

```
In [ ]: import plotly.express as px
import pandas as pd
import plotly.graph_objects as go
import json
import sqlite3
from shapely.geometry import Point
from shapely.geometry.polygon import Polygon
import plotly.io as pio
pio.renderers.default='notebook'
```

```
In [ ]: # Connecting to SQLite database and pulling aggregated speed and red light camera
data
connection = sqlite3.connect("camera_violations.db")
c = connection.cursor()

s = ''' select ADDRESS, CAMERAID, sum(VIOLATIONS) AS 'TOTAL_VIOLATIONS', LATITUD
E, LONGITUDE from speed group by CAMERAID '''
c.execute(s)
table = c.fetchall()

df = pd.DataFrame(table)
df = df.rename(columns = {0:'address',1:'camera_id',2:'violations',3:'lat',4:'lon'})

s2 = ''' SELECT INTERSECTION, CAMERAID, ADDRESS, SUM(VIOLATIONS) as 'TOTAL_VIOLAT
IONS', LATITUDE, LONGITUDE FROM red_light group by ADDRESS '''
c.execute(s2)
table2 = c.fetchall()

df2 = pd.DataFrame(table2)
df2 = df2.rename(columns = {0:'intersection',1:'camera_id', 2:'address',3:'violat
ions',4:'lat',5:'lon'})
```

```
In [ ]: # Loading Census Tract and Race data
f = open("chi_census.geojson")
tracts = json.load(f)
race = pd.read_csv("race_by_tract.csv")
```

```
In [ ]: # Setting reference size for scatter bubbles
sizeref_s = 2. * df['violations'].max() / (8 ** 2)
sizeref_r = 2. * df2['violations'].max() / (8 ** 2)
```

```

In [ ]: # Creating Choropleth map
fig = go.Figure(go.Choroplethmapbox(geojson=tracts, locations=race['Tract_name'],
                                     featureidkey='properties.namesad10', \
                                     z= race['Non-White'], \
                                     colorscale="Inferno_r", \
                                     text=race['Non-White'],
                                     zmin=0.0, zmax=1.2, #text=zhvi_county_inc_pop["te
xt_2yrs"],
                                     marker_opacity=0.5, marker_line_width=0))
fig.update_layout(mapbox_style="carto-positron",
                  mapbox_zoom=10, mapbox_center = {"lat": 41.9, "lon": -87.6})

fig.add_trace(go.Scattermapbox(
    lon = df['lon'],
    lat = df['lat'],
    mode = 'markers',
    marker = dict(
        size = df['violations']/sizeref_s,
        color = 'blue'
    )))
fig.add_trace(go.Scattermapbox(
    lon = df2['lon'],
    lat = df2['lat'],
    mode = 'markers',
    marker = dict(
        size = df2['violations']/sizeref_r,
        color = 'red'
    )))

fig.show()

```



```
In [ ]: # Functions to return numbers of speed/red light cameras in a given polygon
def num_speed_cams_in_poly(pol_name):
    s_cams = 0
    for p in speed_points:
        if pol_name in polsd and polsd[pol_name].contains(p):
            s_cams += 1
    return s_cams
def num_rl_cams_in_poly(pol_name):
    r_cams = 0
    for p in rl_points:
        if pol_name in polsd and polsd[pol_name].contains(p):
            r_cams += 1
    return r_cams
```

```
In [ ]: # Adding number of cameras to the race dataframe, as this is already broken out by census tract
race['num_speed_cams'] = race.apply(lambda row: num_speed_cams_in_poly(row['Tract_name']), axis = 1)
race['num_rl_cams'] = race.apply(lambda row: num_rl_cams_in_poly(row['Tract_name']), axis = 1)
race['all_cams'] = race['num_speed_cams'] + race['num_rl_cams']
race = race.replace("#DIV/0!", '0')
race['Non-White'] = race['Non-White'].astype(float)
```