In [ ]:

Stock Market Analysis Project

Please Note: You are free to treat this as a full exercise, or just view the solutions video as a code along project. This project is meant to be pretty challenging as it will introduce a few new concepts through some hints!

Welcome to your first capstone project! This project is meant to cap off the first half of the course, which mainly dealt with learning the libraries that we use in this course, the second half of the course will deal a lot more with quantitative trading techniques and platforms.

We'll be analyzing stock data related to a few car companies, from Jan 1 2012 to Jan 1 2017. Keep in mind that this project is mainly just to practice your skills with matplotlib, pandas, and numpy. Don't infer financial trading advice from the analysis we do here!

Part 0: Import

Import the various libraries you will need-you can always just come back up here or import as you go along :)

In [24]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

Part 1: Getting the Data

Tesla Stock (Ticker: TSLA on the NASDAQ)

***Note! Not everyone will be working on a computer that will give them open access to download the stock information using pandas_datareader (firewalls, admin permissions, etc...). Because of this, the csv file for the Tesla is provided in a data folder inside this folder. It is called Tesla_Stock.csv. Feel free to just use this with read_csv!**

Use pandas_datareader to obtain the historical stock information for Tesla from Jan 1, 2012 to Jan 1, 2017.

In [5]:
```
import yfinance as yf
```

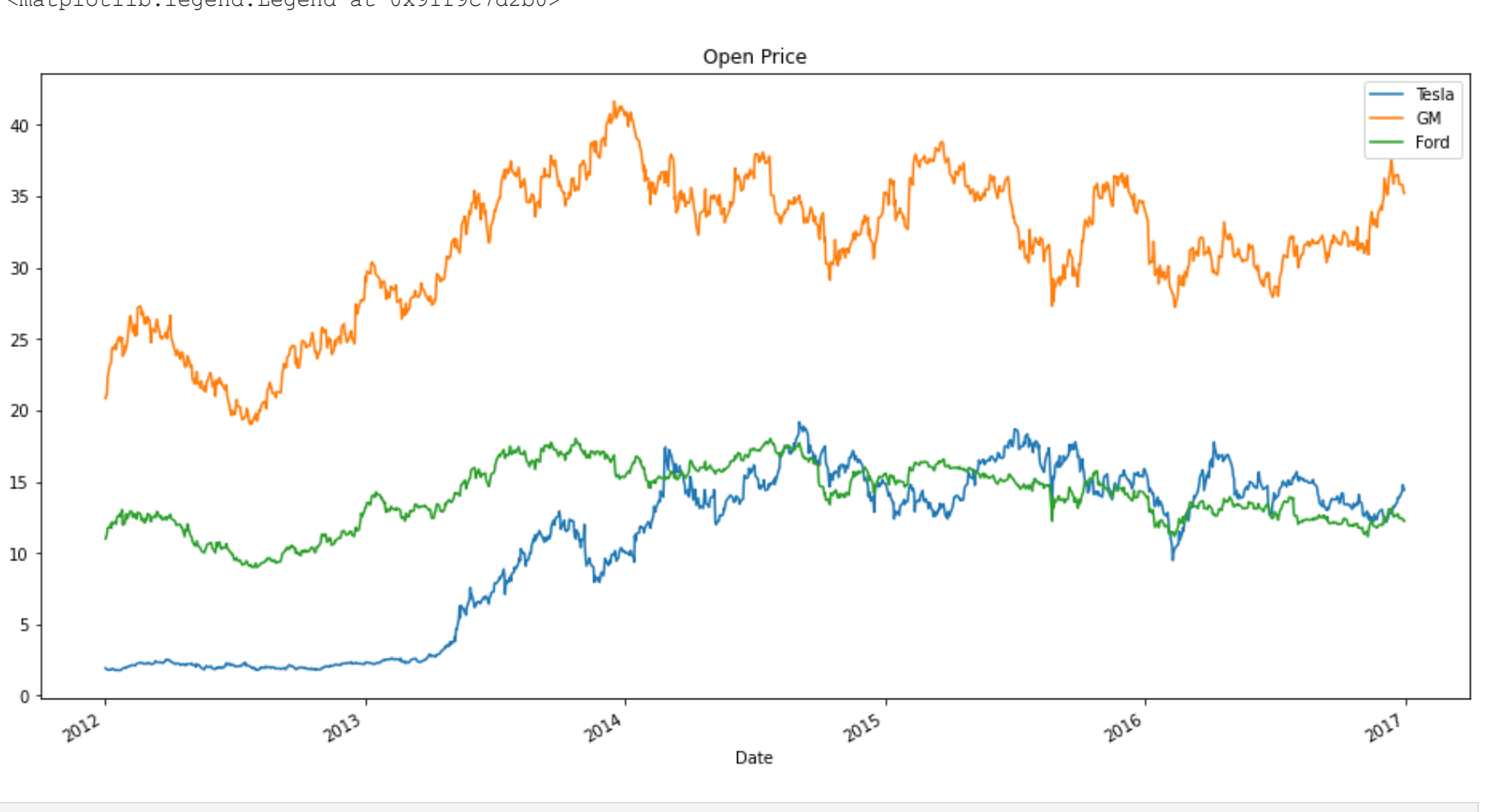
In [6]:
```
tesla = yf.download("TSLA", start="2012-01-01", end="2017-01-01")
```
```
[*********************100%***********************]  1 of 1 completed
```

In [7]:
```
ford = yf.download("F", start="2012-01-01", end="2017-01-01")
```
```
[*********************100%***********************]  1 of 1 completed
```

In [8]:
```
gm = yf.download("GM", start="2012-01-01", end="2017-01-01")
```
```
[*********************100%***********************]  1 of 1 completed
```

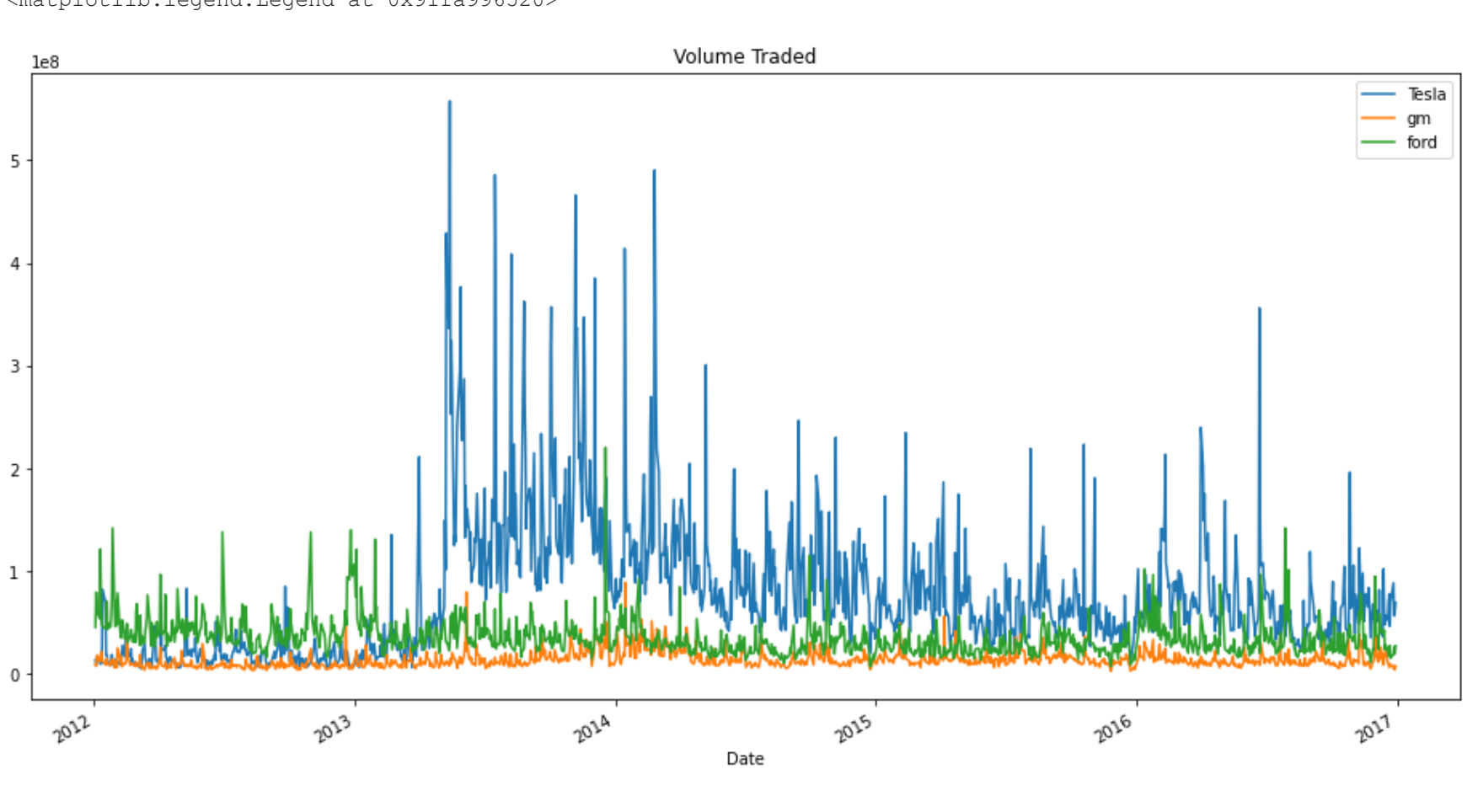
In [9]:
```
tesla['Open'].plot(label='Tesla',figsize=(16,8),title='Open Price')
gm['Open'].plot(label='GM')
ford['Open'].plot(label='Ford')
plt.legend()
```

Out[9]: <matplotlib.legend.Legend at 0x91f9c7d2b0>



In [10]:
```
tesla['Volume'].plot(label='Tesla',figsize=(16,8),title='Volume Traded')
gm['Volume'].plot(label='gm')
ford['Volume'].plot(label='ford')
plt.legend()
```

Out[10]: <matplotlib.legend.Legend at 0x91fa996520>



Other Car Companies

Repeat the same steps to grab data for Ford and GM (General Motors),

In [ ]:

In [11]:
```
ford
```

Out[11]:

Date	Open	High	Low	Close	Adj Close	Volume
2012-01-03	11.00	11.25	10.99	11.13	7.345914	45709900
2012-01-04	11.15	11.53	11.07	11.30	7.458119	79725200
2012-01-05	11.33	11.63	11.24	11.59	7.649521	67877500
2012-01-06	11.74	11.80	11.52	11.71	7.728722	59840700
2012-01-09	11.83	11.95	11.70	11.80	7.788122	53981500
...	...	...	...	...	...	...
2016-12-23	12.43	12.46	12.36	12.46	9.882610	15621200
2016-12-27	12.43	12.51	12.36	12.39	9.827092	19467400
2016-12-28	12.37	12.45	12.22	12.25	9.716052	26875400
2016-12-29	12.25	12.31	12.22	12.23	9.700186	19819100
2016-12-30	12.24	12.28	12.08	12.13	9.620872	27405700

1258 rows × 6 columns

In [12]:
```
ford['Volume'].argmax()
```

Out[12]: 493

In [ ]:

In [ ]:

In [ ]:

Part 2: Visualizing the Data

Time to visualize the data.

Follow along and recreate the plots below according to the instructions and explanations.

Recreate this linear plot of all the stocks' Open price ! Hint: For the legend, use label parameter and plt.legend()

In [13]:
```
# Code Here
```

In [ ]:

Plot the Volume of stock traded each day.

In [ ]:

Interesting, looks like Ford had a really big spike somewhere in late 2013. What was the date of this maximum trading volume for Ford?

Bonus: What happened that day?

In [ ]:

In [ ]:

The Open Price Time Series Visualization makes Tesla look like its always been much more valuable as a company than GM and Ford. But to really understand this we would need to look at the total market cap of the company, not just the stock price. Unfortunately our current data doesn't have that information of total units of stock present. But what we can do as a simple calcualtion to try to represent total money traded would be to multply the Volume column by the Open price. Remember that this still isn't the actual Market Cap, its just a visual representation of the total amount of money being traded around using the time series. (e.g. 100 units of stock at $10 each versus 100000 units of stock at $1 each)

Create a new column for each dataframe called "Total Traded" which is the Open Price multiplied by the Volume Traded.

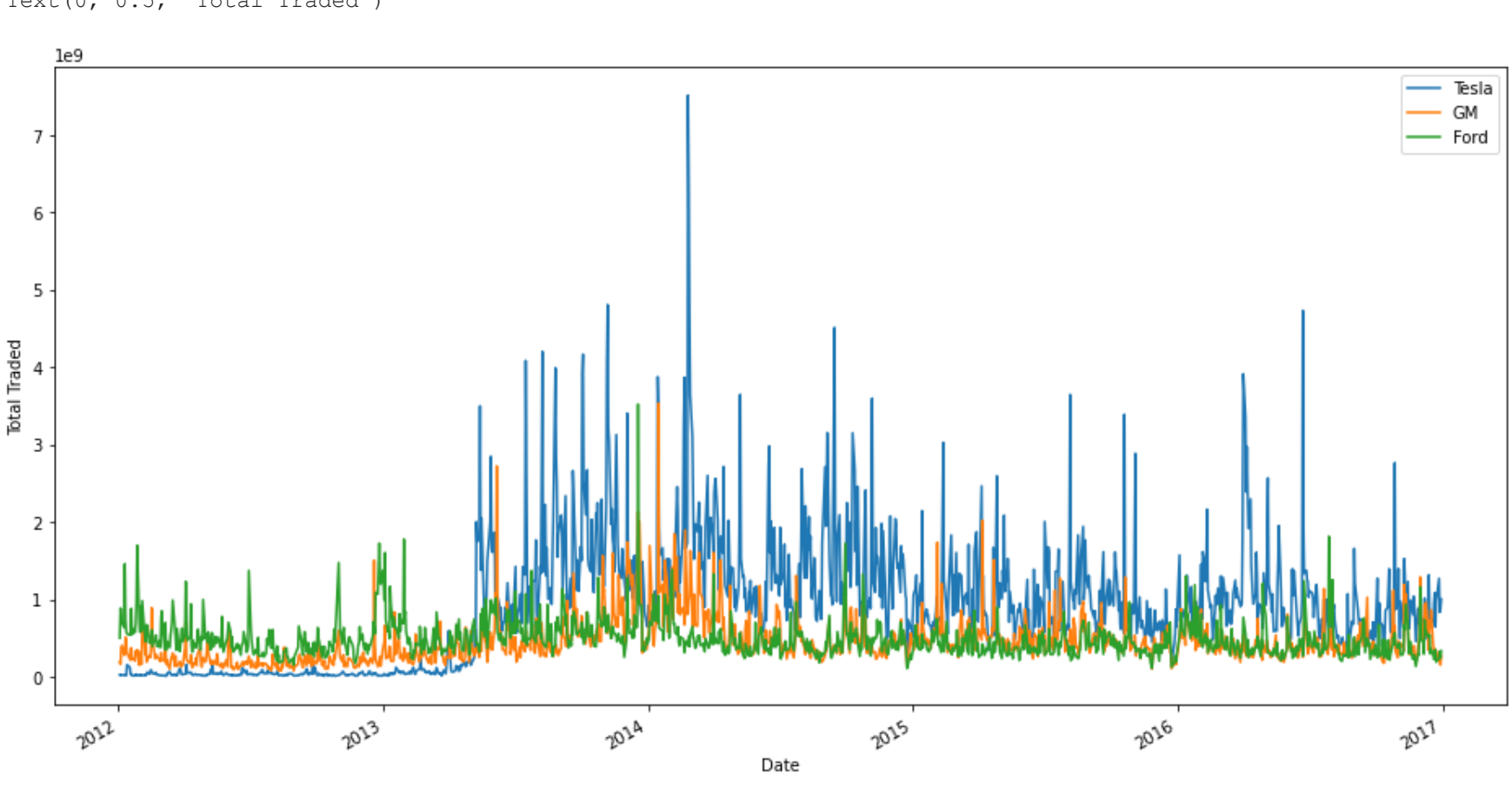
In [14]:
```
tesla['Total Traded'] = tesla['Open']*tesla['Volume']
ford['Total Traded'] = ford['Open']*ford['Volume']
gm['Total Traded'] = gm['Open']*gm['Volume']
```

In [ ]:

Plot this "Total Traded" against the time index.

In [15]:
```
tesla['Total Traded'].plot(label='Tesla',figsize=(16,8))
gm['Total Traded'].plot(label='GM')
ford['Total Traded'].plot(label='Ford')
plt.legend()
plt.ylabel('Total Traded')
```

Out[15]: Text(0, 0.5, 'Total Traded')



In [16]:
```
tesla['Total Traded'].argmax()
```

Out[16]: 538

Interesting, looks like there was huge amount of money traded for Tesla somewhere in early 2014. What date was that and what happened?

In [ ]:

In [ ]:

Let's practice plotting out some MA (Moving Averages). Plot out the MA50 and MA200 for GM.

In [17]:
```
# Code here
```

In [ ]:

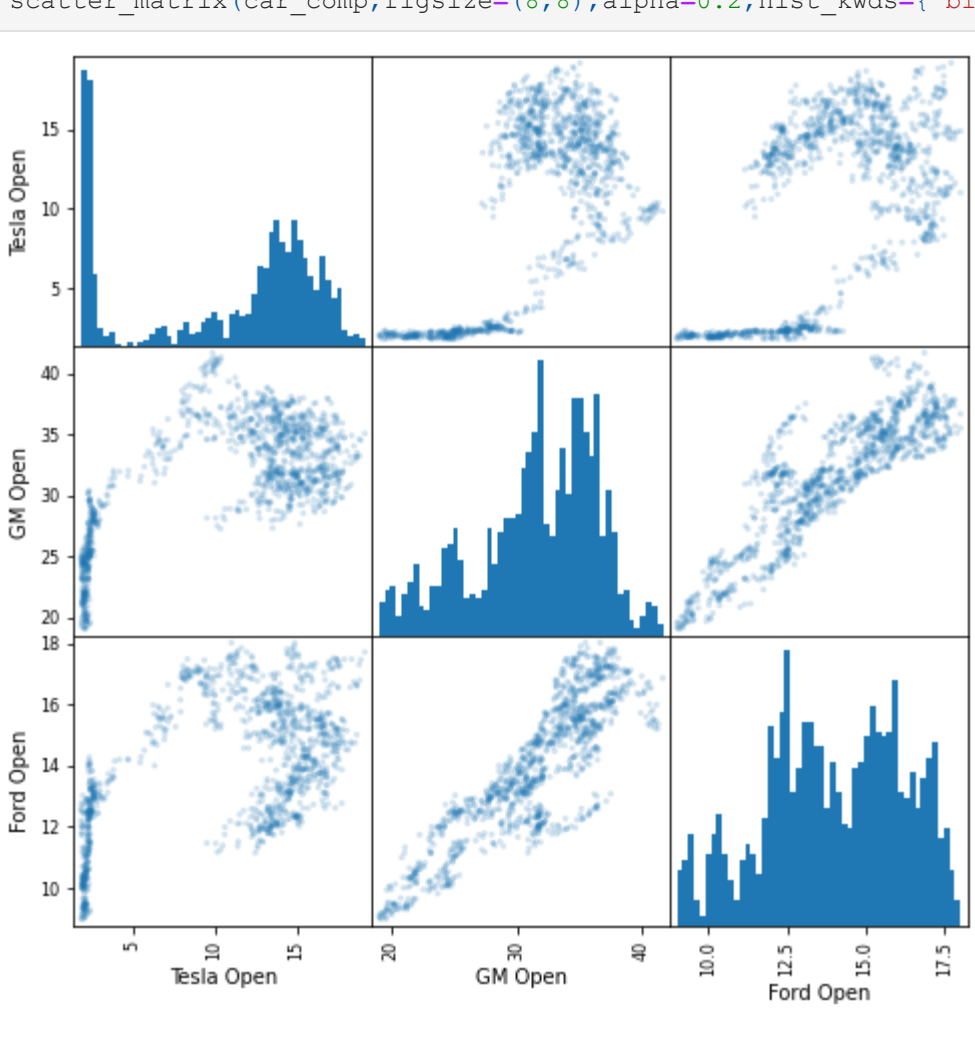
Finally lets see if there is a relationship between these stocks, after all, they are all related to the car industry. We can see this easily through a scatter matrix plot. Import scatter_matrix from pandas.plotting and use it to create a scatter matrix plot of all the stocks'opening price. You may need to rearrange the columns into a new single dataframe. Hints and info can be found here: https://pandas.pydata.org/pandas-docs/stable/visualization.html#scatter-matrix-plot

In [18]:
```
from pandas.plotting import scatter_matrix
```

In [19]:
```
car_comp = pd.concat([tesla['Open'],gm['Open'],ford['Open']],axis=1)
```

In [20]:
```
car_comp.columns = ['Tesla Open','GM Open','Ford Open']
```

In [22]:
```
# You can use a semi-colon to remove the axes print outs
scatter_matrix(car_comp,figsize=(8,8),alpha=0.2,hist_kwds={'bins':50});
```



Bonus Visualization Task! (Note: This is hard!)

Let's now create a candlestick chart! Watch the video if you get stuck on trying to recreate this visualization, there are quite a few steps involved!Refer to the video to understand how to interpret and read this chart. Hints: https://matplotlib.org/examples/pylab_examples/finance_demo.html

Create a CandleStick chart for Ford in January 2012 (too many dates won't look good for a candlestick chart)

In [25]:

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Input In [25], in <cell line: 1>()
----> 1 from matplotlib.finance import candlestick_ohlc
      2 from matplotlib.dates import DateFormatter, date2num, WeekdayLocator, DayLocator, MONDAY
      4 # Rest the index to get a column of January Dates

ModuleNotFoundError: No module named 'matplotlib.finance'
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Part 3: Basic Financial Analysis

Now it is time to focus on a few key financial calculations. This will serve as your transition to the second half of the course. All you need to do is follow along with the instructions, this will mainly be an exercise in converting a mathematical equation or concept into code using python and pandas, something we will do often when working with quantiative data! If you feel very lost in this section, don't worry! Just go to the solutions lecture and treat it as a code-along lecture, use whatever style of learning works best for you!

Let's begin!

Daily Percentage Change

First we will begin by calculating the daily percentage change. Daily percentage change is defined by the following formula:

This defines r_t (return at time t) as equal to the price at time t divided by the price at time t-1 (the previous day) minus 1. Basically this just informs you of your percent gain (or loss) if you bought the stock on day and then sold it the next day. While this isn't necessarily helpful for attempting to predict future values of the stock, its very helpful in analyzing the volatility of the stock. If daily returns have a wide distribution, the stock is more volatile from one day to the next. Let's calculate the percent returns and then plot them with a histogram, and decide which stock is the most stable!

Create a new column for each dataframe called returns. This column will be calculated from the Close price column. There are two ways to do this, either a simple calculation using the .shift() method that follows the formula above, or you can also use pandas' built in pct_change method.

In [29]:
```
# Method 1: Using shift
tesla['returns'] = (tesla['Close'] / tesla['Close'].shift(1) ) - 1
tesla.head()
```

Out[29]:

Date	Open	High	Low	Close	Adj Close	Volume	Total Traded	returns
2012-01-03	1.929333	1.966667	1.843333	1.872000	1.872000	13921500	2.685921e+07	NaN
2012-01-04	1.880667	1.911333	1.833333	1.847333	1.847333	9451500	1.777512e+07	-0.013177
2012-01-05	1.850667	1.862000	1.790000	1.808000	1.808000	15082500	2.791269e+07	-0.021292
2012-01-06	1.813333	1.852667	1.760667	1.794000	1.794000	14794500	2.682736e+07	-0.007743
2012-01-09	1.800000	1.832667	1.741333	1.816667	1.816667	13455000	2.421900e+07	0.012635

In [27]:
```
# Method 2: Using pct_change
tesla['returns'] = tesla['Close'].pct_change(1)
```

In [28]:
```
tesla.head()
```

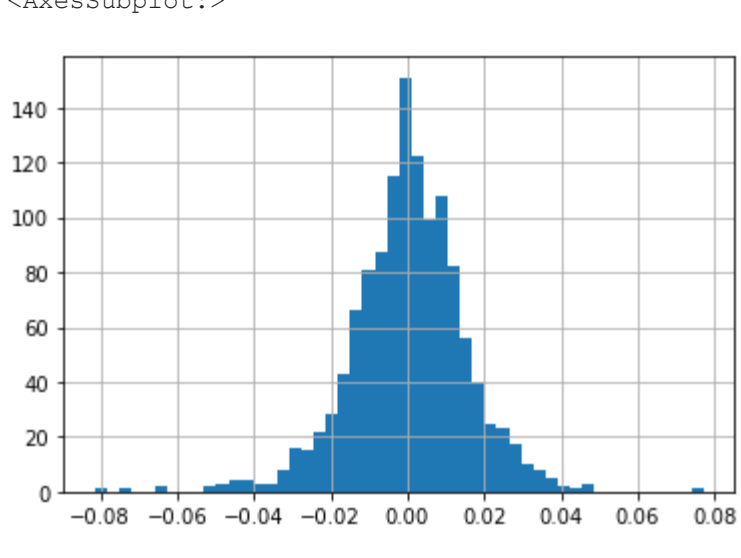
Out[28]:

Date	Open	High	Low	Close	Adj Close	Volume	Total Traded	returns
2012-01-03	1.929333	1.966667	1.843333	1.872000	1.872000	13921500	2.685921e+07	NaN
2012-01-04	1.880667	1.911333	1.833333	1.847333	1.847333	9451500	1.777512e+07	-0.013177
2012-01-05	1.850667	1.862000	1.790000	1.808000	1.808000	15082500	2.791269e+07	-0.021292
2012-01-06	1.813333	1.852667	1.760667	1.794000	1.794000	14794500	2.682736e+07	-0.007743
2012-01-09	1.800000	1.832667	1.741333	1.816667	1.816667	13455000	2.421900e+07	0.012635

In [30]:
```
# Now repeat for the other dataframes
ford['returns'] = ford['Close'].pct_change(1)
gm['returns'] = gm['Close'].pct_change(1)
```

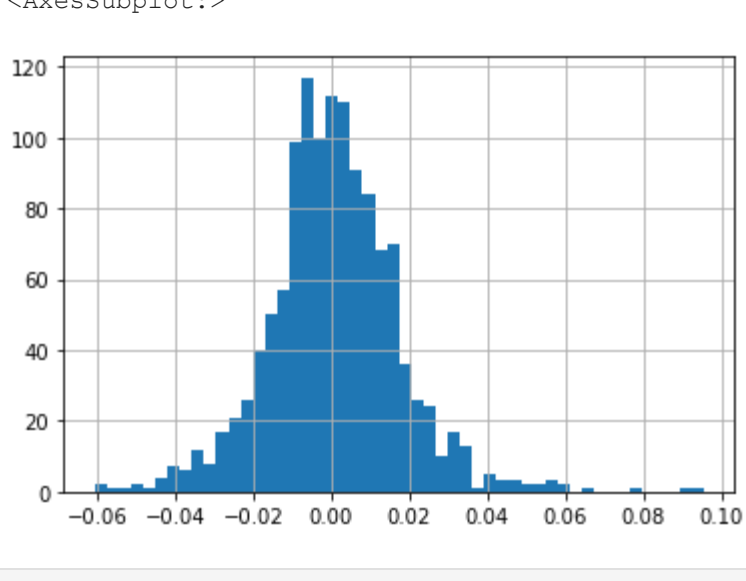
In [31]:
```
ford['returns'].hist(bins=50)
```

Out[31]: <AxesSubplot:>



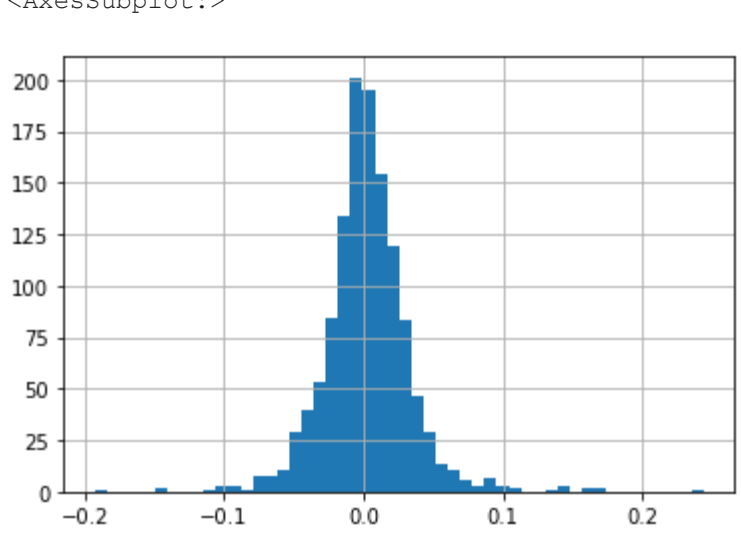
In [32]:
```
gm['returns'].hist(bins=50)
```

Out[32]: <AxesSubplot:>



In [33]:
```
tesla['returns'].hist(bins=50)
```

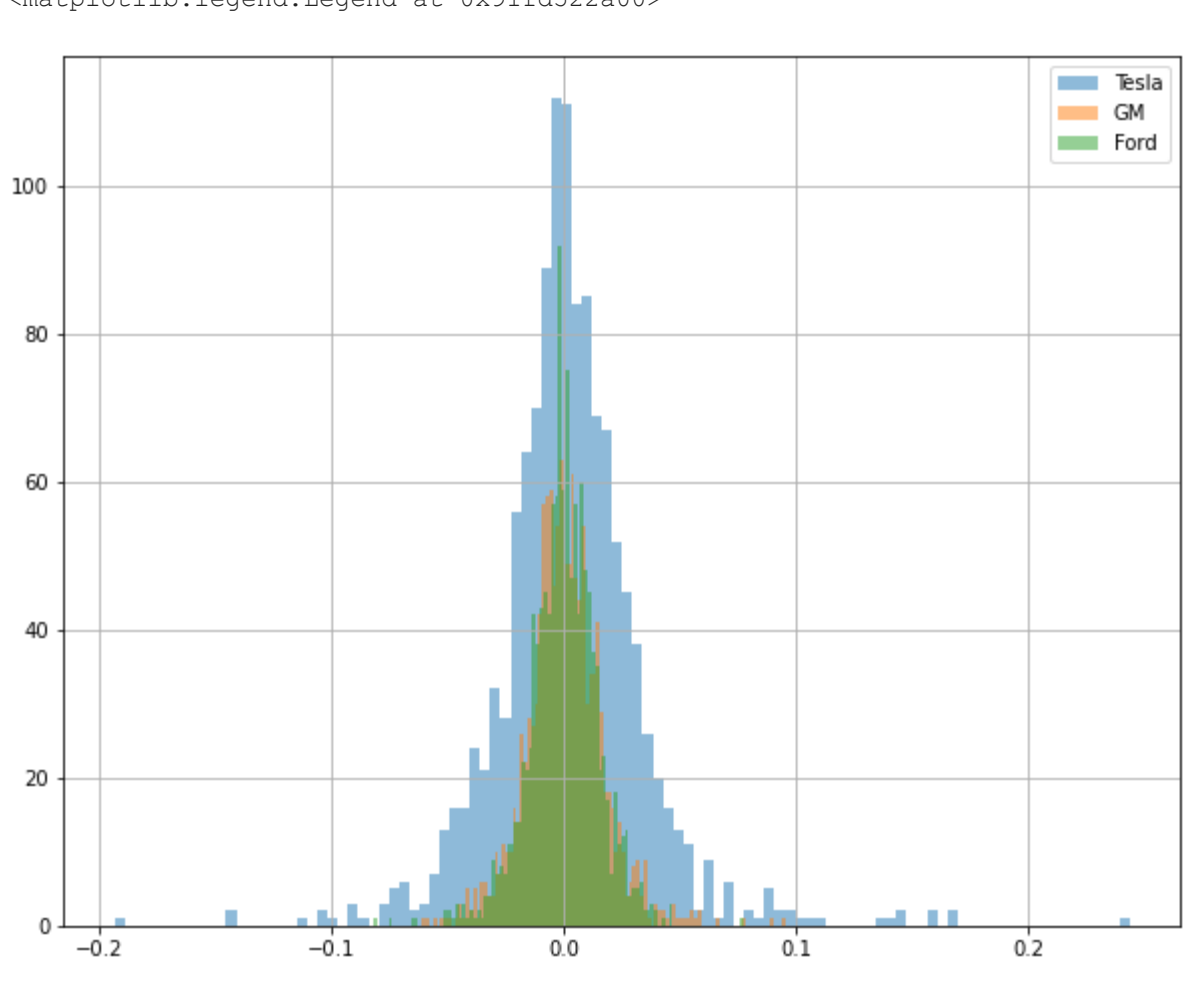
Out[33]: <AxesSubplot:>



Now plot a histogram of each companies returns. Either do them separately, or stack them on top of each other. Which stock is the most "volatile"? (as judged by the variance in the daily returns we will discuss volatility in a lot more detail in future lectures.)

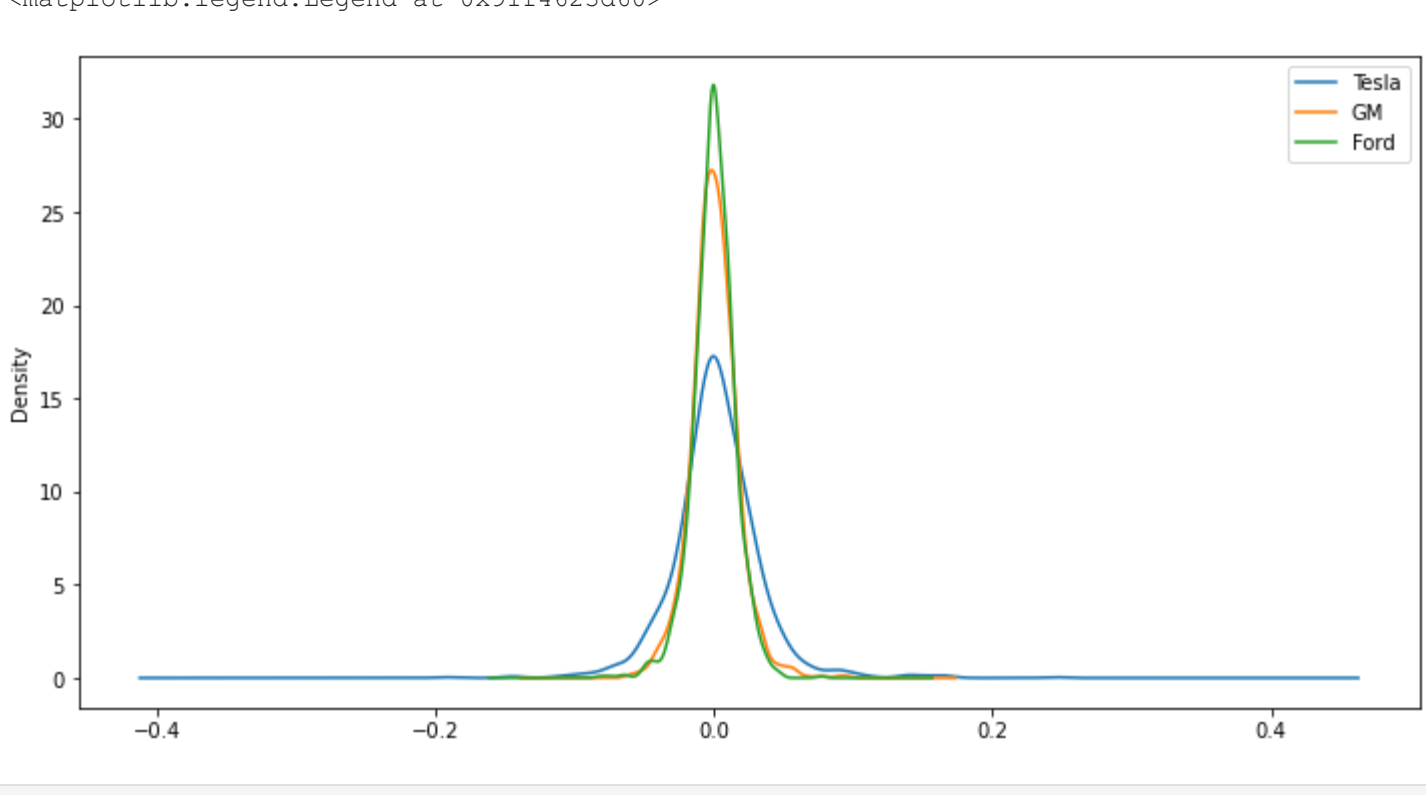
In [36]:
```
tesla['returns'].hist(bins=100,label='Tesla',figsize=(10,8),alpha=0.5)
gm['returns'].hist(bins=100,label='GM',alpha=0.5)
ford['returns'].hist(bins=100,label='Ford',alpha=0.5)
plt.legend()
```

Out[36]: <matplotlib.legend.Legend at 0x91fd522a00>



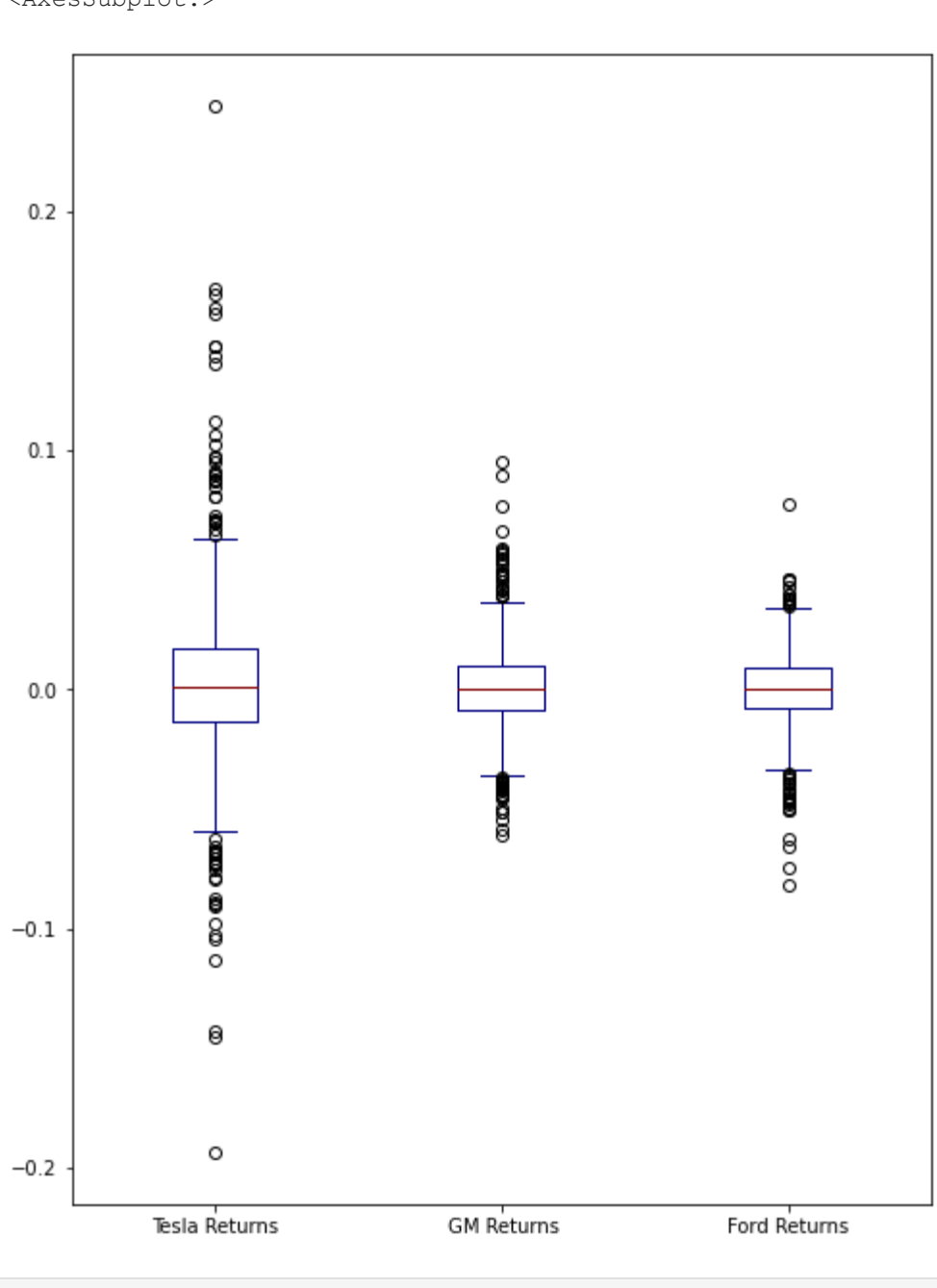
In [35]:
```
tesla['returns'].plot(kind='kde',label='Tesla',figsize=(12,6))
gm['returns'].plot(kind='kde',label='GM')
ford['returns'].plot(kind='kde',label='Ford')
plt.legend()
```

Out[35]: <matplotlib.legend.Legend at 0x91f4623d60>



In [37]:
```
box_df = pd.concat([tesla['returns'],gm['returns'],ford['returns']],axis=1)
box_df.columns = ['Tesla Returns',' GM Returns','Ford Returns']
box_df.plot(kind='box',figsize=(8,11),colormap='jet')
```

Out[37]: <AxesSubplot:>



In [ ]:

Try also plotting a KDE instead of histograms for another view point. Which stock has the widest plot?

In [ ]:

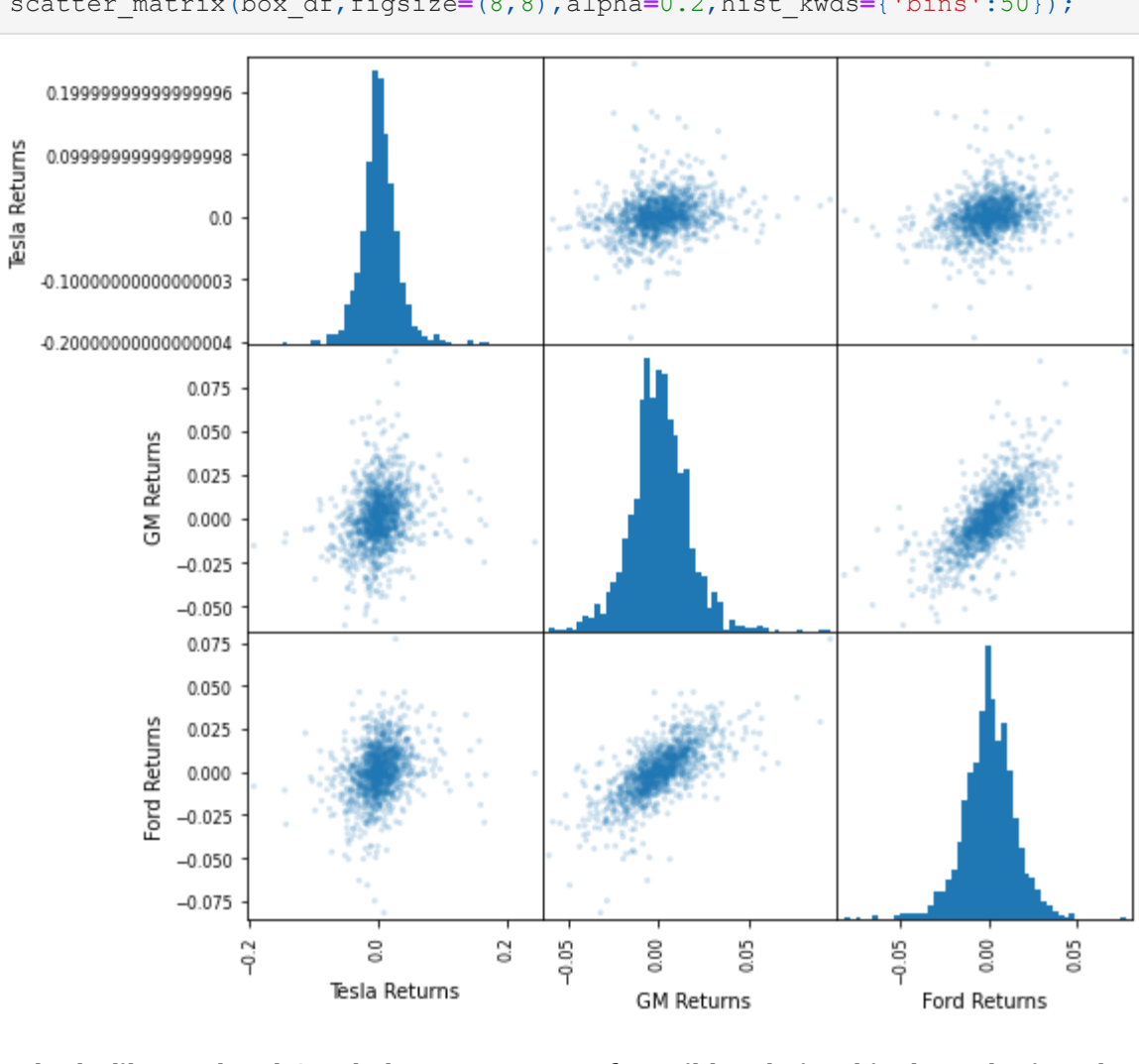
Try also creating some box plots comparing the returns.

In [ ]:

Comparing Daily Returns between Stocks

Create a scatter matrix plot to see the correlation between each of the stocks daily returns. This helps answer the questions of how related the car companies are. Is Tesla begin treated more as a technology company rather than a car company by the market?

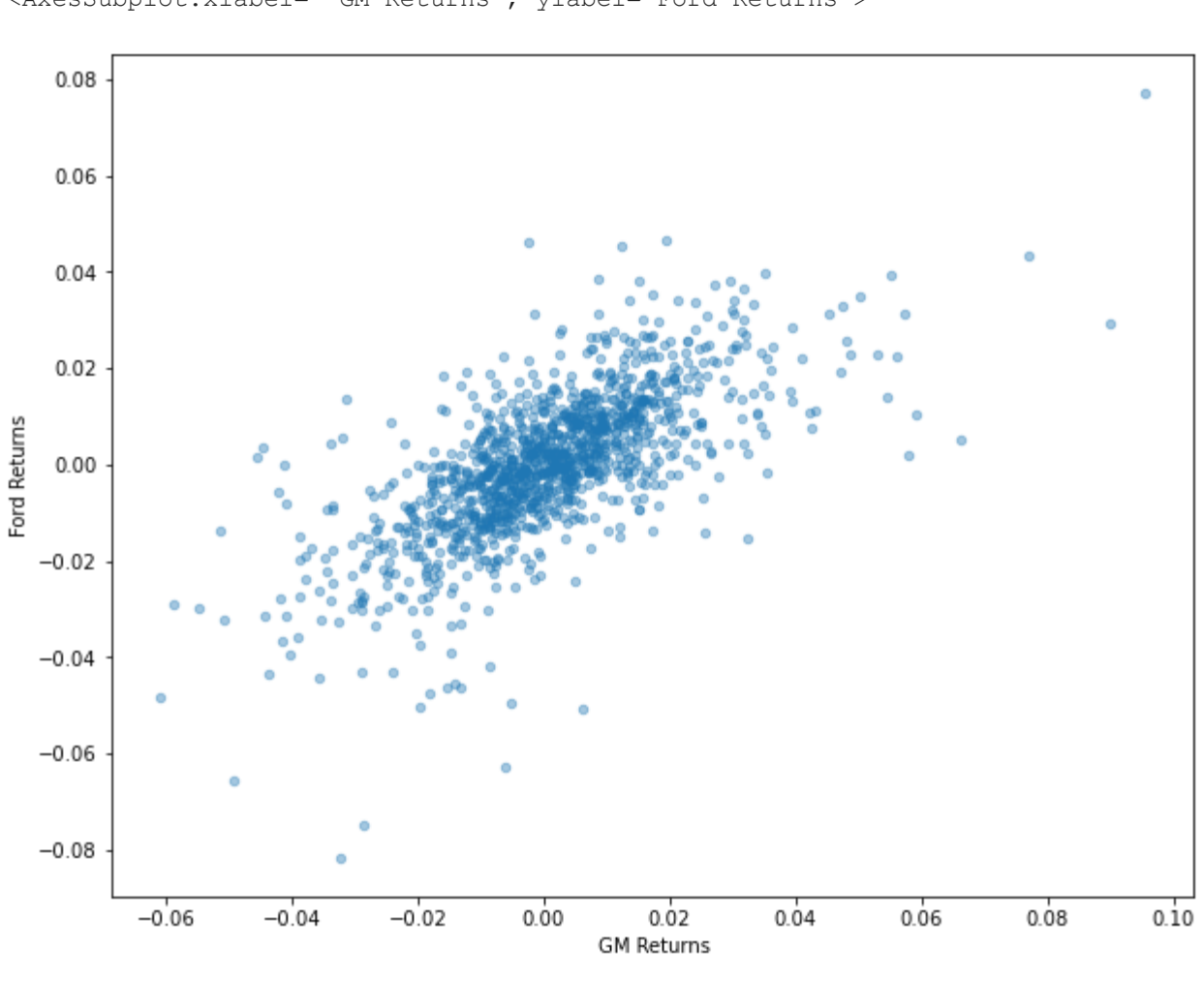
In [38]:
```
scatter_matrix(box_df,figsize=(8,8),alpha=0.2,hist_kwds={'bins':50});
```



It looks like Ford and GM do have some sort of possible relationship, let's plot just these two against eachother in scatter plot to view this more closely!

In [39]:
```
box_df.plot(kind='scatter',x=' GM Returns',y='Ford Returns',alpha=0.4,figsize=(10,8))
```

Out[39]: <AxesSubplot:xlabel=' GM Returns', ylabel='Ford Returns'>



Cumulative Daily Returns

Great! Now we can see which stock was the most wide ranging in daily returns (you should have realized it was Tesla, our original stock price plot should have also made that obvious).

With daily cumulative returns, the question we are trying to answer is the following, if I invested $1 in the company at the beginning of the time series, how much would is be worth today? This is different than just the stock price at the current day, because it will take into account the daily returns. Keep in mind, our simple calculation here won't take into account stocks that give back a dividend. Let's look at some simple examples:

Lets us say there is a stock 'ABC' that is being actively traded on an exchange. ABC has the following prices corresponding to the dates given

```
Date                        Price
01/01/2018                   10
01/02/2018                   15
01/03/2018                   20
01/04/2018                   25
```

Daily Return : Daily return is the profit/loss made by the stock compared to the previous day. (This is what ew just calculated above). A value above one indicates profit, similarly a value below one indicates loss. It is also expressed in percentage to convey the information better. (When expressed as percentage, if the value is above 0, the stock had give you profit else loss). So for the above example the daily returns would be

```
Date                         Daily Return                 %Daily Return
01/01/2018                 10/10 =  1                           -
01/02/2018                 15/10 =  3/2                        50%
01/03/2018                 20/15 =  4/3                        33%
01/04/2018                 25/20 =  5/4                        20%
```

Cumulative Return: While daily returns are useful, it doesn't give the investor a immediate insight into the gains he had made till date, especially if the stock is very volatile. Cumulative return is computed relative to the day investment is made. If cumulative return is above one, you are making profits else you are in loss. So for the above example cumulative gains are as follows

```
Date                       Cumulative Return         %Cumulative Return
01/01/2018                  10/10 =  1                         100 %
01/02/2018                  15/10 =  3/2                       150 %
01/03/2018                  20/10 =  2                         200 %
01/04/2018                  25/10 =  5/2                       250 %
```

The formula for a cumulative daily return is:

Here we can see we are just multiplying our previous investment at i at t-1 by 1+our percent returns. Pandas makes this very simple to calculate with its cumprod() method. Using something in the following manner:

```
df[daily_cumulative_return] = ( 1 + df[pct_daily_return] ).cumprod()
```

Create a cumulative daily return column for each car company's dataframe.

In [41]:
```
tesla['Cumulative Return'] = (1 + tesla['returns']).cumprod()
tesla.head()
```

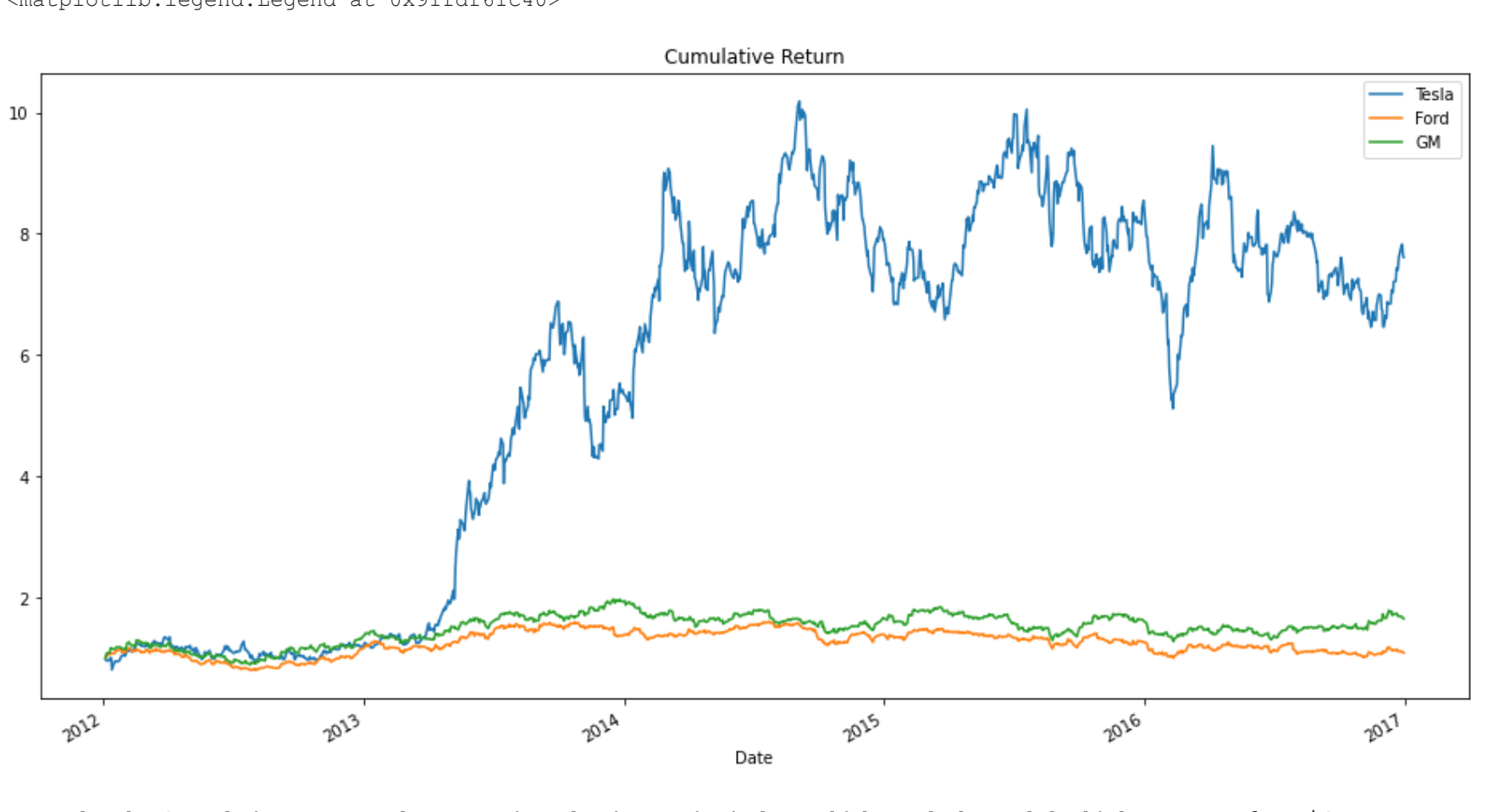
Out[41]:

Date	Open	High	Low	Close	Adj Close	Volume	Total Traded	returns	Cumulative Return
2012-01-03	1.929333	1.966667	1.843333	1.872000	1.872000	13921500	2.685921e+07	NaN	NaN
2012-01-04	1.880667	1.911333	1.833333	1.847333	1.847333	9451500	1.777512e+07	-0.013177	0.986823
2012-01-05	1.850667	1.862000	1.790000	1.808000	1.808000	15082500	2.791269e+07	-0.021292	0.965812
2012-01-06	1.813333	1.852667	1.760667	1.794000	1.794000	14794500	2.682736e+07	-0.007743	0.958333
2012-01-09	1.800000	1.832667	1.741333	1.816667	1.816667	13455000	2.421900e+07	0.012635	0.970442

In [46]:
```
ford['Cumulative Return'] = (1 + ford['returns']).cumprod()
gm['Cumulative Return'] = (1 + gm['returns']).cumprod()
```

In [47]:
```
tesla['Cumulative Return'].plot(label='Tesla',figsize=(16,8),title='Cumulative Return')
ford['Cumulative Return'].plot(label='Ford')
gm['Cumulative Return'].plot(label='GM')
plt.legend()
```

Out[47]: <matplotlib.legend.Legend at 0x91fdf61c40>



Now plot the Cumulative Return columns against the time series index. Which stock showed the highest return for a $1 invested? Which showed the lowest?

In [ ]:

Great Job!

That is it for thsi very basic analysis, this concludes this half of the course, which focuses much more on learning the tools of the trade. The second half of the course is where we really dive into functionality designed for time series, quantitative analysis, algorithmic trading, and much more!