OpenStreetMap Data Case Study

Map Area-:

The place which I have considered for the case study is **Calgary,Alberta, Canada**.

https://www.openstreetmap.org/relation/3227127

https://mapzen.com/data/metro-extracts/metro/calgary_canada/

I chose Calgary because it is a city I hope to visit some day, and I'm interested in what the open street map data would reveal about the city once I begin querying.

Auditing

After analysing the data from the google open street map for the city of Calgary,I could find many problems within the XML data. I have listed below the problems I have decided to recify-:

1. Fix the postal codes to the correct Canadian format.
2. Format the web URL provided by shops in the open street map data.
3. Convert phone numbers to the proper international format.
4. Audit the house numbers entered for consistency.
5. Correct the various street types.

Postal Code

The proper canadian postal code formate is **A1A 1A1.** Some of the postal codes entered in the xml data are not of the mentioned format. Cleaning up this data is required.

```
import re
POSTCODE = re.compile(r'[A-z]\d[A-z]\s?\d[A-z]\d')
def is_street_name(elem):
    return (elem.tag == "tag") and (elem.attrib['k'] == "addr:postcode")

def post_code():

    for event, elem in ET.iterparse('calgary_canada.osm'):
        if is_street_name(elem):
            post_code=elem.attrib['v']
            m=POSTCODE.match(x)
            if m is not None :
                #Checking for spaces
                if " " not in x:
                    #Fixing the format
                    post_code=(post_code[:3] + " " + post_code[3:]).upper()
                else:
                #Printing out the rest to analyze
                    print ('Defects:')
                    print (x)
post_code()
```

Defects-:

```
Defects:
AB T2S 2N1
Defects:
AB T2G 2L2
Defects:
T3K-5P4
Defects:
403-719-6250
Defects:
T2E
Defects:
1212
Defects:
2J 2V9
```

As visible from the above code a few postal codes are defects. Some of them can be cleaned out but certain ones like the '403-719-6250' ,'1212' etc. are incorrect entries in the field. The **'403-719-6250'** seems to be a phone number and is an example of column shift where an entry suitable in another field is placed in the wrong field. Postal codes like these can be ignored.

```
#Correcting postal codes with 'AB'
if 'AB' in post_code:
    post_code=post_code.strip('AB ')
#Correcting Post codes with '-'
if '-' in post_code:
    post_code=post_code.replace('-',' ')
else:
    #Ignoring the rest
    post_code= None
```

Thus the problems in the postal code fields have been corrected.

Web URL

Certain shops list their web address along with other descriptions. While most of the web urls are represented in the **"https://www.xxxx.com"** format, some urls start without **'https://'** and some don't even have **'www.'** in front. Auditing these for consistency is necessary.

```
: url_1=re.compile(r'http',re.IGNORECASE)
def is_street_name(elem):
    return (elem.tag == "tag") and (elem.attrib['k'] == "website")

def audit():

    for event, elem in ET.iterparse('calgary_canada.osm'):
        if is_street_name(elem):
            website=elem.attrib['v']
            if url_1.match(website)== None:
                if 'www.' in website:
                    website= 'https://' + website
                    print(website)
                else:
                    website='https://www.' + website
                    print(website)

audit()
```

Once the above code is executed, the incorrect urls are printed in the http://www.xxxx.com format.

```
https://www.timhortons.ca
https://www.starbucks.ca
https://www.sky360.ca
https://www.stjamesparishrc.com
https://www.dollarama.ca
https://www.bikebike.ca/
https://www.pedalhead.ca
https://www.ilsogno.org
https://www.troyshoppejewellers.com
https://www.lukesdrugmart.com
https://www.protospace.ca
https://www.teraenv.com
https://www.enchantedfloristltd.ca
https://www.nainaskitchen.com
https://www.boogiesburgers.com
https://www.cliveburger.com
https://www.callebautchocolates.ca
https://www.infovet.ca/dalhousiestationvet/
https://www.skylarklimos.ca
```

Phone number

Here I have carried out the conversion of phone numbers entered into the **+1-XXX-XXXX** format

- 4169871234 to +1-416-987-1234

```
In [22]: import xml.etree.cElementTree as ET
         import re
         phone_match = re.compile(r'\+1-\d{3}-\d{3}-\d{4}')
         def is_street_name(elem):
             return (elem.tag == "tag") and (elem.attrib['k'] == "phone")

         def audit():
             for event, elem in ET.iterparse('calgary_canada.osm'):
                 if is_street_name(elem):
                     phone_number=elem.attrib['v']
                     #Looking for match with regular expression
                     m=phone_match.match(phone_number)
                     if m is None:
                         #Removing characters other than numbers
                         phone_number=re.sub('[^0-9]+', '', phone_number)
                         #Checking if the resulting phone number is not moe that 11 characters
                         if len(phone_number) not in range(10,12) :
                             phone_number=None
                         else:

                             if phone_number[0]=='1':
                                 #Formatting if the number starts with '1'
                                 phone_number='+1-'+ phone_number[1:4] + '-' + phone_number[4:7] + '-' + phone_number[7:]
                                 print(phone_number)

                             else:
                                 #Formatting if the number starts with any other number
                                 phone_number='+1-'+ phone_number[0:3] + '-' + phone_number[3:6] + '-' + phone_number[6:]
                                 print (phone_number)
         audit()
```


The above code converts the phone numbers entered into the required format.

```
+1-403-547-5020
+1-403-931-3633
+1-403-275-6577
+1-403-299-2600
+1-403-242-2222
+1-403-288-6033
+1-403-228-5553
+1-403-270-3780
+1-403-244-2333
+1-403-532-7966
+1-403-699-9843
+1-866-758-0462
+1-403-938-3122
+1-587-352-0452
+1-403-279-8134
+1-403-945-3164
+1-403-289-4012
```

Thus we have changed the phone numbers entered into a single stand alone format.

House Number

The XML data provides house numbers of various buildings. These numbers mostly are made of numeric values but some contain the special character '#'. Removing this special character is necessary. Also some house numbers are listed as '-1'. Removing this is necessary.

```
def audit_housenumber(house_number):
    if house_number != '-1':
        if '#' in house_number:
            house_number=house_number.strip('#')
        return house_number
    else:
        return None
```

The above code does the work for us.

Street Type

There are irregularities in which the type of a street is mentioned in the XML file. For example the type street is mentioned as 'ST','St.','St' etc. Bringing all these different ways in which the street types are mentioned into one type is needed.

#Procedure for auditing street type

```
def audit_street_name(street_name,mapping):
    #Checking for match with R.E
    m = street_type_re.search(street_name)
    if m:
        #Extracting Match
        street_type = m.group()
        #Checking in mapping
        if street_type in mapping:
            #Altering the street type
            street_name=street_name[:-len(street_type)]
            street_name=street_name + mapping[street_type]
            #Removing problem streets
        elif problem_street.match(street_type):
            return None
        return street_name
    else:
        return None
```

#Mapping of various street types.

```
mapping= { "St": "Street",
           "St.": "Street",
           "Ave": "Avenue",
           "Rd.": "Road",
          "E": "East",
          "Blvd": "Boulevard",
          "Blvd.": "Boulevard",
          "Cres": "Crescent",
          "Dr": "Drive",
           "N.": "North",
           "N": "North",
           "E.": "East",
           "E": "East",
           "S": "South",
           "S.E": "Southeast",
           "SE": "Southeast",
           "se": "Southeast",
           "South-east": "Southeast",
           "South-west": "Southwest",
           "SW": "Southwest",
           "W.": "West",
           "W": "West",
           "N.E.": "Northeast",
           "n.e.": "Northeat",
           "NE": "Northeast",
           "N.W": "Northwest",
           "N.W.": "Northwest",
           "NW": "Northwest"}
```

The above piece of code helps us in mapping various street types into one type.

Overview of the Data

The Original OSM XML for Calgary,Canada is more than 50mb when uncompressed. Here are a few statics of the dataset:

- **Size of the file : 180 mb**
- **Size of the Sample : 18mb**
- **Number of nodes : 846653**

```
sqlite>
sqlite> select count(*) from nodes;
846653
sqlite>
```

- **Number of ways : 109992**

```
sqlite>
sqlite> select count(*) from ways;
109992
sqlite>
```

- **Number of unique users : 1016**

```
sqlite> SELECT COUNT(DISTINCT(users.uid)) FROM (SELECT uid FROM nodes UNION ALL
SELECT uid FROM ways) users;
1016
sqlite> _
```

- **Number of highways : 12559**

```
sqlite> select count(*) from nodes_tags where key='highway';
12559
sqlite> _
```

- **Number of traffic signals : 3153**

```
sqlite>
sqlite> select count(value) from nodes_tags where value='traffic_signals';
3153
sqlite>
```

- **Most popular cuisine list:**

1. **Burger : 98**
2. **Coffee_shop : 83**
3. **Pizza : 82**
4. **Sandwich : 67**
5. **Chinese : 62**

```
sqlite>
sqlite> SELECT tags.value, count(tags.value) FROM (SELECT * FROM nodes_tags LEFT
 OUTER JOIN ways_tags ON nodes_tags.id = ways_tags.id
    ...> UNION
    ...> SELECT * FROM ways_tags LEFT OUTER JOIN nodes_tags ON ways_tags.id = nod
es_tags.id) as tags where tags.key='cuisine' group by tags.value order by (count
(tags.value)) DESC limit(5);
burger,98
coffee_shop,83
pizza,82
sandwich,67
chinese,62
sqlite>
sqlite>
```

- **Number of Amenities : 4030**

```
sqlite> select count(*) from nodes_tags where key='amenity';
4030
sqlite>
```

- **Most popular shops:**

1. **Mac's : 37**
2. **7-Eleven : 29**
3. **Bell : 26**
4. **Liquor Depot : 15**

```
sqlite> SELECT nodes_tags.value, COUNT(*) as num FROM nodes_tags JOIN (SELECT DI
STINCT(id) FROM nodes_tags WHERE key='shop') q ON nodes_tags.id=q.id WHERE nodes
_tags.key='name' GROUP BY nodes_tags.value ORDER BY num DESC LIMIT 5;
"Mac's",37
7-Eleven,29
Bell,26
"Liquor Depot",15
Safeway,15
sqlite>
```

Other ideas about the dataset

The dataset, though it provides a lot of information about Calgary, Canada, there is room to improve. One of the ways in which this dataset could be improved is by adding descriptions to a lot of keys.

For example:-

The number of shops in the city of Calgary is: **1880**

```
sqlite> select count(*) from nodes_tags where key='shop';
1880
sqlite>
```

But the number of shops with description is: **28**

```
sqlite> select count(*) from ((select id,key from nodes_tags where key='shop') w
 join (select id,key from nodes_tags where key='description') r on w.id=r.id) ;
28
```

Percentage of shops with description = (28/1880)* 100 = **1.5%**

With these 2 queries we can get to know how many shops exist without a description. Only 1.5% of the shops have a description. Adding description to the shop would lead to better understanding for people accessing the data.

The OSM dataset can provide very good information regarding the growth of the city. Dataset comparison between two points in time can reveal how much the city has changed over the years. The development of the city can be viewed by everybody. Also a good overview of the city is provided. This information could be helpful to someone planning to visit the city soon.

Benefits:

- Provides information about a lot of places which would otherwise not be present in another source.
- Accurate statistics.
- Accessible to all.

Anticipated Problems:

- When more users contribute to the database, there might tend to be some irregularities in the database.
- Due to these irregularities more effort must be put into auditing to provide a cleaner database.

Conclusion

Thus the OSM dataset for the city of Calgary, Canada has been analyzed, audited , written into CSV files and imported into the database. Various SQL queries were used to get useful information about the city.