1. Problems Encountered in the Map

After initially downloading a small sample size of Dallas area and running it against a provisional data.py file, I noticed two main problems with the data, in the following order:

- Abbreviated street names
- Inconsistent website information (“McDonalds”: u'http://mcdonalds.com”, u'http://www.mcdonalds.com/', u'http://www.mctexas.com/23450')

Abbreviated Street Names

When the data was audited it was found that were abbreviated street names and it was replaced by appropriate names by cleaning the street names.

Inconsistent website information provided

When the entire data set was narrowed down to ones having name and website on them ,website information was inconsistent about various names.eg :McDonald’s being listed as www.mctexas.com.Using the code snippet below yielded various problems with the website information

```
from pymongo import MongoClient
import pprint
pymongoclient=MongoClient("localhost", 27017)
db = pymongoclient.data_wrangling_project
coll=db.dallas_texas


def make_pipeline():
    # complete the aggregation pipeline

match={"$match":{"name":{"$exists":1},"website":{"$exists":1}}}

group={"$group":{"_id":"$name","website":{"$addToSet":"$website"}
,  "count":{"$sum":1}}}
    limit={"$limit":10}
    sort={"$sort":{"count":-1}}
    pipeline = [match,group,sort,limit]
    return pipeline

def aggregate(db, pipeline):
    return [doc for doc in db.aggregate(pipeline)]
```

```
if __name__ == '__main__':

    pipeline=make_pipeline()
    result = aggregate(coll, pipeline)
    pprint.pprint(result)
```

2. Data Overview

This section contains basic statistics about the dataset and the MongoDB queries used to gather them.

File sizes

```
dallas_texas.osm ......... 598 MB
dallas_texas.osm.json .... 837 MB
```

# Number of documents

```
>
db.dallas_texas.find().count()

2975988
```

# Number of nodes

```
> result= db.dallas_texas.find({"type":"node"}).count()
  print result

      2673394
```

# Number of ways

```
> result= db.dallas_texas.find({"type":"node"}).count()
  print result

      302551
```

# Number of unique users

```
> result=db.dallas_texas.distinct("created.user")
  print len(result)

      1820
```

3. Additional Suggestions

# Top 1 contributing user

```
> def make_pipeline():
    # complete the aggregation pipeline
    group={"$group":{"_id":"$created.user","count": {"$sum":1}}}
    limit={"$limit":10}
    sort={"$sort":{"count":-1}}
    pipeline = [group,sort,limit]
    return pipeline

def aggregate(db, pipeline):
    return [doc for doc in db.aggregate(pipeline)]


if __name__ == '__main__':

    pipeline=make_pipeline()
    result = aggregate(coll, pipeline)
    pprint.pprint(result)
```

It is interesting to note here that nearly 38% of contributions was made by one user **woodpeck_fixbot**

# Highest Number of amenities available

```
match={"$match":{"amenity":{"$exists":1}}}
group={"$group":{"_id":"$amenity", "count":{"$sum":1}}}
limit={"$limit":10}
sort={"$sort":{"count":-1}}
pipeline =[match,group,sort,limit]
```

It was interesting to find that parking lots topped the chart when it came to the count of amenities that were available in Dallas.Next was the most guessed religious places.This makes us to believe that people in Dallas like to stay religious and visit places of worship more.

Sort cities by count, descending

```
match={"$match":{"address.city":{"$exists":1}}}
group={"$group":{"_id":"$address.city", "count":{"$sum":1}}}
limit={"$limit":10}
sort={"$sort":{"count":-1}}
pipeline =[match,group,sort,limit]
```

These results confirmed a point that this is a metroplex extract rather than a city extract which has cities such as Frisco, Plano on the list. Frisco surprisingly heads the list followed by Plano in terms of highest count of cities.

Conclusion

This first step in the cleaning of data was a great exposure for me to get to know the process of auditing and cleaning data. With more advanced techniques one can believe that this dataset can be developed into a highly competent dataset to be used by a GPS system.