Data Wrangling with MongoDB

Map Area: Hyderabad,India

https://mapzen.com/data/metro-extracts/
https://www.openstreetmap.org/node/245640543#map=11/17.3621/78.7150

1.Problems Encountered with the Data set:

1. `postal codes are inconsistent:`
   The postal codes in Hyderabad are 6 digit long. But in this data set there are some postal codes are which are less than that. The postal codes which have length less than 6 characters are ignored. Only the postal codes of length equal to six are taken into account. The other problem with postal codes are they are often combined with the city name (i.e it might be represented as “Hyderabad – 500076”)In this case I extracted the last six characters and made it the postal code.
   //given below is the python code for cleaning data
   ```
   if(t[1]=="postcode") and len(child.attrib['v'][-6:])==6 and any (c.isalpha() for c in
   child.attrib['v'][-6:])==False:
                   child.attrib['v']=child.attrib['v'][-6:]
   dic[t[1]]=child.attrib['v']
   ```

2. They are few characters in data set which are not ascii:
   Example:
   <tag k="name" v="Nature Cure Hospital"/>
   <tag k="name:te" v="ప్రకృతి చికిత్సాలయ రైల్వే స్టేషన్"/>
   Here the same name is represented in different languages for the convenience of the users so instead of storing the redundant information multiple times in the data base I stored only one key value pair which is in English and moreover both of them denote the same information. The other key value pair is ignored and is not processed in python during generation of Jason file. But they are other attributes like user which are only represented in other languages.
   Example:
   <node id="343318352" lat="17.4455638" lon="78.4515418" version="4" timestamp="2013-08-01T15:47:16Z" changeset="17181354" uid="179403" user="వీవెన్"/>
   In this the value of the user is stored as it is in the database.

3. Most of the nodes and ways in the data are empty and incomplete. i .e They do not have any information of the name of the node or the other attributes that describe the node in much greater detail. Most of the nodes and ways do not contain the child attributes they only contain some essential information such as the location and the node id and the name of the contributed user. The nodes which are populated are some of the most popular places in Hyderabad. Other places are not described in much detail.

2.Data Overview

File sizes

hyderabad_india.osm	688.3 MB
hyderabad_india.osm.json	973.2 MB

I have used only nodes and ways which are present in xml file.

Number of documents:

```
    > db.booker.find().count()
    3762029
```

Number of nodes:

```
    > db.booker.find({"type":"node"}).count()
    3038027
```

Number of ways:

```
    > db.booker.find({"type":"way"}).count()
    724002
```

Top 5 user:

```
    >db.booker.aggregate([{"$group":{"_id":"$created.user","count":{"$sum":1}}
    },{"$sort":{"count":-1}},{"$limit":5}])
    { "_id" : null, "count" : 724002 }
    { "_id" : "Rajkumarmishra", "count" : 130660 }
    { "_id" : "venkatkotha", "count" : 110075 }
    { "_id" : "anushapyata", "count" : 100469 }
    { "_id" : "harisha", "count" : 100054 }
```

Number of restaurants in hyderabad:

```
    > db.booker.find({"amenity":"restaurant"}).count()
    189
```

Number of oneways in Hyderabad:

```
    > db.booker.find({"oneway":"yes"}).count()
    2151
```

Number of nodes which contain attribute “place”:

```
    > db.booker.find({"place":{"$exists":1}}).count()
    1248
```

Number of nodes which have substring “sub” in place:

```
    > db.booker.find({"place":{"$regex":"sub"}}).count()
    176
```

Number of ways which have references of NH163 or NH 22:

```
    > db.booker.find({"type":"way","ref":{"$in":["NH163","NH22"]}}).count()
    61
```

Number of restaurants grouped by postal code:

```
    >db.booker.aggregate([{"$match":{"amenity":"restaurant"}},{"$group":{"_id"
    :"$address.postcode","count":{"$sum":1}}},{"$sort":{"count":-1}}])
    { "_id" : null, "count" : 179 }
    { "_id" : "500032", "count" : 2 }
    { "_id" : "500072", "count" : 2 }
    { "_id" : "500081", "count" : 1 }
    { "_id" : "500038", "count" : 1 }
    { "_id" : "500011", "count" : 1 }
    { "_id" : "500062", "count" : 1 }
    { "_id" : "500015", "count" : 1 }
    { "_id" : "500033", "count" : 1 }
This shows that out of 189 restaurants only 10 contain postalcode attribute
all the others do not contain any postal code
```

Number of amenities grouped by postal code:

```
    >db.booker.aggregate([{"$match":{"amenity":{"$exists":1},"address.postcode
    ":{"$exists":1}}},{"$group":{"_id":"$address.postcode","count":{"$sum":1}}
    },{"$sort":{"count":-1}},{"$limit":5}])
    { "_id" : "500047", "count" : 14 }
    { "_id" : "500028", "count" : 7 }
    { "_id" : "500049", "count" : 4 }
    { "_id" : "500062", "count" : 4 }
    { "_id" : "500055", "count" : 3 }
```

Biggest religion:

```
    >db.booker.aggregate([{"$match":{"religion":{"$exists":1}}},{"$group":{"_i
    d":"$religion","count":{"$sum":1}}},{"$sort":{"count":-1}}])
    { "_id" : "hindu", "count" : 93 }
    { "_id" : "muslim", "count" : 25 }
    { "_id" : "christian", "count" : 18 }
    { "_id" : "sikh", "count" : 1 }
```

Different places in Hyderabad by their count.

```
>db.booker.aggregate([{"$match":{"place":{"$exists":1}}},{"$group":{"_id":"$pl
ace","count":{"$sum":1}}},{"$sort":{"count":-1}}])
{ "_id" : "neighbourhood", "count" : 984 }
{ "_id" : "suburb", "count" : 176 }
{ "_id" : "village", "count" : 40 }
{ "_id" : "locality", "count" : 39 }
{ "_id" : "town", "count" : 3 }
{ "_id" : "hamlet", "count" : 3 }
{ "_id" : "yes", "count" : 2 }
{ "_id" : "city", "count" : 1 }
```

The above query shows that it not only includes Hyderabad but it includes all the places which come in greater Hyderabad.

Additional Ideas:

We can use other api's such as google's geo coding api to verify the correctness of the postal code based on the given latitude and longitude. In our case every node has latitude and longitude component if the postal code attribute in the address is missing or incomplete the we can populate it with the results which api returns, it can also be used for validation.

Example:
Reverse Geocoding for a Latitude/Longitude
The following query contains the latitude/longitude value for a location in Brooklyn:
[https://maps.googleapis.com/maps/api/geocode/json?latlng=40.714224,-73.961452&key=**YOUR_API_KEY**](https://maps.googleapis.com/maps/api/geocode/json?latlng=40.714224,-73.961452&key=YOUR_API_KEY)

It returns the following results:

```
{
  "results" : [
    {
      "address_components" : [
        {
          "long_name" : "277",
          "short_name" : "277",
          "types" : [ "street_number" ]
        },
        {
          "long_name" : "Bedford Avenue",
      "short_name" : "Bedford Ave",
      "types" : [ "route" ]
    },
    {
      "long_name" : "Williamsburg",
      "short_name" : "Williamsburg",
      "types" : [ "neighborhood", "political" ]
    },
    {
      "long_name" : "Brooklyn",
      "short_name" : "Brooklyn",
      "types" : [ "sublocality", "political" ]
    },
    {
      "long_name" : "Kings",
      "short_name" : "Kings",
      "types" : [ "administrative_area_level_2", "political" ]
    },
    {
      "long_name" : "New York",
      "short_name" : "NY",
      "types" : [ "administrative_area_level_1", "political" ]
    },
    {
      "long_name" : "United States",
      "short_name" : "US",
      "types" : [ "country", "political" ]
    },
    {
      "long_name" : "11211",
      "short_name" : "11211",
      "types" : [ "postal_code" ]
    }
  ],
  "formatted_address" : "277 Bedford Avenue, Brooklyn, NY 11211, USA",
  "geometry" : {
    "location" : {
      "lat" : 40.714232,
      "lng" : -73.9612889
    },
    "location_type" : "ROOFTOP",
    "viewport" : {
      "northeast" : {
        "lat" : 40.7155809802915,
        "lng" : -73.9599399197085
      },
          "southwest" : {
            "lat" : 40.7128830197085,
            "lng" : -73.96263788029151
          }
        }
      },
      "place_id" : "ChIJd8BlQ2BZwokRAFUEcm_qrcA",
      "types" : [ "street_address" ]
    },

  ... Additional results[] ...
```

We can parse the response and get only the postal codes for the given latitude and longitude.

They are few additional problems encountered while implementing it they are:
1.If the api returns a list of postal codes instead of a single one then in that case if the postal code is missing we are not sure what to populate in the postal code attribute of the address field if it returns only a single postal code we can directly use it and populate the missing or incomplete postal code.
Whereas if it is used in validation we can check if the postal code is present in the list of postal code which it returns.

Conclusion:

It is clear that the number nodes and ways present in the file are incomplete. The map not only has the nodes in Hyderabad but also the nodes and ways from places which are near Hyderabad. This is often called greater Hyderabad. Most of the nodes/ways which are present in the map are not contributed by anyone. The number of nodes/ways which do not have any user contribution is about 724002. We could make sure that the information entering the map is cleaned and of good quality by taking the help of gps and if the information is entered by user we can make sure that it is properly formatted by writing some checks in the code. These checks can vary from place to place based on the information obtained by consulting government authorities or other open sources.