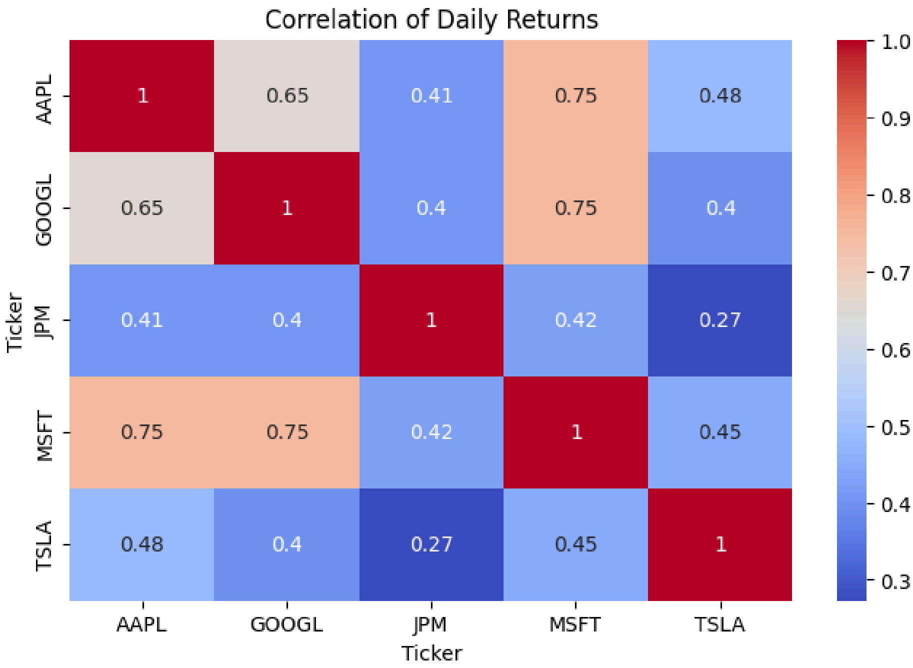
```
# Imports and Data Download
import yfinance as yf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.optimize import minimize
import os


# Tickers and date range
tickers = ['AAPL', 'MSFT', 'TSLA', 'JPM', 'GOOGL']
start_date = '2020-01-01'
end_date = '2024-12-31'


# Download adjusted close prices
data = yf.download(tickers, start=start_date, end=end_date, auto_adjust=False)
data = data['Adj Close']
data.dropna(inplace=True)
# Create 'data' directory
os.makedirs("data", exist_ok=True)
# Save raw data
data.to_csv("data/stock_data.csv")
```

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

```
# Daily Returns
returns = data.pct_change().dropna()


# Summary stats
print(returns.describe())
```

```
Ticker         AAPL        GOOGL          JPM         MSFT         TSLA
count   1256.000000  1256.000000  1256.000000  1256.000000  1256.000000
mean       0.001189     0.001031     0.000745     0.000995     0.003026
std        0.019962     0.020478     0.020498     0.019217     0.042325
min       -0.128647    -0.116341    -0.149649    -0.147390    -0.210628
25%       -0.008426    -0.009504    -0.008412    -0.008248    -0.019882
50%        0.001210     0.001843     0.000686     0.001113     0.001912
75%        0.012017     0.011412     0.009903     0.010947     0.023789
max        0.119808     0.102244     0.180125     0.142169     0.219190
```

```
# Correlation heatmap
plt.figure(figsize=(8, 5))
sns.heatmap(returns.corr(), annot=True, cmap="coolwarm")
plt.title("Correlation of Daily Returns")
plt.show()
```




```
# Portfolio Optimization
mean_returns = returns.mean()
cov_matrix = returns.cov()

def portfolio_perf(weights):
    port_return = np.sum(weights * mean_returns) * 252
    port_vol = np.sqrt(np.dot(weights.T, np.dot(cov_matrix * 252, weights)))
    sharpe = port_return / port_vol
    return port_return, port_vol, sharpe

def negative_sharpe(weights):
    return -portfolio_perf(weights)[2]

def constraint_sum(weights):
    return np.sum(weights) - 1


# Initial guess and bounds
num_assets = len(tickers)
init_guess = [1/num_assets] * num_assets
bounds = [(0, 1)] * num_assets
constraints = ({'type': 'eq', 'fun': constraint_sum})

opt = minimize(negative_sharpe, init_guess, method='SLSQP', bounds=bounds, constraints=constraints)
opt_weights = opt.x.round(4)
print(f"Optimal Weights:\n{dict(zip(tickers, opt_weights))}")
```

```
Optimal Weights:
{'AAPL': np.float64(0.3537), 'MSFT': np.float64(0.1524), 'TSLA': np.float64(0.1137), 'JPM': np.float64(0.0), 'GOOGL': np
```

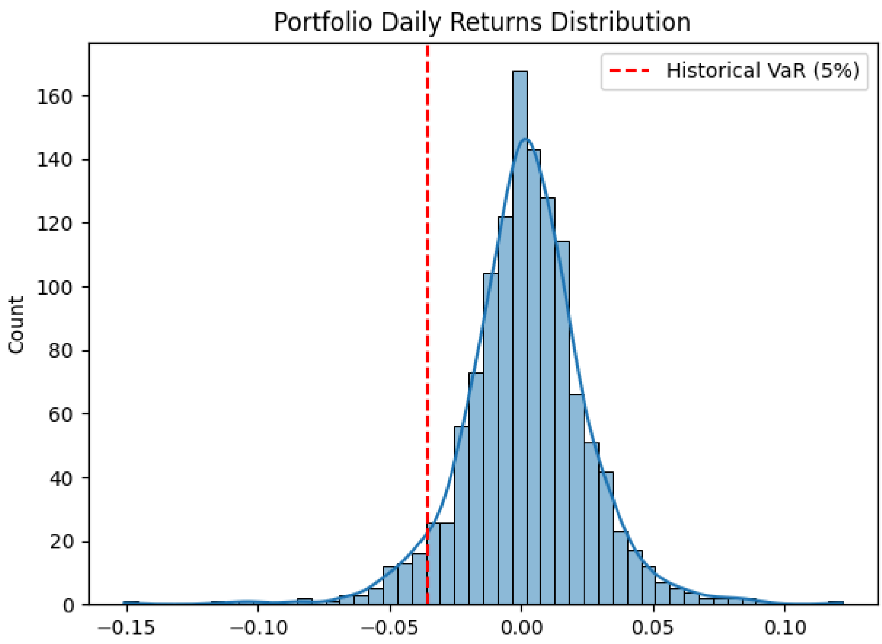
```
# Value at Risk
portfolio_returns = returns.dot(opt_weights)


# Parametric VaR (Gaussian)
conf_level = 0.95
z_score = abs(np.percentile(portfolio_returns, (1 - conf_level) * 100))
parametric_var = np.mean(portfolio_returns) - z_score * np.std(portfolio_returns)
print(f"Parametric VaR (95%): {round(parametric_var * 100, 2)}%")
```

```
Parametric VaR (95%): 0.1%
```

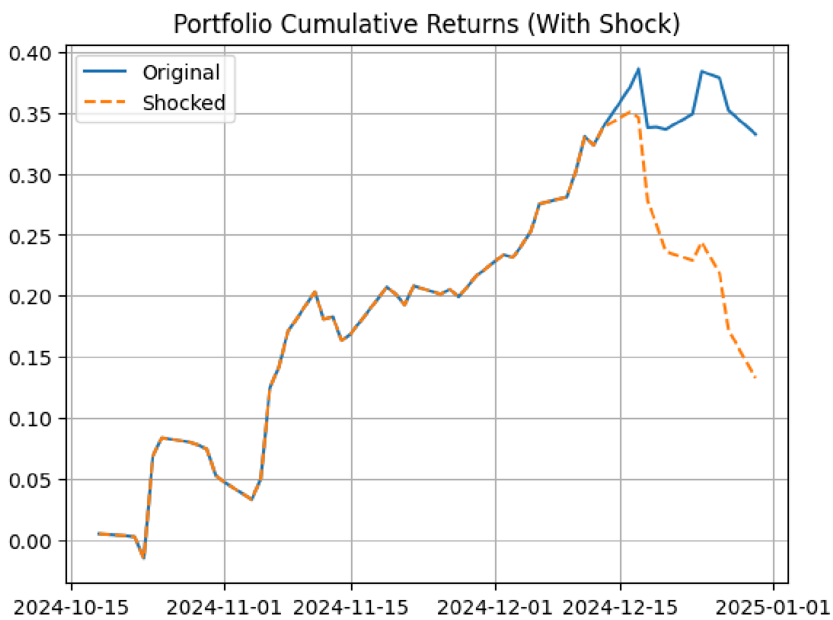
```
# Historical VaR
historical_var = np.percentile(portfolio_returns, 5)
print(f"Historical VaR (5%): {round(historical_var * 100, 2)}%")
```

```
Historical VaR (5%): -3.56%
```

```
# Plot distribution
plt.figure(figsize=(7, 5))
sns.histplot(portfolio_returns, kde=True, bins=50)
plt.axvline(historical_var, color='red', linestyle='--', label='Historical VaR (5%)')
plt.title("Portfolio Daily Returns Distribution")
plt.legend()
plt.show()
```




```
# Simulate Market Shock
shock_returns = portfolio_returns.copy()
shock_returns.iloc[-10:] = shock_returns.iloc[-10:] - 0.02  # simulate 2% daily drop for last 10 days

plt.plot(portfolio_returns[-50:].cumsum(), label='Original')
plt.plot(shock_returns[-50:].cumsum(), label='Shocked', linestyle='--')
plt.title("Portfolio Cumulative Returns (With Shock)")
plt.legend()
plt.grid(True)
plt.show()
```




```
# Efficient Frontier (Monte Carlo Sim)
n_portfolios = 5000
results = np.zeros((3, n_portfolios))
weights_record = []

for i in range(n_portfolios):
```

```
for i in range(n_portfolios):
    weights = np.random.random(num_assets)
    weights /= np.sum(weights)
    weights_record.append(weights)
    port_return, port_vol, sharpe = portfolio_perf(weights)
    results[0,i] = port_return
    results[1,i] = port_vol
    results[2,i] = sharpe


# Plot
plt.scatter(results[1,:], results[0,:], c=results[2,:], cmap='viridis')
plt.colorbar(label='Sharpe Ratio')
opt_ret, opt_vol, _ = portfolio_perf(opt_weights)
plt.scatter(opt_vol, opt_ret, marker='*', color='r', s=200, label='Optimal Portfolio')
plt.xlabel('Volatility (Std Dev)')
plt.ylabel('Expected Return')
plt.title("Efficient Frontier")
plt.legend()
plt.show()
```