



Data Analytics with Twitter

NO SQL FINAL PROJECT

BY MIGUEL MASTACHE 12-15-23

Introduction

This project aims to take tweets made by the national park service, and extract the tweets, and user data, from twitter using their API. Upon extraction, the data is to be saved in a mongo database for querying, and analysis. The analysis culminating with the mapping of data geospatially.

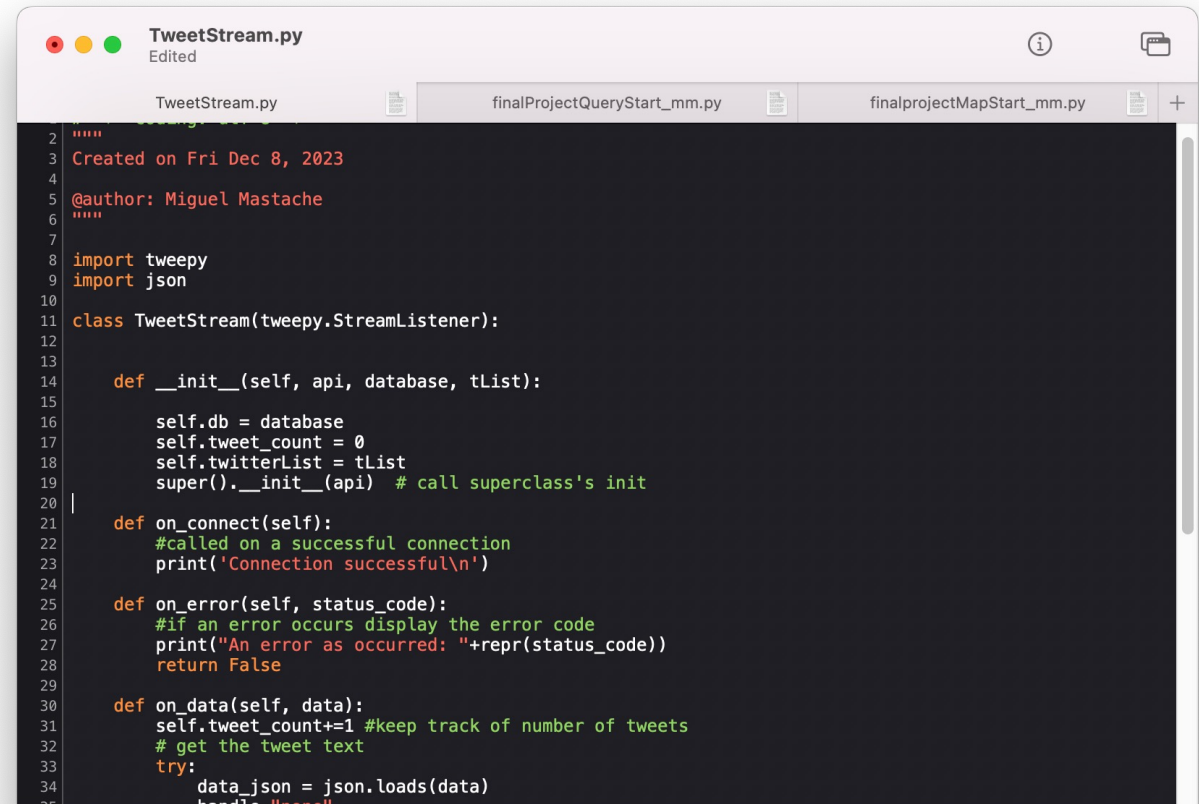
Software

- ▶ Python
- ▶ X –formerly Twitter API
- ▶ JSON
- ▶ Mongo DB
- ▶ Edge
- ▶ Folium



Data

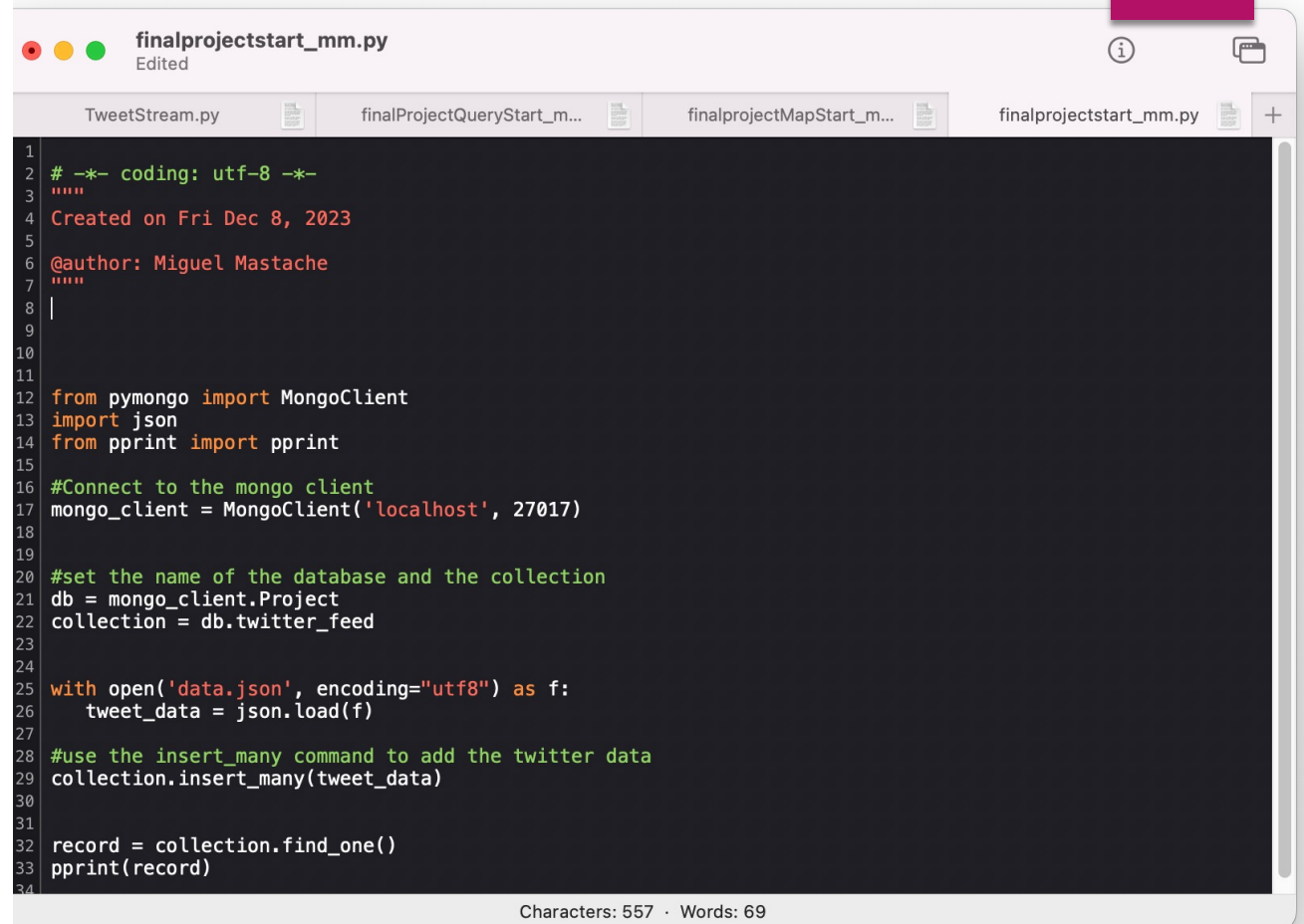
- ▶ We are using using the twitter data for the National Park Service to perform our analysis. We are performing analysis on the national park service popularity associated social media users.



```
2 """
3 Created on Fri Dec 8, 2023
4
5 @author: Miguel Mastache
6 """
7
8 import tweepy
9 import json
10
11 class TweetStream(tweepy.StreamListener):
12
13     def __init__(self, api, database, tList):
14
15         self.db = database
16         self.tweet_count = 0
17         self.twitterList = tList
18         super().__init__(api) # call superclass's init
19
20
21     def on_connect(self):
22         #called on a successful connection
23         print('Connection successful\n')
24
25     def on_error(self, status_code):
26         #if an error occurs display the error code
27         print("An error as occurred: "+repr(status_code))
28         return False
29
30     def on_data(self, data):
31         self.tweet_count+=1 #keep track of number of tweets
32         # get the tweet text
33         try:
34             data_json = json.loads(data)
35             handle=data["user"]
```

Creating the database

Since the X/Twitter data is extracted provided by the Twitter API in JSON format. The Twitter data is best stored using a MongoDB due to its compatibility with JSON.

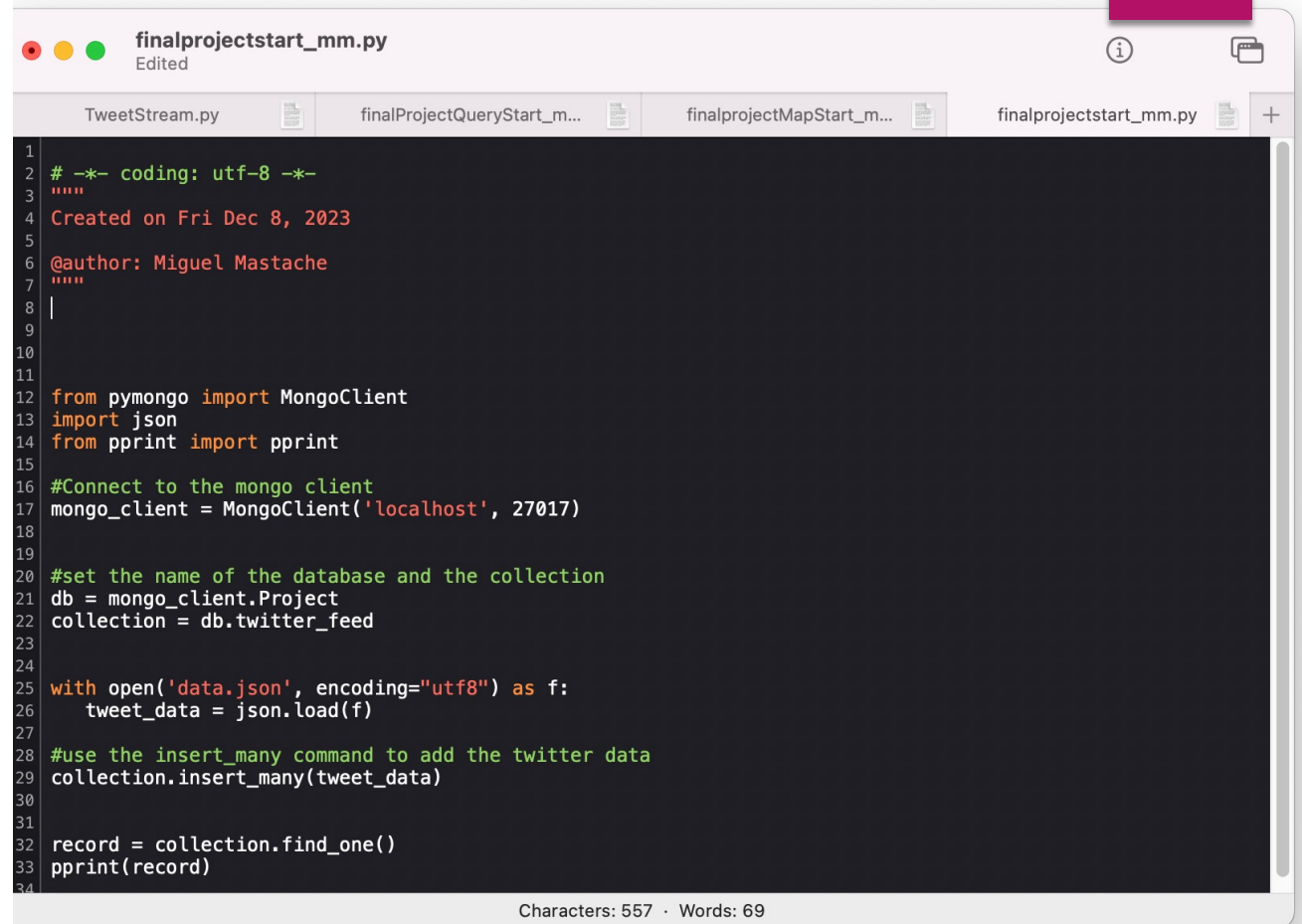


```
1
2 # -*- coding: utf-8 -*-
3
4 Created on Fri Dec 8, 2023
5
6 @author: Miguel Mastache
7
8 |
9
10
11
12 from pymongo import MongoClient
13 import json
14 from pprint import pprint
15
16 #Connect to the mongo client
17 mongo_client = MongoClient('localhost', 27017)
18
19
20 #set the name of the database and the collection
21 db = mongo_client.Project
22 collection = db.twitter_feed
23
24
25 with open('data.json', encoding="utf8") as f:
26     tweet_data = json.load(f)
27
28 #use the insert_many command to add the twitter data
29 collection.insert_many(tweet_data)
30
31
32 record = collection.find_one()
33 pprint(record)
34
```

Characters: 557 · Words: 69

Reports Queries

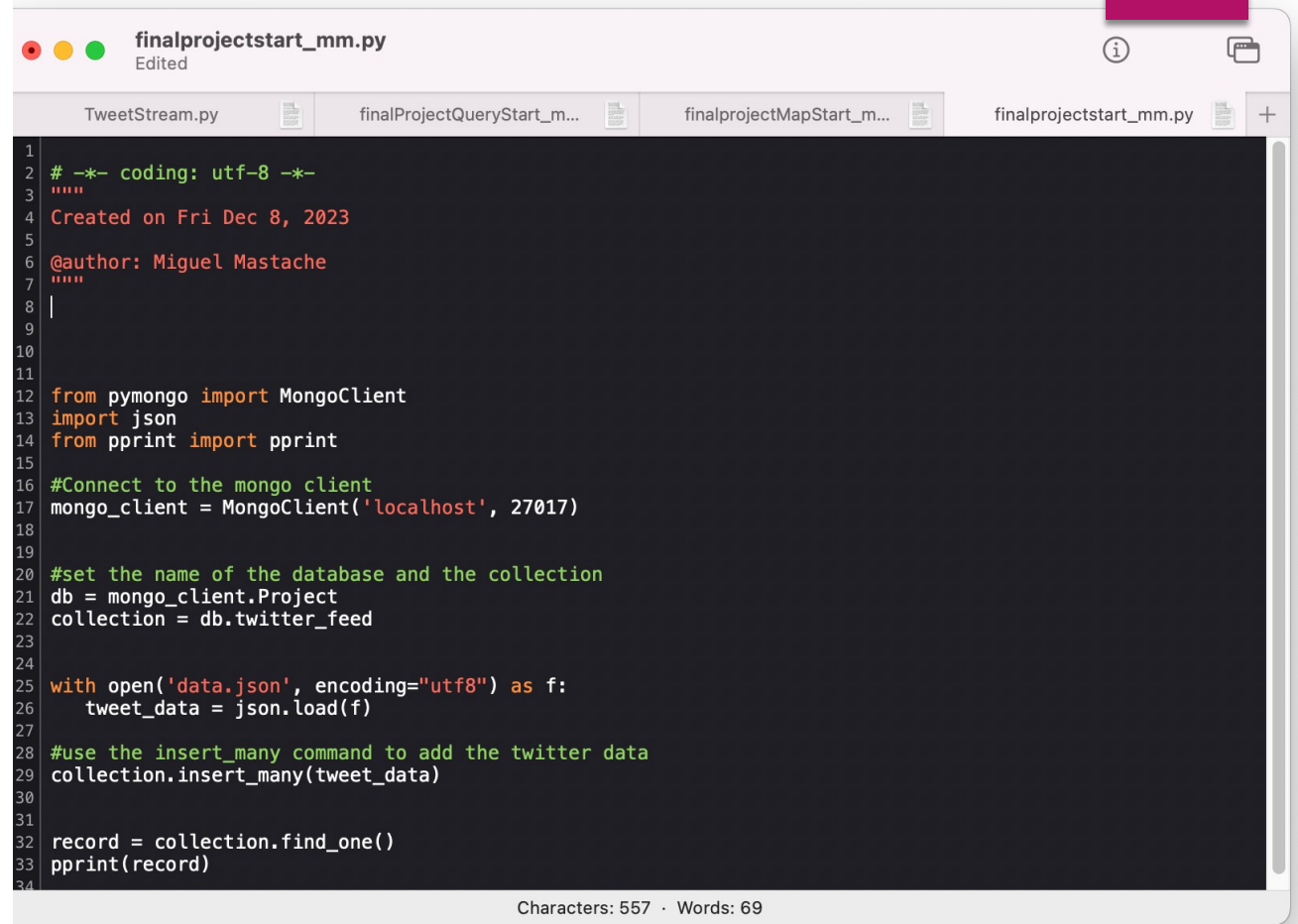
Since the X/Twitter data is extracted provided by the Twitter API in JSON format. The Twitter data is best stored using a MongoDB due to its compatibility with JSON.



```
1
2 # -*- coding: utf-8 -*-
3
4 Created on Fri Dec 8, 2023
5
6 @author: Miguel Mastache
7
8 |
9
10
11
12 from pymongo import MongoClient
13 import json
14 from pprint import pprint
15
16 #Connect to the mongo client
17 mongo_client = MongoClient('localhost', 27017)
18
19
20 #set the name of the database and the collection
21 db = mongo_client.Project
22 collection = db.twitter_feed
23
24
25 with open('data.json', encoding="utf8") as f:
26     tweet_data = json.load(f)
27
28 #use the insert_many command to add the twitter data
29 collection.insert_many(tweet_data)
30
31
32 record = collection.find_one()
33 pprint(record)
34
```

Characters: 557 · Words: 69

Reports Queries



```
1
2 # -*- coding: utf-8 -*-
3
4 Created on Fri Dec 8, 2023
5
6 @author: Miguel Mastache
7
8 |
9
10
11
12 from pymongo import MongoClient
13 import json
14 from pprint import pprint
15
16 #Connect to the mongo client
17 mongo_client = MongoClient('localhost', 27017)
18
19
20 #set the name of the database and the collection
21 db = mongo_client.Project
22 collection = db.twitter_feed
23
24
25 with open('data.json', encoding="utf8") as f:
26     tweet_data = json.load(f)
27
28 #use the insert_many command to add the twitter data
29 collection.insert_many(tweet_data)
30
31
32 record = collection.find_one()
33 pprint(record)
34
```

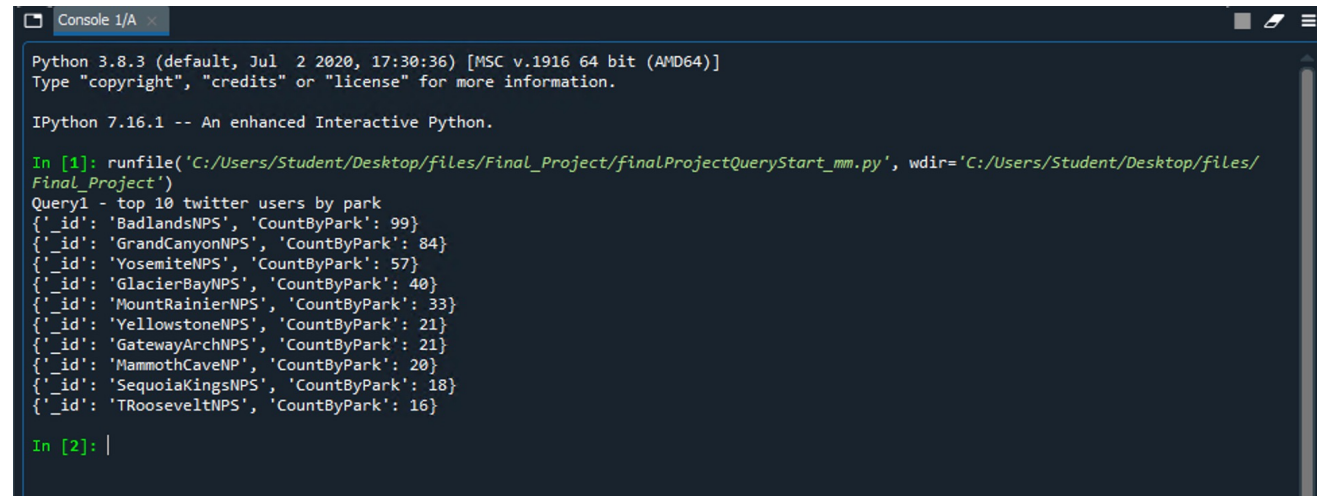
Characters: 557 · Words: 69

Query1 - top 10 twitter users by park

This query is an aggregation of Twitter Handles, and the number of tweets counted for each park.

This then sorts the aggregated data by tweet count and limits it to the top 10 number of tweets by park. It appears that the Badlands has the most twitter users.

```
print("Query1 - top 10 twitter users by park")
record = collection.aggregate(
    [{"$group":{"_id":"$TwitterHandle", "CountByPark": {"$sum":1}}}
    ,{"$sort":{"CountByPark":-1}},
    {"$limit":10}])
```



```
Console 1/A
Python 3.8.3 (default, Jul 2 2020, 17:30:36) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.16.1 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/Student/Desktop/files/Final_Project/finalProjectQueryStart_mm.py', wdir='C:/Users/Student/Desktop/files/Final_Project')
Query1 - top 10 twitter users by park
{'_id': 'BadlandsNPS', 'CountByPark': 99}
{'_id': 'GrandCanyonNPS', 'CountByPark': 84}
{'_id': 'YosemiteNPS', 'CountByPark': 57}
{'_id': 'GlacierBayNPS', 'CountByPark': 40}
{'_id': 'MountRainierNPS', 'CountByPark': 33}
{'_id': 'YellowstoneNPS', 'CountByPark': 21}
{'_id': 'GatewayArchNPS', 'CountByPark': 21}
{'_id': 'MammothCaveNP', 'CountByPark': 20}
{'_id': 'SequoiaKingsNPS', 'CountByPark': 18}
{'_id': 'TRooseveltNPS', 'CountByPark': 16}

In [2]: |
```


Query 2 - parks with a high follower count

This query finds twitter handles with the most number of followers. It filters it by parks with greater than 100,000 followers.

In this case it appears that ZionNPS, and GrandCanyonNPS both have over 100,000 followers.

```
print("\n\nQuery 2 - parks with a high follower count")

record = collection.find(

    {"user.followers_count": {"$gte":100000}},

    {"user.screen_name":1, "TwitterHandle":1, "user.followers_count":1})
```



```
Console 1/A
Query 2 - parks with a high follower count
{'_id': {'oid': '5faaa096e7b7746a92a063c3'}, 'user': {'screen_name': 'ZionNPS', 'followers_count': 133093}, 'TwitterHandle': 'ZionNPS'}
{'_id': {'oid': '5faaa97ae7b7746a92a063ff'}, 'user': {'screen_name': 'MuseumModernArt', 'followers_count': 5361701}, 'TwitterHandle': 'YosemiteNPS'}
{'_id': {'oid': '5faab9fee7b7746a92a06485'}, 'user': {'screen_name': 'GoogleExpertUK', 'followers_count': 248995}, 'TwitterHandle': 'YosemiteNPS'}
{'_id': {'oid': '5faabb04e7b7746a92a06489'}, 'user': {'screen_name': 'GoogleExpertUK', 'followers_count': 248994}, 'TwitterHandle': 'YosemiteNPS'}
{'_id': {'oid': '5faabbdde7b7746a92a0648f'}, 'user': {'screen_name': 'GoogleExpertUK', 'followers_count': 248993}, 'TwitterHandle': 'YosemiteNPS'}
{'_id': {'oid': '5faabbf0e7b7746a92a06490'}, 'user': {'screen_name': 'GoogleExpertUK', 'followers_count': 248993}, 'TwitterHandle': 'YosemiteNPS'}
{'_id': {'oid': '5faabc45e7b7746a92a06495'}, 'user': {'screen_name': 'GoogleExpertUK', 'followers_count': 248993}, 'TwitterHandle': 'YosemiteNPS'}
{'_id': {'oid': '5faabc59e7b7746a92a06496'}, 'user': {'screen_name': 'GoogleExpertUK', 'followers_count': 248995}, 'TwitterHandle': 'YosemiteNPS'}
{'_id': {'oid': '5faac10fe7b7746a92a064c0'}, 'user': {'screen_name': 'Interior', 'followers_count': 4980871}, 'TwitterHandle': 'BadlandsNPS'}
{'_id': {'oid': '5faac5ace7b7746a92a06515'}, 'user': {'screen_name': 'GrandCanyonNPS', 'followers_count': 153485}, 'TwitterHandle': 'GrandCanyonNPS'}
```

Query 3 - number of retweets

This query counts the number of retweets for the national parks, and indicates that there are none. This is to be expected as the national park service is providing information, not memorable tweets to be shared.

```
Console 1/A x
Query 3 - number of retweets
0

Query 4 - search through text for a particular word
{'_id': {'oid': '5faa9fc6e7b7746a92a063bf'}, 'text': "RT @GreatSmokyNPS: They say a picture is worth a thousand words... What's one word this picture brings to mind for you?\n\nNPS Image: Look Ro...", 'TwitterHandle': 'GreatSmokyNPS'}
{'_id': {'oid': '5faaab57e7b7746a92a0640f'}, 'text': "RT @GreatSmokyNPS: They say a picture is worth a thousand words... What's one word this picture brings to mind for you?\n\nNPS Image: Look Ro...", 'TwitterHandle': 'GreatSmokyNPS'}
{'_id': {'oid': '5faaab68e7b7746a92a06410'}, 'text': "RT @GreatSmokyNPS: They say a picture is worth a thousand words... What's one word this picture brings to mind for you?\n\nNPS Image: Look Ro...", 'TwitterHandle': 'GreatSmokyNPS'}
{'_id': {'oid': '5faac3eee7b7746a92a06502'}, 'text': "RT @GreatSmokyNPS: They say a picture is worth a thousand words... What's one word this picture brings to mind for you?\n\nNPS Image: Look Ro...", 'TwitterHandle': 'GreatSmokyNPS'}

In [5]:
```

```
print("\n\nQuery 3 - number of retweets")

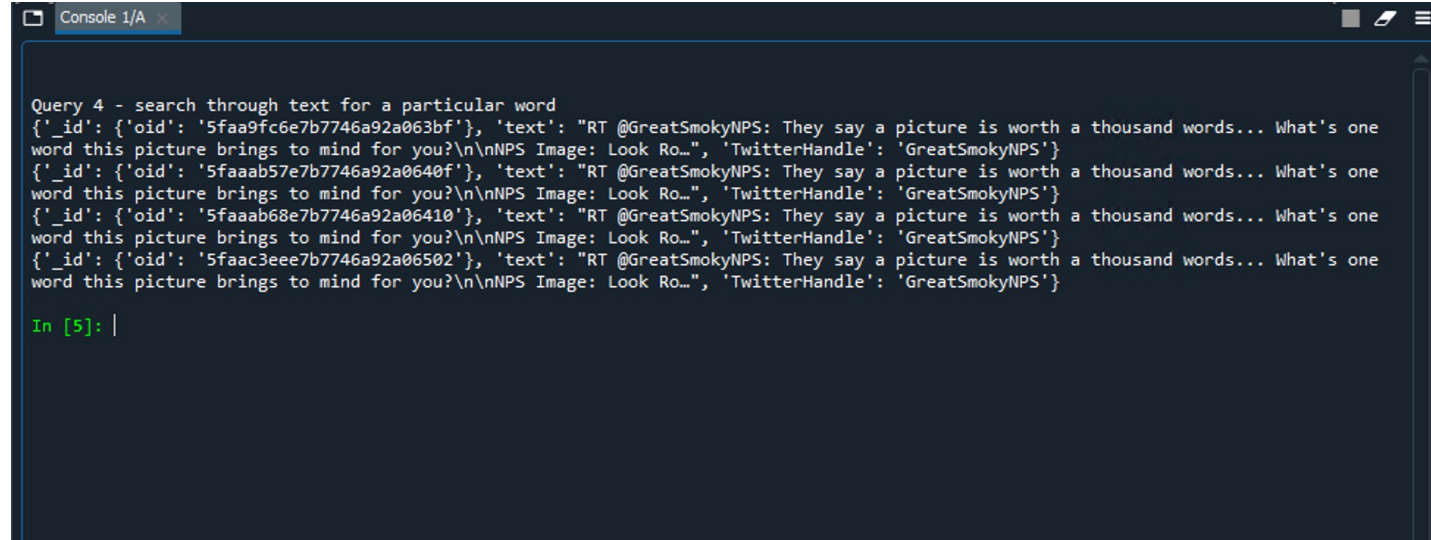
record = collection.count_documents(
    {"retweet_count":{"$gte":1}})

print(record)
```

Query 4 - search through text for a particular word")

This query searches all tweets for the word "picture", and it appears that the Great Smoky park, has tweets containing that word. Indicating that a picture is worth 1000 words.

```
print("\n\nQuery 4 - search through text for a particular word")  
  
record = collection.find(  
    {"text":{"$regex":"picture"}}, {"TwitterHandle":1, "text":1})
```



```
Console 1/A x  
  
Query 4 - search through text for a particular word  
{'_id': {'oid': '5faa9fc6e7b7746a92a063bf'}, 'text': "RT @GreatSmokyNPS: They say a picture is worth a thousand words... What's one word this picture brings to mind for you?\n\nNPS Image: Look Ro...", 'TwitterHandle': 'GreatSmokyNPS'}  
{'_id': {'oid': '5faaab57e7b7746a92a0640f'}, 'text': "RT @GreatSmokyNPS: They say a picture is worth a thousand words... What's one word this picture brings to mind for you?\n\nNPS Image: Look Ro...", 'TwitterHandle': 'GreatSmokyNPS'}  
{'_id': {'oid': '5faaab68e7b7746a92a06410'}, 'text': "RT @GreatSmokyNPS: They say a picture is worth a thousand words... What's one word this picture brings to mind for you?\n\nNPS Image: Look Ro...", 'TwitterHandle': 'GreatSmokyNPS'}  
{'_id': {'oid': '5faac3eee7b7746a92a06502'}, 'text': "RT @GreatSmokyNPS: They say a picture is worth a thousand words... What's one word this picture brings to mind for you?\n\nNPS Image: Look Ro...", 'TwitterHandle': 'GreatSmokyNPS'}  
  
In [5]: |
```

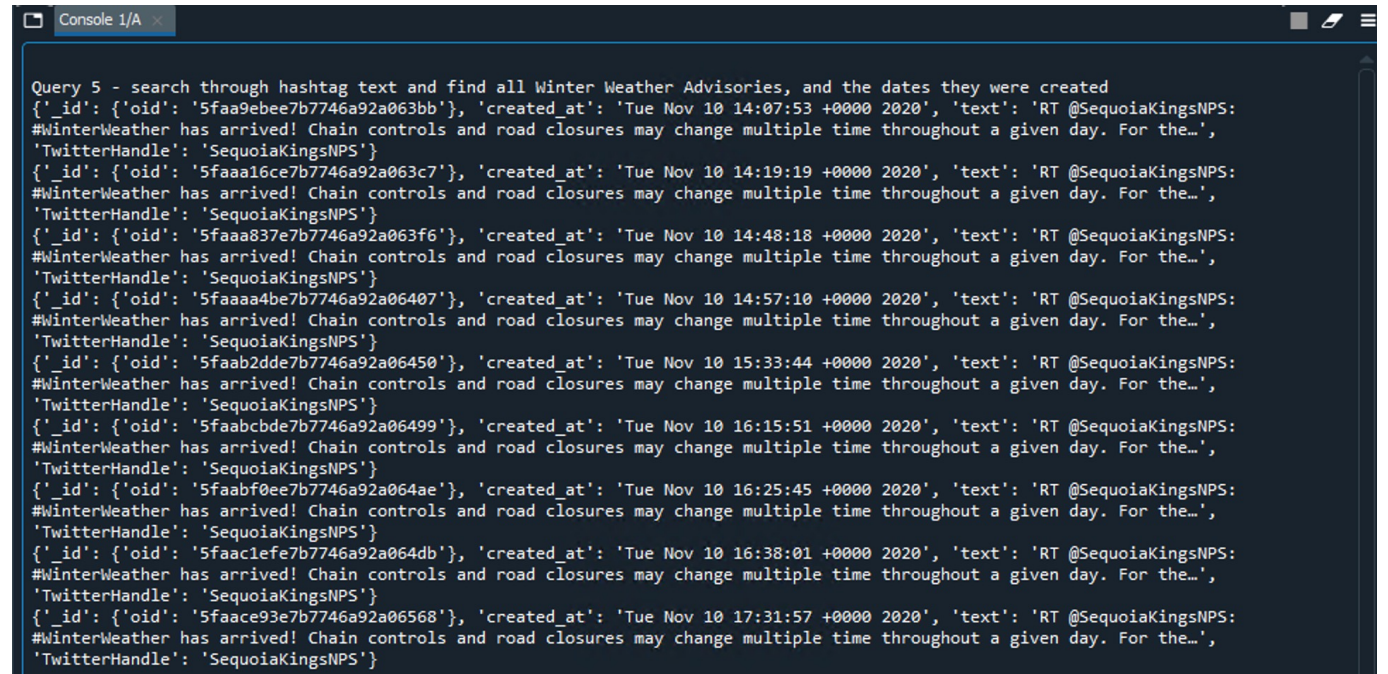
Query5 - search Winter Weather Advisories, and dates

This query searches the hashtags for each user, for the phrase "WinterWeather" and provides the twitter handle, and body of the tweet, along with the date of the tweet.

We can see that only 1 park issued winter weather warnings in the time period. Sequoia King.

#search through hashtag text and find all Winter Weather Advisories, and the dates they were created

```
record = collection.find(  
  
  {"entities.hashtags.text":{"$regex":"WinterWeather"}},  
  
  {"TwitterHandle":1, "created_at":1, "text":1})
```

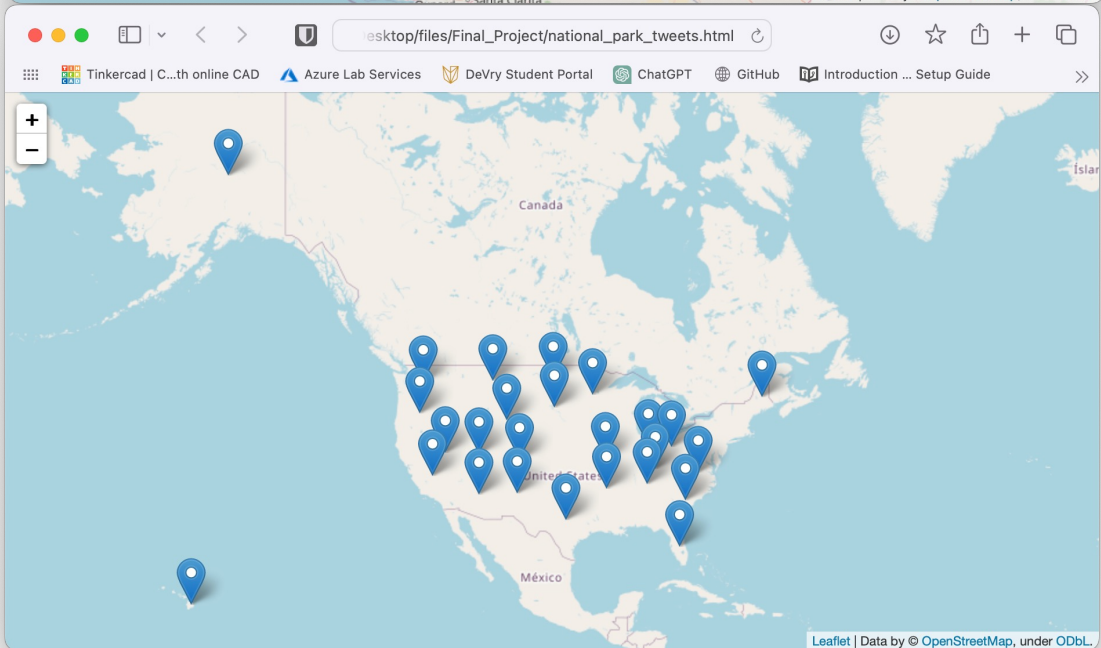
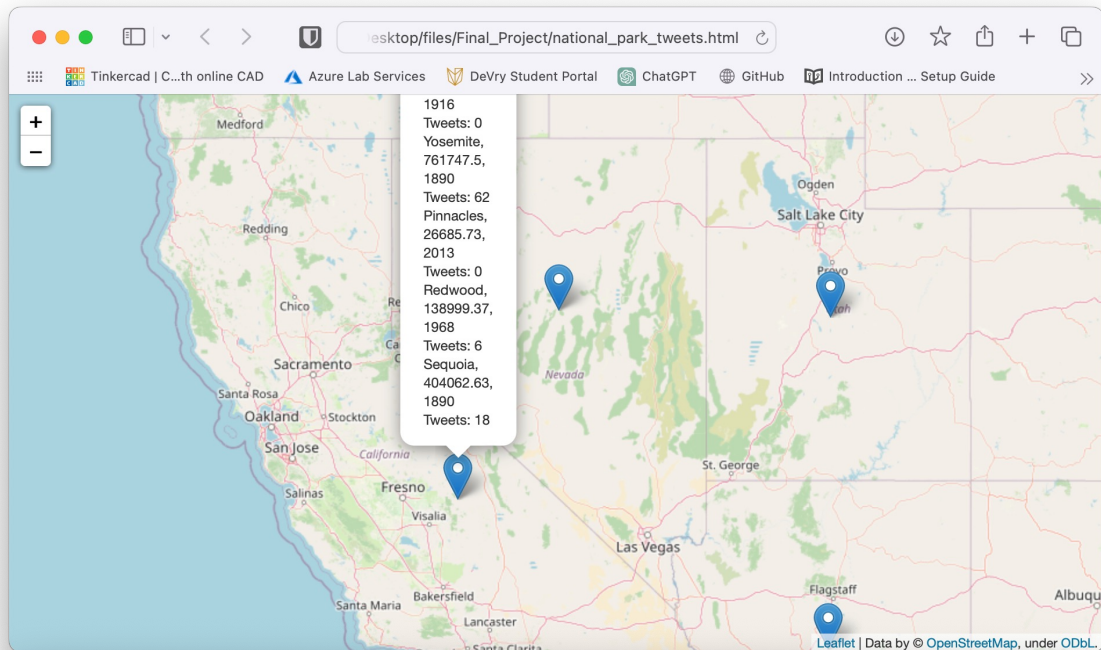


```
Console 1/A  
  
Query 5 - search through hashtag text and find all Winter Weather Advisories, and the dates they were created  
{ '_id': { 'oid': '5faa9ebee7b7746a92a063bb' }, 'created_at': 'Tue Nov 10 14:07:53 +0000 2020', 'text': 'RT @SequoiaKingsNPS:  
#WinterWeather has arrived! Chain controls and road closures may change multiple time throughout a given day. For the...',  
'TwitterHandle': 'SequoiaKingsNPS' }  
{ '_id': { 'oid': '5faaa16ce7b7746a92a063c7' }, 'created_at': 'Tue Nov 10 14:19:19 +0000 2020', 'text': 'RT @SequoiaKingsNPS:  
#WinterWeather has arrived! Chain controls and road closures may change multiple time throughout a given day. For the...',  
'TwitterHandle': 'SequoiaKingsNPS' }  
{ '_id': { 'oid': '5faaa837e7b7746a92a063f6' }, 'created_at': 'Tue Nov 10 14:48:18 +0000 2020', 'text': 'RT @SequoiaKingsNPS:  
#WinterWeather has arrived! Chain controls and road closures may change multiple time throughout a given day. For the...',  
'TwitterHandle': 'SequoiaKingsNPS' }  
{ '_id': { 'oid': '5faaaa4be7b7746a92a06407' }, 'created_at': 'Tue Nov 10 14:57:10 +0000 2020', 'text': 'RT @SequoiaKingsNPS:  
#WinterWeather has arrived! Chain controls and road closures may change multiple time throughout a given day. For the...',  
'TwitterHandle': 'SequoiaKingsNPS' }  
{ '_id': { 'oid': '5faab2dde7b7746a92a06450' }, 'created_at': 'Tue Nov 10 15:33:44 +0000 2020', 'text': 'RT @SequoiaKingsNPS:  
#WinterWeather has arrived! Chain controls and road closures may change multiple time throughout a given day. For the...',  
'TwitterHandle': 'SequoiaKingsNPS' }  
{ '_id': { 'oid': '5faabcbde7b7746a92a06499' }, 'created_at': 'Tue Nov 10 16:15:51 +0000 2020', 'text': 'RT @SequoiaKingsNPS:  
#WinterWeather has arrived! Chain controls and road closures may change multiple time throughout a given day. For the...',  
'TwitterHandle': 'SequoiaKingsNPS' }  
{ '_id': { 'oid': '5faabf0ee7b7746a92a064ae' }, 'created_at': 'Tue Nov 10 16:25:45 +0000 2020', 'text': 'RT @SequoiaKingsNPS:  
#WinterWeather has arrived! Chain controls and road closures may change multiple time throughout a given day. For the...',  
'TwitterHandle': 'SequoiaKingsNPS' }  
{ '_id': { 'oid': '5faac1efe7b7746a92a064db' }, 'created_at': 'Tue Nov 10 16:38:01 +0000 2020', 'text': 'RT @SequoiaKingsNPS:  
#WinterWeather has arrived! Chain controls and road closures may change multiple time throughout a given day. For the...',  
'TwitterHandle': 'SequoiaKingsNPS' }  
{ '_id': { 'oid': '5faace93e7b7746a92a06568' }, 'created_at': 'Tue Nov 10 17:31:57 +0000 2020', 'text': 'RT @SequoiaKingsNPS:  
#WinterWeather has arrived! Chain controls and road closures may change multiple time throughout a given day. For the...',  
'TwitterHandle': 'SequoiaKingsNPS' }
```

```
finalprojectMapStart_mm.py
TweetStream.py finalprojectstart_mm.py finalProjectQueryStart... finalprojectMapStart...
1
2
3 # -*- coding: utf-8 -*-
4 """
5 Created on Sat Nov 7 21:34:00 2020
6
7 @author: Student
8 """
9
10
11 from geopy.geocoders import Nominatim
12
13 from finalproject_statecodes import state_codes
14 import folium
15 from pymongo import MongoClient
16 import pandas as pd
17
18
19 geolocator = Nominatim(user_agent= "myapp")
20 #####
21 #write code in this comment box
22 #create data frame and read from file
23 df=pd.read_csv("nationalparks.csv")
24
25 #create the mongo client
26 mongo_client = MongoClient('localhost', 27017)
27
28 #set the database to Project
29 db = mongo_client.Project
30
31 #create a list named tweets
32 tweets=[]
33
34 #create a list named park_locations
35 park_locations=[]
36
37
38
39 #####
40
41 #use the count_documents method to count the total number of documents in the collection that contains
42 that
43 #text for each park
44 for np in df.TwitterHandle:
45     tweets.append(db.twitter_feed.count_documents({"$text": {"$search": np}}))
46 park_df = df.assign(Tweets = tweets)
47
48 #get the states for each park
49 states = park_df.State.unique()
50 states.sort()
51
52 #Must append a command USA after each state in the list so that the map has accurate location
53 information
54 for s in states:
55     park_locations.append(geolocator.geocode(state_codes[s] + ', USA'))
56
57 #create the map - feel free to adjust location and zoom
58 usmap = folium.Map(location=[39.98334, -82.9833], zoom_start =4)
59
60 #This for loop walks through each item in the collection and returns the data for each park
61 #It then creates a marker with the latitude and longitude of the location.
62 #Then add the marker to the map
63 for index, (name, group) in enumerate(park_df.groupby('State')):
64     text = [state_codes[name]]
65
66     for s in group.itertuples():
67         text.append(f'{s.Name}, {s.Area}, {s.YearEstablished} Tweets: {s.Tweets}')
68     display = '<br>'.join(text)
69     marker = folium.Marker((park_locations[index].latitude, park_locations[index].longitude), popup =
70     display)
71     marker.add_to(usmap)
72
73 #save to a file named national_park_tweets.html
74 usmap.save("national_park_tweets.html")
75 print("Web page created")
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

Characters: 2,108 · Words: 254

Visualization



Visualization

The visualization code uses Folium to create a map visualization, with the location of each national park and the number of tweets.

Skills Obtained & Challenges

During this project the following were learned. Learning to create queries, in nosql, as well as how to populate data and perform queries using Python.

- ▶ MongoDB
- ▶ JSON
- ▶ Docker

- ▶ Some of the challenges were learning to use docker on a Macintosh, and learning to set it up for remote access. Additionally finding the documentation to be able to do so. Additionally knowing the schema of the data proved a little difficult, Requiring much reading.

Conclusion

Mongo and performing the queries in Mongo, is very fast. Compared to SQL queries, it is faster to not have to perform the joins, however, finding the data is a little more difficult at times, because you have to know the layout of the document. However, more streamlined than learning all the tables and needing to perform joins to retrieve the data.