# Introduction

In this project I download the New York xml file from OpenStreetMaps.org, parse it to get data from nodes and ways, and then clean the street names to ensure they meet the same formats. The only real problem is that it seems some street names are completely wrong and would need to either be deleted or further delved into.

As can be seen there are 11.5 million nodes, and 1.8 million ways. A lot of this code will obviously take a while to run.

```
In [26]: tags
```

```
Out[26]: defaultdict(None,
                     {'bounds': 1,
                      'member': 112118,
                      'nd': 14549647,
                      'node': 11511646,
                      'osm': 1,
                      'relation': 9724,
                      'tag': 9698321,
                      'way': 1804132})
```

Most of the problems with the street names is that they seem to be in different formats as seen below:

```
In [29]: keys
```

```
Out[29]: {'lower': 3778075, 'lower_colon': 5789589, 'other': 130657, 'problemchars': 0}
```

After doing some preliminary investigation. I begin to parse the file for street names that need fixing using expected street types as the norm, to which values are added as determined after reviewing the text output:

In [34]:
```
street_type_re = re.compile(r'\b\S+\.?$', re.IGNORECASE) #Regex expression to search for street names at the end of terms.
expected = ["Airport","Alley","Avenue", "Boulevard","Bridge","Building","Circle", \
            "Close","Court","Concourse","Commerce", "Common","Commons","Crescent","Cross","Drive",\
            "Driveway","Expressway","Highway","Lane","Loop","Park","Parkway","Path",\
            "Place","Plaza""Ridge","Road","Route","Run","Slip","Square","Street","Suite",\
            "Terrace","Trace","Trail","Thruway","Turnpike","Walk","Walkway","Way"]
# List of expected names of streets.
def is_street_name(elem):
    return (elem.attrib['k'] == "addr:street")                    #Function to determine if element has a street name.

def audit_street_type(street_types, street_name):
    m = street_type_re.search(street_name)               #Function to add street names that are not in expected to a list.
    if m:
        street_type = m.group()
        if street_type not in expected:
            street_types[street_type].add(street_name)


def audit(osm_file):
    street_types = defaultdict(set)                                 #Function to parse the xml document
    for event, elem in et.iterparse(osm_file, events=("start",)):  #And use the other functions in conjuction
        if elem.tag == "node" or elem.tag == "way":
            for tag in elem.iter("tag"):
                if is_street_name(tag):
                    audit_street_type(street_types, tag.attrib['v'])
    return street_types

with bz2.BZ2File(io.BytesIO(r.content)) as osm_file:
    street_type=audit(osm_file)
```

In [ ]:
```
with open('C://Users/Zohaib/Desktop/Lectures/Udacity/Streets.txt','w') as f:     #Print the street types to a file called
    pprint.pprint(street_type,f)                                                 #Streets.txt to create the mapping dictionary.
```

Here is some of the output of the text file created to determine all of the street types in the file:

```
defaultdict(<class 'set'>,
            {'1': {'36th St Front 1',
                   'Burt Drive ste 1',
                   'Cushing St # 1',
                   'Graham Avenue #1',
                   'Grand Concourse #1',
                   'New Jersey 17 N #1',
                   'North US Highway 1',
                   'ROAD 1',
                   'U.S. 1',
                   'U.S. Highway 1',
                   'US 1',
                   'US HIGHWAY 1',
                   'US Highway 1'},
             '10': {'NJ Route 10', 'Route 10', 'Nwe Jersey 10'},
             '10024': {'West 80th Street NYC 10024'},
             '101': {'20th Ave, Suite 101'},
             '106B': {'Lincoln Highway #106B'},
             '109': {'Central Park South Suite 109',
                     'Route 109',
                     'State Highway 109'},
             '10C': {'78th #10C'},
             '11217': {'305 Schermerhorn St., Brooklyn, NY 11217'},
             '12': {'Main St #12'},
             '120': {'H Highway 34 #120'},
             '1204': {'Journal Square #1204'},
             '1603': {'STREET 1603'},
             '17': {'New Jersey 17', 'Route 17', 'State Route 17'},
             '18': {'State Route 18', 'NJ 18'},
             '1801': {'505th 8th Avenue Suite 1801'},
             '1807': {'5th AVE 1807'},
             '185': {'Morris Avenue - #185'},
             '1A': {'Barnes Ave #1A'},
             '1B': {'East 64th Street, #1B', '26th Avenue #1B'},
             '1H': {'W 56th St #1H'},
             '1st': {'1st'},
             '2': {'55 Riverwalk Pl #2',
                   'Island 2',
                   'S Broadway #2',
                   'W 79th St #2'},
             '201a': {'Tices Lane #201a'},
             '202': {'Route 202'},
             '205': {'Broadway #205'},
             '208': {'200 Perrine Road; Suite 208',
                     'Route 208',
                     'State Route 208'},
             '21': {'Church Street #21'},
             '218650358': {'218650358'},
             '22': {'US- 22', 'US 22', 'Route 22'},
             '236': {'Purchase Street Suite 236'},
             '25A': {'Route 25A', 'NY 25A', 'State Highway 25A'},
```

As can be seen, even in the earlier entries, a lot needs to be fixed, for example Highway is formatted in at least three different ways already.

After parsing the file for odd street types, a mapping dictionary is created below to fix the street types which is seen below:

```
mapping = { "Americas\n":"Americas",
            "Ave.":"Avenue",
            "ave":"Avenue",
            "avenue":"Avenue",
            "Ave,":"Avenue",
            "Avene":"Avenue",
            "Aveneu":"Avenue",
            "Ave":"Avenue",
            "AVE.":"Avenue",
            "AVE":"Avenue",
            "AVenue":"Avenue",
            "AVENUE":"Avenue",
            "bl":"Boulevard",
            "bl":"Building",
            "Blv.":"Boulevard",
            "boulevard":"Boulevard",
            "Blvd.":"Boulevard",
            "Blvd":"Boulevard",
            "BLDG":"Building",
            "BLD":"Building",
            "Cir":"Circle",
            "Ct.":"Court",
            "Ct":"Court",
            "Ctr":"Center",
            "Crst":"Cresecent",
            "Cres":"Crescent",
            "Cmn":"Common",
            "Concrs":"Concourse",
            "Cv":"Cove",
            "drive":"Drive",
            "DRIVE":"Drive",
            "Dr.":"Drive",
            "Dr":"Drive",
            "EAST":"East",
            "E":"East",
            "Expy":"Expressway",
            "Grn":"Green",
            "HIGHWAY":"Highway",
            "Hwy":"Highway",
            "LANE":"Lane",
            "lane":"Lane",
            "Ldg":"Landing",
            "Ln":"Lane",
            "N":"North",
            "north":"North",
            "Pky":"Parkway",
            "Pkwy":"Parkway",
            "PLAZA":"Plaza",
            "PARKWAY":"Parkway",
            "Plz":"Plaza",
            "Pl":"Place",
            "Pl":"Place",
```

```
            "PLACE":"Place",
            "Pt":"Point",
            "Rd.": "Road",
            "Rd":"Road",
            "ROAD":"Road",
            "Rdg":"Ridge",
            "route":"Route",
            "route":"Route",
            "road":"Road",
            "St.": "Street",
            "St": "Street",
            "st.":"Street",
            "st ":"Street",
            "street":"Street",
            "STREET":"Street",
            "ST":"Street",
            "Ste.":"Suite",
            "Ste":"Suite",
            "STE":"Suite",
            "S":"South",
            "SOUTH":"South",
            "STREET":"Street",
            "Turnlike":"Turnpike",
            "Tunrpike":"Turnpike",
            "Tunpike":"Turnpike",
            "Tpke":"Turnpike",
            "Tirnpike":"Turnpike",
            "Ter":"Terrace",
            "Trce":"Trace",
            "WAY":"Way",
            "W.":"West",
            "W":"West",
            "west":"West",
            "WEST":"West"}
```

Finally, after creating the cleaning function, I parsed the XML document completely and outputted the relevant lines into CSV files.  After this step, the file has been iteratively parsed, and the output has been saved to csv files. Thereafter I upload the files to SQLite database called "New_York.db".

Now I access it to run a general query for information.

In [3]:
```
db=sqlite3.connect("C://Users/Zohaib/Desktop/Lectures/Udacity/sqlite_windows/New_York.db")
cursor=db.cursor()   #Connects to the database with the data from the XML file.
```

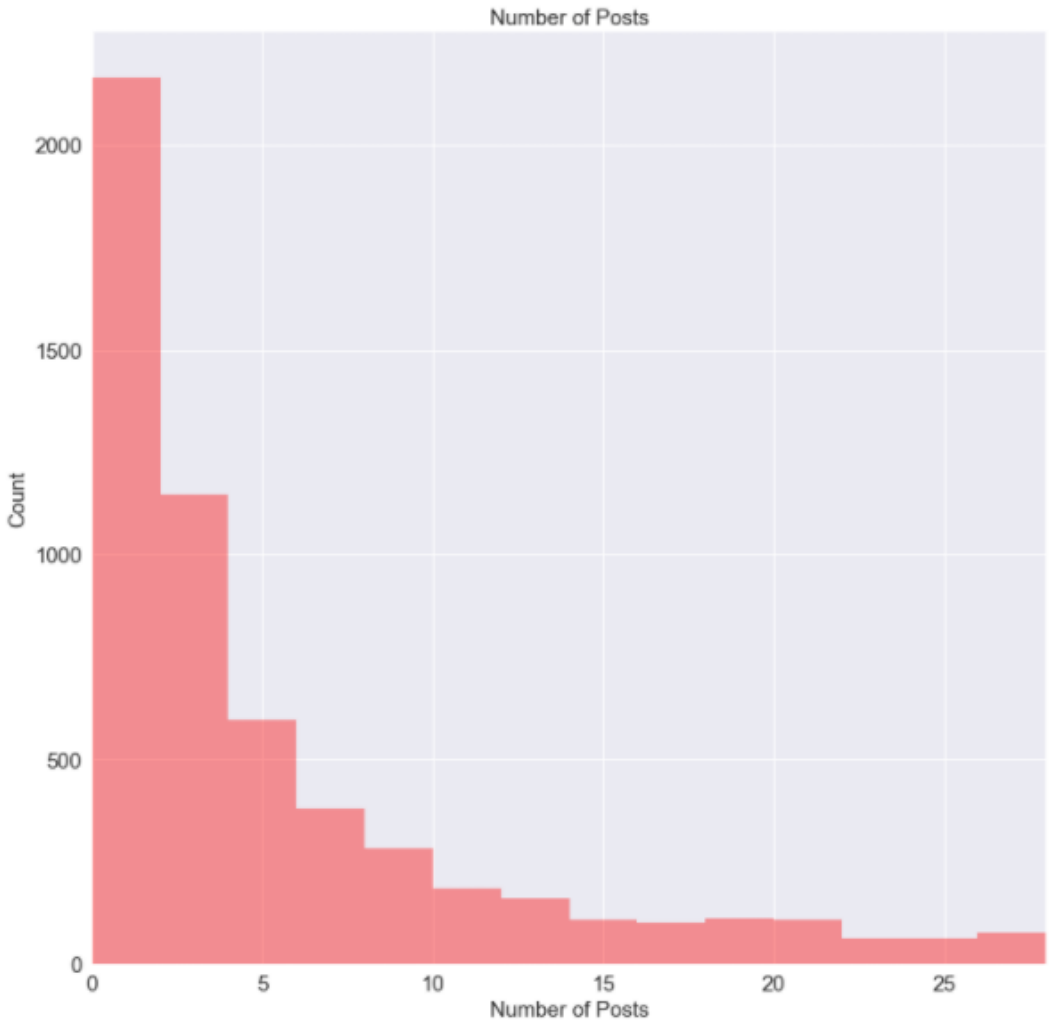
In [38]:
```
cursor.execute("select user,count(*) from nodes group by user;")
number_of_posts=[x[1] for x in cursor.fetchall()];
```

In [39]:
```
cursor.execute("select user,count(*) from ways group by user;")
number2=[x[1] for x in cursor.fetchall()]
number_of_posts=number_of_posts+number2;                          #Returns count of posts by each user.
```

In [43]:
```
stats.mstats.mquantiles(number_of_posts,prob=[.25,.5,.75])
```

Out[43]:
```
array([  1.,   5.,  28.])
```

In [48]:
```
posts_reasonable=[]
for i in number_of_posts:
    if i<=28:                                                     #Plots up to the 75th percentile of the results
        posts_reasonable.append(i)                                #to avoid outliers.
figure=plt.figure(figsize=(12,12))
sns.distplot(posts_reasonable,bins=range(0,29,2),kde=False,color='Red')
plt.xlim(0,28)
plt.title("Number of Posts",fontsize=15)
plt.xlabel("Number of Posts",fontsize=15)
plt.ylabel("Count",fontsize=15)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15);
```




As can be seen above most users post between 1 to about 5 times. Then the number of posts by users decreases steadily. This is to be expected from OpenStreetMaps.org, as many different users are posting.

Furthermore, here is a general overview of the data set as seen through some general queries to confirm what was seen before:

```
In [55]: cursor.execute("select sum1+sum2+sum3+sum4+sum5 from (select count(*) as sum1 from nodes) as sub1,\
                (select count(*) as sum2 from node_tags) as sub2,\
                (select count(*) as sum3 from ways) as sub3,\
                (select count(*) as sum4 from way_tags) as sub4,\
                (select count(*) as sum5 from way_nodes) as sub5;")
         cursor.fetchall()[0][0]

Out[55]: <sqlite3.Cursor at 0x2545e9c5110>

Out[55]: 37506549
```

This means there was a total of 37,506,549 lines of code taken from the xml file for just ways and nodes!

```
In [69]: cursor.execute("select sum1+sum2 from (select count(*) as sum1 from nodes) as sub1,\
                        (select count(*) as sum2 from ways) as sub2")
         cursor.fetchall()[0][0]

Out[69]: <sqlite3.Cursor at 0x2545e9c5110>

Out[69]: 13313487
```

There were 13313487 lines of nodes and ways particularly.

```
In [70]: cursor.execute("select count(*) from (select user from nodes group by user Union\
                        select user from ways group by user) as sub;")
         cursor.fetchall()[0][0]

Out[70]: <sqlite3.Cursor at 0x2545e9c5110>

Out[70]: 4930
```

There were 4930 unique users contributing to ways and nodes in the file.

```
In [74]: cursor.execute("select count(*) from way_tags where key LIKE 'highway';")
         cursor.fetchall()[0][0]

Out[74]: <sqlite3.Cursor at 0x2545e9c5110>

Out[74]: 253981
```

And, just as a fun fact there are 253,981 entries about highways.

# Conclusion

As can be seen above, there is a lot of information in the New York OSM XML file realeased by OpenStreetMaps.org. There are several formatting issues in just street names. Some of the output also definitely needs further inquiry though by OpenStreetMaps moderators, for example there are street names in Spanish, and some that even include full addresses which does not exactly meet the convention of the rest of the output: "340 Convent Ave, New York, NY 10031, Estados Unidos".

There is a lot more cleaning to be done in this dataset but also a lot that can be gained from analyzing it as well. Just a small sample can be seen above as user activity can be visualized, and perhaps with further details of users one can determine where most of the information is coming from, and relegate ads to places that have less users contributing