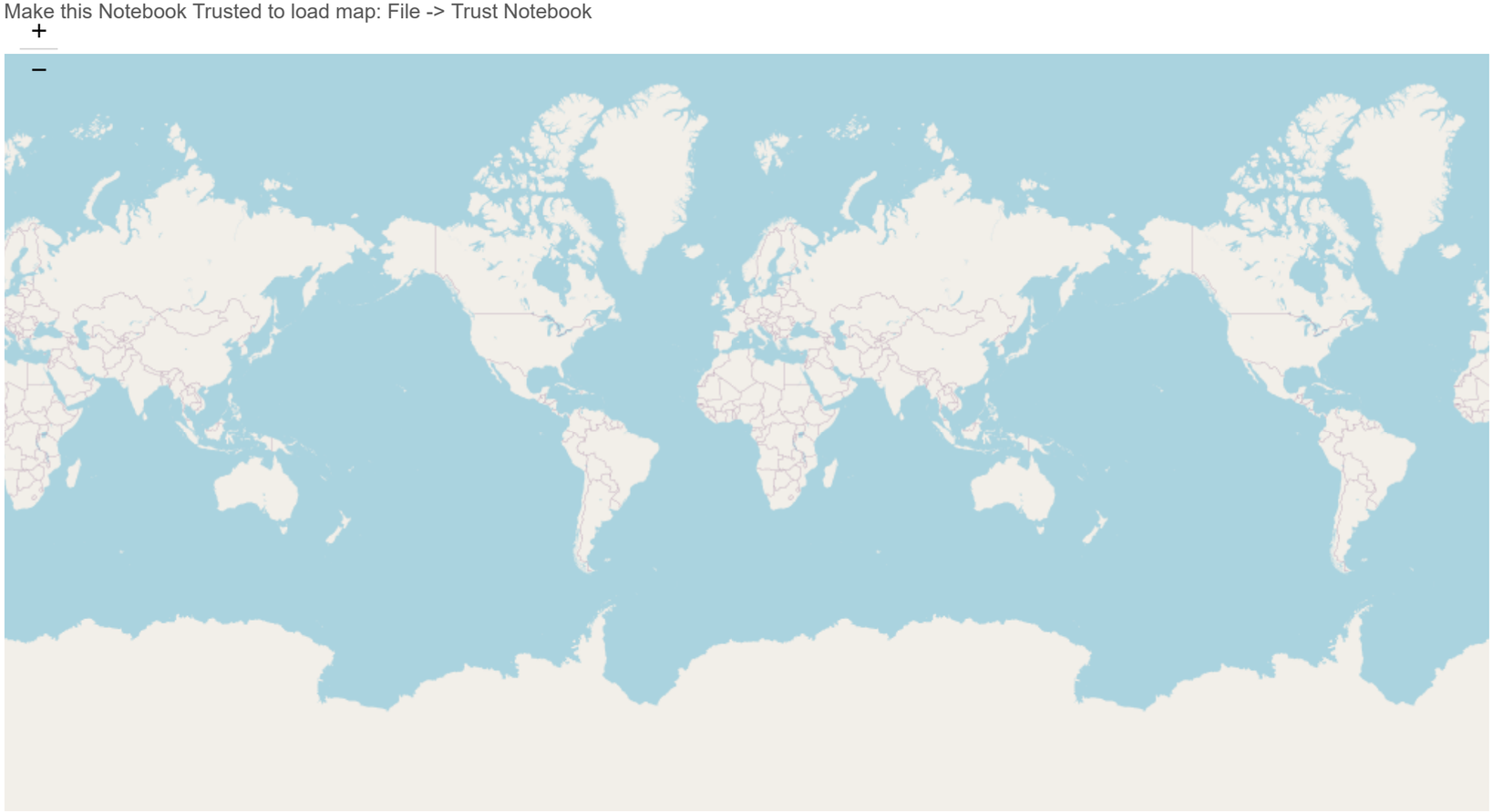
In [1]:
```
import folium
```

In [2]:
```
world_map=folium.Map(zoom_start=1,tile='OpenStreetMap')
world_map
```

Out[2]: Make this Notebook Trusted to load map: File -> Trust Notebook



In [3]:
```
bghm_map=folium.Map(location=[55.3781,-3.4360],zoom_start=5.2,tiles='Stamen Terrain')
bghm_map
```

Out[3]:

In [4]:
```
birmingham=folium.map.FeatureGroup()
birmingham.add_child(folium.features.CircleMarker([52.4862, -1.8904],
                                                  radius=9,color='red',
                                                  fill_color='green'))
bghm_map.add_child(birmingham)
```

Out[4]:

In [5]:
```
folium.Marker([52.4862, -1.8904],popup='birmingham').add_to(bghm_map)
```

Out[5]:
```
<folium.map.Marker at 0x190a4b5bd00>
```

In [6]:
```
import pandas as pd
df_can = pd.read_excel(
    'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-SkillsNetwork/Data%20Files/Canada.xlsx',
    sheet_name='Canada by Citizenship',
    skiprows=range(20),
    skipfooter=2)
print('Data downloaded and read into a dataframe!')
```

```
Data downloaded and read into a dataframe!
```

In [7]:
```
df_can=df_can.iloc[:,:43]
df_can.head()
```

Out[7]:

	Type	Coverage	OdName	AREA	AreaName	REG	RegName	DEV	DevName	1980	...	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013
0	Immigrants	Foreigners	Afghanistan	935	Asia	5501	Southern Asia	902	Developing regions	16	...	2978	3436	3009	2652	2111	1746	1758	2203	2635	2004
1	Immigrants	Foreigners	Albania	908	Europe	925	Southern Europe	901	Developed regions	1	...	1450	1223	856	702	560	716	561	539	620	603
2	Immigrants	Foreigners	Algeria	903	Africa	912	Northern Africa	902	Developing regions	80	...	3616	3626	4807	3623	4005	5393	4752	4325	3774	4331
3	Immigrants	Foreigners	American Samoa	909	Oceania	957	Polynesia	902	Developing regions	0	...	0	0	1	0	0	0	0	0	0	0
4	Immigrants	Foreigners	Andorra	908	Europe	925	Southern Europe	901	Developed regions	0	...	0	0	1	1	0	0	0	0	1	1

5 rows × 43 columns

In [8]:
```
# clean up the dataset to remove unnecessary columns (eg. REG)
df_can.drop(['AREA','REG','DEV','Type','Coverage'], axis=1, inplace=True)

# let's rename the columns so that they make sense
df_can.rename(columns={'OdName':'Country', 'AreaName':'Continent','RegName':'Region'}, inplace=True)

# for sake of consistency, let's also make all column labels of type string
df_can.columns = list(map(str, df_can.columns))

# add total column
df_can['Total'] = df_can.sum(axis=1)

# years that we will be using in this lesson - useful for plotting later on
years = list(map(str, range(1980, 2014)))
print ('data dimensions:', df_can.shape)
```

```
data dimensions: (195, 39)
```

In [9]:
```
df_can.head()
```

Out[9]:

	Country	Continent	Region	DevName	1980	1981	1982	1983	1984	1985	...	2005	2006	2007	2008	2009	2010	2011	2012	2013	Total
0	Afghanistan	Asia	Southern Asia	Developing regions	16	39	39	47	71	340	...	3436	3009	2652	2111	1746	1758	2203	2635	2004	58639
1	Albania	Europe	Southern Europe	Developed regions	1	0	0	0	0	0	...	1223	856	702	560	716	561	539	620	603	15699
2	Algeria	Africa	Northern Africa	Developing regions	80	67	71	69	63	44	...	3626	4807	3623	4005	5393	4752	4325	3774	4331	69439
3	American Samoa	Oceania	Polynesia	Developing regions	0	1	0	0	0	0	...	0	1	0	0	0	0	0	0	0	6
4	Andorra	Europe	Southern Europe	Developed regions	0	0	0	0	0	0	...	0	1	1	0	0	0	0	1	1	15

5 rows × 39 columns

In [10]:
```
world_map = folium.Map(location=[0, 0], zoom_start=2)
```

In [11]:
```
folium.Map(location=[40.4637, 3.7492], zoom_start=6, tiles='Stamen Terrain')
```

Out[11]:

In [12]:
```
URL = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-SkillsNetwork/Data%20Files/Police_Department_Incidents_-_Previous_Year__
df_incidents =  pd.read_csv(URL)
df_incidents.head()
```

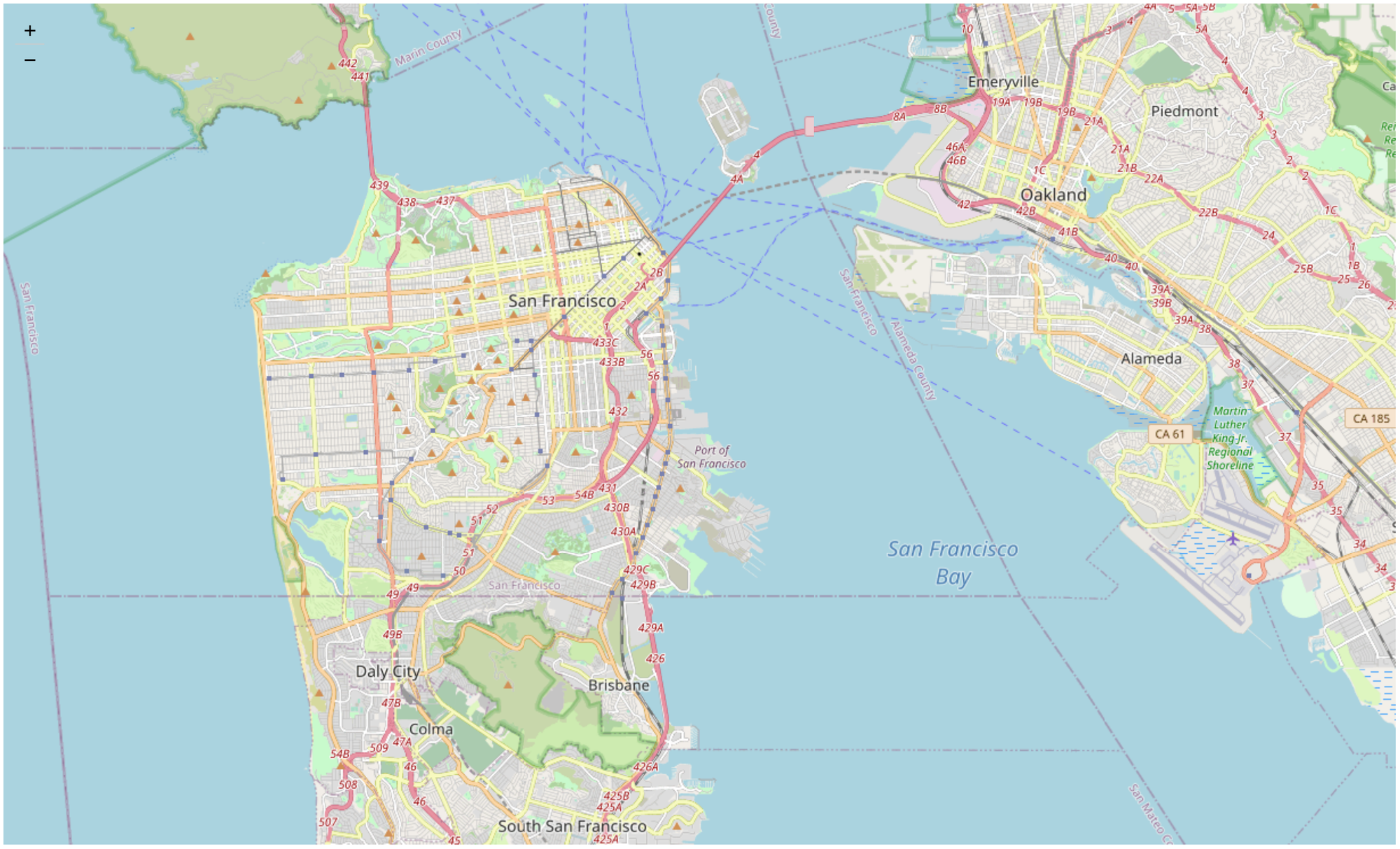
Out[12]:

	IncidntNum	Category	Descript	DayOfWeek	Date	Time	PdDistrict	Resolution	Address	X	Y	Location	PdId
0	120058272	WEAPON LAWS	POSS OF PROHIBITED WEAPON	Friday	01/29/2016 12:00:00 AM	11:00	SOUTHERN	ARREST, BOOKED	800 Block of BRYANT ST	-122.403405	37.775421	(37.775420706711, -122.403404791479)	12005827212120
1	120058272	WEAPON LAWS	FIREARM, LOADED, IN VEHICLE, POSSESSION OR USE	Friday	01/29/2016 12:00:00 AM	11:00	SOUTHERN	ARREST, BOOKED	800 Block of BRYANT ST	-122.403405	37.775421	(37.775420706711, -122.403404791479)	12005827212168
2	141059263	WARRANTS	WARRANT ARREST	Monday	04/25/2016 12:00:00 AM	14:59	BAYVIEW	ARREST, BOOKED	KEITH ST / SHAFTER AV	-122.388856	37.729981	(37.7299809672996, -122.388856204292)	14105926363010
3	160013662	NON-CRIMINAL	LOST PROPERTY	Tuesday	01/05/2016 12:00:00 AM	23:50	TENDERLOIN	NONE	JONES ST / OFARRELL ST	-122.412971	37.785788	(37.7857883766888, -122.412970537591)	16001366271000
4	160002740	NON-CRIMINAL	LOST PROPERTY	Friday	01/01/2016 12:00:00 AM	00:30	MISSION	NONE	16TH ST / MISSION ST	-122.419672	37.765050	(37.7650501214668, -122.419671780296)	16000274071000

In [13]:
```
limit = 100
df_incidents = df_incidents.iloc[0:limit, :]
```

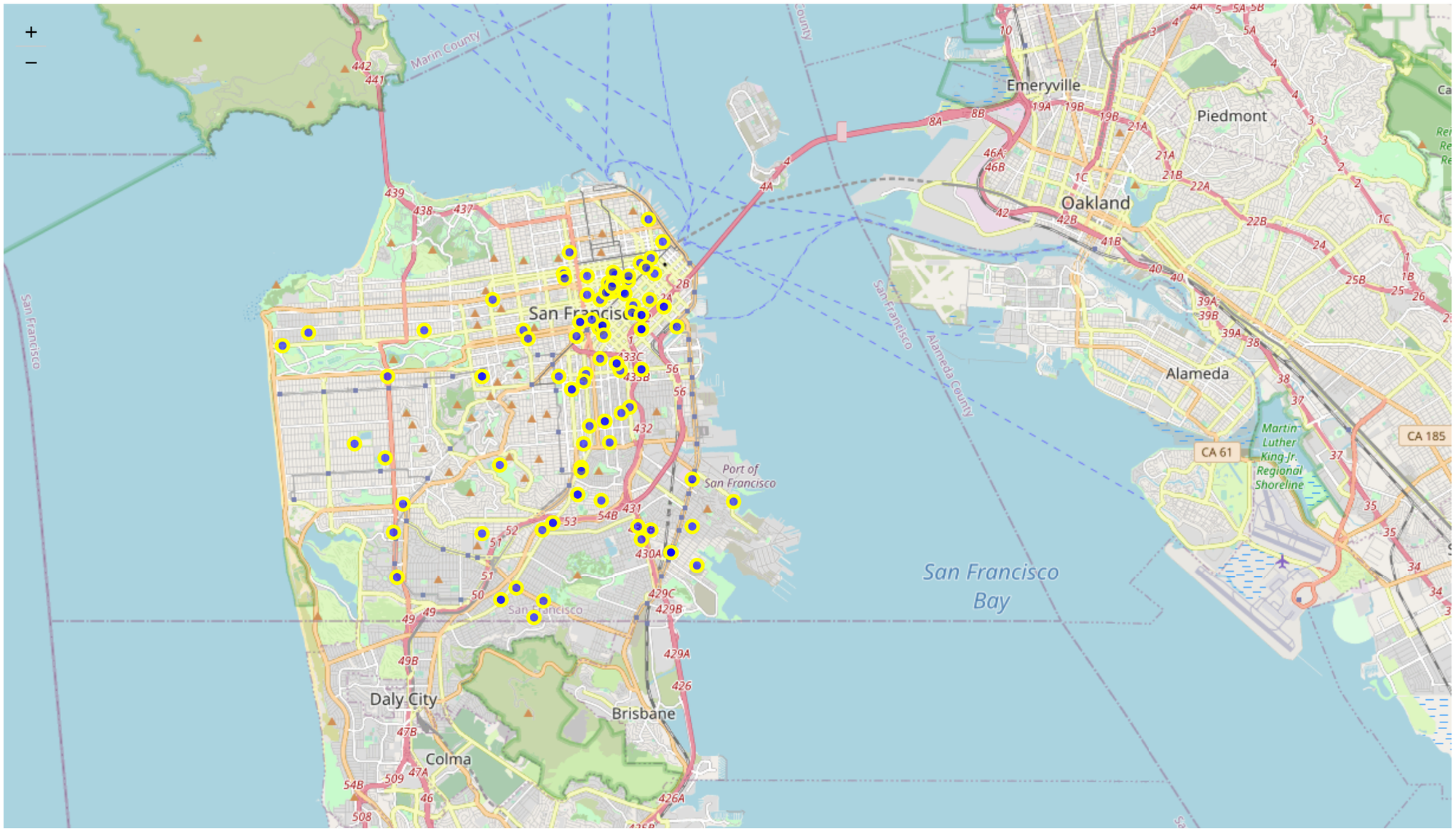
In [14]:
```
# San Francisco latitude and longitude values
latitude = 37.77
longitude = -122.42
# create map and display it
sanfran_map = folium.Map(location=[latitude, longitude], zoom_start=12)

# display the map of San Francisco
sanfran_map
```

Out[14]:



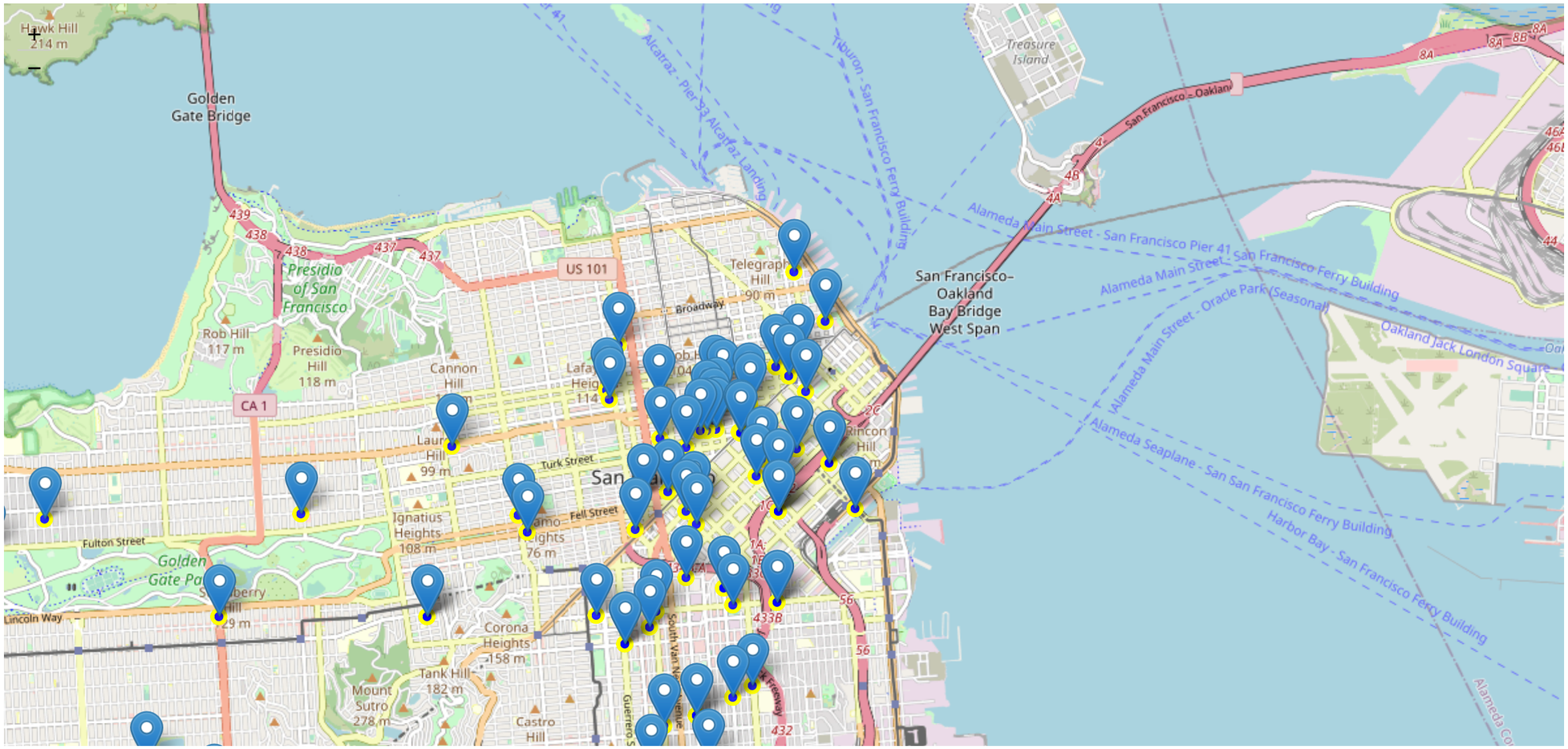
In [15]:
```
# instantiate a feature group for the incidents in the dataframe
incidents = folium.map.FeatureGroup()

# loop through the 100 crimes and add each to the incidents feature group
for lat, lng, in zip(df_incidents.Y, df_incidents.X):
    incidents.add_child(
        folium.features.CircleMarker(
            [lat, lng],
            radius=5, # define how big you want the circle markers to be
            color='yellow',
            fill=True,
            fill_color='blue',
            fill_opacity=0.6
        )
    )

# add incidents to map
sanfran_map.add_child(incidents)
```

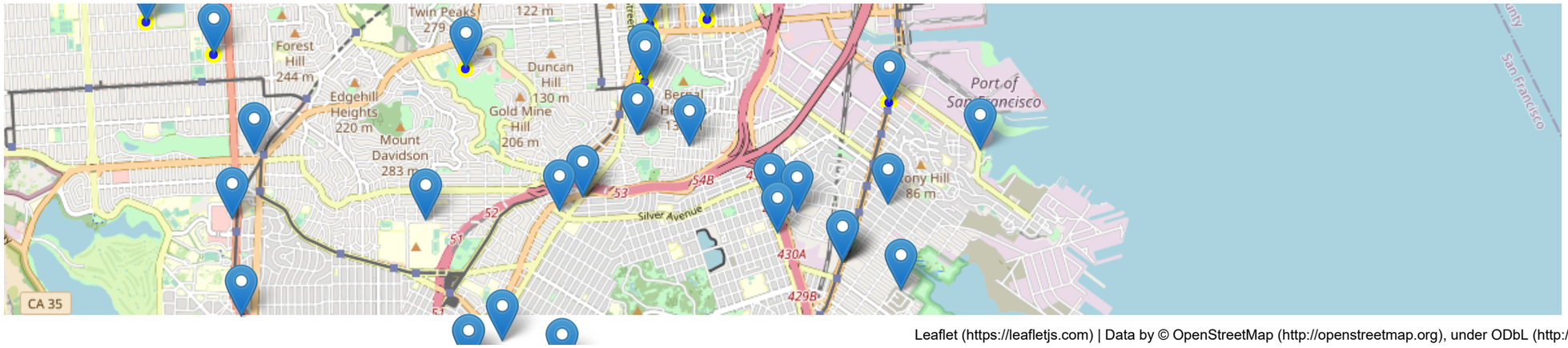
Out[15]:



In [16]:
```
# instantiate a feature group for the incidents in the dataframe
incidents = folium.map.FeatureGroup()

# loop through the 100 crimes and add each to the incidents feature group
for lat, lng, in zip(df_incidents.Y, df_incidents.X):
    incidents.add_child(
        folium.features.CircleMarker(
            [lat, lng],
            radius=5, # define how big you want the circle markers to be
            color='yellow',
            fill=True,
            fill_color='blue',
            fill_opacity=0.6
        )
    )
# add pop-up text to each marker on the map
latitudes = list(df_incidents.Y)
longitudes = list(df_incidents.X)
labels = list(df_incidents.Category)

for lat, lng, label in zip(latitudes, longitudes, labels):
    folium.Marker([lat, lng], popup=label).add_to(sanfran_map)

# add incidents to map

sanfran_map.add_child(incidents)
```

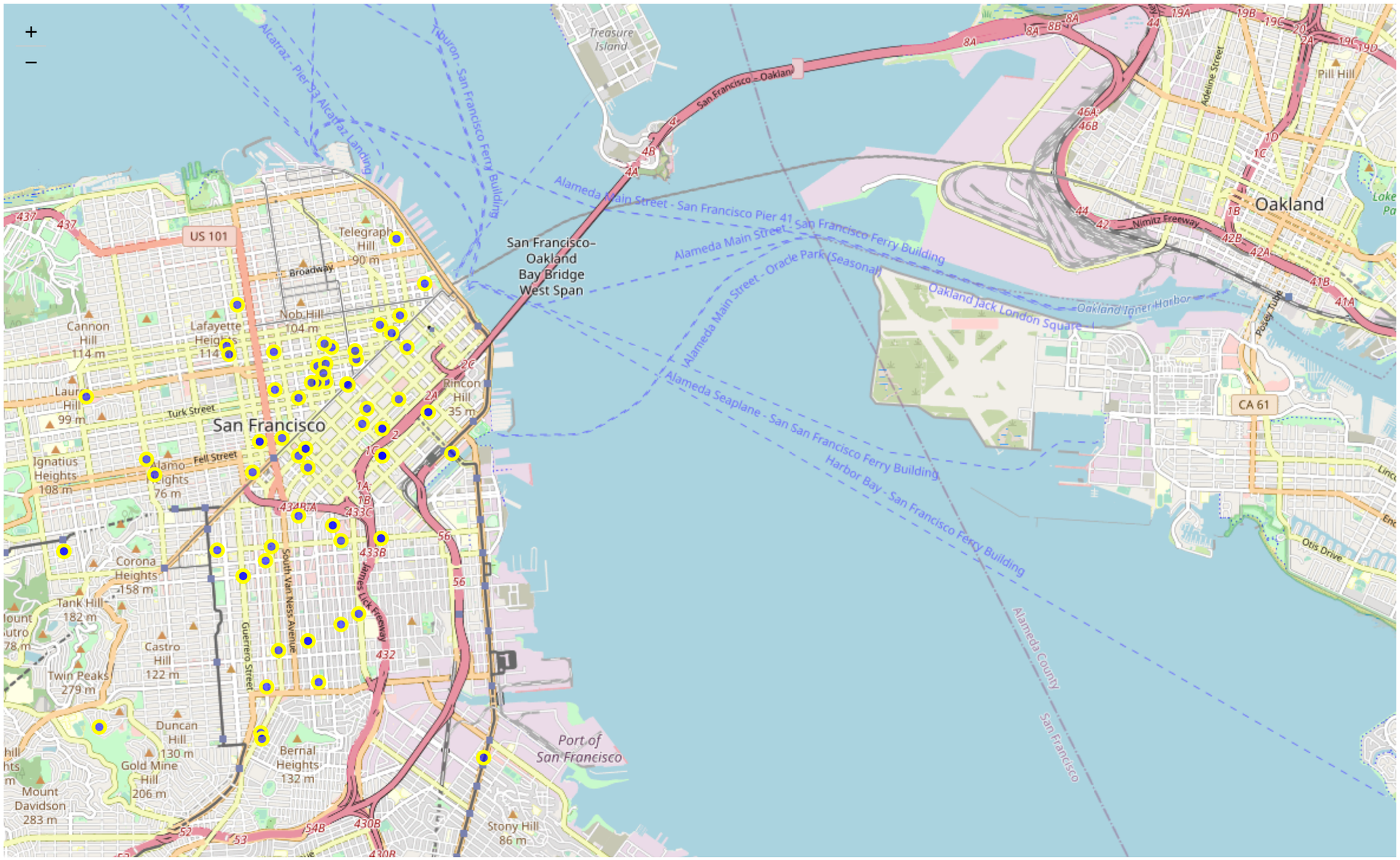
Out[16]:





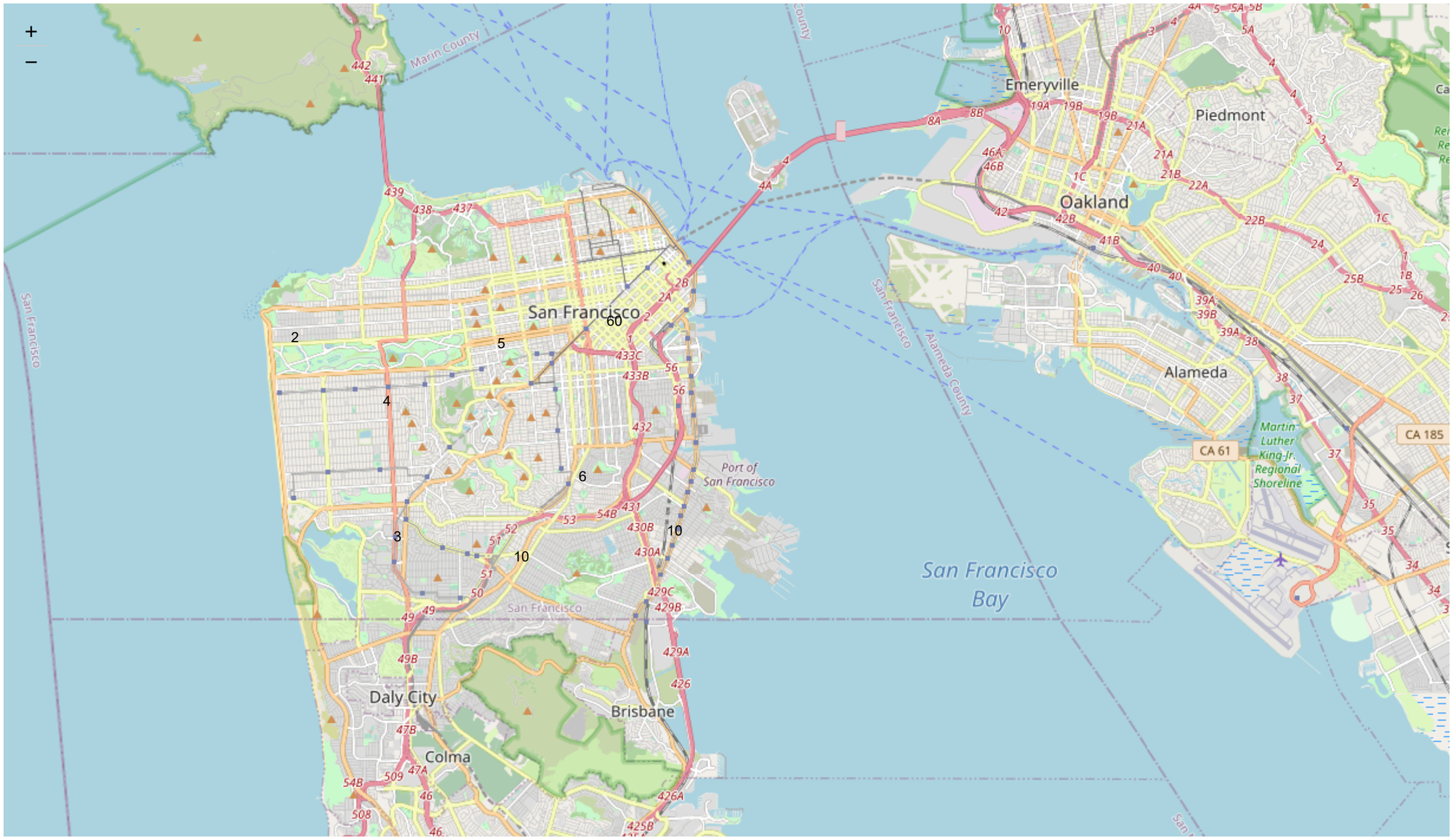
In [17]:
```
# create map and display it
sanfran_map = folium.Map(location=[latitude, longitude], zoom_start=12)

# loop through the 100 crimes and add each to the map
for lat, lng, label in zip(df_incidents.Y, df_incidents.X, df_incidents.Category):
    folium.features.CircleMarker(
        [lat, lng],
        radius=5, # define how big you want the circle markers to be
        color='yellow',
        fill=True,
        popup=label,
        fill_color='blue',
        fill_opacity=0.6
    ).add_to(sanfran_map)

# show map
sanfran_map
```

Out[17]:



In [18]:
```
from folium import plugins

# let's start again with a clean copy of the map of San Francisco
sanfran_map = folium.Map(location = [latitude, longitude], zoom_start = 12)

# instantiate a mark cluster object for the incidents in the dataframe
incidents = plugins.MarkerCluster().add_to(sanfran_map)

# loop through the dataframe and add each data point to the mark cluster
for lat, lng, label, in zip(df_incidents.Y, df_incidents.X, df_incidents.Category):
    folium.Marker(
        location=[lat, lng],
        icon=None,
        popup=label,
    ).add_to(incidents)

# display map
sanfran_map
```

Out[18]:



In [19]:
```
# Import required libraries
import pandas as pd
import plotly.express as px
```

In [20]:
```
URL = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-SkillsNetwork/Data%20Files/airline_data.csv'

airline_data =  pd.read_csv(URL,
                            encoding = "ISO-8859-1",
                            dtype={'Div1Airport': str, 'Div1TailNum': str,
                                   'Div2Airport': str, 'Div2TailNum': str})

print('Data downloaded and read into a dataframe!')
```

```
Data downloaded and read into a dataframe!
```

In [21]:
```
airline_data.head()
```

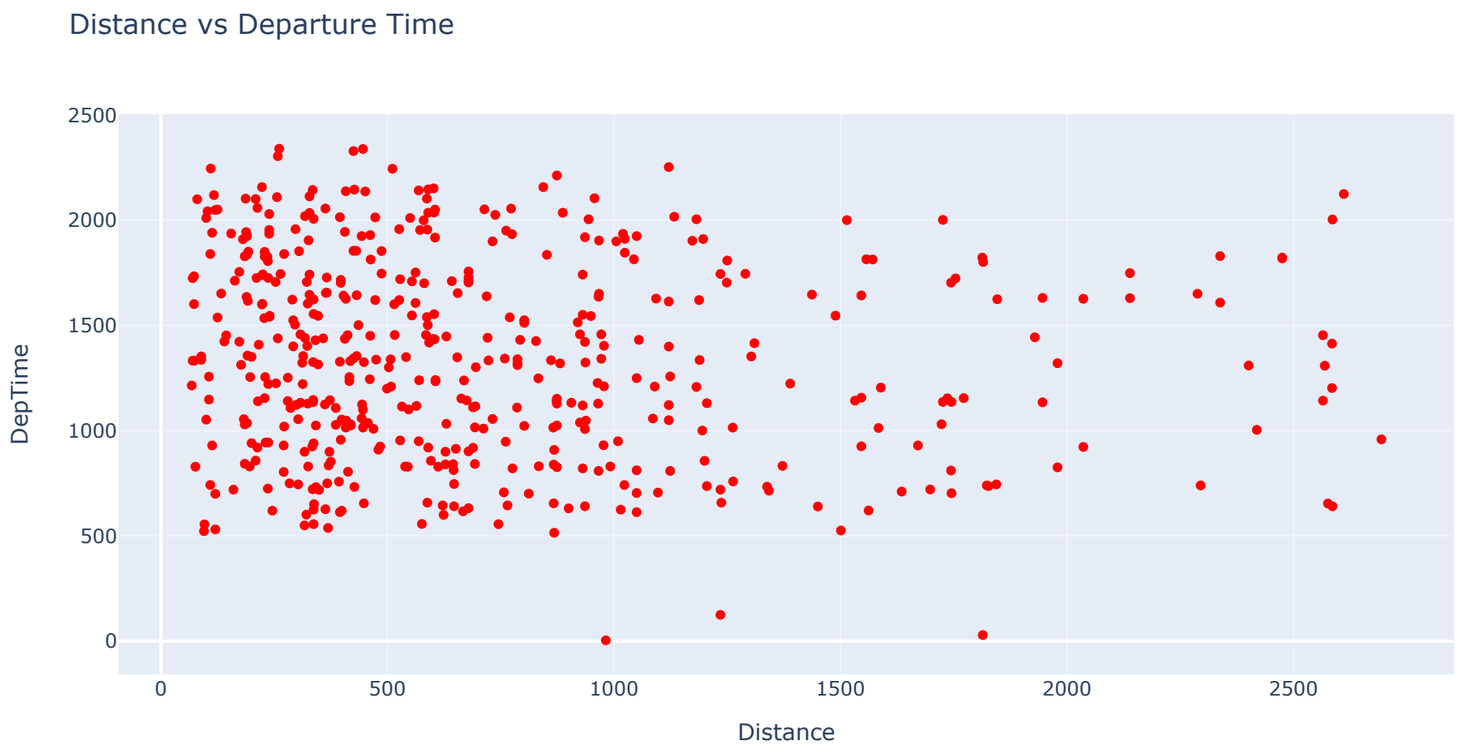
Out[21]:

	Unnamed: 0	Year	Quarter	Month	DayofMonth	DayOfWeek	FlightDate	Reporting_Airline	DOT_ID_Reporting_Airline	IATA_CODE_Reporting_Airline	...	Div4WheelsOff	Div4TailNum	Div5Airport	Div5AirportID	Div5AirportSeq
0	1295781	1998	2	4	2	4	1998-04-02	AS	19930	AS	...	NaN	NaN	NaN	NaN	N
1	1125375	2013	2	5	13	1	2013-05-13	EV	20366	EV	...	NaN	NaN	NaN	NaN	N
2	118824	1993	3	9	25	6	1993-09-25	UA	19977	UA	...	NaN	NaN	NaN	NaN	N
3	634825	1994	4	11	12	6	1994-11-12	HP	19991	HP	...	NaN	NaN	NaN	NaN	N
4	1888125	2017	3	8	17	4	2017-08-17	UA	19977	UA	...	NaN	NaN	NaN	NaN	N

5 rows × 110 columns

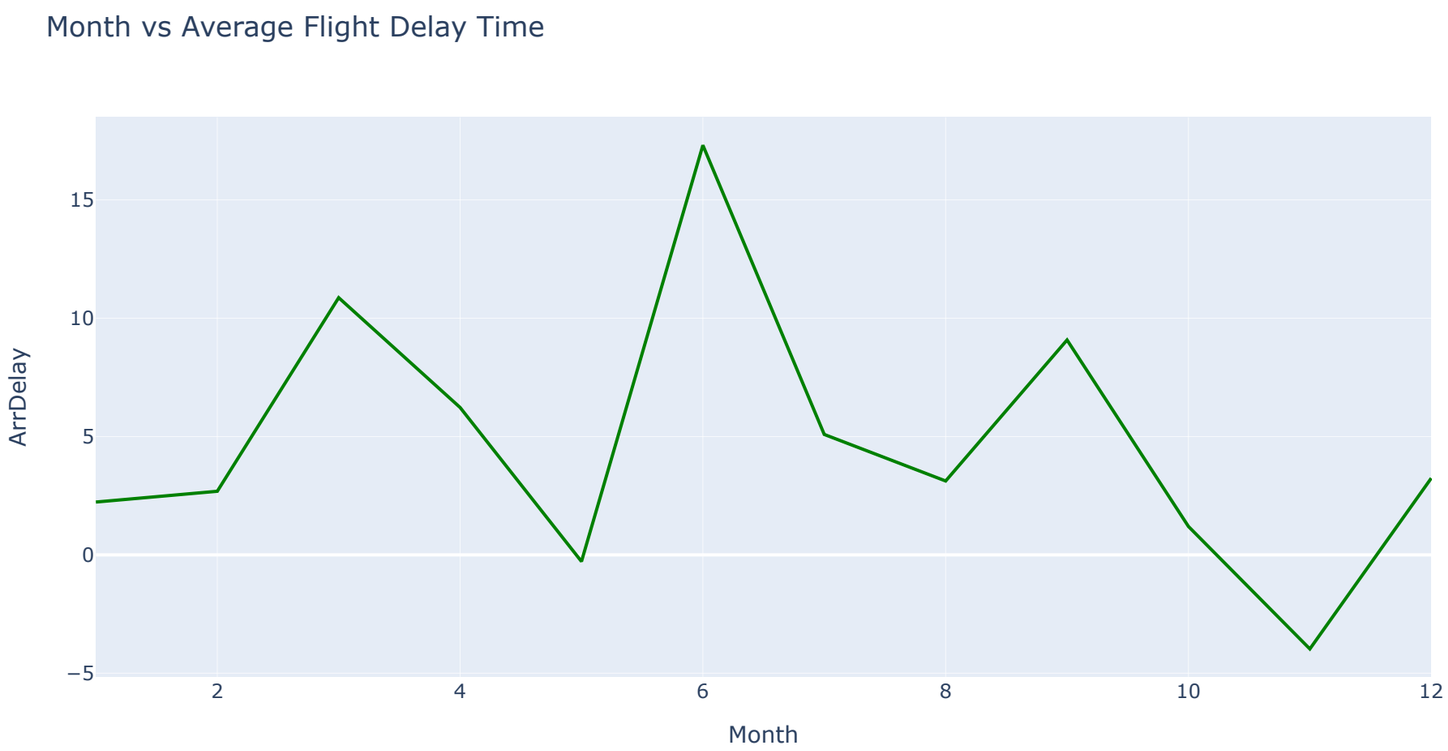
In [22]:

```
import plotly.graph_objs as go
# Randomly sample 500 data points. Setting the random state to be 42 so that we get same result.
data = airline_data.sample(n=500, random_state=42)
# First we create a figure using go.Figure and adding trace to it through go.scatter
fig = go.Figure(data=go.Scatter(x=data['Distance'], y=data['DepTime'], mode='markers', marker=dict(color='red')))
# Updating layout through `update_layout`. Here we are adding title to the plot and providing title to x and y axis.
fig.update_layout(title='Distance vs Departure Time', xaxis_title='Distance', yaxis_title='DepTime')
# Display the figure
fig.show()
```



In [23]:
```
# Group the data by Month and compute average over arrival delay time.
line_data = data.groupby('Month')['ArrDelay'].mean().reset_index()
fig = go.Figure(data=go.Line(x=line_data['Month'], y=line_data['ArrDelay'], mode='lines', marker=dict(color='green')))
fig.update_layout(title='Month vs Average Flight Delay Time', xaxis_title='Month', yaxis_title='ArrDelay')
fig.show()
```

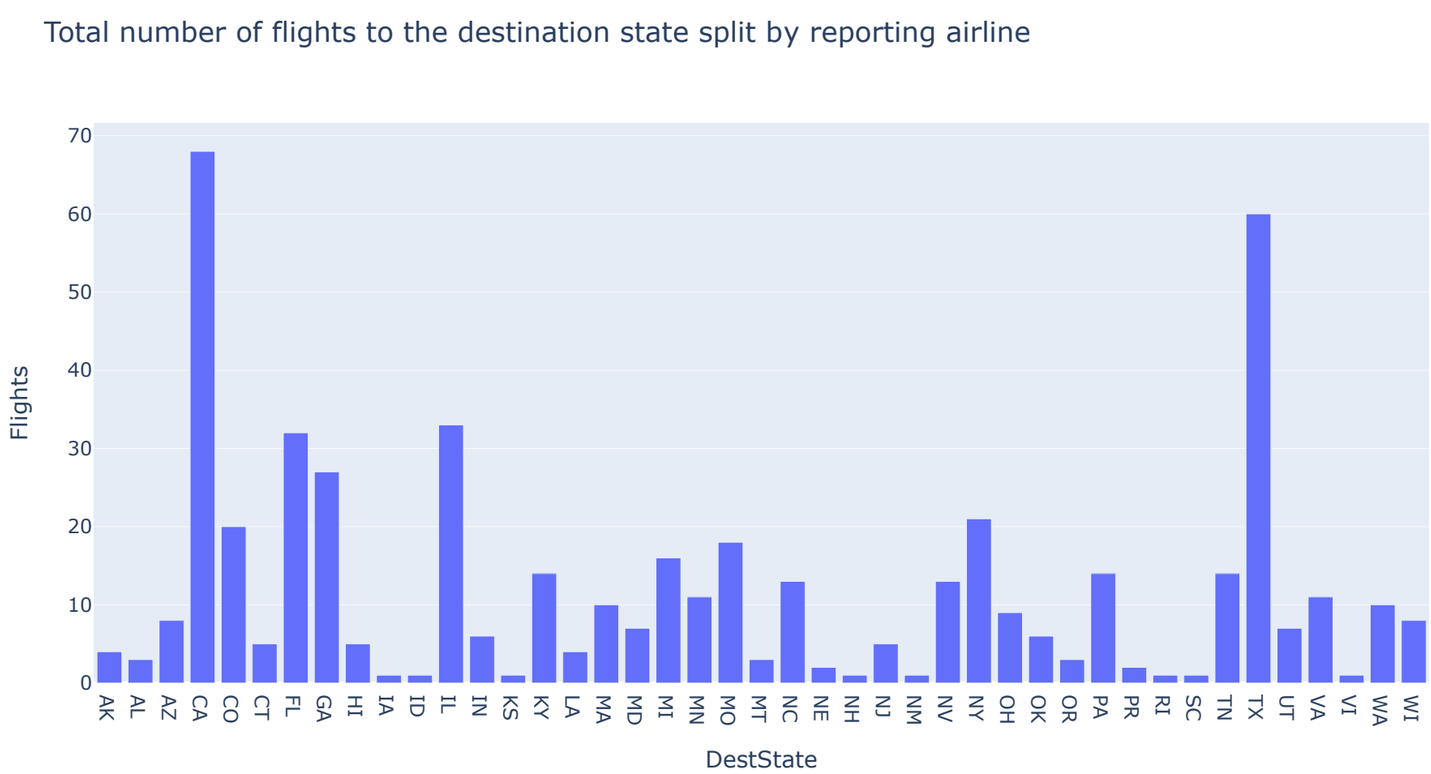
```
C:\Users\yogis\anaconda3\lib\site-packages\plotly\graph_objs\_deprecations.py:378: DeprecationWarning:

plotly.graph_objs.Line is deprecated.
Please replace it with one of the following more specific types
  - plotly.graph_objs.scatter.Line
  - plotly.graph_objs.layout.shape.Line
  - etc.
```



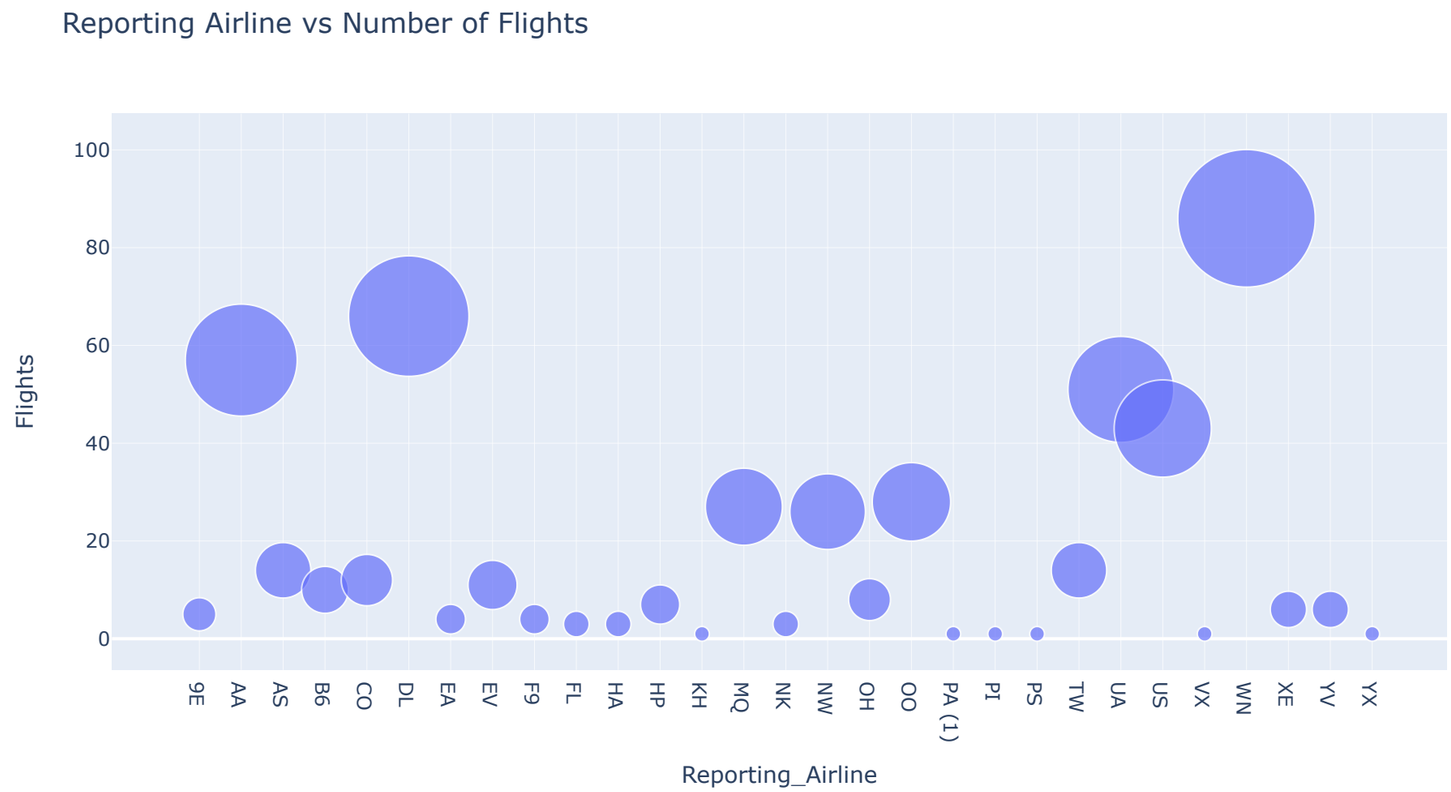
Using Plotly

In [24]:
```
# Group the data by destination state and reporting airline. Compute total number of flights in each combination
bar_data = data.groupby('DestState')['Flights'].sum().reset_index()
```

In [25]:
```
# Use plotly express bar chart function px.bar. Provide input data, x and y axis variable, and title of the chart.
# This will give total number of flights to the destination state.
fig = px.bar(bar_data, x="DestState", y="Flights", title='Total number of flights to the destination state split by reporting airline')
fig.show()
```

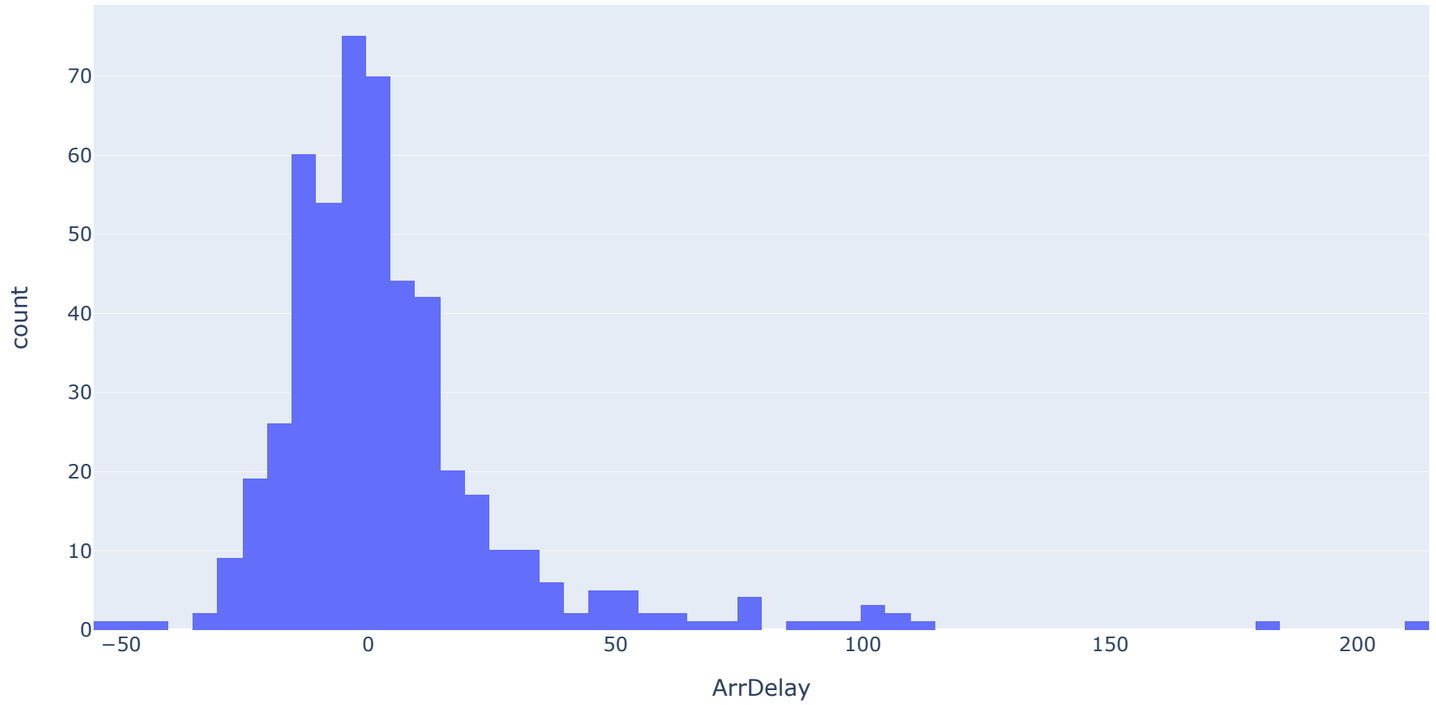


In [26]:
```
# Group the data by reporting airline and get number of flights
bub_data = data.groupby('Reporting_Airline')['Flights'].sum().reset_index()

fig = px.scatter(bub_data, x="Reporting_Airline", y="Flights", size="Flights",
                 hover_name="Reporting_Airline", title='Reporting Airline vs Number of Flights', size_max=60)
fig.show()
```

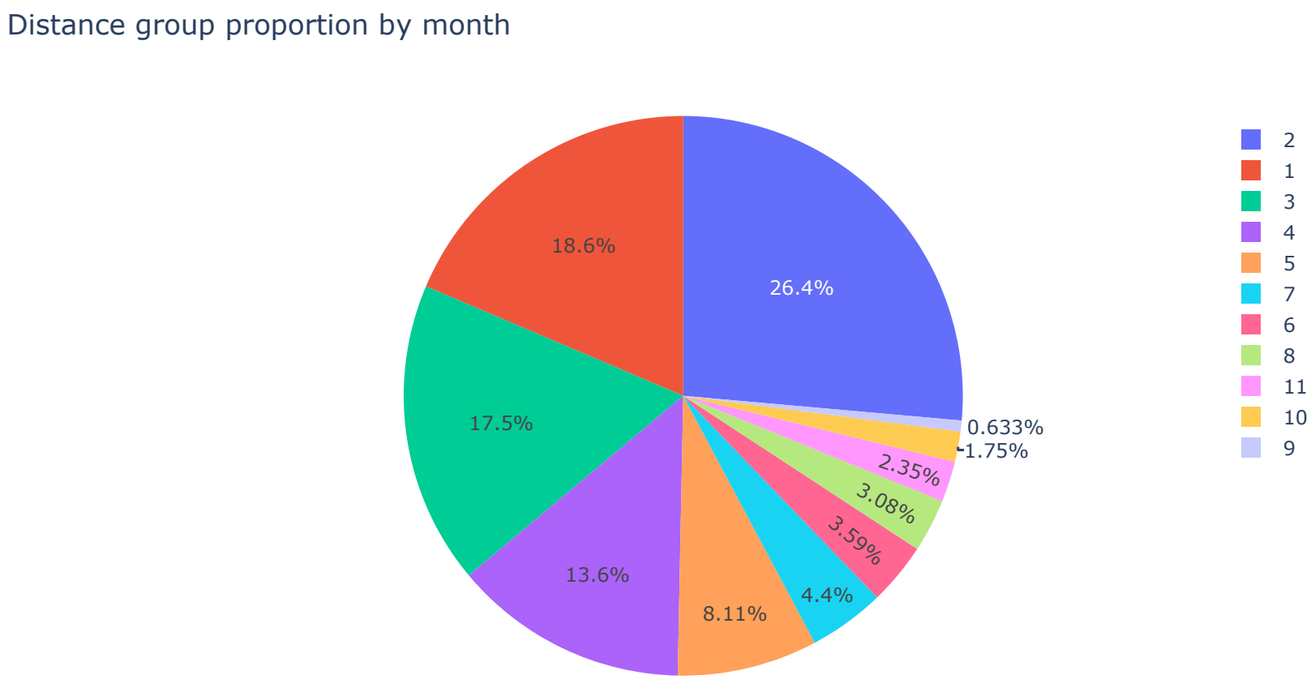


In [27]:
```
# Set missing values to 0
data['ArrDelay'] = data['ArrDelay'].fillna(0)
```

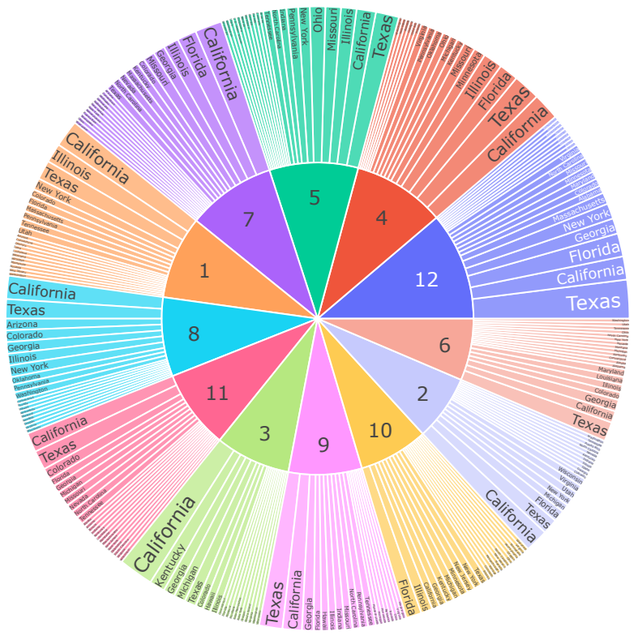
In [28]:
```
fig = px.histogram(data, x="ArrDelay")
fig.show()
```



In [29]:
```
# Use px.pie function to create the chart. Input dataset.
# Values parameter will set values associated to the sector. 'Month' feature is passed to it.
# labels for the sector are passed to the `names` parameter.
fig = px.pie(data, values='Month', names='DistanceGroup', title='Distance group proportion by month')
fig.show()
```



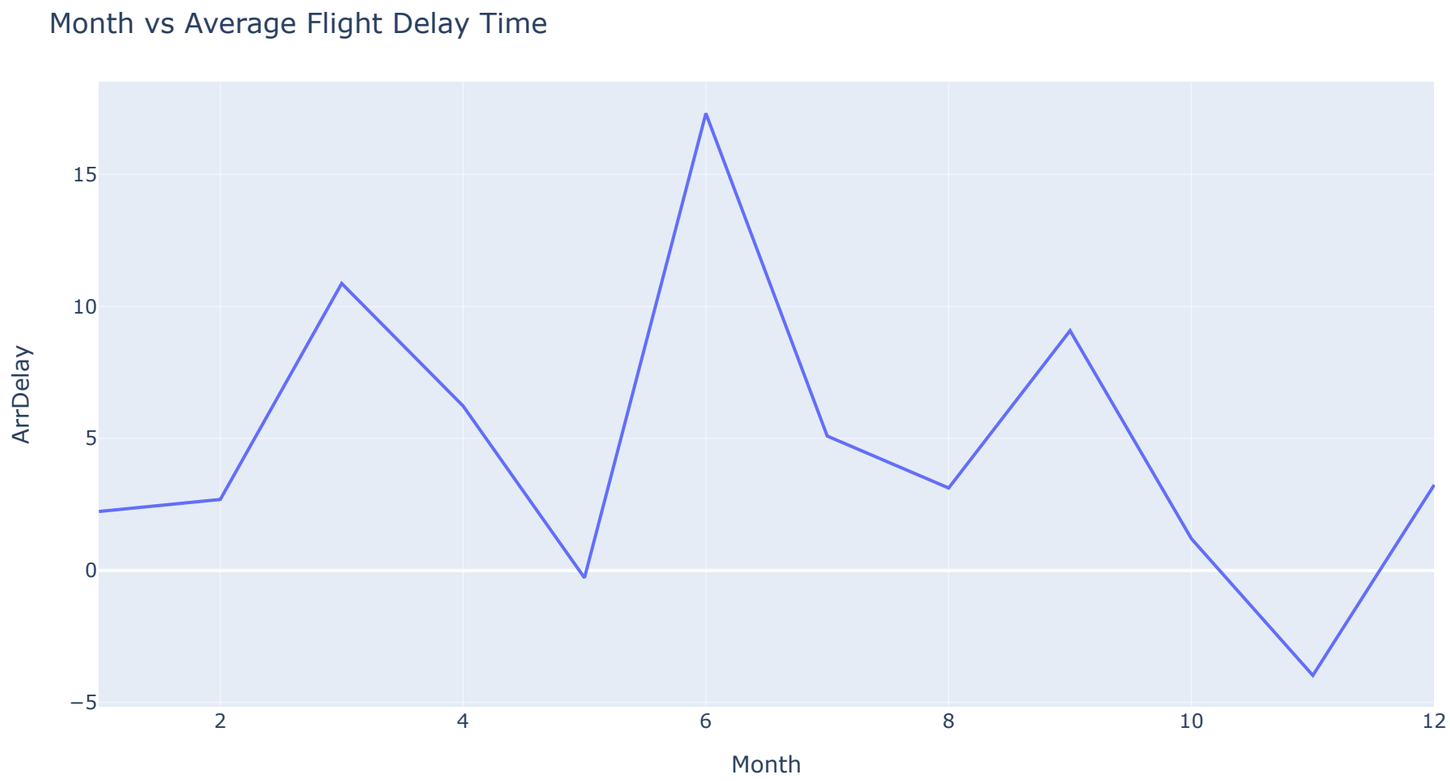
In [30]:
```
fig = px.sunburst(data, path=['Month', 'DestStateName'], values='Flights')
fig.show()
```



In [31]:
```
import dash
import dash_html_components as html
import dash_core_components as dcc
from dash.dependencies import Input, Output
# Create a dash application
app = dash.Dash(__name__)

# Get the layout of the application and adjust it.
# Create an outer division using html.Div and add title to the dashboard using html.H1 component
# Add description about the graph using HTML P (paragraph) component
# Finally, add graph component.
app.layout = html.Div(children=[html.H1('Airline Dashboard',
style={'textAlign': 'center', 'color': '#503D36', 'font-size': 40}),
                                html.P('Proportion of distance group (250 mile distance interval group) by flights.',
                                            style={'textAlign':'center', 'color': '#F57241'}),
                                dcc.Graph(figure=fig),

                    ])

# Run the application
if __name__ == '__main__':
    app.run_server()
```

```
Dash is running on http://127.0.0.1:8050/ (http://127.0.0.1:8050/)

 * Serving Flask app "__main__" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
<ipython-input-31-49295a4396ff>:2: UserWarning:


The dash_html_components package is deprecated. Please replace
`import dash_html_components as html` with `from dash import html`

<ipython-input-31-49295a4396ff>:3: UserWarning:


The dash_core_components package is deprecated. Please replace
`import dash_core_components as dcc` with `from dash import dcc`

 * Running on http://127.0.0.1:8050/ (http://127.0.0.1:8050/) (Press CTRL+C to quit)
```

In [33]:
```
fig = px.line(data, x=line_data['Month'], y=line_data['ArrDelay'])
fig.update_layout(title='Month vs Average Flight Delay Time', xaxis_title='Month', yaxis_title='ArrDelay')
fig.show()
```



In [ ]:
```
```