In [2]:

```
import numpy as np
import pandas as pd
import yfinance as yf
```

MFE 409, Risk; HW 3

Aliaksei Kanstantsinau

Part 1. Problem 1.

Morgan Stanley employs various techniques to compute Value at Risk, reflecting its comprehensive approach to managing market risk. Those techniques employ a combination of historical simulation and Monte Carlo simulation to calculate VaR. The historical simulation involves analyzing daily changes in market indices and other factors, while the Monte Carlo simulation is used to assess name-specific risks in equities and fixed income exposures.
Morgan Stanley employs approximately four years of historical market data to evaluate potential changes in market risk factors with the primary time horizon used in VaR being one day, reflecting the standard practice for trading portfolios. Calculations are also normally based on 95% condidence level.
During fiscal 2007, which might offer a parallel to 2008, Morgan Stanley experienced 15 days where losses exceeded the VaR estimates. These exceptions occurred during periods of unusually high market volatility, particularly in equity, corporate credit, and securitized product markets. Following the heightened market volatility in 2007, Morgan Stanley reviewed and adjusted its VaR models to enhance the accuracy of risk estimations, especially for certain fixed income products. This included broader product coverage and updated mappings of risk exposures to historical price time series.

Problem 2.

Download the daily stock price for the corresponding bank over 2006-2008 from Yahoo finance.
(a) On each day of 2008, compute the 99% 1-day VaR for the stock return using the historical method with all past data in the sample.

In [33]:

```
# download morgan stanley data
ms = yf.download('MS', start='2006-01-01', end='2008-12-31')
```

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

In [35]:

```
# add a column for returns
ms['Return'] = ms['Close'].pct_change()
```

In [36]:

```
# define get var function
def get_var(returns):
    # sort values
    sorted_returns = returns.sort_values()
    # get var
    var = sorted_returns.quantile(.01)
    # return var
    return var
```

In [37]:

```
# initialize nans
ms['VaR'] = float('nan')
returns = ms['Return'].dropna()

for date in ms['2008'].index:
    # get var using data up to current date
    current_var = get_var(returns[:date])
    ms.at[date, 'VaR'] = current_var

ms2008 = ms.loc['2008']
ms2008.head(-10)
```

```
/var/folders/vv/3nnd1g4506z6vdqnf44fkr2c0000gn/T/ipykernel_33525/3771324352.
py:5: FutureWarning: Indexing a DataFrame with a datetimelike index using a
single string to slice the rows, like `frame[string]`, is deprecated and wil
l be removed in a future version. Use `frame.loc[string]` instead.
  for date in ms['2008'].index:
```

Out[37]:

Date	Open	High	Low	Close	Adj Close	Volume	Retur
2008-01-02	52.980000	53.400002	50.310001	50.950001	36.114250	17624100	-0.04067
2008-01-03	51.209999	51.889999	50.580002	50.939999	36.107166	11422200	-0.00019
2008-01-04	49.919998	50.689999	48.860001	49.299999	34.944695	14448500	-0.03219
2008-01-07	49.500000	49.840000	47.950001	48.310001	34.242973	18767500	-0.02008
2008-01-08	48.650002	48.970001	45.880001	46.400002	32.889130	22467500	-0.03953
...	...	...	...	...	...	...	
2008-12-09	15.580000	16.240000	14.910000	14.970000	10.948717	28424800	-0.08159
2008-12-10	15.350000	15.500000	14.010000	14.600000	10.678107	21280200	-0.02471
2008-12-11	14.240000	14.500000	13.550000	13.740000	10.049118	28285100	-0.05890
2008-12-12	13.000000	13.960000	12.650000	13.850000	10.129574	21175100	0.00800
2008-12-15	14.050000	14.300000	13.300000	13.640000	9.975984	20460200	-0.01516

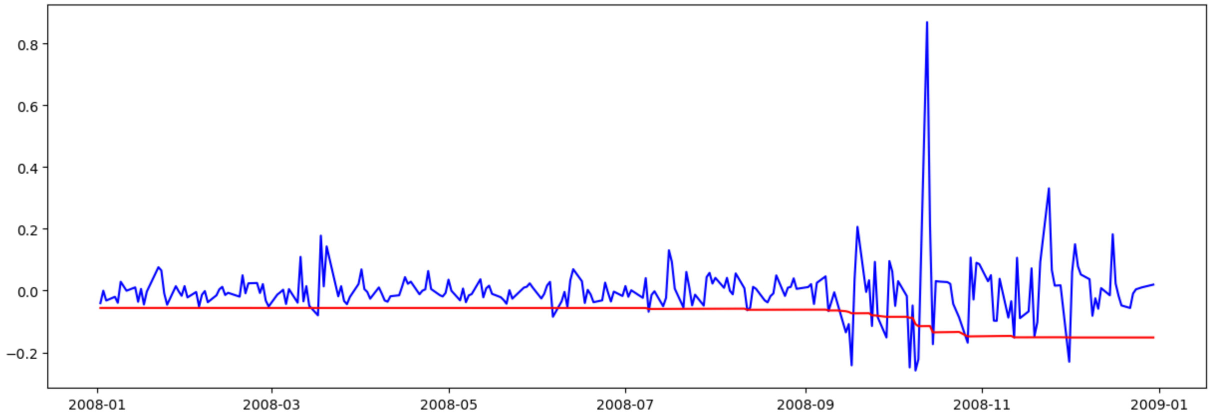
242 rows × 8 columns

(b) If you are at the end of 2008 and want to back-test this approach, what do you do and what do you conclude?

- From the plot we can most definitely conclude that losses exceeding VaR more than 5% of the time. To see how out of scope that is we can calculate probability and go from there.

In [38]:

```
# lets plot returns vs VaR to see where exceeding happend.
import matplotlib.pyplot as plt
plt.figure(figsize=(15, 5))
plt.plot(ms2008.index, ms2008['Return'], color='blue')
plt.plot(ms2008.index, ms2008['VaR'], color='red')
plt.show()
```



(c) Comment on the relation with what you found in the annual report.

- Report states there were 15 trading days with losses, however, our approach indicates 18.

In [40]:
```
losses = ms2008[ms2008['Return'] < ms2008['VaR']]
loss_days = losses.shape[0]
print('Total loss days:', loss_days)
```

```
Total loss days: 18
```

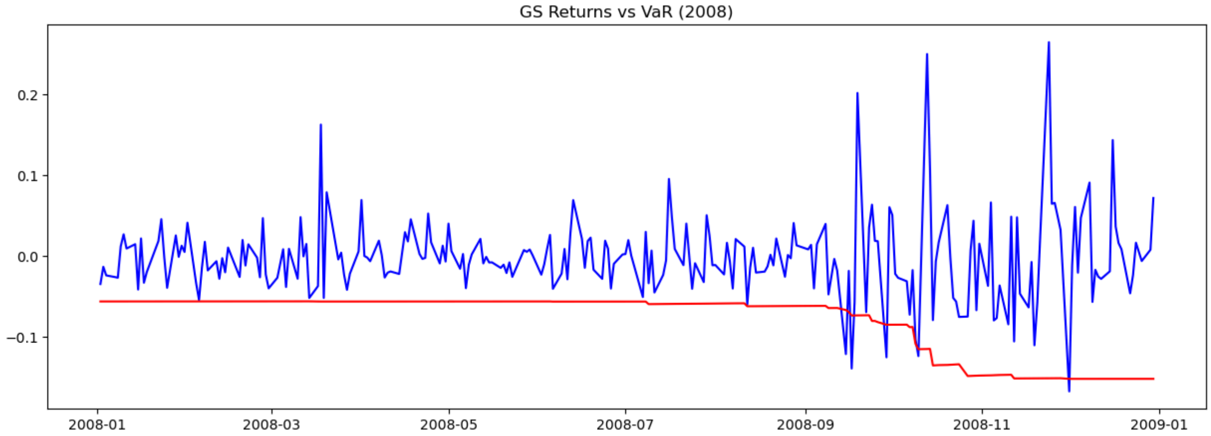
Problem 3.

Add to your dataset the daily stock price for all 10 banks over the same period.
(a) Use the historical method to compute the VaR for a portfolio with \$1m in the odd-numbered banks (1, 3, ...), $2m in the even-numbered banks.

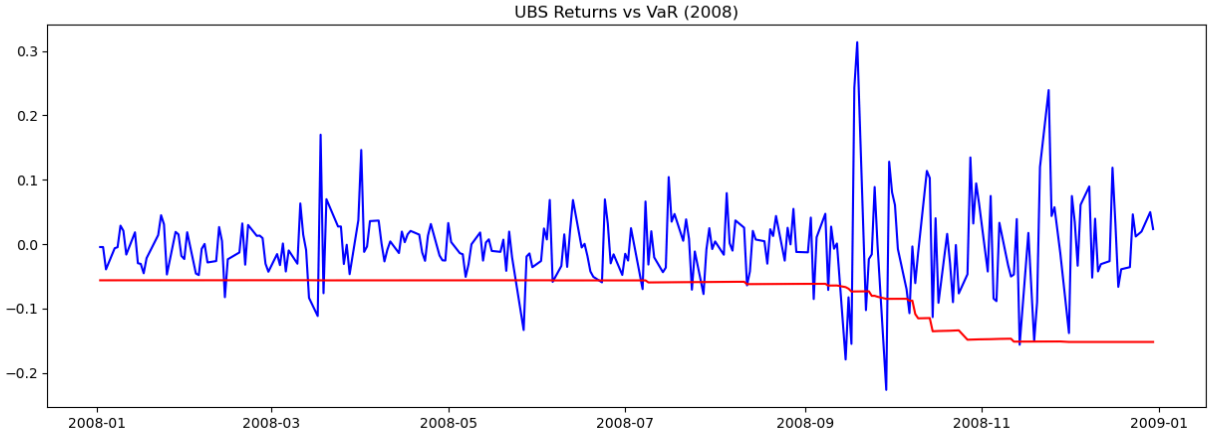
In [56]:
```
ticker_array = ['GS', 'UBS', 'JPM', 'C', 'BCS', 'MS', 'DB', 'BAC', 'BNP.PA']
```

In [61]:
```
dfs = {}

for ticker in ticker_array:
    df = yf.download(ticker, start='2006-01-01', end='2008-12-31')
    df['Return'] = df['Close'].pct_change()
    df['VaR'] = float('nan')
    returns = ms['Return'].dropna()

    for date in df.loc['2008-01-01':'2008-12-31'].index:
        current_var = get_var(returns[:date])
        df.at[date, 'VaR'] = current_var

    dfs[ticker] = df.loc['2008-01-01':'2008-12-31']
    df2008 = df.loc['2008-01-01':'2008-12-31']
    plt.figure(figsize=(15, 5))
    plt.plot(df2008.index, df2008['Return'], color='blue', label=f'{ticker}
    plt.plot(df2008.index, df2008['VaR'], color='red', label=f'{ticker} VaR'
    plt.title(f'{ticker} Returns vs VaR (2008)')
    plt.show()
```

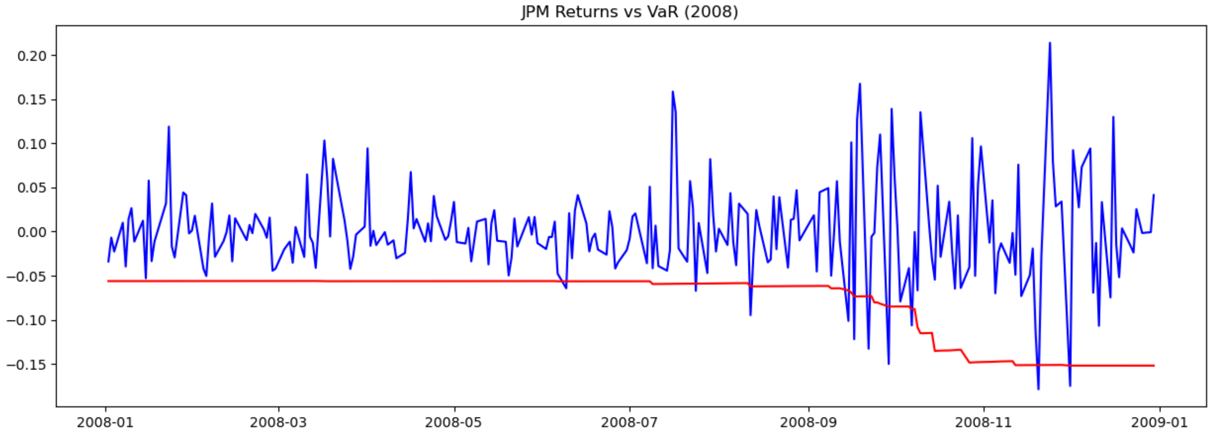
```
[*********************100%%**********************]  1 of 1 completed
```


GS Returns vs VaR (2008)

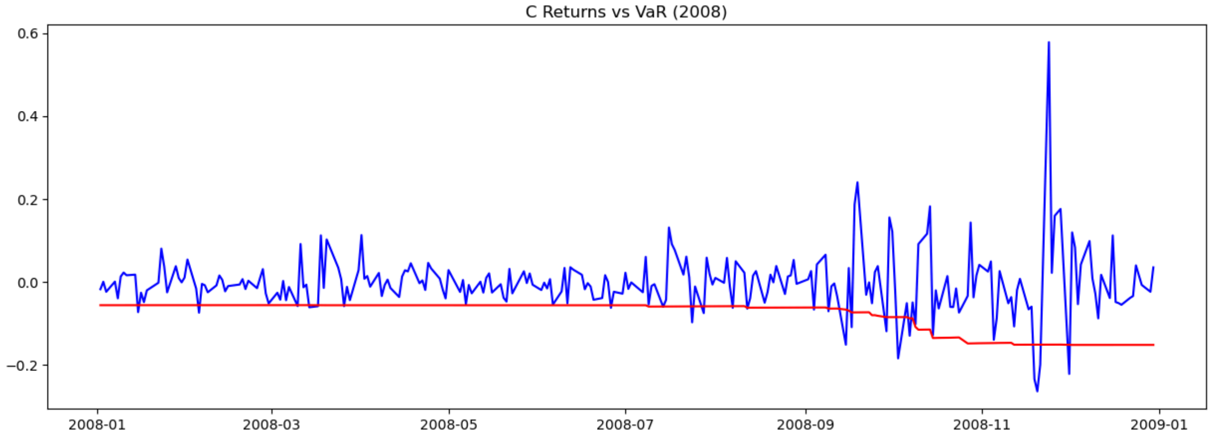
```
[*********************100%%**********************]  1 of 1 completed
```


UBS Returns vs VaR (2008)

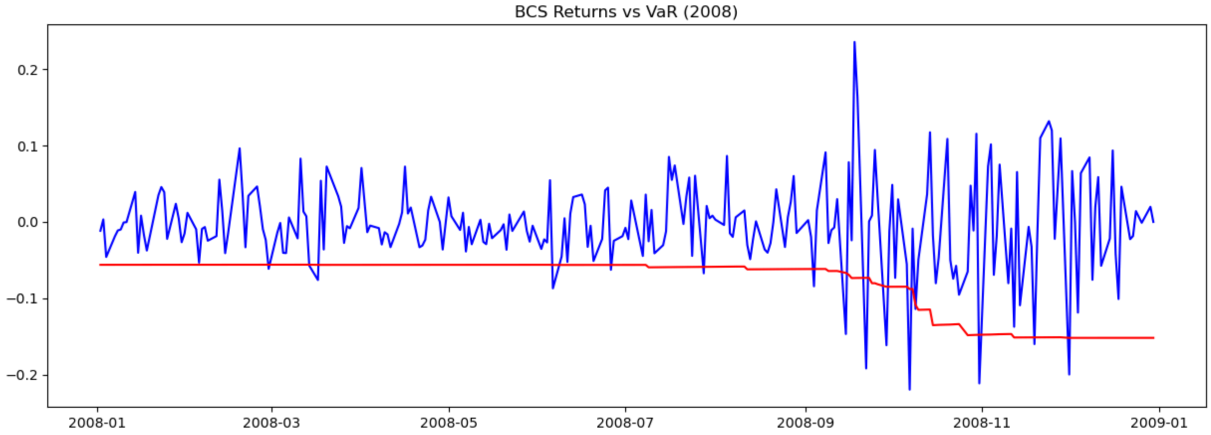
```
[*********************100%%**********************]  1 of 1 completed
```


JPM Returns vs VaR (2008)

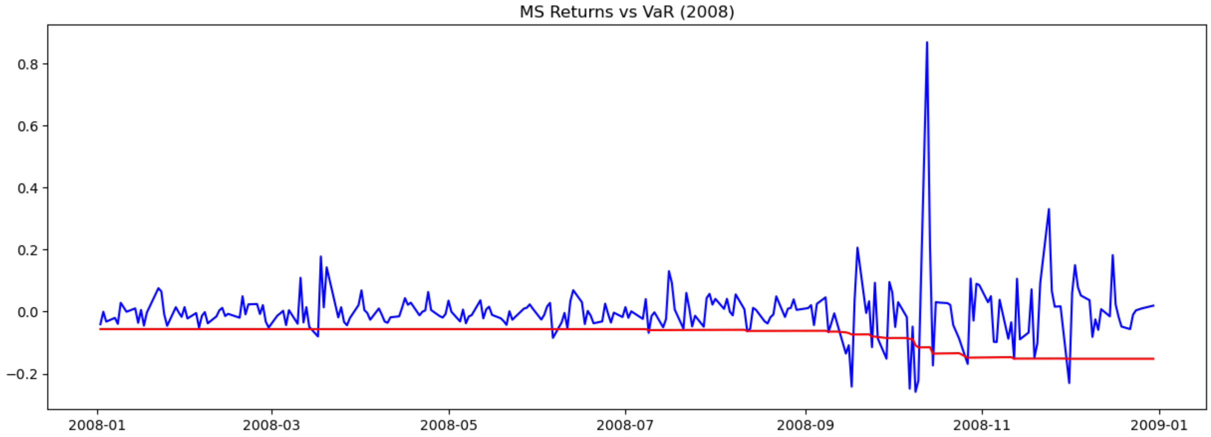
```
[*********************100%%**********************]  1 of 1 completed
```



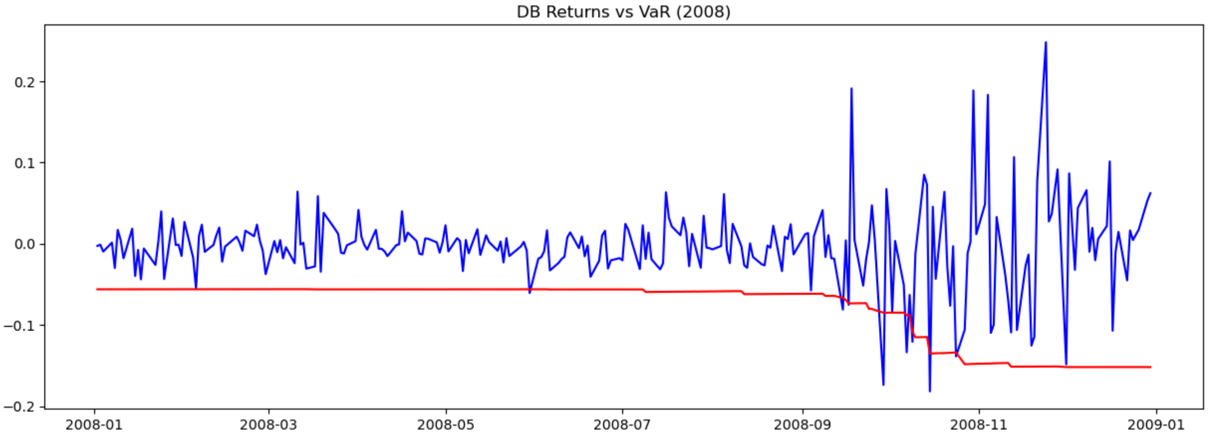
```
[*********************100%%**********************]  1 of 1 completed
```



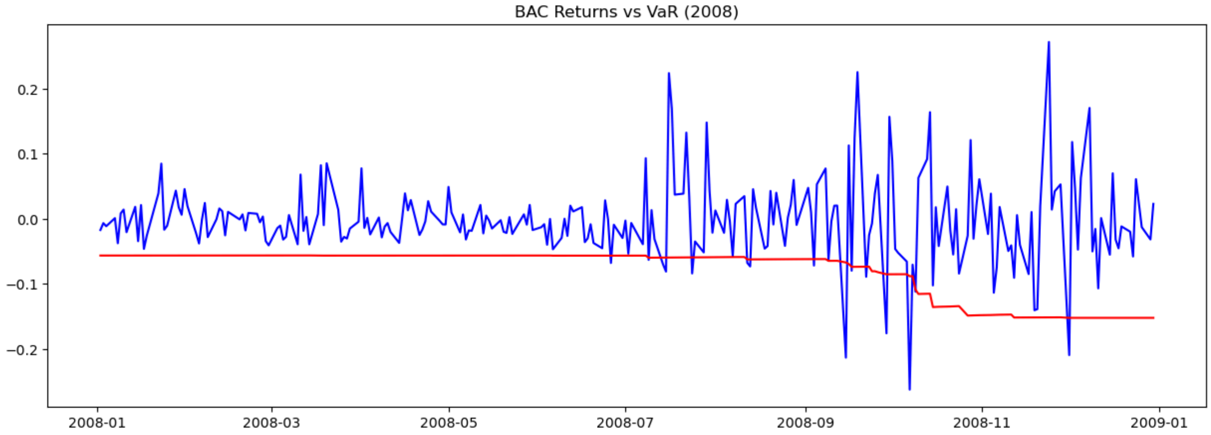
```
[*********************100%%**********************]  1 of 1 completed
```



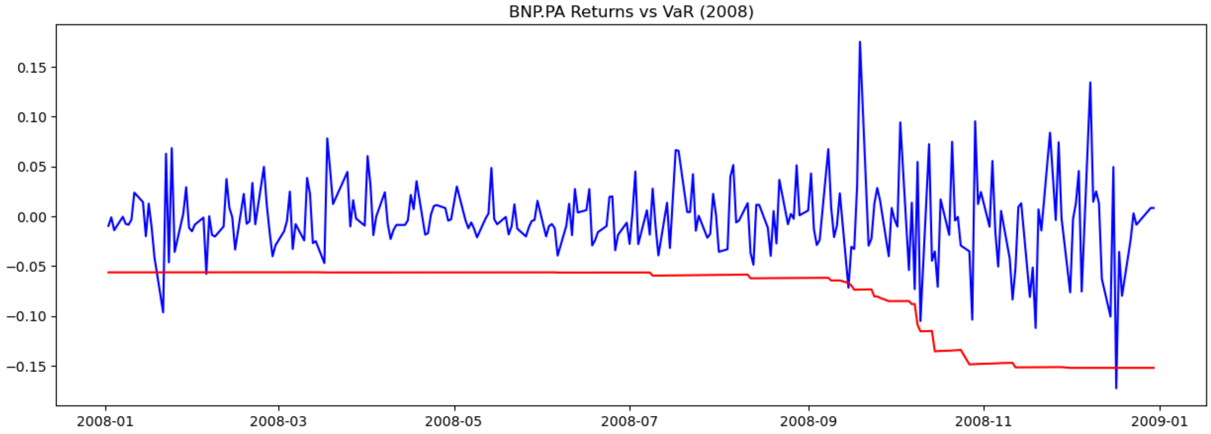
```
[*********************100%%**********************]  1 of 1 completed
```



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



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



In [59]:

```
# initiate new df for vars and set values to 0
portfolio_var = pd.DataFrame(index=dfs[next(iter(dfs))].index)
portfolio_var['Portfolio VaR'] = 0.0
# iterate through dataframes
for i, ticker in enumerate(ticker_array):
    position = 1e6 if i % 2 == 0 else 2e6
    df = dfs[ticker]
    portfolio_var['Portfolio VaR'] += df['VaR'] * position

print(portfolio_var)
```

```
            Portfolio VaR
Date
2008-01-02  -7.305232e+05
2008-01-03  -7.304858e+05
2008-01-04  -7.304484e+05
2008-01-07  -7.304110e+05
2008-01-08  -7.303736e+05
...                   ...
2008-12-23  -1.975419e+06
2008-12-24  -1.975410e+06
2008-12-26            NaN
2008-12-29  -1.975392e+06
2008-12-30  -1.975383e+06

[252 rows x 1 columns]
```

(b) Compute the DVaR and CVaR for each bank.

In [69]:

```python
dvars = {}
cvars = {}

for i, (ticker, df) in enumerate(dfs.items()):
    adjusted_index = i + 1
    # set position
    position = 1e6 if adjusted_index % 2 != 0 else 2e6
    # get dvar
    dvars[ticker] = df['VaR'] - (df['VaR']*(position + 1))
    # get cvar
    cvars[ticker] = df['VaR'] - (df['VaR']*(position*1.01))

print(f'DVaRs:', pd.DataFrame(dvars))
print(f'CVaRs:', pd.DataFrame(cvars))
```

```
DVaRs:                        GS            UBS            JPM
C  \
Date
2008-01-02   56194.089345  112388.178690   56194.089345  112388.178690
2008-01-03   56191.212858  112382.425715   56191.212858  112382.425715
2008-01-04   56188.336370  112376.672740   56188.336370  112376.672740
2008-01-07   56185.459883  112370.919765   56185.459883  112370.919765
2008-01-08   56182.583395  112365.166790   56182.583395  112365.166790
...                   ...            ...            ...            ...
2008-12-23  151955.320577  303910.641154  151955.320577  303910.641154
2008-12-24  151954.625973  303909.251946  151954.625973  303909.251946
2008-12-26  151953.931369  303907.862738  151953.931369  303907.862738
2008-12-29  151953.236765  303906.473530  151953.236765  303906.473530
2008-12-30  151952.542161  303905.084322  151952.542161  303905.084322

                      BCS             MS             DB            BAC  \
Date
2008-01-02   56194.089345  112388.178690   56194.089345  112388.178690
2008-01-03   56191.212858  112382.425715   56191.212858  112382.425715
2008-01-04   56188.336370  112376.672740   56188.336370  112376.672740
2008-01-07   56185.459883  112370.919765   56185.459883  112370.919765
2008-01-08   56182.583395  112365.166790   56182.583395  112365.166790
...                   ...            ...            ...            ...
2008-12-23  151955.320577  303910.641154  151955.320577  303910.641154
2008-12-24  151954.625973  303909.251946  151954.625973  303909.251946
2008-12-26  151953.931369  303907.862738  151953.931369  303907.862738
2008-12-29  151953.236765  303906.473530  151953.236765  303906.473530
2008-12-30  151952.542161  303905.084322  151952.542161  303905.084322

                   BNP.PA
Date
2008-01-02   56194.089345
2008-01-03   56191.212858
2008-01-04   56188.336370
2008-01-07   56185.459883
2008-01-08   56182.583395
...                   ...
2008-12-23  151955.320577
2008-12-24  151954.625973
2008-12-26            NaN
2008-12-29  151953.236765
2008-12-30  151952.542161

[258 rows x 9 columns]
CVaRs:                        GS            UBS            JPM
C  \
Date
2008-01-02   56755.974044  113512.004283   56755.974044  113512.004283
2008-01-03   56753.068795  113506.193781   56753.068795  113506.193781
2008-01-04   56750.163545  113500.383279   56750.163545  113500.383279
2008-01-07   56747.258296  113494.572777   56747.258296  113494.572777
2008-01-08   56744.353046  113488.762276   56744.353046  113488.762276
...                   ...            ...            ...            ...
2008-12-23  153474.721827  306949.595610  153474.721827  306949.595610
2008-12-24  153474.020278  306948.192511  153474.020278  306948.192511
2008-12-26  153473.318729  306946.789411  153473.318729  306946.789411
```

```
2008-12-29  153472.617179  306945.386312  153472.617179  306945.386312
2008-12-30  153471.915630  306943.983212  153471.915630  306943.983212

                      BCS             MS             DB            BAC  \
Date
2008-01-02   56755.974044  113512.004283   56755.974044  113512.004283
2008-01-03   56753.068795  113506.193781   56753.068795  113506.193781
2008-01-04   56750.163545  113500.383279   56750.163545  113500.383279
2008-01-07   56747.258296  113494.572777   56747.258296  113494.572777
2008-01-08   56744.353046  113488.762276   56744.353046  113488.762276
...                   ...            ...            ...            ...
2008-12-23  153474.721827  306949.595610  153474.721827  306949.595610
2008-12-24  153474.020278  306948.192511  153474.020278  306948.192511
2008-12-26  153473.318729  306946.789411  153473.318729  306946.789411
2008-12-29  153472.617179  306945.386312  153472.617179  306945.386312
2008-12-30  153471.915630  306943.983212  153471.915630  306943.983212

                   BNP.PA
Date
2008-01-02   56755.974044
2008-01-03   56753.068795
2008-01-04   56750.163545
2008-01-07   56747.258296
2008-01-08   56744.353046
...                   ...
2008-12-23  153474.721827
2008-12-24  153474.020278
2008-12-26            NaN
2008-12-29  153472.617179
2008-12-30  153471.915630

[258 rows x 9 columns]
```

(c) Comment on the results.
(d) If you had to make a recommendation on how to tilt this portfolio, what would it be based on the data you have?

- Looks like there is a mistake in my calculations but I can't figure out where.

Part 2.

Problem 1.

Prove that if 8 people are born in a three-year period, at least 3 of them are born within the same one-year period. What does it have to do with the class?

- Assume we randomly select 3 out of 8 people and they are all born in different years.

- Next, we select 3 more people at random out of the remaining 5 and they are all born in different years.

- Now, we have 2 people born in year 1, 2 in year 2, and 2 in year 3.

- We have 2 more people to select from, therefore there will be at least 3 people born within same one year period.

- I'd assume it has to deal with unlimited losses and limited number of companies.

Problem 2.

What is the ten-day 99% VaR of a portfolio with a five-day 98% VaR of $10 million?

- 98% 1 day VaR = $\frac{\$10m}{\sqrt{5}} = \$4.472m$
- 98% Z-Score 2.32, 99% Z-Score 2.57. Consider $VaR = Z(c)*\sigma_{position}$
- Then 1-day $99\% VaR = \frac{2.57}{2.32}98\%VaR = 1.108 \cdot \$4.472m = \$4.954m$
- Then 10-day $99\% VaR = \$4.954m * \sqrt{10} = \$15.666m$

Problem 3.

What is the probability of having more than one exception in the same month? Use the answer this question to come up with a test of a VaR measure based on bunching.

- Probability of having more than one exception in the same month equals to: 1 - P(seeing 0 or 1 exception)

$$1 - P((\frac{n!}{k!(n-k)!}(1-c)^kc^{n-k})+(\frac{n!}{k!(n-k)!}(1-c)^kc^{n-k}))$$

- Where k is 0 or 1, and n expected value based on c.

Problem 4.

The next regular FOMC meeting is scheduled for the end of this month. How would you estimate the 2-day 99% VaR of investing $1m in the S&P500 a day before the announcement? Bonus question: Provide a number.\

- 1-day VaR 1m * 2.57 * .16 = 411200
- Lets use higher vol 1m * 2.57 * .25 = 642500
- 2-day = 1-day * $\sqrt{2}$ = 899500

```
In [ ]:
```