In [5]:
```
import yfinance as yf
import pandas as pd

# Define stock symbols
symbols = ["AAPL", "MSFT", "AMD", "CSCO", "QCOM"]

# Download historical data
data = yf.download(symbols, start="2010-01-01", end="2024-10-31")['Adj Clo:

# Calculate daily returns
returns = data.pct_change().dropna()
```

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

In [9]:
```
import numpy as np

# Calculate mean daily returns and covariance of returns
mean_returns = returns.mean()
cov_matrix = returns.cov()

# Define the target return (e.g., 0.001 for 0.1% daily return)
target_return = 0.001
```

In [10]:
```
from scipy.optimize import minimize

# Define the portfolio variance (objective function)
def portfolio_variance(weights, cov_matrix):
    return np.dot(weights.T, np.dot(cov_matrix, weights))

# Define the constraint for target return
def target_return_constraint(weights, mean_returns, target_return):
    return np.dot(weights, mean_returns) - target_return

# Define the constraint for sum of weights to equal 1
def weight_constraint(weights):
    return np.sum(weights) - 1

# Initial guess for weights (equal allocation)
num_assets = len(symbols)
initial_weights = np.ones(num_assets) / num_assets

# Set constraints and bounds
constraints = [
    {'type': 'eq', 'fun': target_return_constraint, 'args': (mean_returns,
    {'type': 'eq', 'fun': weight_constraint}
]
bounds = [(0, 1) for _ in range(num_assets)]

# Perform the optimization
result = minimize(portfolio_variance, initial_weights, args=(cov_matrix,),
                  method='SLSQP', bounds=bounds, constraints=constraints)

# Get the optimal weights
optimal_weights = result.x
print("Optimal Weights:", optimal_weights)
```

```
Optimal Weights: [0.24441185 0.29143629 0.1094986  0.20303126 0.151622  ]
```

In [11]:
```
# Expected portfolio return
portfolio_return = np.dot(optimal_weights, mean_returns)

# Portfolio variance and standard deviation (risk)
portfolio_variance_value = portfolio_variance(optimal_weights, cov_matrix)
portfolio_std_dev = np.sqrt(portfolio_variance_value)

print("Expected Portfolio Return:", portfolio_return)
print("Portfolio Risk (Standard Deviation):", portfolio_std_dev)
```

```
Expected Portfolio Return: 0.0010000000008613225
Portfolio Risk (Standard Deviation): 0.017626181639060608
```

In [ ]: