In [1]:
```
import pandas as pd
import datetime as dt
import matplotlib.pyplot as plt
import numpy as np
import pandas_datareader as pdr
import scipy.stats
import yfinance as yf
```

In [2]:
```
import plotly
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import plotly.express as px
import plotly.figure_factory as ff
```

Step 1&2: Retrieve Data

In [3]:
```
tickers = ['AAPL', 'NKE', 'GOOGL', 'AMZN', 'TSLA', 'PFE', 'ACLS']
start = dt.datetime(2012, 10, 25)
end = dt.datetime(2022, 10, 25)

data = pdr.get_data_yahoo(tickers, start, end, interval="d")
data_returns = data['Close'].ffill().pct_change()
```

Step 3: Mean, Variance, and Correlation Matrix

In [4]:
```
data_returns_mean = data['Close'].ffill().pct_change().mean()

print(data_returns_mean)
```

```
Symbols
AAPL     0.000941
NKE      0.000704
GOOGL    0.000861
AMZN     0.001151
TSLA     0.002540
PFE      0.000346
ACLS     0.001633
dtype: float64
```

Variance

In [5]:
```
data_returns_var = data['Close'].ffill().pct_change().var()
print(data_returns_var)
```

```
Symbols
AAPL     0.000333
NKE      0.000303
GOOGL    0.000278
AMZN     0.000409
TSLA     0.001271
PFE      0.000191
ACLS     0.001057
dtype: float64
```

Correlation Matrix

In [6]:
```
tickers = ['AAPL', 'NKE', 'GOOGL', 'AMZN', 'TSLA', 'PFE', 'ACLS']
data_cor = yf.download(tickers, start = "2012-10-25", end = "2022-10-25")
cor_matrix = data_cor['Close'].pct_change()[1:]
cor_matrix_vis = cor_matrix.corr(method='pearson')
print(cor_matrix_vis)
```

```
[*********************100%***********************]  7 of 7 completed
           AAPL      ACLS      AMZN     GOOGL       NKE       PFE       TS
LA
AAPL   1.000000  0.404270  0.498817  0.559896  0.427635  0.296673  0.3628
84
ACLS   0.404270  1.000000  0.364628  0.443353  0.321005  0.219188  0.3223
39
AMZN   0.498817  0.364628  1.000000  0.621871  0.396396  0.240308  0.3687
69
GOOGL  0.559896  0.443353  0.621871  1.000000  0.472447  0.339341  0.3613
66
NKE    0.427635  0.321005  0.396396  0.472447  1.000000  0.287800  0.2934
93
PFE    0.296673  0.219188  0.240308  0.339341  0.287800  1.000000  0.1312
33
TSLA   0.362884  0.322339  0.368769  0.361366  0.293493  0.131233  1.0000
00
```

Step 4: Efficient Frontier (Simplified Version)

In [7]:
```
n_assets = 7
n_portfolios = 1000
mean_variance_pairs = []
mus = (1+data_returns.mean())**252 - 1
cov = data_returns.cov()*252
np.random.seed(75)

for i in range(n_portfolios):
    assets = np.random.choice(list(data_returns.columns), n_assets, repla
    weights = np.random.rand(n_assets)
    weights = weights/sum(weights)

    portfolio_E_Variance = 0
    portfolio_E_Return = 0
    for i in range(len(assets)):
        portfolio_E_Return += weights[i] * mus.loc[assets[i]]
        for j in range(len(assets)):
            portfolio_E_Variance += weights [i] * weights[j] * cov.loc[as
    mean_variance_pairs.append([portfolio_E_Return, portfolio_E_Variance,
```

In [8]:
```
mean_variance_pairs = np.array(mean_variance_pairs)
risk_free_rate=0.0275

fig = go.Figure()
fig.add_trace(go.Scatter(x=mean_variance_pairs[:,1]**0.5,
                         y=mean_variance_pairs[:,0],
                     marker=dict(color=(mean_variance_pairs[:,0]-risk_fre
                                 showscale=True,
                                 size=7,
                                 line=dict(width=1),
                                 colorscale="RdBu",
                                 colorbar=dict(title="Sharpe<br>Ratio")
                                ),
                         mode='markers'))
fig.update_layout(template='plotly_white',
                  xaxis=dict(title='Annualised Risk (Volatility)'),
                  yaxis=dict(title='Annualised Return'),
                  title='Sample of Random Portfolios',
                  coloraxis_colorbar=dict(title="Sharpe Ratio"))
```

```
/var/folders/z2/909fbvdx5n1f_z6b8_t_fkmm0000gn/T/ipykernel_27623/21564617
3.py:1: VisibleDeprecationWarning: Creating an ndarray from ragged nested
sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with d
ifferent lengths or shapes) is deprecated. If you meant to do this, you m
ust specify 'dtype=object' when creating the ndarray.
  mean_variance_pairs = np.array(mean_variance_pairs)
```

Step 4: Efficient Frontier: Detailed Version

In [9]:
```
ann_sd= data['Close'].ffill().pct_change().apply(lambda x: np.log(1+x)).s
ann_sd
mus = (1+data_returns.mean())**252 - 1
mus
cov_matrix = data_returns.apply(lambda x: np.log(1+x)).cov()
cov_matrix
```

Out[9]:

Symbols	AAPL	NKE	GOOGL	AMZN	TSLA	PFE	ACLS
Symbols							
AAPL	0.000334	0.000137	0.000171	0.000185	0.000240	0.000075	0.000242
NKE	0.000137	0.000301	0.000139	0.000141	0.000185	0.000070	0.000183
GOOGL	0.000171	0.000139	0.000276	0.000210	0.000218	0.000079	0.000243
AMZN	0.000185	0.000141	0.000210	0.000408	0.000268	0.000068	0.000241
TSLA	0.000240	0.000185	0.000218	0.000268	0.001258	0.000067	0.000382
PFE	0.000075	0.000070	0.000079	0.000068	0.000067	0.000190	0.000099
ACLS	0.000242	0.000183	0.000243	0.000241	0.000382	0.000099	0.001070

In [10]:

```
assets = pd.concat([mus, ann_sd], axis =1)
assets.columns = ['Returns', 'Volatility']
assets
```

Out[10]:

	Returns	Volatility
Symbols		
AAPL	0.267340	0.288968
NKE	0.194030	0.274340
GOOGL	0.242346	0.262704
AMZN	0.336349	0.319196
TSLA	0.895131	0.560797
PFE	0.090966	0.218212
ACLS	0.508503	0.517218

In [11]:

```
p_ret = []
p_vol = []
p_weights = []

num_assets = len(data_returns.columns)
num_portfolios = 10000
```

In [12]:

```
for portfolio in range (num_portfolios):
    weights = np.random.random(num_assets)
    weights = weights/np.sum(weights)
    p_weights.append(weights)
    returns = np.dot(weights, mus)

    p_ret.append(returns)
    var = cov_matrix.mul(weights, axis = 0).mul(weights, axis = 1).sum().
    sd = np.sqrt(var)
    ann_sd = sd*np.sqrt(252)
    p_vol.append(ann_sd)
```

In [13]:
```
df = {'Returns' :p_ret, 'Volatility':p_vol}

for counter, symbol in enumerate(data_returns.columns.tolist()):
    df[symbol + ' weight'] = [w[counter] for w in p_weights]
```

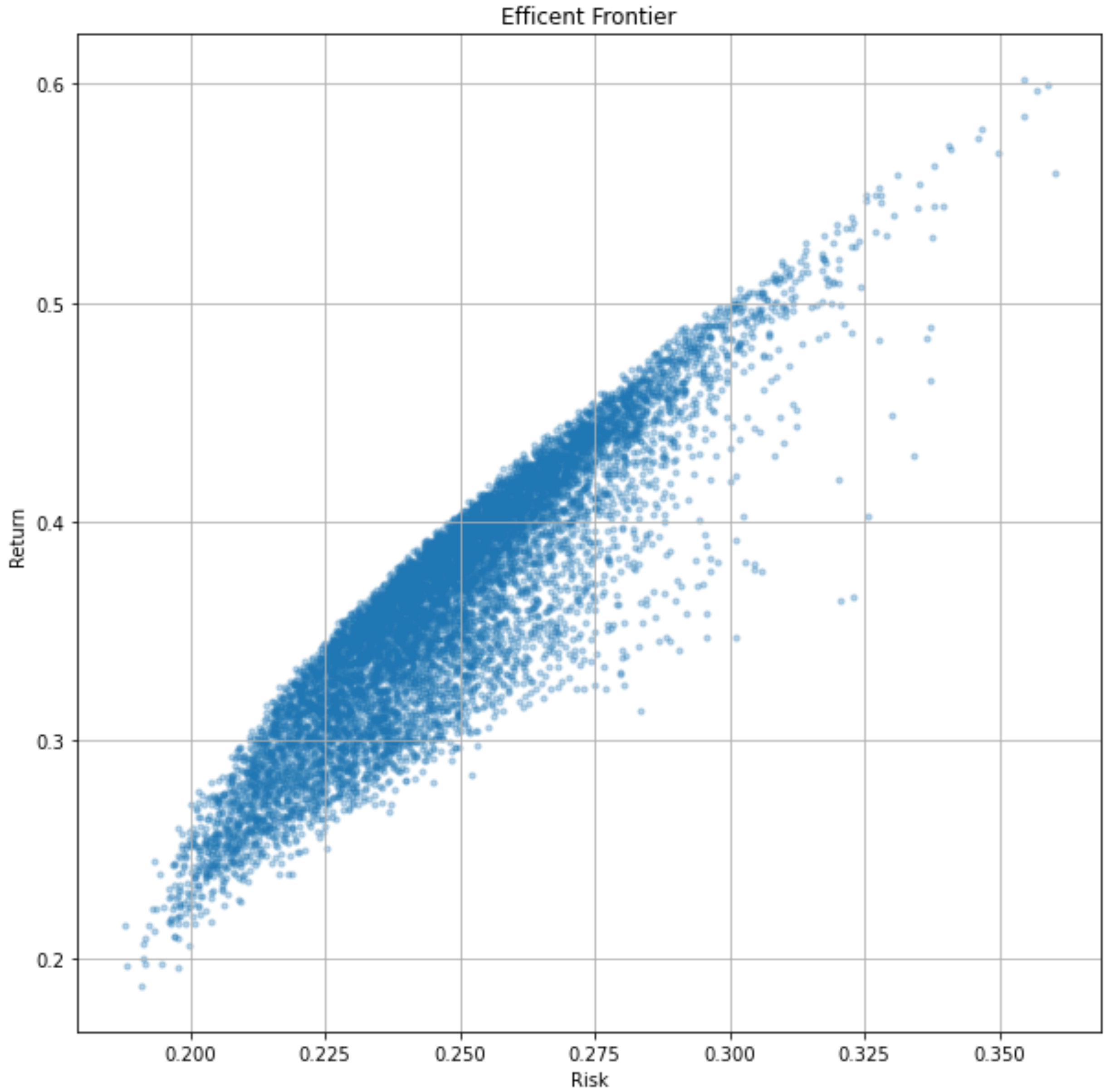
In [14]:
```
portfolios = pd.DataFrame(df)
portfolios.head()
```

Out[14]:

	Returns	Volatility	AAPL weight	NKE weight	GOOGL weight	AMZN weight	TSLA weight	PFE weight	
0	0.325611	0.220780	0.205724	0.163939	0.110254	0.173395	0.148859	0.191797	0.
1	0.496718	0.303892	0.017665	0.097469	0.090482	0.196819	0.284607	0.069322	0.:
2	0.400873	0.257015	0.240000	0.169840	0.060702	0.193410	0.180702	0.040122	0
3	0.385626	0.247226	0.320837	0.142570	0.034699	0.056887	0.221369	0.161009	0.(
4	0.406943	0.257114	0.163985	0.164561	0.054472	0.170643	0.195388	0.100424	0.

In [15]:
```
portfolios.plot.scatter(x='Volatility', y='Returns', marker='o', s=10, al
```

Out[15]:
```
<AxesSubplot:title={'center':'Efficent Frontier'}, xlabel='Risk', ylabel=
'Return'>
```



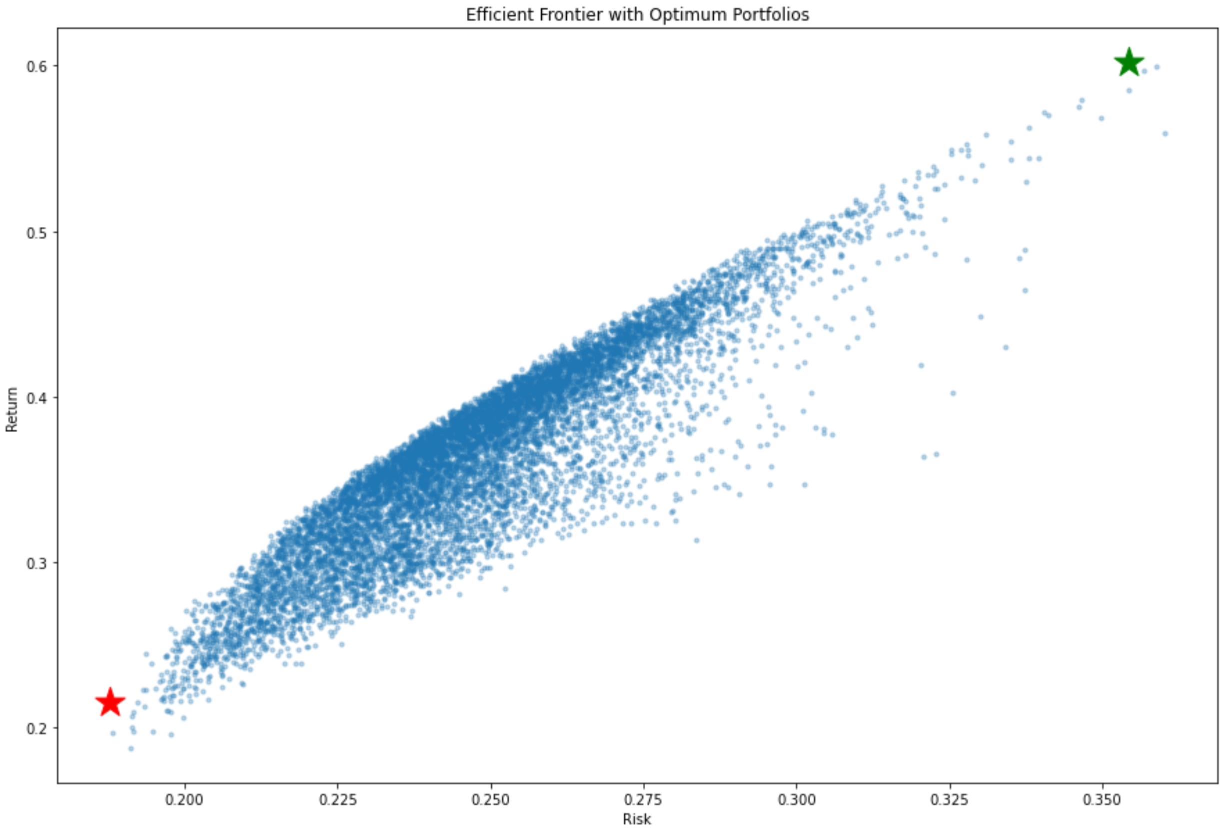
In [16]:
```
min_vol_port = portfolios.iloc[portfolios['Volatility'].idxmin()]
```

In [17]:
```
max_ret_port = portfolios.iloc[portfolios['Returns'].idxmax()]
```

In [18]:
```
rf = 0.03
optimal_risky_port = portfolios.iloc[((portfolios['Returns']-rf)/portfoli
```

In [19]:
```
plt.subplots(figsize=(15, 10))
plt.scatter(portfolios['Volatility'], portfolios['Returns'],marker='o', s
plt.scatter(min_vol_port[1], min_vol_port[0], color='r', marker='*', s=50
plt.scatter(optimal_risky_port[1], optimal_risky_port[0], color='g', mark
plt.xlabel('Risk')
plt.ylabel('Return')
plt.title('Efficient Frontier with Optimum Portfolios')
```

Out[19]:
```
Text(0.5, 1.0, 'Efficient Frontier with Optimum Portfolios')
```



In [20]:
```
vis_portfolio = pd.concat([min_vol_port, max_ret_port, optimal_risky_port
vis_portfolio.columns = ['Minimum Risk', 'Maximum Return', 'Optimal Portf
print(vis_portfolio)
```

```
             Minimum Risk  Maximum Return  Optimal Portfolio
Returns          0.215320        0.602147           0.602147
Volatility       0.187776        0.354436           0.354436
AAPL weight      0.068284        0.162180           0.162180
NKE weight       0.188374        0.016303           0.016303
GOOGL weight     0.116449        0.034253           0.034253
AMZN weight      0.105693        0.175499           0.175499
TSLA weight      0.058610        0.476744           0.476744
PFE weight       0.457320        0.017029           0.017029
ACLS weight      0.005269        0.117992           0.117992
```