ABOUT DATA:

I have chosen Chennai as the place form my OSM project since Chennai is my home town and the data was easily available through metro extract. The following link is a download link to the data set. https://mapzen.com/data/metro-extracts please type Chennai,India in the search box for quick search

1. Problems Encountered in the Map

1. The key which has problematic character such as "=,+.%,#,$,@" was removed from the raw data.

2. The street name in the address was cleaned. There were many instant same name was used differently such as "Street" was mentioned as "street", "Strret", "street". Such different name was replaced into single one name Street. The following changes was made the expected variable has the standard name and the mapping variable has the different name converted into standard name.

```
expected = ["Street",  "Avenue",  "Road",'koyambedu','Nagar','Salai']

mapping  = { 'Ave': "Avenue",
             'NAGAR':'Nagar',
             "St":  "Street",
              'ROAD':'Road",
             "Rd":"Road",
             'Road)':"Road",
             'Road,kodambakkam':"Road",
             'Street,':"Street",
             'Strret':"Street",
             'nagar':'Nagar',
             'road': "Road",
             'street':"Street"
              }
```

1.1 Other Problem:

1.1.1 postal code:

There is a several instance where the postal code need to be clean. Some instance are there is space between every there numbers and there is a period at the end which isn't supposed to be.

The postal code before the cleaning:

```
In [45]: runfile('C:/Users/karthik/Documents/Python
Scripts/count.py', wdir='C:/Users/karthik/Documents/Python Scripts')
600 006
600 089
600 020.
600 028
```

The postal code after the cleaning:

```
file=r"C:\Users\karthik\Desktop\udacity\p3\project\chennai.osm"
count_tags(file)
600006
600089
600020
600028
```

CODE:

```
def count_tags(filename):

    for event,elem in ET.iterparse(filename,events=("start",)):
        if elem.tag=="node" or elem.tag=="way":
            for tag in elem:
                if (re.search("post",tag.attrib['k'])):
                    postcode=tag.attrib['v']
                    if postcode.isdigit():
                        pass
                    elif re.findall("\.",postcode):
                        no_dot=re.sub("\.","",postcode).replace(" ","")
                        print(no_dot)
                    else:
                        no_space=postcode.replace(" ","")
                        print (no_space)

file=r"C:\Users\karthik\Desktop\udacity\p3\project\chennai.osm"
count_tags(file)
```

1.1.2 city name:

There are several instance where the city name chennai was formatted in different format such as cHennai, CHENNAI, etc. every name which Chennai in different formant was converted into simply "Chennai". There are instance where "numbers" is city name this may be because some street name starts with number and street name may be confused with city name these instance where cleared. Other case where city name followed by "Chennai" in those case "Chennai" was removed.

Before cleaning:

```
IPython console
Console 1/A
dict_items([('West Mambalam', 2), ('8', 1), ('Saidapet', 17),
('CHENNAI', 3), ('no', 1), ('chennai-17', 1), ('Guduvancheri',
1), ('10', 1), ('Valasaravakkam', 1), ('chennai', 10), ('50+',
1), ('Mandavelipakkam', 1), ('IIT Chenai', 1), ('Chennai City
Corporation Limits', 4), ('50', 4), ('saidapet', 2), ('Mathur
Village, Oragadam', 2), ('cHennai', 1), ('Chennai, Tamilnadu',
1), ('Neelangarai,Chennai', 1), ('madipakkam, Chennai', 1),
('Chennai', 102), ('200', 1), ('t.nagar', 1), ('30', 2),
('1000', 1), ('100', 2), ('Egmore', 1), ('Guduvanchery', 1)])
```

After cleaning:

```
IPython console
Console 1/A
('Guduvanchery', 1)])
dict_items([('West Mambalam', 2), (' Chennai', 1),
('Saidapet', 17), ('t.nagar', 1), ('Guduvancheri', 1),
('Chennai', 1), ('Valasaravakkam', 1), ('chennai', 121),
('saidapet', 2), ('Mandavelipakkam', 1), ('IIT Chenai', 1), ('
Oragadam', 2), ('Egmore', 1), (' Tamilnadu', 1),
('Guduvanchery', 1)])
```

Code:

```
def city(city):
    if city.isdigit():
        pass
    elif re.search("\,",city):
        s=re.split("\,",city)
        if s!="Chennai":
            return s[1]
        else:
            return s[0]
    elif re.findall("50+",city):
        pass
    elif re.findall("no",city):
        pass
    elif re.findall("chennai",city,flags=re.I):
        chennai="Chennai"
        return chennai
    else:
        return city
```

2. Data Overview

File sizes

chennai.osm----------372 MB
chennai.osm.json-----555 MB

Number of nodes and ways:

- n_o_d=db.chennai.find({"type":"way"}).count()
- Number of ways= 78214
- n_o_n=db.chennai.find({"type":"node"}).count()
- print (n_o_n)
- Number of node=518738

Number of unique user:

```
result=(db.chennai.distinct("created.user"))
```

- a=list(result)
- print (len(a))

output:

Number of unique user=754

Top 5 users by count:

- result = db.chennai.aggregate([{ "$group": { "_id": "$created.user", "count": { "$sum": 1 } } },{"$sort":{"count":-1}},{"$limit":5}])
- pprint.pprint(list(result))

output: top five user

[{'_id': 'maheshrkm', 'count': 497844},

{'_id': 'PlaneMad', 'count': 325164},

{'_id': 'venkatkotha', 'count': 280230},

{'_id': 'shekarn', 'count': 165960},

{'_id': 'praveeng', 'count': 165843}

TOP 5 amenity by count:

- result = db.chennai.aggregate([{"$group":{ "_id":"$amenity", "count":{"$sum":1}}},{"$sort":{"count":-1}},{"$limit":5}])
- pprint.pprint(list(result))

output:

[{'_id': 'place_of_worship', 'count': 1470},

{'_id': 'school', 'count': 1407},

{'_id': 'restaurant', 'count': 864},

{'_id': 'hospital', 'count': 603},

{'_id': 'college', 'count': 531}]

Top 5 restaurant in terms of count:

- result=db.chennai.aggregate([{"$match":{"amenity":"restaurant"}},
  - {"$group":{"_id":"$name",
  - "count":{"$sum":1}}},
  - {"$sort":{"count":-1}},
  - {"$limit":20}])
- pprint.pprint(list(result))

Output:

[{'_id': 'Hotel Saravana Bhavan', 'count': 21},

{'_id': 'Hot Chips', 'count': 15},

{'_id': 'Murugan Idli Shop', 'count': 9},

{'_id': 'Saravana Bhavan', 'count': 9},

{'_id': 'Wangs Kitchen', 'count': 9}]

Top 5 Temple by count:

- result=db.chennai.aggregate([{"$match":{"amenity":" place_of_worship"}},
  - {"$group":{"_id":"$name",
  - "count":{"$sum":1}}},
  - {"$sort":{"count":-1}},
  - {"$limit":20}])
- pprint.pprint(list(result))

Output:

{'_id': 'Temple', 'count': 21},

{'_id': 'Church', 'count': 15},

{'_id': 'Perumal Koil', 'count': 12},

{'_id': 'Advent Christian Church', 'count': 12},

{'_id': 'Shirdi Sai Baba Temple', 'count': 9},

{'_id': 'Vinayagar Temple', 'count': 9},

{'_id': 'Mosque', 'count': 9},

{'_id': 'Amman Temple', 'count': 6},

{'_id': 'Pillayar Temple', 'count': 6}

Top 5 Hospital by count:

- result=db.chennai.aggregate([{"$match":{"amenity":" hospital "}},
  - {"$group":{"_id":"$name",
  - "count":{"$sum":1}}},
  - {"$sort":{"count":-1}},
  - {"$limit":20}])
- pprint.pprint(list(result))

Output:

{'_id': 'ESI Hospital', 'count': 6},

{'_id': 'ACS Medical College And Hospital', 'count': 6},

{'_id': 'SRM Hospital', 'count': 6},

{'_id': 'Vasan Eye Care Hospital', 'count': 6},

{'_id': 'Aswene Soundarya Hospital', 'count': 6}

Top 5 Banks: by count:

- result=db.chennai.aggregate([{"$match":{"amenity":" bank "}},
  - {"$group":{"_id":"$name",
  - "count":{"$sum":1}}},
  - {"$sort":{"count":-1}},
  - {"$limit":20}])
- pprint.pprint(list(result))

Output:

{'_id': 'Indian Bank', 'count': 36},

{'_id': 'Indian Overseas Bank', 'count': 33},

{'_id': 'HDFC Bank', 'count': 33},

{'_id': 'State Bank of India', 'count': 21},

{'_id': 'Karur Vysya Bank', 'count': 18},

{'_id': 'ICICI Bank', 'count': 18}

Other ideas about the datasets:

I quire the data set for new commercial buildings and amenity as I know the place very. Where I couldn't place such as new metro stations which came up few months back. There a lot of space for updating the new places into osm data. One of my method to increase the data collection is gamification. Gamification can help and motivate more people to contribute in data collection. People who collect data can be awarded points on the base on how much they contribute. There points can be converted into rewards such as gift cade, badges and etc.

Potential issues that may arise when implementation:

There are chances that only few people may contribute a large amount of data but creating the leaderboard. People can see how much each person is contributing to the data base. This may create health competition and as well encourage people contributing less to try to contribute more.

Conclusion :

After reviewing the data from Chennai the data is huge and if more coders joined together to work on the cleaning data there data can be improved. If there is system that could run the code automatically every time OSM data is updated which could make the map more accurate and pretty.

Note:

Code for the cleaning data is my problem 6. If you find problem with the format and index I'm happy to resubmit my work with gethug. All other mangodb code are mention above.

```
import xml.etree.cElementTree as ET

import pprint

import re

import codecs

import json


lower = re.compile(r'^([a-z]|_)*$')

lower_colon = re.compile(r'^([a-z]|_)*:([a-z]|_)*$')

problemchars = re.compile(r'[=\+/&<>;\'"\?%#$@\,\. \t\r\n]')


def update_postal(postcode):

    if postcode.isdigit():
```

```
        return postcode
    elif re.findall("\.",postcode):
        return postcode.replace(" ","").replace(".","")


    else:
        return postcode.replace(" ","")
def city(city):
    if city.isdigit():
        pass
    elif re.search("\,",city):
        s=re.split("\,",city)
        if s!="Chennai":
            return s[1]
        else:
            return s[0]
    elif re.findall("50+",city):
        pass
    elif re.findall("no",city):
        pass
    elif re.findall("chennai",city,flags=re.I):
        chennai="Chennai"
        return chennai
    else:
        return city





def update_street(street_name):
    street_type_re = re.compile(r'\b\S+\.?$', re.IGNORECASE)
```

```
    expected = ["Street",  "Avenue",  "Road",'koyambedu','Nagar','Salai']
    mapping  = { 'Ave': "Avenue",
                 'NAGAR':'Nagar',
                 "St":  "Street",
                 'ROAD':"Road",
                 "Rd":"Road",
                 'Road)':"Road",
                 'Road,kodambakkam':"Road",
                 'Street,':"Street",
                 'Strret':"Street",
                 'nagar':'Nagar',
                 'road': "Road",
                 'street':"Street"
                 }
    Street_type=re.search(street_type_re,street_name)

    if Street_type:
        r=Street_type.group()
        if r  in mapping.keys():
            update_street_name=re.sub(street_type_re,mapping[Street_type.group()],street_name)

        else:
            update_street_name=street_name
        return update_street_name




def shape_element(element):
    ref=[]
    address={}
```

```
    node = {}
    a=[0,0]

    CREATED = [ "version",  "changeset", "timestamp",  "user", "uid"]
    if element.tag=="node"  or element.tag == "way" :
        data=element.attrib
        node["created"]={i:data[i]  for i in CREATED}
        for key,value in data.items():
                if key=="lat":
                    lat=float(data[key])
                    a[0]=lat
                elif key=="lon":
                    lon=float(data[key])
                    a[1]=lon
                elif key not in CREATED:
                    node[key]=data[key]
        node["type"]=element.tag
        node["pos"]=a

        for child in element:
            if child.tag=="tag"  or child.tag=="nd"  :
                sub_tab=child.attrib
                for key, value in sub_tab.items():
                    if key=='k':

                        if re.search(problemchars,value):
                            continue
                        if (re.search("city",value)):
                            address["city"]=city(child.attrib['v'])
#                         if re.findall("post",value):
```

```
#                    address["postal_code"]=update_postal(child.attrib["v"])
                if re.search(lower_colon,value):
                    if value=='addr:street':
                        address["street"]=update_street(child.attrib['v'])
                    else:
                        try:
                            m=re.search("(?<=addr:)\w+",value)
                            k=m.group(0)
                            address[k]=sub_tab['v']
                        except :
                            pass
                if re.search(lower,value):
                    node[value]=sub_tab['v']
                node["address"]=address
            elif key=="ref":
                ref.append(value)
        if ref:
            node["node_refs"]=ref
        try:
            if node["address"]=={}:
                del node["address"]
        except:
            pass
        return node
    else:
        return None

def process_map(file_in, pretty = False):
    # You do not need to change this file
    file_out = "{0}.json".format(file_in)
```

```
    data = []
    with codecs.open(file_out, "w",encoding="utf8")   as fo:
        for event, element in ET.iterparse(file_in):
            el = shape_element(element)

            if el:
                data.append(el)
                #print (data)
                if pretty:
                    fo.write(json.dumps(el,  indent=2)+"\n")
                else:
                    fo.write(json.dumps(el)  + "\n")
    return data



file=r"C:\Users\karthik\Desktop\udacity\p3\project\chennai.osm"
data=process_map(file,True)
```