1. **Problems Encountered in the Map:**
I chose Manhattan area in New York City for my Open Street Map project.
After downloading the street data and running against the audit.py file which is used to audit street names in the map data, I found a couple of issues with the data, specifically in the street name:

- **Inconsistent street name type:**

Upon auditing the street name types, I found there are a couple street name types that are inconsistent throughout.

Audit street name type
In [ ]: audit(man)

```
Out[ ]:…
   'Ave': {'10th Ave',
           '64th St and 5th Ave',
           'Anderson Ave',
           'Hudson Ave',
           'Norman Ave',
           'Third Ave'},
          'Avene': {'Madison Avene'},
          'Aveneu': {'St. Nicholas Aveneu'},
          'Avenue,#392': {'Columbus Avenue,#392'},
      …
```

For example, “Avenue” is spelled inconsistently with some spelled as Ave, others Avenue, and some with misspellings like Avene or Aveneu.

- **Missing street type:**

Some streets are missing street type.

In [ ]: audit(man)

```
Out[ ]:…
       '27th': {'W 27th'},
       '29th': {'29th'}
      …
```

For example, some street names only show up as 27th or 29th, with out knowing what kind of street type the address is referring to, it could be 27th street or 27th Avenue.

- **Ober-abbreviated street names:**

Some street names are overly abbreviated, for example, 'Central Park S' may refer to “Central Park South” and W 27th may refer to West 27th [Street] (again, this is related to the issue above, for this particular address, we don’t know exactly what kind of address that is, but my guess is street as I live in Manhattan and that’s my best guess.

- **Incorrect Cities in Manhattan Database:**

The Manhattan city database actually contains many cities that are outside of Manhattan, even including cities in New Jersey. This may be due to incorrect lat lon data from GPS for those cities with the lat lon from Manhattan.

In addition, the city name and borough names seem to be mixed together. For example, Long Island City is a city in Queens Borough, but Queens also shows up in the city name filed.

```
In [ ]:
result = [doc for doc in db.maps.aggregate([
{"$group":{"_id":"$address.city", "count":{"$sum":1}}}])]

pprint.pprint(result)
```

```
Out[ ]:…
[{u'_id': u'New York NY', u'count': 1},
 {u'_id': u'fort lee', u'count': 1},
 {u'_id': u'Roosevelt Island', u'count': 1},
 {u'_id': u'new york', u'count': 2},
 {u'_id': u'Ridgewood', u'count': 2},
 {u'_id': u'Union City', u'count': 1},
 {u'_id': u'Blissville', u'count': 1},
 {u'_id': u'New York city', u'count': 1},
 {u'_id': u'Brooklyn, NY', u'count': 4},
 {u'_id': u'new York', u'count': 2},
 {u'_id': u'NEW YORK CITY', u'count': 3},
 {u'_id': u'West New York', u'count': 2},
 {u'_id': u'Woodside', u'count': 5},
 {u'_id': u'North Bergen', u'count': 1},
 {u'_id': u'Manhattan NYC', u'count': 1},
 {u'_id': u'Sunnyside', u'count': 11},
 {u'_id': u'Queens, NY', u'count': 1},
 {u'_id': u'Fort Lee', u'count': 35},
…
```

2. Data Overview:

#size of the file:

- manhattan_new-york.osm: 238 MB
- manhattan_new-york.osm.json: 301 MB

#number of documents:

```
In [ ]: db.maps.find().count()
Out[ ]:1013930
```

#number of unique users:

```
In [ ]: len(db.maps.distinct("created.user"))
Out[ ]:1258
```

#number of nodes and ways

```
In [ ]:result = [doc for doc in db.maps.aggregate([
{"$group":{"_id":"$nodetype", "count":{"$sum":1}}}])]

In [ ]:pprint.pprint(result)
```

```
Out[]:[{u'_id': u'node', u'count': 874835}, {u'_id': u'way', u'count':
139095}]
```

#number of chosen type of nodes, like cafes, shops etc:

```
In [ ]: result = [doc for doc in db.maps.aggregate([
        {'$match': {'amenity':{'$exists': True}}},
        {"$group":{"_id":"$amenity", 'count':{"$sum":1}}},
        {'$sort':{'count':-1}},
        {'$limit':10}
       ])]

In [ ]: pprint.pprint(result)
```

```
Out[]:[{u'_id': u'bicycle_parking', u'count': 3254},
 {u'_id': u'restaurant', u'count': 1171},
 {u'_id': u'school', u'count': 580},
 {u'_id': u'parking', u'count': 580},
 {u'_id': u'place_of_worship', u'count': 562},
 {u'_id': u'cafe', u'count': 483},
 {u'_id': u'bicycle_rental', u'count': 354},
 {u'_id': u'fast_food', u'count': 338},
 {u'_id': u'bank', u'count': 275},
 {u'_id': u'bench', u'count': 249}]
```

#Top 5 contributing user:

```
In [ ]:result = [doc for doc in db.maps.aggregate([
    {"$group":{"_id":"$created.user", "count":{"$sum":1}}},
    {"$sort":{"count":-1}},
    {'$limit':5}
   ])]
In [ ]:pprint.pprint(result)
```

```
Out[]: [{u'_id': u'Rub21_nycbuildings', u'count': 683689},
        {u'_id': u'lxbarth_nycbuildings', u'count': 76585},
        {u'_id': u'robgeb', u'count': 58256},
        {u'_id': u'Korzun', u'count': 23450},
        {u'_id': u'woodpeck_fixbot', u'count': 10579}]
```

3. Additional Ideas:

When I audit the data, I found out that using the function to ignore elements with problem characters (problemchars = re.compile(r'[=\+/&<>;\'"\?%#$@\,\. \t\r\n]')), we are actually removing something interesting:

```
In [ ]: for _, element in ET.iterparse(man):
```

```
            if element.tag == "tag":
                key = element.attrib['k']
                if problemchars.search(key):
                    print key
```

```
Out[]:cityracks.housenum
cityracks.installed
cityracks.large
cityracks.rackid
…
```

It turns out an element tag in Manhattan area has values like "`cityracks.housenum`", "`city racks.large`", "`cityracks.small`", and because we filter out elements with "." In the value, we lose this information. Whereas cityracks turns out to be bike racks in NYC, and we could possible use this information to build out an app that shows where the bike racks are in the city for bikers, and bikers are able to check where they can park their bike.

Therefore, when audit this data, we can create an exception to exclude problematic tags with cityracks string and use them in our database.

And currently, Manhattan has citibike location (bike share service), and this can also fuel the bikers to find out where to get their biek share.

4. Other Data Explorations

#Most popular cuisines

Seems Italian food, American food, pizza, Mexican and Chinese are pretty popular in Manhattan. Note that many restaurants do not have cuisine type in the map, as there are 410 restaurants with no cuisine type.

```
In []:
result = [doc for doc in db.maps.aggregate([
    {'$match': {'amenity':{'$exists': True}, 'amenity':'restaurant'}},
    {"$group":{"_id":"$cuisine"}, "count":{'$sum':1}},
    {'$sort':{'count':-1}},
    {'$limit':10}
   ])]
pprint.pprint(result)
```

```
Out[]:
[{u'_id': None, u'count': 410},
 {u'_id': u'italian', u'count': 88},
 {u'_id': u'american', u'count': 63},
 {u'_id': u'pizza', u'count': 63},
 {u'_id': u'mexican', u'count': 60},
```

```
 {u'_id': u'chinese', u'count': 43},
 {u'_id': u'french', u'count': 32},
 {u'_id': u'japanese', u'count': 30},
 {u'_id': u'burger', u'count': 29},
 {u'_id': u'thai', u'count': 26}]
```

#Number of cafes in Manhattan

There are 483 cafes in Manhattan, and a quarter of them are starbucks.

```
in[]: db.maps.find({'amenity':'cafe'}).count()
Out[]: 483
```

```
In []:
result = [doc for doc in db.maps.aggregate[
    {'$match': {'amenity':{'$exists': 1}, 'amenity': 'cafe'}},
    {"$group":{"_id":"$name", 'count':{"$sum":1}}},
    {'$sort':{'count':-1}},
    {'$limit':5}
]]
pprint.pprint(result)

Out[]:
[{u'_id': u'Starbucks', u'count': 96},
 {u'_id': u'Starbucks Coffee', u'count': 23},
 {u'_id': u"Dunkin' Donuts", u'count': 22},
 {u'_id': u'Le Pain Quotidien', u'count': 13},
 {u'_id': None, u'count': 7}]
```

5. Conclusions

After reviewing the data of Manhattan, it seems that data data quality is not too bad, though with a couple of incomplete, inconsistent or missing values on the street names. It seems the Manhattan data is pretty well developed, with more than 1 thousand users contributing Manhattan map data to Open Street Map. This is likely that more users live in Manhattan and/or are interested in contributing to the data or correcting the data; therefore, Manhattan data could be readily used to build data products.