EDA Processing

In [21]:

```python
import pandas as pd
import numpy as np
import seaborn as sns
```

In [22]:

```python
data = pd.read_csv(r'C:\Users\HOME\Desktop\DSA\Lab6\data_clean.csv')
#data.describe
data
```

Out[22]:

	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 [23]:

```python
data.tail(10)
```

Out[23]:

	Unnamed: 0	Ozone	Solar.R	Wind	Temp C	Month	Day	Year	Temp	Weather
148	149	30.0	193.0	6.9	70	9	26	2010	70	C
149	150	NaN	145.0	13.2	77	9	27	2010	77	PS
150	151	14.0	191.0	14.3	75	9	28	2010	75	S
151	152	18.0	131.0	8.0	76	9	29	2010	76	PS
152	153	20.0	223.0	11.5	68	9	30	2010	68	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

In [24]:

```
type(data)
data.shape
```

Out[24]:

```
(158, 10)
```

In [25]:

```
data.dtypes
```

Out[25]:

```
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 [26]:

```
data.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 [27]:

```
#checking missing values

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

Out[27]:

	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 [28]:

```
# .copy is used so any changes made in this wont be reflected in original data

data = data2.copy()
data
```

Out[28]:

	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 [30]:

```
data['Month'] = pd.to_numeric(data['Month'],errors='coerce')
data['Temp C'] = pd.to_numeric(data['Temp C'],errors='coerce')
data['Weather'] = data['Weather'].astype('category')
```

In [40]:

```
#dropping  temp column
data = data.drop('Temp',axis=1,inplace=True)
data
```

Out[40]:

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

158 rows × 8 columns

In [47]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 158 entries, 0 to 157
Data columns (total 8 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   Weather  155 non-null    category
dtypes: category(1), float64(5), int64(2)
memory usage: 9.1 KB
```

EDA Processing Starts

Finding Duplicates

In [48]:

```
# Identifying duplicate values from data

data[data.duplicated()].shape

#duplicate
```

Out[48]:

```
(1, 8)
```

In [50]:

```
#printing duplicated row
data[data.duplicated()]
```

Out[50]:

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

In [55]:

```
#dropping duplicated rows

data_cleaned1 = data.drop_duplicates()
data_cleaned1.shape
```

Out[55]:

```
(157, 8)
```

Drop columns

In [57]:

```
# dropping duplicated temp column no need to do since already done hence copyi
#data_cleaned2 = data_cleaned1.drop('Temp C',axis = 1)
data_cleaned2 = data_cleaned1.copy()
data_cleaned2
```

Out[57]:

	Ozone	Solar.R	Wind	Temp C	Month	Day	Year	Weather
0	41.0	190.0	7.4	67.0	5.0	1	2010	S
1	36.0	118.0	8.0	72.0	5.0	2	2010	C
2	12.0	149.0	12.6	74.0	5.0	3	2010	PS
3	18.0	313.0	11.5	62.0	5.0	4	2010	S
4	NaN	NaN	14.3	56.0	5.0	5	2010	S
...	...	...	...	...	...	...	...	...
152	20.0	223.0	11.5	68.0	9.0	30	2010	S
153	41.0	190.0	7.4	67.0	5.0	1	2010	C
154	30.0	193.0	6.9	70.0	9.0	26	2010	PS
155	NaN	145.0	13.2	77.0	9.0	27	2010	S
157	18.0	131.0	8.0	76.0	9.0	29	2010	C

157 rows × 8 columns

Rename column

In [70]:
```
data_cleaned3 = data_cleaned2.rename({'Solar.R':'Solar'},axis=1)
data_cleaned3
```

Out[70]:

	Ozone	Solar	Wind	Temp C	Month	Day	Year	Weather
0	41.0	190.0	7.4	67.0	5.0	1	2010	S
1	36.0	118.0	8.0	72.0	5.0	2	2010	C
2	12.0	149.0	12.6	74.0	5.0	3	2010	PS
3	18.0	313.0	11.5	62.0	5.0	4	2010	S
4	NaN	NaN	14.3	56.0	5.0	5	2010	S
...	...	...	...	...	...	...	...	...
152	20.0	223.0	11.5	68.0	9.0	30	2010	S
153	41.0	190.0	7.4	67.0	5.0	1	2010	C
154	30.0	193.0	6.9	70.0	9.0	26	2010	PS
155	NaN	145.0	13.2	77.0	9.0	27	2010	S
157	18.0	131.0	8.0	76.0	9.0	29	2010	C

157 rows × 8 columns

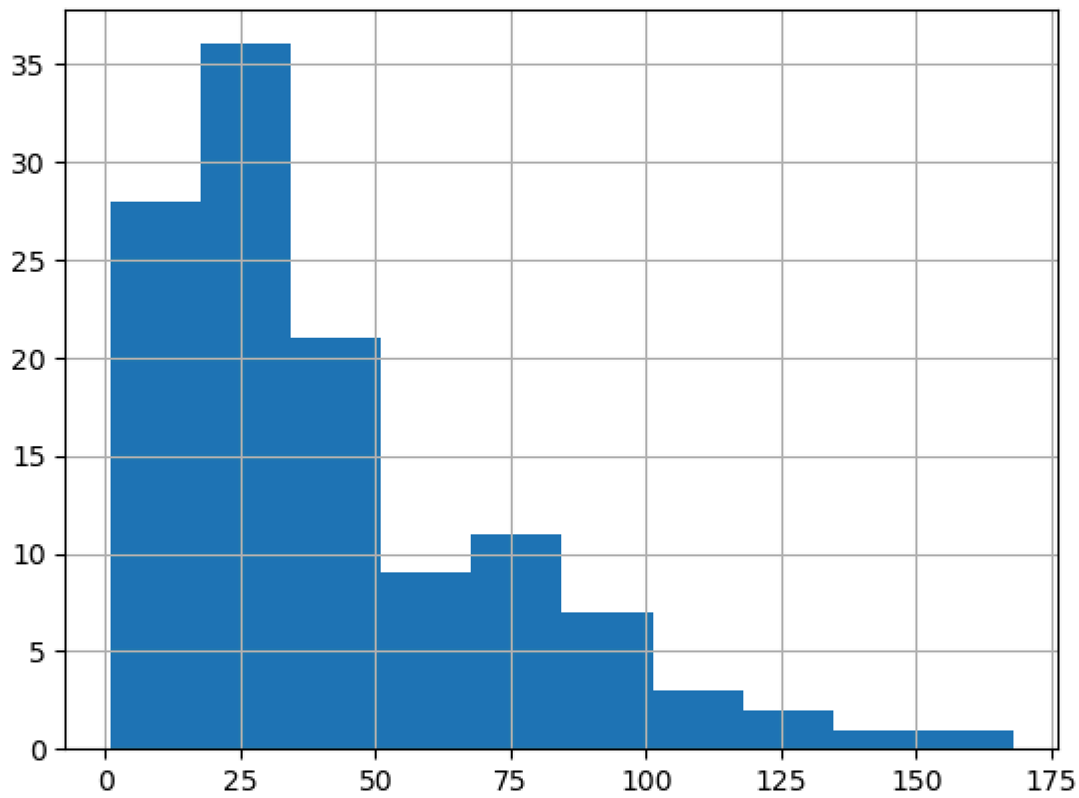
Outlier Detection

In [67]:

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

Out[67]:

```
<Axes: >
```

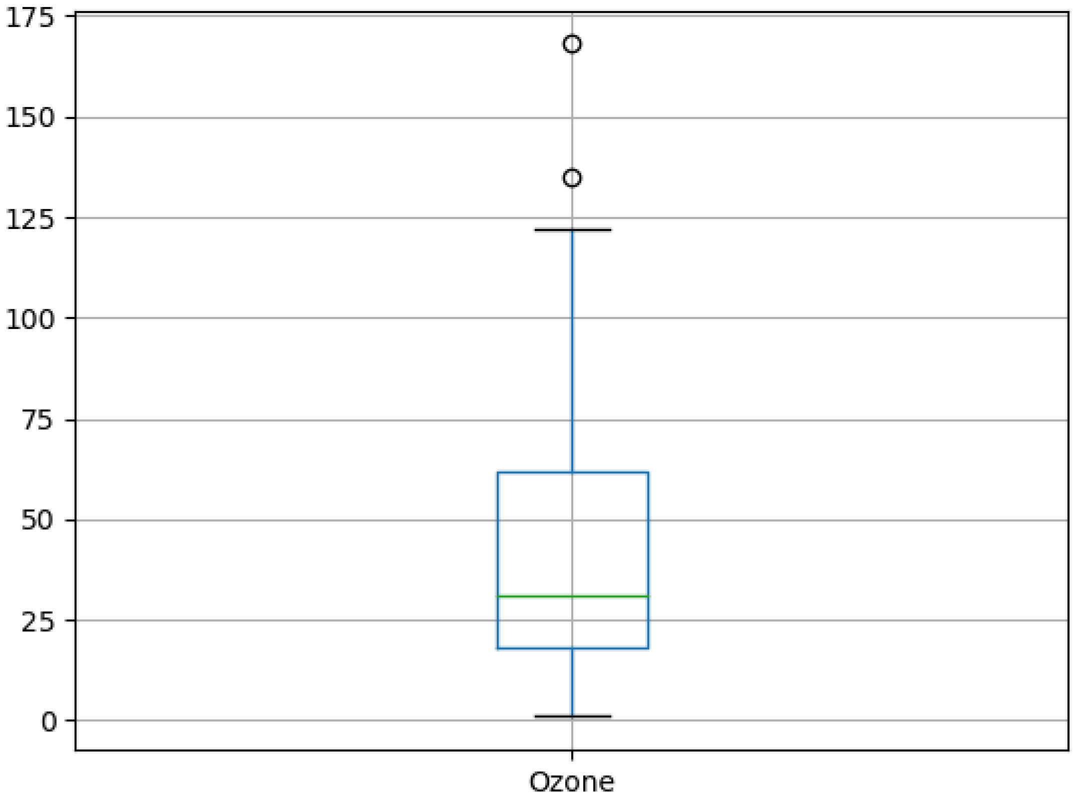


In [65]:

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

Out[65]:

```
<Axes: >
```



In [71]:

```
#Descriptive stat

data_cleaned3['Ozone'].describe()
```

Out[71]:

```
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 [72]:

```
data_cleaned3
```

Out[72]:

	Ozone	Solar	Wind	Temp C	Month	Day	Year	Weather
0	41.0	190.0	7.4	67.0	5.0	1	2010	S
1	36.0	118.0	8.0	72.0	5.0	2	2010	C
2	12.0	149.0	12.6	74.0	5.0	3	2010	PS
3	18.0	313.0	11.5	62.0	5.0	4	2010	S
4	NaN	NaN	14.3	56.0	5.0	5	2010	S
...	...	...	...	...	...	...	...	...
152	20.0	223.0	11.5	68.0	9.0	30	2010	S
153	41.0	190.0	7.4	67.0	5.0	1	2010	C
154	30.0	193.0	6.9	70.0	9.0	26	2010	PS
155	NaN	145.0	13.2	77.0	9.0	27	2010	S
157	18.0	131.0	8.0	76.0	9.0	29	2010	C

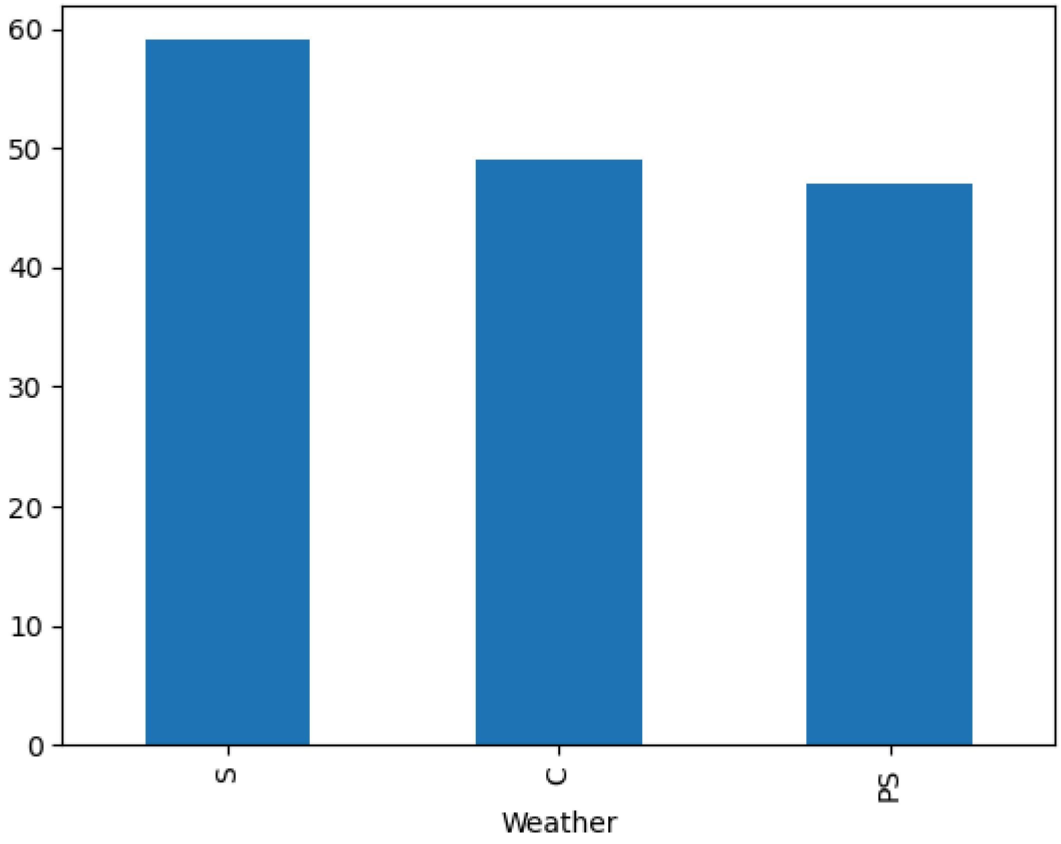
157 rows × 8 columns

In [73]:

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

Out[73]:

```
<Axes: xlabel='Weather'>
```



Missing Values and Imputation

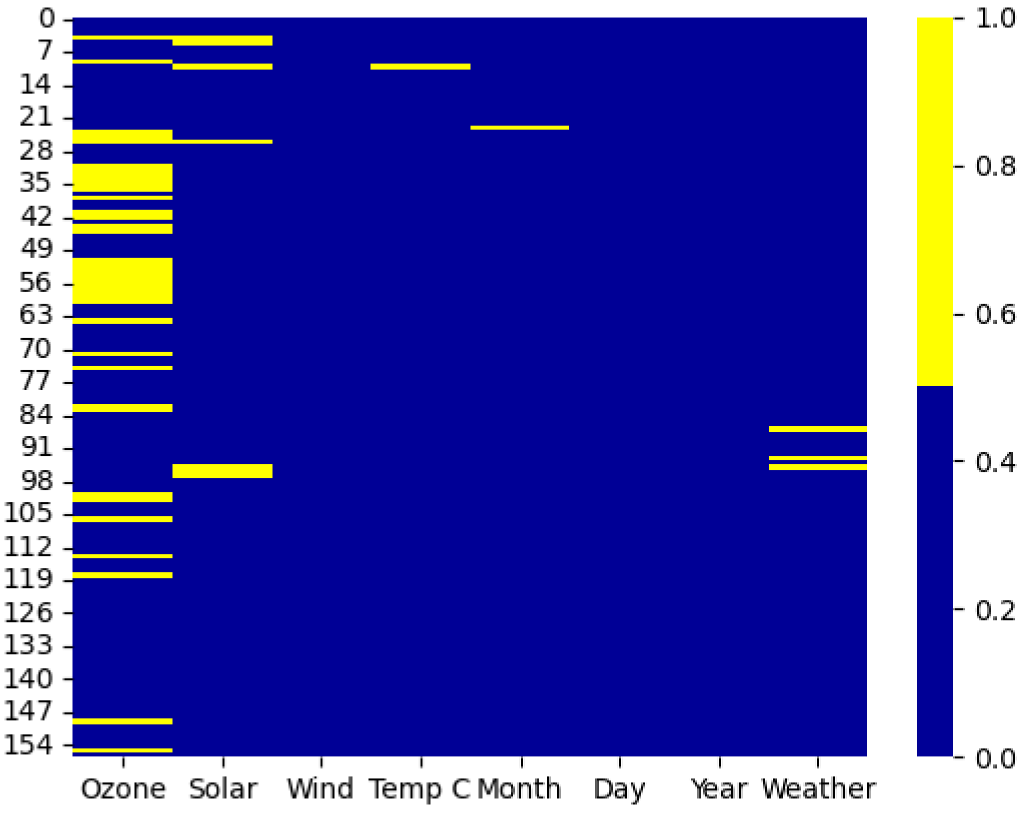
In [82]:

```python
import seaborn as sns

cols = data_cleaned3.columns
colours = ['#000099', '#ffff00']
sns.heatmap(data_cleaned3[cols].isnull(), cmap=sns.color_palette(colours))
```

Out[82]:

```
<Axes: >
```



In [83]:

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

Out[83]:

	Ozone	Solar	Wind	Temp C	Month	Day	Year	Weather
4	NaN	NaN	14.3	56.0	5.0	5	2010	S
5	28.0	NaN	14.9	66.0	5.0	6	2010	C
9	NaN	194.0	8.6	69.0	5.0	10	2010	S
10	7.0	NaN	6.9	NaN	5.0	11	2010	C
23	32.0	92.0	12.0	61.0	NaN	24	2010	C

In [84]:

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

Out[84]:

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

In [85]:

```
#Mean Imputation

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

Out[85]:

```
41.81512605042017
```

In [87]:

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

Out[87]:

	Ozone	Solar	Wind	Temp C	Month	Day	Year	Weather
0	41.000000	190.0	7.4	67.0	5.0	1	2010	S
1	36.000000	118.0	8.0	72.0	5.0	2	2010	C
2	12.000000	149.0	12.6	74.0	5.0	3	2010	PS
3	18.000000	313.0	11.5	62.0	5.0	4	2010	S
4	41.815126	NaN	14.3	56.0	5.0	5	2010	S
...	...	...	...	...	...	...	...	...
152	20.000000	223.0	11.5	68.0	9.0	30	2010	S
153	41.000000	190.0	7.4	67.0	5.0	1	2010	C
154	30.000000	193.0	6.9	70.0	9.0	26	2010	PS
155	41.815126	145.0	13.2	77.0	9.0	27	2010	S
157	18.000000	131.0	8.0	76.0	9.0	29	2010	C

157 rows × 8 columns

In [88]:

```
#Mean Imputation of solar

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

Out[88]:

```
185.36666666666667
```

In [90]:

```
#Mean Imputation of Month

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

Out[90]:

```
7.032051282051282
```

In [92]:

```
#Mean Imputation of Temp C

mean = data_cleaned3['Temp C'].mean()
data_cleaned3['Temp C'] = data_cleaned3['Temp C'].fillna(mean)
mean
```

Out[92]:

```
77.76923076923077
```

In [94]:

```
# Imputation of Weather
obj_columns = data_cleaned3[['Weather']]
obj_columns.isnull().sum()
```

Out[94]:

```
Weather    3
dtype: int64
```

In [95]:

```
obj_columns
```

Out[95]:

	Weather
0	S
1	C
2	PS
3	S
4	S
...	...
152	S
153	C
154	PS
155	S
157	C

157 rows × 1 columns

In [96]:

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

In [97]:

```
obj_columns.mode()
```

Out[97]:

	Weather
0	S

In [99]:

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

Out[99]:

```
Weather    0
dtype: int64
```

In [102]:

```
data_cleaned3.shape
```

Out[102]:

```
(157, 8)
```

In [103]:

```
obj_columns.shape
```

Out[103]:

```
(157, 1)
```

In [105]:

```
#join data set with imputed value

data_cleaned4 = pd.concat([data_cleaned3,obj_columns],axis=1)
```

In [108]:

```
data_cleaned4
```

Out[108]:

	Ozone	Solar	Wind	Temp C	Month	Day	Year	Weather	Weather
0	41.000000	190.000000	7.4	67.0	5.0	1	2010	S	S
1	36.000000	118.000000	8.0	72.0	5.0	2	2010	C	C
2	12.000000	149.000000	12.6	74.0	5.0	3	2010	PS	PS
3	18.000000	313.000000	11.5	62.0	5.0	4	2010	S	S
4	41.815126	185.366667	14.3	56.0	5.0	5	2010	S	S
...	...	...	...	...	...	...	...	...	...
152	20.000000	223.000000	11.5	68.0	9.0	30	2010	S	S
153	41.000000	190.000000	7.4	67.0	5.0	1	2010	C	C
154	30.000000	193.000000	6.9	70.0	9.0	26	2010	PS	PS
155	41.815126	145.000000	13.2	77.0	9.0	27	2010	S	S
157	18.000000	131.000000	8.0	76.0	9.0	29	2010	C	C

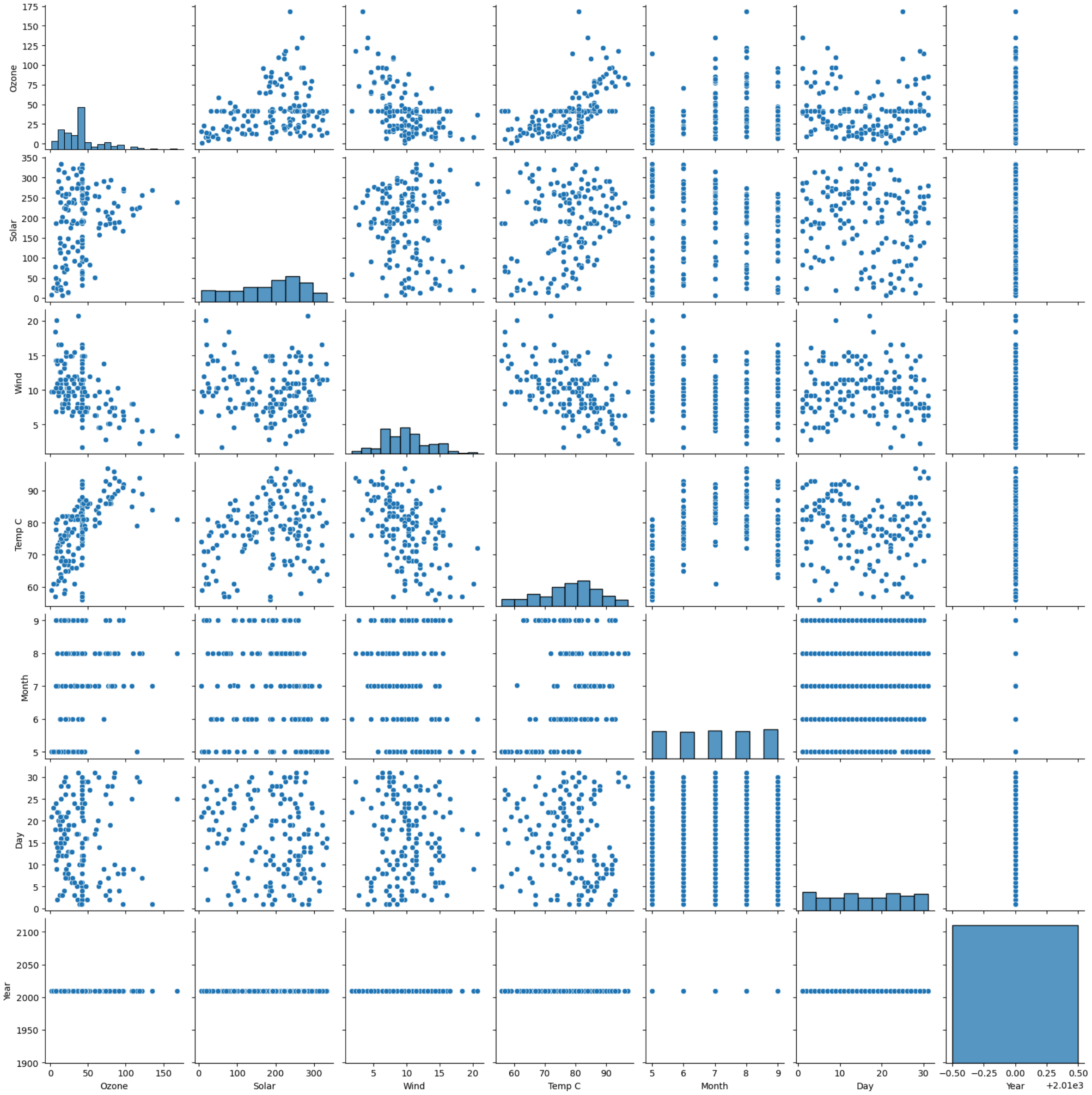
157 rows × 9 columns

Scatter Plot and correlation analysis

```
In [109]: sns.pairplot(data_cleaned3)
```

```
c:\Users\HOME\anaconda3\Lib\site-packages\seaborn\axisgrid.py:123: UserWarnin
g: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

```
Out[109]: <seaborn.axisgrid.PairGrid at 0x22c87856d90>
```



In [111]:

```
# Select only numeric columns
numeric_data = data_cleaned3.select_dtypes(include=np.number)

#compute correlation matrix
correlation_matrix = numeric_data.corr()

#print correlation matrix
correlation_matrix
```

Out[111]:

	Ozone	Solar	Wind	Temp C	Month	Day	Year
Ozone	1.000000	0.304559	-0.520004	0.603660	0.132809	-0.021916	NaN
Solar	0.304559	1.000000	-0.055874	0.260810	-0.090564	-0.151007	NaN
Wind	-0.520004	-0.055874	1.000000	-0.443676	-0.166029	0.029900	NaN
Temp C	0.603660	0.260810	-0.443676	1.000000	0.390957	-0.124262	NaN
Month	0.132809	-0.090564	-0.166029	0.390957	1.000000	0.049924	NaN
Day	-0.021916	-0.151007	0.029900	-0.124262	0.049924	1.000000	NaN
Year	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Transformations

Dummy Variable

In [115]:

```
#creating dummy for weather

data_cleaned4 = pd.get_dummies(data,columns=['Weather'])
data_cleaned4
```

Out[115]:

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

158 rows × 10 columns

In [116]:

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

Normalization of data

In [117]:

```
from numpy import set_printoptions
from sklearn.preprocessing import MinMaxScaler
```

In [118]:

```
data_cleaned4.values
```

Out[118]:

```
array([[41.0, 190.0, 7.4, ..., False, False, True],
       [36.0, 118.0, 8.0, ..., True, False, False],
       [12.0, 149.0, 12.6, ..., False, True, False],
       ...,
       [30.0, 193.0, 6.9, ..., False, True, False],
       [14.0, 191.0, 14.3, ..., False, False, True],
       [18.0, 131.0, 8.0, ..., True, False, False]], dtype=object)
```

In [120]:

```
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  0.27 0.  ]
 [0.21 0.34 0.33 0.39 0.  ]
 [0.07 0.43 0.57 0.44 0.  ]
 [0.1  0.94 0.52 0.15 0.  ]
 [0.24 0.55 0.66 0.   0.  ]]
```

In [121]:

```
#standardize data (0mean,1sd)
from sklearn.preprocessing import StandardScaler
```

In [122]:

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

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

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

Speed up the EDA process

In [123]:

```
#install dtale

import dtale
dtale.show(data)
```

Out[123]: