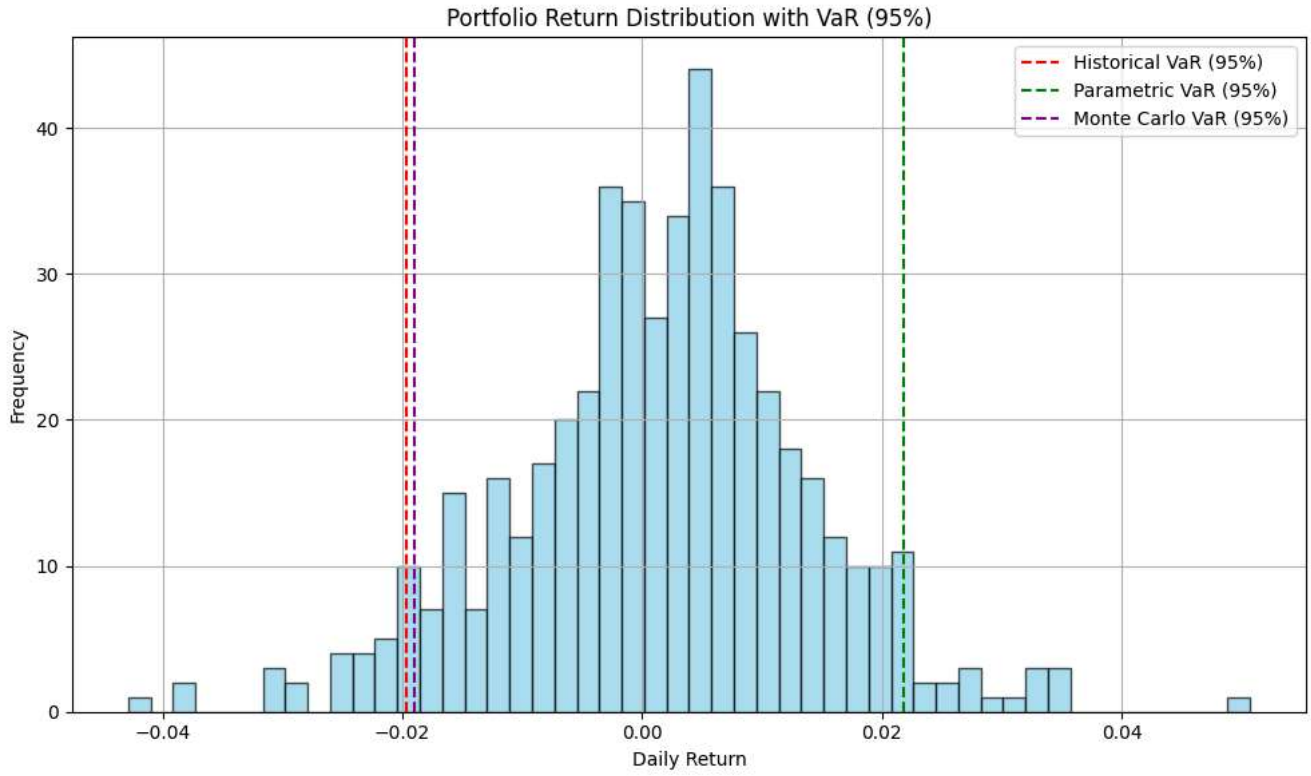
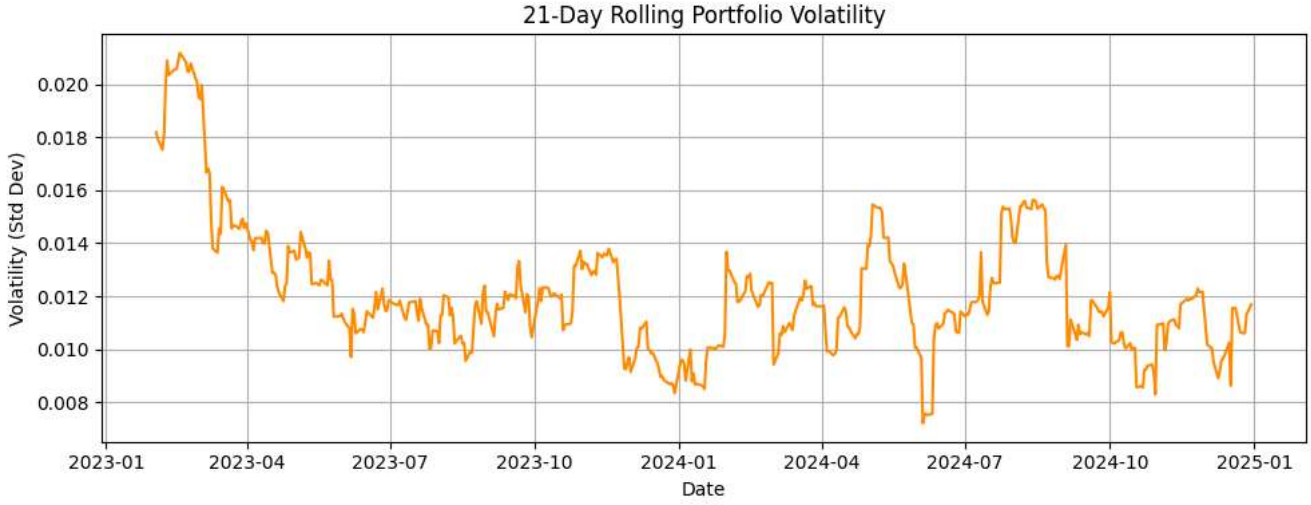
```
import yfinance as yf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm

# Step 1: Download data using 'Close' prices to avoid Adj Close issues
tickers = ['AAPL', 'MSFT', 'GOOG']
data = yf.download(tickers, start='2023-01-01', end='2024-12-31')['Close']

# Step 2: Calculate daily returns
returns = data.pct_change().dropna()

# Step 3: Portfolio weights (must match number of assets)
weights = np.array([0.4, 0.3, 0.3])
if len(weights) != returns.shape[1]:
    raise ValueError("Number of weights must match number of assets.")

# Step 4: Portfolio daily returns
portfolio_returns = returns @ weights

# Step 5: Historical VaR (95%)
confidence_level = 0.95
var_hist = np.percentile(portfolio_returns, (1 - confidence_level) * 100)

# Step 6: Parametric VaR (95%)
mean = portfolio_returns.mean()
std = portfolio_returns.std()
z_score = norm.ppf(confidence_level)
var_parametric = -(mean + z_score * std)

# Step 7: Monte Carlo Simulation VaR (95%)
np.random.seed(42)
simulated_returns = np.random.normal(loc=mean, scale=std, size=10000)
var_monte_carlo = np.percentile(simulated_returns, (1 - confidence_level) * 100)

# Step 8: Plot return distribution with VaR lines
plt.figure(figsize=(10, 6))
plt.hist(portfolio_returns, bins=50, color='skyblue', alpha=0.7, edgecolor='black')
plt.axvline(var_hist, color='red', linestyle='--', label='Historical VaR (95%)')
plt.axvline(-var_parametric, color='green', linestyle='--', label='Parametric VaR (95%)')
plt.axvline(var_monte_carlo, color='purple', linestyle='--', label='Monte Carlo VaR (95%)')
plt.title("Portfolio Return Distribution with VaR (95%)")
plt.xlabel("Daily Return")
plt.ylabel("Frequency")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

# Step 9: Output VaR results
print("=== 1-Day Value-at-Risk (VaR) at 95% Confidence ===")
print(f"Historical VaR      : {var_hist:.2%}")
print(f"Parametric VaR      : {var_parametric:.2%}")
print(f"Monte Carlo VaR     : {var_monte_carlo:.2%}")
```

```
/tmp/ipython-input-1-228737129.py:9: FutureWarning: YF.download() has changed argument auto_adjust default to True
  data = yf.download(tickers, start='2023-01-01', end='2024-12-31')['Close']
[*********************100%***********************]  3 of 3 completed
```



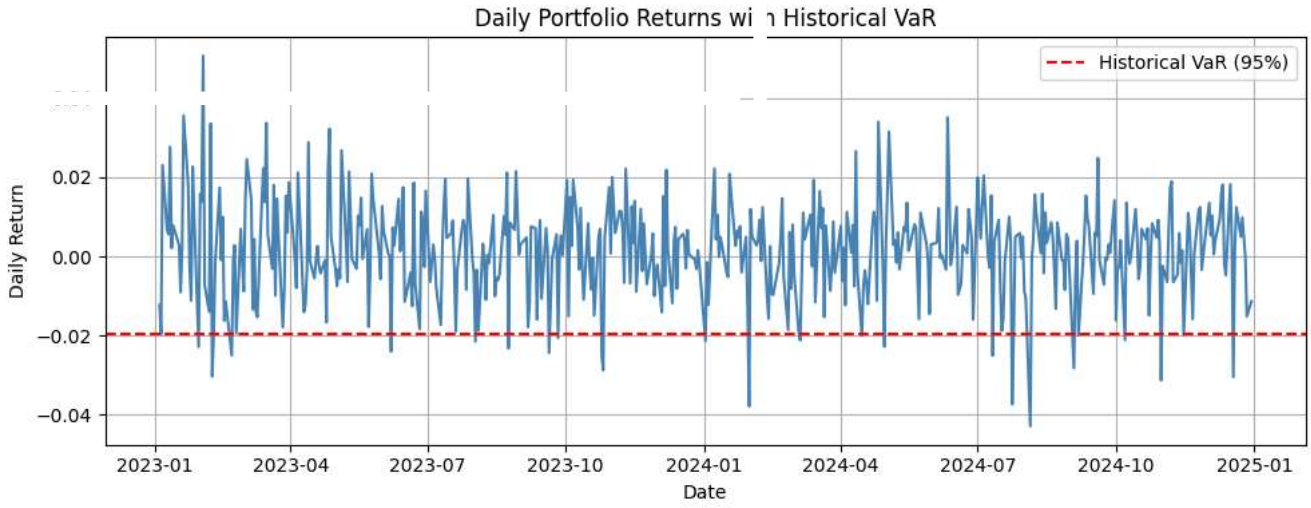

```
=== 1-Day Value-at-Risk (VaR) at 95% Confidence ===
Historical VaR      : -1.97%
Parametric VaR      : -2.19%
Monte Carlo VaR     : -1.90%
```

```
rolling_vol = portfolio_returns.rolling(window=21).std()  # ~1-month rolling volatility

plt.figure(figsize=(10, 4))
plt.plot(rolling_vol, color='darkorange')
plt.title("21-Day Rolling Portfolio Volatility")
plt.xlabel("Date")
plt.ylabel("Volatility (Std Dev)")
plt.grid(True)
plt.tight_layout()
plt.show()
```




```
plt.figure(figsize=(10, 4))
plt.plot(portfolio_returns, color='steelblue')
plt.axhline(var_hist, color='red', linestyle='--', label='Historical VaR (95%)'
plt.title("Daily Portfolio Returns with Historical VaR")
```

```
plt.xlabel("Date")
plt.ylabel("Daily Return")
plt.grid(True)
plt.legend()
plt.tight_layout()
```




```
plt.figure(figsize=(10, 4))
plt.hist(simulated_returns, bins=50, color='orchid', alpha=0.7)
plt.axvline(var_monte_carlo, color='purple', linestyle='--', label='Monte Carlo VaR (95%)')
plt.title("Simulated Return Distribution (Monte Carlo)")
plt.xlabel("Simulated Daily Return")
plt.ylabel("Frequency")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```