
✓ Project Start

Given my background in finance, I was interested in exploring trends in the stock market. I developed 2 questions that I hoped to answer with this project, using specific companies for my analysis. I decided to base my project on the trading activity of “Magnificent Seven” companies. The “Magnificent Seven” companies are

- Apple (NYSE: AAPL)
- Microsoft (NYSE: MSFT)
- Alphabet (NYSE: GOOG)
- Amazon (NYSE: AMZN)
- Nvidia (NYSE: NVDA)
- Meta (NYSE: META)
- Tesla (NYSE: TSLA).

In place of the broader market, I used the Standard & Poor's Depository Receipts ETF also known as SPDR (NYSE: SPY) as a stand-in for the S&P 500.

I first wanted to determine if there was any correlation between the price movements (also known as returns) of these companies, as well as the broader market. I believe that the more technology-based companies, such as Apple or Microsoft may have some correlation with each other, while companies like Nvidia may not. However, I believe each company will have a higher correlation with the market, as these companies represent a large portion of the S&P.

I also wanted to determine if there was any factor that led to a positive return over the period of time chosen. Factors included trading volume, spread between opening and adjusted close, and the spread between the high and low prices. I believe that trading volume and the spread between opening and closing may have a high correlation, but ultimately won't have an effect on positive or negative returns.

```
1 import pandas as pd
2 import yfinance as yf
3 import numpy as np
4 import seaborn as sns
5 import matplotlib.pyplot as plt
6 from datetime import datetime, date
```

✓ Data Download/Pre-processing

My data source for this project was the Yahoo Finance API, which was imported as *yf*. The API takes the ticker for any security (such as a stock, ETF, or index), the start and end dates, and the interval and outputs the opening price, closing price, highest price, lowest price, adjusted closing price, and volume for the security that was input. Options for intervals include daily, weekly, and monthly.

To use the Yahoo Finance API, I began by defining two variables representing the start and end dates for my data collection, utilizing the datetime library. Yahoo Finance's API returns clean data, with null values pre-filtered, simplifying the analysis process. Just to be sure though, I check the number of null values in each column in each dataframe.

Given the distinct requirements of my two guiding questions, I opted to create separate tables for each:

For the first question, I extracted the adjusted closing price for each stock in my list over the specified date range. I then generated a new dataframe calculating the percent change in price for each interval. The initial row, lacking a percent change value, was removed to maintain data integrity. The second question required three distinct data frames to analyze the 3 different factors

For the spread tables, I identified the necessary columns to create the derived data, replicating this process for each company in the tickers list. This approach allowed for a more organized and targeted analysis of the financial data, tailored to address each specific question effectively.

```
1 # Setting the end date into date format
2
3 enddate = date.today()
4 enddate = str(enddate)
5 enddate = datetime.strptime(enddate, '%Y-%m-%d').date()
```

```

1 # Creates date variable for start date. Exception handling implemented for leap year special case
2
3 try:
4     startdate = datetime.strptime(f"{enddate.year - 3}-{enddate.month}-{enddate.day}", '%Y-%m-%d').date()
5 except ValueError:
6     startdate = datetime.strptime(f"{enddate.year - 3}-{enddate.month}-{enddate.day-1}", '%Y-%m-%d').date()

```

```

1 # Creating list of tickers to pull data from API
2
3 tickers = ['AAPL', 'MSFT', 'GOOG', 'AMZN', 'NVDA', 'META', 'TSLA', 'SPY']

```

```

1 # Creating dataframe for question 1
2
3 prices_df = yf.download(tickers, startdate, enddate, interval="1mo")["Adj Close"].round(2)
4
5 if prices_df.empty:
6     raise ValueError("No data available for the specified tickers and dates")
7
8 returns_df = prices_df.pct_change()
9 returns_df.columns = [f"{c} Monthly Returns" for c in returns_df.columns]
10 returns_df = returns_df.dropna()

```

🔄 [*****100%*****] 8 of 8 completed

```

1 returns_df

```



	AAPL Monthly Returns	AMZN Monthly Returns	GOOG Monthly Returns	META Monthly Returns	MSFT Monthly Returns	NVDA Monthly Returns	SPY Monthly Returns	TSLA Monthly Returns	
Date									
2021-11-01 00:00:00+00:00	0.103509	0.039912	-0.039281	0.002759	-0.003093	0.278213	-0.008031	0.027630	
2021-12-01 00:00:00+00:00	0.075778	-0.049216	0.015693	0.036632	0.019175	-0.099939	0.042583	-0.076863	
2022-01-01 00:00:00+00:00	-0.015723	-0.102867	-0.062080	-0.068647	-0.075319	-0.167234	-0.049421	-0.113609	
2022-02-01 00:00:00+00:00	-0.055243	0.026676	-0.005984	-0.326332	-0.039212	-0.004090	-0.029514	-0.070779	
2022-03-01 00:00:00+00:00	0.058780	0.061474	0.035300	0.053660	0.033994	0.118686	0.034383	0.238023	
2022-04-01 00:00:00+00:00	-0.097096	-0.237546	-0.176800	-0.098426	-0.099854	-0.320117	-0.084939	-0.191954	
2022-05-01 00:00:00+00:00	-0.055891	-0.032749	-0.008022	-0.034072	-0.020360	0.007019	0.002261	-0.129199	
2022-06-01 00:00:00+00:00	-0.080114	-0.116463	-0.040963	-0.167254	-0.053142	-0.188204	-0.086423	-0.111889	
2022-07-01 00:00:00+00:00	0.188699	0.270596	0.066453	-0.013373	0.093117	0.198151	0.096820	0.323785	
2022-08-01 00:00:00+00:00	-0.032584	-0.060615	-0.064203	0.024083	-0.068664	-0.168688	-0.040797	-0.072489	
2022-09-01 00:00:00+00:00	-0.119719	-0.108622	-0.119122	-0.167262	-0.107373	-0.196286	-0.096174	-0.037589	
2022-10-01 00:00:00+00:00	0.109518	-0.093451	-0.015431	-0.313373	-0.003276	0.112211	0.085721	-0.142168	
2022-11-01 00:00:00+00:00	-0.034617	-0.057595	0.071693	0.267765	0.099124	0.254451	0.055594	-0.144326	
2022-12-01 00:00:00+00:00	-0.120825	-0.129894	-0.125395	0.018938	-0.057412	-0.136606	-0.061930	-0.367334	
2023-01-01 00:00:00+00:00	0.110550	0.227738	0.125636	0.237873	0.033331	0.336986	0.067763	0.406235	
2023-02-01 00:00:00+00:00	0.021616	-0.086299	-0.095855	0.174320	0.006467	0.188525	-0.025134	0.187565	
2023-03-01 00:00:00+00:00	0.120310	0.096148	0.151754	0.211513	0.158777	0.196552	0.033130	0.008507	
2023-04-01 00:00:00+00:00	0.029032	0.020912	0.040482	0.133933	0.065773	-0.000720	0.019839	-0.207992	
2023-05-01 00:00:00+00:00	0.044607	0.143480	0.140065	0.101544	0.068794	0.363374	0.004625	0.241130	
2023-06-01 00:00:00+00:00	0.095809	0.081108	-0.019501	0.084075	0.039254	0.117927	0.060868	0.283627	
2023-07-01 00:00:00+00:00	0.012817	0.025468	0.100356	0.110164	-0.013579	0.104778	0.036574	0.021622	
2023-08-01 00:00:00+00:00	-0.043701	0.032391	0.031857	-0.071276	-0.024285	0.056305	-0.016258	-0.034962	
2023-09-01 00:00:00+00:00	-0.087432	-0.078907	-0.039997	0.014610	-0.034593	-0.118768	-0.050795	-0.030456	
2023-10-01 00:00:00+00:00	-0.002583	0.046963	-0.049722	0.003541	0.070804	-0.062328	-0.018241	-0.197346	
2023-11-01 00:00:00+00:00	0.112301	0.097678	0.068806	0.085894	0.120683	0.146922	0.091329	0.195379	
2023-12-01 00:00:00+00:00	0.014922	0.040044	0.052324	0.081951	-0.005557	0.058811	0.041444	0.034988	
2024-01-01 00:00:00+00:00	-0.042231	0.021456	0.006189	0.102210	0.057273	0.242375	0.020046	-0.246257	
2024-02-01 00:00:00+00:00	-0.019760	0.138918	-0.014210	0.256292	0.040387	0.285970	0.052184	0.077901	



2024-03-01 00:00:00+00:00	-0.050092	0.020480	0.089286	-0.008227	0.019009	0.142099	0.029492	-0.129235
2024-04-01 00:00:00+00:00	-0.006723	-0.029826	0.081309	-0.114126	-0.074617	-0.043724	-0.037324	0.042608
2024-05-01 00:00:00+00:00	0.128723	0.008229	0.056564	0.085226	0.066275	0.268781	0.050563	-0.028372
2024-06-01 00:00:00+00:00	0.097043	0.095273	0.054400	0.080100	0.078595	0.126904	0.031959	0.111186
2024-07-01 00:00:00+00:00	0.054378	-0.032445	-0.054927	-0.057348	-0.063992	-0.052704	0.015374	0.172781
2024-08-01 00:00:00+00:00	0.031196	-0.045352	-0.046438	0.097875	-0.002898	0.020084	0.023366	-0.077390

Next steps:

[Generate code with returns_df](#)

[View recommended plots](#)

[New interactive sheet](#)

```

1 # Creating dataframes for question 2
2
3 #   Trading Volume
4 t_volume_df = yf.download(tickers, startdate, enddate, interval="1mo")["Volume"].round(2)
5 t_volume_df = t_volume_df.iloc[1:]
6
7 #   High-Low Volatility
8 hl_spreads = pd.DataFrame()
9 for ticker in tickers:
10  if ticker != 'SPY':
11      high_low_df = yf.download(ticker, startdate, enddate, interval="1mo")[["High", "Low"]].round(2)
12      hl_spreads[ticker] = high_low_df["High"]-high_low_df["Low"]
13 hl_spreads = hl_spreads.iloc[1:]
14
15 #   Open-Close Volatility
16 oc_spreads = pd.DataFrame()
17 for ticker in tickers:
18  if ticker != 'SPY':
19      open_close_df = yf.download(ticker, startdate, enddate, interval="1mo")[["Open", "Adj Close"]].round(2)
20      oc_spreads[ticker] = open_close_df["Adj Close"]-open_close_df["Open"]
21 oc_spreads = oc_spreads.iloc[1:]

```

```

→ [*****100%*****] 8 of 8 completed
[*****100%*****] 1 of 1 completed
[*****100%*****] 1 of 1 completed
[*****100%*****] 1 of 1 completed
[*****100%*****] 1 of 1 completed
[*****100%*****] 1 of 1 completed
[*****100%*****] 1 of 1 completed
[*****100%*****] 1 of 1 completed
[*****100%*****] 1 of 1 completed
[*****100%*****] 1 of 1 completed
[*****100%*****] 1 of 1 completed
[*****100%*****] 1 of 1 completed
[*****100%*****] 1 of 1 completed
[*****100%*****] 1 of 1 completed
[*****100%*****] 1 of 1 completed
[*****100%*****] 1 of 1 completed
[*****100%*****] 1 of 1 completed
[*****100%*****] 1 of 1 completed

```

Question 1

How do price movements correlate between the different companies?

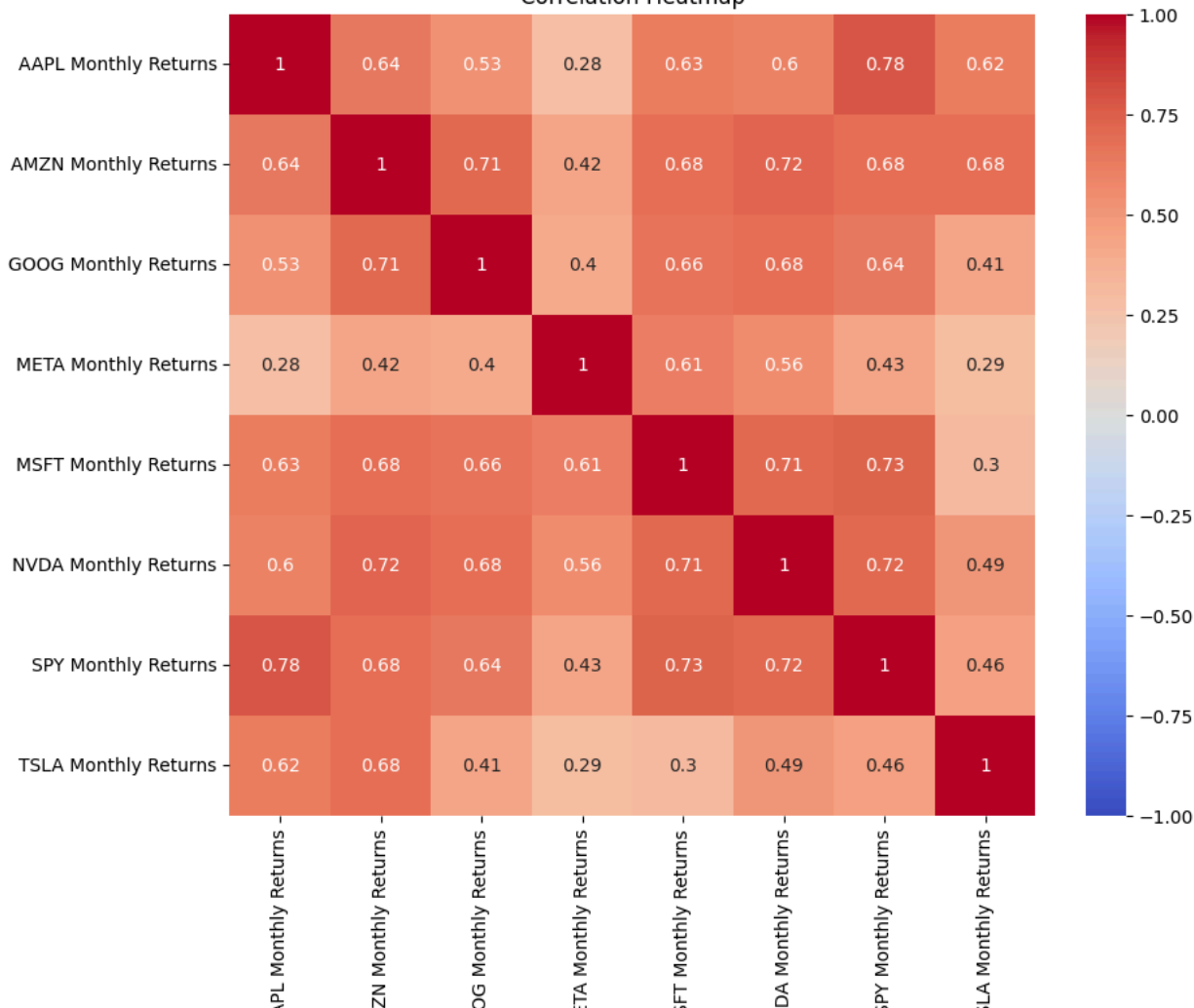
```

1 # Correlation Matrix
2
3 corr_matrix = returns_df.corr()
4 plt.figure(figsize=(10,8))
5 sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', vmin=-1,vmax=1)
6 plt.title('Correlation Heatmap')
7 plt.show()

```



Correlation Heatmap



Story:

This correlation matrix tells an intriguing story about the interplay between major tech stocks and the broader market. Apple (AAPL) emerges as a market leader, showing the strongest correlation (0.78) with the S&P 500 ETF (SPY), suggesting its performance closely mirrors or even influences overall market trends. The tech giants Amazon (AMZN), Google (GOOG), Microsoft (MSFT), and NVIDIA (NVDA) display strong correlations with each other (0.67-0.73), indicating these companies often move in tandem. Interestingly, Meta (META) shows the weakest correlations with other stocks (0.29-0.62), suggesting its performance may be driven by unique factors or company-specific news. Tesla (TSLA) also demonstrates a degree of independence, with lower correlations to most stocks, particularly Meta and Microsoft (0.29-0.30). NVIDIA stands out for its consistently high correlations across the board (0.56-0.73), possibly due to its central role in AI and chip manufacturing affecting various tech sectors. The S&P 500 ETF's strong correlations with most tech stocks (0.65-0.78) highlight the influential role these companies play in driving overall market performance. This matrix paints a picture of a tech-driven market where most major players move together, with a few exceptions that may offer diversification benefits, while also underscoring the outsized influence of companies like Apple on broader market indices.

I was surprised that the correlation for some companies with the market was actually lower than 0.5, as these are all companies that tend to dominate the stock market. It could be that because I am taking data over a longer period of time, the data is normalizing the spikes that indicate fad stocks.

Question 2


What factors can lead to greater returns?

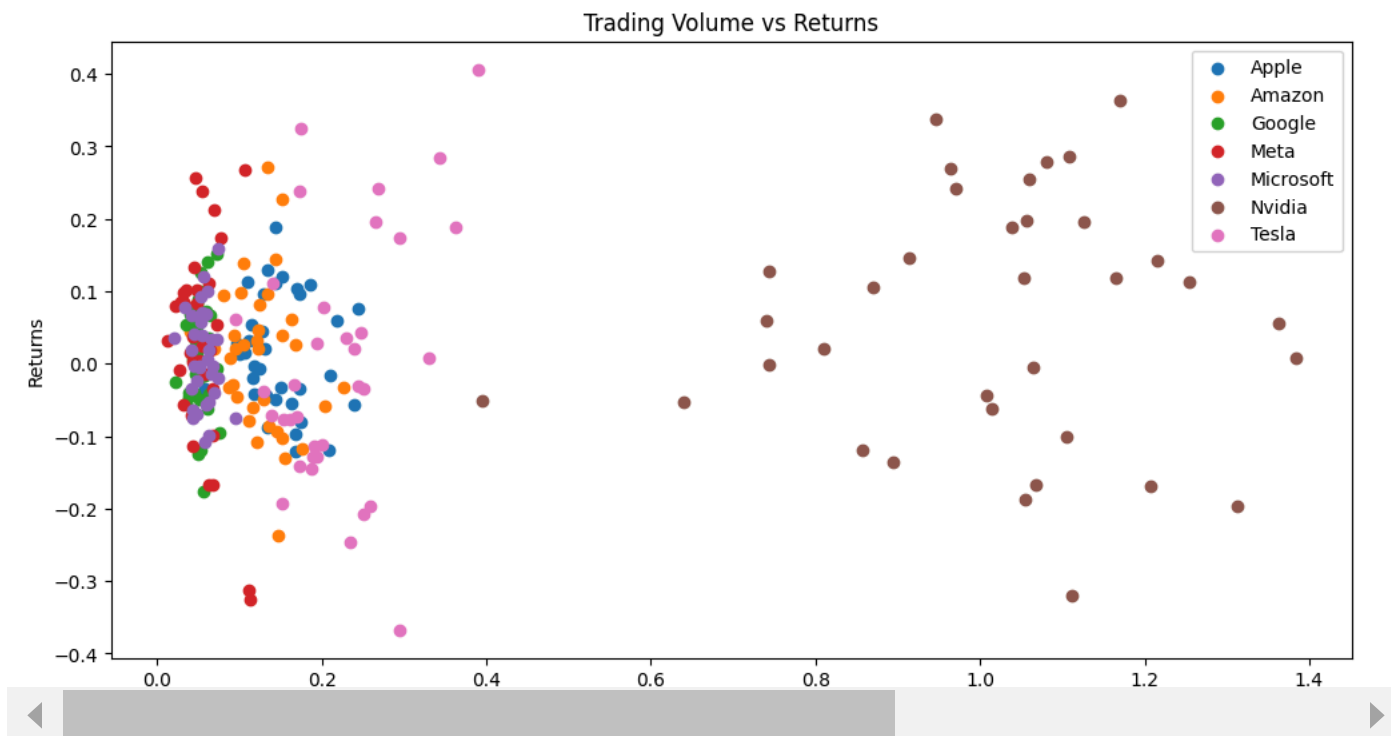
```
1 # Scatterplot - Trading Volume vs Returns
2
```

```

3 plt.figure(figsize=(12, 6))
4
5 plt.scatter(t_volume_df['AAPL'], returns_df['AAPL Monthly Returns'], label='Apple')
6 plt.scatter(t_volume_df['AMZN'], returns_df['AMZN Monthly Returns'], label='Amazon')
7 plt.scatter(t_volume_df['GOOG'], returns_df['GOOG Monthly Returns'], label='Google')
8 plt.scatter(t_volume_df['META'], returns_df['META Monthly Returns'], label='Meta')
9 plt.scatter(t_volume_df['MSFT'], returns_df['MSFT Monthly Returns'], label='Microsoft')
10 plt.scatter(t_volume_df['NVDA'], returns_df['NVDA Monthly Returns'], label='Nvidia')
11 plt.scatter(t_volume_df['TSLA'], returns_df['TSLA Monthly Returns'], label='Tesla')
12
13 plt.title('Trading Volume vs Returns')
14 plt.xlabel('Trading Volume')
15 plt.ylabel('Returns')
16 plt.legend()

```


 <matplotlib.legend.Legend at 0x78b7e64bdea0>



```

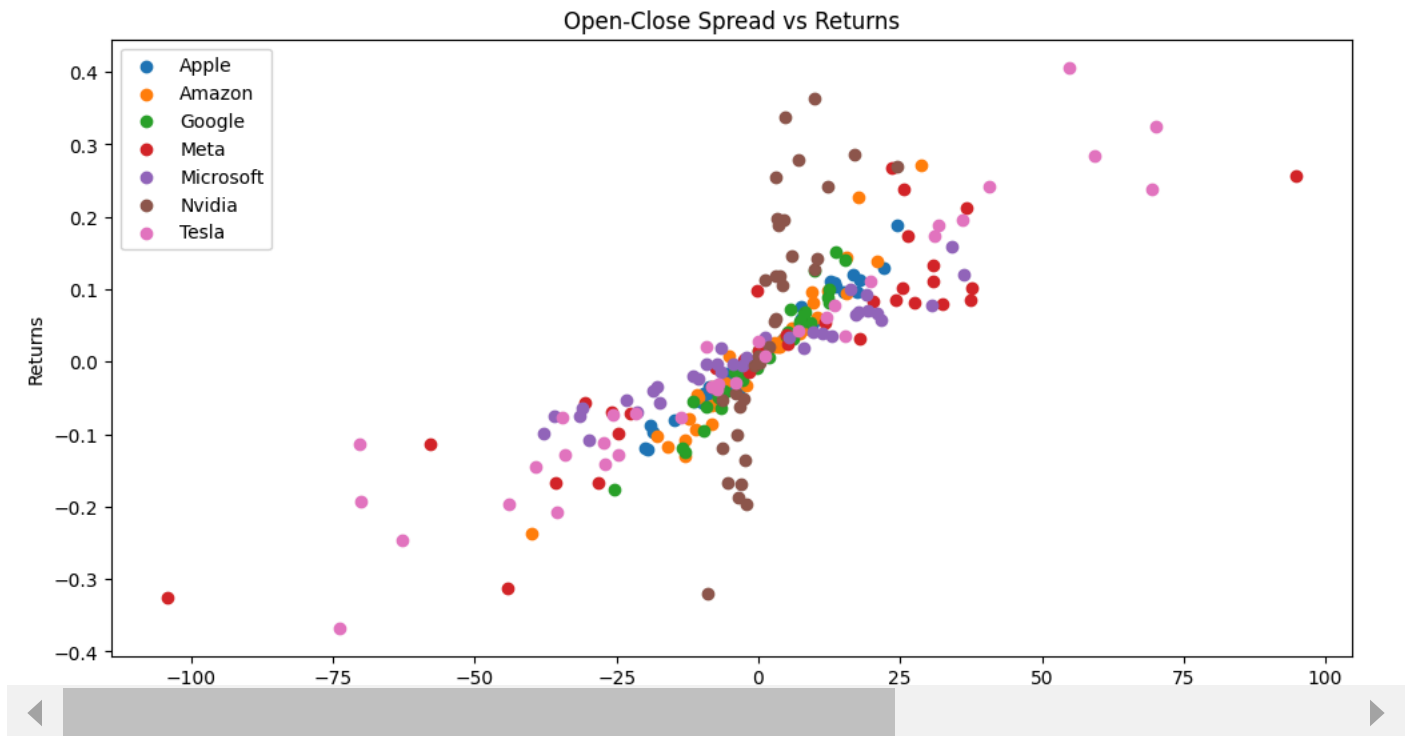
1 # Scatterplot - High-Low Spread vs Returns
2
3 plt.figure(figsize=(12, 6))
4
5 plt.scatter(hl_spreads['AAPL'], returns_df['AAPL Monthly Returns'], label='Apple')
6 plt.scatter(hl_spreads['AMZN'], returns_df['AMZN Monthly Returns'], label='Amazon')
7 plt.scatter(hl_spreads['GOOG'], returns_df['GOOG Monthly Returns'], label='Google')
8 plt.scatter(hl_spreads['META'], returns_df['META Monthly Returns'], label='Meta')
9 plt.scatter(hl_spreads['MSFT'], returns_df['MSFT Monthly Returns'], label='Microsoft')
10 plt.scatter(hl_spreads['NVDA'], returns_df['NVDA Monthly Returns'], label='Nvidia')
11 plt.scatter(hl_spreads['TSLA'], returns_df['TSLA Monthly Returns'], label='Tesla')
12
13 plt.title('High-Low Spread vs Returns')
14 plt.xlabel('High-Low Spread')
15 plt.ylabel('Returns')
16 plt.legend()

```

 <matplotlib.legend.Legend at 0x78b7e6794a60>



```
1 # Scatterplot - Open-Close Spread vs Returns
2
3 plt.figure(figsize=(12, 6))
4
5 plt.scatter(oc_spreads['AAPL'], returns_df['AAPL Monthly Returns'], label='Apple')
6 plt.scatter(oc_spreads['AMZN'], returns_df['AMZN Monthly Returns'], label='Amazon')
7 plt.scatter(oc_spreads['GOOG'], returns_df['GOOG Monthly Returns'], label='Google')
8 plt.scatter(oc_spreads['META'], returns_df['META Monthly Returns'], label='Meta')
9 plt.scatter(oc_spreads['MSFT'], returns_df['MSFT Monthly Returns'], label='Microsoft')
10 plt.scatter(oc_spreads['NVDA'], returns_df['NVDA Monthly Returns'], label='Nvidia')
11 plt.scatter(oc_spreads['TSLA'], returns_df['TSLA Monthly Returns'], label='Tesla')
12
13 plt.title('Open-Close Spread vs Returns')
14 plt.xlabel('Open-Close Spread')
15 plt.ylabel('Returns')
16 plt.legend()
```



Story:

These three scatter plots offer fascinating insights into the trading dynamics of major tech stocks:

1. **Trading Volume vs Returns:** This plot reveals distinct patterns for different companies. Most stocks cluster at lower trading volumes, suggesting typical daily activity. However, Nvidia stands out with consistently higher trading volumes, indicating heightened investor interest or market activity. Tesla shows the widest range of returns, from significant losses to substantial gains, reflecting its volatility. Other tech giants like Apple, Amazon, Google, Meta, and Microsoft display more concentrated patterns, implying relatively stable trading behavior. However, we can see there is no apparent pattern between trading volume and returns, indicating a very weak relationship.
2. **High-Low Spread vs Returns:** This graph illustrates price volatility within trading days. Tesla again shows the widest spread, with some days seeing price swings of over 100 points, highlighting its day-to-day volatility. Nvidia also demonstrates considerable intraday volatility. In contrast, companies like Apple, Google, and Microsoft exhibit tighter spreads, suggesting more stable intraday pricing. Interestingly, there's no clear correlation between the size of the spread and returns, indicating that large price swings don't necessarily translate to higher returns.
3. **Open-Close Spread vs Returns:** This plot reveals a strong positive correlation between open-close spread and returns across all stocks. When stocks close higher than they open (positive spread), they tend to show positive returns, and vice versa. This relationship is particularly pronounced for Tesla and Nvidia, which show the widest spreads in both directions. The tight, diagonal clustering for all stocks suggests that daily price movements are a good indicator of overall returns.

Overall, these plots paint a picture of a tech stock market where companies like Tesla and Nvidia experience high volatility and trading activity, potentially offering greater opportunities for short-term gains but also higher risk. In contrast, established giants like Apple, Google, and Microsoft show more stable patterns, possibly appealing to investors seeking steadier performance. The strong correlation between daily price movements and returns across all stocks underscores the importance of intraday price action in determining overall stock performance.

```
1 # Scatterplot - Trading Volume vs High-Low Spreads
2
3 plt.figure(figsize=(12, 6))
4
5 plt.scatter(t_volume_df['AAPL'], hl_spreads['AAPL'], label='Apple')
6 plt.scatter(t_volume_df['AMZN'], hl_spreads['AMZN'], label='Amazon')
7 plt.scatter(t_volume_df['GOOG'], hl_spreads['GOOG'], label='Google')
8 plt.scatter(t_volume_df['META'], hl_spreads['META'], label='Meta')
9 plt.scatter(t_volume_df['MSFT'], hl_spreads['MSFT'], label='Microsoft')
10 plt.scatter(t_volume_df['NVDA'], hl_spreads['NVDA'], label='Nvidia')
11 plt.scatter(t_volume_df['TSLA'], hl_spreads['TSLA'], label='Tesla')
12
13 plt.title('Trading Volume vs High-Low Spreads')
14 plt.xlabel('Trading Volume')
```

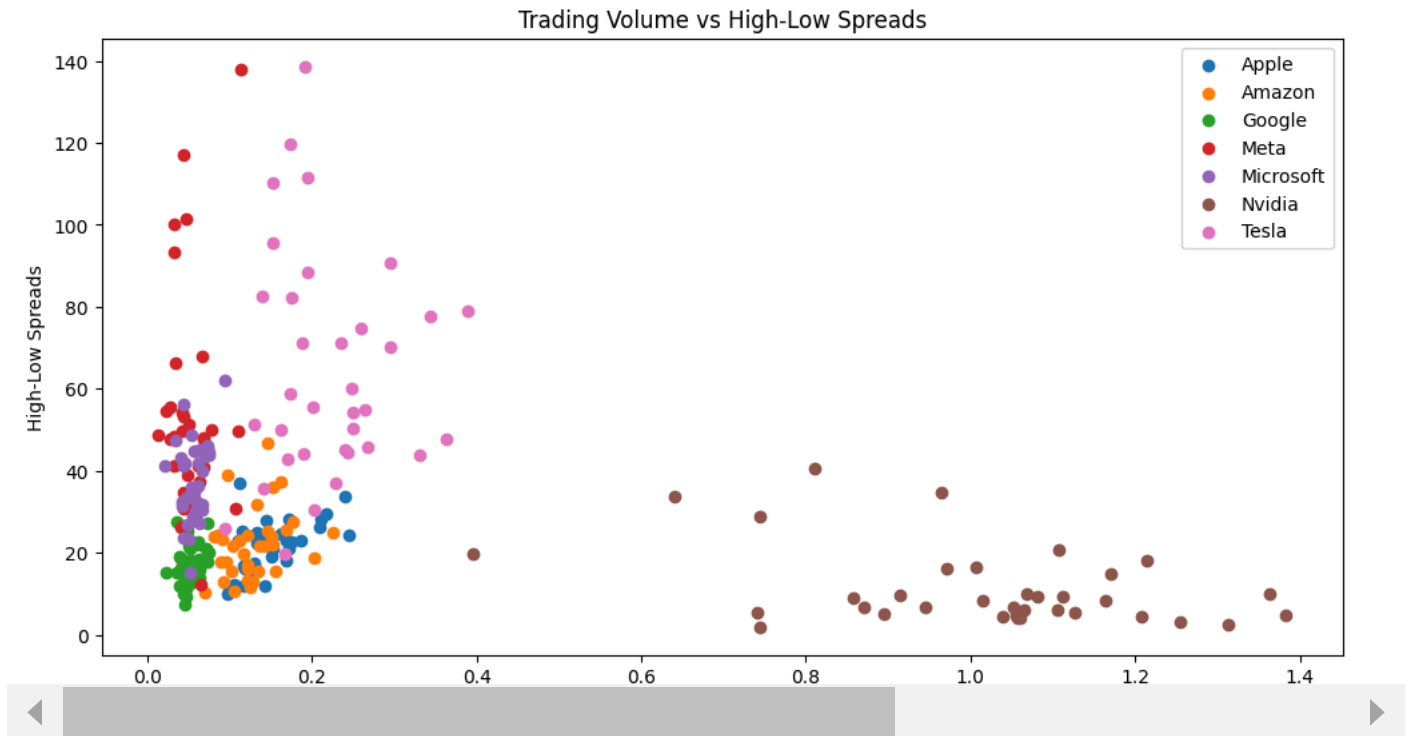


```

15 plt.ylabel('High-Low Spreads')
16 plt.legend()

```

 <matplotlib.legend.Legend at 0x78b7e64be590>



```

1 # Scatterplot - Trading Volume vs Open-Close Spreads
2
3 plt.figure(figsize=(12, 6))
4
5 plt.scatter(t_volume_df['AAPL'], oc_spreads['AAPL'], label='Apple')
6 plt.scatter(t_volume_df['AMZN'], oc_spreads['AMZN'], label='Amazon')
7 plt.scatter(t_volume_df['GOOG'], oc_spreads['GOOG'], label='Google')
8 plt.scatter(t_volume_df['META'], oc_spreads['META'], label='Meta')
9 plt.scatter(t_volume_df['MSFT'], oc_spreads['MSFT'], label='Microsoft')
10 plt.scatter(t_volume_df['NVDA'], oc_spreads['NVDA'], label='Nvidia')
11 plt.scatter(t_volume_df['TSLA'], oc_spreads['TSLA'], label='Tesla')
12
13 plt.title('Trading Volume vs Open-Close Spreads')
14 plt.xlabel('Trading Volume')
15 plt.ylabel('Open-Close Spreads')
16 plt.legend()

```