In [1]:
```
import pandas as pd
import numpy as np
import altair as alt
import plotly.express as px
alt.data_transformers.disable_max_rows()
import warnings
warnings.filterwarnings('ignore')
```

Data Loading

In [2]:
```
crime_df = pd.read_csv('Crime_Data_from_2020_to_Present.csv')
crime_df.shape
```

Out[2]:
```
(852950, 27)
```

In [3]:
```
crime_df.head(1)
```

Out[3]:

	division_number	date_reported	date_occurred	area	area_name	reporting_district	part	crime_code	crime_description	modus_operandi	.
0	10304468	2020-01-08	2020-01-08 22:30:00	3	Southwest	377	2	624	BATTERY - SIMPLE ASSAULT	0444 0913	

1 rows × 27 columns

In [4]:
```
crime_df.columns
```

Out[4]:
```
Index(['division_number', 'date_reported', 'date_occurred', 'area',
       'area_name', 'reporting_district', 'part', 'crime_code',
       'crime_description', 'modus_operandi', 'victim_age', 'victim_sex',
       'victim_descent', 'premise_code', 'premise_description', 'weapon_code',
       'weapon_description', 'status', 'status_description', 'crime_code_1',
       'crime_code_2', 'crime_code_3', 'crime_code_4', 'location',
       'cross_street', 'latitude', 'longitude'],
      dtype='object')
```

Goal is to count the number of incidents by zipcode in los angeles and display that in a choropleth

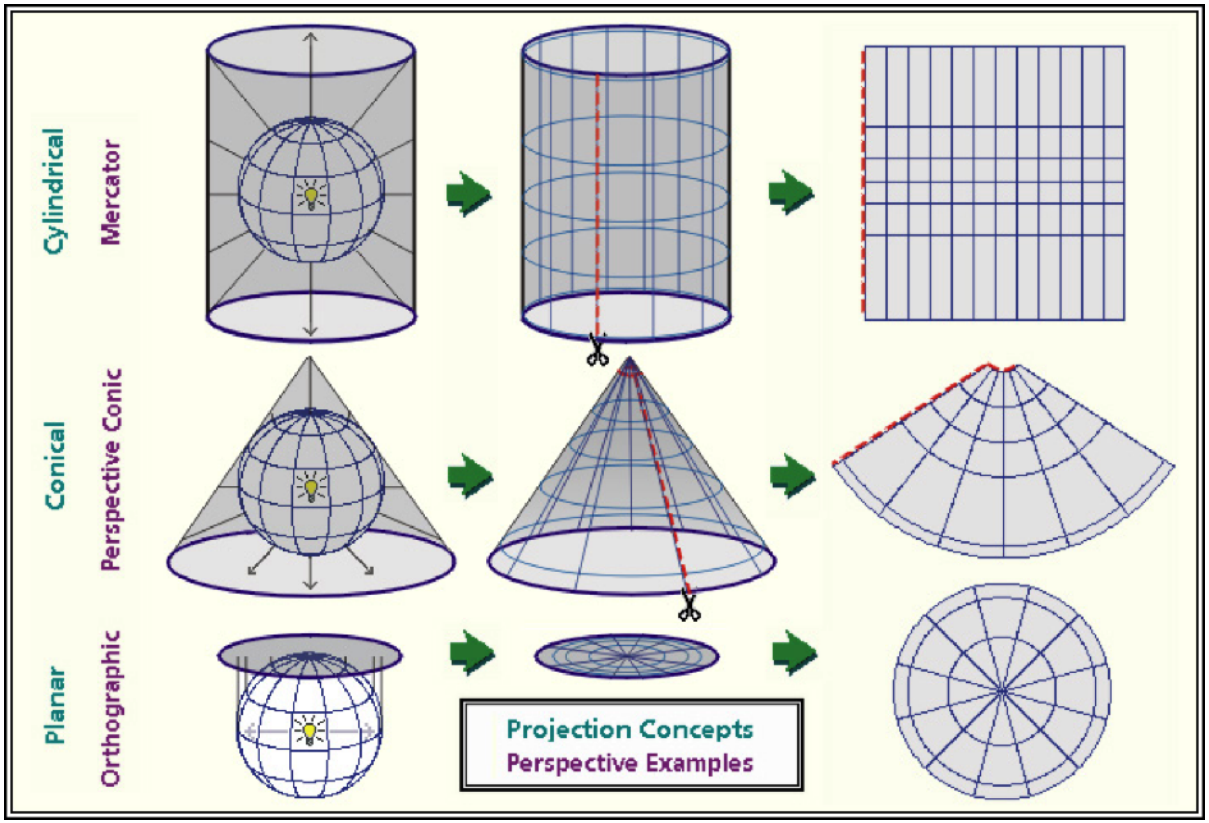
Gameplan:

1) load a geojson of LA zipcodes. 2) Use Geopandas to create a point from the latitude and longitude data 3) Use geopandas to do a geospatial join with the shapefile to find points within each polygon.

In [5]:
```
from datetime import datetime
import json
import geopandas as gpd
from shapely.geometry import Point
```

In [6]:
```
from IPython.display import Image
Image(filename='projections.png',width = 600, height = 400)
```

Out[6]:



In [7]:
```
#step 1: loading geojson
zip_shapes = gpd.read_file('geo/Zip_Codes_(LA_County).geojson')
zip_shapes

#transitioning between lat/long coordinates to 2D cartesian for los angeles area
zip_shapes = zip_shapes.to_crs("EPSG:32611")
```

```
zip_shapes['zipcode_area'] = zip_shapes.geometry.area
zip_shapes = zip_shapes.to_crs("EPSG:4326")
```

In [8]:
```
zip_shapes.head()
```

Out[8]:

	OBJECTID	ZIPCODE	ZIP	TOOLTIP	NLA_URL	geometry	zipcode_area
0	1	90001	90001	Zip Code: 90001	navigatela/reports/zipcode_county_report.cfm?p...	POLYGON ((-118.24338 33.98924, -118.24348 33.9...	8.873743e+06
1	2	90002	90002	Zip Code: 90002	navigatela/reports/zipcode_county_report.cfm?p...	POLYGON ((-118.23431 33.96101, -118.23442 33.9...	7.684036e+06
2	3	90003	90003	Zip Code: 90003	navigatela/reports/zipcode_county_report.cfm?p...	POLYGON ((-118.28285 33.97597, -118.28285 33.9...	9.531227e+06
3	4	90004	90004	Zip Code: 90004	navigatela/reports/zipcode_county_report.cfm?p...	POLYGON ((-118.28410 34.08349, -118.28438 34.0...	7.796605e+06
4	5	90005	90005	Zip Code: 90005	navigatela/reports/zipcode_county_report.cfm?p...	MULTIPOLYGON (((-118.33652 34.06190, -118.3367...	3.426463e+06

In [9]:
```
#step 2 creating points from lat long data
```

In [10]:
```
#using geopandas and lat/long points to create a point
pointsdf = gpd.GeoDataFrame(
    crime_df, geometry=gpd.points_from_xy(crime_df['longitude'], crime_df['latitude'])).set_crs(epsg=4326, inplace=True


pointsdf.head(2)
```

Out[10]:

	division_number	date_reported	date_occurred	area	area_name	reporting_district	part	crime_code	crime_description	modus_operandi	.
0	10304468	2020-01-08	2020-01-08 22:30:00	3	Southwest	377	2	624	BATTERY - SIMPLE ASSAULT	0444 0913	
1	190101086	2020-01-02	2020-01-01 03:30:00	1	Central	163	2	624	BATTERY - SIMPLE ASSAULT	0416 1822 1414	

2 rows × 28 columns

In [11]:
```
#spatial join on areas
zip_areas = pointsdf.sjoin(zip_shapes ,how = 'left',predicate="within").drop_duplicates('ZIPCODE')[['ZIPCODE','zipcode_
zip_areas.head(1)
```

Out[11]:

	ZIPCODE	zipcode_area
0	90037	7.561823e+06

In [12]:
```
# step 3 spatial join with points


crime_joined = pointsdf.sjoin(zip_shapes ,how = 'left',predicate="within").groupby(['ZIPCODE']).size().to_frame().reset
crime_joined.rename(columns = {0: 'crimes'}, inplace = True)


crime_joined = crime_joined.merge(zip_areas, on = 'ZIPCODE')
crime_joined['crimes/m^2'] = crime_joined['crimes'] /crime_joined['zipcode_area']
crime_joined['crimes/sq mi'] = crime_joined['crimes/m^2'] / 3.86102e-7
crime_joined['log_crimes/sq mi'] = np.log(crime_joined['crimes'] / (crime_joined['zipcode_area'] * 3.86102e-7))


crime_joined.head()
```

Out[12]:

	ZIPCODE	crimes	zipcode_area	crimes/m^2	crimes/sq mi	log_crimes/sq mi
0	90001	3147	8.873743e+06	0.000355	918.518364	6.822762
1	90002	8900	7.684036e+06	0.001158	2999.843332	8.006315
2	90003	25861	9.531227e+06	0.002713	7027.396434	8.857572
3	90004	11397	7.796605e+06	0.001462	3786.020358	8.239071
4	90005	7593	3.426463e+06	0.002216	5739.383609	8.655107

Drawing a preliminary choropleth.

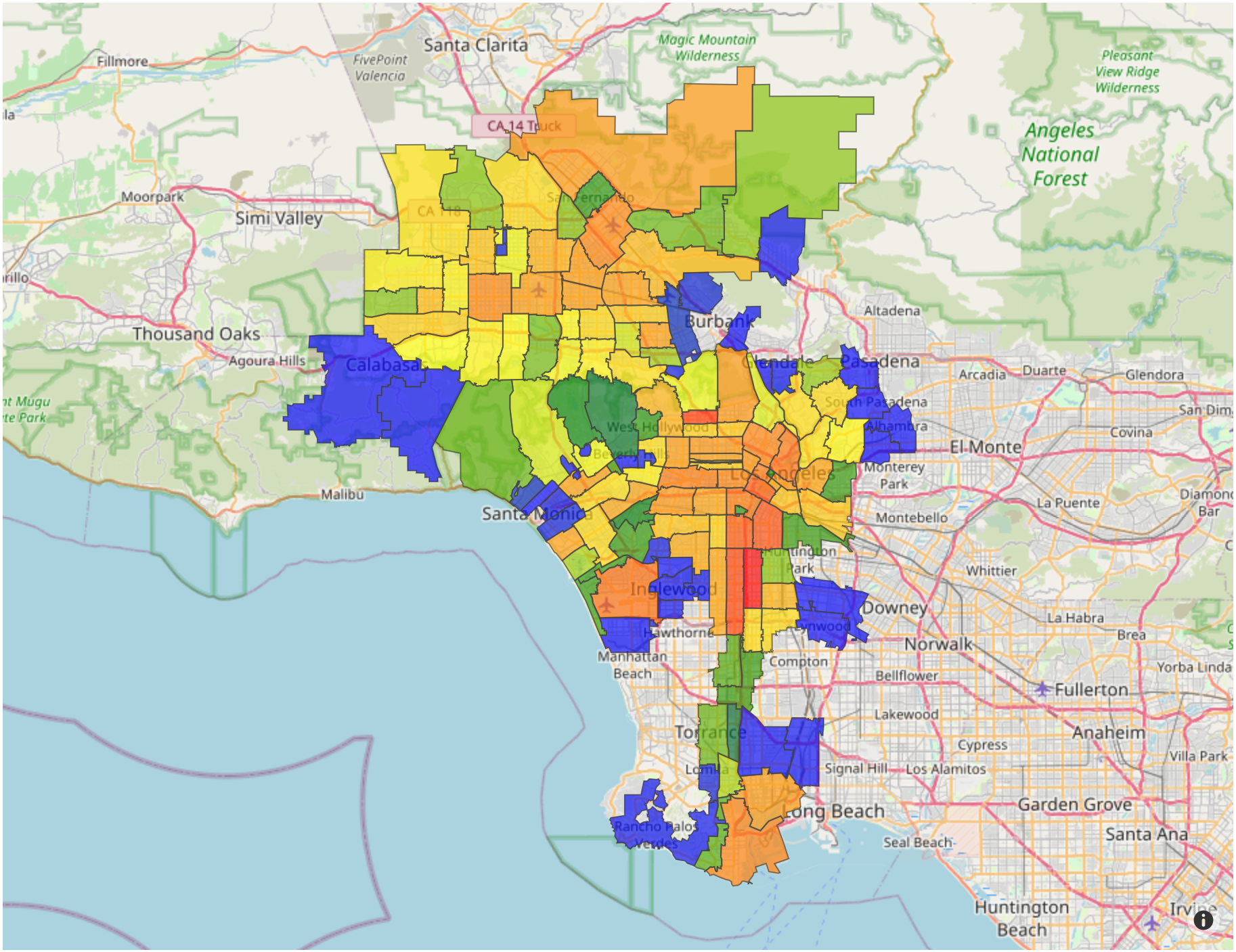
Thoughts: At somepoint ill need to normalize values e.g. (per capita, per square mile) but lets start with a `pure` choropleth with just crime counts in each zipcode

In [13]:
```
#loading in geojson file that is used plotly.express choropleth_mapbox func

with open('geo/Zip_Codes_(LA_County).geojson') as user_file:
    file_contents = user_file.read()

parsed_json = json.loads(file_contents)
```

Choropleth with pure crime counts

In [14]:

```
q1 = crime_joined['crimes'].quantile(0.25)
q2 = crime_joined['crimes'].quantile(0.50)  # median
q3 = crime_joined['crimes'].quantile(0.75)
q4 = crime_joined['crimes'].max()


# The scale will have distinct colors at each quartile
quartile_scale = [
    [0, 'blue'],                 # Start with blue
    [q1/q4, 'green'],            # Green at first quartile
    [q2/q4, 'yellow'],           # Yellow at median
    [q3/q4, 'orange'],           # Orange at third quartile
    [1.0, 'red']                 # Red at maximum
]

pure_fig = px.choropleth_mapbox(crime_joined, geojson= parsed_json, featureidkey ="properties.ZIPCODE",
                            locations= crime_joined.ZIPCODE ,color = 'crimes', color_continuous_scale= quartile_scale,
                            mapbox_style='open-street-map',
                            zoom=9,
                            opacity = 0.65,
                            width = 1000, height = 700,
                            center = {"lat":  34.0500, "lon": -118.4105}

                           )
pure_fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})



pure_fig.show()
```



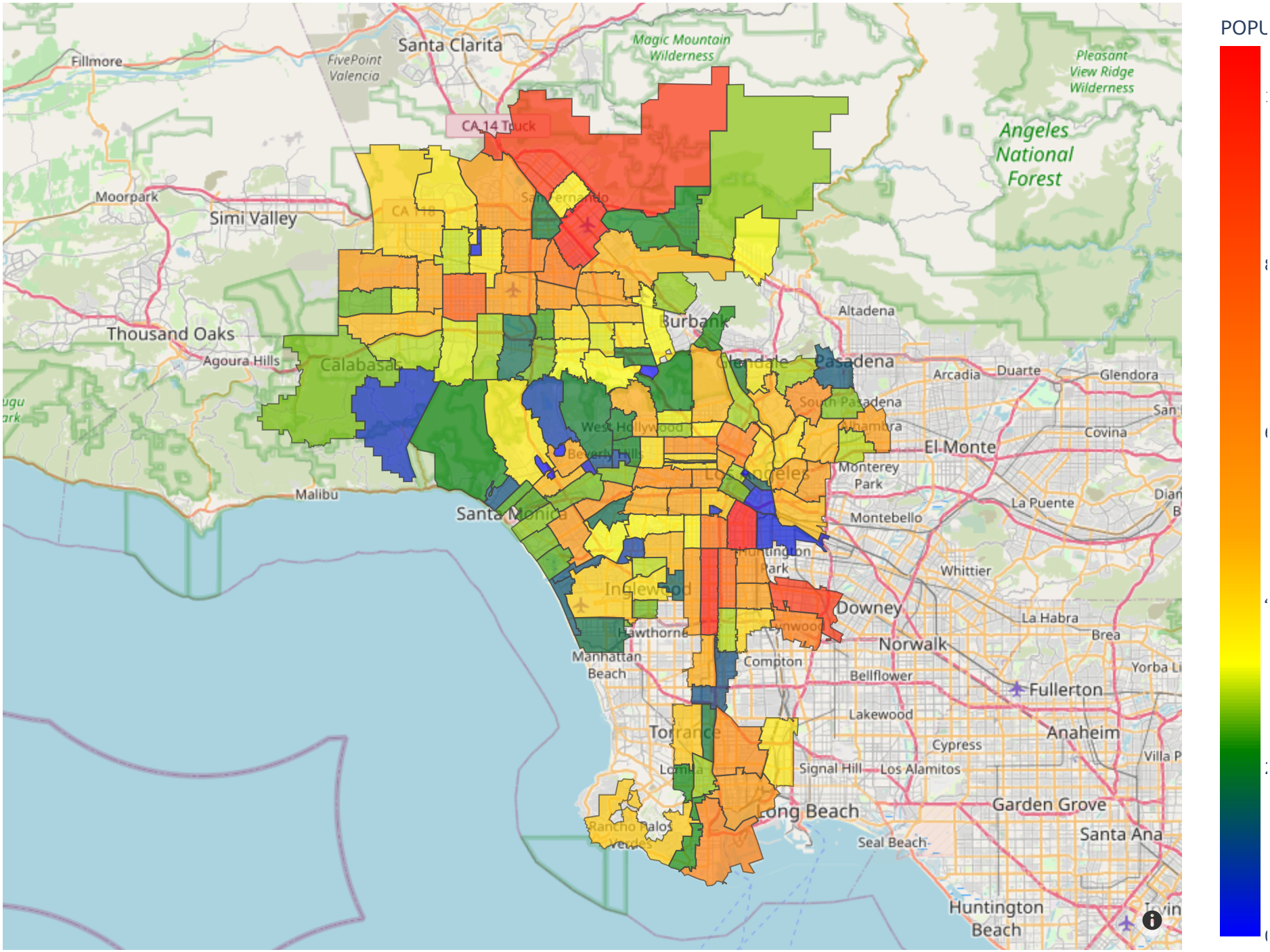
I have a suspicion that the plot above is essentially a population plot instead of revealing information related to crime and area. I will validate that by plotting an actual `population choropleth` in each zipcode

In [15]:
```
#reading in census2022 population data by zipcode
zip_pop = pd.read_excel('ZIP_POP.xlsx')
zip_pop = zip_pop.astype(str)

#merging with original crime by zipcode df
crime_joined = crime_joined.merge(zip_pop, on = 'ZIPCODE')
```

In [16]:
```
#fixing the typing of the population column
crime_joined['POPULATION'] = crime_joined['POPULATION'].astype(int)

crime_joined.iloc[63,6] = 1
```

Choropleth with Population by zipcode

In [17]:
```
q1 = crime_joined['POPULATION'].quantile(0.25)
q2 = crime_joined['POPULATION'].quantile(0.50)  # median
q3 = crime_joined['POPULATION'].quantile(0.75)
q4 = crime_joined['POPULATION'].max()


# The scale will have distinct colors at each quartile
quartile_scale = [
    [0, 'blue'],                  # Start with blue
    [q1/q4, 'green'],             # Green at first quartile
    [q2/q4, 'yellow'],            # Yellow at median
    [q3/q4, 'orange'],            # Orange at third quartile
    [1.0, 'red']                  # Red at maximum
]


pop_fig = px.choropleth_mapbox(crime_joined, geojson= parsed_json, featureidkey ="properties.ZIPCODE",
                           locations= crime_joined.ZIPCODE ,color = 'POPULATION', color_continuous_scale= quartile_sca
                           mapbox_style='open-street-map',
                           zoom=9,
                           opacity = 0.65,
                           width = 1000, height = 700,
                           center = {"lat":  34.0500, "lon": -118.4105}

                          )
pop_fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})



pop_fig.show()
```



Visually, the two maps seem pretty similar, but not exact. Let's see which zipcodes are being binned differently

Quantifying and visualizing similarities between population / crime choropleths

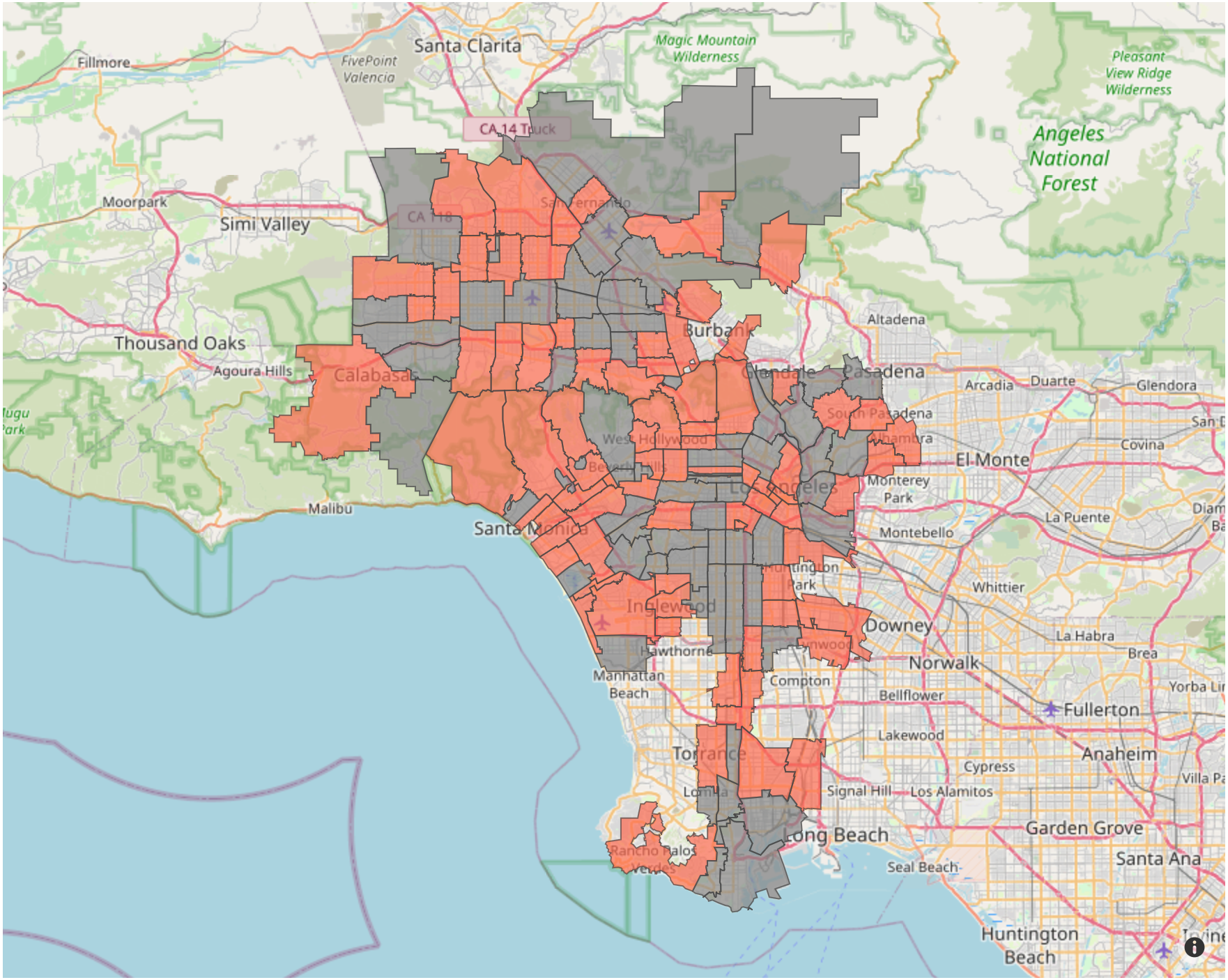
In [18]:
```
#pulling a smaller dataframe
crime_comparison_viz = crime_joined[['ZIPCODE','crimes','POPULATION']]

#getting quartiles for crimes and for population (pandas.qcut is genius)
crime_comparison_viz['crimes_quartile'] = pd.qcut(crime_comparison_viz['crimes'], 4, labels=['Q1', 'Q2', 'Q3', 'Q4'])
crime_comparison_viz['pop_quartile'] = pd.qcut(crime_comparison_viz['POPULATION'], 4, labels=['Q1', 'Q2', 'Q3', 'Q4'])


#make boolean same_quartile column if they are matched
crime_comparison_viz['same_quartile'] = crime_comparison_viz['pop_quartile'] == crime_comparison_viz['crimes_quartile']



#getting the percentage of quartiles that are being binned the same, will display below choropleth
quartile_identity = crime_comparison_viz['same_quartile'].sum() / crime_comparison_viz.shape[0]


crime_comparison_viz.head(3)
```

Out[18]:

	ZIPCODE	crimes	POPULATION	crimes_quartile	pop_quartile	same_quartile
0	90001	3147	57652	Q2	Q4	False
1	90002	8900	53108	Q3	Q4	False
2	90003	25861	75024	Q4	Q4	True

In [19]:
```
fig = px.choropleth_mapbox(crime_comparison_viz, geojson= parsed_json, featureidkey ="properties.ZIPCODE",
                           locations= crime_joined.ZIPCODE ,color = 'same_quartile',
                           mapbox_style='open-street-map', color_discrete_sequence = ['tomato','grey'],
                           zoom=9,
                           opacity = 0.65,
                           width = 1000, height = 700,
                           center = {"lat":  34.0500, "lon": -118.4105}

                          )
fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})



fig.show()

print(f"Matched Zipcodes:{crime_comparison_viz['same_quartile'].sum()}")
```

```
print(f'Total Zipcodes:{crime_comparison_viz.shape[0]}')
print('-------------------------------------------------')
print(f'Percentage of Matched Quartiles: {quartile_identity:.2f}')
```



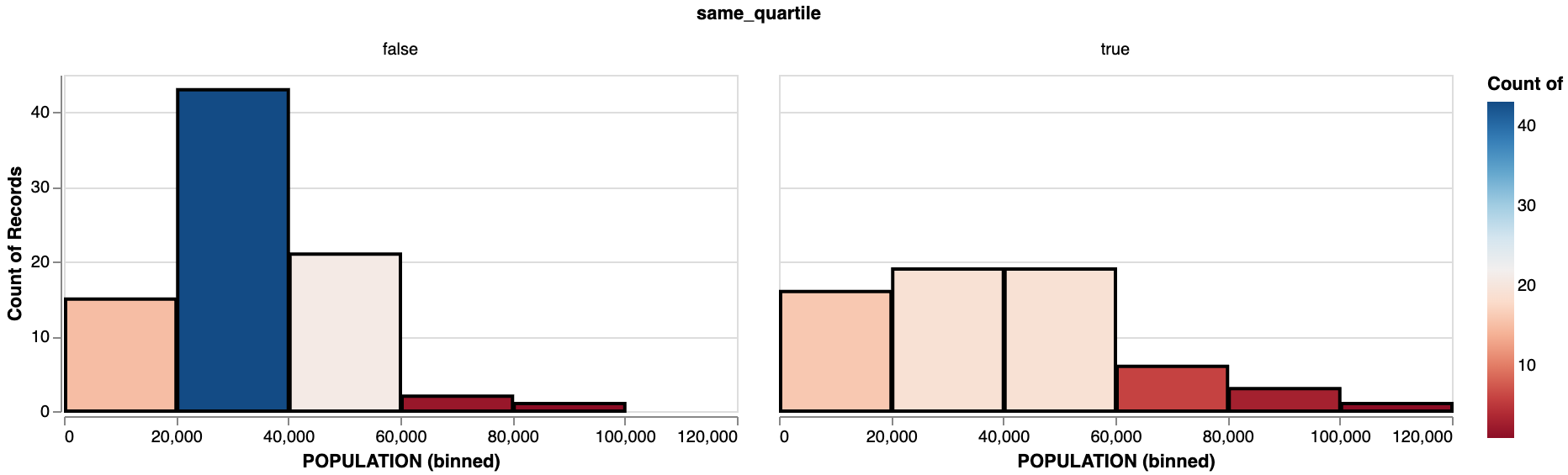
```
Matched Zipcodes:64
Total Zipcodes:146
-------------------------------------------------
Percentage of Matched Quartiles: 0.44
```

Understanding the regions binned differently

Lets first see the distributions of populations that were binned correctly or incorrectly

In [22]:
```
pop_density = crime_joined[['ZIPCODE','population_density']]

crime_with_population_density = crime_comparison_viz.merge(pop_density, on = 'ZIPCODE')


alt.Chart(crime_with_population_density).mark_bar(stroke = 'black', strokeWidth = 2).encode(
    x = alt.X('POPULATION:Q',bin=True),
    y = alt.Y('count()'),
    tooltip = ['count()'],
    color = alt.Color('count()',scale = alt.Scale(scheme = 'redblue'))).properties(width = 400, height = 200).facet('sa
```

Out[22]:

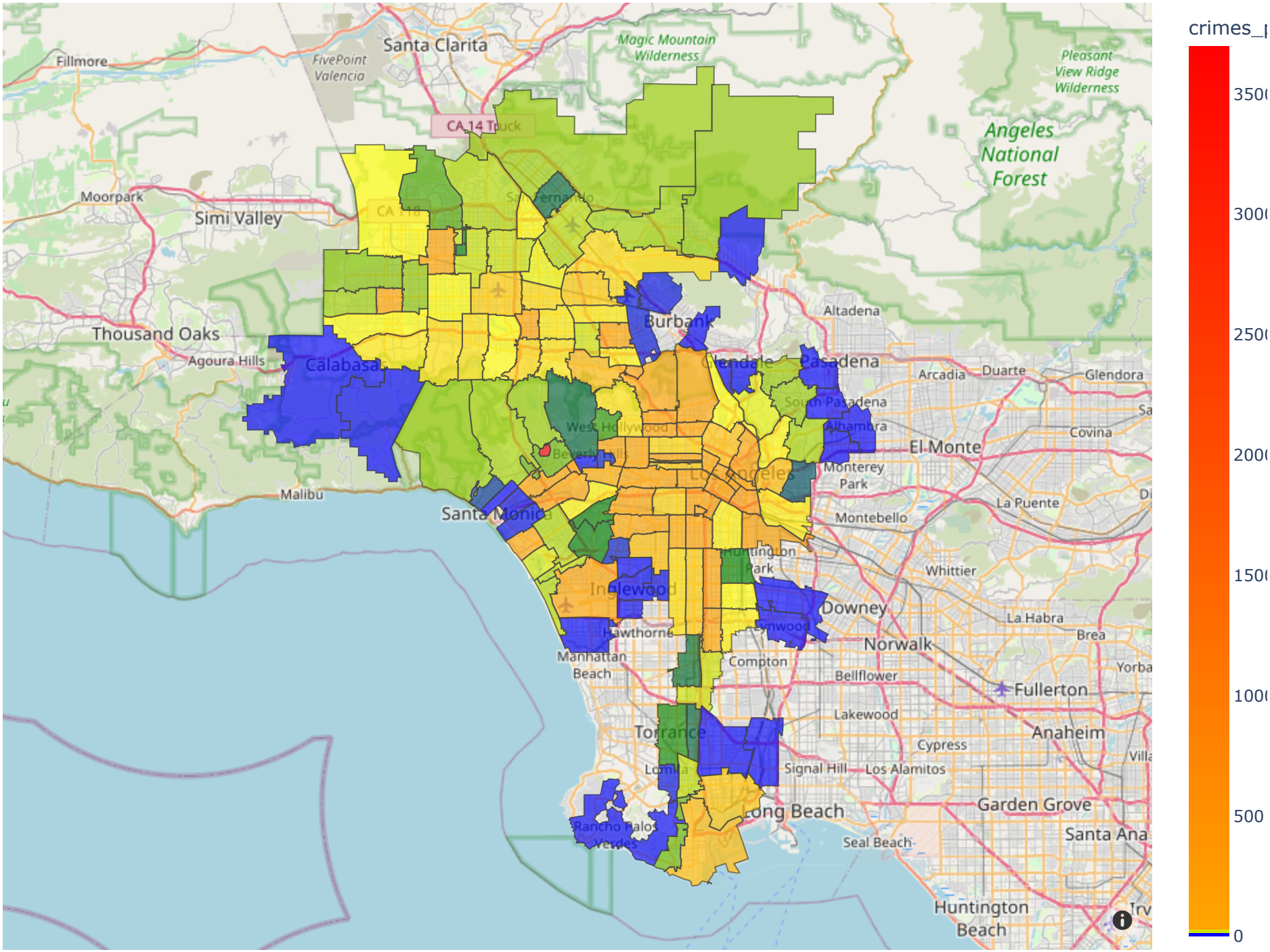


Doesnt seem to be a huge difference in population distributions b/w the values that binned similarly or not

Per Capita Normalization `per 100 people` in zipcode

In [23]:
```
#per capita calculatin - (crimes/population)*100
crime_comparison_viz['crimes_per_100'] = (crime_comparison_viz['crimes'] / crime_comparison_viz['POPULATION']) * 100

#Changing inf values if population == inf to NaN
crime_comparison_viz['crimes_per_100'].replace([np.inf, -np.inf], np.nan, inplace=True)
```

In [24]:
```
q1 = crime_comparison_viz['crimes_per_100'].quantile(0.25)
q2 = crime_comparison_viz['crimes_per_100'].quantile(0.50)  # median
q3 = crime_comparison_viz['crimes_per_100'].quantile(0.75)
q4 = crime_comparison_viz['crimes_per_100'].max()


# The scale will have distinct colors at each quartile
quartile_scale = [
    [0, 'blue'],                # Start with blue
    [q1/q4, 'green'],           # Green at first quartile
    [q2/q4, 'yellow'],          # Yellow at median
    [q3/q4, 'orange'],          # Orange at third quartile
    [1.0, 'red']                # Red at maximum
]


per_capita_fig = px.choropleth_mapbox(crime_comparison_viz, geojson= parsed_json,
                           featureidkey ="properties.ZIPCODE",
                           locations= crime_comparison_viz.ZIPCODE ,color = 'crimes_per_100',
                           color_continuous_scale= quartile_scale,
                           mapbox_style='open-street-map',
                           zoom=9,
                           hover_data = ['crimes_per_100','crimes','POPULATION'],
                           opacity = 0.65,
                           width = 1000, height = 700,
                           center = {"lat":  34.0500, "lon": -118.4105}

                          )
per_capita_fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})



per_capita_fig.show()
```



Population Density Normalized Choropleth

population density in this case is defined as the

$$PopDensity = \frac{population}{area(sqmi)}$$

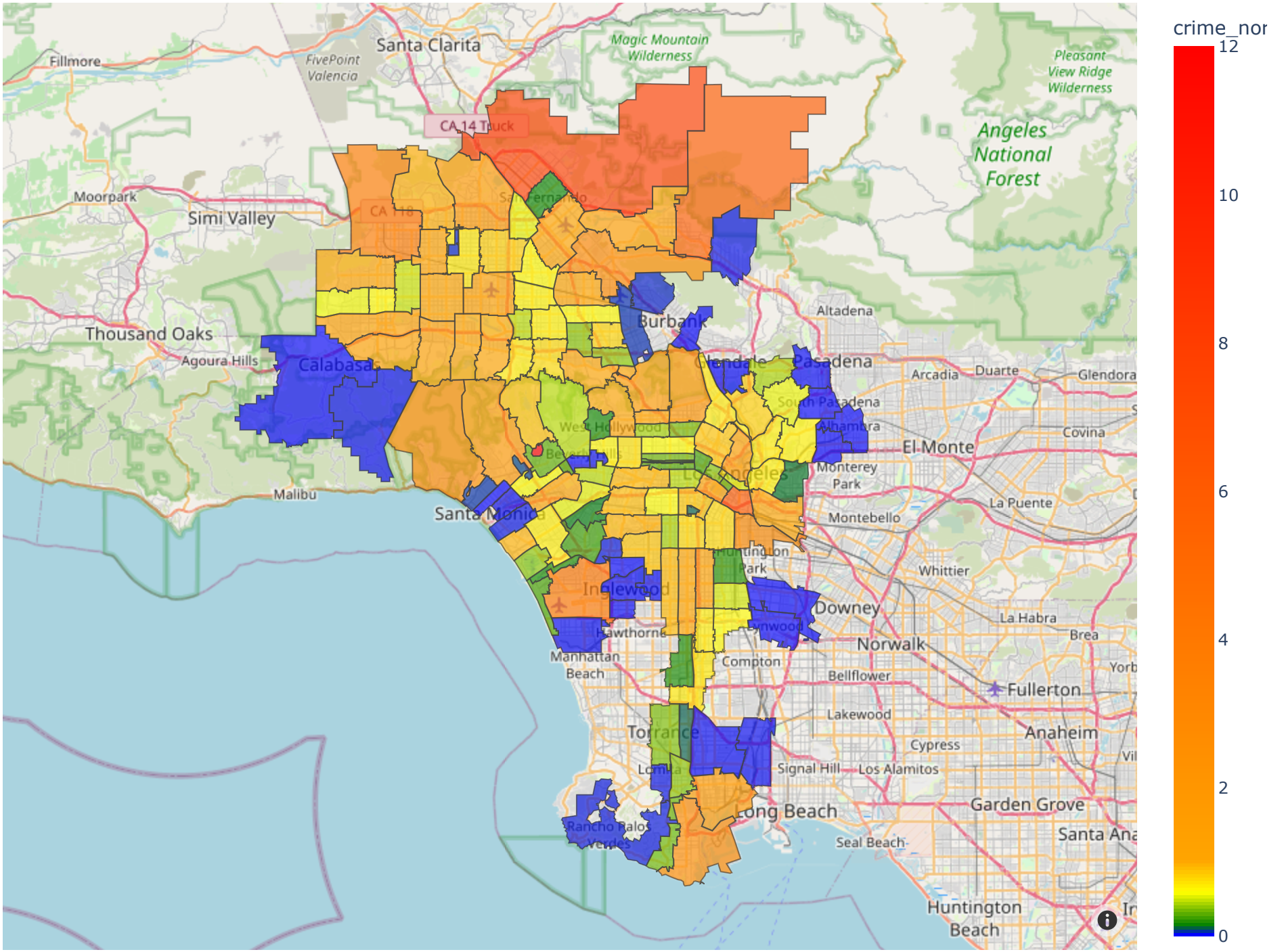
Then the normalization is accomplished by:

$$CrimeNormalized = \frac{crimes}{PopDensity}$$

In [25]:
```
#adding a populaton density column which is population/square mile
crime_joined['population_density'] = crime_joined['POPULATION'] / (crime_joined['zipcode_area'] * 3.86102e-7)


#normalizing crime values by population density
crime_joined['crime_normalized'] = crime_joined['crimes'] / crime_joined['population_density']
```

In [26]:
```
q1 = crime_joined['crime_normalized'].quantile(0.25)
q2 = crime_joined['crime_normalized'].quantile(0.50)  # median
q3 = crime_joined['crime_normalized'].quantile(0.75)
q4 = crime_joined['crime_normalized'].max()


# The scale will have distinct colors at each quartile
quartile_scale = [
    [0, 'blue'],                # Start with blue
    [q1/q4, 'green'],           # Green at first quartile
    [q2/q4, 'yellow'],          # Yellow at median
    [q3/q4, 'orange'],          # Orange at third quartile
    [1.0, 'red']                # Red at maximum
]



density_normalized = px.choropleth_mapbox(crime_joined, geojson= parsed_json, featureidkey ="properties.ZIPCODE",
                           locations= crime_joined.ZIPCODE ,color = 'crime_normalized', color_continuous_scale= quarti
                           mapbox_style='open-street-map',
                           zoom=9,
                           opacity = 0.65,
                          hover_data = ['crimes','population_density','crime_normalized'],
                           width = 1000, height = 700,
                           center = {"lat":  34.0500, "lon": -118.4105}

                          )
density_normalized.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
```

```
density_normalized.show()
```



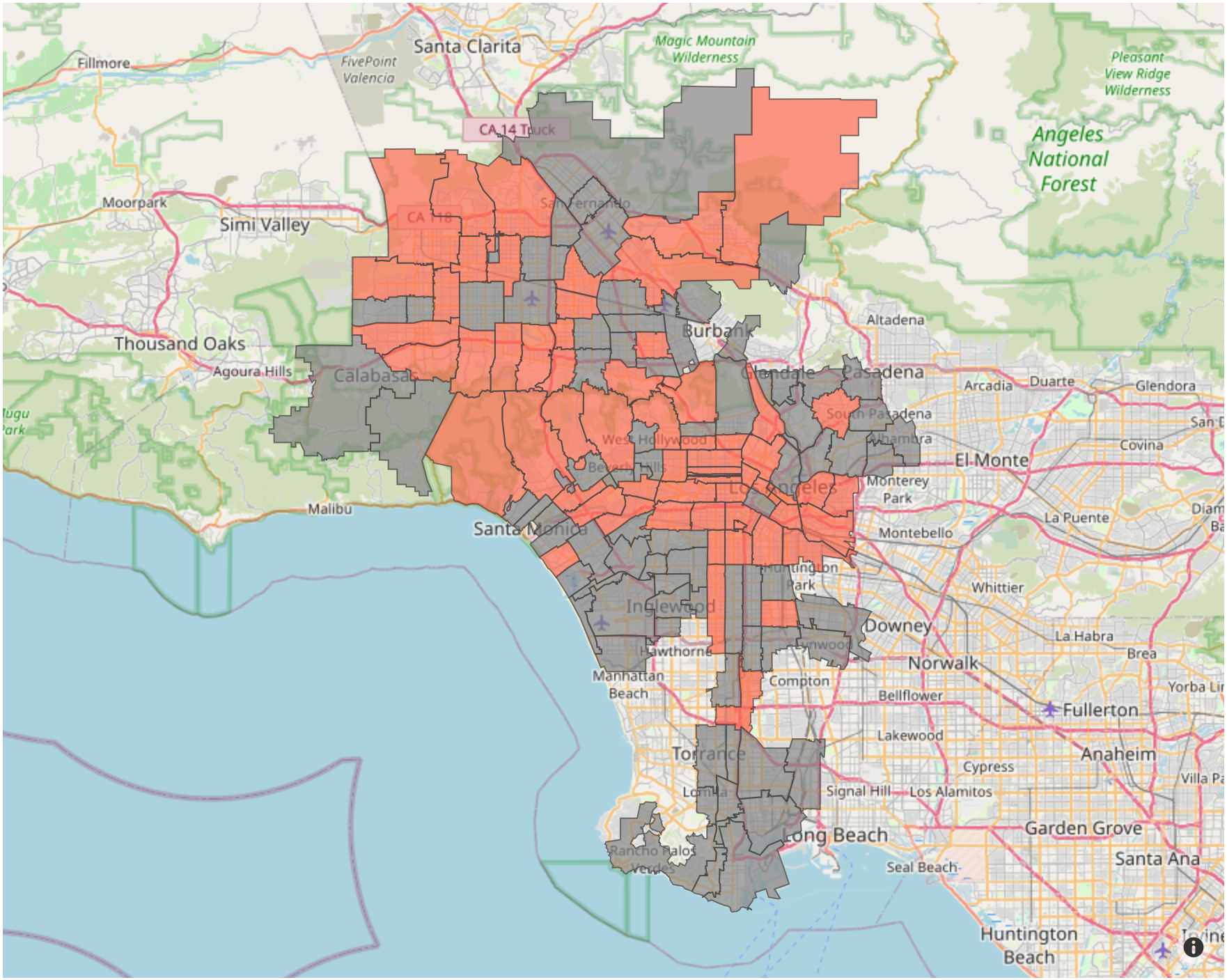
How different is this from the pure crime choropleth?

In [27]:
```
#pulling a smaller dataframe
normalizing_comp_viz = crime_joined[['ZIPCODE','crimes','crime_normalized']]

#getting quartiles for crimes and for population (pandas.qcut is genius)
normalizing_comp_viz['crimes_quartile'] = pd.qcut(normalizing_comp_viz['crimes'], 4, labels=['Q1', 'Q2', 'Q3', 'Q4'])
normalizing_comp_viz['norm_quartile'] = pd.qcut(normalizing_comp_viz['crime_normalized'], 4, labels=['Q1', 'Q2', 'Q3',


#make boolean same_quartile column if they are matched
normalizing_comp_viz['same_quartile'] = normalizing_comp_viz['norm_quartile'] == normalizing_comp_viz['crimes_quartile



#getting the percentage of quartiles that are being binned the same, will display below choropleth
quartile_identity = normalizing_comp_viz['same_quartile'].sum() / normalizing_comp_viz.shape[0]


normalizing_comp_viz.head(3)
```

Out[27]:

	ZIPCODE	crimes	crime_normalized	crimes_quartile	norm_quartile	same_quartile
0	90001	3147	0.187021	Q2	Q2	True
1	90002	8900	0.497189	Q3	Q2	False
2	90003	25861	1.268516	Q4	Q4	True

In [28]:
```
fig = px.choropleth_mapbox(normalizing_comp_viz, geojson= parsed_json, featureidkey ="properties.ZIPCODE",
                           locations= crime_joined.ZIPCODE ,color = 'same_quartile',
                           mapbox_style='open-street-map', color_discrete_sequence = ['grey','tomato'],
                           zoom=9,
                           opacity = 0.65,
                           width = 1000, height = 700,
                           center = {"lat":  34.0500, "lon": -118.4105}

                          )
fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})



fig.show()

print(f"Matched Zipcodes:{crime_comparison_viz['same_quartile'].sum()}")
print(f'Total Zipcodes:{crime_comparison_viz.shape[0]}')
print('-------------------------------------------------')
print(f'Percentage of Matched Quartiles: {quartile_identity:.2f}')
```



Matched Zipcodes:64
Total Zipcodes:146

Percentage of Matched Quartiles: 0.55

In [ ]:

In [ ]:

In [ ]: