Problems Encountered in the Map

Upon initial inspection of the dataset the below problems were encountered.

- Street type abbreviations were inconsistent
    - Street had the greatest number of inconsistencies as can be seen in the mapping variable below. I was surprised that there were not mor inconsistencies found in the dataset
- Phone numbers that did not belong to 262 area code
    - Only one phone number was found that did not have a 262-area code. A few did not follow the +1 262 xxx xxxx format.
- State abbreviations in address instead of spelled out
    - A state code of WI was found where state was also spelled out as Wisconsin in the address.
- Postal codes that were not Kenosha postal codes but from surrounding areas
    - A number of postal codes in the data set belonged to the immediate area around Kenosha, Wisconsin.

Street Types

```
expected_street = ["Street", "Avenue", "Boulevard", "Drive", "Court",
"Place", "Road", "Lane", "Way"] #list of expected street names
mapping = {"St": "Street",
           "st": "Street",
           "STREET": "Street",    #mapping the audit street names to be
updated
           "street": "Street",
           "ST": "Street",
           "Rd": "Road",
           "Ave": "Avenue",
           "Ave.": "Avenue",
           "place": "Place"
           }
```

```
def update_name(name, mapping):
    m = street_type_re.search(name)

    if m:
        for i in mapping:
            if i == m.group():
                name = re.sub(street_type_re, mapping[i], name)

    return name
```

Phone Numbers

```
phone_re = re.compile(r'\+1*.262*.\d\d\d*.\d\d\d\d')


def audit_phone(phone_types, phnumber):
    good_phone = phone_re.search(phnumber)
    if not good_phone:
        phone_types.add(phnumber)
```

```
def is_phone_number(elem):
    return elem.attrib['k'] == 'phone'
```

State Abbreviation

```
def update_state(state):  #updates state abbreviation of WI to Wisconsin
    if state == expected_state:
        return state
    elif state == 'WI':
        return expected_state
    else:
        return "Not in Wisconsin"

expected_state = 'Wisconsin'
```

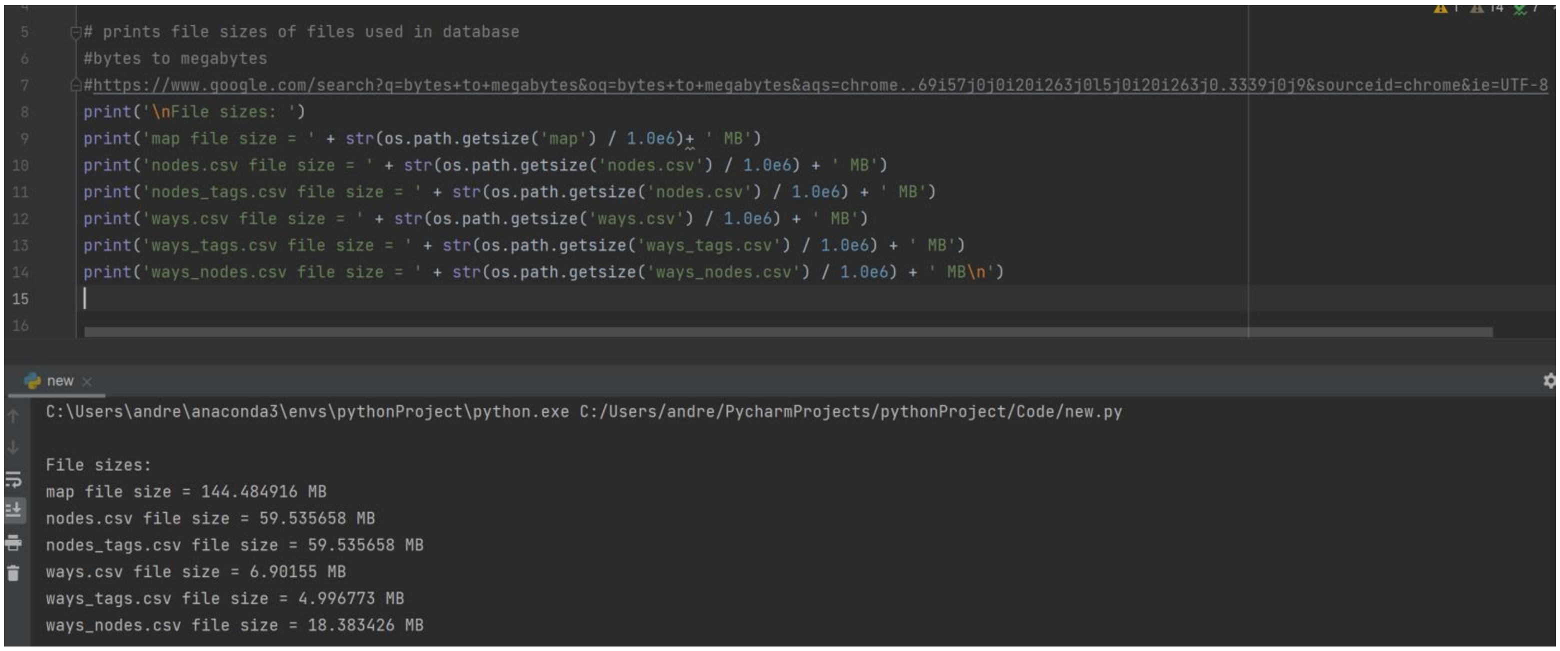
Postal Code

```
expected = ['53140', '53141', '53142', '53143', '53144']
# returns all postal codes that are good postal codes
postalcode_re = re.compile(r'5314[0-4]')


def audit_postalcode(bad_postalcodes, postalcode):
    m = postalcode_re.search(postalcode)
    if m:
        postcode = m.group()
        if postcode not in expected:  # checks postal code vs. list of 
expected postal codes
            bad_postalcodes[postcode] += 1  # if not in expected, adds to 
bad_postalcodes
    else:
        bad_postalcodes[postalcode] += 1


def is_postalcode(elem):
    return elem.attrib['k'] == 'addr:postcode'
```

Overview of the Dataset



```
# prints file sizes of files used in database
#bytes to megabytes
#https://www.google.com/search?q=bytes+to+megabytes&oq=bytes+to+megabytes&aqs=chrome..69i57j0j0i20i263j0l5j0i20i263j0.3339j0j9&sourceid=chrome&ie=UTF-8
print('\nFile sizes: ')
print('map file size = ' + str(os.path.getsize('map') / 1.0e6)+ ' MB')
print('nodes.csv file size = ' + str(os.path.getsize('nodes.csv') / 1.0e6) + ' MB')
print('nodes_tags.csv file size = ' + str(os.path.getsize('nodes.csv') / 1.0e6) + ' MB')
print('ways.csv file size = ' + str(os.path.getsize('ways.csv') / 1.0e6) + ' MB')
print('ways_tags.csv file size = ' + str(os.path.getsize('ways_tags.csv') / 1.0e6) + ' MB')
print('ways_nodes.csv file size = ' + str(os.path.getsize('ways_nodes.csv') / 1.0e6) + ' MB\n')
```

```
C:\Users\andre\anaconda3\envs\pythonProject\python.exe C:/Users/andre/PycharmProjects/pythonProject/Code/new.py

File sizes:
map file size = 144.484916 MB
nodes.csv file size = 59.535658 MB
nodes_tags.csv file size = 59.535658 MB
ways.csv file size = 6.90155 MB
ways_tags.csv file size = 4.996773 MB
ways_nodes.csv file size = 18.383426 MB
```

Database Queries

cursor.execute("SELECT COUNT(*) FROM nodes")
print("Number of Nodes found in nodes.csv = " + str(cursor.fetchall()), "\n")

Number of Nodes found in nodes.csv = [(619787,)]

cursor.execute("SELECT COUNT(*) FROM ways")
print("Number of Ways found in ways.csv = " + str(cursor.fetchall()), "\n")

Number of Ways found in ways.csv = [(94890,)]

cursor.execute("SELECT COUNT(DISTINCT(nodes.uid)) FROM nodes LEFT JOIN ways ON nodes.uid = ways.uid")
print("Number of unique users = ", str(cursor.fetchall()), "\n")

Number of unique users = [(642,)]

cursor.execute("SELECT COUNT(*) FROM nodes_tags WHERE key = 'amenity' AND value = 'bar'")
print("Number of bars found in Kenosha = ", str(cursor.fetchall()), "\n")

Number of bars found in Kenosha = [(5,)]

```
cursor.execute("SELECT COUNT(*) FROM nodes_tags WHERE key = 'cuisine' AND value = 'italian'")
print("Number of Italian restaurants Kenosha = ", str(cursor.fetchall()), "\n")
```

Number of Italian restaurants Kenosha = [(1,)]

```
cursor.execute ('''SELECT value FROM nodes_tags WHERE value LIKE "%ELEMENTARY%"''')
print("List of Elementary schools in Kenosha = ", str(cursor.fetchall()), "\n")
```

List of Elementary schools in Kenosha = [('Bain Elementary School',), ('Columbus Elementary School',), ('Frank Elementary School',), ('Pleasant Prairie Elementary School',), ('Prairie Lane Elementary School',), ('Whittier Elementary School',), ('Union Grove Elementary School',), ('Lincoln Elementary School',)]

```
cursor.execute ('''SELECT nodes_tags.value, nodes.lat, nodes.lon FROM nodes_tags LEFT JOIN nodes
on nodes_tags.id = nodes.id WHERE value LIKE "%ELEMENTARY%"''')
print("Location of Elementary schools by latitude and longitude = ", str(cursor.fetchall()), "\n")
```

Location of Elementary schools by latitude and longitude = [('Bain Elementary School', 42.5883314, -87.8360673), ('Columbus Elementary School', 42.5758535, -87.8395193), ('Frank Elementary School', 42.5830758, -87.8306303), ('Pleasant Prairie Elementary School', 42.5588899, -87.919069), ('Prairie Lane Elementary School', 42.5194647, -87.862853), ('Whittier Elementary School', 42.5505757, -87.8700759), ('Union Grove Elementary School', 42.676931, -88.0467807), ('Lincoln Elementary School', 42.5711314, -87.8300745)]

Other Ideas About the Dataset

I wanted to look at the number of bars in Kenosha because I have recently made a friend form Wisconsin and they specifically pointed out the number of bars in Wisconsin. I was surprised to find only 5 bars in the data set. However, when I did a quick Google search of bars in Kenosha this was fairly accurate with 7 results showing.

I also applied this thinking to the number of Italian restaurants, which in my query resulted in 1 Italian restaurant being found. Again, a quick Google search shows many more Italian eateries than are present in the data set.

Implementing a way to make updating the data set more readily available to the average user could increase the completeness of the data set. The benefits of this would be the ability to complete more complex queries of the data with accurate results. As it stands the results of queries can not fully be trusted. Though, the need to have an accurate representation of the number of bars and Italian restaurants is likely pretty low on anyone priority list. Which bring sus to challenges, one of the largest challenges is the knowledge of Open Street Maps. The only reason I was made aware of it was because of my participation in this project. Given that Open Street Maps has no employees and relies on the contributions of the public, this is unlikely to happen. Although this model has been successful with Wikipedia, the need for a data bade like Open Street Maps seems unnecessary when you have goliaths like Google Maps out there.