In [1]:
```
import pandas as pd
import eeweather as ee
from folium import plugins
import folium
from folium.plugins import HeatMap
import plotly.express as px
import warnings
import matplotlib.pyplot as plt
%matplotlib inline
```

Read in all of the `csv` files as pandas DataFrames. ¶

In [2]:
```
df1 = pd.read_csv('Dataset_1.csv')
df2 = pd.read_csv('Dataset_2.csv')
df3 = pd.read_csv('Dataset_3.csv')
```

Assignment #3

For assignment three, we were asked to build a heat map to visualize which zip codes have the highest number of responses to a survey data set. We're also told that the zip codes are provided either in the zip-five or the zip-nine format. This assignment asked us to use Power BI or other data visualization tools; however, since I don't have Power BI on my laptop, I decided to create a heat map using the folium Python package. In passing, I will say that the approach that I've taken is not the best, and if I had more time, I would prefer to do this using what's called a choropleth map or even doing this analysis in Power BI.

All of the code for this and the other assignments can be found at the Github repository below. https://github.com/grantaguinaldo/bi-work/blob/main/bi-work-assignment-03.ipynb (https://github.com/grantaguinaldo/bi-work/blob/main/bi-work-assignment-03.ipynb)

In [3]:
```
df3.head()
```

Out[3]:

	ResponseID	Response_Zip_Code
0	1000	93440
1	1001	928053811
2	1002	900621538
3	1003	900021746
4	1004	932574653

In [4]:
```
print('The dataset contains {} rows, and {} columns.'.format(df3.shape[0], df3.shape[1]))
```

```
The dataset contains 2386 rows, and 2 columns.
```

High level summary of the data

The first thing that I did was look at the data to see what I have to approach this problem. In this case, I have two columns, a response ID and zip code columns.

Next, I printed out some statistics on the data and saw that we have 2,386 rows, two columns. We see no missing data because no null values are the same as a row count. We also see that both data types are integers.

In [5]:
```
# Understand the data types in the dataset.
df3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2386 entries, 0 to 2385
Data columns (total 2 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   ResponseID         2386 non-null   int64
 1   Response_Zip_Code  2386 non-null   int64
dtypes: int64(2)
memory usage: 37.4 KB
```

In [6]:
```
print('There are {} unique response ID in the dataset.'.format(len(list(set(df3.ResponseID.to_lis
t())))))
```

```
There are 2386 unique response ID in the dataset.
```

In [7]:
```
print('There are {} unique zipcodes (zip-5 and zip-9) in the dataset.'.format(len(list(set(df3.Re
sponse_Zip_Code.to_list())))))
```

```
There are 2316 unique zipcodes (zip-5 and zip-9) in the dataset.
```

Observations about the zip codes

Again, some of the zip code fields contain either five or nine digits. In this case, what I've done in the past is always to call that the zip+five (or the zip code). On the other hand, if I have the zip code in five digits, I'll take that number directly as the zip code.

In [8]:
```
# See that the len of the zipcode differs.
# Assume that the zip-5 is the first five digits from the left
# and the four digits to the left make up the zip-9.
df3.head()
```

Out[8]:

	ResponseID	Response_Zip_Code
0	1000	93440
1	1001	928053811
2	1002	900621538
3	1003	900021746
4	1004	932574653

Converting column names to lower case

In the code below, I am taking the column headers converting it to lowercase.

In [9]:
```
df3.columns = map(str.lower, df3.columns)
```

Converting all zip codes to the zip-five format

In this code block, I'm converting all of the zip codes to a five-digit number. If the number is already in five digits, I returned the five-digit zip code; however, if the number is nine digits, I take the five digits from the left and return that number as the zip code.

In [10]:
```
#Split out the zip-5 portion of the zip code assuming that five digit zip starts from the left.
df3['zip_5'] = df3['response_zip_code'].astype(str).str[:5].astype(int)
df3.head()
```

Out[10]:

	responseid	response_zip_code	zip_5
0	1000	93440	93440
1	1001	928053811	92805
2	1002	900621538	90062
3	1003	900021746	90002
4	1004	932574653	93257

In [11]:
```
# Determine the number of unique zip-5 codes in the dataset.
print('Number of Unique Zipcodes (zip-5): {}'.format(len(set(df3.zip_5.to_list()))))
```

```
Number of Unique Zipcodes (zip-5): 327
```

Groupby Function to aggregate data

Now that I have the five-digit zip codes, I will do a groupby to aggregate the data to get the counts per zip code.

In [12]:
```
# Aggregate the individual records by zipcode and get the count of each.
df_zip_summary = df3.groupby('zip_5', as_index=False).count()[['zip_5', 'response_zip_code']]
df_zip_summary.head()
```

Out[12]:

	zip_5	response_zip_code
0	90001	19
1	90002	29
2	90003	27
3	90004	8
4	90006	3

In [13]:
```
print('The total number of rows of data is: {}'.format(df_zip_summary.response_zip_code.sum()))
```

```
The total number of rows of data is: 2386
```

Reverse Geocode to build heatmap

Since I am doing this visualization by scratch, I will need to do a reverse geocode and get latitude and longitude from each zip code. To do the reverse geocoding, I am using a Python library called EEweather. In passing, I have used this library in the past for the meter-based work I did with the energy efficiency group.

In [14]:
```
# For visualization purposes, we need to get the lat and lon of each zip code.
# We will use the eeweather library maintained from Recurve to get the lat lon.

df_zip_summary['LAT'] = ''
df_zip_summary['LNG'] = ''
index_except = []

for index, row in df_zip_summary.iterrows():
    try:
        df_zip_summary.at[index, 'LAT'] = ee.zcta_to_lat_long(row['zip_5'])[0]
        df_zip_summary.at[index, 'LNG'] = ee.zcta_to_lat_long(row['zip_5'])[1]
    except:
        index_except.append(index)
index_except
```

Out[14]:
```
[]
```

In [15]:
```
# To get the response frequency per zip code, we need to also determine
# the total amount of respondents in each zip code.
print('Total Number of Responses Received: {}'.format(df_zip_summary.response_zip_code.sum()))
```

```
Total Number of Responses Received: 2386
```

In [16]:
```
# Calculate the response frequency per zip code.
df_zip_summary['freq'] = 100*(df_zip_summary['response_zip_code'] / df_zip_summary.response_zip_c
ode.sum())
df_zip_summary.head()
df_zip_summary['freq_scale'] = df_zip_summary['freq'] * 10
```

Summary statistics of the data

The code below returns summary statistics for the data as a whole. We see that the median response rate is about 0.16%, and the highest response rate is about 1.55%.

In [17]:
```
# Compute the summary statistics on the response frequency.
df_zip_summary.describe()
```

Out[17]:

	zip_5	response_zip_code	freq	freq_scale
count	327.000000	327.000000	327.000000	327.000000
mean	91761.036697	7.296636	0.305810	3.058104
std	1159.129777	8.309619	0.348266	3.482657
min	90001.000000	1.000000	0.041911	0.419111
25%	90708.000000	1.000000	0.041911	0.419111
50%	91761.000000	4.000000	0.167645	1.676446
75%	92790.500000	9.000000	0.377200	3.772003
max	93654.000000	37.000000	1.550712	15.507125

Subset data for the top quartile (to show zip codes with highest response rate)

The code below will be used to create the heatmap based on the top quantile of the data. From the map, there not a lot of activity in Orange County. Moreover, from the map, you can see that the response rates are quite high in Los Angeles County, Downtown LA in the Valley, and in Visalia and Hanford -- Hanford has the highest response rate at 1.55%.

In [18]:
```
# Subset the data so that we only return the top quartile of the survey responses.
# Consider these zipcodes to be the higest number of respones.
df_high_resp_q4 = df_zip_summary[df_zip_summary.freq > \
                0.377200].sort_values(by='freq',
                ascending=False).reset_index(drop=True)
df_high_resp_q4.head()
```

Out[18]:

	zip_5	response_zip_code	LAT	LNG	freq	freq_scale
0	93230	37	36.288	-119.623	1.550712	15.507125
1	93436	37	34.6056	-120.397	1.550712	15.507125
2	90011	36	34.0071	-118.259	1.508801	15.088013
3	90008	35	34.0096	-118.347	1.466890	14.668902
4	90044	34	33.9528	-118.292	1.424979	14.249790

In [19]:
```
# Create the Folium map object that will be used to create the heatmap.
m = folium.Map([34.052235, -118.243683], zoom_start=9)

# Create markers on the map that represent the centroid of each zip code in the dataset.
for index, row in df_high_resp_q4.iterrows():
    folium.CircleMarker([row['LAT'], row['LNG']],
                        radius=0.001,
                        fill_color="#3db7e4", alpha=0.1, transparent=True,
                        tooltip='Zipcode: ' + str(row['zip_5']) + ' || Response Freq: ' + str(row
['freq'])
                       ).add_to(m)


#Overlay response data onto the map.
stationArr = df_high_resp_q4[['LAT', 'LNG', 'freq_scale']].values

m.add_child(plugins.HeatMap(stationArr, radius=10))
m
```

```
Out[19]: Make this Notebook Trusted to load map: File -> Trust Notebook
```

Subset data for the lowest 50% of the data (to show zip codes with lowest response rate)

Besides looking at the zip codes with the highest response rates, I also looked at the zip codes representing the bottom 50% of the dataset. When you visualize this data, you start to see lower response rates in Orange County.

In [20]:

```
# Subset the data so that we only return the lower two quartiles of the survey responses.
# Consider these zipcodes to be the higest number of respones.
df_low50 = df_zip_summary[df_zip_summary.freq < \
               0.167645].sort_values(by='freq',
               ascending=False).reset_index(drop=True)
df_low50.head()
```

Out[20]:

	zip_5	response_zip_code	LAT	LNG	freq	freq_scale
0	92802	4	33.8083	-117.924	0.167645	1.676446
1	92860	4	33.9247	-117.552	0.167645	1.676446
2	90007	4	34.0281	-118.285	0.167645	1.676446
3	91786	4	34.1052	-117.662	0.167645	1.676446
4	91789	4	34.0183	-117.855	0.167645	1.676446

In [21]:
```
# Create the Folium map object that will be used to create the heatmap.
m = folium.Map([34.052235, -118.243683], zoom_start=9)

# Create markers on the map that represent the centroid of each zip code in the dataset.
for index, row in df_low50.iterrows():
    folium.CircleMarker([row['LAT'], row['LNG']],
                        radius=0.001,
                        fill_color="#3db7e4", alpha=0.1, transparent=True,
                        tooltip='Zipcode: ' + str(row['zip_5']) + ' || Response Freq: ' + str(row
['freq'])
                       ).add_to(m)


#Overlay response data onto the map.
stationArr = df_low50[['LAT', 'LNG', 'freq_scale']].values

m.add_child(plugins.HeatMap(stationArr, radius=10))
m
```

```
Out[21]: Make this Notebook Trusted to load map: File -> Trust Notebook
```

Summary

While I am unsure of the entire story of the data, you can see from these visualizations that certain zip codes in LA County have high response rates (including Visalia), and certain zip codes in Orange Count have lower response rates. Again, I don't think that the heatmap that I have created is the best for telling a story with these data, but with some supplemental analysis, you can glean that something is going on with the response rates between Orange and LA County.