Name: Jinhan Cheng
UNI: jc4834

STEP 2:

```
import os
from json import load
from twitter_pb2 import *
os.chdir('/Users/hhanchan/csds-material')

tweets=Tweets()
f=open("twitter.pb")
tweets.ParseFromString(f.read())
f.close()

count1=0
for i in tweets.tweets:
    if i.is_delete == True:
        count1 += 1
print(count1)

count2=0
for i in tweets.tweets:
    if i.insert.reply_to > 0:
        count2 += 1
print(count2)

count3=0
for i in tweets.tweets:
    if i.is_delete == False:
        count3 += 1
print(count3)

count={}
for i in tweets.tweets:
    if i.is_delete == False:
        count[str(i.insert.uid)]=0
for i in tweets.tweets:
    if i.is_delete == False:
        if count[str(i.insert.uid)] < 1:
            count[str(i.insert.uid)] = 1
        else:
            count[str(i.insert.uid)] += 1
x=sorted(count.items(), key=lambda d:d[1], reverse=True)
x[0:5:1]
```

```
In [64]: count1=0
    ...: for i in tweets.tweets:
    ...:     if i.is_delete == True:
    ...:         count1 += 1
    ...:
    ...: print(count1)
1554

In [65]: count2=0
    ...: for i in tweets.tweets:
    ...:     if i.insert.reply_to > 0:
    ...:         count2 += 1
    ...:
    ...: print(count2)
2531

In [66]: count3={}
    ...: for i in tweets.tweets:
    ...:     if i.is_delete == False:
    ...:         count3[str(i.insert.uid)]=0
    ...:
    ...: for i in tweets.tweets:
    ...:     if i.is_delete == False:
    ...:         if count3[str(i.insert.uid)] < 1:
    ...:             count3[str(i.insert.uid)] = 1
    ...:         else:
    ...:             count3[str(i.insert.uid)] += 1
    ...:
    ...: x=sorted(count3.items(), key=lambda d:d[1], reverse=True)
    ...: x[0:5:1]
Out[66]:
[('1269521828', 5),
 ('392695315', 4),
 ('424808364', 3),
 ('1706901902', 3),
 ('145396534', 2)]
```

1. The number of deleted messages in the dataset.
Answer: 1554

2. The number of tweets that are replies to another tweet.
Answer: 2531

3. The five user IDs (field name: uid) that have tweeted the most
Answer: 1269521828, 392695315, 424808364, 1706901902, 145396534

STEP 3:

```
Last login: Sun Feb 11 14:59:55 on ttys003
➜  ~ git:(master) ✗ cd csds-material
➜  csds-material git:(ewu-csds) ✗ sqlite3 twitter.db
SQLite version 3.19.3 2017-06-27 16:48:08
Enter ".help" for usage hints.
sqlite> .tables
coords        place_coords  places        tweets
sqlite> .schema tweets
CREATE TABLE tweets (
 is_delete bool,
 id bigint,
 uid bigint,
 truncated bool,
 text text,
 reply_to bigint,
 reply_to_name text,
 favorite_count int,
 source text,
 retweeted bool,
 possibly_sensitive bool,
 lang text,
 created_at varchar(64),
 filter_level text
);
sqlite> select count(is_delete) from tweets where is_delete=1;
1554
sqlite> select count(reply_to) from tweets where reply_to!=0;
2531
sqlite> select uid, count(*) as count
   ...> from tweets
   ...> where is_delete=0
   ...> group by uid
   ...> order by count desc
   ...> limit 5;
1269521828|5
392695315|4
424808364|3
1706901902|3
23991910|2
sqlite>
```

1. The number of deleted messages in the dataset.
Answer: 1554

2. The number of tweets that are replies to another tweet.
Answer: 2531

3. The five user IDs (field name: uid) that have tweeted the most
Answer: 1269521828, 392695315, 424808364, 1706901902, 23991910

STEP 4:

```
vagrant@precise64:/vagrant$ mongo lab2
MongoDB shell version: 3.2.19
connecting to: lab2
> db.tweets.find({"delete":{$exists:1}}).count()
1554
>
> db.tweets.find({"in_reply_to_status_id":{$ne:null}}).count()
2531
> db.tweets.aggregate([{"$match":{"created_at":{"$exists":1}}},{$group:
{_id:"$user.id",count:{$sum:1}}},{$sort:{"count":-1}},{$limit:5}])
{ "_id" : 1269521828, "count" : 5 }
{ "_id" : 392695315, "count" : 4 }
{ "_id" : 424808364, "count" : 3 }
{ "_id" : 1706901902, "count" : 3 }
{ "_id" : 578015986, "count" : 2 }
>
```

1. The number of deleted messages in the dataset.
Answer: 1554

2. The number of tweets that are replies to another tweet.
Answer: 2531

3. The five user IDs (field name: uid) that have tweeted the most
Answer: 1269521828, 392695315, 424808364, 1706901902, 578015986

Reflection Questions

1. Read the schema and protocol buffer definition files. What are the main differences between the two? Are there any similarities?

   Answer:

   The main differences between the two are that the protocol buffer definition is from the twitter.json file that need more effect to clean the data. However the schema file is much easier to handle because it is already processed.

   For the similarity, the types of the data under their categories are the same.

2. Describe one question that would be easier to answer with protocol buffers than via a SQL query.

   Answer:

As for social network such as instagram, there are posts of a certain netuser, and there are replys to this user or the post is a reply to other users. We want to count the number of replys under every original post. Doing this with python would be easier to write some code, however it would be hard to process using SQL.

3. Describe one question that would be easier to answer with MongoDB than via a SQL query.

   Answer:

   Using social network for example again, to see how many immediately total replys or post of your instagram account in the database. Using MongoDB it can record this dynamic procedure while SQL need a period time to update the data.

4. Describe one question that would be easier to answer via a SQL query than using MongoDB.

   Answer:

   Using SQL would be easier in finding the relationship between data. We could interpret the relationship by simply select the two queries and combine them.

5. What fields in the original JSON structure would be difficult to convert to relational database schemas?

   Answer:

   The field of building reply chains in some social network would be difficult to convert into relational database schemas because it require some manual way to build this relationship and cannot be done by simply way.

6. In terms of lines of code, when did various approaches shine? Think about the challenges of defining schemas, loading and storing the data, and running queries.

   Answer:

   MongoDB shines when importing the json file but it didn't have a schema; SQL needs more code to import json but it has schema; Protocol Buffers has long lines of code that we say it needs more pre-processing, but it would work well then. We shall choose different method to process data under special conditions.

7. What other metrics (e.g., time to implement, code redundancy, etc.) can we use to compare these different approaches? Which system is better by those measures?

   Answer:

   We could use flexibility to compare these different approaches. While SQL might need lots of local memory which constrains its flexibility, MongoDB is good at adding new data instantly under distributed system.

   As for the concern of transactions, it's better to use SQL because it could use to describe what happens during a certain period of time more reliably; using MongoDB is not the best option to deal with complex transactions.

8. How long did this lab take you? We want to make sure to target future labs to not take too much of your time.

   Answer:

   This lab takes me 13h.