

✓ Install Module

```
!pip install yfinance pandas openpyxl
```

```
Requirement already satisfied: yfinance in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: pandas in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: openpyxl in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: numpy>=1.16.5 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: requests>=2.31 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: multitasking>=0.0.7 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: platformdirs>=2.0.0 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: pytz>=2022.5 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: frozendict>=2.3.4 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: peewee>=3.16.2 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: beautifulsoup4>=4.11.1 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: curl_cffi>=0.7 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: protobuf>=3.19.0 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: websockets>=13.0 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: et-xmlfile in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: typing-extensions>=4.0.0 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: cffi>=1.12.0 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: certifi>=2024.2.2 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: pycparser in /usr/local/lib/python3.12/dist-packages
```

✓ Pengambilan Data

✓ List Saham

```
import pandas as pd
daftar_sahamperiode = pd.read_excel('Daftar Saham IDXESGL.xlsx', sheet_name='Daftar Saham')
ticker_sahamperiode1 = daftar_sahamperiode['Ticker P1'].dropna().tolist()
ticker_sahamperiode2 = daftar_sahamperiode['Ticker P2'].dropna().tolist()
print("Jumlah saham Periode 1", len(ticker_sahamperiode1))
print(ticker_sahamperiode1)
print("Jumlah saham Periode 2", len(ticker_sahamperiode2))
print(ticker_sahamperiode2)
```

```
Jumlah saham Periode 1 27
['ACES.JK', 'AKRA.JK', 'AUTO.JK', 'BBCA.JK', 'BBNI.JK', 'BBRI.JK', 'BMR1.JK', 'BREN.JK', 'BTHA.JK', 'BTHG.JK', 'BTHI.JK', 'BTHL.JK', 'BTHM.JK', 'BTHN.JK', 'BTHO.JK', 'BTHP.JK', 'BTHQ.JK', 'BTHR.JK', 'BTHS.JK', 'BTHT.JK', 'BTHU.JK', 'BTHV.JK', 'BTHW.JK', 'BTHX.JK', 'BTHY.JK', 'BTHZ.JK', 'BTHA.JK', 'BTHG.JK', 'BTHI.JK', 'BTHL.JK', 'BTHM.JK', 'BTHN.JK', 'BTHO.JK', 'BTHP.JK', 'BTHQ.JK', 'BTHR.JK', 'BTHS.JK', 'BTHT.JK', 'BTHU.JK', 'BTHV.JK', 'BTHW.JK', 'BTHX.JK', 'BTHY.JK', 'BTHZ.JK']
Jumlah saham Periode 2 25
['ACES.JK', 'AKRA.JK', 'AVIA.JK', 'BBCA.JK', 'BBNI.JK', 'BBRI.JK', 'BMR1.JK', 'BREN.JK', 'BTHA.JK', 'BTHG.JK', 'BTHI.JK', 'BTHL.JK', 'BTHM.JK', 'BTHN.JK', 'BTHO.JK', 'BTHP.JK', 'BTHQ.JK', 'BTHR.JK', 'BTHS.JK', 'BTHT.JK', 'BTHU.JK', 'BTHV.JK', 'BTHW.JK', 'BTHX.JK', 'BTHY.JK', 'BTHZ.JK', 'BTHA.JK', 'BTHG.JK', 'BTHI.JK', 'BTHL.JK', 'BTHM.JK', 'BTHN.JK', 'BTHO.JK', 'BTHP.JK', 'BTHQ.JK', 'BTHR.JK', 'BTHS.JK', 'BTHT.JK', 'BTHU.JK', 'BTHV.JK', 'BTHW.JK', 'BTHX.JK', 'BTHY.JK', 'BTHZ.JK']
```

✓ Mengambil Data Saham

```
def Scraping_Saham(tickers, start_date, end_date):  
    import yfinance  
    data_saham = yfinance.download(tickers, start_date, end_date, auto_adjust=True)  
    data_saham = data_saham.dropna()  
    closing_prices = data_saham['Close']  
    return closing_prices
```

```
Ticker_market = ('^JKSE')  
data_marketperiode1 = Scraping_Saham(Ticker_market, '2024-08-23', '2025-02-21')  
data_sahamperiode1 = Scraping_Saham(ticker_sahamperiode1, '2024-08-23', '2025-02-21')  
data_periode1 = pd.concat([data_marketperiode1, data_sahamperiode1], axis=1)  
data_periode1
```

```
[*****100%*****] 1 of 1 completed  
[*****100%*****] 27 of 27 completed
```

Ticker	^JKSE	ACES.JK	AKRA.JK	AUTO.JK	BBCA.JK
Date					
2024-08-23	7544.297852	677.318909	1340.802368	1996.018433	9901.624023
2024-08-26	7606.194824	672.647766	1359.296143	2004.969238	9901.624023
2024-08-27	7597.880859	686.661255	1373.166504	1969.166260	9781.750000
2024-08-28	7658.875000	681.990112	1391.660278	1960.215576	9925.598633
2024-08-29	7627.604004	681.990112	1368.542969	1969.166260	9805.725586
...
2025-02-17	6830.881836	747.386414	1031.030762	1862.769653	8987.230469
2025-02-18	6873.554199	738.044067	1109.629517	1871.945923	8963.136719
2025-02-19	6794.868164	738.044067	1114.252930	1830.653076	8625.814453
2025-02-20	6788.041992	728.701721	1141.993774	1830.653076	8674.003906
2025-02-21	6803.000977	714.688232	1137.370239	1862.769653	8674.003906

122 rows × 28 columns

```
data_marketperiode2 = Scraping_Saham(Ticker_market, '2025-02-24', '2025-02-24')  
data_sahamperiode2 = Scraping_Saham(ticker_sahamperiode2, '2025-02-24', '2025-02-24')  
data_periode2 = pd.concat([data_marketperiode2, data_sahamperiode2], axis=1)  
data_periode2
```

```
[*****100%*****] 1 of 1 completed  
[*****100%*****] 25 of 25 completed
```

Ticker	^JKSE	ACES.JK	AKRA.JK	AVIA.JK	BBCA.JK
--------	-------	---------	---------	---------	---------

Date						
2025-02-24	6749.601074	691.332397	1123.499756	345.610260	8601.719727	3
2025-02-25	6587.086914	653.963135	1109.629517	324.721741	8505.341797	3
2025-02-26	6606.178223	649.291931	1114.252930	328.519653	8457.152344	3
2025-02-27	6485.448242	653.963135	1211.345459	336.115509	8216.208008	3
2025-02-28	6270.597168	602.580261	1146.617065	345.610260	8119.831055	3
...
2025-08-15	7898.375000	472.000000	1260.000000	413.904755	8642.865234	4
2025-08-19	7862.949219	458.000000	1220.000000	413.904755	8444.178711	4
2025-08-20	7943.825195	456.000000	1230.000000	411.952362	8469.014648	4
2025-08-21	7890.714844	460.000000	1225.000000	413.904755	8493.850586	4
2025-08-22	7858.851074	456.000000	1230.000000	415.857117	8394.507812	4

113 rows × 26 columns

```
with pd.ExcelWriter('Data Saham Periode 1 dan 2.xlsx') as writer:
    data_periode1.to_excel(writer, sheet_name='Periode 1')
    data_periode2.to_excel(writer, sheet_name='Periode 2')
print("Data saved to 'Data Saham Periode 1 dan 2.xlsx'")
```

Data saved to 'Data Saham Periode 1 dan 2.xlsx'

```
import matplotlib.pyplot as plt

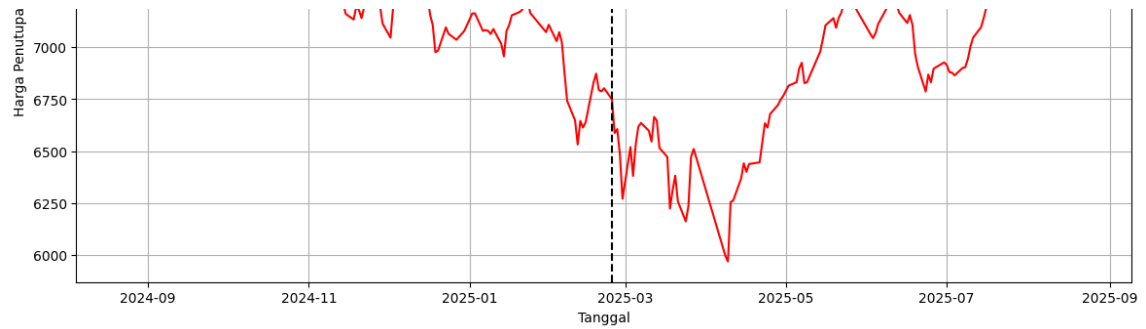
combined_data = pd.concat([data_periode1, data_periode2], axis=0)

plt.figure(figsize=(14, 6))
plt.plot(combined_data.index, combined_data['^JKSE'], color='red')
plt.axvline(pd.to_datetime('2025-02-24'), color='black', linestyle='--')

plt.title('Daily IHSG Price')
plt.xlabel('Tanggal')
plt.ylabel('Harga Penutupan')
plt.legend()
plt.grid(True)
plt.show()
```

/tmp/ipython-input-4029968018.py:12: UserWarning: No artists with labels
plt.legend()





✓ Menyimpan Bunga Bebas Risiko

```
sukubunga_periode1 = pd.read_excel('Suku Bunga BI.xlsx', sheet_name='A{
data_sukubunga_harian1 = sukubunga_periode1['Suku Bunga']/233
risk_freeperiode1 = data_sukubunga_harian1.mean()
risk_freeperiode1
```

```
np.float64(0.00025597792765174736)
```

```
sukubunga_periode2 = pd.read_excel('Suku Bunga BI.xlsx', sheet_name='F{
data_sukubunga_harian2 = sukubunga_periode2['Suku Bunga']/233
risk_freeperiode2 = data_sukubunga_harian2.mean()
risk_freeperiode2
```

```
np.float64(0.00023605150214592278)
```

✓ Return

✓ Return Saham

```
def returns(data):
    import numpy as np
    return_harian = np.log(data / data.shift(1))
    return return_harian
```

```
data_returnperiode1 = returns(data_periode1).dropna()
data_returnperiode1
```

Ticker	^JKSE	ACES.JK	AKRA.JK	AUTO.JK	BBCA.JK	BBNI.JK	
Date							
2024-08-26	0.008171	-0.006920	0.013699	0.004474	0.000000	0.013668	
2024-08-27	-0.001094	0.020619	0.010152	-0.018018	-0.012180	-0.022884	-
2024-08-28	0.007996	-0.006826	0.013378	-0.004556	0.014599	-0.004640	-
2024-08-29	-0.004091	0.000000	-0.016751	0.004556	-0.012151	-0.014052	-
2024-08-30	0.005638	-0.020762	0.010084	0.004535	0.009732	0.009390	
...	
2025-02-17	0.028574	-0.006231	0.004494	0.014889	0.038256	0.044750	
2025-02-18	0.006228	-0.012579	0.073467	0.004914	-0.002684	0.038631	
2025-02-19	-0.011514	0.000000	0.004158	-0.022306	-0.038361	-0.049633	-
2025-02-20	-0.001005	-0.012739	0.024591	0.000000	0.005571	-0.013363	-
2025-02-21	0.002201	-0.019418	-0.004057	0.017392	0.000000	-0.036534	-
121 rows × 28 columns							

```
data_returnperiode2 = returns(data_periode2).dropna()
data_returnperiode2
```

Ticker	^JKSE	ACES.JK	AKRA.JK	AVIA.JK	BBCA.JK	BBNI.JK	
Date							
2025-02-25	-0.024372	-0.055570	-0.012422	-0.062343	-0.011268	0.007117	-
2025-02-26	0.002894	-0.007169	0.004158	0.011628	-0.005682	0.027974	
2025-02-27	-0.018444	0.007169	0.083548	0.022858	-0.028904	-0.002302	-
2025-02-28	-0.033689	-0.081830	-0.054916	0.027857	-0.011799	-0.074108	-
2025-03-03	0.038950	0.038027	-0.004040	0.016349	0.043548	0.055503	
...	
2025-08-15	-0.004154	-0.020965	0.003976	0.000000	-0.008584	-0.004566	
2025-08-19	-0.004495	-0.030110	-0.032261	0.000000	-0.023257	-0.009195	-
2025-08-20	0.010233	-0.004376	0.008163	-0.004728	0.002937	0.022832	
2025-08-21	-0.006708	0.008734	-0.004073	0.004728	0.002928	0.013453	
2025-08-22	-0.004046	-0.008734	0.004073	0.004706	-0.011765	-0.022523	-

112 rows × 26 columns

✓ Statistik Saham

```
def statsaham(data, market_ticker):  
    expected_returns = data.mean()  
    st_dev = data.std()  
    variance = st_dev**2  
    cov_matrix = data.cov()  
    covariance = cov_matrix[market_ticker]  
    stats_df = pd.DataFrame({'Expected Return': expected_returns, 'St.  
    return stats_df
```

```
stat_periode1 = statsaham(data_returnperiode1, Ticker_market)  
stat_periode1
```

	Expected Return	St. Deviation	Variance	Covariance
Ticker				
^JKSE	-0.000855	0.009354	0.000087	0.000087
ACES.JK	0.000444	0.023191	0.000538	0.000021
AKRA.JK	-0.001360	0.023985	0.000575	0.000052
AUTO.JK	-0.000571	0.018273	0.000334	0.000055
BBCA.JK	-0.001094	0.015549	0.000242	0.000079
BBNI.JK	-0.001959	0.022602	0.000511	0.000141
BBRI.JK	-0.002049	0.019612	0.000385	0.000110
BMRI.JK	-0.002717	0.021565	0.000465	0.000141
BMTR.JK	-0.002519	0.015981	0.000255	0.000062
BRPT.JK	-0.002243	0.030375	0.000923	0.000168
BSDE.JK	-0.002204	0.021448	0.000460	0.000089
CTRA.JK	-0.003399	0.025731	0.000662	0.000090
EMTK.JK	0.003884	0.034236	0.001172	0.000074
ERAA.JK	-0.001115	0.029681	0.000881	0.000106
GOTO.JK	0.003403	0.034319	0.001178	0.000145
JSMR.JK	-0.002156	0.016302	0.000266	0.000059
MAPI.JK	-0.000422	0.031410	0.000987	0.000030
MIKA.JK	-0.001769	0.020449	0.000418	0.000010



MNCN.JK	-0.002020	0.016560	0.000274	0.000065
MPMX.JK	-0.000506	0.005837	0.000034	0.000021
PGEO.JK	-0.002067	0.022679	0.000514	0.000051
PWON.JK	-0.001632	0.020188	0.000408	0.000081
SCMA.JK	0.005250	0.032441	0.001052	0.000009
SIDO.JK	-0.001083	0.018306	0.000335	0.000041
TBIG.JK	0.001380	0.014022	0.000197	-0.000008
TLKM.JK	-0.000886	0.021779	0.000474	0.000101
TOWR.JK	-0.002478	0.022500	0.000506	0.000063
UNVR.JK	-0.004777	0.025031	0.000627	0.000075

Next steps: [New interactive sheet](#)

```
stat_periode2 = statsaham(data_returnperiode2, Ticker_market)
stat_periode2
```

	Expected Return	St. Deviation	Variance	Covariance
Ticker				
^JKSE	0.001359	0.015053	0.000227	0.000227
ACES.JK	-0.003715	0.034618	0.001198	0.000259
AKRA.JK	0.000809	0.035307	0.001247	0.000244
AVIA.JK	0.001652	0.019528	0.000381	0.000107
BBCA.JK	-0.000218	0.018051	0.000326	0.000219
BBNI.JK	0.001161	0.025324	0.000641	0.000265
BBRI.JK	0.000904	0.026543	0.000705	0.000311
BMRI.JK	0.000613	0.025854	0.000668	0.000291
BNGA.JK	0.000940	0.012983	0.000169	0.000145
BSDE.JK	0.000388	0.025601	0.000655	0.000239
CMRY.JK	0.000972	0.023652	0.000559	0.000102
CTRA.JK	0.001880	0.027962	0.000782	0.000233
EMTK.JK	0.005896	0.041010	0.001682	0.000294
ERAA.JK	0.002034	0.037237	0.001387	0.000273
GOTO.JK	-0.002309	0.034172	0.001168	0.000279
JSMR.JK	-0.001570	0.023697	0.000562	0.000186
MADI JK	-0.000718	0.033566	0.001127	0.000313

MARKET TICKER	PERIOD 1	PERIOD 2	PERIOD 3	PERIOD 4
MIKA.JK	-0.000150	0.026849	0.000721	0.000125
MNCN.JK	0.000070	0.025268	0.000638	0.000221
PGEO.JK	0.003941	0.038898	0.001513	0.000231
PWON.JK	0.000040	0.023271	0.000542	0.000232
SCMA.JK	0.003103	0.041503	0.001722	0.000283
SIDO.JK	-0.000661	0.018845	0.000355	0.000118
TLKM.JK	0.002642	0.021756	0.000473	0.000178
TOWR.JK	0.000529	0.032226	0.001038	0.000264
UNVR.JK	0.003272	0.039370	0.001550	0.000256

Next steps: [New interactive sheet](#)

✓ Pemilihan Saham Positif

```
def return_positif(data_return, market_ticker):
    saham_return_positif = data_return.loc[:, data_return.mean() > 0]
    if market_ticker not in saham_return_positif.columns:
        data_positif = pd.concat([data_return[market_ticker], saham_return_positif], axis=1)
    else:
        data_positif = saham_return_positif
    return data_positif
```

```
datapositif_periode1 = return_positif(data_returnperiode1, Ticker_market)
print('Saham Positif Periode 1 Sebanyak', len(datapositif_periode1.columns))
print(list(datapositif_periode1.columns))
```

Saham Positif Periode 1 Sebanyak 5 Saham
 ['^JKSE', 'ACES.JK', 'EMTK.JK', 'GOTO.JK', 'SCMA.JK', 'TBIG.JK']

```
datapositif_periode2 = return_positif(data_returnperiode2, Ticker_market)
print('Saham Positif Periode 2 Sebanyak', len(datapositif_periode2.columns))
print(list(datapositif_periode2.columns))
```

Saham Positif Periode 2 Sebanyak 18 Saham
 ['^JKSE', 'AKRA.JK', 'AVIA.JK', 'BBNI.JK', 'BBRI.JK', 'BMRI.JK', 'BNGA.JK', 'BRIS.JK', 'BSDE.JK', 'BSIM.JK', 'BSIN.JK', 'BSIS.JK', 'BSPT.JK', 'BSRI.JK', 'BSST.JK', 'BSUN.JK', 'BSUN.JK']

✓ Parameter Market

```
def parametermarket(stat_data, data_positif, market_ticker):
    stat_positif = stat_data.loc[data_positif.columns]
    market_variance = stat_data.loc[market_ticker, 'Variance']
```

```

expected_market_return = stat_data.loc[market_ticker, 'Expected Return']
beta_saham = stat_positif['Covariance'] / market_variance
alpha_saham = stat_positif['Expected Return'] - (beta_saham * expected_market_return)
residual_variance = stat_positif['Variance'] - (beta_saham**2 * market_variance)

parameter_df = pd.DataFrame({'Beta': beta_saham, 'Alpha': alpha_saham, 'Residual Variance': residual_variance})
return parameter_df.drop(market_ticker)

```

```

parameter_periode1 = parametermarket(stat_periode1, data_positif_periode1)
return parameter_periode1

```




	Beta	Alpha	Residual Variance	
ACES.JK	0.238838	0.000648	0.000533	
EMTK.JK	0.851114	0.004612	0.001109	
GOTO.JK	1.657812	0.004820	0.000937	
SCMA.JK	0.097899	0.005333	0.001052	
TBIG.JK	-0.089543	0.001304	0.000196	

Next steps: [New interactive sheet](#)

```

parameter_periode2 = parametermarket(stat_periode2, data_positif_periode2)
return parameter_periode2

```

	Beta	Alpha	Residual Variance	
Ticker				
AKRA.JK	1.077484	-0.000655	0.000984	
AVIA.JK	0.473799	0.001008	0.000330	
BBNI.JK	1.171596	-0.000431	0.000330	
BBRI.JK	1.370484	-0.000957	0.000279	
BMRI.JK	1.283775	-0.001131	0.000295	
BNGA.JK	0.638973	0.000071	0.000076	
BSDE.JK	1.055903	-0.001046	0.000403	
CMRY.JK	0.449065	0.000362	0.000514	
CTRA.JK	1.030045	0.000481	0.000541	
EMTK.JK	1.298122	0.004132	0.001300	
ERAA.JK	1.206402	0.000395	0.001057	
MNCN.JK	0.976161	-0.001256	0.000423	
PGEO.JK	1.020914	0.002554	0.001277	
PWON.JK	1.025887	0.001254	0.000302	

PWON.JK	1.025887	-0.001354	0.000303
SCMA.JK	1.247695	0.001408	0.001370
TLKM.JK	0.785580	0.001575	0.000333
TOWR.JK	1.162926	-0.001051	0.000732
UNVR.JK	1.131957	0.001734	0.001260

Next steps: [New interactive sheet](#)

✓ Single Index Model

✓ Parameter

```
def simparameter(data_positif, risk_free, parameter_market, market_ticker):
    ERB_saham = (data_positif.mean() - risk_free) / parameter_market['Beta']
    A_saham = ((data_positif.mean() - risk_free) * parameter_market['Beta']) / parameter_market['Residual Variance']
    B_saham = parameter_market['Beta']**2 / parameter_market['Residual Variance']
    Cutoff_saham = (data_positif[market_ticker].var() * A_saham) / (1 + B_saham)
    Max_cutoff = max(Cutoff_saham)
    proporsi_saham = ((ERB_saham - Max_cutoff) * (parameter_market['Beta'] / B_saham)) / (1 + B_saham)

    sim_df = pd.DataFrame({'ERB': ERB_saham, 'Nilai A': A_saham, 'Nilai B': B_saham, 'Cut Off': Cutoff_saham, 'Proporsi Saham': proporsi_saham})
    return sim_df.drop(market_ticker)
```

```
sim_periode1 = simparameter(datapositif_periode1, risk_freeperiode1, parameter_market, market_ticker)
sim_periode1
```

	ERB	Nilai A	Nilai B	Cut Off	Proporsi Saham
ACES.JK	0.000787	0.084206	107.056659	0.000007	0.178852
EMTK.JK	0.004263	2.785380	653.375418	0.000231	2.975124
GOTO.JK	0.001898	5.565701	2932.159329	0.000388	2.671812
SCMA.JK	0.051008	0.464877	9.113868	0.000041	4.712455
TBIG.JK	-0.012557	-0.513892	40.925829	-0.000045	5.916203

Next steps: [New interactive sheet](#)

```
sim_periode2 = simparameter(datapositif_periode2, risk_freeperiode2, parameter_market, market_ticker)
sim_periode2
```

	ERB	Nilai A	Nilai B	Cut Off	Proporsi Saham
--	-----	---------	---------	---------	----------------

Ticker						
AKRA.JK	0.000531	0.627277	1180.435026	0.000112	-0.502303	
AVIA.JK	0.002989	2.030194	679.309124	0.000399	2.865675	
BBNI.JK	0.000789	3.281070	4156.010279	0.000383	-0.710931	
BBRI.JK	0.000488	3.284201	6733.899786	0.000295	-2.467465	
BMRI.JK	0.000293	1.639438	5587.298330	0.000164	-3.031197	
BNGA.JK	0.001101	5.911712	5369.445251	0.000604	0.933608	
BSDE.JK	0.000144	0.399021	2768.098413	0.000056	-2.217146	
CMRY.JK	0.001640	0.643692	392.542970	0.000134	0.568107	
CTRA.JK	0.001596	3.128062	1959.600207	0.000491	1.153612	
EMTK.JK	0.004360	5.651685	1296.230304	0.000990	3.365291	
ERAA.JK	0.001490	2.052107	1377.157198	0.000354	0.571015	
MNCN.JK	-0.000170	-0.383524	2254.991849	-0.000058	-2.679598	
PGEO.JK	0.003629	2.962294	816.255469	0.000566	2.110158	
PWON.JK	-0.000191	-0.664677	3472.951290	-0.000084	-3.998996	
SCMA.JK	0.002298	2.611722	1136.539029	0.000471	1.191536	
TLKM.JK	0.003062	5.667247	1850.629684	0.000905	4.882161	
TOWR.JK	0.000252	0.465668	1847.436973	0.000074	-1.172122	
UNVR.JK	0.002682	2.728374	1017.172791	0.000502	1.520804	

Next steps: [New interactive sheet](#)

Pemilihan Kandidat

```
def kandidatSIM(parameter_sim):
    pemilihan_kandidatSIM = []
    for saham in parameter_sim['ERB'].index:
        if parameter_sim['ERB'][saham] > max(parameter_sim['Cut Off']):
            pemilihan_kandidatSIM.append(saham)
    return pemilihan_kandidatSIM
```

```
kandidatSIM_periode1 = kandidatSIM(sim_periode1)
print('Saham Kandidat SIM Periode 1 Sebanyak', len(kandidatSIM_periode1))
print(kandidatSIM_periode1)
```

```
Saham Kandidat SIM Periode 1 Sebanyak 4 Saham
['ACES.JK', 'EMTK.JK', 'GOTO.JK', 'SCMA.JK']
```

```
kandidatSIM_periode2 = kandidatSIM(sim_periode2)
```

```
print('Saham Kandidat SIM Periode 2 Sebanyak', len(kandidatSIM_periode1))
print(kandidatSIM_periode2)
```

Saham Kandidat SIM Periode 2 Sebanyak 10 Saham

```
['AVIA.JK', 'BNGA.JK', 'CMRY.JK', 'CTRA.JK', 'EMTK.JK', 'ERAA.JK', 'PGEO.JK', 'SCMA.JK', 'TLKM.JK', 'UNVR.JK']
```

✓ Pembentukan Kombinasi Saham

```
import itertools
def kombinasi(kandidat):
    kombinasi = []
    for i in range(2, len(kandidat) + 1):
        for combo in itertools.combinations(kandidat, i):
            kombinasi.append(combo)
    return kombinasi
```

```
kombinasiSIM_periode1 = kombinasi(kandidatSIM_periode1)
kombinasiSIM_periode1
```

```
[('ACES.JK', 'EMTK.JK'),
 ('ACES.JK', 'GOTO.JK'),
 ('ACES.JK', 'SCMA.JK'),
 ('EMTK.JK', 'GOTO.JK'),
 ('EMTK.JK', 'SCMA.JK'),
 ('GOTO.JK', 'SCMA.JK'),
 ('ACES.JK', 'EMTK.JK', 'GOTO.JK'),
 ('ACES.JK', 'EMTK.JK', 'SCMA.JK'),
 ('ACES.JK', 'GOTO.JK', 'SCMA.JK'),
 ('EMTK.JK', 'GOTO.JK', 'SCMA.JK'),
 ('ACES.JK', 'EMTK.JK', 'GOTO.JK', 'SCMA.JK')]
```

```
kombinasiSIM_periode2 = kombinasi(kandidatSIM_periode2)
kombinasiSIM_periode2
```

```
[('AVIA.JK', 'BNGA.JK'),
 ('AVIA.JK', 'CMRY.JK'),
 ('AVIA.JK', 'CTRA.JK'),
 ('AVIA.JK', 'EMTK.JK'),
 ('AVIA.JK', 'ERAA.JK'),
 ('AVIA.JK', 'PGEO.JK'),
 ('AVIA.JK', 'SCMA.JK'),
 ('AVIA.JK', 'TLKM.JK'),
 ('AVIA.JK', 'UNVR.JK'),
 ('BNGA.JK', 'CMRY.JK'),
 ('BNGA.JK', 'CTRA.JK'),
 ('BNGA.JK', 'EMTK.JK'),
 ('BNGA.JK', 'ERAA.JK'),
 ('BNGA.JK', 'PGEO.JK'),
 ('BNGA.JK', 'SCMA.JK'),
 ('BNGA.JK', 'TLKM.JK'),
 ('BNGA.JK', 'UNVR.JK'),
 ('CMRY.JK', 'CTRA.JK'),
 ('CMRY.JK', 'EMTK.JK'),
```

```
( 'CMRY.JK', 'ERAA.JK' ),
( 'CMRY.JK', 'PGEO.JK' ),
( 'CMRY.JK', 'SCMA.JK' ),
( 'CMRY.JK', 'TLKM.JK' ),
( 'CMRY.JK', 'UNVR.JK' ),
( 'CTRA.JK', 'EMTK.JK' ),
( 'CTRA.JK', 'ERAA.JK' ),
( 'CTRA.JK', 'PGEO.JK' ),
( 'CTRA.JK', 'SCMA.JK' ),
( 'CTRA.JK', 'TLKM.JK' ),
( 'CTRA.JK', 'UNVR.JK' ),
( 'EMTK.JK', 'ERAA.JK' ),
( 'EMTK.JK', 'PGEO.JK' ),
( 'EMTK.JK', 'SCMA.JK' ),
( 'EMTK.JK', 'TLKM.JK' ),
( 'EMTK.JK', 'UNVR.JK' ),
( 'ERAA.JK', 'PGEO.JK' ),
( 'ERAA.JK', 'SCMA.JK' ),
( 'ERAA.JK', 'TLKM.JK' ),
( 'ERAA.JK', 'UNVR.JK' ),
( 'PGEO.JK', 'SCMA.JK' ),
( 'PGEO.JK', 'TLKM.JK' ),
( 'PGEO.JK', 'UNVR.JK' ),
( 'SCMA.JK', 'TLKM.JK' ),
( 'SCMA.JK', 'UNVR.JK' ),
( 'TLKM.JK', 'UNVR.JK' ),
( 'AVIA.JK', 'BNGA.JK', 'CMRY.JK' ),
( 'AVIA.JK', 'BNGA.JK', 'CTRA.JK' ),
( 'AVIA.JK', 'BNGA.JK', 'EMTK.JK' ),
( 'AVIA.JK', 'BNGA.JK', 'ERAA.JK' ),
( 'AVIA.JK', 'BNGA.JK', 'PGEO.JK' ),
( 'AVIA.JK', 'BNGA.JK', 'SCMA.JK' ),
( 'AVIA.JK', 'BNGA.JK', 'TLKM.JK' ),
( 'AVIA.JK', 'BNGA.JK', 'UNVR.JK' ),
( 'AVIA.JK', 'CMRY.JK', 'CTRA.JK' ),
( 'AVIA.JK', 'CMRY.JK', 'EMTK.JK' ),
( 'AVIA.JK', 'CMRY.JK', 'ERAA.JK' ),
( 'AVIA.JK', 'CMRY.JK', 'PGEO.JK' ),
( 'AVIA.JK', 'CMRY.JK', 'SCMA.JK' ),
```

✓ Portofolio

✓ Bobot

```
def Bobot_SIM(kombinasi_saham, proporsi_saham):
    Nilai_Bobot = {}
    for kombinasi in kombinasi_saham:
        total_proportion = sum(proporsi_saham[saham] for saham in kombinasi)
        Bobot = {saham: proporsi_saham[saham] / total_proportion for saham in kombinasi}
        Nilai_Bobot[kombinasi] = Bobot
    return Nilai_Bobot
```

```
Robot1 PortofolioSTM = Bobot_SIM(kombinasiSTM periode1, sim periode1['Pr
```

```

Bobot1_PortofolioSIM_list = [{ 'Combination': k, 'Weights': {stock: float
display(pd.DataFrame(Bobot1_PortofolioSIM_list))

```

	Combination	Weights	
0	(ACES.JK, EMTK.JK)	{'ACES.JK': 0.05670679445735575, 'EMTK.JK': 0....	
1	(ACES.JK, GOTO.JK)	{'ACES.JK': 0.06274041478782218, 'GOTO.JK': 0....	
2	(ACES.JK, SCMA.JK)	{'ACES.JK': 0.036565246933359025, 'SCMA.JK': 0...	
3	(EMTK.JK, GOTO.JK)	{'EMTK.JK': 0.526856292763549, 'GOTO.JK': 0.47...	
4	(EMTK.JK, SCMA.JK)	{'EMTK.JK': 0.3870039986541208, 'SCMA.JK': 0.6...	
5	(GOTO.JK, SCMA.JK)	{'GOTO.JK': 0.36182495947870646, 'SCMA.JK': 0....	
6	(ACES.JK, EMTK.JK, GOTO.JK)	{'ACES.JK': 0.03070002745156521, 'EMTK.JK': 0....	

```

Bobot2_PortofolioSIM = Bobot_SIM(kombinasiSIM_periode2, sim_periode2['I
Bobot2_PortofolioSIM_list = [{ 'Combination': k, 'Weights': {stock: floa
display(pd.DataFrame(Bobot2_PortofolioSIM_list))

```

	Combination	Weights	
0	(AVIA.JK, BNGA.JK)	{'AVIA.JK': 0.7542672667054878, 'BNGA.JK': 0.2...	
1	(AVIA.JK, CMRY.JK)	{'AVIA.JK': 0.834553532560817, 'CMRY.JK': 0.16...	
2	(AVIA.JK, CTRA.JK)	{'AVIA.JK': 0.7129808850932551, 'CTRA.JK': 0.2...	
3	(AVIA.JK, EMTK.JK)	{'AVIA.JK': 0.4599086673662129, 'EMTK.JK': 0.5...	
4	(AVIA.JK, ERAA.JK)	{'AVIA.JK': 0.8338475295056604, 'ERAA.JK': 0.1...	
...	
1008	(AVIA.JK, BNGA.JK, CMRY.JK, EMTK.JK, ERAA.JK, ...	{'AVIA.JK': 0.15913032763710355, 'BNGA.JK': 0....	
1009	(AVIA.JK, BNGA.JK, CTRA.JK, EMTK.JK, ERAA.JK, ...	{'AVIA.JK': 0.15411944555538157, 'BNGA.JK': 0....	

Expected Return

```
def ER_PortofolioSIM(bobot, parametermarket, stat_data):
    result_list = []
    expected_market_return = stat_data.loc[Ticker_market, 'Expected Ret




    for kombinasi_name, bobot_dict in bobot.items():
        alpha_port = 0
        beta_port = 0
        for saham, bobot_value in bobot_dict.items():
            alpha_port += bobot_value * parametermarket.loc[saham, 'Al
            beta_port += bobot_value * parametermarket.loc[saham, 'Bet

        er_port = alpha_port + (beta_port * expected_market_return)

        result_list.append({
            'Kombinasi': kombinasi_name,
            'ER Portofolio': er_port
        })

    return pd.DataFrame(result_list)
```



```
expected_return_PortofolioSIM1 = ER_PortofolioSIM(Bobot1_PortofolioSIM,
expected_return_PortofolioSIM1
```

	Kombinasi	ER Portofolio	
0	(ACES.JK, EMTK.JK)	0.003689	
1	(ACES.JK, GOTO.JK)	0.003217	
2	(ACES.JK, SCMA.JK)	0.005074	
3	(EMTK.JK, GOTO.JK)	0.003656	
4	(EMTK.JK, SCMA.JK)	0.004721	
5	(GOTO.JK, SCMA.JK)	0.004581	
6	(ACES.JK, EMTK.JK, GOTO.JK)	0.003558	
7	(ACES.JK, EMTK.JK, SCMA.JK)	0.004624	
8	(ACES.JK, GOTO.JK, SCMA.JK)	0.004484	
9	(EMTK.JK, GOTO.JK, SCMA.JK)	0.004381	
10	(ACES.JK, EMTK.JK, GOTO.JK, SCMA.JK)	0.004314	

Next steps: [New interactive sheet](#)

```
expected_return_PortofolioSIM2 = ER_PortofolioSIM(Bobot2_PortofolioSIM,
expected_return_PortofolioSIM2
```

Kombinasi ER Portofolio 

0	(AVIA.JK, BNGA.JK)	0.001477	
1	(AVIA.JK, CMRY.JK)	0.001540	
2	(AVIA.JK, CTRA.JK)	0.001718	
3	(AVIA.JK, EMTK.JK)	0.003944	
4	(AVIA.JK, ERAA.JK)	0.001715	
...	
1008	(AVIA.JK, BNGA.JK, CMRY.JK, EMTK.JK, ERAA.JK, ...	0.003168	
1009	(AVIA.JK, BNGA.JK, CTRA.JK, EMTK.JK, ERAA.JK, ...	0.003155	
1010	(AVIA.JK, CMRY.JK, CTRA.JK, EMTK.JK, ERAA.JK, ...	0.003201	
1011	(BNGA.JK, CMRY.JK, CTRA.JK, EMTK.JK, ERAA.JK, ...	0.003344	
1012	(AVIA.JK, BNGA.JK, CMRY.JK, CTRA.JK, EMTK.JK, ...	0.003091	

1013 rows × 2 columns

Next steps: [New interactive sheet](#)

✓ Risiko

```
import pandas as pd
def Risk_PortofolioSIM(bobot, parametermarket, stat_data):
    results_list = []
    market_variance = stat_data.loc[Ticker_market, 'Variance']




    for kombinasi_name, bobot_dict in bobot.items():
        beta_port = 0
        resvar_port = 0
        for saham, bobot_value in bobot_dict.items():
            beta_port += (bobot_value * parametermarket.loc[saham, 'Beta'])
            resvar_port += (bobot_value**2 * parametermarket.loc[saham, 'Residual_Variance'])

        risikovar = (beta_port**2) * market_variance + resvar_port
        risiko = risikovar**0.5

        results_list.append({
            'Kombinasi': kombinasi_name,
            'Risikovar_Port': risikovar,
            'Risiko_Port': risiko
        })

    return pd.DataFrame(results_list)
```




```
Risiko_PortofolioSIM1 = Risk_PortofolioSIM(Bobot1_PortofolioSIM, parametermarket, stat_data)
Risiko_PortofolioSIM1
```

	Kombinasi	Risikovar_Port	Risiko_Port	
0	(ACES.JK, EMTK.JK)	0.001047	0.032350	
1	(ACES.JK, GOTO.JK)	0.001041	0.032262	
2	(ACES.JK, SCMA.JK)	0.000978	0.031269	
3	(EMTK.JK, GOTO.JK)	0.000651	0.025506	
4	(EMTK.JK, SCMA.JK)	0.000574	0.023968	
5	(GOTO.JK, SCMA.JK)	0.000589	0.024277	
6	(ACES.JK, EMTK.JK, GOTO.JK)	0.000613	0.024764	
7	(ACES.JK, EMTK.JK, SCMA.JK)	0.000549	0.023437	
8	(ACES.JK, GOTO.JK, SCMA.JK)	0.000563	0.023723	
9	(EMTK.JK, GOTO.JK, SCMA.JK)	0.000416	0.020404	
10	(ACES.JK, EMTK.JK, GOTO.JK, SCMA.JK)	0.000403	0.020074	

Next steps: [New interactive sheet](#)

Risiko_PortofolioSIM2 = Risk_PortofolioSIM(Bobot2_PortofolioSIM, param

Risiko_PortofolioSIM2



	Kombinasi	Risikovar_Port	Risiko_Port	
0	(AVIA.JK, BNGA.JK)	0.000253	0.015892	
1	(AVIA.JK, CMRY.JK)	0.000294	0.017153	
2	(AVIA.JK, CTRA.JK)	0.000304	0.017422	
3	(AVIA.JK, EMTK.JK)	0.000640	0.025308	
4	(AVIA.JK, ERAA.JK)	0.000339	0.018420	
...	
1008	(AVIA.JK, BNGA.JK, CMRY.JK, EMTK.JK, ERAA.JK, ...	0.000302	0.017378	
1009	(AVIA.JK, BNGA.JK, CTRA.JK, EMTK.JK, ERAA.JK, ...	0.000306	0.017481	
1010	(AVIA.JK, CMRY.JK, CTRA.JK, EMTK.JK, ERAA.JK, ...	0.000310	0.017613	
1011	(BNGA.JK, CMRY.JK, CTRA.JK, EMTK.JK, ERAA.JK, ...	0.000357	0.018882	

Next steps: [New interactive sheet](#)



∨ Indeks Sharpe Portofolio Biasa

```
def Indeks_Sharpe(ER_Portofolio, Risiko_Portofolio, risk_free):
    sharpe = (ER_Portofolio['ER Portofolio'] - risk_free) / Risiko_Portofolio
    df_sharpe = pd.DataFrame({'Kombinasi': ER_Portofolio['Kombinasi'], 'Sharpe': sharpe})
    return df_sharpe
```

```
Sharpe_PortofolioSIM1 = Indeks_Sharpe(expected_return_PortofolioSIM1, Risiko_PortofolioSIM1, risk_free)
df_sharpeSIM1 = pd.DataFrame(Sharpe_PortofolioSIM1)
```

	Kombinasi	Sharpe Ratio	
0	(ACES.JK, EMTK.JK)	0.106127	
1	(ACES.JK, GOTO.JK)	0.091785	
2	(ACES.JK, SCMA.JK)	0.154078	
3	(EMTK.JK, GOTO.JK)	0.133322	
4	(EMTK.JK, SCMA.JK)	0.186298	
5	(GOTO.JK, SCMA.JK)	0.178167	
6	(ACES.JK, EMTK.JK, GOTO.JK)	0.133333	
7	(ACES.JK, EMTK.JK, SCMA.JK)	0.186372	
8	(ACES.JK, GOTO.JK, SCMA.JK)	0.178206	
9	(EMTK.JK, GOTO.JK, SCMA.JK)	0.202176	
10	(ACES.JK, EMTK.JK, GOTO.JK, SCMA.JK)	0.202170	

```
Sharpe_PortofolioSIM2 = Indeks_Sharpe(expected_return_PortofolioSIM2, Risiko_PortofolioSIM2, risk_free)
df_sharpeSIM2 = pd.DataFrame(Sharpe_PortofolioSIM2)
```

	Kombinasi	Sharpe Ratio	
0	(AVIA.JK, BNGA.JK)	0.078085	
1	(AVIA.JK, CMRY.JK)	0.075998	
2	(AVIA.JK, CTRA.JK)	0.085038	
3	(AVIA.JK, EMTK.JK)	0.146520	
4	(AVIA.JK, ERAA.JK)	0.080315	
...	
1008	(AVIA.JK, BNGA.JK, CMRY.JK, EMTK.JK, ERAA.JK, ...)	0.168735	
1009	(AVIA.JK, BNGA.JK, CTRA.JK, EMTK.JK, ERAA.JK, ...)	0.167006	
1010	(AVIA.JK, CMRY.JK, CTRA.JK, EMTK.JK, ERAA.JK, ...)	0.168329	
1011	(BNGA.JK, CMRY.JK, CTRA.JK, EMTK.JK, ERAA.JK, ...)	0.164584	
1012	(AVIA.JK, BNGA.JK, CMRY.JK, CTRA.JK, EMTK.JK, ERAA.JK, ...)	0.168578	

1013 rows × 2 columns

✓ Value at Risk Adjusted Sharpe

✓ Excess Return

```
import numpy as np
import pandas as pd

def excess_return_portfolio(bobot_portofolio, data_return, risk_free):
    excess_return_data = {}

    for portfolio_data in bobot_portofolio:
        kombinasi_tickers = portfolio_data['Combination']
        weights = portfolio_data['Weights']
        stock_excess_returns = data_return[list(kombinasi_tickers)] - risk_free
        weighted_excess_return = stock_excess_returns.dropna().dot(weights)
        column_name = str(kombinasi_tickers)
        excess_return_data[column_name] = weighted_excess_return

    excess_return_portfolios_df = pd.DataFrame(excess_return_data)

    return excess_return_portfolios_df
```

```
Excess1 = excess_return_portfolio(Bobot1_PortofolioSIM_list, data_return, risk_free)
display(Excess1)
```

	('ACES.JK', 'EMTK.JK')	('ACES.JK', 'GOTO.JK')	('ACES.JK', 'SCMA.JK')	('EMTK.JK', 'GOTO.JK')	('EMTK.JK', 'SCMA.JK')
Date					
2024-08-26	0.004056	-0.018543	-0.015682	-0.006641	-0.0079
2024-08-27	0.000913	0.001038	-0.014917	-0.000256	-0.0100
2024-08-28	-0.005348	-0.018884	-0.000506	-0.012071	-0.0021
2024-08-29	-0.019313	-0.000256	-0.015922	-0.010900	-0.0180
2024-08-30	0.022328	0.016641	-0.001015	0.022203	0.0094
...
2025-02-17	-0.015948	0.010853	-0.000484	-0.003017	-0.0065
2025-02-18	0.021946	-0.035978	-0.000716	-0.005092	0.0091
2025-02-19	0.007260	0.011534	0.033552	0.009894	0.0245

121 rows × 11 columns

New interactive sheet

```
Excess2 = excess_return_portfolio(Bobot2_PortofolioSIM_list, data_return)
display(Excess2)
```

('AVIA.JK', 'BNGA.JK')	('AVIA.JK', 'CMRY.JK')	('AVIA.JK', 'CTRA.JK')	('AVIA.JK', 'EMTK.JK')	('AVIA.JK', 'ERAA.JK')
---------------------------	---------------------------	---------------------------	---------------------------	---------------------------

Date					
2025-02-25	-0.050176	-0.054437	-0.055124	-0.105871	-0.0467
2025-02-26	0.007063	0.004286	0.006277	0.005112	0.0051
2025-02-27	0.013287	0.016950	0.023105	0.019921	0.0170
2025-02-28	0.017001	0.030812	0.008994	-0.006889	0.0184
2025-03-03	0.018849	0.008996	0.020307	0.021947	0.0248
...
2025-08-15	-0.001665	0.002116	-0.007086	-0.033145	-0.0002
2025-08-19	0.000479	-0.002926	0.001147	0.066198	0.0019
2025-08-20	-0.000961	-0.002495	0.000503	0.000084	-0.0041
2025-08-21	0.001203	0.001684	-0.005145	-0.015771	0.0022
2025-08-22	0.003313	0.007054	-0.001112	0.060373	0.0029

112 rows \times 1013 columns

- Stats Excess Return

```
import pandas as pd

def excessreturn_stat(excess_returns):
```

```
def excessreturn_stat(excess_returns):
    stats_list = []
    for combination, excess_returns in excess_returns.items():
        mean_excess_return = excess_returns.mean()
        std_dev_excess_return = excess_returns.std()
        skewness_excess_return = excess_returns.skew()
        kurtosis_excess_return = excess_returns.kurtosis()

        stats_list.append({
            'Combination': combination,
            'Mean Excess Return': mean_excess_return,
            'Excess Return Std Dev': std_dev_excess_return,
            'Excess Return Skewness': skewness_excess_return,
            'Excess Return Kurtosis': kurtosis_excess_return
        })

    return pd.DataFrame(stats_list)
```


```
excess_return_stats1 = excessreturn_stat(Excess1)
display(excess_return_stats1)
```

	Combination	Mean Excess Return	Excess Return Std Dev	Excess Return Skewness	Excess Return Kurtosis
0	('ACES.JK', 'EMTK.JK')	0.003433	0.032399	0.635840	1.386320
1	('ACES.JK', 'GOTO.JK')	0.002961	0.032381	0.537792	0.440321
2	('ACES.JK', 'SCMA.JK')	0.004818	0.031397	1.509967	5.199282
3	('EMTK.JK', 'GOTO.JK')	0.003401	0.026886	0.273017	-0.119355
4	('EMTK.JK', 'SCMA.JK')	0.004465	0.028924	1.135623	3.390965
5	('GOTO.JK', 'SCMA.JK')	0.004325	0.025723	0.960795	3.138950
6	('ACES.JK', 'EMTK.JK', 'GOTO.JK')	0.003302	0.026153	0.255474	-0.134679

Next steps: [New interactive sheet](#)

```
excess_return_stats2 = excessreturn_stat(Excess2)
display(excess_return_stats2)
```

	Combination	Mean Excess Return	Excess Return Std Dev	Excess Return Skewness	Excess Return Kurtosis
--	-------------	--------------------------	-----------------------------	------------------------------	------------------------------

0	('AVIA.JK', 'BNGA.JK')	0.001241	0.015738	0.161395	2.177744	
1	('AVIA.JK', 'CMRY.JK')	0.001304	0.017531	0.249900	1.669533	
2	('AVIA.JK', 'CTRA.JK')	0.001482	0.017687	0.098624	1.141710	
3	('AVIA.JK', 'EMTK.JK')	0.003708	0.026058	-0.382815	3.076368	
4	('AVIA.JK', 'ERAA.JK')	0.001479	0.018029	0.392984	1.862600	
...	
1008	('AVIA.JK', 'BNGA.JK', 'CMRY.JK', 'EMTK.JK', '...	0.002932	0.018197	-1.128016	3.965734	
1009	('AVIA.JK', 'BNGA.JK', 'CTRA.JK', 'EMTK.JK', '...	0.002919	0.018353	-1.079736	3.698711	

Next steps: [New interactive sheet](#)

✓ VaRSR

```

import numpy as np
import math
from scipy.stats import norm
import pandas as pd

def VaRSR(excessreturn_stats, excessreturn, confidence_level):
    z_score = norm.ppf(confidence_level)
    n = len(excessreturn)
    estimated_sharpe = excessreturn_stats['Mean Excess Return'] / exce
    sharpe_squared = estimated_sharpe**2
    sharpe_squared_term = sharpe_squared / 2
    sharpe_skewness_term = estimated_sharpe * excessreturn_stats['Exce
    sharpe_kurtosis_term = sharpe_squared * ((excessreturn_stats['Exce

    std_dev_sharpe = ((1 / (n - 1)) * (1 + sharpe_squared_term - sharpe
    var_adjusted_sharpe = estimated_sharpe - z_score * std_dev_sharpe

    var_adjusted_sharpe_data = pd.DataFrame({
        'Combination': excessreturn_stats['Combination'],
        'Estimated Sharpe': estimated_sharpe,
        'Std Dev Sharpe': std_dev_sharpe,
        'VaRSR': var_adjusted_sharpe
    })

```

```
return var_adjusted_sharpe_data
```

```
var_adjusted_sharpe_stats1 = VaRSR (excess_return_stats1, Excess1, 0.9!  
display(var_adjusted_sharpe_stats1)
```

	Combination	Estimated Sharpe	Std Dev Sharpe	VaRSR
0	('ACES.JK', 'EMTK.JK')	0.105967	0.088209	-0.039124
1	('ACES.JK', 'GOTO.JK')	0.091447	0.088959	-0.054878
2	('ACES.JK', 'SCMA.JK')	0.153449	0.081292	0.019735
3	('EMTK.JK', 'GOTO.JK')	0.126478	0.089489	-0.020718
4	('EMTK.JK', 'SCMA.JK')	0.154377	0.083613	0.016847
5	('GOTO.JK', 'SCMA.JK')	0.168153	0.084339	0.029429
6	('ACES.JK', 'EMTK.JK', 'GOTO.JK')	0.126253	0.089593	-0.021114
7	('ACES.JK', 'EMTK.JK', 'SCMA.JK')	0.154119	0.083653	0.016523
8	('ACES.JK', 'GOTO.JK', 'SCMA.JK')	0.167615	0.084386	0.028812

Next steps: [New interactive sheet](#)

```
var_adjusted_sharpe_stats2 = VaRSR (excess_return_stats2, Excess2, 0.9!  
display(var_adjusted_sharpe_stats2)
```

	Combination	Estimated Sharpe	Std Dev Sharpe	VaRSR
0	('AVIA.JK', 'BNGA.JK')	0.078850	0.094397	-0.076419
1	('AVIA.JK', 'CMRY.JK')	0.074356	0.094074	-0.080383
2	('AVIA.JK', 'CTRA.JK')	0.083761	0.094535	-0.071735
3	('AVIA.JK', 'EMTK.JK')	0.142302	0.097951	-0.018814
4	('AVIA.JK', 'ERAA.JK')	0.082057	0.093443	-0.071642
...
1008	('AVIA.JK', 'BNGA.JK', 'CMRY.JK', 'EMTK.JK', '...	0.161137	0.104019	-0.009959
1009	('AVIA.JK', 'BNGA.JK', 'CTRA.JK', 'EMTK.JK', '...	0.159066	0.103490	-0.011160
1010	('AVIA.JK', 'CMRY.JK', 'CTRA.JK', 'EMTK.JK', '...	0.159690	0.103300	-0.010223
	('BNGA.JK', 'CMRY.JK')			

1011

(ACES.JK, EMTK.JK)

0.155391

0.102563

-0.013311

Next steps: [New interactive sheet](#)

✓ Kesimpulan

✓ Data Hasil Akhir

```
HasilakhirP1 = pd.concat([expected_return_PortofolioSIM1, Risiko_PortofolioSIM1], axis=1)
display(HasilakhirP1)
```

	Kombinasi	ER Portofolio	Risiko_Port	Sharpe Ratio	VaRSR
0	(ACES.JK, EMTK.JK)	0.003689	0.032350	0.106127	-0.039124
1	(ACES.JK, GOTO.JK)	0.003217	0.032262	0.091785	-0.054878
2	(ACES.JK, SCMA.JK)	0.005074	0.031269	0.154078	0.019735
3	(EMTK.JK, GOTO.JK)	0.003656	0.025506	0.133322	-0.020718
4	(EMTK.JK, SCMA.JK)	0.004721	0.023968	0.186298	0.016847
5	(GOTO.JK, SCMA.JK)	0.004581	0.024277	0.178167	0.029429
6	(ACES.JK, EMTK.JK,	0.003558	0.024764	0.133333	-0.021114

Next steps: [New interactive sheet](#)

```
HasilakhirP2 = pd.concat([expected_return_PortofolioSIM2, Risiko_PortofolioSIM2], axis=1)
display(HasilakhirP2)
```

	Kombinasi	ER Portofolio	Risiko_Port	Sharpe Ratio	VaRSR
0	(AVIA.JK, BNGA.JK)	0.001477	0.015892	0.078085	-0.076419
1	(AVIA.JK, CMRY.JK)	0.001540	0.017153	0.075998	-0.080383
2	(AVIA.JK, CTRA.JK)	0.001718	0.017422	0.085038	-0.071735
3	(AVIA.JK, CTRA.JK)	0.003944	0.025308	0.146520	-0.018814

	EMTK.JK)				
4	(AVIA.JK, ERAA.JK)	0.001715	0.018420	0.080315	-0.071642
...
1008	(AVIA.JK, BNGA.JK, CMRY.JK, EMTK.JK, ERAA.JK, ...	0.003168	0.017378	0.168735	-0.009959

Next steps: [New interactive sheet](#)

```
def save_to_excel(dataframes, file_name):
    with pd.ExcelWriter(file_name) as writer:
        for sheet_name, df in dataframes.items():
            df.to_excel(writer, sheet_name=sheet_name)
    print(f"Data saved to '{file_name}'")

dataframes_to_save = {
    'Stock Data Period 1': data_periode1,
    'Stock Data Period 2': data_periode2,
    'Return Period 1': data_returnperiode1,
    'Return Period 2': data_returnperiode2,
    'Statistics Period 1': stat_periode1,
    'Statistics Period 2': stat_periode2,
    'Parameter Market Period 1': parameter_periode1,
    'Parameter Market Period 2': parameter_periode2,
    'Parameter SIM Period 1': sim_periode1,
    'Parameter SIM Period 2': sim_periode2,
    'Portfolio Weight SIM Period 1': pd.DataFrame(Bobot1_PortofolioSIM_1),
    'Portfolio Weight SIM Period 2': pd.DataFrame(Bobot2_PortofolioSIM_1),
    'Expected Return Portfolio SIM Period 1': expected_return_PortofolioSIM_1,
    'Expected Return Portfolio SIM Period 2': expected_return_PortofolioSIM_2,
    'Risiko Portofolio SIM Period 1': Risiko_PortofolioSIM1,
    'Risiko Portofolio SIM Period 2': Risiko_PortofolioSIM2,
    'Indeks Sharpe SIM Period 1': Sharpe_PortofolioSIM1,
    'Indeks Sharpe SIM Period 2': Sharpe_PortofolioSIM2,
    'Excess Return SIM Period 1': Excess1,
    'Excess Return SIM Period 2': Excess2,
    'Statistics Excess Return SIM Period 1': excess_return_stats1,
    'Statistics Excess Return SIM Period 2': excess_return_stats2,
    'VaRSR SIM Period 1': var_adjusted_sharpe_stats1,
    'VaRSR SIM Period 2': var_adjusted_sharpe_stats2,
    'Final Result Period 1': HasilakhirP1,
    'Final Result Period 2': HasilakhirP2
}
```

```
save_to_excel(dataframes_to_save, 'Portfolio Analysis Result.xlsx')
```

```
/usr/local/lib/python3.12/dist-packages/openpyxl/workbook/child.py:99: UserWarning: Title is more than 31 characters. Some applications may truncate the title.
  warnings.warn("Title is more than 31 characters. Some applications may truncate the title.")
Data saved to 'Portfolio Analysis Result.xlsx'
```

✓ Scatterplot

```
import matplotlib
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import numpy as np
import pandas as pd

def plot_portfolio_scatter(data, metrics=['Sharpe Ratio', 'VaRSR'], max_returns=0.05):
    fig, axes = plt.subplots(1, 2, figsize=(12, 5))
    data['Num Stocks'] = data['Kombinasi'].apply(len)
    markers = ['o', 's', '^', 'v', 'D', 'P', 'X', '*', '<', '>']

    unique_num_stocks = sorted(data['Num Stocks'].unique())
    marker_map = {n: markers[i % len(markers)] for i, n in enumerate(unique_num_stocks)}

    stock_handles = []
    stock_labels = []
    added_labels = set()

    for i, color_by in enumerate(metrics):
        ax = axes[i]
        color_data = data[color_by]
        cmap = matplotlib.colors.LinearSegmentedColormap.from_list("", color_data)
        norm_color = plt.Normalize(color_data.min(), color_data.max())
        first_scatter = None
        for j, num in enumerate(unique_num_stocks):
            subset = data[data['Num Stocks'] == num]
            label = f'{num}' if i == 0 and f'{num}' not in added_labels else None

            scatter = ax.scatter(subset['Risiko_Port'], subset['ER_Port'],
                                c=subset[color_by], cmap=cmap, norm=norm_color,
                                marker=marker_map.get(num, 'o'), alpha=0.5)

            if first_scatter is None:
                first_scatter = scatter
            if i == 0 and f'{num}' not in added_labels:
                h, l = ax.get_legend_handles_labels()
                for handle, current_label in zip(h, l):
                    if current_label == f'{num}':
                        stock_handles.append(handle)
                        stock_labels.append(current_label)
                        added_labels.add(current_label)
                        break

    if not data.empty:
        max_p = data.loc[color_data.idxmax()]
        metric_name = color_by
        print(f"Highest Portfolio Performance by {metric_name}:", max_p)

    ax.scatter(max_p['Risiko_Port'], max_p['ER_Port'], color='red', marker='x', s=100)
```

```

ax.scatter(max_portfolio['Risiko_Port'], max_portfolio['ER Portofolio'],
           c='lime', s=125, marker='*',
           edgecolors='black', linewidth=1, zorder=3)

ax.text(0.5, -0.25, f"★ Optimal Portfolio: {'', ' '.join(max_
horizontalalignment='center', verticalalignment='center',
fontfamily='serif',
fontweight='bold',
fontstyle='italic',
fontvariant='small-caps',
fontsize=9, bbox=dict(boxstyle='round,pad=0.3', facecolor='white',
                        edgecolor='black', zorder=10))

cbar = fig.colorbar(first_scatter, ax=ax)
cbar.set_label(color_by)

ax.set_xlabel("Portfolio's Risk")
ax.set_ylabel("Portfolio's Expected Return")
ax.set_title(f'Color By: {color_by}')
ax.grid(True, linestyle='--', alpha=0.5)

fig.legend(stock_handles, stock_labels,
           title='Portfolio Size',
           loc='upper center',
           bbox_to_anchor=(0.5, 1.02),
           ncol=len(stock_handles),
           fancybox=True,
           shadow=True)

fig.suptitle(main_title, fontsize=14, y=1.08)
plt.tight_layout(rect=[0, 0.05, 1, 0.98])
plt.show()

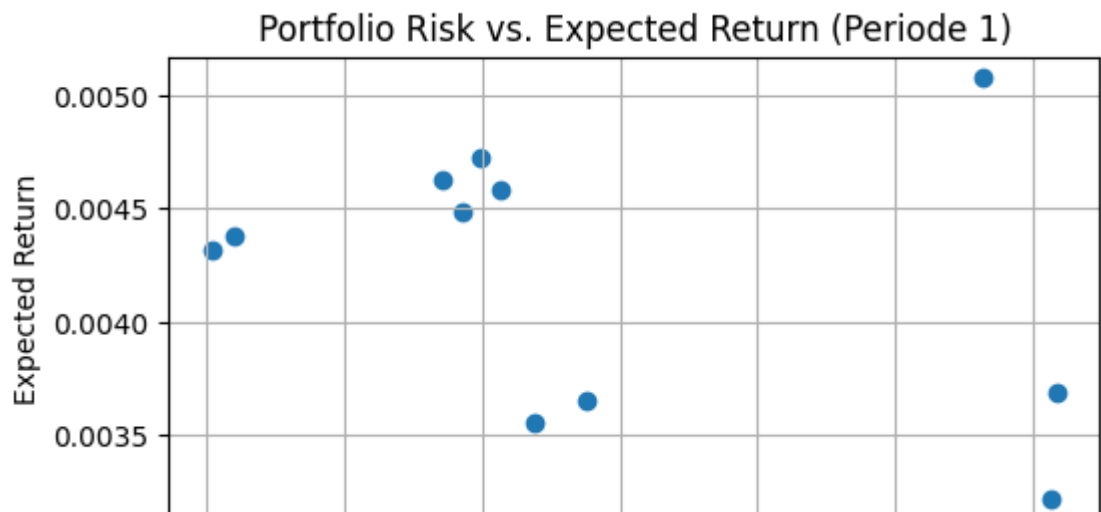
```

```

import matplotlib.pyplot as plt

plt.figure(figsize=(6, 3))
plt.scatter(HasilakhirP1['Risiko_Port'], HasilakhirP1['ER Portofolio'])
plt.xlabel('Portfolio Risk')
plt.ylabel('Expected Return')
plt.title('Portfolio Risk vs. Expected Return (Periode 1)')
plt.grid(True)
plt.show()

```



0.020 0.022 0.024 0.026 0.028 0.030 0.032

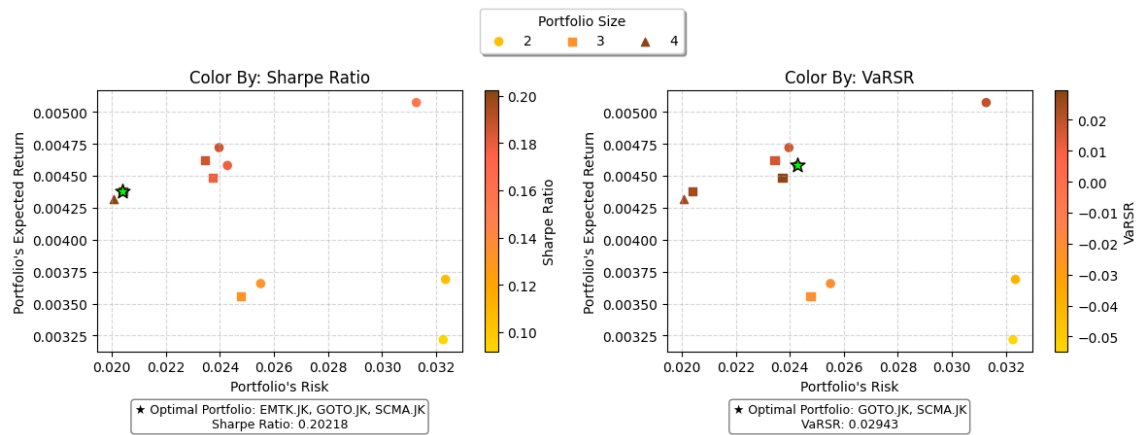
Portfolio Risk

```
plot_portfolio_scatter(HasilakhirP1, main_title = "Portfolio's Expected Return vs. Risk (pre-Danantara Period)")
```

Highest Portfolio Performance by Sharpe Ratio: {'Kombinasi': ('EMTK.JK', 'GOTO.JK', 'SCMA.JK')}

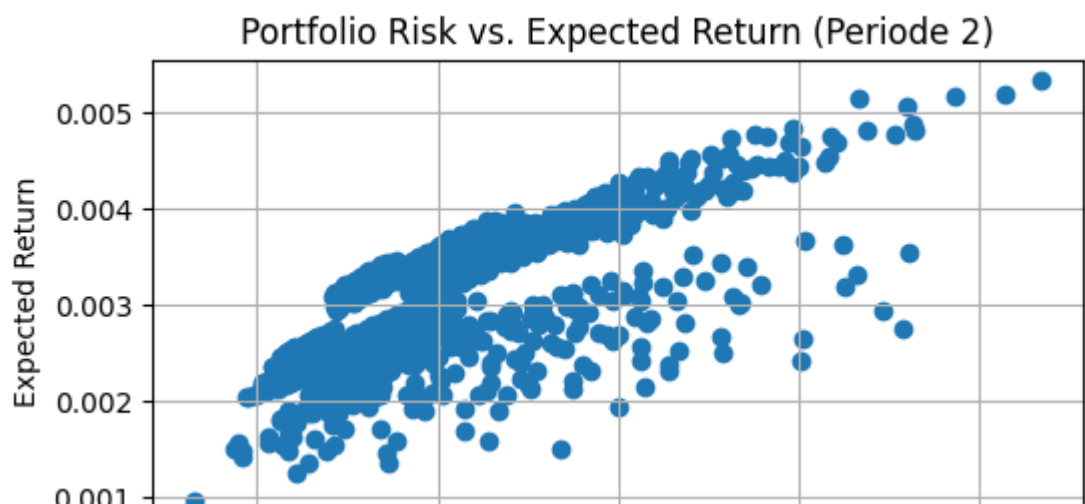
Highest Portfolio Performance by VaRSR: {'Kombinasi': ('GOTO.JK', 'SCMA.JK')}

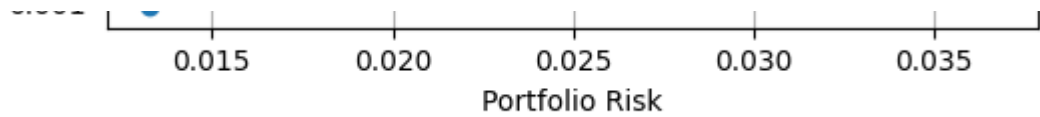
Portfolio's Expected Return vs. Risk (pre-Danantara Period)



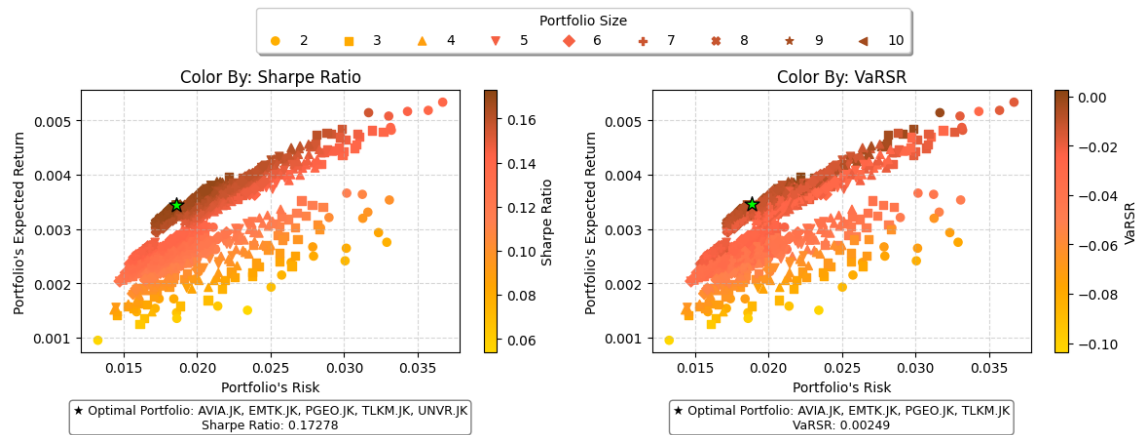
```
import matplotlib.pyplot as plt
```

```
plt.figure(figsize=(6, 3))
plt.scatter((HasilakhirP2['Risiko_Port']), HasilakhirP2['ER Portofolio'])
plt.xlabel('Portfolio Risk')
plt.ylabel('Expected Return')
plt.title('Portfolio Risk vs. Expected Return (Periode 2)')
plt.grid(True)
plt.show()
```





```
plot_portfolio_scatter(HasilakhirP2, main_title= "Portfolio's Expected  
Highest Portfolio Performance by Sharpe Ratio: {'Kombinasi': ('AVIA.JK',  
Highest Portfolio Performance by VaRSR: {'Kombinasi': ('AVIA.JK', 'EMTK.  
Portfolio's Expected Return vs. Risk (post-Danantara Period)
```



```
HasilakhirP1['Periode'] = 'Periode 1'  
HasilakhirP2['Periode'] = 'Periode 2'  
  
if 'Num Stocks' not in HasilakhirP1.columns:  
    HasilakhirP1['Num Stocks'] = HasilakhirP1['Kombinasi'].apply(lambda  
  
if 'Num Stocks' not in HasilakhirP2.columns:  
    HasilakhirP2['Num Stocks'] = HasilakhirP2['Kombinasi'].apply(lambda  
  
averageP1 = HasilakhirP1.groupby('Num Stocks')[['ER Portofolio', 'Risiko  
averageP1
```

	ER Portofolio	Risiko_Port	Sharpe Ratio	VaRSR	
Num Stocks					
2	0.004157	0.028272	0.141630	-0.008118	
3	0.004262	0.023082	0.175022	0.012264	
4	0.004314	0.020074	0.202170	0.024438	

Next steps: [New interactive sheet](#)

```
averageP2 = HasilakhirP2.groupby('Num Stocks')[['ER Portofolio', 'Risiko Portofolio', 'Sharpe Ratio', 'VaRSR']]
averageP2
```

	ER Portofolio	Risiko_Port	Sharpe Ratio	VaRSR	
Num Stocks					
2	0.002899	0.025441	0.101980	-0.054964	
3	0.002993	0.022766	0.119170	-0.041050	
4	0.003035	0.021054	0.131502	-0.032024	
5	0.003057	0.019878	0.140834	-0.025780	
6	0.003070	0.019026	0.148164	-0.021261	
7	0.003079	0.018381	0.154083	-0.017877	
8	0.003084	0.017876	0.158970	-0.015271	
9	0.003088	0.017470	0.163075	-0.013217	
10	0.003091	0.017137	0.166573	-0.011562	

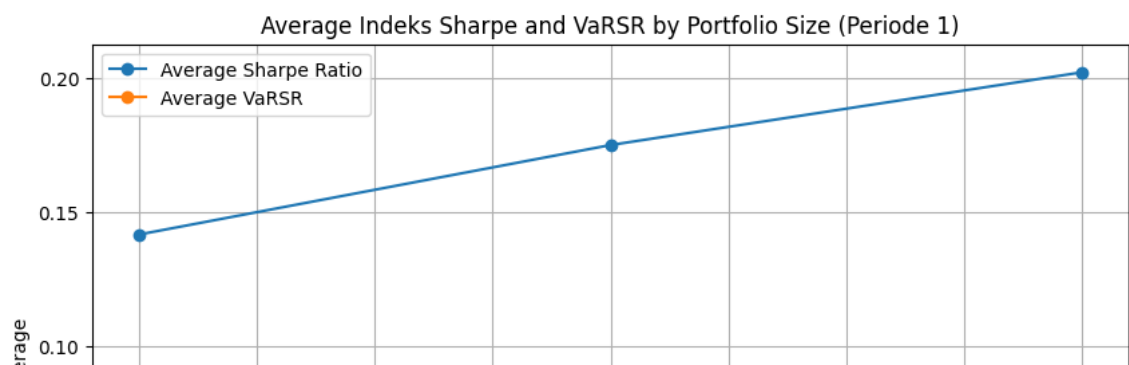
Next steps: [New interactive sheet](#)

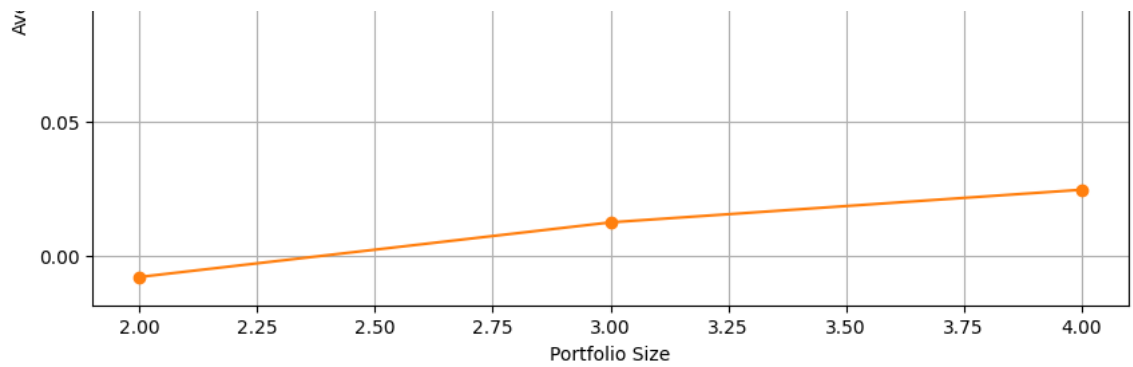
```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))

# Plot the average Sharpe Index and VaRSR against the number of stocks
plt.plot(averageP1.index, averageP1['Sharpe Ratio'], marker='o', label='Average Sharpe Ratio')
plt.plot(averageP1.index, averageP1['VaRSR'], marker='o', label='Average VaRSR')

plt.xlabel('Portfolio Size')
plt.ylabel('Average')
plt.title('Average Indeks Sharpe and VaRSR by Portfolio Size (Periode 1)')
plt.grid(True)
plt.legend()
plt.show()
```





```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))

# Plot the average Sharpe Index and VaRSR against the number of stocks
plt.plot(averageP2.index, averageP2['Sharpe Ratio'], marker='o', label='Average Sharpe Ratio')
plt.plot(averageP2.index, averageP2['VaRSR'], marker='o', label='Average VaRSR')

plt.xlabel('Portfolio Size')
plt.ylabel('Average')
plt.title('Average Indeks Sharpe and VaRSR by Portfolio Size (Periode 2)')
plt.grid(True)
plt.legend()
plt.show()
```

