user_id'])
           tempT.append(tweet['in_reply_to_status_id'])
           tempT.append(tweet['retweet_count'])
           tempT.append(tweet['contributors'])
           cursor.execute("INSERT INTO Tweets VALUES(?,?,?,?,?,?,?,?,?,?);",tempT)

           tempU.append(tweet['id_str'])
           tempU.append(tweet['user']['name'])
           tempU.append(tweet['user']['screen_name'])
           tempU.append(tweet['user']['description'])
           tempU.append(tweet['user']['friends_count'])
           cursor.execute("INSERT INTO USER VALUES(?,?,?,?,?);",tempU)
           if tweet['coordinates'] == None:
               continue
           else:
               tempG.append(tweet['id_str'])
               tempG.append(tweet['geo']['type'])
               tempG.append(tweet['geo']['coordinates'][0])
               tempG.append(tweet['geo']['coordinates'][1])
               cursor.execute("INSERT INTO GEO VALUES(?,?,?,?);",tempG)
    except:
        pass

endB2 = time.time()
countT = cursor.execute('SELECT count(*) FROM Tweets;').fetchone()
countU = cursor.execute('SELECT count(*) FROM User;').fetchone()
countG = cursor.execute('SELECT count(*) FROM Geo;').fetchone()
print('\n1b. The run time for 250,000 tweets is ', endB2 - startB2, 'seconds')
#20.82831835746765 seconds
```



```
>>> 
    = RESTART: D:\Study\Master - Data Science\Study\8. DSC 450 - Database for Analyt
    ics\Module 10 - 05.30.2022 - Final\Final Exam\1b - final.py

    1b. The run time for 50,000 tweets is  5.162639617919922 seconds
    The number of rows in Tweets Table:  (50000,)
    The number of rows in Users Table:  (50000,)
    The number of rows in Geo Table:  (1121,)

    1b. The run time for 250,000 tweets is  20.82831835746765 seconds
    The number of rows in Tweets Table:  (250000,)
    The number of rows in Users Table:  (250000,)
    The number of rows in Geo Table:  (5801,)
>>> 
```

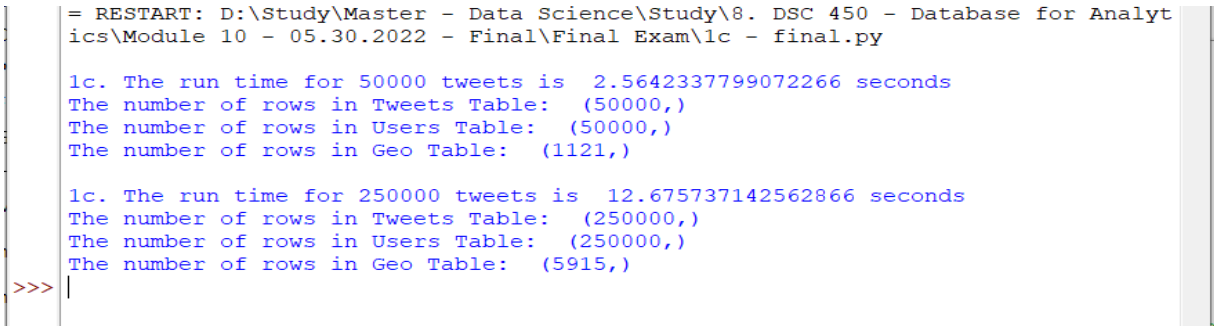
c. Use your locally saved tweet file to repeat the database population step from part-c. That is, load the tweets into the 3-table database using your saved file with tweets. This is the **same** code as in 1-b, but reading tweets from your file, not from the web.

50,000 tweets

```
fileA1 = open("tweets50k.txt", "r",encoding='utf8')

startC1 = time.time()
for i in range(50001):

    try:
        tempT, tempU, tempG = [],[],[]
        allTweets = fileA1.readline()
        tweet = json.loads(allTweets)

        tempT.append(tweet['created_at'])
        tempT.append(tweet['id_str'])
        tempT.append(tweet['id'])
        tempT.append(tweet['text'])
        tempT.append(tweet['source'])
        tempT.append(tweet['in_reply_to_user_id'])
        tempT.append(tweet['in_reply_to_user_id'])
        tempT.append(tweet['in_reply_to_status_id'])
        tempT.append(tweet['retweet_count'])
        tempT.append(tweet['contributors'])
        cursor.execute("INSERT INTO Tweets VALUES(?,?,?,?,?,?,?,?,?,?);",tempT)

        tempU.append(tweet['id_str'])
        tempU.append(tweet['user']['name'])
        tempU.append(tweet['user']['screen_name'])
        tempU.append(tweet['user']['description'])
        tempU.append(tweet['user']['friends_count'])
        cursor.execute("INSERT INTO USER VALUES(?,?,?,?,?);",tempU)

        if tweet['coordinates'] == None:
            continue
        else:
            tempG.append(tweet['id_str'])
            tempG.append(tweet['geo']['type'])
            tempG.append(tweet['geo']['coordinates'][0])
            tempG.append(tweet['geo']['coordinates'][1])
        cursor.execute("INSERT INTO GEO VALUES(?,?,?,?);",tempG)

    except:
        pass

endC1 = time.time()
```

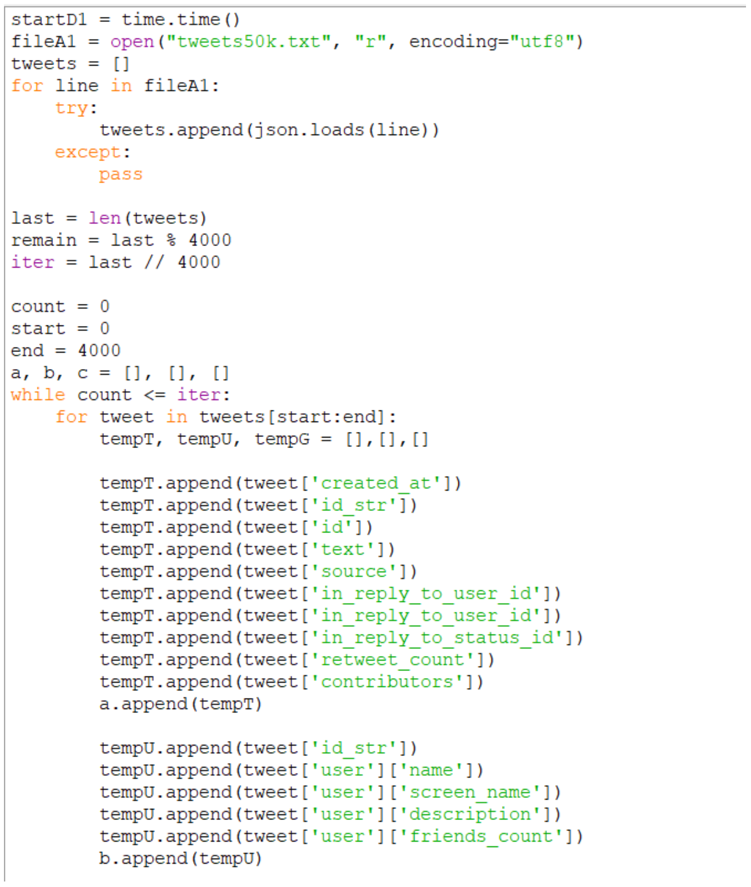
250,000 tweets

```
fileA2 = open("tweets250k.txt", "r",encoding='utf8')

startC1 = time.time()
for i in range(250001):

    try:
        tempT, tempU, tempG = [],[],[]
        allTweets = fileA2.readline()
        tweet = json.loads(allTweets)

        tempT.append(tweet['created_at'])
        tempT.append(tweet['id_str'])
        tempT.append(tweet['id'])
        tempT.append(tweet['text'])
        tempT.append(tweet['source'])
        tempT.append(tweet['in_reply_to_user_id'])
        tempT.append(tweet['in_reply_to_user_id'])
        tempT.append(tweet['in_reply_to_status_id'])
        tempT.append(tweet['retweet_count'])
        tempT.append(tweet['contributors'])
        cursor.execute("INSERT INTO Tweets VALUES(?,?,?,?,?,?,?,?,?,?);",tempT)

        tempU.append(tweet['id_str'])
        tempU.append(tweet['user']['name'])
        tempU.append(tweet['user']['screen_name'])
        tempU.append(tweet['user']['description'])
        tempU.append(tweet['user']['friends_count'])
        cursor.execute("INSERT INTO USER VALUES(?,?,?,?,?);",tempU)

        if tweet['coordinates'] == None:
            continue
        else:
            tempG.append(tweet['id_str'])
            tempG.append(tweet['geo']['type'])
            tempG.append(tweet['geo']['coordinates'][0])
            tempG.append(tweet['geo']['coordinates'][1])
        cursor.execute("INSERT INTO GEO VALUES(?,?,?,?);",tempG)

    except:
        pass

endC1 = time.time()
```



```
= RESTART: D:\Study\Master - Data Science\Study\8. DSC 450 - Database for Analyt
ics\Module 10 - 05.30.2022 - Final\Final Exam\1c - final.py

1c. The run time for 50000 tweets is  2.5642337799072266 seconds
The number of rows in Tweets Table:  (50000,)
The number of rows in Users Table:  (50000,)
The number of rows in Geo Table:  (1121,)

1c. The run time for 250000 tweets is  12.675737142562866 seconds
The number of rows in Tweets Table:  (250000,)
The number of rows in Users Table:  (250000,)
The number of rows in Geo Table:  (5915,)
>>> |
```

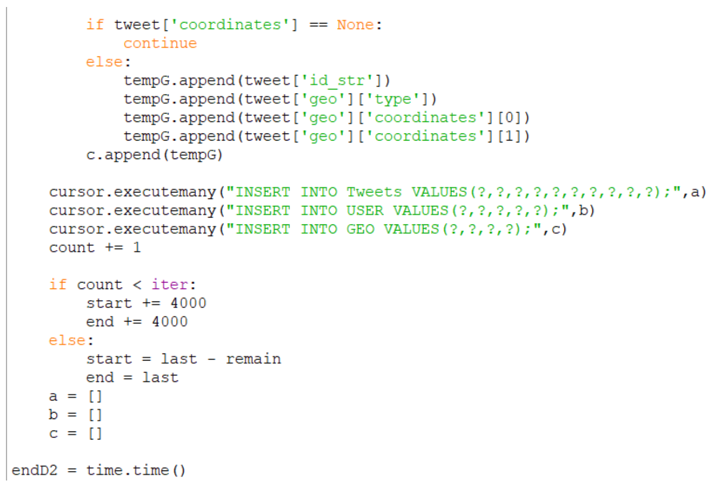
d. Repeat the same step with a batching size of 4000 (i.e. by inserting 4000 rows at a time with executemany instead of doing individual inserts). Since many of the tweets are missing a Geo location, its fine for the batches of Geo inserts to be smaller than 4000.
50,000 tweets



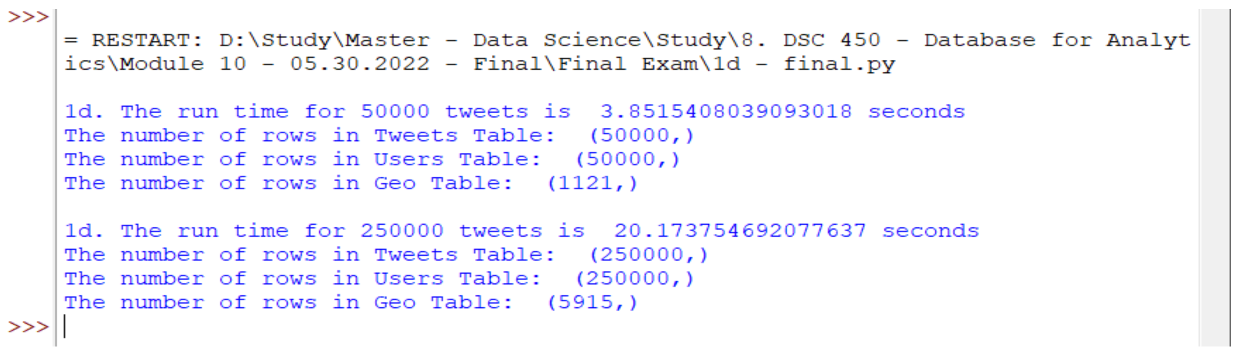
```
startD1 = time.time()
fileA1 = open("tweets50k.txt", "r", encoding="utf8")
tweets = []
for line in fileA1:
    try:
        tweets.append(json.loads(line))
    except:
        pass

last = len(tweets)
remain = last % 4000
iter = last // 4000

count = 0
start = 0
end = 4000
a, b, c = [], [], []
while count <= iter:
    for tweet in tweets[start:end]:
        tempT, tempU, tempG = [],[],[]

        tempT.append(tweet['created_at'])
        tempT.append(tweet['id_str'])
        tempT.append(tweet['id'])
        tempT.append(tweet['text'])
        tempT.append(tweet['source'])
        tempT.append(tweet['in_reply_to_user_id'])
        tempT.append(tweet['in_reply_to_user_id'])
        tempT.append(tweet['in_reply_to_status_id'])
        tempT.append(tweet['retweet_count'])
        tempT.append(tweet['contributors'])
        a.append(tempT)

        tempU.append(tweet['id_str'])
        tempU.append(tweet['user']['name'])
        tempU.append(tweet['user']['screen_name'])
        tempU.append(tweet['user']['description'])
        tempU.append(tweet['user']['friends_count'])
        b.append(tempU)
        if tweet['coordinates'] == None:
            continue
        else:
            tempG.append(tweet['id_str'])
            tempG.append(tweet['geo']['type'])
            tempG.append(tweet['geo']['coordinates'][0])
            tempG.append(tweet['geo']['coordinates'][1])
        c.append(tempG)

    cursor.executemany("INSERT INTO Tweets VALUES(?,?,?,?,?,?,?,?,?,?);",a)
    cursor.executemany("INSERT INTO USER VALUES(?,?,?,?,?);",b)
    cursor.executemany("INSERT INTO GEO VALUES(?,?,?,?);",c)
    count += 1

    if count < iter:
        start += 4000
        end += 4000
    else:
        start = last - remain
        end = last
    a = []
    b = []
    c = []

endD1 = time.time()
```

250,000 tweets

```
startD2 = time.time()
fileA2 = open("tweets250k.txt", "r", encoding="utf8")
tweets = []
for line in fileA2:
    try:
        tweets.append(json.loads(line))
    except:
        pass

last = len(tweets)
remain = last % 4000
iter = last // 4000

count = 0
start = 0
end = 4000
a, b, c = [], [], []
while count <= iter:
    for tweet in tweets[start:end]:
        tempT, tempU, tempG = [],[],[]

        tempT.append(tweet['created_at'])
        tempT.append(tweet['id_str'])
        tempT.append(tweet['id'])
        tempT.append(tweet['text'])
        tempT.append(tweet['source'])
        tempT.append(tweet['in_reply_to_user_id'])
        tempT.append(tweet['in_reply_to_user_id'])
        tempT.append(tweet['in_reply_to_status_id'])
        tempT.append(tweet['retweet_count'])
        tempT.append(tweet['contributors'])
        a.append(tempT)

        tempU.append(tweet['id_str'])
        tempU.append(tweet['user']['name'])
        tempU.append(tweet['user']['screen_name'])
        tempU.append(tweet['user']['description'])
        tempU.append(tweet['user']['friends_count'])
        b.append(tempU)
```

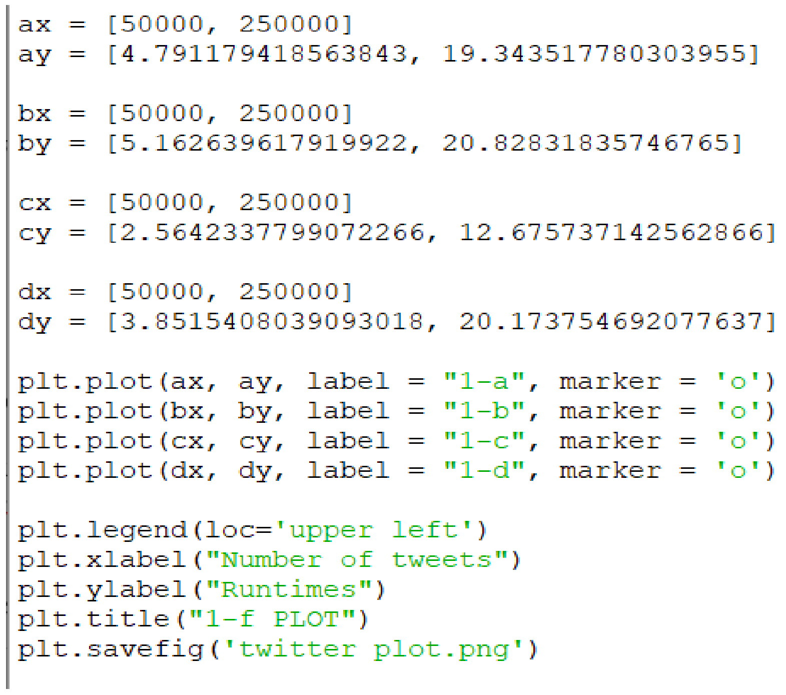


```
        if tweet['coordinates'] == None:
            continue
        else:
            tempG.append(tweet['id_str'])
            tempG.append(tweet['geo']['type'])
            tempG.append(tweet['geo']['coordinates'][0])
            tempG.append(tweet['geo']['coordinates'][1])
        c.append(tempG)

    cursor.executemany("INSERT INTO Tweets VALUES(?,?,?,?,?,?,?,?,?,?);",a)
    cursor.executemany("INSERT INTO USER VALUES(?,?,?,?,?);",b)
    cursor.executemany("INSERT INTO GEO VALUES(?,?,?,?);",c)
    count += 1

    if count < iter:
        start += 4000
        end += 4000
    else:
        start = last - remain
        end = last
    a = []
    b = []
    c = []

endD2 = time.time()
```

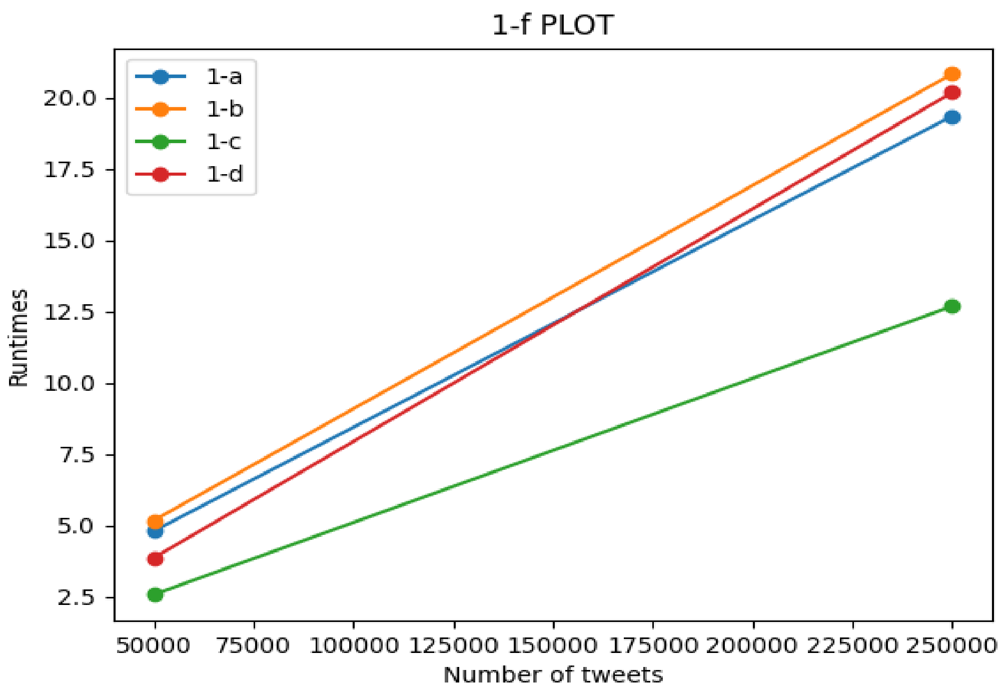


```
>>>
    = RESTART: D:\Study\Master - Data Science\Study\8. DSC 450 - Database for Analyt
    ics\Module 10 - 05.30.2022 - Final\Final Exam\1d - final.py

    1d. The run time for 50000 tweets is  3.8515408039093018 seconds
    The number of rows in Tweets Table:  (50000,)
    The number of rows in Users Table:  (50000,)
    The number of rows in Geo Table:  (1121,)

    1d. The run time for 250000 tweets is  20.173754692077637 seconds
    The number of rows in Tweets Table:  (250000,)
    The number of rows in Users Table:  (250000,)
    The number of rows in Geo Table:  (5915,)
>>>
```

e. Plot the resulting runtimes (# of tweets versus runtimes) using matplotlib for 1-a, 1-b, 1-c, and 1-d. How does the runtime compare?



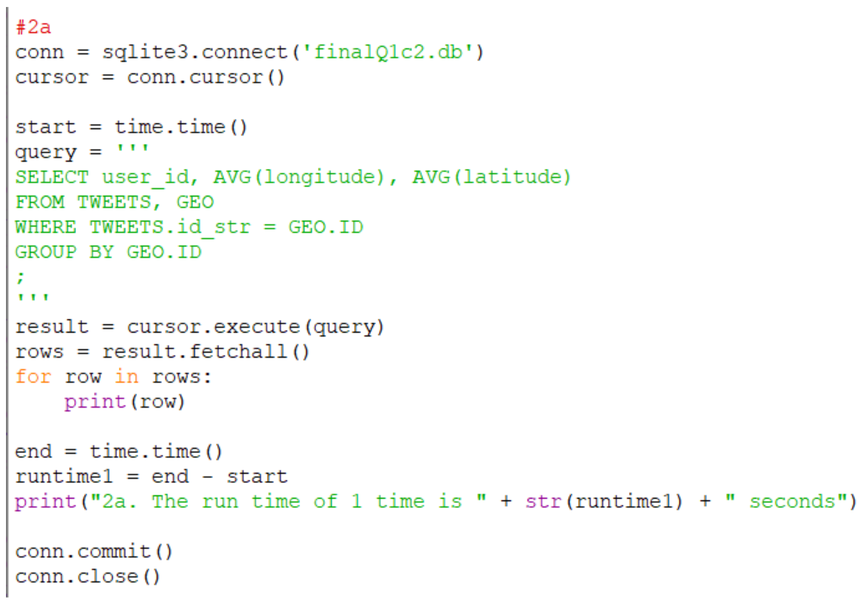
```
ax = [50000, 250000]
ay = [4.791179418563843, 19.343517780303955]

bx = [50000, 250000]
by = [5.162639617919922, 20.82831835746765]

cx = [50000, 250000]
cy = [2.5642337799072266, 12.675737142562866]

dx = [50000, 250000]
dy = [3.8515408039093018, 20.173754692077637]

plt.plot(ax, ay, label = "1-a", marker = 'o')
plt.plot(bx, by, label = "1-b", marker = 'o')
plt.plot(cx, cy, label = "1-c", marker = 'o')
plt.plot(dx, dy, label = "1-d", marker = 'o')

plt.legend(loc='upper left')
plt.xlabel("Number of tweets")
plt.ylabel("Runtimes")
plt.title("1-f PLOT")
plt.savefig('twitter plot.png')
```



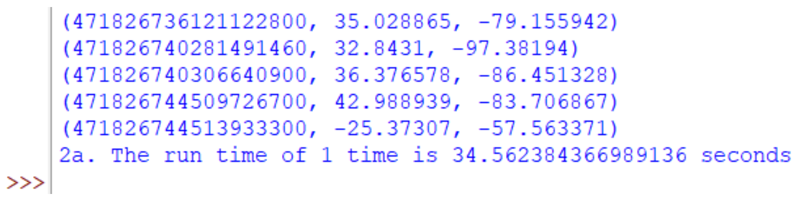
As we can see from the plot above, question 1-c's runtimes is the shortest one, and 1-b is the longest one. Compare 1-a and 1-d, 1-a is faster than 1-d.

Part 2

a. Write and execute a SQL query to find the average longitude and latitude value for each user ID. This query does not need the User table because User ID is a foreign key in the Tweet table. E.g., something like *SELECT UserID, MIN(longitude), MIN(latitude) FROM Tweet, Geo WHERE Tweet.GeoFK = Geo.GeoID GROUP BY UserID;*



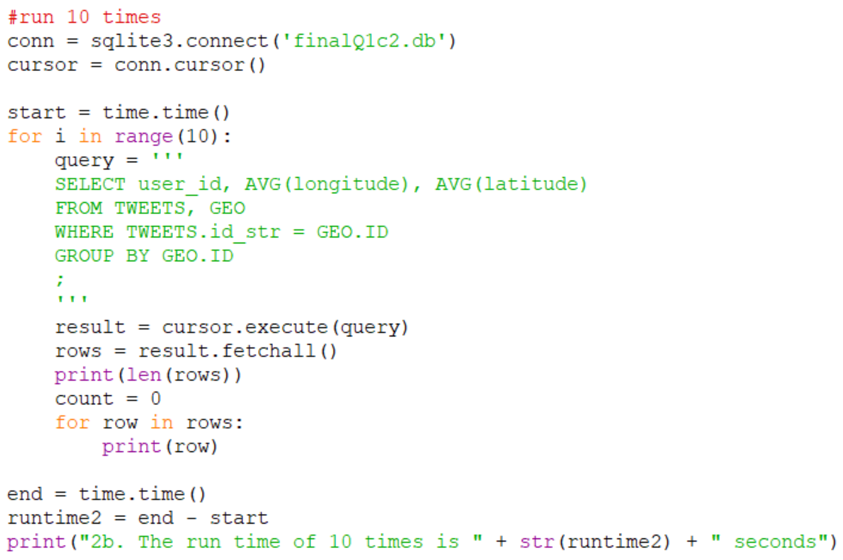
```
#2a
conn = sqlite3.connect('finalQ1c2.db')
cursor = conn.cursor()

start = time.time()
query = '''
SELECT user_id, AVG(longitude), AVG(latitude)
FROM TWEETS, GEO
WHERE TWEETS.id_str = GEO.ID
GROUP BY GEO.ID
;
'''
result = cursor.execute(query)
rows = result.fetchall()
for row in rows:
    print(row)

end = time.time()
runtime1 = end - start
print("2a. The run time of 1 time is " + str(runtime1) + " seconds")

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



```
   (471826736121122800, 35.028865, -79.155942)
   (471826740281491460, 32.8431, -97.38194)
   (471826740306640900, 36.376578, -86.451328)
   (471826744509726700, 42.988939, -83.706867)
   (471826744513933300, -25.37307, -57.563371)
   2a. The run time of 1 time is 34.562384366989136 seconds
>>>
```

b. Re-execute the SQL query in part 2-a 10 times and 50 times and measure the total runtime (just re-run the same exact query multiple times using a for-loop, it is as simple as it looks). Does the runtime scale linearly? (i.e., does it take 10X and 50X as much time?)

Execute 10 times:

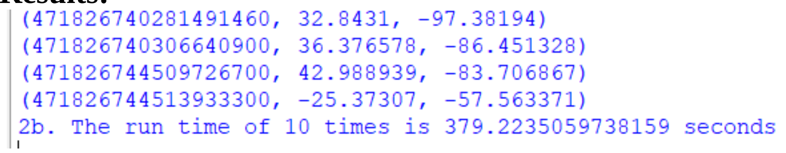


```
#run 10 times
conn = sqlite3.connect('finalQ1c2.db')
cursor = conn.cursor()

start = time.time()
for i in range(10):
    query = '''
    SELECT user_id, AVG(longitude), AVG(latitude)
    FROM TWEETS, GEO
    WHERE TWEETS.id_str = GEO.ID
    GROUP BY GEO.ID
    ;
    '''
    result = cursor.execute(query)
    rows = result.fetchall()
    print(len(rows))
    count = 0
    for row in rows:
        print(row)

end = time.time()
runtime2 = end - start
print("2b. The run time of 10 times is " + str(runtime2) + " seconds")
```

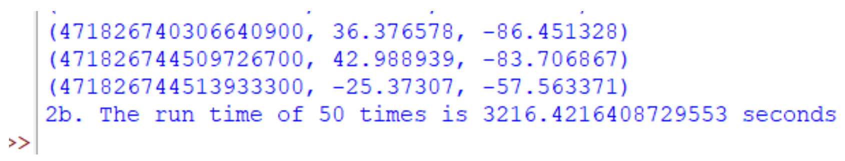
Execute 50 times:

```
#run 50 times
conn = sqlite3.connect('finalQ1c2.db')
cursor = conn.cursor()

start = time.time()
for i in range(50):
    query = '''
    SELECT user_id, AVG(longitude), AVG(latitude)
    FROM TWEETS, GEO
    WHERE TWEETS.id_str = GEO.ID
    GROUP BY GEO.ID
    ;
    '''
    result = cursor.execute(query)
    rows = result.fetchall()
    print(len(rows))
    count = 0
    for row in rows:
        print(row)

end = time.time()
runtime3 = end - start
print("2b. The run time of 50 times is " + str(runtime3) + " seconds")

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

Results:



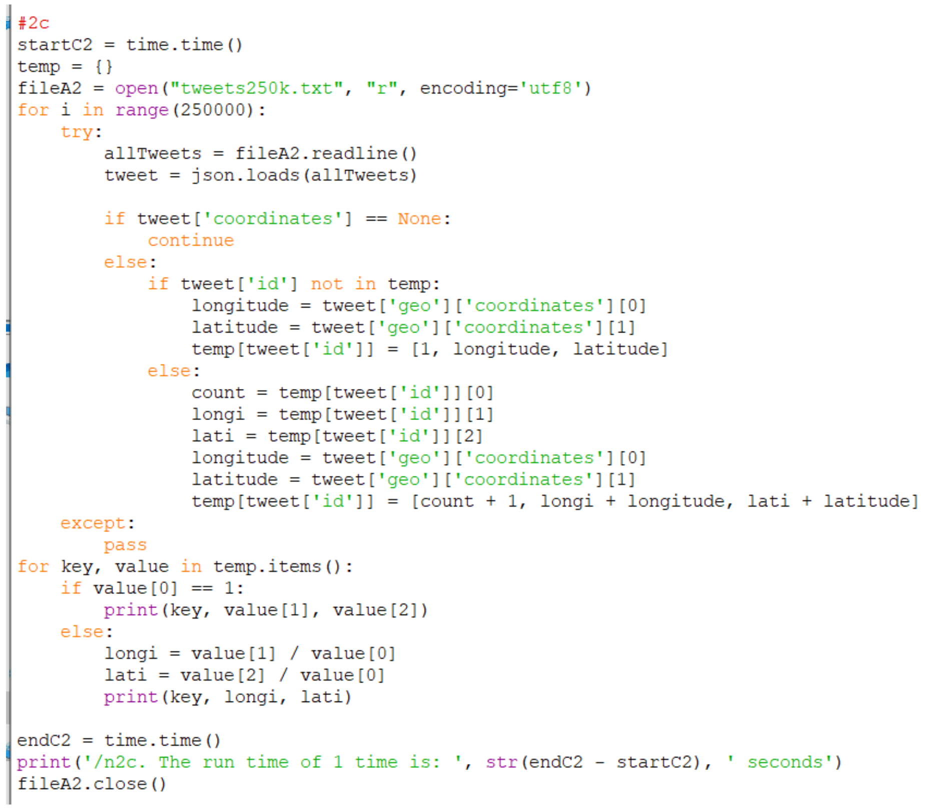
```
(471826740281491460, 32.8431, -97.38194)
(471826740306640900, 36.376578, -86.451328)
(471826744509726700, 42.988939, -83.706867)
(471826744513933300, -25.37307, -57.563371)
2b. The run time of 10 times is 379.2235059738159 seconds
```



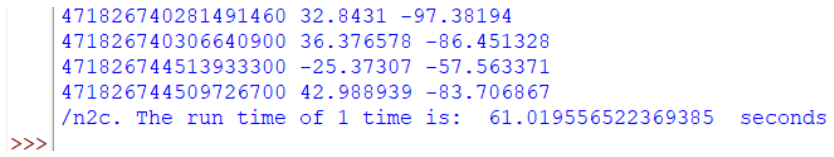
```
(471826740306640900, 36.376578, -86.451328)
(471826744509726700, 42.988939, -83.706867)
(471826744513933300, -25.37307, -57.563371)
2b. The run time of 50 times is 3216.4216408729553 seconds
>>
```

The runtime scales almost linearly.

c. Write the equivalent of the 2-a query in python (without using SQL) by reading it from the file with 250,000 tweets.



```
#2c
startC2 = time.time()
temp = {}
fileA2 = open("tweets250k.txt", "r", encoding='utf8')
for i in range(250000):
    try:
        allTweets = fileA2.readline()
        tweet = json.loads(allTweets)

        if tweet['coordinates'] == None:
            continue
        else:
            if tweet['id'] not in temp:
                longitude = tweet['geo']['coordinates'][0]
                latitude = tweet['geo']['coordinates'][1]
                temp[tweet['id']] = [1, longitude, latitude]
            else:
                count = temp[tweet['id']][0]
                longi = temp[tweet['id']][1]
                lati = temp[tweet['id']][2]
                longitude = tweet['geo']['coordinates'][0]
                latitude = tweet['geo']['coordinates'][1]
                temp[tweet['id']] = [count + 1, longi + longitude, lati + latitude]
    except:
        pass
for key, value in temp.items():
    if value[0] == 1:
        print(key, value[1], value[2])
    else:
        longi = value[1] / value[0]
        lati = value[2] / value[0]
        print(key, longi, lati)

endC2 = time.time()
print('/n2c. The run time of 1 time is: ', str(endC2 - startC2), ' seconds')
fileA2.close()
```



```
471826740281491460 32.8431 -97.38194
471826740306640900 36.376578 -86.451328
471826744513933300 -25.37307 -57.563371
471826744509726700 42.988939 -83.706867
/n2c. The run time of 1 time is:  61.019556522369385  seconds
>>>
```

d. Re-execute the query in part 2-c 10 times and 50 times and measure the total runtime. Does the runtime scale linearly?

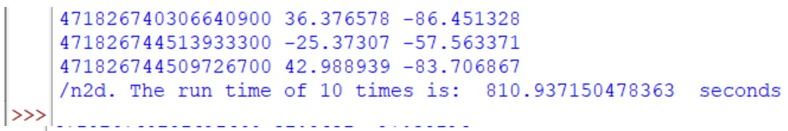
Execute 10 times:

```
##Execute 10 times
startD1 = time.time()
for i in range(10):
    temp = {}
    fileA2 = open("tweets250k.txt", "r", encoding='utf8')
    for i in range(250000):
        try:
            allTweets = fileA2.readline()
            tweet = json.loads(allTweets)

            if tweet['coordinates'] == None:
                continue
            else:
                if tweet['id'] not in temp:
                    longitude = tweet['geo']['coordinates'][0]
                    latitude = tweet['geo']['coordinates'][1]
                    temp[tweet['id']] = [1, longitude, latitude]
                else:
                    count = temp[tweet['id']][0]
                    longi = temp[tweet['id']][1]
                    lati = temp[tweet['id']][2]
                    longitude = tweet['geo']['coordinates'][0]
                    latitude = tweet['geo']['coordinates'][1]
                    temp[tweet['id']] = [count + 1, longi + longitude, lati + latitude]
        except:
            pass
    for key, value in temp.items():
        if value[0] == 1:
            print(key, value[1], value[2])
        else:
            longi = value[1] / value[0]
            lati = value[2] / value[0]
            print(key, longi, lati)

endD1 = time.time()
print('2d. The run time of 10 times is: ', str(endD1 - startD1), ' seconds')
fileA2.close()
```

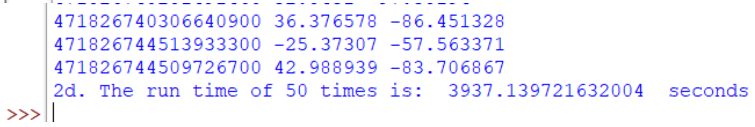
Execute 50 times:

```
##Execute 50 times
startD2 = time.time()
for i in range(50):
    temp = {}
    fileA2 = open("tweets250k.txt", "r", encoding='utf8')
    for i in range(250000):
        try:
            allTweets = fileA2.readline()
            tweet = json.loads(allTweets)

            if tweet['coordinates'] == None:
                continue
            else:
                if tweet['id'] not in temp:
                    longitude = tweet['geo']['coordinates'][0]
                    latitude = tweet['geo']['coordinates'][1]
                    temp[tweet['id']] = [1, longitude, latitude]
                else:
                    count = temp[tweet['id']][0]
                    longi = temp[tweet['id']][1]
                    lati = temp[tweet['id']][2]
                    longitude = tweet['geo']['coordinates'][0]
                    latitude = tweet['geo']['coordinates'][1]
                    temp[tweet['id']] = [count + 1, longi + longitude, lati + latitude]
        except:
            pass
    for key, value in temp.items():
        if value[0] == 1:
            print(key, value[1], value[2])
        else:
            longi = value[1] / value[0]
            lati = value[2] / value[0]
            print(key, longi, lati)

endD2 = time.time()
print('2d. The run time of 50 times is: ', str(endD2 - startD2), ' seconds')
fileA2.close()
```

Result:



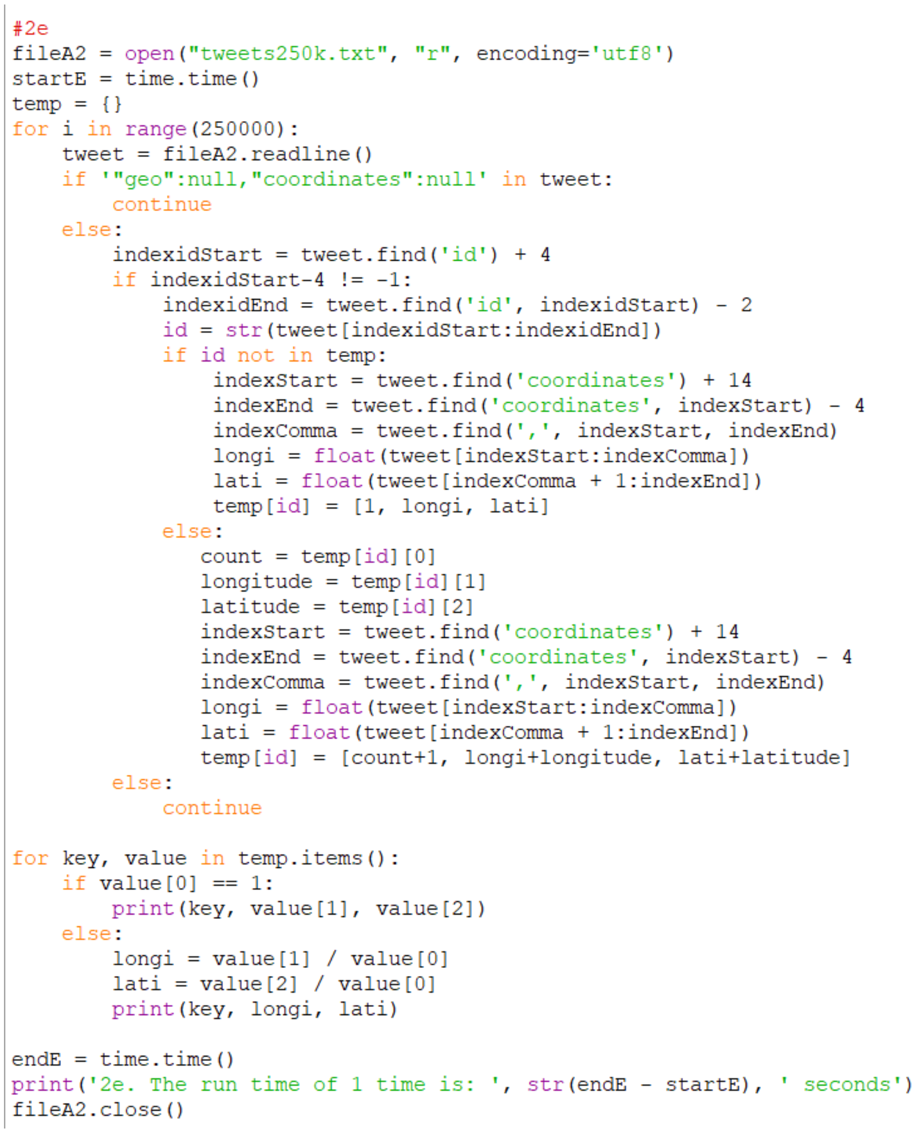
```
471826740306640900 36.376578 -86.451328
471826744513933300 -25.37307 -57.563371
471826744509726700 42.988939 -83.706867
/n2d. The run time of 10 times is:  810.937150478363  seconds
>>>
```



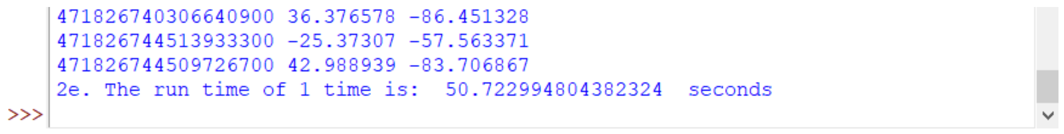
```
471826740306640900 36.376578 -86.451328
471826744513933300 -25.37307 -57.563371
471826744509726700 42.988939 -83.706867
2d. The run time of 50 times is:  3937.139721632004  seconds
>>>
```

The runtime scales almost linearly.

e. Write the equivalent of the 2-a query in python by using regular expressions instead of json.loads(). Do not use json.loads() here. Note that you only need to find userid and geo location (if any) for each tweet, you don't need to parse the whole thing.



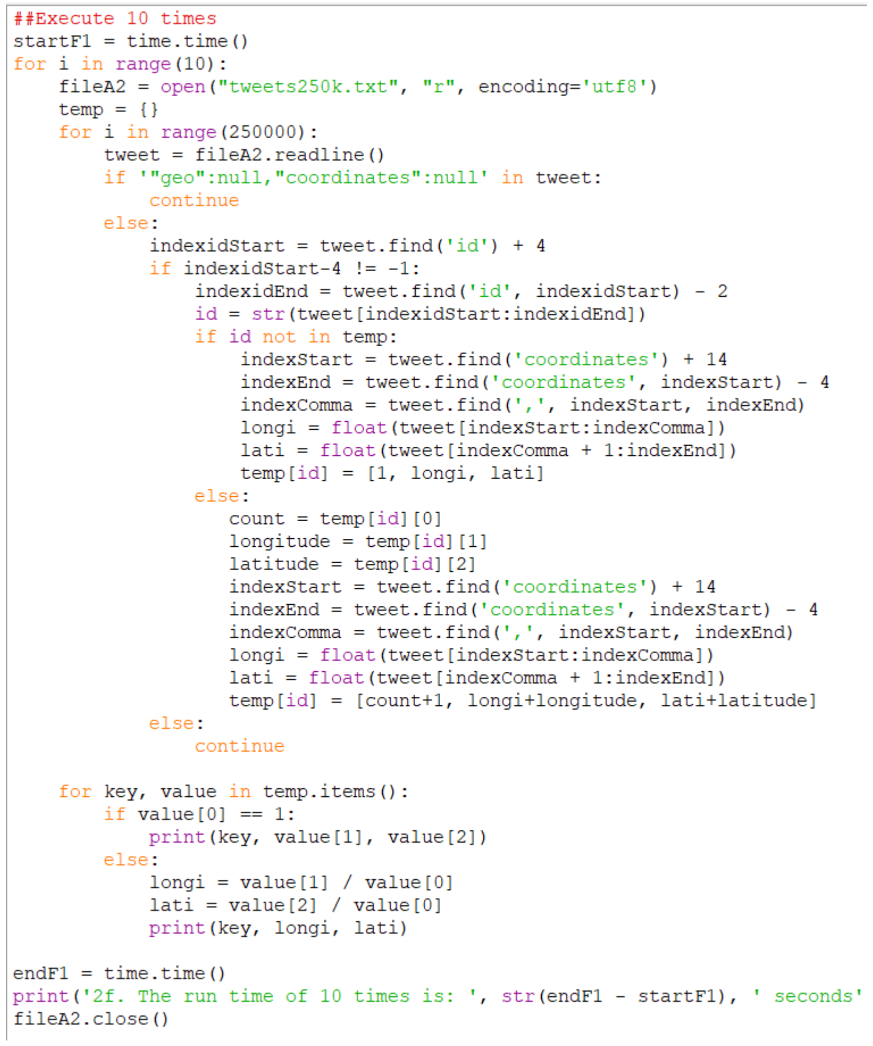
```
#2e
fileA2 = open("tweets250k.txt", "r", encoding='utf8')
startE = time.time()
temp = {}
for i in range(250000):
    tweet = fileA2.readline()
    if '"geo":null,"coordinates":null' in tweet:
        continue
    else:
        indexidStart = tweet.find('id') + 4
        if indexidStart-4 != -1:
            indexidEnd = tweet.find('id', indexidStart) - 2
            id = str(tweet[indexidStart:indexidEnd])
            if id not in temp:
                indexStart = tweet.find('coordinates') + 14
                indexEnd = tweet.find('coordinates', indexStart) - 4
                indexComma = tweet.find(',', indexStart, indexEnd)
                longi = float(tweet[indexStart:indexComma])
                lati = float(tweet[indexComma + 1:indexEnd])
                temp[id] = [1, longi, lati]
            else:
               count = temp[id][0]
               longitude = temp[id][1]
               latitude = temp[id][2]
               indexStart = tweet.find('coordinates') + 14
               indexEnd = tweet.find('coordinates', indexStart) - 4
               indexComma = tweet.find(',', indexStart, indexEnd)
               longi = float(tweet[indexStart:indexComma])
               lati = float(tweet[indexComma + 1:indexEnd])
               temp[id] = [count+1, longi+longitude, lati+latitude]
        else:
            continue

for key, value in temp.items():
    if value[0] == 1:
        print(key, value[1], value[2])
    else:
        longi = value[1] / value[0]
        lati = value[2] / value[0]
        print(key, longi, lati)

endE = time.time()
print('2e. The run time of 1 time is: ', str(endE - startE), ' seconds')
fileA2.close()
```



```
471826740306640900 36.376578 -86.451328
471826744513933300 -25.37307 -57.563371
471826744509726700 42.988939 -83.706867
2e. The run time of 1 time is:  50.722994804382324  seconds
>>>
```

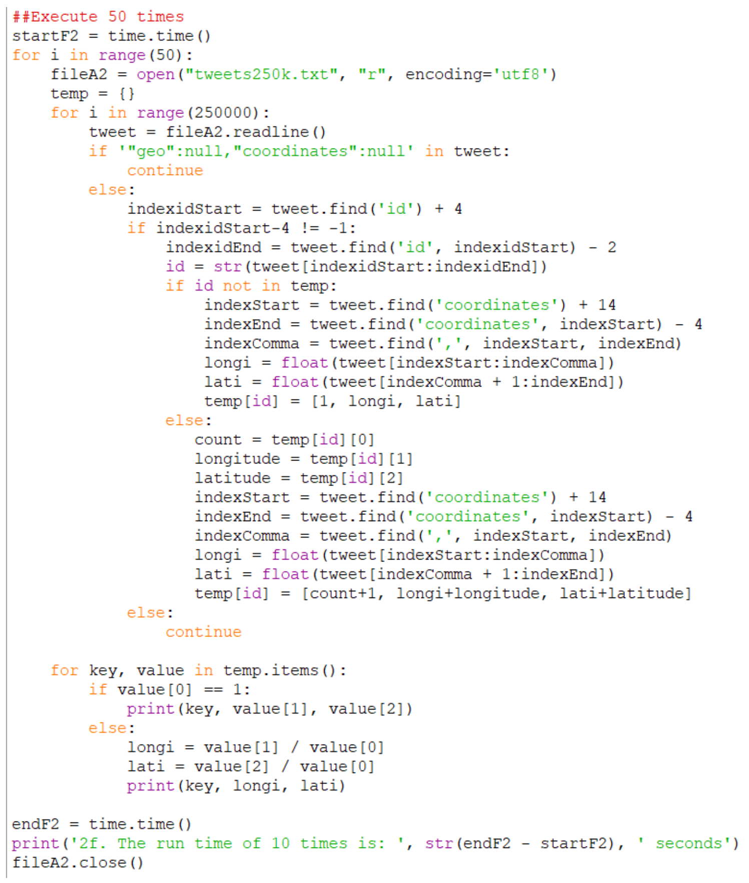
f. Re-execute the query in part 2-e 10 times and 50 times and measure the total runtime. Does the runtime scale linearly?

Execute 10 times:



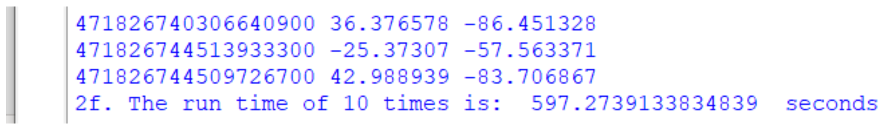
```
##Execute 10 times
startF1 = time.time()
for i in range(10):
    fileA2 = open("tweets250k.txt", "r", encoding='utf8')
    temp = {}
    for i in range(250000):
        tweet = fileA2.readline()
        if '"geo":null,"coordinates":null' in tweet:
            continue
        else:
            indexidStart = tweet.find('id') + 4
            if indexidStart-4 != -1:
                indexidEnd = tweet.find('id', indexidStart) - 2
                id = str(tweet[indexidStart:indexidEnd])
                if id not in temp:
                    indexStart = tweet.find('coordinates') + 14
                    indexEnd = tweet.find('coordinates', indexStart) - 4
                    indexComma = tweet.find(',', indexStart, indexEnd)
                    longi = float(tweet[indexStart:indexComma])
                    lati = float(tweet[indexComma + 1:indexEnd])
                    temp[id] = [1, longi, lati]
                else:
                   count = temp[id][0]
                   longitude = temp[id][1]
                   latitude = temp[id][2]
                   indexStart = tweet.find('coordinates') + 14
                   indexEnd = tweet.find('coordinates', indexStart) - 4
                   indexComma = tweet.find(',', indexStart, indexEnd)
                   longi = float(tweet[indexStart:indexComma])
                   lati = float(tweet[indexComma + 1:indexEnd])
                   temp[id] = [count+1, longi+longitude, lati+latitude]
            else:
                continue

    for key, value in temp.items():
        if value[0] == 1:
            print(key, value[1], value[2])
        else:
            longi = value[1] / value[0]
            lati = value[2] / value[0]
            print(key, longi, lati)

endF1 = time.time()
print('2f. The run time of 10 times is: ', str(endF1 - startF1), ' seconds'
fileA2.close()
```

Execute 50 times:

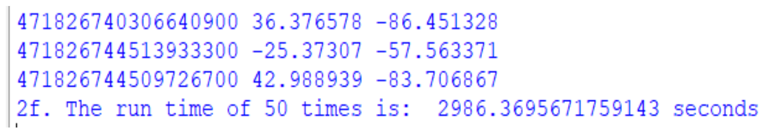


```
##Execute 50 times
startF2 = time.time()
for i in range(50):
    fileA2 = open("tweets250k.txt", "r", encoding='utf8')
    temp = {}
    for i in range(250000):
        tweet = fileA2.readline()
        if '"geo":null,"coordinates":null' in tweet:
            continue
        else:
            indexidStart = tweet.find('id') + 4
            if indexidStart-4 != -1:
                indexidEnd = tweet.find('id', indexidStart) - 2
                id = str(tweet[indexidStart:indexidEnd])
                if id not in temp:
                    indexStart = tweet.find('coordinates') + 14
                    indexEnd = tweet.find('coordinates', indexStart) - 4
                    indexComma = tweet.find(',', indexStart, indexEnd)
                    longi = float(tweet[indexStart:indexComma])
                    lati = float(tweet[indexComma + 1:indexEnd])
                    temp[id] = [1, longi, lati]
                else:
                   count = temp[id][0]
                   longitude = temp[id][1]
                   latitude = temp[id][2]
                   indexStart = tweet.find('coordinates') + 14
                   indexEnd = tweet.find('coordinates', indexStart) - 4
                   indexComma = tweet.find(',', indexStart, indexEnd)
                   longi = float(tweet[indexStart:indexComma])
                   lati = float(tweet[indexComma + 1:indexEnd])
                   temp[id] = [count+1, longi+longitude, lati+latitude]
            else:
                continue

    for key, value in temp.items():
        if value[0] == 1:
            print(key, value[1], value[2])
        else:
            longi = value[1] / value[0]
            lati = value[2] / value[0]
            print(key, longi, lati)

endF2 = time.time()
print('2f. The run time of 10 times is: ', str(endF2 - startF2), ' seconds')
fileA2.close()
```

Result:



```
471826740306640900 36.376578 -86.451328
471826744513933300 -25.37307 -57.563371
471826744509726700 42.988939 -83.706867
2f. The run time of 10 times is:  597.2739133834839  seconds
```

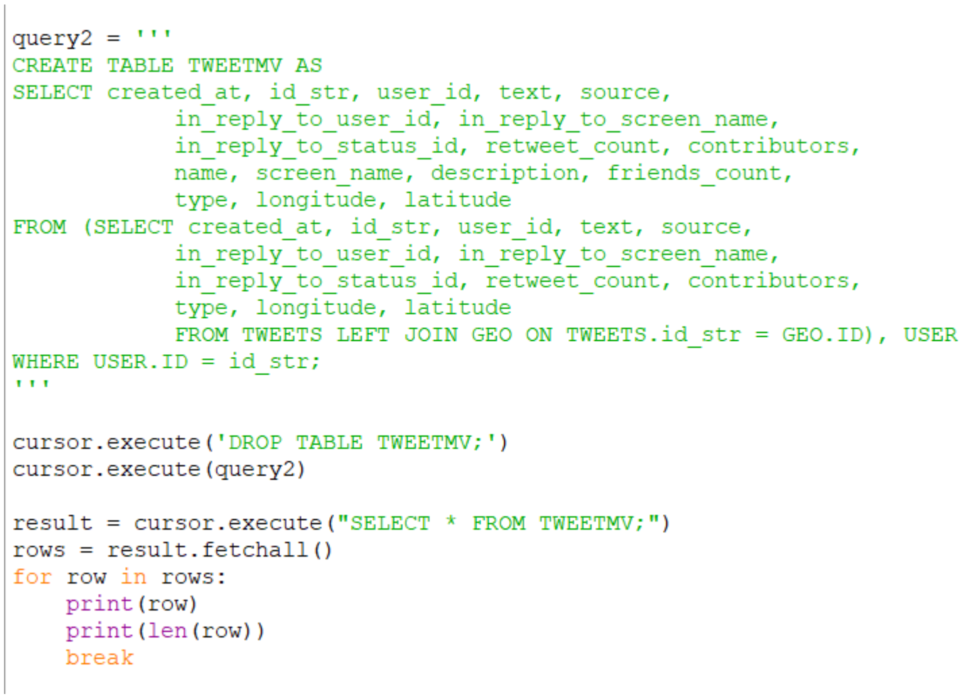


```
471826740306640900 36.376578 -86.451328
471826744513933300 -25.37307 -57.563371
471826744509726700 42.988939 -83.706867
2f. The run time of 50 times is:  2986.3695671759143 seconds
```

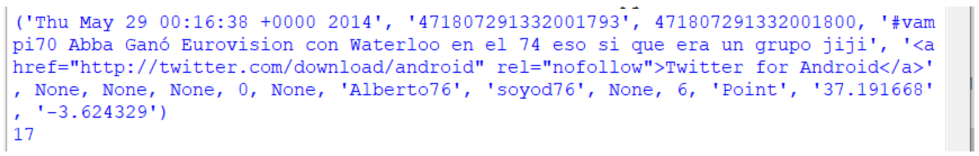
The runtime scales almost linearly.

Part 3

a. Using the database with 250,000 tweets, create a new table that corresponds to the join of all 3 tables in your database, <u>including records without a geo location</u>. This is the equivalent of a materialized view but since SQLite does not support MVs, we will use CREATE TABLE AS SELECT (instead of CREATE MATERIALIZED VIEW AS SELECT).

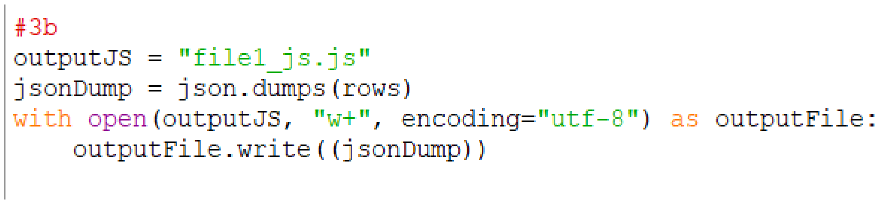


```
query2 = '''
CREATE TABLE TWEETMV AS
SELECT created_at, id_str, user_id, text, source,
            in_reply_to_user_id, in_reply_to_screen_name,
            in_reply_to_status_id, retweet_count, contributors,
            name, screen_name, description, friends_count,
            type, longitude, latitude
FROM (SELECT created_at, id_str, user_id, text, source,
            in_reply_to_user_id, in_reply_to_screen_name,
            in_reply_to_status_id, retweet_count, contributors,
            type, longitude, latitude
            FROM TWEETS LEFT JOIN GEO ON TWEETS.id_str = GEO.ID), USER
WHERE USER.ID = id_str;
'''

cursor.execute('DROP TABLE TWEETMV;')
cursor.execute(query2)

result = cursor.execute("SELECT * FROM TWEETMV;")
rows = result.fetchall()
for row in rows:
    print(row)
    print(len(row))
    break
```



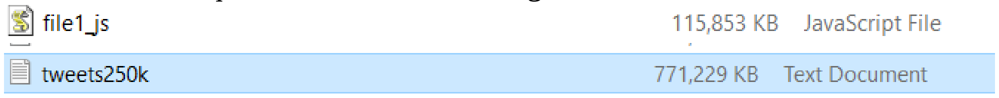
```
('Thu May 29 00:16:38 +0000 2014', '471807291332001793', 471807291332001800, '#vam
pi70 Abba Ganó Eurovision con Waterloo en el 74 eso si que era un grupo jiji', '<a
href="http://twitter.com/download/android" rel="nofollow">Twitter for Android</a>'
, None, None, None, 0, None, 'Alberto76', 'soyod76', None, 6, 'Point', '37.191668'
, '-3.624329')
17
```

b. Export the contents of your table from 3-a into a new JSON file (i.e., create your own JSON file with just the keys you extracted). You do not need to replicate the structure of the input and can come up with any reasonable keys for each field stored in JSON structure (e.g., you can have longitude as "longitude" key when the location is available). How does the file size compare to the original input file?



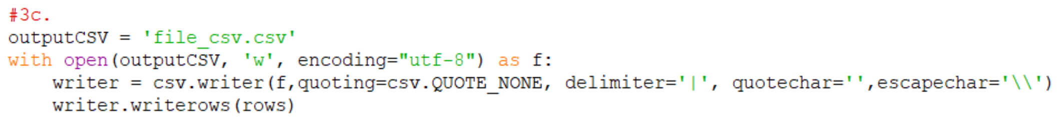
```
#3b
outputJS = "file1_js.js"
jsonDump = json.dumps(rows)
with open(outputJS, "w+", encoding="utf-8") as outputFile:
    outputFile.write((jsonDump))
```

The Json file is quite smaller than the original text file of 250,000 tweets



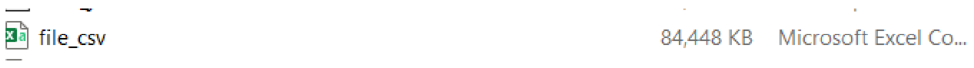
file1_js	115,853 KB	JavaScript File
tweets250k	771,229 KB	Text Document

c. Export the contents of your table from 3-a into a .csv (comma separated value) file. How does the file size compare to the original input file and to the file in 3-b?



```
#3c.
outputCSV = 'file_csv.csv'
with open(outputCSV, 'w', encoding="utf-8") as f:
    writer = csv.writer(f,quoting=csv.QUOTE_NONE, delimiter='|', quotechar='',escapechar='\\')
    writer.writerows(rows)
```

The CSV file is most efficient out of all 3.



file_csv 84,448 KB Microsoft Excel Co...