Daily Weather Data Analysis with Plotly 🌦️📊

Welcome to Exploratory Data Analysis with Plotly! In this notebook, we will embark on an interactive journey to explore a rich dataset containing daily weather information for capital cities around the world. This dataset is packed with over 40 diverse weather-related features, making it a goldmine for climate trend analysis and global weather pattern identification.

Objective:

Our primary objective is to conduct an engaging and interactive exploratory data analysis (EDA) using the powerful Plotly library while keeping this notebook beginner-friendly. Through interactive visualizations, we aim to unlock insights from the data, understand weather trends, and reveal relationships between different weather parameters.

In this journey of data exploration, we'll harness the capabilities of Plotly to create dynamic and interactive plots. These visualizations will not only make the analysis more informative but also user-friendly for individuals with varying levels of data analysis experience.

So, let's embark on this data-driven adventure and start unraveling the fascinating world of weather data through interactive exploration! 🌟

In [1]:

```
# Import necessary libraries

import pandas as pd
import numpy as np

import plotly.express as px
import plotly.graph_objects as go
```

- Plotly Express is a high-level Python library for creating interactive visualizations.
- The pprint module stands for "pretty-printing." It is used for formatting complex or nested data structure
- IPython environments to control the display of HTML and other content.

Basic Data Exploration

In [2]:

```
weather = pd.read_csv("Daily Weather Analysis .csv")
```

In [3]:

```
# This will display all columns without truncation
pd.set_option('display.max_columns', None)
```

In [4]:

```
weather.sample(4)
```

Out[4]:

	country	location_name	latitude	longitude	timezone	last_updated_epoch	last_updated	temperature_celsius	temperature_f
1609	Ecuador	Quito	-0.22	-78.50	America/Guayaquil	1693955700	2023-09-05 18:15	15.0	
1387	Brazil	Bras	-2.08	-58.17	America/Manaus	1693870200	2023-09-04 19:30	29.5	
1493	Norway	Oslo	59.92	10.75	Europe/Oslo	1693870200	2023-09-05 01:30	14.0	
3011	Kazakhstan	Astana	51.18	71.43	Asia/Almaty	1694558700	2023-09-13 04:45	9.0	

In [5]:

```
numeric = weather.describe(include=np.number)
categoric = weather.describe(include="object")
```

In [6]:

```
#Summary Statistic for Numerical features
numeric
```

Out[6]:

	latitude	longitude	last_updated_epoch	temperature_celsius	temperature_fahrenheit	wind_mph	wind_kph	wind_degree	p
count	3119.000000	3119.000000	3.119000e+03	3119.000000	3119.000000	3119.000000	3119.000000	3119.000000	
mean	19.305752	21.809503	1.693916e+09	22.483071	72.469702	6.513113	10.480507	163.006412	
std	24.580523	65.663249	3.924815e+05	6.518389	11.733357	4.507929	7.256625	103.023288	
min	-41.300000	-175.200000	1.693301e+09	-2.000000	28.400000	2.200000	3.600000	1.000000	
25%	3.750000	-6.840000	1.693568e+09	17.900000	64.200000	2.900000	4.700000	75.000000	
50%	17.250000	23.240000	1.693956e+09	23.000000	73.400000	5.400000	8.600000	157.000000	
75%	41.330000	50.580000	1.694257e+09	27.500000	81.500000	8.900000	14.400000	240.000000	

	latitude	longitude	last_updated_epoch	temperature_celsius	temperature_fahrenheit	wind_mph	wind_kph	wind_degree	p
max	63.830000	179.220000	1.694559e+09	45.000000	113.000000	43.800000	70.600000	360.000000	1

In [7]:

```
#Summary Statistic for Categorical features
categoric
```

Out[7]:

	country	location_name	timezone	last_updated	condition_text	wind_direction	sunrise	sunset	moonrise	moonset	moon_phase
count	3119	3119	3119	3119	3119	3119	3119	3119	3119	3119	3119
unique	185	197	183	517	22	16	159	200	725	947	5
top	Bulgaria	Kabul	Asia/Bangkok	2023-09-13 00:45	Partly cloudy	N	05:57 AM	06:04 PM	No moonrise	05:23 AM	Waning Crescent
freq	48	16	48	43	1247	358	75	52	195	12	1170

Plotly LineChart

In [8]:

```
# Temperature Trends
trend = weather.query("country in ['Thailand', 'New Zealand', 'Spain', 'United States of America']")
```

In [9]:

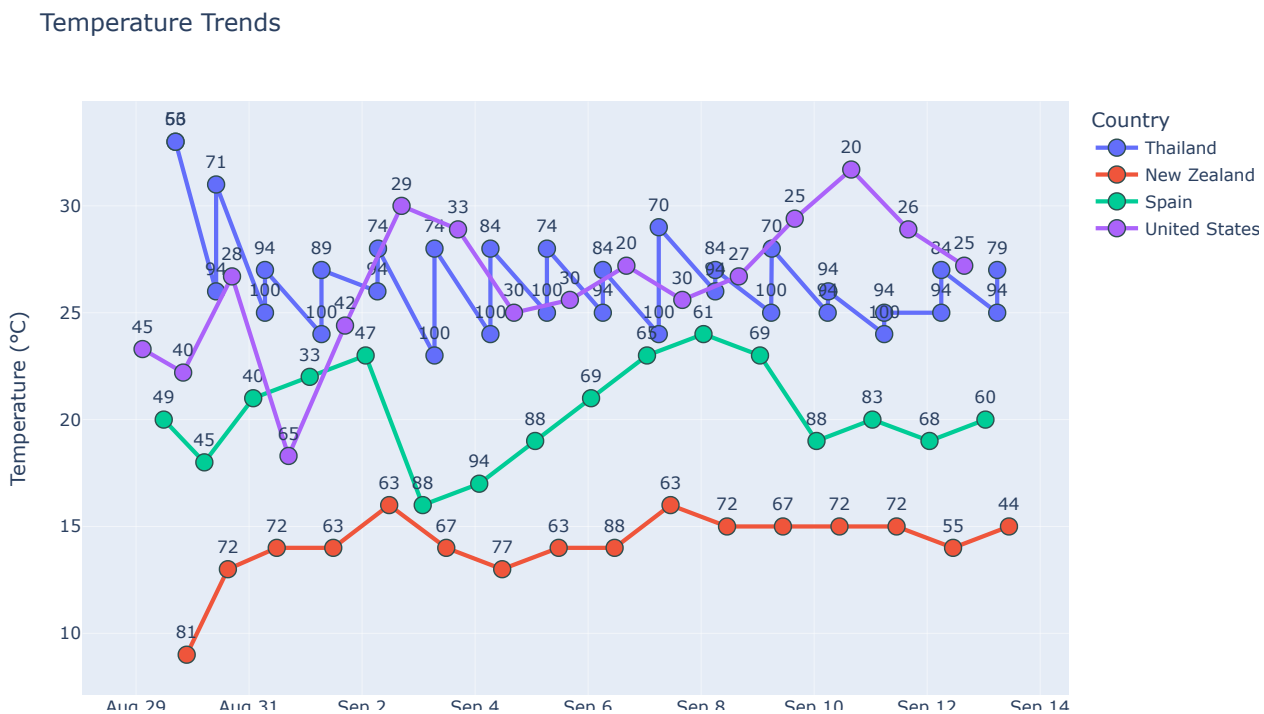
```
# Selected Countires Temperature Trends
fig = px.line(trend,x='last_updated',y='temperature_celsius',title="Temperature Trends",color="country",markers=Tru
fig.update_xaxes(title="Last Updated")
fig.update_yaxes(title="Temperature (°C)")

# Customize the text labels
fig.update_traces(
    #texttemplate='%{y}°C<br>Humidity: %{text}%',  # Customized text label format
    textposition='top center',  # Adjust text label position
    textfont=dict(size=12),  # Adjust text label font size
)

# Customize the legend
fig.update_layout(
    legend_title_text='Country',# Change the legend title
    height=600,
    width=1000
)


# Customize the line and marker styles
fig.update_traces(
    line=dict(width=3),  # Adjust line width
    marker=dict(size=12, line=dict(width=1, color='DarkSlateGrey')),  # Adjust marker size and outline
)


fig.show()
```



Plotly BarChart

In [10]:

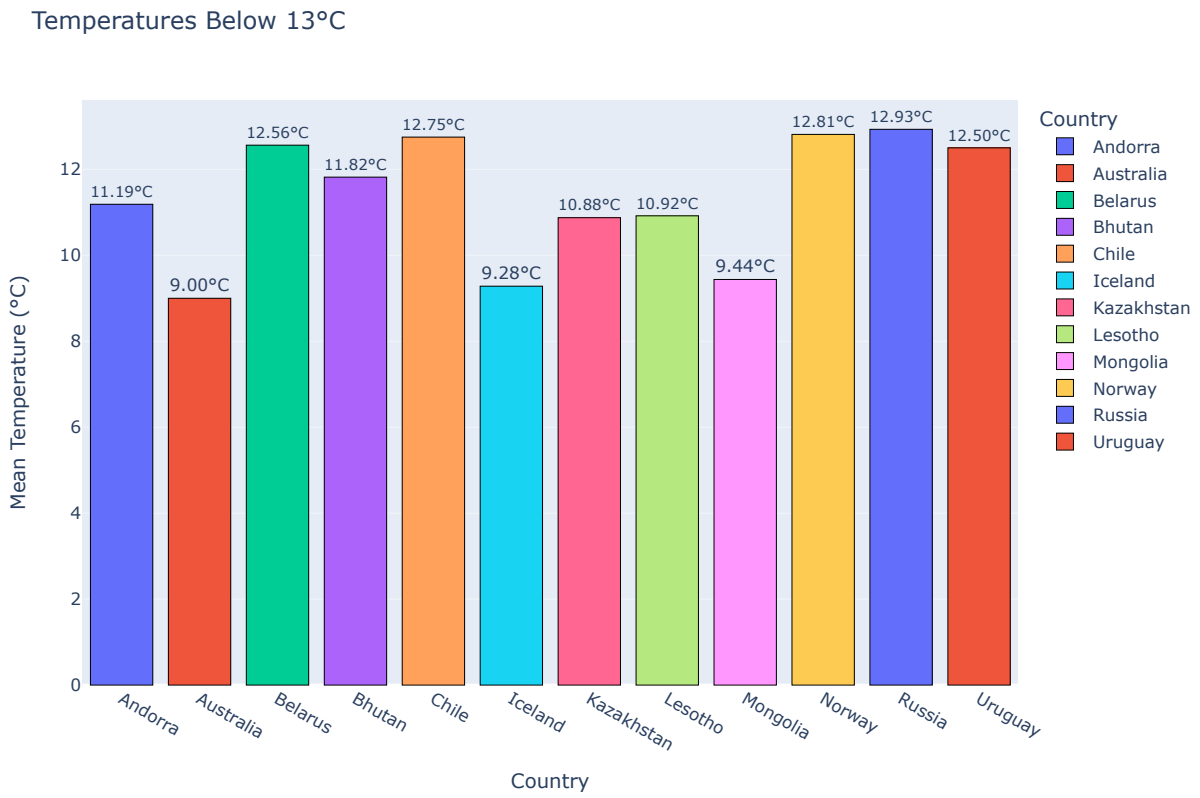
```
# Grouped the countries get the mean temperature
grouped_data = weather.groupby(['country'])['temperature_celsius'].mean().reset_index()
# Get the counties which have low temperature
low_temp = grouped_data.query('temperature_celsius < 13')
```

In [11]:

```
# Counties Temperatures Below 13°C degree
barchart = px.bar(low_temp, x="country", y="temperature_celsius", title="Temperatures Below 13°C", color="country",
barchart.update_xaxes(title="Country")
barchart.update_yaxes(title="Mean Temperature (°C)")

# Add data labels on top of bars
barchart.update_traces(texttemplate="%{y:.2f}°C", textposition="outside")
# Add interactive legends
barchart.update_layout(showlegend=True)
# Customize color palette
barchart.update_traces(marker=dict(line=dict(color='rgb(0,0,0)', width=0.3)))
# Customize hover information
barchart.update_traces(hovertemplate="<b>%{x}</b><br>Mean Temperature: %{y:.2f}°C")
# Customize the legend title
barchart.update_layout(legend_title_text='Country')


barchart.show()
```

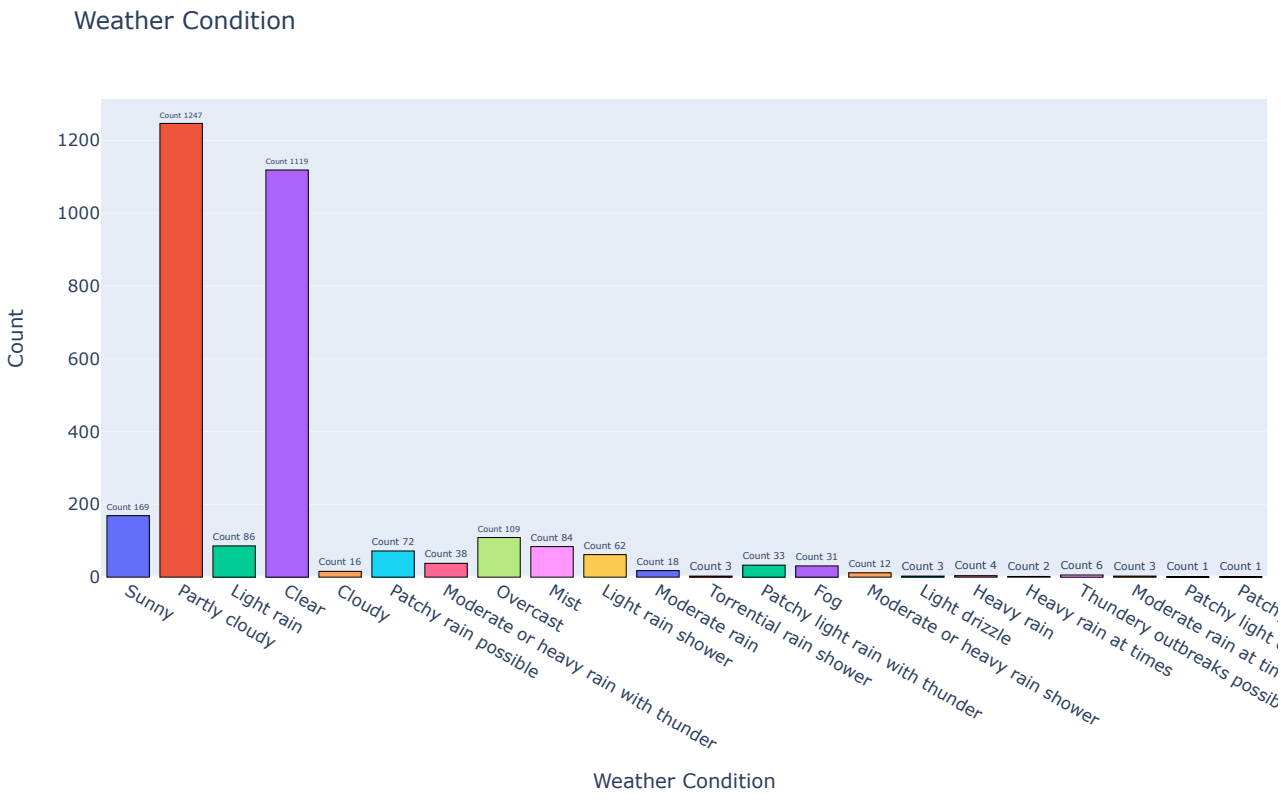


In [12]:

```
# Counties Weather Condition
barchart = px.histogram(weather, x="condition_text", title="Weather Condition", color="condition_text", height=600,
barchart.update_xaxes(title="Weather Condition")
barchart.update_yaxes(title="Count")

# Add data labels on top of bars
barchart.update_traces(texttemplate="Count %{y:2f}", textposition="outside")
# Add interactive legends
barchart.update_layout(showlegend=True)
# Customize color palette
barchart.update_traces(marker=dict(line=dict(color='rgb(0,0,0)', width=0.3)))
# Customize hover information
barchart.update_traces(hovertemplate="<b>%{x}</b><br>Count: %{y:2f}")
# Customize the legend title
barchart.update_layout(legend_title_text="Weather Condition")

barchart.show()
```



Plotly Sunburst Chart

In [13]:

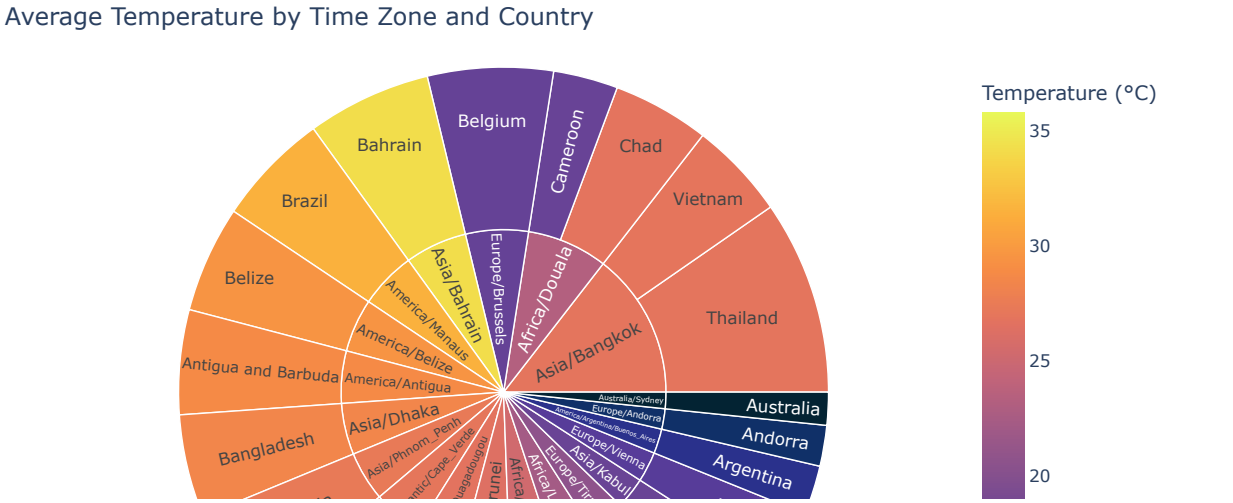
```
zone = ['Asia/Kabul', 'Europe/Tirane', 'Africa/Algiers', 'Europe/Andorra','Africa/Luanda', 'America/Antigua','Ameri
        'Europe/Vienna','Asia/Bahrain', 'Asia/Dhaka','Europe/Brussels', 'America/Belize','America/Manaus', 'Asia/Br
        'Asia/Phnom_Penh', 'Africa/Douala', "Asia/Bangkok"]
```

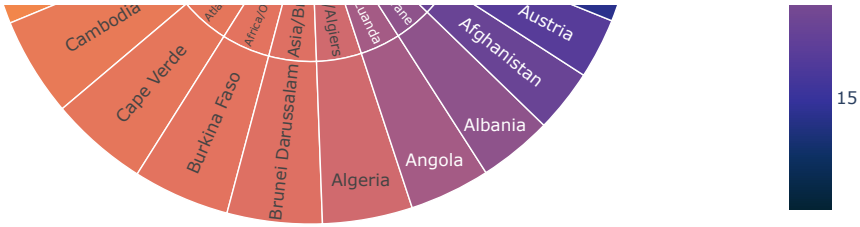
In [14]:

```
weather_zones = weather.query("timezone in @zone")
# Create the Sunburst chart
fig = px.sunburst(
    weather_zones,
    path=['timezone', 'country'], # Define the hierarchical path
    values='temperature_celsius', # Add values
    height=600,
    color='temperature_celsius',  # Color based on temperature
    color_continuous_scale='thermal', # Use the Temps color scale
    color_continuous_midpoint=np.mean(weather_zones['temperature_celsius']), # Set color midpoint to the median tem
    labels={'temperature_celsius': 'Temperature (°C)'},  # Customize the colorbar label
)

# Set title
fig.update_layout(
    title="Average Temperature by Time Zone and Country",
)

fig.show()
```



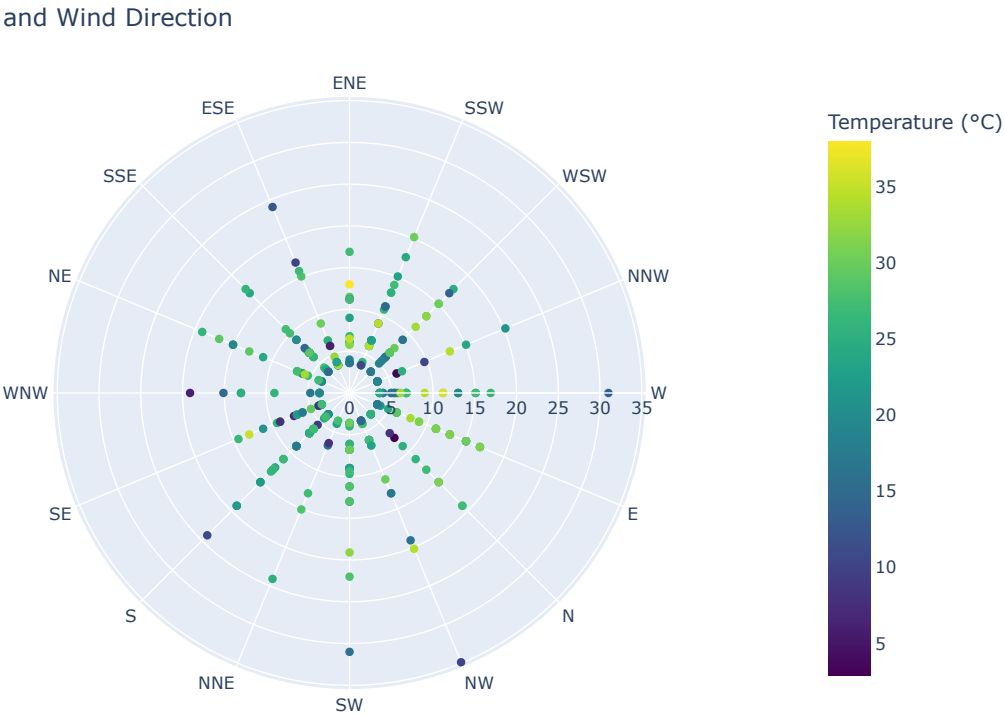


Plotly Polar Chart

In [15]:

```
fig = px.scatter_polar(
    weather_zones,
    r="wind_kph", # Radial distance represents wind speed in kilometers per hour
    theta="wind_direction", # Angular position represents wind direction
    color="temperature_celsius", # Color represents temperature in degrees Celsius
    color_continuous_scale = "viridis", # Use viridis color scale
    title="Wind Speed and Wind Direction",
    labels={'temperature_celsius': 'Temperature (°C)'}, # Customize the colorbar label
    height=600, # Add height to the polar chart
    hover_name="location_name", # Add location names as hover text
)

fig.show()
```



Plotly Scatter Plot

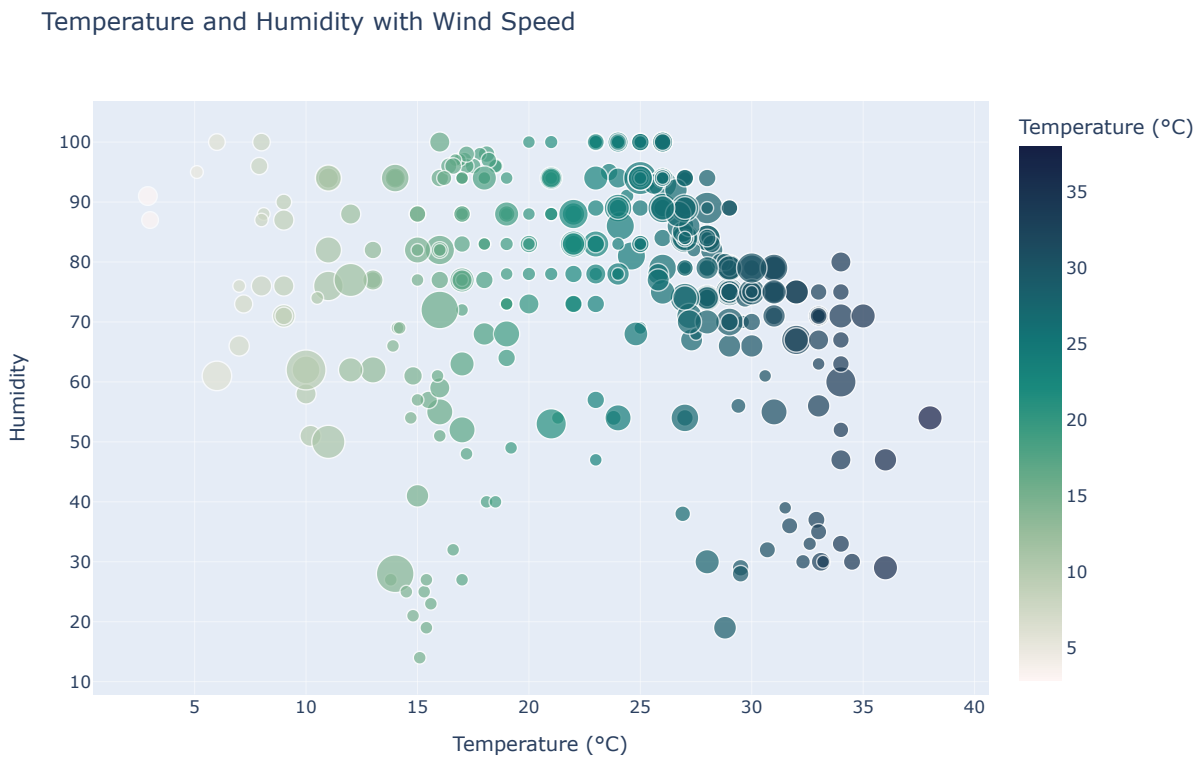
In [16]:

```
fig = px.scatter(
    weather_zones,
    x="temperature_celsius",
    y="humidity",
    color="temperature_celsius",
    color_continuous_scale = "tempo",
    size='wind_kph', # Size of data points based on wind speed in kph
    hover_name='country', # Add hover value
    hover_data=['location_name'],  # Additional data to show on hover (location_name)
    labels=
        {
            'temperature_celsius': 'Temperature (°C)',
            'humidity': 'Humidity',
            'wind_kph': 'Wind speed (kph)',
```

```
            'country': 'Country',
            'condition_text': 'Weather Condition',
            'location_name': 'City'
        },
    title="Temperature and Humidity with Wind Speed",
    height=600,
    )

fig.update_xaxes(title="Temperature (°C)")
fig.update_yaxes(title="Humidity")


fig.show()
```



Plotly Pie Chart

In [17]:

```
# Group the weather condition and calculate the mean of temperature
weather_status = weather_zones.groupby('condition_text')['temperature_celsius'].agg(['mean']).reset_index()
# Rename the columns
weather_status = weather_status.rename(columns={'mean': 'temperature'})
weather_status
```

Out[17]:

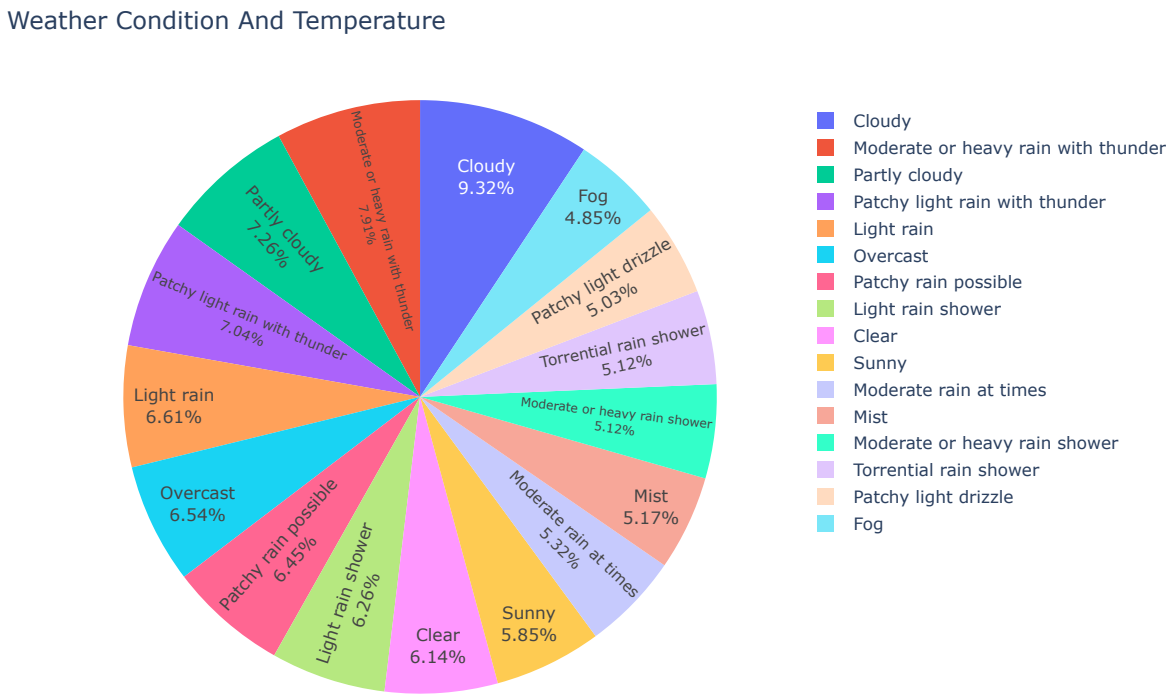
	condition_text	temperature
0	Clear	21.363077
1	Cloudy	32.400000
2	Fog	16.850000
3	Light rain	23.000000
4	Light rain shower	21.785714
5	Mist	17.981818
6	Moderate or heavy rain shower	17.800000
7	Moderate or heavy rain with thunder	27.500000
8	Moderate rain at times	18.500000
9	Overcast	22.755556
10	Partly cloudy	25.237086
11	Patchy light drizzle	17.500000
12	Patchy light rain with thunder	24.500000
13	Patchy rain possible	22.446154
14	Sunny	20.353571
15	Torrential rain shower	17.800000

In [18]:

```
fig = px.pie(
    weather_status,
    names='condition_text',
    values='temperature',
    title='Weather Condition And Temperature',
    height=600,
    labels={
        {
            'condition_text': 'Weather Condition',
            'temperature': 'Temperature (°C)'
        }
    }
)

# Add labels inside the pie chart section
fig.update_traces(textposition='inside', textinfo='label+percent')

fig.show()
```



In [19]:

```
# Group the moon phases and calculate the mean of moon illumination
weather_zones_moon = weather_zones.groupby('moon_phase')['moon_illumination'].agg(['mean']).reset_index()
```

```
# Rename the columns
weather_zones_moon = weather_zones_moon.rename(columns={'mean': 'moon_illumination'})
weather_zones_moon
```

Out[19]:

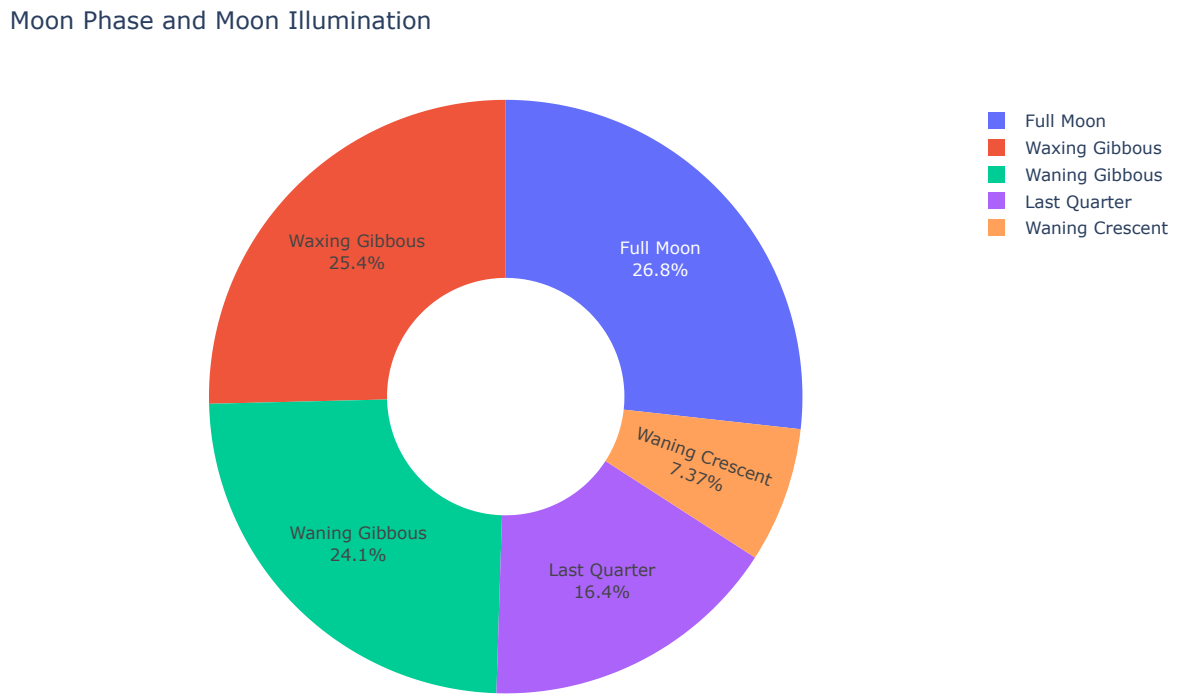
	moon_phase	moon_illumination
0	Full Moon	98.000000
1	Last Quarter	60.000000
2	Waning Crescent	27.000000
3	Waning Gibbous	88.333333
4	Waxing Gibbous	93.000000

In [20]:

```
fig = px.pie(
    weather_zones_moon,
    names='moon_phase',
    values='moon_illumination',
    title='Moon Phase and Moon Illumination',
    height=600,
    hole=0.4,
    labels=(
        {
            'moon_phase': 'Moon Phase',
            'moon_illumination': 'Moon Illumination (%)'
        }
    ),
)

# Add labels inside the pie chart section
fig.update_traces(textposition='inside', textinfo='label+percent')

fig.show()
```



Plotly Gauge Chart

In [21]:

```
temperature_max = weather['temperature_celsius'].max()
temperature_min = weather['temperature_celsius'].min()
temperature_mean = weather['temperature_celsius'].mean()
```

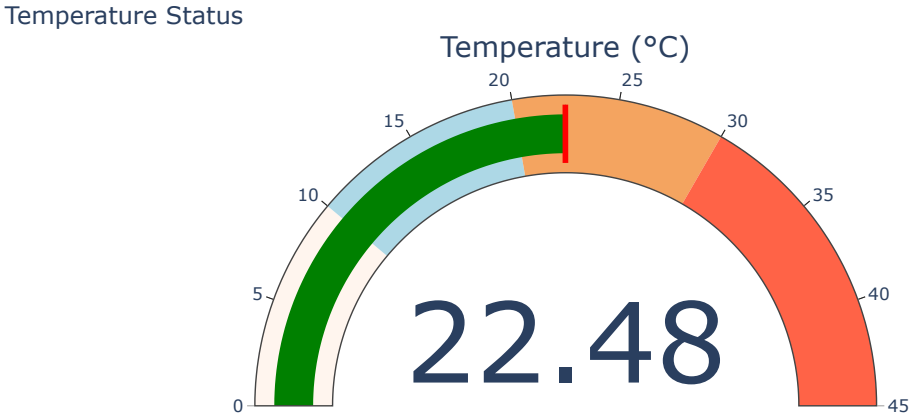
In [22]:

```
fig = go.Figure(go.Indicator(
    mode="gauge+number", # Set the chart mode to gauge with a number
    value= temperature_mean, # Set the value
    title={'text' : 'Temperature (°C)'},  # Set the title of the gauge chart
    gauge= { 'axis' : {'range' : [None, temperature_max]}, # Define the gauge axis range
             'steps' : [
                 {'range' : [0, 10], 'color' : "seashell"}, # Define seashell color ranges for the gauge
```

```
                {'range' : [10, 20], 'color' : "lightblue"}, # Define lightblue color ranges for the gauge
                {'range' : [20, 30], 'color' : "sandybrown"}, # Define sandybrown color ranges for the gauge
                {'range' : [30, temperature_max], 'color' : "tomato"} # Define tomato color ranges for the gauge
            ],
            'threshold' : {
                'line' : {'color' : 'red', 'width' : 4}, # Define the threshold line's appearance
                'thickness' : 0.75, # Set the thickness of the threshold line
                'value' : temperature_mean # Set the value where the threshold line is located
            }
    }
))

fig.update_layout(
    title = "Temperature Status", # Define the title
    height=400 # Define the height
)

fig.show()
```



Plotly MapBox Chart

In [23]:

```
fig = px.scatter_mapbox(
    weather,
    lat="latitude",  # Latitude data column
    lon="longitude", # Longitude data column
    color="temperature_celsius", # Color data for points
    color_continuous_scale=px.colors.cyclical.IceFire, # Color scale
    hover_name= 'location_name', # Add hover value
    size="humidity", # Based on size data will change
    size_max=7,  # Maximum size for points
    labels=(
        {
            'latitude': 'Latitude',
            'longitude': 'Longitude',
            'temperature_celsius' : 'Temperature (°C)',
            'humidity': 'Humidity'
        }),
    height=600,
    width=1000
    )

fig.update_layout(
    mapbox_style='open-street-map', # Map style
    title="Temperature and Humidity ",  # Title of the map
    hovermode='closest',  # Hover mode for interactivity
    mapbox=dict(
        bearing=0, # Bearing of the map
        center=go.layout.mapbox.Center(
            lat=47, # Center latitude
            lon=12 # Center longitude
        ),
        pitch=0, # Map pitch
        zoom=4 # Initial map zoom level
    )
)

fig.show()
```

In [ ]: