Data Wrangling with SQL

OpenStreetMap Sample Project

Location →

Vancouver Canada (https://mapzen.com/data/metro-extracts/metro/vancouver_canada/)
http://www.openstreetmap.org/relation/1852574

Problems encountered in the Map

- Incorrect Postcodes - Some data points have an extra space in the zip code that needs to be corrected)
  - Postcodes are generally represented as 'V5V 4E6' , while in some cases it is recorded as 'V5K3K3', with no spacing
- Incorrect Street Names - Some Data Points have incorrect street names
  - Example, in some cases the street name is captured as 'E 29th Ave. at Slocan St.' instead of 'E 29th Ave. at Slocan Street' or 'Mt Seymour Pky' instead of 'Mt Seymour Parkway'
  - Same problem is in Highway Names as well
- Inconsistent Phone Numbers - There are multiple formats for phone number some time having '-', '.' or '()' in between. Standardising it to have only numbers
- Inconsistent House Numbers - In some data points, the house number start with '#' followed by the number. Standardising the same to have only numbers.
- Inconsistent Province Name, Both 'BC' and 'British Columbia' are used. Standardizing it to use 'British Columbia'

Data Overview

The total file size of the osm is 183MB, with sample file created to test the code of about 62 MB.
To generate sample, the script genetate_samply.py is used
Complete OSM and sample files can be found here

File	Size
Vancouver_canada.osm	183MB
sample_vc.osm	62MB

Auditing the Data

Map Tags

```
def get_element(osm_file, tags=('node', 'way', 'relation')):

    context = ET.iterparse(osm_file, events=('start', 'end'))
    _, root = next(context)
    for event, elem in context:
        if event == 'end':
            yield elem
            root.clear()
def count_tags(filename):
    '''
    Initial Function to get a sense of data, how is the data structured.
    '''
    tags = {}
    for element in get_element(filename):
        if element.tag not in tags.keys():
            tags[element.tag] = 1
        else:
            tags[element.tag] += 1
    return tags
```

Tags	Number (Main File)	Number (Sample File)
‘tag’	273781	91073
‘member’	12169	4154
‘osm’	1	1
‘relation’	1725	575
‘bounds’	1	-
‘node’	806374	268792
‘way’	156252	52084
‘nd’	1000503	333910

For Cleaning the data, of the issues identified above, I have done the audit, cleaning and updation in the data_cleaning.py script

Loading Data in Tables

As part of data_cleaning.py script the cleaned data is loaded into the csv files namely -

- nodes.csv
- nodes_tags.csv
- ways.csv
- ways_nodes.csv
- ways_tags.csv

After creation of csv in the csv_sql.py script the data is read from csv files and loaded onto the sqllite database.

Analysing with SQL

When using the sample_vc.osm file, the stats for each table came out as

Table Name	Number of Records
Nodes	268792
Nodes_tags	12209
Ways	52084
Ways_nodes	333910
Ways_tags	77162

Number of Node : 268,792
Number of Ways : 52,084
SQL Queries

```
sqlite> select count(1) from nodes;
268792
sqlite> select count(1) from nodes_tags;
12209
sqlite> select count(1) from ways;
52084
sqlite> select count(1) from ways_nodes;
333910
sqlite> select count(1) from ways_tags;
77162
```

- **Getting total number of contributors(users)**

```
select count(distinct a.user)
from
(select user from nodes union select user from ways) a
where a. User != ‘-999’;
```

Result →

```
613
```

- **Getting the number of edits done by a single user**

```
select a.user, count(1)
from
(select user from nodes union all select user from ways) a
where a.user != '-999'
group by user
order by count(1) desc
limit 20;
```

Result →

```
keithonearth|111483
michael_moovelmaps|37704
still-a-worm|32398
treeniti2|24997
pdunn|13922
muratc3|12357
WBSKI|10456
rbrtwhite|7453
Siegbaert|7129
pnorman|6643
MetVanRider123acme|6047
pnorman_mechanical|5004
mame-stgt|4241
Nihat|3267
fmarier|2335
z-dude|2262
Adam Dunn|2166
David Metcalfe|1694
Spacecookies|1675
mattropolis|1337
```

- **Analysing the amenities**

```
SELECT value, count(1)
FROM nodes_tags
WHERE key = "amenity"
GROUP BY value
HAVING count(1) >= 15
ORDER BY count(1) DESC;
```

```
bench|264
restaurant|230
bicycle_parking|127
cafe|125
fast_food|121
post_box|79
toilets|46
bank|45
waste_basket|38
bicycle_rental|36
pub|25
drinking_water|22
parking|19
pharmacy|19
bar|17
car_sharing|17
fuel|15
```

- **Number of Highways**

```
select key, count(1)
from ways_tags
where key = 'highway'
group by key
```

```
highway|6555
```

- **Analysing where maximum residential areas are**

```
select key, value, count(1)
from ways_tags
where value = 'residential'
group by key,value
order by count(1) DESC
```

```
limit 15;
```

```
highway|residential|1311
building|residential|604
landuse|residential|205
construction|residential|1
```

- **Instances where node and ways data(values) is exactly the same**

```
select count(1)
from nodes_tags a, nodes b, ways_tags c, ways_nodes d, ways e
where a.id = b.id
and c.id =e.id
and a.id = d.node_id
and d.id =e.id
and a.key = c.key
and a.value = c.value
```

```
33
```

- Analysing most popular cuisines

```
select value, count(1)
from nodes_tags
where key = 'cuisine'
group by value
order by count(1) desc;
```

```
coffee_shop|20
japanese|20
chinese|18
burger|14
pizza|14
vietnamese|11
sushi|10
italian|9
asian|8
sandwich|8
indian|6
mexican|5
thai|4
french|3
malaysian|3
```

```
regional|3
Global|2
Vietnamese|2
chinese;asian|2
international|2
```

- **Maximum ways for which nodes**

```
select a.key,count(c.key)
from nodes_tags a, nodes b, ways_tags c, ways_nodes d, ways e
where a.id = b.id
and c.id =e.id
and a.id = d.node_id
and d.id =e.id
and a.key = c.key
and a.value = c.value
group by a.key;
```

```
bicycle|7
building|2
city|4
cycleway|1
destination|1
foot|5
housenumber|2
is_in|3
postcode|1
source|2
Street|5
```

Additional Ideas

- Further Cleaning can be done, as many of the data points are not in english language (eg - chinese), we can translate those into english and perform further analysis on this
- There are some data discrepancies , such as house number being part of street address.
- We can further validate this data using google maps api and fill the missing information
- We can run analysis on amenities not only on key values but also on ‘values’ based on the name of establishment.

- For the above requirement we would need much more polished data sets or would need to be cleaned further
- Eg - the following query would give much more results that of amentities query performed above

```
select key,value
from nodes_tags
where value like '%Shop%'
or value like '%diner%';
```

```
name|Shoppers Drug Mart
cuisine|coffee_shop
cuisine|coffee_shop
cuisine|coffee_shop
cuisine|coffee_shop
name|Cornerstone Coffee Shop
name|Shoppers Drug Mart
name|The Pet Shop Boys.ca
website|thepetshopboys.ca
name|Dundarave Print Workshop + Gallery
website|www.dundaraveprintworkshop.com
cuisine|coffee_shop
cuisine|coffee_shop
name|Shoppers Drug Mart
name|Cookshop
website|www.cookshop.ca
```

- We can also do analysis on websites if needed instead of treating them like normal amenities.

Conclusion

From Data wrangling of Vancouver Canada, I found multiple data issues in pin code, street names, province. Which were fixed as part of auditing and cleaning.
While analysing I found that around 613 users contributed to the open street map.
There are multiple amenities in the area which are well connected by multiple ways (highways, streets).

As noted above we can perform even further analysis on this data if needed for better understanding.