OpenStreetMap Data Wrangling with SQL

The location we are wrangling is **New Delhi** the capital city of India and the place currently I am living in.

I uses the overpass api (http://overpass-api.de/query_form.html) to export the location data. My query is down below.

```
(node(28.6142,77.2023,28.6983,77.3767);
<;
);
out meta;
```

the output file name is **NewDelhi.osm** whose small sample is in the repository.

Auditing the OSM file

1. This step is performed to gather the general information about the tags.

First we see how many different types of tags occurred how many times.

Find the Code in: tags_type.py

```
member  : 10435
meta    : 1
nd      : 300875
node    : 241026
note    : 1
osm     : 1
relation: 603
tag     : 61147
way     : 47148
```

Then I categorized the tags who are name **tag** in three categories based on their key value.

- **lower:** The keys with lower characters.
- **lower_colon:** The keys with lower characters and colon **(:)**.
- **problemchars:** The keys with special characters like **#,$,@** etc.
- **others:** the keys with rest of other types of values.

Then I check the occurrences of each type of key in tags. The result is below:

```
lower        : 59021
lower_colon  : 2071
problemchars : 22
other        : 33
```

2. In this step we see the problems we encountered in the osm file

Find the Code in: tags_type.py

There are so many types of location to look up for auditing like **house no, street address, ameneties, shops.** I choose to audit street names as it needed so many corrections.

The first problem I found in many places the city name written in wrong format. So I updated it with the more suitable form.

these are the two most common examples of that

- `delhi => Delhi`
- `Delhi. => Delhi`

Another problem is names in hindi which may be difficult to understand for a non hindi speaker. So I updated it with their english meanings.

- `Bagh => Park`
- `Marg => Road`
- `Chowk => Open Market`
- `Bazaar => Market`
- `Nagar => town`

  Then there are abbreviations which needed to be updated with the full word.

- `Ln => Lane`
- `Rd. => Road`

Then the words with lower cases and misspellings.

- `cicus => Circle`
- `lane => Lane`
- `gate => Gate`

Cleaning the OSM file and load Into DB

Find the Code in: data.py

In this part I gather the data in a certain structure which is required to write in to a csv and then to DB.

During the data structuring process I categorise the tags(tags/ways) in to three categories we make during auditing.

- **lower_colon:** The keys with lower characters and colon **(:)**.
- **problemchars:** The keys with special characters like **#,$,@** etc.
- **others:** the keys with rest of other types of values.

  These categories are defined to give the tags a particular **type** and **key**. The **others** tags get categorised as **'regular'**. The **problemchars** tags will be ignored. The **lower_colon** tags gets the type the value before the colon(:) and key the value before the colon(:).

  e.g. if key attribute in tag has value **add:street** then the type will be **add** and key will be **street**

  In the **lower_colon** tags , the values attributed will get updated based on the key it associated with.

  - If tags is of type street then we use the **audit.py** function **update_name** to update the street name.
  - If tags is of type postcode the we check weather the postal code is a correct postal code or not.If the postcode is correct the it went as it is otherwise it went as 'null'.

  After Structuring the data We use the csv dictwrite to write into the **csvs** and then to **DB** as per the required schema.

Data overview of files

```
NewDelhi.osm ......... 53.404 MB
NewDelhi.db .......... 28.490 MB
nodes.csv ............. 19.966 MB
nodes_tags.csv ........ 0.185 MB
ways.csv .............. 2.858 MB
ways_tags.csv ......... 1.801 MB
ways_nodes.csv ......... 7.346 MB
```

SQL Queries

No of unique users

```
SELECT COUNT(DISTINCT(e.uid))
FROM (SELECT uid FROM nodes UNION ALL SELECT uid FROM ways) e;
```

Output:

483

No of Nodes:

```
SELECT COUNT(*) FROM nodes;
```

Output:

241026

No of Ways:

```
SELECT COUNT(*) FROM ways;
```

Output:

47148

No of Shops:

```
SELECT COUNT(*) as count
FROM (SELECT * FROM nodes_tags
      UNION ALL
      SELECT * FROM ways_tags) e
      where e.key="shop";
```

Output:

99

Most common type of shops:

```
SELECT e.value as SHOPS,COUNT(*) as num
    FROM (SELECT * FROM nodes_tags
          UNION ALL
               SELECT * FROM ways_tags) e
    WHERE e.key="shop"
    GROUP BY SHOPS
    ORDER BY num DESC
    LIMIT 5;
```

Output:

```
SHOPS        | NUM
bakery       | 16
clothes      | 13
```

```
electronics  | 8
supermarket  | 7
books        | 5
```

Top 3 Amenities:

```
SELECT value, COUNT(*) as num
FROM nodes_tags
WHERE key='amenity'
GROUP BY value
ORDER BY num DESC
LIMIT 3;
```

Output:

```
Amenities       | Count
Resturant       |87
Atm             |40
Place_of_worship|37
```

No of valid postcodes:

```
SELECT COUNT(*) as count
FROM (SELECT * FROM nodes_tags
      UNION ALL
      SELECT * FROM ways_tags) tags
WHERE tags.key='postcode'
AND tags.value <> 'null';
```

Output:

180

List Unique postcodes in NewDelhi:

```
SELECT tags.value, COUNT(*) as count
FROM (SELECT * FROM nodes_tags
      UNION ALL
      SELECT * FROM ways_tags) tags
WHERE tags.key='postcode'
AND tags.value <> 'null'
GROUP BY tags.value
ORDER BY count DESC;
```

Output:

```
POSTCODE| COUNT
100006  |59
110001  |22
```

```
110055  |21
110002  |12
110006  |10
110063  |10
110003  |5
110005  |5
110015  |5
110053  |5
110054  |4
110008  |3
110011  |3
110092  |3
110007  |2
110021  |2
110026  |2
110035  |2
201301  |2
110010  |1
110060  |1
110064  |1
```

Listing the names of Metro Stations

As NewDelhi is one of the most dense cities in India it has a metro railway service, almost 2 million people travel with metro daily. It comes under the department DMRC(Delhi Mertro Rail Corporation).

```
SELECT Distinct(nodes_tags.value) StationNames
FROM nodes_tags
    JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE value='DMRC') i
    ON nodes_tags.id=i.id
WHERE nodes_tags.key='name';
```

Output:

```
StationNames

Rajouri Garden
Ramesh Nagar
Patel Nagar (East)
Pratap Nagar
Pul Bangash
Kashmere Gate
Rajendra Place
Karol Bagh
Jhandewalan
Kirti Nagar
Shadipur
Moti Nagar
New Delhi Metro Station Gate 1
Udyog Bhavan
```

```
Patel Chowk
Ramakrishna Ashram Marg
Pragati Maidan
Mandi House
Indraprastha
ONGC Shivaji Stadium
Barakhambha Road
New Delhi
Chawri Bazaar
Yamuna Bank
Chandni Chowk Metro Station
Civil Lines
Tis Hazari
Shivaji Park
Paschim Vihar East
Subhash Nagar
Shastri Park
Seelampur
Shastri Nagar
Inderlok
Laxmi Nagar
Tagore Garden
Chandni Chowk - Gate 1
Khan Market
Rajiv Chowk
Rajiv Chowk - Gate 6
New Delhi Metro Station Airport line
New Delhi Airport Express Terminal
Central Secretariat
Punjabi Bagh East
Satguru Ramsingh Marg
Ashok Park Main
Janpath
Gate 2
Delhi Cantonment
Mayapuri
ESI Hospital
Punjabi Bagh West
Welcome
New Delhi Metro Station Gate 3
New Delhi Metro Station Gate 2
New Delhi Metro Station Gate 4
```

Listing The Tourist Attraction

As every body know Delhi is famous for its tourist attractions, we are going to list these.

```
SELECT Distinct(nodes_tags.value) Attraction
FROM nodes_tags
    JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE key='tourism' AND value='a
ttraction') i
```

```
    ON nodes_tags.id=i.id
WHERE nodes_tags.key='name';
```

Output:

```
Attractions

11 Murthi
Teen Murti
Rashtrapati Bhavan (Presidential Palace)
Police Memorial
Jantar Mantar
Diwan-e-Aam
Purana Qila
Jaipur Column
Khooni Darwaza
Mutiny Telegraph Memorial
Jama Masjid
India Gate
Mystery Rooms
MLA office
Punjabi Bagh Chowk
Chandni Chowk Market
Jantar Mantar Entry
```

Major Religions

We can deduce major religions by counting the places of worship in the entire city based on their religion.

```
SELECT nodes_tags.value RELIGION, 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;
```

Output:

```
RELIGION    num
hindu        11
muslim       8
sikh         3
christian    1
jewish       1
zoroastrian  1
```

Listing out the tourism hotels with their websites

It will be a inner join as left join list those hotels also which doesn't have a website.

```
SELECT hotel.value HOTEL, website.value WEBSITE
FROM (SELECT * FROM nodes_tags
        WHERE id in (SELECT DISTINCT(id)
            FROM nodes_tags
            WHERE key='tourism' AND value='hotel')
        AND key='name') hotel
    JOIN
    (SELECT * FROM nodes_tags
        WHERE id in (SELECT DISTINCT(id)
            FROM nodes_tags
            WHERE key='tourism' AND value='hotel')
        AND key='website') website
    ON
    hotel.id = website.id;
```

Output:

```
HOTEL                       WEBSITE
The Ambassador              http://www.vivantabytaj.com/Ambassador-New-Delhi/Overv
iew.html
Claridges Hotel             http://www.claridges.com/index.asp
Hare Krishna Guest House    http://www.hotelharekrishna.com/
Maidens Hotel                      www.maidenshotel.com
Ajanta                      http://www.ajantahotel.com
Hotel Perfect               http://www.hotelperfect.co.in/
Hotel Durga International Dx http://www.hoteldurgainternational.co.in
Hotel Lal's Haveli          http://hotellalhaveli.com
Hotel City Star                    www.hotel-citystar.com
Amax Inn                    http://www.hotelamax.com/
Bloomrooms                  http://bloomrooms.com/hotels-railwayst.php
Smyle Inn                   http://www.smyleinn.com
Shangri-La's Eros Hotel     http://www.shangri-la.com/newdelhi/erosshangrila/
the spot                           www.hotelthespot.in
```

Conclusion

Achived and Benifits:

I think the data set(OSM file) has a relsonable amount of data, but with lot of wrong street names like the abriviations, misspellings, language etc. which all I have cleaned. I checked and ignored the tags with problematic charachters. I cheked for the valid post code. Then I did a resonable amount of qeury to get the most of the data I entered into the database. I think the data of New delhi is quite competable to the google maps as compared to any other metropolitan city in India which has very small data in OSM as compared to google maps.

Problems Encountered

The New Delhi metro is big, I mean it has more than 160 stations and still building and it is divided in to 8 color lines.

What I wanted to achive is to list all the stations along with there color lines. But the problem is there are very few(around 55) metro stations in the data, not every one is with their tag having the key name color, all of stations which have a tag with key color have there value **yellow(mostly)** and for rest of the tags the **hexadecimal** value of color is given like for red line metro station the tag is someting like this.

```
<tag k="colour" v="ff0000"/>
```

Solution Sugesstion for the above problem

We can restrict the user when it is entering a metro station data, it must be with the color value. It would be best if the user write the actual color rather than the hexadecimal value.

If there are **hexadecimal** value in the color tag, then on the cleaning step we can build a mapper which can map hexadecimal value to one of the eight color of the Delhi Metro.

Benifits and Issues from of the solution

Benifits

- We will be able to tell which metro station belong to which color.
- A color given for a metro station can help us answer many question which color line has the maximum station and vice versa, which color metro is near to which part of delhi etc.

Issues

- Restricting the user to fill up extra fields like color may annoy user or user may end up not entering the data at all.
- If there are a hundreds of different hexadecimal value entered by the users, it would be difficult to build a mapper which can map each hexadecimal value to one of the eight color or it will be really time consuming.

Resources Used in the Project

- Udacity Forums already answered question.
- This link (https://stackoverflow.com/questions/33865525/indian-pincode-validation-regex-only-six-digits-shouldnt-start-with-0) about postcode validation.
- Sqlite (http://sqlite.org/docs.html) documentation.
- Sample Project ReadMe