In [1]:
```python
# import all required modules
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import yfinance as yf
import random
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```python
# getting the list of securities under FnO section in NSE
csv = pd.read_csv('sos_scheme.csv')
symbol = csv['Symbol']
```

In [3]:
```python
data = pd.DataFrame()  # creating an empty dataframe

for i in symbol:       # running loop for downloading daily historical stock data of all securities from yahoo finance

    df = yf.download(f'{i}.NS',period='5y')
    df = df.rename(columns={'Adj Close':f'{i}'})  # renaming 'Adj Close' column to symbol ticker
    data[i] = df[i]                               # stacking columns of securities with adj close prices
```

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

In [66]:
```
data
```

Out[66]:

Date	AARTIIND	ABB	ABBOTINDIA	ABCAPITAL	ABFRL	ACC	ADANIENT	ADANIPORTS	ALKEM	AMBUJACEM	...	TRENT	TVSMOTOR	
2018-07-02	286.061188	1135.304688	6853.536133	129.350006	137.449997	1217.956665	66.427162	351.926178	1757.505127	177.911133	...	308.102417	538.766541	10
2018-07-03	289.538544	1127.665649	6791.241699	129.750000	139.550003	1229.673462	67.185966	351.877655	1803.816650	178.801132	...	305.048492	543.610291	11
2018-07-04	285.501434	1133.238770	6844.151367	129.050003	137.899994	1246.280151	68.102859	349.161591	1778.352417	178.890152	...	307.757629	563.227295	11
2018-07-05	283.131714	1136.073608	6865.649902	127.349998	136.399994	1282.122681	66.964653	351.053101	1794.582886	184.630646	...	305.344055	552.667969	11
2018-07-06	286.894806	1128.338257	6868.666992	126.949997	138.850006	1267.361206	70.157959	354.787750	1789.680420	184.719666	...	305.491791	558.771057	11
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
2023-06-22	524.299988	4270.500000	22554.849609	171.649994	209.250000	1831.750000	2397.250000	745.599976	3349.399902	444.850006	...	1710.699951	1326.150024	15
2023-06-23	502.799988	4239.700195	22659.300781	175.000000	203.949997	1766.900024	2233.550049	714.299988	3349.949951	425.799988	...	1724.949951	1301.250000	14
2023-06-26	508.100006	4285.350098	22841.150391	181.250000	208.300003	1792.300049	2295.600098	724.400024	3401.500000	432.049988	...	1753.650024	1290.349976	14
2023-06-27	507.450012	4294.500000	22707.800781	192.300003	210.899994	1788.599976	2284.449951	720.250000	3395.449951	433.850006	...	1742.849976	1304.199951	14
2023-06-30	506.000000	4387.799805	23330.800781	194.449997	214.649994	1808.900024	2366.000000	742.500000	3466.500000	435.799988	...	1774.050049	1332.000000	15

1233 rows × 186 columns

In [48]:
```python
stocks = random.sample(list(symbol),10)  # randomly selecting 10 stocks from the list of securities under FnO section
stocks
```

Out[48]:
```
['GLENMARK',
 'BALRAMCHIN',
 'TATACOMM',
 'HEROMOTOCO',
 'PETRONET',
 'DIVISLAB',
 'HAVELLS',
 'POWERGRID',
 'BOSCHLTD',
 'BHARTIARTL']
```

In [49]:
```python
class Portfolio_Optimization():

    def __init__(self,data,stocks):

        self.data = data
        self.stocks = stocks

    def pct_change(self):                                # calculating daily percent_change of randomly chosen stocks

        data_portfolio = self.data[self.stocks]
        return data_portfolio.pct_change().dropna()

    def corr(self):                                      # calculating correlation of daily percentage changes among the randomly chosen stoc

        data_portfolio = self.data[self.stocks]
        return data_portfolio.pct_change().dropna().corr().round(2)

    def historical_stock_Return_Pct_Returns(self):      # calculating historical annual return of randomly chosen stocks
        data_portfolio_train = self.data[self.stocks][:-252]
        return data_portfolio_train.pct_change().dropna().mean()*252

    def historical_stock_covariance(self):              # calculating covariance of daily percentage changes among the randomly chosen stock

        data_portfolio_train = self.data[self.stocks][:-252]
        return data_portfolio_train.pct_change().cov()*252
```

In [50]:
```python
port_train = data[stocks]     # selecting adj close prices of randomly chosen stocks
```

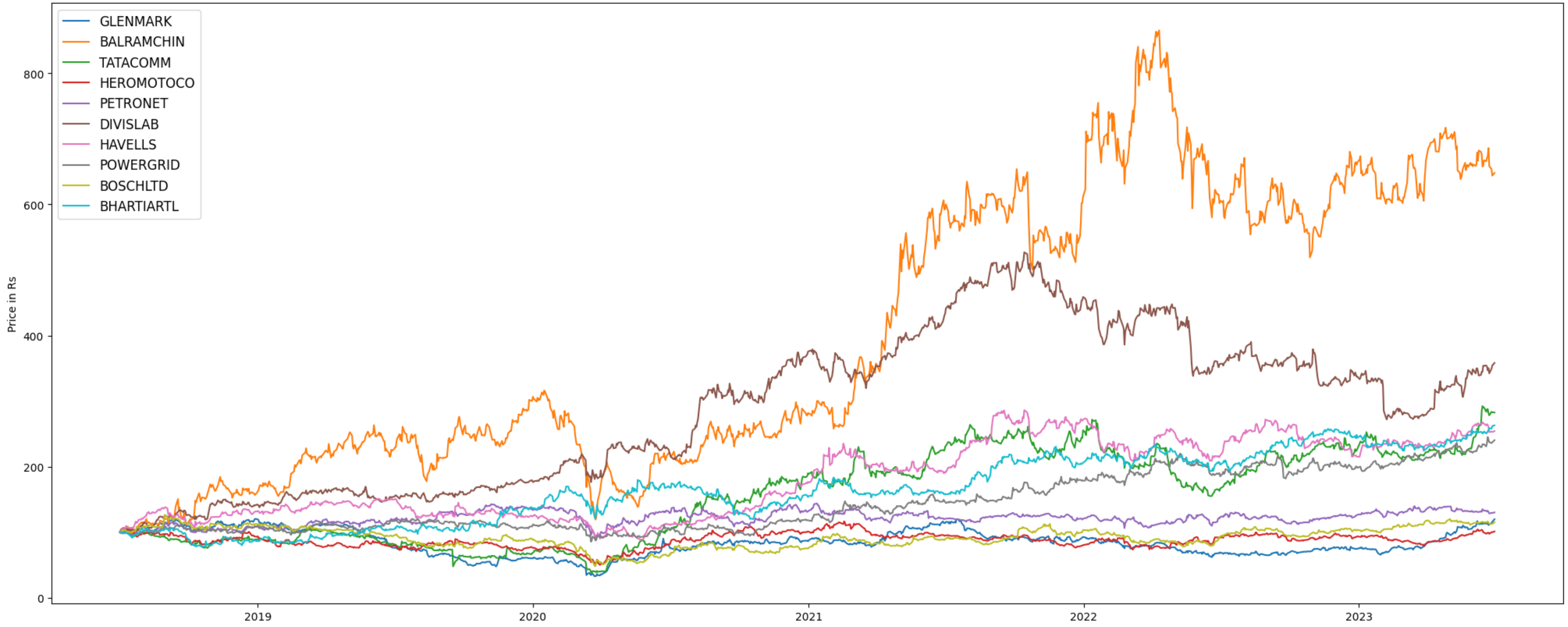
In [67]:
```
port_train
```

Out[67]:

	GLENMARK	BALRAMCHIN	TATACOMM	HEROMOTOCO	PETRONET	DIVISLAB	HAVELLS	POWERGRID	BOSCHLTD	BHARTIARTL
Date										
2018-07-02	566.460327	59.334854	553.352600	2848.199707	170.274185	1011.821411	508.733551	105.323036	16486.957031	330.412506
2018-07-03	574.460693	58.447865	549.475769	2875.094238	170.628174	1045.921387	518.999023	105.038467	16532.378906	332.836426
2018-07-04	582.900024	58.447865	551.811218	2884.392578	173.066833	1061.815308	526.598755	104.156258	16551.380859	329.829010
2018-07-05	568.655518	58.634598	556.341858	2923.545898	172.516190	1037.926025	528.876099	103.387894	17095.361328	326.238037
2018-07-06	568.655518	58.914700	551.717773	3033.082031	170.746185	1034.554688	533.963806	103.473274	16798.201172	325.205627
...	...	...	...	...	...	...	...	...	...	...
2023-06-22	627.700012	407.000000	1556.374634	2825.050049	222.850006	3543.850098	1325.400024	253.550003	18879.300781	842.799988
2023-06-23	633.000000	390.649994	1539.849976	2776.399902	218.800003	3462.949951	1284.949951	250.100006	18616.949219	854.799988
2023-06-26	645.400024	387.950012	1569.150024	2851.250000	220.350006	3534.350098	1290.400024	248.500000	18588.400391	851.950012
2023-06-27	655.700012	382.000000	1565.800049	2844.100098	220.100006	3583.300049	1285.900024	249.550003	18572.599609	864.900024
2023-06-30	678.500000	384.500000	1563.699951	2880.500000	221.649994	3623.500000	1293.250000	253.149994	19003.099609	868.799988

1233 rows × 10 columns

In [51]:
```
plt.figure(figsize=(25,10))                          # plotting normalized adj close prices of randomly chosen stocks
plt.plot(port_train/port_train.iloc[0]*100)
plt.legend(stocks,loc='upper left',fontsize=12)
plt.ylabel('Price in Rs')
plt.show()
```



In [52]:
```
portfolio_train = Portfolio_Optimization(data,stocks)
```

In [53]:
```
weight = []
w_table = pd.DataFrame(columns = list(portfolio_train.historical_stock_covariance().columns))

for i in range(3000):
    w = np.random.random(len(portfolio_train.historical_stock_covariance().columns))
    w = w/(np.sum(w))
    w_table.loc[i] = w                  # building dataframe to store random weights of randomly chosen stocks in the portfolio

returns_of_weighhts = []
for i in range(3000):
        returns = w_table.loc[i]@portfolio_train.historical_stock_Return_Pct_Returns()
        returns_of_weighhts.append(returns)  # calculating return of portfolios

variance_of_weights = []
for i in range(3000):
        var = np.transpose(w_table.loc[i])@portfolio_train.historical_stock_covariance()@w_table.loc[i]
        risk = np.sqrt(var)
        variance_of_weights.append(risk)     # calculating risk of portfolios

w_table["RETURNS"] = returns_of_weighhts
w_table["RISK"] = variance_of_weights

risk_free_rate = 0.071
w_table['SHARPE_RATIO'] = (w_table['RETURNS']-risk_free_rate)/w_table['RISK']  # calculating sharpe ratio of portfolios


high_return = w_table.sort_values(by=['RETURNS'],ascending=False)  # sorting all portfolios on the basis of returns in descending order

low_risk = w_table.sort_values(by=['RISK'],ascending=True)          # sorting all portfolios on the basis of risk in asscending order

high_sharpe = w_table.sort_values(by=['SHARPE_RATIO'],ascending=False)    # sorting all portfolios on the basis of sharpe ratio in descending
```

In [54]:
```
port_amount = int(input("Please enter your portfolio amount: "))
```

```
Please enter your portfolio amount: 10000000
```

In [55]:
```
class PnL():

    def __init__(self,df1,df2,port_amount):

        self.df1 = df1
        self.df2 = df2
        self.port_amount = port_amount

    def value_of_each_security(self):          # calculating amount to funds to be invested in each randomly chosen stock

        value_of_each_security = self.df1.iloc[0,:-3]*self.port_amount
        return value_of_each_security

    def buy_price(self):                       # calculating buy price of stocks on 23-06-2022

        buy_price = self.df2.iloc[-252]
        return buy_price

    def sell_price(self):                      # calculating buy price of stocks on 30-06-2023

        sell_price = self.df2.iloc[-1]
        return sell_price

    def no_of_shares_of_security(self):        # calculating number of shares to buy of each stock

        no_of_shares = PnL.value_of_each_security(self)//PnL.buy_price(self)
        return no_of_shares

    def pnl(self):

        return_from_each_security = ((self.sell_price()-self.buy_price())*self.no_of_shares_of_security()).round(0)  # calculating absolute 
        percent_return_from_each_security = return_from_each_security/self.value_of_each_security()*100  # calculating percentage return of 
        total_return_from_portfolio = np.sum(return_from_each_security).round(0)  # calculating absolute return of the portfolio
        percentage_return_from_portfolio = total_return_from_portfolio/self.port_amount*100  # calcultaing percentage return of the portfoli

        PnL = pd.DataFrame()
        PnL['return_from_each_security'] = return_from_each_security
        PnL['percent_return_from_each_security'] = percent_return_from_each_security

        print(PnL)
        print("Absolute return from portfolio is {}".format(total_return_from_portfolio))
        print("Percentage return from portfolio is {}%".format(percentage_return_from_portfolio.round(2)))
```

In [71]:
```
port_train.iloc[-252]
```

Out[71]:
```
GLENMARK         374.477448
BALRAMCHIN       358.444672
TATACOMM         890.406921
HEROMOTOCO      2541.176514
PETRONET         195.537079
DIVISLAB        3638.446289
HAVELLS         1090.197388
POWERGRID        197.046982
BOSCHLTD       13588.797852
BHARTIARTL       657.278564
Name: 2022-06-23 00:00:00, dtype: float64
```

In [73]:
```
portfolio_test = PnL(high_sharpe,port_train,port_amount)
```

In [57]:
```
portfolio_test.buy_price()
```

Out[57]:
```
GLENMARK         374.477448
BALRAMCHIN       358.444672
TATACOMM         890.406921
HEROMOTOCO      2541.176514
PETRONET         195.537079
DIVISLAB        3638.446289
HAVELLS         1090.197388
POWERGRID        197.046982
BOSCHLTD       13588.797852
BHARTIARTL       657.278564
Name: 2022-06-23 00:00:00, dtype: float64
```

In [58]:
```
portfolio_test.value_of_each_security()
```

Out[58]:
```
GLENMARK      3.654483e+05
BALRAMCHIN    2.414844e+06
TATACOMM      6.726296e+05
HEROMOTOCO    1.003146e+05
PETRONET      4.275865e+05
DIVISLAB      3.187042e+06
HAVELLS       1.729253e+06
POWERGRID     3.819132e+05
BOSCHLTD      1.167286e+05
BHARTIARTL    6.042402e+05
Name: 2558, dtype: float64
```

In [59]:
```
portfolio_test.no_of_shares_of_security()
```

Out[59]:
```
GLENMARK       975.0
BALRAMCHIN    6737.0
TATACOMM       755.0
HEROMOTOCO      39.0
PETRONET      2186.0
DIVISLAB       875.0
HAVELLS       1586.0
POWERGRID     1938.0
BOSCHLTD         8.0
BHARTIARTL     919.0
dtype: float64
```

In [60]:
```
portfolio_test.sell_price()
```

Out[60]:
```
GLENMARK        678.500000
BALRAMCHIN      384.500000
TATACOMM       1563.699951
HEROMOTOCO     2880.500000
PETRONET        221.649994
DIVISLAB       3623.500000
HAVELLS        1293.250000
POWERGRID       253.149994
BOSCHLTD      19003.099609
BHARTIARTL      868.799988
Name: 2023-06-30 00:00:00, dtype: float64
```

In [61]:
```
portfolio_test.pnl()
```

```
            return_from_each_security  percent_return_from_each_security
GLENMARK                     296422.0                          81.111885
BALRAMCHIN                   175535.0                           7.269000
TATACOMM                     508336.0                          75.574433
HEROMOTOCO                    13234.0                          13.192499
PETRONET                      57083.0                          13.350047
DIVISLAB                     -13078.0                          -0.410349
HAVELLS                      322041.0                          18.623132
POWERGRID                    108728.0                          28.469293
BOSCHLTD                      43314.0                          37.106594
BHARTIARTL                   194388.0                          32.170651
Absolute return from portfolio is 1706003.0
Percentage return from portfolio is 17.06%
```

In [62]:
```
high_sharpe
```

Out[62]:

	GLENMARK	BALRAMCHIN	TATACOMM	HEROMOTOCO	PETRONET	DIVISLAB	HAVELLS	POWERGRID	BOSCHLTD	BHARTIARTL	RETURNS	RISK	SHARPE_RATIO
558	0.036545	0.241484	0.067263	0.010031	0.042759	0.318704	0.172925	0.038191	0.011673	0.060424	0.347094	0.244342	1.129950
258	0.030847	0.227704	0.202356	0.003117	0.021879	0.173076	0.106733	0.173045	0.009728	0.051514	0.321779	0.237753	1.054790
528	0.004197	0.197605	0.108270	0.002294	0.110619	0.155509	0.142216	0.170357	0.010727	0.098206	0.302869	0.220466	1.051720
980	0.006945	0.196827	0.045758	0.023376	0.095366	0.199859	0.276718	0.016639	0.004776	0.133738	0.314816	0.232971	1.046553
792	0.071203	0.222603	0.068871	0.002753	0.033058	0.197440	0.215982	0.039738	0.001646	0.146707	0.320124	0.238296	1.045440
...	...	...	...	...	...	...	...	...	...	...	...	...	...
713	0.220512	0.034446	0.062623	0.244931	0.145580	0.018716	0.088008	0.117997	0.046972	0.020215	0.101623	0.220004	0.139191
069	0.202137	0.011380	0.147616	0.172041	0.086822	0.016265	0.101240	0.052008	0.154518	0.055972	0.099671	0.225587	0.127096
498	0.174662	0.016754	0.038967	0.157804	0.217785	0.064644	0.024351	0.121210	0.145469	0.038354	0.097922	0.212038	0.126969
521	0.219453	0.014315	0.168750	0.186627	0.117303	0.010379	0.084098	0.065116	0.101470	0.032489	0.098759	0.226731	0.122431
973	0.061059	0.027523	0.179879	0.252052	0.203993	0.003338	0.044035	0.029838	0.190801	0.007482	0.095667	0.227255	0.108545

)00 rows × 13 columns

In [63]:
```
high_return
```

Out[63]:

	GLENMARK	BALRAMCHIN	TATACOMM	HEROMOTOCO	PETRONET	DIVISLAB	HAVELLS	POWERGRID	BOSCHLTD	BHARTIARTL	RETURNS	RISK	SHARPE_RATI
2558	0.036545	0.241484	0.067263	0.010031	0.042759	0.318704	0.172925	0.038191	0.011673	0.060424	0.347094	0.244342	1.1299
2258	0.030847	0.227704	0.202356	0.003117	0.021879	0.173076	0.106733	0.173045	0.009728	0.051514	0.321779	0.237753	1.0547
792	0.071203	0.222603	0.068871	0.002753	0.033058	0.197440	0.215982	0.039738	0.001646	0.146707	0.320124	0.238296	1.0454
1111	0.015123	0.241507	0.196676	0.023680	0.022121	0.108107	0.015142	0.129800	0.014135	0.233708	0.315586	0.245352	0.9968
1980	0.006945	0.196827	0.045758	0.023376	0.095366	0.199859	0.276718	0.016639	0.004776	0.133738	0.314816	0.232971	1.0465
...	...	...	...	...	...	...	...	...	...	...	...	...	
2713	0.220512	0.034446	0.062623	0.244931	0.145580	0.018716	0.088008	0.117997	0.046972	0.020215	0.101623	0.220004	0.1391
1069	0.202137	0.011380	0.147616	0.172041	0.086822	0.016265	0.101240	0.052008	0.154518	0.055972	0.099671	0.225587	0.1270
1521	0.219453	0.014315	0.168750	0.186627	0.117303	0.010379	0.084098	0.065116	0.101470	0.032489	0.098759	0.226731	0.1224
498	0.174662	0.016754	0.038967	0.157804	0.217785	0.064644	0.024351	0.121210	0.145469	0.038354	0.097922	0.212038	0.1269
2973	0.061059	0.027523	0.179879	0.252052	0.203993	0.003338	0.044035	0.029838	0.190801	0.007482	0.095667	0.227255	0.1085

3000 rows × 13 columns

In [64]:
```
low_risk
```

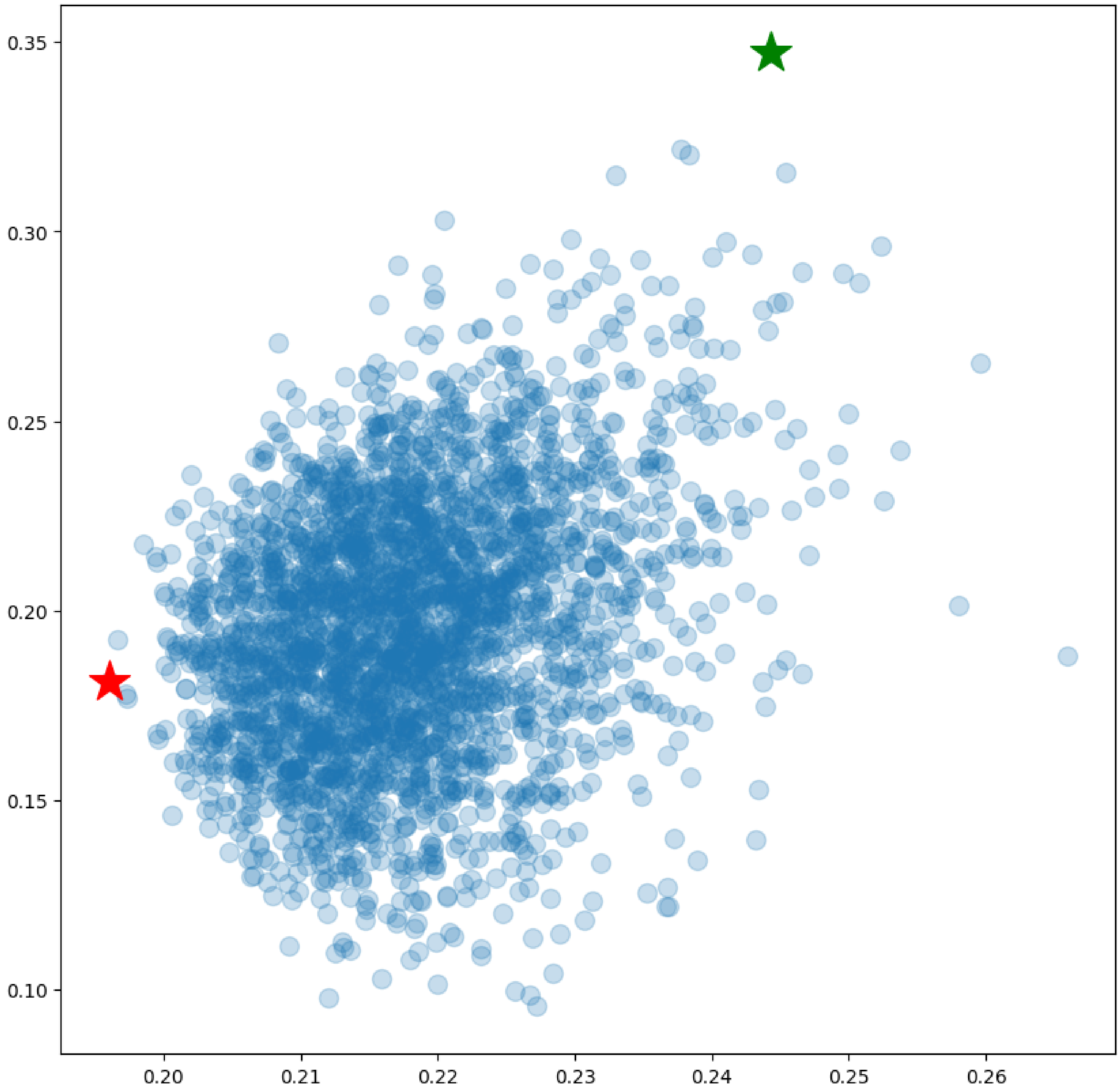
Out[64]:

	GLENMARK	BALRAMCHIN	TATACOMM	HEROMOTOCO	PETRONET	DIVISLAB	HAVELLS	POWERGRID	BOSCHLTD	BHARTIARTL	RETURNS	RISK	SHARPE_RAT
2603	0.063170	0.009518	0.046244	0.070580	0.186930	0.135311	0.166489	0.212911	0.044030	0.064818	0.181107	0.196009	0.5617
144	0.015350	0.014916	0.081144	0.115894	0.166151	0.196786	0.119517	0.231474	0.057925	0.000843	0.192556	0.196627	0.6182
1177	0.031509	0.010956	0.023055	0.082580	0.130388	0.119953	0.124458	0.231509	0.107845	0.137747	0.177948	0.197253	0.5421
1071	0.008468	0.004279	0.053127	0.084368	0.236637	0.120421	0.084372	0.180482	0.056169	0.171678	0.177022	0.197334	0.5372
1591	0.030336	0.026986	0.021134	0.029363	0.213088	0.191949	0.196795	0.185943	0.013590	0.090817	0.217551	0.198493	0.7383
...	...	...	...	...	...	...	...	...	...	...	...	...	
483	0.017681	0.278471	0.040129	0.250253	0.266787	0.014610	0.022639	0.044523	0.026354	0.038554	0.228926	0.252572	0.6252
834	0.091795	0.259223	0.051287	0.052899	0.020495	0.090470	0.027586	0.014025	0.260454	0.131766	0.242286	0.253781	0.6749
218	0.158739	0.209665	0.240419	0.193204	0.019752	0.014683	0.021958	0.004580	0.096852	0.040148	0.201407	0.258007	0.5054
1020	0.062751	0.265007	0.250362	0.098906	0.033960	0.051932	0.030362	0.049650	0.114046	0.043025	0.265178	0.259630	0.7479
1085	0.238221	0.199683	0.234562	0.021022	0.044833	0.033156	0.001574	0.005089	0.208129	0.013730	0.188091	0.265974	0.4402

3000 rows × 13 columns

In [65]:
```
plt.subplots(figsize=(10,10))
plt.scatter(high_sharpe['RISK'],high_sharpe['RETURNS'],marker = 'o', s=100, alpha = 0.25)  # plotting risk and return of all portfolios
plt.scatter(high_sharpe.iloc[0]['RISK'],high_sharpe.iloc[0]['RETURNS'],color='g',marker='*',s=500)  # identifying  portfolio with max sharpe
plt.scatter(low_risk.iloc[0]['RISK'],low_risk.iloc[0]['RETURNS'],color='r',marker='*',s=500)  # identifying portfolio with min risk
```

Out[65]:
```
<matplotlib.collections.PathCollection at 0x1fa3372a4d0>
```


0.35
0.30
0.25
0.20
0.15
0.10
0.20
0.21
0.22
0.23
0.24
0.25
0.26