

- a. Data cleaning
- b. Data integration
- c. Data transformation
- d. Error correcting

```
df.head()
```

		type	so2	no2	rspm	spm
0	Residential, Rural	and other Areas	4.8	17.4	NaN	NaN
1		Industrial Area	3.1	7.0	NaN	NaN
2	Residential, Rural	and other Areas	6.2	28.5	NaN	NaN
3	Residential, Rural	and other Areas	6.3	14.7	NaN	NaN
4		Industrial Area	4.7	7.5	NaN	NaN

```
df.info()
```

#	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.columns

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

```

Data Cleaning

```

# Change data type from float64 to float32 for Space Complexity
df['so2'] = df['so2'].astype('float32')
df['no2'] = df['no2'].astype('float32')
df['rspm'] = df['rspm'].astype('float32')
df['spm'] = df['spm'].astype('float32')
df['date'] = df['date'].astype('string')

```

```
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	float32
7	no2	419509 non-null	float32

```

8   rspm                395520 non-null float32
9   spm                 198355 non-null float32
10  location_monitoring_station 408251 non-null object
11  pm2_5               9314 non-null float64
12  date               435735 non-null string

```

```
dtypes: float32(4), float64(1), object(7), string(1)
```

```
memory usage: 36.6+ MB
```

```
df=df.drop_duplicates()
```

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

```

stn_code                144077
sampling_date              3
state                    0
location                 3
agency                 149466
type                    5357
so2                     34632
no2                     16222
rspm                    40035
spm                   236908
location_monitoring_station 27303
pm2_5                   425754
date                      7

```

```
dtype: int64
```

```
percent_missing = df.isnull().sum() * 100 / len(df)
```

```
percent_missing.sort_values(ascending=False)
```

```

pm2_5                97.859185
spm                 54.453097
agency              34.354630
stn_code            33.115973
rspm                9.202010
so2                 7.960135
location_monitoring_station 6.275571
no2                 3.728613
type                1.231302
date                0.001609
sampling_date       0.000690
location            0.000690
state               0.000000

```

```
dtype: float64
```

```

df=df.drop(['stn_code',
'agency', 'sampling_date', 'location_monitoring_station', 'pm2_5'], axis
= 1)

```

```
df.head()
```

	state	location	type	so2
no2 \				
0	Andhra Pradesh	Hyderabad	Residential, Rural and other Areas	4.8
17.4				
1	Andhra Pradesh	Hyderabad	Industrial Area	3.1
7.0				
2	Andhra Pradesh	Hyderabad	Residential, Rural and other Areas	6.2
28.5				
3	Andhra Pradesh	Hyderabad	Residential, Rural and other Areas	6.3
14.7				
4	Andhra Pradesh	Hyderabad	Industrial Area	4.7
7.5				

	rspm	spm	date
0	NaN	NaN	1990-02-01
1	NaN	NaN	1990-02-01
2	NaN	NaN	1990-02-01
3	NaN	NaN	1990-03-01
4	NaN	NaN	1990-03-01

```
df.columns
```

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

```
col_var = ['state', 'location', 'type', 'date']
```

```
col_num = ['so2', 'no2', 'rspm', 'spm']
```

```
for col in df.columns:
```

```
    if df[col].dtype == 'object' or df[col].dtype == 'string':
```

```
        df[col] = df[col].fillna(df[col].mode()[0])
```

```
    else:
```

```
        df[col] = df[col].fillna(df[col].mean())
```

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

state	0
location	0
type	0
so2	0
no2	0
rspm	0
spm	0
date	0

```
dtype: int64
```

```
df
```

	state	location
0	Andhra Pradesh	Hyderabad
1	Andhra Pradesh	Hyderabad

2	Andhra Pradesh	Hyderabad
3	Andhra Pradesh	Hyderabad
4	Andhra Pradesh	Hyderabad
...
435737	West Bengal	ULUBERIA
435738	West Bengal	ULUBERIA
435739	andaman-and-nicobar-islands	Guwahati
435740	Lakshadweep	Guwahati
435741	Tripura	Guwahati

	type	so2	no2
rspm \			
0	Residential, Rural and other Areas	4.800000	17.400000
108.871712			
1	Industrial Area	3.100000	7.000000
108.871712			
2	Residential, Rural and other Areas	6.200000	28.500000
108.871712			
3	Residential, Rural and other Areas	6.300000	14.700000
108.871712			
4	Industrial Area	4.700000	7.500000
108.871712			
...
...			
435737	RIRU0	22.000000	50.000000
143.000000			
435738	RIRU0	20.000000	46.000000
171.000000			
435739	Residential, Rural and other Areas	10.830467	25.823299
108.871712			
435740	Residential, Rural and other Areas	10.830467	25.823299
108.871712			
435741	Residential, Rural and other Areas	10.830467	25.823299
108.871712			

	spm	date
0	220.774796	1990-02-01
1	220.774796	1990-02-01
2	220.774796	1990-02-01
3	220.774796	1990-03-01
4	220.774796	1990-03-01
...
435737	220.774796	2015-12-24
435738	220.774796	2015-12-29
435739	220.774796	2015-03-19
435740	220.774796	2015-03-19
435741	220.774796	2015-03-19

[435068 rows x 8 columns]

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

```
state      0
location   0
type       0
so2        0
no2        0
rspm       0
spm        0
date       0
dtype: int64
```

Data integration

```
subSet1 = df[['state', 'type']]
subSet2 = df[['state', 'location']]
```

```
subSet1.head()
```

	state	type
0	Andhra Pradesh	Residential, Rural and other Areas
1	Andhra Pradesh	Industrial Area
2	Andhra Pradesh	Residential, Rural and other Areas
3	Andhra Pradesh	Residential, Rural and other Areas
4	Andhra Pradesh	Industrial Area

```
subSet2.head()
```

	state	location
0	Andhra Pradesh	Hyderabad
1	Andhra Pradesh	Hyderabad
2	Andhra Pradesh	Hyderabad
3	Andhra Pradesh	Hyderabad
4	Andhra Pradesh	Hyderabad

```
concatenated_df = pd.concat([subSet1, subSet2], axis=1)
```

```
concatenated_df
```

	state
type \	
0	Andhra Pradesh Residential, Rural and other Areas
1	Andhra Pradesh Industrial Area
2	Andhra Pradesh Residential, Rural and other Areas
3	Andhra Pradesh Residential, Rural and other Areas

```

4          Andhra Pradesh          Industrial
Area
...          ...          ..
.
435737          West Bengal
RIRU0
435738          West Bengal
RIRU0
435739  andaman-and-nicobar-islands  Residential, Rural and other
Areas
435740          Lakshadweep  Residential, Rural and other
Areas
435741          Tripura  Residential, Rural and other
Areas

          state  location
0          Andhra Pradesh  Hyderabad
1          Andhra Pradesh  Hyderabad
2          Andhra Pradesh  Hyderabad
3          Andhra Pradesh  Hyderabad
4          Andhra Pradesh  Hyderabad
...          ...          ...
435737          West Bengal  ULUBERIA
435738          West Bengal  ULUBERIA
435739  andaman-and-nicobar-islands  Guwahati
435740          Lakshadweep  Guwahati
435741          Tripura  Guwahati

[435068 rows x 4 columns]

```

Error Correcting

```

def remove_outliers(column):
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    threshold = 1.5 * IQR
    outlier_mask = (column < Q1 - threshold) | (column > Q3 +
threshold)
    return column[~outlier_mask]

df.columns

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

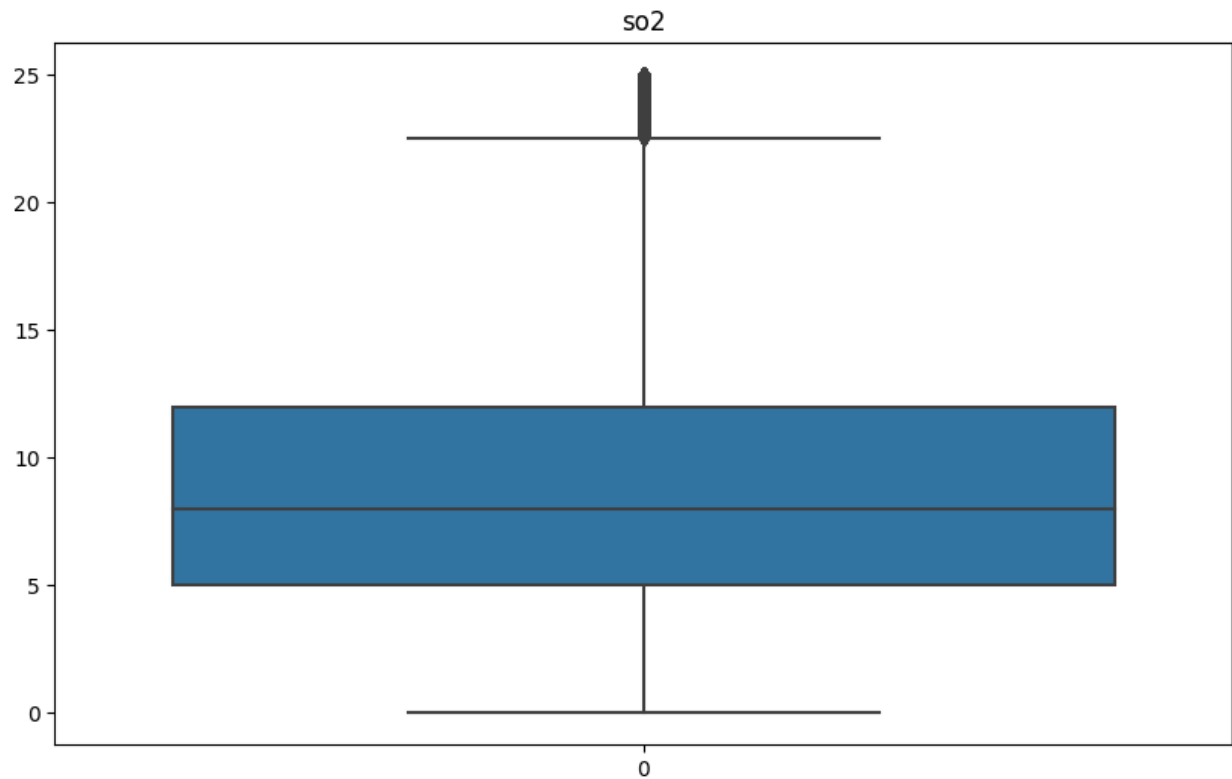
# Remove outliers for each column using a loop
col_name = ['so2', 'no2', 'rspm', 'spm']
for col in col_name:
    df[col] = remove_outliers(df[col])

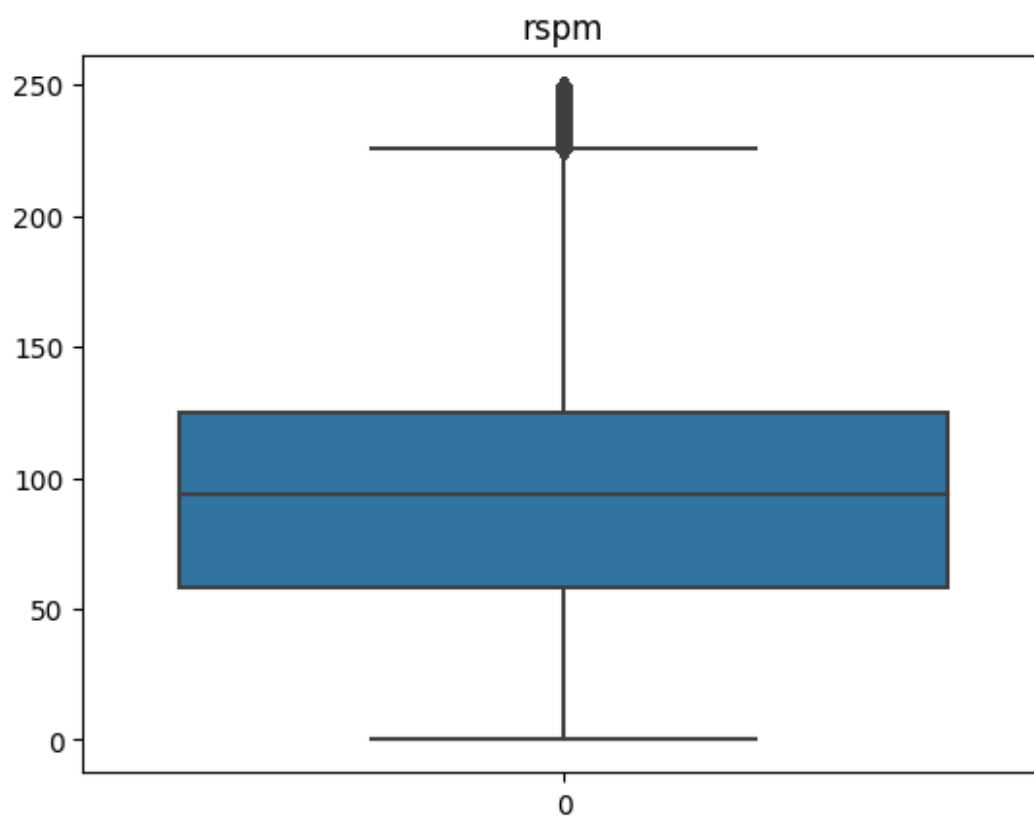
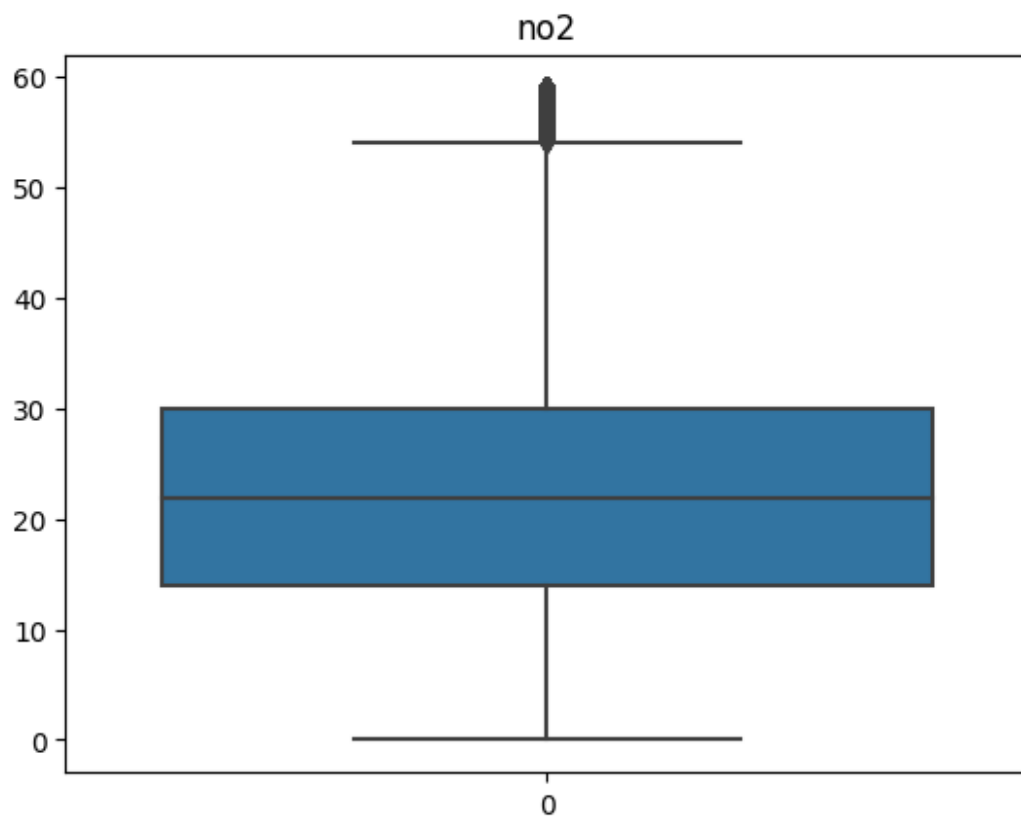
```

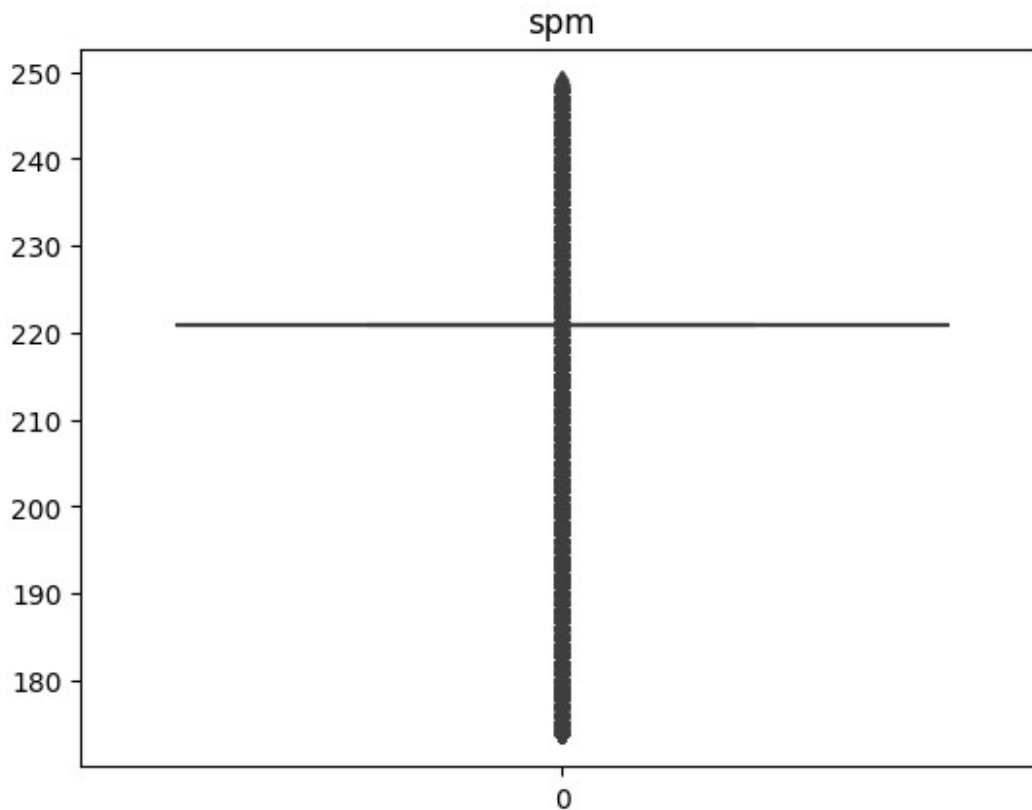
```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6)) # Adjust the figure size if needed

for col in col_name:
    sns.boxplot(data=df[col])
    plt.title(col)
    plt.show()
```







Data Transform

```
from sklearn.preprocessing import LabelEncoder

col_label= ['state','location','type']
# Initialize LabelEncoder

encoder = LabelEncoder()
# Iterate over columns
for col in df.columns:
    # Fit and transform the column
    df[col] = encoder.fit_transform(df[col])
```

df

	state	location	type	so2	no2	rspm	spm	date
0	0	114	6	446	1489	2030	464	213
1	0	114	1	197	250	2030	464	213
2	0	114	6	790	3096	2030	464	213
3	0	114	6	823	1144	2030	464	214
4	0	114	1	427	301	2030	464	214
...
435737	35	282	3	2888	5307	2534	464	5059
435738	35	282	3	2809	5113	3098	464	5064
435739	36	100	6	1638	2696	2030	464	4779

435740	17	100	6	1638	2696	2030	464	4779
435741	31	100	6	1638	2696	2030	464	4779

[435068 rows x 8 columns]