DATA WRANGLING

Hyderabad is the capital of the southern Indian state of Telangana and the de jure capital of Andhra Pradesh. I've chosen Hyderabad because, it's one most of the most commercial and developed cities in India.

Reference link for Hyderabad.osm : https://mapzen.com/data/metro-extracts/metro/hyderabad_india/

Questions Explored :

- Firstly, I've identified some street types from the imported data by matching the last name of street name with regular expression function, I've found that many of the last names of the street name were entered wrongly or by using shortcuts (i.e. Street as St or St.etc..) and some street names ended with numerical numbers representing the serial number of respective street name and some ended with white spaces.
- Secondly, since Hyderabad is a district I want to differentiate Hyderabad city from Hyderabad district. This was done using the Postal Codes. Postal Codes ranging Between 500010 and 500070 are considered to be in the city and the remaining are outside the city.

Cleaning and Auditing :

- Street Names :

To deal with correcting street names, I opted to use regular expressions, correcting them to their respective mappings in the update_street_name function. A few of the names post correction are listed below

a) Before: (Municipal No. 15-25-531) Road No-1, Phase No-1, Kukatpally Housing Board Colony,
   After: (Municipal 15-25-531) Road No-1 Phase No-1 Kukatpally Housing Board Colony
b) Before: EFLU
   After: None
c) Before: 9
   After: None
d) Before: Raj Bhavan Rd
   After: Raj Bhavan Road

- Postal Codes :

  In the first iteration of postal code Auditing ,I found and updated the postal codes which has white space characters present in postal codes. In the second iteration,I found that some Postal codes were entered as string and In third iteration I found some postal codes wrongly entered as a string and colon " : " present in the string with help of regular expression("^([a-z]|_)+:") I've updated the both postal codes as None.

  A few of the names post correction are listed below

a) Before : 509218
   After : None
b) Before : 500047
   After : 500047
c) Before : 50004
   After : None
d) Before : 500032
   After : 500032

Preparing for Data Base:

After auditing the data , the cleaned data is exported into respective dictionaries nodes,nodes_tags,ways,ways_nodes and ways _tags in CSV format the sizes of the files are as follows

FILE SIZE

nodes.csv - 262 MB
nodes_tags.csv - 835 KB
ways.csv - 45.7 MB
ways_nodes.csv – 97.3 MB
ways_tags.csv - 27.3 MB

QUERIES USING DATA BASE:

I've created a data base named Hyderabad_india.db with schema as specified and performed the following queries

1. Number of Nodes

In [118]:
```
#number of nodes

QUERY ='''SELECT count(*)as num from nodes'''
cur.execute(QUERY)
all_nodes = cur.fetchall()
df = pd.DataFrame(all_nodes)
print df
```

```
         0
0  3227936
```

There are 3227936 nodes.

2. Number of Ways

In [120]:
```
#number of ways tags
QUERY ='''SELECT count(*)as num from ways'''
cur.execute(QUERY)
all_ways = cur.fetchall()
df = pd.DataFrame(all_ways)
print df
```

```
        0
0  770099
```

There are 770099 way tags.

3. Number of Distinct Users

In [122]:
```
QUERY = '''SELECT DISTINCT(user)
FROM (SELECT user from nodes UNION SELECT user from ways)
GROUP BY user
ORDER BY count(user)
DESC
;

'''
cur.execute(QUERY)
top_unique_users = cur.fetchall()
df = pd.DataFrame(top_unique_users)
print "The number of distinct users:" , len(df)
```

```
The number of distinct users: 1003
```

4. Number of Amenities

In [105]:
```
# number of amenities
QUERY ='''' SELECT value, COUNT(*) as num
FROM (select value , key from nodes_tags UNION ALL select value, key from ways_tags)
WHERE key='amenity'
GROUP BY value
ORDER BY num DESC
;
'''
cur.execute(QUERY)
all_amenities = cur.fetchall()
df = pd.DataFrame(all_amenities)
print "total number of amenities:", len(df)
```

```
total number of amenities: 76
```

5. Top 10 amenities

In [106]:
```
QUERY ='''' SELECT value, COUNT(*) as num
FROM (select value , key from nodes_tags UNION ALL select value, key from ways_tags)
WHERE key='amenity'
GROUP BY value
ORDER BY num DESC
limit 10
;
'''
cur.execute(QUERY)
all_amenities = cur.fetchall()
df = pd.DataFrame(all_amenities)
print df
```

```
                0    1
0  place_of_worship  387
1        restaurant  282
2               atm  264
3              bank  260
4            school  196
5              fuel  167
6          hospital  161
7           parking  117
8          pharmacy  114
9              cafe  101
```

place of worship tops the list with 387 followed by restaurant with 282

Additional Ideas

1. Number of users and contribution values

```
In [84]: #number of users
QUERY ='''' SELECT user,count(user) from( select user from nodes UNION ALL select user from ways)
GROUP BY user
ORDER BY count(user)
DESC;
'''
cur.execute(QUERY)
all_unique_users = cur.fetchall()
import pandas as pd
df = pd. DataFrame(all_unique_users)
print df[1].describe()
print('\n')
print "Total users:" , df[1].sum()
```

```
count      1003.00000
mean       3986.07677
std       16866.43680
min           1.00000
25%           2.00000
50%           7.00000
75%          51.00000
max      144934.00000
Name: 1, dtype: float64


Total users: 3998035
```

- The total number of users are 3998035
- The average number of posts are 3986

2. Top 5 cuisines :

```
In [93]: #top 5 cuisisnes
QUERY ='''' SELECT value, COUNT(*) as num
FROM (select value , key from nodes_tags UNION ALL select value, key from ways_tags)
WHERE key='cuisine'
GROUP BY value
ORDER BY num DESC
LIMIT 5;

'''
cur.execute(QUERY)
all_cuisines = cur.fetchall()
df = pd.DataFrame(all_cuisines)
print df
```

```
            0   1
0      indian  41
1    regional  27
2  coffee_shop  18
3       pizza  14
4     chinese  10
```

3. Number of places :

In [128]:
```
#number of places in hyderabad
QUERY ='''SELECT tags.value, COUNT(*) as count
FROM (SELECT * FROM nodes_tags UNION ALL
      SELECT * FROM ways_tags) tags
WHERE tags.key LIKE '%city'
GROUP BY tags.value
ORDER BY count DESC;'''

cur.execute(QUERY)
all_cITIES = cur.fetchall()
df = pd.DataFrame(all_cITIES)
print "number of places :", len(df)
```

```
number of places : 56
```

4. Most followed religion

In [111]:
```
#most followed religion
QUERY = '''SELECT nodes_tags.value, COUNT(*) as num
FROM nodes_tags
    JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE value='place_of_worship') i
    ON nodes_tags.id=i.id
WHERE nodes_tags.key='religion'
GROUP BY nodes_tags.value
ORDER BY num DESC
LIMIT 1;'''
cur.execute(QUERY)
religion = cur.fetchall()
df = pd.DataFrame(religion)
print df
```

```
      0    1
0  hindu  118
```

Advantages and Disadvantages :

The main advantage is that Open Street Map data is open-source and therefore free to use. This means anyone can use the data to create their own maps (and then use services like Map Box to generate and host customised map tiles). This means the developer doesn't have to work within Google's constraints.

The only imaginable downside to me is quality. Get me right, 99% is the good stuff, but as all crowd sourced data it's hard to maintain consistent quality control. Of course is free to alter and complete the data, but there's no guarantees as there would with a company behind it.

Conclusion:

After series of iterations in Auditing process, I believe that the data has been cleaned precisely and analysed well in exploration phase .