Install Module

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

```
Collecting yfinance
  Downloading yfinance-0.2.65-py2.py3-none-any.whl.metadata (5.8 kB)
Requirement already satisfied: pandas in /usr/local/lib/python3.12/dist-
Collecting openpyxl
  Downloading openpyxl-3.1.5-py2.py3-none-any.whl.metadata (2.5 kB)
Requirement already satisfied: numpy>=1.16.5 in /usr/local/lib/python3.1
Requirement already satisfied: requests>=2.31 in /usr/local/lib/python3.
Collecting multitasking>=0.0.7 (from yfinance)
  Downloading multitasking-0.0.12.tar.gz (19 kB)
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Requirement already satisfied: platformdirs>=2.0.0 in /usr/local/lib/pyt
Requirement already satisfied: pytz>=2022.5 in /usr/local/lib/python3.12
Collecting frozendict>=2.3.4 (from yfinance)
  Downloading frozendict-2.4.6-py312-none-any.whl.metadata (23 kB)
Collecting peewee>=3.16.2 (from yfinance)
  Downloading peewee-3.18.2.tar.gz (949 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 949.2/949.2 kB 16.0 MB/s eta
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Requirement already satisfied: beautifulsoup4>=4.11.1 in /usr/local/lib/
Collecting curl_cffi>=0.7 (from yfinance)
  Downloading curl_cffi-0.13.0-cp39-abi3-manylinux_2_17_x86_64.manylinux
Requirement already satisfied: protobuf>=3.19.0 in /usr/local/lib/python
Requirement already satisfied: websockets>=13.0 in /usr/local/lib/python
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.
Collecting et-xmlfile (from openpyxl)
  Downloading et_xmlfile-2.0.0-py3-none-any.whl.metadata (2.7 kB)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.1
Requirement already satisfied: typing-extensions>=4.0.0 in /usr/local/li
Requirement already satisfied: cffi>=1.12.0 in /usr/local/lib/python3.12
Requirement already satisfied: certifi>=2024.2.2 in /usr/local/lib/pytho
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dis
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/li
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/pyth
Requirement already satisfied: pycparser in /usr/local/lib/python3.12/di
Downloading yfinance-0.2.65-py2.py3-none-any.whl (119 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 119.4/119.4 kB 7.4 MB/s eta
Downloading openpyxl-3.1.5-py2.py3-none-any.whl (250 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 250.9/250.9 kB 15.5 MB/s eta
Downloading curl_cffi-0.13.0-cp39-abi3-manylinux_2_17_x86_64.manylinux2
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.3/8.3 MB 105.4 MB/s eta 0
Downloading frozendict-2.4.6-py312-none-any.whl (16 kB)
Downloading et_xmlfile-2.0.0-py3-none-any.whl (18 kB)
Building wheels for collected packages: multitasking, peewee
  Building wheel for multitasking (pyproject.toml) ... done
  Created wheel for multitasking: filename=multitasking-0.0.12-py3-none-
```

```
  Stored in directory: /root/.cache/pip/wheels/cc/bd/6f/664d62c99327abee
  Building wheel for peewee (pyproject.toml) ... done
  Created wheel for peewee: filename=peewee-3.18.2-cp312-cp312-linux_x86
  Stored in directory: /root/.cache/pip/wheels/d1/df/a9/0202b051c65b11c9
Successfully built multitasking peewee
Installing collected packages: peewee, multitasking, frozendict, et-xmlf
Successfully installed curl_cffi-0.13.0 et-xmlfile-2.0.0 frozendict-2.4.
```

Pengambilan Data

```
import pandas as pd
daftar_saham = pd.read_excel('Daftar Saham IDX80.xlsx', sheet_name='Sal
ticker_saham = daftar_saham['Kode'].tolist()
ticker_saham
```

```
['ACES.JK',
 'ADRO.JK',
 'AKRA.JK',
 'AMRT.JK',
 'ANTM.JK',
 'ARTO.JK',
 'ASII.JK',
 'BBCA.JK',
 'BBNI.JK',
 'BBRI.JK',
 'BBTN.JK',
 'BFIN.JK',
 'BMRI.JK',
 'BRIS.JK',
 'BRMS.JK',
 'BRPT.JK',
 'BSDE.JK',
 'BUKA.JK',
 'CPIN.JK',
 'CTRA.JK',
 'EMTK.JK',
 'ENRG.JK',
 'ERAA.JK',
 'ESSA.JK',
 'EXCL.JK',
 'GGRM.JK',
 'GOTO.JK',
 'HEAL.JK',
 'HRUM.JK',
 'ICBP.JK',
 'INCO.JK',
 'INDF.JK',
 'INDY.JK',
 'INKP.JK',
 'INTP.JK',
 'ISAT.JK',
 'ITMG.JK',
 'JPFA.JK',
 'JSMR.JK',
 'KLBF.JK',
 'MAPI.JK',
```

```
 'MDKA.JK',
 'MEDC.JK',
 'MIKA.JK',
 'MNCN.JK',
 'MTEL.JK',
 'PGAS.JK',
 'PTBA.JK',
 'PWON.JK',
 'SCMA.JK',
 'SIDO.JK',
 'SMGR.JK',
 'SMRA.JK',
 'SRTG.JK',
 'TKIM.JK',
 'TLKM.JK',
 'TOWR.JK',
 'UNTR.JK',
```

```
# Mengambil Data Saham
import yfinance as yf

def Scraping_Saham(tickers, start_date, end_date):
  try:
    data = yf.download(tickers, start=start_date, end=end_date)
    closing_prices = data['Close']
    return closing_prices
  except Exception as e:
    return None

start_date = '2022-12-31'
end_date = '2024-12-31'
Ticker_market = ('^JKSE')
data_market = Scraping_Saham(Ticker_market, start_date, end_date)

data_saham = pd.DataFrame()
ticker_gagal = []

for ticker in ticker_saham:
    temp_data = Scraping_Saham(ticker, start_date, end_date)
    if temp_data is not None:
      data_saham = pd.concat([data_saham, temp_data], axis=1)
    else:
      ticker_gagal.append(ticker)

while ticker_gagal:
    ticker_gagal_baru = []
    for ticker in ticker_gagal:
        temp_data = Scraping_Saham(ticker, start_date, end_date)
        if temp_data is not None:
            data_saham = pd.concat([data_saham, temp_data], axis=1)
        else:
            ticker_gagal_baru.append(ticker)
    ticker_gagal = ticker_gagal_baru
```

Show hidden output

```
data_close = pd.concat([data_market, data_saham], axis=1)
data_close
```

Ticker	^JKSE	ACES.JK	ADRO.JK	AKRA.JK	AMRT.JK
Date					
2023-01-02	6850.983887	466.891663	1744.700684	1188.366821	2688.893799
2023-01-03	6888.757812	466.891663	1710.681519	1158.335938	2718.334229
2023-01-04	6813.238770	457.736908	1603.763916	1106.854248	2688.893799
2023-01-05	6653.840820	446.751221	1506.566162	1055.372803	2757.588379
2023-01-06	6684.558105	448.582184	1526.005737	1089.693726	2747.774658
...	...	...	...	...	...
2024-12-20	6983.865234	795.000000	2550.000000	1140.000000	2780.000000
2024-12-23	7096.444824	810.000000	2540.000000	1130.000000	2940.000000
2024-12-24	7065.746094	795.000000	2510.000000	1120.000000	2780.000000
2024-12-27	7036.570801	795.000000	2540.000000	1120.000000	2800.000000
2024-12-30	7079.904785	790.000000	2430.000000	1120.000000	2850.000000

476 rows × 60 columns

```
# Menyimpan Data Bunga Bebas Risiko
data_sukubunga = pd.read_excel('Suku Bunga BI.xlsx', sheet_name='Des 2
data_sukubunga_harian = data_sukubunga['Suku Bunga']/250
risk_free = data_sukubunga_harian.mean()
risk_free
```

```
0.0002365384615384615
```

Return

```
def returns(data):
    return_harian = ((data) - (data).shift(1)) / (data).shift(1)
    return return_harian

return_saham = returns(data_saham)
return_market = returns(data_market)
data_return = pd.concat([return_market, return_saham], axis=1).iloc[1:
data_return
```

Ticker	^JKSE	ACES.JK	ADRO.JK	AKRA.JK	AMRT.JK	ANTM.JK	-
Date							
2023-01-03	0.005514	0.000000	-0.019499	-0.025271	0.010949	0.010101	-
2023-01-04	-0.010963	-0.019608	-0.062500	-0.044445	-0.010830	0.025000	-
2023-01-05	-0.023395	-0.024000	-0.060606	-0.046511	0.025548	-0.041463	-
2023-01-06	0.004616	0.004098	0.012903	0.032520	-0.003559	0.010178	-
2023-01-09	0.000555	0.004082	-0.035032	-0.003937	-0.021429	0.037783	-
...	...	...	...	...	...	...	
2024-12-20	0.000950	-0.006250	0.003937	0.017857	-0.010676	-0.034014	-
2024-12-23	0.016120	0.018868	-0.003922	-0.008772	0.057554	0.024648	
2024-12-24	-0.004326	-0.018519	-0.011811	-0.008850	-0.054422	-0.013746	-
2024-12-27	-0.004129	0.000000	0.011952	0.000000	0.007194	0.062718	-
2024-12-30	0.006158	-0.006289	-0.043307	0.000000	0.017857	0.000000	

475 rows × 60 columns

```
# Expected Return
def expected_returns(data):
  return data.mean()
ER_saham = expected_returns(data_return)

#Standar Deviasi
def st_dev(data):
  return data.std()
std_saham = st_dev(data_return)

#Variance
def var(data):
  return data.var()
var_saham = var(data_return)

#Covarian
def covarian(data):
  cov_matrix = data.cov()
  return cov_matrix[Ticker_market]
cov_saham = covarian(data_return)

pd.DataFrame({'Expected Return': ER_saham, 'St. Deviation': std_saham,
```

Ticker	Expected Return	St. Deviation	Variance	Covariance
^JKSE	0.000096	0.007254	0.000053	0.000053

ACES.JK	0.001452	0.026486	0.000701	0.000033
ADRO.JK	0.001077	0.027376	0.000749	0.000053
AKRA.JK	0.000067	0.019607	0.000384	0.000037
AMRT.JK	0.000286	0.018119	0.000328	0.000045
ANTM.JK	-0.000089	0.020364	0.000415	0.000045
ARTO.JK	-0.000069	0.039900	0.001592	0.000093
ASII.JK	0.000240	0.015314	0.000235	0.000049
BBCA.JK	0.000450	0.012668	0.000160	0.000050
BBNI.JK	0.000193	0.016034	0.000257	0.000064
BBRI.JK	0.000060	0.015884	0.000252	0.000072
BBTN.JK	-0.000009	0.018570	0.000345	0.000066
BFIN.JK	0.000235	0.023808	0.000567	0.000051
BMRI.JK	0.000656	0.016981	0.000288	0.000078
BRIS.JK	0.001935	0.025447	0.000648	0.000071
BRMS.JK	0.002241	0.035634	0.001270	0.000052
BRPT.JK	0.001164	0.039746	0.001580	0.000127
BSDE.JK	0.000242	0.018727	0.000351	0.000055
BUKA.JK	-0.001162	0.029107	0.000847	0.000059
CPIN.JK	-0.000137	0.019094	0.000365	0.000039
CTRA.JK	0.000415	0.020643	0.000426	0.000054
EMTK.JK	-0.001051	0.030131	0.000908	0.000079
ENRG.JK	-0.000048	0.031332	0.000982	0.000062
ERAA.JK	0.000583	0.025405	0.000645	0.000055
ESSA.JK	0.000402	0.033395	0.001115	0.000061
EXCL.JK	0.000387	0.021265	0.000452	0.000034
GGRM.JK	-0.000331	0.021096	0.000445	0.000040
GOTO.JK	0.000239	0.041563	0.001728	0.000101
HEAL.JK	0.000286	0.020355	0.000414	0.000018
HRUM.JK	-0.000600	0.026236	0.000688	0.000063
ICBP.JK	0.000447	0.016211	0.000263	0.000028
INCO.JK	-0.001144	0.021568	0.000465	0.000029
INDF.JK	0.000516	0.012652	0.000160	0.000023
INDY.JK	-0.000589	0.025734	0.000662	0.000062

INKP.JK	-0.000240	0.022146	0.000490	0.000049
INTP.JK	-0.000385	0.018850	0.000355	0.000029
ISAT.JK	0.001649	0.022558	0.000509	0.000040
ITMG.JK	0.000130	0.018557	0.000344	0.000044
JPFA.JK	0.001197	0.021314	0.000454	0.000034
JSMR.JK	0.000946	0.019455	0.000379	0.000040
KLBF.JK	-0.000627	0.018297	0.000335	0.000030
MAPI.JK	0.000415	0.028089	0.000789	0.000047
MDKA.JK	-0.001598	0.027490	0.000756	0.000069
MEDC.JK	0.000656	0.031301	0.000980	0.000060
MIKA.JK	-0.000056	0.021033	0.000442	0.000005
MNCN.JK	-0.001769	0.024822	0.000616	0.000060
MTEL.JK	-0.000145	0.016540	0.000274	0.000022
PGAS.JK	0.000392	0.017758	0.000315	0.000041
PTBA.JK	0.000668	0.022177	0.000492	0.000051
PWON.JK	-0.000062	0.016564	0.000274	0.000044
SCMA.JK	0.000162	0.027902	0.000779	0.000045
SIDO.JK	-0.000088	0.018643	0.000348	0.000032
SMGR.JK	-0.001137	0.020122	0.000405	0.000050
SMRA.JK	-0.000106	0.022893	0.000524	0.000060
SRTG.JK	0.000210	0.029756	0.000885	0.000068
TKIM.JK	-0.000077	0.023279	0.000542	0.000064
TLKM.JK	-0.000374	0.015904	0.000253	0.000040
TOWR.JK	-0.000892	0.019503	0.000380	0.000052
UNTR.JK	0.000991	0.019905	0.000396	0.000039
UNVR.JK	-0.001455	0.021160	0.000448	0.000035

```
# Ambil yang Positif saja
def return_positif(data_return):
    saham_return_positif = data_return.loc[:, data_return.mean() > 0]
    return saham_return_positif

data_return = return_positif(data_return)
data_return = pd.concat([data_return], axis=1)
data_return
```

Ticker	^JKSE	ACES.JK	ADRO.JK	AKRA.JK	AMRT.JK	ASII.JK	
Date							
2023-01-03	0.005514	0.000000	-0.019499	-0.025271	0.010949	-0.008772	
2023-01-04	-0.010963	-0.019608	-0.062500	-0.044445	-0.010830	0.004425	-
2023-01-05	-0.023395	-0.024000	-0.060606	-0.046511	0.025548	-0.052863	-
2023-01-06	0.004616	0.004098	0.012903	0.032520	-0.003559	0.013953	
2023-01-09	0.000555	0.004082	-0.035032	-0.003937	-0.021429	-0.018349	
...	...	...	...	...	...	...	
2024-12-20	0.000950	-0.006250	0.003937	0.017857	-0.010676	-0.002049	-
2024-12-23	0.016120	0.018868	-0.003922	-0.008772	0.057554	0.016427	
2024-12-24	-0.004326	-0.018519	-0.011811	-0.008850	-0.054422	-0.014141	-
2024-12-27	-0.004129	0.000000	0.011952	0.000000	0.007194	0.010246	
2024-12-30	0.006158	-0.006289	-0.043307	0.000000	0.017857	-0.006085	-

475 rows × 34 columns

Single Index Model

```
# Beta
def beta(data):
  betas = {}
  for saham in data.columns:
    if saham == Ticker_market:
      betas[saham] = 1
    else:
      betas[saham] = cov_saham[saham] / var_saham[Ticker_market]
  return pd.Series(betas)
beta_saham = beta(data_return).iloc[1:]

# Alpha
def alpha(data):
  alpha_saham = expected_returns(data) - beta(data) * ER_saham[Ticker_r
  return alpha_saham
alpha_saham = alpha(data_return).iloc[1:]

# Residual Variance
def residual_variance(data):
    residual_variance = var(data) - (beta(data)* beta(data) * var_sahar
    return residual_variance
resvar_saham = residual_variance(data_return).iloc[1:]

pd.DataFrame({'Beta': beta_saham, 'Alpha': alpha_saham, 'Residual Varia
```

```
# Excess Return to Beta
def ERB(data):
  ERB_saham = (expected_returns(data) - risk_free) / beta(data)
  return ERB_saham
ERB_saham = ERB(data_return).iloc[1:]
ERB_saham

# A
def nilai_A(data):
  nilai_A = ((expected_returns(data) - risk_free) * beta(data)) / resi
  return nilai_A
A_saham = nilai_A(data_return).iloc[1:]
A_saham

# B
def nilai_B(data):
  nilai_B = beta(data)**2 / residual_variance(data)
  return nilai_B
B_saham = nilai_B(data_return).iloc[1:]
B_saham

# Cutoff
def Cutoff(data):
  Cutoff_value = (var_saham[Ticker_market]*nilai_A(data))/(1+(var_saha
  return Cutoff_value
Cutoff_saham = Cutoff(data_return).iloc[1:]
Cutoff_saham


pd.DataFrame({'ERB': ERB_saham, 'Nilai A': A_saham, 'Nilai B': B_saham,
```

```
#Proportion
Max_cutoff = max(Cutoff_saham)
def proporsi(data):
  nilai_proporsi = ((ERB(data) - Max_cutoff) * (beta(data) / residual_v
  return nilai_proporsi
proporsi_saham = proporsi(data_return).iloc[1:]
proporsi_saham
```

```
#Pemilihan Kandidat Portofolio

def kandidat(ERB_saham, Max_cutoff, proporsi_saham):
  pemilihan_kandidat = []
  for saham in ERB_saham.index:
    if ERB_saham[saham] > Max_cutoff and proporsi_saham[saham] > 0:
      pemilihan_kandidat.append(saham)
  return pemilihan_kandidat

pemilihan_kandidat = kandidat(ERB_saham, Max_cutoff, proporsi_saham)
pemilihan_kandidat
```

```
['ACES.JK',
 'ADRO.JK',
 'BBCA.JK',
 'BMRI.JK',
 'BRIS.JK',
 'BRMS.JK',
 'BRPT.JK',
 'ERAA.JK',
 'EXCL.JK',
 'ICBP.JK',
 'INDF.JK',
 'ISAT.JK',
 'JPFA.JK',
 'JSMR.JK',
 'MAPI.JK',
 'MEDC.JK',
 'PGAS.JK',
 'PTBA.JK',
 'UNTR.JK']
```

Pembentukan Kombinasi Saham

```
import itertools

def kombinasi(pemilihan_kandidat, max_stocks=len(pemilihan_kandidat)):
  kombinasi = []
  for i in range(1, len(pemilihan_kandidat) + 1):
      for combo in itertools.combinations(pemilihan_kandidat, i):
          kombinasi.append(combo)
  return kombinasi

kombinasi_saham = kombinasi(pemilihan_kandidat)
kombinasi_saham
```

```
[('ACES.JK',),
 ('ADRO.JK',),
 ('BBCA.JK',),
 ('BMRI.JK',),
 ('BRIS.JK',),
 ('BRMS.JK',),
 ('BRPT.JK',),
 ('ERAA.JK',),
```

```
('ERAA.JK',),
('EXCL.JK',),
('ICBP.JK',),
('INDF.JK',),
('ISAT.JK',),
('JPFA.JK',),
('JSMR.JK',),
('MAPI.JK',),
('MEDC.JK',),
('PGAS.JK',),
('PTBA.JK',),
('UNTR.JK',),
('ACES.JK', 'ADRO.JK'),
('ACES.JK', 'BBCA.JK'),
('ACES.JK', 'BMRI.JK'),
('ACES.JK', 'BRIS.JK'),
('ACES.JK', 'BRMS.JK'),
('ACES.JK', 'BRPT.JK'),
('ACES.JK', 'ERAA.JK'),
('ACES.JK', 'EXCL.JK'),
('ACES.JK', 'ICBP.JK'),
('ACES.JK', 'INDF.JK'),
('ACES.JK', 'ISAT.JK'),
('ACES.JK', 'JPFA.JK'),
('ACES.JK', 'JSMR.JK'),
('ACES.JK', 'MAPI.JK'),
('ACES.JK', 'MEDC.JK'),
('ACES.JK', 'PGAS.JK'),
('ACES.JK', 'PTBA.JK'),
('ACES.JK', 'UNTR.JK'),
('ADRO.JK', 'BBCA.JK'),
('ADRO.JK', 'BMRI.JK'),
('ADRO.JK', 'BRIS.JK'),
('ADRO.JK', 'BRMS.JK'),
('ADRO.JK', 'BRPT.JK'),
('ADRO.JK', 'ERAA.JK'),
('ADRO.JK', 'EXCL.JK'),
('ADRO.JK', 'ICBP.JK'),
('ADRO.JK', 'INDF.JK'),
('ADRO.JK', 'ISAT.JK'),
('ADRO.JK', 'JPFA.JK'),
('ADRO.JK', 'JSMR.JK'),
('ADRO.JK', 'MAPI.JK'),
('ADRO.JK', 'MEDC.JK'),
('ADRO.JK', 'PGAS.JK'),
('ADRO.JK', 'PTBA.JK'),
('ADRO.JK', 'UNTR.JK'),
('BBCA.JK', 'BMRI.JK'),
('BBCA.JK', 'BRIS.JK'),
('BBCA.JK', 'BRMS.JK'),
('BBCA.JK', 'BRPT.JK'),
```

```
def banyak_kombinasi(kombinasi):
  banyaknya_kombinasi = {}
  for N in kombinasi:
    length = len(N)
    banyaknya_kombinasi[length] = banyaknya_kombinasi.get(length, 0) +
  return banyaknya_kombinasi
```

```
combination_counts = banyak_kombinasi(kombinasi_saham)
combination_counts
```

```
{1: 19,
 2: 171,
 3: 969,
 4: 3876,
 5: 11628,
 6: 27132,
 7: 50388,
 8: 75582,
 9: 92378,
 10: 92378,
 11: 75582,
 12: 50388,
 13: 27132,
 14: 11628,
 15: 3876,
 16: 969,
 17: 171,
 18: 19,
 19: 1}
```

```
sum(combination_counts.values())
```

```
524287
```

Portofolio

```
# Penghitungan Bobot
import pandas as pd

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

Bobot_Portofolio = Bobot_SIM(kombinasi_saham, proporsi_saham)
Bobot_Portofolio
```

```
{('ACES.JK',): {'ACES.JK': 1.0},
 ('ADRO.JK',): {'ADRO.JK': 1.0},
 ('BBCA.JK',): {'BBCA.JK': 1.0},
 ('BMRI.JK',): {'BMRI.JK': 1.0},
 ('BRIS.JK',): {'BRIS.JK': 1.0},
 ('BRMS.JK',): {'BRMS.JK': 1.0},
 ('BRPT.JK',): {'BRPT.JK': 1.0},
 ('ERAA.JK',): {'ERAA.JK': 1.0},
 ('EXCL.JK',): {'EXCL.JK': 1.0},
 ('ICBP.JK',): {'ICBP.JK': 1.0},
```

```
 ('INDF.JK',): {'INDF.JK': 1.0},
 ('ISAT.JK',): {'ISAT.JK': 1.0},
 ('JPFA.JK',): {'JPFA.JK': 1.0},
 ('JSMR.JK',): {'JSMR.JK': 1.0},
 ('MAPI.JK',): {'MAPI.JK': 1.0},
 ('MEDC.JK',): {'MEDC.JK': 1.0},
 ('PGAS.JK',): {'PGAS.JK': 1.0},
 ('PTBA.JK',): {'PTBA.JK': 1.0},
 ('UNTR.JK',): {'UNTR.JK': 1.0},
 ('ACES.JK', 'ADRO.JK'): {'ACES.JK': 0.6321256602221618,
  'ADRO.JK': 0.3678743397778383},
 ('ACES.JK', 'BBCA.JK'): {'ACES.JK': 0.8297368436049756,
  'BBCA.JK': 0.1702631563950244},
 ('ACES.JK', 'BMRI.JK'): {'ACES.JK': 0.6579735076525173,
  'BMRI.JK': 0.34202649234748267},
 ('ACES.JK', 'BRIS.JK'): {'ACES.JK': 0.38102173934215405,
  'BRIS.JK': 0.618978260657846},
 ('ACES.JK', 'BRMS.JK'): {'ACES.JK': 0.5192858013044964,
  'BRMS.JK': 0.48071419869550364},
 ('ACES.JK', 'BRPT.JK'): {'ACES.JK': 0.8103840913932429,
  'BRPT.JK': 0.1896159086067571},
 ('ACES.JK', 'ERAA.JK'): {'ACES.JK': 0.8616596677769065,
  'ERAA.JK': 0.1383403322230935},
 ('ACES.JK', 'EXCL.JK'): {'ACES.JK': 0.9580958251593821,
  'EXCL.JK': 0.041904174840617966},
 ('ACES.JK', 'ICBP.JK'): {'ACES.JK': 0.7801665180055087,
  'ICBP.JK': 0.21983348199449132},
 ('ACES.JK', 'INDF.JK'): {'ACES.JK': 0.5484531146383742,
  'INDF.JK': 0.45154688536162574},
 ('ACES.JK', 'ISAT.JK'): {'ACES.JK': 0.37788860778408306,
  'ISAT.JK': 0.6221113922159168},
 ('ACES.JK', 'JPFA.JK'): {'ACES.JK': 0.4536724573299998,
  'JPFA.JK': 0.5463275426700003},
 ('ACES.JK', 'JSMR.JK'): {'ACES.JK': 0.49690300749172905,
  'JSMR.JK': 0.503096992508271},
 ('ACES.JK', 'MAPI.JK'): {'ACES.JK': 0.9895347754043577,
  'MAPI.JK': 0.010465224595642272},
 ('ACES.JK', 'MEDC.JK'): {'ACES.JK': 0.8754582138920418,
  'MEDC.JK': 0.12454178610795816},
 ('ACES.JK', 'PGAS.JK'): {'ACES.JK': 0.9747344420632007,
  'PGAS.JK': 0.025265557936799212},
 ('ACES.JK', 'PTBA.JK'): {'ACES.JK': 0.7385721186840285,
  'PTBA.JK': 0.26142788131597144},
 ('ACES.JK', 'UNTR.JK'): {'ACES.JK': 0.49004730547393993,
  'UNTR.JK': 0.5099526945260601},
 ('ADRO.JK', 'BBCA.JK'): {'ADRO.JK': 0.7393160183244291,
  'BBCA.JK': 0.2606839816755709},
 ('ADRO.JK', 'BMRI.JK'): {'ADRO.JK': 0.5282024538299569,
```

```
# Penghitungan Expected Return
def ER_Portofolio(bobot, kombinasi_saham, alpha_saham, beta_saham):
  ER_Portofolio = {}
  for kombinasi, bobot in bobot.items():
    er_port = 0
    for saham, bobot in bobot.items():
      er_port += (bobot * alpha_saham[saham]) + (bobot * beta_saham[sal
    ER_Portofolio[kombinasi] = er_port
  return ER_Portofolio
```

```
expected_return_Portofolio = ER_Portofolio(Bobot_Portofolio, kombinasi_
pd.DataFrame.from_dict(expected_return_Portofolio, orient = 'index')
```

```
# Penghitungan Risiko
def Risk_Portofolio(bobot, kombinasi_saham, beta_saham):
  Risiko_Port = {}
  for kombinasi_saham, bobot in bobot.items():
      beta_port = 0
      resvar_port = 0
      for saham, bobot in bobot.items():
          beta_port += (bobot * beta_saham[saham])
          resvar_port += (bobot**2 * resvar_saham[saham])
      Risiko_Port[kombinasi_saham] = (beta_port**2) * var_saham[Ticker_
  return Risiko_Port

Risiko_Portofolio = Risk_Portofolio(Bobot_Portofolio, kombinasi_saham,
pd.DataFrame.from_dict(Risiko_Portofolio, orient = 'index')
```

```
import math
# Sharpe Index
def Indeks_Sharpe(ER_Portofolio, Risiko_portofolio):
  Indeks_Sharpe = {}
  for kombinasi, ER in ER_Portofolio.items():
    sharpe = (ER - risk_free) / math.sqrt(Risiko_portofolio[kombinasi])
    Indeks_Sharpe[kombinasi] = sharpe
  return Indeks_Sharpe

Sharpe_Portofolio = Indeks_Sharpe(expected_return_Portofolio, Risiko_P
pd.DataFrame.from_dict(Sharpe_Portofolio, orient = 'index')
```

Kesimpulan

```
def kesimpulan(expected_return_Portofolio, Risiko_Portofolio, Sharpe_Po
    lengths = range(1, len(pemilihan_kandidat) + 1)
    average_values = {}

    for length in lengths:
        Portofolio_subset = [
            key for key in expected_return_Portofolio.keys() if len(key
        ]
        if Portofolio_subset:
            average_expected_return = sum(
                expected_return_Portofolio[Portofolio] for Portofolio i
            ) / len(Portofolio_subset)

            average_risk = sum(
                Risiko_Portofolio[Portofolio] for Portofolio in Portofo
            ) / len(Portofolio_subset)

            average_sharpe = sum(
                Sharpe_Portofolio[Portofolio] for Portofolio in Portofo
            ) / len(Portofolio_subset)

            average_values[length] = {
                'expected_return_portofolio': average_expected_return,
                'Risiko_Portofolio': average_risk,
                'Sharpe_Portofolio': average_sharpe,
            }

    return average_values

Hasil_Portofolio = kesimpulan(expected_return_Portofolio, Risiko_Portof

# Create DataFrame
Hasil_Portofolio = pd.DataFrame.from_dict(Hasil_Portofolio, orient='ind
Hasil_Portofolio
```

```
def kombinasi_riskmin(Risiko_Portofolio, expected_return_Portofolio, Sh
  info_min = {}
  for kombinasi, risk in Risiko_Portofolio.items():
      n = len(kombinasi)
      if n not in info_min or risk < info_min[n]['min_risiko']:
          info_min[n] = {
              'kombinasi': kombinasi,
              'min_risiko': risk,
              'expected_return': expected_return_Portofolio[kombinasi],
              'sharpe': Sharpe_Portofolio[kombinasi]
          }
  return info_min

Minimrisk_portofolio = kombinasi_riskmin(Risiko_Portofolio, expected_re
pd.DataFrame.from_dict(Minimrisk_portofolio, orient='index')
```

```
import matplotlib.pyplot as plt

average_risk = Hasil_Portofolio['Risiko_Portofolio'].values
min_risk = [value['min_risiko'] for value in Minimrisk_portofolio.value

plt.figure(figsize=(10, 6))
plt.plot(average_risk, label ='Rata-Rata Risiko', marker ='o', color =
plt.plot(min_risk, label ='Minimum Risiko', marker = 'o', color = 'r')
plt.xlabel('Jumlah saham dalam portofolio')
plt.ylabel('Risiko')
plt.title('Rata-Rata Risiko vs. Minimum Risiko')
plt.legend()
plt.grid(True)
plt.xticks(range(len(average_risk)), range(1, len(average_risk) + 1))
plt.show()
```

```
def kombinasi_ermax(Risiko_Portofolio, expected_return_Portofolio, Shar
  info_max = {}
  for kombinasi, ER in expected_return_Portofolio.items():
      n = len(kombinasi)
      if n not in info_max or ER > info_max[n]['ER Maksimal']:
          info_max[n] = {
              'Kombinasi': kombinasi,
              'ER Maksimal': ER,
              'Risiko Portofolio': Risiko_Portofolio[kombinasi],
              'Sharpe': Sharpe_Portofolio[kombinasi]
          }
  return info_max

ERmax_portofolio = kombinasi_ermax(Risiko_Portofolio, expected_return_P
pd.DataFrame.from_dict(ERmax_portofolio, orient='index')
```

```
import matplotlib.pyplot as plt

average_return = Hasil_Portofolio['expected_return_portofolio'].values
ERmaks = [value['ER Maksimal'] for value in ERmax_portofolio.values()]

plt.figure(figsize=(10, 6))
plt.plot(average_return, label ='Rata-rata Return', marker ='o', color
plt.plot(ERmaks, label ='Return Maksimal', marker = 'o', color = 'r')
plt.xlabel('Jumlah saham dalam portofolio')
plt.ylabel('Return Portofolio')
plt.title('Rata-Rata Return vs Return Maksimal')
plt.legend()
plt.grid(True)
plt.xticks(range(len(average_risk)), range(1, len(average_risk) + 1))
plt.show()
```

```
def kombinasi_sharpemax(Risiko_Portofolio, expected_return_Portofolio, S
  info_maxs = {}
  for kombinasi, Sharpe in Sharpe_Portofolio.items():
      n = len(kombinasi)
      if n not in info_maxs or Sharpe > info_maxs[n]['Sharpe Maksimal']:
          info_maxs[n] = {
              'Kombinasi': kombinasi,
              'Sharpe Maksimal': Sharpe,
              'Expected Return': expected_return_Portofolio[kombinasi],
              'Risiko Portofolio': Risiko_Portofolio[kombinasi]
          }
  return info_maxs

Sharpemax_portofolio = kombinasi_sharpemax(Risiko_Portofolio, expected_r
pd.DataFrame.from_dict(Sharpemax_portofolio, orient='index')
```

```
import matplotlib.pyplot as plt

average_sharpe = Hasil_Portofolio['Sharpe_Portofolio'].values
Sharpemaks = [value['Sharpe Maksimal'] for value in Sharpemax_portofol

plt.figure(figsize=(10, 6))
plt.plot(average_sharpe, label ='Rata-rata Sharpe', marker ='o', color
plt.plot(Sharpemaks, label ='Sharpe Maksimal', marker = 'o', color = '
plt.xlabel('Jumlah saham dalam portofolio')
plt.ylabel('Sharpe Portofolio')
plt.title('Rata-Rata Sharpe vs Sharpe Maksimal')
plt.legend()
plt.grid(True)
plt.xticks(range(len(average_risk)), range(1, len(average_risk) + 1))
plt.savefig('sharpe_comparison.png')
plt.show()
```