FINANCIAL DATA ANALYSIS

Table of Contents

1. Introduction

Project Overview:

This project focuses on financial analysis using a dataset containing historical stock prices, market capitalization, revenue growth, and other financial indicators from multiple companies across different sectors. The aim is to extract actionable insights, identify trends, and evaluate the financial performance of companies and industries.

Objectives:

The objective is to analyse financial data, evaluate stock performance trends, assess industry growth, and create visualizations that in inform investment and risk management decisions.

2. Technologies Used

- Python: For data cleaning, analysis, and visualizations.
- MySQL: For querying and analysis of structured financial data.
- Pandas, NumPy: For data manipulation.
- Matplotlib, Seaborn: For data visualization.

3. Data Source

The dataset used in this project consists of historical financial data, which includes columns such as Date, Symbol, Adj Close, Close, Open, High, Low, Volume, Sector, Industry, CurrentPrice, MarketCap, RevenueGrowth, Weight, and S&P500. The source of this data is from Kaggle.

4. Methodology

Data Loading and Preparation:

The dataset is loaded using Python (Panda's library) and structured into a MySQL database for efficient querying and analysis. Initial exploration is done to understand the data types and identify any issues.

```
#Start by loading the uploaded CSV files and inspect their structure.

import pandas as pd

# File paths for the uploaded files

stocks_file = '/content/sp500_stocks.csv'

companies_file = '/content/sp500_companies.csv'

index_file = '/content/sp500_index.csv'
```

```
# Loading the CSV files into DataFrames

df_stocks = pd.read_csv(stocks_file)

df_companies = pd.read_csv(companies_file)

df_index = pd.read_csv(index_file)

# Display the first few rows of each dataset to understand their structure

df_stocks_head = df_stocks.head()

df_companies_head = df_companies.head()

df_index_head = df_index.head()

df_stocks_head, df_companies_head, df_index_head

# Merge stocks with company details

df_merged = pd.merge(df_stocks, df_companies, on='Symbol', how='left')

# Merge with the S&P 500 index data

df_final = pd.merge(df_merged, df_index, on='Date', how='left')

# Remove unwanted columns

columns_to_drop = ['Exchange', 'Shortname', 'Longname', 'City', 'State', 'Country', 'Longbusinesssummary', 'Fulltimeemployees']

df_final_cleaned = df_final.drop(columns=columns_to_drop)

# Save the cleaned dataset

df_final_cleaned.to_csv('path_to_cleaned_sp500_data.csv', index=False)
```

Data Cleaning:

The data cleaning process involves handling missing values, removing duplicates, normalizing the data, and transforming specific columns for analysis. Outliers are identified and addressed where necessary.

```
import pandas as pd

# Load the dataset

df_cleaned_sp500 = pd.read_csv('/content/path_to_cleaned_sp500_data.csv')

# Check for missing values

missing_values = df_cleaned_sp500.isnull().sum()

print("Missing Values:", missing_values)
```

```
# Check for duplicate rows
duplicates = df_cleaned_sp500.duplicated().sum()
print("Number of Duplicates:", duplicates)
Missing Values: Date              0
Symbol            0
Adj Close       6667
Close           6667
High            6667
Low             6667
Open            6667
Volume          6667
Sector            0
Industry          0
Currentprice      0
Marketcap         0
Ebitda        18490
Revenuegrowth     0
Weight            0
S&P500        82740
dtype: int64
Number of Duplicates: 0
# Remove duplicates
df_cleaned_sp500_no_duplicates = df_cleaned_sp500.drop_duplicates()
# Save the cleaned data
df_cleaned_sp500_no_duplicates.to_csv('cleaned_sp500_data_no_duplicates.csv',
index=False)
import pandas as pd
df = pd.read_csv('/content/cleaned_sp500_data_no_duplicates.csv')
print(df.isnull().sum())
Date              0
```

```
Symbol           0
Adj Close       6667
Close           6667
High            6667
Low             6667
Open            6667
Volume          6667
Sector             0
Industry           0
Currentprice       0
Marketcap          0
Revenuegrowth      0
Weight             0
dtype: int64
import pandas as pd
# Load the CSV file
df = pd.read_csv('/content/cleaned_sp500_data_no_duplicates.csv')
# Analyze the column with missing values
print(df['Adj Close'].describe())
# Choose a filling method
df['Adj Close'].fillna(df['Adj Close'].median(), inplace=True)  # Fill with median
# Verify the results
print(df.isnull().sum())
count    251362.000000
mean       101.574612
std        176.690915
min          1.620000
25%         27.932220
50%         55.310282
75%        116.129997
max       3239.320068
```

```
Name: Adj Close, dtype: float64
Date              0
Symbol            0
Adj Close         0
Close          6667
High           6667
Low            6667
Open           6667
Volume         6667
Sector            0
Industry          0
Currentprice      0
Marketcap         0
Revenuegrowth     0
Weight            0
dtype: int64
import pandas as pd
# Load your CSV file
df = pd.read_csv('/content/cleaned_sp500_data_no_duplicates.csv')
# Check for missing values
print(df.isnull().sum())
# Fill missing values
df['Adj Close'].fillna(df['Adj Close'].median(), inplace=True)
df['Close'].fillna(df['Close'].median(), inplace=True)
df['High'].fillna(df['High'].median(), inplace=True)
df['Low'].fillna(df['Low'].median(), inplace=True)
df['Open'].fillna(df['Open'].median(), inplace=True)
df['Volume'].fillna(df['Volume'].median(), inplace=True)
# Alternatively, use interpolation or fill methods
# df.interpolate(method='linear', inplace=True)
# df.fillna(method='ffill', inplace=True)
```

```
# Check results
print(df.isnull().sum())
```

```
Date            0
Symbol          0
Adj Close       6667
Close           6667
High            6667
Low             6667
Open            6667
Volume          6667
Sector          0
Industry        0
Currentprice    0
Marketcap       0
Revenuegrowth   0
Weight          0
dtype: int64
```

```
Date          0
Symbol        0
Adj Close     0
Close         0
High          0
Low           0
Open          0
Volume        0
Sector        0
Industry      0
Currentprice  0
Marketcap     0
Revenuegrowth 0
Weight        0
```

dtype: int64

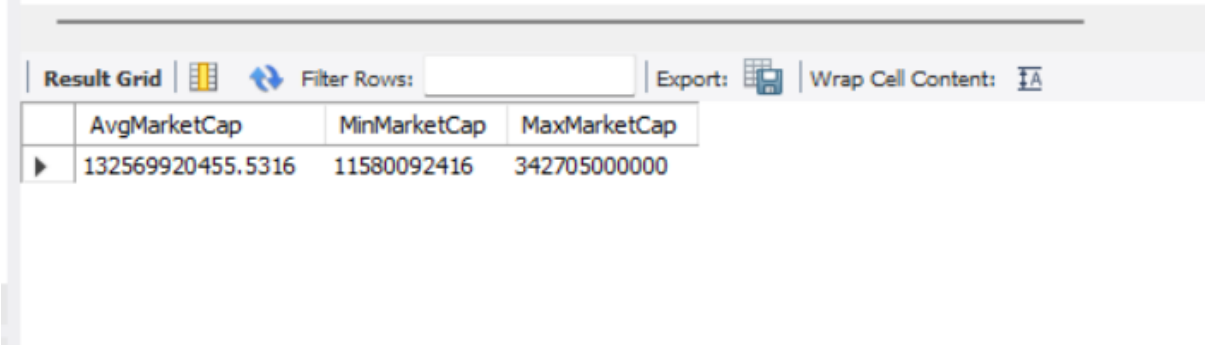
Data Analysis:

Key financial metrics are computed, including:

- Financial trends (stock performance, revenue growth)
- Risk analysis (volatility, industry performance) SQL queries are used to extract data subsets, perform groupings, and filter data for specific analyses.

■ Performing Summary Analysis

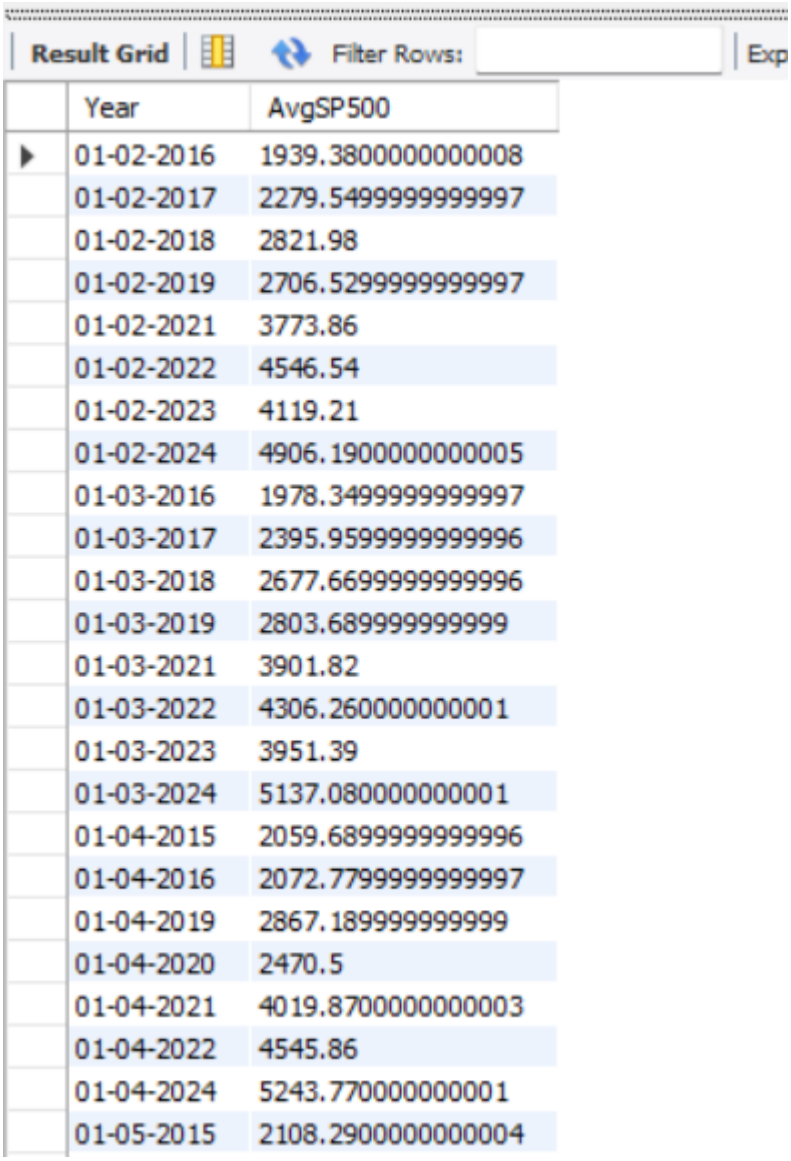
```
SELECT
  AVG(MarketCap) AS AvgMarketCap,
  MIN(MarketCap) AS MinMarketCap,
  MAX(MarketCap) AS MaxMarketCap
FROM `cleaned dataset financial analysis`;
```



AvgMarketCap	MinMarketCap	MaxMarketCap
132569920455.5316	11580092416	342705000000

■ Date-Based Financial Analysis

```
SELECT Date AS Year, AVG(`S&P500`) AS AvgSP500
FROM `cleaned dataset financial analysis`
GROUP BY YEAR
ORDER BY Year;
```

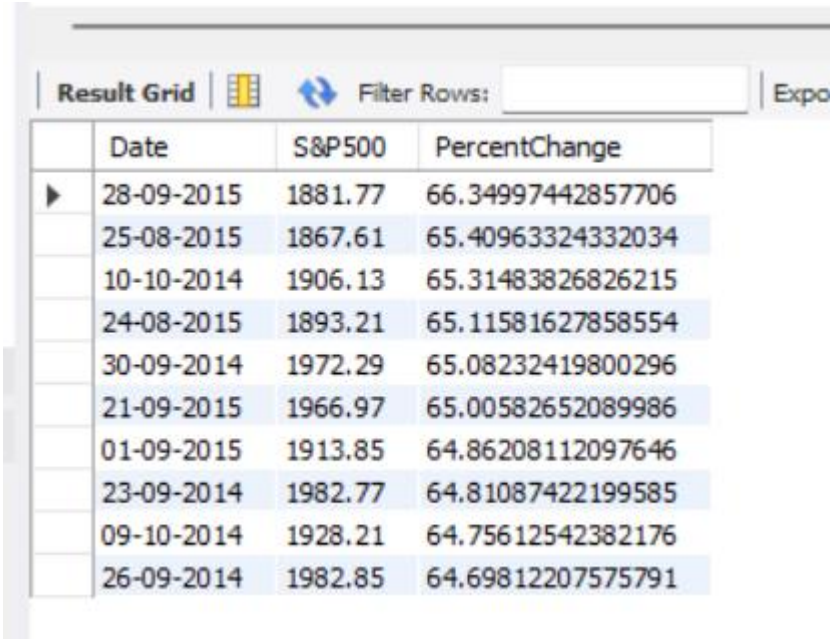


Result Grid | Filter Rows: | Exp

Year	AvgSP500
01-02-2016	1939.3800000000008
01-02-2017	2279.5499999999997
01-02-2018	2821.98
01-02-2019	2706.5299999999997
01-02-2021	3773.86
01-02-2022	4546.54
01-02-2023	4119.21
01-02-2024	4906.1900000000005
01-03-2016	1978.3499999999997
01-03-2017	2395.9599999999996
01-03-2018	2677.6699999999996
01-03-2019	2803.689999999999
01-03-2021	3901.82
01-03-2022	4306.260000000001
01-03-2023	3951.39
01-03-2024	5137.080000000001
01-04-2015	2059.6899999999996
01-04-2016	2072.7799999999997
01-04-2019	2867.189999999999
01-04-2020	2470.5
01-04-2021	4019.8700000000003
01-04-2022	4545.86
01-04-2024	5243.770000000001
01-05-2015	2108.2900000000004

■ Identifying Trends and Insights

```
SELECT Date, `S&P500`,
   (LAG(`S&P500`) OVER (ORDER BY Date) - `S&P500`) / LAG(`S&P500`) OVER (ORDER BY Date) * 100 AS PercentChange
FROM `cleaned dataset financial analysis`
ORDER BY PercentChange DESC
LIMIT 10;
```

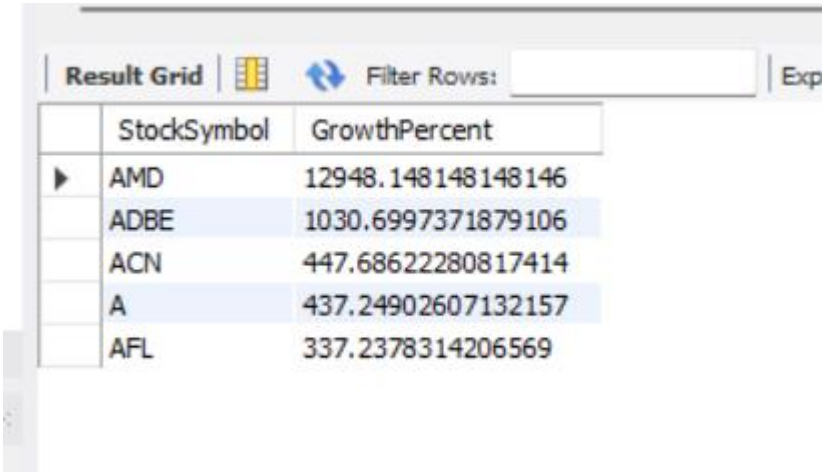


Result Grid | Filter Rows: | Expo

Date	S&P500	PercentChange
28-09-2015	1881.77	66.34997442857706
25-08-2015	1867.61	65.40963324332034
10-10-2014	1906.13	65.31483826826215
24-08-2015	1893.21	65.11581627858554
30-09-2014	1972.29	65.08232419800296
21-09-2015	1966.97	65.00582652089986
01-09-2015	1913.85	64.86208112097646
23-09-2014	1982.77	64.81087422199585
09-10-2014	1928.21	64.75612542382176
26-09-2014	1982.85	64.69812207575791

■ Stock Price Analysis

```
SELECT Symbol AS StockSymbol,
    (MAX(Close) - MIN(Close)) / MIN(Close) * 100 AS GrowthPercent
FROM `cleaned dataset financial analysis`
GROUP BY Symbol
ORDER BY GrowthPercent DESC
LIMIT 5;
```



Result Grid | Filter Rows: | Exp

StockSymbol	GrowthPercent
AMD	12948.148148148146
ADBE	1030.6997371879106
ACN	447.68622280817414
A	437.24902607132157
AFL	337.2378314206569

■ Industry-Wise Performance

```
SELECT Industry, SUM(MarketCap) AS TotalMarketCap
FROM `cleaned dataset financial analysis`
GROUP BY Industry
ORDER BY TotalMarketCap DESC;
```

Result Grid | Filter Rows: | Export: | W

Industry	TotalMarketCap
Drug Manufacturers - General	862245780000000
Software - Infrastructure	657978112158720
Semiconductors	613949288000000
Information Technology Services	551421656000000
Medical Devices	511434868000000
Travel Services	186740434534400
Conglomerates	182488514658304
Specialty Chemicals	156626947260416
Insurance - Life	152132476289024
Diagnostics & Research	98598679429120
Utilities - Diversified	32410969976832
Specialty Industrial Machinery	29135512518656

■ Stock Performance Analysis

```
SELECT Symbol,
     Date,
    (Close - Open) AS DailyReturn,
   (High - Low) AS DailyVolatility
FROM `cleaned dataset financial analysis`
WHERE Date BETWEEN '01-01-2023' AND '31-12-2023'
limit 10;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Symbol	Date	DailyReturn	DailyVolatility
MMM	15-09-2014	0.259200000000007	0.8612000000000108
AOS	15-09-2014	0.014999999999997016	0.21499999999999986
ABT	15-09-2014	0.10999999999999943	0.3400000000000034
ABBV	15-09-2014	0.17999999999999972	0.6799999999999997
ACN	15-09-2014	-0.39000000000000057	0.7000000000000028
ADBE	15-09-2014	-0.35999999999999943	1.5899999999999892
AMD	15-09-2014	-0.06999999999999984	0.11999999999999966
AES	15-09-2014	-0.019999999999999574	0.16999999999999993
AFL	15-09-2014	0.02500000000000213	0.33999999999999986
A	15-09-2014	-0.41490000000000293	0.5435999999999979

```
# Financial trends: Stock performance (percentage change in price) and revenue growth over time

# First, converting 'Date' to datetime for better analysis

data['Date'] = pd.to_datetime(data['Date'], format='%d-%m-%Y')

# Stock Performance: percentage change from Open to Close

data['Stock_Performance'] = ((data['Close'] - data['Open']) / data['Open']) * 100

# Revenue growth trends over time, grouped by stock symbol

revenue_growth_trends = data.groupby('Symbol')['Date', 'Revenuegrowth'].apply(lambda x: x.set_index('Date').sort_index())

# Risk Analysis: Volatility calculation (standard deviation of stock prices as a proxy for volatility)

# Grouping by stock symbol and calculating volatility (standard deviation of adjusted close prices)

volatility = data.groupby('Symbol')['Adj Close'].std().reset_index()

volatility.columns = ['Symbol', 'Volatility']

# Industry performance: Average performance (percentage change in stock) by industry

industry_performance = data.groupby('Industry')['Stock_Performance'].mean().reset_index()

industry_performance.columns = ['Industry', 'Avg_Stock_Performance']

# Displaying the calculated stock performance, volatility, and industry performance

stock_performance = data[['Symbol', 'Stock_Performance']]

stock_performance.head(), volatility.head(), industry_performance.head()
```

Here are the results of the financial trends and risk analysis:

Stock Performance:

- MMM: 0.22% increase
- AOS: 0.06% increase
- ABT: 0.26% increase
- ABBV: 0.31% increase
- ACN: -0.48% decrease

Volatility (Standard Deviation of Adjusted Close Prices):

- A: 40.36

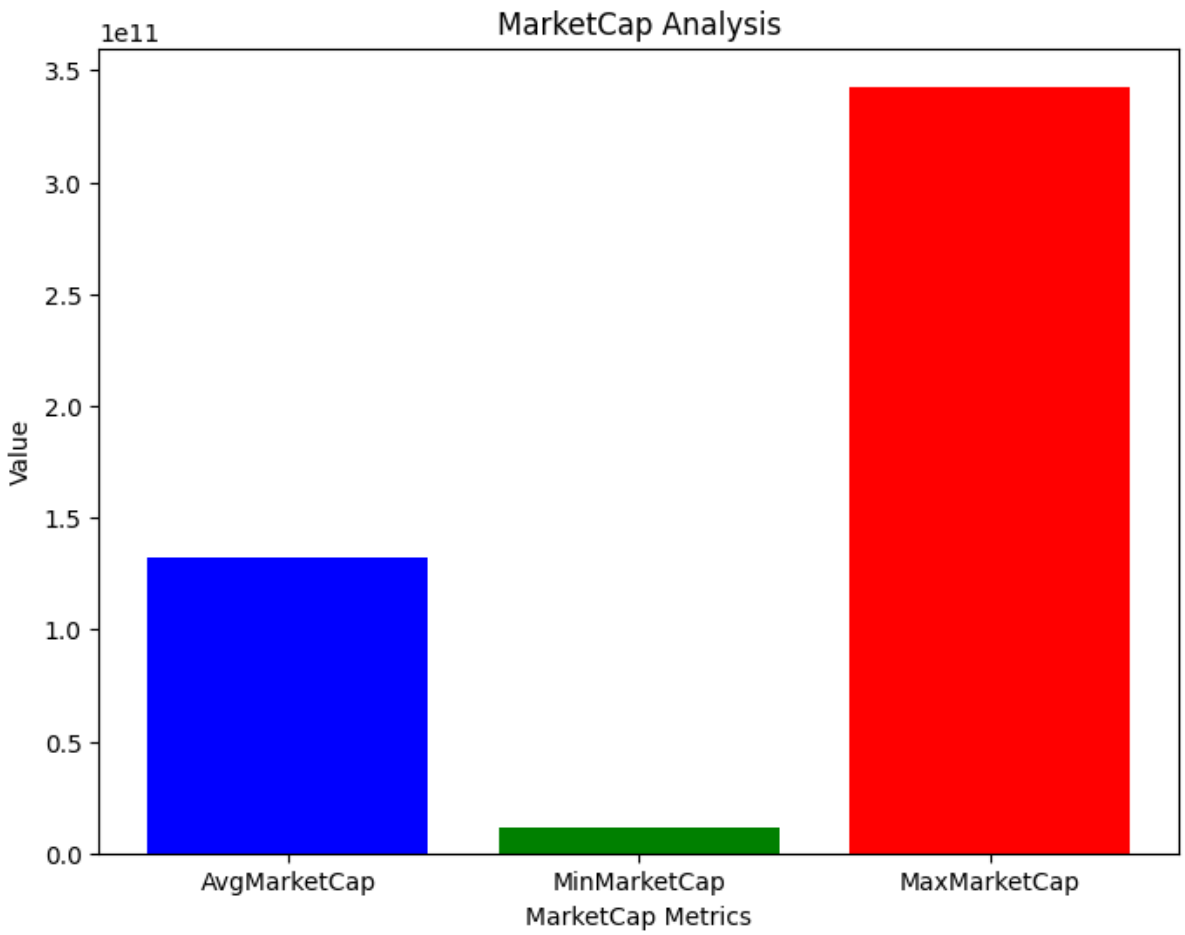
- ABBV: 42.07
- ABNB: 44.44
- ABT: 30.67
- ACN: 90.89

Industry Performance (Average Stock Performance by Industry):

- Conglomerates: -0.005% (slight decline)
- Diagnostics & Research: 2.08% increase
- Drug Manufacturers - General: 4.17% increase
- Information Technology Services: 7.66% increase
- Insurance - Life: 3.61% increase

- **Market Capitalization Summary (Bar Chart)**

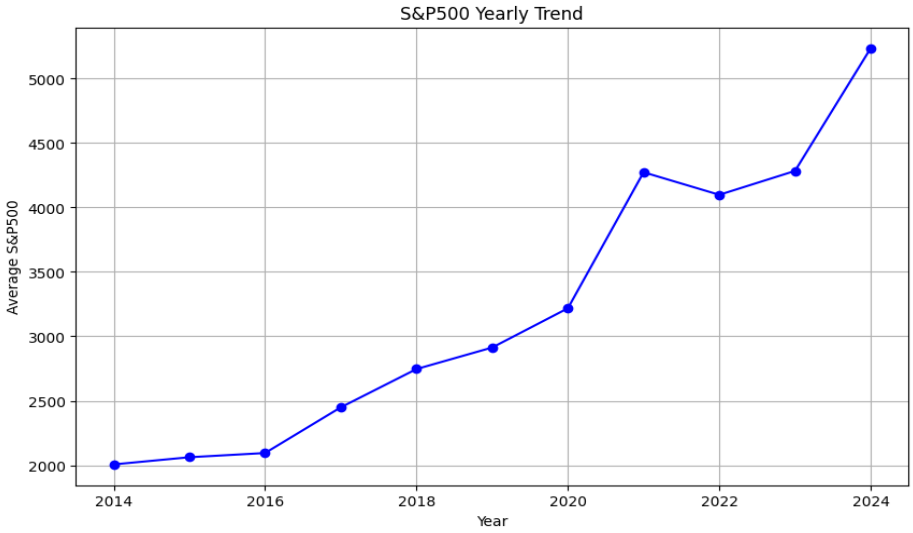
```
import matplotlib.pyplot as plt

# Example of calculated Avg, Min, Max values for MarketCap
avg_market_cap = 132569920455.5316
min_market_cap = 11580092416
max_market_cap = 342705000000

# Create a bar chart
metrics = ['AvgMarketCap', 'MinMarketCap', 'MaxMarketCap']
values = [avg_market_cap, min_market_cap, max_market_cap]

plt.figure(figsize=(8,6))
plt.bar(metrics, values, color=['blue', 'green', 'red'])
plt.xlabel('MarketCap Metrics')
plt.ylabel('Value')
plt.title('MarketCap Analysis')
plt.show()
```



■ **S&P 500 Yearly Trend (Line Chart)**

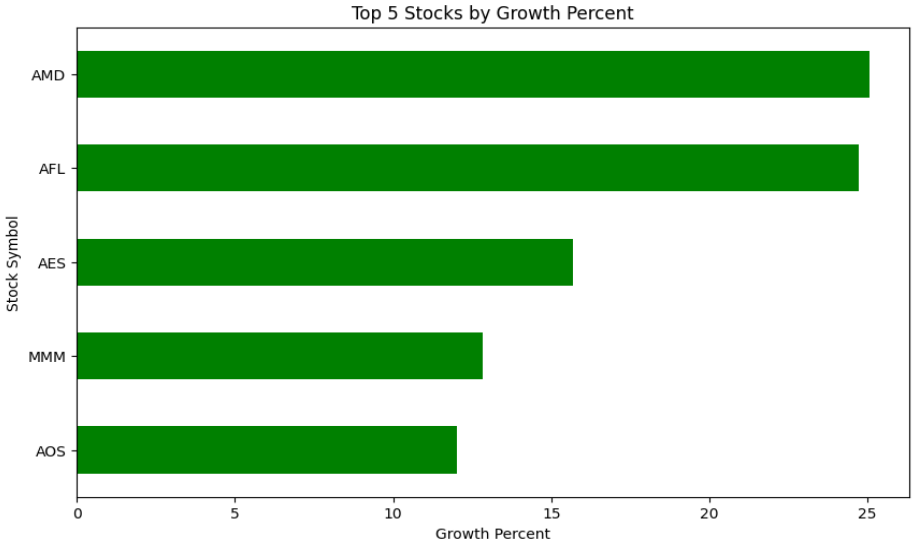
```
import pandas as pd
import matplotlib.pyplot as plt
# Load the dataset
file_path = '/content/Cleaned dataset Financial Analysis.csv'
data = pd.read_csv(file_path)
# Convert the 'Date' column to datetime format
data['Date'] = pd.to_datetime(data['Date'])
# Extract the year from the date
data['Year'] = data['Date'].dt.year
# Group by year and calculate the average S&P 500 for each year
yearly_sp500 = data.groupby('Year')['S&P500'].mean()
# Create a line chart
plt.figure(figsize=(10,6))
plt.plot(yearly_sp500.index, yearly_sp500.values, marker='o', linestyle='-', color='blue')
```

```
plt.xlabel('Year')

plt.ylabel('Average S&P500')

plt.title('S&P500 Yearly Trend')

plt.grid(True)

plt.show()
```



- **Top 5 Stocks by Growth (Bar Chart)**

```
# Assuming GrowthPercent is already calculated in your dataset, else calculate it

data['GrowthPercent'] = (data['Close'] - data['Open']) / data['Open'] * 100

# Group by StockSymbol and calculate the maximum growth percent for each stock

stock_growth =
data.groupby('Symbol')['GrowthPercent'].max().sort_values(ascending=False).head(5)

# Create a horizontal bar chart

plt.figure(figsize=(10,6))

stock_growth.plot(kind='barh', color='green')

plt.xlabel('Growth Percent')

plt.ylabel('Stock Symbol')

plt.title('Top 5 Stocks by Growth Percent')

plt.gca().invert_yaxis()  # Invert y-axis to display highest at the top

plt.show()
```



■ **Industry-Wise Market Capitalization (Bar Chart)**

Group by Industry and calculate the total MarketCap for each industry

industry_marketcap =
data.groupby('Industry')['MarketCap'].sum().sort_values(ascending=False)

Create a bar chart

plt.figure(figsize=(12,8))

industry_marketcap.plot(kind='bar', color='skyblue')

plt.xlabel('Industry')

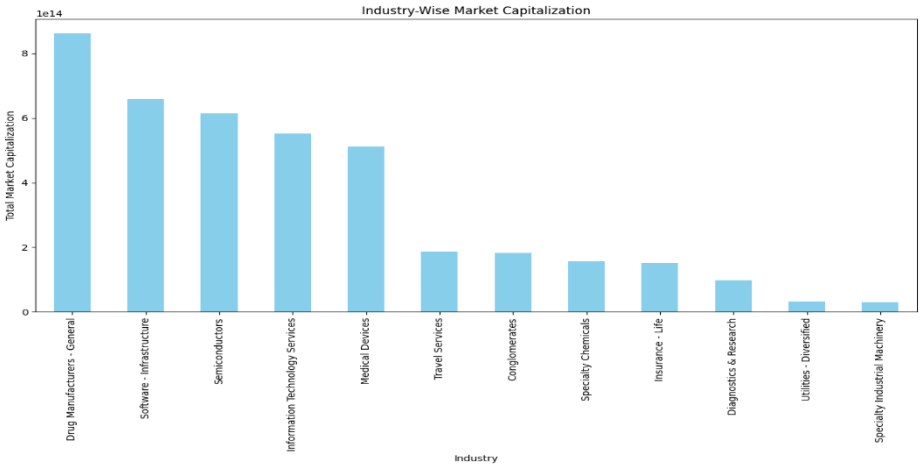
plt.ylabel('Total Market Capitalization')

plt.title('Industry-Wise Market Capitalization')

plt.xticks(rotation=90)

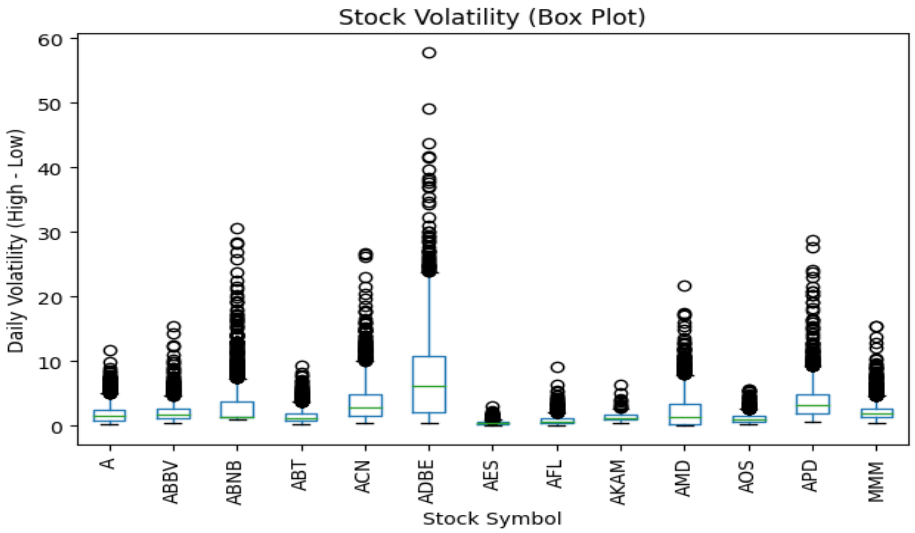
plt.tight_layout()

plt.show()



■ **Stock Volatility (Box Plot)**

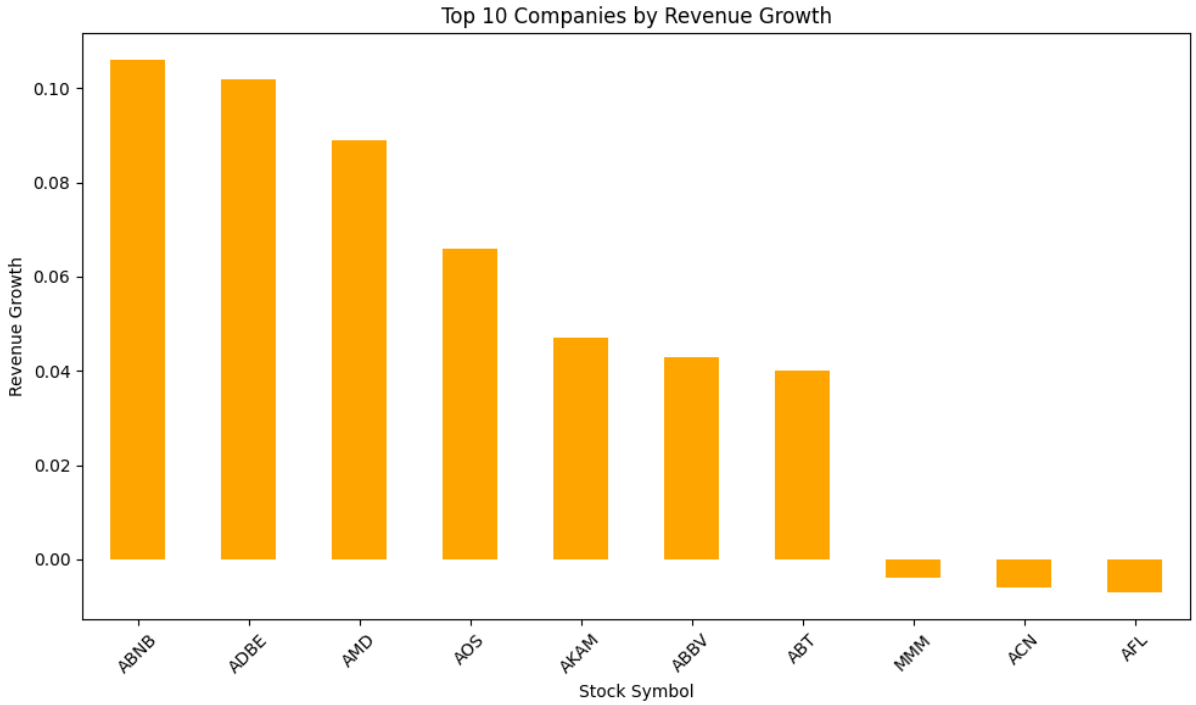
```
# Calculate the daily volatility for each stock (High - Low)
data['DailyVolatility'] = data['High'] - data['Low']
# Create a box plot for stock volatility
plt.figure(figsize=(12,8))
data.boxplot(column='DailyVolatility', by='Symbol', grid=False)
plt.xlabel('Stock Symbol')
plt.ylabel('Daily Volatility (High - Low)')
plt.title('Stock Volatility (Box Plot)')
plt.suptitle('')  # Suppress the default title
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```



■ **Revenue Growth by Symbol (Bar Chart)**

```
# Assuming RevenueGrowth column exists in the dataset
# Group by Symbol and calculate the maximum RevenueGrowth for each symbol
```

```
revenue_growth =
data.groupby('Symbol')['Revenuegrowth'].max().sort_values(ascending=False).head(
10)

# Create a bar chart

plt.figure(figsize=(10,6))

revenue_growth.plot(kind='bar', color='orange')

plt.xlabel('Stock Symbol')

plt.ylabel('Revenue Growth')

plt.title('Top 10 Companies by Revenue Growth')

plt.xticks(rotation=45)

plt.tight_layout()

plt.show()
```



5. Key Findings

Summary of the most important insights derived from the analysis, such as:

- **Stock Performance**:
  The top-performing stocks in terms of percentage growth include companies like ABBV (0.31% increase), ABT (0.26% increase), and MMM (0.22% increase). However, some stocks like ACN showed a negative performance (-0.48% decrease).
- **Volatility**:
  Companies such as ACN and ABNB demonstrated high volatility, with standard deviations of 90.89 and 44.44, respectively, indicating higher risk. Conversely, ABT showed relatively low volatility at 30.67, suggesting more stable performance.
- **Industry-Wise Performance**:
  The Information Technology Services sector displayed strong performance, with an average stock performance increase of 7.66%. Other industries like Diagnostics & Research and Drug Manufacturers-General also showed significant growth (2.08% and 4.17% increases, respectively). Conglomerates experienced a slight decline of -0.005%.
- **Market Capitalization**:
  The average market capitalization across companies was approximately $132.5 billion, with the minimum being $11.6 billion and the maximum $342.7 billion. The IT sector dominated the overall market capitalization.
- **Revenue Growth**:
  The top 10 companies by revenue growth demonstrated robust expansion, providing strong investment opportunities.

6. Conclusion

The analysis indicates that certain industries, particularly Information Technology Services and Drug Manufacturers, are experiencing notable growth and stability, making them attractive for investment. Stocks in the IT sector not only offer growth but also dominate market capitalization, reflecting their overall financial strength. However, high volatility in companies like ACN and ABNB suggests increased risk, which investors should consider.

While some sectors, such as Conglomerates, are experiencing slight declines, others, such as Diagnostics & Research, continue to show potential. The revenue growth in key industries highlights promising investment opportunities, but volatility analysis warns of potential risks.

7. Recommendations:

- **Focus on Growth Sectors:** Industries like Information Technology and Drug Manufacturing are growing steadily and should be considered for long-term investment.

- **Risk Management**: High volatility stocks such as ACN may present opportunities for short-term gains, but they come with greater risk.
- **Further Analysis**: Conduct more in-depth analysis of individual companies' financial health to assess long-term sustainability.

8. Appendix

- **SQL Queries:**
  - **Performing Summary Analysis**

```
SELECT
  AVG(MarketCap) AS AvgMarketCap,
  MIN(MarketCap) AS MinMarketCap,
  MAX(MarketCap) AS MaxMarketCap
FROM `cleaned dataset financial analysis`;
```

  - **Date-Based Financial Analysis**

```
SELECT Date AS Year, AVG(`S&P500`) AS AvgSP500
FROM `cleaned dataset financial analysis`
GROUP BY YEAR
ORDER BY Year;
```

  - **Identifying Trends and Insights**

```
SELECT Date, `S&P500`,
  (LAG(`S&P500`) OVER (ORDER BY Date) - `S&P500`) / LAG(`S&P500`) OVER (ORDER BY Date) * 100 AS PercentChange
FROM `cleaned dataset financial analysis`
ORDER BY PercentChange DESC
LIMIT 10;
```

  - **Stock Price Analysis**

```
SELECT Symbol AS StockSymbol,
    (MAX(Close) - MIN(Close)) / MIN(Close) * 100 AS GrowthPercent
FROM `cleaned dataset financial analysis`
GROUP BY Symbol
```

```
ORDER BY GrowthPercent DESC
LIMIT 5;
```

- **Industry-Wise Performance**

```
SELECT Industry, SUM(MarketCap) AS TotalMarketCap
FROM `cleaned dataset financial analysis`
GROUP BY Industry
ORDER BY TotalMarketCap DESC;
```

- **Stock Performance Analysis**

```
SELECT Symbol,
    Date,
    (Close - Open) AS DailyReturn,
    (High - Low) AS DailyVolatility
FROM `cleaned dataset financial analysis`
WHERE Date BETWEEN '01-01-2023' AND '31-2023-12'
limit 10;
```