In [1]:
```
import requests
import pandas as pd
import datetime
```

In [3]:
```
# Set your OpenWeatherMap API key
api_key = 'fb365aa6104829b44455572365ff3b4e'
```

In [5]:
```
# Set the location for which you want to retrieve weather data
lat = 18.184135
lon = 74.610764
```

In [7]:
```
# https://openweathermap.org/api/one-call-3
# how        How to use api call
# Construct the API URL
api_url = f"http://api.openweathermap.org/data/2.5/forecast?lat={lat}&lon={lor
```

In [9]:
```
# Send a GET request to the API
response = requests.get(api_url)
weather_data = response.json()
weather_data.keys()
len(weather_data['list'])
weather_data['list'][0]['weather'][0]['description']
```

Out[9]:
```
'overcast clouds'
```

In [11]:
```
# Getting the data from dictionary and taking into one variable
# Extract relevant weather attributes using list comprehension
temperatures = [item['main']['temp'] for item in weather_data['list']]

# It will extract all values (40) and putting into one variable
timestamps = [pd.to_datetime(item['dt'], unit='s') for item in weather_data['l
temperature = [item['main']['temp'] for item in weather_data['list']]
humidity = [item['main']['humidity'] for item in weather_data['list']]
wind_speed = [item['wind']['speed'] for item in weather_data['list']]
weather_description = [item['weather'][0]['description'] for item in weather_d
```

In [13]:
```
# Create a pandas DataFrame with the extracted weather data
weather_df = pd.DataFrame({'Timestamp': timestamps,
                           'Temperature': temperatures,
                           'humidity': humidity,
                           'wind_speed':wind_speed,
                           'weather_description': weather_description})
```

In [15]:
```
# Set the Timestamp column as the DataFrame's index
weather_df.set_index('Timestamp', inplace=True)
max_temp = weather_df['Temperature'].max()
print(f"Maximum Temperature - {max_temp}")
min_temp = weather_df['Temperature'].min()
print(f"Minimum Temperature - {min_temp}")
```

```
Maximum Temperature - 302.12
Minimum Temperature - 294.42
```

In [17]:
```
# Clean and preprocess the data # Handling missing values
weather_df.fillna(0, inplace=True) # Replace missing values with 0 or appropri
```

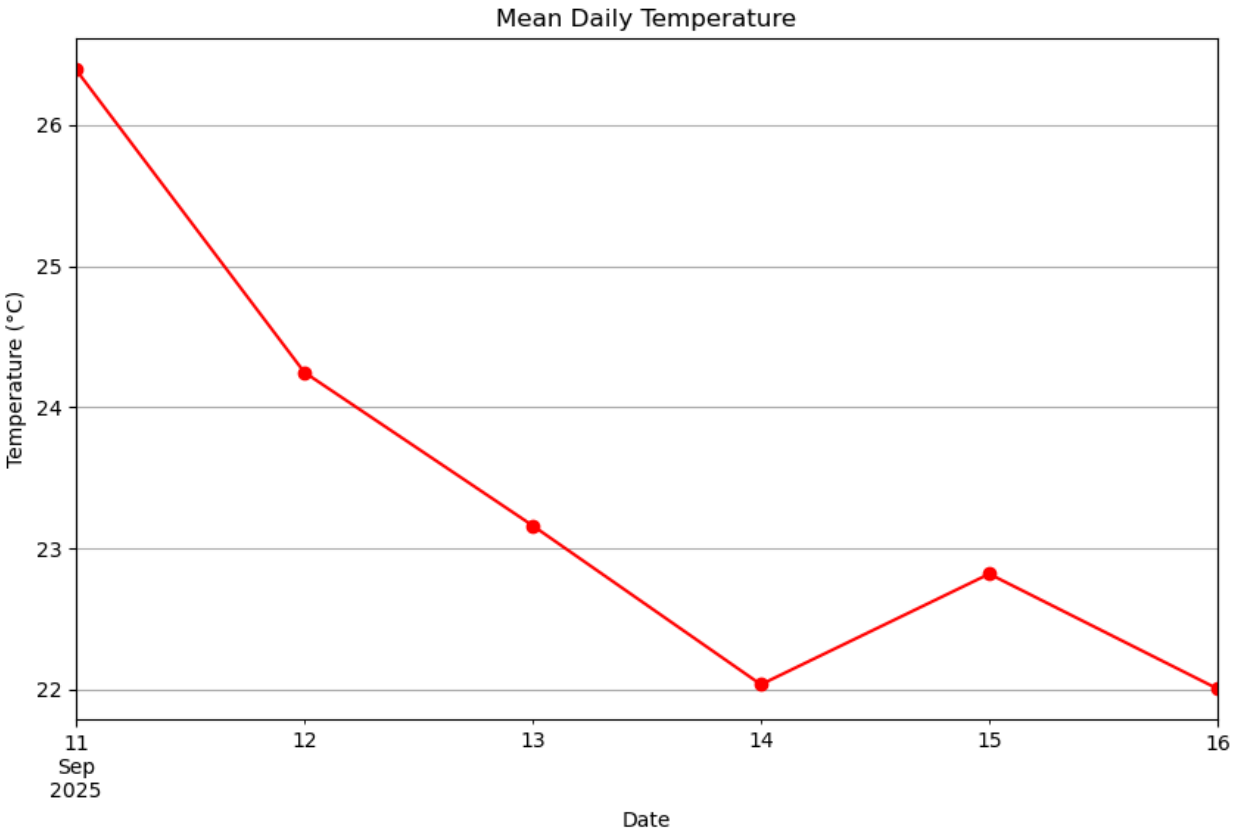
In [19]:
```
# Handling inconsistent format (if applicable)
weather_df['Temperature'] = weather_df['Temperature'].apply(lambda x: x - 273.
```

In [21]:
```
# Convert temperature from Kelvin to Celsius
# Print the cleaned and preprocessed data print(weather_df)
print(weather_df)
```

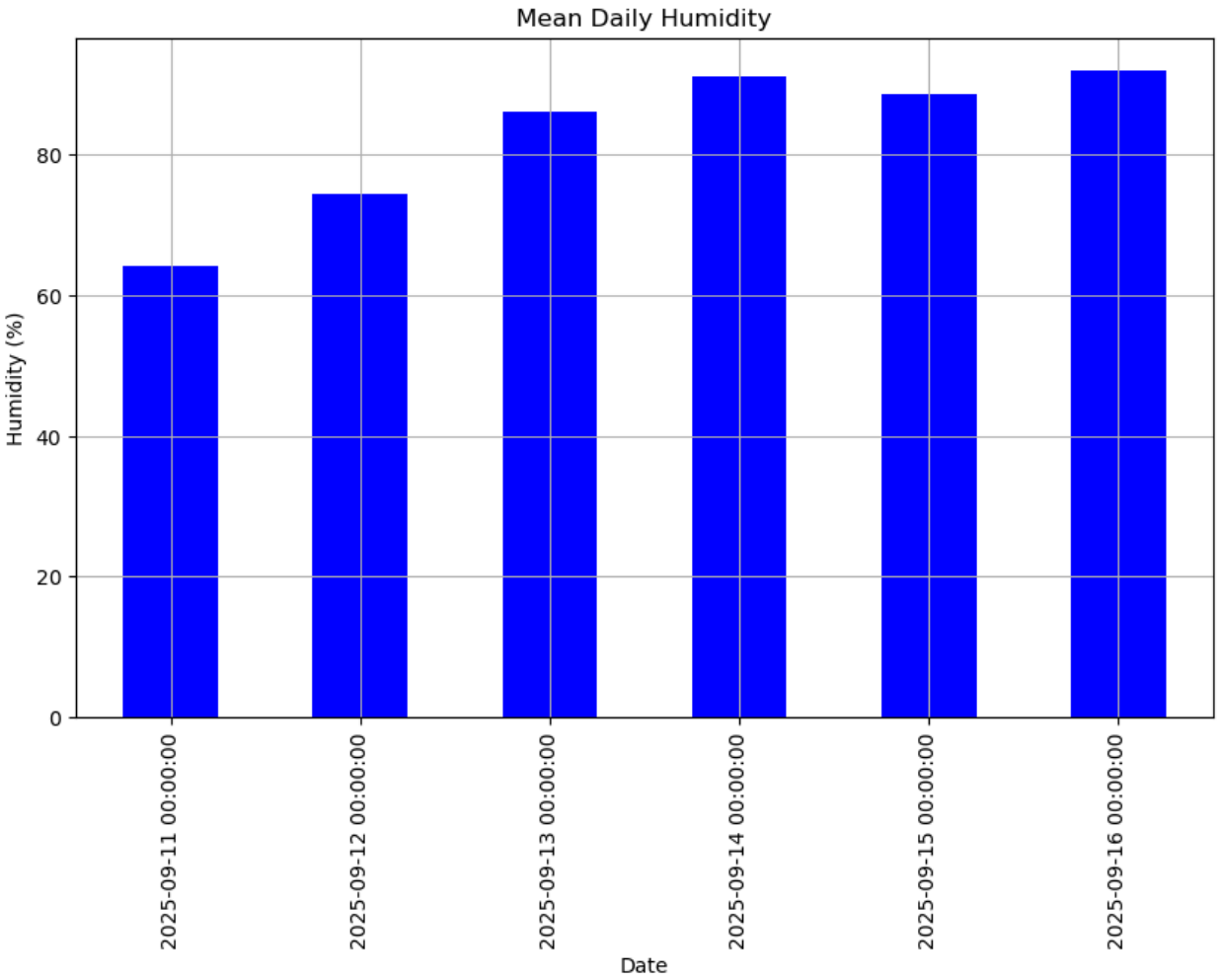
Timestamp	Temperature	humidity	wind_speed	weather_description
2025-09-11 06:00:00	28.51	54	3.59	overcast clouds
2025-09-11 09:00:00	28.97	52	4.23	overcast clouds
2025-09-11 12:00:00	27.75	58	6.40	light rain
2025-09-11 15:00:00	25.38	71	6.31	moderate rain
2025-09-11 18:00:00	24.34	73	5.34	light rain
2025-09-11 21:00:00	23.40	77	3.28	light rain
2025-09-12 00:00:00	23.03	78	2.83	light rain
2025-09-12 03:00:00	23.27	75	3.89	overcast clouds
2025-09-12 06:00:00	24.94	64	4.07	overcast clouds
2025-09-12 09:00:00	28.58	52	4.44	overcast clouds
2025-09-12 12:00:00	23.99	78	5.45	light rain
2025-09-12 15:00:00	24.09	79	3.32	overcast clouds
2025-09-12 18:00:00	23.49	83	4.44	overcast clouds
2025-09-12 21:00:00	22.59	87	3.91	overcast clouds
2025-09-13 00:00:00	22.41	89	3.51	light rain
2025-09-13 03:00:00	22.47	87	4.72	light rain
2025-09-13 06:00:00	23.41	81	5.34	light rain
2025-09-13 09:00:00	25.97	72	5.90	light rain
2025-09-13 12:00:00	24.63	80	4.69	light rain
2025-09-13 15:00:00	22.76	90	4.40	moderate rain
2025-09-13 18:00:00	22.37	92	4.71	moderate rain
2025-09-13 21:00:00	21.27	99	4.88	very heavy rain
2025-09-14 00:00:00	21.47	98	3.62	heavy intensity rain
2025-09-14 03:00:00	21.39	93	3.95	moderate rain
2025-09-14 06:00:00	21.66	92	4.30	light rain
2025-09-14 09:00:00	23.03	87	5.48	light rain
2025-09-14 12:00:00	22.67	88	4.86	overcast clouds
2025-09-14 15:00:00	22.39	89	4.46	overcast clouds
2025-09-14 18:00:00	21.92	91	4.24	overcast clouds
2025-09-14 21:00:00	21.75	92	4.13	overcast clouds
2025-09-15 00:00:00	21.80	93	3.54	overcast clouds
2025-09-15 03:00:00	22.59	90	3.51	overcast clouds
2025-09-15 06:00:00	24.03	82	3.41	overcast clouds
2025-09-15 09:00:00	23.85	84	4.25	light rain
2025-09-15 12:00:00	23.51	86	5.06	light rain
2025-09-15 15:00:00	22.89	88	5.03	overcast clouds
2025-09-15 18:00:00	22.20	92	3.33	overcast clouds
2025-09-15 21:00:00	21.68	94	3.20	light rain
2025-09-16 00:00:00	21.69	94	2.97	light rain
2025-09-16 03:00:00	22.32	90	3.01	light rain

In [23]:
```
import matplotlib.pyplot as plt
daily_mean_temp = weather_df['Temperature'].resample('D').mean()
daily_mean_humidity = weather_df['humidity'].resample('D').mean()
daily_mean_wind_speed = weather_df['wind_speed'].resample('D').mean()
```

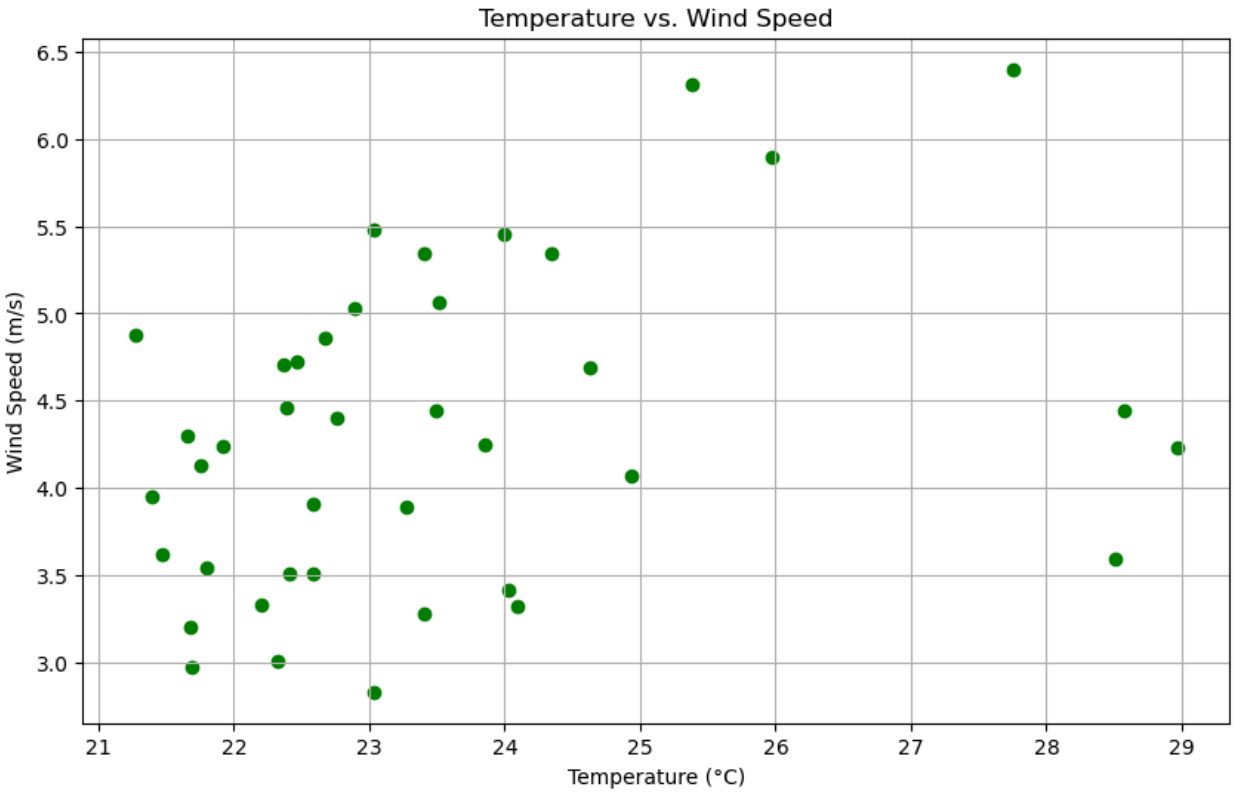
In [27]:
```
# Plot the mean daily temperature over time (Line plot)
plt.figure(figsize=(10, 6))
daily_mean_temp.plot(color='red', linestyle='-', marker='o')
plt.title('Mean Daily Temperature')
plt.xlabel('Date')
plt.ylabel('Temperature (°C)')
plt.grid(True)
plt.show()
```



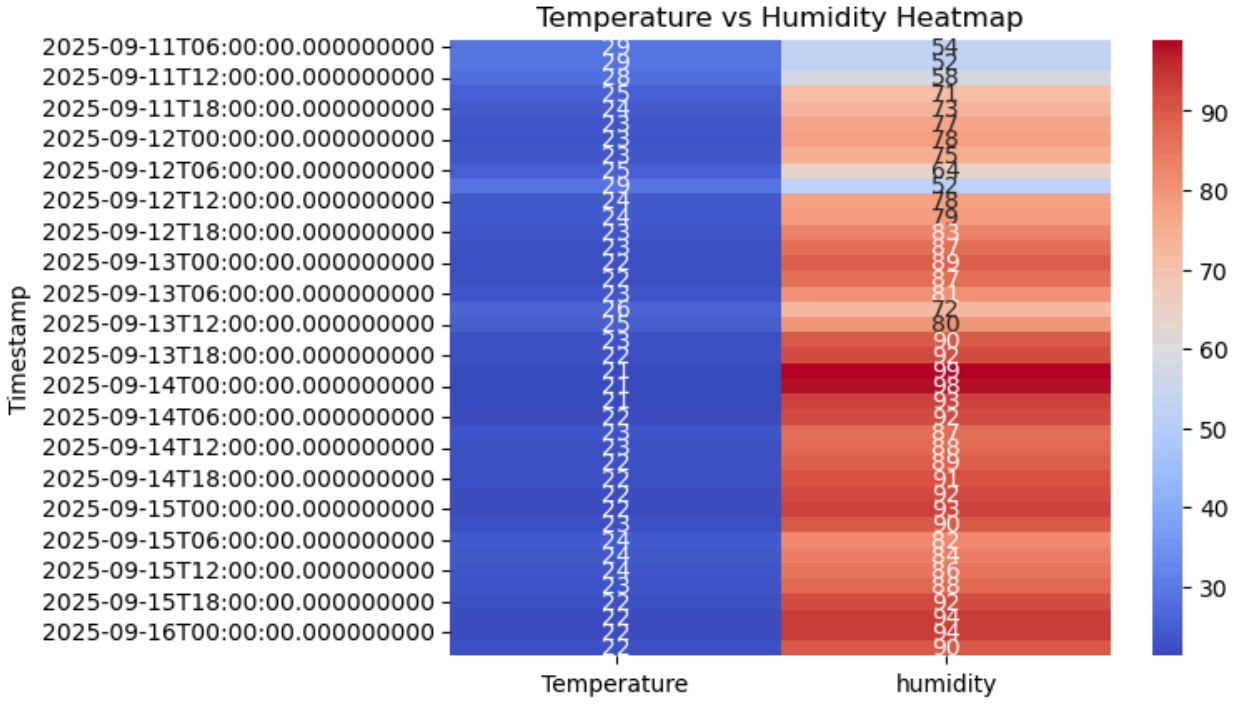
In [29]:
```
# Plot the mean daily humidity over time (Bar plot)
plt.figure(figsize=(10, 6))
daily_mean_humidity.plot(kind='bar', color='blue')
plt.title('Mean Daily Humidity')
plt.xlabel('Date')
plt.ylabel('Humidity (%)')
plt.grid(True)
plt.show()
```



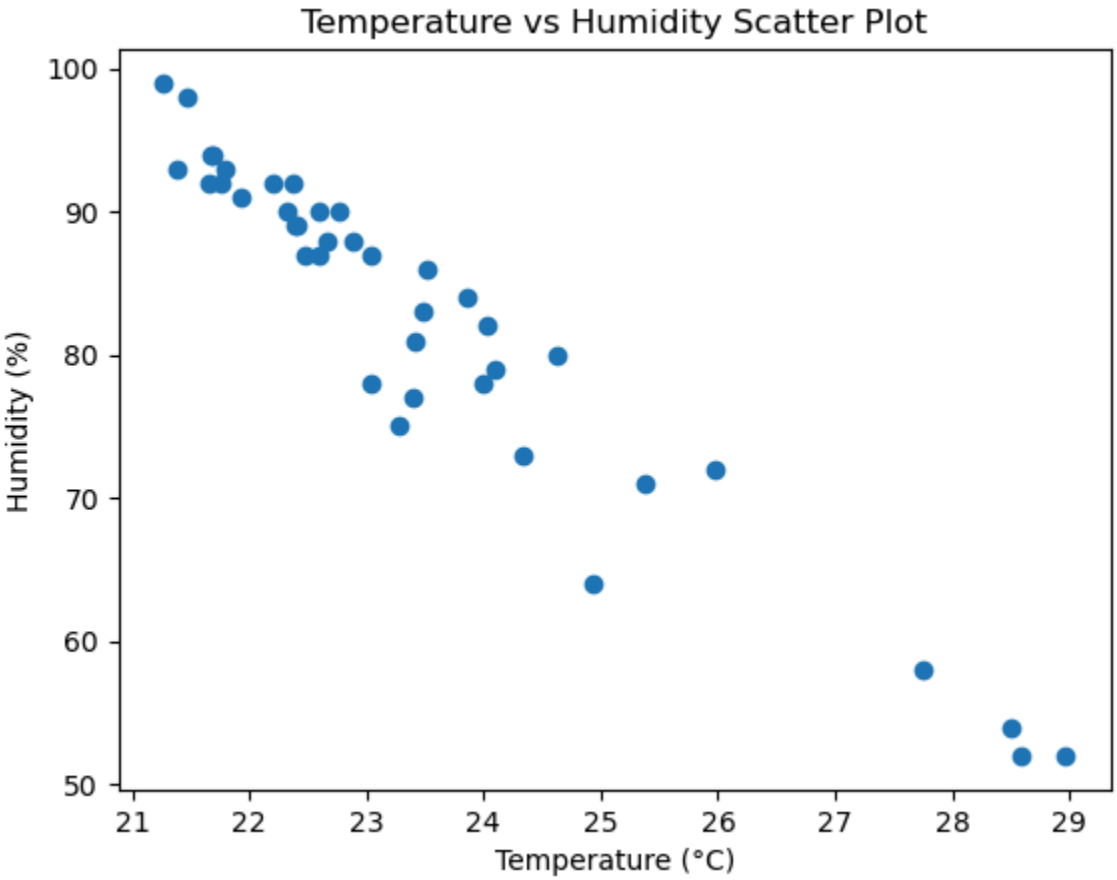
In [31]:
```
# Plot the relationship between temperature and wind speed (Scatter plot)
plt.figure(figsize=(10, 6))
plt.scatter(weather_df['Temperature'], weather_df['wind_speed'], color='green'
plt.title('Temperature vs. Wind Speed')
plt.xlabel('Temperature (°C)')
plt.ylabel('Wind Speed (m/s)')
plt.grid(True)
plt.show()
```



In [33]:
```
# Heatmap
import seaborn as sns
heatmap_data = weather_df[['Temperature', 'humidity']]
sns.heatmap(heatmap_data, annot=True, cmap='coolwarm')
plt.title('Temperature vs Humidity Heatmap')
plt.show()
```



In [35]:
```
# Create a scatter plot to visualize the relationship between temperature and
plt.scatter(weather_df['Temperature'], weather_df['humidity'])
plt.xlabel('Temperature (°C)')
plt.ylabel('Humidity (%)')
plt.title('Temperature vs Humidity Scatter Plot')
plt.show()
```



In [ ]: