Exploratory Data Analysis-1

In [1]:

```
#Load the libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [2]:

```
data1=pd.read_csv("C:/Users/Ashraf/Documents/Datafiles/data_clean.csv")
```

In [3]:

```
data1.head()
```

Out[3]:

	Unnamed: 0	Ozone	Solar.R	Wind	Temp C	Month	Day	Year	Temp	Weather
0	1	41.0	190.0	7.4	67	5	1	2010	67	S
1	2	36.0	118.0	8.0	72	5	2	2010	72	C
2	3	12.0	149.0	12.6	74	5	3	2010	74	PS
3	4	18.0	313.0	11.5	62	5	4	2010	62	S
4	5	NaN	NaN	14.3	56	5	5	2010	56	S

In [4]:

```
data1.tail()
```

Out[4]:

	Unnamed: 0	Ozone	Solar.R	Wind	Temp C	Month	Day	Year	Temp	Weather
153	154	41.0	190.0	7.4	67	5	1	2010	67	C
154	155	30.0	193.0	6.9	70	9	26	2010	70	PS
155	156	NaN	145.0	13.2	77	9	27	2010	77	S
156	157	14.0	191.0	14.3	75	9	28	2010	75	S
157	158	18.0	131.0	8.0	76	9	29	2010	76	C

In [5]:

```
#Data Structure
type(data1)
data1.shape
```

Out[5]:

```
(158, 10)
```

In [6]:

```
#data types
data1.dtypes
```

Out[6]:

```
Unnamed: 0      int64
Ozone         float64
Solar.R       float64
Wind          float64
Temp C         object
Month          object
Day             int64
Year            int64
Temp            int64
Weather        object
dtype: object
```

Data type conversion

In [7]:

```
data1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 158 entries, 0 to 157
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Unnamed: 0  158 non-null    int64  
 1   Ozone       120 non-null    float64
 2   Solar.R     151 non-null    float64
 3   Wind        158 non-null    float64
 4   Temp C      158 non-null    object 
 5   Month       158 non-null    object 
 6   Day         158 non-null    int64  
 7   Year        158 non-null    int64  
 8   Temp        158 non-null    int64  
 9   Weather     155 non-null    object 
dtypes: float64(3), int64(4), object(3)
memory usage: 12.5+ KB
```

In [8]:

```
data1
```

Out[8]:

	Unnamed: 0	Ozone	Solar.R	Wind	Temp C	Month	Day	Year	Temp	Weather
0	1	41.0	190.0	7.4	67	5	1	2010	67	S
1	2	36.0	118.0	8.0	72	5	2	2010	72	C
2	3	12.0	149.0	12.6	74	5	3	2010	74	PS
3	4	18.0	313.0	11.5	62	5	4	2010	62	S
4	5	NaN	NaN	14.3	56	5	5	2010	56	S
...	...	...	...	...	...	...	...	...	...	...
153	154	41.0	190.0	7.4	67	5	1	2010	67	C
154	155	30.0	193.0	6.9	70	9	26	2010	70	PS
155	156	NaN	145.0	13.2	77	9	27	2010	77	S
156	157	14.0	191.0	14.3	75	9	28	2010	75	S
157	158	18.0	131.0	8.0	76	9	29	2010	76	C

158 rows × 10 columns

In [9]:

```
data2=data1.iloc[:,1:]
data2
```

Out[9]:

	Ozone	Solar.R	Wind	Temp C	Month	Day	Year	Temp	Weather
0	41.0	190.0	7.4	67	5	1	2010	67	S
1	36.0	118.0	8.0	72	5	2	2010	72	C
2	12.0	149.0	12.6	74	5	3	2010	74	PS
3	18.0	313.0	11.5	62	5	4	2010	62	S
4	NaN	NaN	14.3	56	5	5	2010	56	S
...	...	...	...	...	...	...	...	...	...
153	41.0	190.0	7.4	67	5	1	2010	67	C
154	30.0	193.0	6.9	70	9	26	2010	70	PS
155	NaN	145.0	13.2	77	9	27	2010	77	S
156	14.0	191.0	14.3	75	9	28	2010	75	S
157	18.0	131.0	8.0	76	9	29	2010	76	C

158 rows × 9 columns

In [10]:

```
#The method .copy() is used here so that any changes made in new DataFrame don't get reflec
data3=data2.copy()
```

In [11]:

```
data3['Month']=pd.to_numeric(data3['Month'],errors='coerce')
data3['Temp C']=pd.to_numeric(data3['Temp C'],errors='coerce')# coerce will introduce NA va
data3['Weather']=data3['Weather'].astype('category')           #data['Wind']=data['Wind'].a
```

In [12]:

```
data3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 158 entries, 0 to 157
Data columns (total 9 columns):
 #   Column   Non-Null Count  Dtype   
---  ------   --------------  -----   
 0   Ozone    120 non-null    float64 
 1   Solar.R  151 non-null    float64 
 2   Wind     158 non-null    float64 
 3   Temp C   157 non-null    float64 
 4   Month    157 non-null    float64 
 5   Day      158 non-null    int64   
 6   Year     158 non-null    int64   
 7   Temp     158 non-null    int64   
 8   Weather  155 non-null    category
dtypes: category(1), float64(5), int64(3)
memory usage: 10.3 KB
```

In [13]:

```
#Duplicates

#Count of duplicated rows
data3[data3.duplicated()].shape
```

Out[13]:

```
(1, 9)
```

In [14]:

```
data3
```

Out[14]:

	Ozone	Solar.R	Wind	Temp C	Month	Day	Year	Temp	Weather
0	41.0	190.0	7.4	67.0	5.0	1	2010	67	S
1	36.0	118.0	8.0	72.0	5.0	2	2010	72	C
2	12.0	149.0	12.6	74.0	5.0	3	2010	74	PS
3	18.0	313.0	11.5	62.0	5.0	4	2010	62	S
4	NaN	NaN	14.3	56.0	5.0	5	2010	56	S
...	...	...	...	...	...	...	...	...	...
153	41.0	190.0	7.4	67.0	5.0	1	2010	67	C
154	30.0	193.0	6.9	70.0	9.0	26	2010	70	PS
155	NaN	145.0	13.2	77.0	9.0	27	2010	77	S
156	14.0	191.0	14.3	75.0	9.0	28	2010	75	S
157	18.0	131.0	8.0	76.0	9.0	29	2010	76	C

158 rows × 9 columns

In [15]:

```
#Print the duplicated rows
data3[data3.duplicated()]
```

Out[15]:

	Ozone	Solar.R	Wind	Temp C	Month	Day	Year	Temp	Weather
156	14.0	191.0	14.3	75.0	9.0	28	2010	75	S

In [16]:

```
data_cleaned1=data3.drop_duplicates()
```

In [17]:

```
data_cleaned1.shape
```

Out[17]:

```
(157, 9)
```

In [18]:

```
#Drop columns
data_cleaned2=data_cleaned1.drop('Temp C',axis=1)
data_cleaned2
```

Out[18]:

	Ozone	Solar.R	Wind	Month	Day	Year	Temp	Weather
0	41.0	190.0	7.4	5.0	1	2010	67	S
1	36.0	118.0	8.0	5.0	2	2010	72	C
2	12.0	149.0	12.6	5.0	3	2010	74	PS
3	18.0	313.0	11.5	5.0	4	2010	62	S
4	NaN	NaN	14.3	5.0	5	2010	56	S
...	...	...	...	...	...	...	...	...
152	20.0	223.0	11.5	9.0	30	2010	68	S
153	41.0	190.0	7.4	5.0	1	2010	67	C
154	30.0	193.0	6.9	9.0	26	2010	70	PS
155	NaN	145.0	13.2	9.0	27	2010	77	S
157	18.0	131.0	8.0	9.0	29	2010	76	C

157 rows × 8 columns

In [19]:

```
#Rename the columns
#rename the Solar column
data_cleaned3 = data_cleaned2.rename({'Solar.R': 'Solar'}, axis=1)
data_cleaned3
```

Out[19]:

	Ozone	Solar	Wind	Month	Day	Year	Temp	Weather
0	41.0	190.0	7.4	5.0	1	2010	67	S
1	36.0	118.0	8.0	5.0	2	2010	72	C
2	12.0	149.0	12.6	5.0	3	2010	74	PS
3	18.0	313.0	11.5	5.0	4	2010	62	S
4	NaN	NaN	14.3	5.0	5	2010	56	S
...	...	...	...	...	...	...	...	...
152	20.0	223.0	11.5	9.0	30	2010	68	S
153	41.0	190.0	7.4	5.0	1	2010	67	C
154	30.0	193.0	6.9	9.0	26	2010	70	PS
155	NaN	145.0	13.2	9.0	27	2010	77	S
157	18.0	131.0	8.0	9.0	29	2010	76	C

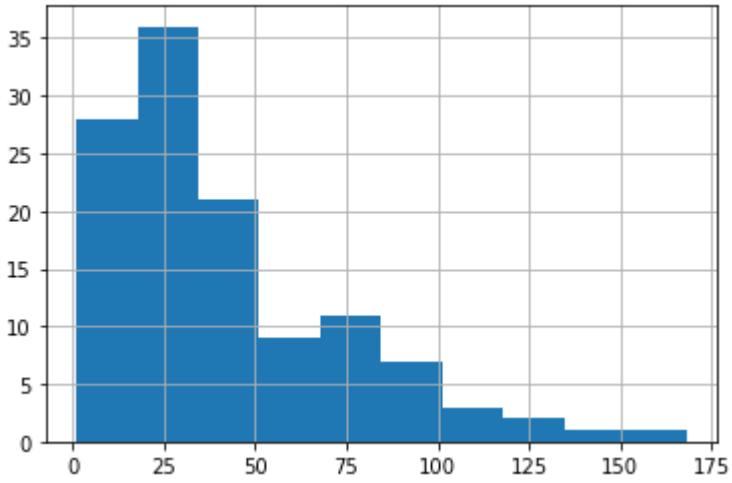
157 rows × 8 columns

In [20]:

```
#Outlier Detection
# histogram of Ozone
data_cleaned3['Ozone'].hist()
```

Out[20]:

```
<AxesSubplot:>
```

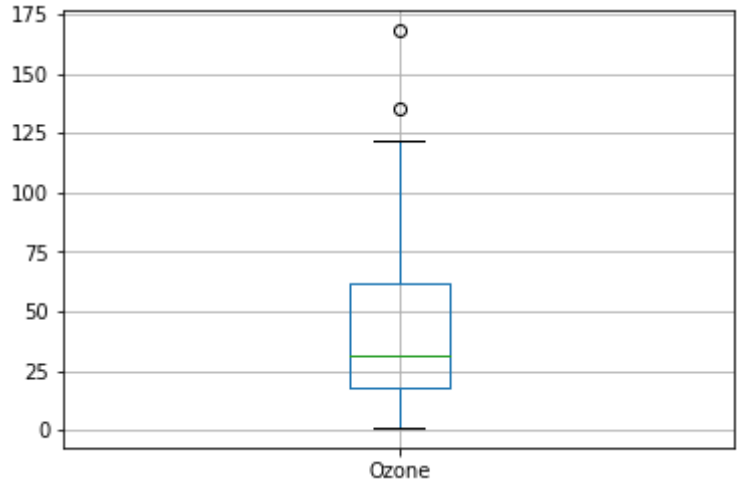


In [21]:

```
#Box plot
data_cleaned3.boxplot(column=['Ozone'])
```

Out[21]:

<AxesSubplot:>



In [22]:

```
#Descriptive stat
data_cleaned3['Ozone'].describe()
```

Out[22]:

```
count    119.000000
mean      41.815126
std       32.659249
min        1.000000
25%       18.000000
50%       31.000000
75%       62.000000
max      168.000000
Name: Ozone, dtype: float64
```

In [23]:

```
data_cleaned3
```

Out[23]:

	Ozone	Solar	Wind	Month	Day	Year	Temp	Weather
0	41.0	190.0	7.4	5.0	1	2010	67	S
1	36.0	118.0	8.0	5.0	2	2010	72	C
2	12.0	149.0	12.6	5.0	3	2010	74	PS
3	18.0	313.0	11.5	5.0	4	2010	62	S
4	NaN	NaN	14.3	5.0	5	2010	56	S
...	...	...	...	...	...	...	...	...
152	20.0	223.0	11.5	9.0	30	2010	68	S
153	41.0	190.0	7.4	5.0	1	2010	67	C
154	30.0	193.0	6.9	9.0	26	2010	70	PS
155	NaN	145.0	13.2	9.0	27	2010	77	S
157	18.0	131.0	8.0	9.0	29	2010	76	C

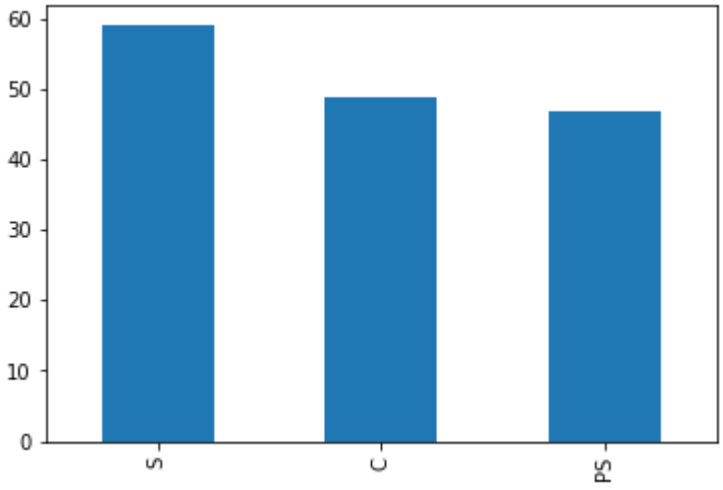
157 rows × 8 columns

In [24]:

```
#Bar plot
data3['Weather'].value_counts().plot.bar()
```

Out[24]:

```
<AxesSubplot:>
```



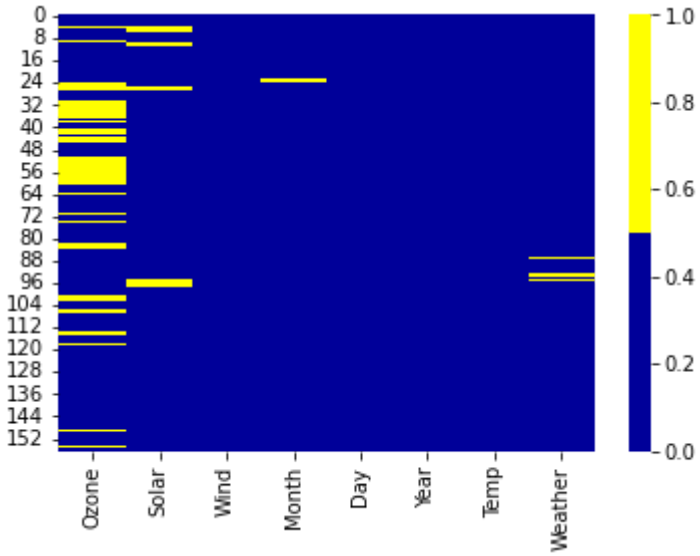
Missing Values and Imputation

In [25]:

```
import seaborn as sns
cols = data_cleaned3.columns
colours = ['#000099', '#ffff00'] # specify the colours - yellow is missing. blue is not mis
sns.heatmap(data_cleaned3[cols].isnull(),
            cmap=sns.color_palette(colours))
```

Out[25]:

<AxesSubplot:>



In [26]:

```
data_cleaned3[data_cleaned3.isnull().any(axis=1)].head()
```

Out[26]:

	Ozone	Solar	Wind	Month	Day	Year	Temp	Weather
4	NaN	NaN	14.3	5.0	5	2010	56	S
5	28.0	NaN	14.9	5.0	6	2010	66	C
9	NaN	194.0	8.6	5.0	10	2010	69	S
10	7.0	NaN	6.9	5.0	11	2010	74	C
23	32.0	92.0	12.0	NaN	24	2010	61	C

In [27]:

```
data_cleaned3.isnull().sum()
```

Out[27]:

```
Ozone      38
Solar       7
Wind        0
Month       1
Day         0
Year        0
Temp        0
Weather     3
dtype: int64
```

In [28]:

```
#Mean Imputation
mean = data_cleaned3['Ozone'].mean()
print(mean)
```

```
41.81512605042017
```

In [29]:

```
data_cleaned3['Ozone'] = data_cleaned3['Ozone'].fillna(mean)
```

In [30]:

```
data_cleaned3
```

Out[30]:

	Ozone	Solar	Wind	Month	Day	Year	Temp	Weather
0	41.000000	190.0	7.4	5.0	1	2010	67	S
1	36.000000	118.0	8.0	5.0	2	2010	72	C
2	12.000000	149.0	12.6	5.0	3	2010	74	PS
3	18.000000	313.0	11.5	5.0	4	2010	62	S
4	41.815126	NaN	14.3	5.0	5	2010	56	S
...	...	...	...	...	...	...	...	...
152	20.000000	223.0	11.5	9.0	30	2010	68	S
153	41.000000	190.0	7.4	5.0	1	2010	67	C
154	30.000000	193.0	6.9	9.0	26	2010	70	PS
155	41.815126	145.0	13.2	9.0	27	2010	77	S
157	18.000000	131.0	8.0	9.0	29	2010	76	C

157 rows × 8 columns

In [31]:

```
#Missing value imputation for categorical vlaue
#Get the object columns
obj_columns=data_cleaned3[['Weather']]
```

In [32]:

```
obj_columns.isnull().sum()
```

Out[32]:

```
Weather    3
dtype: int64
```

In [33]:

```
#Missing value imputation for categorical vlaue
obj_columns=obj_columns.fillna(obj_columns.mode().iloc[0])
```

In [34]:

```
obj_columns.isnull().sum()
```

Out[34]:

```
Weather    0
dtype: int64
```

In [35]:

```
data_cleaned3.shape
```

Out[35]:

```
(157, 8)
```

In [36]:

```
obj_columns.shape
```

Out[36]:

```
(157, 1)
```

In [37]:

```
#Join the data set with imputed object dataset
data_cleaned4=pd.concat([data_cleaned3,obj_columns],axis=1)
```

In [38]:

```
data_cleaned4.isnull().sum()
```

Out[38]:

```
Ozone      0
Solar      7
Wind       0
Month      1
Day        0
Year       0
Temp       0
Weather    3
Weather    0
dtype: int64
```

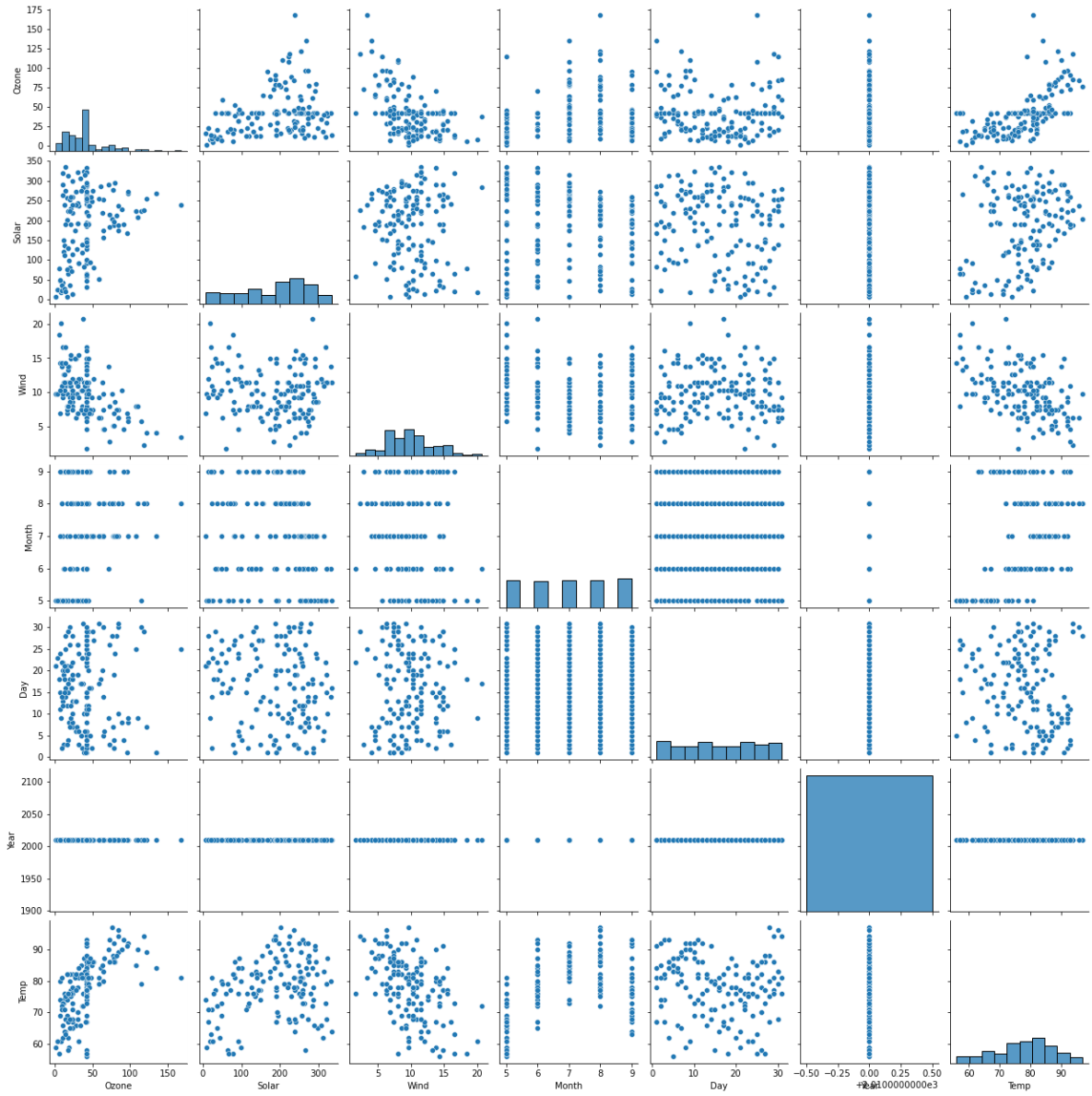
Scatter plot and Correlation analysis

In [39]:

```
# Seaborn visualization library
import seaborn as sns
# Create the default pairplot
sns.pairplot(data_cleaned3)
```

Out[39]:

```
<seaborn.axisgrid.PairGrid at 0x35eadb95b0>
```



In [40]:

```
#Correlation
data_cleaned3.corr()
```

Out[40]:

	Ozone	Solar	Wind	Month	Day	Year	Temp
Ozone	1.000000	0.308687	-0.520004	0.132860	-0.021916	NaN	0.606500
Solar	0.308687	1.000000	-0.057407	-0.094012	-0.155663	NaN	0.273558
Wind	-0.520004	-0.057407	1.000000	-0.166216	0.029900	NaN	-0.441228
Month	0.132860	-0.094012	-0.166216	1.000000	0.050055	NaN	0.398516
Day	-0.021916	-0.155663	0.029900	0.050055	1.000000	NaN	-0.122787
Year	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Temp	0.606500	0.273558	-0.441228	0.398516	-0.122787	NaN	1.000000

Transformations

In [42]:

```
#Dummy Variable

#Creating dummy variable for Weather column
data_cleaned4=pd.get_dummies(data3,columns=['Weather'])
```

In [43]:

```
data_cleaned4
```

Out[43]:

	Ozone	Solar.R	Wind	Temp C	Month	Day	Year	Temp	Weather_C	Weather_PS	Weather
0	41.0	190.0	7.4	67.0	5.0	1	2010	67	0	0	
1	36.0	118.0	8.0	72.0	5.0	2	2010	72	1	0	
2	12.0	149.0	12.6	74.0	5.0	3	2010	74	0	1	
3	18.0	313.0	11.5	62.0	5.0	4	2010	62	0	0	
4	NaN	NaN	14.3	56.0	5.0	5	2010	56	0	0	
...	...	...	...	...	...	...	...	...	...	...	
153	41.0	190.0	7.4	67.0	5.0	1	2010	67	1	0	
154	30.0	193.0	6.9	70.0	9.0	26	2010	70	0	1	
155	NaN	145.0	13.2	77.0	9.0	27	2010	77	0	0	
156	14.0	191.0	14.3	75.0	9.0	28	2010	75	0	0	
157	18.0	131.0	8.0	76.0	9.0	29	2010	76	1	0	

158 rows × 11 columns

In [44]:

```
data_cleaned4=data_cleaned4.dropna()
```

Normalization of the data

In [45]:

```
#Normalization of the data
from numpy import set_printoptions
from sklearn.preprocessing import MinMaxScaler
```

In [46]:

```
data_cleaned4.values
```

Out[46]:

```
array([[ 41. , 190. ,   7.4, ...,   0. ,   0. ,   1. ],
       [ 36. , 118. ,   8. , ...,   1. ,   0. ,   0. ],
       [ 12. , 149. ,  12.6, ...,   0. ,   1. ,   0. ],
       ...,
       [ 30. , 193. ,   6.9, ...,   0. ,   1. ,   0. ],
       [ 14. , 191. ,  14.3, ...,   0. ,   0. ,   1. ],
       [ 18. , 131. ,   8. , ...,   1. ,   0. ,   0. ]])
```

In [47]:

```
array = data_cleaned3.values

scaler = MinMaxScaler(feature_range=(0,1))
rescaledX = scaler.fit_transform(array[:,0:5])

#transformed data
set_printoptions(precision=2)
print(rescaledX[0:5,:])
```

```
[[0.24 0.56 0.3  0.   0.  ]
 [0.21 0.34 0.33 0.   0.03]
 [0.07 0.43 0.57 0.   0.07]
 [0.1  0.94 0.52 0.   0.1 ]
 [0.24  nan 0.66 0.   0.13]]
```

In [49]:

```
# Standardize data (0 mean, 1 stdev)
from sklearn.preprocessing import StandardScaler
```

In [50]:

```
array = data_cleaned4.values
scaler = StandardScaler().fit(array)
rescaledX = scaler.transform(array)

# summarize transformed data
set_printoptions(precision=2)
print(rescaledX[0:5,:])
```

```
[[-0.02  0.05 -0.71 -1.15 -1.53 -1.7   0.   -1.15 -0.64 -0.68  1.28]
 [-0.17 -0.75 -0.54 -0.62 -1.53 -1.59  0.   -0.62  1.57 -0.68 -0.78]
 [-0.9  -0.41  0.77 -0.4  -1.53 -1.48  0.   -0.4  -0.64  1.47 -0.78]
 [-0.72  1.43  0.45 -1.69 -1.53 -1.36  0.   -1.69 -0.64 -0.68  1.28]
 [-0.57  1.27 -0.37 -1.37 -1.53 -1.02  0.   -1.37 -0.64  1.47 -0.78]]
```

Speed up the EDA process

In [ ]:

```
import pandas_profiling as pp
import sweetviz as sv
```

In [ ]:

```
EDA_report= pp.ProfileReport(data)
EDA_report.to_file(output_file='report.html')
```

In [ ]:

```
sweet_report = sv.analyze(data)
sweet_report.show_html('weather_report.html')
```