In [5]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Read CSV file

In [8]:
```
df = pd.read_csv('data.csv')
df.head()
```

Out[8]:

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
0	2006-04-01 00:00:00.000 +0200	Partly Cloudy	rain	9.472222	7.388889	0.89	14.1197	251.0	15.8263	0.0	1015.13	Partly cloudy throughout the day.
1	2006-04-01 01:00:00.000 +0200	Partly Cloudy	rain	9.355556	7.227778	0.86	14.2646	259.0	15.8263	0.0	1015.63	Partly cloudy throughout the day.
2	2006-04-01 02:00:00.000 +0200	Mostly Cloudy	rain	9.377778	9.377778	0.89	3.9284	204.0	14.9569	0.0	1015.94	Partly cloudy throughout the day.
3	2006-04-01 03:00:00.000 +0200	Partly Cloudy	rain	8.288889	5.944444	0.83	14.1036	269.0	15.8263	0.0	1016.41	Partly cloudy throughout the day.
4	2006-04-01 04:00:00.000 +0200	Mostly Cloudy	rain	8.755556	6.977778	0.83	11.0446	259.0	15.8263	0.0	1016.51	Partly cloudy throughout the day.

Check a particular column type

In [9]:
```
df['Humidity'].dtype
```

Out[9]:
```
dtype('float64')
```

Check types for all the columns

In [10]:
```
df.dtypes
```

Out[10]:
```
Formatted Date               object
Summary                      object
Precip Type                  object
Temperature (C)             float64
Apparent Temperature (C)    float64
Humidity                    float64
Wind Speed (km/h)           float64
Wind Bearing (degrees)      float64
Visibility (km)             float64
Loud Cover                  float64
Pressure (millibars)        float64
Daily Summary                object
dtype: object
```

Data Frames attributes

In [ ]:
```
#  dtypes   --->  list thetypes of the columns
#  columns  --->  list the column names
#  axes     --->  list therowlabelsandcolumnnames
#  ndim     --->  number of dimensions
#  size     --->  number of elements
#  shape    --->  return a tuple representing the dimensionality
#  values   --->  numpy representation of the data
```

In [11]:
```
df.columns
```

Out[11]:
```
Index(['Formatted Date', 'Summary', 'Precip Type', 'Temperature (C)',
       'Apparent Temperature (C)', 'Humidity', 'Wind Speed (km/h)',
       'Wind Bearing (degrees)', 'Visibility (km)', 'Loud Cover',
       'Pressure (millibars)', 'Daily Summary'],
      dtype='object')
```

In [12]:
```
df.axes
```

Out[12]:
```
[RangeIndex(start=0, stop=96453, step=1),
 Index(['Formatted Date', 'Summary', 'Precip Type', 'Temperature (C)',
        'Apparent Temperature (C)', 'Humidity', 'Wind Speed (km/h)',
        'Wind Bearing (degrees)', 'Visibility (km)', 'Loud Cover',
        'Pressure (millibars)', 'Daily Summary'],
       dtype='object')]
```

In [13]:
```
df.ndim
```

Out[13]:
```
2
```

In [14]:
```
df.size
```

Out[14]:
```
1157436
```

In [15]:
```
df.shape
```

Out[15]:
```
(96453, 12)
```

In [16]:
```
df.values
```

Out[16]:
```
array([['2006-04-01 00:00:00.000 +0200', 'Partly Cloudy', 'rain', ...,
        0.0, 1015.13, 'Partly cloudy throughout the day.'],
       ['2006-04-01 01:00:00.000 +0200', 'Partly Cloudy', 'rain', ...,
        0.0, 1015.63, 'Partly cloudy throughout the day.'],
       ['2006-04-01 02:00:00.000 +0200', 'Mostly Cloudy', 'rain', ...,
        0.0, 1015.94, 'Partly cloudy throughout the day.'],
       ...,
       ['2016-09-09 21:00:00.000 +0200', 'Partly Cloudy', 'rain', ...,
        0.0, 1015.66, 'Partly cloudy starting in the morning.'],
       ['2016-09-09 22:00:00.000 +0200', 'Partly Cloudy', 'rain', ...,
        0.0, 1015.95, 'Partly cloudy starting in the morning.'],
       ['2016-09-09 23:00:00.000 +0200', 'Partly Cloudy', 'rain', ...,
        0.0, 1016.16, 'Partly cloudy starting in the morning.']],
      dtype=object)
```

Data Frames Methods

In [17]:
```
# Standard devation

df["Humidity"].std()
```

Out[17]:
```
0.19547273906722662
```

In [18]:
```
# Return Mean
df["Humidity"].mean()
```

Out[18]:
```
0.7348989663358906
```

In [20]:
```
# Return a random sample of the data frame

df.sample(10)
```

Out[20]:

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
70377	2014-04-18 09:00:00.000 +0200	Mostly Cloudy	rain	11.138889	11.138889	0.76	10.9802	351.0	16.1000	0.0	1008.93	Mostly cloudy throughout the day.
17836	2008-04-20 04:00:00.000 +0200	Partly Cloudy	rain	9.566667	8.911111	0.87	6.2790	200.0	11.6403	0.0	1008.19	Partly cloudy starting in the morning.
36370	2010-08-30 10:00:00.000 +0200	Partly Cloudy	rain	18.750000	18.750000	0.57	7.8890	101.0	11.2056	0.0	1011.66	Mostly cloudy until night.
58203	2012-03-28 04:00:00.000 +0200	Clear	rain	5.000000	2.411111	0.65	11.2700	340.0	16.1000	0.0	1022.00	Partly cloudy until night.
67013	2013-03-03 05:00:00.000 +0100	Clear	snow	-2.866667	-2.866667	1.00	1.6583	203.0	4.7495	0.0	1021.41	Partly cloudy starting in the morning continui...
92177	2016-06-13 20:00:00.000 +0200	Mostly Cloudy	rain	17.850000	17.850000	0.91	4.7012	6.0	10.3523	0.0	1000.30	Mostly cloudy throughout the day.
73519	2014-01-28 07:00:00.000 +0100	Foggy	snow	-4.861111	-4.861111	0.92	0.1449	129.0	2.5599	0.0	1005.59	Foggy until morning.
63275	2013-12-25 11:00:00.000 +0100	Partly Cloudy	rain	7.133333	3.733333	0.56	20.3987	160.0	11.9784	0.0	1015.30	Partly cloudy starting in the morning.
71812	2014-12-17 04:00:00.000 +0100	Foggy	rain	6.044444	6.044444	0.96	2.8175	157.0	3.0912	0.0	1010.28	Foggy in the morning.
54847	2012-02-10 07:00:00.000 +0100	Foggy	snow	-21.822222	-21.822222	0.80	3.0751	323.0	1.3685	0.0	1033.66	Foggy starting in the morning continuing until...

In [21]:
```
df['Humidity'].sample(5)
```

Out[21]:
```
19744    0.75
65756    0.79
41705    0.93
5513     0.56
13129    0.62
Name: Humidity, dtype: float64
```

In [23]:
```

```

Out[23]:

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
0	2006-04-01 00:00:00.000 +0200	Partly Cloudy	rain	9.472222	7.388889	0.89	14.1197	251.0	15.8263	0.0	1015.13	Partly cloudy throughout the day.
1	2006-04-01 01:00:00.000 +0200	Partly Cloudy	rain	9.355556	7.227778	0.86	14.2646	259.0	15.8263	0.0	1015.63	Partly cloudy throughout the day.
2	2006-04-01 02:00:00.000 +0200	Mostly Cloudy	rain	9.377778	9.377778	0.89	3.9284	204.0	14.9569	0.0	1015.94	Partly cloudy throughout the day.
3	2006-04-01 03:00:00.000 +0200	Partly Cloudy	rain	8.288889	5.944444	0.83	14.1036	269.0	15.8263	0.0	1016.41	Partly cloudy throughout the day.
4	2006-04-01 04:00:00.000 +0200	Mostly Cloudy	rain	8.755556	6.977778	0.83	11.0446	259.0	15.8263	0.0	1016.51	Partly cloudy throughout the day.
...	...	...	...	...	...	...	...	...	...	...	...	...
96448	2016-09-09 19:00:00.000 +0200	Partly Cloudy	rain	26.016667	26.016667	0.43	10.9963	31.0	16.1000	0.0	1014.36	Partly cloudy starting in the morning.
96449	2016-09-09 20:00:00.000 +0200	Partly Cloudy	rain	24.583333	24.583333	0.48	10.0947	20.0	15.5526	0.0	1015.16	Partly cloudy starting in the morning.
96450	2016-09-09 21:00:00.000 +0200	Partly Cloudy	rain	22.038889	22.038889	0.56	8.9838	30.0	16.1000	0.0	1015.66	Partly cloudy starting in the morning.
96451	2016-09-09 22:00:00.000 +0200	Partly Cloudy	rain	21.522222	21.522222	0.60	10.5294	20.0	16.1000	0.0	1015.95	Partly cloudy starting in the morning.
96452	2016-09-09 23:00:00.000 +0200	Partly Cloudy	rain	20.438889	20.438889	0.61	5.8765	39.0	15.5204	0.0	1016.16	Partly cloudy starting in the morning.

96453 rows × 12 columns

Some examples uses of Data Frames Methods

In [24]:
```
# median()

df[["Humidity", "Temperature (C)"]].median()
```

Out[24]:
```
Humidity            0.78
Temperature (C)    12.00
dtype: float64
```

In [26]:
```
# describe ()

df[["Humidity", "Temperature (C)"]].describe()
```

Out[26]:

	Humidity	Temperature (C)
count	96453.000000	96453.000000
mean	0.734899	11.932678
std	0.195473	9.551546
min	0.000000	-21.822222
25%	0.600000	4.688889
50%	0.780000	12.000000
75%	0.890000	18.838889
max	1.000000	39.905556

In [27]:
```
# Aggregating statistics for given columns can be defined using the DataFrame.agg() method:
df.agg(
        {
          "Humidity": ["min", "max", "median", "skew"],
            "Temperature (C)": ["min", "max", "median", "mean"],
        })
```

Out[27]:

	Humidity	Temperature (C)
min	0.00000	-21.822222
max	1.00000	39.905556
median	0.78000	12.000000
skew	-0.71588	NaN
mean	NaN	11.932678

Selecting a Column in a Data Frame

In [28]:
```
df["Temperature (C)"]
```

Out[28]:
```
0         9.472222
1         9.355556
2         9.377778
3         8.288889
4         8.755556
           ...
96448    26.016667
96449    24.583333
96450    22.038889
96451    21.522222
96452    20.438889
Name: Temperature (C), Length: 96453, dtype: float64
```

In [31]:
```
df.Humidity
```

Out[31]:
```
0        0.89
1        0.86
2        0.89
3        0.83
4        0.83
         ...
96448    0.43
96449    0.48
96450    0.56
96451    0.60
96452    0.61
Name: Humidity, Length: 96453, dtype: float64
```

Data Frames Groupby Method

In [34]:
```
# Using "group by" methodwecan:
#  • Split the data into groups based on somecriteria
#  • Calculate statistics (or apply a function) to each group


df_Summary=df.groupby(['Summary']) # return the object


df_Summary.mean()   # iterate the object according to the mean
```

Out[34]:

Summary	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)
Breezy	7.922016	3.387654	0.637778	32.143948	233.018519	9.577115	0.0	563.917593
Breezy and Dry	21.111111	21.111111	0.260000	33.810000	240.000000	9.982000	0.0	1021.600000
Breezy and Foggy	-0.510317	-7.403492	0.938571	33.477880	160.628571	1.621960	0.0	1008.934000
Breezy and Mostly Cloudy	11.093411	8.680588	0.637054	33.386345	227.639535	11.478302	0.0	1000.622984
Breezy and Overcast	7.241614	3.492235	0.763144	33.037566	213.526515	11.067012	0.0	1002.114924
Breezy and Partly Cloudy	12.492761	9.989349	0.545803	33.532796	259.282383	11.326058	0.0	996.398212
Clear	11.925109	11.040338	0.729708	8.141352	179.180257	11.441788	0.0	951.763532
Dangerously Windy and Partly Cloudy	8.944444	3.483333	0.490000	63.852600	307.000000	11.447100	0.0	1009.050000
Drizzle	10.847578	10.011681	0.867949	10.356428	177.307692	8.069815	0.0	1014.931538
Dry	29.083660	28.273529	0.230294	14.713979	230.294118	10.250965	0.0	1016.391765
Dry and Mostly Cloudy	26.838492	25.929365	0.242143	13.667750	187.785714	10.115400	0.0	1014.872143
Dry and Partly Cloudy	26.605749	25.982235	0.240814	12.304519	224.465116	10.987501	0.0	1017.242558
Foggy	1.464035	-0.210419	0.950765	7.171649	168.668439	1.551411	0.0	1007.475207
Humid and Mostly Cloudy	20.886389	20.886389	0.874250	10.058877	153.425000	9.732852	0.0	1012.887250
Humid and Overcast	21.515079	21.515079	0.881429	9.740500	138.857143	9.052800	0.0	1014.550000
Humid and Partly Cloudy	21.568301	21.568301	0.848824	9.938435	201.647059	10.633576	0.0	1011.974118
Light Rain	10.021517	8.560317	0.888095	14.673489	180.761905	6.657989	0.0	1011.054762
Mostly Cloudy	12.629334	11.624994	0.725069	11.418404	192.049299	11.117234	0.0	1010.840591
Overcast	7.516502	5.789636	0.837232	12.027738	183.532747	9.275112	0.0	1005.632402
Partly Cloudy	16.024782	15.394033	0.648571	10.115130	190.161094	11.811517	0.0	1013.079063
Rain	10.096111	9.607222	0.947000	5.797610	211.800000	1.779050	0.0	1017.318000
Windy	6.804861	2.009028	0.572500	42.165900	319.750000	10.712537	0.0	127.376250

Summary	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)
Windy and Dry	27.222222	26.344444	0.240000	40.250000	150.000000	9.982000	0.0	1020.200000
Windy and Foggy	11.876389	9.769444	0.900000	44.178400	155.000000	1.903825	0.0	1011.975000
Windy and Mostly Cloudy	11.834603	9.754286	0.600000	43.117640	261.428571	11.159600	0.0	979.825143
Windy and Overcast	7.932963	3.696543	0.708667	43.378409	244.311111	9.806331	0.0	1006.446000
Windy and Partly Cloudy	9.968076	6.551244	0.528806	44.610937	295.119403	11.484106	0.0	953.291194

In [39]:
```
# df.groupby(['Summary'])["Temperature (C)"].mean()

# corrected code
df.groupby('Summary')[["Temperature (C)"]].mean()
```

Out[39]:

Summary	Temperature (C)
Breezy	7.922016
Breezy and Dry	21.111111
Breezy and Foggy	-0.510317
Breezy and Mostly Cloudy	11.093411
Breezy and Overcast	7.241614
Breezy and Partly Cloudy	12.492761
Clear	11.925109
Dangerously Windy and Partly Cloudy	8.944444
Drizzle	10.847578
Dry	29.083660
Dry and Mostly Cloudy	26.838492
Dry and Partly Cloudy	26.605749
Foggy	1.464035
Humid and Mostly Cloudy	20.886389
Humid and Overcast	21.515079
Humid and Partly Cloudy	21.568301
Light Rain	10.021517
Mostly Cloudy	12.629334
Overcast	7.516502
Partly Cloudy	16.024782
Rain	10.096111
Windy	6.804861
Windy and Dry	27.222222

Summary	Temperature (C)
Windy and Foggy	11.876389
Windy and Mostly Cloudy	11.834603
Windy and Overcast	7.932963
Windy and Partly Cloudy	9.968076

In [41]:
```
df.groupby(['Summary'], sort = False )[["Temperature (C)"]].mean()
```

Out[41]:

Summary	Temperature (C)
Breezy	7.922016
Breezy and Dry	21.111111
Breezy and Foggy	-0.510317
Breezy and Mostly Cloudy	11.093411
Breezy and Overcast	7.241614
Breezy and Partly Cloudy	12.492761
Clear	11.925109
Dangerously Windy and Partly Cloudy	8.944444
Drizzle	10.847578
Dry	29.083660
Dry and Mostly Cloudy	26.838492
Dry and Partly Cloudy	26.605749
Foggy	1.464035
Humid and Mostly Cloudy	20.886389
Humid and Overcast	21.515079
Humid and Partly Cloudy	21.568301
Light Rain	10.021517
Mostly Cloudy	12.629334
Overcast	7.516502
Partly Cloudy	16.024782
Rain	10.096111
Windy	6.804861
Windy and Dry	27.222222

	Temperature (C)
Summary	
Windy and Foggy	11.876389
Windy and Mostly Cloudy	11.834603
Windy and Overcast	7.932963
Windy and Partly Cloudy	9.968076

Data Frame : Filtering

In [50]:
```
#Calculate mean salary for each professor rank:

df_sub = df[df['Temperature (C)'] > 10.847578]  # Ensure no unnecessary spaces before this line
df_sub
```

Out[50]:

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
9	2006-04-01 09:00:00.000 +0200	Partly Cloudy	rain	13.772222	13.772222	0.72	12.5258	279.0	9.9820	0.0	1017.22	Partly cloudy throughout the day.
10	2006-04-01 10:00:00.000 +0200	Partly Cloudy	rain	16.016667	16.016667	0.67	17.5651	290.0	11.2056	0.0	1017.42	Partly cloudy throughout the day.
11	2006-04-01 11:00:00.000 +0200	Partly Cloudy	rain	17.144444	17.144444	0.54	19.7869	316.0	11.4471	0.0	1017.74	Partly cloudy throughout the day.
12	2006-04-01 12:00:00.000 +0200	Partly Cloudy	rain	17.800000	17.800000	0.55	21.9443	281.0	11.2700	0.0	1017.59	Partly cloudy throughout the day.
13	2006-04-01 13:00:00.000 +0200	Partly Cloudy	rain	17.333333	17.333333	0.51	20.6885	289.0	11.2700	0.0	1017.48	Partly cloudy throughout the day.
...	...	...	...	...	...	...	...	...	...	...	...	...
96448	2016-09-09 19:00:00.000 +0200	Partly Cloudy	rain	26.016667	26.016667	0.43	10.9963	31.0	16.1000	0.0	1014.36	Partly cloudy starting in the morning.
96449	2016-09-09 20:00:00.000 +0200	Partly Cloudy	rain	24.583333	24.583333	0.48	10.0947	20.0	15.5526	0.0	1015.16	Partly cloudy starting in the morning.
96450	2016-09-09 21:00:00.000 +0200	Partly Cloudy	rain	22.038889	22.038889	0.56	8.9838	30.0	16.1000	0.0	1015.66	Partly cloudy starting in the morning.
96451	2016-09-09 22:00:00.000 +0200	Partly Cloudy	rain	21.522222	21.522222	0.60	10.5294	20.0	16.1000	0.0	1015.95	Partly cloudy starting in the morning.
96452	2016-09-09 23:00:00.000 +0200	Partly Cloudy	rain	20.438889	20.438889	0.61	5.8765	39.0	15.5204	0.0	1016.16	Partly cloudy starting in the morning.

51776 rows × 12 columns

In [53]:
```
df_ = df[df['Temperature (C)']==13.77222]  # Ensure no unnecessary spaces before this line
df_
```

Out[53]:

Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary

In [54]:
```
# value_counts()

#The value_counts() method counts the number of records for each category in a column.

df["Summary"].value_counts()
```

Out[54]:

```
Partly Cloudy                          31733
Mostly Cloudy                          28094
Overcast                               16597
Clear                                  10890
Foggy                                   7148
Breezy and Overcast                      528
Breezy and Mostly Cloudy                 516
Breezy and Partly Cloudy                 386
Dry and Partly Cloudy                     86
Windy and Partly Cloudy                   67
Light Rain                                63
Breezy                                    54
Windy and Overcast                        45
Humid and Mostly Cloudy                   40
Drizzle                                   39
Breezy and Foggy                          35
Windy and Mostly Cloudy                   35
Dry                                       34
Humid and Partly Cloudy                   17
Dry and Mostly Cloudy                     14
Rain                                      10
Windy                                      8
Humid and Overcast                         7
Windy and Foggy                            4
Windy and Dry                              1
Dangerously Windy and Partly Cloudy        1
Breezy and Dry                             1
Name: Summary, dtype: int64
```

In [56]:

```
# The function is a shortcut, as it is actually a groupby operation in combination with counting of the number of records within

df.groupby("Summary")["Summary"].count()
```

Out[56]:

```
Summary
Breezy                                    54
Breezy and Dry                             1
Breezy and Foggy                          35
Breezy and Mostly Cloudy                 516
Breezy and Overcast                      528
Breezy and Partly Cloudy                 386
Clear                                  10890
Dangerously Windy and Partly Cloudy        1
Drizzle                                   39
Dry                                       34
Dry and Mostly Cloudy                     14
Dry and Partly Cloudy                     86
Foggy                                   7148
Humid and Mostly Cloudy                   40
Humid and Overcast                         7
Humid and Partly Cloudy                   17
Light Rain                                63
Mostly Cloudy                          28094
Overcast                               16597
Partly Cloudy                          31733
Rain                                      10
Windy                                      8
Windy and Dry                              1
Windy and Foggy                            4
Windy and Mostly Cloudy                   35
Windy and Overcast                        45
Windy and Partly Cloudy                   67
Name: Summary, dtype: int64
```

Data Frames : method loc

In [57]:

```
# Select rows by their labels :

df.loc[10 :20 ,["Humidity","Daily Summary","Loud Cover"]]
```

Out[57]:

	Humidity	Daily Summary	Loud Cover
10	0.67	Partly cloudy throughout the day.	0.0
11	0.54	Partly cloudy throughout the day.	0.0
12	0.55	Partly cloudy throughout the day.	0.0
13	0.51	Partly cloudy throughout the day.	0.0
14	0.47	Partly cloudy throughout the day.	0.0
15	0.46	Partly cloudy throughout the day.	0.0
16	0.60	Partly cloudy throughout the day.	0.0
17	0.63	Partly cloudy throughout the day.	0.0
18	0.69	Partly cloudy throughout the day.	0.0
19	0.70	Partly cloudy throughout the day.	0.0
20	0.77	Partly cloudy throughout the day.	0.0

In [58]:

```
# Select rows by their labels :

df.iloc[10:20,[0,2,4,5,6]]
```

Out[58]:

	Formatted Date	Precip Type	Apparent Temperature (C)	Humidity	Wind Speed (km/h)
10	2006-04-01 10:00:00.000 +0200	rain	16.016667	0.67	17.5651
11	2006-04-01 11:00:00.000 +0200	rain	17.144444	0.54	19.7869
12	2006-04-01 12:00:00.000 +0200	rain	17.800000	0.55	21.9443
13	2006-04-01 13:00:00.000 +0200	rain	17.333333	0.51	20.6885
14	2006-04-01 14:00:00.000 +0200	rain	18.877778	0.47	15.3755
15	2006-04-01 15:00:00.000 +0200	rain	18.911111	0.46	10.4006
16	2006-04-01 16:00:00.000 +0200	rain	15.388889	0.60	14.4095
17	2006-04-01 17:00:00.000 +0200	rain	15.550000	0.63	11.1573
18	2006-04-01 18:00:00.000 +0200	rain	14.255556	0.69	8.5169
19	2006-04-01 19:00:00.000 +0200	rain	13.144444	0.70	7.6314

Data Frames : method iloc (Summary)

In [61]:
```
df.iloc[0]  #Firstrowofadataframe
# df.iloc[5]  #(i+1)throw
# df.iloc[-1] #Last row
```

Out[61]:
```
Formatted Date                  2006-04-01 00:00:00.000 +0200
Summary                                          Partly Cloudy
Precip Type                                               rain
Temperature (C)                                       9.472222
Apparent Temperature (C)                              7.388889
Humidity                                                  0.89
Wind Speed (km/h)                                        14.1197
Wind Bearing (degrees)                                     251.0
Visibility (km)                                          15.8263
Loud Cover                                                   0.0
Pressure (millibars)                                     1015.13
Daily Summary                  Partly cloudy throughout the day.
Name: 0, dtype: object
```

In [64]:
```
df.iloc[:,0] # First column
# df.iloc[:,-1]# Last column
```

Out[64]:
```
0        2006-04-01 00:00:00.000 +0200
1        2006-04-01 01:00:00.000 +0200
2        2006-04-01 02:00:00.000 +0200
3        2006-04-01 03:00:00.000 +0200
4        2006-04-01 04:00:00.000 +0200
                    ...
96448    2016-09-09 19:00:00.000 +0200
96449    2016-09-09 20:00:00.000 +0200
96450    2016-09-09 21:00:00.000 +0200
96451    2016-09-09 22:00:00.000 +0200
96452    2016-09-09 23:00:00.000 +0200
Name: Formatted Date, Length: 96453, dtype: object
```

In [65]:
```
df.iloc[0:7]          #First7rows
df.iloc[:,0:2]        #First2columns
df.iloc[1:3,0:2]      #Secondthroughthirdrowsandfirst2columns
df.iloc[[0,5],[1,3]] #1st  and6th rowsand2nd and4th columns
```

Out[65]:

	Summary	Temperature (C)
0	Partly Cloudy	9.472222
5	Partly Cloudy	9.222222

Data Frames : Sorting

In [ ]:
```
# We can sort the data by a value in the column. By default the sorting will occur in
# ascending order and a new dataframeis return.
```

In [66]:
```
 # Create a new data frame from the original sorted by the column Salary
df_sorted = df.sort_values( by ='Humidity')
df_sorted.head()
```

Out[66]:

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
54858	2012-02-10 18:00:00.000 +0100	Foggy	snow	-15.000000	-15.000000	0.0	0.00	0.0	2.576	0.0	1034.4	Foggy starting in the morning continuing until...
55469	2012-02-08 05:00:00.000 +0100	Foggy	snow	-17.777778	-17.777778	0.0	4.83	200.0	1.932	0.0	1033.9	Foggy until morning.
55412	2012-02-05 20:00:00.000 +0100	Foggy	snow	-17.777778	-23.616667	0.0	8.05	320.0	2.576	0.0	1032.2	Foggy in the evening.
55352	2012-02-03 08:00:00.000 +0100	Overcast	snow	-12.222222	-21.144444	0.0	22.54	20.0	11.270	0.0	1029.7	Foggy starting in the morning continuing until...
55350	2012-02-03 06:00:00.000 +0100	Overcast	snow	-12.222222	-20.827778	0.0	20.93	30.0	9.982	0.0	1029.4	Foggy starting in the morning continuing until...

In [71]:
```
df_sorted = df.sort_values(by =['Humidity','Visibility (km)'], ascending = [True, False])
df_sorted.head(10)
```

Out[71]:

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
55352	2012-02-03 08:00:00.000 +0100	Overcast	snow	-12.222222	-21.144444	0.0	22.5400	20.0	11.270	0.0	1029.70	Foggy starting in the morning continuing until...
19958	2008-02-17 14:00:00.000 +0100	Partly Cloudy	snow	-1.111111	-1.111111	0.0	4.4275	12.0	9.982	0.0	1043.04	Partly cloudy starting in the morning continui...
28101	2009-12-20 21:00:00.000 +0100	Clear	snow	-15.000000	-15.000000	0.0	3.2200	250.0	9.982	0.0	1015.10	Mostly cloudy starting overnight continuing un...
28103	2009-12-20 23:00:00.000 +0100	Clear	snow	-15.555556	-20.150000	0.0	6.4400	160.0	9.982	0.0	1015.90	Mostly cloudy starting overnight continuing un...
28110	2009-12-21 06:00:00.000 +0100	Clear	snow	-13.888889	-23.266667	0.0	22.5400	160.0	9.982	0.0	1016.80	Mostly cloudy starting in the morning.
54872	2012-02-11 08:00:00.000 +0100	Overcast	snow	-15.000000	-22.738889	0.0	14.4900	30.0	9.982	0.0	1029.80	Foggy starting overnight continuing until morn...
54873	2012-02-11 09:00:00.000 +0100	Overcast	snow	-13.888889	-20.350000	0.0	11.2700	40.0	9.982	0.0	1029.90	Foggy starting overnight continuing until morn...
55086	2012-02-02 06:00:00.000 +0100	Partly Cloudy	snow	-12.777778	-19.011111	0.0	11.2700	20.0	9.982	0.0	1027.60	Mostly cloudy throughout the day.
55088	2012-02-02 08:00:00.000	Mostly Cloudy	snow	-12.777778	-19.011111	0.0	11.2700	20.0	9.982	0.0	1028.10	Mostly cloudy throughout

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
	+0100											the day.
55349	2012-02-03 05:00:00.000 +0100	Overcast	snow	-12.222222	-20.133333	0.0	17.7100	30.0	9.982	0.0	1030.00	Foggy starting in the morning continuing until...

Missing Values

Drop missing observations

In [ ]:
```
# dataframe.dropna(axis, how, thresh, subset, inplace)
```

In [73]:
```
import pandas as pd
df = pd.DataFrame({'A': [1, 2, None], 'B': [None, 3, 4]})
df_cleaned = df.dropna()
print(df_cleaned)
```

```
     A    B
1  2.0  3.0
```

Drop observations where all cells is NA

In [72]:
```
df = pd.DataFrame({'A': [1, None, None], 'B': [None, None, None]})
df_cleaned = df.dropna(how='all')
print(df_cleaned) # not delete column
```

```
     A     B
0  1.0  None
```

Drop column if all the values are missing

In [74]:
```
df = pd.DataFrame({'A': [1, 2, 3], 'B': [None, None, None]})
df_cleaned = df.dropna(axis=1, how='all')
print(df_cleaned)
```

```
   A
0  1
1  2
2  3
```

Drop rows that have less than 2 non-null values

In [76]:
```
data = {
    'A': [1, np.nan, 3, 4, np.nan],
    'B': [np.nan, 2, 3, np.nan, np.nan],
    'C': [1, 2, np.nan, 4, 5],
    'D': [np.nan, np.nan, np.nan, np.nan, 5]
}

# Drop rows that have less than 2 non-null values
df3 = df.dropna(thresh=2)
print("\nAfter dropping rows with less than 2 non-null values:")
print(df3)
```

```
After dropping rows with less than 2 non-null values:
     A    B  C
0  1.0  NaN  1
1  2.0  2.0  2
2  NaN  3.0  3
3  4.0  4.0  4
4  5.0  5.0  5
```

Filling Missing Values with Different Strategies

In [77]:
```
# Filling NaN with 0
df_filled = df.fillna(0)
print("\nAfter replacing missing values with 0:")
print(df_filled)
```

```
After replacing missing values with 0:
     A    B  C
0  1.0  0.0  1
1  2.0  2.0  2
2  0.0  3.0  3
3  4.0  4.0  4
4  5.0  5.0  5
```

In [78]:
```
# Filling missing values with the column mean
df_filled_mean = df.apply(lambda col: col.fillna(col.mean()), axis=0)
print("\nAfter filling missing values with column mean:")
print(df_filled_mean)
```

```
After filling missing values with column mean:
     A    B  C
0  1.0  3.5  1
1  2.0  2.0  2
2  3.0  3.0  3
3  4.0  4.0  4
4  5.0  5.0  5
```

In [79]:
```
# Filling missing values with forward fill method (propagates the last valid value)
df_ffill = df.fillna(method='ffill')
print("\nAfter forward fill:")
print(df_ffill)
```

```
After forward fill:
     A    B  C
0  1.0  NaN  1
1  2.0  2.0  2
2  2.0  3.0  3
3  4.0  4.0  4
4  5.0  5.0  5
```

In [80]:
```
# Filling missing values with backward fill method
df_bfill = df.fillna(method='bfill')
print("\nAfter backward fill:")
print(df_bfill)
```

```
After backward fill:
     A    B  C
0  1.0  2.0  1
1  2.0  2.0  2
2  4.0  3.0  3
3  4.0  4.0  4
4  5.0  5.0  5
```

returns True if the value is missing

In [81]:
```
# Selecting rows where column 'B' has NaN values
df_null_B = df[df['B'].isnull()]
```

```
print("\nRows where 'B' is NaN:")
print(df_null_B)
```

```
Rows where 'B' is NaN:
     A   B  C
0  1.0 NaN  1
```

In [82]:
```
# Selecting rows where column 'C' is NOT NaN
df_notnull_C = df[df['C'].notnull()]
print("\nRows where 'C' is NOT NaN:")
print(df_notnull_C)
```

```
Rows where 'C' is NOT NaN:
     A    B  C
0  1.0  NaN  1
1  2.0  2.0  2
2  NaN  3.0  3
3  4.0  4.0  4
4  5.0  5.0  5
```

In [83]:
```
# Counting the number of missing values per column
missing_counts = df.isnull().sum()
print("\nCount of missing values per column:")
print(missing_counts)
```

```
Count of missing values per column:
A    1
B    1
C    0
dtype: int64
```

In [ ]: