

Practical No. 7

Title: Data loading , storage and file formats

Name : Tavhare Ruchita Sharad

Roll No : 61

In [126... #Step 1 - Load the Sales Dataset.

In [128... import pandas as pd

In [130... df = pd.read_csv("SuperMarket Analysis.csv")

In [132... df.head()

Out[132...

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Sales
0	750-67-8428	Alex	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715
1	226-31-3081	Giza	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80.2200
2	631-41-3108	Alex	Yangon	Normal	Female	Home and lifestyle	46.33	7	16.2155	340.5255
3	123-19-1176	Alex	Yangon	Member	Female	Health and beauty	58.22	8	23.2880	489.0480
4	373-73-7910	Alex	Yangon	Member	Female	Sports and travel	86.31	7	30.2085	634.3785

In [134... df.head()

Out[134...

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Sales
0	750-67-8428	Alex	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715
1	226-31-3081	Giza	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80.2200
2	631-41-3108	Alex	Yangon	Normal	Female	Home and lifestyle	46.33	7	16.2155	340.5255
3	123-19-1176	Alex	Yangon	Member	Female	Health and beauty	58.22	8	23.2880	489.0480
4	373-73-7910	Alex	Yangon	Member	Female	Sports and travel	86.31	7	30.2085	634.3785

In [136... #Step 2- Explore the Structure and Content.

In [138... df.describe()

Out[138...]

	Unit price	Quantity	Tax 5%	Sales	cogs	gross margin percentage	gross income	
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1.000000e+03	1000.000000	10
mean	55.672130	5.510000	15.379369	322.966749	307.58738	4.761905e+00	15.379369	
std	26.494628	2.923431	11.708825	245.885335	234.17651	6.131498e-14	11.708825	
min	10.080000	1.000000	0.508500	10.678500	10.17000	4.761905e+00	0.508500	
25%	32.875000	3.000000	5.924875	124.422375	118.49750	4.761905e+00	5.924875	
50%	55.230000	5.000000	12.088000	253.848000	241.76000	4.761905e+00	12.088000	
75%	77.935000	8.000000	22.445250	471.350250	448.90500	4.761905e+00	22.445250	
max	99.960000	10.000000	49.650000	1042.650000	993.00000	4.761905e+00	49.650000	

In [140...]

`df.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 17 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Invoice ID       1000 non-null   object  
 1   Branch           1000 non-null   object  
 2   City              1000 non-null   object  
 3   Customer type    1000 non-null   object  
 4   Gender            1000 non-null   object  
 5   Product line     1000 non-null   object  
 6   Unit price       1000 non-null   float64 
 7   Quantity          1000 non-null   int64  
 8   Tax 5%            1000 non-null   float64 
 9   Sales             1000 non-null   float64 
 10  Date              1000 non-null   object  
 11  Time              1000 non-null   object  
 12  Payment           1000 non-null   object  
 13  cogs              1000 non-null   float64 
 14  gross margin percentage 1000 non-null   float64 
 15  gross income      1000 non-null   float64 
 16  Rating            1000 non-null   float64 
dtypes: float64(7), int64(1), object(9)
memory usage: 132.9+ KB
```

In [142...]

`#step 3. Perform data cleaning operations.`

In [144...]

`df.isnull().sum()`

```
Out[144... Invoice ID      0  
Branch          0  
City            0  
Customer type   0  
Gender          0  
Product line    0  
Unit price     0  
Quantity        0  
Tax 5%         0  
Sales           0  
Date            0  
Time            0  
Payment         0  
cogs            0  
gross margin percentage 0  
gross income    0  
Rating          0  
dtype: int64
```

```
In [146... df.duplicated().sum()
```

```
Out[146... 0
```

```
In [148... #Step 4 - Convert the Data into a Unified Format.
```

```
In [150... df.columns = [col.lower() for col in df.columns]  
df['date'] = pd.to_datetime(df['date'])  
df['unit price'] = df['unit price'].astype(float)  
df['quantity'] = df['quantity'].astype(int)  
df['sales'] = df['sales'].astype(float)
```

```
In [152... df.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 1000 entries, 0 to 999  
Data columns (total 17 columns):  
 #   Column           Non-Null Count Dtype  
 ---  --  
 0   invoice id       1000 non-null  object  
 1   branch           1000 non-null  object  
 2   city              1000 non-null  object  
 3   customer type    1000 non-null  object  
 4   gender            1000 non-null  object  
 5   product line     1000 non-null  object  
 6   unit price       1000 non-null  float64  
 7   quantity          1000 non-null  int32  
 8   tax 5%           1000 non-null  float64  
 9   sales             1000 non-null  float64  
 10  date              1000 non-null  datetime64[ns]  
 11  time              1000 non-null  object  
 12  payment           1000 non-null  object  
 13  cogs              1000 non-null  float64  
 14  gross margin percentage 1000 non-null  float64  
 15  gross income      1000 non-null  float64  
 16  rating            1000 non-null  float64  
dtypes: datetime64[ns](1), float64(7), int32(1), object(8)  
memory usage: 129.0+ KB
```

```
In [154... #Step 5 - Perform Data Transformation and Analyze the Data.
```

In [156... df['total_sales'] = df['unit price'] * df['quantity']

In [158... df

Out[158...

	invoice id	branch	city	customer type	gender	product line	unit price	quantity	tax 5%	sal
0	750-67-8428	Alex	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.97
1	226-31-3081	Giza	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80.22
2	631-41-3108	Alex	Yangon	Normal	Female	Home and lifestyle	46.33	7	16.2155	340.52
3	123-19-1176	Alex	Yangon	Member	Female	Health and beauty	58.22	8	23.2880	489.04
4	373-73-7910	Alex	Yangon	Member	Female	Sports and travel	86.31	7	30.2085	634.37
...
995	233-67-5758	Giza	Naypyitaw	Normal	Male	Health and beauty	40.35	1	2.0175	42.36
996	303-96-2227	Cairo	Mandalay	Normal	Female	Home and lifestyle	97.38	10	48.6900	1022.49
997	727-02-1313	Alex	Yangon	Member	Male	Food and beverages	31.84	1	1.5920	33.43
998	347-56-2442	Alex	Yangon	Normal	Male	Home and lifestyle	65.82	1	3.2910	69.11
999	849-09-3807	Alex	Yangon	Member	Female	Fashion accessories	88.34	7	30.9190	649.29

1000 rows × 18 columns

In [162... total_sales = df['total_sales']
total_sales.sum()

Out[162... 307587.38

In [166... average_order_value = df['total_sales'].mean()
average_order_value

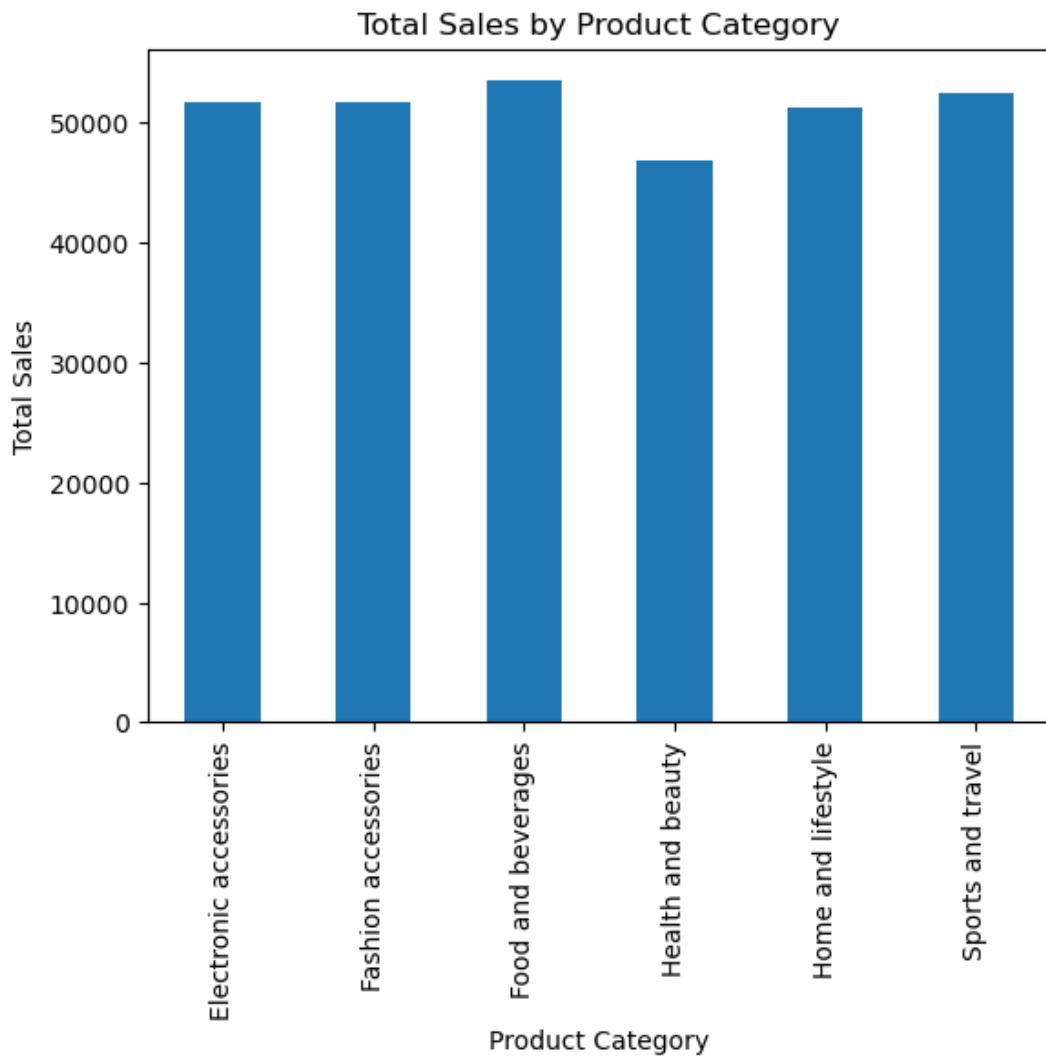
Out[166... 307.58738

In [170... category_sales = df.groupby('product line')['total_sales'].sum()
category_sales

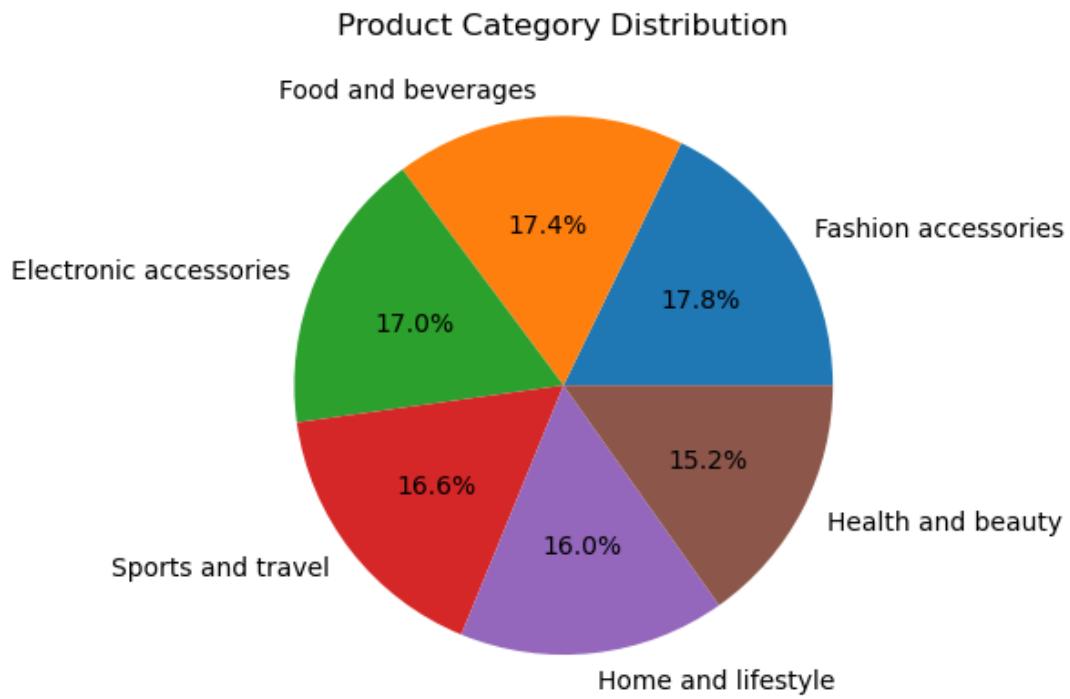
```
Out[170... product line
Electronic accessories      51750.03
Fashion accessories        51719.90
Food and beverages         53471.28
Health and beauty          46851.18
Home and lifestyle          51297.06
Sports and travel           52497.93
Name: total_sales, dtype: float64
```

```
In [172... #Step 6 -Create Visualizations.
```

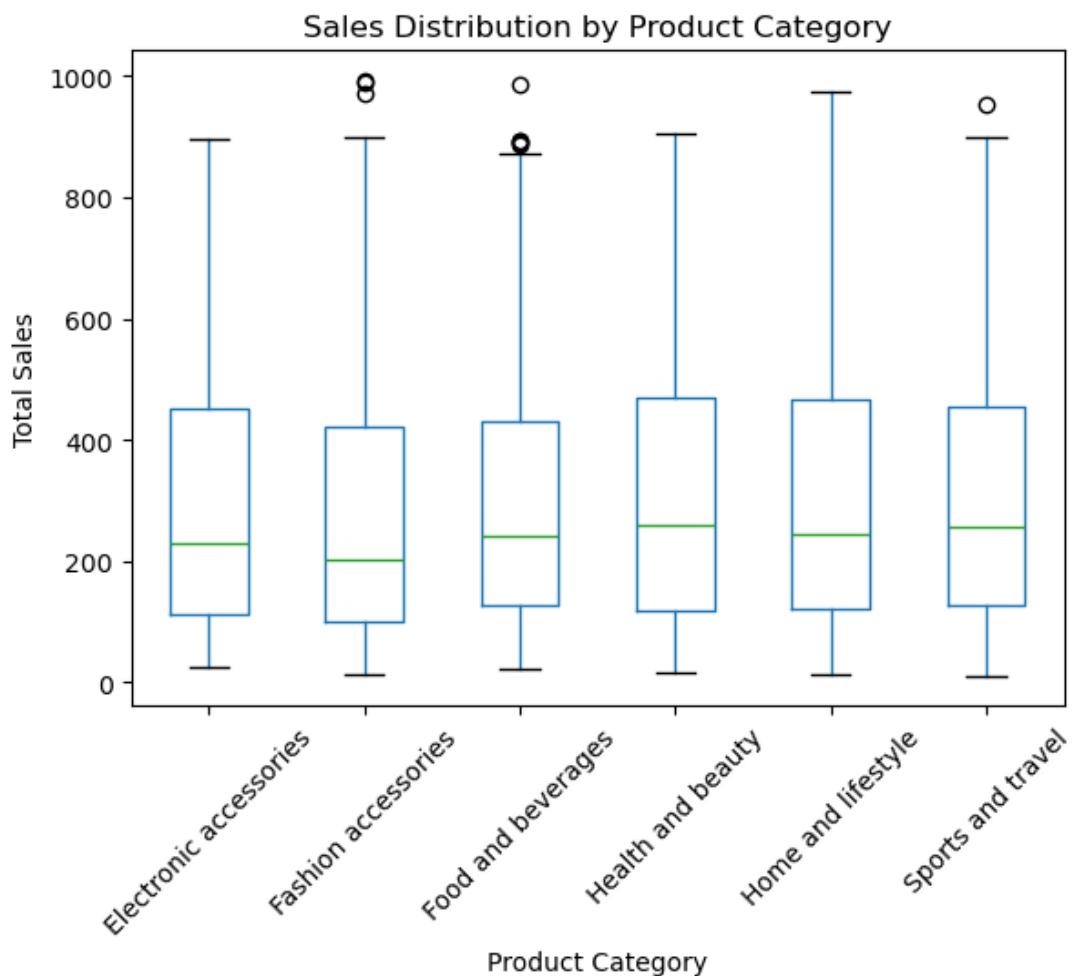
```
In [174... import matplotlib.pyplot as plt
category_sales.plot(kind='bar', title='Total Sales by Product Category')
plt.xlabel('Product Category')
plt.ylabel('Total Sales')
plt.show()
```



```
In [176... category_distribution = df['product line'].value_counts()
category_distribution.plot(kind='pie', title='Product Category Distribution',
                           autopct='%1.1f%%')
plt.ylabel('')
plt.show()
```



```
In [180]: df.boxplot(column='total_sales', by='product_line', grid=False, rot=45)
plt.title('Sales Distribution by Product Category')
plt.suptitle('')
plt.xlabel('Product Category')
plt.ylabel('Total Sales')
plt.show()
```



In []:

Practical No. 8

Title: Interacting with Web APIs

Name : Tavhare Ruchita Sharad

Roll No : 61

```
In [7]: import requests
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime
import seaborn as sns
```

```
In [9]: # API details
api_key = 'c15c676d2f9d0dada63d7fa10c76ce01'
location = 'India'
url = f'http://api.openweathermap.org/data/2.5/forecast?q={location}&appid={api_key}&units=me
```

```
In [11]: # Fetch data
response = requests.get(url)
data = response.json()

if response.status_code == 200:
    print("Data retrieved successfully for India")
else:
    print(f"Error: {data.get('message', 'Failed to retrieve data')}")
```

```
Data retrieved successfully for India
```

```
Out[11]: {'cod': '200',
'message': 0,
'cnt': 40,
'list': [{'dt': 1760238000,
'main': {'temp': 5.14,
'feels_like': 5.14,
'temp_min': 5.14,
'temp_max': 5.14,
'pressure': 1028,
'sea_level': 1028,
'grnd_level': 825,
'humidity': 86,
'temp_kf': 0},
'weather': [{"id": 800,
'main': 'Clear',
'description': 'clear sky',
'icon': '01n'}],
'clouds': {'all': 1},
'wind': {'speed': 0.5, 'deg': 322, 'gust': 0.44},
'vesibility': 10000,
'pop': 0,
'sys': {'pod': 'n'},
'dt_txt': '2025-10-12 03:00:00'},
{'dt': 1760248800,
'main': {'temp': 4.88,
'feels_like': 4.88,
'temp_min': 4.36,
'temp_max': 4.88,
'pressure': 1028,
'sea_level': 1028,
'grnd_level': 825,
'humidity': 83,
'temp_kf': 0.52},
'weather': [{"id": 800,
'main': 'Clear',
'description': 'clear sky',
'icon': '01d'}],
'clouds': {'all': 1},
'wind': {'speed': 0.64, 'deg': 300, 'gust': 0.53},
'vesibility': 10000,
'pop': 0,
'sys': {'pod': 'd'},
'dt_txt': '2025-10-12 06:00:00'},
{'dt': 1760259600,
'main': {'temp': 9.49,
'feels_like': 9.49,
'temp_min': 9.49,
'temp_max': 11.67,
'pressure': 1026,
'sea_level': 1026,
'grnd_level': 825,
'humidity': 59,
'temp_kf': -2.18},
'weather': [{"id": 800,
'main': 'Clear',
'description': 'clear sky',
'icon': '01d'}],
'clouds': {'all': 0},
'wind': {'speed': 0.9, 'deg': 251, 'gust': 0.9},
'vesibility': 10000,
'pop': 0,
```

```
'sys': {'pod': 'd'},
'dt_txt': '2025-10-12 09:00:00'},
{'dt': 1760270400,
'main': {'temp': 14.92,
'feels_like': 13.33,
'temp_min': 14.92,
'temp_max': 14.92,
'pressure': 1022,
'sea_level': 1022,
'grnd_level': 823,
'humidity': 33,
'temp_kf': 0},
'weather': [{"id": 800,
'main': 'Clear',
'description': 'clear sky',
'icon': '01d'}],
'clouds': {'all': 0},
'wind': {'speed': 1.14, 'deg': 261, 'gust': 1.39},
'vesibility': 10000,
'pop': 0,
'sys': {'pod': 'd'},
'dt_txt': '2025-10-12 12:00:00'},
{'dt': 1760281200,
'main': {'temp': 13.52,
'feels_like': 12.08,
'temp_min': 13.52,
'temp_max': 13.52,
'pressure': 1021,
'sea_level': 1021,
'grnd_level': 822,
'humidity': 44,
'temp_kf': 0},
'weather': [{"id": 800,
'main': 'Clear',
'description': 'clear sky',
'icon': '01d'}],
'clouds': {'all': 0},
'wind': {'speed': 0.93, 'deg': 237, 'gust': 1.33},
'vesibility': 10000,
'pop': 0,
'sys': {'pod': 'd'},
'dt_txt': '2025-10-12 15:00:00'},
{'dt': 1760292000,
'main': {'temp': 9.37,
'feels_like': 9.37,
'temp_min': 9.37,
'temp_max': 9.37,
'pressure': 1024,
'sea_level': 1024,
'grnd_level': 823,
'humidity': 89,
'temp_kf': 0},
'weather': [{"id": 800,
'main': 'Clear',
'description': 'clear sky',
'icon': '01n'}],
'clouds': {'all': 0},
'wind': {'speed': 1.14, 'deg': 84, 'gust': 1.03},
'vesibility': 10000,
'pop': 0,
'sys': {'pod': 'n'},
'dt_txt': '2025-10-12 18:00:00'},
```

```
{'dt': 1760302800,
'main': {'temp': 8.71,
'feels_like': 8.71,
'temp_min': 8.71,
'temp_max': 8.71,
'pressure': 1025,
'sea_level': 1025,
'grnd_level': 823,
'humidity': 87,
'temp_kf': 0},
'weather': [ {'id': 801,
'main': 'Clouds',
'description': 'few clouds',
'icon': '02n'}],
'clouds': {'all': 11},
'wind': {'speed': 0.77, 'deg': 11, 'gust': 0.76},
'vesibility': 10000,
'pop': 0,
'sys': {'pod': 'n'},
'dt_txt': '2025-10-12 21:00:00'},
{'dt': 1760313600,
'main': {'temp': 7.19,
'feels_like': 7.19,
'temp_min': 7.19,
'temp_max': 7.19,
'pressure': 1024,
'sea_level': 1024,
'grnd_level': 822,
'humidity': 80,
'temp_kf': 0},
'weather': [ {'id': 801,
'main': 'Clouds',
'description': 'few clouds',
'icon': '02n'}],
'clouds': {'all': 23},
'wind': {'speed': 0.91, 'deg': 21, 'gust': 0.9},
'vesibility': 10000,
'pop': 0,
'sys': {'pod': 'n'},
'dt_txt': '2025-10-13 00:00:00'},
{'dt': 1760324400,
'main': {'temp': 6.82,
'feels_like': 6.82,
'temp_min': 6.82,
'temp_max': 6.82,
'pressure': 1024,
'sea_level': 1024,
'grnd_level': 821,
'humidity': 69,
'temp_kf': 0},
'weather': [ {'id': 804,
'main': 'Clouds',
'description': 'overcast clouds',
'icon': '04n'}],
'clouds': {'all': 87},
'wind': {'speed': 0.8, 'deg': 37, 'gust': 0.78},
'vesibility': 10000,
'pop': 0,
'sys': {'pod': 'n'},
'dt_txt': '2025-10-13 03:00:00'},
{'dt': 1760335200,
'main': {'temp': 6.41,
```

```
'feels_like': 6.41,
'temp_min': 6.41,
'temp_max': 6.41,
'pressure': 1024,
'sea_level': 1024,
'grnd_level': 821,
'humidity': 60,
'temp_kf': 0},
'weather': [{"id": 804,
  'main': 'Clouds',
  'description': 'overcast clouds',
  'icon': '04d'}],
'clouds': {'all': 88},
'wind': {'speed': 0.84, 'deg': 38, 'gust': 0.87},
'vesibility': 10000,
'pop': 0,
'sys': {'pod': 'd'},
'dt_txt': '2025-10-13 06:00:00'},
{'dt': 1760346000,
'main': {'temp': 14.84,
  'feels_like': 13.24,
  'temp_min': 14.84,
  'temp_max': 14.84,
  'pressure': 1021,
  'sea_level': 1021,
  'grnd_level': 821,
  'humidity': 33,
  'temp_kf': 0},
'weather': [{"id": 800,
  'main': 'Clear',
  'description': 'clear sky',
  'icon': '01d'}],
'clouds': {'all': 7},
'wind': {'speed': 0.38, 'deg': 117, 'gust': 0.66},
'vesibility': 10000,
'pop': 0,
'sys': {'pod': 'd'},
'dt_txt': '2025-10-13 09:00:00'},
{'dt': 1760356800,
'main': {'temp': 17.52,
  'feels_like': 16.06,
  'temp_min': 17.52,
  'temp_max': 17.52,
  'pressure': 1019,
  'sea_level': 1019,
  'grnd_level': 820,
  'humidity': 28,
  'temp_kf': 0},
'weather': [{"id": 800,
  'main': 'Clear',
  'description': 'clear sky',
  'icon': '01d'}],
'clouds': {'all': 4},
'wind': {'speed': 1.05, 'deg': 206, 'gust': 1.23},
'vesibility': 10000,
'pop': 0,
'sys': {'pod': 'd'},
'dt_txt': '2025-10-13 12:00:00'},
{'dt': 1760367600,
'main': {'temp': 15.49,
  'feels_like': 14.22,
  'temp_min': 15.49,
```

```
'temp_max': 15.49,
'pressure': 1018,
'sea_level': 1018,
'grnd_level': 820,
'humidity': 43,
'temp_kf': 0},
'weather': [{"id": 800,
  'main': 'Clear',
  'description': 'clear sky',
  'icon': '01d'}],
'clouds': {"all": 7},
'wind': {"speed": 1.12, "deg": 189, "gust": 1.23},
'vesibility': 10000,
'pop': 0,
'sys': {"pod": "d"},
'dt_txt': '2025-10-13 15:00:00'},
{'dt': 1760378400,
'main': {"temp": 8.69,
  'feels_like': 8.29,
  'temp_min': 8.69,
  'temp_max': 8.69,
  'pressure': 1022,
  'sea_level': 1022,
  'grnd_level': 821,
  'humidity': 91,
  'temp_kf': 0},
'weather': [{"id": 500,
  'main': 'Rain',
  'description': 'light rain',
  'icon': '10n'}],
'clouds': {"all": 25},
'wind': {"speed": 1.38, "deg": 95, "gust": 1.37},
'vesibility': 10000,
'pop': 0.2,
'rain': {"3h": 0.13},
'sys': {"pod": "n"},
'dt_txt': '2025-10-13 18:00:00'},
{'dt': 1760389200,
'main': {"temp": 7.8,
  'feels_like': 7.8,
  'temp_min': 7.8,
  'temp_max': 7.8,
  'pressure': 1023,
  'sea_level': 1023,
  'grnd_level': 821,
  'humidity': 91,
  'temp_kf': 0},
'weather': [{"id": 802,
  'main': 'Clouds',
  'description': 'scattered clouds',
  'icon': '03n'}],
'clouds': {"all": 31},
'wind': {"speed": 0.84, "deg": 50, "gust": 0.77},
'vesibility': 10000,
'pop': 0,
'sys': {"pod": "n"},
'dt_txt': '2025-10-13 21:00:00'},
{'dt': 1760400000,
'main': {"temp": 8.92,
  'feels_like': 8.92,
  'temp_min': 8.92,
  'temp_max': 8.92,
```

```
'pressure': 1022,
'sea_level': 1022,
'grnd_level': 820,
'humidity': 86,
'temp_kf': 0},
'weather': [{id': 803,
'main': 'Clouds',
'description': 'broken clouds',
'icon': '04n'}],
'clouds': {'all': 59},
'wind': {'speed': 0.49, 'deg': 346, 'gust': 0.61},
'vesibility': 10000,
'pop': 0,
'sys': {'pod': 'n'},
'dt_txt': '2025-10-14 00:00:00'},
{'dt': 1760410800,
'main': {'temp': 8.56,
'feels_like': 8.56,
'temp_min': 8.56,
'temp_max': 8.56,
'pressure': 1021,
'sea_level': 1021,
'grnd_level': 820,
'humidity': 85,
'temp_kf': 0},
'weather': [{id': 804,
'main': 'Clouds',
'description': 'overcast clouds',
'icon': '04n'}],
'clouds': {'all': 100},
'wind': {'speed': 0.39, 'deg': 359, 'gust': 0.68},
'vesibility': 10000,
'pop': 0,
'sys': {'pod': 'n'},
'dt_txt': '2025-10-14 03:00:00'},
{'dt': 1760421600,
'main': {'temp': 8.55,
'feels_like': 8.55,
'temp_min': 8.55,
'temp_max': 8.55,
'pressure': 1021,
'sea_level': 1021,
'grnd_level': 820,
'humidity': 82,
'temp_kf': 0},
'weather': [{id': 804,
'main': 'Clouds',
'description': 'overcast clouds',
'icon': '04d'}],
'clouds': {'all': 100},
'wind': {'speed': 0.26, 'deg': 360, 'gust': 0.64},
'vesibility': 10000,
'pop': 0,
'sys': {'pod': 'd'},
'dt_txt': '2025-10-14 06:00:00'},
{'dt': 1760432400,
'main': {'temp': 10.9,
'feels_like': 9.8,
'temp_min': 10.9,
'temp_max': 10.9,
'pressure': 1021,
'sea_level': 1021,
```

```
'grnd_level': 820,
'humidity': 67,
'temp_kf': 0},
'weather': [{"id": 804,
'main': 'Clouds',
'description': 'overcast clouds',
'icon': '04d'}],
'clouds': {'all': 100},
'wind': {'speed': 0.29, 'deg': 110, 'gust': 0.61},
'vesibility': 10000,
'pop': 0,
'sys': {'pod': 'd'},
'dt_txt': '2025-10-14 09:00:00'},
{'dt': 1760443200,
'main': {'temp': 13.24,
'feels_like': 12,
'temp_min': 13.24,
'temp_max': 13.24,
'pressure': 1019,
'sea_level': 1019,
'grnd_level': 819,
'humidity': 53,
'temp_kf': 0},
'weather': [{"id": 804,
'main': 'Clouds',
'description': 'overcast clouds',
'icon': '04d'}],
'clouds': {'all': 100},
'wind': {'speed': 0.16, 'deg': 300, 'gust': 0.88},
'vesibility': 10000,
'pop': 0,
'sys': {'pod': 'd'},
'dt_txt': '2025-10-14 12:00:00'},
{'dt': 1760454000,
'main': {'temp': 11.4,
'feels_like': 10.32,
'temp_min': 11.4,
'temp_max': 11.4,
'pressure': 1019,
'sea_level': 1019,
'grnd_level': 819,
'humidity': 66,
'temp_kf': 0},
'weather': [{"id": 804,
'main': 'Clouds',
'description': 'overcast clouds',
'icon': '04d'}],
'clouds': {'all': 100},
'wind': {'speed': 1.16, 'deg': 80, 'gust': 1.3},
'vesibility': 10000,
'pop': 0.1,
'sys': {'pod': 'd'},
'dt_txt': '2025-10-14 15:00:00'},
{'dt': 1760464800,
'main': {'temp': 8.11,
'feels_like': 8.11,
'temp_min': 8.11,
'temp_max': 8.11,
'pressure': 1021,
'sea_level': 1021,
'grnd_level': 820,
'humidity': 83,
```

```
'temp_kf': 0},
'weather': [{'id': 500,
  'main': 'Rain',
  'description': 'light rain',
  'icon': '10n'}],
'clouds': {'all': 85},
'wind': {'speed': 1.14, 'deg': 64, 'gust': 1.09},
'vesibility': 10000,
'pop': 0.26,
'rain': {'3h': 0.1},
'sys': {'pod': 'n'},
'dt_txt': '2025-10-14 18:00:00'},
{'dt': 1760475600,
  'main': {'temp': 5.43,
    'feels_like': 5.43,
    'temp_min': 5.43,
    'temp_max': 5.43,
    'pressure': 1023,
    'sea_level': 1023,
    'grnd_level': 820,
    'humidity': 93,
    'temp_kf': 0},
  'weather': [{'id': 803,
    'main': 'Clouds',
    'description': 'broken clouds',
    'icon': '04n'}],
  'clouds': {'all': 52},
  'wind': {'speed': 1.23, 'deg': 52, 'gust': 1.04},
  'vesibility': 10000,
  'pop': 0,
  'sys': {'pod': 'n'},
  'dt_txt': '2025-10-14 21:00:00'},
{'dt': 1760486400,
  'main': {'temp': 4.73,
    'feels_like': 4.73,
    'temp_min': 4.73,
    'temp_max': 4.73,
    'pressure': 1024,
    'sea_level': 1024,
    'grnd_level': 820,
    'humidity': 92,
    'temp_kf': 0},
  'weather': [{'id': 803,
    'main': 'Clouds',
    'description': 'broken clouds',
    'icon': '04n'}],
  'clouds': {'all': 70},
  'wind': {'speed': 1.04, 'deg': 36, 'gust': 0.97},
  'vesibility': 10000,
  'pop': 0,
  'sys': {'pod': 'n'},
  'dt_txt': '2025-10-15 00:00:00'},
{'dt': 1760497200,
  'main': {'temp': 3.87,
    'feels_like': 3.87,
    'temp_min': 3.87,
    'temp_max': 3.87,
    'pressure': 1024,
    'sea_level': 1024,
    'grnd_level': 819,
    'humidity': 91,
    'temp_kf': 0},
```

```
'weather': [ {'id': 804,
    'main': 'Clouds',
    'description': 'overcast clouds',
    'icon': '04n'}],
'clouds': { 'all': 86},
'wind': { 'speed': 0.96, 'deg': 28, 'gust': 0.93},
'vesibility': 10000,
'pop': 0,
'sys': { 'pod': 'n'},
'dt_txt': '2025-10-15 03:00:00'},
{'dt': 1760508000,
    'main': { 'temp': 3.95,
        'feels_like': 3.95,
        'temp_min': 3.95,
        'temp_max': 3.95,
        'pressure': 1024,
        'sea_level': 1024,
        'grnd_level': 819,
        'humidity': 88,
        'temp_kf': 0},
    'weather': [ {'id': 803,
        'main': 'Clouds',
        'description': 'broken clouds',
        'icon': '04d'}],
'clouds': { 'all': 71},
'wind': { 'speed': 1.05, 'deg': 42, 'gust': 1.01},
'vesibility': 10000,
'pop': 0,
'sys': { 'pod': 'd'},
'dt_txt': '2025-10-15 06:00:00'},
{'dt': 1760518800,
    'main': { 'temp': 11.7,
        'feels_like': 10.39,
        'temp_min': 11.7,
        'temp_max': 11.7,
        'pressure': 1020,
        'sea_level': 1020,
        'grnd_level': 820,
        'humidity': 56,
        'temp_kf': 0},
    'weather': [ {'id': 801,
        'main': 'Clouds',
        'description': 'few clouds',
        'icon': '02d'}],
'clouds': { 'all': 20},
'wind': { 'speed': 0.48, 'deg': 127, 'gust': 0.3},
'vesibility': 10000,
'pop': 0,
'sys': { 'pod': 'd'},
'dt_txt': '2025-10-15 09:00:00'},
{'dt': 1760529600,
    'main': { 'temp': 13.41,
        'feels_like': 12.06,
        'temp_min': 13.41,
        'temp_max': 13.41,
        'pressure': 1019,
        'sea_level': 1019,
        'grnd_level': 819,
        'humidity': 48,
        'temp_kf': 0},
    'weather': [ {'id': 802,
        'main': 'Clouds',
```

```
'description': 'scattered clouds',
'icon': '03d'}],
'clouds': {'all': 38},
'wind': {'speed': 1.16, 'deg': 195, 'gust': 1.52},
'vesibility': 10000,
'pop': 0,
'sys': {'pod': 'd'},
'dt_txt': '2025-10-15 12:00:00'},
{'dt': 1760540400,
'main': {'temp': 11.29,
'feels_like': 10.17,
'temp_min': 11.29,
'temp_max': 11.29,
'pressure': 1019,
'sea_level': 1019,
'grnd_level': 819,
'humidity': 65,
'temp_kf': 0},
'weather': [{"id": 500,
'main': 'Rain',
'description': 'light rain',
'icon': '10d'}],
'clouds': {'all': 97},
'wind': {'speed': 1.53, 'deg': 125, 'gust': 1.24},
'vesibility': 10000,
'pop': 0.2,
'rain': {'3h': 0.24},
'sys': {'pod': 'd'},
'dt_txt': '2025-10-15 15:00:00'},
{'dt': 1760551200,
'main': {'temp': 7.8,
'feels_like': 7.32,
'temp_min': 7.8,
'temp_max': 7.8,
'pressure': 1021,
'sea_level': 1021,
'grnd_level': 820,
'humidity': 92,
'temp_kf': 0},
'weather': [{"id": 500,
'main': 'Rain',
'description': 'light rain',
'icon': '10n'}],
'clouds': {'all': 93},
'wind': {'speed': 1.35, 'deg': 98, 'gust': 1.64},
'vesibility': 10000,
'pop': 0.2,
'rain': {'3h': 0.19},
'sys': {'pod': 'n'},
'dt_txt': '2025-10-15 18:00:00'},
{'dt': 1760562000,
'main': {'temp': 7.98,
'feels_like': 7.98,
'temp_min': 7.98,
'temp_max': 7.98,
'pressure': 1022,
'sea_level': 1022,
'grnd_level': 820,
'humidity': 92,
'temp_kf': 0},
'weather': [{"id": 500,
'main': 'Rain',
```

```
'description': 'light rain',
'icon': '10n'}],
'clouds': {'all': 100},
'wind': {'speed': 0.83, 'deg': 121, 'gust': 1},
'vesibility': 10000,
'pop': 0.2,
'rain': {'3h': 0.14},
'sys': {'pod': 'n'},
'dt_txt': '2025-10-15 21:00:00'},
{'dt': 1760572800,
'main': {'temp': 7.31,
'feels_like': 7.31,
'temp_min': 7.31,
'temp_max': 7.31,
'pressure': 1022,
'sea_level': 1022,
'grnd_level': 820,
'humidity': 93,
'temp_kf': 0},
'weather': [{"id": 500,
'main': 'Rain',
'description': 'light rain',
'icon': '10n'}],
'clouds': {'all': 96},
'wind': {'speed': 0.43, 'deg': 168, 'gust': 0.53},
'vesibility': 10000,
'pop': 0.2,
'rain': {'3h': 0.12},
'sys': {'pod': 'n'},
'dt_txt': '2025-10-16 00:00:00'},
{'dt': 1760583600,
'main': {'temp': 6.16,
'feels_like': 6.16,
'temp_min': 6.16,
'temp_max': 6.16,
'pressure': 1022,
'sea_level': 1022,
'grnd_level': 819,
'humidity': 92,
'temp_kf': 0},
'weather': [{"id": 803,
'main': 'Clouds',
'description': 'broken clouds',
'icon': '04n'}],
'clouds': {'all': 65},
'wind': {'speed': 0.7, 'deg': 7, 'gust': 0.68},
'vesibility': 10000,
'pop': 0,
'sys': {'pod': 'n'},
'dt_txt': '2025-10-16 03:00:00'},
{'dt': 1760594400,
'main': {'temp': 5.09,
'feels_like': 5.09,
'temp_min': 5.09,
'temp_max': 5.09,
'pressure': 1022,
'sea_level': 1022,
'grnd_level': 819,
'humidity': 92,
'temp_kf': 0},
'weather': [{"id": 803,
'main': 'Clouds',
```

```
'description': 'broken clouds',
'icon': '04d'}],
'clouds': {'all': 54},
'wind': {'speed': 0.71, 'deg': 329, 'gust': 0.74},
'vesibility': 10000,
'pop': 0,
'sys': {'pod': 'd'},
'dt_txt': '2025-10-16 06:00:00'},
{'dt': 1760605200,
'main': {'temp': 10.73,
'feels_like': 9.63,
'temp_min': 10.73,
'temp_max': 10.73,
'pressure': 1020,
'sea_level': 1020,
'grnd_level': 819,
'humidity': 68,
'temp_kf': 0},
'weather': [{"id": 801,
'main': 'Clouds',
'description': 'few clouds',
'icon': '02d'}],
'clouds': {'all': 16},
'wind': {'speed': 0.79, 'deg': 228, 'gust': 0.82},
'vesibility': 10000,
'pop': 0,
'sys': {'pod': 'd'},
'dt_txt': '2025-10-16 09:00:00'},
{'dt': 1760616000,
'main': {'temp': 13.66,
'feels_like': 12.36,
'temp_min': 13.66,
'temp_max': 13.66,
'pressure': 1018,
'sea_level': 1018,
'grnd_level': 819,
'humidity': 49,
'temp_kf': 0},
'weather': [{"id": 801,
'main': 'Clouds',
'description': 'few clouds',
'icon': '02d'}],
'clouds': {'all': 18},
'wind': {'speed': 1.24, 'deg': 218, 'gust': 1.51},
'vesibility': 10000,
'pop': 0,
'sys': {'pod': 'd'},
'dt_txt': '2025-10-16 12:00:00'},
{'dt': 1760626800,
'main': {'temp': 11.55,
'feels_like': 10.41,
'temp_min': 11.55,
'temp_max': 11.55,
'pressure': 1018,
'sea_level': 1018,
'grnd_level': 818,
'humidity': 63,
'temp_kf': 0},
'weather': [{"id": 803,
'main': 'Clouds',
'description': 'broken clouds',
'icon': '04d'}],
```

```
'clouds': {'all': 65},
'wind': {'speed': 1.94, 'deg': 177, 'gust': 1.6},
'vesibility': 10000,
'pop': 0,
'sys': {'pod': 'd'},
'dt_txt': '2025-10-16 15:00:00'},
{'dt': 1760637600,
'main': {'temp': 7.49,
'feels_like': 7.49,
'temp_min': 7.49,
'temp_max': 7.49,
'pressure': 1021,
'sea_level': 1021,
'grnd_level': 819,
'humidity': 87,
'temp_kf': 0},
'weather': [{"id": 803,
'main': 'Clouds',
'description': 'broken clouds',
'icon': '04n'}],
'clouds': {'all': 68},
'wind': {'speed': 0.96, 'deg': 119, 'gust': 1.23},
'vesibility': 10000,
'pop': 0,
'sys': {'pod': 'n'},
'dt_txt': '2025-10-16 18:00:00'},
{'dt': 1760648400,
'main': {'temp': 7.25,
'feels_like': 7.25,
'temp_min': 7.25,
'temp_max': 7.25,
'pressure': 1022,
'sea_level': 1022,
'grnd_level': 820,
'humidity': 86,
'temp_kf': 0},
'weather': [{"id": 804,
'main': 'Clouds',
'description': 'overcast clouds',
'icon': '04n'}],
'clouds': {'all': 88},
'wind': {'speed': 0.19, 'deg': 102, 'gust': 0.23},
'vesibility': 10000,
'pop': 0,
'sys': {'pod': 'n'},
'dt_txt': '2025-10-16 21:00:00'},
{'dt': 1760659200,
'main': {'temp': 7.99,
'feels_like': 7.99,
'temp_min': 7.99,
'temp_max': 7.99,
'pressure': 1021,
'sea_level': 1021,
'grnd_level': 819,
'humidity': 81,
'temp_kf': 0},
'weather': [{"id": 804,
'main': 'Clouds',
'description': 'overcast clouds',
'icon': '04n'}],
'clouds': {'all': 94},
'wind': {'speed': 0.5, 'deg': 292, 'gust': 0.58},
```

```
'visibility': 10000,
'pop': 0,
'sys': {'pod': 'n'},
'dt_txt': '2025-10-17 00:00:00}],
'city': {'id': 3168508,
'name': 'Innichen',
'coord': {'lat': 46.7406, 'lon': 12.2797},
'country': 'IT',
'population': 3107,
'timezone': 7200,
'sunrise': 1760246652,
'sunset': 1760286611}}
```

Step No. 03 - Extract Relevant Weather Attributes.

```
In [14]: # Parse weather data
weather_list = data['list']
weather_data = {
    'datetime': [],
    'temperature': [],
    'humidity': [],
    'wind_speed': [],
    'precipitation': []
}
for entry in weather_list:
    weather_data['datetime'].append(datetime.fromtimestamp(entry['dt']))
    weather_data['temperature'].append(entry['main']['temp'])
    weather_data['humidity'].append(entry['main']['humidity'])
    weather_data['wind_speed'].append(entry['wind']['speed'])
    precipitation = entry.get('rain', {}).get('3h', 0)
    weather_data['precipitation'].append(precipitation)
```

```
In [16]: # Convert to DataFrame
df = pd.DataFrame(weather_data)
```

Step No. 04 - Clean and Preprocess the Data.

```
In [19]: print(df.isnull().sum())
      datetime      0
      temperature     0
      humidity       0
      wind_speed      0
      precipitation    0
      dtype: int64
```

Step No. 05 - Perform Data Modelling.

```
In [22]: avg_temp = df['temperature'].mean()
avg_temp
```

Out[22]: 9.1965

```
In [24]: max_temp = df['temperature'].max()
max_temp
```

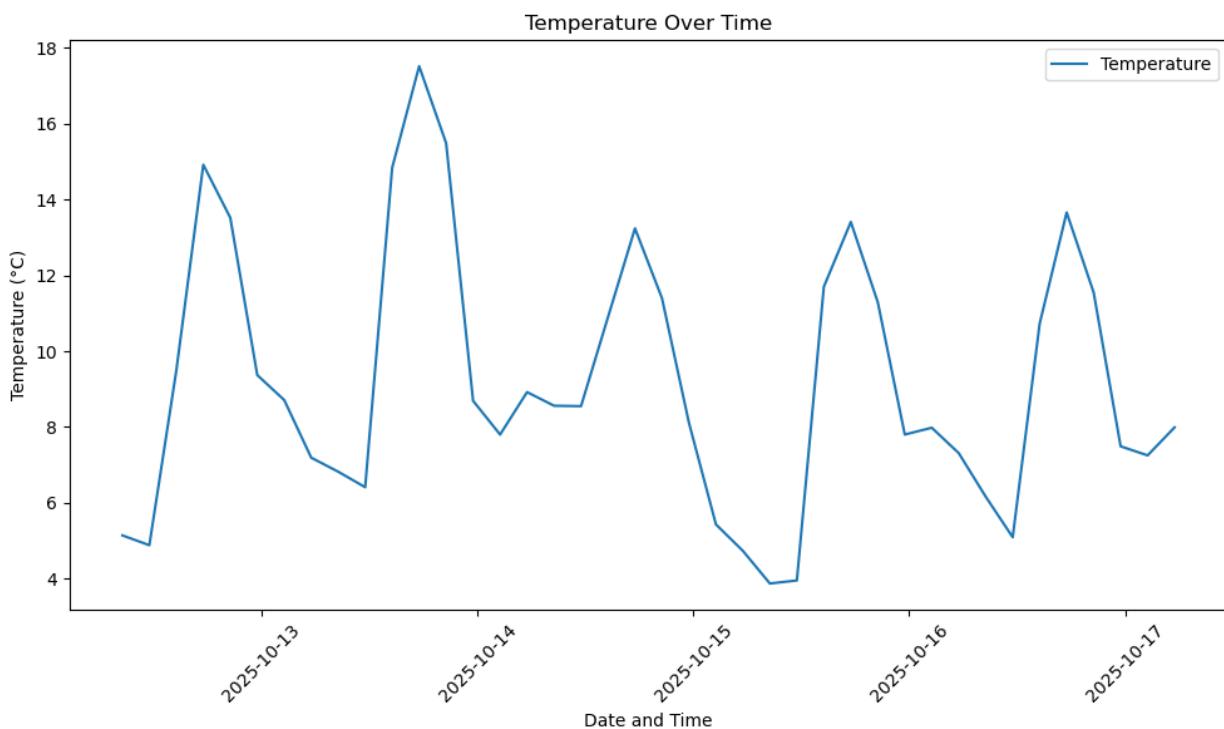
Out[24]: 17.52

```
In [26]: min_temp = df['temperature'].min()
min_temp
```

Out[26]: 3.87

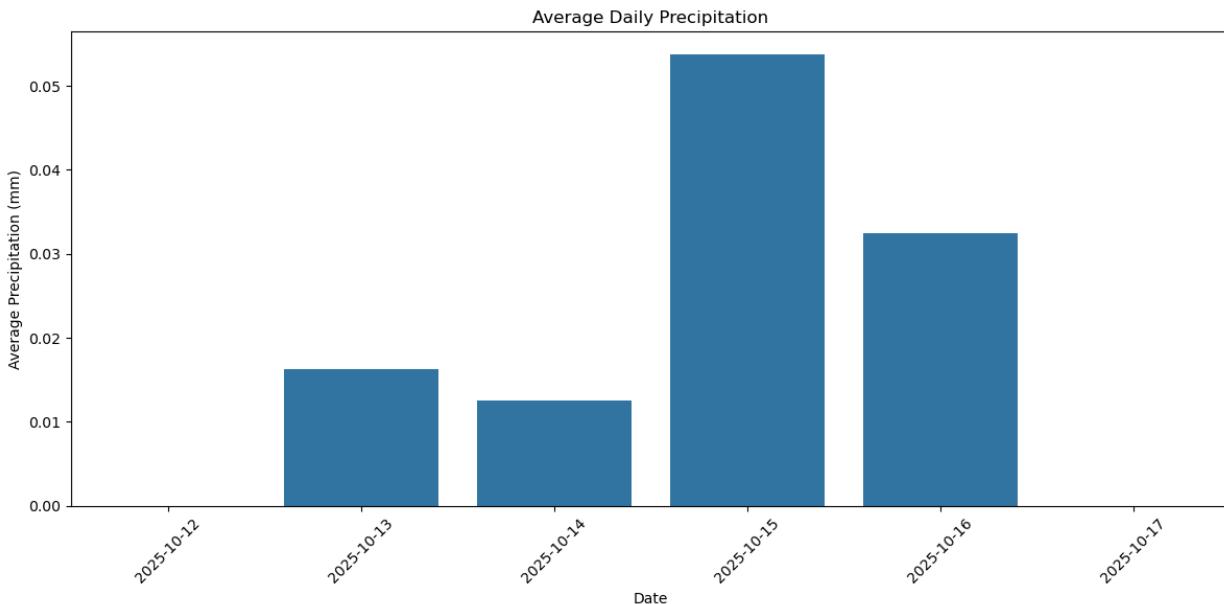
Step no. 06 - Visualize the Weather Data.

```
In [29]: # 1. Line Chart: Temperature over time
plt.figure(figsize=(10, 6))
plt.plot(df['datetime'], df['temperature'], label='Temperature')
plt.xlabel('Date and Time')
plt.ylabel('Temperature (°C)')
plt.title('Temperature Over Time')
plt.xticks(rotation=45)
plt.legend()
plt.tight_layout()
plt.show()
```



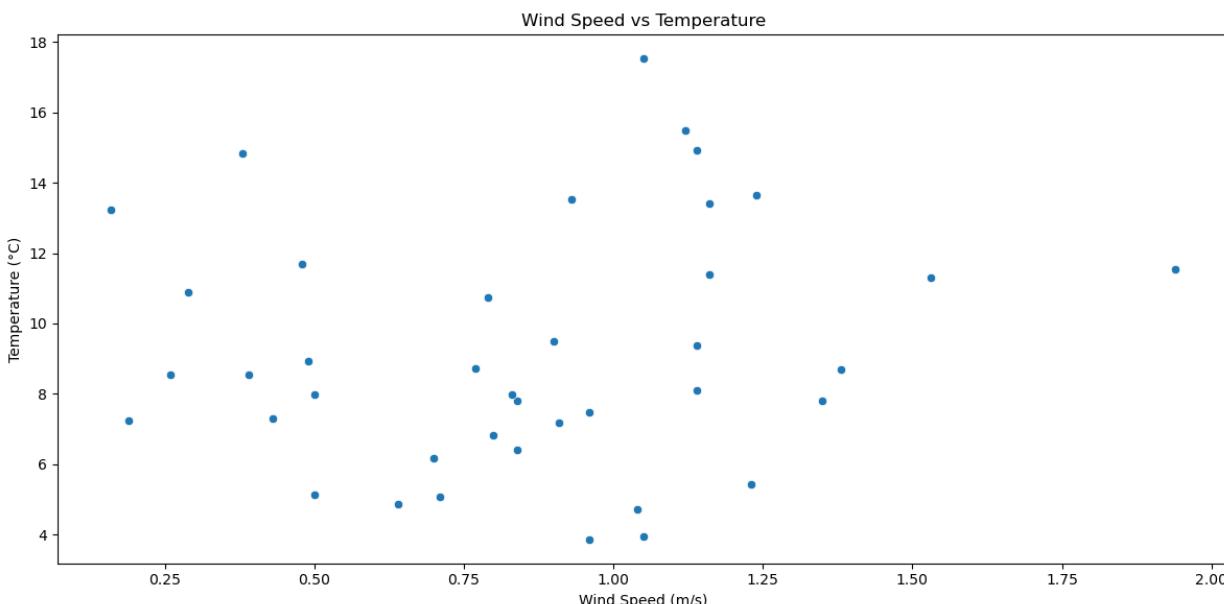
```
In [31]: # 2. Bar Plot: Daily average precipitation
df['date'] = df['datetime'].dt.date
daily_precipitation = df.groupby('date')['precipitation'].mean().reset_index()

plt.figure(figsize=(12, 6))
sns.barplot(data=daily_precipitation, x='date', y='precipitation')
plt.xlabel('Date')
plt.ylabel('Average Precipitation (mm)')
plt.title('Average Daily Precipitation')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



In [33]:

```
# 3. Scatter Plot: Wind Speed vs Temperature
plt.figure(figsize=(12, 6))
sns.scatterplot(data=df, x='wind_speed', y='temperature')
plt.xlabel('Wind Speed (m/s)')
plt.ylabel('Temperature (°C)')
plt.title('Wind Speed vs Temperature')
plt.tight_layout()
plt.show()
```



Step No. 07 - Apply Data Aggregation Techniques.

In [36]:

```
df['datetime'] = pd.to_datetime(df['datetime'])
df = df.set_index('datetime')

# Resample to daily frequency
daily_weather = df.resample('D').agg({
    'temperature': 'mean',
    'humidity': 'mean',
    'wind_speed': 'max'
})
```

```
# Display the result
print(daily_weather.head())
```

datetime	temperature	humidity	wind_speed
2025-10-12	9.553333	65.666667	1.14
2025-10-13	10.708750	61.375000	1.38
2025-10-14	9.685000	76.625000	1.16
2025-10-15	7.772500	78.125000	1.53
2025-10-16	8.746250	79.500000	1.94

In [38]: # 2. Monthly Aggregation.

```
monthly_weather = df.resample('ME').agg({'temperature': 'mean', 'humidity': 'mean', 'wind_speed': 'mean'})
monthly_weather.head()
```

Out[38]:

datetime	temperature	humidity	wind_speed
2025-10-31	9.1965	73.15	1.94

In [40]: # 3. Seasonal Aggregation.

```
def get_season(month):
    if month in [12, 1, 2]:
        return 'Winter'
    elif month in [3, 4, 5]:
        return 'Spring'
    elif month in [6, 7, 8]:
        return 'Summer'
    else:
        return 'Autumn'
df['season'] = df.index.month.map(get_season)
seasonal_weather = df.groupby('season').agg({'temperature': 'mean', 'humidity': 'mean', 'wind_speed': 'mean'})
seasonal_weather
```

Out[40]:

season	temperature	humidity	wind_speed
Autumn	9.1965	73.15	1.94

Step No. 08 - Incorporate Geographical Information.

In [43]: pip install folium

Requirement already satisfied: folium in c:\users\ruchi\anaconda3\lib\site-packages (0.20.0)
Requirement already satisfied: branca>=0.6.0 in c:\users\ruchi\anaconda3\lib\site-packages (from folium) (0.8.1)
Requirement already satisfied: jinja2>=2.9 in c:\users\ruchi\anaconda3\lib\site-packages (from folium) (3.1.4)
Requirement already satisfied: numpy in c:\users\ruchi\anaconda3\lib\site-packages (from folium) (1.26.4)
Requirement already satisfied: requests in c:\users\ruchi\anaconda3\lib\site-packages (from folium) (2.32.3)
Requirement already satisfied: xyzservices in c:\users\ruchi\anaconda3\lib\site-packages (from folium) (2022.9.0)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\ruchi\anaconda3\lib\site-packages (from jinja2>=2.9->folium) (2.1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\ruchi\anaconda3\lib\site-packages (from requests->folium) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in c:\users\ruchi\anaconda3\lib\site-packages (from requests->folium) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\ruchi\anaconda3\lib\site-packages (from requests->folium) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\ruchi\anaconda3\lib\site-packages (from requests->folium) (2025.7.14)
Note: you may need to restart the kernel to use updated packages.

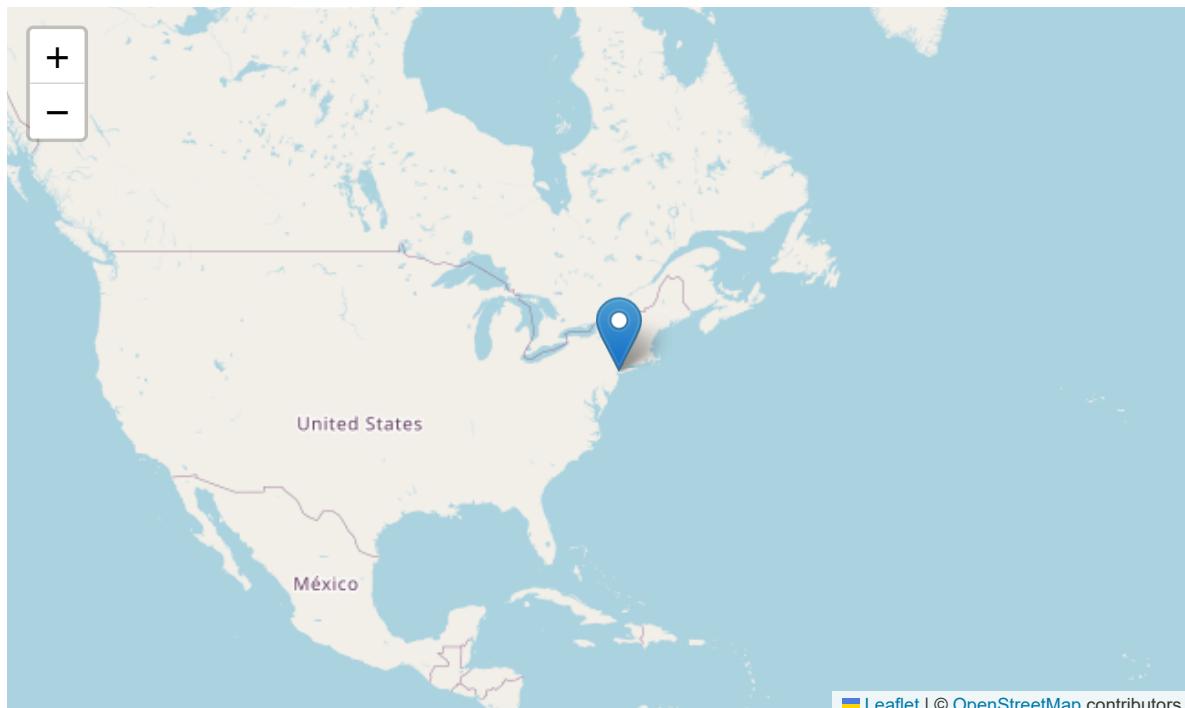
```
In [45]: # 1. Fetch Weather Data for Multiple Locations.  
# Replace 'your_api_key' with your actual API key  
api_key = 'c15c676d2f9d0dada63d7fa10c76ce01'  
locations = [  
    {'name': 'New York', 'lat': 40.7128, 'lon': -74.0060},  
    {'name': 'London', 'lat': 51.5074, 'lon': -0.1278},  
    {'name': 'Tokyo', 'lat': 35.6895, 'lon': 139.6917},  
    # Add more locations as needed  
]  
weather_data = []  
for loc in locations:  
    url = f'http://api.openweathermap.org/data/2.5/weather?lat={loc["lat"]}&lon={loc["lon"]}&a  
    response = requests.get(url)  
    data = response.json()  
    if response.status_code == 200:  
        weather_data.append({  
            'name': loc['name'],  
            'temperature': data['main']['temp'],  
            'humidity': data['main']['humidity'],  
            'wind_speed': data['wind']['speed'],  
            'latitude': loc['lat'],  
            'longitude': loc['lon']  
        })  
    else:  
        print(f"Error fetching data for {loc['name']}: {data.get('message', 'Unknown error')}")  
weather_data
```

```
Out[45]: [{'name': 'New York',
   'temperature': 16.16,
   'humidity': 82,
   'wind_speed': 5.49,
   'latitude': 40.7128,
   'longitude': -74.006},
  {'name': 'London',
   'temperature': 7.93,
   'humidity': 90,
   'wind_speed': 1.16,
   'latitude': 51.5074,
   'longitude': -0.1278},
  {'name': 'Tokyo',
   'temperature': 22.22,
   'humidity': 59,
   'wind_speed': 1.68,
   'latitude': 35.6895,
   'longitude': 139.6917}]
```

In [264...]

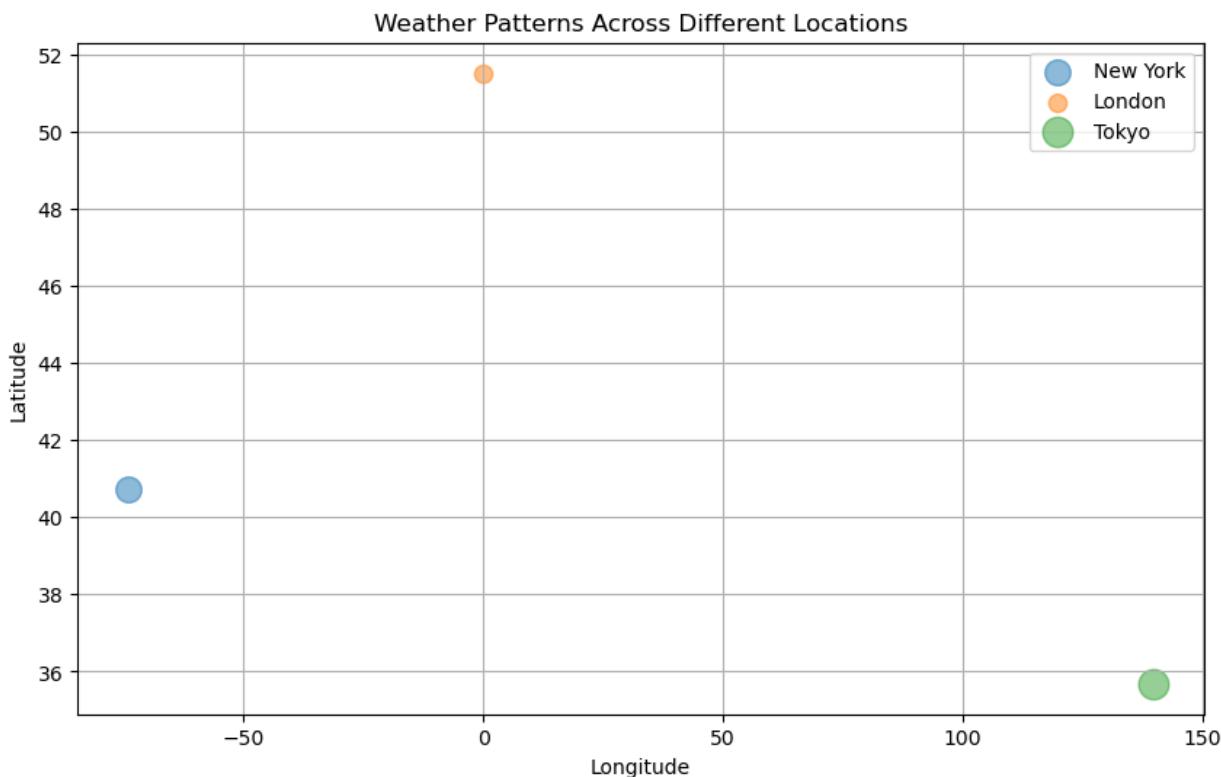
```
import folium
# 2. Create a Geospatial visualization using Folium.
# Create a map centered at a specific location (e.g., New York)
map_center = [40.7128, -74.006]
mymap = folium.Map(location=map_center, zoom_start=3)
# Add markers for each location with weather information
for data in weather_data:
    popup_text = f"<b>{data['name']}</b><br>Temperature:{data['temperature']} °C<br>Humidity: {data['humidity']}<br>Wind Speed: {data['wind_speed']}<br>Latitude: {data['latitude']}<br>Longitude: {data['longitude']}"
    folium.Marker(location=[data['latitude'], data['longitude']], popup=popup_text).add_to(mymap)
# Save the map as an HTML file
mymap.save('weather_map.html')
mymap
```

Out[264...]



```
In [47]: # 3. Visualize Weather Patterns on a Static Map using Matplotlib.
# Plot each location with a scatter plot based on temperature
plt.figure(figsize=(10, 6))
for data in weather_data:
```

```
plt.scatter(data['longitude'], data['latitude'], s=data['temperature']*10, alpha=0.5,
label=data['name'])
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.title('Weather Patterns Across Different Locations')
plt.legend()
plt.grid(True)
plt.show()
```

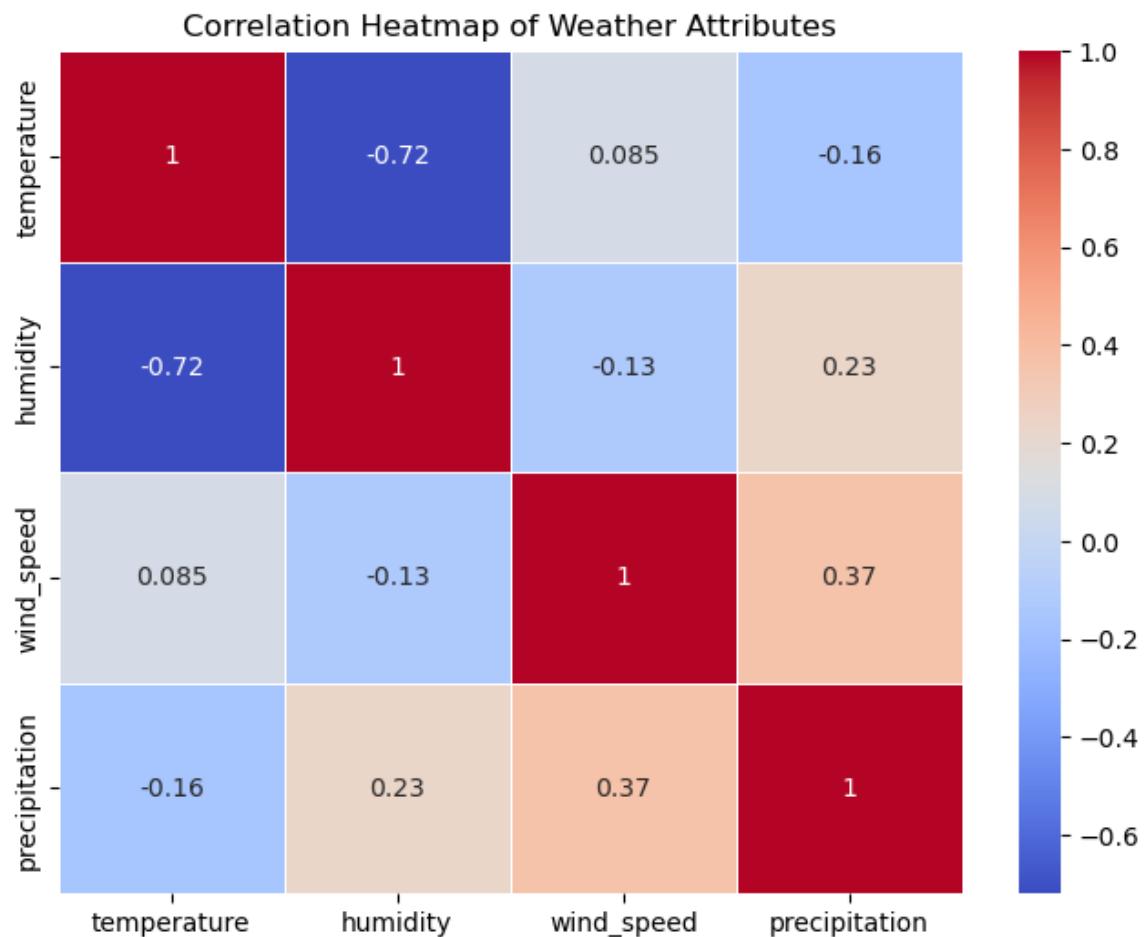


Step No. 09 - Explore and Visualize Realationships.

In [269...]

```
# Calculate correlation matrix
correlation_matrix = df[['temperature', 'humidity', 'wind_speed', 'precipitation']].corr()
print(correlation_matrix)
# Plot heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Heatmap of Weather Attributes')
plt.show()
```

	temperature	humidity	wind_speed	precipitation
temperature	1.000000	-0.719488	0.085031	-0.158885
humidity	-0.719488	1.000000	-0.126451	0.233421
wind_speed	0.085031	-0.126451	1.000000	0.366726
precipitation	-0.158885	0.233421	0.366726	1.000000



In []:

In []:

Practical No. 9

Title: Data cleaning and preparation

Name : Tavhare Ruchita Sharad

Roll No : 61

```
In [2]: import pandas as pd
import numpy as np
df = pd.read_csv("Telco-Customer-Churn.xls")
df.head()
```

Out[2]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService
0	7590-VHVEG	Female	0	Yes	No	1	No	No	No phone service
1	5575-GNVDE	Male	0	No	No	34	Yes	No	No
2	3668-QPYBK	Male	0	No	No	2	Yes	No	No
3	7795-CFOCW	Male	0	No	No	45	No	No	No phone service
4	9237-HQITU	Female	0	No	No	2	Yes	No	No

5 rows × 21 columns

```
In [4]: df.tail()
```

Out[4]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService
7038	6840-RESVB	Male	0	Yes	Yes	24	Yes	Yes	Yes
7039	2234-XADUH	Female	0	Yes	Yes	72	Yes	Yes	Yes
7040	4801-JZAZL	Female	0	Yes	Yes	11	No	No	No phone service
7041	8361-LTMKD	Male	1	Yes	No	4	Yes	Yes	Yes
7042	3186-AJIEK	Male	0	No	No	66	Yes	No	No

5 rows × 21 columns

Step No.2 - Explore the Dataset.

```
In [7]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   customerID      7043 non-null    object  
 1   gender          7043 non-null    object  
 2   SeniorCitizen   7043 non-null    int64  
 3   Partner         7043 non-null    object  
 4   Dependents     7043 non-null    object  
 5   tenure          7043 non-null    int64  
 6   PhoneService    7043 non-null    object  
 7   MultipleLines   7043 non-null    object  
 8   InternetService 7043 non-null    object  
 9   OnlineSecurity  7043 non-null    object  
 10  OnlineBackup    7043 non-null    object  
 11  DeviceProtection 7043 non-null    object  
 12  TechSupport    7043 non-null    object  
 13  StreamingTV    7043 non-null    object  
 14  StreamingMovies 7043 non-null    object  
 15  Contract        7043 non-null    object  
 16  PaperlessBilling 7043 non-null    object  
 17  PaymentMethod   7043 non-null    object  
 18  MonthlyCharges 7043 non-null    float64 
 19  TotalCharges   7043 non-null    object  
 20  Churn           7043 non-null    object  
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
```

In [9]: df.columns

```
Out[9]: Index(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
       'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling',
       'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn'],
      dtype='object')
```

In [11]: df.describe()

	SeniorCitizen	tenure	MonthlyCharges
count	7043.000000	7043.000000	7043.000000
mean	0.162147	32.371149	64.761692
std	0.368612	24.559481	30.090047
min	0.000000	0.000000	18.250000
25%	0.000000	9.000000	35.500000
50%	0.000000	29.000000	70.350000
75%	0.000000	55.000000	89.850000
max	1.000000	72.000000	118.750000

Step No.3 - Handle Missing Values.

In [14]: df.isnull().sum()

```
Out[14]: customerID      0  
gender          0  
SeniorCitizen   0  
Partner         0  
Dependents     0  
tenure          0  
PhoneService    0  
MultipleLines   0  
InternetService 0  
OnlineSecurity   0  
OnlineBackup    0  
DeviceProtection 0  
TechSupport     0  
StreamingTV    0  
StreamingMovies 0  
Contract        0  
PaperlessBilling 0  
PaymentMethod   0  
MonthlyCharges  0  
TotalCharges    0  
Churn           0  
dtype: int64
```

Step No.4 - Remove Duplicate Records.

```
In [17]: df.duplicated()
```

```
Out[17]: 0      False  
1      False  
2      False  
3      False  
4      False  
...  
7038  False  
7039  False  
7040  False  
7041  False  
7042  False  
Length: 7043, dtype: bool
```

```
In [19]: df = df.drop_duplicates()  
df.duplicated().sum()
```

```
Out[19]: 0
```

Step No.5 - Check for Inconsistent Data.

```
In [22]: def standardize_text(df, text_columns):  
    for col in text_columns:  
        df[col] = df[col].str.strip().str.lower()  
    return df  
text_columns = df.select_dtypes(include='object').columns  
text_columns
```

```
Out[22]: Index(['customerID', 'gender', 'Partner', 'Dependents', 'PhoneService',  
               'MultipleLines', 'InternetService', 'OnlineSecurity', 'OnlineBackup',  
               'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies',  
               'Contract', 'PaperlessBilling', 'PaymentMethod', 'TotalCharges',  
               'Churn'],  
               dtype='object')
```

```
In [24]: df = standardize_text(df, text_columns)
df.head()
```

```
Out[24]:   customerID  gender  SeniorCitizen  Partner  Dependents  tenure  PhoneService  MultipleLines  InternetService
0  7590-vhveg  Female          0        Yes        No         1      No  No phone service
1  5575-gnvde    Male          0       No        No        34      Yes      No
2  3668-qpybk    Male          0       No        No         2      Yes      No
3  7795-cfocw    Male          0       No        No        45      No  No phone service
4  9237-hqjtu  Female          0       No        No         2      Yes      No
```

5 rows × 21 columns

Step No.6 - Convert Columns to Correct Data Types.

```
In [27]: df.dtypes
```

```
Out[27]:   customerID      object
            gender        object
            SeniorCitizen    int64
            Partner        object
            Dependents     object
            tenure        int64
            PhoneService    object
            MultipleLines    object
            InternetService  object
            OnlineSecurity    object
            OnlineBackup     object
            DeviceProtection  object
            TechSupport      object
            StreamingTV      object
            StreamingMovies    object
            Contract        object
            PaperlessBilling  object
            PaymentMethod     object
            MonthlyCharges   float64
            TotalCharges      object
            Churn           object
            dtype: object
```

```
In [29]: df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')
df['SeniorCitizen'] = df['SeniorCitizen'].astype(bool)
df.dtypes
```

```
Out[29]: customerID          object
         gender            object
         SeniorCitizen      bool
         Partner           object
         Dependents        object
         tenure            int64
         PhoneService      object
         MultipleLines     object
         InternetService   object
         OnlineSecurity    object
         OnlineBackup       object
         DeviceProtection  object
         TechSupport        object
         StreamingTV       object
         StreamingMovies   object
         Contract          object
         PaperlessBilling  object
         PaymentMethod     object
         MonthlyCharges    float64
         TotalCharges      float64
         Churn             object
         dtype: object
```

Step No.7 - Identify and Handle Outliers.

```
In [32]: # Function to identify outliers using the IQR method
def identify_outliers_iqr(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return df[(df[column] < lower_bound) | (df[column] > upper_bound)]
# Identify outliers for each numerical column
numerical_columns = df.select_dtypes(include=[np.number]).columns
outliers = {col: identify_outliers_iqr(df, col) for col in numerical_columns}
# Display the number of outliers for each numerical column
outlier_counts = {col: len(outliers[col]) for col in outliers}
outlier_counts
```

```
Out[32]: {'tenure': 0, 'MonthlyCharges': 0, 'TotalCharges': 0}
```

```
In [34]: # Function to remove outliers using the IQR method
def remove_outliers_iqr(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]
# Remove outliers for each numerical column
for col in numerical_columns:
    df = remove_outliers_iqr(df, col)
# Display the dataframe shape after outlier removal
df.shape
```

```
Out[34]: (7032, 21)
```

Step No.8 - Perform Feature Engineering.

```
In [37]: # Create a new feature for total services count
service_columns = [
    'PhoneService', 'MultipleLines', 'InternetService', 'OnlineSecurity',
    'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies'
]
df['TotalServices'] = df[service_columns].apply(lambda x: x.eq('yes').sum(), axis=1)
# Create a new feature for the ratio of MonthlyCharges to TotalCharges
df['ChargesRatio'] = df['MonthlyCharges'] / (df['TotalCharges'] + 1) # Add 1 to avoid division by zero
# Create tenure groups
def tenure_group(tenure):
    if tenure <= 12:
        return '0-1 year'
    elif tenure <= 24:
        return '1-2 years'
    elif tenure <= 48:
        return '2-4 years'
    elif tenure <= 60:
        return '4-5 years'
    else:
        return '5+ years'
df['TenureGroup'] = df['tenure'].apply(tenure_group)
# Display the first few rows to verify the new features
df[['TotalServices', 'ChargesRatio', 'TenureGroup']].head()
```

Out[37]:

	TotalServices	ChargesRatio	TenureGroup
0	0	0.967585	0-1 year
1	0	0.030124	2-4 years
2	0	0.493358	0-1 year
3	0	0.022967	2-4 years
4	0	0.463151	0-1 year

Step No.9 - Normalize or Scale the Data.

```
In [39]: from sklearn.preprocessing import MinMaxScaler
# List of numerical columns to scale
numerical_columns = ['tenure', 'MonthlyCharges', 'TotalCharges']
# Initialize the scaler
scaler = MinMaxScaler()
# Apply the scaler to the numerical columns
df[numerical_columns] = scaler.fit_transform(df[numerical_columns])
# Display the first few rows to verify the changes
df[numerical_columns].head()
```

Out[39]:

	tenure	MonthlyCharges	TotalCharges
0	0.000000	0.115423	0.001275
1	0.464789	0.385075	0.215867
2	0.014085	0.354229	0.010310
3	0.619718	0.239303	0.210241
4	0.014085	0.521891	0.015330

```
In [40]: from sklearn.preprocessing import StandardScaler
# List of numerical columns to scale
numerical_columns = ['tenure', 'MonthlyCharges', 'TotalCharges']
# Initialize the scaler
scaler = StandardScaler()
# Apply the scaler to the numerical columns
df[numerical_columns] = scaler.fit_transform(df[numerical_columns])
# Display the first few rows to verify the changes
df[numerical_columns].head()
```

```
Out[40]:    tenure  MonthlyCharges  TotalCharges
0   -1.280248      -1.161694     -0.994194
1    0.064303      -0.260878     -0.173740
2   -1.239504      -0.363923     -0.959649
3    0.512486      -0.747850     -0.195248
4   -1.239504       0.196178     -0.940457
```

Step No.10 - Split the Dataset into Training and Testing Sets.

```
In [43]: from sklearn.model_selection import train_test_split
# Separate features (X) and target variable (y)
X = df.drop('Churn', axis=1) # Features
y = df['Churn'] # Target variable
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Display the shapes of the resulting datasets
(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
((5625, 23), (1407, 23), (5625,), (1407,))
```

```
Out[43]: ((5625, 23), (1407, 23), (5625,), (1407,))
```

Step no.11 - Export the Cleaned Dataset.

```
In [48]: df.to_csv("Cleaned_Telecom_Customer_Churn.csv", index=False)
```

```
In [ ]:
```

Practical No. 10

Title: Data Wrangling

Name : Tavhare Ruchita Sharad

Roll No : 61

```
In [45]: import pandas as pd  
import numpy as np
```

```
In [47]: df = pd.read_csv("Mumbai House Prices.csv")
```

```
In [49]: df.columns
```

```
Out[49]: Index(['bhk', 'type', 'locality', 'area', 'price', 'price_unit', 'region',  
               'status', 'age'],  
              dtype='object')
```

```
In [51]: df.columns = df.columns.str.replace(' ', '_').str.replace('(' , '').str.replace(')' , '').str.l  
df.columns
```

```
Out[51]: Index(['bhk', 'type', 'locality', 'area', 'price', 'price_unit', 'region',  
               'status', 'age'],  
              dtype='object')
```

```
In [53]: df.head()
```

```
Out[53]:
```

	bhk	type	locality	area	price	price_unit	region	status	age
0	3	Apartment	Lak And Hanware The Residency Tower	685	2.50	Cr	Andheri West	Ready to move	New
1	2	Apartment	Radheya Sai Enclave Building No 2	640	52.51	L	Naigaon East	Under Construction	New
2	2	Apartment	Romell Serene	610	1.73	Cr	Borivali West	Under Construction	New
3	2	Apartment	Soundlines Codename Urban Rainforest	876	59.98	L	Panvel	Under Construction	New
4	2	Apartment	Origin Oriana	659	94.11	L	Mira Road East	Under Construction	New

```
In [55]: df.tail()
```

	bhk	type	locality	area	price	price_unit	region	status	age
76033	3	Apartment	Parinee Liva Roca	1527	7.00	Cr	Juhu	Ready to move	Unknown
76034	5	Apartment	Parinee Liva Roca	3049	12.00	Cr	Juhu	Ready to move	Unknown
76035	4	Apartment	Lodha Seaview	3313	10.00	Cr	Napeansea Road	Ready to move	Unknown
76036	2	Apartment	Hubtown Serene	1305	4.25	Cr	Bandra East	Ready to move	Unknown
76037	5	Apartment	Sunteck Signature Island	5200	25.00	Cr	Bandra Kurla Complex	Ready to move	Unknown

In [57]: `df.isnull().sum()`

Out[57]:

bhk	0
type	0
locality	0
area	0
price	0
price_unit	0
region	0
status	0
age	0
dtype: int64	

In [59]: `additional_df = pd.read_csv("Mumbai House Prices.csv")
additional_df.columns`

Out[59]:

```
Index(['bhk', 'type', 'locality', 'area', 'price', 'price_unit', 'region',
       'status', 'age'],
      dtype='object')
```

In [61]: `additional_df.columns = additional_df.columns.str.replace(' ', '_').str.replace('(', '').str.replace(')', '')
additional_df.columns`

Out[61]:

```
Index(['bhk', 'type', 'locality', 'area', 'price', 'price_unit', 'region',
       'status', 'age'],
      dtype='object')
```

In [63]: `df_merged = pd.merge(df, additional_df, left_index=True, right_index=True, how='left')
print(df_merged)`

	bhk_x	type_x	locality_x	area_x	\	
0	3	Apartment	Lak And Hanware The Residency Tower	685		
1	2	Apartment	Radheya Sai Enclave Building No 2	640		
2	2	Apartment	Romell Serene	610		
3	2	Apartment	Soundlines Codename Urban Rainforest	876		
4	2	Apartment	Origin Oriana	659		
...		
76033	3	Apartment	Parinee Liva Roca	1527		
76034	5	Apartment	Parinee Liva Roca	3049		
76035	4	Apartment	Lodha Seaview	3313		
76036	2	Apartment	Hubtown Serene	1305		
76037	5	Apartment	Sunteck Signature Island	5200		
	price_x	price_unit_x	region_x	status_x	\	
0	2.50	Cr	Andheri West	Ready to move		
1	52.51	L	Naigaon East	Under Construction		
2	1.73	Cr	Borivali West	Under Construction		
3	59.98	L	Panvel	Under Construction		
4	94.11	L	Mira Road East	Under Construction		
...		
76033	7.00	Cr	Juhu	Ready to move		
76034	12.00	Cr	Juhu	Ready to move		
76035	10.00	Cr	Napeansea Road	Ready to move		
76036	4.25	Cr	Bandra East	Ready to move		
76037	25.00	Cr	Bandra Kurla Complex	Ready to move		
	age_x	bhk_y	type_y	locality_y	\	
0	New	3	Apartment	Lak And Hanware The Residency Tower		
1	New	2	Apartment	Radheya Sai Enclave Building No 2		
2	New	2	Apartment	Romell Serene		
3	New	2	Apartment	Soundlines Codename Urban Rainforest		
4	New	2	Apartment	Origin Oriana		
...		
76033	Unknown	3	Apartment	Parinee Liva Roca		
76034	Unknown	5	Apartment	Parinee Liva Roca		
76035	Unknown	4	Apartment	Lodha Seaview		
76036	Unknown	2	Apartment	Hubtown Serene		
76037	Unknown	5	Apartment	Sunteck Signature Island		
	area_y	price_y	price_unit_y	region_y	status_y	\
0	685	2.50	Cr	Andheri West	Ready to move	
1	640	52.51	L	Naigaon East	Under Construction	
2	610	1.73	Cr	Borivali West	Under Construction	
3	876	59.98	L	Panvel	Under Construction	
4	659	94.11	L	Mira Road East	Under Construction	
...	
76033	1527	7.00	Cr	Juhu	Ready to move	
76034	3049	12.00	Cr	Juhu	Ready to move	
76035	3313	10.00	Cr	Napeansea Road	Ready to move	
76036	1305	4.25	Cr	Bandra East	Ready to move	
76037	5200	25.00	Cr	Bandra Kurla Complex	Ready to move	
	age_y					
0	New					
1	New					
2	New					
3	New					
4	New					
...	...					
76033	Unknown					
76034	Unknown					
76035	Unknown					

```
76036 Unknown
76037 Unknown
```

[76038 rows x 18 columns]

```
In [67]: df_filtered_location = df[df['locality'] == '3 Aces The Signature Tower']
df_filtered_location
```

	bhk	type	locality	area	price	price_unit	region	status	age
37908	2	Apartment	3 Aces The Signature Tower	806	1.24		Cr Vikhroli	Under Construction	New
38720	2	Apartment	3 Aces The Signature Tower	887	1.26		Cr Vikhroli	Under Construction	New
40773	2	Apartment	3 Aces The Signature Tower	806	1.14		Cr Vikhroli	Under Construction	New
50529	2	Apartment	3 Aces The Signature Tower	815	1.22		Cr Vikhroli	Ready to move	New
72135	1	Apartment	3 Aces The Signature Tower	593	78.99		L Vikhroli	Under Construction	New
72136	2	Apartment	3 Aces The Signature Tower	700	1.07		Cr Vikhroli	Under Construction	New

```
In [69]: df_filtered_location.tail()
```

	bhk	type	locality	area	price	price_unit	region	status	age
38720	2	Apartment	3 Aces The Signature Tower	887	1.26		Cr Vikhroli	Under Construction	New
40773	2	Apartment	3 Aces The Signature Tower	806	1.14		Cr Vikhroli	Under Construction	New
50529	2	Apartment	3 Aces The Signature Tower	815	1.22		Cr Vikhroli	Ready to move	New
72135	1	Apartment	3 Aces The Signature Tower	593	78.99		L Vikhroli	Under Construction	New
72136	2	Apartment	3 Aces The Signature Tower	700	1.07		Cr Vikhroli	Under Construction	New

```
In [73]: print(df.columns)
```

```
Index(['bhk', 'type', 'locality', 'area', 'price', 'price_unit', 'region',
       'status', 'age'],
      dtype='object')
```

```
In [83]: df_filtered_property_type = df[(df['area'] == 806) & (df['price'] > 1.14)]
df_filtered_property_type.head()
```

Out[83]:

	bhk	type	locality	area	price	price_unit	region	status	age
10374	2	Apartment	K Raheja Ascencio	806	1.98	Cr	Powai	Under Construction	New
10375	2	Apartment	K Raheja Ascencio	806	1.98	Cr	Powai	Under Construction	New
12355	2	Apartment	Mahaavir Pride	806	65.00	L	Dombivali	Under Construction	New
14470	2	Apartment	Chaitanya Anand Lunkhod CHSL	806	1.80	Cr	Andheri West	Under Construction	New
14657	1	Apartment	Space Residency	806	61.51	L	Mira Road East	Ready to move	Resale

In [85]: df_filtered_property_type.tail()

Out[85]:

	bhk	type	locality	area	price	price_unit	region	status	age
55745	2	Apartment	Swami Samarth Srishti Phase 2 Wing A	806	1.25	Cr	Bhandup West	Under Construction	Unknown
64279	2	Apartment	Ornate Kallisto Phase II	806	50.00	L	Bhiwandi	Under Construction	Unknown
66182	2	Apartment	Hiranandani Highland	806	2.50	Cr	Powai	Under Construction	Unknown
69881	2	Apartment	Neelkanth Valley II	806	38.00	L	Khopoli	Ready to move	Unknown
73691	2	Apartment	Paradise Sai Pearls	806	96.00	L	Kharghar	Ready to move	Unknown

In [87]: df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 76038 entries, 0 to 76037
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype  
 ---  --          --          --    
 0   bhk         76038 non-null   int64  
 1   type        76038 non-null   object 
 2   locality    76038 non-null   object 
 3   area         76038 non-null   int64  
 4   price        76038 non-null   float64
 5   price_unit  76038 non-null   object 
 6   region       76038 non-null   object 
 7   status       76038 non-null   object 
 8   age          76038 non-null   object 
dtypes: float64(1), int64(2), object(6)
memory usage: 5.2+ MB
```

In [91]: df['locality'].unique()

```
Out[91]: array(['Lak And Hanware The Residency Tower',
   'Radheya Sai Enclave Building No 2', 'Romell Serene', ...,
   'Ahuja Prasadam Phase III', 'HBS Marineview', 'Hubtown Serene'],
  dtype=object)
```

```
In [93]: df['area'].unique()
```

```
Out[93]: array([ 685,  640,  610, ..., 1974, 1634, 3049], dtype=int64)
```

```
In [95]: df_encoded = pd.get_dummies(df, columns=['locality', 'area'])
df_encoded.head()
```

```
Out[95]:
```

	bhk	type	price	price_unit	region	status	age	locality_AJ Shri Jay	locality_Akruti	locality_Amber Heights	...	are
0	3	Apartment	2.50		Cr Andheri West	Ready to move	New	False	False	False	...	
1	2	Apartment	52.51		L Naigaon East	Under Construction	New	False	False	False	...	
2	2	Apartment	1.73		Cr Borivali West	Under Construction	New	False	False	False	...	
3	2	Apartment	59.98		L Panvel	Under Construction	New	False	False	False	...	
4	2	Apartment	94.11		L Mira Road East	Under Construction	New	False	False	False	...	

5 rows × 12120 columns

```
In [97]: from sklearn.preprocessing import LabelEncoder
```

```
In [99]: label_encoder = LabelEncoder()
```

```
In [105...]: df['locality_encoded'] = label_encoder.fit_transform(df['locality'])
df['area_encoded'] = label_encoder.fit_transform(df['area'])
df[['locality', 'locality_encoded', 'area', 'area_encoded']].head()
```

```
Out[105...]
```

	locality	locality_encoded	area	area_encoded
0	Lak And Hanware The Residency Tower	3019	685	489
1	Radheya Sai Enclave Building No 2	4965	640	444
2	Romell Serene	7031	610	414
3	Soundlines Codename Urban Rainforest	8484	876	680
4	Origin Oriana	4288	659	463

```
In [111...]: average_price_by_neighborhood = df.groupby('locality')['price'].mean().reset_index()
average_price_by_neighborhood
```

Out[111...]

	locality	price
0	AJ Shri Jay	62.000000
1	Akruti	20.000000
2	Amber Heights	28.000000
3	Arcade	62.375000
4	Asbury Park	3.266667
...
9777	Zee Swastik	2.125000
9778	Zee Tulsi	3.500000
9779	Zinal Angel Residency	17.000000
9780	Zire The Kollage	3.785455
9781	Zoeb Aayesha Palace	1.000000

9782 rows × 2 columns

In [113...]

```
average_price_by_area_type = df.groupby('area')['price'].mean().reset_index()
average_price_by_area_type
```

Out[113...]

	area	price
0	127	61.600000
1	136	31.000000
2	139	19.870000
3	149	28.702000
4	150	65.000000
...
2326	9000	30.000000
2327	10000	29.250000
2328	10893	55.550000
2329	12400	35.000000
2330	16000	46.666667

2331 rows × 2 columns

In [115...]

```
from scipy import stats
```

In [121...]

```
z_scores = stats.zscore(df['price'])
df['z_score'] = z_scores
z_scores
```

```
Out[121... 0      -0.817010
       1      0.702901
       2      -0.840412
       3      0.929931
       4      1.967215
       ...
       76033  -0.680245
       76034  -0.528285
       76035  -0.589069
       76036  -0.763824
       76037  -0.133187
Name: price, Length: 76038, dtype: float64
```

```
In [163... outliers_z_score = df[(df['z_score'] > 2.14) | (df['z_score'] < -2.14)]
```

```
In [165... outliers_z_score
```

Out[165...]

	bhk	type	locality	area	price	price_unit	region	status	age	Location
10417	2	Apartment	Kalpataru Eternia At Kalpataru Parkcity	531	99.99	L	Thane West	Under Construction	New	
10418	2	Apartment	Kalpataru Eternia At Kalpataru Parkcity	531	99.99	L	Thane West	Under Construction	New	
10452	2	Apartment	Kalpataru Eternia At Kalpataru Parkcity	531	99.99	L	Thane West	Under Construction	New	
10453	2	Apartment	Kalpataru Eternia At Kalpataru Parkcity	531	99.99	L	Thane West	Under Construction	New	
14330	2	Apartment	Kalpataru Srishti Namaah	900	99.98	L	Mira Road East	Ready to move	Resale	
...
72538	2	Apartment	JP North	981	99.90	L	Mira Road East	Ready to move	New	
72861	3	Apartment	Runwal Gardens	918	99.99	L	Dombivali	Under Construction	New	
73245	3	Apartment	Runwal Gardens	918	99.99	L	Dombivali	Under Construction	New	
73483	2	Apartment	JP North	981	99.90	L	Mira Road East	Ready to move	New	
73599	2	Apartment	Paradigm Antalya	673	99.99	L	Jogeshwari West	Under Construction	Unknown	

67 rows × 14 columns

```
In [167...]: df_cleaned = df[(df['z_score'] <= 3) & (df['z_score'] >= -3)]
```

```
In [169...]: df_cleaned = df_cleaned.drop(columns=['z_score'])
df_cleaned.head()
```

	Out[169...]	bhk	type	locality	area	price	price_unit	region	status	age	Location_encoded
0	3	Apartment	Lak And Hanware The Residency Tower	685	2.50		Cr	Andheri West	Ready to move	New	3019
1	2	Apartment	Radheya Sai Enclave Building No 2	640	52.51		L	Naigaon East	Under Construction	New	4965
2	2	Apartment	Romell Serene	610	1.73		Cr	Borivali West	Under Construction	New	7031
3	2	Apartment	Soundlines Codename Urban Rainforest	876	59.98		L	Panvel	Under Construction	New	8484
4	2	Apartment	Origin Oriana	659	94.11		L	Mira Road East	Under Construction	New	4288

Practical No. 11

Title: Analyzing Air Quality Index (AQI) Trends in a City

Name : Tavhare Ruchita Sharad

Roll No : 61

Step No.1 - Import The Dataset.

```
In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
In [3]: df = pd.read_csv("city_day.csv")
df
```

Out[3]:

	City	Date	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Ber
0	Ahmedabad	2015-01-01	NaN	NaN	0.92	18.22	17.15	NaN	0.92	27.64	133.36	
1	Ahmedabad	2015-01-02	NaN	NaN	0.97	15.69	16.46	NaN	0.97	24.55	34.06	
2	Ahmedabad	2015-01-03	NaN	NaN	17.40	19.30	29.70	NaN	17.40	29.07	30.70	
3	Ahmedabad	2015-01-04	NaN	NaN	1.70	18.48	17.97	NaN	1.70	18.59	36.08	
4	Ahmedabad	2015-01-05	NaN	NaN	22.10	21.42	37.76	NaN	22.10	39.33	39.31	
...	
29526	Visakhapatnam	2020-06-27	15.02	50.94	7.68	25.06	19.54	12.47	0.47	8.55	23.30	
29527	Visakhapatnam	2020-06-28	24.38	74.09	3.42	26.06	16.53	11.99	0.52	12.72	30.14	
29528	Visakhapatnam	2020-06-29	22.91	65.73	3.45	29.53	18.33	10.71	0.48	8.42	30.96	
29529	Visakhapatnam	2020-06-30	16.64	49.97	4.05	29.26	18.80	10.03	0.52	9.84	28.30	
29530	Visakhapatnam	2020-07-01	15.00	66.00	0.40	26.85	14.05	5.20	0.59	2.10	17.05	

29531 rows × 16 columns

```
In [5]: df.head()
```

Out[5]:

	City	Date	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	T
0	Ahmedabad	2015-01-01	NaN	NaN	0.92	18.22	17.15	NaN	0.92	27.64	133.36	0.00	
1	Ahmedabad	2015-01-02	NaN	NaN	0.97	15.69	16.46	NaN	0.97	24.55	34.06	3.68	
2	Ahmedabad	2015-01-03	NaN	NaN	17.40	19.30	29.70	NaN	17.40	29.07	30.70	6.80	
3	Ahmedabad	2015-01-04	NaN	NaN	1.70	18.48	17.97	NaN	1.70	18.59	36.08	4.43	
4	Ahmedabad	2015-01-05	NaN	NaN	22.10	21.42	37.76	NaN	22.10	39.33	39.31	7.01	

```
In [7]: df.tail()
```

	City	Date	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene
29526	Visakhapatnam	2020-06-27	15.02	50.94	7.68	25.06	19.54	12.47	0.47	8.55	23.30	2.2
29527	Visakhapatnam	2020-06-28	24.38	74.09	3.42	26.06	16.53	11.99	0.52	12.72	30.14	0.7
29528	Visakhapatnam	2020-06-29	22.91	65.73	3.45	29.53	18.33	10.71	0.48	8.42	30.96	0.0
29529	Visakhapatnam	2020-06-30	16.64	49.97	4.05	29.26	18.80	10.03	0.52	9.84	28.30	0.0
29530	Visakhapatnam	2020-07-01	15.00	66.00	0.40	26.85	14.05	5.20	0.59	2.10	17.05	Na

Step No.2 - Explore the Structure and Content Of Dataset.

In [9]: `df.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29531 entries, 0 to 29530
Data columns (total 16 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   City        29531 non-null   object  
 1   Date         29531 non-null   object  
 2   PM2.5       24933 non-null   float64 
 3   PM10        18391 non-null   float64 
 4   NO           25949 non-null   float64 
 5   NO2          25946 non-null   float64 
 6   NOx          25346 non-null   float64 
 7   NH3          19203 non-null   float64 
 8   CO           27472 non-null   float64 
 9   SO2          25677 non-null   float64 
 10  O3           25509 non-null   float64 
 11  Benzene      23908 non-null   float64 
 12  Toluene      21490 non-null   float64 
 13  Xylene       11422 non-null   float64 
 14  AQI          24850 non-null   float64 
 15  AQI_Bucket   24850 non-null   object  
dtypes: float64(13), object(3)
memory usage: 3.6+ MB
```

In [11]: `df.describe()`

	PM2.5	PM10	NO	NO2	NOx	NH3
count	24933.000000	18391.000000	25949.000000	25946.000000	25346.000000	19203.000000
mean	67.450578	118.127103	17.574730	28.560659	32.309123	23.483476
std	64.661449	90.605110	22.785846	24.474746	31.646011	25.684275
min	0.040000	0.010000	0.020000	0.010000	0.000000	0.010000
25%	28.820000	56.255000	5.630000	11.750000	12.820000	8.580000
50%	48.570000	95.680000	9.890000	21.690000	23.520000	15.850000
75%	80.590000	149.745000	19.950000	37.620000	40.127500	30.020000
max	949.990000	1000.000000	390.680000	362.210000	467.630000	352.890000

In [13]: `df.isnull().sum()`

```
Out[13]: City          0
          Date         0
          PM2.5      4598
          PM10     11140
          NO        3582
          NO2       3585
          NOx      4185
          NH3      10328
          CO        2059
          SO2      3854
          O3        4022
          Benzene    5623
          Toluene    8041
          Xylene    18109
          AQI       4681
          AQI_Bucket 4681
          dtype: int64
```

```
In [17]: df.dropna(inplace=True)
df.isnull().sum()
```

```
Out[17]: City          0
          Date         0
          PM2.5      0
          PM10       0
          NO         0
          NO2       0
          NOx       0
          NH3       0
          CO         0
          SO2       0
          O3         0
          Benzene    0
          Toluene    0
          Xylene    0
          AQI       0
          AQI_Bucket 0
          dtype: int64
```

```
In [19]: df.columns
```

```
Out[19]: Index(['City', 'Date', 'PM2.5', 'PM10', 'NO', 'NO2', 'NOx', 'NH3', 'CO', 'SO2',
       'O3', 'Benzene', 'Toluene', 'Xylene', 'AQI', 'AQI_Bucket'],
       dtype='object')
```

Step No.3 - Identify the relevant variables for visualizing AQI trends.

```
In [21]: relevant_columns = ['Date', 'AQI', 'PM2.5', 'PM10', 'CO', 'NO2', 'SO2', 'O3']
relevant_columns
```

```
Out[21]: ['Date', 'AQI', 'PM2.5', 'PM10', 'CO', 'NO2', 'SO2', 'O3']
```

```
In [23]: # Identify the available relevant columns in the dataset
available_relevant_columns = [col for col in relevant_columns if col in df.columns]
available_relevant_columns
```

```
Out[23]: ['Date', 'AQI', 'PM2.5', 'PM10', 'CO', 'NO2', 'SO2', 'O3']
```

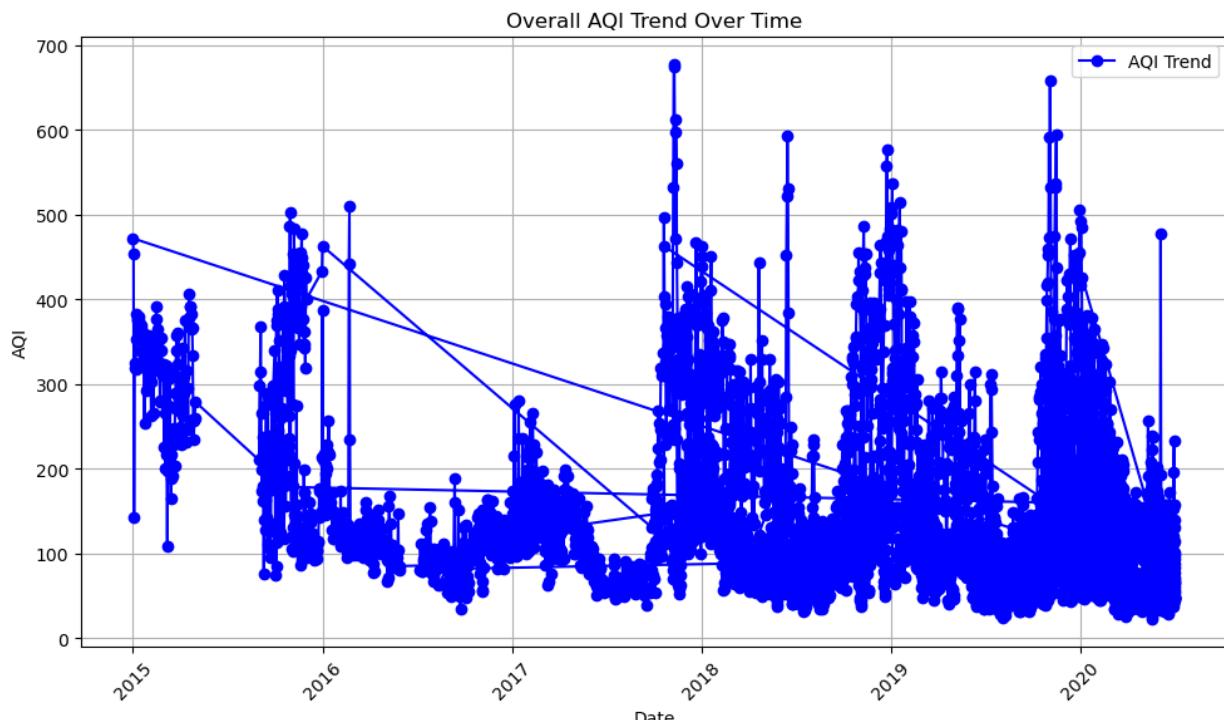
```
In [29]: if 'Date' in available_relevant_columns:
    df['Date'] = pd.to_datetime(df['Date'])
df[available_relevant_columns].head()
```

Out[29]:

	Date	AQI	PM2.5	PM10	CO	NO2	SO2	O3
2123	2017-11-25	184.0	81.40	124.50	0.12	20.50	15.24	127.09
2124	2017-11-26	197.0	78.32	129.06	0.14	26.00	26.96	117.44
2125	2017-11-27	198.0	88.76	135.32	0.11	30.85	33.59	111.81
2126	2017-11-28	188.0	64.18	104.09	0.09	28.07	19.00	138.18
2127	2017-11-29	173.0	72.47	114.84	0.16	23.20	10.55	109.74

Step No.4 - Create line plots or time series plots to visualize the overall AQI trend over time.

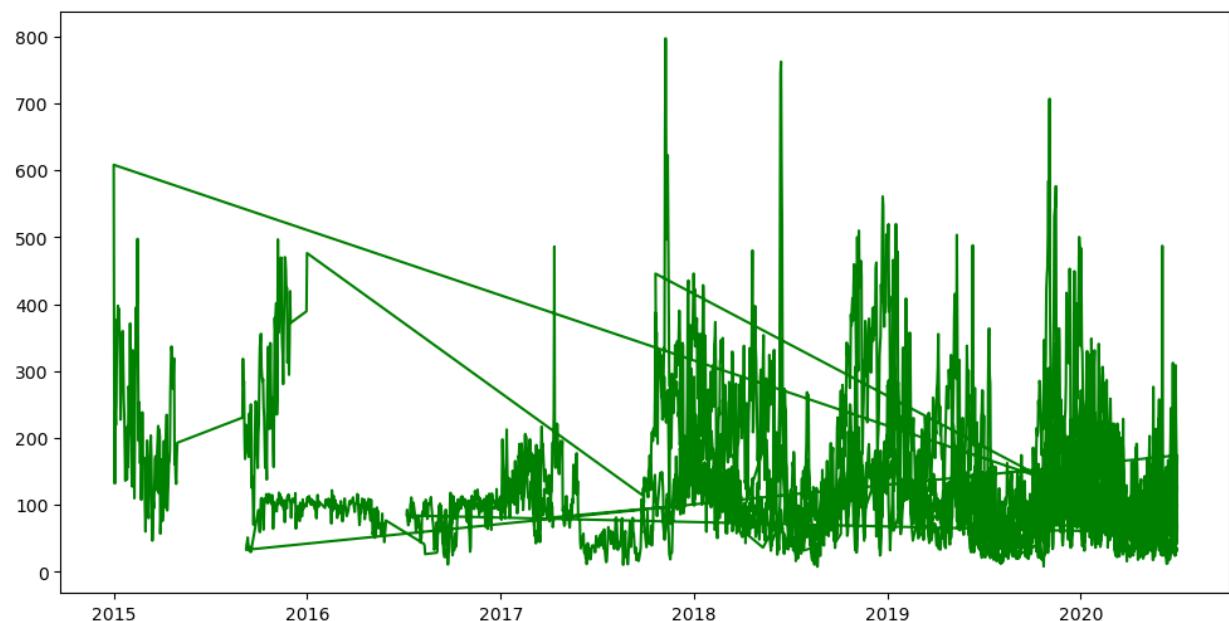
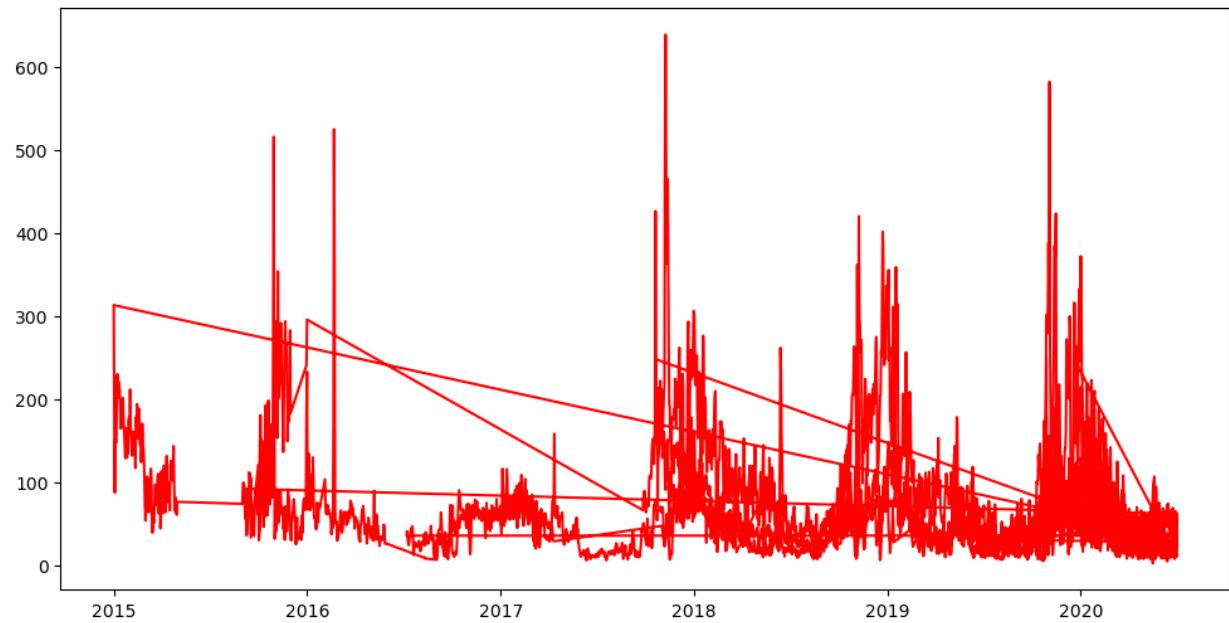
```
In [32]: plt.figure(figsize=(10, 6))
plt.plot(df['Date'], df['AQI'], marker='o', linestyle='-', color='b', label='AQI Trend')
plt.xlabel('Date')
plt.ylabel('AQI')
plt.title('Overall AQI Trend Over Time')
plt.xticks(rotation=45)
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```

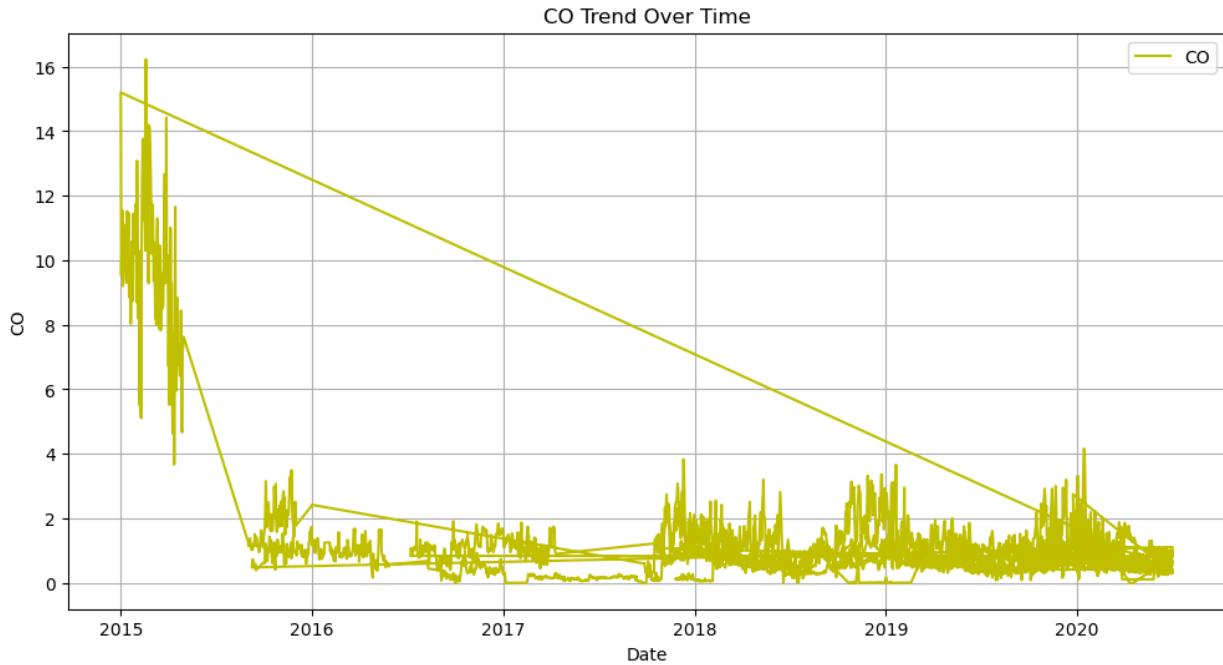


Step No.5 - Plot individual pollutant levels over time

```
In [37]: pollutants = ['PM2.5', 'PM10', 'CO']
for pollutant in pollutants:
    plt.figure(figsize=(12, 6))
    plt.plot(df['Date'], df[pollutant], label=pollutant, color='r' if pollutant == 'PM2.5' else 'y')
```

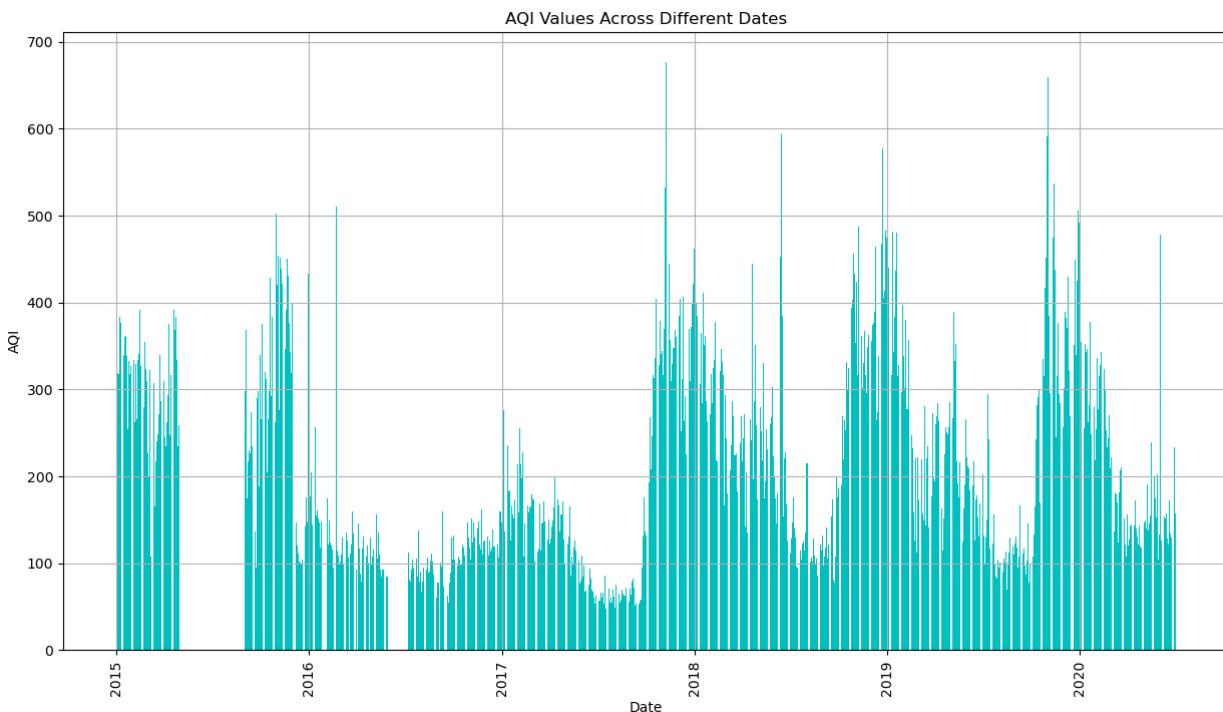
```
plt.xlabel('Date')
plt.ylabel(pollutant)
plt.title(f'{pollutant} Trend Over Time')
plt.legend()
plt.grid(True)
plt.show()
```





Step No. 6 - Use bar plots or stacked bar plots to compare the AQI values across different dates or time periods.

```
In [40]: # Plot bar plot for AQI values across different dates
plt.figure(figsize=(15, 8))
plt.bar(df['Date'], df['AQI'], color='c')
plt.xlabel('Date')
plt.ylabel('AQI')
plt.title('AQI Values Across Different Dates')
plt.xticks(rotation=90)
plt.grid(True)
plt.show()
```



```
In [43]: # Plot stacked bar plot for AQI values with different pollutants
```

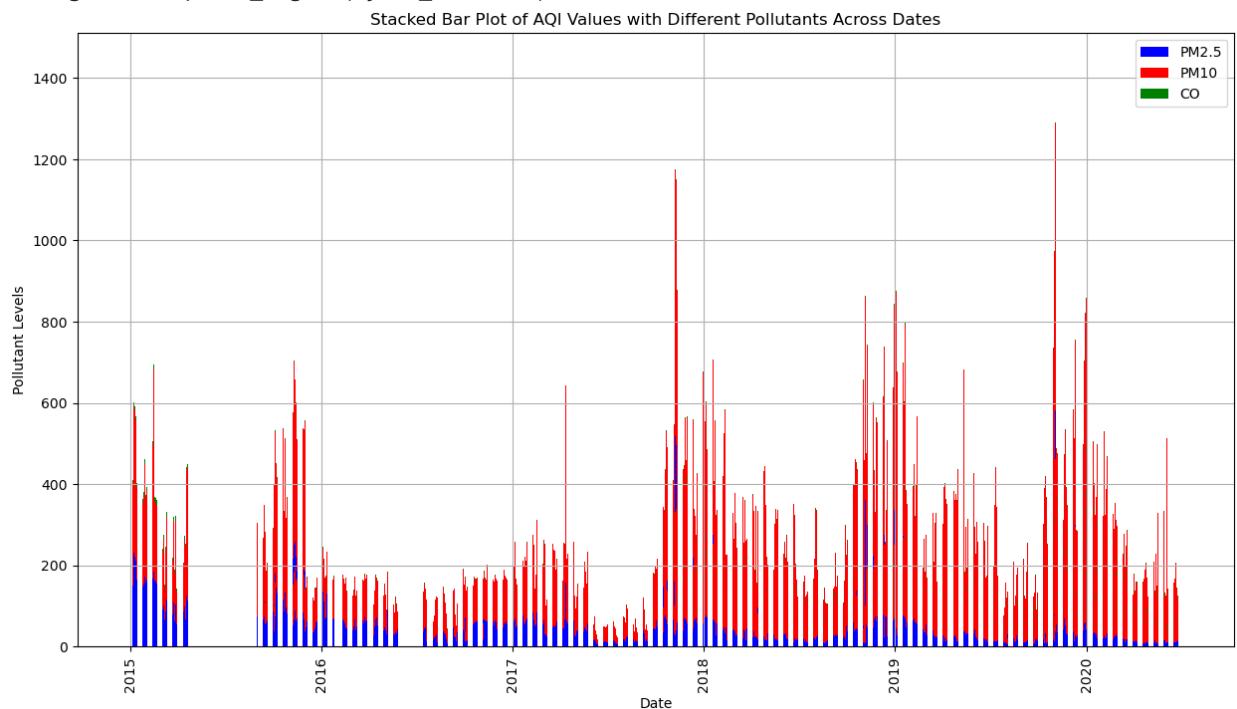
```

plt.figure(figsize=(15, 8))
bar_width = 0.5
plt.bar(df['Date'], df['PM2.5'], label='PM2.5', color='b', width=bar_width)
plt.bar(df['Date'], df['PM10'], bottom=df['PM2.5'], label='PM10', color='r', width=bar_width)
plt.bar(df['Date'], df['CO'], bottom=df['PM2.5'] + df['PM10'], label='CO', color='g', width=bar_width)
plt.xlabel('Date')
plt.ylabel('Pollutant Levels')
plt.title('Stacked Bar Plot of AQI Values with Different Pollutants Across Dates')
plt.xticks(rotation=90)
plt.legend()
plt.grid(True)
plt.show()

```

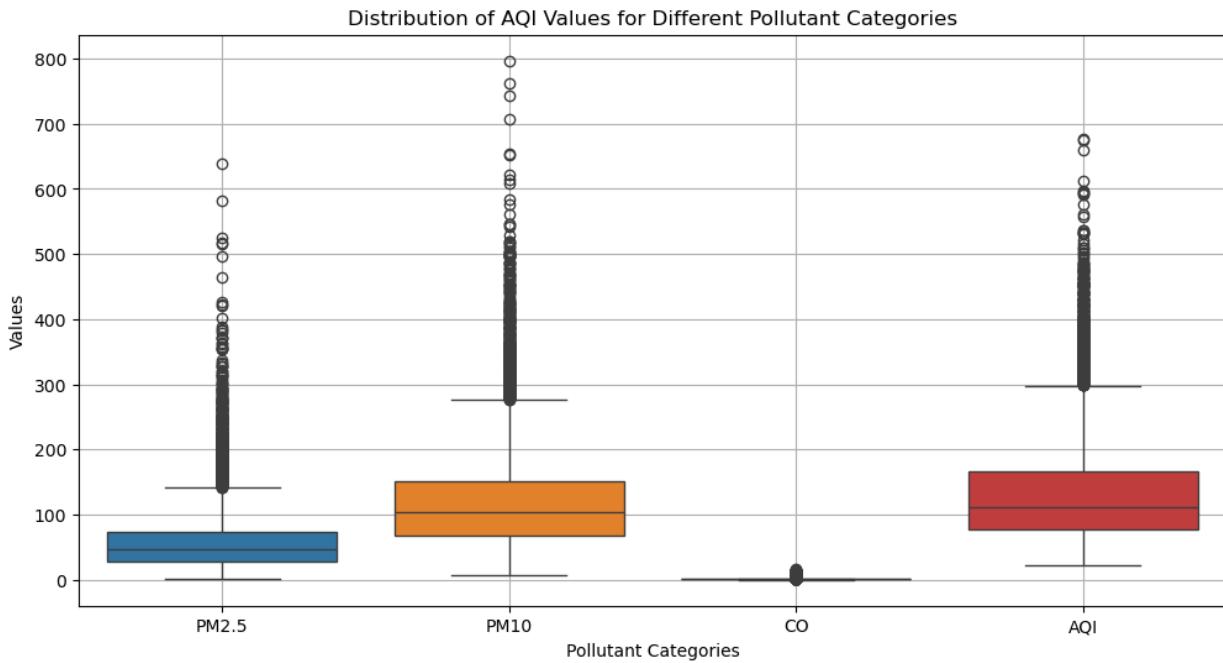
C:\Users\ruchi\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:170: UserWarning: Creating legend with loc="best" can be slow with large amounts of data.

```
fig.canvas.print_figure(bytes_io, **kw)
```

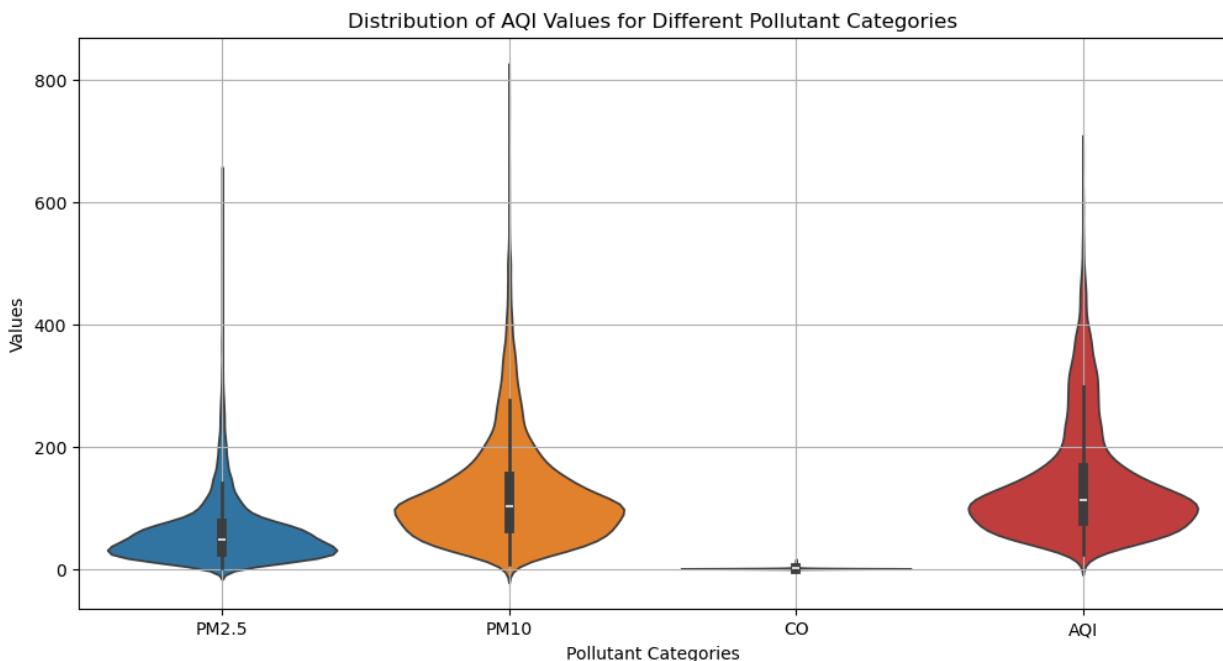


Step No. 7 - Create box plots or violin plots to analyze the distribution of AQI values for different pollutant categories.

```
In [46]: # Create box plot for AQI values by pollutant categories
plt.figure(figsize=(12, 6))
sns.boxplot(data=df[['PM2.5', 'PM10', 'CO', 'AQI']])
plt.xlabel('Pollutant Categories')
plt.ylabel('Values')
plt.title('Distribution of AQI Values for Different Pollutant Categories')
plt.grid(True)
plt.show()
```



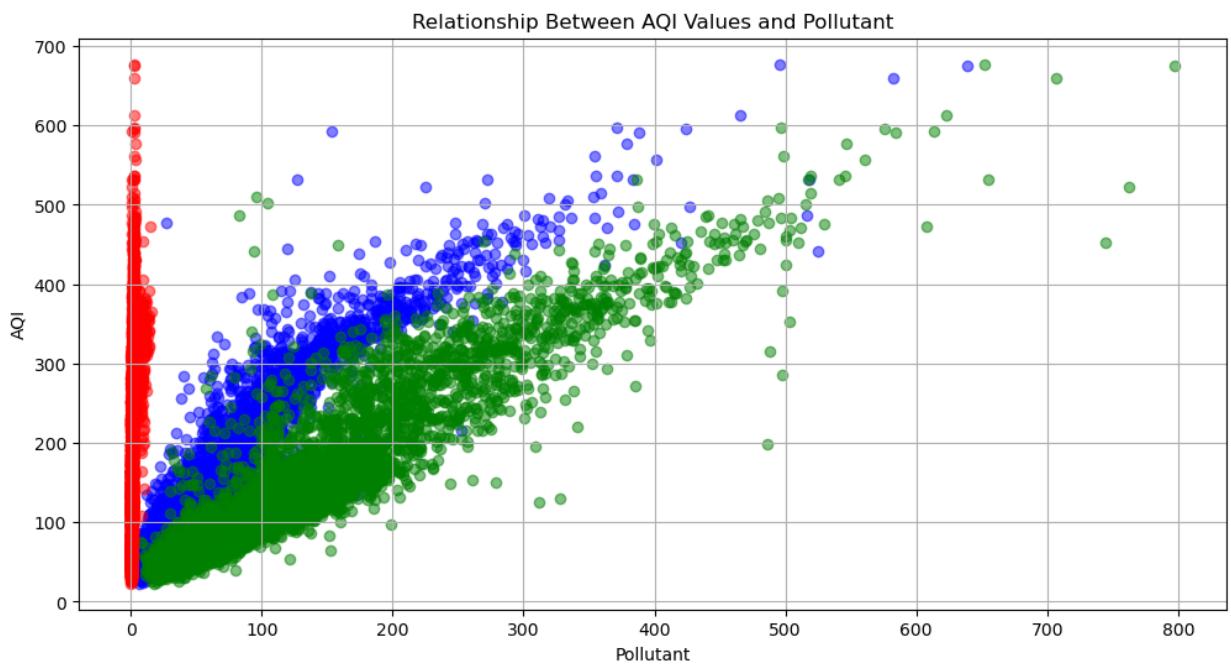
```
In [48]: # Create violin plot for AQI values by pollutant categories
plt.figure(figsize=(12, 6))
sns.violinplot(data=df[['PM2.5', 'PM10', 'CO', 'AQI']])
plt.xlabel('Pollutant Categories')
plt.ylabel('Values')
plt.title('Distribution of AQI Values for Different Pollutant Categories')
plt.grid(True)
plt.show()
```



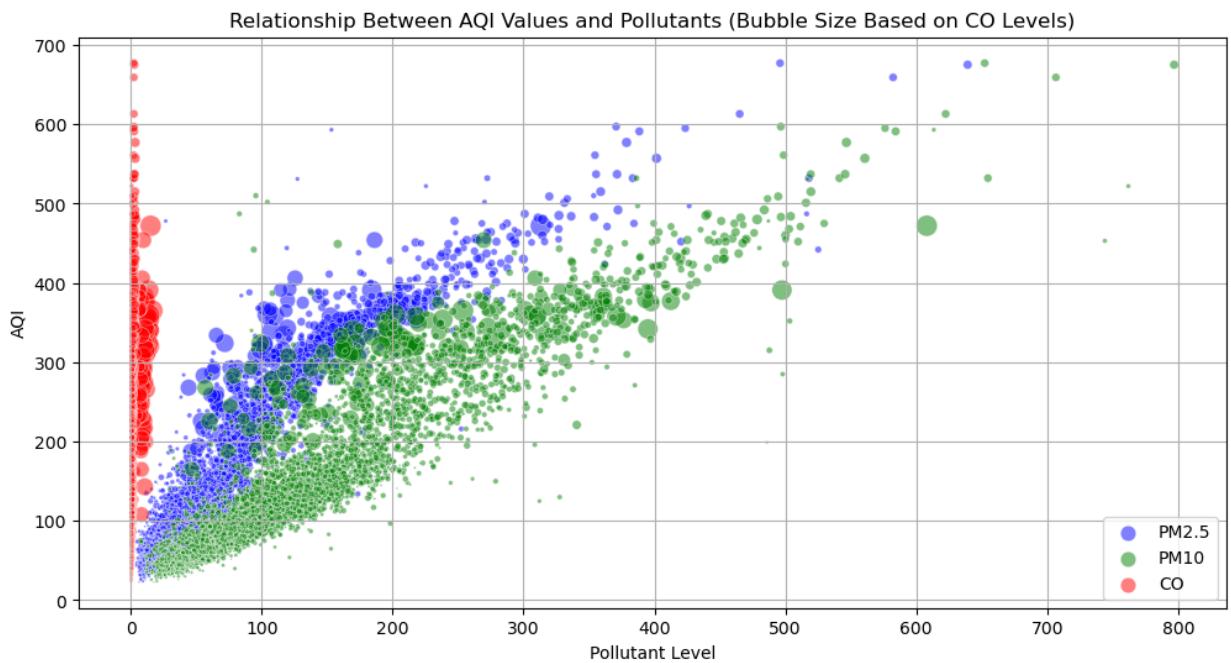
Step No. 8 - Use scatter plots or bubble charts to explore the relationship between AQI values and pollutant levels.

```
In [51]: # Scatter plot for AQI values vs. Pollutants
plt.figure(figsize=(12, 6))
plt.scatter(df['PM2.5'], df['AQI'], alpha=0.5, color='b')
plt.scatter(df['PM10'], df['AQI'], alpha=0.5, color='g')
```

```
plt.scatter(df['CO'], df['AQI'], alpha=0.5, color='r')
plt.xlabel('Pollutant')
plt.ylabel('AQI')
plt.title('Relationship Between AQI Values and Pollutant')
plt.grid(True)
plt.show()
```



```
In [65]: plt.figure(figsize=(12, 6))
plt.scatter(df['PM2.5'], df['AQI'], s=df['CO']*10, alpha=0.5, color='b', edgecolors='w', linewidth=1)
plt.scatter(df['PM10'], df['AQI'], s=df['CO']*10, alpha=0.5, color='g', edgecolors='w', linewidth=1)
plt.scatter(df['CO'], df['AQI'], s=df['CO']*10, alpha=0.5, color='r', edgecolors='w', linewidth=1)
plt.xlabel('Pollutant Level')
plt.ylabel('AQI')
plt.title('Relationship Between AQI Values and Pollutants (Bubble Size Based on CO Levels)')
plt.legend(loc='best')
plt.grid(True)
plt.show()
```



```
In [ ]:
```