PART-A

Selection of five stocks, traded on Bombay Stock Exchange (BSE) or National Stock Exchange (NSE) and download of prices from yahoo finance

In [35]:
```
import pandas as pd
import numpy as np
from pandas_datareader import data as pdr
import yfinance as yf
import datetime as dt
import matplotlib.pyplot as plt
import pandas_datareader as web
from scipy import stats
from pandas_datareader import data as pdr
import yfinance as yf
```

In [36]:
```
tickers=['^BSESN','GILLETTE.BO','NESCO.BO','SONATSOFTW.BO','TATAELXSI.BO','TATAMOT(
```

In [37]:
```
data = pdr.get_data_yahoo(tickers, start="2017-01-01",end="2022-01-01")['Adj Close
returns = data.pct_change(1).dropna()
returns.head(10)
```

Out[37]:

Symbols	^BSESN	GILLETTE.BO	NESCO.BO	SONATSOFTW.BO	TATAELXSI.BO	TATAMOTORS.BO
Date						
2017-01-03	0.001797	-0.005411	0.032281	0.026429	-0.006449	-0.012314
2017-01-04	-0.000379	-0.000237	0.039474	-0.007427	0.012233	0.010909
2017-01-05	0.009203	-0.002260	0.008322	0.038663	0.010289	0.031757
2017-01-06	-0.004428	0.000427	-0.001945	-0.054275	-0.028424	-0.008268
2017-01-09	-0.001221	-0.000818	-0.001972	-0.010157	-0.006138	0.005123
2017-01-10	0.006473	0.001080	-0.001000	0.015393	0.019106	0.029879
2017-01-11	0.008954	-0.002086	-0.005553	0.007327	-0.008222	0.006986
2017-01-12	0.003933	-0.000059	-0.013565	-0.007524	0.002716	-0.001060
2017-01-13	-0.000334	0.000154	0.003013	0.003791	-0.003207	-0.007138
2017-01-16	0.001840	-0.002458	0.000412	-0.009315	0.011870	0.023122

In [38]:
```
data
```

Out[38]:

Symbols	^BSESN	GILLETTE.BO	NESCO.BO	SONATSOFTW.BO	TATAELXSI.BO	TATAMOTORS.
Date						
2017-01-02	26595.449219	3896.872314	378.418213	160.056503	663.228088	487.250
2017-01-03	26643.240234	3875.785400	390.633759	164.286697	658.950806	481.250
2017-01-04	26633.130859	3874.868408	406.053619	163.066467	667.012024	486.500
2017-01-05	26878.240234	3866.113037	409.432770	169.371109	673.874634	501.950
2017-01-06	26759.230469	3867.762939	408.636536	160.178528	654.720398	497.799
...	...	...	...	...	...	
2021-12-27	57420.238281	5214.617188	571.096313	838.112183	5417.303223	471.299
2021-12-28	57897.480469	5196.808594	578.457581	834.271423	5501.772949	480.100
2021-12-29	57806.488281	5172.240723	578.159180	845.696472	5678.522949	475.799
2021-12-30	57794.320312	5191.233887	575.274353	846.863281	5783.239258	470.350
2021-12-31	58253.820312	5241.010254	578.955017	851.481934	5838.059570	482.350

1236 rows × 6 columns

In [ ]:

In [39]:
```
mean_daily_ret = data.pct_change(1).mean()
mean_daily_ret
```

Out[39]:
```
Symbols
^BSESN          0.000704
GILLETTE.BO     0.000305
NESCO.BO        0.000555
SONATSOFTW.BO   0.001678
TATAELXSI.BO    0.002045
TATAMOTORS.BO   0.000447
dtype: float64
```

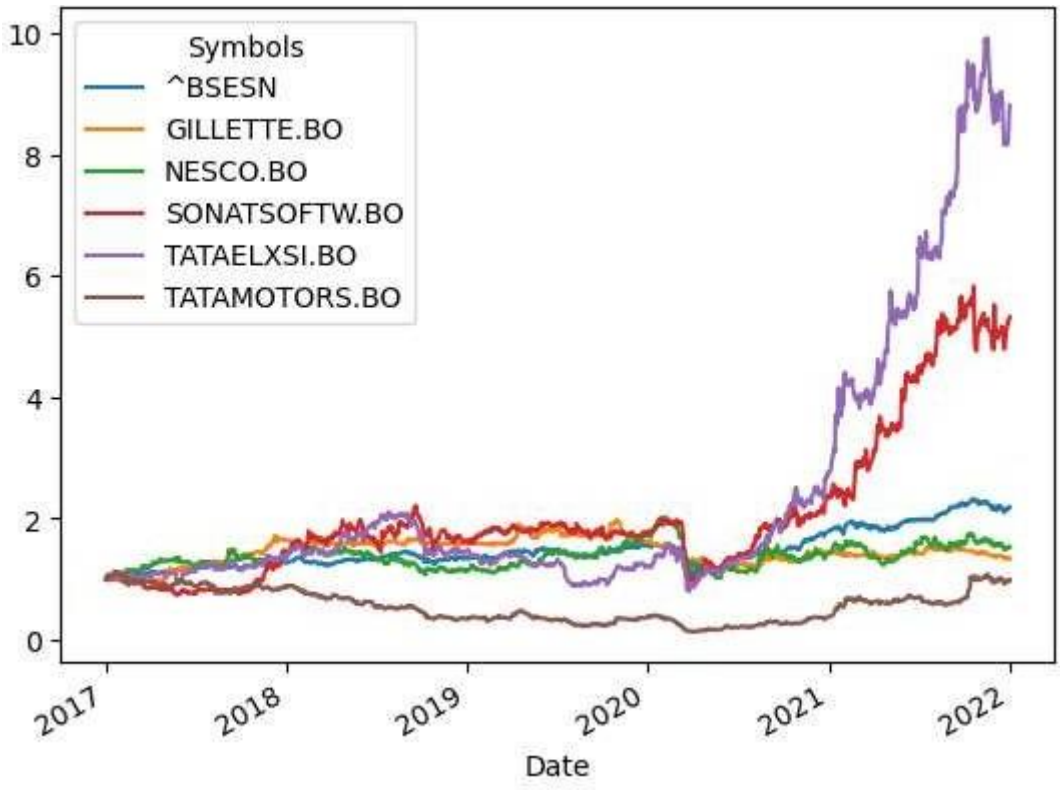
In [40]:
```
data.pct_change(1).corr()
```

Out[40]:

Symbols	^BSESN	GILLETTE.BO	NESCO.BO	SONATSOFTW.BO	TATAELXSI.BO	TATAMOT
Symbols						
^BSESN	1.000000	0.348633	0.378005	0.297889	0.393798	0
GILLETTE.BO	0.348633	1.000000	0.164589	0.191244	0.264459	0
NESCO.BO	0.378005	0.164589	1.000000	0.150330	0.240774	0
SONATSOFTW.BO	0.297889	0.191244	0.150330	1.000000	0.269108	0
TATAELXSI.BO	0.393798	0.264459	0.240774	0.269108	1.000000	0
TATAMOTORS.BO	0.510698	0.237534	0.218541	0.183235	0.278511	1

In [41]:
```
data_normed = data/data.iloc[0]
data_normed.plot()
```

Out[41]:
```
<AxesSubplot:xlabel='Date'>
```



In [42]:
```
data_daily_ret = data.pct_change(1)
data_daily_ret
```

Out[42]:

Symbols	^BSESN	GILLETTE.BO	NESCO.BO	SONATSOFTW.BO	TATAELXSI.BO	TATAMOTORS.BO
Date						
2017-01-02	NaN	NaN	NaN	NaN	NaN	NaN
2017-01-03	0.001797	-0.005411	0.032281	0.026429	-0.006449	-0.012314
2017-01-04	-0.000379	-0.000237	0.039474	-0.007427	0.012233	0.010909
2017-01-05	0.009203	-0.002260	0.008322	0.038663	0.010289	0.031757
2017-01-06	-0.004428	0.000427	-0.001945	-0.054275	-0.028424	-0.008268
...	...	...	...	...	...	...
2021-12-27	0.005180	-0.016332	0.001395	0.025826	-0.010297	0.008021
2021-12-28	0.008311	-0.003415	0.012890	-0.004583	0.015593	0.018672
2021-12-29	-0.001572	-0.004727	-0.000516	0.013695	0.032126	-0.008957
2021-12-30	-0.000210	0.003672	-0.004990	0.001380	0.018441	-0.011454
2021-12-31	0.007951	0.009589	0.006398	0.005454	0.009479	0.025513

1236 rows × 6 columns

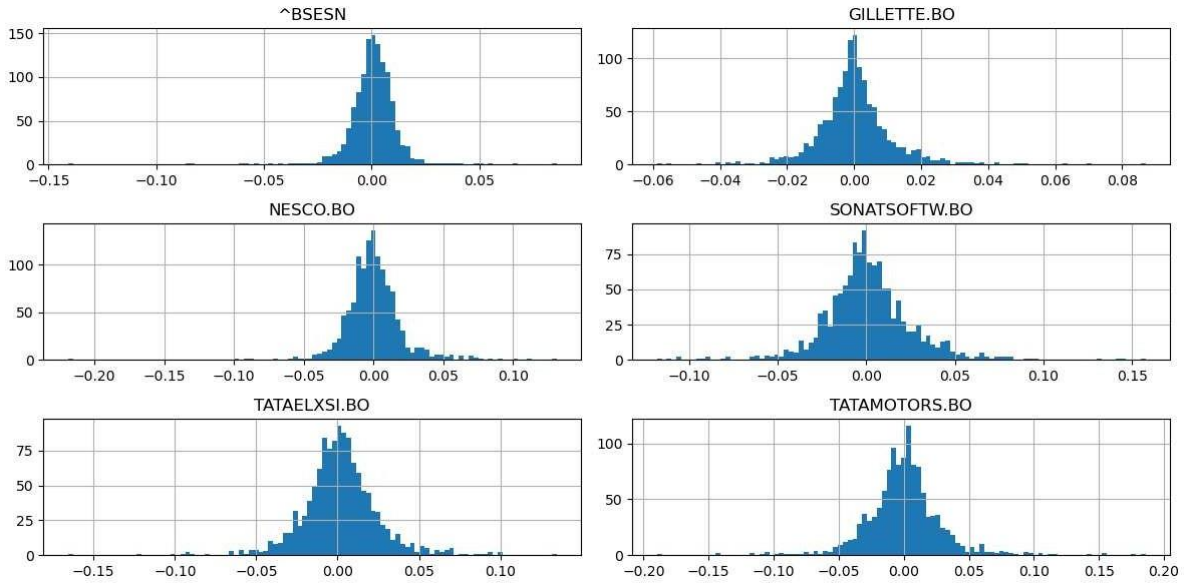
In [43]:
```
log_ret = np.log(data/data.shift(1))
log_ret
```

Out[43]:

Symbols	^BSESN	GILLETTE.BO	NESCO.BO	SONATSOFTW.BO	TATAELXSI.BO	TATAMOTORS.BO
Date						
2017-01-02	NaN	NaN	NaN	NaN	NaN	NaN
2017-01-03	0.001795	-0.005426	0.031770	0.026086	-0.006470	-0.012390
2017-01-04	-0.000380	-0.000237	0.038715	-0.007455	0.012159	0.010850
2017-01-05	0.009161	-0.002262	0.008287	0.037934	0.010236	0.031264
2017-01-06	-0.004438	0.000427	-0.001947	-0.055803	-0.028836	-0.008302
...	...	...	...	...	...	...
2021-12-27	0.005167	-0.016467	0.001394	0.025498	-0.010350	0.007989
2021-12-28	0.008277	-0.003421	0.012807	-0.004593	0.015472	0.018500
2021-12-29	-0.001573	-0.004739	-0.000516	0.013602	0.031621	-0.008997
2021-12-30	-0.000211	0.003665	-0.005002	0.001379	0.018273	-0.011520
2021-12-31	0.007919	0.009543	0.006378	0.005439	0.009435	0.025193

1236 rows × 6 columns

In [44]:
```
log_ret.hist(bins=100,figsize=(12,6))
plt.tight_layout()
```



In [45]:
```
log_ret.describe().transpose()
```

Out[45]:

Symbols	count	mean	std	min	25%	50%	75%	max
^BSESN	1225.0	0.000602	0.011774	-0.141017	-0.004297	0.000898	0.006319	0.085947
GILLETTE.BO	1235.0	0.000240	0.011402	-0.058816	-0.005079	-0.000237	0.004869	0.086985
NESCO.BO	1235.0	0.000344	0.020516	-0.219048	-0.010030	-0.000595	0.009272	0.131324
SONATSOFTW.BO	1235.0	0.001353	0.025360	-0.118081	-0.011613	-0.000221	0.012632	0.157527
TATAELXSI.BO	1235.0	0.001761	0.023751	-0.165787	-0.010036	0.001283	0.013274	0.134614
TATAMOTORS.BO	1235.0	-0.000008	0.030136	-0.189675	-0.013630	0.000358	0.012556	0.185860

In [46]:
```
log_ret.mean()*252
```

Out[46]:
```
Symbols
^BSESN           0.151709
GILLETTE.BO      0.060468
NESCO.BO         0.086767
SONATSOFTW.BO    0.341057
TATAELXSI.BO     0.443813
TATAMOTORS.BO   -0.002062
dtype: float64
```

In [47]:
```
log_ret.cov()
```

Out[47]:

Symbols	^BSESN	GILLETTE.BO	NESCO.BO	SONATSOFTW.BO	TATAELXSI.BO	TATAMOT
Symbols						
^BSESN	0.000139	0.000047	0.000096	0.000093	0.000112	0
GILLETTE.BO	0.000047	0.000130	0.000041	0.000057	0.000073	0
NESCO.BO	0.000096	0.000041	0.000421	0.000084	0.000122	0
SONATSOFTW.BO	0.000093	0.000057	0.000084	0.000643	0.000168	0
TATAELXSI.BO	0.000112	0.000073	0.000122	0.000168	0.000564	0
TATAMOTORS.BO	0.000181	0.000083	0.000141	0.000148	0.000204	0

In [48]:
```
log_ret.cov()*252
```

Out[48]:

Symbols	^BSESN	GILLETTE.BO	NESCO.BO	SONATSOFTW.BO	TATAELXSI.BO	TATAMOT
Symbols						
^BSESN	0.034936	0.011939	0.024245	0.023463	0.028247	0
GILLETTE.BO	0.011939	0.032762	0.010238	0.014351	0.018456	0
NESCO.BO	0.024245	0.010238	0.106065	0.021163	0.030734	0
SONATSOFTW.BO	0.023463	0.014351	0.021163	0.162068	0.042306	0
TATAELXSI.BO	0.028247	0.018456	0.030734	0.042306	0.142158	0
TATAMOTORS.BO	0.045562	0.020889	0.035495	0.037293	0.051381	0

In [49]:
```
np.random.seed()
print('Stocks :',data.columns)
weights = np.array(np.random.random(6))
print('\nCreating Random Weights :',weights)
weights = weights / np.sum(weights)
print('\nRebalance to sum to 1.0 :',weights)
exp_ret = np.sum(log_ret.mean() * weights) *252
print('\nExpected Portfolio Return :',exp_ret)
exp_vol = np.sqrt(np.dot(weights.T, np.dot(log_ret.cov() * 252, weights)))
print('\nExpected Volatility :',exp_vol)
sr = exp_ret/exp_vol
print('\nSharpe Ratio :',sr)
```

```
Stocks : Index(['^BSESN', 'GILLETTE.BO', 'NESCO.BO', 'SONATSOFTW.BO', 'TATAELXSI.B
O',
       'TATAMOTORS.BO'],
      dtype='object', name='Symbols')

Creating Random Weights : [0.79161769 0.52892013 0.50700797 0.35200363 0.74982148
0.95643073]

Rebalance to sum to 1.0 : [0.20372056 0.13611609 0.13047706 0.09058713 0.19296443
0.24613473]

Expected Portfolio Return : 0.1664858514902215

Expected Volatility : 0.2222490588279674

Sharpe Ratio : 0.7490958673489385
```

In [50]:
```
num_ports = 20000

all_weights = np.zeros((num_ports,len(data.columns)))
ret_arr = np.zeros(num_ports)
vol_arr = np.zeros(num_ports)
sharpe_arr = np.zeros(num_ports)

for ind in range(num_ports):
    weights = np.array(np.random.random(6))
    weights = weights / np.sum(weights)
    all_weights[ind,:] = weights
    ret_arr[ind] = np.sum((log_ret.mean() * weights) *252)
    vol_arr[ind] = np.sqrt(np.dot(weights.T, np.dot(log_ret.cov() * 252, weights)))
    sharpe_arr[ind] = ret_arr[ind]/vol_arr[ind]
```

In [ ]:
```
```

In [51]:
```
variance = df_log_ret['^BSESN'].var()
beta_val = np.array(cov['^BSESN']/variance)
for stocks in list(df.columns):
    print('Beta for ',stocks,' is: ',round(beta_val[list(df.columns).index(stocks)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_24580\2248056719.py in <module>
----> 1 variance = df_log_ret['^BSESN'].var()
      2 beta_val = np.array(cov['^BSESN']/variance)
      3 for stocks in list(df.columns):
      4     print('Beta for ',stocks,' is: ',round(beta_val[list(df.columns).index
(stocks)],5))

NameError: name 'df_log_ret' is not defined
```

In [52]:
```
sharpe_arr.max()
```

Out[52]:
```
1.2957094247162397
```

In [53]:
```
sharpe_arr.argmax()
```

Out[53]:
```
7933
```

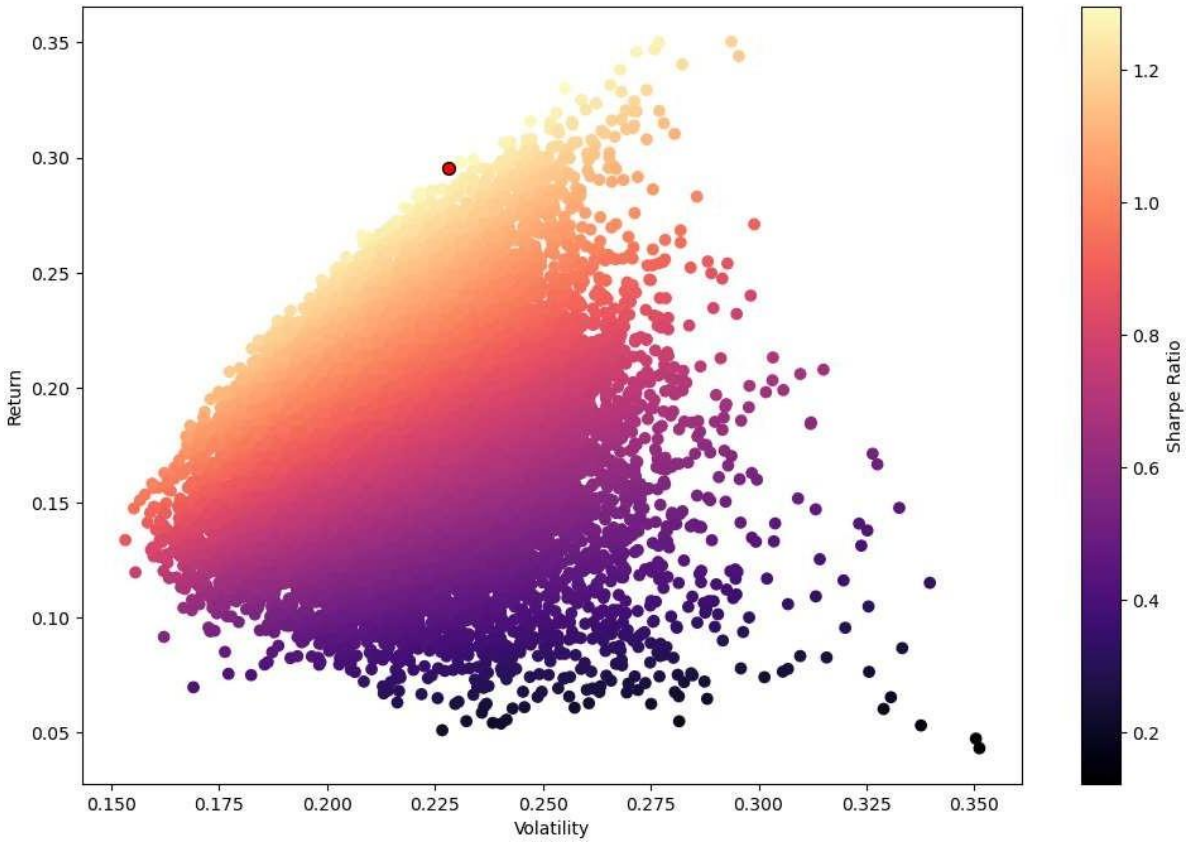
In [54]:
```
all_weights[sharpe_arr.argmax(),:]
```

Out[54]:
```
array([0.35780022, 0.03639236, 0.00216187, 0.25331642, 0.34346823,
       0.00686091])
```

In [55]:
```
max_sr_ret = ret_arr[sharpe_arr.argmax()]
max_sr_vol = vol_arr[sharpe_arr.argmax()]
```

In [56]:
```
plt.figure(figsize=(12,8))
plt.scatter(vol_arr,ret_arr,c=sharpe_arr,cmap='magma')
plt.colorbar(label='Sharpe Ratio')
plt.xlabel('Volatility')
plt.ylabel('Return')
plt.scatter(max_sr_vol,max_sr_ret,c='red',s=50,edgecolors='black')
```

Out[56]:
```
<matplotlib.collections.PathCollection at 0x210d7c5d2e0>
```



In [57]:
```
def get_ret_vol_sr(weights):
    weights = np.array(weights)
    ret = np.sum(log_ret.mean() * weights) * 252
    vol = np.sqrt(np.dot(weights.T, np.dot(log_ret.cov() * 252, weights)))
    sr = ret/vol
    return np.array([ret,vol,sr])
```

In [58]:
```
from scipy.optimize import minimize
```

In [59]:
```
def neg_sharpe(weights):
```

```
    return  get_ret_vol_sr(weights)[2] * -1
```

In [60]:
```
def check_sum(weights):
    return np.sum(weights) - 1
```

In [61]:
```
cons = ({'type':'eq','fun': check_sum})
```

In [62]:
```
bounds = ((0, 1), (0, 1), (0, 1), (0, 1),(0, 1),(0,1))
```

In [63]:
```
init_guess = [0.25,0.25,0.25,0.25,0.25,0.25]
```

In [64]:
```
opt_results = minimize(neg_sharpe,init_guess,method='SLSQP',bounds=bounds,constrai
```

In [65]:
```
opt_results
```

Out[65]:
```
     fun: -1.3200806461771595
     jac: array([-5.76227903e-05,  8.22551548e-02,  2.04943761e-01, -3.85180116e-0
4,
        2.31519341e-04,  9.64068234e-01])
 message: 'Optimization terminated successfully'
    nfev: 50
     nit: 7
    njev: 7
  status: 0
 success: True
       x: array([2.92553215e-01, 0.00000000e+00, 0.00000000e+00, 2.38245210e-01,
       4.69201574e-01, 2.94902991e-17])
```

In [66]:
```
opt_results.x
```

Out[66]:
```
array([2.92553215e-01, 0.00000000e+00, 0.00000000e+00, 2.38245210e-01,
       4.69201574e-01, 2.94902991e-17])
```

In [67]:
```
get_ret_vol_sr(opt_results.x)
```

Out[67]:
```
array([0.33387599, 0.2529209 , 1.32008065])
```

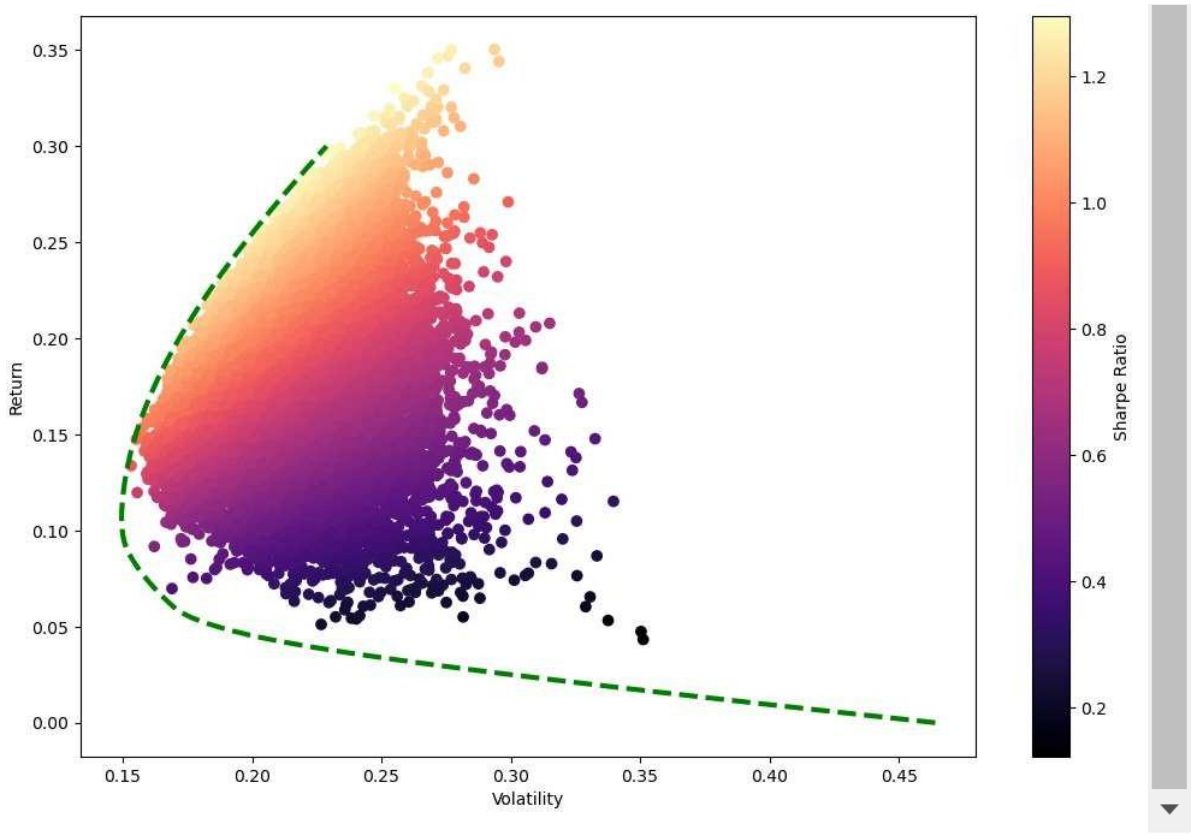
In [68]:
```
frontier_y = np.linspace(0,0.3,100)
```

In [69]:
```
def minimize_volatility(weights):
    return  get_ret_vol_sr(weights)[1]
```

In [70]:
```
frontier_volatility = []

for possible_return in frontier_y:
    cons = ({'type':'eq','fun': check_sum},
            {'type':'eq','fun': lambda w: get_ret_vol_sr(w)[0] - possible_return})

    result = minimize(minimize_volatility,init_guess,method='SLSQP',bounds=bounds,

    frontier_volatility.append(result['fun'])
```

In [71]:
```
plt.figure(figsize=(12,8))
plt.scatter(vol_arr,ret_arr,c=sharpe_arr,cmap='magma')
plt.colorbar(label='Sharpe Ratio')
plt.xlabel('Volatility')
plt.ylabel('Return')
plt.plot(frontier_volatility,frontier_y,'g--',linewidth=3)
```

Out[71]:
```
[<matplotlib.lines.Line2D at 0x210d7edd340>]
```



PART-B

Calculation of portfolio mean, portfolio standard deviation and Sharpe Ratio for 100 portfolios and associated risk and return (portfolio mean) for each of the 100 portfolios

In [72]:
```
import pandas as pd
import numpy as np
from numpy import random
np.random.seed(101)
from dateutil.relativedelta import relativedelta

df = pd.DataFrame(random.rand(3000,6),columns=['^BSESN','GILLETTE.BO','NESCO.BO','
df
```

Out[72]:

	^BSESN	GILLETTE.BO	NESCO.BO	SONATSOFTW.BO	TATAELXSI.BO	TATAMOTORS.BO
0	0.516399	0.570668	0.028474	0.171522	0.685277	0.833897
1	0.306966	0.893613	0.721544	0.189939	0.554228	0.352132
2	0.181892	0.785602	0.965483	0.232354	0.083561	0.603548
3	0.728993	0.276239	0.685306	0.517867	0.048485	0.137869
4	0.186967	0.994318	0.520665	0.578790	0.734819	0.541962
...	...	...	...	...	...	...
2995	0.667841	0.111362	0.398333	0.481424	0.124680	0.426233
2996	0.487444	0.519960	0.259894	0.865307	0.688912	0.423924
2997	0.544872	0.730864	0.429949	0.689675	0.000470	0.164036
2998	0.357891	0.621519	0.155025	0.871035	0.084572	0.716462
2999	0.783917	0.790562	0.687668	0.874072	0.842086	0.766649

3000 rows × 6 columns

In [73]:
```
mean_list = list(df.mean())
std_list = list(df.std())
col_list = list(df.columns)
for i in range(len(mean_list)):
    print(col_list[i],' has mean ',mean_list[i],' and standard deviation ',std_lis
```

```
^BSESN  has mean  0.5107562957575797  and standard deviation  0.28801751874640025
GILLETTE.BO  has mean  0.508203670977288  and standard deviation  0.28760151950620
05
NESCO.BO  has mean  0.501506341157331  and standard deviation  0.2906261580845586
SONATSOFTW.BO  has mean  0.4956941292977073  and standard deviation  0.28903138927
56788
TATAELXSI.BO  has mean  0.4891618785758223  and standard deviation  0.289534287346
13545
TATAMOTORS.BO  has mean  0.4987611532817007  and standard deviation  0.28545327083
340066
```

In [74]:
```
#Risk free rate is 0.00
for i in range(len(col_list)):
    new_mean = (df.iloc[:,i].pct_change().mean())/100
    sr = new_mean/std_list[i]
    print('Sharpe ratio for ',col_list[i],' is: ',sr)
```

```
Sharpe ratio for  ^BSESN  is:  0.09811867364132237
Sharpe ratio for  GILLETTE.BO  is:  0.12588204074766407
Sharpe ratio for  NESCO.BO  is:  0.13149616784754695
Sharpe ratio for  SONATSOFTW.BO  is:  0.09041096373499277
Sharpe ratio for  TATAELXSI.BO  is:  1.1838656467452686
Sharpe ratio for  TATAMOTORS.BO  is:  0.11219078810286652
```

In [75]:
```
log_ret = np.log(df/df.shift(1)).dropna()
log_ret.describe().transpose()
```

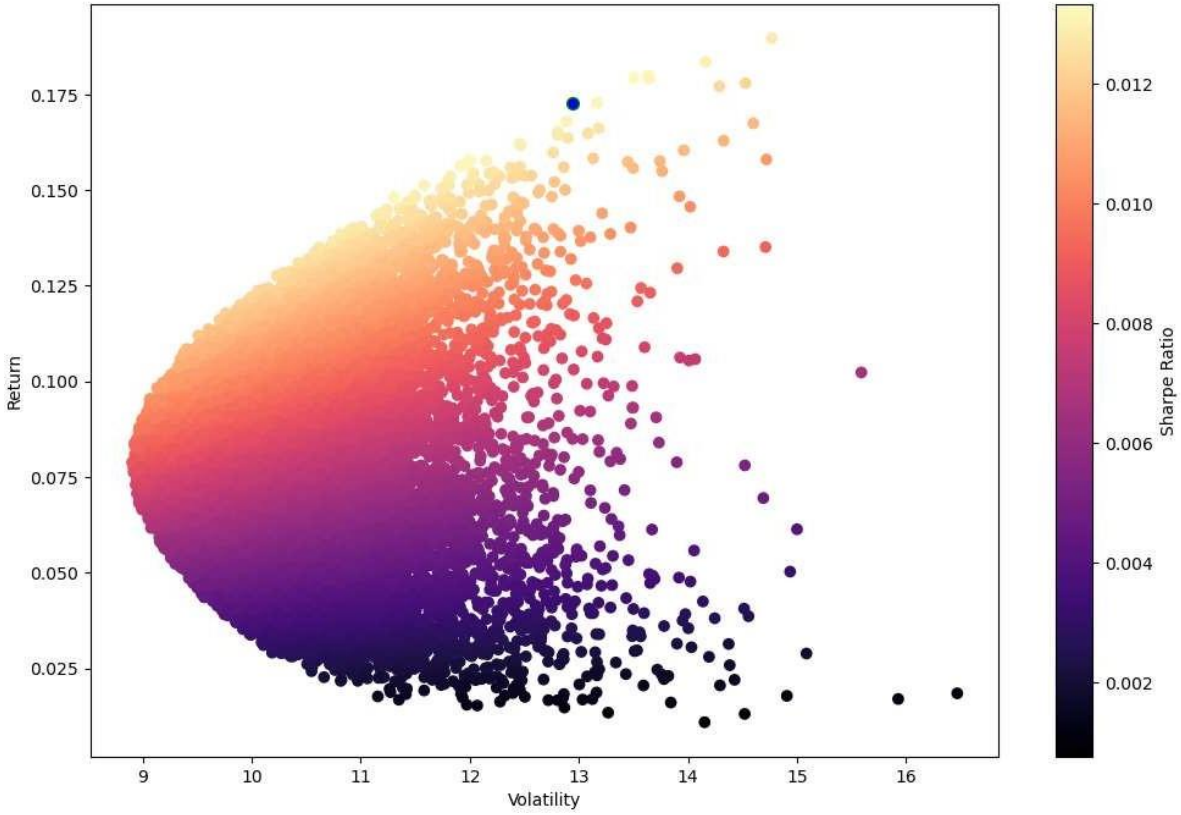
Out[75]:

	count	mean	std	min	25%	50%	75%	ma
^BSESN	2999.0	0.000139	1.360786	-6.603991	-0.693482	-0.005890	0.646401	7.34922
GILLETTE.BO	2999.0	0.000109	1.426938	-6.857274	-0.647084	-0.013128	0.683810	6.91376
NESCO.BO	2999.0	0.001062	1.436526	-7.281439	-0.702512	0.013378	0.693557	7.39456
SONATSOFTW.BO	2999.0	0.000543	1.394466	-7.077870	-0.720443	0.007553	0.701706	6.55414
TATAELXSI.BO	2999.0	0.000069	1.441250	-9.550984	-0.713369	-0.013456	0.734111	11.45681
TATAMOTORS.BO	2999.0	-0.000028	1.384347	-6.824319	-0.680133	0.019910	0.653448	7.27265

In [76]:
```
#Plotting of the statistics
ports = 20000

ret_arr = np.zeros(ports)
vol_arr = np.zeros(ports)
var_covar = log_ret.cov()*252 #Construct var_covar matrix
sharpe_arr = np.zeros(ports)

for vals in range(ports):

    weights = random.random(len(col_list))
    weights = weights/np.sum(weights)
    ret_arr[vals] = np.sum(log_ret.mean() * weights)*252 #Yearly return
    vol_arr[vals] = np.sqrt(np.dot(weights,np.dot(var_covar,weights.T)))
    sharpe_arr[vals] = ret_arr[vals]/vol_arr[vals]
```

In [77]:
```
import matplotlib.pyplot as plt
%matplotlib inline

plt.figure(figsize=(12,8))
plt.scatter(vol_arr,ret_arr,c=sharpe_arr,cmap='magma')
plt.colorbar(label='Sharpe Ratio')
plt.xlabel('Volatility')
plt.ylabel('Return')

# Add red dot for max SR
plt.scatter(vol_arr[sharpe_arr.argmax()],ret_arr[sharpe_arr.argmax()],c='Blue',s=5(
```

Out[77]:
```
<matplotlib.collections.PathCollection at 0x210d7d3b280>
```



Optimizing the Sharpe Ratio and finding the weights corresponding to Maximum Sharpe Ratio

In [78]:
```
from scipy.optimize import minimize
```

In [79]:
```
def ret_vol_sr(weights,rf):

    ret = np.sum((log_ret.mean()*weights))*252
    vol = np.sqrt(np.dot(weights,np.dot(var_covar,weights.T)))
    sharpe_arr = (ret - rf)/vol
    return np.array([ret,vol,sharpe_arr],dtype=object)


def neg_sharpe(weights):

    rf = rsk
    return ret_vol_sr(weights,rf)[2]*-1

def check_weight(weights):
    return (np.sum(weights)-1)

cons = ({'type':'eq','fun':check_weight})

bounds = ((0,1),(0,1),(0,1),(0,1),(0,1),(0,1))

init_guess = np.array([0.166,0.166,0.166,0.166,0.166,0.17],dtype=object)

risk = np.sort(np.array([0.03,0.053,0.62,0.058,0.082,0.059,0.263,0.149,0.262,0.076

# Getting input from user for risk level
# Considering the risk level is upto 8
risk = risk[:8]
```

```
sharpe_w = []
weights_w = []

for rsk in risk:

    opt_results = minimize(neg_sharpe, init_guess, method='SLSQP', bounds=bounds,
    sharpe = ret_vol_sr(opt_results.x,rsk)[2]
    sharpe_w.append(sharpe)
    weights_w.append(opt_results.x)

temp = {}
for rsk in range(len(risk)):

    temp[risk[rsk]] = [sharpe_w[rsk]]+list(weights_w[rsk])

df1 = pd.DataFrame(temp)
df1 = df1.transpose()
df1.index.name = 'Risk Rate'
df1.columns = ['Sharpe','w1','w2','w3','w4','w5','w6']
df1
```

Out[79]:

Risk Rate	Sharpe	w1	w2	w3	w4	w5	w6
0.030	0.011381	0.038116	1.106679e-02	0.651866	2.989518e-01	0.000000e+00	8.673617e-19
0.053	0.010049	0.000000	0.000000e+00	0.700799	2.992011e-01	3.903128e-18	0.000000e+00
0.058	0.009763	0.000000	7.589415e-18	0.700664	2.993358e-01	0.000000e+00	8.890458e-18
0.059	0.009705	0.000000	0.000000e+00	0.700618	2.993819e-01	1.734723e-18	0.000000e+00
0.076	0.008773	0.000000	0.000000e+00	0.758118	2.418823e-01	2.629448e-18	7.283973e-18
0.082	0.008446	0.000000	0.000000e+00	0.770259	2.297406e-01	2.540517e-18	7.282722e-18
0.149	0.005200	0.000000	1.071368e-17	1.000000	0.000000e+00	0.000000e+00	2.076088e-18
0.262	0.000244	0.000000	2.304681e-17	1.000000	1.942890e-16	6.078821e-18	0.000000e+00

In [80]:
```
df1.head(1)
```

Out[80]:

Risk Rate	Sharpe	w1	w2	w3	w4	w5	w6
0.03	0.011381	0.038116	0.011067	0.651866	0.298952	0.0	8.673617e-19

In [81]:
```
#Get Beta Values
#Formula = cov(X,Y)/var(^NSEI(market-standard))

variance = df_log_ret['^BSESN'].var()
beta_val = np.array(cov['^BSESN']/variance)
for stocks in list(df.columns):
    print('Beta for ',stocks,' is: ',round(beta_val[list(df.columns).index(stocks)

    #Beta for Market standards is always 1.0
```