CL-I DMV 8

July 16, 2025

[2]:
```
''' NAME:Aher Swami Sandip
ROLL NO.01
COURSE: AI&DS
CLASS: BE
SUB:Computer Laboratory-I (DMV) '''
```

[87]:
```
pip install folium
```

```
Collecting folium
  Downloading folium-0.20.0-py2.py3-none-any.whl.metadata (4.2 kB)
Collecting branca>=0.6.0 (from folium)
  Downloading branca-0.8.1-py3-none-any.whl.metadata (1.5 kB)
Requirement already satisfied: jinja2>=2.9 in d:\anaconda3\lib\site-packages
(from folium) (3.1.4)
Requirement already satisfied: numpy in d:\anaconda3\lib\site-packages (from
folium) (1.26.4)
Requirement already satisfied: requests in d:\anaconda3\lib\site-packages (from
folium) (2.32.3)
Requirement already satisfied: xyzservices in d:\anaconda3\lib\site-packages
(from folium) (2022.9.0)
Requirement already satisfied: MarkupSafe>=2.0 in d:\anaconda3\lib\site-packages
(from jinja2>=2.9->folium) (2.1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in
d:\anaconda3\lib\site-packages (from requests->folium) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in d:\anaconda3\lib\site-packages
(from requests->folium) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in d:\anaconda3\lib\site-
packages (from requests->folium) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in d:\anaconda3\lib\site-
packages (from requests->folium) (2025.6.15)
Downloading folium-0.20.0-py2.py3-none-any.whl (113 kB)
Downloading branca-0.8.1-py3-none-any.whl (26 kB)
Installing collected packages: branca, folium
Successfully installed branca-0.8.1 folium-0.20.0
Note: you may need to restart the kernel to use updated packages.
```

0.1 Interacting with Web APIs

[16]:
```
# Problem Statement: Analyzing Weather Data from OpenWeatherMap API
# Dataset: Weather data retrieved from OpenWeatherMap API
# Description: The goal is to interact with the OpenWeatherMap API to retrieve␣
 ↪weather data
# for a specific location and perform data modeling and visualization to␣
 ↪analyze weather
# patterns over time.
# Tasks to Perform:
```

0.1.1 1. Register and obtain API key from OpenWeatherMap.

0.1.2 2. Interact with the OpenWeatherMap API using the API key to retrieve weather data for a specific location.

[1]:
```
import requests

api_key = '18896d417b61718248835471bedf9b54'
location = 'Pune'
url = f'http://api.openweathermap.org/data/2.5/weather?
 ↪q={location}&appid={api_key}'

response = requests.get(url)
weather_data = response.json()

# Display the retrieved weather data
print(weather_data)
```

```
{'coord': {'lon': 73.8553, 'lat': 18.5196}, 'weather': [{'id': 804, 'main':
'Clouds', 'description': 'overcast clouds', 'icon': '04n'}], 'base': 'stations',
'main': {'temp': 298.74, 'feels_like': 299.52, 'temp_min': 298.74, 'temp_max':
298.74, 'pressure': 1006, 'humidity': 83, 'sea_level': 1006, 'grnd_level': 934},
'visibility': 10000, 'wind': {'speed': 3, 'deg': 253, 'gust': 5.6}, 'clouds':
{'all': 96}, 'dt': 1752677882, 'sys': {'country': 'IN', 'sunrise': 1752626190,
'sunset': 1752673460}, 'timezone': 19800, 'id': 1259229, 'name': 'Pune', 'cod':
200}
```

[3]:
```
# Extract relevant weather attributes
temperature = weather_data['main']['temp'] - 273.15  # Convert from Kelvin to␣
 ↪Celsius
humidity = weather_data['main']['humidity']
wind_speed = weather_data['wind']['speed']
weather_description = weather_data['weather'][0]['description']
```

0.1.3 3. Extract relevant weather attributes such as temperature, humidity, wind speed, and precipitation from the API response.

[5]:
```
print(f"Temperature: {temperature}°C")
print(f"Humidity: {humidity}%")
print(f"Wind Speed: {wind_speed} m/s")
print(f"Weather Description: {weather_description}")
```

```
Temperature: 25.590000000000032°C
Humidity: 83%
Wind Speed: 3 m/s
Weather Description: overcast clouds
```

[7]:
```
import pandas as pd
from datetime import datetime

# Example data collection over time
data = {
    'datetime': [datetime.now()],
    'temperature': [temperature],
    'humidity': [humidity],
    'wind_speed': [wind_speed],
    'weather_description': [weather_description]
}

df = pd.DataFrame(data)
print(f"Temperature: {temperature}°C")
print(f"Humidity: {humidity}%")
print(f"Wind Speed: {wind_speed} m/s")
print(f"Weather Description: {weather_description}")
df.head()
```

```
Temperature: 25.590000000000032°C
Humidity: 83%
Wind Speed: 3 m/s
Weather Description: overcast clouds
```

[7]:

	datetime	temperature	humidity	wind_speed	weather_description
0	2025-07-16 20:29:16.858013	25.59	83	3	overcast clouds

[9]:
```
# For example, calculating average values
average_temperature = df['temperature'].mean()
average_humidity = df['humidity'].mean()

print(f"Average Temperature: {average_temperature}°C")
print(f"Average Humidity: {average_humidity}%")
```

```
Average Temperature: 25.590000000000032°C
Average Humidity: 83.0%
```

0.1.4 4. Clean and preprocess the retrieved data, handling missing values or inconsistent formats.

[14]:
```
df.isnull().sum()
```

[14]:
```
datetime              0
temperature           0
humidity              0
wind_speed            0
weather_description   0
dtype: int64
```

[16]:
```
df.dtypes
```

[16]:
```
datetime              datetime64[ns]
temperature                  float64
humidity                       int64
wind_speed                     int64
weather_description           object
dtype: object
```

[18]:
```
from sklearn.preprocessing import LabelEncoder

encoder = LabelEncoder()
df["weather_description"] = encoder.fit_transform(df["weather_description"])
df["datetime"] = encoder.fit_transform(df["datetime"])
```

0.1.5 5. Perform data modeling to analyze weather patterns, such as calculating average temperature, maximum/minimum values, or trends over time.

[20]:
```
average_temp = df["temperature"].mean()  # Replace "Temperature" with your␣
 ↪actual column name
print(f"Average Temperature: {average_temp:.2f}")  # Format to two decimal␣
 ↪places
```

```
Average Temperature: 25.59
```

[63]:
```
max_temp = df["temperature"].max()
min_temp = df["temperature"].min()

print(f"Maximum Temperature: {max_temp:.2f}")
print(f"Minimum Temperature: {min_temp:.2f}")
```

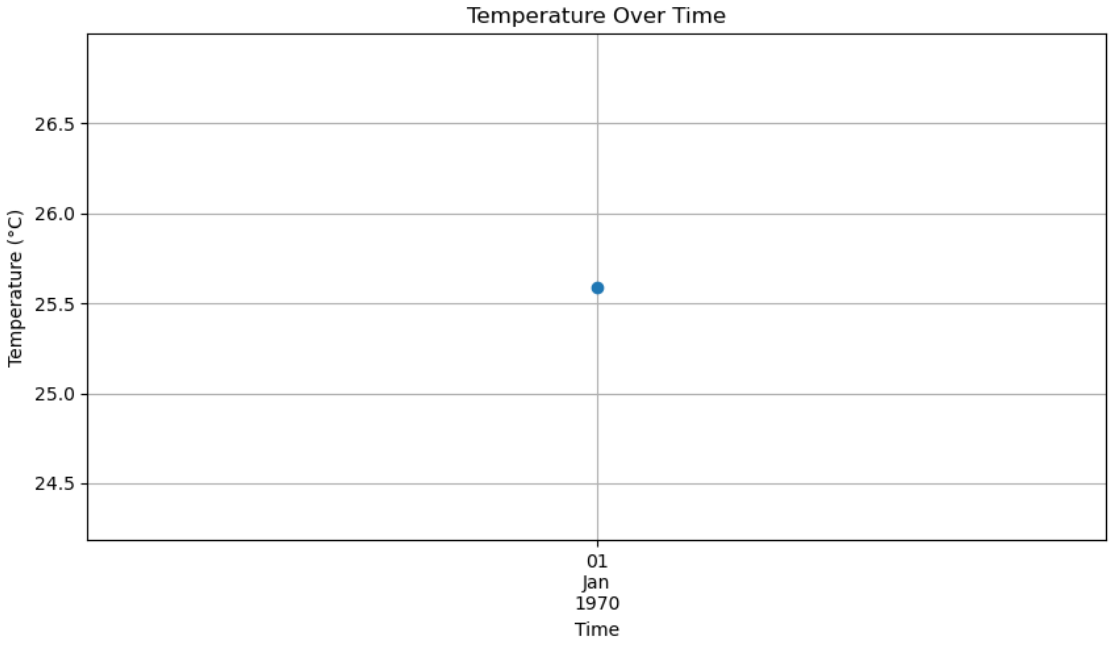
```
Maximum Temperature: 26.49
Minimum Temperature: 26.49
```

[22]:
```
df["datetime"] = pd.to_datetime(df["datetime"])  # Assuming you have a "date" column

# Daily average temperature
daily_average_temp = df.resample("D", on="datetime")["temperature"].mean()  # Resample by day
print(daily_average_temp.head())  # Print first few days

# You can similarly calculate weekly, monthly, or yearly averages by changing "D" to "W", "M", or "Y" in resample
```

```
datetime
1970-01-01    25.59
Freq: D, Name: temperature, dtype: float64
```

[24]:
```
#trends over time
# Line chart for temperature over time
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 5))
daily_average_temp.plot( marker='o',linestyle='-')
plt.title('Temperature Over Time')
plt.xlabel('Time')
plt.ylabel('Temperature (°C)')
plt.grid(True)
plt.show()
```

```
D:\Anaconda3\Lib\site-packages\pandas\plotting\_matplotlib\core.py:1561:
UserWarning: Attempting to set identical low and high xlims makes transformation
singular; automatically expanding.
  ax.set_xlim(left, right)
```



0.1.6 6. Visualize the weather data using appropriate plots, such as line charts, bar plots, or scatter plots, to represent

0.1.7 temperature changes, precipitation levels, or wind speed variations.

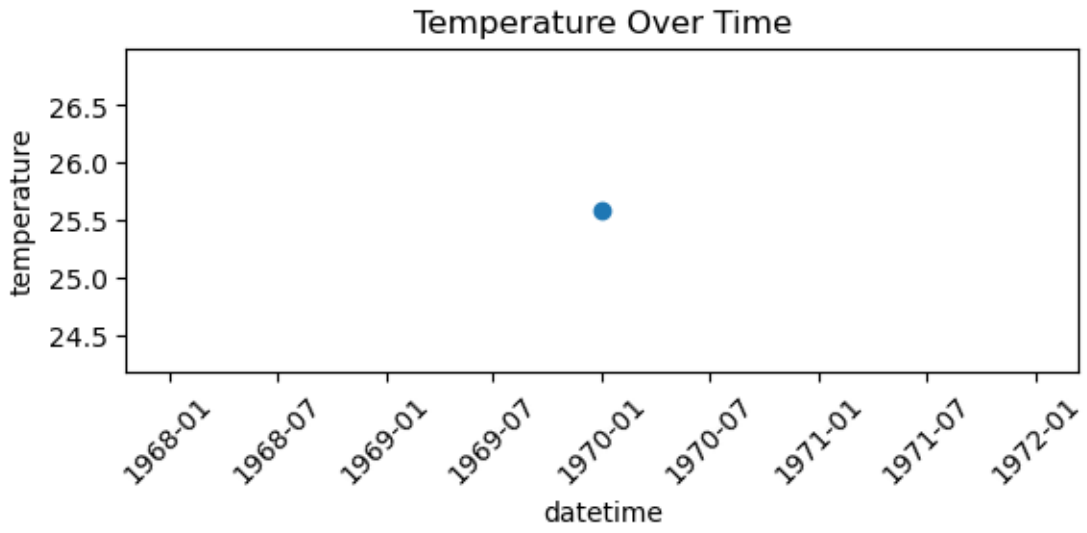
[26]:
```
df.columns
```

[26]:
```
Index(['datetime', 'temperature', 'humidity', 'wind_speed',
       'weather_description'],
      dtype='object')
```

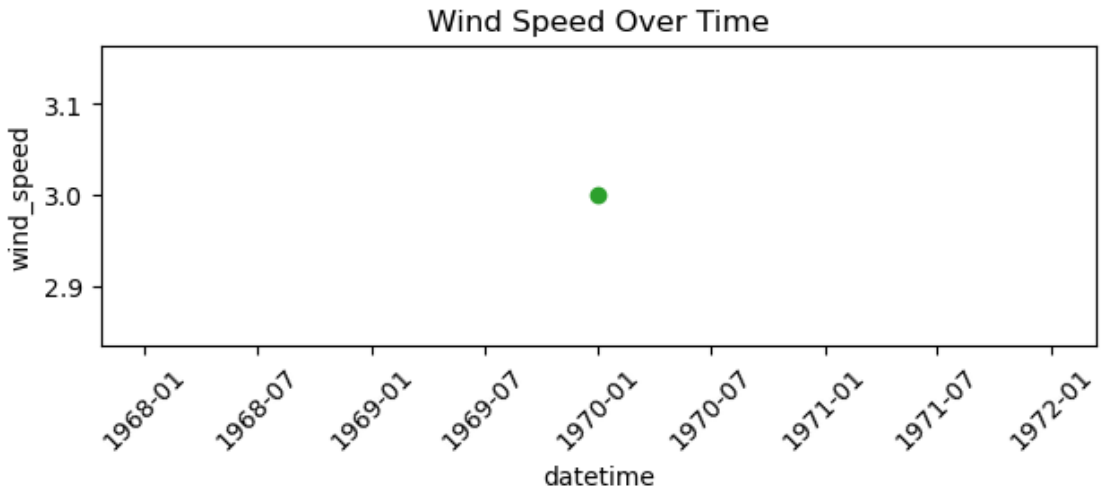
[28]:
```
# Temperature Plot
plt.subplot(2, 1, 1)
plt.plot(df['datetime'], df['temperature'], marker='o', color='tab:blue')
plt.title('Temperature Over Time')
plt.xlabel('datetime')
plt.ylabel('temperature')
plt.xticks(rotation=45)
```

[28]:
```
(array([-731., -549., -365., -184.,    0.,  181.,  365.,  546.,  730.]),
 [Text(-731.0, 0, '1968-01'),
  Text(-549.0, 0, '1968-07'),
  Text(-365.0, 0, '1969-01'),
  Text(-184.0, 0, '1969-07'),
  Text(0.0, 0, '1970-01'),
  Text(181.0, 0, '1970-07'),
```

```
Text(365.0, 0, '1971-01'),
Text(546.0, 0, '1971-07'),
Text(730.0, 0, '1972-01')])
```



[30]:
```
# Wind Speed Scatter Plot
plt.subplot(2, 1, 2)
plt.scatter(df['datetime'], df['wind_speed'], color='tab:green')
plt.title('Wind Speed Over Time')
plt.xlabel('datetime')
plt.ylabel('wind_speed')
plt.xticks(rotation=45)

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



0.1.8 7. Apply data aggregation techniques to summarize weather statistics by specific time

0.1.9 periods (e.g., daily, monthly, seasonal).

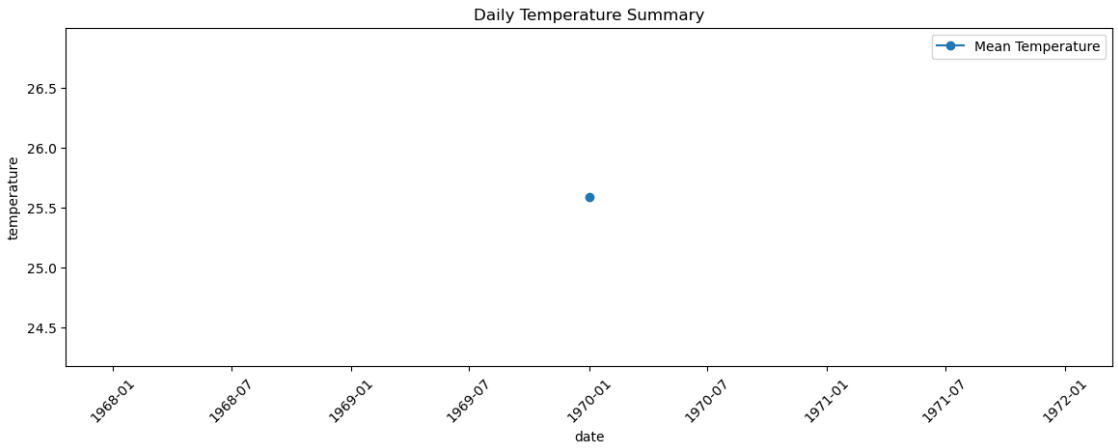
[32]:
```
# Convert 'Date' to datetime format
df["datetime"] = pd.to_datetime(df["datetime"])


# Set 'Date' as the index
df.set_index('datetime', inplace=True)

# Resample and aggregate data
daily_summary = df.resample('D').agg({
    'temperature': ['mean', 'max', 'min'],
    'wind_speed': ['mean', 'max', 'min']
})

monthly_summary = df.resample('ME').agg({
    'temperature': ['mean', 'max', 'min'],
    'wind_speed': ['mean', 'max', 'min']
})

# Print the summary statistics
print("Daily Summary:\n", daily_summary)
print("\nMonthly Summary:\n", monthly_summary)
```

```
Daily Summary:
            temperature                wind_speed
                  mean    max    min       mean max min
datetime
```

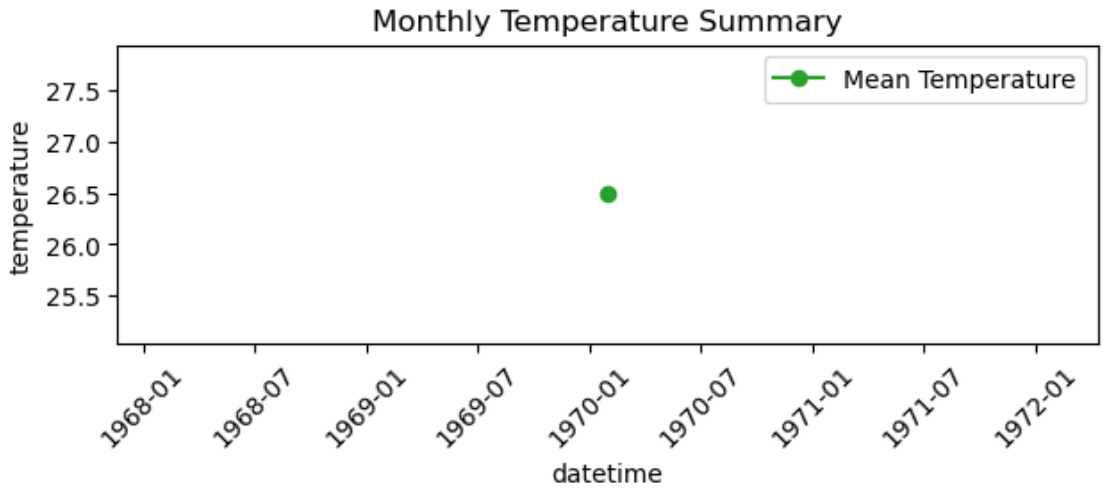
```
1970-01-01       25.59  25.59  25.59        3.0    3    3

Monthly Summary:
            temperature                wind_speed
                  mean    max    min       mean max min
datetime
1970-01-31       25.59  25.59  25.59        3.0    3    3
```

[34]:
```
# Plot the aggregated data
plt.figure(figsize=(14, 10))

# Daily Temperature Summary
plt.subplot(2, 1, 1)
plt.plot(daily_summary.index, daily_summary[('temperature', 'mean')],␣
 ↪marker='o', color='tab:blue', label='Mean Temperature')
plt.fill_between(daily_summary.index, daily_summary[('temperature', 'min')],␣
 ↪daily_summary[('temperature', 'max')], color='tab:blue', alpha=0.2)
plt.title('Daily Temperature Summary')
plt.xlabel('date')
plt.ylabel('temperature')
plt.legend()
plt.xticks(rotation=45)
```

[34]:
```
(array([-731., -549., -365., -184.,    0.,  181.,  365.,  546.,  730.]),
 [Text(-731.0, 0, '1968-01'),
  Text(-549.0, 0, '1968-07'),
  Text(-365.0, 0, '1969-01'),
  Text(-184.0, 0, '1969-07'),
  Text(0.0, 0, '1970-01'),
  Text(181.0, 0, '1970-07'),
  Text(365.0, 0, '1971-01'),
  Text(546.0, 0, '1971-07'),
  Text(730.0, 0, '1972-01')])
```



[83]:
```
# Monthly Temperature Summary
plt.subplot(2, 1, 2)
plt.plot(monthly_summary.index, monthly_summary[('temperature', 'mean')],␣
 ↪marker='o', color='tab:green', label='Mean Temperature')
plt.fill_between(monthly_summary.index, monthly_summary[('temperature',␣
 ↪'min')], monthly_summary[('temperature', 'max')], color='tab:green', alpha=0.
 ↪2)
plt.title('Monthly Temperature Summary')
plt.xlabel('datetime')
plt.ylabel('temperature')
plt.legend()
plt.xticks(rotation=45)

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



0.1.10 8. Incorporate geographical information, if available, to create maps or geospatial

0.1.11 visualizations representing weather patterns across different locations.

[36]:
```
import requests
import pandas as pd
import folium

# Define a list of locations
locations = ['Pune', 'Mumbai', 'Delhi', 'Chennai', 'Kolkata']
```

```
# Fetch weather data for multiple locations
api_key = '18896d417b61718248835471bedf9b54'
weather_data = []

for location in locations:
    url = f'http://api.openweathermap.org/data/2.5/weather?
 ↪q={location}&appid={api_key}'
    response = requests.get(url)
    data = response.json()
    weather_info = {
        'Location': location,
        'Latitude': data['coord']['lat'],
        'Longitude': data['coord']['lon'],
        'Temperature (C)': data['main']['temp'] - 273.15,
        'Wind Speed (m/s)': data['wind']['speed']
    }
    weather_data.append(weather_info)

# Print the raw weather data to debug
print("Weather Data:\n", weather_data)

# Create a DataFrame
df = pd.DataFrame(weather_data)

# Print the DataFrame to debug
print("\nDataFrame:\n", df)

# Create a map centered around India
m = folium.Map(location=[20.5937, 78.9629], zoom_start=5)

# Add markers to the map
for i, row in df.iterrows():
    folium.Marker(
        location=[row['Latitude'], row['Longitude']],
        popup=(
            f"Location: {row['Location']}<br>"
            f"Temperature (C): {row['Temperature (C)']:.2f}<br>"
            f"Wind Speed (m/s): {row['Wind Speed (m/s)']:.2f}"
        ),
        tooltip=row['Location']
    ).add_to(m)

# Save the map to an HTML file
m.save('weather_map.html')

# Display the map
m
```

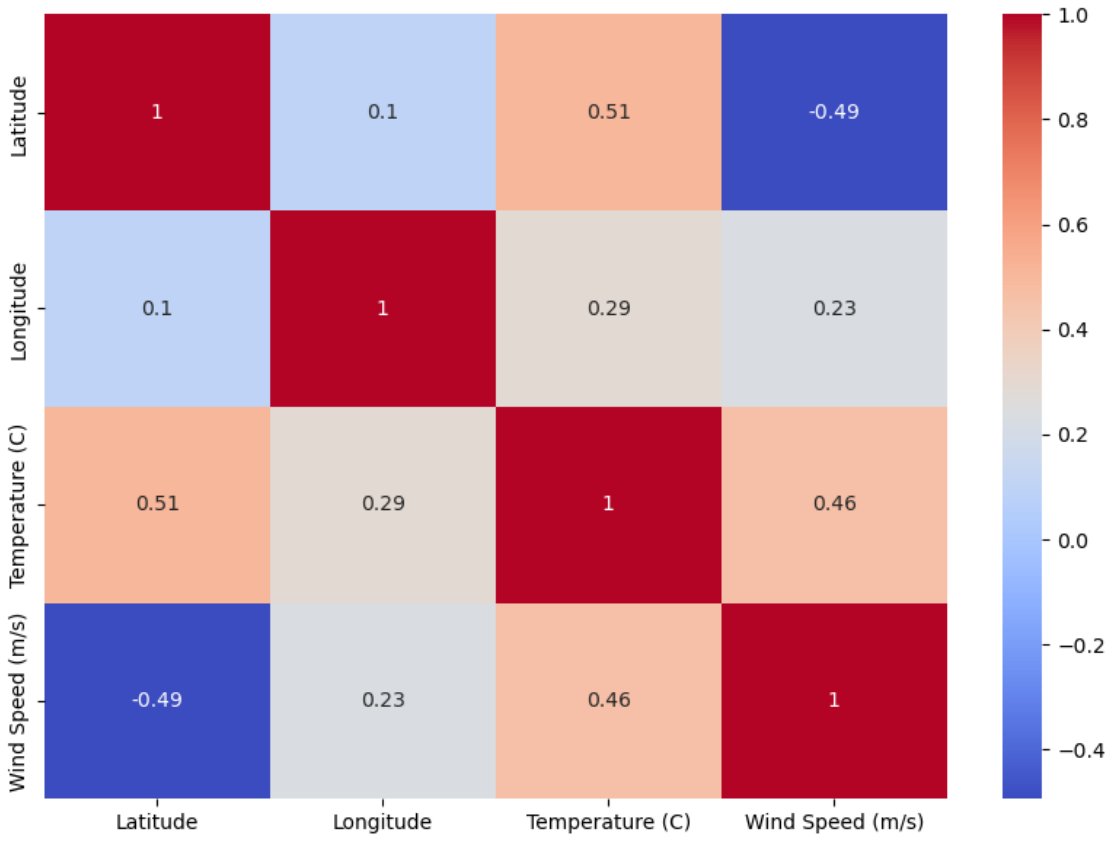
```
Weather Data:
 [{'Location': 'Pune', 'Latitude': 18.5196, 'Longitude': 73.8553, 'Temperature
(C)': 25.600000000000023, 'Wind Speed (m/s)': 3}, {'Location': 'Mumbai',
'Latitude': 19.0144, 'Longitude': 72.8479, 'Temperature (C)':
27.660000000000025, 'Wind Speed (m/s)': 4.37}, {'Location': 'Delhi', 'Latitude':
28.6667, 'Longitude': 77.2167, 'Temperature (C)': 32.93000000000001, 'Wind Speed
(m/s)': 4.06}, {'Location': 'Chennai', 'Latitude': 13.0878, 'Longitude':
80.2785, 'Temperature (C)': 30.170000000000016, 'Wind Speed (m/s)': 5.89},
{'Location': 'Kolkata', 'Latitude': 22.5697, 'Longitude': 88.3697, 'Temperature
(C)': 28.970000000000027, 'Wind Speed (m/s)': 4.03}]

DataFrame:
   Location  Latitude  Longitude  Temperature (C)  Wind Speed (m/s)
0     Pune   18.5196    73.8553            25.60              3.00
1   Mumbai   19.0144    72.8479            27.66              4.37
2    Delhi   28.6667    77.2167            32.93              4.06
3  Chennai   13.0878    80.2785            30.17              5.89
4  Kolkata   22.5697    88.3697            28.97              4.03
```

[36]:
```
<folium.folium.Map at 0x2846c52e6f0>
```

0.1.12 9. Explore and visualize relationships between weather attributes, such as temperature and humidity,

0.1.13 using correlation plots or heatmaps.

[38]:
```
import seaborn as sns
## Correlation Heatmap
numeric_df = df.select_dtypes(include=['number'])

# Compute the correlation matrix
correlation_matrix = numeric_df.corr()

# Plot the heatmap
plt.figure(figsize=(10, 7))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.show()
```


Latitude
Longitude
Temperature (C)
Wind Speed (m/s)
1
0.1
0.51
-0.49
0.1
1
0.29
0.23
0.51
0.29
1
0.46
-0.49
0.23
0.46
1
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

[ ]: