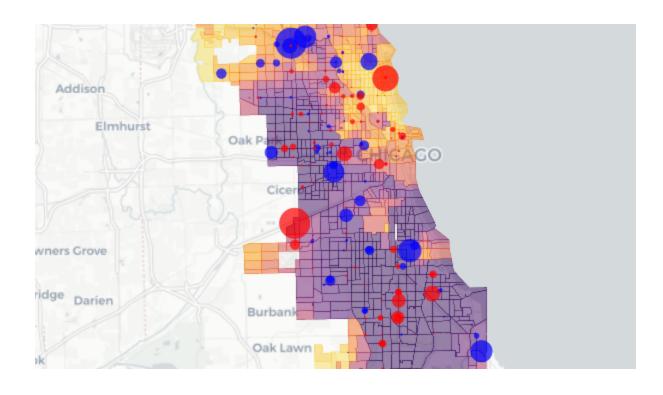
```
In [ ]: import plotly.express as px
        import pandas as pd
        import plotly.graph objects as go
        import json
        import sqlite3
        from shapely.geometry import Point
        from shapely.geometry.polygon import Polygon
        import plotly.io as pio
        pio.renderers.default='notebook'
In [ ]: # Connecting to SQLite database and pulling aggregated speed and red light camera
        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:'lo
        n'})
        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.namelsad10', \
                                         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 [ ]: # Creating a list of polygon objects for all census tracts
        polsd = \{\}
        for x in tracts['features']:
            pol = Polygon(x['geometry']['coordinates'][0][0])
            polsd[x['properties']['namelsad10']] = pol
In [ ]: # Creating lists of point objects for all speed and red light cameras
        speed_points = []
        for r in df.itertuples():
            if r[4] and r[5]:
                loc = (float(r[5]), float(r[4]))
                speed_points.append(Point(loc))
        rl points = []
        for r in df2.itertuples():
            if r[5] and r[6]:
                loc = (float(r[6]), float(r[5]))
                rl_points.append(Point(loc))
```

```
In [ ]: # Adding number of cameras to the race dataframe, as this is already broken out b
    y 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)
```