In [1]:
```
# Install & Import json pandas

# To mange datarames and data
# ! pip install pandas
import pandas as pd

# To normalize json estraction
# Install & Import json to read json
import json

# To do information requesto API
# Install & Import request data for API
# ! pip install requests
import requests

# To time stamp in UNIX format
# Get time stamps
from datetime import datetime, timezone, timedelta
```

In [2]:
```
# # Get timestap for 1 year ago


# Get current UTC time
utc_now = datetime.utcnow()
NowTStmap = int(utc_now.replace(tzinfo=timezone.utc).timestamp())

# Calculate the date one year ago (364 days ago)
one_year_ago = utc_now - timedelta(days=364)

# Get Unix timestamp for one year ago in UTC
StartDate = int(one_year_ago.replace(tzinfo=timezone.utc).timestamp())



print(f"The Unix timestamp for one year(364 days) ago in UTC is: {StartDate}")
```

```
The Unix timestamp for one year(364 days) ago in UTC is: 1675570477
```

In [3]:
```
# Current time stamp verification
NowTStmap
```

Out[3]:
```
1707020077
```

In [4]:
```
# Initial time stap verificatio
StartDate
```

Out[4]:
```
1675570477
```

In [5]:
```
# View start date of date extraction

# Convert Unix timestamp to a datetime object
IntialDate = datetime.utcfromtimestamp(StartDate)

IntialDateView = IntialDate.strftime("%Y-%m-%d %H:%M:%S")

print(f"This is going to be start date and time date is: {IntialDateView}")
```

```
This is going to be start date and time date is: 2023-02-05 04:14:37
```

In [6]:
```
# Open Weather API extration as a string created with variables

#URL Extraction form API OW documentation Documentation sample
# https://history.openweathermap.org/data/2.5/history/city?lat={lat}&lon={lon}

# We need to define input variables for data extraction

lat = "37.77493"
lon = "-122.41942"
# In terms of extraction for OW API 168 represents a Week
cnt = "168"
appid = '6ca2e283a3b779eb2b4a87a89af49444'
units="metric"

# Conver strat date timestamp to string
STRStartDate = str(StartDate)


# Build string for data extraction

full_url = "https://history.openweathermap.org/data/2.5/history/city?"+"lat="+
# First initial URL
full_url
```

Out[6]:
```
'https://history.openweathermap.org/data/2.5/history/city?lat=37.77493&lon=-
122.41942&type=hour&start=1675570477&cnt=168&units=metric&appid=6ca2e283a3b7
79eb2b4a87a89af49444'
```

In [7]:
```
# Loop that append/stack extracted infomraiton per week

# Initialize emty data frame for df2
df2 = pd.DataFrame()

# loop of weeks to extract
count = 0
WeekStamp = StartDate
while (WeekStamp < NowTStmap):
    # Count shows week number added
    count = count + 1
    # Convert time stamp to human and print for visual control
    WeekDate = datetime.utcfromtimestamp(WeekStamp)
    IntialDateView = WeekDate.strftime("%Y-%m-%d %H:%M:%S")
    print(f" Week:  {count}"+ f" WeekDate {WeekDate}"  )
    # Print time stamp for url visual control
    WeekStamp
    WeekStampSTR = str(WeekStamp)
    full_url = "https://history.openweathermap.org/data/2.5/history/city?"+"la
    print(f" Full URL {full_url}"  )
    print(f" Timestamp {WeekStamp}"  )

    # Read new week data from API in json format
    r = requests.get(url = full_url)
    dataAPI = r.json()

    # Dataframes append/add using concatenate function
    df1 = pd.json_normalize(dataAPI['list'],max_level=3)
    df1 = pd.DataFrame(df1)
    df2 = pd.concat([df1, df2], ignore_index=True, axis=0)

    # Add 1 week in seconds for new timestamp input
    WeekStamp = WeekStamp + 604800
```

```
 Week:  28 WeekDate 2023-08-13 04:14:37
 Full URL https://history.openweathermap.org/data/2.5/history/city?lat=3
7.77493&lon=-122.41942&type=hour&start=1691900077&cnt=168&units=metric&ap
pid=6ca2e283a3b779eb2b4a87a89af49444 (https://history.openweathermap.org/
data/2.5/history/city?lat=37.77493&lon=-122.41942&type=hour&start=1691900
077&cnt=168&units=metric&appid=6ca2e283a3b779eb2b4a87a89af49444)
 Timestamp 1691900077
 Week:  29 WeekDate 2023-08-20 04:14:37
 Full URL https://history.openweathermap.org/data/2.5/history/city?lat=3
7.77493&lon=-122.41942&type=hour&start=1692504877&cnt=168&units=metric&ap
pid=6ca2e283a3b779eb2b4a87a89af49444 (https://history.openweathermap.org/
data/2.5/history/city?lat=37.77493&lon=-122.41942&type=hour&start=1692504
877&cnt=168&units=metric&appid=6ca2e283a3b779eb2b4a87a89af49444)
 Timestamp 1692504877
 Week:  30 WeekDate 2023-08-27 04:14:37
 Full URL https://history.openweathermap.org/data/2.5/history/city?lat=3
7.77493&lon=-122.41942&type=hour&start=1693109677&cnt=168&units=metric&ap
pid=6ca2e283a3b779eb2b4a87a89af49444 (https://history.openweathermap.org/
data/2.5/history/city?lat=37.77493&lon=-122.41942&type=hour&start=1693109
677&cnt=168&units=metric&appid=6ca2e283a3b779eb2b4a87a89af49444)
```

In [8]:
```
# Appended data set verification and backup copy to work with
print(df2)
dfALL = pd.DataFrame()
dfALL = df2
```

```
             dt                                                weather  \
0     1706418000  [{'id': 803, 'main': 'Clouds', 'description': ...
1     1706421600  [{'id': 804, 'main': 'Clouds', 'description': ...
2     1706425200  [{'id': 803, 'main': 'Clouds', 'description': ...
3     1706428800  [{'id': 803, 'main': 'Clouds', 'description': ...
4     1706432400  [{'id': 803, 'main': 'Clouds', 'description': ...
...          ...                                               ...
8731  1676160000  [{'id': 802, 'main': 'Clouds', 'description': ...
8732  1676163600  [{'id': 802, 'main': 'Clouds', 'description': ...
8733  1676167200  [{'id': 802, 'main': 'Clouds', 'description': ...
8734  1676170800  [{'id': 801, 'main': 'Clouds', 'description': ...
8735  1676174400  [{'id': 803, 'main': 'Clouds', 'description': ...

      main.temp  main.feels_like  main.pressure  main.humidity  main.temp_mi
n  \
0         14.58            14.05           1023             75          12.2
2
1         14.46            13.92           1023             75          11.6
6
2         14.13            13.56           1023             75          11.1
1
3         14.23            13.61           1022             73          11.1
1
4         14.05            13.47           1021             75          11.1
4
...         ...              ...            ...            ...
...
8731      12.43            11.37           1012             63          10.6
0
8732      11.47            10.50           1013             70          10.0
0
8733      10.12             9.09           1013             73           8.1
1
8734       8.98             6.69           1014             77           6.5
7
8735       8.91             7.87           1014             76           5.9
6

      main.temp_max  wind.speed  wind.deg  clouds.all  wind.gust  rain.1h  \
0             15.95        3.60        20          75        NaN      NaN
1             16.08        3.13        83         100       5.81      NaN
2             15.77        3.13        85          75       4.92      NaN
3             16.12        4.02        68          75       7.15      NaN
4             15.77        6.71        45          75      10.28      NaN
...             ...         ...       ...         ...        ...      ...
8731          14.34        8.94       315          40      10.28      NaN
8732          13.65        7.60       315          40       8.49      NaN
8733          11.88        5.66       280          40        NaN      NaN
8734          10.76        4.12       310          20        NaN      NaN
8735          10.77        2.06       250          75        NaN      NaN

      rain.3h
0         NaN
1         NaN
2         NaN
3         NaN
4         NaN
```

```
...       ...
8731      NaN
8732      NaN
8733      NaN
8734      NaN
8735      NaN

[8736 rows x 14 columns]
```

In [9]:
```
# Statistical description of data set
# Libraries neededpip install pandas

# Library for chart plots
#! pip install matplotlib.pyplot
import matplotlib.pyplot as plt
#! pip install math
import math
#! pip install statistics
import statistics
#! pip install numpy
import numpy as np
#! pip install scipy.stats
import scipy.stats

# This all requeries pandas as pd but pandas was importe before
```

In [10]:
```
# List varaibles/colums in dataframe

list(dfALL.columns)
```

Out[10]:
```
['dt',
 'weather',
 'main.temp',
 'main.feels_like',
 'main.pressure',
 'main.humidity',
 'main.temp_min',
 'main.temp_max',
 'wind.speed',
 'wind.deg',
 'clouds.all',
 'wind.gust',
 'rain.1h',
 'rain.3h']
```

In [11]:
```
# Range

## dt
print(f"Range of dt Min:  {np.amin(dfALL['dt'])}")
print(f"Range of dt Max:  {np.amax(dfALL['dt'])}")

# main.temp

print(f"Range of main.temp Min:  {np.amin(dfALL['main.temp'])}")
print(f"Range of main.temp Max:  {np.amax(dfALL['main.temp'])}")

# main.feels_like

print(f"Range of main.feels_like Min:  {np.amin(dfALL['main.feels_like'])}")
print(f"Range of main.feels_like Max:  {np.amax(dfALL['main.feels_like'])}")

# main.pressure

print(f"Range of main.pressure Min:  {np.amin(dfALL['main.pressure'])}")
print(f"Range of main.pressure Max:  {np.amax(dfALL['main.pressure'])}")

# main.humidity

print(f"Range of main.humidity Min:  {np.amin(dfALL['main.humidity'])}")
print(f"Range of main.humidity Max:  {np.amax(dfALL['main.humidity'])}")

# main.temp_min

print(f"Range of main.temp_min Min:  {np.amin(dfALL['main.temp_min'])}")
print(f"Range of main.temp_min Max:  {np.amax(dfALL['main.temp_min'])}")

# main.temp_max

print(f"Range of main.temp_max Min:  {np.amin(dfALL['main.temp_max'])}")
print(f"Range of main.temp_max Max:  {np.amax(dfALL['main.temp_max'])}")

# wind.speed

print(f"Range of wind.speed Min:  {np.amin(dfALL['wind.speed'])}")
print(f"Range of wind.speed Max:  {np.amax(dfALL['wind.speed'])}")

# wind.deg
print(f"Range of wind.deg Min:  {np.amin(dfALL['wind.deg'])}")
print(f"Range of wind.deg Max:  {np.amax(dfALL['wind.deg'])}")

# wind.gust
print(f"Range of wind.gust Min:  {np.amin(dfALL['wind.gust'])}")
print(f"Range of wind.gust Max:  {np.amax(dfALL['wind.gust'])}")

# clouds.all

print(f"Range of clouds.all Min:  {np.amin(dfALL['clouds.all'])}")
print(f"Range of clouds.all Max:  {np.amax(dfALL['clouds.all'])}")

# rain.1h

print(f"Range of rain.1h Min:  {np.amin(dfALL['rain.1h'])}")
```

```
print(f"Range of rain.1h Max:  {np.amax(dfALL['rain.1h'])}")

# rain.3h

print(f"Range of rain.3h Min:  {np.amin(dfALL['rain.3h'])}")
print(f"Range of rain.3h Max:  {np.amax(dfALL['rain.3h'])}")
```

```
Range of dt Min:  1675573200
Range of dt Max:  1707019200
Range of main.temp Min:  3.21
Range of main.temp Max:  33.48
Range of main.feels_like Min:  -2.29
Range of main.feels_like Max:  32.68
Range of main.pressure Min:  983
Range of main.pressure Max:  1031
Range of main.humidity Min:  18
Range of main.humidity Max:  95
Range of main.temp_min Min:  -0.06
Range of main.temp_min Max:  31.21
Range of main.temp_max Min:  4.95
Range of main.temp_max Max:  41.84
Range of wind.speed Min:  0.0
Range of wind.speed Max:  23.25
Range of wind.deg Min:  0
Range of wind.deg Max:  360
Range of wind.gust Min:  0.45
Range of wind.gust Max:  32.41
Range of clouds.all Min:  0
Range of clouds.all Max:  100
Range of rain.1h Min:  0.1
Range of rain.1h Max:  7.43
Range of rain.3h Min:  1.0
Range of rain.3h Max:  1.0
```

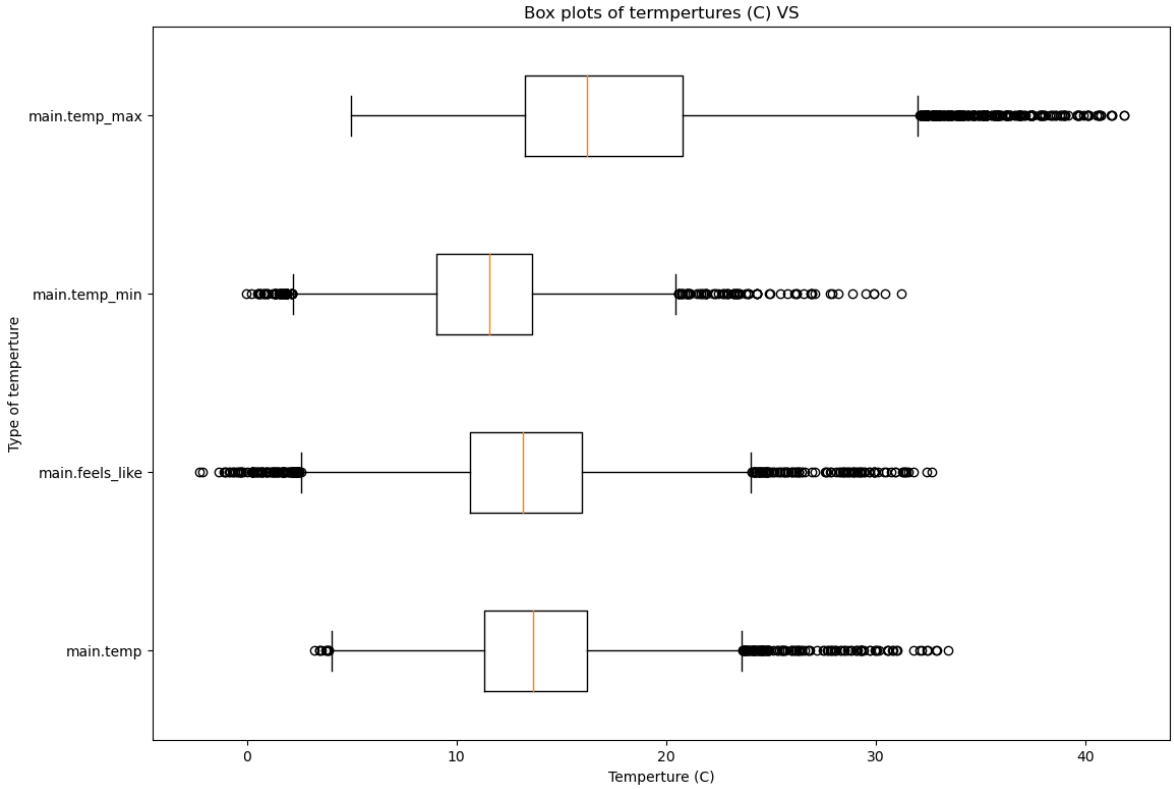
In [12]:
```
# Prepare data with date, year, month, day and in human format.

# Ininitialize data set
dfdate = pd.DataFrame()

dfALL['datetime'] = pd.to_datetime(dfALL['dt'], unit='s')
dfALL['Date'] = dfALL['datetime'].dt.date
dfALL['Year'] = dfALL['datetime'].dt.year
dfALL['Month'] = dfALL['datetime'].dt.month
dfALL['Day'] = dfALL['datetime'].dt.day
dfALL['Hour'] = dfALL['datetime'].dt.hour

# Get week of year
# Convert 'Date' to date format
dfALL['Date'] = pd.to_datetime(dfALL['Date'])
dfALL['Week_of_Year'] = dfALL['Date'].dt.isocalendar().week

# dfALL['Year_Month'] = str(str(dfALL['Year'])+'-'+str(dfALL['Month']))

print(dfALL)
```

```
             dt                                                weather  \
0     1706418000  [{'id': 803, 'main': 'Clouds', 'description': ...
1     1706421600  [{'id': 804, 'main': 'Clouds', 'description': ...
2     1706425200  [{'id': 803, 'main': 'Clouds', 'description': ...
3     1706428800  [{'id': 803, 'main': 'Clouds', 'description': ...
4     1706432400  [{'id': 803, 'main': 'Clouds', 'description': ...
...          ...                                                ...
8731  1676160000  [{'id': 802, 'main': 'Clouds', 'description': ...
8732  1676163600  [{'id': 802, 'main': 'Clouds', 'description': ...
8733  1676167200  [{'id': 802, 'main': 'Clouds', 'description': ...
8734  1676170800  [{'id': 801, 'main': 'Clouds', 'description': ...
8735  1676174400  [{'id': 803, 'main': 'Clouds', 'description': ...

      main.temp  main.feels_like  main.pressure  main.humidity  main.temp_mi
n  \
0         14.58            14.05           1023             75          12.2
2
1         14.46            13.92           1023             75          11.6
6
2         14.13            13.56           1023             75          11.1
1
3         14.23            13.61           1022             73          11.1
1
4         14.05            13.47           1021             75          11.1
4
...         ...              ...            ...            ...
...
8731      12.43            11.37           1012             63          10.6
0
8732      11.47            10.50           1013             70          10.0
0
8733      10.12             9.09           1013             73           8.1
1
8734       8.98             6.69           1014             77           6.5
7
8735       8.91             7.87           1014             76           5.9
6

      main.temp_max  wind.speed  wind.deg  ...  wind.gust  rain.1h  rain.3h
\
0             15.95        3.60        20  ...        NaN      NaN      NaN
1             16.08        3.13        83  ...       5.81      NaN      NaN
2             15.77        3.13        85  ...       4.92      NaN      NaN
3             16.12        4.02        68  ...       7.15      NaN      NaN
4             15.77        6.71        45  ...      10.28      NaN      NaN
...             ...         ...       ...  ...        ...      ...      ...
8731          14.34        8.94       315  ...      10.28      NaN      NaN
8732          13.65        7.60       315  ...       8.49      NaN      NaN
8733          11.88        5.66       280  ...        NaN      NaN      NaN
8734          10.76        4.12       310  ...        NaN      NaN      NaN
8735          10.77        2.06       250  ...        NaN      NaN      NaN

                datetime       Date  Year  Month  Day  Hour  Week_of_Year
0    2024-01-28 05:00:00 2024-01-28  2024      1   28     5             4
1    2024-01-28 06:00:00 2024-01-28  2024      1   28     6             4
2    2024-01-28 07:00:00 2024-01-28  2024      1   28     7             4
3    2024-01-28 08:00:00 2024-01-28  2024      1   28     8             4
```

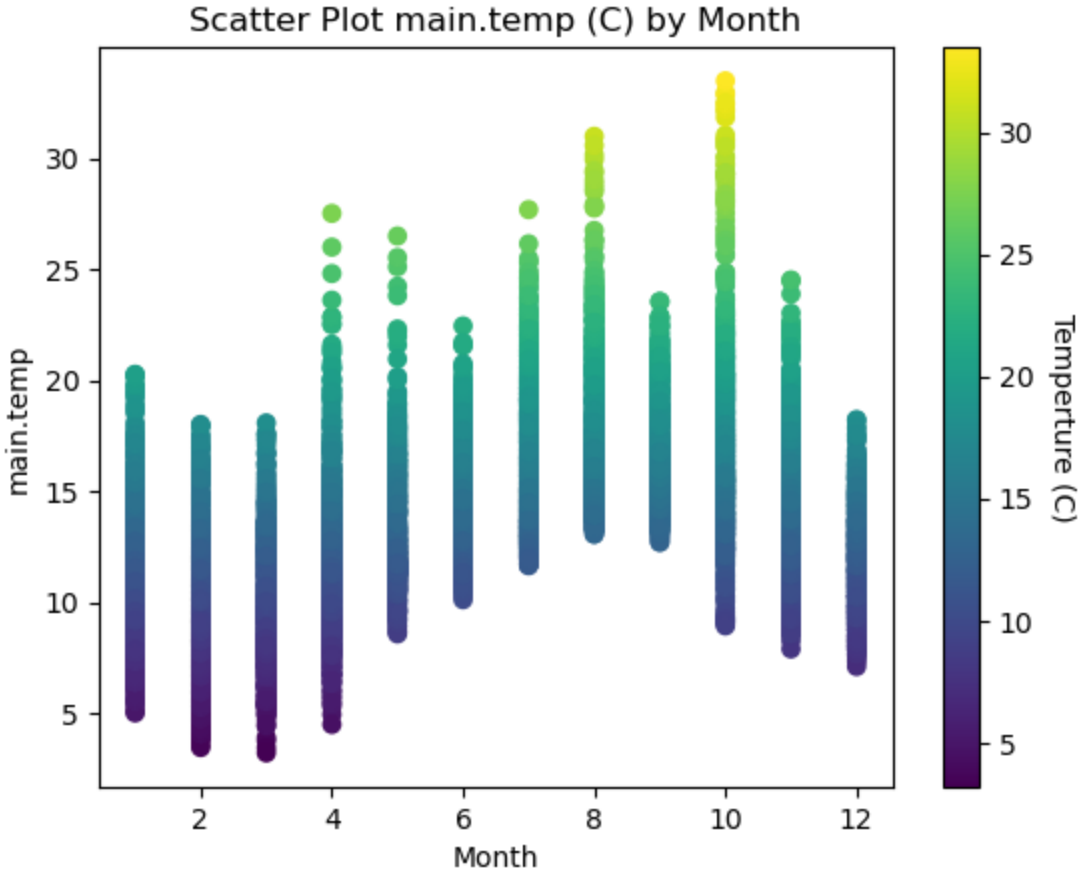
```
4    2024-01-28 09:00:00 2024-01-28  2024      1   28     9             4
...                  ...        ...   ...    ...  ...   ...           ...
8731 2023-02-12 00:00:00 2023-02-12  2023      2   12     0             6
8732 2023-02-12 01:00:00 2023-02-12  2023      2   12     1             6
8733 2023-02-12 02:00:00 2023-02-12  2023      2   12     2             6
8734 2023-02-12 03:00:00 2023-02-12  2023      2   12     3             6
8735 2023-02-12 04:00:00 2023-02-12  2023      2   12     4             6

[8736 rows x 21 columns]
```

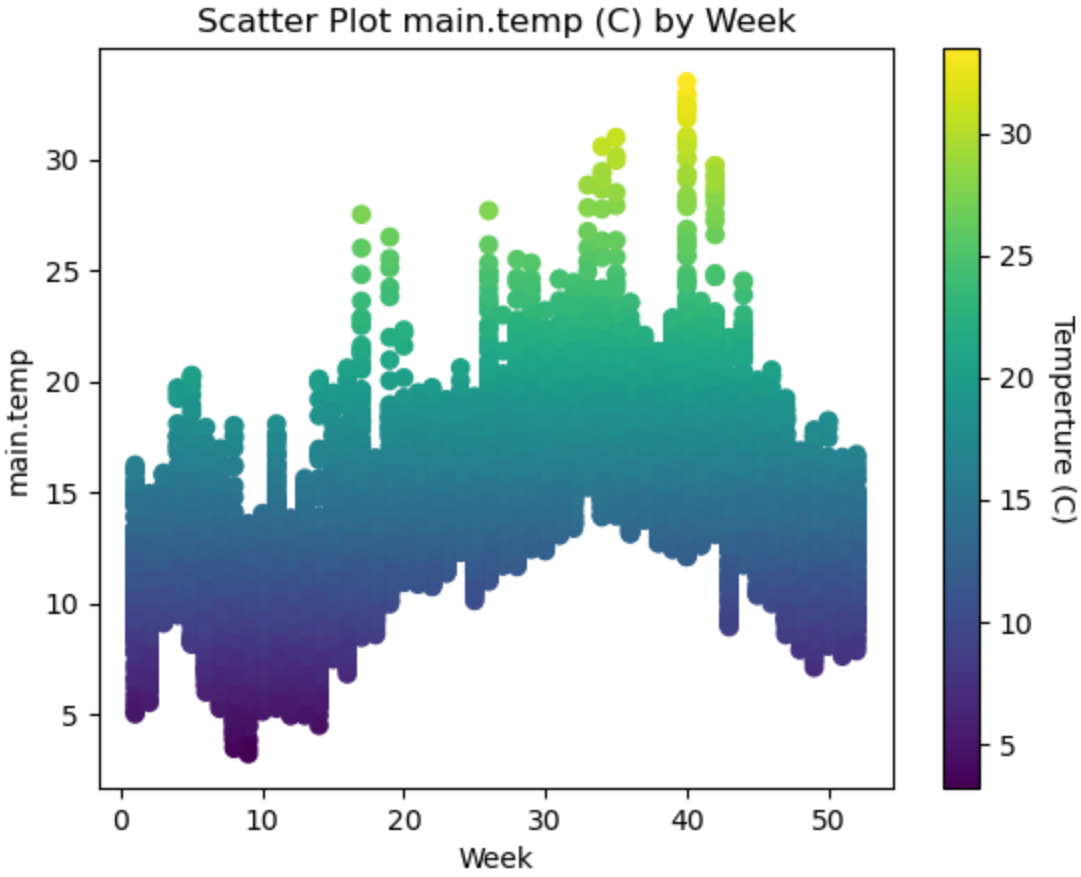
In [13]:
```
# Box plot termpertures

# Difine fugure for plot
fig = plt.figure(figsize =(10, 7))

# Creating axes instance
ax = fig.add_axes([0, 0, 1, 1])

# Creating plot
column_namesTEMP = ['main.temp', 'main.feels_like', 'main.temp_min','main.temp
selected_columnsTEMP = dfALL[column_namesTEMP]
dfALL[column_namesTEMP]

ax.set_yticklabels(column_namesTEMP)

bp = ax.boxplot(dfALL[column_namesTEMP], vert = 0)

# Add title and labels
plt.title("Box plots of termpertures (C) VS")
plt.xlabel('Temperture (C)')
plt.ylabel('Type of temperture')

# show plot
plt.show()
```

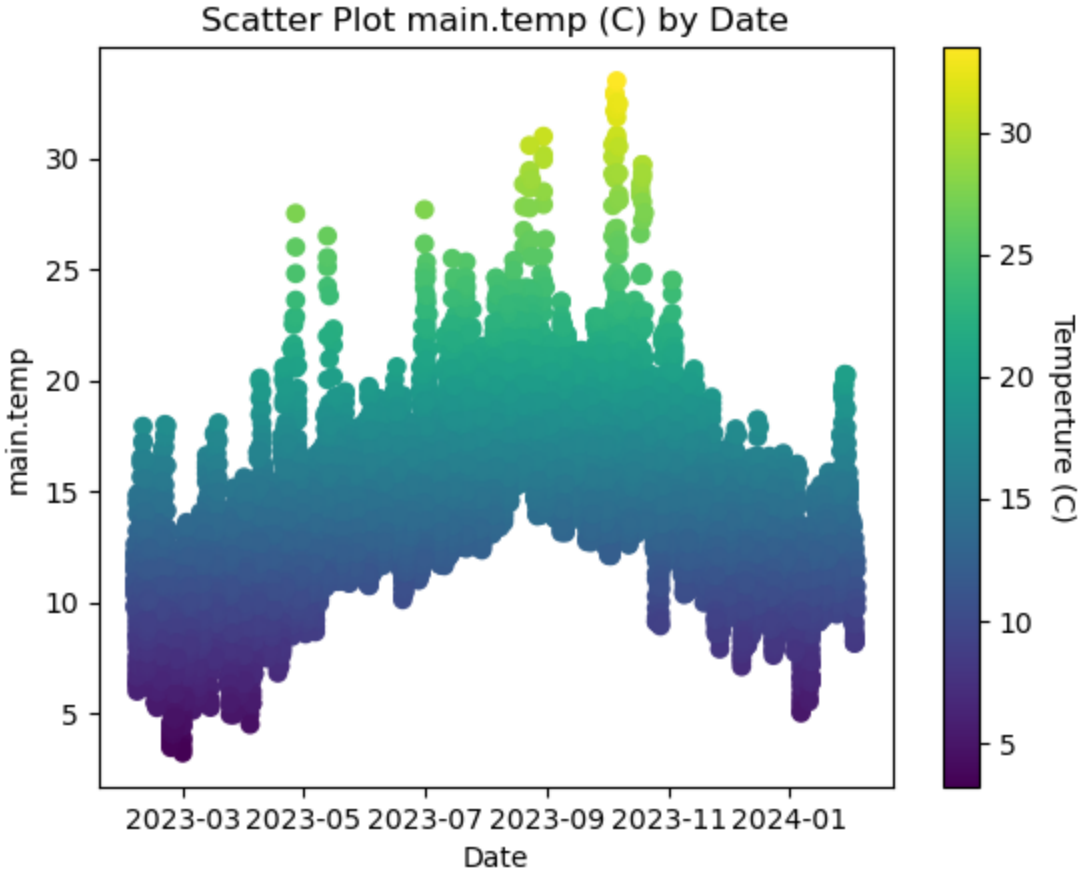
```
C:\Users\mario\AppData\Local\Temp\ipykernel_2880\3785422921.py:14: UserWarni
ng: FixedFormatter should only be used together with FixedLocator
  ax.set_yticklabels(column_namesTEMP)
```



In [14]:
```
# Scatter plot over time

# Scatter plot by months
plt.scatter(dfALL['Month'], dfALL['main.temp'], c=dfALL['main.temp'])

# Adding Title to the Plot
plt.title("Scatter Plot main.temp (C) by Month")

# Setting the X and Y labels
plt.xlabel('Month')
plt.ylabel('main.temp')

# Plot color bar
plt.colorbar().set_label('Temperture (C)', rotation=270, labelpad=15)

# Scatter by Week
plt.figure()

plt.scatter(dfALL['Week_of_Year'], dfALL['main.temp'], c=dfALL['main.temp'])

# Adding Title to the Plot
plt.title("Scatter Plot main.temp (C) by Week")

# Setting the X and Y labels
plt.xlabel('Week')
plt.ylabel('main.temp')

# Plot color bar
plt.colorbar().set_label('Temperture (C)', rotation=270, labelpad=15)

# Scatter by Date
plt.figure()

plt.scatter(dfALL['Date'], dfALL['main.temp'], c=dfALL['main.temp'])

# Adding Title to the Plot
plt.title("Scatter Plot main.temp (C) by Date")

# Setting the X and Y labels
plt.xlabel('Date')
plt.ylabel('main.temp')

plt.colorbar().set_label('Temperture (C)', rotation=270, labelpad=15)

plt.show()
```

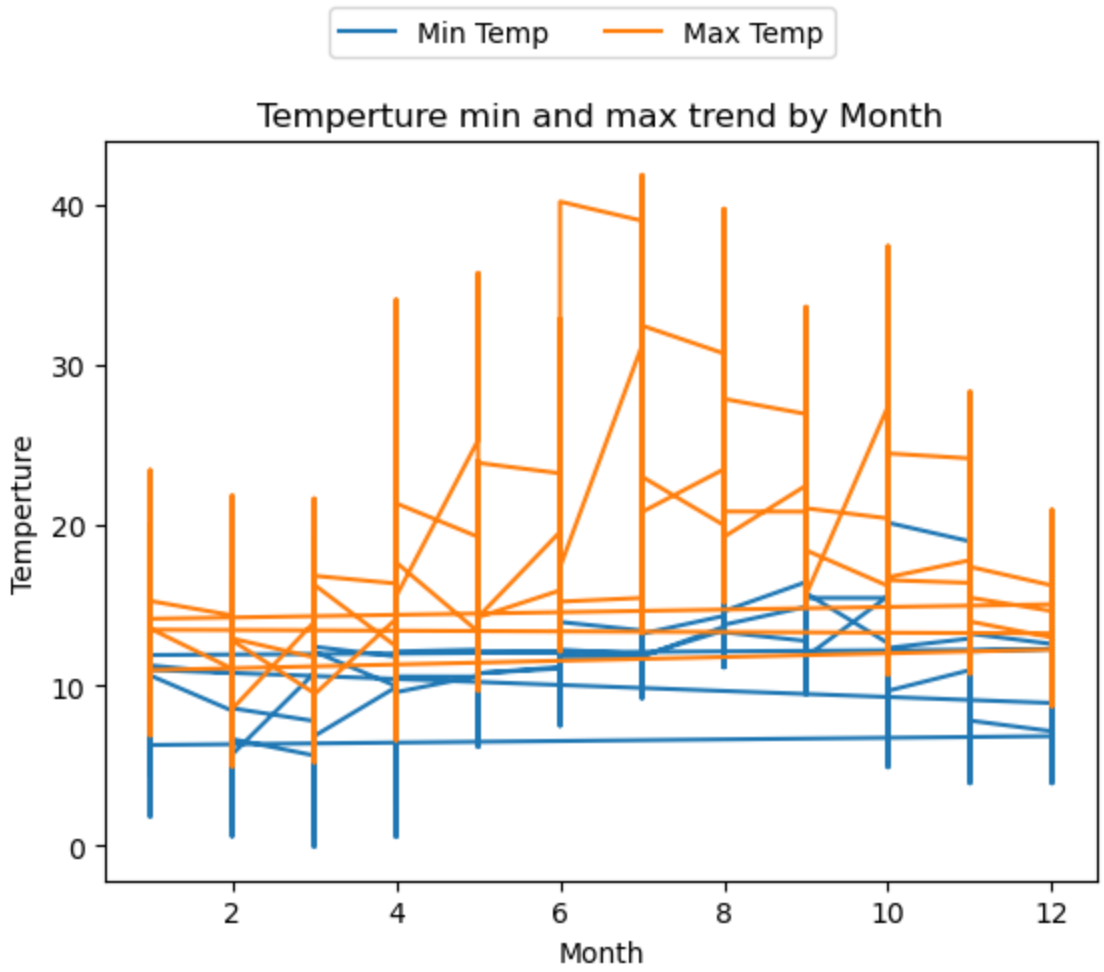

Scatter Plot main.temp (C) by Month
main.temp
Month
Temperture (C)
5
10
15
20
25
30
2
4
6
8
10
12

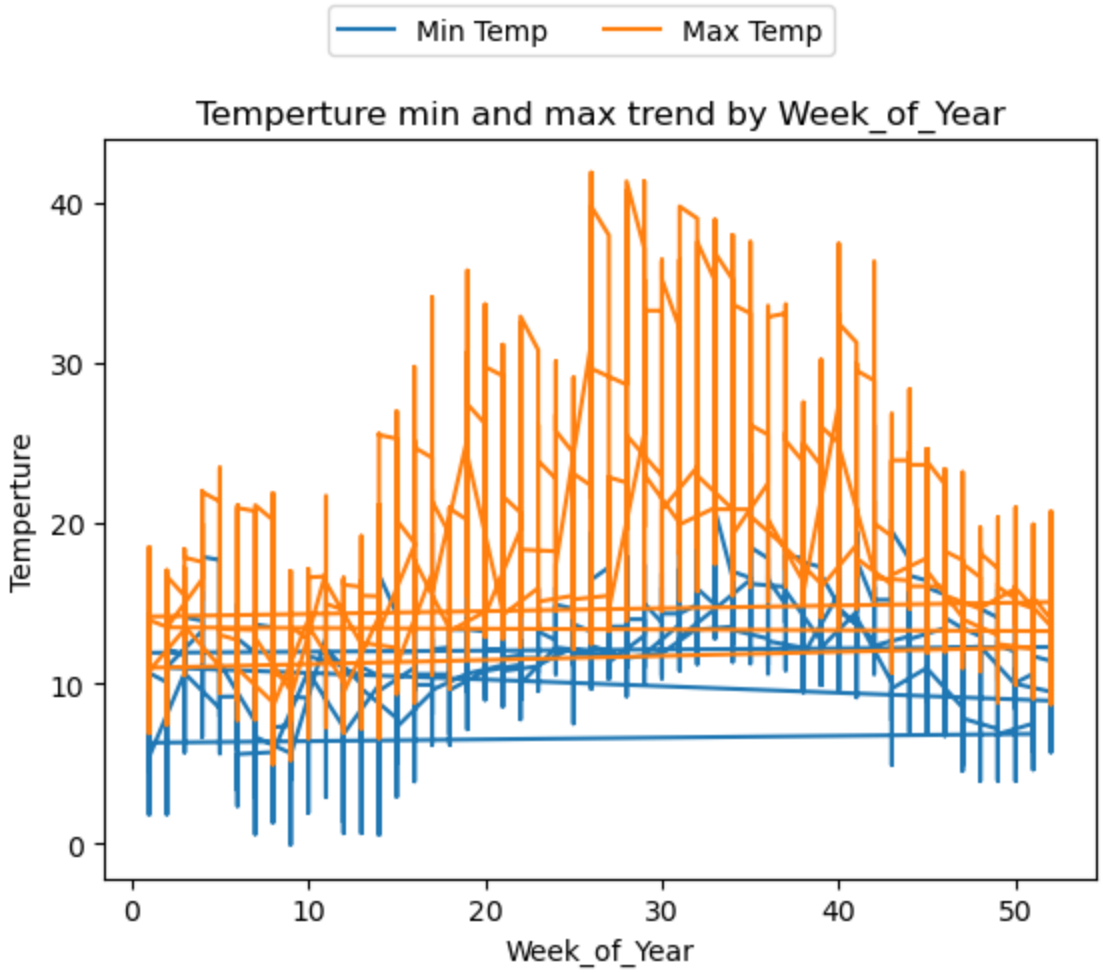

Scatter Plot main.temp (C) by Week
main.temp
Week
Temperture (C)
5
10
15
20
25
30
0
10
20
30
40
50

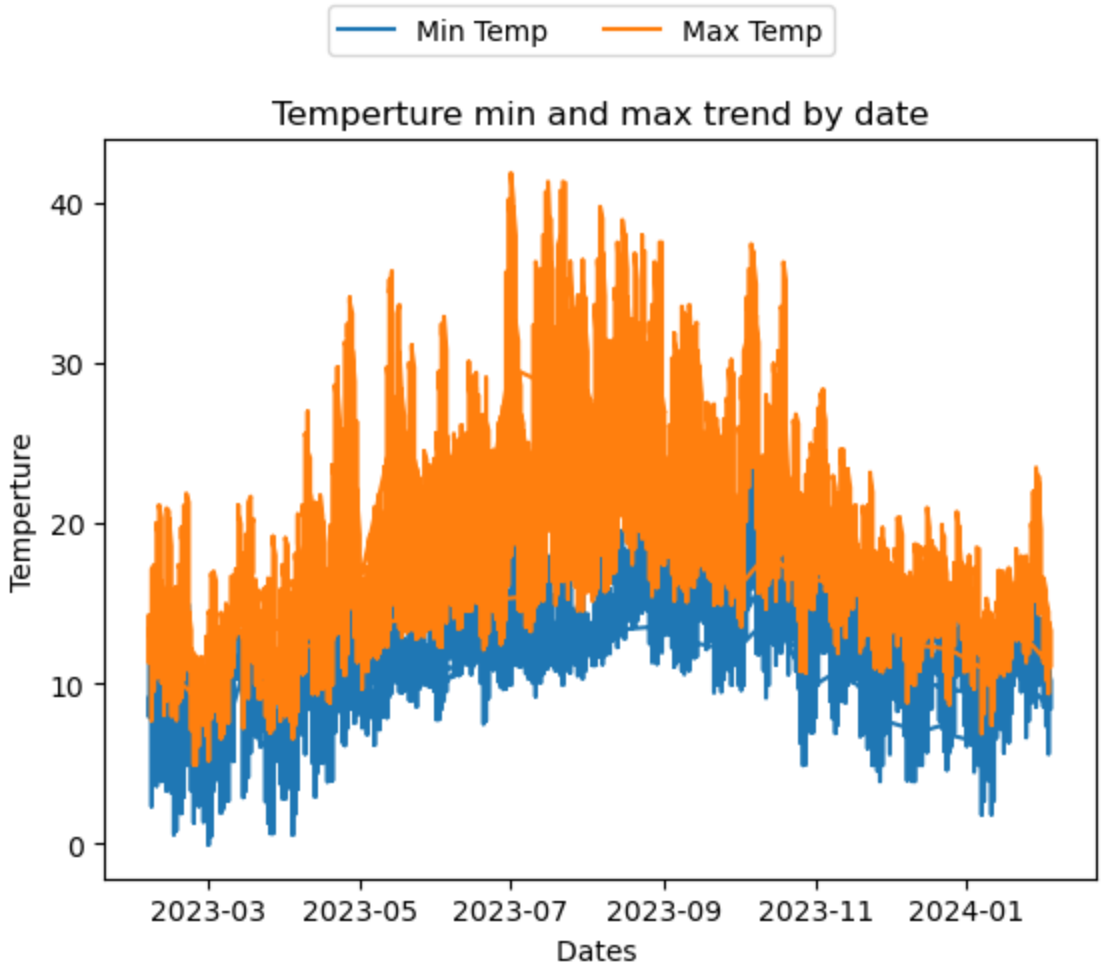

Scatter Plot main.temp (C) by Date
main.temp
30
25
20
15
10
5
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01
Date
Temperture (C)
30
25
20
15
10
5

In [15]:
```
# import matplotlib.pyplot as plt
# import numpy as np

# first plot with Month and MIX temperture dat
plt.plot(dfALL['Month'], dfALL['main.temp_min'], label="Min Temp")

# second plot with Month and MAX temperture data
plt.plot(dfALL['Month'], dfALL['main.temp_max'], label="Max Temp")

# Legend of data
# plt.legend(["Min Temp", "Max Temp"], loc="upper right")

plt.legend(bbox_to_anchor=(0.75, 1.2), ncol=2)

plt.xlabel("Month")
plt.ylabel("Temperture")
plt.title('Temperture min and max trend by Month')

# Figure to add 2nd chart
plt.figure()

# first plot with Month and MIX temperture dat
plt.plot(dfALL['Week_of_Year'], dfALL['main.temp_min'], label="Min Temp")

# second plot with Month and MAX temperture data
plt.plot(dfALL['Week_of_Year'], dfALL['main.temp_max'], label="Max Temp")

# Legend of data
# plt.legend(["Min Temp", "Max Temp"], loc="upper right")

plt.legend(bbox_to_anchor=(0.75, 1.2), ncol=2)

plt.xlabel("Week_of_Year")
plt.ylabel("Temperture")
plt.title('Temperture min and max trend by Week_of_Year')


# Figure to add 2rd chart
plt.figure()

# first plot with Date and MIX temperture dat
plt.plot(dfALL['Date'], dfALL['main.temp_min'], label="Min Temp")

# second plot with Date and MAX temperture data
plt.plot(dfALL['Date'], dfALL['main.temp_max'], label="Max Temp")

# Legend of data
# plt.legend(["Min Temp", "Max Temp"], loc="upper right")

plt.legend(bbox_to_anchor=(0.75, 1.2), ncol=2)

plt.xlabel("Dates ")
plt.ylabel("Temperture")
plt.title('Temperture min and max trend by date')

# plot all charts
```

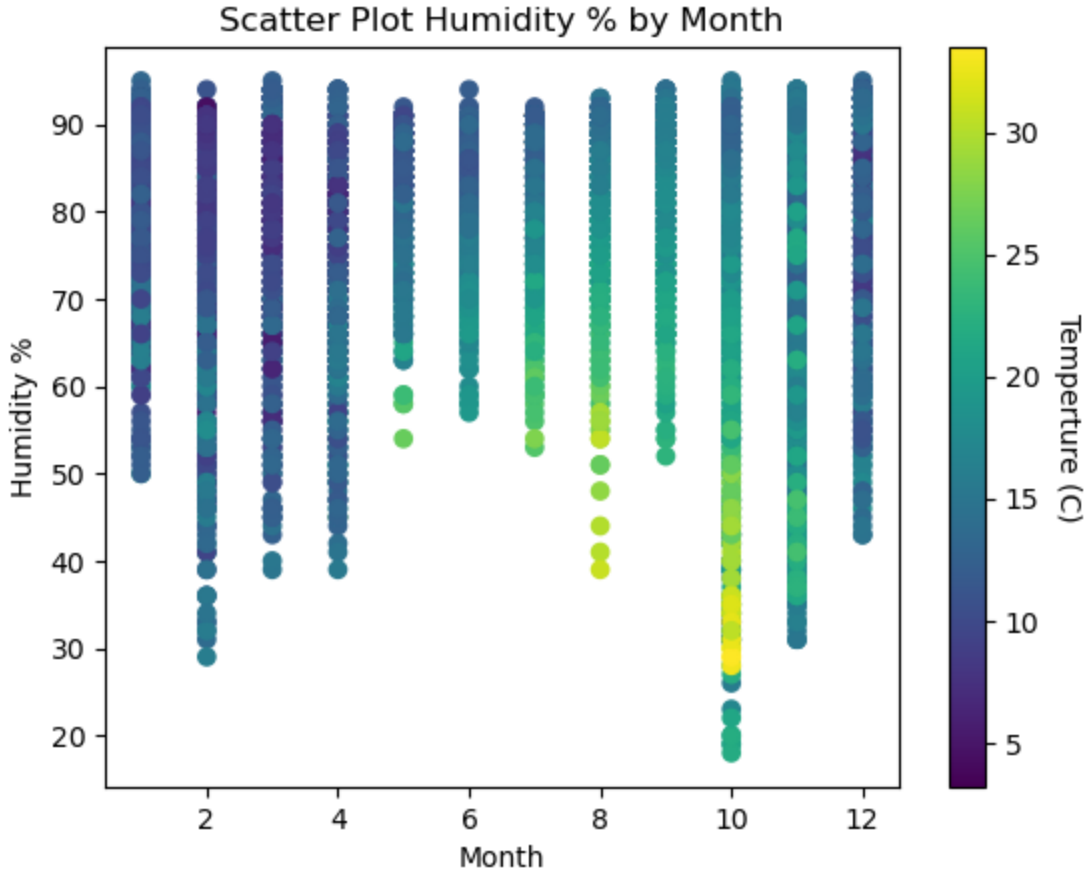
```
plt.show()
```

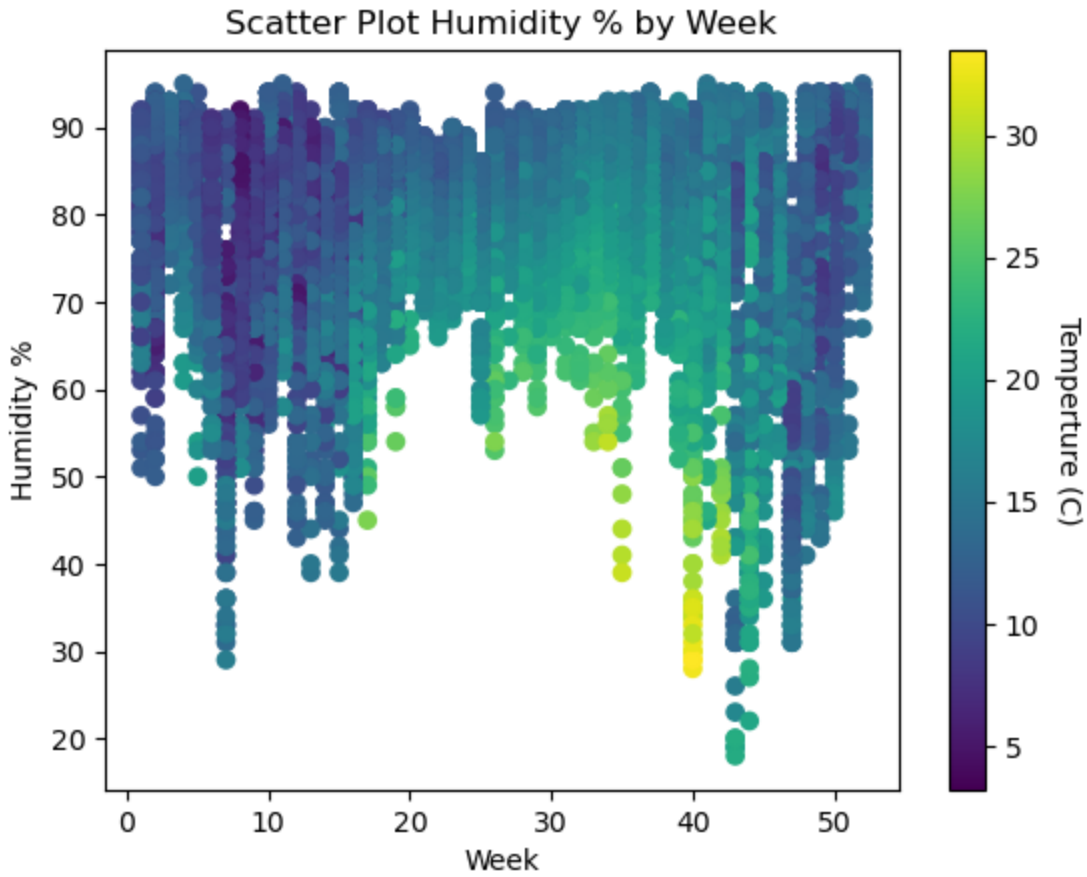

Min Temp
Max Temp
Temperture min and max trend by Month
Temperture
Month
0
10
20
30
40
2
4
6
8
10
12

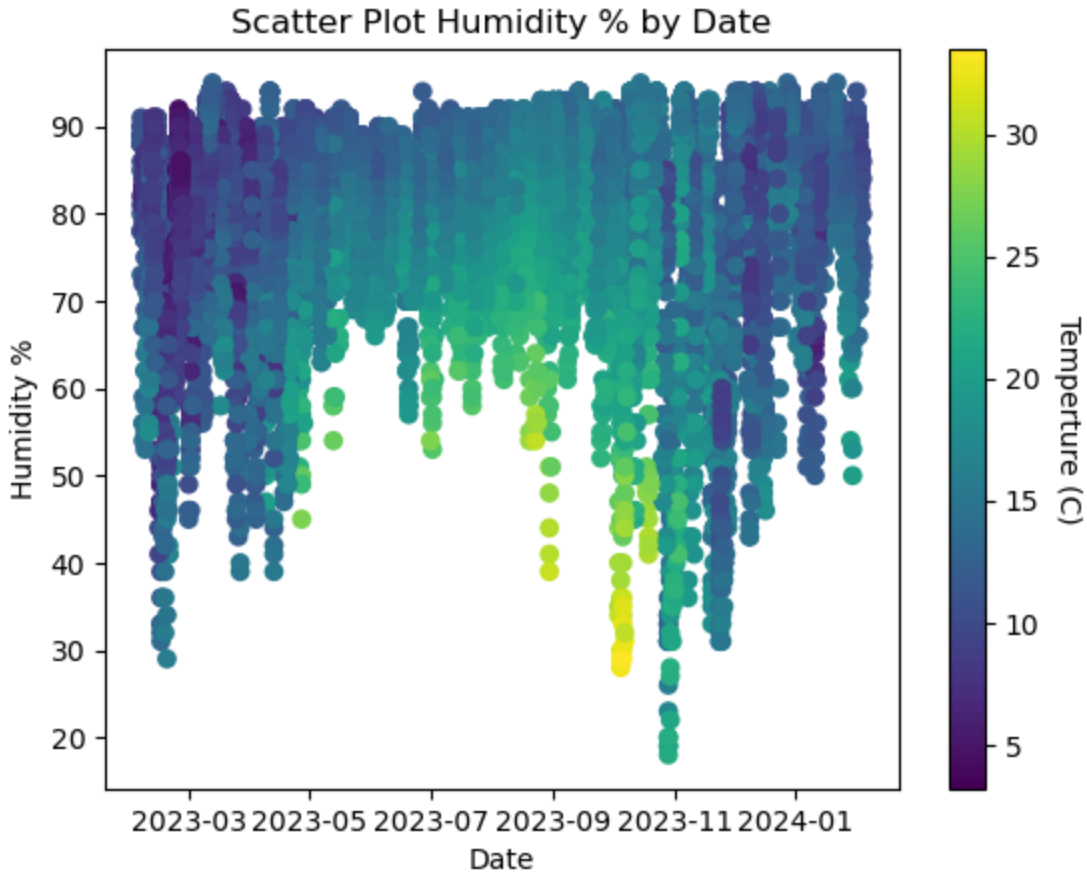

Min Temp
Max Temp
Temperture min and max trend by Week_of_Year
Temperture
Week_of_Year
0
10
20
30
40
50


Min Temp
Max Temp
Temperture min and max trend by date
Temperture
0
10
20
30
40
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01
Dates

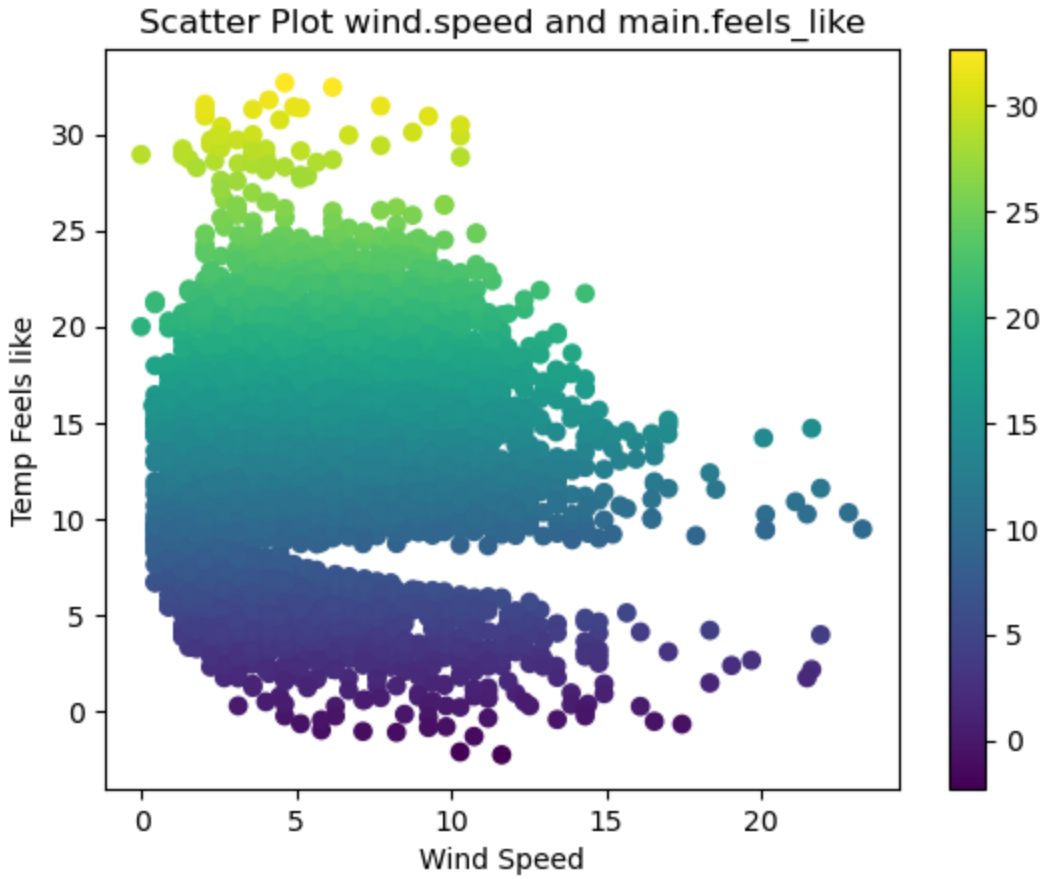
In [16]:
```
# Plots

import matplotlib.pyplot as plt

# Scatter plot by months
plt.scatter(dfALL['Month'], dfALL['main.humidity'], c=dfALL['main.temp'])

# Adding Title to the Plot
plt.title("Scatter Plot Humidity % by Month")

# Setting the X and Y labels
plt.xlabel('Month')
plt.ylabel('Humidity %')

# Plot color bar
plt.colorbar().set_label('Temperture (C)', rotation=270, labelpad=15)

# Scatter by Week
plt.figure()

plt.scatter(dfALL['Week_of_Year'], dfALL['main.humidity'], c=dfALL['main.temp'

# Adding Title to the Plot
plt.title("Scatter Plot Humidity % by Week")

# Setting the X and Y labels
plt.xlabel('Week')
plt.ylabel('Humidity %')

# Plot color bar
plt.colorbar().set_label('Temperture (C)', rotation=270, labelpad=15)

# Scatter by Date
plt.figure()

plt.scatter(dfALL['Date'], dfALL['main.humidity'], c=dfALL['main.temp'])

# Adding Title to the Plot
plt.title("Scatter Plot Humidity % by Date")

# Setting the X and Y labels
plt.xlabel('Date')
plt.ylabel('Humidity %')

# Plot color with label
plt.colorbar().set_label('Temperture (C)', rotation=270, labelpad=15)

plt.show()
```

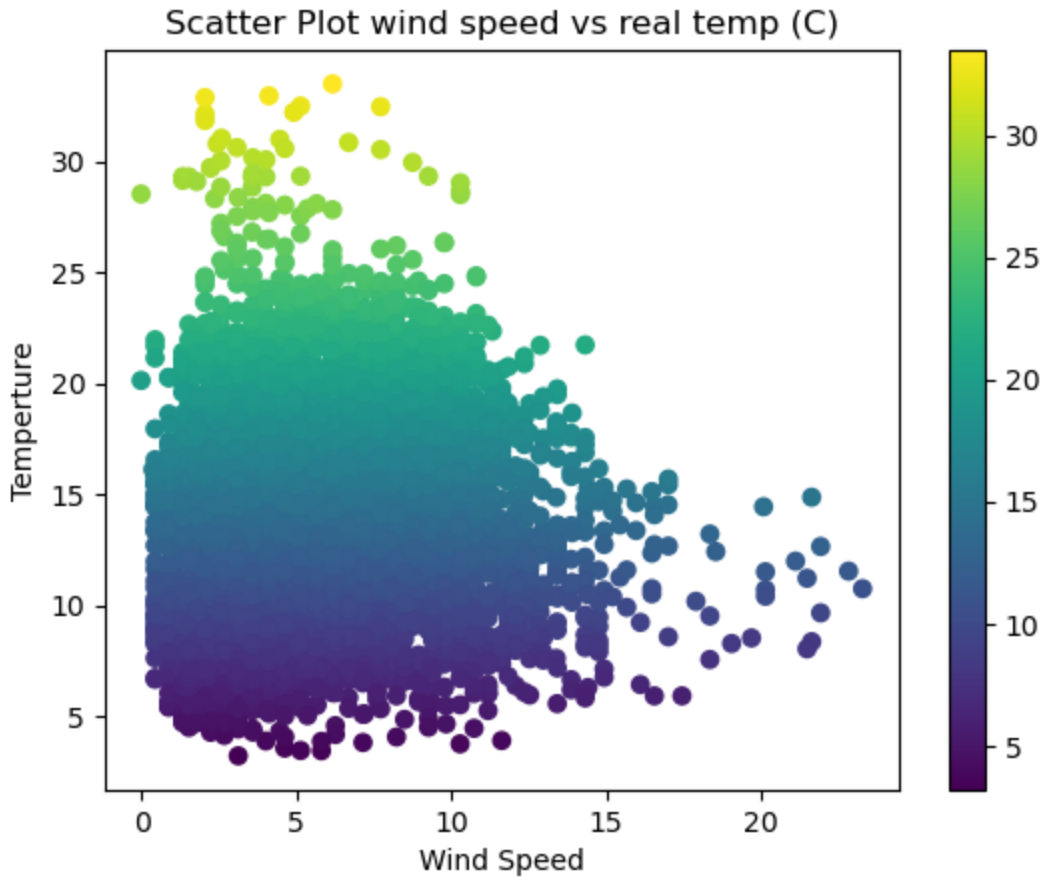

Scatter Plot Humidity % by Month
Humidity %
Month
Temperture (C)
90
80
70
60
50
40
30
20
2
4
6
8
10
12
30
25
20
15
10
5


Scatter Plot Humidity % by Week
Humidity %
Week
Temperture (C)
90
80
70
60
50
40
30
20
0
10
20
30
40
50
30
25
20
15
10
5

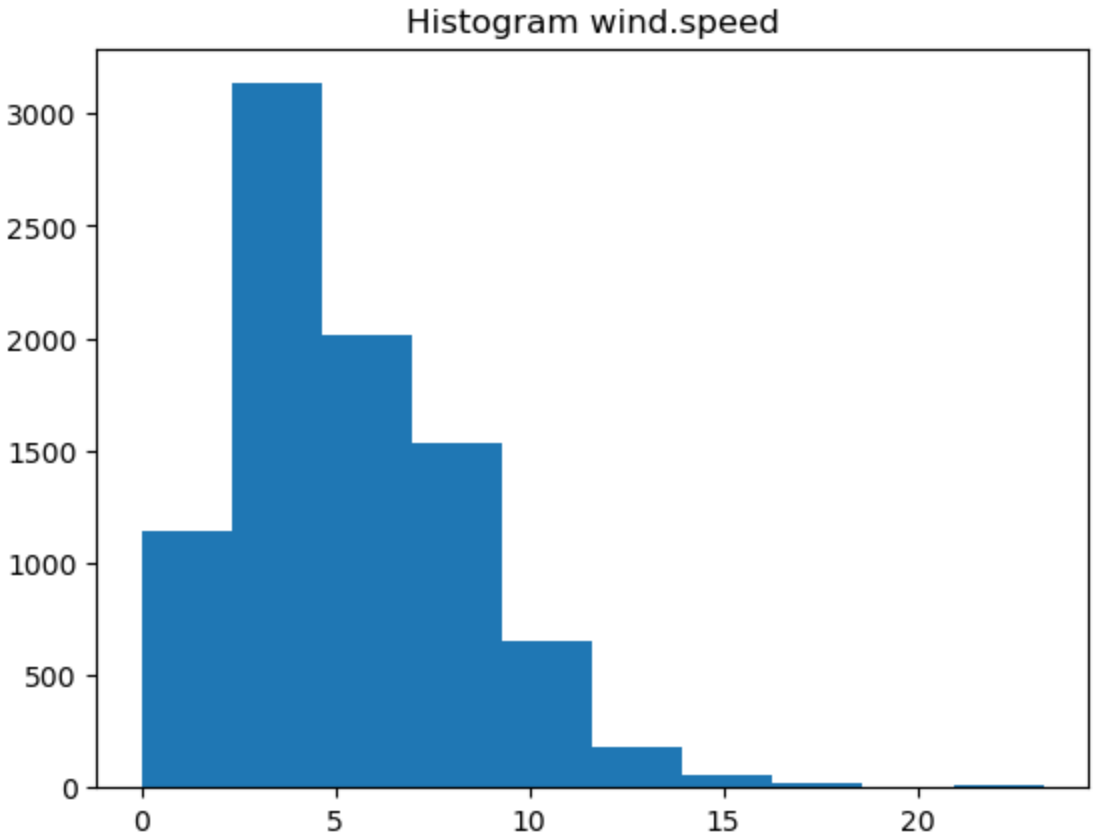

Scatter Plot Humidity % by Date
Humidity %
90
80
70
60
50
40
30
20
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01
Date
Temperture (C)
30
25
20
15
10
5

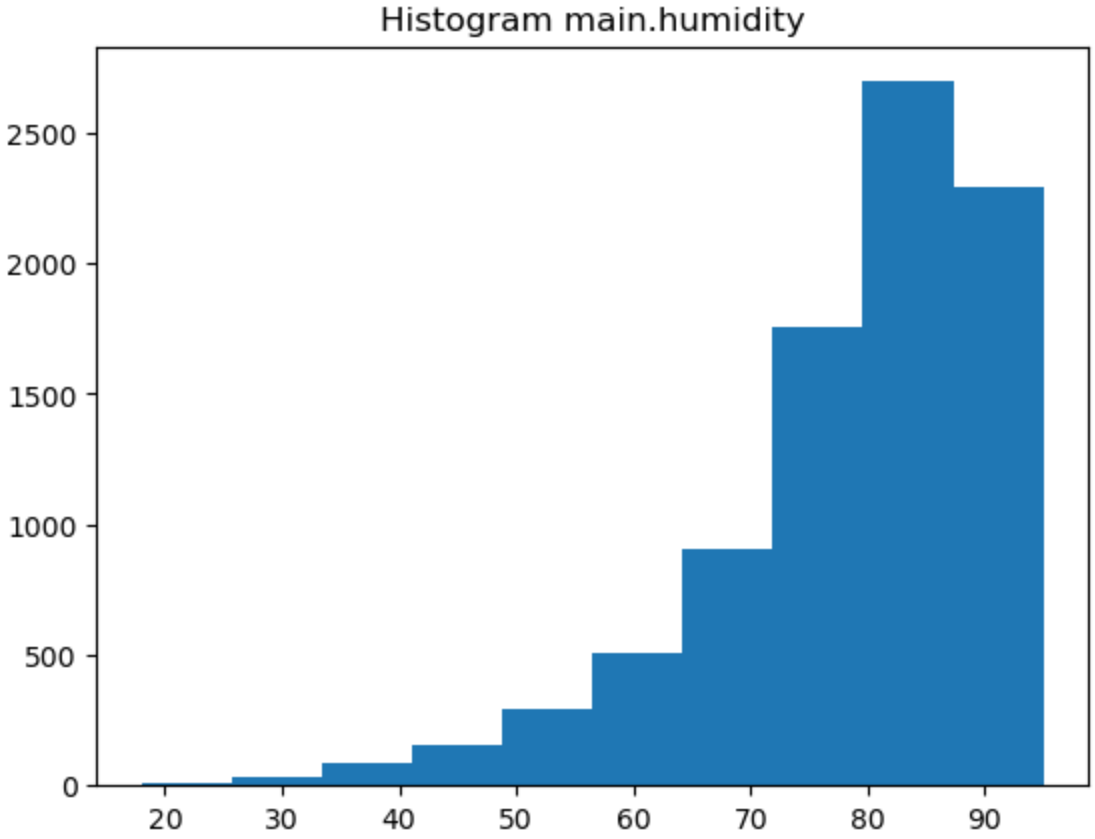
In [17]:
```
# Relation between temper feels like and wind speed.

# Scatter plot wind speed vs Temp Feels Like
plt.scatter(dfALL['wind.speed'], dfALL['main.feels_like'], c=dfALL['main.feels

# Adding Title to the Plot
plt.title("Scatter Plot wind.speed and main.feels_like")

# Setting the X and Y labels
plt.xlabel('Wind Speed')
plt.ylabel('Temp Feels like')

# Plot color bar
plt.colorbar()

# Second chart
plt.figure()

# Scatter plot wind speed vs real temp
plt.scatter(dfALL['wind.speed'], dfALL['main.temp'], c=dfALL['main.temp'])

# Adding Title to the Plot
plt.title("Scatter Plot wind speed vs real temp (C)")

# Setting the X and Y labels
plt.xlabel('Wind Speed')
plt.ylabel('Temperture')

plt.colorbar()

plt.show()
# Wind helps to have a lower temperture felling and in temperture feel like th
```

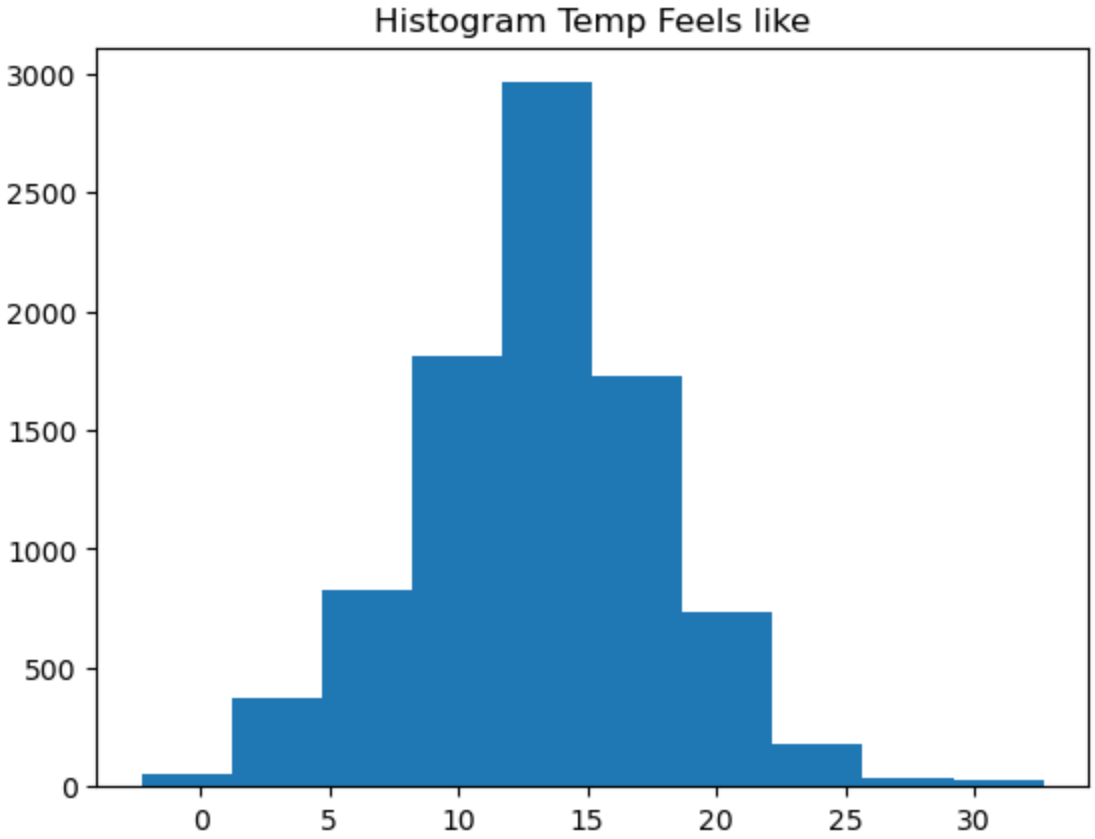

Scatter Plot wind.speed and main.feels_like
Temp Feels like
Wind Speed
0
5
10
15
20
25
30


Scatter Plot wind speed vs real temp (C)
Temperture
Wind Speed
5
10
15
20
25
30

In [18]:
```
# histogram of varaibles of interes
# Adding various charts to one output

plt.hist(dfALL['wind.speed'])
plt.title("Histogram wind.speed")

# Adding the legends
plt.show()

# This function helps to add other chart

plt.figure()

plt.hist(dfALL['main.humidity'])
plt.title("Histogram main.humidity")

# Adding the legends
plt.show()

plt.figure()

plt.hist(dfALL['main.feels_like'])
plt.title("Histogram Temp Feels like")

# Adding the legends
plt.show()
```



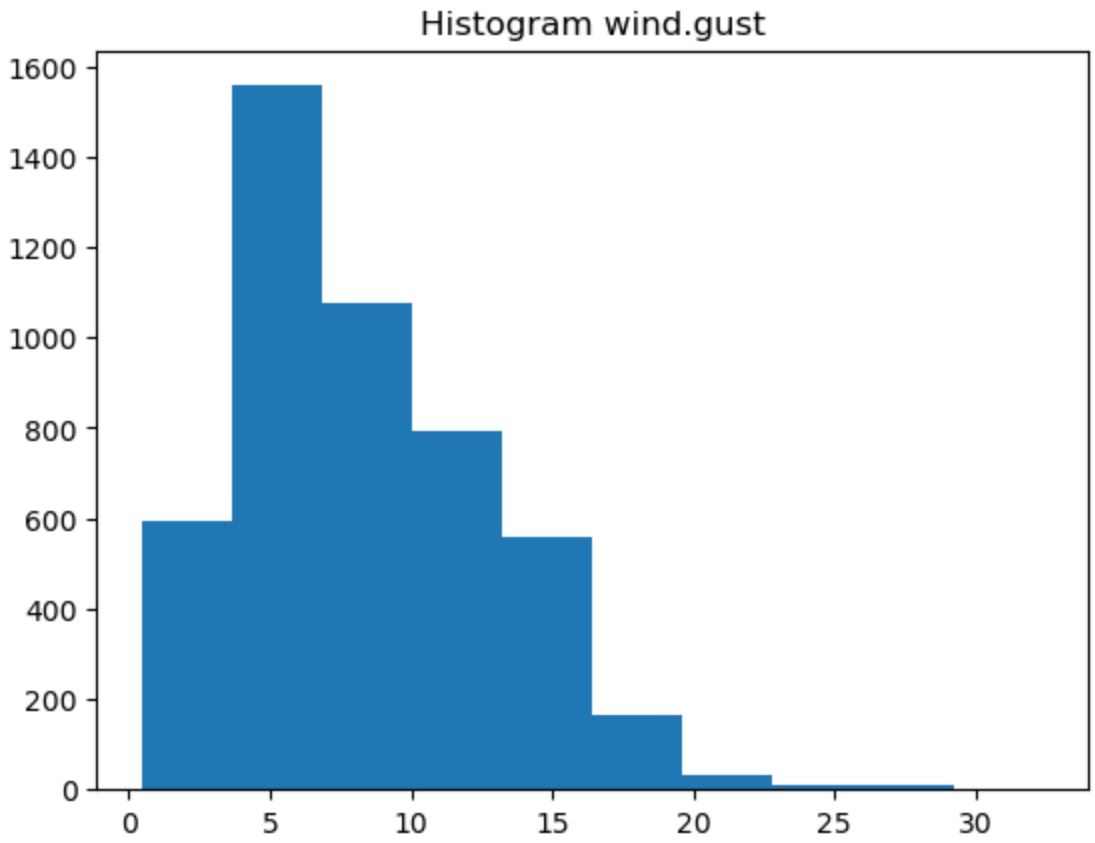

Histogram main.humidity
2500
2000
1500
1000
500
0
20
30
40
50
60
70
80
90

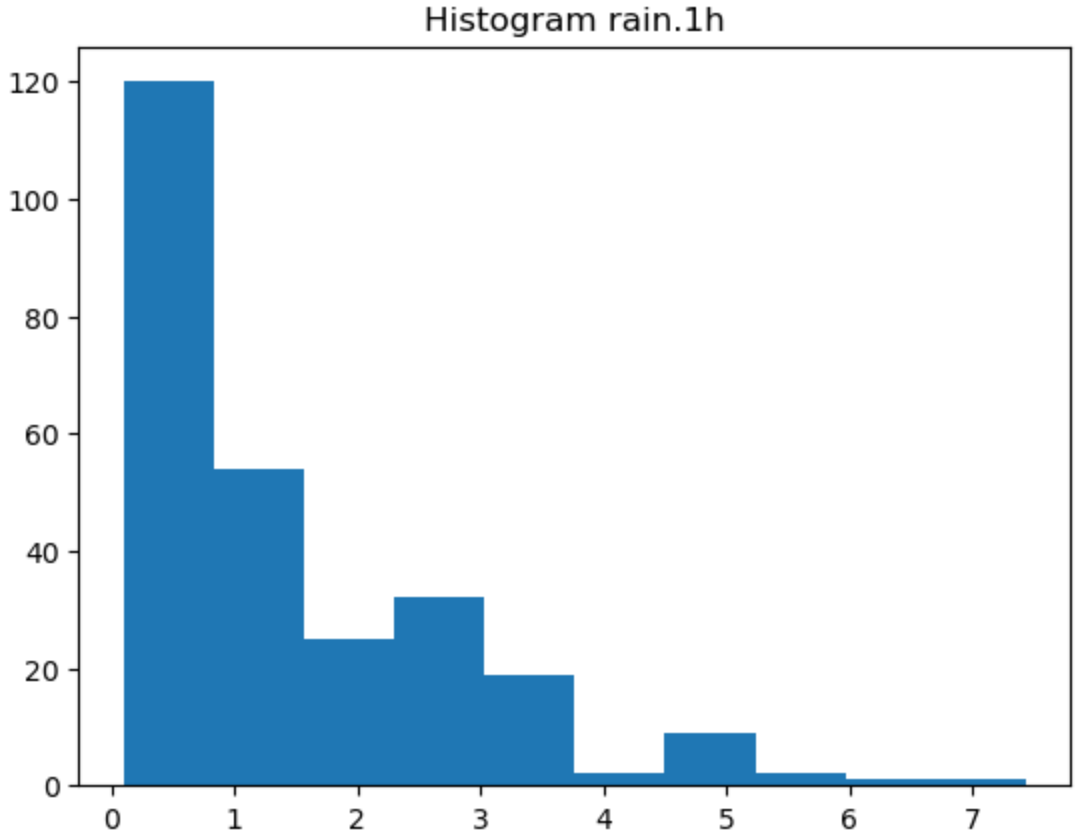

Histogram Temp Feels like
3000
2500
2000
1500
1000
500
0
0
5
10
15
20
25
30

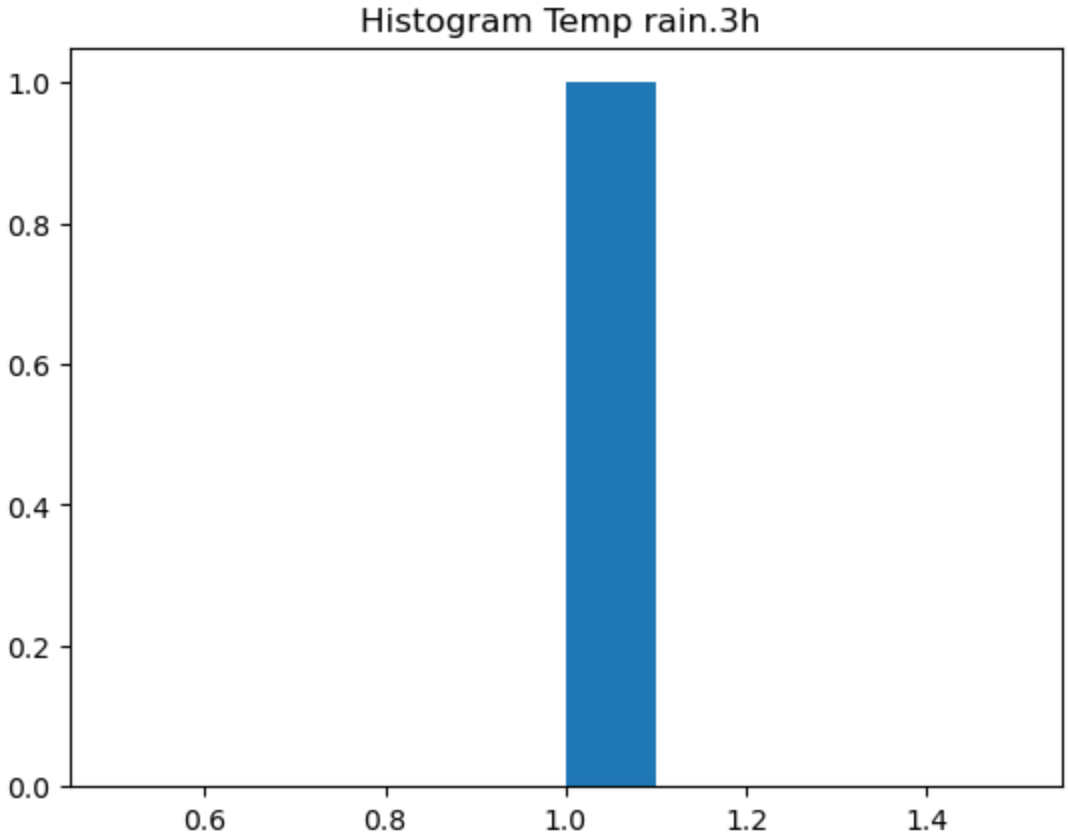
In [19]:
```
# Find missing values

dfALL.isnull().sum()

# Below you see ist of missing values pero variable
```

Out[19]:
```
dt                   0
weather              0
main.temp            0
main.feels_like      0
main.pressure        0
main.humidity        0
main.temp_min        0
main.temp_max        0
wind.speed           0
wind.deg             0
clouds.all           0
wind.gust         3935
rain.1h           8471
rain.3h           8735
datetime             0
Date                 0
Year                 0
Month                0
Day                  0
Hour                 0
Week_of_Year         0
dtype: int64
```

In [21]:
```
# histogram of varaibles of interes, the ones with missing values
# Adding various charts to one output

plt.hist(dfALL['wind.gust'])
plt.title("Histogram wind.gust")

# Adding the legends
plt.show()

# This function helps to add other chart

plt.figure()

plt.hist(dfALL['rain.1h'])
plt.title("Histogram rain.1h")

# Adding the legends
plt.show()

plt.figure()

plt.hist(dfALL['rain.3h'])
plt.title("Histogram Temp rain.3h ")

# Adding the legends
plt.show()
```




Histogram rain.1h
120
100
80
60
40
20
0
0
1
2
3
4
5
6
7

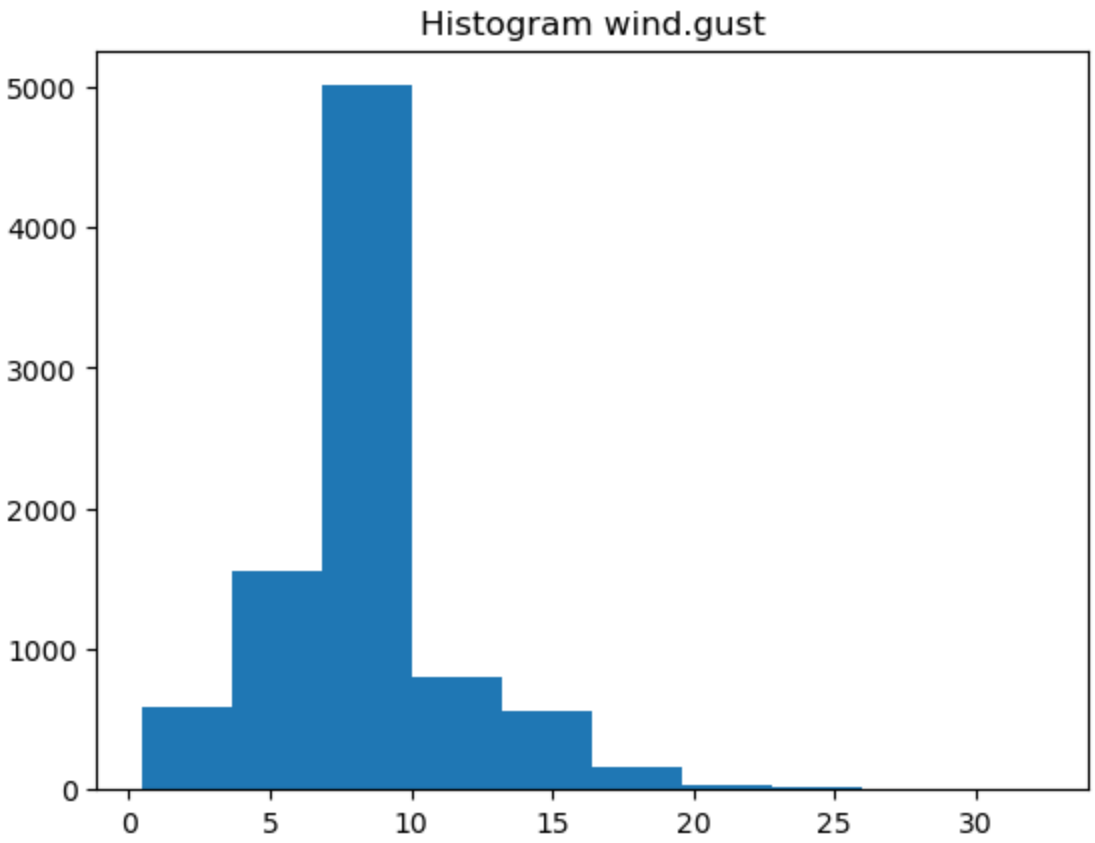

Histogram Temp rain.3h
1.0
0.8
0.6
0.4
0.2
0.0
0.6
0.8
1.0
1.2
1.4

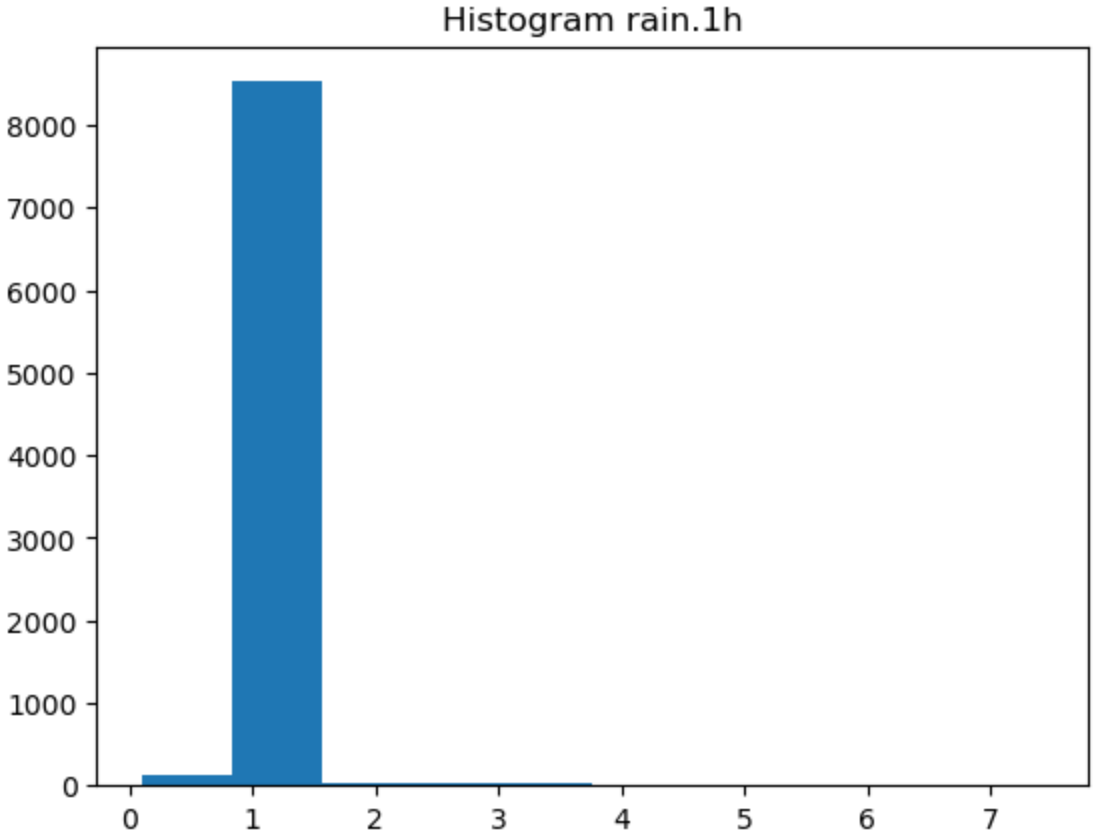
In [23]:
```
# Value inputation with meadian

# find mean of value

#mean_value = dfALL['wind.gust'].mean()

# Mean & MEdian for Wind Gust

mean = dfALL['wind.gust'].mean()
median = dfALL['wind.gust'].median()

print('Wind Gust Mean :'+str(mean))
print('Wind Gust Median :'+str(median))

# IMPUTATION: Replace NaN witn mean and median
# Mean of values in the same column

# Because mean and median for Gust are similar to exercise I am going to fill
dfALL['wind.gust'].fillna(value=dfALL['wind.gust'].mean(), inplace=True)

print('Wind Gust Mean is impute value :'+str(mean))
```

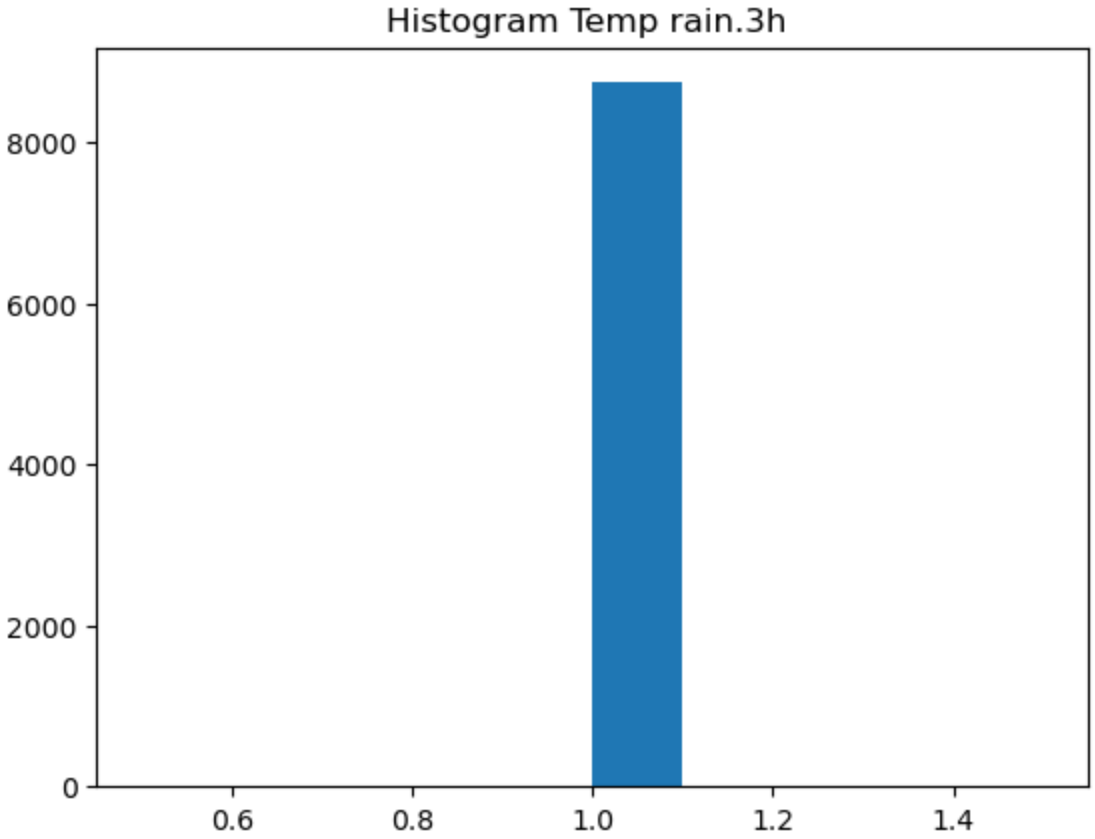
```
Wind Gust Mean :8.425038533638824
Wind Gust Median :7.6
Wind Gust Mean is impute value :8.425038533638824
```

In [24]:
```
# imoputation of Rain 1 and Rain 3
# Rain 1 has 97% of Nan and Rain 3 (100%), so this a is pactical way and emty
# just for exercising inpunt I am goin to impute median in Rain 1
# and then i am goin to impute Rain 1 meandia into Rain 3 missing values


# IMPUTE Rain 1 missing values with Rain 1 Median
dfALL['rain.1h'].fillna(value=dfALL['rain.1h'].median(), inplace=True)

# IMPUTE Rain 3 missing values with Rain 1 Median

dfALL['rain.3h'].fillna(value=dfALL['rain.1h'].median(), inplace=True)
```

In [25]:
```
# histogram of varaibles of interes, the ones with missing values
# Adding various charts to one output

plt.hist(dfALL['wind.gust'])
plt.title("Histogram wind.gust")

# Adding the legends
plt.show()

# This function helps to add other chart

plt.figure()

plt.hist(dfALL['rain.1h'])
plt.title("Histogram rain.1h")

# Adding the legends
plt.show()

plt.figure()

plt.hist(dfALL['rain.3h'])
plt.title("Histogram Temp rain.3h ")

# Adding the legends
plt.show()
```




Histogram rain.1h
8000
7000
6000
5000
4000
3000
2000
1000
0
0
1
2
3
4
5
6
7


Histogram Temp rain.3h
8000
6000
4000
2000
0
0.6
0.8
1.0
1.2
1.4

In [26]:
```
# Now lets check that there are not missing values to verify there are none

dfALL.isnull().sum()
```

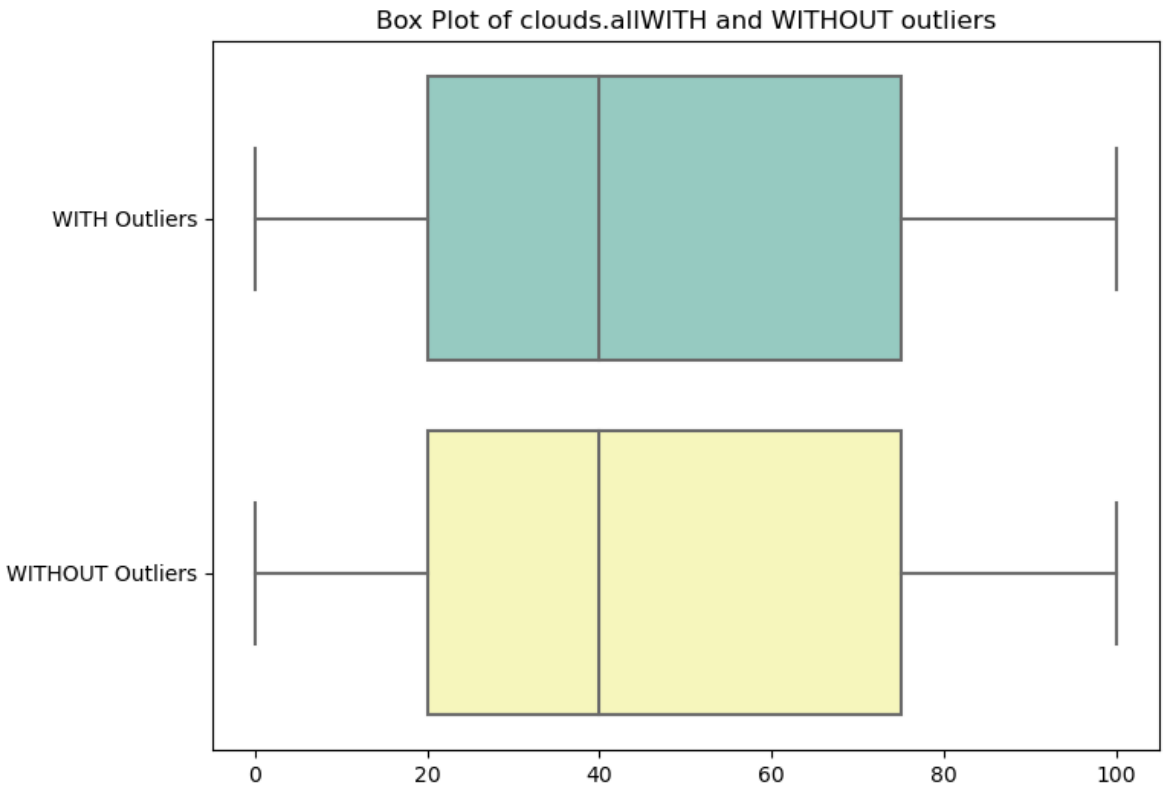
Out[26]:
```
dt                0
weather           0
main.temp         0
main.feels_like   0
main.pressure     0
main.humidity     0
main.temp_min     0
main.temp_max     0
wind.speed        0
wind.deg          0
clouds.all        0
wind.gust         0
rain.1h           0
rain.3h           0
datetime          0
Date              0
Year              0
Month             0
Day               0
Hour              0
Week_of_Year      0
dtype: int64
```

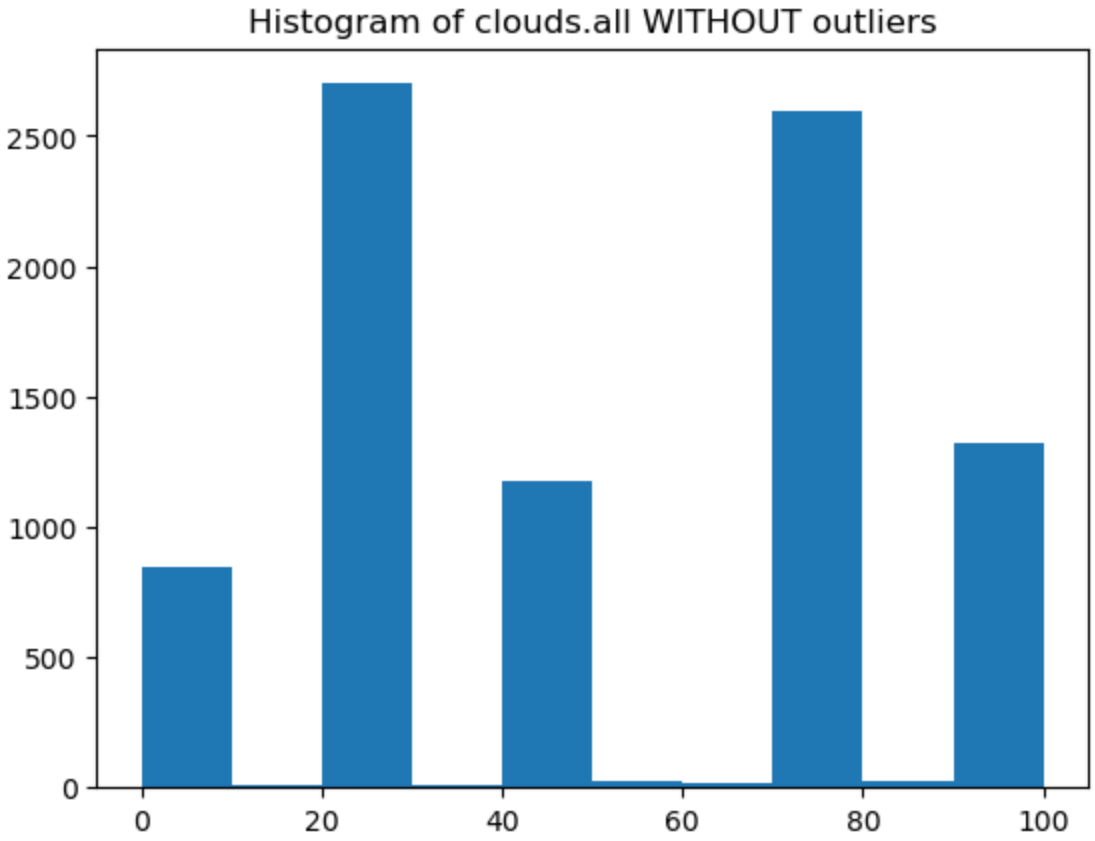
In [27]:
```
# Detect ouliers
# Importing

import sklearn
import seaborn as sns
```

In [37]:
```
# IQR outliers detection
# Calculate the upper and lower limits, each varible can put as 'Column' so to
dfOUT = dfALL

#varx = 'main.temp'
#varx = 'main.feels_like'
#varx = 'main.temp_min'
#varx = 'main.temp_max'
#varx = 'wind.speed'
#varx = 'wind.deg'
#varx = 'wind.gust'
varx = 'clouds.all'
#varx = 'rain.1h'
#varx = 'rain.3h'

# convertir la valirible a analiar en un arrenge
arr1 = dfALL[varx]


# finding the 1st quartile
q1 = np.quantile(dfALL[varx], 0.25)

# finding the 3rd quartile
q3 = np.quantile(dfALL[varx], 0.75)
med = np.median(dfALL[varx])

# finding the iqr region
iqr = q3-q1

# finding upper and lower whiskers
upper_bound = q3+(1.5*iqr)
lower_bound = q1-(1.5*iqr)

print(varx+' IQR is: '+str(round(iqr,2)))
print(varx+' Upper bound is: '+str(round(upper_bound,2)))
print(varx+' Lower bound is: '+str(round(lower_bound,2)))


print('Name is: '+str(arr1.name))

outliers = arr1[(arr1 <= lower_bound) | (arr1 >= upper_bound)]
#print('The following are the outliers in the boxplot:{}'.format(outliers))
print('Total Outliers:' +str(len(outliers)))

# Filtrar los outliers en 2 pasos
arr2 = arr1[(arr1 >= lower_bound) & (arr1 <= upper_bound)]

# Box plot

data = pd.DataFrame({
    'WITH Outliers': arr1,
    'WITHOUT Outliers': arr2
})

# Create a box plot
plt.figure(figsize=(8, 6))
sns.boxplot(data=data, palette="Set3", orient='h')
```

```
plt.title('Box Plot of '+varx+'WITH and WITHOUT outliers')
plt.show()


# Histogram of chart WITHOUT outliers
plt.figure()
plt.hist(arr2)
plt.title("Histogram of "+varx+" WITHOUT outliers")

# Histogram of chart OUTLIERS outliers

# Total of OULIERS

#plt.figure()
#plt.hist(outliers)
#plt.title("OUTLIERS Histogram of "+varx+" OUTLIERS")

plt.show()
```

```
clouds.all IQR is: 55.0
clouds.all Upper bound is: 157.5
clouds.all Lower bound is: -62.5
Name is: clouds.all
Total Outliers:0
```





In [ ]:
```
# DEVELOPMENT TEST
# loop of weeks to extract
count = 0
WeekStamp = StartDate
while (WeekStamp < NowTStmap):
    count = count + 1
    WeekDate = datetime.utcfromtimestamp(WeekStamp)
    IntialDateView = WeekDate.strftime("%Y-%m-%d %H:%M:%S")
    print(f" Week:  {count}"+ f" WeekDate {WeekDate}"  )
    WeekStamp = WeekStamp + 604800
```

In [ ]:
```
# DEV TEST Extract for jason list inorder and normalizate data
pd.json_normalize(dataAPI['list'])
df1 = pd.json_normalize(dataAPI['list'],max_level=3)
df1
```

In [ ]:
```
# DEVELOPMENT TES: Add dataframes

# Add data frames test

r1 = requests.get(url = full_url)
dataAPI = r1.json()
# print(dataAPI)
df1 = pd.json_normalize(dataAPI['list'],max_level=3)
df1 = pd.DataFrame(df1)
# df1

r2 = requests.get(url = url)
ataURL = r2.json()
# print(dataURL)
df2 = pd.json_normalize(dataURL['list'],max_level=3)
df2 = pd.DataFrame(df2)
# df2

pd.concat([df1, df2], ignore_index=True, axis=0)
```

In [ ]:
```
# DEV TEST Prepare data with month and week for grpahs.

# Ininitialize data set
dfdate = pd.DataFrame()

dfdate['datetime'] = pd.to_datetime(dfALL['dt'], unit='s')
dfdate['Year'] = dfdate['datetime'].dt.year
dfdate['Month'] = dfdate['datetime'].dt.month
print(dfdate)
```

In [ ]:
```
# DEV TEST of link working. Initial Shows full_url varaible works to get info
r = requests.get(url = full_url)
dataAPI = r.json()
print(dataAPI)
```

In [ ]:
```
# Box plot of clouds
plt.title("Box plot clouds.all")
plt.boxplot(dfALL['clouds.all'])
plt.ylabel('Clouds %')

plt.show()
```