```
import pandas as pd
import numpy as np
df=pd.read_csv("c:/Users/Kajal/Downloads/AirQuality.csv",encoding='cp1252')
```

```
C:\Users\Kajal\AppData\Local\Temp\ipykernel_17868\4117198006.py:3: DtypeWarning: Columns (0) have mixed types. Specify dtype option on i
  df=pd.read_csv("c:/Users/Kajal/Downloads/AirQuality.csv",encoding='cp1252')
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
#import missingno as msno # To visualize missing value
#import plotly.graph_objects as go # To Generate Graphs
#import plotly.express as px # To Generate box plot for statistical representation
%matplotlib inlineimport os
```

```
UsageError: unrecognized arguments: os
```

```
df.head()
```

	stn_code	sampling_date	state	location	agency	type	so2	no2	rspm	spm	location_monitoring_station	pm2_5
0	150.0	February - M021990	Andhra Pradesh	Hyderabad	NaN	Residential, Rural and other Areas	4.8	17.4	NaN	NaN	NaN	NaN
1	151.0	February - M021990	Andhra Pradesh	Hyderabad	NaN	Industrial Area	3.1	7.0	NaN	NaN	NaN	NaN
2	152.0	February - M021990	Andhra Pradesh	Hyderabad	NaN	Residential, Rural and other Areas	6.2	28.5	NaN	NaN	NaN	NaN
3	150.0	March - M031990	Andhra Pradesh	Hyderabad	NaN	Residential, Rural and other Areas	6.3	14.7	NaN	NaN	NaN	NaN

```
df.describe()
```

	so2	no2	rspm	spm	pm2_5
count	401096.000000	419509.000000	395520.000000	198355.000000	9314.000000
mean	10.829414	25.809623	108.832784	220.783480	40.791467
std	11.177187	18.503086	74.872430	151.395457	30.832525
min	0.000000	0.000000	0.000000	0.000000	3.000000
25%	5.000000	14.000000	56.000000	111.000000	24.000000
50%	8.000000	22.000000	90.000000	187.000000	32.000000
75%	13.700000	32.200000	142.000000	296.000000	46.000000
max	909.000000	876.000000	6307.033333	3380.000000	504.000000

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 435742 entries, 0 to 435741
Data columns (total 13 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   stn_code                    291665 non-null  object
 1   sampling_date               435739 non-null  object
 2   state                       435742 non-null  object
 3   location                    435739 non-null  object
 4   agency                      286261 non-null  object
 5   type                        430349 non-null  object
 6   so2                         401096 non-null  float64
 7   no2                         419509 non-null  float64
 8   rspm                        395520 non-null  float64
 9   spm                         198355 non-null  float64
 10  location_monitoring_station 408251 non-null  object
 11  pm2_5                       9314 non-null    float64
```

```
 12  date                        435735 non-null  object
dtypes: float64(5), object(8)
memory usage: 43.2+ MB
```

```
df.shape
```

```
(435742, 13)
```

```
df.isnull().sum()
```

```
stn_code                     144077
sampling_date                     3
state                             0
location                          3
agency                       149481
type                           5393
so2                           34646
no2                           16233
rspm                          40222
spm                          237387
location_monitoring_station   27491
pm2_5                        426428
date                              7
dtype: int64
```

```
df.count()
```

```
stn_code                     291665
sampling_date                435739
state                        435742
location                     435739
agency                       286261
type                         430349
so2                          401096
no2                          419509
rspm                         395520
spm                          198355
location_monitoring_station  408251
pm2_5                          9314
date                         435735
dtype: int64
```

```
df.describe()
```

	so2	no2	rspm	spm	pm2_5
count	401096.000000	419509.000000	395520.000000	198355.000000	9314.000000
mean	10.829414	25.809623	108.832784	220.783480	40.791467
std	11.177187	18.503086	74.872430	151.395457	30.832525
min	0.000000	0.000000	0.000000	0.000000	3.000000
25%	5.000000	14.000000	56.000000	111.000000	24.000000
50%	8.000000	22.000000	90.000000	187.000000	32.000000
75%	13.700000	32.200000	142.000000	296.000000	46.000000
max	909.000000	876.000000	6307.033333	3380.000000	504.000000

a) Cleaning the dataset

```
#  Dropping columns which are not required
df = df.drop(['stn_code','agency', 'location_monitoring_station'],axis=1)
```

```
df.isna().sum()
```

```
sampling_date         3
state                 0
location              3
type               5393
so2               34646
no2               16233
rspm              40222
```

```
spm             237387
pm2_5           426428
date                 7
dtype: int64
```

```
df = df.dropna(subset=['date'])
```

```
df.isna().sum()
```

```
sampling_date         0
state                 0
location              0
type               5390
so2               34643
no2               16230
rspm              40219
spm              237380
pm2_5            426421
date                  0
dtype: int64
```

```
df.columns
```

```
Index(['sampling_date', 'state', 'location', 'type', 'so2', 'no2', 'rspm',
       'spm', 'pm2_5', 'date'],
      dtype='object')
```

Changing the types to uniform format

```
df['type'].unique()
```

```
array(['Residential, Rural and other Areas', 'Industrial Area', nan,
       'Sensitive Area', 'Industrial Areas', 'Residential and others',
       'Sensitive Areas', 'Industrial', 'Residential', 'RIRUO',
       'Sensitive'], dtype=object)
```

```
types = {

    "Residential" : "K",
    "Residential and others" : "RO",
    "Industrial Area":"I" ,
    "Industrial Areas" : "I",
    "Industrial" : "I" ,
    "Sensitive Area": "s",
    "Sensitive Areas":"s",
    "Sensitive":"s",
    "NaN":"PRO",
    "Residential, Rural and other Areas":"MO"
 }
```

```
df.type = df.type.replace(types)
```

```
df['type'].unique()
```

```
array(['MO', 'I', nan, 's', 'RO', 'K', 'RIRUO'], dtype=object)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 435735 entries, 0 to 435738
Data columns (total 10 columns):
 #   Column         Non-Null Count   Dtype  
---  ------         --------------   -----  
 0   sampling_date  435735 non-null  object 
 1   state          435735 non-null  object 
 2   location       435735 non-null  object 
 3   type           430345 non-null  object 
 4   so2            401092 non-null  float64
 5   no2            419505 non-null  float64
 6   rspm           395516 non-null  float64
```

```
 7   spm            198355 non-null  float64
 8   pm2_5          9314 non-null    float64
 9   date           435735 non-null  object
dtypes: float64(5), object(5)
memory usage: 36.6+ MB
```

```
# convert date to 'date' format

df['date']=pd.to_datetime(df['date'], errors="coerce")
df.head(5)
```

	sampling_date	state	location	type	so2	no2	rspm	spm	pm2_5	date
0	February - M021990	Andhra Pradesh	Hyderabad	MO	4.8	17.4	NaN	NaN	NaN	1990-02-01
1	February - M021990	Andhra Pradesh	Hyderabad	I	3.1	7.0	NaN	NaN	NaN	1990-02-01
2	February - M021990	Andhra Pradesh	Hyderabad	MO	6.2	28.5	NaN	NaN	NaN	1990-02-01
3	March - M031990	Andhra Pradesh	Hyderabad	MO	6.3	14.7	NaN	NaN	NaN	1990-03-01
4	March - M031990	Andhra Pradesh	Hyderabad	I	4.7	7.5	NaN	NaN	NaN	1990-03-01

```
# Create a new column 'year'  from 'date' column

df['year']=df.date.dt.year
df.head()
```

	sampling_date	state	location	type	so2	no2	rspm	spm	pm2_5	date	year
0	February - M021990	Andhra Pradesh	Hyderabad	MO	4.8	17.4	NaN	NaN	NaN	1990-02-01	1990
1	February - M021990	Andhra Pradesh	Hyderabad	I	3.1	7.0	NaN	NaN	NaN	1990-02-01	1990
2	February - M021990	Andhra Pradesh	Hyderabad	MO	6.2	28.5	NaN	NaN	NaN	1990-02-01	1990
3	March - M031990	Andhra Pradesh	Hyderabad	MO	6.3	14.7	NaN	NaN	NaN	1990-03-01	1990
4	March - M031990	Andhra Pradesh	Hyderabad	I	4.7	7.5	NaN	NaN	NaN	1990-03-01	1990

Handling missing values

```
#  define columns of importance, which shall be used regularely

COLS = ['so2','no2', 'rspm', 'spm', 'pm2_5']
```

```
import numpy as np
```

```
from sklearn.impute import SimpleImputer
```

```
imputer = SimpleImputer(missing_values = np.nan, strategy='mean')
```

```
df[COLS] = imputer.fit_transform(df[COLS])
```

```
df.head()
```

	sampling_date	state	location	type	so2	no2	rspm	spm	pm2_5	date	year
0	February - M021990	Andhra Pradesh	Hyderabad	MO	4.8	17.4	108.833091	220.78348	40.791467	1990-02-01	1990
1	February - M021990	Andhra Pradesh	Hyderabad	I	3.1	7.0	108.833091	220.78348	40.791467	1990-02-01	1990
2	February - M021990	Andhra Pradesh	Hyderabad	MO	6.2	28.5	108.833091	220.78348	40.791467	1990-02-01	1990
3	March - M031990	Andhra Pradesh	Hyderabad	MO	6.3	14.7	108.833091	220.78348	40.791467	1990-03-01	1990
4	March - M031990	Andhra Pradesh	Hyderabad	I	4.7	7.5	108.833091	220.78348	40.791467	1990-03-01	1990

Data Transformation

Removing Duplicates: Duplicate rows may be found in a DataFrame for any number of reasons.

```
df.nunique()
```

```
sampling_date    5482
state              34
location          304
type                6
so2              4198
no2              6865
rspm             6066
spm              6669
pm2_5             434
date             5067
year               29
dtype: int64
```

```
df.duplicated().sum()
```

```
1135
```

```
df.drop_duplicates()
```

	sampling_date	state	location	type	so2	no2	rspm	spm	p
0	February - M021990	Andhra Pradesh	Hyderabad	MO	4.8	17.4	108.833091	220.78348	40.79
1	February - M021990	Andhra Pradesh	Hyderabad	I	3.1	7.0	108.833091	220.78348	40.79
2	February - M021990	Andhra Pradesh	Hyderabad	MO	6.2	28.5	108.833091	220.78348	40.79
3	March - M031990	Andhra Pradesh	Hyderabad	MO	6.3	14.7	108.833091	220.78348	40.79
4	March - M031990	Andhra Pradesh	Hyderabad	I	4.7	7.5	108.833091	220.78348	40.79
...	...	...	...	...	...	...	...	...	
435734	15-12-15	West Bengal	ULUBERIA	RIRUO	20.0	44.0	148.000000	220.78348	40.79
435735	18-12-15	West Bengal	ULUBERIA	RIRUO	17.0	44.0	131.000000	220.78348	40.79

Simple Replacement of Categorical Data with a Number

```
df.head()
```

	sampling_date	state	location	type	so2	no2	rspm	spm	pm2_5	date	year
0	February - M021990	Andhra Pradesh	Hyderabad	MO	4.8	17.4	108.833091	220.78348	40.791467	1990-02-01	1990
1	February - M021990	Andhra Pradesh	Hyderabad	I	3.1	7.0	108.833091	220.78348	40.791467	1990-02-01	1990
2	February - M021990	Andhra Pradesh	Hyderabad	MO	6.2	28.5	108.833091	220.78348	40.791467	1990-02-01	1990
3	March - M031990	Andhra Pradesh	Hyderabad	MO	6.3	14.7	108.833091	220.78348	40.791467	1990-03-01	1990
4	March - M031990	Andhra Pradesh	Hyderabad	I	4.7	7.5	108.833091	220.78348	40.791467	1990-03-01	1990

```
df['type'].value_counts()
```

```
MO       179013
I        148069
RO        86791
s         15010
RIRUO      1304
K           158
Name: type, dtype: int64
```

```
df['type'].replace({  'MO':1, 'I':2, 's':3 , 'RO':4, 'K':5, 'RIRUO':6  }, inplace=True)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 435735 entries, 0 to 435738
Data columns (total 11 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   sampling_date  435735 non-null  object
 1   state          435735 non-null  object
 2   location       435735 non-null  object
 3   type           430345 non-null  float64
 4   so2            435735 non-null  float64
 5   no2            435735 non-null  float64
 6   rspm           435735 non-null  float64
 7   spm            435735 non-null  float64
 8   pm2_5          435735 non-null  float64
 9   date           435735 non-null  datetime64[ns]
 10  year           435735 non-null  int64
dtypes: datetime64[ns](1), float64(6), int64(1), object(3)
memory usage: 39.9+ MB
```

```
df['type']
```

```
0         1.0
1         2.0
2         1.0
3         1.0
4         2.0
          ...
435734    6.0
435735    6.0
435736    6.0
435737    6.0
435738    6.0
Name: type, Length: 435735, dtype: float64
```

```
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
df['state'] =labelencoder.fit_transform(df['state'])
df.head()
```

	sampling_date	state	location	type	so2	no2	rspm	spm	pm2_5	date	year
0	February - M021990	0	Hyderabad	1.0	4.8	17.4	108.833091	220.78348	40.791467	1990-02-01	1990
1	February - M021990	0	Hyderabad	2.0	3.1	7.0	108.833091	220.78348	40.791467	1990-02-01	1990
2	February - M021990	0	Hyderabad	1.0	6.2	28.5	108.833091	220.78348	40.791467	1990-02-01	1990
3	March - M031990	0	Hyderabad	1.0	6.3	14.7	108.833091	220.78348	40.791467	1990-03-01	1990
4	March - M031990	0	Hyderabad	2.0	4.7	7.5	108.833091	220.78348	40.791467	1990-03-01	1990

One-hot encoding is used to convert categorical variables into a format that can be readily used by machine learning algorithms.

```
dfAndhra = df[df['state']==0]
dfAndhra
```

	sampling_date	state	location	type	so2	no2	rspm	spm	pm2_5	date	year
0	February - M021990	0	Hyderabad	1.0	4.8	17.4	108.833091	220.78348	40.791467	1990-02-01	1990

```
dfAndhra['location'].value_counts()
```

```
Hyderabad         7764
Visakhapatnam     7108
Vijayawada        2093
Chittoor          1003
Tirupati           986
Kurnool            857
Patancheru         698
Guntur             629
Nalgonda           618
Ramagundam         554
Nellore            408
Khammam            385
Warangal           336
Ananthapur         324
Ongole             317
Kadapa             316
Srikakulam         315
Rajahmundry        311
Eluru              300
Vishakhapatnam     297
Kakinada           288
Vizianagaram       282
Sangareddy          85
Karimnagar          67
Nizamabad           27
Name: location, dtype: int64
```

```
from sklearn.preprocessing import OneHotEncoder
onehotencoder = OneHotEncoder(sparse=False, handle_unknown='error', drop='first')
```

```
pd.DataFrame(onehotencoder.fit_transform(dfAndhra[['location']]))
```

	0	1	2	3	4	5	6	7	8	9	...	14	15	16	17	18	19	20	21	22	23
0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
26363	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
26364	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
26365	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
26366	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
26367	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

26368 rows × 24 columns

```
dfAndhra['location'].value_counts()
```

```
Hyderabad         7764
Visakhapatnam     7108
Vijayawada        2093
Chittoor          1003
Tirupati           986
Kurnool            857
Patancheru         698
Guntur             629
Nalgonda           618
Ramagundam         554
Nellore            408
Khammam            385
Warangal           336
Ananthapur         324
Ongole             317
```

```
Kadapa             316
Srikakulam         315
Rajahmundry        311
Eluru              300
Vishakhapatnam     297
Kakinada           288
Vizianagaram       282
Sangareddy          85
Karimnagar          67
Nizamabad           27
Name: location, dtype: int64
```

Error correction

```
df.isnull().sum()
```

```
sampling_date       0
state               0
location            0
type             5390
so2                 0
no2                 0
rspm                0
spm                 0
pm2_5               0
date                0
year                0
dtype: int64
```

```
df=df.fillna(df.median())
df.isnull().sum()
```

```
C:\Users\Kajal\AppData\Local\Temp\ipykernel_17868\1225245910.py:1: FutureWarning: DataFrame.mean and DataFrame.median with numeric_only=
  df=df.fillna(df.median())
C:\Users\Kajal\AppData\Local\Temp\ipykernel_17868\1225245910.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (
  df=df.fillna(df.median())
sampling_date    0
state            0
location         0
type             0
so2              0
no2              0
rspm             0
spm              0
pm2_5            0
date             0
year             0
dtype: int64
```

Detecting and Filtering Outliers

```
df.describe()
```

	state	type	so2	no2	rspm	spm	pm2_5	year
count	435735.000000	435735.000000	435735.000000	435735.000000	435735.000000	435735.00000	435735.000000	435735.000000
mean	17.966833	2.035042	10.829428	25.809659	108.833091	220.78348	40.791467	2009.534123
std	9.471742	1.136631	10.723716	18.155263	71.333594	102.14629	4.507577	4.791559
min	0.000000	1.000000	0.000000	0.000000	0.000000	0.00000	3.000000	1987.000000
25%	12.000000	1.000000	5.000000	14.000000	59.000000	203.00000	40.791467	2007.000000
50%	18.000000	2.000000	9.000000	22.300000	97.666667	220.78348	40.791467	2010.000000
75%	26.000000	2.000000	13.000000	32.000000	135.000000	220.78348	40.791467	2013.000000
max	33.000000	6.000000	909.000000	876.000000	6307.033333	3380.00000	504.000000	2015.000000

```
df[df['so2']>100]=0
```