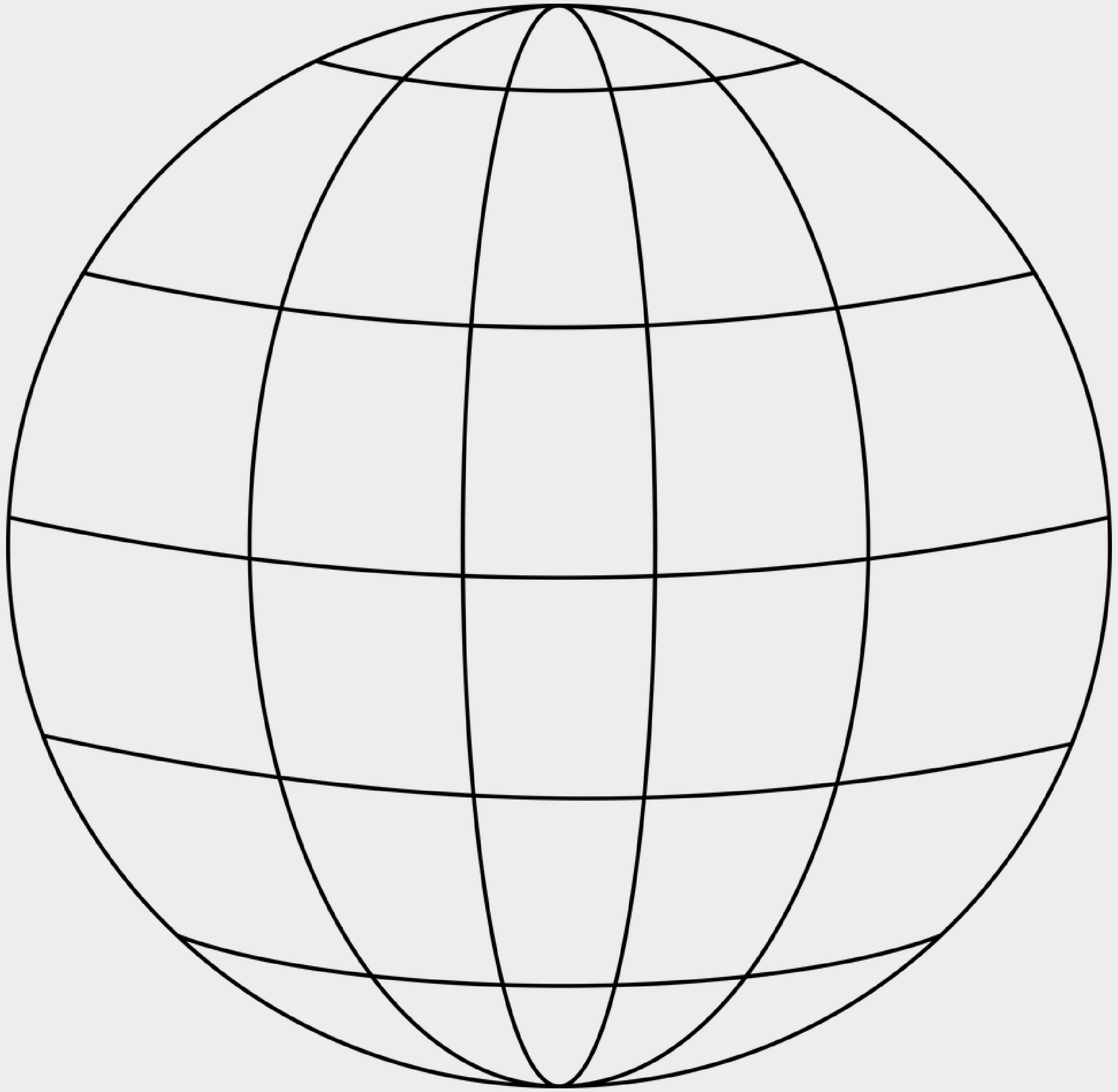
DATA SCIENCE & ANALYTICS

Python API Challenge

Prepared & Presented By:
Toni Edwards, BA, MBA

PYTHON API CHALLENGE

Table of Contents



PART 1:
Challenge Description

PART 2:
Analytical Process

PART 3:
Analysis Review

PART 4:
Conclusion

Challenge Description

WeatherPy

Create a Python script to visualize the weather of 500+ cities across the world of varying distance from the equator. Create a representative model of weather across the world's cities.

Resources

Python Library
OpenWeatherAPI

VacationPy

Create Google Maps displaying the location of Hotels in cities having the most ideal weather across the globe. Add a heat map to display the level of humidity in each of the cities contained in the dataset.

Resources

Python Library
Google Maps

PYTHON API CHALLENGE

Analysis Process

Overview

In this challenge the data was retrieved from API sources and transformed into visual representations such as charts, plots, and maps in order to demonstrate analysis and determination of weather patterns across the globe.

1

Gather Data:
Import libraries Python scripting and building the API queries.

2

Clean Data:
Clean the data result using filters on the JSON object and remove null values.

3

Store Data:
Create data file from the dataset and export to the repository.

4

Analyze Data:
Create the Pandas Data Frame and use statistical libraries for analysis.

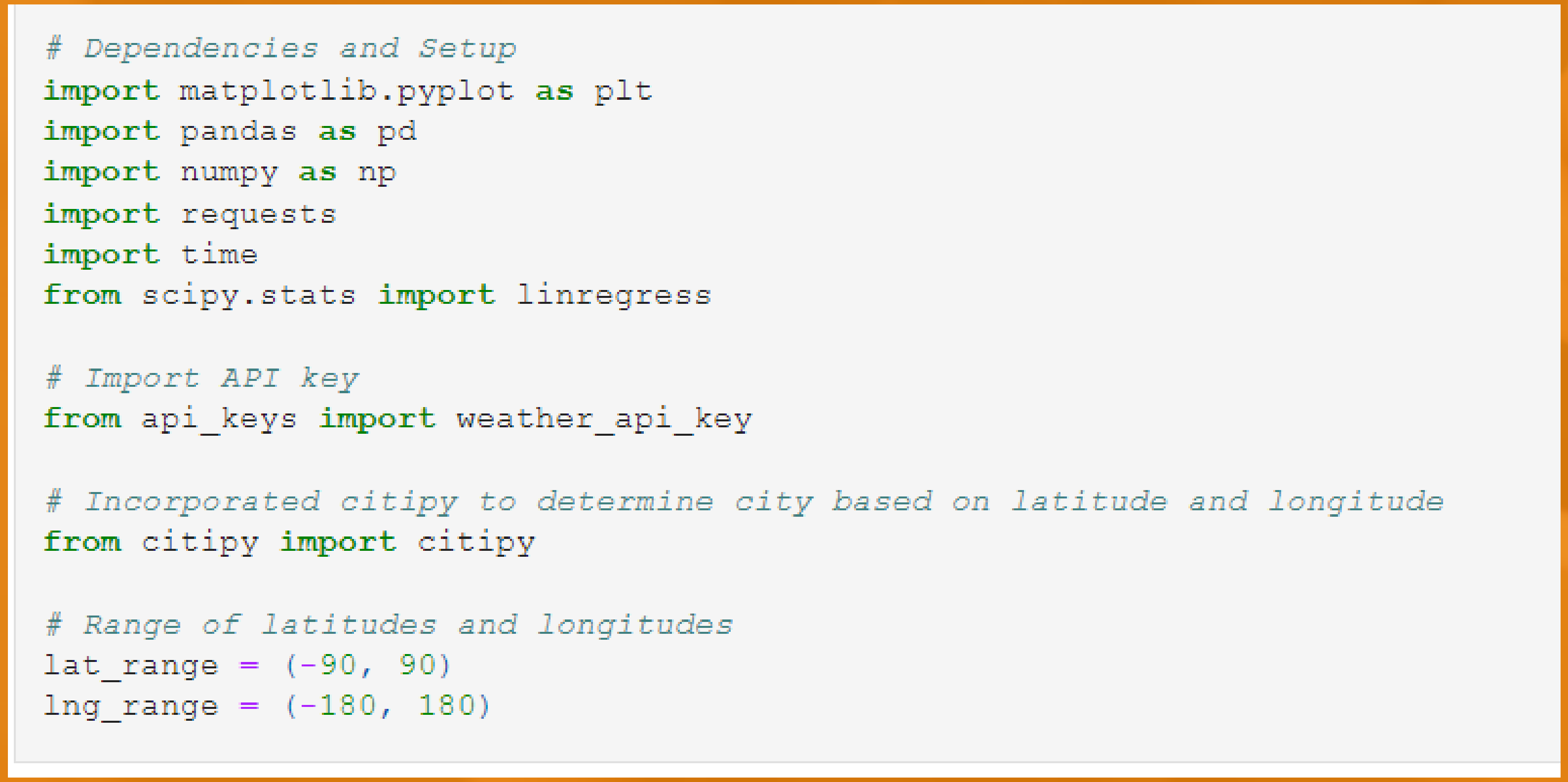
5

Visualize Data:
Create Pandas plots and Google Maps to interpret findings.

PYTHON API CHALLENGE

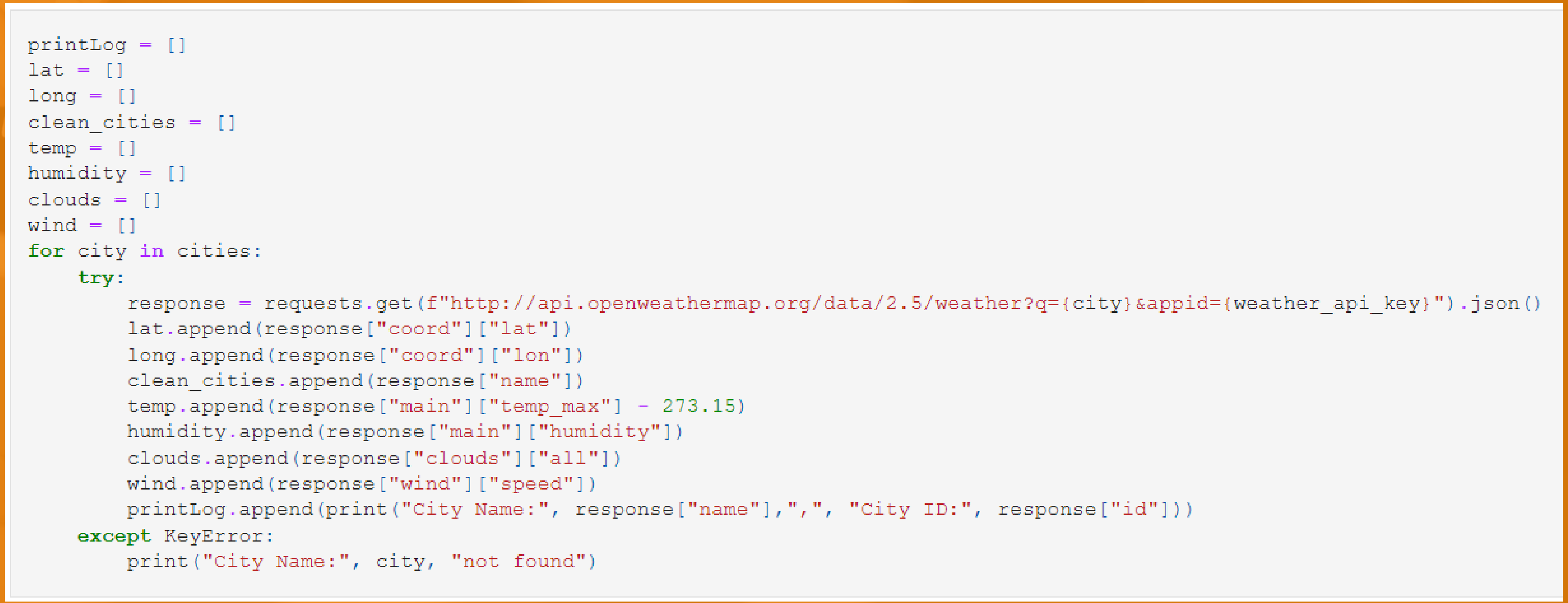
Analysis Review: Gather Data

LIBRARIES & MODULES



```
# Dependencies and Setup
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import requests
import time
from scipy.stats import linregress

# Import API key
from api_keys import weather_api_key

# Incorporated citipy to determine city based on latitude and longitude
from citipy import citipy

# Range of latitudes and longitudes
lat_range = (-90, 90)
lng_range = (-180, 180)
```

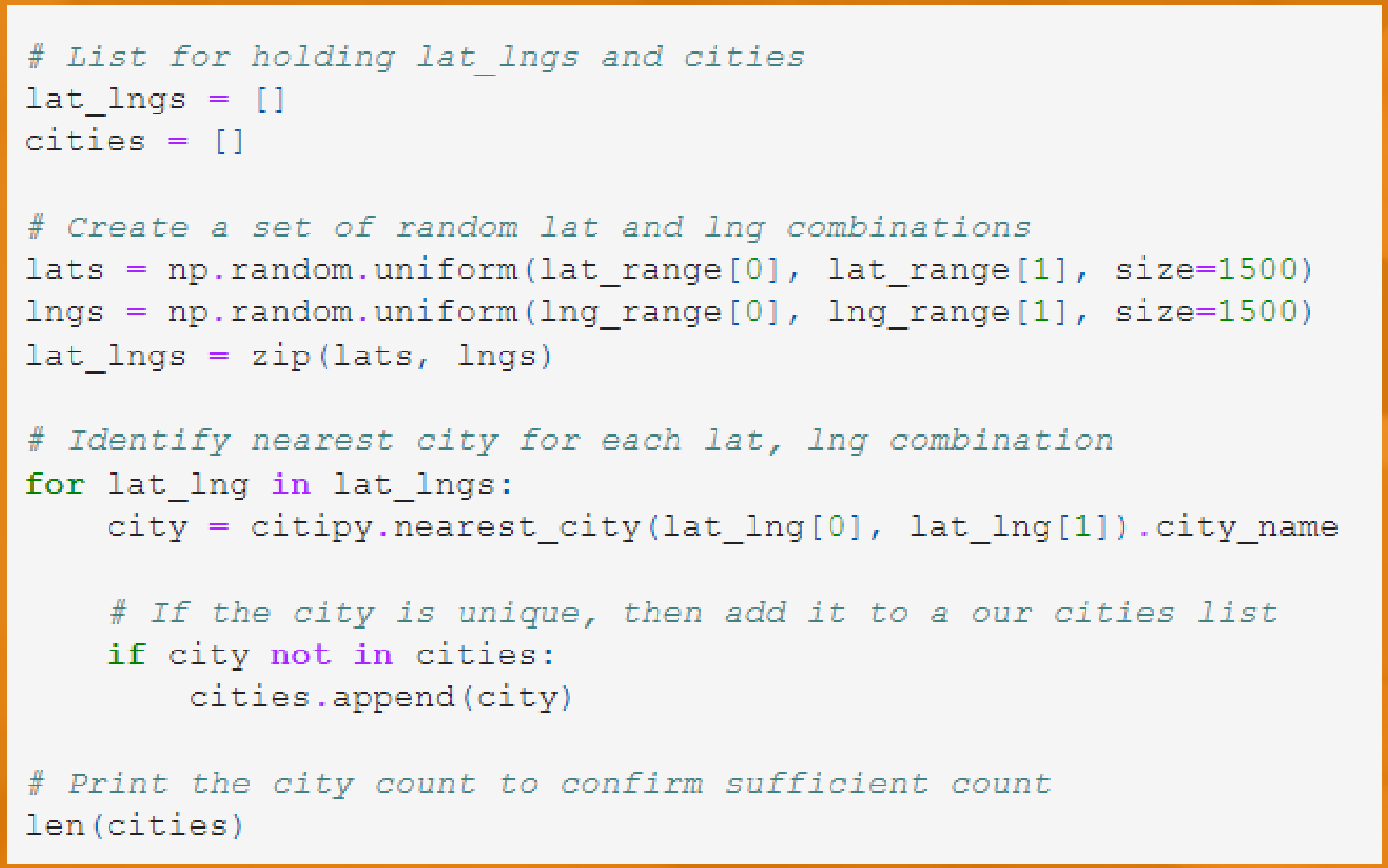
API REQUESTS



```
printLog = []
lat = []
long = []
clean_cities = []
temp = []
humidity = []
clouds = []
wind = []
for city in cities:
    try:
        response = requests.get(f"http://api.openweathermap.org/data/2.5/weather?q={city}&appid={weather_api_key}").json()
        lat.append(response["coord"]["lat"])
        long.append(response["coord"]["lon"])
        clean_cities.append(response["name"])
        temp.append(response["main"]["temp_max"] - 273.15)
        humidity.append(response["main"]["humidity"])
        clouds.append(response["clouds"]["all"])
        wind.append(response["wind"]["speed"])
        printLog.append(print("City Name:", response["name"],",", "City ID:", response["id"]))
    except KeyError:
        print("City Name:", city, "not found")
```

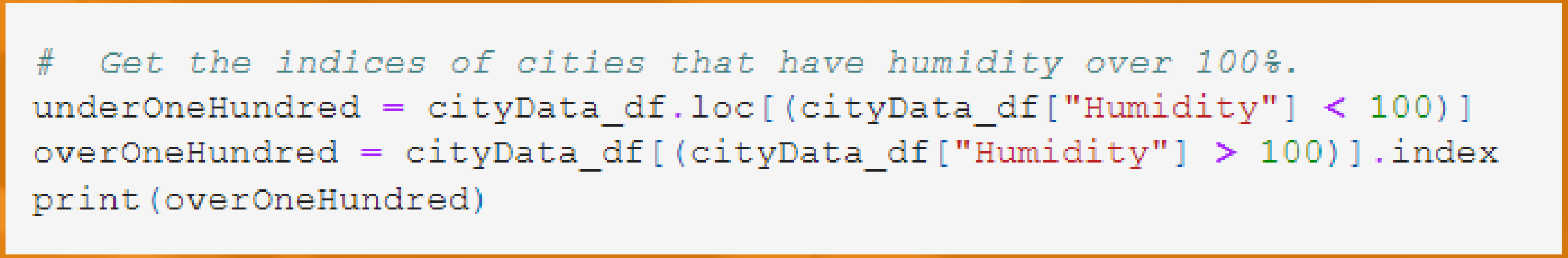
Analysis Review: Clean Data

LISTS & LOOPS



```python
# List for holding lat_lngs and cities
lat_lngs = []
cities = []

# Create a set of random lat and lng combinations
lats = np.random.uniform(lat_range[0], lat_range[1], size=1500)
lngs = np.random.uniform(lng_range[0], lng_range[1], size=1500)
lat_lngs = zip(lats, lngs)

# Identify nearest city for each lat, lng combination
for lat_lng in lat_lngs:
    city = citipy.nearest_city(lat_lng[0], lat_lng[1]).city_name

    # If the city is unique, then add it to a our cities list
    if city not in cities:
        cities.append(city)

# Print the city count to confirm sufficient count
len(cities)
```

FILTERING



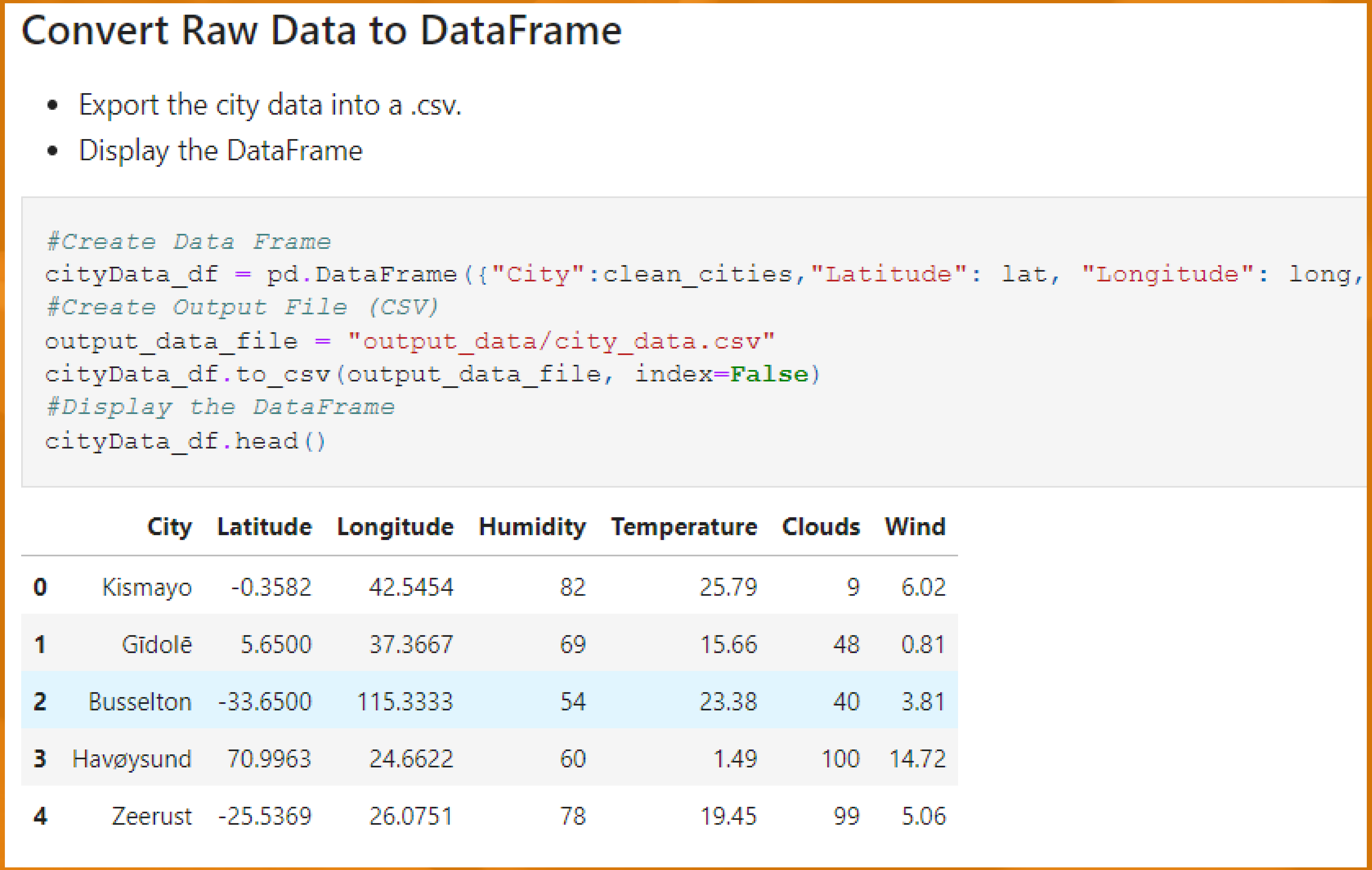
```python
#  Get the indices of cities that have humidity over 100%.
underOneHundred = cityData_df.loc[(cityData_df["Humidity"] < 100)]
overOneHundred = cityData_df[(cityData_df["Humidity"] > 100)].index
print(overOneHundred)
```

PYTHON API CHALLENGE

Analysis Review: Store Data

CSV EXPORT →

DATA FRAME →



Convert Raw Data to DataFrame

- Export the city data into a .csv.
- Display the DataFrame

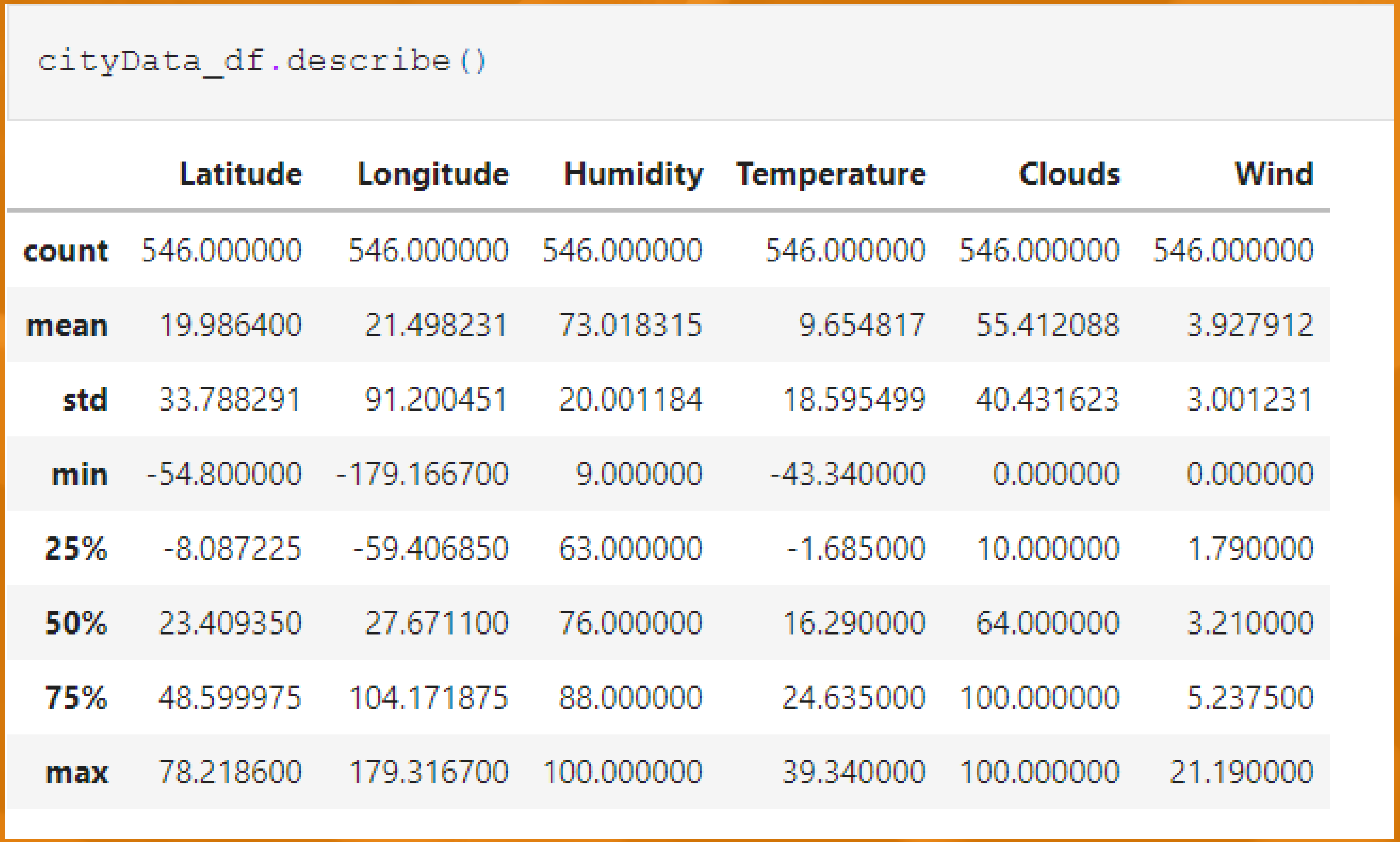
```
#Create Data Frame
cityData_df = pd.DataFrame({"City":clean_cities,"Latitude": lat, "Longitude": long,
#Create Output File (CSV)
output_data_file = "output_data/city_data.csv"
cityData_df.to_csv(output_data_file, index=False)
#Display the DataFrame
cityData_df.head()
```

	City	Latitude	Longitude	Humidity	Temperature	Clouds	Wind
0	Kismayo	-0.3582	42.5454	82	25.79	9	6.02
1	Gīdolē	5.6500	37.3667	69	15.66	48	0.81
2	Busselton	-33.6500	115.3333	54	23.38	40	3.81
3	Havøysund	70.9963	24.6622	60	1.49	100	14.72
4	Zeerust	-25.5369	26.0751	78	19.45	99	5.06

PYTHON API CHALLENGE

Analysis Review: Analyze Data

STATISTICS



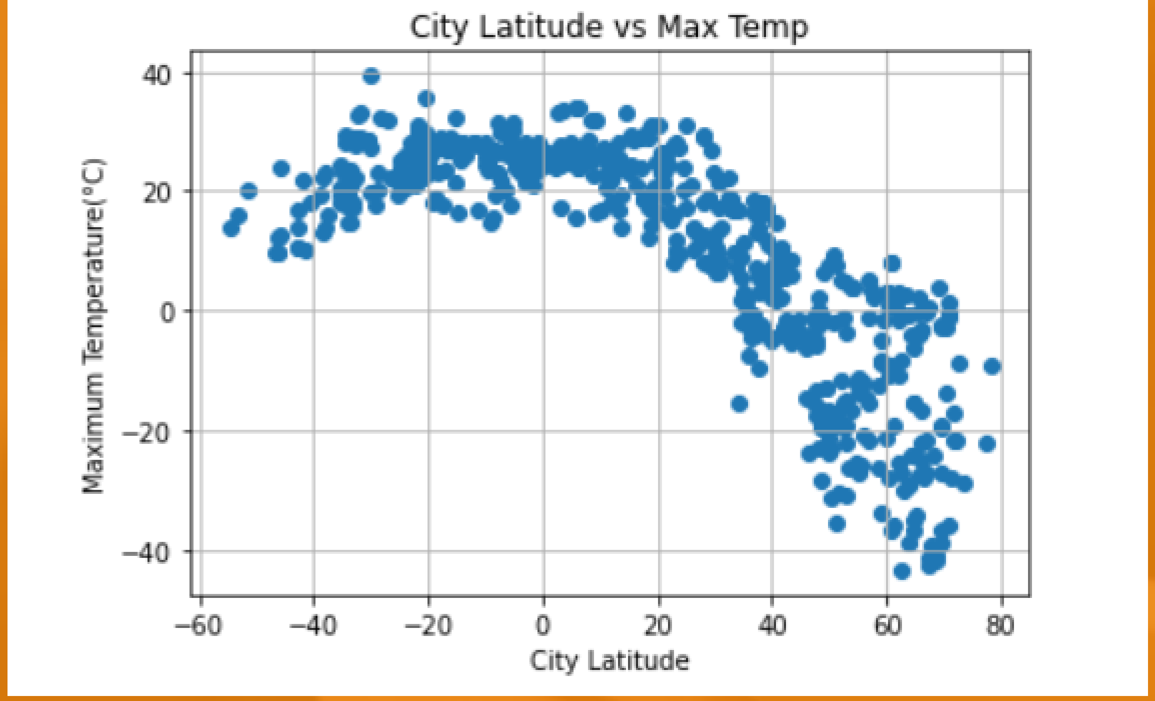
```
cityData_df.describe()
```

	Latitude	Longitude	Humidity	Temperature	Clouds	Wind
count	546.000000	546.000000	546.000000	546.000000	546.000000	546.000000
mean	19.986400	21.498231	73.018315	9.654817	55.412088	3.927912
std	33.788291	91.200451	20.001184	18.595499	40.431623	3.001231
min	-54.800000	-179.166700	9.000000	-43.340000	0.000000	0.000000
25%	-8.087225	-59.406850	63.000000	-1.685000	10.000000	1.790000
50%	23.409350	27.671100	76.000000	16.290000	64.000000	3.210000
75%	48.599975	104.171875	88.000000	24.635000	100.000000	5.237500
max	78.218600	179.316700	100.000000	39.340000	100.000000	21.190000

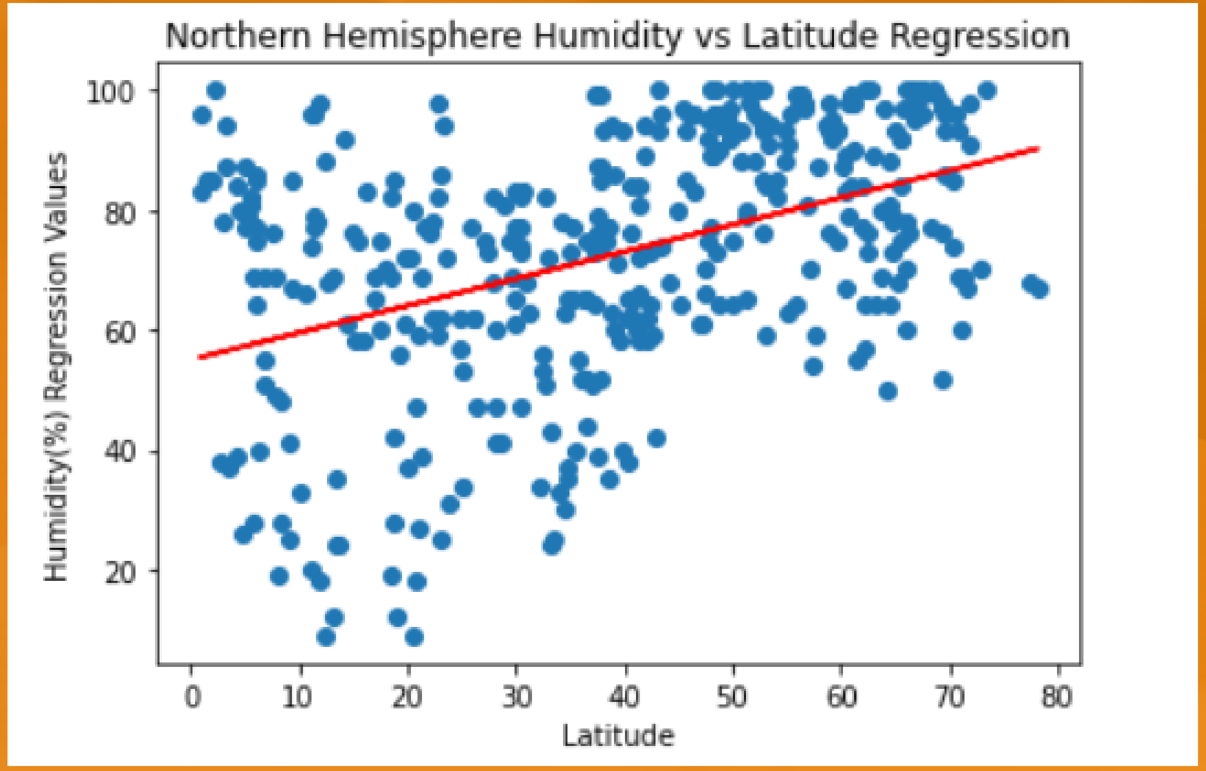
PYTHON API CHALLENGE

Analysis Review: Visualize Data

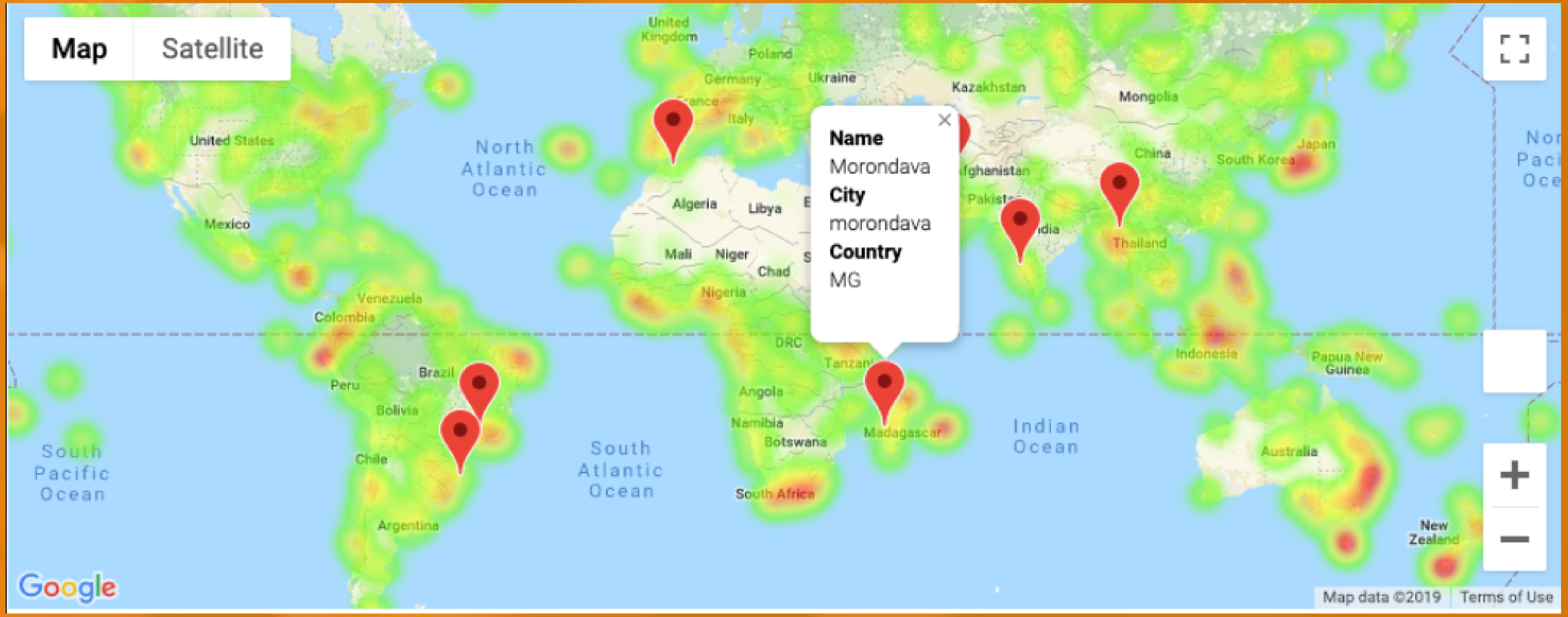
PLOTS



LINEAR REGRESSION



MAPS



Name
Morondava
City
morondava
Country
MG

Python API Challenge

CONCLUSION

Weather patterns differ in the Northern and Southern Hemispheres.

The statistical and visual analysis proves that Southern Hemisphere cities have higher max temperatures in comparison to cities located in the Northern Hemisphere.

Cities located in the Northern Hemisphere and further away from the equator have the highest wind speeds.

Cloud coverage has even distribution % amongst cities in both hemispheres.

Although humidity does increase in cities further away from the equator, there are no cities which have 100% humidity index.