FINAL CAPSTONE PROJECT 2

- In this project, we will visualize stock prices using Seaborn and Matplotlib. The 4 stocks that will be visualized are JP Morgan Chase (JP), Procter and Gamble (P&G) (PG), Apple (AAPL) and United Airlines (UAL).
- Using the “stock_daily_prices.csv” and “stocks_daily_returns.csv” included in the attachment, write Python scripts to perform the following tasks:
  - Import both data using Pandas.
  - Using Matplotlib, plot a lineplot showing all daily prices of 4 stocks in one figure.
  - Using Matplotlib, plot the daily prices of the 4 stocks in several subplots.
  - Using Matplotlib, visualize 4 plots on subplots next to each other (all figures in one row).
  - Using Matplotlib, visualize a scatterplot between Apple's daily returns and JP Morgan's daily returns.
  - Using Seaborn, visualize a scatterplot between Apple's daily returns and JP Morgan's daily returns.
  - Assume that you decide to be bullish on AAPL and you allocate 70% of your assets in it. You also decide to divide your remaining assets equally among other stocks (JPM, PG, and UAL). Using Matplotlib, plot a piechart showing these allocations. Use the ‘explode’ attribute to increase separation between AAPL and the rest of the portfolio.
  - Using Matplotlib, plot histograms for United Airlines and P&G returns using 40 bins in red. Show the average and Standard deviation for both stocks above the image. What do you conclude from the graph?
  - Using Seaborn, plot a heatmap showing the correlation between daily stock returns. Comment on the correlation between UAL and P&G.
  - Create a 3D plot showing all daily returns from JPM, AAPL, and UAL [need to figure out how to do this personally].

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [ ]:
```
daily_prices_df = pd.read_csv('stocks_daily_prices.csv')
daily_returns_df = pd.read_csv('stocks_daily_returns.csv')
```

In [ ]:
```
daily_prices_df
```

Out[ ]:

	Date	AAPL	JPM	PG	UAL
0	1/2/2008	5.949703	29.448614	47.058838	29.915234
1	1/3/2008	5.952452	29.246094	47.058838	29.690800
2	1/4/2008	5.498071	28.582682	46.870098	31.000002
3	1/7/2008	5.424478	28.868999	47.175980	29.180000
4	1/8/2008	5.229351	27.723747	47.299629	24.389999
...	...	...	...	...	...
3637	6/13/2022	131.880005	115.989998	138.149994	37.020000
3638	6/14/2022	132.759995	114.059998	133.839996	36.990002
3639	6/15/2022	135.429993	115.410004	132.509995	37.889999
3640	6/16/2022	130.059998	113.430000	133.320007	34.779999
3641	6/17/2022	131.559998	113.029999	132.360001	36.279999

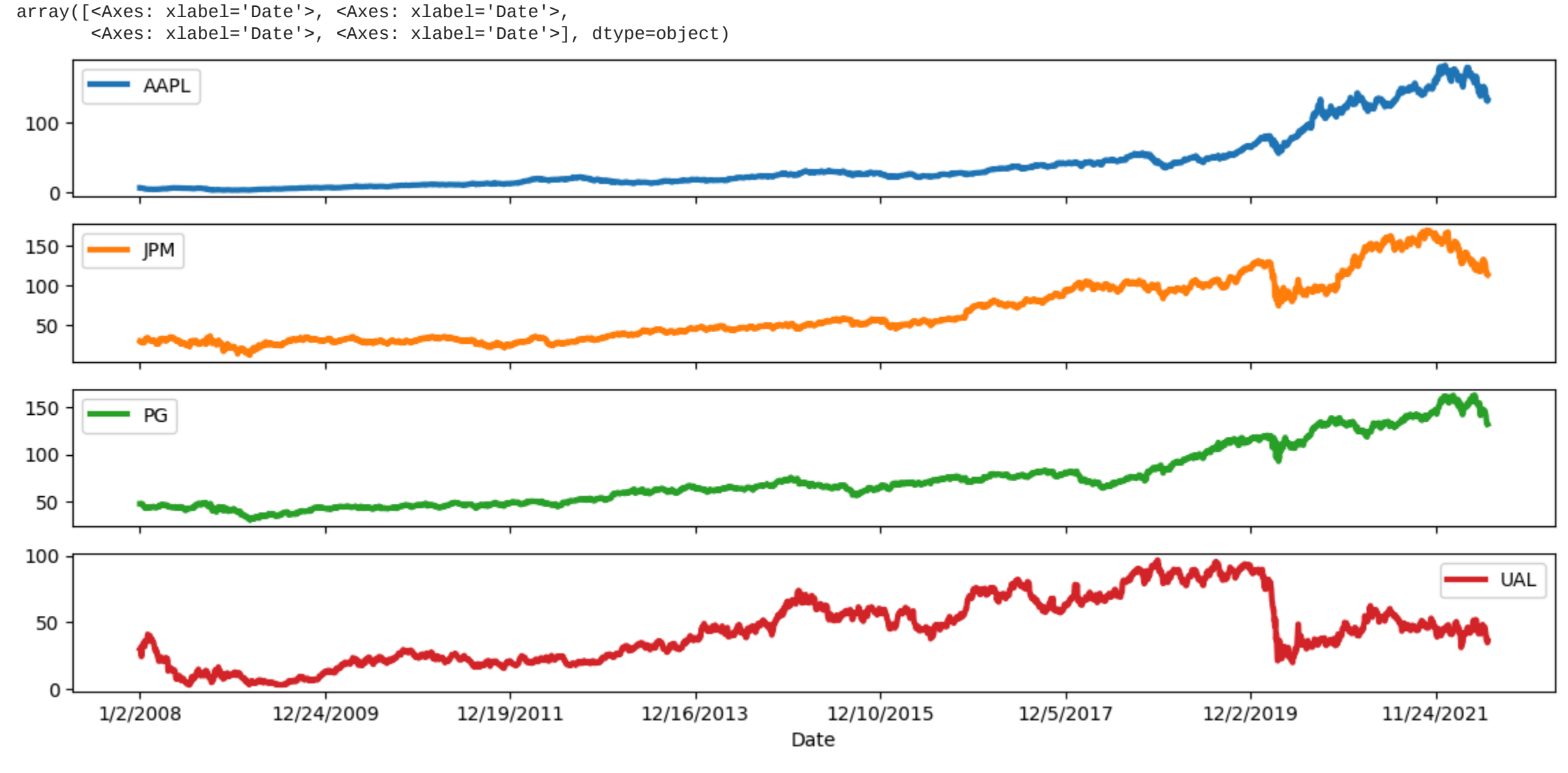
3642 rows × 5 columns

In [ ]:
```
daily_prices_df.plot(x = 'Date', y = [ 'AAPL', 'JPM', 'PG', 'UAL'], linewidth = 3, figsize = (14,6))
```

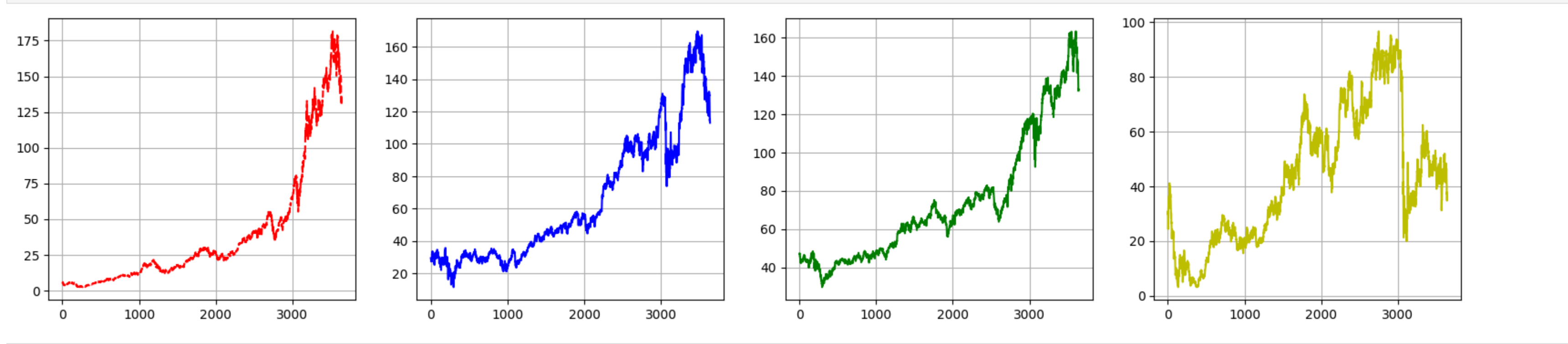
Out[ ]:
```
<Axes: xlabel='Date'>
```

In [ ]:
```
daily_prices_df.plot(x = 'Date', y = [ 'AAPL', 'JPM', 'PG', 'UAL'], linewidth = 3, figsize = (14,6), subplots=True)
```

Out[ ]:
```
array([<Axes: xlabel='Date'>, <Axes: xlabel='Date'>,
       <Axes: xlabel='Date'>, <Axes: xlabel='Date'>], dtype=object)
```



In [ ]:
```
plt.figure(figsize=(20,4))

plt.subplot(1,4,1)
plt.plot(daily_prices_df['AAPL'], 'r--')
plt.grid()

plt.subplot(1,4,2)
plt.plot(daily_prices_df['JPM'], 'b')
plt.grid()

plt.subplot(1,4,3)
plt.plot(daily_prices_df['PG'], 'g')
plt.grid()

plt.subplot(1,4,4)
plt.plot(daily_prices_df['UAL'], 'y')
plt.grid()
```



In [ ]:
```
daily_returns_df
```

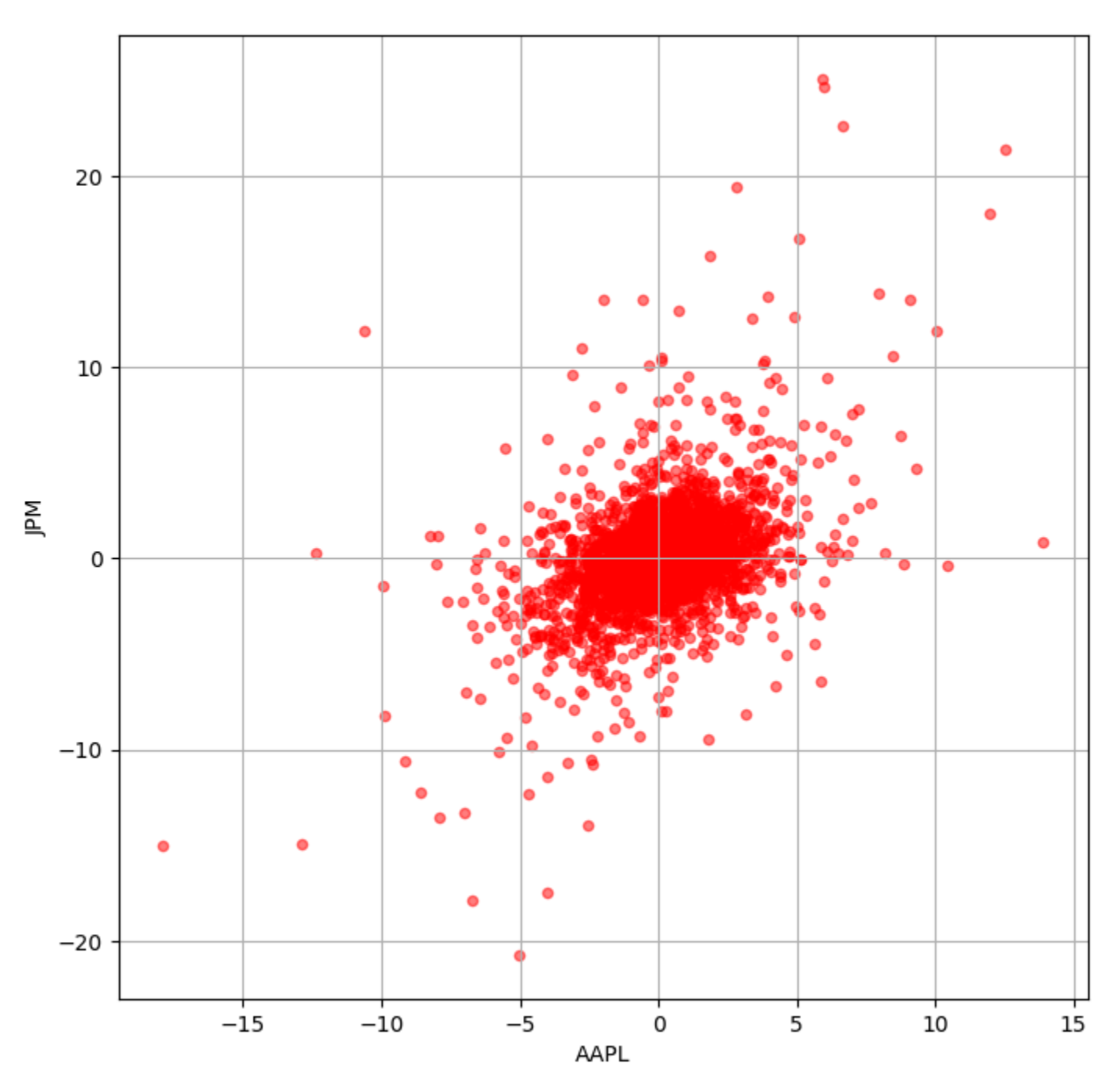
Out[ ]:

	Date	AAPL	JPM	PG	UAL
0	1/2/2008	0.000000	0.000000	0.000000	0.000000
1	1/3/2008	0.046203	-0.687708	0.000000	-0.750233
2	1/4/2008	-7.633517	-2.268378	-0.401072	4.409454
3	1/7/2008	-1.338518	1.001718	0.652615	-5.870973
4	1/8/2008	-3.597157	-3.967066	0.262103	-16.415356
...	...	...	...	...	...
3637	6/13/2022	-3.828484	-2.977838	-2.677001	-10.058308
3638	6/14/2022	0.667265	-1.663937	-3.119796	-0.081034
3639	6/15/2022	2.011147	1.183593	-0.993725	2.433084
3640	6/16/2022	-3.965145	-1.715625	0.611284	-8.207972
3641	6/17/2022	1.153314	-0.352642	-0.720077	4.312824

3642 rows × 5 columns

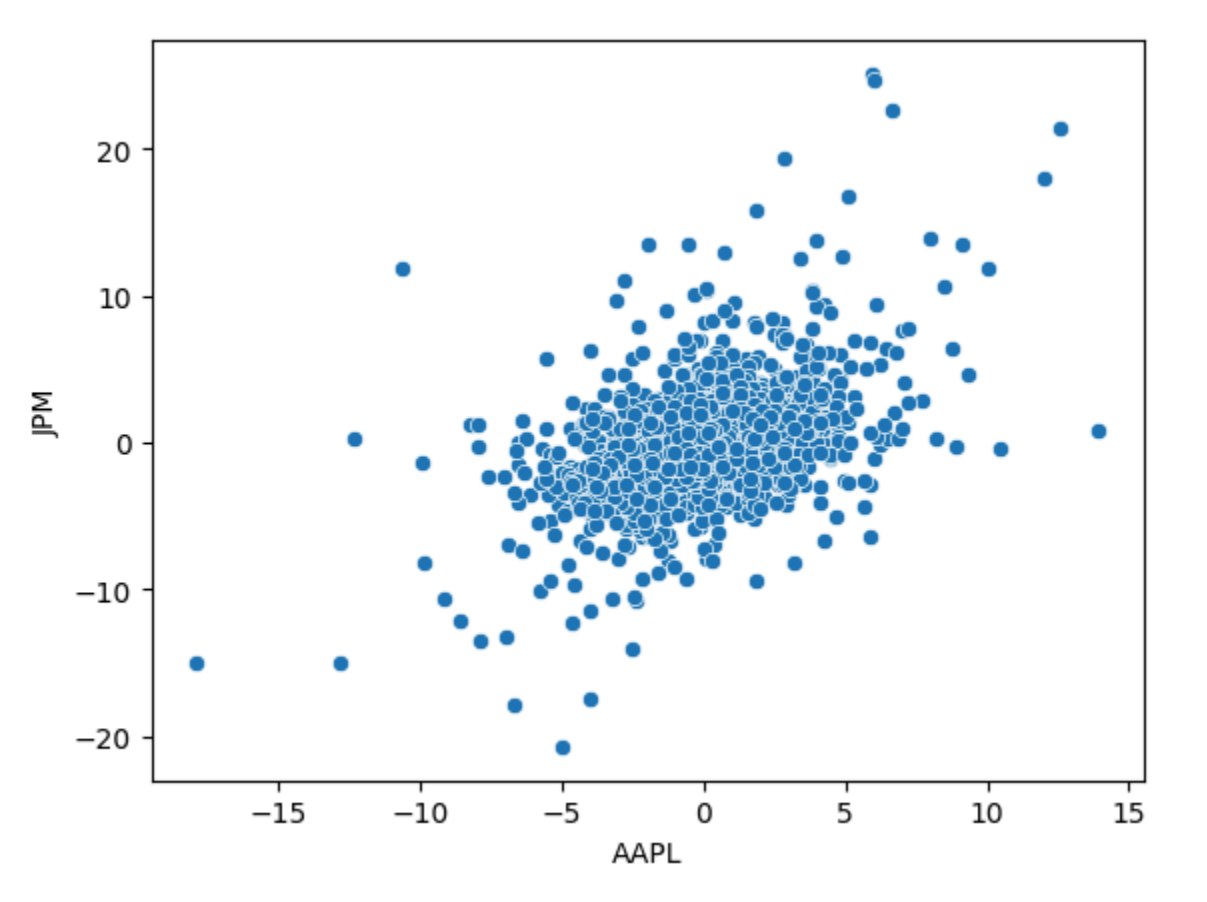
In [ ]:
```
daily_returns_df.plot.scatter('AAPL', 'JPM', grid=True, alpha=0.5, figsize=(8,8), color = 'r')
```

Out[ ]:
```
<Axes: xlabel='AAPL', ylabel='JPM'>
```



In [ ]:
```
sns.scatterplot(data=daily_returns_df, x='AAPL', y='JPM')
```

Out[ ]:
```
<Axes: xlabel='AAPL', ylabel='JPM'>
```



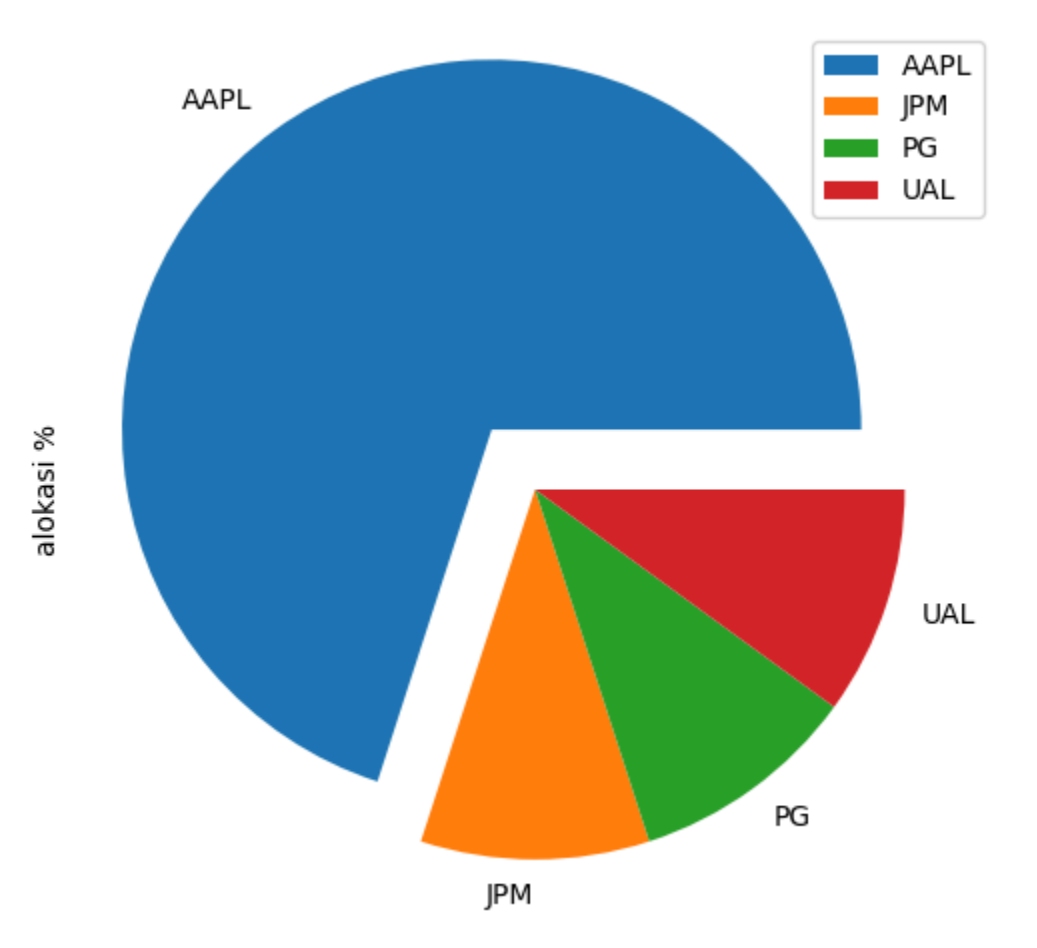
In [ ]:
```
data = {'alokasi %' : [70,10,10,10]}
index = ['AAPL','JPM','PG','UAL']
bullish_pd = pd.DataFrame(data, index)

bullish_pd
```

Out[ ]:

	alokasi %
AAPL	70
JPM	10
PG	10
UAL	10

In [ ]:
```
bullish_pd.plot.pie(y='alokasi %', explode = (0.2,0,0,0), figsize=(6,6))
```

Out[ ]:
```
<Axes: ylabel='alokasi %'>
```



In [ ]:
```
daily_returns_df
```

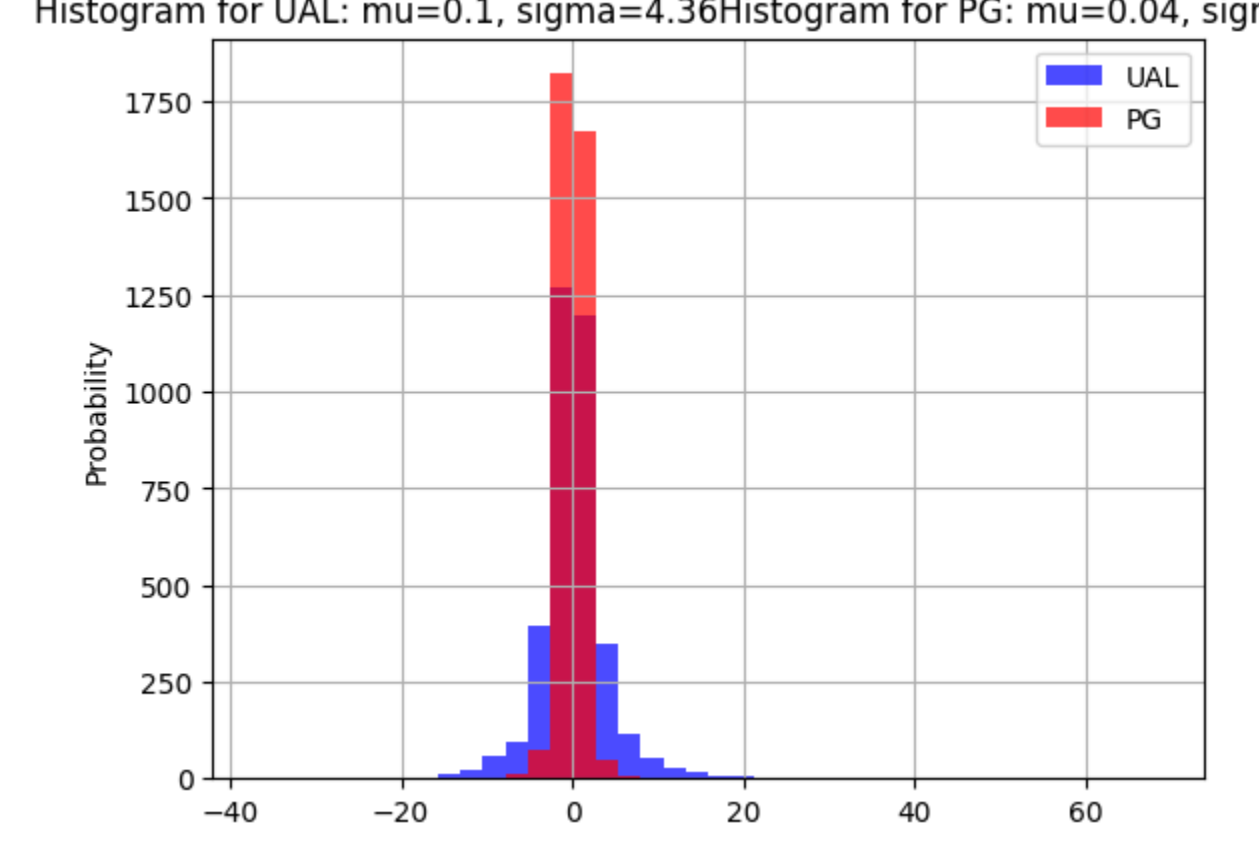
Out[ ]:

	Date	AAPL	JPM	PG	UAL
0	1/2/2008	0.000000	0.000000	0.000000	0.000000
1	1/3/2008	0.046203	-0.687708	0.000000	-0.750233
2	1/4/2008	-7.633517	-2.268378	-0.401072	4.409454
3	1/7/2008	-1.338518	1.001718	0.652615	-5.870973
4	1/8/2008	-3.597157	-3.967066	0.262103	-16.415356
...	...	...	...	...	...
3637	6/13/2022	-3.828484	-2.977838	-2.677001	-10.058308
3638	6/14/2022	0.667265	-1.663937	-3.119796	-0.081034
3639	6/15/2022	2.011147	1.183593	-0.993725	2.433084
3640	6/16/2022	-3.965145	-1.715625	0.611284	-8.207972
3641	6/17/2022	1.153314	-0.352642	-0.720077	4.312824

3642 rows × 5 columns

In [ ]:
```
mu1 = round(daily_returns_df['UAL'].mean(), 2) # mean of distribution
sigma1 = round(daily_returns_df['UAL'].std(), 2) # standard deviation of distribution

mu2 = round(daily_returns_df['PG'].mean(), 2) # mean of distribution
sigma2 = round(daily_returns_df['PG'].std(), 2) # standard deviation of distribution

num_bins = 40

plt.figure(figsize = (20, 15))

daily_returns_df[['UAL','PG']].plot.hist(bins = num_bins, color = ('b','r'), alpha = 0.7)

plt.grid()
plt.ylabel('Probability')
plt.title('Histogram for UAL: mu=' + str(mu1) + ', sigma=' + str(sigma1) + 'Histogram for PG: mu=' + str(mu2) + ', sigma=' + str(sigma2))
```

Out[ ]:
```
Text(0.5, 1.0, 'Histogram for UAL: mu=0.1, sigma=4.36Histogram for PG: mu=0.04, sigma=1.2')
<Figure size 2000x1500 with 0 Axes>
```



In [ ]:
```
daily_prices_df
```

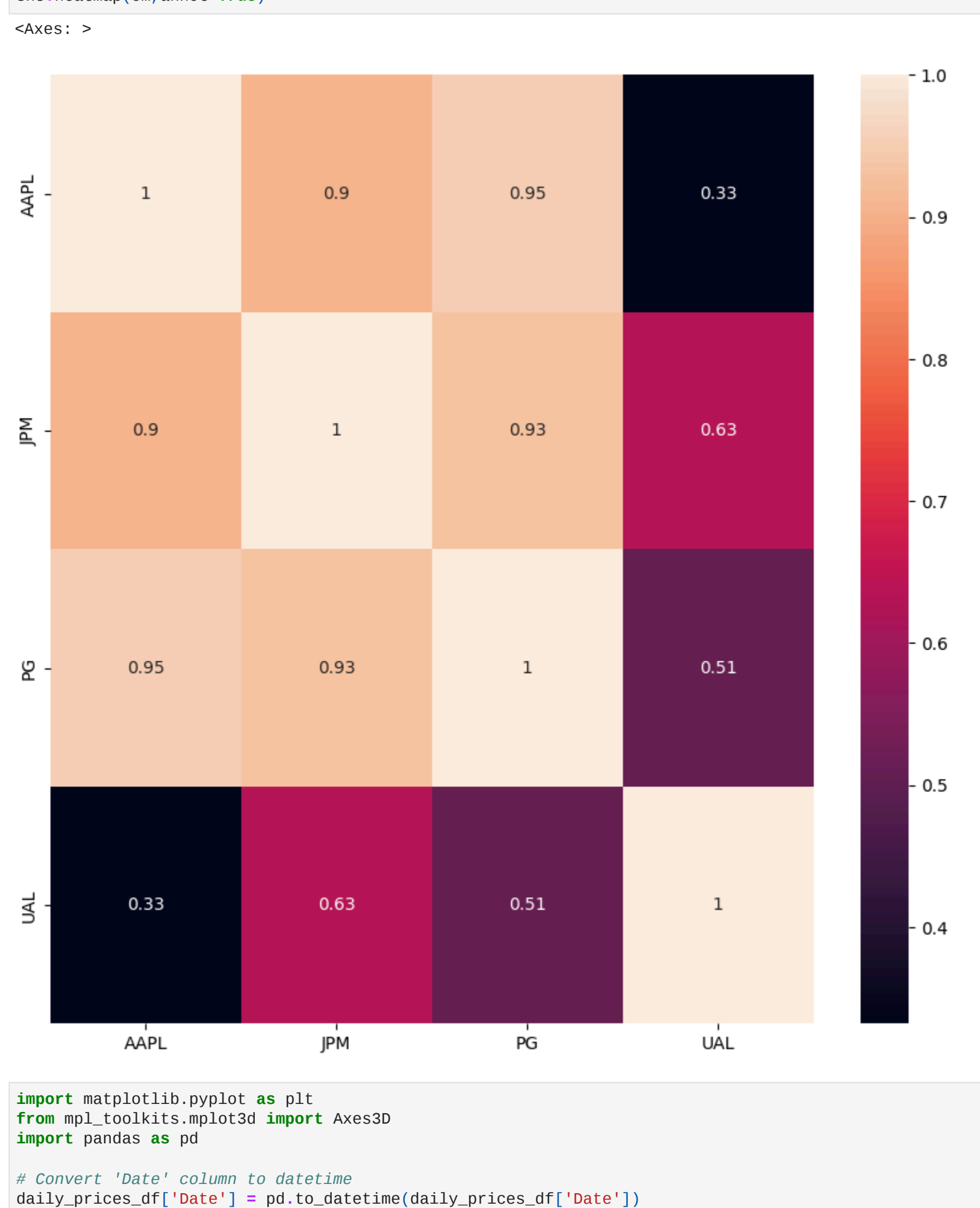
Out[ ]:

	AAPL	JPM	PG	UAL
0	5.949703	29.448614	47.058838	29.915234
1	5.952452	29.246094	47.058838	29.690800
2	5.498071	28.582682	46.870098	31.000002
3	5.424478	28.868999	47.175980	29.180000
4	5.229351	27.723747	47.299629	24.389999
...	...	...	...	...
3637	131.880005	115.989998	138.149994	37.020000
3638	132.759995	114.059998	133.839996	36.990002
3639	135.429993	115.410004	132.509995	37.889999
3640	130.059998	113.430000	133.320007	34.779999
3641	131.559998	113.029999	132.360001	36.279999

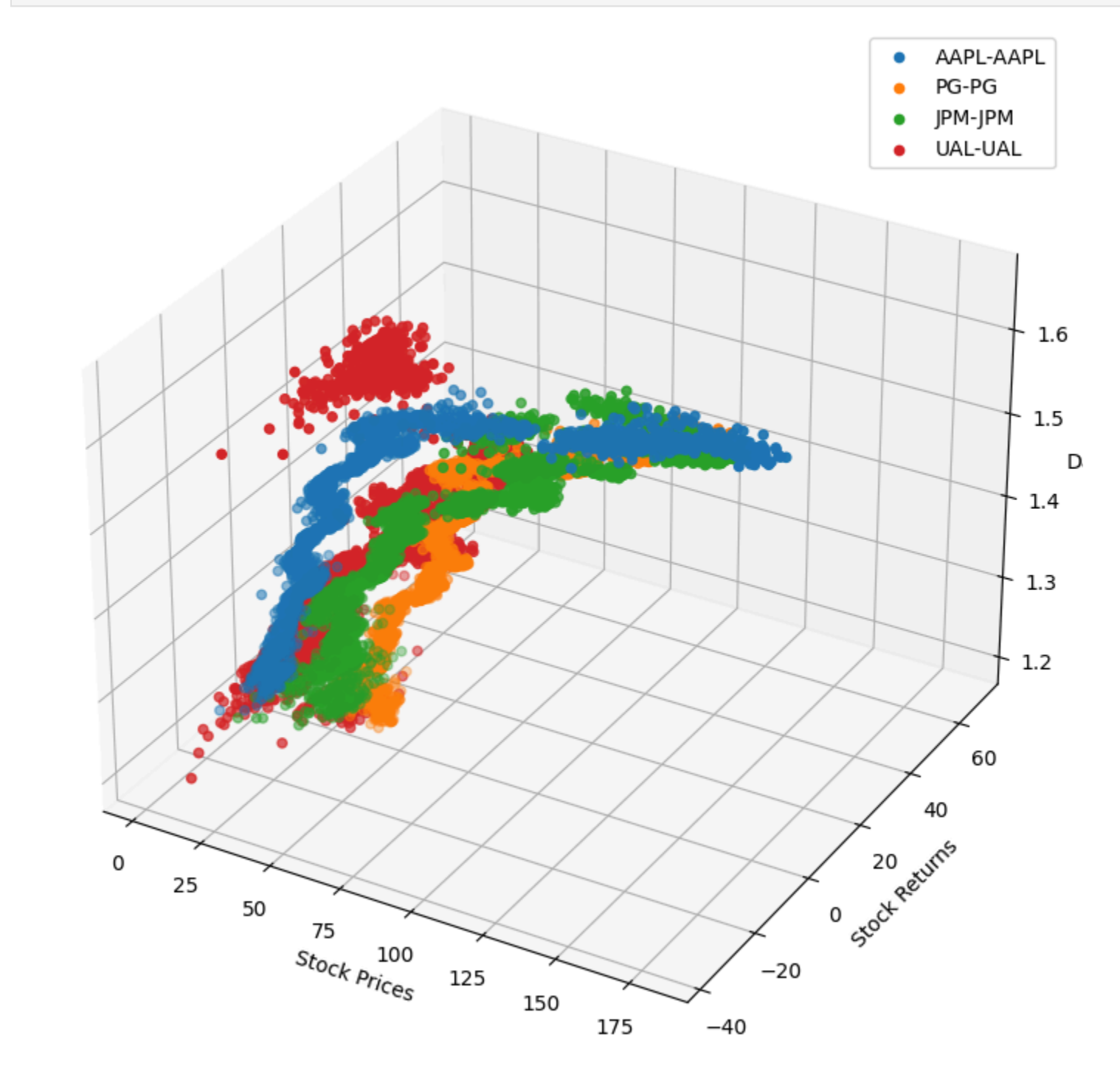
3642 rows × 4 columns

In [ ]:
```
plt.figure(figsize=(10,10))
# deletedate = daily_prices_df.drop('Date',axis='columns', inplace=True)
cm= daily_prices_df.corr()
sns.heatmap(cm,annot=True)
```

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



In [ ]:
```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import pandas as pd

# Convert 'Date' column to datetime
daily_prices_df['Date'] = pd.to_datetime(daily_prices_df['Date'])

fig = plt.figure(figsize=(20, 15))
ax1 = fig.add_subplot(121, projection='3d')

# Convert 'Date' to numerical representation
z = pd.to_numeric(daily_prices_df['Date'])
x_columns = ['AAPL', 'PG', 'JPM', 'UAL']
y_columns = ['AAPL', 'PG', 'JPM', 'UAL']

for x_col, y_col in zip(x_columns, y_columns):
    x = daily_prices_df[x_col].astype(float)  # Ensure data is numerical
    y = daily_returns_df[y_col].astype(float)  # Ensure data is numerical
    ax1.scatter(x, y, zs=z, label=f'{x_col}-{y_col}')

ax1.set_xlabel('Stock Prices')
ax1.set_ylabel('Stock Returns')
ax1.set_zlabel('Date')
ax1.legend()

plt.show()
```



In [ ]:
```
daily_returns_df[['AAPL', 'PG', 'JPM', 'UAL']]
```

Out[ ]:

	AAPL	PG	JPM	UAL
0	0.000000	0.000000	0.000000	0.000000
1	0.046203	0.000000	-0.687708	-0.750233
2	-7.633517	-0.401072	-2.268378	4.409454
3	-1.338518	0.652615	1.001718	-5.870973
4	-3.597157	0.262103	-3.967066	-16.415356
...	...	...	...	...
3637	-3.828484	-2.677001	-2.977838	-10.058308
3638	0.667265	-3.119796	-1.663937	-0.081034
3639	2.011147	-0.993725	1.183593	2.433084
3640	-3.965145	0.611284	-1.715625	-8.207972
3641	1.153314	-0.720077	-0.352642	4.312824

3642 rows × 4 columns

In [ ]: