Date: 29/02/2024

EXPERIMENT NO. 6

AIM: To study data cleaning and Exploratory data analysis.

SOFTWARE USED: Jupyter Notebook.

THEORY:

Data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset. When combining multiple data sources, there are many opportunities for data to be duplicated or mislabeled. If data is incorrect, outcomes and algorithms are unreliable, even though they may look correct. There is no one absolute way to prescribe the exact steps in the data cleaning process because the processes will vary from dataset to dataset. But it is crucial to establish a template for your data cleaning process so you know you are doing it the right way every time.

5 characteristics of quality data:

- Validity. The degree to which your data conforms to defined business rules or constraints.
- Accuracy. Ensure your data is close to the true values.
- Completeness. The degree to which all required data is known.
- Consistency. Ensure your data is consistent within the same dataset and/or across multiple data sets.
- Uniformity. The degree to which the data is specified using the same unit of measure.

Advantages and benefits of data cleaning:

Having clean data will ultimately increase overall productivity and allow for the highest quality information in your decision-making. Benefits include:

- Removal of errors when multiple sources of data are at play.
- Fewer errors make for happier clients and less-frustrated employees.
- Ability to map the different functions and what your data is intended to do.
- Monitoring errors and better reporting to see where errors are coming from, making it easier to fix incorrect or corrupt data for future applications.
- Using tools for data cleaning will make for more efficient business practices and quicker decision-making.

The main purpose of EDA is to help look at data before making any assumptions. It can help identify obvious errors, as well as better understand patterns within the data, detect outliers or anomalous events, find interesting relations among the variables.

Data scientists can use exploratory analysis to ensure the results they produce are valid and applicable to any desired business outcomes and goals. EDA also helps stakeholders by confirming they are asking the right questions. EDA can help answer questions about standard deviations, categorical variables, and confidence intervals. Once EDA is complete and insights are drawn, its features can then be used for more sophisticated data analysis or modeling, including machine learning.

OUTPUT:

In [1]:
```
#data cleaning
```

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import datasets
iris = datasets.load_iris()
```

In [3]:
```
data1 = pd.read_csv('data_clean.csv')
```

In [4]:
```
data1.head()
```

Out[4]:

	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 [5]:
```
data1.tail()
```

Out[5]:

	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 [6]:
```
type(data1)
```

Out[6]:
```
pandas.core.frame.DataFrame
```

In [7]:
```
data1.shape
```

Out[7]:
```
(158, 10)
```

In [8]:
```
data1.dtypes
```

Out[8]:
```
Unnamed: 0      int64
Ozone         float64
Solar.R       float64
Wind          float64
Temp C         object
Month          object
Day             int64
Year            int64
Temp            int64
Weather        object
```

In [9]:
```
#there is false information because temp and month are object should be int
```

In [10]:
```
#data type conversion
```

In [11]:
```
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 [12]:
```
data2 = data1.iloc[:,1:]
```

In [13]:
```
data2
```

Out[13]:

	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 [14]:
```
data = data2.copy()
```

In [15]:
```
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 [16]:
```
data.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 [17]:
```
#duplicates
```

In [18]:
```
data[data.duplicated()].shape
```

Out[18]: (1, 9)

In [19]:
```
data
```

Out[19]:

	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 [20]:
```
data[data.duplicated()]
```

Out[20]:

	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 [21]:
```
data_cleaned1 = data.drop_duplicates()
```

In [22]:
```
data_cleaned1.shape
```
Out[22]:
```
(157, 9)
```
In [23]:
```
#drop columns
```
In [24]:
```
data_cleaned2 = data_cleaned1.drop('Temp C', axis=1)
```
In [25]:
```
data_cleaned2
```
Out[25]:

	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

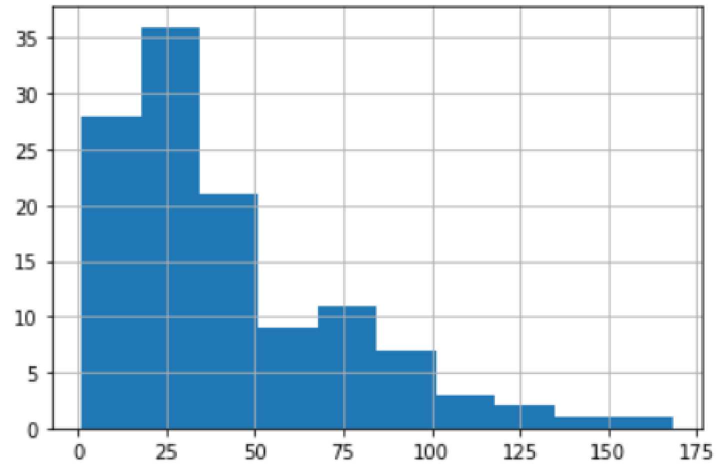
In [26]:
```
#rename the columns
```
In [27]:
```
data_cleaned3 = data_cleaned2.rename({'Solar.R':'Solar'}, axis=1)
```
In [28]:
```
data_cleaned3
```
Out[28]:

	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

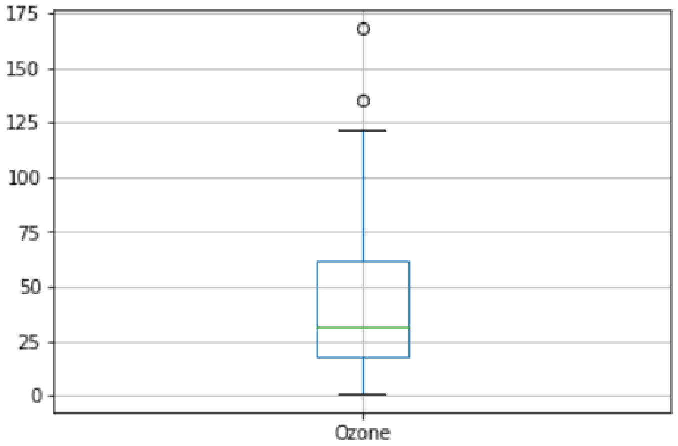
157 rows × 8 columns

In [29]:
```
#outlier detection
```
In [30]:
```
data_cleaned3['Ozone'].hist()
```

In [30]:
```
data_cleaned3['Ozone'].hist()
```
Out[30]: <Axes: >



In [31]:
```
data_cleaned3.boxplot(column=['Ozone'])
```
Out[31]: <Axes: >

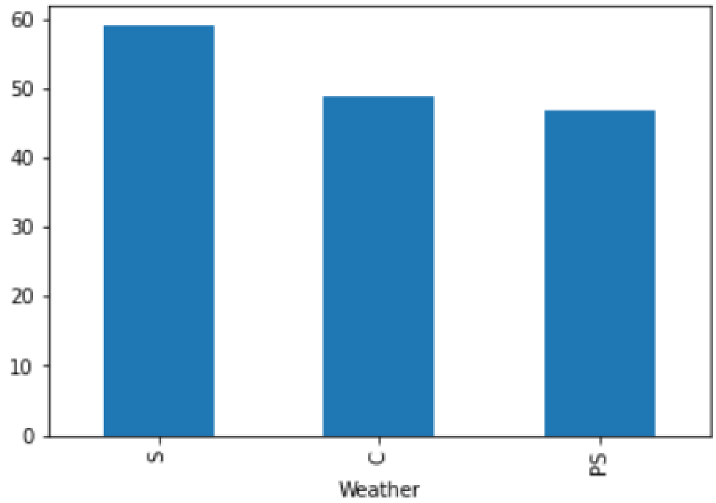


In [32]:
```
data_cleaned3['Ozone'].describe()
```
Out[32]:
```
count    119.000000
mean      41.815126
std       32.659249
min        1.000000
25%       18.000000
50%       31.000000
75%       62.000000
max      168.000000
Name: Ozone, dtype: float64
```

In [33]:
```
data['Weather'].value_counts().plot.bar()
```

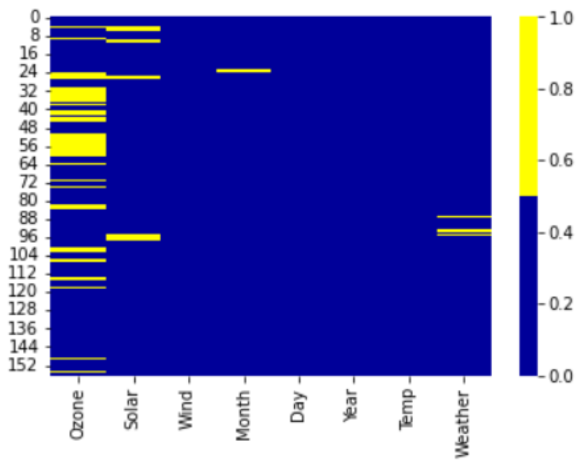
In [33]:
```
data['Weather'].value_counts().plot.bar()
```

Out[33]: <Axes: xlabel='Weather'>



In [34]:
```
#missing values and imputation
import seaborn as sns
cols = data_cleaned3.columns
colours = ['#000099','#ffff00']
sns.heatmap(data_cleaned3[cols].isnull(),cmap = sns.color_palette(colours))
#yellow is for missing value and blue is non missing values
```

Out[34]: <Axes: >



In [35]:
```
data_cleaned3[data_cleaned3.isnull().any(axis=1)].head()
```
Out[35]:

	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 [36]:
```
data_cleaned3.isnull().sum()
```
Out[36]:
```
Ozone      38
Solar       7
Wind        0
Month       1
Day         0
Year        0
Temp        0
Weather     3
dtype: int64
```

In [37]:
```
mean = data_cleaned3['Ozone'].mean()
print(mean)
```
```
41.81512605042017
```

In [38]:
```
data_cleaned3['Ozone'] = data_cleaned3['Ozone'].fillna(mean)
```

In [39]:
```
data_cleaned3
```
Out[39]:

	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 [40]:
```
mean = data_cleaned3['Solar'].mean()
print(mean)
data_cleaned3['Solar'] = data_cleaned3['Solar'].fillna(mean)
```
```
185.36666666666667
```

In [41]:
```
mean = data_cleaned3['Month'].mean()
print(mean)
data_cleaned3['Month'] = data_cleaned3['Month'].fillna(mean)
```

```
7.032051282051282
```

In [42]:
```
obj_columns = data_cleaned3['Weather']
```

In [43]:
```
obj_columns.isnull().sum()
```

Out[43]:
```
3
```

In [44]:
```
obj_columns = obj_columns.fillna(obj_columns.mode().iloc[0])
```

In [45]:
```
obj_columns.isnull().sum()
```

Out[45]:
```
0
```

In [46]:
```
data_cleaned3.shape
```

Out[46]:
```
(157, 8)
```

In [47]:
```
data_cleaned4 = pd.concat([data_cleaned3,obj_columns], axis=1)
```

In [48]:
```
data_cleaned4.isnull().sum()
```

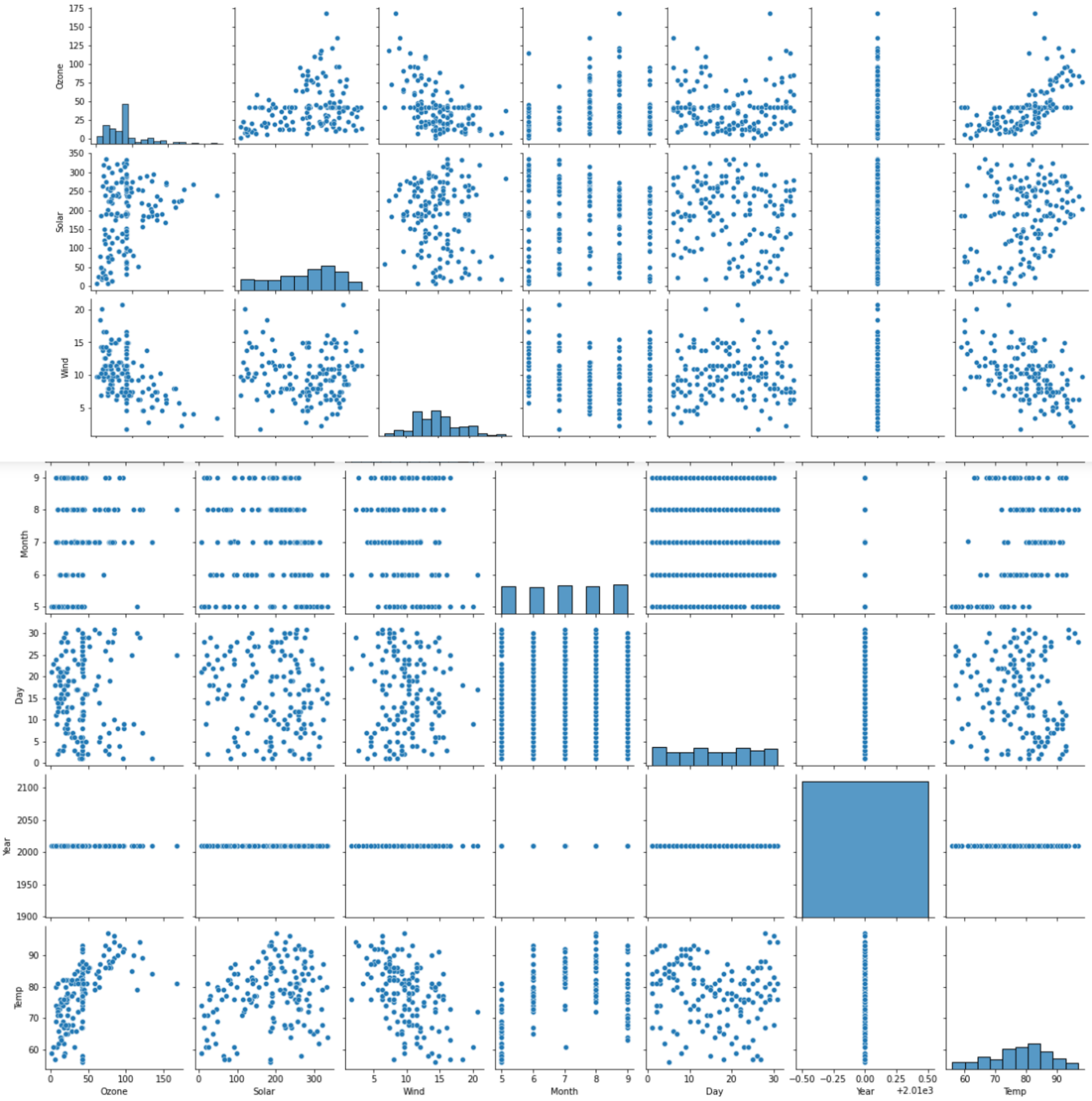
Out[48]:
```
Ozone      0
Solar      0
Wind       0
Month      0
Day        0
Year       0
Temp       0
Weather    3
Weather    0
dtype: int64
```

In [49]:
```
#scatter plot and correlation
```

In [50]:
```python
sns.pairplot(data_cleaned3)
```

```
C:\Users\anjal\anaconda3\envs\myenv\lib\site-packages\seaborn\axisgrid.py:123: UserWarning: The figure layout has changed to ti
ght
  self._figure.tight_layout(*args, **kwargs)
```

Out[50]:
```
<seaborn.axisgrid.PairGrid at 0x16ae47117b0>
```



In [51]:
```python
numeric_data = data_cleaned3.select_dtypes(include=np.number)
correlation_matrix = numeric_data.corr()
print(correlation_matrix)
```

```
          Ozone     Solar      Wind     Month       Day  Year      Temp
Ozone  1.000000  0.304559 -0.520004  0.132809 -0.021916   NaN  0.606500
Solar  0.304559  1.000000 -0.055874 -0.090564 -0.151007   NaN  0.260677
Wind  -0.520004 -0.055874  1.000000 -0.166029  0.029900   NaN -0.441228
Month  0.132809 -0.090564 -0.166029  1.000000  0.049924   NaN  0.394420
Day   -0.021916 -0.151007  0.029900  0.049924  1.000000   NaN -0.122787
Year        NaN       NaN       NaN       NaN       NaN   NaN       NaN
Temp   0.606500  0.260677 -0.441228  0.394420 -0.122787   NaN  1.000000
```

In [52]:
```python
#transformations
```

In [53]:
```python
data_cleaned4 = pd.get_dummies(data,columns=['Weather'])
```

In [54]:
```
data_cleaned4
```

Out[54]:

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

158 rows × 11 columns

In [55]:
```
from numpy import set_printoptions
from sklearn.preprocessing import MinMaxScaler
```

In [56]:
```
data_cleaned4.values
```

Out[56]:
```
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],
       ...,
       [nan, 145.0, 13.2, ..., False, False, True],
       [14.0, 191.0, 14.3, ..., False, False, True],
       [18.0, 131.0, 8.0, ..., True, False, False]], dtype=object)
```

In [57]:
```
array = data_cleaned3.values

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

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 0.55 0.66 0.   0.13]]
```

In [58]:
```
#Standardization
from sklearn.preprocessing import StandardScaler
```

In [59]:
```
array = data_cleaned4.values

scaler = StandardScaler().fit(array)
rescaledX = scaler.transform(array)

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

```
[[-0.02  0.05 -0.73 -1.15 -1.43 -1.67  0.   -1.15 -0.67 -0.65  1.3 ]
 [-0.17 -0.76 -0.56 -0.61 -1.43 -1.56  0.   -0.61  1.49 -0.65 -0.77]
 [-0.91 -0.41  0.75 -0.4  -1.43 -1.45  0.   -0.4  -0.67  1.54 -0.77]
 [-0.73  1.44  0.44 -1.68 -1.43 -1.34  0.   -1.68 -0.67 -0.65  1.3 ]
 [  nan   nan  1.24 -2.32 -1.43 -1.23  0.   -2.32 -0.67 -0.65  1.3 ]]
```

In [60]:
```
#auto EDA library
import dtale
dtale.show(data)
```

▸ 9 158	Ozone	Solar.R	Wind	Temp C	Month	Day	Year	Temp	Weather
0	41.00	190.00	7.40	67.00	5.00	1	2010	67	S
1	36.00	118.00	8.00	72.00	5.00	2	2010	72	C
2	12.00	149.00	12.60	74.00	5.00	3	2010	74	PS
3	18.00	313.00	11.50	62.00	5.00	4	2010	62	S
4	nan	nan	14.30	56.00	5.00	5	2010	56	S
5	28.00	nan	14.90	66.00	5.00	6	2010	66	C
6	23.00	299.00	8.60	65.00	5.00	7	2010	65	PS
7	19.00	99.00	13.80	59.00	5.00	8	2010	59	C
8	8.00	19.00	20.10	61.00	5.00	9	2010	61	PS

1	36.00	118.00	8.00	72.00	5.00	2	2010	72	C
2	12.00	149.00	12.60	74.00	5.00	3	2010	74	PS
3	18.00	313.00	11.50	62.00	5.00	4	2010	62	S
4	nan	nan	14.30	56.00	5.00	5	2010	56	S
5	28.00	nan	14.90	66.00	5.00	6	2010	66	C
6	23.00	299.00	8.60	65.00	5.00	7	2010	65	PS
7	19.00	99.00	13.80	59.00	5.00	8	2010	59	C
8	8.00	19.00	20.10	61.00	5.00	9	2010	61	PS
9	nan	194.00	8.60	69.00	5.00	10	2010	69	S
10	7.00	nan	6.90	nan	5.00	11	2010	74	C
11	16.00	256.00	9.70	69.00	5.00	12	2010	69	PS
12	11.00	290.00	9.20	66.00	5.00	13	2010	66	S
13	14.00	274.00	10.90	68.00	5.00	14	2010	68	S
14	18.00	65.00	13.20	58.00	5.00	15	2010	58	C
15	14.00	334.00	11.50	64.00	5.00	16	2010	64	S
16	34.00	307.00	12.00	66.00	5.00	17	2010	66	S
17	6.00	78.00	18.40	57.00	5.00	18	2010	57	C

ut[60]:

CONCLUSION:

The data cleaning experiment emphasized the importance of enhancing data quality for accurate analysis. Through techniques like error detection and validation, we improved data integrity. Moving forward, documenting cleaning processes will ensure transparency. Overall, the experiment highlights the critical role of data cleaning in facilitating informed decision-making.