Data, Metadata and APIs

Part 5: The Google Maps API and Open Data

Now that you've extracted GPS coordinates from JPEG metadata and mapped it using the Google Maps API, you might be wondering what else you can do with the Google Maps API. The short answer is... a lot.

In this notebook, you'll see how to combine your knowledge of the Google Maps API with your knowledge of data analysis with Pandas.

Find an Open Data Set that contains Location Data

Here's a data set that tracks the location of all potholes filled by the City of Chicago for the past 7 days. Chicago is known for its potholes (https://www.wbez.org/shows/curious-city/city-of-big-potholes-is-asphalt-the-best-choice-for-chicagos-streets/8bbd9e7a-b27e-4e00-a868-aa0b826b53b2), so this should be good.

We will load this *.csv* file in from a URL so that it is guaranteed to be the most up-to-date as possible:

In [1]:
```
# Note: the spike in traffic from Fremd may get us IP-banned by Chicago's Open Data portal.
#       If this happens, your teacher will share a static copy of Potholes_Patched.csv,
#       and you'll need to run the code "potholes_DF = pd.read_csv('Potholes_Patched.csv')"

import pandas as pd

potholes_DF = pd.read_csv("Potholes_Patched.csv")

# display the 3 most recent potholes that were filled
potholes_DF[-3:]
```

Out[1]:

	ADDRESS	REQUEST DATE	COMPLETION DATE	NUMBER OF POTHOLES FILLED ON BLOCK	LATITUDE	LONGITUDE	LOCA
106304	600 W 59TH ST	4/4/2022 13:51	4/4/2022 13:52	23	41.787195	-87.640291	P((-87.64029114 41.787194823
106305	1000 W 59TH ST	4/4/2022 13:56	4/4/2022 13:57	10	41.787045	-87.649957	P((-87.64995714 41.787044902
106306	6328 N LINCOLN AVE	3/31/2022 17:21	4/4/2022 12:14	1	41.996206	-87.717424	P((-87.71742410 41.996205985

Check how many potholes were filled in the last week since the spreadsheet was generated:

In [2]:
```
print(len(potholes_DF))
```

```
106307
```

That's a lot of potholes. Now extract the location data, clean out the "nan" values, and store it as a list of tuples:

In [3]:
```
import numpy as np

lat = list(potholes_DF["LATITUDE"])

lon = list(potholes_DF["LONGITUDE"])

tuple_list = []

'''
for i in range(len(lat)):
    coord = (lat[i],lon[i])
    tuple_list.append(coord)
'''

tuple_list = [(lat[i],lon[i]) for i in range(len(lat))]

tuple_list = [x for x in tuple_list if not np.isnan(x[1])]
```

Let's compare the length of *potholes_DF* to *tuple_list* to see how many "nan" values we cleaned out:

In [4]:
```
print(len(potholes_DF),len(tuple_list))
```

```
106307 105955
```

Depending on the week, there may be a handful of "nan" values to clean out. If you were lucky, there were none.

Now let's look at a few of the tuples in the list:

In [5]:
```
tuple_list[-10:]
```

Out[5]:
```
[(41.745690100000004, -87.60546472),
 (41.79259436, -87.79514811),
 (41.96039461, -87.68338403),
 (41.78973533, -87.70494202),
 (41.81948525, -87.69375105),
 (41.99997707, -87.69576195),
 (41.73013202, -87.54693733),
 (41.78719482, -87.64029115),
 (41.7870449, -87.64995715),
 (41.99620599, -87.71742411)]
```

Google Maps API with Markers

Let's put a marker every place we found a pothole.

WARNING: Adding more than 500 marker points could potentially crash your kernel! To combat this, we are creating a list of 500 random entries from the original tuple list.

In [6]:
```
import numpy as np

tuple_list_500 = []
indicies_used = []
for i in range(500):                                    # Loop 500 times
    random = np.random.randint(0,500)                   # Generate random index nu
mber
    if random not in indicies_used:                     # Check if number has alre
ady been generated
        indicies_used.append(random)                    # Add new number to list o
f used numbers
        tuple_list_500.append(tuple_list[random])   # Add the tuple from that
 index to the new list of 500
print(indicies_used[:50])
#indicies_used = [random for np.random.randint(0,500) in range(500) if random
 not in indicies_used]
print(tuple_list_500[:10])
```

```
[170, 370, 229, 394, 248, 241, 367, 247, 475, 99, 477, 377, 290, 160, 495, 46
4, 345, 194, 168, 459, 135, 246, 306, 266, 398, 383, 64, 260, 359, 342, 81, 6
1, 29, 280, 255, 179, 299, 401, 55, 336, 444, 304, 23, 313, 105, 276, 36, 29
2, 166, 461]
[(41.83689406, -87.64893936), (41.97174516, -87.75038074), (41.90953839999999
5, -87.75349723), (42.00091753, -87.69481883), (42.00186853, -87.67016764),
(41.92156264, -87.66863099), (41.912562, -87.791459), (41.73480702, -87.59914
147), (41.71587378, -87.64927423), (41.9012417, -87.63442009999999)]
```

In [7]:
```
# Import the gmaps python module and load in your API Key:
import gmaps
gmaps.configure(api_key="AIzaSyCLla6Q7krE9xNg6SnNMoGNIzjCLddE9EU")
```

In [8]:
```
from ipywidgets.embed import embed_minimal_html # Allows us to create a separt
e file for the Google Maps

markers = gmaps.marker_layer(tuple_list_500)    # Create markers for each tupl
e/coordinate
markermap = gmaps.Map()                         # Create a GMap variable
markermap.add_layer(markers)                    # Add the layer of markers to
 GMap

embed_minimal_html('output/MarkerMap1.html', views=[markermap])
print("*** If no map appears, uncomment the line above, re-run this cell, and
 check your 'Metadata Part 5' folder to find the new HTML file name \"MarkerMa
p1.html\". ***")

markermap
```

```
*** If no map appears, uncomment the line above, re-run this cell, and check
your 'Metadata Part 5' folder to find the new HTML file name "MarkerMap1.htm
l". ***
```

Question 1: Look at the marker map at various zoom levels. What do you notice above the graph? Comment on anything interesting you see and try to summarize "the good" and "the bad" in this visualization.

Your Answer: Most of the potholes are all in chicago and there aren't much outside chicago.

Google Maps API to Create a Heatmap

Instead of markers, let's make a heat map:

WARNING: Adding more than 500 marker points could potentially crash your kernel! To combat this, we are again using the list of 500 random entries from the original tuple_list.

In [9]:
```
from ipywidgets.embed import embed_minimal_html # Allows us to create a separt
e file for the Google Maps

heatm = gmaps.Map()
heatm.add_layer(gmaps.heatmap_layer(tuple_list_500))

embed_minimal_html('output/HeatMap1.html', views=[markermap])
print("*** If no map appears, uncomment the line above, re-run this cell, and
 check your 'Metadata Part 5' folder to find the new HTML file name \"HeatMap
1.html\". ***")

heatm
```

```
*** If no map appears, uncomment the line above, re-run this cell, and check
your 'Metadata Part 5' folder to find the new HTML file name "HeatMap1.html".
***
```

Question 2: Look at the heatmap at various zoom levels. What do you notice above the graph? Comment on anything interesting you see and try to summarize "the good" and "the bad" in this visualization.

Your Answer: The most red parts are right in chicago.

Task 1: Find your own dataset!

You are going to create a marker map **and** a heatmap from a dataset you have found. For Task 1, find a dataset with location data (GPS coordinates!). Fill in the following:

Name: Charan Chandran

Date: 5/18/22

Source for Data Set: Kaggle

URL for Data Set: https://www.kaggle.com/datasets/andrewmvd/us-schools-dataset/download (https://www.kaggle.com/datasets/andrewmvd/us-schools-dataset/download)

Description of Data Set: Data on 130k+ schools in the US with georeferences.

File Format for Data Set: csv

Age of Data Set: November 29, 2021

Task 2: Show some entries fom your dataset

Import your data set as a Pandas Data Frame, then show the last 10 entries:

In [10]:
```
import pandas as pd
#import io, requests
#import json

#url_to_file = requests.get('https://data.montgomerycountymd.gov/api/views/772
q-4wm8/rows.csv?accessType=DOWNLOAD').content
#public_schools = pd.read_csv(io.StringIO(url_to_file.decode('utf-8')))
public_schools = pd.read_csv('./data/public_schools.csv')

public_schools[-10:]
```

Out[10]:

	X	Y	OBJECTID	NCESID	NAME	ADDRESS	
102324	-8.643788e+06	5.333242e+06	102325	362475003390	SCHOOL 16-JOHN WALTON SPENCER	625 SCIO ST	
102325	-1.083862e+07	4.518220e+06	102326	200705000366	REX ELEM	1100 W. GRAND	
102326	-9.558473e+06	5.291177e+06	102327	261884005566	SOUTH ELEMENTARY SCHOOL	4900 40TH	H
102327	-9.141667e+06	4.704477e+06	102328	540078000625	POINT PLEASANT JUNIOR/SENIOR HIGH SCHOOL	280 SCENIC ROAD	
102328	-1.028888e+07	3.673479e+06	102329	220129000115	RAPIDES TRAINING ACADEMY	901 CREPE MYRTLE STREET	
102329	-1.075152e+07	5.688108e+06	102330	270015003019	LISMORE COLONY SCHOOL	80391 COUNTY RD 60	
102330	-1.031719e+07	4.598066e+06	102331	290699000174	OSAGE BEACH ELEM.	1241 NICHOLS ROAD	
102331	-8.512030e+06	4.450964e+06	102332	510264001238	POINT OPTION ALTERNATIVE SCHOOL	813 DILIGENCE DR.	
102332	-9.024169e+06	4.580376e+06	102333	540030000182	GATEWOOD ELEMENTARY	5094 GATEWOOD ROAD	F
102333	-1.345981e+07	4.856354e+06	102334	69110211706	PLUMAS COUNTY OPPORTUNITY	1446 E. MAIN ST.	

10 rows × 33 columns

Task 3: Create a list of tuples

Use your dataset to create a list of tuples (a list of DD coordinates) representing the locations in your dataset:

WARNING: Adding more than 500 marker points could potentially crash your kernel! To combat this, create a list of 500 random entries from the original list of tuples.

In [11]:
```
'''
import numpy as np

x = list(public_schools["X"])

y = list(public_schools["X"])

tuple_list = [(x[i],y[i]) for i in range(len(x))]

tuple_list = [x for x in tuple_list if not np.isnan(x[1])]

tuple_list[:10]
'''
''
```

Out[11]: ''

In [12]:
```
'''
import pyproj

# Spatial Reference System
proj = pyproj.Transformer.from_crs(3857, 4326, always_xy = True)

tuple_list_lat_lon = []
for i in range(len(tuple_list)):
    tuple = (proj.transform(tuple_list[i][0],tuple_list[i][1]))
    tuple = (tuple[1],tuple[0])
    tuple_list_lat_lon.append(tuple)

tuple_list_lat_lon[:10]
'''
''
```

Out[12]: ''

In [13]:
```
import numpy as np

lat = list(public_schools["LATITUDE"])

lon = list(public_schools["LONGITUDE"])

tuple_list = [(lat[i],lon[i]) for i in range(len(lat))]

tuple_list = [x for x in tuple_list if not np.isnan(x[1])]

tuple_list[:10]
```

Out[13]:
```
[(42.465566095, -88.431010375),
 (35.23190937, -80.911501287),
 (39.489752551, -86.0624298709999),
 (33.242888236999995, -111.687306008),
 (33.222170136, -111.68376840299999),
 (33.2558117040001, -111.706043188),
 (33.4095317000001, -112.45141325899999),
 (36.9125226350001, -111.45826466),
 (33.3237236090001, -111.59965325799999),
 (38.609867949, -121.37027199299999)]
```

In [14]:
```
tuple_list_500 = []
indicies_used = []
for i in range(3000):                                  # Loop 500 times
    random = np.random.randint(0, len(tuple_list))               # Generate ra
ndom index number
    if random not in indicies_used:                  # Check if number has alre
ady been generated
        indicies_used.append(random)                 # Add new number to list o
f used numbers
        tuple_list_500.append(tuple_list[random])    # Add the tuple from that
 index to the new list of 500
print(indicies_used[:50])
#indicies_used = [random for np.random.randint(0,500) in range(500) if random
 not in indicies_used]
print(tuple_list_500[:10])
```

```
[36699, 12532, 13708, 97243, 64707, 44644, 84452, 68356, 52328, 32107, 72258,
49982, 24226, 33592, 54063, 99408, 85372, 64355, 17008, 56897, 8822, 376, 101
232, 22599, 75704, 75811, 80870, 12577, 39991, 47879, 38219, 95171, 6457, 918
80, 2495, 10450, 80583, 39710, 62045, 87098, 68082, 5507, 14400, 72640, 6669
4, 68112, 98745, 26691, 97987, 13541]
[(41.4136421110001, -80.5794275059999), (36.918328532, -76.252983774), (39.27
8585053, -81.5448999659999), (43.7873175060001, -88.46549747700001), (42.9433
147660001, -88.858172065), (32.915874186000096, -97.119681326), (38.77757744
4, -121.371971484), (41.690334703000104, -93.82218771), (42.322266771, -83.16
9139569), (40.6865595250001, -95.856042365)]
```

Task 4: Create a marker map from your data

Use the Google Maps API to create a marker map using your list of tuples from above.

In [15]:
```
# Import the gmaps python module and load in your API Key:
import gmaps
gmaps.configure(api_key="AIzaSyCLla6Q7krE9xNg6SnNMoGNIzjCLddE9EU")
```

In [16]:
```
from ipywidgets.embed import embed_minimal_html # Allows us to create a separat
e file for the Google Maps

markers = gmaps.marker_layer(tuple_list_500)    # Create markers for each tupl
e/coordinate
markermap = gmaps.Map()                         # Create a GMap variable
markermap.add_layer(markers)                    # Add the layer of markers to
 GMap

embed_minimal_html('output/MarkerMap2.html', views=[markermap])
print("*** If no map appears, uncomment the line above, re-run this cell, and
 check your 'Metadata Part 5' folder to find the new HTML file name \"MarkerMa
p1.html\". ***")

markermap
```

```
*** If no map appears, uncomment the line above, re-run this cell, and check
your 'Metadata Part 5' folder to find the new HTML file name "MarkerMap1.htm
l". ***
```

Task 5: Create a heatmap from your data

Use the Google Maps API to create a **heatmap** using your list of tuples from above.

Note: The Google Maps API can struggle with heatmaps that have more than 1000 datapoints. If your map is not working, try reducing your list to fewer tuples (try creating a list with just the most recent 100 entries in the dataset). Once this works, you can always add in a few more tuples!

In [17]:
```
from ipywidgets.embed import embed_minimal_html # Allows us to create a separat
e file for the Google Maps

heatm = gmaps.Map()
heatm.add_layer(gmaps.heatmap_layer(tuple_list_500))

embed_minimal_html('output/HeatMap2.html', views=[markermap])
print("*** If no map appears, uncomment the line above, re-run this cell, and
 check your 'Metadata Part 5' folder to find the new HTML file name \"HeatMap
1.html\". ***")

heatm
```

```
*** If no map appears, uncomment the line above, re-run this cell, and check
your 'Metadata Part 5' folder to find the new HTML file name "HeatMap1.html".
***
```

Task 6: Comment on what you see

Look at your marker map and your heatmap at various zoom levels. Comment on anything interesting or notable that you see.

Your Answer: From the data I can conclude that many of the school in the US are in the east.

Task 7: Brainstorm further study

If you had more time and resources, what else would you like to explore using the GPS data in this dataset?

Your Answer: I would try to see which states have the most schools and interesting data such as the average population in schools. Also check what percent of schools have their own website.