DSC 450: Database Processing for Large-Scale Analytics

Part 1

We will use one full day worth of tweets as our input (there are total of 4.4M tweets in this file, but we will intentionally use fewer tweets to run this final):
http://rasinsrv07.cstcis.cti.depaul.edu/CSC455/OneDayOfTweets.txt
Execute the following tasks with 50,000 tweets, 200,000 tweets, and 600,000 tweets.

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.

NOTE: Do not call read() or readlines(). That command will attempt to read the entire file which is too much data.

```
import urllib
import json
import time
import sqlite3

#question 1-a
file1 = open("dataset1.txt", "wb")
file2 = open("dataset2.txt", "wb")
file3 = open("dataset3.txt", "wb")
tweetData = 'http://rasinsrv07.cstcis.cti.depaul.edu/CSC455/OneDayOfTweets.txt'
webFD = urllib.request.urlopen(tweetData)

#50000 tweets
start = time.time()
count1 = 0
for line in webFD:
    file1.write(line)
    count1 += 1
    print(count1)
    if count1 == 50000:
        break
end = time.time()

#50000 tweets time (289.9412348270416)
time1 = end - start
print(str(time1) + ' seconds')



#200000 tweets
start = time.time()
count2 = 0
for line in webFD:
    file2.write(line)
    count2 += 1
    print(count2)
    if count2 == 200000:
        break

end = time.time()

#200000 tweets time (800.8470940589905)
time2 = end - start
print(str(time2) + ' seconds')


#600000 tweets
start = time.time()
count3 = 0
for line in webFD:
    file3.write(line)
    count3 += 1
    print(count3)
    if count3 == 600000:
        break

end = time.time()

#600000 tweets time (2002.7778232097626)
time3 = end - start
print(str(time3) + ' seconds')
```

b. For *text*, *in_reply_to_user_id* and *in_reply_to_screenname* in Tweet table and for *screen_name* in User table, find the length of the longest string in the file in 1-a and compare it to your data type size. You only need to do 1-b for 600,000 tweets and you don't need the SQLite database for it (i.e., you can do that based on the file itself).

```
#question 1-b
file1b = open("dataset3.txt", 'r')

x, y, z = 0, 0, 0
for i in range(600000):
    n,m,u = [],[],[]
    allTweets = file1b.readline()
    tweet = json.loads(allTweets)


    replyid = tweet['in_reply_to_user_id']
    replyscreenname = tweet['in_reply_to_screen_name']
    screenname = tweet['user']['screen_name']

    if replyid == None:
        a = 0
    else:
        a = len(str(replyid))

    if replyscreenname == None:
        b = 0
    else:
        b = len(str(replyscreenname))

    if screenname == None:
        c = 0
    else:
        c = len(str(screenname))


    if x < a:
        x = a
    if y < b:
        y = b
    if z < c:
        z = c
    print(i)

print(x)
print(y)
print(z)
```

```
In [97]: print(x)
    ...: print(y)
    ...: print(z)
10
15
15
```

c. 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.

NOTE: If your schema contains a foreign key in the Geo table or relies on TweetID as the primary key for the Geo table, you should fix your schema. Geo entries should be identified based on the location they represent. There should **not** be any "blank" Geo entries such as (ID, None, None, None).

To create the three tables' code shows below, which will be use in the question 1-c, 1-d, 1-e.

```
#question 1-c
conn = sqlite3.connect('ExamQ1c1.db')
cursor = conn.cursor()

dropTweet = """DROP TABLE TWEET"""
cursor.execute(dropTweet)

dropUser = """DROP TABLE USER"""
cursor.execute(dropUser)

dropGeo = """DROP TABLE GEO"""
cursor.execute(dropGeo)


Tweettable = """
CREATE TABLE TWEET(
    created_at               VARCHAR(100),
    id_str                   VARCHAR(30)  primary key UNIQUE,
    user_id                  VARCHAR(30),
    text                     VARCHAR(1000),
    source                   VARCHAR(100),
    in_reply_to_user_id      VARCHAR(20),
    in_reply_to_screen_name  VARCHAR(20),
    in_reply_to_status_id    VARCHAR(50),
    retweet_count            NUMBER,
    contributors             VARCHAR(50)

);
"""
cursor.execute(Tweettable)

Usertable = """
CREATE TABLE USER(
    id             VARCHAR(30) primary key UNIQUE,
    name           VARCHAR(50),
    screen_name    VARCHAR(20),
    description    VARCHAR(50),
    friends_count  VARCHAR(50),

    FOREIGN KEY(id)
        REFERENCEs TWEET(id_str)
);
"""
cursor.execute(Usertable)

Geotable = """
CREATE TABLE GEO(
    ID          VARCHAR(30) primary key UNIQUE,
    type        VARCHAR(30),
    longitude   VARCHAR(10),
    latitude    VARCHAR(100),

    CONSTRAINT gep_fk
        FOREIGN KEY (ID)
            REFERENCES TWEET (id_str)

);
"""
cursor.execute(Geotable)
```

For 50000 tweets:

```
tweetData = 'http://rasinsrv07.cstcis.cti.depaul.edu/CSC455/OneDayOfTweets.txt'
webFD = urllib.request.urlopen(tweetData)

#50000 tweets
start = time.time()
count = 0
for line in webFD:
    try:
        tempT, tempU, tempG = [],[],[]
        tweet = json.loads(line)
        count += 1
        print(count)

        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_screen_name'])
            tempT.append(tweet['in_reply_to_status_id'])
            tempT.append(tweet['retweet_count'])
            tempT.append(tweet['contributors'])
            cursor.execute("INSERT INTO TWEET 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

end = time.time()

#50000 tweets time (343.7849521636963)
time1 = end - start
print(str(time1) + ' seconds')
```

For 200000 tweets:

```
tweetData = 'http://rasinsrv07.cstcis.cti.depaul.edu/CSC455/OneDayOfTweets.txt'
webFD = urllib.request.urlopen(tweetData)

#200000 tweets
start = time.time()
count = 0
for line in webFD:
    try:
        tempT, tempU, tempG = [],[],[]
        tweet = json.loads(line)
        count += 1
        print(count)

        if count == 200001:
            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_screen_name'])
            tempT.append(tweet['in_reply_to_status_id'])
            tempT.append(tweet['retweet_count'])
            tempT.append(tweet['contributors'])
            cursor.execute("INSERT INTO TWEET 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

end = time.time()

#200000 tweets time (1558.7821018695831)
time2 = end - start
print(str(time2) + ' seconds')
```

For 600000 tweets:

```
tweetData = 'http://rasinsrv07.cstcis.cti.depaul.edu/CSC455/OneDayOfTweets.txt'
webFD = urllib.request.urlopen(tweetData)

#50000 tweets
start = time.time()
count = 0
for line in webFD:
    try:
        tempT, tempU, tempG = [],[],[]
        tweet = json.loads(line)
        count += 1
        print(count)

        if count == 600001:
            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_screen_name'])
            tempT.append(tweet['in_reply_to_status_id'])
            tempT.append(tweet['retweet_count'])
            tempT.append(tweet['contributors'])
            cursor.execute("INSERT INTO TWEET 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

end = time.time()

#600000 tweets time (3329.1564939022064)
time3 = end - start
print(str(time3) + ' seconds')
```

d. 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-c, but reading tweets from your file, not from the web.
This is the code for 50000 tweets, for the rest of two files of code in the code file.

After created three tables, the code has been shown in question 1-c, for 50000 tweets:

```
file1b = open("dataset1.txt", 'r')

start = time.time()

for i in range(50001):
    try:
        tempT, tempU, tempG = [],[],[]
        allTweets = file1b.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_screen_name'])
        tempT.append(tweet['in_reply_to_status_id'])
        tempT.append(tweet['retweet_count'])
        tempT.append(tweet['contributors'])
        cursor.execute("INSERT INTO TWEET 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)
        print(i)
    except:
        pass


end = time.time()
timeInsert1 = end - start
# 3.214629888534546 seconds
print(str(timeInsert1) + ' seconds')
```

For 200000 tweets:

```
file1b = open("dataset2.txt", 'r')

start = time.time()

for i in range(200001):
    try:
        tempT, tempU, tempG = [],[],[]
        allTweets = file1b.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_screen_name'])
        tempT.append(tweet['in_reply_to_status_id'])
        tempT.append(tweet['retweet_count'])
        tempT.append(tweet['contributors'])
        cursor.execute("INSERT INTO TWEET 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)
        print(i)
    except:
        pass


end = time.time()
timeInsert2 = end - start
# 13.039237022399902 seconds
print(timeInsert2)
```

For 600000 tweets:

```
file1b = open("dataset3.txt", 'r')

start = time.time()

for i in range(600001):
    try:
        tempT, tempU, tempG = [],[],[]
        allTweets = file1b.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_screen_name'])
        tempT.append(tweet['in_reply_to_status_id'])
        tempT.append(tweet['retweet_count'])
        tempT.append(tweet['contributors'])
        cursor.execute("INSERT INTO TWEET 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)
        print(i)
    except:
        pass


end = time.time()
timeInsert3 = end - start
# 39.804277181625366 seconds
print(str(timeInsert3) + ' seconds')
```

e. Repeat the same step with a batching size of 1000 (i.e. by inserting 1000 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 (however many geo locations were accumulated for each 1000 tweets).

After create the table, the code shows below for 50000 tweets:

```
file1b = open("dataset1.txt", 'r')

start = time.time()

for i in range(50):
    try:
        a, b, c = [],[],[]
        for j in range(1000):
            tempT, tempU, tempG = [],[],[]
            allTweets = file1b.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_screen_name'])
            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 TWEET VALUES(?,?,?,?,?,?,?,?,?,?);",a)
        cursor.executemany("INSERT INTO USER VALUES(?,?,?,?,?);",b)
        cursor.executemany("INSERT INTO GEO VALUES(?,?,?,?);",c)
    except:
        pass

end = time.time()
timeInsert1 = end - start
# 9.228225946426392 seconds
print(str(timeInsert1) + ' seconds')
```

For 200000 tweets:

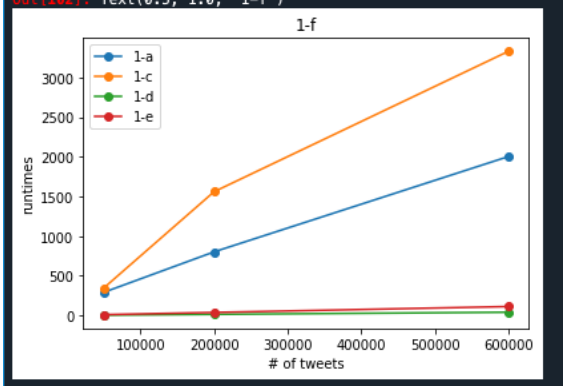
```
file1b = open("dataset2.txt", 'r')

start = time.time()

for i in range(200):
    try:
        a, b, c = [],[],[]
        for j in range(1000):
            tempT, tempU, tempG = [],[],[]
            allTweets = file1b.readline()
            tweet = json.loads(allTweets)
            m = i*1000 + j
            print(str(m))

            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_screen_name'])
            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 TWEET VALUES(?,?,?,?,?,?,?,?,?,?);",a)
        cursor.executemany("INSERT INTO USER VALUES(?,?,?,?,?);",b)
        cursor.executemany("INSERT INTO GEO VALUES(?,?,?,?);",c)
    except:
        pass

end = time.time()
timeInsert1 = end - start
# 36.993664026260376 seconds
print(str(timeInsert1) + ' seconds')
```

For 600000 tweets:

```
file1b = open("dataset3.txt", 'r')

start = time.time()

for i in range(600):
    try:
        a, b, c = [],[],[]
        for j in range(1000):
            tempT, tempU, tempG = [],[],[]
            allTweets = file1b.readline()
            tweet = json.loads(allTweets)
            m = i*1000 + j
            print(str(m))

            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_screen_name'])
            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 TWEET VALUES(?,?,?,?,?,?,?,?,?,?);",a)
        cursor.executemany("INSERT INTO USER VALUES(?,?,?,?,?);",b)
        cursor.executemany("INSERT INTO GEO VALUES(?,?,?,?);",c)
    except:
        pass

end = time.time()
timeInsert1 = end - start
# 111.64353394508362 seconds
print(str(timeInsert1) + ' seconds')
```

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

```
...: import matplotlib.pyplot as plt
...:
...: plt.plot(ax, ay, label = "1-a", marker = 'o')
...: plt.plot(cx, cy, label = "1-c", marker = 'o')
...: plt.plot(dx, dy, label = "1-d", marker = 'o')
...: plt.plot(ex, ey, label = "1-e", marker = 'o')
...:
...: plt.legend(loc='upper left')
...: plt.xlabel("# of tweets")
...: plt.ylabel("runtimes")
...: plt.title("1-f")
Out[102]: Text(0.5, 1.0, '1-f')
```



Question 1-d's runtimes is the shortest one, then 1-e. Compare 1-a and 1-c, 1-a is faster than 1-c.

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, AVG(longitude), AVG(latitude) FROM Tweet, Geo WHERE Tweet.GeoID = Geo.GeoID;*

```
query = """
SELECT user_id, (SELECT AVG(longitude)
                FROM GEO
                WHERE TWEET.id_str = GEO.ID
                GROUP BY GEO.ID), (SELECT  AVG(latitude)
                                   FROM GEO
                                   WHERE TWEET.id_str = GEO.ID
                                   GROUP BY GEO.ID)
FROM TWEET;
"""
result = cursor.execute(query)
for row in result.fetchall():
    print(row)
```

b. Re-execute the query in part 2-a 10 times and 100 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 100X as much time?)

```
running time of 1 times is 69.02033996582031 seconds
running time of 10 times is 692.867840051651 seconds
running time of 100 times is 6908.57946538215 seconds
```

c. Write the equivalent of the 2-a query in python (without using SQL) by reading it from the file with 600,000 tweets.

```
#part 2-c
file1b = open("dataset3.txt", 'r')

start = time.time()
temp = {}
for i in range(600000):
    try:
        allTweets = file1b.readline()
        tweet = json.loads(allTweets)

        #print(i)
        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

#14472
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)

end = time.time()
timeInsert3 = end - start
#26.564638137817383
print("running time of 1 times is " + str(timeInsert3) + " seconds")
```

d. Re-execute the query in part 2-c 10 times and 100 times and measure the total runtime. Does the runtime scale linearly?

```
running time of 1 times is 26.564638137817383 seconds
```

```
running time of 10 times is 273.7922170162201 seconds
```

```
running time of 100 times is 2738.15487564854 seconds
```

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.

```
#part 2-e
file1b = open("dataset3.txt", 'r')

start = time.time()

temp = {}
for i in range(600000):

    tweet = file1b.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


#14472
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)


end = time.time()
timeInsert3 = end - start
#9.229596853256226
print("running time of 1 times is " + str(timeInsert3) + " seconds")
```

f. Re-execute the query in part 2-e 10 times and 100 times and measure the total runtime. Does the runtime scale linearly?

```
running time of 1 times is 9.229596853256226 seconds
```

```
running time of 10 times is 92.98136281967163 seconds
```

```
running time of 100 times is 957.8620500564575 seconds
```

Part 3

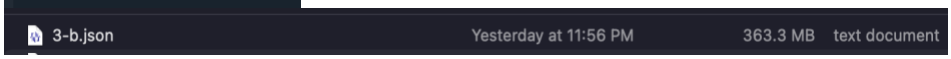
a. Using the database with 600,000 tweets, create a new table that corresponds to the pre-join of all 3 tables in your database, including records without a geo location. This is the equivalent of a materialized view but since SQLite does not support MVs, we will use CREATE TABLE AS SELECT (instead of CREATE MATERIALIZED VIEW AS SELECT).

```
#part 3-a
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 TWEET LEFT JOIN GEO ON TWEET.id_str = GEO.ID), USER
WHERE USER.ID = id_str;
"""
result = cursor.execute(query2)

query3 = """
SELECT *
FROM TWEETMV;
"""

result = cursor.execute(query3)
for row in result.fetchall():
    print(row)
```

```
In [142]: result = cursor.execute(query3)
     ...: for row in result.fetchall():
     ...:     print(row)
     ...:     break
('Thu May 29 01:04:03 +0000 2014', '471819224126881793', '471819224126881800', '@radranchito on', '<a href="http://twitter.com/
download/android" rel="nofollow">Twitter for Android</a>', '164431210', 'radranchito', '471819054336856060', 0, None, 'Leslie',
'lesten_', 'you look so cool.♡ x x x x', '106', None, None, None)
```

b. Export the contents of your table from 3-a into a new JSON file. 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 present). How does the file size compare to the original input file?

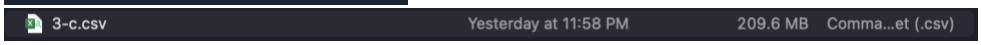
```
#question 3-b
import pandas as pd
df = pd.read_csv (r'3-c.csv')
df.to_json(r'3-b.json')
```



The file size is 363.3MB.

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

```
#question 3-c
import csv
file3c = csv.writer(open("3-c.csv", 'w'))
for row in result.fetchall():
    file3c.writerow(row)
```



The file size is 209.6.mb, which is smaller than the .json file.