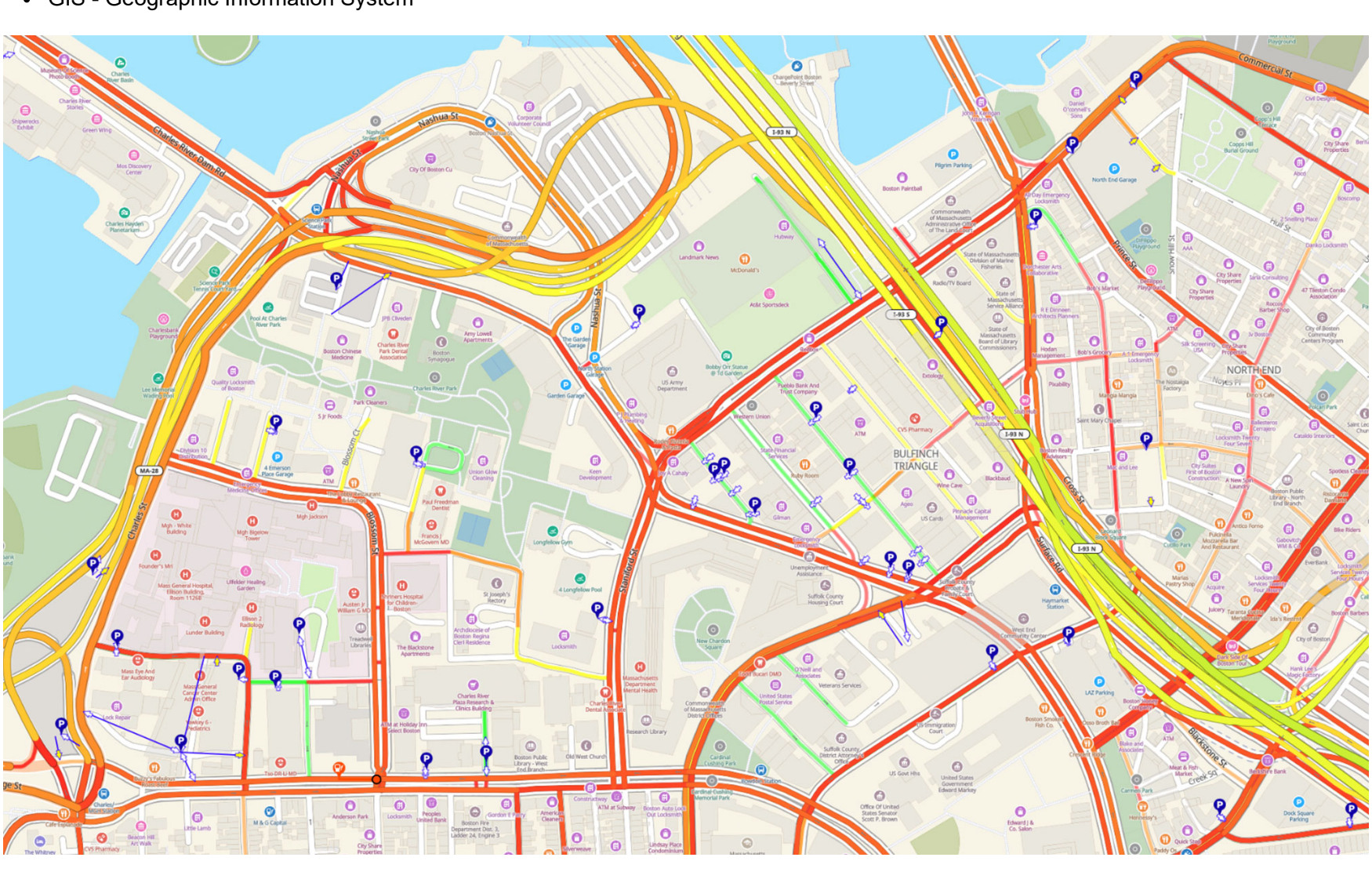
Today's agenda

- Location analytics
  - Real-time geographic DA
  - Historical geographic DA
- Important fields where LA is used
- Python pacakges for Geospatial visualization
- Earthquake data preparation
- Earthquake data visualization

In [ ]:

Location Analytics - Geospatial Visualization

- The ability to gain insight from the location or geographic component of business data.
- The important component is the location data.
- GIS - Geographic Information System



Credits - Image from Internet

In [ ]:

Real-time Geographic Data Analytics

Getting insights from the data that comes into the system and relating that to a particluar location is called Real-time Geographic data analytics.

- Getting route navigation in Google Maps
- Courier and postal services
- Military serives
  - Getting the exact location of the enemy movements on the map to get informed

Credits - Image from Internet

In [ ]:

Historic Geographic Data Analytics

Getting insights from the historic data so as to predict the furture occurrence or disaster (purely by chance) completely based on the geographic location.

- Predict and prevent (taking necessary steps) the occurrence based on the past data
- Disaster prevention efforts
  - Floods
  - Volcanoes
  - Earthquakes

- Chain retailers benefit from the historical spending habits based on location to increase the sales

Credits - Image from Internet

In [ ]:

Important fields/topics

Location analytics (Geospatial science) can be used in every field/study where data is generated with respect to the location.

Credits - Image from Internet

- Demographic study - statistical study of population of all the areas based on location
  - Analysis based on education details
  - Analysis based on ethnicity
  - Analysis based on religion
  - Analysis based on births and deaths
  - ...
- Measurement of **Natural Occurrences** that happen on timely basis
- Increase or decrease of pollution in a particular city or country
- Geographic crime analysis
  - Analyse the crime data
  - Identify the crime types and hotspots
  - Help law enforcement plan crime prevention
- ETC

In [ ]:

Python packages for Geospatial Visualization

- GeoViews → explore & visualize geographical, meteorological, and oceanographic datasets
- Folium → widely used geospatial data visualization library built on Leaflet.js framework
- Plotly → Interactive map visualization - effectively uses Mapbox api
- KeplerGL → built exclusively for jupyter notebook to visualize big geospatial data
- GeoPandas → workhorse for working geo-data
- geonamescache → used to retrieve location datasets in the form of python dictionaries

In [ ]:

`import` packages

In [1]:
```
import pandas as pd
import plotly.graph_objects as go
```

If you do not have the above packages, you can install by typing these commands on `Command Prompt` (CMD) -

- `pip install pandas --user`
- `pip install plotly --user`

Dataset description

Earthquake data (from Yesterday) - The data is obtained from `USGS` datasources. The data is updated every 1 minute. In this example we don't deal with streaming data.

- time
- latitude
- longitude
- mag (magnitude)
- place

In [2]:
```
eqdf = pd.read_csv('all_day.csv')
eqdf.shape
```

Out[2]:
```
(355, 22)
```

In [3]:
```
print(eqdf.columns)
```

```
Index(['time', 'latitude', 'longitude', 'depth', 'mag', 'magType', 'nst',
       'gap', 'dmin', 'rms', 'net', 'id', 'updated', 'place', 'type',
       'horizontalError', 'depthError', 'magError', 'magNst', 'status',
       'locationSource', 'magSource'],
      dtype='object')
```

In [4]:
```
eqdf = eqdf[['time', 'latitude', 'longitude', 'mag', 'place']]
eqdf.head()
```

Out[4]:

	time	latitude	longitude	mag	place
0	2020-12-16T10:15:32.270Z	38.802502	-122.818337	0.55	6km WNW of The Geysers, CA
1	2020-12-16T10:13:43.276Z	35.867401	-96.679527	1.80	9 km WSW of Shamrock, Oklahoma
2	2020-12-16T09:55:30.640Z	19.504999	-155.652328	1.86	22 km ENE of Honaunau-Napoopoo, Hawaii
3	2020-12-16T09:50:35.827Z	17.773700	-65.705800	3.50	31 km SE of Emajagua, Puerto Rico
4	2020-12-16T09:35:49.950Z	37.635834	-118.879669	1.31	9km E of Mammoth Lakes, CA

In [5]:
```
eqdf.shape
```

Out[5]:
```
(355, 5)
```

In [6]:
```
eqdf['place'].head(5)
```

Out[6]:
```
0                6km WNW of The Geysers, CA
1            9 km WSW of Shamrock, Oklahoma
2    22 km ENE of Honaunau-Napoopoo, Hawaii
3         31 km SE of Emajagua, Puerto Rico
4                9km E of Mammoth Lakes, CA
Name: place, dtype: object
```

In [ ]:

Data Preperation

In [7]:
```
# example
dp = '6km WNW of The Geysers, CA'
sa, a = dp.split(', ')
print(sa)
print(a)
```

```
6km WNW of The Geysers
CA
```

Separate `place` by **','**

In [8]:
```
# area_list → split
area_list = eqdf['place'].str.split(', ').to_list()
```

Print first `5` from `area_list`

In [9]:
```
area_list[:5]
```

Out[9]:
```
[['6km WNW of The Geysers', 'CA'],
 ['9 km WSW of Shamrock', 'Oklahoma'],
 ['22 km ENE of Honaunau-Napoopoo', 'Hawaii'],
 ['31 km SE of Emajagua', 'Puerto Rico'],
 ['9km E of Mammoth Lakes', 'CA']]
```

Make two columns `area` and `sub_area` from area_list

In [10]:
```
area = []
sub_area = []

for p in area_list:
    if len(p) == 2:
        sub_area.append(p[0])
        area.append(p[1])
    else:
        sub_area.append(p[0])
        area.append(p[0])
```

Print first `5` from `area`

In [11]:
```
area[:5]
```

Out[11]:
```
['CA', 'Oklahoma', 'Hawaii', 'Puerto Rico', 'CA']
```

Print first `5` from `sub_area`

In [12]:
```
sub_area[:5]
```

Out[12]:
```
['6km WNW of The Geysers',
 '9 km WSW of Shamrock',
 '22 km ENE of Honaunau-Napoopoo',
 '31 km SE of Emajagua',
 '9km E of Mammoth Lakes']
```

Add `area` and `sub_area` as columns in `eqdf`

In [13]:
```
eqdf['sub_area'] = sub_area
eqdf['area'] = area
```

In [14]:
```
# head()
eqdf.head()
```

Out[14]:

	time	latitude	longitude	mag	place	sub_area	area
0	2020-12-16T10:15:32.270Z	38.802502	-122.818337	0.55	6km WNW of The Geysers, CA	6km WNW of The Geysers	CA
1	2020-12-16T10:13:43.276Z	35.867401	-96.679527	1.80	9 km WSW of Shamrock, Oklahoma	9 km WSW of Shamrock	Oklahoma
2	2020-12-16T09:55:30.640Z	19.504999	-155.652328	1.86	22 km ENE of Honaunau-Napoopoo, Hawaii	22 km ENE of Honaunau-Napoopoo	Hawaii
3	2020-12-16T09:50:35.827Z	17.773700	-65.705800	3.50	31 km SE of Emajagua, Puerto Rico	31 km SE of Emajagua	Puerto Rico
4	2020-12-16T09:35:49.950Z	37.635834	-118.879669	1.31	9km E of Mammoth Lakes, CA	9km E of Mammoth Lakes	CA

In [15]:
```
eqdf.shape
```

Out[15]:
```
(355, 7)
```

Remove the column `place` from `eqdf`

In [16]:
```
# prep_df
prep_df = eqdf.drop(columns=['place'], axis=1)
```

In [17]:
```
# head()
prep_df.head()
```

Out[17]:

	time	latitude	longitude	mag	sub_area	area
0	2020-12-16T10:15:32.270Z	38.802502	-122.818337	0.55	6km WNW of The Geysers	CA
1	2020-12-16T10:13:43.276Z	35.867401	-96.679527	1.80	9 km WSW of Shamrock	Oklahoma
2	2020-12-16T09:55:30.640Z	19.504999	-155.652328	1.86	22 km ENE of Honaunau-Napoopoo	Hawaii
3	2020-12-16T09:50:35.827Z	17.773700	-65.705800	3.50	31 km SE of Emajagua	Puerto Rico
4	2020-12-16T09:35:49.950Z	37.635834	-118.879669	1.31	9km E of Mammoth Lakes	CA

Consider the data where magnitude is greater than or equal to `2`

In [18]:
```
# prep_df
prep_df = prep_df[prep_df['mag'] >= 2]
```

In [19]:
```
# shape
prep_df.shape
```

Out[19]:
```
(70, 6)
```

In [ ]:

Data Visualization

Check the frequency of occurrence by `area`

In [20]:
```
# occ_freq → value_counts()
occ_freq = prep_df['area'].value_counts().to_frame()
```

default plot will not be so interactive

In [21]:
```
# occ_freq.plot(kind='pie', subplots=True)
```

In [22]:
```
# head()
occ_freq.head()
```

Out[22]:

	area
Puerto Rico	15
CA	13
Alaska	11
Nevada	8
Hawaii	5

In [23]:
```
occ_freq.index
```

Out[23]:
```
Index(['Puerto Rico', 'CA', 'Alaska', 'Nevada', 'Hawaii', 'Chile', 'Idaho',
       'Japan', 'Philippines', 'Oklahoma', 'Dominican Republic', 'Banda Sea',
       'Washington', 'Honduras', 'Vanuatu'],
      dtype='object')
```

`Pie` chart with `Plotly`

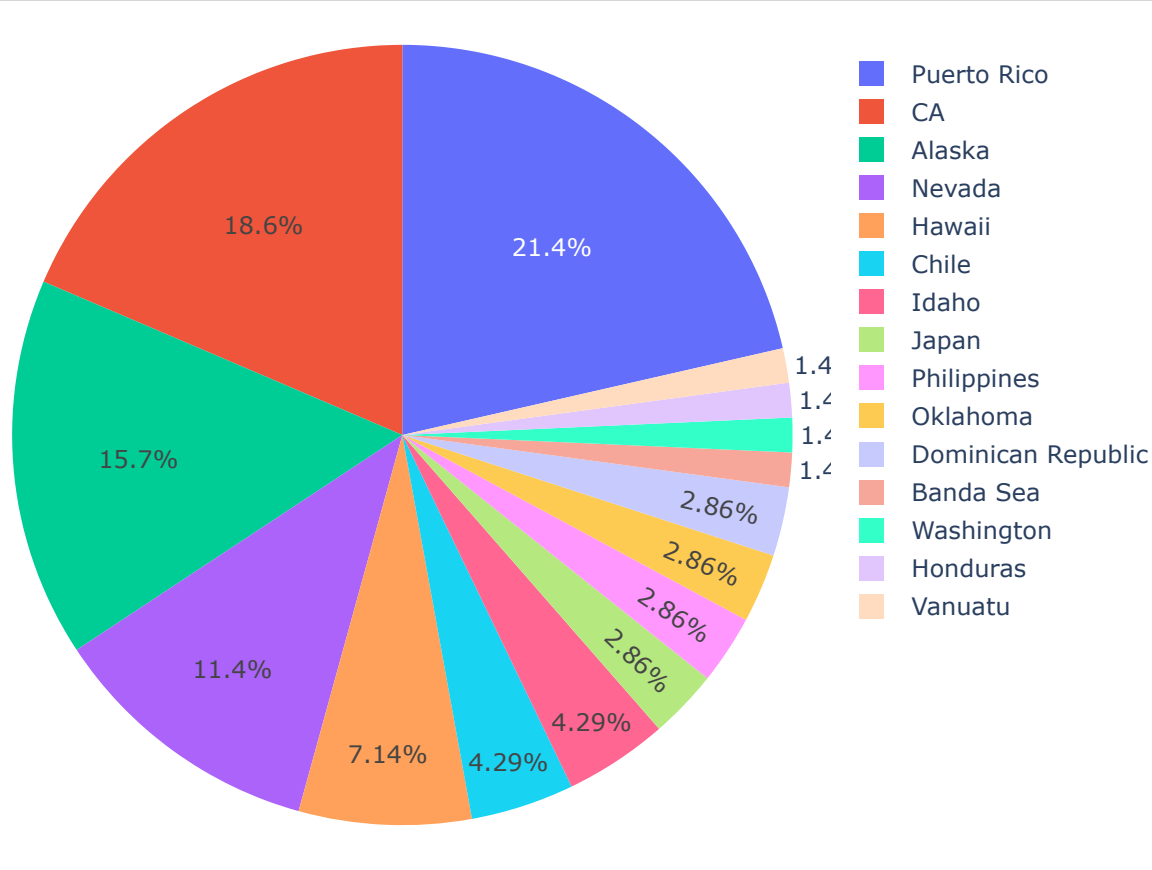
- labels → index
- values → area
- lrtb → 0

refer → https://plotly.com/python/pie-charts/

In [24]:
```
trace = go.Pie(
    labels=occ_freq.index,
    values=occ_freq.area
)

layout = go.Layout(
    height=400,
    width=600,
    margin=dict(l=0, r=0, b=0, t=0)
)

fig = go.Figure(data=[trace], layout=layout)
fig.show()
```



In [ ]:

`Bar` chart of magnitude based on area with `Plotly`

- x → sub_area
- y → mag
- lrtb → 0

refer → https://plotly.com/python/bar-charts/

In [25]:
```
region = 'CA'
# rdf from prep_df
rdf = prep_df[prep_df['area'] == region]
```

In [26]:
```
# head()
rdf.head()
```

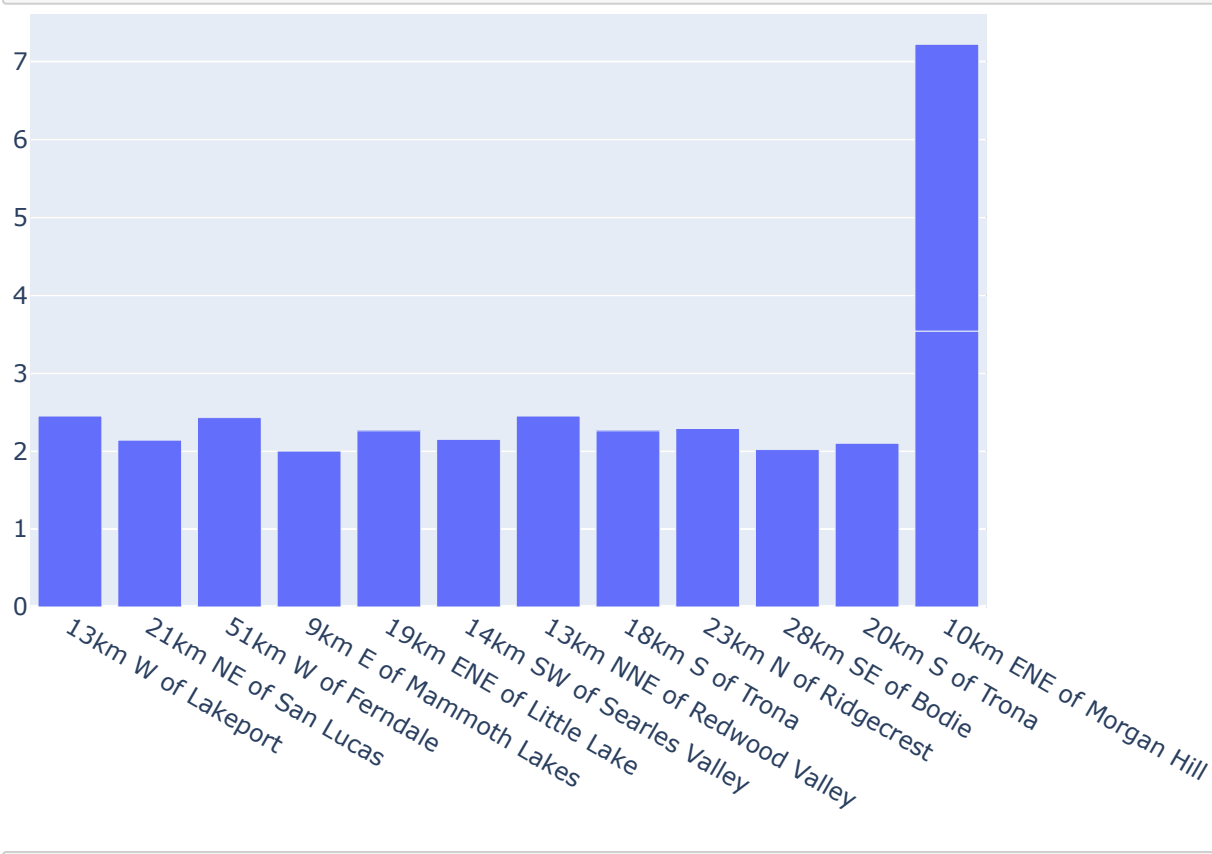
Out[26]:

	time	latitude	longitude	mag	sub_area	area
15	2020-12-16T08:56:57.370Z	39.028500	-123.065002	2.45	13km W of Lakeport	CA
33	2020-12-16T06:57:10.920Z	36.264668	-120.854500	2.14	21km NE of San Lucas	CA
72	2020-12-16T04:18:39.070Z	40.564499	-124.866501	2.43	51km W of Ferndale	CA
81	2020-12-16T03:43:29.610Z	37.635666	-118.880333	2.00	9km E of Mammoth Lakes	CA
88	2020-12-16T03:27:52.560Z	35.990167	-117.706333	2.26	19km ENE of Little Lake	CA

In [27]:
```
trace = go.Bar(
    x=rdf['sub_area'],
    y=rdf['mag']
)

layout = go.Layout(
    height=400,
    width=600,
    margin=dict(l=0, r=0, b=0, t=0)
)

fig = go.Figure(data=[trace], layout=layout)
fig.show()
```



In [ ]:

Map Visualization

- lat → latitudes
- lon → longitudes
- mode → markers
  - size → mag_size
  - color → red
- text → sub_area

refer → https://plotly.com/python/scattermapbox/

In [28]:
```
region = 'CA'

rdf = prep_df[prep_df['area'] == region]
rdf.head()
```

Out[28]:

	time	latitude	longitude	mag	sub_area	area
15	2020-12-16T08:56:57.370Z	39.028500	-123.065002	2.45	13km W of Lakeport	CA
33	2020-12-16T06:57:10.920Z	36.264668	-120.854500	2.14	21km NE of San Lucas	CA
72	2020-12-16T04:18:39.070Z	40.564499	-124.866501	2.43	51km W of Ferndale	CA
81	2020-12-16T03:43:29.610Z	37.635666	-118.880333	2.00	9km E of Mammoth Lakes	CA
88	2020-12-16T03:27:52.560Z	35.990167	-117.706333	2.26	19km ENE of Little Lake	CA

In [29]:
```
region_lats = rdf['latitude'].to_list()
region_lons = rdf['longitude'].to_list()
region_text = rdf['sub_area'].to_list()
```

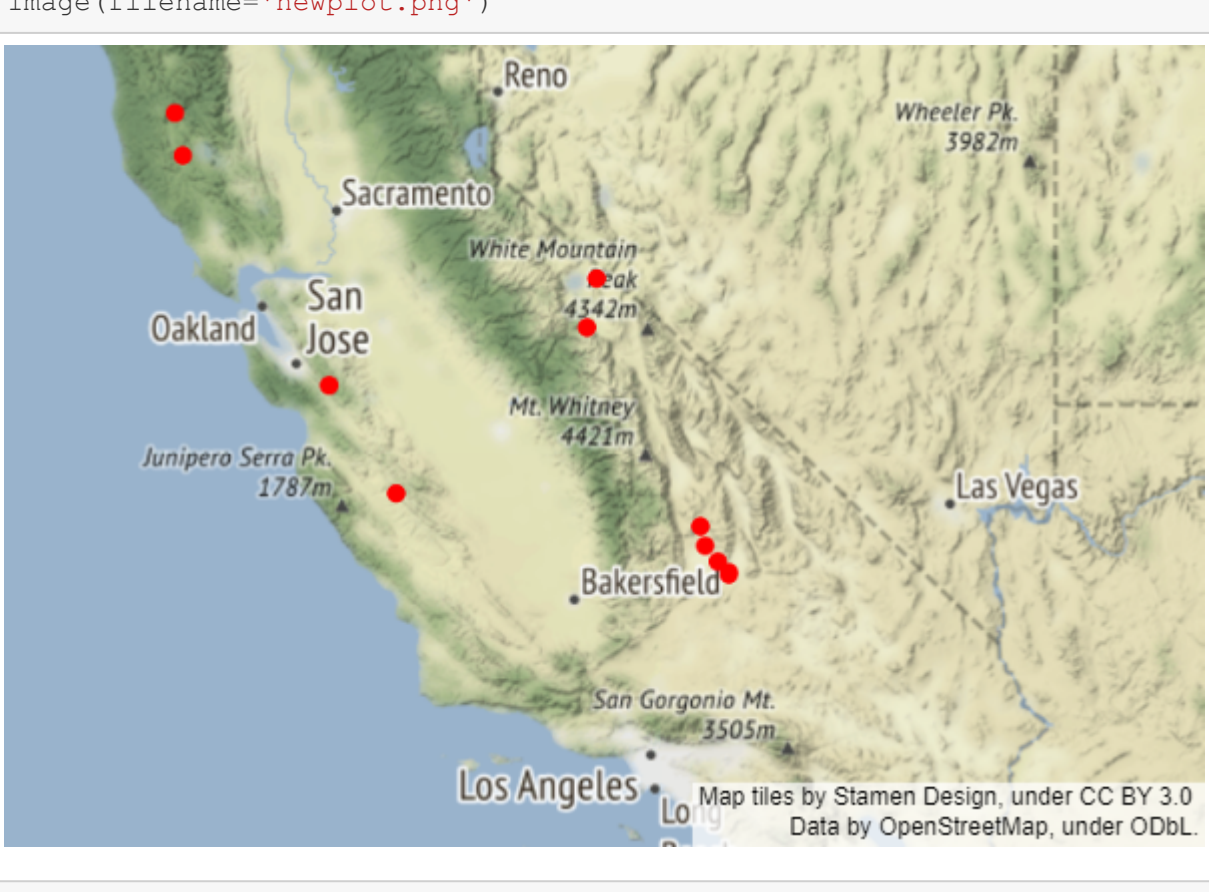
In [30]:
```
trace = go.Scattermapbox(
    lat=region_lats,
    lon=region_lons,
    mode='markers',
    marker=dict(
        size=10,
        color='red'
    ),
    text=region_text,
    hoverinfo='text',
    showlegend=False
)

layout = go.Layout(
    height=400,
    width=600,
    margin=dict(l=0, r=0, t=0, b=0),
    mapbox_style='stamen-terrain',
    mapbox=dict(
        center=dict(
            lat=region_lats[0],
            lon=region_lons[0]
        ),
        zoom=3
    )
)

# fig = go.Figure(data=[trace], layout=layout)
# fig.show()
```

In [31]:
```
from IPython.display import Image
Image(filename='newplot.png')
```

Out[31]:



In [ ]:

What did we learn?

- Location Analytics
- Understand why and how it is important for the business or the govt agency
- Python modules that support geospatial visualization
- Earthquake data - using pandas
  - Data Cleaning
  - Data Filtering
- Earthquake data visualization