2.2: Project Planning and Sourcing Data with an API

PART 1:

1. In order to create your Dashboard get a list of the elements that you would like to evaluate.

   From the bike city service, I will be using the following data:

   - **Trip Details:** `started_at`, `ended_time`, and `trip_duration`.
   - **Station Information:** `from_station_name` and `to_station_name`.
   - **User Demographics:** `member`, .

   From the NOAA API, I will use the `TAVG` (average temperature), as the goal is to understand how weather fluctuations might impact this city bike service. Based on the results, I aim to reinforce the infrastructure and support for high-demand stations.

2. Write down some questions to guide your analysis in a new word-processing document and explain how you intend to visualize the result to answer each of your questions.

   Bike Trip Analysis

   - Analyze the average, maximum, and minimum trip durations.

   Station Analysis

   - Identify the most frequently used stations and the least used stations.

   User Demographics

   - Examine whether users are subscribers or new users.

- Determine which stations have the highest concentration of new users.
- Provide a gender breakdown of users.
- Calculate users' ages based on their birth year.

Weather and Usage Correlation

- The bike city data spans a 12-month period. Analyze bike usage trends using NOAA weather data from LaGuardia Airport, focusing on understanding how average temperature might impact service usage

PART 2: Gather and merge the data

1. In a new notebook, import all necessary libraries, read in your data, and join it. Hint: what's the most effective way to import and join data in such a format.

Created a **shortcut** to access the folder:

```
folderpath = r"C:\Users\carol\Documents\Data Specialization\nyc_noaa_analysis"
```

Then using a **list comprehension**, that is a method that returns a list of all the files in the folderpath directory:

```
filepaths = [os.path.join(folderpath,name) for name in os.listdir(folderpath)]
```

The new CityBikes data had 36 csv files divided by month and each month had a different number of files.

# Join all the csv files using a **generator** with function pd.concat joining the data vertically (no need for a key)

```
df = pd.concat((pd.read_csv(f) for f in filepaths),ignore_index=True)
```

This method is efficient as it requires less code. And as all the files follow the same format the joining will be consistent across all the files.