OpenStreet Map Wrangling Project with MongoDB

Pradyut Vatsa

Map Area: City Bengaluru, Bangalore Urban, Karnataka, 560001, India

http://www.openstreetmap.org/node/3401391999 (http://www.openstreetmap.org/node/3401391999)

In [1]:
```
import string
import codecs
import re
import collections
import json
import pymongo
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
```

In [2]:
```
import pprint
import xml.etree.ElementTree as ET  # Use cElementTree or lxml if too slow

OSM_FILE = "bengaluru_india.osm"  # Replace this with your osm file
SAMPLE_FILE = "bengaluru_sample.osm"

k = 5 # 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-pytho
    """
    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>')
```

In [2]:
```
import pprint
import xml.etree.ElementTree as ET  # Use cElementTree or lxml if too slow

OSM_FILE = "bengaluru_india.osm"  # Replace this with your osm file
SAMPLE_FILE = "bengaluru_sample.osm"

k = 5 # 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-pytho
    """
    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>')
```

In [3]:
```
#Let's try to build a processing file to dive into and get a sense of the data for auditing and cleaning process
tags = {}
postcode_vals = []
street_names = []
k_values = {}
for event,elem in ET.iterparse(SAMPLE_FILE):
    if tags.has_key(elem.tag):
        tags[elem.tag] += 1
    else:
        tags[elem.tag] = 1
    for tag in elem.iter("tag"):
        if tag.attrib['k'] == 'addr:postcode' or tag.attrib['k'] == 'postal_code':
            postcode_vals.append(tag.attrib['v'])
        elif tag.attrib['k'] == 'addr:street':
            street_names.append(tag.attrib['v'])

        if k_values.has_key(tag.attrib['k']):
            k_values[tag.attrib['k']] += 1
        else:
            k_values[tag.attrib['k']] = 1
```

In [4]:
```
tags
```

Out[4]:
```
{'member': 942,
 'nd': 703959,
 'node': 568587,
 'osm': 1,
 'relation': 188,
 'tag': 155572,
 'way': 130535}
```

In [5]:
```
#Look at the typical street names in the sample file. It should give us an inclination on how to wrangle this data.
print len(street_names)
pprint.pprint(street_names)
```

```
2565
['Outer Ring Road',
 'Outer Ring Road',
 'Sarjapur Road',
 'Sarjapur Road',
 'Sarjapur Road',
 'Sarjapur Road',
 'Rani Sarla Devi Circle',
 'Rani Sarla Devi Circle',
 'Outer Ring Road',
 'Outer Ring Road',
 'BTM Main Road',
 'BTM Main Road',
 '100 Feet Road',
 '100 Feet Road',
 "Saint Mark's Road",
 "Saint Mark's Road",
 'New BEL Road',
 'New BEL Road',
 'Dr B R Ambedkar Veedhi'
```

In [6]:
```
pprint.pprint(k_values)
```

```
{'FIXME': 21,
 'Friary': 3,
 'GPS_Trail': 3,
 'access': 1494,
 'addr:city': 951,
 'addr:country': 36,
 'addr:full': 18,
 'addr:housename': 447,
 'addr:housenumber': 1614,
 'addr:interpolation': 27,
 'addr:locality': 3,
 'addr:number': 18,
 'addr:place': 12,
 'addr:postcode': 957,
 'addr:state': 27,
 'addr:street': 2565,
 'addr:suburb': 3,
 'admin_level': 12,
 'aerodrome:type': 3,
```

In [7]:
```
for event, elem in ET.iterparse(SAMPLE_FILE):
    for tag in elem.iter('tag'):
        if tag.attrib['k'] == 'name:kn' or tag.attrib['k'] == 'name:kn:iso15919' or tag.attrib['k'] == 'name:ta':
            pprint.pprint(elem.attrib)
```

```
{'k': 'name:kn',
 'v': u'\u0ca4\u0cbf.\u0ca8\u0c82.\u0cb6\u0ccd\u0cb0\u0cc0 \u0cb5\u0cc3\u0ca4\u0ccd\u0ca4'}
{'changeset': '30186610',
 'id': '248776656',
 'lat': '12.9367887',
 'lon': '77.5800287',
 'timestamp': '2015-04-13T12:47:38Z',
 'uid': '827808',
 'user': 'yogi_ks',
 'version': '28'}
{'k': 'name:kn',
 'v': u'\u0cae\u0cbe\u0cb0\u0cc1\u0ca4\u0cbf \u0cb5\u0cc3\u0ca4\u0ccd\u0ca4'}
{'changeset': '31435775',
 'id': '248838888',
 'lat': '12.9435335',
 'lon': '77.5582819',
 'timestamp': '2015-05-25T06:09:41Z',
 'uid': '2897134',
 'user': 'Naresh08',
```

In [8]:
```
#cleaning up postcodes
def postcode_audit(postcode):
    '''
    This function audits the postal code information. It is found in the 'k' attribute of the tag tag within node, way or
    relation. It takes in the 'v' attribute value of the tag i.e. the postal code for that data item and checks for format
    error in the first step and then runs the value through a regular expression check

    >>> postcode_audit('560100')
    '560100'
    >>> postcode_audit('560 071')
    '560071'
    >>> postcode_audit('5623456')
    'FIXME: Incorrect Value'

    Args:
    postcode: a string input

    Returns:
    string: Value if regex match is found else error message 'FIXME: Incorrect Value' is returned
    '''

    postcode = postcode.strip(" ")
    if len(postcode) != 6:
        postcode = postcode.replace(" ", "")
    postcode = postcode
    error_msg = 'FIXME: Incorrect Value'
    '''
    Further, checking for postcode errors in the file. After taking care of the seemingly obvious error of formatting, I'm
    a regex search to see whether all postcodes are in sync with the postcode format of my selected region
    '''
    if re.search(r'560\d\d\d', postcode):
        return postcode
    return error_msg
```

In [9]:

```
#Cleaning up street names
street_type_re = re.compile(r'\b\S+\.?$', re.IGNORECASE)
expected = ["Street", "Road", "Avenue", "Circle", "street", "road", "Road)", "road)"]
def audit_street_names(name):
    '''
    This function is responsible for cleaning street names. These names are potentially found in the 'k' attribute of the
    element in one of the top-level elements i.e. node, way or relation. The cleaning is done based on the rules discussed
    project as defined.

    >>>audit_street_names('Dr B R Ambedkar Veedhi')
    'FIXME:Incorrect street name'
    >>>audit_street_names('Sarjapur Road')
    'Sarjapur Road'
    >>>audit_street_names('Old Airport Rd')
    'Old Airport Road'
    >>>audit_street_names('27th Main')
    '27th Main Road'
    >>>audit_street_names('Hosur Road, Koramangala')
    'Hosur Road, Koramangala'

    Args:
    name: string that is the original 'v'attribute of the tag whose corresponding 'k' value is 'addr:street'
    Returns:
    a string cleaned as per the process above
    '''
    name = name.strip(" ")
    name = name.strip(",")
    m = street_type_re.search(name)
    if m:
        street_type = m.group()
        if street_type not in expected:
            if re.search(r'(MAIN|CROSS)', street_type, re.IGNORECASE):
                name = name + ' Road'
                return name
            else:
                if re.search(r'road', name, re.IGNORECASE):
                    return name
                elif re.search(r'rd', name, re.IGNORECASE):
                    mObj = re.search(r'rd', name, re.IGNORECASE).group()
                    name = name.replace(mObj, 'Road')
                    return name
                elif re.search(r'(MAIN)', name, re.IGNORECASE):
```

```
                mObj = re.search(r'(MAIN)', name, re.IGNORECASE).group()
                name = name.replace(mObj, mObj + ' Road')
                return name
            elif re.search(r'(CROSS)', name, re.IGNORECASE):
                mObj = re.search(r'(CROSS)', name, re.IGNORECASE).group()
                name = name.replace(mObj, mObj + ' Road')
                return name
            else:
                #prob_street_names.append(item)
                return 'FIXME:Incorrect street name'
    else:
        return name
```

Now that we've got all the helper functions to identify rogue tag values (e.g. FIXME), clean street names based on my domain knowledge of Bengaluru as a region, and correct the formatting of postalcodes data wrangling is nearing completion and we can put this into a data model. To understand what kind of a data model we're talking about, we might want to look at the tag and associated data structure with it to give us a clear idea of how to structure the data. One of the helper functions to do that is the tags collection dictionary that gives us an idea of the types of tags that are abundant in this xml file. From the OSM XML documentation, we know that node, way and relation are three data primitives represented by top-level tags here in the file. Let's first look at the desired data model for the same. When it is a node, we can look at the following data model: { "id": "2406124091", "type: "node", "visible":"true", "created": { "version":"2", "changeset":"17206049", "timestamp":"2013-08-03T16:43:42Z", "user":"linuxUser16", "uid":"1219059" }, "pos": [41.9757030, -87.6921867], "address": { "housenumber": "5157", "postcode": "60625", "street": "North Lincoln Ave" }, "amenity": "restaurant", "cuisine": "mexican", "name": "La Cabana De Don Luis", "phone": "1 (773)-271-5176" } This implies that the output will be a list of dictionaries. So, whatever be the original shape of data with the associated node tags, I need to, with the help of my helper functions, shape the data into this form for consumption into database.

Additionally for way, specifically we should do the following: for "way" specifically:

should be turned into "node_refs": ["305896090", "1719825889"]

For relation, we should look at the following model: { "id": "1332095", "type: "relation", "created": { "version":"1", "changeset":"6717067", "timestamp":"2010-12-20T15:35:51Z", "user":"Alexander Hunziker", "uid":"21825" }, "ref_info": { "ref": ["90631804",90631813] "role": ["outer",inner] "type": ["way", "way"] }, "type": "multipolygon" }

In [10]:

```python
CREATED = [ "version", "changeset", "timestamp", "user", "uid"]
member_ref = ["ref", "role", "type"]
def shape_element(element):
    '''
    The shape element function shapes the element as per the desired data model explained in the above section of this doc
    and the project document as well. It first filters the element as either a node, way or a relation (the three data
    primitives of the OSM XML data format) and then cleans the street names and the postcodes as per the helper functions
    Then it reads in each element info iteratively and shapes it as per the format

    >>>shape_element('node')
     "id": "2406124091", "type: "node", "visible":"true", "created": { "version":"2", "changeset":"17206049",
     "timestamp":"2013-08-03T16:43:42Z", "user":"linuxUser16", "uid":"1219059" }, "pos": [41.9757030, -87.6921867],
     "address": { "housenumber": "5157", "postcode": "60625", "street": "North Lincoln Ave" }, "amenity": "restaurant",
     "cuisine": "mexican", "name": "La Cabana De Don Luis", "phone": "1 (773)-271-5176" }

    Args: Takes in a string which is an element of the input xml file format
    Returns: The final shaped element that can be written into the json output file. One can visualize it as a dictionary
    returned for every input element which is then outputted per line of a json document
    '''
    node = {}
    way = {}
    relation = {}
    address_val = {}
    details_val = {}
    ref_val = {}
    ref_values = []
    if element.tag == "node":
        # Cleaning street names as discussed earlier
        for tag in element.iter("tag"):
            k_value = tag.attrib['k']
            if k_value == "addr:street":
                tag.attrib['v'] = audit_street_names(tag.attrib['v'])

            elif k_value == "addr:postcode" or k_value == "postal_code":
                    tag.attrib['v'] = postcode_audit(tag.attrib['v'])

        #Shaping the element as discussed above

        node["id"] = element.attrib["id"]
        if element.attrib.has_key('visible'):
            node["visible"] = element.attrib["visible"]
        node["type"] = element.tag
        created_vals_dict = {item: element.attrib[item] for item in CREATED}
        node['created'] = created_vals_dict
        pos_values = [float(element.attrib["lat"]), float(element.attrib["lon"])]
        node['pos'] = pos_values

        for tag in element.iter("tag"):
            k_value = tag.attrib['k']
            if k_value == 'FIXME' or k_value == 'fixme':
                continue
            elif len(k_value.split(":")) > 2:
                continue
            else:
                if (k_value.startswith("addr:")):
                    address_val[k_value[5:]] = tag.attrib['v']
                    node['address'] = address_val

                else:
                    if len(k_value.split(":")) > 1:
                        details_val[k_value.split(":")[1]] = tag.attrib['v']
                        node['details'] = details_val
                    else:
                        node[k_value] = tag.attrib['v']

        return node
    elif element.tag == "way":
        # Cleaning street names as discussed earlier
        for tag in element.iter("tag"):
            k_value = tag.attrib['k']
            if k_value == "addr:street":
                tag.attrib['v'] = audit_street_names(tag.attrib['v'])

            elif k_value == "addr:postcode" or k_value == "postal_code":
                    tag.attrib['v'] = postcode_audit(tag.attrib['v'])

        #Shaping the element as discussed above
        way["id"] = element.attrib["id"]
        way["type"] = element.tag
        created_vals_dict = {item: element.attrib[item] for item in CREATED}
        way['created'] = created_vals_dict
        #pos_values = [float(element.attrib["lat"]), float(element.attrib["lon"])]
        #way['pos'] = pos_values
        for tag in element.iter("tag"):
            k_value = tag.attrib['k']
            if k_value == 'FIXME' or k_value == 'fixme':
                continue
            elif len(k_value.split(":")) > 2:
                continue
            else:
                if (k_value.startswith("addr:")):
                    address_val[k_value[5:]] = tag.attrib['v']
                    way['address'] = address_val

                else:
                    if len(k_value.split(":")) > 1:
                        details_val[k_value.split(":")[1]] = tag.attrib['v']
                        way['details'] = details_val
                    else:
                        way[k_value] = tag.attrib['v']

        for tag in element.iter("nd"):
            ref_values.append(tag.attrib['ref'])
        way["node_refs"] = ref_values

        return way

    elif element.tag == "relation":
        # Cleaning street names as discussed earlier
        for tag in element.iter("tag"):
            k_value = tag.attrib['k']
            if k_value == "addr:street":
                tag.attrib['v'] = audit_street_names(tag.attrib['v'])

            elif k_value == "addr:postcode" or k_value == "postal_code":
                    tag.attrib['v'] = postcode_audit(tag.attrib['v'])

        #Shaping the element as discussed above
        relation["id"] = element.attrib["id"]
        relation["type"] = element.tag
        created_vals_dict = {item: element.attrib[item] for item in CREATED}
        relation['created'] = created_vals_dict
        ref_val = {item:[] for item in member_ref}

        for tag in element.iter('member'):
            ref_val["ref"].append(tag.attrib['ref'])
            ref_val["role"].append(tag.attrib['role'])
            ref_val["type"].append(tag.attrib['type'])
        relation["ref_info"] = ref_val

        for tag in element.iter("tag"):
            k_value = tag.attrib['k']
            if k_value == 'FIXME' or k_value == 'fixme':
                continue
            elif len(k_value.split(":")) > 2:
                continue
            else:
                if (k_value.startswith("addr:")):
                    address_val[k_value[5:]] = tag.attrib['v']
                    relation['address'] = address_val

                else:
                    if len(k_value.split(":")) > 1:
                        details_val[k_value.split(":")[1]] = tag.attrib['v']
                        relation['details'] = details_val
                    else:
                        relation[k_value] = tag.attrib['v']
        return relation
    else:
        return None
```

In [12]:
```
#taking in the sample xml file and ouputting the data in json format for ingestion into the database
file_out = "{0}.json".format(SAMPLE_FILE)
data = []
with codecs.open(file_out, "w") as fo:
    for event,elem in ET.iterparse(SAMPLE_FILE):

        el = shape_element(elem)
        if el:
            data.append(el)
            fo.write(json.dumps(el, indent=2)+"\n")
```

In [13]:
```
from pymongo import MongoClient
client = MongoClient("mongodb://localhost:27017")
db = client.quiz
```

In [15]:
```
node_count = db.project.find({"type": "node"}).count()
print node_count
way_count = db.project.find({"type": "way"}).count()
print way_count
```

```
568585
130529
```

In [16]:
```
#Who is the most active user?
most_active_user_list = db.project.aggregate([
                 {"$group": {"_id": "$created.user",
                           "count" : {"$sum" : 1}}},
                 {"$sort": {"count": -1}},
                 {"$limit" : 20}])
for item in most_active_user_list:
    pprint.pprint(item)
```

```
{u'_id': u'jasvinderkaur', u'count': 25228}
{u'_id': u'akhilsai', u'count': 23909}
{u'_id': u'premkumar', u'count': 23260}
{u'_id': u'saikumar', u'count': 23133}
{u'_id': u'shekarn', u'count': 19932}
{u'_id': u'vamshikrishna', u'count': 18897}
{u'_id': u'PlaneMad', u'count': 18245}
{u'_id': u'himalay', u'count': 17690}
{u'_id': u'himabindhu', u'count': 17455}
{u'_id': u'sdivya', u'count': 16980}
{u'_id': u'hareesh11', u'count': 16519}
{u'_id': u'vamshiN', u'count': 16217}
{u'_id': u'harishk', u'count': 15395}
{u'_id': u'sampath reddy', u'count': 14379}
{u'_id': u'bindhu', u'count': 13996}
{u'_id': u'kranthikumar', u'count': 13677}
{u'_id': u'Navaneetha', u'count': 13492}
{u'_id': u'samuelmj', u'count': 12878}
{u'_id': u'shivajim', u'count': 12661}
{u'_id': u'praveeng', u'count': 12657}
```

In [16]:
```
#Finding number of contributions per year. How has the contribution trend changed over the years?
```

contributions/year year

In [17]:
```
contribution_16 = db.project.find( { "created.timestamp": { "$regex": "2016" } } ).count()
contribution_15 =  db.project.find( { "created.timestamp": { "$regex": "2015" } } ).count()
contribution_14 =  db.project.find( { "created.timestamp": { "$regex": "2014" } } ).count()
contribution_13 =  db.project.find( { "created.timestamp": { "$regex": "2013" } } ).count()
contribution_12 =  db.project.find( { "created.timestamp": { "$regex": "2012" } } ).count()
contribution_11 =  db.project.find( { "created.timestamp": { "$regex": "2011" } } ).count()
contribution_10 =  db.project.find( { "created.timestamp": { "$regex": "2010" } } ).count()
contribution_09 =  db.project.find( { "created.timestamp": { "$regex": "2009" } } ).count()
contribution_08 =  db.project.find( { "created.timestamp": { "$regex": "2008" } } ).count()
```

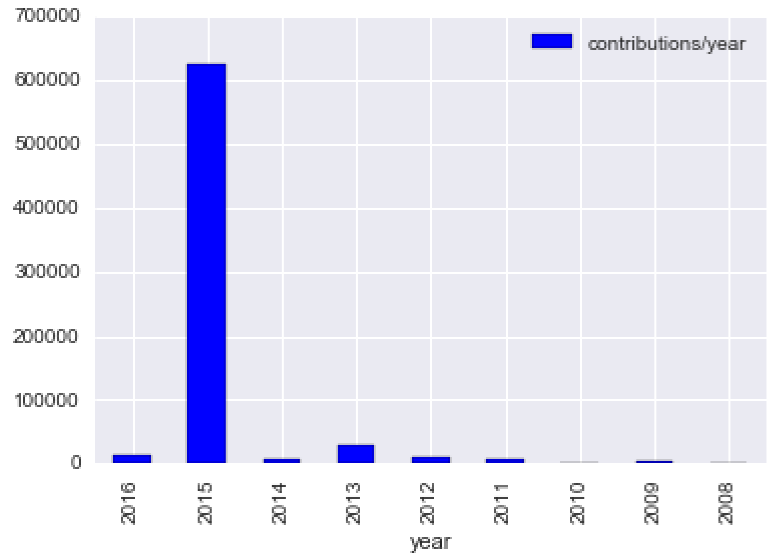
In [18]:
```
map_contribution_per_year = [contribution_16, contribution_15, contribution_14, contribution_13, contribution_12, contribu
                             contribution_10, contribution_09,contribution_08]
year = [2016, 2015, 2014, 2013, 2012, 2011, 2010, 2009, 2008]
contribution_dataset = list(zip(map_contribution_per_year, year))
contribution_df = pd.DataFrame(data = contribution_dataset, columns=['contributions/year', 'year'])
contribution_df
```

Out[18]:

| | **contributions/year** | **year** |
|---|---|---|
| **0** | 13574 | 2016 |
| **1** | 625477 | 2015 |
| **2** | 7038 | 2014 |
| **3** | 27993 | 2013 |
| **4** | 9981 | 2012 |
| **5** | 6576 | 2011 |
| **6** | 1790 | 2010 |
| **7** | 5409 | 2009 |
| **8** | 1466 | 2008 |

```
In [19]: %matplotlib inline
         contribution_df.plot(kind = 'bar', x = 'year', y = 'contributions/year')
```

```
Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x96613358>
```



In [20]:
```
pipeline = [{"$group": {"_id": "$address.city",
                        "count": {"$sum" : 1}}},
            {"$sort": {"count": -1}}]
for item in db.project.aggregate(pipeline):
    pprint.pprint(item)
```

```
{u'_id': None, u'count': 698993}
{u'_id': u'Bangalore', u'count': 219}
{u'_id': u'Bengaluru', u'count': 37}
{u'_id': u'Marathahalli, Bangalore', u'count': 7}
{u'_id': u'bangalore', u'count': 7}
{u'_id': u'Basaveshwara Nagar, Bangalore', u'count': 5}
{u'_id': u'Whitefield, Bangalore', u'count': 4}
{u'_id': u'Koramangala', u'count': 4}
{u'_id': u'Mahadevapura, Bangalore', u'count': 3}
{u'_id': u'Gandhi Nagar, Bangalore', u'count': 3}
{u'_id': u'BENGALURU', u'count': 3}
{u'_id': u'Mathikere', u'count': 2}
{u'_id': u'Belandur, Bangalore', u'count': 1}
{u'_id': u'Kalyan Nagar, Bangalore', u'count': 1}
{u'_id': u'Hoodi, Mahadevapura, Bangalore', u'count': 1}
{u'_id': u'Kodihalli, Bangalore', u'count': 1}
{u'_id': u'Adakamranahalli', u'count': 1}
{u'_id': u'Seshadripuram, Bangalore', u'count': 1}
{u'_id': u'Bommanahalli, Bangalore', u'count': 1}
{u'_id': u'Bidadi', u'count': 1}
{u'_id': u'Begur', u'count': 1}
{u'_id': u'K.R Puram, Bangalore', u'count': 1}
{u'_id': u'Marathhalli', u'count': 1}
{u'_id': u'Marutinagar, Yelahanka, Bangalore', u'count': 1}
{u'_id': u'hindupur', u'count': 1}
{u'_id': u'Singasandra', u'count': 1}
{u'_id': u'Banglore', u'count': 1}
{u'_id': u'peenya 2nd stage, bengaluru', u'count': 1}
{u'_id': u'Shivaji Nagar, Bangalore', u'count': 1}
{u'_id': u'Hoodi, Bangalore', u'count': 1}
{u'_id': u'Cheemasandra', u'count': 1}
{u'_id': u'K.R Puram', u'count': 1}
{u'_id': u'Abbigere Village', u'count': 1}
{u'_id': u'Bangalore Urban', u'count': 1}
{u'_id': u'banglore', u'count': 1}
```

In [21]:
```
pipeline = [{"$group": {"_id": "$amenity",
                        "count": {"$sum" : 1}}},
            {"$sort": {"count": -1}}]
for item in db.project.aggregate(pipeline):
    pprint.pprint(item)
```

```
{u'_id': None, u'count': 697593}
{u'_id': u'restaurant', u'count': 247}
{u'_id': u'place_of_worship', u'count': 183}
{u'_id': u'atm', u'count': 130}
{u'_id': u'bank', u'count': 129}
{u'_id': u'school', u'count': 121}
{u'_id': u'hospital', u'count': 89}
{u'_id': u'fast_food', u'count': 81}
{u'_id': u'college', u'count': 72}
{u'_id': u'fuel', u'count': 67}
{u'_id': u'pharmacy', u'count': 60}
{u'_id': u'cafe', u'count': 58}
{u'_id': u'bench', u'count': 56}
{u'_id': u'parking', u'count': 46}
{u'_id': u'bar', u'count': 28}
{u'_id': u'bus_station', u'count': 25}
{u'_id': u'toilets', u'count': 19}
{u'_id': u'clinic', u'count': 19}
{u'_id': u'police', u'count': 18}
{u'_id': u'post_office', u'count': 18}
{u'_id': u'community_centre', u'count': 18}
{u'_id': u'library', u'count': 17}
{u'_id': u'university', u'count': 16}
{u'_id': u'cinema', u'count': 15}
{u'_id': u'kindergarten', u'count': 15}
{u'_id': u'theatre', u'count': 14}
{u'_id': u'pub', u'count': 13}
{u'_id': u'swimming_pool', u'count': 12}
{u'_id': u'public_building', u'count': 11}
{u'_id': u'recycling', u'count': 11}
{u'_id': u'marketplace', u'count': 9}
{u'_id': u'waste_basket', u'count': 8}
{u'_id': u'dentist', u'count': 7}
{u'_id': u'telephone', u'count': 5}
{u'_id': u'veterinary', u'count': 5}
{u'_id': u'fountain', u'count': 4}
{u'_id': u'shelter', u'count': 4}
{u'_id': u'doctors', u'count': 4}
{u'_id': u'parking_space', u'count': 3}
{u'_id': u'courthouse', u'count': 3}
{u'_id': u'ice_cream', u'count': 3}
{u'_id': u'taxi', u'count': 3}
```

```
{u'_id': u'townhall', u'count': 3}
{u'_id': u'bicycle_parking', u'count': 3}
{u'_id': u'waste_disposal', u'count': 3}
{u'_id': u'coworking_space', u'count': 3}
{u'_id': u'office', u'count': 2}
{u'_id': u'studio', u'count': 2}
{u'_id': u'post_box', u'count': 2}
{u'_id': u'arts_centre', u'count': 2}
{u'_id': u'car_wash', u'count': 2}
{u'_id': u'childcare', u'count': 2}
{u'_id': u'motorcycle_parking', u'count': 2}
{u'_id': u'bureau_de_change', u'count': 2}
{u'_id': u'dojo', u'count': 1}
{u'_id': u'kennels', u'count': 1}
{u'_id': u'parking_entrance', u'count': 1}
{u'_id': u'Forex', u'count': 1}
{u'_id': u'vending_machine', u'count': 1}
{u'_id': u'Residence', u'count': 1}
{u'_id': u'nursing_home', u'count': 1}
{u'_id': u'diagnostic_centre', u'count': 1}
{u'_id': u'bar;pub;restaurant', u'count': 1}
{u'_id': u'emergency_phone', u'count': 1}
{u'_id': u'food_court', u'count': 1}
{u'_id': u'drinking_water', u'count': 1}
{u'_id': u'streetfood', u'count': 1}
{u'_id': u'opthalmologist', u'count': 1}
{u'_id': u'milk-booth', u'count': 1}
{u'_id': u'Departmental Store', u'count': 1}
{u'_id': u'car_rental', u'count': 1}
{u'_id': u'printer', u'count': 1}
{u'_id': u'embassy', u'count': 1}
{u'_id': u'Syndicate bank', u'count': 1}
{u'_id': u'spa', u'count': 1}
{u'_id': u'planetarium', u'count': 1}
{u'_id': u'grave_yard', u'count': 1}
```

In [22]:
```
pipeline = [{"$match" : {"amenity" : "restaurant"}},
            {"$group": {"_id": "$cuisine",
                        "count": {"$sum" : 1}}},
            {"$sort": {"count": -1}}]
for item in db.project.aggregate(pipeline):
    pprint.pprint(item)
```

```
{u'_id': None, u'count': 114}
{u'_id': u'indian', u'count': 49}
{u'_id': u'chinese', u'count': 14}
{u'_id': u'vegetarian', u'count': 13}
{u'_id': u'regional', u'count': 10}
{u'_id': u'pizza', u'count': 7}
{u'_id': u'international', u'count': 6}
{u'_id': u'italian', u'count': 5}
{u'_id': u'South_Indian', u'count': 3}
{u'_id': u'Punjabi', u'count': 2}
{u'_id': u'Andhra', u'count': 2}
{u'_id': u'chicken', u'count': 2}
{u'_id': u'kebab', u'count': 2}
{u'_id': u'regional,_goan', u'count': 1}
{u'_id': u'coorg', u'count': 1}
{u'_id': u'kerala', u'count': 1}
{u'_id': u'punjabi', u'count': 1}
{u'_id': u'south_Indian', u'count': 1}
{u'_id': u'Hyderabadi_Biryani', u'count': 1}
{u'_id': u'fish', u'count': 1}
{u'_id': u'chinese;indian', u'count': 1}
{u'_id': u'ice_cream', u'count': 1}
{u'_id': u'american', u'count': 1}
{u'_id': u'portuguese', u'count': 1}
{u'_id': u'North_Indian,_Italian', u'count': 1}
{u'_id': u'briyani', u'count': 1}
{u'_id': u'Chettinad', u'count': 1}
{u'_id': u'mexican', u'count': 1}
{u'_id': u'andhra', u'count': 1}
{u'_id': u'arab', u'count': 1}
{u'_id': u'coffee_shop', u'count': 1}
```

```
In [23]: pipeline = [{"$match" : {"amenity" : "place_of_worship"}},
                     {"$group": {"_id": "$religion",
                                 "count": {"$sum" : 1}}},
                     {"$sort": {"count": -1}}]
         for item in db.project.aggregate(pipeline):
             pprint.pprint(item)
```

```
{u'_id': u'hindu', u'count': 108}
{u'_id': u'christian', u'count': 30}
{u'_id': None, u'count': 25}
{u'_id': u'muslim', u'count': 18}
{u'_id': u'hinduism', u'count': 1}
{u'_id': u'sikh', u'count': 1}
```

In [24]:
```
pipeline = [{"$match" : {"amenity" : "atm"}},
            {"$group": {"_id": "$name",
                        "count": {"$sum" : 1}}},
            {"$sort": {"count": -1}}]
for item in db.project.aggregate(pipeline):
    pprint.pprint(item)
```

```
{u'_id': None, u'count': 88}
{u'_id': u'Canara Bank', u'count': 3}
{u'_id': u'Axis Bank ATM', u'count': 3}
{u'_id': u'Canara Bank ATM', u'count': 2}
{u'_id': u'SBI ATM', u'count': 2}
{u'_id': u'ICICI Bank', u'count': 2}
{u'_id': u'ICICI', u'count': 2}
{u'_id': u'State Bank of Travancore', u'count': 1}
{u'_id': u'Punjab National Bank ATM', u'count': 1}
{u'_id': u'Citibank ATM', u'count': 1}
{u'_id': u'ING', u'count': 1}
{u'_id': u'State Bank of India', u'count': 1}
{u'_id': u'Bharatiya Mahila Bank', u'count': 1}
{u'_id': u'Karnataka Bank', u'count': 1}
{u'_id': u'HDFC bank ATM', u'count': 1}
{u'_id': u'IDBI', u'count': 1}
{u'_id': u'HDFC Bank ATM', u'count': 1}
{u'_id': u'SBI', u'count': 1}
{u'_id': u'Sbi', u'count': 1}
{u'_id': u'Citibank', u'count': 1}
{u'_id': u'SBI 24x7 e-Lobby', u'count': 1}
{u'_id': u'DCB Bank', u'count': 1}
{u'_id': u'HDFC Bank', u'count': 1}
{u'_id': u'Bandhan Bank', u'count': 1}
{u'_id': u'UCO Bank', u'count': 1}
{u'_id': u'HDFC', u'count': 1}
{u'_id': u'HDFC ATM', u'count': 1}
{u'_id': u'Lakshmi Vilas Bank ATM', u'count': 1}
{u'_id': u'Icici Bank ATM', u'count': 1}
{u'_id': u'PNB', u'count': 1}
{u'_id': u'Syndicate Bank ATM', u'count': 1}
{u'_id': u'Vijaya Bank', u'count': 1}
{u'_id': u'Axis Bank', u'count': 1}
{u'_id': u'ICICI Bank ATM', u'count': 1}
{u'_id': u'Axis ATM', u'count': 1}
```

In [26]:
```
pipeline = [{"$match" : {"amenity" : "hospital"}},
            {"$group": {"_id": "$name",
                        "count": {"$sum" : 1}}},
            {"$sort": {"count": -1}}]
for item in db.project.aggregate(pipeline):
    pprint.pprint(item)
```

```
{u'_id': None, u'count': 5}
{u'_id': u'Apollo Hospital', u'count': 2}
{u'_id': u'Sanjay Gandhi Accident and Trauma Hospital', u'count': 2}
{u'_id': u'Fortis Hospital', u'count': 1}
{u'_id': u'Government Maternity Hospital', u'count': 1}
{u'_id': u'Hema Poly Clinic', u'count': 1}
{u'_id': u'Dianova Diabetes Center and Hospital', u'count': 1}
{u'_id': u'Bharati Nursing Home', u'count': 1}
{u'_id': u'Clinic', u'count': 1}
{u'_id': u'Bangalore Hospital', u'count': 1}
{u'_id': u'BGS Global Hospital', u'count': 1}
{u'_id': u'KK Hospital', u'count': 1}
{u'_id': u'Mallya Audikesh Diagnostics Center', u'count': 1}
{u'_id': u'Presidency Hospital', u'count': 1}
{u'_id': u'Deeksha Orthopaedic Hospital', u'count': 1}
{u'_id': u'Brookfield Hospital', u'count': 1}
{u'_id': u'Ragavendra Hospital', u'count': 1}
{u'_id': u'SDS Tuberculosis and Chest Diseases Hospital', u'count': 1}
{u'_id': u'Maiya Nursing Home', u'count': 1}
{u'_id': u'Manipal Cure and Care', u'count': 1}
{u'_id': u'Nagappa Hadli Hospital', u'count': 1}
{u'_id': u'Koshi Hospital', u'count': 1}
{u'_id': u'Apollo Cradle', u'count': 1}
{u'_id': u'R C Hospital', u'count': 1}
{u'_id': u'Health Centre', u'count': 1}
{u'_id': u'Sapthagiri Hospital', u'count': 1}
{u'_id': u'Vikram Eye Clinic', u'count': 1}
{u'_id': u'Shalini Nursing & MAternity Home', u'count': 1}
{u'_id': u'Srinivasa Nursing Home', u'count': 1}
{u'_id': u'Karuna Clinic', u'count': 1}
{u'_id': u'Dhanavantri Hospital', u'count': 1}
{u'_id': u'Shanta Nethralaya', u'count': 1}
{u'_id': u'Devi Eye Hospital', u'count': 1}
{u'_id': u'Krishna Specialty Clinic', u'count': 1}
{u'_id': u'Diacon Hospital', u'count': 1}
{u'_id': u'Prerana Dental Clinic', u'count': 1}
{u'_id': u'Venketeshwara Hospital', u'count': 1}
{u'_id': u'Sri Sapthagiri Clinic', u'count': 1}
{u'_id': u'Rajbal Skin Clinic', u'count': 1}
{u'_id': u'Vignan Hospital', u'count': 1}
{u'_id': u'Ayurveda Clinic', u'count': 1}
{u'_id': u'Spruthy Hospital', u'count': 1}
```

```
{u'_id': u'Lakshmi Nursing Home', u'count': 1}
{u'_id': u'Citi Hospital', u'count': 1}
{u'_id': u'NIMHANS Casuality', u'count': 1}
{u'_id': u'Specialist Hospital', u'count': 1}
{u'_id': u'Vijay Hospital', u'count': 1}
{u'_id': u'Karthik Hospital', u'count': 1}
{u'_id': u'Dr. Ambedkar Medical College Hospital', u'count': 1}
{u'_id': u'Jayashree Hospital', u'count': 1}
{u'_id': u'Shobha Hospital', u'count': 1}
{u'_id': u'Columbia Asia', u'count': 1}
{u'_id': u'Aikya Healthcare', u'count': 1}
{u'_id': u'Shanbhag Hospital', u'count': 1}
{u'_id': u'Magnus Diagnostic Lab', u'count': 1}
{u'_id': u'Vaidyaratnam Oushadhasala', u'count': 1}
{u'_id': u'Sagar Chandramma', u'count': 1}
{u'_id': u'St Johns Medical College Hospital', u'count': 1}
{u'_id': u'Rajarajeshwari Dental College', u'count': 1}
{u'_id': u'Krishna Nursing Home', u'count': 1}
{u'_id': u'Ameena Medical centre', u'count': 1}
{u'_id': u'Kidwai Memorial Institute of Oncology', u'count': 1}
{u'_id': u'NG Imaging & Diagnostics', u'count': 1}
{u'_id': u'Sri Lakshmi Clinic', u'count': 1}
{u'_id': u'KLE Dental College', u'count': 1}
{u'_id': u'Dr. Agarwal Eye Hospital', u'count': 1}
{u'_id': u'Sanjay Gandhi Hospital', u'count': 1}
{u'_id': u'BEL Hospital', u'count': 1}
{u'_id': u'Bowring and Lady Curzon Hospitals', u'count': 1}
{u'_id': u'Rajashekar Hospital', u'count': 1}
{u'_id': u'Pushpa Nursing Home', u'count': 1}
{u'_id': u'Isolation Hospital', u'count': 1}
{u'_id': u"Employees' State Insurance Corporation Model Hospital", u'count': 1}
{u'_id': u'Minto Eye Hospital', u'count': 1}
{u'_id': u'Netra Eye Hospital', u'count': 1}
{u'_id': u'Sri Krishna Sevashrama Hospital', u'count': 1}
{u'_id': u'HAL Hospital', u'count': 1}
{u'_id': u'Netradhama Hospital', u'count': 1}
{u'_id': u'Meghana Nursing and Maternity Home', u'count': 1}
{u'_id': u'Institute of Ayurveda and Integrative Medicine (I-AIM)',
 u'count': 1}
{u'_id': u'Leprosy Hospital', u'count': 1}
{u'_id': u'CSI Hospital', u'count': 1}
{u'_id': u'Sagar Hospitals', u'count': 1}
```

In [ ]:

```

```