benin_solar_data_analysis

December 8, 2024

[ ]:

[5]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Load the CSV file into a DataFrame
df = pd.read_csv('../data/benin_cleaned_data.csv')

# Convert the Timestamp column to datetime
df['Timestamp'] = pd.to_datetime(df['Timestamp'])

# Set the Timestamp column as the index
df.set_index('Timestamp', inplace=True)

# Display the first few rows to verify
df.head()
```

[5]:

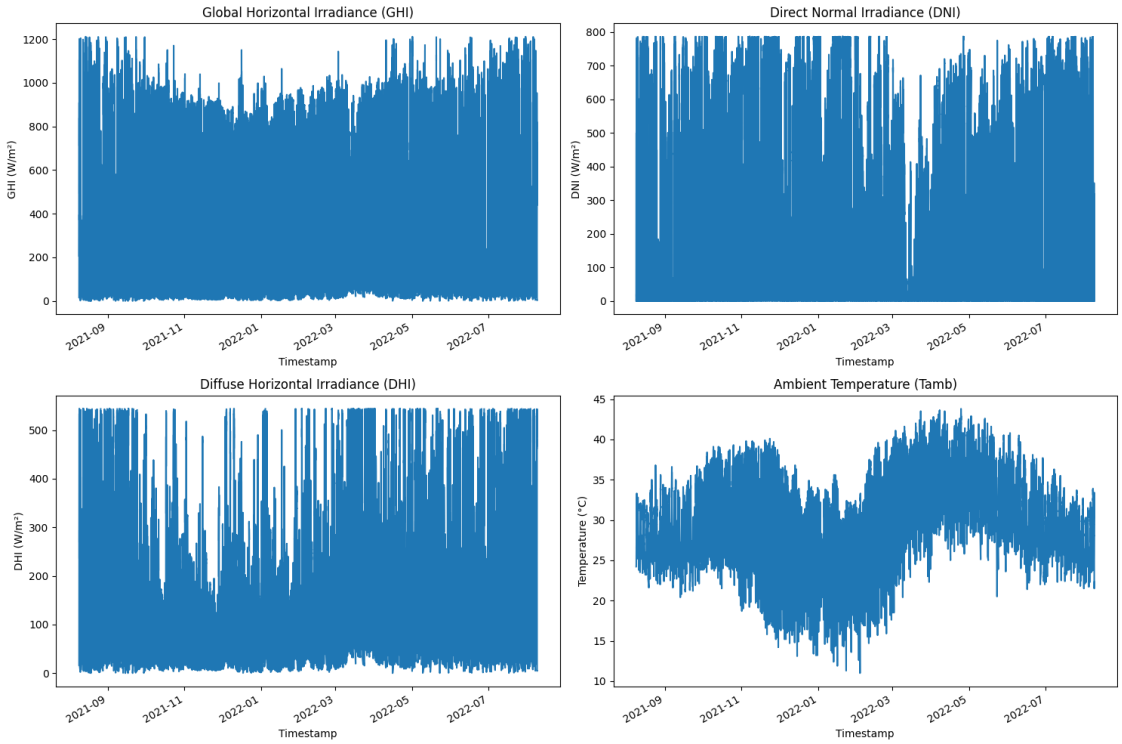
Timestamp	GHI	DNI	DHI	ModA	ModB	Tamb	RH	WS	WSgust
2021-08-09 06:54:00	16.7	0.0	16.5	16.1	16.3	24.2	98.8	0.0	0.0
2021-08-09 06:55:00	18.2	0.1	18.0	17.4	17.6	24.2	98.8	0.0	0.0
2021-08-09 06:56:00	19.7	0.3	19.5	18.7	18.9	24.2	98.8	0.0	0.0
2021-08-09 06:57:00	21.1	0.6	20.9	19.9	20.1	24.2	98.9	0.0	0.0
2021-08-09 06:58:00	22.5	1.1	22.2	21.1	21.3	24.2	98.9	0.0	0.0

Timestamp	WSstdev	WD	WDstdev	BP	Cleaning	Precipitation
2021-08-09 06:54:00	0.0	0.0	0.0	997	0	0.0
2021-08-09 06:55:00	0.0	0.0	0.0	997	0	0.0
2021-08-09 06:56:00	0.0	0.0	0.0	997	0	0.0
2021-08-09 06:57:00	0.0	0.0	0.0	997	0	0.0
2021-08-09 06:58:00	0.0	0.0	0.0	997	0	0.0

Timestamp	TModA	TModB	Comments
2021-08-09 06:54:00	24.2	23.7	NaN
2021-08-09 06:55:00	24.3	23.8	NaN

```
2021-08-09 06:56:00   24.3   23.9       NaN
2021-08-09 06:57:00   24.4   23.9       NaN
2021-08-09 06:58:00   24.5   24.0       NaN
```

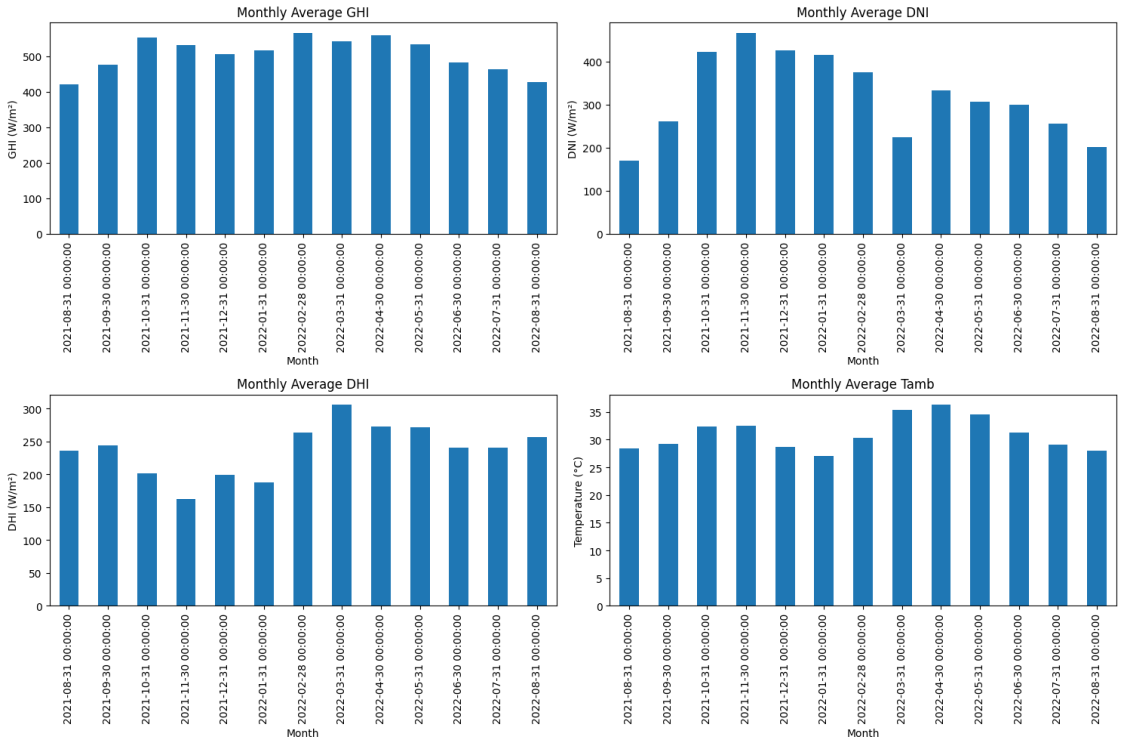
[6]:
```python
# Plot GHI, DNI, DHI, and Tamb over time
plt.figure(figsize=(15, 10))

plt.subplot(2, 2, 1)
df['GHI'].plot(title='Global Horizontal Irradiance (GHI)')
plt.xlabel('Timestamp')
plt.ylabel('GHI (W/m²)')

plt.subplot(2, 2, 2)
df['DNI'].plot(title='Direct Normal Irradiance (DNI)')
plt.xlabel('Timestamp')
plt.ylabel('DNI (W/m²)')

plt.subplot(2, 2, 3)
df['DHI'].plot(title='Diffuse Horizontal Irradiance (DHI)')
plt.xlabel('Timestamp')
plt.ylabel('DHI (W/m²)')

plt.subplot(2, 2, 4)
df['Tamb'].plot(title='Ambient Temperature (Tamb)')
plt.xlabel('Timestamp')
plt.ylabel('Temperature (°C)')

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



[7]:
```
# Resample the data to monthly frequency and calculate mean
monthly_data = df.resample('M').mean()

# Plot monthly averages
plt.figure(figsize=(15, 10))

plt.subplot(2, 2, 1)
monthly_data['GHI'].plot(kind='bar', title='Monthly Average GHI')
plt.xlabel('Month')
plt.ylabel('GHI (W/m²)')

plt.subplot(2, 2, 2)
monthly_data['DNI'].plot(kind='bar', title='Monthly Average DNI')
plt.xlabel('Month')
plt.ylabel('DNI (W/m²)')

plt.subplot(2, 2, 3)
monthly_data['DHI'].plot(kind='bar', title='Monthly Average DHI')
plt.xlabel('Month')
plt.ylabel('DHI (W/m²)')

plt.subplot(2, 2, 4)
monthly_data['Tamb'].plot(kind='bar', title='Monthly Average Tamb')
```

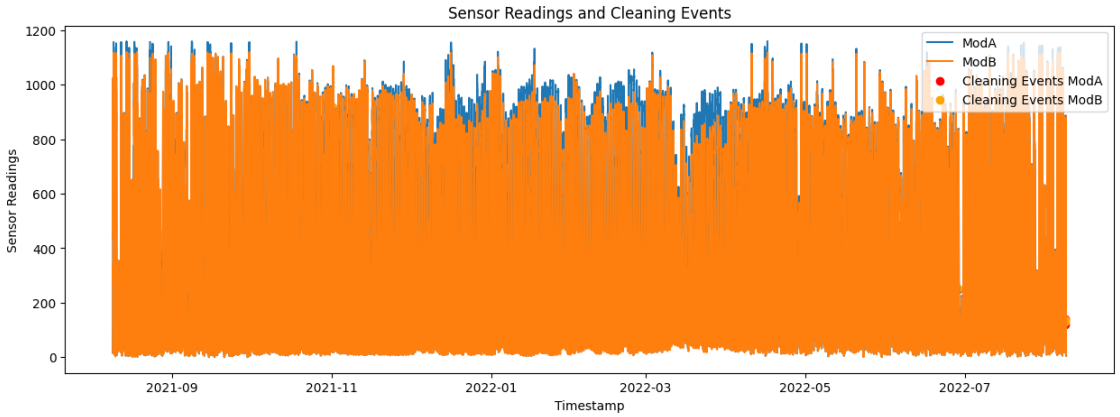
```
plt.xlabel('Month')
plt.ylabel('Temperature (°C)')

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

```
/tmp/ipykernel_9318/2882144599.py:2: FutureWarning: 'M' is deprecated and will
be removed in a future version, please use 'ME' instead.
  monthly_data = df.resample('M').mean()
```

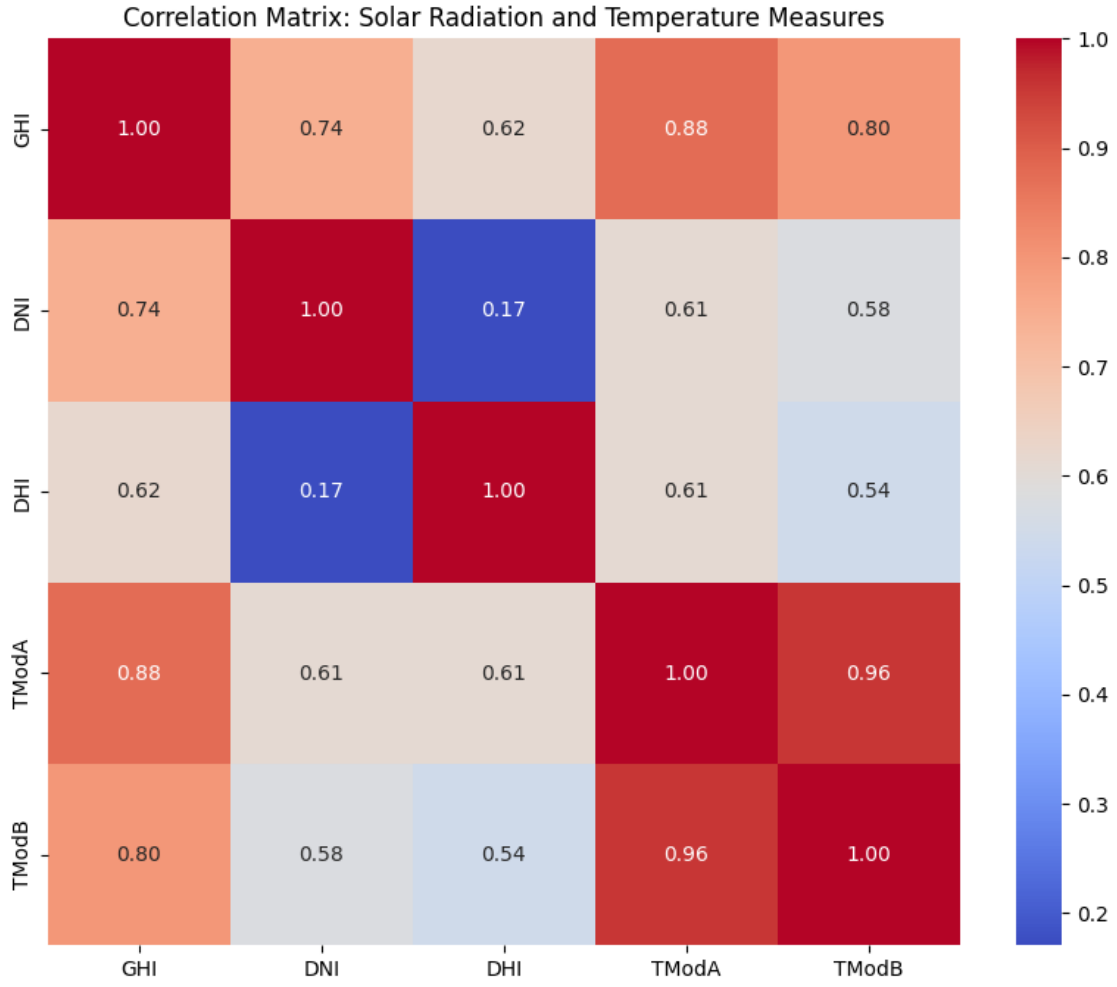


[8]:
```
# Plot sensor readings (ModA, ModB) with Cleaning events
plt.figure(figsize=(15, 5))

plt.plot(df.index, df['ModA'], label='ModA')
plt.plot(df.index, df['ModB'], label='ModB')
plt.scatter(df[df['Cleaning'] > 0].index, df[df['Cleaning'] > 0]['ModA'],␣
 ↪color='red', label='Cleaning Events ModA')
plt.scatter(df[df['Cleaning'] > 0].index, df[df['Cleaning'] > 0]['ModB'],␣
 ↪color='orange', label='Cleaning Events ModB')
plt.title('Sensor Readings and Cleaning Events')
plt.xlabel('Timestamp')
plt.ylabel('Sensor Readings')
plt.legend()
```

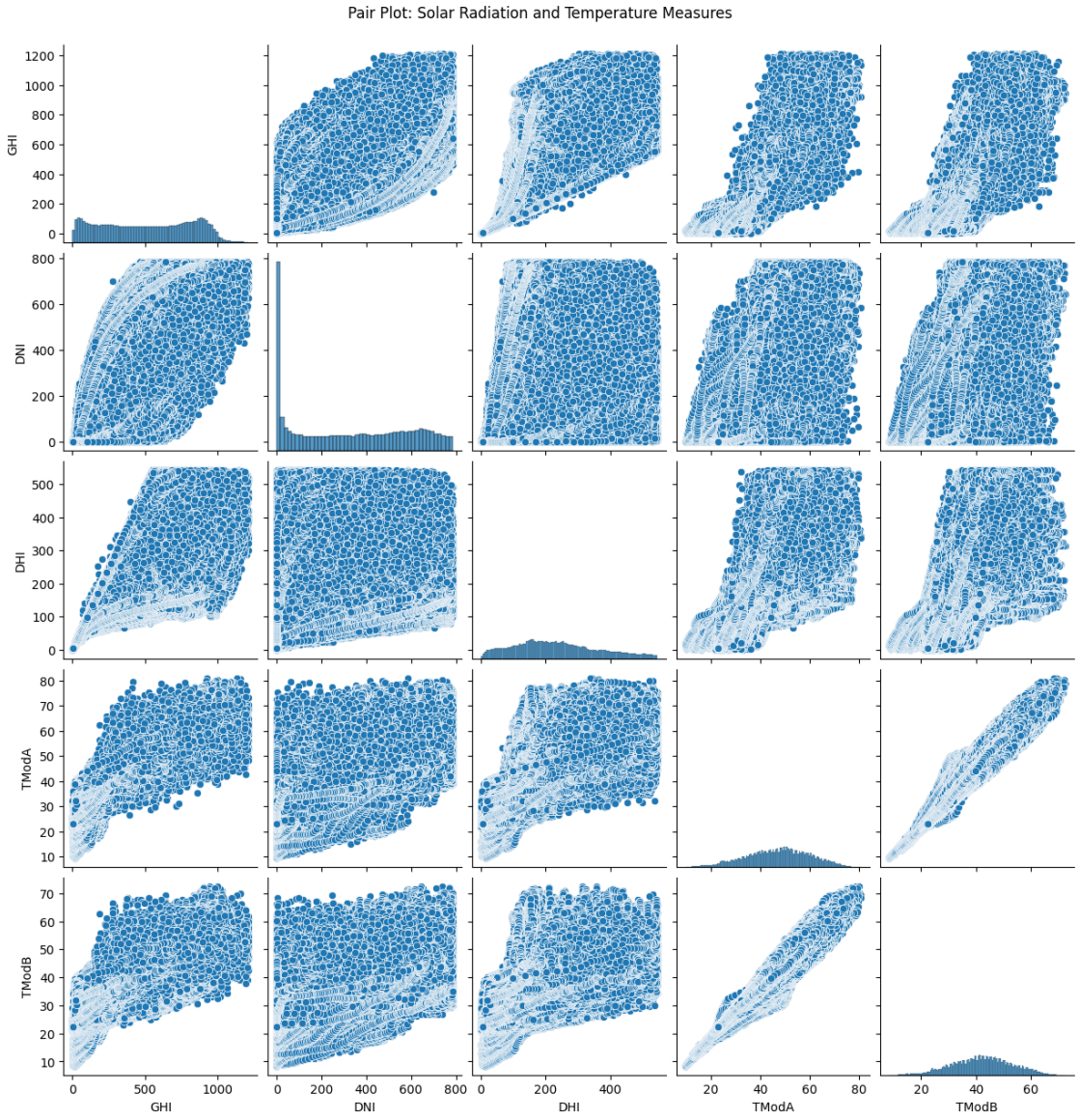
```
plt.show()
```



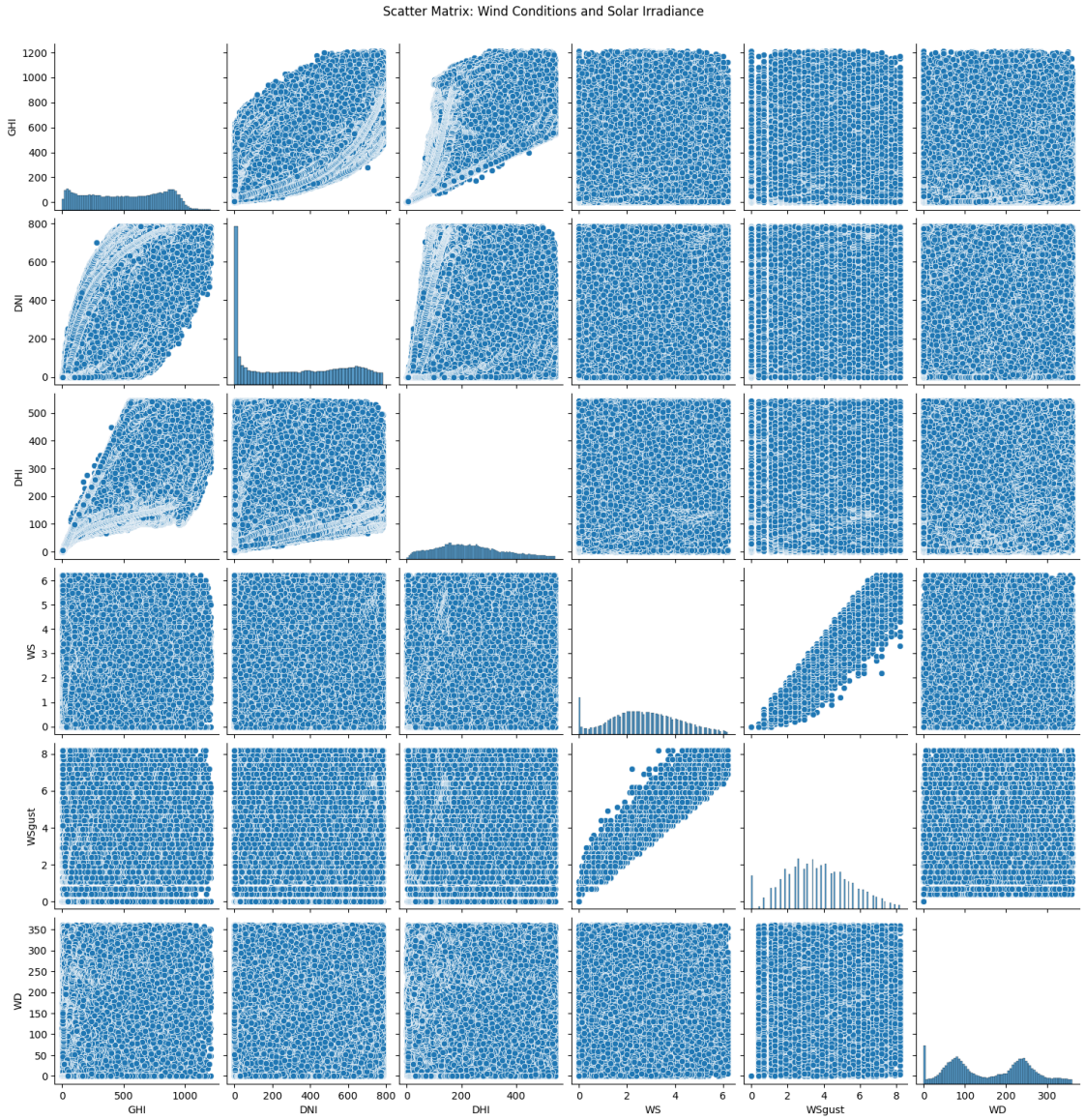
[9]:
```
import seaborn as sns
# Select relevant columns for correlation analysis
solar_temp_cols = ['GHI', 'DNI', 'DHI', 'TModA', 'TModB']

# Calculate the correlation matrix
corr_matrix = df[solar_temp_cols].corr()

# Plot the heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Matrix: Solar Radiation and Temperature Measures')
plt.show()
```



[11]:
```
# Pair plot for solar radiation components and temperature measures
sns.pairplot(df[solar_temp_cols])
plt.suptitle('Pair Plot: Solar Radiation and Temperature Measures', y=1.02)
plt.show()
```

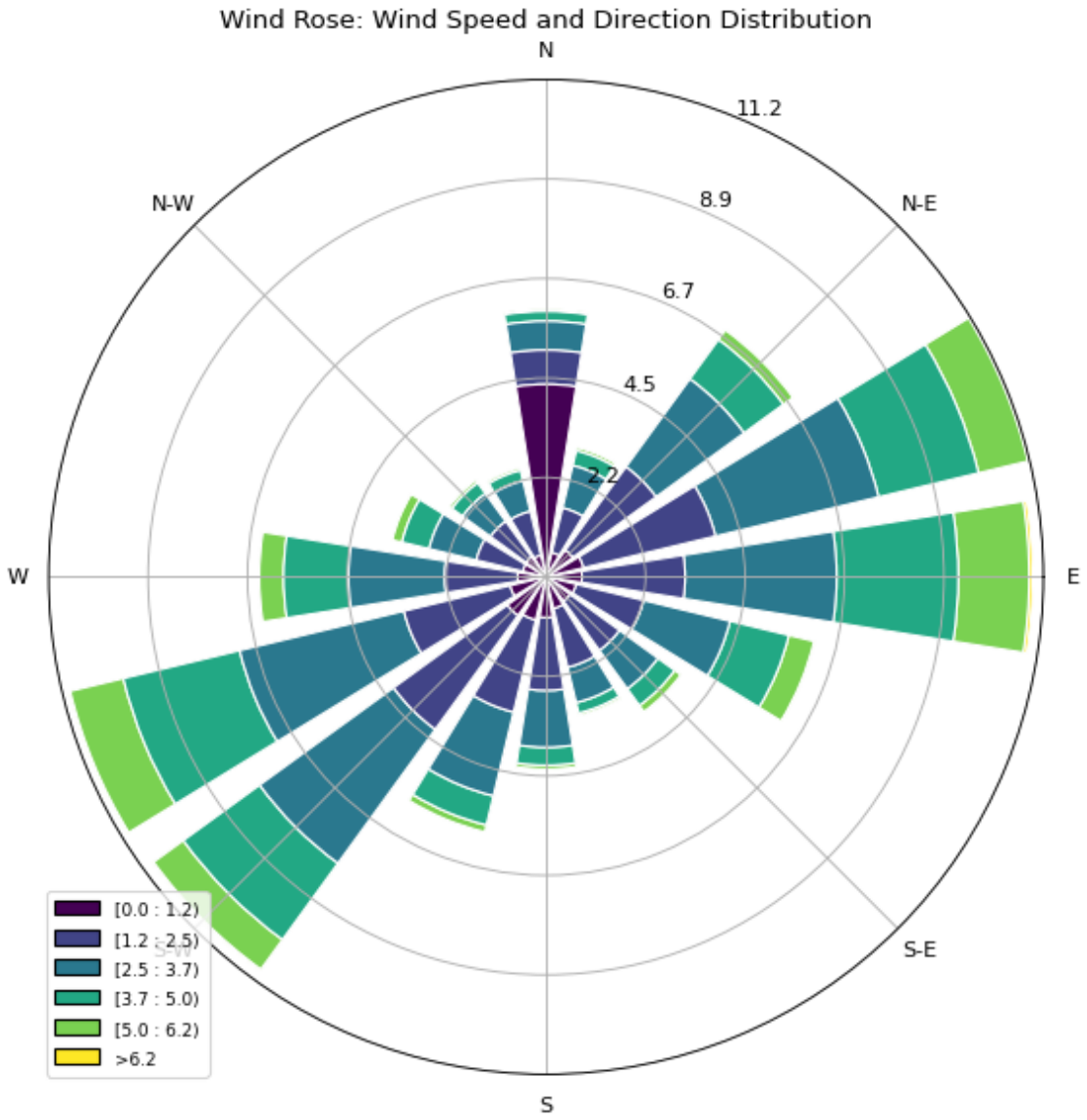


[12]:
```
# Select relevant columns for scatter matrix analysis
wind_solar_cols = ['GHI', 'DNI', 'DHI', 'WS', 'WSgust', 'WD']

# Plot scatter matrix
sns.pairplot(df[wind_solar_cols])
plt.suptitle('Scatter Matrix: Wind Conditions and Solar Irradiance', y=1.02)
plt.show()
```

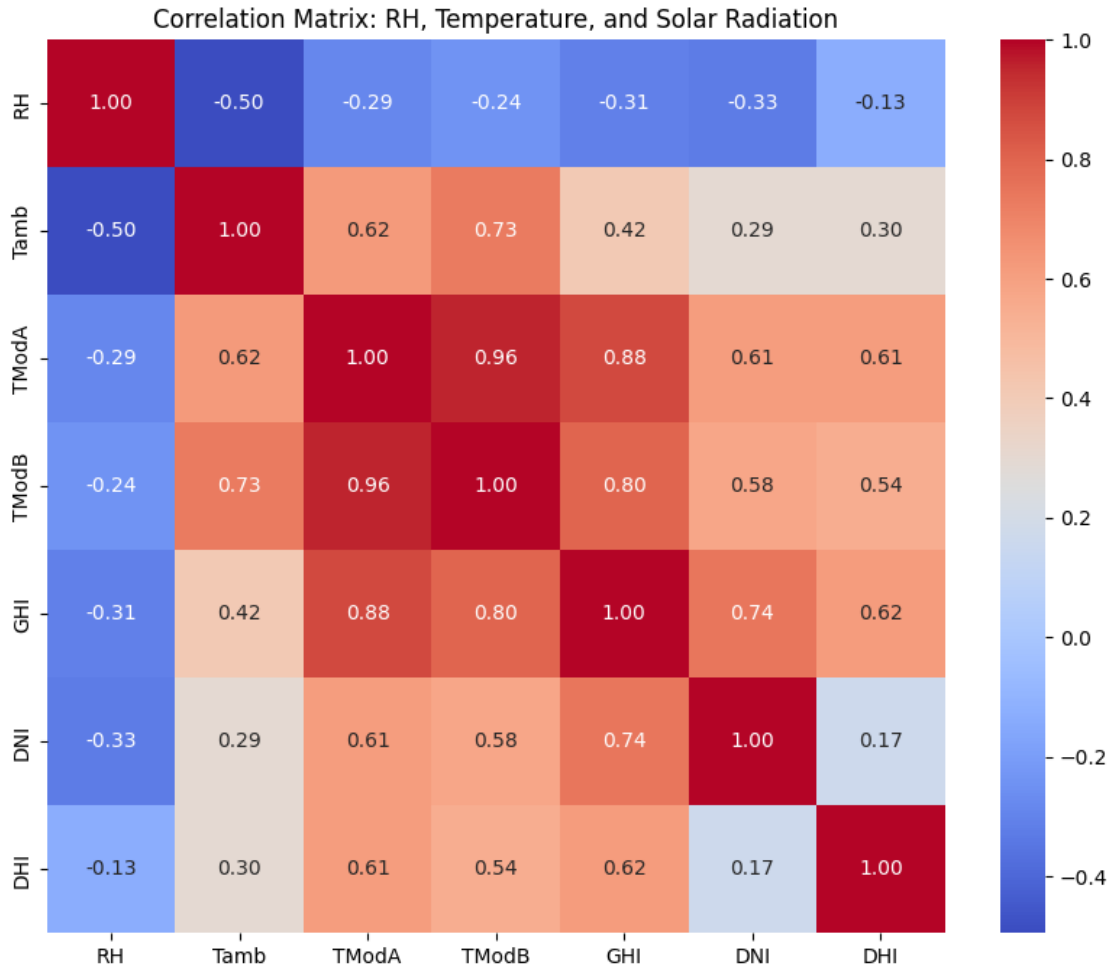


[13]:
```
import matplotlib.pyplot as plt  # Import Matplotlib with alias plt
from windrose import WindroseAxes  # Import WindroseAxes class
# Create a wind rose plot
plt.figure(figsize=(10, 8))
ax = WindroseAxes.from_ax()
ax.bar(df['WD'], df['WS'], normed=True, opening=0.8, edgecolor='white')
ax.set_legend()
plt.title('Wind Rose: Wind Speed and Direction Distribution')
plt.show()
```

```
<Figure size 1000x800 with 0 Axes>
```

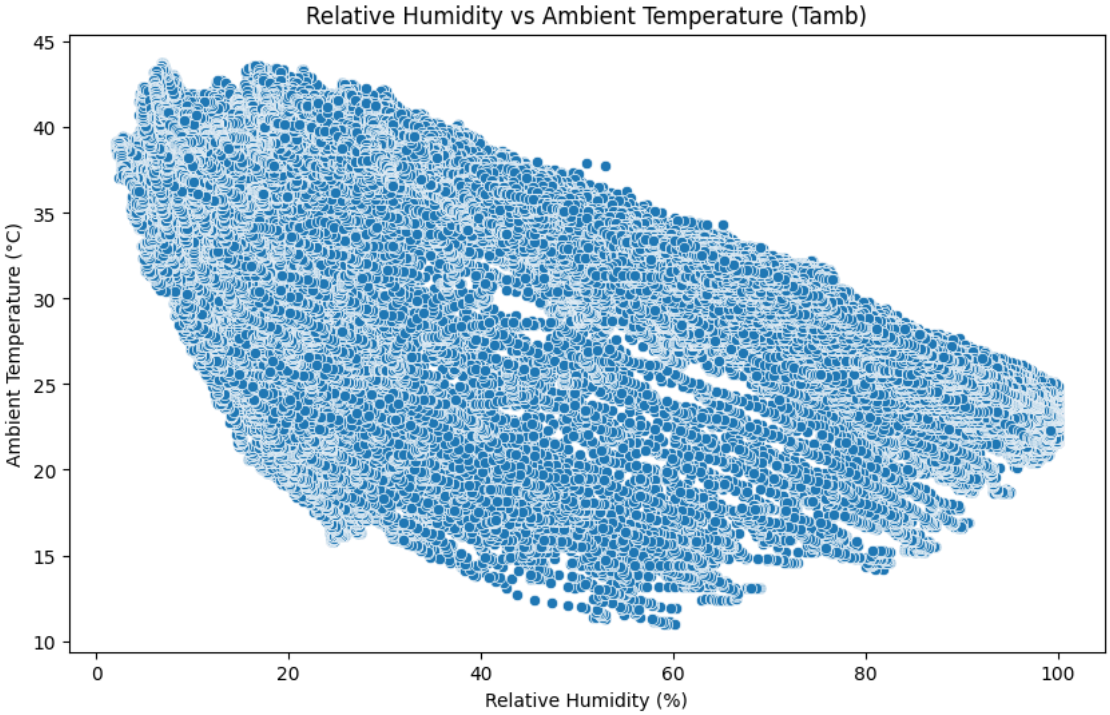


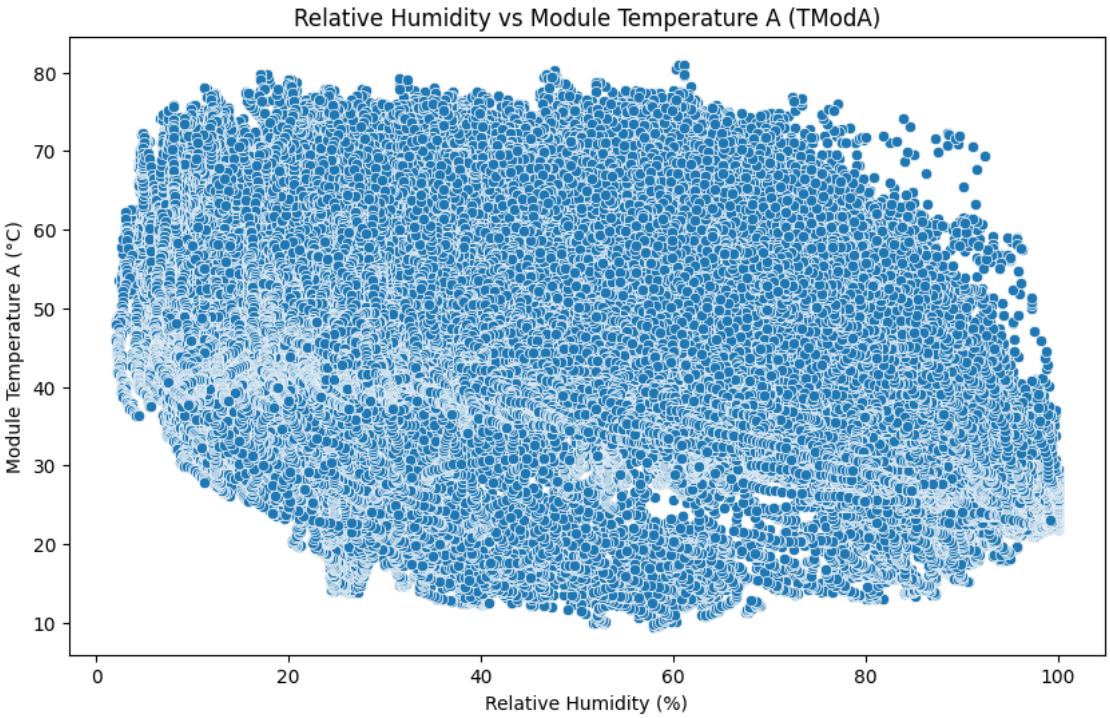
[14]:
```
# Select relevant columns for correlation analysis
rh_temp_solar_cols = ['RH', 'Tamb', 'TModA', 'TModB', 'GHI', 'DNI', 'DHI']

# Calculate the correlation matrix
corr_matrix_rh_temp_solar = df[rh_temp_solar_cols].corr()

# Plot the heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix_rh_temp_solar, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Matrix: RH, Temperature, and Solar Radiation')
plt.show()
```

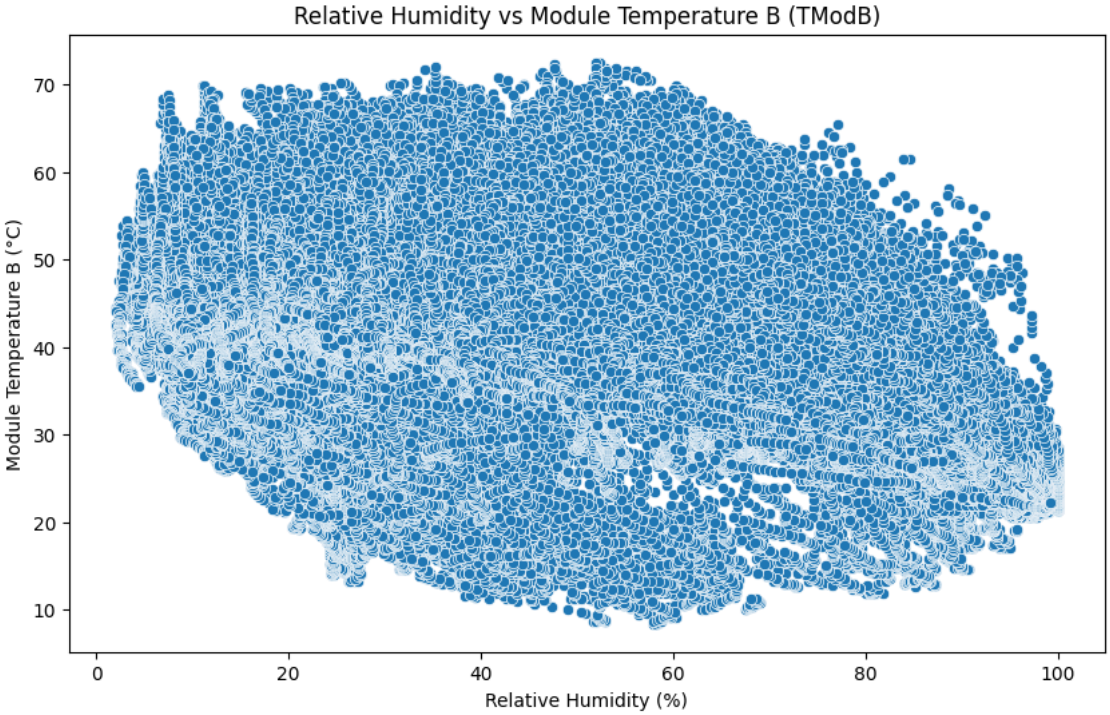


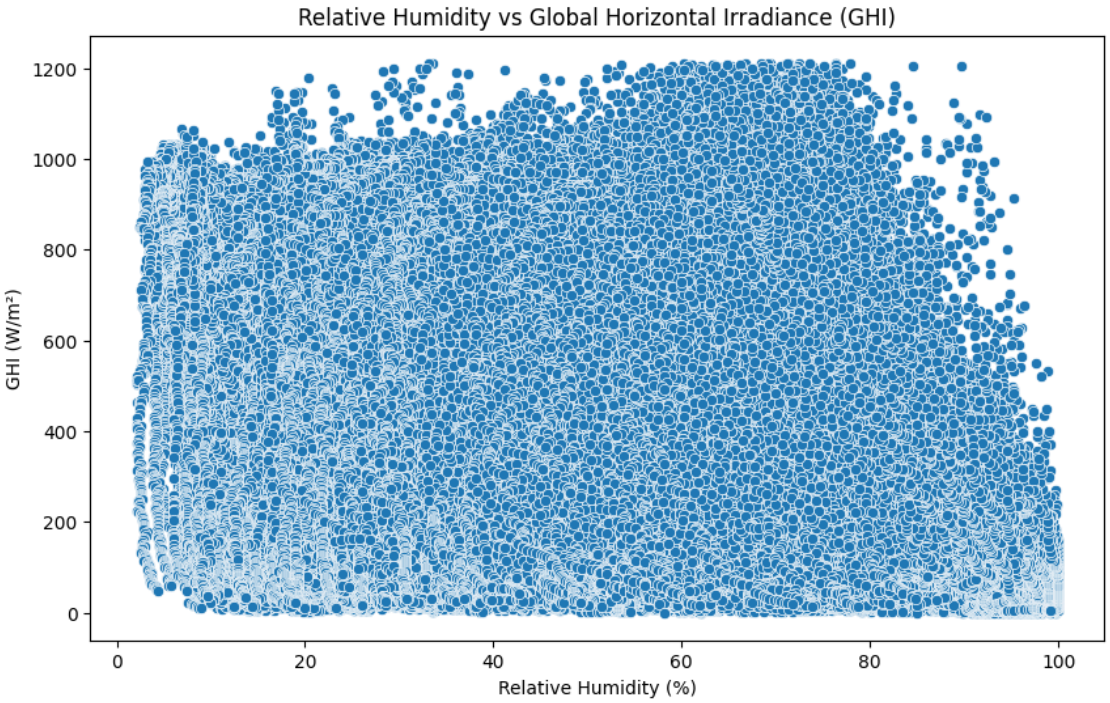
[15]:
```
# Scatter plot for RH vs Tamb
plt.figure(figsize=(10, 6))
sns.scatterplot(x='RH', y='Tamb', data=df)
plt.title('Relative Humidity vs Ambient Temperature (Tamb)')
plt.xlabel('Relative Humidity (%)')
plt.ylabel('Ambient Temperature (°C)')
plt.show()

# Scatter plot for RH vs TModA
plt.figure(figsize=(10, 6))
sns.scatterplot(x='RH', y='TModA', data=df)
plt.title('Relative Humidity vs Module Temperature A (TModA)')
plt.xlabel('Relative Humidity (%)')
plt.ylabel('Module Temperature A (°C)')
plt.show()

# Scatter plot for RH vs TModB
```

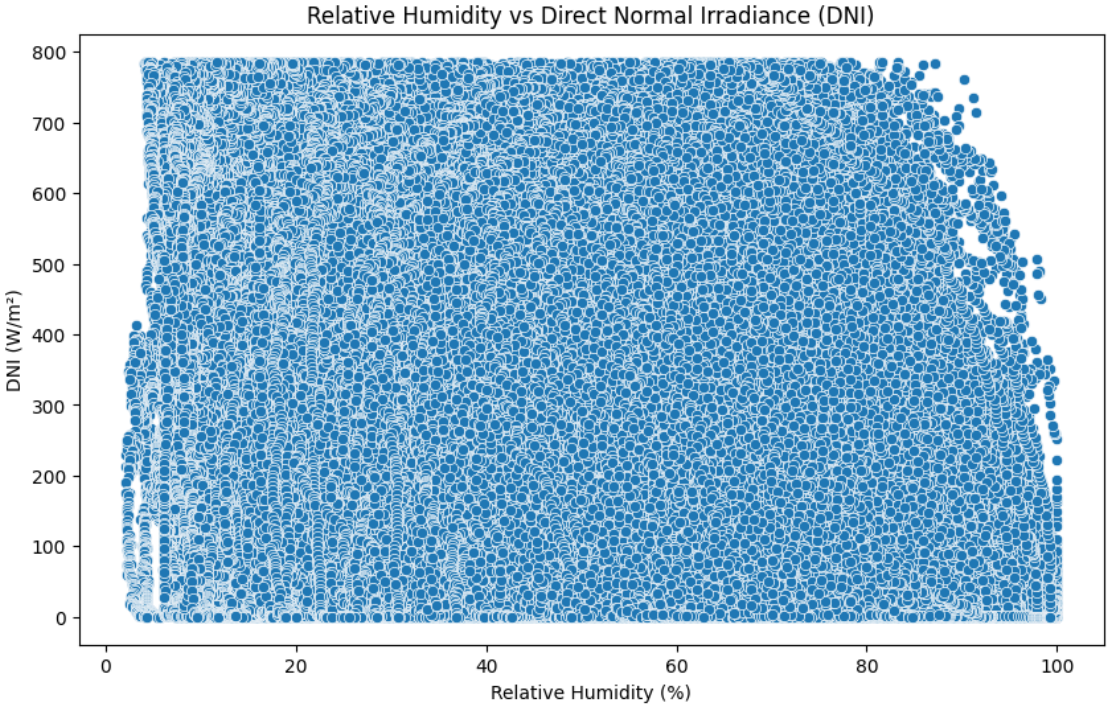
```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='RH', y='TModB', data=df)
plt.title('Relative Humidity vs Module Temperature B (TModB)')
plt.xlabel('Relative Humidity (%)')
plt.ylabel('Module Temperature B (°C)')
plt.show()

# Scatter plot for RH vs GHI
plt.figure(figsize=(10, 6))
sns.scatterplot(x='RH', y='GHI', data=df)
plt.title('Relative Humidity vs Global Horizontal Irradiance (GHI)')
plt.xlabel('Relative Humidity (%)')
plt.ylabel('GHI (W/m²)')
plt.show()

# Scatter plot for RH vs DNI
plt.figure(figsize=(10, 6))
sns.scatterplot(x='RH', y='DNI', data=df)
plt.title('Relative Humidity vs Direct Normal Irradiance (DNI)')
plt.xlabel('Relative Humidity (%)')
plt.ylabel('DNI (W/m²)')
plt.show()

# Scatter plot for RH vs DHI
plt.figure(figsize=(10, 6))
sns.scatterplot(x='RH', y='DHI', data=df)
plt.title('Relative Humidity vs Diffuse Horizontal Irradiance (DHI)')
plt.xlabel('Relative Humidity (%)')
plt.ylabel('DHI (W/m²)')
plt.show()
```

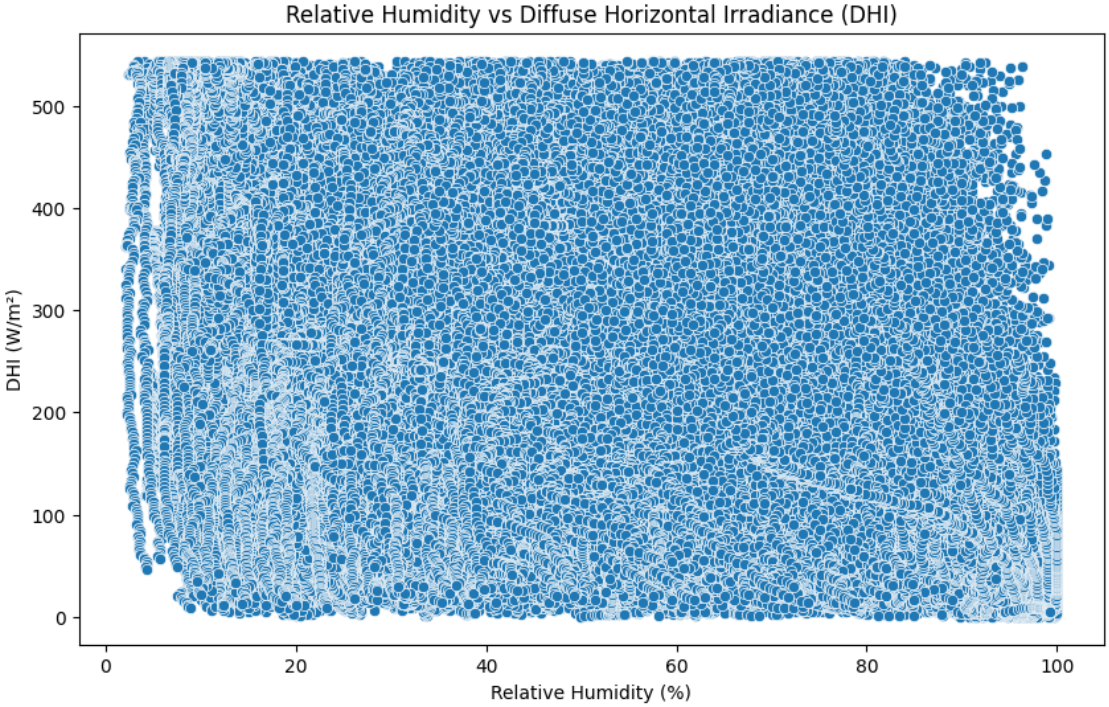

Relative Humidity vs Ambient Temperature (Tamb)
Ambient Temperature (°C)
Relative Humidity (%)
45
40
35
30
25
20
15
10
0
20
40
60
80
100


Relative Humidity vs Module Temperature A (TModA)
Module Temperature A (°C)
Relative Humidity (%)
80
70
60
50
40
30
20
10
0
20
40
60
80
100

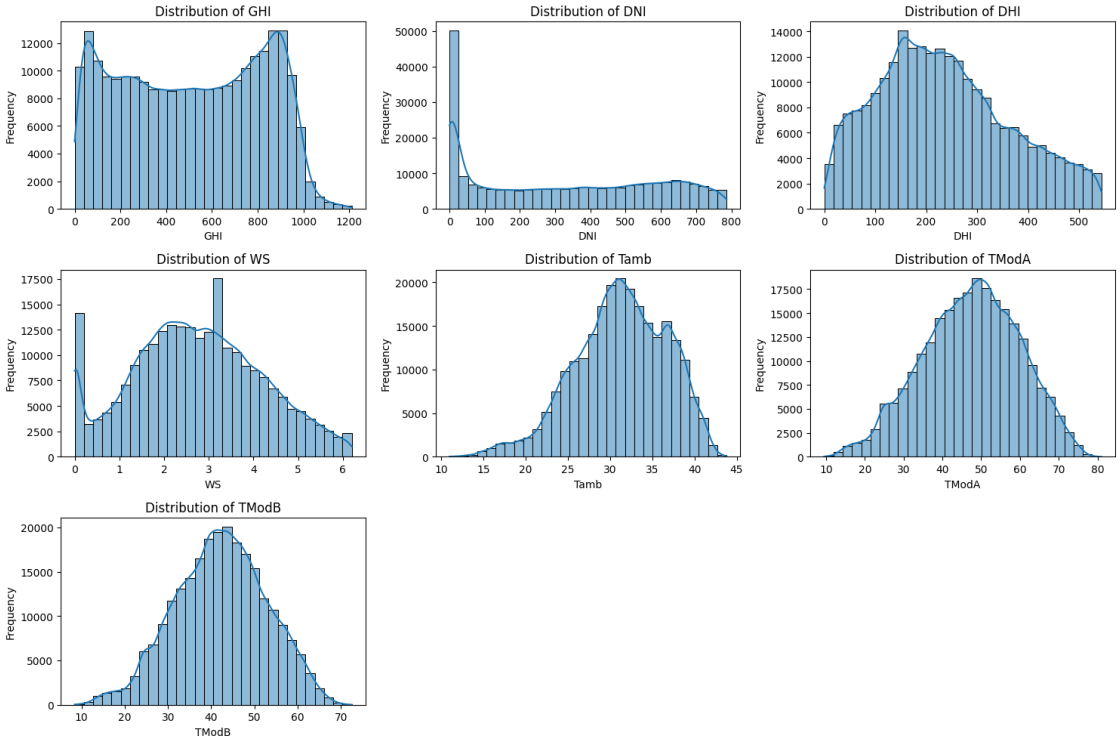

Relative Humidity vs Module Temperature B (TModB)
Module Temperature B (°C)
Relative Humidity (%)
70
60
50
40
30
20
10
0
20
40
60
80
100


Relative Humidity vs Global Horizontal Irradiance (GHI)
GHI (W/m²)
Relative Humidity (%)
1200
1000
800
600
400
200
0
0
20
40
60
80
100


Relative Humidity vs Direct Normal Irradiance (DNI)
DNI (W/m²)
Relative Humidity (%)
0
100
200
300
400
500
600
700
800
0
20
40
60
80
100


Relative Humidity vs Diffuse Horizontal Irradiance (DHI)
DHI (W/m²)
Relative Humidity (%)
0
100
200
300
400
500
0
20
40
60
80
100

[16]:
```
# Variables to visualize
variables = ['GHI', 'DNI', 'DHI', 'WS', 'Tamb', 'TModA', 'TModB']

# Create histograms
plt.figure(figsize=(15, 10))

for i, var in enumerate(variables, 1):
    plt.subplot(3, 3, i)
    sns.histplot(df[var].dropna(), kde=True, bins=30)
    plt.title(f'Distribution of {var}')
    plt.xlabel(var)
    plt.ylabel('Frequency')

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



[17]:
```
# Columns to calculate Z-scores for
columns_to_analyze = ['GHI', 'DNI', 'DHI', 'WS', 'Tamb', 'TModA', 'TModB']

# Calculate Z-scores
z_scores = df[columns_to_analyze].apply(lambda x: (x - x.mean()) / x.std())

# Flag data points with Z-scores greater than 3 or less than -3
```

```
outliers = (z_scores > 3) | (z_scores < -3)

# Add a flag column for outliers
df['Outlier'] = outliers.any(axis=1)

# Display the DataFrame with the Outlier flag
df.head()
```

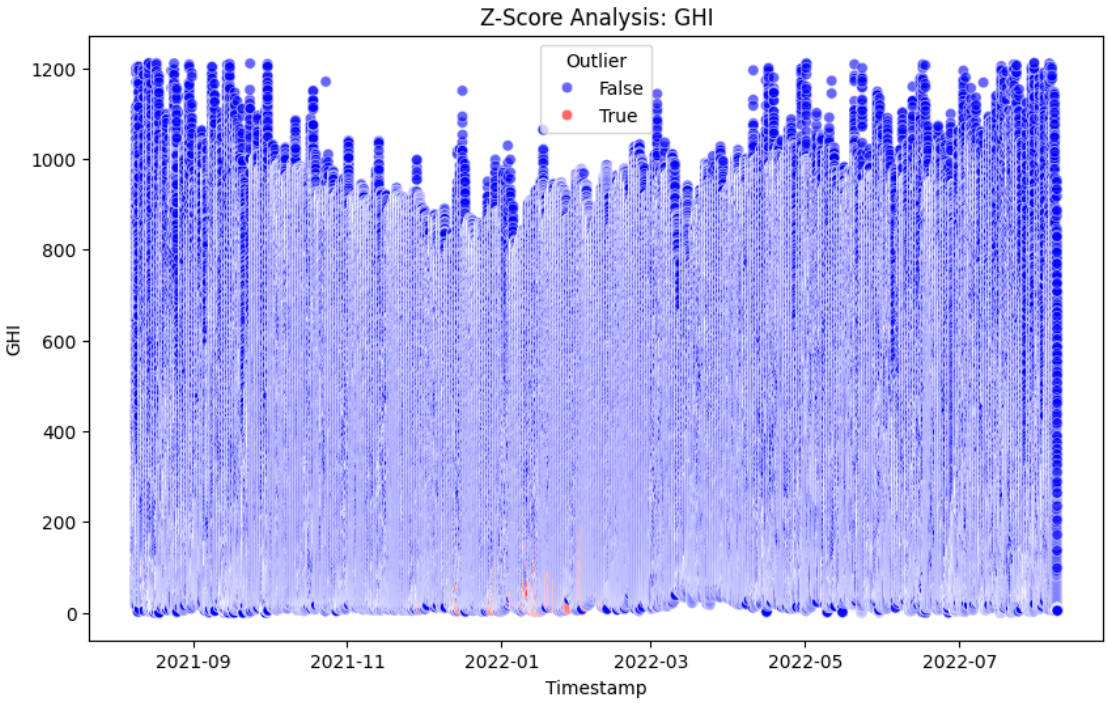
[17]:

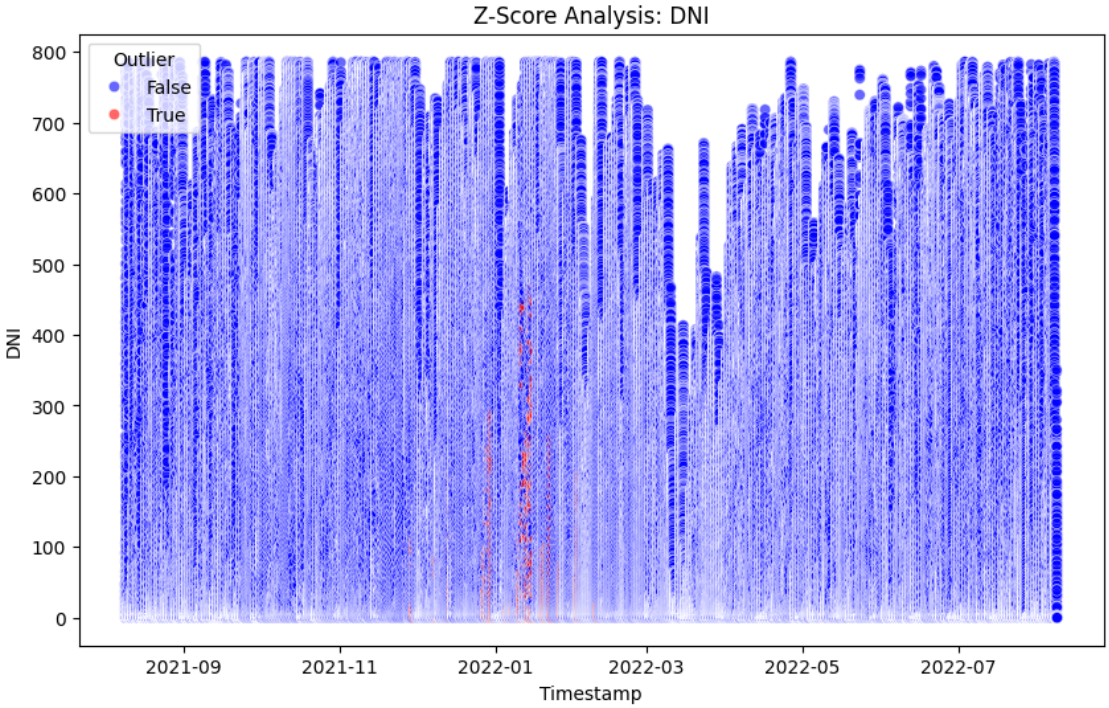
Timestamp	GHI	DNI	DHI	ModA	ModB	Tamb	RH	WS	WSgust
2021-08-09 06:54:00	16.7	0.0	16.5	16.1	16.3	24.2	98.8	0.0	0.0
2021-08-09 06:55:00	18.2	0.1	18.0	17.4	17.6	24.2	98.8	0.0	0.0
2021-08-09 06:56:00	19.7	0.3	19.5	18.7	18.9	24.2	98.8	0.0	0.0
2021-08-09 06:57:00	21.1	0.6	20.9	19.9	20.1	24.2	98.9	0.0	0.0
2021-08-09 06:58:00	22.5	1.1	22.2	21.1	21.3	24.2	98.9	0.0	0.0

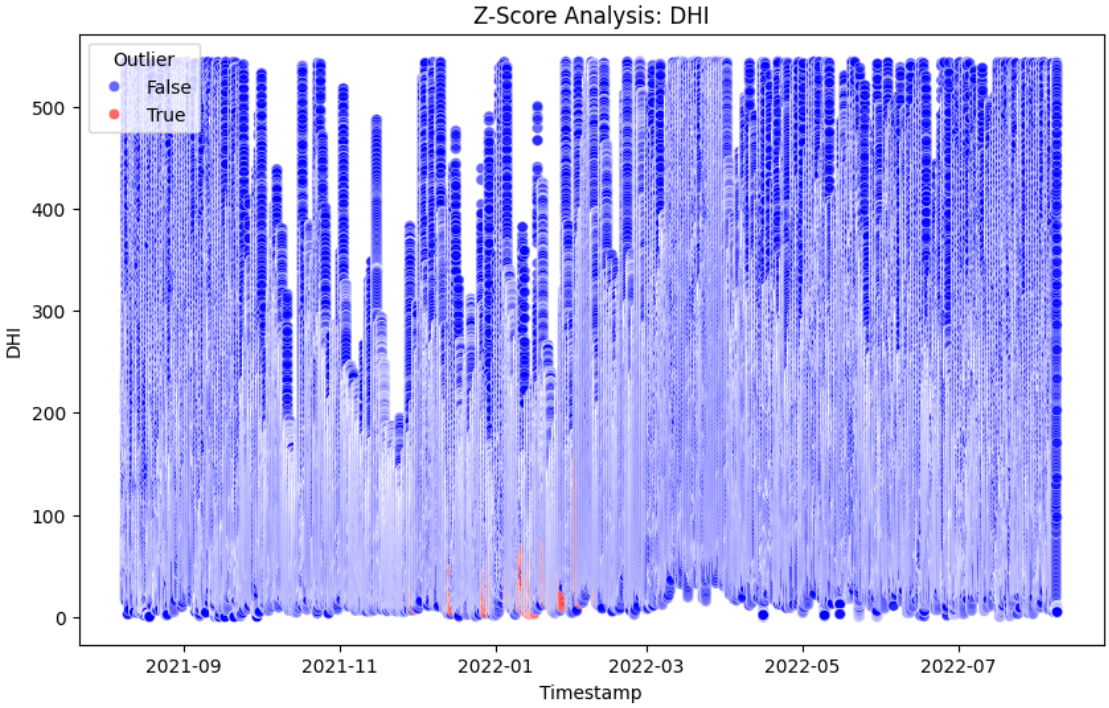
Timestamp	WSstdev	WD	WDstdev	BP	Cleaning	Precipitation
2021-08-09 06:54:00	0.0	0.0	0.0	997	0	0.0
2021-08-09 06:55:00	0.0	0.0	0.0	997	0	0.0
2021-08-09 06:56:00	0.0	0.0	0.0	997	0	0.0
2021-08-09 06:57:00	0.0	0.0	0.0	997	0	0.0
2021-08-09 06:58:00	0.0	0.0	0.0	997	0	0.0

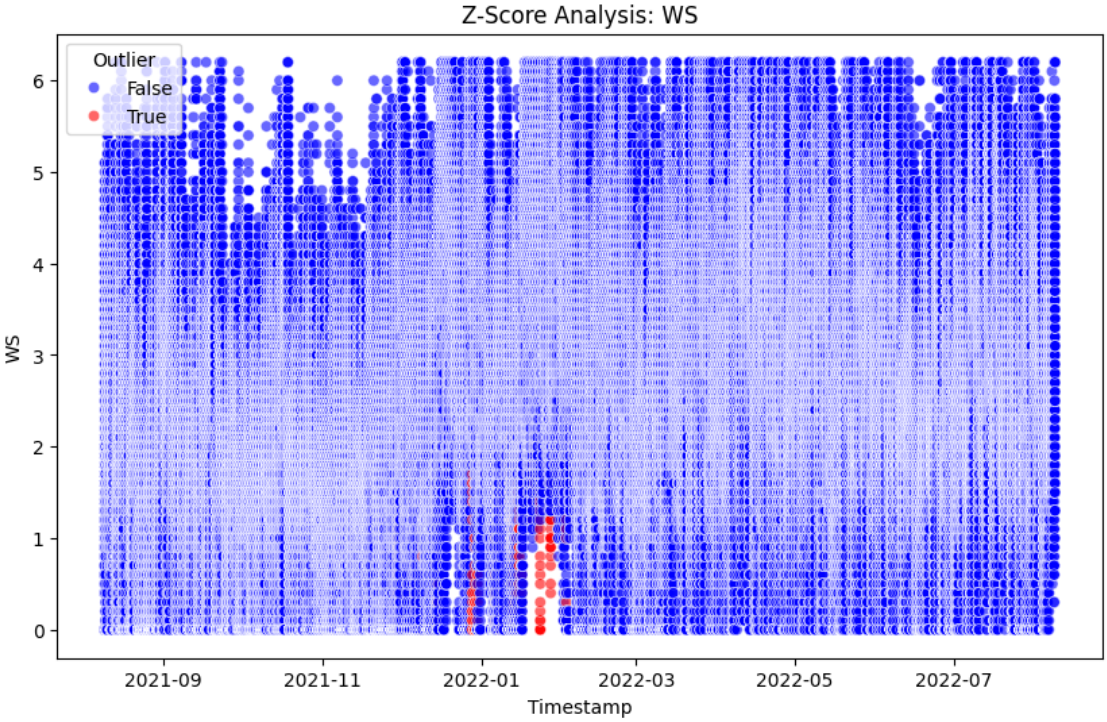
Timestamp	TModA	TModB	Comments	Outlier
2021-08-09 06:54:00	24.2	23.7	NaN	False
2021-08-09 06:55:00	24.3	23.8	NaN	False
2021-08-09 06:56:00	24.3	23.9	NaN	False
2021-08-09 06:57:00	24.4	23.9	NaN	False
2021-08-09 06:58:00	24.5	24.0	NaN	False

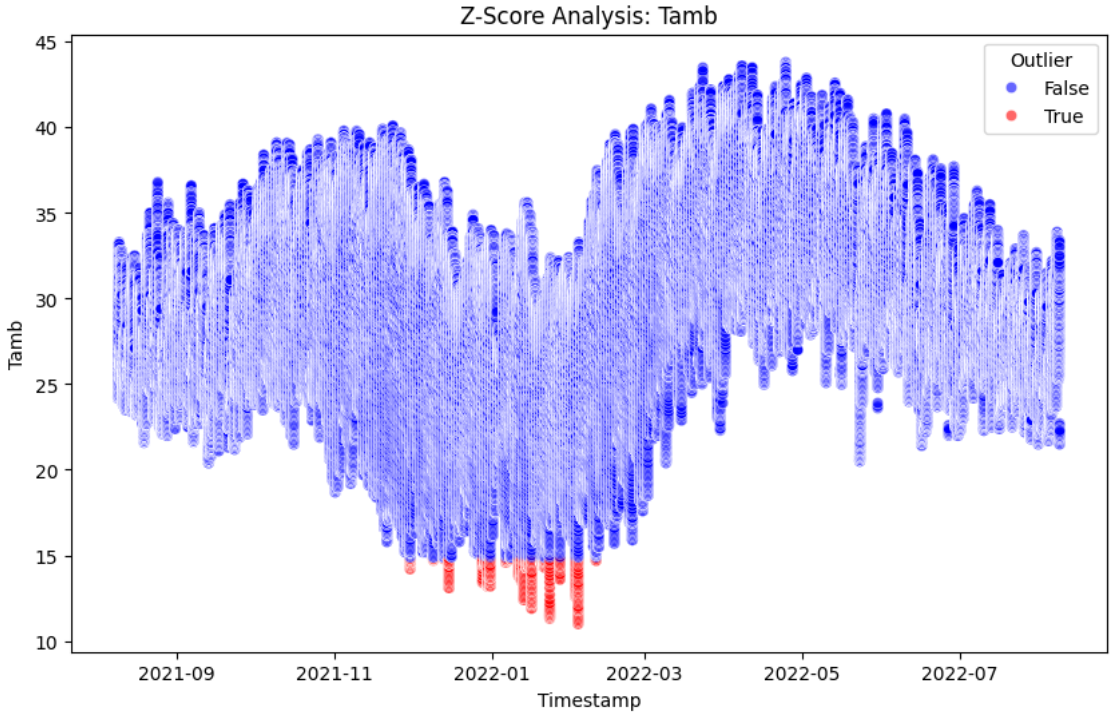
[18]:
```
# Plotting function for each variable
def plot_z_scores(variable):
    plt.figure(figsize=(10, 6))
    sns.scatterplot(data=df, x='Timestamp', y=variable, hue='Outlier',␣
 ↪palette={True: 'red', False: 'blue'}, alpha=0.6)
    plt.title(f'Z-Score Analysis: {variable}')
    plt.xlabel('Timestamp')
    plt.ylabel(variable)
    plt.legend(title='Outlier', loc='best')
    plt.show()

# Plot for each variable
for var in columns_to_analyze:
    plot_z_scores(var)
```

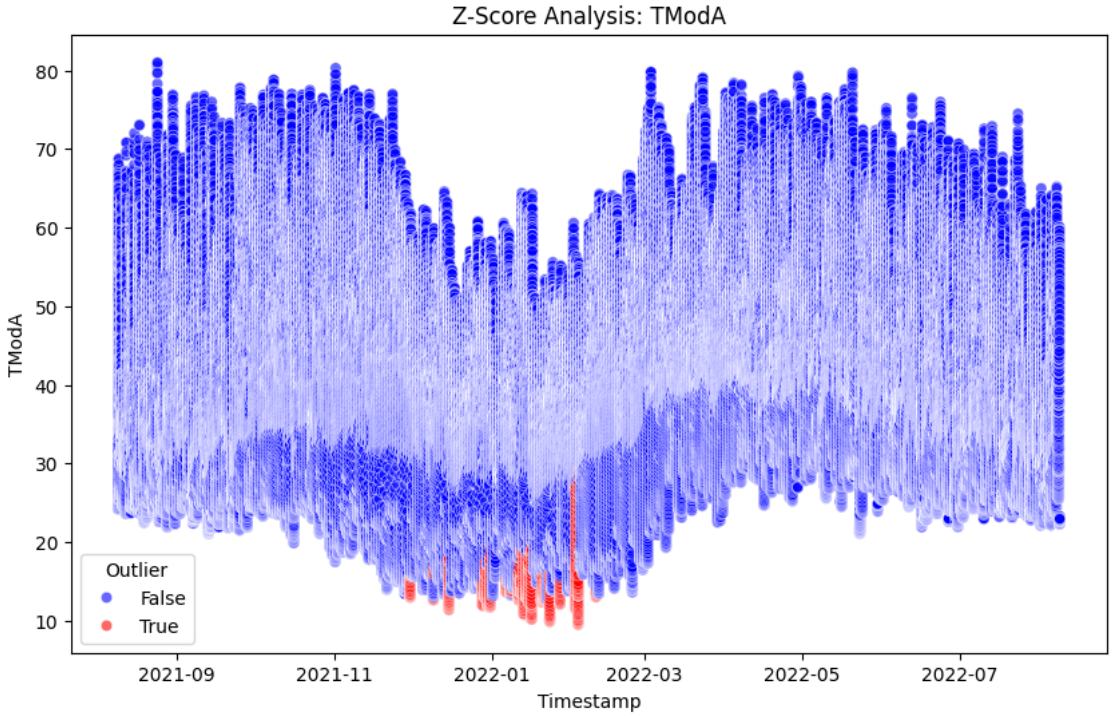

Z-Score Analysis: GHI
Outlier
False
True
GHI
Timestamp
0
200
400
600
800
1000
1200
2021-09
2021-11
2022-01
2022-03
2022-05
2022-07

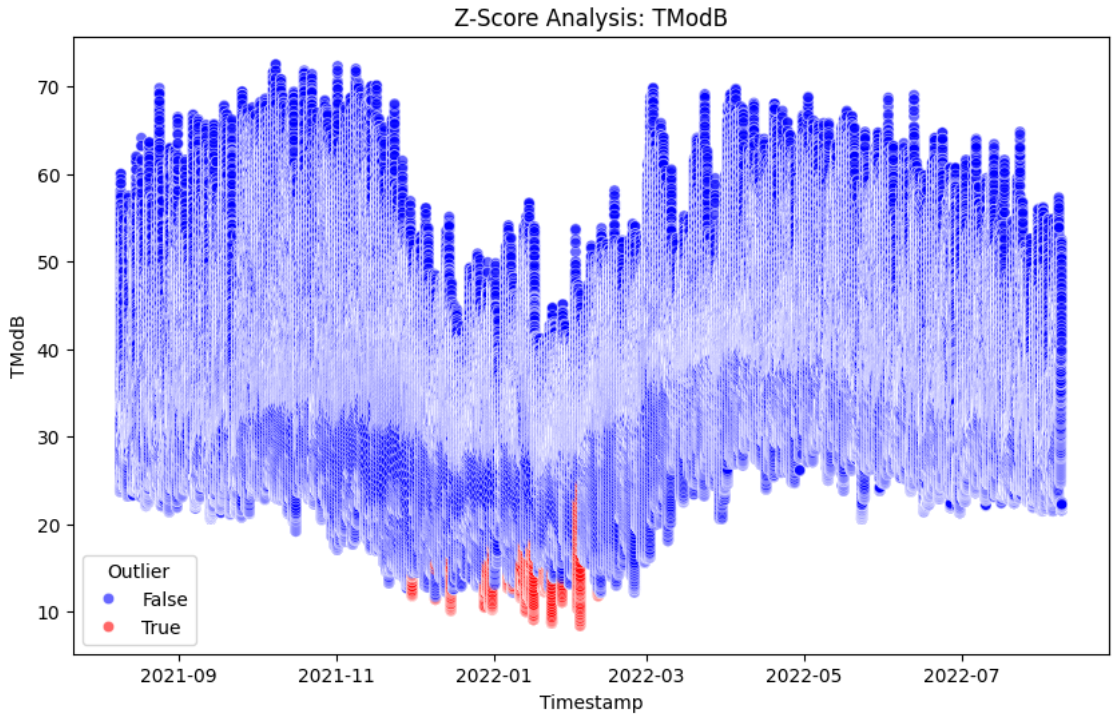

Z-Score Analysis: DNI
Outlier
False
True
DNI
Timestamp
0
100
200
300
400
500
600
700
800
2021-09
2021-11
2022-01
2022-03
2022-05
2022-07


Z-Score Analysis: DHI
Outlier
False
True
DHI
0
100
200
300
400
500
2021-09
2021-11
2022-01
2022-03
2022-05
2022-07
Timestamp

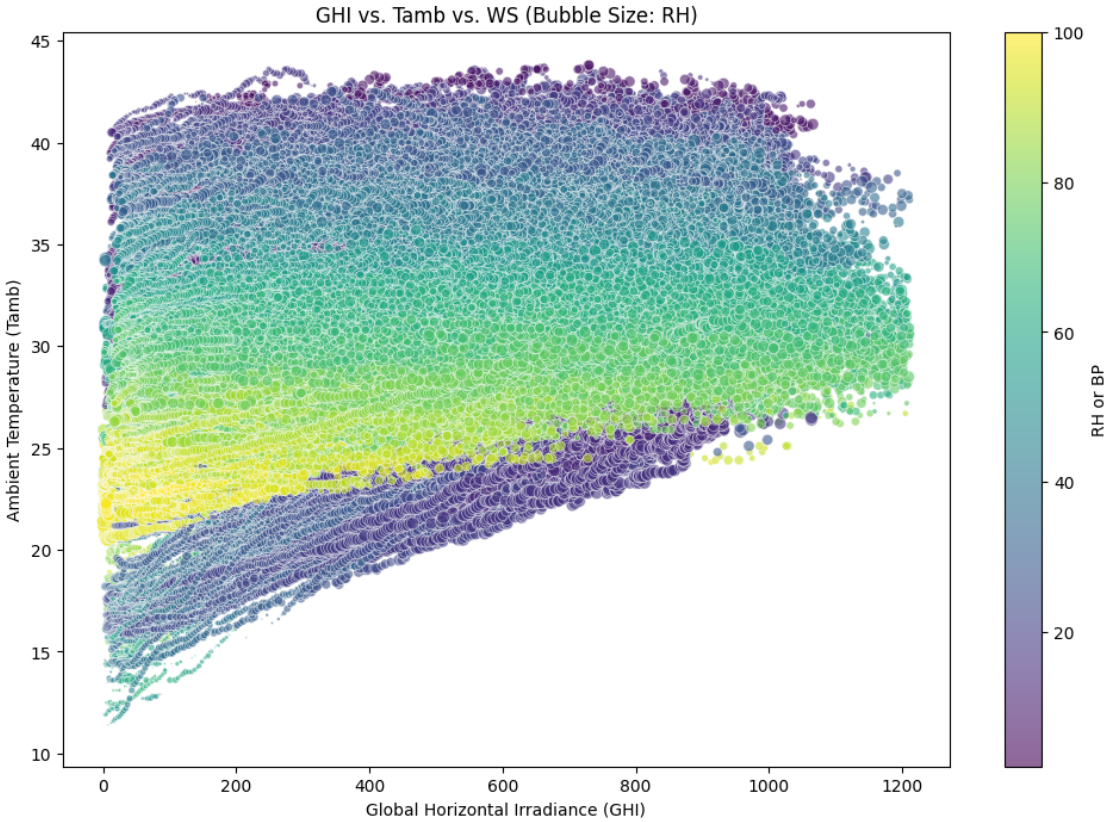

Z-Score Analysis: WS
Outlier
False
True
WS
0
1
2
3
4
5
6
2021-09
2021-11
2022-01
2022-03
2022-05
2022-07
Timestamp

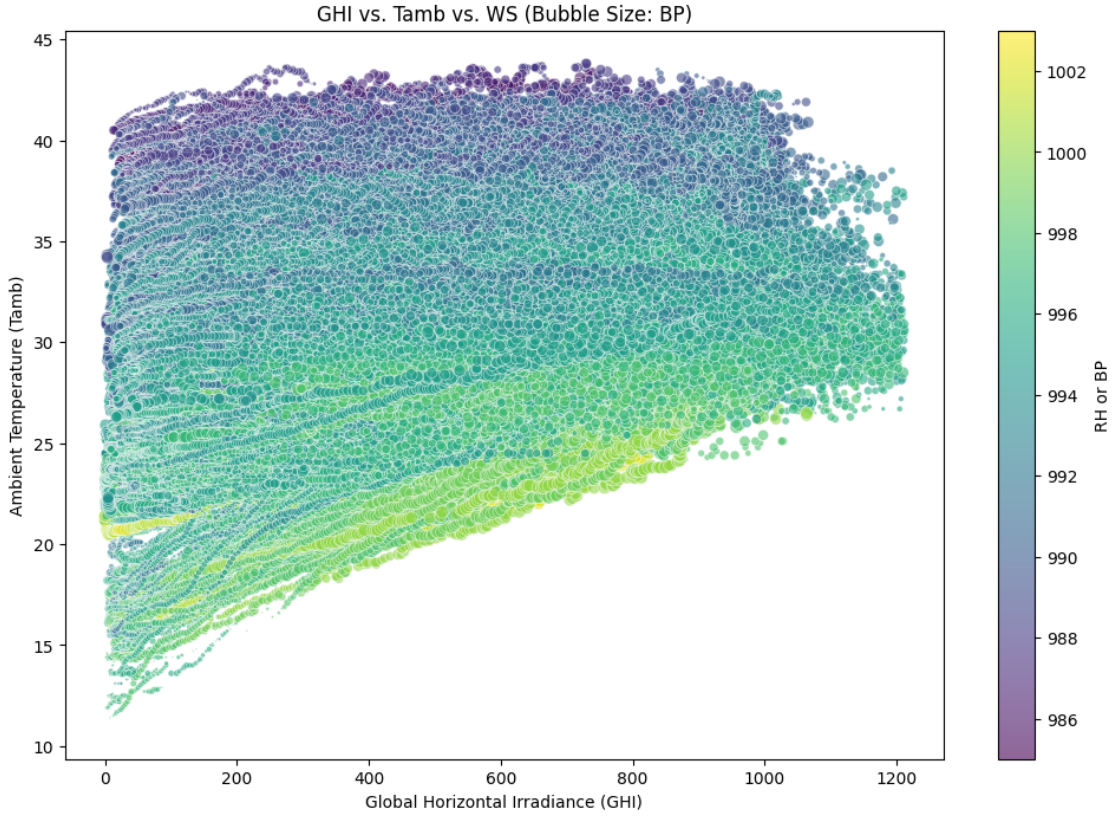

Z-Score Analysis: Tamb
Outlier
False
True
Tamb
Timestamp
45
40
35
30
25
20
15
10
2021-09
2021-11
2022-01
2022-03
2022-05
2022-07


Z-Score Analysis: TModA
Outlier
False
True
TModA
Timestamp
80
70
60
50
40
30
20
10
2021-09
2021-11
2022-01
2022-03
2022-05
2022-07



[19]:
```
# Bubble chart function
def bubble_chart(x, y, size, color, xlabel, ylabel, title):
    plt.figure(figsize=(12, 8))
    scatter = plt.scatter(x, y, s=size, c=color, alpha=0.6, cmap='viridis',␣
 ↪edgecolors='w', linewidth=0.5)
    plt.colorbar(scatter, label='RH or BP')
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.title(title)
    plt.show()

# Plot GHI vs. Tamb vs. WS with bubble size representing RH
bubble_chart(df['GHI'], df['Tamb'], df['WS']*10, df['RH'], 'Global Horizontal␣
 ↪Irradiance (GHI)', 'Ambient Temperature (Tamb)', 'GHI vs. Tamb vs. WS␣
 ↪(Bubble Size: RH)')

# Plot GHI vs. Tamb vs. WS with bubble size representing BP
bubble_chart(df['GHI'], df['Tamb'], df['WS']*10, df['BP'], 'Global Horizontal␣
 ↪Irradiance (GHI)', 'Ambient Temperature (Tamb)', 'GHI vs. Tamb vs. WS␣
 ↪(Bubble Size: BP)')
```


GHI vs. Tamb vs. WS (Bubble Size: RH)
Ambient Temperature (Tamb)
Global Horizontal Irradiance (GHI)
RH or BP
45
40
35
30
25
20
15
10
0
200
400
600
800
1000
1200
100
80
60
40
20


GHI vs. Tamb vs. WS (Bubble Size: BP)
Ambient Temperature (Tamb)
Global Horizontal Irradiance (GHI)
RH or BP
45
40
35
30
25
20
15
10
0
200
400
600
800
1000
1200
1002
1000
998
996
994
992
990
988
986