Project: EXPLORATORY DATA ANALYSIS

INTRODUCTION

This project is to perform an exploratory data anaysis for the `10 academy week 0` challenge. The analysis will utlize by using Solar radiation measurement data. The dataset contains the values for solar radiation, air temperature, relative humidity, barometric pressure, precipitation, wind speed, and wind direction, cleaned and soiled radiance sensor (soiling measurement) and cleaning events.

I will use `python libraries` to do the the exploratory data analysis. Firstly, I gather the data hence the data provided by 10 Academy is already downloaded. The data folder contains three CSV files, each representing solar radiation data from an African country. Here is the list of data:

- `benin-malanville.csv`
- `sierraleone-bumbuna.csv`
- `togo-dapaong_qc.csv`

Then, I will convert each country's data into a separate `Pandas DataFrame`. Secondly, The data will be assessed both visually and programmatically. Thirdly, The data will be cleaned, and then the three DataFrames will be merged into one. Finally, I going to anayze and vsiulaize the data.

In [1]:
```
#import the required libraries for the EDA
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Data Gathering

In [2]:
```
# Convert CSV file into Panadas Dataframe
benin_df = pd.read_csv("../data/benin-malanville.csv")
sierraleone_df = pd.read_csv("../data/sierraleone-bumbuna.csv")
togo_df = pd.read_csv("../data/togo-dapaong_qc.csv")
```

Assessing Data

Visual Assessemnt

In [3]:
```
benin_df
```

Out[3]:

	Timestamp	GHI	DNI	DHI	ModA	ModB	Tamb	RH	WS	WSgust	WSstde
0	2021-08-09 00:01	-1.2	-0.2	-1.1	0.0	0.0	26.2	93.4	0.0	0.4	0.
1	2021-08-09 00:02	-1.1	-0.2	-1.1	0.0	0.0	26.2	93.6	0.0	0.0	0.0
2	2021-08-09 00:03	-1.1	-0.2	-1.1	0.0	0.0	26.2	93.7	0.3	1.1	0.5
3	2021-08-09 00:04	-1.1	-0.1	-1.0	0.0	0.0	26.2	93.3	0.2	0.7	0.4
4	2021-08-09 00:05	-1.0	-0.1	-1.0	0.0	0.0	26.2	93.3	0.1	0.7	0.3
...	...	...	...	...	...	...	...	...	...	...	..
525595	2022-08-08 23:56	-5.5	-0.1	-5.9	0.0	0.0	23.1	98.3	0.3	1.1	0.5
525596	2022-08-08 23:57	-5.5	-0.1	-5.8	0.0	0.0	23.1	98.3	0.2	0.7	0.4
525597	2022-08-08 23:58	-5.5	-0.1	-5.8	0.0	0.0	23.1	98.4	0.6	1.1	0.5
525598	2022-08-08 23:59	-5.5	-0.1	-5.8	0.0	0.0	23.1	98.3	0.9	1.3	0.5
525599	2022-08-09 00:00	-5.5	-0.1	-5.7	0.0	0.0	23.1	98.3	1.2	1.6	0.3

525600 rows × 19 columns

- Missing values in the `comments` column
- `GHI`, `DNI`, and `DHI` columns contains negative values

In [4]:
```
sierraleone_df
```

Out[4]:

	Timestamp	GHI	DNI	DHI	ModA	ModB	Tamb	RH	WS	WSgust	WSstde
0	2021-10-30 00:01	-0.7	-0.1	-0.8	0.0	0.0	21.9	99.1	0.0	0.0	0
1	2021-10-30 00:02	-0.7	-0.1	-0.8	0.0	0.0	21.9	99.2	0.0	0.0	0
2	2021-10-30 00:03	-0.7	-0.1	-0.8	0.0	0.0	21.9	99.2	0.0	0.0	0
3	2021-10-30 00:04	-0.7	0.0	-0.8	0.0	0.0	21.9	99.3	0.0	0.0	0
4	2021-10-30 00:05	-0.7	-0.1	-0.8	0.0	0.0	21.9	99.3	0.0	0.0	0
...	...	...	...	...	...	...	...	...	...	...	
525595	2022-10-29 23:56	-1.6	-0.1	-2.9	0.0	0.0	24.0	100.0	0.0	0.0	0
525596	2022-10-29 23:57	-1.7	-0.1	-3.0	0.0	0.0	24.0	100.0	0.0	0.0	0
525597	2022-10-29 23:58	-1.7	-0.1	-3.1	0.0	0.0	24.0	100.0	0.0	0.0	0
525598	2022-10-29 23:59	-1.7	-0.2	-3.3	0.0	0.0	23.9	100.0	0.0	0.0	0
525599	2022-10-30 00:00	-1.7	-0.1	-3.4	0.0	0.0	23.9	100.0	0.0	0.0	0

525600 rows × 19 columns

- Missing values in the `comments` column
- `GHI`, `DNI`, and `DHI` columns contains negative values

In [5]:

```
togo_df
```

Out[5]:

	Timestamp	GHI	DNI	DHI	ModA	ModB	Tamb	RH	WS	WSgust	WSstdev
0	2021-10-25 00:01	-1.3	0.0	0.0	0.0	0.0	24.8	94.5	0.9	1.1	0.4
1	2021-10-25 00:02	-1.3	0.0	0.0	0.0	0.0	24.8	94.4	1.1	1.6	0.4
2	2021-10-25 00:03	-1.3	0.0	0.0	0.0	0.0	24.8	94.4	1.2	1.4	0.3
3	2021-10-25 00:04	-1.2	0.0	0.0	0.0	0.0	24.8	94.3	1.2	1.6	0.3
4	2021-10-25 00:05	-1.2	0.0	0.0	0.0	0.0	24.8	94.0	1.3	1.6	0.4
...	...	...	...	...	...	...	...	...	...	...	...
525595	2022-10-24 23:56	-0.8	0.0	0.0	0.0	0.0	25.2	53.8	0.0	0.0	0.0
525596	2022-10-24 23:57	-0.9	0.0	0.0	0.0	0.0	25.3	53.5	0.0	0.0	0.0
525597	2022-10-24 23:58	-1.0	0.0	0.0	0.0	0.0	25.3	53.4	0.0	0.0	0.0
525598	2022-10-24 23:59	-1.1	0.0	0.0	0.0	0.0	25.4	53.5	0.0	0.0	0.0
525599	2022-10-25 00:00	-1.2	0.0	0.0	0.0	0.0	25.4	52.3	0.0	0.0	0.0

525600 rows × 19 columns

- Missing values in the `comments` column
- `GHI` column contains negative values

Programmatic Assessement

In [6]:

```
benin_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 525600 entries, 0 to 525599
Data columns (total 19 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   Timestamp      525600 non-null  object
 1   GHI            525600 non-null  float64
 2   DNI            525600 non-null  float64
 3   DHI            525600 non-null  float64
 4   ModA           525600 non-null  float64
 5   ModB           525600 non-null  float64
 6   Tamb           525600 non-null  float64
 7   RH             525600 non-null  float64
 8   WS             525600 non-null  float64
 9   WSgust         525600 non-null  float64
 10  WSstdev        525600 non-null  float64
 11  WD             525600 non-null  float64
 12  WDstdev        525600 non-null  float64
 13  BP             525600 non-null  int64
 14  Cleaning       525600 non-null  int64
 15  Precipitation  525600 non-null  float64
 16  TModA          525600 non-null  float64
 17  TModB          525600 non-null  float64
 18  Comments       0 non-null       float64
dtypes: float64(16), int64(2), object(1)
memory usage: 76.2+ MB
```

- Erroneous datatype `Timestamp` column should be `Datatime`

In [7]:
```
#check duplicated values in benin_df
sum(benin_df.duplicated())
```

Out[7]:
```
0
```

In [8]:
```
# check null values in benin_df
benin_df.isna().sum()
```

Out[8]:
```
Timestamp             0
GHI                   0
DNI                   0
DHI                   0
ModA                  0
ModB                  0
Tamb                  0
RH                    0
WS                    0
WSgust                0
WSstdev               0
WD                    0
WDstdev               0
BP                    0
Cleaning              0
Precipitation         0
TModA                 0
TModB                 0
Comments         525600
dtype: int64
```

In [9]:
```
# Summary Statistics
benin_df.describe()
```

Out[9]:

	GHI	DNI	DHI	ModA	ModB
count	525600.000000	525600.000000	525600.000000	525600.000000	525600.000000
mean	240.559452	167.187516	115.358961	236.589496	228.883576
std	331.131327	261.710501	158.691074	326.894859	316.536515
min	-12.900000	-7.800000	-12.600000	0.000000	0.000000
25%	-2.000000	-0.500000	-2.100000	0.000000	0.000000
50%	1.800000	-0.100000	1.600000	4.500000	4.300000
75%	483.400000	314.200000	216.300000	463.700000	447.900000
max	1413.000000	952.300000	759.200000	1342.300000	1342.300000

In [10]:
```
sierraleone_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 525600 entries, 0 to 525599
Data columns (total 19 columns):
 #   Column         Non-Null Count   Dtype  
---  ------         --------------   -----  
 0   Timestamp      525600 non-null  object 
 1   GHI            525600 non-null  float64
 2   DNI            525600 non-null  float64
 3   DHI            525600 non-null  float64
 4   ModA           525600 non-null  float64
 5   ModB           525600 non-null  float64
 6   Tamb           525600 non-null  float64
 7   RH             525600 non-null  float64
 8   WS             525600 non-null  float64
 9   WSgust         525600 non-null  float64
 10  WSstdev        525600 non-null  float64
 11  WD             525600 non-null  float64
 12  WDstdev        525600 non-null  float64
 13  BP             525600 non-null  int64  
 14  Cleaning       525600 non-null  int64  
 15  Precipitation  525600 non-null  float64
 16  TModA          525600 non-null  float64
 17  TModB          525600 non-null  float64
 18  Comments       0 non-null       float64
dtypes: float64(16), int64(2), object(1)
memory usage: 76.2+ MB
```

- Erroneous datatype `Timestamp` column should be `Datatime`

In [11]:
```
#check duplicated values in benin_df
sum(sierraleone_df.duplicated())
```

Out[11]: 0

In [12]:
```
# check null values in benin_df
sierraleone_df.isna().sum()
```

Out[12]:
```
Timestamp             0
GHI                   0
DNI                   0
DHI                   0
ModA                  0
ModB                  0
Tamb                  0
RH                    0
WS                    0
WSgust                0
WSstdev               0
WD                    0
WDstdev               0
BP                    0
Cleaning              0
Precipitation         0
TModA                 0
TModB                 0
Comments         525600
dtype: int64
```

In [13]:
```
# Summary Statistics
sierraleone_df.describe()
```

Out[13]:

	GHI	DNI	DHI	ModA	ModB
count	525600.000000	525600.000000	525600.000000	525600.000000	525600.000000
mean	201.957515	116.376337	113.720571	206.643095	198.114691
std	298.495150	218.652659	158.946032	300.896893	288.889073
min	-19.500000	-7.800000	-17.900000	0.000000	0.000000
25%	-2.800000	-0.300000	-3.800000	0.000000	0.000000
50%	0.300000	-0.100000	-0.100000	3.600000	3.400000
75%	362.400000	107.000000	224.700000	359.500000	345.400000
max	1499.000000	946.000000	892.000000	1507.000000	1473.000000

In [14]:
```
togo_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 525600 entries, 0 to 525599
Data columns (total 19 columns):
 #   Column         Non-Null Count   Dtype  
---  ------         --------------   -----  
 0   Timestamp      525600 non-null  object 
 1   GHI            525600 non-null  float64
 2   DNI            525600 non-null  float64
 3   DHI            525600 non-null  float64
 4   ModA           525600 non-null  float64
 5   ModB           525600 non-null  float64
 6   Tamb           525600 non-null  float64
 7   RH             525600 non-null  float64
 8   WS             525600 non-null  float64
 9   WSgust         525600 non-null  float64
 10  WSstdev        525600 non-null  float64
 11  WD             525600 non-null  float64
 12  WDstdev        525600 non-null  float64
 13  BP             525600 non-null  int64  
 14  Cleaning       525600 non-null  int64  
 15  Precipitation  525600 non-null  float64
 16  TModA          525600 non-null  float64
 17  TModB          525600 non-null  float64
 18  Comments       0 non-null       float64
dtypes: float64(16), int64(2), object(1)
memory usage: 76.2+ MB
```

- Erroneous datatype `Timestamp` column should be `Datatime`

In [15]:

```
#check duplicated values in benin_df
sum(togo_df.duplicated())
```

Out[15]: 0

In [16]:

```
# check null values in benin_df
togo_df.isna().sum()
```

Out[16]:

```
Timestamp             0
GHI                   0
DNI                   0
DHI                   0
ModA                  0
ModB                  0
Tamb                  0
RH                    0
WS                    0
WSgust                0
WSstdev               0
WD                    0
WDstdev               0
BP                    0
Cleaning              0
Precipitation         0
TModA                 0
TModB                 0
Comments         525600
dtype: int64
```

In [17]:
```
# Summary Statistics
togo_df.describe()
```

Out[17]:

	GHI	DNI	DHI	ModA	ModB
count	525600.000000	525600.000000	525600.000000	525600.000000	525600.000000
mean	230.555040	151.258469	116.444352	226.144375	219.568588
std	322.532347	250.956962	156.520714	317.346938	307.932510
min	-12.700000	0.000000	0.000000	0.000000	0.000000
25%	-2.200000	0.000000	0.000000	0.000000	0.000000
50%	2.100000	0.000000	2.500000	4.400000	4.300000
75%	442.400000	246.400000	215.700000	422.525000	411.000000
max	1424.000000	1004.500000	805.700000	1380.000000	1367.000000

Quality issues

1. Missing values in the `comments` column.
2. `GHI`, `DNI`, and `DHI` columns contains negative values.
3. Erroneous datatype `Timestamp` column should be `Datatime`.

Data Cleaning

Quality issues

In [18]:
```
# make a copies of the original data
benin_df_clean = benin_df.copy()
sierraleone_df_clean = sierraleone_df.copy()
togo_df_clean = togo_df.copy()
```

Issue #1:

The `comments` column contains missing values

Define

The `comments` column contains enterily null value in all the three dataframe. I use pandas `drop` method to drop the column.

Code

In [19]:
```
benin_df_clean.drop("Comments", axis = 1, inplace=True)
sierraleone_df_clean.drop("Comments", axis = 1, inplace=True)
togo_df_clean.drop("Comments", axis = 1, inplace=True)
```

Test

In [20]:
```
print(benin_df_clean.columns)
print(sierraleone_df_clean.columns)
print(togo_df_clean.columns)
```

```
Index(['Timestamp', 'GHI', 'DNI', 'DHI', 'ModA', 'ModB', 'Tamb', 'RH', 'W
S',
       'WSgust', 'WSstdev', 'WD', 'WDstdev', 'BP', 'Cleaning', 'Precipitat
ion',
       'TModA', 'TModB'],
      dtype='object')
Index(['Timestamp', 'GHI', 'DNI', 'DHI', 'ModA', 'ModB', 'Tamb', 'RH', 'W
S',
       'WSgust', 'WSstdev', 'WD', 'WDstdev', 'BP', 'Cleaning', 'Precipitat
ion',
       'TModA', 'TModB'],
      dtype='object')
Index(['Timestamp', 'GHI', 'DNI', 'DHI', 'ModA', 'ModB', 'Tamb', 'RH', 'W
S',
       'WSgust', 'WSstdev', 'WD', 'WDstdev', 'BP', 'Cleaning', 'Precipitat
ion',
       'TModA', 'TModB'],
      dtype='object')
```

Issue #2:

`GHI` , `DNI` , and `DHI` columns contains negative values.

Define

`GHI` , `DNI` , and `DHI` columns contains negative values in all the three dataframe. I use 'abs' function to convert each value into absolute value.

Code

In [21]:
```
benin_df_clean[['GHI', 'DNI', 'DHI']] = benin_df_clean[['GHI', 'DNI', 'DH
sierraleone_df_clean[['GHI', 'DNI', 'DHI']] = sierraleone_df_clean[['GHI'
togo_df_clean[['GHI', 'DNI', 'DHI']] = togo_df_clean[['GHI', 'DNI', 'DHI'
```

Test

In [22]:
```
benin_df_clean[['GHI', 'DNI', 'DHI']]
```

Out[22]:

	GHI	DNI	DHI
0	1.2	0.2	1.1
1	1.1	0.2	1.1
2	1.1	0.2	1.1
3	1.1	0.1	1.0
4	1.0	0.1	1.0
...	...	...	...
525595	5.5	0.1	5.9
525596	5.5	0.1	5.8
525597	5.5	0.1	5.8
525598	5.5	0.1	5.8
525599	5.5	0.1	5.7

525600 rows × 3 columns

In [23]:

```
sierraleone_df_clean[['GHI', 'DNI', 'DHI']]
```

Out[23]:

	GHI	DNI	DHI
0	0.7	0.1	0.8
1	0.7	0.1	0.8
2	0.7	0.1	0.8
3	0.7	0.0	0.8
4	0.7	0.1	0.8
...	...	...	...
525595	1.6	0.1	2.9
525596	1.7	0.1	3.0
525597	1.7	0.1	3.1
525598	1.7	0.2	3.3
525599	1.7	0.1	3.4

525600 rows × 3 columns

In [24]:

```
togo_df_clean[['GHI', 'DNI', 'DHI']]
```

Out[24]:

	GHI	DNI	DHI
0	1.3	0.0	0.0
1	1.3	0.0	0.0
2	1.3	0.0	0.0
3	1.2	0.0	0.0
4	1.2	0.0	0.0
...	...	...	...
525595	0.8	0.0	0.0
525596	0.9	0.0	0.0
525597	1.0	0.0	0.0
525598	1.1	0.0	0.0
525599	1.2	0.0	0.0

525600 rows × 3 columns

Issue #3:

Erroneous datatype `Timestamp` column should be Datatime.

Define

`Timestamp` column contains wrong datatype in all the three dataframes, it should has a datetime datatype. I use `to_datetime` pandas method to convert `object` type into `datetime` type.

Code

In [26]:

```
# convert the datatype
benin_df_clean['Timestamp'] = pd.to_datetime(benin_df_clean['Timestamp'])
sierraleone_df_clean['Timestamp'] = pd.to_datetime(sierraleone_df_clean['
togo_df_clean['Timestamp'] = pd.to_datetime(togo_df_clean['Timestamp']).d
```

Test

In [27]:

```
# check the datatype
print(benin_df_clean['Timestamp'].dtype)
print(sierraleone_df_clean['Timestamp'].dtype)
print(togo_df_clean['Timestamp'].dtype)
```

```
datetime64[ns]
datetime64[ns]
datetime64[ns]
```

Concatination

Concatinate all the three dataframe into one dataframe.

In [28]:
```
all_clean_df = pd.concat([benin_df_clean, sierraleone_df_clean, togo_df_c
all_clean_df.sample(5)
```

Out[28]:

	Timestamp	GHI	DNI	DHI	ModA	ModB	Tamb	RH	WS	WSgust	W
305798	2022-05-30	242.4	27.4	228.5	226.9	221.6	24.4	93.7	0.0	0.0	
325883	2022-06-13	162.4	359.0	86.4	109.7	104.6	23.4	99.3	0.0	0.0	
117825	2022-01-19	13.0	0.6	13.0	0.0	0.0	24.6	51.0	0.0	0.0	
479717	2022-09-28	0.8	0.1	3.7	0.0	0.0	22.2	99.7	0.0	0.0	
24908	2021-11-16	30.4	0.1	30.3	31.9	30.2	23.3	100.0	0.0	0.0	

In [33]:
```
all_clean_df.shape
```

Out[33]: (1576800, 18)

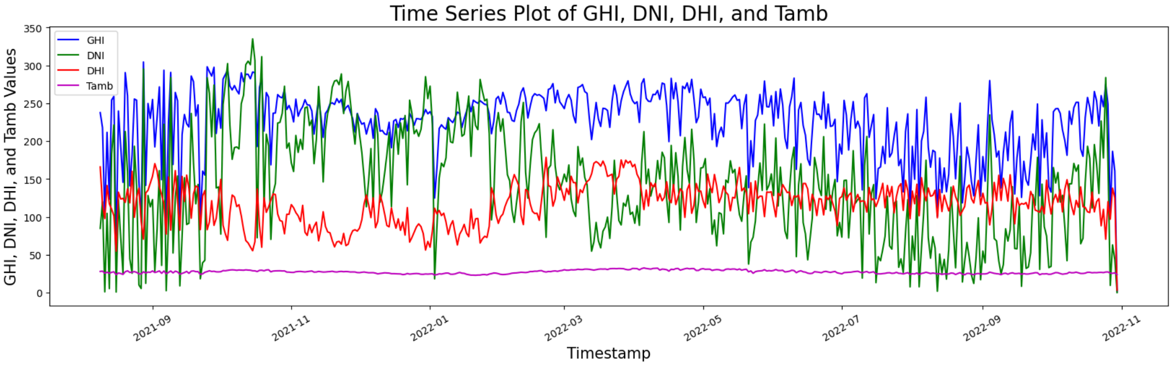
Time Series Analysis

Analyze how variables like GHI, DNI, DHI, and Tamb change over time.

In [29]:
```
# group data by Timestamp column for Timeseries plot
grouped_df = all_clean_df.groupby(['Timestamp']).mean()[['GHI', 'DNI', 'D
grouped_df.reset_index(inplace=True)
```

In [30]:
```
cols = ['GHI', 'DNI', 'DHI', 'Tamb']
colors = ['b', 'g', 'r', 'm']  # blue, green, red, magenta
fig = plt.subplots(figsize=(20, 5))
for col, color in zip(cols, colors):
    plt.plot(grouped_df['Timestamp'], grouped_df[col], label=col, color=c
plt.title('Time Series Plot of GHI, DNI, DHI, and Tamb', fontsize=20)
plt.xlabel('Timestamp', fontsize=15)
plt.ylabel('GHI, DNI, DHI, and Tamb Values', fontsize=15)
plt.xticks(rotation=30)
plt.legend()
```

Out[30]: <matplotlib.legend.Legend at 0x74d3d246e950>



As we can see from the above graph, the `Tamb` data seems fixed throughout all the years, and the rest of the column data (`DHI, GHI, DNI`) vary throughout the years.

Correlation Analysis

Determine the correlation between different variables like solar radiation components (`GHI, DHI, DNI`) and temperature measures (`TModA, TModB`) to uncover relationships.

To analyze the relationships between variables, I employ correlation coefficients calculated using the `corr` method of pandas DataFrames.

In [31]:
```
#Calculate correlation coefficients
correlation_matrix = all_clean_df[["GHI", "DHI", "DNI", "TModA", "TModB"]
print(correlation_matrix)
```

```
            GHI       DHI       DNI     TModA     TModB
GHI    1.000000  0.851051  0.876611  0.905345  0.898338
DHI    0.851051  1.000000  0.531800  0.800149  0.797958
DNI    0.876611  0.531800  1.000000  0.784409  0.775983
TModA  0.905345  0.800149  0.784409  1.000000  0.969891
TModB  0.898338  0.797958  0.775983  0.969891  1.000000
```

As we can see from the correlation coefficient results, there is a strong relationship between the solar radiation components (`GHI, DHI, DNI`) and temperature measures (`TModA, TModB`). Notably, `GHI` shows the strongest correlation with the temperature measures (`TModA = 0.905345` and `TModB = 0.898338`).

Wind Analysis

Explore wind speed (`WS, WSgust, WSstdev`) and wind direction (`WD, WDstdev`) data to identify any trends or notable wind events.

I employ summary statistics (`mean, median, standard deviation, minimum, maximum`) for wind speed (`WS, WSgust, and WSstdev`) to understand central tendency and variability.

In [33]:
```
wind_speed_stats = all_clean_df[["WS", "WSgust", "WSstdev"]].describe()
print(wind_speed_stats)
```

```
                 WS        WSgust       WSstdev
count  1.576800e+06  1.576800e+06  1.576800e+06
mean   1.878440e+00  2.576763e+00  4.649840e-01
std    1.536357e+00  1.961275e+00  2.904002e-01
min    0.000000e+00  0.000000e+00  0.000000e+00
25%    6.000000e-01  1.100000e+00  4.000000e-01
50%    1.800000e+00  2.400000e+00  5.000000e-01
75%    2.800000e+00  3.600000e+00  6.000000e-01
max    1.950000e+01  2.660000e+01  4.700000e+00
```

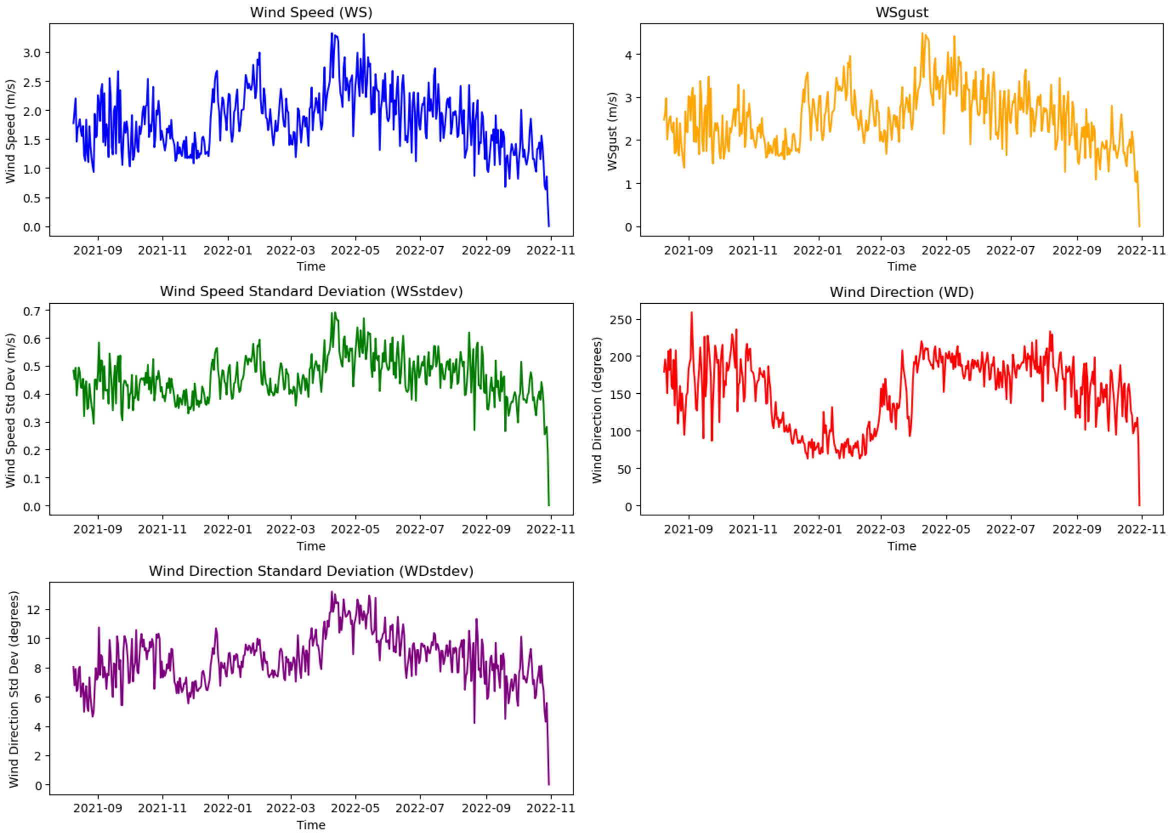
Based on the summary statistics output:

- The average wind speed is `1.88 m/s`, indicating a moderate wind regime.
- The standard deviation of wind speed is `1.54 m/s`, suggesting a moderate variability in wind speeds around the mean.
- The wind speed ranged from `0` m/s (calm winds) to a maximum of `19.5` m/s (strong breeze).

- The quartiles (25th and 75th percentiles) show that wind speeds are distributed between `0.6` m/s and `2.8` m/s for most of the time (IQR). The 50th percentile (median) is `1.8 m/s`, which is close to the mean, suggesting a symmetrical distribution.

In [40]:
```
wind_data = all_clean_df.groupby(['Timestamp']).mean()[['WS', 'WSgust', '
wind_data.reset_index(inplace=True)
```

In [41]:
```
# Plotting time-series graphs for each column
plt.figure(figsize=(14, 10))

# WS
plt.subplot(3, 2, 1)
plt.plot(wind_data['Timestamp'], wind_data['WS'], color='blue')
plt.title('Wind Speed (WS)')
plt.xlabel('Time')
plt.ylabel('Wind Speed (m/s)')

# WSgust
plt.subplot(3, 2, 2)
plt.plot(wind_data['Timestamp'], wind_data['WSgust'], color='orange')
plt.title('WSgust')
plt.xlabel('Time')
plt.ylabel('WSgust (m/s)')

# WSstdev
plt.subplot(3, 2, 3)
plt.plot(wind_data['Timestamp'], wind_data['WSstdev'], color='green')
plt.title('Wind Speed Standard Deviation (WSstdev)')
plt.xlabel('Time')
plt.ylabel('Wind Speed Std Dev (m/s)')

# WD
plt.subplot(3, 2, 4)
plt.plot(wind_data['Timestamp'], wind_data['WD'], color='red')
plt.title('Wind Direction (WD)')
plt.xlabel('Time')
plt.ylabel('Wind Direction (degrees)')

# WDstdev
plt.subplot(3, 2, 5)
plt.plot(wind_data['Timestamp'], wind_data['WDstdev'], color='purple')
plt.title('Wind Direction Standard Deviation (WDstdev)')
plt.xlabel('Time')
plt.ylabel('Wind Direction Std Dev (degrees)')

plt.tight_layout()
```



The time-series graphs reveal that wind speed and wind direction have the same pattern.

Temperature Analysis

Compare module temperatures (`TModA, TModB`) with ambient temperature (`Tamb`) to see how they are related or vary under different conditions.

I calculate the correlation coefficient between each module temperature and ambient temperature. This will quantify the strength and direction of the linear relationship.

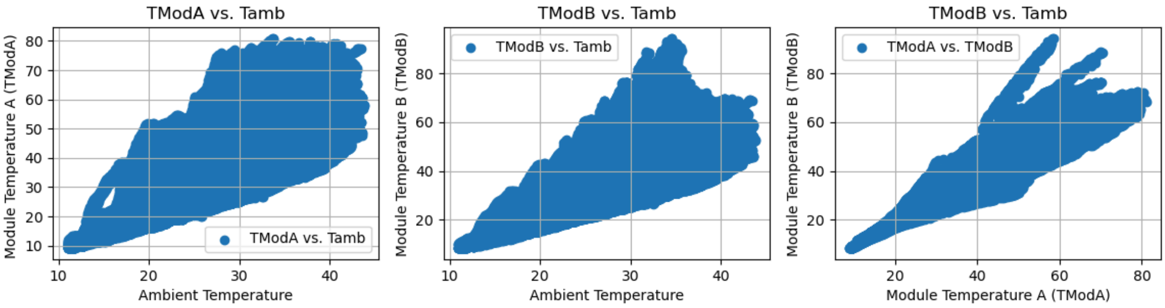
In [43]:
```
# calculate the correlation coefficient
correlation_matrix = all_clean_df[["TModA", "TModB", "Tamb"]].corr()
print(correlation_matrix)
```

```
          TModA     TModB      Tamb
TModA  1.000000  0.969891  0.787788
TModB  0.969891  1.000000  0.789931
Tamb   0.787788  0.789931  1.000000
```

In [46]:
```
# ploting Scatter plot

plt.figure(figsize=(14, 10))

#TModA vs Tamb
plt.subplot(3, 3, 1)
plt.scatter(all_clean_df["Tamb"], all_clean_df["TModA"], label="TModA vs.
plt.xlabel("Ambient Temperature")
plt.ylabel("Module Temperature A (TModA) ")
plt.title("TModA vs. Tamb")
plt.grid(True)
plt.legend()
```

```
# TModB vs Tamb
plt.subplot(3, 3, 2)
plt.scatter(all_clean_df["Tamb"], all_clean_df["TModB"], label="TModB vs.
plt.xlabel("Ambient Temperature")
plt.ylabel("Module Temperature B (TModB)")
plt.title("TModB vs. Tamb")
plt.grid(True)
plt.legend()

# TModA vs TModB
plt.subplot(3, 3, 3)
plt.scatter(all_clean_df["TModA"], all_clean_df["TModB"], label="TModA vs
plt.xlabel("Module Temperature A (TModA)")
plt.ylabel("Module Temperature B (TModB)")
plt.title("TModB vs. Tamb")
plt.grid(True)
plt.legend()
```

Out[46]: <matplotlib.legend.Legend at 0x74d395b0b190>



Both the correlation coefficient and the scatter plot output shows there relationships between the module temperatures (`TModA, TModB`) and ambient temperature (`Tamb`).

- The correlation coefficient of `0.97` indicates a very strong positive linear relationship between `TModA` and `TModB`. This suggests that both module temperatures tend to move in the same direction and experience similar changes.
- The correlation coefficients between module temperatures (around `0.79`) and ambient temperature (`Tamb`) are positive, indicating a tendency for module temperatures to increase as ambient temperature increases. However, the value is not as high as the correlation between the modules themselves.

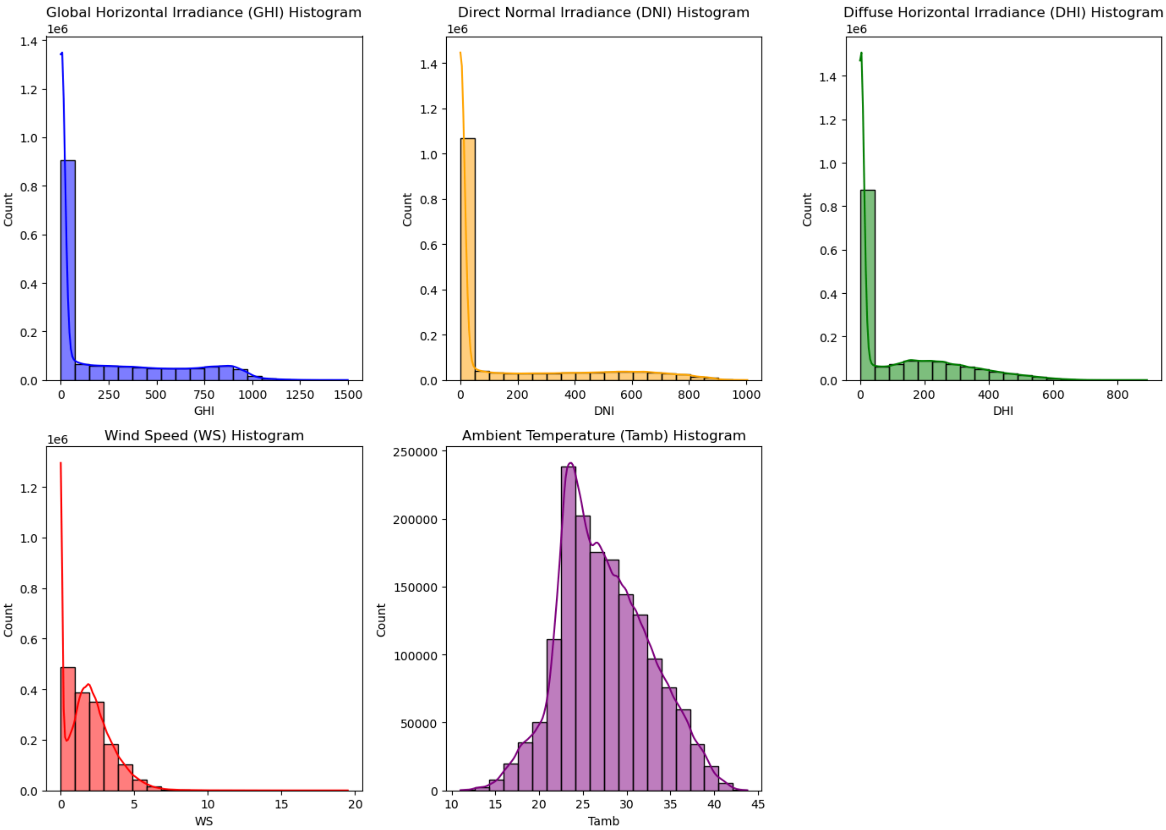
Histograms

Create histograms for variables like `GHI`, `DNI`, `DHI`, `WS`, and temperatures to visualize the frequency distribution of these variables.

In [47]:
```
# Histograms plot
plt.figure(figsize=(14, 10))

plt.subplot(2, 3, 1)
sns.histplot(all_clean_df['GHI'], bins=20, kde=True, color='blue')
plt.title('Global Horizontal Irradiance (GHI) Histogram')

plt.subplot(2, 3, 2)
```

```
sns.histplot(all_clean_df['DNI'], bins=20, kde=True, color='orange')
plt.title('Direct Normal Irradiance (DNI) Histogram')

plt.subplot(2, 3, 3)
sns.histplot(all_clean_df['DHI'], bins=20, kde=True, color='green')
plt.title('Diffuse Horizontal Irradiance (DHI) Histogram')

plt.subplot(2, 3, 4)
sns.histplot(all_clean_df['WS'], bins=20, kde=True, color='red')
plt.title('Wind Speed (WS) Histogram')

plt.subplot(2, 3, 5)
sns.histplot(all_clean_df['Tamb'], bins=20, kde=True, color='purple')
plt.title('Ambient Temperature (Tamb) Histogram')

plt.tight_layout()
plt.show()
```

```
/home/derbew/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:11
19: FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/home/derbew/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:11
19: FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/home/derbew/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:11
19: FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/home/derbew/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:11
19: FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/home/derbew/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:11
19: FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

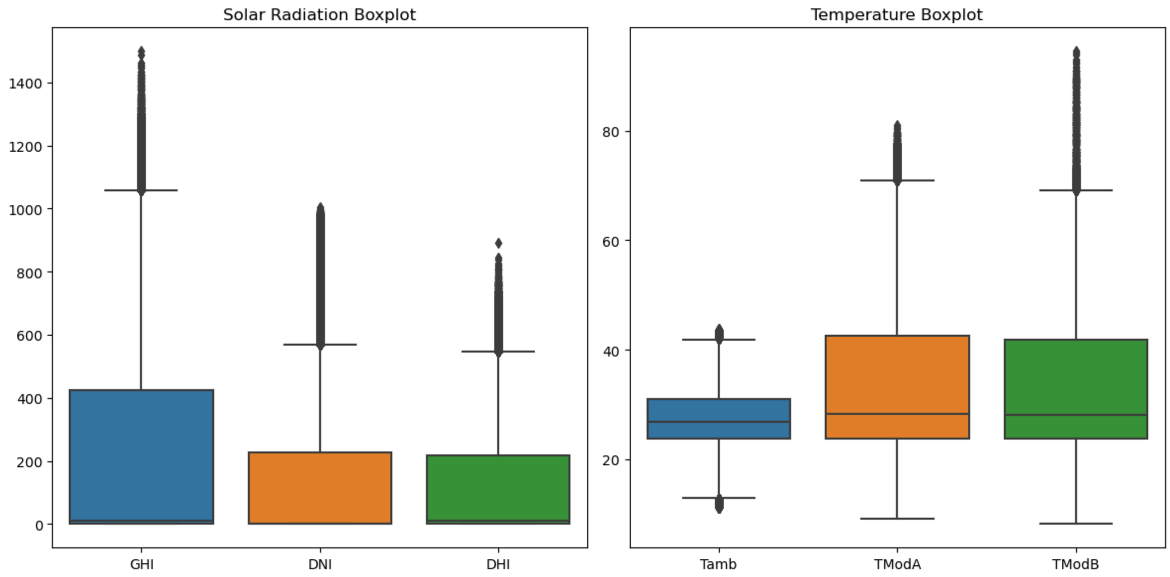


As seen from the above histogram plot, the parameters `GHI, DNI, DHI, and WS` exhibit right-skewed (positive skewness) distributions. In contrast, the `Tamb` parameter has a symmetrical distribution, suggesting that its data is evenly distributed on both sides of the central tendency.

Box Plots

Use box plots to examine the spread and presence of outliers in the solar radiation and temperature data.

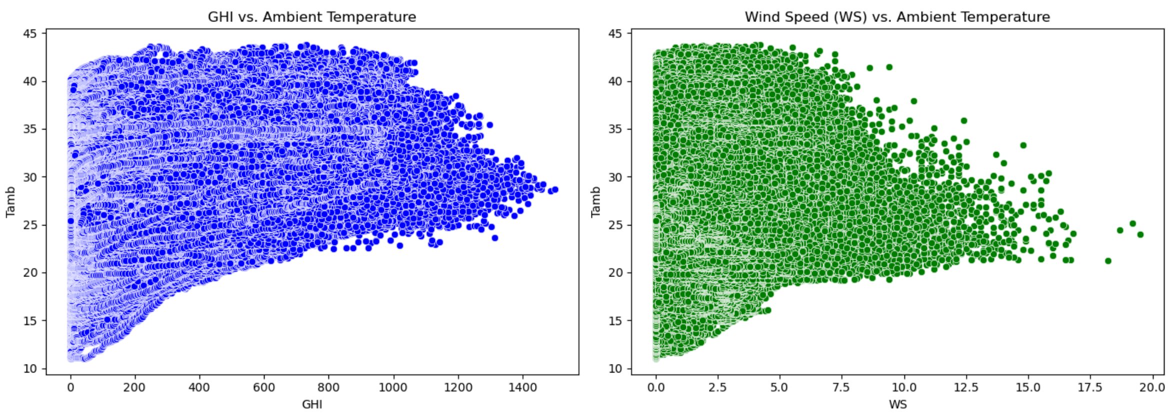
In [48]:
```
# Box Plots
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
sns.boxplot(data=all_clean_df[['GHI', 'DNI', 'DHI']])
plt.title('Solar Radiation Boxplot')

plt.subplot(1, 2, 2)
sns.boxplot(data=all_clean_df[['Tamb', 'TModA', 'TModB']])
plt.title('Temperature Boxplot')

plt.tight_layout()
plt.show()
```



Scatter Plots

Generate scatter plots to explore the relationships between pairs of variables, such as `GHI` vs. `Tamb`, `WS` vs. `WSgust`, or any other potentially interesting pairs.

In [49]:
```python
# Scatter Plots
plt.figure(figsize=(14, 5))

plt.subplot(1, 2, 1)
sns.scatterplot(x='GHI', y='Tamb', data=all_clean_df, color='blue')
plt.title('GHI vs. Ambient Temperature')

plt.subplot(1, 2, 2)
sns.scatterplot(x='WS', y='Tamb', data=all_clean_df, color='green')
plt.title('Wind Speed (WS) vs. Ambient Temperature')

plt.tight_layout()
plt.show()
```



As we can see from the above scatter plot, there appears to be no correlation between `GHI` and `Tamb`, nor between `WS` and `Tamb`.

Conclusions

This project aimed to practice exploratory data analysis (EDA) skills using real-world solar radiation measurement data collected from three African countries: Benin, Sierra Leone, and Togo. The EDA process consisted of three phases:

1. Data Gathering: The data was provided by 10 Academy.
2. Data Cleaning and Preprocessing: The data was cleaned and prepared for analysis.
3. Data Analysis and Visualization: The data was analyzed and visualized to extract insights.

Here is the Key Insights:

- A strong positive correlation exists between module temperatures (TModA and TModB).
- A moderate positive correlation exists between module and ambient temperatures.
- There is no correlation between GHI and Tamb, nor between WS and Tamb.
- The parameters GHI, DNI, DHI, and WS exhibit right-skewed (positive skewness) distributions.
- The Tamb parameter has a symmetrical distribution.
- The time-series graphs reveal that wind speed and wind direction exhibit a similar pattern.

Limitations

Due to time limitations, I was only able to assess three key aspects of the dataset. A more comprehensive analysis would require additional time to explore all potential issues. In the analysis and visualization stage, I focused on obtaining a few initial insights from the dataset. Further exploration through EDA can reveal additional valuable information.

References

Pandas Documentation

Matplotlib Documentation

Seaborn Documentation

Exploratory Data Analysis

```
In [ ]:
```