To visualize the provided data using pandas and matplotlib in a way that conveys the information effectively, you can create various types of plots and charts based on the specific aspects of the data you want to showcase. Here are some common types of plots and charts that can help you convey the data:

1. **Time Series Plot**: If you want to visualize how some of the numeric values change over time (e.g., WSPD, ATMP), you can create a time series plot. This will show how these values evolve over the recorded time. You can use the `RTC` column as the x-axis (time) and plot various numeric values against it.
2. **Wind Rose Plot**: Since you have wind-related data (WSPD and WDIR), you can create a Wind Rose plot to visualize wind speed and direction patterns. The WindroseAxes you mentioned earlier is suitable for this purpose. You can customize it to make it more colorful and informative, as shown in a previous response.
3. **Temperature and Humidity Plot**: Create a line or area plot to display the variations in temperature (ATMP) and humidity (HUMD) over time. This can help viewers understand how these environmental factors change.
4. **Bar Chart**: If you want to compare values between different records or devices, a bar chart can be useful. For instance, you can compare WSPD values for different records or devices using a bar chart.
5. **Scatter Plot**: If you want to explore relationships between two variables (e.g., WSPD and ATMP), a scatter plot can help visualize correlations and trends.
6. **Box Plot**: A box plot can be used to visualize the distribution of numerical values like WSPD, ATMP, or HUMD. It shows the median, quartiles, and potential outliers in the data.

Here's a simple example of how to create a time series plot using pandas and matplotlib:

```
import pandas as pd
import matplotlib.pyplot as plt

# Assuming your data is in a DataFrame named 'df'
# Convert the 'RTC' column to the datetime format if it's not already
df['RTC'] = pd.to_datetime(df['RTC'])

# Set 'RTC' as the index for time series plotting
df.set_index('RTC', inplace=True)

# Create a time series plot for WSPD and ATMP
plt.figure(figsize=(12, 6))
plt.plot(df.index, df['WSPD'], label='Wind Speed (m/s)')
```

```
plt.plot(df.index, df['ATMP'], label='Temperature (°C)')
plt.xlabel('Time')
plt.ylabel('Value')
plt.title('Wind Speed and Temperature Over Time')
plt.legend()
plt.grid()
plt.show()
```

Depending on your specific goals and the insights you want to convey, you can combine multiple types of plots or create more complex visualizations. The choice of visualization type should be guided by the questions you want to answer and the story you want to tell with your data.

Certainly! To create a plot that visualizes temperature (ATMP) and humidity (HUMD) over time, you can use the following code as a starting point. Make sure you have the necessary Python libraries installed, such as pandas and matplotlib.

```
import pandas as pd
import matplotlib.pyplot as plt

# Assuming your data is in a DataFrame named 'df'
# Convert the 'RTC' column to the datetime format if it's not already
df['RTC'] = pd.to_datetime(df['RTC'])

# Set 'RTC' as the index for time series plotting
df.set_index('RTC', inplace=True)

# Create a time series plot for ATMP (Temperature) and HUMD (Humidity)
plt.figure(figsize=(12, 6))
plt.plot(df.index, df['ATMP'], label='Temperature (°C)', color='blue')
plt.plot(df.index, df['HUMD'], label='Humidity (%)', color='green')
plt.xlabel('Time')
plt.ylabel('Value')
plt.title('Temperature and Humidity Over Time')
plt.legend()
plt.grid()
plt.show()
```

This code will generate a plot that shows how temperature and humidity values change over time, with temperature represented in blue and humidity in green. You can further customize the plot's appearance, such as adding titles, legends, and adjusting colors, to make it more informative and visually appealing.

Certainly! To create a bar chart that compares values between different records or devices, you can use the following code as a starting point. In this example, we'll create

a bar chart to compare WSPD (wind speed) values for different records. You can adjust the code to compare values for different devices or other variables as needed.

```
import pandas as pd
import matplotlib.pyplot as plt

# Assuming your data is in a DataFrame named 'df'

# Select the data you want to compare (e.g., WSPD values)
data_to_compare = df['WSPD']

# Create a bar chart
plt.figure(figsize=(10, 6))
data_to_compare.plot(kind='bar', color='skyblue')
plt.xlabel('Record ID or Device')
plt.ylabel('Wind Speed (m/s)')
plt.title('Comparison of Wind Speed Between Records')
plt.xticks(range(len(data_to_compare)), data_to_compare.index, rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

In this code:

- We assume you have a DataFrame named 'df' that contains your data.
- We select the 'WSPD' column as the data to compare.
- We create a bar chart using the `plot(kind='bar')` method with 'skyblue' as the bar color.
- The x-axis represents the record ID or device, and the y-axis represents wind speed (WSPD).
- We set the chart title, labels, and customize the appearance.
- We use `xticks` to label each bar with the corresponding record ID or device.
- We add grid lines to make the chart more readable.

You can modify the code to compare different variables or devices by selecting the appropriate data in the `data_to_compare` variable.

If you want to include only the numeric columns with decimal values (e.g., Decimal('13.04300000')), you can use the `select_dtypes` method to filter columns with the 'decimal' data type. To ensure you're selecting columns with decimal values, you should first convert them to a decimal data type using the `to_numeric` method.

Here's how you can modify the code to include only decimal columns:

```
import pandas as pd
import matplotlib.pyplot as plt

# Assuming your data is in a DataFrame named 'df'

# Convert relevant columns to decimal data type
decimal_columns = ['WSPD', 'WDIR', 'ATMP', 'HUMD', 'RAIN', 'SRAD', 'BPRS',
'WDCH', 'DWPT', 'P12', 'P13', 'P14', 'P15', 'P16']
df[decimal_columns] = df[decimal_columns].apply(pd.to_numeric,
errors='coerce', downcast='float')

# Select only the decimal columns
decimal_df = df[decimal_columns]

# Calculate the mean of each decimal column
mean_values = decimal_df.mean()

# Create a bar chart to display the means
plt.figure(figsize=(12, 6))
mean_values.plot(kind='bar', color='lightblue')
plt.xlabel('Variables')
plt.ylabel('Mean Value')
plt.title('Mean Values of Decimal Variables')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

In this code:

- We first specify the columns that should be converted to decimal data type in the `decimal_columns` list.
- We use `apply` with `pd.to_numeric` to convert the specified columns to decimal data type.
- We then select only the decimal columns using `df[decimal_columns]`.
- The mean is calculated for these decimal columns.
- The bar chart is created based on the mean values of the decimal variables.

This code ensures that only columns with decimal values are included in the calculation of the mean and the creation of the bar chart.

To create a scatter plot between two values, 'ATMP' (Temperature) and 'HUMD' (Humidity), you can use the following code. This code assumes you have a DataFrame named 'df' with the relevant data:

```
import pandas as pd
import matplotlib.pyplot as plt
```

```
# Assuming your data is in a DataFrame named 'df'

# Create a scatter plot between 'ATMP' and 'HUMD'
plt.figure(figsize=(8, 8))
plt.scatter(df['ATMP'], df['HUMD'], c='blue', alpha=0.5)
plt.xlabel('Temperature (°C)')
plt.ylabel('Humidity (%)')
plt.title('Scatter Plot of Temperature vs. Humidity')
plt.grid()
plt.show()
```

In this code:

- We use the `scatter` function to create a scatter plot, with 'ATMP' on the x-axis and 'HUMD' on the y-axis.
- The 'c' parameter sets the color of the points, and 'alpha' controls the transparency of the points.
- We add labels to the axes, a title, and a grid for better readability.

This scatter plot will help you visualize the relationship or distribution between temperature (ATMP) and humidity (HUMD) in your data. Each point represents a data record, and its position on the plot indicates both temperature and humidity values.

To create a box plot for the distribution of numerical values like 'WSPD', 'ATMP', or 'HUMD,' you can use the following code. This code assumes you have a DataFrame named 'df' with the relevant data:

```
import pandas as pd
import matplotlib.pyplot as plt

# Assuming your data is in a DataFrame named 'df'

# Select the numerical columns you want to create box plots for
decimal_columns = ['WSPD', 'WDIR', 'ATMP', 'HUMD', 'RAIN', 'SRAD', 'BPRS',
'WDCH', 'DWPT', 'P12', 'P13', 'P14', 'P15', 'P16']
df[decimal_columns] = df[decimal_columns].apply(pd.to_numeric,
errors='coerce', downcast='float')

# Create a box plot for the selected numerical columns
plt.figure(figsize=(10, 6))
df[decimal_columns].plot(kind='box', vert=False, sym='r+')
plt.xlabel('Value')
plt.title('Box Plot of Numerical Variables')
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.show()
```

In this code:

- We first specify the numerical columns for which you want to create box plots in the `numerical_columns` list.
- We use the `plot(kind='box')` method to create a box plot for the selected numerical columns.
- The 'vert' parameter is set to `False` to create horizontal box plots, and 'sym' is set to 'r+' to display red crosses for potential outliers.
- We add labels, a title, and a grid for better readability.

This box plot will help you visualize the distribution of numerical values and identify potential outliers in your data for the selected columns.