# 1 Weeks 3/4

This week was an exploration about querying and withdrawing data from the databases. Querying is a big part of any modeling going forward, and it was important to understand the basics of querying in Mongo. Here is a review of my work this past week:

## 1.1 Querying

A query is used to retrieve data from a database. Simple queries (with our database in mind), may just be all the tweets in English. More complex queries may be searching with multiple factors such as trend, location, and time. Most queries will return a set of data, but incredibly complex queries may return a single document.

A special type of query, the empty query, will return every single document in the collection (or table) being queried. Empty queries are generally discouraged, since parsing through the query result is essentially parsing through the entire database.

## 1.2 Querying in MongoDB

Referring back to the structure of a MongoDB outlined, querying happens in individual collection. The find command works on a single collection and finds documents based on the given parameters. Here's how the search command is laid out:

```
db.collection.find({field: match})
```

This command will return a cursor that references all the tweets that match the condition. The cursor is iterable which means that it can be iterated through like a list or a dictionary. It also has a variety of features such as count - which returns the number of tweets the cursor refers to.

Here's a breakdown of what's happening in the attached code snippet. Tweets in the senate database are stored, unsorted. It's not clear which race each tweet belongs to. In a separate document, the search terms for each race are organized. The code race sort does three things:

1º Reads the file containing the search terms for each race. This is usually the name of the candidate and the name of the race. For example two terms for the Arizona Senate Race to replace Jeff Flake's seat are "Krysten Sinema" and "AZSenate".

2º Uses regular expression to search for terms within the text of each tweet. Currently, each tweet is a JSON object that holds a lot of data beyond the text. This means that the text is a field by itself, and to search for particular races we have to search within the field. We use regular expressions - which is a method of pattern searching to accomplish this task.

3º As mentioned above, the completed search will return a cursor. This code calls the function count, which returns the number of tweets collected in the search. Future code will look to label the tweets with an appropriate race.

### 1.3 Problems I'm Struggling With

There's definitely some overlap between search terms of various races. To properly count a total requires the use of combinatoric principles, and I'm still unsure how to do that. Another issue that is still out there is that iterating through a cursor that refers to a large number of tweets times out. That needs to be fixed.

### 1.4 Code Written This Week

```python
import datetime
import ast
from configparser import ConfigParser
from pymongo import MongoClient

FILE_NAME = 'races.ini'
MONGO_HOST = 'mongodb://localhost/housedb'

def section_to_dict(section, parser):
    #change path to your ini file if running locally
    parser.read(FILE_NAME)
    out_dict = {}
    for key in parser[section]:
        temp = ast.literal_eval(parser[section][key])
        out_dict[temp[-1]] = temp[:-1]
    return(out_dict)

def gather_tweets(race, collection):
    total = 0
    for i in race:
        count = collection.find({"text": {"$regex" : i , "
    $options": "$i"}}).count()
        print(count)
        total += count
    return(total)

if __name__ == "__main__":
    config = ConfigParser()
    client = MongoClient(MONGO_HOST)
    db = client.housedb
    presort = db.presort
    senate = section_to_dict('HOUSE', config)
    f = open('house_count.csv', 'w+')
    f.write('Race,Tweets\n')
    for i in senate:
        print(i)
        count = gather_tweets(i, presort)
        print(count)
        f.write(i + ',' + str(count) + '\n')
    f.close()
```