VacationPy ¶

Starter Code to Import Libraries and Load the Weather and Coordinates Data

In [1]:
```
# Dependencies and Setup
import pandas as pd
import hvplot.pandas
import requests
import geoviews

# Import API key
from api_keys import geoapify_key
```

In [6]:
```
# Load the CSV file created in Part 1 into a Pandas DataFrame
city_data_df = pd.read_csv("output_data/cities.csv")

# Drop results with null values
city_data_df = city_data_df.dropna()

# Display sample data
city_data_df.head(500)
```

Out[6]:

	City_ID	City	Lat	Lng	Max Temp	Humidity	Cloudiness	Windspeed	Country	Date
0	0	isafjordur	66.0755	-23.1240	33.01	77	80	5.28	IS	1682987513
1	1	traverse city	44.7631	-85.6206	35.67	95	100	16.11	US	1682987467
2	2	west island	-12.1568	96.8225	80.58	83	40	16.11	CC	1682987513
3	3	kinkala	-4.3614	14.7644	71.96	88	100	1.57	CG	1682987513
4	4	westport	41.1415	-73.3579	56.21	73	100	10.36	US	1682987257
...	...	...	...	...	...	...	...	...	...	...
499	499	colchani	-20.3000	-66.9333	50.70	22	0	4.88	BO	1682987600
500	500	bayanhongor	46.7167	100.1167	35.44	26	100	4.36	MN	1682987600
501	501	auki	-8.7676	160.7034	84.43	70	100	4.38	SB	1682987600
502	502	dade city	28.3647	-82.1959	75.29	67	0	8.05	US	1682987600
503	503	bor	56.3567	44.0669	42.73	86	99	7.16	RU	1682987600

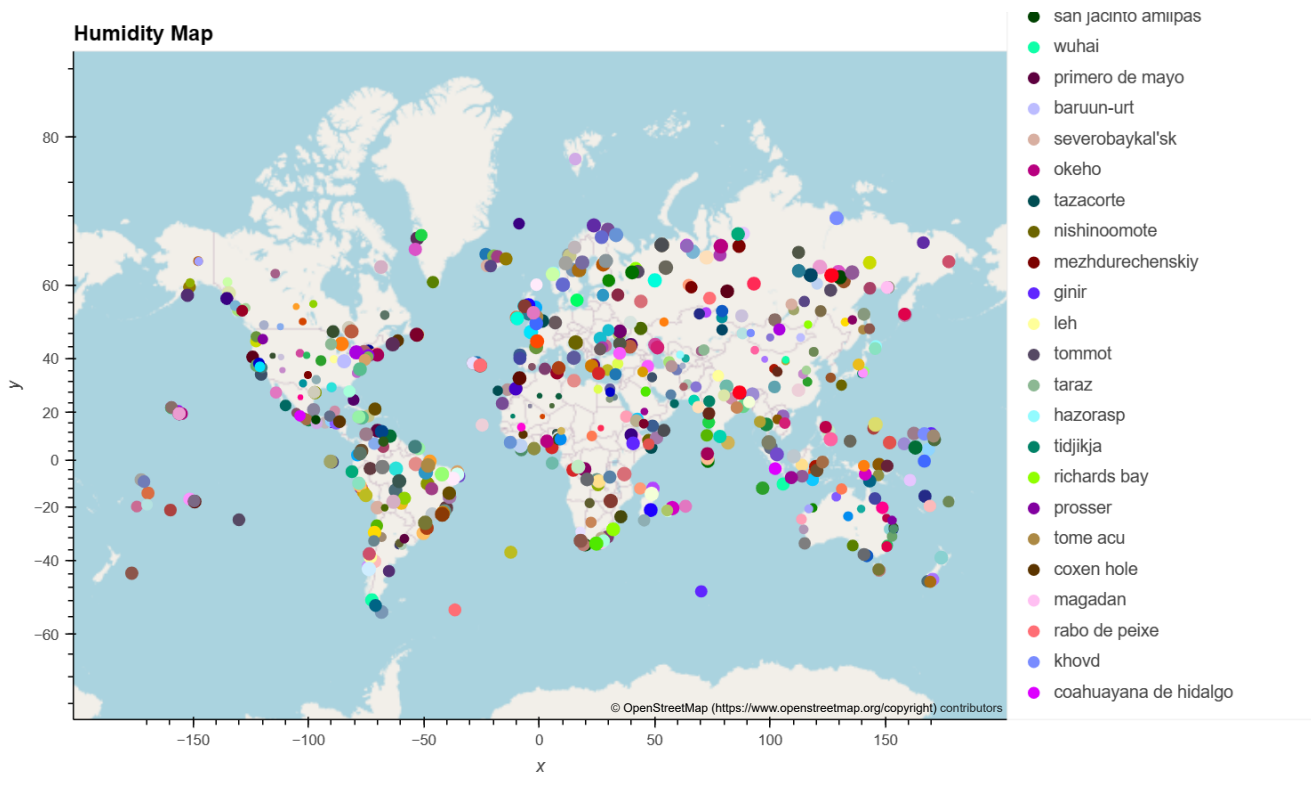
500 rows × 10 columns

Step 1: Create a map that displays a point for every city in the `city_data_df` DataFrame. The size of the point should be the humidity in each city.

In [3]:
```
%%capture --no-display
# Configure the map plot

coordinates = {
    "Lat": (city_data_df["Lat"]),
    "Lng": (city_data_df["Lng"]),
    "City": (city_data_df["City"]),
    "Humidity": (city_data_df["Humidity"])
}

coordinates_df = pd.DataFrame(coordinates)

humidity_map = coordinates_df.hvplot.points(
    "Lng",
    "Lat",
    geo = True,
    tiles = "OSM",
    frame_width = 700,
    frame_height = 500,
    size = "Humidity",
    color = "City",
    title = "Humidity Map"
)
# Save figure
hvplot.save(humidity_map, 'output_data/Humidity_Map.html')
#Show the plot
humidity_map
```

```
WARNING:bokeh.core.validation.check:W-1005 (FIXED_SIZING_MODE): 'fixed' sizing mode requires width and height to be set: Row(id
='1004', ...)
```

Out[3]:



In [9]:
```
# Narrow down cities that fit criteria and drop any results with null values
#A max temperature lower than 27 degrees but higher than 21
ideal_temp = city_data_df.loc[(city_data_df["Max Temp"] < 27) & (city_data_df["Max Temp"] > 21)]
#Wind speed less than 4.5 m/S
ideal_wind = ideal_temp.loc[(city_data_df["Windspeed"] < 10)]
#Cloudiness 0%
ideal_cloud = ideal_wind.loc[(city_data_df["Cloudiness"] < 101)]
# Drop results with null values
ideal_weather= ideal_cloud.dropna()
# Display sample data
ideal_weather.head()
```

Out[9]:

	City_ID	City	Lat	Lng	Max Temp	Humidity	Cloudiness	Windspeed	Country	Date
32	32	tiksi	71.6872	128.8694	22.51	93	100	9.19	RU	1682987519
165	165	salekhard	66.5300	66.6019	24.60	95	100	9.10	RU	1682987541

Step 2: Narrow down the `city_data_df` DataFrame to find your ideal weather condition

Note: The above table are based on the conditions:

A max temperature lower than 27 degrees but higher than 21 Wind speed less than 4.5 m/s Cloudiness less than 95 Reason being, is that if the cloudiness is set to 0, then the table will be blank, since there are no cities with cloudiness of 0.

Step 3: Create a new DataFrame called `hotel_df`.

In [10]:
```
# Use the Pandas copy function to create DataFrame called hotel_df to store the city, country, coordinates, and humidity
hotel_df = ideal_weather[["City", "Country", "Lat", "Lng", "Humidity"]].copy()

# Add an empty column, "Hotel Name," to the DataFrame so you can store the hotel found using the Geoapify API
hotel_df["Hotel Name"] = " "

# Display sample data
hotel_df
```

Out[10]:

	City	Country	Lat	Lng	Humidity	Hotel Name
32	tiksi	RU	71.6872	128.8694	93	
165	salekhard	RU	66.5300	66.6019	95	

Step 4: For each city, use the Geoapify API to find the first hotel located within 10,000 metres of your coordinates.

In [11]:
```
# Set parameters to search for a hotel
radius = 10000
params = {
    "categories": "accommodation.hotel",
    "apiKey":geoapify_key
}

# Print a message to follow up the hotel search
print("Starting hotel search")

# Iterate through the hotel_df DataFrame
for index, row in hotel_df.iterrows():
    # get latitude, longitude from the DataFrame
    lat = row["Lat"]
    lon = row["Lng"]

    # Add filter and bias parameters with the current city's latitude and longitude to the params dictionary
    params["filter"] = f"circle:{lon},{lat},{radius}"
    params["bias"] = f"proximity:{lon},{lat}"

    # Set base URL
    base_url = "https://api.geoapify.com/v2/places"

    # Make and API request using the params dictionary
    name_address = requests.get(base_url, params=params)

    # Convert the API response to JSON format
    name_address = name_address.json()

    # Grab the first hotel from the results and store the name in the hotel_df DataFrame
    try:
        hotel_df.loc[index, "Hotel Name"] = name_address["features"][0]["properties"]["name"]
    except (KeyError, IndexError):
        # If no hotel is found, set the hotel name as "No hotel found".
        hotel_df.loc[index, "Hotel Name"] = "No hotel found"

    # Log the search results
    print(f"{hotel_df.loc[index, 'City']} - nearest hotel: {hotel_df.loc[index, 'Hotel Name']}")

# Display sample data
hotel_df
```

```
Starting hotel search
tiksi - nearest hotel: Арктика
salekhard - nearest hotel: Гостиница
```

Out[11]:

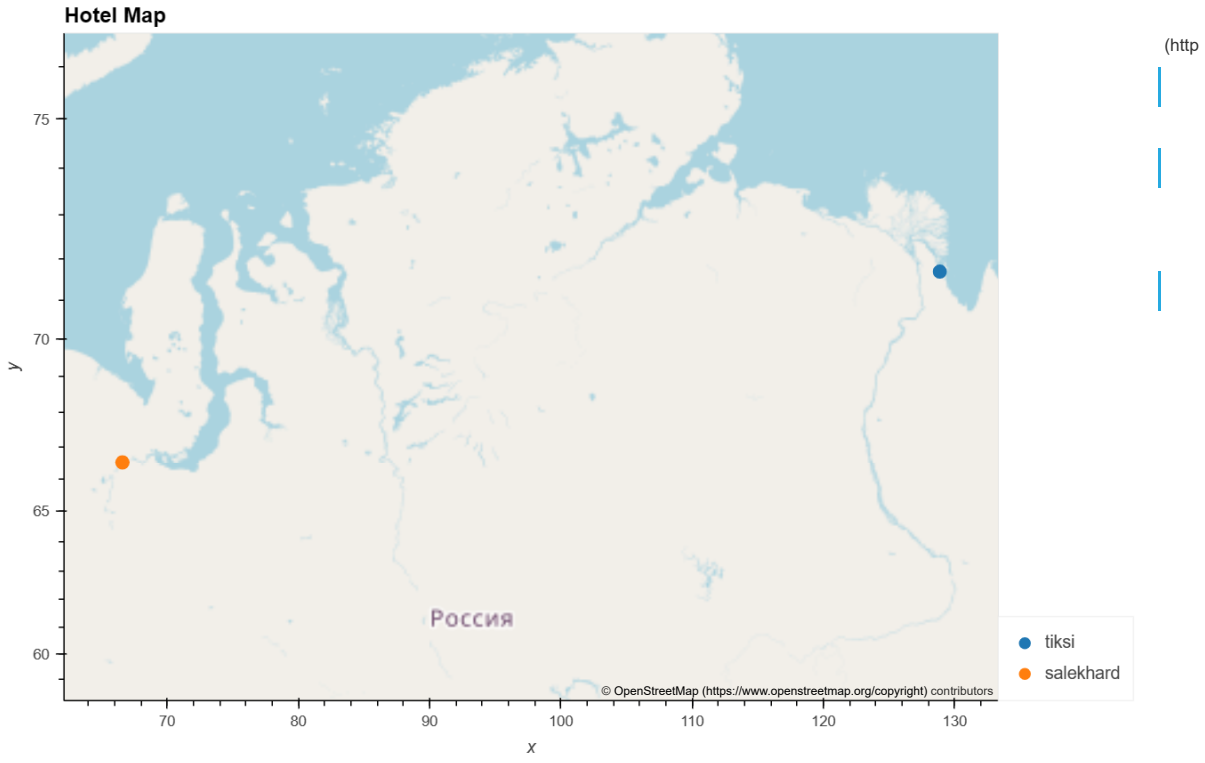
	City	Country	Lat	Lng	Humidity	Hotel Name
32	tiksi	RU	71.6872	128.8694	93	Арктика
165	salekhard	RU	66.5300	66.6019	95	Гостиница

Step 5: Add the hotel name and the country as additional information in the hover message for each city in the map.

In [12]:
```
%%capture --no-display

# Configure the map plot
hotel_map = hotel_df.hvplot.points(
    "Lng",
    "Lat",
    geo = True,
    tiles = "OSM",
    frame_width = 700,
    frame_height = 500,
    size = "Humidity",
    color = "City",
    title = "Hotel Map",
    hover_cols = ["Hotel Name", "Country"]
)
# Save figure
hvplot.save(hotel_map, 'output_data/Hotel_Map.html')
#Show the plot
hotel_map
```

```
WARNING:bokeh.core.validation.check:W-1005 (FIXED_SIZING_MODE): 'fixed' sizing mode requires width and height to be set: Row(id
='1297', ...)
```

Out[12]:



In [ ]:

In [ ]: