LIBRARY

In [ ]:

```
import pickle
import pandas as pd
import numpy as np
import pyfolio as pf
from multi_factor_util import get_performance_metrics, get_data_from_dict
import matplotlib.pyplot as plt

plt.style.use('seaborn-darkgrid')
```

```
WARNING (theano.tensor.blas): Using NumPy C-API based implementation for BLAS functions.
```

DATA

In [ ]:

```
data_filename = 'multifactor_data_2017_2022.bz2'
with open(data_filename, 'rb') as file:
    data = pickle.load(file)
```

In [ ]:

```
close_prices = get_data_from_dict(data, "Close")
total_equity = get_data_from_dict(data, "Total Equity")
total_liabilities = get_data_from_dict(data, "Total Liabilities")
```

In [ ]:

```
close_prices.to_csv("close_prices.csv", index=False)
total_equity.to_csv("total_equity.csv", index=False)
total_liabilities.to_csv("total_liabilities.csv", index=False)
```

FINANCIAL INDICATORS

1. DEBT TO EQUITY RATIO

In [ ]:

```
de_ratio = total_liabilities / total_equity
de_ratio.head()
```

Out[ ]:

	AAPL	ABBV	ACN	ADBE	AIG	AMGN	AMT	AMZN	ELV	AON	...	UNP	UPS	USB	RTX	V	VRTX	VZ	WBA	WFC	XOM
2017-10-02	1.799906	9.294601	1.336707	0.70209	5.890278	1.492507	2.914455	3.674629	1.632539	3.940728	...	1.997076	25.871995	8.305323	1.999751	1.075	0.773651	7.951601	1.334618	8.355486	0.846885
2017-10-03	1.799906	9.294601	1.336707	0.70209	5.890278	1.492507	2.914455	3.674629	1.632539	3.940728	...	1.997076	25.871995	8.305323	1.999751	1.075	0.773651	7.951601	1.334618	8.355486	0.846885
2017-10-04	1.799906	9.294601	1.336707	0.70209	5.890278	1.492507	2.914455	3.674629	1.632539	3.940728	...	1.997076	25.871995	8.305323	1.999751	1.075	0.773651	7.951601	1.334618	8.355486	0.846885
2017-10-05	1.799906	9.294601	1.336707	0.70209	5.890278	1.492507	2.914455	3.674629	1.632539	3.940728	...	1.997076	25.871995	8.305323	1.999751	1.075	0.773651	7.951601	1.334618	8.355486	0.846885
2017-10-06	1.799906	9.294601	1.336707	0.70209	5.890278	1.492507	2.914455	3.674629	1.632539	3.940728	...	1.997076	25.871995	8.305323	1.999751	1.075	0.773651	7.951601	1.334618	8.355486	0.846885

5 rows × 100 columns

2. RETURN ON EQUITY (ROE)

In [ ]:

```
net_income = get_data_from_dict(data, "Net Income")
roe = net_income / total_equity
roe.head()
```

Out[ ]:

	AAPL	ABBV	ACN	ADBE	AIG	AMGN	AMT	AMZN	ELV	AON	...	UNP	UPS	USB	RTX	V	VRTX	VZ	WBA	WFC	XOM
2017-10-02	0.079927	0.243906	0.096028	0.051352	-0.023818	0.062707	0.038431	0.010341	0.028788	0.035258	...	0.062347	0.818064	0.031671	0.041407	0.065324	-0.057093	0.127236	0.028365	0.021961	0.020983
2017-10-03	0.079927	0.243906	0.096028	0.051352	-0.023818	0.062707	0.038431	0.010341	0.028788	0.035258	...	0.062347	0.818064	0.031671	0.041407	0.065324	-0.057093	0.127236	0.028365	0.021961	0.020983
2017-10-04	0.079927	0.243906	0.096028	0.051352	-0.023818	0.062707	0.038431	0.010341	0.028788	0.035258	...	0.062347	0.818064	0.031671	0.041407	0.065324	-0.057093	0.127236	0.028365	0.021961	0.020983
2017-10-05	0.079927	0.243906	0.096028	0.051352	-0.023818	0.062707	0.038431	0.010341	0.028788	0.035258	...	0.062347	0.818064	0.031671	0.041407	0.065324	-0.057093	0.127236	0.028365	0.021961	0.020983
2017-10-06	0.079927	0.243906	0.096028	0.051352	-0.023818	0.062707	0.038431	0.010341	0.028788	0.035258	...	0.062347	0.818064	0.031671	0.041407	0.065324	-0.057093	0.127236	0.028365	0.021961	0.020983

5 rows × 100 columns

PORTFOLIO CONSTRUCTION & REBALANCING

In [ ]:

```
rebalancing_schedule = pd.DataFrame(index=roe.index)
rebalancing_schedule['is_start_of_month'] = rebalancing_schedule.index.to_series().dt.month != rebalancing_schedule.index.to_series().shift(1).dt.month
start_of_month_roe = roe[roe.index.isin(rebalancing_schedule[rebalancing_schedule['is_start_of_month']].index)]
start_of_month_roe.head()
```

Out[ ]:

	AAPL	ABBV	ACN	ADBE	AIG	AMGN	AMT	AMZN	ELV	AON	...	UNP	UPS	USB	RTX	V	VRTX	VZ	WBA	WFC	XOM
2017-10-02	0.079927	0.243906	0.096028	0.051352	-0.023818	0.062707	0.038431	0.010341	0.028788	0.035258	...	0.062347	0.818064	0.031671	0.041407	0.065324	-0.057093	0.127236	0.028365	0.021961	0.020983
2017-11-01	0.079927	0.243906	0.096028	0.051352	-0.023818	0.062707	0.038431	0.010341	0.028788	0.035258	...	0.062347	0.818064	0.031671	0.041407	0.065324	-0.057093	0.127236	0.028365	0.021961	0.020983
2017-12-01	0.079927	0.243906	0.113576	0.059286	-0.023818	0.062707	0.038431	0.010341	0.028788	0.035258	...	0.062347	0.818064	0.031671	0.041407	0.065324	-0.057093	0.127236	0.030685	0.021961	0.020983
2018-01-02	0.143118	0.010202	0.113576	0.059286	-0.101358	-0.168932	0.029984	0.067018	0.046448	-0.004088	...	0.292807	1.070312	0.033866	0.012582	0.075507	0.049299	0.417773	0.030685	0.029561	0.043085
2018-02-01	0.143118	0.010202	0.113576	0.059286	-0.101358	-0.168932	0.029984	0.067018	0.046448	-0.004088	...	0.292807	1.070312	0.033866	0.012582	0.075507	0.049299	0.417773	0.030685	0.029561	0.043085

5 rows × 100 columns

RANKING STOCKS BASED ON ROE

In [ ]:

```
roe_ranks = start_of_month_roe.rank(ascending=False, axis=1)
roe_ranks.head()
```

Out[ ]:

	AAPL	ABBV	ACN	ADBE	AIG	AMGN	AMT	AMZN	ELV	AON	...	UNP	UPS	USB	RTX	V	VRTX	VZ	WBA	WFC	XOM
2017-10-02	27.0	7.0	23.0	47.0	97.0	40.0	61.0	89.0	74.0	69.0	...	41.0	4.0	71.0	57.0	33.0	98.0	15.0	75.0	83.0	85.0
2017-11-01	27.0	7.0	23.0	47.0	97.0	40.0	61.0	89.0	74.0	69.0	...	41.0	4.0	71.0	57.0	33.0	98.0	15.0	75.0	83.0	85.0
2017-12-01	26.0	7.0	21.0	42.0	97.0	38.0	62.0	89.0	74.0	68.0	...	39.0	4.0	70.0	57.0	31.0	98.0	15.0	73.0	83.0	85.0
2018-01-02	16.0	68.0	21.0	35.0	90.0	96.0	56.0	32.0	41.0	71.0	...	8.0	3.0	50.0	67.0	25.0	40.0	6.0	55.0	58.0	45.0
2018-02-01	16.0	68.0	21.0	35.0	90.0	96.0	56.0	32.0	41.0	71.0	...	8.0	3.0	50.0	67.0	25.0	40.0	6.0	55.0	58.0	45.0

5 rows × 100 columns

RANKING STOCKS BASED ON DE RATIO

In [ ]:

```
start_of_month_de = de_ratio[de_ratio.index.isin(rebalancing_schedule[rebalancing_schedule['is_start_of_month']].index)]
de_ratio_ranks = start_of_month_de.rank(ascending=True, axis=1)
de_ratio_ranks.head()
```

Out[ ]:

	AAPL	ABBV	ACN	ADBE	AIG	AMGN	AMT	AMZN	ELV	AON	...	UNP	UPS	USB	RTX	V	VRTX	VZ	WBA	WFC	XOM
2017-10-02	47.0	93.0	31.0	9.0	83.0	37.0	67.0	73.0	41.0	75.0	...	53.0	97.0	91.0	54.0	22.0	10.0	90.0	30.0	92.0	17.0
2017-11-01	47.0	93.0	31.0	9.0	83.0	37.0	67.0	73.0	41.0	75.0	...	53.0	97.0	91.0	54.0	22.0	10.0	90.0	30.0	92.0	17.0
2017-12-01	47.0	93.0	30.0	10.0	83.0	36.0	67.0	73.0	41.0	75.0	...	53.0	97.0	91.0	54.0	22.0	11.0	90.0	35.0	92.0	17.0
2018-01-02	46.0	96.0	28.0	11.0	89.0	53.0	68.0	73.0	40.0	78.0	...	29.0	98.0	91.0	50.0	20.0	12.0	79.0	34.0	92.0	15.0
2018-02-01	46.0	96.0	28.0	11.0	89.0	53.0	68.0	73.0	40.0	78.0	...	29.0	98.0	91.0	50.0	20.0	12.0	79.0	34.0	92.0	15.0

5 rows × 100 columns

CALCULATING AND RANKING GROWTH RATE

In [ ]:

```
start_of_month_net_income = net_income[net_income.index.isin(rebalancing_schedule[rebalancing_schedule['is_start_of_month']].index)]
net_income_growth = start_of_month_net_income.pct_change(3)
growth_rate_ranks = net_income_growth.rank(ascending=False, axis=1)
growth_rate_ranks.head()
```

Out[ ]:

	AAPL	ABBV	ACN	ADBE	AIG	AMGN	AMT	AMZN	ELV	AON	...	UNP	UPS	USB	RTX	V	VRTX	VZ	WBA	WFC	XOM
2017-10-02	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2017-11-01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2017-12-01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2018-01-02	22.0	70.0	34.0	35.0	13.0	91.0	52.0	7.0	25.0	72.0	...	9.0	49.0	41.0	64.0	36.0	86.0	11.0	43.0	32.0	20.0
2018-02-01	22.0	70.0	34.0	35.0	13.0	91.0	52.0	7.0	25.0	72.0	...	9.0	49.0	41.0	64.0	36.0	86.0	11.0	43.0	32.0	20.0

5 rows × 100 columns

IMPLEMENTING STRATEGY

In [ ]:

```
roe_ranks = roe_ranks.drop([roe_ranks.index[0], roe_ranks.index[1], roe_ranks.index[2]])
de_ratio_ranks = de_ratio_ranks.drop([de_ratio_ranks.index[0], de_ratio_ranks.index[1], de_ratio_ranks.index[2]])
growth_rate_ranks = growth_rate_ranks.dropna()

total_ranks = roe_ranks + de_ratio_ranks + growth_rate_ranks
total_ranks.head()

total_ranks = total_ranks.rank(ascending=True, axis=1)
```

```
top_ranks = total_ranks[total_ranks <= 10]
top_ranks.head()
```

Out[ ]:

	AAPL	ABBV	ACN	ADBE	AIG	AMGN	AMT	AMZN	ELV	AON	...	UNP	UPS	USB	RTX	V	VRTX	VZ	WBA	WFC	XOM
2018-01-02	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	3.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2018-02-01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	3.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2018-03-01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	3.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2018-04-02	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	3.0	NaN	NaN	NaN	NaN
2018-05-01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	3.0	NaN	NaN	NaN	NaN

5 rows × 100 columns

SIGNAL GENERATION

In [ ]:

```
rebalancing_schedule = pd.DataFrame(index=close_prices.index)
rebalancing_schedule['is_start_of_month'] = rebalancing_schedule.index.to_series().dt.month != rebalancing_schedule.index.to_series().shift(1).dt.month

monthly_signals = top_ranks.applymap(lambda x: 1 if x <= 10 else x)
monthly_signals.head()

monthly_signals.fillna(0, inplace=True)
daily_signals = monthly_signals.reindex(rebalancing_schedule.index)
daily_signals = daily_signals.ffill()
daily_signals.tail()
```

Out[ ]:

	AAPL	ABBV	ACN	ADBE	AIG	AMGN	AMT	AMZN	ELV	AON	...	UNP	UPS	USB	RTX	V	VRTX	VZ	WBA	WFC	XOM
2022-06-24	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2022-06-27	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2022-06-28	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2022-06-29	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2022-06-30	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 100 columns

CALCULATION OF RETURNS

In [ ]:

```
daily_returns = close_prices.pct_change(axis=0)
strategy_returns = daily_signals.shift(1) * daily_returns
strategy_returns.dropna(inplace=True)
strategy_returns['Mean_Returns'] = strategy_returns.apply(lambda row: row[row != 0].mean(), axis=1)
strategy_returns.head()
strategy_returns.tail()
```

Out[ ]:

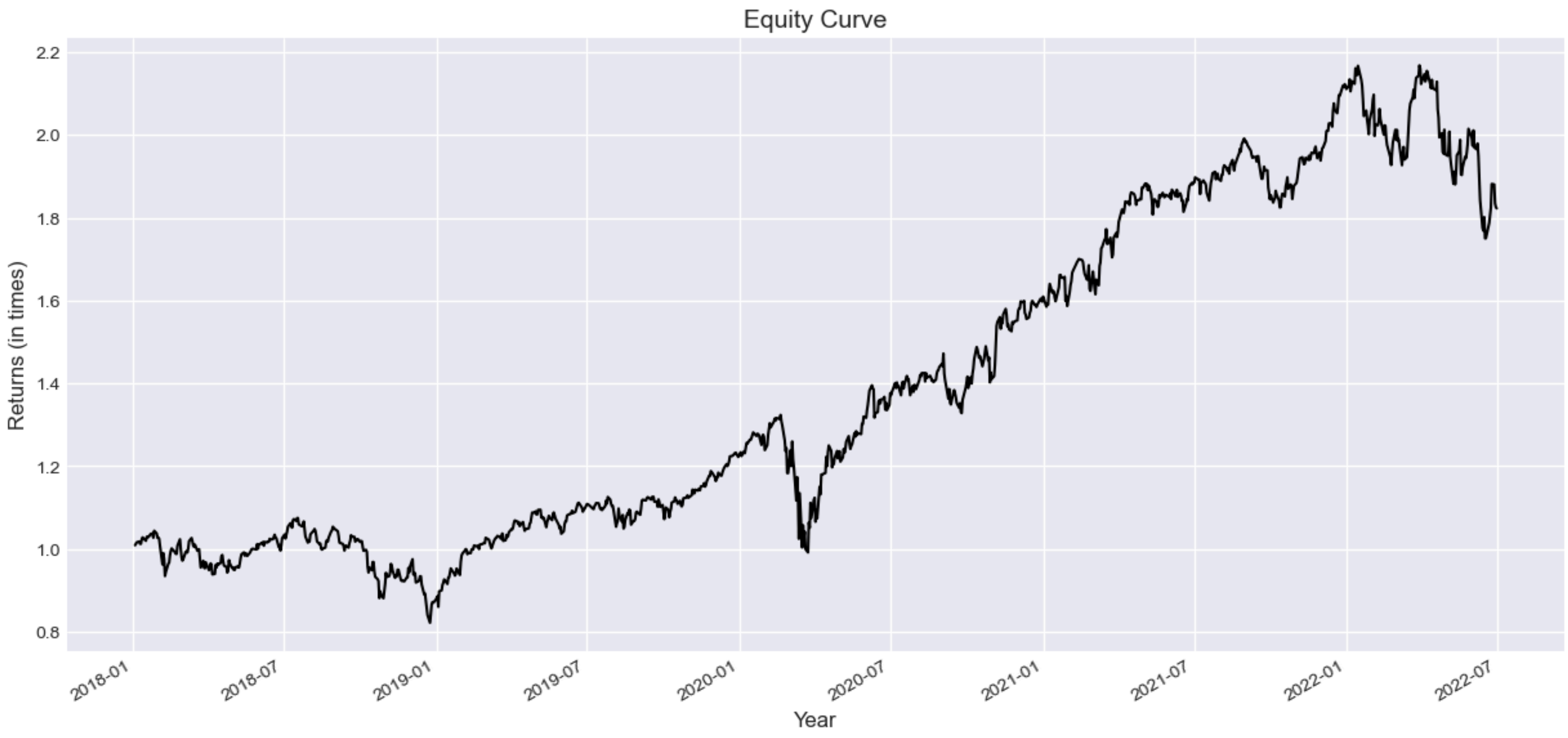
	AAPL	ABBV	ACN	ADBE	AIG	AMGN	AMT	AMZN	ELV	AON	...	UPS	USB	RTX	V	VRTX	VZ	WBA	WFC	XOM	Mean_Returns
2022-06-24	0.0	0.0	0.047371	0.0	0.0	0.0	0.0	0.0	-0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	-0.0	0.0	0.0	0.0	0.031461
2022-06-27	0.0	0.0	-0.022547	-0.0	-0.0	-0.0	-0.0	-0.0	0.0	-0.0	...	-0.0	-0.0	0.0	-0.0	-0.0	0.0	0.0	-0.0	0.0	-0.000663
2022-06-28	-0.0	-0.0	-0.030142	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	...	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	0.0	-0.024272
2022-06-29	0.0	0.0	-0.014059	0.0	-0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	-0.0	-0.0	0.0	0.0	0.0	-0.0	-0.0	-0.0	-0.004862
2022-06-30	-0.0	-0.0	-0.007720	-0.0	0.0	-0.0	0.0	-0.0	-0.0	0.0	...	0.0	-0.0	0.0	-0.0	0.0	-0.0	-0.0	-0.0	-0.0	-0.001758

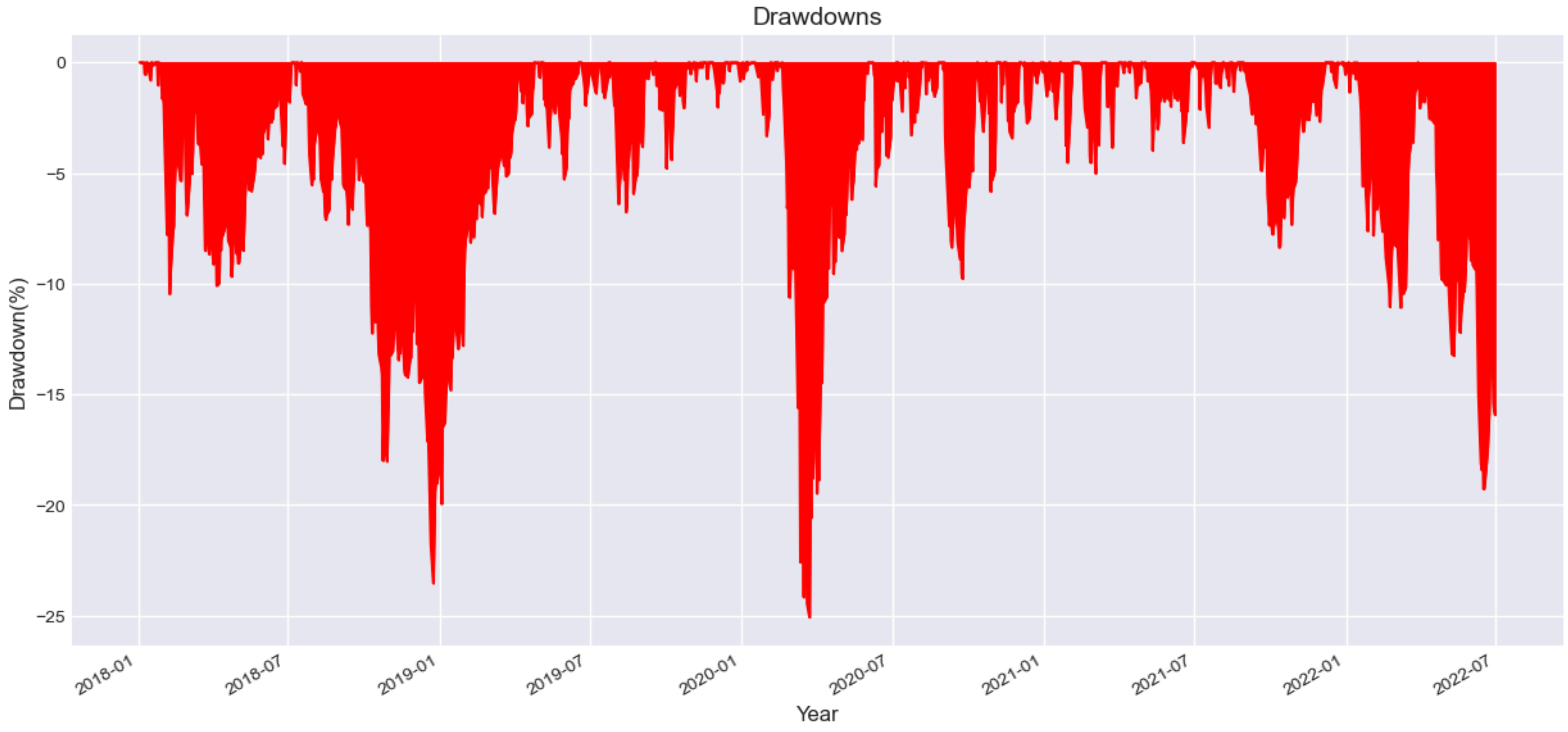
5 rows × 101 columns

PERFORMANCE CALCULATION

In [ ]:

```
get_performance_metrics(strategy_returns)
```





```
                  Strategy
CAGR                14.31%
Sharpe Ratio           0.7
Maximum Drawdown   -25.07%
```

Out[ ]:

	Strategy
CAGR	14.31%
Sharpe Ratio	0.7
Maximum Drawdown	-25.07%