OpenStreetMap Project

Udacity Data Analyst Nanodegree Project 3: Data Wrangling with MongoDB

Florina Georgescu - Airbus Operations SL

Github: https://github.com/inageorgescu (https://github.com/inageorgescu)

Map Area: Málaga, Spain

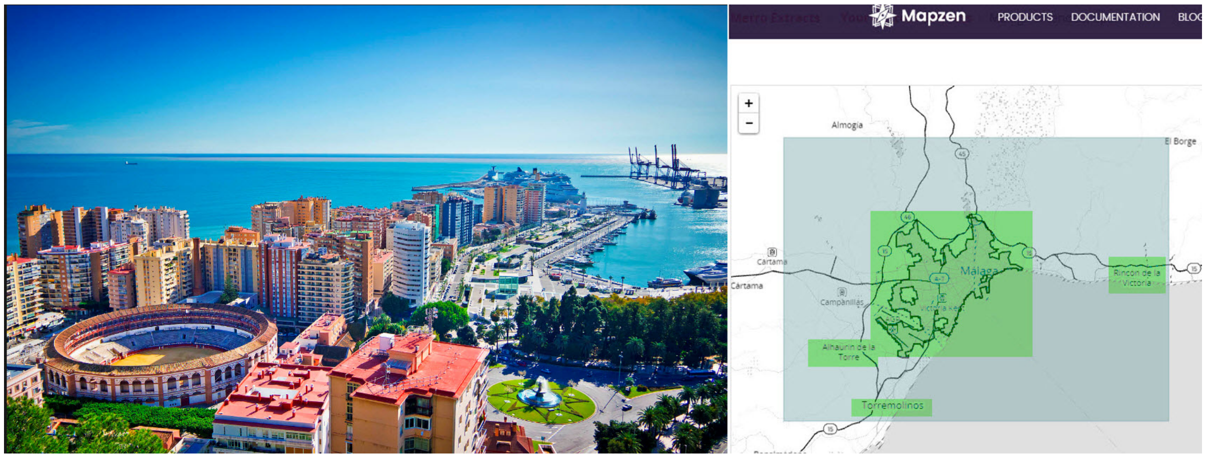
https://www.openstreetmap.org/relation/340746 (https://www.openstreetmap.org/relation/340746)

http://www.malaga.eu/ (http://www.malaga.eu/)

I will analize Málaga and 3 nearby cities(Rincon de la Victoria, Alhaurin de la Torre and Torremolinos) that are located in the Province of Málaga. This is the place where I got married and I usually visit in summer time. I will use data wrangling techniques and querry the data base with MongoDB. The data has been downloaded with Mapzen.

```
In [1]: from IPython.display import Image
        Image("Malaga_map.jpg")
```

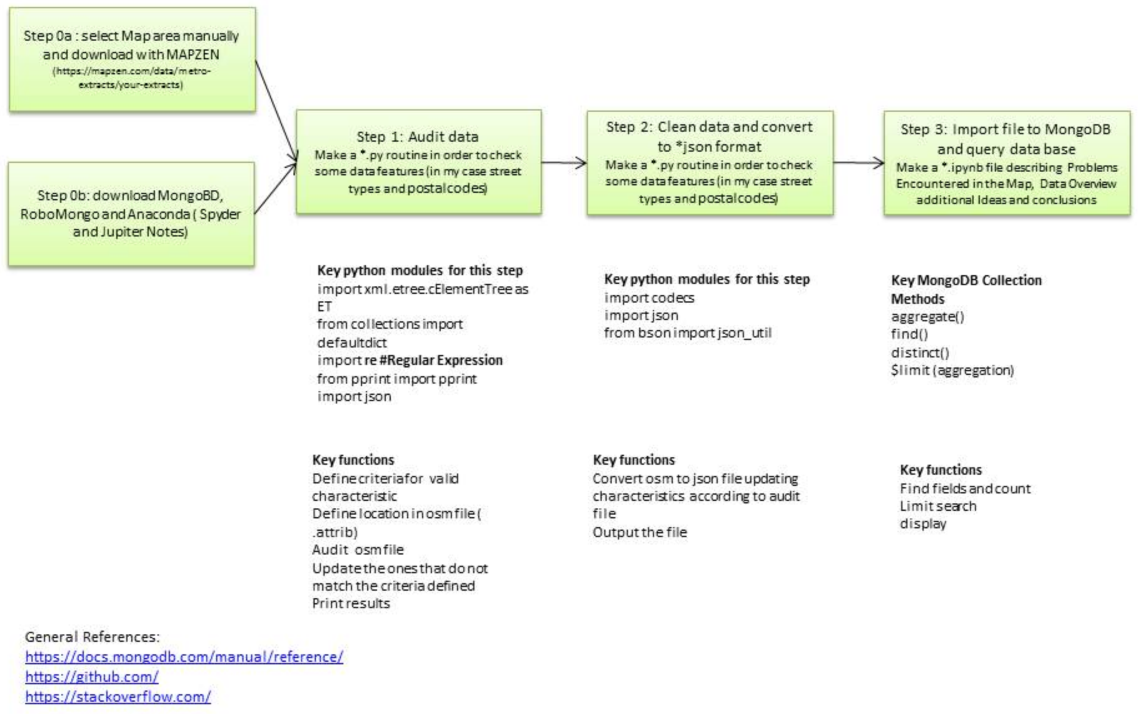
Out[1]:



In order to complete the project the following steps shown in the diagram below must be followed

In [2]:
```
Image("workflow.jpg")
```

Out[2]:


Step 0a : select Map area manually and download with MAPZEN (https://mapzen.com/data/metro-extracts/your-extracts)
Step 0b: download MongoBD, RoboMongo and Anaconda (Spyder and Jupiter Notes)
Step 1: Audit data Make a *.py routine in order to check some data features (in my case street types and postal codes)
Step 2: Clean data and convert to *json format Make a *.py routine in order to check some data features (in my case street types and postal codes)
Step 3: Import file to MongoDB and query data base Make a *.ipynb file describing Problems Encountered in the Map, Data Overview additional Ideas and conclusions
Key python modules for this step
import xml.etree.cElementTree as ET
from collections import defaultdict
import re #Regular Expression
from pprint import pprint
import json
Key python modules for this step
import codecs
import json
from bson import json_util
Key MongoDB Collection Methods
aggregate()
find()
distinct()
$limit (aggregation)
Key functions
Define criteria for valid characteristic
Define location in osm file (.attrib)
Audit osm file
Update the ones that do not match the criteria defined
Print results
Key functions
Convert osm to json file updating characteristics according to audit file
Output the file
Key functions
Find fields and count
Limit search
display
General References:
https://docs.mongodb.com/manual/reference/
https://github.com/
https://stackoverflow.com/

In [3]:
```
#Import all necesary modules as follows:
#import flexible container object, designed to store hierarchical data structu
res in memory
import xml.etree.cElementTree as ET
#import function to supply missing values
from collections import defaultdict
#import Regular expression operations
import re
#import “pretty-print” arbitrary Python data structures in a form which can be
 used as input to the interpreter
import pprint
#import Codec registry and base classes
import codecs
#import JSON encoder and decoder
import json
import pymongo
import os
```

In [4]:
```
#Create a sample file in order to perform the tests and be able to visualize e
asily

OSM_FILE = "Malaga.osm"  # OSM file for Malaga
SAMPLE_FILE = "sample.osm"

k = 10 # Parameter: take every k-th top level element

def get_element(osm_file, tags=('node', 'way', 'relation')):
    """Yield element if it is the right type of tag

    Reference:
    http://stackoverflow.com/questions/3095434/inserting-newlines-in-xml-file-
generated-via-xml-etree-elementtree-in-python
    """
    context = iter(ET.iterparse(osm_file, events=('start', 'end')))
    _, root = next(context)
    for event, elem in context:
        if event == 'end' and elem.tag in tags:
            yield elem
            root.clear()


with open(SAMPLE_FILE, 'wb') as output:
    output.write('<?xml version="1.0" encoding="UTF-8"?>\n')
    output.write('<osm>\n  ')

    # Write every kth top level element
    for i, element in enumerate(get_element(OSM_FILE)):
        if i % k == 0:
            output.write(ET.tostring(element, encoding='utf-8'))

    output.write('</osm>')
```

1. Audit data

The data that I will audit is a OSM file for a given map, in this case Málaga. OSM XML is list of instances of data (nodes, ways, and relations) that represent points on the map. Ways contain node references to form either a polyline or polygon on the map. Nodes and ways both contain children tag elements that represent key value pairs of descriptive information about a given node or way.

Problems Encountered in the Map

After using the a sample file of the Malaga area and running it against a provisional audit_atributes.py file, I noticed three main problems with the data, which I will tackle:

1.1. Street type

In Europe the street type is written after the name of the street and must have a standard naming. It is common to use abreviation for it therefore you can have same street type called different ways e.g. "Avenida" can be founf as "AV". "AVDA" etc. This problem was programmatically solved by function update_names() before creating the database.

1.2. Street without a type of street

There are streets without a street type. This probelm can be solved only if you have a list with street names and their type. It is not our case therefore I will only list (with the function audit()) the streets that meet this condition in order to identify that there is a problem in the map.

1.3 Postal codes

There are some postal codes that might have not been corectly introduced because they do not belong to the valid postal codes in the area or they are wornlgly introduced. This problem was programmatically solved by function update_postal_codes() before creating the database.

In [5]:
```
#Parse file and count number of unique element types
def count_tags(filename):
        tags = {}
        for event, elem in ET.iterparse(filename):
            if elem.tag in tags:
                tags[elem.tag] += 1
            else:
                tags[elem.tag] = 1
        return tags
view_tags = count_tags(OSM_FILE)
pprint.pprint(view_tags)
```

```
{'bounds': 1,
 'member': 11951,
 'nd': 432582,
 'node': 326246,
 'osm': 1,
 'relation': 1224,
 'tag': 166419,
 'way': 48186}
```

In [6]:
```
from IPython.display import display
from audit_atributes import audit, update_names, audit_pc, update_postal_codes

OSM_FILE ='./Malaga.osm'
CREATED = [ "version", "changeset", "timestamp", "user", "uid"]
street_types = audit(OSM_FILE)
display(street_types)
```

```
defaultdict(set,
            {'AU': {'AU AP-7  224,00  I'},
             'AV': {'AV DE VELAZQUEZ  104', 'AV VALLE INCLAN, S/N'},
             u'AVD.': {u'AVD. DE VELAZQUEZ N\xba113'},
             'AVDA': {'AVDA MANUEL FRAGA IRIBARNE'},
             'Alozaina': {'Alozaina'},
             u'Arroyo': {u'Arroyo de los \xc1ngeles'},
             'Atarazanas': {'Atarazanas'},
             u'Autov\xeda': {u'Autov\xeda A7 M\xe1laga - Algeciras'},
             u'Av.': {'Av. Comandante Benitez',
              u'Av. Juan Sebasti\xe1n Elcano'},
             'Avd': {'Avd Postas'},
             u'Avda.': {u'Avda. Juan Sebasti\xe1n Elcano'},
             'Bela': {'Bela Bartok'},
             'C/': {'C/ Decano Higueras del Castillo', 'C/ Pepe Tobalo, 1'},
             'C/La': {'C/La Hoz'},
             'CC': {'CC MA-401 KM 9.158'},
             'CL': {'CL AVENIDA DE MALAGA, S.N.',
              'CL AYALA, SN',
              'CL CONCEPCION ARENAL ,S.N. (ESQ.CR',
              'CL HERMANOS LUMIERE, SN',
              'CL PZ. CRUZ HUMILLADERO'},
             'CR': {'CR N-348, M.D. RAMAL AEROPUERTO'},
             u'Cari\xf1ena': {u'Cari\xf1ena'},
             'Carreteria': {'Carreteria'},
             'Carril': {'Carril de Gamarra', 'Carril de los toros'},
             'Conde': {'Conde de las Navas'},
             'Cruz': {'Cruz del Humilladero'},
             u'Diego': {u'Diego de Silo\xe9'},
             'Dolores': {'Dolores Cassini'},
             u'Finca': {u'Finca La Lira. Ctra. C\xe1rtama, km 12'},
             u'Fiscal': {u'Fiscal Luis Portero Garc\xeda'},
             'Franz': {'Franz Kafka'},
             'Granados.': {'Granados.'},
             'Gustavo': {'Gustavo Pitaluga'},
             'Hermes,': {'Hermes, 7'},
             'Juan': {'Juan Sebastian Elcano', u'Juan Sebasti\xe1n Elcano'},
             'Jumilla': {'Jumilla'},
             'Luxemburgo': {'Luxemburgo'},
             'Manuel': {'Manuel Altolaguirre', 'Manuel Gorria'},
             'Maria': {'Maria Barrabino'},
             u'Mar\xeda': {u'Mar\xeda de la Cruz'},
             'Molina': {'Molina Lario'},
             'N-340': {'N-340 KM 244.000'},
             u'Obispo': {u'Obispo Bartolom\xe9 Espejo'},
             'PROSPER': {'PROSPER MERIMEE'},
             'Palmeral': {'Palmeral de las Sorpresas'},
             'Pasillo': {'Pasillo Atocha', 'Pasillo de Santa Isabel'},
             'Poeta': {'Poeta Manuel Alcantara'},
             'Portales': {'Portales de la Victoria'},
             'Presbitero': {'Presbitero Carrasco Panal'},
             'Puerto': {'Puerto de Malaga'},
             'Ramal': {'Ramal del Carmelo'},
             'Realenga': {'Realenga de San Luis'},
             u'Reino': {u'Reino de Le\xf3n'},
             u'Rep\xfablica': {u'Rep\xfablica Argentina'},
             'Ribeiro': {'Ribeiro'},
             'Rio': {'Rio Bergantes 12, local 4'},
             u'San': {u'San Andr\xe9s', 'San Juan Bosco'},
             'Sin': {'Sin nombre'},
             'Tres': {'Tres Cruces'},
             'Urb.': {'Urb. La Torre'},
             'teruel': {'teruel 14'},
             'zoSan': {'zoSan Loren'}})
```

In [27]:

```
#Update street type
updates = update_names(OSM_FILE)
display(updates)
```

```
{'AV DE VELAZQUEZ  104': 'Avenida DE VELAZQUEZ  104',
 'AV VALLE INCLAN, S/N': 'Avenida VALLE INCLAN, S/N',
 'AVDA MANUEL FRAGA IRIBARNE': 'Avenida MANUEL FRAGA IRIBARNE',
 'Av. Comandante Benitez': 'Avenida Comandante Benitez',
 u'Av. Juan Sebasti\xe1n Elcano': 'Avenida Juan Sebasti\xc3\xa1n Elcano',
 u'Avda. Juan Sebasti\xe1n Elcano': 'Avenida Juan Sebasti\xc3\xa1n Elcano',
 'C/ Decano Higueras del Castillo': 'Calle Decano Higueras del Castillo',
 'C/ Pepe Tobalo, 1': 'Calle Pepe Tobalo, 1',
 'CL AVENIDA DE MALAGA, S.N.': 'Calle AVENIDA DE MALAGA, S.N.',
 'CL AYALA, SN': 'Calle AYALA, SN',
 'CL CONCEPCION ARENAL ,S.N. (ESQ.CR': 'Calle CONCEPCION ARENAL ,S.N. (ESQ.C
R',
 'CL HERMANOS LUMIERE, SN': 'Calle HERMANOS LUMIERE, SN',
 'CL PZ. CRUZ HUMILLADERO': 'Calle PZ. CRUZ HUMILLADERO'}
```

In [8]:

```
updates_pc = update_postal_codes(OSM_FILE)
display(updates_pc)
```

```
{'24190': '29001',
 '29.730': '29001',
 '2910': '29001',
 '29130': '29001',
 '29196': '29001',
 '29260': '29001',
 '29620': '29001',
 '29720': '29001',
 '29730': '29001',
 '29738': '29001'}
```

2. Data Overview with MongoDB

In [9]:

```
import os
print "The downloaded OSM file is {}
MB".format(os.path.getsize('Malaga.osm')/1.0e6) # convert from bytes to megaby
tes
```

```
The downloaded OSM file is 73.625084 MB
```

In [10]:
```
from pymongo import MongoClient

db_name = 'openstreetmap1'

# Connect to Mongo DB
client = MongoClient('localhost:27017')
# Database 'openstreetmap' will be created
db = client[db_name]
collection = db['Malaga']
```

There is a code for running directly from python the mongoimport (see https://github.com/bestkao/data-wrangling-with-openstreetmap-and-mongodb (https://github.com/bestkao/data-wrangling-with-openstreetmap-and-mongodb)) but as I do not have administrator right for my computer I had to manually instroduce the command In order to create the data base the for the openstreet execute in console cmd the following code: mongoimport -h 127.0.0.1:27017 --db openstreetmap1 --collection Malaga --file C:\Users\c41237\Desktop\Udacity\P3\P3_OpenStreetMap\Malaga.osm.json

In [11]:
```
# Number of documents
documents = collection.find().count()
display(documents)
```

```
374432
```

In [12]:
```
#Number of unique users
users=len(collection.distinct('created.user'))
display(users)
```

```
614
```

In [13]:
```
#number of nodes & ways
nodes=collection.find({'type':'node'}).count()
ways=collection.find({'type':'way'}).count()
display(nodes)
display(ways)
```

```
326202

48175
```

In [14]:
```
#disply types and number of nodes & ways
node_way = collection.aggregate([
        {"$group" : {"_id" : "$type", "count" : {"$sum" : 1}}}])

pprint.pprint(list(node_way))
```

```
[{u'_id': u'gas', u'count': 1},
 {u'_id': u'oil', u'count': 3},
 {u'_id': u'floating dock', u'count': 1},
 {u'_id': u'unspecified', u'count': 5},
 {u'_id': u'multipolygon', u'count': 1},
 {u'_id': u'way', u'count': 48175},
 {u'_id': u'plate', u'count': 2},
 {u'_id': u'palm', u'count': 42},
 {u'_id': u'node', u'count': 326202}]
```

In [15]:
```
#top3 contributors to the map
top3 = collection.aggregate([{ '$group' : {'_id' : '$created.user',
                                           'count' : { '$sum' : 1}}},
                             { '$sort' : {'count' : -1}},
                             { '$limit' : 3 }])
display(list(top3))
```

```
[{u'_id': u'dcapillae', u'count': 74082},
 {u'_id': u'CPrados', u'count': 53828},
 {u'_id': u'L\xfcbeck', u'count': 41827}]
```

In [16]:
```
#number of documents with street addresses
addresses=collection.find({'address.street': {'$exists': 1}}).count()
print(addresses)
```

```
1233
```

In [17]:
```
#top10 postal codes
top10_pc = collection.aggregate([{ '$group' : {'_id' : '$address.postcode',
                                           'count' : { '$sum' : 1}}},
                             { '$sort' : {'count' : -1}},
                             { '$limit' : 3 }])
display(list(top10_pc))
```

```
[{u'_id': None, u'count': 373914},
 {u'_id': u'29001', u'count': 255},
 {u'_id': u'29018', u'count': 44}]
```

In [18]:
```
#list of postal codes
postal_codes = collection.aggregate([
        {"$match" : {"address.postcode" : {"$exists" : 1}}}, \
        {"$group" : {"_id" : "$address.postcode", "count" : {"$sum" : 1}}}, \
        {"$sort" : {"count" : -1}}])

pprint.pprint(list(postal_codes))
```

```
[{u'_id': u'29001', u'count': 255},
 {u'_id': u'29018', u'count': 44},
 {u'_id': u'29004', u'count': 38},
 {u'_id': u'29002', u'count': 32},
 {u'_id': u'29016', u'count': 21},
 {u'_id': u'29014', u'count': 15},
 {u'_id': u'29017', u'count': 15},
 {u'_id': u'29007', u'count': 13},
 {u'_id': u'29006', u'count': 12},
 {u'_id': u'29010', u'count': 11},
 {u'_id': u'29012', u'count': 11},
 {u'_id': u'29015', u'count': 10},
 {u'_id': u'29005', u'count': 10},
 {u'_id': u'29008', u'count': 7},
 {u'_id': u'29013', u'count': 7},
 {u'_id': u'29071', u'count': 4},
 {u'_id': u'29140', u'count': 4},
 {u'_id': u'29011', u'count': 3},
 {u'_id': u'29003', u'count': 3},
 {u'_id': u'29590', u'count': 2},
 {u'_id': u'29009', u'count': 1}]
```

In [19]:
```
#list top 10 street names in data base
streets = collection.aggregate([
        {"$match" : {"address.street" : {"$exists" : 1}}}, \
        {"$group" : {"_id" : "$address.street", "count" : {"$sum" : 1}}}, \
        {"$sort" : {"count" : -1}},
        {"$limit":10}])

pprint.pprint(list(streets))
```

```
[{u'_id': u'Calle H\xe9roe de Sostoa', u'count': 83},
 {u'_id': u'Paseo Mar\xedtimo', u'count': 55},
 {u'_id': u'Calle Villafuente', u'count': 42},
 {u'_id': u'Calle Jamaica', u'count': 41},
 {u'_id': u'Calle Mendoza', u'count': 41},
 {u'_id': u'Calle de Tom\xe1s Echevarr\xeda', u'count': 27},
 {u'_id': u'Calle la Hoz', u'count': 26},
 {u'_id': u'Calle Ayala', u'count': 26},
 {u'_id': u'Calle Jose Bergamin', u'count': 18},
 {u'_id': u'Plaza Federico Garc\xeda Lorca', u'count': 17}]
```

In [20]:
```
#Top 10 amenities
top10_amenities = collection.aggregate([{"$match":{"amenity":{"$exists":1}}},
                             {"$group":{"_id":"$amenity","count":{"$sum":1}}},

                             {"$sort":{"count":-1}},
                             {"$limit":10}])
display(list(top10_amenities))
```

```
[{u'_id': u'restaurant', u'count': 365},
 {u'_id': u'parking', u'count': 349},
 {u'_id': u'school', u'count': 242},
 {u'_id': u'bench', u'count': 103},
 {u'_id': u'cafe', u'count': 86},
 {u'_id': u'bank', u'count': 83},
 {u'_id': u'bar', u'count': 81},
 {u'_id': u'place_of_worship', u'count': 76},
 {u'_id': u'fuel', u'count': 67},
 {u'_id': u'pharmacy', u'count': 62}]
```

3. Additional data exploration using MongoDB queries

In [21]:
```
#Top 5 building types
type_buildings = collection.aggregate([
    {'$match': {'building': {'$exists': 1}}},
    {'$group': { '_id': '$building','count': {'$sum': 1}}},
    {'$sort': {'count': -1}}, {'$limit': 5}
])

pprint.pprint(list(type_buildings))
```

```
[{u'_id': u'yes', u'count': 12729},
 {u'_id': u'residential', u'count': 1785},
 {u'_id': u'house', u'count': 597},
 {u'_id': u'commercial', u'count': 155},
 {u'_id': u'school', u'count': 123}]
```

In [22]:
```
#Top 3 cult buildings
religion_buildings = collection.aggregate([
        {"$match" : {"amenity" : "place_of_worship"}}, \
        {"$group" : {"_id" : {"religion" : "$religion", "denomination" : "$den
omination"}, "count" : {"$sum" : 1}}}, \
        {"$sort" : {"count" : -1}}, {'$limit': 3}])

pprint.pprint(list(religion_buildings))
```

```
[{u'_id': {u'denomination': u'catholic', u'religion': u'christian'},
  u'count': 47},
 {u'_id': {u'religion': u'christian'}, u'count': 19},
 {u'_id': {u'denomination': u'roman_catholic', u'religion': u'christian'},
  u'count': 5}]
```

In [23]:
```
#Top 10 leisures
leisures = collection.aggregate([{"$match" : {"leisure" : {"$exists" : 1}}}, \
                           {"$group" : {"_id" : "$leisure", "count" : {"$sum"
: 1}}}, \
                           {"$sort" : {"count" : -1}}, \
                           {"$limit" : 10}])
pprint.pprint(list(leisures))
```

```
[{u'_id': u'garden', u'count': 893},
 {u'_id': u'swimming_pool', u'count': 590},
 {u'_id': u'pitch', u'count': 504},
 {u'_id': u'park', u'count': 257},
 {u'_id': u'playground', u'count': 164},
 {u'_id': u'sports_centre', u'count': 47},
 {u'_id': u'fitness_station', u'count': 41},
 {u'_id': u'common', u'count': 28},
 {u'_id': u'water_park', u'count': 16},
 {u'_id': u'stadium', u'count': 8}]
```

In [24]:
```
#Top 10 univeristies
universities = collection.aggregate([
        {"$match" : {"amenity" : "university"}}, \
        {"$group" : {"_id" : {"name" : "$name"}, "count" : {"$sum" : 1}}}, \
        {"$sort" : {"count" : -1}},
        {"$limit":10}
    ])

pprint.pprint(list(universities))
```

```
[{u'_id': {u'name': None}, u'count': 7},
 {u'_id': {u'name': u'Facultad de Filosof\xeda y Letras'}, u'count': 7},
 {u'_id': {u'name': u'Talleres Edificio Ingenier\xedas'}, u'count': 1},
 {u'_id': {u'name': u'Biblioteca de la Facultad de Filosof\xeda y Letras'},
  u'count': 1},
 {u'_id': {u'name': u'Aulario L\xf3pez Pe\xf1alver'}, u'count': 1},
 {u'_id': {u'name': u'Escuela T\xe9cnica Superior de Ingenieros Industriale
s'},
  u'count': 1},
 {u'_id': {u'name': u'Aulario Comunicaci\xf3n'}, u'count': 1},
 {u'_id': {u'name': u'CIMES'}, u'count': 1},
 {u'_id': {u'name': u'Servicio Central de Inform\xe1tica (SCI)'}, u'count':
 1},
 {u'_id': {u'name': u'Servicios de Investigaci\xf3n (SCAI)'}, u'count': 1}]
```

In [25]:
```
#Top 10 cuisines
restaurant = collection.aggregate([
        {"$match":{"cuisine":{"$exists":1},"amenity":"restaurant"}},
        {"$group":{"_id":"$cuisine","count":{"$sum":1}}},
        {"$sort":{"count":-1}}, {"$limit":10} ])
pprint.pprint(list(restaurant))
```

```
[{u'_id': u'regional', u'count': 87},
 {u'_id': u'spanish', u'count': 13},
 {u'_id': u'tapas', u'count': 8},
 {u'_id': u'chinese', u'count': 5},
 {u'_id': u'pizza', u'count': 5},
 {u'_id': u'italian', u'count': 4},
 {u'_id': u'seafood', u'count': 3},
 {u'_id': u'international', u'count': 3},
 {u'_id': u'burger', u'count': 3},
 {u'_id': u'asian', u'count': 3}]
```

4. Ideas for additional improvements

The data base needs to be updated with the missing street types as there are a large number without it.

In [26]:
```
noaddresses=collection.find({'address.street': {'$exists': 0}}).count()
display(noaddresses)
```

```
373199
```

This can be done either by having the information first hand, entering the data one by one or using other map data such as https://www.google.es/maps (https://www.google.es/maps) or https://www.viamichelin.es/ (https://www.viamichelin.es/).

It can be observed that the maps have to be cleaned for any imput done by hand and not chosen from a list. The same operation that has been performed for the street types can be performed also for the phone numbers.

Resources:

https://github.com/bestkao/data-wrangling-with-openstreetmap-and-mongodb (https://github.com/bestkao/data-wrangling-with-openstreetmap-and-mongodb)

https://github.com/lyvinhhung/Udacity-Data-Analyst-Nanodegree/blob/master/p3%20-%20Wrangle%20OpenStreetMap%20Data/P3%20-%20Data%20Wrangling%20with%20MongoDB.ipynb (https://github.com/lyvinhhung/Udacity-Data-Analyst-Nanodegree/blob/master/p3%20-%20Wrangle%20OpenStreetMap%20Data/P3%20-%20Data%20Wrangling%20with%20MongoDB.ipynb)

```
In [ ]:
```