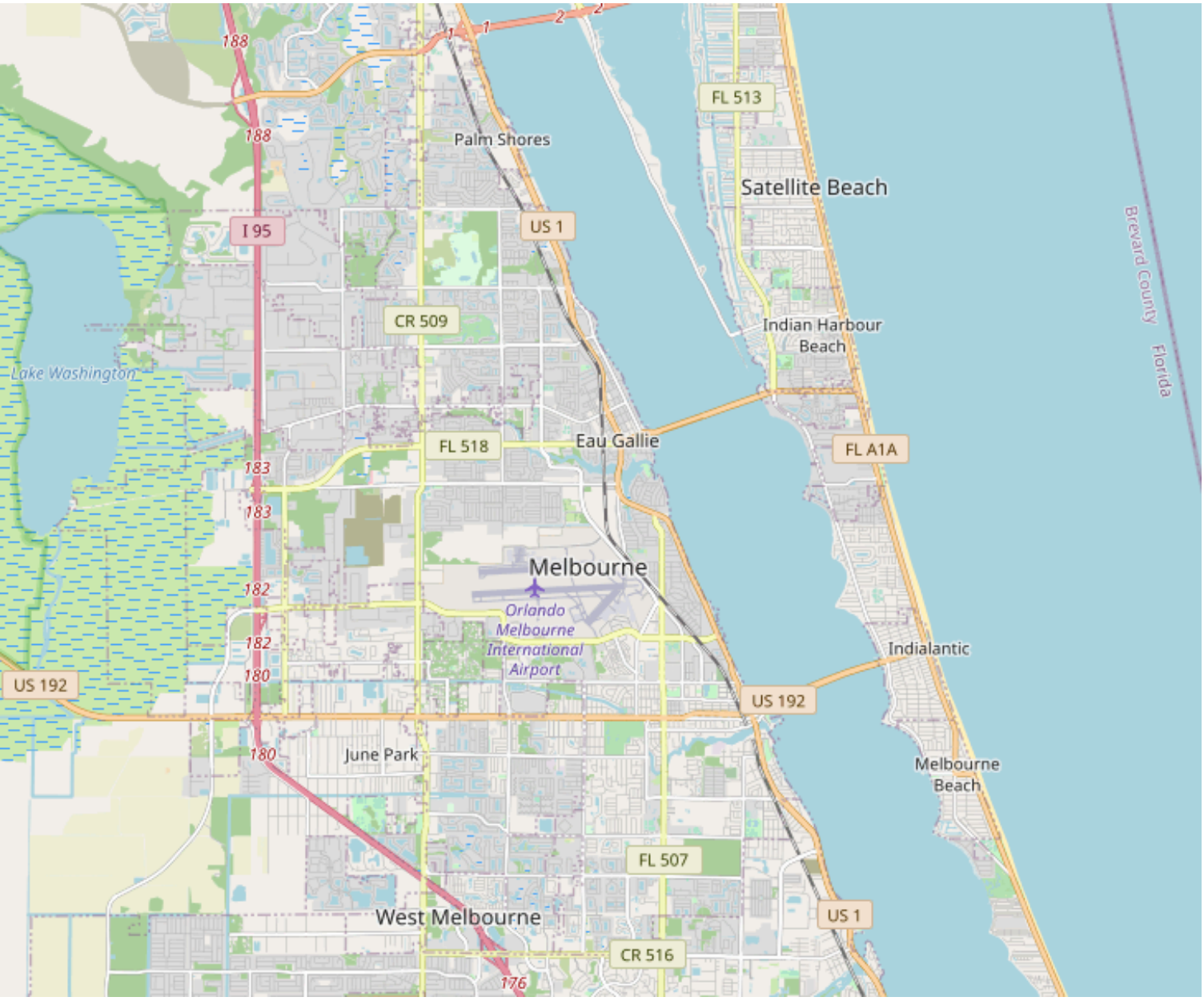
OpenStreetMap Project: Data Wrangling with MongoDB:

By: **Mohammed Bougayou**



Map Area: Melbourne, Florida

Located Melbourne, Florida from the map :
https://www.openstreetmap.org/relation/117646
OMS FIle : https://www.openstreetmap.org/export#map=12/28.1176/-80.6613

OpenStreetMap:

OpenStreetMap (**OSM**) is a collaborative project to create a free editable map of the world. The geodata underlying the map is considered the primary output of the project. The creation and growth of OSM has been motivated by restrictions on use or availability of map data across much of the world, and the advent of inexpensive portable satellite navigation devices.
#https://en.wikipedia.org/wiki/OpenStreetMap

Project Summary:

cleaning the data and normalizing the data before putting it into a database, using python: audit techniques to iterate over the XML and perform certain activities.Then write out a JSON file of the cleaned data, and import the JSON file to MongoDB Database. Finally make queries to provide statistical information and numbers related to it.

OSM Dataset:

the main reason I picked Melbourne, Florida because it's my hometown, and I was just curious to look for information related to it. And get to know more about my city using MongoDB queries.

Encounter problems:

before start and work on the OSM file, I reviews and investigate a sample set of data. Data entered is based on individuals user entries, therefore part of the Street names type and cardinal directions are abbreviated.
I had some challenges with MongoDB loading on my mac, and for some reason the DB server disconnect. anyway i had problems with my wifi so that could be the reason.

```
Central Park Dr => of Central Park Drive
Pineapple Ave => Pineapple Avenue
E Avenue D => E Avenue D => East Avenue D
```

Files Sizes:

Melbourne_florida.osm -> 134.5 MB.
Sample.json -> 213.4 MB.

I started first by counting how many tags in our XML file>

```
    #Reads XML file: sample.xml and count each XML tag within the document

    for _, elemt in ET.iterparse(inputfile):
        tag = elemt.tag
        if tag not in tags:
            tags[tag] = 1
        else:
            tags[tag] += 1
    #returns a Dictionary consist of tag_names:(counts as values)

    return tags

pprint.pprint(count_tags(inputfile))
```

```
{'bounds': 1,
 'member': 23942,
 'meta': 1,
 'nd': 668786,
 'node': 556381,
 'note': 1,
 'osm': 1,
 'relation': 569,
 'tag': 210698,
 'wa':
```

Identifying Unique Users:

```
#identifying unique users contribute the to the map area: Melbouren, FL.
"""
Your task is to explore the data a bit more.
The first task is a fun one - find out how many unique users
have contributed to the map in this particular area!

The function process_map should return a set of unique user IDs ("uid")
"""

def process_map(inputfile):
    users = set()
    for _, element in ET.iterparse(inputfile):
        for e in element:
            if 'uid' in e.attrib:
                users.add(e.attrib['uid'])

    return users
#the function process_map return a set of unique user "uid"
users = process_map(inputfile)
print('Number of unique users:', len(users))
```

```
Number of unique users: 797
```

Once I have an idea of the data set, starting up by cleaning up. from dictionary that we build and using regex to search.

```
#https://docs.python.org/3/howto/regex.html
#Scan through a string, looking for any location where this RE matches.
def update_name(name, mapping, regex):
    m = regex.search(name)
    if m:
        street_type = m.group()
        if street_type in mapping:
            name = re.sub(regex, mapping[street_type], name)

    return name
```

```
street_type_mapping = {'Ave'  : 'Avenue',
                       'Blvd' : 'Boulevard',
                       'Dr'   : 'Drive',
                       'Ln'   : 'Lane',
                       'Pkwy' : 'Parkway',
                       'Rd'   : 'Road',
                       'St'   : 'Street'}
```

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

```
for street_type, ways in street_types.items():
    for name in ways:
        better_name = update_name(name, street_type_mapping, street_type_re)
        print(name, "=>", better_name)
```

Data queries:

Number of documents:
>db = client["openstreetmap"]
melbourne_florida = db["melbourne_florida"]
#Number of documents
#https://classroom.udacity.com/courses/ud032/lessons/745498943/concepts/7347306510923
print(melbourne_florida.find().count())

————————621451—————————

GovernmentType:
>I want to find document that do have a govermentType field: I was surprised that they were None. result is 0.

#https://classroom.udacity.com/courses/ud032/lessons/745498943/concepts/7347306470923
#i want to find document that do have a govermentType field.
db.melbourne_florida.find({"governmentType" : {"$exists" : 1}}).count()

Unique users that contribute to the data:
>len(melbourne_florida.distinct('created.user'))
————778————————

Number of nodes:
>db.melbourne_florida.find({“type”:{"$in":['node']}}).count()
————556381————

Number os ways:
>db.melbourne_florida.find({“type”:{"$in":['way']}}).count()
————-65070————

List of the main contributors to our dataset:

>cursor= db.melbourne_florida.aggregate([{"$group":{"_id":"$created.user", "count":{"$sum":1}}}, {"$sort":{"count":-1}}, {"$limit":10}])

for document in cursor:

    print(document)

```
{'_id': 'Ian Swire', 'count': 111093}
{'_id': 'floridaeditor', 'count': 66009}
{'_id': 'Panther37', 'count': 62191}
{'_id': 'NE2', 'count': 61076}
{'_id': 'chachafish', 'count': 32837}
{'_id': 'mahahahaneapneap', 'count': 25559}
{'_id': 'LeTopographeFou', 'count': 22796}
{'_id': 'grouper', 'count': 21930}
{'_id': 'Easky30', 'count': 19119}
{'_id': 'GlittrGrl', 'count': 12856}
```

Now i will querying and get information from the data regarding amenities and count, also want to know how many bars are in town.

#count of bars in melbourne based of the match.

```
results = db.melbourne_florida.aggregate([{'$match': {'amenity': 'bar'}}])
```

Very surprised that there is only 12 bars, I thought lot more than that.

#count for different types of amenities. only the top 10.

```
results = db.melbourne_florida.aggregate([{"$group":{"_id":"$amenity", "count":
{"$sum":1}}}, {"$sort":{"count": -1}},{'$limit': 11}])
for amenity in results:
    print(amenity)
```

```
{'_id': None, 'count': 619100}
{'_id': 'parking', 'count': 893}
{'_id': 'restaurant', 'count': 186}
{'_id': 'fountain', 'count': 150}
{'_id': 'fast_food', 'count': 142}
{'_id': 'place_of_worship', 'count': 131}
{'_id': 'school', 'count': 110}
{'_id': 'fuel', 'count': 96}
{'_id': 'shelter', 'count': 72}
{'_id': 'bank', 'count': 50}
```

very interesting to collect this returns, and how you can use it in your favor. we can say that there is more restaurant than fast food places.

Conclusion :

lot information that can we produced from the dataset and helpful information that can we used. I found MongoDb are lot simpler and easier to utilize. Accuracy of the data is very crucial once you have it normalized and cleaned, lot information is added by users and sometimes needs more analyzing and investigation. This project is an eye opening to the future and how we can use data in our favor.