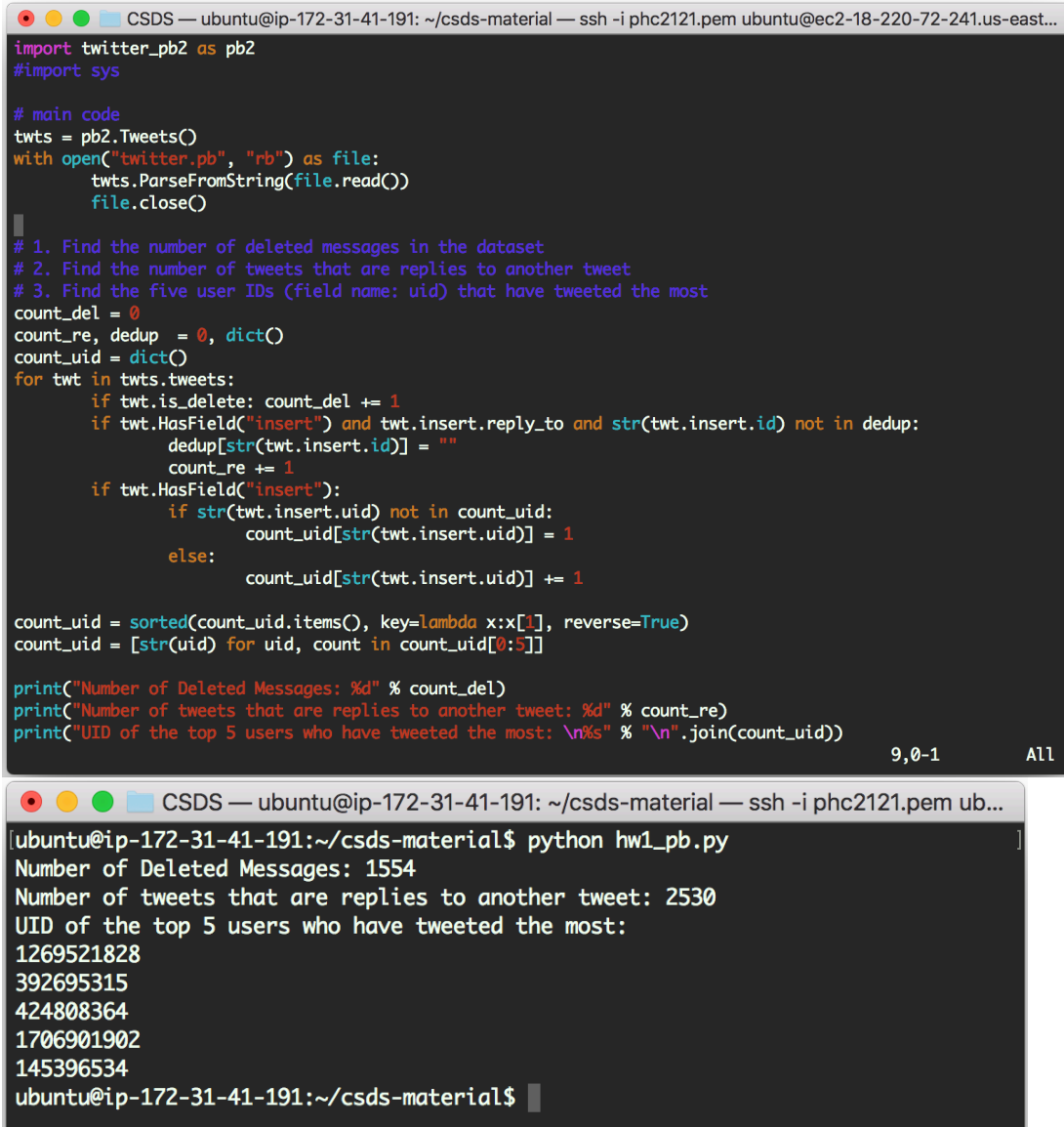


**Name: Pin-Hao Chen | UNI: phc2121**

### Results of Analyses

- Protocol Buffers

The python program goes through the data once and determine the 3 small questions as the 2<sup>nd</sup> image. The 1<sup>st</sup> answer is 1,554, the 2<sup>nd</sup> answer is 2,530, and the 3<sup>rd</sup> answer is the remaining 5 uids.



The first terminal screenshot shows the execution of a Python script named `hw1_pb.py`. The script reads a file `twitter.pb` and processes the data to answer three questions: 1. Find the number of deleted messages in the dataset. 2. Find the number of tweets that are replies to another tweet. 3. Find the five user IDs (field name: uid) that have tweeted the most. The script uses a dictionary to track the number of replies and a list to track the top 5 user IDs. The output is printed to the console.

```
import twitter_pb2 as pb2
#import sys

# main code
twts = pb2.Tweets()
with open("twitter.pb", "rb") as file:
    twts.ParseFromString(file.read())
    file.close()

# 1. Find the number of deleted messages in the dataset
# 2. Find the number of tweets that are replies to another tweet
# 3. Find the five user IDs (field name: uid) that have tweeted the most
count_del = 0
count_re, dedup = 0, dict()
count_uid = dict()
for twt in twts.tweets:
    if twt.is_delete: count_del += 1
    if twt.HasField("insert") and twt.insert.reply_to and str(twt.insert.id) not in dedup:
        dedup[str(twt.insert.id)] = ""
        count_re += 1
    if twt.HasField("insert"):
        if str(twt.insert.uid) not in count_uid:
            count_uid[str(twt.insert.uid)] = 1
        else:
            count_uid[str(twt.insert.uid)] += 1

count_uid = sorted(count_uid.items(), key=lambda x:x[1], reverse=True)
count_uid = [str(uid) for uid, count in count_uid[0:5]]

print("Number of Deleted Messages: %d" % count_del)
print("Number of tweets that are replies to another tweet: %d" % count_re)
print("UID of the top 5 users who have tweeted the most: \n%s" % "\n".join(count_uid))
```

The second terminal screenshot shows the output of the script:

```
ubuntu@ip-172-31-41-191:~/csds-material$ python hw1_pb.py
Number of Deleted Messages: 1554
Number of tweets that are replies to another tweet: 2530
UID of the top 5 users who have tweeted the most:
1269521828
392695315
424808364
1706901902
145396534
ubuntu@ip-172-31-41-191:~/csds-material$
```

- Database Records – SQLite3

For the 1<sup>st</sup> SQL query, the answer is 1,554.

For the 2<sup>nd</sup> SQL query, the answer is 2,530.

For the 3<sup>rd</sup> SQL query, the answer is the remaining 5 uids.

```
CSDS — ubuntu@ip-172-31-41-191: ~/csds-material — ssh -i phc2121.pem ubuntu@ec2-18-220-72-241.us-east...

-- 1. the number of deleted messages in the dataset
select count('is_delete') from tweets where is_delete = 1;

-- 2. the number of tweets that are replies to another tweet
-- select count('reply_to') from tweets where reply_to != 0;
select count(distinct id) from tweets where reply_to != 0;

-- 3. the five user uid that have tweeted the most
select uid from tweets where text != "" group by uid order by count(distinct id) DESC limit 5;

CSDS — ubuntu@ip-172-31-41-191: ~/csds-material — ssh -i phc2121.pem ub...

[ubuntu@ip-172-31-41-191:~/csds-material$ sqlite3 twitter.db
SQLite version 3.8.2 2013-12-06 14:53:30
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> .read hw1_sql.sql
1554
2530
1269521828
392695315
424808364
1706901902
23991910

sqlite>
```

- MongoDB

For the 1<sup>st</sup> Mongo query, the answer is 1,554.

For the 2<sup>nd</sup> Mongo query, the answer is 2,530.

For the 3<sup>rd</sup> Mongo query, the answer is the data in the curly brackets.

```
CSDS — ubuntu@ip-172-31-41-191: ~/csds-material — ssh -i phc2121.pem ubuntu@ec2-18-220-72-241.us-east...

// 1. the number of deleted messages in the dataset
db.tweets.find(
{ "delete": { $exists: true } }
).count()

// 2. the number of tweets that are replies to another tweet
db.tweets.distinct(
"id", { "in_reply_to_status_id": { $ne: null } }
).length

// 3. the five user IDs (field name: uid) that have tweeted the most
db.tweets.aggregate([
{ $match: { "user.id": { $ne: null } } },
{
  $group: {
    _id: "$user.id_str",
    count: { $sum: 1 }
  }
},
{ $sort: { count: -1 } },
{ $limit: 5 }
])

1,1 Top
```

```
CSDS — ubuntu@ip-172-31-41-191: ~/csds-material — ssh -i phc2121.pem ub...
[ubuntu@ip-172-31-41-191:~/csds-material$ mongo lab2 < hw1_mongo.js
MongoDB shell version: 3.2.19
connecting to: lab2
1554
2530
{ "_id" : "1269521828", "count" : 5 }
{ "_id" : "392695315", "count" : 4 }
{ "_id" : "424808364", "count" : 3 }
{ "_id" : "1706901902", "count" : 3 }
{ "_id" : "1369322330", "count" : 2 }
bye
ubuntu@ip-172-31-41-191:~/csds-material$ ]
```

### Reflection Questions

1. The main difference between schema and protocol buffer is the database structure and data types. Protocol buffer has a nesting structure that *message* (Object type) *tweets* contains several messages *Coord*, *Place*, etc. However, *tweets*, *coords*, and *places* are separated tables in schema. Moreover, schema does not have *Insert* and *Delete* tables; and Protocol buffer does not have message *place\_coords*.

The most similar attribute for schema and protocol buffer is that both of them can only store the structured dataset. If the data is unstructured, we should do preprocessing, like data cleaning, before we store the data into database.

2. Fetching data that is in different tables per query. Since protocol buffer is a nesting and object (*message*) based structure, we can just use pointer to efficiently fetch the data. However, for SQL, we need to join different tables in order to fetch the data.
3. Storing the whole documents—such as medical records, bank statement, etc.—to a database. Since MongoDB is able to directly store whole documents, but SQL can only store structured and tabular data, MongoDB will be better than SQL query.
4. Updating or fetching thousands of values per query in structured dataset. Since SQL has great optimizing to structured data, fetching numerous data once will be more efficient than MongoDB.
5. If the original JSON file contains elements with features that are not in the schema or if different elements have extra features, it will be difficult to convert to relational database schema because it will be hard to confirm how many columns we require.
6. Protocol buffer is good at finding whether the tweet is deleted or not, since *message* (Object) has a feature *is\_delete* that can efficiently point to a Boolean value.

SQL is good at sorting the data and picking up specific numbers of results. However, since our JSON contains unstructured data, storing data to database required preprocessing.

MongoDB is able to store same data with less memory, since MongoDB stores features referring to elements. We don't need to spend extra memory to store extra features that elements do not contain.

7. According to my code, since we design protocol buffer manually, the program only needs to go through dataset once in order to answer three small questions. However, for a user, SQL is more friendly since there are fewer lines of code to answer the questions. Last but not least, if we collect a complex or hierarchical dataset such as XML file, MongoDB will be the better tool.
8. Around 24 hours.