Overview

Hi and hope all is well :)

First of all, I would like to thank you for offering me the opportunity to work on this tech task.
It was a very good incentive for me to start looking into data manipulation and visualisation libraries
such as [pandas (https://pandas.pydata.org)](https://pandas.pydata.org), [plotly (https://plotly.com)](https://plotly.com) as well as [jupyter notebook/lab (https://jupyter.org/)](https://jupyter.org/).

I was aware of their existence, but previously have not worked on a project which required their usage,
so this time I decided to take the opportunity and see what these libraries can offer.

Known limitations

Mixture of tools

I had to timebox this exercise so that it does not divert me too much from the Django learning path,
to which I have committed myself earlier this year. I am consciously leaving a mixture of both standard libraries data manipulation code as well as parts of pandas-based code to make it clear that I explored a range of options.

Focus on several hypotheses

We all work in the industry where time is a very important factor.

Very rarely we have the opportunity to spend an unlimited amount of time
to deliver a fully complete and polished project.
To show my appreciation of that, I decided to focus my attention on several hypotheses
which I treated as an [MVP (https://www.agilealliance.org/glossary/mvp)](https://www.agilealliance.org/glossary/mvp), namely:

1. data exploration:
   A. how large is the data set
   B. which timeframe does it cover
   C. structure of records
   D. consistency of data
   E. etc
2. correlations between:
   A. pressure and humidity
   B. temperature and pressure
   C. pressure and wind speed
   D. temperature and lat
   E. temperature and proximity to city centre
3. visualisation of some findings

Git history

I was trying to work on this mini project as much as I could, which meant frequently switching contexts, leaving unworking code and coming back to it at the next available opportunity. As a result I had to commit my changes in bulk and sometimes in unstable state. I could re-write git history, but instead I decided to spend more time working on and documenting that mini project because I see these activities as core business requirement which have higher business value than a pretty git history.

Let's see

Without further adieu, please take a look at below jupyter notebook.

It represents the flow of

1. me exploring the data
2. testing hypotheses
3. conclusions and observations

So, let's see...

In [1]:
```
import pandas as pd
import data_explorer
import numpy as np
import gmaps
import plotly.express as px
from collections import Counter

import render
```

In [2]:
```python
import importlib

importlib.reload(data_explorer)
_ = importlib.reload(render)
```

In [3]:
```python
data_explorer.get_data()
```

Get a feeling of the data and boundaries

In [4]:
```python
records = data_explorer.get_records('./data/weather.json')
```

In [5]:
```python
## What is the shape of data

records[0]
```

Out[5]:
```
{'city': {'id': 14256,
  'name': 'Azadshahr',
  'findname': 'AZADSHAHR',
  'country': 'IR',
  'coord': {'lon': 48.570728, 'lat': 34.790878},
  'zoom': 10},
 'time': 1554462304,
 'main': {'temp': 287.07,
  'pressure': 1022,
  'humidity': 71,
  'temp_min': 284.15,
  'temp_max': 289.15},
 'wind': {'speed': 4.1, 'deg': 340},
 'clouds': {'all': 90},
 'weather': [{'id': 804,
   'main': 'Clouds',
   'description': 'overcast clouds',
   'icon': '04d'}]}
```

In [6]:
```python
## What is the time interval we are dealing with?

start_time, end_time = data_explorer.get_time_interval_from_records(records)
print(f"start time: {start_time}, end time: {end_time}")
```

```
start time: 2019-04-05 12:05:04, end time: 2019-04-05 12:08:26
```

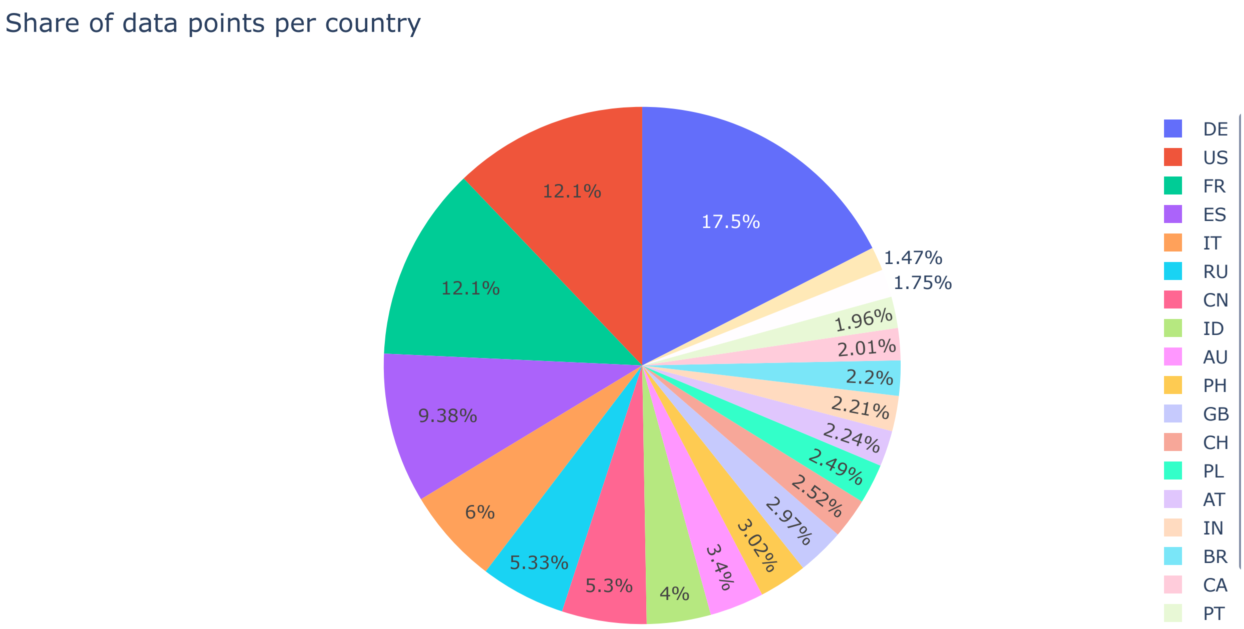
In [7]:
```python
# Observation: a very narrow window, limiting "over time" analysis
```

Distribution of data points by country

In [8]:
```python
countries_counted = Counter((x["city"]["country"] for x in records if len(x["city"]["country"]) >= 2))
countries_df = pd.DataFrame(
    {"countries" : list(countries_counted.keys()),
     "frequencies" : list(countries_counted.values())
    }
)

# only show countries with > 1% of data points
one_percent_threashold = countries_df['frequencies'].sum() / 100
filt = countries_df['frequencies'] > one_percent_threashold
fig = px.pie(countries_df[filt],
             values='frequencies',
             names='countries',
             title='Share of data points per country')

fig.show()
```



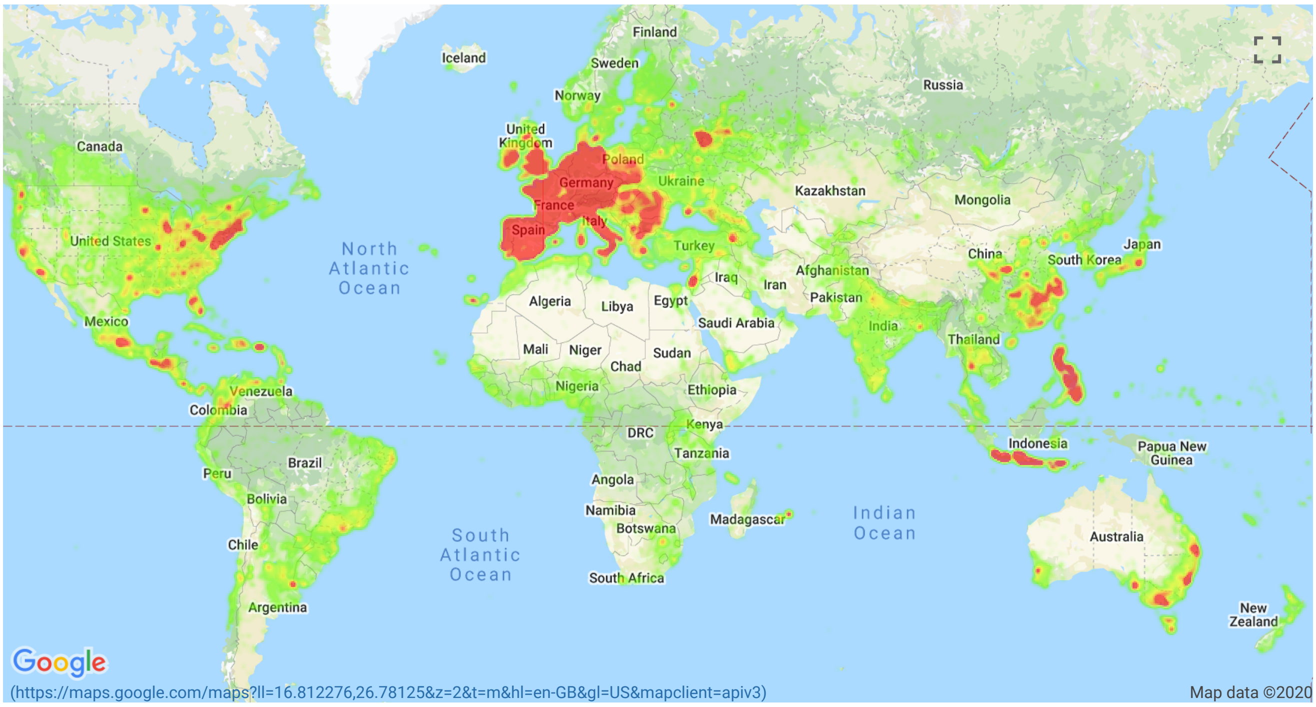
PROS: easy to see breakdown of data points by country as well as relative sizes of each share.
CONS: distribution of data points within each country is not visible

Locations of the data points

In [9]:
```python
coords = data_explorer.get_coords(records)
```

In [10]:
```python
gmaps.configure(api_key=render.get_api_key())
```

In [11]:
```
fig = render.get_data_points_figure(coords)
fig
```



PROS: easy to see location and density of data points.
CONS: dissipate on zoom (solvable by using gmaps.symbol_layer, but it is less visually appealing and loses intensity colouring)

Correlation between pressure and humidity

In [12]:
```
pressure, humidity = data_explorer.get_pressure_to_humidity_pairs(records)
data_explorer.report_correlation(pressure, humidity)
```

Out[12]: `'0.04494809564870188 # negligible uphill correlation'`

Correlation between temperature and pressure

In [13]:
```
temp, pressure = data_explorer.get_temperature_to_pressure_pairs(records)
data_explorer.report_correlation(temp, pressure)
```

Out[13]: `'-0.01819578815160411 # negligible downhill correlation'`

Correlation between pressure and wind speed

In [14]:
```
pressure, wind_speed = data_explorer.get_pressure_to_wind_speed_pairs(records)
data_explorer.report_correlation(pressure, wind_speed)
```

Out[14]: `'-0.11241533790810916 # very weak downhill correlation'`

Correlation between temperature and lat

In [15]:
```
temp, lat = data_explorer.get_temperature_to_lat_pairs(records)
data_explorer.report_correlation(temp, lat)
```

Out[15]: `'-0.5748398480804624 # moderate downhill correlation'`

Correlation between temperature and proximity to city centre

In [16]:
```python
sample_cities = {
    "paris" : {"lat" : 48.862016, "lon" : 2.343970},
    "zurich" : {"lat": 47.377217, "lon" : 8.541865},
    "london" : {"lat": 51.507348, "lon" : -0.127600},
    "manila" : {"lat": 14.662474, "lon" : 120.958869},
}

catchment_area = 30 # radius around city centre
points_near_cities = [point for sample_city_centre in sample_cities.values()
                      for point in data_explorer.get_points_within_distance(
                          sample_city_centre,
                          catchment_area, records)
                     ]
```

In [17]:
```python
temp, distance_from_center, countries = data_explorer.get_temperature_to_distance_triplets(points_near_cities)

temperatures_series = pd.Series(temp)
distance_from_center_series = pd.Series(distance_from_center)
countries_series = pd.Series(countries)

temperatures_series.corr(distance_from_center_series)
# Conclusion: negligible uphill linear correlation
```

Out[17]: 0.04092917490543774

In [18]:
```python
# visualisation

df = pd.DataFrame({
    "distance from city centre" : distance_from_center_series,
    "temprerature" : temperatures_series,
    "country" : countries_series })

fig = px.scatter(df,
                 x="distance from city centre",
                 y="temprerature",
                 color="country",
                 title="Temperature by proximity to city center (km)",
                 width=900)

fig.show() # no evidence of temperatures being higher near to city centre
```