In [54]:

```
cd C:\Users\harsha.teja\Desktop\myg\congitensor
```

```
C:\Users\harsha.teja\Desktop\myg\congitensor
```

In [223]:

```
import pandas as pd
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
import numpy as np
import seaborn as sns

import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import math
import os

from mpl_finance import candlestick_ohlc
import matplotlib.dates as mpl_dates

from datetime import datetime
from sklearn import preprocessing;
from sklearn.model_selection import cross_validate
from sklearn.model_selection import train_test_split
from sklearn import linear_model;


import warnings
warnings.filterwarnings('ignore')
from pylab import rcParams
%matplotlib inline
```

```
C:\Users\harsha.teja\Anaconda3\lib\site-packages\mpl_finance.py:16: Deprec
ationWarning:


  ======================================================================

   WARNING: `mpl_finance` is deprecated:

    Please use `mplfinance` instead (no hyphen, no underscore).

    To install: `pip install --upgrade mplfinance`

   For more information, see: https://pypi.org/project/mplfinance/

  ======================================================================
```

In [166]:

```
df = pd.read_csv("cs-1.csv")
```

In [167]:

```
df.head()
```

Out[167]:

	date	open	high	low	close	volume	Name
0	2013-02-08	15.07	15.12	14.63	14.75	8407500	AAL
1	2013-02-11	14.89	15.01	14.26	14.46	8882000	AAL
2	2013-02-12	14.45	14.51	14.10	14.27	8126000	AAL
3	2013-02-13	14.30	14.94	14.25	14.66	10259500	AAL
4	2013-02-14	14.94	14.96	13.16	13.99	31879900	AAL

"1. Volatility Index - Out of all the 500 stocks in the dataset, establish a weekly" "volatilityindex which ranks stocks on the basis of intraday price movements.(Weekly volatility Index "implies that it is to be calculated on a weekly time frame and bothintraday "as well as weekly change in price needs to be used in calculating volatility)"

In [168]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 619040 entries, 0 to 619039
Data columns (total 7 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   date    619040 non-null  object 
 1   open    619029 non-null  float64
 2   high    619032 non-null  float64
 3   low     619032 non-null  float64
 4   close   619040 non-null  float64
 5   volume  619040 non-null  int64  
 6   Name    619040 non-null  object 
dtypes: float64(4), int64(1), object(2)
memory usage: 33.1+ MB
```

In [169]:

```
df['date'] = df['date'].map(lambda t: datetime.strptime(str(t), '%Y-%m-%d'))
```

In [170]:

```
#df = df.set_index('date')
```

In [171]:

```
df.head()
```

Out[171]:

	date	open	high	low	close	volume	Name
0	2013-02-08	15.07	15.12	14.63	14.75	8407500	AAL
1	2013-02-11	14.89	15.01	14.26	14.46	8882000	AAL
2	2013-02-12	14.45	14.51	14.10	14.27	8126000	AAL
3	2013-02-13	14.30	14.94	14.25	14.66	10259500	AAL
4	2013-02-14	14.94	14.96	13.16	13.99	31879900	AAL

In [172]:

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

Out[172]:

```
date       0
open      11
high       8
low        8
close      0
volume     0
Name       0
dtype: int64
```

In [173]:

```
df = df.dropna()
```

In [174]:

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

Out[174]:

```
date      0
open      0
high      0
low       0
close     0
volume    0
Name      0
dtype: int64
```

In [175]:

```
df['year'] = df['date'].dt.year
```

In [176]:

```
df.head()
```

Out[176]:

	date	open	high	low	close	volume	Name	year
0	2013-02-08	15.07	15.12	14.63	14.75	8407500	AAL	2013
1	2013-02-11	14.89	15.01	14.26	14.46	8882000	AAL	2013
2	2013-02-12	14.45	14.51	14.10	14.27	8126000	AAL	2013
3	2013-02-13	14.30	14.94	14.25	14.66	10259500	AAL	2013
4	2013-02-14	14.94	14.96	13.16	13.99	31879900	AAL	2013

In [177]:

```
df.describe()
```

Out[177]:

	open	high	low	close	volume	
count	619029.000000	619029.000000	619029.000000	619029.000000	6.190290e+05	619029.00
mean	83.023334	83.778419	82.256200	83.043305	4.321892e+06	2015.12
std	97.378769	98.207735	96.507634	97.388913	8.693671e+06	1.44
min	1.620000	1.690000	1.500000	1.590000	1.010000e+02	2013.00
25%	40.220000	40.620000	39.830000	40.240800	1.070351e+06	2014.00
50%	62.590000	63.150000	62.020000	62.620000	2.082165e+06	2015.00
75%	94.370000	95.180000	93.540000	94.410000	4.284550e+06	2016.00
max	2044.000000	2067.990000	2035.110000	2049.000000	6.182376e+08	2018.00

In [178]:

```
#For the sake of visualization, Let's create extract year from the date column
df['year'] = pd.DatetimeIndex(df["date"]).year
df['month'] = pd.DatetimeIndex(df["date"]).month
df['date'] = pd.DatetimeIndex(df["date"]).date
```

In [179]:

```
#Since the year 2017 is the most recent year with dataset of over 4 months, let's explo
re that
# Creating a ColumnDataSource instance to act as a reusable data source for ploting
```

In [180]:

```
df["Name"].unique()
```

Out[180]:

```
array(['AAL', 'AAPL', 'AAP', 'ABBV', 'ABC', 'ABT', 'ACN', 'ADBE', 'ADI',
       'ADM', 'ADP', 'ADSK', 'ADS', 'AEE', 'AEP', 'AES', 'AET', 'AFL',
       'AGN', 'AIG', 'AIV', 'AIZ', 'AJG', 'AKAM', 'ALB', 'ALGN', 'ALK',
       'ALLE', 'ALL', 'ALXN', 'AMAT', 'AMD', 'AME', 'AMGN', 'AMG', 'AMP',
       'AMT', 'AMZN', 'ANDV', 'ANSS', 'ANTM', 'AON', 'AOS', 'APA', 'APC',
       'APD', 'APH', 'APTV', 'ARE', 'ARNC', 'ATVI', 'AVB', 'AVGO', 'AVY',
       'AWK', 'AXP', 'AYI', 'AZO', 'A', 'BAC', 'BAX', 'BA', 'BBT', 'BBY',
       'BDX', 'BEN', 'BF.B', 'BHF', 'BHGE', 'BIIB', 'BK', 'BLK', 'BLL',
       'BMY', 'BRK.B', 'BSX', 'BWA', 'BXP', 'CAG', 'CAH', 'CAT', 'CA',
       'CBG', 'CBOE', 'CBS', 'CB', 'CCI', 'CCL', 'CDNS', 'CELG', 'CERN',
       'CFG', 'CF', 'CHD', 'CHK', 'CHRW', 'CHTR', 'CINF', 'CI', 'CLX',
       'CL', 'CMA', 'CMCSA', 'CME', 'CMG', 'CMI', 'CMS', 'CNC', 'CNP',
       'COF', 'COG', 'COL', 'COO', 'COP', 'COST', 'COTY', 'CPB', 'CRM',
       'CSCO', 'CSRA', 'CSX', 'CTAS', 'CTL', 'CTSH', 'CTXS', 'CVS', 'CVX',
       'CXO', 'C', 'DAL', 'DE', 'DFS', 'DGX', 'DG', 'DHI', 'DHR', 'DISCA',
       'DISCK', 'DISH', 'DIS', 'DLR', 'DLTR', 'DOV', 'DPS', 'DRE', 'DRI',
       'DTE', 'DUK', 'DVA', 'DVN', 'DWDP', 'DXC', 'D', 'EA', 'EBAY',
       'ECL', 'ED', 'EFX', 'EIX', 'EL', 'EMN', 'EMR', 'EOG', 'EQIX',
       'EQR', 'EQT', 'ESRX', 'ESS', 'ES', 'ETFC', 'ETN', 'ETR', 'EVHC',
       'EW', 'EXC', 'EXPD', 'EXPE', 'EXR', 'FAST', 'FBHS', 'FB', 'FCX',
       'FDX', 'FE', 'FFIV', 'FISV', 'FIS', 'FITB', 'FLIR', 'FLR', 'FLS',
       'FL', 'FMC', 'FOXA', 'FOX', 'FRT', 'FTI', 'FTV', 'F', 'GD', 'GE',
       'GGP', 'GILD', 'GIS', 'GLW', 'GM', 'GOOGL', 'GOOG', 'GPC', 'GPN',
       'GPS', 'GRMN', 'GS', 'GT', 'GWW', 'HAL', 'HAS', 'HBAN', 'HBI',
       'HCA', 'HCN', 'HCP', 'HD', 'HES', 'HIG', 'HII', 'HLT', 'HOG',
       'HOLX', 'HON', 'HPE', 'HPQ', 'HP', 'HRB', 'HRL', 'HRS', 'HSIC',
       'HST', 'HSY', 'HUM', 'IBM', 'ICE', 'IDXX', 'IFF', 'ILMN', 'INCY',
       'INFO', 'INTC', 'INTU', 'IPG', 'IP', 'IQV', 'IRM', 'IR', 'ISRG',
       'ITW', 'IT', 'IVZ', 'JBHT', 'JCI', 'JEC', 'JNJ', 'JNPR', 'JPM',
       'JWN', 'KEY', 'KHC', 'KIM', 'KLAC', 'KMB', 'KMI', 'KMX', 'KORS',
       'KO', 'KR', 'KSS', 'KSU', 'K', 'LB', 'LEG', 'LEN', 'LH', 'LKQ',
       'LLL', 'LLY', 'LMT', 'LNC', 'LNT', 'LOW', 'LRCX', 'LUK', 'LUV',
       'LYB', 'L', 'MAA', 'MAC', 'MAR', 'MAS', 'MAT', 'MA', 'MCD', 'MCHP',
       'MCK', 'MCO', 'MDLZ', 'MDT', 'MET', 'MGM', 'MHK', 'MKC', 'MLM',
       'MMC', 'MMM', 'MNST', 'MON', 'MOS', 'MO', 'MPC', 'MRK', 'MRO',
       'MSFT', 'MSI', 'MS', 'MTB', 'MTD', 'MU', 'MYL', 'M', 'NAVI', 'NBL',
       'NCLH', 'NDAQ', 'NEE', 'NEM', 'NFLX', 'NFX', 'NI', 'NKE', 'NLSN',
       'NOC', 'NOV', 'NRG', 'NSC', 'NTAP', 'NTRS', 'NUE', 'NVDA', 'NWL',
       'NWSA', 'NWS', 'OKE', 'OMC', 'ORCL', 'ORLY', 'OXY', 'O', 'PAYX',
       'PBCT', 'PCAR', 'PCG', 'PCLN', 'PDCO', 'PEG', 'PEP', 'PFE', 'PFG',
       'PGR', 'PG', 'PHM', 'PH', 'PKG', 'PKI', 'PLD', 'PM', 'PNC', 'PNR',
       'PNW', 'PPG', 'PPL', 'PRGO', 'PRU', 'PSA', 'PSX', 'PVH', 'PWR',
       'PXD', 'PX', 'PYPL', 'QCOM', 'QRVO', 'RCL', 'REGN', 'REG', 'RE',
       'RF', 'RHI', 'RHT', 'RJF', 'RL', 'RMD', 'ROK', 'ROP', 'ROST',
       'RRC', 'RSG', 'RTN', 'SBAC', 'SBUX', 'SCG', 'SCHW', 'SEE', 'SHW',
       'SIG', 'SJM', 'SLB', 'SLG', 'SNA', 'SNI', 'SNPS', 'SO', 'SPGI',
       'SPG', 'SRCL', 'SRE', 'STI', 'STT', 'STX', 'STZ', 'SWKS', 'SWK',
       'SYF', 'SYK', 'SYMC', 'SYY', 'TAP', 'TDG', 'TEL', 'TGT', 'TIF',
       'TJX', 'TMK', 'TMO', 'TPR', 'TRIP', 'TROW', 'TRV', 'TSCO', 'TSN',
       'TSS', 'TWX', 'TXN', 'TXT', 'T', 'UAA', 'UAL', 'UA', 'UDR', 'UHS',
       'ULTA', 'UNH', 'UNM', 'UNP', 'UPS', 'URI', 'USB', 'UTX', 'VAR',
       'VFC', 'VIAB', 'VLO', 'VMC', 'VNO', 'VRSK', 'VRSN', 'VRTX', 'VTR',
       'VZ', 'V', 'WAT', 'WBA', 'WDC', 'WEC', 'WFC', 'WHR', 'WLTW', 'WMB',
       'WMT', 'WM', 'WRK', 'WU', 'WYNN', 'WYN', 'WY', 'XEC', 'XEL',
       'XLNX', 'XL', 'XOM', 'XRAY', 'XRX', 'XYL', 'YUM', 'ZBH', 'ZION',
       'ZTS'], dtype=object)
```

In [182]:

```
#Give an exploratory analysis on any one stock describing it’s key statisticaltendencies.
```

In [183]:

```
#We'll focus on one
walmart = df.loc[df['Name'] == 'WMT']
walmart.head()
```

Out[183]:

	date	open	high	low	close	volume	Name	year	month
595716	2013-02-08	71.20	71.64	71.070	71.48	5906823	WMT	2013	2
595717	2013-02-11	71.25	71.51	70.530	71.40	6202534	WMT	2013	2
595718	2013-02-12	71.49	71.66	71.100	71.40	4761910	WMT	2013	2
595719	2013-02-13	71.29	71.70	71.210	71.39	3969807	WMT	2013	2
595720	2013-02-14	71.10	71.23	70.755	70.82	6820952	WMT	2013	2

In [184]:

```
walmart.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1259 entries, 595716 to 596974
Data columns (total 9 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   date    1259 non-null   object 
 1   open    1259 non-null   float64
 2   high    1259 non-null   float64
 3   low     1259 non-null   float64
 4   close   1259 non-null   float64
 5   volume  1259 non-null   int64  
 6   Name    1259 non-null   object 
 7   year    1259 non-null   int64  
 8   month   1259 non-null   int64  
dtypes: float64(4), int64(3), object(2)
memory usage: 98.4+ KB
```

In [185]:

```
#Create a copy to avoid the SettingWarning .loc issue
walmart_df = walmart.copy()

# Change to datetime datatype.
walmart_df.loc[:, 'date'] = pd.to_datetime(walmart.loc[:,'date'], format="%Y/%m/%d")
```

In [186]:

```
walmart_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1259 entries, 595716 to 596974
Data columns (total 9 columns):
 #   Column  Non-Null Count  Dtype         
---  ------  --------------  -----         
 0   date    1259 non-null   datetime64[ns]
 1   open    1259 non-null   float64       
 2   high    1259 non-null   float64       
 3   low     1259 non-null   float64       
 4   close   1259 non-null   float64       
 5   volume  1259 non-null   int64         
 6   Name    1259 non-null   object        
 7   year    1259 non-null   int64         
 8   month   1259 non-null   int64         
dtypes: datetime64[ns](1), float64(4), int64(3), object(1)
memory usage: 98.4+ KB
```

In [195]:

```
#Let's calculate returns
walmart_df['Returns'] = (walmart_df['open'] - walmart_df['close'])/walmart_df['open']
```

In [196]:

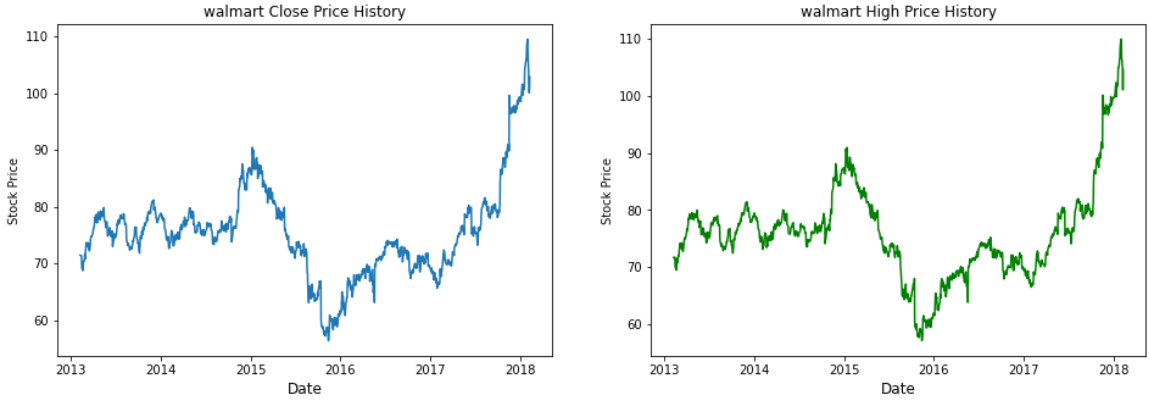
```
walmart_df['Returns'].hist()

plt.title('walmart Stock Price Returns Distribution')

plt.show()
```

In [203]:

```
# Let us plot Walmart Stock Price
# First Subplot
f, (ax1, ax2) = plt.subplots(1,2, figsize=(16,5))
ax1.plot(walmart_df["date"], walmart_df["close"])
ax1.set_xlabel("Date", fontsize=12)
ax1.set_ylabel("Stock Price")
ax1.set_title("walmart Close Price History")
ax2.plot(walmart_df["date"], walmart_df["high"], color="green")
ax2.set_xlabel("Date", fontsize=12)
ax2.set_ylabel("Stock Price")
ax2.set_title("walmart High Price History")
```

Out[203]:

```
Text(0.5, 1.0, 'walmart High Price History')
```

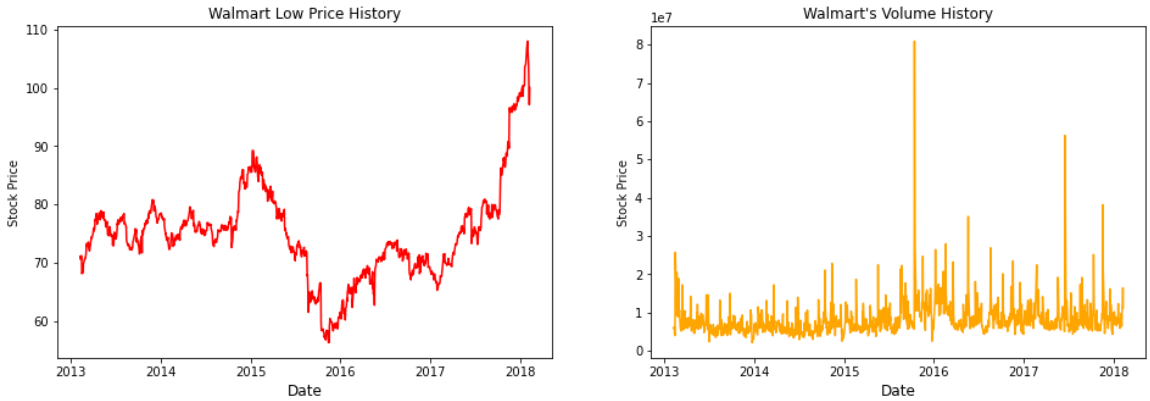


In [202]:

```
# Second Subplot
f, (ax1, ax2) = plt.subplots(1,2, figsize=(16,5))
ax1.plot(walmart_df["date"], walmart_df["low"], color="red")
ax1.set_xlabel("Date", fontsize=12)
ax1.set_ylabel("Stock Price")
ax1.set_title("Walmart Low Price History")
ax2.plot(walmart_df["date"], walmart_df["volume"], color="orange")
ax2.set_xlabel("Date", fontsize=12)
ax2.set_ylabel("Stock Price")
ax2.set_title("Walmart's Volume History")
```

Out[202]:

```
Text(0.5, 1.0, "Walmart's Volume History")
```

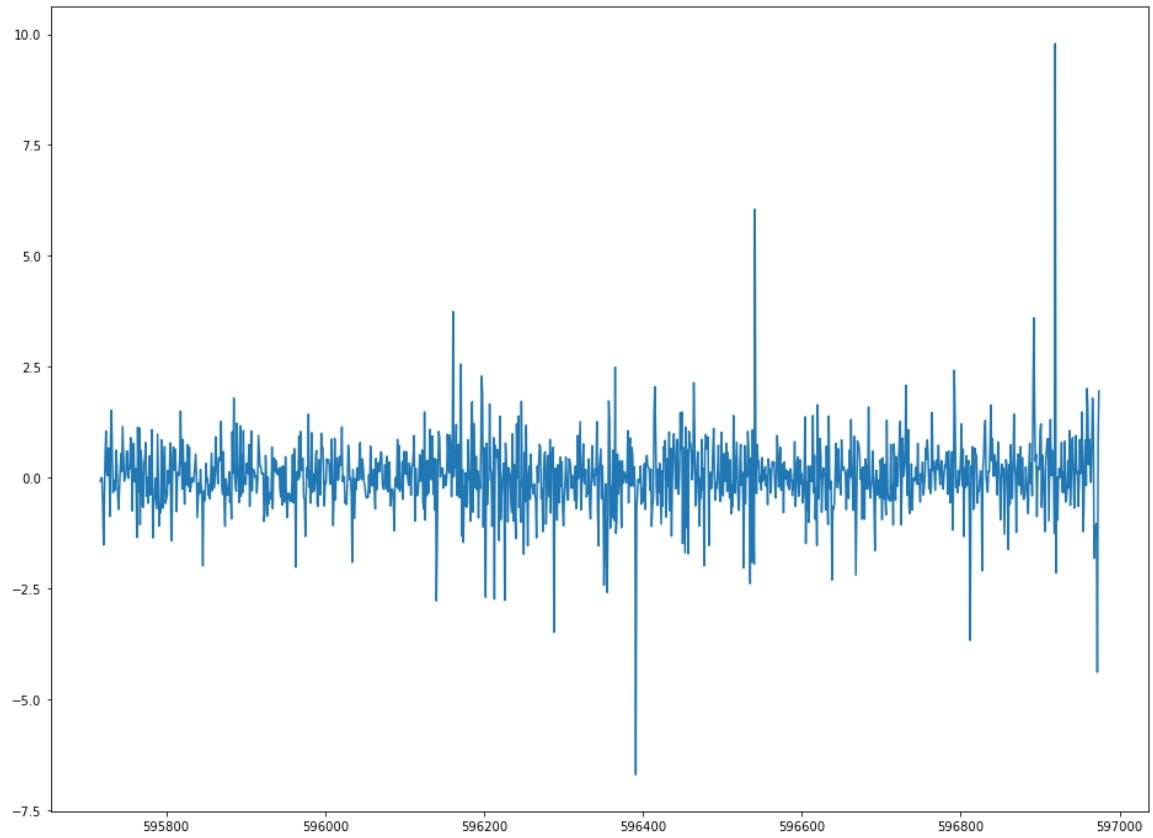


In [204]:

```
walmart_df['First Difference'] = walmart_df['close'] - walmart_df['close'].shift()
walmart_df['First Difference'].plot(figsize=(16, 12))
```

Out[204]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1519b332fd0>
```



In [210]:

```
#Adding the new features to understand the price variations of stock within a day and f
rom previous day, these additional features will help in predicting the day close stock
price with atmost accuracy as well as these features helps the model to classsify the v
olatility of a stock
```

In [212]:

```
walmart_df['changeduringday'] = ((walmart_df['high'] - walmart_df['low'] )/ walmart_df[
'low'])*100

walmart_df['changefrompreviousday'] = (abs(walmart_df['close'].shift() - walmart_df['cl
ose'] )/ walmart_df['close'])*100
```

In [213]:

```
print("**The new features 'changeduring day & change from previous day are added to the dataset. Note: The first row for change from previous day for each stock is NA or blank always")
walmart_df.head()
```

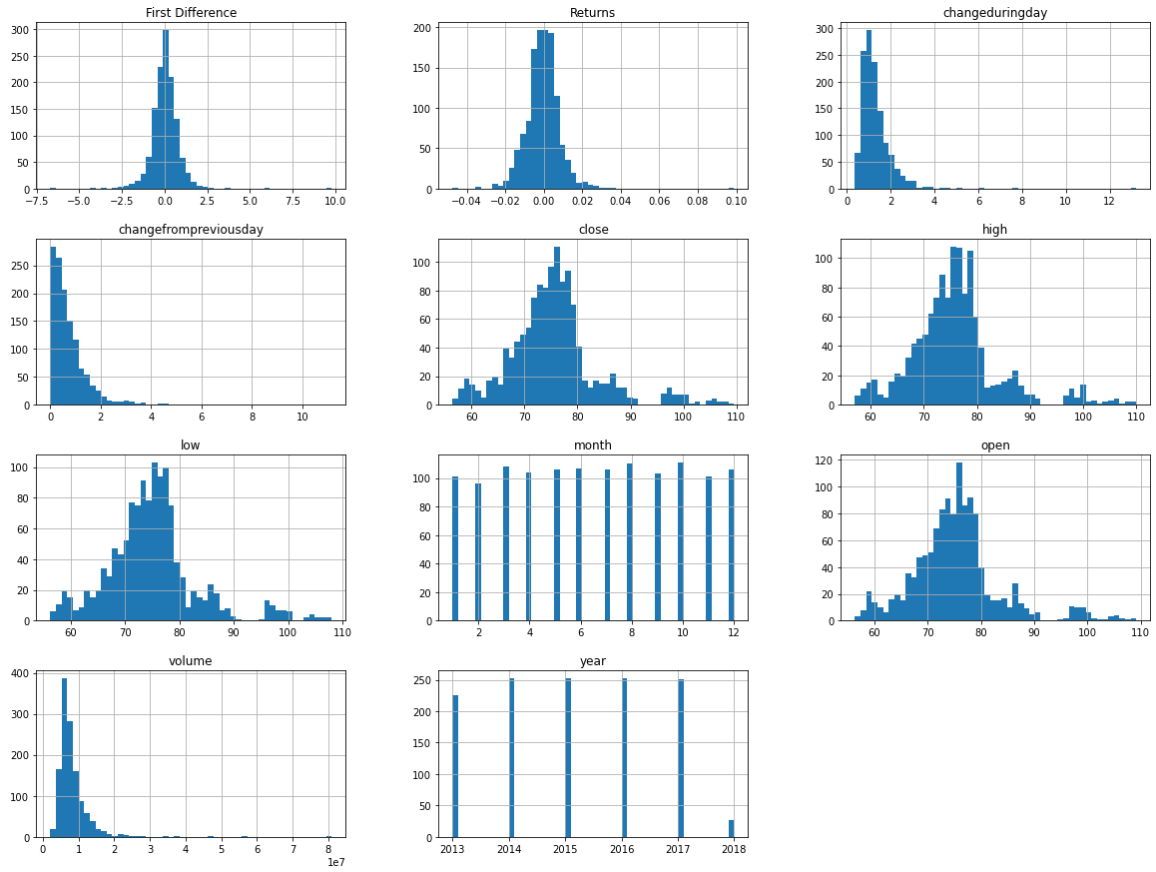
```
**The new features 'changeduring day & change from previous day are added
to the dataset. Note: The first row for change from previous day for each
stock is NA or blank always
```

Out[213]:

	date	open	high	low	close	volume	Name	year	month	Returns	Fi Differer
595716	2013-02-08	71.20	71.64	71.070	71.48	5906823	WMT	2013	2	-0.003933	N
595717	2013-02-11	71.25	71.51	70.530	71.40	6202534	WMT	2013	2	-0.002105	-0
595718	2013-02-12	71.49	71.66	71.100	71.40	4761910	WMT	2013	2	0.001259	0
595719	2013-02-13	71.29	71.70	71.210	71.39	3969807	WMT	2013	2	-0.001403	-0
595720	2013-02-14	71.10	71.23	70.755	70.82	6820952	WMT	2013	2	0.003938	-0

In [214]:

```
walmart_df.hist(bins=50, figsize=(20,15))
plt.show()
```



In [216]:

```
corr_matrix = walmart_df.corr()
```

In [217]:

```
corr_matrix["close"].sort_values(ascending=False)
```

Out[217]:

```
close                    1.000000
low                      0.998538
high                     0.998479
open                     0.996840
year                     0.114433
First Difference         0.093011
month                   -0.025481
changefrompreviousday   -0.049879
Returns                 -0.052419
changeduringday         -0.062056
volume                  -0.190085
Name: close, dtype: float64
```

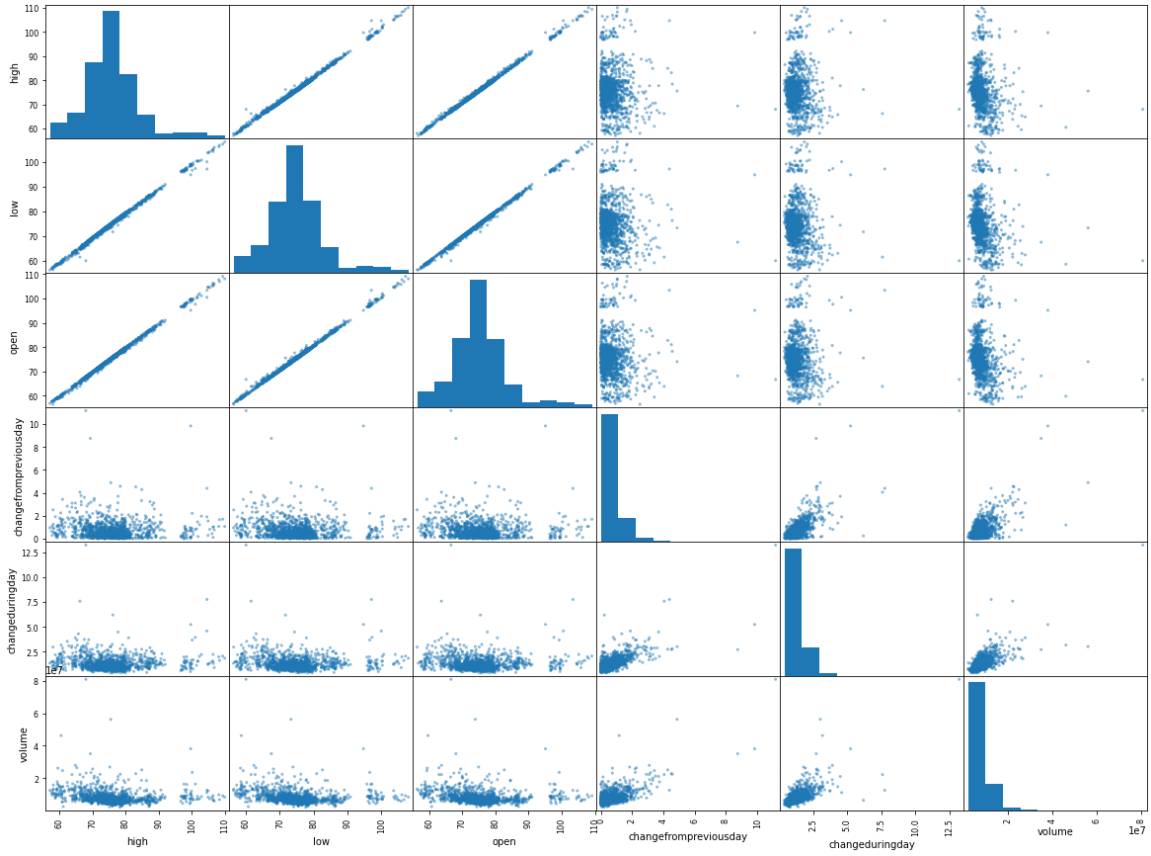
In [218]:

```
from pandas.plotting import scatter_matrix

attributes = ["high", "low", "open", "changefrompreviousday", "changeduringday", "volum
e"]

scatter_matrix(walmart_df[attributes], figsize=(20, 15))
```

Out[218]:

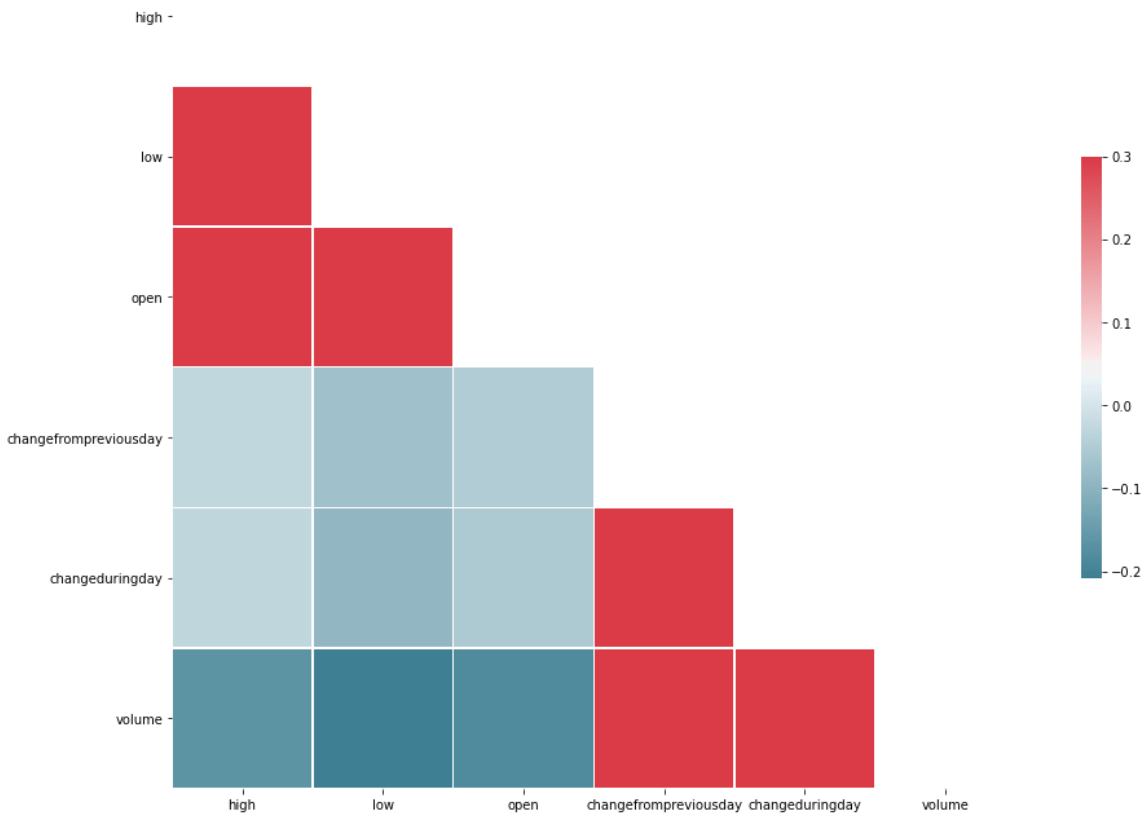
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x00000151834F346
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000151F336307
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000015182EBBA9
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001519B45F5E
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001519B461A6
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000015183213B8
0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x00000151832138B
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000151835848E
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000015182FD691
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000151F33996A
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000015183454A6
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000151F02FCA9
0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x00000151FE485D3
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000151897E5E2
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000151861BD28
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001518E3E716
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000151928C988
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000151E54F997
0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000001519266797
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000151835897C
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000015182FB2BE
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001518E33C5B
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000151832E488
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001519264EC7
0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000015182C58E2
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000015188495E5
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000151926455B
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001519ADA9AF
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000015182FD407
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000151D9EF116
```

```
0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x00000151897D89A
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001519293D97
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001518EA04FD
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001519289255
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000151926D107
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000151835908E
0>]],
      dtype=object)
```



In [219]:

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

%matplotlib inline
corr = walmart_df[["high", "low", "open", "changefrompreviousday", "changeduringday",
"volume"]].corr()

# generate a mask for the lower triangle
mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

# set up the matplotlib figure
f, ax = plt.subplots(figsize=(18, 12))

# generate a custom diverging colormap
cmap = sns.diverging_palette(220, 10, as_cmap=True)

# draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3,
            square=True,
            linewidths=.5, cbar_kws={"shrink": .5}, ax=ax);
```



In [205]:

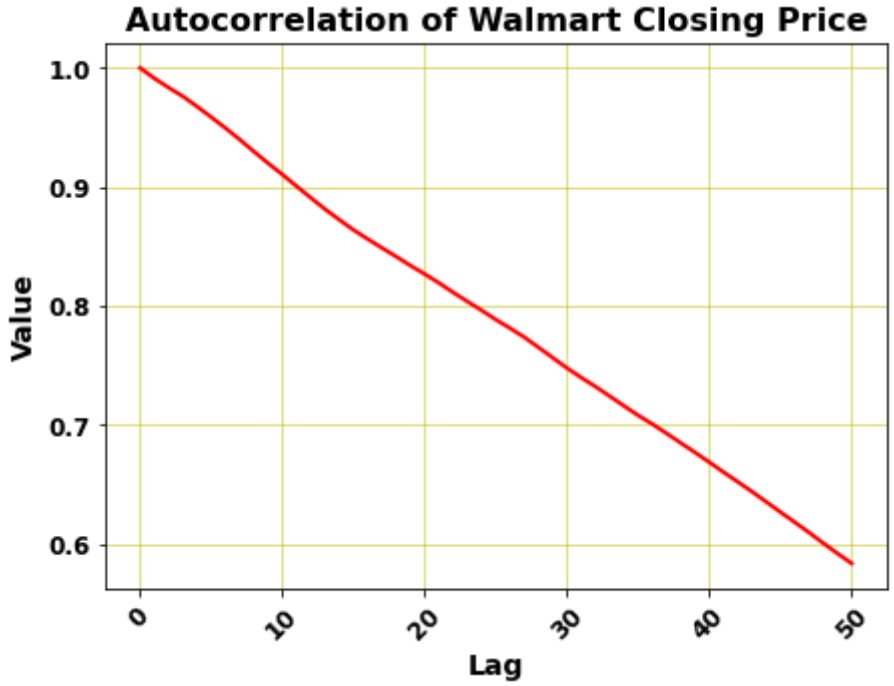
```
df_close = pd.DataFrame(walmart_df['close'])
df_close.index = pd.to_datetime(walmart_df['date'])
```

In [207]:

```
#Plotting again the closing price graph for walmart
from pandas.plotting import register_matplotlib_converters
register_matplotlib_converters()

plt.figure(figsize=(8, 6))
plt.plot(df_close, color='g')
plt.title('Walmart Closing Price', weight='bold', fontsize=16)
plt.xlabel('Date', weight='bold', fontsize=14)
plt.ylabel('Stock Price', weight='bold', fontsize=14)
plt.xticks(weight='bold', fontsize=12, rotation=45)
plt.yticks(weight='bold', fontsize=12)
plt.grid(color = 'y', linewidth = 0.5)
```

In [209]:

```
#Autocorrelation plot
from statsmodels.tsa import stattools

acf_djia, confint_djia, qstat_djia, pvalues_djia = stattools.acf(df_close,
                                                                 unbiased=True,
                                                                 nlags=50,
                                                                 qstat=True,
                                                                 fft=True,
                                                                 alpha = 0.05)

plt.figure(figsize=(7, 5))
plt.plot(pd.Series(acf_djia), color='r', linewidth=2)
plt.title('Autocorrelation of Walmart Closing Price', weight='bold', fontsize=16)
plt.xlabel('Lag', weight='bold', fontsize=14)
plt.ylabel('Value', weight='bold', fontsize=14)
plt.xticks(weight='bold', fontsize=12, rotation=45)
plt.yticks(weight='bold', fontsize=12)
plt.grid(color = 'y', linewidth = 0.5)
```



In [188]:

```
def prepare_data(df,forecast_col,forecast_out,test_size):
    label = df[forecast_col].shift(-forecast_out);#creating new column called label wit
h the last 5 rows are nan
    X = np.array(df[[forecast_col]]); #creating the feature array
    X = preprocessing.scale(X) #processing the feature array
    X_lately = X[-forecast_out:] #creating the column i want to use later in the predic
ting method
    X = X[:-forecast_out] # X that will contain the training and testing
    label.dropna(inplace=True); #dropping na values
    y = np.array(label)  # assigning Y
    X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size=test_size) #cro
ss validation

    response = [X_train,X_test , Y_train, Y_test , X_lately];
    return response;
```

In [191]:

```
forecast_col = 'close'#choosing which column to forecast
forecast_out = 50 #how far to forecast
test_size = 0.2; #the size of my test set

X_train, X_test, Y_train, Y_test , X_lately =prepare_data(walmart_df,forecast_col,forec
ast_out,test_size); #calling the method were the cross validation and data preperation
 is in

learner = linear_model.LinearRegression(); #initializing linear regression model
learner.fit(X_train,Y_train); #training the linear regression model

score=learner.score(X_test,Y_test);#testing the linear regression model
forecast= learner.predict(X_lately); #set that will contain the forecasted data

response={};#creting json object
response['test_score']=score;
response['forecast_set']=forecast;

print(response);
```

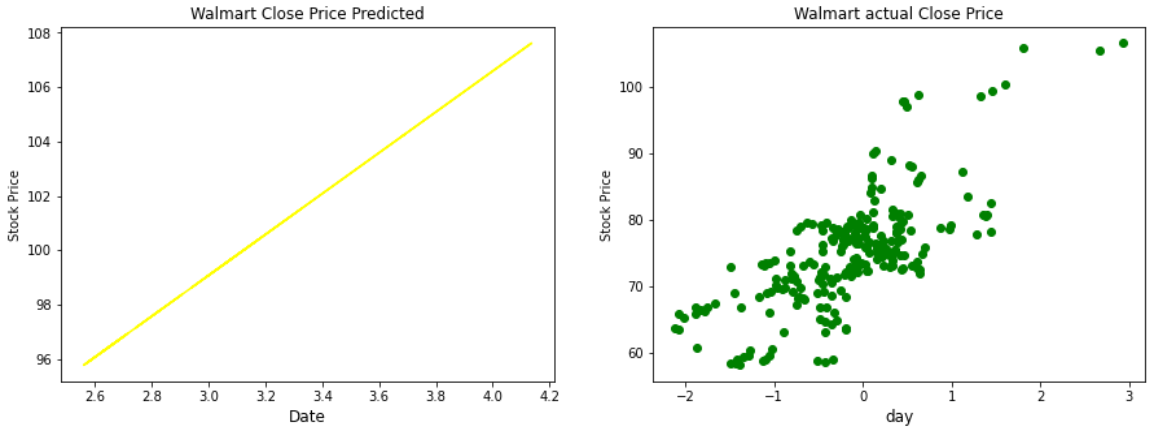
```
{'test_score': 0.534604861004895, 'forecast_set': array([ 95.82990133,  9
5.96671147,  96.68724487,  96.38626256,
        96.49571067,  96.18560769,  96.93350312,  96.43186594,
        95.97583214,  95.7660566 ,  96.11264228,  95.90286674,
        96.86965838,  96.2950558 ,  96.27681445,  96.99734785,
        97.81820868,  97.7726053 ,  97.14327866,  97.2800888 ,
        98.14655301,  98.23775977,  98.36544923,  97.7726053 ,
        97.62667449,  98.41105261,  98.4931387 ,  99.03125858,
       100.38111861,  99.26839615,  98.61170748,  98.93093114,
        99.70618859,  99.54201643, 101.37527229, 102.83458043,
       103.09908004, 103.88345816, 104.29388858, 104.19356115,
       104.9323359 , 106.56493688, 107.62293529, 105.96297227,
       104.9323359 , 103.9473029 , 102.9987526 ,  98.99477587,
        99.73355062, 101.51208243])}
```

In [192]:

```
f, (ax1, ax2) = plt.subplots(1, 2, figsize=(15,5))
ax1.plot(X_lately, forecast, color="yellow")
ax1.set_xlabel("Date", fontsize=12)
ax1.set_ylabel("Stock Price")
ax1.set_title("Walmart Close Price Predicted")

ax2.scatter(X_test, Y_test, color="green")
ax2.set_xlabel("day", fontsize=12)
ax2.set_ylabel("Stock Price")
ax2.set_title("Walmart actual Close Price")
```

Out[192]:

```
Text(0.5, 1.0, 'Walmart actual Close Price')
```



In [193]:

```
score
```

Out[193]:

```
0.534604861004895
```

In [113]:

```
count_df = pd.DataFrame(df.Name.value_counts(),columns=["Name","Count"]).reset_index()
```

In [114]:

```
lis_valid_share = list(count_df["index"])
```

In [115]:

```
final_df = df[df.Name.isin(lis_valid_share)]
```

In [117]:

```
data_by_year = final_df.groupby("year")
data_by_asset = final_df.groupby("Name")
```

In [134]:

```
df.year.value_counts()
```

Out[134]:

```
2017    126031
2016    125320
2015    123713
2014    122397
2013    108438
2018     13130
Name: year, dtype: int64
```

In [153]:

```
#for year 2013
year = 2013
data2 = data_by_year.get_group(year)
final_pivot = data2.pivot(index = "date",columns = "Name",values = "close")
daily_volatility = final_pivot.pct_change().apply(lambda x:np.log(1+x)).std()
weekly_volatility  = daily_volatility.apply(lambda x: x*np.sqrt(5))
WV = pd.DataFrame(weekly_volatility).reset_index()
WV.columns = ["Name","Volatility"]
```

In [221]:

```
#ToP Volatile stocks
weekly_top10_volatility = WV.sort_values(by = "Volatility",ascending = False)
weekly_low10_volatility = WV.sort_values(by = "Volatility",ascending = True)
print("weekly_hig10_volatility.")
weekly_top10_volatility.head(10)
```

```
weekly_hig10_volatility.
```

Out[221]:

	Name	Volatility
455	VRTX	0.086312
232	INCY	0.074054
32	AMD	0.070818
168	EXPE	0.068285
438	ULTA	0.066162
318	NFLX	0.065282
172	FB	0.062368
62	BBY	0.059760
317	NEM	0.057881
353	PHM	0.057730

In [220]:

```
#Least Volatile stocks
weekly_low10_volatility = WV.sort_values(by = "Volatility",ascending = True)
print("weekly_low10_volatility.")
weekly_low10_volatility.head(10)
```

low volatilty

Out[220]:

	Name	Volatility
286	MCD	0.017335
466	WMT	0.017586
444	USB	0.017917
247	JNJ	0.018365
251	K	0.018410
475	XOM	0.018608
143	DUK	0.018644
401	SO	0.018658
369	PX	0.018774
149	ED	0.018921

In [ ]:

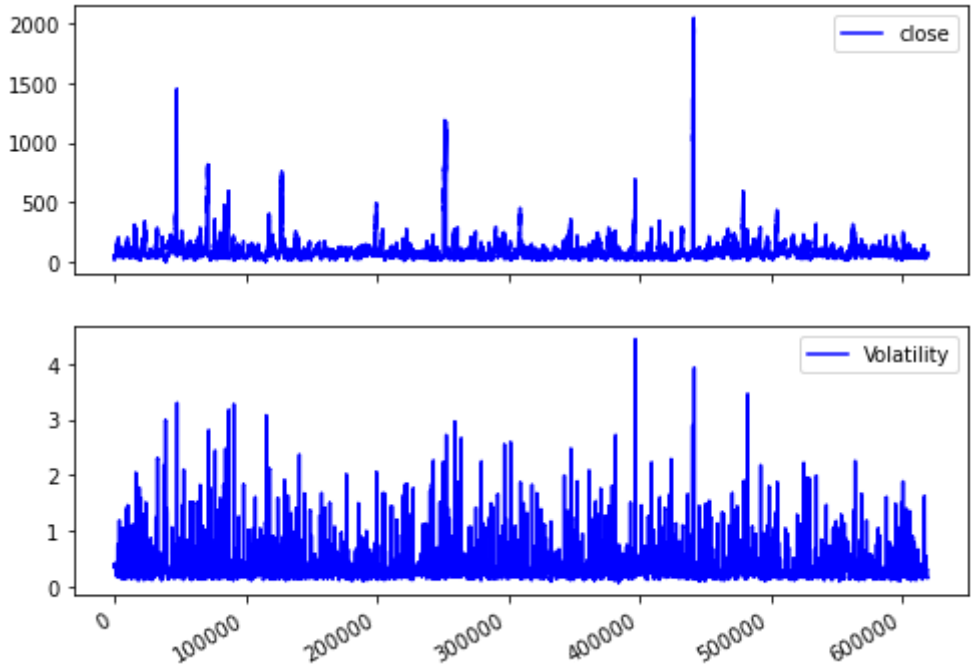
```
# Compute the logarithmic returns using the Closing price
df['Log_Ret'] = np.log(df['close'] / df['close'].shift(1))

# Compute Volatility using the pandas rolling standard deviation function
df['Volatility'] = df['Log_Ret'].rolling(window=252).std() * np.sqrt(252)
#print(df.tail(15))
```

In [69]:

```
# Plot the NIFTY Price series and the Volatility
df[['close', 'Volatility']].plot(subplots=True, color='blue',figsize=(8, 6))
```

Out[69]:

```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x00000151CD6F23A0
>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x00000151D9ECB520
>],
      dtype=object)
```



In [ ]: