VacationPy

Note

- Keep an eye on your API usage. Use https://developers.google.com/maps/reporting/gmp-reporting (https://developers.google.com/maps/reporting/gmp-reporting) as reference for how to monitor your usage and billing.
- Instructions have been included for each segment. You do not have to follow them exactly, but they are included to help you think through the steps.

In [1]:
```
# Dependencies and Setup
import openweathermapy as ow
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import requests
import gmaps
import json
import os
import pprint as pprint

# Import API key
from api_keys import g_key
#print(g_key)
```

Store Part I results into DataFrame

- Load the csv exported in Part I to a DataFrame

In [2]:
```
# Importing Part 1 csv file in DataFrame
raw_data_file_df = pd.read_csv("../output_data/weather_json.csv")

# Displaying first 5 rows of dataframe
raw_data_file_df.head()
```

Out[2]:

	City	Lat	Lng	Max Temp	Humidity	Cloudiness	Wind Speed	Country	Date
0	Port Augusta	-32.50	137.77	66.20	77	90	16.11	AU	1608127912
1	Ushuaia	-54.80	-68.30	46.40	70	75	20.80	AR	1608127754
2	Albany	42.60	-73.97	15.01	72	75	8.05	US	1608127711
3	Yellowknife	62.46	-114.35	-40.00	72	20	8.05	CA	1608127913
4	Juneau	58.30	-134.42	30.99	92	90	6.91	US	1608127913

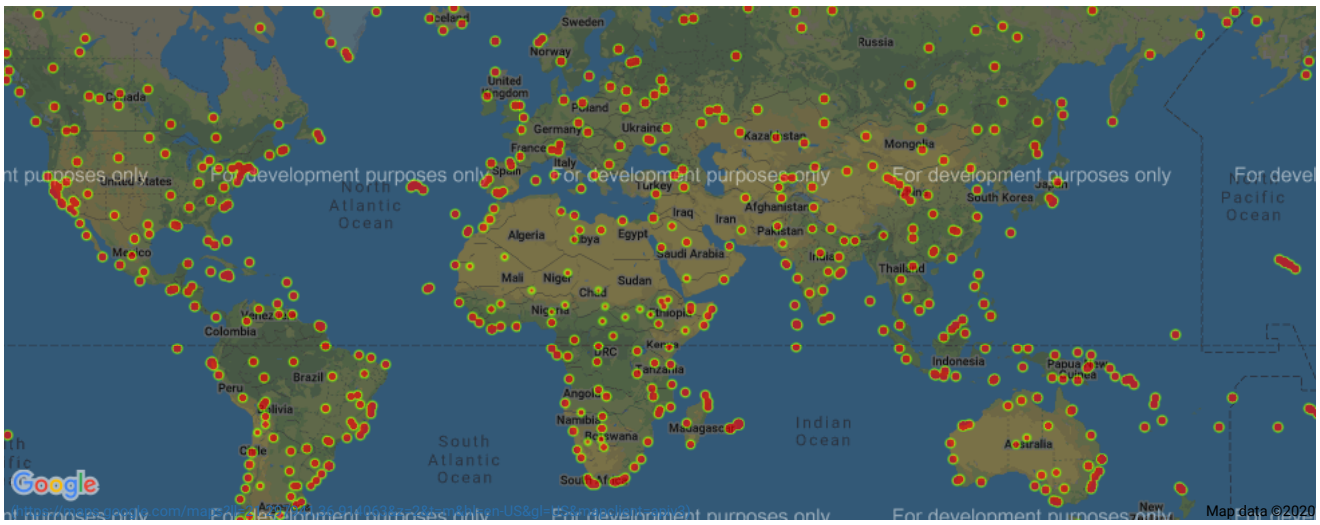
Humidity Heatmap

- Configure gmaps.
- Use the Lat and Lng as locations and Humidity as the weight.
- Add Heatmap layer to map.

In [ ]:
```
```

In [3]:
```
# Store latitude and longitude in locations
locations = raw_data_file_df[['Lat','Lng']]

# Retrieving Humidity data and converting to float
rating = raw_data_file_df["Humidity"].astype(float)
```

In [4]:
```
# Plot Heatmap
fig = gmaps.figure()

# Create heat layer
heat_layer = gmaps.heatmap_layer(locations, weights=rating,
                                 dissipating=False, max_intensity=10,
                                 point_radius=1)


# Add layer
fig.add_layer(heat_layer)

# Display figure
fig
```



The above heat map shows the coordinates from the imported dataframe

Create new DataFrame fitting weather criteria

- Narrow down the cities to fit weather conditions.
- Drop any rows will null values.

In [5]:
```
# Checking the length or number of rows in the dataframe ahead of cleaning for mapping
len(raw_data_file_df)
```

Out[5]:
```
579
```

In [6]:
```
# Dropping all NaN cells from the dataframe
clean_weather_data = raw_data_file_df.dropna(how='any')

# Showing counts across all rows, only 5 rows dropped
clean_weather_data.count()
```

Out[6]:
```
City          574
Lat           574
Lng           574
Max Temp      574
Humidity      574
Cloudiness    574
Wind Speed    574
Country       574
Date          574
dtype: int64
```

In [7]:
```
# Sorting data for cities with temperatures between 70 and 80 degrees F
ideal_temp_df = clean_weather_data.loc[(clean_weather_data['Max Temp'] > 70) & (clean_weather_data['Max Temp'] < 80)]

# Presenting first 5 rows of dataframe
ideal_temp_df.head()
```

Out[7]:

	City	Lat	Lng	Max Temp	Humidity	Cloudiness	Wind Speed	Country	Date
8	Atuona	-9.80	-139.03	78.28	77	11	22.91	PF	1608127915
11	Mar del Plata	-38.00	-57.56	72.00	40	0	21.92	AR	1608127635
13	Andapa	-14.65	49.65	79.07	69	33	4.12	MG	1608127916
18	East London	-33.02	27.91	73.40	69	20	9.17	ZA	1608127918
20	Hermanus	-34.42	19.23	73.99	62	47	8.01	ZA	1608127918

In [8]:
```
# Dropping off rows with Wind Speed less than 10 mph
cut_windsp_df = ideal_temp_df.loc[ideal_temp_df['Wind Speed'] < 10]

# Displaying first 5 rows of dataframe
cut_windsp_df.head()
```

Out[8]:

	City	Lat	Lng	Max Temp	Humidity	Cloudiness	Wind Speed	Country	Date
13	Andapa	-14.65	49.65	79.07	69	33	4.12	MG	1608127916
18	East London	-33.02	27.91	73.40	69	20	9.17	ZA	1608127918
20	Hermanus	-34.42	19.23	73.99	62	47	8.01	ZA	1608127918
34	Kandrian	-6.22	149.55	79.18	85	100	6.24	PG	1608127922
35	Puerto Ayora	-0.74	-90.35	71.01	96	63	4.00	EC	1608127922

In [9]:
```
# Dropping off rows with Cloudiness of 0%
cld_zero_df = cut_windsp_df.loc[cut_windsp_df['Cloudiness'] == 0]

# Displaying first 5 rows of dataframe
cld_zero_df.head()
```

Out[9]:

	City	Lat	Lng	Max Temp	Humidity	Cloudiness	Wind Speed	Country	Date
63	Rikitea	-23.12	-134.97	76.55	73	0	9.19	PF	1608127931
93	San Rafael	-34.62	-68.33	79.32	17	0	6.55	AR	1608127939
122	Kruisfontein	-34.00	24.73	78.01	40	0	5.99	ZA	1608127947
235	Port Alfred	-33.59	26.89	73.99	75	0	9.95	ZA	1608127977
251	Nicoya	10.15	-85.45	73.40	94	0	3.36	CR	1608127981

In [10]:
```
# Displaying length or number of rows of dataframe with sorted temperatures of between 70 and 80 F
len(ideal_temp_df)
```

Out[10]: 122

In [11]:
```
# Displaying length or number of rows of dataframe with sorted Wind Speeds of less than 10 mph
len(cut_windsp_df)
```

Out[11]: 78

In [12]:
```
# Displaying length or number of rows of dataframe with sorted zero cloudiness
len(cld_zero_df)
```

Out[12]: 15

In [13]:
```
# Dropping off rows with no numbers or data
hotel_df = cld_zero_df.dropna(how='any')

# Displaying hotel dataframe. 15 cities finalilsts
hotel_df
```

Out[13]:

	City	Lat	Lng	Max Temp	Humidity	Cloudiness	Wind Speed	Country	Date
63	Rikitea	-23.12	-134.97	76.55	73	0	9.19	PF	1608127931
93	San Rafael	-34.62	-68.33	79.32	17	0	6.55	AR	1608127939
122	Kruisfontein	-34.00	24.73	78.01	40	0	5.99	ZA	1608127947
235	Port Alfred	-33.59	26.89	73.99	75	0	9.95	ZA	1608127977
251	Nicoya	10.15	-85.45	73.40	94	0	3.36	CR	1608127981
279	Uitenhage	-33.76	25.40	78.01	50	0	9.17	ZA	1608127967
283	Port Elizabeth	-33.92	25.57	75.20	50	0	9.17	ZA	1608127990
337	Bahir Dar	11.59	37.39	77.59	22	0	7.18	ET	1608128005
379	Gopālpur	19.27	84.92	73.85	71	0	4.79	IN	1608128015
429	Olavarría	-36.89	-60.32	73.00	46	0	4.00	AR	1608128028
432	Trat	12.50	102.50	76.50	82	0	4.85	TH	1608128029
451	Viedma	-40.81	-63.00	72.00	31	0	7.00	AR	1608127826
468	Santa Fe	-31.63	-60.70	75.99	58	0	3.00	AR	1608128039
472	Najrān	17.49	44.13	78.80	20	0	6.93	SA	1608128040
476	Verāval	20.90	70.37	75.60	41	0	8.72	IN	1608128042

In [ ]:
```
```

Hotel Map

- Store into variable named `hotel_df`.
- Add a "Hotel Name" column to the DataFrame.
- Set parameters to search for hotels with 5000 meters.
- Hit the Google Places API for each city's coordinates.
- Store the first Hotel result into the DataFrame.
- Plot markers on top of the heatmap.

In [14]:
```
# Adding "Hotel Name" column to the dataframe
hotel_df["Hotel Name"] = ""
```

In [15]:
```
# Displaying hotel dataframe (first 5 rows)
hotel_df.head()
```

Out[15]:

	City	Lat	Lng	Max Temp	Humidity	Cloudiness	Wind Speed	Country	Date	Hotel Name
63	Rikitea	-23.12	-134.97	76.55	73	0	9.19	PF	1608127931	
93	San Rafael	-34.62	-68.33	79.32	17	0	6.55	AR	1608127939	
122	Kruisfontein	-34.00	24.73	78.01	40	0	5.99	ZA	1608127947	
235	Port Alfred	-33.59	26.89	73.99	75	0	9.95	ZA	1608127977	
251	Nicoya	10.15	-85.45	73.40	94	0	3.36	CR	1608127981	

In [16]:
```
# base url
base_url = "https://maps.googleapis.com/maps/api/place/nearbysearch/json"

# Preparing parameters dataframe with keys and values
target_search = "Hotel"
target_radius = 5000
target_type = "hotel"
params = {"keyword": target_search,"radius": target_radius,"type": target_type,"key": g_key}

# Using FOR loop for populating parameters dictionary
for index, row in hotel_df.iterrows():
    lat = row["Lat"]
    lng = row['Lng']
    params['location'] = f"{lat},{lng}"
    response = requests.get(base_url, params=params)
    if response.status_code == 200:
        hotels_data = response.json()
        print(json.dumps(hotels_data, indent=4, sort_keys=True))

    # Each try to locating parameters of 5000 meters around destination, some hotels go missing, so it shows only successful ones
    try:
        hotel_df.loc[index, "Hotel Name"] = hotels_data["results"][0]["name"]
    except (KeyError, IndexError):
        print("Missing field/result...skipping.")
hotel_df
```


```
{
    "html_attributions": [],
    "results": [
        {
            "business_status": "OPERATIONAL",
            "geometry": {
                "location": {
                    "lat": -23.1276524,
                    "lng": -134.9656596
                },
                "viewport": {
                    "northeast": {
                        "lat": -23.12637127010728,
                        "lng": -134.9643498701072
                    },
                    "southwest": {
                        "lat": -23.12907092989272,
                        "lng": -134.9670495298927
                    }
```


In [17]:
```
# Information for the first hotel in the response from json showing Hotel name, city and Country
print(hotels_data["results"][0]["name"])
print(hotels_data['results'][0]['geometry']['location']['lat'])
print(hotels_data['results'][0]['geometry']['location']['lng'])
print(hotels_data['results'][0]['plus_code']['compound_code'][17:])
```

```
Hotel The Grand Astoria Somnath
20.9359593
70.3559585
Gujarat, India
```

In [18]:
```
# Displaying all the hotels found at about 5000 meters of the coordinates (first 5 rows)
hotel_df.head()
```

Out[18]:

	City	Lat	Lng	Max Temp	Humidity	Cloudiness	Wind Speed	Country	Date	Hotel Name
63	Rikitea	-23.12	-134.97	76.55	73	0	9.19	PF	1608127931	Pension Bianca & Benoit
93	San Rafael	-34.62	-68.33	79.32	17	0	6.55	AR	1608127939	Azalea Luxury Lodge
122	Kruisfontein	-34.00	24.73	78.01	40	0	5.99	ZA	1608127947	Humansdorp Boutique Hotel
235	Port Alfred	-33.59	26.89	73.99	75	0	9.95	ZA	1608127977	Royal St Andrews Hotel
251	Nicoya	10.15	-85.45	73.40	94	0	3.36	CR	1608127981	Hotel Doña Marta

In [19]:
```
# NOTE: Do not change any of the code in this cell

# Using the template add the hotel marks to the heatmap
info_box_template = """
<dl>
<dt>Name</dt><dd>{Hotel Name}</dd>
<dt>City</dt><dd>{City}</dd>
<dt>Country</dt><dd>{Country}</dd>
</dl>
"""
# Store the DataFrame Row
# NOTE: be sure to update with your DataFrame name
hotel_info = [info_box_template.format(**row) for index, row in hotel_df.iterrows()]
locations = hotel_df[["Lat", "Lng"]]
```

In [20]:
```
# Displaying the locations dataframe with Lat and Long data
locations
```

Out[20]:

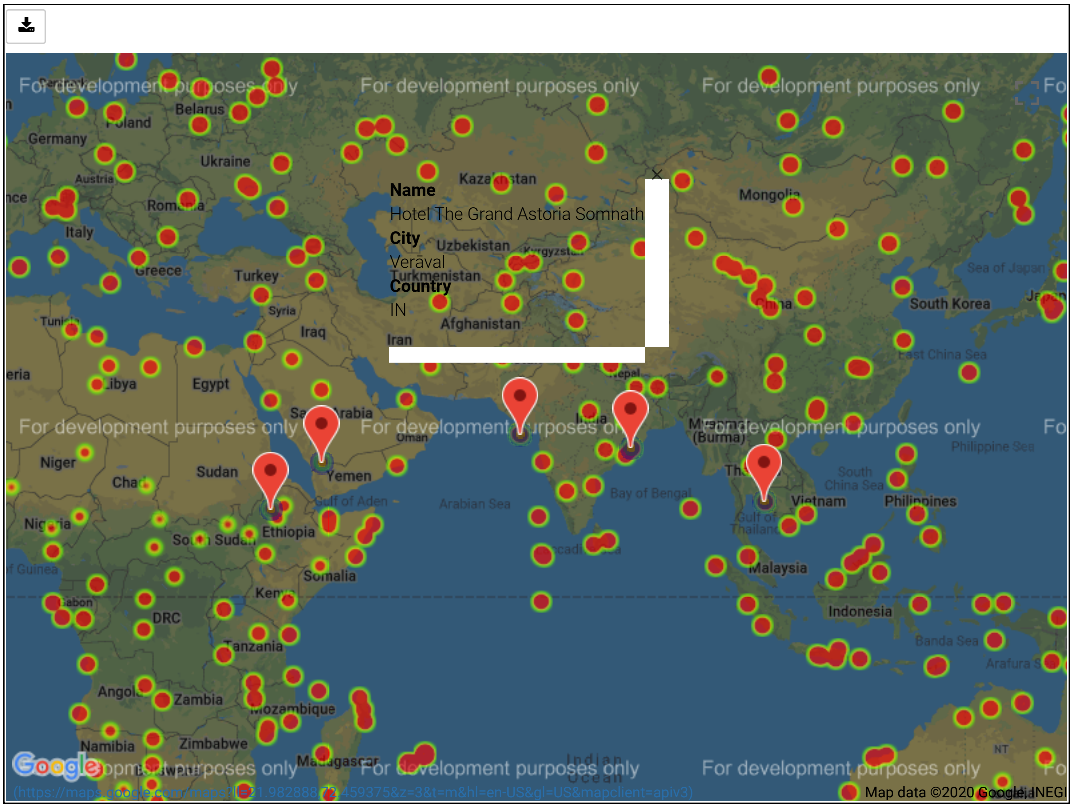
	Lat	Lng
63	-23.12	-134.97
93	-34.62	-68.33
122	-34.00	24.73
235	-33.59	26.89
251	10.15	-85.45
279	-33.76	25.40
283	-33.92	25.57
337	11.59	37.39
379	19.27	84.92
429	-36.89	-60.32
432	12.50	102.50
451	-40.81	-63.00
468	-31.63	-60.70
472	17.49	44.13
476	20.90	70.37

In [21]:
```
# Generating map hotel information
hotel_infor = hotel_df["Hotel Name"].tolist()
```

In [22]:
```
# # Create hotel layer
hotel_layer = gmaps.symbol_layer(
    locations, fill_color='rgba(0, 150, 0, 0.4)',
    stroke_color='rgba(0, 0, 150, 0.4)', scale=6,
    info_box_content=[f"{hotel}" for hotel in hotel_info]
)

# Add the layer to the map
fig = gmaps.figure()
fig.add_layer(heat_layer)
fig.add_layer(hotel_layer)

# Display figure
fig

#
```

The spots in the above map with more colored centers represent the regions with the locations identified in our search

In [23]:
```
# Add marker layer on top of heat map
figure_layout = {
    'width': '800px',
    'height': '600px',
    'border': '1px solid black',
    'padding': '1px',
    'margin': '0 auto 0 auto'
}
# Assign the marker layer to a variable
markers = gmaps.marker_layer(locations, info_box_content=[f"{hotel}" for hotel in hotel_info])

# Add the layer to the map
fig = gmaps.figure(layout=figure_layout, zoom_level=4, center=(20.94,70.35))

# Layering all maps
fig.add_layer(heat_layer)
fig.add_layer(hotel_layer)
fig.add_layer(markers)

# Display figure
fig
```



The above map shows the spot of the first hotel in India. It often changes with every run, but often in India, which would imply they have the best weather and a hotel to help enjoy that weather.

In [ ]:

(
(
(
(
(