Turkish Aviation Stock Market Analysis Project (Burakhan Sinal)

That is a very basic analysis but you can just change stock market name and make an analysise

quantitative trading techniques and platforms.

https://www.linkedin.com/in/burakhan-sinal/ We'll be analyzing stock data related to a few car companies, from Jan 1 2023.Connect yahoo finance 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 [1]:
```
import numpy as np
import pandas as pd
import pandas_datareader.data as web
import datetime as dt
import matplotlib. pyplot as plt
import yfinance as yf # connection yahoo finance  python3 -m pip install yfinanc
yf.pdr_override() # <== that's all it takes :-)
from pandas_datareader import data as pdr
```

Part 1: Getting the Data yahoo finance

TUrkish Aviation Stock (Ticker: THY and Pegasus)

* 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...).

In [2]:
```
#Get yahoo finance Tesla data its take more than 10s dont worry :)
THYAO = pdr.get_data_yahoo('THYAO.IS', start='2023-1-1')
```

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

In [3]:
```
THYAO
```

Out[3]:

Date	Open	High	Low	Close	Adj Close	Volume
2023-01-02	142.399994	147.100006	142.100006	146.800003	146.800003	60745183
2023-01-03	148.300003	150.399994	146.399994	147.399994	147.399994	73564352
2023-01-04	147.899994	148.199997	143.600006	144.500000	144.500000	50055440
2023-01-05	144.800003	147.199997	140.300003	141.800003	141.800003	66261587
2023-01-06	135.199997	144.199997	132.399994	142.500000	142.500000	84430370
...	...	...	...	...	...	...
2023-12-25	232.399994	232.399994	223.899994	226.199997	226.199997	16966698
2023-12-26	227.399994	232.699997	226.100006	227.699997	227.699997	24860552
2023-12-27	227.600006	229.500000	221.600006	222.199997	222.199997	25992461
2023-12-28	222.500000	231.000000	222.500000	229.000000	229.000000	25168141
2023-12-29	229.000000	231.000000	227.899994	228.600006	228.600006	17712965

249 rows × 6 columns

In [6]:
```
#PGSUS.IS Data
PGSUS = pdr.get_data_yahoo('PGSUS.IS', start='2023-1-1')
```

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

In [7]:
```
#Pegasus
PGSUS.head()
```

Out[7]:

Date	Open	High	Low	Close	Adj Close	Volume
2023-01-02	484.200012	504.299988	483.000000	503.899994	503.899994	3238738
2023-01-03	506.600006	508.600006	492.500000	492.700012	492.700012	3979465
2023-01-04	494.299988	499.500000	482.000000	483.299988	483.299988	3323756
2023-01-05	486.000000	493.500000	469.299988	477.500000	477.500000	3937310
2023-01-06	455.000000	484.100006	439.899994	478.399994	478.399994	3998544

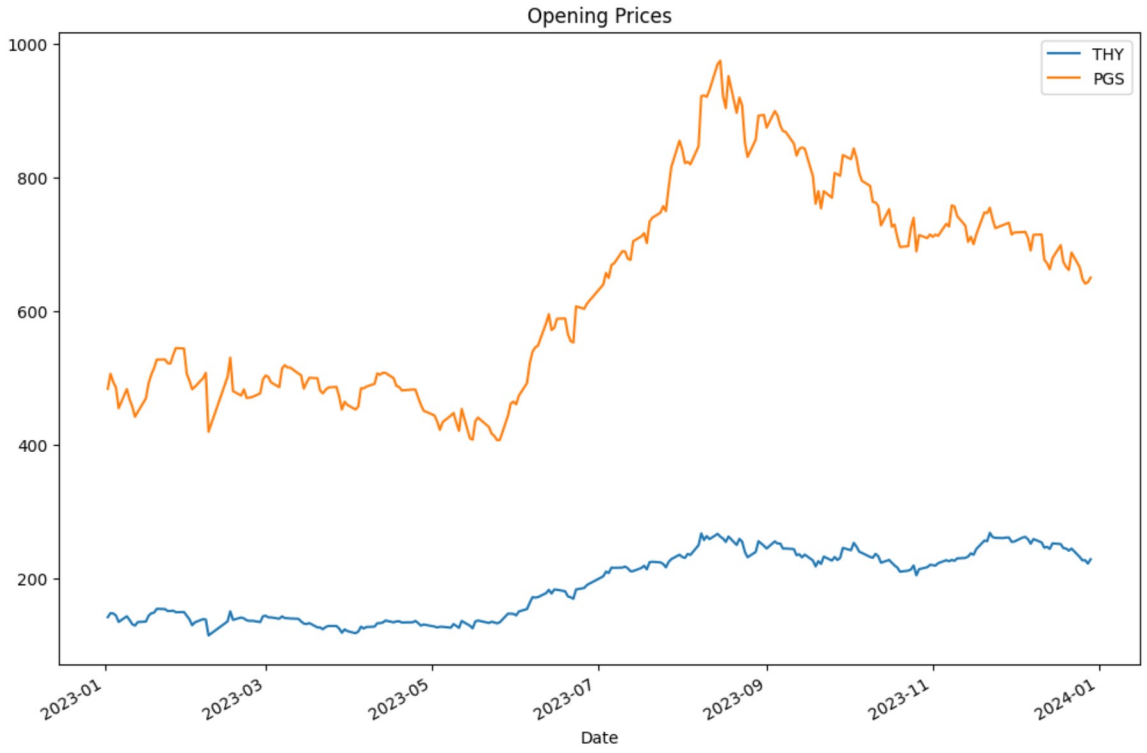
Part 2: Visualizing the Data

Time to visualize the data.

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

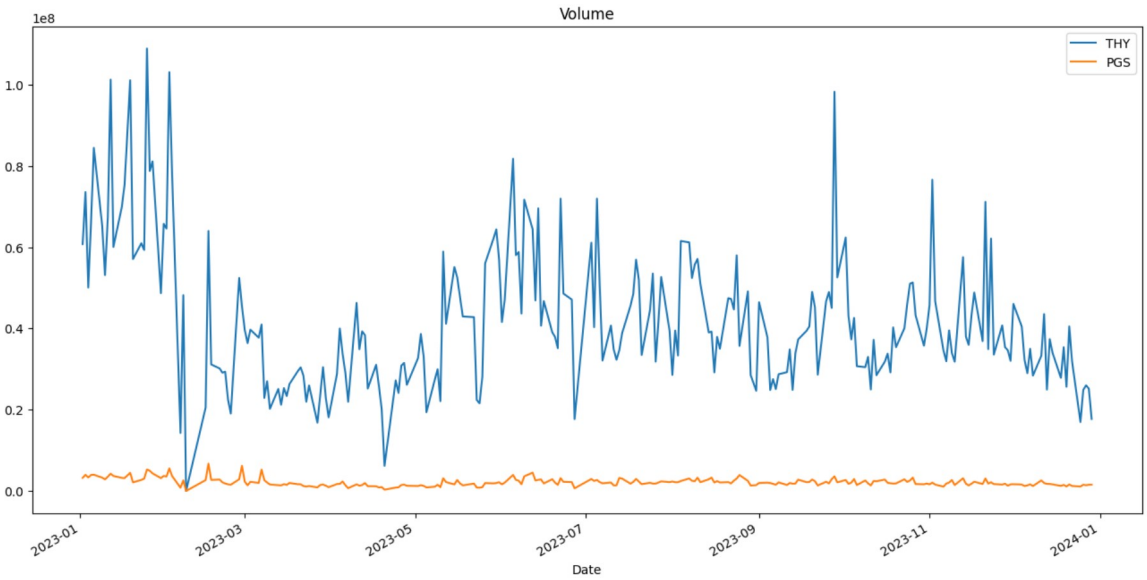
Recreate this linear plot of all the stocks' Open price ! Hint: For the legend, use label parameter and plt.legend()

In [8]:
```
THYAO["Open"].plot(label="THY",figsize=(12,8),title='Opening Prices')
PGSUS["Open"].plot(label="PGS")
plt.legend();
```



Plot the Volume of stock traded each day.

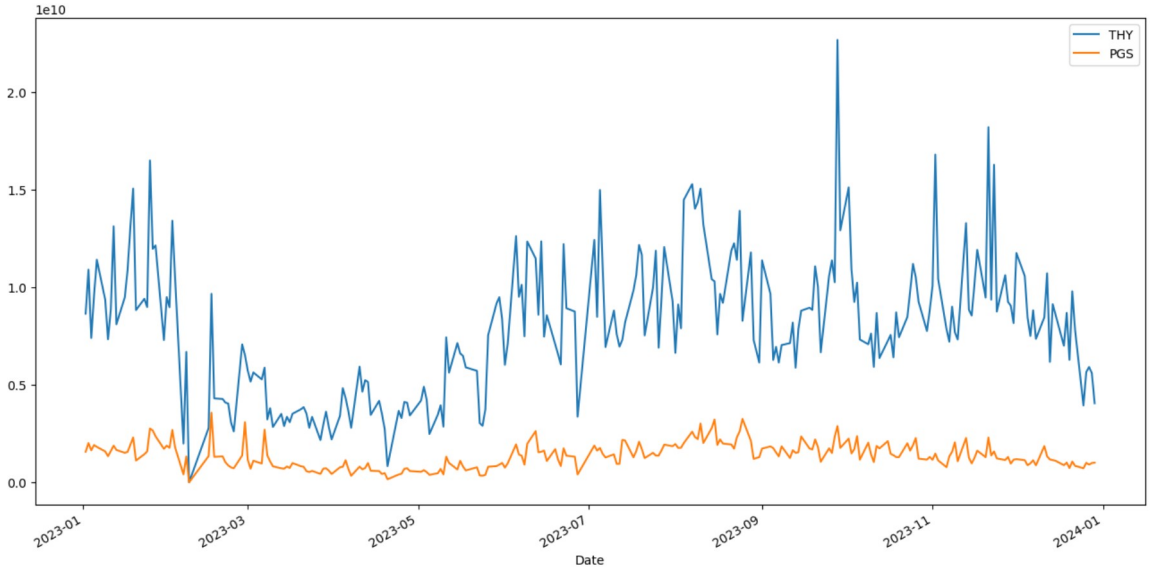
In [10]:
```
THYAO['Volume'].plot(label='THY',figsize=(16,8),title='Volume')
PGSUS['Volume'].plot(label='PGS')
plt.legend();
```



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

In [11]:
```
THYAO['Total Traded'] = THYAO['Open']*THYAO['Volume']
PGSUS['Total Traded'] = PGSUS['Open']*PGSUS['Volume']
```

In [12]:
```
THYAO['Total Traded'].plot(label="THY",figsize=(16,8))
PGSUS['Total Traded'].plot(label="PGS")
plt.legend();
```

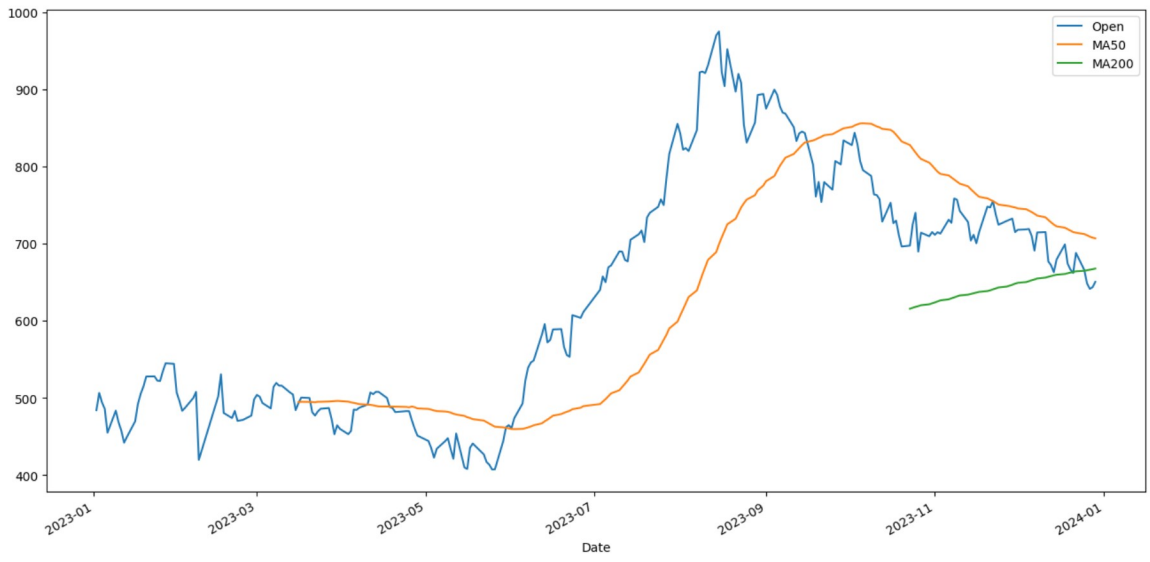


Plot this "Total Traded" against the time index.

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

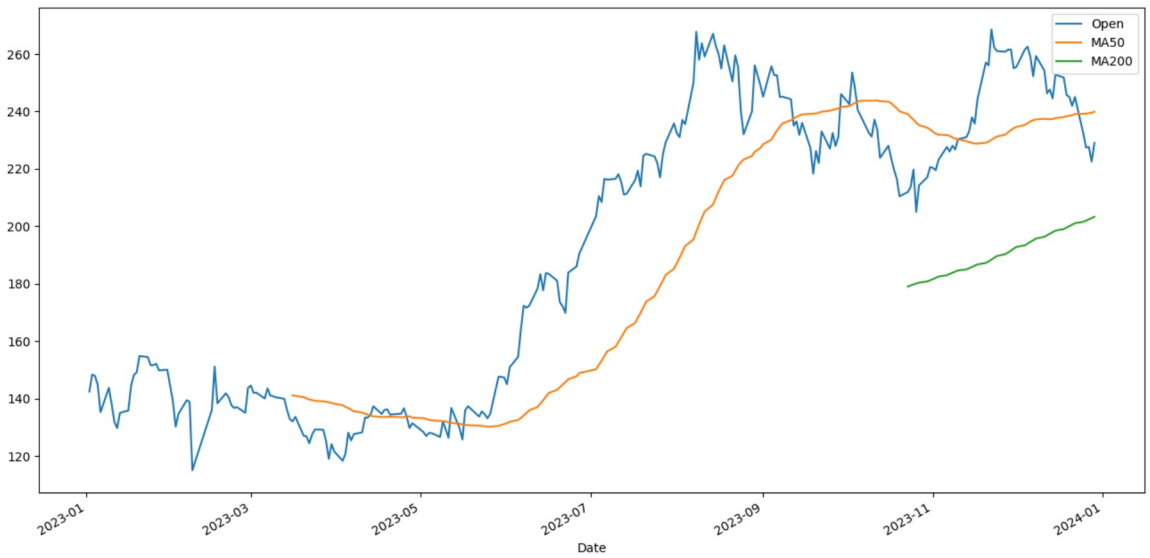
In [13]:
```
PGSUS["MA50"]= PGSUS["Open"].rolling(50).mean()
PGSUS["MA200"]= PGSUS["Open"].rolling(200).mean()
PGSUS[["Open","MA50","MA200"]].plot(figsize=(16,8))
```

Out[13]:
```
<Axes: xlabel='Date'>
```



In [14]:
```
THYAO["MA50"]= THYAO["Open"].rolling(50).mean()
THYAO["MA200"]= THYAO["Open"].rolling(200).mean()
THYAO[["Open","MA50","MA200"]].plot(figsize=(16,8))
```

Out[14]: <Axes: xlabel='Date'>



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 [15]:
```
from pandas.plotting import scatter_matrix
```

In [16]:
```
av_comp = pd.concat([THYAO['Open'],PGSUS['Open']],axis=1)
```

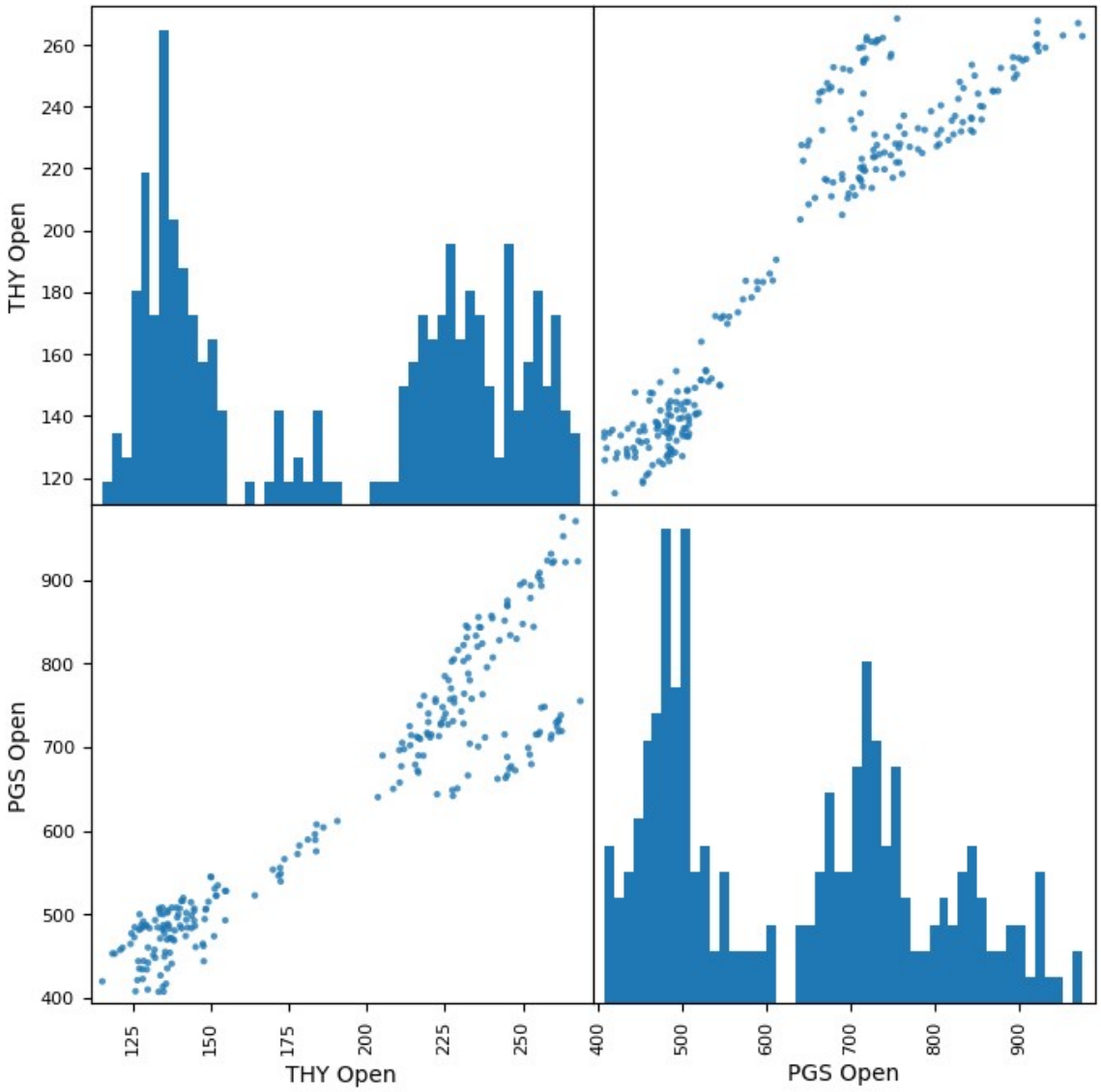
In [17]:
```
av_comp.head()
```

Out[17]:

Date	Open	Open
2023-01-02	142.399994	484.200012
2023-01-03	148.300003	506.600006
2023-01-04	147.899994	494.299988
2023-01-05	144.800003	486.000000
2023-01-06	135.199997	455.000000

In [18]:
```
av_comp.columns = ['THY Open','PGS Open']
```

In [20]:
```
# You can use a semi-colon to remove the axes print outs
scatter_matrix(av_comp,figsize=(8,8),alpha=0.8,hist_kwds={'bins':50});
```



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:

$$r_t = \frac{p_t}{p_{t-1}} - 1$$

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 [21]:
```
# Method 1: Using shift
THYAO['returns'] = (THYAO['Close'] / THYAO['Close'].shift(1) ) - 1
```

In [22]:
```
THYAO.head()
```

Out[22]:

Date	Open	High	Low	Close	Adj Close	Volume	Total Trade
2023-01-02	142.399994	147.100006	142.100006	146.800003	146.800003	60745183	8.650114e+0
2023-01-03	148.300003	150.399994	146.399994	147.399994	147.399994	73564352	1.090959e+1
2023-01-04	147.899994	148.199997	143.600006	144.500000	144.500000	50055440	7.403199e+0
2023-01-05	144.800003	147.199997	140.300003	141.800003	141.800003	66261587	9.594678e+0
2023-01-06	135.199997	144.199997	132.399994	142.500000	142.500000	84430370	1.141499e+1

In [23]:
```
THYAO['returns'] = THYAO['Close'].pct_change(1)
```

In [24]:
```
THYAO.head()
```

Out[24]:

Date	Open	High	Low	Close	Adj Close	Volume	Total Trade
2023-01-02	142.399994	147.100006	142.100006	146.800003	146.800003	60745183	8.650114e+0
2023-01-03	148.300003	150.399994	146.399994	147.399994	147.399994	73564352	1.090959e+1
2023-01-04	147.899994	148.199997	143.600006	144.500000	144.500000	50055440	7.403199e+0
2023-01-05	144.800003	147.199997	140.300003	141.800003	141.800003	66261587	9.594678e+0
2023-01-06	135.199997	144.199997	132.399994	142.500000	142.500000	84430370	1.141499e+1

In [25]:
```
# Now repeat for the other dataframes
THYAO['returns'] = THYAO['Close'].pct_change(1)
PGSUS['returns'] = PGSUS['Close'].pct_change(1)
```

In [26]:
```
PGSUS.head()
```

Out[26]:

Date	Open	High	Low	Close	Adj Close	Volume	Total Traded
2023-01-02	484.200012	504.299988	483.000000	503.899994	503.899994	3238738	1.568197e+09
2023-01-03	506.600006	508.600006	492.500000	492.700012	492.700012	3979465	2.015997e+09
2023-01-04	494.299988	499.500000	482.000000	483.299988	483.299988	3323756	1.642933e+09
2023-01-05	486.000000	493.500000	469.299988	477.500000	477.500000	3937310	1.913533e+09
2023-01-06	455.000000	484.100006	439.899994	478.399994	478.399994	3998544	1.819338e+09

In [27]:
```
THYAO.head()
```

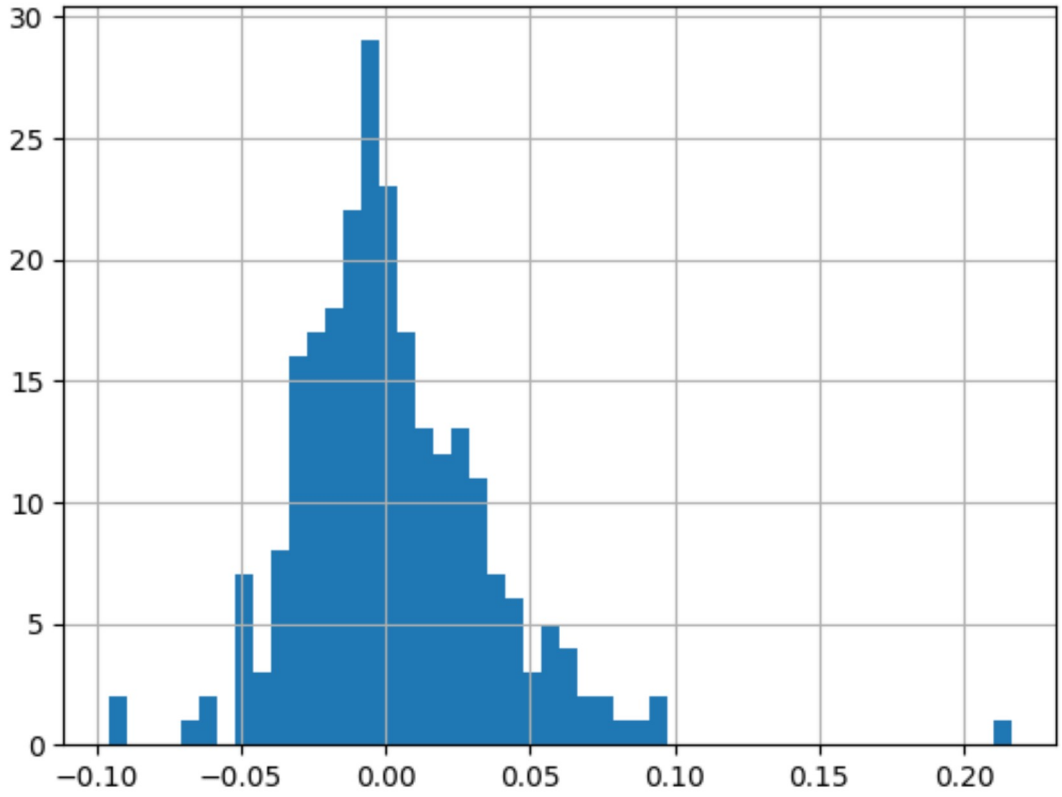
Out[27]:

Date	Open	High	Low	Close	Adj Close	Volume	Total Trade
2023-01-02	142.399994	147.100006	142.100006	146.800003	146.800003	60745183	8.650114e+0
2023-01-03	148.300003	150.399994	146.399994	147.399994	147.399994	73564352	1.090959e+1
2023-01-04	147.899994	148.199997	143.600006	144.500000	144.500000	50055440	7.403199e+0
2023-01-05	144.800003	147.199997	140.300003	141.800003	141.800003	66261587	9.594678e+0
2023-01-06	135.199997	144.199997	132.399994	142.500000	142.500000	84430370	1.141499e+1

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.)

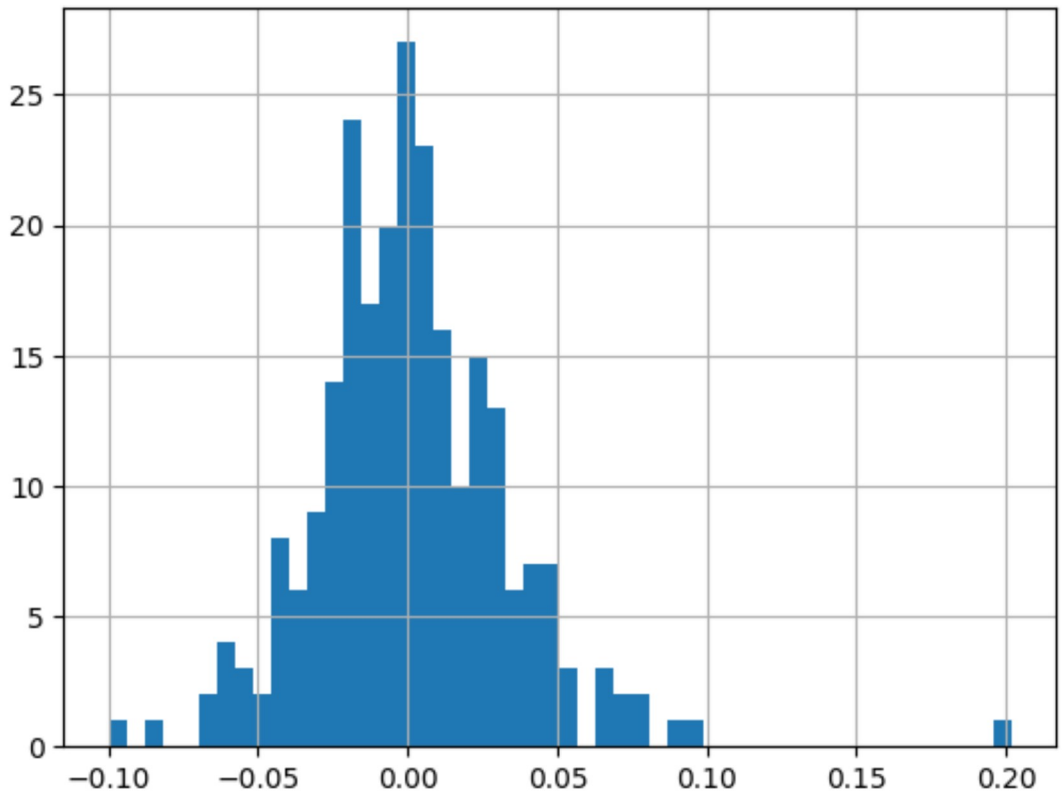
In [28]:
```
THYAO['returns'].hist(bins=50)
```

Out[28]:
```
<Axes: >
```



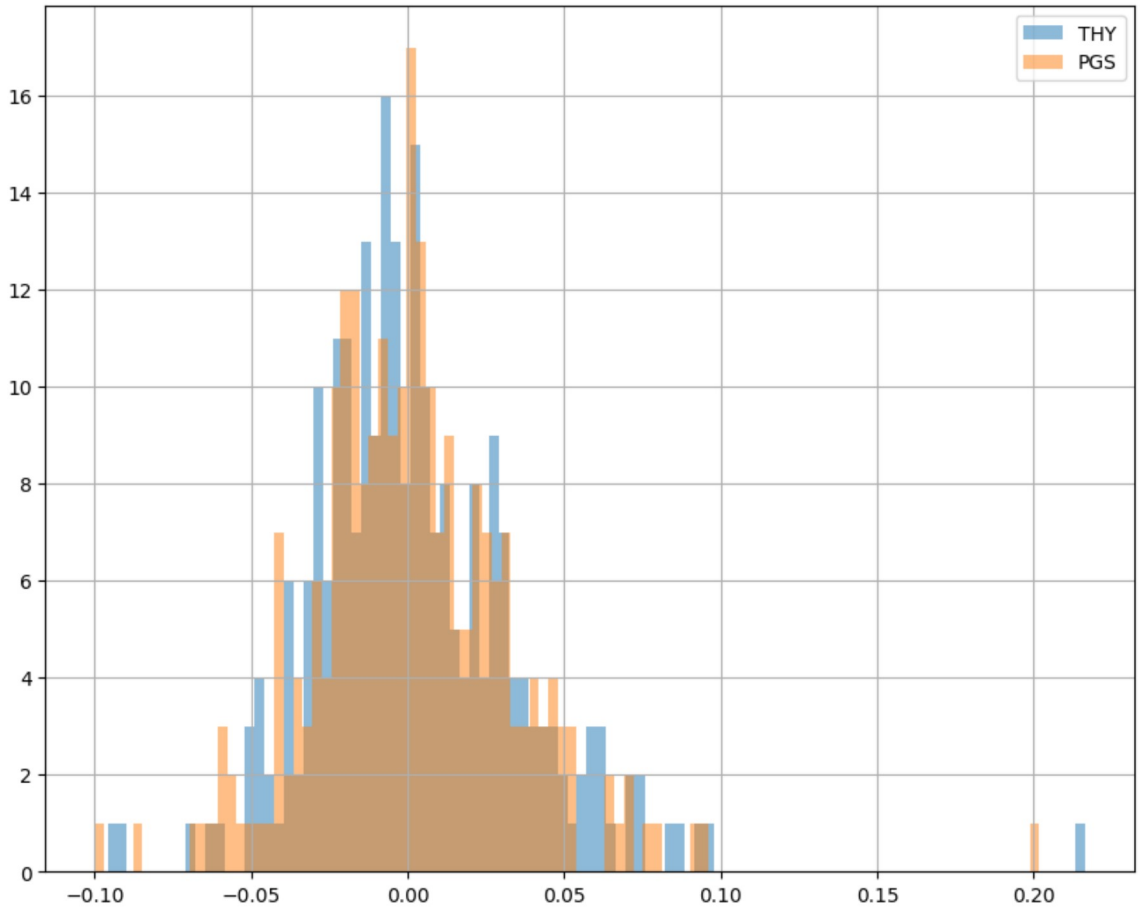
In [29]:
```
PGSUS['returns'].hist(bins=50)
```

Out[29]:
```
<Axes: >
```



In [30]:
```
THYAO['returns'].hist(bins=100,label='THY',figsize=(10,8),alpha=0.5)
PGSUS['returns'].hist(bins=100,label='PGS',alpha=0.5)
plt.legend()
```

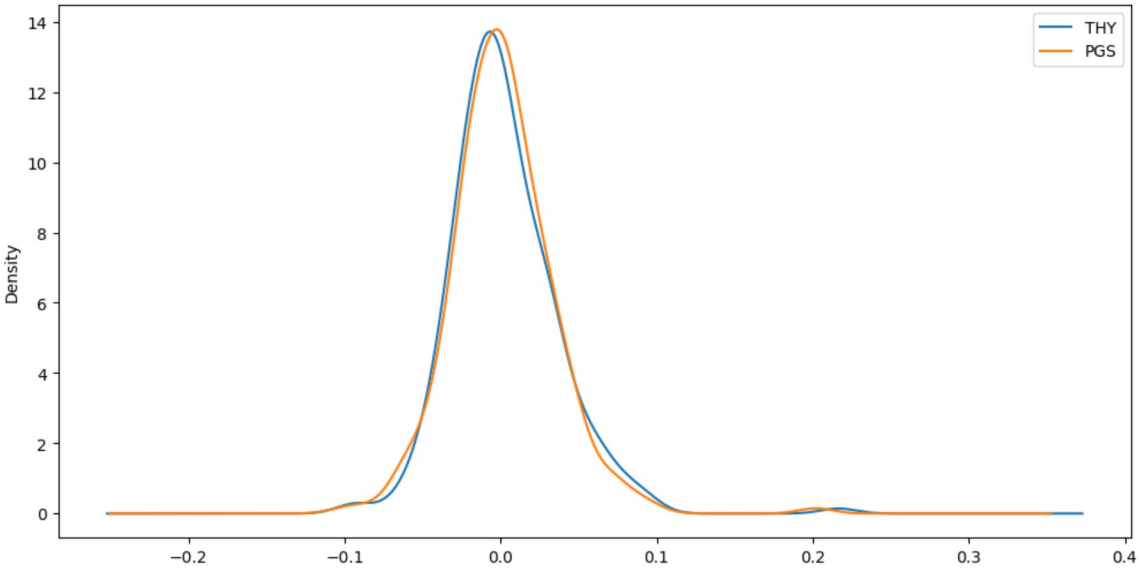
Out[30]:
```
<matplotlib.legend.Legend at 0x232d3146860>
```



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

In [31]:
```
THYAO['returns'].plot(kind='kde',label='THY',figsize=(12,6))
PGSUS['returns'].plot(kind='kde',label='PGS')
plt.legend()
```

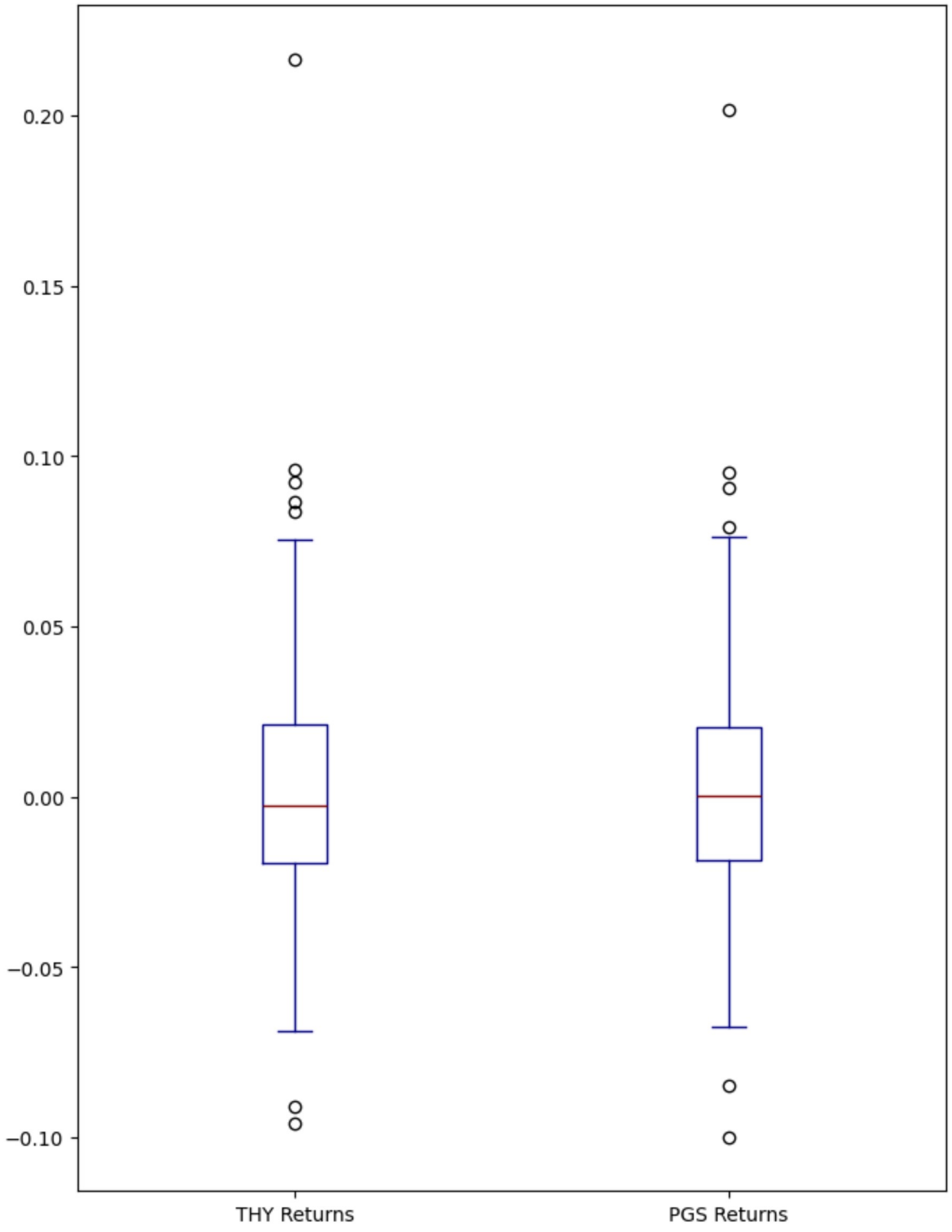
Out[31]:
```
<matplotlib.legend.Legend at 0x232d3892260>
```



Try also creating some box plots comparing the returns.

In [35]:
```
box_df = pd.concat([THYAO['returns'],PGSUS['returns']],axis=1)
box_df.columns = ['THY Returns','PGS Returns']
box_df.plot(kind='box',figsize=(8,11),colormap='jet')
```

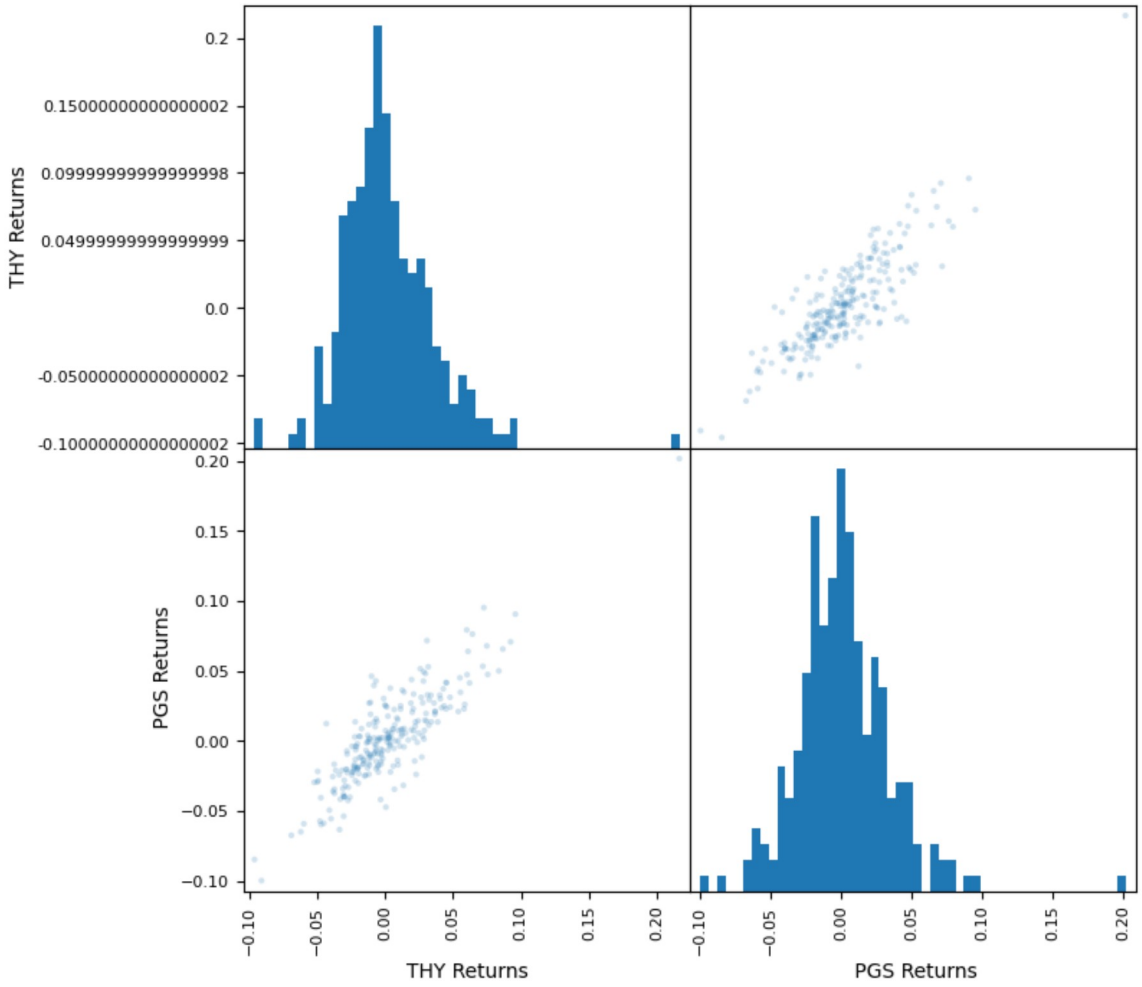
Out[35]: <Axes: >



Comparing Daily Returns between Stocks

** Create a scatter matrix plot to see the correlation between each of the stocks daily returns.

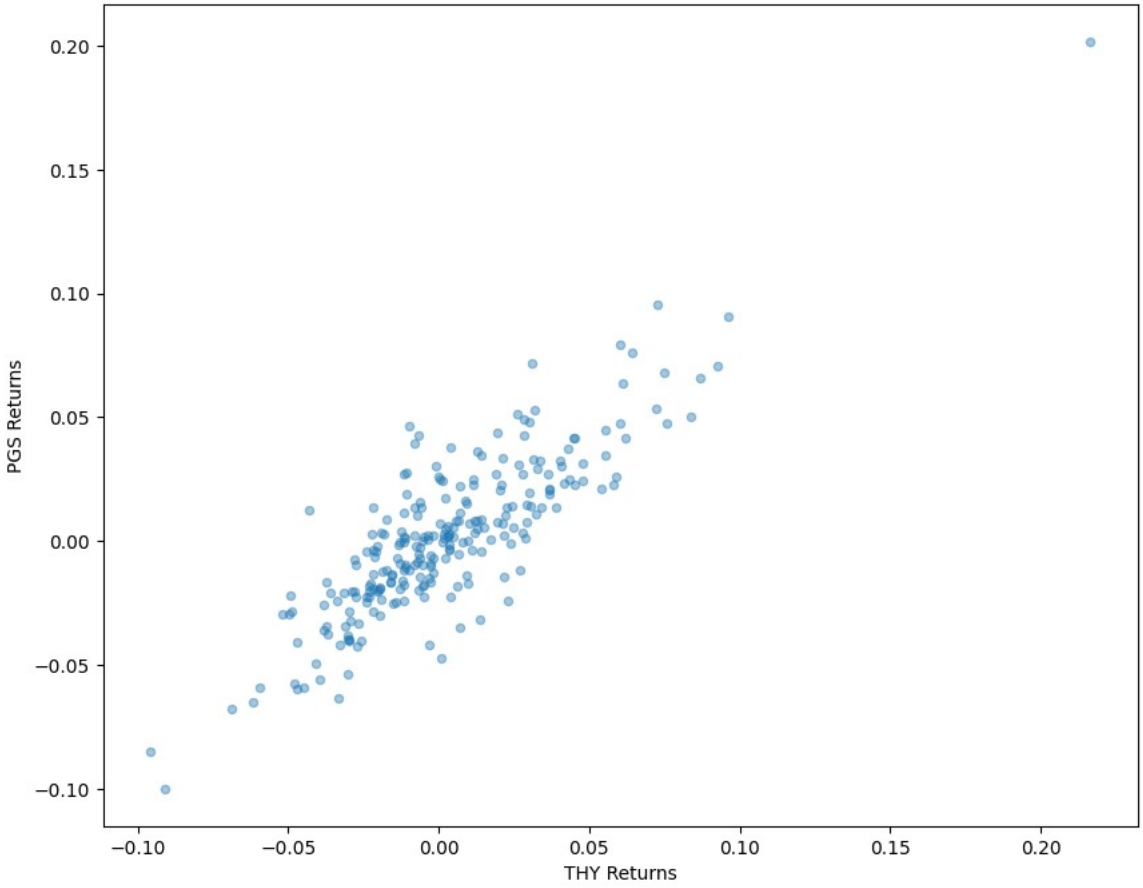
In [33]:
```
scatter_matrix(box_df,figsize=(8,8),alpha=0.2,hist_kwds={'bins':50});
```



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

In [36]:
```
box_df.plot(kind='scatter',x='THY Returns',y='PGS Returns',alpha=0.4,figsize=(10
```

Out[36]:
```
<Axes: xlabel='THY Returns', ylabel='PGS 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:

$i_i = (1 + r_t) * i_{t-1}$

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 [37]:
```
THYAO['Cumulative Return'] = (1 + THYAO['returns']).cumprod()
```

In [38]:
```
THYAO.head()
```

Out[38]:

Date	Open	High	Low	Close	Adj Close	Volume	Total Trade
2023-01-02	142.399994	147.100006	142.100006	146.800003	146.800003	60745183	8.650114e+0
2023-01-03	148.300003	150.399994	146.399994	147.399994	147.399994	73564352	1.090959e+1
2023-01-04	147.899994	148.199997	143.600006	144.500000	144.500000	50055440	7.403199e+0
2023-01-05	144.800003	147.199997	140.300003	141.800003	141.800003	66261587	9.594678e+0
2023-01-06	135.199997	144.199997	132.399994	142.500000	142.500000	84430370	1.141499e+1

In [39]:
```
THYAO['Cumulative Return'] = (1 + THYAO['returns']).cumprod()
PGSUS['Cumulative Return'] = (1 + PGSUS['returns']).cumprod()
```

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 [40]:
```
THYAO['Cumulative Return'].plot(label='THY',figsize=(16,8),title='Cumulative Ret
PGSUS['Cumulative Return'].plot(label='PGS')
plt.legend()
```

Out[40]:
```
<matplotlib.legend.Legend at 0x232d32cc280>
```