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

Street Name' 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 these below to modify the postal code and the street address, all the code is in the clean.py file. Correctroad is a dictionary that I have created where each wrong case is the key and the values is the corrected one. Wrongzip in state is a dictionary of only the wrong zip code corrected. 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, I wrote a functions that splits all the word contained in a list and replace the first one with the dictionary value , joins the new list and create a new string. In order to rectify the street errors, I use two function one in the case for nodes, the other one specifically for ways nodes, because they do not have a key "addr:street" but " name". So not all the name are streets.

While using these functions and the rest in clean.py I create 5 Csv files that I will now import into the SQL database.

```
def zipcode(item,wrongzip):
    postalcode = item.get('v')
    if postalcode in wrongzip:
        newcode = wrongzip[postalcode]
    else:
        newcode= postalcode
    return newcode

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

def roadname(item,correctroad):
    local = item.get('v')
    while True:
        try:
            diviso = local.split(" ")
            first = diviso[0]
            if first in correctroad:
                diviso[0] = correctroad[first]
                newlocal = " ".join(diviso)
                break
            else:
                newlocal = local
                break
        except KeyError:
            newlocal = local
            break
    return newlocal
```

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 i will look for the 20 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 20;
```

```
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
00074,2
00077,2
15057,2
00036,3
00052,3
```

Zip code in Rome are from 00118 to 00199, but there are a lot of small city all around that are contained in the area, and tat 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 code 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 should be handle one by one, like in the following cases.

```
# Investigating the '001963 value , so i look for the ways id

SELECT id
FROM ways_tags
WHERE key='postcode'
AND value = '001963';

429260650

SELECT * FROM ways_tags WHERE id=429260650 and type='addr';

429260650,city,Roma,addr
429260650,street,"Via Ennio Quirino Visconti",addr
429260650,postcode,001963,addr
429260650,housenumber,13,addrCT * FROM ways_tags WHERE id=429260650 and type='addr';

# Via Ennio Quirino Visconti in Roma has a postal code of '00193' so this example wo
# With this code the correct postal code is now releted

UPDATE ways_tags
SET value = '00193'
WHERE id ='429260650'
AND key = 'postcode';

SELECT * FROM ways_tags WHERE id=429260650 and type='addr';

429260650|city|Roma|addr
429260650|street|Via Ennio Quirino Visconti|addr
429260650|postcode|00193|addr
429260650|housenumber|13|addr
```

So we have found some minor error in postal code that can be handled one by one , but fortunately they do not seem to be to many. Note that this last code have to be modify to correct error and especially postal code can be on tags tables of ways as well as nodes.

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 20;
```

```
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
Anguillara Sabazia,14
Grottaferrata,14
Torvaianica,14
Manziana,11
Marino,10
```

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
```

Most active users (Top 15)

```
SELECT users.user ,  COUNT(*) as count
FROM (SELECT user FROM ways UNION ALL SELECT user FROM nodes) users
GROUP BY users.user
ORDER BY count DESC
LIMIT 15;

Davlak|324439
Michele Aquilani|258469
marcoSt|210525
vlattanzi|180955
dieterdreist|149349
Emistrac|115334
mcheckimport|109712
pelatom|87357
gnastyle|52785
ALn_668|48128
Mathbau|39392
GPS-Marco|37858
Dario Orlovich|37803
Paolo Gianfrancesco|35428
procuste|34823
```

Numbers of users that appeare 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 appeared just once in 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 Rome and i know that we can drink water everywhere but I would expected this kind of results. Rome normally has more than 900 churchs, so i guess something is missing from the file.
It very interesting also notice that for this data there are still 153 telephone, and in 2017, with 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, bar and cafe.

```
SELECT nodes_tags.value, COUNT(*) as count
FROM nodes_tags ,(SELECT DISTINCT(id) FROM nodes_tags WHERE value='restaurant') as r
WHERE nodes_tags.id=restaurant.id
AND nodes_tags.key='cuisine'
GROUP BY nodes_tags.value
ORDER BY count DESC
LIMIT 10;
```

```
italian|346
pizza|172
regional|102
chinese|43
japanese|17
italian_pizza|13
burger|12
italian;pizza|9
asian|8
fish|7

SELECT nodes_tags.value, COUNT(*) as count
FROM nodes_tags ,(SELECT DISTINCT(id) FROM nodes_tags WHERE value='bar') as restaurc
WHERE nodes_tags.id=restaurant.id
AND nodes_tags.key='cuisine'
GROUP BY nodes_tags.value
ORDER BY count DESC
LIMIT 10;

italian|8
ice_cream|3
pizza|2
regional|2
bar|1
burger;breakfast;grill;coffee_shop;friture;pasta;italian;pizza;chicken|1
coffee_shop|1
coffee_shop;breakfast|1
italian;breakfast;donut;tea;coffee_shop|1
italian_pizza;italian|1

SELECT nodes_tags.value, COUNT(*) as count
FROM nodes_tags ,(SELECT DISTINCT(id) FROM nodes_tags WHERE value='cafe') as restaur
WHERE nodes_tags.id=restaurant.id
AND nodes_tags.key='cuisine'
GROUP BY nodes_tags.value
ORDER BY count DESC
LIMIT 10;

italian|23
ice_cream|20
coffee_shop|11
regional|4
sandwich|2
Gelateria|1
american;bagel;breakfast;cake;sandwich|1
bar|1
bistro|1
```

```
bistrot|1
```

It seems that in all 3 categories the most common cuisine is Italian. As foreign cuisine in Rome seems to be difficulties to find other then asian.

Final Conclusion and further ideas

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.