test_scientific_export

May 1, 2019

[131]:
```
# -*- coding: utf-8 -*-


import json
import cchardet as chardet
from pprint import pprint
from tqdm import tnrange, tqdm_notebook
from time import sleep
from bs4 import BeautifulSoup
from pymongo import MongoClient
from pymongo.mongo_client import database
from pymongo.collection import Collection
import pandas as pd
import requests
from pprint import pprint
from pymongo import IndexModel, ASCENDING, DESCENDING

def get_suffix_mapping():
    """get a mapping for street suffixes"""
    headers = {'User-agent': 'Mozilla/5.0'}
    r = requests.get(r"https://pe.usps.com/text/pub28/28apc_002.htm",␣
 ↪headers=headers)

    soup = BeautifulSoup(r.content, 'lxml')

    postal_table = soup.find('table', {'id':'ep533076'})

    df = pd.read_html(str(postal_table), header=0)[0]
    df.columns = ['Primary', 'Common', 'Standard']



    from collections import OrderedDict, defaultdict
    keys = df.iloc[:,1:3]
```

```python
    return keys.to_dict(orient='records')

def tqdm_ipython_test():
    for i in tnrange(3, desc='1st loop'):
        for j in tqdm_notebook(range(100), desc='2nd loop'):
            sleep(0.01)


def read_osm_file(filename: str):
    with open(filename, "r", encoding='UTF-8') as f:
        msg = f.read()
        # result = chardet.detect(msg)
    return msg


def get_soup(file, tags):
    soup = BeautifulSoup(file, 'xml')
    return [{tag: soup.find_all(tag)} for tag in tqdm_notebook(tags)]


def get_dict_data(result_set_item):
    list_of_dicts = []
    for k, v in result_set_item:
        primary_tag = k
        result_set = v
    for entry in result_set:
        entry_data_dict = {}
        entry_data_dict['type'] = primary_tag
        for k, v in entry.attrs.items():
            entry_data_dict[k] = v
        for tag in entry.find_all('tag'):
            entry_data_dict[tag['k']] = tag['v']
        list_of_dicts.append(entry_data_dict)
    return list_of_dicts


def json_to_mongo(col: database.Collection, json_file: str ="rochester_osm.
 ↪json" ):
    # data = []
    with open(json_file) as f:
        data = json.load(f)
        #for line in f:
        #    data.append(json.loads(line))
    for node_dict in data:
        col.insert_many(node_dict)
    # return col.insert_many(data)
```

```


def get_col(db_name="udacity", collection="rochester_osm"):
    from pymongo import MongoClient
    client = MongoClient('localhost:27017')
    db = client[db_name]
    col = db[collection]
    return col


def main():
    file_name = "rochester_ny.osm"
    osm_file = read_osm_file(filename=file_name)
    tag_list = ['node', 'way']
    result_set_list = get_soup(file=osm_file, tags=tag_list)
    osm_dicts = [get_dict_data(res) for res in result_set_list.values()]
    with open("osm_dicts", 'w') as f:
        f.write(osm_dicts)
```

0.0.1 Code to convert the osm to json in preparation for insertion to MongoDB

```
# Loading The osm file
file_name = r"rochester_ny.osm"
osm_file = read_osm_file(filename=file_name)
# Loading the osm file into beautiful soup and grabbing all node and way tags
tag_list = ['node', 'way']
%time result_set_list = get_soup(file=osm_file, tags=tag_list)
# make list of dictionaries containing the attribute and tag data for the result set
osm_dicts = [get_dict_data(res.items()) for res in result_set_list]
# osm_dicts = [get_dict_data(res) for res in result_set_list.values()]
# dump this dict data to a json object so that parsing doesn't need to be re-run
json_osm = json.dumps(osm_dicts)
# write the json to file
with open('rochester_osm.json', 'w') as f:
    f.write(json_osm)
```

0.0.2 Initial MongoDB collection creation

- Insert all records from json file
- Create compound unique index on ‘id’ and ‘type’ fields

[132]:
```
# read the json file we just read to verify it's working
from importlib import import_module
j2m = import_module('json_to_mongo')
%time j2m.main()
```

```
457947 records inserted from rochester_osm.json
[{'key': SON([('_id', 1)]),
  'name': '_id_',
  'ns': 'udacity.rochester_osm',
  'v': 2},
 {'key': SON([('id', 1), ('type', -1)]),
  'name': 'id_type_unique_index',
  'ns': 'udacity.rochester_osm',
  'unique': True,
  'v': 2}]
Wall time: 15.2 s
```

[133]:
```
# setup connection for data exploration and cleaning
osm_col = get_col() # type: MongoClient
```

0.0.3 Query total document count

[134]:
```
total_docs = osm_col.count_documents({})
total_docs
```

[134]:
```
457947
```

0.0.4 Get count of each key in collection

[135]:
```
key_counts_dict = dict()
for entry in tqdm_notebook(osm_col.find(), total=total_docs):
    for k in entry.keys():
        key_counts_dict.setdefault(k, 0)
        key_counts_dict[k] += 1
```

```
HBox(children=(IntProgress(value=0, max=457947), HTML(value='')))
```

[136]:
```
[print(f"{k}, {v} ") for k,v in key_counts_dict.items() if ':city' in k]
```

```
addr:city, 5104
is_in:city, 1
addr:city_1, 2
```

[136]:
```
[None, None, None]
```

[137]:
```
#itemgetter used with sorted to allow sorting by key values
from operator import itemgetter
pprint(sorted(key_counts_dict.items(), key=itemgetter(1), reverse=True))
```

```
[('_id', 457947),
 ('type', 457947),
 ('id', 457947),
 ('version', 457947),
 ('timestamp', 457947),
 ('changeset', 457947),
 ('uid', 457947),
 ('user', 457947),
 ('lat', 405420),
 ('lon', 405420),
 ('highway', 32238),
 ('name', 17997),
 ('building', 14960),
 ('tiger:county', 12249),
 ('tiger:cfcc', 12242),
 ('tiger:name_base', 11459),
 ('tiger:name_type', 9816),
 ('tiger:zip_left', 8696),
 ('tiger:zip_right', 8478),
 ('tiger:reviewed', 8253),
 ('service', 6392),
 ('addr:street', 5587),
 ('addr:housenumber', 5238),
 ('addr:postcode', 5123),
 ('addr:city', 5104),
 ('addr:state', 4612),
 ('surface', 3766),
 ('amenity', 3739),
 ('oneway', 3347),
 ('power', 2814),
 ('source', 2228),
 ('access', 2052),
 ('addr:country', 1994),
 ('ref', 1887),
 ('footway', 1880),
 ('leisure', 1528),
 ('lanes', 1439),
 ('landuse', 1400),
 ('railway', 1242),
 ('tiger:source', 1225),
 ('tiger:upload_uuid', 1224),
 ('tiger:name_base_1', 1217),
 ('tiger:tlid', 1211),
 ('operator', 1021),
 ('hgv', 1005),
 ('parking', 995),
 ('name_1', 985),
 ('sport', 916),
('tiger:separated', 879),
('natural', 855),
('maxspeed', 831),
('bridge', 815),
('crossing', 807),
('building:levels', 801),
('foot', 800),
('layer', 789),
('bicycle', 727),
('golf', 684),
('barrier', 654),
('entrance', 640),
('hgv:state_network', 578),
('ele', 568),
('source:hgv:state_network', 555),
('shop', 533),
('wheelchair', 516),
('horse', 511),
('gnis:feature_id', 503),
('lit', 497),
('website', 481),
('fee', 435),
('park_ride', 419),
('waterway', 414),
('supervised', 406),
('gnis:created', 392),
('gnis:state_id', 384),
('gnis:county_id', 383),
('tiger:name_direction_suffix', 355),
('phone', 343),
('cuisine', 341),
('old_railway_operator', 341),
('sidewalk', 307),
('gauge', 290),
('motor_vehicle', 287),
('electrified', 277),
('description', 244),
('created_by', 224),
('tunnel', 219),
('tiger:name_direction_prefix', 218),
('historic', 204),
('old_ref', 203),
('water', 192),
('tourism', 191),
('tiger:name_type_1', 191),
('area', 186),
('covered', 168),
('hgv:national_network', 157),
('source:hgv:national_network', 157),
('width', 157),
('note', 152),
('man_made', 147),
('brand', 147),
('cycleway', 138),
('religion', 137),
('NHS', 136),
('memorial', 133),
('nat_ref', 129),
('opening_hours', 126),
('usage', 126),
('wikidata', 125),
('traffic_signals', 120),
('alt_name', 117),
('place', 111),
('destination:ref', 110),
('brand:wikidata', 106),
('aeroway', 104),
('traffic_signals:direction', 103),
('old_name', 102),
('tiger:zip_left_1', 102),
('office', 99),
('destination', 99),
('brand:wikipedia', 98),
('unsigned_ref', 98),
('rcn_ref', 98),
('gnis:county_name', 91),
('mown', 90),
('gnis:import_uuid', 85),
('gnis:reviewed', 82),
('is_in', 80),
('drive_through', 79),
('smoking', 78),
('gnis:Class', 76),
('gnis:County', 76),
('gnis:County_num', 76),
('gnis:ST_alpha', 76),
('gnis:ST_num', 76),
('gnis:id', 76),
('import_uuid', 76),
('denomination', 75),
('OBJECTID', 73),
('destination:street', 72),
('kerb', 71),
('wikipedia', 70),
('addr:unit', 68),
('junction:ref', 68),
('outdoor_seating', 67),
('healthcare', 67),
('capacity', 64),
('source:imagery', 63),
('boundary', 62),
('historic:amenity', 55),
('attraction', 55),
('internet_access', 54),
('takeaway', 54),
('atm', 53),
('admin_level', 53),
('delivery', 52),
('level', 50),
('tiger:name_base_2', 50),
('maxspeed:advisory', 49),
('shelter', 44),
('tracktype', 44),
('incline', 44),
('nysdot_ref', 43),
('trail_visibility', 43),
('network', 41),
('public_transport', 39),
('name_2', 39),
('bus', 38),
('bench', 37),
('par', 36),
('history', 35),
('healthcare:speciality', 34),
('payment:cash', 34),
('tiger:name_direction_prefix_1', 34),
('addr:housenumber_1', 34),
('addr:housename', 33),
('emergency', 33),
('junction', 33),
('tiger:zip_right_1', 32),
('building:part', 32),
('url', 31),
('basin', 31),
('gnis:edited', 30),
('shelter_type', 30),
('leaf_type', 30),
('internet_access:fee', 29),
('designation', 29),
('lanes:backward', 29),
('lanes:forward', 29),
('payment:credit_cards', 27),
('cycleway:right', 27),
('sac_scale', 26),
('passenger_information_display', 26),
('construction', 26),
('leaf_cycle', 26),
('tactile_paving', 25),
('information', 25),
('start_date', 25),
('cycleway:left', 25),
('height', 25),
('maxheight', 24),
('intermittent', 24),
('driveway', 24),
('noexit', 23),
('fence_type', 23),
('fineley_mown', 23),
('railway:historic', 22),
('payment:debit_cards', 21),
('embankment', 21),
('handrail', 21),
('dist:white', 21),
('direction', 20),
('boat', 20),
('segregated', 20),
('expressway', 19),
('noref', 18),
('contact:website', 18),
('deep_draft', 18),
('motorboat', 18),
('ship', 18),
('dist:black', 18),
('dist:blue', 18),
('dist:red', 18),
('handicap', 18),
('fixme', 17),
('artwork_type', 17),
('turn:lanes:forward', 17),
('tiger:zip_left_2', 17),
('source:deep_draft', 17),
('roof:shape', 17),
('social_facility:for', 16),
('owner', 16),
('social_facility', 16),
('fax', 15),
('dispensing', 15),
('email', 15),
('tiger:name_type_2', 15),
('motorcar', 15),
('roof:colour', 15),
('population', 14),
('craft', 14),
('wetap:status', 13),
('service:vehicle:car_repair', 13),
('turn:lanes:backward', 13),
('tower:type', 12),
('colour', 12),
('bicycle_parking', 12),
('service:vehicle:oil_change', 12),
('loc_ref', 12),
('census:population', 11),
('distance', 11),
('diet:vegan', 11),
('diet:vegetarian', 11),
('contact:phone', 11),
('name:en', 11),
('traffic_calming', 11),
('service:vehicle:tires', 11),
('destination:symbol', 11),
('capacity:disabled', 11),
('proposed', 11),
('stop', 10),
('service:vehicle:brakes', 10),
('service:vehicle:diagnostics', 10),
('service:vehicle:inspection', 10),
('name:railway', 10),
('border_type', 10),
('motorcycle', 10),
('psd', 10),
('FIXME', 9),
('artist_name', 9),
('kerb:left', 9),
('kerb:right', 9),
('service:vehicle:batteries', 9),
('gnis:feature_type', 8),
('fuel:octane_91', 8),
('government', 8),
('indoor', 8),
('payment:mastercard', 8),
('payment:visa', 8),
('building_1', 8),
('service:vehicle:body_repair', 8),
('cutting', 8),
('maxstay', 8),
('building:colour', 8),
('beauty', 7),
('second_hand', 7),
('ref:walmart', 7),
('service:vehicle:car_parts', 7),
('tiger:zip_right_2', 7),
('wetland', 7),
('lock', 7),
('comment', 7),
('crop', 7),
('roller_coaster', 7),
('room', 7),
('disused', 6),
('route_ref', 6),
('generator:source', 6),
('ramp:wheelchair', 6),
('door', 6),
('service:vehicle:air_conditioning', 6),
('service:vehicle:new_car_sales', 6),
('service:vehicle:used_car_sales', 6),
('tiger:zip_left_3', 6),
('ski', 6),
('snowmobile', 6),
('substation', 6),
('waterway_1', 6),
('wall', 6),
('ford', 5),
('attribution', 5),
('bridge:support', 5),
('fuel:octane_87', 5),
('payment:bitcoin', 5),
('amenity_1', 5),
('rooms', 5),
('payment:discover_card', 5),
('generator:method', 5),
('name:ar', 5),
('internet_access:ssid', 5),
('service:vehicle:alignment', 5),
('service:vehicle:electrical', 5),
('service:vehicle:glass', 5),
('cables', 5),
('frequency', 5),
('maxweight', 5),
('lock_name', 5),
('building:material', 5),
('payment:coins', 5),
('addr:housenumber_2', 5),
('addr:housenumber_3', 5),
('addr:housenumber_4', 5),
('addr:housenumber_5', 5),
('stormwater', 5),
('road', 5),
('smoothness', 5),
('wifi', 4),
('number', 4),
('fuel:octane_93', 4),
('playground', 4),
('stars', 4),
('fuel:octane_95', 4),
('fuel:diesel', 4),
('route', 4),
('payment:american_express', 4),
('service:vehicle:wheels', 4),
('tower:construction', 4),
('voltage', 4),
('destination:ref:to', 4),
('heritage', 4),
('heritage:operator', 4),
('ref:nrhp', 4),
('landuse_1', 4),
('mtb:scale:uphill', 4),
('leisure_1', 4),
('payment:amex', 4),
('payment:notes', 4),
('location', 4),
('service:vehicle:repairs', 4),
('addr:street_1', 4),
('dist:gold', 4),
('ref:right', 3),
('ref:left', 3),
('train', 3),
('material', 3),
('fuel:octane_89', 3),
('toilets:wheelchair', 3),
('fireplace', 3),
('building_2', 3),
('payment:cheque', 3),
('facebook', 3),
('service:vehicle:muffler', 3),
('studio', 3),
('inscription', 3),
('product', 3),
('destination:lanes', 3),
('destination:ref:lanes', 3),
('toll', 3),
('oneway:bicycle', 3),
('bridge:name', 3),
('tiger:name_direction_prefix_2', 3),
('FIXME:hgv:state_network', 3),
('extrude', 3),
('source:name', 3),
('capacity:parent', 3),
('capacity:women', 3),
('military', 3),
('mtb:scale', 3),
('courts', 3),
('service:vehicle:Transmission_Repair', 3),
('waterway:historic', 3),
('role', 3),
('landuse_2', 3),
('short_name', 3),
('turn:lanes', 3),
('green', 3),
('wildlife', 3),
('denotation', 3),
('surface:colour', 3),
('tomb', 3),
('service_times', 2),
('backrest', 2),
('distance_marker', 2),
('currency:USD', 2),
('diesel', 2),
('station', 2),
('board_type', 2),
('addr:floor', 2),
('official_name', 2),
('collection_times', 2),
('guidepost', 2),
('fire_hydrant:type', 2),
('addr:place', 2),
('repair', 2),
('year', 2),
('power_supply', 2),
('communication:mobile_phone', 2),
('communication:television', 2),
('source:noname', 2),
('bridge:movable', 2),
('seasonal', 2),
('tiger:zip_right_3', 2),
('FIXME:oneway', 2),
('elevation', 2),
('placement:forward', 2),
('step_count', 2),
('opening_date', 2),
('toilets:disposal', 2),
('unisex', 2),
('runnability', 2),
('generator:type', 2),
('hoops', 2),
('alt_name_1', 2),
('end_date', 2),
('plant:source', 2),
('industrial', 2),
('way', 2),
('addr:city_1', 2),
('addr:street_2', 2),
('produce', 2),
('maxspeed:backward', 2),
('maxspeed:forward', 2),
('monitoring:air_traffic', 2),
('payment:tap_to_pay', 2),
('floating', 2),
('sloped_curb', 1),
('is_in:city', 1),
('church', 1),
('disused:amenity', 1),
('disused:cuisine', 1),
('disused:name', 1),
('seats', 1),
('clock', 1),
('disused:shop', 1),
('drive_in', 1),
('contact:email', 1),
('is_in:continent', 1),
('name:ru', 1),
('name:uk', 1),
('state', 1),
('fuel:e10', 1),
('Routes', 1),
('automated', 1),
('self_service', 1),
('operator:wikidata', 1),
('operator:wikipedia', 1),
('addr:province', 1),
('addr:floot', 1),
('lawyer', 1),
('isced:field', 1),
('traffic_sign', 1),
('addr:pobox', 1),
('category', 1),
('name:es', 1),
('name:ht', 1),
('name:pl', 1),
('service:bicycle:repair', 1),
('service:bicycle:screwdriver', 1),
('service:bicycle:tools', 1),
('exit', 1),
('email_1', 1),
('payment:visa_debit', 1),
('brewery', 1),
('clothes', 1),
('brand_1', 1),
('brand_2', 1),
('brand_3', 1),
('wholesale', 1),
('cash_in', 1),
('animal_shelter', 1),
('community_centre:for', 1),
('contact:twitter', 1),
('lgbtq', 1),
('fire_hydrant:wrench', 1),
('water_source', 1),
('addr:full', 1),
('source:maxspeed', 1),
('access:conditional', 1),
('bridge_type', 1),
('noname', 1),
('tiger:zip_left_4', 1),
('tiger:zip_right_4', 1),
('tiger:name_direction_suffix_1', 1),
('FIXME:motorboat', 1),
('FIXME:ship', 1),
('is_in:country', 1),
('is_in:country_code', 1),
('is_in:iso_3166_2', 1),
('is_in:state', 1),
('is_in:state_code', 1),
('tiger:CLASSFP', 1),
('tiger:CPI', 1),
('tiger:FUNCSTAT', 1),
('tiger:LSAD', 1),
('tiger:MTFCC', 1),
('tiger:NAME', 1),
('tiger:NAMELSAD', 1),
('tiger:PCICBSA', 1),
('tiger:PCINECTA', 1),
('tiger:PLACEFP', 1),
('tiger:PLACENS', 1),
('tiger:PLCIDFP', 1),
('tiger:STATEFP', 1),
('bridge:historic', 1),
('gnis:county', 1),
('gnis:feature', 1),
('ref:Amtrak', 1),
('building:max_level', 1),
('building:min_level', 1),
('operator_1', 1),
('ramp', 1),
('park', 1),
('playground_1', 1),
('playground_2', 1),
('flickr', 1),
('golf:course', 1),
('mtb:scale:imba', 1),
('indistinct', 1),
('generator:output:electricity', 1),
('historic:civilization', 1),
('recycling_type', 1),
('iata', 1),
('icao', 1),
('mowed', 1),
('fineley_mowed', 1),
('dist:white:', 1),
('name:historic', 1),
('water_supply', 1),
('store_ref', 1),
('abandoned', 1),
('fuel:octane_92', 1),
('building:number', 1),
('payment:Monthly_Pass', 1),
('alt_name_2', 1),
('addr:street_3', 1),
('addr:housenumber_6', 1),
('addr:housenumber_7', 1),
('water_1', 1),
('check_date', 1),
('source:url', 1),
('plant:output:electricity', 1),
('payment:electronic_purses', 1),
('source:addr', 1),
('roof:levels', 1),
('historic:tunnel', 1),
('payment:apple_pay', 1),
('facility', 1),
('governance', 1),
('protect_class', 1),
('protection_object', 1),
('protection_title', 1),
('related_law', 1),
('site_ownership', 1),
('bridge:structure', 1),
('memorial_1', 1),
('memorial_2', 1),
('monument', 1),
('monument_1', 1),
('footway_1', 1),
('ref:store_number', 1),
('contact:facebook', 1),
('content', 1),
('note:lanes', 1),
('line', 1),
('faa', 1),
('rooftop', 1),
('beds', 1),
('building:height', 1),
('memorial:text', 1),
('urgent_care', 1),
('historic_1', 1),
('bulk_purchase', 1),
('organic', 1),
('psv', 1),
('min_height', 1),
('phases', 1),
('transformer', 1),
('voltage:primary', 1),
('managed', 1),
('seamark:harbour:category', 1),
('seamark:type', 1),
('abandoned:building', 1),
('fuel:gasoline', 1),
('fire_station:type', 1),
('centre_turn_lane', 1),
('tidal', 1)]
```

0.0.5 Get a list of fields that begin with address

[138]:
```
address_fields = {k:v for (k, v) in key_counts_dict.items() if 'addr' in k}
pprint(sorted(address_fields.items(), key=itemgetter(1), reverse=True))
```

```
[('addr:street', 5587),
 ('addr:housenumber', 5238),
 ('addr:postcode', 5123),
 ('addr:city', 5104),
 ('addr:state', 4612),
 ('addr:country', 1994),
 ('addr:unit', 68),
 ('addr:housenumber_1', 34),
 ('addr:housename', 33),
 ('addr:housenumber_2', 5),
 ('addr:housenumber_3', 5),
 ('addr:housenumber_4', 5),
 ('addr:housenumber_5', 5),
 ('addr:street_1', 4),
 ('addr:floor', 2),
 ('addr:place', 2),
 ('addr:city_1', 2),
 ('addr:street_2', 2),
 ('addr:province', 1),
 ('addr:floot', 1),
 ('addr:pobox', 1),
 ('addr:full', 1),
 ('addr:street_3', 1),
 ('addr:housenumber_6', 1),
 ('addr:housenumber_7', 1),
 ('source:addr', 1)]
```

[139]:
```
# Get a list of distinct streets
distinct_streets = osm_col.distinct('addr:street')
```

[140]:
```
change_needed = list()

mapping_dict = get_suffix_mapping()
distinct_suffix = set(x.split()[-1] for x in distinct_streets)
```

[164]:
```
pprint(mapping_dict[0:5])
```

```
[{'Common': 'ALLEE', 'Standard': 'ALY'},
 {'Common': 'ALLEY', 'Standard': 'ALY'},
 {'Common': 'ALLY', 'Standard': 'ALY'},
 {'Common': 'ALY', 'Standard': 'ALY'},
 {'Common': 'ANEX', 'Standard': 'ANX'}]
```

[142]:
```
for suffix in distinct_suffix:
    for mapping in mapping_dict:
        if mapping['Common'] == suffix.upper():
            print(f"Changing {suffix} to {mapping['Standard']}")
            continue
```

```
Changing Way to WAY
Changing Passage to PSGE
Changing Bend to BND
Changing Cir to CIR
Changing Avenu to AVE
Changing Dr to DR
Changing Green to GRN
Changing Square to SQ
Changing Court to CT
Changing Manor to MNR
Changing Meadows to MDWS
Changing Hill to HL
Changing Blvd to BLVD
Changing Ct to CT
Changing Parkway to PKWY
Changing Landing to LNDG
Changing Run to RUN
Changing Crescent to CRES
Changing Road to RD
Changing Highway to HWY
Changing Street to ST
Changing ave to AVE
Changing Avenue to AVE
Changing St to ST
Changing Rd to RD
Changing Circle to CIR
Changing Drive to DR
Changing Lane to LN
Changing Ave to AVE
Changing Trail to TRL
Changing Boulevard to BLVD
Changing Center to CTR
Changing Park to PARK
Changing Bridge to BRG
```

[143]:
```python
modified_count = 0
for entry in tqdm_notebook(distinct_streets):
    suffix = entry.split()[-1]
    for mapping in mapping_dict:
        if mapping['Common'] == suffix.upper():
            #print(f"Changing {suffix} to {mapping['Standard']}")
            #print(entry.replace(suffix, mapping['Standard']))
            result = osm_col.update_many({'addr:street': entry}, {"$set":␣
 ↪{'addr:street': entry.replace(suffix, mapping['Standard'])}})
            modified_count += result.modified_count
            continue
print(f"{modified_count} address suffixes updated")
```

```
HBox(children=(IntProgress(value=0, max=548), HTML(value='')))



4976 address suffixes updated
```

[144]:
```python
# Get a list of distinct street types
pprint(set(x.split()[-1] for x in distinct_streets if x.split()[-1].isalpha()))
```

```
{'Apartment',
 'Ave',
 'Avenu',
 'Avenue',
 'Bend',
 'Blvd',
 'Boulelvard',
 'Boulevard',
 'Bridge',
 'Center',
 'Cir',
 'Circle',
 'Court',
 'Crescent',
 'Ct',
 'Dr',
 'Drive',
 'Drop',
 'East',
 'Green',
 'Highway',
 'Hill',
 'Homes',
 'Landing',
 'Lane',
 'Manor',
 'Market',
 'Meadows',
 'N',
 'North',
 'Oaks',
 'PW',
 'Park',
 'Parkway',
 'Passage',
 'Place',
 'Race',
 'Rd',
 'Rise',
 'Road',
 'Run',
 'S',
 'South',
 'Spruce',
 'Square',
 'St',
 'Stree',
 'Street',
 'Trail',
 'Villas',
 'W',
 'Way',
 'West',
 'Woods',
 'ave',
 'line'}
```

[145]:
```
# Get a list of distinct street types
pprint(set(x.split()[-1] for x in distinct_streets))
```

```
{'#102',
 '#2',
 '#A-2',
 '31',
 '92',
 'Apartment',
 'Ave',
 'Ave.',
 'Avenu',
 'Avenue',
 'Bend',
 'Blvd',
 'Boulelvard',
 'Boulevard',
 'Bridge',
 'Center',
 'Cir',
 'Circle',
 'Court',
 'Crescent',
 'Ct',
 'Dr',
 'Drive',
 'Drop',
 'East',
 'Green',
 'Highway',
 'Hill',
 'Homes',
 'Landing',
 'Lane',
 'Manor',
 'Market',
 'Meadows',
 'N',
 'North',
 'Oaks',
 'PW',
 'Park',
 'Parkway',
 'Passage',
 'Place',
 'Race',
 'Rd',
 'Rd.',
 'Rise',
 'Road',
 'Run',
 'S',
 'South',
 'Spruce',
 'Square',
 'St',
 'St.',
 'Stree',
 'Street',
 'Trail',
 'Villas',
 'W',
 'Way',
 'West',
 'Woods',
 'ave',
 'line'}
```

0.0.6 find all address codes in collection

[146]:
```
unique_zip_codes = osm_col.distinct('addr:postcode')
pprint(unique_zip_codes)
```

```
['14607',
 '14624',
 '14617',
 '14623',
 '14622',
 '14612',
 '14626',
 '14450',
 '14618',
 '14616',
 '14526',
 '14502',
 '14514',
 '14615',
 '14580',
 '14620',
 '14625',
 '14445',
 '14608',
 '14609',
 '14606',
 '14559',
 '14621',
 '14613',
 '14534',
 '14604',
 '14614',
 '14620-1327',
 'West Main Street',
 '14694',
 '14605',
 '14610',
 '14611',
 '14468',
 '14607-2082',
 '14519',
 '14642',
 '14627',
 '14624-4721',
 '14617-1822',
 '14467',
 '14692',
 '14568',
 '14543',
 '14586',
 '14428',
 '1445033',
 '14424',
 '14619']
```

[147]:
```
update_dict = {'modified': 0,
               'deleted': 0,
               'good': 0}
for zip in tqdm_notebook(unique_zip_codes):
    if zip[0:5].isdigit() and len(zip) > 5:
        result = osm_col.update_many({'addr:postcode': zip}, {"$set": {'addr:
 ↪postcode': zip[0:5]}})
        update_dict['modified'] += result.modified_count
    elif not zip.isdigit() and len(zip)!=5:
        result = osm_col.delete_many({'addr:postcode': zip})
        update_dict['deleted'] += result.deleted_count
    elif zip.isdigit() and len(zip)==5:
        update_dict['good'] += 1

pprint(update_dict)
```

```
HBox(children=(IntProgress(value=0, max=49), HTML(value='')))
```

```
{'deleted': 1, 'good': 43, 'modified': 6}
```

[148]:
```
#
updated_address_code_list = list(osm_col.find({'addr:postcode': {'$exists':␣
 ↪True}}, {'addr:postcode': 1, '_id': 0}))
set([x['addr:postcode'] for x in updated_address_code_list])
```

[148]:
```
{'14424',
 '14428',
 '14445',
 '14450',
 '14467',
 '14468',
 '14502',
 '14514',
 '14519',
 '14526',
 '14534',
 '14543',
 '14559',
 '14568',
 '14580',
 '14586',
 '14604',
 '14605',
 '14606',
 '14607',
 '14608',
 '14609',
 '14610',
 '14611',
 '14612',
 '14613',
 '14614',
 '14615',
 '14616',
 '14617',
 '14618',
 '14619',
 '14620',
 '14621',
 '14622',
 '14623',
 '14624',
 '14625',
 '14626',
 '14627',
 '14642',
 '14692',
 '14694'}
```

Rochester Zip codes > After running our function we can see that all the unique zip codes in the database are valid Rochester Zip codes

[149]:
```
pprint(list(osm_col.aggregate([
    {
        '$group': {
            '_id': '$addr:postcode',
            'count': {
                '$sum': 1
            }
        }
    }, {
        '$sort': {
            'count': -1
        }
    }
]))[:10])
```

```
[{'_id': None, 'count': 452824},
 {'_id': '14450', 'count': 1624},
 {'_id': '14624', 'count': 445},
 {'_id': '14618', 'count': 421},
 {'_id': '14623', 'count': 392},
 {'_id': '14534', 'count': 346},
 {'_id': '14626', 'count': 300},
 {'_id': '14514', 'count': 264},
 {'_id': '14612', 'count': 226},
 {'_id': '14620', 'count': 177}]
```

1 User Counts

[150]:
```
def get_single_users(col: Collection):
    user_counts_dict = list(col.aggregate([
        {
            '$sortByCount': '$user'
        }, {
            '$sort': {
                'count': 1
            }
        }
    ]))
    single_doc_user = list()
    for entry in user_counts_dict:
        if entry['count'] == 1:
           single_doc_user.append(entry['_id'])
        else:
            break
    pprint(single_doc_user[0:5])
    pprint(f"{len(single_doc_user)} users with only one post out of␣
 ↪{len(user_counts_dict)}")
    # return single_doc_user
get_single_users(osm_col)
```

```
['dgitto', 'Takuto', 'lonvia', 'glglgl', 'ayazhaider9']
'146 users with only one post out of 719'
```

[151]:
```
user_df = pd.DataFrame.from_dict(list(osm_col.aggregate([ {
            '$sortByCount': '$user'
        }])))
user_df['percent']= user_df['count']/user_df['count'].sum()
```

[152]:
```
# Percent of entries that came from top two users
user_df[0:2]['percent'].sum()*100
```

[152]:
```
23.01821612155145
```

[153]:
```
# Combined top 10 users contribution
user_df[0:10]['percent'].sum()*100
```

[153]:
```
61.08449467841187
```

[154]:
```
# Combined perecent of users who individually contribute less then 1% of the␣
 ↪entries in the database
user_df[user_df.percent <= .01].percent.sum()*100
```

[154]:
```
24.901407589541126
```

[155]:
```
user_df._id
```

[155]:
```
0        woodpeck_fixbot
1                 wambag
2               dankpoet
3               idrive66
4                 ECRock
5                 sivart
6             RussNelson
```

```
7                   timr
8                  T-Rex
9        visionsofkenobi
10      URcommunications
11                 stuuf
12        Craig Williams
13             paperboat
14           devrintalen
15                gadget
16                  fx99
17                ColumM
18              bot-mode
19        Nathan Willard
20                 Hooka
21                McColl
22                canisd
23                deejoe
24                Hwyfan
25             jwernerny
26              kbzimmer
27            slugmuffin
28             njtbusfan
29               jsb2092
             ...
689                hakan
690            Mickael S
691             dmouhama
692             Bhojaraj
693              Anthony
694                 SK53
695        JustinColeGIS
696    Marcussacapuces91
697          Wendy Marks
698             dpstreet
699              erikjos
700    steven mccandlish
701     Joshua_Heiberger
702        adam1aldridge
703          Jochen Topf
704                  EoE
705            Bootprint
706              Hparekh
707     Thomas Warmerdam
708              Jacob T
709              Rockear
710     Jerome Bernardes
711              baloona
```

```
712     royalphotography
713                GerdP
714      ComradeCosmobot
715          musclemaint
716                 JAG2
717        hawverdisplay
718               Tma339
Name: _id, Length: 719, dtype: object
```

[156]:
```
def top_ten_amenities(col: Collection):
    top_amenities= list(col.aggregate([
    {
        '$match': {
            'type': 'way'
        }
    }, {
        '$sortByCount': '$amenity'
    }
]))
    return top_amenities
top_ten_amenities(col=osm_col)[0:10]
```

[156]:
```
[{'_id': None, 'count': 49953},
 {'_id': 'parking', 'count': 1828},
 {'_id': 'restaurant', 'count': 127},
 {'_id': 'school', 'count': 85},
 {'_id': 'fuel', 'count': 56},
 {'_id': 'fast_food', 'count': 54},
 {'_id': 'place_of_worship', 'count': 49},
 {'_id': 'bank', 'count': 46},
 {'_id': 'shelter', 'count': 43},
 {'_id': 'fire_station', 'count': 32}]
```

[157]:
```
df = pd.DataFrame.from_dict(top_ten_amenities(osm_col))
pprint(df[0:10])
```

```
                _id  count
0              None  49953
1           parking   1828
2        restaurant    127
3            school     85
4              fuel     56
5         fast_food     54
6  place_of_worship     49
7              bank     46
8           shelter     43
9      fire_station     32
```

[158]:
```
df['percent'] = df['count']/df['count'].sum()
```

```
_id|count|percent\r\n|49953|0.951069055461417\r\nparking|1828|0.034803800239894905\r\nrestaura
```

[159]:
```
df[1:].describe()
```

[159]:
```
             count    percent
count    48.000000  48.000000
mean     53.541667   0.001019
std     262.708232   0.005002
min       1.000000   0.000019
25%       3.000000   0.000057
50%       6.500000   0.000124
75%      14.250000   0.000271
max    1828.000000   0.034804
```

[160]:
```
df.shape[0]
```

[160]:
```
49
```

[161]:
```
# Biggest Religion
religion = list(osm_col.aggregate([
    {
        '$match': {
            'amenity': {
                '$eq': 'place_of_worship'
            }
        }
    }, {
        '$group': {
            '_id': '$religion',
            'count': {
                '$sum': 1
            }
        }
    }, {
        '$sort': {
            'count': -1
        }
    }
]))
pprint(religion)
```

```
[{'_id': 'christian', 'count': 119},
 {'_id': None, 'count': 19},
 {'_id': 'jewish', 'count': 2},
 {'_id': 'buddhist', 'count': 1},
 {'_id': 'muslim', 'count': 1}]
```

[162]:
```
# Most popular cuisine in restaurants
cuisine = list(osm_col.aggregate([
    {
        '$match': {
            'amenity': {
                '$eq': 'restaurant'
            }
        }
    }, {
        '$group': {
            '_id': '$cuisine',
            'count': {
                '$sum': 1
            }
        }
    }, {
        '$sort': {
            'count': -1
        }
    }
]))
pprint(cuisine[0:10])
```

```
[{'_id': None, 'count': 119},
 {'_id': 'pizza', 'count': 31},
 {'_id': 'american', 'count': 25},
 {'_id': 'italian', 'count': 15},
 {'_id': 'burger', 'count': 14},
 {'_id': 'sandwich', 'count': 8},
 {'_id': 'mexican', 'count': 8},
 {'_id': 'chinese', 'count': 7},
 {'_id': 'japanese', 'count': 5},
 {'_id': 'thai', 'count': 4}]
```

[163]:
```
#City counts
city_counts = list(osm_col.aggregate([
    {
        '$group': {
            '_id': '$addr:city',
            'count': {
                '$sum': 1
            }
        }
    }, {
        '$sort': {
            'count': -1
        }
    }
]))
print('\n'.join('{_id!s:<20}{count}'.format(**x) for x in city_counts))
```

```
None                452842
Rochester           2227
Fairport            1612
Pittsford           350
North Chili         264
Brighton            205
Greece              161
Webster             86
Henrietta           35
Churchville         29
West Henrietta      18
East Rochester      17
Rochester, NY       16
Penfield            16
Perinton            15
Hilton              15
Spencerport         11
North Greece        5
Macedon             5
Riga                2
rochester           2
Ontario, NY         2
Walworth            2
Gates               2
Irondequoit         2
Ontario             1
W Commercial St     1
Rochestet           1
pittsford           1
East Rochester Town 1
```

Command	Description
`git status`	List all *new or modified* files
`git diff`	Show file differences that **haven't been** staged

[ ]:

```

```