Working on the weather dataset using python

About the dataset

The name of the dataset is ‘Weather Data’. This dataset contains information about the weather information per hour of a certain location .In this dataset the table contains the date & time, temperature, dew point temperature, relative humidity, pressure, visibility, wind speed and weather type of each hour .The data is imported from kaggle .The dataset is in .csv format.

Importing the library

First we will import all useful libraries of python

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import datetime
```

Now to import the data set we are using read_csv() function under which we are providing the location of our data. You can also upload the data set in the folder in which you are having your source code file , in this case you just have to write the name to your data file(It is only for the jupyter notebook).

In [3]:
```
# The name of the data set is 1.Weather Data with extension .csv
data = pd.read_csv('C:/Users/abc/Downloads/1. Weather Data.csv')
```

In [18]:
```
 # Here data will print the first and last 5 rows of the dataset
data
```

Out[18]:

	Date/Time	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa	Weather
0	1/1/2012 0:00	-1.8	-3.9	86	4	8.0	101.24	Fog
1	1/1/2012 1:00	-1.8	-3.7	87	4	8.0	101.24	Fog
2	1/1/2012 2:00	-1.8	-3.4	89	7	4.0	101.26	Freezing Drizzle,Fog
3	1/1/2012 3:00	-1.5	-3.2	88	6	4.0	101.27	Freezing Drizzle,Fog
4	1/1/2012 4:00	-1.5	-3.3	88	7	4.8	101.23	Fog
...	...	...	...	...	...	...	...	...
8779	12/31/2012 19:00	0.1	-2.7	81	30	9.7	100.13	Snow
8780	12/31/2012 20:00	0.2	-2.4	83	24	9.7	100.03	Snow
8781	12/31/2012 21:00	-0.5	-1.5	93	28	4.8	99.95	Snow
8782	12/31/2012 22:00	-0.2	-1.8	89	28	9.7	99.91	Snow
8783	12/31/2012	0.0	-2.1	86	30	11.3	99.89	Snow

23:00

8784 rows × 8 columns

so as we can see that our data has 8784 rows and 8 columns.

Exploring the dataset

For exploring the dataset we will use many functions present in the libraries of the python

1.Using the head() function : The head() will print the first 5 rows of the dataset

In [19]:
```
data.head()
```

Out[19]:

	Date/Time	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa	Weather
0	1/1/2012 0:00	-1.8	-3.9	86	4	8.0	101.24	Fog
1	1/1/2012 1:00	-1.8	-3.7	87	4	8.0	101.24	Fog
2	1/1/2012 2:00	-1.8	-3.4	89	7	4.0	101.26	Freezing Drizzle,Fog
3	1/1/2012 3:00	-1.5	-3.2	88	6	4.0	101.27	Freezing Drizzle,Fog
4	1/1/2012 4:00	-1.5	-3.3	88	7	4.8	101.23	Fog

2. Using the tail(): The tail() function will print the last 5 rows of the dataset.

In [20]:
```
data.tail()
```

Out[20]:

	Date/Time	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa	Weather
8779	12/31/2012 19:00	0.1	-2.7	81	30	9.7	100.13	Snow
8780	12/31/2012 20:00	0.2	-2.4	83	24	9.7	100.03	Snow
8781	12/31/2012 21:00	-0.5	-1.5	93	28	4.8	99.95	Snow
8782	12/31/2012 22:00	-0.2	-1.8	89	28	9.7	99.91	Snow
8783	12/31/2012 23:00	0.0	-2.1	86	30	11.3	99.89	Snow

3.Using the shape: The shape function will give us the no. of rows and no. of columns in tuple.

In [22]:
```
data.shape
```

Out[22]:
```
(8784, 8)
```

Result : We can see that we get (8784,8) as the output it means that there are 8784 rows and 8 columns in the dataset

4.Using the size : The size will give us total no. of entries in the dataset that is nothing but (no.of rows * no. of columns)

In [23]:
```
data.size
```

Out[23]:
```
70272
```

Result : The output is 70272 , it means that there are 70272 values in the dataset.

5.Using the columns: The columns will give the name of all the columns present in the dataset.

NOTE: We are using columns not column

In [26]:
```
data.columns
```

Out[26]:
```
Index(['Date/Time', 'Temp_C', 'Dew Point Temp_C', 'Rel Hum_%',
       'Wind Speed_km/h', 'Visibility_km', 'Press_kPa', 'Weather'],
      dtype='object')
```

Result: Here we can see that we get an array of all columns present in the dataset.

6.Using dtypes: It will show the datatype of all the columns.

NOTE: use dtypes not dtype .

In [28]:
```
data.dtypes
```

Out[28]:
```
Date/Time            object
Temp_C              float64
Dew Point Temp_C    float64
Rel Hum_%             int64
Wind Speed_km/h       int64
Visibility_km       float64
Press_kPa           float64
Weather              object
dtype: object
```

Result : Above we can see that we get the datatype of all columns present in the dataset.

So as we can see that in the above result we are having Date/Time in object type which can create problem in the analysis so we have to convert it's datatype in datetime type, for this we wil write the following code:

In [17]:
```
data['Date/Time']=data['Date/Time'].astype('datetime64[ns]')
data.dtypes
```

Out[17]:
```
Date/Time           datetime64[ns]
Temp_C                     float64
Dew Point Temp_C           float64
Rel Hum_%                    int64
Wind Speed_km/h              int64
Visibility_km              float64
Press_kPa                  float64
Weather                     object
dtype: object
```

Now we can see that the datatype of the Date/Time is changed to datetime64[ns].

In [55]:
```
data.head()
```

Out[55]:

	Date/Time	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa	Weather

0	2012-01-01 00:00:00	-1.8	-3.9	86	4	8.0	101.24	Fog
1	2012-01-01 01:00:00	-1.8	-3.7	87	4	8.0	101.24	Fog
2	2012-01-01 02:00:00	-1.8	-3.4	89	7	4.0	101.26	Freezing Drizzle,Fog
3	2012-01-01 03:00:00	-1.5	-3.2	88	6	4.0	101.27	Freezing Drizzle,Fog
4	2012-01-01 04:00:00	-1.5	-3.3	88	7	4.8	101.23	Fog

7.Using the unique(): It will give the all the unique value present in a particular column present in the dataset.

So here we want to know the unique values present in the weather column

In [30]:
```
data['Weather'].unique()
```

Out[30]:
```
array(['Fog', 'Freezing Drizzle,Fog', 'Mostly Cloudy', 'Cloudy', 'Rain',
       'Rain Showers', 'Mainly Clear', 'Snow Showers', 'Snow', 'Clear',
       'Freezing Rain,Fog', 'Freezing Rain', 'Freezing Drizzle',
       'Rain,Snow', 'Moderate Snow', 'Freezing Drizzle,Snow',
       'Freezing Rain,Snow Grains', 'Snow,Blowing Snow', 'Freezing Fog',
       'Haze', 'Rain,Fog', 'Drizzle,Fog', 'Drizzle',
       'Freezing Drizzle,Haze', 'Freezing Rain,Haze', 'Snow,Haze',
       'Snow,Fog', 'Snow,Ice Pellets', 'Rain,Haze', 'Thunderstorms,Rain',
       'Thunderstorms,Rain Showers', 'Thunderstorms,Heavy Rain Showers',
       'Thunderstorms,Rain Showers,Fog', 'Thunderstorms',
       'Thunderstorms,Rain,Fog',
       'Thunderstorms,Moderate Rain Showers,Fog', 'Rain Showers,Fog',
       'Rain Showers,Snow Showers', 'Snow Pellets', 'Rain,Snow,Fog',
       'Moderate Rain,Fog', 'Freezing Rain,Ice Pellets,Fog',
       'Drizzle,Ice Pellets,Fog', 'Drizzle,Snow', 'Rain,Ice Pellets',
       'Drizzle,Snow,Fog', 'Rain,Snow Grains', 'Rain,Snow,Ice Pellets',
       'Snow Showers,Fog', 'Moderate Snow,Blowing Snow'], dtype=object)
```

Result: As we can see that we got all unique weather condition present in the weather column.

8. Using nunique(): It will count the all unique values present in each columns of the dataset.

In [31]:
```
data.nunique()
```

Out[31]:
```
Date/Time           8784
Temp_C               533
Dew Point Temp_C     489
Rel Hum_%             83
Wind Speed_km/h       34
Visibility_km         24
Press_kPa            518
Weather               50
dtype: int64
```

Result: So we got the list of all the columns and how many unique values they have. for e.g. Temp_C has 533 unique values of temperature in it.

9.Using count(): It will count the non-null values in each columns.

In [32]:
```
data.count()
```

Out[32]:
```
Date/Time           8784
Temp_C              8784
```

```
Dew Point Temp_C    8784
Rel Hum_%           8784
Wind Speed_km/h     8784
Visibility_km       8784
Press_kPa           8784
Weather             8784
dtype: int64
```

Here we can see that all the columns are having 8784 non-null values it means that there is no missing value in any column of the dataset. We can also check the count of the null value by using isnull() function.

10.Using isnull().sum(): It will count the null values present in each columns of the dataset.

In [34]:
```
data.isnull().sum()
```

Out[34]:
```
Date/Time           0
Temp_C              0
Dew Point Temp_C    0
Rel Hum_%           0
Wind Speed_km/h     0
Visibility_km       0
Press_kPa           0
Weather             0
dtype: int64
```

Result: Here from the output we can say that there is not any null value present in the dataset.

11.Using value_counts(): It is applied on a single column and returns the count of each unique values of that column.

NOTE: Notice that there is 's' in count means it is counts not count.

In [5]:
```
data.value_counts()
# so here you can see that if we apply the value_counts() on whole dataset then it will
# and such info is not that much useful for anlaysis so we will apply the value_counts()
```

Out[5]:
```
Date/Time        Temp_C  Dew Point Temp_C  Rel Hum_%  Wind Speed_km/h  Visibility_km  Pr
ess_kPa  Weather
1/1/2012 0:00    -1.8    -3.9              86         4                8.0            10
1.24     Fog              1
6/1/2012 12:00    19.3     3.3              35         20               48.3           10
1.32     Cloudy           1
5/9/2012 7:00     14.3     12.5             89         15               4.8            10
0.12     Fog              1
5/9/2012 8:00     14.3     12.3             88         17               6.4            10
0.12     Fog              1
5/9/2012 9:00     14.0     12.3             89         9                4.0            10
0.10     Drizzle,Fog      1

                          ..
12/8/2012 3:00    2.1     -1.5             77         6                25.0           10
1.18     Cloudy           1
12/8/2012 2:00    2.0     -1.9             75         7                25.0           10
1.17     Cloudy           1
12/8/2012 23:00   1.3      0.6             95         17               8.0            10
0.96     Drizzle,Fog      1
12/8/2012 22:00   1.2      0.6             96         13               6.4            10
0.84     Fog              1
9/9/2012 9:00     14.8     8.8             67         17               48.3           10
0.65     Mainly Clear     1
Length: 8784, dtype: int64
```

In [6]:
```
data.Weather.value_counts()
```

Out[6]:
```
Mainly Clear                                2106
Mostly Cloudy                               2069
Cloudy                                      1728
Clear                                       1326
Snow                                         390
Rain                                         306
Rain Showers                                 188
Fog                                          150
Rain,Fog                                     116
Drizzle,Fog                                   80
Snow Showers                                  60
Drizzle                                       41
Snow,Fog                                      37
Snow,Blowing Snow                             19
Rain,Snow                                     18
Thunderstorms,Rain Showers                    16
Haze                                          16
Drizzle,Snow,Fog                              15
Freezing Rain                                 14
Freezing Drizzle,Snow                         11
Freezing Drizzle                               7
Snow,Ice Pellets                               6
Freezing Drizzle,Fog                           6
Snow,Haze                                      5
Freezing Fog                                   4
Snow Showers,Fog                               4
Moderate Snow                                  4
Rain,Snow,Ice Pellets                          4
Freezing Rain,Fog                              4
Freezing Drizzle,Haze                          3
Rain,Haze                                      3
Thunderstorms,Rain                             3
Thunderstorms,Rain Showers,Fog                 3
Freezing Rain,Haze                             2
Drizzle,Snow                                   2
Rain Showers,Snow Showers                      2
Thunderstorms                                  2
Moderate Snow,Blowing Snow                     2
Rain Showers,Fog                               1
Thunderstorms,Moderate Rain Showers,Fog        1
Snow Pellets                                   1
Rain,Snow,Fog                                  1
Moderate Rain,Fog                              1
Freezing Rain,Ice Pellets,Fog                  1
Drizzle,Ice Pellets,Fog                        1
Thunderstorms,Rain,Fog                         1
Rain,Ice Pellets                               1
Rain,Snow Grains                               1
Thunderstorms,Heavy Rain Showers               1
Freezing Rain,Snow Grains                      1
Name: Weather, dtype: int64
```

Result : As in the dataset we have the record per hour, so from the above result we can say that there were 2106 hours when weather was mostly clear.

12.Using info() : It is very useful function ,it will return the columns name along with the count of the non-null values in it and the datatype of the column

In [40]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8784 entries, 0 to 8783
Data columns (total 8 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
```

```
 0   Date/Time          8784 non-null   object
 1   Temp_C             8784 non-null   float64
 2   Dew Point Temp_C   8784 non-null   float64
 3   Rel Hum_%          8784 non-null   int64
 4   Wind Speed_km/h    8784 non-null   int64
 5   Visibility_km      8784 non-null   float64
 6   Press_kPa          8784 non-null   float64
 7   Weather            8784 non-null   object
dtypes: float64(4), int64(2), object(2)
memory usage: 549.1+ KB
```

Result: We get alot of the information about the dataset by using info() function for e.g. the datatype of data is dataframe ,rangeindex and the no. of the columns in the data as well as the name of all columns and the count of the non-null values in it and it's datatype

13. Using describe() : It is also very helpful function to analyse the data.It will give a short statistical summary of each column having numeric values in it .It returns the count,min,max,standard deviation,mean and quartiles of each columns having numeric value

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

Out[7]:

	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa
count	8784.000000	8784.000000	8784.000000	8784.000000	8784.000000	8784.000000
mean	8.798144	2.555294	67.431694	14.945469	27.664447	101.051623
std	11.687883	10.883072	16.918881	8.688696	12.622688	0.844005
min	-23.300000	-28.500000	18.000000	0.000000	0.200000	97.520000
25%	0.100000	-5.900000	56.000000	9.000000	24.100000	100.560000
50%	9.300000	3.300000	68.000000	13.000000	25.000000	101.070000
75%	18.800000	11.800000	81.000000	20.000000	25.000000	101.590000
max	33.000000	24.400000	100.000000	83.000000	48.300000	103.650000

Result : From the output we can tell the count,mean,max,min,standard deviation and all quartile of each columns present in the dataset.Note here that this function don't touch the the columns having categorical data

14. Using duplicated(): This function is useful to find the duplicate values in the dataset.

In [47]:
```
dup=data.duplicated()   # data.duplicated() we return the result in boolean type means i
data[dup].sum()  # so to print the no. of duplicated value we put the dup in data, it wi
# to count the no. of duplicate we apply sum() function on it
```

Out[47]:
```
Date/Time           0.0
Temp_C              0.0
Dew Point Temp_C    0.0
Rel Hum_%           0.0
Wind Speed_km/h     0.0
Visibility_km       0.0
Press_kPa           0.0
Weather             0.0
dtype: float64
```

Result :From the output we can say there is not any duplicate values in the dataset because each column is having 0 count.

While exploring the dataset we have cleaned our dataset that means our dataset is now ready to do the analysis nd we also get familiar with the dataset .So now we are ready to use our dataset to fetch the

answers of our questions.

Extracting the info from the dataset :

Now we will find the answer of some of the questions related to the dataset.

1.Find all the unique value of wind speed in the data.

To answer this question we are using unique() and nunique() function . The unique() function will give us an array having all the unique values present in the wind speed column of the dataset. The code is following:

In [59]:
```
data['Wind Speed_km/h'].unique()
```

Out[59]:
```
array([ 4,  7,  6,  9, 15, 13, 20, 22, 19, 24, 30, 35, 39, 32, 33, 26, 44,
       43, 48, 37, 28, 17, 11,  0, 83, 70, 57, 46, 41, 52, 50, 63, 54,  2],
      dtype=int64)
```

Now we use the nunique() function that will give us the total number of the unique values present in the wind speed column. The code is following:

In [60]:
```
data['Wind Speed_km/h'].nunique()
```

Out[60]:
```
34
```

So there are 34 unique values present in the wind speed columns.

2. Find no. of times when weather was exactly clear.

To answer this question we are using the concept of group by present in pandas .By using the groupby() function over the weather we will be able to make groups of the rows based on their weather type and to get the rows of a particular weather which is "Clear" in this case we will use the get_group() function whose argument will be that particular weather which you want, so first we will make the group of the rows based on the weather using groupby('Weather') then we will select the rows having "Clear" weather using get_group('Clear').The code is following:

In [67]:
```
data1=data.groupby('Weather')
data1.get_group('Clear')
```

Out[67]:

	Date/Time	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa	Weather
67	2012-01-03 19:00:00	-16.9	-24.8	50	24	25.0	101.74	Clear
114	2012-01-05 18:00:00	-7.1	-14.4	56	11	25.0	100.71	Clear
115	2012-01-05 19:00:00	-9.2	-15.4	61	7	25.0	100.80	Clear
116	2012-01-05 20:00:00	-9.8	-15.7	62	9	25.0	100.83	Clear
117	2012-01-05 21:00:00	-9.0	-14.8	63	13	25.0	100.83	Clear
...	...	...	...	...	...	...	...	...

8646	2012-12-26 06:00:00	-13.4	-14.8	89	4	25.0	102.47	Clear
8698	2012-12-28 10:00:00	-6.1	-8.6	82	19	24.1	101.27	Clear
8713	2012-12-29 01:00:00	-11.9	-13.6	87	11	25.0	101.31	Clear
8714	2012-12-29 02:00:00	-11.8	-13.1	90	13	25.0	101.33	Clear
8756	2012-12-30 20:00:00	-13.8	-16.5	80	24	25.0	101.52	Clear

1326 rows × 8 columns

so here we can see that total 1326 rows are having clear weather.

we can also answer this question by using value_counts() function , the code is following:

In [68]:
```
data.Weather.value_counts()
```

Out[68]:
```
Mainly Clear                                2106
Mostly Cloudy                               2069
Cloudy                                      1728
Clear                                       1326
Snow                                         390
Rain                                         306
Rain Showers                                 188
Fog                                          150
Rain,Fog                                     116
Drizzle,Fog                                   80
Snow Showers                                  60
Drizzle                                       41
Snow,Fog                                      37
Snow,Blowing Snow                             19
Rain,Snow                                     18
Thunderstorms,Rain Showers                    16
Haze                                          16
Drizzle,Snow,Fog                              15
Freezing Rain                                 14
Freezing Drizzle,Snow                         11
Freezing Drizzle                               7
Snow,Ice Pellets                               6
Freezing Drizzle,Fog                           6
Snow,Haze                                      5
Freezing Fog                                   4
Snow Showers,Fog                               4
Moderate Snow                                  4
Rain,Snow,Ice Pellets                          4
Freezing Rain,Fog                              4
Freezing Drizzle,Haze                          3
Rain,Haze                                      3
Thunderstorms,Rain                             3
Thunderstorms,Rain Showers,Fog                 3
Freezing Rain,Haze                             2
Drizzle,Snow                                   2
Rain Showers,Snow Showers                      2
Thunderstorms                                  2
Moderate Snow,Blowing Snow                     2
Rain Showers,Fog                               1
Thunderstorms,Moderate Rain Showers,Fog        1
Snow Pellets                                   1
Rain,Snow,Fog                                  1
```

```
Moderate Rain,Fog                          1
Freezing Rain,Ice Pellets,Fog              1
Drizzle,Ice Pellets,Fog                    1
Thunderstorms,Rain,Fog                     1
Rain,Ice Pellets                           1
Rain,Snow Grains                           1
Thunderstorms,Heavy Rain Showers           1
Freezing Rain,Snow Grains                  1
Name: Weather, dtype: int64
```

so here is the list of all the weathers and their frequency , according to the above list we can see that clear weather occured 1326 times in the dataset.

3. Find the number of times when wind speed was exactly "4Km/h" .

So let's use the value_counts() to answer this question

In [73]:
```
data['Wind Speed_km/h'].value_counts()
```

Out[73]:
```
9     830
11    791
13    735
15    719
7     677
17    666
19    616
6     609
20    496
4     474
22    439
24    374
0     309
26    242
28    205
30    161
32    139
33     85
35     53
37     45
39     24
41     22
44     14
43     13
48     13
46     11
52      7
57      5
50      4
2       2
83      1
70      1
63      1
54      1
Name: Wind Speed_km/h, dtype: int64
```

As we can see that this is giving the table of the all the wind speed with their frequency , acoording to this we can say that 474 times the wind speed was 4km/h, but it may be time taking to find a certain value in the above table so to tackel this problem we will use group by function , the code is following :

In [77]:
```
data.groupby('Wind Speed_km/h').get_group(4)
```

Out[77]:

	Date/Time	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa	Weather
0	2012-01-01 00:00:00	-1.8	-3.9	86	4	8.0	101.24	Fog
1	2012-01-01 01:00:00	-1.8	-3.7	87	4	8.0	101.24	Fog
96	2012-01-05 00:00:00	-8.8	-11.7	79	4	9.7	100.32	Snow
101	2012-01-05 05:00:00	-7.0	-9.5	82	4	4.0	100.19	Snow
146	2012-01-07 02:00:00	-8.1	-11.1	79	4	19.3	100.15	Cloudy
...	...	...	...	...	...	...	...	...
8768	2012-12-31 08:00:00	-8.6	-10.3	87	4	3.2	101.14	Snow Showers
8769	2012-12-31 09:00:00	-8.1	-9.6	89	4	2.4	101.09	Snow
8770	2012-12-31 10:00:00	-7.4	-8.9	89	4	6.4	101.05	Snow,Fog
8772	2012-12-31 12:00:00	-5.8	-7.5	88	4	12.9	100.78	Snow
8773	2012-12-31 13:00:00	-4.6	-6.6	86	4	12.9	100.63	Snow

474 rows × 8 columns

so now we can easily see that for 474 times the speed of wind was 4km/h.

here is also another method that is following:

In [80]:
```
data[data['Wind Speed_km/h']==4]     # from this we also get the same result as from the
```

Out[80]:

	Date/Time	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa	Weather
0	2012-01-01 00:00:00	-1.8	-3.9	86	4	8.0	101.24	Fog
1	2012-01-01 01:00:00	-1.8	-3.7	87	4	8.0	101.24	Fog
96	2012-01-05 00:00:00	-8.8	-11.7	79	4	9.7	100.32	Snow
101	2012-01-05 05:00:00	-7.0	-9.5	82	4	4.0	100.19	Snow
146	2012-01-07 02:00:00	-8.1	-11.1	79	4	19.3	100.15	Cloudy
...	...	...	...	...	...	...	...	...
8768	2012-12-31 08:00:00	-8.6	-10.3	87	4	3.2	101.14	Snow Showers
8769	2012-12-31 09:00:00	-8.1	-9.6	89	4	2.4	101.09	Snow
8770	2012-12-31 10:00:00	-7.4	-8.9	89	4	6.4	101.05	Snow,Fog
8772	2012-12-31	-5.8	-7.5	88	4	12.9	100.78	Snow

	12:00:00							
8773	2012-12-31 13:00:00	-4.6	-6.6	86	4	12.9	100.63	Snow

474 rows × 8 columns

4. Rename the column 'weather' to 'weather condition'.

To rename the column "Weather" we will use the following code :

In [81]:
```
data.rename(columns={"Weather":"Weather_Condition"},inplace=True)
```

The first argument of the rename is columns in which we have the old name : the new name that we want ,the second argument is the inplace=True it will make the changes permanently.

In [82]:
```
data.head(2)
```

Out[82]:

	Date/Time	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa	Weather_Condition
0	2012-01-01 00:00:00	-1.8	-3.9	86	4	8.0	101.24	Fog
1	2012-01-01 01:00:00	-1.8	-3.7	87	4	8.0	101.24	Fog

so here we see that our column name has changed.

5. What is mean of the visibility ?

We can answer this question by 2 methods.

Method 1: In this we will simply apply the mean function on the column "Visibility_km".The code is following:

In [85]:
```
print("The mean of the visibility is :")
data.Visibility_km.mean()
```

Out[85]:
```
The mean of the visibility is :
27.664446721311478
```

Method 2: By using the describe() function we can also access the mean of the visibility column.

In [88]:
```
data.describe()
```

Out[88]:

	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa
count	8784.000000	8784.000000	8784.000000	8784.000000	8784.000000	8784.000000
mean	8.798144	2.555294	67.431694	14.945469	27.664447	101.051623
std	11.687883	10.883072	16.918881	8.688696	12.622688	0.844005
min	-23.300000	-28.500000	18.000000	0.000000	0.200000	97.520000
25%	0.100000	-5.900000	56.000000	9.000000	24.100000	100.560000
50%	9.300000	3.300000	68.000000	13.000000	25.000000	101.070000
75%	18.800000	11.800000	81.000000	20.000000	25.000000	101.590000

max	33.000000	24.400000	100.000000	83.000000	48.300000	103.650000

As we can see that the describe() function is giving us a dataframe and our desired data is located at the (1,4) position in dataframe so by using the iloc[] we can access it .

In [5]:
```
print(data.describe().iloc[1,4])
```

```
27.664446721311478
```

In [6]:
```
# use the round() function to round the mean value
print(round(data.describe().iloc[1,4],2))
```

```
27.66
```

Result: The mean of the visibility is 27.66 km/h.

6. What is the standard deviation of the pressure column?

Here we use the std() function over the Press_kPa column to find the standard deviation of the pressure.

In [7]:
```
print("The standard deviation of the pressure is ",round(data.Press_kPa.std(),2),"kPa")
```

```
The standard deviation of the pressure is  0.84 kPa
```

Result: The standars deviation of the pressure is 0.84kPa

7.What is the variance of the "Relative Humidity"?

To find the variance of the relative humidity we use var() function over the column 'Rel Hum_%'.The code is following:

In [8]:
```
print("The variance of the relative humidity is ",round(data['Rel Hum_%'].var(),2))
```

```
The variance of the relative humidity is  286.25
```

NOTE: Here in the code we can see that this time we are using the [] for the column name inplace of dot .We are doing this because in the Rel Hum_% we have gap between Rel and Hum, if there will be no gaps in the name of the column then we can use the dot.

8.Find all the instance when "Snow" was recorded.

If we want to find those rows when only "snow" was the weather condition then we can use the groupby() function on "Weather_Condition" column . It will give us the the rows when weather condition was snow only. The code is following:

In [126…
```
data.groupby(['Weather_Condition']).get_group("Snow")
```

Out[126]:

	Date/Time	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa	Weather_Condition
55	2012-01-03 07:00:00	-14.0	-19.5	63	19	25.0	100.95	Snow
84	2012-01-04 12:00:00	-13.7	-21.7	51	11	24.1	101.25	Snow

86	2012-01-04 14:00:00	-11.3	-19.0	53	7	19.3	100.97	Snow
87	2012-01-04 15:00:00	-10.2	-16.3	61	11	9.7	100.89	Snow
88	2012-01-04 16:00:00	-9.4	-15.5	61	13	19.3	100.79	Snow
...	...	...	...	...	...	...	...	...
8779	2012-12-31 19:00:00	0.1	-2.7	81	30	9.7	100.13	Snow
8780	2012-12-31 20:00:00	0.2	-2.4	83	24	9.7	100.03	Snow
8781	2012-12-31 21:00:00	-0.5	-1.5	93	28	4.8	99.95	Snow
8782	2012-12-31 22:00:00	-0.2	-1.8	89	28	9.7	99.91	Snow
8783	2012-12-31 23:00:00	0.0	-2.1	86	30	11.3	99.89	Snow

390 rows × 8 columns

So here we can see that there are 390 rows having snow weather condition .

But during the exploration of the data we have noticed that there were some more weather conditions when snow occured. It means we have to count all those rows when snow present in the weather condition.

for this we use the str.contains() function in which we put the "Snow" as the input .So this function will include all those rows which have the string "Snow" in it and to print the rows we put it into data.

In [128…
```
data[data.Weather_Condition.str.contains("Snow")]
```

Out[128]:

	Date/Time	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa	Weather_Condition
41	2012-01-02 17:00:00	-2.1	-9.5	57	22	25.0	99.66	Snow Showers
44	2012-01-02 20:00:00	-5.6	-13.4	54	24	25.0	100.07	Snow Showers
45	2012-01-02 21:00:00	-5.8	-12.8	58	26	25.0	100.15	Snow Showers
47	2012-01-02 23:00:00	-7.4	-14.1	59	17	19.3	100.27	Snow Showers
48	2012-01-03 00:00:00	-9.0	-16.0	57	28	25.0	100.35	Snow Showers
...	...	...	...	...	...	...	...	...
8779	2012-12-31 19:00:00	0.1	-2.7	81	30	9.7	100.13	Snow
8780	2012-12-31 20:00:00	0.2	-2.4	83	24	9.7	100.03	Snow
8781	2012-12-31 21:00:00	-0.5	-1.5	93	28	4.8	99.95	Snow
8782	2012-12-31 22:00:00	-0.2	-1.8	89	28	9.7	99.91	Snow
8783	2012-12-31	0.0	-2.1	86	30	11.3	99.89	Snow

23:00:00

583 rows × 8 columns

Result: Here we can see that in first row of the result we get false and 8201 , it means that there are 8201 rows which dont have 'Snow' string as entry and in second row we can see that there are 583 rows having "Snow" string included in their weather condition.

9.Find all the instance where "Wind speed is above than 24" and "Visibility is 25".

To find the required rows first we will put the condition on Wind Speed_km/h column then on the Visibility_km column then we merge both the condition by using & operator(NOTE : Don't use the "and" in the place of &). Then to print the rows we will put the both conditions in the data[].

In [136…
```
data[(data['Wind Speed_km/h'] > 24) & (data['Visibility_km']==25)]
```

Out[136]:

	Date/Time	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa	Weather_Condition
23	2012-01-01 23:00:00	5.3	2.0	79	30	25.0	99.31	Cloudy
24	2012-01-02 00:00:00	5.2	1.5	77	35	25.0	99.26	Rain Showers
25	2012-01-02 01:00:00	4.6	0.0	72	39	25.0	99.26	Cloudy
26	2012-01-02 02:00:00	3.9	-0.9	71	32	25.0	99.26	Mostly Cloudy
27	2012-01-02 03:00:00	3.7	-1.5	69	33	25.0	99.30	Mostly Cloudy
...	...	...	...	...	...	...	...	...
8705	2012-12-28 17:00:00	-8.6	-12.0	76	26	25.0	101.34	Mainly Clear
8753	2012-12-30 17:00:00	-12.1	-15.8	74	28	25.0	101.26	Mainly Clear
8755	2012-12-30 19:00:00	-13.4	-16.5	77	26	25.0	101.47	Mainly Clear
8759	2012-12-30 23:00:00	-12.1	-15.1	78	28	25.0	101.52	Mostly Cloudy
8760	2012-12-31 00:00:00	-11.1	-14.4	77	26	25.0	101.51	Cloudy

308 rows × 8 columns

so there are 308 rows having the wind speed more than 24 and visibility 25

10. Find the mean of each column on the basis of the weather.

To find the mean of each column on the basis of each weather condition , first we have to group the rows on the basis of the weather condition for this we are using groupby() function .Now to get the mean of the each

column we add the mean() function in it.

In [144…
```
data.groupby('Weather_Condition').mean()
```

```
C:\Users\abc\AppData\Local\Temp\ipykernel_6748\84874417.py:1: FutureWarning: The default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a future version, numeric_only will default to False. Either specify numeric_only or select only columns which should be valid for the function.
  data.groupby('Weather_Condition').mean()
```

Out[144]:

Weather_Condition	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa
Clear	6.825716	0.089367	64.497738	10.557315	30.153243	101.587443
Cloudy	7.970544	2.375810	69.592593	16.127315	26.625752	100.911441
Drizzle	7.353659	5.504878	88.243902	16.097561	17.931707	100.435366
Drizzle,Fog	8.067500	7.033750	93.275000	11.862500	5.257500	100.786625
Drizzle,Ice Pellets,Fog	0.400000	-0.700000	92.000000	20.000000	4.000000	100.790000
Drizzle,Snow	1.050000	0.150000	93.500000	14.000000	10.500000	100.890000
Drizzle,Snow,Fog	0.693333	0.120000	95.866667	15.533333	5.513333	99.281333
Fog	4.303333	3.159333	92.286667	7.946667	6.248000	101.184067
Freezing Drizzle	-5.657143	-8.000000	83.571429	16.571429	9.200000	100.202857
Freezing Drizzle,Fog	-2.533333	-4.183333	88.500000	17.000000	5.266667	100.441667
Freezing Drizzle,Haze	-5.433333	-8.000000	82.000000	10.333333	2.666667	100.316667
Freezing Drizzle,Snow	-5.109091	-7.072727	86.090909	16.272727	5.872727	100.520909
Freezing Fog	-7.575000	-9.250000	87.750000	4.750000	0.650000	102.320000
Freezing Rain	-3.885714	-6.078571	84.642857	19.214286	8.242857	99.647143
Freezing Rain,Fog	-2.225000	-3.750000	89.500000	15.500000	7.550000	99.945000
Freezing Rain,Haze	-4.900000	-7.450000	82.500000	7.500000	2.400000	100.375000
Freezing Rain,Ice Pellets,Fog	-2.600000	-3.700000	92.000000	28.000000	8.000000	100.950000
Freezing Rain,Snow Grains	-5.000000	-7.300000	84.000000	32.000000	4.800000	98.560000
Haze	-0.200000	-2.975000	81.625000	10.437500	7.831250	101.482500
Mainly Clear	12.558927	4.581671	60.667142	14.144824	34.264862	101.248832
Moderate Rain,Fog	1.700000	0.800000	94.000000	17.000000	6.400000	99.980000
Moderate Snow	-5.525000	-7.250000	87.750000	33.750000	0.750000	100.275000
Moderate Snow,Blowing Snow	-5.450000	-6.500000	92.500000	40.000000	0.600000	100.570000
Mostly Cloudy	10.574287	3.131174	62.102465	15.813920	31.253842	101.025288
Rain	9.786275	7.042810	83.624183	19.254902	18.856536	100.233333
Rain Showers	13.722340	9.187766	75.159574	17.132979	22.816489	100.404043
Rain Showers,Fog	12.800000	12.100000	96.000000	13.000000	6.400000	99.830000
Rain Showers,Snow Showers	2.150000	-1.500000	76.500000	22.500000	21.700000	101.100000
Rain,Fog	8.273276	7.219828	93.189655	14.793103	6.873276	100.500862
Rain,Haze	4.633333	2.066667	83.333333	11.666667	6.700000	100.540000
Rain,Ice Pellets	0.600000	-0.600000	92.000000	24.000000	9.700000	100.120000
Rain,Snow	1.055556	-0.566667	89.000000	28.388889	11.672222	99.951111

Rain,Snow Grains	1.900000	-2.100000	75.000000	26.000000	25.000000	100.600000
Rain,Snow,Fog	0.800000	0.300000	96.000000	9.000000	6.400000	100.730000
Rain,Snow,Ice Pellets	1.100000	-0.175000	91.500000	23.250000	6.000000	100.105000
Snow	-4.524103	-7.623333	79.307692	20.038462	11.171795	100.536103
Snow Pellets	0.700000	-6.400000	59.000000	35.000000	2.400000	99.700000
Snow Showers	-3.506667	-7.866667	72.350000	19.233333	20.158333	100.963500
Snow Showers,Fog	-10.675000	-11.900000	90.750000	13.750000	7.025000	101.292500
Snow,Blowing Snow	-5.410526	-7.621053	84.473684	34.842105	4.105263	99.704737
Snow,Fog	-5.075676	-6.364865	90.675676	17.324324	4.537838	100.688649
Snow,Haze	-4.020000	-6.860000	80.600000	5.000000	4.640000	100.782000
Snow,Ice Pellets	-1.883333	-3.666667	87.666667	23.833333	7.416667	100.548333
Thunderstorms	24.150000	19.750000	77.000000	7.500000	24.550000	100.230000
Thunderstorms,Heavy Rain Showers	10.900000	9.000000	88.000000	9.000000	2.400000	100.260000
Thunderstorms,Moderate Rain Showers,Fog	19.600000	18.500000	93.000000	15.000000	3.200000	100.010000
Thunderstorms,Rain	20.433333	18.533333	89.000000	15.666667	19.833333	100.420000
Thunderstorms,Rain Showers	20.037500	17.618750	86.375000	18.312500	15.893750	100.233750
Thunderstorms,Rain Showers,Fog	21.600000	18.700000	84.000000	19.666667	9.700000	100.063333
Thunderstorms,Rain,Fog	20.600000	18.600000	88.000000	19.000000	4.800000	100.080000

Result: Above we have the mean of each columns on the basis of each weather condition.

11. What is the maximum and minimum of each column against each weather condition?

So again we are going to use the groupby() function over the Weather_Condition column then we will apply the min() function on it .

In [149…
```
data.groupby('Weather_Condition').min()
```

Out[149]:

Weather_Condition	Date/Time	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa
Clear	2012-01-03 19:00:00	-23.3	-28.5	20	0	11.3	99.52
Cloudy	2012-01-01 17:00:00	-21.4	-26.8	18	0	11.3	98.39
Drizzle	2012-01-23 21:00:00	1.1	-0.2	74	0	6.4	97.84
Drizzle,Fog	2012-01-23 20:00:00	0.0	-1.6	85	0	1.0	98.65
Drizzle,Ice Pellets,Fog	2012-12-17 09:00:00	0.4	-0.7	92	20	4.0	100.79
Drizzle,Snow	2012-12-17	0.9	0.1	92	9	9.7	100.63

	15:00:00						
Drizzle,Snow,Fog	2012-12-18 21:00:00	0.3	-0.1	92	7	2.4	97.79
Fog	2012-01-01 00:00:00	-16.0	-17.2	80	0	0.2	98.31
Freezing Drizzle	2012-01-07 11:00:00	-9.0	-12.2	78	6	4.8	98.44
Freezing Drizzle,Fog	2012-01-01 02:00:00	-6.4	-9.0	82	6	3.6	98.74
Freezing Drizzle,Haze	2012-02-01 11:00:00	-5.8	-8.3	81	9	2.0	100.28
Freezing Drizzle,Snow	2012-01-13 03:00:00	-8.3	-10.4	79	6	2.4	99.19
Freezing Fog	2012-01-22 06:00:00	-19.0	-22.9	71	0	0.2	101.97
Freezing Rain	2012-01-07 10:00:00	-6.5	-9.0	81	7	2.8	98.22
Freezing Rain,Fog	2012-01-07 09:00:00	-6.1	-8.7	82	7	2.8	98.32
Freezing Rain,Haze	2012-02-01 14:00:00	-4.9	-7.5	82	6	2.0	100.34
Freezing Rain,Ice Pellets,Fog	2012-12-17 03:00:00	-2.6	-3.7	92	28	8.0	100.95
Freezing Rain,Snow Grains	2012-01-13 09:00:00	-5.0	-7.3	84	32	4.8	98.56
Haze	2012-01-22 12:00:00	-11.5	-16.0	68	0	4.8	100.35
Mainly Clear	2012-01-02 12:00:00	-22.8	-28.0	20	0	12.9	98.67
Moderate Rain,Fog	2012-12-10 08:00:00	1.7	0.8	94	17	6.4	99.98
Moderate Snow	2012-01-12 15:00:00	-6.3	-7.6	83	26	0.6	99.88
Moderate Snow,Blowing Snow	2012-12-27 10:00:00	-5.5	-6.6	92	39	0.6	100.50
Mostly Cloudy	2012-01-01 16:00:00	-23.2	-28.5	18	0	11.3	98.36
Rain	2012-01-01 18:00:00	0.3	-5.7	40	0	4.0	97.52
Rain Showers	2012-01-01 22:00:00	1.6	-7.2	37	0	6.4	98.51
Rain Showers,Fog	2012-10-20 03:00:00	12.8	12.1	96	13	6.4	99.83
Rain Showers,Snow Showers	2012-11-04 08:00:00	2.1	-1.8	75	17	19.3	101.09
Rain,Fog	2012-01-23 18:00:00	0.0	-1.2	83	0	2.0	98.61
Rain,Haze	2012-03-13 07:00:00	4.0	1.0	81	7	4.0	100.50
Rain,Ice Pellets	2012-12-18 05:00:00	0.6	-0.6	92	24	9.7	100.12
Rain,Snow	2012-01-10 05:00:00	0.6	-1.7	81	13	2.4	98.18

Rain,Snow Grains	2012-12-21 00:00:00	1.9	-2.1	75	26	25.0	100.60
Rain,Snow,Fog	2012-12-08 21:00:00	0.8	0.3	96	9	6.4	100.73
Rain,Snow,Ice Pellets	2012-12-21 01:00:00	0.9	-0.7	88	17	4.8	99.85
Snow	2012-01-03 07:00:00	-16.7	-24.6	41	0	1.0	97.75
Snow Pellets	2012-11-24 15:00:00	0.7	-6.4	59	35	2.4	99.70
Snow Showers	2012-01-02 17:00:00	-13.3	-19.3	52	0	2.4	99.49
Snow Showers,Fog	2012-12-26 09:00:00	-11.3	-12.7	89	7	4.0	100.63
Snow,Blowing Snow	2012-01-13 21:00:00	-12.0	-16.2	70	24	0.6	98.11
Snow,Fog	2012-02-10 23:00:00	-10.1	-12.0	77	4	1.2	99.38
Snow,Haze	2012-02-01 17:00:00	-4.3	-7.2	80	0	4.0	100.61
Snow,Ice Pellets	2012-03-03 04:00:00	-4.3	-5.9	76	19	2.8	99.40
Thunderstorms	2012-07-04 16:00:00	21.6	19.4	67	0	24.1	99.84
Thunderstorms,Heavy Rain Showers	2012-05-29 06:00:00	10.9	9.0	88	9	2.4	100.26
Thunderstorms,Moderate Rain Showers,Fog	2012-07-17 06:00:00	19.6	18.5	93	15	3.2	100.01
Thunderstorms,Rain	2012-05-25 20:00:00	19.4	18.2	83	4	16.1	100.19
Thunderstorms,Rain Showers	2012-05-29 04:00:00	11.0	7.0	68	7	6.4	99.65
Thunderstorms,Rain Showers,Fog	2012-06-29 03:00:00	19.5	16.1	80	7	9.7	99.71
Thunderstorms,Rain,Fog	2012-07-17 05:00:00	20.6	18.6	88	19	4.8	100.08

to find the max of each column over all kind of the weather condition we will use groupby on Weather_Condition column then apply max() on it.

In [155…
```
data.groupby('Weather_Condition').max()
```

Out[155]:

Weather_Condition	Date/Time	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa
Clear	2012-12-30 20:00:00	32.8	20.4	99	33	48.3	103.63
Cloudy	2012-12-31 06:00:00	30.5	22.6	99	54	48.3	103.65
Drizzle	2012-12-22 01:00:00	18.8	17.7	96	30	25.0	101.56

Drizzle,Fog	2012-12-19 10:00:00	19.9	19.1	100	28	9.7	102.07
Drizzle,Ice Pellets,Fog	2012-12-17 09:00:00	0.4	-0.7	92	20	4.0	100.79
Drizzle,Snow	2012-12-19 18:00:00	1.2	0.2	95	19	11.3	101.15
Drizzle,Snow,Fog	2012-12-22 03:00:00	1.1	0.6	98	32	9.7	100.15
Fog	2012-12-29 10:00:00	20.8	19.6	100	22	9.7	103.04
Freezing Drizzle	2012-12-17 00:00:00	-2.3	-3.3	93	26	12.9	101.02
Freezing Drizzle,Fog	2012-12-10 05:00:00	-0.3	-2.3	94	33	8.0	101.27
Freezing Drizzle,Haze	2012-02-01 13:00:00	-5.0	-7.7	83	11	4.0	100.36
Freezing Drizzle,Snow	2012-12-28 02:00:00	-3.3	-4.6	94	24	12.9	101.18
Freezing Fog	2012-03-17 06:00:00	-0.1	-0.3	99	9	0.8	102.85
Freezing Rain	2012-12-17 02:00:00	0.3	-1.7	92	28	16.1	101.00
Freezing Rain,Fog	2012-12-17 01:00:00	0.1	-0.9	93	26	9.7	101.01
Freezing Rain,Haze	2012-02-01 15:00:00	-4.9	-7.4	83	9	2.8	100.41
Freezing Rain,Ice Pellets,Fog	2012-12-17 03:00:00	-2.6	-3.7	92	28	8.0	100.95
Freezing Rain,Snow Grains	2012-01-13 09:00:00	-5.0	-7.3	84	32	4.8	98.56
Haze	2012-12-13 12:00:00	14.1	11.1	86	17	9.7	102.97
Mainly Clear	2012-12-30 22:00:00	33.0	21.2	99	63	48.3	103.59
Moderate Rain,Fog	2012-12-10 08:00:00	1.7	0.8	94	17	6.4	99.98
Moderate Snow	2012-12-27 09:00:00	-4.9	-6.7	93	39	0.8	100.67
Moderate Snow,Blowing Snow	2012-12-27 12:00:00	-5.4	-6.4	93	41	0.6	100.64
Mostly Cloudy	2012-12-31 03:00:00	32.4	24.4	100	83	48.3	103.65
Rain	2012-12-21 21:00:00	22.8	20.4	99	52	48.3	102.26
Rain Showers	2012-12-14 11:00:00	26.4	23.0	97	41	48.3	102.31
Rain Showers,Fog	2012-10-20 03:00:00	12.8	12.1	96	13	6.4	99.83
Rain Showers,Snow Showers	2012-12-05 10:00:00	2.2	-1.2	78	28	24.1	101.11
Rain,Fog	2012-12-10 17:00:00	21.7	19.5	100	46	9.7	101.77
Rain,Haze	2012-03-13	5.5	2.9	86	17	9.7	100.61

	09:00:00						
Rain,Ice Pellets	2012-12-18 05:00:00	0.6	-0.6	92	24	9.7	100.12
Rain,Snow	2012-12-21 09:00:00	1.7	0.5	94	52	25.0	101.07
Rain,Snow Grains	2012-12-21 00:00:00	1.9	-2.1	75	26	25.0	100.60
Rain,Snow,Fog	2012-12-08 21:00:00	0.8	0.3	96	9	6.4	100.73
Rain,Snow,Ice Pellets	2012-12-21 05:00:00	1.3	0.1	94	28	6.4	100.47
Snow	2012-12-31 23:00:00	3.7	0.3	96	57	25.0	102.73
Snow Pellets	2012-11-24 15:00:00	0.7	-6.4	59	35	2.4	99.70
Snow Showers	2012-12-31 08:00:00	2.9	-0.7	94	37	48.3	102.50
Snow Showers,Fog	2012-12-29 13:00:00	-10.0	-11.1	92	22	9.7	102.52
Snow,Blowing Snow	2012-12-27 19:00:00	-1.4	-2.9	91	48	9.7	100.62
Snow,Fog	2012-12-31 10:00:00	1.1	0.8	99	35	9.7	102.07
Snow,Haze	2012-02-01 21:00:00	-3.6	-6.4	81	15	6.4	100.99
Snow,Ice Pellets	2012-12-17 06:00:00	0.8	-1.7	92	33	11.3	100.96
Thunderstorms	2012-07-16 01:00:00	26.7	20.1	87	15	25.0	100.62
Thunderstorms,Heavy Rain Showers	2012-05-29 06:00:00	10.9	9.0	88	9	2.4	100.26
Thunderstorms,Moderate Rain Showers,Fog	2012-07-17 06:00:00	19.6	18.5	93	15	3.2	100.01
Thunderstorms,Rain	2012-07-23 18:00:00	21.3	19.1	93	30	24.1	100.83
Thunderstorms,Rain Showers	2012-09-14 20:00:00	25.5	23.1	98	32	25.0	101.06
Thunderstorms,Rain Showers,Fog	2012-07-31 20:00:00	22.9	21.3	91	35	9.7	100.64
Thunderstorms,Rain,Fog	2012-07-17 05:00:00	20.6	18.6	88	19	4.8	100.08

Result: Above we have the min and max of each columns on the basis of each weather category.

12. Find all the records where weather condition is fog.

To answer this question we will use the groupby function over Weather_Condition and after that use get_group with argument 'Fog'

In [156…

```
data.groupby('Weather_Condition').get_group('Fog')
# you can also write the above code in following way :
```

```
# data[data.groupby('Weather_Condition')=='Fog']
# it will also give the same result
```

Out[156]:

	Date/Time	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa	Weather_Condition
0	2012-01-01 00:00:00	-1.8	-3.9	86	4	8.0	101.24	Fog
1	2012-01-01 01:00:00	-1.8	-3.7	87	4	8.0	101.24	Fog
4	2012-01-01 04:00:00	-1.5	-3.3	88	7	4.8	101.23	Fog
5	2012-01-01 05:00:00	-1.4	-3.3	87	9	6.4	101.27	Fog
6	2012-01-01 06:00:00	-1.5	-3.1	89	7	6.4	101.29	Fog
...	...	...	...	...	...	...	...	...
8716	2012-12-29 04:00:00	-16.0	-17.2	90	6	9.7	101.25	Fog
8717	2012-12-29 05:00:00	-14.8	-15.9	91	4	6.4	101.25	Fog
8718	2012-12-29 06:00:00	-13.8	-15.3	88	4	9.7	101.25	Fog
8719	2012-12-29 07:00:00	-14.8	-16.4	88	7	8.0	101.22	Fog
8722	2012-12-29 10:00:00	-12.0	-13.3	90	7	6.4	101.15	Fog

150 rows × 8 columns

Result: Here we can see that there 150 hours when weather was having fog condition .

13. Find all the instance where "Weather is Clear" or 'Visibility is above 40'.

Here we will use the or operator between the given conditions. The code is following:

In [166…

```
data[(data.groupby('Weather_Condition')=='Clear')| (data['Visibility_km']>40)]
```

Out[166]:

	Date/Time	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa	Weather_Condition
106	2012-01-05 10:00:00	-6.0	-10.0	73	17	48.3	100.45	Mainly Clear
107	2012-01-05 11:00:00	-5.6	-10.2	70	22	48.3	100.41	Mainly Clear
108	2012-01-05 12:00:00	-4.7	-9.6	69	20	48.3	100.38	Mainly Clear
109	2012-01-05 13:00:00	-4.4	-9.7	66	26	48.3	100.40	Mainly Clear
110	2012-01-05 14:00:00	-5.1	-10.7	65	22	48.3	100.46	Mainly Clear
...	...	...	...	...	...	...	...	...

8748	2012-12-30 12:00:00	-12.2	-15.7	75	26	48.3	100.91	Mostly Cloudy
8749	2012-12-30 13:00:00	-12.4	-16.2	73	37	48.3	100.92	Mostly Cloudy
8750	2012-12-30 14:00:00	-11.8	-16.1	70	37	48.3	100.96	Mainly Clear
8751	2012-12-30 15:00:00	-11.3	-15.6	70	32	48.3	101.05	Mainly Clear
8752	2012-12-30 16:00:00	-11.4	-15.5	72	26	48.3	101.15	Mainly Clear

2014 rows × 8 columns

Result: So there are 2014 hour when the weather condition was clear or the visibility was more than 40 km.

14. Find the instances where weather is clear and realtive humidity is above 50 or visibility is above 40 .

Here we will use the and and or operator to answer this query.

In [168…
```
data[((data['Weather_Condition']=='Clear') & (data['Rel Hum_%']>50)) | (data['Visibility
```

Out[168]:

	Date/Time	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa	Weather_Condition
106	2012-01-05 10:00:00	-6.0	-10.0	73	17	48.3	100.45	Mainly Clear
107	2012-01-05 11:00:00	-5.6	-10.2	70	22	48.3	100.41	Mainly Clear
108	2012-01-05 12:00:00	-4.7	-9.6	69	20	48.3	100.38	Mainly Clear
109	2012-01-05 13:00:00	-4.4	-9.7	66	26	48.3	100.40	Mainly Clear
110	2012-01-05 14:00:00	-5.1	-10.7	65	22	48.3	100.46	Mainly Clear
111	2012-01-05 15:00:00	-4.3	-12.0	55	26	48.3	100.52	Mainly Clear
114	2012-01-05 18:00:00	-7.1	-14.4	56	11	25.0	100.71	Clear
115	2012-01-05 19:00:00	-9.2	-15.4	61	7	25.0	100.80	Clear
116	2012-01-05 20:00:00	-9.8	-15.7	62	9	25.0	100.83	Clear
117	2012-01-05 21:00:00	-9.0	-14.8	63	13	25.0	100.83	Clear
183	2012-01-08 15:00:00	-6.6	-12.9	61	20	48.3	102.04	Mostly Cloudy
241	2012-01-11 01:00:00	-10.7	-17.8	56	17	25.0	101.49	Clear
242	2012-01-11 02:00:00	-12.0	-18.9	56	19	25.0	101.57	Clear
243	2012-01-11 03:00:00	-12.7	-19.4	57	19	25.0	101.64	Clear

244	2012-01-11 04:00:00	-13.4	-20.1	57	17	25.0	101.66	Clear
324	2012-01-14 12:00:00	-17.5	-23.8	58	20	48.3	101.16	Mostly Cloudy
325	2012-01-14 13:00:00	-17.1	-24.1	55	17	48.3	101.18	Mainly Clear
326	2012-01-14 14:00:00	-16.7	-23.4	56	17	48.3	101.20	Mostly Cloudy
327	2012-01-14 15:00:00	-16.7	-23.4	56	22	48.3	101.24	Mostly Cloudy
344	2012-01-15 08:00:00	-23.3	-28.5	62	7	24.1	102.45	Clear
345	2012-01-15 09:00:00	-22.2	-27.8	60	9	48.3	102.57	Mainly Clear
346	2012-01-15 10:00:00	-20.6	-26.8	58	9	48.3	102.66	Mainly Clear
347	2012-01-15 11:00:00	-19.3	-26.1	55	9	48.3	102.68	Mainly Clear
348	2012-01-15 12:00:00	-18.0	-25.5	52	13	48.3	102.67	Mainly Clear
349	2012-01-15 13:00:00	-16.8	-24.2	53	20	48.3	102.65	Mainly Clear
350	2012-01-15 14:00:00	-16.0	-23.4	53	26	48.3	102.66	Mainly Clear
351	2012-01-15 15:00:00	-15.4	-22.8	53	24	48.3	102.71	Clear
352	2012-01-15 16:00:00	-15.1	-22.8	52	24	48.3	102.79	Clear
353	2012-01-15 17:00:00	-16.2	-23.2	55	15	25.0	102.85	Clear
354	2012-01-15 18:00:00	-16.3	-22.9	57	17	25.0	102.89	Clear
355	2012-01-15 19:00:00	-16.3	-22.7	58	20	25.0	102.94	Clear
356	2012-01-15 20:00:00	-16.7	-22.4	61	17	25.0	102.98	Clear
357	2012-01-15 21:00:00	-16.9	-22.2	63	15	25.0	103.02	Clear
358	2012-01-15 22:00:00	-16.9	-21.7	66	17	25.0	103.07	Clear
359	2012-01-15 23:00:00	-16.9	-21.4	68	15	25.0	103.09	Clear
360	2012-01-16 00:00:00	-17.1	-21.9	66	15	25.0	103.09	Clear
361	2012-01-16 01:00:00	-17.8	-22.7	65	6	25.0	103.05	Clear
362	2012-01-16 02:00:00	-18.5	-23.0	68	6	25.0	103.08	Clear
363	2012-01-16 03:00:00	-19.2	-23.8	67	6	25.0	103.07	Clear
364	2012-01-16 04:00:00	-18.7	-23.5	66	0	25.0	103.05	Clear

365	2012-01-16 05:00:00	-19.1	-23.3	69	0	25.0	103.02	Clear
366	2012-01-16 06:00:00	-18.7	-23.2	68	0	25.0	103.01	Clear
367	2012-01-16 07:00:00	-17.9	-21.5	73	4	48.3	102.96	Mainly Clear
368	2012-01-16 08:00:00	-17.9	-22.6	67	9	48.3	102.90	Mostly Cloudy
369	2012-01-16 09:00:00	-14.4	-20.7	59	22	48.3	102.75	Cloudy
370	2012-01-16 10:00:00	-13.6	-18.7	65	19	48.3	102.62	Mostly Cloudy
371	2012-01-16 11:00:00	-12.7	-17.7	66	20	48.3	102.44	Mostly Cloudy
372	2012-01-16 12:00:00	-10.7	-16.9	60	19	48.3	102.17	Mostly Cloudy
373	2012-01-16 13:00:00	-10.1	-16.1	61	13	48.3	101.92	Cloudy
374	2012-01-16 14:00:00	-9.0	-13.8	68	13	48.3	101.71	Mostly Cloudy

15. Plot a graph to show the average temperature of the each month.

Now we have to plot the graph , for plotting the graph we will use the matplotlib.pyplot library of the python . Before that we have to calculate the average temperature of each month .For that we have to find the mean of the temperature group by each month. for this we have the following code:

In [23]:
```
# here we have group the Date/time according to the month and then find mean of the Temp
y=data.groupby(data['Date/Time'].dt.month)['Temp_C'].mean()
print(y)
```

```
Date/Time
1     -7.371505
2     -4.225000
3      3.121237
4      7.009306
5     16.237769
6     20.134028
7     22.790054
8     22.279301
9     16.484444
10    10.954973
11     0.931389
12    -3.306317
Name: Temp_C, dtype: float64
```

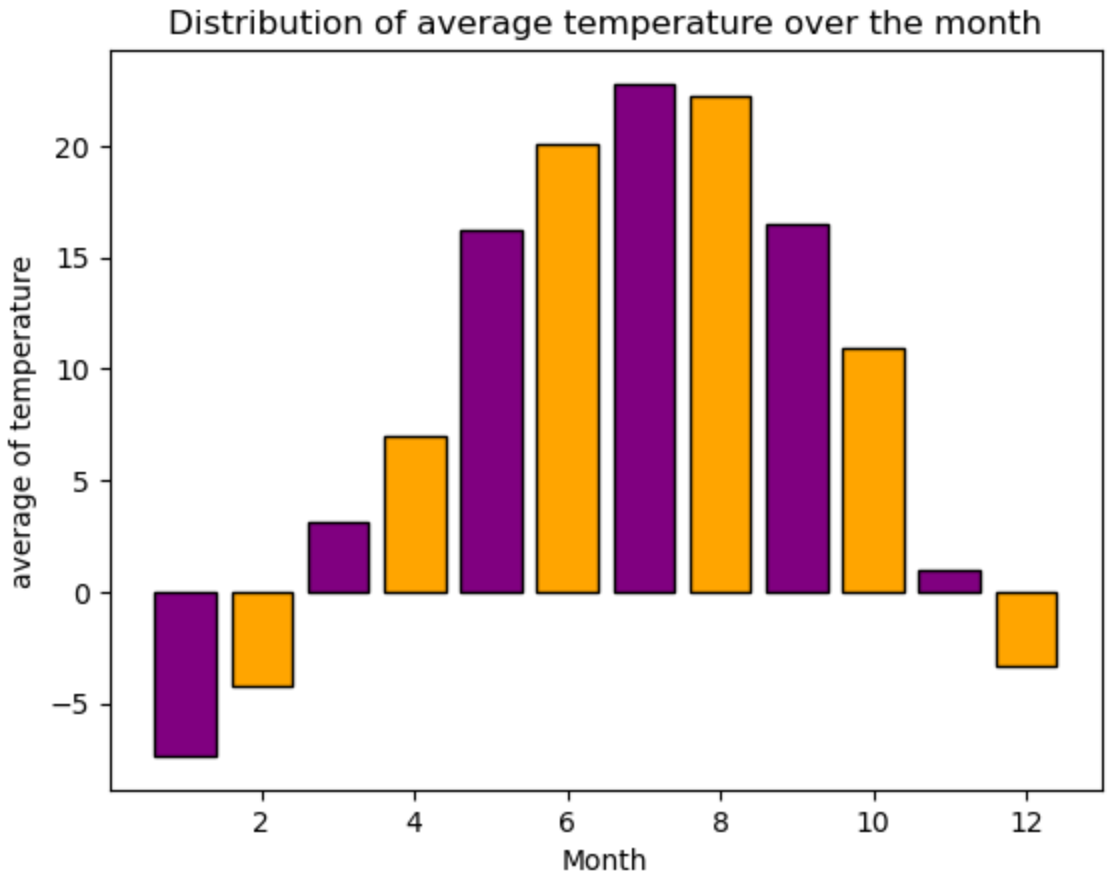
so here we have the average temperature of each month for e.g. in 4th month the average temperature was 7.009306 degree Celsius.

In the above series of mean of temperature the indexes are the month , so we have to plot the graph between the above series and it's indexes.

In [212…
```
# assign a variable to the indexes of the above series
x=y.index    # now x will represent the index of the y series
```

```
# we are going to plot the bar graph so for that we will use the plt.bar() function
plt.bar(x,y,color=['purple','orange'],edgecolor='black')
# Now we will add the title in the graph
plt.title("Distribution of average temperature over the month")
# we will also add the labels to the axis
plt.xlabel("Month")                    #label for x axis
plt.ylabel("average of temperature")   #label for y axis
```

Out[212]:
```
Text(0, 0.5, 'average of temperature')
```

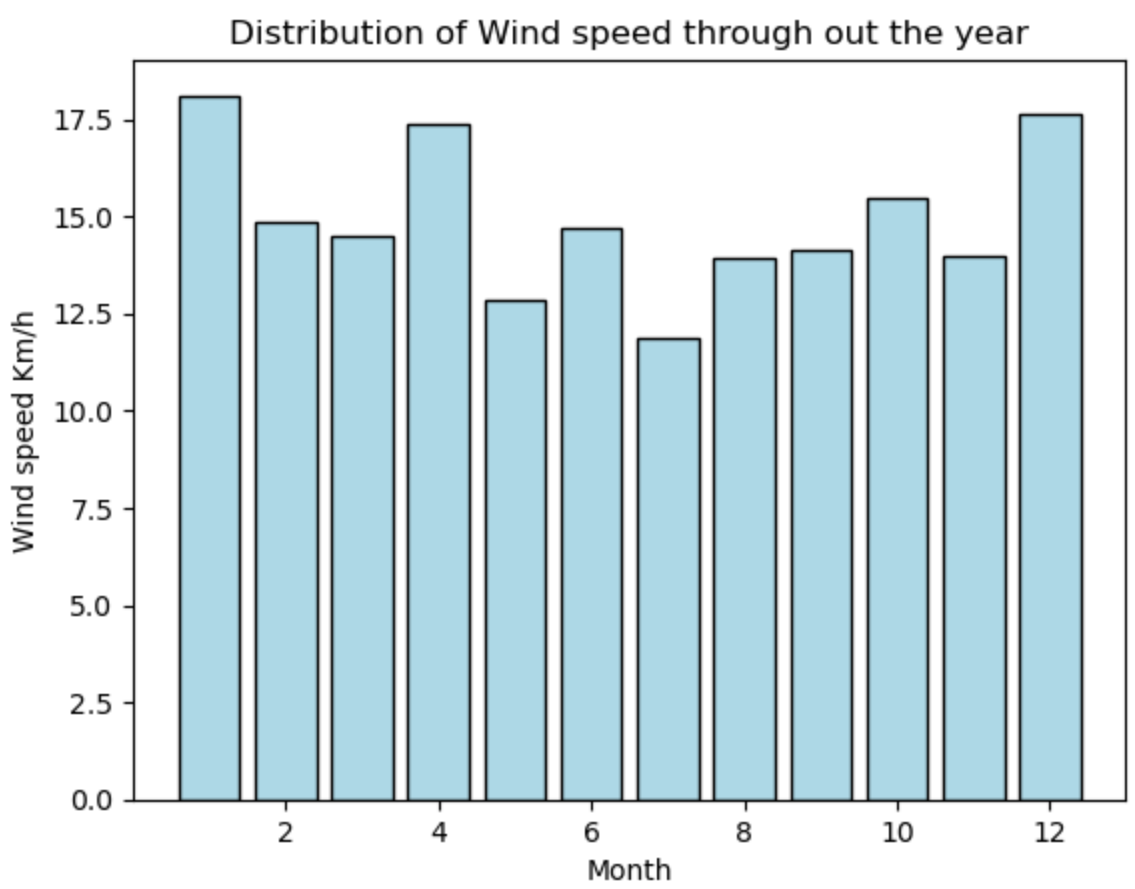


In [ ]:
```
From the graph we can say that in July the average temperature was maximum and minimum i
```

16. Plot the graph for the average of the wind speed for each month.

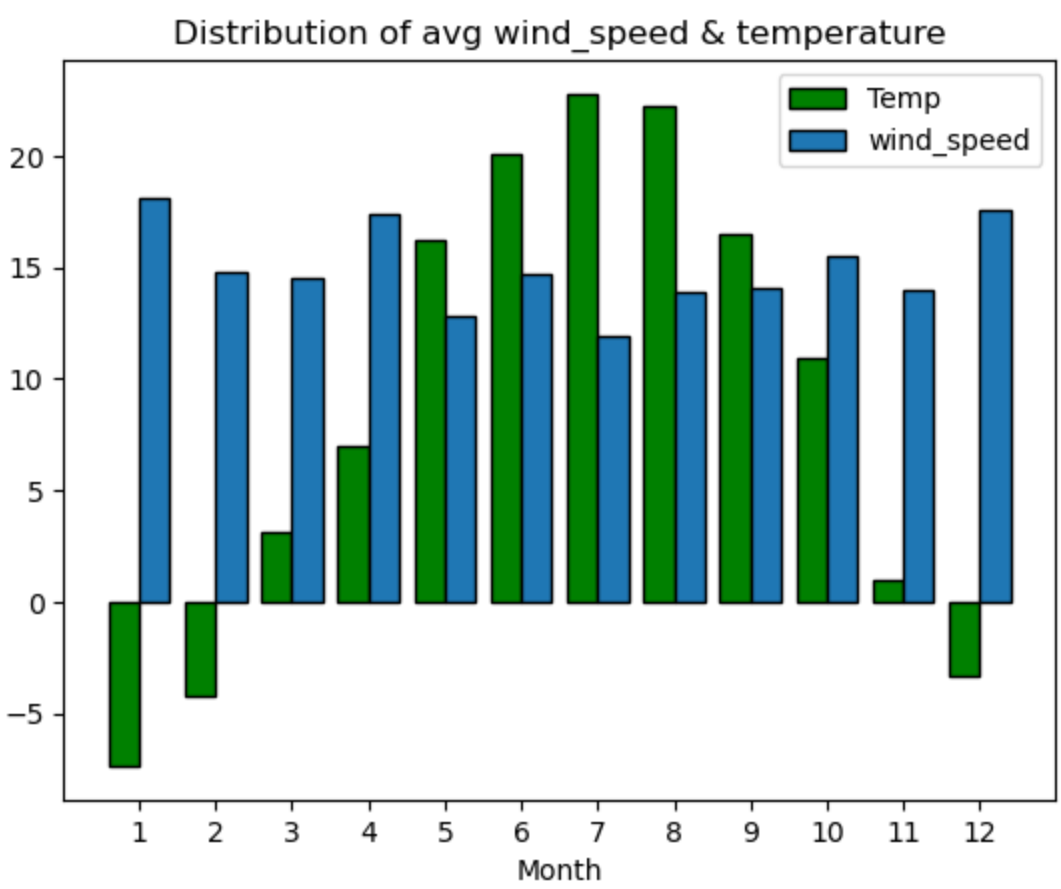
In [226…
```
y1=data.groupby(data['Date/Time'].dt.month)['Wind Speed_km/h'].mean()
plt.title('Distribution of Wind speed through out the year')
plt.xlabel('Month')
plt.ylabel('Wind speed Km/h')
plt.bar(x,y1,edgecolor='black',color='lightblue')
```

Out[226]:
```
<BarContainer object of 12 artists>
```



From the graph we can say that wind speed was maximum in January and minimum in the month of July.

In [54]:
```
x1=np.arange(len(x))
plt.bar(x1-0.2,y,color='green',width=0.4,edgecolor='black',label="Temp")
plt.bar(x1+0.2,y1,width=0.4,edgecolor='black',label='wind_speed')
plt.xticks(x1,x)
plt.title("Distribution of avg wind_speed & temperature")
plt.xlabel("Month")
plt.legend()
plt.show()
```



so from the graph we can say that when the temperature was lowest then wind speed was highest and this situation occured in month of January , and in July when the temperature was highest then wind speed was lowest.

17. Find the hottest day in July.

In the dataset we have the data for each hour for each day so to find the hottest day in july we have first grouped the data according to the month then find the data having month july then we find the mean of all columns grouped by date

In [151…
```
data2=data.groupby(data['Date/Time'].dt.month).get_group(7)
data3=data2.groupby([data2['Date/Time'].dt.date]).mean()
data3
```

```
C:\Users\abc\AppData\Local\Temp\ipykernel_7248\3247483057.py:2: FutureWarning: The defau
lt value of numeric_only in DataFrameGroupBy.mean is deprecated. In a future version, nu
meric_only will default to False. Either specify numeric_only or select only columns whi
ch should be valid for the function.
  data3=data2.groupby([data2['Date/Time'].dt.date]).mean()
```

Out[151]:

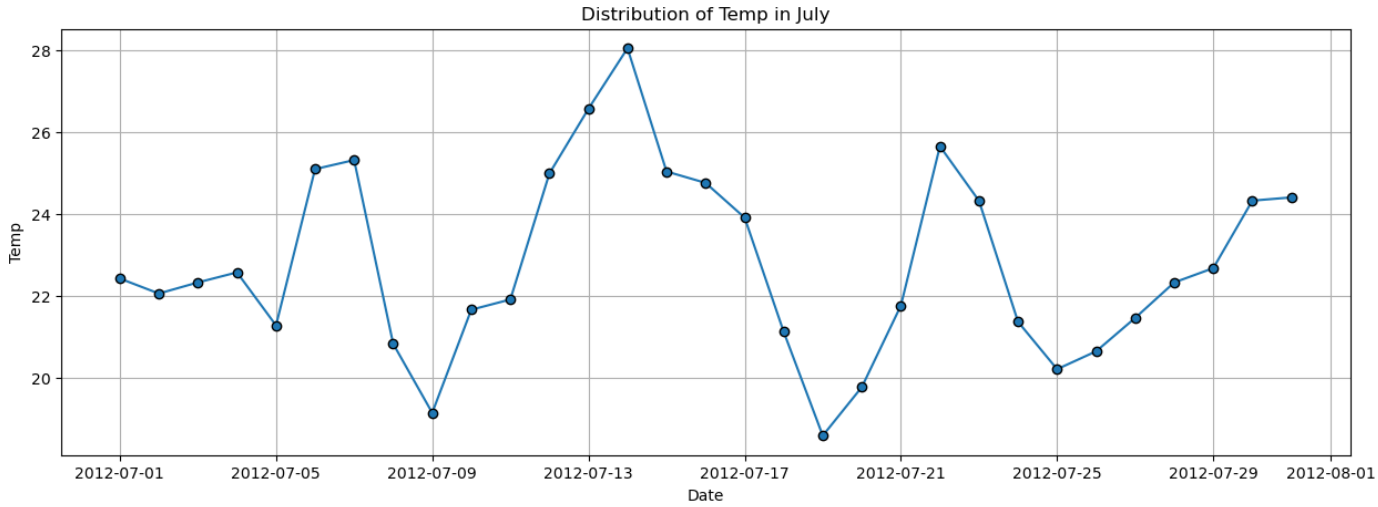
Date/Time	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa
2012-07-01	22.420833	13.729167	58.416667	10.833333	35.454167	100.453333
2012-07-02	22.054167	12.579167	56.916667	12.833333	38.516667	100.727500
2012-07-03	22.325000	11.875000	53.125000	9.041667	40.495833	100.696250
2012-07-04	22.570833	18.333333	77.583333	14.625000	20.791667	100.165417
2012-07-05	21.275000	16.008333	72.791667	10.625000	36.462500	100.632083
2012-07-06	25.095833	18.616667	68.375000	12.666667	32.466667	100.573333

2012-07-07	25.316667	16.125000	57.500000	14.375000	32.504167	100.404583
2012-07-08	20.825000	9.566667	50.291667	15.916667	39.487500	100.741667
2012-07-09	19.137500	7.895833	50.750000	11.791667	39.525000	101.008333
2012-07-10	21.658333	10.416667	50.666667	12.666667	40.495833	101.173333
2012-07-11	21.908333	9.191667	46.500000	10.041667	38.516667	101.593750
2012-07-12	24.987500	13.737500	51.375000	14.416667	38.516667	101.466667
2012-07-13	26.570833	14.962500	51.083333	14.000000	35.491667	101.466250
2012-07-14	28.050000	17.229167	52.875000	15.416667	33.475000	101.354583
2012-07-15	25.037500	19.308333	71.458333	10.416667	26.025000	101.026250
2012-07-16	24.758333	19.079167	72.125000	8.416667	23.125000	100.470000
2012-07-17	23.908333	19.933333	79.000000	12.541667	18.145833	99.960833
2012-07-18	21.116667	12.875000	61.708333	15.125000	34.445833	100.718750
2012-07-19	18.579167	9.500000	57.750000	8.458333	40.533333	101.240000
2012-07-20	19.766667	10.016667	54.916667	8.333333	41.504167	101.448750
2012-07-21	21.758333	13.095833	60.291667	7.333333	38.516667	101.297500
2012-07-22	25.654167	16.966667	59.041667	17.583333	37.545833	101.118333
2012-07-23	24.320833	18.862500	73.166667	13.333333	28.700000	100.545000
2012-07-24	21.362500	15.333333	70.750000	15.416667	28.137500	100.035000
2012-07-25	20.204167	12.866667	64.166667	13.708333	33.550000	100.590417
2012-07-26	20.641667	16.008333	75.083333	7.083333	24.066667	99.971250
2012-07-27	21.458333	15.945833	72.416667	5.666667	24.400000	100.502917
2012-07-28	22.325000	12.170833	54.125000	13.333333	39.525000	101.150417
2012-07-29	22.675000	15.429167	64.166667	9.375000	37.508333	101.172917
2012-07-30	24.325000	16.287500	63.500000	9.208333	38.516667	101.150417
2012-07-31	24.404167	18.512500	70.625000	13.916667	26.862500	100.822500

In [ ]:
```
Here we have written the code to plot the graph
```

In [209…
```
plt.figure(figsize=(15,5))        # figure(figsize=()) will be helpful to increase the p
plt.plot(days,data3.Temp_C,marker="o",mec='black')  # by plot() we can draw the curve
plt.grid(axis='y')    #grid() will add the grid lines in graph
plt.grid(axis='x')
plt.title('Distribution of Temp in July')
plt.xlabel('Date')
plt.ylabel('Temp')
```

Out[209]:
```
Text(0, 0.5, 'Temp')
```



so from the graph we can see that at 14/07/2012 we had the hottest day of July with temperature 28 degree Celcius.

18. Find the coolest day fo the year?

In [9]:
```
data.iloc[data.Temp_C.idxmin()]
```

Out[9]:
```
Date/Time           1/15/2012 8:00
Temp_C                       -23.3
Dew Point Temp_C             -28.5
Rel Hum_%                       62
Wind Speed_km/h                  7
Visibility_km                 24.1
Press_kPa                   102.45
Weather                      Clear
Name: 344, dtype: object
```

19. Find the hottest day of the year?

In [10]:
```
data.iloc[data.Temp_C.idxmax()]
```

Out[10]:
```
Date/Time           6/21/2012 15:00
Temp_C                         33.0
Dew Point Temp_C               19.0
Rel Hum_%                        44
Wind Speed_km/h                  24
Visibility_km                 24.1
Press_kPa                    100.2
Weather                Mainly Clear
Name: 4143, dtype: object
```

In [ ]:
```

```