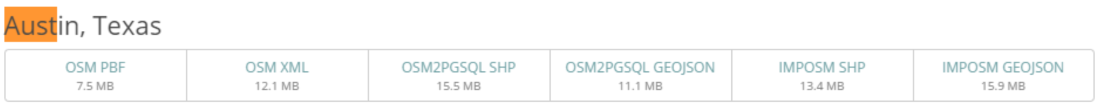
Project **P2: Data Wrangle OpenStreetMaps Data**

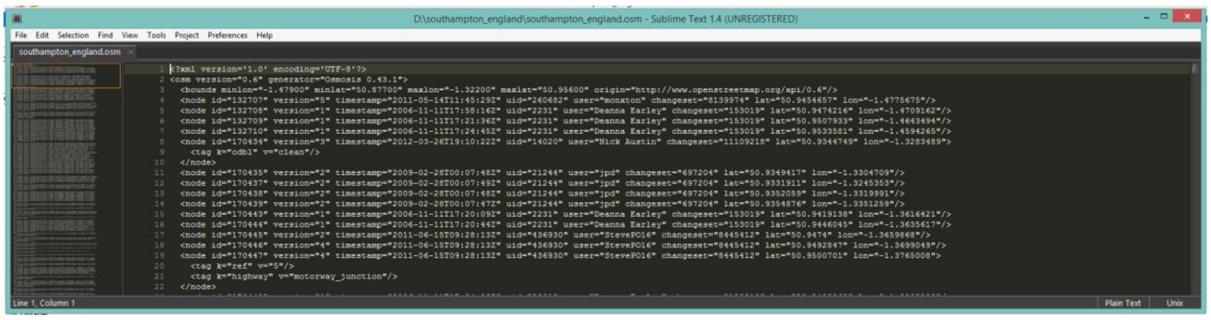
# Olakunle Kuye

## Chosen Map Area

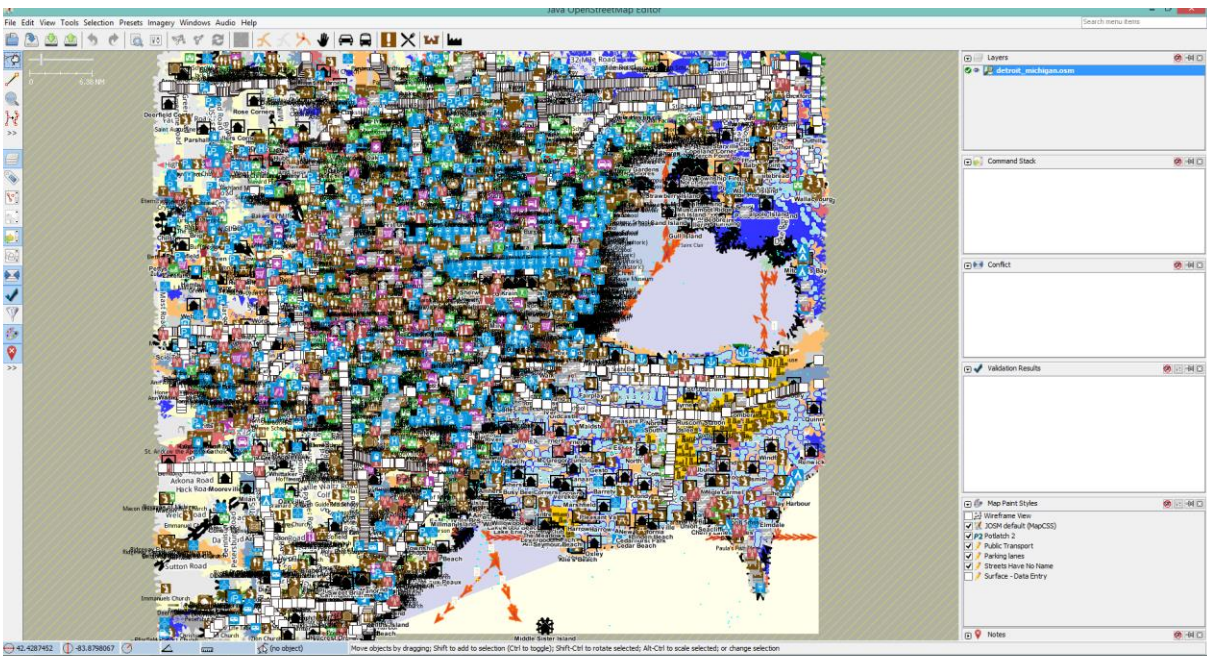
I downloaded the Austin Texas from mapzen metro – extracts:


Austin, Texas

| OSM PBF | OSM XML | OSM2PGSQL SHP | OSM2PGSQL GEOJSON | IMPOSM SHP | IMPOSM GEOJSON |
|---|---|---|---|---|---|
| 7.5 MB | 12.1 MB | 15.5 MB | 11.1 MB | 13.4 MB | 15.9 MB |


D:\southampton_england\southampton_england.osm - Sublime Text 1.4 (UNREGISTERED)
File Edit Selection Find View Tools Project Preferences Help
southampton_england.osm
Line 1, Column 1
Plain Text
Unix

```
<?xml version='1.0' encoding='UTF-8'?>
<osm version="0.6" generator="Osmosis 0.43.1">
  <bounds minlon="-1.47900" minlat="50.87700" maxlon="-1.32200" maxlat="50.95600" origin="http://www.openstreetmap.org/api/0.6"/>
  <node id="132707" version="5" timestamp="2011-05-14T11:45:29Z" uid="260682" user="monxton" changeset="8139974" lat="50.9454657" lon="-1.4775675"/>
  <node id="132708" version="1" timestamp="2006-11-11T17:58:16Z" uid="2231" user="Deanna Earley" changeset="153019" lat="50.9474216" lon="-1.4709162"/>
  <node id="132709" version="1" timestamp="2006-11-11T17:21:36Z" uid="2231" user="Deanna Earley" changeset="153019" lat="50.9507933" lon="-1.4643494"/>
  <node id="132710" version="1" timestamp="2006-11-11T17:24:45Z" uid="2231" user="Deanna Earley" changeset="153019" lat="50.9533581" lon="-1.4594265"/>
  <node id="170434" version="3" timestamp="2012-03-26T19:10:22Z" uid="14020" user="Nick Austin" changeset="11109218" lat="50.9344749" lon="-1.3283489">
    <tag k="odbl" v="clean"/>
  </node>
  <node id="170435" version="2" timestamp="2009-02-28T00:07:48Z" uid="21244" user="jpd" changeset="697204" lat="50.9349417" lon="-1.3304709"/>
  <node id="170437" version="2" timestamp="2009-02-28T00:07:49Z" uid="21244" user="jpd" changeset="697204" lat="50.9331911" lon="-1.3245353"/>
  <node id="170438" version="2" timestamp="2009-02-28T00:07:48Z" uid="21244" user="jpd" changeset="697204" lat="50.9352059" lon="-1.3319991"/>
  <node id="170439" version="2" timestamp="2009-02-28T00:07:47Z" uid="21244" user="jpd" changeset="697204" lat="50.9354876" lon="-1.3351259"/>
  <node id="170443" version="1" timestamp="2006-11-11T17:20:09Z" uid="2231" user="Deanna Earley" changeset="153019" lat="50.9419138" lon="-1.3616421"/>
  <node id="170444" version="1" timestamp="2006-11-11T17:20:44Z" uid="2231" user="Deanna Earley" changeset="153019" lat="50.9446045" lon="-1.3635617"/>
  <node id="170445" version="2" timestamp="2011-06-15T09:28:13Z" uid="436930" user="StevePO16" changeset="8445412" lat="50.9474" lon="-1.3659868"/>
  <node id="170446" version="4" timestamp="2011-06-15T09:28:13Z" uid="436930" user="StevePO16" changeset="8445412" lat="50.9492847" lon="-1.3699049"/>
  <node id="170447" version="4" timestamp="2011-06-15T09:28:13Z" uid="436930" user="StevePO16" changeset="8445412" lat="50.9500701" lon="-1.3765008">
    <tag k="ref" v="5"/>
    <tag k="highway" v="motorway_junction"/>
  </node>
```

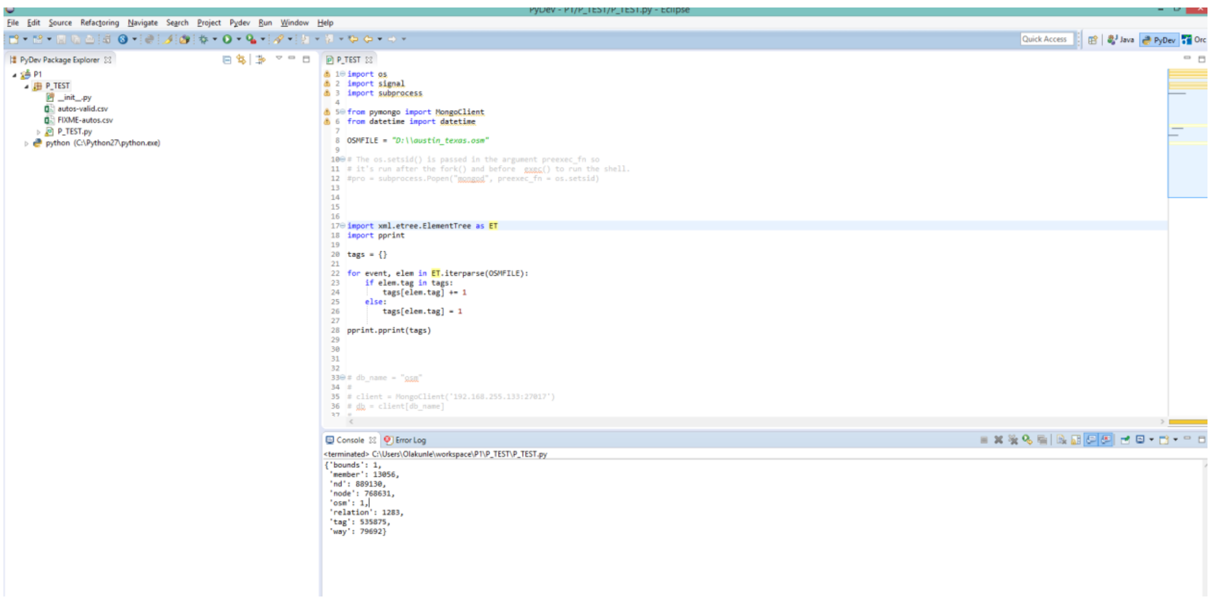

Java OpenStreetMap Editor
File Edit View Tools Selection Presets Imagery Windows Audio Help
Layers
detroit_michigan.osm
Command Stack
Conflict
Validation Results
Map Paint Styles
Wireframe View
JOSM default (MapCSS)
P2 PoBatch 2
Public Transport
Parking lanes
Streets Have No Name
Surface - Data Entry
Notes
42.4287452
-83.8798067
(no object)
Move objects by dragging; Shift to add to selection (Ctrl to toggle); Shift-Ctrl to rotate selected; Alt-Ctrl to scale selected; or change selection

## Auditing the data

After downloading the OSM XML data, I parsed it using the code I used for the iterative Parsing exercise.

With the result being:

```
{'bounds': 1,
 'member': 13056,
 'nd': 889130,
 'node': 768631,
 'osm': 1,
 'relation': 1283,
 'tag': 535875,
 'way': 79692}
```


PyDev - P1/P_TEST/P_TEST.py - Eclipse

```
import os
import signal
import subprocess

from pymongo import MongoClient
from datetime import datetime

OSMFILE = "D:\\austin_texas.osm"

# The os.setsid() is passed in the argument preexec_fn so
# it's run after the fork() and before  exec() to run the shell.
#pro = subprocess.Popen("mongod", preexec_fn = os.setsid)




import xml.etree.ElementTree as ET
import pprint

tags = {}

for event, elem in ET.iterparse(OSMFILE):
    if elem.tag in tags:
        tags[elem.tag] += 1
    else:
        tags[elem.tag] = 1

pprint.pprint(tags)




# db_name = "osm"
#
# client = MongoClient('192.168.255.133:27017')
# db = client[db_name]
```

```
<terminated> C:\Users\Olakunle\workspace\P1\P_TEST\P_TEST.py
{'bounds': 1,
 'member': 13056,
 'nd': 889130,
 'node': 768631,
 'osm': 1,
 'relation': 1283,
 'tag': 535875,
 'way': 79692}
```

I ran the code I used for the Tag Type exercise whereby checks on the dataset were made to determine if they are certain patterns in the tags present in the data.

This is where there are variables declared to search for the following three regular expressions: **lower**, **lower_colon**, and **problemchars**.

**lower**: This variable searches for strings containing lower case characters

**lower_colon**: Searches for strings containing lower case characters and a single colon within the string

**problemchars**: Searches for  characters that cannot be used within keys in **MongoDB**.

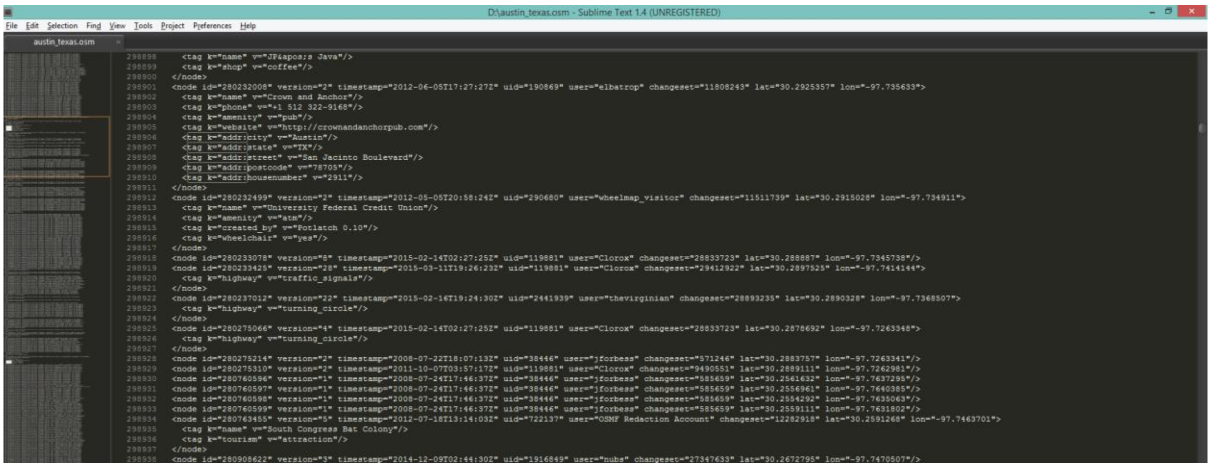
```
lower = re.compile(r'^([a-z]|_)*$')

lower_colon = re.compile(r'^([a-z]|_)*:([a-z]|_)*$')

problemchars = re.compile(r'[=\+/&<>;\'"\?%#$@\,\.
\t\r\n]')
```

As the image below shows there are a number of tags with multiple children tags that begin with address .

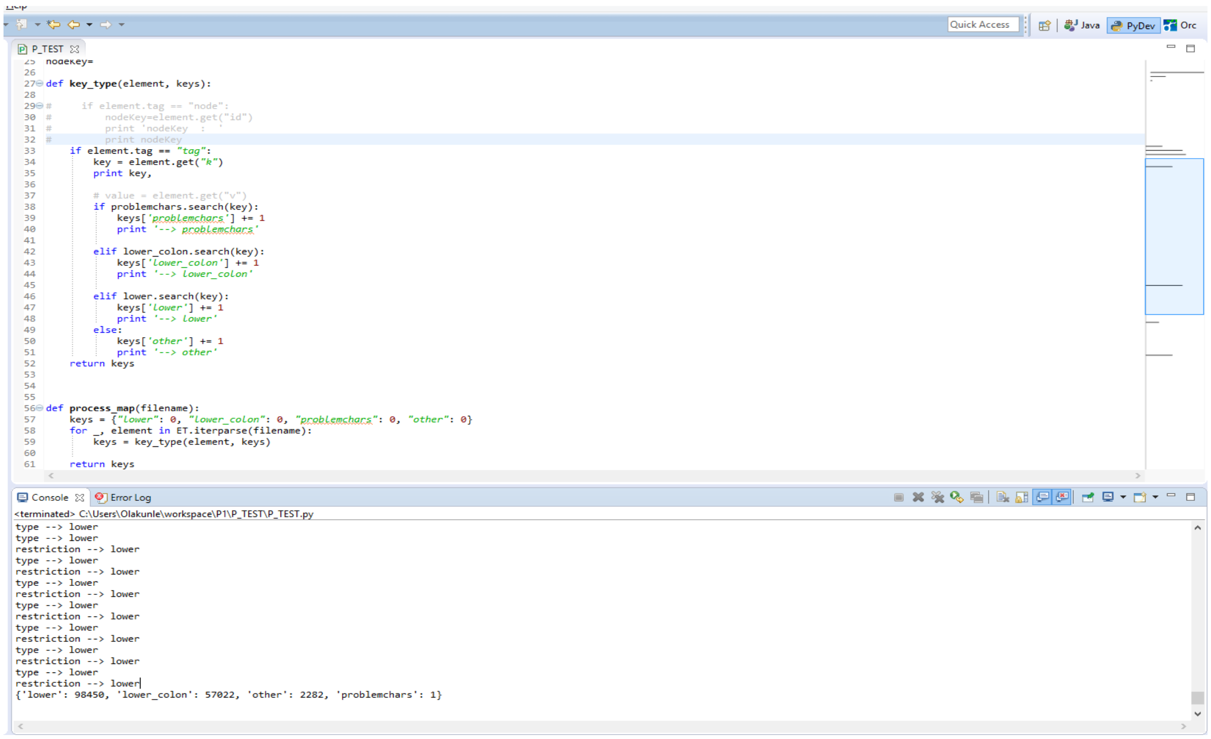
These will be used with the lower_colon regex to find these keys so that it’s possible to build a single address document within a larger json document.


D:\austin_texas.osm - Sublime Text 1.4 (UNREGISTERED)
File Edit Selection Find View Tools Project Preferences Help
austin_texas.osm

```
    <tag k="name" v="JP&apos;s Java"/>
    <tag k="shop" v="coffee"/>
  </node>
  <node id="280232008" version="2" timestamp="2012-06-05T17:27:27Z" uid="190869" user="elbatrop" changeset="11808243" lat="30.2925357" lon="-97.735633">
    <tag k="name" v="Crown and Anchor"/>
    <tag k="phone" v="+1 512 322-9168"/>
    <tag k="amenity" v="pub"/>
    <tag k="website" v="http://crownandanchorpub.com"/>
    <tag k="addr:city" v="Austin"/>
    <tag k="addr:state" v="TX"/>
    <tag k="addr:street" v="San Jacinto Boulevard"/>
    <tag k="addr:postcode" v="78705"/>
    <tag k="addr:housenumber" v="2911"/>
  </node>
  <node id="280232499" version="2" timestamp="2012-05-05T20:58:24Z" uid="290680" user="wheelmap_visitor" changeset="11511739" lat="30.2915028" lon="-97.734911">
    <tag k="name" v="University Federal Credit Union"/>
    <tag k="amenity" v="atm"/>
    <tag k="created_by" v="Potlatch 0.10"/>
    <tag k="wheelchair" v="yes"/>
  </node>
  <node id="280233078" version="8" timestamp="2015-02-14T02:27:25Z" uid="119881" user="Clorox" changeset="28833723" lat="30.288887" lon="-97.7345738"/>
  <node id="280233425" version="28" timestamp="2015-03-11T19:26:23Z" uid="119881" user="Clorox" changeset="29412922" lat="30.2897525" lon="-97.7414144">
    <tag k="highway" v="traffic_signals"/>
  </node>
  <node id="280237012" version="22" timestamp="2015-02-16T19:24:30Z" uid="2441939" user="thevirginian" changeset="28893235" lat="30.2890328" lon="-97.7368507">
    <tag k="highway" v="turning_circle"/>
  </node>
  <node id="280275066" version="4" timestamp="2015-02-14T02:27:25Z" uid="119881" user="Clorox" changeset="28833723" lat="30.2878692" lon="-97.7263348">
    <tag k="highway" v="turning_circle"/>
  </node>
  <node id="280275214" version="2" timestamp="2008-07-22T18:07:13Z" uid="38446" user="jforbess" changeset="571246" lat="30.2883757" lon="-97.7263341"/>
  <node id="280275310" version="2" timestamp="2011-10-07T03:57:17Z" uid="119881" user="Clorox" changeset="9490551" lat="30.2889111" lon="-97.7262981"/>
  <node id="280760596" version="1" timestamp="2008-07-24T17:46:37Z" uid="38446" user="jforbess" changeset="585659" lat="30.2561632" lon="-97.7637295"/>
  <node id="280760597" version="1" timestamp="2008-07-24T17:46:37Z" uid="38446" user="jforbess" changeset="585659" lat="30.2556961" lon="-97.7640385"/>
  <node id="280760598" version="1" timestamp="2008-07-24T17:46:37Z" uid="38446" user="jforbess" changeset="585659" lat="30.2554292" lon="-97.7635063"/>
  <node id="280760599" version="1" timestamp="2008-07-24T17:46:37Z" uid="38446" user="jforbess" changeset="585659" lat="30.2559111" lon="-97.7631802"/>
  <node id="280763455" version="5" timestamp="2012-07-18T13:14:03Z" uid="722137" user="OSMF Redaction Account" changeset="12282918" lat="30.2591268" lon="-97.7463701">
    <tag k="name" v="South Congress Bat Colony"/>
    <tag k="tourism" v="attraction"/>
  </node>
  <node id="280908622" version="3" timestamp="2014-12-09T02:44:30Z" uid="1916849" user="nubs" changeset="27347633" lat="30.2672795" lon="-97.7470507"/>
```

Here is a partial output and a screen shot from running the code used in the Tag Type exercise:

```
{'lower': 226464, 'lower_colon': 299286,
'other': 10123, 'problemchars': 2}
```



```
nodeKey=

def key_type(element, keys):

#       if element.tag == "node":
#           nodeKey=element.get("id")
#           print 'nodeKey  :  '
#           print nodeKey
    if element.tag == "tag":
        key = element.get("k")
        print key,

        # value = element.get("v")
        if problemchars.search(key):
            keys['problemchars'] += 1
            print '--> problemchars'

        elif lower_colon.search(key):
            keys['lower_colon'] += 1
            print '--> lower_colon'

        elif lower.search(key):
            keys['lower'] += 1
            print '--> lower'
        else:
            keys['other'] += 1
            print '--> other'
    return keys



def process_map(filename):
    keys = {"lower": 0, "lower_colon": 0, "problemchars": 0, "other": 0}
    for _, element in ET.iterparse(filename):
        keys = key_type(element, keys)

    return keys
```

```
<terminated> C:\Users\Olakunle\workspace\P1\P_TEST\P_TEST.py
type --> lower
type --> lower
restriction --> lower
type --> lower
restriction --> lower
type --> lower
restriction --> lower
type --> lower
restriction --> lower
type --> lower
restriction --> lower
type --> lower
restriction --> lower
type --> lower
restriction --> lower
{'lower': 98450, 'lower_colon': 57022, 'other': 2282, 'problemchars': 1}
```

By modifying the code used in Tag Type exercise to build a set of unique ids found in the xml, outputting the length of the set that represents the number of users making edits in the chosen map area.

```
def process_map(filename):
    users = set()
    for _, element in ET.iterparse(filename):
        if "uid" in element.attrib and element.tag in
('node', 'way'):
            users.add(element.attrib["uid"])
    return users


def test():
    # You can use another testfile 'map.osm' to look at
your solution
    # Note that the assertions will be incorrect then.
    #keys =
process_map('D:\\southampton_england\\southampton_england.
osm')
    users =
process_map('D:\\southampton_england\\southampton_england.
osm')

    print len(users)
```

**The output returned was: 910**

## Problems with the Data

### Street Names

The majority of this project will be devoted to auditing and cleaning street names seen within the OSM XML.

Street types used by users in the process of mapping are quite often abbreviated. I will attempt to find these abbreviations and replace them with their full text form. The plan of action is as follows:

The data contained in the OSM XML file will be audited to clean the street types used by users, which are often abbreviated. These abbreviation’s will be found and replaced with the full street name.

In order to do this, the following steps will be taken:

- A regex designed to match the last token in a string will be developed, finding the street type in an address.
- I would then proceeded to build a list of expected street types that don’t need to be cleaned.
- The XML file will be parsed for tag elements with the **k="addr:street" attributes** .

- A search will be conducted using the regex on the value v attribute of the street name string.
- A dictionary will be built with keys that match the street types in the regex, setting the street names where a specific key was found as a value, this

enables the determination of what needs to be cleaned.

- Another dictionary will be built to contain a map from an offending street type. This along with another regex and a function that will be developed and used to match the offending street types will be used to clean the data. The function will return a cleaned string.

```
import xml.etree.cElementTree as ET
from collections import defaultdict
import re
import pprint

OSMFILE = "example.osm"
street_type_re = re.compile(r'\b\S+\.?$', re.IGNORECASE)


expected = ["Street", "Avenue", "Boulevard", "Drive",
"Court", "Place", "Square", "Lane", "Road",
            "Trail", "Parkway", "Commons"]
```

The **audit_string** takes in the dictionary of street types, an audit string a regex match and a list of expected street types.

The function will search the string passed to it for a regex and if it finds a match will that is not in the dictionary used to hold expected dictionaries it will be added to it.

```
def audit_string(match_set_dict, string_to_audit, regex,
expected_matches):
    m = regex.search(string_to_audit)
    if m:
        match_string = m.group()
        if match_string not in expected_matches:

match_set_dict[match_string].add(string_to_audit)
```

An audit function is defined and it’s used to parse and audit street names. The function audits tag elements where **k="addr:street"** and it also matches the tag filter function . The audit function takes in a regex and a list of expected matches.

```
def audit(osmfile, tag_filter, regex, expected_matches = 
[]):
    osm_file = open(osmfile, "r")
    match_sets = defaultdict(set)
    # iteratively parse the mapping xml
    for event, elem in ET.iterparse(osm_file, 
events=("start",)):
        # node and way tags are of special interest
        if elem.tag == "node" or elem.tag == "way":
            # iterate the "tag" tags within a node or way
            for tag in elem.iter("tag"):
                if tag_filter(tag):
                    audit_string(match_sets, 
tag.attrib['v'], regex, expected_matches)
    return match_sets
```

The function **is_street_name** determines if an element contains an attribute **k="addr:street".** I will use **is_street_name** as the **tag_filter** when I call the audit function to audit street names.

```
def is_street_name(elem):
    return (elem.attrib['k'] == "addr:street")
```

Here is a pretty print of the output of the audit function.

```
street_types = audit(OSMFILE, tag_filter = is_street_name,
regex = street_type_re,
expected_matches = expected)
pprint.pprint(dict(street_types))
```

```
{'100': set(['Jollyville Road Suite 100', 'Old Jollyville Road, Suite 100']),
 '101': set(['4207 James Casey st #101']),
 '104': set(['11410 Century Oaks Terrace Suite #104']),
 '1100': set(['Hwy 290 W, Bldg 1100']),
 '12': set(['Ranch to Market Road 12']),
 '120': set(['Building B Suite 120']),
 '150': set(['Farm-to-Market Road 150']),
 '1626': set(['F.M. 1626', 'FM 1626', 'Farm-to-Market Road 1626']),
 '163': set(['Bee Cave Road Suite 163']),
 '170': set(['County Road 170']),
 '1825': set(['FM 1825']),
 '1826': set(['Farm To Market Road 1826', 'Ranch to Market Road 1826']),
 '183': set(['N HWY 183', 'U.S. 183', 'US 183', 'US Highway 183']),
 '203': set(['West Ben White Boulevard #203',
             'West Ben White Boulevard, #203']),
 '210': set(['Greystone Dr Ste 210']),
 '275': set(['West 11th Street, Suite 275']),
 '290': set(['E Hwy 290',
             'East Hwy 290',
             'Highway 290',
             'U.S. 290',
             'US 290',
             'W HWY 290',
             'W Highway 290',
             'W Hwy 290',
             'W. Highway 290',
             'West Highway 290']),
 '3': set(['Shoal Creek Blvd, Bld 3']),
 '300': set(['321 W Ben White Blvd #300', 'West 5th Street #300']),
 '301': set(['Hwy 290 W. Ste. 301', 'S Bell Blvd., Suite 301']),
 '3177': set(['FM 3177']),
 '35': set(['9600 S IH 35',
           'I 35',
           'Interstate 35',
           'Interstate Highway 35',
           'N I H 35',
           'N Interstate Highway 35',
           'N. IH 35',
           'North IH 35',
           'North Interstate Highway 35',
           'S Interstate 35',
           'S Interstate Highway 35',
           'South Interstate 35']),
 '4': set(['4201 South Congress Avenue #4']),
 '400': set(['North Lakeline Boulevard, Ste. 400']),
 '406': set(['North Highway 183 #406']),
 '45': set(['TX 45']),
 '535': set(['FM 535']),
 '6': set(['N IH 35 Bldg. 6']),
 '620': set(['FM 620',
             'N FM 620',
             'RM 620',
             'RR 620',
             'Ranch Road 620',
             'Ranch-to-Market Road 620']),
 '685': set(['FM 685', 'Farm to Market 685']),
 '7': set(['N I H 35 Bldg 7']),
 '71': set(['TX 71', 'W Hwy 71', 'West Highway 71']),
 '78738': set(['12200 Bee Cave Pkwy, Bee Cave, TX 78738']),
 '79': set(['U.S. 79']),
 '969': set(['FM 969']),
 'A-15': set(['W. Anderson Lane, Suite A-15']),
 'A500': set(['Burnet Rd Ste, A500']),
 'Austin': set(['Research Blvd, Austin']),
 'Ave': set(['Congress Ave',
             'E. University Ave',
             'Round Rock Ave',
             'S Congress Ave',
             'South Congress Ave']),
 'Ave.': set(['Woodrow Ave.']),
 'Avene': set(['South Congress Avene']),
 'B': set(['Avenue B']),
 'B100': set(['North Lamar Boulevard suite #B100']),
 'Blvd': set(['10000 Research Blvd',
              'Airport Blvd',
              'Escarpment Blvd',
              'Industrial Blvd',
              'N Bell Blvd',
              'North Lamar Blvd',
              'North Research Blvd',
              'Research Blvd',
              'South Bell Blvd',
              'South Lamar Blvd',
              'Stonelake Blvd',
              'Swenson Farms Blvd',
              'West Ben White Blvd',
              'West Louis Henna Blvd']),
 'Blvd.': set(['East Old Settlers Blvd.',
               'Research Blvd.',
               'South Exposition Blvd.',
               'South Lamar Blvd.',
               'Tandem Blvd.',
               'W North Loop Blvd.']),
 'C-200': set(['9600 IH 35 C-200']),
 'CR': set(['Rain Lily CR']),
 'Cannon': set(['West William Cannon']),
 'Chavez': set(['West Cesar Chavez']),
 'Cir': set(['Courtland Cir', 'Dry Run Cir', 'Tanager Cir']),
 'Circle': set(['Coronado Circle', 'Morado Circle']),
 'Cove': set(['Glacier Cove',
              'Knoll Cove',
              'Mustang Cove',
              'Rain Dance Cove',
              'Yucca Cove']),
 'Crossing': set(['Esperanza Crossing', 'Kyle Crossing', 'Nuckols Crossing']),
 'Ct': set(['Briarpatch Ct', 'Grapevine Ct']),
 'Cv': set(['Blue Sky Cv',
            'Clear Pond Cv',
            'Enchanted Cv',
            'Mallard Cv',
            'Red River Cv']),
 'D1': set(['W Hwy 71, STE D1']),
 'Dr': set(['Blazing Star Dr',
            'Carranzo Dr',
            'Country Creek Dr',
            'Executive Center Dr',
            'Forest Creek Dr',
            'Jack Nicklaus Dr',
            'Jimmy Clay Dr',
            'Kirkglen Dr',
            'Kyle Center Dr',
            'Round Rock West Dr',
            'W Lake Dr',
            'W William Cannon Dr',
            'Walter Seaholm Dr',
            'Wild Rose Dr',
            'Wild Turkey Dr']),
 'Dr.': set(['11928 Stonehollow Dr.', 'Sportsplex Dr.', 'Stonebridge Dr.']),
 'East': set(['Hwy 290 East', 'US 290 East']),
 'Expressway': set(['North Mo-Pac Expressway',
                    'North MoPac Expressway',
                    'North Mopac Expressway',
                    'South MoPac Expressway']),
 'Expwy': set(['N Mopac Expwy']),
 'FM1431': set(['FM1431']),
 'Fork': set(['Roaring Fork']),
 'Fortuna': set(['Via Fortuna']),
 'Gonzales': set(['Gonzales']),
 'Grande': set(['Rio Grande']),
 'Guadalupe': set(['Guadalupe']),
 'H': set(['Avenue H']),
 'Highway': set(['Capital of Texas Highway',
                 'North Capitol of Texas Highway',
                 'S Capital of Texas Highway',
                 'South Capital of Texas Highway']),
 'Hollow': set(['Lost Oasis Hollow']),
 'Hwy': set(['S Capital of Texas Hwy']),
 'I35': set(['I35']),
 'IH-35': set(['N IH-35', 'South IH-35']),
 'IH35,': set(['N. IH35,']),
 'J': set(['Business Interstate Highway 35 J']),
 'Jr': set(['Robert Martinez Jr']),
 'Liberty': set(['W. Liberty']),
 'Ln': set(['Honeybee Ln', 'Rockwood Ln', 'W Anderson Ln']),
 'Loop': set(['Amberwood Loop', 'E. North Loop', 'Mirasol Loop']),
 'N': set(['Capital of TX Hwy N', 'Farm to Market 620 N']),
 'North': set(['Ashwood North',
               'Cuernavaca Drive North',
               'FM 620 North',
               'IH 35 North',
               'Interstate Highway 35 Service Road North',
               'Ranch Road 620 North']),
 'Northwest': set(['Carlos G Parker Boulevard Northwest']),
 'Oaks': set(['Sierra Oaks']),
 'Oltorf': set(['East Oltorf']),
 'Pass': set(['Continental Pass', 'Turks Cap Pass']),
 'Path': set(['Daisy Path']),
 'Pflugerville': set(['N. IH 35 Pflugerville']),
 'Picadilly': set(['Picadilly']),
 'Pkwy': set(['Riata Trace Pkwy', 'Roger Hanks Pkwy']),
 'Quarry': set(['Quarry']),
 'RD': set(['Grassy Field RD']),
 'Rd': set(['Barley Rd',
            'Barton Springs Rd',
            'Big Meadow Rd',
            'Burnet Rd',
            'Commons Rd',
            'Dry Creek Rd',
            'Elderberry Rd',
            'Grassy Field Rd',
            'N Interstate 35 Frontage Rd',
            'Open Sky Rd',
            'Saddleback Rd',
            'Spicewood Springs Rd',
            'W US Highway 290 Service Rd']),
 'Real': set(['El Camino Real']),
 'River': set(['Red River']),
 'SB': set(['South I-35 Service SB']),
 'Seco': set(['Arroyo Seco']),
 'South': set(['Venture Boulevard South']),
 'Speedway': set(['Speedway']),
 'Spicewood': set(['State Highway 71 W Spicewood']),
 'St': set(['Duval St',
            'E 43rd St',
            'E Live Oak St',
            'E Oltorf St',
            'Guadalupe St',
            'N Main St',
            'Pecan St',
            'Rio Grande St',
            'S 1st St',
            'W 6th St',
            'W Annie St',
            'West Lynn St']),
 'St.': set(['E 38th 1/2 St.', 'E. 43rd St.', 'Guadalupe St.', 'Pecan St.']),
 'Sutton': set(['Sutton']),
 'Terrace': set(['Norseman Terrace']),
 'W': set(['East HWY 290 W', 'Highway 71 W', 'Hwy 290 W']),
 'Way': set(['Brista Way',
             'Dell Way',
             'Desert Willow Way',
             'Mountain Laurel Way',
             'Pure Brook Way',
             'Running Water Way',
             'Technology Way',
             'Whispering Wind Way',
             'Wild Plum Way',
             'Winecap Way']),
 'West': set(['Old 2243 West', 'US Highway 290 West']),
 'Willo': set(['Switch Willo']),
 'Wow': set(['Pow Wow']),
 'lane': set(['brigadoon lane']),
 'street': set(['East main street', 'South 1st street']),
 'suite#L131': set(['N. Lamar suite#L131']),
 'texas': set(['9901 N Capital Of texas'])}
```

There were a number unexpected abbreviated street types as well as cardinal directions and highway directions. I created a function to replace the abbreviated street types taking a regex to search for, updating a mapping directory.

```
def update(string_to_update, mapping, regex):
    m = regex.search(string_to_update)
    if m:
        match = m.group()
        if match in mapping:
            string_to_update = re.sub(regex, 
mapping[match], string_to_update)
    return string_to_update
```

The results of the audit function were used to build a dictionary to map abbreviations to their full and clean representation.

```
map_street_types = \
{
"Ave" : "Avenue",
"BLVD" : "Boulevard",
"Blvd" : "Boulevard",
"Blvd." : "Boulevard",
"Cir" : "Circle",
"Dr" : "Drive",
"Ln" : "Lane",
"Ln" : "Lane",
"Pkwy" : "Parkway",
"Rd" : "Road",
"Rd." : "Road",
"St" : "Street",
"St." : "Street"
}
```

The keys are replaced where they appear in a string using:

```
bad_streets =
"|".join(map_street_types.keys()).replace('.', '')

street_type_updater_re = re.compile(r'\b(' + bad_streets +
r')\b\.?', re.IGNORECASE)
```

Here are the results showing the abbreviated street types have been updated as expected:

```
W US Highway 290 Service Rd => W US Highway 290 Service Road
Saddleback Rd => Saddleback Road
Barley Rd => Barley Road
Open Sky Rd => Open Sky Road
Elderberry Rd => Elderberry Road
N Interstate 35 Frontage Rd => N Interstate 35 Frontage Road
Spicewood Springs Rd => Spicewood Springs Road
Big Meadow Rd => Big Meadow Road
Commons Rd => Commons Road
Burnet Rd => Burnet Road
Barton Springs Rd => Barton Springs Road
Dry Creek Rd => Dry Creek Road
Grassy Field Rd => Grassy Field Road
Roger Hanks Pkwy => Roger Hanks Parkway
Riata Trace Pkwy => Riata Trace Parkway
Tanager Cir => Tanager Circle
Dry Run Cir => Dry Run Circle
Courtland Cir => Courtland Circle
Congress Ave => Congress Avenue
E. University Ave => E. University Avenue
South Congress Ave => South Congress Avenue
Round Rock Ave => Round Rock Avenue
S Congress Ave => S Congress Avenue
Tandem Blvd. => Tandem Boulevard
South Lamar Blvd. => South Lamar Boulevard
South Exposition Blvd. => South Exposition Boulevard
```

```
W North Loop Blvd. => W North Loop Boulevard
East Old Settlers Blvd. => East Old Settlers Boulevard
Research Blvd. => Research Boulevard
Pecan St. => Pecan Street
Guadalupe St. => Guadalupe Street
E. 43rd St. => E. 43rd Street
E 38th 1/2 St. => E 38th 1/2 Street
Honeybee Ln => Honeybee Lane
W Anderson Ln => W Anderson Lane
Rockwood Ln => Rockwood Lane
Kyle Center Dr => Kyle Center Drive
Kirkglen Dr => Kirkglen Drive
Country Creek Dr => Country Creek Drive
Blazing Star Dr => Blazing Star Drive
Walter Seaholm Dr => Walter Seaholm Drive
Wild Turkey Dr => Wild Turkey Drive
W Lake Dr => W Lake Drive
W William Cannon Dr => W William Cannon Drive
Executive Center Dr => Executive Center Drive
Jack Nicklaus Dr => Jack Nicklaus Drive
Wild Rose Dr => Wild Rose Drive
Round Rock West Dr => Round Rock West Drive
Carranzo Dr => Carranzo Drive
Forest Creek Dr => Forest Creek Drive
Jimmy Clay Dr => Jimmy Clay Drive
Pecan St => Pecan Street
N Main St => N Main Street
S 1st St => S 1st Street
E Oltorf St => E Oltorf Street
E 43rd St => E 43rd Street
E Live Oak St => E Live Oak Street
W Annie St => W Annie Street
Rio Grande St => Rio Grande Street
W 6th St => W 6th Street
West Lynn St => West Lynn Street
Guadalupe St => Guadalupe Street
Duval St => Duval Street
West Louis Henna Blvd => West Louis Henna Boulevard
Airport Blvd => Airport Boulevard
Swenson Farms Blvd => Swenson Farms Boulevard
West Ben White Blvd => West Ben White Boulevard
N Bell Blvd => N Bell Boulevard
North Lamar Blvd => North Lamar Boulevard
Industrial Blvd => Industrial Boulevard
Stonelake Blvd => Stonelake Boulevard
Escarpment Blvd => Escarpment Boulevard
South Lamar Blvd => South Lamar Boulevard
South Bell Blvd => South Bell Boulevard
North Research Blvd => North Research Boulevard
Research Blvd => Research Boulevard
10000 Research Blvd => 10000 Research Boulevard
```

## The cardinal directions North, South, East and West appear to be abbreviated in

the data, so using a similar technique to set the correct the abbreviated cardinals by creating a regex to match NSEW at the begging of a string, followed by an optional period.

Here is the output of auditing using

**cardinal_directions = audit(OSMFILE, is_street_name, cardinal_dir_re)**

**outputs :**

```
{'E': set(['E 38th 1/2 St.',
           'E 43rd St',
           'E Hwy 290',
           'E Live Oak St',
           'E Main Street',
           'E Oltorf St',
           'E Palm Valley Boulevard',
           'E Riverside Drive']),
 'E.': set(['E. 43rd St.', 'E. North Loop', 'E. University Ave']),
 'N': set(['N Bell Blvd',
           'N FM 620',
           'N HWY 183',
           'N I H 35',
           'N I H 35 Bldg 7',
           'N IH 35 Bldg. 6',
           'N IH-35',
           'N Interstate 35 Frontage Rd',
           'N Interstate Highway 35',
           'N Lake Creek Drive',
           'N Main St',
           'N Mopac Expwy']),
 'N.': set(['N. IH 35',
            'N. IH 35 Pflugerville',
            'N. IH35,',
            'N. Lamar suite#L131']),
 'S': set(['S 1st St',
           'S Bell Blvd., Suite 301',
           'S Capital of Texas Highway',
           'S Capital of Texas Hwy',
           'S Congress Ave',
           'S Interstate 35',
           'S Interstate Highway 35']),
 'W': set(['W 35th Street',
           'W 6th St',
           'W Anderson Ln',
           'W Annie St',
           'W HWY 290',
           'W Highway 290',
           'W Hwy 290',
           'W Hwy 71',
           'W Hwy 71, STE D1',
           'W Lake Dr',
           'W North Loop Blvd.',
           'W Parmer Lane',
           'W US Highway 290 Service Rd',
           'W William Cannon Dr']),
 'W.': set(['W. Anderson Lane, Suite A-15',
            'W. Highway 290',
            'W. Liberty',
            'W. Slaughter Lane'])}
```

The output shows the cardinals at the beginning street names, but creating an exhaustive mapping will resolve this using:

```
map_cardinal_directions = \
    {
        "E" : "East",
        "E." : "East",
        "N" : "North",
        "N." : "North",
        "S" : "South",
        "S." : "South",
        "W" : "West",
        "W." : "West"
    }
```

Whereby the keys will be replaced anywhere in the string using a regex: exhaustive mapping

```
bad_directions =
"|".join(map_cardinal_directions.keys()).replace('.', '')
```

```
cardinal_dir_updater_re = re.compile(r'\b(' +
bad_directions + r')\b\.?', re.IGNORECASE)
```

I modified the code used in the audit.py for exercise 6, traversing the dictionary applying updates for street types and the appropriate cardinal direction.

Using:

```
    cardinal_directions = audit(OSMFILE, is_street_name,
cardinal_dir_re)
    for cardinal_direction, ways in
cardinal_directions.iteritems():
        if cardinal_direction in map_cardinal_directions:
            for name in ways:
                better_name = update(name,
map_street_types, street_type_updater_re)
                best_name = update(better_name,
map_cardinal_directions, cardinal_dir_updater_re)
                print name, "=>", better_name, "=>",
best_name
```

**This produces the output:**

```
E. University Ave => E. University Avenue => East University Avenue
E. 43rd St. => E. 43rd Street => East 43rd Street
E. North Loop => E. North Loop => East North Loop
E Oltorf St => E Oltorf Street => East Oltorf Street
E Live Oak St => E Live Oak Street => East Live Oak Street
E Main Street => E Main Street => East Main Street
E 43rd St => E 43rd Street => East 43rd Street
E Riverside Drive => E Riverside Drive => East Riverside Drive
E Hwy 290 => E Hwy 290 => East Hwy 290
E 38th 1/2 St. => E 38th 1/2 Street => East 38th 1/2 Street
E Palm Valley Boulevard => E Palm Valley Boulevard => East Palm Valley Boulevard
N. IH35, => N. IH35, => North IH35,
N. Lamar suite#L131 => N. Lamar suite#L131 => North Lamar suite#L131
```

```
N. IH 35 => N. IH 35 => North IH 35
N. IH 35 Pflugerville => N. IH 35 Pflugerville => North IH 35 Pflugerville
W. Liberty => W. Liberty => West Liberty
W. Slaughter Lane => W. Slaughter Lane => West Slaughter Lane
W. Highway 290 => W. Highway 290 => West Highway 290
W. Anderson Lane, Suite A-15 => W. Anderson Lane, Suite A-15 => West Anderson
Lane, Suite A-15
N Interstate Highway 35 => N Interstate Highway 35 => North Interstate Highway 35
N Bell Blvd => N Bell Boulevard => North Bell Boulevard
N Interstate 35 Frontage Rd => N Interstate 35 Frontage Road => North Interstate
35 Frontage Road
N IH 35 Bldg. 6 => N IH 35 Bldg. 6 => North IH 35 Bldg. 6
N FM 620 => N FM 620 => North FM 620
N IH-35 => N IH-35 => North IH-35
N I H 35 Bldg 7 => N I H 35 Bldg 7 => North I H 35 Bldg 7
N Lake Creek Drive => N Lake Creek Drive => North Lake Creek Drive
N I H 35 => N I H 35 => North I H 35
N Main St => N Main Street => North Main Street
N HWY 183 => N HWY 183 => North HWY 183
N Mopac Expwy => N Mopac Expwy => North Mopac Expwy
S Bell Blvd., Suite 301 => S Bell Boulevard, Suite 301 => South Bell Boulevard,
Suite 301
S 1st St => S 1st Street => South 1st Street
S Interstate 35 => S Interstate 35 => South Interstate 35
S Capital of Texas Highway => S Capital of Texas Highway => South Capital of Texas
Highway
S Congress Ave => S Congress Avenue => South Congress Avenue
S Interstate Highway 35 => S Interstate Highway 35 => South Interstate Highway 35
S Capital of Texas Hwy => S Capital of Texas Hwy => South Capital of Texas Hwy
W US Highway 290 Service Rd => W US Highway 290 Service Road => West US Highway
290 Service Road
W Hwy 290 => W Hwy 290 => West Hwy 290
W 35th Street => W 35th Street => West 35th Street
W Hwy 71, STE D1 => W Hwy 71, STE D1 => West Hwy 71, STE D1
W Parmer Lane => W Parmer Lane => West Parmer Lane
W Annie St => W Annie Street => West Annie Street
W North Loop Blvd. => W North Loop Boulevard => West North Loop Boulevard
W Anderson Ln => W Anderson Lane => West Anderson Lane
W Lake Dr => W Lake Drive => West Lake Drive
W 6th St => W 6th Street => West 6th Street
W William Cannon Dr => W William Cannon Drive => West William Cannon Drive
W Hwy 71 => W Hwy 71 => West Hwy 71
W Highway 290 => W Highway 290 => West Highway 290
W HWY 290 => W HWY 290 => West HWY 290
```

## To load data into MongoDB I transformed the data.

The rules I chose included the following:

To only process two types of top level tags “node” and “way”.

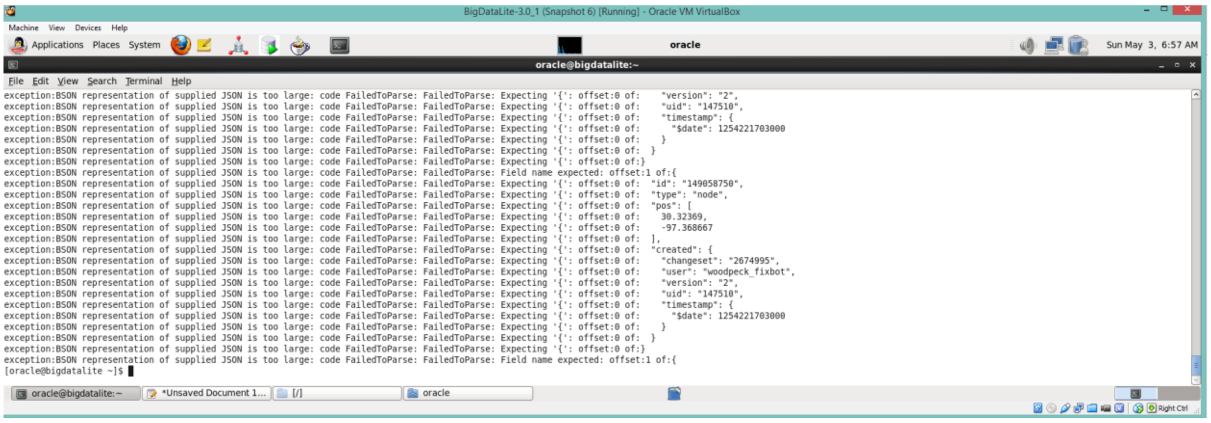
Attributes except from created: “version, changeset, timestamp, user, uid” should be added a key.
The longitudes and latitudes will be added to an array “pos”, to be used in a geospatial indexing, ensuring the values inside the “pos” are floats and not strings.
Problematics characters in the second level of the tag “k” would be ignored.
The dictionary in the second tag level with values that starts with “addr” but containing “:” should be added.
The presence of a second tag containing character “:” should be ignored .the character separates the type and direction of a street.

For the transformation will make use of a function that processes an element.
Within the function there is an update function that uses regex and mapping dictionaries to clean addresses.

## Overview Of the Data

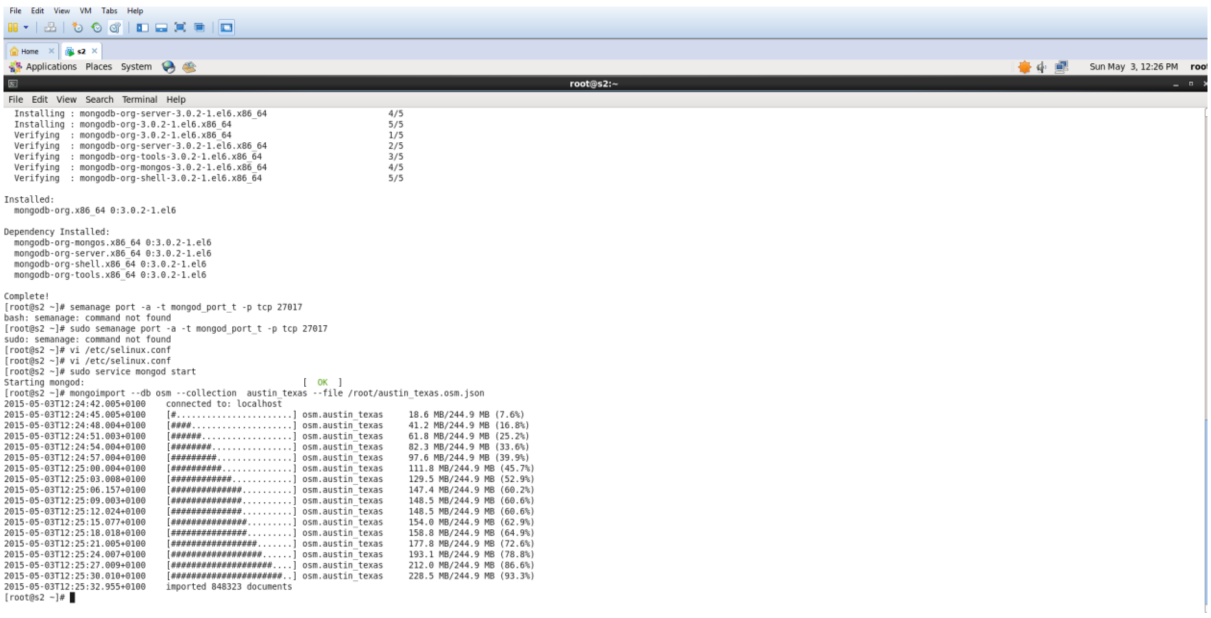
The downloaded file is 176.584186 MB
The json file is 256.771823 MB

The diagram below shows the json file being imported into MongoDB.



```
exception:BSON representation of supplied JSON is too large: code FailedToParse: FailedToParse: Expecting '{': offset:0 of:    "version": "2",
exception:BSON representation of supplied JSON is too large: code FailedToParse: FailedToParse: Expecting '{': offset:0 of:    "uid": "147510",
exception:BSON representation of supplied JSON is too large: code FailedToParse: FailedToParse: Expecting '{': offset:0 of:    "timestamp": {
exception:BSON representation of supplied JSON is too large: code FailedToParse: FailedToParse: Expecting '{': offset:0 of:      "$date": 1254221703000
exception:BSON representation of supplied JSON is too large: code FailedToParse: FailedToParse: Expecting '{': offset:0 of:    }
exception:BSON representation of supplied JSON is too large: code FailedToParse: FailedToParse: Expecting '{': offset:0 of:  }
exception:BSON representation of supplied JSON is too large: code FailedToParse: FailedToParse: Expecting '{': offset:0 of:}
exception:BSON representation of supplied JSON is too large: code FailedToParse: FailedToParse: Field name expected: offset:1 of:{
exception:BSON representation of supplied JSON is too large: code FailedToParse: FailedToParse: Expecting '{': offset:0 of:  "id": "149058750",
exception:BSON representation of supplied JSON is too large: code FailedToParse: FailedToParse: Expecting '{': offset:0 of:  "type": "node",
exception:BSON representation of supplied JSON is too large: code FailedToParse: FailedToParse: Expecting '{': offset:0 of:  "pos": [
exception:BSON representation of supplied JSON is too large: code FailedToParse: FailedToParse: Expecting '{': offset:0 of:    30.32369,
exception:BSON representation of supplied JSON is too large: code FailedToParse: FailedToParse: Expecting '{': offset:0 of:    -97.368667
exception:BSON representation of supplied JSON is too large: code FailedToParse: FailedToParse: Expecting '{': offset:0 of:  ],
exception:BSON representation of supplied JSON is too large: code FailedToParse: FailedToParse: Expecting '{': offset:0 of:  "created": {
exception:BSON representation of supplied JSON is too large: code FailedToParse: FailedToParse: Expecting '{': offset:0 of:    "changeset": "2674995",
exception:BSON representation of supplied JSON is too large: code FailedToParse: FailedToParse: Expecting '{': offset:0 of:    "user": "woodpeck_fixbot",
exception:BSON representation of supplied JSON is too large: code FailedToParse: FailedToParse: Expecting '{': offset:0 of:    "version": "2",
exception:BSON representation of supplied JSON is too large: code FailedToParse: FailedToParse: Expecting '{': offset:0 of:    "uid": "147510",
exception:BSON representation of supplied JSON is too large: code FailedToParse: FailedToParse: Expecting '{': offset:0 of:    "timestamp": {
exception:BSON representation of supplied JSON is too large: code FailedToParse: FailedToParse: Expecting '{': offset:0 of:      "$date": 1254221703000
exception:BSON representation of supplied JSON is too large: code FailedToParse: FailedToParse: Expecting '{': offset:0 of:    }
exception:BSON representation of supplied JSON is too large: code FailedToParse: FailedToParse: Expecting '{': offset:0 of:  }
exception:BSON representation of supplied JSON is too large: code FailedToParse: FailedToParse: Expecting '{': offset:0 of:}
exception:BSON representation of supplied JSON is too large: code FailedToParse: FailedToParse: Field name expected: offset:1 of:{
[oracle@bigdatalite ~]$ 
```

The diagram below shows the json file being imported into MongoDB.



```
  Installing : mongodb-org-server-3.0.2-1.el6.x86_64                              4/5
  Installing : mongodb-org-3.0.2-1.el6.x86_64                                     5/5
  Verifying  : mongodb-org-3.0.2-1.el6.x86_64                                     1/5
  Verifying  : mongodb-org-server-3.0.2-1.el6.x86_64                              2/5
  Verifying  : mongodb-org-tools-3.0.2-1.el6.x86_64                               3/5
  Verifying  : mongodb-org-mongos-3.0.2-1.el6.x86_64                              4/5
  Verifying  : mongodb-org-shell-3.0.2-1.el6.x86_64                               5/5

Installed:
  mongodb-org.x86_64 0:3.0.2-1.el6

Dependency Installed:
  mongodb-org-mongos.x86_64 0:3.0.2-1.el6
  mongodb-org-server.x86_64 0:3.0.2-1.el6
  mongodb-org-shell.x86_64 0:3.0.2-1.el6
  mongodb-org-tools.x86_64 0:3.0.2-1.el6

Complete!
[root@s2 ~]# semanage port -a -t mongod_port_t -p tcp 27017
bash: semanage: command not found
[root@s2 ~]# sudo semanage port -a -t mongod_port_t -p tcp 27017
sudo: semanage: command not found
[root@s2 ~]# vi /etc/selinux.conf
[root@s2 ~]# vi /etc/selinux.conf
[root@s2 ~]# sudo service mongod start
Starting mongod:                                           [  OK  ]
[root@s2 ~]# mongoimport --db osm --collection  austin_texas --file /root/austin_texas.osm.json
2015-05-03T12:24:42.005+0100    connected to: localhost
2015-05-03T12:24:45.005+0100    [#.......................] osm.austin_texas     18.6 MB/244.9 MB (7.6%)
2015-05-03T12:24:48.004+0100    [####....................] osm.austin_texas     41.2 MB/244.9 MB (16.8%)
2015-05-03T12:24:51.003+0100    [######..................] osm.austin_texas     61.8 MB/244.9 MB (25.2%)
2015-05-03T12:24:54.004+0100    [########................] osm.austin_texas     82.3 MB/244.9 MB (33.6%)
2015-05-03T12:24:57.004+0100    [#########...............] osm.austin_texas     97.6 MB/244.9 MB (39.9%)
2015-05-03T12:25:00.004+0100    [##########..............] osm.austin_texas     111.8 MB/244.9 MB (45.7%)
2015-05-03T12:25:03.008+0100    [############............] osm.austin_texas     129.5 MB/244.9 MB (52.9%)
2015-05-03T12:25:06.157+0100    [##############..........] osm.austin_texas     147.4 MB/244.9 MB (60.2%)
2015-05-03T12:25:09.003+0100    [##############..........] osm.austin_texas     148.5 MB/244.9 MB (60.6%)
2015-05-03T12:25:12.024+0100    [##############..........] osm.austin_texas     148.5 MB/244.9 MB (60.6%)
2015-05-03T12:25:15.077+0100    [###############.........] osm.austin_texas     154.0 MB/244.9 MB (62.9%)
2015-05-03T12:25:18.018+0100    [###############.........] osm.austin_texas     158.8 MB/244.9 MB (64.9%)
2015-05-03T12:25:21.005+0100    [#################.......] osm.austin_texas     177.8 MB/244.9 MB (72.6%)
2015-05-03T12:25:24.007+0100    [##################......] osm.austin_texas     193.1 MB/244.9 MB (78.8%)
2015-05-03T12:25:27.009+0100    [####################....] osm.austin_texas     212.0 MB/244.9 MB (86.6%)
2015-05-03T12:25:30.010+0100    [######################..] osm.austin_texas     228.5 MB/244.9 MB (93.3%)
2015-05-03T12:25:32.955+0100    imported 848323 documents
[root@s2 ~]# 
```

Using the following bit of code to connect to the MongoDB database, I was able to determine the number of documents.

```
db_name = "osm"

client =
MongoClient('192.168.255.133:27017')
db = client[db_name]

collection = db.austin_texas

print collection.count()
```

**Result : 848323**

The number of unique users can be determined by modifying the previous code to include:

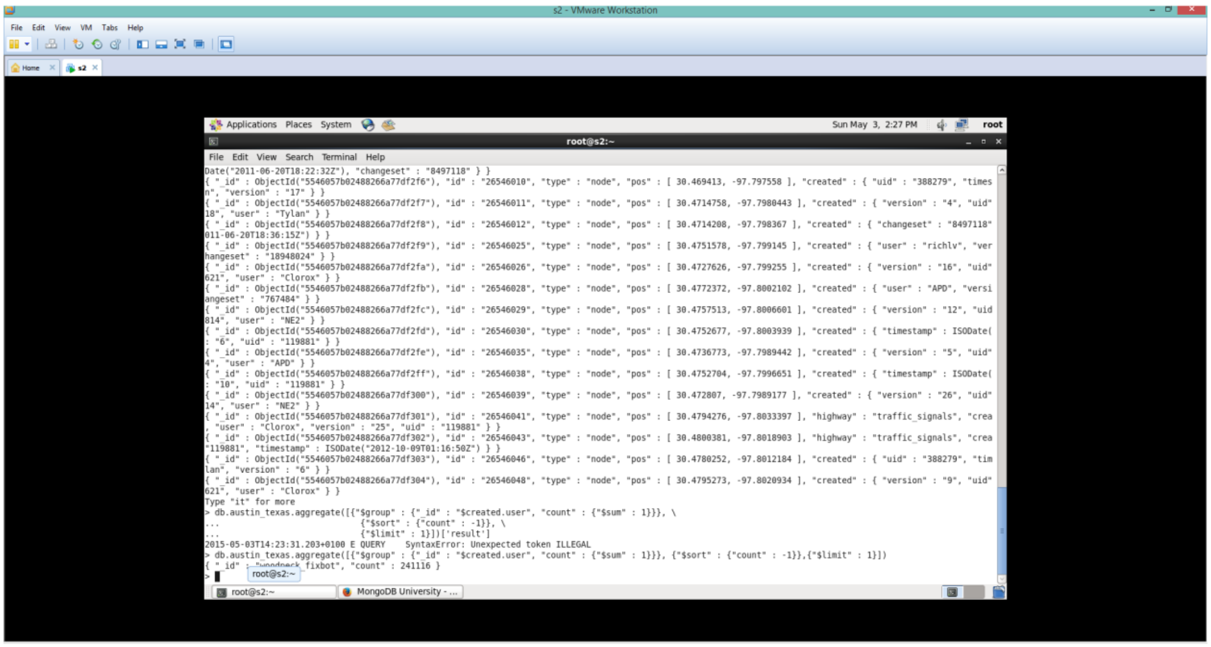
```
Print
len(collection.distinct('created.user'))
```

**Result: 910**

The top contributing user was known by running the following code :

```
db.austin_texas.aggregate([{"$group" :
{"_id" : "$created.user", "count" :
{"$sum" : 1}}}, {"$sort" : {"count" : -
1}},{"$limit" : 1}])
```

```
{ "_id" : "woodpeck_fixbot", "count" :
241116 }
```



s2 - VMware Workstation
File Edit View VM Tabs Help
Home
s2
Applications Places System
Sun May 3, 2:27 PM
root
root@s2:~
File Edit View Search Terminal Help

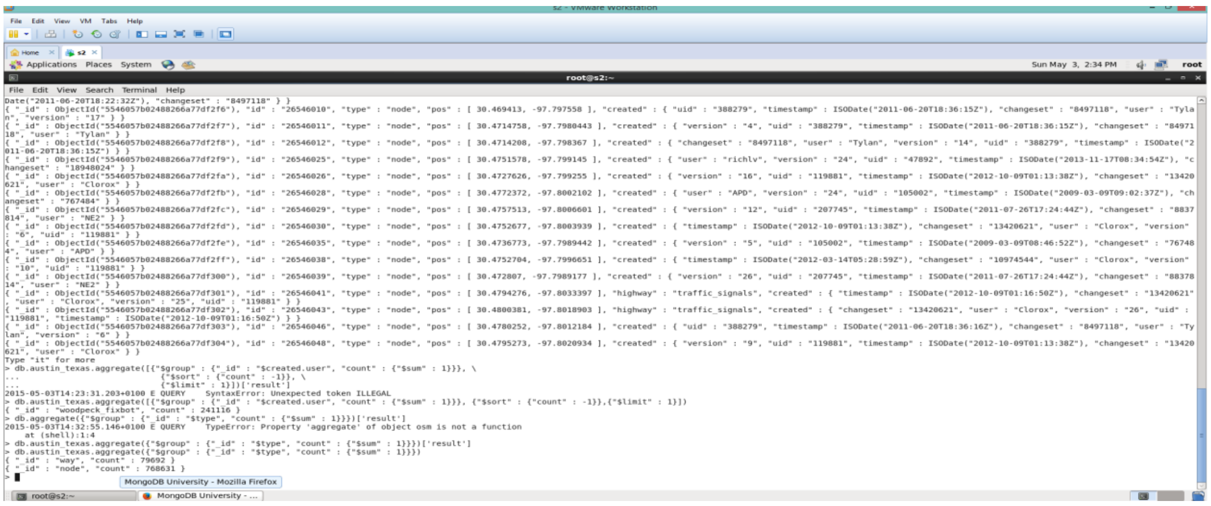
```
Date("2011-06-20T18:22:32Z"), "changeset" : "8497118" } }
{ "_id" : ObjectId("5546057b02488266a77df2f6"), "id" : "26546010", "type" : "node", "pos" : [ 30.469413, -97.797558 ], "created" : { "uid" : "388279", "times
n", "version" : "17" } }
{ "_id" : ObjectId("5546057b02488266a77df2f7"), "id" : "26546011", "type" : "node", "pos" : [ 30.4714758, -97.7980443 ], "created" : { "version" : "4", "uid"
18", "user" : "Tylan" } }
{ "_id" : ObjectId("5546057b02488266a77df2f8"), "id" : "26546012", "type" : "node", "pos" : [ 30.4714208, -97.798367 ], "created" : { "changeset" : "8497118"
011-06-20T18:36:15Z") } }
{ "_id" : ObjectId("5546057b02488266a77df2f9"), "id" : "26546025", "type" : "node", "pos" : [ 30.4751578, -97.799145 ], "created" : { "user" : "richlv", "ver
hangeset" : "18948024" } }
{ "_id" : ObjectId("5546057b02488266a77df2fa"), "id" : "26546026", "type" : "node", "pos" : [ 30.4727626, -97.799255 ], "created" : { "version" : "16", "uid"
621", "user" : "Clorox" } }
{ "_id" : ObjectId("5546057b02488266a77df2fb"), "id" : "26546028", "type" : "node", "pos" : [ 30.4772372, -97.8002102 ], "created" : { "user" : "APD", "versi
angeset" : "767484" } }
{ "_id" : ObjectId("5546057b02488266a77df2fc"), "id" : "26546029", "type" : "node", "pos" : [ 30.4757513, -97.8006601 ], "created" : { "version" : "12", "uid
814", "user" : "NE2" } }
{ "_id" : ObjectId("5546057b02488266a77df2fd"), "id" : "26546030", "type" : "node", "pos" : [ 30.4752677, -97.8003939 ], "created" : { "timestamp" : ISODate(
: "6", "uid" : "119881" } }
{ "_id" : ObjectId("5546057b02488266a77df2fe"), "id" : "26546035", "type" : "node", "pos" : [ 30.4736773, -97.7989442 ], "created" : { "version" : "5", "uid"
4", "user" : "APD" } }
{ "_id" : ObjectId("5546057b02488266a77df2ff"), "id" : "26546038", "type" : "node", "pos" : [ 30.4752704, -97.7996651 ], "created" : { "timestamp" : ISODate(
: "10", "uid" : "119881" } }
{ "_id" : ObjectId("5546057b02488266a77df300"), "id" : "26546039", "type" : "node", "pos" : [ 30.472807, -97.7989177 ], "created" : { "version" : "26", "uid"
14", "user" : "NE2" } }
{ "_id" : ObjectId("5546057b02488266a77df301"), "id" : "26546041", "type" : "node", "pos" : [ 30.4794276, -97.8033397 ], "highway" : "traffic_signals", "crea
, "user" : "Clorox", "version" : "25", "uid" : "119881" } }
{ "_id" : ObjectId("5546057b02488266a77df302"), "id" : "26546043", "type" : "node", "pos" : [ 30.4800381, -97.8018903 ], "highway" : "traffic_signals", "crea
"119881", "timestamp" : ISODate("2012-10-09T01:16:50Z") } }
{ "_id" : ObjectId("5546057b02488266a77df303"), "id" : "26546046", "type" : "node", "pos" : [ 30.4780252, -97.8012184 ], "created" : { "uid" : "388279", "tim
lan", "version" : "6" } }
{ "_id" : ObjectId("5546057b02488266a77df304"), "id" : "26546048", "type" : "node", "pos" : [ 30.4795273, -97.8020934 ], "created" : { "version" : "9", "uid"
621", "user" : "Clorox" } }
Type "it" for more
> db.austin_texas.aggregate([{"$group" : {"_id" : "$created.user", "count" : {"$sum" : 1}}}, \
...                              {"$sort" : {"count" : -1}}, \
...                              {"$limit" : 1}])['result']
2015-05-03T14:23:31.203+0100 E QUERY    SyntaxError: Unexpected token ILLEGAL
> db.austin_texas.aggregate([{"$group" : {"_id" : "$created.user", "count" : {"$sum" : 1}}}, {"$sort" : {"count" : -1}},{"$limit" : 1}])
{ "_id" : "woodpeck_fixbot", "count" : 241116 }
> 
```

The number of nodes and ways can be known by running the following code in the database (within MongoDB) :

```
aggregate({"$group" : {"_id" : "$type",
"count" : {"$sum" : 1}}})['result']
```

To give :

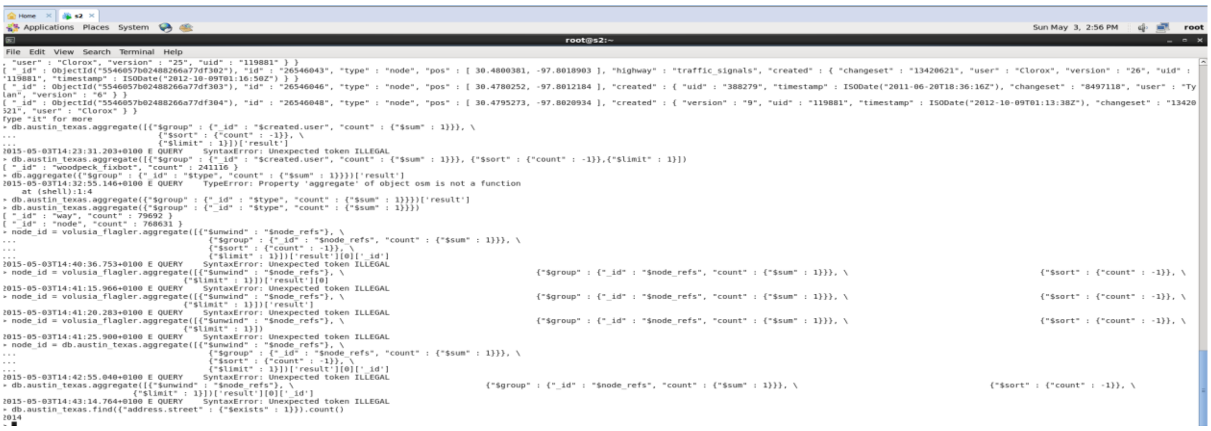
```
{ "_id" : "way", "count" : 79692 }
{ "_id" : "node", "count" : 768631 }
```



```
Date("2011-06-20T18:22:32Z"), "changeset" : "8497118" } }
{ "_id" : ObjectId("5546057b02488266a77df2f6"), "id" : "26546010", "type" : "node", "pos" : [ 30.469413, -97.797558 ], "created" : { "uid" : "388279", "timestamp" : ISODate("2011-06-20T18:36:15Z"), "changeset" : "8497118", "user" : "Tyla
n", "version" : "17" } }
{ "_id" : ObjectId("5546057b02488266a77df2f7"), "id" : "26546011", "type" : "node", "pos" : [ 30.4714758, -97.7980443 ], "created" : { "version" : "4", "uid" : "388279", "timestamp" : ISODate("2011-06-20T18:36:15Z"), "changeset" : "84971
18", "user" : "Tylan" } }
{ "_id" : ObjectId("5546057b02488266a77df2f8"), "id" : "26546012", "type" : "node", "pos" : [ 30.4714208, -97.798367 ], "created" : { "changeset" : "8497118", "user" : "Tylan", "version" : "14", "uid" : "388279", "timestamp" : ISODate("2
011-06-20T18:36:15Z") } }
{ "_id" : ObjectId("5546057b02488266a77df2f9"), "id" : "26546025", "type" : "node", "pos" : [ 30.4751578, -97.799145 ], "created" : { "user" : "richlv", "version" : "24", "uid" : "47892", "timestamp" : ISODate("2013-11-17T08:34:54Z"), "c
hangeset" : "18948024" } }
{ "_id" : ObjectId("5546057b02488266a77df2fa"), "id" : "26546026", "type" : "node", "pos" : [ 30.4727626, -97.799255 ], "created" : { "version" : "16", "uid" : "119881", "timestamp" : ISODate("2012-10-09T01:13:38Z"), "changeset" : "13420
621", "user" : "Clorox" } }
{ "_id" : ObjectId("5546057b02488266a77df2fb"), "id" : "26546028", "type" : "node", "pos" : [ 30.4772372, -97.8002102 ], "created" : { "user" : "APD", "version" : "24", "uid" : "105002", "timestamp" : ISODate("2009-03-09T09:02:37Z"), "ch
angeset" : "767484" } }
{ "_id" : ObjectId("5546057b02488266a77df2fc"), "id" : "26546029", "type" : "node", "pos" : [ 30.4757513, -97.8006601 ], "created" : { "version" : "12", "uid" : "207745", "timestamp" : ISODate("2011-07-26T17:24:44Z"), "changeset" : "8837
814", "user" : "NE2" } }
{ "_id" : ObjectId("5546057b02488266a77df2fd"), "id" : "26546030", "type" : "node", "pos" : [ 30.4752677, -97.8003939 ], "created" : { "timestamp" : ISODate("2012-10-09T01:13:38Z"), "changeset" : "13420621", "user" : "Clorox", "version"
: "6", "uid" : "119881" } }
{ "_id" : ObjectId("5546057b02488266a77df2fe"), "id" : "26546035", "type" : "node", "pos" : [ 30.4736773, -97.7989442 ], "created" : { "version" : "5", "uid" : "105002", "timestamp" : ISODate("2009-03-09T08:46:52Z"), "changeset" : "76748
4", "user" : "APD" } }
{ "_id" : ObjectId("5546057b02488266a77df2ff"), "id" : "26546038", "type" : "node", "pos" : [ 30.4752704, -97.7996651 ], "created" : { "timestamp" : ISODate("2012-03-14T05:28:59Z"), "changeset" : "10974544", "user" : "Clorox", "version"
: "10", "uid" : "119881" } }
{ "_id" : ObjectId("5546057b02488266a77df300"), "id" : "26546039", "type" : "node", "pos" : [ 30.472807, -97.7989177 ], "created" : { "version" : "26", "uid" : "207745", "timestamp" : ISODate("2011-07-26T17:24:44Z"), "changeset" : "88378
14", "user" : "NE2" } }
{ "_id" : ObjectId("5546057b02488266a77df301"), "id" : "26546041", "type" : "node", "pos" : [ 30.4794276, -97.8033397 ], "highway" : "traffic_signals", "created" : { "timestamp" : ISODate("2012-10-09T01:16:50Z"), "changeset" : "13420621"
, "user" : "Clorox", "version" : "25", "uid" : "119881" } }
{ "_id" : ObjectId("5546057b02488266a77df302"), "id" : "26546043", "type" : "node", "pos" : [ 30.4800381, -97.8018903 ], "highway" : "traffic_signals", "created" : { "changeset" : "13420621", "user" : "Clorox", "version" : "26", "uid" :
"119881", "timestamp" : ISODate("2012-10-09T01:16:50Z") } }
{ "_id" : ObjectId("5546057b02488266a77df303"), "id" : "26546046", "type" : "node", "pos" : [ 30.4780252, -97.8012184 ], "created" : { "uid" : "388279", "timestamp" : ISODate("2011-06-20T18:36:16Z"), "changeset" : "8497118", "user" : "Ty
lan", "version" : "6" } }
{ "_id" : ObjectId("5546057b02488266a77df304"), "id" : "26546048", "type" : "node", "pos" : [ 30.4795273, -97.8020934 ], "created" : { "version" : "9", "uid" : "119881", "timestamp" : ISODate("2012-10-09T01:13:38Z"), "changeset" : "13420
621", "user" : "Clorox" } }
Type "it" for more
> db.austin_texas.aggregate([{"$group" : {"_id" : "$created.user", "count" : {"$sum" : 1}}}, \
...                           {"$sort" : {"count" : -1}}, \
...                           {"$limit" : 1}])['result']
2015-05-03T14:23:31.203+0100 E QUERY    SyntaxError: Unexpected token ILLEGAL
> db.austin_texas.aggregate([{"$group" : {"_id" : "$created.user", "count" : {"$sum" : 1}}}, {"$sort" : {"count" : -1}},{"$limit" : 1}])
{ "_id" : "woodpeck_fixbot", "count" : 241116 }
> db.aggregate({"$group" : {"_id" : "$type", "count" : {"$sum" : 1}}})['result']
2015-05-03T14:32:55.146+0100 E QUERY    TypeError: Property 'aggregate' of object osm is not a function
    at (shell):1:4
> db.austin_texas.aggregate({"$group" : {"_id" : "$type", "count" : {"$sum" : 1}}})['result']
> db.austin_texas.aggregate({"$group" : {"_id" : "$type", "count" : {"$sum" : 1}}})
{ "_id" : "way", "count" : 79692 }
{ "_id" : "node", "count" : 768631 }
> 
```

To determine the number of containing a street address, I modified the previous code (within MongoDB) to use:
db.austin_texas.find({"address.street" : {"$exists" : 1}}).count()

**Results: 2014**



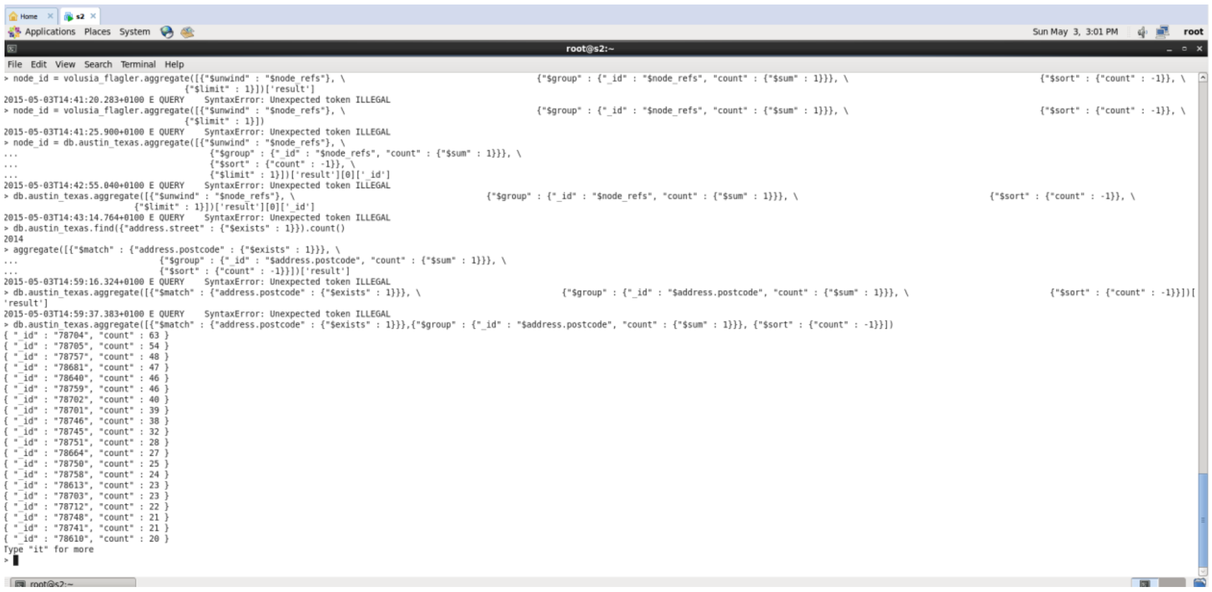
```
, "user" : "Clorox", "version" : "25", "uid" : "119881" } }
{ "_id" : ObjectId("5546057b02488266a77df302"), "id" : "26546043", "type" : "node", "pos" : [ 30.4800381, -97.8018903 ], "highway" : "traffic_signals", "created" : { "changeset" : "13420621", "user" : "Clorox", "version" : "26", "uid" :
"119881", "timestamp" : ISODate("2012-10-09T01:16:50Z") } }
{ "_id" : ObjectId("5546057b02488266a77df303"), "id" : "26546046", "type" : "node", "pos" : [ 30.4780252, -97.8012184 ], "created" : { "uid" : "388279", "timestamp" : ISODate("2011-06-20T18:36:16Z"), "changeset" : "8497118", "user" : "Ty
lan", "version" : "6" } }
{ "_id" : ObjectId("5546057b02488266a77df304"), "id" : "26546048", "type" : "node", "pos" : [ 30.4795273, -97.8020934 ], "created" : { "version" : "9", "uid" : "119881", "timestamp" : ISODate("2012-10-09T01:13:38Z"), "changeset" : "13420
621", "user" : "Clorox" } }
Type "it" for more
> db.austin_texas.aggregate([{"$group" : {"_id" : "$created.user", "count" : {"$sum" : 1}}}, \
...                           {"$sort" : {"count" : -1}}, \
...                           {"$limit" : 1}])['result']
2015-05-03T14:23:31.203+0100 E QUERY    SyntaxError: Unexpected token ILLEGAL
> db.austin_texas.aggregate([{"$group" : {"_id" : "$created.user", "count" : {"$sum" : 1}}}, {"$sort" : {"count" : -1}},{"$limit" : 1}])
{ "_id" : "woodpeck_fixbot", "count" : 241116 }
> db.aggregate({"$group" : {"_id" : "$type", "count" : {"$sum" : 1}}})['result']
2015-05-03T14:32:55.146+0100 E QUERY    TypeError: Property 'aggregate' of object osm is not a function
    at (shell):1:4
> db.austin_texas.aggregate({"$group" : {"_id" : "$type", "count" : {"$sum" : 1}}})['result']
> db.austin_texas.aggregate({"$group" : {"_id" : "$type", "count" : {"$sum" : 1}}})
{ "_id" : "way", "count" : 79692 }
{ "_id" : "node", "count" : 768631 }
> node_id = volusia_flagler.aggregate([{"$unwind" : "$node_refs"}, \
...                                    {"$group" : {"_id" : "$node_refs", "count" : {"$sum" : 1}}}, \
...                                    {"$sort" : {"count" : -1}}, \
...                                    {"$limit" : 1}])['result'][0]['_id']
2015-05-03T14:40:36.753+0100 E QUERY    SyntaxError: Unexpected token ILLEGAL
> node_id = volusia_flagler.aggregate([{"$unwind" : "$node_refs"}, \                        {"$group" : {"_id" : "$node_refs", "count" : {"$sum" : 1}}}, \                        {"$sort" : {"count" : -1}}, \
                                {"$limit" : 1}])['result'][0]
2015-05-03T14:41:15.966+0100 E QUERY    SyntaxError: Unexpected token ILLEGAL
> node_id = volusia_flagler.aggregate([{"$unwind" : "$node_refs"}, \                        {"$group" : {"_id" : "$node_refs", "count" : {"$sum" : 1}}}, \                        {"$sort" : {"count" : -1}}, \
                                {"$limit" : 1}])['result']
2015-05-03T14:41:20.283+0100 E QUERY    SyntaxError: Unexpected token ILLEGAL
> node_id = volusia_flagler.aggregate([{"$unwind" : "$node_refs"}, \                        {"$group" : {"_id" : "$node_refs", "count" : {"$sum" : 1}}}, \                        {"$sort" : {"count" : -1}}, \
                                {"$limit" : 1}])
2015-05-03T14:41:25.900+0100 E QUERY    SyntaxError: Unexpected token ILLEGAL
> node_id = db.austin_texas.aggregate([{"$unwind" : "$node_refs"}, \
...                                    {"$group" : {"_id" : "$node_refs", "count" : {"$sum" : 1}}}, \
...                                    {"$sort" : {"count" : -1}}, \
...                                    {"$limit" : 1}])['result'][0]['_id']
2015-05-03T14:42:55.040+0100 E QUERY    SyntaxError: Unexpected token ILLEGAL
> db.austin_texas.aggregate([{"$unwind" : "$node_refs"}, \                    {"$group" : {"_id" : "$node_refs", "count" : {"$sum" : 1}}}, \                    {"$sort" : {"count" : -1}}, \
                        {"$limit" : 1}])['result'][0]['_id']
2015-05-03T14:43:14.764+0100 E QUERY    SyntaxError: Unexpected token ILLEGAL
> db.austin_texas.find({"address.street" : {"$exists" : 1}}).count()
2014
> 
```

To display a list of cleaned zip codes I used the following code (within MongoDB) :
db.austin_texas.aggregate([{"$match" : {"address.postcode" : {"$exists" :

```
1}}}},{"$group" : {"_id" :
"$address.postcode", "count" : {"$sum" :
1}}}, {"$sort" : {"count" : -1}}])
```

**Results :**

```
{ "_id" : "78704", "count" : 63 }
{ "_id" : "78705", "count" : 54 }
{ "_id" : "78757", "count" : 48 }
{ "_id" : "78681", "count" : 47 }
{ "_id" : "78640", "count" : 46 }
{ "_id" : "78759", "count" : 46 }
{ "_id" : "78702", "count" : 40 }
{ "_id" : "78701", "count" : 39 }
{ "_id" : "78746", "count" : 38 }
{ "_id" : "78745", "count" : 32 }
```


Home
s2
Applications Places System
Sun May 3, 3:01 PM
root
root@s2:~
File Edit View Search Terminal Help
root@s2:~

```
> node_id = volusia_flagler.aggregate([{"$unwind" : "$node_refs"}, \                    {"$group" : {"_id" : "$node_refs", "count" : {"$sum" : 1}}}, \                    {"$sort" : {"count" : -1}}, \
                                  {"$limit" : 1}])['result']
2015-05-03T14:41:20.283+0100 E QUERY    SyntaxError: Unexpected token ILLEGAL
> node_id = volusia_flagler.aggregate([{"$unwind" : "$node_refs"}, \                    {"$group" : {"_id" : "$node_refs", "count" : {"$sum" : 1}}}, \                    {"$sort" : {"count" : -1}}, \
                                  {"$limit" : 1}])
2015-05-03T14:41:25.900+0100 E QUERY    SyntaxError: Unexpected token ILLEGAL
> node_id = db.austin_texas.aggregate([{"$unwind" : "$node_refs"}, \
...                                   {"$group" : {"_id" : "$node_refs", "count" : {"$sum" : 1}}}, \
...                                   {"$sort" : {"count" : -1}}, \
...                                   {"$limit" : 1}])['result'][0]['_id']
2015-05-03T14:42:55.040+0100 E QUERY    SyntaxError: Unexpected token ILLEGAL
> db.austin_texas.aggregate([{"$unwind" : "$node_refs"}, \                    {"$group" : {"_id" : "$node_refs", "count" : {"$sum" : 1}}}, \                    {"$sort" : {"count" : -1}}, \
                         {"$limit" : 1}])['result'][0]['_id']
2015-05-03T14:43:14.764+0100 E QUERY    SyntaxError: Unexpected token ILLEGAL
> db.austin_texas.find({"address.street" : {"$exists" : 1}}).count()
2014
> aggregate([{"$match" : {"address.postcode" : {"$exists" : 1}}}, \
...                           {"$group" : {"_id" : "$address.postcode", "count" : {"$sum" : 1}}}, \
...                           {"$sort" : {"count" : -1}}])['result']
2015-05-03T14:59:16.324+0100 E QUERY    SyntaxError: Unexpected token ILLEGAL
> db.austin_texas.aggregate([{"$match" : {"address.postcode" : {"$exists" : 1}}}, \                    {"$group" : {"_id" : "$address.postcode", "count" : {"$sum" : 1}}}, \                    {"$sort" : {"count" : -1}}])[
'result']
2015-05-03T14:59:37.383+0100 E QUERY    SyntaxError: Unexpected token ILLEGAL
> db.austin_texas.aggregate([{"$match" : {"address.postcode" : {"$exists" : 1}}},{"$group" : {"_id" : "$address.postcode", "count" : {"$sum" : 1}}}, {"$sort" : {"count" : -1}}])
{ "_id" : "78704", "count" : 63 }
{ "_id" : "78705", "count" : 54 }
{ "_id" : "78757", "count" : 48 }
{ "_id" : "78681", "count" : 47 }
{ "_id" : "78640", "count" : 46 }
{ "_id" : "78759", "count" : 46 }
{ "_id" : "78702", "count" : 40 }
{ "_id" : "78701", "count" : 39 }
{ "_id" : "78746", "count" : 38 }
{ "_id" : "78745", "count" : 32 }
{ "_id" : "78751", "count" : 28 }
{ "_id" : "78664", "count" : 27 }
{ "_id" : "78750", "count" : 25 }
{ "_id" : "78758", "count" : 24 }
{ "_id" : "78613", "count" : 23 }
{ "_id" : "78703", "count" : 23 }
{ "_id" : "78712", "count" : 22 }
{ "_id" : "78748", "count" : 21 }
{ "_id" : "78741", "count" : 21 }
{ "_id" : "78610", "count" : 20 }
Type "it" for more
> 
```

To determine the top five cites most common using the following code (within MongoDB) :

```
db.austin_texas.aggregate([{"$match" :
{"address.city" : {"$exists" : 1}}},
{"$group" : {"_id" : "$address.city",
"count" : {"$sum" : 1}}}, {"$sort" :
{"count" : -1}}, {"$limit" : 5}])
```

**Results :**

```
{ "_id" : "Austin", "count" : 541 }
{ "_id" : "Round Rock", "count" : 65 }
{ "_id" : "Austin, TX", "count" : 48 }
{ "_id" : "Kyle", "count" : 47 }
{ "_id" : "Buda", "count" : 21 }
```

To determine the top ten amenities using
: db.austin_texas.aggregate([{"$match" :
{"amenity" : {"$exists" : 1}}}, {"$group"
: {"_id" : "$amenity", "count" : {"$sum"
: 1}}}, {"$sort" : {"count" : -1}},
{"$limit" : 10}])

**Results :**

```
{ "_id" : "parking", "count" : 1845 }
{ "_id" : "restaurant", "count" : 684 }
{ "_id" : "waste_basket", "count" : 591 }
{ "_id" : "school", "count" : 562 }
{ "_id" : "fast_food", "count" : 503 }
{ "_id" : "place_of_worship", "count" :
488 }
{ "_id" : "fuel", "count" : 369 }
{ "_id" : "bench", "count" : 349 }
{ "_id" : "shelter", "count" : 231 }
{ "_id" : "bank", "count" : 150 }
```

Top religions with denominations can be determined by running the following code in mongodb to give :

```
db.austin_texas.aggregate([{"$match" :
{"amenity" :
"place_of_worship"}},{"$group" : {"_id" :
{"religion" : "$religion", "denomination"
: "$denomination"}, "count" : {"$sum" :
1}}},{"$sort" : {"count" : -1}}])
```

**Results :**

```
{ "_id" : { "religion" : "christian" },
"count" : 196 }
{ "_id" : { "religion" : "christian",
"denomination" : "baptist" }, "count" :
108 }
{ "_id" : { "religion" : "christian",
"denomination" : "lutheran" }, "count" :
30 }
```

```
{ "_id" : { "religion" : "christian",
"denomination" : "methodist" }, "count" :
29 }
{ "_id" : {  }, "count" : 25 }
{ "_id" : { "religion" : "christian",
"denomination" : "catholic" }, "count" :
22 }
{ "_id" : { "religion" : "christian",
"denomination" : "presbyterian" },
"count" : 21 }
{ "_id" : { "religion" : "christian",
"denomination" : "pentecostal" }, "count"
: 12 }
{ "_id" : { "religion" : "christian",
"denomination" : "mormon" }, "count" : 6
}
{ "_id" : { "religion" : "buddhist" },
"count" : 5 }
{ "_id" : { "religion" : "christian",
"denomination" : "roman_catholic" },
"count" : 5 }
{ "_id" : { "religion" : "christian",
"denomination" : "seventh_day_adventist"
}, "count" : 4 }
{ "_id" : { "religion" : "jewish" },
"count" : 3 }
{ "_id" : { "religion" : "christian",
"denomination" : "jehovahs_witness" },
"count" : 2 }
```

```
{ "_id" : { "religion" : "christian",
"denomination" : "anglican" }, "count" :
2 }
{ "_id" : { "religion" : "muslim" },
"count" : 2 }
{ "_id" : { "religion" :
"unitarian_universalist" }, "count" : 2 }
{ "_id" : { "religion" : "christian",
"denomination" : "episcopal" }, "count" :
2 }
{ "_id" : { "religion" : "christian",
"denomination" : "mormon3" }, "count" : 1
}
{ "_id" : { "denomination" : "baptist" },
"count" : 1 }
```

The top 10 leisure within the data can be determined using

```
db.austin_texas.aggregate([{"$match" :
{"leisure" : {"$exists" : 1}}}, {"$group"
: {"_id" : "$leisure", "count" : {"$sum"
: 1}}}, {"$sort" : {"count" : -1}},
{"$limit" : 10}])
```

**Results :**

```
{ "_id" : "pitch", "count" : 828 }
{ "_id" : "park", "count" : 421 }
{ "_id" : "playground", "count" : 104 }
{ "_id" : "sports_centre", "count" : 70 }
{ "_id" : "golf_course", "count" : 70 }
{ "_id" : "track", "count" : 64 }
{ "_id" : "swimming_pool", "count" : 55 }
{ "_id" : "garden", "count" : 21 }
{ "_id" : "stadium", "count" : 20 }
{ "_id" : "slipway", "count" : 12 }
```

## About the Dataset:

I believe the data is well formed and structured making it suitable for extension whereby user will be able to add reviews with regards to bicycle routes, restaurants and schools etc.