In [1]:
```python
import pandas as pd
```

In [2]:
```python
df=pd.read_csv("E://my_project/Weather_live_data_using_pandas/GlobalWeatherReposito
```

In [3]:
```python
df.head()
```

Out[3]:

	country	location_name	latitude	longitude	timezone	last_updated_epoch	las
0	Afghanistan	Kabul	34.52	69.18	Asia/Kabul	1715849100	2
1	Albania	Tirana	41.33	19.82	Europe/Tirane	1715849100	2
2	Algeria	Algiers	36.76	3.05	Africa/Algiers	1715849100	2
3	Andorra	Andorra La Vella	42.50	1.52	Europe/Andorra	1715849100	2
4	Angola	Luanda	-8.84	13.23	Africa/Luanda	1715849100	2

5 rows × 41 columns

In [4]:
```python
df.columns
```

Out[4]:
```
Index(['country', 'location_name', 'latitude', 'longitude', 'timezone',
       'last_updated_epoch', 'last_updated', 'temperature_celsius',
       'temperature_fahrenheit', 'condition_text', 'wind_mph', 'wind_kph',
       'wind_degree', 'wind_direction', 'pressure_mb', 'pressure_in',
       'precip_mm', 'precip_in', 'humidity', 'cloud', 'feels_like_celsius',
       'feels_like_fahrenheit', 'visibility_km', 'visibility_miles',
       'uv_index', 'gust_mph', 'gust_kph', 'air_quality_Carbon_Monoxide',
       'air_quality_Ozone', 'air_quality_Nitrogen_dioxide',
       'air_quality_Sulphur_dioxide', 'air_quality_PM2.5', 'air_quality_PM10',
       'air_quality_us-epa-index', 'air_quality_gb-defra-index', 'sunrise',
       'sunset', 'moonrise', 'moonset', 'moon_phase', 'moon_illumination'],
      dtype='object')
```

In [5]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6620 entries, 0 to 6619
Data columns (total 41 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   country                       6620 non-null   object 
 1   location_name                 6620 non-null   object 
 2   latitude                      6620 non-null   float64
 3   longitude                     6620 non-null   float64
 4   timezone                      6620 non-null   object 
 5   last_updated_epoch            6620 non-null   int64  
 6   last_updated                  6620 non-null   object 
 7   temperature_celsius           6620 non-null   float64
 8   temperature_fahrenheit        6620 non-null   float64
 9   condition_text                6620 non-null   object 
 10  wind_mph                      6620 non-null   float64
 11  wind_kph                      6620 non-null   float64
 12  wind_degree                   6620 non-null   int64  
 13  wind_direction                6620 non-null   object 
 14  pressure_mb                   6620 non-null   float64
 15  pressure_in                   6620 non-null   float64
 16  precip_mm                     6620 non-null   float64
 17  precip_in                     6620 non-null   float64
 18  humidity                      6620 non-null   int64  
 19  cloud                         6620 non-null   int64  
 20  feels_like_celsius            6620 non-null   float64
 21  feels_like_fahrenheit         6620 non-null   float64
 22  visibility_km                 6620 non-null   float64
 23  visibility_miles              6620 non-null   float64
 24  uv_index                      6620 non-null   float64
 25  gust_mph                      6620 non-null   float64
 26  gust_kph                      6620 non-null   float64
 27  air_quality_Carbon_Monoxide   6620 non-null   float64
 28  air_quality_Ozone             6620 non-null   float64
 29  air_quality_Nitrogen_dioxide  6620 non-null   float64
 30  air_quality_Sulphur_dioxide   6620 non-null   float64
 31  air_quality_PM2.5             6620 non-null   float64
 32  air_quality_PM10              6620 non-null   float64
 33  air_quality_us-epa-index      6620 non-null   int64  
 34  air_quality_gb-defra-index    6620 non-null   int64  
 35  sunrise                       6620 non-null   object 
 36  sunset                        6620 non-null   object 
 37  moonrise                      6620 non-null   object 
 38  moonset                       6620 non-null   object 
 39  moon_phase                    6620 non-null   object 
 40  moon_illumination             6620 non-null   int64  
dtypes: float64(23), int64(7), object(11)
memory usage: 2.1+ MB
```

In [6]:
```
df.shape
```

Out[6]:
```
(6620, 41)
```

In [7]:
```
df.describe()
```

Out[7]:

	latitude	longitude	last_updated_epoch	temperature_celsius	temperature_fahı
count	6620.000000	6620.000000	6.620000e+03	6620.000000	6620.
mean	19.209139	21.620124	1.717210e+09	25.655317	78.
std	24.508364	65.625498	8.433390e+05	7.167907	12.
min	-41.300000	-175.200000	1.715849e+09	-1.900000	28.
25%	3.480000	-6.840000	1.716473e+09	21.200000	70.
50%	17.250000	23.240000	1.717208e+09	26.100000	79.
75%	41.320000	49.880000	1.717942e+09	30.000000	86.
max	63.830000	179.220000	1.718634e+09	46.700000	116.

8 rows × 30 columns

In [8]:
```
df.isnull().sum()
```

```
Out[8]: country                         0
        location_name                   0
        latitude                        0
        longitude                       0
        timezone                        0
        last_updated_epoch              0
        last_updated                    0
        temperature_celsius             0
        temperature_fahrenheit          0
        condition_text                  0
        wind_mph                        0
        wind_kph                        0
        wind_degree                     0
        wind_direction                  0
        pressure_mb                     0
        pressure_in                     0
        precip_mm                       0
        precip_in                       0
        humidity                        0
        cloud                           0
        feels_like_celsius              0
        feels_like_fahrenheit           0
        visibility_km                   0
        visibility_miles                0
        uv_index                        0
        gust_mph                        0
        gust_kph                        0
        air_quality_Carbon_Monoxide     0
        air_quality_Ozone               0
        air_quality_Nitrogen_dioxide    0
        air_quality_Sulphur_dioxide     0
        air_quality_PM2.5               0
        air_quality_PM10                0
        air_quality_us-epa-index        0
        air_quality_gb-defra-index      0
        sunrise                         0
        sunset                          0
        moonrise                        0
        moonset                         0
        moon_phase                      0
        moon_illumination               0
        dtype: int64
```

Q-Which country has the highest temperature in Fahrenheit and what is the time there?

```
In [9]: df=df.sort_values(by="temperature_celsius" ,ascending=False)
        df.head()
```

Out[9]:

	country	location_name	latitude	longitude	timezone	last_updated_epoch	last_
6506	Iraq	Baghdad	33.34	44.39	Asia/Baghdad	1718632800	202
4568	Kuwait	Kuwait City	29.37	47.96	Asia/Kuwait	1717768800	202
6313	Iraq	Baghdad	33.34	44.39	Asia/Baghdad	1718545500	202
5594	Qatar	Doha	25.29	51.53	Asia/Qatar	1718201700	202
6516	Kuwait	Kuwait City	29.37	47.96	Asia/Kuwait	1718632800	202

5 rows × 41 columns

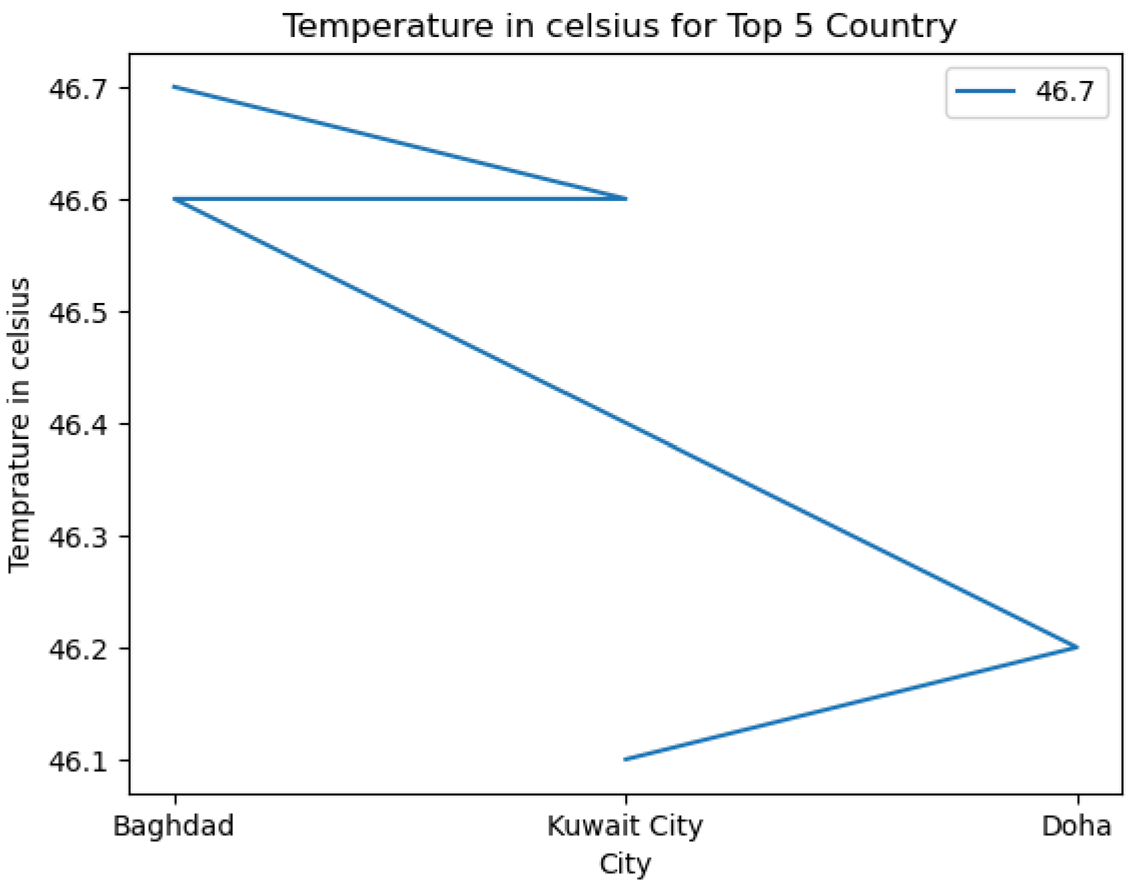
In [10]:
```
highest_temp=df.iloc[0]
location_name=highest_temp['location_name']
country=highest_temp['country']
time=highest_temp['timezone']
temp=highest_temp['temperature_celsius']
date=highest_temp['last_updated']

print(f"The country with the highest temperature in celsius is {country}, {location
```

```
The country with the highest temperature in celsius is Iraq, Baghdad, Asia/Baghdad
with a temperature of 46.7'C and dated: 2024-06-17 17:00
```

In [11]:
```
location=df['location_name'].head(5)
temp=df['temperature_celsius'].head(5)
```

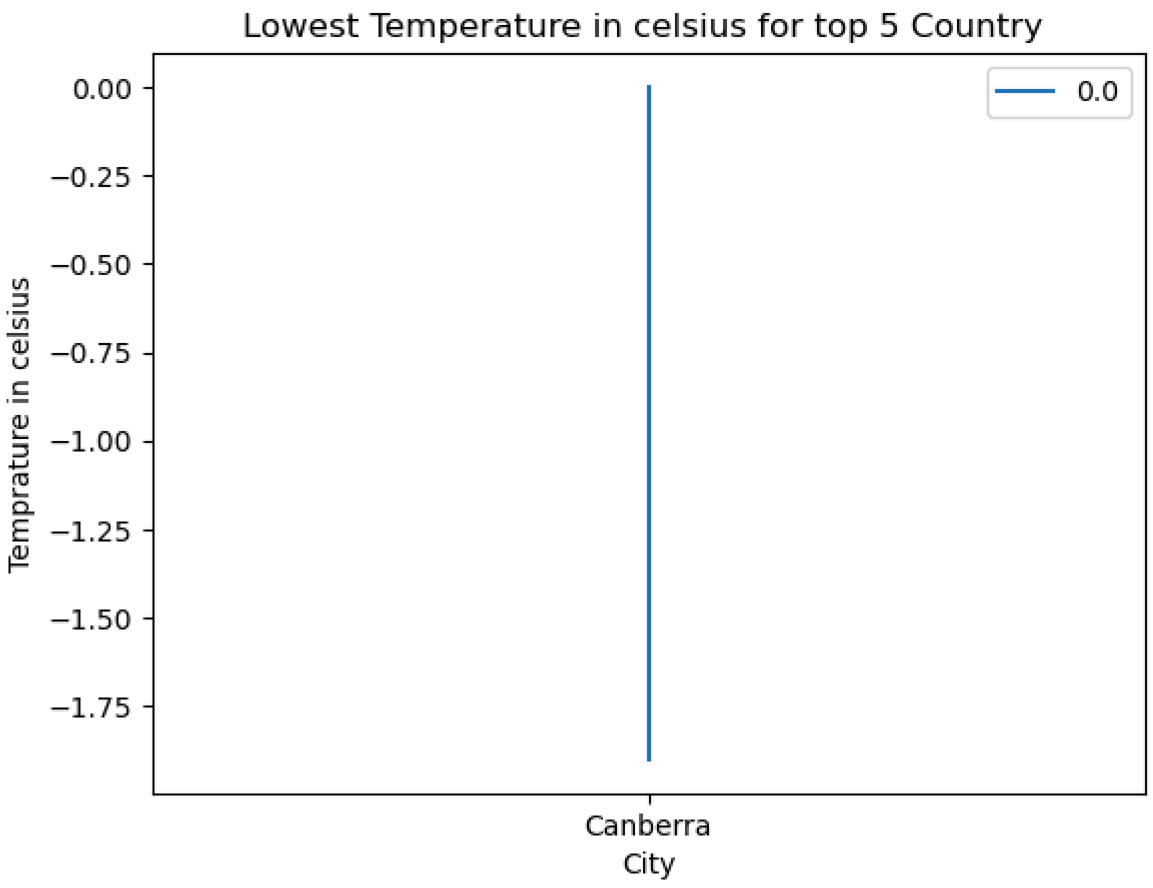
In [12]:
```
import matplotlib.pyplot as plt
import seaborn as sn
```

In [13]:
```
plt.plot(location,temp)
plt.title('Temperature in celsius for Top 5 Country ')
plt.xlabel('City')
plt.ylabel("Temprature in celsius")
plt.legend(temp)
plt.show()
```



```
In [14]: location=df['location_name'].tail(5)
         temp=df['temperature_celsius'].tail(5)
```

```
In [15]: plt.plot(location,temp)
         plt.title('Lowest Temperature in celsius for top 5 Country ')
         plt.xlabel('City')
         plt.ylabel("Temprature in celsius")
         plt.legend(temp)
         plt.show()
```



Q- Which country has the highest difference between temperature and feels like temperature in Celsius and what is the time there?

```
In [16]: df['temp_diff']=df['temperature_celsius']-df['feels_like_celsius']
```

```
In [17]: high_temp=df.loc[df['temp_diff'].idxmax()]

         location=high_temp['location_name']
         country=high_temp['country']
         time=high_temp['timezone']
         last_update=high_temp['last_updated']

         print(f"""The Country with highest difference between temperature and feels like te
         {location},{country},{time} with a difference of {high_temp['temp_diff']}'C. The  t
```

```
The Country with highest difference between temperature and feels like temperature i
n Celsius is
Grindavik,Iceland,Atlantic/Reykjavik with a difference of 6.3'C. The  time there was
2024-05-19 14:15
```

What is the average 'temperature_celsius' across all locations?

```
In [18]: avg=df['temperature_celsius'].mean()
         print(avg)
```

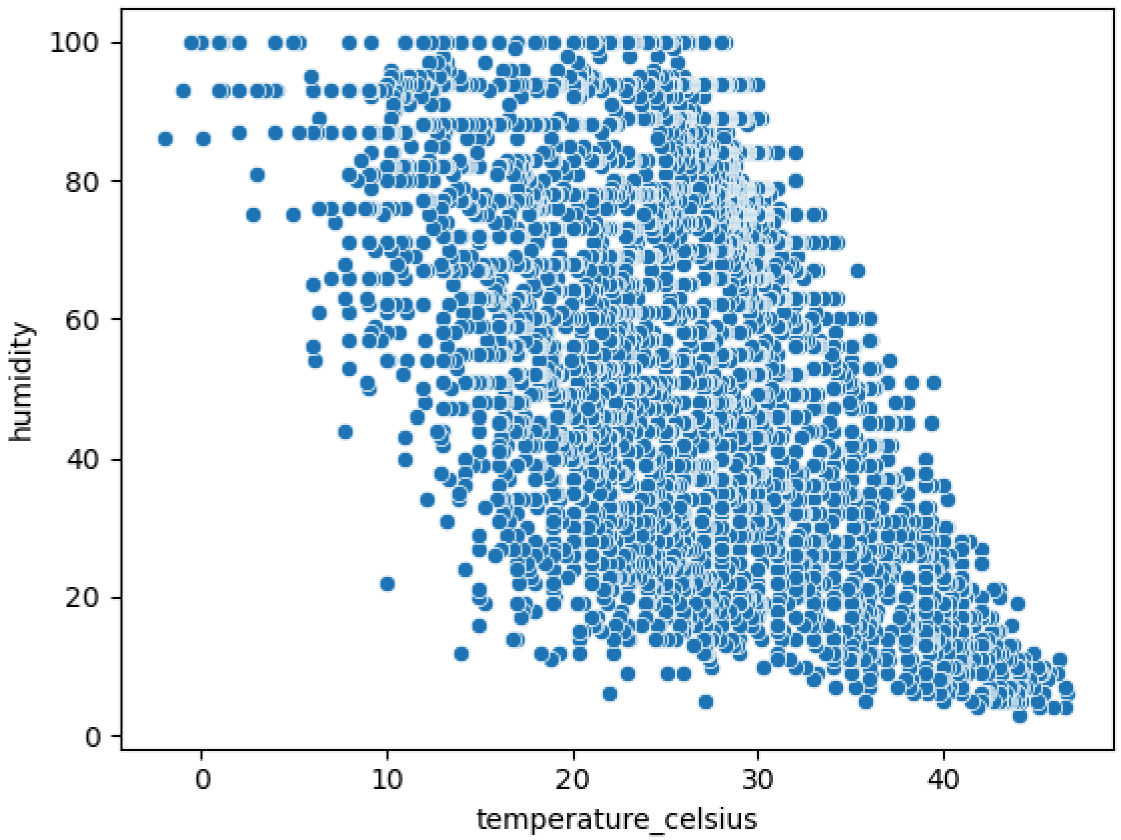
```
25.655317220543807
```

Is there a correlation between 'temperature_celsius' and 'humidity'?

```
In [19]: corr=df['temperature_celsius'].corr(df.humidity)
         print(corr)
```

```
-0.41090700101358013
```

```
In [20]: sn.scatterplot(x=df.temperature_celsius,y=df.humidity)
         plt.show()
```



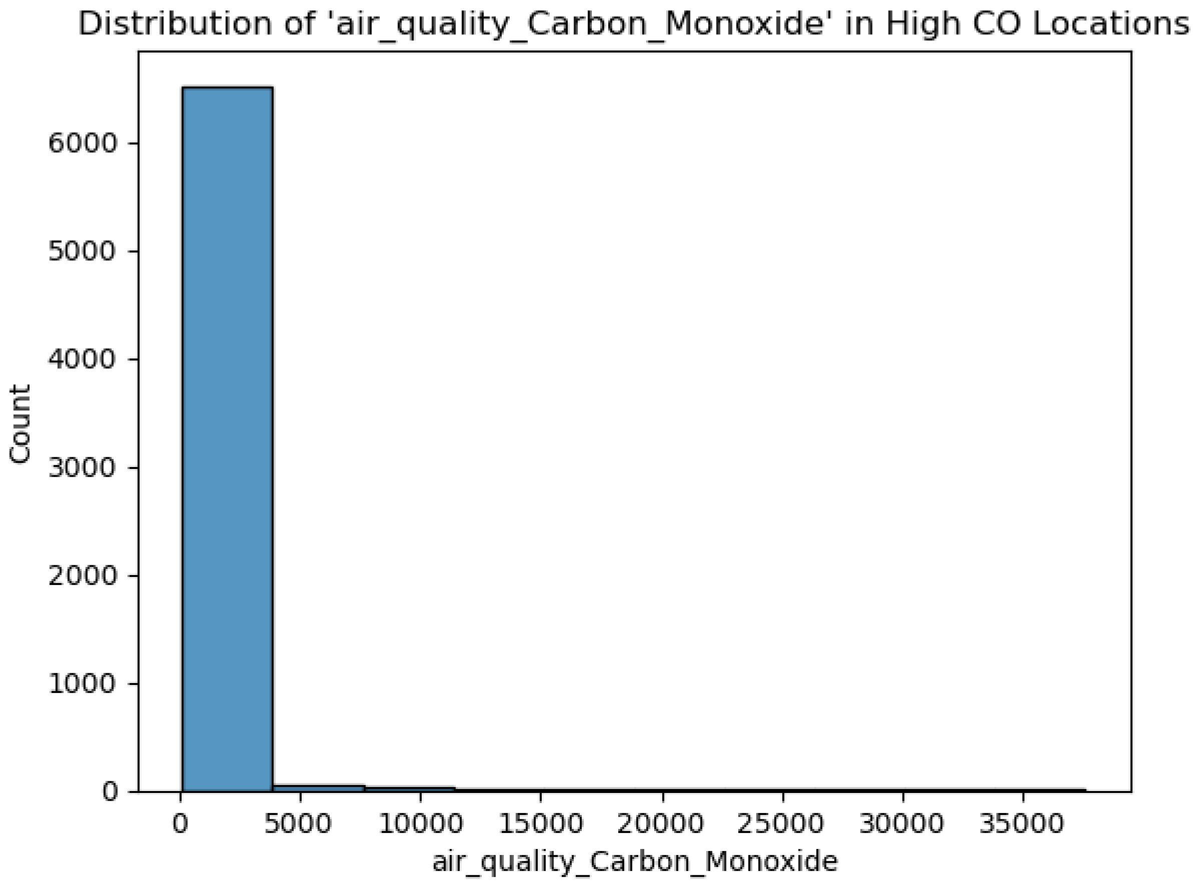
Are there any locations where the 'air_quality_Carbon_Monoxide' is above a certain threshold?

```
In [21]: df.air_quality_Nitrogen_dioxide.mean()
```

```
Out[21]: 9.795377643504532
```

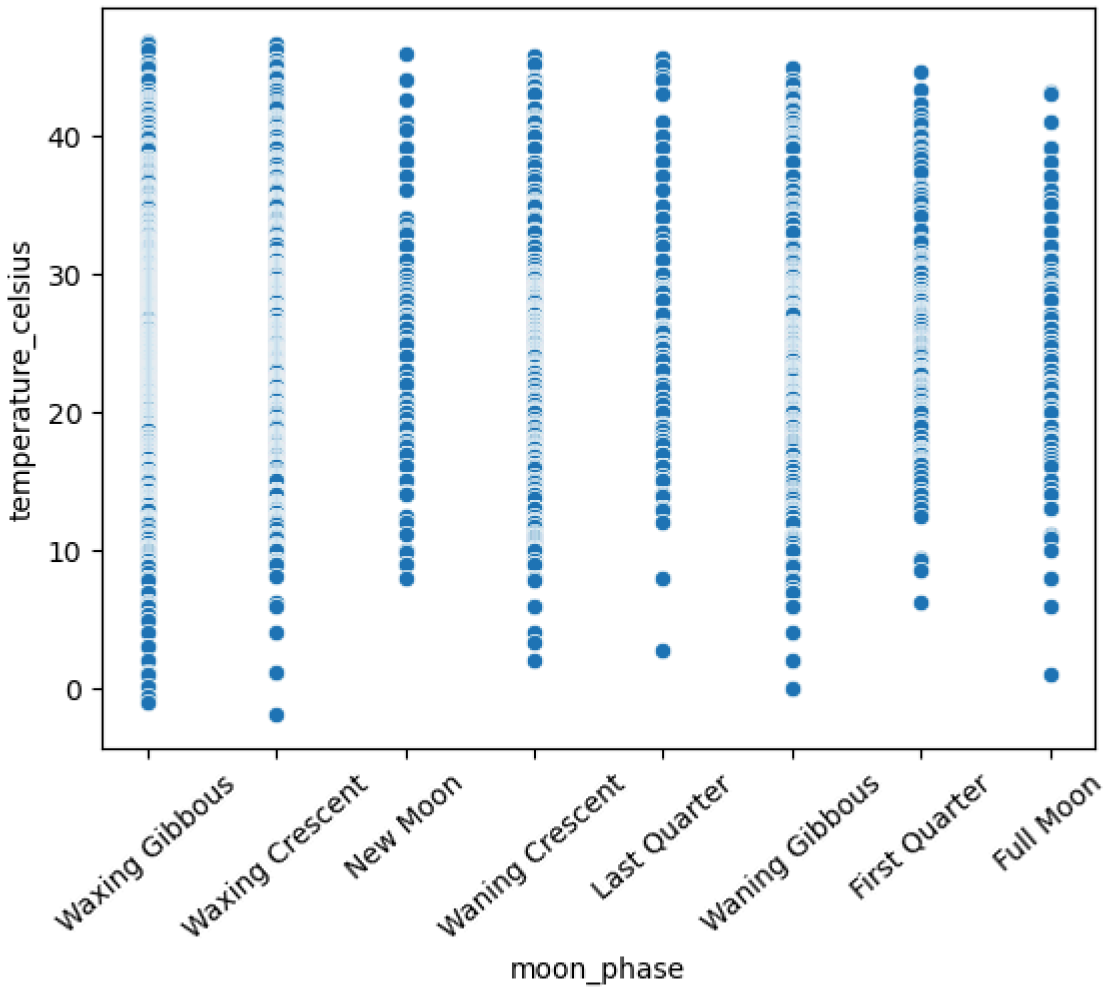
```
In [22]: thershold=9.795377643504532
         high=df[df.air_quality_Carbon_Monoxide>thershold]
         #print(high)
         sn.histplot(x=high['air_quality_Carbon_Monoxide'],bins=10)
         plt.title("Distribution of 'air_quality_Carbon_Monoxide' in High CO Locations")
         plt.show()
```

```
C:\Users\SSD\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: us
e_inf_as_na option is deprecated and will be removed in a future version. Convert in
f values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

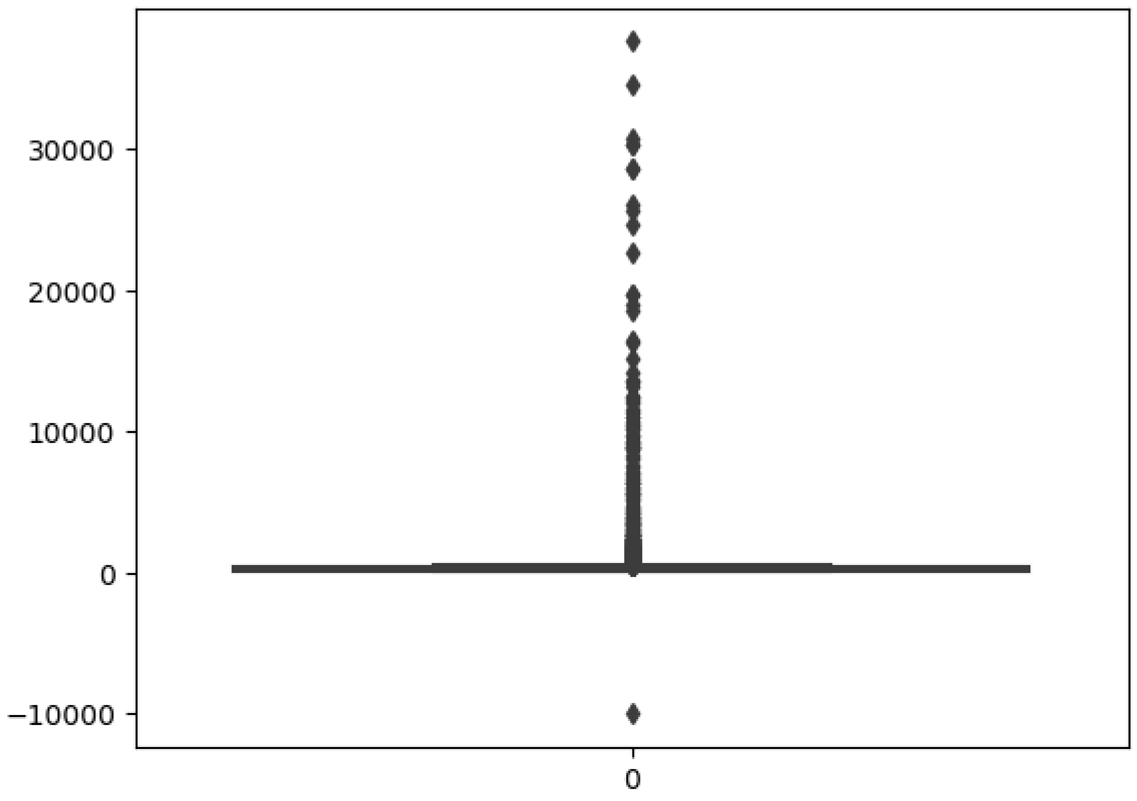


Is there any relationship between 'moon_phase' and 'temperature_celsius'?

```
In [38]: sn.scatterplot(x=df.moon_phase,y=df.temperature_celsius,alpha=1)
         plt.xticks(rotation=41)
         plt.show()
```



```
In [41]: sn.boxplot(df.air_quality_Carbon_Monoxide) # finding outliers
         plt.show()
```



```
In [ ]:
```