DMV P1

October 9, 2025

[ ]:
```
Name: Thorave Avishkar Shrikrushna
Roll No: 65
```

1 Title : Analyzing Sales Data from Multiple File Formats

[9]:
```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import json
```

[4]:
```
csv = pd.read_csv("./datasets/sales_data_sample.csv", encoding="cp1252")
```

[7]:
```
ed = pd.read_excel("./datasets/Sample-Sales-Data.xlsx")
```

[10]:
```
with open("./datasets/customers.json", "r") as json_file:
    json_data = json.load(json_file)
```

[11]:
```
csv.tail()
```

[11]:
```
      ORDERNUMBER  QUANTITYORDERED  PRICEEACH  ORDERLINENUMBER    SALES  \
2818        10350               20     100.00               15  2244.40
2819        10373               29     100.00                1  3978.51
2820        10386               43     100.00                4  5417.57
2821        10397               34      62.24                1  2116.16
2822        10414               47      65.52                9  3079.44

           ORDERDATE    STATUS  QTR_ID  MONTH_ID  YEAR_ID  ...  \
2818  12/2/2004 0:00   Shipped       4        12     2004  ...
2819  1/31/2005 0:00   Shipped       1         1     2005  ...
2820   3/1/2005 0:00  Resolved       1         3     2005  ...
2821  3/28/2005 0:00   Shipped       1         3     2005  ...
2822   5/6/2005 0:00   On Hold       2         5     2005  ...

              ADDRESSLINE1  ADDRESSLINE2    CITY STATE POSTALCODE  COUNTRY  \
2818     C/ Moralzarzal, 86           NaN  Madrid   NaN      28034    Spain
2819            Torikatu 38           NaN    Oulu   NaN      90110  Finland
2820     C/ Moralzarzal, 86           NaN  Madrid   NaN      28034    Spain
```

```
2821   1 rue Alsace-Lorraine           NaN  Toulouse    NaN       31000    France
2822      8616 Spinnaker Dr.           NaN    Boston     MA       51003       USA

     TERRITORY CONTACTLASTNAME CONTACTFIRSTNAME DEALSIZE
2818      EMEA          Freyre            Diego    Small
2819      EMEA       Koskitalo           Pirkko   Medium
2820      EMEA          Freyre            Diego   Medium
2821      EMEA          Roulet          Annette    Small
2822       NaN         Yoshido             Juri   Medium

[5 rows x 25 columns]
```

[12]:
```
csv.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2823 entries, 0 to 2822
Data columns (total 25 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   ORDERNUMBER       2823 non-null   int64
 1   QUANTITYORDERED   2823 non-null   int64
 2   PRICEEACH         2823 non-null   float64
 3   ORDERLINENUMBER   2823 non-null   int64
 4   SALES             2823 non-null   float64
 5   ORDERDATE         2823 non-null   object
 6   STATUS            2823 non-null   object
 7   QTR_ID            2823 non-null   int64
 8   MONTH_ID          2823 non-null   int64
 9   YEAR_ID           2823 non-null   int64
 10  PRODUCTLINE       2823 non-null   object
 11  MSRP              2823 non-null   int64
 12  PRODUCTCODE       2823 non-null   object
 13  CUSTOMERNAME      2823 non-null   object
 14  PHONE             2823 non-null   object
 15  ADDRESSLINE1      2823 non-null   object
 16  ADDRESSLINE2      302 non-null    object
 17  CITY              2823 non-null   object
 18  STATE             1337 non-null   object
 19  POSTALCODE        2747 non-null   object
 20  COUNTRY           2823 non-null   object
 21  TERRITORY         1749 non-null   object
 22  CONTACTLASTNAME   2823 non-null   object
 23  CONTACTFIRSTNAME  2823 non-null   object
 24  DEALSIZE          2823 non-null   object
dtypes: float64(2), int64(7), object(16)
memory usage: 551.5+ KB
```

[13]:
```
csv.describe()
```

[13]:

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	\
count	2823.000000	2823.000000	2823.000000	2823.000000	
mean	10258.725115	35.092809	83.658544	6.466171	
std	92.085478	9.741443	20.174277	4.225841	
min	10100.000000	6.000000	26.880000	1.000000	
25%	10180.000000	27.000000	68.860000	3.000000	
50%	10262.000000	35.000000	95.700000	6.000000	
75%	10333.500000	43.000000	100.000000	9.000000	
max	10425.000000	97.000000	100.000000	18.000000	

	SALES	QTR_ID	MONTH_ID	YEAR_ID	MSRP
count	2823.000000	2823.000000	2823.000000	2823.00000	2823.000000
mean	3553.889072	2.717676	7.092455	2003.81509	100.715551
std	1841.865106	1.203878	3.656633	0.69967	40.187912
min	482.130000	1.000000	1.000000	2003.00000	33.000000
25%	2203.430000	2.000000	4.000000	2003.00000	68.000000
50%	3184.800000	3.000000	8.000000	2004.00000	99.000000
75%	4508.000000	4.000000	11.000000	2004.00000	124.000000
max	14082.800000	4.000000	12.000000	2005.00000	214.000000

[14]:

```
csv.dropna()
```

[14]:

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	\
10	10223	37	100.00	1	3965.66	
21	10361	20	72.55	13	1451.00	
40	10270	21	100.00	9	4905.39	
47	10347	30	100.00	1	3944.70	
51	10391	24	100.00	4	2416.56	
...	...	...	...	...	...	
2667	10120	43	76.00	14	3268.00	
2673	10223	26	67.20	15	1747.20	
2685	10361	44	100.00	10	5001.92	
2764	10361	35	100.00	11	4277.35	
2791	10361	23	95.20	12	2189.60	

	ORDERDATE	STATUS	QTR_ID	MONTH_ID	YEAR_ID	...	\
10	2/20/2004 0:00	Shipped	1	2	2004	...	
21	12/17/2004 0:00	Shipped	4	12	2004	...	
40	7/19/2004 0:00	Shipped	3	7	2004	...	
47	11/29/2004 0:00	Shipped	4	11	2004	...	
51	3/9/2005 0:00	Shipped	1	3	2005	...	
...	...	...	...	...	...	...	
2667	4/29/2003 0:00	Shipped	2	4	2003	...	
2673	2/20/2004 0:00	Shipped	1	2	2004	...	
2685	12/17/2004 0:00	Shipped	4	12	2004	...	
2764	12/17/2004 0:00	Shipped	4	12	2004	...	
2791	12/17/2004 0:00	Shipped	4	12	2004	...	

```
                                ADDRESSLINE1  ADDRESSLINE2          CITY  \
10                         636 St Kilda Road       Level 3     Melbourne
21    Monitor Money Building, 815 Pacific Hwy       Level 6     Chatswood
40    Monitor Money Building, 815 Pacific Hwy       Level 6     Chatswood
47                         636 St Kilda Road       Level 3     Melbourne
51                         201 Miller Street      Level 15  North Sydney
...                                      ...           ...           ...
2667                       636 St Kilda Road       Level 3     Melbourne
2673                       636 St Kilda Road       Level 3     Melbourne
2685  Monitor Money Building, 815 Pacific Hwy       Level 6     Chatswood
2764  Monitor Money Building, 815 Pacific Hwy       Level 6     Chatswood
2791  Monitor Money Building, 815 Pacific Hwy       Level 6     Chatswood

         STATE POSTALCODE    COUNTRY TERRITORY CONTACTLASTNAME  \
10    Victoria       3004  Australia      APAC        Ferguson
21         NSW       2067  Australia      APAC          Huxley
40         NSW       2067  Australia      APAC          Huxley
47    Victoria       3004  Australia      APAC        Ferguson
51         NSW       2060  Australia      APAC          O'Hara
...        ...        ...        ...       ...             ...
2667  Victoria       3004  Australia      APAC        Ferguson
2673  Victoria       3004  Australia      APAC        Ferguson
2685       NSW       2067  Australia      APAC          Huxley
2764       NSW       2067  Australia      APAC          Huxley
2791       NSW       2067  Australia      APAC          Huxley

     CONTACTFIRSTNAME DEALSIZE
10              Peter   Medium
21             Adrian    Small
40             Adrian   Medium
47              Peter   Medium
51               Anna    Small
...               ...      ...
2667            Peter   Medium
2673            Peter    Small
2685           Adrian   Medium
2764           Adrian   Medium
2791           Adrian    Small

[147 rows x 25 columns]
```

[15]:
```
csv.drop_duplicates()
```

[15]:
```
      ORDERNUMBER  QUANTITYORDERED  PRICEEACH  ORDERLINENUMBER    SALES  \
0           10107               30      95.70                2  2871.00
1           10121               34      81.35                5  2765.90
```

```
2       10134              41      94.74              2  3884.34
3       10145              45      83.26              6  3746.70
4       10159              49     100.00             14  5205.27
...       ...             ...        ...            ...      ...
2818    10350              20     100.00             15  2244.40
2819    10373              29     100.00              1  3978.51
2820    10386              43     100.00              4  5417.57
2821    10397              34      62.24              1  2116.16
2822    10414              47      65.52              9  3079.44

            ORDERDATE    STATUS  QTR_ID  MONTH_ID  YEAR_ID  ...  \
0      2/24/2003 0:00   Shipped       1         2     2003  ...
1       5/7/2003 0:00   Shipped       2         5     2003  ...
2       7/1/2003 0:00   Shipped       3         7     2003  ...
3      8/25/2003 0:00   Shipped       3         8     2003  ...
4     10/10/2003 0:00   Shipped       4        10     2003  ...
...               ...       ...     ...       ...      ...  ...
2818   12/2/2004 0:00   Shipped       4        12     2004  ...
2819   1/31/2005 0:00   Shipped       1         1     2005  ...
2820    3/1/2005 0:00  Resolved       1         3     2005  ...
2821   3/28/2005 0:00   Shipped       1         3     2005  ...
2822    5/6/2005 0:00   On Hold       2         5     2005  ...

                       ADDRESSLINE1  ADDRESSLINE2           CITY STATE  \
0           897 Long Airport Avenue           NaN            NYC    NY
1                59 rue de l'Abbaye           NaN          Reims   NaN
2     27 rue du Colonel Pierre Avia           NaN          Paris   NaN
3               78934 Hillside Dr.            NaN       Pasadena    CA
4                   7734 Strong St.           NaN  San Francisco    CA
...                             ...           ...            ...   ...
2818             C/ Moralzarzal, 86           NaN         Madrid   NaN
2819                    Torikatu 38           NaN           Oulu   NaN
2820             C/ Moralzarzal, 86           NaN         Madrid   NaN
2821          1 rue Alsace-Lorraine           NaN       Toulouse   NaN
2822             8616 Spinnaker Dr.           NaN         Boston    MA

     POSTALCODE  COUNTRY TERRITORY CONTACTLASTNAME CONTACTFIRSTNAME DEALSIZE
0         10022      USA       NaN              Yu             Kwai    Small
1         51100   France      EMEA         Henriot             Paul    Small
2         75508   France      EMEA        Da Cunha           Daniel   Medium
3         90003      USA       NaN           Young            Julie   Medium
4           NaN      USA       NaN           Brown            Julie   Medium
...         ...      ...       ...             ...              ...      ...
2818      28034    Spain      EMEA          Freyre            Diego    Small
2819      90110  Finland      EMEA       Koskitalo           Pirkko   Medium
2820      28034    Spain      EMEA          Freyre            Diego   Medium
2821      31000   France      EMEA          Roulet          Annette    Small
```

```
2822      51003      USA        NaN         Yoshido               Juri    Medium

[2823 rows x 25 columns]
```

[16]:
```
ed.head()
```

[16]:
```
   Postcode  Sales_Rep_ID Sales_Rep_Name  Year         Value
0      2121           456           Jane  2011  84219.497311
1      2092           789         Ashish  2012  28322.192268
2      2128           456           Jane  2013  81878.997241
3      2073           123           John  2011  44491.142121
4      2134           789         Ashish  2012  71837.720959
```

[17]:
```
ed.tail()
```

[17]:
```
     Postcode  Sales_Rep_ID Sales_Rep_Name  Year         Value
385      2164           123           John  2012  88884.535217
386      2193           456           Jane  2013  79440.290813
387      2031           123           John  2011  65643.689454
388      2130           456           Jane  2012  66247.874869
389      2116           456           Jane  2013   3195.699054
```

[18]:
```
ed.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 390 entries, 0 to 389
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Postcode        390 non-null    int64
 1   Sales_Rep_ID    390 non-null    int64
 2   Sales_Rep_Name  390 non-null    object
 3   Year            390 non-null    int64
 4   Value           390 non-null    float64
dtypes: float64(1), int64(3), object(1)
memory usage: 15.4+ KB
```

[19]:
```
ed.describe()
```

[19]:
```
          Postcode  Sales_Rep_ID         Year         Value
count   390.000000    390.000000   390.000000    390.000000
mean   2098.430769    456.000000  2012.000000  49229.388305
std      58.652206    272.242614     0.817545  28251.271309
min    2000.000000    123.000000  2011.000000    106.360599
25%    2044.000000    123.000000  2011.000000  26101.507357
50%    2097.500000    456.000000  2012.000000  47447.363750
75%    2142.000000    789.000000  2013.000000  72277.800608
max    2206.000000    789.000000  2013.000000  99878.489209
```

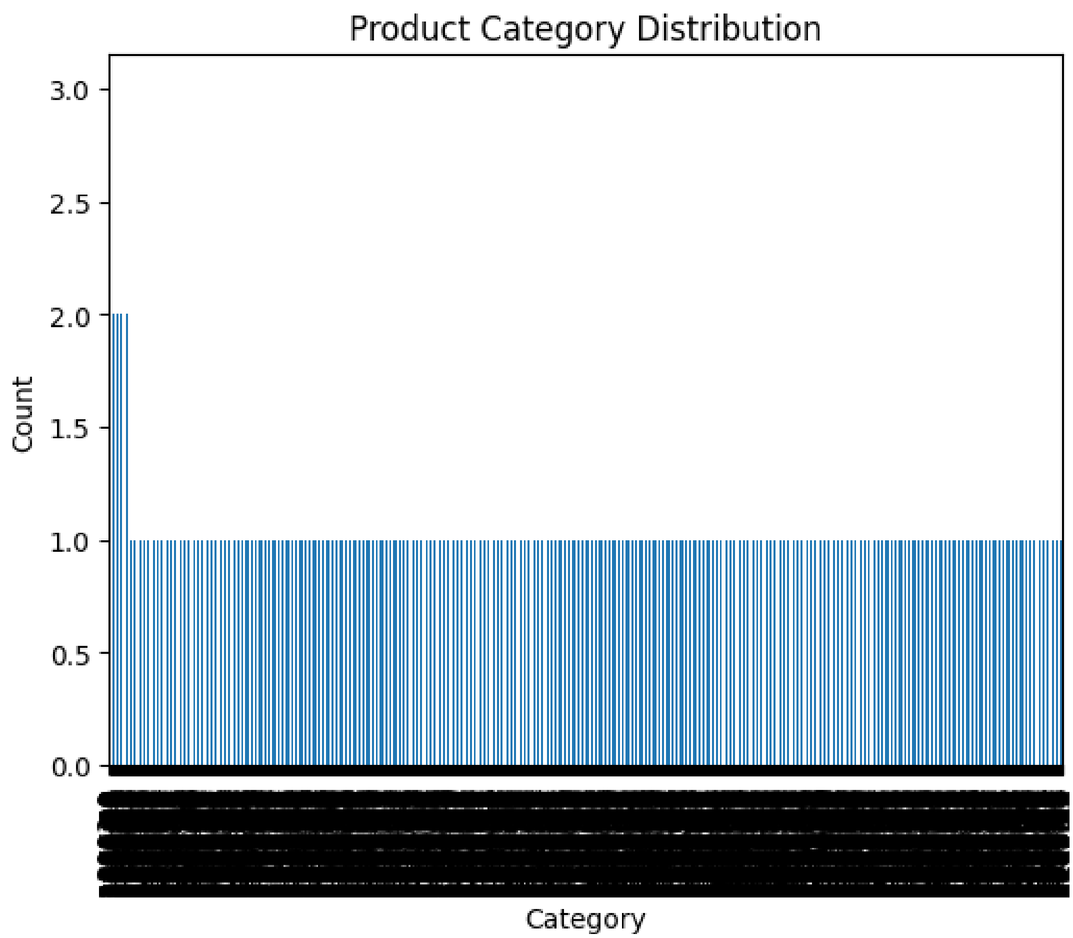
[20]:
```
unified_data = pd.concat([csv, ed], ignore_index=True)
```

[21]:
```
total_sales = unified_data['SALES'].sum()
print("Total Sales:", total_sales)
```

```
Total Sales: 10032628.85
```

[22]:
```
category_sales = unified_data.groupby('ORDERNUMBER')['SALES'].mean()
```

[23]:
```
category_counts = unified_data['SALES'].value_counts()
category_counts.plot(kind='bar')
plt.title('Product Category Distribution')
plt.xlabel('Category')
plt.ylabel('Count')
plt.show()
```



[ ]:
```
```

[ ]:
```
Name: Thorave Avishkar Shrikrushna
Roll No: 65
```

2 Title : Analyzing Weather Data from OpenWeatherMap API

[2]:
```python
import requests
import pandas as pd
import datetime
```

[3]:
```python
# Set your OpenWeatherMap API key
api_key = 'fb365aa6104829b44455572365ff3b4e'
```

[4]:
```python
# Set the location for which you want to retrieve weather data
lat = 18.184135
lon = 74.610764
```

[5]:
```python
# https://openweathermap.org/api/one-call-3
# how       How to use api call
# Construct the API URL
api_url = f"http://api.openweathermap.org/data/2.5/forecast?
↪lat={lat}&lon={lon}&appid={api_key}"
```

[8]:
```python
# Send a GET request to the API
response = requests.get(api_url)
weather_data = response.json()
weather_data.keys()
len(weather_data['list'])
weather_data['list'][0]['weather'][0]['description']
```

[8]: 'scattered clouds'

[11]:
```python
# Getting the data from dictionary and taking into one variable
# Extract relevant weather attributes using list comprehension
temperatures = [item['main']['temp'] for item in weather_data['list']]

# It will extract all values (40) and putting into one variable
timestamps = [pd.to_datetime(item['dt'], unit='s') for item in␣
↪weather_data['list']]
temperature = [item['main']['temp'] for item in weather_data['list']]
humidity = [item['main']['humidity'] for item in weather_data['list']]
wind_speed = [item['wind']['speed'] for item in weather_data['list']]
weather_description = [item['weather'][0]['description'] for item in␣
↪weather_data['list']]
```

[21]:
```python
# Create a pandas DataFrame with the extracted weather data
weather_df = pd.DataFrame({'Timestamp': timestamps,
                           'Temperature': temperatures,
```

```
                              'humidity': humidity,
                              'wind_speed':wind_speed,
                              'weather_description': weather_description})
```

[22]:
```
# Set the Timestamp column as the DataFrame's index
weather_df.set_index('Timestamp', inplace=True)
max_temp = weather_df['Temperature'].max()
print(f"Maximum Temperature - {max_temp}")
min_temp = weather_df['Temperature'].min()
print(f"Minimum Temperature - {min_temp}")
```

```
Maximum Temperature - 305.27
Minimum Temperature - 292.37
```

[23]:
```
# Clean and preprocess the data # Handling missing values
weather_df.fillna(0, inplace=True) # Replace missing values with 0 or␣
 ↪appropriate value
```

[24]:
```
# Handling inconsistent format (if applicable)
weather_df['Temperature'] = weather_df['Temperature'].apply(lambda x: x - 273.15␣
 ↪if isinstance(x, float)else x)
```

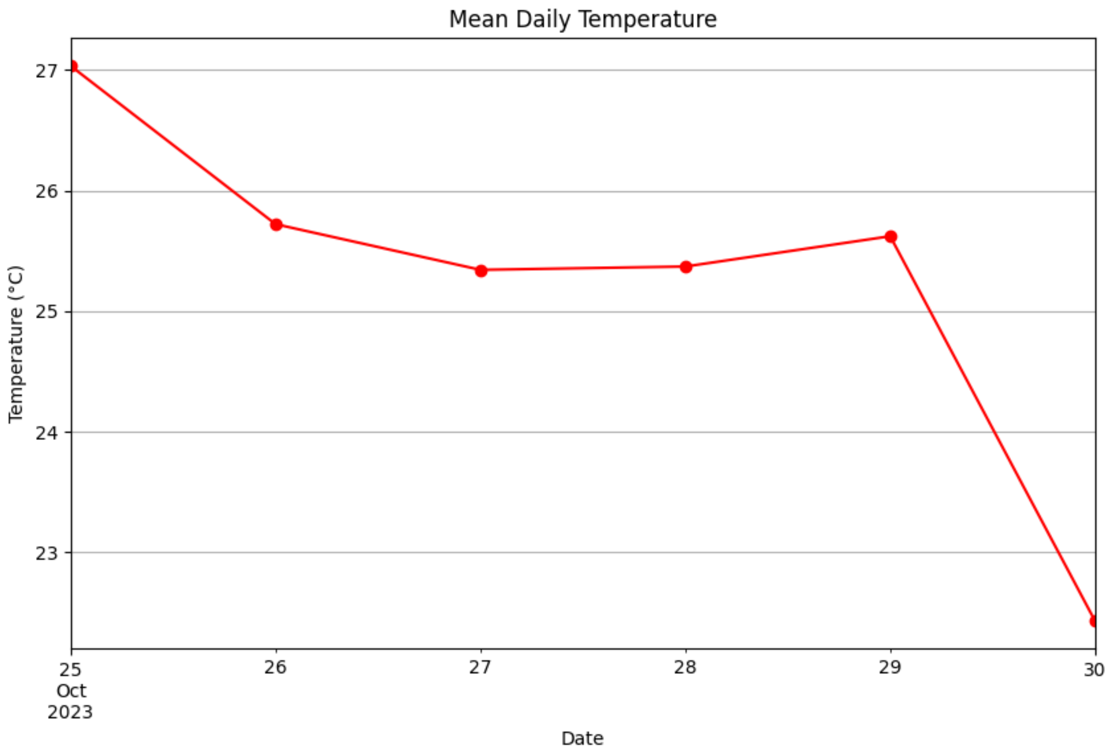
[25]:
```
# Convert temperature from Kelvin to Celsius
# Print the cleaned and preprocessed data print(weather_df)
print(weather_df)
```

```
                     Temperature  humidity  wind_speed weather_description
Timestamp
2023-10-25 06:00:00        29.99        30        3.15    scattered clouds
2023-10-25 09:00:00        30.67        28        3.55    scattered clouds
2023-10-25 12:00:00        30.23        27        5.39    scattered clouds
2023-10-25 15:00:00        26.19        31        4.05           clear sky
2023-10-25 18:00:00        23.68        40        3.66           clear sky
2023-10-25 21:00:00        21.44        49        1.62          few clouds
2023-10-26 00:00:00        20.01        55        0.29          few clouds
2023-10-26 03:00:00        24.58        40        1.43    scattered clouds
2023-10-26 06:00:00        30.17        23        4.54    scattered clouds
2023-10-26 09:00:00        32.12        18        5.11           clear sky
2023-10-26 12:00:00        29.53        23        5.13          few clouds
2023-10-26 15:00:00        25.40        28        3.91       broken clouds
2023-10-26 18:00:00        23.00        35        3.30     overcast clouds
2023-10-26 21:00:00        20.96        43        2.51       broken clouds
2023-10-27 00:00:00        19.22        49        1.40       broken clouds
2023-10-27 03:00:00        23.84        37        1.19    scattered clouds
2023-10-27 06:00:00        29.78        24        4.07    scattered clouds
2023-10-27 09:00:00        31.47        20        3.52          few clouds
2023-10-27 12:00:00        29.73        24        4.14          few clouds
2023-10-27 15:00:00        25.00        30        4.00    scattered clouds
```

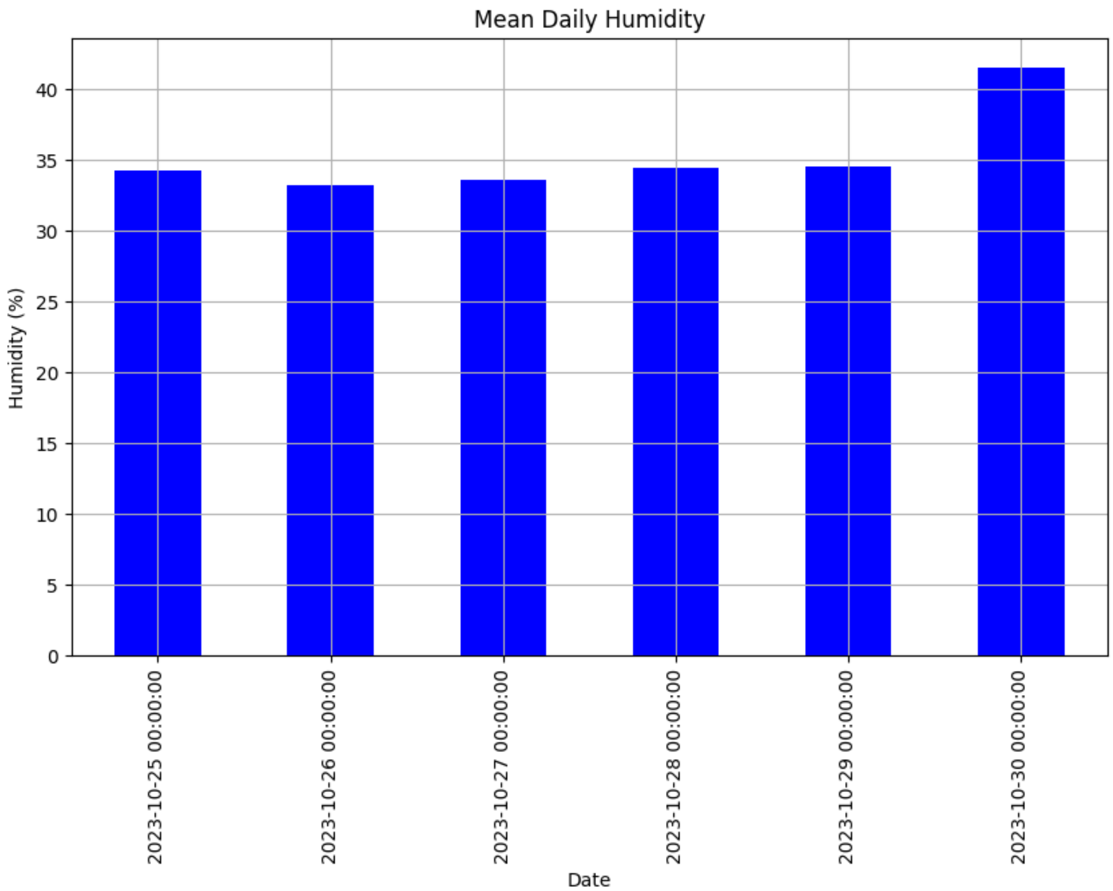
```
2023-10-27 18:00:00         22.82         38         3.37       broken clouds
2023-10-27 21:00:00         20.88         46         2.51    scattered clouds
2023-10-28 00:00:00         19.34         51         1.55    scattered clouds
2023-10-28 03:00:00         23.97         39         1.71           clear sky
2023-10-28 06:00:00         29.53         26         3.38           clear sky
2023-10-28 09:00:00         31.25         21         2.25           clear sky
2023-10-28 12:00:00         29.82         27         1.25           clear sky
2023-10-28 15:00:00         25.50         29         3.19           clear sky
2023-10-28 18:00:00         22.93         37         3.46    scattered clouds
2023-10-28 21:00:00         20.62         45         0.47       broken clouds
2023-10-29 00:00:00         19.50         48         1.13       broken clouds
2023-10-29 03:00:00         24.43         36         1.03    scattered clouds
2023-10-29 06:00:00         29.71         27         3.59    scattered clouds
2023-10-29 09:00:00         31.48         22         1.53          few clouds
2023-10-29 12:00:00         30.15         29         1.03          few clouds
2023-10-29 15:00:00         25.65         32         1.04           clear sky
2023-10-29 18:00:00         23.04         38         2.08           clear sky
2023-10-29 21:00:00         21.01         44         0.45           clear sky
2023-10-30 00:00:00         20.03         47         1.56           clear sky
2023-10-30 03:00:00         24.83         36         1.70           clear sky
```

[26]:
```
import matplotlib.pyplot as plt
daily_mean_temp = weather_df['Temperature'].resample('D').mean()
daily_mean_humidity = weather_df['humidity'].resample('D').mean()
daily_mean_wind_speed = weather_df['wind_speed'].resample('D').mean()
```

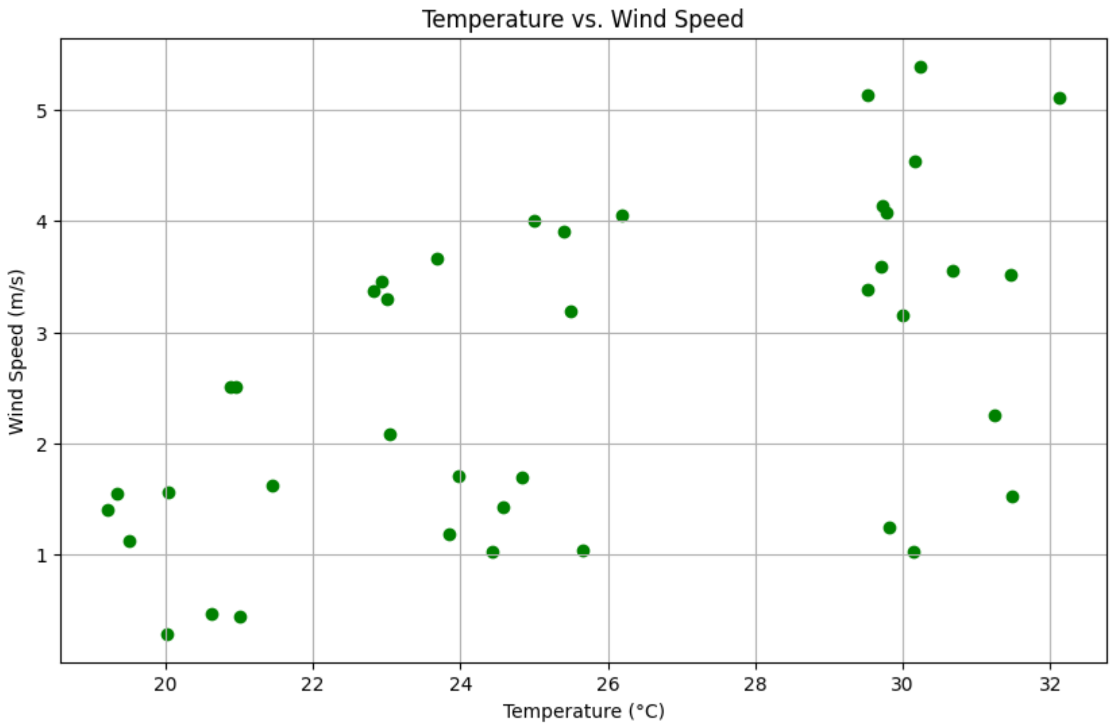
[27]:
```
# Plot the mean daily temperature over time (Line plot)
plt.figure(figsize=(10, 6))
daily_mean_temp.plot(color='red', linestyle='-', marker='o')
plt.title('Mean Daily Temperature')
plt.xlabel('Date')
plt.ylabel('Temperature (°C)')
plt.grid(True)
plt.show()
```



[28]:
```
# Plot the mean daily humidity over time (Bar plot)
plt.figure(figsize=(10, 6))
daily_mean_humidity.plot(kind='bar', color='blue')
plt.title('Mean Daily Humidity')
plt.xlabel('Date')
plt.ylabel('Humidity (%)')
plt.grid(True)
plt.show()
```

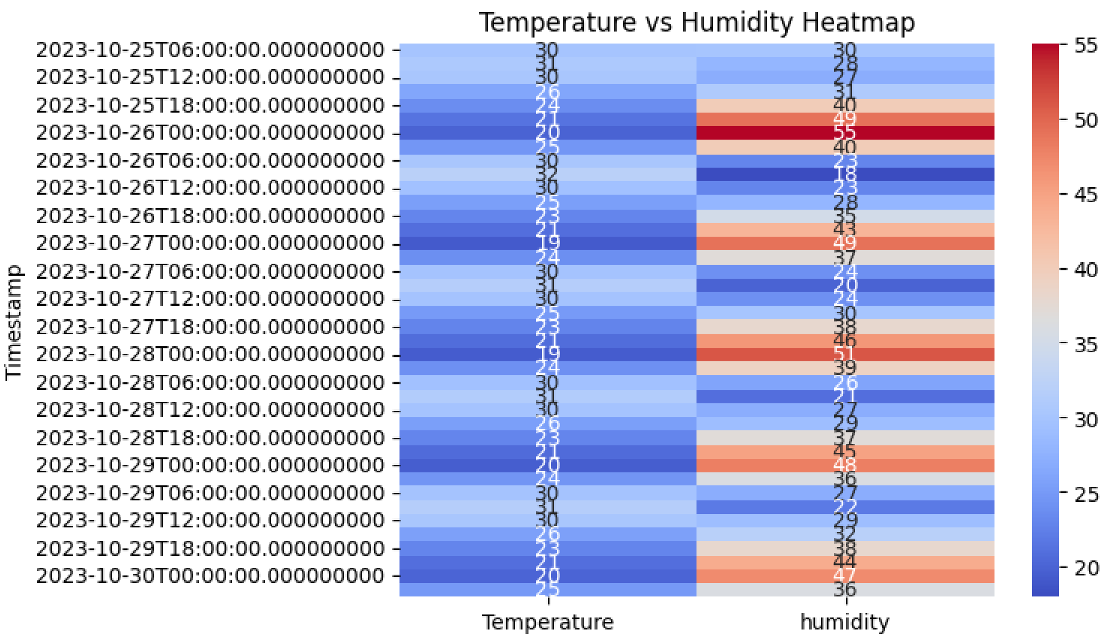


[29]:
```
# Plot the relationship between temperature and wind speed (Scatter plot)
plt.figure(figsize=(10, 6))
plt.scatter(weather_df['Temperature'], weather_df['wind_speed'], color='green')
plt.title('Temperature vs. Wind Speed')
plt.xlabel('Temperature (°C)')
plt.ylabel('Wind Speed (m/s)')
plt.grid(True)
plt.show()
```

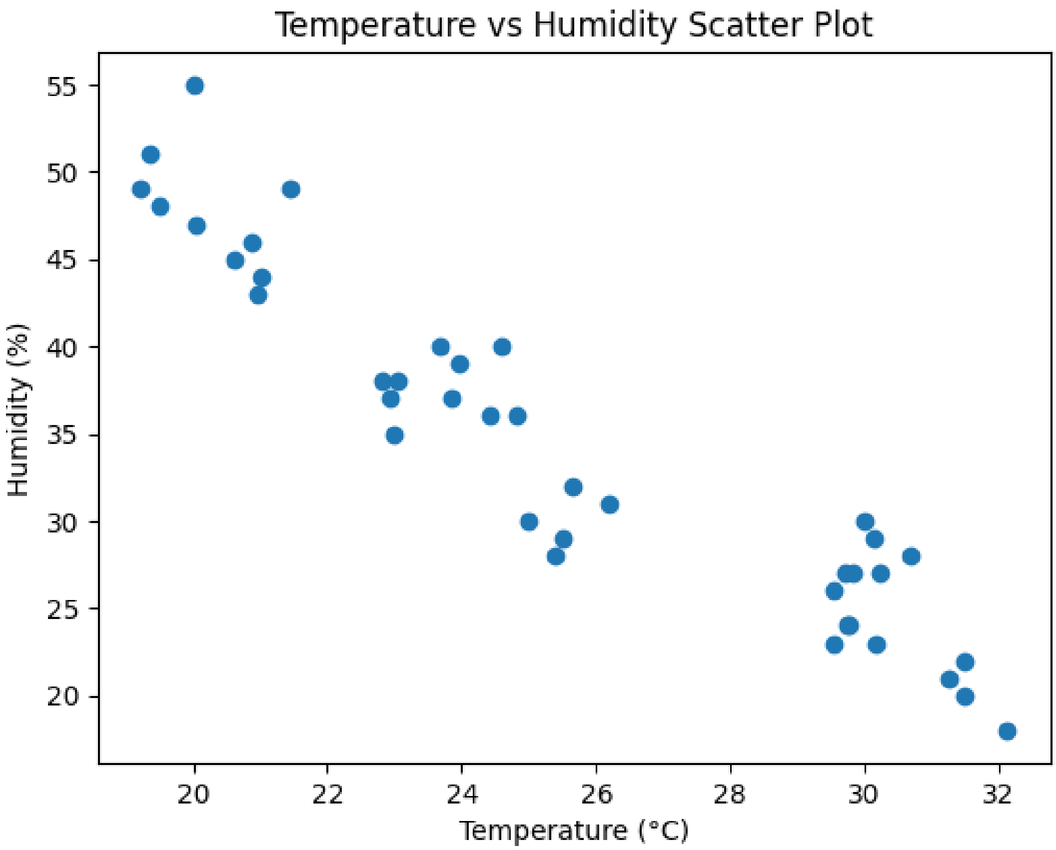


[30]:

```
# Heatmap
import seaborn as sns
heatmap_data = weather_df[['Temperature', 'humidity']]
sns.heatmap(heatmap_data, annot=True, cmap='coolwarm')
plt.title('Temperature vs Humidity Heatmap')
plt.show()
```



[31]:
```python
# Create a scatter plot to visualize the relationship between temperature and
↪humidity
plt.scatter(weather_df['Temperature'], weather_df['humidity'])
plt.xlabel('Temperature (°C)')
plt.ylabel('Humidity (%)')
plt.title('Temperature vs Humidity Scatter Plot')
plt.show()
```


Temperature vs Humidity Scatter Plot
Humidity (%)
Temperature (°C)
55
50
45
40
35
30
25
20
20
22
24
26
28
30
32

```
[ ]:
```

[ ]:
```
Name: Thorave Avishkar Shrikrushna
Roll No: 65
```

3 Title : Analyzing Customer Churn in a Telecommunications Company

[29]:
```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn import metrics
import seaborn as sns
import matplotlib.pyplot as plt
```

[8]:
```python
data = pd.read_csv("./datasets/Telcom_Customer_Churn.csv")
print(data.index)
```

```
RangeIndex(start=0, stop=7043, step=1)
```

[9]:
```python
print(data)
```

```
      customerID  gender  SeniorCitizen Partner Dependents  tenure  \
0     7590-VHVEG  Female              0     Yes         No       1
1     5575-GNVDE    Male              0      No         No      34
2     3668-QPYBK    Male              0      No         No       2
3     7795-CFOCW    Male              0      No         No      45
4     9237-HQITU  Female              0      No         No       2
...          ...     ...            ...     ...        ...     ...
7038  6840-RESVB    Male              0     Yes        Yes      24
7039  2234-XADUH  Female              0     Yes        Yes      72
7040  4801-JZAZL  Female              0     Yes        Yes      11
7041  8361-LTMKD    Male              1     Yes         No       4
7042  3186-AJIEK    Male              0      No         No      66

     PhoneService     MultipleLines InternetService OnlineSecurity  ...  \
0              No  No phone service             DSL             No  ...
1             Yes                No             DSL            Yes  ...
2             Yes                No             DSL            Yes  ...
3              No  No phone service             DSL            Yes  ...
4             Yes                No     Fiber optic             No  ...
...           ...               ...             ...            ...  ...
7038          Yes               Yes             DSL            Yes  ...
7039          Yes               Yes     Fiber optic             No  ...
7040           No  No phone service             DSL            Yes  ...
7041          Yes               Yes     Fiber optic             No  ...
7042          Yes                No     Fiber optic            Yes  ...

     DeviceProtection TechSupport StreamingTV StreamingMovies        Contract  \
```

```
0                  No          No          No              No  Month-to-month
1                 Yes          No          No              No        One year
2                  No          No          No              No  Month-to-month
3                 Yes         Yes          No              No        One year
4                  No          No          No              No  Month-to-month
...               ...         ...         ...             ...             ...
7038              Yes         Yes         Yes             Yes        One year
7039              Yes          No         Yes             Yes        One year
7040               No          No          No              No  Month-to-month
7041               No          No          No              No  Month-to-month
7042              Yes         Yes         Yes             Yes        Two year

     PaperlessBilling              PaymentMethod MonthlyCharges  TotalCharges  \
0                 Yes           Electronic check          29.85         29.85
1                  No               Mailed check          56.95        1889.5
2                 Yes               Mailed check          53.85        108.15
3                  No  Bank transfer (automatic)          42.30       1840.75
4                 Yes           Electronic check          70.70        151.65
...               ...                        ...            ...           ...
7038              Yes               Mailed check          84.80        1990.5
7039              Yes    Credit card (automatic)         103.20        7362.9
7040              Yes           Electronic check          29.60        346.45
7041              Yes               Mailed check          74.40         306.6
7042              Yes  Bank transfer (automatic)         105.65        6844.5

     Churn
0       No
1       No
2      Yes
3       No
4      Yes
...    ...
7038    No
7039    No
7040    No
7041   Yes
7042    No

[7043 rows x 21 columns]
```

[10]:
```
print(data.columns)
```

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

[11]:
```
data.shape
```

[11]:
```
(7043, 21)
```

[12]:
```
print(data.head())
```

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

      MultipleLines InternetService OnlineSecurity  ... DeviceProtection  \
0  No phone service             DSL             No  ...               No
1                No             DSL            Yes  ...              Yes
2                No             DSL            Yes  ...               No
3  No phone service             DSL            Yes  ...              Yes
4                No     Fiber optic             No  ...               No

  TechSupport StreamingTV StreamingMovies        Contract PaperlessBilling  \
0          No          No              No  Month-to-month              Yes
1          No          No              No        One year               No
2          No          No              No  Month-to-month              Yes
3         Yes          No              No        One year               No
4          No          No              No  Month-to-month              Yes

               PaymentMethod MonthlyCharges  TotalCharges Churn
0           Electronic check          29.85         29.85    No
1               Mailed check          56.95        1889.5    No
2               Mailed check          53.85        108.15   Yes
3  Bank transfer (automatic)          42.30       1840.75    No
4           Electronic check          70.70        151.65   Yes

[5 rows x 21 columns]
```

[13]:
```
print(data.tail())
```

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

     PhoneService     MultipleLines InternetService OnlineSecurity  ...  \
7038          Yes               Yes             DSL            Yes  ...
7039          Yes               Yes     Fiber optic             No  ...
```

```
7040          No  No phone service             DSL             Yes  ...
7041         Yes               Yes     Fiber optic              No  ...
7042         Yes                No     Fiber optic             Yes  ...

     DeviceProtection TechSupport StreamingTV StreamingMovies        Contract  \
7038              Yes         Yes         Yes             Yes        One year
7039              Yes          No         Yes             Yes        One year
7040               No          No          No              No  Month-to-month
7041               No          No          No              No  Month-to-month
7042              Yes         Yes         Yes             Yes        Two year

     PaperlessBilling             PaymentMethod MonthlyCharges  TotalCharges  \
7038              Yes              Mailed check          84.80        1990.5
7039              Yes   Credit card (automatic)         103.20        7362.9
7040              Yes          Electronic check          29.60        346.45
7041              Yes              Mailed check          74.40         306.6
7042              Yes Bank transfer (automatic)         105.65        6844.5

     Churn
7038    No
7039    No
7040    No
7041   Yes
7042    No

[5 rows x 21 columns]
```

[14]:
```
data.nunique()
```

[14]:
```
customerID          7043
gender                 2
SeniorCitizen          2
Partner                2
Dependents             2
tenure                73
PhoneService           2
MultipleLines          3
InternetService        3
OnlineSecurity         3
OnlineBackup           3
DeviceProtection       3
TechSupport            3
StreamingTV            3
StreamingMovies        3
Contract               3
PaperlessBilling       2
PaymentMethod          4
```

```
MonthlyCharges      1585
TotalCharges        6531
Churn                  2
dtype: int64
```

[15]:
```
data.isna().sum()
```

[15]:
```
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
```

[16]:
```
data.isnull().sum()
```

[16]:
```
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
```

[17]:
```
# Check the number of rows before removing duplicates
print("Number of rows before removing duplicates:", len(data))
```

```
Number of rows before removing duplicates: 7043
```

[18]:
```
# Remove duplicate records
data_cleaned = data.drop_duplicates()
```

[19]:
```
# Remove duplicate records
data_cleaned = data.drop_duplicates()
```

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

[20]:

	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

[23]:
```
# Measure of frequency destribution
unique, counts = np.unique(data['tenure'], return_counts=True)
print(unique, counts)
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71
 72] [ 11 613 238 200 176 133 110 131 123 119 116  99 117 109  76  99  80  87
  97  73  71  63  90  85  94  79  79  72  57  72  72  65  69  64  65  88
  50  65  59  56  64  70  65  65  51  61  74  68  64  66  68  68  80  70
  68  64  80  65  67  60  76  76  70  72  80  76  89  98 100  95 119 170
 362]
```

[24]:
```
# Measure of frequency destribution
unique, counts = np.unique(data['MonthlyCharges'], return_counts=True)
print(unique, counts)
```

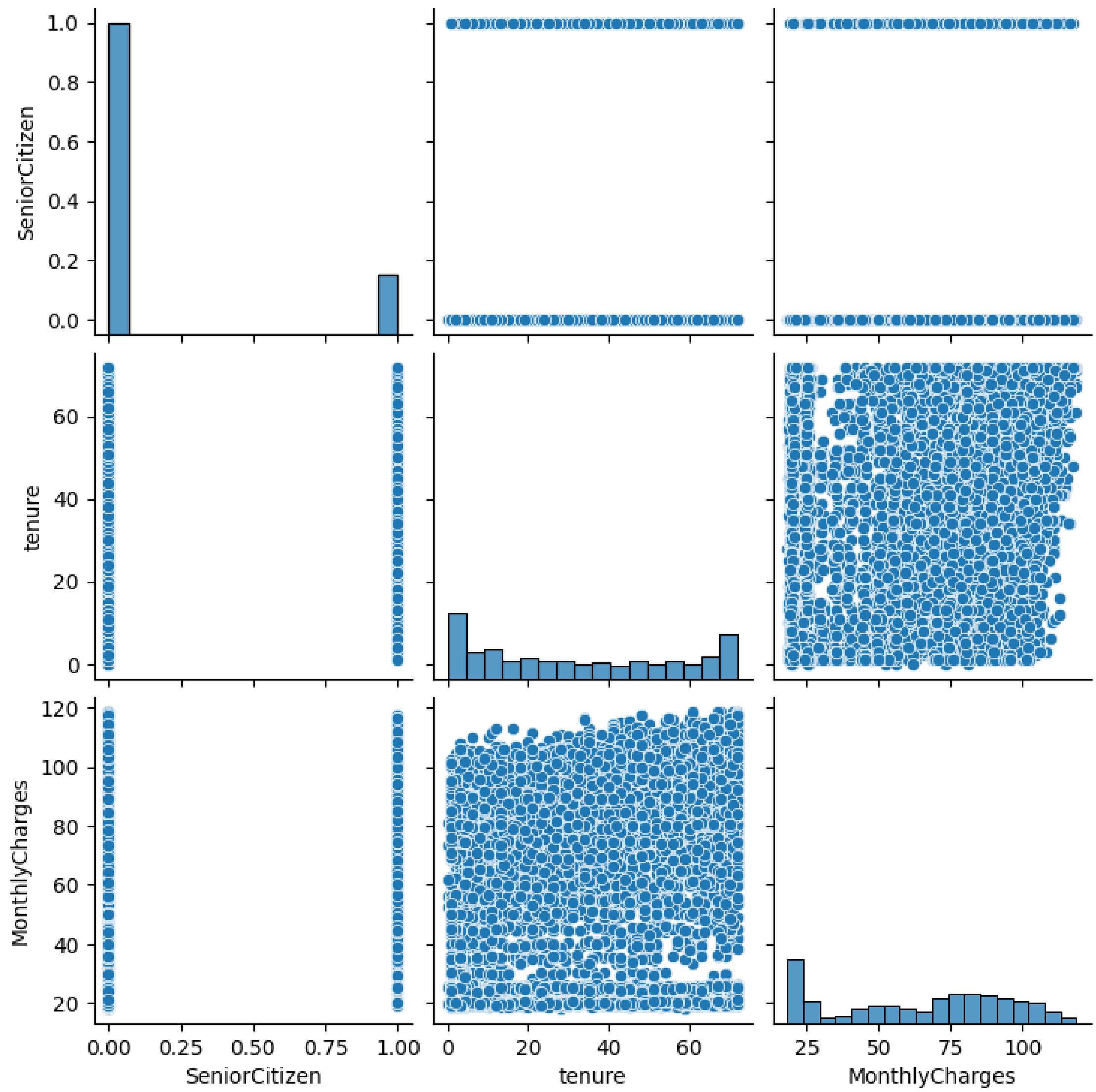
```
[ 18.25  18.4   18.55 ... 118.6   118.65 118.75] [1 1 1 ... 2 1 1]
```

[25]:
```
# Measure of frequency destribution
unique, counts = np.unique(data['TotalCharges'], return_counts=True)
print(unique, counts)
```

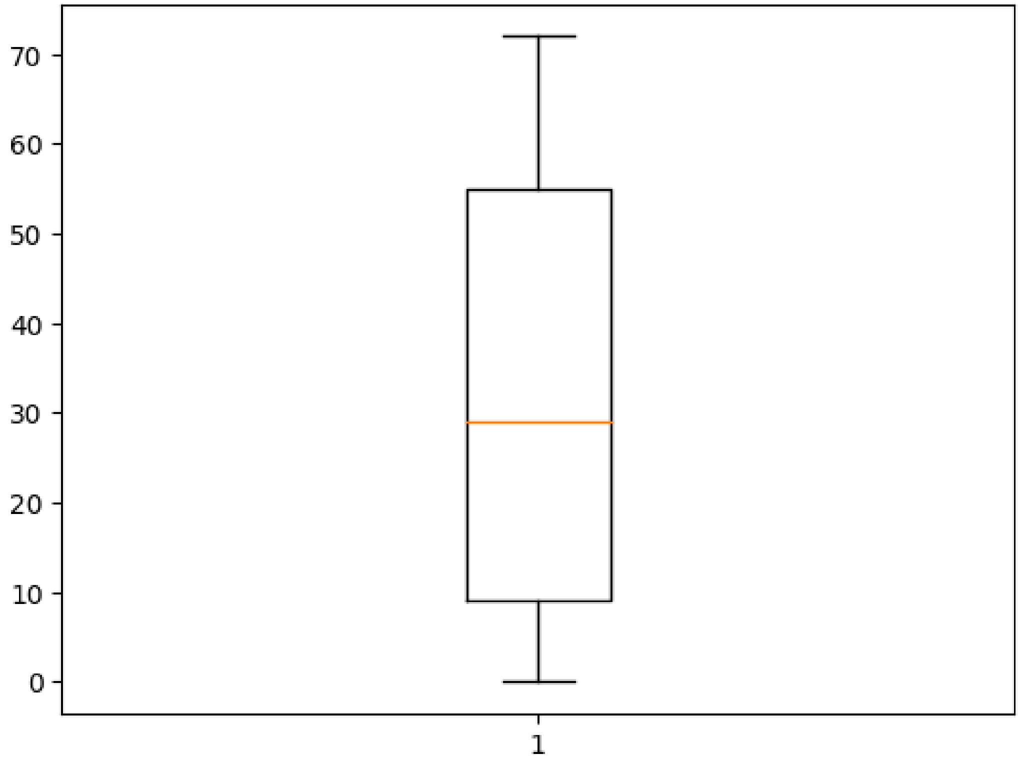
```
[' ' '100.2' '100.25' ... '999.45' '999.8' '999.9'] [11  1  1 ...  1  1  1]
```

[27]:
```
sns.pairplot(data)
```

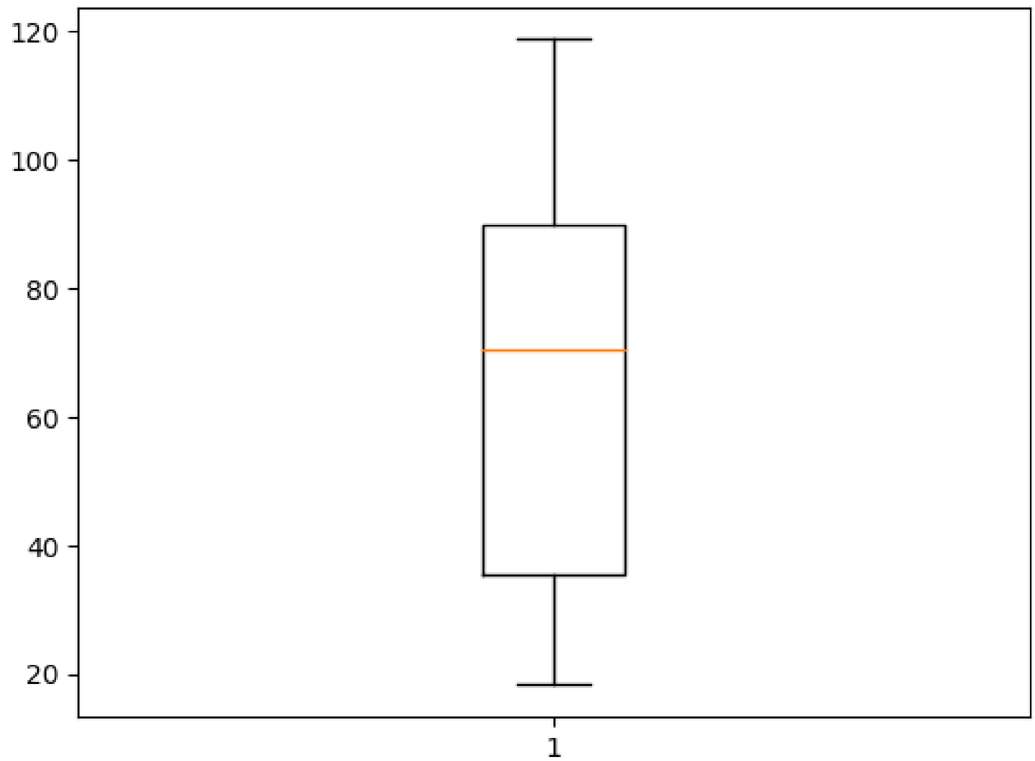
[27]:
```
<seaborn.axisgrid.PairGrid at 0x245ff749610>
```



[30]:
```
plt.boxplot(data['tenure'])
plt.show()
```



[31]:
```
plt.boxplot(data['MonthlyCharges'])
plt.show()
```



[32]:
```
X = data.drop("Churn", axis=1)
y = data["Churn"]
```

[33]:
```
# 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)
```

[34]:
```
X_train.shape
```

[34]: (5634, 20)

[35]:
```
y_train.shape
```

[35]: (5634,)

[36]:
```
X_test.shape
```

[36]: (1409, 20)

[37]:
```
y_test.shape
```

[37]: (1409,)

[39]:
```python
# Export the cleaned dataset to a CSV file
data.to_csv("./datasets/Cleaned_Telecom_Customer_Churn.csv", index=False)
```

[ ]:
```python

```

[ ]:
```
Name: Thorave Avishkar Shrikrushna
Roll No: 65
```

4 Title : Data Wrangling on Real Estate Market

[2]:
```python
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import warnings
```

[3]:
```python
# Supressing update warnings
warnings.filterwarnings('ignore')
```

[4]:
```python
df1 = pd.read_csv("./datasets/Bengaluru_House_Data.csv")
```

[5]:
```python
df1.head()
```

[5]:
```
              area_type   availability                  location       size  \
0  Super built-up  Area         19-Dec  Electronic City Phase II      2 BHK
1            Plot  Area  Ready To Move          Chikka Tirupathi  4 Bedroom
2        Built-up  Area  Ready To Move               Uttarahalli      3 BHK
3  Super built-up  Area  Ready To Move        Lingadheeranahalli      3 BHK
4  Super built-up  Area  Ready To Move                  Kothanur      2 BHK

   society total_sqft  bath  balcony   price
0  Coomee        1056   2.0      1.0   39.07
1  Theanmp       2600   5.0      3.0  120.00
2      NaN       1440   2.0      3.0   62.00
3  Soiewre       1521   3.0      1.0   95.00
4      NaN       1200   2.0      1.0   51.00
```

[6]:
```python
df1.shape
```

[6]:
```
(13320, 9)
```

[7]:
```python
df1.columns
```

[7]:
```
Index(['area_type', 'availability', 'location', 'size', 'society',
       'total_sqft', 'bath', 'balcony', 'price'],
      dtype='object')
```

[8]:
```python
df1['area_type']
```

[8]:
```
0        Super built-up  Area
1                  Plot  Area
2              Built-up  Area
3        Super built-up  Area
```

```
4        Super built-up  Area
                  ...
13315          Built-up  Area
13316    Super built-up  Area
13317          Built-up  Area
13318    Super built-up  Area
13319    Super built-up  Area
Name: area_type, Length: 13320, dtype: object
```

[9]:
```
df1['area_type'].unique()
```

[9]:
```
array(['Super built-up  Area', 'Plot  Area', 'Built-up  Area',
       'Carpet  Area'], dtype=object)
```

[10]:
```
df1['area_type'].value_counts()
```

[10]:
```
Super built-up  Area    8790
Built-up  Area          2418
Plot  Area              2025
Carpet  Area              87
Name: area_type, dtype: int64
```

[11]:
```
df2 = df1.drop(['area_type','society','balcony','availability'],axis='columns')
```

[12]:
```
df2.shape
```

[12]:
```
(13320, 5)
```

[13]:
```
df2.isnull().sum()
```

[13]:
```
location       1
size          16
total_sqft     0
bath          73
price          0
dtype: int64
```

[14]:
```
df2.shape
```

[14]:
```
(13320, 5)
```

[15]:
```
df3 = df2.dropna()
df3.isnull().sum()
```

[15]:
```
location      0
size          0
total_sqft    0
bath          0
```

```
price          0
dtype: int64
```

[16]:
```
df3.shape
```

[16]:
```
(13246, 5)
```

[17]:
```
df3['size'].unique()
```

[17]:
```
array(['2 BHK', '4 Bedroom', '3 BHK', '4 BHK', '6 Bedroom', '3 Bedroom',
       '1 BHK', '1 RK', '1 Bedroom', '8 Bedroom', '2 Bedroom',
       '7 Bedroom', '5 BHK', '7 BHK', '6 BHK', '5 Bedroom', '11 BHK',
       '9 BHK', '9 Bedroom', '27 BHK', '10 Bedroom', '11 Bedroom',
       '10 BHK', '19 BHK', '16 BHK', '43 Bedroom', '14 BHK', '8 BHK',
       '12 Bedroom', '13 BHK', '18 Bedroom'], dtype=object)
```

[18]:
```
df3['bhk'] = df3['size'].apply(lambda x: int(x.split(' ')[0]))
```

[19]:
```
df3.head()
```

[19]:

	location	size	total_sqft	bath	price	bhk
0	Electronic City Phase II	2 BHK	1056	2.0	39.07	2
1	Chikka Tirupathi	4 Bedroom	2600	5.0	120.00	4
2	Uttarahalli	3 BHK	1440	2.0	62.00	3
3	Lingadheeranahalli	3 BHK	1521	3.0	95.00	3
4	Kothanur	2 BHK	1200	2.0	51.00	2

[20]:
```
df3.bhk.unique()
```

[20]:
```
array([ 2,  4,  3,  6,  1,  8,  7,  5, 11,  9, 27, 10, 19, 16, 43, 14, 12,
       13, 18], dtype=int64)
```

[21]:
```
df3[df3.bhk>20]
```

[21]:

	location	size	total_sqft	bath	price	bhk
1718	2Electronic City Phase II	27 BHK	8000	27.0	230.0	27
4684	Munnekollal	43 Bedroom	2400	40.0	660.0	43

[22]:
```
df3.total_sqft.unique()
```

[22]:
```
array(['1056', '2600', '1440', ..., '1133 - 1384', '774', '4689'],
      dtype=object)
```

[23]:
```
def is_float(x):
    try:
        float(x)
        return True
    except(ValueError, TypeError):
```

```
        return False
```

[24]:
```
df3[~df3['total_sqft'].apply(is_float)].head(10)
```

[24]:

	location	size	total_sqft	bath	price	bhk
30	Yelahanka	4 BHK	2100 - 2850	4.0	186.000	4
122	Hebbal	4 BHK	3067 - 8156	4.0	477.000	4
137	8th Phase JP Nagar	2 BHK	1042 - 1105	2.0	54.005	2
165	Sarjapur	2 BHK	1145 - 1340	2.0	43.490	2
188	KR Puram	2 BHK	1015 - 1540	2.0	56.800	2
410	Kengeri	1 BHK	34.46Sq. Meter	1.0	18.500	1
549	Hennur Road	2 BHK	1195 - 1440	2.0	63.770	2
648	Arekere	9 Bedroom	4125Perch	9.0	265.000	9
661	Yelahanka	2 BHK	1120 - 1145	2.0	48.130	2
672	Bettahalsoor	4 Bedroom	3090 - 5002	4.0	445.000	4

[25]:
```
def convert_sqft_to_num(x):
    tokens = x.split('-')
    if len(tokens) == 2:
        try:
            return (float(tokens[0])+float(tokens[1]))/2
        except ValueError:
            return None
    try:
        return float(x)
    except ValueError:
        return None

result = convert_sqft_to_num('2100 - 2850')
print(result)
```

2475.0

[26]:
```
convert_sqft_to_num('34.46Sq. Meter')
df4 = df3.copy()
df4.total_sqft = df4.total_sqft.apply(convert_sqft_to_num)
df4
```

[26]:

	location	size	total_sqft	bath	price	bhk
0	Electronic City Phase II	2 BHK	1056.0	2.0	39.07	2
1	Chikka Tirupathi	4 Bedroom	2600.0	5.0	120.00	4
2	Uttarahalli	3 BHK	1440.0	2.0	62.00	3
3	Lingadheeranahalli	3 BHK	1521.0	3.0	95.00	3
4	Kothanur	2 BHK	1200.0	2.0	51.00	2
...	...	...	...	...	...	...
13315	Whitefield	5 Bedroom	3453.0	4.0	231.00	5
13316	Richards Town	4 BHK	3600.0	5.0	400.00	4
13317	Raja Rajeshwari Nagar	2 BHK	1141.0	2.0	60.00	2

```
13318           Padmanabhanagar      4 BHK      4689.0   4.0  488.00    4
13319              Doddathoguru      1 BHK       550.0   1.0   17.00    1

[13246 rows x 6 columns]
```

[27]:
```python
df4 = df4[df4.total_sqft.notnull()]
df4
```

[27]:
```
                       location       size  total_sqft  bath   price  bhk
0      Electronic City Phase II      2 BHK      1056.0   2.0   39.07    2
1              Chikka Tirupathi  4 Bedroom      2600.0   5.0  120.00    4
2                   Uttarahalli      3 BHK      1440.0   2.0   62.00    3
3            Lingadheeranahalli      3 BHK      1521.0   3.0   95.00    3
4                      Kothanur      2 BHK      1200.0   2.0   51.00    2
...                         ...        ...         ...   ...     ...  ...
13315                Whitefield  5 Bedroom      3453.0   4.0  231.00    5
13316             Richards Town      4 BHK      3600.0   5.0  400.00    4
13317     Raja Rajeshwari Nagar      2 BHK      1141.0   2.0   60.00    2
13318           Padmanabhanagar      4 BHK      4689.0   4.0  488.00    4
13319              Doddathoguru      1 BHK       550.0   1.0   17.00    1

[13200 rows x 6 columns]
```

[28]:
```python
df4.loc[30]
```

[28]:
```
location      Yelahanka
size              4 BHK
total_sqft       2475.0
bath                4.0
price             186.0
bhk                   4
Name: 30, dtype: object
```

[29]:
```python
df5 = df4.copy()
df5['price_per_sqft'] = df5['price']*100000/df5['total_sqft']
df5.head()
```

[29]:
```
                   location       size  total_sqft  bath   price  bhk  \
0  Electronic City Phase II      2 BHK      1056.0   2.0   39.07    2
1          Chikka Tirupathi  4 Bedroom      2600.0   5.0  120.00    4
2               Uttarahalli      3 BHK      1440.0   2.0   62.00    3
3        Lingadheeranahalli      3 BHK      1521.0   3.0   95.00    3
4                  Kothanur      2 BHK      1200.0   2.0   51.00    2

   price_per_sqft
0     3699.810606
1     4615.384615
```

```
2      4305.555556
3      6245.890861
4      4250.000000
```

[30]:
```
df5_stats = df5['price_per_sqft'].describe()
df5_stats
```

[30]:
```
count    1.320000e+04
mean     7.920759e+03
std      1.067272e+05
min      2.678298e+02
25%      4.267701e+03
50%      5.438331e+03
75%      7.317073e+03
max      1.200000e+07
Name: price_per_sqft, dtype: float64
```

[31]:
```
df5.to_csv("./datasets/bhp.csv",index=False)
```

[32]:
```
df5.location = df5.location.apply(lambda x: x.strip())
location_stats = df5['location'].value_counts(ascending=False)
location_stats
```

[32]:
```
Whitefield                     533
Sarjapur  Road                 392
Electronic City                304
Kanakpura Road                 264
Thanisandra                    235
                              ...
Rajanna Layout                   1
Subramanyanagar                  1
Lakshmipura Vidyaanyapura        1
Malur Hosur Road                 1
Abshot Layout                    1
Name: location, Length: 1287, dtype: int64
```

[33]:
```
len(location_stats[location_stats>10])
```

[33]: 240

[34]:
```
len(location_stats)
```

[34]: 1287

[35]:
```
len(location_stats[location_stats<=10])
```

[35]: 1047

[36]:
```
location_stats_less_than_10 = location_stats[location_stats<=10]
location_stats_less_than_10
```

[36]:
```
BTM 1st Stage                10
Gunjur Palya                 10
Nagappa Reddy Layout         10
Sector 1 HSR Layout          10
Thyagaraja Nagar             10
                             ..
Rajanna Layout                1
Subramanyanagar               1
Lakshmipura Vidyaanyapura     1
Malur Hosur Road              1
Abshot Layout                 1
Name: location, Length: 1047, dtype: int64
```

[37]:
```
len(df5.location.unique())
```

[37]:
```
1287
```

[38]:
```
df5.location = df5.location.apply(lambda x: 'other' if x in␣
 ↪location_stats_less_than_10 else x)
len(df5.location.unique())
```

[38]:
```
241
```

[39]:
```
df5.head(10)
```

[39]:
```
                 location       size  total_sqft  bath   price  bhk  \
0  Electronic City Phase II      2 BHK      1056.0   2.0   39.07    2
1        Chikka Tirupathi  4 Bedroom      2600.0   5.0  120.00    4
2             Uttarahalli      3 BHK      1440.0   2.0   62.00    3
3      Lingadheeranahalli      3 BHK      1521.0   3.0   95.00    3
4                Kothanur      2 BHK      1200.0   2.0   51.00    2
5              Whitefield      2 BHK      1170.0   2.0   38.00    2
6        Old Airport Road      4 BHK      2732.0   4.0  204.00    4
7            Rajaji Nagar      4 BHK      3300.0   4.0  600.00    4
8            Marathahalli      3 BHK      1310.0   3.0   63.25    3
9                   other  6 Bedroom      1020.0   6.0  370.00    6

   price_per_sqft
0     3699.810606
1     4615.384615
2     4305.555556
3     6245.890861
4     4250.000000
5     3247.863248
6     7467.057101
```

```
7     18181.818182
8      4828.244275
9     36274.509804
```

[40]:
```
df5[df5.total_sqft/df5.bhk<300].head()
```

[40]:
```
               location       size  total_sqft  bath  price  bhk  \
9                 other  6 Bedroom      1020.0   6.0  370.0    6
45           HSR Layout  8 Bedroom       600.0   9.0  200.0    8
58         Murugeshpalya  6 Bedroom      1407.0   4.0  150.0    6
68  Devarachikkanahalli  8 Bedroom      1350.0   7.0   85.0    8
70                other  3 Bedroom       500.0   3.0  100.0    3

    price_per_sqft
9     36274.509804
45    33333.333333
58    10660.980810
68     6296.296296
70    20000.000000
```

[41]:
```
df5.shape
```

[41]:
```
(13200, 7)
```

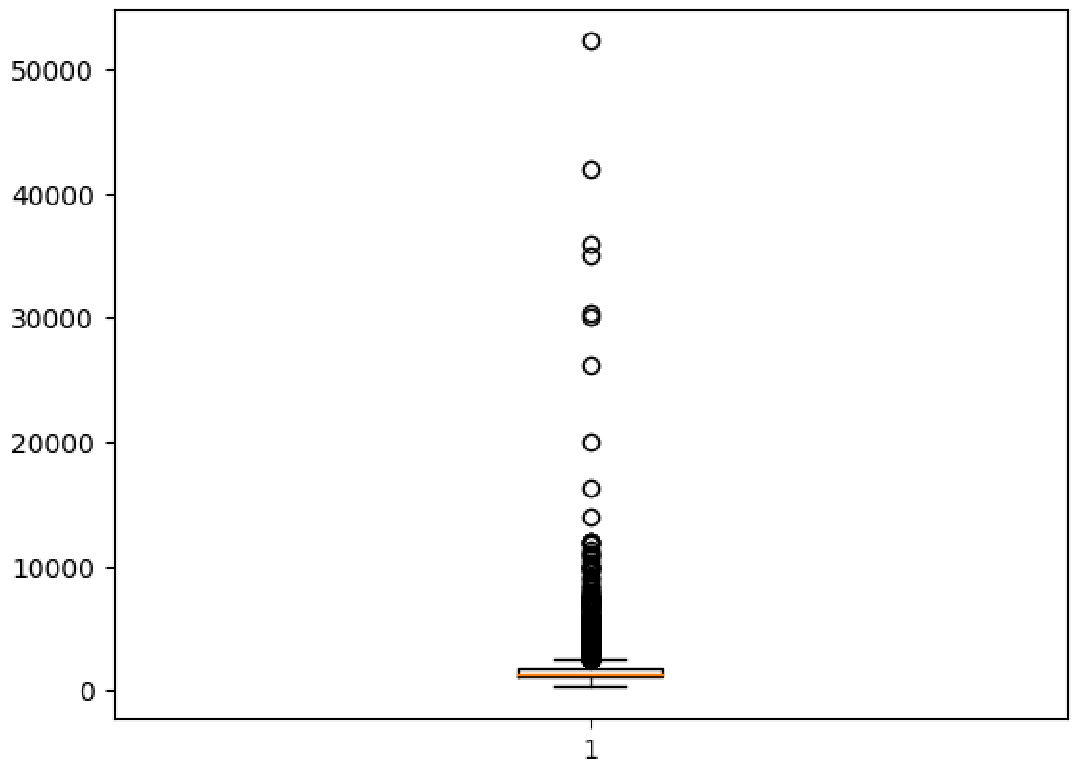
[42]:
```
df6 = df5[~(df5.total_sqft/df5.bhk<300)]
df6.shape
```

[42]:
```
(12456, 7)
```

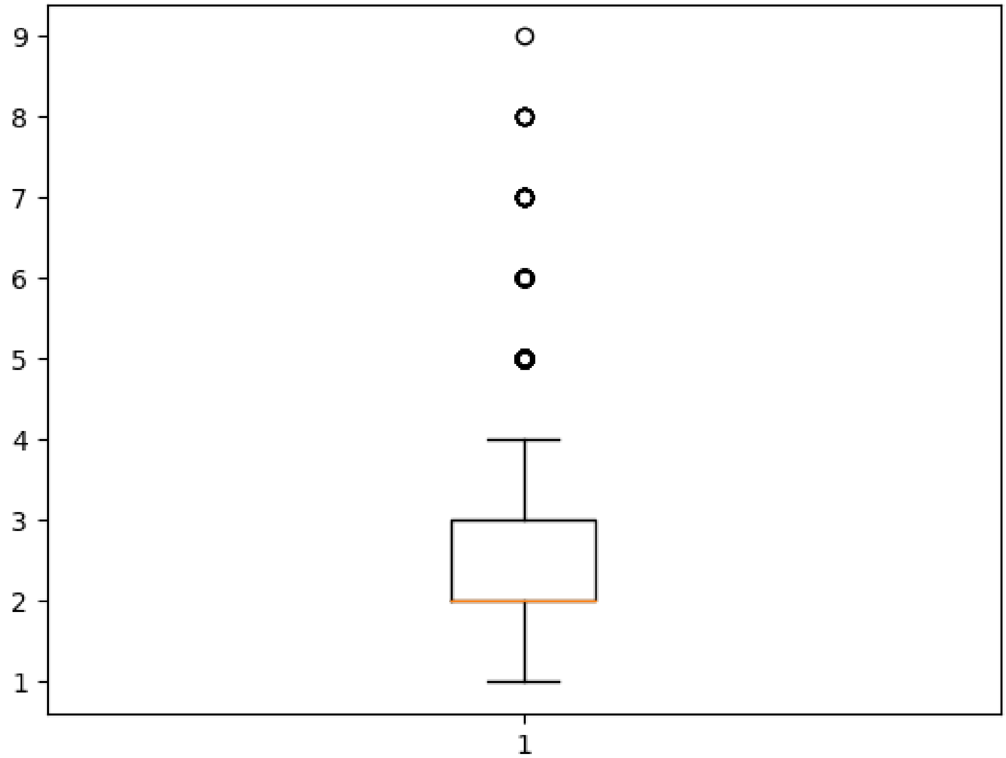
[43]:
```
df6.columns
```

[43]:
```
Index(['location', 'size', 'total_sqft', 'bath', 'price', 'bhk',
       'price_per_sqft'],
      dtype='object')
```

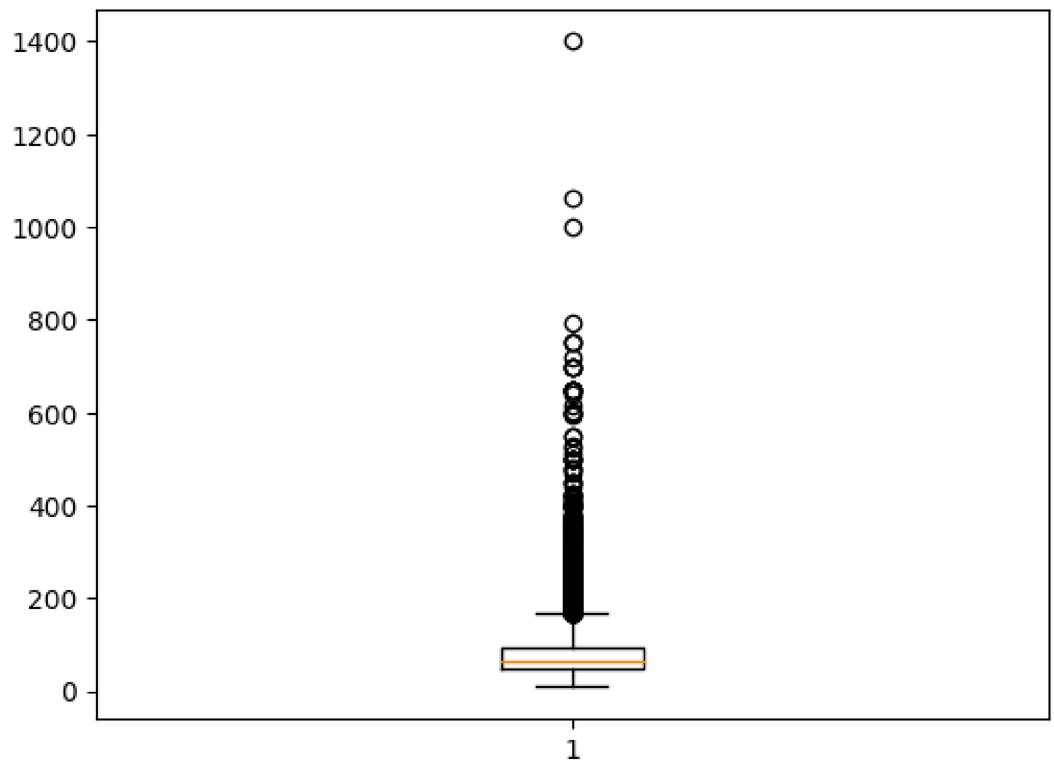
[44]:
```
plt.boxplot(df6['total_sqft'])
plt.show()
```



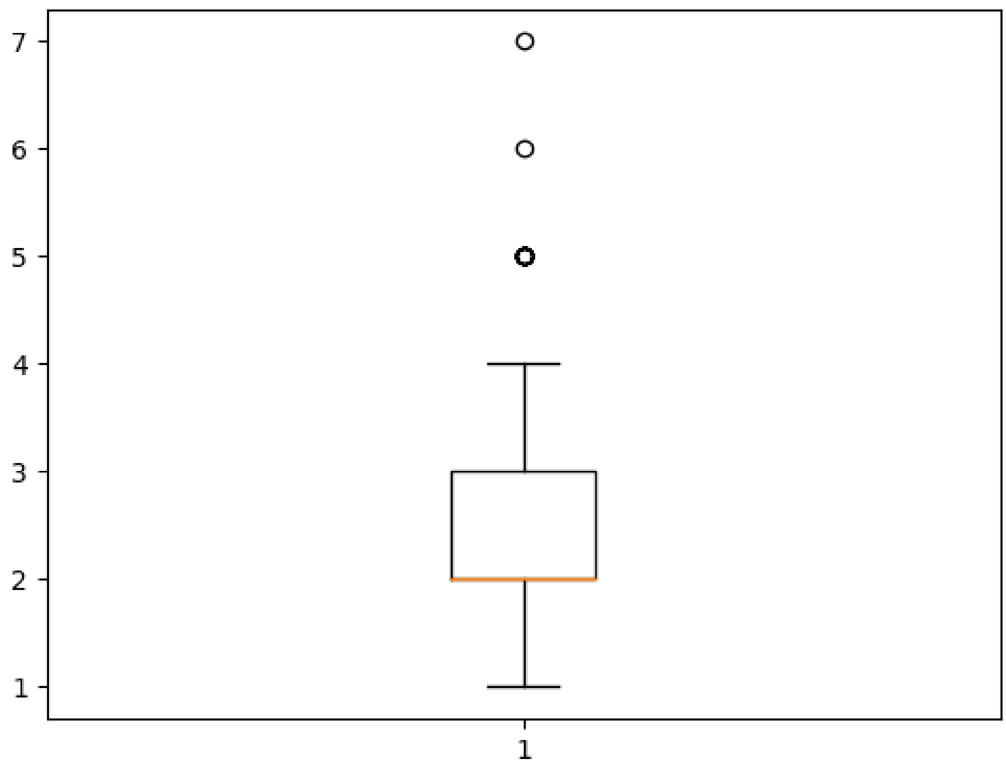
[45]:
```
Q1 = np.percentile(df6['total_sqft'], 25.) # 25th percentile of the data of the␣
↪given feature
Q3 = np.percentile(df6['total_sqft'], 75.) # 75th percentile of the data of the␣
↪given feature
IQR = Q3-Q1 #Interquartile Range
ll = Q1 - (1.5*IQR)
ul = Q3 + (1.5*IQR)
upper_outliers = df6[df6['total_sqft'] > ul].index.tolist()
lower_outliers = df6[df6['total_sqft'] < ll].index.tolist()
bad_indices = list(set(upper_outliers + lower_outliers))
drop = True
if drop:
    df6.drop(bad_indices, inplace = True, errors = 'ignore')

plt.boxplot(df6['bath'])
plt.show()
```



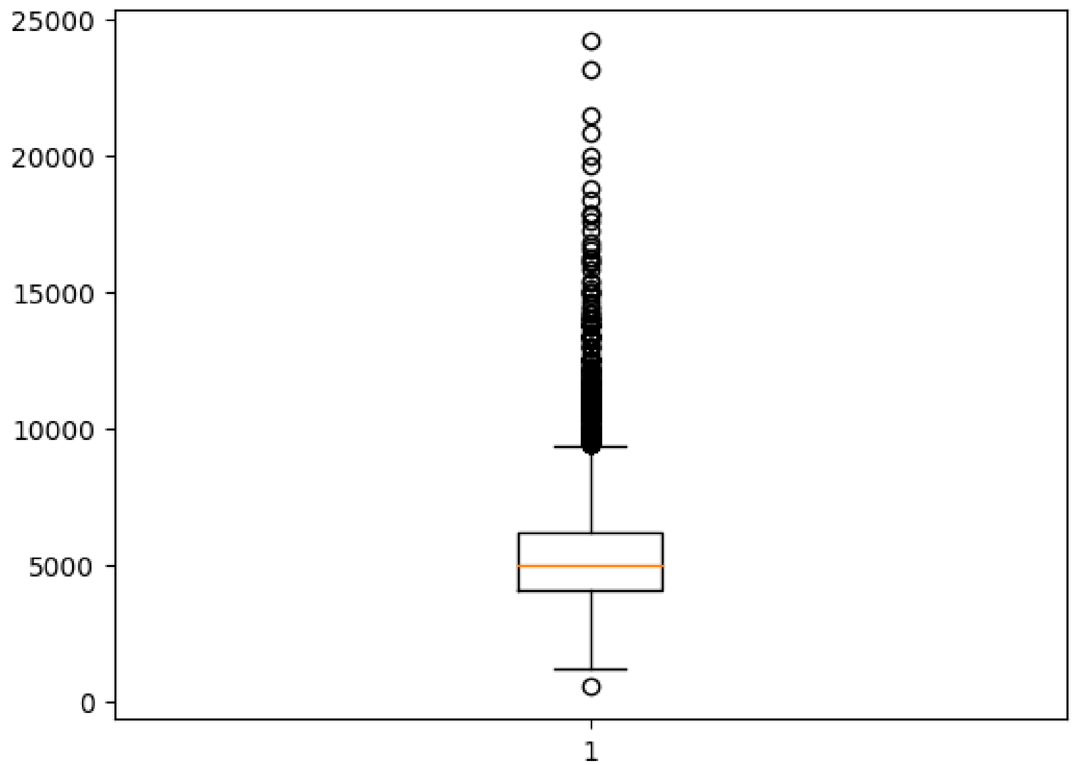
[46]:
```
Q1 = np.percentile(df6['bath'], 25.) # 25th percentile of the data of the given␣
↪feature
Q3 = np.percentile(df6['bath'], 75.) # 75th percentile of the data of the given␣
↪feature
IQR = Q3-Q1 #Interquartile Range
ll = Q1 - (1.5*IQR)
ul = Q3 + (1.5*IQR)
upper_outliers = df6[df6['bath'] > ul].index.tolist()
lower_outliers = df6[df6['bath'] < ll].index.tolist()
bad_indices = list(set(upper_outliers + lower_outliers))
drop = True
if drop:
    df6.drop(bad_indices, inplace = True, errors = 'ignore')
plt.boxplot(df6['price'])
plt.show()
```



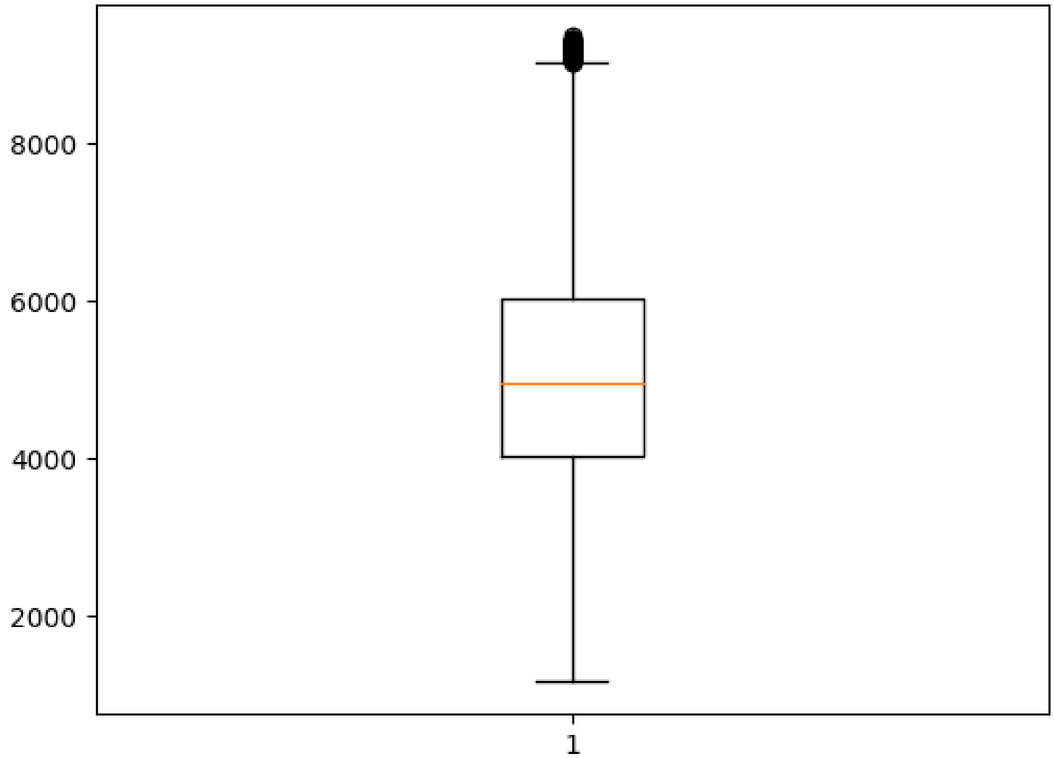
[47]:
```
Q1 = np.percentile(df6['price'], 25.) # 25th percentile of the data of the given␣
↪feature
Q3 = np.percentile(df6['price'], 75.) # 75th percentile of the data of the given␣
↪feature
IQR = Q3-Q1 #Interquartile Range
ll = Q1 - (1.5*IQR)
ul = Q3 + (1.5*IQR)

upper_outliers = df6[df6['price'] > ul].index.tolist()
lower_outliers = df6[df6['price'] < ll].index.tolist()
bad_indices = list(set(upper_outliers + lower_outliers))
drop = True
if drop:
    df6.drop(bad_indices, inplace = True, errors = 'ignore')

plt.boxplot(df6['bhk'])
plt.show()
```



[48]:
```
Q1 = np.percentile(df6['bhk'], 25.) # 25th percentile of the data of the given␣
↪feature
Q3 = np.percentile(df6['bhk'], 75.) # 75th percentile of the data of the given␣
↪feature
IQR = Q3-Q1 #Interquartile Range
ll = Q1 - (1.5*IQR)
ul = Q3 + (1.5*IQR)
upper_outliers = df6[df6['bhk'] > ul].index.tolist()
lower_outliers = df6[df6['bhk'] < ll].index.tolist()
bad_indices = list(set(upper_outliers + lower_outliers))
drop = True
if drop:
    df6.drop(bad_indices, inplace = True, errors = 'ignore')

plt.boxplot(df6['price_per_sqft'])
plt.show()
```



[49]:
```
Q1 = np.percentile(df6['price_per_sqft'], 25.) # 25th percentile of the data of␣
↪the given feature
Q3 = np.percentile(df6['price_per_sqft'], 75.) # 75th percentile of the data of␣
↪the given feature
IQR = Q3-Q1 #Interquartile Range
ll = Q1 - (1.5*IQR)
ul = Q3 + (1.5*IQR)
upper_outliers = df6[df6['price_per_sqft'] > ul].index.tolist()
lower_outliers = df6[df6['price_per_sqft'] < ll].index.tolist()
bad_indices = list(set(upper_outliers + lower_outliers))
drop = True
if drop:
    df6.drop(bad_indices, inplace = True, errors = 'ignore')

plt.boxplot(df6['price_per_sqft'])
plt.show()
```



[50]:
```
df6.shape
```

[50]: (10090, 7)

[51]:
```
X = df6.drop(['price'],axis='columns')
X.head(3)
```

[51]:

	location	size	total_sqft	bath	bhk	price_per_sqft
0	Electronic City Phase II	2 BHK	1056.0	2.0	2	3699.810606
2	Uttarahalli	3 BHK	1440.0	2.0	3	4305.555556
3	Lingadheeranahalli	3 BHK	1521.0	3.0	3	6245.890861

[52]:
```
X.shape
```

[52]: (10090, 6)

[53]:
```
y = df6.price
y.head(3)
```

[53]:
```
0    39.07
2    62.00
3    95.00
```

```
Name: price, dtype: float64
```

[54]:
```
len(y)
```

[54]:
```
10090
```

[55]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.
 ↪2,random_state=10)

X_train.shape
```

[55]:
```
(8072, 6)
```

[56]:
```
y_train.shape
```

[56]:
```
(8072,)
```

[57]:
```
X_test.shape
```

[57]:
```
(2018, 6)
```

[58]:
```
y_test.shape
```

[58]:
```
(2018,)
```

[ ]:
```

```

```
[ ]: Name: Thorave Avishkar Shrikrushna
     Roll No:65
```

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

```python
[54]: import numpy as np
      import pandas as pd
      import matplotlib.pyplot as plt
      import seaborn as sns
      from sklearn.impute import SimpleImputer
      import warnings
```

```python
[55]: # Supressing update warnings
      warnings.filterwarnings('ignore')
```

```python
[56]: data = pd.read_csv("./datasets/data.csv", encoding="cp1252")
      data
```

```
[56]:        stn_code       sampling_date                           state   location  \
      0         150.0  February - M021990                  Andhra Pradesh  Hyderabad
      1         151.0  February - M021990                  Andhra Pradesh  Hyderabad
      2         152.0  February - M021990                  Andhra Pradesh  Hyderabad
      3         150.0     March - M031990                  Andhra Pradesh  Hyderabad
      4         151.0     March - M031990                  Andhra Pradesh  Hyderabad
      ...         ...                 ...                             ...        ...
      435737     SAMP            24-12-15                     West Bengal   ULUBERIA
      435738     SAMP            29-12-15                     West Bengal   ULUBERIA
      435739      NaN                 NaN  andaman-and-nicobar-islands          NaN
      435740      NaN                 NaN                     Lakshadweep        NaN
      435741      NaN                 NaN                         Tripura        NaN

                                                agency  \
      0                                            NaN
      1                                            NaN
      2                                            NaN
      3                                            NaN
      4                                            NaN
      ...                                          ...
      435737  West Bengal State Pollution Control Board
      435738  West Bengal State Pollution Control Board
      435739                                       NaN
      435740                                       NaN
      435741                                       NaN

                                          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
...                                   ...   ...   ...    ...  ...
435737                              RIRUO  22.0  50.0  143.0  NaN
435738                              RIRUO  20.0  46.0  171.0  NaN
435739                                NaN   NaN   NaN    NaN  NaN
435740                                NaN   NaN   NaN    NaN  NaN
435741                                NaN   NaN   NaN    NaN  NaN

             location_monitoring_station  pm2_5        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
...                                  ...    ...         ...
435737  Inside Rampal Industries,ULUBERIA    NaN  2015-12-24
435738  Inside Rampal Industries,ULUBERIA    NaN  2015-12-29
435739                               NaN    NaN         NaN
435740                               NaN    NaN         NaN
435741                               NaN    NaN         NaN

[435742 rows x 13 columns]
```

[41]:

```
data.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
```

[42]:
```
# Cleaning up name changes
data.state = data.state.replace({'Uttaranchal':'Uttarakhand'})
data.state[data.location == "Jamshedpur"] = data.state[data.location ==␣
↪'Jamshedpur'].replace({"Bihar":"Jharkhand"})
```

[43]:
```
# Changing types to uniform format
types = {
    "Residential": "R",
    "Residential and others": "RO",
    "Residential, Rural and other Areas": "RRO",
    "Industrial Area": "I",
    "Industrial Areas": "I",
    "Industrial": "I",
    "Sensitive Area": "S",
    "Sensitive Areas": "S",
    "Sensitive": "S",
    np.nan: "RRO"
}

data.type = data.type.replace(types)
data.head()
```

[43]:
```
   stn_code       sampling_date           state   location agency type  so2  \
0     150.0  February - M021990  Andhra Pradesh  Hyderabad    NaN  RRO  4.8
1     151.0  February - M021990  Andhra Pradesh  Hyderabad    NaN    I  3.1
2     152.0  February - M021990  Andhra Pradesh  Hyderabad    NaN  RRO  6.2
3     150.0     March - M031990  Andhra Pradesh  Hyderabad    NaN  RRO  6.3
4     151.0     March - M031990  Andhra Pradesh  Hyderabad    NaN    I  4.7

    no2  rspm  spm location_monitoring_station  pm2_5        date
0  17.4   NaN  NaN                         NaN    NaN  1990-02-01
1   7.0   NaN  NaN                         NaN    NaN  1990-02-01
2  28.5   NaN  NaN                         NaN    NaN  1990-02-01
3  14.7   NaN  NaN                         NaN    NaN  1990-03-01
4   7.5   NaN  NaN                         NaN    NaN  1990-03-01
```

[44]:
```
# defining columns of importance, which shall be used reguarly
VALUE_COLS = ['so2', 'no2', 'rspm', 'spm', 'pm2_5']
```

[45]:
```
# invoking SimpleImputer to fill missing values
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
data[VALUE_COLS] = imputer.fit_transform(data[VALUE_COLS])
```

[46]:
```
# checking to see if the dataset has any null values left over and the format
print(data.isnull().sum())
data.tail()
```

```
stn_code                               144077
```

```
sampling_date                      3
state                              0
location                           3
agency                        149481
type                               0
so2                                0
no2                                0
rspm                               0
spm                                0
location_monitoring_station    27491
pm2_5                              0
date                               7
dtype: int64
```

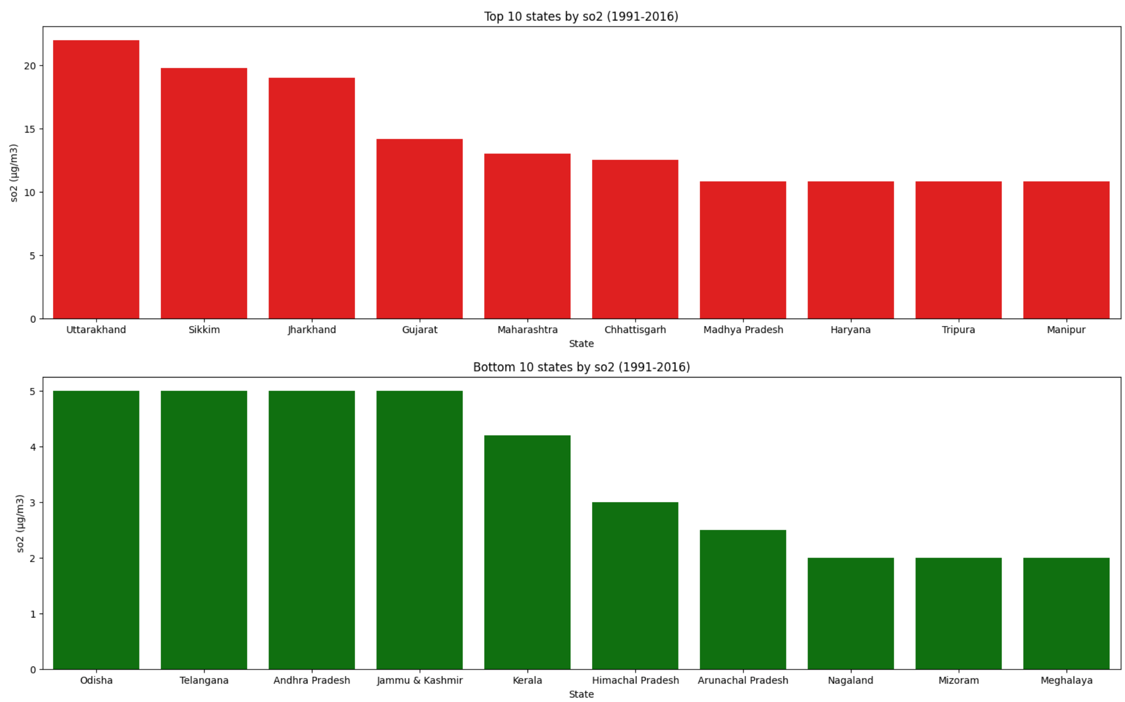
[46]:

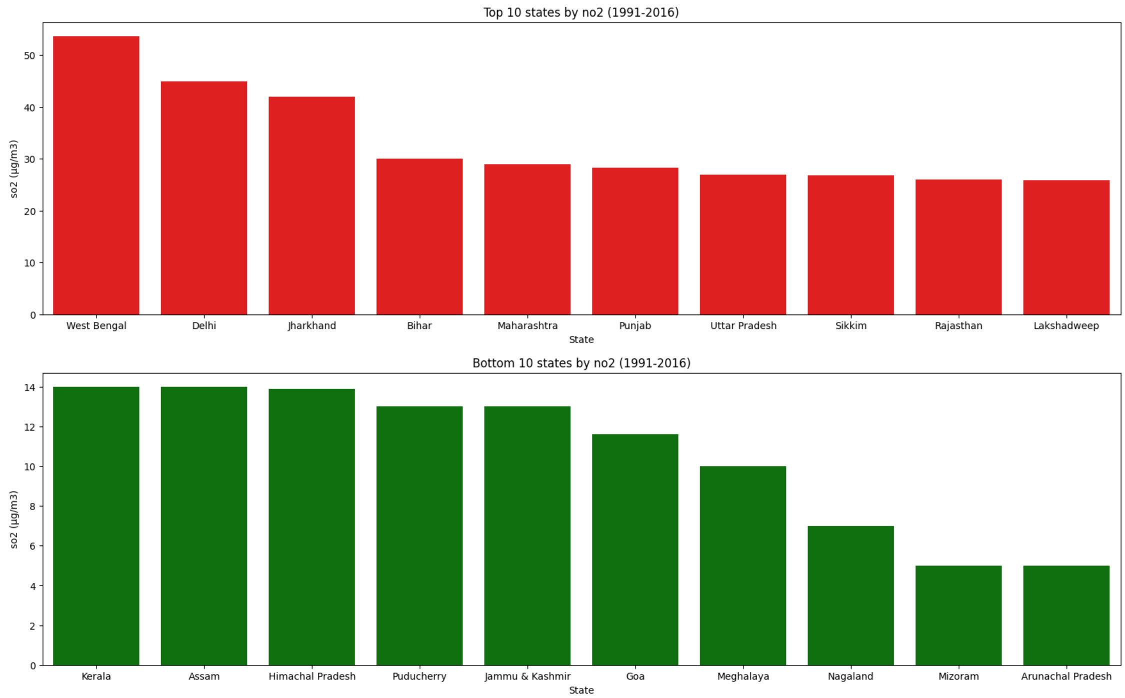
```
       stn_code sampling_date                         state  location  \
435737     SAMP      24-12-15                   West Bengal  ULUBERIA
435738     SAMP      29-12-15                   West Bengal  ULUBERIA
435739      NaN           NaN  andaman-and-nicobar-islands       NaN
435740      NaN           NaN                   Lakshadweep       NaN
435741      NaN           NaN                       Tripura       NaN

                                         agency   type        so2  \
435737  West Bengal State Pollution Control Board  RIRUO  22.000000
435738  West Bengal State Pollution Control Board  RIRUO  20.000000
435739                                        NaN    RRO  10.829414
435740                                        NaN    RRO  10.829414
435741                                        NaN    RRO  10.829414

              no2        rspm        spm       location_monitoring_station  \
435737  50.000000  143.000000  220.78348  Inside Rampal Industries,ULUBERIA
435738  46.000000  171.000000  220.78348  Inside Rampal Industries,ULUBERIA
435739  25.809623  108.832784  220.78348                                NaN
435740  25.809623  108.832784  220.78348                                NaN
435741  25.809623  108.832784  220.78348                                NaN

            pm2_5        date
435737  40.791467  2015-12-24
435738  40.791467  2015-12-29
435739  40.791467         NaN
435740  40.791467         NaN
435741  40.791467         NaN
```

[48]:

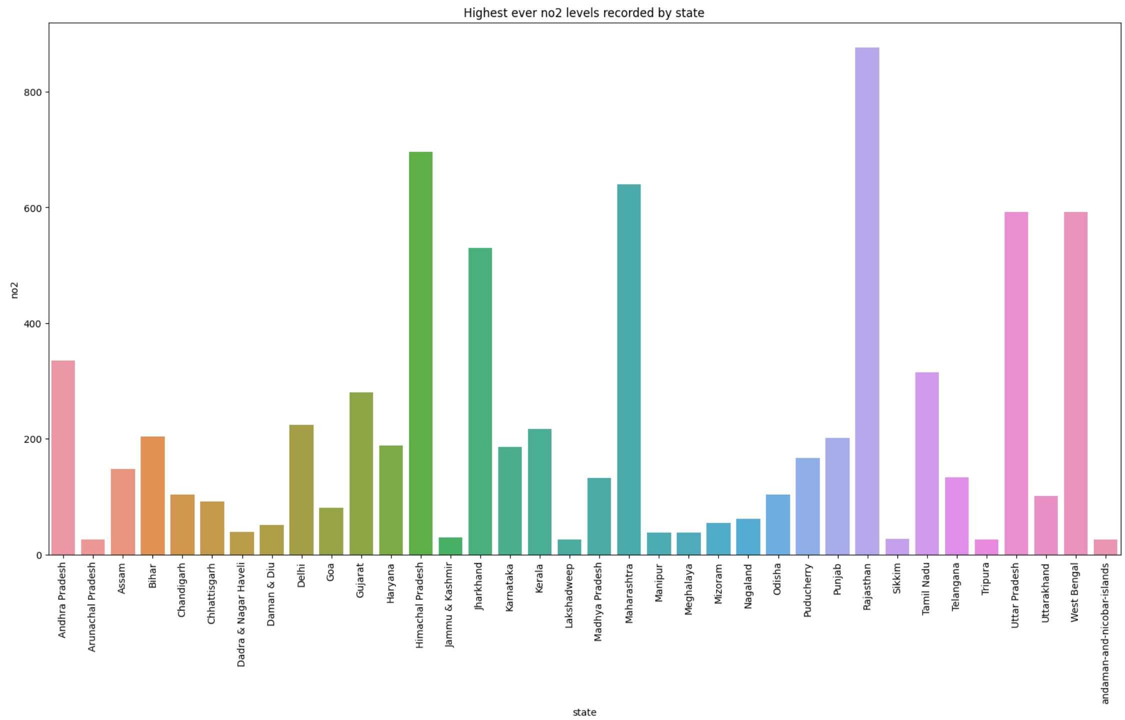
```
# Plotting highest and lowest ranking states
# defining a function to find and plot the top 10 and bottom 10 states for a␣
↪given indicator (defaults to SO2)
def top_and_bottom_10_states(indicator="so2"):
    fig, ax = plt.subplots(2,1, figsize=(20, 12))
```

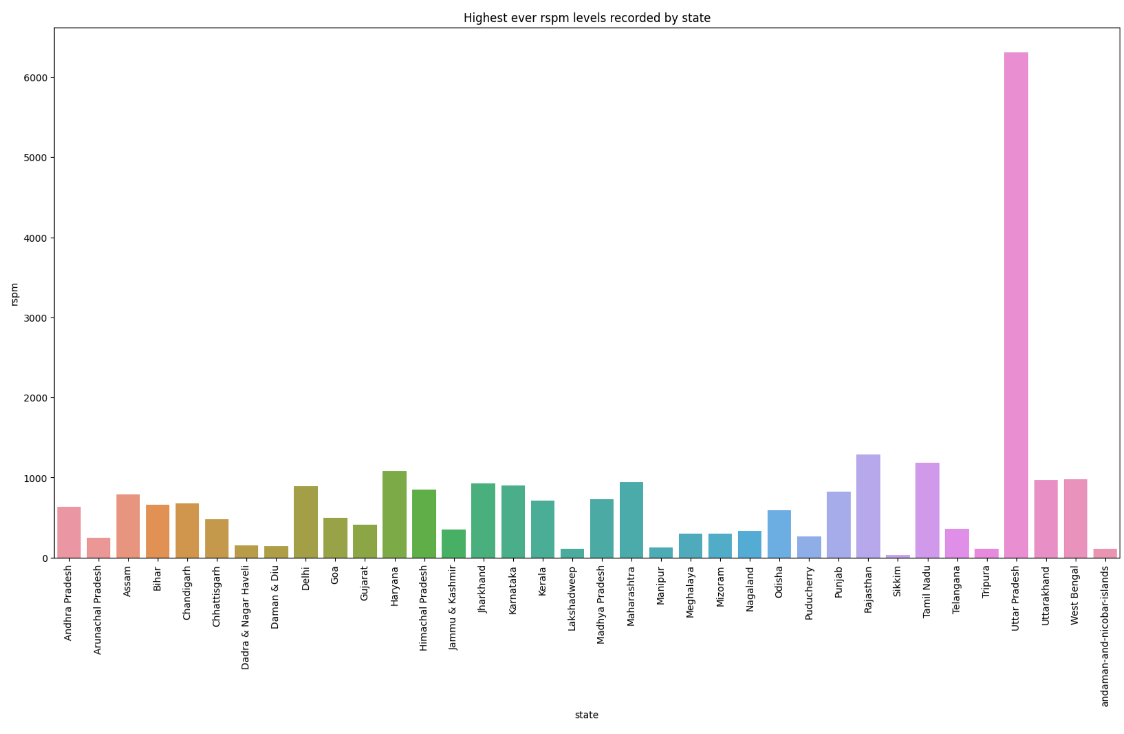
```
    ind = data[[indicator, 'state']].groupby('state', as_index=False).median().sort_values(by=indicator,ascending=False)
    top10 = sns.barplot(x='state', y=indicator, data=ind[:10], ax=ax[0], color='red')
    top10.set_title("Top 10 states by {} (1991-2016)".format(indicator))
    top10.set_ylabel("so2 (µg/m3)")
    top10.set_xlabel("State")
    bottom10 = sns.barplot(x='state', y=indicator, data=ind[-10:], ax=ax[1], color='green')
    bottom10.set_title("Bottom 10 states by {} (1991-2016)".format(indicator))
    bottom10.set_ylabel("so2 (µg/m3)")
    bottom10.set_xlabel("State")

top_and_bottom_10_states("so2")
top_and_bottom_10_states("no2")
```





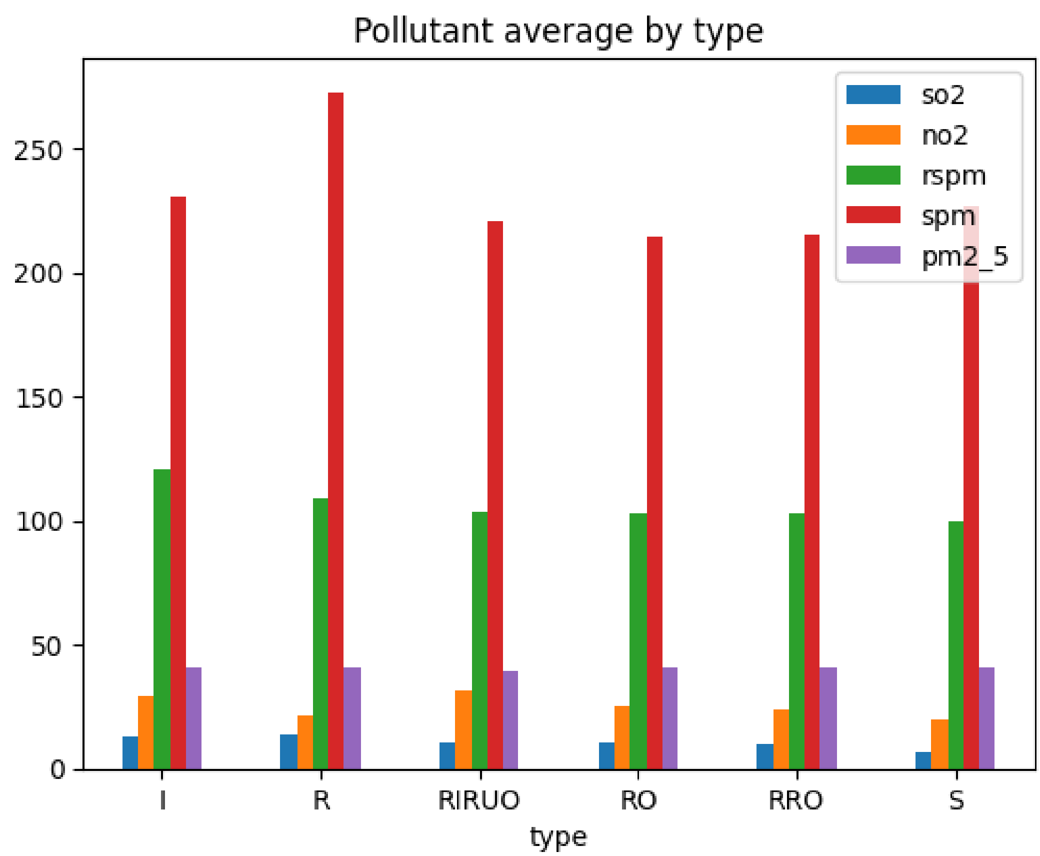
[49]:
```python
# Plotting the highest ever recorded levels
# defining a function to find the highest ever recorded levels for a given
↪indicator (defaults to SO2) by state
# sidenote: mostly outliers
def highest_levels_recorded(indicator="so2"):
    plt.figure(figsize=(20,10))
    ind = data[[indicator, 'location', 'state', 'date']].groupby('state',
↪as_index=False).max()
    highest = sns.barplot(x='state', y=indicator, data=ind)
    highest.set_title("Highest ever {} levels recorded by state".
↪format(indicator))
    plt.xticks(rotation=90)

highest_levels_recorded("no2")
highest_levels_recorded("rspm")
```


Highest ever no2 levels recorded by state
no2
0
200
400
600
800
state
Andhra Pradesh
Arunachal Pradesh
Assam
Bihar
Chandigarh
Chhattisgarh
Dadra & Nagar Haveli
Daman & Diu
Delhi
Goa
Gujarat
Haryana
Himachal Pradesh
Jammu & Kashmir
Jharkhand
Karnataka
Kerala
Lakshadweep
Madhya Pradesh
Maharashtra
Manipur
Meghalaya
Mizoram
Nagaland
Odisha
Puducherry
Punjab
Rajasthan
Sikkim
Tamil Nadu
Telangana
Tripura
Uttar Pradesh
Uttarakhand
West Bengal
andaman-and-nicobar-islands


Highest ever rspm levels recorded by state
rspm
0
1000
2000
3000
4000
5000
6000
state
Andhra Pradesh
Arunachal Pradesh
Assam
Bihar
Chandigarh
Chhattisgarh
Dadra & Nagar Haveli
Daman & Diu
Delhi
Goa
Gujarat
Haryana
Himachal Pradesh
Jammu & Kashmir
Jharkhand
Karnataka
Kerala
Lakshadweep
Madhya Pradesh
Maharashtra
Manipur
Meghalaya
Mizoram
Nagaland
Odisha
Puducherry
Punjab
Rajasthan
Sikkim
Tamil Nadu
Telangana
Tripura
Uttar Pradesh
Uttarakhand
West Bengal
andaman-and-nicobar-islands

[52]:

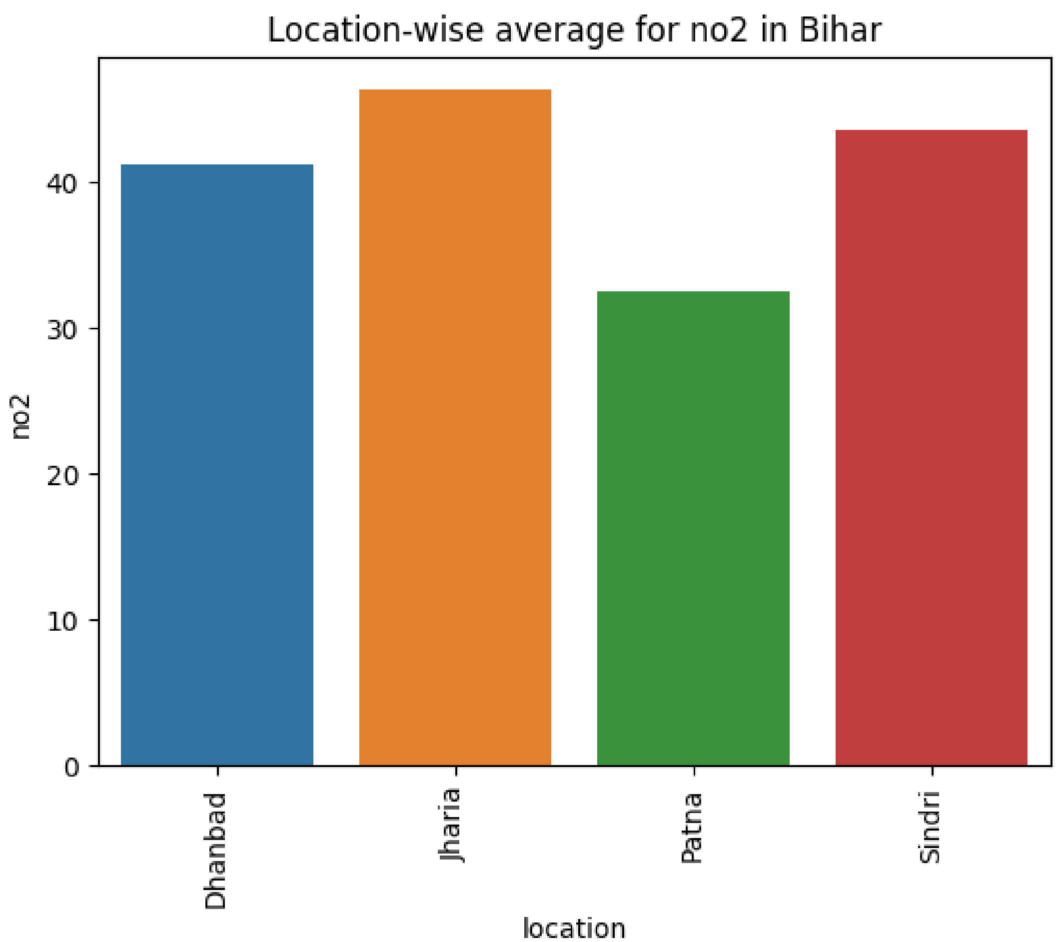
```python
# Plotting pollutant average by type
# defining a function to plot pollutant averages by type for a given indicator
def type_avg(indicator=""):
    type_avg = data[VALUE_COLS + ['type', 'date']].groupby("type").mean()
    if not indicator:
        t = type_avg[indicator].plot(kind='bar')
        plt.xticks(rotation = 0)
        plt.title("Pollutant average by type for {}".format(indicator))
    else:
        t = type_avg.plot(kind='bar')
        plt.xticks(rotation = 0)
        plt.title("Pollutant average by type")

type_avg('so2')
```



[53]:

```python
# Plotting pollutant averages by locations/state
# defining a function to plot pollutant averages for a given indicator (defaults␣
↪to SO2) by locations in a given state
def location_avgs(state, indicator="so2"):
```

```
    locs = data[VALUE_COLS + ['state', 'location', 'date']].groupby(['state',␣
↪'location']).mean()
    state_avgs = locs.loc[state].reset_index()
    sns.barplot(x='location', y=indicator, data=state_avgs)
    plt.title("Location-wise average for {} in {}".format(indicator, state))
    plt.xticks(rotation = 90)

location_avgs("Bihar", "no2")
```



[ ]:

```

```

[ ]:
```
Name : Thorave Avishkar Shrikrushna
Roll No: 65
```

6 Title : Analyzing Sales Performance by Region in a Retail Company

[2]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

[3]:
```
df = pd.read_csv("./datasets/customer_shopping_data.csv")
df.head()
```

[3]:
```
  invoice_no customer_id  gender  age  category  quantity    price  \
0    I138884     C241288  Female   28  Clothing         5  1500.40
1    I317333     C111565    Male   21     Shoes         3  1800.51
2    I127801     C266599    Male   20  Clothing         1   300.08
3    I173702     C988172  Female   66     Shoes         5  3000.85
4    I337046     C189076  Female   53     Books         4    60.60

  payment_method invoice_date   shopping_mall
0    Credit Card     5/8/2022          Kanyon
1     Debit Card   12/12/2021  Forum Istanbul
2           Cash    9/11/2021       Metrocity
3    Credit Card   16/05/2021    Metropol AVM
4           Cash   24/10/2021          Kanyon
```

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

[4]:
```
      invoice_no customer_id  gender  age         category  quantity    price  \
99452    I219422     C441542  Female   45         Souvenir         5    58.65
99453    I325143     C569580    Male   27  Food & Beverage         2    10.46
99454    I824010     C103292    Male   63  Food & Beverage         2    10.46
99455    I702964     C800631    Male   56       Technology         4  4200.00
99456    I232867     C273973  Female   36         Souvenir         3    35.19

      payment_method invoice_date     shopping_mall
99452    Credit Card   21/09/2022            Kanyon
99453           Cash   22/09/2021    Forum Istanbul
99454     Debit Card   28/03/2021         Metrocity
99455           Cash   16/03/2021      Istinye Park
99456    Credit Card   15/10/2022  Mall of Istanbul
```

[6]:
```
# To check the count of records grouped by region/branch of the mall
df.groupby("shopping_mall").count()
```

[6]:
```
                   invoice_no  customer_id  gender    age  category  quantity  \
shopping_mall
Cevahir AVM              4991         4991    4991   4991      4991      4991
Emaar Square Mall        4811         4811    4811   4811      4811      4811
Forum Istanbul           4947         4947    4947   4947      4947      4947
Istinye Park             9781         9781    9781   9781      9781      9781
Kanyon                  19823        19823   19823  19823     19823     19823
Mall of Istanbul        19943        19943   19943  19943     19943     19943
Metrocity               15011        15011   15011  15011     15011     15011
Metropol AVM            10161        10161   10161  10161     10161     10161
Viaport Outlet           4914         4914    4914   4914      4914      4914
Zorlu Center             5075         5075    5075   5075      5075      5075

                   price  payment_method  invoice_date
shopping_mall
Cevahir AVM         4991            4991          4991
Emaar Square Mall   4811            4811          4811
Forum Istanbul      4947            4947          4947
Istinye Park        9781            9781          9781
Kanyon             19823           19823         19823
Mall of Istanbul   19943           19943         19943
Metrocity          15011           15011         15011
Metropol AVM       10161           10161         10161
Viaport Outlet      4914            4914          4914
Zorlu Center        5075            5075          5075
```

[8]:
```python
# To check the count of records grouped by the product categories
df.groupby("category").count()
```

[8]:
```
                 invoice_no  customer_id  gender    age  quantity  price  \
category
Books                  4981         4981    4981   4981      4981   4981
Clothing              34487        34487   34487  34487     34487  34487
Cosmetics             15097        15097   15097  15097     15097  15097
Food & Beverage       14776        14776   14776  14776     14776  14776
Shoes                 10034        10034   10034  10034     10034  10034
Souvenir               4999         4999    4999   4999      4999   4999
Technology             4996         4996    4996   4996      4996   4996
Toys                  10087        10087   10087  10087     10087  10087

                 payment_method  invoice_date  shopping_mall
category
Books                      4981          4981           4981
Clothing                  34487         34487          34487
Cosmetics                 15097         15097          15097
Food & Beverage           14776         14776          14776
Shoes                     10034         10034          10034
```

```
Souvenir                    4999        4999          4999
Technology                  4996        4996          4996
Toys                       10087       10087         10087
```

[9]:
```
# total sales for each mall branch
branch_sales = df.groupby("shopping_mall").sum()
branch_sales
```

[9]:
```
                      age  quantity        price
shopping_mall
Cevahir AVM        215474     14949   3433671.84
Emaar Square Mall  209575     14501   3390408.31
Forum Istanbul     215380     14852   3336073.82
Istinye Park       424335     29465   6717077.54
Kanyon             862280     59457  13710755.24
Mall of Istanbul   866333     60114  13851737.62
Metrocity          652968     44894  10249980.07
Metropol AVM       439086     30530   6937992.99
Viaport Outlet     212771     14716   3414019.46
Zorlu Center       220926     15234   3509649.02
```

[11]:
```
# total sales for each category of product
category_sales = df.groupby("category").sum()
category_sales
```

[11]:
```
                     age  quantity        price
category
Books             216882     14982    226977.30
Clothing         1497054    103558  31075684.64
Cosmetics         657937     45465   1848606.90
Food & Beverage   640605     44277    231568.71
Shoes             436027     30217  18135336.89
Souvenir          216922     14871    174436.83
Technology        216669     15021  15772050.00
Toys              437032     30321   1086704.64
```

[12]:
```
# to get the top performing branches
branch_sales.sort_values(by = "price", ascending = False)
```

[12]:
```
                      age  quantity        price
shopping_mall
Mall of Istanbul   866333     60114  13851737.62
Kanyon             862280     59457  13710755.24
Metrocity          652968     44894  10249980.07
Metropol AVM       439086     30530   6937992.99
Istinye Park       424335     29465   6717077.54
Zorlu Center       220926     15234   3509649.02
```

```
Cevahir AVM        215474     14949   3433671.84
Viaport Outlet     212771     14716   3414019.46
Emaar Square Mall  209575     14501   3390408.31
Forum Istanbul     215380     14852   3336073.82
```

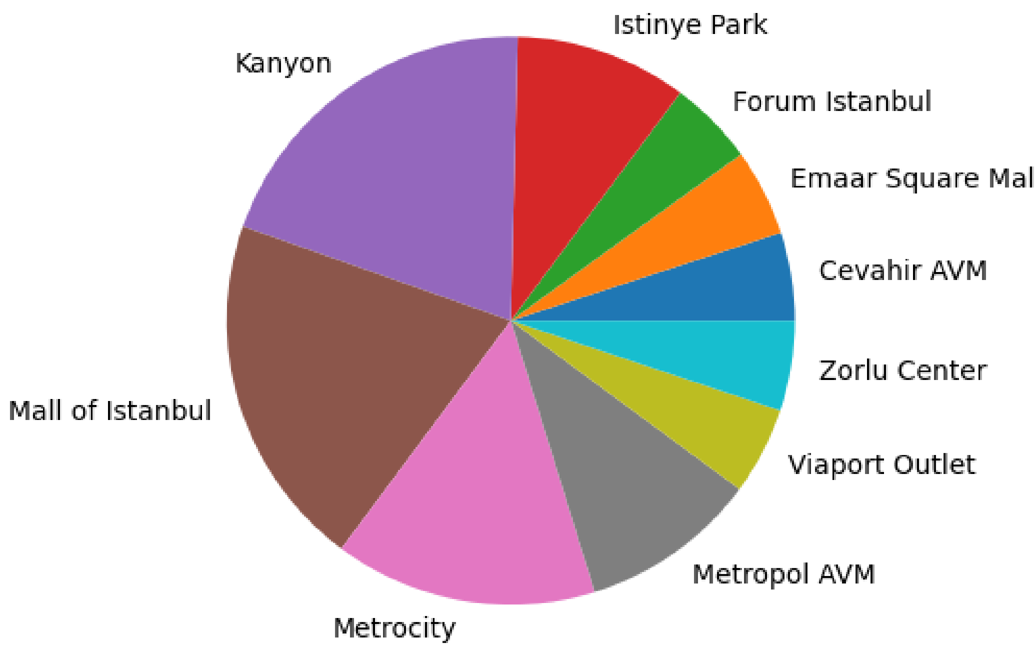
[13]:
```python
# to get the top selling categories
category_sales.sort_values(by = "price", ascending = False)
```

[13]:
```
                     age  quantity        price
category
Clothing         1497054    103558  31075684.64
Shoes             436027     30217  18135336.89
Technology        216669     15021  15772050.00
Cosmetics         657937     45465   1848606.90
Toys              437032     30321   1086704.64
Food & Beverage   640605     44277    231568.71
Books             216882     14982    226977.30
Souvenir          216922     14871    174436.83
```

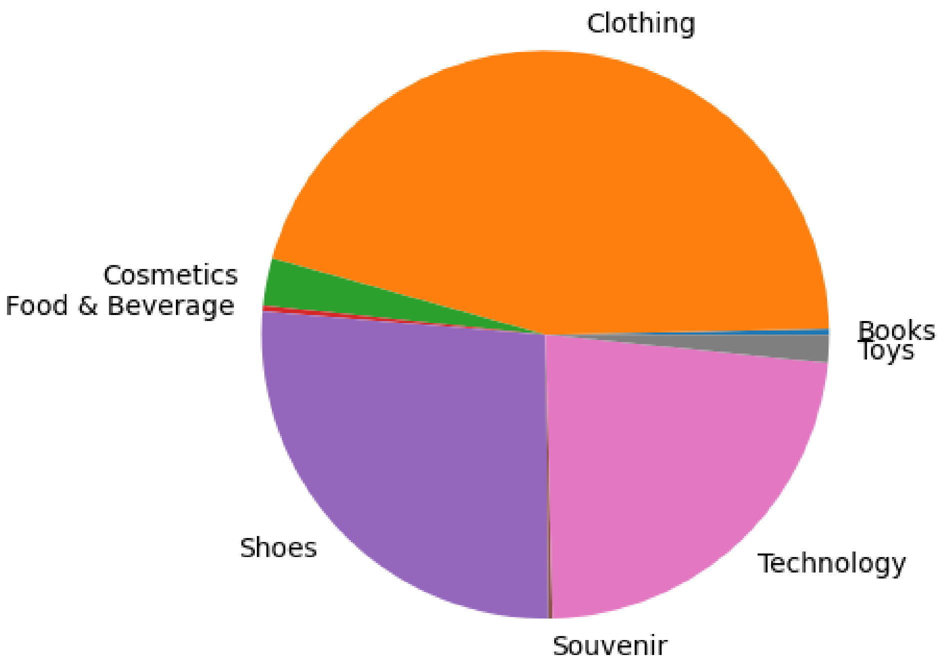
[15]:
```python
# to get total sales for each combination of branch and product_category
combined_branch_category_sales = df.groupby(["shopping_mall", "category"]).sum()
combined_branch_category_sales
```

[15]:
```
                                 age  quantity       price
shopping_mall category
Cevahir AVM   Books            11464       792    11998.80
              Clothing         74729      5180  1554414.40
              Cosmetics        31142      2174    88394.84
              Food & Beverage  33269      2293    11992.39
              Shoes            21211      1473   884050.41
...                              ...       ...         ...
Zorlu Center  Food & Beverage  32687      2216    11589.68
              Shoes            22949      1589   953670.13
              Souvenir         10727       716     8398.68
              Technology       10533       765   803250.00
              Toys             22395      1526    54691.84

[80 rows x 3 columns]
```

[16]:
```python
# pie chart for sales by branch
plt.pie(branch_sales["price"], labels = branch_sales.index)
plt.show()
```

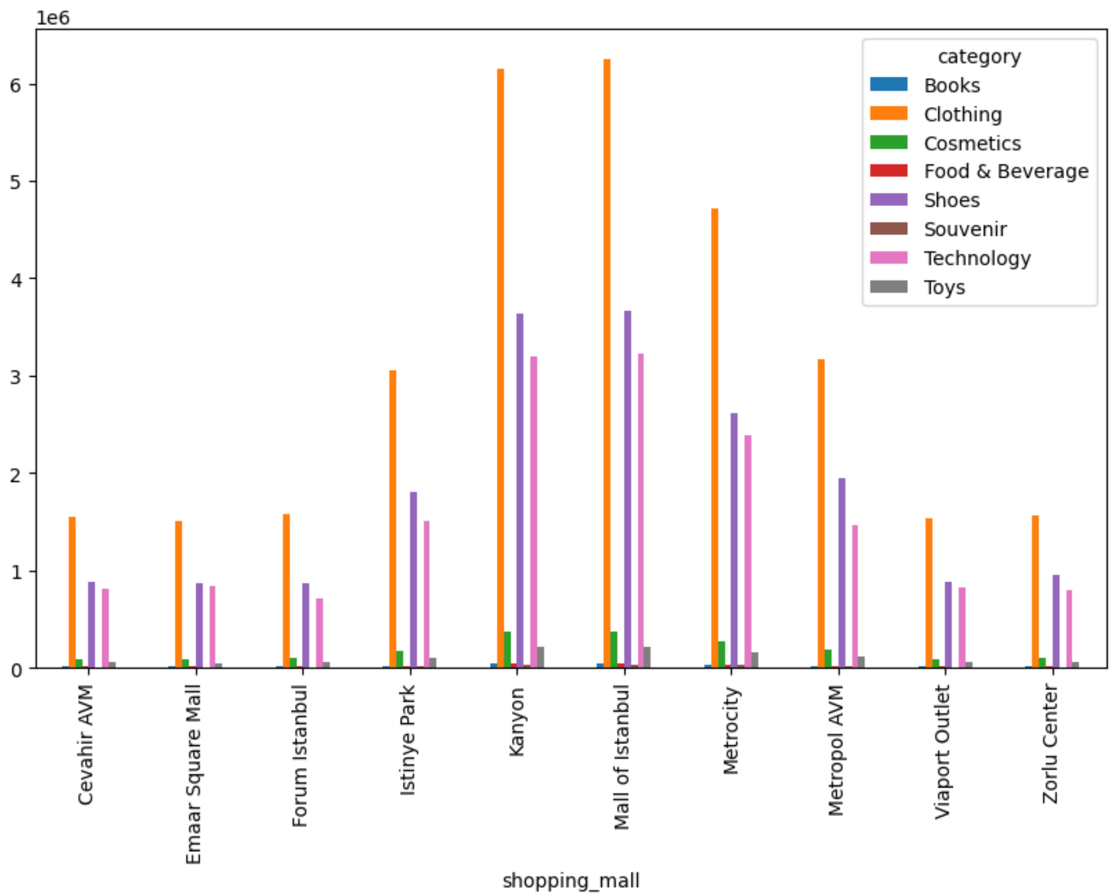


[17]:
```
# pie chart for sales by product category
plt.pie(category_sales["price"], labels = category_sales.index)
plt.show()
```



[19]:
```
combined_pivot = df.pivot_table(index="shopping_mall", columns="category",␣
 ↪values="price", aggfunc="sum")
```

[20]:
```
# grouped bar chart for sales of different categories at different branches
combined_pivot.plot(kind="bar", figsize=(10, 6))
plt.show()
```


1e6
6
5
4
3
2
1
0
category
Books
Clothing
Cosmetics
Food & Beverage
Shoes
Souvenir
Technology
Toys
Cevahir AVM
Emaar Square Mall
Forum Istanbul
Istinye Park
Kanyon
Mall of Istanbul
Metrocity
Metropol AVM
Viaport Outlet
Zorlu Center
shopping_mall

[ ]:

[ ]:
```
Name: Thorave Avishkar Shrikrushna
Roll No: 65
```

7 Analysis and Visualization of Stock Market Data

[45]:
```python
# import necessary libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import datetime
import warnings
warnings.filterwarnings('ignore')
```

[46]:
```python
# read data from csv file

HDFC_df = pd.read_csv("HDFC.csv")
HDFC_df.head()
```

[46]:
```
         Date         Open         High          Low        Close  \
0  2018-02-15  1828.900024  1851.000000  1819.150024  1829.500000
1  2018-02-16  1835.500000  1836.949951  1804.199951  1815.500000
2  2018-02-19  1827.750000  1830.199951  1801.000000  1814.050049
3  2018-02-20  1832.900024  1840.000000  1802.500000  1811.750000
4  2018-02-21  1825.000000  1832.699951  1816.000000  1824.800049

     Adj Close     Volume
0  1780.624512  3382968.0
1  1766.998535  2368880.0
2  1765.587524  1603737.0
3  1763.348633  2523482.0
4  1776.050171  3795216.0
```

[47]:
```python
# round-off some values

HDFC_df = HDFC_df.round(2)
HDFC_df.head(2)
```

[47]:
```
         Date    Open     High      Low   Close  Adj Close     Volume
0  2018-02-15  1828.9  1851.00  1819.15  1829.5    1780.62  3382968.0
1  2018-02-16  1835.5  1836.95  1804.20  1815.5    1767.00  2368880.0
```

[4]:
```python
# shape of dataframe

HDFC_df.shape
```

[4]: (491, 7)

[49]:
```
# columns of dataframe

HDFC_df.columns
```

[49]:
```
Index(['Date', 'Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume'],
dtype='object')
```

[48]:
```
# determining null entries

HDFC_df.isnull().sum()
```

[48]:
```
Date         0
Open         1
High         1
Low          1
Close        1
Adj Close    1
Volume       1
dtype: int64
```

[50]:
```
# check the null entry

HDFC_df[HDFC_df.Open.isnull()]
```

[50]:

	Date	Open	High	Low	Close	Adj Close	Volume
413	2019-10-27	NaN	NaN	NaN	NaN	NaN	NaN

[51]:
```
# drop null values

HDFC_df.dropna(inplace = True, axis = 0)
```

[52]:
```
# check the datatype

HDFC_df.dtypes
```

[52]:
```
Date          object
Open         float64
High         float64
Low          float64
Close        float64
Adj Close    float64
Volume       float64
dtype: object
```

[53]:
```
# Convert the date column type to datetime
```

```
HDFC_df['Date'] = pd.to_datetime(HDFC_df['Date'])
HDFC_df.head(2)
```

[53]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2018-02-15	1828.9	1851.00	1819.15	1829.5	1780.62	3382968.0
1	2018-02-16	1835.5	1836.95	1804.20	1815.5	1767.00	2368880.0

[54]:
```
# total number of days under consideration

HDFC_df['Date'].max() - HDFC_df['Date'].min()
```

[54]: Timedelta('729 days 00:00:00')

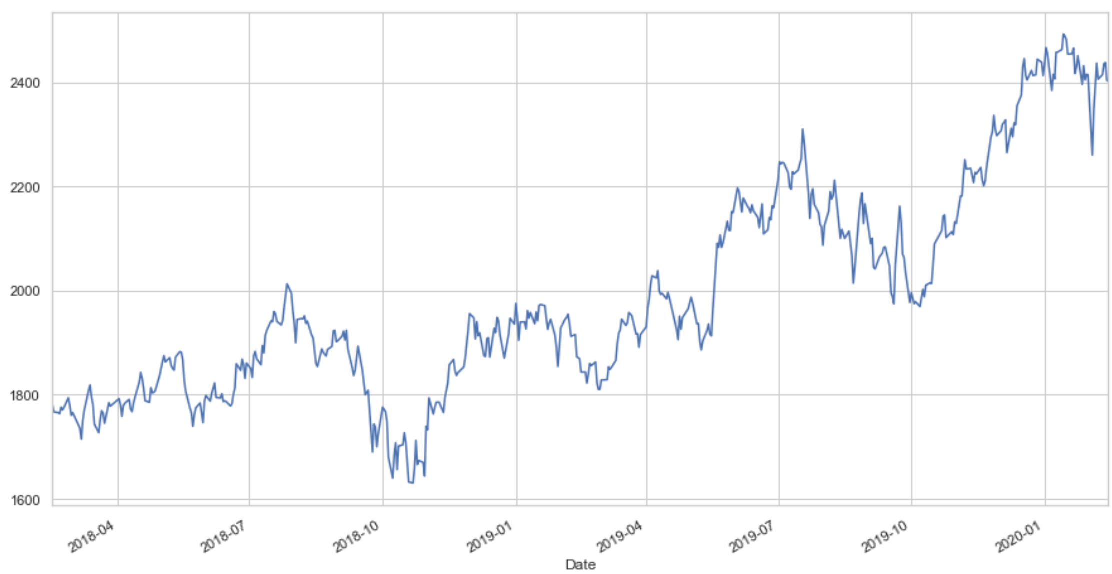
[55]:
```
# general stats for last 90 days

HDFC_df.iloc[-90:].describe().astype(int)
```

[55]:

	Open	High	Low	Close	Adj Close	Volume
count	90	90	90	90	90	90
mean	2302	2325	2281	2307	2307	3164814
std	142	143	142	142	142	1351297
min	1974	1996	1963	1969	1969	945874
25%	2209	2232	2194	2214	2214	2263895
50%	2323	2348	2301	2331	2331	2776124
75%	2425	2444	2404	2421	2421	3497831
max	2486	2499	2471	2492	2492	8808006

[56]:
```
# Set the Date columns as index of the dataframe for further analysis

HDFC_df.index = HDFC_df['Date']
```

[57]:
```
# observe general price variation of the closing price

sns.set_style('whitegrid')
HDFC_df['Adj Close'].plot(figsize = (15,8))
plt.show()
```



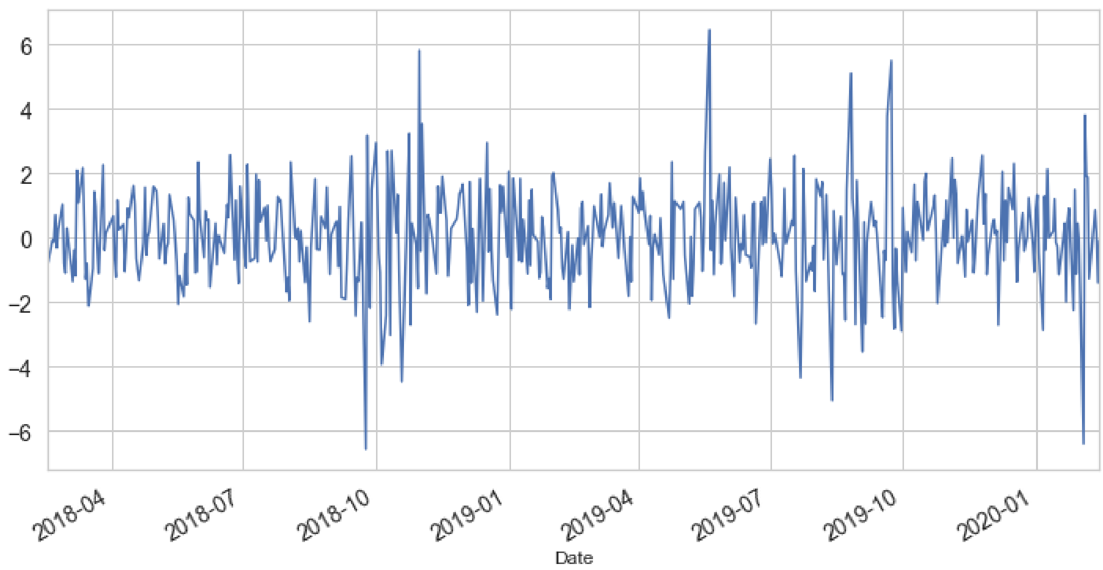
[59]:
```
# Add a new column 'Day_Perc_Change' which give the daily returns

HDFC_df['Day_Perc_Change'] = HDFC_df['Adj Close'].pct_change()*100
```

[60]:
```
# Replace NaN with 0

HDFC_df['Day_Perc_Change'] = HDFC_df['Day_Perc_Change'].fillna(0)
HDFC_df.head()
```

[60]:
```
                  Date     Open     High      Low    Close  Adj Close  \
Date
2018-02-15  2018-02-15  1828.90  1851.00  1819.15  1829.50    1780.62
2018-02-16  2018-02-16  1835.50  1836.95  1804.20  1815.50    1767.00
2018-02-19  2018-02-19  1827.75  1830.20  1801.00  1814.05    1765.59
2018-02-20  2018-02-20  1832.90  1840.00  1802.50  1811.75    1763.35
2018-02-21  2018-02-21  1825.00  1832.70  1816.00  1824.80    1776.05

               Volume  Day_Perc_Change
Date
2018-02-15  3382968.0         0.000000
2018-02-16  2368880.0        -0.764902
2018-02-19  1603737.0        -0.079796
2018-02-20  2523482.0        -0.126870
2018-02-21  3795216.0         0.720220
```

[61]:
```
# daily returns(day-to-day percentage change) plot

HDFC_df['Day_Perc_Change'].plot(figsize = (12, 6), fontsize = 14)
```

[61]: <matplotlib.axes._subplots.AxesSubplot at 0x14e33df0>



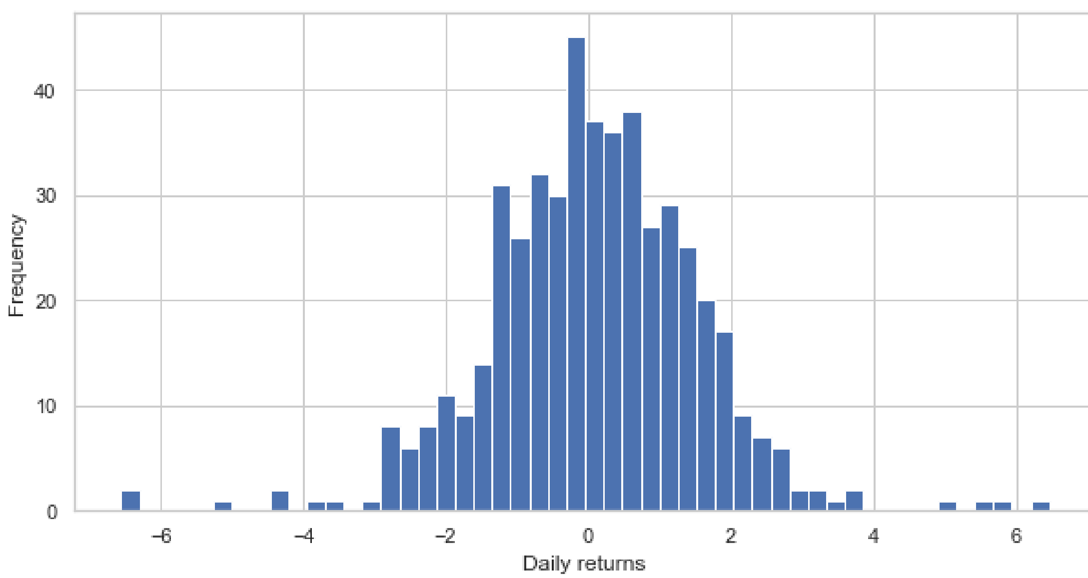
[62]:

```
# Add a new column trend whose values are determined by the below relationship

def trend(x):
    if x > -0.5 and x <= 0.5:
        return 'Slight or No change'
    elif x > 0.5 and x <= 1:
        return 'Slight Positive'
    elif x > -1 and x <= -0.5:
        return 'Slight Negative'
    elif x > 1 and x <= 3:
        return 'Positive'
    elif x > -3 and x <= -1:
        return 'Negative'
    elif x > 3 and x <= 7:
        return 'Among top gainers'
    elif x > -7 and x <= -3:
        return 'Among top losers'
    elif x > 7:
        return 'Bull run'
    elif x <= -7:
        return 'Bear drop'

HDFC_df['Trend'] = np.zeros(HDFC_df['Day_Perc_Change'].count())
HDFC_df['Trend'] = HDFC_df['Day_Perc_Change'].apply(lambda x: trend(x))
```

[64]:
```
# display first few entires

HDFC_df.head()
```

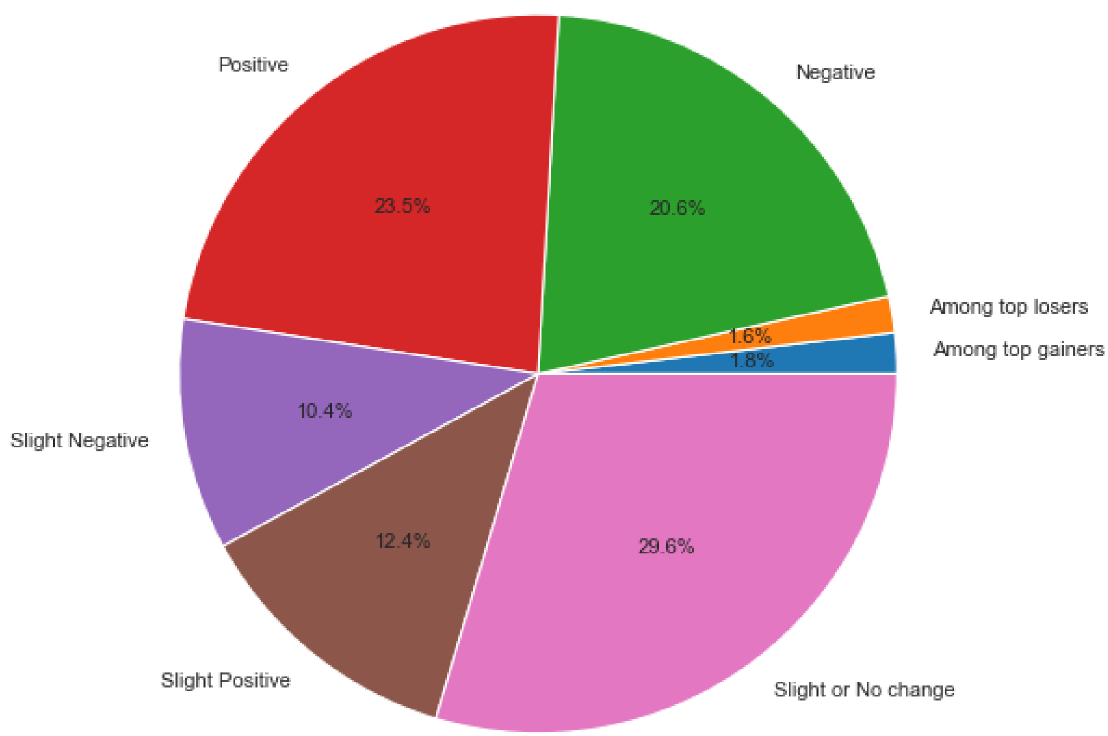
[64]:
```
                  Date     Open     High      Low    Close  Adj Close  \
Date
2018-02-15  2018-02-15  1828.90  1851.00  1819.15  1829.50    1780.62
2018-02-16  2018-02-16  1835.50  1836.95  1804.20  1815.50    1767.00
2018-02-19  2018-02-19  1827.75  1830.20  1801.00  1814.05    1765.59
2018-02-20  2018-02-20  1832.90  1840.00  1802.50  1811.75    1763.35
2018-02-21  2018-02-21  1825.00  1832.70  1816.00  1824.80    1776.05

               Volume  Day_Perc_Change                Trend
Date
2018-02-15  3382968.0         0.000000  Slight or No change
2018-02-16  2368880.0        -0.764902      Slight Negative
2018-02-19  1603737.0        -0.079796  Slight or No change
2018-02-20  2523482.0        -0.126870  Slight or No change
2018-02-21  3795216.0         0.720220      Slight Positive
```

[65]:
```
# Daily returns histogram

HDFC_df['Day_Perc_Change'].hist(bins = 50, figsize = (10,5))
plt.xlabel('Daily returns')
plt.ylabel('Frequency')
plt.show()

# satistics
HDFC_df.Day_Perc_Change.describe()
```



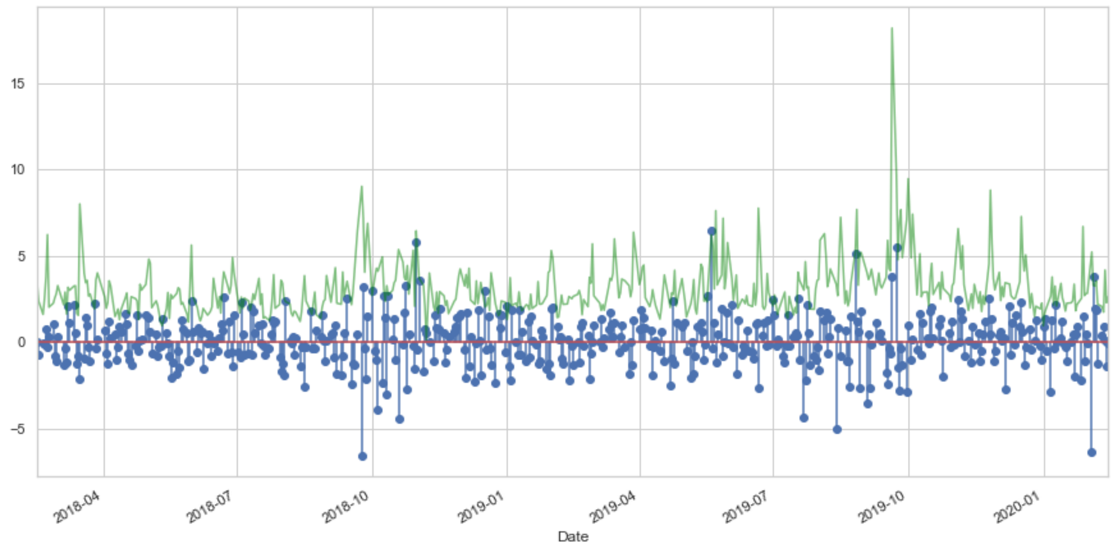
```
[65]: count    490.000000
      mean       0.072191
      std        1.491135
      min       -6.561574
      25%       -0.804370
      50%        0.056327
      75%        1.009923
      max        6.463177
      Name: Day_Perc_Change, dtype: float64
```

```
[26]: # pie-chart of the trend

      fig = plt.figure(dpi = 80)
      HDFC_pie_data = HDFC_df.groupby('Trend')
      pie_label = sorted([i for i in HDFC_df.loc[:, 'Trend'].unique()])
      plt.pie(HDFC_pie_data['Trend'].count(), labels = pie_label,
              autopct = '%1.1f%%', radius = 2)
      #plt.label(size=8, weight="bold")
      plt.show()
```



[67]:
```
# Superimpose the daily volume plot upon the daily percentage change stem plot

plt.stem(HDFC_df['Date'], HDFC_df['Day_Perc_Change'])
(HDFC_df['Volume']/1000000).plot(figsize = (15, 7.5), color = 'green', alpha = 0.
↪5)
```

[67]: <matplotlib.axes._subplots.AxesSubplot at 0x1362e4b0>



[68]:
```
# import multiple stocks together: HDFC, Jindal Steel, Jubilant Foods, Sun␣
↪Pharma, TCS along with market index.

import pandas_datareader.data as web
start = datetime.datetime(2018, 2, 15)
end = datetime.datetime(2020, 2, 14)
combined_df = web.DataReader(['HDFC.NS', 'JINDALSTEL.NS', 'JUBLFOOD.NS',
                              'SUNPHARMA.NS', 'TCS.NS', '^NSEI'],
                              'yahoo', start = start, end = end)['Adj Close']
combined_df.head()
```

[68]:
```
Symbols         HDFC.NS  JINDALSTEL.NS  JUBLFOOD.NS  SUNPHARMA.NS  \
Date
2018-02-15  1780.624512     265.350006   996.540466    566.225708
2018-02-16  1766.998535     251.500000   966.697205    565.734009
2018-02-19  1765.587524     250.000000   987.599854    552.113159
2018-02-20  1763.348633     252.000000   989.259155    550.244568
2018-02-21  1776.050171     247.300003   985.841492    517.052856

Symbols          TCS.NS         ^NSEI
```

```
Date
2018-02-15  1381.652588  10545.500000
2018-02-16  1385.052368  10452.299805
2018-02-19  1380.585327  10378.400391
2018-02-20  1390.577393  10360.400391
2018-02-21  1436.637939  10397.450195
```

[4]:
```
# check for null values

combined_df.isnull().sum()
```

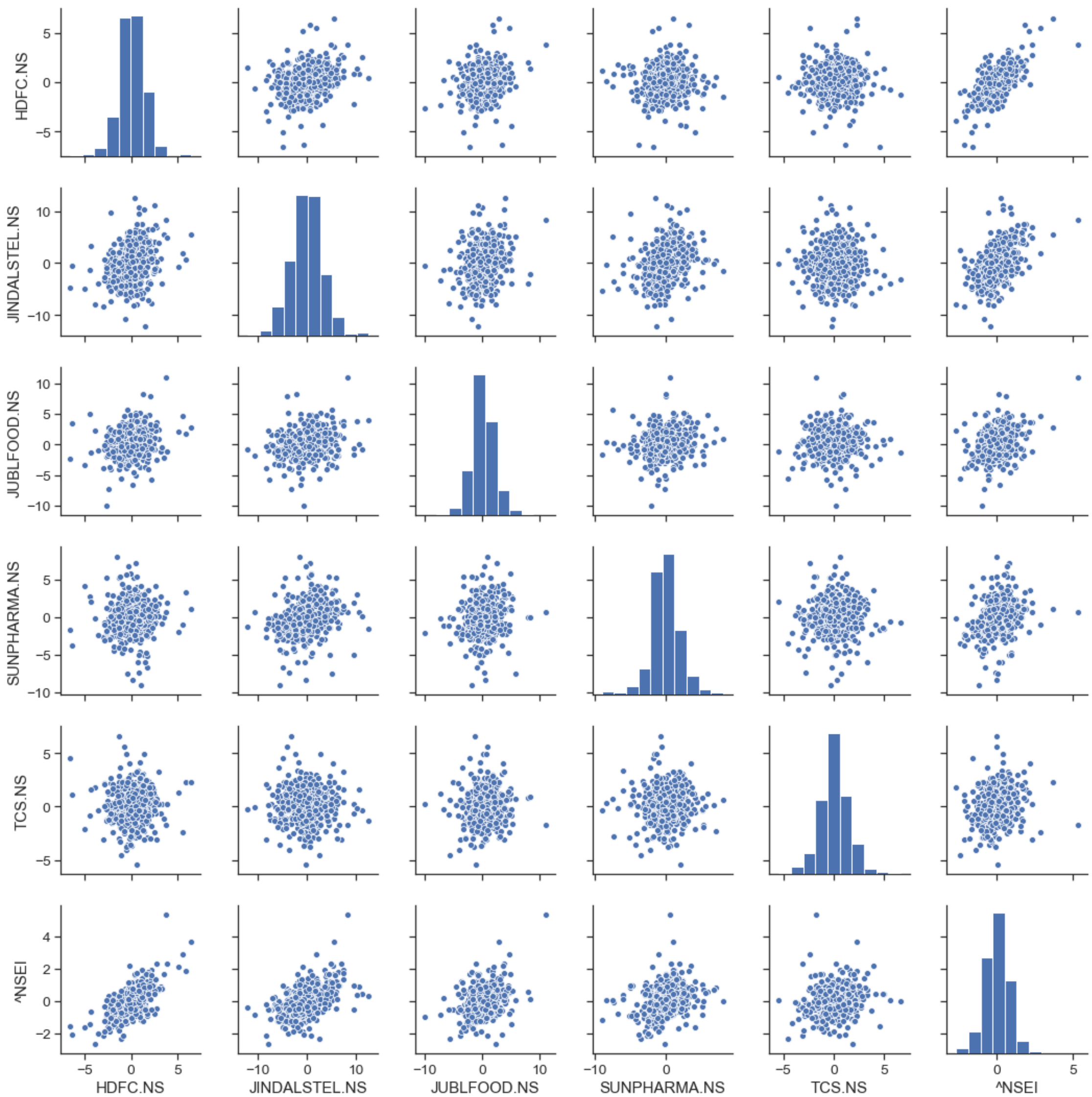
[4]:
```
Symbols
HDFC.NS          0
JINDALSTEL.NS    0
JUBLFOOD.NS      0
SUNPHARMA.NS     0
TCS.NS           0
^NSEI            1
dtype: int64
```

[70]:
```
# drop null values

combined_df.dropna(inplace = True, axis = 0)
combined_df.isnull().sum()
```

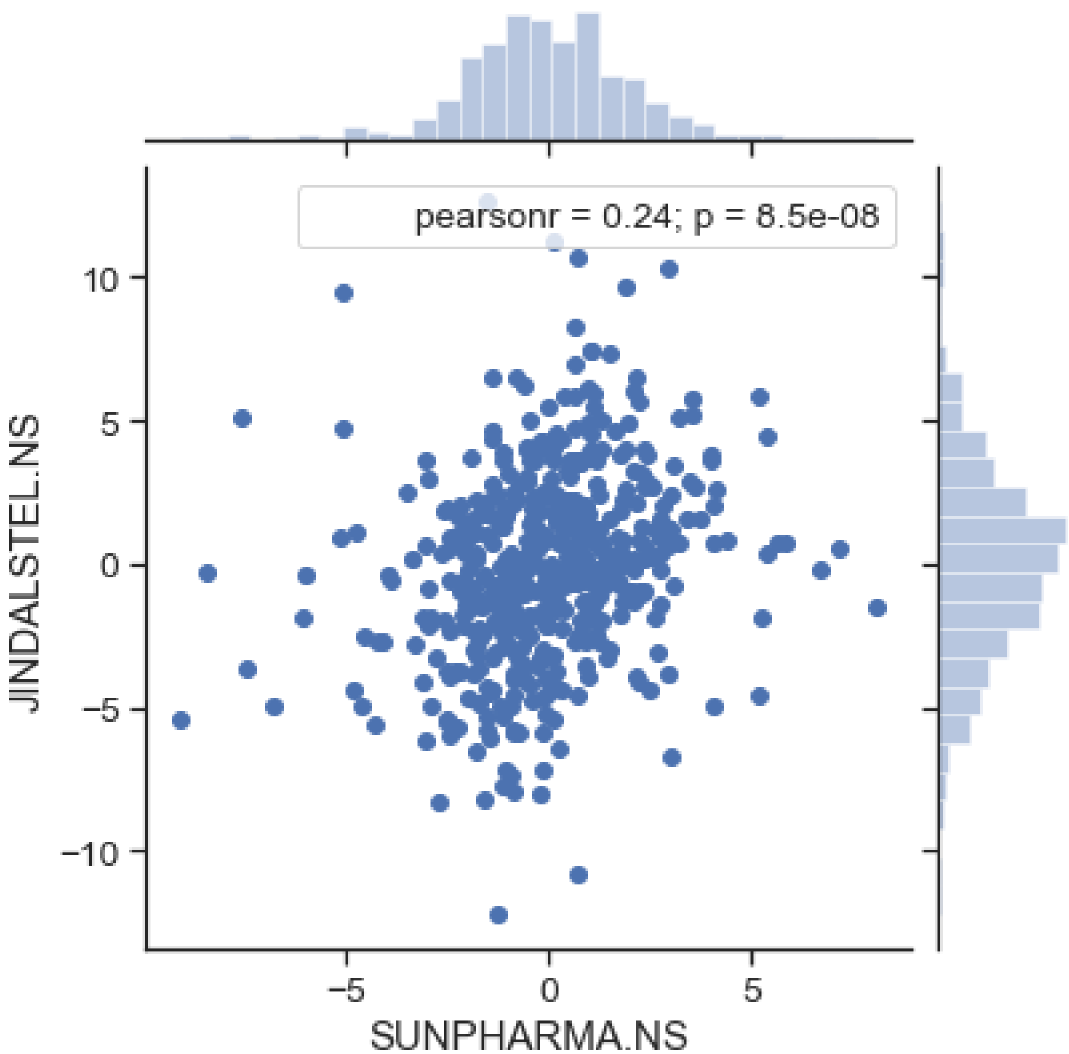
[70]:
```
Symbols
HDFC.NS          0
JINDALSTEL.NS    0
JUBLFOOD.NS      0
SUNPHARMA.NS     0
TCS.NS           0
^NSEI            0
dtype: int64
```

[71]:
```
# plot the pair plot of daily percentage of the close price (or daily returns)␣
↪for all stocks

pct_chg_df = combined_df.pct_change()*100
pct_chg_df.dropna(inplace = True, how = 'any', axis = 0)

import seaborn as sns
sns.set(style = 'ticks', font_scale = 1.25)
sns.pairplot(pct_chg_df)
```

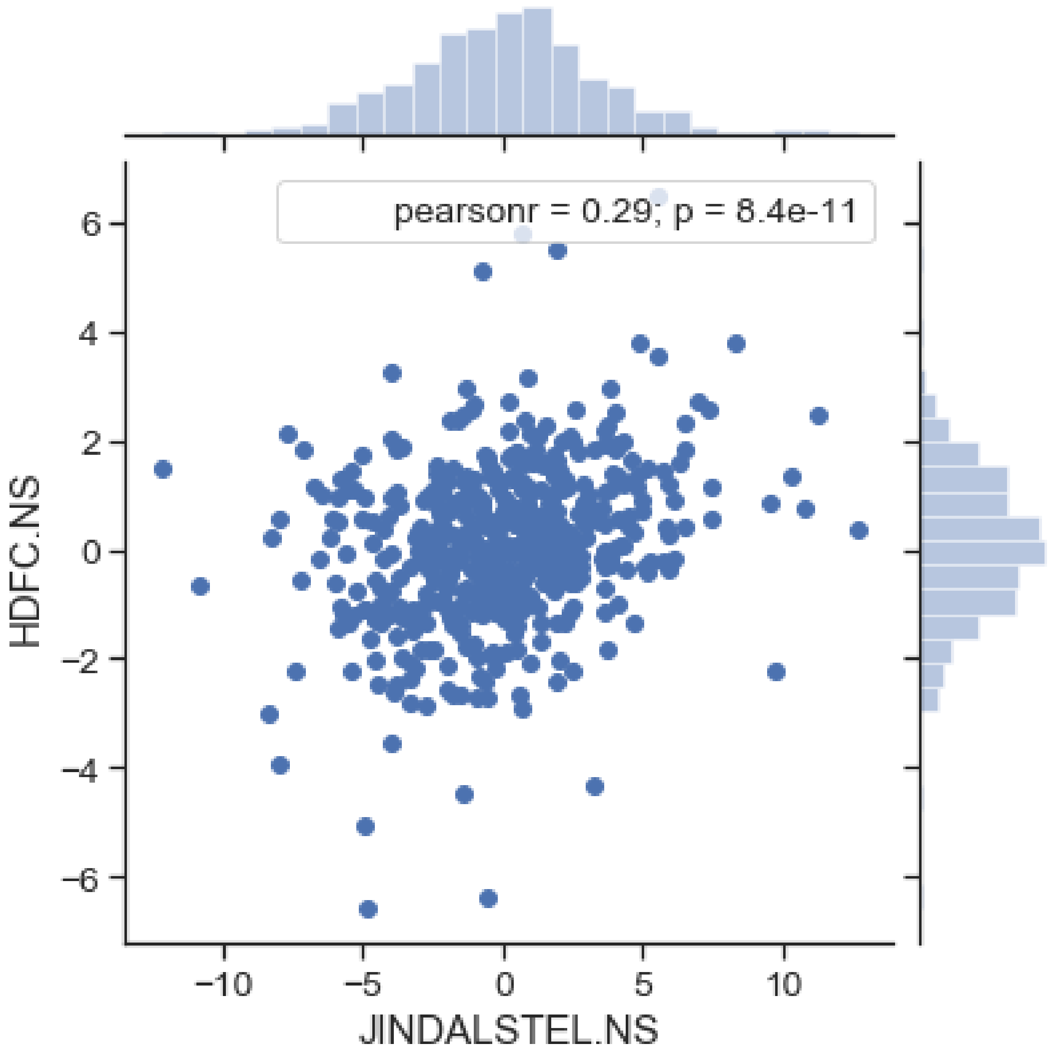
[71]:
```
<seaborn.axisgrid.PairGrid at 0x14e4f610>
```



[72]:
```
# plot join plots for Sun Pharma v/s Jindal Steel and Jindal Steel v/s HDFC

from scipy.stats import stats

sns.jointplot('SUNPHARMA.NS', 'JINDALSTEL.NS', pct_chg_df, kind = 'scatter').
 ↪annotate(stats.pearsonr)
sns.jointplot('JINDALSTEL.NS', 'HDFC.NS', pct_chg_df, kind = 'scatter').
 ↪annotate(stats.pearsonr)
plt.show()
```

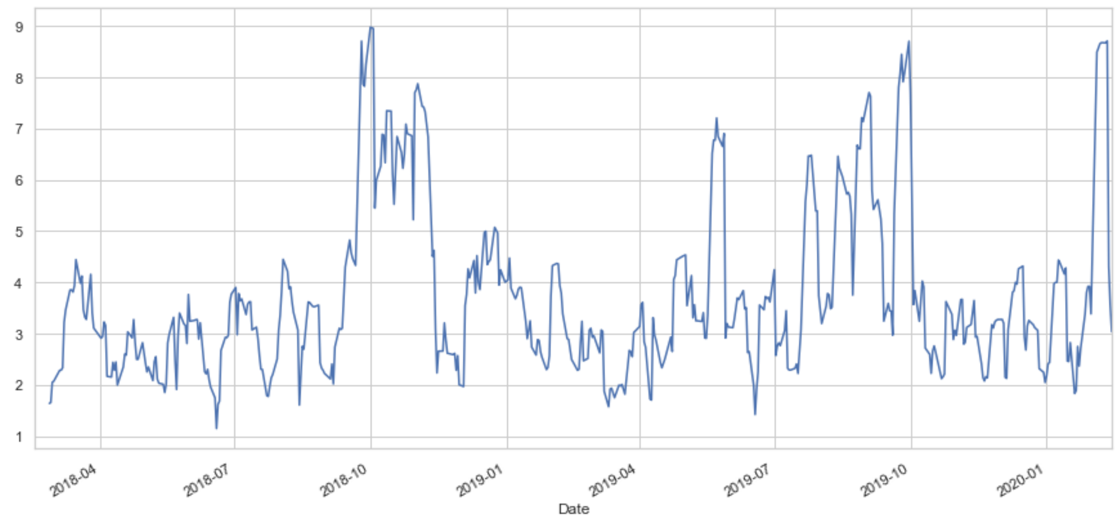


pearsonr = 0.24; p = 8.5e-08
10
5
0
−5
−10
JINDALSTEL.NS
−5
0
5
SUNPHARMA.NS



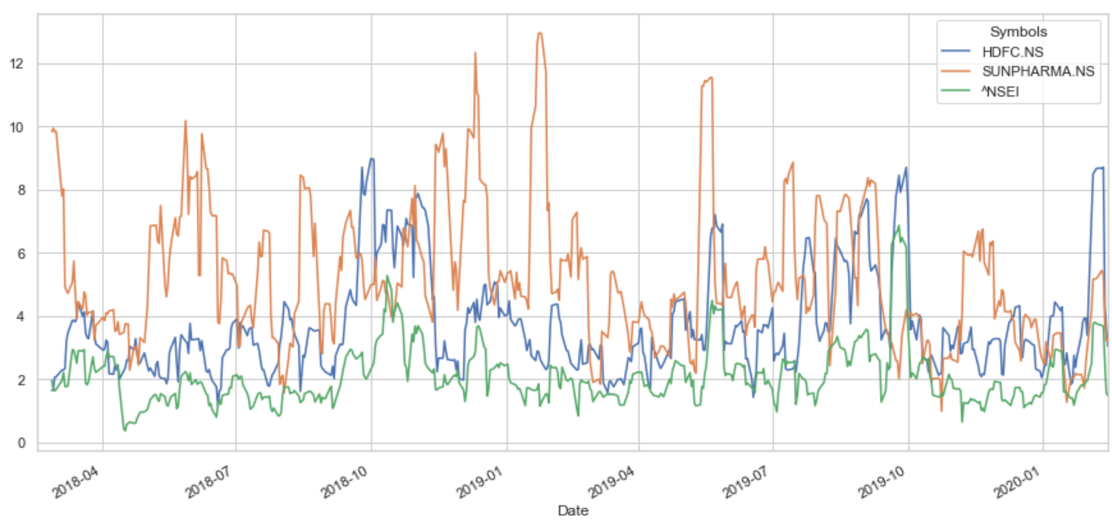
[73]:
```
# Determining and plotting volatility(standard deviation) for HDFC stock by␣
↪taking 7-day rolling window

sns.set(style = 'whitegrid')
HDFC_vol = pct_chg_df['HDFC.NS'].rolling(7).std()*np.sqrt(7)
HDFC_vol.plot(figsize = (15,7))
```

[73]: <matplotlib.axes._subplots.AxesSubplot at 0x1681b550>



[43]:
```
# volatility plot of HDFC, Sun Pharma and Nifty index

volatiliy = pct_chg_df[['HDFC.NS', 'SUNPHARMA.NS', '^NSEI']].rolling(7).std()*np.
 ↪sqrt(7)
volatiliy.plot(figsize = (15, 7))
```

[43]: <matplotlib.axes._subplots.AxesSubplot at 0x1333a0b0>



[ ]: