**Data-Driven Stock Analysis: Organizing, Cleaning, and Visualizing Market Trends**

**Overview**

This project focuses on developing a data-driven solution for analyzing and visualizing stock performance trends for the Nifty 50 over the past year. It provides insights into stock performance, market trends, and sector-based analysis to aid investors, analysts, and enthusiasts in making informed decisions.

**Domain**

Finance / Data Analytics

**Problem Statement**

The Stock Performance Dashboard provides a comprehensive analysis of the Nifty 50 stocks’ performance using daily stock data, including metrics like open, close, high, low, and volume values. The project cleans and processes data, generates performance insights, and visualizes top-performing stocks in terms of price changes and average metrics. Interactive dashboards are created using Streamlit and Power BI to deliver actionable insights.

**Business Use Cases**

1. **Stock Performance Ranking**: Identify the top 10 best-performing (green) and worst-performing (red) stocks over the past year.
2. **Market Overview**: Summarize overall market performance, including average metrics and the percentage of green vs. red stocks.
3. **Investment Insights**: Highlight consistently growing stocks and those with significant declines.
4. **Decision Support**: Provide metrics like average prices, volatility, and stock behavior trends for retail and institutional traders.

**Approach**

**Data Extraction and Transformation**

* **Source Format**: Data provided in YAML format organized by months.
* **Processing Steps**:
  1. Extract data from YAML files.
  2. Transform it into CSV format, organized by stock symbols.
  3. Generate 50 individual CSV files, one for each stock symbol.

import yaml

import pandas as pd

from pathlib import Path

def yaml\_to\_csv(input\_file, output\_file):

# Load YAML file

with open(input\_file, 'r') as file:

data = yaml.safe\_load(file)

# Check if the YAML data is list-like (for conversion to DataFrame)

if isinstance(data, list):

df = pd.DataFrame(data)

elif isinstance(data, dict): # Handle dictionary-type YAML

df = pd.DataFrame([data])

else:

raise ValueError("Unsupported YAML structure for CSV conversion.")

# Write to CSV

Path(output\_file).parent.mkdir(parents=True, exist\_ok=True)

df.to\_csv(output\_file, index=False)

print(f"CSV file created at: {output\_file}")

# Example usage

input\_yaml = "path/to/input.yaml"

output\_csv = "path/to/output.csv"

yaml\_to\_csv(input\_yaml, output\_csv)

**Data Analysis and Visualization Requirements**

**1. Python DataFrame for Key Metrics**

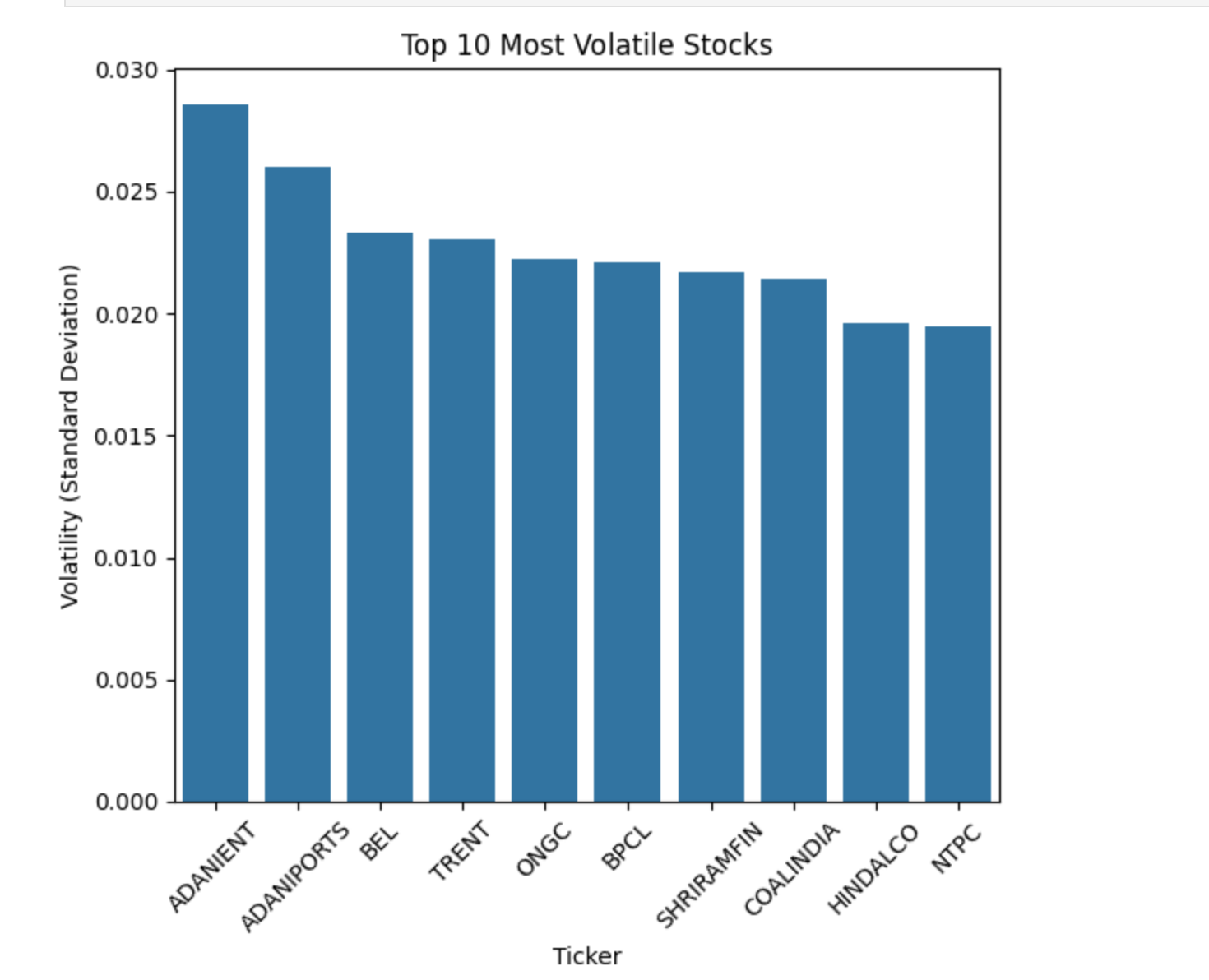
* **Top 10 Green Stocks**: Sort stocks by yearly return and select the top 10.
* **Top 10 Loss Stocks**: Sort stocks by yearly return and select the bottom 10.
* **Market Summary**:
  + Calculate the total number of green vs. red stocks.
  + Compute average stock price and average volume across all stocks.
* **Green Stocks Count**: This counts how many stocks had positive yearly returns (yearly\_return > 0).
* **Red Stocks Count**: This counts how many stocks had negative yearly returns (yearly\_return < 0).

**2. Volatility Analysis**

* **Objective**: Measure stock price fluctuations.
* **Metric**: Standard deviation of daily returns.
* **Visualization**: Bar chart showing volatility of the top 10 most volatile stocks.

**daily returns** = (Close Price - Previous Close Price) / Previous Close Price

volatility= **standard deviation** of daily returns for each stock



**3. Cumulative Return Over Time**

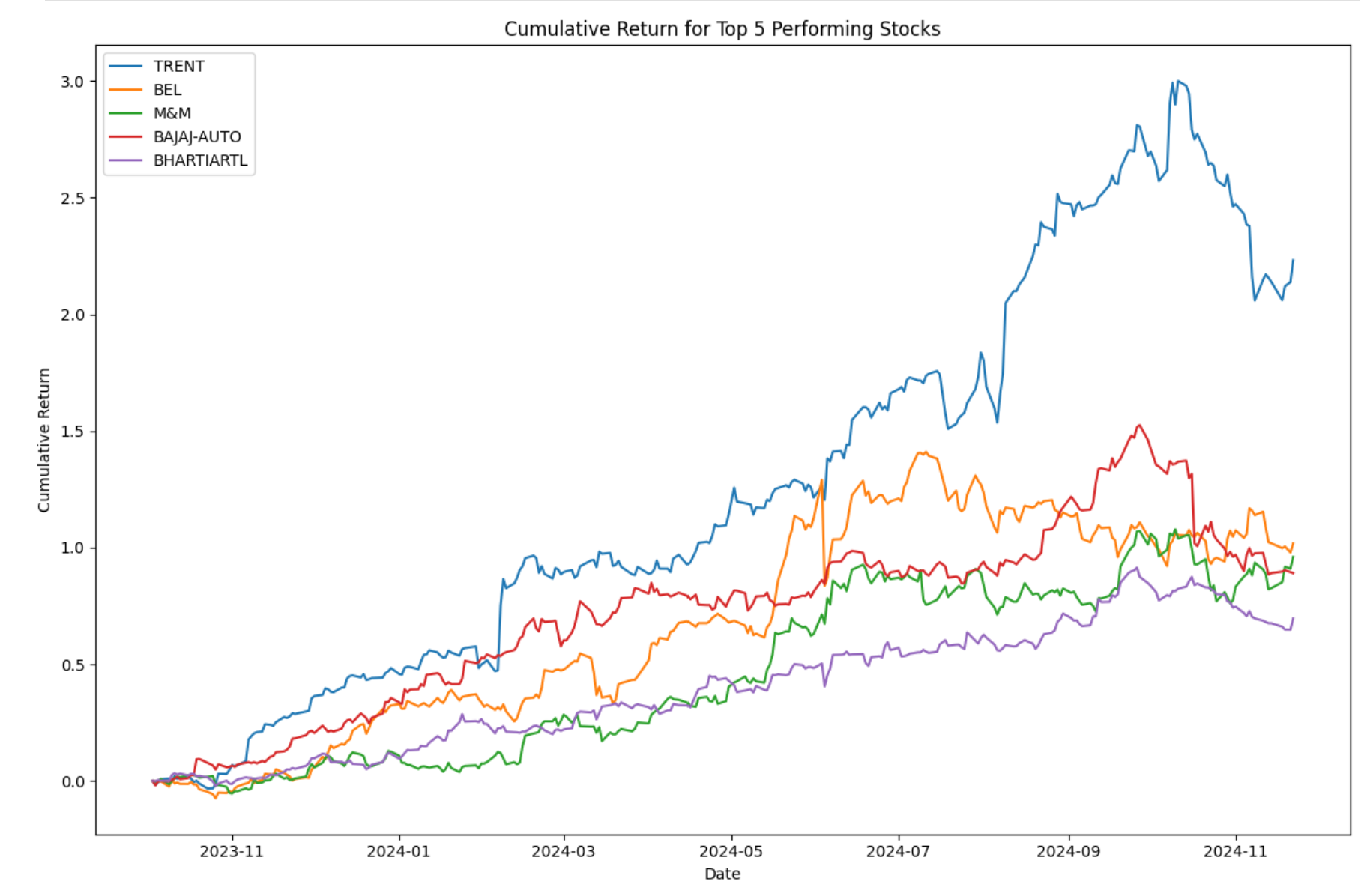
* **Objective**: Display stock growth trends.
* **Metric**: Cumulative return for each stock.
* **Visualization**: Line chart for the top 5 performing stocks.

cumulative return for each stock ticker based on its daily returns. The cumulative return is essentially the overall return if you had invested in the stock from the start date to the current date.

df['cumulative\_return'] = (df.groupby('Ticker')['daily\_return']

.apply(lambda x: (1 + x).cumprod() - 1)

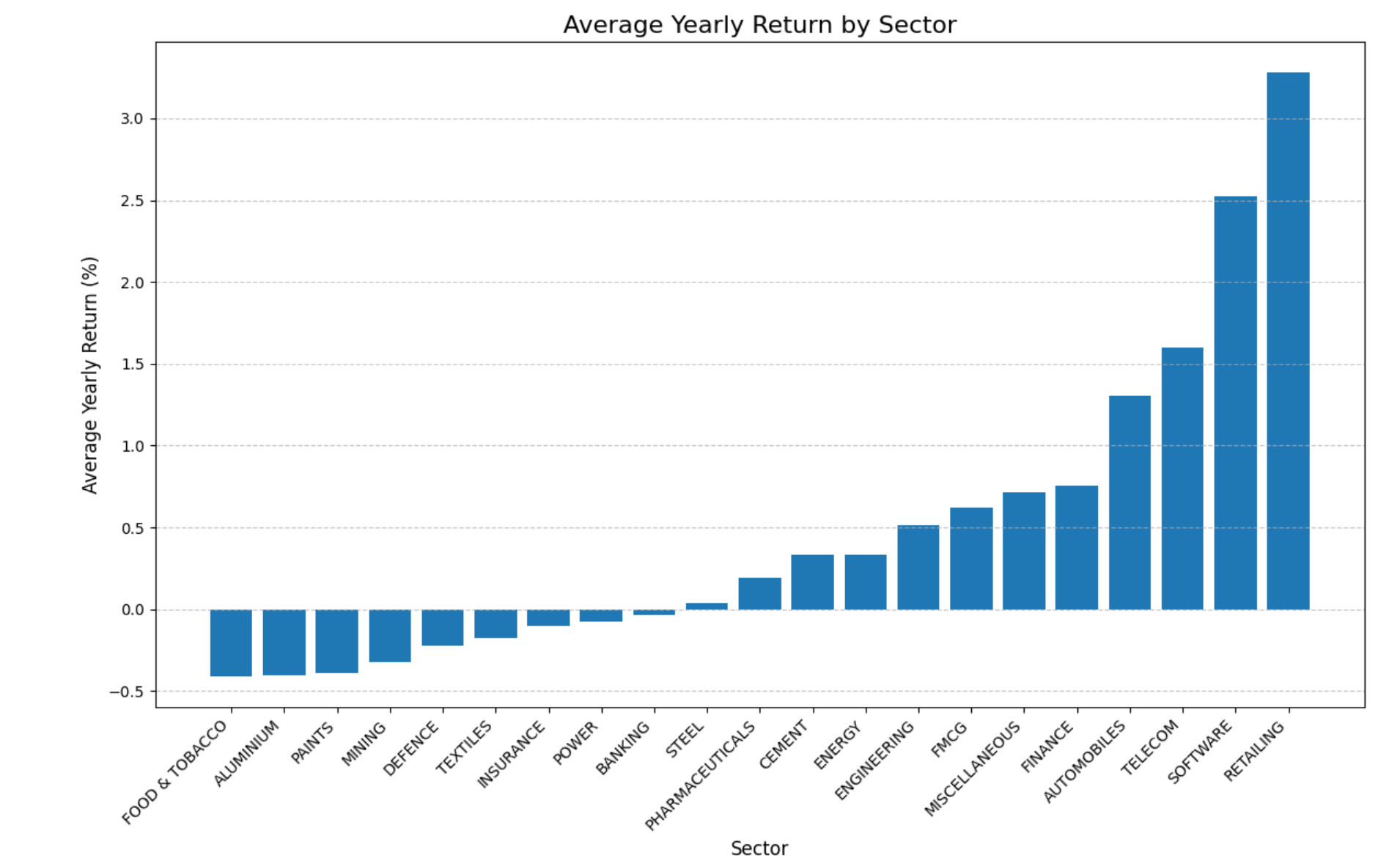
.reset\_index(level=0, drop=True))



* 1. **Sector-wise Performance**
* **Objective**: Analyze stock performance by sector.
* **Metric**: Average yearly return by sector.
* **Visualization**: Bar chart of average sector returns.

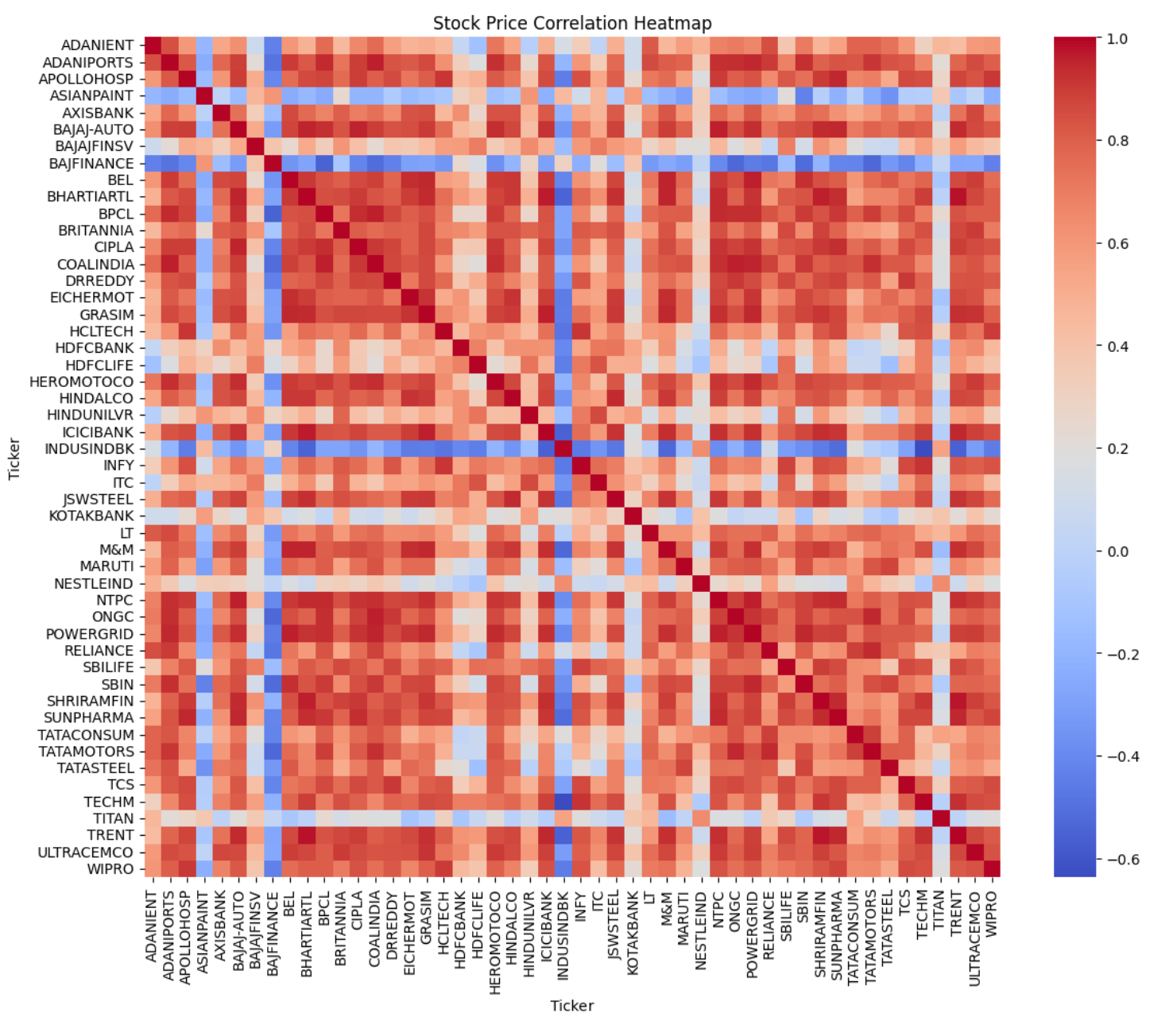
**sector\_avg\_returns = yearly\_returns.groupby('sector')['yearly\_return'].mean().sort\_values()**

For each sector, it calculates the average yearly return of all the stocks in that sector.



**5. Stock Price Correlation**

* **Objective**: Understand stock relationships.
* **Metric**: Correlation coefficients between stock prices.
* **Visualization**: Heatmap of stock price correlations.

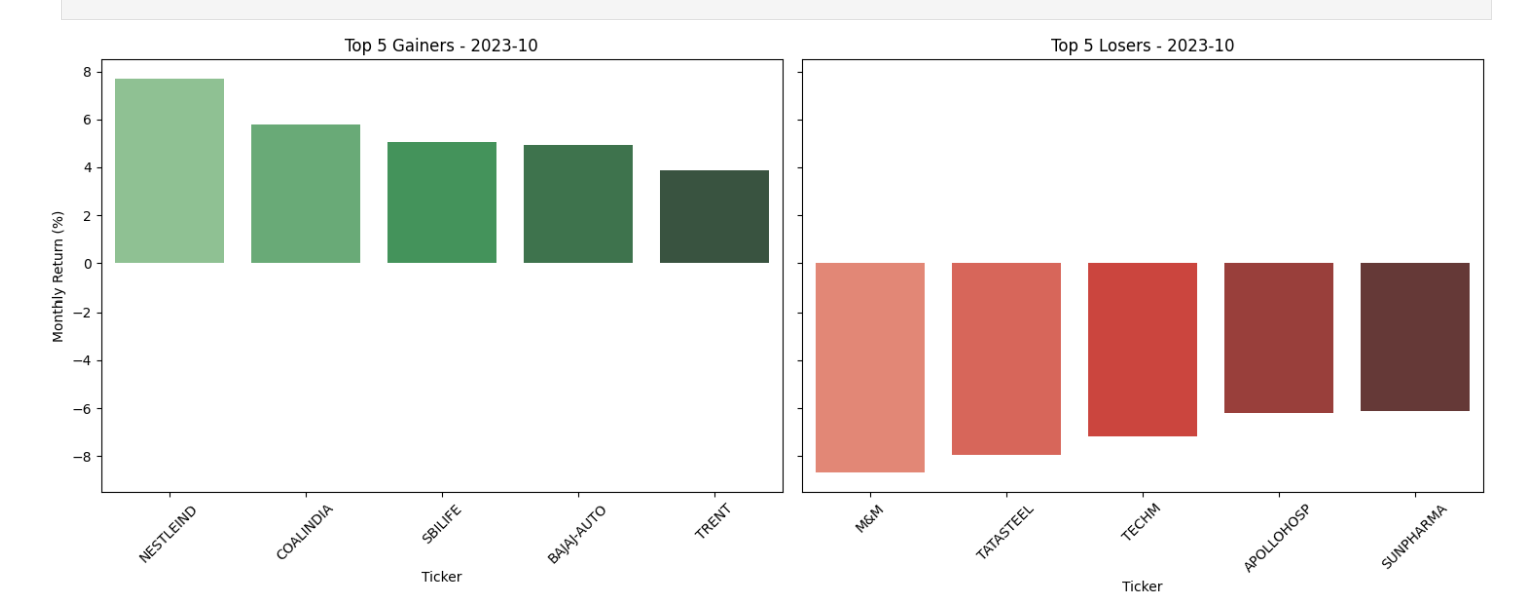


**6. Top 5 Gainers and Losers (Month-wise)**

* **Objective**: Identify monthly trends.
* **Metric**: Monthly returns for top gainers and losers.
* **Visualization**: Bar charts for top 5 gainers and losers across 12 months.

** top\_5\_gainers = month\_data.nlargest(5, 'monthly\_return'): This line finds the top 5 stocks with the highest monthly returns for the current month.**

** top\_5\_losers = month\_data.nsmallest(5, 'monthly\_return'): This line finds the bottom 5 stocks with the lowest monthly returns for the current month.**



**Tools and Technologies**

* **Languages**: Python
* **Database**: MySQL/PostgreSQL
* **Visualization Tools**: Streamlit, Power BI
* **Libraries**: Pandas, Matplotlib, SQLAlchemy

**SQL Database Integration**

**Approach:**Use Python's sqlite3 or another SQL database connector (like mysql-connector-python for MySQL) to store the scraped data.

**Connecting to the MySQL Database:**

The script establishes a connection to the MySQL database using pymysql.

# Step 2: Connect to MySQL

    conn **=** pymysql**.**connect(

host**=**'localhost',

user**=**'root',

password**=**'Abcd1234',

database**=**'stock\_analysis'

)

cursor **=** conn**.**cursor()

**Creating the Database Schema:**

The script creates a table named stock\_data1if it doesn't already exist.

*# Table name*

table\_name **=** "stock\_data1"

create\_table\_query **=** f"""

CREATE TABLE IF NOT EXISTS {table\_name} (

id INT AUTO\_INCREMENT PRIMARY KEY,

Ticker VARCHAR(10),

close FLOAT,

date DATE,

high FLOAT,

low FLOAT,

open FLOAT,

volume BIGINT,

daily\_return FLOAT,

cumulative\_return FLOAT,

yearly\_return FLOAT,

volatility FLOAT,

monthly\_return FLOAT,

company VARCHAR(50),

sector VARCHAR(50),

year INT

);

"""

cursor**.**execute(create\_table\_query)

**Inserting Data into the Database:**

* **insert\_query = f"""**
* **INSERT INTO {table\_name} (**
* **Ticker, close, date, high, low, open, volume, daily\_return,cumulative\_return,**
* **yearly\_return,volatility, monthly\_return, company, sector, year**
* **) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)**
* **"""**

**Committing the Transaction and Closing the Connection:**

After inserting all the data, the transaction is committed to the database to ensure the data is saved.

* # Commit the transaction and close the connection
* conn.commit()

The database connection is then closed to free up resources.

* conn.close()

**Streamlit App Development**

The code that is supplied creates a Streamlit application that can retrieve b data from a MySQL database, let users filter the data according to different standards, and then show the data that has been filtered. A download button to export the filtered data as a CSV file is also provided by the program. A thorough description of the code's operation may be found below

* Importing Required Libraries:
* import streamlit as st
* from sqlalchemy import create\_engine
* import pandas as pd
* Database Connection Using SQLAlchemy:
* engine = create\_engine('mysql+pymysql://root:Abcd1234@127.0.0.1:3306/ stock\_analysis')

**# Function to fetch data from MySQL**

@st.cache\_data

def fetch\_data():

query = "SELECT \* FROM stock\_data1"

df = pd.read\_sql(query, engine)

return df

**References**

* [Streamlit Documentation](https://docs.streamlit.io/library/api-reference)
* [Power BI Documentation](https://docs.microsoft.com/en-us/power-bi/)

**Results:**

A functional dashboard displaying top-performing and worst-performing stocks over the last year.Clear insights into overall market trends.Interactive visualizations using Streamlit and Power BI for easy data access and interpretation.

