# Final Project_PC

## Import library

In [1]:
```
import pandas as pd
import numpy as np
from numpy import *
from numpy.linalg import inv, multi_dot
import matplotlib.pyplot as plt
from seaborn import heatmap
import yfinance as yf
import cvxpy as cp
from sklearn.linear_model import LinearRegression
```

In [2]:
```
asset_symbols = ['VUG','VTV','SPHQ','SPHD','VHT','IYW','SCHX','SCHA','MTUM','EPS','MCHI','ECNS','FXI','KWEB','CHIR','CQQQ','PGJ','SCHO','
numofasset = len(asset_symbols)
numofportfolio = 5000
```

In [3]:
```
data = yf.download(asset_symbols, start="2021-01-01", end="2023-12-31")["Adj Close"]
data
```

```
[*********************100%%**********************]  25 of 25 completed
```

Out[3]:

| Date | CBON | CHIR | CQQQ | DBA | ECNS | EPS | FXI | IAU | IEO | IYW | ... | SCHO | SCHX | SPHD | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2021-01-04 | 22.385866 | 40.742649 | 82.585106 | 15.330893 | 43.823479 | 37.695938 | 43.141300 | 37.099998 | 32.500584 | 82.761574 | ... | 48.638527 | 42.955570 | 32.925514 | 3 |
| 2021-01-05 | 22.353634 | 41.005466 | 84.294441 | 15.416540 | 43.999660 | 37.933788 | 44.315346 | 37.180000 | 34.355942 | 83.432999 | ... | 48.629066 | 43.252129 | 33.201023 | 4 |
| 2021-01-06 | 22.284575 | 41.122276 | 83.340401 | 15.349924 | 44.669121 | 38.333397 | 43.802864 | 36.599998 | 35.378654 | 81.863075 | ... | 48.610134 | 43.515224 | 34.258533 | 4 |
| 2021-01-07 | 22.197088 | 41.148823 | 84.224876 | 15.340409 | 45.039085 | 38.828144 | 44.110348 | 36.480000 | 36.356121 | 84.292023 | ... | 48.610134 | 44.194477 | 34.151901 | 4 |
| 2021-01-08 | 22.174067 | 41.281559 | 87.961586 | 15.359441 | 45.858303 | 38.932804 | 45.023487 | 35.259998 | 35.939796 | 84.894325 | ... | 48.600674 | 44.443214 | 34.116344 | 4 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| 2023-12-22 | 21.919172 | 13.439922 | 34.630001 | 20.889999 | 24.639999 | 50.009998 | 23.049999 | 38.860001 | 94.339996 | 122.599998 | ... | 48.380001 | 56.230000 | 42.220001 | 5 |
| 2023-12-26 | 21.993999 | 13.406602 | 34.599998 | 20.980000 | 24.610001 | 50.220001 | 23.219999 | 39.139999 | 95.650002 | 123.250000 | ... | 48.369999 | 56.480000 | 42.450001 | 5 |
| 2023-12-27 | 21.990000 | 13.289001 | 34.500000 | 21.000000 | 24.690001 | 50.240002 | 23.250000 | 39.340000 | 95.169998 | 123.250000 | ... | 48.430000 | 56.570000 | 42.459999 | 5 |
| 2023-12-28 | 22.101999 | 13.680000 | 35.509998 | 21.049999 | 25.379999 | 50.270000 | 23.879999 | 39.099998 | 93.580002 | 123.309998 | ... | 48.430000 | 56.599998 | 42.610001 | 5 |
| 2023-12-29 | 22.100000 | 13.713000 | 35.950001 | 20.740000 | 25.650000 | 50.119999 | 24.030001 | 39.029999 | 93.190002 | 122.750000 | ... | 48.450001 | 56.400002 | 42.410000 | 5 |

753 rows × 25 columns

In [4]:
```
data2=yf.download('SPY ^IRX', start="2021-01-01", end="2023-12-31")["Adj Close"]

data2
```

```
[*********************100%%**********************]  2 of 2 completed
```

Out[4]:

| Date | SPY | ^IRX |
|---|---|---|
| 2021-01-04 | 352.767242 | 0.068 |
| 2021-01-05 | 355.196716 | 0.078 |
| 2021-01-06 | 357.320343 | 0.078 |
| 2021-01-07 | 362.629211 | 0.080 |
| 2021-01-08 | 364.695374 | 0.080 |
| ... | ... | ... |
| 2023-12-22 | 473.649994 | 5.208 |
| 2023-12-26 | 475.649994 | 5.203 |
| 2023-12-27 | 476.510010 | 5.235 |
| 2023-12-28 | 476.690002 | 5.218 |
| 2023-12-29 | 475.309998 | 5.180 |

753 rows × 2 columns

In [5]:
```
data.to_csv('data.csv')
data2.to_csv('data2.csv')
```

In [6]:
```
asset_symbols_total = ['SPY','VUG','VTV','SPHQ','SPHD','VHT','IYW','SCHX','SCHA','MTUM','EPS','MCHI','ECNS','FXI','KWEB','CHIR','CQQQ','F
data3=yf.download(asset_symbols_total, start="2021-01-01", end="2023-12-31")["Adj Close"]

data3
```

```
[*********************100%%**********************]  26 of 26 completed
```

Out[6]:

| Date | CBON | CHIR | CQQQ | DBA | ECNS | EPS | FXI | IAU | IEO | IYW | ... | SCHX | SPHD | SPHQ | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2021-01-04 | 22.385866 | 40.742649 | 82.585106 | 15.330893 | 43.823479 | 37.695938 | 43.141300 | 37.099998 | 32.500584 | 82.761574 | ... | 42.955570 | 32.925514 | 39.840996 | 3 |
| 2021-01-05 | 22.353634 | 41.005466 | 84.294441 | 15.416540 | 43.999660 | 37.933788 | 44.315346 | 37.180000 | 34.355942 | 83.432999 | ... | 43.252129 | 33.201023 | 40.137745 | 3 |
| 2021-01-06 | 22.284575 | 41.122276 | 83.340401 | 15.349924 | 44.669121 | 38.333397 | 43.802864 | 36.599998 | 35.378654 | 81.863075 | ... | 43.515224 | 34.258533 | 40.281338 | 3 |
| 2021-01-07 | 22.197088 | 41.148823 | 84.224876 | 15.340409 | 45.039085 | 38.828144 | 44.110348 | 36.480000 | 36.356121 | 84.292023 | ... | 44.194477 | 34.151901 | 40.874832 | 3 |
| 2021-01-08 | 22.174067 | 41.281559 | 87.961586 | 15.359441 | 45.858303 | 38.932804 | 45.023487 | 35.259998 | 35.939796 | 84.894325 | ... | 44.443214 | 34.116344 | 41.037563 | 3 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| 2023-12-22 | 21.919172 | 13.439922 | 34.630001 | 20.889999 | 24.639999 | 50.009998 | 23.049999 | 38.860001 | 94.339996 | 122.599998 | ... | 56.230000 | 42.220001 | 53.990002 | 4 |
| 2023-12-26 | 21.993999 | 13.406602 | 34.599998 | 20.980000 | 24.610001 | 50.220001 | 23.219999 | 39.139999 | 95.650002 | 123.250000 | ... | 56.480000 | 42.450001 | 54.220001 | 4 |
| 2023-12-27 | 21.990000 | 13.289001 | 34.500000 | 21.000000 | 24.690001 | 50.240002 | 23.250000 | 39.340000 | 95.169998 | 123.250000 | ... | 56.570000 | 42.459999 | 54.209999 | 4 |
| 2023-12-28 | 22.101999 | 13.680000 | 35.509998 | 21.049999 | 25.379999 | 50.270000 | 23.879999 | 39.099998 | 93.580002 | 123.309998 | ... | 56.599998 | 42.610001 | 54.150002 | 4 |
| 2023-12-29 | 22.100000 | 13.713000 | 35.950001 | 20.740000 | 25.650000 | 50.119999 | 24.030001 | 39.029999 | 93.190002 | 122.750000 | ... | 56.400002 | 42.410000 | 54.080002 | 4 |

753 rows × 26 columns

In [7]:
```
data3.to_csv('data3.csv')
```

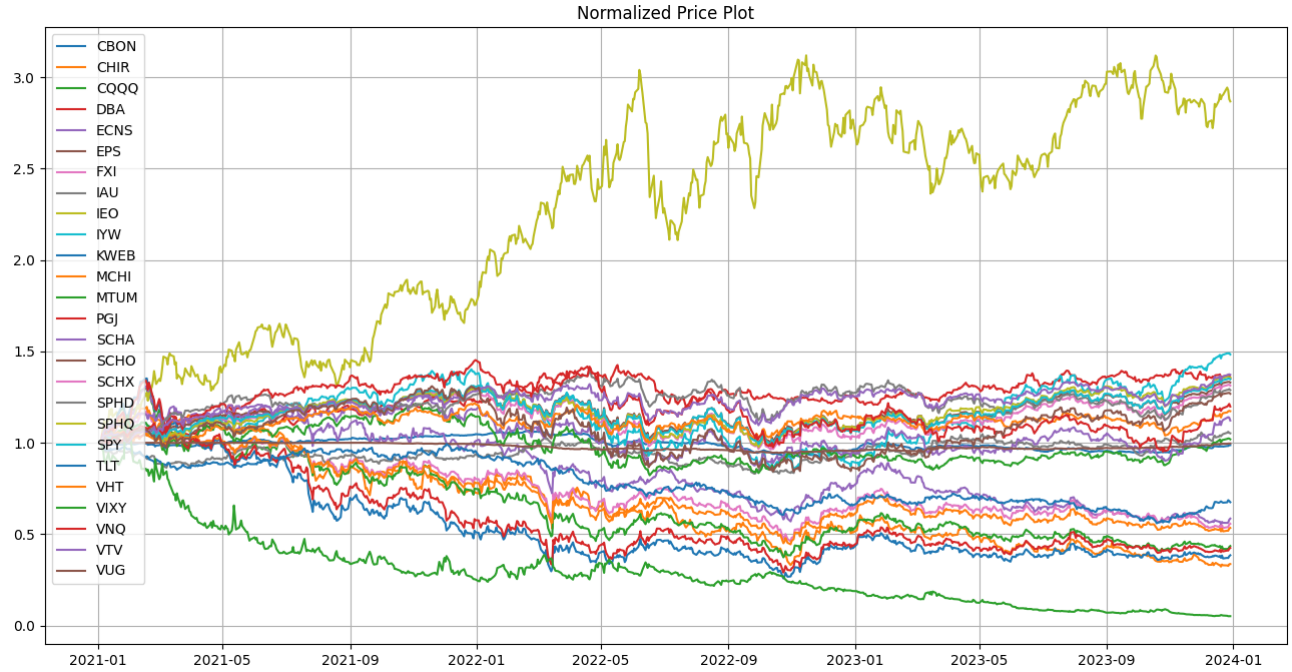
## Descriptive Statistics

In [8]:
```
summary = data.describe().T
print(summary)
summary_bm = data2.describe().T
print(summary_bm)
```

```
      count        mean        std         min         25%         50%  \
CBON  753.0   22.224314   0.729854   20.507000   21.677324   22.172417
CHIR  753.0   27.065884   9.790802   13.155718   19.133823   24.268414
CQQQ  753.0   54.539110  18.165754   30.827799   40.349461   46.510021
DBA   753.0   19.151461   1.518656   15.245245   18.290487   19.183060
ECNS  753.0   37.398467   8.663334   24.299999   29.597698   35.329769
EPS   753.0   44.026618   2.796661   37.649864   41.994453   44.043587
FXI   753.0   32.390296   7.106563   19.891132   26.907719   29.860435
IAU   753.0   35.089509   1.805845   30.820000   33.840000   34.840000
IEO   753.0   73.143614  19.278599   32.500584   54.118584   79.352219
IYW   753.0   95.386410  12.429266   70.271111   85.433052   93.775162
KWEB  753.0   38.641328  17.260561   18.098618   27.201998   30.180748
MCHI  753.0   55.185931  13.700858   33.893887   44.601433   49.570782
MTUM  753.0  151.737147  14.419757  126.597885  140.257034  146.377655
PGJ   753.0   36.288919  14.472503   17.781139   26.492886   29.372334
SCHA  753.0   44.421101   3.910334   36.842209   41.045567   44.020000
SCHO  753.0   47.556997   0.846180   45.912491   46.905720   47.330193
SCHX  753.0   49.039382   3.445832   41.418915   46.074978   48.856758
SPHD  753.0   40.430625   2.173642   32.925514   39.341740   40.424675
SPHQ  753.0   46.488779   3.508920   38.920799   43.930008   46.790051
TLT   753.0  114.806523  18.371216   81.962341   99.396301  109.904358
VHT   753.0  237.471171   9.569988  209.558731  231.255890  238.732269
VIXY  753.0   88.719987  63.204852   15.450000   44.049999   81.550003
VNQ   753.0   86.869364   8.786095   70.184471   79.673103   85.190147
VTV   753.0  133.442903   7.425253  108.974991  129.809586  134.793579
VUG   753.0  263.976571  28.866485  206.677948  240.934875  266.069092

             75%         max
CBON   22.694588   23.881083
CHIR   33.678555   46.885769
CQQQ   68.383644  107.440140
DBA    20.454918   21.849613
ECNS   45.787830   57.459419
EPS    46.276794   50.270000
FXI    37.646042   50.753922
IAU    36.540001   39.340000
IEO    89.431709  101.376060
IYW   105.990204  123.309998
KWEB   44.874565   94.879745
MCHI   65.049500   91.201508
MTUM  163.787109  186.955704
PGJ    43.320442   82.188820
SCHA   48.226463   53.229191
SCHO   48.565456   48.682201
SCHX   51.783684   56.599998
SPHD   41.813976   45.820721
SPHQ   49.352295   54.220001
TLT   133.422958  146.488831
VHT   244.126984  260.110840
VIXY  106.699997  346.000000
VNQ    94.983917  107.116745
VTV   138.466629  149.820007
VUG   286.935272  320.993195
      count        mean        std         min         25%         50%  \
SPY   753.0  410.823470  28.688917  349.616455  388.830292  409.29715
^IRX  753.0    2.351887   2.235035    0.003000    0.045000    1.73800

             75%         max
SPY   433.884186  476.690002
^IRX    4.728000    5.348000
```

## Visualize Data

In [9]:
```
# Visualize the data
fig = plt.figure(figsize=(16,8))
ax = plt.axes()

ax.set_title('Normalized Price Plot')
ax.plot(data3[-753:]/data3.iloc[-753] * 1)
ax.legend(data3.columns, loc='upper left')
ax.grid(True)
```




## Calculate Returns

In [10]:
```
# Calculate returns
returns = data.pct_change().fillna(0)
returns.head()
```

Out[10]:

| Date | CBON | CHIR | CQQQ | DBA | ECNS | EPS | FXI | IAU | IEO | IYW | ... | SCHO | SCHX | SPHD | SPHQ | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2021-01-04 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.0 |
| 2021-01-05 | -0.001440 | 0.006451 | 0.020698 | 0.005587 | 0.004020 | 0.006310 | 0.027214 | 0.002156 | 0.057087 | 0.008113 | ... | -0.000195 | 0.006904 | 0.008368 | 0.007448 | -0.0 |
| 2021-01-06 | -0.003089 | 0.002849 | -0.011318 | -0.004321 | 0.015215 | 0.010534 | -0.011564 | -0.015600 | 0.029768 | -0.018817 | ... | -0.000389 | 0.006083 | 0.031852 | 0.003578 | -0.0 |
| 2021-01-07 | -0.003926 | 0.000646 | 0.010613 | -0.000620 | 0.008282 | 0.012906 | 0.007020 | -0.003279 | 0.027629 | 0.029671 | ... | 0.000000 | 0.015610 | -0.003113 | 0.014734 | -0.0 |
| 2021-01-08 | -0.001037 | 0.003226 | 0.044366 | 0.001241 | 0.018189 | 0.002695 | 0.020701 | -0.033443 | -0.011451 | 0.007145 | ... | -0.000195 | 0.005628 | -0.001041 | 0.003981 | -0.0 |

5 rows × 25 columns

In [11]:
```
returns_bm = data2.pct_change().fillna(0)
returns_bm.head()
```

Out[11]:

| Date | SPY | ^IRX |
|---|---|---|
| 2021-01-04 | 0.000000 | 0.000000 |
| 2021-01-05 | 0.006887 | 0.147059 |
| 2021-01-06 | 0.005979 | 0.000000 |
| 2021-01-07 | 0.014857 | 0.025641 |
| 2021-01-08 | 0.005698 | 0.000000 |

## Log Returns

In [12]:
```
log_returns = np.log(data) - np.log(data.shift(1))
log_returns.fillna(0)
```

Out[12]:

| Date | CBON | CHIR | CQQQ | DBA | ECNS | EPS | FXI | IAU | IEO | IYW | ... | SCHO | SCHX | SPHD | SPH |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2021-01-04 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.00000 |
| 2021-01-05 | -0.001441 | 0.006430 | 0.020487 | 0.005571 | 0.004012 | 0.006290 | 0.026850 | 0.002154 | 0.055517 | 0.008080 | ... | -0.000195 | 0.006880 | 0.008333 | 0.0074 |
| 2021-01-06 | -0.003094 | 0.002845 | -0.011382 | -0.004330 | 0.015101 | 0.010479 | -0.011632 | -0.015723 | 0.029334 | -0.018996 | ... | -0.000389 | 0.006064 | 0.031355 | 0.0035 |
| 2021-01-07 | -0.003934 | 0.000645 | 0.010557 | -0.000620 | 0.008248 | 0.012824 | 0.006995 | -0.003284 | 0.027254 | 0.029239 | ... | 0.000000 | 0.015489 | -0.003117 | 0.01462 |
| 2021-01-08 | -0.001038 | 0.003221 | 0.043410 | 0.001240 | 0.018026 | 0.002692 | 0.020490 | -0.034015 | -0.011517 | 0.007120 | ... | -0.000195 | 0.005612 | -0.001042 | 0.00397 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| 2023-12-22 | -0.001819 | -0.012608 | -0.032669 | 0.004798 | -0.008889 | 0.002002 | -0.028231 | 0.004643 | 0.001485 | 0.001306 | ... | 0.000207 | 0.001958 | 0.004986 | 0.00092 |
| 2023-12-26 | 0.003408 | -0.002482 | -0.000867 | 0.004299 | -0.001218 | 0.004190 | 0.007348 | 0.007179 | 0.013790 | 0.005288 | ... | -0.000207 | 0.004436 | 0.005433 | 0.0042 |
| 2023-12-27 | -0.000182 | -0.008811 | -0.002894 | 0.000953 | 0.003245 | 0.000398 | 0.001291 | 0.005097 | -0.005031 | 0.000000 | ... | 0.001240 | 0.001592 | 0.000236 | -0.00018 |
| 2023-12-28 | 0.005080 | 0.028998 | 0.028855 | 0.002378 | 0.027563 | 0.000597 | 0.026736 | -0.006119 | -0.016848 | 0.000487 | ... | 0.000000 | 0.000530 | 0.003527 | -0.0011 |
| 2023-12-29 | -0.000090 | 0.002409 | 0.012315 | -0.014836 | 0.010582 | -0.002988 | 0.006262 | -0.001792 | -0.004176 | -0.004552 | ... | 0.000413 | -0.003540 | -0.004705 | -0.00129 |

753 rows × 25 columns

In [13]:
```
log_returns_bm = np.log(data2) - np.log(data2.shift(1))
log_returns_bm.fillna(0)
```

Out[13]:

| Date | SPY | ^IRX |
|---|---|---|
| 2021-01-04 | 0.000000 | 0.000000 |
| 2021-01-05 | 0.006863 | 0.137201 |
| 2021-01-06 | 0.005961 | 0.000000 |
| 2021-01-07 | 0.014748 | 0.025318 |
| 2021-01-08 | 0.005682 | 0.000000 |
| ... | ... | ... |
| 2023-12-22 | 0.002008 | -0.000384 |
| 2023-12-26 | 0.004214 | -0.000961 |
| 2023-12-27 | 0.001806 | 0.006131 |
| 2023-12-28 | 0.000378 | -0.003253 |
| 2023-12-29 | -0.002899 | -0.007309 |

753 rows × 2 columns

## Annualized Returns

In [14]:
```
# Calculate annual returns
annual_returns = (returns.mean() * 252)
annual_returns
```

Out[14]:
```
CBON   -0.002786
CHIR   -0.268983
CQQQ   -0.201775
DBA     0.110689
ECNS   -0.135532
EPS     0.109313
FXI    -0.141219
IAU     0.026889
IEO     0.413054
IYW     0.166998
KWEB   -0.172463
MCHI   -0.157723
MTUM    0.027125
PGJ    -0.158880
SCHA    0.064884
SCHO   -0.001081
SCHX    0.107229
SPHD    0.096225
SPHQ    0.117605
TLT    -0.115759
VHT     0.064782
VIXY   -0.768757
VNQ     0.080969
VTV     0.116644
VUG     0.106828
dtype: float64
```

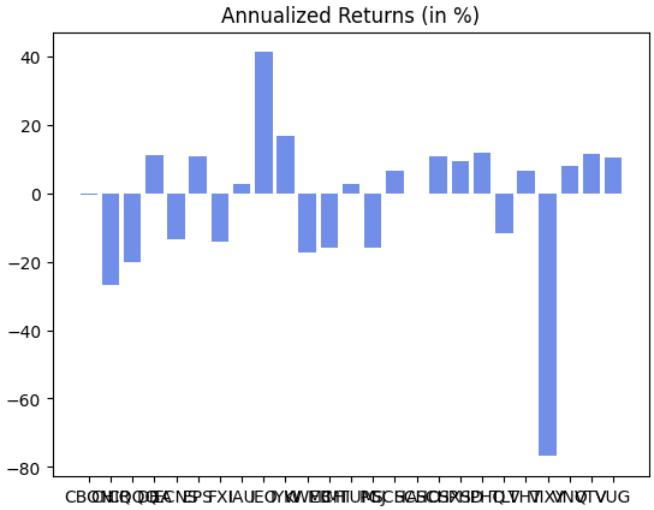
In [15]:
```
annual_returns_bm = (returns_bm.mean() * 252)
annual_returns_bm
```

Out[15]:
```
SPY     0.115257
^IRX    4.862025
dtype: float64
```

In [16]:
```
average_rf = data2['^IRX'].mean()/100
average_rf
```

Out[16]:
```
0.023518871237433566
```

In [17]:
```
# Visualize the data
fig = plt.figure()
ax =plt.axes()

ax.bar(annual_returns.index, annual_returns*100, color='royalblue', alpha=0.75)
ax.set_title('Annualized Returns (in %)');
```




## Calculate cumulative return

In [18]:
```
cumulative_return = (1 + returns).cumprod() - 1
cumulative_return.iloc[-1]
```

Out[18]:
```
CBON   -0.012770
CHIR   -0.663424
CQQQ   -0.564691
DBA     0.352824
ECNS   -0.414697
EPS     0.329586
FXI    -0.442993
IAU     0.052022
IEO     1.867333
IYW     0.483176
KWEB   -0.615322
MCHI   -0.465595
MTUM    0.018967
PGJ    -0.575253
SCHA    0.121687
SCHO   -0.003876
SCHX    0.312985
SPHD    0.288059
SPHQ    0.357396
TLT    -0.325000
VHT     0.172836
VIXY   -0.948472
VNQ     0.198221
VTV     0.371874
VUG     0.268272
Name: 2023-12-29 00:00:00, dtype: float64
```

In [19]:
```
cumulative_return_bm = (1 + returns_bm).cumprod() - 1
cumulative_return_bm.iloc[-1]
```

Out[19]:
```
SPY      0.347376
^IRX    75.176464
Name: 2023-12-29 00:00:00, dtype: float64
```

## Calculate Volatility

In [20]:
```
vols = returns.std()
vols
```

Out[20]:
```
CBON    0.003471
CHIR    0.027684
CQQQ    0.024893
DBA     0.008702
ECNS    0.018712
EPS     0.010529
FXI     0.020996
IAU     0.008876
IEO     0.021864
IYW     0.016693
KWEB    0.034797
MCHI    0.020471
MTUM    0.012856
PGJ     0.032204
SCHA    0.014501
SCHO    0.001318
SCHX    0.011300
SPHD    0.009558
SPHQ    0.011031
TLT     0.011188
VHT     0.009523
VIXY    0.042555
VNQ     0.012753
VTV     0.009266
VUG     0.014720
dtype: float64
```

In [21]:
```
vols_bm = returns_bm.std()
vols_bm
```

Out[21]:
```
SPY     0.011078
^IRX    0.192891
dtype: float64
```

## Annualized Volatilities

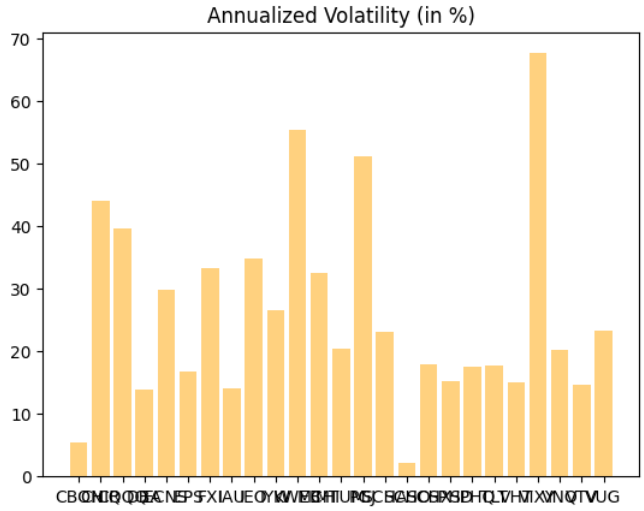
In [22]:
```
# Calculate annualized volatilities
annual_vols = vols*sqrt(252)
annual_vols
```

Out[22]:
```
CBON    0.055096
CHIR    0.439468
CQQQ    0.395171
DBA     0.138133
ECNS    0.297050
EPS     0.167137
FXI     0.333302
IAU     0.140901
IEO     0.347083
IYW     0.264986
KWEB    0.552384
MCHI    0.324971
MTUM    0.204075
PGJ     0.511224
SCHA    0.230201
SCHO    0.020925
SCHX    0.179387
SPHD    0.151731
SPHQ    0.175106
TLT     0.177610
VHT     0.151167
VIXY    0.675532
VNQ     0.202442
VTV     0.147089
VUG     0.233675
dtype: float64
```

In [23]:
```
annual_vols_bm = vols_bm*sqrt(252)
annual_vols_bm
```

Out[23]:
```
SPY     0.175851
^IRX    3.062050
dtype: float64
```

In [24]:
```
# Visualize the data
fig = plt.figure()
ax = plt.axes()

ax.bar(annual_vols.index, annual_vols*100, color='orange', alpha=0.5)
ax.set_title('Annualized Volatility (in %)');
```

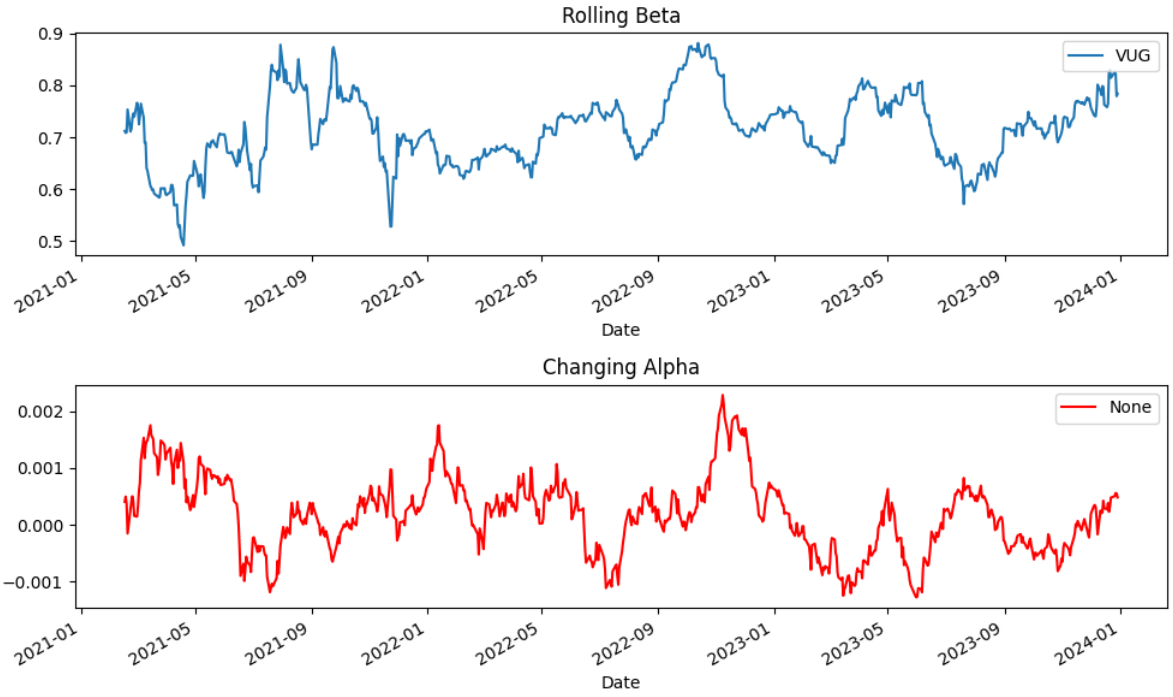



## Factor analysis

Method1:linear Regression

In [25]:
```
window_size = 30  # 滚动窗口大小
rolling_beta = []
changing_alpha = []

for i in range(len(log_returns) - window_size + 1):
    factor_window_data = log_returns.iloc[i:i+window_size].copy()
    benchmark_window_data = log_returns_bm.iloc[i:i+window_size].copy()
    factor_window_data.dropna(inplace=True)
    benchmark_window_data.dropna(inplace=True)


    X = factor_window_data[['VUG']]
    y = benchmark_window_data['SPY']

    model = LinearRegression().fit(X, y)

    rolling_beta.append(model.coef_)
    changing_alpha.append(model.intercept_)

rolling_beta = pd.DataFrame(rolling_beta, columns=['VUG'], index=log_returns.index[window_size-1:])
changing_alpha = pd.Series(changing_alpha, index=log_returns.index[window_size-1:])
```

In [26]:
```
plt.figure(figsize=(10, 6))

plt.subplot(2, 1, 1)
rolling_beta.plot(title='Rolling Beta', ax=plt.gca())
plt.legend(loc='upper right')

plt.subplot(2, 1, 2)
changing_alpha.plot(title='Changing Alpha', color='r')
plt.legend(loc='upper right')

plt.tight_layout()
plt.show()
```

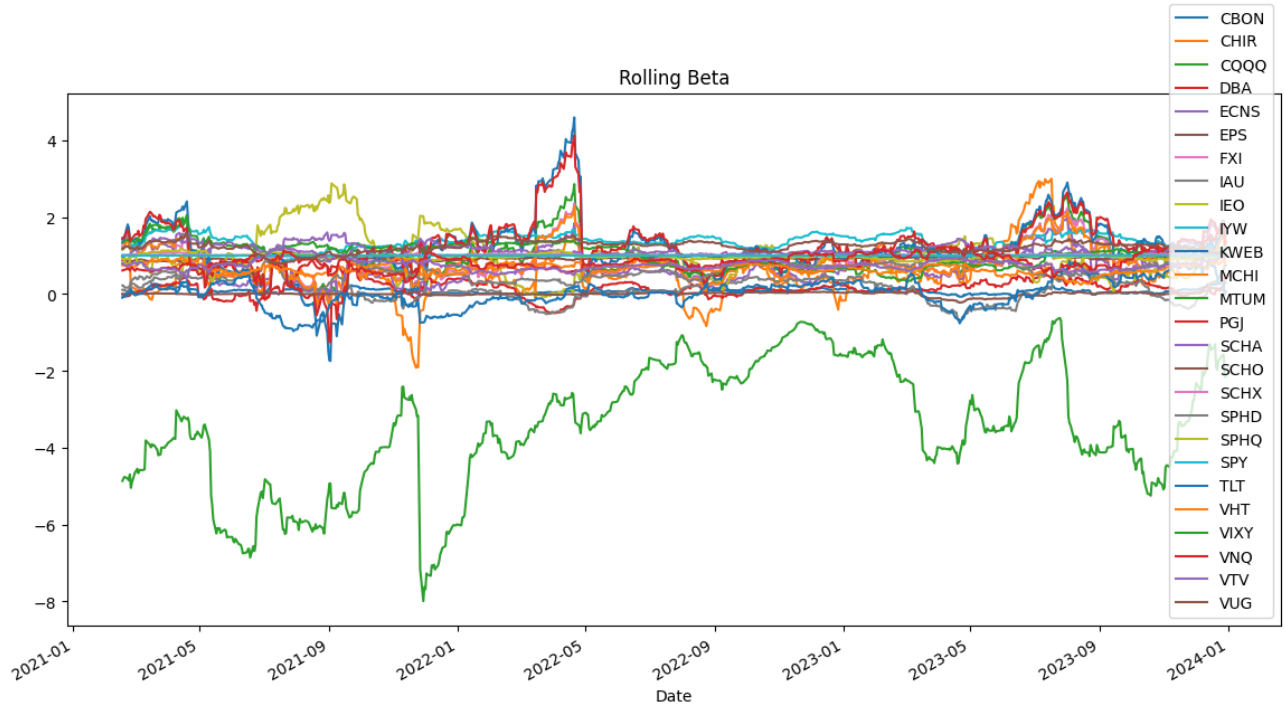


Method2 normal

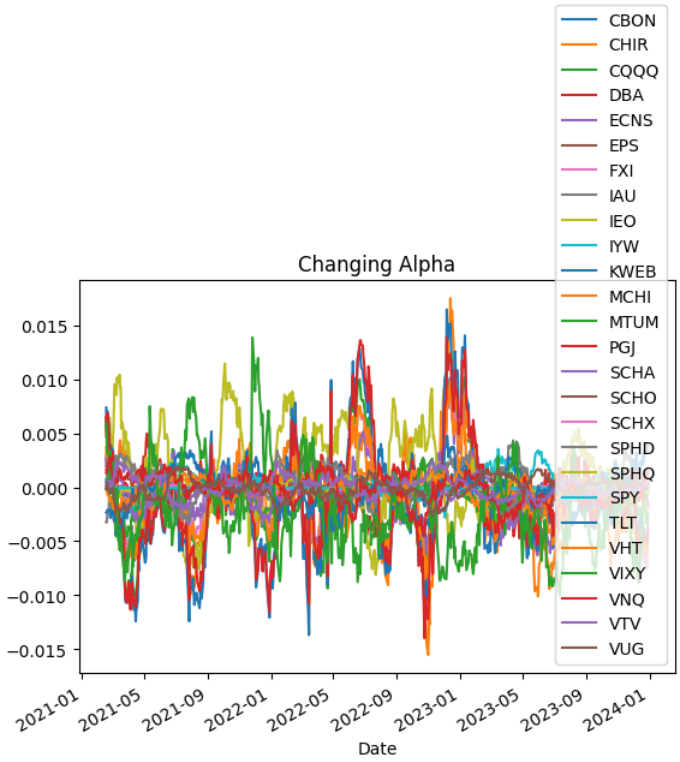
In [27]:
```
returns_alphabeta = data3.pct_change()
def calculate_rolling_beta(asset_returns, market_returns, window):
    return asset_returns.rolling(window=window).cov(market_returns) / market_returns.rolling(window=window).var()

rolling_betas = returns_alphabeta.apply(lambda x: calculate_rolling_beta(x, returns_alphabeta['SPY'], window_size))


market_excess_return = returns_alphabeta['SPY']
#average_rf
def calculate_changing_alpha(asset_returns, market_excess_return, beta, window):
    alpha = asset_returns.rolling(window=window).mean() - beta * market_excess_return.rolling(window=window).mean()
    return alpha

changing_alphas = returns_alphabeta.apply(lambda x: calculate_changing_alpha(x, market_excess_return, rolling_betas[x.name], window_size)

rolling_betas['VUG'].dropna().tail(), changing_alphas['VUG'].dropna().tail()
```

Out[27]:
```
(Date
 2023-12-22    1.059907
 2023-12-26    1.019803
 2023-12-27    1.022546
 2023-12-28    1.015694
 2023-12-29    1.022625
 Name: VUG, dtype: float64,
 Date
 2023-12-22   -0.000070
 2023-12-26   -0.000092
 2023-12-27   -0.000140
 2023-12-28   -0.000130
 2023-12-29   -0.000111
 Name: VUG, dtype: float64)
```

In [28]:
```
plt.figure(figsize=(14, 7))

plt.subplot()
rolling_betas.plot(title='Rolling Beta', ax=plt.gca())
plt.legend(loc='lower right')


plt.show()
```




In [29]:
```
changing_alphas.plot(title='Changing Alpha')
plt.legend(loc='lower right')

plt.subplot()
plt.figure(figsize=(14, 7))
plt.tight_layout()
plt.show()
```




```
<Figure size 1400x700 with 0 Axes>
```

## Equilibrium Returns

### covariance matrix

In [30]:
```
cov_matrix = returns.cov()*252
cov_matrix
```

Out[30]:

| | CBON | CHIR | CQQQ | DBA | ECNS | EPS | FXI | IAU | IEO | IYW | ... | SCHO | SCHX | SPHD | SPH |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **CBON** | 0.003036 | 0.007349 | 0.008048 | 0.001085 | 0.006238 | 0.002107 | 0.006768 | 0.003019 | 0.002800 | 0.003479 | ... | 0.000289 | 0.002392 | 0.001609 | 0.00233 |
| **CHIR** | 0.007349 | 0.193132 | 0.096439 | 0.005632 | 0.082389 | 0.016290 | 0.093994 | 0.010755 | 0.027548 | 0.022879 | ... | -0.000183 | 0.017015 | 0.010726 | 0.01739 |
| **CQQQ** | 0.008048 | 0.096439 | 0.156160 | 0.003089 | 0.100886 | 0.026774 | 0.122856 | 0.009067 | 0.026378 | 0.049667 | ... | 0.000344 | 0.031398 | 0.011014 | 0.02941 |
| **DBA** | 0.001085 | 0.005632 | 0.003089 | 0.019081 | 0.003150 | 0.003737 | 0.003988 | 0.004281 | 0.016051 | 0.003442 | ... | -0.000098 | 0.003693 | 0.004062 | 0.00382 |
| **ECNS** | 0.006238 | 0.082389 | 0.100886 | 0.003150 | 0.088239 | 0.019517 | 0.085942 | 0.006619 | 0.021623 | 0.033472 | ... | 0.000109 | 0.022413 | 0.010397 | 0.02068 |
| **EPS** | 0.002107 | 0.016290 | 0.026774 | 0.003737 | 0.019517 | 0.027935 | 0.023111 | 0.003211 | 0.027189 | 0.039037 | ... | 0.000213 | 0.029507 | 0.019572 | 0.02835 |
| **FXI** | 0.006768 | 0.093994 | 0.122856 | 0.003988 | 0.085942 | 0.023111 | 0.111090 | 0.008202 | 0.025043 | 0.038596 | ... | 0.000131 | 0.026140 | 0.012074 | 0.02477 |
| **IAU** | 0.003019 | 0.010755 | 0.009067 | 0.004281 | 0.006619 | 0.003211 | 0.008202 | 0.019853 | 0.009085 | 0.004663 | ... | 0.001373 | 0.003633 | 0.004341 | 0.00333 |
| **IEO** | 0.002800 | 0.027548 | 0.026378 | 0.016051 | 0.021623 | 0.027189 | 0.025043 | 0.009085 | 0.120467 | 0.024758 | ... | -0.000675 | 0.026603 | 0.027744 | 0.02791 |
| **IYW** | 0.003479 | 0.022879 | 0.049667 | 0.003442 | 0.033472 | 0.039037 | 0.038596 | 0.004663 | 0.024758 | 0.070218 | ... | 0.000518 | 0.044045 | 0.019009 | 0.04103 |
| **KWEB** | 0.009661 | 0.137529 | 0.207522 | 0.004433 | 0.136302 | 0.034443 | 0.173366 | 0.010099 | 0.032290 | 0.064325 | ... | 0.000170 | 0.040648 | 0.013253 | 0.03744 |
| **MCHI** | 0.006980 | 0.089899 | 0.122862 | 0.003885 | 0.085469 | 0.022332 | 0.106898 | 0.008217 | 0.023969 | 0.038730 | ... | 0.000163 | 0.025700 | 0.010843 | 0.02418 |
| **MTUM** | 0.002490 | 0.019526 | 0.036657 | 0.005177 | 0.026093 | 0.028024 | 0.028365 | 0.002543 | 0.032186 | 0.042981 | ... | -0.000021 | 0.031307 | 0.016079 | 0.03052 |
| **PGJ** | 0.009855 | 0.124167 | 0.187269 | 0.005604 | 0.124753 | 0.036100 | 0.155534 | 0.009066 | 0.036141 | 0.066249 | ... | 0.000191 | 0.042563 | 0.014680 | 0.03837 |
| **SCHA** | 0.003151 | 0.023812 | 0.042480 | 0.005251 | 0.030556 | 0.033232 | 0.034820 | 0.004300 | 0.042314 | 0.046129 | ... | 0.000233 | 0.036212 | 0.025399 | 0.03362 |
| **SCHO** | 0.000289 | -0.000183 | 0.000344 | -0.000098 | 0.000109 | 0.000213 | 0.000131 | 0.001373 | -0.000675 | 0.000518 | ... | 0.000438 | 0.000309 | 0.000261 | 0.00021 |
| **SCHX** | 0.002392 | 0.017015 | 0.031398 | 0.003693 | 0.022413 | 0.029507 | 0.026140 | 0.003633 | 0.026603 | 0.044045 | ... | 0.000309 | 0.032180 | 0.019408 | 0.03041 |
| **SPHD** | 0.001609 | 0.010726 | 0.011014 | 0.004062 | 0.010397 | 0.019572 | 0.012074 | 0.004341 | 0.027744 | 0.019009 | ... | 0.000261 | 0.019408 | 0.023022 | 0.01882 |
| **SPHQ** | 0.002333 | 0.017398 | 0.029415 | 0.003821 | 0.020683 | 0.028355 | 0.024773 | 0.003332 | 0.027910 | 0.041033 | ... | 0.000216 | 0.030415 | 0.018829 | 0.03066 |
| **TLT** | 0.001286 | -0.002572 | 0.002650 | -0.002088 | 0.001211 | 0.000853 | 0.002233 | 0.008266 | -0.009389 | 0.004387 | ... | 0.002135 | 0.001970 | 0.001059 | 0.00055 |
| **VHT** | 0.001778 | 0.009577 | 0.018222 | 0.001863 | 0.014137 | 0.020133 | 0.015472 | 0.002806 | 0.014302 | 0.026789 | ... | 0.000371 | 0.021785 | 0.014921 | 0.02096 |
| **VIXY** | -0.007748 | -0.063094 | -0.098392 | -0.013510 | -0.072885 | -0.083547 | -0.083205 | -0.004963 | -0.090867 | -0.119425 | ... | 0.000728 | -0.090961 | -0.055214 | -0.08594 |
| **VNQ** | 0.002537 | 0.015586 | 0.022608 | 0.003562 | 0.017403 | 0.025363 | 0.020870 | 0.005847 | 0.022257 | 0.032753 | ... | 0.000806 | 0.027485 | 0.024232 | 0.02519 |
| **VTV** | 0.001575 | 0.012457 | 0.016654 | 0.004084 | 0.013365 | 0.022558 | 0.015822 | 0.003089 | 0.030027 | 0.025355 | ... | 0.000097 | 0.023048 | 0.020081 | 0.02279 |
| **VUG** | 0.003109 | 0.021121 | 0.044175 | 0.003259 | 0.030141 | 0.035683 | 0.035103 | 0.004039 | 0.023136 | 0.060811 | ... | 0.000502 | 0.040213 | 0.018656 | 0.03719 |

25 rows × 25 columns

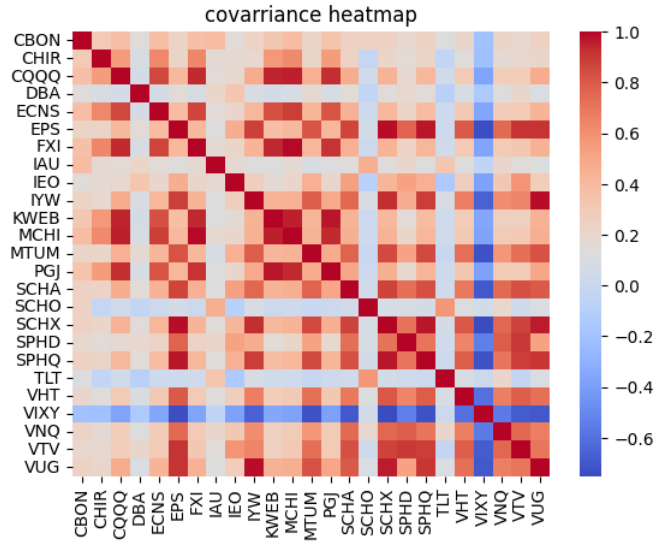
In [31]:
```
corr_matrix = returns.corr()
corr_matrix
```

Out[31]:

| | CBON | CHIR | CQQQ | DBA | ECNS | EPS | FXI | IAU | IEO | IYW | ... | SCHO | SCHX | SPHD | SPHQ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **CBON** | 1.000000 | 0.303518 | 0.369644 | 0.142507 | 0.381141 | 0.228831 | 0.368560 | 0.388852 | 0.146407 | 0.238273 | ... | 0.250448 | 0.242050 | 0.192486 | 0.241829 |
| **CHIR** | 0.303518 | 1.000000 | 0.555316 | 0.092771 | 0.631123 | 0.221778 | 0.641707 | 0.173691 | 0.180606 | 0.196468 | ... | -0.019944 | 0.215834 | 0.160850 | 0.226079 |
| **CQQQ** | 0.369644 | 0.555316 | 1.000000 | 0.056594 | 0.859440 | 0.405382 | 0.932772 | 0.162832 | 0.192316 | 0.474311 | ... | 0.041586 | 0.442923 | 0.183683 | 0.425089 |
| **DBA** | 0.142507 | 0.092771 | 0.056594 | 1.000000 | 0.076756 | 0.161882 | 0.086629 | 0.219953 | 0.334793 | 0.094024 | ... | -0.033811 | 0.149041 | 0.193803 | 0.157986 |
| **ECNS** | 0.381141 | 0.631123 | 0.859440 | 0.076756 | 1.000000 | 0.393113 | 0.868035 | 0.158147 | 0.209725 | 0.425236 | ... | 0.017608 | 0.420605 | 0.230677 | 0.397628 |
| **EPS** | 0.228831 | 0.221778 | 0.405382 | 0.161882 | 0.393113 | 1.000000 | 0.414861 | 0.136366 | 0.468693 | 0.881425 | ... | 0.060859 | 0.984160 | 0.771770 | 0.968854 |
| **FXI** | 0.368560 | 0.641707 | 0.932772 | 0.086629 | 0.868035 | 0.414861 | 1.000000 | 0.174655 | 0.216478 | 0.436999 | ... | 0.018808 | 0.437199 | 0.238747 | 0.424468 |
| **IAU** | 0.388852 | 0.173691 | 0.162832 | 0.219953 | 0.158147 | 0.136366 | 0.174655 | 1.000000 | 0.185762 | 0.124885 | ... | 0.465573 | 0.143732 | 0.203058 | 0.135033 |
| **IEO** | 0.146407 | 0.180606 | 0.192316 | 0.334793 | 0.209725 | 0.468693 | 0.216478 | 0.185762 | 1.000000 | 0.269185 | ... | -0.092975 | 0.427266 | 0.526828 | 0.459218 |
| **IYW** | 0.238273 | 0.196468 | 0.474311 | 0.094024 | 0.425236 | 0.881425 | 0.436999 | 0.124885 | 0.269185 | 1.000000 | ... | 0.093446 | 0.926573 | 0.472789 | 0.884321 |
| **KWEB** | 0.317450 | 0.566536 | 0.950690 | 0.058101 | 0.830675 | 0.373069 | 0.941639 | 0.129751 | 0.168419 | 0.439454 | ... | 0.014739 | 0.410210 | 0.158129 | 0.387097 |
| **MCHI** | 0.389845 | 0.629482 | 0.956729 | 0.086542 | 0.885385 | 0.411167 | 0.986928 | 0.179456 | 0.212506 | 0.449761 | ... | 0.023998 | 0.440850 | 0.219910 | 0.424958 |
| **MTUM** | 0.221484 | 0.217719 | 0.454548 | 0.183650 | 0.430425 | 0.821618 | 0.417025 | 0.088428 | 0.454401 | 0.794811 | ... | -0.004900 | 0.855184 | 0.519285 | 0.854070 |
| **PGJ** | 0.349897 | 0.552673 | 0.926978 | 0.079355 | 0.821502 | 0.422498 | 0.912803 | 0.125867 | 0.203681 | 0.489038 | ... | 0.017837 | 0.464117 | 0.189257 | 0.428621 |
| **SCHA** | 0.248460 | 0.235377 | 0.466974 | 0.165123 | 0.446848 | 0.863723 | 0.453822 | 0.132559 | 0.529596 | 0.756211 | ... | 0.048282 | 0.876908 | 0.727179 | 0.834038 |
| **SCHO** | 0.250448 | -0.019944 | 0.041586 | -0.033811 | 0.017608 | 0.060859 | 0.018808 | 0.465573 | -0.092975 | 0.093446 | ... | 1.000000 | 0.082274 | 0.082218 | 0.059058 |
| **SCHX** | 0.242050 | 0.215834 | 0.442923 | 0.149041 | 0.420605 | 0.984160 | 0.437199 | 0.143732 | 0.427266 | 0.926573 | ... | 0.082274 | 1.000000 | 0.713051 | 0.968269 |
| **SPHD** | 0.192486 | 0.160850 | 0.183683 | 0.193803 | 0.230677 | 0.771770 | 0.238747 | 0.203058 | 0.526828 | 0.472789 | ... | 0.082218 | 0.713051 | 1.000000 | 0.708697 |
| **SPHQ** | 0.241829 | 0.226079 | 0.425089 | 0.157986 | 0.397628 | 0.968854 | 0.424468 | 0.135033 | 0.459218 | 0.884321 | ... | 0.059058 | 0.968269 | 0.708697 | 1.000000 |
| **TLT** | 0.131441 | -0.032957 | 0.037754 | -0.085122 | 0.022953 | 0.028738 | 0.037723 | 0.330318 | -0.152300 | 0.093217 | ... | 0.574587 | 0.061846 | 0.039292 | 0.017714 |
| **VHT** | 0.213471 | 0.144160 | 0.305041 | 0.089196 | 0.314830 | 0.796851 | 0.307087 | 0.131737 | 0.272592 | 0.668763 | ... | 0.117232 | 0.803352 | 0.650533 | 0.792121 |
| **VIXY** | -0.208175 | -0.212528 | -0.368579 | -0.144777 | -0.363214 | -0.739967 | -0.369543 | -0.052143 | -0.387547 | -0.667151 | ... | 0.051480 | -0.750621 | -0.538676 | -0.726535 |
| **VNQ** | 0.227423 | 0.175192 | 0.282598 | 0.127379 | 0.289391 | 0.749583 | 0.309301 | 0.204990 | 0.316765 | 0.610559 | ... | 0.190365 | 0.756836 | 0.788876 | 0.710847 |
| **VTV** | 0.194355 | 0.192709 | 0.286523 | 0.201008 | 0.305878 | 0.917585 | 0.322739 | 0.149036 | 0.588158 | 0.650516 | ... | 0.031450 | 0.873505 | 0.899754 | 0.884876 |
| **VUG** | 0.241500 | 0.205675 | 0.478390 | 0.100976 | 0.434231 | 0.913640 | 0.450710 | 0.122682 | 0.285256 | 0.982080 | ... | 0.102638 | 0.959322 | 0.526189 | 0.908985 |

25 rows × 25 columns

In [32]:
```
heatmap(corr_matrix, fmt='.2f', cmap='coolwarm')
plt.title('covarriance heatmap')
plt.show()
```




## Getting market lambda

SPY is the market benchmark, which volatility is 0.175851.

In [33]:
```
excess_Returns_bm = returns_bm['SPY']-(average_rf)/252
daily_market_sr = excess_Returns_bm.mean()/excess_Returns_bm.std()
annual_market_sr = daily_market_sr*sqrt(252)
annual_market_sr
```

Out[33]:
```
0.5216800415429224
```

In [34]:
```
delta_market=annual_market_sr/(excess_Returns_bm.std()*sqrt(252))
delta_market
```

Out[34]:
```
2.966601129702064
```

## Define different Risk-aversion

In [35]:
```
risk_aversion = {'Market':delta_market/2,'Near_Kelly_Investors':0.005,'Average_Investors':1.12,'Risk_verse_Investors':3}
risk_aversion
```

Out[35]:
```
{'Market': 1.483300564851032,
 'Near_Kelly_Investors': 0.005,
 'Average_Investors': 1.12,
 'Risk_verse_Investors': 3}
```

## Equal Weighting

In [36]:
```
#Assume a portoflio composed of all assets with equal weighting
w_equal = numofasset * [1./numofasset]
w_equal = array(w_equal)[:,newaxis]
w_equal
```

Out[36]:
```
array([[0.04],
       [0.04],
       [0.04],
       [0.04],
       [0.04],
       [0.04],
       [0.04],
       [0.04],
       [0.04],
       [0.04],
       [0.04],
       [0.04],
       [0.04],
       [0.04],
       [0.04],
       [0.04],
       [0.04],
       [0.04],
       [0.04],
       [0.04],
       [0.04],
       [0.04],
       [0.04],
       [0.04],
       [0.04],
       [0.04]])
```

### the Prior return (Reverse Optimisation for equal weighting)

In [37]:
```
pi_equal = 2*risk_aversion['Market']*cov_matrix @ w_equal
#pi_equal.columns =['The Prior Return(Equal Weighting)']
pi_equal = pi_equal.to_numpy()
pi_equal
```

Out[37]:
```
array([[ 0.01011334],
       [ 0.11740277],
       [ 0.1512066 ],
       [ 0.01145506],
       [ 0.10777308],
       [ 0.05241746],
       [ 0.12910523],
       [ 0.0171757 ],
       [ 0.06394662],
       [ 0.08115439],
       [ 0.20886053],
       [ 0.12713713],
       [ 0.06100749],
       [ 0.19644072],
       [ 0.07404876],
       [ 0.00102283],
       [ 0.05804095],
       [ 0.033945  ],
       [ 0.05506842],
       [ 0.0083117 ],
       [ 0.03804112],
       [-0.15274267],
       [ 0.05109481],
       [ 0.03983962],
       [ 0.07388543]])
```

## Market Capacity

In [38]:
```
num=np.arange(len(asset_symbols))
market_cap=np.zeros(len(asset_symbols))
for i in num:
    etf = yf.Ticker(asset_symbols[i])
    market_cap[i] = etf.info.get('totalAssets')

market_cap
```

Out[38]:
```
array([2.00549532e+11, 1.55890270e+11, 6.75627827e+09, 2.91879142e+09,
       1.94692485e+10, 1.41168343e+10, 3.67796593e+10, 1.58197545e+10,
       8.40248883e+09, 7.48741248e+08, 5.93319424e+09, 5.61811680e+07,
       4.38037146e+09, 5.37600307e+09, 6.33580600e+06, 6.53983872e+08,
       1.58199024e+08, 1.18931968e+10, 5.16463329e+10, 3.26031680e+07,
       7.31421568e+08, 2.64252723e+10, 7.02153024e+08, 6.48568054e+10,
       1.57311184e+08])
```

In [39]:
```
w_market=market_cap/market_cap.sum()
w_market = array(w_market)[:,newaxis]
w_market
```

Out[39]:
```
array([[3.16094359e-01],
       [2.45705062e-01],
       [1.06488479e-02],
       [4.60042712e-03],
       [3.06862828e-02],
       [2.22501227e-02],
       [5.79699326e-02],
       [2.49341652e-02],
       [1.32435080e-02],
       [1.18012185e-03],
       [9.35155129e-03],
       [8.85494479e-05],
       [6.90408348e-03],
       [8.47333939e-03],
       [9.98612423e-06],
       [1.03077086e-03],
       [2.49343984e-04],
       [1.87453563e-02],
       [8.14019080e-02],
       [5.13871931e-05],
       [1.15282359e-03],
       [4.16499578e-02],
       [1.10669224e-03],
       [1.02223477e-01],
       [2.47944622e-04]])
```

In [90]:
```
table5_1 = pd.DataFrame(annual_returns.index)
table5_1.columns=['Assets Class(ETFs)']
table5_1['Weight_market']=w_market*100
table5_1['Weight_market'] = table5_1['Weight_market'].apply(lambda x: '{:.4f}%'.format(x))
fig, ax = plt.subplots(figsize=(12, 2))  # set size frame
ax.axis('tight')
ax.axis('off')
ax.table(cellText=table5_1.values, colLabels=table5_1.columns, loc='center')
plt.show()
```

| Assets Class(ETFs) | Weight_market |
|---|---|
| CBON | 31.6094% |
| CHIR | 24.5705% |
| CQQQ | 1.0649% |
| DBA | 0.4600% |
| ECNS | 3.0686% |
| EPS | 2.2250% |
| FXI | 5.7970% |
| IAU | 2.4934% |
| IEO | 1.3244% |
| IYW | 0.1180% |
| KWEB | 0.9352% |
| MCHI | 0.0089% |
| MTUM | 0.6904% |
| PGJ | 0.8473% |
| SCHA | 0.0010% |
| SCHO | 0.1031% |
| SCHX | 0.0249% |
| SPHD | 1.8745% |
| SPHQ | 8.1402% |
| TLT | 0.0051% |
| VHT | 0.1153% |
| VIXY | 4.1650% |
| VNQ | 0.1107% |
| VTV | 10.2223% |
| VUG | 0.0248% |

## The Prior return (Reverse Optimisation for market_cap weighting)

In [40]:
```
pi_market=2*risk_aversion['Market']*cov_matrix@w_market
#pi_market.columns =['The Prior Return(Market_Cap Weighting)']
pi_market = pi_market.to_numpy()
pi_market
```

Out[40]:
```
array([[ 0.01145027],
       [ 0.18573988],
       [ 0.12882926],
       [ 0.00877027],
       [ 0.10293075],
       [ 0.03088255],
       [ 0.11964309],
       [ 0.01705379],
       [ 0.04610663],
       [ 0.04389938],
       [ 0.17901983],
       [ 0.11547687],
       [ 0.03467408],
       [ 0.16413491],
       [ 0.04356448],
       [ 0.00047136],
       [ 0.03288022],
       [ 0.02184677],
       [ 0.03293695],
       [ 0.00065852],
       [ 0.02089315],
       [-0.08533008],
       [ 0.02987875],
       [ 0.02455419],
       [ 0.04028511]])
```

In [91]:
```
table5_2 = pd.DataFrame(annual_returns.index)
table5_2.columns=['Assets Class(ETFs)']
table5_2['Equilibrium Return']=pi_market*100
table5_2['Equilibrium Return'] = table5_2['Equilibrium Return'].apply(lambda x: '{:.4f}%'.format(x))
fig, ax = plt.subplots(figsize=(12, 2))  # set size frame
ax.axis('tight')
ax.axis('off')
```

```
ax.table(cellText=table5_2.values, colLabels=table5_2.columns, loc='center')
plt.show()
```

| Assets Class(ETFs) | Equilibrium Return |
|---|---|
| CBON | 1.1450% |
| CHIR | 18.5740% |
| CQQQ | 12.8829% |
| DBA | 0.8770% |
| ECNS | 10.2931% |
| EPS | 3.0883% |
| FXI | 11.9643% |
| IAU | 1.7054% |
| IEO | 4.6107% |
| IYW | 4.3899% |
| KWEB | 17.9020% |
| MCHI | 11.5477% |
| MTUM | 3.4674% |
| PGJ | 16.4135% |
| SCHA | 4.3564% |
| SCHO | 0.0471% |
| SCHX | 3.2880% |
| SPHD | 2.1847% |
| SPHQ | 3.2937% |
| TLT | 0.0659% |
| VHT | 2.0893% |
| VIXY | -8.5330% |
| VNQ | 2.9879% |
| VTV | 2.4554% |
| VUG | 4.0285% |

## View

In [41]:
```
def views():
    p = np.array(
        [
            [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0],
            [0,0,0,0,0,0,0,0,0,-1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0],
            [0.125,0.125,0.125,0,0.125,0,0.125,0,-1,0,0.125,0.125,0,0.125,0,0,0,0,0,0,0,0,0,0]
        ]
    )
    q = np.array([0.03, 0.05, 0.07])
    q = array(q)[:,newaxis]
    return p,q
```

In [42]:
```
P,Q =views()
P
```

Out[42]:
```
array([[ 0.   ,  0.   ,  0.   ,  0.   ,  0.   ,  0.   ,  0.   ,  0.   ,
         0.   ,  0.   ,  0.   ,  0.   ,  0.   ,  0.   ,  0.   ,  0.   ,
         0.   ,  0.   ,  0.   ,  0.   ,  0.   ,  0.   ,  0.   ,  1.   ,
         0.   ],
       [ 0.   ,  0.   ,  0.   ,  0.   ,  0.   ,  0.   ,  0.   ,  0.   ,
         0.   , -1.   ,  0.   ,  0.   ,  0.   ,  0.   ,  0.   ,  0.   ,
         0.   ,  0.   ,  0.   ,  0.   ,  1.   ,  0.   ,  0.   ,  0.   ,
         0.   ],
       [ 0.125,  0.125,  0.125,  0.   ,  0.125,  0.   ,  0.125,  0.   ,
        -1.   ,  0.   ,  0.125,  0.125,  0.   ,  0.125,  0.   ,  0.   ,
         0.   ,  0.   ,  0.   ,  0.   ,  0.   ,  0.   ,  0.   ,  0.   ,
         0.   ]])
```

## Tau

In [43]:
```
TAU = 0.04
tau_vec = np.array([0.04,0.04,0.04])
print(tau_vec)
tau_vec = np.diag(tau_vec)
print(tau_vec)
```

```
[0.04 0.04 0.04]
[[0.04 0.   0.  ]
 [0.   0.04 0.  ]
 [0.   0.   0.04]]
```

## Error Covariance

In [44]:
```
omega = multi_dot([P, cov_matrix, P.T])
omega =np.diag(np.diag(omega))*tau_vec
omega
```

Out[44]:
```
array([[0.00086541, 0.        , 0.        ],
       [0.        , 0.00157967, 0.        ],
       [0.        , 0.        , 0.00723898]])
```

## Calculate the black litterman return(the posterior)

In [45]:
```
def cal_posterior_return(pi: np.array, covariance_matrix: np.matrix, P:np.matrix, Q: np.array, omega: np.matrix):

    """
    params pi: The equilibrium return.
    params cov_matrix: The covariance matrix.
    params P: The matrix that identifies the assets involved in the views.
    params Q: The view vector.
    params omega: The diagonal covariance matrix of error terms.
    return The posterior return vector
    """

    part1 = multi_dot([
        covariance_matrix,
        P.T,
        inv(TAU * multi_dot([P, covariance_matrix, P.T]) + omega)
    ])
```

```
    part2 = Q - np.dot(P, pi)
    posterior_return = pi + TAU * np.dot(part1, part2)
    return posterior_return
```

In [69]:
```
post_bl_return= cal_posterior_return(pi_market,cov_matrix,P,Q,omega)
#post_bl_return.columns =['The Posterior Return']
post_bl_return
```

Out[69]:
```
array([[ 0.01040752],
       [ 0.17855663],
       [ 0.10607036],
       [ 0.00824257],
       [ 0.0900493 ],
       [ 0.01962731],
       [ 0.10422637],
       [ 0.01620726],
       [ 0.04318977],
       [ 0.0104257 ],
       [ 0.14962731],
       [ 0.09953031],
       [ 0.02164131],
       [ 0.13407708],
       [ 0.03108727],
       [ 0.00037883],
       [ 0.0186495 ],
       [ 0.0237532 ],
       [ 0.02066323],
       [-0.00125991],
       [ 0.02253642],
       [-0.050748  ],
       [ 0.02504162],
       [ 0.02329869],
       [ 0.01425312]])
```

### weight with no constraints

In [70]:
```
def cal_posterior_weight(risk_aversion, post_return, covariance_matrix):
    params=1/(2*risk_aversion)
    post_weight=params*np.dot(inv(covariance_matrix),post_return)
    return post_weight
```

In [71]:
```
post_bl_weight= cal_posterior_weight(risk_aversion['Average_Investors'],post_bl_return,cov_matrix)
post_bl_weight
```

Out[71]:
```
array([[ 4.19898541e-01],
       [ 3.26676679e-01],
       [ 1.53739874e-02],
       [ 6.09269299e-03],
       [ 4.19110753e-02],
       [ 2.94675175e-02],
       [ 7.80448728e-02],
       [ 3.30221977e-02],
       [ 7.37206464e-03],
       [-4.26750690e-01],
       [ 1.36558796e-02],
       [ 1.38818684e-03],
       [ 9.14359904e-03],
       [ 1.24927972e-02],
       [ 1.32253783e-05],
       [ 1.36512768e-03],
       [ 3.30225064e-04],
       [ 2.48258907e-02],
       [ 1.07806693e-01],
       [ 6.80559397e-05],
       [ 4.29840386e-01],
       [ 5.51601838e-02],
       [ 1.46567609e-03],
       [ 2.73659661e-01],
       [ 3.28371784e-04]])
```

## Optimisation

### Global Minimum Variance Portfolio

In [72]:
```
def optimize_gmvp(cov_matrix):
    num_assets = len(cov_matrix)
    weights_GMVP = cp.Variable(num_assets)
    risk = cp.quad_form(weights_GMVP, cov_matrix)
    objective = cp.Minimize(risk)
    constraints =[]
    constraints.append(cp.sum(weights_GMVP) == 1)
    constraints.append(weights_GMVP >=0)
    constraints.append(weights_GMVP <=1)
    problem = cp.Problem(objective, constraints)
    problem.solve()

    if problem.status == 'optimal':
        optimal_weights_GMVP = weights_GMVP.value.round(16)+0.0
        optimal_weights_GMVP = optimal_weights_GMVP.reshape(-1,1)
        print("Optimal Weights:", optimal_weights_GMVP)
        return optimal_weights_GMVP

    else:
        return none
```

In [73]:
```
optimal_weights_GMVP = optimize_gmvp(cov_matrix)
```

```
Optimal Weights: [[3.64328430e-02]
 [5.47155326e-04]
 [0.00000000e+00]
 [1.97087024e-02]
 [0.00000000e+00]
 [0.00000000e+00]
 [0.00000000e+00]
 [0.00000000e+00]
 [0.00000000e+00]
 [3.95922052e-03]
 [0.00000000e+00]
 [0.00000000e+00]
 [0.00000000e+00]
 [9.91658206e-03]
 [0.00000000e+00]
 [0.00000000e+00]
 [9.16259807e-01]
 [0.00000000e+00]
 [0.00000000e+00]
 [0.00000000e+00]
 [0.00000000e+00]
 [0.00000000e+00]
 [4.92423413e-03]
 [0.00000000e+00]
 [8.25145527e-03]
 [0.00000000e+00]]
```

## Markowitz Mean-Variance Portfolio

In [231…

```python
def optimize_meanvariance(returns,cov_matrix,risk_aversion):
    num_assets = len(returns)
    weights_mv = cp.Variable(num_assets)
    #reshape the original returns

    #formula
    expected_returns= weights_mv @ returns
    risk = cp.quad_form(weights_mv, cov_matrix)
    objective = cp.Maximize(expected_returns-risk_aversion*risk)
    #constraits
    constraints =[]
    constraints.append(cp.sum(weights_mv) == 1)
    constraints.append(weights_mv >=0)
    constraints.append(weights_mv <=1)
    problem = cp.Problem(objective, constraints)
    problem.solve()

    if problem.status == 'optimal':
        optimal_weights_mv = weights_mv.value.round(16)+0.0
        optimal_weights_mv = optimal_weights_mv.reshape(-1,1)

        return optimal_weights_mv

    else:
        return none
```

In [75]:

```python
optimal_weights_mv = optimize_meanvariance(post_bl_return,cov_matrix,risk_aversion['Average_Investors'])
```

```
Optimal Weights: [[0.        ]
 [0.35155319]
 [0.        ]
 [0.        ]
 [0.0182441 ]
 [0.        ]
 [0.08341698]
 [0.00313612]
 [0.03652592]
 [0.        ]
 [0.        ]
 [0.01158331]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.17932221]
 [0.        ]
 [0.        ]
 [0.23833823]
 [0.07787994]
 [0.        ]
 [0.        ]
 [0.        ]]
```

## Maximum Sharpe Ratio Portfolio

In [53]:

```python
def optimize_maxsharpe(returns,cov_matrix,riskfree):
    num_assets = len(returns)
    weights_ms = cp.Variable(num_assets)
    #epsilon = 1e-6
    #cov_matrix += epsilon * np.eye(num_assets)
    #formula
    expected_returns= weights_ms @ returns
    #risk = cp.quad_form(weights_ms, cov_matrix)
    risk = cp.norm(cp.sqrt(cov_matrix) @ weights_ms)
    sharpe_ratio = (expected_returns-riskfree)/risk
    objective = cp.Minimize(-1 / cp.square(sharpe_ratio))
    #constraits
    constraints =[]
    constraints.append(cp.sum(weights_ms) == 1)
    #constraints.append(weights_ms >=0)
    #constraints.append(weights_ms <=1)
    problem = cp.Problem(objective, constraints)
    problem.solve(solver=cp.SCS)
```

```
    if problem.status == 'optimal':
        optimal_weights_ms = weights_ms.value.round(16)+0.0
        optimal_weights_ms = optimal_weights_ms.reshape(-1,1)
        print("Optimal Weights:", optimal_weights_ms)
        return optimal_weights_ms

    else:
        return none
```

In [63]:
```
#optimal_weights_sr = optimize_maxsharpe(post_bl_return,cov_matrix,average_rf)
#optimal_weights_sr
```

In [55]:
```
def sharpe_ratio(weights, expected_returns, cov_matrix, risk_free_rate):
    expected_return = np.sum(weights * expected_returns)
    volatility = np.sqrt(np.dot(weights.T, np.dot(cov_matrix, weights)))
    return (expected_return - risk_free_rate) / volatility
```

In [81]:
```
num_assets = len(asset_symbols)
weights_maxsr = np.random.rand(num_assets)
weights_maxsr /= np.sum(weights_maxsr)

iterations = 1000
learning_rate = 0.01

for i in range(iterations):
    gradient = np.zeros(num_assets)
    for j in range(num_assets):
        delta = 1e-5
        weights_up = np.copy(weights_maxsr)
        weights_up[j] += delta
        gradient[j] = (sharpe_ratio(weights_up, post_bl_return,cov_matrix,average_rf) -
                       sharpe_ratio(weights_maxsr, post_bl_return,cov_matrix,average_rf)) / delta
    weights_maxsr += learning_rate * gradient
    weights_maxsr = np.maximum(weights_maxsr, 0)
    weights_maxsr /= np.sum(weights_maxsr)

print("Optimal Weights:", weights_maxsr)
```

```
Optimal Weights: [0.0642856  0.02780961 0.02848576 0.05399182 0.03565834 0.04839909
 0.02910223 0.03469657 0.02381401 0.0413295  0.02964714 0.0317156
 0.0598815  0.02926967 0.03974181 0.06758912 0.04749669 0.0430333
 0.04712653 0.038102   0.05055217 0.         0.03353917 0.05097634
 0.04375643]
```

In [82]:
```
weights_maxsr = weights_maxsr[:,newaxis]
weights_maxsr
```

Out[82]:
```
array([[0.0642856 ],
       [0.02780961],
       [0.02848576],
       [0.05399182],
       [0.03565834],
       [0.04839909],
       [0.02910223],
       [0.03469657],
       [0.02381401],
       [0.0413295 ],
       [0.02964714],
       [0.0317156 ],
       [0.0598815 ],
       [0.02926967],
       [0.03974181],
       [0.06758912],
       [0.04749669],
       [0.0430333 ],
       [0.04712653],
       [0.038102  ],
       [0.05055217],
       [0.        ],
       [0.03353917],
       [0.05097634],
       [0.04375643]])
```

## Results and visualization

### With no contraints

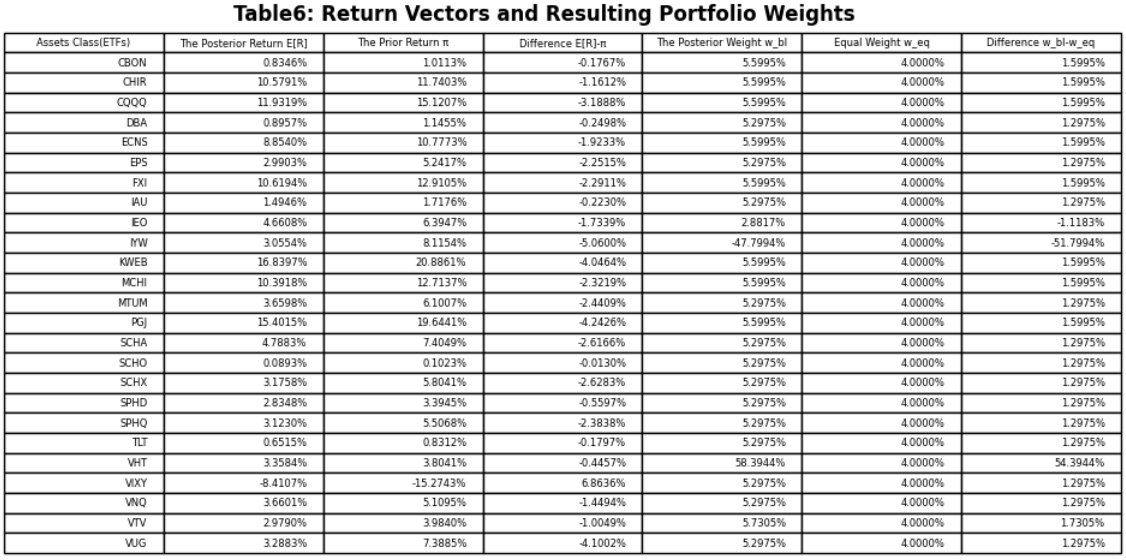
In [58]:
```
table6 = pd.DataFrame(annual_returns.index)
table6.columns=['Assets Class(ETFs)']
table6['The Posterior Return E[R]']=post_bl_return*100
table6['The Posterior Return E[R]'] = table6['The Posterior Return E[R]'].apply(lambda x: '{:.4f}%'.format(x))

table6['The Prior Return π']=pi_equal*100
table6['The Prior Return π'] = table6['The Prior Return π'].apply(lambda x: '{:.4f}%'.format(x))

table6['Difference E[R]-π']=(post_bl_return-pi_equal)*100
table6['Difference E[R]-π'] = table6['Difference E[R]-π'].apply(lambda x: '{:.4f}%'.format(x))

table6['The Posterior Weight w_bl']=post_bl_weight*100
table6['The Posterior Weight w_bl'] = table6['The Posterior Weight w_bl'].apply(lambda x: '{:.4f}%'.format(x))

table6['Equal Weight w_eq']=w_equal*100
table6['Equal Weight w_eq'] = table6['Equal Weight w_eq'].apply(lambda x: '{:.4f}%'.format(x))

table6['Difference w_bl-w_eq']=(post_bl_weight-w_equal)*100
table6['Difference w_bl-w_eq'] = table6['Difference w_bl-w_eq'].apply(lambda x: '{:.4f}%'.format(x))

table6
```

Out[58]:

| | Assets Class(ETFs) | The Posterior Return E[R] | The Prior Return π | Difference E[R]-π | The Posterior Weight w_bl | Equal Weight w_eq | Difference w_bl-w_eq |
|---|---|---|---|---|---|---|---|
| 0 | CBON | 0.8346% | 1.0113% | -0.1767% | 5.5995% | 4.0000% | 1.5995% |
| 1 | CHIR | 10.5791% | 11.7403% | -1.1612% | 5.5995% | 4.0000% | 1.5995% |
| 2 | CQQQ | 11.9319% | 15.1207% | -3.1888% | 5.5995% | 4.0000% | 1.5995% |
| 3 | DBA | 0.8957% | 1.1455% | -0.2498% | 5.2975% | 4.0000% | 1.2975% |
| 4 | ECNS | 8.8540% | 10.7773% | -1.9233% | 5.5995% | 4.0000% | 1.5995% |
| 5 | EPS | 2.9903% | 5.2417% | -2.2515% | 5.2975% | 4.0000% | 1.2975% |
| 6 | FXI | 10.6194% | 12.9105% | -2.2911% | 5.5995% | 4.0000% | 1.5995% |
| 7 | IAU | 1.4946% | 1.7176% | -0.2230% | 5.2975% | 4.0000% | 1.2975% |
| 8 | IEO | 4.6608% | 6.3947% | -1.7339% | 2.8817% | 4.0000% | -1.1183% |
| 9 | IYW | 3.0554% | 8.1154% | -5.0600% | -47.7994% | 4.0000% | -51.7994% |
| 10 | KWEB | 16.8397% | 20.8861% | -4.0464% | 5.5995% | 4.0000% | 1.5995% |
| 11 | MCHI | 10.3918% | 12.7137% | -2.3219% | 5.5995% | 4.0000% | 1.5995% |
| 12 | MTUM | 3.6598% | 6.1007% | -2.4409% | 5.2975% | 4.0000% | 1.2975% |
| 13 | PGJ | 15.4015% | 19.6441% | -4.2426% | 5.5995% | 4.0000% | 1.5995% |
| 14 | SCHA | 4.7883% | 7.4049% | -2.6166% | 5.2975% | 4.0000% | 1.2975% |
| 15 | SCHO | 0.0893% | 0.1023% | -0.0130% | 5.2975% | 4.0000% | 1.2975% |
| 16 | SCHX | 3.1758% | 5.8041% | -2.6283% | 5.2975% | 4.0000% | 1.2975% |
| 17 | SPHD | 2.8348% | 3.3945% | -0.5597% | 5.2975% | 4.0000% | 1.2975% |
| 18 | SPHQ | 3.1230% | 5.5068% | -2.3838% | 5.2975% | 4.0000% | 1.2975% |
| 19 | TLT | 0.6515% | 0.8312% | -0.1797% | 5.2975% | 4.0000% | 1.2975% |
| 20 | VHT | 3.3584% | 3.8041% | -0.4457% | 58.3944% | 4.0000% | 54.3944% |
| 21 | VIXY | -8.4107% | -15.2743% | 6.8636% | 5.2975% | 4.0000% | 1.2975% |
| 22 | VNQ | 3.6601% | 5.1095% | -1.4494% | 5.2975% | 4.0000% | 1.2975% |
| 23 | VTV | 2.9790% | 3.9840% | -1.0049% | 5.7305% | 4.0000% | 1.7305% |
| 24 | VUG | 3.2883% | 7.3885% | -4.1002% | 5.2975% | 4.0000% | 1.2975% |

In [68]:

```
fig, ax = plt.subplots(figsize=(12, 2))  # set size frame
plt.suptitle('Table6: Return Vectors and Resulting Portfolio Weights', fontweight='bold',y=1.7)
ax.axis('tight')
ax.axis('off')
ax.table(cellText=table6.values, colLabels=table6.columns, loc='center')
plt.show()
```




In [85]:

```
table7 = pd.DataFrame(annual_returns.index)
table7.columns=['Assets Class(ETFs)']
table7['The Posterior Return E[R]']=post_bl_return*100
table7['The Posterior Return E[R]'] = table7['The Posterior Return E[R]'].apply(lambda x: '{:.4f}%'.format(x))

table7['The Prior Return π']=pi_market*100
table7['The Prior Return π'] = table7['The Prior Return π'].apply(lambda x: '{:.4f}%'.format(x))

table7['Difference E[R]-π']=(post_bl_return-pi_market)*100
table7['Difference E[R]-π'] = table7['Difference E[R]-π'].apply(lambda x: '{:.4f}%'.format(x))

table7['The Posterior Weight w_bl']=post_bl_weight*100
table7['The Posterior Weight w_bl'] = table7['The Posterior Weight w_bl'].apply(lambda x: '{:.4f}%'.format(x))

table7['Market_Cap Weight w_mrk']=w_market*100
table7['Market_Cap Weight w_mrk'] = table7['Market_Cap Weight w_mrk'].apply(lambda x: '{:.4f}%'.format(x))

table7['Difference w_bl-w_mrk']=(post_bl_weight-w_market)*100
table7['Difference w_bl-w_mrk'] = table7['Difference w_bl-w_mrk'].apply(lambda x: '{:.4f}%'.format(x))

table7
```

Out[85]:

| | Assets Class(ETFs) | The Posterior Return E[R] | The Prior Return π | Difference E[R]-π | The Posterior Weight w_bl | Market_Cap Weight w_mrk | Difference w_bl-w_mrk |
|---|---|---|---|---|---|---|---|
| 0 | CBON | 1.0408% | 1.1450% | -0.1043% | 41.9899% | 31.6094% | 10.3804% |
| 1 | CHIR | 17.8557% | 18.5740% | -0.7183% | 32.6677% | 24.5705% | 8.0972% |
| 2 | CQQQ | 10.6070% | 12.8829% | -2.2759% | 1.5374% | 1.0649% | 0.4725% |
| 3 | DBA | 0.8243% | 0.8770% | -0.0528% | 0.6093% | 0.4600% | 0.1492% |
| 4 | ECNS | 9.0049% | 10.2931% | -1.2881% | 4.1911% | 3.0686% | 1.1225% |
| 5 | EPS | 1.9627% | 3.0883% | -1.1255% | 2.9468% | 2.2250% | 0.7217% |
| 6 | FXI | 10.4226% | 11.9643% | -1.5417% | 7.8045% | 5.7970% | 2.0075% |
| 7 | IAU | 1.6207% | 1.7054% | -0.0847% | 3.3022% | 2.4934% | 0.8088% |
| 8 | IEO | 4.3190% | 4.6107% | -0.2917% | 0.7372% | 1.3244% | -0.5871% |
| 9 | IYW | 1.0426% | 4.3899% | -3.3474% | -42.6751% | 0.1180% | -42.7931% |
| 10 | KWEB | 14.9627% | 17.9020% | -2.9393% | 1.3656% | 0.9352% | 0.4304% |
| 11 | MCHI | 9.9530% | 11.5477% | -1.5947% | 0.1388% | 0.0089% | 0.1300% |
| 12 | MTUM | 2.1641% | 3.4674% | -1.3033% | 0.9144% | 0.6904% | 0.2240% |
| 13 | PGJ | 13.4077% | 16.4135% | -3.0058% | 1.2493% | 0.8473% | 0.4019% |
| 14 | SCHA | 3.1087% | 4.3564% | -1.2477% | 0.0013% | 0.0010% | 0.0003% |
| 15 | SCHO | 0.0379% | 0.0471% | -0.0093% | 0.1365% | 0.1031% | 0.0334% |
| 16 | SCHX | 1.8649% | 3.2880% | -1.4231% | 0.0330% | 0.0249% | 0.0081% |
| 17 | SPHD | 2.3753% | 2.1847% | 0.1906% | 2.4826% | 1.8745% | 0.6081% |
| 18 | SPHQ | 2.0663% | 3.2937% | -1.2274% | 10.7807% | 8.1402% | 2.6405% |
| 19 | TLT | -0.1260% | 0.0659% | -0.1918% | 0.0068% | 0.0051% | 0.0017% |
| 20 | VHT | 2.2536% | 2.0893% | 0.1643% | 42.9840% | 0.1153% | 42.8688% |
| 21 | VIXY | -5.0748% | -8.5330% | 3.4582% | 5.5160% | 4.1650% | 1.3510% |
| 22 | VNQ | 2.5042% | 2.9879% | -0.4837% | 0.1466% | 0.1107% | 0.0359% |
| 23 | VTV | 2.3299% | 2.4554% | -0.1256% | 27.3660% | 10.2223% | 17.1436% |
| 24 | VUG | 1.4253% | 4.0285% | -2.6032% | 0.0328% | 0.0248% | 0.0080% |

In [87]:
```
fig, ax = plt.subplots(figsize=(12, 2))  # set size frame
plt.suptitle('Table7: Return Vectors and Resulting Portfolio Weights(Market_cap)', fontweight='bold',y=1.7)
ax.axis('tight')
ax.axis('off')
ax.table(cellText=table7.values, colLabels=table7.columns, loc='center')
plt.show()
```

**Table7: Return Vectors and Resulting Portfolio Weights(Market_cap)**

| Assets Class(ETFs) | The Posterior Return E[R] | The Prior Return π | Difference E[R]-π | The Posterior Weight w_bl | Market_Cap Weight w_mrk | Difference w_bl-w_mrk |
|---|---|---|---|---|---|---|
| CBON | 1.0408% | 1.1450% | -0.1043% | 41.9899% | 31.6094% | 10.3804% |
| CHIR | 17.8557% | 18.5740% | -0.7183% | 32.6677% | 24.5705% | 8.0972% |
| CQQQ | 10.6070% | 12.8829% | -2.2759% | 1.5374% | 1.0649% | 0.4725% |
| DBA | 0.8243% | 0.8770% | -0.0528% | 0.6093% | 0.4600% | 0.1492% |
| ECNS | 9.0049% | 10.2931% | -1.2881% | 4.1911% | 3.0686% | 1.1225% |
| EPS | 1.9627% | 3.0883% | -1.1255% | 2.9468% | 2.2250% | 0.7217% |
| FXI | 10.4226% | 11.9643% | -1.5417% | 7.8045% | 5.7970% | 2.0075% |
| IAU | 1.6207% | 1.7054% | -0.0847% | 3.3022% | 2.4934% | 0.8088% |
| IEO | 4.3190% | 4.6107% | -0.2917% | 0.7372% | 1.3244% | -0.5871% |
| IYW | 1.0426% | 4.3899% | -3.3474% | -42.6751% | 0.1180% | -42.7931% |
| KWEB | 14.9627% | 17.9020% | -2.9393% | 1.3656% | 0.9352% | 0.4304% |
| MCHI | 9.9530% | 11.5477% | -1.5947% | 0.1388% | 0.0089% | 0.1300% |
| MTUM | 2.1641% | 3.4674% | -1.3033% | 0.9144% | 0.6904% | 0.2240% |
| PGJ | 13.4077% | 16.4135% | -3.0058% | 1.2493% | 0.8473% | 0.4019% |
| SCHA | 3.1087% | 4.3564% | -1.2477% | 0.0013% | 0.0010% | 0.0003% |
| SCHO | 0.0379% | 0.0471% | -0.0093% | 0.1365% | 0.1031% | 0.0334% |
| SCHX | 1.8649% | 3.2880% | -1.4231% | 0.0330% | 0.0249% | 0.0081% |
| SPHD | 2.3753% | 2.1847% | 0.1906% | 2.4826% | 1.8745% | 0.6081% |
| SPHQ | 2.0663% | 3.2937% | -1.2274% | 10.7807% | 8.1402% | 2.6405% |
| TLT | -0.1260% | 0.0659% | -0.1918% | 0.0068% | 0.0051% | 0.0017% |
| VHT | 2.2536% | 2.0893% | 0.1643% | 42.9840% | 0.1153% | 42.8688% |
| VIXY | -5.0748% | -8.5330% | 3.4582% | 5.5160% | 4.1650% | 1.3510% |
| VNQ | 2.5042% | 2.9879% | -0.4837% | 0.1466% | 0.1107% | 0.0359% |
| VTV | 2.3299% | 2.4554% | -0.1256% | 27.3660% | 10.2223% | 17.1436% |
| VUG | 1.4253% | 4.0285% | -2.6032% | 0.0328% | 0.0248% | 0.0080% |

In [122…
```
table8 = pd.DataFrame(annual_returns.index)
table8.columns=['Assets Class(ETFs)']

table8['Market_Cap\nWeight w_mrk']=w_market*100
table8['Market_Cap\nWeight w_mrk'] = table8['Market_Cap\nWeight w_mrk'].apply(lambda x: '{:.4f}%'.format(x))

table8['The BL Weight\n(with no contraints)']=post_bl_weight*100
table8['The BL Weight\n(with no contraints)'] = table8['The BL Weight\n(with no contraints)'].apply(lambda x: '{:.4f}%'.format(x))

table8['GMVP\nWeight']=optimal_weights_GMVP*100
table8['GMVP\nWeight'] = table8['GMVP\nWeight'].apply(lambda x: '{:.4f}%'.format(x))

table8['Mean-Varriance\nWeight']=optimal_weights_mv*100
table8['Mean-Varriance\nWeight'] = table8['Mean-Varriance\nWeight'].apply(lambda x: '{:.4f}%'.format(x))

table8['Maximum_SR\nWeight']=weights_maxsr*100
table8['Maximum_SR\nWeight'] = table8['Maximum_SR\nWeight'].apply(lambda x: '{:.4f}%'.format(x))

table8
```

Out[122]:

| | Assets Class(ETFs) | Market_Cap\nWeight w_mrk | The BL Weight\n(with no contraints) | GMVP\nWeight | Mean-Varriance\nWeight | Maximum_SR\nWeight |
|---|---|---|---|---|---|---|
| **0** | CBON | 31.6094% | 41.9899% | 3.6433% | 0.0000% | 6.4286% |
| **1** | CHIR | 24.5705% | 32.6677% | 0.0547% | 35.1553% | 2.7810% |
| **2** | CQQQ | 1.0649% | 1.5374% | 0.0000% | 0.0000% | 2.8486% |
| **3** | DBA | 0.4600% | 0.6093% | 1.9709% | 0.0000% | 5.3992% |
| **4** | ECNS | 3.0686% | 4.1911% | 0.0000% | 1.8244% | 3.5658% |
| **5** | EPS | 2.2250% | 2.9468% | 0.0000% | 0.0000% | 4.8399% |
| **6** | FXI | 5.7970% | 7.8045% | 0.0000% | 8.3417% | 2.9102% |
| **7** | IAU | 2.4934% | 3.3022% | 0.0000% | 0.3136% | 3.4697% |
| **8** | IEO | 1.3244% | 0.7372% | 0.3959% | 3.6526% | 2.3814% |
| **9** | IYW | 0.1180% | -42.6751% | 0.0000% | 0.0000% | 4.1330% |
| **10** | KWEB | 0.9352% | 1.3656% | 0.0000% | 0.0000% | 2.9647% |
| **11** | MCHI | 0.0089% | 0.1388% | 0.0000% | 1.1583% | 3.1716% |
| **12** | MTUM | 0.6904% | 0.9144% | 0.9917% | 0.0000% | 5.9882% |
| **13** | PGJ | 0.8473% | 1.2493% | 0.0000% | 0.0000% | 2.9270% |
| **14** | SCHA | 0.0010% | 0.0013% | 0.0000% | 0.0000% | 3.9742% |
| **15** | SCHO | 0.1031% | 0.1365% | 91.6260% | 0.0000% | 6.7589% |
| **16** | SCHX | 0.0249% | 0.0330% | 0.0000% | 0.0000% | 4.7497% |
| **17** | SPHD | 1.8745% | 2.4826% | 0.0000% | 17.9322% | 4.3033% |
| **18** | SPHQ | 8.1402% | 10.7807% | 0.0000% | 0.0000% | 4.7127% |
| **19** | TLT | 0.0051% | 0.0068% | 0.0000% | 0.0000% | 3.8102% |
| **20** | VHT | 0.1153% | 42.9840% | 0.0000% | 23.8338% | 5.0552% |
| **21** | VIXY | 4.1650% | 5.5160% | 0.4924% | 7.7880% | 0.0000% |
| **22** | VNQ | 0.1107% | 0.1466% | 0.0000% | 0.0000% | 3.3539% |
| **23** | VTV | 10.2223% | 27.3660% | 0.8251% | 0.0000% | 5.0976% |
| **24** | VUG | 0.0248% | 0.0328% | 0.0000% | 0.0000% | 4.3756% |

In [123…

```
fig, ax = plt.subplots(figsize=(12, 2))  # set size frame
plt.suptitle('Table8: Three Type of Optimisation Weights(Market_cap)', fontweight='bold',y=2.8)
ax.axis('tight')
ax.axis('off')

table = ax.table(cellText=table8.values, colLabels=table8.columns, loc='center')
table.scale(1, 2)
plt.show()
```

**Table8: Three Type of Optimisation Weights(Market_cap)**

| Assets Class(ETFs) | Market_Cap Weight w_mrk | The BL Weight (with no contraints) | GMVP Weight | Mean-Varriance Weight | Maximum_SR Weight |
|---|---|---|---|---|---|
| CBON | 31.6094% | 41.9899% | 3.6433% | 0.0000% | 6.4286% |
| CHIR | 24.5705% | 32.6677% | 0.0547% | 35.1553% | 2.7810% |
| CQQQ | 1.0649% | 1.5374% | 0.0000% | 0.0000% | 2.8486% |
| DBA | 0.4600% | 0.6093% | 1.9709% | 0.0000% | 5.3992% |
| ECNS | 3.0686% | 4.1911% | 0.0000% | 1.8244% | 3.5658% |
| EPS | 2.2250% | 2.9468% | 0.0000% | 0.0000% | 4.8399% |
| FXI | 5.7970% | 7.8045% | 0.0000% | 8.3417% | 2.9102% |
| IAU | 2.4934% | 3.3022% | 0.0000% | 0.3136% | 3.4697% |
| IEO | 1.3244% | 0.7372% | 0.3959% | 3.6526% | 2.3814% |
| IYW | 0.1180% | -42.6751% | 0.0000% | 0.0000% | 4.1330% |
| KWEB | 0.9352% | 1.3656% | 0.0000% | 0.0000% | 2.9647% |
| MCHI | 0.0089% | 0.1388% | 0.0000% | 1.1583% | 3.1716% |
| MTUM | 0.6904% | 0.9144% | 0.9917% | 0.0000% | 5.9882% |
| PGJ | 0.8473% | 1.2493% | 0.0000% | 0.0000% | 2.9270% |
| SCHA | 0.0010% | 0.0013% | 0.0000% | 0.0000% | 3.9742% |
| SCHO | 0.1031% | 0.1365% | 91.6260% | 0.0000% | 6.7589% |
| SCHX | 0.0249% | 0.0330% | 0.0000% | 0.0000% | 4.7497% |
| SPHD | 1.8745% | 2.4826% | 0.0000% | 17.9322% | 4.3033% |
| SPHQ | 8.1402% | 10.7807% | 0.0000% | 0.0000% | 4.7127% |
| TLT | 0.0051% | 0.0068% | 0.0000% | 0.0000% | 3.8102% |
| VHT | 0.1153% | 42.9840% | 0.0000% | 23.8338% | 5.0552% |
| VIXY | 4.1650% | 5.5160% | 0.4924% | 7.7880% | 0.0000% |
| VNQ | 0.1107% | 0.1466% | 0.0000% | 0.0000% | 3.3539% |
| VTV | 10.2223% | 27.3660% | 0.8251% | 0.0000% | 5.0976% |
| VUG | 0.0248% | 0.0328% | 0.0000% | 0.0000% | 4.3756% |

## Different risk aversion

Kelly

In [128…
```
pi_market_kelly=2*risk_aversion['Near_Kelly_Investors']*cov_matrix@w_market
#pi_market_kelly.columns =['The Prior Return(Market_Cap Weighting)']
pi_market_kelly = pi_market_kelly.to_numpy()
post_bl_return_kelly= cal_posterior_return(pi_market_kelly,cov_matrix,P,Q,omega)
#post_bl_return.columns =['The Posterior Return']
post_bl_weight_kelly= cal_posterior_weight(risk_aversion['Near_Kelly_Investors'],post_bl_return_kelly,cov_matrix)
post_bl_weight_kelly
```

Out[128]:
```
array([[ 3.58275195e+00],
       [ 3.51236265e+00],
       [ 3.27730644e+00],
       [ 4.60042712e-03],
       [ 3.29734387e+00],
       [ 2.22501227e-02],
       [ 3.32462752e+00],
       [ 2.49341652e-02],
       [-2.61200172e+01],
       [-7.72815416e+01],
       [ 3.27600914e+00],
       [ 3.26674614e+00],
       [ 6.90408348e-03],
       [ 3.27513093e+00],
       [ 9.98612397e-06],
       [ 1.03077086e-03],
       [ 2.49343985e-04],
       [ 1.87453563e-02],
       [ 8.14019080e-02],
       [ 5.13871930e-05],
       [ 7.72838745e+01],
       [ 4.16499578e-02],
       [ 1.10669224e-03],
       [ 9.26353566e+01],
       [ 2.47944623e-04]])
```

Average

In [129…
```
pi_market_average=2*risk_aversion['Average_Investors']*cov_matrix@w_market
#pi_market_average.columns =['The Prior Return(Market_Cap Weighting)']
pi_market_average = pi_market_average.to_numpy()
post_bl_return_average= cal_posterior_return(pi_market_average,cov_matrix,P,Q,omega)
#post_bl_return_average.columns =['The Posterior Return']
post_bl_weight_average= cal_posterior_weight(risk_aversion['Average_Investors'],post_bl_return_average,cov_matrix)
post_bl_weight_average
```

Out[129]:
```
array([[ 3.20636864e-01],
       [ 2.50247567e-01],
       [ 1.51913529e-02],
       [ 4.60042712e-03],
       [ 3.52287878e-02],
       [ 2.22501227e-02],
       [ 6.25124376e-02],
       [ 2.49341652e-02],
       [-2.30965323e-02],
       [-4.06661696e-01],
       [ 1.38940563e-02],
       [ 4.63105448e-03],
       [ 6.90408348e-03],
       [ 1.30158444e-02],
       [ 9.98612423e-06],
       [ 1.03077086e-03],
       [ 2.49343983e-04],
       [ 1.87453563e-02],
       [ 8.14019080e-02],
       [ 5.13871931e-05],
       [ 4.08994641e-01],
       [ 4.16499578e-02],
       [ 1.10669224e-03],
       [ 3.08038660e-01],
       [ 2.47944623e-04]])
```

Trustee

In [130…
```
pi_market_trustee=2*risk_aversion['Risk_verse_Investors']*cov_matrix@w_market
#pi_market_trustee.columns =['The Prior Return(Market_Cap Weighting)']
pi_market_trustee = pi_market_trustee.to_numpy()
post_bl_return_trustee= cal_posterior_return(pi_market_trustee,cov_matrix,P,Q,omega)
#post_bl_return_trustee.columns =['The Posterior Return']
post_bl_weight_trustee= cal_posterior_weight(risk_aversion['Risk_verse_Investors'],post_bl_return_trustee,cov_matrix)
post_bl_weight_trustee
```

Out[130]:
```
array([[ 3.11469785e-01],
       [ 2.41080487e-01],
       [ 6.02427314e-03],
       [ 4.60042712e-03],
       [ 2.60617080e-02],
       [ 2.22501227e-02],
       [ 5.33453579e-02],
       [ 2.49341652e-02],
       [ 5.02401058e-02],
       [-1.90630643e-01],
       [ 4.72697657e-03],
       [-4.53602528e-03],
       [ 6.90408348e-03],
       [ 3.84876467e-03],
       [ 9.98612423e-06],
       [ 1.03077086e-03],
       [ 2.49343983e-04],
       [ 1.87453563e-02],
       [ 8.14019080e-02],
       [ 5.13871931e-05],
       [ 1.92963589e-01],
       [ 4.16499578e-02],
       [ 1.10669224e-03],
       [ 4.85837158e-02],
       [ 2.47944623e-04]])
```

In [131…
```
table9 = pd.DataFrame(annual_returns.index)
table9.columns=['Assets Class(ETFs)']
table9['weight_BL\nRealMarket']=post_bl_weight*100
table9['weight_BL\nRealMarket'] = table9['weight_BL\nRealMarket'].apply(lambda x: '{:.4f}%'.format(x))

table9['weight_BL\nKelly']=post_bl_weight_kelly*100
table9['weight_BL\nKelly'] = table9['weight_BL\nKelly'].apply(lambda x: '{:.4f}%'.format(x))

table9['weight_BL\nAverageMarket']=post_bl_weight_average*100
table9['weight_BL\nAverageMarket'] = table9['weight_BL\nAverageMarket'].apply(lambda x: '{:.4f}%'.format(x))


table9['weight_BL\nTrustee']=post_bl_weight_trustee*100
table9['weight_BL\nTrustee'] = table9['weight_BL\nTrustee'].apply(lambda x: '{:.4f}%'.format(x))

table9
```

Out[131]:

| | Assets Class(ETFs) | weight_BL\nRealMarket | weight_BL\nKelly | weight_BL\nAverageMarket | weight_BL\nTrustee |
|---|---|---|---|---|---|
| 0 | CBON | 41.9899% | 358.2752% | 32.0637% | 31.1470% |
| 1 | CHIR | 32.6677% | 351.2363% | 25.0248% | 24.1080% |
| 2 | CQQQ | 1.5374% | 327.7306% | 1.5191% | 0.6024% |
| 3 | DBA | 0.6093% | 0.4600% | 0.4600% | 0.4600% |
| 4 | ECNS | 4.1911% | 329.7344% | 3.5229% | 2.6062% |
| 5 | EPS | 2.9468% | 2.2250% | 2.2250% | 2.2250% |
| 6 | FXI | 7.8045% | 332.4628% | 6.2512% | 5.3345% |
| 7 | IAU | 3.3022% | 2.4934% | 2.4934% | 2.4934% |
| 8 | IEO | 0.7372% | -2612.0017% | -2.3097% | 5.0240% |
| 9 | IYW | -42.6751% | -7728.1542% | -40.6662% | -19.0631% |
| 10 | KWEB | 1.3656% | 327.6009% | 1.3894% | 0.4727% |
| 11 | MCHI | 0.1388% | 326.6746% | 0.4631% | -0.4536% |
| 12 | MTUM | 0.9144% | 0.6904% | 0.6904% | 0.6904% |
| 13 | PGJ | 1.2493% | 327.5131% | 1.3016% | 0.3849% |
| 14 | SCHA | 0.0013% | 0.0010% | 0.0010% | 0.0010% |
| 15 | SCHO | 0.1365% | 0.1031% | 0.1031% | 0.1031% |
| 16 | SCHX | 0.0330% | 0.0249% | 0.0249% | 0.0249% |
| 17 | SPHD | 2.4826% | 1.8745% | 1.8745% | 1.8745% |
| 18 | SPHQ | 10.7807% | 8.1402% | 8.1402% | 8.1402% |
| 19 | TLT | 0.0068% | 0.0051% | 0.0051% | 0.0051% |
| 20 | VHT | 42.9840% | 7728.3875% | 40.8995% | 19.2964% |
| 21 | VIXY | 5.5160% | 4.1650% | 4.1650% | 4.1650% |
| 22 | VNQ | 0.1466% | 0.1107% | 0.1107% | 0.1107% |
| 23 | VTV | 27.3660% | 9263.5357% | 30.8039% | 4.8584% |
| 24 | VUG | 0.0328% | 0.0248% | 0.0248% | 0.0248% |

In [132…

```
fig, ax = plt.subplots(figsize=(12, 2))  # set size frame
plt.suptitle('Table9: Black-Litterman Results under Different Risk Aversion Scenarios', fontweight='bold',y=2.8)
ax.axis('tight')
ax.axis('off')

table = ax.table(cellText=table9.values, colLabels=table9.columns, loc='center')
table.scale(1, 2)
plt.show()
```

**Table9: Black-Litterman Results under Different Risk Aversion Scenarios**

| Assets Class(ETFs) | weight_BL RealMarket | weight_BL Kelly | weight_BL AverageMarket | weight_BL Trustee |
|---|---|---|---|---|
| CBON | 41.9899% | 358.2752% | 32.0637% | 31.1470% |
| CHIR | 32.6677% | 351.2363% | 25.0248% | 24.1080% |
| CQQQ | 1.5374% | 327.7306% | 1.5191% | 0.6024% |
| DBA | 0.6093% | 0.4600% | 0.4600% | 0.4600% |
| ECNS | 4.1911% | 329.7344% | 3.5229% | 2.6062% |
| EPS | 2.9468% | 2.2250% | 2.2250% | 2.2250% |
| FXI | 7.8045% | 332.4628% | 6.2512% | 5.3345% |
| IAU | 3.3022% | 2.4934% | 2.4934% | 2.4934% |
| IEO | 0.7372% | -2612.0017% | -2.3097% | 5.0240% |
| IYW | -42.6751% | -7728.1542% | -40.6662% | -19.0631% |
| KWEB | 1.3656% | 327.6009% | 1.3894% | 0.4727% |
| MCHI | 0.1388% | 326.6746% | 0.4631% | -0.4536% |
| MTUM | 0.9144% | 0.6904% | 0.6904% | 0.6904% |
| PGJ | 1.2493% | 327.5131% | 1.3016% | 0.3849% |
| SCHA | 0.0013% | 0.0010% | 0.0010% | 0.0010% |
| SCHO | 0.1365% | 0.1031% | 0.1031% | 0.1031% |
| SCHX | 0.0330% | 0.0249% | 0.0249% | 0.0249% |
| SPHD | 2.4826% | 1.8745% | 1.8745% | 1.8745% |
| SPHQ | 10.7807% | 8.1402% | 8.1402% | 8.1402% |
| TLT | 0.0068% | 0.0051% | 0.0051% | 0.0051% |
| VHT | 42.9840% | 7728.3875% | 40.8995% | 19.2964% |
| VIXY | 5.5160% | 4.1650% | 4.1650% | 4.1650% |
| VNQ | 0.1466% | 0.1107% | 0.1107% | 0.1107% |
| VTV | 27.3660% | 9263.5357% | 30.8039% | 4.8584% |
| VUG | 0.0328% | 0.0248% | 0.0248% | 0.0248% |

## Different volatility(reference model vs posterior model)

for i in range(0,1.01,0.01) arange(0, 1.01, 0.01) t_vec = np.array([0.04,0.04,0.04]) print(tau_vec) tau_vec = np.diag(tau_vec) print(tau_vec) omega_tau = multi_dot([P, cov_matrix, P.T]) omega_tau=np.diag(np.diag(omega))*tau_vec omega_tau post_bl_returnt_tau= cal_posterior_return(pi_market,cov_matrix,P,Q,omega) post_bl_weight_tau= cal_posterior_weight(risk_aversion['Average_Investors'],post_bl_return_tau,cov_matrix) eight_tau[10],weight_tau[11], weight_tau[12],weight_tau[13],weight_tau[14],weight_tau[15],weight_tau[16],weight_tau[17],weight_tau[18],weight_tau[19],weight_tau[20],weight_tau[21],weight_tau

In [246…

```
omega_tau = multi_dot([P, cov_matrix, P.T])
omega_tau=np.diag(np.diag(omega_tau))
weight_tau = pd.DataFrame(annual_returns.index)
for i in np.arange(0,1,0.1):
    t_vec=np.zeros(3)
    t_vec[:]=i
    t_vec=np.diag(t_vec)
    factor=omega_tau* t_vec
    post_bl_return_tau= cal_posterior_return(pi_market,cov_matrix,P,Q,factor)
    post_bl_weight_tau= optimize_meanvariance(post_bl_return_tau,cov_matrix,risk_aversion['Average_Investors'])
    #post_bl_weight_tau=post_bl_weight_tau.reshape(-1,1)
    weight_tau[str("{:.1f}".format(i))] = post_bl_weight_tau


print(weight_tau)
```

| | 0 | 0.0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | CBON | 0.000000 | 0.098594 | 0.137050 | 0.149095 | 0.155819 | 0.158730 | 0.160407 | 0.161640 | 0.162558 | 0.163268 |
| 1 | CHIR | 0.382487 | 0.338379 | 0.332618 | 0.330873 | 0.329972 | 0.329524 | 0.329246 | 0.329047 | 0.328896 | 0.328778 |
| 2 | CQQQ | 0.000000 | 0.000000 | 0.003963 | 0.013866 | 0.018514 | 0.019495 | 0.019706 | 0.019861 | 0.019981 | 0.020077 |
| 3 | DBA | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 4 | ECNS | 0.000000 | 0.025730 | 0.031320 | 0.032626 | 0.033639 | 0.034631 | 0.035406 | 0.035985 | 0.036426 | 0.036774 |
| 5 | EPS | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 6 | FXI | 0.046888 | 0.073779 | 0.068409 | 0.071325 | 0.072295 | 0.072555 | 0.072636 | 0.072669 | 0.072696 | 0.072717 |
| 7 | IAU | 0.000000 | 0.015851 | 0.025402 | 0.030105 | 0.032653 | 0.033819 | 0.034484 | 0.034916 | 0.035246 | 0.035506 |
| 8 | IEO | 0.000000 | 0.040288 | 0.034009 | 0.029764 | 0.027350 | 0.026394 | 0.025891 | 0.025564 | 0.025309 | 0.025104 |
| 9 | IYW | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 10 | KWEB | 0.000000 | 0.000000 | 0.004852 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 11 | MCHI | 0.000000 | 0.041611 | 0.042174 | 0.029318 | 0.021536 | 0.018099 | 0.016121 | 0.014734 | 0.013680 | 0.012851 |
| 12 | MTUM | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 13 | PGJ | 0.000000 | 0.000000 | 0.003149 | 0.011768 | 0.014918 | 0.016676 | 0.017809 | 0.018600 | 0.019204 | 0.019681 |
| 14 | SCHA | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 15 | SCHO | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 16 | SCHX | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 17 | SPHD | 0.197089 | 0.138788 | 0.087062 | 0.062724 | 0.049376 | 0.048717 | 0.049670 | 0.049606 | 0.049560 | 0.049524 |
| 18 | SPHQ | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.020656 | 0.039638 | 0.053217 | 0.063558 | 0.071696 |
| 19 | TLT | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 20 | VHT | 0.299097 | 0.169085 | 0.111674 | 0.084839 | 0.070148 | 0.057821 | 0.048537 | 0.041637 | 0.036382 | 0.032247 |
| 21 | VIXY | 0.074439 | 0.057894 | 0.048219 | 0.044677 | 0.042695 | 0.042083 | 0.041826 | 0.041658 | 0.041530 | 0.041428 |
| 22 | VNQ | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000419 | 0.001433 | 0.002206 | 0.002814 |
| 23 | VTV | 0.000000 | 0.000000 | 0.070100 | 0.109021 | 0.131084 | 0.120801 | 0.108204 | 0.099430 | 0.092768 | 0.087536 |
| 24 | VUG | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |

## performance my portfolio vs bench mark

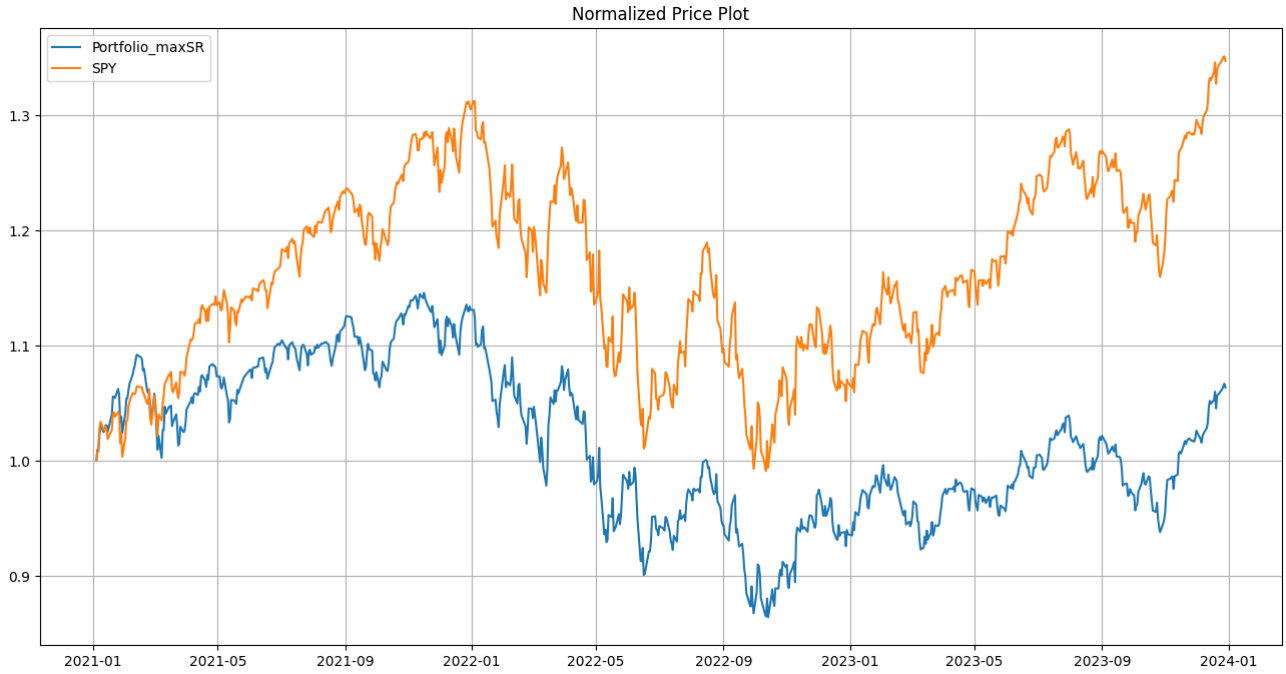
In [212…
```
weighted_log_returns = np.dot(log_returns, weights_maxsr)
spy_returns=log_returns_bm['SPY'].to_numpy()
data4=pd.DataFrame(index=log_returns.index)
data4['Portfolio_maxSR']=weighted_log_returns
data4['SPY']=spy_returns
data4.fillna(0)
data4
```

Out[212]:

| Date | Portfolio_maxSR | SPY |
|---|---|---|
| 2021-01-04 | NaN | NaN |
| 2021-01-05 | 0.010541 | 0.006863 |
| 2021-01-06 | 0.000591 | 0.005961 |
| 2021-01-07 | 0.009669 | 0.014748 |
| 2021-01-08 | 0.007533 | 0.005682 |
| ... | ... | ... |
| 2023-12-22 | -0.003446 | 0.002008 |
| 2023-12-26 | 0.004748 | 0.004214 |
| 2023-12-27 | 0.001429 | 0.001806 |
| 2023-12-28 | 0.005685 | 0.000378 |
| 2023-12-29 | -0.001703 | -0.002899 |

753 rows × 2 columns

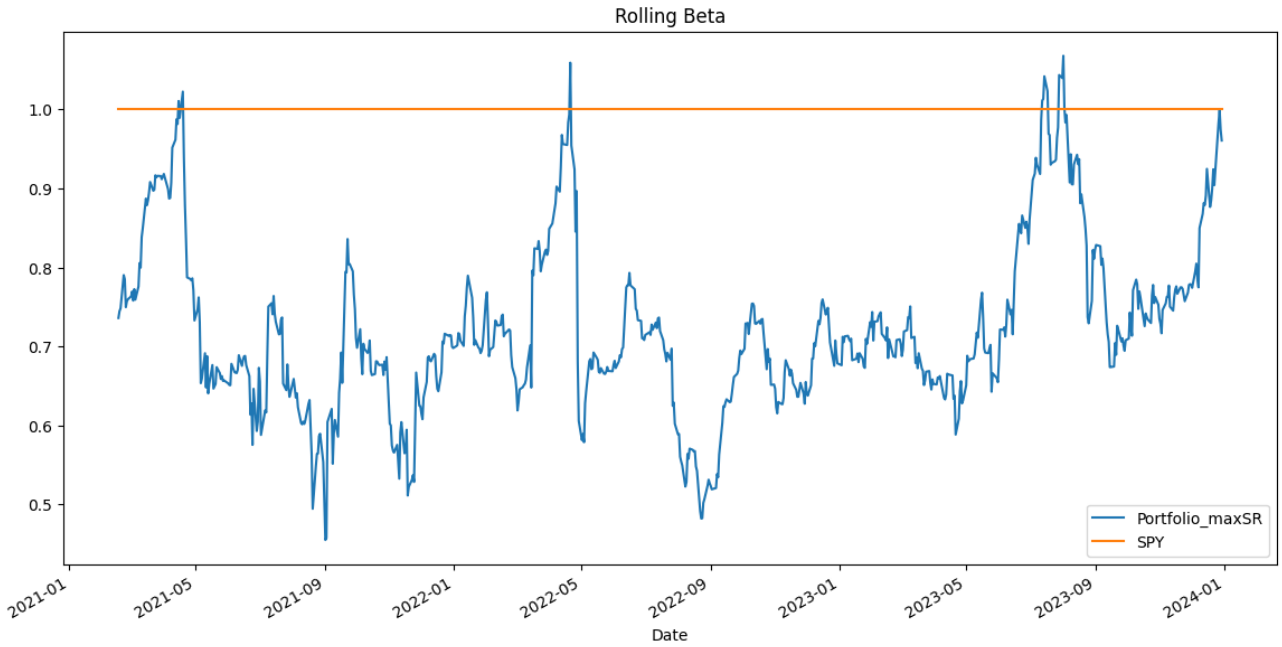
In [215…
```
weighted_close = np.dot(data, weights_maxsr)
spy_close=data2['SPY'].to_numpy()
data5=pd.DataFrame(index=log_returns.index)
data5['Portfolio_maxSR']=weighted_close
data5['SPY']=spy_close
data5.fillna(0)
data5
# Visualize the data
fig = plt.figure(figsize=(16,8))
ax = plt.axes()

ax.set_title('Normalized Price Plot')
ax.plot(data5[-753:]/data5.iloc[-753] * 1)
ax.legend(data5.columns, loc='upper left')
ax.grid(True)
```



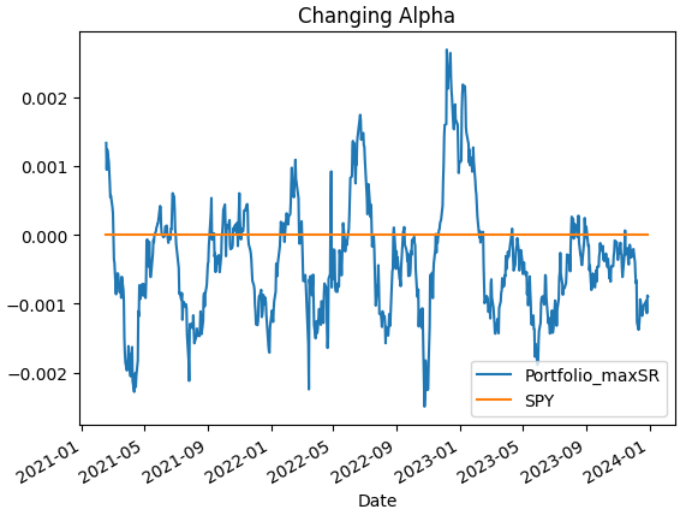

In [216…
```
returns_alphabeta_portfolio = data4


rolling_betas_portfolio = returns_alphabeta_portfolio.apply(lambda x: calculate_rolling_beta(x, returns_alphabeta_portfolio['SPY'], windo


market_excess_return = returns_alphabeta_portfolio['SPY']
#average_rf


changing_alphas_portfolio = returns_alphabeta_portfolio.apply(lambda x: calculate_changing_alpha(x, market_excess_return, rolling_betas_p

rolling_betas_portfolio['Portfolio_maxSR'].dropna().tail(), changing_alphas_portfolio['Portfolio_maxSR'].dropna().tail()
```

Out[216]:
```
(Date
 2023-12-22    0.904123
 2023-12-26    0.982751
 2023-12-27    0.999180
 2023-12-28    0.974026
 2023-12-29    0.960621
 Name: Portfolio_maxSR, dtype: float64,
 Date
 2023-12-22   -0.001014
 2023-12-26   -0.000958
 2023-12-27   -0.001137
 2023-12-28   -0.000882
 2023-12-29   -0.000895
 Name: Portfolio_maxSR, dtype: float64)
```

In [217…
```
plt.figure(figsize=(14, 7))

plt.subplot()
rolling_betas_portfolio.plot(title='Rolling Beta', ax=plt.gca())
plt.legend(loc='lower right')


plt.show()
```




In [218…
```
changing_alphas_portfolio.plot(title='Changing Alpha')
plt.legend(loc='lower right')

plt.subplot()
plt.figure(figsize=(14, 7))
plt.tight_layout()
plt.show()
```




```
<Figure size 1400x700 with 0 Axes>
```

In [ ]: