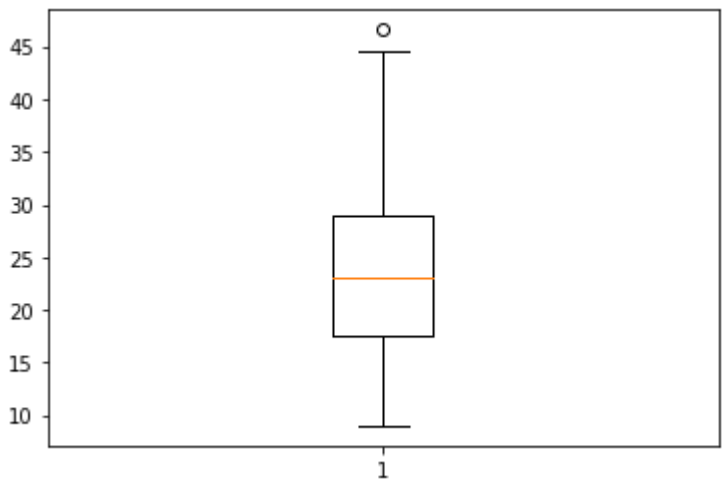
Chap.-2 Advance Visualization

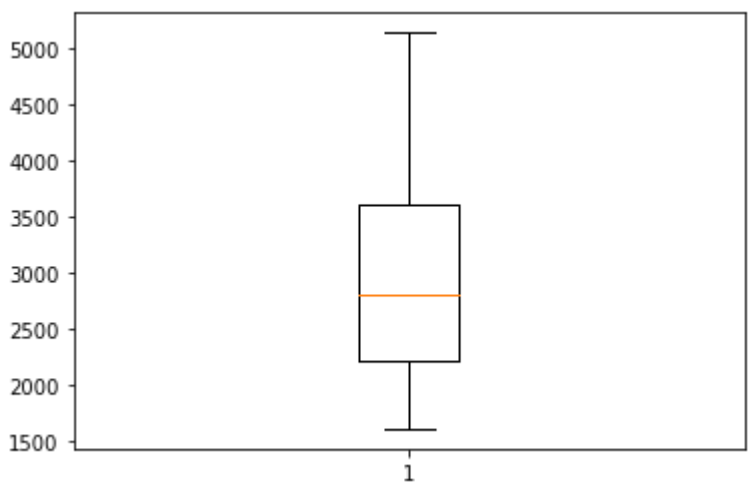
1. Box Plot (boxplot) - to identify outlier (a value that doesn't given)
    notch
    vert
    widths
    patch_artist
2. Scatter (scatter) - relation between 2 var or 2 attribute
    c='r'
    marker='D'
    edgecolor='g'
    linewidths=2
    alpha=0.8
3. Heatmap -
4. Wafflechart - alternate of piechat
5. Regression plot.
6. Wordcloud
7. Area Plot (Area Charts)
8. Geospatial data : latitude and longitudinal

Boxplot - to identify outlier (a value that doesn't given)

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('auto-mpg.csv')
plt.boxplot(df['mpg'])
plt.show()
```

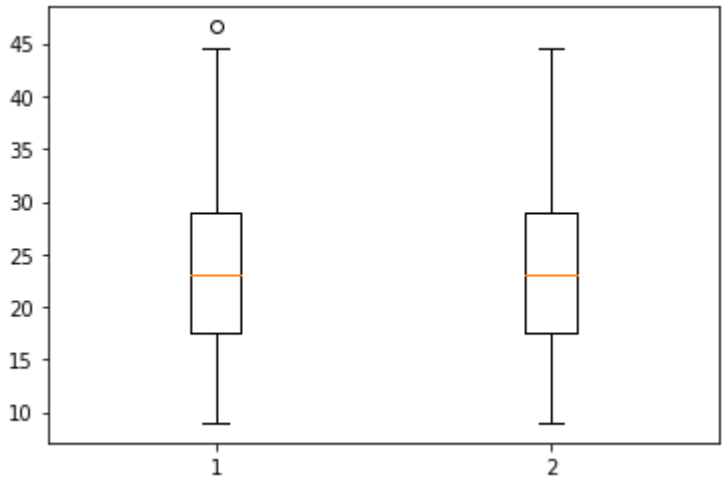


In [2]:
```
df = pd.read_csv('auto-mpg.csv')
plt.boxplot(df['weight'])
plt.show()
```

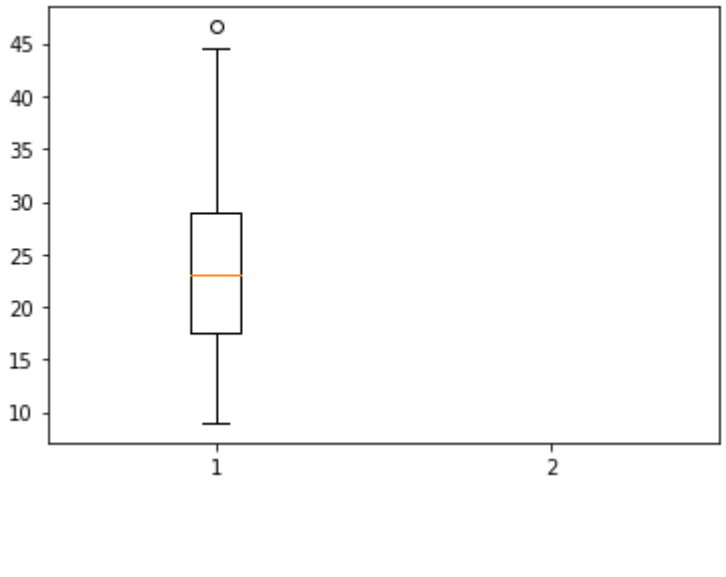


In [3]:
```
df = pd.read_csv('auto-mpg.csv')

def remove_outliers(df,col):
    q1 = df[col].quantile(0.25)
    q3 = df[col].quantile(0.75)
    iqr = q3-q1
    lower = q1-1.5*iqr
    upper  = q3+1.5*iqr
    return (df[(df[col]>=lower)&(df[col]<=upper)])
ndf = remove_outliers(df, 'mpg')
plt.boxplot([df['mpg'],ndf['mpg']])
```

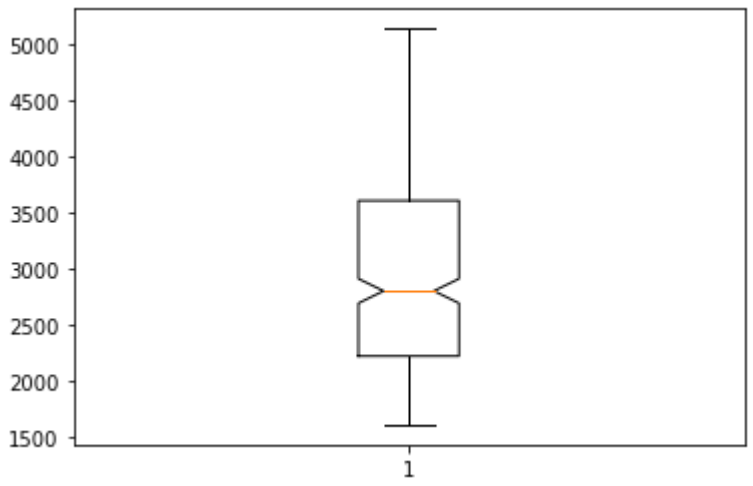
Out[3]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x1e4cef4ccd0>,
  <matplotlib.lines.Line2D at 0x1e4cef59070>,
  <matplotlib.lines.Line2D at 0x1e4cef654f0>,
  <matplotlib.lines.Line2D at 0x1e4cef65850>],
 'caps': [<matplotlib.lines.Line2D at 0x1e4cef593d0>,
  <matplotlib.lines.Line2D at 0x1e4cef59730>,
  <matplotlib.lines.Line2D at 0x1e4cef65bb0>,
  <matplotlib.lines.Line2D at 0x1e4cef65f10>],
 'boxes': [<matplotlib.lines.Line2D at 0x1e4cef4c970>,
  <matplotlib.lines.Line2D at 0x1e4cef65190>],
 'medians': [<matplotlib.lines.Line2D at 0x1e4cef59a90>,
  <matplotlib.lines.Line2D at 0x1e4cef712b0>],
 'fliers': [<matplotlib.lines.Line2D at 0x1e4cef59df0>,
  <matplotlib.lines.Line2D at 0x1e4cef71610>],
 'means': []}
```



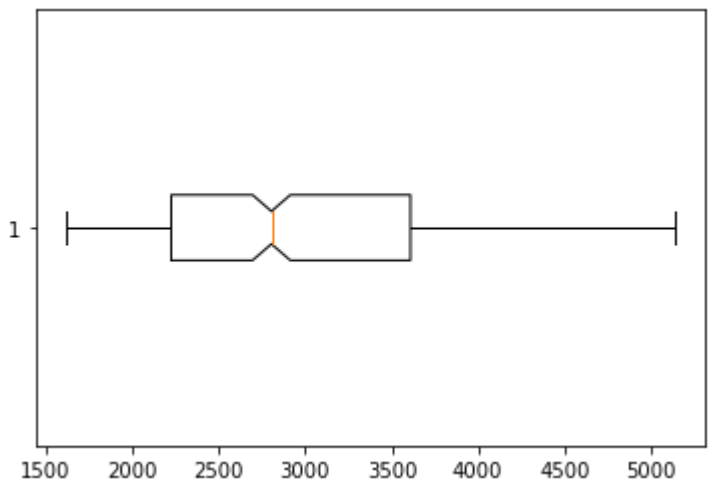
In [4]:
```
df = pd.read_csv('auto-mpg.csv')

def remove_outliers(df,col):
    q1 = df[col].quantile(0.25)
    q3 = df[col].quantile(0.75)
    iqr = q3-q1
    lower = q1-1.5*iqr
    upper  = q3+1.5*iqr
    return (df[(df[col]<lower)&(df[col]>upper)])
ndf = remove_outliers(df, 'mpg')
plt.boxplot([df['mpg'],ndf['mpg']])
plt.show()
```



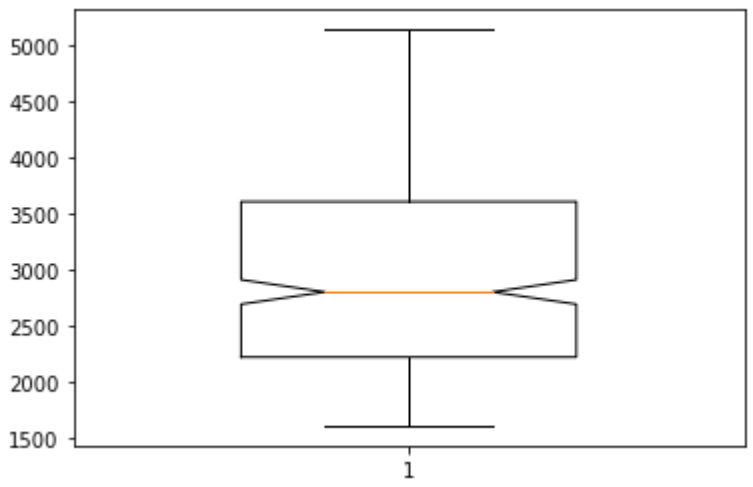
In [5]:
```
df = pd.read_csv('auto-mpg.csv')
plt.boxplot(df['weight'], notch=True)
plt.show()
```



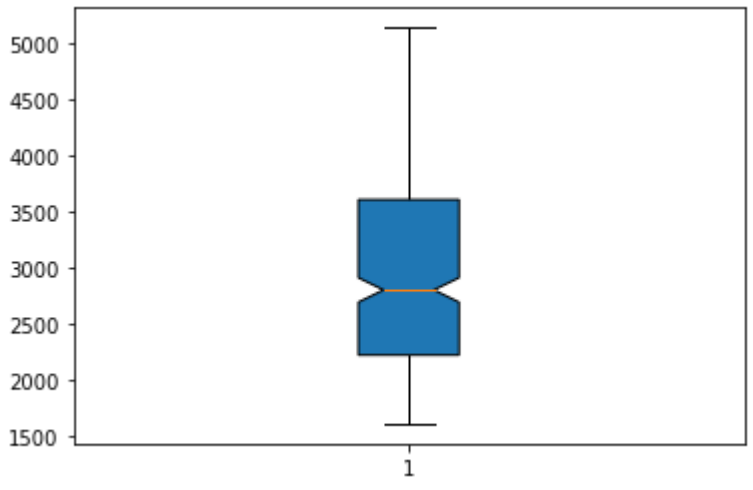
In [6]:
```
df = pd.read_csv('auto-mpg.csv')
plt.boxplot(df['weight'], notch=True, vert=False)
plt.show()
```



In [7]:
```
df = pd.read_csv('auto-mpg.csv')
plt.boxplot(df['weight'], notch=True, widths=0.5)
plt.show()
```



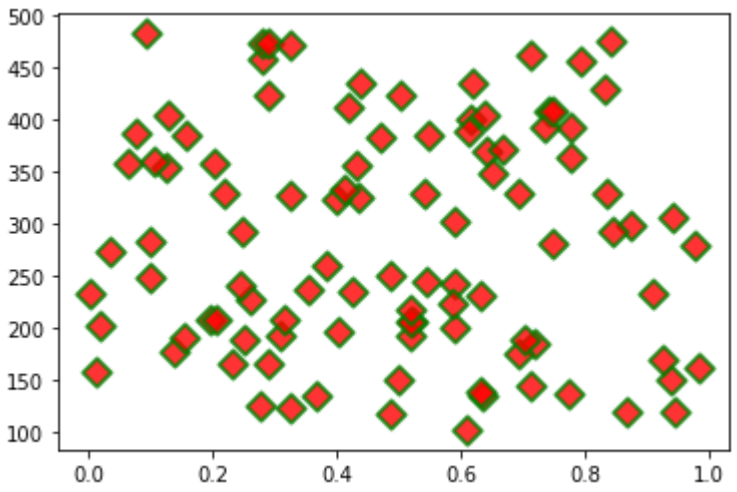
In [8]:
```
df = pd.read_csv('auto-mpg.csv')
plt.boxplot(df['weight'], notch=True, patch_artist=True)
plt.show()
```



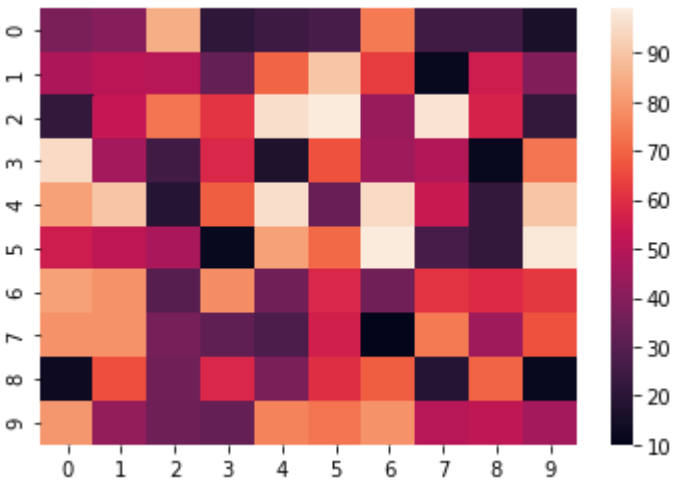
Scatter - relation between 2 var or 2 attribute

In [9]:
```
import numpy as np
x = np.random.rand(100)
y = np.random.randint(low=100,high=500,size=100)
plt.scatter(x,y,s=100,c='r',marker='D',edgecolor='g',linewidths=2,alpha=0.8)
```

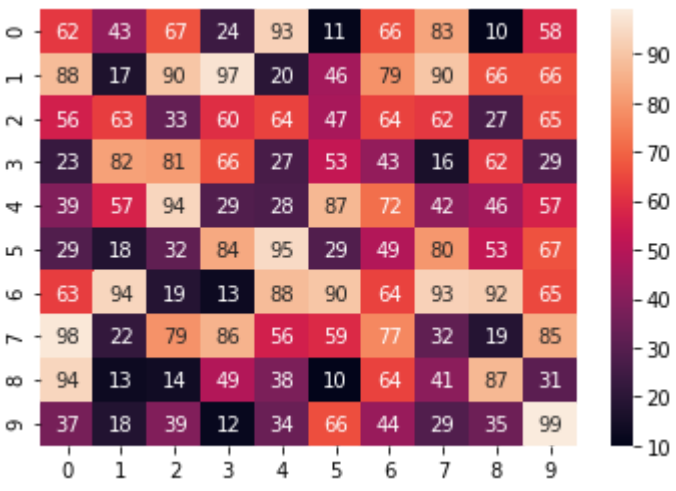
Out[9]: <matplotlib.collections.PathCollection at 0x1e4ceb14be0>



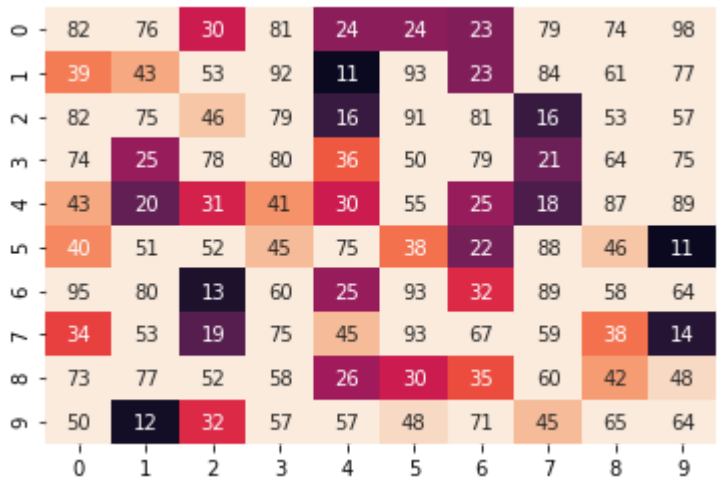
In [10]:
```
import seaborn as sn
import numpy as np
import matplotlib.pyplot as plt
data = np.random.randint(low=10,high=100,size=(10,10))
sn.heatmap(data)
plt.show()
```



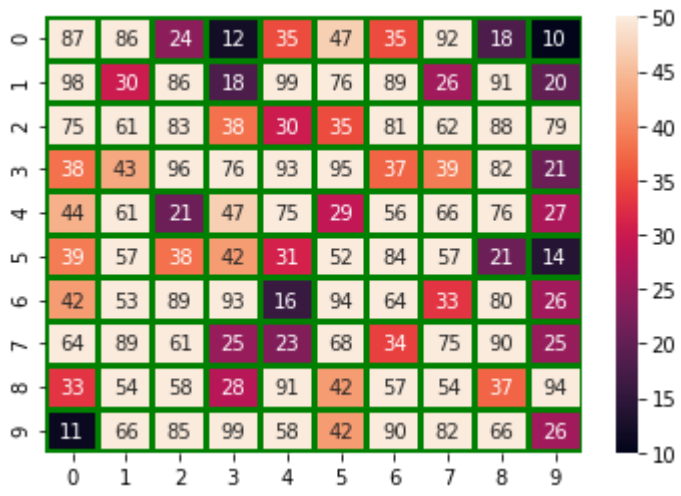
In [11]:
```
import seaborn as sn
import numpy as np
import matplotlib.pyplot as plt
data=np.random.randint(low=10,high=100,size=(10,10))
sn.heatmap(data,annot=True)#annot=Display number
plt.show()
```



In [12]:
```
import seaborn as sn
import numpy as np
import matplotlib.pyplot as plt
data=np.random.randint(low=10,high=100,size=(10,10))
sn.heatmap(data,annot=True,vmin=10,vmax=50,cbar=False)#cbar =False remove cbar
plt.show()
```

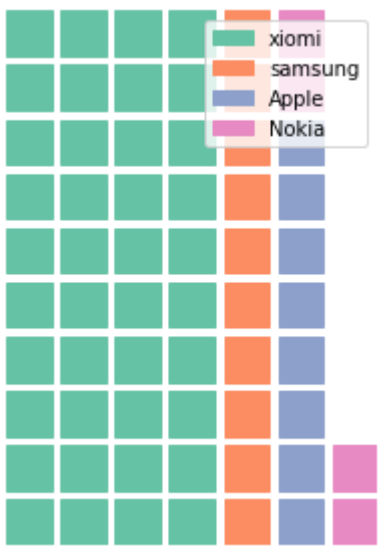


In [13]:
```
import seaborn as sn
import numpy as np
import matplotlib.pyplot as plt
data=np.random.randint(low=10,high=100,size=(10,10))
sn.heatmap(data,annot=True,vmin=10,vmax=50,linecolor='g',linewidth=2)
plt.show()
```



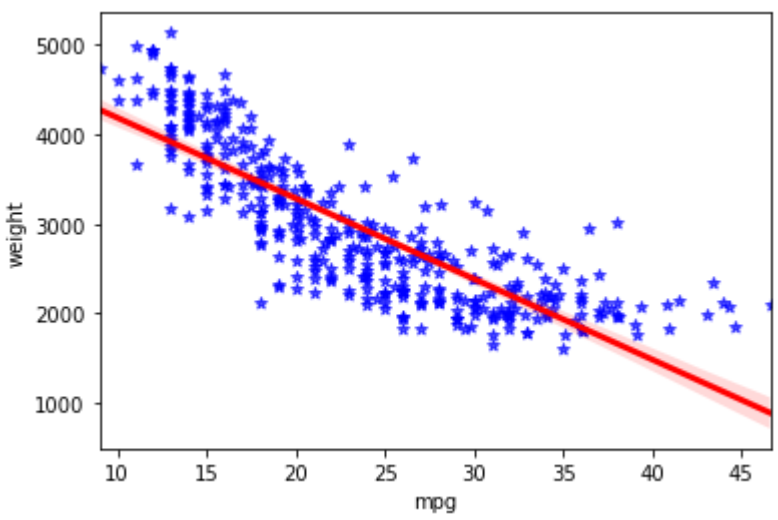
Wafflechart - alternate of piechat

In [18]:
```
import pandas as pd
from pywaffle import Waffle
import matplotlib.pyplot as plt

df=pd.read_csv('auto-mpg.csv')
df=pd.DataFrame(data)
data={'phone':['xiomi','samsung','Apple','Nokia'],'stock':[40,10,8,4]}
new_fig=plt.figure(FigureClass=Waffle,rows=10,values=df.stock,labels=list(df.phone))
```

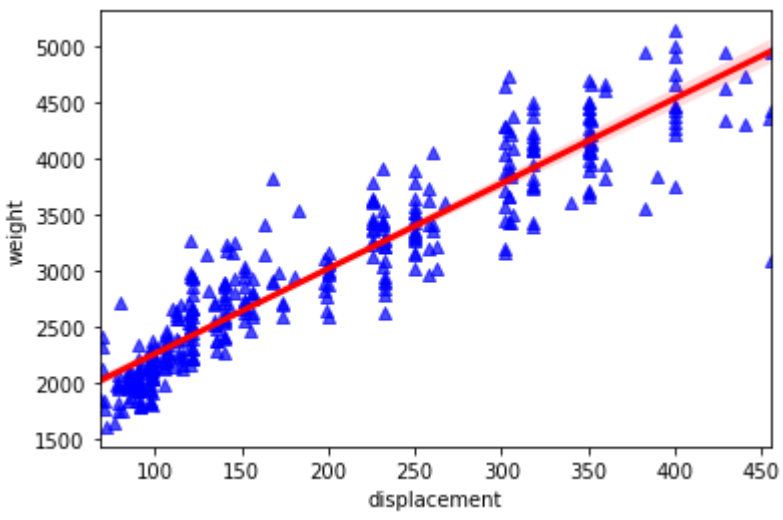


In [15]:
```
import pandas as pd
import matplotlib.pyplot as plt
from pywaffle import Waffle
data={'phone':['samsung','iphone','nokia','redmi','vivo','oppo','google','microsoft'],
'stock':[44,35,67,89,10,69,45,34]}
df=pd.DataFrame(data)
df.head() #It prints first five rows
fig=plt.figure(FigureClass=Waffle,rows=10,values=df.stock,labels=list(df.phone))
```

In [19]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
df=pd.read_csv('auto-mpg.csv')
sns.regplot(y=df['weight'],x=df['mpg'],marker='*',
            scatter_kws={'color':'blue','alpha':0.7},
            line_kws={'color':'red','linewidth':3})
plt.show()
```



In [20]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
df=pd.read_csv('auto-mpg.csv')
sns.regplot(y=df['weight'],x=df['displacement'],marker='^',
            scatter_kws={'color':'blue','alpha':0.7},
            line_kws={'color':'red','linewidth':3})
plt.show()
```



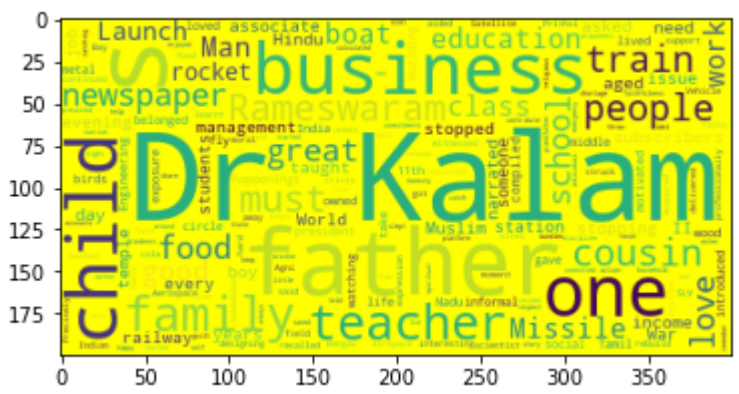
In [ ]:
```
pip install wordcloud
```

In [21]:
```
from wordcloud import WordCloud, STOPWORDS
stopwords = set(STOPWORDS)
print(stopwords)
# stopwords will not give any meaning it is only used for joining as suppose "the" we are using much so it
# it should be stopped and removed from respective textfile
```

```
{"how's", 'a', 'are', 'because', "wouldn't", 'over', 'yourselves', 'otherwise', 'himself', 'an', "they're", 'where', "m
ustn't", 'am', 'is', 'themselves', "when's", 'between', 'each', "that's", 'further', "you'll", 'also', "we'll", "they'l
l", 'who', 'some', "he'd", 'what', 'to', 'nor', 'all', "he'll", 'do', 'would', 'the', 'whom', 'than', "she'll", 'into',
'hers', 'had', 'ours', "hadn't", 'these', 'its', 'few', "we'd", 'again', 'but', 'theirs', 'this', 'herself', 'why', 'm
y', "haven't", 'on', "what's", "we've", 'only', 'out', "we're", "here's", 'has', "you're", 'there', "can't", 'doing',
'can', 'other', 'she', 'before', 'being', 'it', 'such', 'most', "she's", 'you', 'during', 'i', 'below', 'for', "were
n't", 'their', 'own', "there's", 'them', "i'm", "they'd", 'which', 'same', "i'll", 'above', 'however', 'that', 'http',
"shouldn't", 'with', 'him', 'those', 'yourself', 'hence', 'until', 'as', 'did', 'ever', 'from', 'while', "shan't", "are
n't", 'any', 'so', "doesn't", 'under', "don't", 'were', 'up', 'should', 'cannot', 'if', 'once', 'com', 'have', "wher
e's", 'he', 'www', "you'd", 'therefore', "wasn't", 'after', 'too', 'ought', "isn't", 'k', 'how', 'against', 'here', 'ge
t', "hasn't", 'not', 'else', "it's", 'at', 'of', 'itself', 'me', 'was', 'having', 'your', "why's", 'they', 'r', 'both',
'be', 'her', "couldn't", 'our', 'by', 'or', 'very', 'his', 'shall', 'just', "who's", 'and', 'off', 'yours', 'been', 'do
es', 'could', 'myself', 'then', 'when', 'we', "she'd", 'in', 'about', 'ourselves', 'since', "he's", "let's", "you've",
'more', 'no', "won't", 'down', "didn't", 'like', "i've", 'through', "i'd", "they've"}
```

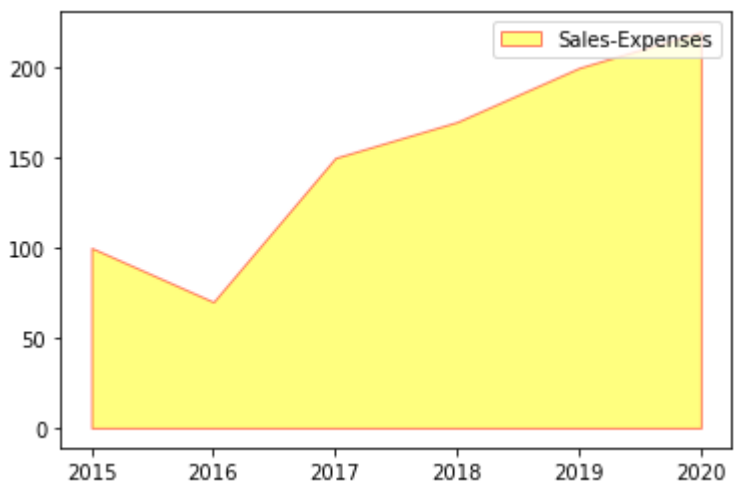
In [22]:
```
f = open('demo.txt','r', encoding='utf-8')
apj = f.read()
wc = WordCloud(background_color='yellow',max_words=200,stopwords=set(STOPWORDS))
#max_words to limit the words
wc.generate(apj)
plt.imshow(wc)
```

Out[22]: <matplotlib.image.AxesImage at 0x1e4d15cd340>

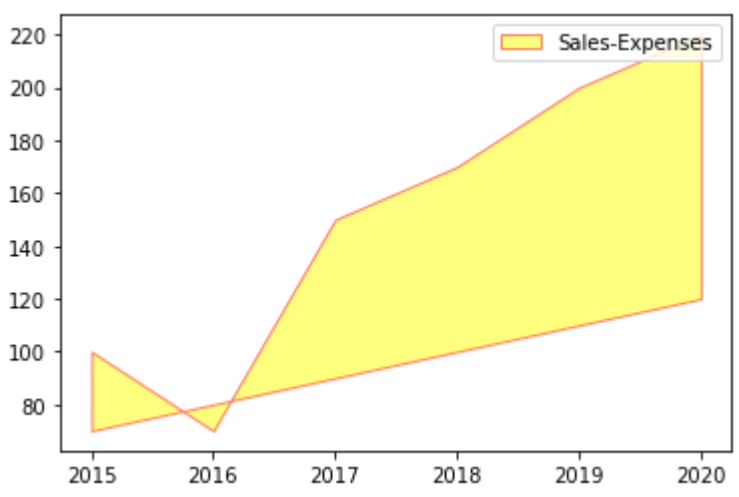


Area Plot (Area Charts)

In [23]:
```
import pandas as pd
import matplotlib.pyplot as plt

data = {'Year':[2015,2016,2017,2018,2019,2020],
        'Sales':[100,70,150,170,200,220],
        'Expenses':[70,80,90,100,110,120]}
df = pd.DataFrame(data)
plt.fill_between(df['Year'],df['Sales'],
                 facecolor='yellow',
                 edgecolor='red',
                 alpha=0.5,
                 label='Sales-Expenses')

plt.legend()
plt.show()
```

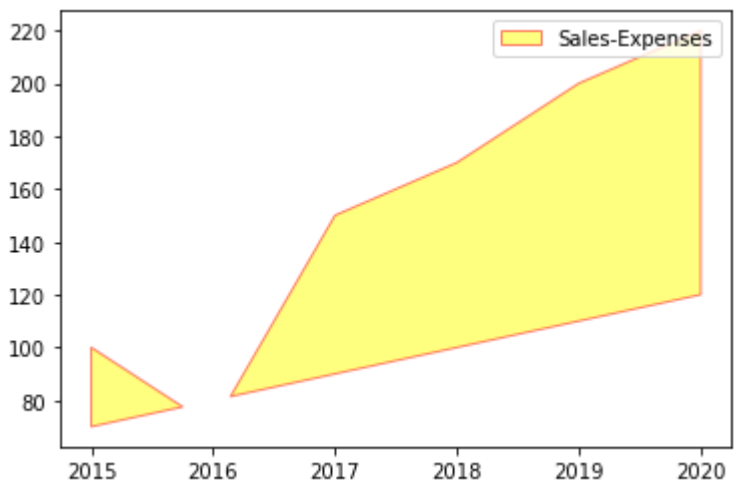


In [25]:
```
import pandas as pd
import matplotlib.pyplot as plt

data = {'Year':[2015,2016,2017,2018,2019,2020],
       'Sales':[100,70,150,170,200,220],
       'Expenses':[70,80,90,100,110,120]}
df = pd.DataFrame(data)
plt.fill_between(df['Year'],df['Sales'],df['Expenses'],
                 facecolor='yellow',
                 edgecolor='red',
                 alpha=0.5,
                 label='Sales-Expenses')

plt.legend()
plt.show()
```

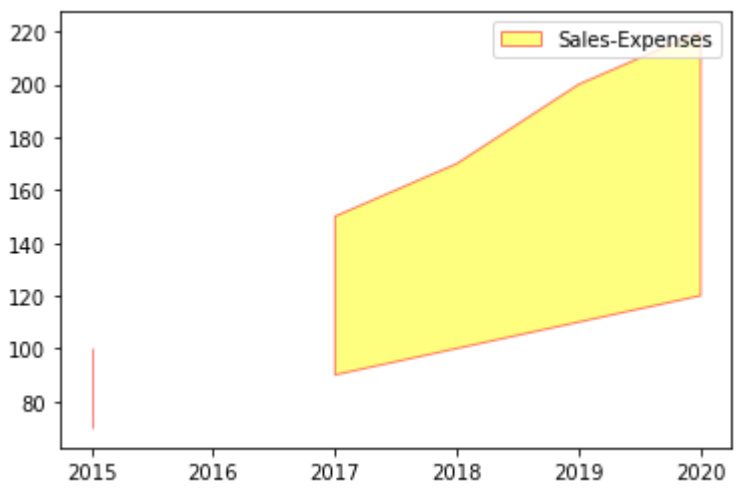


- **interpolate - to indentify intersection**

In [26]:
```
import pandas as pd
import matplotlib.pyplot as plt

data = {'Year':[2015,2016,2017,2018,2019,2020],
       'Sales':[100,70,150,170,200,220],
       'Expenses':[70,80,90,100,110,120]}
df = pd.DataFrame(data)
plt.fill_between(df['Year'],df['Sales'],df['Expenses'],
                  where=df['Sales']>df['Expenses'],interpolate=True,
                 facecolor='yellow',
                 edgecolor='red',
                 alpha=0.5,
                 label='Sales-Expenses')

plt.legend()
plt.show()
```

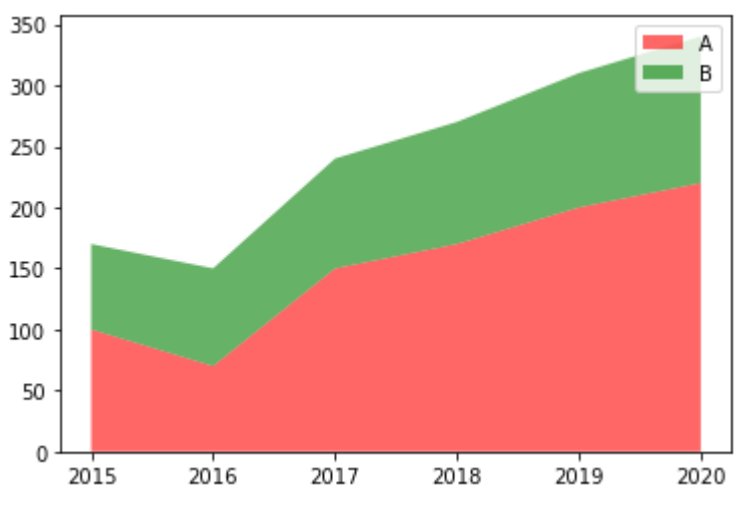


In [27]:
```
import pandas as pd
import matplotlib.pyplot as plt

data = {'Year':[2015,2016,2017,2018,2019,2020],
       'Sales':[100,70,150,170,200,220],
       'Expenses':[70,80,90,100,110,120]}
df = pd.DataFrame(data)
plt.fill_between(df['Year'],df['Sales'],df['Expenses'],
                 where=df['Sales']>df['Expenses'],
                facecolor='yellow',
                edgecolor='red',
                alpha=0.5,
                label='Sales-Expenses')

plt.legend()
plt.show()
```

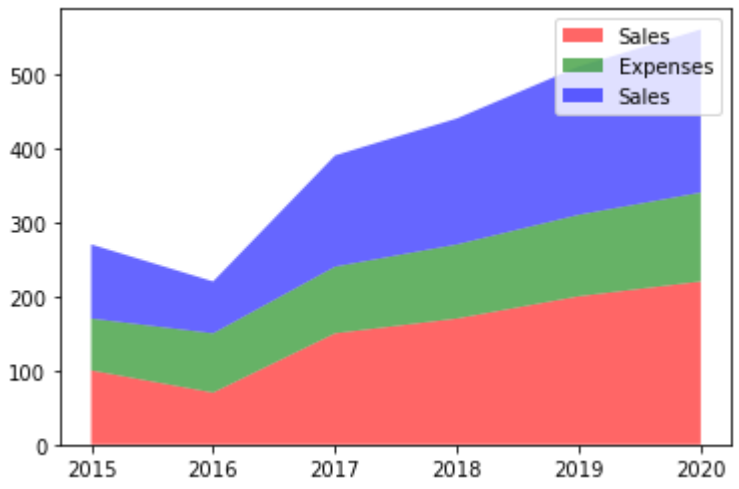


Stackplot - A variation of Area Plot

In [28]:
```
data = {'Year':[2015,2016,2017,2018,2019,2020],
       'Sales':[100,70,150,170,200,220],
       'Expenses':[70,80,90,100,110,120]}
df = pd.DataFrame(data)
plt.stackplot(df['Year'],df['Sales'],df['Expenses'],
                labels=['A','B'],
                colors=['r','g'],
                alpha=0.6)

plt.legend()
plt.show()
```



In [29]:
```
data = {'Year':[2015,2016,2017,2018,2019,2020],
        'Sales':[100,70,150,170,200,220],
        'Expenses':[70,80,90,100,110,120]}
df = pd.DataFrame(data)
plt.stackplot(df['Year'],df['Sales'],df['Expenses'],df['Sales'],
                labels=['Sales','Expenses','Sales'],
                colors=['r','g','b'],
                alpha=0.6)

plt.legend()
plt.show()
```

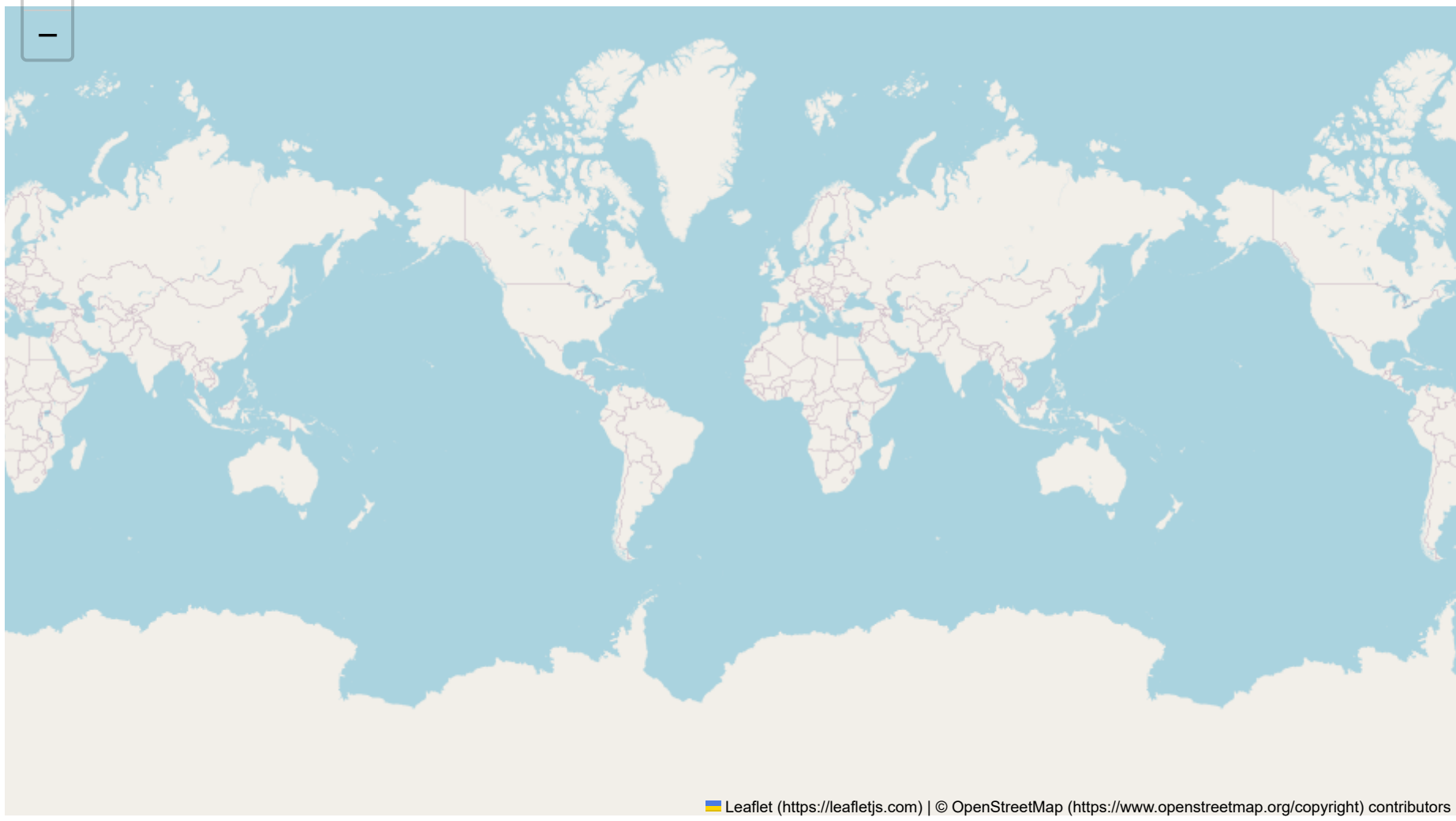


In [ ]:
```
pip install folium
```

Geospatial data : latitude and longitudinal

In [30]:
```
import folium
world_map = folium.Map()
world_map
```

Out[30]:
Make this Notebook Trusted to load map: File -> Trust Notebook



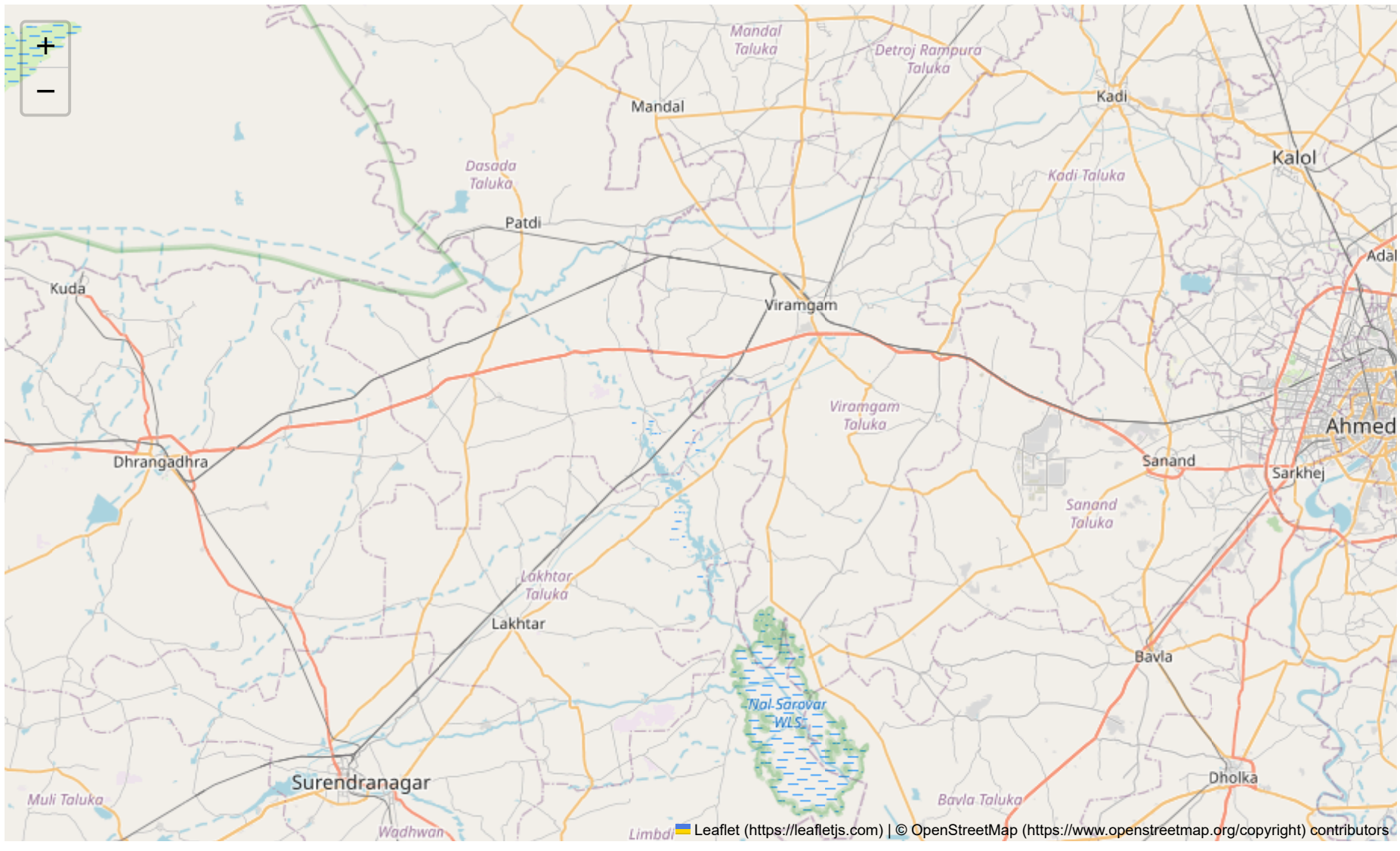
In [31]:
```
import folium
world_map=folium.Map(location=[20.5937,78.9629],zoom_start=10)
#North positive value, South positive value, east positive value, west negative value
world_map
```

Out[31]:

In [36]:
```
import folium
mapObj=folium.Map(location=[23,72],zoom_start=10)
folium.Circle(location=[23,72],radius=50000).add_to(mapObj)
mapObj.save('Output.html')
```

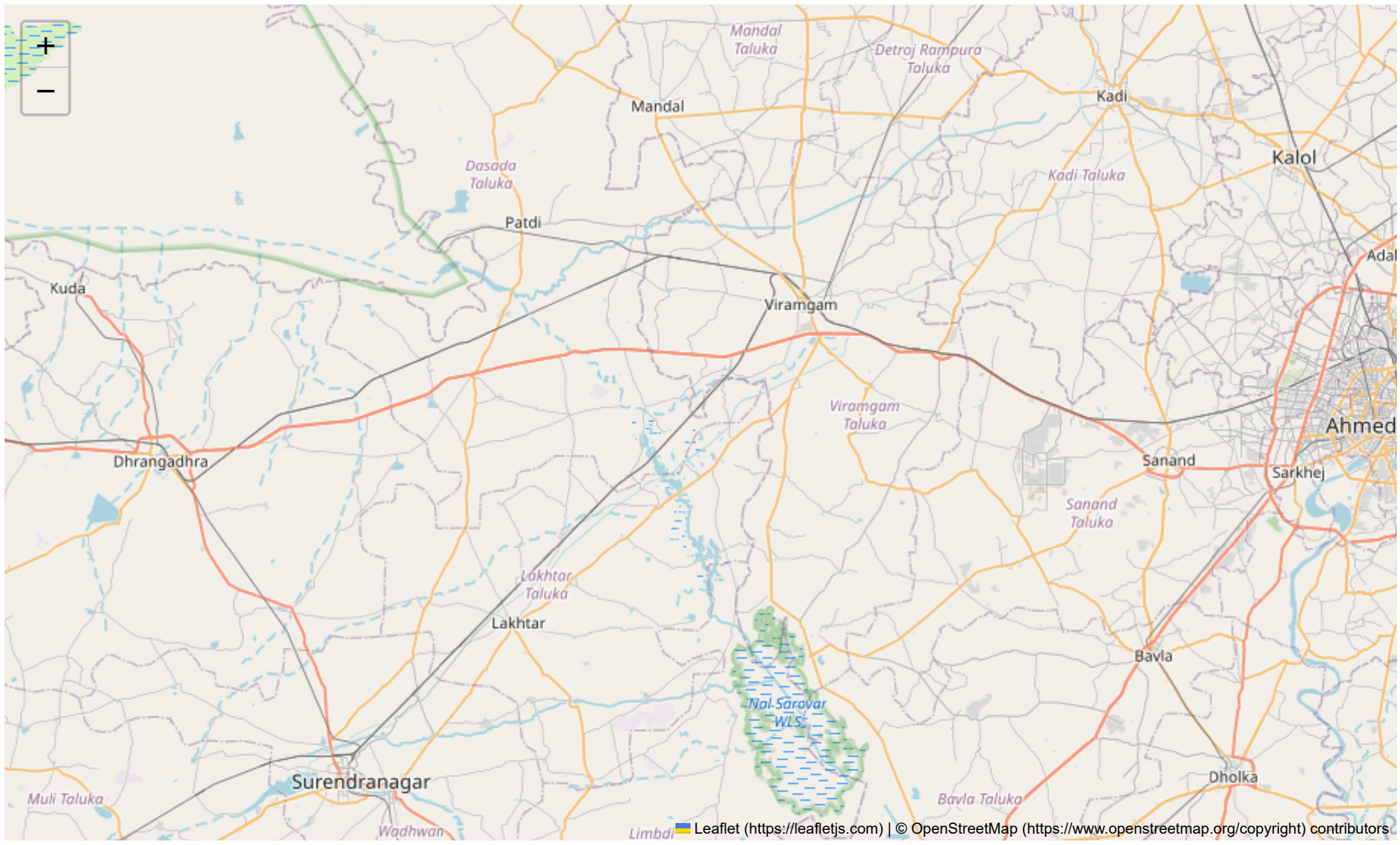
In [40]:
```
import folium
mapObj=folium.Map(location=[23,72],zoom_start=10)
folium.Circle(location=[23,72],radius=50000,color='green',weight=6,fill_color='red',
              opacity=0.1)
mapObj
```

Out[40]:



In [41]:
```
import folium
mapObj=folium.Map(location=[23,72],zoom_start=10)
folium.Circle(location=[23,72],radius=50000,tooltip="This is tooltip text",
              popup=folium.Popup("""<h1>Hiie</h1>"""))
mapObj
```

Out[41]:

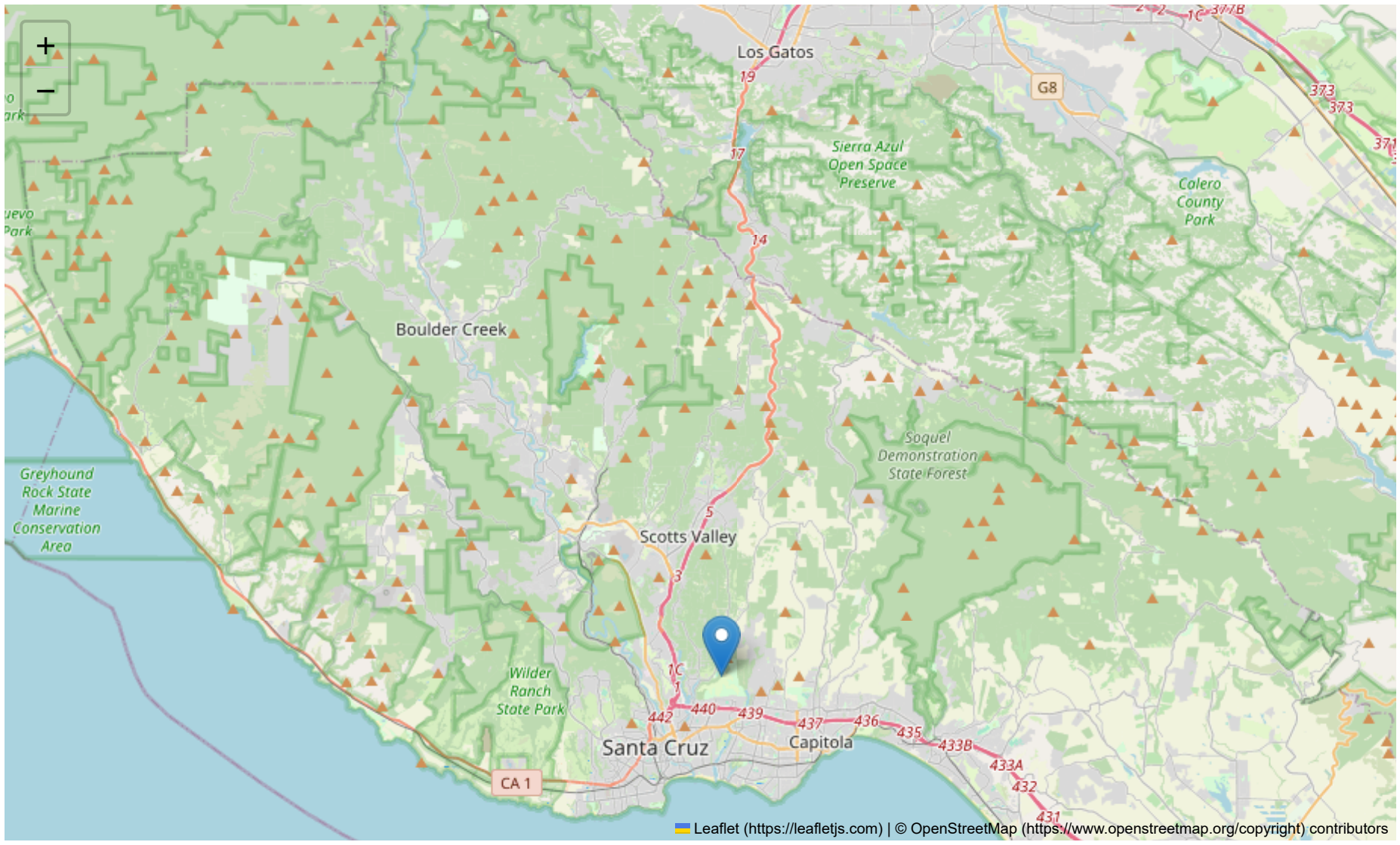


In [43]:
```
import folium
mapObj=folium.Map(location=[23,72],zoom_start=10)
folium.Circle(location=[23,72],radius=50000,tooltip="This is tooltip text",
              popup=folium.Popup("""<h1>Hiie</h1>"""))
folium.LayerControl().add_to(mapObj)
mapObj.save('output.html')
```

In [44]:
```
import folium
mapObj=folium.Map(location=[23,72],zoom_start=10)
folium.CircleMarker(location=[23,72],radius=50).add_to(mapObj)
mapObj.save('Output4.html')
```

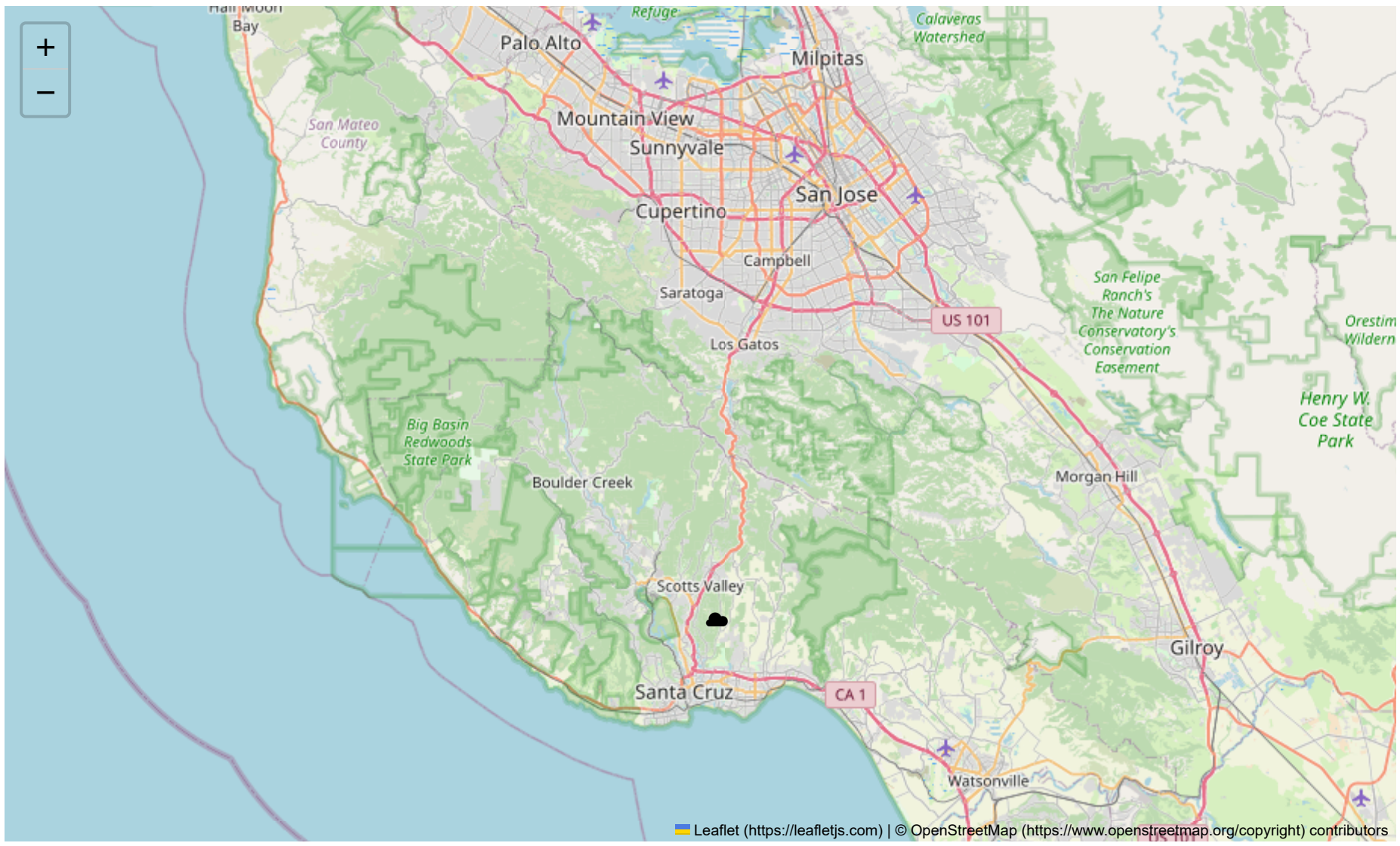
In [45]:
```
import folium
mapObj = folium.Map(location=[37,-122])
icon=folium.features.Icon(icon='cloud')
marker=folium.Marker(location=[37,-122])
marker.add_to(mapObj)
mapObj
```

Out[45]:



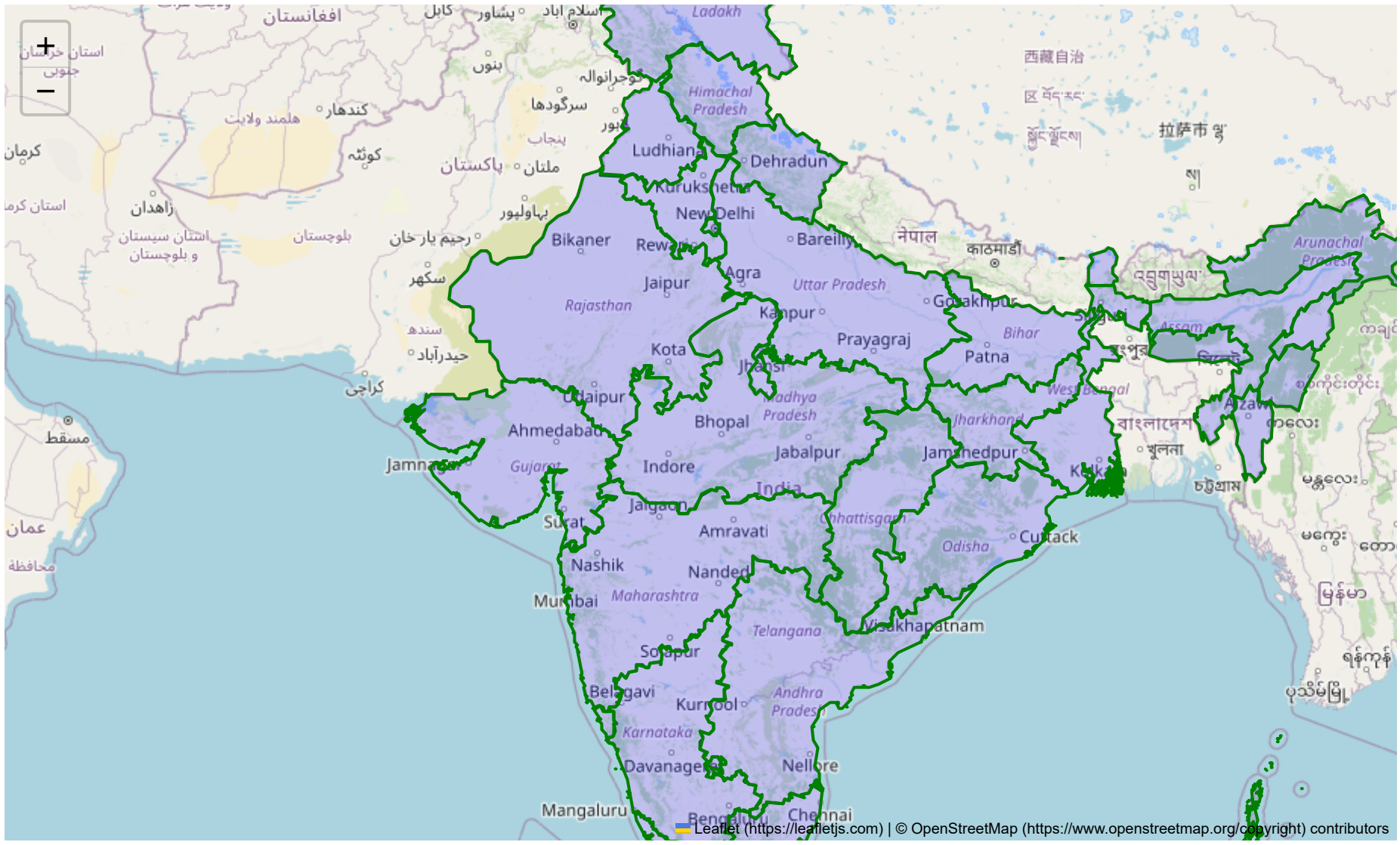
In [46]:
```
import folium
mapObj = folium.Map(location=[37,-122])
icon=folium.features.Icon(icon='cloud')
marker=folium.Marker(location=[37,-122],icon=icon)
marker.add_to(mapObj)
mapObj
```

Out[46]:



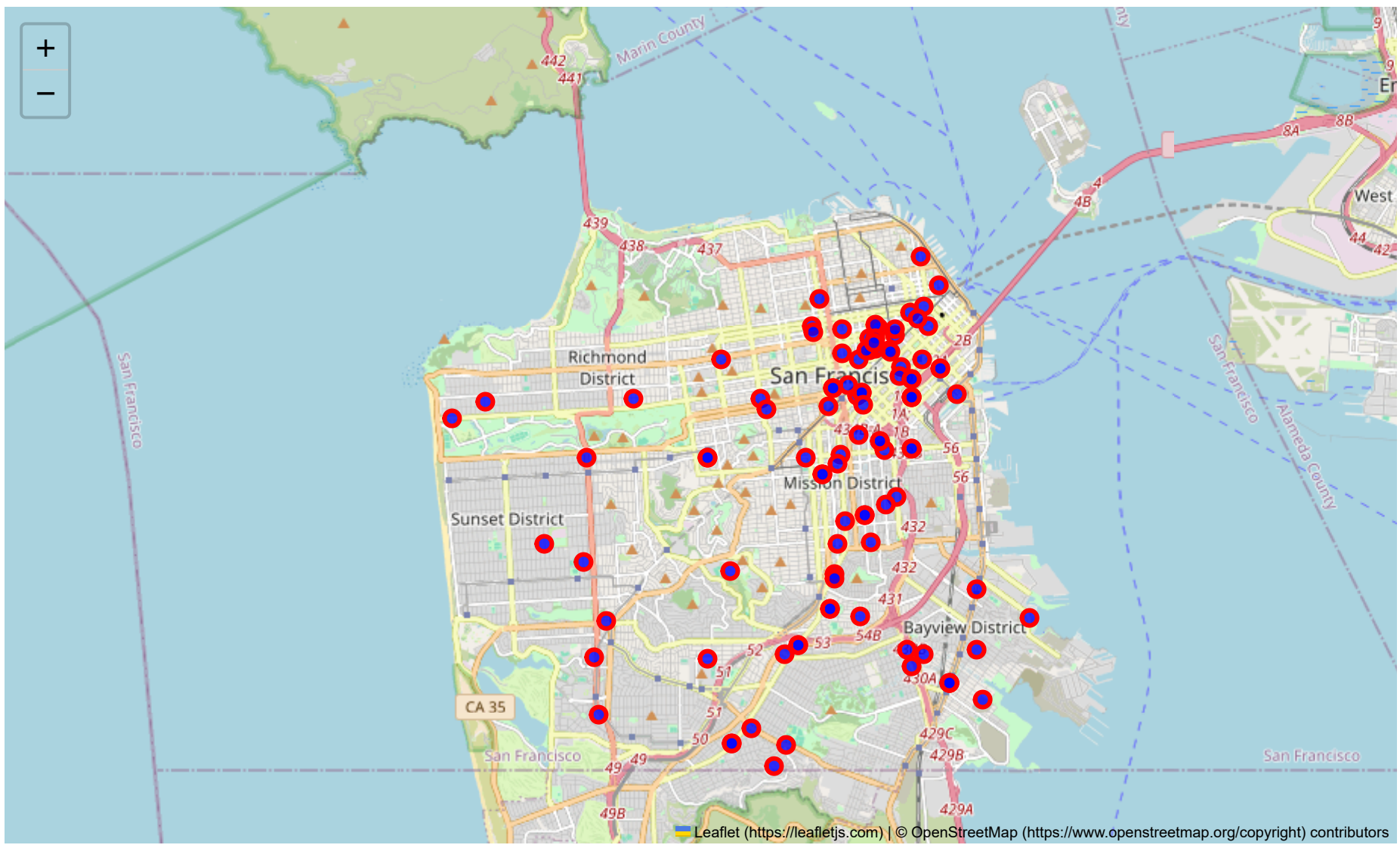
In [50]:
```
import folium
mob = folium.Map(location=[23,72])
folium.GeoJson(data=(open("Indian_States",'r').read()),
               name="India", color='green', weight=2,
               fillColor='blue', fillOpacity=0.2).add_to(mob)
mob
```

Out[50]:



In [53]:
```
df = pd.read_csv('Datasets/Police_Department_Incidents_-_Previous_Year__2016_.csv')
df = df.iloc[0:100,:]
mob = folium.Map(location=[37,-122])
incidents = folium.map.FeatureGroup()
for lat,lon in zip(df.Y,df.X):
    incidents.add_child(folium.features.CircleMarker([lat,lon],
        radius=5,color='red',
        fill=True,fill_color='blue',
        fill_opacity=0.8,))
mob.add_child(incidents)

#zip can be done using df.X and df.Y if both have some number of datas to create a child node
```

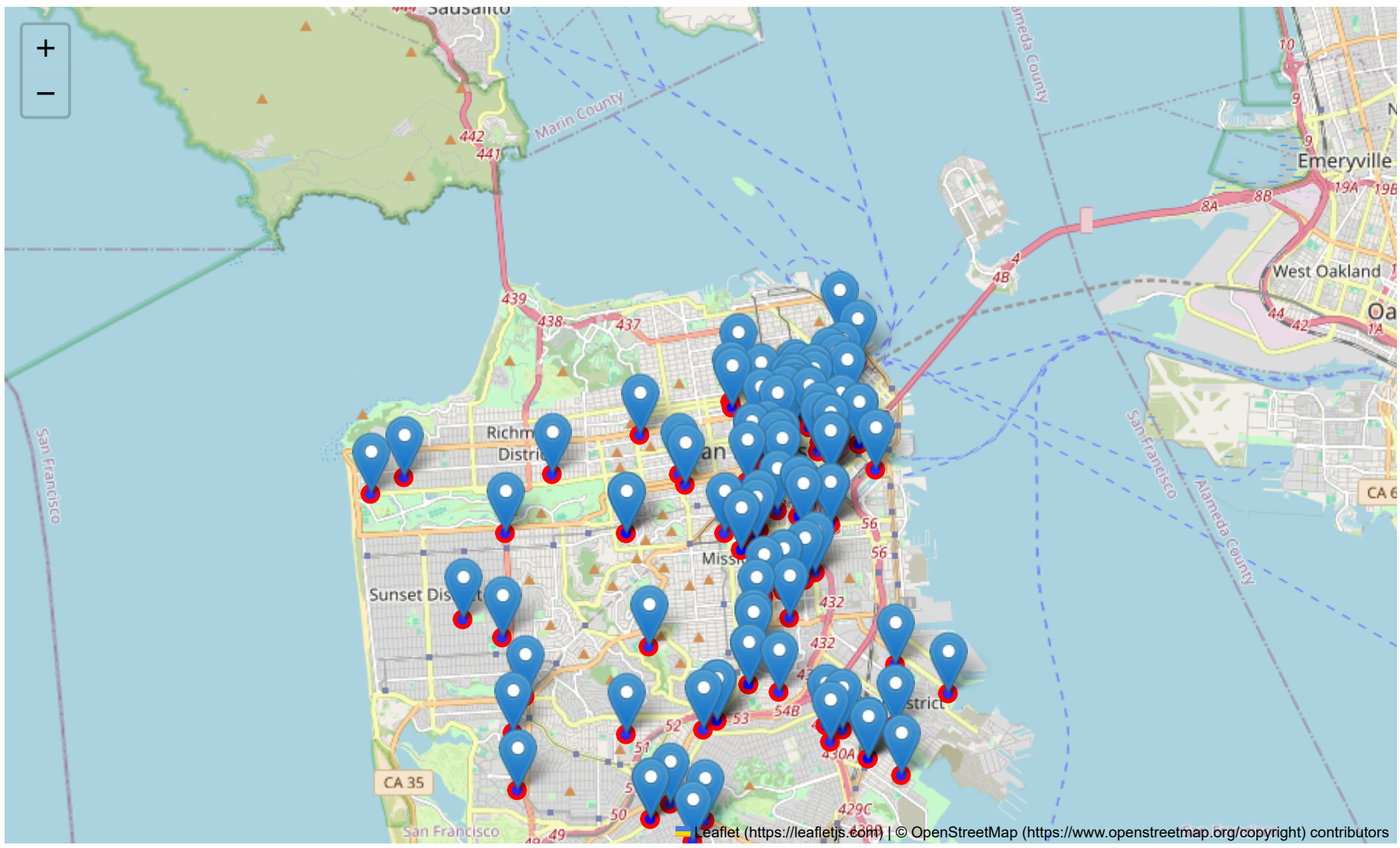
Out[53]:



In [57]:
```
#police file for san fransico
import folium
import pandas as pd
df_incidents=pd.read_csv('Datasets/Police_Department_Incidents_-_Previous_Year__2016_.csv')
print(df_incidents.head())
df_incidents.shape
df=df_incidents.iloc[0:100,:] #passed rows (0 to 100) and columns (all 13)
df.shape
sanfran_map=folium.Map(location=[37.77,-122.44],zoom_start=12)
incidents=folium.map.FeatureGroup()
for lat,lon in zip(df.Y,df.X):
    incidents.add_child(folium.features.CircleMarker([lat,lon],
    radius=5,color='red',
    fill=True,fill_color='blue',
    fill_opacity=0.8,))
for lat,lon,label in zip(df.Y,df.X,df.Category):
    folium.Marker([lat,lon],popup=label).add_to(sanfran_map)
#to show circle with filled color
#iloc in folium for limiting data
sanfran_map.add_child(incidents)
```

```
   IncidntNum      Category                                   Descript  \
0   120058272   WEAPON LAWS                     POSS OF PROHIBITED WEAPON
1   120058272   WEAPON LAWS  FIREARM, LOADED, IN VEHICLE, POSSESSION OR USE
2   141059263      WARRANTS                            WARRANT ARREST
3   160013662  NON-CRIMINAL                             LOST PROPERTY
4   160002740  NON-CRIMINAL                             LOST PROPERTY

  DayOfWeek                Date   Time  PdDistrict      Resolution  \
0    Friday  01/29/2016 12:00:00 AM  11:00    SOUTHERN  ARREST, BOOKED
1    Friday  01/29/2016 12:00:00 AM  11:00    SOUTHERN  ARREST, BOOKED
2    Monday  04/25/2016 12:00:00 AM  14:59     BAYVIEW  ARREST, BOOKED
3   Tuesday  01/05/2016 12:00:00 AM  23:50  TENDERLOIN            NONE
4    Friday  01/01/2016 12:00:00 AM  00:30     MISSION            NONE

                  Address           X          Y  \
0  800 Block of BRYANT ST -122.403405  37.775421
1  800 Block of BRYANT ST -122.403405  37.775421
2   KEITH ST / SHAFTER AV -122.388856  37.729981
3  JONES ST / OFARRELL ST -122.412971  37.785788
4    16TH ST / MISSION ST -122.419672  37.765050

                            Location           PdId
0   (37.775420706711, -122.403404791479)  12005827212120
1   (37.775420706711, -122.403404791479)  12005827212168
2  (37.7299809672996, -122.388856204292)  14105926363010
3  (37.7857883766888, -122.412970537591)  16001366271000
4  (37.7650501214668, -122.419671780296)  16000274071000
```

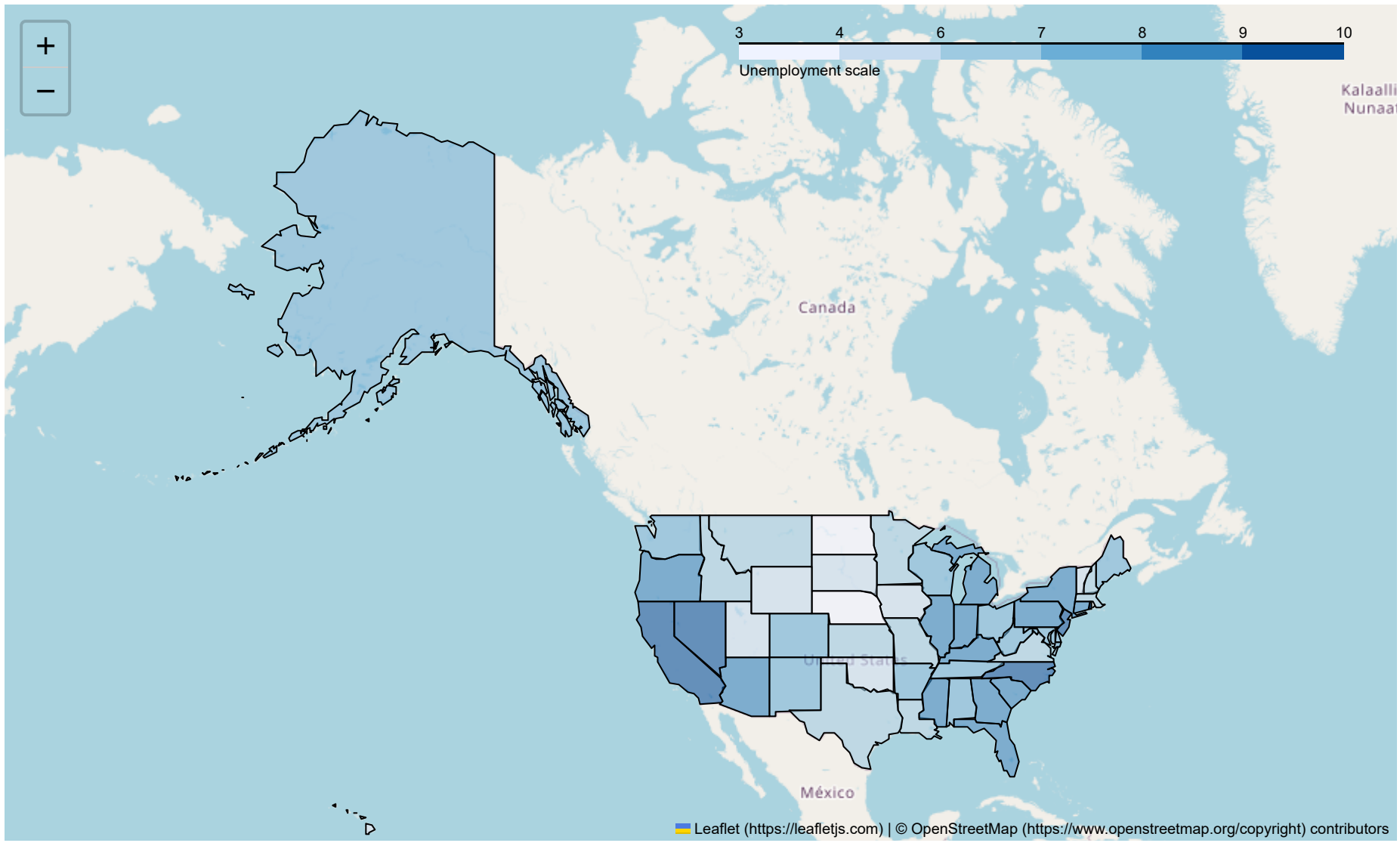
Out[57]:



Choropleth Maps : to make such type of heat maps in folium (world map)

In [65]:
```
import folium
import pandas as pd
state_unemp=pd.read_csv('Datasets/US_Unemployment_Oct2012.csv')
state_geo='us-states.json'
m=folium.Map(location=[48,-102])
folium.Choropleth(geo_data=state_geo,name='choropleth',data=state_unemp,
columns=['State','Unemployment'],key_on='feature.id',
legend_name='Unemployment scale').add_to(m)
m
```

Out[65]:



In [ ]:
```

```