Project: Portfolio Diversification

Name: Donald Shi

In [1]:
```
from IPython.display import display, Math, Latex
import pandas as pd
import yfinance as yf
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
new_df = pd.read_csv("industries.csv")
student_id = 21060501
picked_stocks = ['CSU.TO', 'GIB-A.TO', 'CTS.TO', 'KXS.TO', 'BB.TO']
starting = "2019-01-01"
ending = "2021-12-02"
data = yf.download(picked_stocks, start = starting, end = ending, interval =
data.index = data.index.strftime("%Y-%m-%d")
data = data.dropna()
data.head()
```

```
[*********************100%***********************]  5 of 5 completed
```

Out[2]:

Ticker	BB.TO	CSU.TO	CTS.TO	GIB-A.TO	KXS.TO
Date					
2019-01-01	10.58	980.619995	0.60	86.870003	78.129997
2019-02-01	11.44	1123.729980	0.56	88.230003	82.730003
2019-03-01	13.47	1132.500000	0.85	91.870003	77.970001
2019-04-01	12.29	1182.069946	1.12	96.430000	73.209999
2019-05-01	10.61	1170.359985	0.90	98.519997	78.309998

In [3]:
```
investment = 10000
invest_per_stock = investment / len(picked_stocks)
portfolio = data / data.iloc[0] * invest_per_stock
portfolio["Total Value"] = portfolio.sum(axis = 1)
portfolio.head()
```

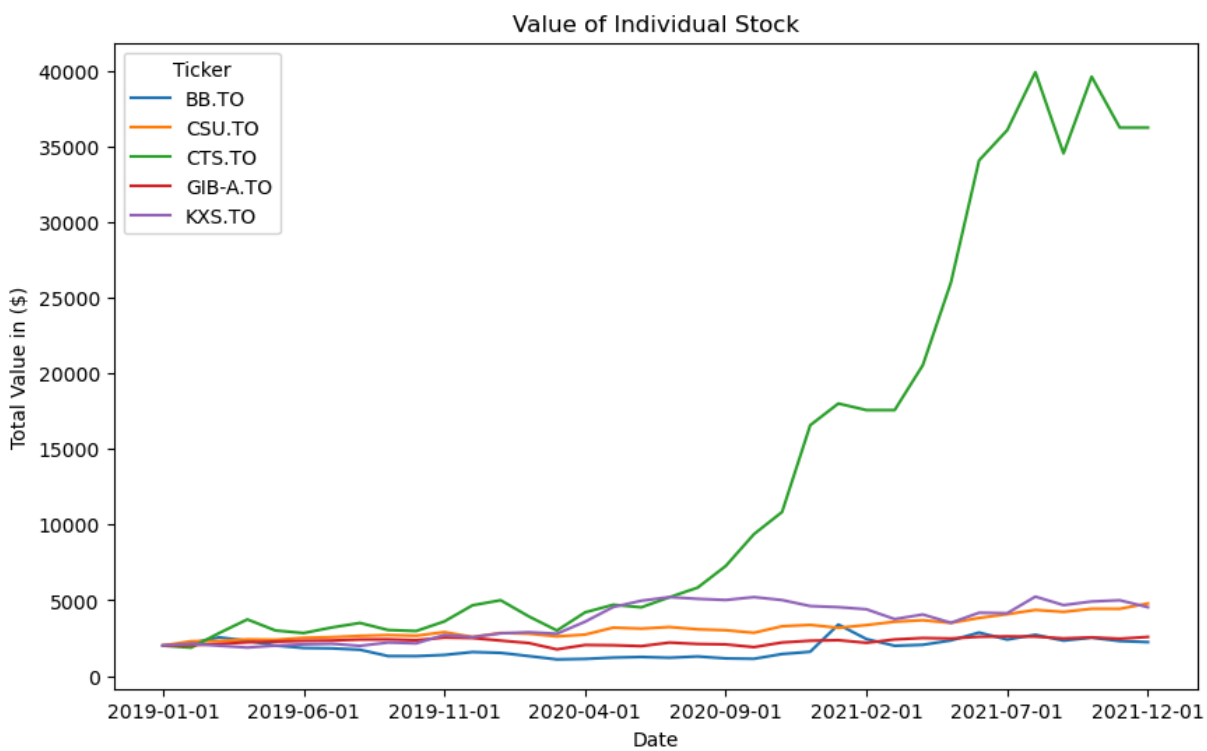
Out[3]:

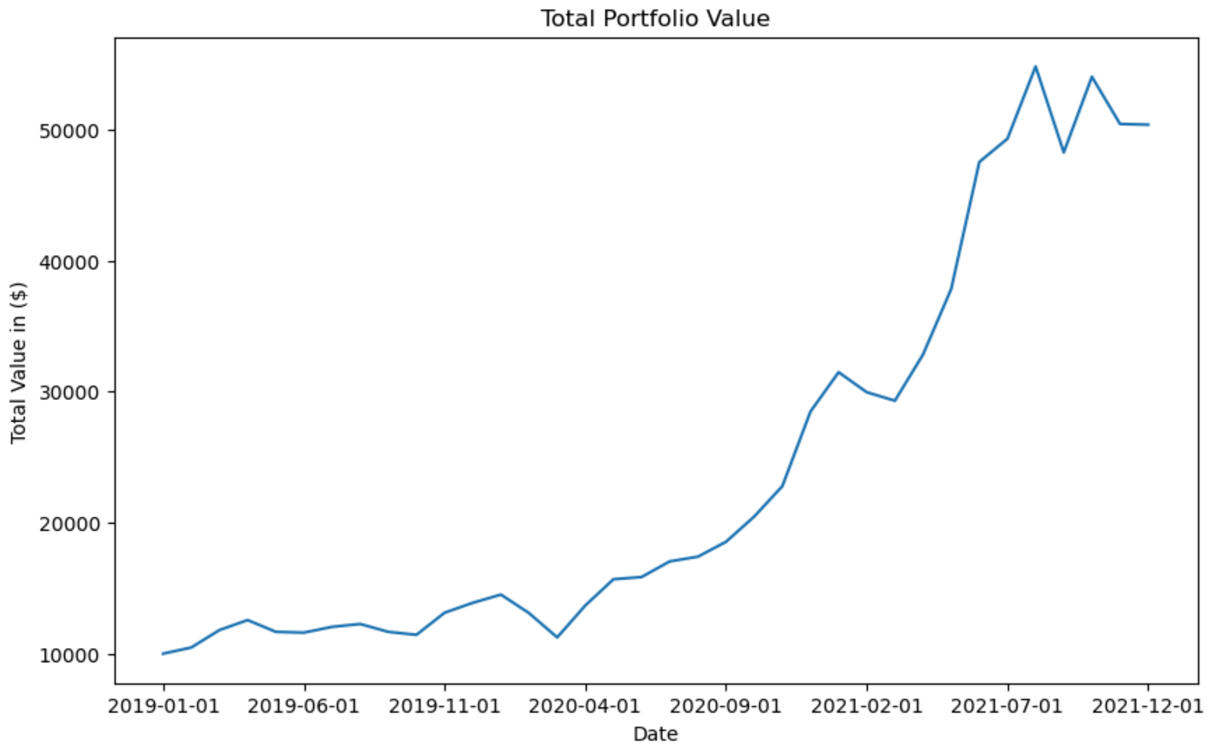
Ticker	BB.TO	CSU.TO	CTS.TO	GIB-A.TO	KXS.TO	Total
Date						
2019-01-01	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	10000.00
2019-02-01	2162.570825	2291.876539	1866.666600	2031.311168	2117.752624	10470.17
2019-03-01	2546.313868	2309.763223	2833.333300	2115.114535	1995.904364	11800.42
2019-04-01	2323.251427	2410.862418	3733.333201	2220.098935	1874.056103	12561.60
2019-05-01	2005.671027	2386.979648	2999.999801	2268.216727	2004.607713	11665.47

In [4]:

```
plt.figure(figsize = (10,6))
portfolio.iloc[:, :-1].plot(ax=plt.gca())
plt.ylabel("Total Value in ($)")
plt.title("Value of Individual Stock")


plt.figure(figsize = (10,6))
portfolio.iloc[:,-1].plot(ax=plt.gca())
plt.ylabel("Total Value in ($)")
plt.title("Total Portfolio Value")
```

Out[4]: `Text(0.5, 1.0, 'Total Portfolio Value')`





In [5]:
```
returns = portfolio.pct_change()
returns *=100
returns = returns.drop(returns.index[0])
returns.head()
```

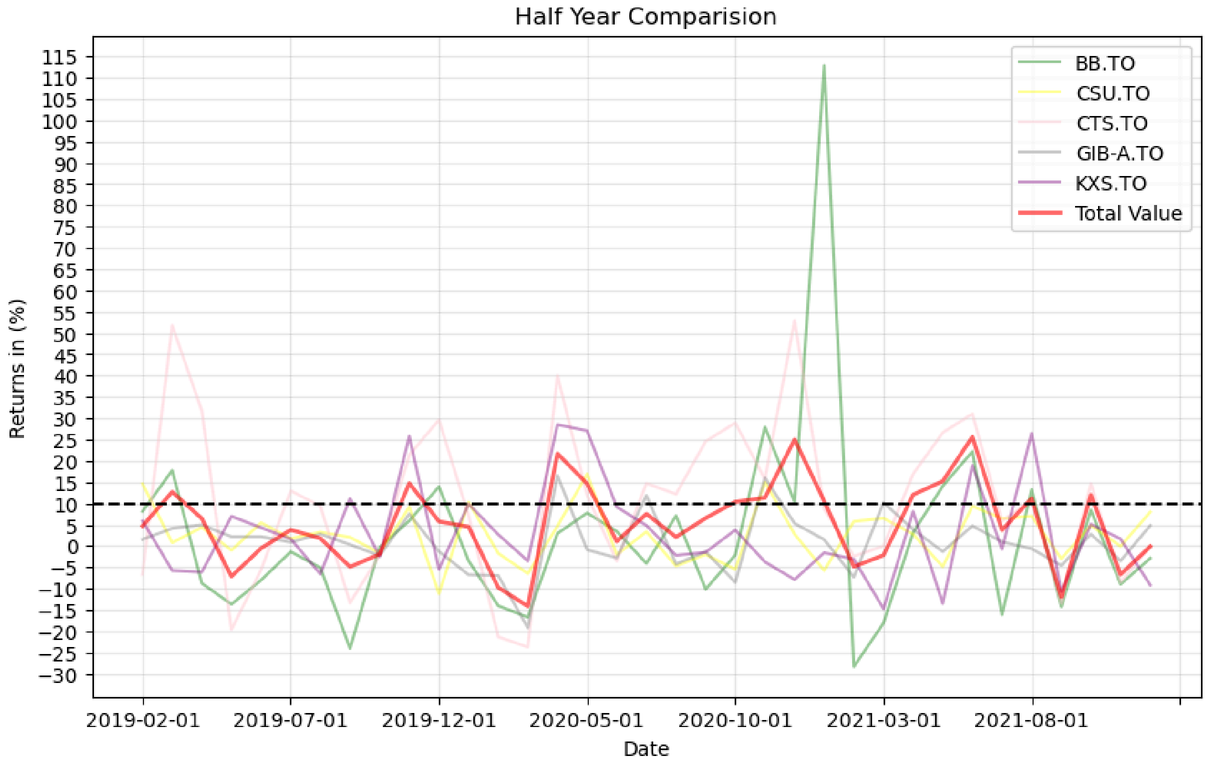
Out[5]:

Ticker Date	BB.TO	CSU.TO	CTS.TO	GIB-A.TO	KXS.TO	Total Value
2019-02-01	8.128541	14.593827	-6.666670	1.565558	5.887631	4.701778
2019-03-01	17.744762	0.780438	51.785718	4.125580	-5.753659	12.705148
2019-04-01	-8.760210	4.377037	31.764703	4.963533	-6.104915	6.450382
2019-05-01	-13.669653	-0.990632	-19.642860	2.167371	6.966259	-7.133860
2019-06-01	-8.011305	5.459005	-5.555550	2.192452	4.290641	-0.525495

In [6]:
```
plt.figure(figsize = (10,6))
colour = ["green", "yellow", "pink", "grey", "purple"]
index = 0
for col in returns.columns:
    if col != "Total Value":
        returns[col].plot(ax = plt.gca(), color = colour[index], alpha = 0.4
        index +=1
    else:
        returns[col].plot(ax = plt.gca(), color = 'red', alpha = 0.6, label

plt.grid(alpha = 0.3)
plt.ylabel("Returns in (%)")
plt.title("Half Year Comparision")
plt.axhline(y = 10, color= 'black', linestyle= '--')
```

```
plt.yticks(range(-30, 120, 5))
plt.legend()
```

Out[6]: <matplotlib.legend.Legend at 0x311ce1290>



In [7]:

```
print('Returns')
print(returns.mean())
print("\n")

print("Volatility")
print(returns.std())

print("\n")

# Compare each individual stock and the portfolio with the Technology Sector
tech = "XLK"
tech_data = yf.download(tech, start = starting, end = ending, interval = "1m
tech_data_returns = tech_data.pct_change()
tech_data_returns *= 100
tech_data_returns = tech_data_returns.drop(tech_data_returns.index[0])
# Beta: Cov(Stock, Market)/Var(Market)

# Converting the index to the same type for Covariance Calculations
returns.index = pd.to_datetime(returns.index)
tech_data_returns.index = pd.to_datetime(tech_data_returns.index)

print("Beta: ")
for col in returns.columns:
    if col == "Total Value":
        print("This is the formed Portfolio: ")
    covariance = returns[col].cov(tech_data_returns)
```

```
    beta = np.round(covariance / tech_data_returns.var(), 4)
    print("Stock : {}\nBeta: {}".format(col, beta))
```

```
[*********************100%***********************]  1 of 1 completed
Returns
Ticker
BB.TO           2.242746
CSU.TO          2.720265
CTS.TO         10.252164
GIB-A.TO        0.957097
KXS.TO          2.946714
Total Value     5.169756
dtype: float64


Volatility
Ticker
BB.TO          23.232901
CSU.TO          6.441887
CTS.TO         19.228485
GIB-A.TO        6.900518
KXS.TO         11.334195
Total Value     9.769559
dtype: float64


Beta:
Stock : BB.TO
Beta: 0.8214
Stock : CSU.TO
Beta: 0.5567
Stock : CTS.TO
Beta: 1.4611
Stock : GIB-A.TO
Beta: 0.7082
Stock : KXS.TO
Beta: 0.6635
This is the formed Portfolio:
Stock : Total Value
Beta: 0.9305
```

Observations:

1. Diversification helps reducing the overall risk. The portfolio has a beta of 0.93, which indicates that the portfolio reduces less risk compared to the technology sector ETF but with similiar market exposure This portfolio who wants market partipation but also risk adverse.
2. Volatility measures both idiosyncratic risk and systematic risk diversification only helps to balance out the idiosyncratic risk, from the volatilities calcualted, the portfolio (total value) has a relative lower volatility compared to others
3. High Beta stock like BB.TO and CTS.TO should be closely monitored due to high volatility as they have

a bigger impact on the portfolio value when market fluctuations are volatile.

In [8]:
```
%%latex
\newpage
```

\newpage

In [9]:
```
def function_get_data(stock_list):
    new_data = yf.download(stock_list, start = starting, end = ending, inter
    new_data.index = new_data.index.strftime("%Y-%m-%d")
    new_data = new_data.dropna()
    return new_data
```

In [10]:
```
# Stocks picked from each sector:

#Industrials
industrials_sector  = ['AC.TO', 'TIH.TO', 'TFII.TO', 'CAE.TO', 'WCN.TO']
industrails_data = function_get_data(industrials_sector)

#Healthcare
healthcare_sector  = ['EXE.TO', 'OGI.TO', 'GUD.TO', 'WEED.TO', 'WELL.TO']
healthcare_data = function_get_data(healthcare_sector)

#Energy
energy_sector  = ['PPL.TO', 'POU.TO', 'CNQ.TO', 'MEG.TO', 'IMO.TO']
energy_data = function_get_data(energy_sector)

#Financial Services
finance_sector  = ['CM.TO', 'MFC.TO', 'SLF.TO', 'BNS.TO', 'FFH.TO']
finance_data = function_get_data(finance_sector)
```

```
[*********************100%***********************]  5 of 5 completed
[*********************100%***********************]  5 of 5 completed
[*********************100%***********************]  5 of 5 completed
[*********************100%***********************]  5 of 5 completed
```

In [11]:
```
# Adding Sectors into Portfolios
data_list = [data, industrails_data, healthcare_data, energy_data, finance_d
industries_tickers = [industrials_sector, healthcare_sector, energy_sector, 
names = ["Technology", "Industrials", "Healthcare", "Energy", "Financial Ser

def add_sector_to_portfolio(initial_investment, portfolio, industries, indus
    """
    Adds a new sector to our existing portfolio by keeping the initial_inves
    the function keeps adding data from the monthy_data_llist.

    add_sector_to_portfolio: Int Dataframe listof Str listof Str listof Data

    """
    updated_port = portfolio[["Total Value"]].rename(columns={"Total Value":
    tickers_combined = []
    combined_monthly_data = pd.DataFrame()
    for i in range(len(industries)):
        ticker = industries[i]
        monthly_data = monthly_data_list[i]
```

```
        # adding into the list
        tickers_combined.extend(ticker)
        combined_monthly_data = pd.concat([combined_monthly_data, monthly_da
        # Calculating the amount to invest
        investment_per_stock = initial_investment / len(tickers_combined)
        price_ratio = combined_monthly_data / combined_monthly_data.iloc[0]
        new_port = price_ratio * investment_per_stock
        col_name = ' & '.join(industry_names[:i + 1])
        updated_port[col_name] = new_port.sum(axis=1)
    return updated_port

integradeted_portfolio = add_sector_to_portfolio(10000, portfolio, industrie

integradeted_portfolio.head()
```

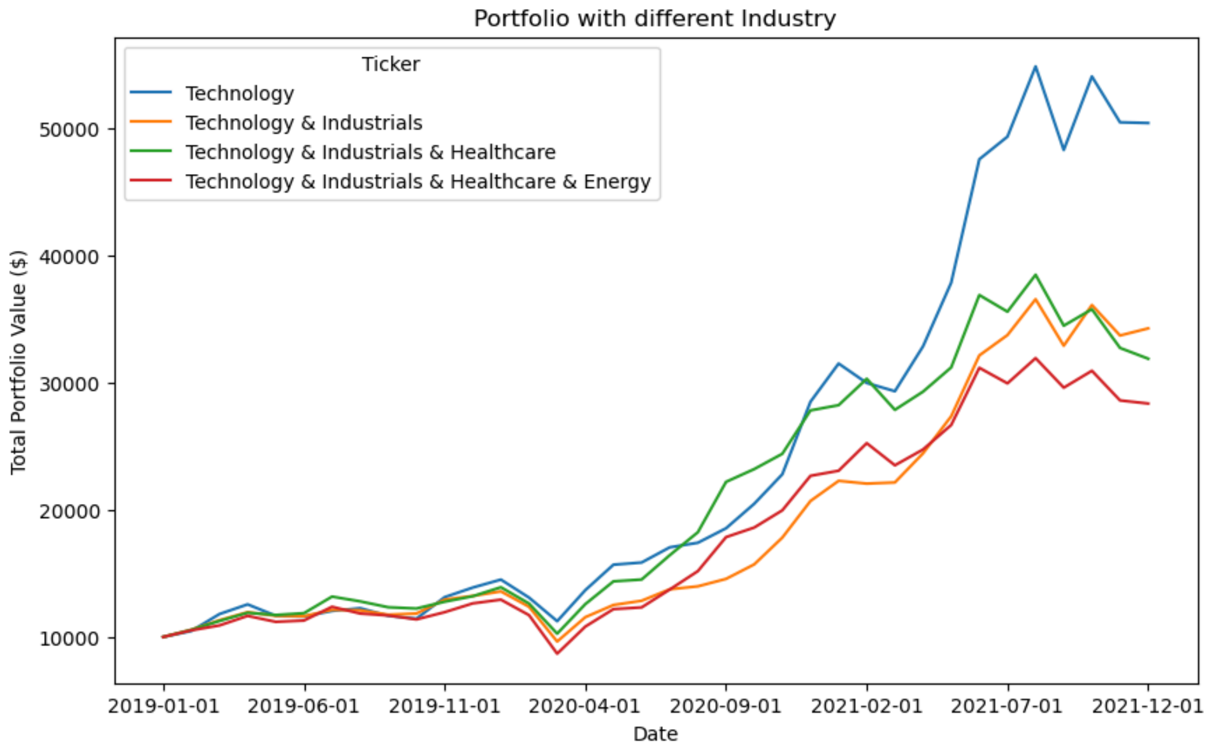
Out[11]:

Ticker Date	Technology	Technology & Industrials	Technology & Industrials & Healthcare	Technology & Industrials & Healthcare & Energy
2019-01-01	10000.000000	10000.000000	10000.000000	10000.000000
2019-02-01	10470.177757	10566.980739	10560.806493	10519.289699
2019-03-01	11800.429290	11315.167408	11258.381224	10906.396609
2019-04-01	12561.602084	11945.464221	11901.960471	11654.459435
2019-05-01	11665.474916	11660.912912	11727.420625	11188.983305

In [12]:

```
plt.figure(figsize = (10,6))
integradeted_portfolio.plot(ax = plt.gca())
plt.ylabel("Total Portfolio Value ($)")
plt.title("Portfolio with different Industry")
```

Out[12]: `Text(0.5, 1.0, 'Portfolio with different Industry')`



In [13]:
```
new_returns = integradeted_portfolio.pct_change()
new_returns *=100
new_returns = new_returns.drop(index = new_returns.index[0])
new_returns.head()
```

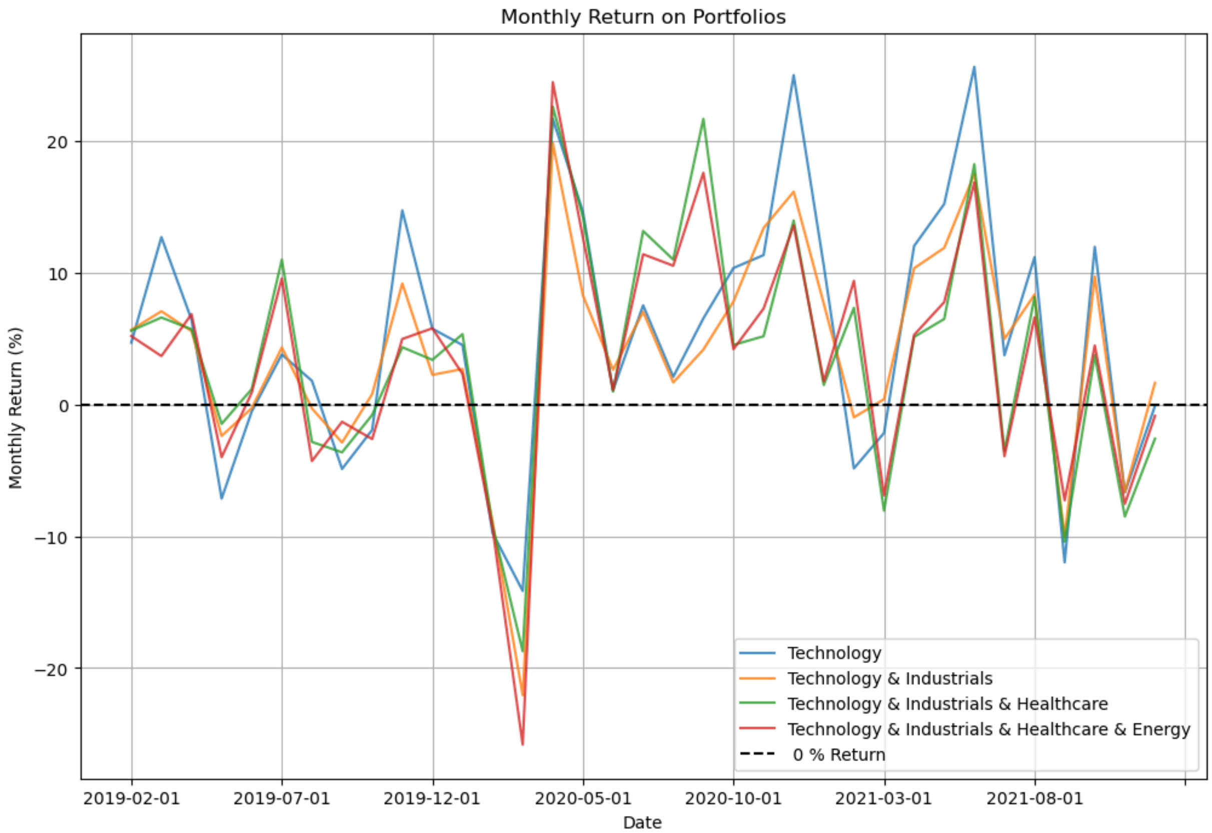
Out[13]:

Ticker Date	Technology	Technology & Industrials	Technology & Industrials & Healthcare	Technology & Industrials & Healthcare & Energy
2019-02-01	4.701778	5.669807	5.608065	5.192897
2019-03-01	12.705148	7.080420	6.605317	3.679972
2019-04-01	6.450382	5.570371	5.716446	6.858937
2019-05-01	-7.133860	-2.382087	-1.466480	-3.993974
2019-06-01	-0.525495	-0.270308	1.176162	0.903273

In [14]:
```
plt.figure(figsize=(12,8))
for col in new_returns.columns:
    new_returns[col].plot(ax = plt.gca(), label = col, alpha = 0.8)

plt.axhline(y=0, color='black', linestyle='--', label = " 0 % Return")
plt.ylabel("Monthly Return (%)")
plt.title("Monthly Return on Portfolios")
```

```
plt.grid()
plt.legend()
```

Out[14]: <matplotlib.legend.Legend at 0x311bc9350>



In [15]:
```
print("Mean: ")
print(new_returns.mean())
print()

print("Volatility: ")
print(new_returns.std())
```

```
Mean: 
Ticker
Technology                                        5.169756
Technology & Industrials                          3.906993
Technology & Industrials & Healthcare             3.754123
Technology & Industrials & Healthcare & Energy    3.440767
dtype: float64

Volatility: 
Ticker
Technology                                        9.769559
Technology & Industrials                          8.162624
Technology & Industrials & Healthcare             9.040895
Technology & Industrials & Healthcare & Energy    9.226686
dtype: float64
```

Note:

1. The portfolio made of sectors in Technology & Industrials has a average return of 3.9 % and a relatively low volatility of 8.16%

2. Risk Tolerant investors should choose Technology Portfolio with an average return of (5.17%)

3. Adding Industrials reduces volatility and adding Healthcare & Energy increases volatility.

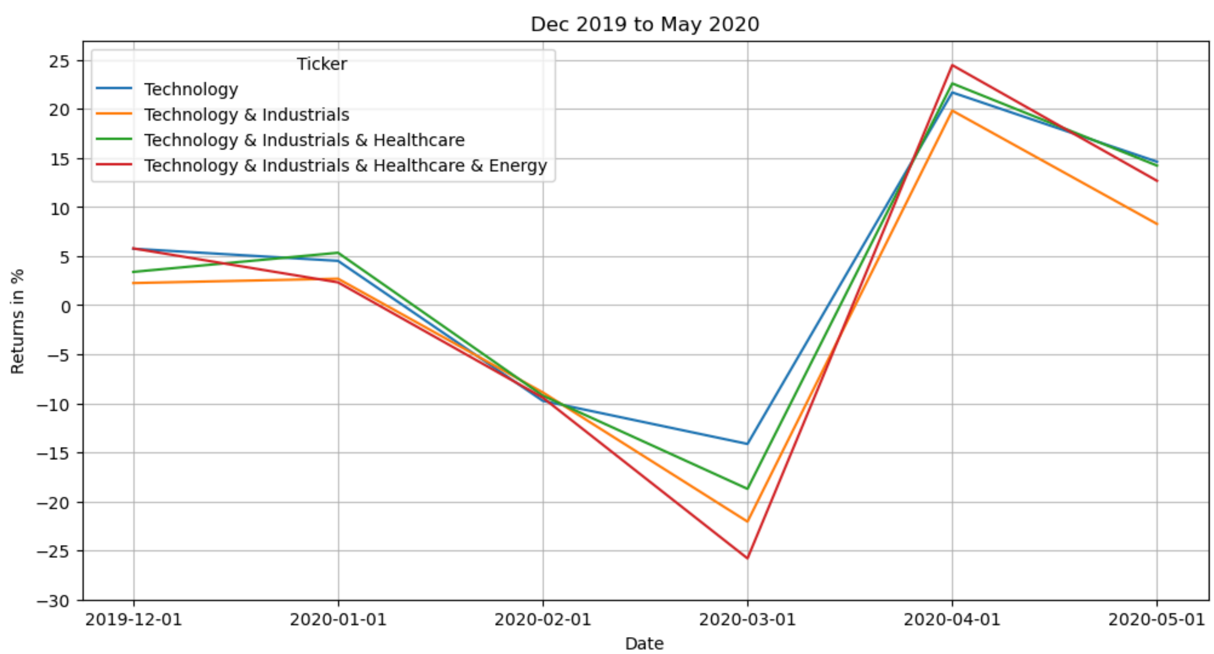
   - Industrail sector includes companies with steady cash flows and demand (infrastructure, manufacture, production)

   - Industrails have lower sensitivity to changes in economic conditions

   - Healthcare could be influenced by research & development, changes in policies (i.e covering a drug in insurance)

   - Energy is highly volatile and depends on oil & gas. - could be influenced by political, geological issues -> i.e War - Production costs

In [16]:
```
period_2019 = new_returns.loc["2019-12-01": "2020-05-01"]
plt.figure(figsize = (12,6))
period_2019.plot(ax=plt.gca())
plt.title("Dec 2019 to May 2020")
plt.ylabel("Returns in %")
plt.yticks(range(-30,30,5))
plt.grid(alpha = 0.8)
```



In [17]:
```
%%latex
\newpage
```

\newpage

In [18]:
```
bitcoin = "BTC-CAD"
etherium = "ETH-CAD"
bitcoin_data = yf.download(bitcoin, start = starting, end = ending, interval
etherium_data = yf.download(etherium, start = starting, end = ending, interv
combined_crypto = pd.concat([bitcoin_data,etherium_data], axis = 1)
combined_crypto.columns = ["Bitcoin", "Etherium"]
combined_crypto.index = combined_crypto.index.strftime("%Y-%m-%d")
combined_crypto.head()
```

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

Out[18]:

Date	Bitcoin	Etherium
2019-01-01	4540.254883	140.576462
2019-02-01	5074.797852	180.025482
2019-03-01	5480.324707	188.907867
2019-04-01	7165.559570	217.168701
2019-05-01	11601.275391	362.756836

In [19]:
```
invest_per_cryp = investment / len(combined_crypto.columns)
price_ratio = combined_crypto / combined_crypto.iloc[0]
crypto_port = price_ratio * invest_per_cryp
crypto_port["Total Value"] = crypto_port.sum(axis = 1)
crypto_port.head()
```

Out[19]:

Date	Bitcoin	Etherium	Total Value
2019-01-01	5000.000000	5000.000000	10000.000000
2019-02-01	5588.670661	6403.116136	11991.786798
2019-03-01	6035.261068	6719.043324	12754.304393
2019-04-01	7891.142409	7724.219916	15615.362324
2019-05-01	12776.017746	12902.474259	25678.492005

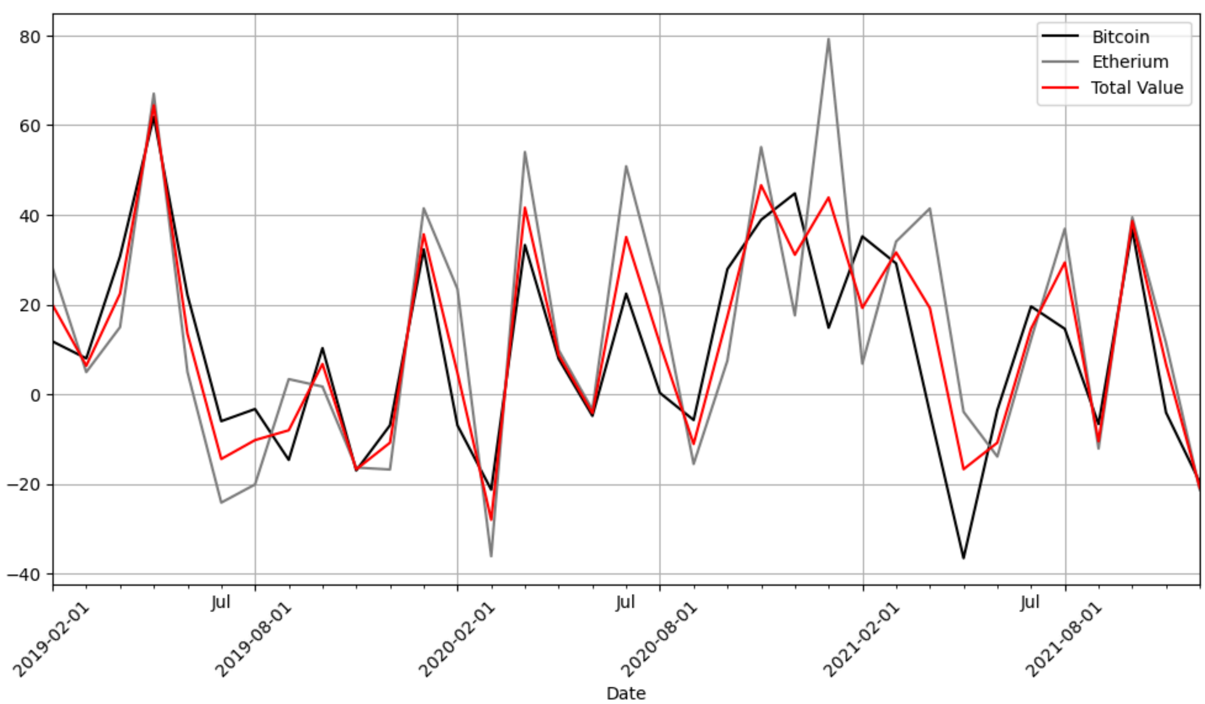
In [20]:
```
crypto_returns = crypto_port.pct_change()
crypto_returns = crypto_returns.drop(index = crypto_returns.index[0])
crypto_returns *=100
crypto_returns.index = pd.to_datetime(crypto_returns.index)
crypto_returns.index = crypto_returns.index.strftime("%Y-%m-%d")
crypto_returns.head()
```

Out[20]:

Date	Bitcoin	Etherium	Total Value
2019-02-01	11.773413	28.062323	19.917868
2019-03-01	7.990995	4.933960	6.358665
2019-04-01	30.750639	14.960115	22.432097
2019-05-01	61.903272	67.039188	64.443780
2019-06-01	22.062937	4.905906	13.442175

In [21]:

```
plt.figure(figsize= (12,6))
crypto_returns.index = pd.to_datetime(crypto_returns.index)
for column in crypto_returns.columns:
    if column == "Bitcoin":
        crypto_returns[column].plot(ax = plt.gca(), label = column, color =
    elif column == "Etherium":
        crypto_returns[column].plot(ax = plt.gca(), label = column, color =
    else:
        crypto_returns[column].plot(ax = plt.gca(), label = column, color =

plt.xticks(ticks=crypto_returns.index[::6],
           labels=crypto_returns.index[::6].strftime("%Y-%m-%d"), rotation=4
plt.legend()
plt.grid()
```



In [22]:

```
%%latex
\newpage
```

\newpage

In [23]:
```
# Combining the stocks with the crypto currencies
stock_data = pd.concat([data, industrails_data, healthcare_data, energy_data
crypto_data = combined_crypto
stock_crypto = pd.concat([stock_data, crypto_data], axis = 1)
stock_crypto.head()
```

Out[23]:

	BB.TO	CSU.TO	CTS.TO	GIB-A.TO	KXS.TO	AC.TO	CAE.TO	
Date								
2019-01-01	10.58	980.619995	0.60	86.870003	78.129997	29.670000	27.920000	38
2019-02-01	11.44	1123.729980	0.56	88.230003	82.730003	33.110001	27.750000	40
2019-03-01	13.47	1132.500000	0.85	91.870003	77.970001	32.209999	29.610001	3
2019-04-01	12.29	1182.069946	1.12	96.430000	73.209999	32.160000	31.160000	44
2019-05-01	10.61	1170.359985	0.90	98.519997	78.309998	39.900002	34.470001	4

5 rows × 27 columns

In [24]:
```
ratio = stock_crypto / stock_crypto.iloc[0]
total_portfolio_value = 10000
number_of_stock = 25
number_of_crypto = 2
stock_weight = total_portfolio_value / number_of_stock / 2
crypto_weight = total_portfolio_value / number_of_crypto / 2

stock_crypto_port = ratio.copy()
for i, col in enumerate(stock_crypto_port.columns):
    if i < len(stock_crypto_port.columns) - 2:
        stock_crypto_port[col] = ratio[col] * stock_weight
    else:
        stock_crypto_port[col]= ratio[col] * crypto_weight

stock_crypto_port["Total Mixed Portfolio Value"] = stock_crypto_port.sum(axi
stock_crypto_port.head()
```

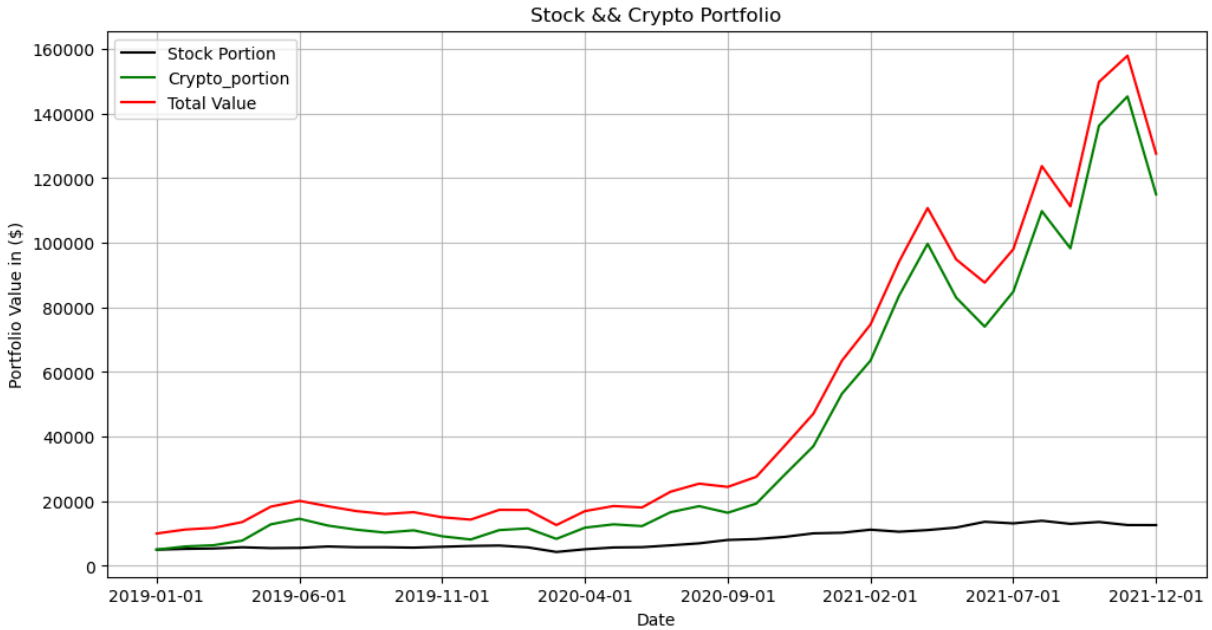
Out[24]:

	BB.TO	CSU.TO	CTS.TO	GIB-A.TO	KXS.TO	AC.TO	
Date							
2019-01-01	200.000000	200.000000	200.00000	200.000000	200.000000	200.000000	20
2019-02-01	216.257082	229.187654	186.66666	203.131117	211.775262	223.188409	19
2019-03-01	254.631387	230.976322	283.33333	211.511454	199.590436	217.121665	2'
2019-04-01	232.325143	241.086242	373.33332	222.009894	187.405610	216.784629	22
2019-05-01	200.567103	238.697965	299.99998	226.821673	200.460771	268.958554	24

5 rows × 28 columns

In [25]:

```
stock_portfolio = stock_crypto_port.iloc[: , :-3]
stock_portfolio["Total Value"] = stock_portfolio.sum(axis = 1)

crypto_portfolio = stock_crypto_port.iloc[: , -3:-1]
crypto_portfolio["Total Value"] = crypto_portfolio.sum(axis = 1)

mixed_port = pd.concat([stock_portfolio["Total Value"], crypto_portfolio["To
mixed_port.columns = ["Stock Portion", "Crypto_portion"]

mixed_port["Total Value"] = mixed_port.sum(axis = 1)

plt.figure(figsize = (12,6))
for col in mixed_port:
    if col == "Stock Portion":
        mixed_port[col].plot(ax = plt.gca(), label = col, color = "black")
    elif col == "Crypto_portion":
        mixed_port[col].plot(ax = plt.gca(), label = col, color = "green")
    else:
        mixed_port[col].plot(ax = plt.gca(), label = col, color = "red")

plt.legend()

plt.title("Stock && Crypto Portfolio")
plt.ylabel("Portfolio Value in ($)")
plt.grid(alpha = 0.8)
```



In [26]:
```
mixed_port_return = mixed_port.pct_change()
mixed_port_return = mixed_port_return.drop(index = mixed_port_return.index[0
mixed_port_return *=100

print("Average Return in %")
print(mixed_port_return.mean())

print()

print("Volatility")
print(mixed_port_return.std())
```

```
Average Return in %
Stock Portion      3.023132
Crypto_portion    11.603977
Total Value        8.887711
dtype: float64

Volatility
Stock Portion      8.303849
Crypto_portion    22.638405
Total Value       17.129152
dtype: float64
```

In [27]:
```
%%latex
\newpage
```

\newpage

In [28]:
```
stock_crypto.head()
```

Out[28]:

Date	BB.TO	CSU.TO	CTS.TO	GIB-A.TO	KXS.TO	AC.TO	CAE.TO	
2019-01-01	10.58	980.619995	0.60	86.870003	78.129997	29.670000	27.920000	38
2019-02-01	11.44	1123.729980	0.56	88.230003	82.730003	33.110001	27.750000	40
2019-03-01	13.47	1132.500000	0.85	91.870003	77.970001	32.209999	29.610001	3
2019-04-01	12.29	1182.069946	1.12	96.430000	73.209999	32.160000	31.160000	44
2019-05-01	10.61	1170.359985	0.90	98.519997	78.309998	39.900002	34.470001	4

5 rows × 27 columns

In [29]:
```
# Returns: in %
returns = stock_crypto.pct_change().dropna()
returns *= 100

# Dummy Variables:
curr_max = 0
optimal_weights = (0,0)
ratios = []
risk_free_rate = 0 # Rf can be adjusted
investment_amount = 10000

stock_returns = returns.iloc[:, :-2].mean(axis = 1)
crypto_returns = returns.iloc[:, -2:].mean(axis = 1)

# To find the weight
for i in range(0,101):
    # initial weights
    stock_weights = i / 100
    crypto_weights = 1 - stock_weights
    # return
    total_portfolio_return  = (stock_returns * stock_weights) +  (crypto_ret
    combined_returns = pd.concat([stock_returns, crypto_returns], axis=1).re
    weights = np.array([stock_weights, crypto_weights])
    # calculating the standard deviation
    covariance_matrix = combined_returns.cov().values
    portfolio_std = np.sqrt(np.dot(weights.T, np.dot(covariance_matrix, weig

    # Sharpe ratio
    sharpe_ratio = (total_portfolio_return.mean() - risk_free_rate) / portfo
    ratios.append(sharpe_ratio)
    if sharpe_ratio > curr_max:
        curr_max = sharpe_ratio
        optimal_weights = (stock_weights, crypto_weights)
```

```
print("Optimal Weights: Stock = {} %,Crypto = {} %".format(np.round(optimal_
print("Maximum Sharpe Ratio: {}".format(np.round(curr_max, 3)))
```

```
Optimal Weights: Stock = 27.0 %,Crypto = 73.0 %
Maximum Sharpe Ratio: 0.516
```