Open Street Map Project

Map Area

Rome, Italy

Source https://mapzen.com/data/metro-extracts/metro/rome_italy/

For this study I will use the OSM XML file from Mapzen of my hometown, Rome. I am curious to perform some queries on this area. Before import the data into an SQL database , I will try to clean a bit the informations contained into this file, hoping to contribute at the openstreetmap database.

Encountered Problems

While looking into the file i will focus my attention to postal code and street's name. This is the typical issues that I have found using the clean.py code I wrote for.

Postal Code's Problems

- Wrong postal code format entered , or integer in state of string ['00184 Roma' , I-00128, 159' in state of '00159']
- Missing of a correct value ['Roma' , '00144;00159', '001963']
- Incoherent values ['84010' , '00014' , '15057']

Streets Name's Problems

In Italian language the name of the street is at the end , while it begins with the kind of street (Via, Vicolo, Piazza , etc...). While looking in all the data using the clean.py code , I found multiple typo and some missing values:

- Wrong spelling or typo ('VIa', 'Vi', 'Vía', that should all be 'Via')
- Abbreviation ('P.za' instate of Piazza)
- Missing values ('Alberto Caldolo' in state of ' Via Alberto Caldolo')

Cleaning

I preferred to clean it before exporting to the csv file. I use a function like this below to modify the postal code and the street address, all the code is in clean.py file. I work these two values error in different ways, while for postal code I write a function to replace the values if wrong with the good one. For streets, like the example code below, I wrote a functions that splits all the word contained into a list and replace the first one with the dictionary value , then joins the new list and create a new string. With the clean.py code , I export then 5 Csv files that I will import into the SQL database.

```
def road(item,correctroad):
    local = item.get('v')
    while True:
```

```
        try:
            diviso = local.split(" ")
            first = diviso[0]
            diviso[0] = correctroad[first]
            newlocal = " ".join(diviso)
            break
        except KeyError:
            newlocal = local
            break
    return
```

SQL DataBase

Postal Code

I will now use the sql database to verify the postal code, I will look for value where key is equal to postcode or postal_code. Supposing that errors are rare Is will look for the 15 less commun results:

```
SELECT tags.value, COUNT(*) as num
FROM (SELECT * FROM nodes_tags
      UNION ALL
      SELECT * FROM ways_tags) tags
WHERE tags.key = 'postcode'
OR tags.key = 'postal_code'
GROUP BY tags.value
ORDER BY num
LIMIT 15;

00014,1
00039,1
00063,1
00072,1
00078,1
00115,1
00144;00159,1
001963,1
0747,1
2378,1
82011,1
84010,1
Roma,1
00055,2
00057,2
```

Zip codes in Rome are from 00118 to 00199, but there are a lot of small city all around that are contained in the area, and that are actually part of Rome's Province , that begin with 00010 to 00079. So most of these results are actually correct.
But there are a couple of other issues, first there are postal codes that do not really belong to this area and are postal code from an other region, like 84010 and 82011, and than there are some errors, like '00144;00159' or '001963'. Both problems should be handle one by one, as long they are, like in this example, not many. The code used to fix these issues are not reported here for

presenting propose. For more information please refer to the sqlqueries.sql file included in this repository.

Higher frequency cities

I would like to look at postal code in the other way around and discover which one's have the highest frequency.

```
SELECT tags.value, COUNT(*) as num
FROM (SELECT * FROM nodes_tags
      UNION ALL
      SELECT * FROM ways_tags) tags
WHERE tags.key = 'postcode'
OR tags.key = 'postal_code'
GROUP BY tags.value
ORDER BY num DESC
LIMIT 20;

00185,647
00159,330
00122,288
00144,256
00156,236
00044,171
00186,170
00153,152
00192,148
00176,141
00169,139
00136,132
00173,118
00184,114
00161,108
00040,107
00187,101
00155,95
00174,93
00121,78
```

This results show us that most of the postal code are from the city center of Rome, but we have also in the top 20 the '00040' postal code that is 'Rocca di Papa'. I would like to know the city with the highest frequency.

```
SELECT tags.value, COUNT(*) as count
FROM (SELECT * FROM nodes_tags UNION ALL
      SELECT * FROM ways_tags) tags
WHERE tags.key == 'city'
GROUP BY tags.value
ORDER BY count DESC
LIMIT 15;

Roma,8844
```

```
Frascati,159
Velletri,75
Bracciano,57
Fiumicino,34
San Cesareo,33
Pomezia,26
Tivoli,25
Ciampino,22
Fregene,22
Nemi,17
Rocca di Papa,17
Ariccia,16
Formello,16
Albano Laziale,14
```

As we can see all these cities are in Rome Area but we can see that while before we saw 107 times the postal code of "Rocca di Papa", we have only 17. So it means that we have more often the name of the city in state of the postal code for tags.

Overview Statistics

Size

File Name	Type	Size
rome italy.osm	OSM XML	497.9 Mb
rome italy osm.db	SQLITE DB	260.8 Mb
nodes.csv	CSV	185 Mb
nodes_tags.csv	CSV	7.4 Mb
ways.csv	CSV	20.1 Mb
ways_nodes.csv	CSV	65.5 Mb
ways_tags.csv	CSV	23 Mb

Number of Unique Users

```
SELECT COUNT(DISTINCT(users.uid)) as count
FROM (SELECT uid FROM ways UNION ALL SELECT uid FROM nodes) users;

2438
```

Numbers of Nodes and Ways

```
SELECT COUNT(*) as count
FROM nodes;
2226142

SELECT COUNT(*) as count
FROM ways;
331838
```

Numbers of users that appear just one time

```
SELECT COUNT(*)
FROM (SELECT users.user,COUNT(*) as count
       FROM (SELECT user FROM ways UNION ALL SELECT user FROM nodes) users
       GROUP BY users.user
       HAVING count = 1) zero;

660
```

Almost 25% of the total users have contributed just once to this dataset.

More deeper on Amenities

Top 20 Amenities

```
SELECT value, COUNT(*) as count
FROM nodes_tags
WHERE key='amenity'
GROUP BY value
ORDER BY count DESC
LIMIT 20;

drinking_water|1871
restaurant|1633
cafe|855
bench|749
bar|720
fuel|680
parking|446
pharmacy|421
fast_food|406
bank|366
school|200
fountain|192
recycling|174
post_office|161
bicycle_parking|156
telephone|153
waste_basket|145
toilets|144
parking_entrance|140
place_of_worship|138
```

As we can see there are much more drinking water point than place of worship. I know very well Rome and I know that we can drink water everywhere but I wouldn't expected this kind of results. Rome normally has more than 900 churches, so I think that we miss something.
It very interesting also notice that for this data there are still 153 telephone, and in 2017, in the smartphone era, I wouldn't expect so.

Kind of cuisine

I will now take a deeper look to the kind of cuisine for the amenity restaurant.

```
SELECT nodes_tags.value, COUNT(*) as count
FROM nodes_tags ,(SELECT DISTINCT(id) FROM nodes_tags WHERE value='restaurant') as
restaurant
WHERE nodes_tags.id=restaurant.id
AND nodes_tags.key='cuisine'
GROUP BY nodes_tags.value
ORDER BY count DESC
LIMIT 5;

italian|346
pizza|172
regional|102
chinese|43
japanese|17
```

Tourists come to Italy to eat Italian cuisine, and Roman are not so open to foreign cuisine, and this results seems to confirm it.

Further Ideas

There are more and more smartphone application that use your position to propose a service around you, for example deliveroo or Foodora. This service in the first instance use the geo-localisation, then before confirming your order, ask you to verify the address and confirm it. So this data could be use to confirm the exact location of a node, using latitude and longitude and then update or create the informations of the building.

Benefits	Anticipated Problems
Easy to fetch nodes with the data	Privacy matter and customers would not like to participate
Each user could update his hime address, but also his work place or his friends house	Node oriented, this will not update information about public place

Final Conclusion

In general this file seems complete, It has some compiling error, but for postal code it was possible to correct almost of it, while for all the street names It can be possible to still have issue.
The file contains non only Rome, but all the area around the city.
When I looked for amenities, I found only less than 200 place of Worship, while in Rome there are more than 900 Cattolic's Church plus all the other religion. So I guess the file it can not be considered complete. And further improvement will mean add this important nodes.
I think that with the help of clean.py, the database will contain less typo errors, but sure there are also singular and specific mistake that need more dedicated attention.