Data Wrangling with MongoDB

Project-2: Data Wrangle OpenStreetMaps Data

Name: Vaishnav Mohanan Mavilakandy

Map Area: Charlotte, NC, United States

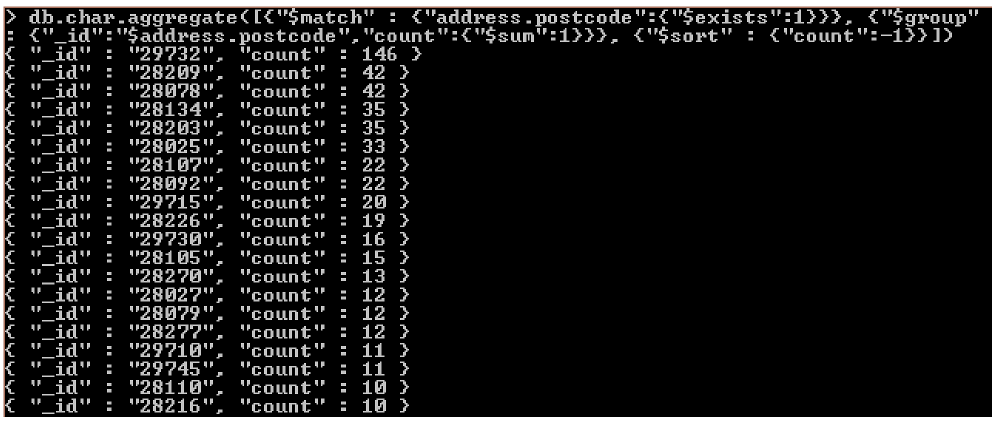
https://s3.amazonaws.com/metro-extracts.mapzen.com/charlotte_north-carolina.osm.bz2

I preferred taking my hometown (Kannur, Kerala, India) to wrangle with. But it seemed too small (less than 50 MB) to analyse. So my brother who is settled in Charlotte, North Carolina, suggested me to take this. I would most probably be doing my Masters there soon. So he said it's good to be familiarised with this place.

I downloaded my data from OpenStreetMaps's previously prepared metro extracts. This led me to www.mapzen.com/metro-extracts. I unzipped the compressed version and started passing it to my codes of mapparser.py, tags.py, users.py, audit.py, data.py.

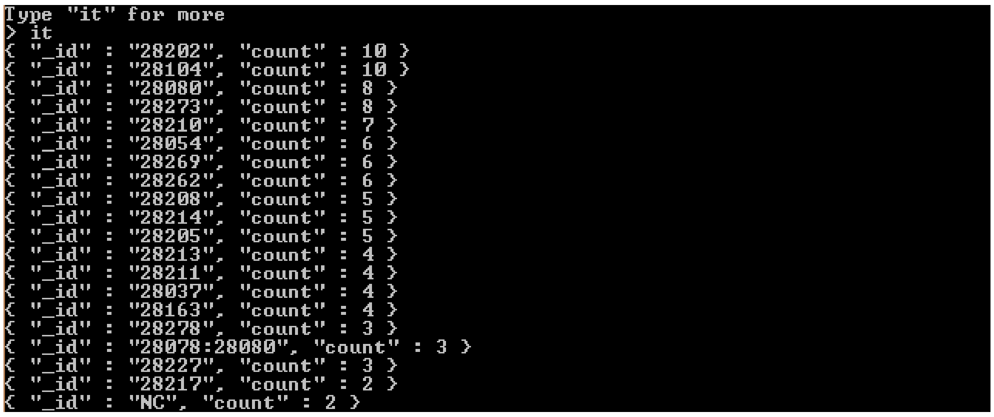
1. Problems Encountered in the Map

- I ran it against my audit.py code and corrected the over-abbreviated street names.
  For example there where street names like,
  (a) Virginia Pine Ln
  (b) West Stanly St.
  (c) 10352 Park Rd
  Using the update function of audit.py, I modified these street names by iterating over each word of these street names. After the modifications all the street names were rid of these abbreviations. For example, these turned to
  (a) Virginia Pine Lane
  (b) West Stanly Street
  (c) 10352 Park Road
- I went on to find any discrepancies in the JSON file uploaded into MongoDB. So I performed a queries on the postal code. The output was:

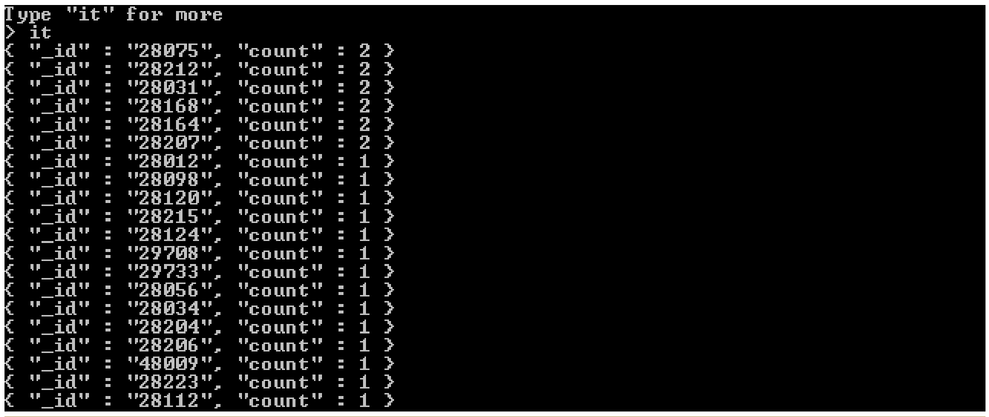


```
> db.char.aggregate([{"$match" : {"address.postcode":{"$exists":1}}}, {"$group"
: {"_id":"$address.postcode","count":{"$sum":1}}}, {"$sort" : {"count":-1}}])
{ "_id" : "29732", "count" : 146 }
{ "_id" : "28209", "count" : 42 }
{ "_id" : "28078", "count" : 42 }
{ "_id" : "28134", "count" : 35 }
{ "_id" : "28203", "count" : 35 }
{ "_id" : "28025", "count" : 33 }
{ "_id" : "28107", "count" : 22 }
{ "_id" : "28092", "count" : 22 }
{ "_id" : "29715", "count" : 20 }
{ "_id" : "28226", "count" : 19 }
{ "_id" : "29730", "count" : 16 }
{ "_id" : "28105", "count" : 15 }
{ "_id" : "28270", "count" : 13 }
{ "_id" : "28027", "count" : 12 }
{ "_id" : "28079", "count" : 12 }
{ "_id" : "28277", "count" : 12 }
{ "_id" : "29710", "count" : 11 }
{ "_id" : "29745", "count" : 11 }
{ "_id" : "28110", "count" : 10 }
{ "_id" : "28216", "count" : 10 }
```

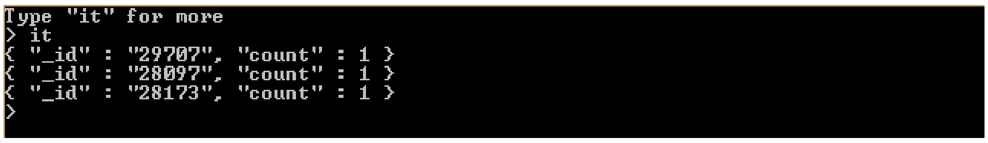
On entering the "it" command,



```
Type "it" for more
> it
{ "_id" : "28202", "count" : 10 }
{ "_id" : "28104", "count" : 10 }
{ "_id" : "28080", "count" : 8 }
{ "_id" : "28273", "count" : 8 }
{ "_id" : "28210", "count" : 7 }
{ "_id" : "28054", "count" : 6 }
{ "_id" : "28269", "count" : 6 }
{ "_id" : "28262", "count" : 6 }
{ "_id" : "28208", "count" : 5 }
{ "_id" : "28214", "count" : 5 }
{ "_id" : "28205", "count" : 5 }
{ "_id" : "28213", "count" : 4 }
{ "_id" : "28211", "count" : 4 }
{ "_id" : "28037", "count" : 4 }
{ "_id" : "28163", "count" : 4 }
{ "_id" : "28278", "count" : 3 }
{ "_id" : "28078:28080", "count" : 3 }
{ "_id" : "28227", "count" : 3 }
{ "_id" : "28217", "count" : 2 }
{ "_id" : "NC", "count" : 2 }
```



```
Type "it" for more
> it
{ "_id" : "28075", "count" : 2 }
{ "_id" : "28212", "count" : 2 }
{ "_id" : "28031", "count" : 2 }
{ "_id" : "28168", "count" : 2 }
{ "_id" : "28164", "count" : 2 }
{ "_id" : "28207", "count" : 2 }
{ "_id" : "28012", "count" : 1 }
{ "_id" : "28098", "count" : 1 }
{ "_id" : "28120", "count" : 1 }
{ "_id" : "28215", "count" : 1 }
{ "_id" : "28124", "count" : 1 }
{ "_id" : "29708", "count" : 1 }
{ "_id" : "29733", "count" : 1 }
{ "_id" : "28056", "count" : 1 }
{ "_id" : "28034", "count" : 1 }
{ "_id" : "28204", "count" : 1 }
{ "_id" : "28206", "count" : 1 }
{ "_id" : "48009", "count" : 1 }
{ "_id" : "28223", "count" : 1 }
{ "_id" : "28112", "count" : 1 }
```

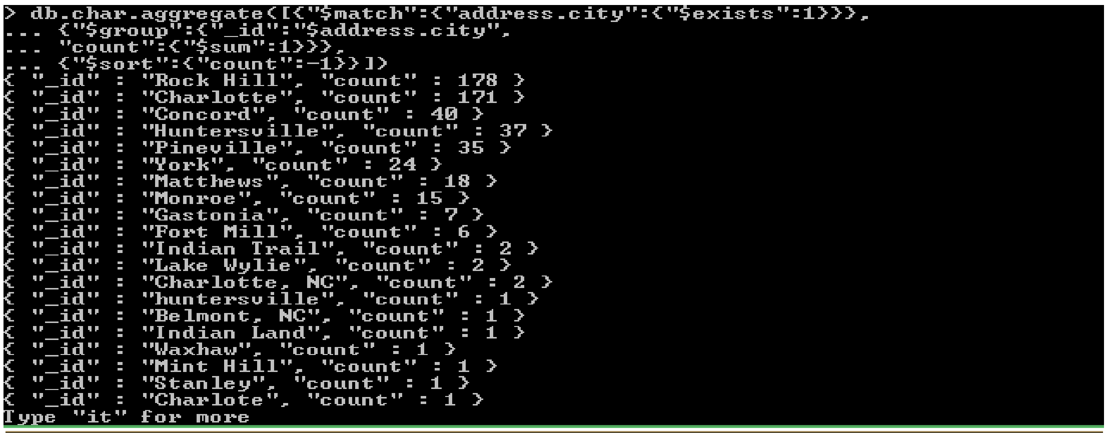


```
Type "it" for more
> it
{ "_id" : "29707", "count" : 1 }
{ "_id" : "28097", "count" : 1 }
{ "_id" : "28173", "count" : 1 }
>
```

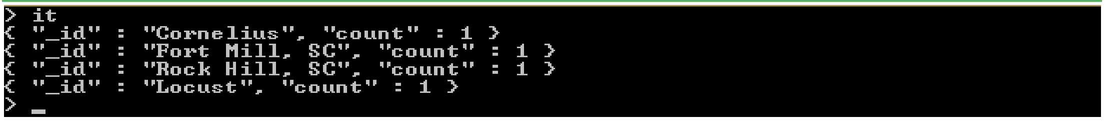
I found a problem in the result of the above query. The range of postal codes seemed to vary a lot according to me, which I didn't find it to be correct, considering a particular region. So, I googled the postal code for Charlotte, NC. I found the link below very useful.

http://www.city-data.com/zipmaps/Charlotte-North-Carolina.html

From this link, I understood that all the Charlotte, NC postal codes begin with "282" and the postal codes adjacent to the city boundary are 28078, 28105, and 28134. There are several codes that doesn't belong to the Charlotte city (postal code: 29732, with a count of 146, is for Rock Hill, South Carolina) and includes invalid postal codes entries like NC, 28078:28080. I was curious to know to which all are the other cities that have been listed in this dataset. Finding it out manually, for each postal code, would have been a tedious process. So, I decided to run a query.



```
> db.char.aggregate([{"$match":{"address.city":{"$exists":1}}},
... {"$group":{"_id":"$address.city",
... "count":{"$sum":1}}},
... {"$sort":{"count":-1}}])
{ "_id" : "Rock Hill", "count" : 178 }
{ "_id" : "Charlotte", "count" : 171 }
{ "_id" : "Concord", "count" : 40 }
{ "_id" : "Huntersville", "count" : 37 }
{ "_id" : "Pineville", "count" : 35 }
{ "_id" : "York", "count" : 24 }
{ "_id" : "Matthews", "count" : 18 }
{ "_id" : "Monroe", "count" : 15 }
{ "_id" : "Gastonia", "count" : 7 }
{ "_id" : "Fort Mill", "count" : 6 }
{ "_id" : "Indian Trail", "count" : 2 }
{ "_id" : "Lake Wylie", "count" : 2 }
{ "_id" : "Charlotte, NC", "count" : 2 }
{ "_id" : "huntersville", "count" : 1 }
{ "_id" : "Belmont, NC", "count" : 1 }
{ "_id" : "Indian Land", "count" : 1 }
{ "_id" : "Waxhaw", "count" : 1 }
{ "_id" : "Mint Hill", "count" : 1 }
{ "_id" : "Stanley", "count" : 1 }
{ "_id" : "Charlote", "count" : 1 }
Type "it" for more
```



```
> it
{ "_id" : "Cornelius", "count" : 1 }
{ "_id" : "Fort Mill, SC", "count" : 1 }
{ "_id" : "Rock Hill, SC", "count" : 1 }
{ "_id" : "Locust", "count" : 1 }
> _
```

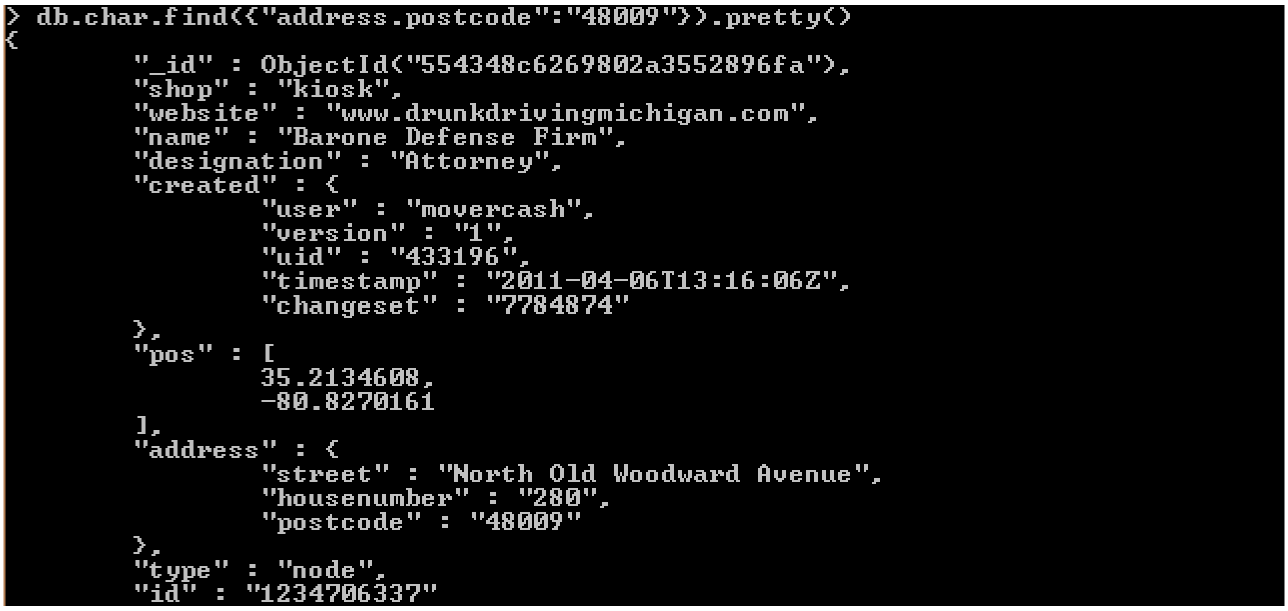
That query gave me the result above.

Since majority of the postal codes were close to 282xx, I got this intuition that the remaining postal codes could be the surrounding cities of this metro. This led me to search for Charlotte Metropolitan Area. I got to study about that from the link below.

http://en.wikipedia.org/wiki/Charlotte_metropolitan_area

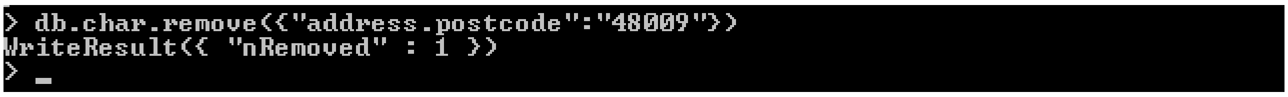
From the info box in the above link, the postal codes of this area are 280xx, 281xx, 282xx, 286xx, 297xx. So, we can come to a conclusion that this dataset is for Charlotte Metropolitan Area.

There is however one exception in the case of the postal code: 48009. On googling for the corresponding city, I found that this postal code is for Birmingham, Michigan. To support this I performed a query.



```
> db.char.find({"address.postcode":"48009"}).pretty()
{
        "_id" : ObjectId("554348c6269802a3552896fa"),
        "shop" : "kiosk",
        "website" : "www.drunkdrivingmichigan.com",
        "name" : "Barone Defense Firm",
        "designation" : "Attorney",
        "created" : {
                "user" : "movercash",
                "version" : "1",
                "uid" : "433196",
                "timestamp" : "2011-04-06T13:16:06Z",
                "changeset" : "7784874"
        },
        "pos" : [
                35.2134608,
                -80.8270161
        ],
        "address" : {
                "street" : "North Old Woodward Avenue",
                "housenumber" : "280",
                "postcode" : "48009"
        },
        "type" : "node",
        "id" : "1234706337"
```

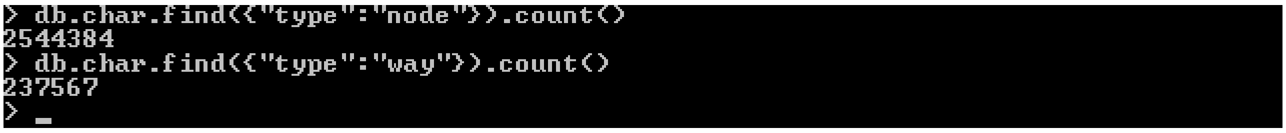
North Old Woodward Avenue is indeed a street in Birmingham but the latitude and longitude falls in Charlotte, NC. This entry is definitely an error, so, I decided to remove it.



```
> db.char.remove({"address.postcode":"48009"})
WriteResult({ "nRemoved" : 1 })
> _
```

- After running mapparser.py, my result is as follows:
  defaultdict(<type 'int'>, {'node': 2544385, 'nd': 2824335, 'bounds': 1, 'member': 5278, 'tag': 843095, 'relation': 516, 'way': 237567, 'osm': 1})

  I decided to run queries to check if these values were correct.



```
> db.char.find({"type":"node"}).count()
2544384
> db.char.find({"type":"way"}).count()
237567
> _
```

It suddenly realised that this difference was due to the fact that I had removed the Birmingham, Michigan document. I had performed mapparser.py before doing so.

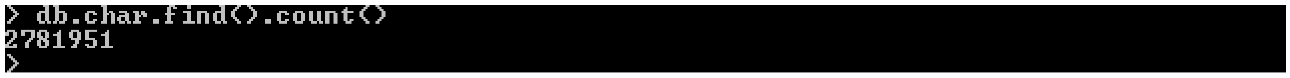
Overview of the Data

File size

charlotte.osm → 511 MB
charlotte.osm.json → 576 MB

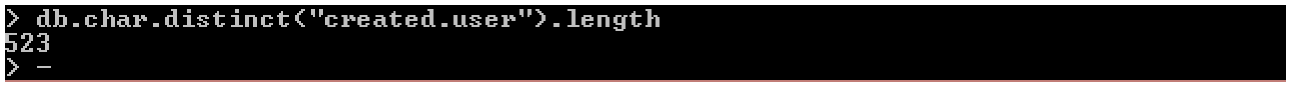
MongoDB queries

Total number of documents



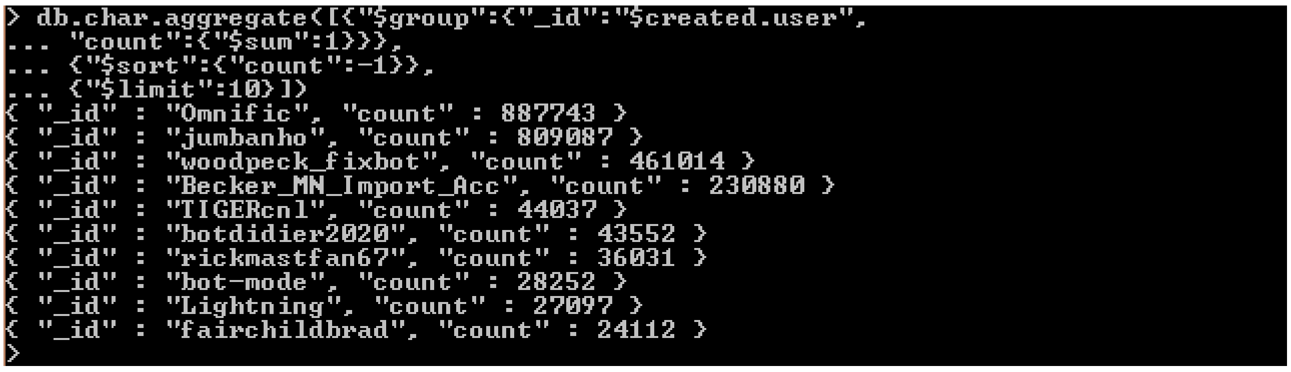
```
> db.char.find().count()
2781951
>
```

Total number of unique users



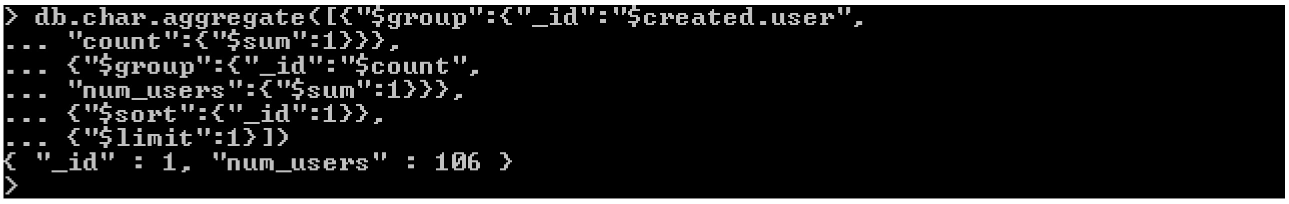
```
> db.char.distinct("created.user").length
523
> _
```

Top 10 contributing users



```
> db.char.aggregate([{"$group":{"_id":"$created.user",
... "count":{"$sum":1}}},
... {"$sort":{"count":-1}},
... {"$limit":10}])
{ "_id" : "Omnific", "count" : 887743 }
{ "_id" : "jumbanho", "count" : 809087 }
{ "_id" : "woodpeck_fixbot", "count" : 461014 }
{ "_id" : "Becker_MN_Import_Acc", "count" : 230880 }
{ "_id" : "TIGERcnl", "count" : 44037 }
{ "_id" : "botdidier2020", "count" : 43552 }
{ "_id" : "rickmastfan67", "count" : 36031 }
{ "_id" : "bot-mode", "count" : 28252 }
{ "_id" : "Lightning", "count" : 27097 }
{ "_id" : "fairchildbrad", "count" : 24112 }
>
```

Number of users who have made only one post



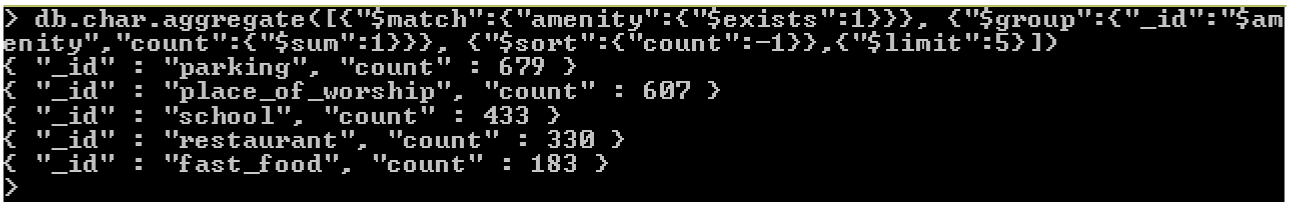
```
> db.char.aggregate([{"$group":{"_id":"$created.user",
... "count":{"$sum":1}}},
... {"$group":{"_id":"$count",
... "num_users":{"$sum":1}}},
... {"$sort":{"_id":1}},
... {"$limit":1}])
{ "_id" : 1, "num_users" : 106 }
>
```

Additional ideas about users

- The Top user(Omnific) contibution is: 31.91%
- Top 3 users contribution is: 77.57%
- Top 10 users contribution is: 93.17%

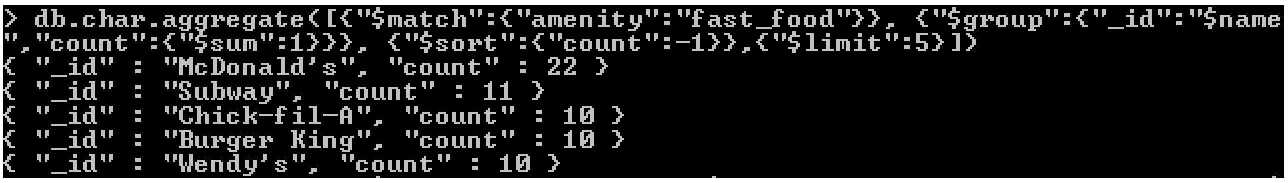
Additional MongoDB Queries

Top 5 amenity types



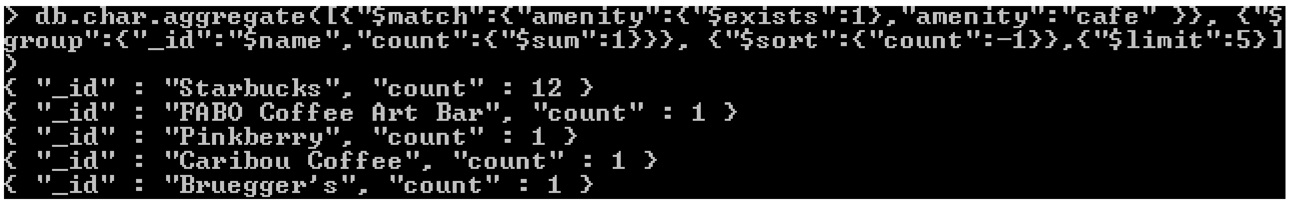
```
> db.char.aggregate([{"$match":{"amenity":{"$exists":1}}}, {"$group":{"_id":"$am
enity","count":{"$sum":1}}}, {"$sort":{"count":-1}},{"$limit":5}])
{ "_id" : "parking", "count" : 679 }
{ "_id" : "place_of_worship", "count" : 607 }
{ "_id" : "school", "count" : 433 }
{ "_id" : "restaurant", "count" : 330 }
{ "_id" : "fast_food", "count" : 183 }
>
```

Top 5 fast food restaurants



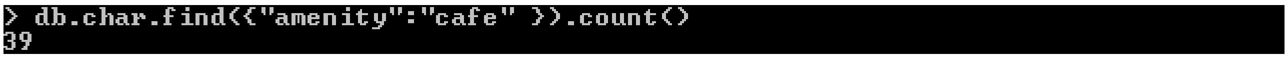
```
> db.char.aggregate([{"$match":{"amenity":"fast_food"}}, {"$group":{"_id":"$name
","count":{"$sum":1}}}, {"$sort":{"count":-1}},{"$limit":5}])
{ "_id" : "McDonald's", "count" : 22 }
{ "_id" : "Subway", "count" : 11 }
{ "_id" : "Chick-fil-A", "count" : 10 }
{ "_id" : "Burger King", "count" : 10 }
{ "_id" : "Wendy's", "count" : 10 }
```

Top 5 café shops



```
> db.char.aggregate([{"$match":{"amenity":{"$exists":1},"amenity":"cafe" }}, {"$
group":{"_id":"$name","count":{"$sum":1}}}, {"$sort":{"count":-1}},{"$limit":5}]
)
{ "_id" : "Starbucks", "count" : 12 }
{ "_id" : "FABO Coffee Art Bar", "count" : 1 }
{ "_id" : "Pinkberry", "count" : 1 }
{ "_id" : "Caribou Coffee", "count" : 1 }
{ "_id" : "Bruegger's", "count" : 1 }
```

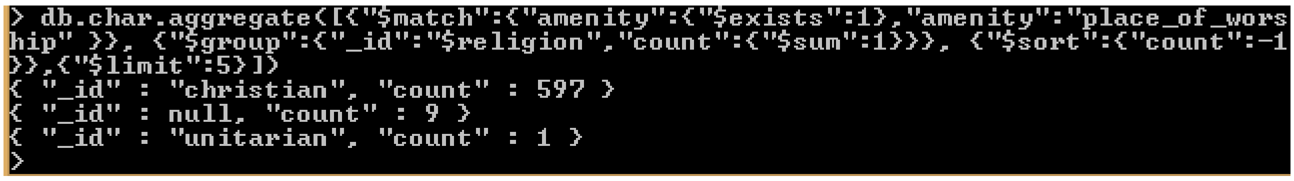
Total number of café shops



```
> db.char.find({"amenity":"cafe" }).count()
39
```

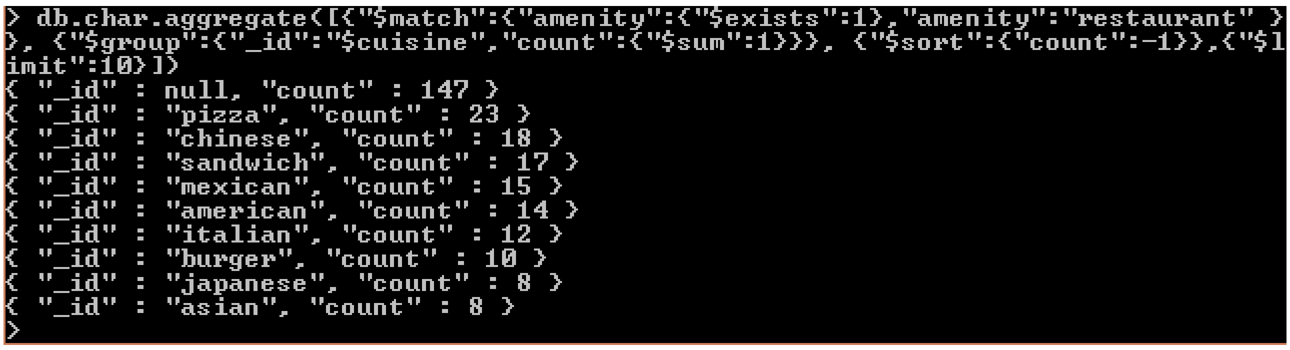
Top 3 religious groups

I gave a query for top 5, but there were only 3 groups.



```
> db.char.aggregate([{"$match":{"amenity":{"$exists":1},"amenity":"place_of_wors
hip" }}, {"$group":{"_id":"$religion","count":{"$sum":1}}}, {"$sort":{"count":-1
}},{"$limit":5}])
{ "_id" : "christian", "count" : 597 }
{ "_id" : null, "count" : 9 }
{ "_id" : "unitarian", "count" : 1 }
>
```

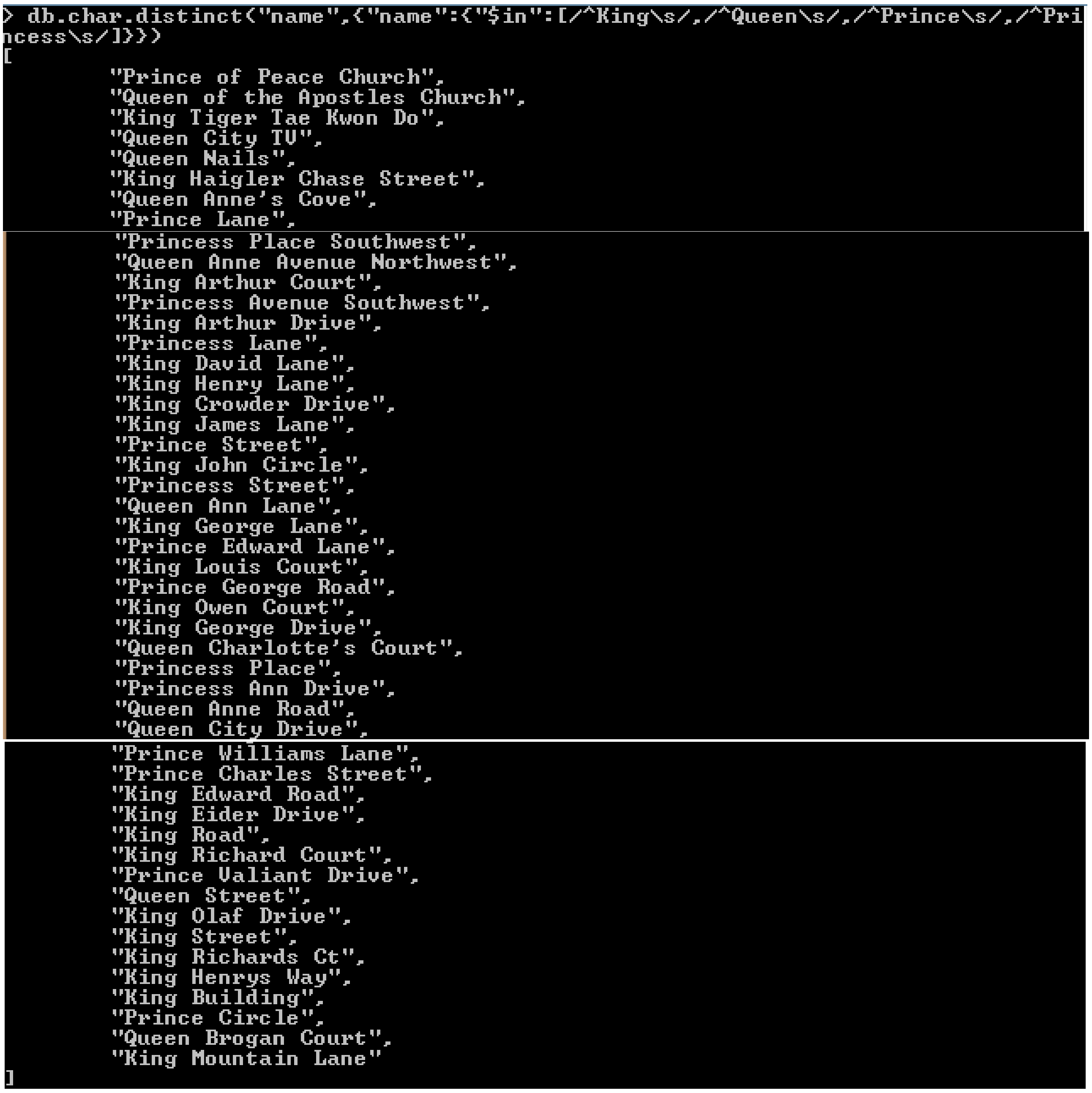
Top 10 cuisines



```
> db.char.aggregate([{"$match":{"amenity":{"$exists":1},"amenity":"restaurant" }
}, {"$group":{"_id":"$cuisine","count":{"$sum":1}}}, {"$sort":{"count":-1}},{"$l
imit":10}])
{ "_id" : null, "count" : 147 }
{ "_id" : "pizza", "count" : 23 }
{ "_id" : "chinese", "count" : 18 }
{ "_id" : "sandwich", "count" : 17 }
{ "_id" : "mexican", "count" : 15 }
{ "_id" : "american", "count" : 14 }
{ "_id" : "italian", "count" : 12 }
{ "_id" : "burger", "count" : 10 }
{ "_id" : "japanese", "count" : 8 }
{ "_id" : "asian", "count" : 8 }
>
```

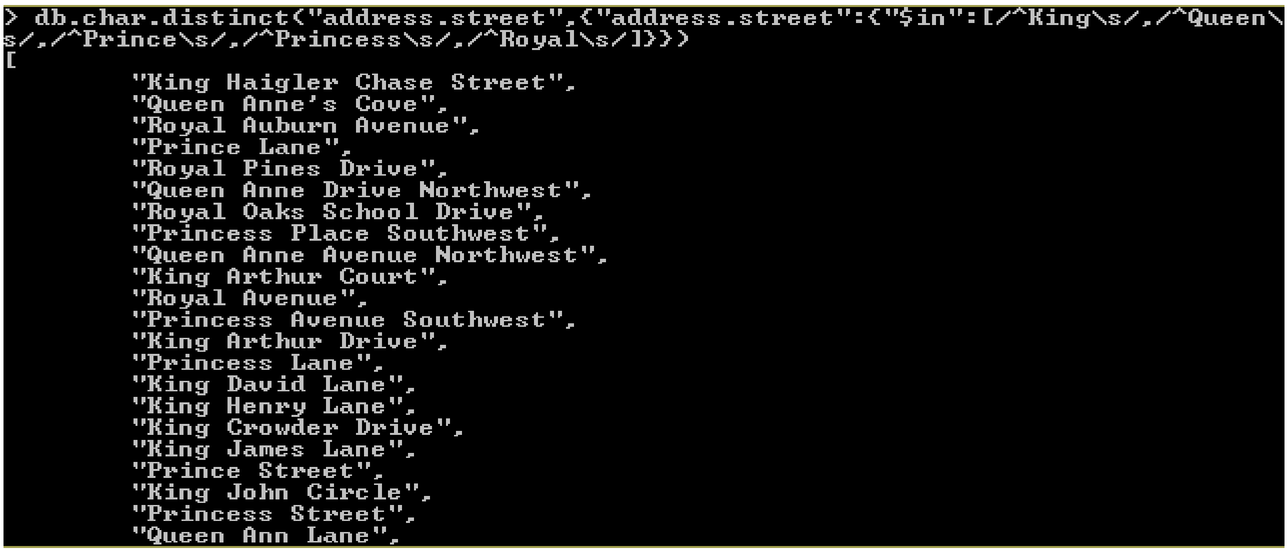
Unfortunately several types of cuisines are not documented.

I noticed that there were several streets, landmarks and buildings named after royalty like “King”, “Queen”, “Prince”, and “Princess”.

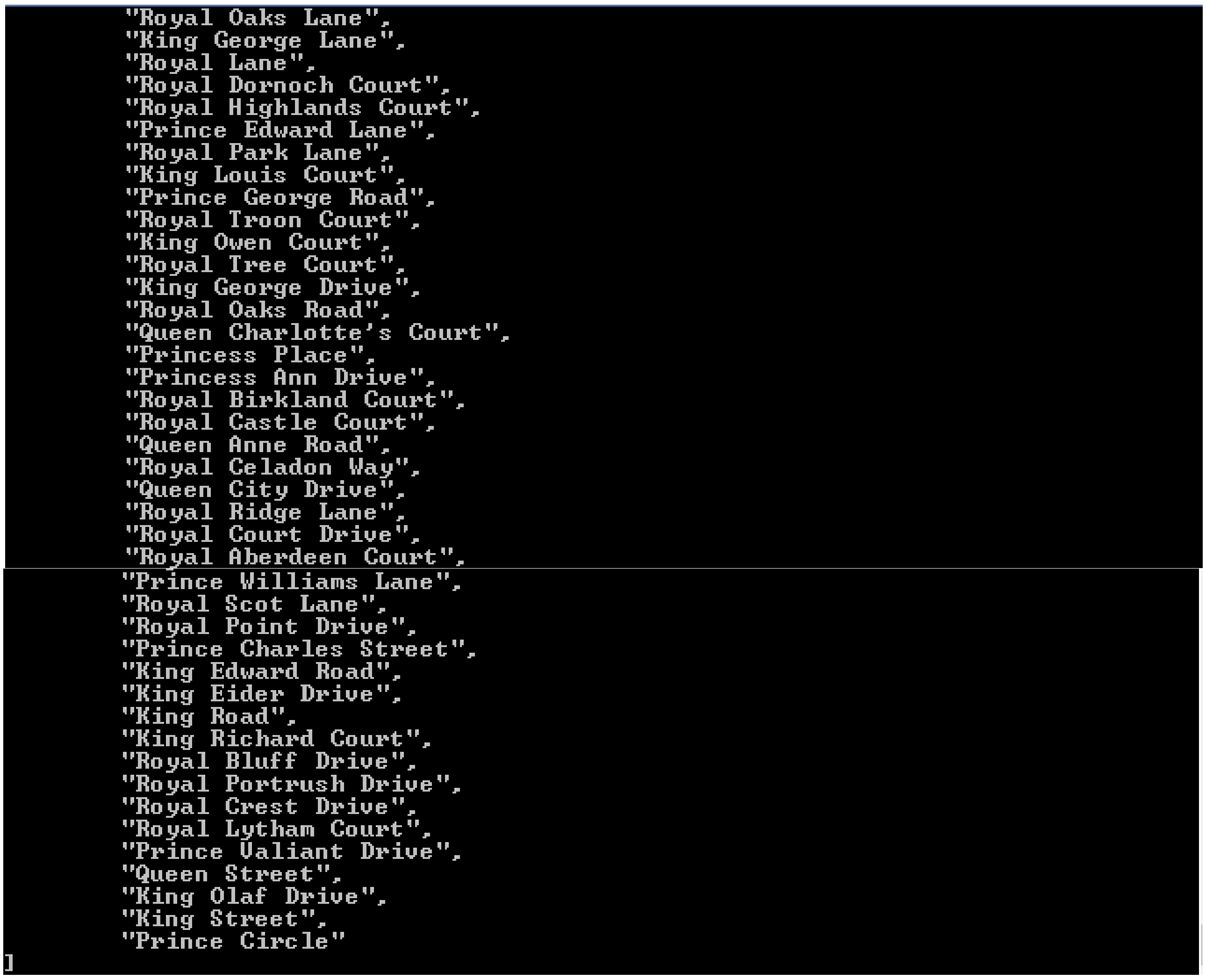


```
> db.char.distinct("name",{"name":{"$in":[/^King\s/,/^Queen\s/,/^Prince\s/,/^Pri
ncess\s/]}})
[
        "Prince of Peace Church",
        "Queen of the Apostles Church",
        "King Tiger Tae Kwon Do",
        "Queen City TV",
        "Queen Nails",
        "King Haigler Chase Street",
        "Queen Anne's Cove",
        "Prince Lane",
        "Princess Place Southwest",
        "Queen Anne Avenue Northwest",
        "King Arthur Court",
        "Princess Avenue Southwest",
        "King Arthur Drive",
        "Princess Lane",
        "King David Lane",
        "King Henry Lane",
        "King Crowder Drive",
        "King James Lane",
        "Prince Street",
        "King John Circle",
        "Princess Street",
        "Queen Ann Lane",
        "King George Lane",
        "Prince Edward Lane",
        "King Louis Court",
        "Prince George Road",
        "King Owen Court",
        "King George Drive",
        "Queen Charlotte's Court",
        "Princess Place",
        "Princess Ann Drive",
        "Queen Anne Road",
        "Queen City Drive",
        "Prince Williams Lane",
        "Prince Charles Street",
        "King Edward Road",
        "King Eider Drive",
        "King Road",
        "King Richard Court",
        "Prince Valiant Drive",
        "Queen Street",
        "King Olaf Drive",
        "King Street",
        "King Richards Ct",
        "King Henrys Way",
        "King Building",
        "Prince Circle",
        "Queen Brogan Court",
        "King Mountain Lane"
]
```

We can see below that these street names have names of royalty.



```
> db.char.distinct("address.street",{"address.street":{"$in":[/^King\s/,/^Queen\
s/,/^Prince\s/,/^Princess\s/,/^Royal\s/]}})
[
        "King Haigler Chase Street",
        "Queen Anne's Cove",
        "Royal Auburn Avenue",
        "Prince Lane",
        "Royal Pines Drive",
        "Queen Anne Drive Northwest",
        "Royal Oaks School Drive",
        "Princess Place Southwest",
        "Queen Anne Avenue Northwest",
        "King Arthur Court",
        "Royal Avenue",
        "Princess Avenue Southwest",
        "King Arthur Drive",
        "Princess Lane",
        "King David Lane",
        "King Henry Lane",
        "King Crowder Drive",
        "King James Lane",
        "Prince Street",
        "King John Circle",
        "Princess Street",
        "Queen Ann Lane",
```



```
        "Royal Oaks Lane",
        "King George Lane",
        "Royal Lane",
        "Royal Dornoch Court",
        "Royal Highlands Court",
        "Prince Edward Lane",
        "Royal Park Lane",
        "King Louis Court",
        "Prince George Road",
        "Royal Troon Court",
        "King Owen Court",
        "Royal Tree Court",
        "King George Drive",
        "Royal Oaks Road",
        "Queen Charlotte's Court",
        "Princess Place",
        "Princess Ann Drive",
        "Royal Birkland Court",
        "Royal Castle Court",
        "Queen Anne Road",
        "Royal Celadon Way",
        "Queen City Drive",
        "Royal Ridge Lane",
        "Royal Court Drive",
        "Royal Aberdeen Court",
        "Prince Williams Lane",
        "Royal Scot Lane",
        "Royal Point Drive",
        "Prince Charles Street",
        "King Edward Road",
        "King Eider Drive",
        "King Road",
        "King Richard Court",
        "Royal Bluff Drive",
        "Royal Portrush Drive",
        "Royal Crest Drive",
        "Royal Lytham Court",
        "Prince Valiant Drive",
        "Queen Street",
        "King Olaf Drive",
        "King Street",
        "Prince Circle"
]
```

Other ideas about the dataset

One of the main problem we are facing over here is the accuracy of the dataset. It seems to me that most of the users have submitted the information through computers. Rather than that, if we had used mobile phones, we could have used its GPS feature to accurately submit the information regarding the respective location. With a GPS data processor, in addition to a more robust data processor like data.py, we would be able to submit a cleaner data to openstreetmap.org

I believe that the data structure is flexible enough to incorporate a vast multitude of user generated quantitative and qualitative data beyond that of simply defining a virtual map. I believe that extending this open source project to include data such as user reviews of establishments, subjective areas of what bound a good and bad neighbourhood, housing price data, school reviews, walkability, quality of mass transit, and on would form a solid foundation of robust recommender systems. These recommender systems could aid users in anything from finding a new home or apartment to helping a user decide where to spend a weekend afternoon.

Unfortunately, it appears that, at least for the area of the world that I analysed, the mapping data is far too incomplete to be able to implement such recommender systems. I believe that the OpenStreetMap project would greatly benefit from visualizing data on content generation within their maps. For example, a heat map layer could be overlaid on the map showing how frequently or how recently certain regions of the map have been updated. These map layers could help guide users towards areas of the map that need attention in order to help more fully complete the data set.

Conclusion

While working with the Charlotte dataset, I found that most of the data contributed by the users to be not standardised. This dataset was meant for the Charlotte city, but, it seems to be for the Charlotte Metropolitan Area. By using my audit.py and data.py, I hope I have cleaned my dataset well and I hope that many benefit with these findings and cleaner data.