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/

I have created a sample file of the Hyderabad osm by taking the k value as 10.
Size of sample.osm = 70mb (uncompressed)

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:

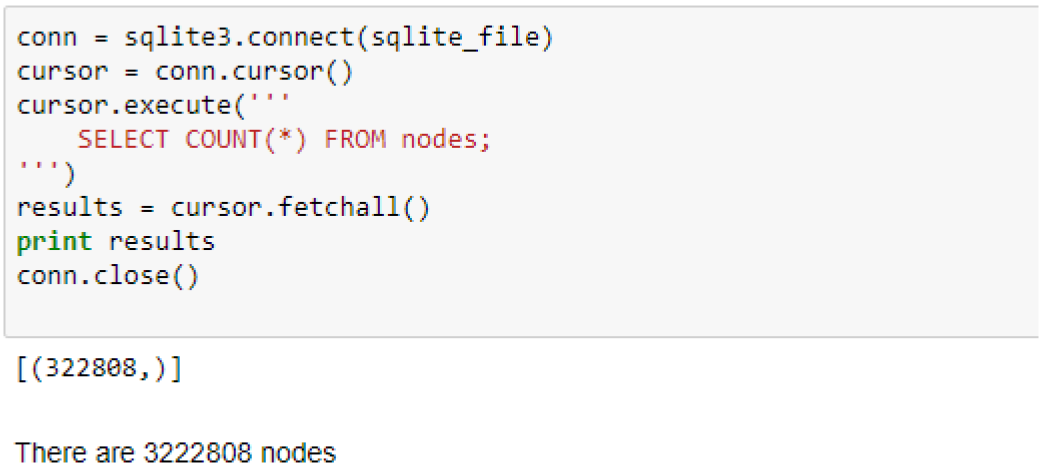
After auditing the data , the cleaned data is exported into respective dictionaries nodes,nodes_tags,ways,ways_nodes and ways _tags in CSV format the sizes of the files are as follows

FILE	SIZE
nodes.csv	26 MB
nodes_tags.csv	88 KB
ways.csv	4.97 MB
ways_nodes.csv	9.73 MB
ways_tags.csv	2.73 MB

QUERIES USING DATA BASE:

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

1. Number of Nodes

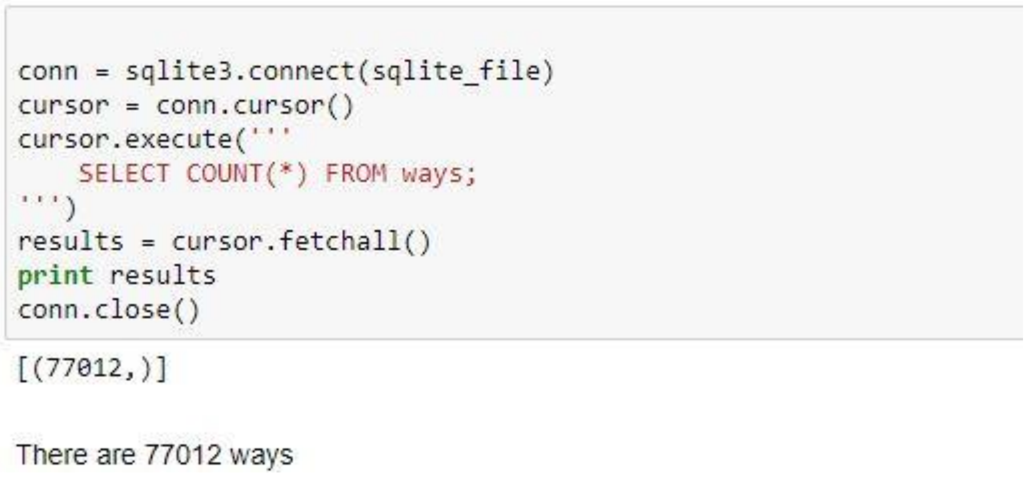


```
conn = sqlite3.connect(sqlite_file)
cursor = conn.cursor()
cursor.execute('''
    SELECT COUNT(*) FROM nodes;
''')
results = cursor.fetchall()
print results
conn.close()
```

```
[(322808,)]
```

There are 3222808 nodes

2.Number of ways



```
conn = sqlite3.connect(sqlite_file)
cursor = conn.cursor()
cursor.execute('''
    SELECT COUNT(*) FROM ways;
''')
results = cursor.fetchall()
print results
conn.close()
```

```
[(77012,)]
```

There are 77012 ways

3. Number of distinct users

```
conn = sqlite3.connect(sqlite_file)
cursor = conn.cursor()
cursor.execute('''
    SELECT Count(DISTINCT both.user)
    FROM (SELECT user FROM nodes
    UNION ALL SELECT user FROM ways) as both;
''')
results = cursor.fetchall()
print results
conn.close
```

```
[(571,)]

<function close>
```

There 571 distinct users.

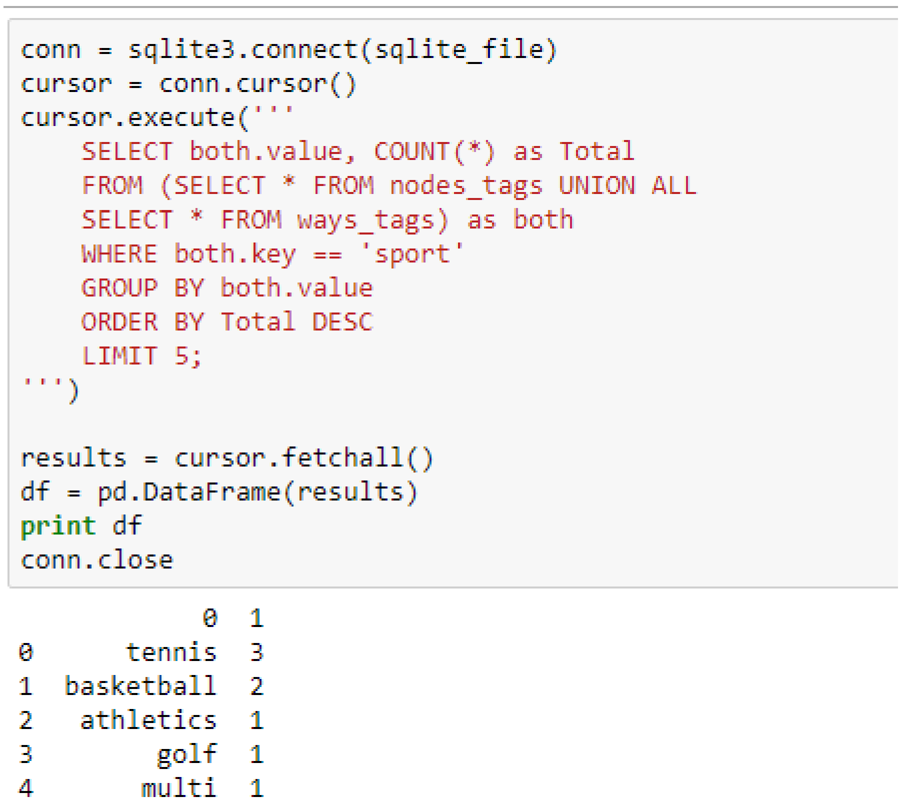
4. Postcodes distribution
500001> Outside city >500070

```
import pandas as pd

conn = sqlite3.connect(sqlite_file)
cursor = conn.cursor()
cursor.execute('''
    SELECT both.value, COUNT(*) as Total
    FROM (SELECT * FROM nodes_tags UNION ALL
    SELECT * FROM ways_tags) as both
    WHERE both.key == 'postcode'
    GROUP BY both.value
    ORDER BY Total DESC
    LIMIT 5;
''')

results = cursor.fetchall()
df = pd.DataFrame(results)
print df
conn.close
```

```
              0   1
0  outside_city  27
1        500038   5
2        500032   4
3        500028   3
4        500047   3
```

Additional Information :

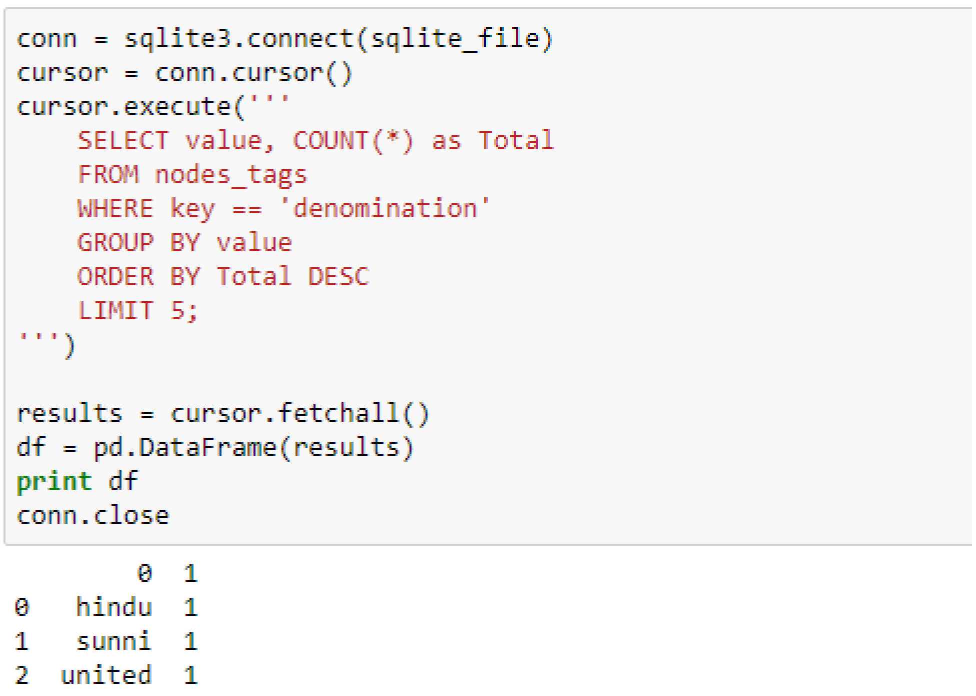
5. Number of users who play particular sports



```
conn = sqlite3.connect(sqlite_file)
cursor = conn.cursor()
cursor.execute('''
    SELECT both.value, COUNT(*) as Total
    FROM (SELECT * FROM nodes_tags UNION ALL
    SELECT * FROM ways_tags) as both
    WHERE both.key == 'sport'
    GROUP BY both.value
    ORDER BY Total DESC
    LIMIT 5;
''')

results = cursor.fetchall()
df = pd.DataFrame(results)
print df
conn.close
```

```
            0  1
0      tennis  3
1  basketball  2
2   athletics  1
3        golf  1
4       multi  1
```

6. Number of users who follow particular religion



```
conn = sqlite3.connect(sqlite_file)
cursor = conn.cursor()
cursor.execute('''
    SELECT value, COUNT(*) as Total
    FROM nodes_tags
    WHERE key == 'denomination'
    GROUP BY value
    ORDER BY Total DESC
    LIMIT 5;
''')

results = cursor.fetchall()
df = pd.DataFrame(results)
print df
conn.close
```

```
        0  1
0   hindu  1
1   sunni  1
2  united  1
```

7. No of Restaurants according to cuisines

```
conn = sqlite3.connect(sqlite_file)
cursor = conn.cursor()
cursor.execute('''
    SELECT both.value, COUNT(*) as Total
    FROM (SELECT * FROM nodes_tags UNION ALL
    SELECT * FROM ways_tags)
    as both
    WHERE both.key == 'cuisine'
    GROUP BY both.value
    ORDER BY Total DESC
    LIMIT 5;
''')

results = cursor.fetchall()
df = pd.DataFrame(results)
print df
conn.close
```

```
            0  1
0    regional  5
1      indian  4
2  coffee_shop  2
3     chicken  1
4   ice_cream  1
```

8. User distribution
Mean=700 is average number of posts.

```
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      571.000000
mean       700.210158
std       2189.332352
min          1.000000
25%          1.000000
50%          4.000000
75%         37.500000
max      14568.000000
Name: 1, dtype: float64


Total users: 399820
```

Other ideas :
When it comes to user Contribution, I'm reminded of "gamification" as an interesting force for contribution within the context of the OpenStreetMap, if user knowledge were a lot of conspicuously displayed, maybe others would take associate degree initiative in submitting a lot of edits to the map. And, if everybody sees that solely a few of power users area unit making over 80-90% a of given map, that may spur the creation of a lot of economical bots, particularly if bound gamification components were gift, like rewards, badges, or a leader board.

Benefits :
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.
OpenStreetMap is a wiki-like map that anyone in the world can edit. If a store is missing from the map, it can be added in by a store owner or even a customer. In terms of display (rendering), each person or company who creates a map is free to render it how they like, but the main map on OpenStreetMap.org uses FLOSS (Free/Libre Open Source Software) rendering software and a liberally licensed stylesheet which anyone can build on.

Issues:
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 it 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.