##Module 6 Notes

##Overview

(As captured from Data Bootcamp Module 6)

In this module, as assigned Data Analyst, I will analysis, visualization, and statistical skills by retrieving and analyzing weather data for a hypothetical travel company, PlanMyTrip. Successfully completing the tasks will draw on your knowledge of Python, decision and repetition statements, data structures, Pandas, and Matplotlib.

By the end of this module, as assigned Data Analyst, I will complete the following::

* Perform tasks using new Python libraries and modules.
* Retrieve and use data from an API "get" request to a server.
* Retrieve and store values from a JSON array.
* Use try and except blocks to resolve errors.
* Write Python functions.
* Create scatter plots using the Matplotlib library, and apply styles and features to a plot.
* Perform linear regression, and add regression lines to scatter plots.
* Create maps by using GeoViews and the Geoapify API.

**Purpose**

(This module is built around a project that mirrors a real-world scenario that would require data analysis and visualization. In module6 , I will practice your analysis, visualization, and statistical skills by retrieving and analyzing weather data for a hypothetical travel company, PlanMyTrip.)

As a Data Analyst, I am helping Jack, head of analysis for the **PLANMYTRIP** user interface team. **PLANMYTRIP** is a top travel technology company that specializes in internet-related services in the hotel and lodging industry. Jack is the head of analysis for the user interface team. He's asked you to help him collect and present data for customers via the search page, which they will then filter based on their preferred travel criteria in order to find their ideal hotel, anywhere in the world.

Jack, the client, has asked I help him collect and present data for customers via the search page, which they will then filter based on their preferred travel criteria in order to find their ideal hotel, anywhere in the world. As part of my analysis, I will need to perform statistical calculations on the data and the weather parameters in the Northern and Southern hemispheres.

Next as assigned Data Analyst, I will export the data, clean it, and use the weather data to choose the best cities for vacation based on certain weather criteria, and then map these cities using GeoViews and the Geoapify API.

In addition, I will practice my analysis, visualization, and statistical skills by retrieving and analyzing weather data for a hypothetical travel company, PlanMyTrip. Successfully completing the tasks will draw on my knowledge of Python, decision and repetition statements, data structures, Pandas, and Matplotlib.

##Approach

**Basic Project Plan**

Here's an outline of your project plan:

* **Task:** Collect and analyze weather data across cities worldwide.
* **Purpose:** PlanMyTrip will use the data to recommend ideal hotels based on clients' weather preferences.
* **Method:** Create a Pandas DataFrame with 500 or more of the world's unique cities and their weather data in real time. This process will entail collecting, analyzing, and visualizing the data.

Analysis of the data will be split into three main parts, or stages.

1. **Collect the Data**
2. **Exploratory Analysis with Visualization**
3. **Visualize Travel Data**

##Results

#### #Deliverable 1: Retrieve Weather Data

For this deliverable, you'll generate a set of 2,000 random latitudes and longitudes, retrieve the nearest city, and perform an API call with OpenWeatherMap. In addition to the city weather data you gathered in this module, use your API skills to retrieve the current weather description for each city. Then, create a new DataFrame containing the updated weather data.

1. Create a folder called Weather\_Database to save all the files related with this deliverable.

GitHub Location: https://github.com/LouFoster/Weather\_Database.git

1. Save the Weather\_Database\_starter\_code.ipynb starter code to the Weather\_Database folder and rename it as Weather\_Database.ipynb.
2. Use the np.random.uniform function to generate a new set of 2,000 random latitudes and 2,000 longitudes.

Text

Description automatically generated

1. Use the citipy module to get the nearest city for each latitude and longitude combination.

Graphical user interface, text, application

Description automatically generated

1. Import your OpenWeatherMap's API key and assemble the API call URL as a string variable. Recall to edit the config.py file to add your API key; also, it's critical to avoid publishing your API key on your GitHub repository.

Graphical user interface, text, application

Description automatically generated

1. Retrieve the following information from the API call:
   * Latitude and longitude
   * Maximum temperature
   * Percent humidity
   * Percent cloudiness
   * Wind speed
   * Weather description (for example, clouds, fog, light rain, clear sky)

Graphical user interface, text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

1. Add the weather data to a new DataFrame.
   * Before continue to the next step, take a moment to confirm that the DataFrame looks similar to the image below (note that cities and countries may vary):

Graphical user interface

Description automatically generated

1. Export the DataFrame as a CSV file, and save it as WeatherPy\_Database.csv in the Weather\_Database folder.

Graphical user interface, text, application

Description automatically generated

#### #Deliverable 2: Create a Customer Travel Destinations Map

In this deliverable, you'll employ input statements to retrieve customer weather preferences. Next, you'll use those preferences to identify potential travel destinations and nearby hotels. Finally, you'll show those destinations on a map.

1. Create a folder called Vacation\_Search to save all the files related with this deliverable.
2. Download the Vacation\_Search\_starter\_code.ipynb Jupyter notebook,save it into your Vacation\_Search folder, and rename it as Vacation\_Search.ipynb.
3. In the Vacation\_Search.ipynb file, ensure that the dependencies and the Geoapify API key is imported correctly.

Graphical user interface, text, application, email

Description automatically generated

1. From the Weather\_Database folder you created in the "Deliverable 1," import the WeatherPy\_Database.csv file as a Pandas DataFrame named city\_data\_df.

Graphical user interface, text

Description automatically generated with medium confidence

1. Write two input statements that prompt the user to enter their minimum and maximum temperature criteria for their vacation.

A picture containing application

Description automatically generated

1. Create a new Pandas DataFrame by using the loc Pandas method to filter the city\_data\_df DataFrame for temperature criteria collected. Name the DataFrame as preferred\_cities\_df.

Graphical user interface, application

Description automatically generated with medium confidence

1. Create a new Pandas DataFrame named clean\_travel\_cities by using the Pandas dropna function on the preferred\_cities\_df to drop any empty rows.

Graphical user interface, text, application, email

Description automatically generated

1. Use the copy Pandas function to create a new DataFrame, called hotel\_df, by copying the following columns from the clean\_travel\_cities DataFrame: "City", "Country", "Max Temp", "Current Description", "Lat", "Lng".

Graphical user interface, application

Description automatically generated

1. Add a new empty column named Hotel Name to the hotel\_df DataFrame.

Graphical user interface

Description automatically generated

1. Review the hotel search parameters provided. These parameters are the same we used in this module; you'll use them to search for a hotel for each city.

Graphical user interface, text, application, email

Description automatically generated

1. Use a for loop to iterate through the hotel\_df DataFrame. Retrieve the latitude and longitude of each city and use the Geoapify API to find the nearest hotel based on the search parameters provided, then add the hotel name to the hotel\_df DataFrame. If a hotel isn't found, set the hotel name as "No hotel found".

Graphical user interface, text, application

Description automatically generated

1. Drop any rows in the hotel\_df DataFrame where a hotel name is not found and store the resulting data into a new DataFrame named clean\_hotel\_df.
   * Take a moment to confirm that your clean\_hotel\_df DataFrame looks similar to the image below (note that the data may vary).

Table

Description automatically generated with medium confidence

1. Create an CSV file to store the clean\_hotel\_df DataFrame as WeatherPy\_vacation.csv in the Vacation\_Search folder.

Graphical user interface, text, application

Description automatically generated

1. Use GeoViews to create a map that displays a point for every city in the clean\_hotel\_df DataFrame. In the point for each city add:
   * The city name
   * The country code
   * The weather description
   * The point's size should be the maximum temperature for the city

Graphical user interface

Description automatically generated

1. Take a screenshot of your map and save it to the Vacation\_Search folder as WeatherPy\_vacation\_map.png.
   * The map should look similar to the following image:

Graphical user interface, application

Description automatically generated

#### ##Deliverable 3: Create a Travel Itinerary Map

For this deliverable, you'll use the Geoapify Routing API to create a travel itinerary that shows the route between four cities chosen from the customer’s possible travel destinations. Then, you'll create a map with a pop-up marker for each city on the itinerary.

1. Create a folder called Vacation\_Itinerary to store all the files for this deliverable.
2. Download the Vacation\_Itinerary\_starter\_code.ipynb file into your Vacation\_Itinerary folder and rename it Vacation\_Itinerary.ipynb.
3. Make sure the initial dependencies and the Geoapify API key are imported.

Graphical user interface, text, application, email

Description automatically generated

1. From your Vacation\_Search folder from Deliverable 2, import the WeatherPy\_vacation.csv file as a DataFrame named vacation\_df.

Text

Description automatically generated with low confidence

1. Use GeoViews to create a map that shows all the cities in the vacation\_df DataFrame. Configure the map as follows:
   * The point's size should be the maximum temperature for the city
   * The point's color should be the city's name
   * Use the hover\_cols parameter to the the "Hotel Name", "Country", and "Current Description" columns to each point as additional information.

Graphical user interface

Description automatically generated

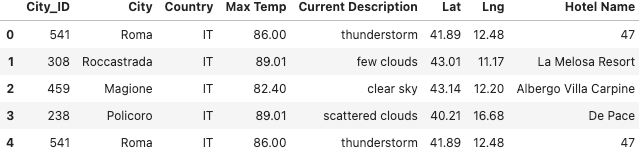
1. From the map, *choose four cities* that a customer might want to visit. They should be close together and in the same country. Use the loc method to create separate DataFrames for each city on the travel route.

Text

Description automatically generated

**HINT**

1. Use the Pandas concat function to merge the DataFrame from each city in the itinerary to create a new DataFrame named itinerary\_df to store the itinerary details. Your final DataFrame should look as in the image below.



Graphical user interface, application

Description automatically generated

1. Use the Pandas copy function to create a new DataFrame named waypoints\_df to store the longitude and latitude for each city in itinerary\_df.

Graphical user interface, text, application

Description automatically generated

1. Use GeoViews to create a map that shows the four cities in the itinerary. You may end with a map similar to the image below.

Graphical user interface, application

Description automatically generated

1. Next, you'll use the Geoapify Routing API to find a route between the cities in the itinerary. Review the code that sets the initial parameters and fetches the coordinates from each city to define the waypoints parameter by using a for loop.

Graphical user interface, text, application, email

Description automatically generated

1. Use the Geoapify Routing API to retrieve the route's directions for your itinerary.

Graphical user interface, text

Description automatically generated

1. From the JSON response, store the route's legs coordinates in a variable called legs.

A picture containing website

Description automatically generated

1. Loop through the route legs coordinates to fetch the latitude and longitude for each step. Store the latitude and longitude values into two Python lists named longitude and latitude.
2. Use the longitude and latitude Python lists to create a new DataFrame named route\_df.

Graphical user interface, text, application, email

Description automatically generated

1. Use the GeoViews Path function to configure a line plot by using route\_df. Set a custom color and width for the line that may contrast with the map.
2. Use the asterisk operator to display a composed plot that shows the itinerary's route over the map containing the cities. Your map may look as in the following image.