Paige Rosynek

CS 3860 041

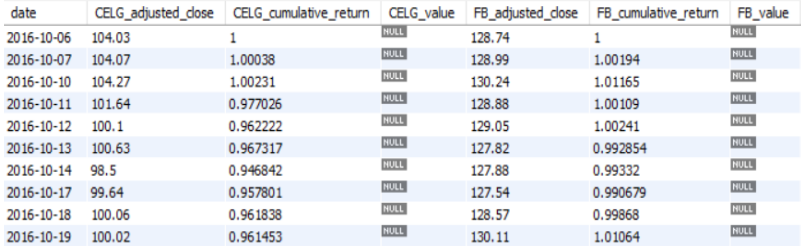
Dr. Magana

10.20.2022

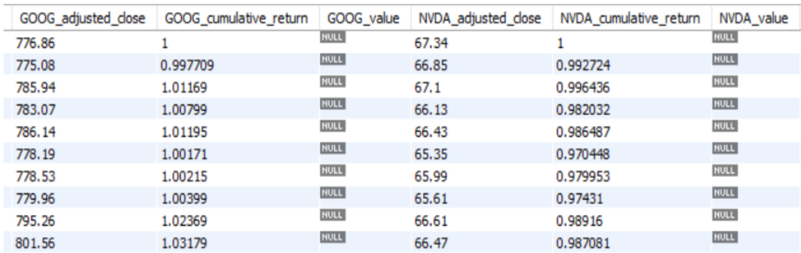
Lab 5 – Problem Solving with Data Analysis

Step 1: Portfolio Table

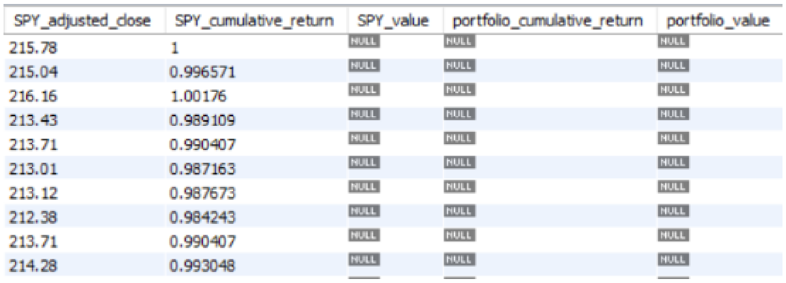
The screenshot below shows the first 10 rows of the portfolio table; there are a total of 250 rows in the table. The SQL queries used to calculate the cumulative return for each stock is included at the top of the following page; I hard coded the start date for all the calculations for simplicity. Each of the UPDATE queries has two aliases for the portfolio table to avoid RW errors. Note: the screenshots below are of the same table (portfolio) but I included separate screenshots for readability.



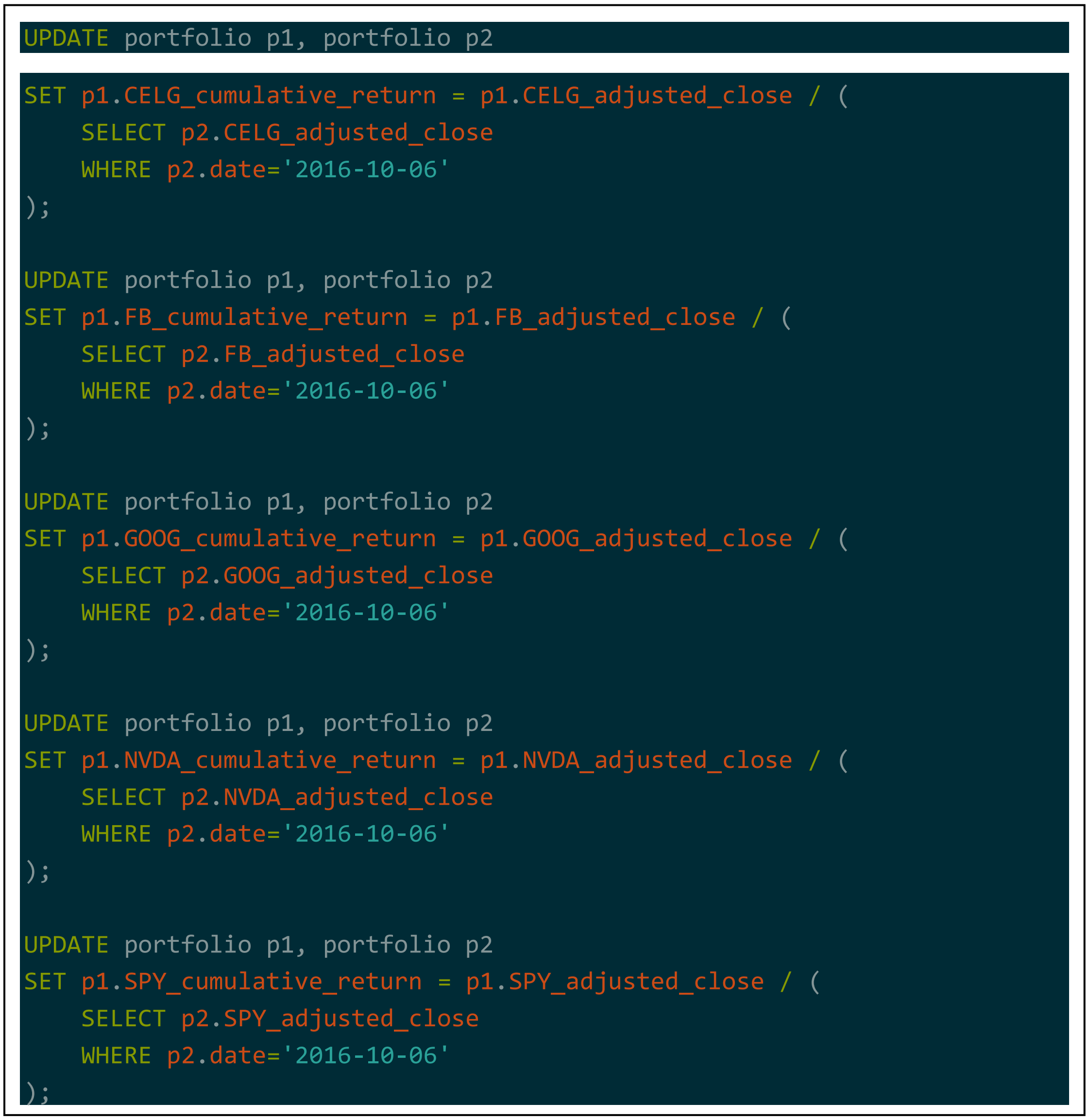
date	CELG_adjusted_close	CELG_cumulative_return	CELG_value	FB_adjusted_close	FB_cumulative_return	FB_value
2016-10-06	104.03	1	NULL	128.74	1	NULL
2016-10-07	104.07	1.00038	NULL	128.99	1.00194	NULL
2016-10-10	104.27	1.00231	NULL	130.24	1.01165	NULL
2016-10-11	101.64	0.977026	NULL	128.88	1.00109	NULL
2016-10-12	100.1	0.962222	NULL	129.05	1.00241	NULL
2016-10-13	100.63	0.967317	NULL	127.82	0.992854	NULL
2016-10-14	98.5	0.946842	NULL	127.88	0.99332	NULL
2016-10-17	99.64	0.957801	NULL	127.54	0.990679	NULL
2016-10-18	100.06	0.961838	NULL	128.57	0.99868	NULL
2016-10-19	100.02	0.961453	NULL	130.11	1.01064	NULL



GOOG_adjusted_close	GOOG_cumulative_return	GOOG_value	NVDA_adjusted_close	NVDA_cumulative_return	NVDA_value
776.86	1	NULL	67.34	1	NULL
775.08	0.997709	NULL	66.85	0.992724	NULL
785.94	1.01169	NULL	67.1	0.996436	NULL
783.07	1.00799	NULL	66.13	0.982032	NULL
786.14	1.01195	NULL	66.43	0.986487	NULL
778.19	1.00171	NULL	65.35	0.970448	NULL
778.53	1.00215	NULL	65.99	0.979953	NULL
779.96	1.00399	NULL	65.61	0.97431	NULL
795.26	1.02369	NULL	66.61	0.98916	NULL
801.56	1.03179	NULL	66.47	0.987081	NULL



SPY_adjusted_close	SPY_cumulative_return	SPY_value	portfolio_cumulative_return	portfolio_value
215.78	1	NULL	NULL	NULL
215.04	0.996571	NULL	NULL	NULL
216.16	1.00176	NULL	NULL	NULL
213.43	0.989109	NULL	NULL	NULL
213.71	0.990407	NULL	NULL	NULL
213.01	0.987163	NULL	NULL	NULL
213.12	0.987673	NULL	NULL	NULL
212.38	0.984243	NULL	NULL	NULL
213.71	0.990407	NULL	NULL	NULL
214.28	0.993048	NULL	NULL	NULL



```
UPDATE portfolio p1, portfolio p2
SET p1.CELG_cumulative_return = p1.CELG_adjusted_close / (
    SELECT p2.CELG_adjusted_close
    WHERE p2.date='2016-10-06'
);

UPDATE portfolio p1, portfolio p2
SET p1.FB_cumulative_return = p1.FB_adjusted_close / (
    SELECT p2.FB_adjusted_close
    WHERE p2.date='2016-10-06'
);

UPDATE portfolio p1, portfolio p2
SET p1.GOOG_cumulative_return = p1.GOOG_adjusted_close / (
    SELECT p2.GOOG_adjusted_close
    WHERE p2.date='2016-10-06'
);

UPDATE portfolio p1, portfolio p2
SET p1.NVDA_cumulative_return = p1.NVDA_adjusted_close / (
    SELECT p2.NVDA_adjusted_close
    WHERE p2.date='2016-10-06'
);

UPDATE portfolio p1, portfolio p2
SET p1.SPY_cumulative_return = p1.SPY_adjusted_close / (
    SELECT p2.SPY_adjusted_close
    WHERE p2.date='2016-10-06'
);
```

Step 2: Write a portfolio simulation function

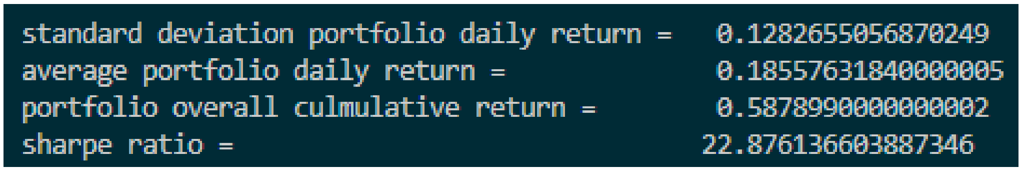
The code for the simulation function is included in the **lab05_rosynek.py** file. The simulation function loads the entire portfolio table into a pandas Dataframe to allow for calculations to be made outside of SQL. It also allows for all changes to be committed all at once to update the table using less commits.

Simulation 1

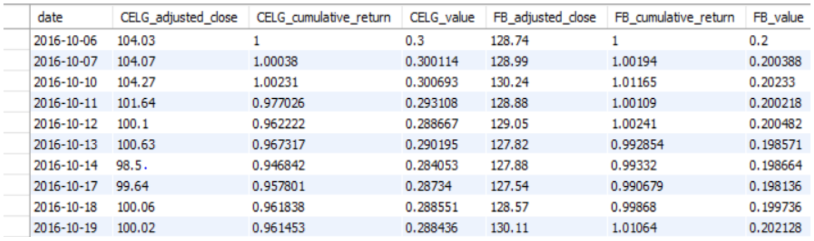
Input:

```
Stocks = ['GOOG', 'CELG', 'NVDA', 'FB']

Allocation = [0.3, 0.3, 0.2, 0.2]
```

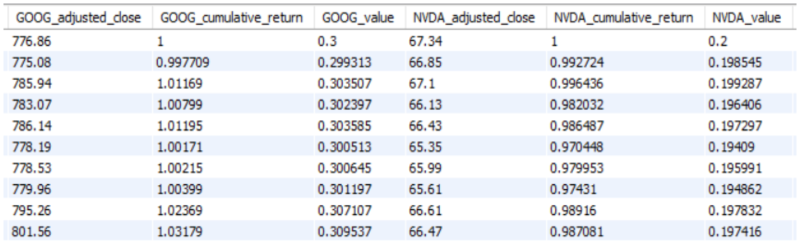
Output: full table in ./lab5 results/weights03_03_02_02.csv



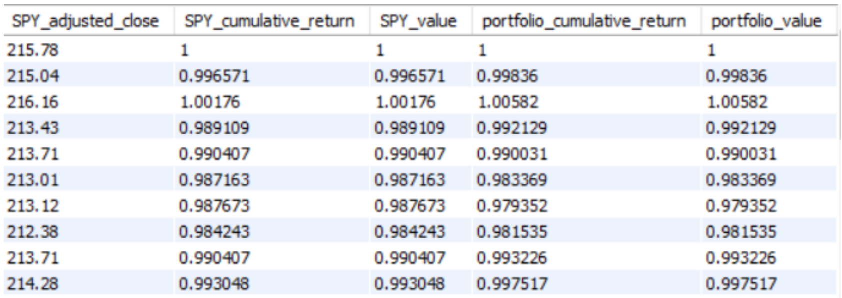
```
standard deviation portfolio daily return =   0.1282655056870249
average portfolio daily return =              0.18557631840000005
portfolio overall culmulative return =        0.5878990000000002
sharpe ratio =                               22.876136603887346
```



date	CELG_adjusted_close	CELG_cumulative_return	CELG_value	FB_adjusted_close	FB_cumulative_return	FB_value
2016-10-06	104.03	1	0.3	128.74	1	0.2
2016-10-07	104.07	1.00038	0.300114	128.99	1.00194	0.200388
2016-10-10	104.27	1.00231	0.300693	130.24	1.01165	0.20233
2016-10-11	101.64	0.977026	0.293108	128.88	1.00109	0.200218
2016-10-12	100.1	0.962222	0.288667	129.05	1.00241	0.200482
2016-10-13	100.63	0.967317	0.290195	127.82	0.992854	0.198571
2016-10-14	98.5.	0.946842	0.284053	127.88	0.99332	0.198664
2016-10-17	99.64	0.957801	0.28734	127.54	0.990679	0.198136
2016-10-18	100.06	0.961838	0.288551	128.57	0.99868	0.199736
2016-10-19	100.02	0.961453	0.288436	130.11	1.01064	0.202128



GOOG_adjusted_close	GOOG_cumulative_return	GOOG_value	NVDA_adjusted_close	NVDA_cumulative_return	NVDA_value
776.86	1	0.3	67.34	1	0.2
775.08	0.997709	0.299313	66.85	0.992724	0.198545
785.94	1.01169	0.303507	67.1	0.996436	0.199287
783.07	1.00799	0.302397	66.13	0.982032	0.196406
786.14	1.01195	0.303585	66.43	0.986487	0.197297
778.19	1.00171	0.300513	65.35	0.970448	0.19409
778.53	1.00215	0.300645	65.99	0.979953	0.195991
779.96	1.00399	0.301197	65.61	0.97431	0.194862
795.26	1.02369	0.307107	66.61	0.98916	0.197832
801.56	1.03179	0.309537	66.47	0.987081	0.197416



SPY_adjusted_close	SPY_cumulative_return	SPY_value	portfolio_cumulative_return	portfolio_value
215.78	1	1	1	1
215.04	0.996571	0.996571	0.99836	0.99836
216.16	1.00176	1.00176	1.00582	1.00582
213.43	0.989109	0.989109	0.992129	0.992129
213.71	0.990407	0.990407	0.990031	0.990031
213.01	0.987163	0.987163	0.983369	0.983369
213.12	0.987673	0.987673	0.979352	0.979352
212.38	0.984243	0.984243	0.981535	0.981535
213.71	0.990407	0.990407	0.993226	0.993226
214.28	0.993048	0.993048	0.997517	0.997517

The first 10 rows and calculation results of the simulation function, using the inputs above, are shown above. It can be observed from the screenshots of the portfolio table, that the value of each of the stocks on the first day is equal to each of their respective weights because the cumulative return for the first date is always 1 and the assumed first-day value of the portfolio is $1. Based on the equation for calculating a stock's daily value: weight of stock × portfolio value × daily cumulative return for stock, it can be observed that as the daily cumulative return for a stock decreases, the value also decreases. This is shown in the portfolio table above. The value for the portfolio reflects the cumulative return of the portfolio and the value of SPY reflects the cumulative return of SPY. The sharpe ratio calculated for the entire portfolio was approximately **22.876**. This means that with a portfolio with 30% of assets in GOOG, 30% in CELG, 20% in NVDA, and 20% in FB the ratio of the measure of excess return per unit of deviation relative to SPY is 22.876 dollars. The results of the simulation function produced the following values:

STD of portfolio daily return = 0.1283

AVG of portfolio daily return = 0.1856

Overall portfolio cumulative return = 0.5879

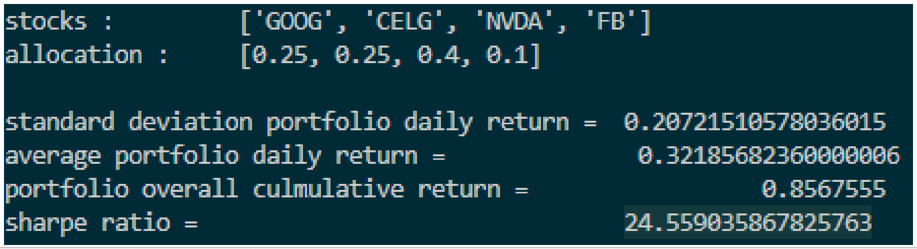
Sharpe ratio = 22.8761

<u>Simulation 2: Find a better sharpe ratio</u>

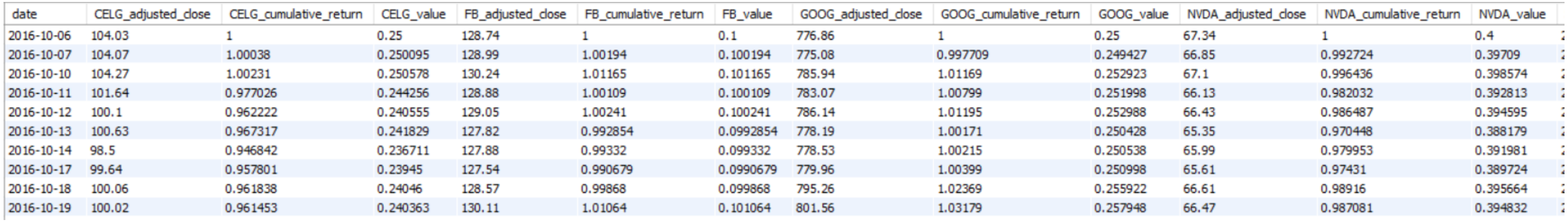
Weights tried for stocks = ['GOOG', 'CELG', 'NVDA', 'FB']

```
stock_allocation = [0.25, 0.25, 0.4, 0.1]

sharpe ratio = 24.559035867825763
```

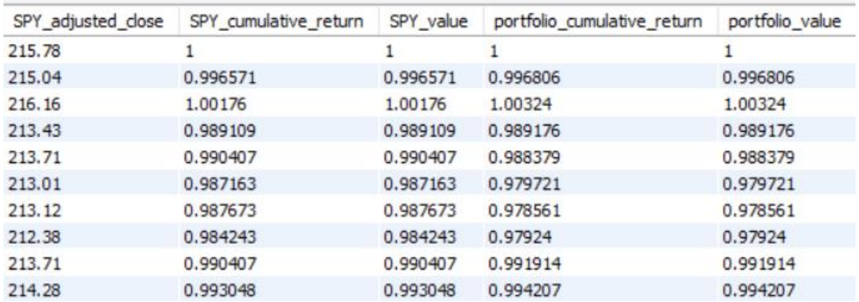
Results: full table in ./lab5 results/weights025_025_04_01.csv



```
stocks :        ['GOOG', 'CELG', 'NVDA', 'FB']
allocation :    [0.25, 0.25, 0.4, 0.1]

standard deviation portfolio daily return =  0.20721510578036015
average portfolio daily return =              0.32185682360000006
portfolio overall culmulative return =                 0.8567555
sharpe ratio =                               24.559035867825763
```



date	CELG_adjusted_close	CELG_cumulative_return	CELG_value	FB_adjusted_close	FB_cumulative_return	FB_value	GOOG_adjusted_close	GOOG_cumulative_return	GOOG_value	NVDA_adjusted_close	NVDA_cumulative_return	NVDA_value
2016-10-06	104.03	1	0.25	128.74	1	0.1	776.86	1	0.25	67.34	1	0.4
2016-10-07	104.07	1.00038	0.250095	128.99	1.00194	0.100194	775.08	0.997709	0.249427	66.85	0.992724	0.39709
2016-10-10	104.27	1.00231	0.250578	130.24	1.01165	0.101165	785.94	1.01169	0.252923	67.1	0.996436	0.398574
2016-10-11	101.64	0.977026	0.244256	128.88	1.00109	0.100109	783.07	1.00799	0.251998	66.13	0.982032	0.392813
2016-10-12	100.1	0.962222	0.240555	129.05	1.00241	0.100241	786.14	1.01195	0.252988	66.43	0.986487	0.394595
2016-10-13	100.63	0.967317	0.241829	127.82	0.992854	0.0992854	778.19	1.00171	0.250428	65.35	0.970448	0.388179
2016-10-14	98.5	0.946842	0.236711	127.88	0.99332	0.099332	778.53	1.00215	0.250538	65.99	0.979953	0.391981
2016-10-17	99.64	0.957801	0.23945	127.54	0.990679	0.0990679	779.96	1.00399	0.250998	65.61	0.97431	0.389724
2016-10-18	100.06	0.961838	0.24046	128.57	0.99868	0.099868	795.26	1.02369	0.255922	66.61	0.98916	0.395664
2016-10-19	100.02	0.961453	0.240363	130.11	1.01064	0.101064	801.56	1.03179	0.257948	66.47	0.987081	0.394832



SPY_adjusted_close	SPY_cumulative_return	SPY_value	portfolio_cumulative_return	portfolio_value
215.78	1	1	1	1
215.04	0.996571	0.996571	0.996806	0.996806
216.16	1.00176	1.00176	1.00324	1.00324
213.43	0.989109	0.989109	0.989176	0.989176
213.71	0.990407	0.990407	0.988379	0.988379
213.01	0.987163	0.987163	0.979721	0.979721
213.12	0.987673	0.987673	0.978561	0.978561
212.38	0.984243	0.984243	0.97924	0.97924
213.71	0.990407	0.990407	0.991914	0.991914
214.28	0.993048	0.993048	0.994207	0.994207

In order to find a set of stock allocation that produced a better sharpe value, I arbitrarily chose weights. The first set of weights I tried was [0.25, 0.25, 0.4, 0.1], for the stocks: ['GOOG', 'CELG', 'NVDA', 'FB'], which produced a sharpe ratio of **24.5590** which is greater than the sharpe ratio from the previous section. Therefore, these weights produced a higher return with less risk than the original weights. The results of the simulation function with the defined weights produced the following values:

STD of portfolio daily return = 0.2072

AVG of portfolio daily return = 0.3219

Overall portfolio cumulative return = 0.8568

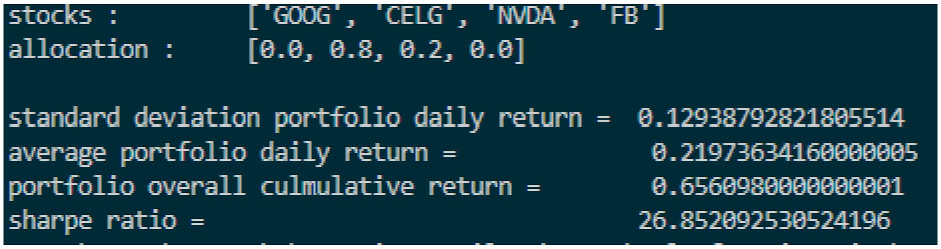
Sharpe ratio = 24.5590

Step 3: Optimal sharpe ratio & stock weights

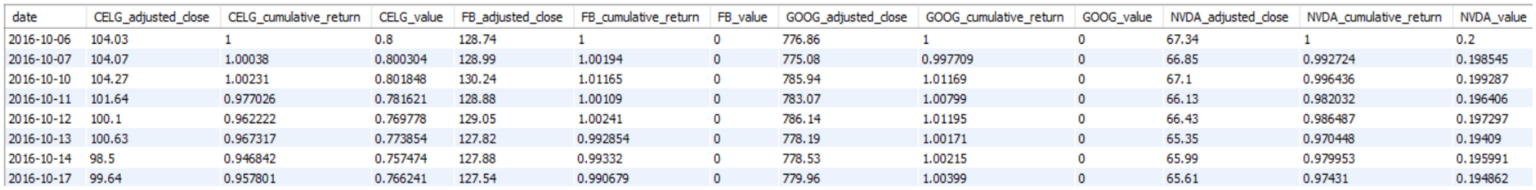
Optimal stock allocation:

```
Stocks = ['GOOG', 'CELG', 'NVDA', 'FB']

allocation = [0.0, 0.8, 0.2, 0.0]
```

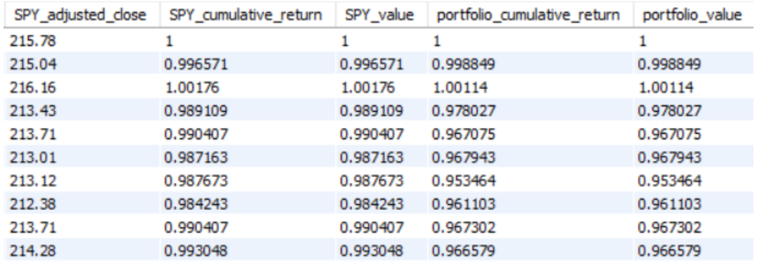
Output: full table in ./lab5 results/max_sharpe



```
stocks :          ['GOOG', 'CELG', 'NVDA', 'FB']
allocation :      [0.0, 0.8, 0.2, 0.0]

standard deviation portfolio daily return =  0.12938792821805514
average portfolio daily return =              0.21973634160000005
portfolio overall culmulative return =        0.6560980000000001
sharpe ratio =                               26.852092530524196
```



date	CELG_adjusted_close	CELG_cumulative_return	CELG_value	FB_adjusted_close	FB_cumulative_return	FB_value	GOOG_adjusted_close	GOOG_cumulative_return	GOOG_value	NVDA_adjusted_close	NVDA_cumulative_return	NVDA_value
2016-10-06	104.03	1	0.8	128.74	1	0	776.86	1	0	67.34	1	0.2
2016-10-07	104.07	1.00038	0.800304	128.99	1.00194	0	775.08	0.997709	0	66.85	0.992724	0.198545
2016-10-10	104.27	1.00231	0.801848	130.24	1.01165	0	785.94	1.01169	0	67.1	0.996436	0.199287
2016-10-11	101.64	0.977026	0.781621	128.88	1.00109	0	783.07	1.00799	0	66.13	0.982032	0.196406
2016-10-12	100.1	0.962222	0.769778	129.05	1.00241	0	786.14	1.01195	0	66.43	0.986487	0.197297
2016-10-13	100.63	0.967317	0.773854	127.82	0.992854	0	778.19	1.00171	0	65.35	0.970448	0.19409
2016-10-14	98.5	0.946842	0.757474	127.88	0.99332	0	778.53	1.00215	0	65.99	0.979953	0.195991
2016-10-17	99.64	0.957801	0.766241	127.54	0.990679	0	779.96	1.00399	0	65.61	0.97431	0.194862



SPY_adjusted_close	SPY_cumulative_return	SPY_value	portfolio_cumulative_return	portfolio_value
215.78	1	1	1	1
215.04	0.996571	0.996571	0.998849	0.998849
216.16	1.00176	1.00176	1.00114	1.00114
213.43	0.989109	0.989109	0.978027	0.978027
213.71	0.990407	0.990407	0.967075	0.967075
213.01	0.987163	0.987163	0.967943	0.967943
213.12	0.987673	0.987673	0.953464	0.953464
212.38	0.984243	0.984243	0.961103	0.961103
213.71	0.990407	0.990407	0.967302	0.967302
214.28	0.993048	0.993048	0.966579	0.966579

In order to find the set of stock allocations that maximizes the sharpe ratio of the portfolio, I looped through every permutation of the values from 0-1 with a step of 0.1 and stored the weights that summed to zero. Then for each of the test weights I ran the simulation function to find the optimal set of weights. The optimal weights were `[0.0, 0.8, 0.2, 0.0]` for `['GOOG', 'CELG', 'NVDA', 'FB']` and produced a maximum sharpe ratio of **26.8521**. Since the sharpe value is maximized, the following values are also maximized:

STD of portfolio daily return = 0.1294

AVG of portfolio daily return = 0.2197

Overall portfolio cumulative return = 0.6561

Sharpe ratio = 26.8521