Wrangle OpenStreetMap Data

KNOXVILLE, TENNESSEE

For this Wrangle OpenStreetMap Data, I have chosen the city of Knoxville. As I have travelled in most of the states in United states, Knoxville is one city I have never been to. Therefore, I wanted to explore the city of Knoxville and eager to find out the attractions and places in Knoxville.

Knoxville is a city in and the county seat of Knox County in the U.S. state of Tennessee. As of the 2020 United States census, Knoxville's population was 190,740, making it the largest city in the East Tennessee Grand Division and the state's third largest city after Nashville and Memphis.

I downloaded the file from the link provided by the course mentor. Following is the link I used for this project: https://drive.google.com/file/d/1p8xM6n_VkGSJGxtunB9rz9fAxqtopzIm/view?usp=sharing

Introduction:

OpenStreetMap is a free, editable map of the whole world that is being built by volunteers largely from scratch and released with an open-content license. In this project, the osm file was audited, cleaned and prepared to be inserted into the sql database. The iterative parsing was done to find out about the tags and their numbers in the dataset.

Problems Encountered in the Map:

Following issues were observed while cleaning and auditing the map:

1. Improper Street names: In the map, there were some inconsistent abbreviations and, in some cases, also the spelling errors. Some of the examples are: Aaron Ln, AM Luttrell rd, Arch st, Adcock Ave etc. To fix this issue, “update_name” “function was used. Each misspelled address was replaced with the correct ones. Following function was used to fix the errors:

```
 def update_name (name, mapping):     m = street.
type_re. search (name)          if m. group () not in
expected: if m. group () in     mapping. Keys (): name =
re.sub (m. group (),     mapping [m. group ()], name)
return name
```

So, Aaron Ln was replaced with Aaron Lane.AM Luttrell Rd was replaced with AM Luttrell Road. Arch St was replaced with Arch Street. Adcock Ave was replaced with Adcock Avenue.

2. Postal Code Errors: When I ran the queries for the postal codes, the output appears to be ZIP+4 codes. Examples: 379179-0700, 37930-0001 etc. To fix this error, I used the following function so that the postal codes reveal 5 digits and that there is a uniformity in the postal codes.

```
            def update_postcode(postcode):
     match = re. match (r' ^\D*(\d {5}). *', postcode)    clean_postcode =
match. group (1): return clean_postcode
```

Data Overview:

First of all, the Knoxville. Osm file was cleaned and audited. After that, it was converted and stored in a CSV format. Then the csv files were imported as tables in the database. In the database, we ran some queries to explore various information. Following are some of the findings:

FILE SIZES:

Knoxville. Osm = 68.6 MB (71,944,733 bytes)

Nodes.csv = 121 bytes (121 bytes)

Ways.csv = 95 bytes (95 bytes)

Ways_nodes.csv = 48 bytes (48 bytes)

Nodes_tags.csv = 55 bytes (55 bytes)

Ways_tags.csv = 55 bytes (55 bytes)

Number of Nodes:

```
Cur.execute ('SELECT COUNT (*) from
nodes') 6304534
```

Number of Ways:

```
Cur.execute ('SELECT COUNT (*) from
ways') 996230
```

Total number of Users:

```
Cur.execute (' ' ' SELECT COUNT (*) as num
From (SELECT users from nodes UNION ALL
SELECT user from ways)' ' ' ) 7300764
```

Total number of Unique Users:

```
Cur.execute(''' SELECT COUNT (distinct uid))

FROM (SELECT uid FROM nodes UNION ALL SELECT uid from ways);''') 1666
```

Total number of shops:

```
select tags. Key, count (*) as num\
       from (select * from nodes_tags union all select * from ways_tags) tags \
where tags. Key = 'shop';

211
```

Conclusion:

While auditing and cleaning the data, only few parts were audited and cleaned. However, there are still lot of uncleaned data consisting typos, invalid and incorrect keywords that needs an improvement. Since OpenStreetMap data is an opensource which can be used by anyone, there is always a high risk of input of incorrect and inconsistent data. Therefore, I suggest a Data monitoring tool to monitor and check every data for validity to minimize errors.

Benefits of Data Monitoring Tool:

Data Monitoring Tool will detect and reject any inputs from the user that contains misspelled streets, typos etc. It creates a proper standard for everyone. So, it makes data clean and readable.

Anticipated Problems:

The main issue with the Data monitoring tool is to standardize all over the world. Each nation has their own standards when it comes to naming the streets, alphabets and postal code numbers. Hence, it creates various challenges and standardizing the tool becomes ineffective.