CQF Exam One

June 2025 Cohort

Solution for Task-1

The portfolio in discussion comprises assets A, B, C &D with the following characteristics:

Asset	Return (μ)	Standard Deviation (σ)	Allocated Weights*
A	0.02	0.05	W1
B	0.07	0.12	W2
C	0.15	0.17	W3
D	0.20	0.25	W4

*The allocated weights in this problem refer to the weights of the assets in the tangency portfolios corresponding to risk-free rates r_f = 25bps, 125bps, 200bps

The correlation matrix for the above-specified list of assets is:

1	0.3	0.3	0.3
0.3	1	0.6	0.6
0.3	0.6	1	0.6
0.3	0.6	0.6	1

a. For calculating the tangency portfolio for each of the 3 cases corresponding to the 3 risk-free rates, the analytical solutions to the following optimization problem:

Maximize: $$\frac{w'\mu - \boldsymbol{r_f}}{\sqrt{w'\Sigma w}} = \frac{\mu_\pi - r_f}{\sigma_\pi}$$

Subject to: $$w'1 = 1$$

were obtained to determine the portfolio weights of each of the constituent assets in each case, using the Lagrange method for solving the above optimization problem, which results in the following solution for the portfolio weights:

$$w_T^* = \frac{\Sigma^{-1}(\mu - r1)}{1'\Sigma^{-1}(\mu - r1)}$$

w_T^* was calculated using the following Python function:

```python
def tagency_portfolio_characteristics(df,correl,rf):
  l=np.array(df["Returns"])
  k=np.array(df["Volatility"])
  y=np.diag(k)@correl@np.diag(k)
  inverse_matrix=np.linalg.inv(y)
  rf_arr=np.full(k.shape[0],rf)
  n_wt=inverse_matrix@(l-rf_arr)
  d_wt=np.ones(k.shape[0])@inverse_matrix@(l-rf_arr)
  wt=n_wt/d_wt
  p_sigma=np.sqrt(wt@y@wt)
  p_return=l@wt
  p_sharpe=(p_return-rf)/p_sigma
  return p_sigma,p_return,p_sharpe,wt
```

The function assumes that the input data will comprise the following:

a. df: dataframe containing the characteristics of the constituent assets in 2 different columns, titled “Returns” (for asset returns) and “Volatility”(for individual standard deviations)
b. correl: correlation data
c. rf: risk-free rate

The function returns the standard deviation, return, Sharpe ratio, and the weights of the individual assets.

The weights and portfolio variances for each of the portfolios are presented in the following table:

	Tangency Portfolio for 0.25% Risk-Free Rate	**Tangency Portfolio for 1.25% Risk-Free Rate**	**Tangency Portfolio for 2.0% Risk-Free Rate**
W1	0.187415826	-1.928804846	3.549557224
W2	-0.166481009	-0.946643452	1.073000527
W3	0.663185075	2.538414084	-2.316081298
W4	0.315880108	1.337034214	-1.306476453
σ_π	0.162749125	0.592549806	0.538335483

b. For plotting, the following steps were performed:

Step-1:

The characteristics of the efficient frontier, namely the return, standard deviation, were first calculated for the global minimum variance(GMV) portfolio.

For calculating the weights for the GMV portfolio, the analytical solution to the following optimization problem was used:

Minimize: $\frac{1}{2} w' \Sigma w$

Subject to: $w'1 = 1$

where the symbols have their usual meanings

The solution to this problem, which gives the weights of the constituent assets as follows:

$$w_g = \frac{\Sigma^{-1}1}{1'\Sigma^{-1}1}$$

was used to formulate the following function in Python

```
def gmv_portfolio(asset_characteristics,correl_matrix):
  l=np.array(df["Returns"])
  k=np.array(df["Volatility"])
  y=np.diag(k)@correl@np.diag(k)
  inverse_matrix=np.linalg.inv(y)
  A=np.full(k.shape[0],1)@inverse_matrix@np.full(k.shape[0],1)
  B=l@inverse_matrix@np.full(k.shape[0],1)
  C=l@inverse_matrix@l
  wg=(inverse_matrix@np.full(k.shape[0],1))/A
  mg=B/A
  sig=np.sqrt(wg@y@wg)
  return wg,mg,sig
```

The function assumes that the input data will comprise the following:

a. asset_characteristics: dataframe containing the characteristics of the constituent assets in 2 different columns, titled “Returns” (for asset returns) and “Volatility”(for individual standard deviations)
b. correl_matrix: correlation data

The function returns the standard deviation, return, and the weights of the individual assets.

The return of the GMV portfolio in this case was 1.60688 %.

Step-2:

Keeping the return and standard deviation of the GMV portfolio as reference, a series of asset returns was generated for each of which the standard deviation was calculated by finding the allocations for each of the returns in the series and using the same, along with the correlation matrix, to find the corresponding variance so that the efficient frontier could be plotted.

This was done by using the analytical solution to the following optimization problem:

Minimize: $\frac{1}{2} w' \Sigma w$

Subject to: $w'\mu = m,\ w'1 = 1$

This gives the following final form for the asset allocation when solved using Lagrange's method:

$$w^* = \frac{1}{AC - B^2} \Sigma^{-1}[(A\mu - B1)m + (C1 - B\mu)]$$

Where

$$A = 1'\Sigma^{-1}1$$

$$B = \mu'\Sigma^{-1}1$$

$$C = \mu'\Sigma^{-1}\mu$$

The volatility was obtained thereof as $\sqrt{w' \Sigma w}$.

This was implemented using the following Python function:

```
def benchmark_return_portfolio(asset_characteristics,correl_matrix,benchmark_return):
  s1_d=np.diag(asset_characteristics.Volatility.values)
  cov1=s1_d@correl_matrix@s1_d
  A=np.ones(s1_d.shape[0])@np.linalg.inv(cov1)@np.ones(s1_d.shape[0])
  B=np.array(asset_characteristics.Returns)@np.linalg.inv(cov1)@np.ones(s1_d.shape[0])
  C=np.array(asset_characteristics.Returns)@np.linalg.inv(cov1)@np.array(asset_characteristics.Returns)
  m1=benchmark_return*np.ones(s1_d.shape[0])
  lambda1=(A*benchmark_return-B)/(A*C-B**2)
  gamma1=(C-B*benchmark_return)/(A*C-B**2)
  term1=(A*np.array(asset_characteristics.Returns)-B*np.ones(s1_d.shape[0]))*benchmark_return
  term2=C*np.ones(s1_d.shape[0])-B*np.array(asset_characteristics.Returns)
  w=(np.linalg.inv(cov1)@(term1+term2))/(A*C-B**2)
  asset_characteristics["Weights(%)"]=w*100
  return np.sqrt(w@cov1@w)
```

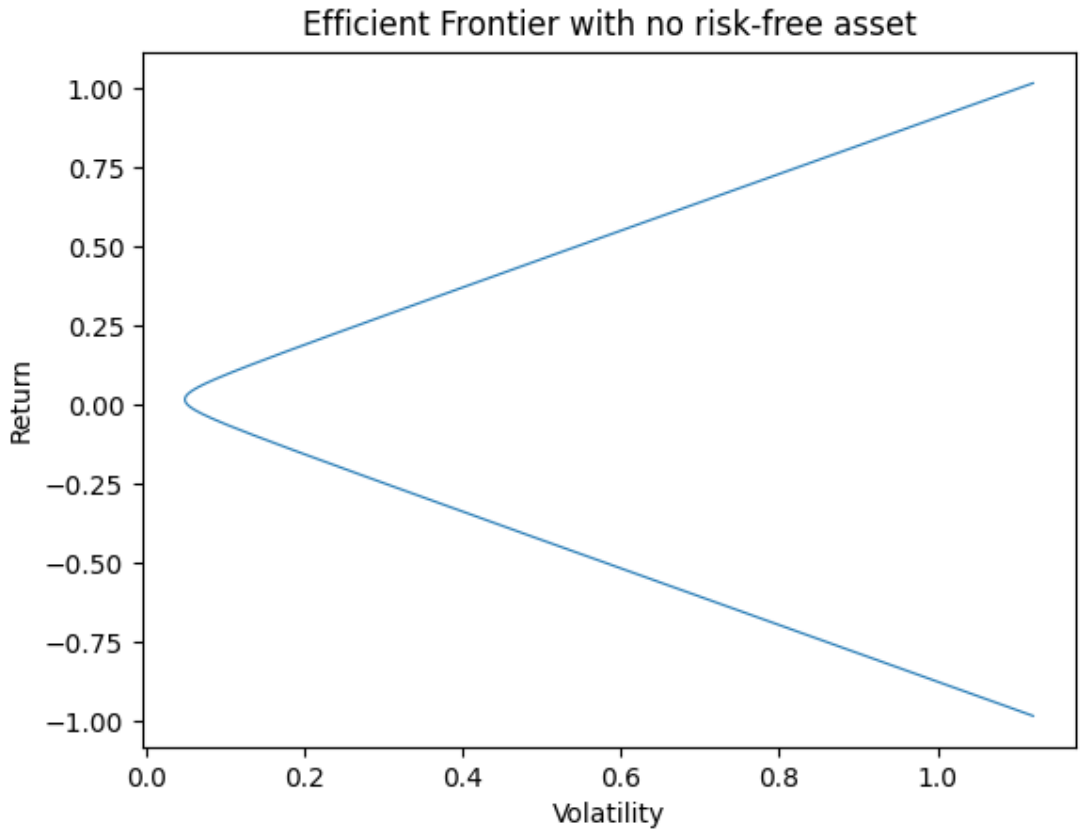
For ease of computation, some expressions were calculated separately for repeated use in the subsequent steps (A,B,C).

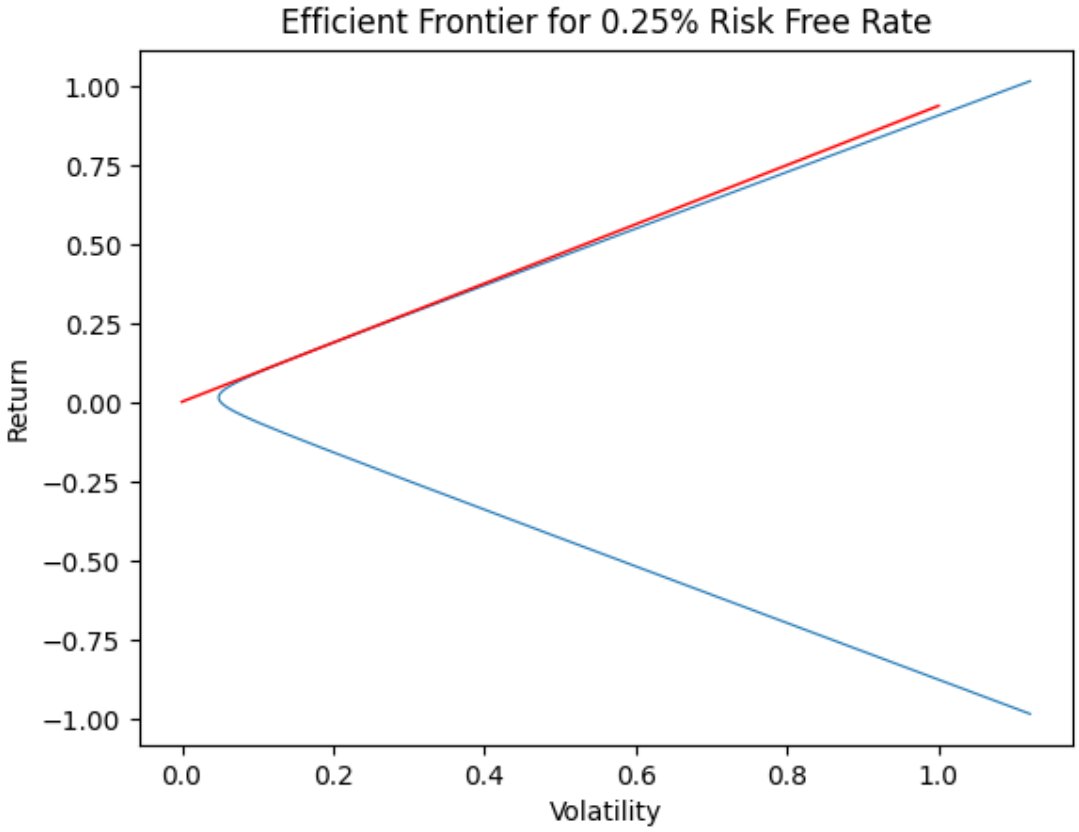
The function assumes that the input data will comprise the following:

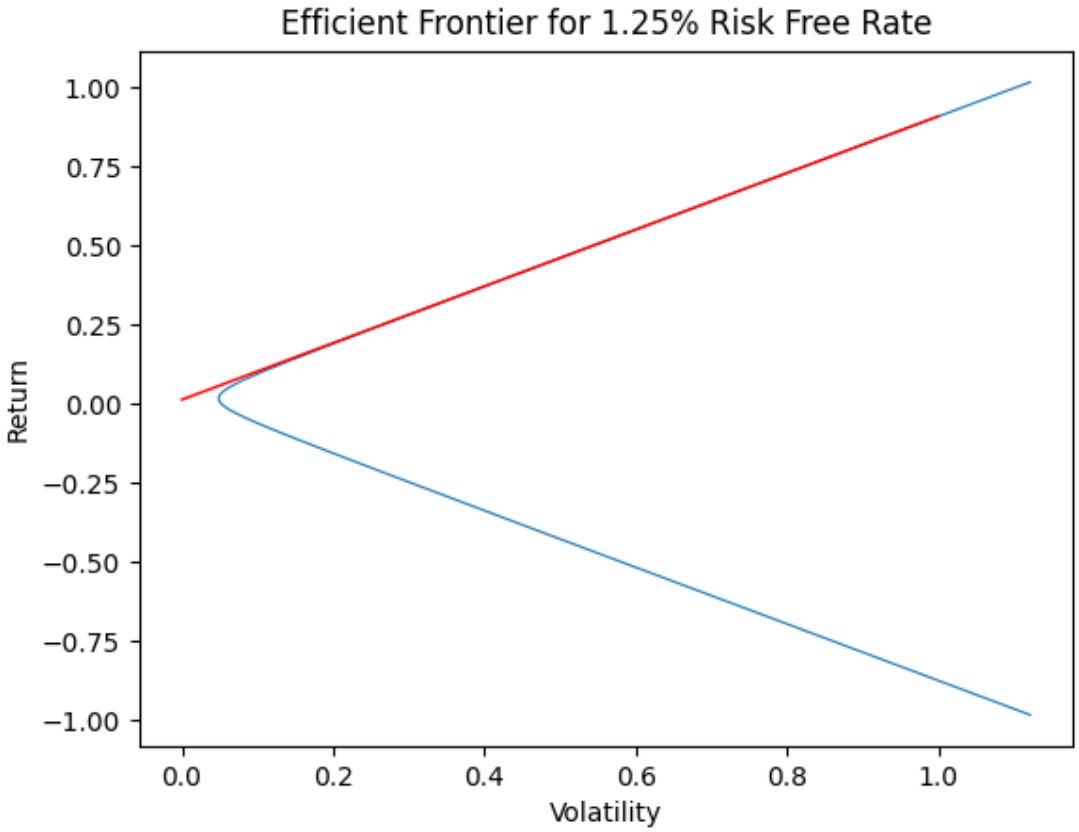
a. asset_characteristics: dataframe containing the characteristics of the constituent assets in 2 different columns, titled “Returns” (for asset returns) and “Volatility”(for individual standard deviations)
b. correl_matrix: correlation data
c. benchmark_return: target return for which the portfolio variance is being calculated.

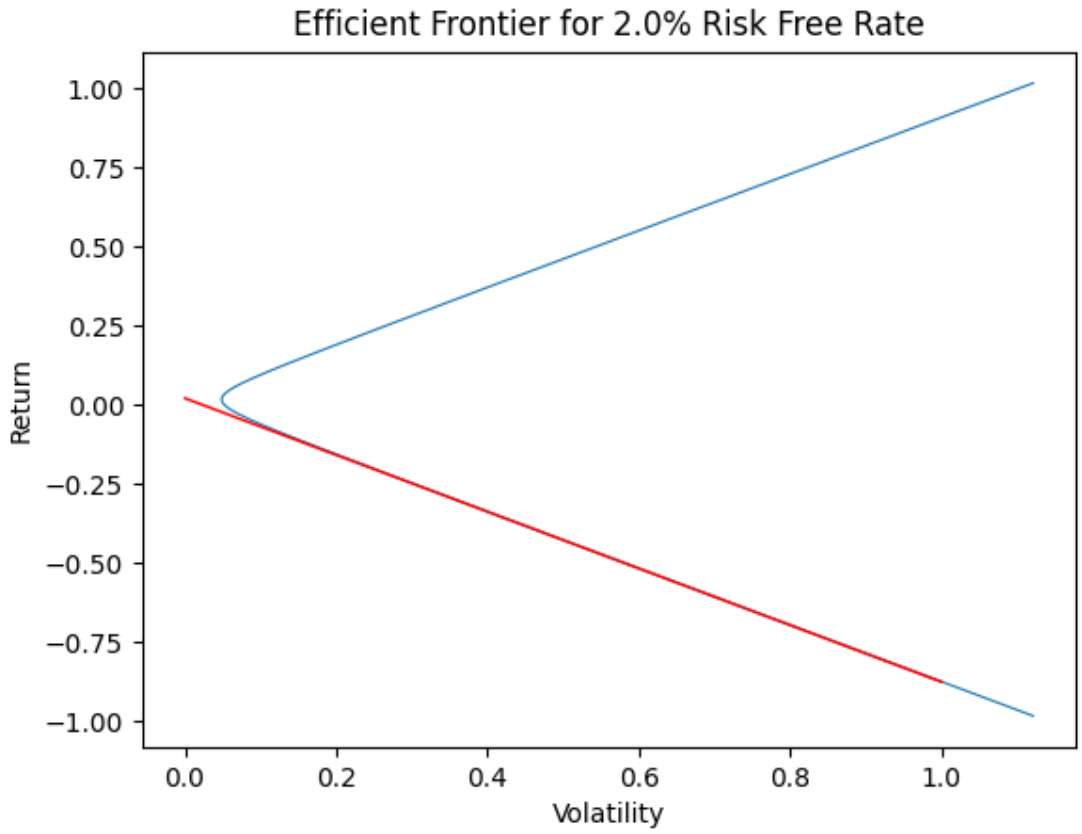
The function returns the standard deviation of each of the portfolios.

The following plots were obtained thereof:




Efficient Frontier for 0.25% Risk Free Rate
Return
Volatility
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.0
0.2
0.4
0.6
0.8
1.0


Efficient Frontier for 1.25% Risk Free Rate
Return
Volatility
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.0
0.2
0.4
0.6
0.8
1.0



C. To determine if, for each additional unit of risk taken by the investor in the tangency portfolio comprising assets A, B, C, D in a scenario where the risk-free rate was 200bps, the investor increased or reduced their excess return, the Sharpe ratio of the corresponding portfolio was examined.

The Sharpe ratio of the portfolio was -0.8964787192460693, which indicated that with each increasing unit of risk, the excess return will reduce.

Solutions for Task-2

The portfolio in discussion comprises assets A, B, C &D with the following characteristics:

Asset	Return (μ)	Standard Deviation (σ)	Allocated Weights*
A	0.05	0.07	W1
B	0.07	0.28	W2
C	0.15	0.25	W3
D	0.22	0.31	W4

*The allocated weights in this problem refer to the weights of the assets in the optimal portfolios with benchmark return m=7 %

The correlation matrix for the above-specified list of assets is:

1	0.4	0.3	0.3
0.4	1	0.27	0.42
0.3	0.27	1	0.5
0.3	0.42	0.5	1

The optimal portfolios needed to be obtained were for three scenarios, which are as follows:

1. where the correlation matrix is scaled by a factor of 1
2. where the correlation matrix is scaled by a factor of 1.3
3. where the correlation matrix is scaled by a factor of 1.8

These are subject to the constraint that:

1. Correlations of the assets with themselves remain the same (hence, diagonal elements remain 1)
2. Correlations between different assets cannot exceed 0.99.

To ensure this, the following Python function was used to calculate the correlation matrix in each of the cases:

```
def correl_adjust(factor,correl_matrix):
  correl2=np.zeros((correl_matrix.shape[0],correl_matrix.shape[1]))
  for i in range(correl_matrix.shape[0]):
     for j in range(correl_matrix.shape[1]):
         if i==j:
            correl2[i,j]=1
         else:
            cr1=factor*correl1[i,j]
            if cr1>=0.99:
               cr1=0.99
            elif cr1<0.99:
               cr1=cr1
            correl2[i,j]=cr1
  return correl2
```

The function takes the following inputs:

1. Factor: The factor by which the correlation matrix will be scaled.
2. Correl_matrix: The correlation matrix between the constituent assets.

The function returns the scaled correlation matrix, calculated subject to the constraints.

For calculating the weights for the optimal portfolio, the solution to the following optimization problem was used:

Minimize: $\frac{1}{2} w' \Sigma w$

Subject to: $w'\mu = m,\ w'1 = 1$

This gives the following final form for the asset allocation when solved using Lagrange's method:

$$w^* = \frac{1}{AC - B^2} \Sigma^{-1} [(A\mu - B1)m + (C1 - B\mu)]$$

Where

$$A = 1'\Sigma^{-1}1$$

$$B = \mu'\Sigma^{-1}1$$

$$C = \mu'\Sigma^{-1}\mu$$

The portfolio risk was obtained thereof as $\sqrt{w' \Sigma w}$.

The following Python function was used for implementing the solution:

```
def benchmark_return_portfolio(asset_characteristics,correl_matrix,benchmark_return):
  s1_d=np.diag(asset_characteristics.Volatility.values)
  cov1=s1_d@correl_matrix@s1_d
  A=np.ones(s1_d.shape[0])@np.linalg.inv(cov1)@np.ones(s1_d.shape[0])
  B=np.array(asset_characteristics.Returns)@np.linalg.inv(cov1)@np.ones(s1_d.shape[0])
  C=np.array(asset_characteristics.Returns)@np.linalg.inv(cov1)@np.array(asset_characteristics.Returns)
  m1=benchmark_return*np.ones(s1_d.shape[0])
  lambda1=(A*benchmark_return-B)/(A*C-B**2)
  gamma1=(C-B*benchmark_return)/(A*C-B**2)
  term1=(A*np.array(asset_characteristics.Returns)-B*np.ones(s1_d.shape[0]))*benchmark_return
  term2=C*np.ones(s1_d.shape[0])-B*np.array(asset_characteristics.Returns)
  w=(np.linalg.inv(cov1)@(term1+term2))/(A*C-B**2)
  return w
```

The function assumes that the input data will comprise the following:

a. asset_characteristics: dataframe containing the characteristics of the constituent assets in 2 different columns, titled "Returns" (for asset returns) and "Volatility"(for individual standard deviations)
b. correl_matrix: correlation data
c. benchmark_return: target return for which the portfolio variance is being calculated.

The function returns the weights of the constituent assets for each of the cases.

	Weights for correlation scaling factor = 1	Weights for correlation scaling factor = 1.3	Weights for correlation scaling factor = 1.8
W1	0.924135462	0.996471429	1.450877719
W2	-0.072894076	-0.135126388	-0.407703371
W3	0.054729757	0.012411647	-0.50705295
W4	0.094028858	0.126243312	0.463878603
Portfolio Risk	0.947102799	0.994393013	1.396532999

Solutions for Task-3

The following Python function was used to identify the Value at Risk(VaR) breaches for each of the datasets:

```
def custom_VaR_analyzer(df,alpha,lookback,period):
  df['Log Return']=np.log(df["Closing Price"].pct_change()+1)
  df['Standard_Deviations']=df["Log Return"].rolling(window=lookback).std()
  df['Period_Vol']=df['Standard_Deviations']*np.sqrt(period)
  factor=norm.ppf(1-alpha)
  df['Period_Returns']=np.log(df['Closing Price'].shift(-period)/df['Closing Price'])
  df['Period_VaR']=df['Period_Vol']*factor
  df['Breach']=np.where(df['Period_Returns']<df['Period_VaR'],1,0)
  df['Daily Breach']=np.where(df['Log Return']<df['Standard_Deviations']*factor,1,0)
  return df
```

The function takes in the following inputs:

1. df: Pandas dataframe containing the daily closing prices in a column titled "Closing Price" indexed by the dates.
2. alpha: The confidence level for VaR calculation
3. lookback: The number of periods over which the standard deviation in daily returns was calculated for estimating the VaR
4. period: The number of days over which the forward realized returns had to be computed for comparison against the estimated VaR over the same period for identifying the breaches.

The function performs the following steps:

1. Calculation of daily returns(logarithmic) using the closing prices.
2. Calculation of the rolling standard deviation in daily returns using the lookback as the length of the rolling window.
3. Calculation of the estimated standard deviation for the specified period
4. Calculation of the periodic returns(logarithmic) for the specified period
5. Calculation of the periodic VaR using the factor as determined from the confidence level and the corresponding estimated standard deviation for the period.

6. Identification of the breaches by comparing the estimated periodic VaR and the realized returns
7. Compilation of all the calculated data with the initial dataframe that was passed as an input

The function returns a dataframe containing all the calculated data for further operations.

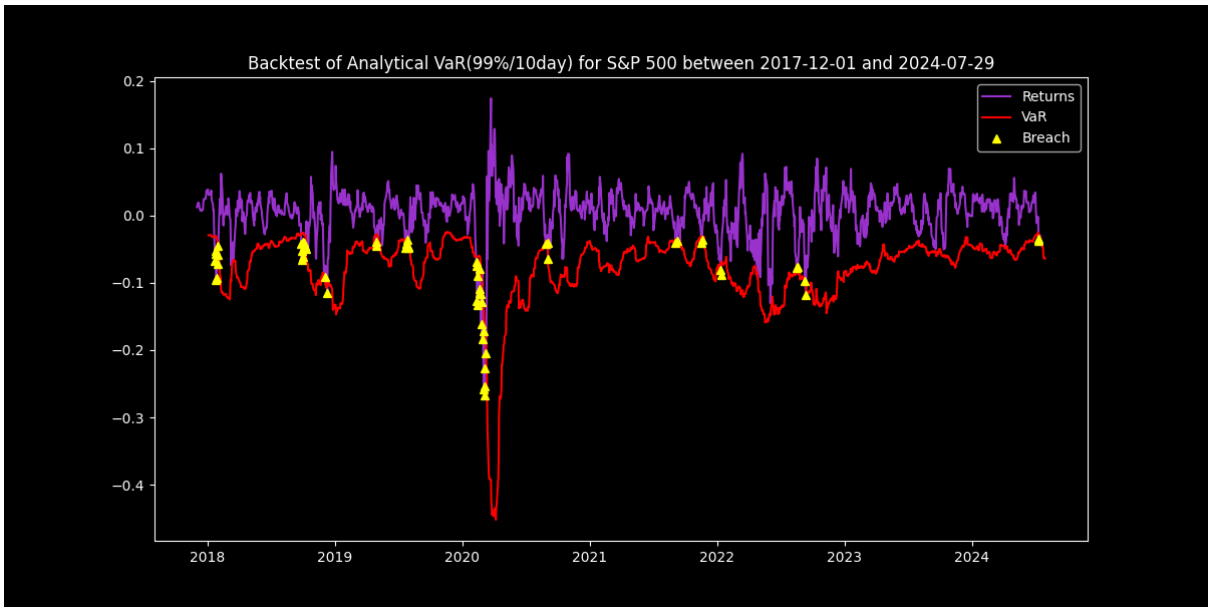
As per the lookback period in the question, periodic returns have been calculated as per the following formula:

$$\ln\left(\frac{S_{t+10}}{S_t}\right)$$

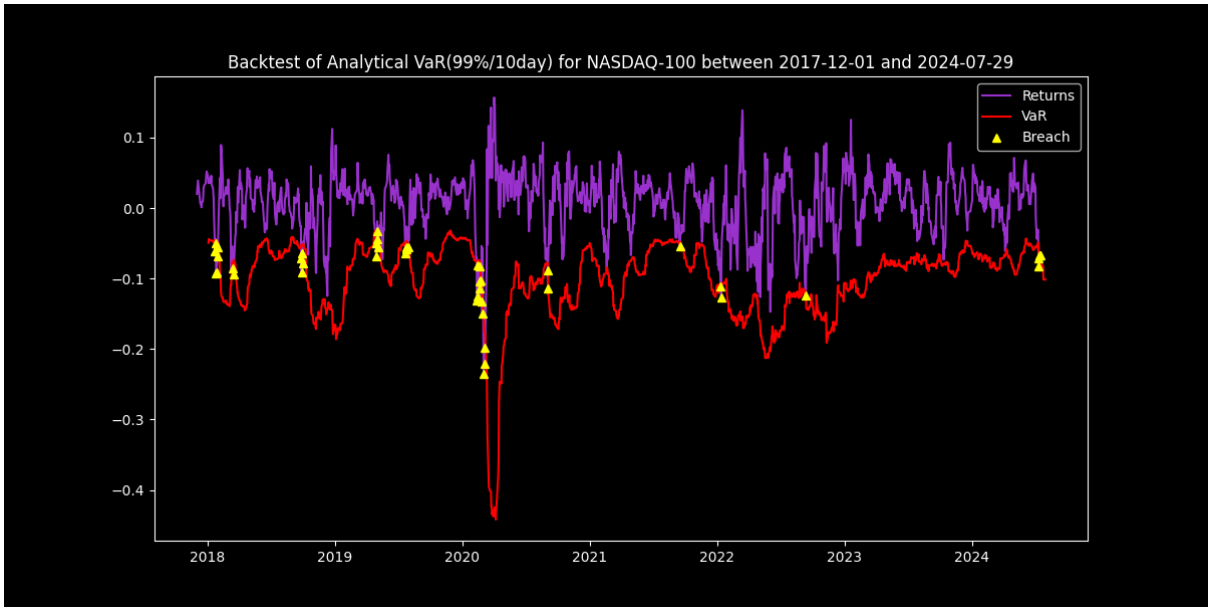
a. The count and percentage of breaches are presented in the following table:

	Actual Number of Breaches	Percentage of Actual Number of Breaches
SP500	61	3.646145
NASDAQ100	43	2.570233

b. Plot for S&P 500



Plot for NASDAQ-100



c. Breach list for S&P500

Date	Closing Price	Log Return	VaR_10D	Ret_10D
2018-01-22	2832.969971	0.008034364	-0.030955324	-0.067166083
2018-01-23	2839.129883	0.002172005	-0.030421812	-0.052047515
2018-01-24	2837.540039	-0.000560133	-0.030868284	-0.05650152
2018-01-25	2839.25	0.000602439	-0.030626317	-0.095363012
2018-01-26	2872.870117	0.011771638	-0.033195544	-0.092309005
2018-01-29	2853.530029	-0.006754736	-0.036757992	-0.071735603
2018-01-30	2822.429932	-0.010958643	-0.04300625	-0.058167438
2018-01-31	2823.810059	0.000488866	-0.041245497	-0.045342864
2018-09-26	2905.969971	-0.0032947	-0.027208592	-0.042275247
2018-09-27	2914	0.002759476	-0.027529541	-0.065822304
2018-09-28	2913.97998	-6.87014E-06	-0.025972271	-0.051709198
2018-10-01	2924.590088	0.003634492	-0.02546756	-0.061266173
2018-10-02	2923.429932	-0.000396769	-0.025495881	-0.039601576
2018-10-03	2925.51001	0.000711267	-0.025272125	-0.040565537
2018-10-04	2901.610107	-0.008203035	-0.028450694	-0.046858992
2018-10-05	2885.570068	-0.005543315	-0.029293142	-0.041676912
2018-10-08	2884.429932	-0.000395195	-0.029044341	-0.045590515
2018-10-09	2880.340088	-0.00141891	-0.029011422	-0.049698681
2018-12-03	2790.370117	0.010882002	-0.087383969	-0.091587938
2018-12-10	2637.719971	0.001760603	-0.100496271	-0.115031551
2019-04-29	2943.030029	0.001070949	-0.029541482	-0.045589889
2019-04-30	2945.830078	0.000950965	-0.028544251	-0.038556866
2019-07-22	2985.030029	0.002824702	-0.035037336	-0.048138135
2019-07-23	3005.469971	0.006824145	-0.03632172	-0.042029257
2019-07-24	3019.560059	0.004677193	-0.036531067	-0.045939867
2019-07-26	3025.860107	0.007360536	-0.034672445	-0.036074237
2019-07-29	3020.969971	-0.001617422	-0.034912778	-0.046850624

2019-07-31	2980.379883	-0.010945207	-0.038008158	-0.048035438
2020-02-10	3352.090088	0.00729969	-0.06121519	-0.069123115
2020-02-11	3357.75	0.001687049	-0.060851038	-0.074595861
2020-02-12	3379.449951	0.006441853	-0.060728863	-0.12620585
2020-02-13	3373.939941	-0.001631776	-0.060743735	-0.132846537
2020-02-14	3380.159912	0.001841836	-0.060743075	-0.089677504
2020-02-18	3370.290039	-0.002924214	-0.059897766	-0.115263775
2020-02-19	3386.149902	0.004694748	-0.060030178	-0.078622174
2020-02-20	3373.22998	-0.003822817	-0.060240969	-0.109310139
2020-02-21	3337.75	-0.010573809	-0.062928092	-0.115937273
2020-02-24	3225.889893	-0.034088079	-0.083572962	-0.160859607
2020-02-26	3116.389893	-0.003785697	-0.092687009	-0.128213846
2020-02-27	2978.76001	-0.045168136	-0.113487748	-0.182990562
2020-03-02	3090.22998	0.045010869	-0.138603337	-0.258572758
2020-03-03	3003.370117	-0.028510485	-0.142708307	-0.17183596
2020-03-04	3130.120117	0.041336349	-0.159139086	-0.266394582
2020-03-05	3023.939941	-0.034510782	-0.164807444	-0.227187039
2020-03-06	2972.370117	-0.017200944	-0.16396881	-0.254313719
2020-03-09	2746.560059	-0.079010413	-0.201082997	-0.205034807
2020-08-27	3484.550049	0.001671646	-0.038442423	-0.04207775
2020-09-01	3526.649902	0.007496831	-0.037592085	-0.040849735
2020-09-02	3580.840088	0.01524905	-0.042255377	-0.064546735
2021-09-03	4535.430176	-0.000335087	-0.035247804	-0.039968661
2021-09-07	4520.029785	-0.003401352	-0.035970008	-0.037379997
2021-11-16	4700.899902	0.003857779	-0.03132244	-0.040782966
2021-11-18	4704.540039	0.003379065	-0.030937426	-0.035946832
2022-01-11	4713.069824	0.009118286	-0.071796054	-0.08017967
2022-01-12	4726.350098	0.002813792	-0.070206365	-0.088392097
2022-08-17	4274.040039	-0.007264115	-0.076666723	-0.077578924
2022-08-18	4283.740234	0.00226699	-0.076635675	-0.076854161
2022-09-09	4067.360107	0.015156015	-0.094654621	-0.096492758
2022-09-12	4110.410156	0.010528651	-0.096653019	-0.11741577
2024-07-10	5633.910156	0.010156315	-0.027761907	-0.037393295
2024-07-11	5584.540039	-0.00880165	-0.03315073	-0.033747534

Breach list for NASDAQ100:

Date	Closing Price	Log Return	VaR_10D	Ret_10D
2018-01-22	6906.279785	0.01047266	-0.045830847	-0.061256836
2018-01-24	6919.350098	-0.006354623	-0.048341759	-0.049980157
2018-01-25	6916.299805	-0.000440932	-0.048180074	-0.092363482
2018-01-26	7022.970215	0.01530532	-0.049832394	-0.090908928
2018-01-29	6988.319824	-0.004946077	-0.051618321	-0.068775451
2018-01-30	6930.72998	-0.008275015	-0.055070061	-0.055910976
2018-03-14	7040.97998	-0.000785063	-0.077088907	-0.085992656

2018-03-16	7019.950195	-0.001568584	-0.073569278	-0.093890433
2018-09-26	7563.089844	-1.32349E-05	-0.053035696	-0.071032646
2018-09-27	7629.569824	0.008751648	-0.05483042	-0.091273165
2018-09-28	7627.649902	-0.000251674	-0.051329188	-0.063659556
2018-10-01	7645.450195	0.002330935	-0.051356914	-0.078438393
2019-04-29	7839.040039	0.001577951	-0.032649594	-0.067942022
2019-04-30	7781.459961	-0.007372407	-0.03581298	-0.05000997
2019-05-01	7751.850098	-0.003812439	-0.032134521	-0.032595277
2019-05-03	7845.72998	0.015629292	-0.039721784	-0.044575842
2019-05-06	7794.089844	-0.006603698	-0.041843485	-0.055039324
2019-07-22	7905.120117	0.008922566	-0.051379161	-0.063912648
2019-07-23	7954.560059	0.006234691	-0.051894835	-0.056003725
2019-07-24	8010.609863	0.007021539	-0.052531601	-0.058967722
2019-07-29	7989.080078	-0.003482456	-0.049339977	-0.054982208
2020-02-12	9613.200195	0.009967104	-0.07523232	-0.130549503
2020-02-13	9595.700195	-0.001822073	-0.074753847	-0.12574963
2020-02-14	9623.580078	0.002901243	-0.074622836	-0.080642238
2020-02-18	9629.799805	0.000646092	-0.073798653	-0.113741109
2020-02-19	9718.730469	0.009192563	-0.074448394	-0.082481887
2020-02-20	9627.830078	-0.009397129	-0.076802056	-0.104597616
2020-02-21	9446.69043	-0.01899341	-0.084088348	-0.102035259
2020-02-24	9079.629883	-0.039631031	-0.106606446	-0.133109358
2020-02-27	8436.669922	-0.050510909	-0.13283063	-0.149705228
2020-03-02	8877.980469	0.048008635	-0.156445801	-0.234756776
2020-03-04	8949.280273	0.040451785	-0.174706636	-0.22094524
2020-03-06	8530.339844	-0.016431054	-0.177351952	-0.198535094
2020-09-01	12292.86035	0.014929259	-0.077827931	-0.088863897
2020-09-02	12420.54004	0.010332922	-0.078245503	-0.114124122
2021-09-16	15515.91016	0.000798202	-0.047502585	-0.054724829
2022-01-11	15844.12012	0.014602994	-0.108917474	-0.111476665
2022-01-12	15905.09961	0.003841327	-0.106605124	-0.127360315
2022-09-12	12739.71973	0.011957631	-0.119533586	-0.123991224
2024-07-10	20675.38086	0.010813136	-0.049896498	-0.082801365
2024-07-11	20211.35938	-0.02269887	-0.065807557	-0.070762643
2024-07-12	20331.49023	0.005926136	-0.065628413	-0.066487451
2024-07-15	20386.88086	0.002720672	-0.063209744	-0.067326446

d. As a reference for the discussion, a few more data points are presented in the following table

	Pandemic Period (Feb 2020-March 2020)				Correction Period (2021-2022)			
	Daily Volatility	**Daily Breaches**	**10_Volatility**	**10D VaR Breaches**	**Daily Volatility**	**Daily Breaches**	**10D_Volatility**	**10D VaR Breaches**
SP500	0.04429213	6	0.11488	18	0.01227	12	0.035383	10
NASDAQ100	0.04392867	5	0.10193	12	0.0167	7	0.047044	4

For perspective, the daily and 10-day volatility, as well as the count of instances when the VaR level was breached in either time frame, have been summarized for both periods: the onset of the COVID-19 pandemic (February 2020-March 2020) and the correction period that followed (2021-2022).

Although it seems very obvious from the volatility and number of VaR breaches for 10-day returns that the S&P500 was more risky than the NASDAQ100, the volatility and the number of VaR breaches for the 1-day timeframe reveal that the risk was comparable for both indices. However, the higher number of VaR breaches in the 10-day timeframe despite the higher volatility does indicate that the S&P500 was marginally riskier than the NASDAQ100 at the onset of the pandemic.

For the correction period, the number of VaR breaches was significantly lower in the case of the NASDAQ100 for both the 10-day and daily timeframes than it was for the S&P500. While the relatively higher volatility can be a reason for this, the data does indicate that the S&P500 was relatively riskier than the NASDAQ100