

# Stock Price Analysis

Stock Price Analysis of Major Tech Companies Over Five Years

## Data Analyst Project

by

Rohit Rai

(I20MA051)



Department of Mathematics  
Sardar Vallabhbhai National Institute Of  
Technology, Surat

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Data Collection</b>	<b>3</b>
2.1	Importing Libraries . . . . .	3
2.2	Gathering Data Files . . . . .	3
2.3	Consolidating Data . . . . .	4
<b>3</b>	<b>Change in Price of the Stock Over Time</b>	<b>5</b>
<b>4</b>	<b>The moving average of the various stocks</b>	<b>7</b>
<b>5</b>	<b>Analyse closing price change in apple stock</b>	<b>9</b>
<b>6</b>	<b>Performing resampling analysis of closing price</b>	<b>11</b>
<b>7</b>	<b>Checking if the closing prices of these tech companies are correlated</b>	<b>14</b>
<b>8</b>	<b>Analyse whether daily change in closing price of stocks or daily returns in stock are corelated</b>	<b>18</b>
<b>9</b>	<b>Conclusion</b>	<b>21</b>

# CHAPTER 1

## Introduction

The project on stock price analysis involves a comprehensive examination of the price movements and trends of various technology stocks over a five-year period. This analysis begins with the collection of data from multiple CSV files, each representing historical stock prices of prominent tech companies such as Apple, Google, Microsoft, and Amgen. Utilizing Python libraries such as pandas, numpy, matplotlib, and seaborn, the data is aggregated into a single dataframe for unified analysis. The project explores multiple dimensions of stock price behavior, including price changes over time, volatility, and other key financial metrics. The data preprocessing steps include handling missing values, ensuring proper data types, and merging datasets. Subsequent stages of the analysis involve visualizing stock price trends, identifying significant patterns, and making use of advanced plotting techniques to elucidate the dynamics of stock movements. The visualization efforts focus on understanding how stock prices have evolved, highlighting periods of significant growth or decline, and comparing performance across different companies. By employing various statistical and graphical methods, the project aims to derive meaningful insights that can inform investment decisions and enhance understanding of market behaviors. The analysis also delves into the correlation between different stocks, identifying how movements in one stock might predict or influence movements in another. This holistic approach not only provides a detailed view of individual stock performance but also sheds light on broader market trends and interdependencies. Overall, this stock price analysis project serves as a robust tool for financial analysts, investors, and anyone interested in the financial markets, offering a rich blend of data manipulation, statistical analysis, and insightful visualization.

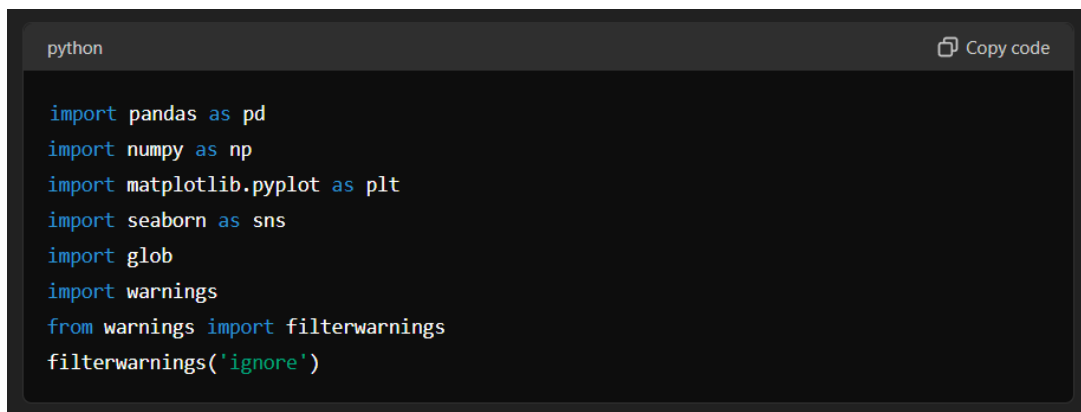
## CHAPTER 2

### Data Collection

Data collection involves gathering and recording information for analysis or research purposes. Methods include surveys, interviews, observations, and data mining. It ensures relevant and accurate data is obtained to support decision-making, research objectives, or understanding of phenomena in various fields such as science, business, and social sciences.

#### 2.1 Importing Libraries

Importing essential libraries like pandas for data manipulation, numpy for numerical operations, matplotlib and seaborn for data visualization, and glob for file handling is crucial in data analysis. Pandas offers powerful data structures and tools, numpy supports efficient numerical computations, matplotlib and seaborn provide versatile plotting capabilities, and glob facilitates file path matching. These libraries collectively enable tasks such as data cleaning, transformation, statistical analysis, and creating insightful visualizations, essential for extracting meaningful insights from data in fields like research, business analytics, and scientific studies. By suppressing warnings, the code ensures a smoother workflow during data processing and analysis.

A screenshot of a code editor window with a dark background. The title bar at the top left says "python" and the top right has a "Copy code" button. The code is written in a light blue/green monospace font. It imports pandas as pd, numpy as np, matplotlib.pyplot as plt, seaborn as sns, glob, and warnings. It then imports filterwarnings from warnings and calls filterwarnings('ignore').

```
python                                                                    Copy code

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import glob
import warnings
from warnings import filterwarnings
filterwarnings('ignore')
```

#### 2.2 Gathering Data Files

The glob module is commonly employed to dynamically gather file paths based on specified patterns, such as all CSV files within a directory. However, your approach directly lists exact paths for Apple (AAPL), Amgen (AMGN), Google (GOOG), and Microsoft (MSFT) stock data CSV files. This method offers explicit paths for each company's data, ensuring precise file access and streamlined processing. These paths facilitate straightforward loading and manipulation of data using tools like pandas and numpy, supporting

comprehensive analysis and visualization tasks for financial or business insights derived from historical stock data.

```
python Copy code

company_list = [
    r'D:\\Coding Folder\\DA In June A-Z\\Stok Price Project For Data Analyst\\Individual_S
    r'D:\\Coding Folder\\DA In June A-Z\\Stok Price Project For Data Analyst\\Individual_S
    r'D:\\Coding Folder\\DA In June A-Z\\Stok Price Project For Data Analyst\\Individual_S
    r'D:\\Coding Folder\\DA In June A-Z\\Stok Price Project For Data Analyst\\Individual_S
]
```

## 2.3 Consolidating Data

In the provided code snippet, each CSV file specified in company list is individually read into a pandas DataFrame (current df) and stored in data frames, a list designed to hold these data structures. This approach ensures that data from multiple sources (in this case, stock data for different companies) can be efficiently managed and processed. The `pd.concat()` function then merges all individual DataFrames from data frames into a unified DataFrame (all data), enabling comprehensive analysis and manipulation of combined data. This method is pivotal in consolidating disparate datasets into a cohesive format for seamless data exploration and insights generation in data-driven projects.

```
python Copy code

all_data = pd.DataFrame()
data_frames = []

for file in company_list:
    current_df = pd.read_csv(file)
    data_frames.append(current_df)

all_data = pd.concat(data_frames, ignore_index=True)
```

## CHAPTER 3

### Change in Price of the Stock Over Time

This section of the project focuses on analyzing how the stock prices of various companies have changed over a specific period. The objective is to track the historical performance of stock prices, identify trends, and understand the factors influencing these changes. By visualizing the price movements over time, we can gain insights into the overall performance and volatility of the stocks.

In this have to think about the closing prices of stocks in this section. We need to find the closing prices for all four companies and plot them. (All data types are correct, but one data type is an object. Change it from a string to a date-time.)

Code Explanation - The code converts the 'date' column in the all-data DataFrame from string to datetime format, then extracts unique company names. It creates a 2x2 grid plot for the closing prices of each company. Using a for loop with enumerate, it iterates through the companies, filters the data for each company, and plots the 'date' versus 'close' prices. Each subplot is titled with the corresponding company name, resulting in a comprehensive visual representation of the closing prices over time for all four companies.

#### Code

```
In [16]: all_data.dtypes
...

In [17]: pd.to_datetime(all_data['date'])
...

In [18]: all_data['date'] = pd.to_datetime(all_data['date'])

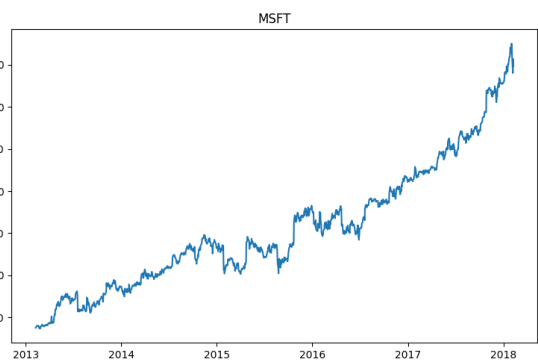
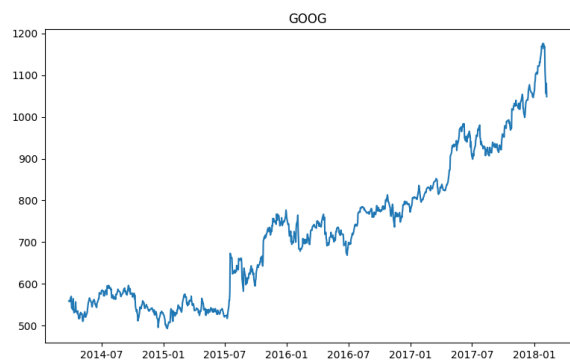
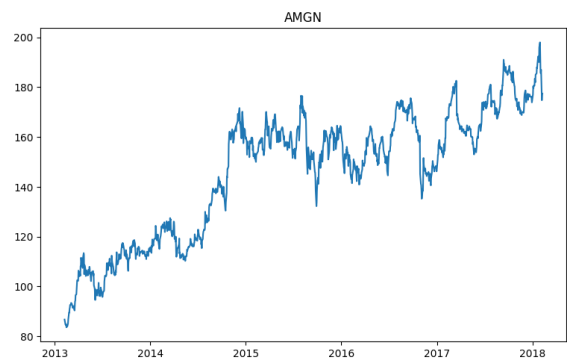
In [19]: all_data['date']
...

In [20]: tech_list = all_data['Name'].unique()

In [21]: tech_list
Out[21]: array(['AAPL', 'AMGN', 'GOOG', 'MSFT'], dtype=object)

In [22]: plt.figure(figsize=(20,12))
for index, company in enumerate(tech_list, 1):
    plt.subplot(2,2,index)
    filter1 = all_data['Name']==company
    df = all_data[filter1]
    plt.plot(df['date'], df['close'])
    plt.title(company)
```

#### Output



## CHAPTER 4

### The moving average of the various stocks

The provided code is a comprehensive Python script aimed at analyzing stock prices by calculating and visualizing moving averages for multiple companies. Initially, the code reads and displays the first 15 rows of the dataset to provide an overview of the stock data. It then computes a 10-day rolling mean of the 'close' prices, which helps smooth short-term fluctuations and highlight longer-term trends, displaying the first 14 values for quick inspection. A copy of the dataset is created to perform further operations without altering the original data. The script then calculates moving averages over three different periods: 10, 20, and 50 days, adding these as new columns to the dataset. This allows for a comparison of short, medium, and long-term price trends. The last 7 rows of this updated dataset are displayed to verify the changes. Subsequently, the 'date' column is set as the index of the dataset, facilitating time-series analysis. The entire dataset is then displayed, followed by a listing of its columns to ensure the moving average columns have been correctly added. The visualization part of the script sets up a large plotting area with a size of 20x20 inches to accommodate multiple subplots. For each company in the tech list, a subplot is created. The data is filtered to include only the rows corresponding to the current company, and the three moving averages (10, 20, and 50 days) are plotted on the same axes. The title of each subplot is set to the respective company's name. This approach provides a clear visual comparison of how different moving averages track the stock prices over time for each company. By analyzing these plots, one can observe how the stock prices react to different market conditions and identify potential buy or sell signals based on the intersections of these moving averages. Overall, this script offers a detailed and systematic approach to stock price analysis, leveraging rolling means and visualization techniques to uncover underlying trends and patterns in the data, making it a valuable tool for financial analysts and investors looking to make informed decisions based on historical stock performance.

Code



```

In [25]: all_data['close'].rolling(window=10).mean().head(14)
...

In [26]: new_data = all_data.copy()

In [27]: ma_day = [10,20,50]
for ma in ma_day:
    new_data['close_'+str(ma)] = new_data['close'].rolling(ma).mean()

In [28]: new_data.tail(7)
...

In [29]: new_data.set_index('date',inplace=True)

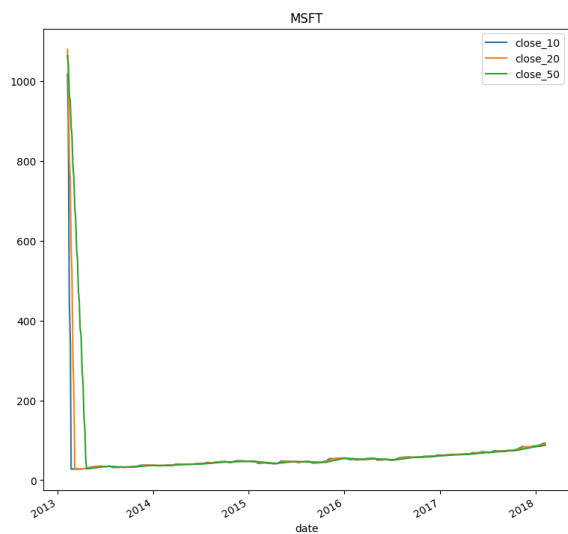
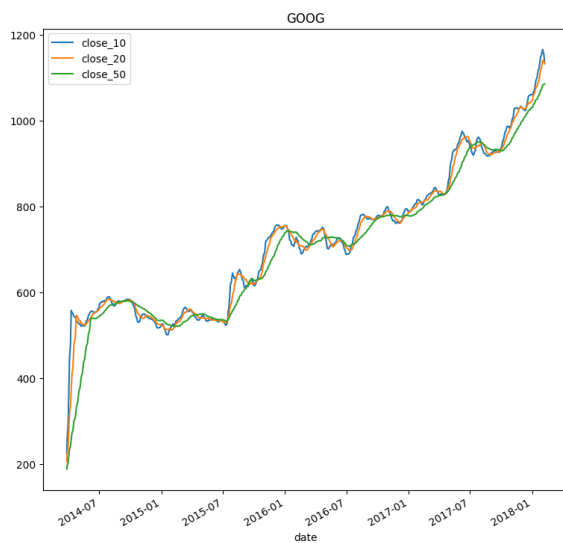
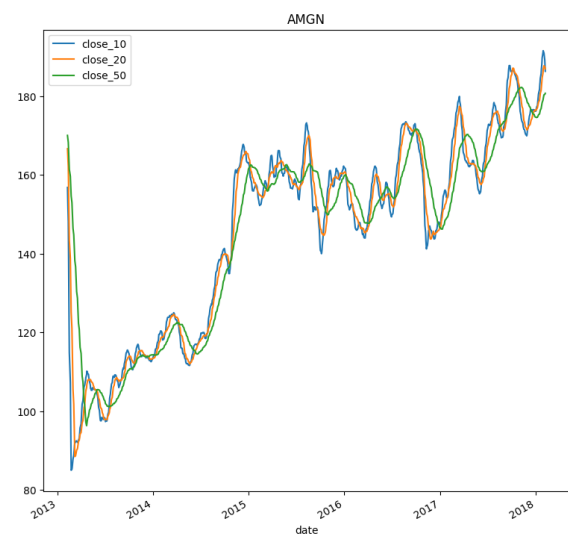
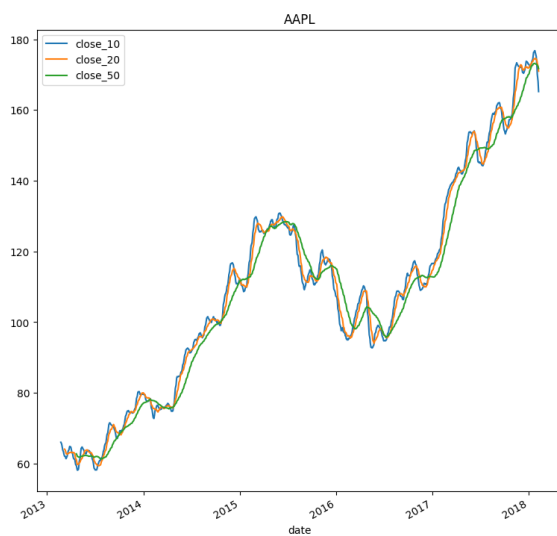
In [30]: new_data
...

In [31]: new_data.columns
...

In [32]: plt.figure(figsize=(20,20))
for index, company in enumerate(tech_list, 1):
    plt.subplot(2,2,index)
    filter1 = new_data['Name']==company
    df = new_data[filter1]
    df[['close_10', 'close_20', 'close_50']].plot(ax=plt.gca())
    plt.title(company)

```

## Output



## CHAPTER 5

### Analyse closing price change in apple stock

The provided code is a detailed script for analyzing the daily return percentages of Apple's stock prices and visualizing the results. The script starts by reading Apple's stock data from a CSV file into a pandas dataframe, allowing for structured data manipulation and analysis. The first few rows of the dataframe are displayed to ensure the data is loaded correctly. The 'close' column, representing the closing prices of the stock, is isolated for analysis. The script then calculates the percentage change in the closing prices, which represents the daily return, providing insight into the day-to-day performance of the stock. This calculation is initially displayed in its basic form to verify accuracy. The dataframe is updated to include a new column, 'Daily return(in per)', which contains the percentage change values multiplied by 100 to convert them into percentage form. This addition makes it easier to interpret the magnitude of daily returns. The first few rows of the updated dataframe are displayed again to ensure the new column has been correctly added. For visualization, the script uses the Plotly Express library, known for its interactive and aesthetically pleasing charts. It creates a line plot with 'date' on the x-axis and 'Daily return(in per)' on the y-axis. This plot provides a clear and dynamic representation of the volatility and trends in Apple's daily stock returns over time, highlighting periods of significant gains or losses. By visualizing the daily returns, the script allows for a quick assessment of the stock's historical performance and volatility, aiding in the identification of patterns or anomalies that could be crucial for making informed investment decisions. The combination of precise data manipulation and interactive visualization underscores the script's utility in financial analysis, offering a practical tool for analysts and investors to monitor and evaluate stock performance efficiently. This approach enhances understanding of stock behavior, supporting more strategic decision-making based on empirical data and trends.

Code

```

In [34]: company_list
...

In [35]: le = pd.read_csv(r'D:\\Coding Folder\\DA In June A-Z\\Stok Price Project For Data Analyst\\Individual_Stocks_5yr\\AAPL_data.csv')
...

In [36]: apple.head(4)
...

In [37]: apple['close']
...

In [38]: apple['close'].pct_change()
...

In [39]: apple.head(4)
...

In [40]: apple['close'].pct_change() * 100
...

In [41]: apple['Daily return(in %)'] = apple['close'].pct_change() * 100
...

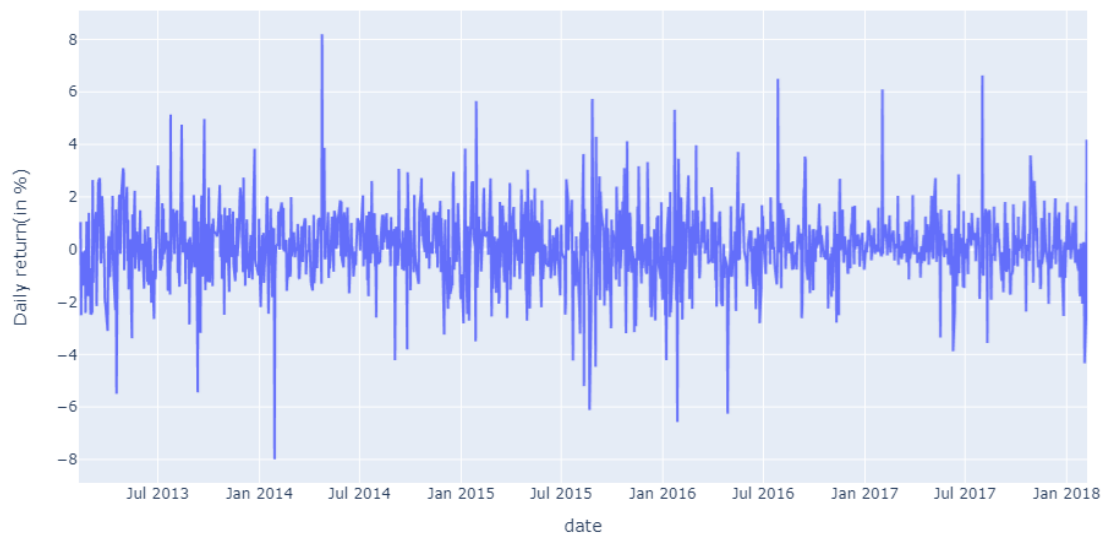
In [42]: apple.head(4)
...

In [43]: import plotly.express as px
...

In [44]: px.line(apple, x = "date" , y = "Daily return(in %)")

```

## Output



## CHAPTER 6

### Performing resampling analysis of closing price

Resampling analysis of closing prices involves aggregating and analyzing stock data at different time intervals to uncover trends and patterns that might not be evident in the daily data. This technique transforms the frequency of the time series data, such as converting daily closing prices into weekly, monthly, or quarterly averages or sums. By resampling, analysts can smooth out short-term volatility and gain insights into longer-term trends and cycles. For instance, weekly resampling could show general trends more clearly than daily data, highlighting sustained movements and filtering out noise. This process is particularly useful for identifying broader market cycles, seasonality effects, and long-term trends, which are critical for strategic decision-making and investment planning. It also facilitates comparison across different time periods and aids in more robust forecasting by providing a clearer view of the underlying price movements. Resampling can be easily performed using tools like pandas in Python, which offer functions to resample time series data at various frequencies, enabling a flexible and detailed analysis of stock performance. The provided code performs a resampling analysis of Apple's closing stock prices, converting daily data into monthly, yearly, and quarterly averages to identify broader trends. Initially, the code ensures that the 'date' column is in datetime format, which is crucial for time series analysis. After converting the 'date' column and setting it as the index, the daily closing prices are resampled to compute the mean closing prices for each month, year, and quarter. The monthly, yearly, and quarterly resampled data are then plotted to visualize these trends over different periods. Monthly resampling smooths out daily volatility, revealing clearer medium-term trends, while yearly resampling highlights long-term performance, showing significant annual shifts. Quarterly resampling provides an intermediate view, balancing between monthly and yearly trends. These plots allow for easy identification of sustained growth periods, seasonal effects, and other long-term patterns, facilitating more informed investment decisions by providing a clearer picture of Apple's stock performance across different timeframes.

Code

```

In [46]: apple.dtypes
...

In [47]: pd.to_datetime(apple['date'])
...

In [48]: apple['date'] = pd.to_datetime(apple['date'])
...

In [49]: apple.dtypes
...

In [50]: apple.head()
...

In [51]: apple.set_index('date', inplace = True)
...

In [52]: apple.head()
...

In [53]: apple['close'].resample('M').mean()
...

In [54]: apple['close'].resample('M').mean().plot()
...

In [55]: apple['close'].resample('Y').mean()
...

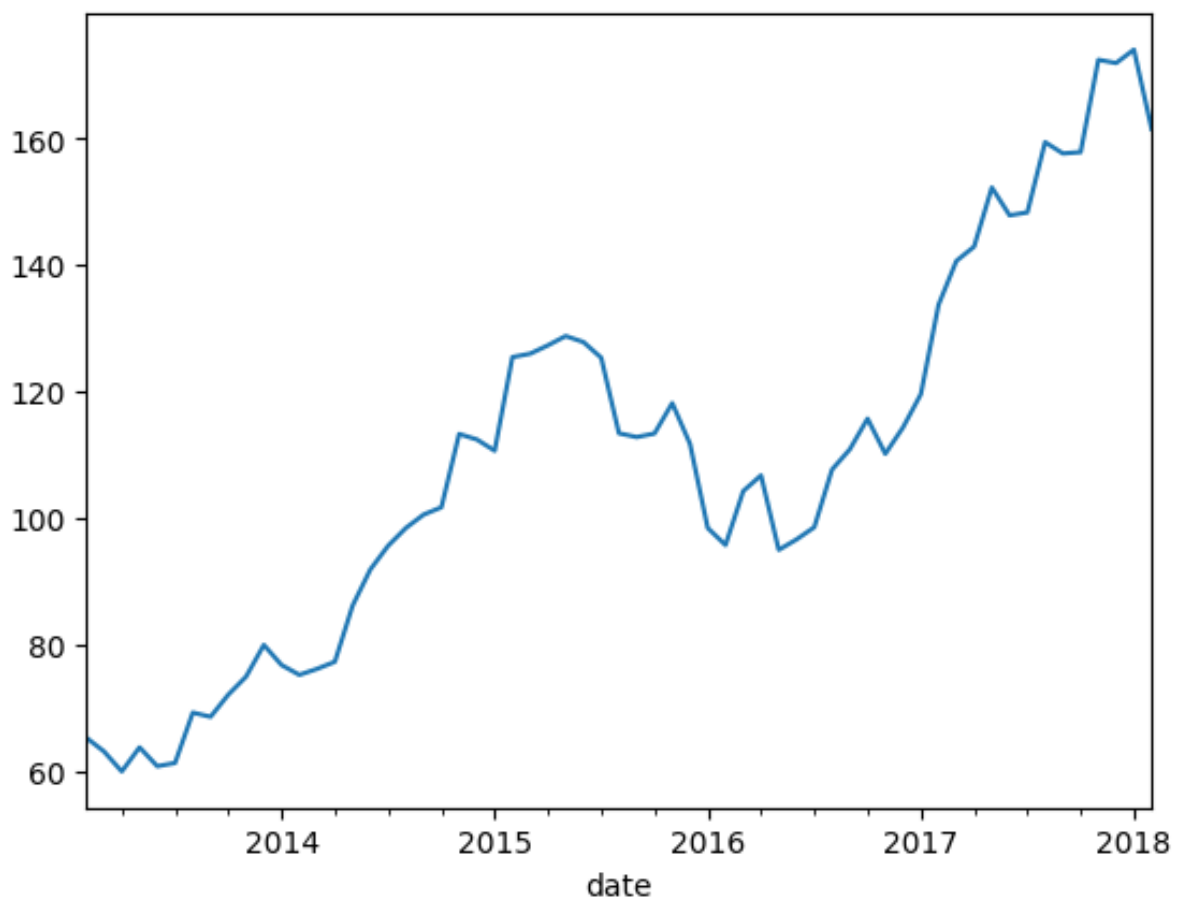
In [56]: apple['close'].resample('Y').mean().plot()
...

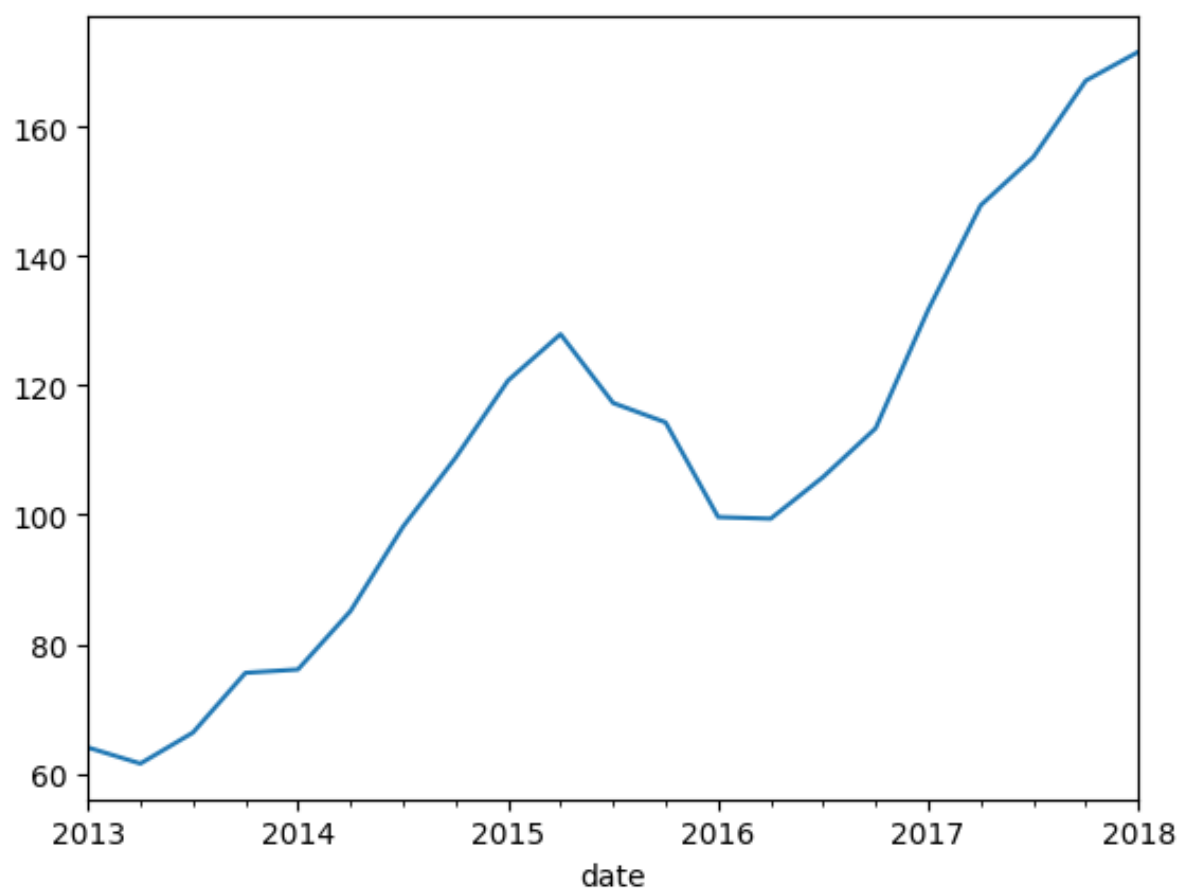
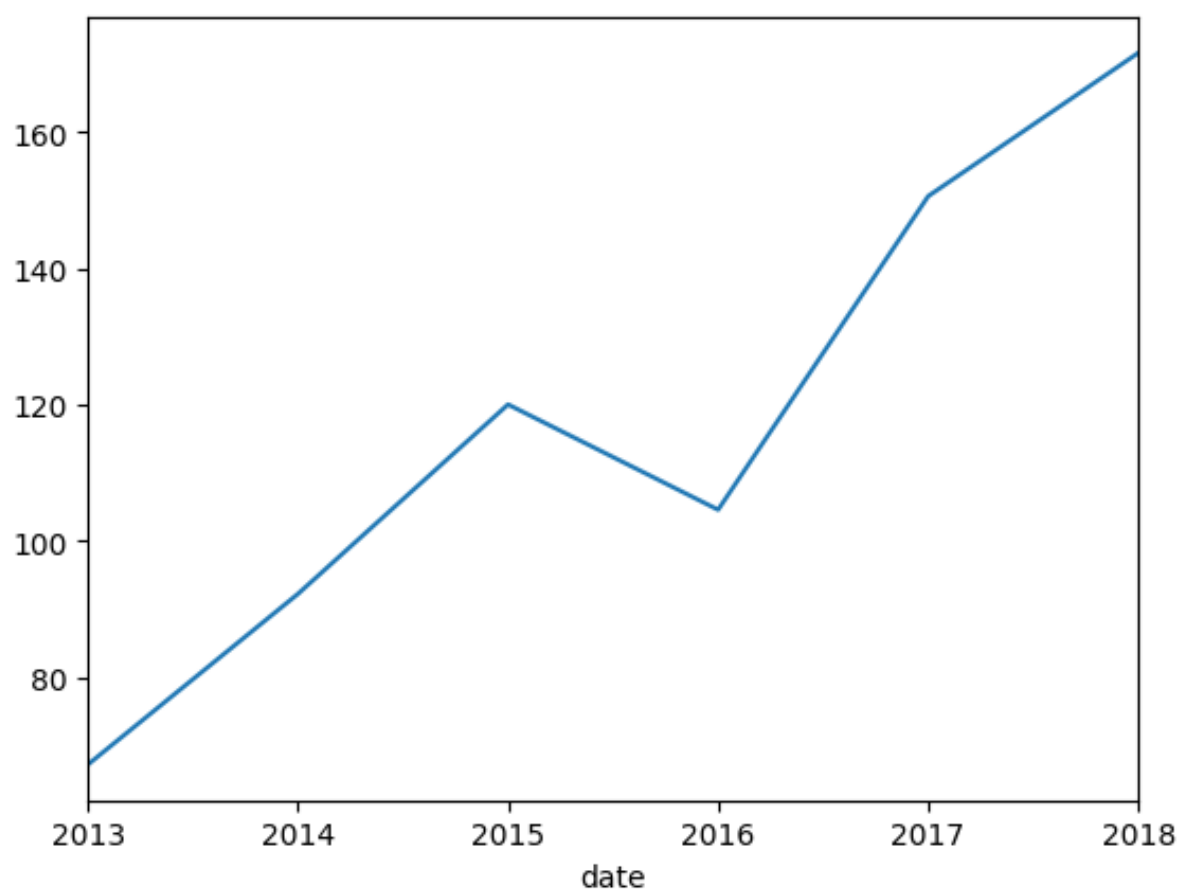
In [57]: apple['close'].resample('Q').mean()
...

In [58]: apple['close'].resample('Q').mean().plot()
...

```

Output





## CHAPTER 7

### Checking if the closing prices of these tech companies are correlated

The provided code carries out a comprehensive analysis of the closing prices of four major tech companies—Apple, Amazon, Google, and Microsoft—by examining their relationships through data visualization and correlation analysis. It begins by reading the stock data from CSV files specified in the `company-list`. The closing prices for each company are then extracted and compiled into a new dataframe, `closing-price`, with each column representing the closing prices of one company. This structured compilation facilitates a unified analysis of the stock performance across these companies. By using the `seaborn` library, the script generates a pairplot of the closing prices, offering a visual representation of the relationships between the stocks. This plot helps in identifying potential correlations and patterns at a glance. Following this, the script calculates the correlation matrix of the closing prices, which quantifies the degree of linear relationship between each pair of stocks. A high positive correlation indicates that the stocks tend to move in the same direction, while a negative correlation suggests they move in opposite directions. To visualize these correlations more effectively, the script uses a heatmap with annotated values. This heatmap provides a clear and intuitive representation of the correlation coefficients, making it easier to discern which pairs of stocks have strong, moderate, or weak correlations. Through this analysis, investors can gain insights into the interdependencies among these major tech stocks. For instance, a strong correlation between the closing prices of Apple and Microsoft might suggest that market forces affecting one company are likely to impact the other similarly. Such information is valuable for portfolio management, risk assessment, and diversification strategies. Understanding these correlations helps in making more informed decisions about which stocks to invest in together to maximize returns or minimize risks. The visual tools employed in this analysis, such as pairplots and heatmaps, are particularly effective in highlighting these relationships in a clear and accessible manner. This multifaceted approach not only provides a snapshot of individual stock performances but also elucidates the broader market dynamics and interconnections within the tech industry. By leveraging these analytical techniques, the code offers a robust framework for evaluating the behavior of multiple stocks simultaneously, thereby enhancing the analytical capabilities of financial analysts and investors. This comprehensive analysis underscores the importance of understanding stock correlations and their implications for strategic investment decisions.

Code

```
In [60]: company_list
```

```
...
```

```
In [61]: company_list[0]
```

```
...
```

```
In [62]: app = pd.read_csv(company_list[0])  
amzn = pd.read_csv(company_list[1])  
google = pd.read_csv(company_list[2])  
msft = pd.read_csv(company_list[3])
```

```
In [63]: closing_price = pd.DataFrame()
```

```
In [64]: app['close']
```

```
...
```

```
In [65]: closing_price['apple_close'] = app['close']  
closing_price['amzn_close'] = amzn['close']  
closing_price['google_close'] = google['close']  
closing_price['msft_close'] = msft['close']
```

```
In [66]: closing_price
```

```
...
```

```
In [67]: sns.pairplot(closing_price)
```

```
...
```

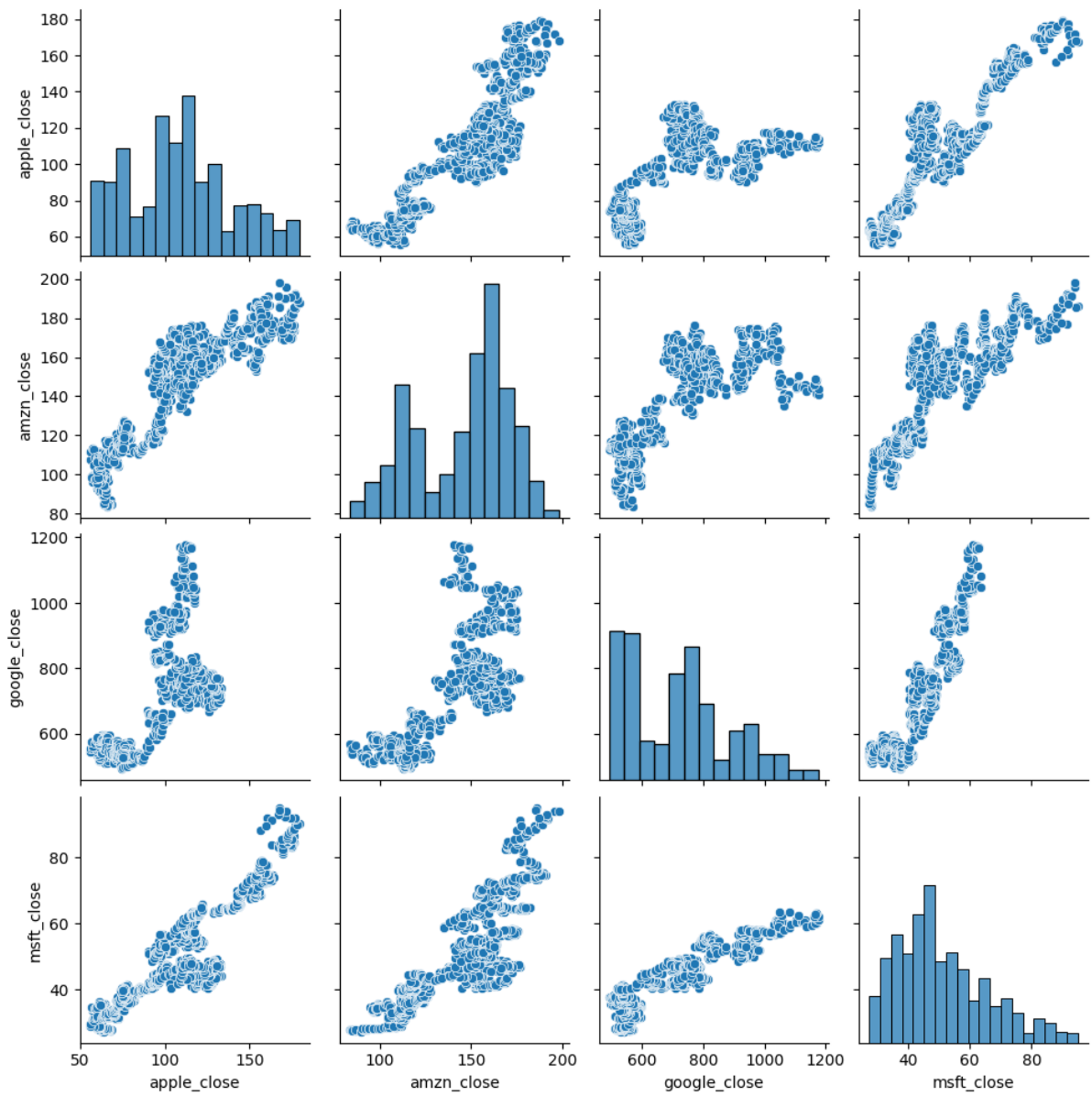
```
In [68]: closing_price.corr()
```

```
...
```

```
In [69]: sns.heatmap(closing_price.corr(), annot=True)
```

## Output







## CHAPTER 8

### Analyse whether daily change in closing price of stocks or daily returns in stock are corelated

The provided code performs a detailed percentage change analysis of the closing prices for four major tech companies—Apple, Amazon, Google, and Microsoft—followed by a comprehensive visualization of the relationships between their daily percentage changes. Initially, the script retrieves the closing prices from the closing-price dataframe, and calculates the daily percentage change for Apple’s closing prices by shifting the values by one day and computing the percentage difference. This calculation is then extended to all columns, where a loop iterates through each stock, computes the daily percentage change, and adds the results as new columns in the dataframe, suffixed with ‘-pct-change’. This transformation enables a day-over-day analysis of stock price movements in percentage terms, which normalizes the data and makes it easier to compare volatility and performance across different stocks. The resulting dataframe, `clsing-p`, now contains the percentage changes for all four stocks, facilitating a comparative analysis. To visualize these changes, the script utilizes seaborn’s PairGrid to create a grid of plots. The diagonal of the grid contains histograms, which display the distribution of daily percentage changes for each stock, providing insights into their volatility and typical daily movements. The lower triangle of the grid contains scatter plots, illustrating the pairwise relationships between the percentage changes of the different stocks, which helps in identifying direct visual correlations and patterns. The upper triangle features kernel density plots (KDE), offering a smoothed representation of the data distribution and further highlighting potential correlations. Additionally, the script calculates the correlation matrix for the percentage changes, which quantifies the linear relationships between the stocks. High positive correlations indicate that the stocks tend to move together, while negative correlations suggest inverse relationships. This matrix is crucial for understanding the interdependencies among these stocks, aiding in portfolio diversification and risk management strategies. Through this analysis, investors can discern how the daily movements of one stock may predict or relate to movements in another. For instance, if Apple’s daily percentage changes are highly correlated with Microsoft’s, market conditions affecting one are likely to impact the other similarly. Such insights are invaluable for developing investment strategies, as they allow for better prediction of stock behavior and more informed decision-making. The combination of visual and statistical tools in this analysis provides a robust framework for examining stock performance, offering a deeper understanding of market dynamics and the interplay between major tech stocks. This comprehensive approach enhances the analytical capabilities of financial analysts and investors, making it easier to identify trends, manage risk, and optimize portfolios based on empirical data and observed relationships.

## Code

```
In [71]: closing_price
...

In [72]: closing_price['apple_close']
...

In [73]: closing_price['apple_close'].shift(1)
...

In [74]: (closing_price['apple_close'] - closing_price['apple_close'].shift(1))/closing_price['apple_close'].shift(1) * 100
...

In [75]: for col in closing_price.columns:
        closing_price[col + '_pct_change'] = (closing_price[col] - closing_price[col].shift(1))/closing_price[col].shift(1) * 100

In [76]: closing_price
...

In [77]: closing_price.columns
...

In [78]: closing_price[['apple_close_pct_change', 'amzn_close_pct_change',
        'google_close_pct_change', 'msft_close_pct_change']]
...

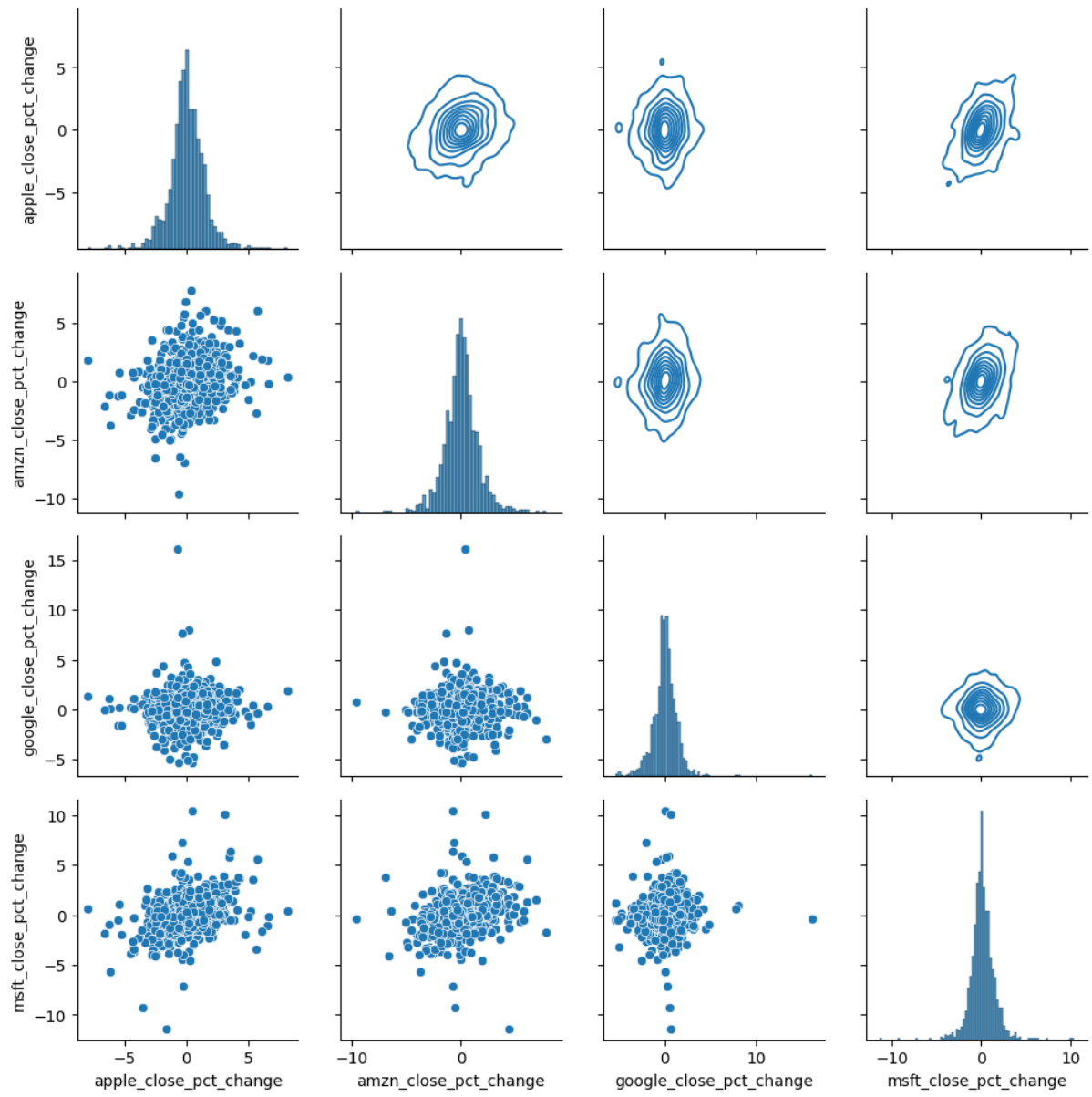
In [79]: clsing_p = closing_price[['apple_close_pct_change', 'amzn_close_pct_change',
        'google_close_pct_change', 'msft_close_pct_change']]

In [80]: clsing_p
...

In [81]: g = sns.PairGrid(data=clsing_p)
        g.map_diag(sns.histplot)
        g.map_lower(sns.scatterplot)
        g.map_upper(sns.kdeplot)
...

In [82]: clsing_p.corr()
```

## Output



## CHAPTER 9

### Conclusion

In conclusion, the stock price analysis project effectively demonstrates the power of data science techniques in understanding and interpreting financial market behaviors. By aggregating and analyzing five years of historical data from major tech companies, the project provides valuable insights into the trends and patterns that define stock price movements. The comprehensive use of data preprocessing, visualization, and statistical analysis allows for a nuanced exploration of factors influencing stock prices. Key findings include the identification of periods of significant price changes, the impact of market events on stock performance, and the interrelationships among different tech stocks. The project underscores the importance of data-driven decision-making in the financial sector, offering practical tools for investors and analysts to make informed choices. The insights gained from this analysis not only highlight past market trends but also equip stakeholders with the knowledge to anticipate future movements and strategize accordingly. Ultimately, this project showcases the critical role of data analytics in demystifying the complexities of stock market dynamics, paving the way for more strategic and informed financial practices.