01 NIFTY 100 Portfolio Optimization

In [54]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import datetime as dt
import pandas_datareader as web
import random
```

File to tickers

In [55]:

```
nifty_file=pd.read_csv('ind_nifty100list.csv')
nifty_file['Yahoo_Symbol']='Hello World'
nifty_file.Yahoo_Symbol= nifty_file.Symbol + '.NS'
nifty_file.to_csv('Nifty_yahoo_ticker.csv')
```

Tickers into a list

In [56]:

```
nif100 = nifty_file['Yahoo_Symbol'].to_list()
```

In [57]:

```
nif100
```

```
Out[57]: ['ACC.NS',
           'ABBOTINDIA.NS',
           'ADANIGREEN.NS',
           'ADANIPORTS.NS',
           'ADANITRANS.NS',
           'ALKEM.NS',
           'AMBUJACEM.NS',
           'ASIANPAINT.NS',
           'AUROPHARMA.NS',
           'DMART.NS',
           'AXISBANK.NS',
           'BAJAJ-AUTO.NS',
           'BAJFINANCE.NS',
           'BAJAJFINSV.NS',
           'BAJAJHLDNG.NS',
           'BANDHANBNK.NS',
           'BANKBARODA.NS',
           'BERGEPAINT.NS',
           'BPCL.NS',
           'BHARTIARTL.NS',
           'BIOCON.NS',
           'BOSCHLTD.NS',
           'BRITANNIA.NS',
           'CADILAHC.NS',
           'CIPLA.NS',
           'COALINDIA.NS',
           'COLPAL.NS',
           'CONCOR.NS',
           'DLF.NS',
           'DABUR.NS',
           'DIVISLAB.NS',
           'DRREDDY.NS',
           'EICHERMOT.NS',
           'GAIL.NS',
           'GICRE.NS',
           'GODREJCP.NS',
           'GRASIM.NS',
           'HCLTECH.NS',
           'HDFCAMC.NS',
           'HDFCBANK.NS',
           'HDFCLIFE.NS',
           'HAVELLS.NS',
           'HEROMOTOCO.NS',
           'HINDALCO.NS',
           'HINDPETRO.NS',
           'HINDUNILVR.NS',
           'HINDZINC.NS',
           'HDFC.NS',
           'ICICIBANK.NS',
           'ICICIGI.NS',
           'ICICIPRULI.NS',
           'ITC.NS',
           'IOC.NS',
           'IGL.NS',
           'INDUSTOWER.NS',
           'INDUSINDBK.NS',
           'NAUKRI.NS',
           'INFY.NS',
           'INDIGO.NS',
           'JSWSTEEL.NS',
           'KOTAKBANK.NS',
           'LTI.NS',
           'LT.NS',
           'LUPIN.NS',
           'M&M.NS',
           'MARICO.NS',
           'MARUTI.NS',
           'MOTHERSUMI.NS',
           'MUTHOOTFIN.NS',
           'NMDC.NS',
           'NTPC.NS',
           'NESTLEIND.NS',
           'ONGC.NS',
           'OFSS.NS',
           'PETRONET.NS',
           'PIDILITIND.NS',
           'PEL.NS',
           'PFC.NS',
           'POWERGRID.NS',
           'PGHH.NS',
           'PNB.NS',
           'RELIANCE.NS',
           'SBICARD.NS',
           'SBILIFE.NS',
           'SHREECEM.NS',
           'SIEMENS.NS',
           'SBIN.NS',
           'SUNPHARMA.NS',
           'TCS.NS',
           'TATACONSUM.NS',
           'TATAMOTORS.NS',
           'TATASTEEL.NS',
           'TECHM.NS',
           'TITAN.NS',
           'TORNTPHARM.NS',
           'UPL.NS',
           'ULTRACEMCO.NS',
           'UBL.NS',
           'MCDOWELL-N.NS',
           'WIPRO.NS']
```

Exception Handling

In [82]:

```
end = datetime.today()
begin=end-pd.DateOffset(365*3)
st=begin.strftime('%Y-%m-%d')
ed=end.strftime('%Y-%m-%d')
data=[]
niftyd_list=[]
for i,k in enumerate(nif1):

    try:
        data.append(pdr.get_data_yahoo(k,st,ed)['Adj Close'])
        niftyd_list.append(k)

    except Exception:
        print('Not found',k)

        pass
```

Data List(Print is too large)

Confirmed Data Stocks

In [84]:

```
niftyd_list
```

```
Out[84]: ['ACC.NS',
           'ABBOTINDIA.NS',
           'ADANIGREEN.NS',
           'ADANIPORTS.NS',
           'ADANITRANS.NS',
           'ALKEM.NS',
           'AMBUJACEM.NS',
           'ASIANPAINT.NS',
           'AUROPHARMA.NS',
           'DMART.NS',
           'AXISBANK.NS',
           'BAJAJ-AUTO.NS',
           'BAJFINANCE.NS',
           'BAJAJFINSV.NS',
           'BAJAJHLDNG.NS',
           'BANDHANBNK.NS',
           'BANKBARODA.NS',
           'BERGEPAINT.NS',
           'BPCL.NS',
           'BHARTIARTL.NS',
           'BIOCON.NS',
           'BOSCHLTD.NS',
           'BRITANNIA.NS',
           'CADILAHC.NS',
           'CIPLA.NS',
           'COALINDIA.NS',
           'COLPAL.NS',
           'CONCOR.NS',
           'DLF.NS',
           'DABUR.NS',
           'DIVISLAB.NS',
           'DRREDDY.NS',
           'EICHERMOT.NS',
           'GAIL.NS',
           'GICRE.NS',
           'GODREJCP.NS',
           'GRASIM.NS',
           'HCLTECH.NS',
           'HDFCAMC.NS',
           'HDFCBANK.NS',
           'HDFCLIFE.NS',
           'HAVELLS.NS',
           'HEROMOTOCO.NS',
           'HINDALCO.NS',
           'HINDPETRO.NS',
           'HINDUNILVR.NS',
           'HINDZINC.NS',
           'HDFC.NS',
           'ICICIBANK.NS',
           'ICICIGI.NS',
           'ICICIPRULI.NS',
           'ITC.NS',
           'IOC.NS',
           'IGL.NS',
           'INDUSTOWER.NS',
           'INDUSINDBK.NS',
           'NAUKRI.NS',
           'INFY.NS',
           'INDIGO.NS',
           'JSWSTEEL.NS',
           'KOTAKBANK.NS',
           'LTI.NS',
           'LT.NS',
           'LUPIN.NS',
           'M&M.NS',
           'MARICO.NS',
           'MARUTI.NS',
           'MOTHERSUMI.NS',
           'MUTHOOTFIN.NS',
           'NMDC.NS',
           'NTPC.NS',
           'NESTLEIND.NS',
           'ONGC.NS',
           'OFSS.NS',
           'PETRONET.NS',
           'PIDILITIND.NS',
           'PEL.NS',
           'PFC.NS',
           'POWERGRID.NS',
           'PGHH.NS',
           'PNB.NS',
           'RELIANCE.NS']
```

In [85]:

```
nif2=pd.DataFrame()
for i in niftyd_list:
   nif2[i]=web.DataReader(i,'yahoo',st, ed)['Adj Close']
```

In [86]:

```
nif2.head()
```

Out[86]:

	ACC.NS	ABBOTINDIA.NS	ADANIGREEN.NS	ADANIPORTS.NS	ADANITRANS.NS	ALI
Date						
2018-02-02	1570.710693	5416.321777	NaN	411.091797	193.800003	2141
2018-02-05	1534.570801	5278.685059	NaN	401.501801	194.399994	2114
2018-02-06	1513.812500	5138.213379	NaN	398.157562	190.649994	2093
2018-02-07	1484.311523	5240.375488	NaN	400.321472	193.350006	2107
2018-02-08	1583.287476	5346.158203	NaN	398.452637	197.550003	2122

5 rows × 82 columns

Misssing Values Treatment

In [88]:

```
nif3 = nif2.dropna()
nif3.head()
```

Out[88]:

	ACC.NS	ABBOTINDIA.NS	ADANIGREEN.NS	ADANIPORTS.NS	ADANITRANS.NS	ALI
Date						
2018-08-06	1455.936646	7953.461914	73.300003	392.799957	170.949997	2135
2018-08-07	1469.932495	7876.536133	69.650002	368.036804	176.050003	2091
2018-08-08	1467.687378	7881.044434	66.199997	370.310486	172.100006	2088
2018-08-09	1500.598877	7760.462402	62.900002	373.671570	168.449997	2064
2018-08-10	1476.285278	7836.163086	66.000000	374.956635	163.949997	2015

5 rows × 82 columns

In [90]:

```
nif3.mean()
```

Out[90]:

```
ACC.NS              1440.650368
ABBOTINDIA.NS      11625.906055
ADANIGREEN.NS        252.358360
ADANIPORTS.NS        365.212156
ADANITRANS.NS        250.993524
                        ...
PFC.NS                97.496751
POWERGRID.NS         172.189845
PGHH.NS            10334.789879
PNB.NS                57.272787
RELIANCE.NS         1475.688587
Length: 82, dtype: float64
```

In [89]:

```
nif3.std()
```

Out[89]:

```
ACC.NS              154.265542
ABBOTINDIA.NS      3728.350971
ADANIGREEN.NS       314.721792
ADANIPORTS.NS        50.304682
ADANITRANS.NS        70.474859
                        ...
PFC.NS               13.296248
POWERGRID.NS         11.916080
PGHH.NS             687.346927
PNB.NS               20.848792
RELIANCE.NS         360.599201
Length: 82, dtype: float64
```

Log Returns

In [94]:
```
l_ret = np.log(nif2/nif2.shift())
l_ret.head(100)
```

Out[94]:

	ACC.NS	ABBOTINDIA.NS	ADANIGREEN.NS	ADANIPORTS.NS	ADANITRANS.NS	ALKEM
Date						
2018-02-02	NaN	NaN	NaN	NaN	NaN	N
2018-02-05	-0.023277	-0.025740	NaN	-0.023605	0.003091	-0.012
2018-02-06	-0.013619	-0.026972	NaN	-0.008364	-0.019479	-0.010
2018-02-07	-0.019680	0.019688	NaN	0.005420	0.014063	0.007
2018-02-08	0.064552	0.019985	NaN	-0.004679	0.021490	0.006
...	...	...	...	...	...	
2018-06-22	0.019184	0.016311	-0.006768	-0.003003	0.055152	0.004
2018-06-25	0.016014	0.029199	-0.017124	-0.012793	0.025287	-0.001
2018-06-26	0.031464	-0.021971	-0.005195	0.004834	-0.029871	-0.002
2018-06-27	-0.036608	-0.041388	-0.049832	-0.004972	-0.071731	-0.002
2018-06-28	-0.013422	0.018219	-0.050525	0.001937	-0.036185	-0.015

100 rows × 82 columns

In [93]:
```
l_ret2 = l_ret.dropna()
l_ret2.rows
```

Out[93]:

	ACC.NS	ABBOTINDIA.NS	ADANIGREEN.NS	ADANIPORTS.NS	ADANITRANS.NS	ALKEM
Date						
2018-08-07	0.009567	-0.009719	-0.051078	-0.065118	0.029397	-0.020
2018-08-08	-0.001529	0.000572	-0.050802	0.006159	-0.022692	-0.001
2018-08-09	0.022176	-0.015419	-0.051134	0.009035	-0.021437	-0.011
2018-08-10	-0.016335	0.009707	0.048109	0.003433	-0.027077	-0.024
2018-08-13	-0.014700	-0.013224	-0.051293	-0.005684	-0.015986	-0.029

5 rows × 82 columns

In [96]:
```
l_ret2.count()
```

Out[96]:
```
ACC.NS           609
ABBOTINDIA.NS    609
ADANIGREEN.NS    609
ADANIPORTS.NS    609
ADANITRANS.NS    609
                ...
PFC.NS           609
POWERGRID.NS     609
PGHH.NS          609
PNB.NS           609
RELIANCE.NS      609
Length: 82, dtype: int64
```

In [98]:
```
l_ret2.shape
```

Out[98]:
```
(609, 82)
```

Mean log returns

In [102]:

```
a_ret = l_ret2.mean()
a_ret
```

Out[102]:

```
ACC.NS            0.000240
ABBOTINDIA.NS     0.000963
ADANIGREEN.NS     0.004330
ADANIPORTS.NS     0.000531
ADANITRANS.NS     0.001700
                    ...
PFC.NS            0.000626
POWERGRID.NS      0.000227
PGHH.NS           0.000135
PNB.NS           -0.001517
RELIANCE.NS       0.000791
Length: 82, dtype: float64
```

In [106]:

```
a_ret.shape
```

Out[106]:

```
(82,)
```

Annualized Returns

In [127]:

```
ann_ret = a_ret * 252
ann_ret
```

Out[127]:

```
ACC.NS            0.060572
ABBOTINDIA.NS     0.242738
ADANIGREEN.NS     1.091237
ADANIPORTS.NS     0.133799
ADANITRANS.NS     0.428497
                    ...
PFC.NS            0.157689
POWERGRID.NS      0.057275
PGHH.NS           0.034056
PNB.NS           -0.382268
RELIANCE.NS       0.199426
Length: 82, dtype: float64
```

Equal Weights

```
In [128]: weights = np.full((1,82),1/82)
          weights
```

```
Out[128]: array([[0.01219512, 0.01219512, 0.01219512, 0.01219512, 0.01219512,
                  0.01219512, 0.01219512, 0.01219512, 0.01219512, 0.01219512,
                  0.01219512, 0.01219512, 0.01219512, 0.01219512, 0.01219512,
                  0.01219512, 0.01219512, 0.01219512, 0.01219512, 0.01219512,
                  0.01219512, 0.01219512, 0.01219512, 0.01219512, 0.01219512,
                  0.01219512, 0.01219512, 0.01219512, 0.01219512, 0.01219512,
                  0.01219512, 0.01219512, 0.01219512, 0.01219512, 0.01219512,
                  0.01219512, 0.01219512, 0.01219512, 0.01219512, 0.01219512,
                  0.01219512, 0.01219512, 0.01219512, 0.01219512, 0.01219512,
                  0.01219512, 0.01219512, 0.01219512, 0.01219512, 0.01219512,
                  0.01219512, 0.01219512, 0.01219512, 0.01219512, 0.01219512,
                  0.01219512, 0.01219512, 0.01219512, 0.01219512, 0.01219512,
                  0.01219512, 0.01219512, 0.01219512, 0.01219512, 0.01219512,
                  0.01219512, 0.01219512, 0.01219512, 0.01219512, 0.01219512,
                  0.01219512, 0.01219512, 0.01219512, 0.01219512, 0.01219512,
                  0.01219512, 0.01219512, 0.01219512, 0.01219512, 0.01219512,
                  0.01219512, 0.01219512]])
```

```
In [129]: weights.shape
```

```
Out[129]: (1, 82)
```

```
In [145]: p_ret = np.dot(weights,ann_ret).item()
          round(p_ret, 4)
```

```
Out[145]: 0.096
```

```
In [154]: p_var = np.dot(weights,np.dot(l_ret2.cov()*252, weights.T)).item()
          round(p_var,4)
```

```
Out[154]: 0.0477
```

Equally weighted portfolio returns and Variance

```
In [155]: portfolio_returns = str(round(p_ret, 4) * 100) + '%'
          print(portfolio_returns)
```

```
9.6%
```

```
In [153]: portfolio_variance = str(round(p_var, 4) * 100) + '%'
          print(portfolio_variance)
```

```
4.77%
```

Optimization of portfolio using Sharpe Ratio

In [173]:
```
np.random.seed(101)
num_ports=10000
#np zeroes: Return a new array of given shape and type, filled with zeros.
all_weights=np.zeros((num_ports,len(nif3.columns)))
ret_array=np.zeros(num_ports)
vol_array=np.zeros(num_ports)
sr_array=np.zeros(num_ports)

#np.dot:Dot product of two arrays
for i in range(num_ports):
    weights=np.array(np.random.random(82))
    weights=weights/np.sum(weights)
    all_weights[i,:] = weights
    ret_array[i]=np.sum(l_ret.mean()*weights*252)
    vol_array[i] = np.sqrt(np.dot(weights, np.dot(l_ret.cov() * 252, weights.T
)))
    sr_array[i]=ret_array[i]/vol_array[i]
```

In [179]:
```
sr_array.max()
```

Out[179]:
```
0.7706859746065876
```

In [180]:
```
sr_array.argmax()
```

Out[180]:
```
3102
```

Optimal Portfolio Weights

In [181]:
```
all_weights[3102,:]
```

Out[181]:
```
array([0.00333317, 0.02017557, 0.02107545, 0.01416869, 0.02180399,
       0.00153266, 0.02590597, 0.02241389, 0.00241168, 0.0077785 ,
       0.02495771, 0.02347711, 0.0092581 , 0.02147628, 0.00750247,
       0.00615901, 0.00184109, 0.02721361, 0.00453808, 0.0008369 ,
       0.01411182, 0.00657939, 0.01537755, 0.02571603, 0.01781174,
       0.00522274, 0.00889654, 0.00204158, 0.00292056, 0.01007755,
       0.02183637, 0.02559264, 0.02405584, 0.009058  , 0.01962202,
       0.02743757, 0.00980499, 0.01441451, 0.00944886, 0.01049714,
       0.00770051, 0.00279661, 0.01822779, 0.00635191, 0.014728  ,
       0.02144023, 0.02396875, 0.02244952, 0.01186999, 0.01583287,
       0.01591403, 0.00515882, 0.0070739 , 0.01059775, 0.00188152,
       0.00790996, 0.02114156, 0.00998756, 0.01515489, 0.00842781,
       0.01580932, 0.00247235, 0.00395173, 0.01088278, 0.00074869,
       0.01238925, 0.00657293, 0.00243438, 0.02036036, 0.0095927 ,
       0.00225274, 0.01455513, 0.00868327, 0.01693556, 0.00905796,
       0.01972236, 0.00690206, 0.00373103, 0.02328824, 0.00626483,
       0.00282683, 0.00356816])
```

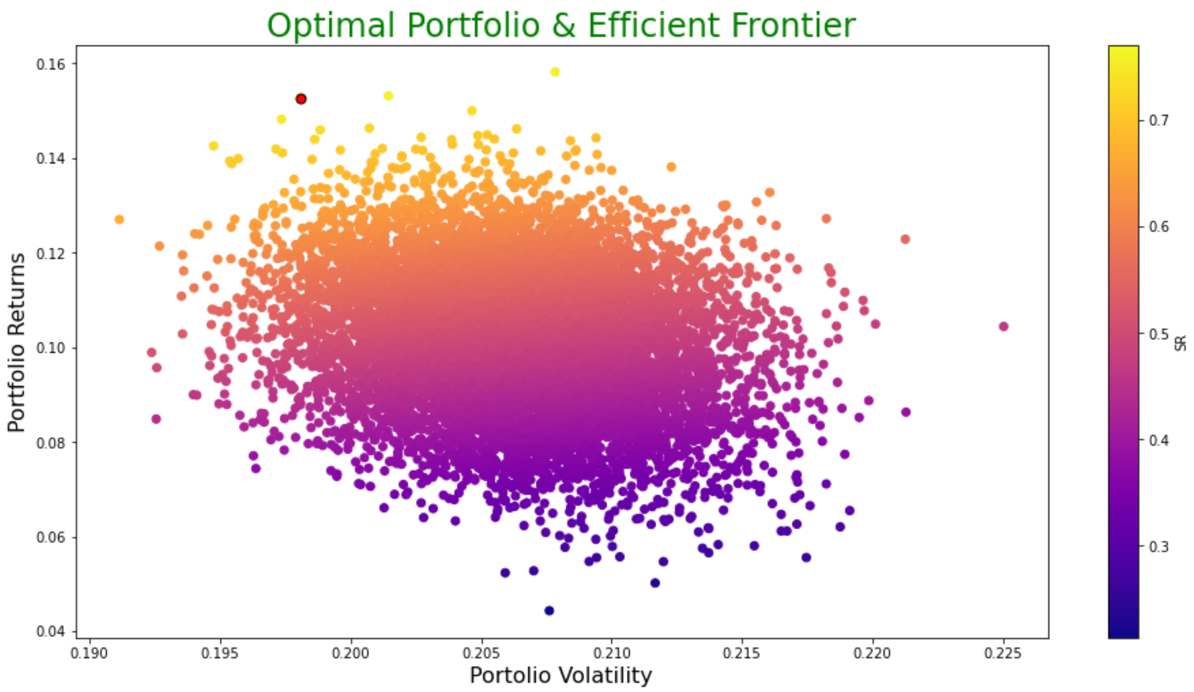
In [188]:
```
max_sr_ret = ret_array[3102]
max_sr_vol = vol_array[3102]
```

In [200]:

```
plt.figure(figsize=(16,8))
plt.scatter(vol_array,ret_array,c=sr_array, cmap='plasma')
plt.colorbar(label='SR')
plt.title('Optimal Portfolio & Efficient Frontier', fontsize=24, color='Green'
)
plt.xlabel('Portolio Volatility', fontsize=16)
plt.ylabel('Portfolio Returns', fontsize=16)
plt.scatter(max_sr_vol,max_sr_ret,c='red',s=50,edgecolors='black')
```

Out[200]:

```
<matplotlib.collections.PathCollection at 0x241662740a0>
```



In [190]:

```
max_sr_ret
```

Out[190]:

```
0.1526562895491424
```

In [191]:

```
max_sr_vol
```

Out[191]:

```
0.19807845812565736
```

Final Ouput

Optimal Portfolio Returns and Volatiltity

In [192]:

```
Optimal_portfolio_returns = str(round(max_sr_ret, 4) * 100) + '%'
print(Optimal_portfolio_returns)
```

```
15.27%
```

In [193]:

```
Optimal_portfolio_volatility = str(round(max_sr_vol, 4) * 100) + '%'
print(Optimal_portfolio_volatility)
```

```
19.81%
```

THE END