Data Wrangling with MongoDB - Final Project:
Exploring OpenStreetMap Data

For my OpenStreetMaps dataset, I chose to explore Nashville, TN, USA, a.k.a. Music City U.S.A., a.k.a my hometown. I started the project by downloading an OSM of an appropriate size (~238 MB unzipped), quickly inspected and cleaned the data using some python scripts, converted the data into a json format, and uploaded into MongoDB and finally queried the data there.

Section 1: Problems Encountered in the Map

1) Wide variation in street ending abbreviations

As shown below, the there are a number of distinct spellings of Avenue, Court, Drive, Lane, Pike, Parkway, Street, etc. In cleaning the data, these values were corrected to the the most formal spelling, i.e. capitalized first letter and full name. So Dr became Drive and st became Street.

	A	B
1	#105	1
2	#112	1
3	#300	1
4	100	2
5	37128	1
6	4	1
7	41	1
8	7th	1
9	96	1
10	Alley	17
11	ave	1
12	Ave	13
13	avenue	2
14	Avenue	50
15	B220	1
16	Blvd	11
17	Boulevard	115
18	Broadway	13
19	Bypass	1
20	Circle	3
21	Court	82
22	Ct	1
23	Dr	15
24	Drive	359
25	East	1
26	Flagpole	1
27	Fork	1
28	Foxborough	9
29	Foxland	2
30	Glen	1
31	Heights	1
32	Highway	6
33	Hollow	1
34	Lane	197
35	Ln	1
36	Loop	1
37	Mission	1
38	North	18
39	Padgett	1
40	Parkway	18
41	Pass	2
42	Pike	115
43	pike	1
44	Pk	3
45	Pkwy	1
46	Pky	1
47	Pl	3
48	Place	8
49	Rd	19
50	Road	91
51	South	15
52	Springs	1
53	Square	5
54	St	3
55	st	2
56	St.	1
57	Street	146
58	Terrace	2
59	Trace	1
60	Trail	1
61	Vandy	1
62	Way	12
63	Wynthrope	1

2) Non-traditional street endings

In addition to variations on traditional addresses (e.g. Street vs. st.), there were a number of seemingly non-traditional endings (e.g. Hollow). Further exploration of these showed that while non-traditional, streets like 'Hollow' are region specific endings and did not, in fact, require correction. The same was true in many cases with numbers, such as #105, where the full address was a non-traditional but correct. (e.g. Highway #105.)

3) Clearly incorrect endings

Surprisingly, in only one case was a zipcode incorrectly entered as a street (37218.) This was clearly incorrect and was deleted in the cleaning process. This is in contrast to cases with too little information to infer a correct address (7th instead of 7th Ave or 7th St.) but which were not overly incorrect and thus omitted from cleaning.

4) Incorrect zip codes

I wrote a quick script to look at zipcodes in the OSM file.

```
Nashville = open("nashville.osm", "r")

def is_zip(elem):
    return (elem.tag == "tag") and (elem.attrib['k'] ==
"tiger:zip_left")

#The following function writes the zip codes to a separate csv
#for inspection
def print_sorted_dict(d):
    keys = d.keys()
    writer = csv.writer(open('zip_codes2.csv', 'wb'))
    keys = sorted(keys, key = lambda s: s.lower())
    for k in keys:
        v = d[k]
        writer.writerow([k,v])

#this function parses the OSM file and cycles through elements
#with 'way' tags. If these way tags have a zip_code (see
#is_zip), it either adds that zip to a dictionary or adds one
#to an existing zip code.

def zip(filename):
    zip_code = {}
    for event, elem in ET.iterparse(filename):
         if elem.tag == 'way':
               for tags in elem.iter("tag"):
```

```
                        if is_zip(tags):
                              zips = tags.attrib['v']
                              if zips in zip_code:
                              #if zipcode exists in dictionary add
                              #one
                                    zip_code[zips] += 1
                              else:
                              #if not, add zipcode to dictionary
                                    zip_code[zips] = 1
    print_sorted_dict(zip_code)

if __name__ == '__main__':
    zip(nashville)
```

From this I found a combination of dual zip codes (with colons) and blatantly out-of-Nashville zip codes (i.e. zip codes that do not start with ’37’). While clearly not Nashville zip codes, I was hesitant to change or clean them out as the roads could start in Kentucky (i.e. being labeled with a ’42’ zip code) and then enter into Nashville. Similarly, the multiple zip codes (with colons) I left unchanged as it was unclear whether this was ‘dirty’ data or a location between zip codes. Overall, this part was difficult to interpret as I found the OSM documentation unclear on how ‘way’ zip codes were defined.

38487	33
38544:38548	1
38547	40
38548	28
38552	8
38560	33
38562	21
38563	41
38564	21
38567	22
38567:38569	1
38569	11
38582	7
38583	3
38588	9
42120	20
42133	7
42134	13
42140	32
42153	3
42164	7
42202	5
42204	2
42220	1
42223	69
42234	30
42240	3
42254	1
42262	43
42265	1
42286	2

Section 2: Data Overview

File Sizes:

Initial Extract (Unzipped) - 'nashville.osm' - 238 MB
Audited + Corrected File - 'nashville_audit.osm - 240.9 MB

Imported json file into MongoDB using mongoimport:

```
>mongoimport -d examples -c nashville --file nashville_mongo.json
      File size 262.5 MB
```

There are just over 1.1 million documents uploaded into the Nashville collection. The vast majority of these documents are nodes (~1milliion) with the rest populated by ways (~100K).

```
> print "The number of files is %d" % db.nashville.find().count()
      1138391
> print "The number of nodes is %d" % db.nashville.find({"type":
"node"}).count()
      1037910
> print "The number of ways is %d" % db.nashville.find({"type":
"way"}).count()
      100449
> print "The number of unique users is %d" %
len(db.nashville.distinct("created.user"))
      724
> restaurants = db.nashville.find({'amenity':'restaurant',
'type':'node'}).count()
> print restaurants
      238
```

Section 3: Thoughts on the data….

1) I don't think I was surprised to see that the ratio of nodes to ways was ~10:1 but it did make me curious as to what the ratio was like for other cities. Would more densely packed cities (i.e. NYC) have a higher ratio or nodes to ways and would lower density cities (i.e. LA) be lower? It struck me that this ratio could possibly make for an interesting cross city comparison.

2) Knowing that Nashville is an especially religious city, I queried based on how many locations were religiously affiliated. I found that there are nearly 2500 religious institutions.

```
> db.nashville.find({'amenity':'place_of_worship','type':'node',
'name' : {'$exists':1}}).count()
      2456
```

I had a hunch of how this would be distributed across religions and queried again, this time grouping the place of worship by type of religion.

```
>db.nashville.aggregate([{"$match":{"name":{"$exists":1},
"amenity":"place_of_worship"}},{"$group":{"_id":"$religion","count":
{"$sum":1}}}, {"$sort":{"count":-1}},{"$limit":1}])
```

```
      [{“_id":"christian","count":2406}]
```

Overwhelmingly, the vast majority of places of worship (97.96%!) are affiliated with Christianity. While I knew it would be a majority, having lived there, this seemed a bit too high. Out of curiosity I queried for another type of religion institution; this time Jewish.

```
> db.nashville.find({‘amenity':'place_of_worship', ‘religion’:
‘jewish’, ’type’:’node’, 'name' : {‘$exists':1}}).count()
      1

>pprint.pprint(db.nashville.find_one({'amenity':'place_of_worship','re
ligion': 'jewish','type':'node', 'name' : {'$exists':1}}))

{u'_id': ObjectId('55464ecc1531b7221589d495'),
 u'amenity': u'place_of_worship',
 u'created': {u'changeset': u'3802940',
           u'timestamp': u'2010-02-06T07:41:49Z',
           u'uid': u'51045',
           u'user': u'Geogast',
           u'version': u'2'},
 u'ele': u'160',
 u'gnis:county_id': u'037',
 u'gnis:created': u'03/01/1990',
 u'gnis:feature_id': u'1324817',
 u'gnis:state_id': u'47',
 u'id': u'356879675',
 u'name': u'West End Synagogue',
 u'pos': [36.1325578, -86.8319442],
 u'religion': u'jewish',
 u'type': u’node'}
```

I only received 1 node, the location of which I am anecdotally familiar with. This tells me that the dataset is far from complete because even though the number of religion institutions is most likely overwhelmingly Christian, I know there to be more than one synagogue in Nashville. This is not only an indication that the dataset is incomplete but also brings up the point of the influence of contributing users. If users typically contribute to places they are familiar with, it is possible that there are very few Jewish contributors from Nashville on the site. Further exploration would be interesting regarding other minority related establishments/communities (e.g. historically black, hispanic, LBTQ) and the prevalence (or lack thereof) they have on OpenStreetMaps based on lack of contributing representation. Have more robust user data (i.e. gender, age, race) would be useful in determining contributing biases but this begins to get into a dangerous area of privacy invasion very quickly. Users would most likely be adverse to providing this depth of information and contributions would likely go down.

Metro Area: While initial inspection of the data included places I would not call 'Nashville' (i.e. Mt. Juilet, TN), this is not in and of itself a problem with the dataset, as you could make the case that some 'outlying' areas of metro Nashville are *still* Nashville. This does inform me, however, that the general definition of Nashville by the Open Street Maps dataset is very broad and reflects the larger metro area rather than the city itself.

```
>db.nashville.aggregate([{"$match":{"is_in":{"$exists":1}, "type":"node"}},{"$group":
{"_id":"$is_in","count": {"$sum":1}}}, {"$sort":{"count":-1}}])

    [{"_id":"Rutherford,Tennessee,Tenn.,TN,USA","count":226},
    {"_id":"Williamson,Tennessee,Tenn.,TN,USA","count":207},
    {"_id":"Montgomery,Tennessee,Tenn.,TN,USA","count":151},
    {"_id":"Davidson,Tennessee,Tenn.,TN,USA","count":117}...
```

Interestingly, Davidson County, the county where Nashville (city) is located, is ranked fourth in terms of nodes with identifying "is_in" attributes. This is puzzling (considering it has the city proper) but, given the low node counts for each of these counties, it is likely that "in_in" is quite inconsistently assigned. Tagging additional nodes through either manual assignment (which would be quite taxing) or through a script assigning 'is_in' based on zip codes would help in quickly querying where the majority of nodes are located. Alternatively, a better solution (although more complicated one) would be based on lon + lat values, which are far more prevalent and accurate in nodes than zip codes.

Conclusion:

It's clear to me that there is plenty of room for improvement with the Nashville dataset. There were some obvious issues with the data (i.e. data put in the wrong fields, data misspelled,etc.). Other areas that caused problems with querying included biased contributions and lack of data categorization (i.e. unknown country locations). But the most frustrating problem I experienced in processing this dataset was how to interpret some data fields and determine whether they were corrupt or not. The perfect example of this was the multiple zip codes in 'way' types, which could be interpreted as correct (being that roads pass through multiple zip codes or could start on the border between two) or incorrect (through improper entry) and requiring cleaning. This is potentially damaging as it could cause some to throw out or change perfectly good data due to a lack of understanding of how it should be interpreted. Conversely, it could also cause bad data to be left in by mistake.