Module 6 Challenge:

The “Python\_API\_Challenge” repository contains the following folders and files:

* WeatherPy folder: Contains all analysis Jupyter Notebooks, an output folder with CSV files, and output graphs.
* Readme file.
* Gitignore with updates.

Part 1: WeatherPy

Used the starter code to generate random geographic coordinates and identify the nearest city to each latitude and longitude combination.

Creating Plots to Showcase the Relationship Between Weather Variables and Latitude

To fulfill the first requirement, we used the OpenWeatherMap API to retrieve weather data for the cities list generated in the starter code. We created a series of scatter plots to showcase the following relationships:

* Latitude vs. Temperature
* Latitude vs. Humidity
* Latitude vs. Cloudiness
* Latitude vs. Wind Speed
* Computing Linear Regression

To fulfill the second requirement, we computed the linear regression for each relationship by separating the plots into the Northern Hemisphere (greater than or equal to 0 degrees latitude) and the Southern Hemisphere (less than 0 degrees latitude). We defined a function to create the linear regression plots.

Next, we created a series of scatter plots for the following variables, including the linear regression line, the model's formula, and the r2 values. The plots created were:

* Northern Hemisphere: Temperature (C) vs. Latitude
* Southern Hemisphere: Temperature (C) vs. Latitude
* Northern Hemisphere: Humidity (%) vs. Latitude
* Southern Hemisphere: Humidity (%) vs. Latitude
* Northern Hemisphere: Cloudiness (%) vs. Latitude
* Southern Hemisphere: Cloudiness (%) vs. Latitude
* Northern Hemisphere: Wind Speed (m/s) vs. Latitude
* Southern Hemisphere: Wind Speed (m/s) vs. Latitude

After each pair of plots, summaries are provided to explain what the linear regression is modelling.

Part 2: VacationPy

In this task, we used Jupyter Notebooks, the GeoViews Python library, and the Geoapify API to plan future vacations using weather data skills.

We used the starter code provided, along with the CSV file containing the weather and coordinates data for each city created in Part 1.

Creating a Map of Cities

We created a map that displays a point for every city in the city\_data\_df DataFrame, with the size of the point representing the humidity in each city.

Narrowing Down for Ideal Weather Conditions

We narrowed down the DataFrame to find the ideal weather conditions as follows and created a new DataFrame called Hotel\_df:

* A maximum temperature lower than 27 degrees but higher than 21
* Wind speed less than 4.5 m/s
* Zero cloudiness

For each city, we used the Geoapify API to find the first hotel located within 10,000 meters of our coordinates. We then added the hotel name and the country as additional information in the hover message for each city on the map.