Two_Stock_Comparing_Dividend

September 29, 2021

Simple Stock Analysis

Stock analysis is the evaluation or examination of the stock market. There are many trading tools to use to analysis stocks such as fundamental and technical analysis. Fundamental analysis is more focused on data from the financial statements, economic reports, and company assets. Technical analysis is based on the study of the past of historical price to predict the future price movement. However, this tutorial is not to get rich quick. Therefore, do not use your money to trade based on this stock analysis. Please do not use this method to invest with your money and I am not responsible for you loss.

Simple stock is a basic stock analysis tutorial. There are 7 parts in this tutorial. 1. Import Libraries 2. Get data from Yahoo 3. Analysis Data 4. Understand the Data based on Statistics 5. Calculate Prices 6. Plot Charts 7. Calculate Holding Period Return

I. Import Libraries

[1]:
```
# Libaries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

import warnings
warnings.filterwarnings("ignore")

import fix_yahoo_finance as yf
yf.pdr_override()
```

II. Get Data from Yahoo!

This section we will pull the data from the website in Yahoo. We will be using the company of Apple and the symbol is 'AAPL'. Also, we will have a starting date and ending date.

[2]:
```
stock = 'AAPL'
start = '2015-01-01'
end = '2017-01-01'
df = yf.download(stock, start, end)
```

```
[**********************100%**********************]  1 of 1 downloaded
```

III. Analysis Data

[3]:
```
df.head() # the first 5 rows
```

[3]:
```
                 Open        High         Low       Close   Adj Close  \
Date
2015-01-02  111.389999  111.440002  107.349998  109.330002  102.319397
2015-01-05  108.290001  108.650002  105.410004  106.250000   99.436890
2015-01-06  106.540001  107.430000  104.629997  106.260002   99.446251
2015-01-07  107.199997  108.199997  106.699997  107.750000  100.840714
2015-01-08  109.230003  112.150002  108.699997  111.889999  104.715256

              Volume
Date
2015-01-02  53204600
2015-01-05  64285500
2015-01-06  65797100
2015-01-07  40105900
2015-01-08  59364500
```

[4]:
```
df.tail() # the last 5 rows
```

[4]:
```
                 Open        High         Low       Close   Adj Close  \
Date
2016-12-23  115.589996  116.519997  115.589996  116.519997  113.373917
2016-12-27  116.519997  117.800003  116.489998  117.260002  114.093948
2016-12-28  117.519997  118.019997  116.199997  116.760002  113.607445
2016-12-29  116.449997  117.110001  116.400002  116.730003  113.578247
2016-12-30  116.650002  117.199997  115.430000  115.820000  112.692825

              Volume
Date
2016-12-23  14249500
2016-12-27  18296900
2016-12-28  20905900
2016-12-29  15039500
2016-12-30  30586300
```

[5]:
```
df.shape # (rows, columns)
```

[5]:
```
(504, 6)
```

[6]:
```
df.columns # Shows names of columns
```

[6]:
```
Index(['Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume'], dtype='object')
```

[7]:
```
df.dtypes # Shows data types
```

[7]:
```
Open          float64
High          float64
Low           float64
Close         float64
Adj Close     float64
Volume          int32
dtype: object
```

[8]:
```
df.info() # Shows information about DataFrame
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 504 entries, 2015-01-02 to 2016-12-30
Data columns (total 6 columns):
Open          504 non-null float64
High          504 non-null float64
Low           504 non-null float64
Close         504 non-null float64
Adj Close     504 non-null float64
Volume        504 non-null int32
dtypes: float64(5), int32(1)
memory usage: 25.6 KB
```

[9]:
```
df.describe() # Shows summary statistics based on stock data
```

[9]:

	Open	High	Low	Close	Adj Close	Volume
count	504.000000	504.000000	504.000000	504.000000	504.000000	5.040000e+02
mean	112.341766	113.335853	111.276687	112.322004	107.062920	4.512653e+07
std	10.940972	10.906952	10.880557	10.875574	9.718720	2.035743e+07
min	90.000000	91.669998	89.470001	90.339996	86.980591	1.147590e+07
25%	105.415001	106.294998	104.612501	105.632498	100.535162	3.145228e+07
50%	112.024998	112.915001	110.639999	112.120003	107.452751	4.018980e+07
75%	120.792502	121.472500	119.299999	120.185000	114.287114	5.310688e+07
max	134.460007	134.539993	131.399994	133.000000	125.050819	1.622063e+08

IV. Understand the Data based on Statistics

We will be using "Adj. Closing" price to find the minimum, maximum, average and standard deviation prices. The reason we are using "Adj. Closing" because is mostly use for historical

returns. Also, the Adjusting Prices is change where the stock was accounts for the dividend and splits. However, the "Closing" price was not including with dividend and splits.

[10]:
```
# Use only Adj. Closing
# Find the minimum
df['Adj Close'].min()
```

[10]:
```
86.980591
```

[11]:
```
# Find the maximum
df['Adj Close'].max()
```

[11]:
```
125.05081899999999
```

[12]:
```
# Find the average
df['Adj Close'].mean()
```

[12]:
```
107.06292045039675
```

[13]:
```
# Find the standard deviation
df['Adj Close'].std()
```

[13]:
```
9.718719888205204
```

V. Calculate the Prices

This section, we will be calculating the daily returns, log returns, and other technical indicators such as RSI(Relative Strength Index), MA(Moving Average), SMA(Simple Moving Averga), EMA(Exponential Moving Average), and VWAP(Voume Weighted Average Price). Also, we will calculate drawdowns.

[14]:
```
# Daily Returns
# Formula: (Today Price / Yesterday Price) - 1
df['Daily_Returns'] = df['Adj Close'].shift(1) / df['Adj Close']  - 1
df['Daily_Returns'].head()
```

[14]:
```
Date
2015-01-02         NaN
2015-01-05    0.028988
2015-01-06   -0.000094
2015-01-07   -0.013828
2015-01-08   -0.037001
Name: Daily_Returns, dtype: float64
```

[15]:
```
# Another way of calculating Daily Returns in simple way
DR = df['Adj Close'].pct_change(1) # 1 is for "One Day" in the past
DR.head()
```

[15]:
```
Date
2015-01-02         NaN
2015-01-05   -0.028172
2015-01-06    0.000094
2015-01-07    0.014022
2015-01-08    0.038422
Name: Adj Close, dtype: float64
```

[16]:
```
# Log Returns
# Formula: log(Today Price/ Yesterday Price)
df['Log_Returns'] = np.log(df['Adj Close']) - np.log(df['Adj Close'].shift(1))
```

In this part of this section, we will be using the library of technical analysis. This packages has many different types of technical indicators. However, it does not have every single technical indicators. We do not need to do calculation since the library has done it for us. https://mrjbq7.github.io/ta-lib/doc_index.html

[17]:
```
import talib as ta

# Creating Indicators
n=30 # number of periods

# RSI(Relative Strength Index)
# RSI is technical analysis indicator
# https://www.investopedia.com/terms/r/rsi.asp
df['RSI']=ta.RSI(np.array(df['Adj Close'].shift(1)), timeperiod=n)

# MA(Moving Average)
# https://www.investopedia.com/terms/m/movingaverage.asp
df['MA']=ta.MA(np.array(df['Adj Close'].shift(1)), timeperiod=n, matype=0)

# SMA(Simple Moving Average)
# https://www.investopedia.com/terms/s/sma.asp
df['SMA']=ta.SMA(np.array(df['Adj Close'].shift(1)))

# EMA(Exponential Moving Average)
# https://www.investopedia.com/terms/e/ema.asp
df['EMA']=ta.EMA(np.array(df['Adj Close'].shift(1)), timeperiod=n)
```

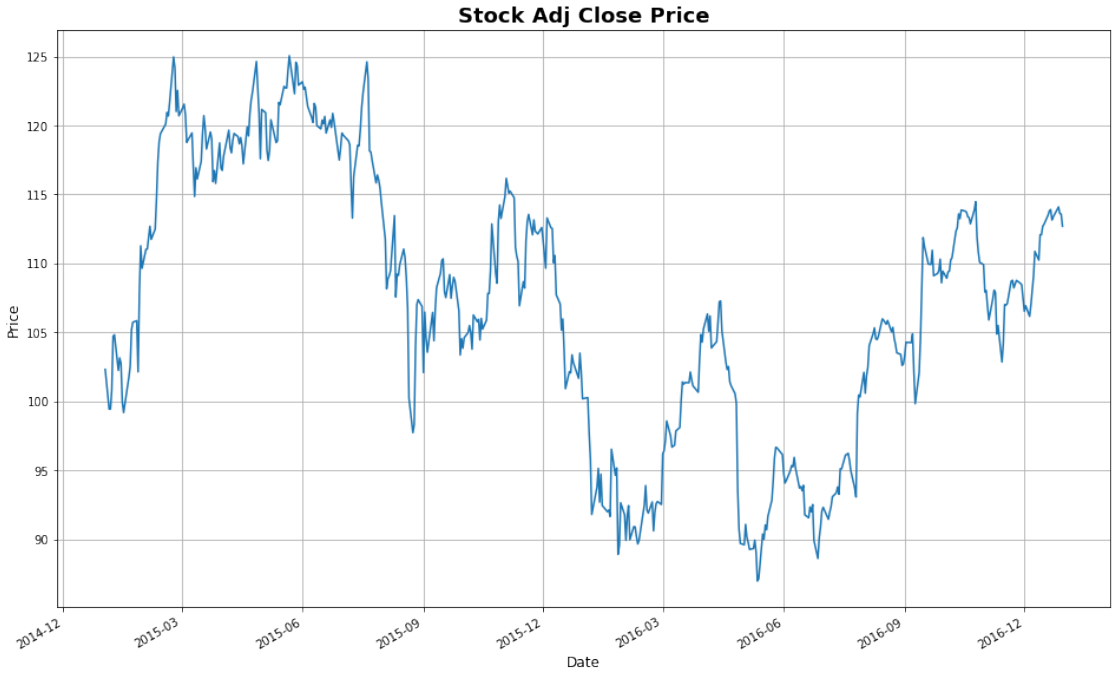
[18]:
```
# Volume Weighted Average Price - VWAP
# http://stockcharts.com/school/doku.php?id=chart_school:technical_indicators:
 ↪vwap_intraday
df['VWAP'] = round(np.cumsum(df['Volume']*(df['High']+df['Low'])/2) / np.
 ↪cumsum(df['Volume']), 2)
df.head()
```

[18]:
```
                 Open        High         Low       Close   Adj Close  \
Date
2015-01-02  111.389999  111.440002  107.349998  109.330002  102.319397
2015-01-05  108.290001  108.650002  105.410004  106.250000   99.436890
2015-01-06  106.540001  107.430000  104.629997  106.260002   99.446251
2015-01-07  107.199997  108.199997  106.699997  107.750000  100.840714
2015-01-08  109.230003  112.150002  108.699997  111.889999  104.715256

               Volume  Daily_Returns  Log_Returns  RSI  MA  SMA  EMA    VWAP
Date
2015-01-02  53204600            NaN          NaN  NaN NaN  NaN  NaN  109.40
2015-01-05  64285500       0.028988    -0.028576  NaN NaN  NaN  NaN  108.10
2015-01-06  65797100      -0.000094     0.000094  NaN NaN  NaN  NaN  107.36
2015-01-07  40105900      -0.013828     0.013925  NaN NaN  NaN  NaN  107.37
2015-01-08  59364500      -0.037001     0.037703  NaN NaN  NaN  NaN  108.01
```

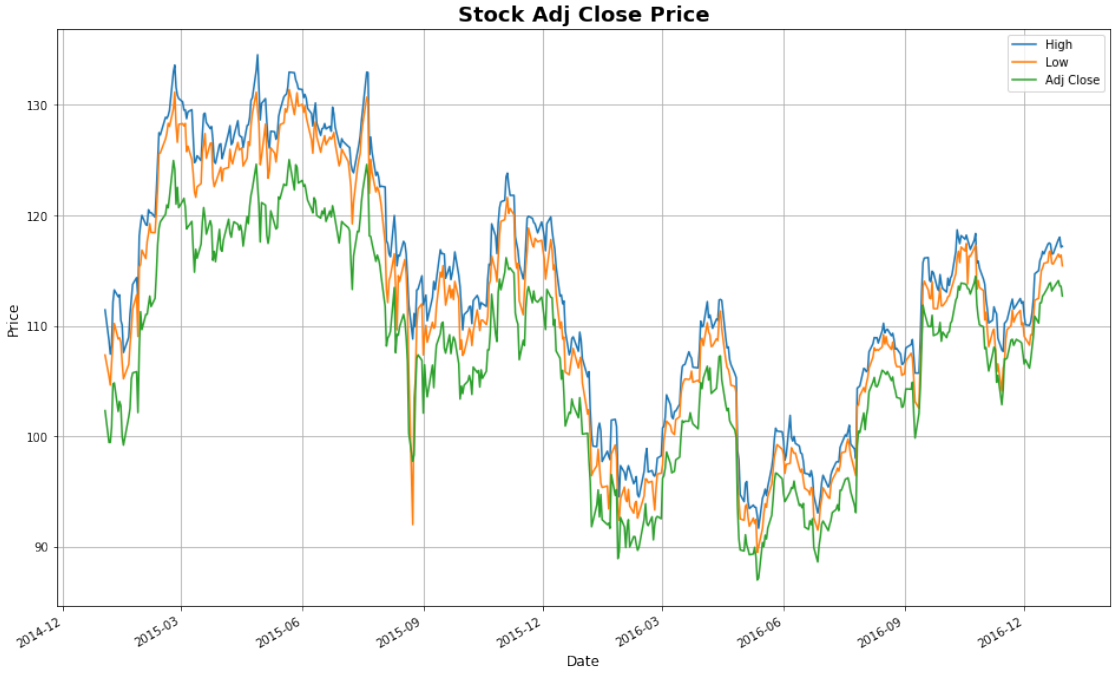
[19]:
```
# Drawdown
# Drawdown shows the decline price since the stock began trading
# https://www.investopedia.com/terms/d/drawdown.asp
# There are 252 trading day in a year
window = 252

# Calculate the maximum drawdown
# Use the min_period of 1 (1 is the least valid observations) for the first 252
 ↪day in the data
Maximum_Drawdown = df['Adj Close'].rolling(window, min_periods=1).max()
Daily_Drawdown = df['Adj Close']/Maximum_Drawdown - 1.0

# Calculate the negative drawdown
Negative_Drawdown = Daily_Drawdown.rolling(window, min_periods=1).min()
```

VI. Plot Charts

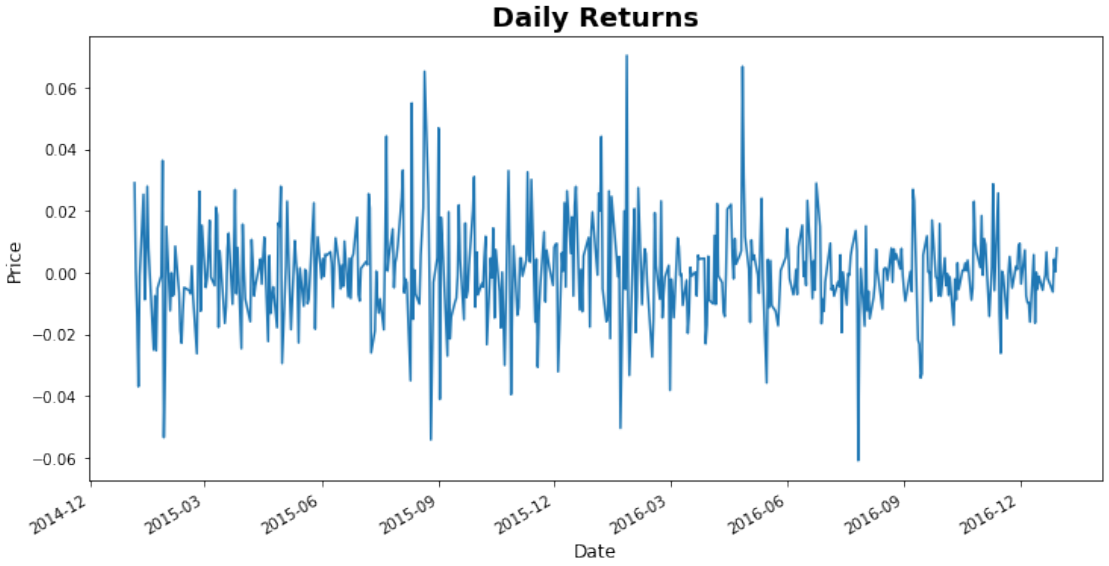
[20]:
```
# Plot Simple Line Chart
# Plot Adj Close

plt.figure(figsize=(16,10))
df['Adj Close'].plot(grid=True)
plt.title("Stock Adj Close Price", fontsize=18, fontweight='bold')
plt.xlabel("Date", fontsize=12)
plt.ylabel("Price",fontsize=12)
plt.show()
```



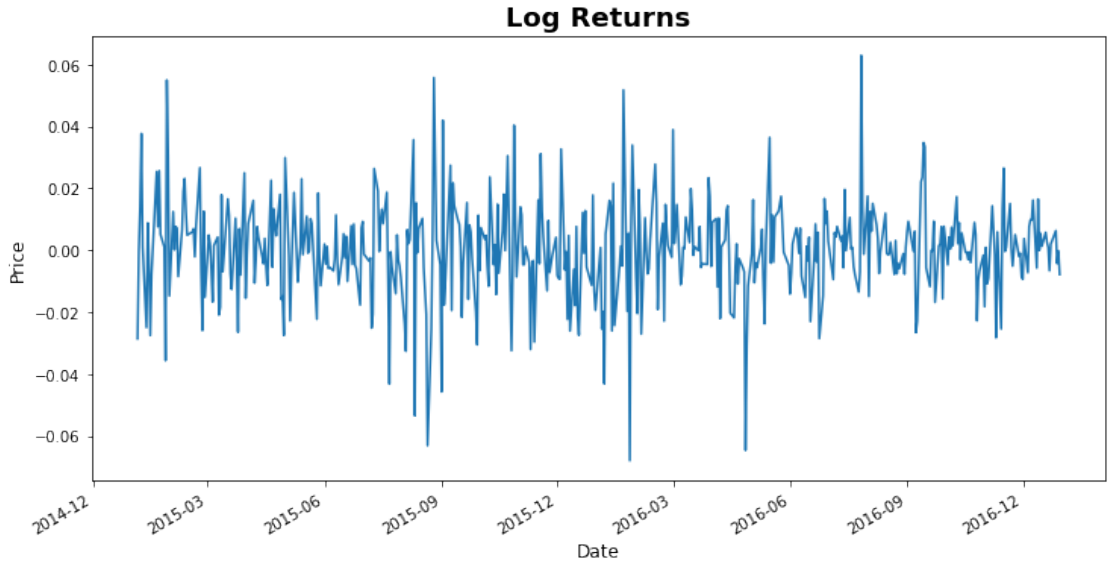
[21]:
```
# Plot High, Low, Adj Close
df[['High', 'Low', 'Adj Close']].plot(figsize=(16,10), grid=True)
plt.title("Stock Adj Close Price", fontsize=18, fontweight='bold')
plt.xlabel("Date", fontsize=12)
plt.ylabel("Price", fontsize=12)
plt.show()
```



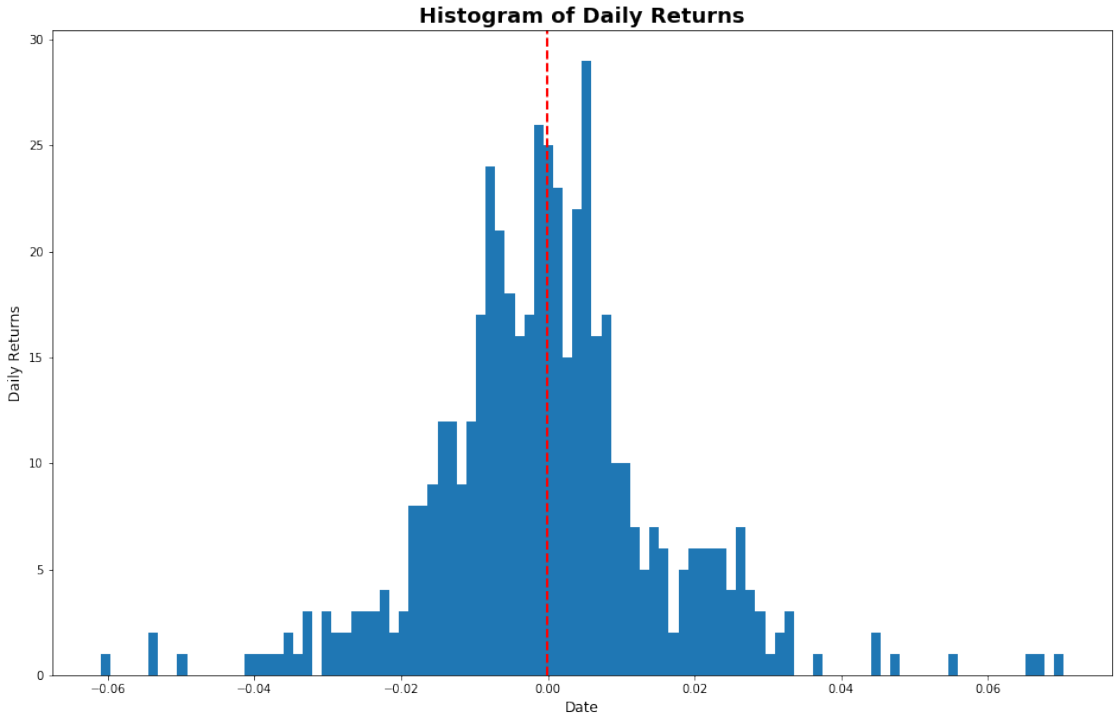
[22]:
```
# Plot Daily Returns
df['Daily_Returns'].plot(figsize=(12,6))
plt.title("Daily Returns",fontsize=18, fontweight='bold')
plt.xlabel("Date", fontsize=12)
plt.ylabel("Price", fontsize=12)
plt.show()
```



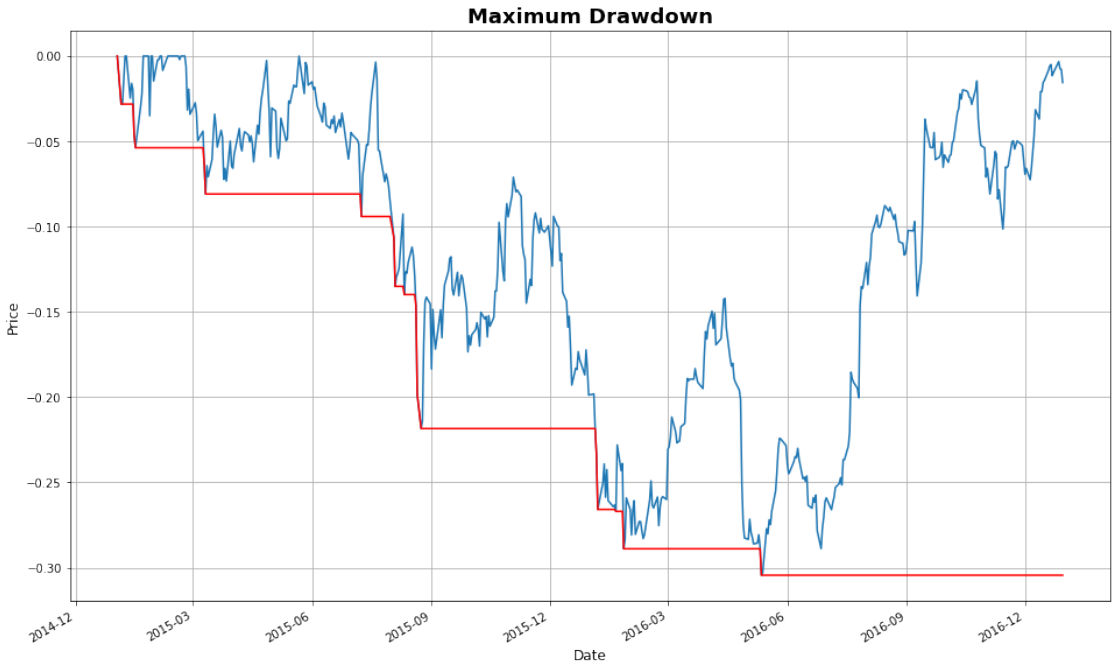
[23]:
```
# Plot Log Returns
df['Log_Returns'].plot(figsize=(12,6))
plt.title("Log Returns", fontsize=18, fontweight='bold')
plt.xlabel("Date", fontsize=12)
plt.ylabel("Price", fontsize=12)
plt.show()
```



[24]:
```
# Histogram of Daily Returns
# Histogram is distribution of numerical data and has a rectangle whose area is␣
 ↪prportional to the frequency of a variable.
plt.figure(figsize=(16,10))
plt.hist(df['Daily_Returns'].dropna(), bins=100, label='Daily Returns data') #␣
 ↪Drop NaN
plt.title("Histogram of Daily Returns", fontsize=18, fontweight='bold')
plt.axvline(df['Daily_Returns'].mean(), color='r', linestyle='dashed',␣
 ↪linewidth=2) # Shows the average line
plt.xlabel("Date", fontsize=12)
plt.ylabel("Daily Returns", fontsize=12)
plt.show()
```



[25]:
```
# Plot Drawdown
plt.figure(figsize=(16,10))
Daily_Drawdown.plot()
Negative_Drawdown.plot(color='r',grid=True)
plt.title("Maximum Drawdown", fontsize=18, fontweight='bold')
plt.xlabel("Date", fontsize=12)
plt.ylabel("Price", fontsize=12)
plt.show()
```



VII. Holding Period Return(HPR)

Holding period return (HPR) is the rate of return on an individual stocks or portfolio over the whole period during the time it was held and it a measurement of investment performance.

[26]:
```
# https://www.investopedia.com/exam-guide/series-65/quantitative-methods/
 ↪holding-period-return.asp
# Formula: (Ending Value of Investment + Dividend - Beginning Value of␣
 ↪Investment) / Beginning Value of Investment
# To get dividend in Yahoo!
DIV = yf.download(stock, start, end, actions=True)['Dividends']
```

```
[**********************100%**********************]  1 of 1 downloaded
```

[27]:
```
# See how much dividends and splits was given during the time period
DIV
```

[27]:
```
Date
2015-01-02    0.00
2015-01-05    0.00
2015-01-06    0.00
2015-01-07    0.00
2015-01-08    0.00
2015-01-09    0.00
2015-01-12    0.00
2015-01-13    0.00
```

```
2015-01-14    0.00
2015-01-15    0.00
2015-01-16    0.00
2015-01-20    0.00
2015-01-21    0.00
2015-01-22    0.00
2015-01-23    0.00
2015-01-26    0.00
2015-01-27    0.00
2015-01-28    0.00
2015-01-29    0.00
2015-01-30    0.00
2015-02-02    0.00
2015-02-03    0.00
2015-02-04    0.00
2015-02-05    0.47
2015-02-06    0.00
2015-02-09    0.00
2015-02-10    0.00
2015-02-11    0.00
2015-02-12    0.00
2015-02-13    0.00
              …
2016-11-17    0.00
2016-11-18    0.00
2016-11-21    0.00
2016-11-22    0.00
2016-11-23    0.00
2016-11-25    0.00
2016-11-28    0.00
2016-11-29    0.00
2016-11-30    0.00
2016-12-01    0.00
2016-12-02    0.00
2016-12-05    0.00
2016-12-06    0.00
2016-12-07    0.00
2016-12-08    0.00
2016-12-09    0.00
2016-12-12    0.00
2016-12-13    0.00
2016-12-14    0.00
2016-12-15    0.00
2016-12-16    0.00
2016-12-19    0.00
2016-12-20    0.00
2016-12-21    0.00
```

```
2016-12-22    0.00
2016-12-23    0.00
2016-12-27    0.00
2016-12-28    0.00
2016-12-29    0.00
2016-12-30    0.00
Name: Dividends, Length: 504, dtype: float64
```

[28]:
```
# Add all the dividend
Total_Dividend = DIV.sum()
Total_Dividend
```

[28]:
```
4.26
```

[29]:
```
# You invest beginning 2015 and sold it end of 2017
HPR = (df['Adj Close'][502] + Total_Dividend - df['Adj Close'][0]) / df['Adj␣
 ↪Close'][0]
HPR
```

[29]:
```
0.15167065536947996
```

[30]:
```
# You can use round for 4 decimal points
print('Holding Period Return: ', str(round(HPR,4)*100)+"%")
```

```
Holding Period Return:  15.17%
```

We going to pick another stocks that is Microsoft and we will compare it to Apple.

[31]:
```
MSFT =  yf.download('MSFT', start, end)['Adj Close'] # Use Adj Close only
MSFT_DIV = yf.download('MSFT', start, end, actions=True)['Dividends']
```

```
[**********************100%**********************]  1 of 1 downloaded
[**********************100%**********************]  1 of 1 downloaded
```

[32]:
```
MSFT.head() # Shows only Date and Adj Close
```

[32]:
```
Date
2015-01-02    42.783936
2015-01-05    42.390499
2015-01-06    41.768314
2015-01-07    42.299000
2015-01-08    43.543350
Name: Adj Close, dtype: float64
```

[33]:
```
MSFT_DIV # Shows how much dividend was given
```

[33]:
```
Date
2015-01-02    0.0
```

```
2015-01-05    0.0
2015-01-06    0.0
2015-01-07    0.0
2015-01-08    0.0
2015-01-09    0.0
2015-01-12    0.0
2015-01-13    0.0
2015-01-14    0.0
2015-01-15    0.0
2015-01-16    0.0
2015-01-20    0.0
2015-01-21    0.0
2015-01-22    0.0
2015-01-23    0.0
2015-01-26    0.0
2015-01-27    0.0
2015-01-28    0.0
2015-01-29    0.0
2015-01-30    0.0
2015-02-02    0.0
2015-02-03    0.0
2015-02-04    0.0
2015-02-05    0.0
2015-02-06    0.0
2015-02-09    0.0
2015-02-10    0.0
2015-02-11    0.0
2015-02-12    0.0
2015-02-13    0.0
              …
2016-11-17    0.0
2016-11-18    0.0
2016-11-21    0.0
2016-11-22    0.0
2016-11-23    0.0
2016-11-25    0.0
2016-11-28    0.0
2016-11-29    0.0
2016-11-30    0.0
2016-12-01    0.0
2016-12-02    0.0
2016-12-05    0.0
2016-12-06    0.0
2016-12-07    0.0
2016-12-08    0.0
2016-12-09    0.0
2016-12-12    0.0
```

```
2016-12-13    0.0
2016-12-14    0.0
2016-12-15    0.0
2016-12-16    0.0
2016-12-19    0.0
2016-12-20    0.0
2016-12-21    0.0
2016-12-22    0.0
2016-12-23    0.0
2016-12-27    0.0
2016-12-28    0.0
2016-12-29    0.0
2016-12-30    0.0
Name: Dividends, Length: 504, dtype: float64
```

[34]:
```
MSFT_Dividend = MSFT_DIV.sum()
MSFT_Dividend
```

[34]: 2.76

[35]:
```
# You invest beginning 2015 and sold it end of 2017
MSFT_HPR = (MSFT[502] + Total_Dividend - MSFT[0]) / MSFT[0]
MSFT_HPR
```

[35]: 0.5192736124137809

[36]:
```
# You can use round for 4 decimal points
print('Apple Holding Period Return: ', str(round(HPR,4)*100)+"%")
print('Microsoft Holding Period Return: ', str(round(MSFT_HPR,4)*100)+"%")
```

```
Apple Holding Period Return:  15.17%
Microsoft Holding Period Return:  51.93%
```

In the conclusion, we use 2 stocks to compare holding period return. Therefore, Microsoft had higher holding period return than Apple. Therefore, I would invest in Microsoft based on the stock analysis. However, if you comparing 2 stocks or 2 portfolio. You would pick the ones with the highest rate of return.