Project P3: Wrangle OpenStreetMap Data

Introduction

This project deals with wrangling a dataset and using SQL queries to gain insight into the data.

I used the map data from OpenStreetMap for the city of Bengaluru. Bengaluru, is the capital of the Indian state of Karnataka. Bengaluru (previously and popularly known as Bangalore) is also the city I grew up in, attended school and college in. It is also known as the Silicon valley of India owing to the large presence of the IT industry and an expanding base of multinational companies and startup ventures.

Problems Encountered During Wrangling

I used a 68 MB xml file obtained from Mapzen for the purpose of this project. While analyzing the data there were a few key issues I found and corrected programmatically. Below is a list of the inconsistencies I found followed by a brief description of each.

1. Incorrect Street Names
2. Incorrect phone number formatting
3. Varying versions of the city name city name is different
4. Incorrect values for the building field

Incorrect Street Names

The valid street types in Bengaluru are among 'Road', 'Nagar', 'Street', 'Cross', 'Main', 'Avenue', 'Block' and 'Veedhi' The different types of street types I found in the dataset are below.

```
set(['Rd', 'Sadashivanagar', 'Kodandarampura', 'Chickpet,', 'Nagar,',
'Chikpete,', 'Rajajinagar', 'Kodihalli,', 'ROAD', 'Subramanyanagar', '560080"',
'Block,Rajajinagar,', 'Karnataka', u'Stage,\xa0Rajajinagar', 'Habitat', 'Town,',
'Vyalikaval', 'Road,Malleshwaram', 'Nagar', 'Quarters', 'Station', 'Naga',
'ROad', 'Marg', 'West', 'street', 'International', 'Circle', 'East',
'Malleswaram', 'Road)', 'abc', 'Junction', 'Theatre', 'local_knowledge', 'RTO',
'Park', 'Bengaluru,', 'Cross,', 'Town', 'malleshwaram', 'Veedhi',
u'Cross,\xa0Malleshwaram', 'Block', 'Grounds', 'Block,', 'Road,Shivajinagar',
'Peenya', 'extension', 'Street,Shivajinagar', 'lavelle', 'road', 'Chamarajpet',
'Sheshadripuram,', 'Layout', 'Crossroad', 'Nagarathpete,', 'rajajinagar',
'Malleshwaram', 'Main,Malleshwaram', 'Street,', 'Shivajinagar', 'gudahalli',
'Care', 'Lane', 'Jeevanahalli', 'road,', 'Flyover', 'Road', 'Jayamahal',
'Seshadripuram', 'block', 'Rd.', 'hotel,rajajinagar', 'Seshadripuram,',
'Chickpet', 'Cubbonpet,', 'Guttahalli,', 'Majestic,', 'Subramanyanagar,2',
'Avenue', u'Road,\xa0Malleshwaram', 'Ground', 'City', 'Chamrajpet,',
'Sadashivnagar', 'Floor', 'Road,Rajajinagar', 'Estate,', 'Cross',
u'Stage,\xa0Rajajinaga', 'Main', 'block,rajajinagar', 'Tharagupet',
'sultanpet,', 'Rajajinagar,', 'Yeshwanthpur,', 'cross', 'Basavanagudi', 'Bank',
'Cross,Malleshwaram', 'Malleshwaram,', 'Main,malleshwaram', 'Chamarajpet,',
'Sheshadripuram', u'Road,', 'Nagar,,', 'Street', 'Crescent',
'Theatre,Rajajinaga', u'Shivanagar,\xa0Rajajinagar', 'Kalasipalayam'])
```

I thereby audited each street name that was not an expected value, used python's sequence matcher function to compare it with the expected type (I looked for a 60% match) and updated the value using the following code.

```python
#Function to create a mapping against expected street names using difflib's
sequence matcher
def street_mapping(street_types, EXPECTED):
    count = 0
    for b in street_types:
        for a in EXPECTED:
            seq = difflib.SequenceMatcher(None,a,b)
            if seq.ratio() >= 0.6:
                mapping[b] = a
                count += 1
    return count, mapping

# Function to update the street name based on the mapping dictionary
def update_street_name(street_type, MAPPING):
#take a text and replace words that match a key in a dictionary with the
associated value, return the changed text
    rc = re.compile('|'.join(map(re.escape, MAPPING)))
    def translate(match):
        return MAPPING[match.group(0)]
    return rc.sub(translate, street_type)
```

Below is dictionary created to map inconsistent values to expected street types.

```
MAPPING = {'block': 'Block', 'Cross': 'Cross', 'Rd': 'Road', 'street': 'Street',
'Crossroad': 'Cross', 'Road)': 'Road', 'Main': 'Main', u'Road,': 'Road',
'Nagar,': 'Nagar', 'cross': 'Cross', 'Cross,': 'Cross', 'Street,': 'Street',
'Road': 'Road', 'road,': 'Road', 'Veedhi': 'Veedhi', 'Block': 'Block',
'Nagar,,': 'Nagar', 'Nagar': 'Nagar', 'Street': 'Street', 'ROad': 'Road',
'Naga': 'Nagar', 'Block,': 'Block', 'Avenue': 'Avenue', 'road': 'Road'}
```

Incorrect phone number formatting

IN India, the phone number is 10 digits long. If it is a landline number, it is 8 digits long, preceded by a 2 digit long state code. All mobile numbers a 10 digits by default. Every number irrespective of mobile or landline is preceded by a country code of +91. The dataset however, had a mixture of various formats of phone numbers. Below is a sample of the various formatting. The entire list is available in the python notebook files attached as part of the project.

```
 ['+91-80-22070713;+91-80-22070709', '+91-80-41139572', '+918022220022', '080
41280614', '+91-80-32718989', '+91-80-41113333', '080 2222 1111', '69886988',
'+91-1-800-102-4353', '+91 80 2220 5205', '+91-80-23634305;+91-80-23430462',
'+91-80-23536658;+91-80-23536517;+91-80-28914493;+91-80-51287718;+91-80-
51289455', '+91-80-25589333;+91-80-51783344;+91-80-25588697', '+918025591800',
'+918022868423', '+91-80-22254111', '080 2220 3333', '+91-80-23561123', '+91-
80-22265544', '+91-80-4037 3737', '+91-80-26705420', '+91-80-22269898', '+91-
80-25551555', '+91-80-41291300', '+91-80-41291300', '+91-80-40441234; +91-80-
39884433', '+91-80-26615865;+91-80-26613237', '+(91)-80-23595565', '080
23423955', '+(91)-80-32466651', '+(91)-80-23131778', '080 2344 9632',
'+918042692727']
```

I used David Drysdale's Python port of Google's libphonenumber library to parse and update the phone numbers to the standard format.

```
import phonenumbers

def update_phone_number(value):
    global val
    val = None

    test = value.split(";")[0]
    # remove unwanted characters and spaces
    test =''.join(x for x in test if x.isdigit())
    # Add the City code of 80
    if len(test) == 8:
        test = '80'.join(test)
    #  For valid 10 digit phone numbers, parse and update the format using the
phonenumbers library
    if len(test) == 10:
        y = phonenumbers.parse(test, "IN")
        val = phonenumbers.format_number(y,
phonenumbers.PhoneNumberFormat.INTERNATIONAL)
    else:
        val = "Unknown"
    return val
```

Varying versions of the city name

Bangalore was officially renamed to "*Bengaluru*" on 1 November 2014 but it is still popularly and unofficially referred to as Bangalore. I saw multiple inconsistencies in the data with respect to the city name. Apart from multiple instances of the older name, there were also occurrences of the state name and other incorrect values.

```
set(['Malleswaram, Bangalore', 'Silver Oak Marg', 'Bengaluru', 'Gandhi Nagar,
Bangalore', 'Bengalore', 'Chickpet, Bangalore', 'Malleshwaram,Bangalore',
'Bengaluru,', 'bengaluru', '560009', 'Bangalore', 'Bengaluru, Karnatak',
'banglore', 'Cottonpet, Bangalore', 'Malleshwaram,2nd Stage,Bangalore', 'V V
Puram, Bangalore', 'Malleshwaram,Bangaloe', 'Shivaji Nagar, Bangalore',
'bangalore', 'Seshadripuram, Bengaluru', 'Gymkhana campus, Indian Institute of
Science', 'Seshadripuram, Bangalore', 'Sheshadripuram, Bangalore', '3rd Main
road palace guttahalli bangalore', 'BANGLORE'])
```

I simply replaced all of it with the official name of Bengaluru using the below code.

```
def update_city(value):
    val = value.replace(value, "Bengaluru")
    return val
```

Inconsistent values for the building tag

The value for the 'building' tag was inconsistent and did not serve much of a purpose in terms of providing information. I found multiple instances a 'yes' and also instances of the tag describing the type of the building. Although the wiki for the tag (http://wiki.openstreetmap.org/wiki/Key:building#Possible_Tagging_Mistakes) allows for yes, I replaced it with unknown and also removed underscores in the data to make it look clean.

```
set(['G/8(A),Swastik_Manandi_Arcade,_No.401/2,_S.C.Road', 'shed',
'industrial', 'office', 'apartments', 'house', 'parking', 'library',
'college', 'church', 'yes', 'service', 'community_centre', 'residential',
'hospital', 'cinema', 'mosque', 'public', 'garage', 'electronics', 'gym',
'publishers', 'transportation', 'commercial', 'company', 'hotel', 'Telephone
Office', 'community_center', 'dentist', 'services', 'bank', 'school',
'astrologer', 'restaurant', 'hockey_stadium_seating', 'university', 'roof',
'temple', 'train_station', 'mall', 'architect', 'retail', 'auditorium'])

## Function to clean and update the building names
def update_building(value):
    val = re.sub("yes", "Unknown", value)
    val = re.sub("_", " ", val)
    return val
```

Overview of the Data

After auditing and cleaning up the data, I shaped it to fit a chosen schema. I then converted the data into to CSV files and used Pandas DataFrames (_to_sql) to insert it into my SQLite database. I also used Pandas DataFrames (read_sql_query) to run SQL queries on the data and derive the below information.

Project: Wrangle OpenStreetMap Data
Apurva Peri

Size of Data

```
OSM FILE ................... 61.2MB
Database ................... 33.4MB
```

Number of Unique Users

```
select count(user) as 'Unique Users'
from (select user from nodes union select user from ways);
```

Unique Users
489

Number of Nodes

```
select count((id)) as 'unique nodes'
from nodes;
```

unique nodes
282985

Number of Ways

```
select count((id)) as 'unique ways'
from ways;
```

unique ways
65883

Number of Each kind of Amenity

Bengaluru is known for its food and variety in cuisines. It is no wonder that restaurants are the most commonly found amenity.

```
select value as  'Amenity', count(*) as  'Number of Occurences'
from  (select value, key from nodes_tags union all select value, key from
ways_tags)
where key = 'amenity'
group by value
order by count(*) desc limit 20;
```

Amenity	Number of Occurrences
restaurant	308
place_of_worship	205
bank	145
atm	118
school	101
parking	100
pharmacy	95
hospital	87

college	84
cafe	75
fast_food	74
university	58
toilets	51
fuel	48
cinema	42
police	33
pub	26
clinic	24
community_centre	22
post_office	21

Top Contributing User

```
select user, count(user) as 'Number of Contributions'
from (select user from nodes union all select user from ways)
group by user
order by count(*) desc limit 10;
```

user	Number of Contributions
saikumard	33746
sdivya	33523
himalay	31653
sathishshetty	28002
PlaneMad	23971
hareesh11	15251
shivajim	14224
harisha	13434
kushwanth	13434
subhashini	13343

Most Popular Religion

Majority of Indians follow Hinduism. The query returned expected numbers.

```
select value as 'Religion', count(value) as 'Number of Practioners'
from (select key,value from nodes_tags union all select key, value from
ways_tags)
where key = 'religion'
group by value
order by count(*) desc limit 10;
```

Religion	Number of Practioners
hindu	112
muslim	38
christian	34
jain	2
buddhist	1

Most Popular Cuisine

Bengaluru has a quite a few local roadside stalls that serve freshly prepared regional food for breakfast and lunch. It also has a large vegetarian community. This is supported by the below queried numbers.

```
select value as 'Cuisine', count(value) as 'Number of Joints'
from (select key,value from nodes_tags union all select key, value from
ways_tags)
where key = 'cuisine'
group by value
order by count(*) desc limit 10;
```

Cuisine	Number of Joints
regional	67
indian	44
vegetarian	22
chinese	12
coffee_shop	9
ice_cream	8
pizza	8
burger	7
italian	6
South_Indian	4

Restaurant with Maximum Branches across the city

Adyar Ananda Bhavan , commonly known as A2B is popular breakfast joint that has multiple branches across the city.

```
select  nodes_tags.value as 'Restaurant', count(nodes_tags.value) as 'Number
of Branches'
from
(select id, value  from (select * from nodes_tags union all select *from
ways_tags) where value = 'restaurant') rst, nodes_tags
where rst.id = nodes_tags.id and nodes_tags.key = 'name'
group by nodes_tags.value
order by count(nodes_tags.value) desc limit 10;
```

Restaurant	Number of Branches
Adyar Ananda Bhavan	4
Mainland China	4
Pizza Hut	3
Adiga's	2
Canteen	2
Chung Wah	2
Chung's	2
Dasaprakash	2
Empire	2
Konark	2

Other Ideas about the Dataset

While there are quite a few area of improvement for the data, I will discuss a few I encountered that I did not correct in this project.

The street types data still needs further thorough cleaning as some names still contain invalid street types and also contain city name, state name and area name which are not relevant to the tag.

Unfortunately, this requires exhaustive cleaning and manual intervention. The data will have to be examined row by row and cleaned.

```
SELECT value
FROM nodes_tags
WHERE key='street';
```

```
Ground Floor Parking, Block,UB City, Concorde, 24, Vittal Mallya Road KG
Halli, Shanthala Nagar Ashok Nagar Bengaluru
No. 76, 5th Cross, 5th Cross Road, 5th Block, Rajaji Nagar, Karnataka
Bellary Road Ganga Nagar Ganga Nagar Dena Bank Colony, Ganga Nagar
4th Main Road 18th Cross, Malleshwaram, Malleshwaram
No.7, Kumara Park East,, Kumara park East Extension, Kumara Park East,
Sheshadripuram
212, My School Road Opposite Narmada Vidyalaya, Near K H Road 4th Cross,
Lalbagh Road C K C Garden, 4th Cross, Vinobha Nagar Sudhama Nagar
31, J C Street J C Street Kalasipalyam New Extension, Kalasipalayam
Laxman Rao, 63, BVK Iyengar Road, Basettypet, Huriopet, Chickpet,
Sri Venkata Nilaya, 1st Cross Road, Jai Bheema Nagar DN Ramaiah Layout,
Guttahalli,
414, Avenue Road, Old Tharagupet, Mamulpet, Nagarathpete,
50/A, Kalasipalyam Main Road, K. R. Market, Kalasipalayam, Bengaluru,
1, P G Lane, Laxmanrao Road B V K Iyengar Road Cross, B V K Iyengar Road
37/2, K V Temple Road Opposite-Charan Bank, Santhusapet, Chickpet
1st Floor, Swastik Complex, Seshadripuram, Seshadripuram, Kumara Park East,
Sheshadripuram,
668/12, 1st Floor Durugal Mutt Road V V Puram, Near-Apex Bank, Chamarajpet,
Chamrajpet,
B.V.K. Iyengar Road
2, 3 Road K V Lane, Akkipet Cross, Akkipet Cross,
386, OTC Road Balepete, Chickpet, Bengaluru, Karnataka
```

The other problem I observed in the dataset is that a lot of the data is not relevant to the tags and not appropriately filled by users. While there is material to educate users (http://wiki.openstreetmap.org/wiki/Tagging_Roads_in_India) the awareness seems to be poor. Also, as seen above, the number of unique users is only 489 which is an abysmally low number for a city as big and cosmopolitan as Bengaluru. The penetration needs to be improved. This can be done by creating incentives and improving awareness. Bengaluru is tech savvy city and has a large pool of users who will be eager to contribute but may not be aware. Initiatives to tap these user will greatly help the data and in terms of quality and accuracy.

In order to improve user base, Openstreetmap might have to create incentives for users but this involves financial challenges and also the most obvious challenge of motivating users to contribute to a mapping initiative other than Google maps.
In order to run an inexpensive ad campaign to raise awareness, Openstreetmap can build partnerships with startup ventures and other corporate events which attracts large user base. Motivating users to use something other than Google is a rather hard challenge because Google maps are very popular in India and used by all classes of the society. However, the fact that The OpenStreetMap Foundation is an international not-for-profit organization unlike Google is an inspiration and may stimulate users to contribute for the greater good.

References

https://www.python.org/dev/peps/pep-0249/

http://www.mapzen.com/data/metro-extracts/

https://pypi.python.org/pypi/phonenumbers

https://github.com/daviddrysdale/python-phonenumbers

https://docs.python.org/2/library/difflib.html

Project: Wrangle OpenStreetMap Data
Apurva Peri

https://www.tutorialspoint.com/python/python_database_access.htm

https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.to_sql.html

https://simply-python.com/2015/01/16/rapid-input-data-from-list-of-files-to-sqlite-db/