```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder, StandardScaler,
RobustScaler
from sklearn.linear_model import LinearRegression,Ridge,Lasso
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import
AdaBoostRegressor,RandomForestRegressor,GradientBoostingRegressor
from sklearn.metrics import
r2_score,mean_absolute_error,mean_squared_error,mean_squared_log_error
from sklearn.neighbors import
KNeighborsRegressor,RadiusNeighborsRegressor,NearestCentroid
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings('ignore')
```

Loading the Dataset

```
df=pd.read_csv('./data.csv')

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21 entries, 0 to 20
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Date            21 non-null     object
 1   series          21 non-null     object
 2   OPEN            21 non-null     object
 3   HIGH            21 non-null     object
 4   LOW             21 non-null     object
 5   PREV. CLOSE     21 non-null     object
 6   ltp             21 non-null     object
 7   close           21 non-null     object
 8   vwap            21 non-null     object
 9   52W H           21 non-null     object
 10  52W L           21 non-null     object
 11  VOLUME          21 non-null     object
 12  VALUE           21 non-null     object
 13  No of trades    21 non-null     object
dtypes: object(14)
memory usage: 2.4+ KB
```

```
df.columns
```

```
Index(['Date ', 'series ', 'OPEN ', 'HIGH ', 'LOW ', 'PREV. CLOSE ',
'ltp ',
```

```
       'close ', 'vwap ', '52W H ', '52W L ', 'VOLUME ', 'VALUE ',
       'No of trades '],
      dtype='object')
```

```
df.rename(columns={'Date ': 'Date'},inplace=True)
df.rename(columns={'series ':'series'},inplace=True)
df.rename(columns={'OPEN ':'OPEN'},inplace=True)
df.rename(columns={'HIGH ':'HIGH'},inplace=True)
df.rename(columns={'LOW ':'LOW'},inplace=True)
df.rename(columns={'PREV. CLOSE ':'PREV_CLOSE'},inplace=True)
df.rename(columns={'ltp ':'Last_Traded_Price'},inplace=True)
df.rename(columns={'close ':'close'},inplace=True)
df.rename(columns={'vwap ':'Volume_weighted_avg_price'},inplace=True)
df.rename(columns={'52W H ':'52W_H'},inplace=True)
df.rename(columns={'52W L ':'52W_L'},inplace=True)
df.rename(columns={'VOLUME ':'VOLUME'},inplace=True)
df.rename(columns={'VALUE ':'VALUE'},inplace=True)
df.rename(columns={'No of trades ':'No_of_trades'},inplace=True)

df
```

```
           Date series      OPEN      HIGH       LOW PREV_CLOSE  \
0   06-Aug-2024     EQ  3,084.00  3,142.80  3,057.25   3,038.20
1   05-Aug-2024     EQ  3,085.00  3,133.05  2,996.30   3,160.90
2   02-Aug-2024     EQ  3,200.00  3,215.00  3,111.00   3,217.25
3   01-Aug-2024     EQ  3,180.00  3,258.00  3,151.70   3,169.40
4   31-Jul-2024     EQ  3,141.00  3,197.50  3,135.00   3,128.75
5   30-Jul-2024     EQ  3,092.00  3,155.00  3,066.65   3,089.35
6   29-Jul-2024     EQ  3,094.30  3,120.35  3,074.00   3,080.50
7   26-Jul-2024     EQ  2,995.00  3,109.00  2,988.10   2,973.50
8   25-Jul-2024     EQ  2,960.70  3,017.65  2,945.10   2,970.70
9   24-Jul-2024     EQ  2,995.35  3,007.50  2,959.30   2,995.35
10  23-Jul-2024     EQ  3,020.00  3,038.00  2,886.35   3,000.85
11  22-Jul-2024     EQ  3,005.70  3,026.90  2,972.15   3,005.70
12  19-Jul-2024     EQ  3,092.00  3,094.50  3,000.00   3,092.20
13  18-Jul-2024     EQ  3,106.00  3,111.00  3,057.15   3,109.30
14  16-Jul-2024     EQ  3,110.00  3,137.75  3,096.10   3,090.40
15  15-Jul-2024     EQ  3,066.10  3,103.50  3,058.35   3,065.45
16  12-Jul-2024     EQ  3,090.00  3,098.80  3,058.35   3,078.30
17  11-Jul-2024     EQ  3,118.70  3,129.80  3,074.10   3,096.00
18  10-Jul-2024     EQ  3,120.10  3,127.25  3,063.40   3,110.75
19  09-Jul-2024     EQ  3,115.95  3,158.00  3,100.50   3,113.60
20  08-Jul-2024     EQ  3,147.90  3,158.20  3,075.00   3,147.90

   Last_Traded_Price     close Volume_weighted_avg_price     52W_H
52W_L  \
0           3,078.25  3,072.70                   3,108.00  3,743.90
2,142.00
1           3,024.00  3,038.20                   3,062.45  3,743.90
2,142.00
```

```
2           3,161.40  3,160.90                  3,174.06  3,743.90
2,142.00
3           3,225.10  3,217.25                  3,212.46  3,743.90
2,142.00
4           3,168.00  3,169.40                  3,171.25  3,743.90
2,142.00
5           3,133.00  3,128.75                  3,120.89  3,743.90
2,142.00
6           3,088.40  3,089.35                  3,097.85  3,743.90
2,142.00
7           3,072.25  3,080.50                  3,066.52  3,743.90
2,142.00
8           2,995.00  2,973.50                  2,980.02  3,743.90
2,142.00
9           2,968.80  2,970.70                  2,980.98  3,743.90
2,142.00
10          2,999.40  2,995.35                  2,989.95  3,743.90
2,142.00
11          3,000.00  3,000.85                  3,002.66  3,743.90
2,142.00
12          3,007.95  3,005.70                  3,033.24  3,743.90
2,142.00
13          3,090.00  3,092.20                  3,083.68  3,743.90
2,142.00
14          3,106.50  3,109.30                  3,115.88  3,743.90
2,142.00
15          3,091.00  3,090.40                  3,081.16  3,743.90
2,142.00
16          3,066.00  3,065.45                  3,075.05  3,743.90
2,142.00
17          3,087.00  3,078.30                  3,097.76  3,743.90
2,142.00
18          3,093.90  3,096.00                  3,097.88  3,743.90
2,142.00
19          3,112.00  3,110.75                  3,125.40  3,743.90
2,142.00
20          3,112.00  3,113.60                  3,104.41  3,743.90
2,142.00
```

	VOLUME	VALUE	No_of_trades
0	21,25,201	6,60,51,15,019.85	1,24,189
1	22,38,921	6,85,65,87,918.85	1,39,819
2	33,60,209	10,66,55,13,275.40	1,49,926
3	61,49,883	19,75,62,22,591.20	2,34,840
4	24,14,005	7,65,54,20,169.35	1,13,298
5	16,47,048	5,14,02,61,264.10	98,232
6	10,37,474	3,21,39,35,332.10	68,832
7	19,42,427	5,95,64,90,441.70	1,11,808
8	11,05,721	3,29,50,71,825.30	75,405

```
9    6,93,702   2,06,79,11,056.30       54,185
10  14,43,331   4,31,54,94,351.30     1,03,491
11   6,62,085   1,98,80,13,002.45       50,675
12  10,26,941   3,11,49,56,135.35       68,966
13  10,21,653   3,15,04,50,900.40       80,772
14   9,88,357   3,07,96,03,432.30       60,074
15   7,24,174   2,23,12,95,323.25       44,784
16   7,56,036   2,32,48,51,552.85       51,794
17   8,99,781   2,78,73,01,793.45       85,332
18   5,86,710   1,81,75,56,522.45       44,240
19   9,22,403   2,88,28,81,547.45       57,330
20  11,55,112   3,58,59,41,290.20       84,099
```

```
df.columns
```

```
Index(['Date', 'series', 'OPEN', 'HIGH', 'LOW', 'PREV_CLOSE',
       'Last_Traded_Price', 'close', 'Volume_weighted_avg_price',
'52W_H',
       '52W_L', 'VOLUME', 'VALUE', 'No_of_trades'],
      dtype='object')
```

```
print(df['HIGH'].dtype)
```

```
object
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21 entries, 0 to 20
Data columns (total 14 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   Date                        21 non-null     object
 1   series                      21 non-null     object
 2   OPEN                        21 non-null     object
 3   HIGH                        21 non-null     object
 4   LOW                         21 non-null     object
 5   PREV_CLOSE                  21 non-null     object
 6   Last_Traded_Price           21 non-null     object
 7   close                       21 non-null     object
 8   Volume_weighted_avg_price   21 non-null     object
 9   52W_H                       21 non-null     object
 10  52W_L                       21 non-null     object
 11  VOLUME                      21 non-null     object
 12  VALUE                       21 non-null     object
 13  No_of_trades                21 non-null     object
dtypes: object(14)
memory usage: 2.4+ KB
```

```
df['OPEN'] = df['OPEN'].str.replace(',', '',)
df['HIGH'] = df['HIGH'].str.replace(',', '')
```

```
df['LOW'] = df['LOW'].str.replace(',', '')
df['PREV_CLOSE'] = df['PREV_CLOSE'].str.replace(',', '')
df['Last_Traded_Price'] = df['Last_Traded_Price'].str.replace(',', '')
df['close'] = df['close'].str.replace(',', '')
df['52W_H'] = df['52W_H'].str.replace(',', '')
df['Volume_weighted_avg_price'] =
df['Volume_weighted_avg_price'].str.replace(',', '')
df['52W_L'] = df['52W_L'].str.replace(',', '')

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21 entries, 0 to 20
Data columns (total 14 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Date                       21 non-null     object
 1   series                     21 non-null     object
 2   OPEN                       21 non-null     object
 3   HIGH                       21 non-null     object
 4   LOW                        21 non-null     object
 5   PREV_CLOSE                 21 non-null     object
 6   Last_Traded_Price          21 non-null     object
 7   close                      21 non-null     object
 8   Volume_weighted_avg_price  21 non-null     object
 9   52W_H                      21 non-null     object
 10  52W_L                      21 non-null     object
 11  VOLUME                     21 non-null     object
 12  VALUE                      21 non-null     object
 13  No_of_trades               21 non-null     object
dtypes: object(14)
memory usage: 2.4+ KB
```

```
df.drop('series', axis=1, inplace=True)

df
```

```
           Date     OPEN     HIGH      LOW PREV_CLOSE
Last_Traded_Price  \
0   06-Aug-2024  3084.00  3142.80  3057.25    3038.20
3078.25
1   05-Aug-2024  3085.00  3133.05  2996.30    3160.90
3024.00
2   02-Aug-2024  3200.00  3215.00  3111.00    3217.25
3161.40
3   01-Aug-2024  3180.00  3258.00  3151.70    3169.40
3225.10
4   31-Jul-2024  3141.00  3197.50  3135.00    3128.75
3168.00
5   30-Jul-2024  3092.00  3155.00  3066.65    3089.35
```

```
3133.00
6   29-Jul-2024  3094.30  3120.35  3074.00    3080.50
3088.40
7   26-Jul-2024  2995.00  3109.00  2988.10    2973.50
3072.25
8   25-Jul-2024  2960.70  3017.65  2945.10    2970.70
2995.00
9   24-Jul-2024  2995.35  3007.50  2959.30    2995.35
2968.80
10  23-Jul-2024  3020.00  3038.00  2886.35    3000.85
2999.40
11  22-Jul-2024  3005.70  3026.90  2972.15    3005.70
3000.00
12  19-Jul-2024  3092.00  3094.50  3000.00    3092.20
3007.95
13  18-Jul-2024  3106.00  3111.00  3057.15    3109.30
3090.00
14  16-Jul-2024  3110.00  3137.75  3096.10    3090.40
3106.50
15  15-Jul-2024  3066.10  3103.50  3058.35    3065.45
3091.00
16  12-Jul-2024  3090.00  3098.80  3058.35    3078.30
3066.00
17  11-Jul-2024  3118.70  3129.80  3074.10    3096.00
3087.00
18  10-Jul-2024  3120.10  3127.25  3063.40    3110.75
3093.90
19  09-Jul-2024  3115.95  3158.00  3100.50    3113.60
3112.00
20  08-Jul-2024  3147.90  3158.20  3075.00    3147.90
3112.00

      close Volume_weighted_avg_price    52W_H    52W_L     VOLUME  \
0   3072.70                   3108.00  3743.90  2142.00  21,25,201
1   3038.20                   3062.45  3743.90  2142.00  22,38,921
2   3160.90                   3174.06  3743.90  2142.00  33,60,209
3   3217.25                   3212.46  3743.90  2142.00  61,49,883
4   3169.40                   3171.25  3743.90  2142.00  24,14,005
5   3128.75                   3120.89  3743.90  2142.00  16,47,048
6   3089.35                   3097.85  3743.90  2142.00  10,37,474
7   3080.50                   3066.52  3743.90  2142.00  19,42,427
8   2973.50                   2980.02  3743.90  2142.00  11,05,721
9   2970.70                   2980.98  3743.90  2142.00   6,93,702
10  2995.35                   2989.95  3743.90  2142.00  14,43,331
11  3000.85                   3002.66  3743.90  2142.00   6,62,085
12  3005.70                   3033.24  3743.90  2142.00  10,26,941
13  3092.20                   3083.68  3743.90  2142.00  10,21,653
14  3109.30                   3115.88  3743.90  2142.00   9,88,357
15  3090.40                   3081.16  3743.90  2142.00   7,24,174
```

```
16  3065.45                    3075.05  3743.90  2142.00   7,56,036
17  3078.30                    3097.76  3743.90  2142.00   8,99,781
18  3096.00                    3097.88  3743.90  2142.00   5,86,710
19  3110.75                    3125.40  3743.90  2142.00   9,22,403
20  3113.60                    3104.41  3743.90  2142.00  11,55,112
```

```
                   VALUE No_of_trades
0    6,60,51,15,019.85     1,24,189
1    6,85,65,87,918.85     1,39,819
2   10,66,55,13,275.40     1,49,926
3   19,75,62,22,591.20     2,34,840
4    7,65,54,20,169.35     1,13,298
5    5,14,02,61,264.10       98,232
6    3,21,39,35,332.10       68,832
7    5,95,64,90,441.70     1,11,808
8    3,29,50,71,825.30       75,405
9    2,06,79,11,056.30       54,185
10   4,31,54,94,351.30     1,03,491
11   1,98,80,13,002.45       50,675
12   3,11,49,56,135.35       68,966
13   3,15,04,50,900.40       80,772
14   3,07,96,03,432.30       60,074
15   2,23,12,95,323.25       44,784
16   2,32,48,51,552.85       51,794
17   2,78,73,01,793.45       85,332
18   1,81,75,56,522.45       44,240
19   2,88,28,81,547.45       57,330
20   3,58,59,41,290.20       84,099
```

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

```
Date                          0
OPEN                          0
HIGH                          0
LOW                           0
PREV_CLOSE                    0
Last_Traded_Price             0
close                         0
Volume_weighted_avg_price     0
52W_H                         0
52W_L                         0
VOLUME                        0
VALUE                         0
No_of_trades                  0
dtype: int64
```

```
for i in df.columns:
    if i!="Date" and df[i].dtype=="object":
        if ',' in df[i][0]:
            df[i] = df[i].str.replace(',','')
```

```
        df[i] = df[i].astype('float64')
    print(f"Done {i}")
```

```
Done Date
Done OPEN
Done HIGH
Done LOW
Done PREV_CLOSE
Done Last_Traded_Price
Done close
Done Volume_weighted_avg_price
Done 52W_H
Done 52W_L
Done VOLUME
Done VALUE
Done No_of_trades
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21 entries, 0 to 20
Data columns (total 13 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   Date                       21 non-null     object 
 1   OPEN                       21 non-null     float64
 2   HIGH                       21 non-null     float64
 3   LOW                        21 non-null     float64
 4   PREV_CLOSE                 21 non-null     float64
 5   Last_Traded_Price          21 non-null     float64
 6   close                      21 non-null     float64
 7   Volume_weighted_avg_price  21 non-null     float64
 8   52W_H                      21 non-null     float64
 9   52W_L                      21 non-null     float64
 10  VOLUME                     21 non-null     float64
 11  VALUE                      21 non-null     float64
 12  No_of_trades               21 non-null     float64
dtypes: float64(12), object(1)
memory usage: 2.3+ KB
```

```
df
```

```
           Date     OPEN     HIGH      LOW  PREV_CLOSE  
Last_Traded_Price  \
0   06-Aug-2024  3084.00  3142.80  3057.25     3038.20  
3078.25
1   05-Aug-2024  3085.00  3133.05  2996.30     3160.90  
3024.00
2   02-Aug-2024  3200.00  3215.00  3111.00     3217.25  
3161.40
```

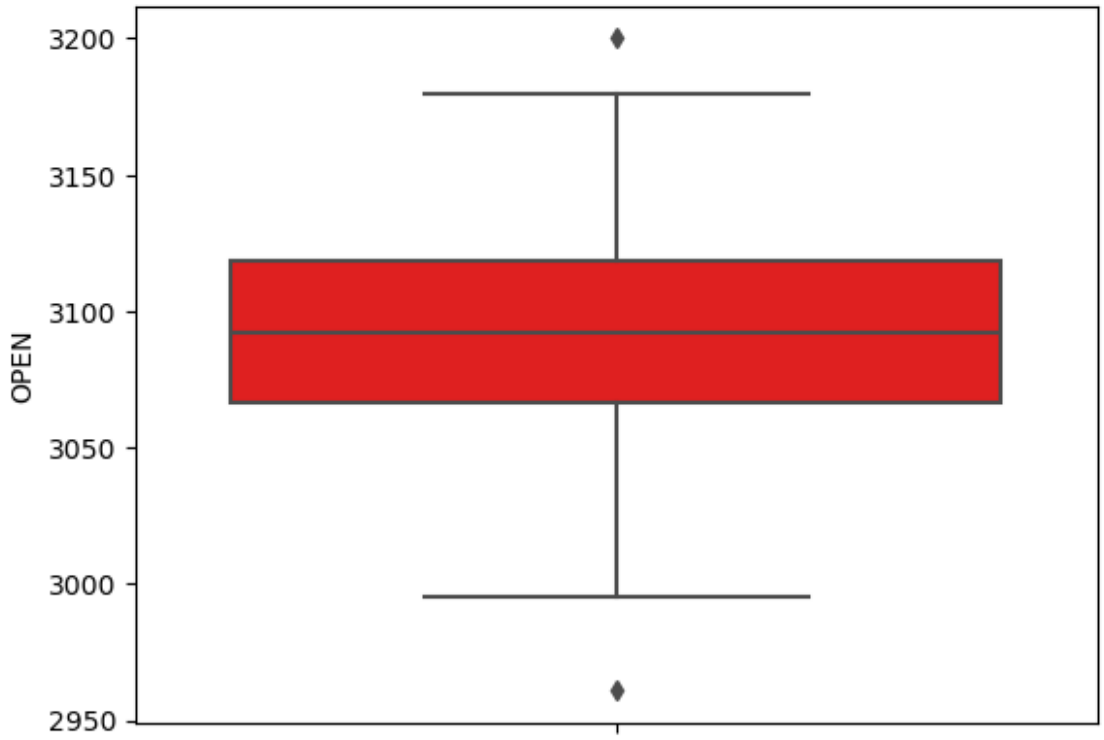
```
3   01-Aug-2024  3180.00  3258.00  3151.70     3169.40
3225.10
4   31-Jul-2024  3141.00  3197.50  3135.00     3128.75
3168.00
5   30-Jul-2024  3092.00  3155.00  3066.65     3089.35
3133.00
6   29-Jul-2024  3094.30  3120.35  3074.00     3080.50
3088.40
7   26-Jul-2024  2995.00  3109.00  2988.10     2973.50
3072.25
8   25-Jul-2024  2960.70  3017.65  2945.10     2970.70
2995.00
9   24-Jul-2024  2995.35  3007.50  2959.30     2995.35
2968.80
10  23-Jul-2024  3020.00  3038.00  2886.35     3000.85
2999.40
11  22-Jul-2024  3005.70  3026.90  2972.15     3005.70
3000.00
12  19-Jul-2024  3092.00  3094.50  3000.00     3092.20
3007.95
13  18-Jul-2024  3106.00  3111.00  3057.15     3109.30
3090.00
14  16-Jul-2024  3110.00  3137.75  3096.10     3090.40
3106.50
15  15-Jul-2024  3066.10  3103.50  3058.35     3065.45
3091.00
16  12-Jul-2024  3090.00  3098.80  3058.35     3078.30
3066.00
17  11-Jul-2024  3118.70  3129.80  3074.10     3096.00
3087.00
18  10-Jul-2024  3120.10  3127.25  3063.40     3110.75
3093.90
19  09-Jul-2024  3115.95  3158.00  3100.50     3113.60
3112.00
20  08-Jul-2024  3147.90  3158.20  3075.00     3147.90
3112.00

      close  Volume_weighted_avg_price   52W_H   52W_L     VOLUME  \
0   3072.70                     3108.00  3743.9  2142.0  2125201.0
1   3038.20                     3062.45  3743.9  2142.0  2238921.0
2   3160.90                     3174.06  3743.9  2142.0  3360209.0
3   3217.25                     3212.46  3743.9  2142.0  6149883.0
4   3169.40                     3171.25  3743.9  2142.0  2414005.0
5   3128.75                     3120.89  3743.9  2142.0  1647048.0
6   3089.35                     3097.85  3743.9  2142.0  1037474.0
7   3080.50                     3066.52  3743.9  2142.0  1942427.0
8   2973.50                     2980.02  3743.9  2142.0  1105721.0
9   2970.70                     2980.98  3743.9  2142.0   693702.0
10  2995.35                     2989.95  3743.9  2142.0  1443331.0
```

```
11  3000.85                    3002.66  3743.9  2142.0   662085.0
12  3005.70                    3033.24  3743.9  2142.0  1026941.0
13  3092.20                    3083.68  3743.9  2142.0  1021653.0
14  3109.30                    3115.88  3743.9  2142.0   988357.0
15  3090.40                    3081.16  3743.9  2142.0   724174.0
16  3065.45                    3075.05  3743.9  2142.0   756036.0
17  3078.30                    3097.76  3743.9  2142.0   899781.0
18  3096.00                    3097.88  3743.9  2142.0   586710.0
19  3110.75                    3125.40  3743.9  2142.0   922403.0
20  3113.60                    3104.41  3743.9  2142.0  1155112.0
```

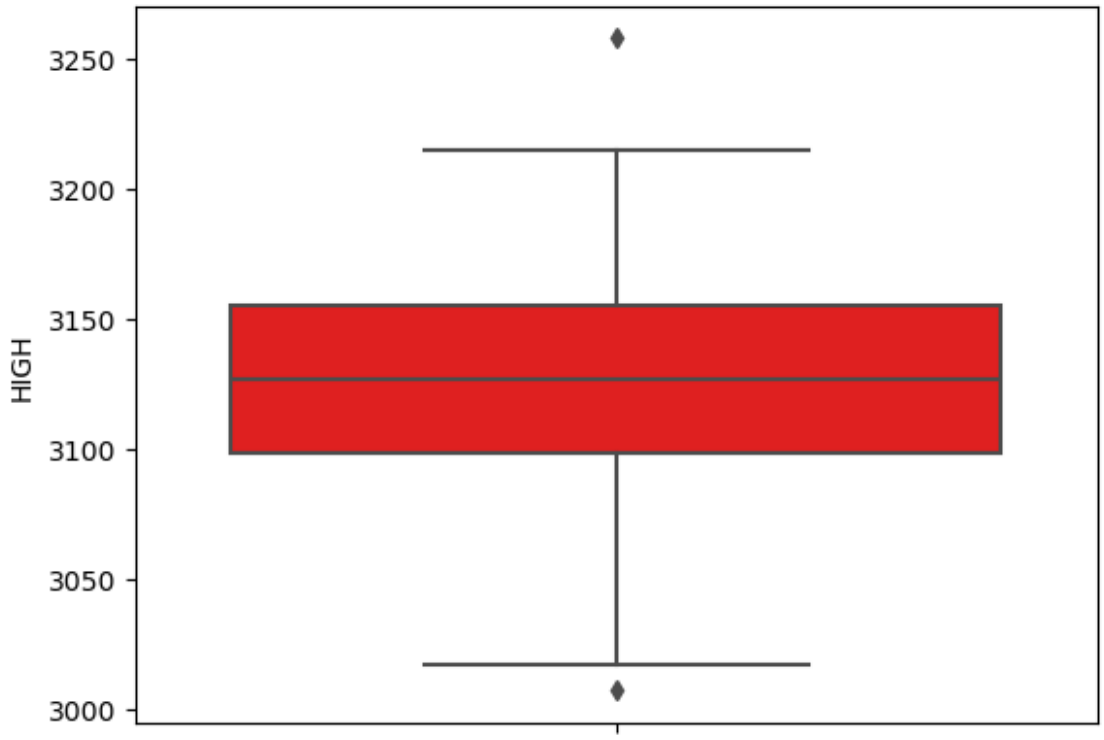
```
            VALUE  No_of_trades
0   6.605115e+09      124189.0
1   6.856588e+09      139819.0
2   1.066551e+10      149926.0
3   1.975622e+10      234840.0
4   7.655420e+09      113298.0
5   5.140261e+09       98232.0
6   3.213935e+09       68832.0
7   5.956490e+09      111808.0
8   3.295072e+09       75405.0
9   2.067911e+09       54185.0
10  4.315494e+09      103491.0
11  1.988013e+09       50675.0
12  3.114956e+09       68966.0
13  3.150451e+09       80772.0
14  3.079603e+09       60074.0
15  2.231295e+09       44784.0
16  2.324852e+09       51794.0
17  2.787302e+09       85332.0
18  1.817557e+09       44240.0
19  2.882882e+09       57330.0
20  3.585941e+09       84099.0
```

```
#Boxplot
for i in df.columns:
    if df[i].dtype!='object':
        print(f"==========={i}===========")
        sns.boxplot(y=df[i],color='red')
        plt.show()
```

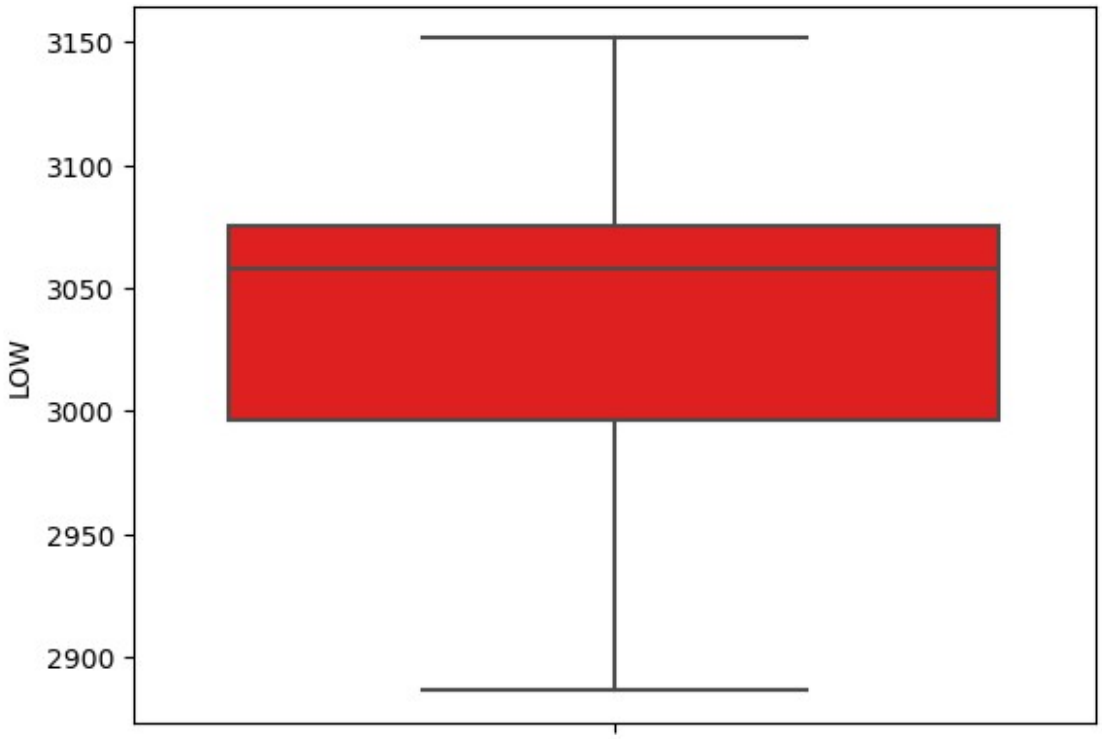
```
===========OPEN===========
```


3200
3150
3100
3050
3000
2950
OPEN

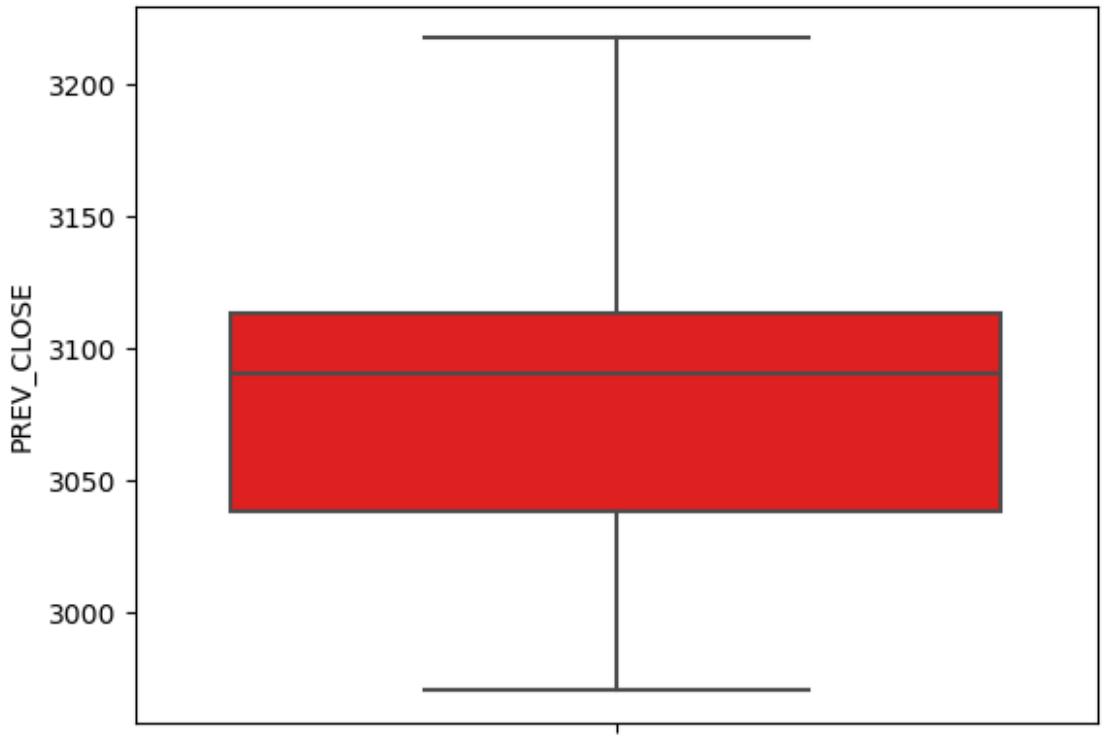
```
==========HIGH==========
```


3250
3200
3150
3100
3050
3000
HIGH

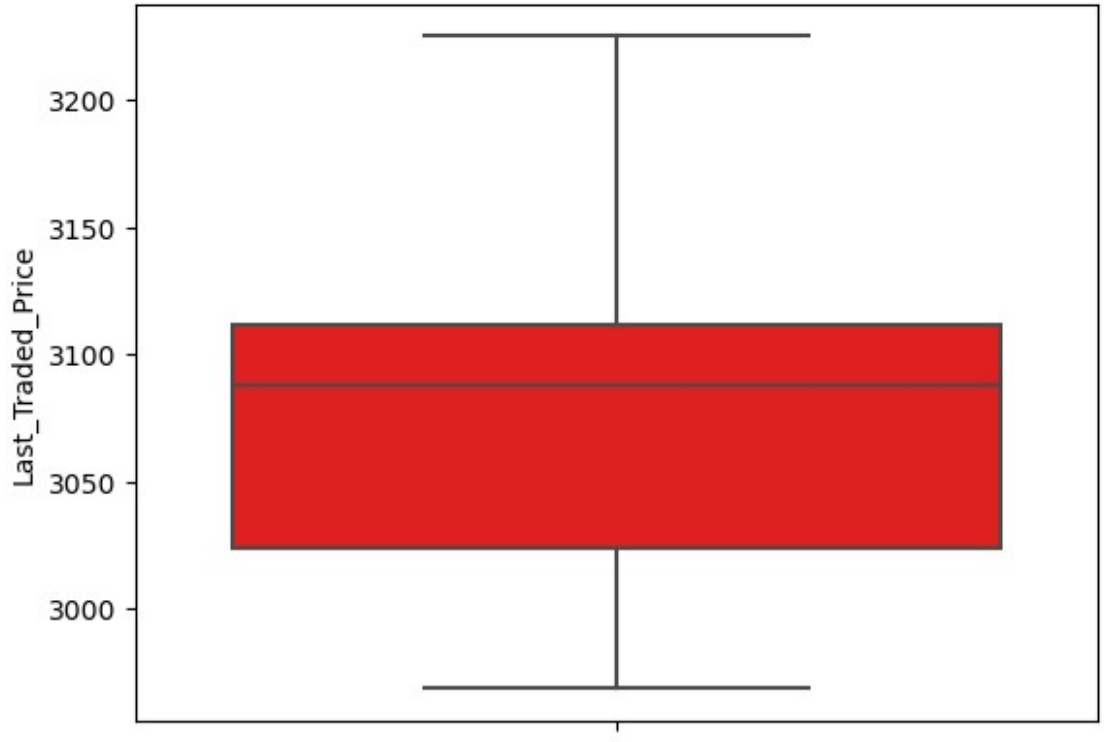
```
==========LOW==========
```



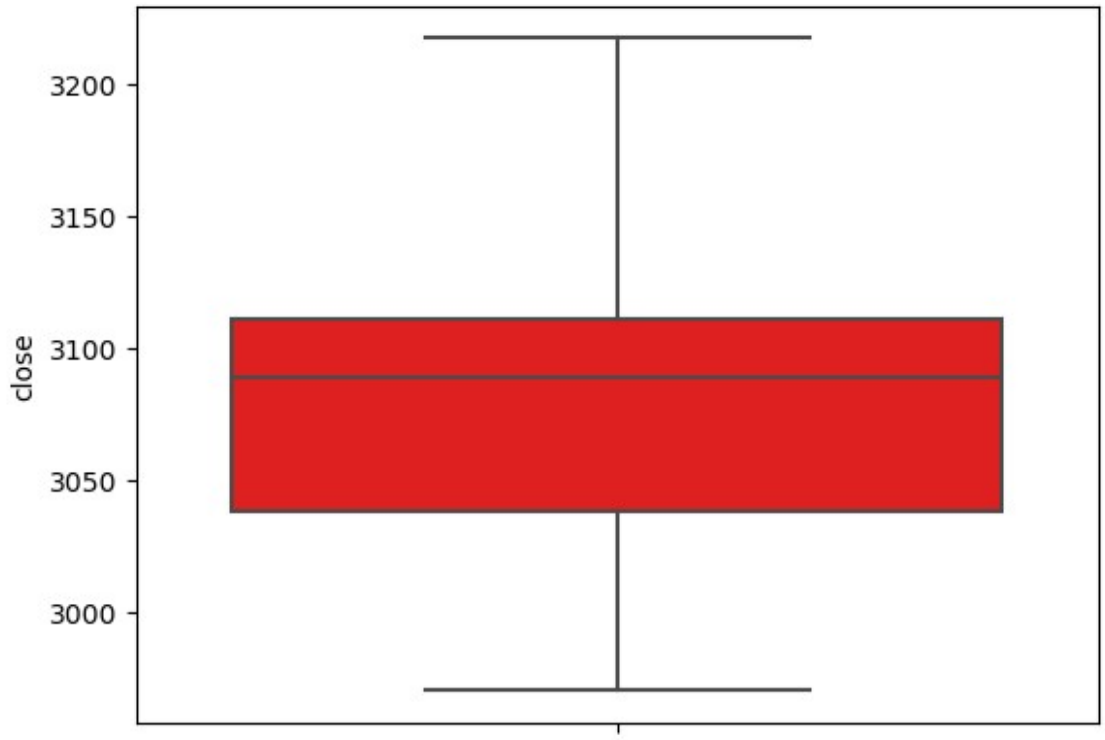

```
==========PREV_CLOSE==========
```



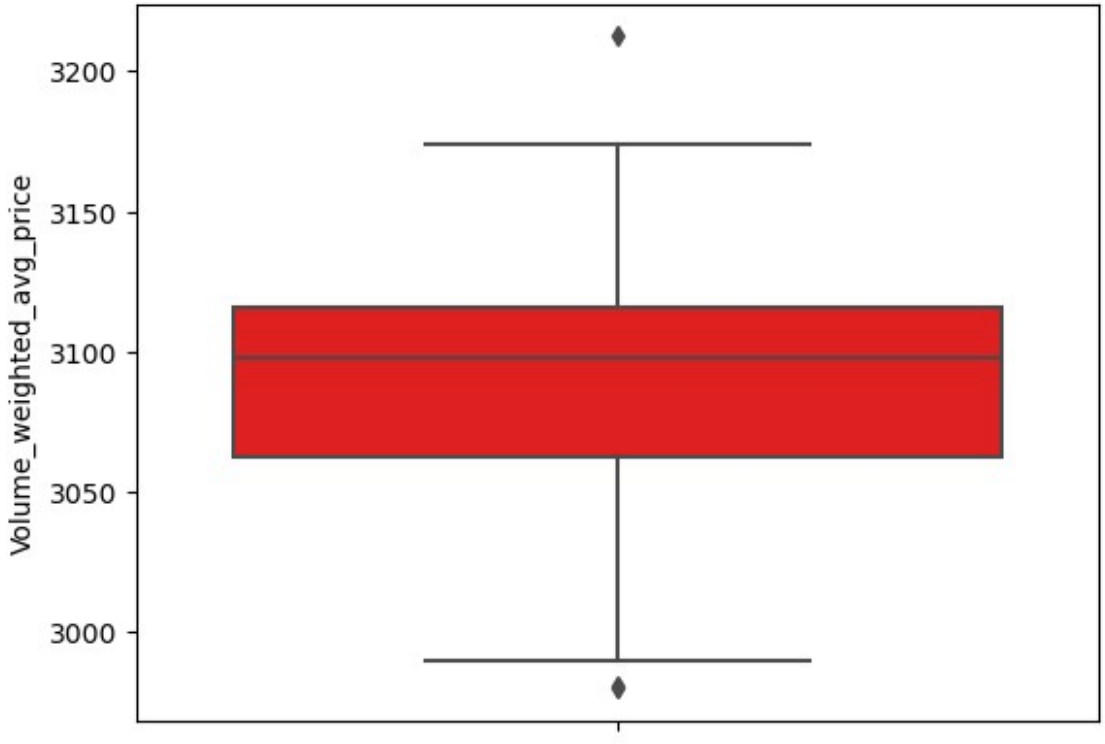

```
==========Last_Traded_Price==========
```




```
==========close==========
```



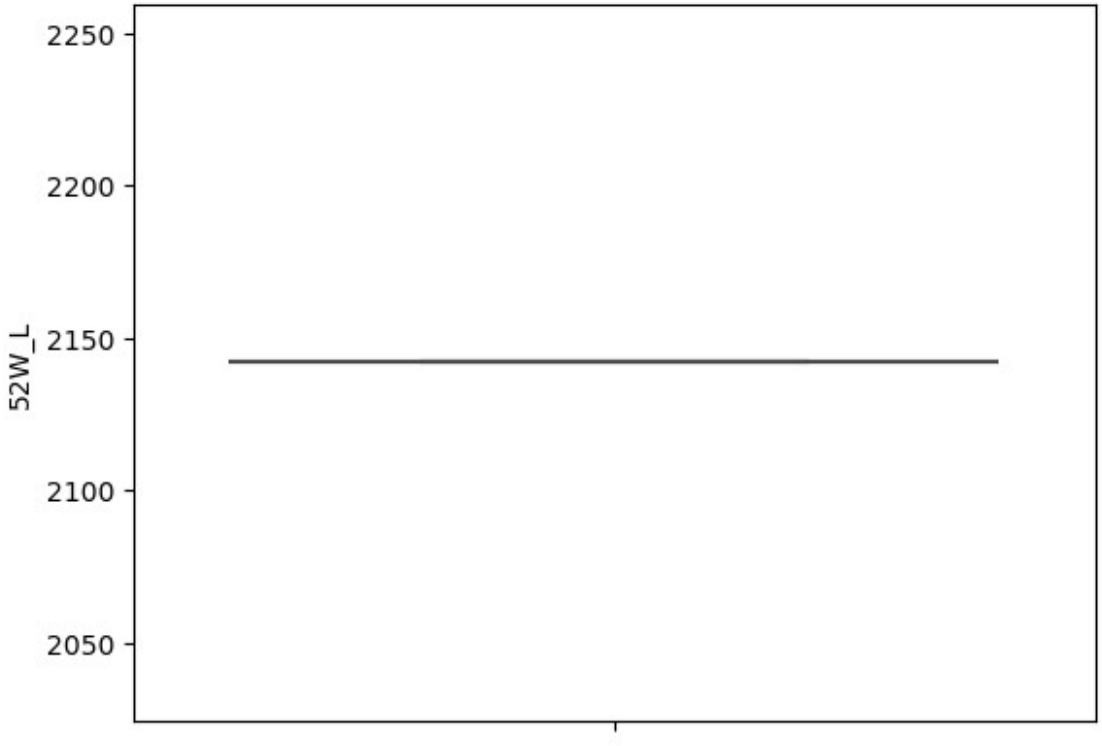

```
==========Volume_weighted_avg_price==========
```



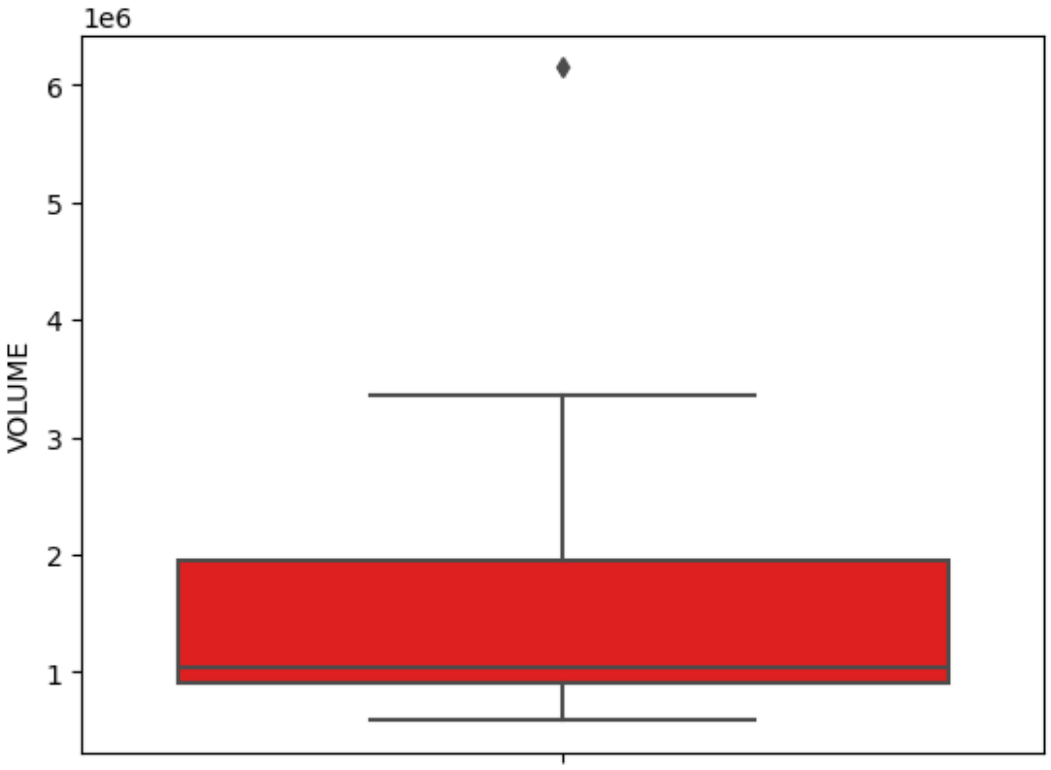

```
==========52W_H==========
```


52W_H
3900
3850
3800
3750
3700
3650
3600
3550

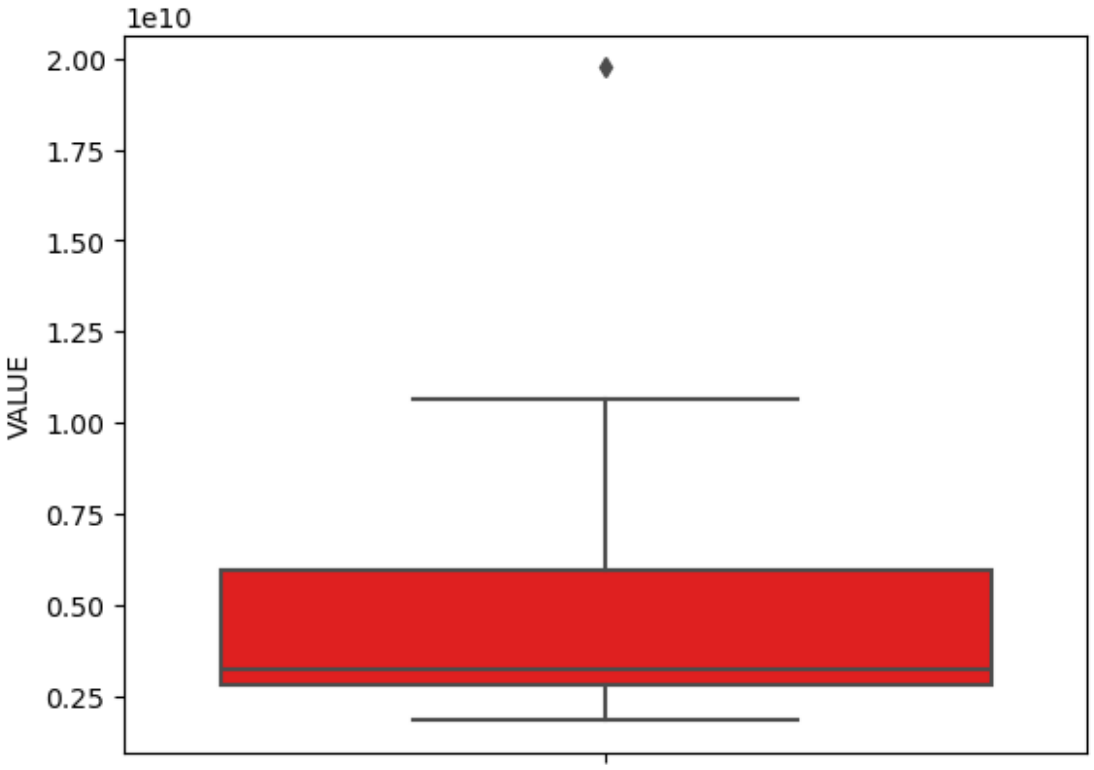
```
==========52W_L==========
```


52W_L
2250
2200
2150
2100
2050

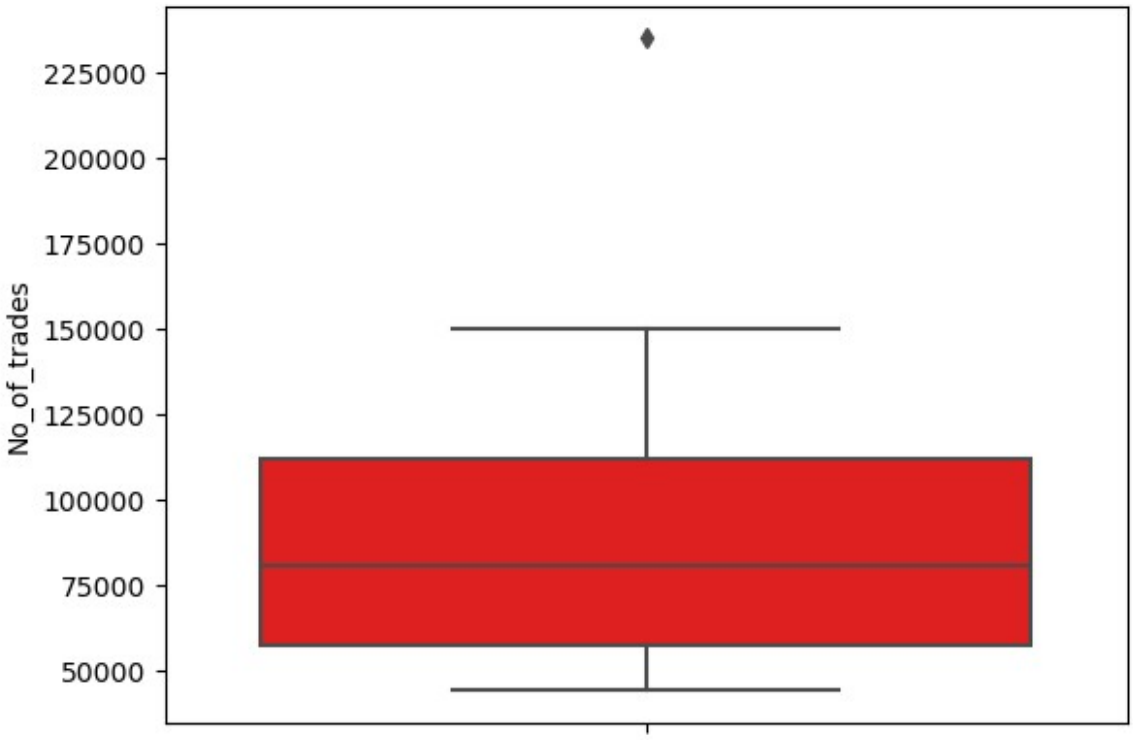
```
==========VOLUME==========
```



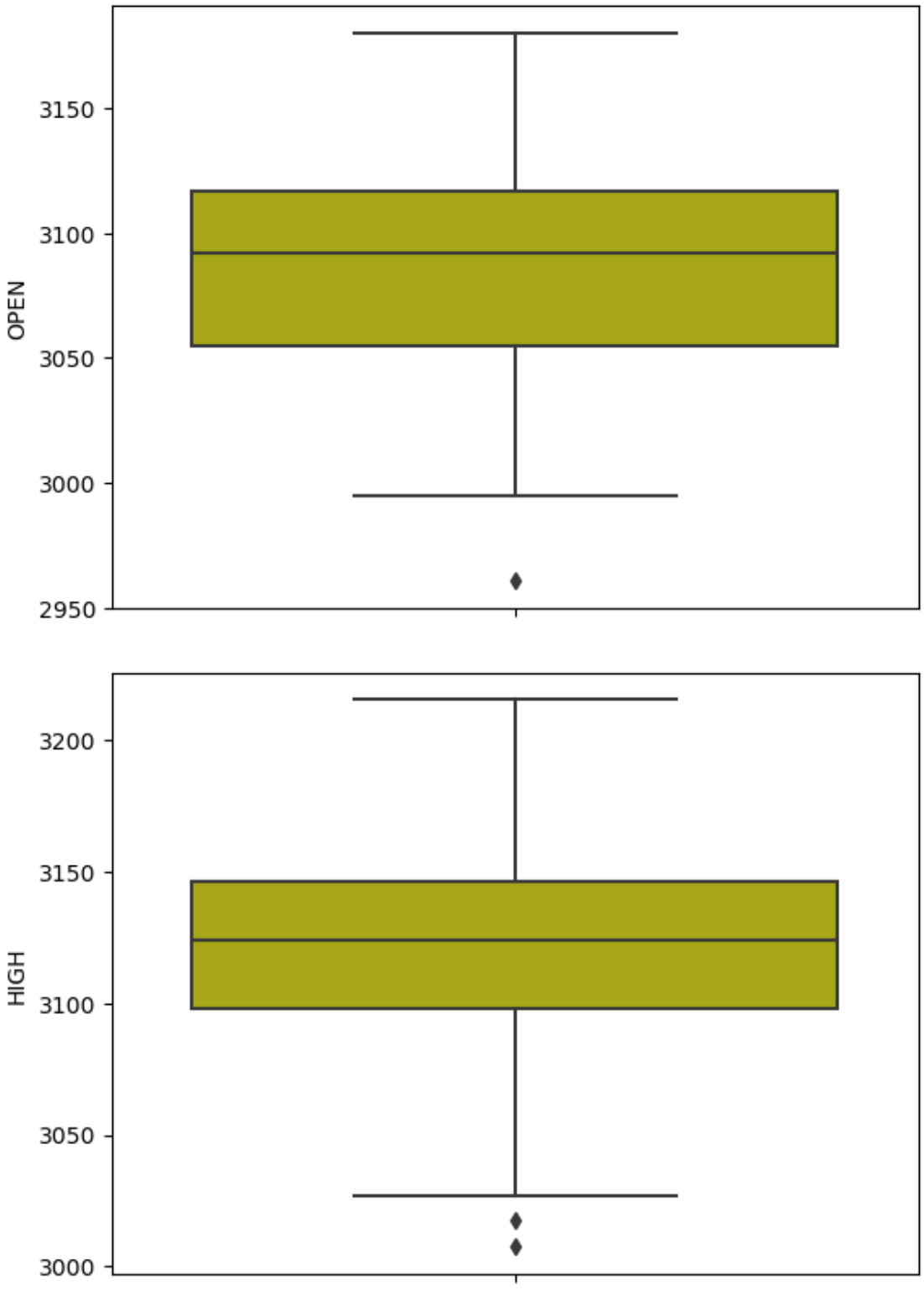

```
==========VALUE==========
```

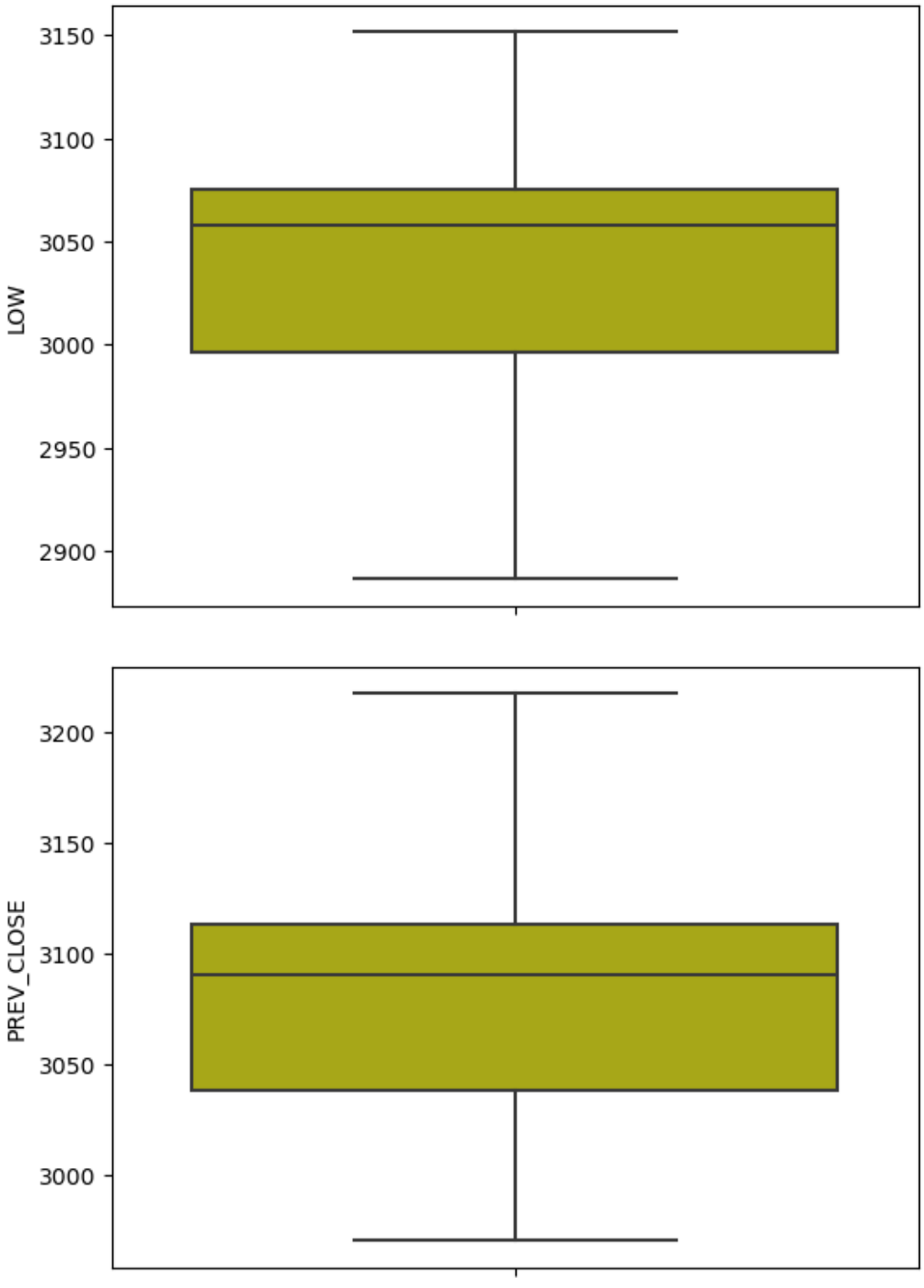


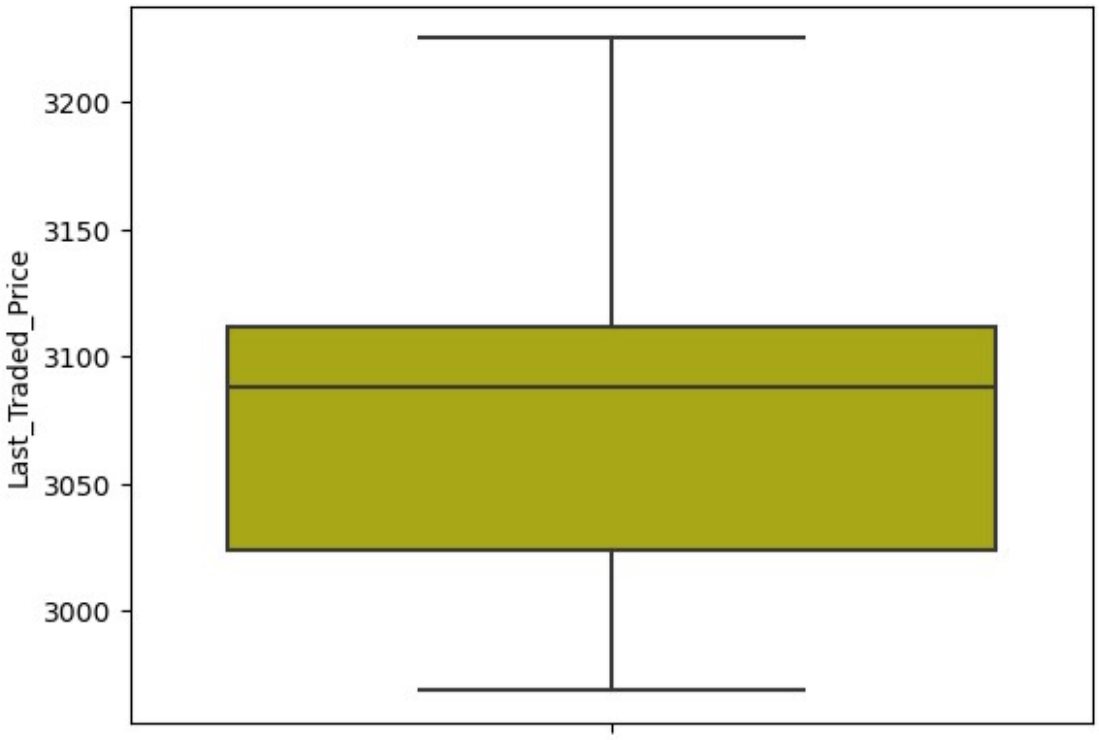
```
==========No_of_trades==========
```

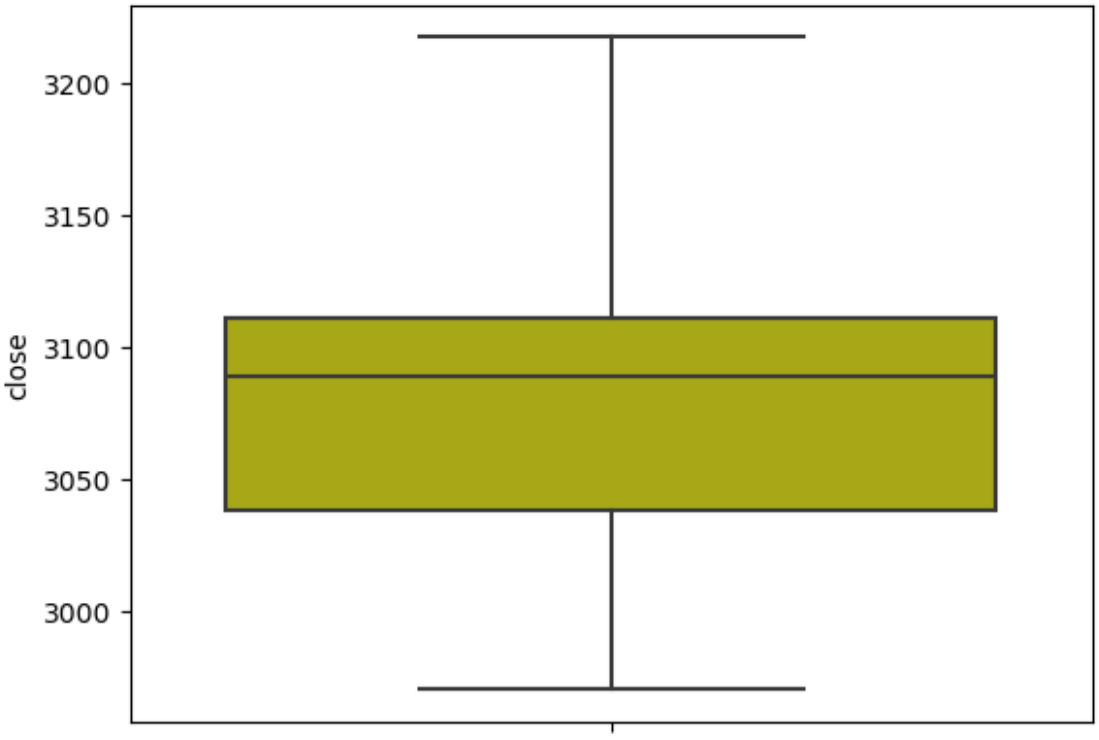


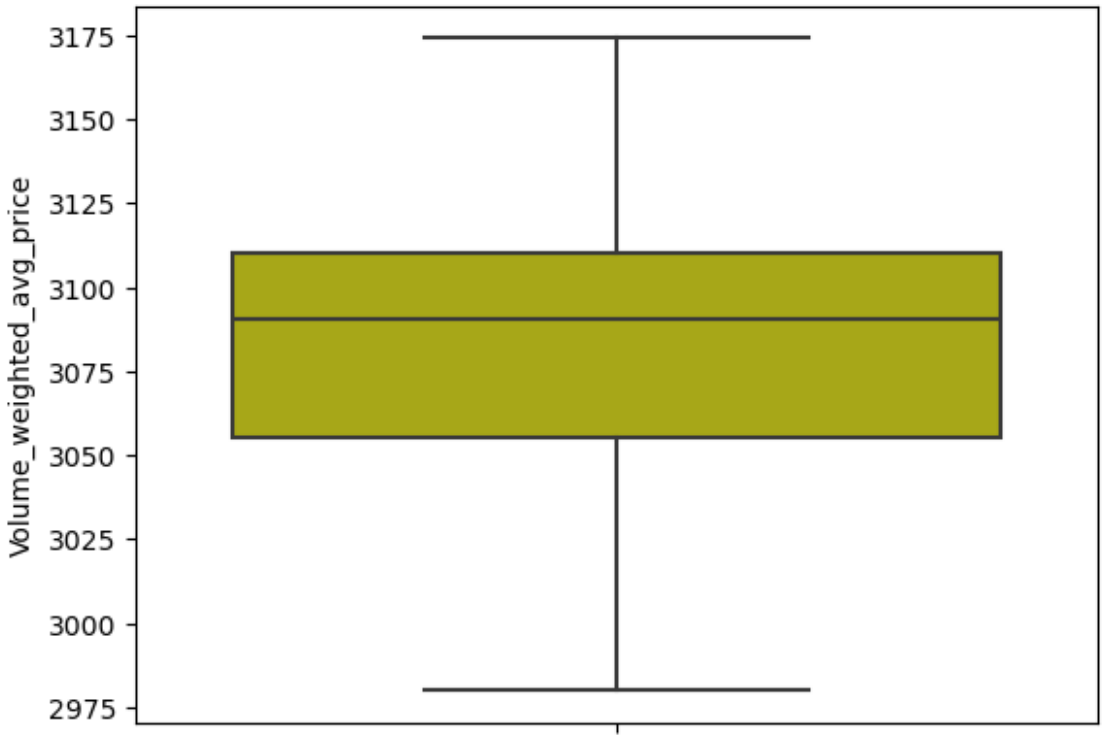

```
def outlier_limit(col):
 Q3,Q1=np.nanpercentile(col,[75,25])
 IQR=Q3-Q1
 UL=Q3+1.5*IQR
 LL=Q1-1.5*Q1
 return UL,LL

for column in df.columns:
   if df[column].dtype!='object':
     UL,LL=outlier_limit(df[column])
     df[column]=np.where((df[column]>UL)|
(df[column]<LL),np.nan,df[column])

for i in df.columns:
  if df[i].dtype!='object':
    sns.boxplot(y=df[i],color='y')
    plt.show()
```

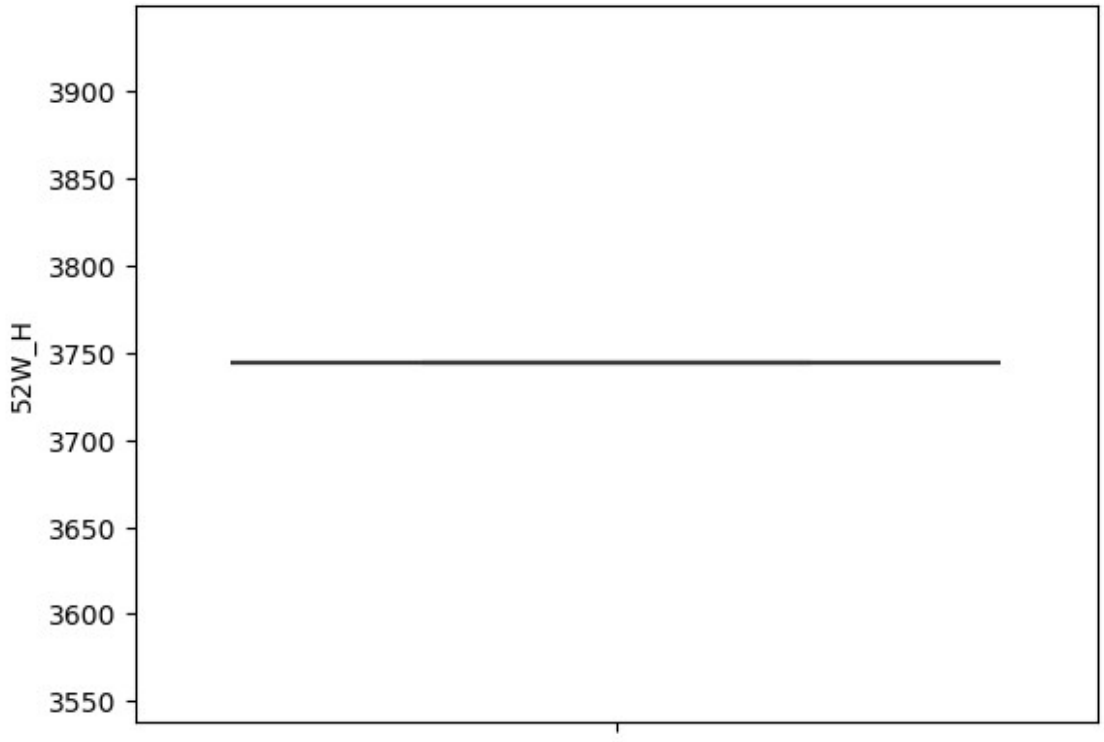

OPEN
3150
3100
3050
3000
2950
HIGH
3200
3150
3100
3050
3000


LOW
3150
3100
3050
3000
2950
2900
PREV_CLOSE
3200
3150
3100
3050
3000

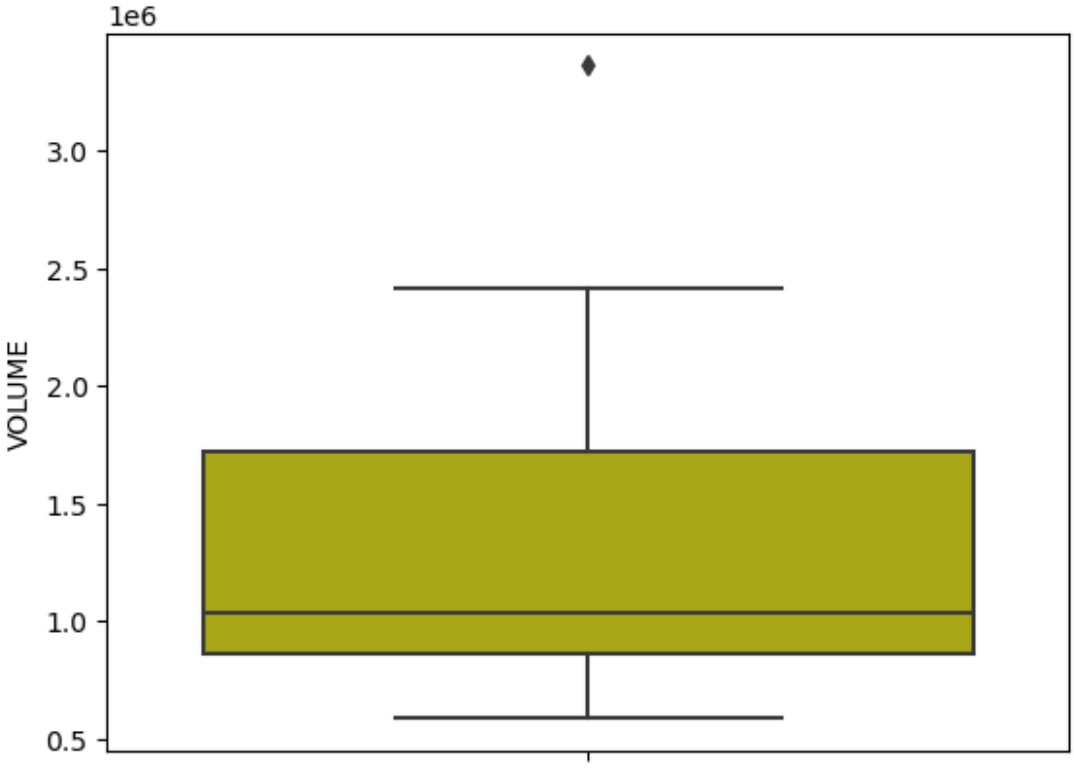

Last_Traded_Price
3200
3150
3100
3050
3000

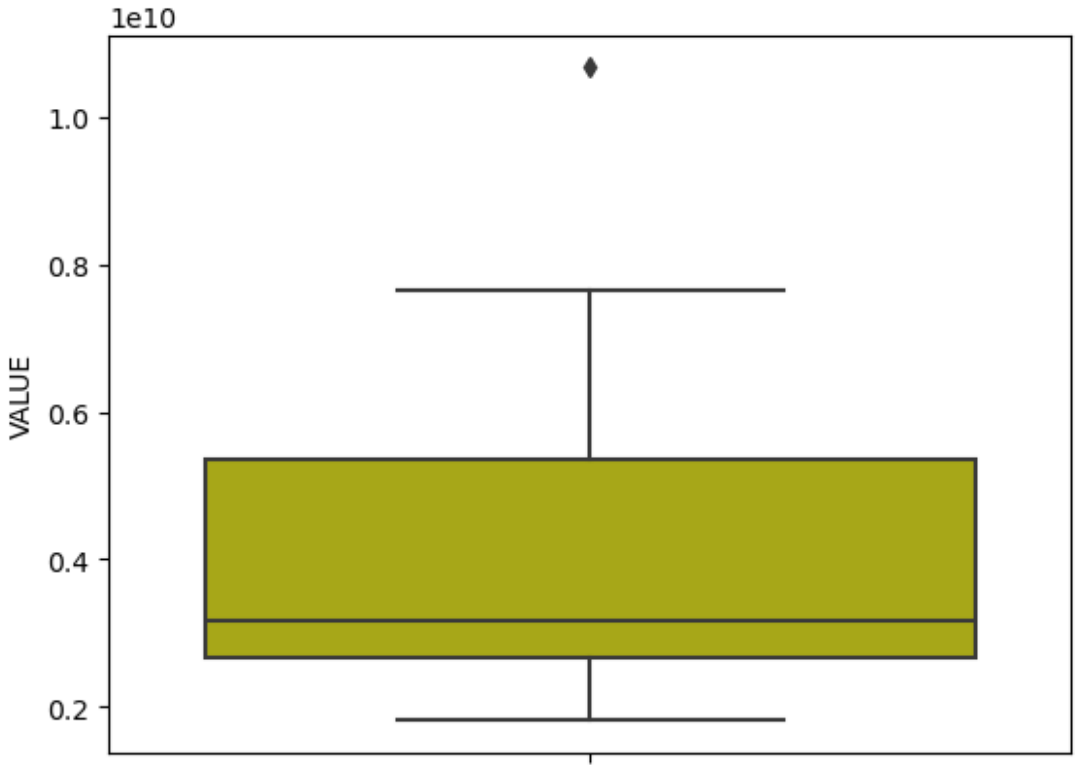

close
3200
3150
3100
3050
3000

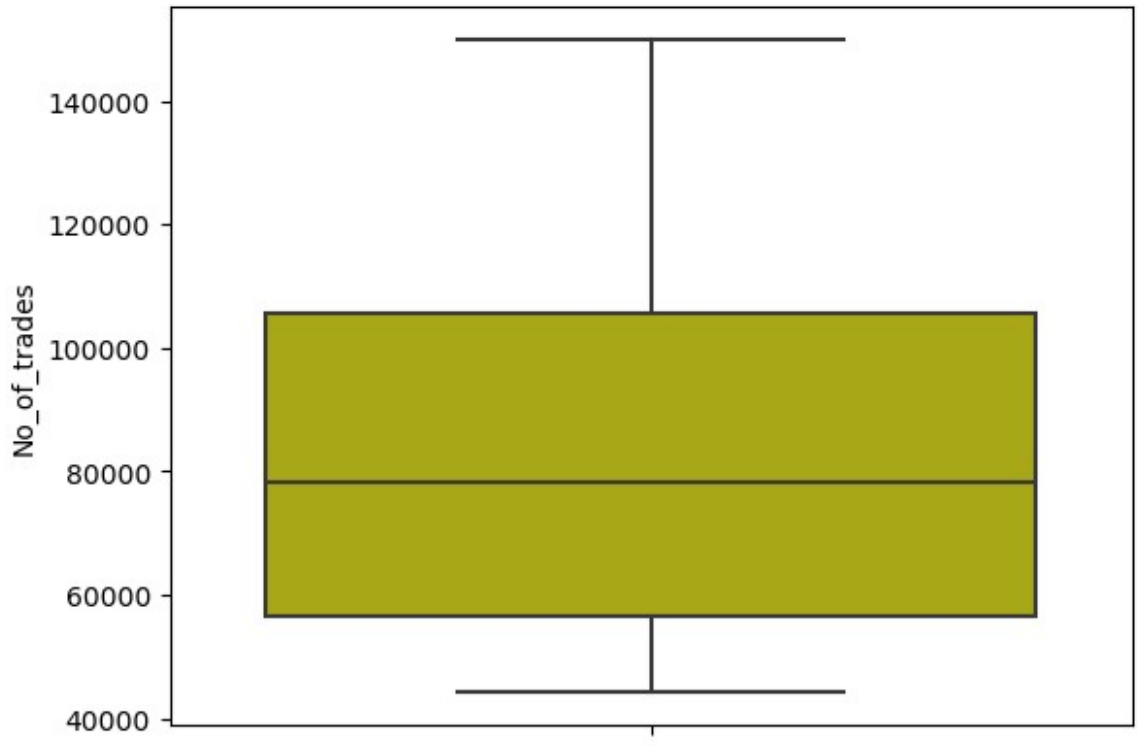

Volume_weighted_avg_price
3175
3150
3125
3100
3075
3050
3025
3000
2975


52W_H
3900
3850
3800
3750
3700
3650
3600
3550


2250
2200
2150
2100
2050
52W_L

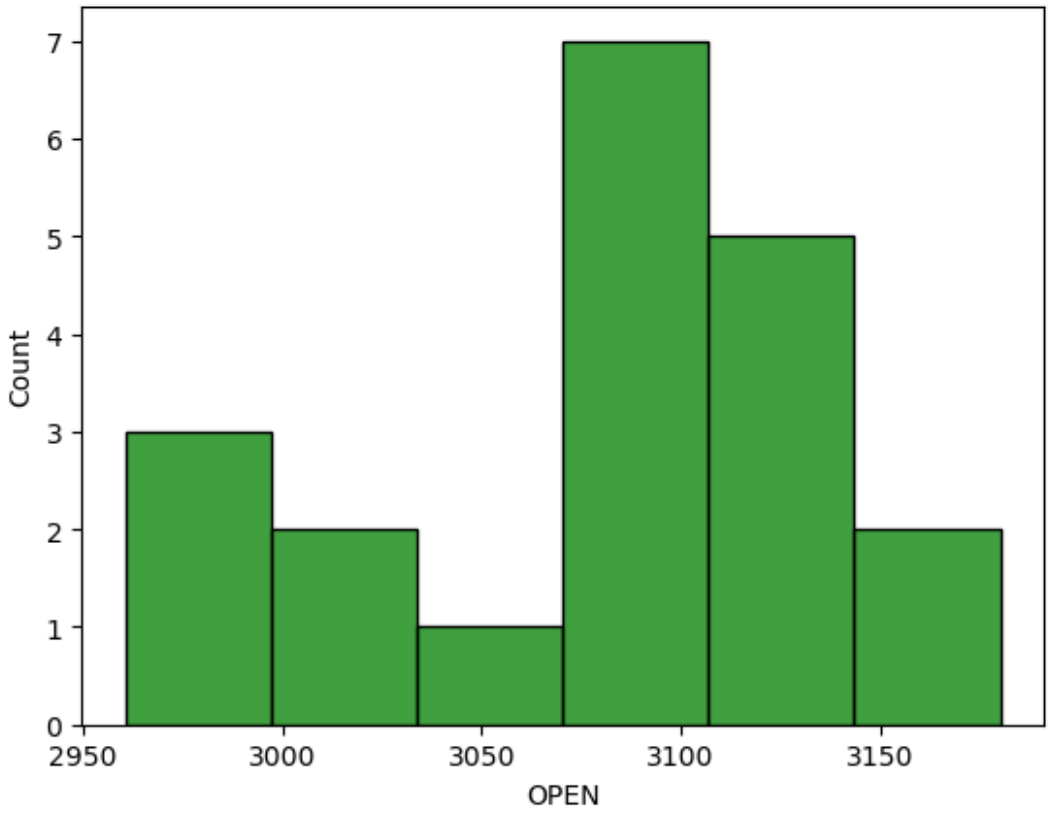

1e6
3.0
2.5
2.0
1.5
1.0
0.5
VOLUME

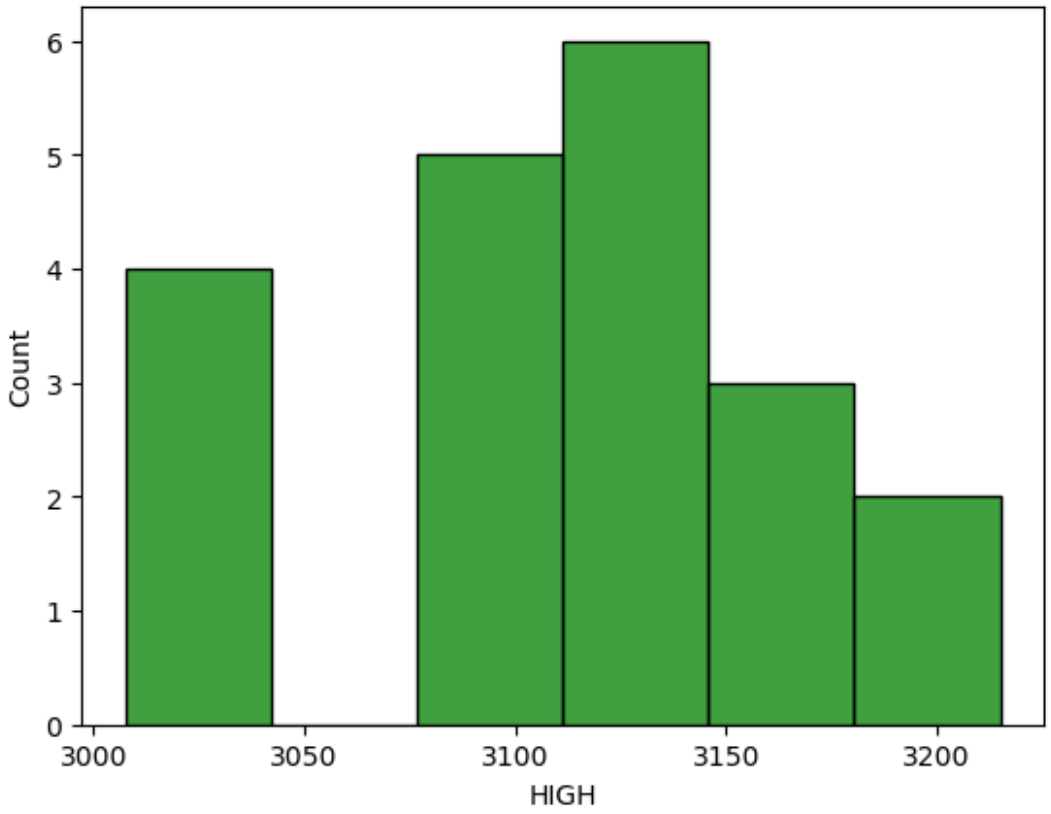


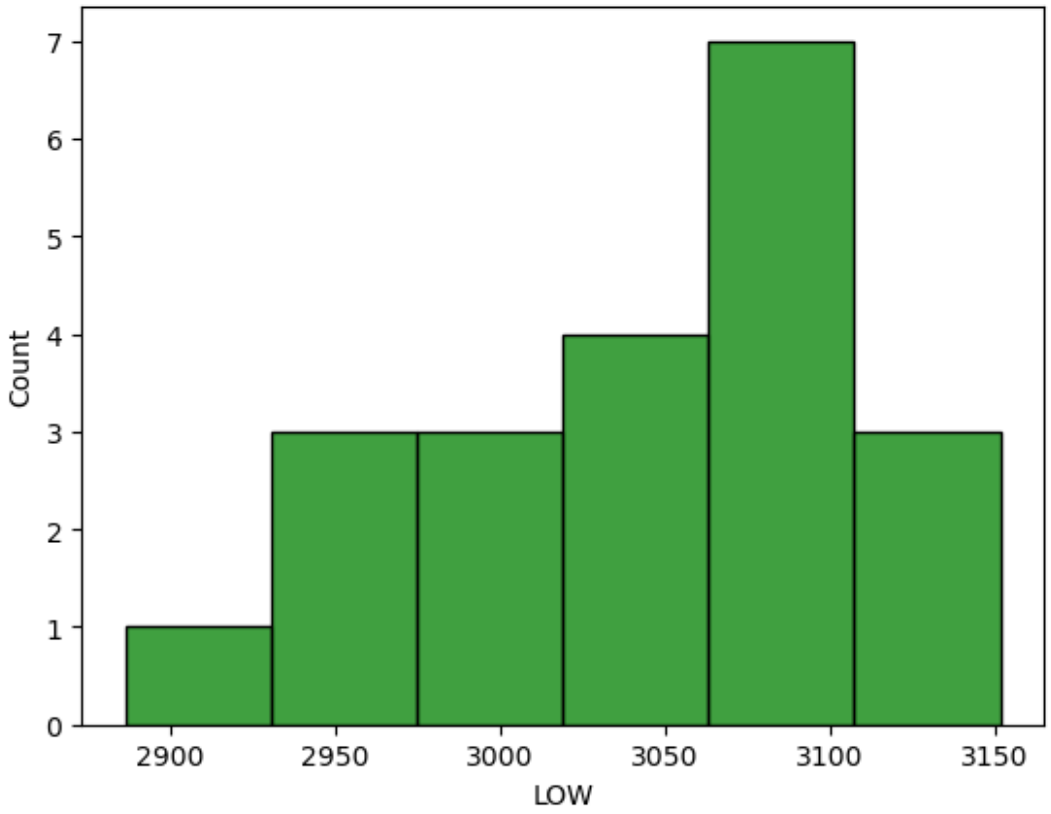


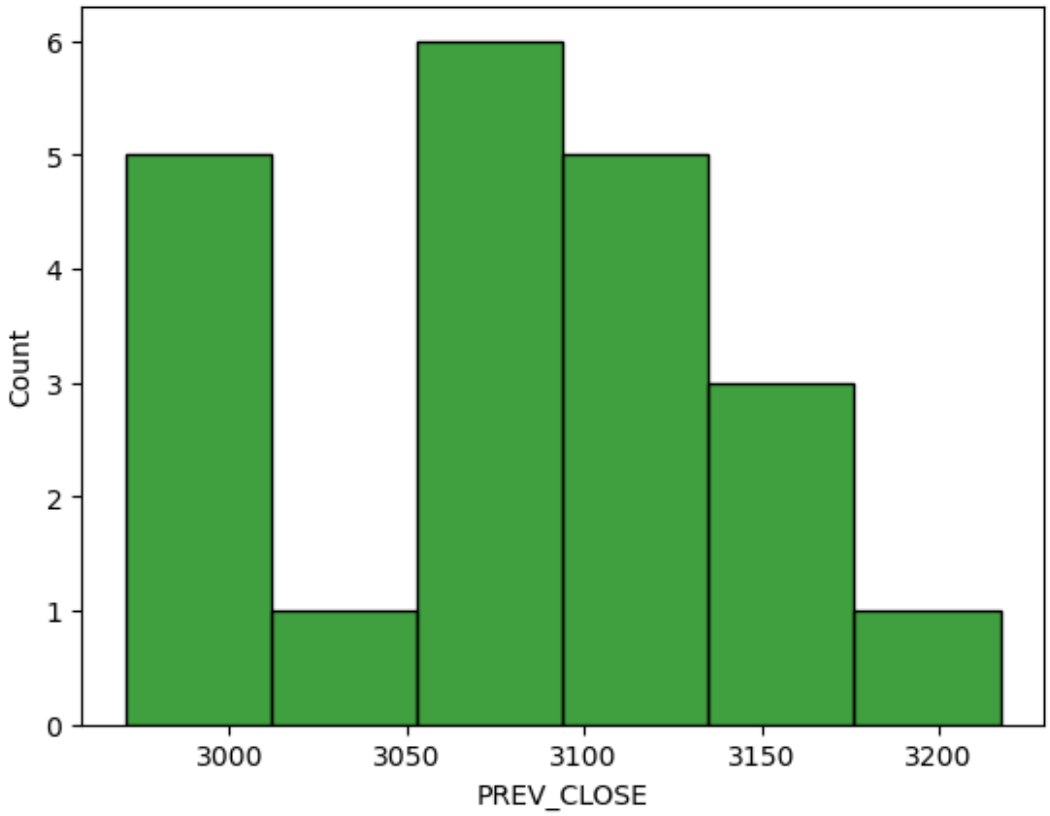

```
for i in df.columns:
  if df[i].dtype !="object":
```

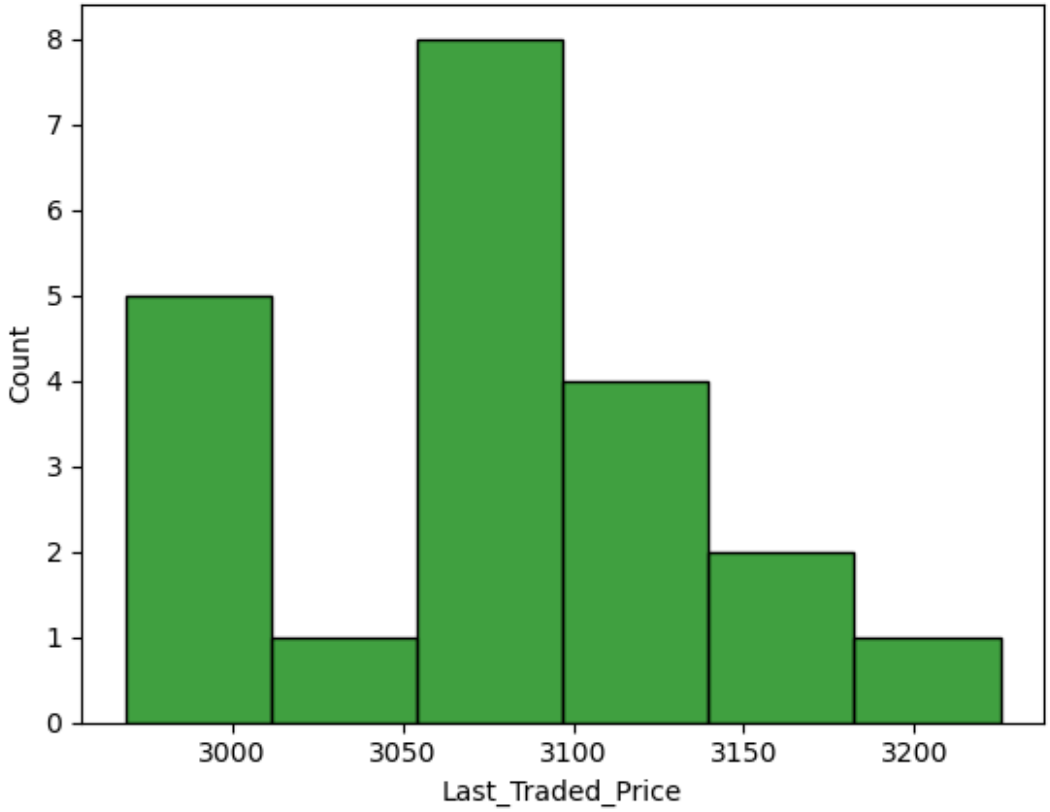
```
sns.histplot(x=df[i],color='green')
plt.show()
```

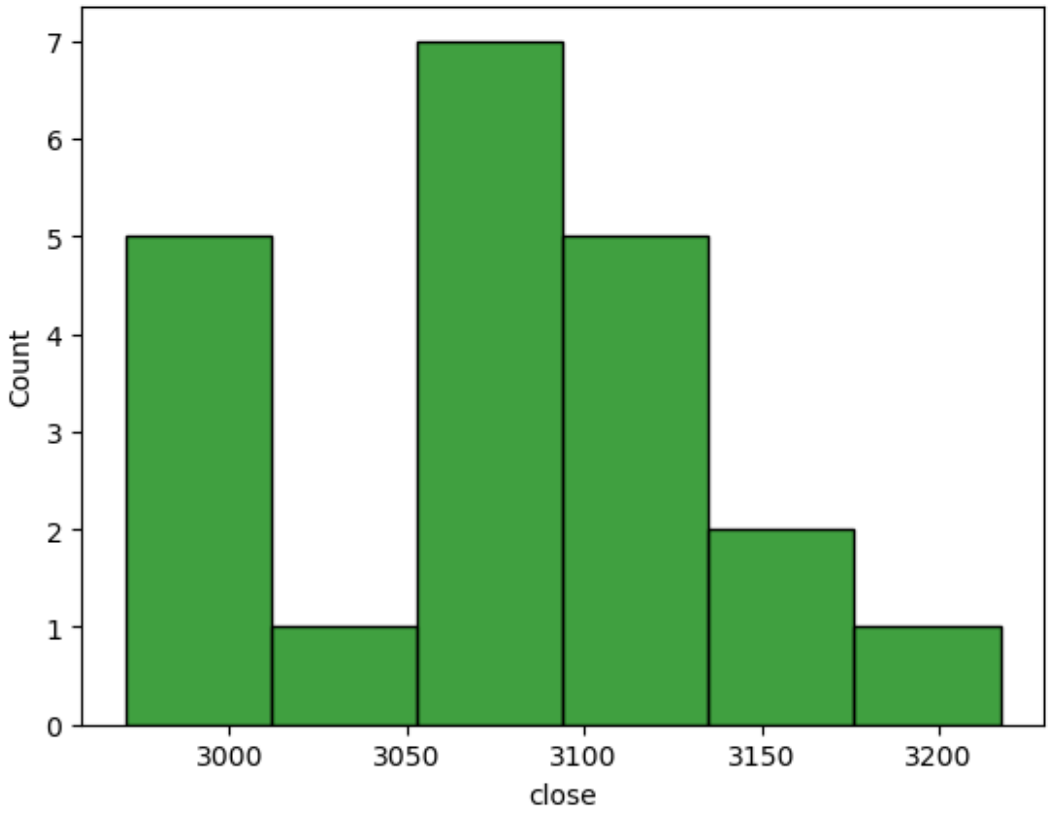


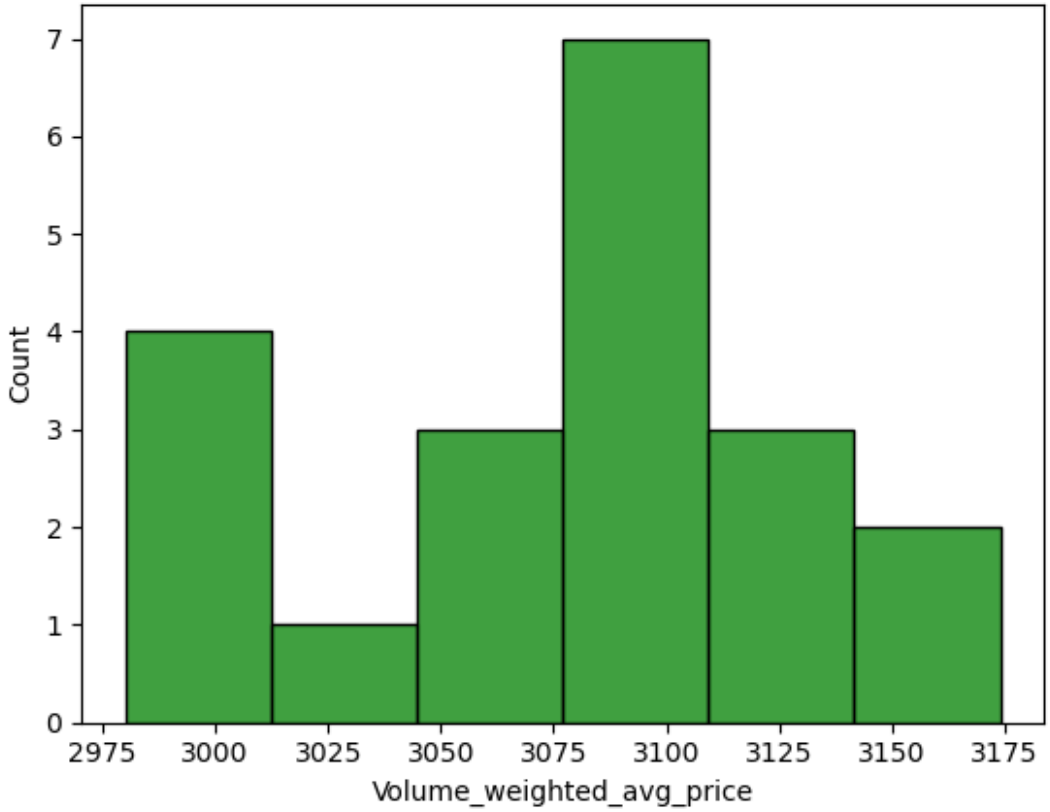

Count
0
1
2
3
4
5
6
3000
3050
3100
3150
3200
HIGH

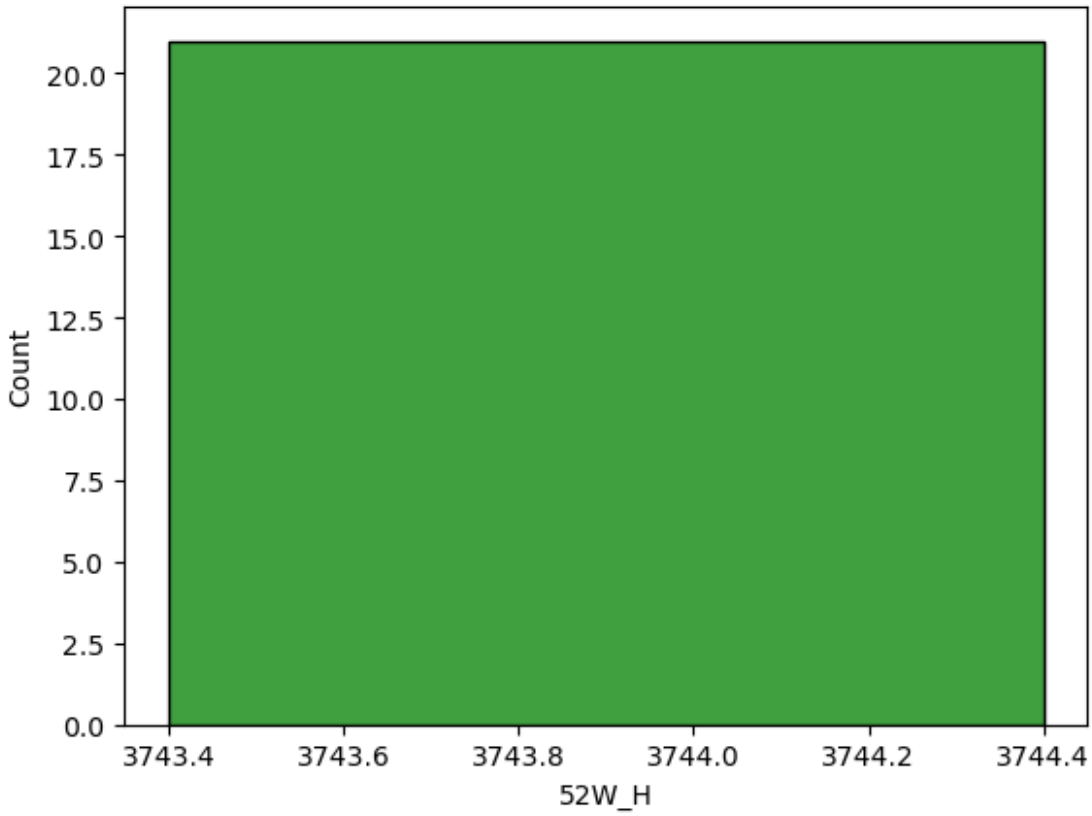

Count
0
1
2
3
4
5
6
7
2900
2950
3000
3050
3100
3150
LOW

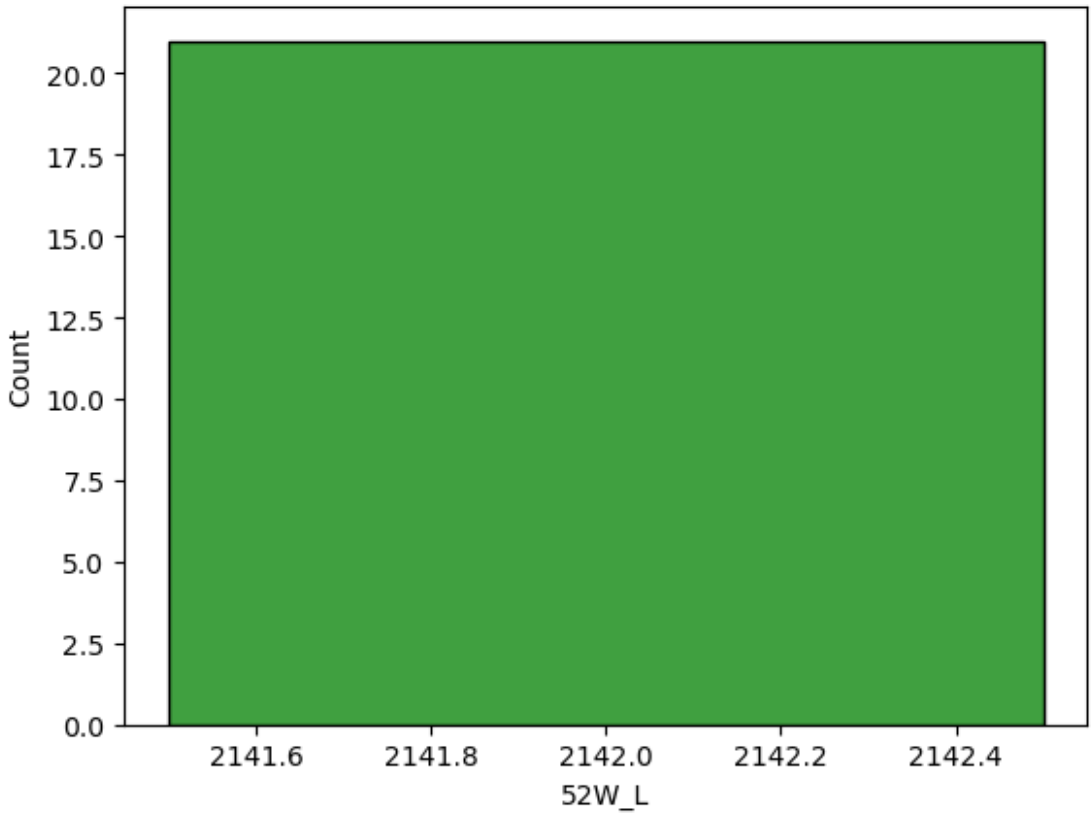

Count
0
1
2
3
4
5
6
3000
3050
3100
3150
3200
PREV_CLOSE

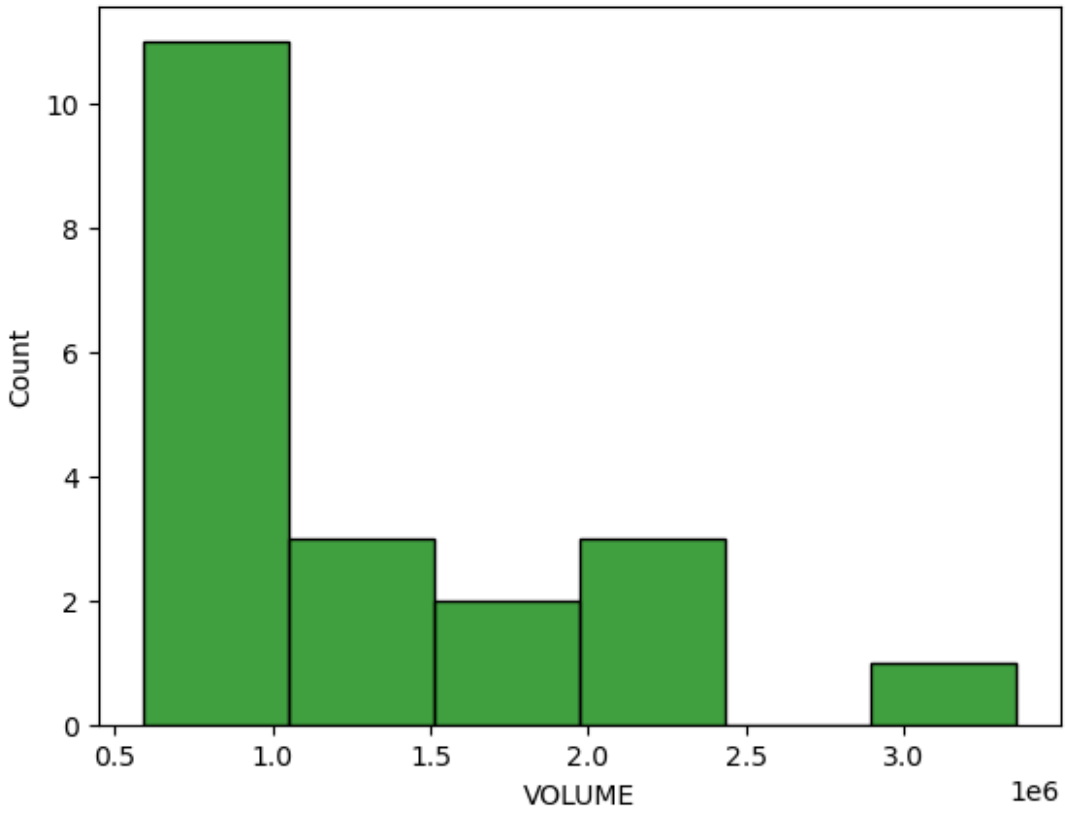

Count
0
1
2
3
4
5
6
7
8
3000
3050
3100
3150
3200
Last_Traded_Price

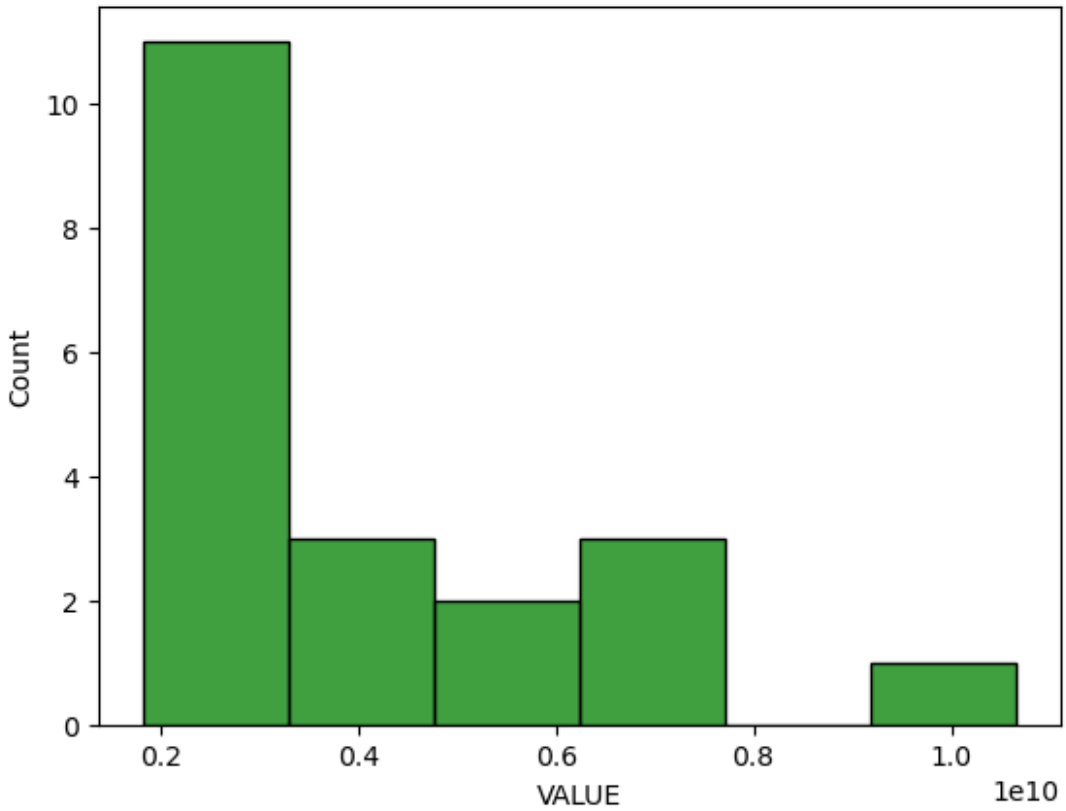

Count
0
1
2
3
4
5
6
7
3000
3050
3100
3150
3200
close

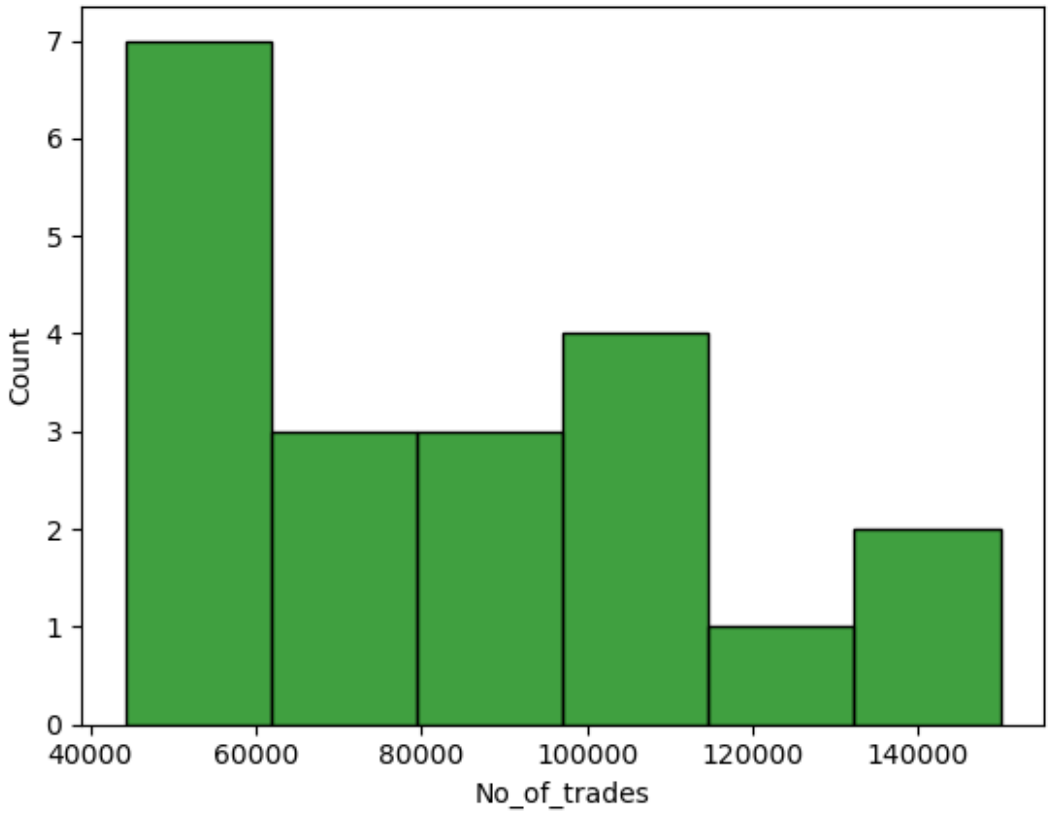

Count
0
1
2
3
4
5
6
7
2975
3000
3025
3050
3075
3100
3125
3150
3175
Volume_weighted_avg_price


Count
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
3743.4
3743.6
3743.8
3744.0
3744.2
3744.4
52W_H

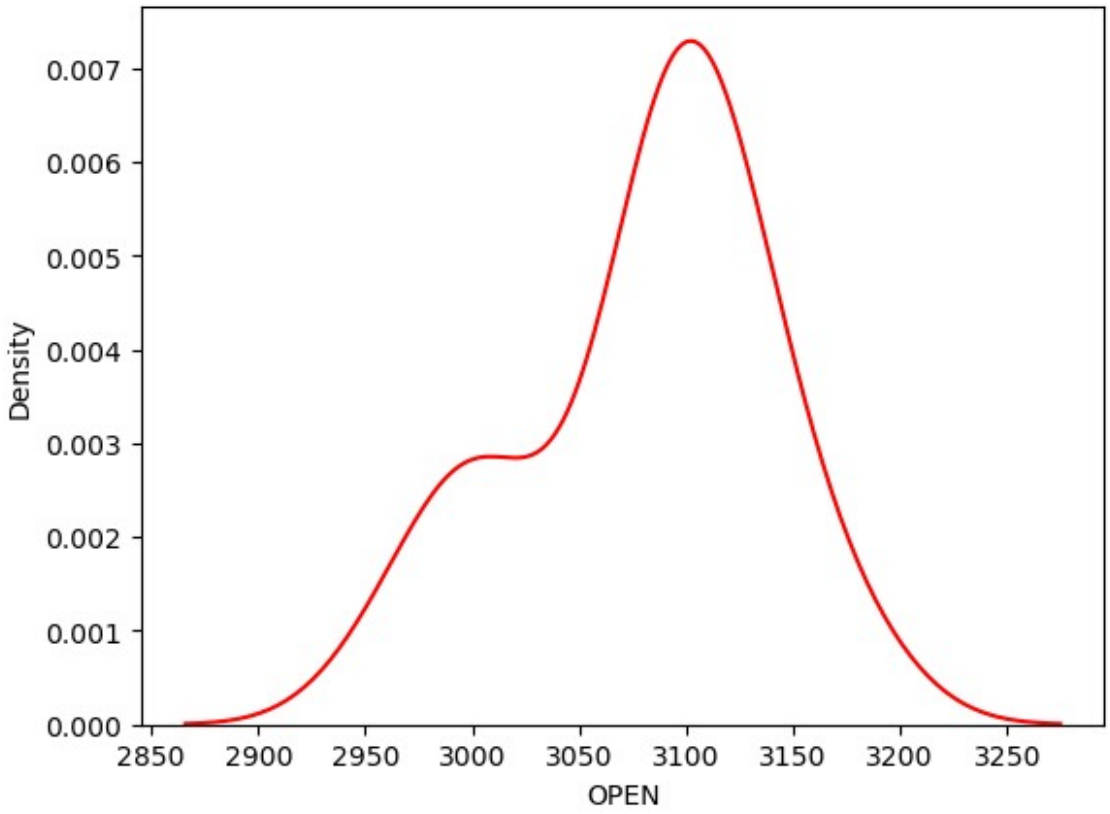

Count
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
2141.6
2141.8
2142.0
2142.2
2142.4
52W_L

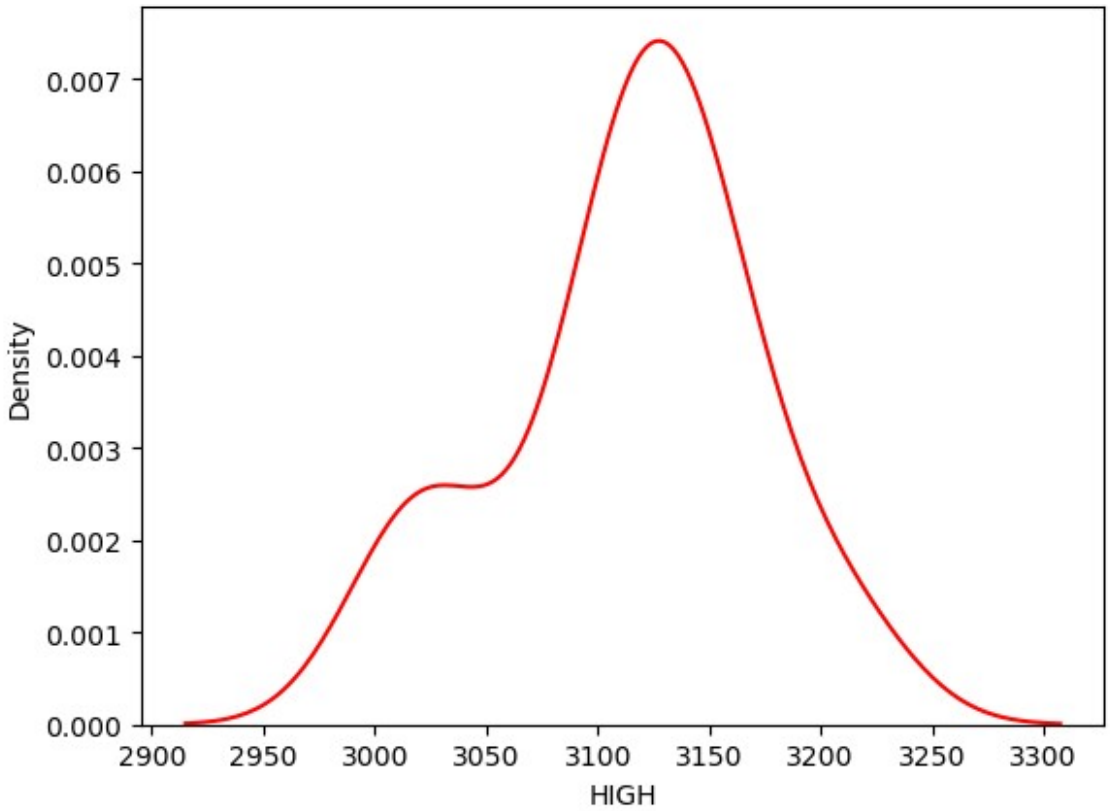

Count
VOLUME
1e6
0
2
4
6
8
10
0.5
1.0
1.5
2.0
2.5
3.0

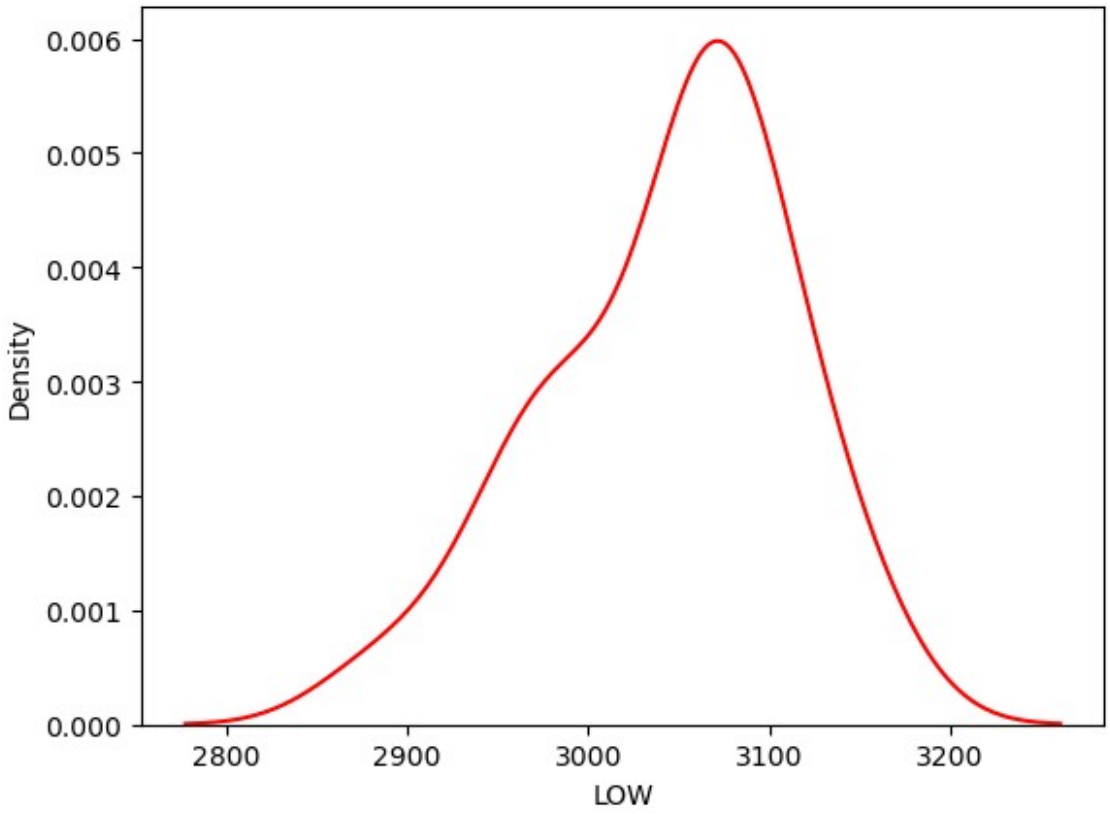

Count
VALUE
1e10
0
2
4
6
8
10
0.2
0.4
0.6
0.8
1.0

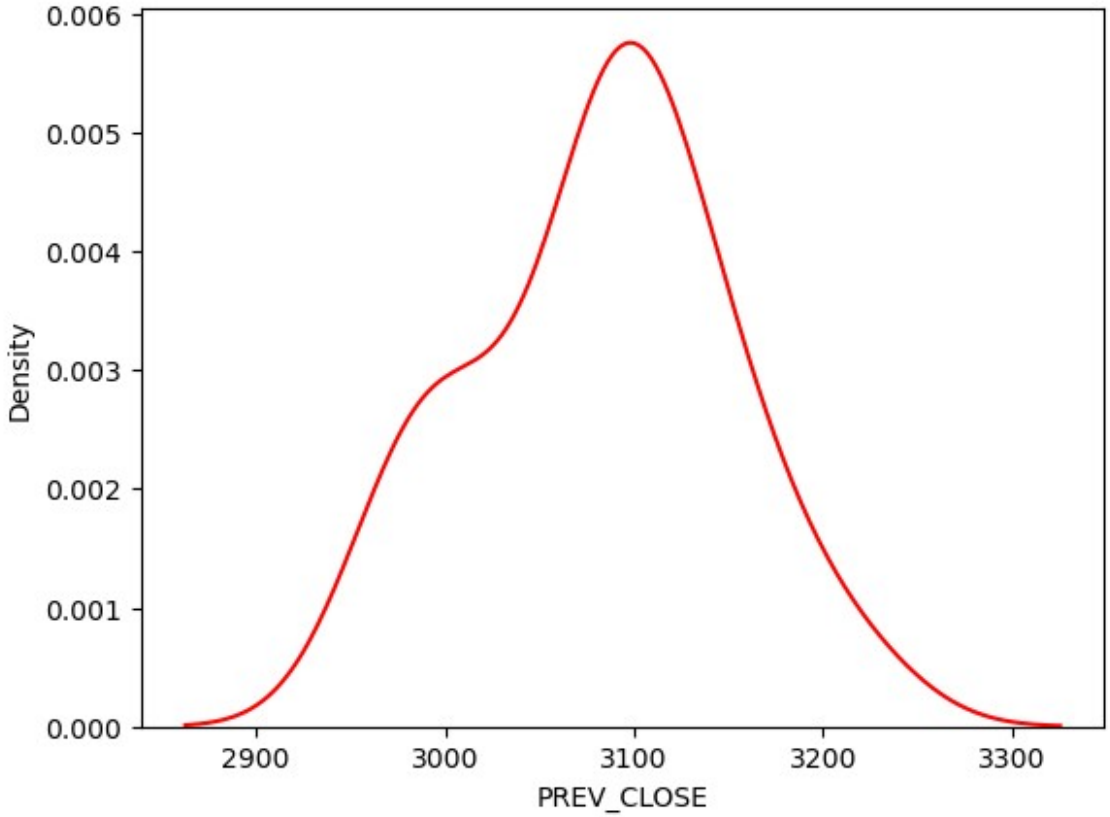


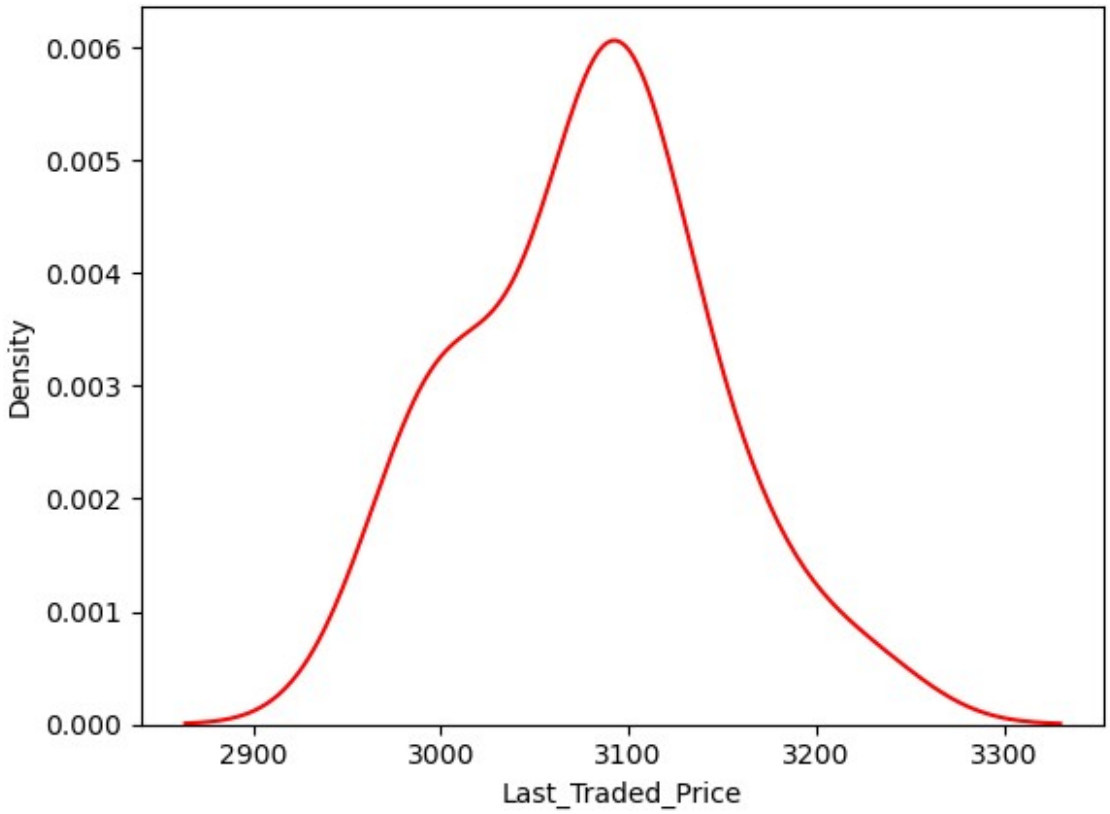

```
#kdeplot
for i in df.columns:
  if df[i].dtype !="object":
   sns.kdeplot(x=df[i],color='red')
   plt.show()
```

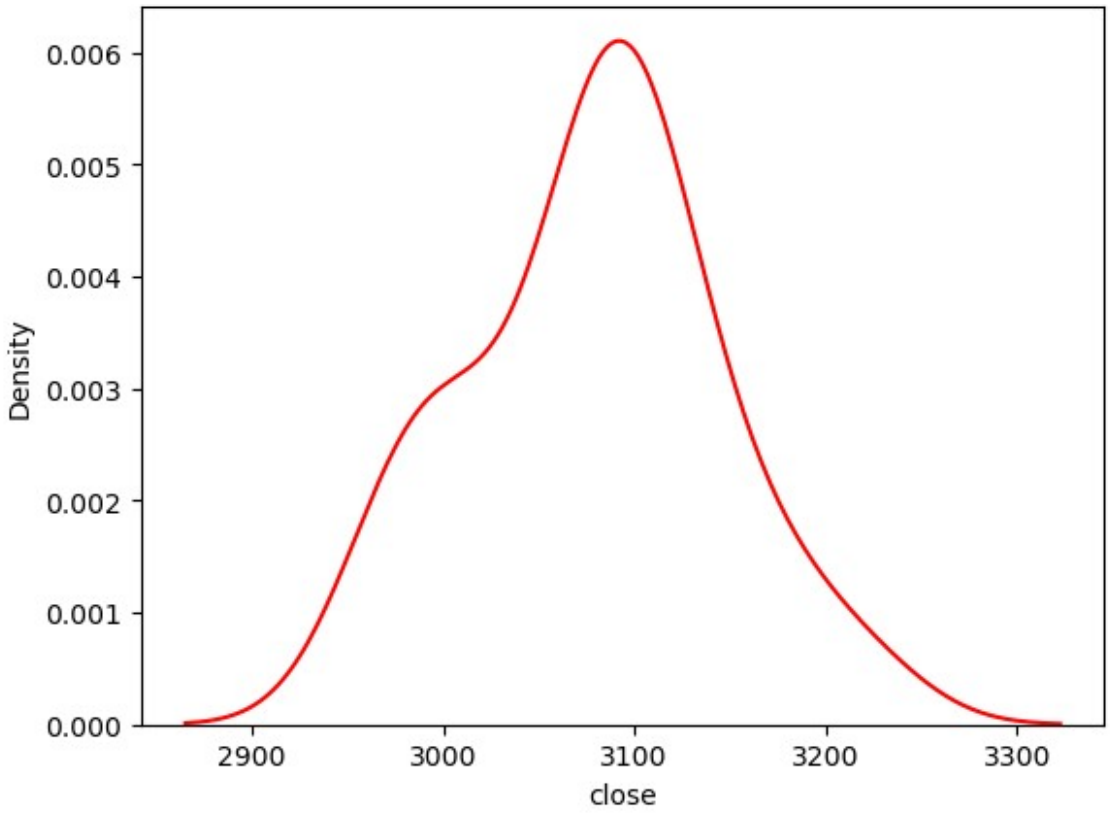

0.007
0.006
0.005
0.004
0.003
0.002
0.001
0.000
Density
2850
2900
2950
3000
3050
3100
3150
3200
3250
OPEN

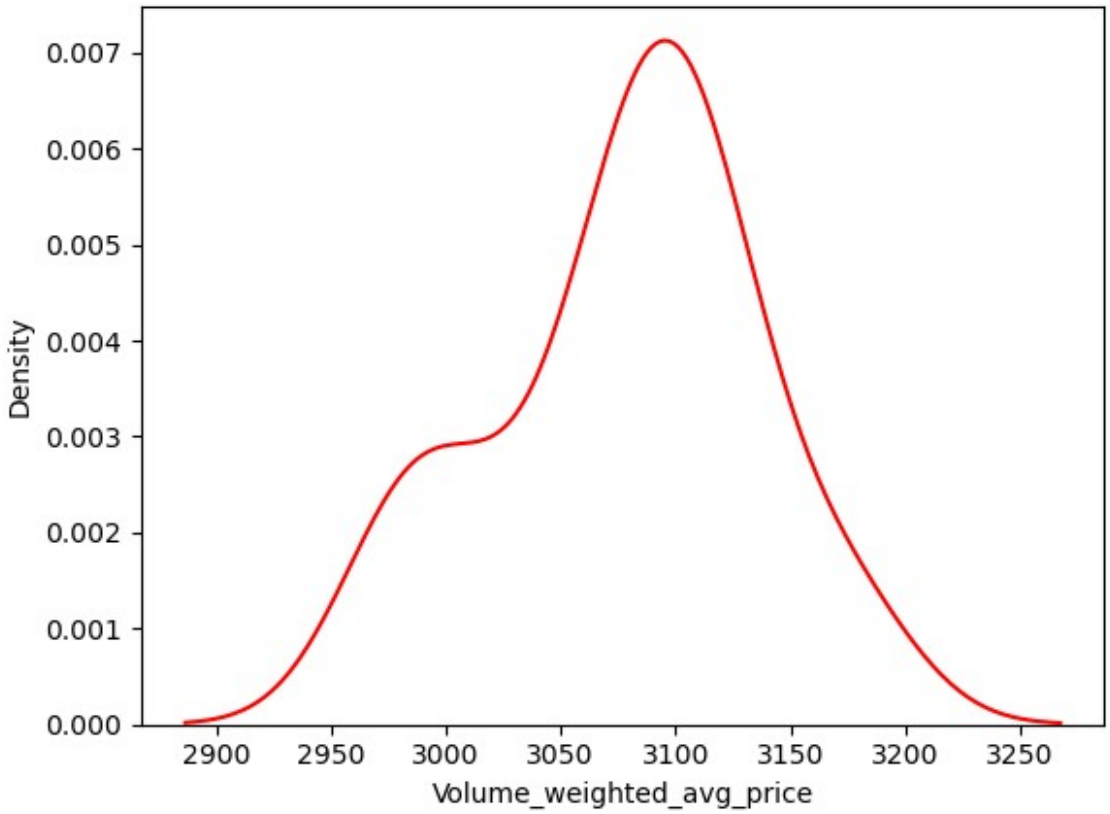

0.007
0.006
0.005
0.004
0.003
0.002
0.001
0.000
Density
2900
2950
3000
3050
3100
3150
3200
3250
3300
HIGH

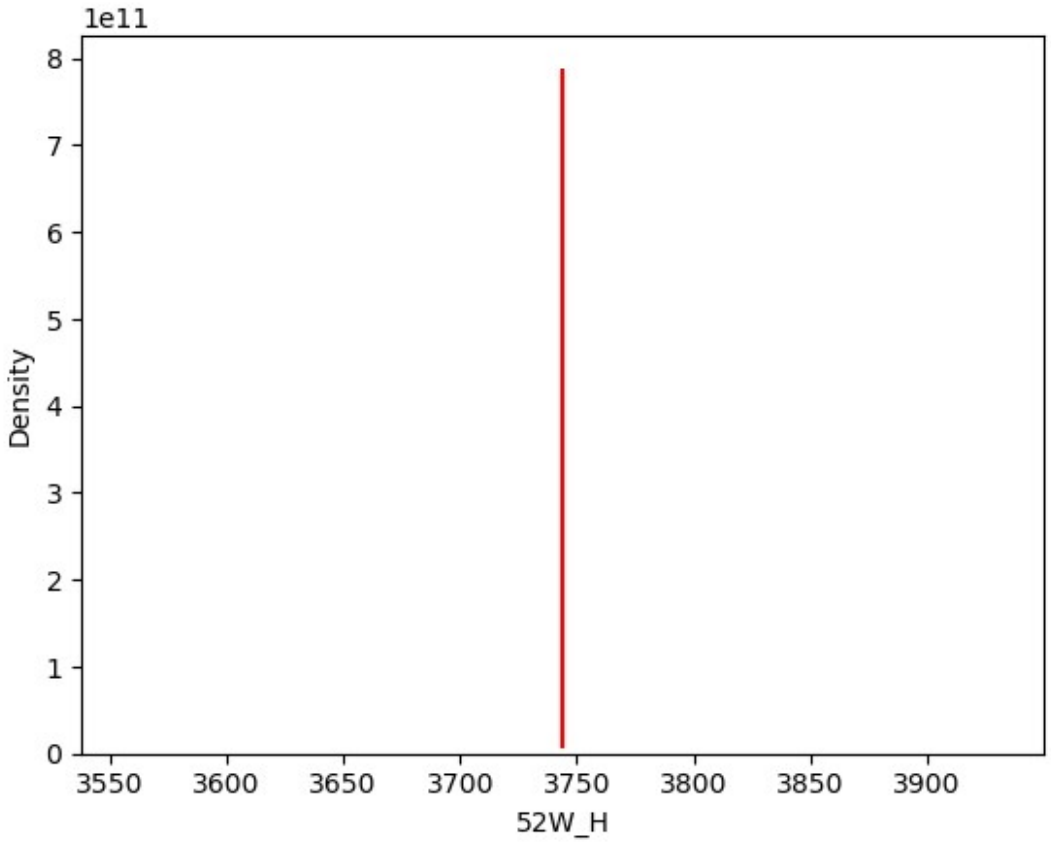

0.006
0.005
0.004
0.003
0.002
0.001
0.000
Density
2800
2900
3000
3100
3200
LOW

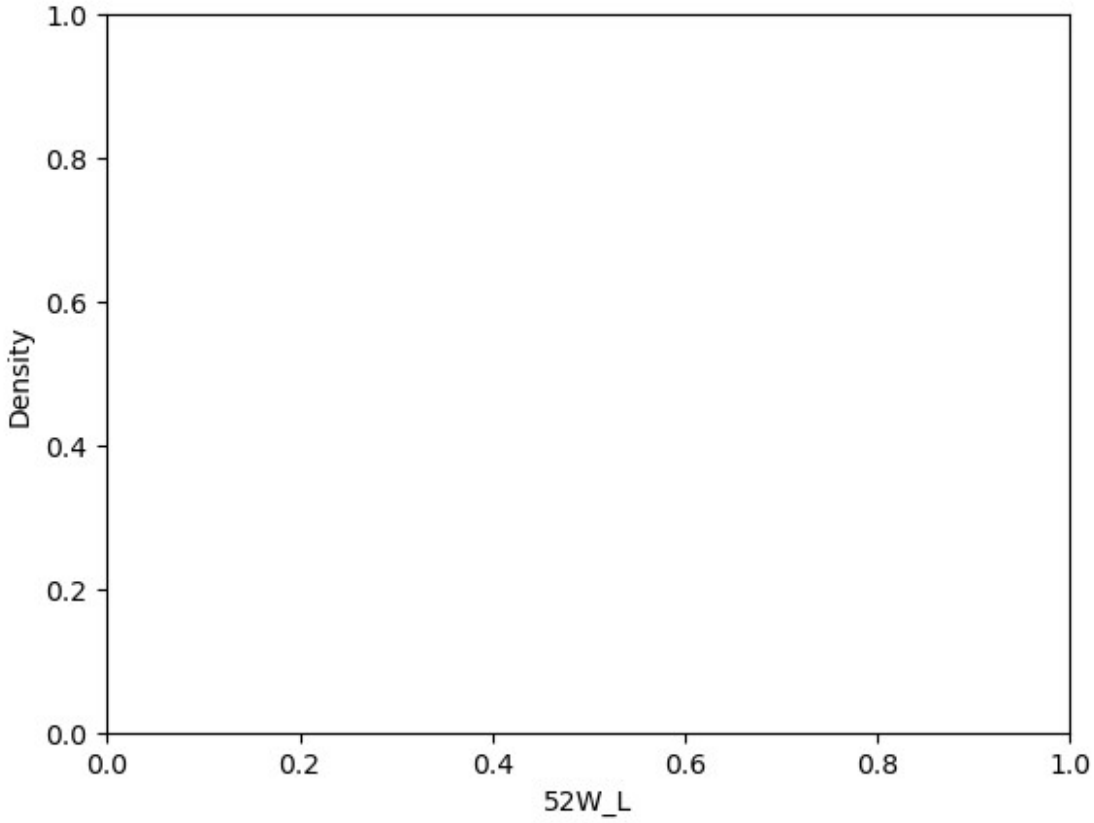

0.006
0.005
0.004
0.003
0.002
0.001
0.000
Density
2900
3000
3100
3200
3300
PREV_CLOSE

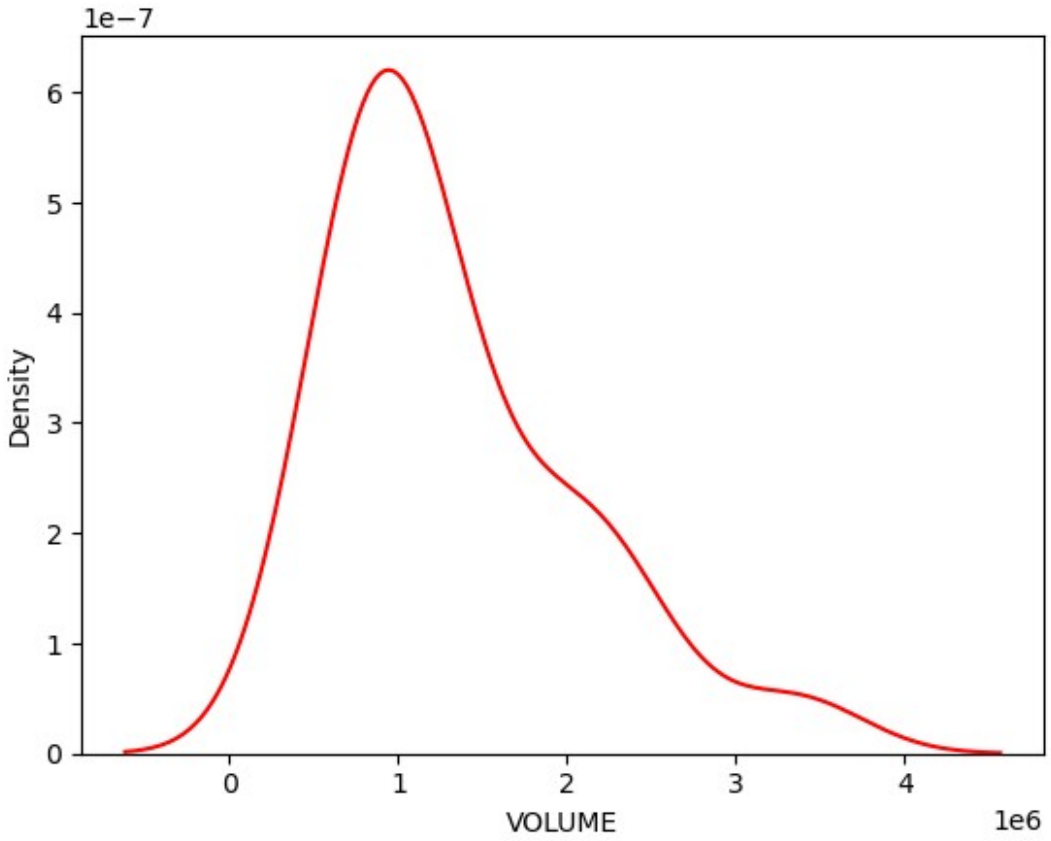

0.006
0.005
0.004
0.003
0.002
0.001
0.000
Density
2900
3000
3100
3200
3300
Last_Traded_Price

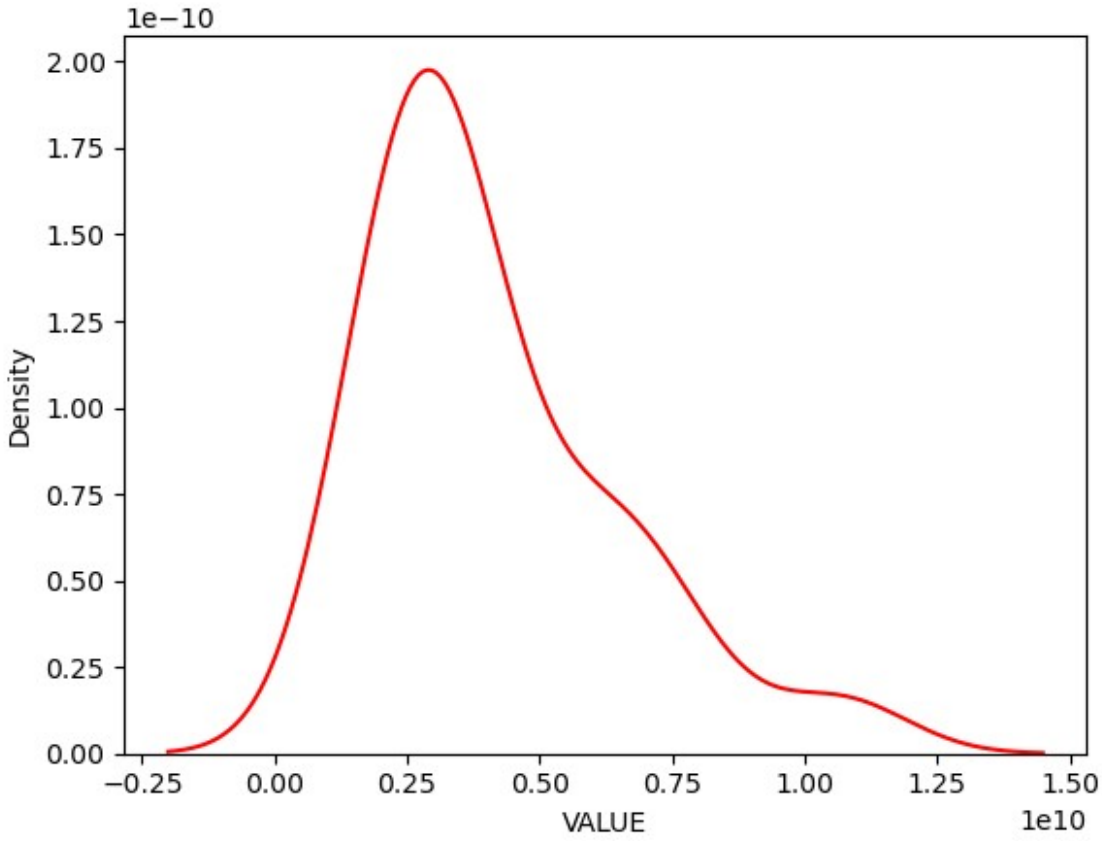

0.006
0.005
0.004
0.003
0.002
0.001
0.000
Density
2900
3000
3100
3200
3300
close

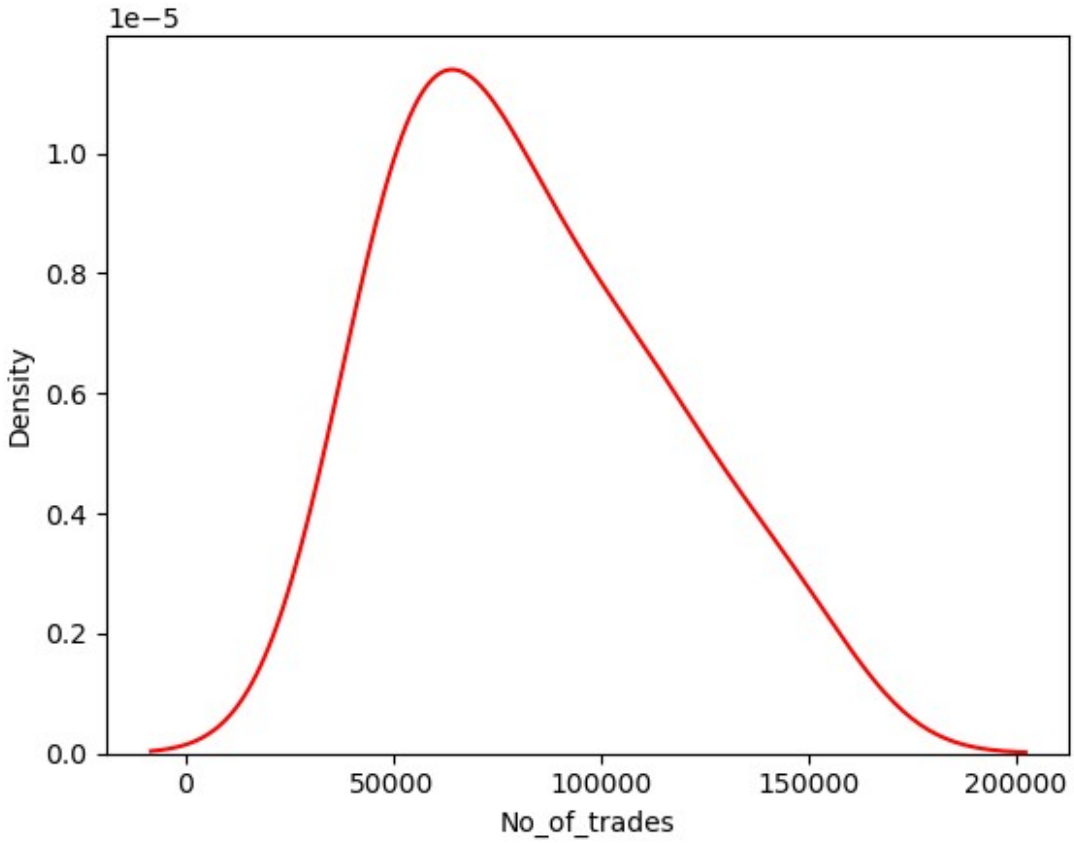

0.007
0.006
0.005
0.004
0.003
0.002
0.001
0.000
Density
2900
2950
3000
3050
3100
3150
3200
3250
Volume_weighted_avg_price


1e11
8
7
6
5
4
3
2
1
0
Density
3550
3600
3650
3700
3750
3800
3850
3900
52W_H

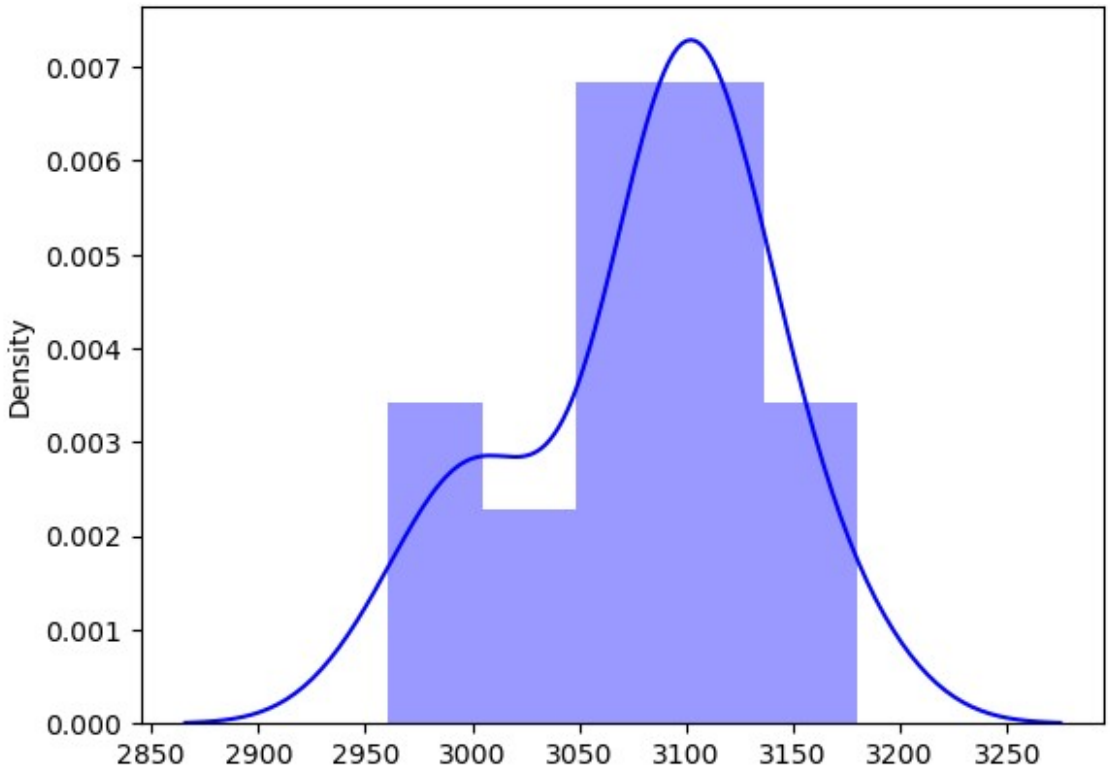

1.0
0.8
0.6
0.4
0.2
0.0
Density
0.0
0.2
0.4
0.6
0.8
1.0
52W_L

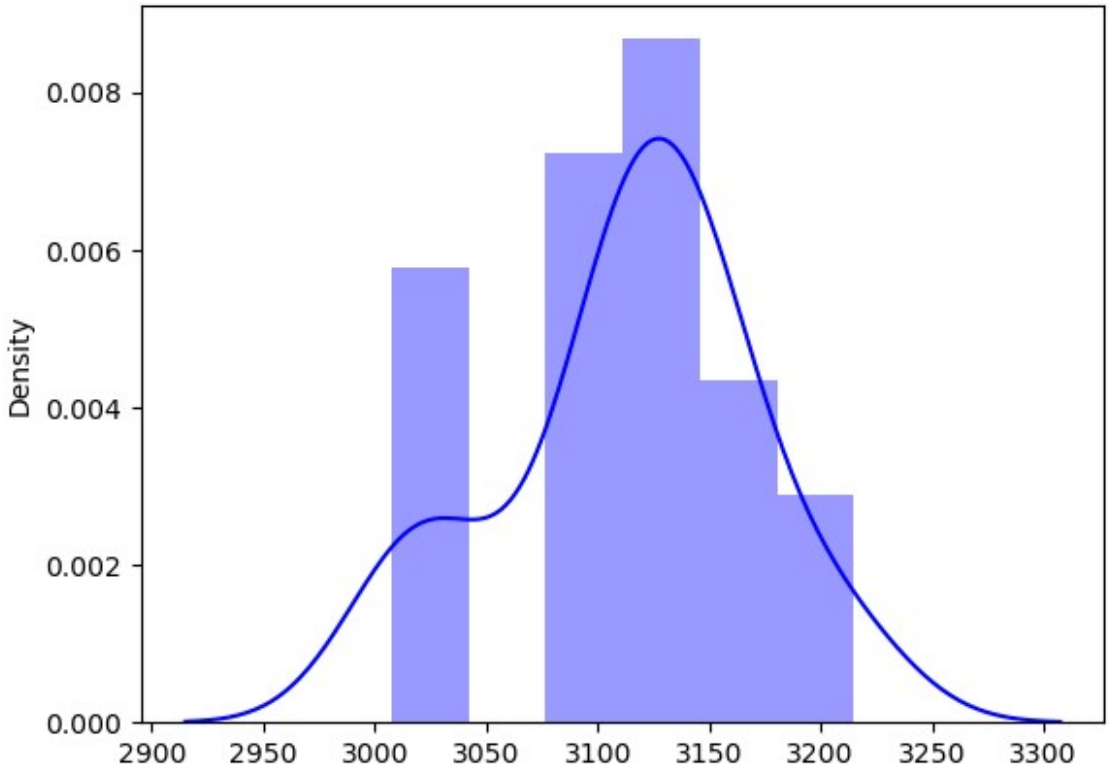

1e−7
6
5
4
3
2
1
0
Density
0
1
2
3
4
VOLUME
1e6

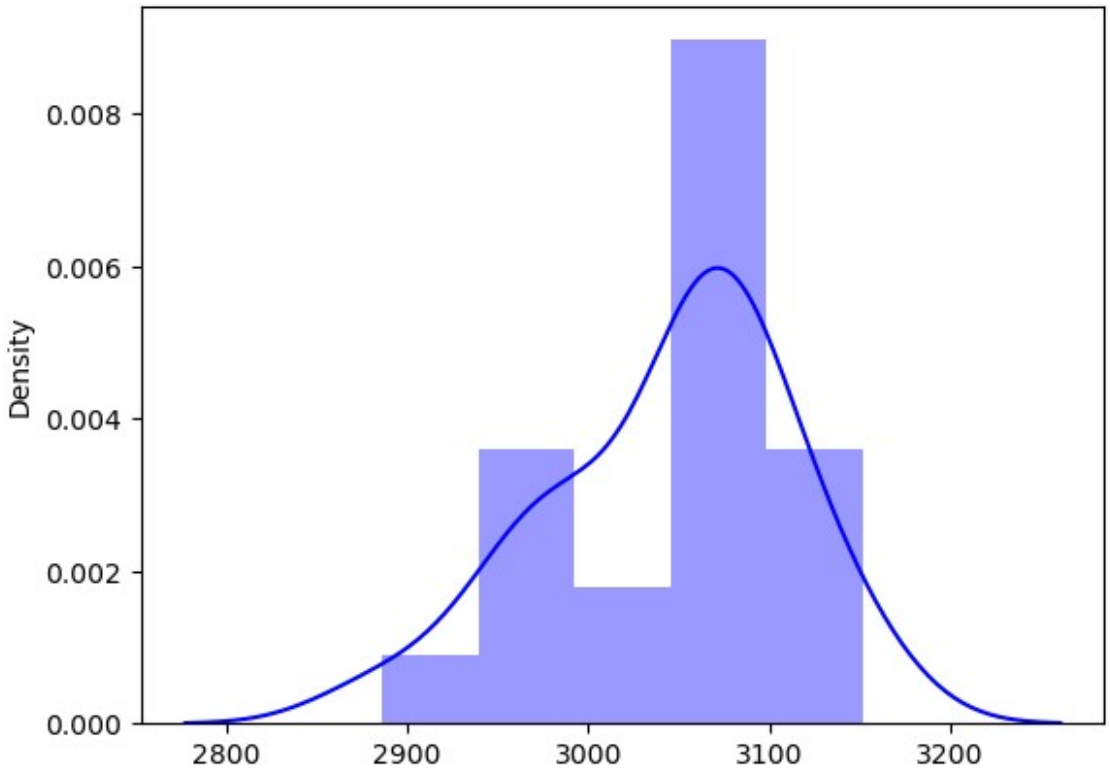

1e−10
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Density
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
1.50
VALUE
1e10

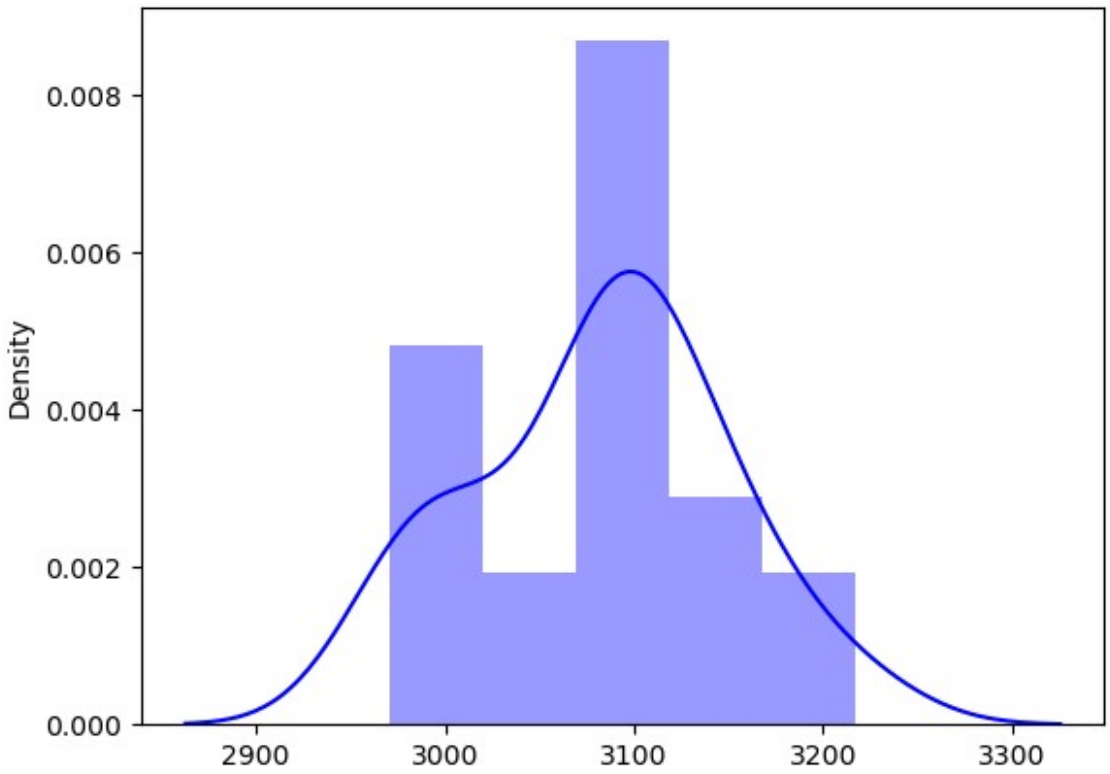


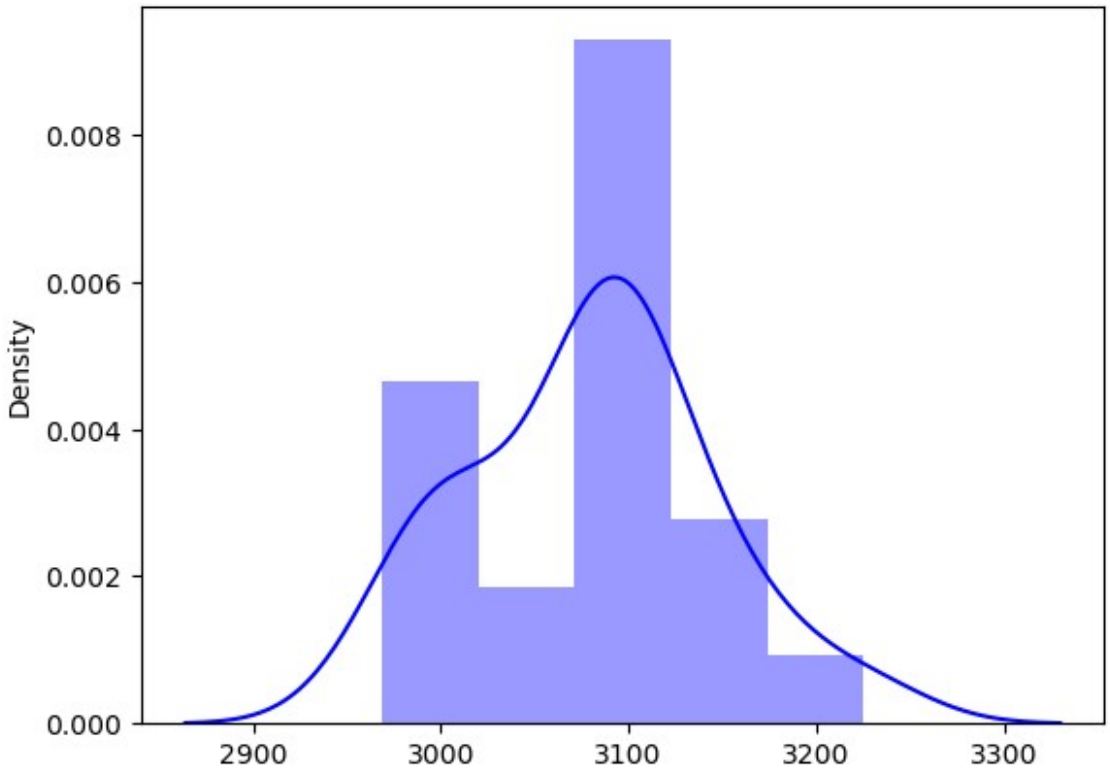

```
#Distplot
for i in df.columns:
   if df[i].dtypes != "object":
    sns.distplot(x =df[i],color='blue' )
    plt.show()
```

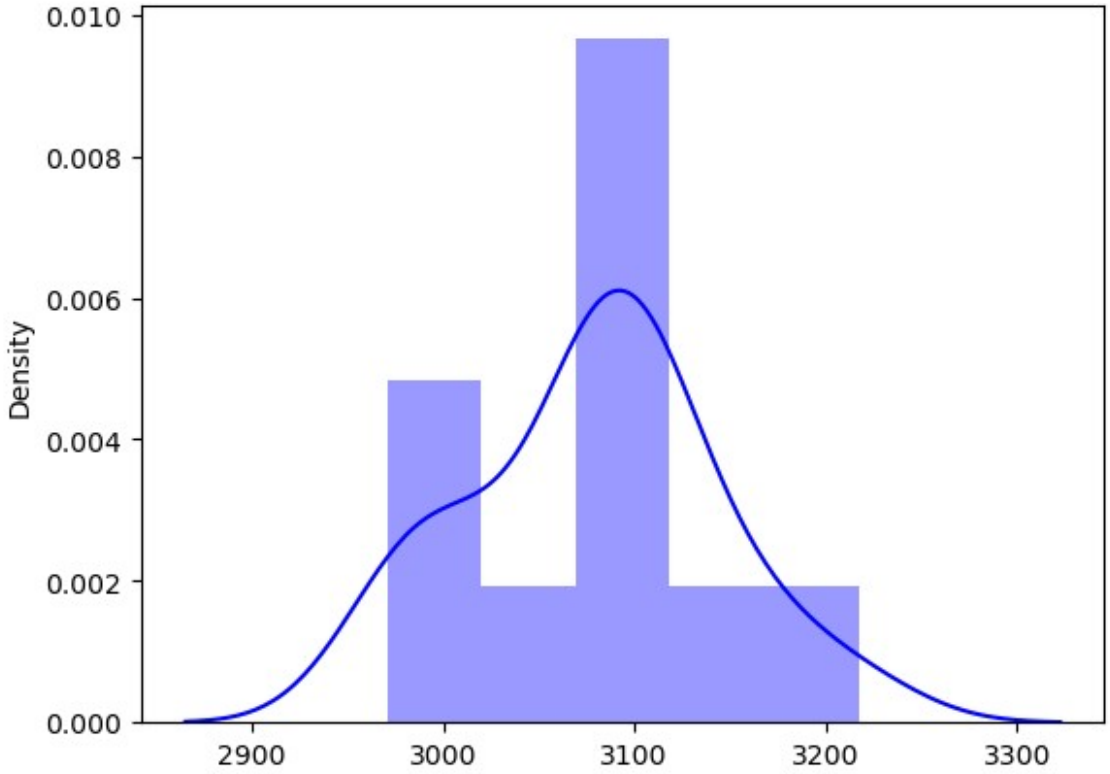

Density
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0.000
2850
2900
2950
3000
3050
3100
3150
3200
3250

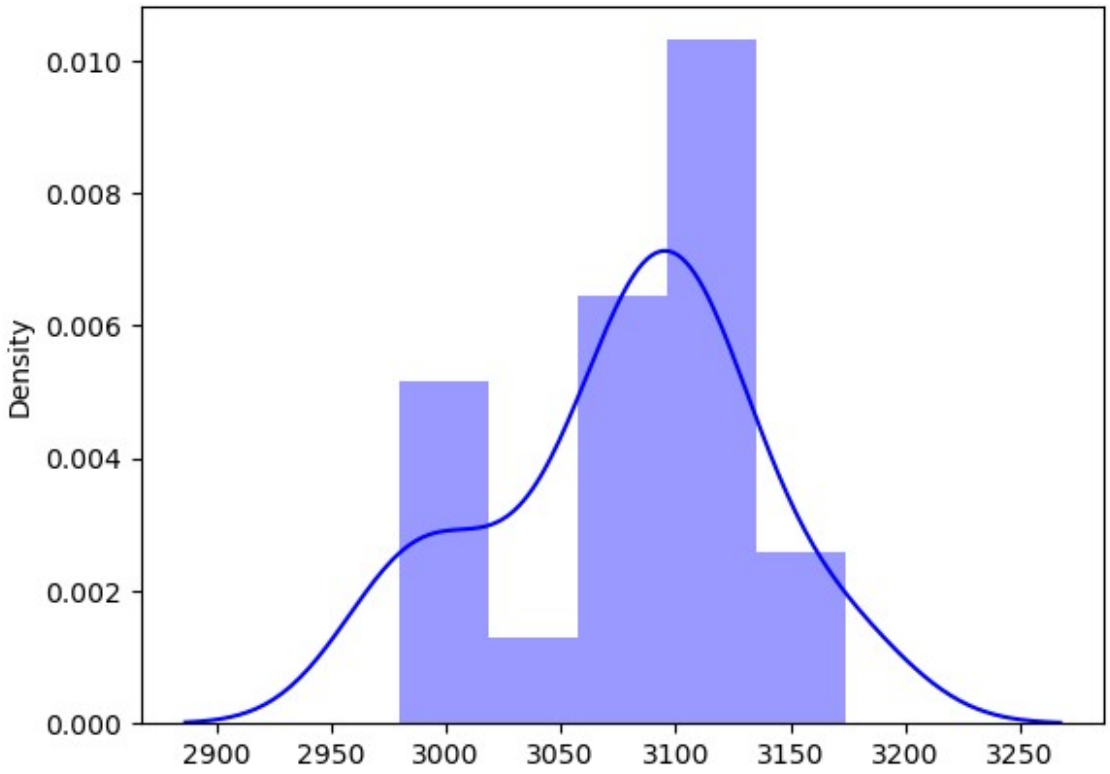

Density
0.008
0.006
0.004
0.002
0.000
2900
2950
3000
3050
3100
3150
3200
3250
3300

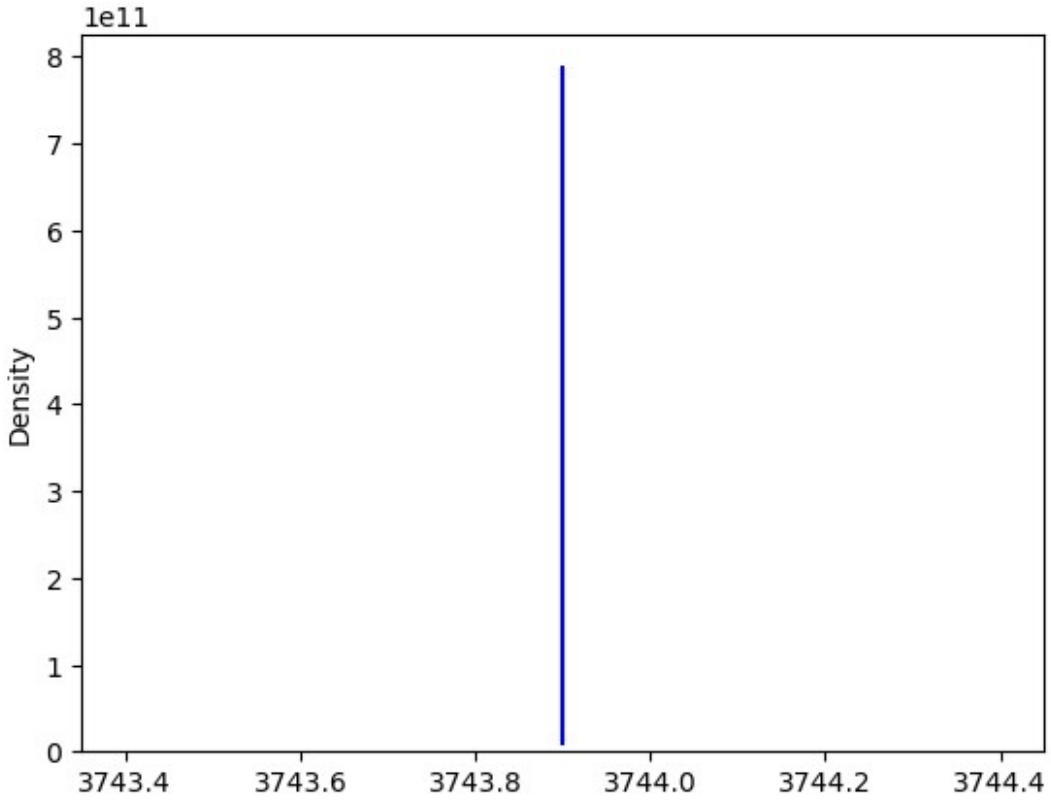

0.008
0.006
0.004
0.002
0.000
Density
2800
2900
3000
3100
3200

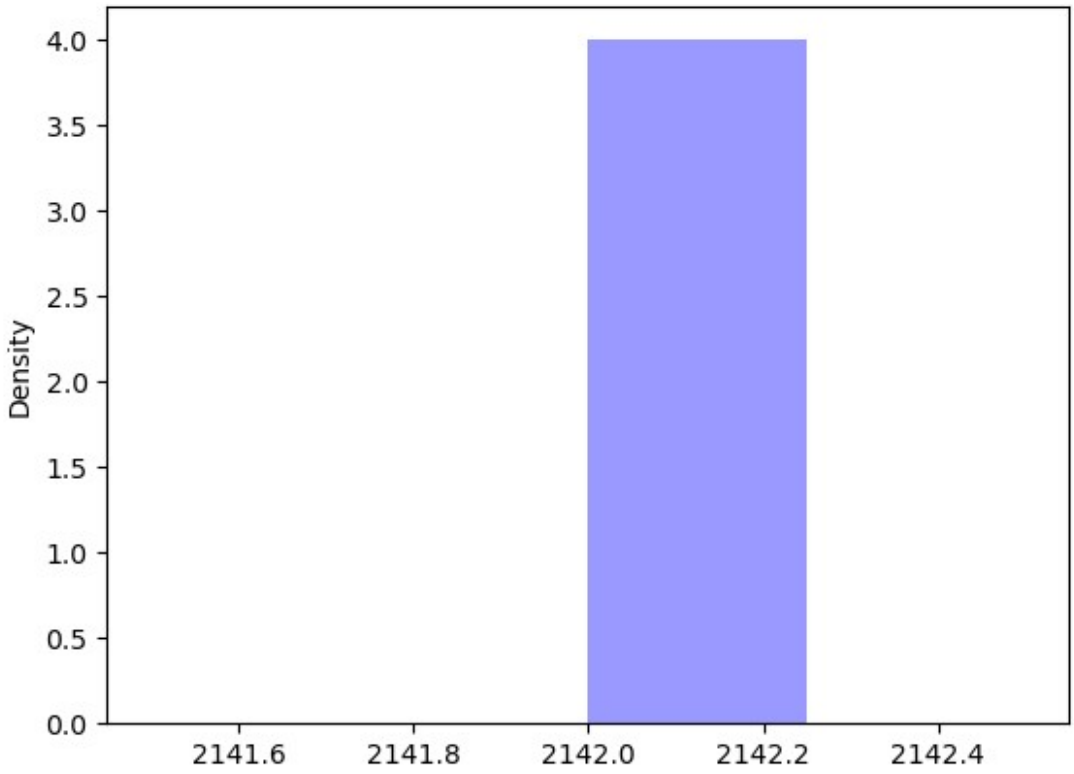

0.008
0.006
0.004
0.002
0.000
Density
2900
3000
3100
3200
3300

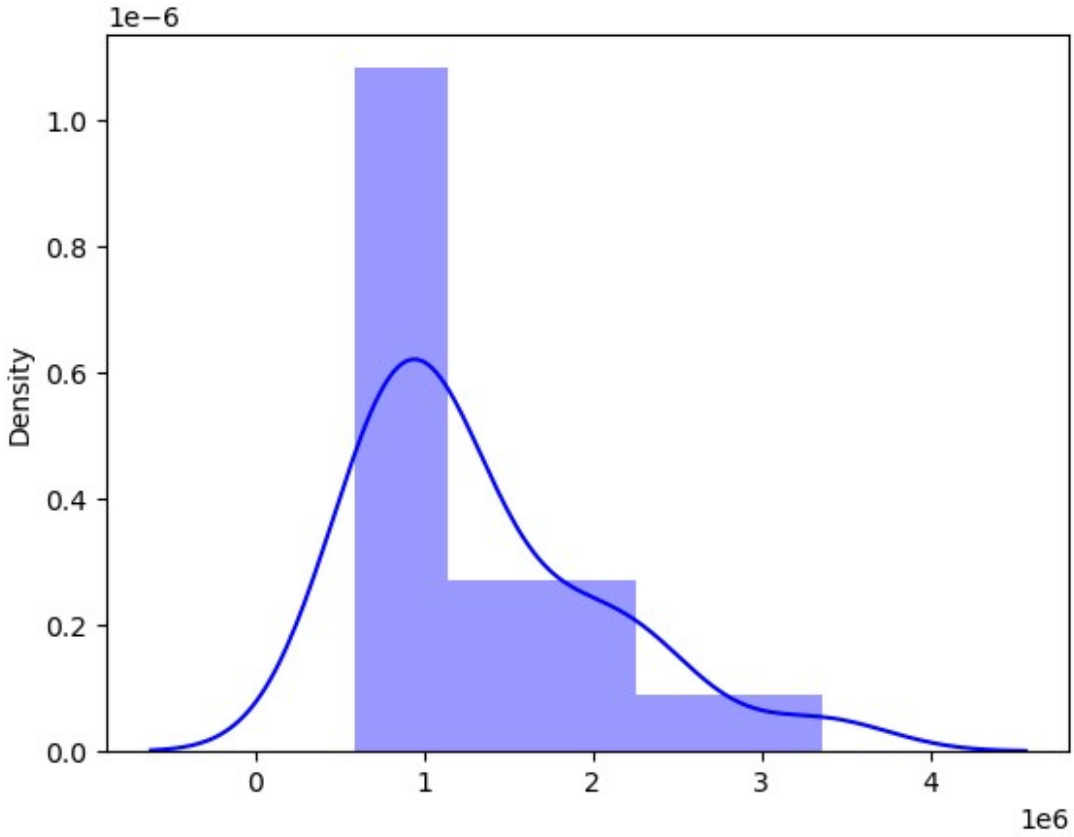

Density
0.008
0.006
0.004
0.002
0.000
2900
3000
3100
3200
3300

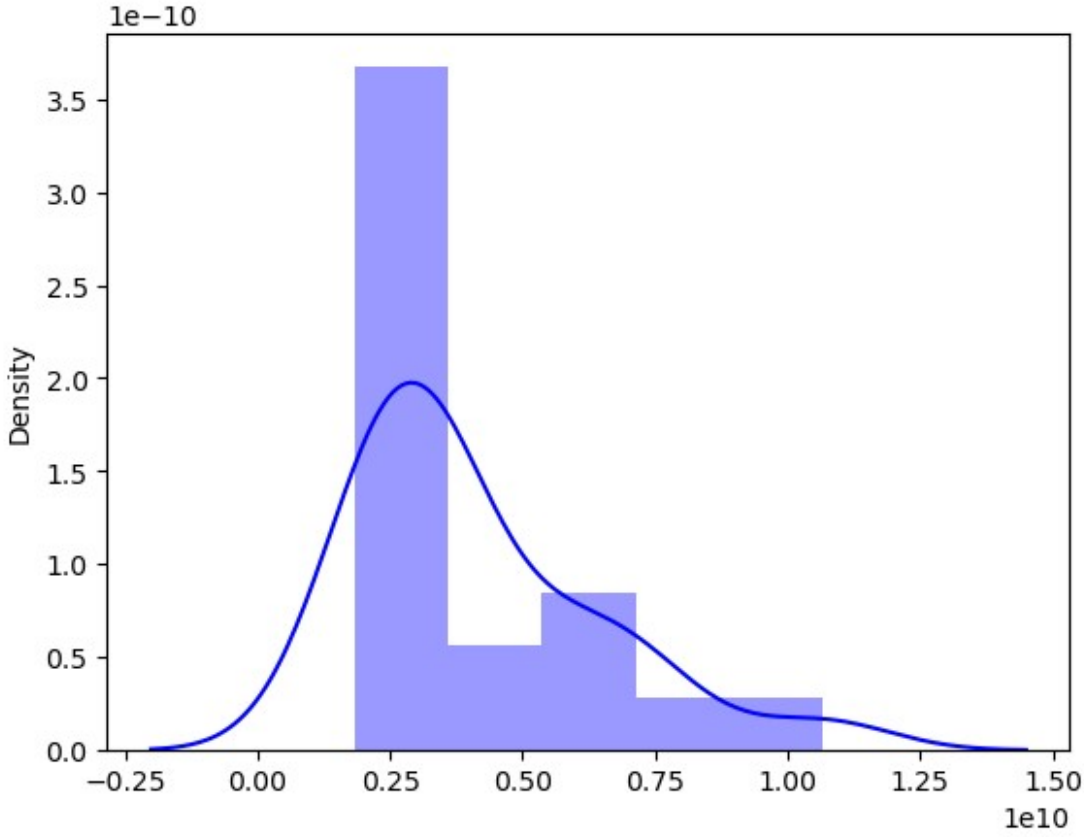

Density
0.010
0.008
0.006
0.004
0.002
0.000
2900
3000
3100
3200
3300

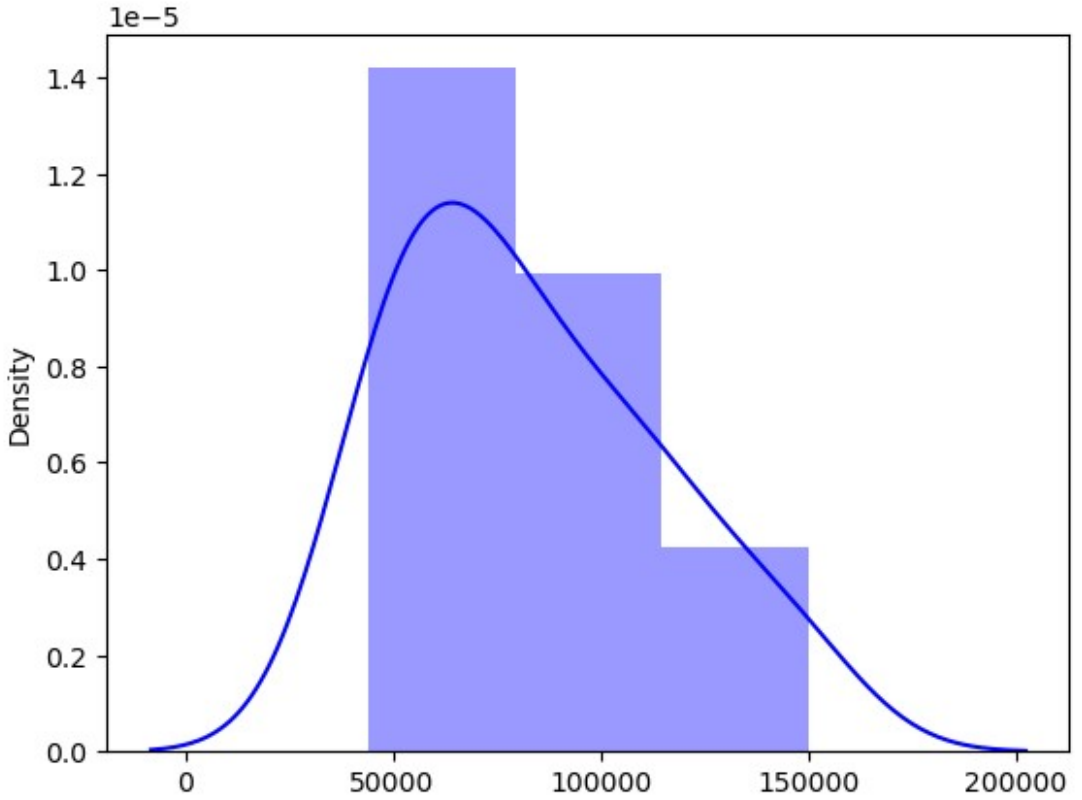

0.010
0.008
0.006
0.004
0.002
0.000
Density
2900
2950
3000
3050
3100
3150
3200
3250


1e11
8
7
6
5
4
3
2
1
0
Density
3743.4
3743.6
3743.8
3744.0
3744.2
3744.4

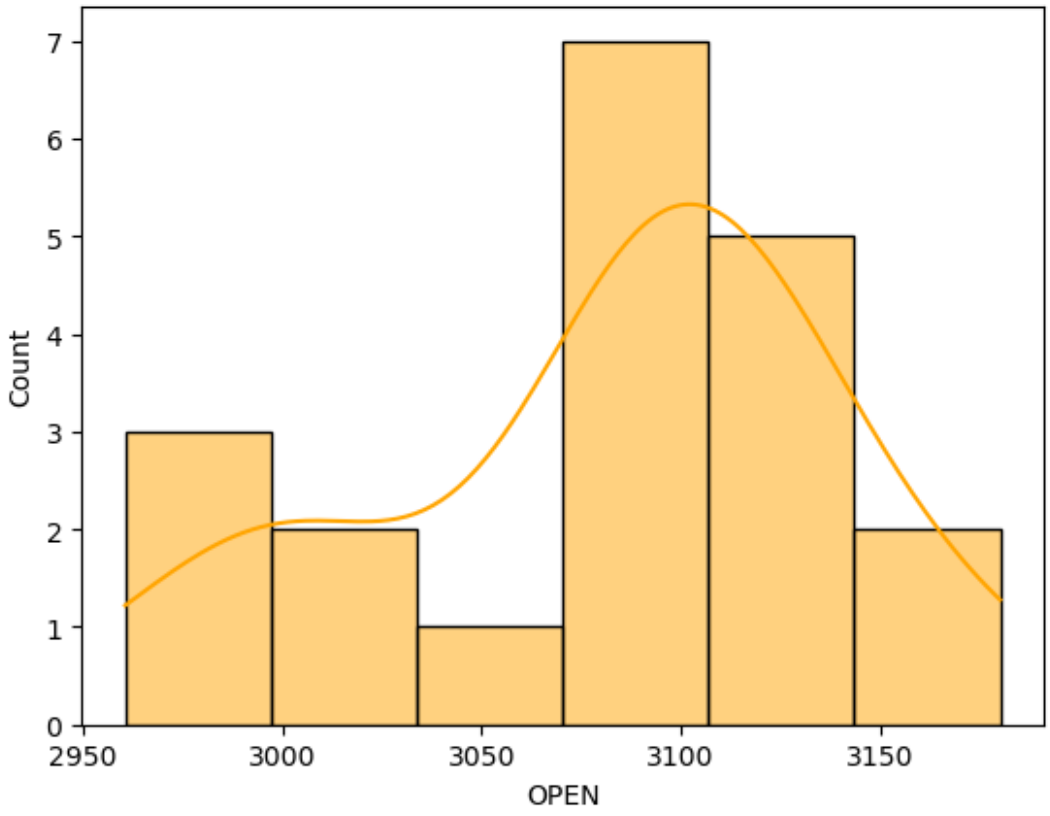

Density
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
2141.6
2141.8
2142.0
2142.2
2142.4

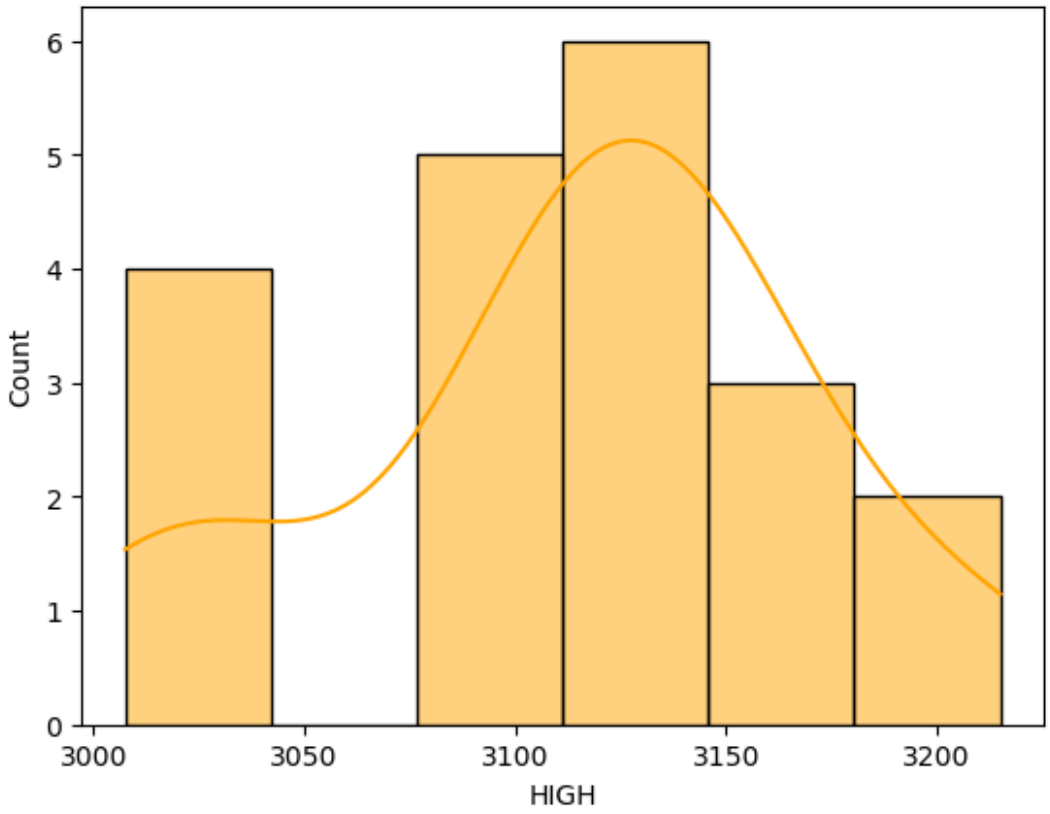

1e−6
Density
1.0
0.8
0.6
0.4
0.2
0.0
0
1
2
3
4
1e6

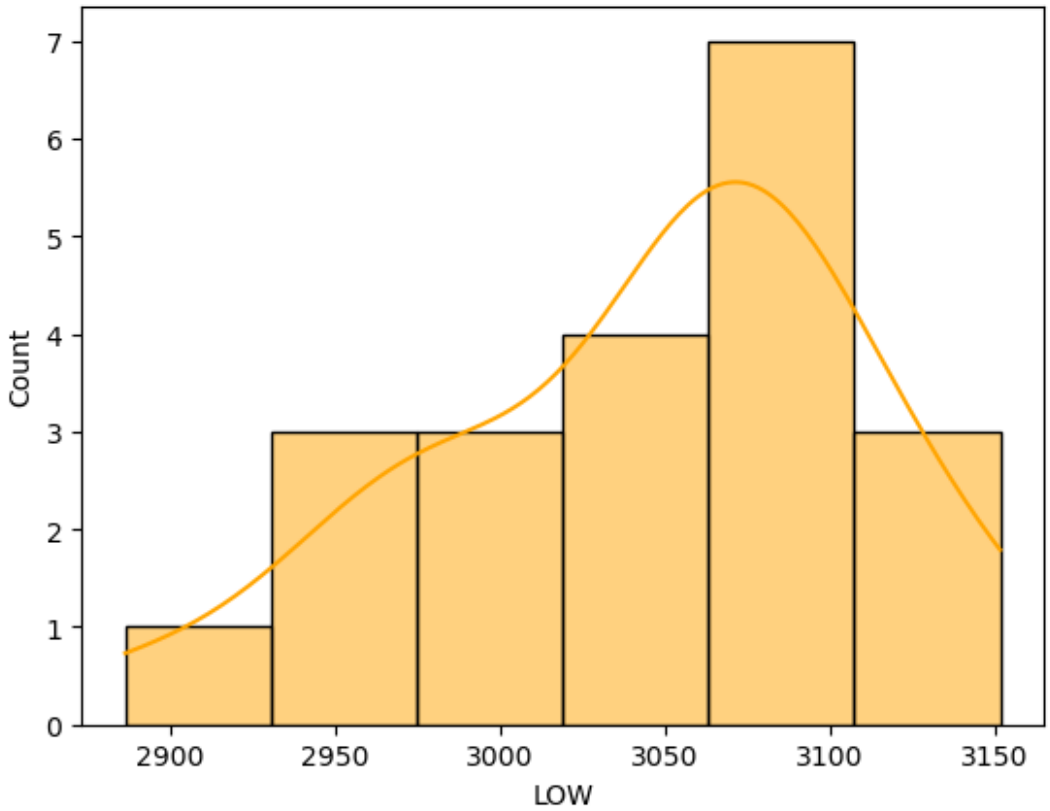

1e−10
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Density
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1e10

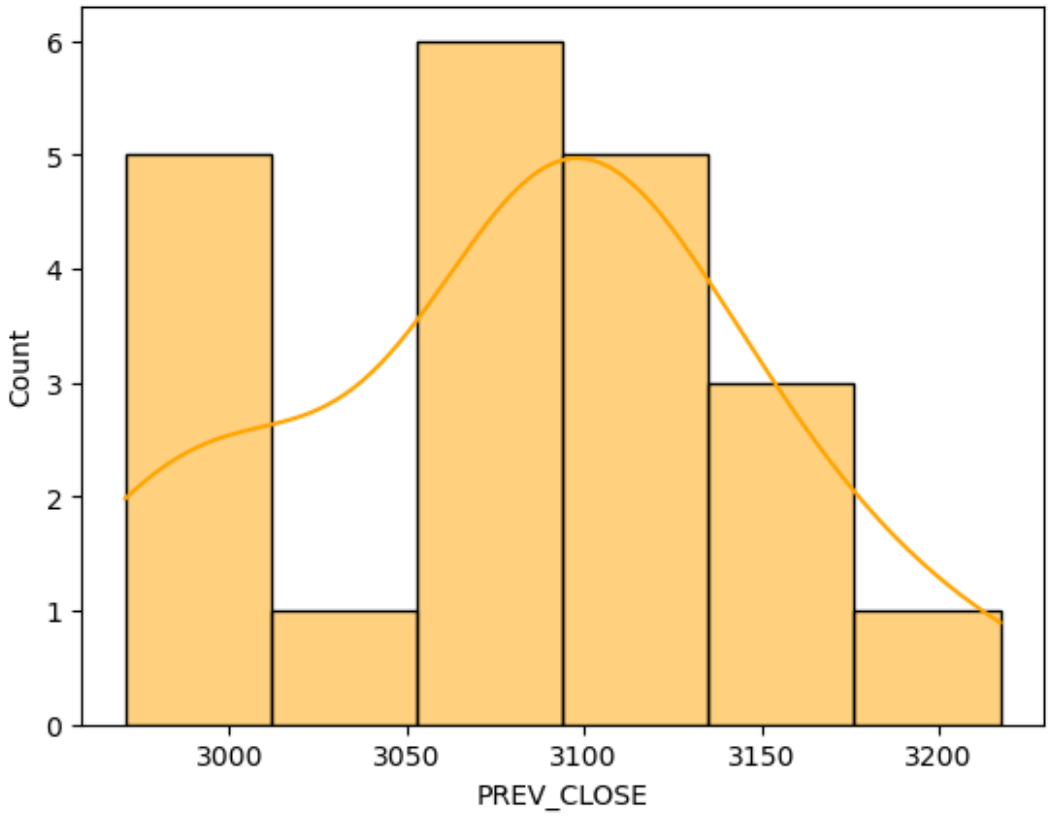

1e−5
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Density
0
50000
100000
150000
200000

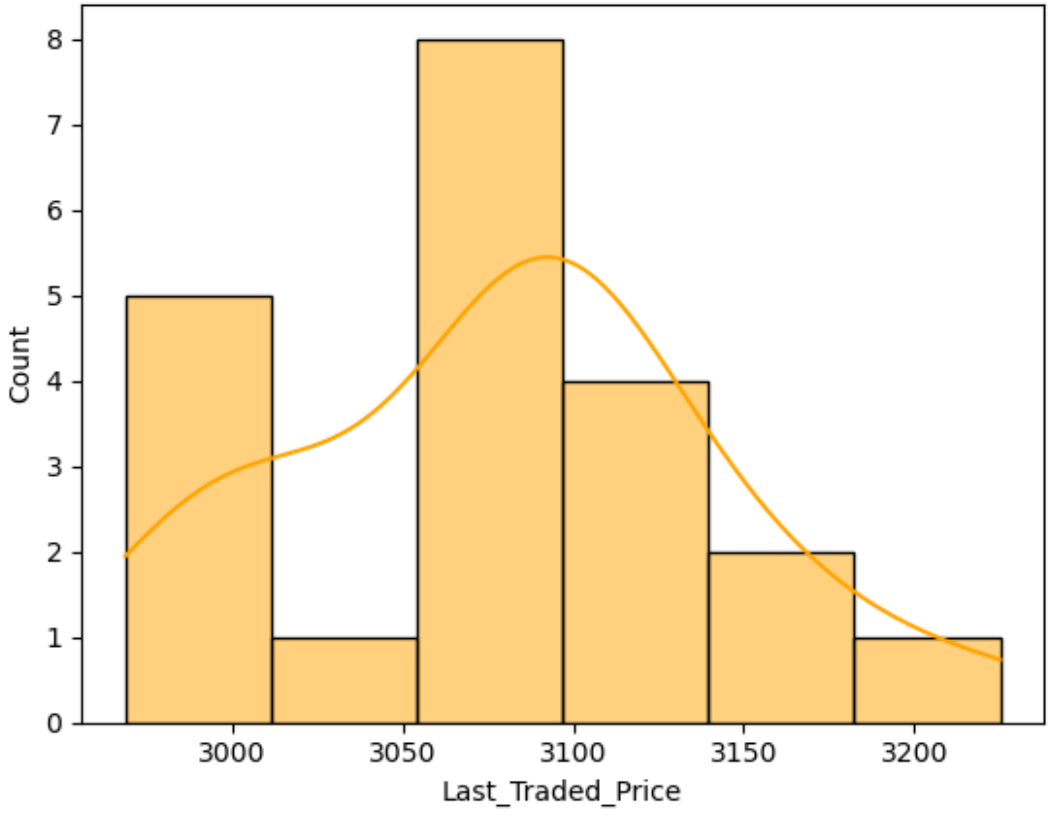
```
#Histplot
for column in df.columns:
  if df[column].dtypes !='object':
    sns.histplot(x=df[column],kde=True,color='orange')
    plt.show()
```

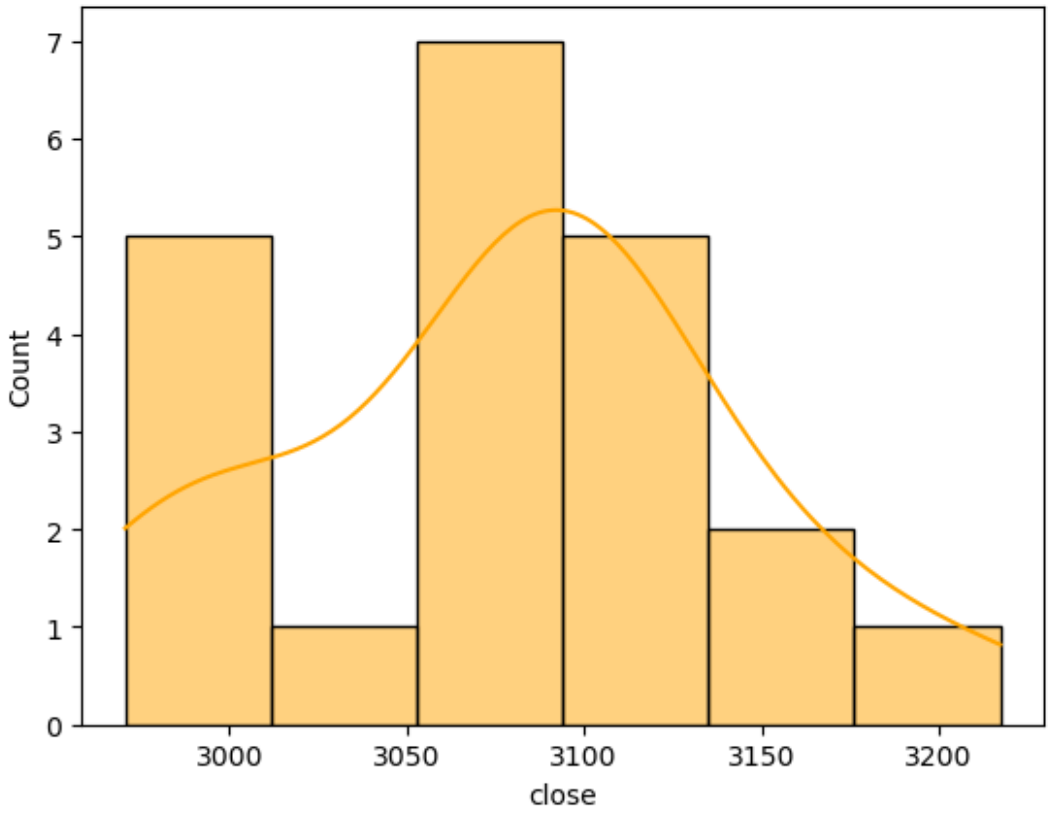


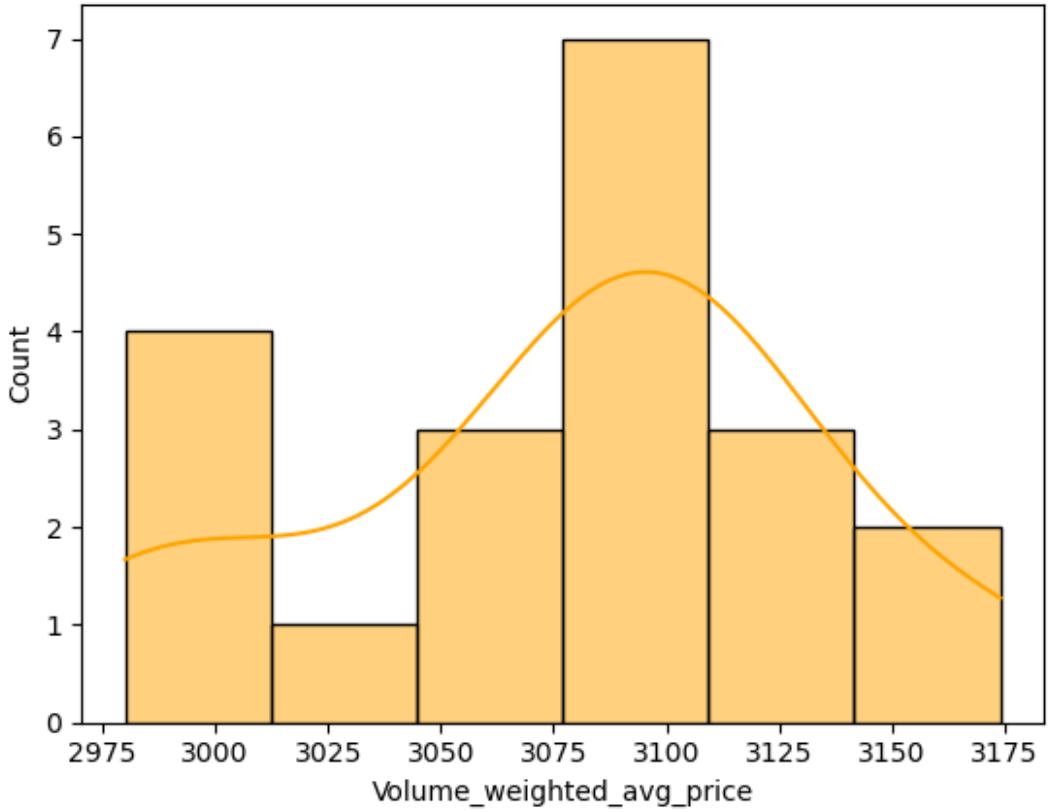

Count
HIGH
0
1
2
3
4
5
6
3000
3050
3100
3150
3200

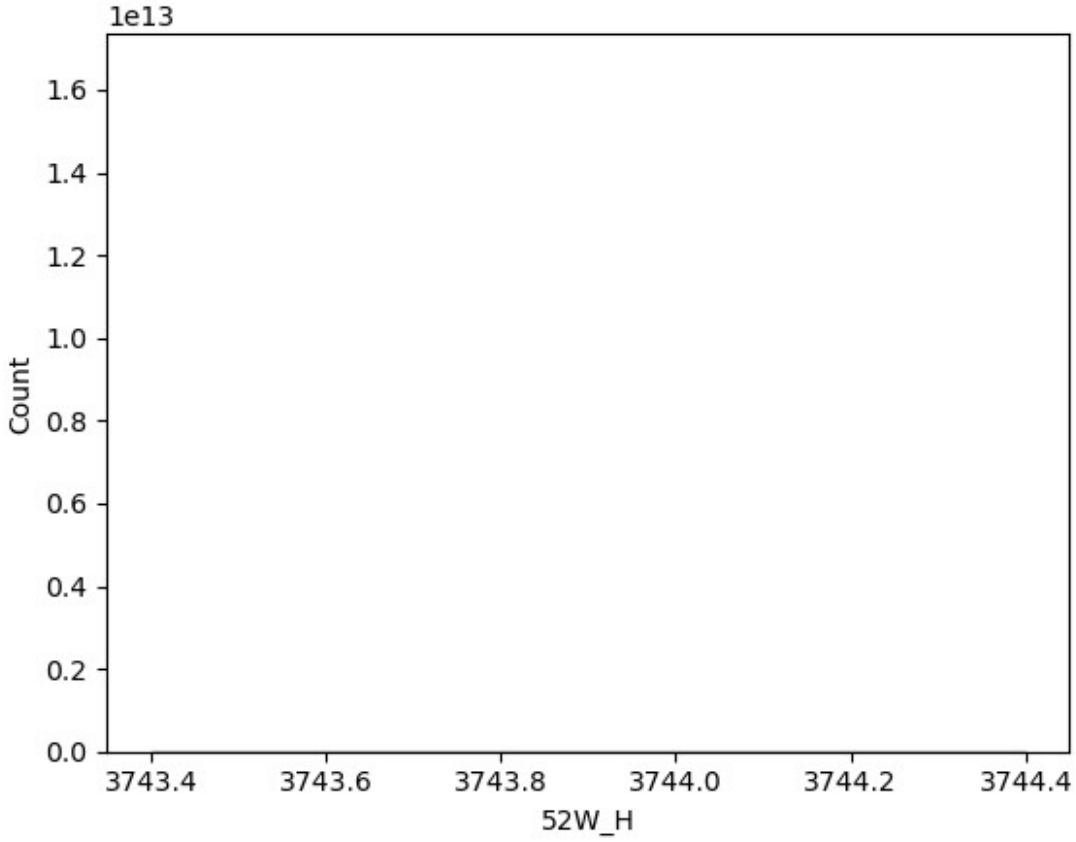

Count
LOW
0
1
2
3
4
5
6
7
2900
2950
3000
3050
3100
3150

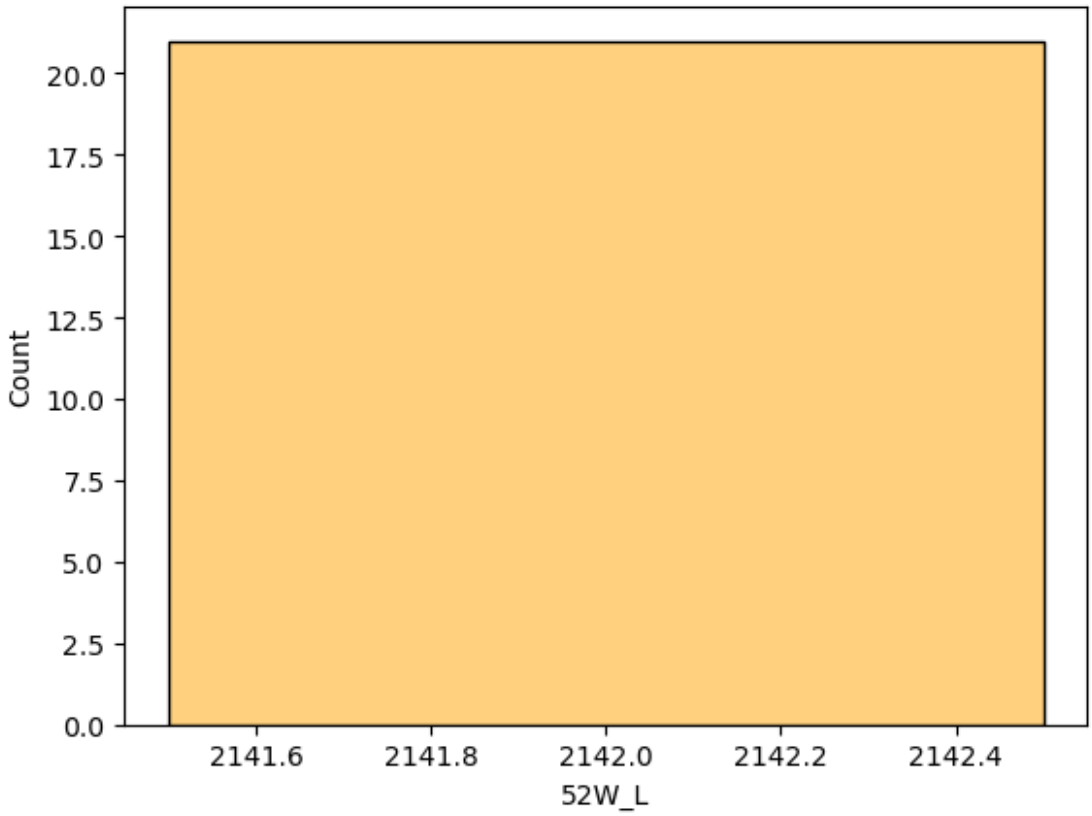

Count
PREV_CLOSE
0
1
2
3
4
5
6
3000
3050
3100
3150
3200

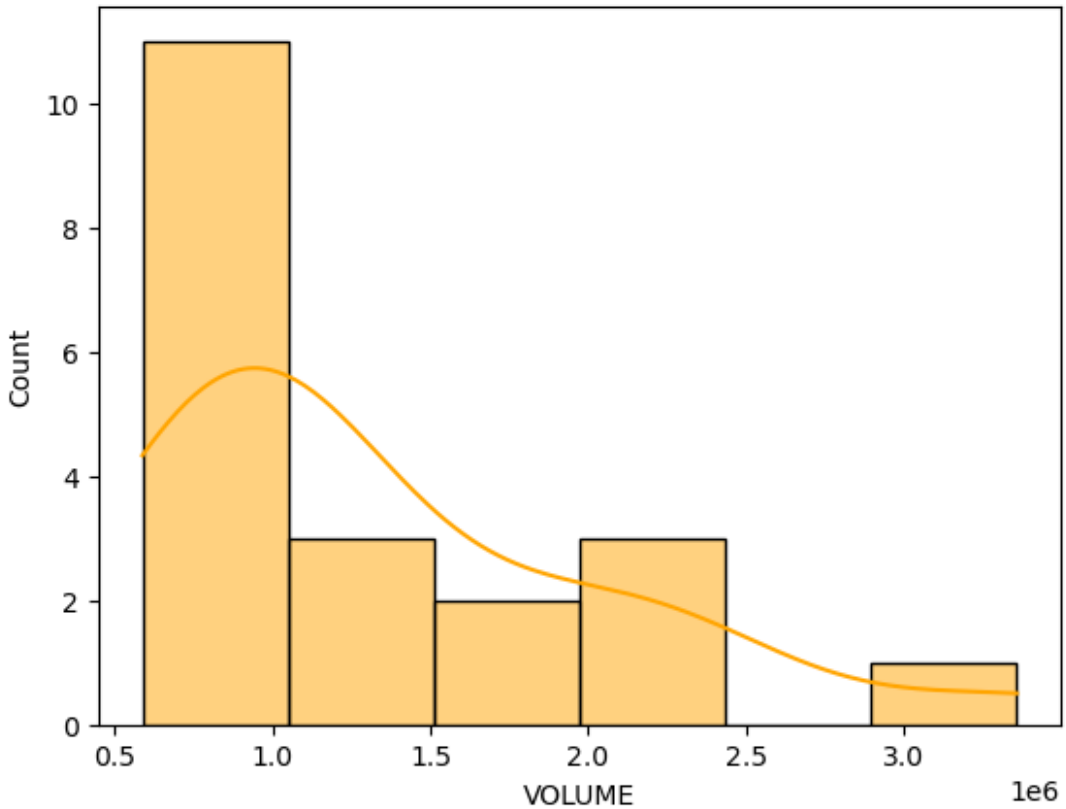

Count
Last_Traded_Price
0
1
2
3
4
5
6
7
8
3000
3050
3100
3150
3200

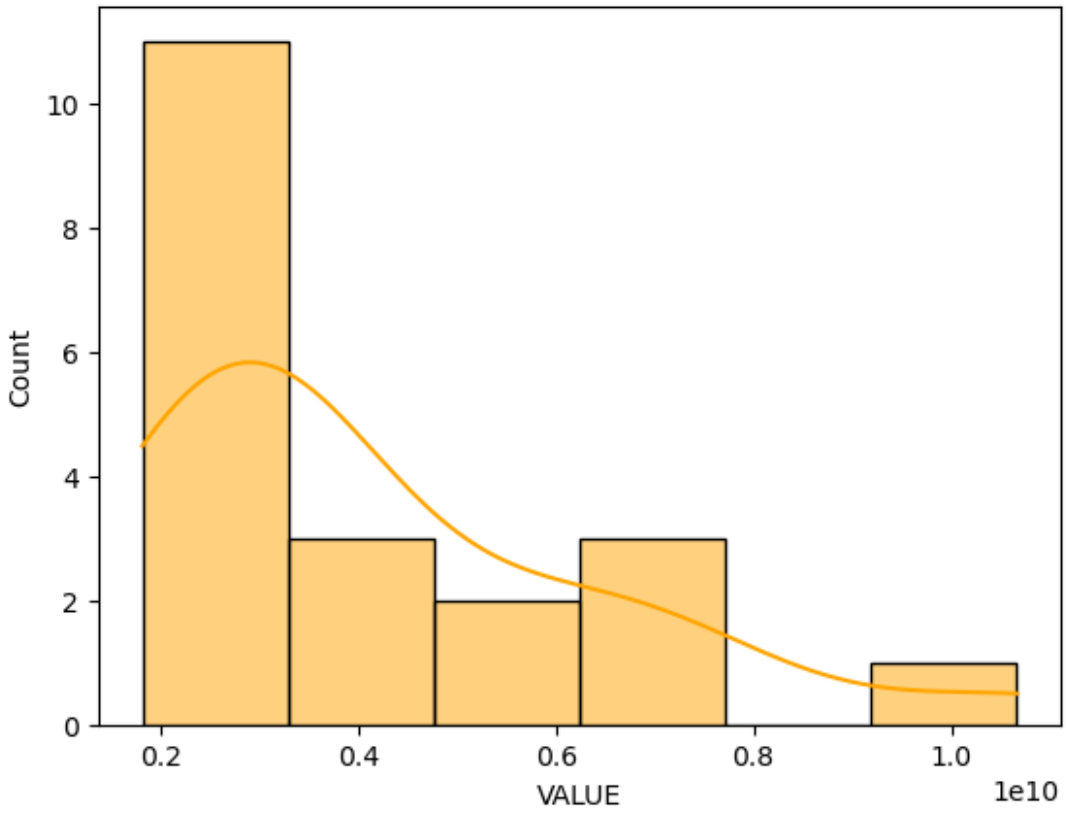

Count
close
0
1
2
3
4
5
6
7
3000
3050
3100
3150
3200

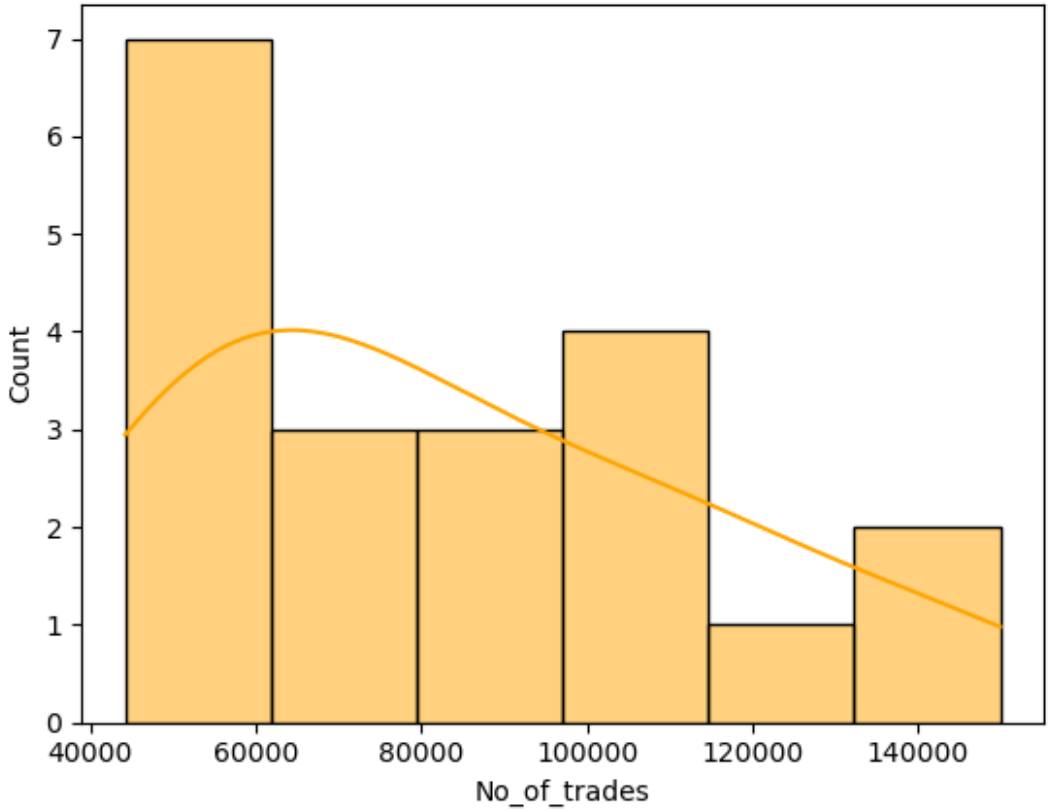

Count
Volume_weighted_avg_price
0
1
2
3
4
5
6
7
2975
3000
3025
3050
3075
3100
3125
3150
3175


1e13
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Count
3743.4
3743.6
3743.8
3744.0
3744.2
3744.4
52W_H

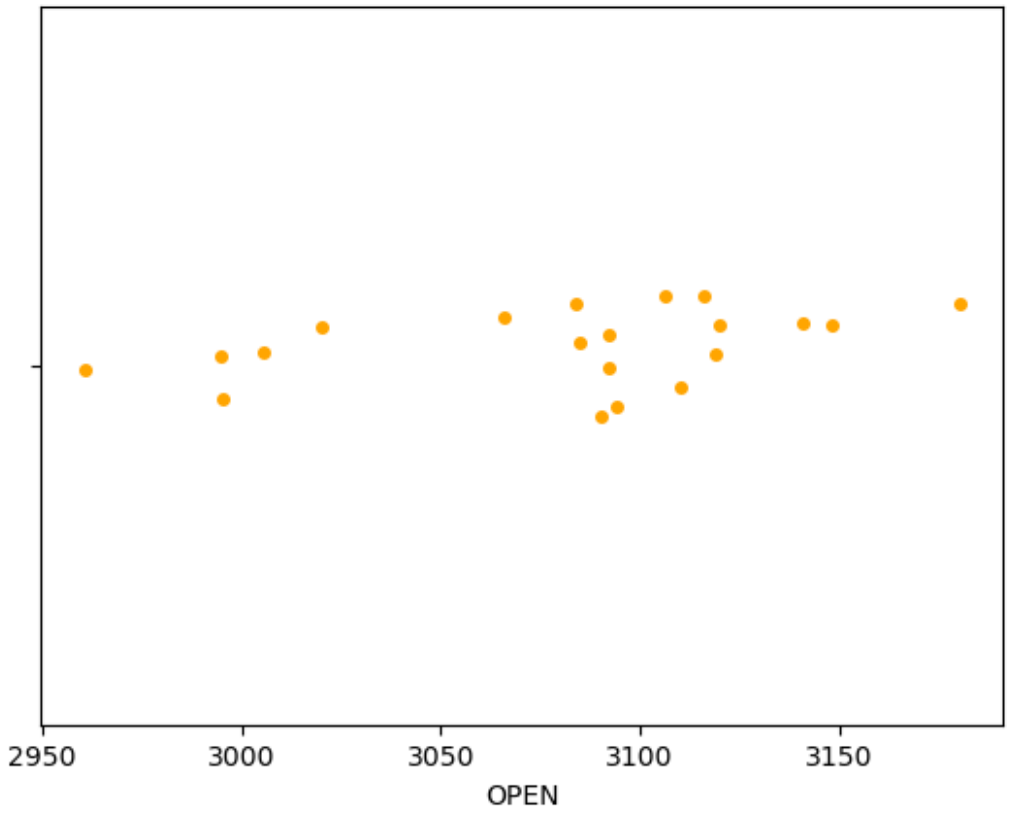

Count
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
2141.6
2141.8
2142.0
2142.2
2142.4
52W_L

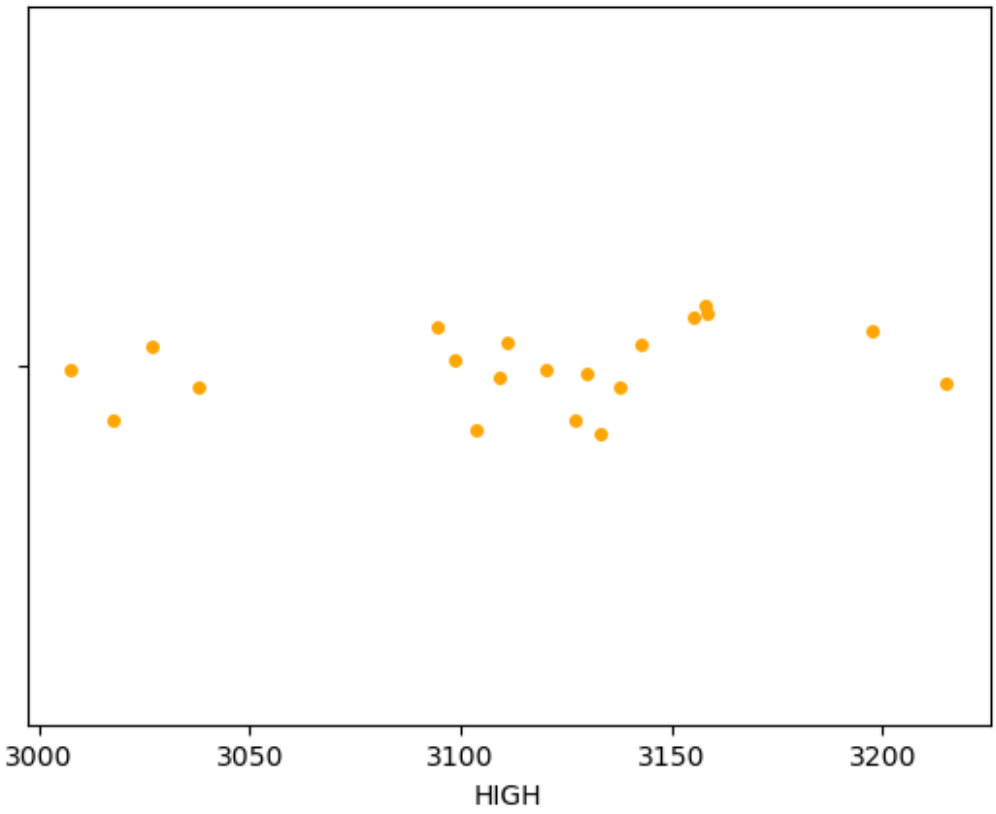

Count
10
8
6
4
2
0
0.5
1.0
1.5
2.0
2.5
3.0
VOLUME
1e6

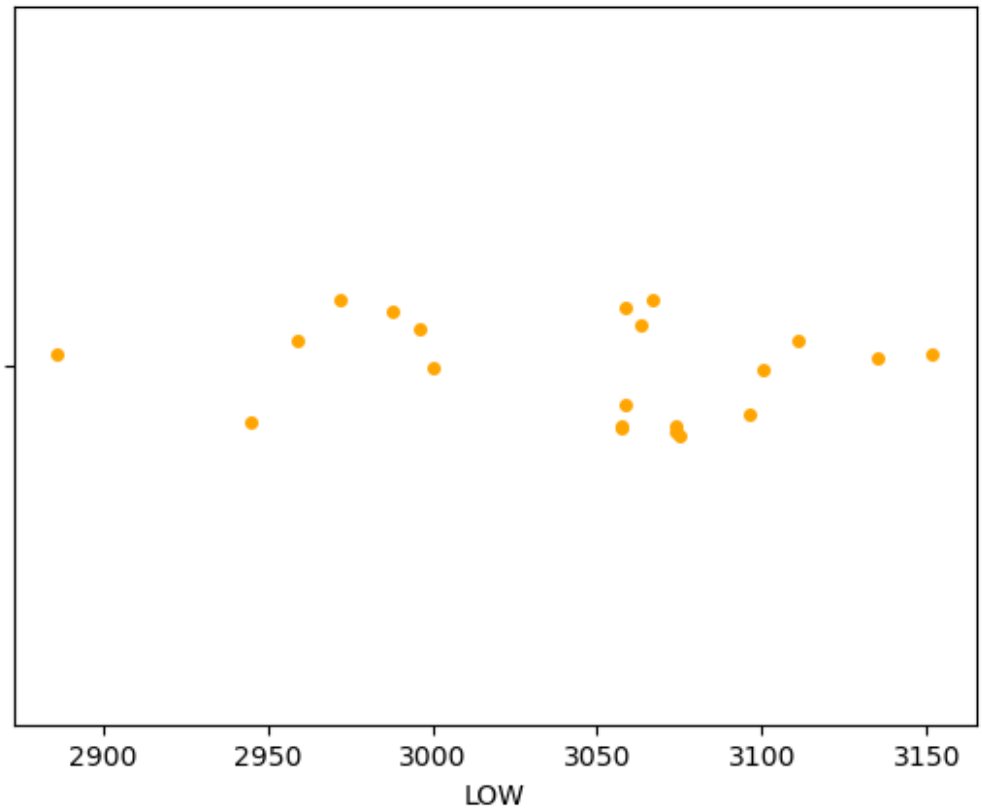

Count
10
8
6
4
2
0
0.2
0.4
0.6
0.8
1.0
VALUE
1e10

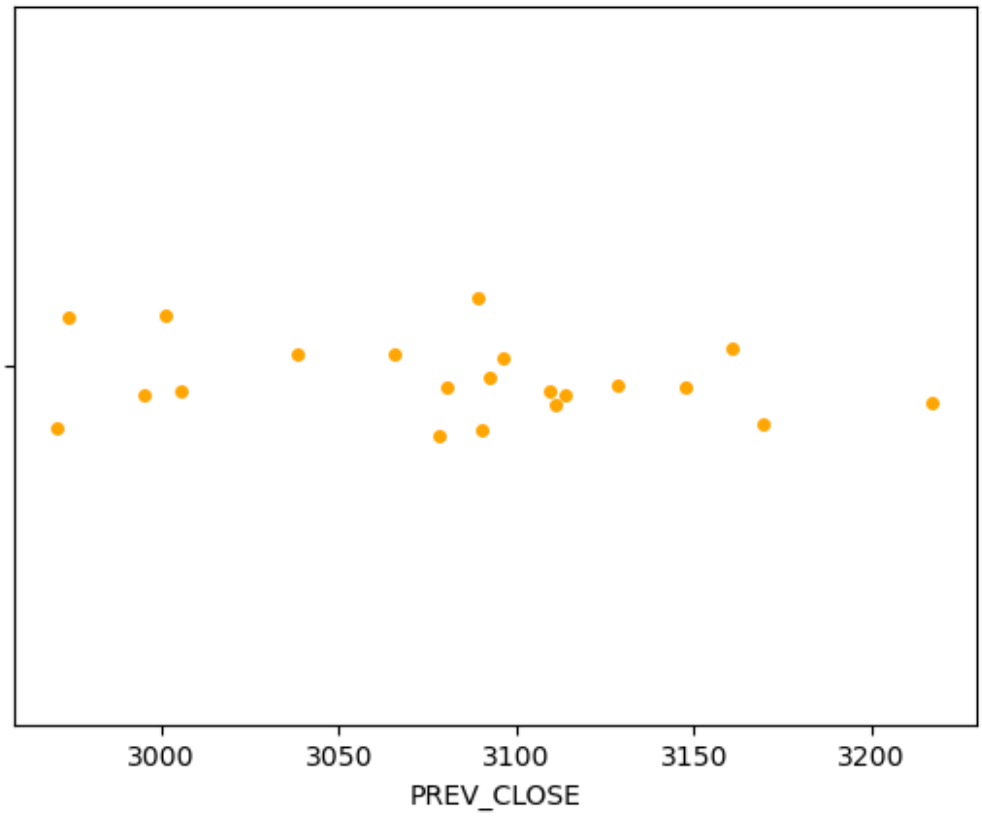

Count
7
6
5
4
3
2
1
0
40000
60000
80000
100000
120000
140000
No_of_trades

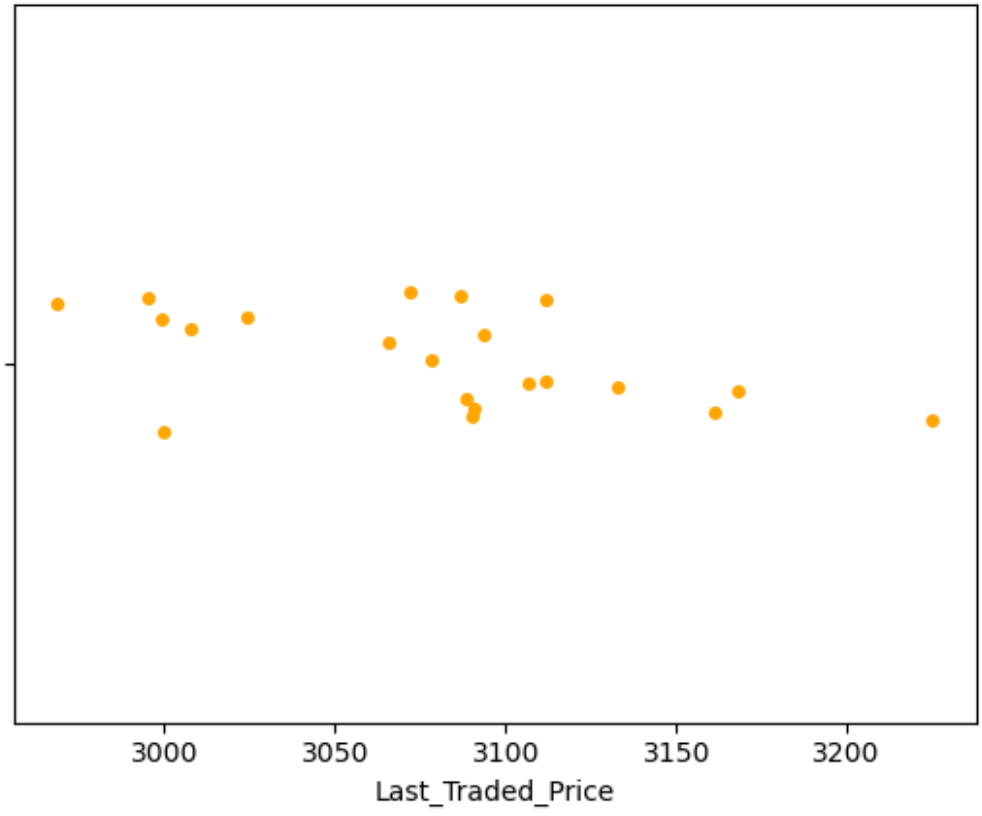
```
#Striplot
for i in df.columns:
  if df[i].dtypes != "object":
   sns.stripplot(x =df[i],color='orange' )
   plt.show()
```

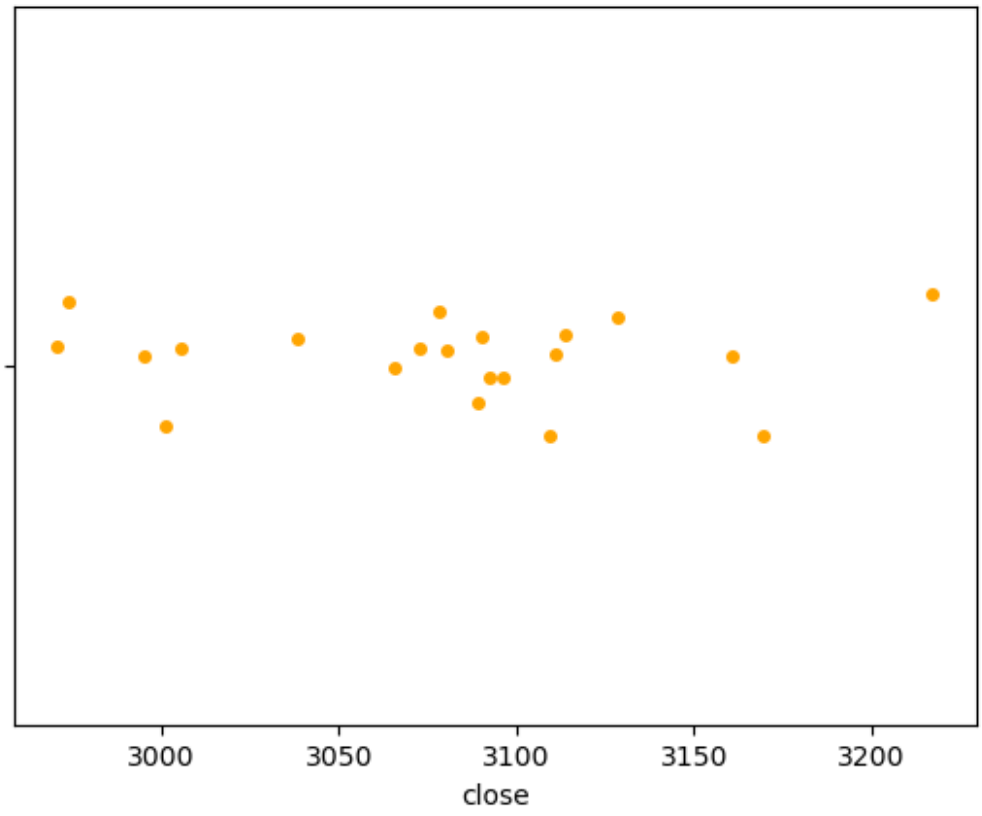


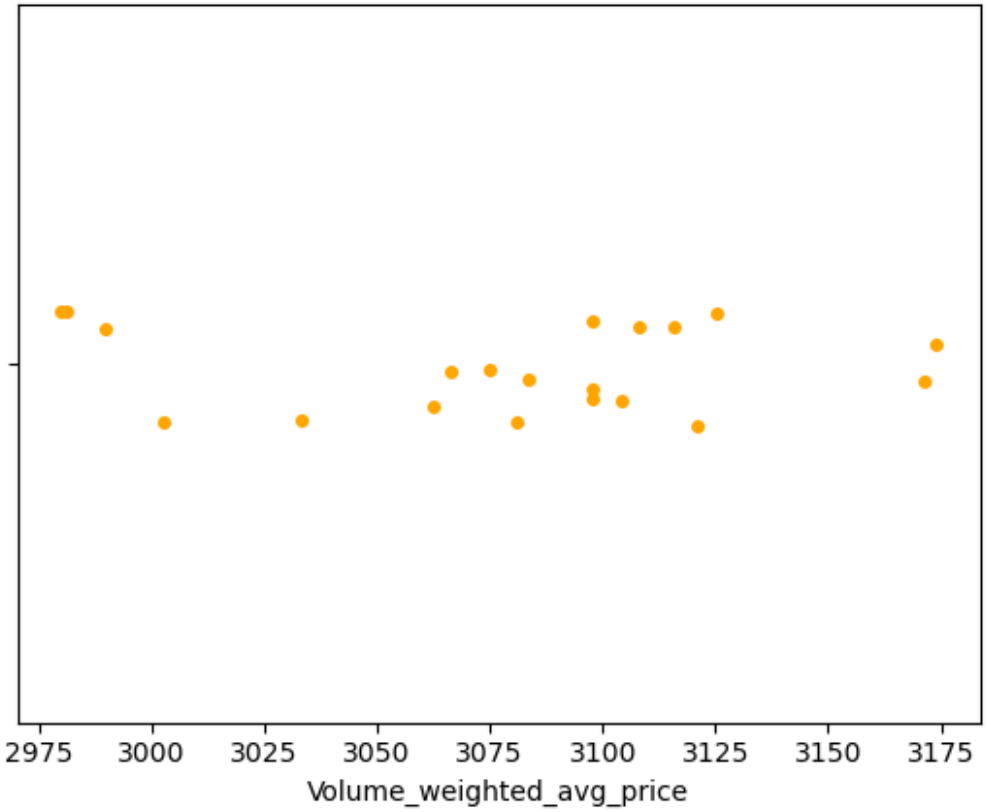

3000
3050
3100
3150
3200
HIGH

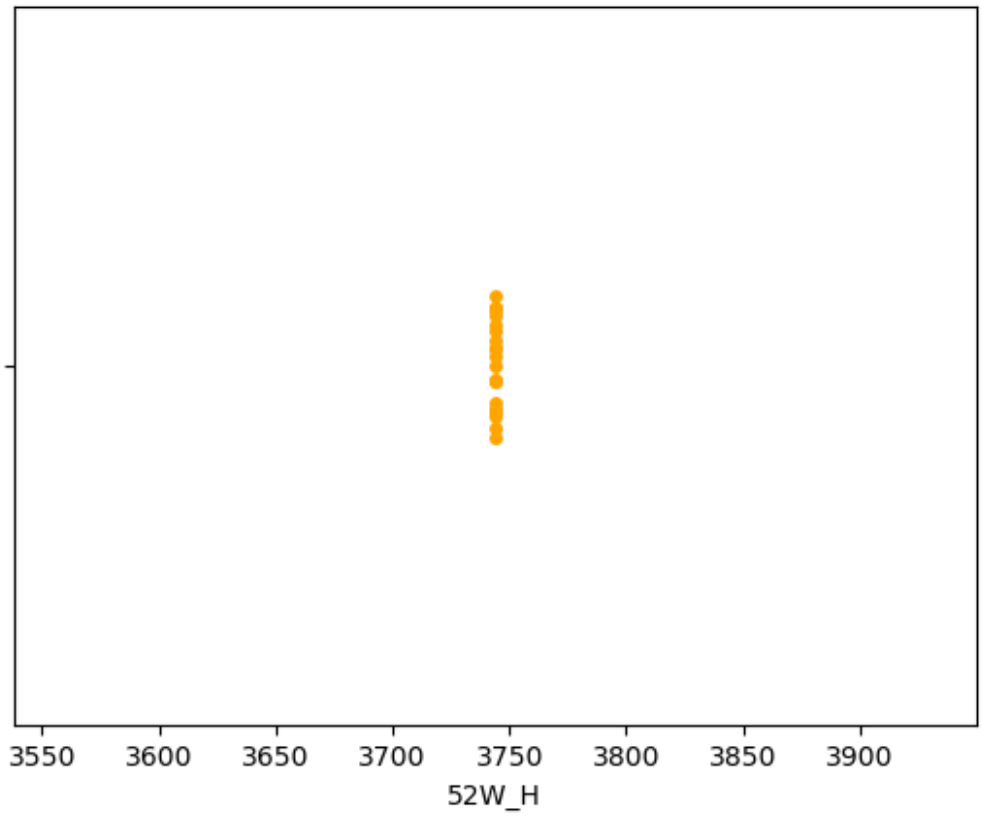

2900
2950
3000
3050
3100
3150
LOW

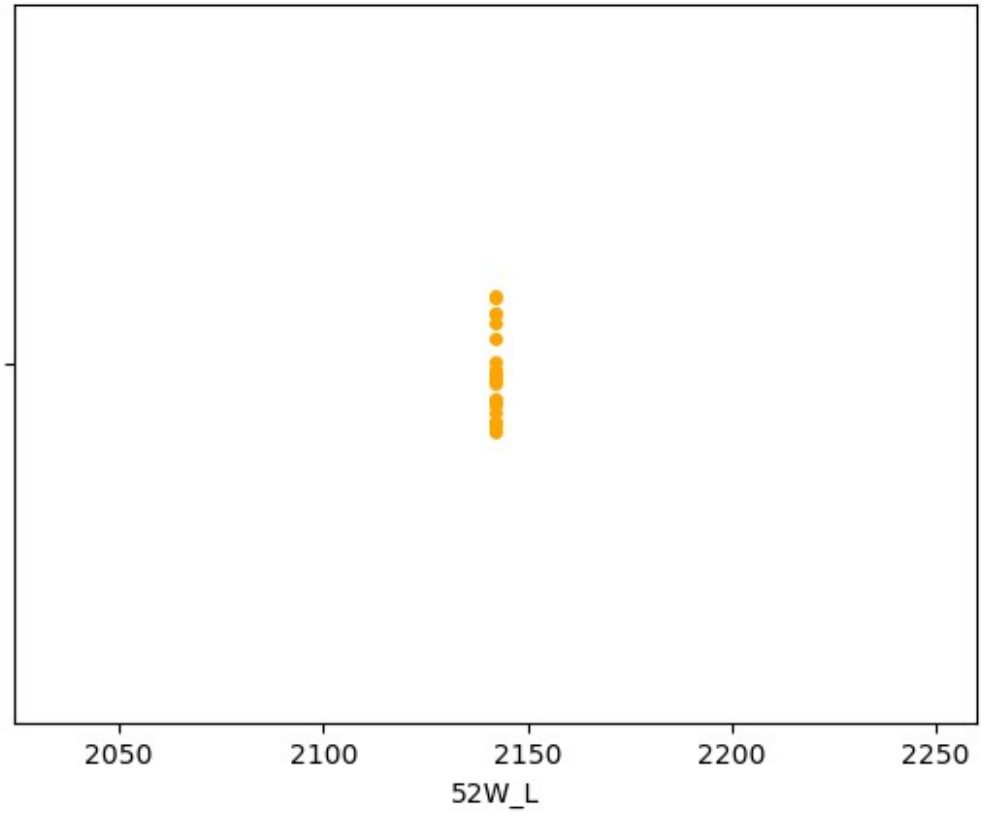

3000
3050
3100
3150
3200
PREV_CLOSE

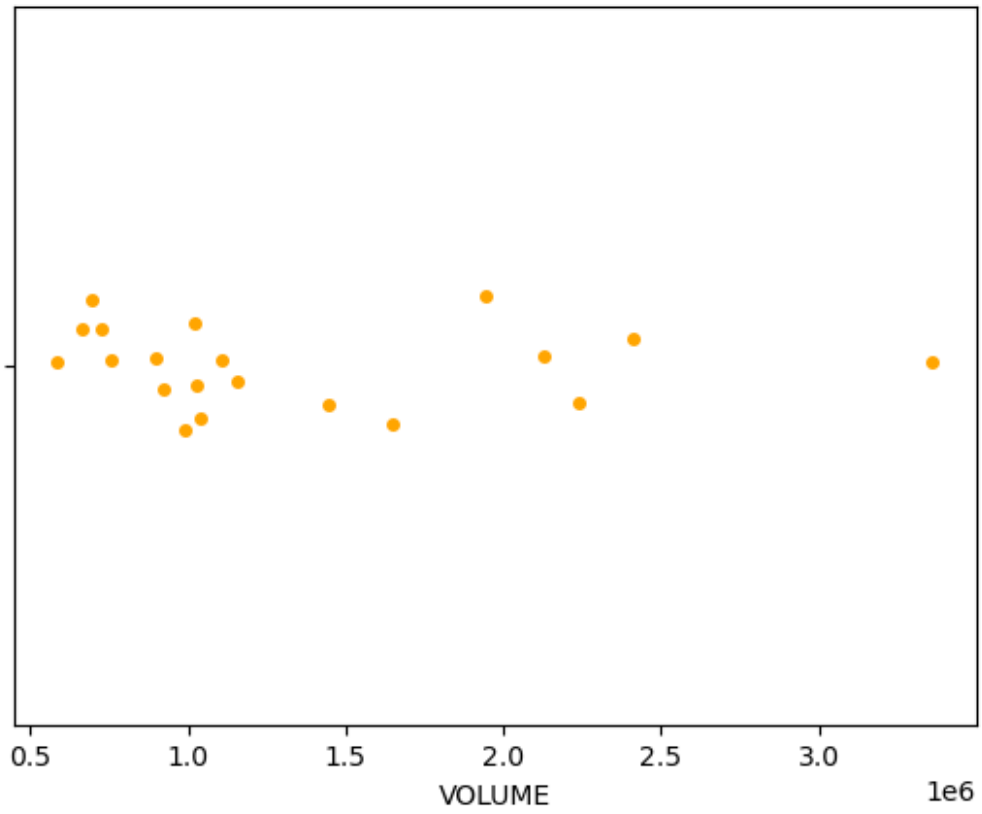

3000
3050
3100
3150
3200
Last_Traded_Price

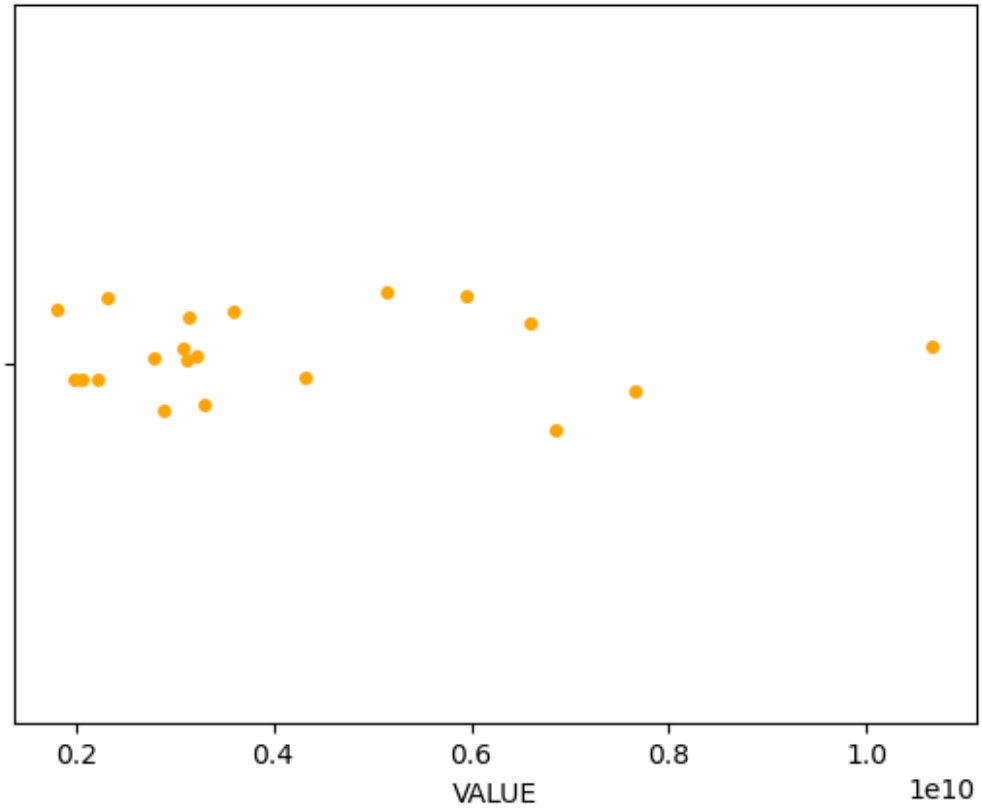

3000
3050
3100
3150
3200
close

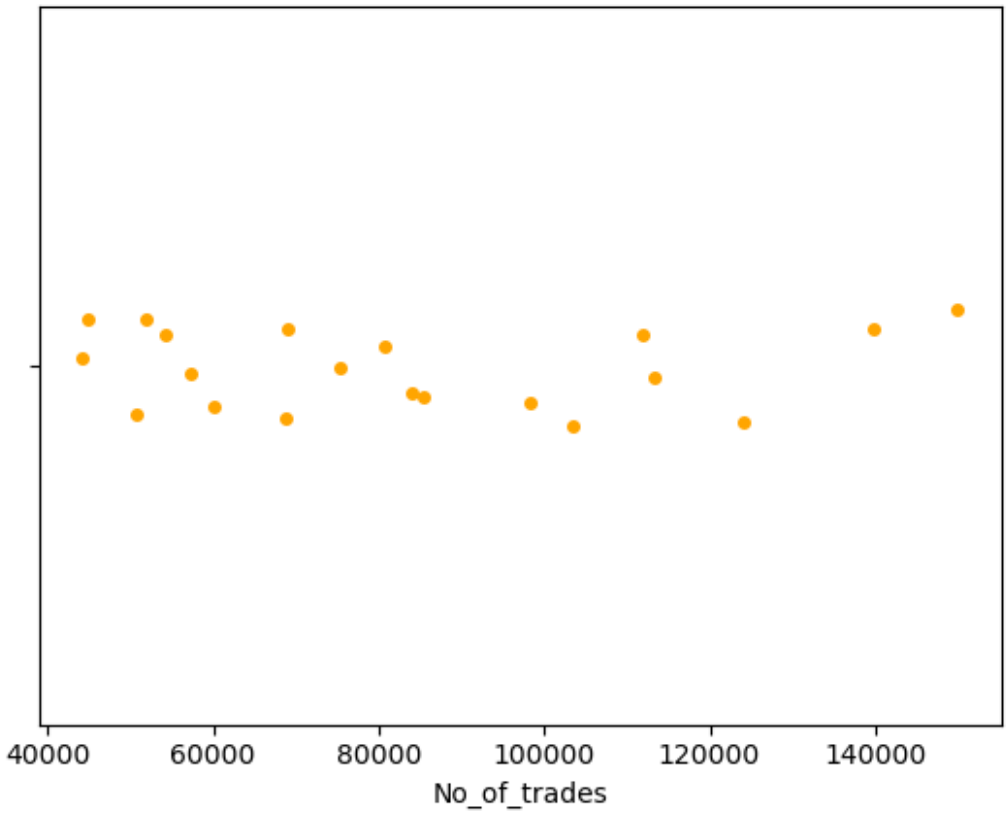

2975
3000
3025
3050
3075
3100
3125
3150
3175
Volume_weighted_avg_price


3550
3600
3650
3700
3750
3800
3850
3900
52W_H

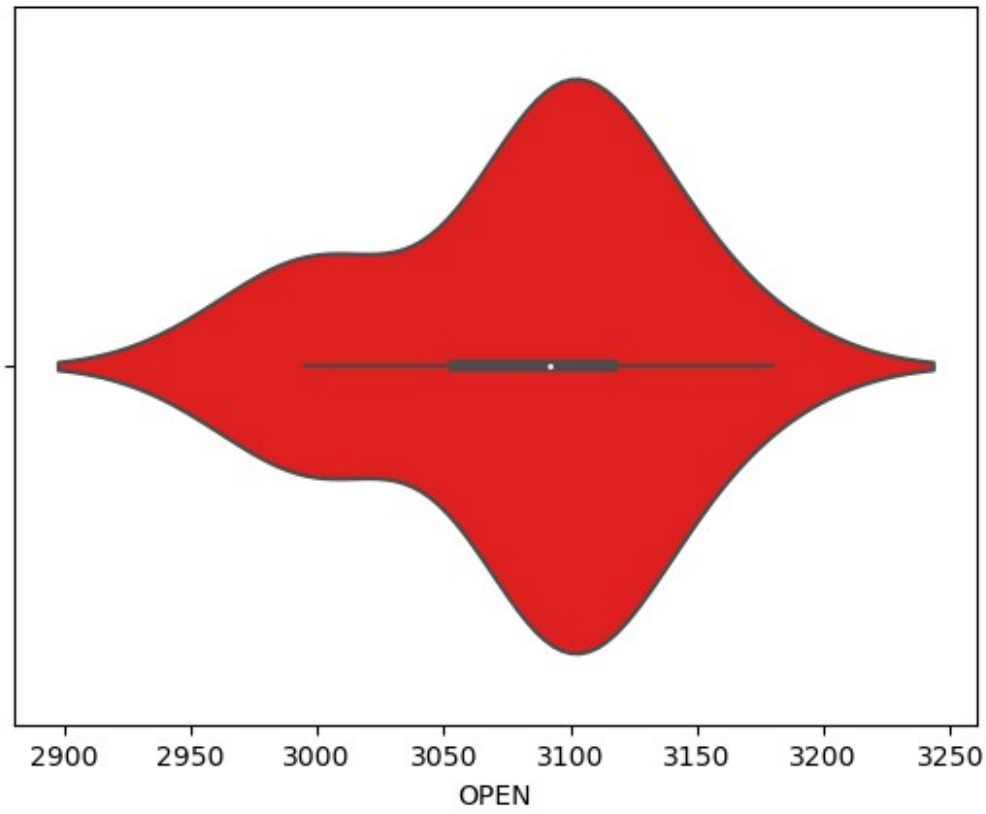

2050
2100
2150
2200
2250
52W_L

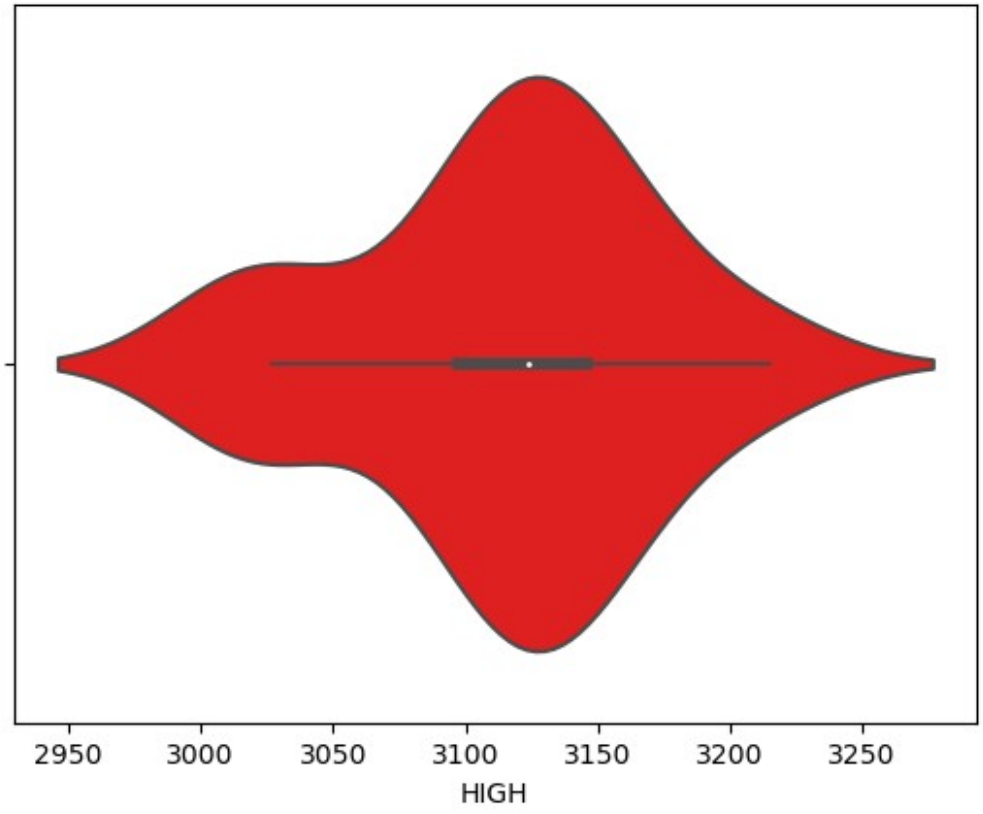

0.5
1.0
1.5
2.0
2.5
3.0
VOLUME
1e6

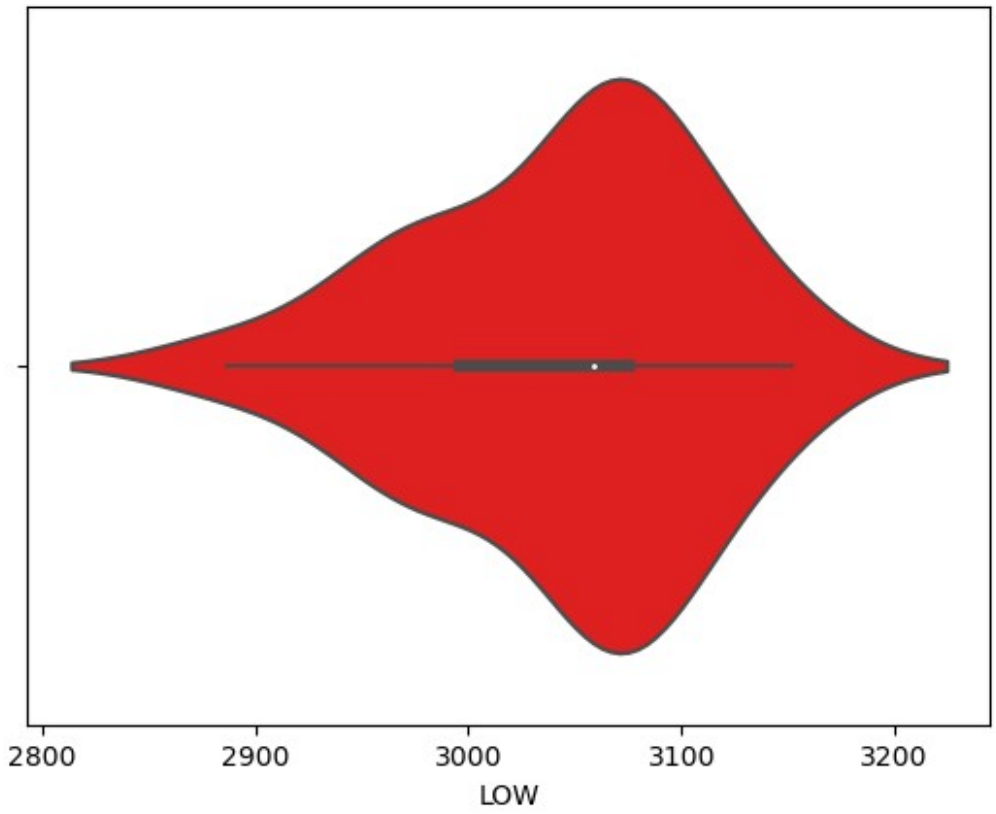

0.2
0.4
0.6
0.8
1.0
VALUE
1e10

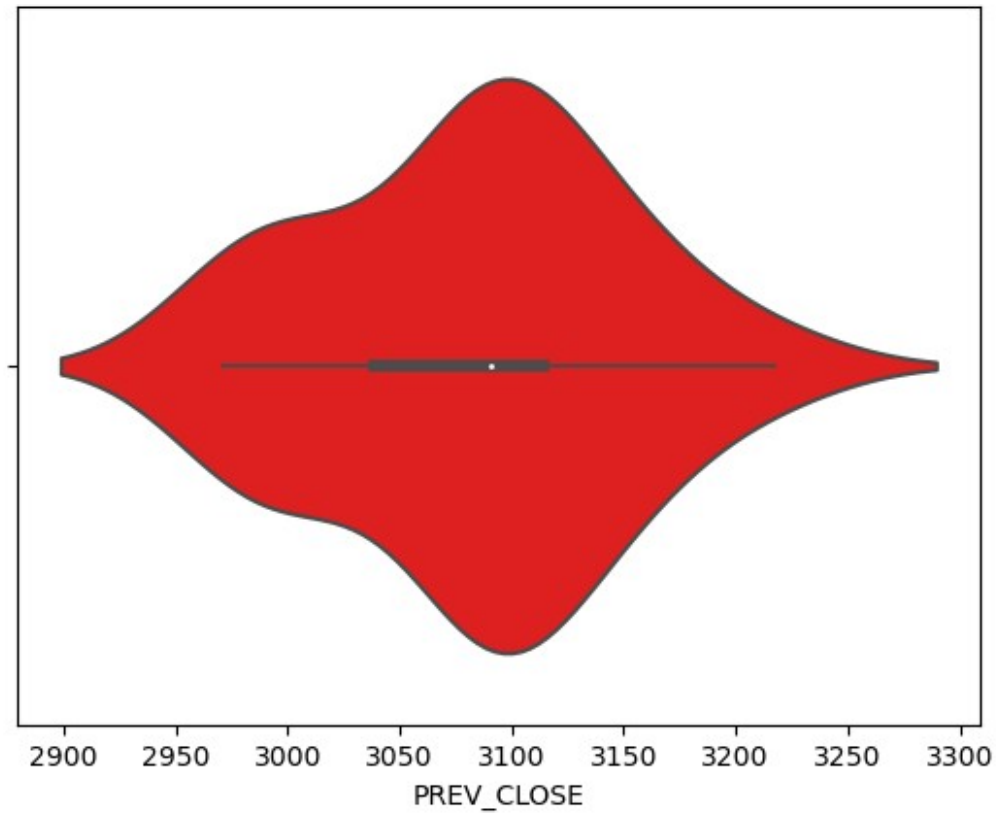


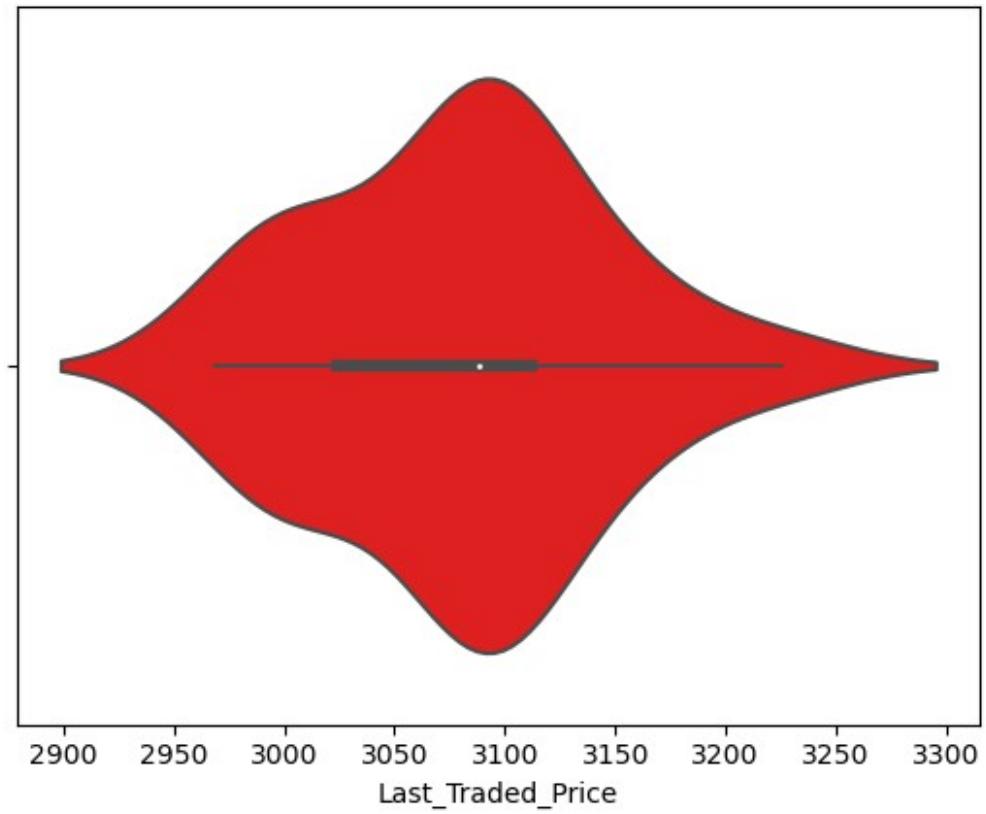

```
#Violinplot
for i in df.columns:
  if df[i].dtypes != "object":
    sns.violinplot(x =df[i],color='red' )
    plt.show()
```

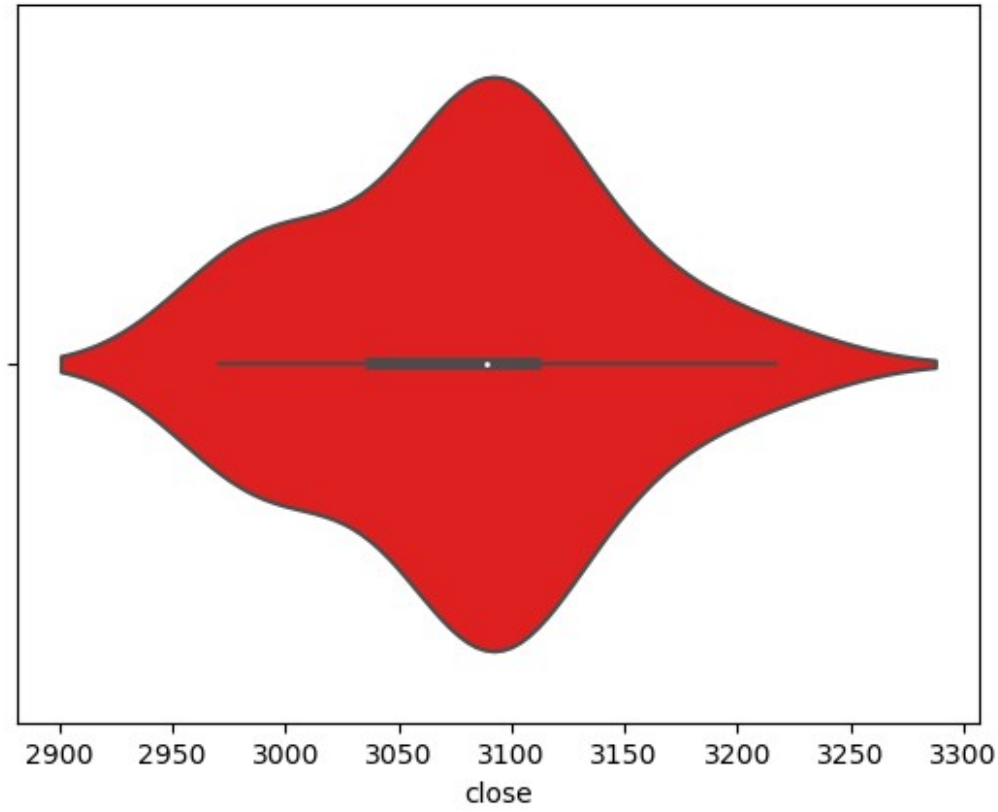

2900
2950
3000
3050
3100
3150
3200
3250
OPEN

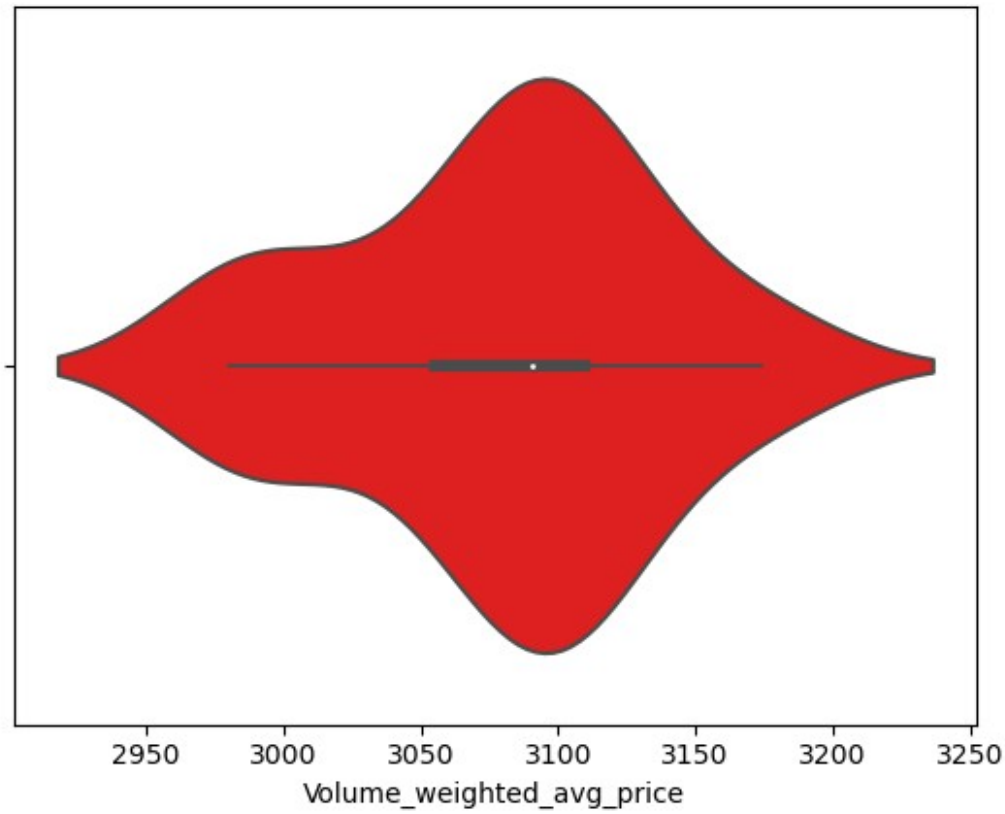

2950
3000
3050
3100
3150
3200
3250
HIGH

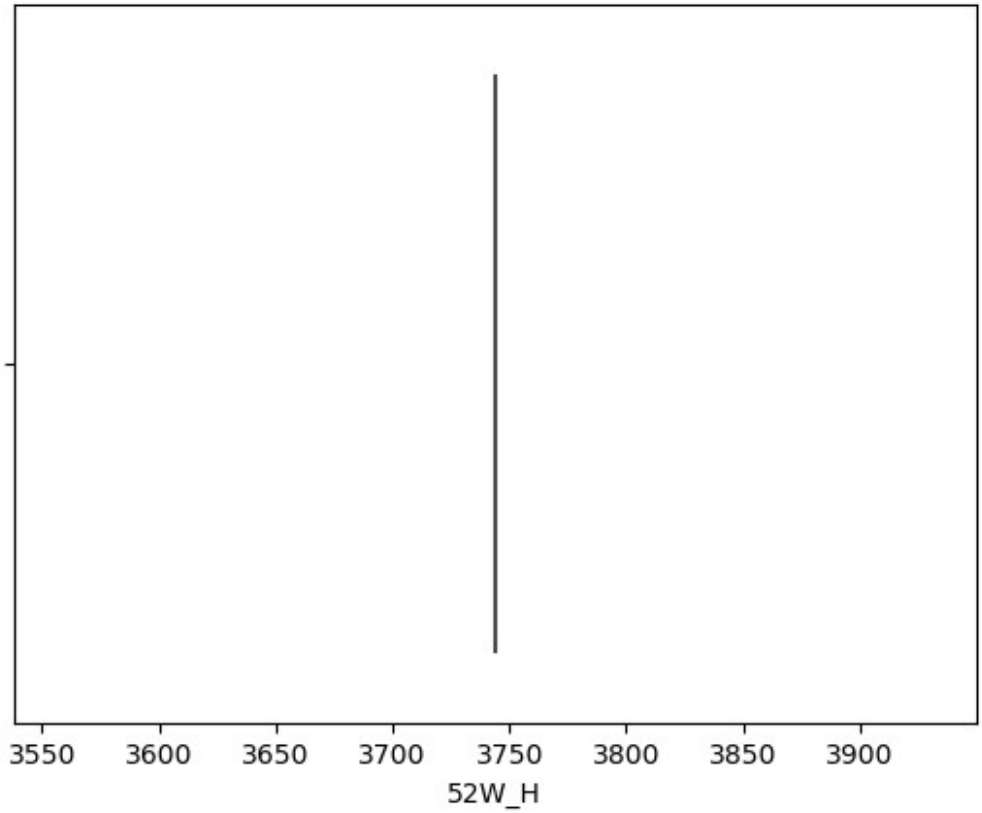

2800
2900
3000
3100
3200
LOW

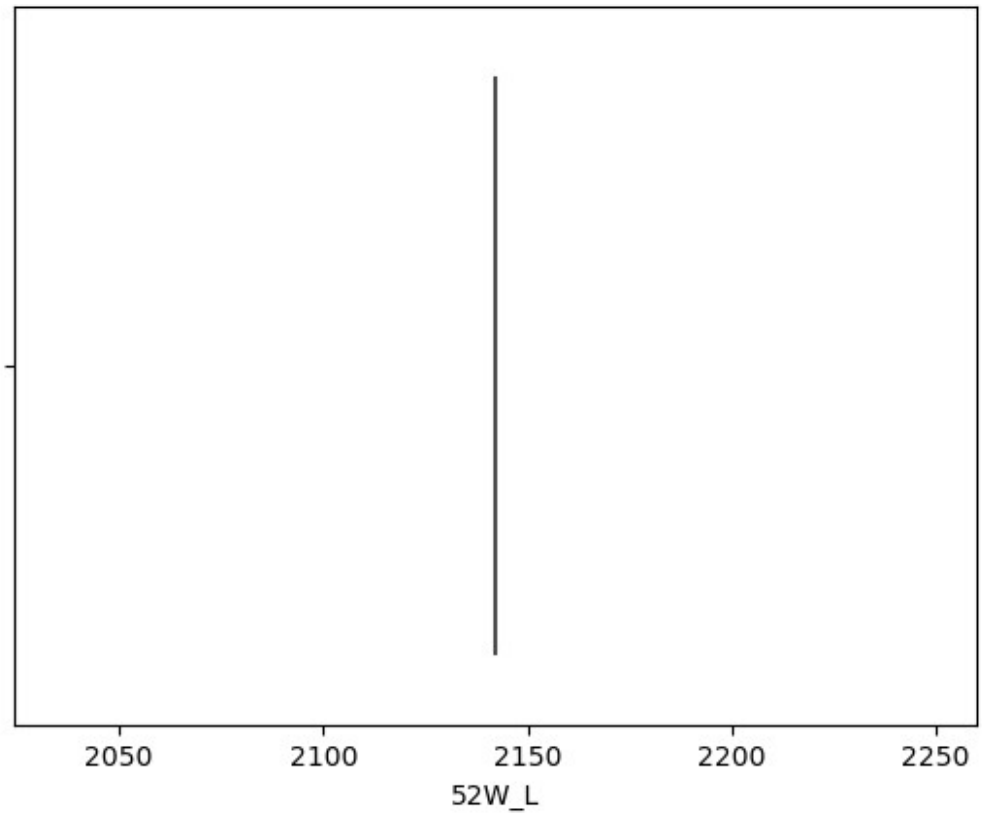

2900
2950
3000
3050
3100
3150
3200
3250
3300
PREV_CLOSE

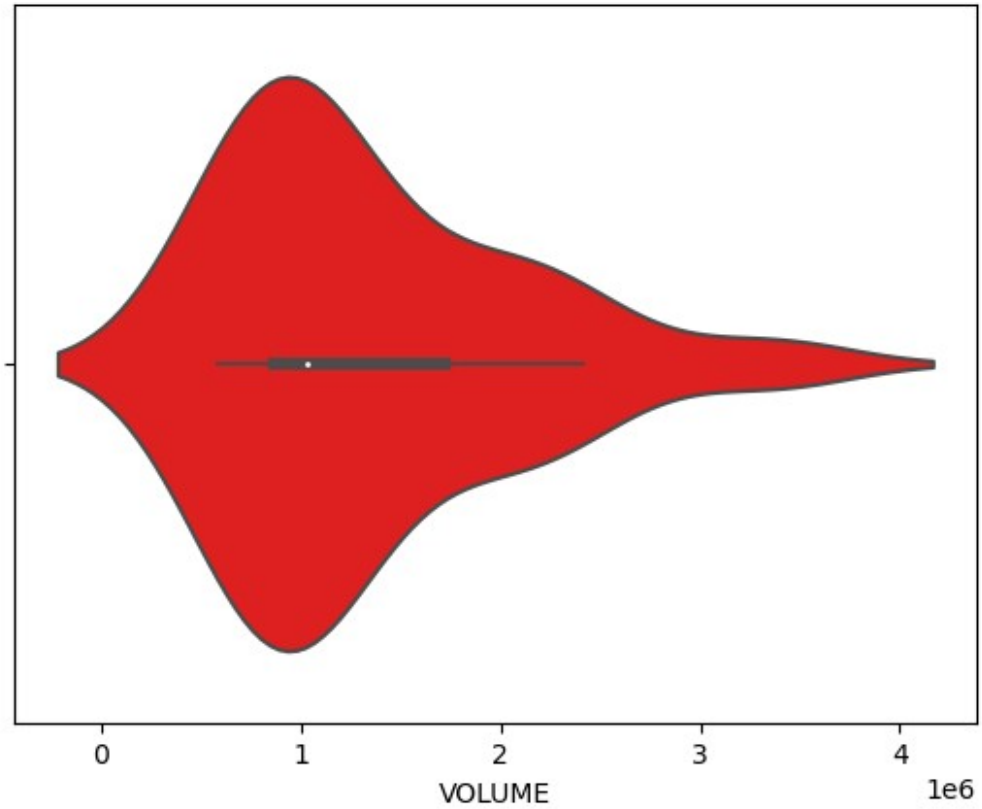

2900
2950
3000
3050
3100
3150
3200
3250
3300
Last_Traded_Price

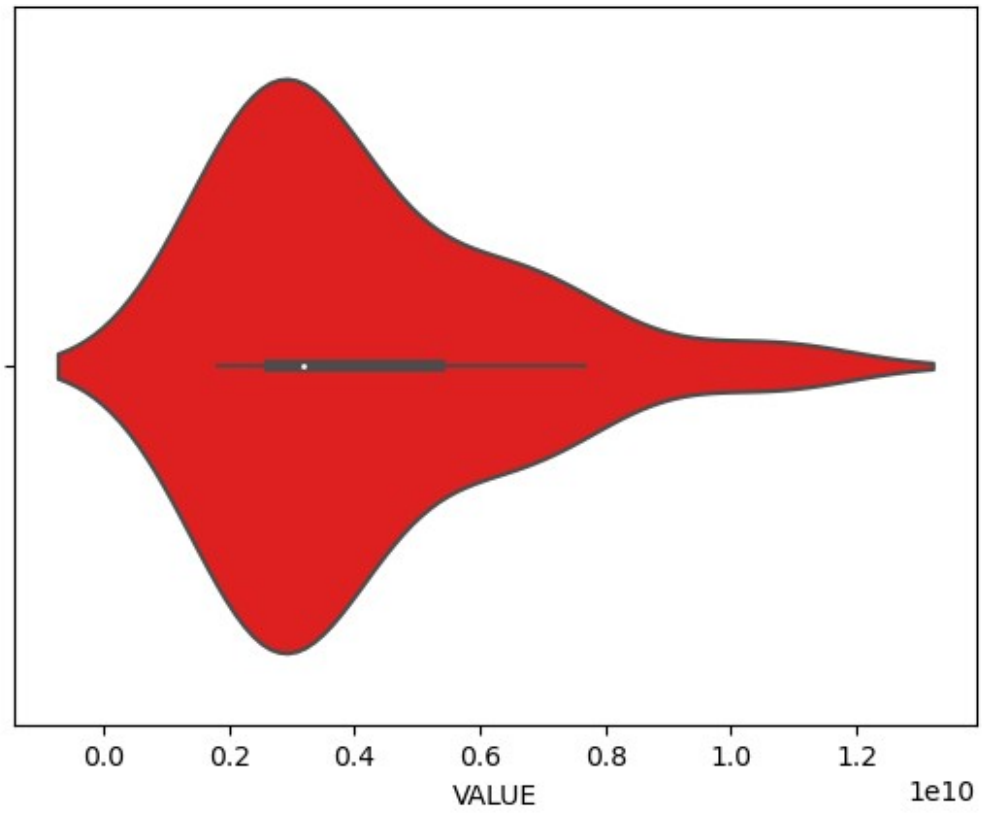

2900
2950
3000
3050
3100
3150
3200
3250
3300
close

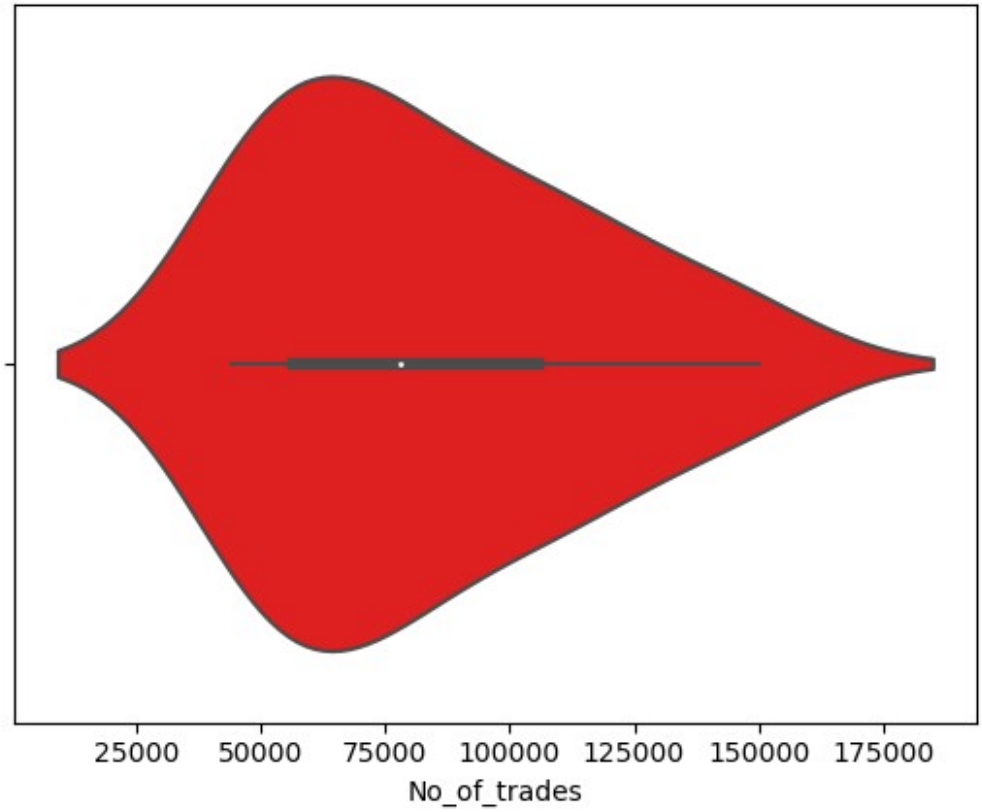

2950
3000
3050
3100
3150
3200
3250
Volume_weighted_avg_price


3550
3600
3650
3700
3750
3800
3850
3900
52W_H

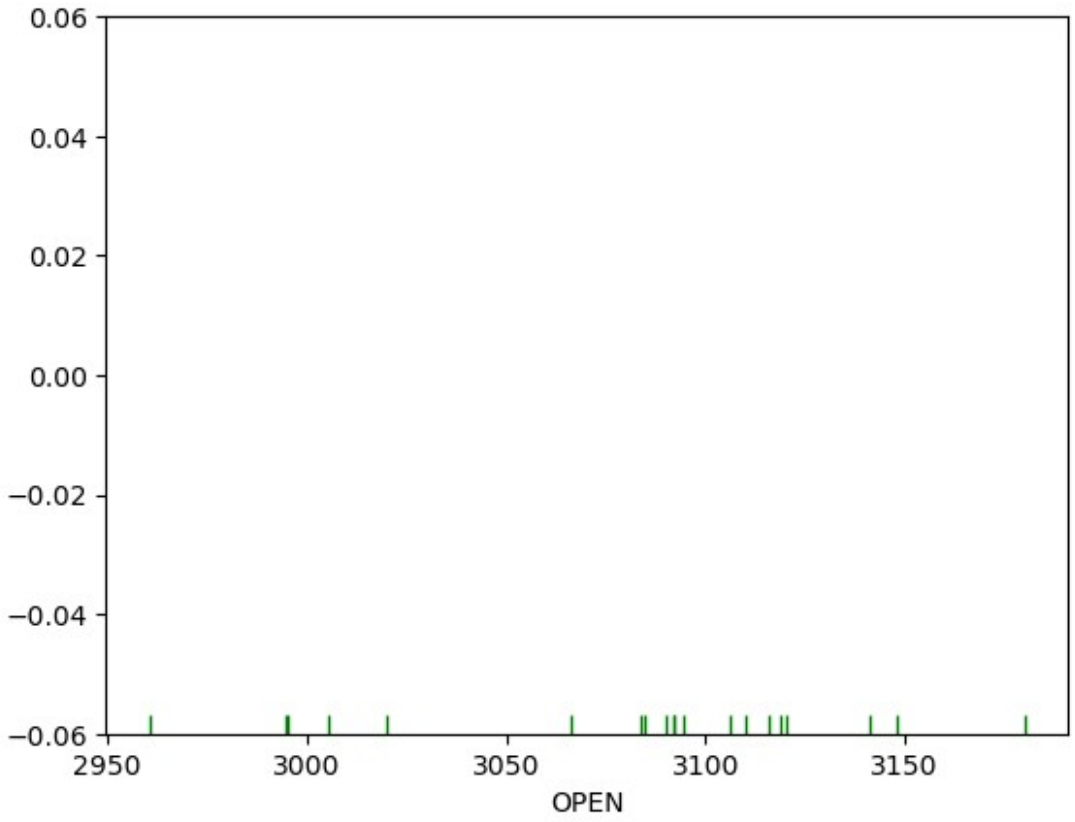

2050
2100
2150
2200
2250
52W_L

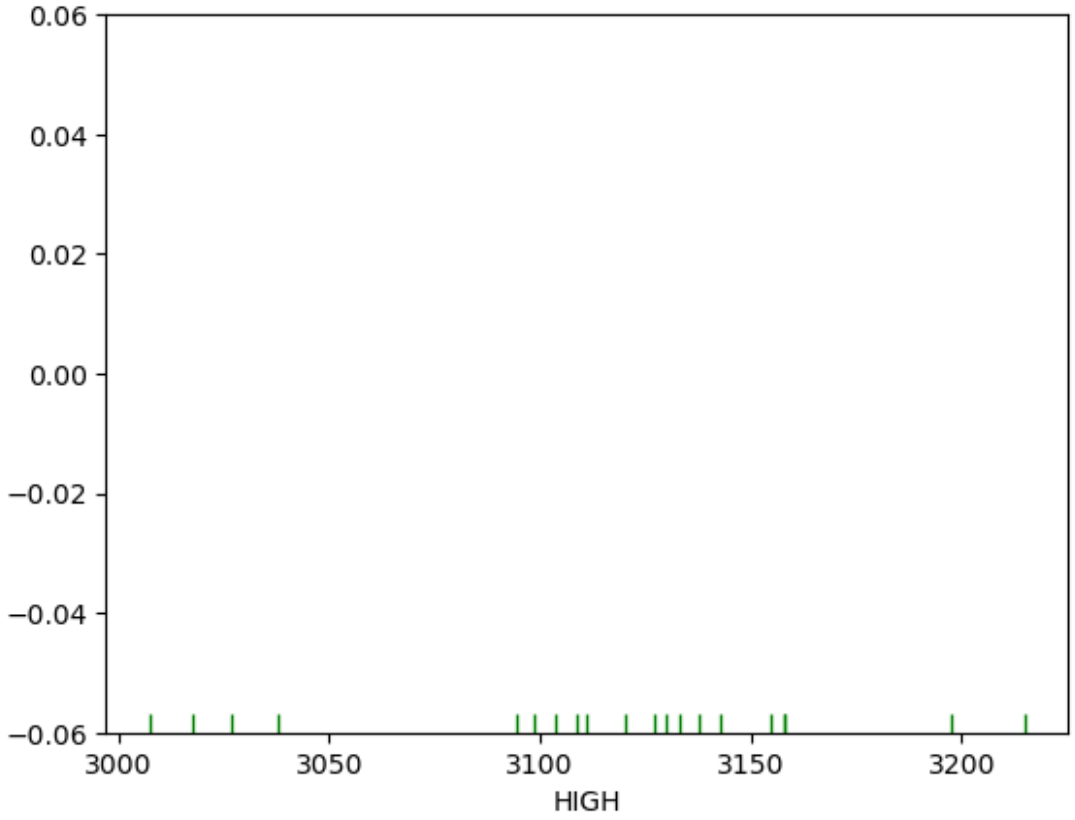

0
1
2
3
4
VOLUME
1e6

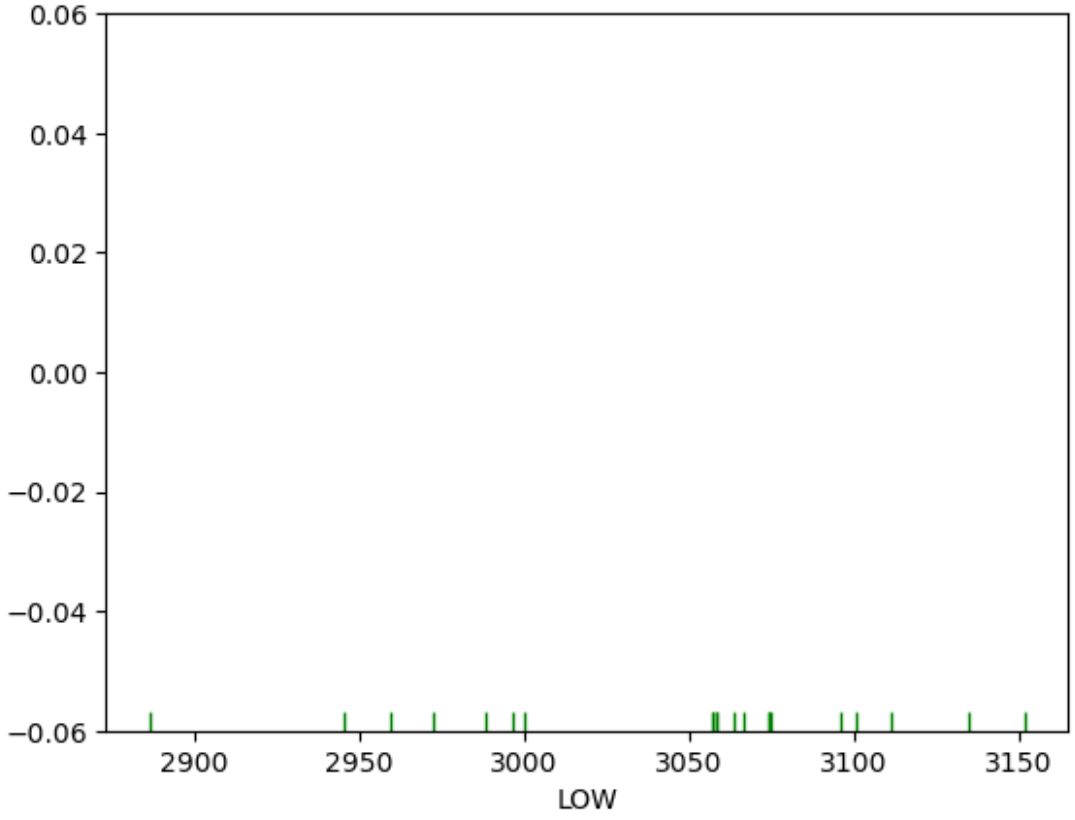

0.0
0.2
0.4
0.6
0.8
1.0
1.2
VALUE
1e10

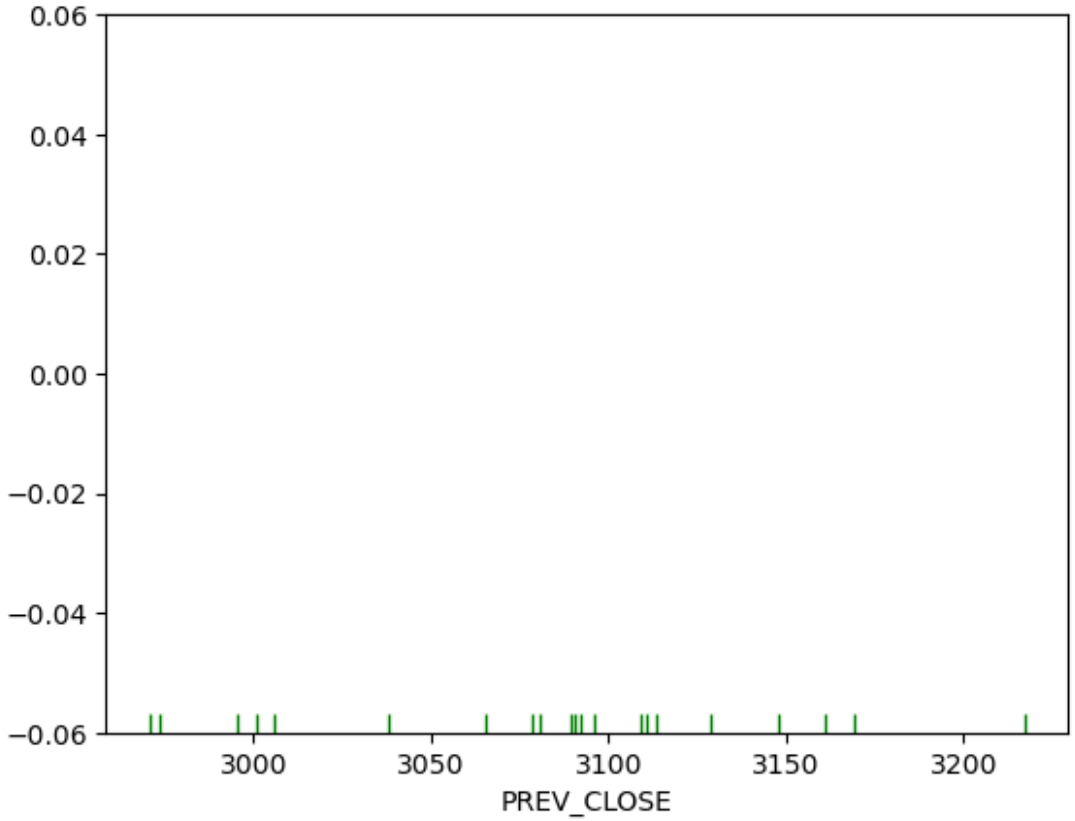

25000
50000
75000
100000
125000
150000
175000
No_of_trades

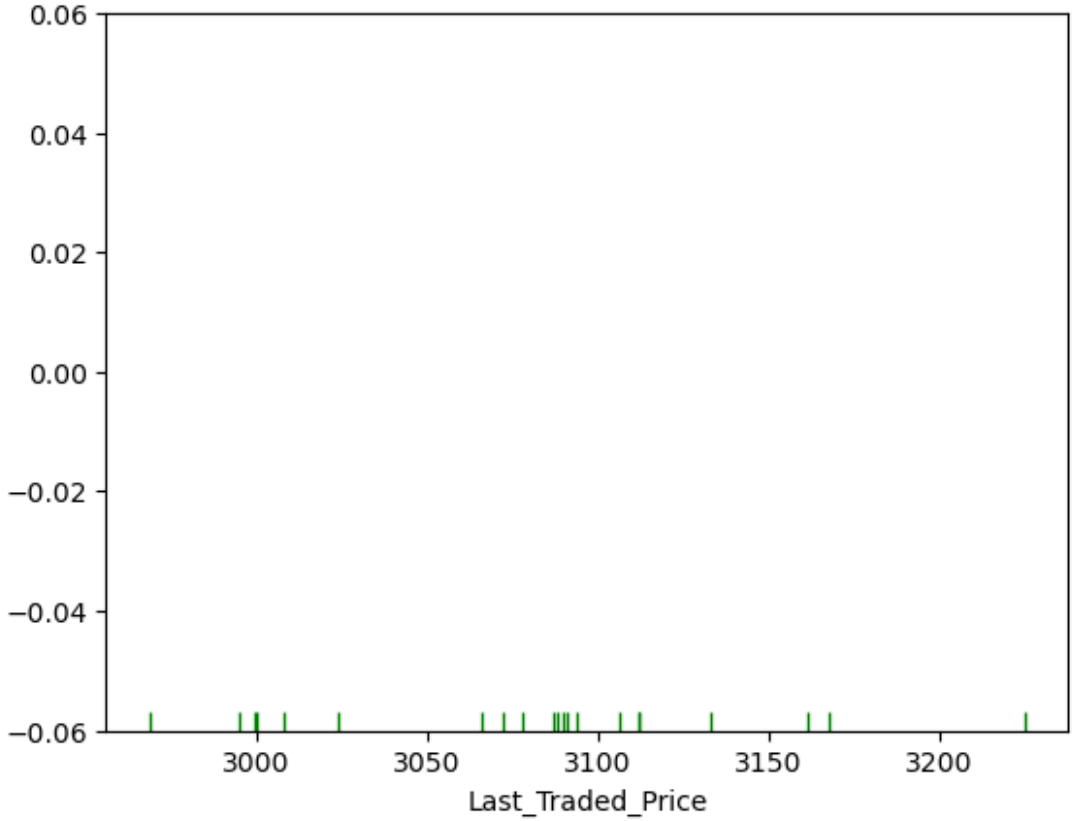
```
#Rugplot
for i in df.columns:
  if df[i].dtypes != "object":
   sns.rugplot(x =df[i],color='green' )
   plt.show()
```

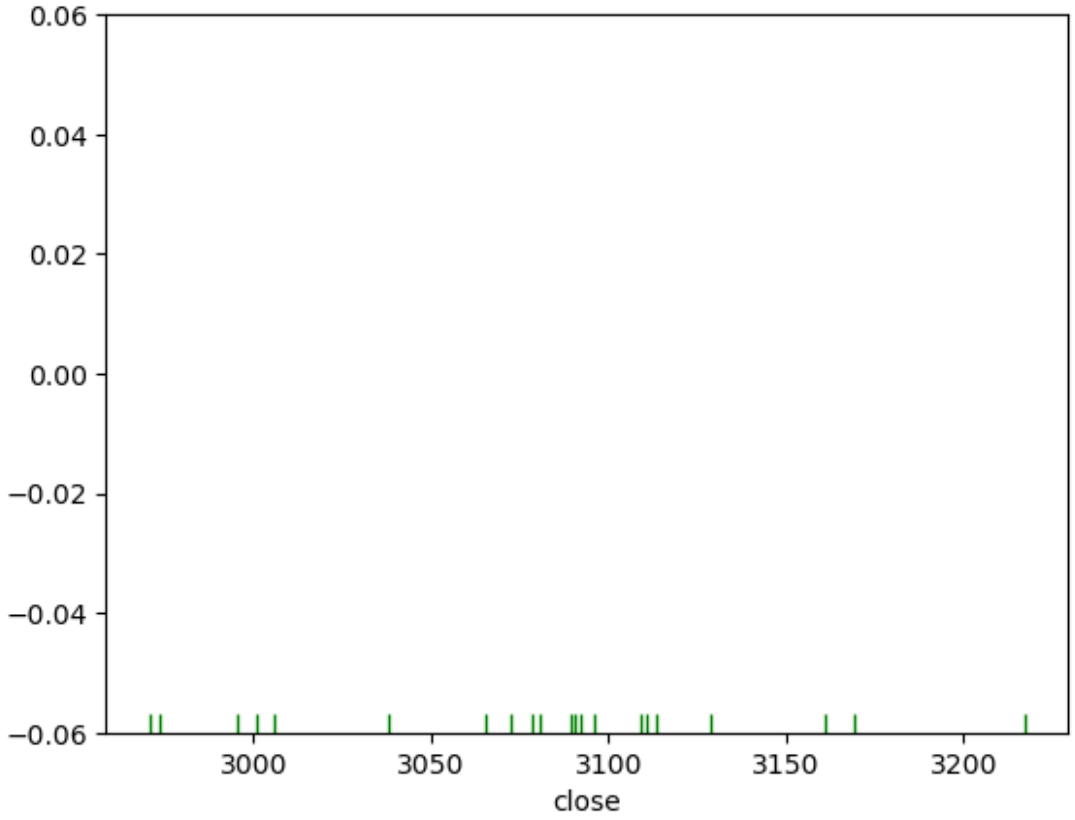


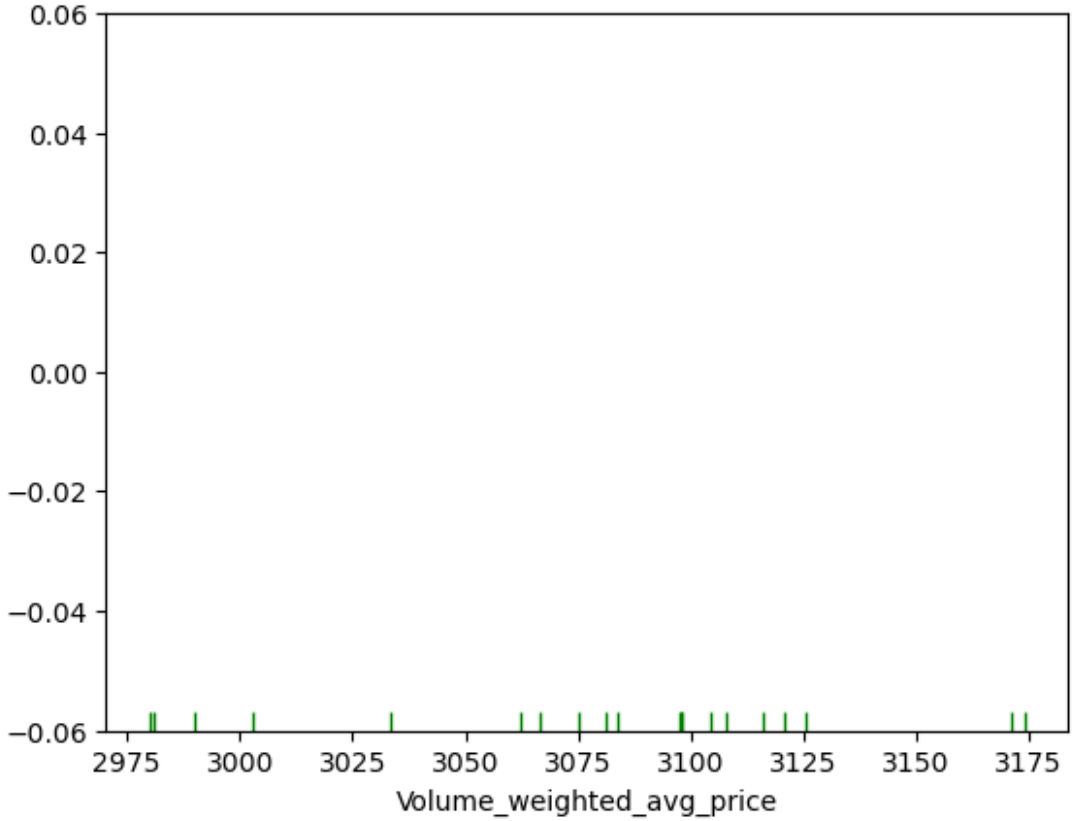

0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
3000
3050
3100
3150
3200
HIGH

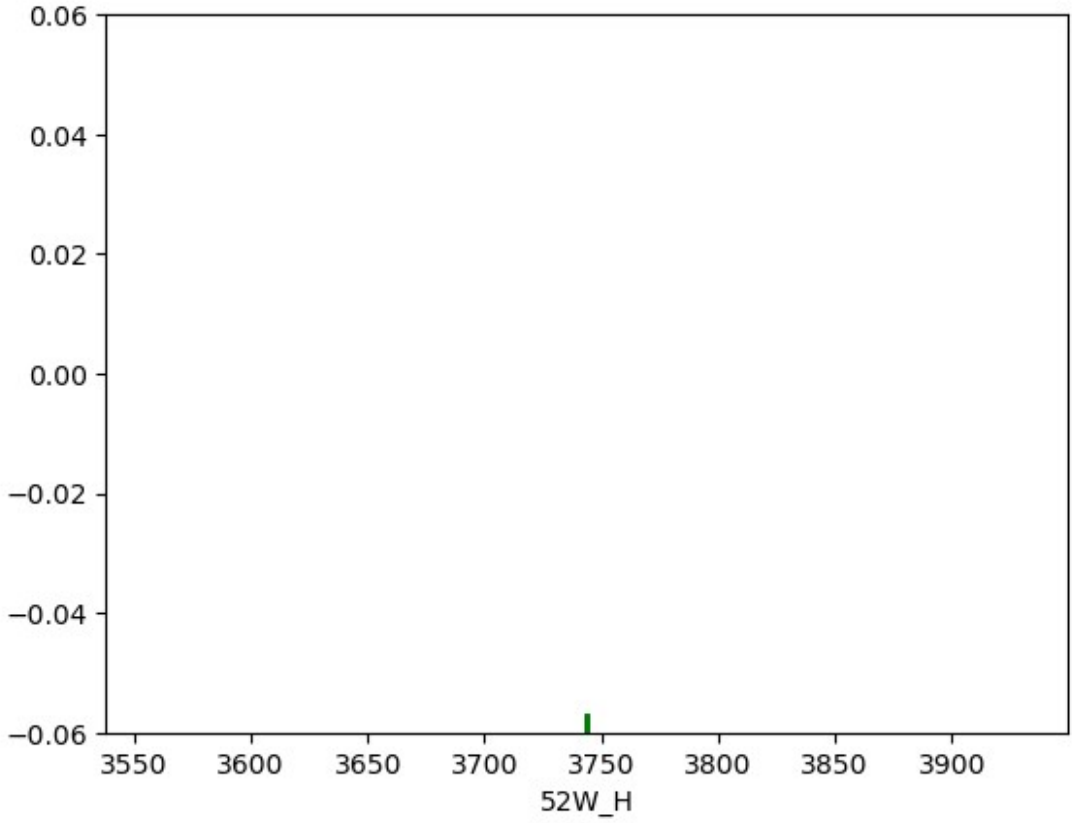

0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
2900
2950
3000
3050
3100
3150
LOW

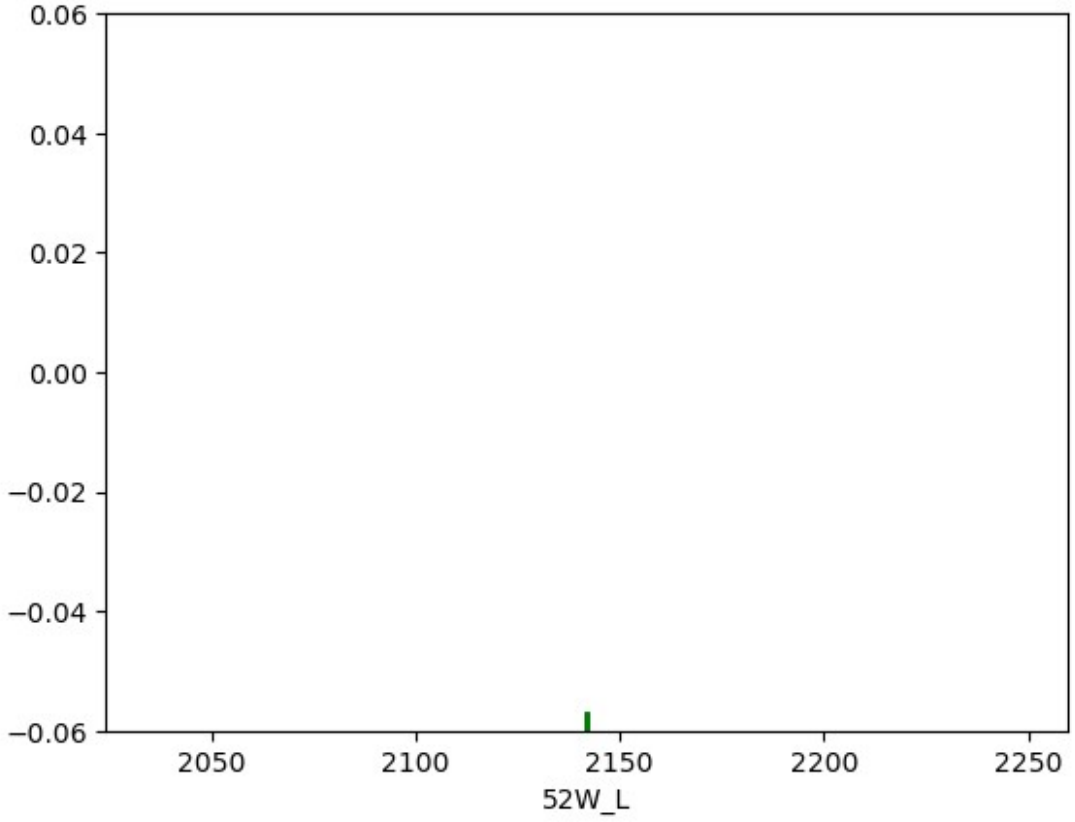

0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
3000
3050
3100
3150
3200
PREV_CLOSE

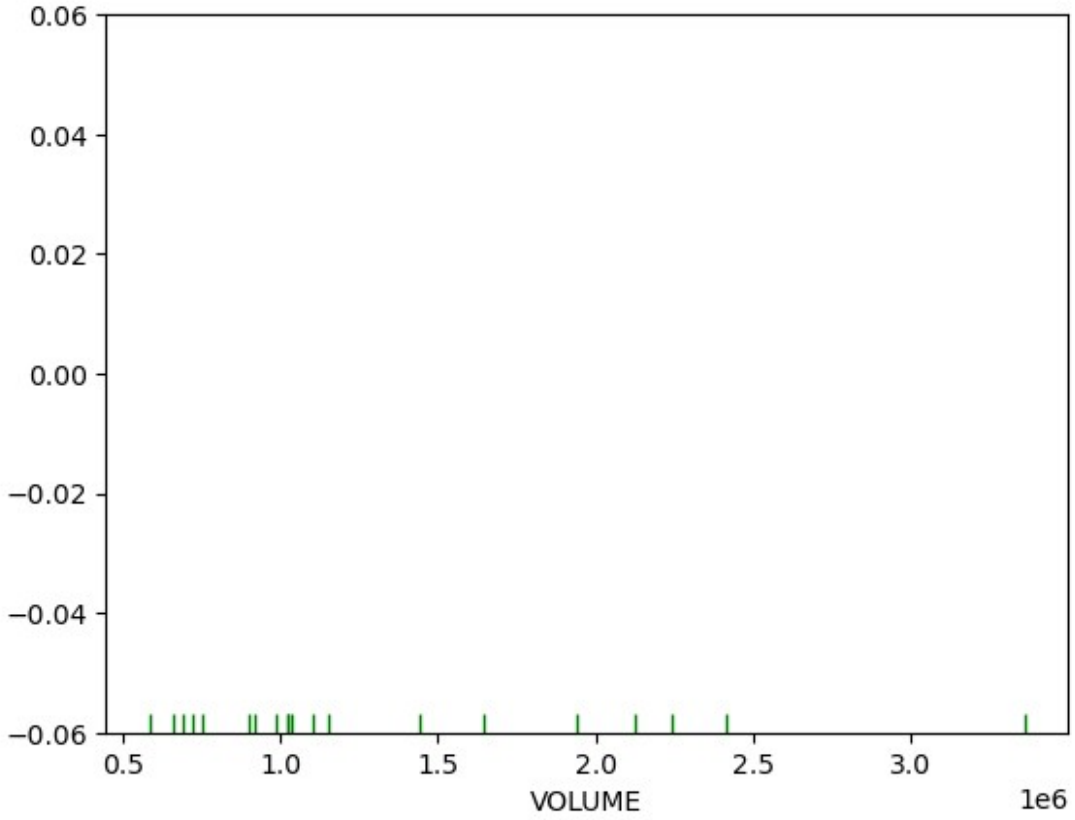

0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
3000
3050
3100
3150
3200
Last_Traded_Price

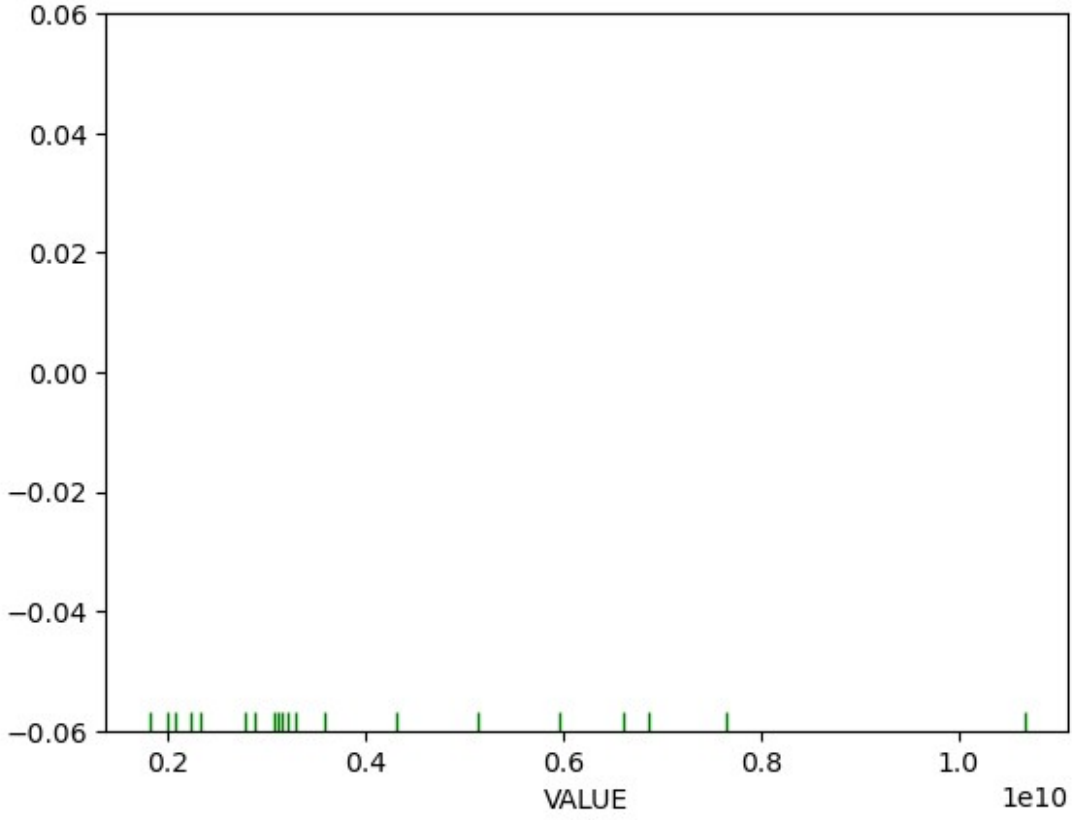

0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
3000
3050
3100
3150
3200
close

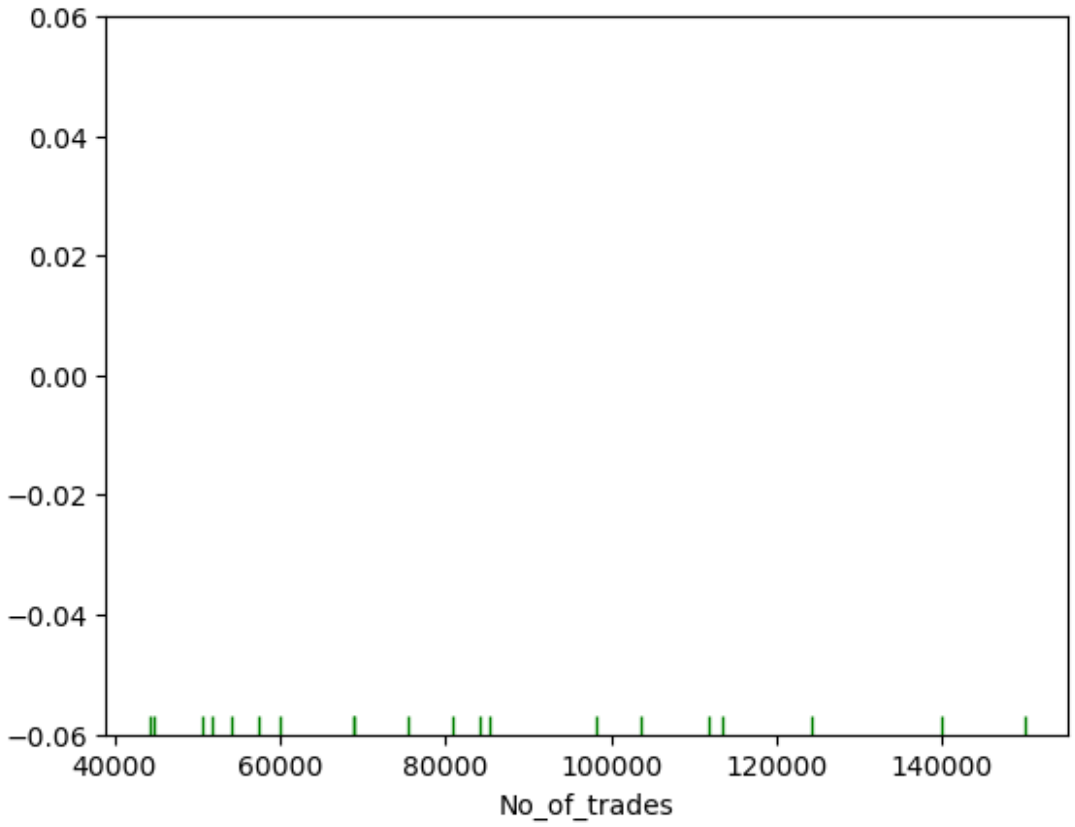

0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
2975
3000
3025
3050
3075
3100
3125
3150
3175
Volume_weighted_avg_price


0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
3550
3600
3650
3700
3750
3800
3850
3900
52W_H

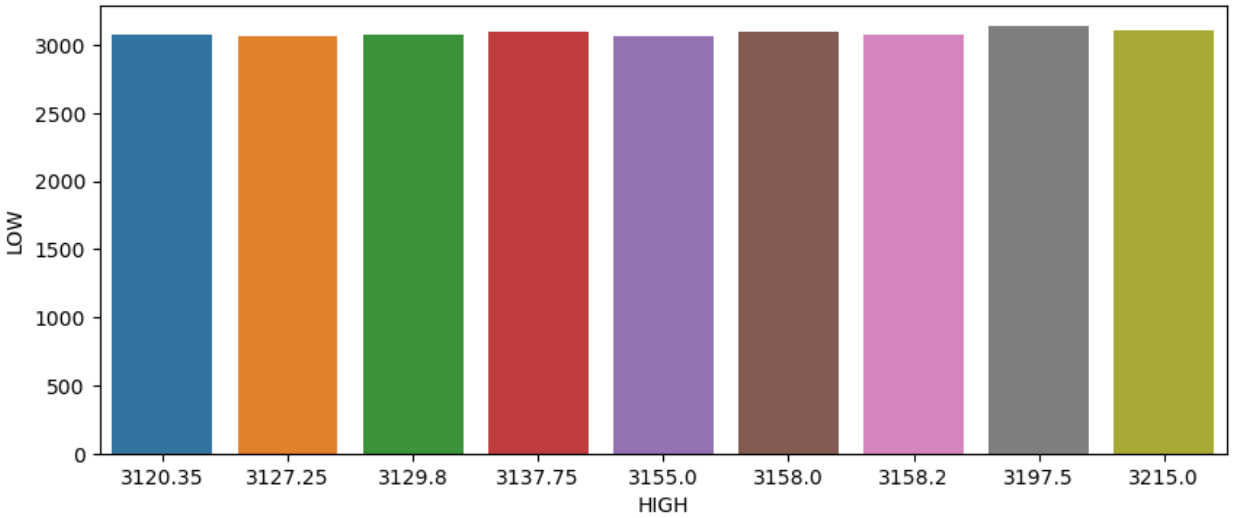

0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
2050
2100
2150
2200
2250
52W_L


0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
0.5
1.0
1.5
2.0
2.5
3.0
VOLUME
1e6


0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
0.2
0.4
0.6
0.8
1.0
VALUE
1e10

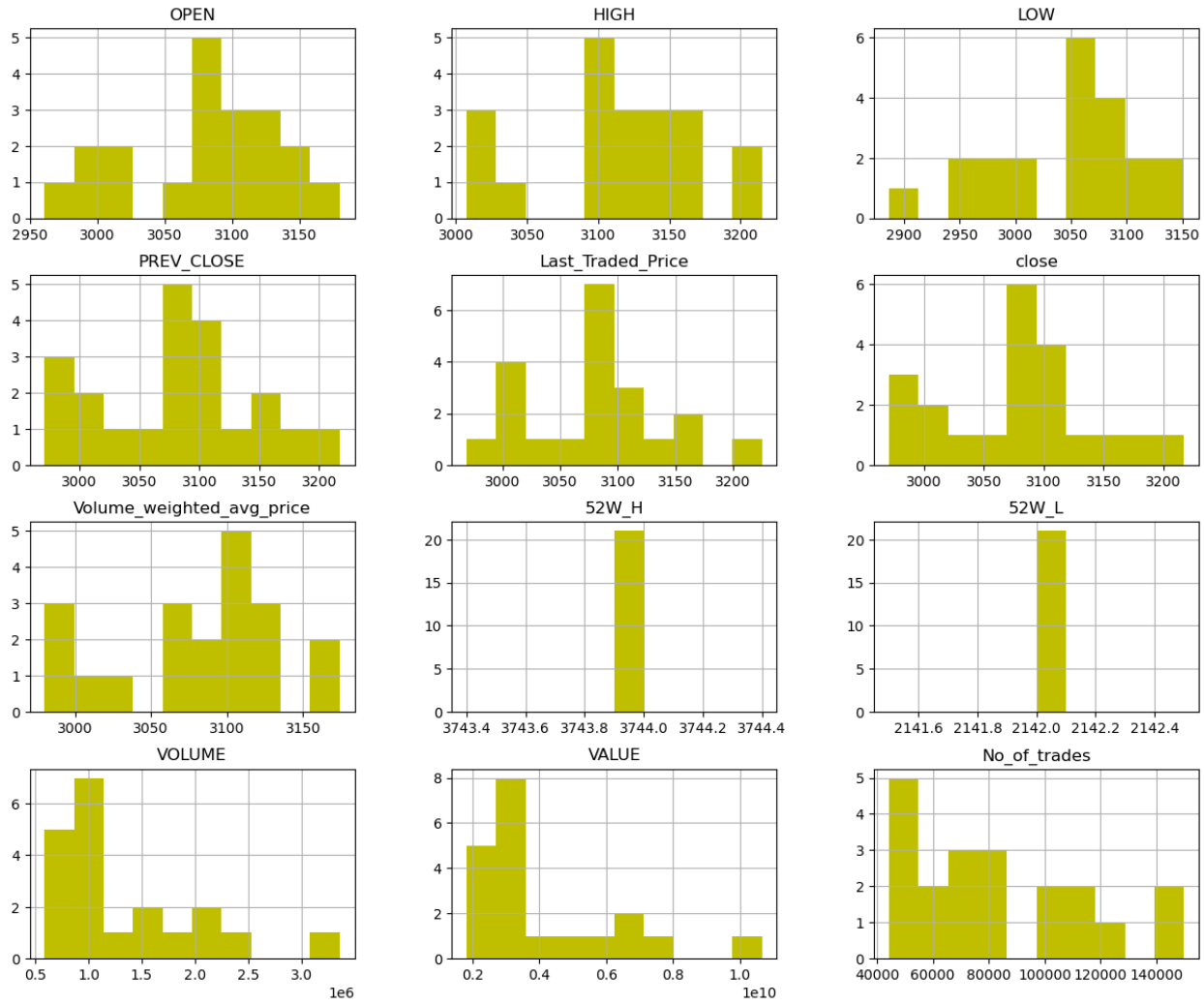



```
#Barplot
plt.figure(figsize=(10,4))
sns.barplot(x='HIGH',y='LOW',data=df.sort_values(by='LOW',ascending=Fa
lse)[:10]);
```



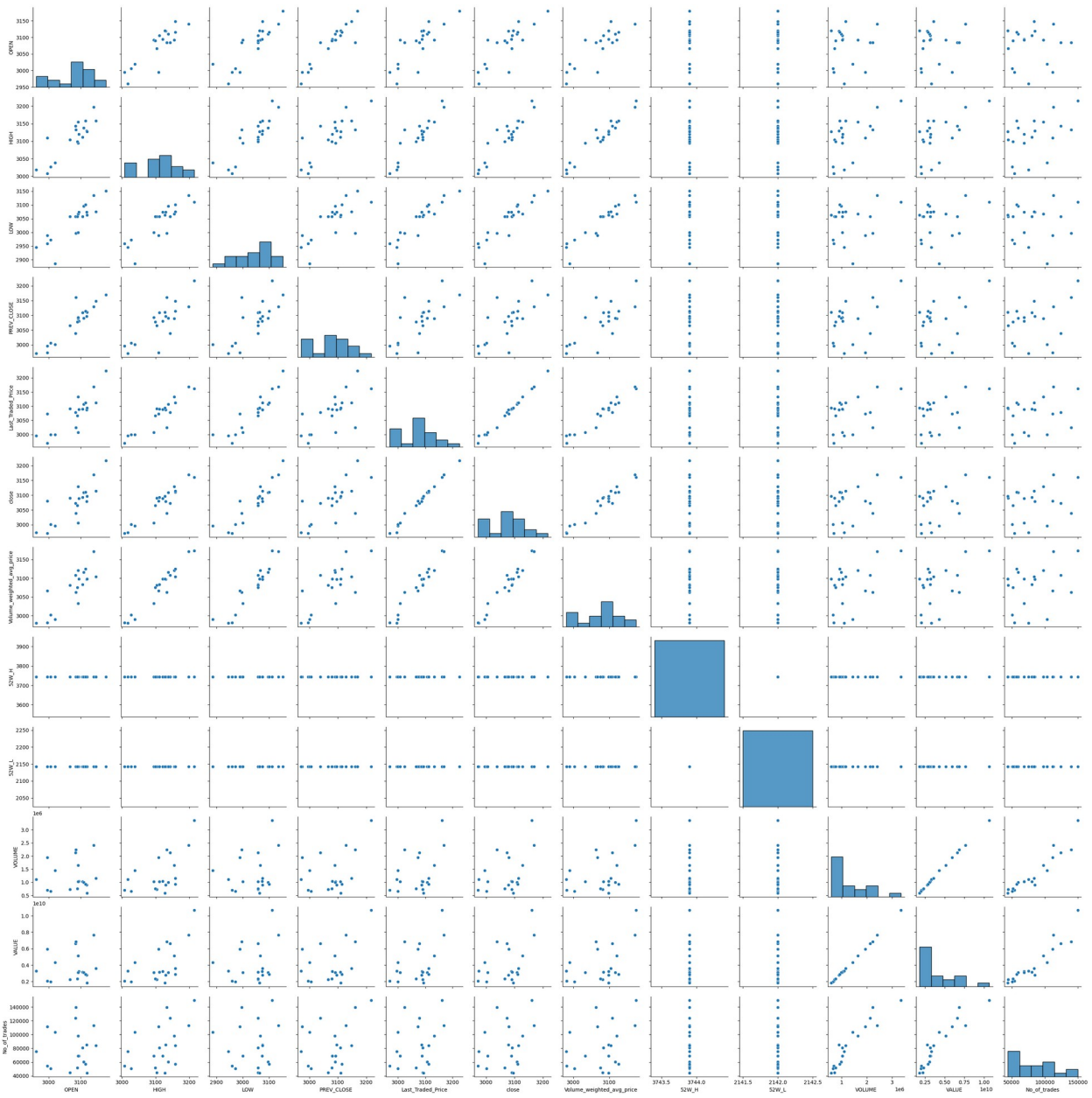
```
#histogram
df.hist(figsize=(15,12),color='y');
plt.show
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



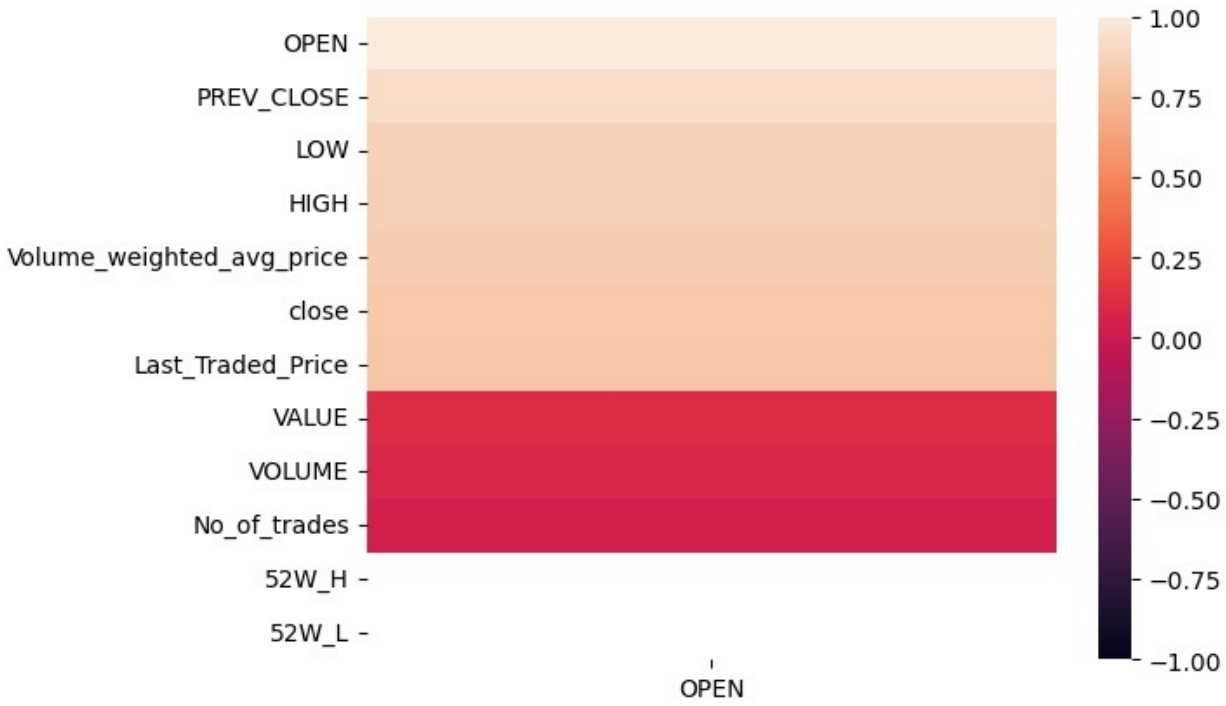

```
# pairplot of dataframe
sns.pairplot( df )
```

```
<seaborn.axisgrid.PairGrid at 0x1e11aae5e90>
```

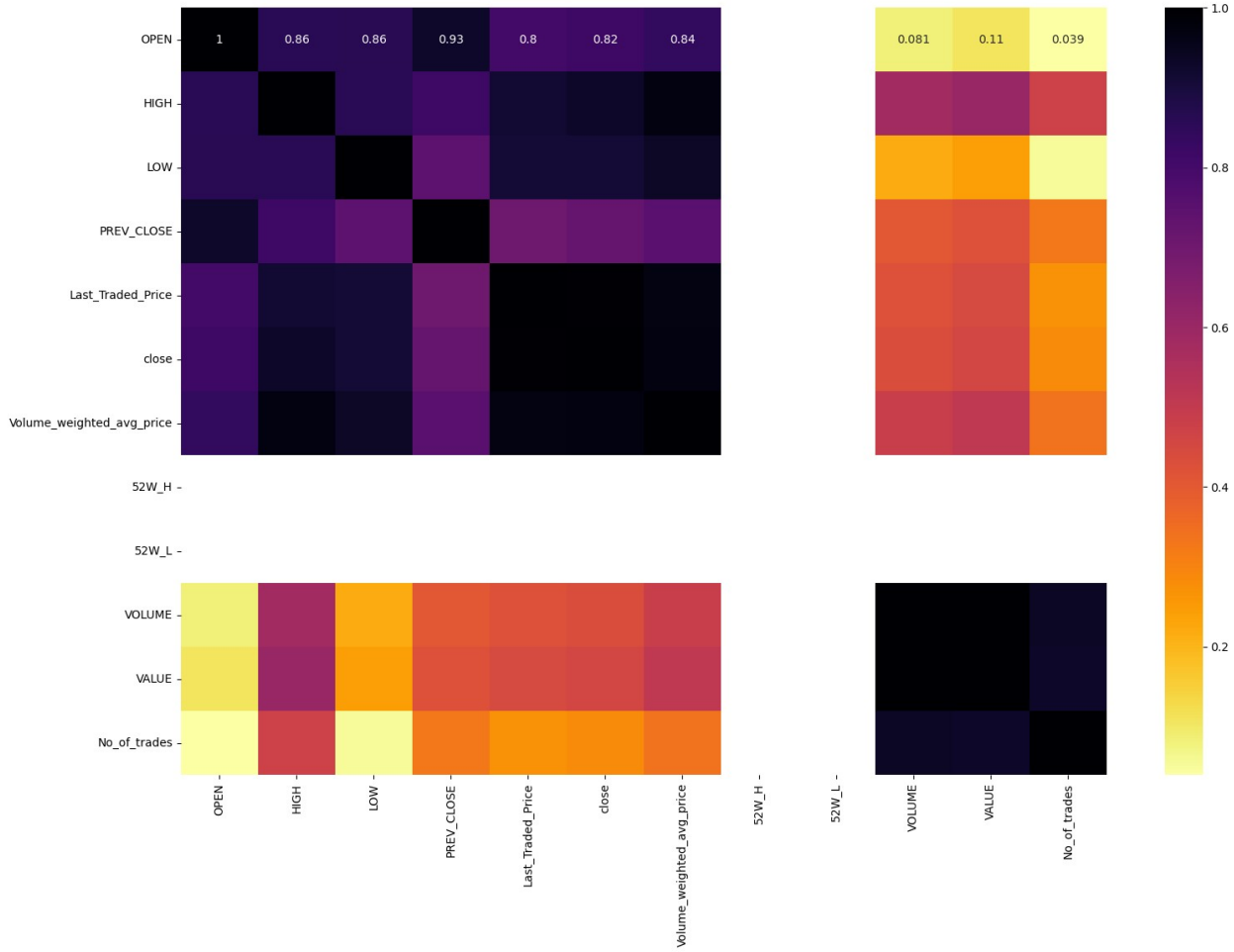



```
### Heatmap
sns.heatmap(df.drop(["Date"], axis=1).corr()
[['OPEN']].sort_values(by='OPEN', ascending=False), vmin=-1, vmax=1)
```

```
<Axes: >
```




```
plt.figure(figsize=(18,12))
sns.heatmap(df.drop(["Date"],
axis=1).corr(),annot=True,cmap="inferno_r")
plt.show()
```




```
df.columns

Index(['Date', 'OPEN', 'HIGH', 'LOW', 'PREV_CLOSE',
'Last_Traded_Price',
       'close', 'Volume_weighted_avg_price', '52W_H', '52W_L',
'VOLUME',
       'VALUE', 'No_of_trades'],
      dtype='object')

# seperating the dependent and independent variables
x=df[['HIGH','LOW','Last_Traded_Price','close']]
y=df[['OPEN']]

sc=StandardScaler()

x=sc.fit_transform(x)

x

array([[ 0.52567404,  0.20217793, -0.02786714, -0.10001145],
       [ 0.34723129, -0.73406534, -0.89292538, -0.64707296],
       [ 1.84706547,  1.02782314,  1.29802397,  1.29856319],
```

```
       [        nan,  1.65300938,  2.31376978,  2.192097  ],
       [ 1.52678362,  1.39648333,  1.40326608,  1.43334646],
       [ 0.74895625,  0.34656984,  0.84516399,  0.78876529],
       [ 0.11479817,  0.45947202,  0.13398247,  0.16400519],
       [-0.09292749, -0.86002424, -0.12354178,  0.02367202],
       [-1.76479879, -1.52054041, -1.35535283, -1.67301151],
       [-1.95056227, -1.30241647, -1.77313211, -1.7174107 ],
       [-1.39235674, -2.42298983, -1.28519142, -1.32653922],
       [-1.59550695, -1.10502966, -1.27562396, -1.23932651],
       [-0.35830389, -0.67723023, -1.14885505, -1.16242076],
       [-0.05632385,  0.20064184,  0.15949571,  0.20919723],
       [ 0.43324984,  0.79894661,  0.42260098,  0.48034946],
       [-0.19358751,  0.21907485,  0.17544148,  0.18065489],
       [-0.27960606,  0.21907485, -0.22320287, -0.21497365],
       [ 0.28775037,  0.4610081 ,  0.11165838, -0.01121306],
       [ 0.24108073,  0.2966471 ,  0.22168423,  0.26945328],
       [ 0.80386171,  0.86653431,  0.51030274,  0.5033419 ],
       [ 0.80752207,  0.47483286,  0.51030274,  0.54853393]])
```

```
x = pd.DataFrame(x)
x = x.fillna(np.median(x[0]))
x.isna().sum()
```

```
0    1
1    0
2    0
3    0
dtype: int64
```

```
x
```

```
           0         1         2         3
0   0.525674  0.202178 -0.027867 -0.100011
1   0.347231 -0.734065 -0.892925 -0.647073
2   1.847065  1.027823  1.298024  1.298563
3        NaN  1.653009  2.313770  2.192097
4   1.526784  1.396483  1.403266  1.433346
5   0.748956  0.346570  0.845164  0.788765
6   0.114798  0.459472  0.133982  0.164005
7  -0.092927 -0.860024 -0.123542  0.023672
8  -1.764799 -1.520540 -1.355353 -1.673012
9  -1.950562 -1.302416 -1.773132 -1.717411
10 -1.392357 -2.422990 -1.285191 -1.326539
11 -1.595507 -1.105030 -1.275624 -1.239327
12 -0.358304 -0.677230 -1.148855 -1.162421
13 -0.056324  0.200642  0.159496  0.209197
14  0.433250  0.798947  0.422601  0.480349
15 -0.193588  0.219075  0.175441  0.180655
16 -0.279606  0.219075 -0.223203 -0.214974
17  0.287750  0.461008  0.111658 -0.011213
```

```
18  0.241081  0.296647  0.221684  0.269453
19  0.803862  0.866534  0.510303  0.503342
20  0.807522  0.474833  0.510303  0.548534
```

```
x = x.drop(2)
y = y.drop(2)
```

```
x
```

```
           0         1         2         3
0   0.525674  0.202178 -0.027867 -0.100011
1   0.347231 -0.734065 -0.892925 -0.647073
3        NaN  1.653009  2.313770  2.192097
4   1.526784  1.396483  1.403266  1.433346
5   0.748956  0.346570  0.845164  0.788765
6   0.114798  0.459472  0.133982  0.164005
7  -0.092927 -0.860024 -0.123542  0.023672
8  -1.764799 -1.520540 -1.355353 -1.673012
9  -1.950562 -1.302416 -1.773132 -1.717411
10 -1.392357 -2.422990 -1.285191 -1.326539
11 -1.595507 -1.105030 -1.275624 -1.239327
12 -0.358304 -0.677230 -1.148855 -1.162421
13 -0.056324  0.200642  0.159496  0.209197
14  0.433250  0.798947  0.422601  0.480349
15 -0.193588  0.219075  0.175441  0.180655
16 -0.279606  0.219075 -0.223203 -0.214974
17  0.287750  0.461008  0.111658 -0.011213
18  0.241081  0.296647  0.221684  0.269453
19  0.803862  0.866534  0.510303  0.503342
20  0.807522  0.474833  0.510303  0.548534
```

```
y
```

```
       OPEN
0   3084.00
1   3085.00
3   3180.00
4   3141.00
5   3092.00
6   3094.30
7   2995.00
8   2960.70
9   2995.35
10  3020.00
11  3005.70
12  3092.00
13  3106.00
14  3110.00
15  3066.10
16  3090.00
```

```
17  3118.70
18  3120.10
19  3115.95
20  3147.90
```

```
x = x.fillna(np.mean(x[0]))
x
```

```
           0         1         2         3
0   0.525674  0.202178 -0.027867 -0.100011
1   0.347231 -0.734065 -0.892925 -0.647073
3  -0.097214  1.653009  2.313770  2.192097
4   1.526784  1.396483  1.403266  1.433346
5   0.748956  0.346570  0.845164  0.788765
6   0.114798  0.459472  0.133982  0.164005
7  -0.092927 -0.860024 -0.123542  0.023672
8  -1.764799 -1.520540 -1.355353 -1.673012
9  -1.950562 -1.302416 -1.773132 -1.717411
10 -1.392357 -2.422990 -1.285191 -1.326539
11 -1.595507 -1.105030 -1.275624 -1.239327
12 -0.358304 -0.677230 -1.148855 -1.162421
13 -0.056324  0.200642  0.159496  0.209197
14  0.433250  0.798947  0.422601  0.480349
15 -0.193588  0.219075  0.175441  0.180655
16 -0.279606  0.219075 -0.223203 -0.214974
17  0.287750  0.461008  0.111658 -0.011213
18  0.241081  0.296647  0.221684  0.269453
19  0.803862  0.866534  0.510303  0.503342
20  0.807522  0.474833  0.510303  0.548534
```

```
from sklearn.model_selection import cross_val_score
models={
 'LinearRegression':LinearRegression(),
 'Lasso':Lasso(),
 'Ridge':Ridge(),
 'GradientBoostingRegressor':GradientBoostingRegressor(),
 'AdaBoostRegressor':AdaBoostRegressor(),
 'RandomForestRegressor':RandomForestRegressor(),
 'KneghborsRegressor':KNeighborsRegressor()
}
```

```
import sklearn
sklearn.metrics.get_scorer_names()
```

```
['accuracy',
 'adjusted_mutual_info_score',
 'adjusted_rand_score',
 'average_precision',
 'balanced_accuracy',
 'completeness_score',
```

```
 'explained_variance',
 'f1',
 'f1_macro',
 'f1_micro',
 'f1_samples',
 'f1_weighted',
 'fowlkes_mallows_score',
 'homogeneity_score',
 'jaccard',
 'jaccard_macro',
 'jaccard_micro',
 'jaccard_samples',
 'jaccard_weighted',
 'matthews_corrcoef',
 'max_error',
 'mutual_info_score',
 'neg_brier_score',
 'neg_log_loss',
 'neg_mean_absolute_error',
 'neg_mean_absolute_percentage_error',
 'neg_mean_gamma_deviance',
 'neg_mean_poisson_deviance',
 'neg_mean_squared_error',
 'neg_mean_squared_log_error',
 'neg_median_absolute_error',
 'neg_negative_likelihood_ratio',
 'neg_root_mean_squared_error',
 'normalized_mutual_info_score',
 'positive_likelihood_ratio',
 'precision',
 'precision_macro',
 'precision_micro',
 'precision_samples',
 'precision_weighted',
 'r2',
 'rand_score',
 'recall',
 'recall_macro',
 'recall_micro',
 'recall_samples',
 'recall_weighted',
 'roc_auc',
 'roc_auc_ovo',
 'roc_auc_ovo_weighted',
 'roc_auc_ovr',
 'roc_auc_ovr_weighted',
 'top_k_accuracy',
 'v_measure_score']
```

```
for name, model in models.items():

scores=cross_val_score(model,x,y,scoring='neg_mean_squared_error',cv=10,n_jobs=-1)
   print('ss validaton model:{}'.format(name))
   rmse=np.sqrt(-scores)
   rmse_avarage=np.mean(rmse)
   ##print(scores)
   print('AVARAGE RMSE:',rmse_avarage)
   print('*'*100)
```

```
ss validaton model:LinearRegression
AVARAGE RMSE: 37.99514831888207
****************************************************************************************************
ss validaton model:Lasso
AVARAGE RMSE: 33.438711903662615
****************************************************************************************************
ss validaton model:Ridge
AVARAGE RMSE: 32.60484668251619
****************************************************************************************************
ss validaton model:GradientBoostingRegressor
AVARAGE RMSE: 32.61980392334731
****************************************************************************************************
ss validaton model:AdaBoostRegressor
AVARAGE RMSE: 28.858540625686807
****************************************************************************************************
ss validaton model:RandomForestRegressor
AVARAGE RMSE: 32.528517986839304
****************************************************************************************************
ss validaton model:KneghborsRegressor
AVARAGE RMSE: 33.28487992749787
****************************************************************************************************
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)

LR=LinearRegression()

LR.fit(x_train,y_train)

LinearRegression()

print("model trained with {}".format(LR))
training_score = LR.score(x_train, y_train)*100
```

```
testing_score = LR.score(x_test, y_test)*100
score = r2_score(y_test, LR.predict(x_test))*100
mae = mean_absolute_error(y_test, LR.predict(x_test))
mse = mean_squared_error(y_test, LR.predict(x_test))
rmse = np.sqrt(mse)
print("r2score: ",score)
print("training_score: ", training_score)
print("testing_score: ", testing_score)
print("mae: ", mae)
print("mse: ", mse)
print("rmse_test: ", rmse)
```

```
model trained with LinearRegression()
r2score:  75.9860738976689
training_score:  76.1524497684364
testing_score:  75.9860738976689
mae:  24.129974638666226
mse:  777.272253987584
rmse_test:  27.87960283052081
```

```
y_pred = LR.predict(x)
```