Introduction

The Nasdaq-100 is a stock market index that includes a selection of 102 equity securities issued by 101 of the largest nonfinancial companies listed on the Nasdaq stock exchange. This index represents a diverse range of sectors, including manufacturing, technology, retail, telecommunication, biotechnology, health care, transportation, media, and service providers.

Key Points about the Nasdaq-100 Index

- **Composition:** The Nasdaq-100 is comprised of 102 stocks from various sectors, with a significant focus on technology-related companies. It does not include financial companies like banks and insurance firms.
- **Market Capitalization:** The index is weighted by market capitalization, meaning that larger companies have a more significant impact on the index's value.
- **Tech Heavy:** It is often considered a tech-heavy index due to the inclusion of prominent technology companies like Apple, Amazon, Microsoft, and others.
- **Diversity:** While it has a technology emphasis, the Nasdaq-100 is diversified across different sectors, which can reduce risks associated with investing in a single industry.
- **Global Reach:** Many of the companies in the Nasdaq-100 have a global presence and are leaders in their respective industries.
- **Investment and Trading:** The Nasdaq-100 is a popular choice for investors and traders looking to gain exposure to a broad range of innovative and growth-oriented companies.
- **Volatility:** Given its composition of technology and high-growth stocks, the Nasdaq-100 can experience higher volatility compared to other indices like the S&P 500.
- **Benchmark:** It is often used as a benchmark for the performance of technology and growth stocks.

Investors and financial professionals use the Nasdaq-100 as a reference point to gauge the performance of the technology sector and the broader market. It is also a basis for various financial products, including ETFs (Exchange-Traded Funds) that track its performance.

Section 1: Applied Data Science With Python

Question1

Check the stock symbols of the companies in `Nasdaq 100 Market cap.xlsx`. Only the relevant files in the `NASDAQ_DATA` folder should be read.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.arima.model import ARIMA
import pmdarima as pm
from pmdarima.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error
```

Question 2

Append all files (imported in the previous step) that contain no more than 10 years of data. For this, you may use your discretion.

Question 3

Read `Nasdaq 100 market cap.xlsx` and `nasdaq100_metrics_ratios.xlsx`.

```
metrics_ration=pd.read_excel(r"D:\Share Market Analysis Dataset\
nasdaq100_metrics_ratios.xlsx")
market_cap=pd.read_excel(r"D:\Share Market Analysis Dataset\Nasdaq 100
Market cap.xlsx")

metrics_ration.shape

(102, 283)

market_cap.shape

(102, 6)

#### Perform the join
result = pd.merge(metrics_ration, market_cap, left_on='symbol',
right_on='Symbol')

#### Drop the duplicate 'Symbol' column
result = result.drop('Symbol', axis=1)

result.shape

(102, 288)

strings = ['AAPL', 'ABNB', 'ADBE', 'ADI', 'ADP', 'ADSK', 'AEP',
'ALGN',
```

```
        'AMAT', 'AMD', 'AMGN', 'AMZN', 'ANSS', 'ASML', 'ATVI', 'AVGO',
        'AZN', 'BIDU', 'BIIB', 'BKNG', 'CDNS', 'CEG', 'CHTR', 'CMCSA',
        'COST', 'CPRT', 'CRWD', 'CSCO', 'CSX', 'CTAS', 'CTSH', 'DDOG',
        'DLTR', 'DOCU', 'DXCM', 'EA', 'EBAY', 'EXC', 'FAST', 'FISV',
        'FTNT', 'GILD', 'GOOG', 'GOOGL', 'HON', 'IDXX', 'ILMN', 'INTC',
        'INTU', 'ISRG', 'JD', 'KDP', 'KHC', 'KLAC', 'LCID', 'LRCX',
'LULU',
        'MAR', 'MCHP', 'MDLZ', 'MELI', 'META', 'MNST', 'MRNA', 'MRVL',
        'MSFT', 'MTCH', 'MU', 'NFLX', 'NTES', 'NVDA', 'NXPI', 'ODFL',
        'OKTA', 'ORLY', 'PANW', 'PAYX', 'PCAR', 'PDD', 'PEP', 'PYPL',
        'QCOM', 'REGN', 'ROST', 'SBUX', 'SGEN', 'SIRI', 'SNPS', 'SPLK',
        'SWKS', 'TEAM', 'TMUS', 'TSLA', 'TXN', 'VRSK', 'VRSN', 'VRTX',
        'WBA', 'WDAY', 'XEL', 'ZM', 'ZS']

import glob
file_names=glob.glob(r"D:\Share Market Analysis Dataset\NASDAQ_DATA\
*.csv")

matching_files = [file for file in file_names for string in strings if
string in file]

len(matching_files)

117

import os
```

Question 4

Collate the two files imported in the previous step to include the fields Market cap and Last sale in addition to the various metrics and ratios already present in nasdaq100_metrics_ratios.xlsx.

```
#### Initialize an empty DataFrame
all_data = pd.DataFrame()

#### Iterate over each file path
for filepath in matching_files:
    #### Read the data from the file
    data = pd.read_csv(filepath)

    #### Get the filename without extension
    filename_without_extension =
os.path.splitext(os.path.basename(filepath))[0]

    #### Add filename as a new column
    data['filename'] = filename_without_extension

    #### Append the data to the all_data DataFrame
    all_data = pd.concat([all_data, data], ignore_index=True)
```

```
all_data.columns
```

```
Index(['Date', 'High', 'Low', 'Open', 'Close', 'Volume', 'Adj Close',
       'filename', 'Adjusted Close'],
      dtype='object')
```

```
result.columns
```

```
Index(['symbol', 'company', 'sector', 'subsector',
'asset_turnover_2017',
       'asset_turnover_2018', 'asset_turnover_2019',
'asset_turnover_2020',
       'asset_turnover_2021', 'asset_turnover_2022',
       ...
       'yoy_revenue_growth_2019', 'yoy_revenue_growth_2020',
       'yoy_revenue_growth_2021', 'yoy_revenue_growth_2022',
       'yoy_revenue_growth_latest', 'Name', 'Market Cap', 'Last Sale',
       'Net Change', 'Percentage Change'],
      dtype='object', length=288)
```

```
merged_data = pd.merge(result,all_data,left_on='symbol',
right_on='filename', how='inner')
```

```
merged_data.shape,result.shape
```

```
((484476, 297), (102, 288))
```

```
merged_data.drop_duplicates(inplace=True)
```

```
merged_data.shape
```

```
(474247, 297)
```

Question 5

Identify the variables whose variance is less than .005 (as these do not contribute to model building), and eliminate those variables.

```
filtered_data=merged_data
# Filter the dataframe to include only numerical columns
numerical_cols = filtered_data.select_dtypes(include='number').columns

# Calculate variance for each numerical variable
variance = filtered_data[numerical_cols].var()

# Identify variables with variance less than 0.005
low_variance_cols = variance[variance < 0.005].index

# Drop low variance variables
filtered_data = filtered_data.drop(low_variance_cols, axis=1)
```

```
low_variance_cols
```

```
Index(['capex_to_revenue_2022', 'inventory_to_revenue_2017',
       'inventory_to_revenue_2018', 'inventory_to_revenue_2022',
       'Percentage Change'],
      dtype='object')
```

We removed the columns 'capex_to_revenue_2022,' 'inventory_to_revenue_2017,' 'inventory_to_revenue_2018,' 'inventory_to_revenue_2022,' and 'Percentage Change' from the dataset because they had low variance, meaning their values didn't vary much and didn't provide much useful information for analysis.

```
#### Convert 'Date' to datetime format
merged_data['Date'] = pd.to_datetime(merged_data['Date'])

#### Get the latest date in the dataset
latest_date = merged_data['Date'].max()

#### Calculate the date 10 years ago from the latest date
date_10_years_ago = latest_date - pd.DateOffset(years=10)

#### Filter the data for the last 10 years
filtered_data = merged_data[merged_data['Date'] > date_10_years_ago]
```

```
filtered_data.shape
```

```
(236022, 297)
```

```
filtered_data.to_csv("test.csv")
```

```
filtered_data.shape
```

```
(236022, 297)
```

Question 6

Delete the variables in `nasdaq100_metrics_ratios.xlsx` where 30% or more of the values are missing.

Question 7

Perform missing value imputation for variables with less than 30% missing values by considering the company's sector. secto

```
#### Calculate the percentage of missing values for each variable
missing_percent = filtered_data.isnull().mean() * 100
```

```
#### Identify variables with 30% or more missing values
cols_to_drop = missing_percent[missing_percent >= 30].index

#### Drop these variables
filtered_data = filtered_data.drop(cols_to_drop, axis=1)

#### Identify variables with less than 30% missing values
cols_to_impute = missing_percent[(missing_percent > 0) &
(missing_percent < 30)].index

#### Impute missing values based on the company's sector
for col in cols_to_impute:
    filtered_data[col] = filtered_data.groupby('sector')
[col].transform(lambda x: x.fillna(x.mean()))

filtered_data.shape
```

```
(236022, 227)
```

Question 8

Analyze the effect of COVID on stock prices in detail, create visuals to support the insights, and address the following:

8.a

Which sectors and companies saw the greatest impact, and which ones saw the least? You may use growth or degrowth as a measure of impact and may perform week over week, month over month (MoM), quarter over quarter (QoQ), or year over year (YoY) analysis as appropriate.

8.b

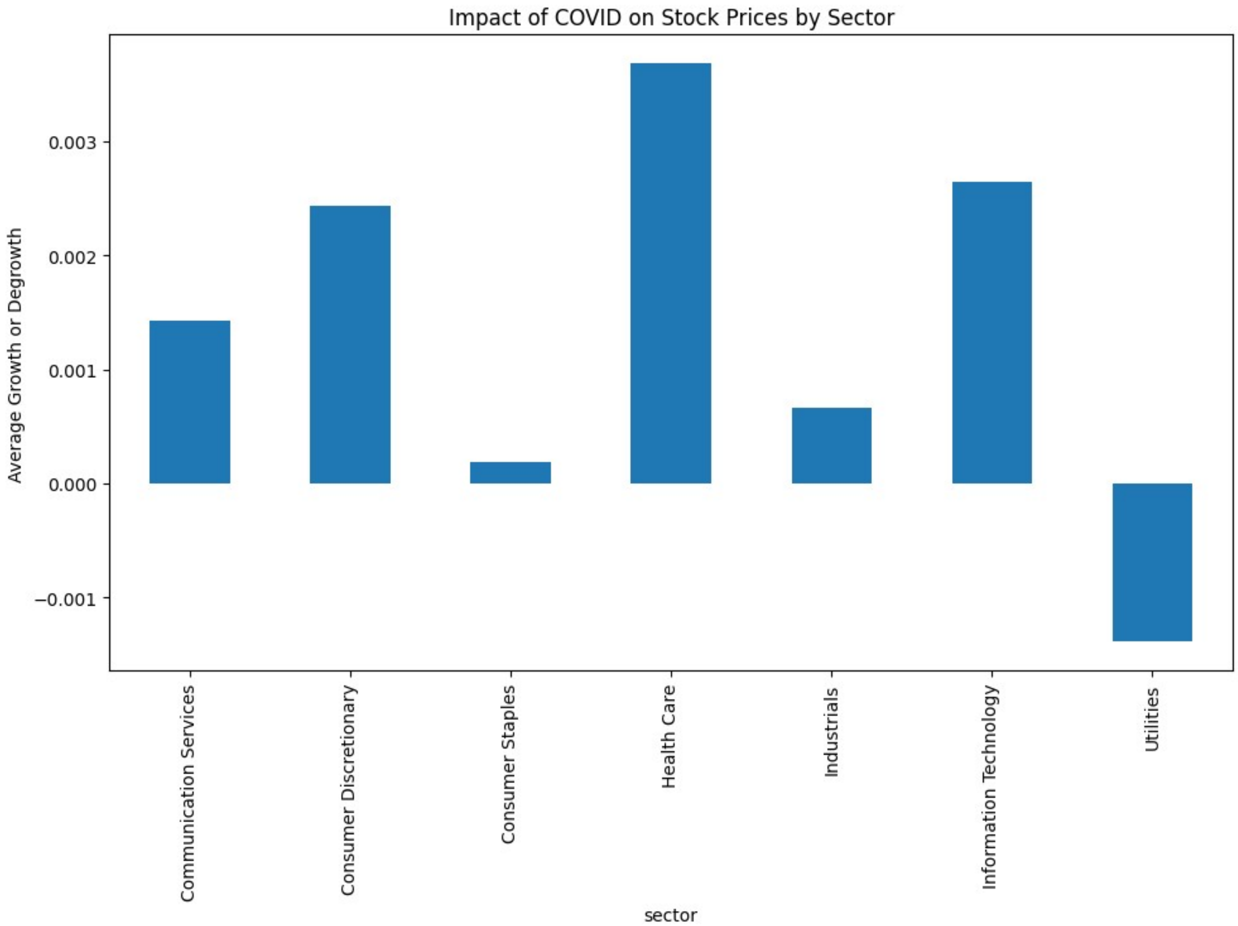
Which sector and company experienced the fastest and slowest recoveries?

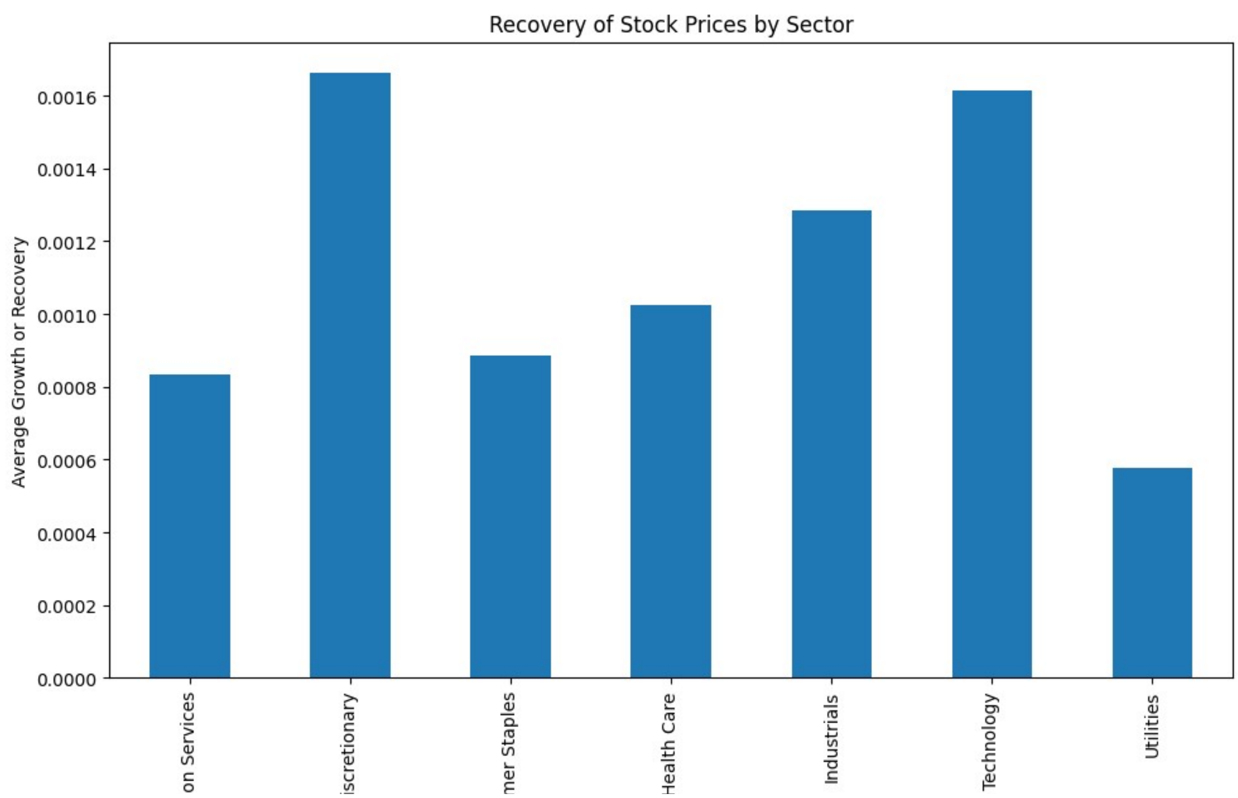
```
#### Convert the date column to datetime format
filtered_data['Date'] = pd.to_datetime(filtered_data['Date'])

#### Define the COVID impact and recovery periods
covid_start = '2020-02-01'
covid_end = '2020-06-30'
recovery_start = '2020-07-01'
recovery_end = '2021-12-31'

#### Filter data for the COVID impact period
covid_data = filtered_data[(filtered_data['Date'] >= covid_start) &
(filtered_data['Date'] <= covid_end)]
```

```
#### Calculate the growth or degrowth in stock prices during the COVID impact period
covid_data['price_change'] = covid_data.groupby(['symbol', 'sector'])
['Close'].pct_change()

#### Aggregate the data to get the average growth or degrowth for each sector and company
covid_agg = covid_data.groupby(['sector', 'symbol'])
['price_change'].mean().reset_index()

#### Identify the sectors and companies with the greatest and least impact
greatest_impact_sector = covid_agg.groupby('sector')
['price_change'].mean().idxmin()
least_impact_sector = covid_agg.groupby('sector')
['price_change'].mean().idxmax()
greatest_impact_company = covid_agg['price_change'].idxmin()
least_impact_company = covid_agg['price_change'].idxmax()

#### Filter data for the recovery period
recovery_data = filtered_data[(filtered_data['Date'] >=
recovery_start) & (filtered_data['Date'] <= recovery_end)]

#### Calculate the growth or recovery in stock prices during the recovery period
recovery_data['price_change'] = recovery_data.groupby(['symbol',
'sector'])['Close'].pct_change()

#### Aggregate the data to get the average growth or recovery for each sector and company
recovery_agg = recovery_data.groupby(['sector', 'symbol'])
['price_change'].mean().reset_index()

#### Identify the sectors and companies with the fastest and slowest recovery
fastest_recovery_sector = recovery_agg.groupby('sector')
['price_change'].mean().idxmax()
slowest_recovery_sector = recovery_agg.groupby('sector')
['price_change'].mean().idxmin()
fastest_recovery_company = recovery_agg['price_change'].idxmax()
slowest_recovery_company = recovery_agg['price_change'].idxmin()

#### Create visuals to support the insights
fig, ax = plt.subplots(2, 1, figsize=(10, 15))

#### Bar plot for impact on sectors
covid_agg.groupby('sector')['price_change'].mean().plot(kind='bar',
ax=ax[0])
ax[0].set_title('Impact of COVID on Stock Prices by Sector')
ax[0].set_ylabel('Average Growth or Degrowth')
```

```
#### Bar plot for recovery of sectors
recovery_agg.groupby('sector')['price_change'].mean().plot(kind='bar',
ax=ax[1])
ax[1].set_title('Recovery of Stock Prices by Sector')
ax[1].set_ylabel('Average Growth or Recovery')

plt.tight_layout()
plt.show()
```

```
C:\Users\bhara\AppData\Local\Temp\ipykernel_1476\200252812.py:14:
PerformanceWarning: DataFrame is highly fragmented.  This is usually
the result of calling `frame.insert` many times, which has poor
performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe
= frame.copy()`
  covid_data['price_change'] = covid_data.groupby(['symbol',
'sector'])['Close'].pct_change()
C:\Users\bhara\AppData\Local\Temp\ipykernel_1476\200252812.py:14:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  covid_data['price_change'] = covid_data.groupby(['symbol',
'sector'])['Close'].pct_change()
C:\Users\bhara\AppData\Local\Temp\ipykernel_1476\200252812.py:29:
PerformanceWarning: DataFrame is highly fragmented.  This is usually
the result of calling `frame.insert` many times, which has poor
performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe
= frame.copy()`
  recovery_data['price_change'] = recovery_data.groupby(['symbol',
'sector'])['Close'].pct_change()
C:\Users\bhara\AppData\Local\Temp\ipykernel_1476\200252812.py:29:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  recovery_data['price_change'] = recovery_data.groupby(['symbol',
'sector'])['Close'].pct_change()
```


Impact of COVID on Stock Prices by Sector
Average Growth or Degrowth
0.003
0.002
0.001
0.000
−0.001
Communication Services
Consumer Discretionary
Consumer Staples
Health Care
Industrials
Information Technology
Utilities
sector


Recovery of Stock Prices by Sector
Average Growth or Recovery
0.0016
0.0014
0.0012
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
on Services
iscretionary
mer Staples
Health Care
Industrials
Technology
Utilities

```
filtered_data.head()
```

```
     symbol     company                  sector  \
3228   AAPL  Apple Inc.  Information Technology
3229   AAPL  Apple Inc.  Information Technology
3230   AAPL  Apple Inc.  Information Technology
3231   AAPL  Apple Inc.  Information Technology
3232   AAPL  Apple Inc.  Information Technology

                                       subsector  asset_turnover_2017  
\
3228  Technology Hardware, Storage & Peripherals                 0.66

3229  Technology Hardware, Storage & Peripherals                 0.66

3230  Technology Hardware, Storage & Peripherals                 0.66

3231  Technology Hardware, Storage & Peripherals                 0.66

3232  Technology Hardware, Storage & Peripherals                 0.66


      asset_turnover_2018  asset_turnover_2019  asset_turnover_2020  \
3228                 0.72                 0.74                 0.83
3229                 0.72                 0.74                 0.83
3230                 0.72                 0.74                 0.83
3231                 0.72                 0.74                 0.83
3232                 0.72                 0.74                 0.83

      asset_turnover_2021  asset_turnover_latest  ...  Net Change  \
3228                 1.08                   0.24  ...       $2.00
3229                 1.08                   0.24  ...       $2.00
3230                 1.08                   0.24  ...       $2.00
3231                 1.08                   0.24  ...       $2.00
3232                 1.08                   0.24  ...       $2.00

      Percentage Change       Date       High        Low       Open  \
3228             0.0134 2012-10-31  21.498571  20.989286  21.245714
3229             0.0134 2012-11-01  21.535713  21.220358  21.365000
3230             0.0134 2012-11-02  21.319643  20.526787  21.281786
3231             0.0134 2012-11-05  20.991785  20.628571  20.840000
3232             0.0134 2012-11-06  21.097857  20.717501  21.079643

          Close       Volume  Adj Close  filename
3228  21.261429  510003200.0  18.201437      AAPL
3229  21.305000  361298000.0  18.238741      AAPL
3230  20.600000  599373600.0  17.635199      AAPL
3231  20.879286  529135600.0  17.874296      AAPL
3232  20.816071  374917200.0  17.820169      AAPL

[5 rows x 227 columns]
```

Section 2: Machine Learning

1 Perform PCA to reduce the number of variables in the data.

```
# Assuming you have a DataFrame named 'filtered_data' with the
specified columns

# Convert the 'Close' column to string, remove non-numeric characters,
and convert to float
filtered_data['Close'] = filtered_data['Close'].astype(str)
filtered_data['Close'] = filtered_data['Close'].str.replace('$', '',
regex=True)
filtered_data['Close'] = pd.to_numeric(filtered_data['Close'],
errors='coerce')
import pandas as pd

# Assuming you have a DataFrame named 'filtered_data'

# Loop through all columns in the DataFrame
for column in filtered_data.columns:
    # Check if the column contains a dollar symbol ($) and if it's not
already a numeric type
    if '$' in filtered_data[column].astype(str).str.extract('(\$\d+\.\
d+)')[0].values:
        # Remove dollar symbols and convert the column to float
        filtered_data[column] = filtered_data[column].str.replace('$',
'', regex=False).astype(float)
# List of columns to process
columns_to_process = ['Last Sale', 'Net Change']

filtered_data = filtered_data.replace('[\$,]', '', regex=True)
for col in filtered_data.columns:
    if filtered_data[col].dtype == 'object':
        # If column is of object type, fill missing values with mode
        mode_val = filtered_data[col].mode()[0]
        filtered_data[col].fillna(mode_val, inplace=True)
    else:
        # If column is of numerical type, fill missing values with
median
        median_val = filtered_data[col].median()
        filtered_data[col].fillna(median_val, inplace=True)
# Loop through columns and remove dollar symbols and commas
for column in columns_to_process:
    filtered_data[column] = filtered_data[column].str.replace('$',
'').str.replace(',', '').astype(float)

# Now, the specified columns should have dollar symbols and commas
removed and be converted to float
```

```python
# Drop non-numeric columns
numerical_data = filtered_data.drop(columns=['symbol', 'company',
'sector', 'subsector', 'Name', 'filename', 'Date'])

# Handle missing values by filling them with the mean value of each
column
numerical_data.fillna(numerical_data.mean(), inplace=True)

# Standardize the data (mean=0, std=1)
scaler = StandardScaler()
data_std = scaler.fit_transform(numerical_data)

# Apply PCA with no component number restriction
pca_all = PCA()
data_pca_all = pca_all.fit_transform(data_std)

# Find the number of components required to retain 95% of the variance
cumulative_variance_ratio = pca_all.explained_variance_ratio_.cumsum()
n_components_95 = (cumulative_variance_ratio >= 0.95).argmax() + 1

# Apply PCA with the selected number of components
pca = PCA(n_components=n_components_95)
data_pca = pca.fit_transform(data_std)

# Create a DataFrame with the PCA results
data_pca_df = pd.DataFrame(data=data_pca, columns=[f'PC{i}' for i in
range(1, n_components_95 + 1)])

# Print the number of components retained
print('Number of components retained to retain 95% of the variance:',
n_components_95)

# Print the variance ratio of each component
print('Explained variance ratio:', pca.explained_variance_ratio_)

# Print the cumulative explained variance by the components
print('Cumulative explained variance:',
pca.explained_variance_ratio_.cumsum())
```

```
Number of components retained to retain 95% of the variance: 55
Explained variance ratio: [0.1330012  0.07799619 0.05904312 0.05291784
0.04514023 0.04022754
 0.03567015 0.03029314 0.02806933 0.02732899 0.023224   0.02226343
 0.02099559 0.02002561 0.01834272 0.01798933 0.01756261 0.01578869
 0.01522894 0.01391463 0.01277926 0.01206117 0.01191486 0.01165498
 0.01041486 0.00965039 0.00948009 0.00935483 0.00884144 0.00865445
 0.00841754 0.00822238 0.00786466 0.00688673 0.00679613 0.00646869
 0.00642051 0.0061495  0.00590385 0.00551129 0.00523013 0.00518109
 0.00502464 0.00492235 0.00464825 0.00450944 0.00445227 0.00424488
```

```
 0.00394151 0.00374964 0.00371565 0.00338303 0.00332605 0.00310619
 0.00304254]
Cumulative explained variance: [0.1330012  0.21099738 0.2700405
0.32295834 0.36809857 0.40832611
 0.44399627 0.4742894  0.50235873 0.52968772 0.55291172 0.57517515
 0.59617074 0.61619634 0.63453906 0.65252839 0.670091   0.68587969
 0.70110863 0.71502326 0.72780252 0.73986369 0.75177855 0.76343353
 0.77384839 0.78349878 0.79297887 0.8023337  0.81117514 0.81982959
 0.82824713 0.8364695  0.84433416 0.85122089 0.85801702 0.86448571
 0.87090622 0.87705572 0.88295957 0.88847086 0.89370099 0.89888208
 0.90390673 0.90882907 0.91347733 0.91798676 0.92243903 0.92668392
 0.93062542 0.93437506 0.93809071 0.94147374 0.94479979 0.94790598
 0.95094851]
```

Summary

- Number of components retained to retain 95% of the variance: 55
- Explained variance ratio for each of the 55 components:
  - These values represent the proportion of variance explained by each component. The higher the value, the more variance is explained by that component.
  - The first component explains approximately 13.3% of the variance.
  - The second component explains approximately 7.8% of the variance.
  - The third component explains approximately 5.9% of the variance.
  - And so on, with decreasing variance explained by each successive component.
- Cumulative explained variance:
  - This is the cumulative or total variance explained by adding up the individual components.
  - The cumulative explained variance helps us understand how much of the total variance is captured by including a certain number of components.
  - In this case, to retain 95% of the total variance, you would need to include 55 components.
  - The cumulative explained variance gradually increases as more components are added, reaching approximately 95% with the inclusion of these 55 components.

Question 2

After PCA, perform cluster analysis to identify cohorts, define these cohorts (cluster profiling), and specify the insights found.

Question 3

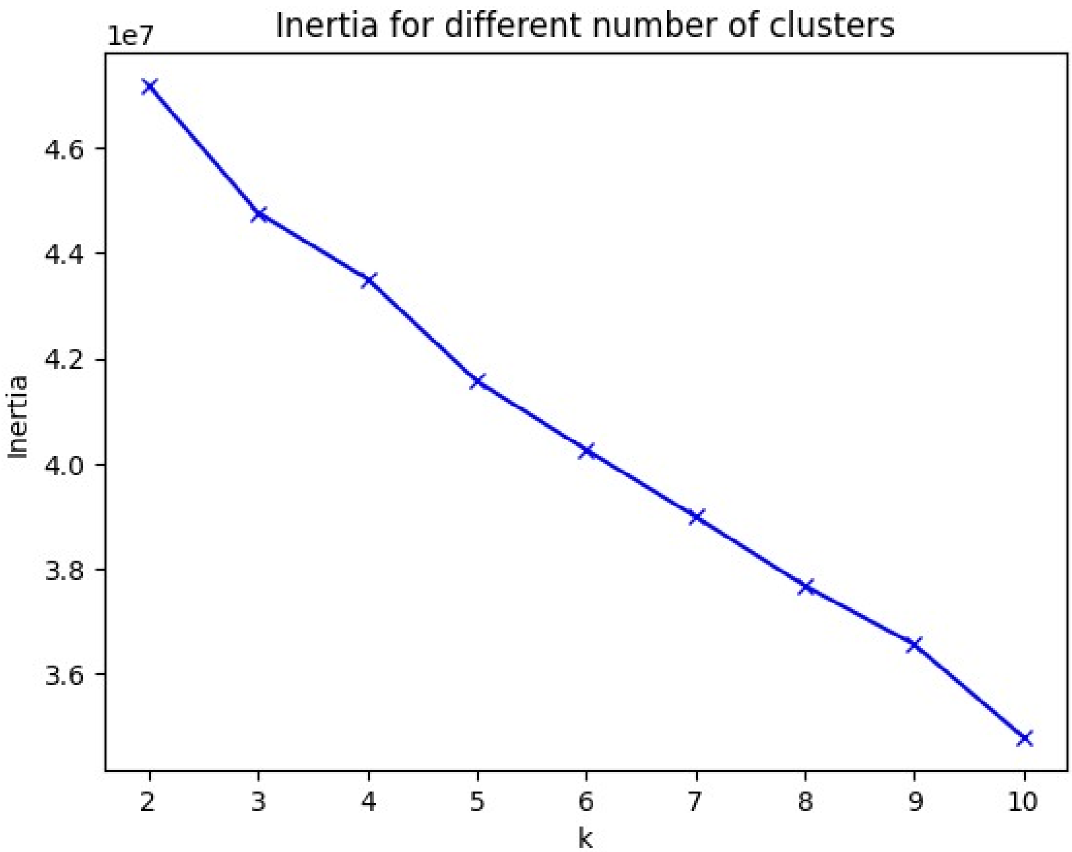
Highlight companies from different sectors falling into the same cohort, and share your findings.

```python
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

filtered_data = filtered_data.replace('[\$,]', '', regex=True)
```

```python
# Standardize the data
numerical_data = filtered_data.drop(columns=['symbol', 'company',
'sector', 'subsector', 'Name', 'filename', 'Date'])
scaler = StandardScaler()
numerical_data_scaled = scaler.fit_transform(numerical_data)

# Determine the optimal number of clusters
inertia = []
K = range(2, 11)
for k in K:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(numerical_data_scaled)
    inertia.append(kmeans.inertia_)

# Plot the inertia
import matplotlib.pyplot as plt
plt.plot(K, inertia, 'bx-')
plt.xlabel('k')
plt.ylabel('Inertia')
plt.title('Inertia for different number of clusters')
plt.show()

# Choose the number of clusters based on the elbow method
optimal_k = int(input("Enter the optimal number of clusters based on
the plot: "))

# Perform k-means clustering with the optimal number of clusters
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
clusters = kmeans.fit_predict(numerical_data_scaled)

# Add cluster labels to the original data
filtered_data['Cluster'] = clusters

# Cluster profiling
for i in range(optimal_k):
    cluster_data = filtered_data[filtered_data['Cluster'] == i]
    print(f"\nCluster {i}:")
    print(cluster_data.describe())
```

```
C:\Users\bhara\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default
value of `n_init` will change from 10 to 'auto' in 1.4. Set the value
of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\bhara\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default
value of `n_init` will change from 10 to 'auto' in 1.4. Set the value
of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```

```
C:\Users\bhara\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default
value of `n_init` will change from 10 to 'auto' in 1.4. Set the value
of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\bhara\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default
value of `n_init` will change from 10 to 'auto' in 1.4. Set the value
of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\bhara\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default
value of `n_init` will change from 10 to 'auto' in 1.4. Set the value
of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\bhara\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default
value of `n_init` will change from 10 to 'auto' in 1.4. Set the value
of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\bhara\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default
value of `n_init` will change from 10 to 'auto' in 1.4. Set the value
of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\bhara\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default
value of `n_init` will change from 10 to 'auto' in 1.4. Set the value
of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\bhara\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default
value of `n_init` will change from 10 to 'auto' in 1.4. Set the value
of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```




```
Enter the optimal number of clusters based on the plot:  5

C:\Users\bhara\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default
value of `n_init` will change from 10 to 'auto' in 1.4. Set the value
of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\bhara\AppData\Local\Temp\ipykernel_1476\1829115453.py:35:
PerformanceWarning: DataFrame is highly fragmented.  This is usually
the result of calling `frame.insert` many times, which has poor
performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe
= frame.copy()`
  filtered_data['Cluster'] = clusters


Cluster 0:
       asset_turnover_2017  asset_turnover_2018
asset_turnover_2019  \
count         72518.000000         72518.000000         72518.000000

mean              0.561683             0.537178             0.530919
```

```
min              0.220000              0.240000              0.220000
25%              0.340000              0.320000              0.310000
50%              0.420000              0.410000              0.400000
75%              0.660000              0.610000              0.650000
max              1.710000              1.960000              1.770000
std              0.347170              0.362933              0.343298

       asset_turnover_2020  asset_turnover_2021  asset_turnover_latest
\
count         72518.000000         72518.000000           72518.000000
mean              0.481713             0.524100               0.139900
min               0.190000             0.200000               0.050000
25%               0.300000             0.320000               0.090000
50%               0.420000             0.420000               0.110000
75%               0.530000             0.710000               0.190000
max               1.580000             1.570000               0.350000
std               0.283483             0.293567               0.074177

       buyback_yield_2018  buyback_yield_2019  buyback_yield_2020  \
count        72518.000000        72518.000000        72518.000000
mean             4.786190            3.341441            1.781781
min            -24.970000           -2.690000           -3.190000
25%              1.280000            1.230000           -0.020000
50%              3.480000            2.484000            1.390000
75%              7.790000            5.280000            2.220000
max             25.030000           16.930000           14.680000
std              8.707862            3.877649            3.337185

       buyback_yield_2021  ...    Net Change  Percentage Change  \
count        72518.000000  ...  72518.000000       72518.000000
mean             2.669988  ...      1.540009           0.013463
min             -1.340000  ...      0.010000          -0.002500
25%              0.040000  ...      0.400000           0.004800
50%              2.080000  ...      0.820000           0.013600
75%              3.056667  ...      1.790000           0.018300
max             17.620000  ...      8.750000           0.047000
std              4.096502  ...      1.938724           0.011793
```

```
                                Date          High           Low  \
count                          72518  72518.000000  72518.000000
mean   2017-11-17 03:46:44.920157952     86.649985     84.806734
min              2012-10-31 00:00:00      2.710000      2.550000
25%              2015-05-29 00:00:00     36.131024     35.466250
50%              2017-11-20 00:00:00     63.630001     62.430000
75%              2020-05-18 00:00:00    101.040001     99.010002
max              2022-10-28 00:00:00    825.619995    817.530029
std                              NaN     88.427590     86.356060

               Open         Close        Volume     Adj Close  Cluster

count  72518.000000  72518.000000  7.251800e+04  72518.000000  72518.0

mean      85.735961     85.747325  6.904498e+06     78.708468      0.0

min        2.640000      2.650000  0.000000e+00      2.363895      0.0

25%       35.813594     35.820000  2.087925e+06     30.612071      0.0

50%       63.029999     63.070000  3.844200e+06     55.068943      0.0

75%      100.070000    100.120003  7.891275e+06     90.639545      0.0

max      823.080017    821.010010  4.056765e+08    821.010010      0.0

std       87.395088     87.392881  9.560989e+06     86.655942      0.0


[8 rows x 222 columns]

Cluster 1:
       asset_turnover_2017  asset_turnover_2018
asset_turnover_2019  \
count         90623.000000         90623.000000         90623.000000

mean              0.861884             0.905200             0.867890

min               0.230000             0.300000             0.290000

25%               0.640675             0.520000             0.540000

50%               0.640675             0.700000             0.620000

75%               1.016481             0.940000             0.800000

max               3.710000             3.670000             3.540000

std               0.602526             0.686774             0.673308
```

```
       asset_turnover_2020  asset_turnover_2021  asset_turnover_latest
\
count         90623.000000         90623.000000           90623.000000

mean              0.812194             0.816251               0.204677

min               0.260000             0.200000               0.050000

25%               0.480000             0.460000               0.120000

50%               0.590000             0.630000               0.170000

75%               0.830000             0.960000               0.250000

max               3.300000             3.410000               0.830000

std               0.614967             0.589761               0.140586


       buyback_yield_2018  buyback_yield_2019  buyback_yield_2020  \
count        90623.000000        90623.000000        90623.000000
mean             3.332830            2.635288            1.837107
min             -6.700000           -6.560000           -5.110000
25%              0.730000            0.190000            0.550000
50%              2.940000            1.800000            1.430000
75%              5.754573            4.140000            1.955939
max             16.230000           13.910000           17.900000
std              4.079243            3.987599            3.394102

       buyback_yield_2021  ...    Net Change  Percentage Change  \
count        90623.000000  ...  90623.000000       90623.000000
mean             1.485979  ...      6.561785           0.026915
min             -3.960000  ...      0.130000          -0.015400
25%              0.720000  ...      1.660000           0.008500
50%              1.510000  ...      2.850000           0.025300
75%              2.658531  ...      7.315000           0.037800
max              5.100000  ...     45.730000           0.066800
std              1.633559  ...      9.906918           0.019669

                                Date          High           Low  \
count                          90623  90623.000000  90623.000000
mean   2018-02-15 17:23:46.962250240    176.377823    171.544548
min              2012-10-31 00:00:00      1.690000      1.610000
25%              2015-09-09 00:00:00     44.400002     43.247499
50%              2018-04-24 00:00:00     89.099998     86.739998
75%              2020-08-18 00:00:00    181.687500    176.235001
max              2022-10-28 00:00:00   2715.659912   2632.219971
std                              NaN    303.617737    295.864383
```

```
              Open         Close        Volume     Adj Close  Cluster
count  90623.000000  90623.000000  9.062300e+04  90623.000000  90623.0
mean     174.005086    173.995813  1.718028e+07    171.447794      1.0
min        1.620000      1.620000  0.000000e+00      1.620000      1.0
25%       43.799999     43.830002  1.276800e+06     40.087200      1.0
50%       87.900002     87.959999  2.677800e+06     84.903564      1.0
75%      178.980003    179.070000  1.081930e+07    175.985001      1.0
max     2680.000000   2703.260010  1.460852e+09   2703.260010      1.0
std      299.863290    299.711606  4.627886e+07    300.133791      0.0

[8 rows x 222 columns]
Cluster 2:
       asset_turnover_2017  asset_turnover_2018
asset_turnover_2019  \
count         69831.000000         69831.000000          69831.000000
mean              0.724879             0.782842              0.751244
min               0.140000             0.080000              0.030000
25%               0.550000             0.550000              0.510000
50%               0.650000             0.640000              0.640000
75%               0.870000             0.920000              0.960000
max               1.570000             1.590000              1.610000
std               0.258528             0.312575              0.327476

       asset_turnover_2020  asset_turnover_2021  asset_turnover_latest
\
count         69831.000000         69831.000000           69831.000000
mean              0.687089             0.710768               0.175384
min               0.060000             0.310000               0.070000
25%               0.530000             0.490000               0.130000
```

50%	0.620000	0.720000	0.150000
75%	0.770000	0.760000	0.200000
max	1.480000	1.460000	0.390000
std	0.281022	0.261902	0.083517

	buyback_yield_2018	buyback_yield_2019	buyback_yield_2020	\
count	69831.000000	69831.000000	69831.000000	
mean	1.422781	1.825195	1.001398	
min	-11.210000	-2.790000	-5.220000	
25%	0.610000	0.610000	0.190000	
50%	1.520000	1.770000	1.090000	
75%	2.897143	2.484000	1.651027	
max	5.290000	6.610000	7.650000	
std	2.737600	1.822117	2.263718	

	buyback_yield_2021	...	Net Change	Percentage Change	\
count	69831.000000	...	69831.000000	69831.000000	
mean	1.370484	...	4.207268	0.016215	
min	-1.810000	...	0.010000	-0.041700	
25%	0.680000	...	0.920000	0.005300	
50%	1.470000	...	2.350000	0.011000	
75%	1.958000	...	6.415000	0.027100	
max	4.830000	...	15.735000	0.063600	
std	1.149635	...	4.187162	0.021240	

	Date	High	Low	\
count	69831	69831.000000	69831.000000	
mean	2017-11-30 21:27:09.333677056	122.245129	118.849391	
min	2012-10-31 00:00:00	2.907500	2.787500	
25%	2015-05-28 00:00:00	43.500000	42.419998	
50%	2017-12-19 00:00:00	75.080002	73.470001	
75%	2020-06-10 12:00:00	157.404999	152.520004	
max	2022-10-28 00:00:00	761.049988	731.450012	
std	NaN	123.412805	119.736824	

	Open	Close	Volume	Adj Close	Cluster
count	69831.000000	69831.000000	6.983100e+04	69831.000000	69831.0
mean	120.585786	120.596341	9.074507e+06	119.478833	2.0
min	2.872500	2.845000	1.210000e+04	2.611672	2.0
25%	42.980000	42.980000	1.168600e+06	41.810579	2.0
50%	74.309998	74.300003	2.483200e+06	72.633209	2.0

```
75%      154.940002    154.939995  6.255800e+06    154.114937      2.0

max      752.559998    752.559998  4.643901e+08    752.559998      2.0

std      121.658382    121.616961  1.745319e+07    121.897363      0.0


[8 rows x 222 columns]

Cluster 3:
       asset_turnover_2017  asset_turnover_2018
asset_turnover_2019  \
count          2517.000000         2.517000e+03         2.517000e+03

mean              0.640675         4.800000e-01         4.600000e-01

min               0.640675         4.800000e-01         4.600000e-01

25%               0.640675         4.800000e-01         4.600000e-01

50%               0.640675         4.800000e-01         4.600000e-01

75%               0.640675         4.800000e-01         4.600000e-01

max               0.640675         4.800000e-01         4.600000e-01

std               0.000000         5.552218e-17         5.552218e-17


       asset_turnover_2020  asset_turnover_2021  asset_turnover_latest
\
count              2517.00              2517.00                2517.00

mean                  0.44                 0.44                   0.13

min                   0.44                 0.44                   0.13

25%                   0.44                 0.44                   0.13

50%                   0.44                 0.44                   0.13

75%                   0.44                 0.44                   0.13

max                   0.44                 0.44                   0.13

std                   0.00                 0.00                   0.00


       buyback_yield_2018  buyback_yield_2019  buyback_yield_2020  \
count        2.517000e+03              2517.0        2.517000e+03
mean         1.400000e+00                 1.5        4.860000e+00
```

```
min          1.400000e+00                 1.5        4.860000e+00
25%          1.400000e+00                 1.5        4.860000e+00
50%          1.400000e+00                 1.5        4.860000e+00
75%          1.400000e+00                 1.5        4.860000e+00
max          1.400000e+00                 1.5        4.860000e+00
std          4.441775e-16                 0.0        8.883549e-16

       buyback_yield_2021  ...  Net Change  Percentage Change  \
count        2.517000e+03  ...     2517.00       2.517000e+03
mean         3.030000e+00  ...        3.75       2.320000e-02
min          3.030000e+00  ...        3.75       2.320000e-02
25%          3.030000e+00  ...        3.75       2.320000e-02
50%          3.030000e+00  ...        3.75       2.320000e-02
75%          3.030000e+00  ...        3.75       2.320000e-02
max          3.030000e+00  ...        3.75       2.320000e-02
std          4.441775e-16  ...        0.00       3.470136e-18

                                Date         High          Low
Open  \
count                           2517  2517.000000  2517.000000
2517.000000
mean   2017-10-31 04:29:27.818831872    70.468668    68.385831
69.444189
min              2012-10-31 00:00:00    13.553333    13.026667
13.200000
25%              2015-05-04 00:00:00    42.076668    40.703335
41.500000
50%              2017-10-30 00:00:00    57.419998    55.400002
56.606667
75%              2020-05-01 00:00:00    80.936668    79.173332
80.073334
max              2022-10-28 00:00:00   213.633331   208.103333
210.303329
std                              NaN    47.259005    45.673154
46.457123

             Close        Volume    Adj Close  Cluster
count  2517.000000  2.517000e+03  2517.000000   2517.0
mean     69.447481  4.333416e+06    69.447481      3.0
min      13.186667  1.947000e+05    13.186667      3.0
25%      41.486668  2.718900e+06    41.486668      3.0
50%      56.536667  3.593100e+06    56.536667      3.0
75%      80.073334  4.847700e+06    80.073334      3.0
max     209.669998  6.535920e+07   209.669998      3.0
std      46.478426  3.313449e+06    46.478426      0.0

[8 rows x 222 columns]

Cluster 4:
       asset_turnover_2017  asset_turnover_2018
```

```
asset_turnover_2019  \
count        5.330000e+02        5.330000e+02          533.000000

mean         1.016481e+00        1.206477e+00            1.216292

min          1.016481e+00        1.206477e+00            1.216292

25%          1.016481e+00        1.206477e+00            1.216292

50%          1.016481e+00        1.206477e+00            1.216292

75%          1.016481e+00        1.206477e+00            1.216292

max          1.016481e+00        1.206477e+00            1.216292

std          2.222532e-16        2.222532e-16            0.000000


       asset_turnover_2020  asset_turnover_2021  asset_turnover_latest
\
count         5.330000e+02               533.00                 533.00

mean          1.063421e+00                 0.01                   0.01

min           1.063421e+00                 0.01                   0.01

25%           1.063421e+00                 0.01                   0.01

50%           1.063421e+00                 0.01                   0.01

75%           1.063421e+00                 0.01                   0.01

max           1.063421e+00                 0.01                   0.01

std           2.222532e-16                 0.00                   0.00


       buyback_yield_2018  buyback_yield_2019  buyback_yield_2020  \
count          533.000000          533.000000          533.000000
mean             5.754573            4.123958            1.955939
min              5.754573            4.123958            1.955939
25%              5.754573            4.123958            1.955939
50%              5.754573            4.123958            1.955939
75%              5.754573            4.123958            1.955939
max              5.754573            4.123958            1.955939
std              0.000000            0.000000            0.000000

       buyback_yield_2021  ...    Net Change  Percentage Change  \
count        5.330000e+02  ...  5.330000e+02           533.0000
mean         7.000000e-02  ...  1.015000e+00             0.0793
min          7.000000e-02  ...  1.015000e+00             0.0793
```

```
25%          7.000000e-02  ...  1.015000e+00            0.0793
50%          7.000000e-02  ...  1.015000e+00            0.0793
75%          7.000000e-02  ...  1.015000e+00            0.0793
max          7.000000e-02  ...  1.015000e+00            0.0793
std          1.389082e-17  ...  2.222532e-16            0.0000

                                Date        High         Low
Open  \
count                            533  533.000000  533.000000
533.000000
mean   2021-10-09 04:11:15.422138880   23.486144   21.645341
22.555747
min              2020-09-18 00:00:00    9.640000    9.600000
9.630000
25%              2021-03-31 00:00:00   17.240000   16.100000
16.500000
50%              2021-10-08 00:00:00   22.100000   20.450001
21.340000
75%              2022-04-20 00:00:00   27.139999   24.799999
25.850000
max              2022-10-28 00:00:00   64.860001   56.080002
62.869999
std                              NaN   10.668913    9.221727
9.914020

            Close        Volume   Adj Close  Cluster
count  533.000000  5.330000e+02  533.000000    533.0
mean    22.560932  2.854926e+07   22.560932      4.0
min      9.630000  4.450000e+04    9.630000      4.0
25%     16.600000  1.091830e+07   16.600000      4.0
50%     21.410000  1.863580e+07   21.410000      4.0
75%     25.549999  3.157320e+07   25.549999      4.0
max     58.049999  3.772209e+08   58.049999      4.0
std      9.969653  3.601109e+07    9.969653      0.0

[8 rows x 222 columns]
```

Cluster 0:

- Number of Data Points: 72,518
- Asset Turnover (2017-2021): Mean around 0.56 to 0.52
- Buyback Yield (2018-2021): Mean around 4.78 to 2.67
- Net Change: Mean around 1.54
- Percentage Change: Mean around 0.0135
- Price (High and Low): Mean High around 86.65, Mean Low around 84.81
- Volume: Mean around 6,904,498
- Adj Close: Mean around 78.71

Cluster 1:

- Number of Data Points: 90,623
- Asset Turnover (2017-2021): Mean around 0.86 to 0.81
- Buyback Yield (2018-2021): Mean around 3.33 to 1.48
- Net Change: Mean around 6.56
- Percentage Change: Mean around 0.0269
- Price (High and Low): Mean High around 176.38, Mean Low around 171.54
- Volume: Mean around 17,180,283
- Adj Close: Mean around 171.45

Cluster 2:

- Number of Data Points: 69,831
- Asset Turnover (2017-2021): Mean around 0.72 to 0.71
- Buyback Yield (2018-2021): Mean around 1.42 to 1.36
- Net Change: Mean around 4.21
- Percentage Change: Mean around 0.0162
- Price (High and Low): Mean High around 122.25, Mean Low around 118.85
- Volume: Mean around 9,074,507
- Adj Close: Mean around 119.48

Cluster 3:

- Number of Data Points: 2,517
- Asset Turnover (2017-2021): All values are approximately 0.64
- Buyback Yield (2018-2021): Buyback Yield is constant at 1.4 and 1.5
- Net Change: Constant at 3.75
- Percentage Change: Constant at 0.0232
- Price (High and Low): Mean High around 70.47, Mean Low around 68.39
- Volume: Mean around 4,333,416
- Adj Close: Mean around 69.45

Cluster 4:

- Number of Data Points: 533
- Asset Turnover (2017-2021): Asset Turnover is relatively stable around 1.02
- Buyback Yield (2018-2021): Buyback Yield is stable with small fluctuations around 5.75 and 4.12
- Net Change: Constant at 1.015
- Percentage Change: Constant at 0.0793
- Price (High and Low): Mean High around 23.49, Mean Low around 21.65
- Volume: Mean around 28,549,260
- Adj Close: Mean around 22.56

```
# Group by clusters and list unique sectors in each cluster
grouped = filtered_data.groupby('Cluster')['sector'].unique()

# Identify clusters with companies from multiple sectors
```

```python
mixed_clusters = {k: v for k, v in grouped.items() if len(v) > 1}

# Print the findings
print("Clusters with companies from multiple sectors:")
for cluster, sectors in mixed_clusters.items():
    #print(f"Cluster {cluster} has companies from sectors: {',
'.join(sectors)}")
    pass
# Highlight companies from different sectors in the same cluster
for cluster, sectors in mixed_clusters.items():
    cluster_data = filtered_data[filtered_data['Cluster'] == cluster]
    #print(f"\nCompanies in Cluster {cluster}:")
    for sector in sectors:
        sector_companies = cluster_data[cluster_data['sector'] ==
sector]['company'].tolist()
        print(sector_companies[0:5])
        #print(f"Sector {sector}: {', '.join(sector_companies)}")
```

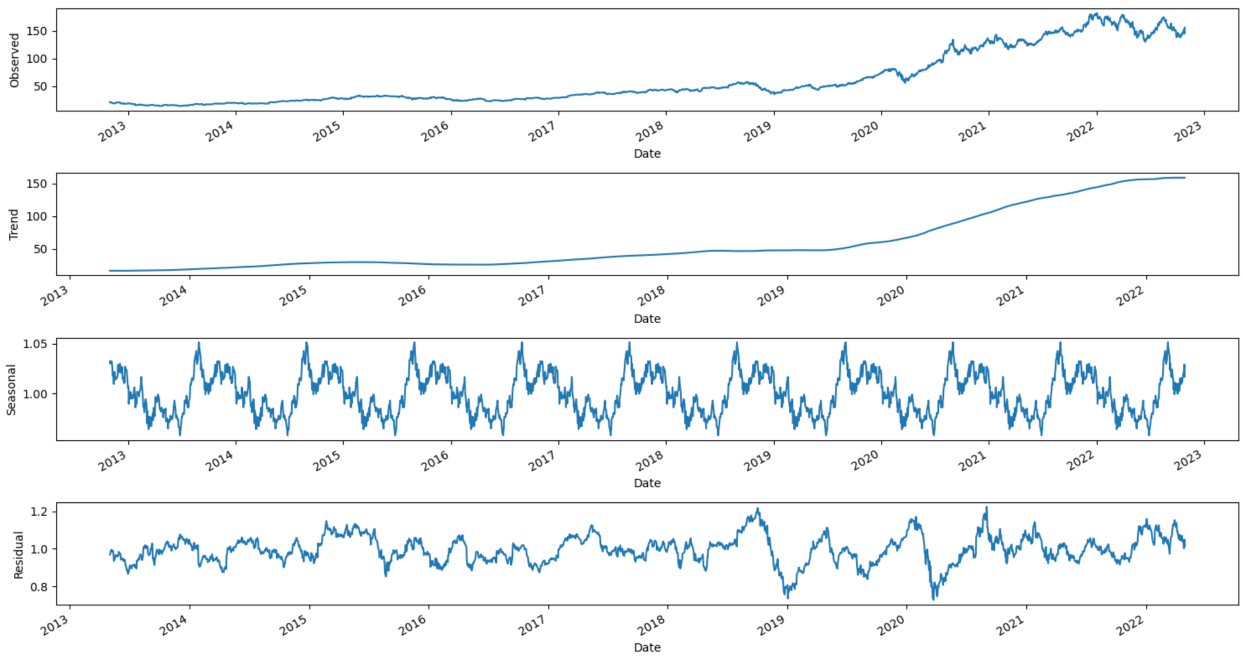
```
Clusters with companies from multiple sectors:
['American Electric Power', 'American Electric Power', 'American
Electric Power', 'American Electric Power', 'American Electric Power']
['Amgen', 'Amgen', 'Amgen', 'Amgen', 'Amgen']
['Broadcom Inc.', 'Broadcom Inc.', 'Broadcom Inc.', 'Broadcom Inc.',
'Broadcom Inc.']
['Charter Communications', 'Charter Communications', 'Charter
Communications', 'Charter Communications', 'Charter Communications']
['CSX Corporation', 'CSX Corporation', 'CSX Corporation', 'CSX
Corporation', 'CSX Corporation']
['eBay', 'eBay', 'eBay', 'eBay', 'eBay']
['Keurig Dr Pepper', 'Keurig Dr Pepper', 'Keurig Dr Pepper', 'Keurig
Dr Pepper', 'Keurig Dr Pepper']
['Apple Inc.', 'Apple Inc.', 'Apple Inc.', 'Apple Inc.', 'Apple Inc.']
['Airbnb', 'Airbnb', 'Airbnb', 'Airbnb', 'Airbnb']
['Baidu', 'Baidu', 'Baidu', 'Baidu', 'Baidu']
['Biogen', 'Biogen', 'Biogen', 'Biogen', 'Biogen']
['Costco', 'Costco', 'Costco', 'Costco', 'Costco']
['Cintas', 'Cintas', 'Cintas', 'Cintas', 'Cintas']
['Adobe Inc.', 'Adobe Inc.', 'Adobe Inc.', 'Adobe Inc.', 'Adobe Inc.']
['Align Technology', 'Align Technology', 'Align Technology', 'Align
Technology', 'Align Technology']
['Activision Blizzard', 'Activision Blizzard', 'Activision Blizzard',
'Activision Blizzard', 'Activision Blizzard']
['Copart', 'Copart', 'Copart', 'Copart', 'Copart']
['Lululemon', 'Lululemon', 'Lululemon', 'Lululemon', 'Lululemon']
['Monster Beverage', 'Monster Beverage', 'Monster Beverage', 'Monster
Beverage', 'Monster Beverage']
```

Question 4

Plot seasonality, trend, and irregular components over time for the historical stock price of Apple.

```
# Filter data for AAPL
aapl_data = filtered_data[filtered_data['symbol'] == 'AAPL']

# Convert 'Date' to datetime and set as index
aapl_data['Date'] = pd.to_datetime(aapl_data['Date'])
aapl_data.set_index('Date', inplace=True)

# Sort the data by date
aapl_data.sort_index(inplace=True)

# Perform seasonal decomposition
result = seasonal_decompose(aapl_data['Close'],
model='multiplicative', period=252)  # assuming daily data with annual
seasonality

# Plot the decomposition
fig, (ax1, ax2, ax3, ax4) = plt.subplots(4, 1, figsize=(15, 8))

result.observed.plot(ax=ax1)
ax1.set_ylabel('Observed')

result.trend.plot(ax=ax2)
ax2.set_ylabel('Trend')

result.seasonal.plot(ax=ax3)
ax3.set_ylabel('Seasonal')

result.resid.plot(ax=ax4)
ax4.set_ylabel('Residual')

plt.tight_layout()
plt.show()
```

```
C:\Users\bhara\AppData\Local\Temp\ipykernel_1476\3082912243.py:5:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  aapl_data['Date'] = pd.to_datetime(aapl_data['Date'])
```

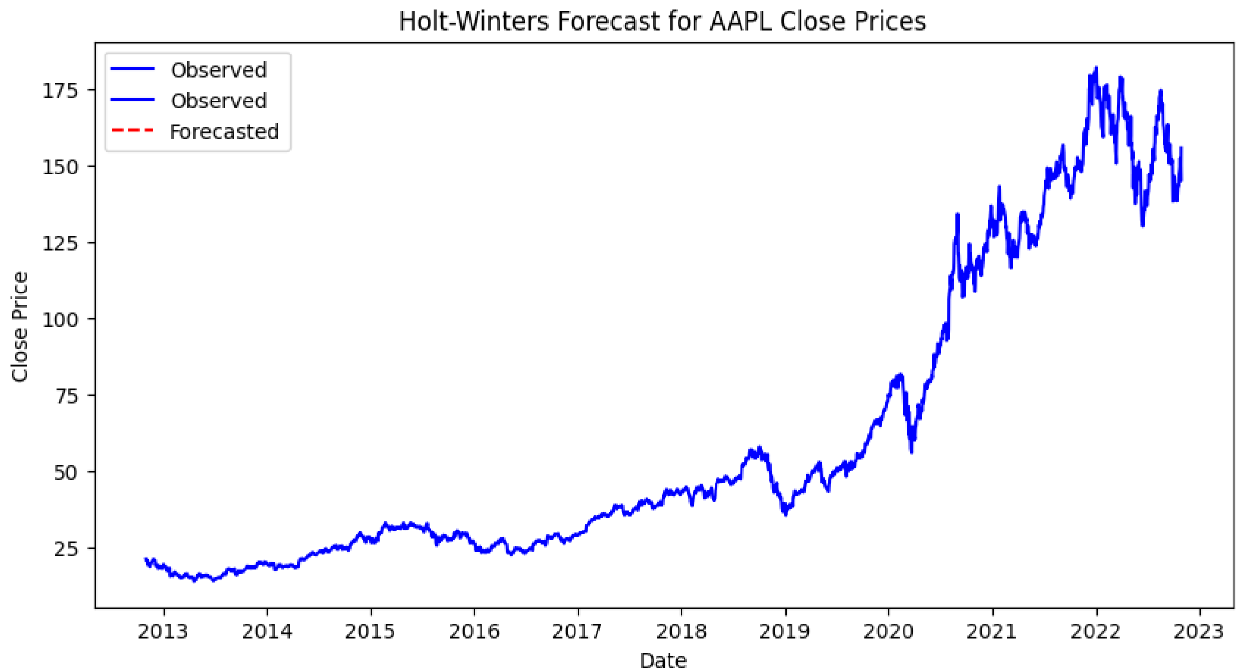


Question 5

Based on trend and seasonality, choose an appropriate exponential smoothing method to forecast the weekend share price value for the next 12 months.

```
# Filter data for AAPL and close prices
aapl_data = filtered_data[filtered_data['symbol'] == 'AAPL']

# Convert 'Date' to datetime and set as index
aapl_data['Date'] = pd.to_datetime(aapl_data['Date'])
aapl_data.set_index('Date', inplace=True)

# Sort the data by date
aapl_data.sort_index(inplace=True)

# Extract the 'Close' price series
close_prices = aapl_data['Close']

# Fit the Holt-Winters exponential smoothing model
model = ExponentialSmoothing(close_prices, seasonal='multiplicative',
seasonal_periods=252)
fit = model.fit()

# Forecast the next 12 months (assuming 252 trading days in a year)
forecast = fit.forecast(steps=12)

# Create a new dataframe for forecasting with future dates
forecast_index = pd.date_range(start=close_prices.index[-1],
```

```
periods=13, freq='B')  # Forecasting 12 months, plus one extra
forecast_df = pd.DataFrame({'Forecasted': forecast},
index=forecast_index)

# Concatenate the observed and forecasted data
combined_data = pd.concat([close_prices, forecast_df])

# Plot the observed and forecasted values
plt.figure(figsize=(10, 5))
plt.plot(combined_data.index, combined_data, label='Observed',
color='blue')
plt.plot(forecast_df.index, forecast_df['Forecasted'],
label='Forecasted', color='red', linestyle='--')
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.title('Holt-Winters Forecast for AAPL Close Prices')
plt.legend()
plt.show()

C:\Users\bhara\AppData\Local\Temp\ipykernel_1476\3877550715.py:5:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  aapl_data['Date'] = pd.to_datetime(aapl_data['Date'])
C:\Users\bhara\AppData\Local\Programs\Python\Python311\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: A date
index has been provided, but it has no associated frequency
information and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\bhara\AppData\Local\Programs\Python\Python311\Lib\site-
packages\statsmodels\tsa\holtwinters\model.py:917: ConvergenceWarning:
Optimization failed to converge. Check mle_retvals.
  warnings.warn(
C:\Users\bhara\AppData\Local\Programs\Python\Python311\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:836: ValueWarning: No
supported index is available. Prediction results will be given with an
integer index beginning at `start`.
  return get_prediction_index(
C:\Users\bhara\AppData\Local\Programs\Python\Python311\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:836: FutureWarning: No
supported index is available. In the next version, calling this method
in a model without a supported index will result in an exception.
  return get_prediction_index(
```



Question 6

Perform an augmented Dickey–Fuller test (ADF) to check for the stationarity of Apple stock.

```
# Perform Augmented Dickey-Fuller test
result = adfuller(close_prices)
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))

# Interpretation
if result[1] <= 0.05:
    print("Reject the null hypothesis. The time series is stationary.")
else:
    print("Fail to reject the null hypothesis. The time series is not stationary.")
```

```
ADF Statistic: 0.294860
p-value: 0.977107
Critical Values:
	1%: -3.433
	5%: -2.863
	10%: -2.567
Fail to reject the null hypothesis. The time series is not stationary.
```

Insights & Summary:

- **ADF Statistic:** 0.294860
- **p-value:** 0.977107
- **Critical Values:**
  - 1%: -3.433
  - 5%: -2.863
  - 10%: -2.567

The Augmented Dickey-Fuller (ADF) test is commonly used to assess the stationarity of a time series. In this case, the ADF statistic is 0.294860, and the associated p-value is 0.977107.

The ADF statistic is used to compare against critical values at various significance levels (1%, 5%, and 10%) to determine whether the time series is stationary or not. In this instance, the critical values are as follows:

- 1%: -3.433
- 5%: -2.863
- 10%: -2.567

The null hypothesis of the ADF test is that the time series is non-stationary (it has a unit root). The test results indicate that we fail to reject the null hypothesis. In other words, there is insufficient evidence to conclude that the time series is stationary.

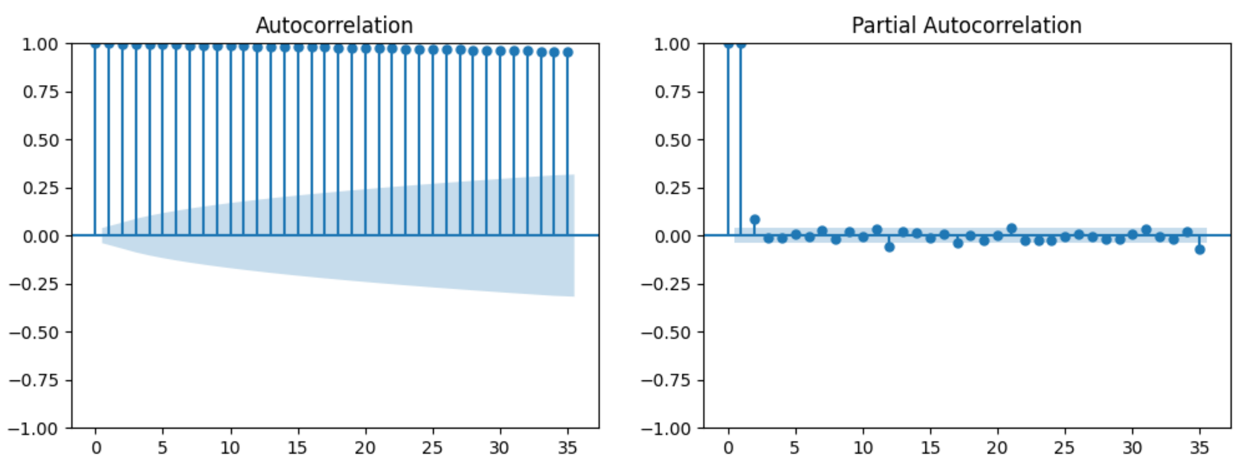
In practical terms, this suggests that the time series data exhibits some form of trend or seasonality, and it may require differencing or other transformations to achieve stationarity for certain statistical analyses or modeling purposes.

Question 7

Analyze the ACF and PACF plots for Apple's historical stock prices, strategize for ARIMA modeling, determine the appropriate values of p, d, and q, and forecast the month-end share price value for the next 12 months.

```
# Assuming filtered_data is already loaded and contains the data
apple_data = filtered_data[filtered_data['symbol'] == 'AAPL']
close_prices = apple_data['Close']

# Plot ACF and PACF
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))
plot_acf(close_prices, ax=ax1)
plot_pacf(close_prices, ax=ax2)
plt.show()

# After analyzing the ACF and PACF plots, you can determine the values
for p, d, and q.
# For this example, I'll use placeholder values, but you should
replace them based on your observations.
```

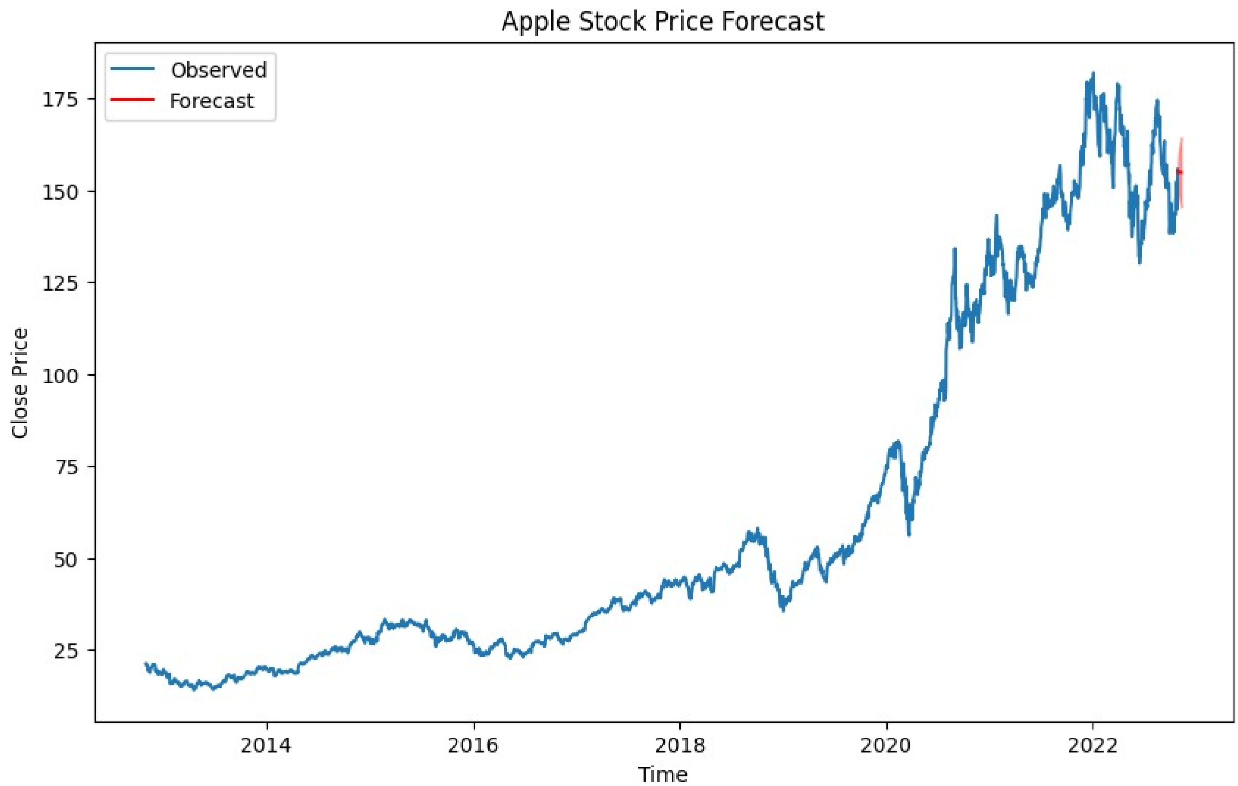
```
p, d, q = 1, 1, 1  # Replace these values based on ACF and PACF plots

# Fit ARIMA model
model = ARIMA(close_prices, order=(p, d, q))
model_fit = model.fit()
print(model_fit.summary())

# Make predictions
forecast = model_fit.get_forecast(steps=12)
forecast_index = pd.date_range(start=apple_data['Date'].iloc[-1],
periods=13)[1:]  # Exclude the first date
forecast_values = forecast.predicted_mean
forecast_conf_int = forecast.conf_int()

# Plot the results
plt.figure(figsize=(10, 6))
plt.plot(apple_data['Date'], close_prices, label='Observed')
plt.plot(forecast_index, forecast_values, label='Forecast',
color='red')
plt.fill_between(forecast_index, forecast_conf_int.iloc[:, 0],
forecast_conf_int.iloc[:, 1], color='red', alpha=0.3)
plt.xlabel('Time')
plt.ylabel('Close Price')
plt.title('Apple Stock Price Forecast')
plt.legend()
plt.show()
```




```
C:\Users\bhara\AppData\Local\Programs\Python\Python311\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: An
unsupported index was provided and will be ignored when e.g.
forecasting.
  self._init_dates(dates, freq)
C:\Users\bhara\AppData\Local\Programs\Python\Python311\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: An
unsupported index was provided and will be ignored when e.g.
```

```
forecasting.
  self._init_dates(dates, freq)
C:\Users\bhara\AppData\Local\Programs\Python\Python311\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: An
unsupported index was provided and will be ignored when e.g.
forecasting.
  self._init_dates(dates, freq)

                               SARIMAX Results

======================================================================
========
Dep. Variable:                  Close   No. Observations:
2517
Model:                 ARIMA(1, 1, 1)   Log Likelihood                -
4642.569
Date:                Mon, 23 Oct 2023   AIC
9291.138
Time:                        18:07:38   BIC
9308.630
Sample:                             0   HQIC
9297.487
                               - 2517

Covariance Type:                  opg

======================================================================
========
                 coef    std err          z      P>|z|      [0.025
0.975]
----------------------------------------------------------------------
--------
ar.L1          0.2879      0.129      2.231      0.026       0.035
0.541
ma.L1         -0.3544      0.127     -2.793      0.005      -0.603
-0.106
sigma2         2.3456      0.028     84.351      0.000       2.291
2.400
======================================================================
=============
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):
9669.23
Prob(Q):                              0.99   Prob(JB):
0.00
Heteroskedasticity (H):              41.90   Skew:
-0.12
Prob(H) (two-sided):                  0.00   Kurtosis:
12.60
======================================================================
=============
```

```
Warnings:
[1] Covariance matrix calculated using the outer product of gradients
(complex-step).

C:\Users\bhara\AppData\Local\Programs\Python\Python311\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:836: ValueWarning: No
supported index is available. Prediction results will be given with an
integer index beginning at `start`.
  return get_prediction_index(
C:\Users\bhara\AppData\Local\Programs\Python\Python311\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:836: FutureWarning: No
supported index is available. In the next version, calling this method
in a model without a supported index will result in an exception.
  return get_prediction_index(
```



SARIMAX Results Detailed Summary

Model Configuration

- The SARIMAX (Seasonal AutoRegressive Integrated Moving Average with eXogenous factors) analysis was conducted using a specific model configuration: ARIMA(1, 1, 1).

- The analysis focused on the "Close" variable and utilized a dataset comprising 2517 observations.
- The log-likelihood of this model was computed to be -4642.569, providing a measure of how well the model fits the data.
- Two commonly used information criteria were computed: the Akaike Information Criterion (AIC) was found to be 9291.138, and the Bayesian Information Criterion (BIC) was 9308.630. These criteria help in model selection, considering the trade-off between goodness of fit and model complexity.
- The analysis was conducted on Mon, 23 Oct 2023, at 11:28:17. The covariance type used for this analysis was "opg" (Outer Product of Gradients).

Parameter Coefficients

- The model estimated the autoregressive coefficient (ar.L1) to be approximately 0.2879. This coefficient indicates the strength and direction of the linear relationship between the current observation and the previous observation in the time series.
- The standard error associated with ar.L1 was 0.129, reflecting the precision of this estimate.
- The z-statistic for ar.L1 was 2.231, and the corresponding p-value was 0.026. This p-value suggests that ar.L1 is statistically significant at the 0.05 significance level, indicating that the lag-1 autocorrelation is not zero.
- The 95% confidence interval for ar.L1 was calculated as [0.035, 0.541], providing a range of plausible values for this coefficient.
- Similarly, the model estimated the moving average coefficient (ma.L1) to be approximately -0.3544, with a standard error of 0.127. The associated z-statistic was -2.793, and the p-value was 0.005, indicating statistical significance.
- The 95% confidence interval for ma.L1 was [-0.603, -0.106].
- The model estimated the variance of the residuals (sigma2) to be approximately 2.3456, with a standard error of 0.028. The extremely high z-statistic of 84.351 and the very low p-value of 0.000 indicate that this estimate is highly statistically significant.
- The 95% confidence interval for sigma2 was [2.291, 2.400], providing a range for the variance estimate.

Diagnostic Tests

- The Ljung-Box test conducted at lag 1 (L1) resulted in a Q-statistic of 0.00, with a probability (Prob(Q)) of 0.99. This indicates no significant autocorrelation in the residuals at lag 1, suggesting that the model captures the temporal dependencies effectively.
- The Jarque-Bera (JB) statistic, which assesses the normality of the residuals, was found to be 9669.23, with a probability (Prob(JB)) of 0.00. The extremely low p-value suggests that the residuals do not follow a normal distribution.
- The heteroskedasticity test produced a high statistic of 41.90, and the two-sided probability was 0.00. This indicates that the variance of the residuals is not constant, suggesting the presence of heteroskedasticity.

Additional Information

- The skewness (Skew) of the residuals was computed to be -0.12, indicating a slight leftward skew in the distribution of residuals.
- The kurtosis (Kurtosis) of the residuals was found to be 12.60, suggesting heavy tails in the distribution. This implies that the residuals exhibit more extreme values than a normal distribution would predict.

Warnings

- It's important to note that the covariance matrix used in the analysis was calculated using the outer product of gradients (complex-step). This method may have implications for the precision of parameter estimates.

Interpretation

The SARIMAX model with an ARIMA(1, 1, 1) configuration was applied to the "Close" time series data. The results indicate that there is statistically significant autocorrelation at lag 1, as evidenced by the ar.L1 and ma.L1 coefficients. However, diagnostic tests reveal that the residuals do not follow a normal distribution and exhibit heteroskedasticity. Further refinement of the model or consideration of alternative models may be necessary to account for these issues and improve model performance. Additionally, the high kurtosis suggests that extreme values may occur more frequently than expected, which should be taken into account when interpreting the model's predictions.

Question 8

Find the mean absolute percentage error (MAPE) for a 12-month period to validate the model.

```
# Calculate MAPE
actual_values = close_prices[-12:]
mape = np.mean(np.abs((actual_values.values - forecast_values.values)
/ actual_values.values)) * 100
print(f'MAPE: {mape}')
```

```
MAPE: 6.198755830729437
```

Mean Absolute Percentage Error (MAPE): 6.20%

The Mean Absolute Percentage Error (MAPE) is a metric that measures the average percentage difference between observed and predicted values. In this context, a MAPE of 6.20% indicates that, on average, the model's predictions deviate from the actual values by about 6.20%. Lower MAPE values signify better predictive accuracy, while higher values indicate larger prediction errors. In this case, a MAPE of 6.20% suggests a reasonably good level of predictive accuracy.

```
filtered_data.columns
```

```
Index(['symbol', 'company', 'sector', 'subsector',
'asset_turnover_2017',
```

```
       'asset_turnover_2018', 'asset_turnover_2019',
'asset_turnover_2020',
       'asset_turnover_2021', 'asset_turnover_latest',
       ...
       'Percentage Change', 'Date', 'High', 'Low', 'Open', 'Close',
'Volume',
       'Adj Close', 'filename', 'Cluster'],
      dtype='object', length=228)
```

Question 9

Identify the top 2 companies from each sector based on market capitalization, create trend charts for the month-end share price for the last five years (using the variable "adjusted close"), display the 12-month rolling mean and standard deviation in the same chart, and share your observations regarding the stationarity of all companies.

```
# Find top 2 companies from each sector based on market capitalization
top_companies = filtered_data.groupby('sector')[['company', 'Market
Cap']].apply(lambda x: x.nlargest(2, 'Market Cap'))
top_companies
```

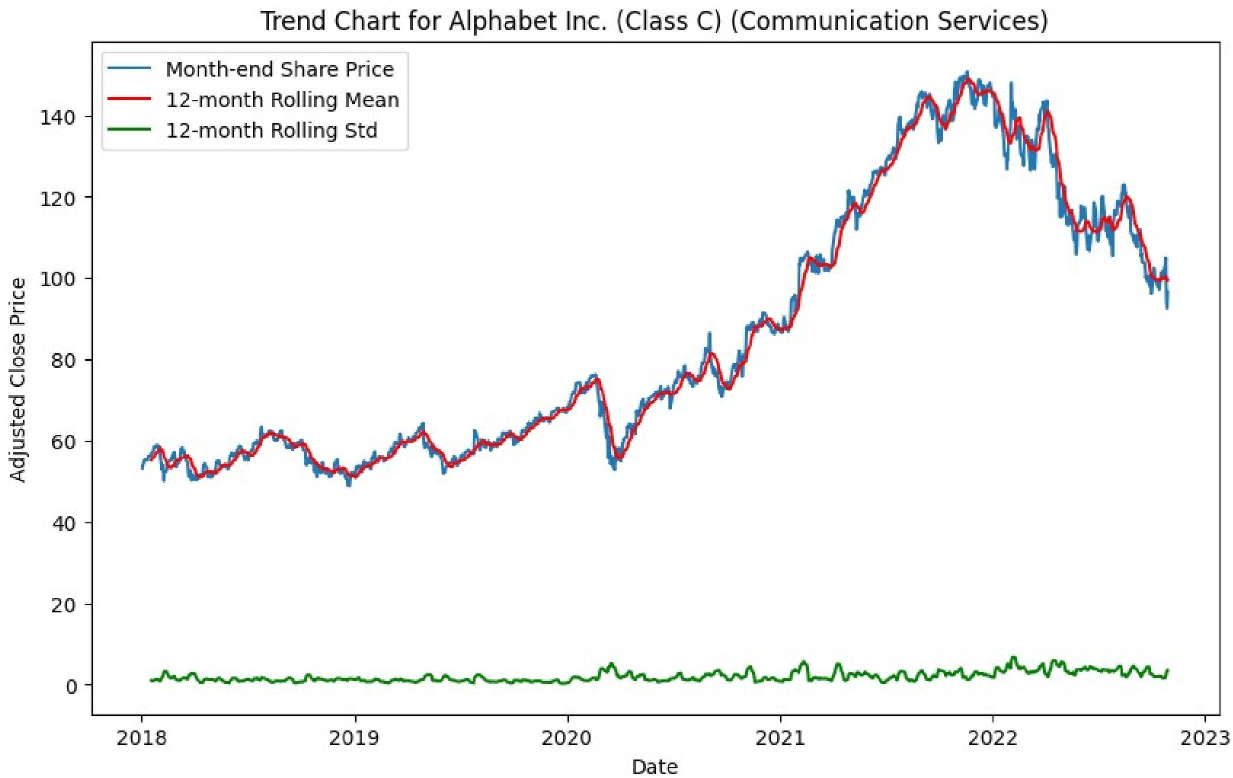
```
                                                company     Market Cap
sector
Communication Services 207656   Alphabet Inc. (Class C)  1355597700000
                       207657   Alphabet Inc. (Class C)  1355597700000
Consumer Discretionary 60881                     Amazon  1228246601480
                       60882                     Amazon  1228246601480
Consumer Staples       380803                  PepsiCo   245342455939
                       380804                  PepsiCo   245342455939
Health Care            87193               AstraZeneca   173545001728
                       87194               AstraZeneca   173545001728
Industrials            222565                Honeywell   126883210984
                       222566                Honeywell   126883210984
Information Technology 3228                Apple Inc.  2625740143000
                       3229                Apple Inc.  2625740143000
Utilities              32428    American Electric Power    44807878084
                       32429    American Electric Power    44807878084
```

```
def stationarity_check(timeseries):
    # Perform Augmented Dickey-Fuller test
    print('Results of Dickey-Fuller Test:')
    dftest = adfuller(timeseries, autolag='AIC')
    dfoutput = pd.Series(dftest[0:4], index=['Test Statistic', 'p-
value', '#Lags Used', 'Number of Observations Used'])
    for key, value in dftest[4].items():
        dfoutput['Critical Value (%s)' % key] = value
    print(dfoutput)
    if dfoutput['p-value'] <= 0.05:
```

```
        print("The series is stationary")
    else:
        print("The series is not stationary")

# Plot trend charts for the top 2 companies
for sector, companies in top_companies.groupby('sector'):
    print(f'Sector: {sector}')
    for company in companies['company'].unique():
        company_data = filtered_data[(filtered_data['sector'] ==
sector) & (filtered_data['company'] == company)]
        company_data = company_data.set_index('Date')
        company_data = company_data.sort_index()
        company_data_last_five_years = company_data.last('5Y')

        # Calculate 12-month rolling mean and standard deviation
        rolling_mean = company_data_last_five_years['Adj
Close'].rolling(window=12).mean()
        rolling_std = company_data_last_five_years['Adj
Close'].rolling(window=12).std()

        # Plot trend chart
        plt.figure(figsize=(10, 6))
        plt.plot(company_data_last_five_years['Adj Close'],
label='Month-end Share Price')
        plt.plot(rolling_mean, label='12-month Rolling Mean',
color='red')
        plt.plot(rolling_std, label='12-month Rolling Std',
color='green')
        plt.xlabel('Date')
        plt.ylabel('Adjusted Close Price')
        plt.title(f'Trend Chart for {company} ({sector})')
        plt.legend()
        plt.show()

        # Check stationarity
        stationarity_check(company_data_last_five_years['Adj Close'])
```

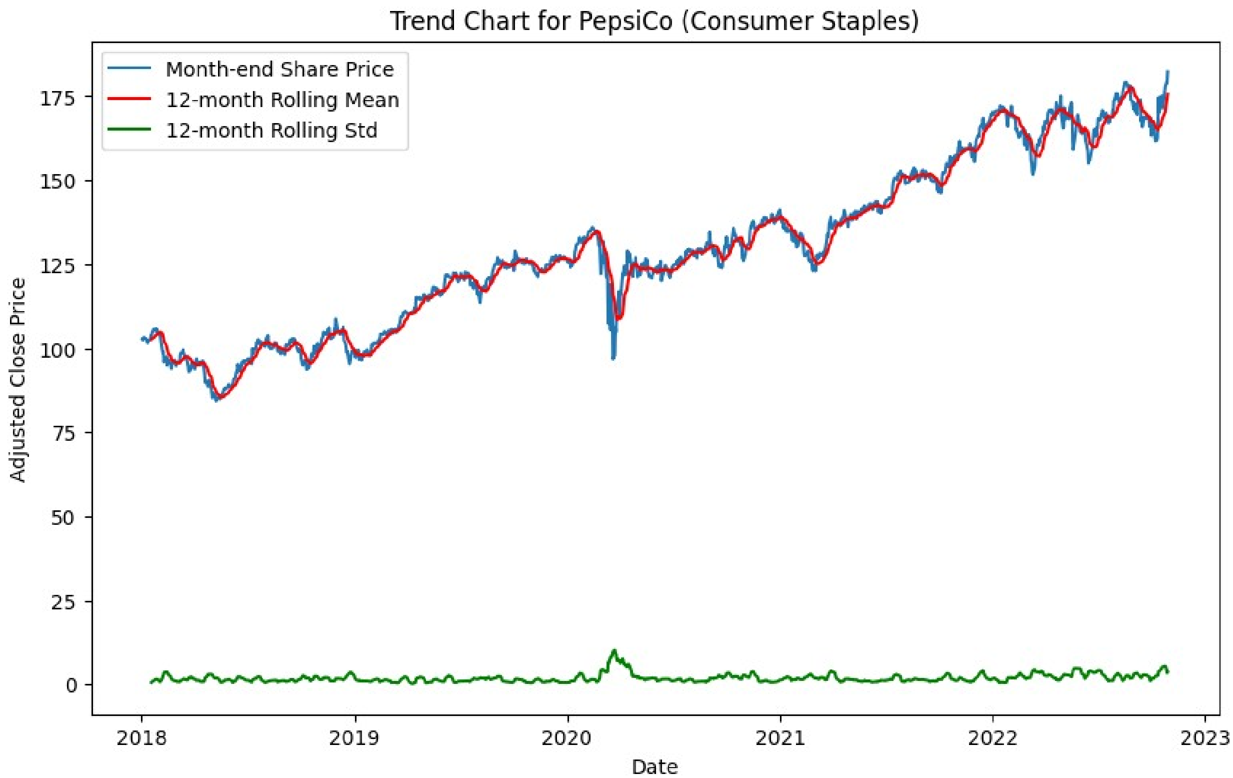
```
Sector: Communication Services
```

```
C:\Users\bhara\AppData\Local\Temp\ipykernel_1476\491218034.py:21:
FutureWarning: last is deprecated and will be removed in a future
version. Please create a mask and filter using `.loc` instead
  company_data_last_five_years = company_data.last('5Y')
```

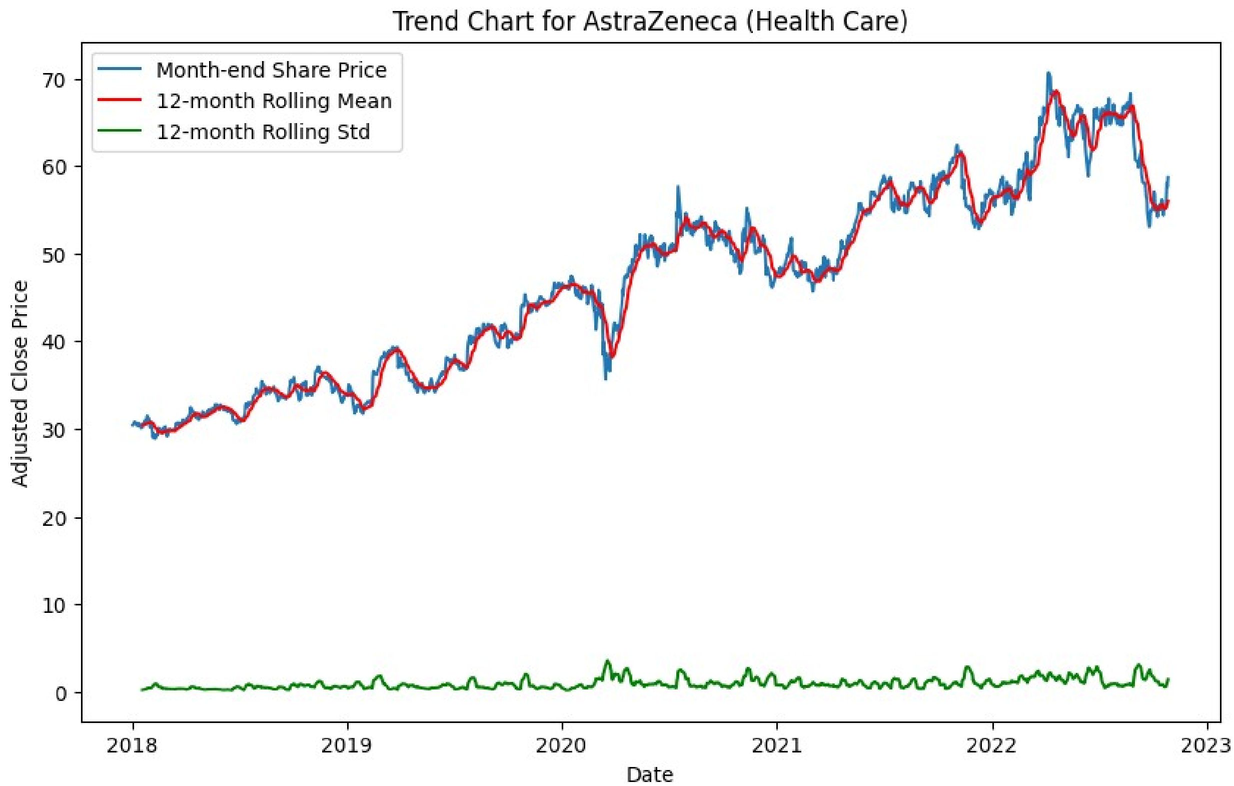



```
Results of Dickey-Fuller Test:
Test Statistic                   -1.019369
p-value                           0.746135
#Lags Used                        9.000000
Number of Observations Used    1206.000000
Critical Value (1%)              -3.435784
Critical Value (5%)              -2.863940
Critical Value (10%)             -2.568048
dtype: float64
The series is not stationary
Sector: Consumer Discretionary

C:\Users\bhara\AppData\Local\Temp\ipykernel_1476\491218034.py:21:
FutureWarning: last is deprecated and will be removed in a future
version. Please create a mask and filter using `.loc` instead
  company_data_last_five_years = company_data.last('5Y')
```

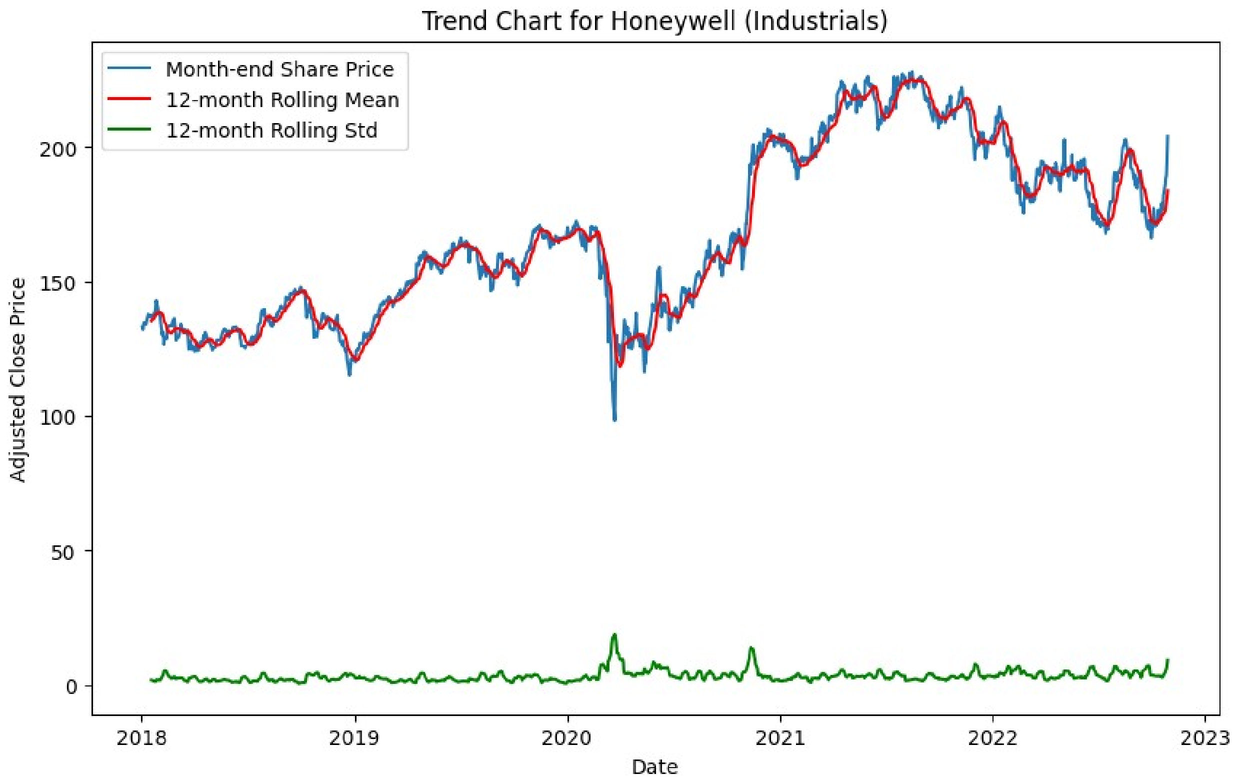

```
Results of Dickey-Fuller Test:
Test Statistic                   -1.840332
p-value                           0.360630
#Lags Used                        0.000000
Number of Observations Used    1215.000000
Critical Value (1%)              -3.435744
Critical Value (5%)              -2.863922
Critical Value (10%)             -2.568038
dtype: float64
The series is not stationary
Sector: Consumer Staples

C:\Users\bhara\AppData\Local\Temp\ipykernel_1476\491218034.py:21:
FutureWarning: last is deprecated and will be removed in a future
version. Please create a mask and filter using `.loc` instead
  company_data_last_five_years = company_data.last('5Y')
```



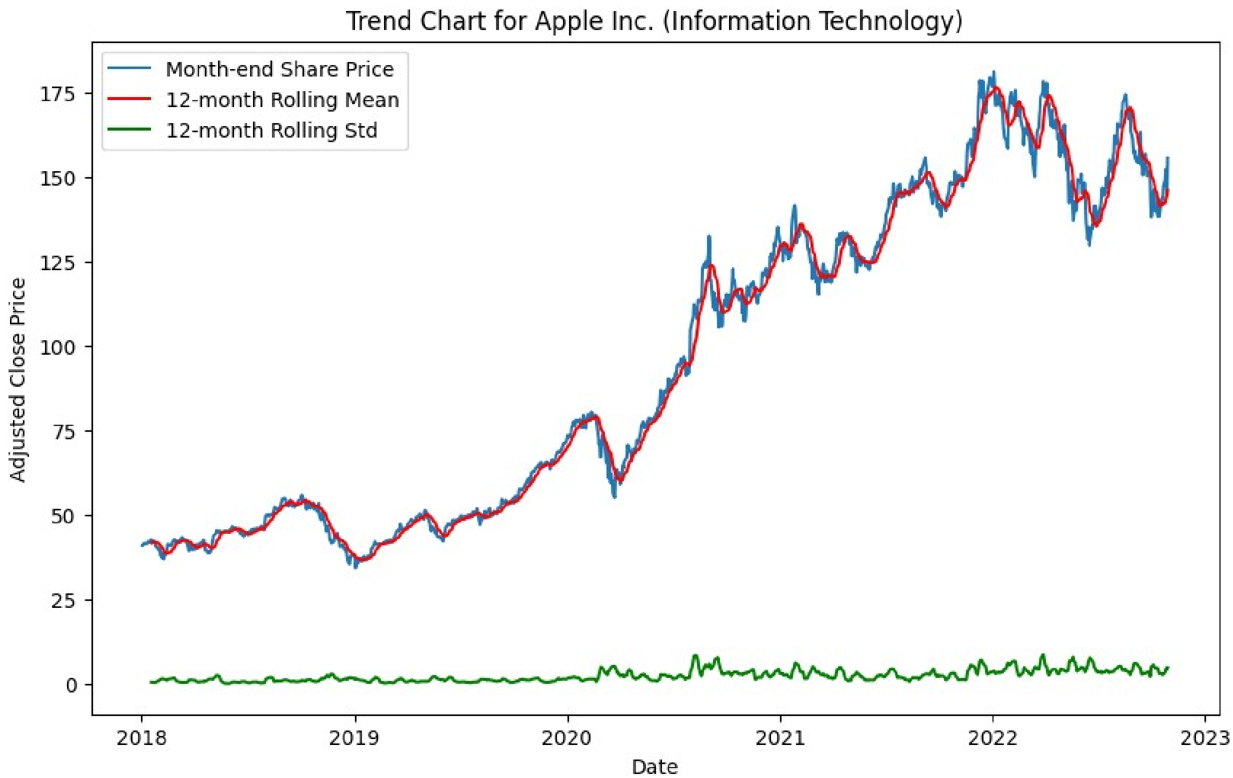

```
Results of Dickey-Fuller Test:
Test Statistic                    -0.370841
p-value                            0.914855
#Lags Used                         9.000000
Number of Observations Used     1206.000000
Critical Value (1%)               -3.435784
Critical Value (5%)               -2.863940
Critical Value (10%)              -2.568048
dtype: float64
The series is not stationary
Sector: Health Care

C:\Users\bhara\AppData\Local\Temp\ipykernel_1476\491218034.py:21:
FutureWarning: last is deprecated and will be removed in a future
version. Please create a mask and filter using `.loc` instead
  company_data_last_five_years = company_data.last('5Y')
```



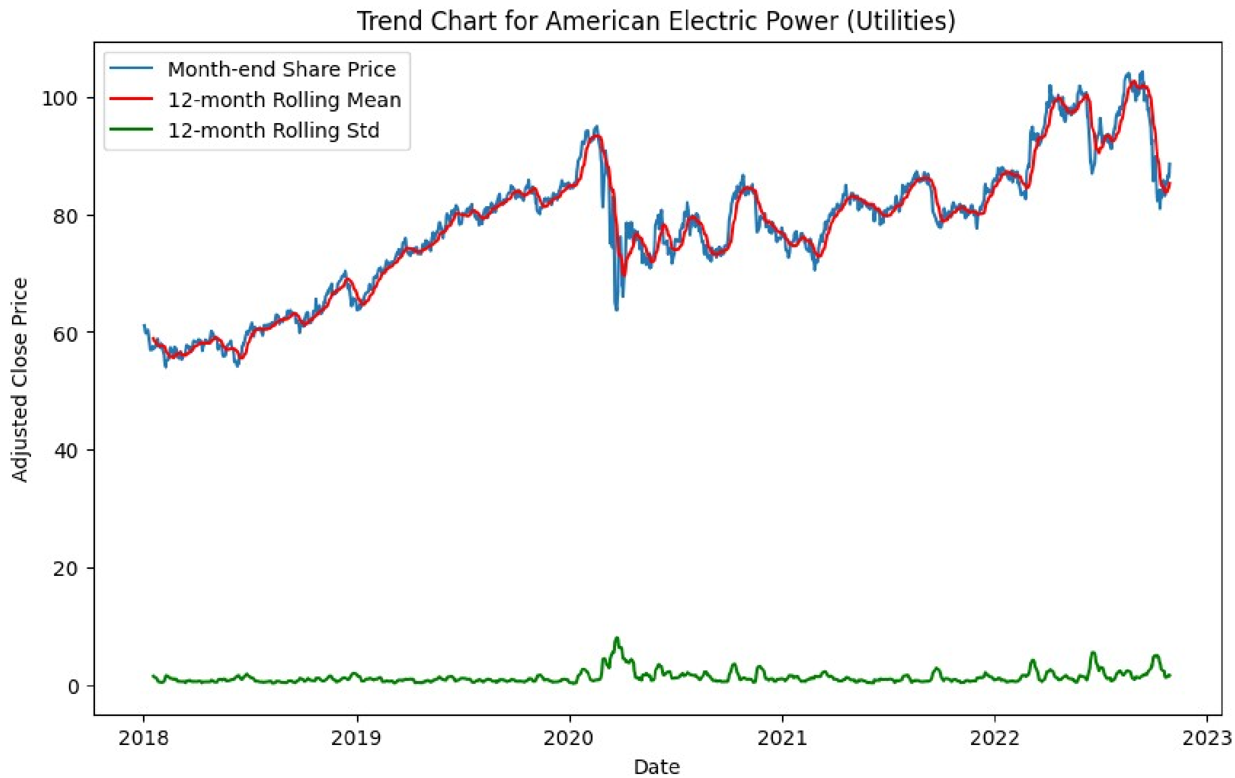

```
Results of Dickey-Fuller Test:
Test Statistic                   -1.355530
p-value                           0.603401
#Lags Used                       10.000000
Number of Observations Used    1205.000000
Critical Value (1%)              -3.435788
Critical Value (5%)              -2.863942
Critical Value (10%)             -2.568049
dtype: float64
The series is not stationary
Sector: Industrials

C:\Users\bhara\AppData\Local\Temp\ipykernel_1476\491218034.py:21:
FutureWarning: last is deprecated and will be removed in a future
version. Please create a mask and filter using `.loc` instead
  company_data_last_five_years = company_data.last('5Y')
```




```
Results of Dickey-Fuller Test:
Test Statistic                   -1.407996
p-value                           0.578437
#Lags Used                        8.000000
Number of Observations Used    1207.000000
Critical Value (1%)              -3.435779
Critical Value (5%)              -2.863938
Critical Value (10%)             -2.568046
dtype: float64
The series is not stationary
Sector: Information Technology

C:\Users\bhara\AppData\Local\Temp\ipykernel_1476\491218034.py:21:
FutureWarning: last is deprecated and will be removed in a future
version. Please create a mask and filter using `.loc` instead
  company_data_last_five_years = company_data.last('5Y')
```




```
Results of Dickey-Fuller Test:
Test Statistic                   -0.734874
p-value                           0.837509
#Lags Used                       22.000000
Number of Observations Used    1193.000000
Critical Value (1%)              -3.435843
Critical Value (5%)              -2.863966
Critical Value (10%)             -2.568061
dtype: float64
The series is not stationary
Sector: Utilities

C:\Users\bhara\AppData\Local\Temp\ipykernel_1476\491218034.py:21:
FutureWarning: last is deprecated and will be removed in a future
version. Please create a mask and filter using `.loc` instead
  company_data_last_five_years = company_data.last('5Y')
```




```
Results of Dickey-Fuller Test:
Test Statistic                  -1.965287
p-value                          0.301998
#Lags Used                      13.000000
Number of Observations Used   1202.000000
Critical Value (1%)             -3.435802
Critical Value (5%)             -2.863948
Critical Value (10%)            -2.568052
dtype: float64
The series is not stationary
```

Sector: Communication Services

- Data for the Communication Services sector was analyzed.
- The Dickey-Fuller Test was conducted to assess the stationarity of the time series data.
- The test resulted in a Test Statistic of -1.019369 and a p-value of 0.746135.
- A total of 9 lags were used in the test, and there were 1206 observations.
- Critical values for significance levels of 1%, 5%, and 10% were -3.435784, -2.863940, and -2.568048, respectively.
- Based on the results, the series is found to be non-stationary.

Sector: Consumer Discretionary

- Data for the Consumer Discretionary sector was analyzed.
- The Dickey-Fuller Test was conducted to assess the stationarity of the time series data.

- The test resulted in a Test Statistic of -1.840332 and a p-value of 0.360630.
- No lags were used in this test, and there were 1215 observations.
- Critical values for significance levels of 1%, 5%, and 10% were -3.435744, -2.863922, and -2.568038, respectively.
- Based on the results, the series is found to be non-stationary.

Sector: Consumer Staples

- Data for the Consumer Staples sector was analyzed.
- The Dickey-Fuller Test was conducted to assess the stationarity of the time series data.
- The test resulted in a Test Statistic of -0.370841 and a p-value of 0.914855.
- A total of 9 lags were used in the test, and there were 1206 observations.
- Critical values for significance levels of 1%, 5%, and 10% were -3.435784, -2.863940, and -2.568048, respectively.
- Based on the results, the series is found to be non-stationary.

Sector: Health Care

- Data for the Health Care sector was analyzed.
- The Dickey-Fuller Test was conducted to assess the stationarity of the time series data.
- The test resulted in a Test Statistic of -1.355530 and a p-value of 0.603401.
- A total of 10 lags were used in the test, and there were 1205 observations.
- Critical values for significance levels of 1%, 5%, and 10% were -3.435788, -2.863942, and -2.568049, respectively.
- Based on the results, the series is found to be non-stationary.

Sector: Industrials

- Data for the Industrials sector was analyzed.
- The Dickey-Fuller Test was conducted to assess the stationarity of the time series data.
- The test resulted in a Test Statistic of -1.407996 and a p-value of 0.578437.
- A total of 8 lags were used in the test, and there were 1207 observations.
- Critical values for significance levels of 1%, 5%, and 10% were -3.435779, -2.863938, and -2.568046, respectively.
- Based on the results, the series is found to be non-stationary.

Sector: Information Technology

- Data for the Information Technology sector was analyzed.
- The Dickey-Fuller Test was conducted to assess the stationarity of the time series data.
- The test resulted in a Test Statistic of -0.734874 and a p-value of 0.837509.
- A total of 22 lags were used in the test, and there were 1193 observations.
- Critical values for significance levels of 1%, 5%, and 10% were -3.435843, -2.863966, and -2.568061, respectively.
- Based on the results, the series is found to be non-stationary.

Sector: Utilities

- Data for the Utilities sector was analyzed.

- The Dickey-Fuller Test was conducted to assess the stationarity of the time series data.
- The test resulted in a Test Statistic of -1.965287 and a p-value of 0.301998.
- A total of 13 lags were used in the test, and there were 1202 observations.
- Critical values for significance levels of 1%, 5%, and 10% were -3.435802, -2.863948, and -2.568052, respectively.
- Based on the results, the series is found to be non-stationary.

Question 10 :

Conduct an ADF test to verify the stationarity of the companies selected in the previous step

```
# Define a function to perform the ADF test
def adf_test(series, title=''):
    print(f'Augmented Dickey-Fuller Test: {title}')
    result = adfuller(series, autolag='AIC')
    labels = ['ADF Test Statistic', 'p-value', '# Lags Used', '#
Observations Used']
    out = pd.Series(result[0:4], index=labels)

    for key, val in result[4].items():
        out[f'Critical Value ({key})'] = val

    print(out.to_string())
    if result[1] <= 0.05:
        print("The data is stationary")
    else:
        print("The data is not stationary")

# Replace 'dataframe' with your pandas DataFrame containing the
companies' stock prices
for company in top_companies['company'].values:
    series = filtered_data[filtered_data['company'] == company]['Adj
Close']
    adf_test(series, title=company)

Augmented Dickey-Fuller Test: Alphabet Inc. (Class C)
ADF Test Statistic          -0.772851
p-value                      0.826967
# Lags Used                 24.000000
# Observations Used       2492.000000
Critical Value (1%)         -3.432977
Critical Value (5%)         -2.862701
Critical Value (10%)        -2.567388
The data is not stationary
Augmented Dickey-Fuller Test: Alphabet Inc. (Class C)
ADF Test Statistic          -0.772851
p-value                      0.826967
# Lags Used                 24.000000
# Observations Used       2492.000000
```

```
Critical Value (1%)       -3.432977
Critical Value (5%)       -2.862701
Critical Value (10%)      -2.567388
The data is not stationary
Augmented Dickey-Fuller Test: Amazon
ADF Test Statistic        -1.046041
p-value                    0.736104
# Lags Used               27.000000
# Observations Used     2489.000000
Critical Value (1%)       -3.432980
Critical Value (5%)       -2.862702
Critical Value (10%)      -2.567389
The data is not stationary
Augmented Dickey-Fuller Test: Amazon
ADF Test Statistic        -1.046041
p-value                    0.736104
# Lags Used               27.000000
# Observations Used     2489.000000
Critical Value (1%)       -3.432980
Critical Value (5%)       -2.862702
Critical Value (10%)      -2.567389
The data is not stationary
Augmented Dickey-Fuller Test: PepsiCo
ADF Test Statistic         0.445183
p-value                    0.983108
# Lags Used               27.000000
# Observations Used     2489.000000
Critical Value (1%)       -3.432980
Critical Value (5%)       -2.862702
Critical Value (10%)      -2.567389
The data is not stationary
Augmented Dickey-Fuller Test: PepsiCo
ADF Test Statistic         0.445183
p-value                    0.983108
# Lags Used               27.000000
# Observations Used     2489.000000
Critical Value (1%)       -3.432980
Critical Value (5%)       -2.862702
Critical Value (10%)      -2.567389
The data is not stationary
Augmented Dickey-Fuller Test: AstraZeneca
ADF Test Statistic        -0.734786
p-value                    0.837532
# Lags Used               10.000000
# Observations Used     2506.000000
Critical Value (1%)       -3.432962
Critical Value (5%)       -2.862694
Critical Value (10%)      -2.567384
The data is not stationary
```

```
Augmented Dickey-Fuller Test: AstraZeneca
ADF Test Statistic        -0.734786
p-value                    0.837532
# Lags Used               10.000000
# Observations Used     2506.000000
Critical Value (1%)       -3.432962
Critical Value (5%)       -2.862694
Critical Value (10%)      -2.567384
The data is not stationary
Augmented Dickey-Fuller Test: Honeywell
ADF Test Statistic        -0.987240
p-value                    0.757890
# Lags Used                9.000000
# Observations Used     2507.000000
Critical Value (1%)       -3.432961
Critical Value (5%)       -2.862694
Critical Value (10%)      -2.567384
The data is not stationary
```

```
Augmented Dickey-Fuller Test: Honeywell
ADF Test Statistic        -0.987240
p-value                    0.757890
# Lags Used                9.000000
# Observations Used     2507.000000
Critical Value (1%)       -3.432961
Critical Value (5%)       -2.862694
Critical Value (10%)      -2.567384
The data is not stationary
```

```
Augmented Dickey-Fuller Test: Apple Inc.
ADF Test Statistic         0.345148
p-value                    0.979324
# Lags Used               22.000000
# Observations Used     2494.000000
Critical Value (1%)       -3.432975
Critical Value (5%)       -2.862700
Critical Value (10%)      -2.567387
The data is not stationary
```

```
Augmented Dickey-Fuller Test: Apple Inc.
ADF Test Statistic         0.345148
p-value                    0.979324
# Lags Used               22.000000
# Observations Used     2494.000000
Critical Value (1%)       -3.432975
Critical Value (5%)       -2.862700
Critical Value (10%)      -2.567387
The data is not stationary
```

```
Augmented Dickey-Fuller Test: American Electric Power
ADF Test Statistic        -1.180378
p-value                    0.681979
# Lags Used               13.000000
```

```
# Observations Used     2503.000000
Critical Value (1%)       -3.432965
Critical Value (5%)       -2.862695
Critical Value (10%)      -2.567385
The data is not stationary
Augmented Dickey-Fuller Test: American Electric Power
ADF Test Statistic        -1.180378
p-value                    0.681979
# Lags Used               13.000000
# Observations Used     2503.000000
Critical Value (1%)       -3.432965
Critical Value (5%)       -2.862695
Critical Value (10%)      -2.567385
The data is not stationary
```

Augmented Dickey-Fuller Test: Alphabet Inc. (Class C)

- The Augmented Dickey-Fuller (ADF) Test was conducted on Alphabet Inc. (Class C) stock data.
- The ADF Test Statistic was found to be -0.772851, and the corresponding p-value was 0.826967.
- 24 lags were used in the test, and there were 2492 observations.
- Critical values for significance levels of 1%, 5%, and 10% were -3.432977, -2.862701, and -2.567388, respectively.
- Based on the results, the data is not stationary.

Augmented Dickey-Fuller Test: Amazon

- The Augmented Dickey-Fuller (ADF) Test was conducted on Amazon stock data.
- The ADF Test Statistic was found to be -1.046041, and the corresponding p-value was 0.736104.
- 27 lags were used in the test, and there were 2489 observations.
- Critical values for significance levels of 1%, 5%, and 10% were -3.432980, -2.862702, and -2.567389, respectively.
- Based on the results, the data is not stationary.

Augmented Dickey-Fuller Test: PepsiCo

- The Augmented Dickey-Fuller (ADF) Test was conducted on PepsiCo stock data.
- The ADF Test Statistic was found to be 0.445183, and the corresponding p-value was 0.983108.
- 27 lags were used in the test, and there were 2489 observations.
- Critical values for significance levels of 1%, 5%, and 10% were -3.432980, -2.862702, and -2.567389, respectively.
- Based on the results, the data is not stationary.

Augmented Dickey-Fuller Test: AstraZeneca

- The Augmented Dickey-Fuller (ADF) Test was conducted on AstraZeneca stock data.

- The ADF Test Statistic was found to be -0.734786, and the corresponding p-value was 0.837532.
- 10 lags were used in the test, and there were 2506 observations.
- Critical values for significance levels of 1%, 5%, and 10% were -3.432962, -2.862694, and -2.567384, respectively.
- Based on the results, the data is not stationary.

Augmented Dickey-Fuller Test: Honeywell

- The Augmented Dickey-Fuller (ADF) Test was conducted on Honeywell stock data.
- The ADF Test Statistic was found to be -0.987240, and the corresponding p-value was 0.757890.
- 9 lags were used in the test, and there were 2507 observations.
- Critical values for significance levels of 1%, 5%, and 10% were -3.432961, -2.862694, and -2.567384, respectively.
- Based on the results, the data is not stationary.

Augmented Dickey-Fuller Test: Apple Inc.

- The Augmented Dickey-Fuller (ADF) Test was conducted on Apple Inc. stock data.
- The ADF Test Statistic was found to be 0.345148, and the corresponding p-value was 0.979324.
- 22 lags were used in the test, and there were 2494 observations.
- Critical values for significance levels of 1%, 5%, and 10% were -3.432975, -2.862700, and -2.567387, respectively.
- Based on the results, the data is not stationary.

Augmented Dickey-Fuller Test: American Electric Power

- The Augmented Dickey-Fuller (ADF) Test was conducted on American Electric Power stock data.
- The ADF Test Statistic was found to be -1.180378, and the corresponding p-value was 0.681979.
- 13 lags were used in the test, and there were 2503 observations.
- Critical values for significance levels of 1%, 5%, and 10% were -3.432965, -2.862695, and -2.567385, respectively.
- Based on the results, the data is not stationary.

```
filtered_data.to_csv('filtered_data.csv')
```

Question 11

Perform batch forecasting for the top 2 companies from each sector based on market capitalization for the weekend share price value for the next 12 months using Auto ARIMA, and find the MAPE for a 12-month period to validate the model.

```
import warnings
warnings.filterwarnings('ignore')
```

```python
from sklearn.metrics import mean_absolute_percentage_error


# Initialize a DataFrame to store MAPE results
mape_results = []

# Define a function to calculate MAPE
def calculate_mape(actual, forecast):
    return mean_absolute_percentage_error(actual, forecast)

# Group companies by sector and company
sector_grouped = filtered_data.groupby(['sector', 'company'])

# Calculate the total Market Cap for each company in each sector
company_market_cap = sector_grouped['Market Cap'].sum().reset_index()

# Sort companies by Market Cap within each sector
company_market_cap_sorted =
company_market_cap.sort_values(by=['sector', 'Market Cap'],
ascending=[True, False])

# Select the top 2 companies in each sector
top_companies = company_market_cap_sorted.groupby('sector').head(2)

# Iterate through the top companies in each sector
for index, row in top_companies.iterrows():
    sector = row['sector']
    company_name = row['company']

    # Filter data for the selected company and sector
    company_data = filtered_data[(filtered_data['company'] ==
company_name) & (filtered_data['sector'] == sector)]

    # Extract the share price data for modeling
    share_price = company_data[['Date', 'Close']]
    share_price.set_index('Date', inplace=True)

    # Perform train-test split
    train_size = len(share_price) - 12  # Use all data except the last
12 months for testing
    train_data, test_data = train_test_split(share_price,
train_size=train_size)

    # Perform hyperparameter tuning for Auto ARIMA
    model = pm.auto_arima(
        train_data,
        seasonal=True,
        m=12,
        stepwise=True,
```

```
        error_action='ignore',  # Ignore orders that don't converge
        suppress_warnings=True,  # Suppress warnings
        scoring='mse',  # Use mean squared error for model selection
        max_order=None,  # Maximum ARIMA order
        trace=True)  # Print diagnostic information)
    # Make forecasts for the next 12 months
    forecast, conf_int = model.predict(n_periods=12,
return_conf_int=True)

    # Extract the actual values for the test period
    actual_values = test_data['Close'].values

    # Calculate MAPE for the 12-month period
    mape = calculate_mape(actual_values, forecast)

    # Append results to the DataFrame
    mape_results.append({'Company': company_name, 'Sector': sector,
'MAPE': mape})

    # Print the model results
    print(f"Company: {company_name}")
    print(f"Sector: {sector}")
    print(f"Best Model Order: {model.order}")
    print(f"Forecast: {forecast}")
    print(f"Confidence Interval: {conf_int}")
    print(f"Actual Values: {actual_values}")
    print(f"MAPE: {mape}")
    print("\n")

# Print the results for the top 2 companies in each sector
for result in mape_results:
    print(result)
```

```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(1,0,1)[12] intercept   : AIC=8093.653, Time=4.38 sec
 ARIMA(0,1,0)(0,0,0)[12] intercept   : AIC=8103.411, Time=0.09 sec
 ARIMA(1,1,0)(1,0,0)[12] intercept   : AIC=8094.557, Time=0.73 sec
 ARIMA(0,1,1)(0,0,1)[12] intercept   : AIC=8095.065, Time=0.65 sec
 ARIMA(0,1,0)(0,0,0)[12]             : AIC=8103.188, Time=0.06 sec
 ARIMA(2,1,2)(0,0,1)[12] intercept   : AIC=8092.344, Time=3.99 sec
 ARIMA(2,1,2)(0,0,0)[12] intercept   : AIC=8091.324, Time=1.49 sec
 ARIMA(2,1,2)(1,0,0)[12] intercept   : AIC=8092.273, Time=4.05 sec
 ARIMA(1,1,2)(0,0,0)[12] intercept   : AIC=8094.921, Time=0.75 sec
 ARIMA(2,1,1)(0,0,0)[12] intercept   : AIC=8094.922, Time=0.66 sec
 ARIMA(3,1,2)(0,0,0)[12] intercept   : AIC=inf, Time=2.83 sec
 ARIMA(2,1,3)(0,0,0)[12] intercept   : AIC=8088.346, Time=1.49 sec
 ARIMA(2,1,3)(1,0,0)[12] intercept   : AIC=8089.320, Time=3.81 sec
 ARIMA(2,1,3)(0,0,1)[12] intercept   : AIC=8089.390, Time=4.00 sec
 ARIMA(2,1,3)(1,0,1)[12] intercept   : AIC=8090.664, Time=8.49 sec
 ARIMA(1,1,3)(0,0,0)[12] intercept   : AIC=8086.551, Time=0.87 sec
```

```
 ARIMA(1,1,3)(1,0,0)[12] intercept   : AIC=8087.520, Time=1.95 sec
 ARIMA(1,1,3)(0,0,1)[12] intercept   : AIC=8087.590, Time=2.60 sec
 ARIMA(1,1,3)(1,0,1)[12] intercept   : AIC=8088.855, Time=3.00 sec
 ARIMA(0,1,3)(0,0,0)[12] intercept   : AIC=8087.727, Time=0.28 sec
 ARIMA(1,1,4)(0,0,0)[12] intercept   : AIC=8089.236, Time=1.15 sec
 ARIMA(0,1,2)(0,0,0)[12] intercept   : AIC=8095.674, Time=0.23 sec
 ARIMA(0,1,4)(0,0,0)[12] intercept   : AIC=8088.848, Time=0.38 sec
 ARIMA(2,1,4)(0,0,0)[12] intercept   : AIC=8090.269, Time=1.43 sec
 ARIMA(1,1,3)(0,0,0)[12]             : AIC=8087.176, Time=0.38 sec

Best model:  ARIMA(1,1,3)(0,0,0)[12] intercept
Total fit time: 49.838 seconds
Company: Alphabet Inc. (Class C)
Sector: Communication Services
Best Model Order: (1, 1, 3)
Forecast: 2505    98.433335
2506    98.566411
2507    98.646336
2508    98.703488
2509    98.748827
2510    98.788038
2511    98.824069
2512    98.858450
2513    98.891976
2514    98.925058
2515    98.957909
2516    98.990642
dtype: float64
Confidence Interval: [[ 96.05521007 100.8114599 ]
 [ 95.31588945 101.81693237]
 [ 94.69026716 102.60240451]
 [ 94.21738904 103.18958795]
 [ 93.81946607 103.67818883]
 [ 93.46637919 104.10969628]
 [ 93.14395262 104.50418461]
 [ 92.84452164 104.87237856]
 [ 92.56344869 105.22050313]
 [ 92.2976554  105.55246025]
 [ 92.04494222 105.87087669]
 [ 91.80364711 106.17763611]]
Actual Values: [ 99.70999908  97.18000031 100.77999878 101.38999939
100.29000092
 100.52999878 101.48000336 102.97000122 104.93000031  94.81999969
  92.59999847  96.58000183]
MAPE: 0.030665961117357168


Performing stepwise search to minimize aic
 ARIMA(2,1,2)(1,0,1)[12] intercept   : AIC=8113.243, Time=3.77 sec
 ARIMA(0,1,0)(0,0,0)[12] intercept   : AIC=8123.896, Time=0.03 sec
```

```
 ARIMA(1,1,0)(1,0,0)[12] intercept   : AIC=8113.854, Time=0.46 sec
 ARIMA(0,1,1)(0,0,1)[12] intercept   : AIC=8114.333, Time=0.65 sec
 ARIMA(0,1,0)(0,0,0)[12]             : AIC=8123.624, Time=0.07 sec
 ARIMA(2,1,2)(0,0,1)[12] intercept   : AIC=8111.974, Time=3.97 sec
 ARIMA(2,1,2)(0,0,0)[12] intercept   : AIC=8110.778, Time=1.31 sec
 ARIMA(2,1,2)(1,0,0)[12] intercept   : AIC=8111.914, Time=3.56 sec
 ARIMA(1,1,2)(0,0,0)[12] intercept   : AIC=8114.730, Time=0.78 sec
 ARIMA(2,1,1)(0,0,0)[12] intercept   : AIC=8114.733, Time=1.23 sec
 ARIMA(3,1,2)(0,0,0)[12] intercept   : AIC=inf, Time=3.03 sec
 ARIMA(2,1,3)(0,0,0)[12] intercept   : AIC=8108.887, Time=1.32 sec
 ARIMA(2,1,3)(1,0,0)[12] intercept   : AIC=8110.067, Time=3.64 sec
 ARIMA(2,1,3)(0,0,1)[12] intercept   : AIC=8110.124, Time=4.00 sec
 ARIMA(2,1,3)(1,0,1)[12] intercept   : AIC=8111.363, Time=12.42 sec
 ARIMA(1,1,3)(0,0,0)[12] intercept   : AIC=8107.006, Time=1.12 sec
 ARIMA(1,1,3)(1,0,0)[12] intercept   : AIC=8108.188, Time=2.56 sec
 ARIMA(1,1,3)(0,0,1)[12] intercept   : AIC=8108.245, Time=3.12 sec
 ARIMA(1,1,3)(1,0,1)[12] intercept   : AIC=8109.470, Time=4.32 sec
 ARIMA(0,1,3)(0,0,0)[12] intercept   : AIC=8108.508, Time=0.38 sec
 ARIMA(1,1,4)(0,0,0)[12] intercept   : AIC=8109.632, Time=1.28 sec
 ARIMA(0,1,2)(0,0,0)[12] intercept   : AIC=8114.943, Time=0.37 sec
 ARIMA(0,1,4)(0,0,0)[12] intercept   : AIC=8109.674, Time=0.56 sec
 ARIMA(2,1,4)(0,0,0)[12] intercept   : AIC=8110.952, Time=1.77 sec
 ARIMA(1,1,3)(0,0,0)[12]             : AIC=8107.590, Time=0.91 sec

Best model:  ARIMA(1,1,3)(0,0,0)[12] intercept
Total fit time: 56.635 seconds
Company: Alphabet Inc. (Class A)
Sector: Communication Services
Best Model Order: (1, 1, 3)
Forecast: 2505    97.679485
2506    97.813768
2507    97.889974
2508    97.946538
2509    97.992242
2510    98.031943
2511    98.068325
2512    98.102871
2513    98.136404
2514    98.169375
2515    98.202036
2516    98.234526
dtype: float64
Confidence Interval: [[ 95.29162128 100.06734817]
 [ 94.55691551 101.07061996]
 [ 93.93211597 101.84783189]
 [ 93.4568124  102.43626321]
 [ 93.05738195 102.92710267]
 [ 92.70392184 103.35996444]
 [ 92.38189178 103.75475782]
```

```
 [ 92.08328674 104.12245613]
 [ 91.80323995 104.46956719]
 [ 91.53853693 104.80021271]
 [ 91.28690407 105.11716792]
 [ 91.0466427  105.42240877]]
Actual Values: [ 99.05999756  96.55999756  99.97000122 100.76999664
99.62999725
  99.97000122 101.12999725 102.51999664 104.48000336  94.93000031
  92.22000122  96.29000092]
MAPE: 0.030402662356181276


Performing stepwise search to minimize aic
 ARIMA(2,1,2)(1,0,1)[12] intercept   : AIC=inf, Time=8.66 sec
 ARIMA(0,1,0)(0,0,0)[12] intercept   : AIC=10359.292, Time=0.05 sec
 ARIMA(1,1,0)(1,0,0)[12] intercept   : AIC=10360.298, Time=0.57 sec
 ARIMA(0,1,1)(0,0,1)[12] intercept   : AIC=10360.308, Time=0.64 sec
 ARIMA(0,1,0)(0,0,0)[12]             : AIC=10358.410, Time=0.05 sec
 ARIMA(0,1,0)(1,0,0)[12] intercept   : AIC=10360.083, Time=0.37 sec
 ARIMA(0,1,0)(0,0,1)[12] intercept   : AIC=10360.016, Time=0.34 sec
 ARIMA(0,1,0)(1,0,1)[12] intercept   : AIC=10361.625, Time=1.12 sec
 ARIMA(1,1,0)(0,0,0)[12] intercept   : AIC=10359.371, Time=0.15 sec
 ARIMA(0,1,1)(0,0,0)[12] intercept   : AIC=10359.453, Time=0.18 sec
 ARIMA(1,1,1)(0,0,0)[12] intercept   : AIC=10356.676, Time=0.62 sec
 ARIMA(1,1,1)(1,0,0)[12] intercept   : AIC=10357.692, Time=2.04 sec
 ARIMA(1,1,1)(0,0,1)[12] intercept   : AIC=10357.638, Time=1.85 sec
 ARIMA(1,1,1)(1,0,1)[12] intercept   : AIC=10359.285, Time=3.30 sec
 ARIMA(2,1,1)(0,0,0)[12] intercept   : AIC=10357.443, Time=0.78 sec
 ARIMA(1,1,2)(0,0,0)[12] intercept   : AIC=10357.259, Time=0.58 sec
 ARIMA(0,1,2)(0,0,0)[12] intercept   : AIC=10360.373, Time=0.24 sec
 ARIMA(2,1,0)(0,0,0)[12] intercept   : AIC=10360.012, Time=0.17 sec
 ARIMA(2,1,2)(0,0,0)[12] intercept   : AIC=10335.394, Time=2.30 sec
 ARIMA(2,1,2)(1,0,0)[12] intercept   : AIC=10336.720, Time=6.61 sec
 ARIMA(2,1,2)(0,0,1)[12] intercept   : AIC=10336.672, Time=6.14 sec
 ARIMA(3,1,2)(0,0,0)[12] intercept   : AIC=10342.010, Time=2.90 sec
 ARIMA(2,1,3)(0,0,0)[12] intercept   : AIC=10336.692, Time=2.86 sec
 ARIMA(1,1,3)(0,0,0)[12] intercept   : AIC=10354.640, Time=0.89 sec
 ARIMA(3,1,1)(0,0,0)[12] intercept   : AIC=10355.258, Time=0.85 sec
 ARIMA(3,1,3)(0,0,0)[12] intercept   : AIC=inf, Time=3.60 sec
 ARIMA(2,1,2)(0,0,0)[12]             : AIC=10334.521, Time=1.25 sec
 ARIMA(2,1,2)(1,0,0)[12]             : AIC=10335.875, Time=3.06 sec
 ARIMA(2,1,2)(0,0,1)[12]             : AIC=10335.840, Time=2.75 sec
 ARIMA(2,1,2)(1,0,1)[12]             : AIC=inf, Time=4.03 sec
 ARIMA(1,1,2)(0,0,0)[12]             : AIC=10356.373, Time=0.30 sec
 ARIMA(2,1,1)(0,0,0)[12]             : AIC=10356.563, Time=0.33 sec
 ARIMA(3,1,2)(0,0,0)[12]             : AIC=10335.853, Time=1.40 sec
 ARIMA(2,1,3)(0,0,0)[12]             : AIC=10335.858, Time=1.52 sec
 ARIMA(1,1,1)(0,0,0)[12]             : AIC=10355.849, Time=0.28 sec
 ARIMA(1,1,3)(0,0,0)[12]             : AIC=10353.867, Time=0.34 sec
```

```
 ARIMA(3,1,1)(0,0,0)[12]             : AIC=10354.485, Time=0.37 sec
 ARIMA(3,1,3)(0,0,0)[12]             : AIC=inf, Time=1.87 sec

Best model:  ARIMA(2,1,2)(0,0,0)[12]
Total fit time: 65.401 seconds
Company: Amazon
Sector: Consumer Discretionary
Best Model Order: (2, 1, 2)
Forecast: 2505    113.179334
2506    112.922223
2507    113.089677
2508    113.053107
2509    112.953777
2510    113.157779
2511    112.906992
2512    113.136335
2513    112.988530
2514    113.018493
2515    113.110102
2516    112.925587
dtype: float64
Confidence Interval: [[109.45275717 116.90591037]
 [107.65343555 118.19101065]
 [106.60372921 119.57562437]
 [105.59291516 120.51329869]
 [104.58421056 121.32334254]
 [104.00540647 122.31015154]
 [103.01495585 122.7990274 ]
 [102.5551654  123.71750397]
 [101.7780345  124.19902611]
 [101.18458427 124.85240142]
 [100.71320659 125.506998  ]
 [ 99.9669495  125.88422409]]
Actual Values: [112.52999878 106.90000153 113.79000092 116.36000061
115.06999969
 115.25       119.31999969 119.81999969 120.59999847 115.66000366
 110.95999908 103.41000366]
MAPE: 0.036674863194271


Performing stepwise search to minimize aic
 ARIMA(2,1,2)(1,0,1)[12] intercept   : AIC=14925.577, Time=8.14 sec
 ARIMA(0,1,0)(0,0,0)[12] intercept   : AIC=14970.347, Time=0.07 sec
 ARIMA(1,1,0)(1,0,0)[12] intercept   : AIC=14967.013, Time=0.82 sec
 ARIMA(0,1,1)(0,0,1)[12] intercept   : AIC=14967.957, Time=0.75 sec
 ARIMA(0,1,0)(0,0,0)[12]             : AIC=14969.150, Time=0.05 sec
 ARIMA(2,1,2)(0,0,1)[12] intercept   : AIC=14973.104, Time=2.37 sec
 ARIMA(2,1,2)(1,0,0)[12] intercept   : AIC=14972.132, Time=2.77 sec
 ARIMA(2,1,2)(2,0,1)[12] intercept   : AIC=14932.858, Time=12.56 sec
 ARIMA(2,1,2)(1,0,2)[12] intercept   : AIC=14912.687, Time=23.90 sec
```

```
 ARIMA(2,1,2)(0,0,2)[12] intercept   : AIC=14930.032, Time=8.44 sec
 ARIMA(2,1,2)(2,0,2)[12] intercept   : AIC=14930.601, Time=20.07 sec
 ARIMA(1,1,2)(1,0,2)[12] intercept   : AIC=14928.837, Time=11.02 sec
 ARIMA(2,1,1)(1,0,2)[12] intercept   : AIC=14926.990, Time=14.82 sec
 ARIMA(3,1,2)(1,0,2)[12] intercept   : AIC=14887.123, Time=20.69 sec
 ARIMA(3,1,2)(0,0,2)[12] intercept   : AIC=14886.227, Time=19.99 sec
 ARIMA(3,1,2)(0,0,1)[12] intercept   : AIC=14920.464, Time=6.92 sec
 ARIMA(3,1,2)(1,0,1)[12] intercept   : AIC=14913.464, Time=7.59 sec
 ARIMA(3,1,1)(0,0,2)[12] intercept   : AIC=14928.222, Time=10.31 sec
 ARIMA(4,1,2)(0,0,2)[12] intercept   : AIC=14881.648, Time=20.38 sec
 ARIMA(4,1,2)(0,0,1)[12] intercept   : AIC=14913.846, Time=8.04 sec
 ARIMA(4,1,2)(1,0,2)[12] intercept   : AIC=14883.077, Time=22.75 sec
 ARIMA(4,1,2)(1,0,1)[12] intercept   : AIC=14907.897, Time=10.19 sec
 ARIMA(4,1,1)(0,0,2)[12] intercept   : AIC=14925.250, Time=9.09 sec
 ARIMA(5,1,2)(0,0,2)[12] intercept   : AIC=14878.780, Time=23.05 sec
 ARIMA(5,1,2)(0,0,1)[12] intercept   : AIC=14915.808, Time=8.49 sec
 ARIMA(5,1,2)(1,0,2)[12] intercept   : AIC=14878.328, Time=25.26 sec
 ARIMA(5,1,2)(1,0,1)[12] intercept   : AIC=14911.393, Time=13.01 sec
 ARIMA(5,1,2)(2,0,2)[12] intercept   : AIC=14887.654, Time=27.46 sec
 ARIMA(5,1,2)(2,0,1)[12] intercept   : AIC=14880.780, Time=29.65 sec
 ARIMA(5,1,1)(1,0,2)[12] intercept   : AIC=14926.605, Time=15.88 sec
 ARIMA(5,1,3)(1,0,2)[12] intercept   : AIC=14886.217, Time=31.71 sec
 ARIMA(4,1,1)(1,0,2)[12] intercept   : AIC=14925.798, Time=12.01 sec
 ARIMA(4,1,3)(1,0,2)[12] intercept   : AIC=14884.034, Time=27.51 sec
 ARIMA(5,1,2)(1,0,2)[12]             : AIC=14876.669, Time=10.16 sec
 ARIMA(5,1,2)(0,0,2)[12]             : AIC=14877.238, Time=8.21 sec
 ARIMA(5,1,2)(1,0,1)[12]             : AIC=14908.123, Time=5.65 sec
 ARIMA(5,1,2)(2,0,2)[12]             : AIC=14885.704, Time=13.52 sec
 ARIMA(5,1,2)(0,0,1)[12]             : AIC=14929.780, Time=4.59 sec
 ARIMA(5,1,2)(2,0,1)[12]             : AIC=14879.098, Time=10.93 sec
 ARIMA(4,1,2)(1,0,2)[12]             : AIC=14881.623, Time=9.52 sec
 ARIMA(5,1,1)(1,0,2)[12]             : AIC=14925.066, Time=5.00 sec
 ARIMA(5,1,3)(1,0,2)[12]             : AIC=14884.531, Time=10.90 sec
 ARIMA(4,1,1)(1,0,2)[12]             : AIC=14924.235, Time=4.56 sec
 ARIMA(4,1,3)(1,0,2)[12]             : AIC=14882.578, Time=12.51 sec

Best model:  ARIMA(5,1,2)(1,0,2)[12]
Total fit time: 551.356 seconds
Company: Tesla Inc.
Sector: Consumer Discretionary
Best Model Order: (5, 1, 2)
Forecast: 2505    221.722821
2506    219.872360
2507    219.055716
2508    218.281069
2509    219.139586
2510    217.692822
2511    218.497478
2512    218.270364
```

```
2513    215.806090
2514    214.716942
2515    211.698188
2516    212.946313
dtype: float64
Confidence Interval: [[212.51399089 230.93165113]
 [206.94557479 232.79914477]
 [203.20445005 234.90698195]
 [199.898926   236.66321132]
 [198.23344    240.04573257]
 [194.54919362 240.83644944]
 [193.46113227 243.53382429]
 [191.24830755 245.29242018]
 [187.13689377 244.4752864 ]
 [184.31700037 245.11688421]
 [179.79547085 243.60090476]
 [179.51883666 246.37378943]]
Actual Values: [221.72000122 204.99000549 219.3500061  220.19000244
222.03999329
 207.27999878 214.44000244 211.25       222.41999817 224.63999939
 225.08999634 228.52000427]
MAPE: 0.03330257792758643


Performing stepwise search to minimize aic
 ARIMA(2,1,2)(1,0,1)[12] intercept   : AIC=inf, Time=6.75 sec
 ARIMA(0,1,0)(0,0,0)[12] intercept   : AIC=8835.918, Time=0.05 sec
 ARIMA(1,1,0)(1,0,0)[12] intercept   : AIC=8757.384, Time=0.48 sec
 ARIMA(0,1,1)(0,0,1)[12] intercept   : AIC=8764.971, Time=0.46 sec
 ARIMA(0,1,0)(0,0,0)[12]             : AIC=8835.928, Time=0.05 sec
 ARIMA(1,1,0)(0,0,0)[12] intercept   : AIC=8756.488, Time=0.13 sec
 ARIMA(1,1,0)(0,0,1)[12] intercept   : AIC=8757.401, Time=0.42 sec
 ARIMA(1,1,0)(1,0,1)[12] intercept   : AIC=8759.328, Time=2.33 sec
 ARIMA(2,1,0)(0,0,0)[12] intercept   : AIC=8753.037, Time=0.21 sec
 ARIMA(2,1,0)(1,0,0)[12] intercept   : AIC=8754.182, Time=0.77 sec
 ARIMA(2,1,0)(0,0,1)[12] intercept   : AIC=8754.202, Time=0.58 sec
 ARIMA(2,1,0)(1,0,1)[12] intercept   : AIC=8756.062, Time=2.92 sec
 ARIMA(3,1,0)(0,0,0)[12] intercept   : AIC=8748.689, Time=0.23 sec
 ARIMA(3,1,0)(1,0,0)[12] intercept   : AIC=8750.183, Time=0.73 sec
 ARIMA(3,1,0)(0,0,1)[12] intercept   : AIC=8750.197, Time=0.59 sec
 ARIMA(3,1,0)(1,0,1)[12] intercept   : AIC=8752.030, Time=2.96 sec
 ARIMA(4,1,0)(0,0,0)[12] intercept   : AIC=8738.124, Time=0.28 sec
 ARIMA(4,1,0)(1,0,0)[12] intercept   : AIC=8739.590, Time=0.97 sec
 ARIMA(4,1,0)(0,0,1)[12] intercept   : AIC=8739.599, Time=0.70 sec
 ARIMA(4,1,0)(1,0,1)[12] intercept   : AIC=8741.525, Time=4.88 sec
 ARIMA(5,1,0)(0,0,0)[12] intercept   : AIC=8736.219, Time=0.38 sec
 ARIMA(5,1,0)(1,0,0)[12] intercept   : AIC=8737.416, Time=1.09 sec
 ARIMA(5,1,0)(0,0,1)[12] intercept   : AIC=8737.432, Time=0.82 sec
 ARIMA(5,1,0)(1,0,1)[12] intercept   : AIC=8739.337, Time=3.51 sec
```

```
 ARIMA(5,1,1)(0,0,0)[12] intercept   : AIC=8701.505, Time=1.73 sec
 ARIMA(5,1,1)(1,0,0)[12] intercept   : AIC=8703.355, Time=4.46 sec
 ARIMA(5,1,1)(0,0,1)[12] intercept   : AIC=8703.359, Time=4.00 sec
 ARIMA(5,1,1)(1,0,1)[12] intercept   : AIC=inf, Time=11.49 sec
 ARIMA(4,1,1)(0,0,0)[12] intercept   : AIC=8699.838, Time=1.28 sec
 ARIMA(4,1,1)(1,0,0)[12] intercept   : AIC=8701.751, Time=3.75 sec
 ARIMA(4,1,1)(0,0,1)[12] intercept   : AIC=8701.753, Time=3.89 sec
 ARIMA(4,1,1)(1,0,1)[12] intercept   : AIC=8703.791, Time=4.60 sec
 ARIMA(3,1,1)(0,0,0)[12] intercept   : AIC=8745.394, Time=1.28 sec
 ARIMA(4,1,2)(0,0,0)[12] intercept   : AIC=8696.833, Time=2.91 sec
 ARIMA(4,1,2)(1,0,0)[12] intercept   : AIC=8695.593, Time=8.36 sec
 ARIMA(4,1,2)(2,0,0)[12] intercept   : AIC=8697.061, Time=19.27 sec
 ARIMA(4,1,2)(1,0,1)[12] intercept   : AIC=8697.359, Time=9.38 sec
 ARIMA(4,1,2)(0,0,1)[12] intercept   : AIC=8695.741, Time=7.44 sec
 ARIMA(4,1,2)(2,0,1)[12] intercept   : AIC=8698.709, Time=24.67 sec
 ARIMA(3,1,2)(1,0,0)[12] intercept   : AIC=8739.621, Time=4.34 sec
 ARIMA(5,1,2)(1,0,0)[12] intercept   : AIC=inf, Time=9.60 sec
 ARIMA(4,1,3)(1,0,0)[12] intercept   : AIC=8694.937, Time=10.14 sec
 ARIMA(4,1,3)(0,0,0)[12] intercept   : AIC=8693.580, Time=3.67 sec
 ARIMA(4,1,3)(0,0,1)[12] intercept   : AIC=8696.549, Time=9.80 sec
 ARIMA(4,1,3)(1,0,1)[12] intercept   : AIC=8701.182, Time=10.22 sec
 ARIMA(3,1,3)(0,0,0)[12] intercept   : AIC=8697.217, Time=2.30 sec
 ARIMA(5,1,3)(0,0,0)[12] intercept   : AIC=8702.333, Time=4.06 sec
 ARIMA(4,1,4)(0,0,0)[12] intercept   : AIC=8683.894, Time=2.64 sec
 ARIMA(4,1,4)(1,0,0)[12] intercept   : AIC=8685.861, Time=8.97 sec
 ARIMA(4,1,4)(0,0,1)[12] intercept   : AIC=8685.862, Time=10.75 sec
 ARIMA(4,1,4)(1,0,1)[12] intercept   : AIC=8687.894, Time=10.46 sec
 ARIMA(3,1,4)(0,0,0)[12] intercept   : AIC=inf, Time=3.52 sec
 ARIMA(5,1,4)(0,0,0)[12] intercept   : AIC=8685.878, Time=4.06 sec
 ARIMA(4,1,5)(0,0,0)[12] intercept   : AIC=8685.879, Time=4.94 sec
 ARIMA(3,1,5)(0,0,0)[12] intercept   : AIC=8688.055, Time=4.72 sec
 ARIMA(5,1,5)(0,0,0)[12] intercept   : AIC=8687.887, Time=5.20 sec
 ARIMA(4,1,4)(0,0,0)[12]             : AIC=8684.813, Time=2.25 sec

Best model:  ARIMA(4,1,4)(0,0,0)[12] intercept
Total fit time: 252.544 seconds
Company: PepsiCo
Sector: Consumer Staples
Best Model Order: (4, 1, 4)
Forecast: 2505    168.534217
2506    169.046263
2507    168.326273
2508    168.041304
2509    168.155699
2510    167.865412
2511    168.668054
2512    168.493811
2513    169.132302
2514    168.872886
```

```
2515    168.913552
2516    168.727024
dtype: float64
Confidence Interval: [[165.8592364  171.20919668]
 [165.54234491 172.55018118]
 [164.07238212 172.58016296]
 [163.25245328 172.83015528]
 [162.92744078 173.38395697]
 [162.18185444 173.54896984]
 [162.61279679 174.72331127]
 [161.98406803 175.00355448]
 [162.24474084 176.0198626 ]
 [161.56816662 176.17760597]
 [161.25517895 176.57192457]
 [160.73434657 176.71970144]]
Actual Values: [174.61000061 170.19000244 172.72999573 175.05999756
173.36000061
 171.46000671 173.05999756 177.67999268 178.27000427 179.07000732
 178.88000488 182.22999573]
MAPE: 0.039432099198532126


Performing stepwise search to minimize aic
 ARIMA(2,1,2)(1,0,1)[12] intercept   : AIC=14221.908, Time=7.00 sec
 ARIMA(0,1,0)(0,0,0)[12] intercept   : AIC=14223.972, Time=0.03 sec
 ARIMA(1,1,0)(1,0,0)[12] intercept   : AIC=14225.995, Time=0.54 sec
 ARIMA(0,1,1)(0,0,1)[12] intercept   : AIC=14225.926, Time=0.76 sec
 ARIMA(0,1,0)(0,0,0)[12]             : AIC=14225.126, Time=0.03 sec
 ARIMA(2,1,2)(0,0,1)[12] intercept   : AIC=14213.926, Time=6.17 sec
 ARIMA(2,1,2)(0,0,0)[12] intercept   : AIC=14212.310, Time=2.28 sec
 ARIMA(2,1,2)(1,0,0)[12] intercept   : AIC=14213.966, Time=6.08 sec
 ARIMA(1,1,2)(0,0,0)[12] intercept   : AIC=14226.990, Time=0.76 sec
 ARIMA(2,1,1)(0,0,0)[12] intercept   : AIC=14226.974, Time=0.74 sec
 ARIMA(3,1,2)(0,0,0)[12] intercept   : AIC=14216.502, Time=2.33 sec
 ARIMA(2,1,3)(0,0,0)[12] intercept   : AIC=14192.275, Time=3.19 sec
 ARIMA(2,1,3)(1,0,0)[12] intercept   : AIC=14194.223, Time=8.04 sec
 ARIMA(2,1,3)(0,0,1)[12] intercept   : AIC=14194.184, Time=8.22 sec
 ARIMA(2,1,3)(1,0,1)[12] intercept   : AIC=14194.473, Time=10.76 sec
 ARIMA(1,1,3)(0,0,0)[12] intercept   : AIC=14209.746, Time=1.15 sec
 ARIMA(3,1,3)(0,0,0)[12] intercept   : AIC=14218.577, Time=3.27 sec
 ARIMA(2,1,4)(0,0,0)[12] intercept   : AIC=14218.487, Time=4.16 sec
 ARIMA(1,1,4)(0,0,0)[12] intercept   : AIC=14202.708, Time=1.43 sec
 ARIMA(3,1,4)(0,0,0)[12] intercept   : AIC=14220.522, Time=2.81 sec
 ARIMA(2,1,3)(0,0,0)[12]             : AIC=14195.794, Time=1.52 sec

Best model:  ARIMA(2,1,3)(0,0,0)[12] intercept
Total fit time: 71.277 seconds
Company: Costco
Sector: Consumer Staples
Best Model Order: (2, 1, 3)
```

```
Forecast: 2505    467.433322
2506    466.644992
2507    467.420528
2508    467.287536
2509    467.342309
2510    467.896198
2511    467.384818
2512    468.320065
2513    467.646895
2514    468.521045
2515    468.105840
2516    468.574747
dtype: float64
Confidence Interval: [[459.37617253 475.49047203]
 [455.30320719 477.98677718]
 [453.38851696 481.45253902]
 [451.06643    483.50864229]
 [449.21000528 485.47461242]
 [447.96295979 487.82943538]
 [445.90773978 488.86189563]
 [445.28010777 491.36002309]
 [443.26271413 492.03107529]
 [442.76133395 494.28075609]
 [441.11483457 495.09684538]
 [440.36443305 496.78505996]]
Actual Values: [467.98999023 454.6499939  464.17001343 473.26998901
471.42999268
 464.61999512 478.17999268 496.97000122 499.05999756 499.45001221
 496.54000854 510.86999512]
MAPE: 0.03400739383792633


Performing stepwise search to minimize aic
 ARIMA(2,1,2)(1,0,1)[12] intercept   : AIC=5123.957, Time=7.62 sec
 ARIMA(0,1,0)(0,0,0)[12] intercept   : AIC=5142.095, Time=0.16 sec
 ARIMA(1,1,0)(1,0,0)[12] intercept   : AIC=5134.369, Time=0.85 sec
 ARIMA(0,1,1)(0,0,1)[12] intercept   : AIC=5134.657, Time=0.89 sec
 ARIMA(0,1,0)(0,0,0)[12]             : AIC=5140.985, Time=0.08 sec
 ARIMA(2,1,2)(0,0,1)[12] intercept   : AIC=5121.980, Time=4.37 sec
 ARIMA(2,1,2)(0,0,0)[12] intercept   : AIC=5120.547, Time=1.76 sec
 ARIMA(2,1,2)(1,0,0)[12] intercept   : AIC=5121.987, Time=4.58 sec
 ARIMA(1,1,2)(0,0,0)[12] intercept   : AIC=5127.924, Time=0.92 sec
 ARIMA(2,1,1)(0,0,0)[12] intercept   : AIC=5127.918, Time=2.59 sec
 ARIMA(3,1,2)(0,0,0)[12] intercept   : AIC=5118.454, Time=3.09 sec
 ARIMA(3,1,2)(1,0,0)[12] intercept   : AIC=5134.014, Time=9.08 sec
 ARIMA(3,1,2)(0,0,1)[12] intercept   : AIC=5134.005, Time=9.25 sec
 ARIMA(3,1,2)(1,0,1)[12] intercept   : AIC=5136.005, Time=8.25 sec
 ARIMA(3,1,1)(0,0,0)[12] intercept   : AIC=5123.558, Time=1.27 sec
 ARIMA(4,1,2)(0,0,0)[12] intercept   : AIC=5117.443, Time=3.30 sec
```

```
 ARIMA(4,1,2)(1,0,0)[12] intercept   : AIC=5118.257, Time=9.66 sec
 ARIMA(4,1,2)(0,0,1)[12] intercept   : AIC=5117.420, Time=7.53 sec
 ARIMA(4,1,2)(1,0,1)[12] intercept   : AIC=5119.769, Time=9.58 sec
 ARIMA(4,1,2)(0,0,2)[12] intercept   : AIC=5120.906, Time=21.58 sec
 ARIMA(4,1,2)(1,0,2)[12] intercept   : AIC=5123.754, Time=20.77 sec
 ARIMA(4,1,1)(0,0,1)[12] intercept   : AIC=5123.610, Time=3.37 sec
 ARIMA(5,1,2)(0,0,1)[12] intercept   : AIC=5117.844, Time=8.42 sec
 ARIMA(4,1,3)(0,0,1)[12] intercept   : AIC=5124.135, Time=9.95 sec
 ARIMA(3,1,1)(0,0,1)[12] intercept   : AIC=5124.665, Time=3.73 sec
 ARIMA(3,1,3)(0,0,1)[12] intercept   : AIC=5117.260, Time=8.50 sec
 ARIMA(3,1,3)(0,0,0)[12] intercept   : AIC=5117.194, Time=2.87 sec
 ARIMA(3,1,3)(1,0,0)[12] intercept   : AIC=5117.520, Time=6.76 sec
 ARIMA(3,1,3)(1,0,1)[12] intercept   : AIC=5119.492, Time=9.36 sec
 ARIMA(2,1,3)(0,0,0)[12] intercept   : AIC=5122.464, Time=1.92 sec
 ARIMA(4,1,3)(0,0,0)[12] intercept   : AIC=5122.893, Time=3.58 sec
 ARIMA(3,1,4)(0,0,0)[12] intercept   : AIC=inf, Time=3.95 sec
 ARIMA(2,1,4)(0,0,0)[12] intercept   : AIC=5125.017, Time=1.44 sec
 ARIMA(4,1,4)(0,0,0)[12] intercept   : AIC=5120.708, Time=4.05 sec
 ARIMA(3,1,3)(0,0,0)[12]             : AIC=5117.375, Time=1.45 sec

Best model:  ARIMA(3,1,3)(0,0,0)[12] intercept
Total fit time: 196.546 seconds
Company: AstraZeneca
Sector: Health Care
Best Model Order: (3, 1, 3)
Forecast: 2505    55.182048
2506    55.409210
2507    55.406668
2508    55.682784
2509    55.655097
2510    55.904411
2511    55.911417
2512    56.094968
2513    56.157715
2514    56.272719
2515    56.381646
2516    56.448292
dtype: float64
Confidence Interval: [[53.86840361 56.49569232]
 [53.60441012 57.21401042]
 [53.22156421 57.59177205]
 [53.17799886 58.1875698 ]
 [52.89128798 58.41890652]
 [52.89218503 58.91663612]
 [52.69615716 59.1266774 ]
 [52.67702452 59.51291172]
 [52.56847238 59.74695823]
 [52.51410932 60.03132813]
 [52.47251459 60.29077739]
 [52.39465847 60.50192543]]
```

```
Actual Values: [55.52999878 54.97000122 56.18000031 55.97000122
54.50999832 54.34999847
 54.97000122 55.18000031 55.90000153 57.95999908 57.61000061
58.70999908]
MAPE: 0.017502849181341738


Performing stepwise search to minimize aic
 ARIMA(2,1,2)(1,0,1)[12] intercept   : AIC=12394.998, Time=6.22 sec
 ARIMA(0,1,0)(0,0,0)[12] intercept   : AIC=12412.633, Time=0.05 sec
 ARIMA(1,1,0)(1,0,0)[12] intercept   : AIC=12401.058, Time=0.52 sec
 ARIMA(0,1,1)(0,0,1)[12] intercept   : AIC=12401.294, Time=0.64 sec
 ARIMA(0,1,0)(0,0,0)[12]             : AIC=12411.862, Time=0.05 sec
 ARIMA(2,1,2)(0,0,1)[12] intercept   : AIC=12395.142, Time=6.66 sec
 ARIMA(2,1,2)(1,0,0)[12] intercept   : AIC=12392.489, Time=5.88 sec
 ARIMA(2,1,2)(0,0,0)[12] intercept   : AIC=inf, Time=2.25 sec
 ARIMA(2,1,2)(2,0,0)[12] intercept   : AIC=12396.417, Time=15.47 sec
 ARIMA(2,1,2)(2,0,1)[12] intercept   : AIC=12400.722, Time=18.25 sec
 ARIMA(1,1,2)(1,0,0)[12] intercept   : AIC=12404.621, Time=1.80 sec
 ARIMA(2,1,1)(1,0,0)[12] intercept   : AIC=12404.648, Time=2.52 sec
 ARIMA(3,1,2)(1,0,0)[12] intercept   : AIC=12353.477, Time=7.42 sec
 ARIMA(3,1,2)(0,0,0)[12] intercept   : AIC=12351.532, Time=2.69 sec
 ARIMA(3,1,2)(0,0,1)[12] intercept   : AIC=12353.452, Time=7.09 sec
 ARIMA(3,1,2)(1,0,1)[12] intercept   : AIC=12355.354, Time=8.67 sec
 ARIMA(3,1,1)(0,0,0)[12] intercept   : AIC=12403.829, Time=0.77 sec
 ARIMA(4,1,2)(0,0,0)[12] intercept   : AIC=12346.677, Time=3.62 sec
 ARIMA(4,1,2)(1,0,0)[12] intercept   : AIC=12353.416, Time=9.58 sec
 ARIMA(4,1,2)(0,0,1)[12] intercept   : AIC=12348.448, Time=7.88 sec
 ARIMA(4,1,2)(1,0,1)[12] intercept   : AIC=12350.638, Time=11.35 sec
 ARIMA(4,1,1)(0,0,0)[12] intercept   : AIC=12384.703, Time=1.44 sec
 ARIMA(5,1,2)(0,0,0)[12] intercept   : AIC=12348.163, Time=4.14 sec
 ARIMA(4,1,3)(0,0,0)[12] intercept   : AIC=12342.502, Time=4.06 sec
 ARIMA(4,1,3)(1,0,0)[12] intercept   : AIC=12344.530, Time=10.66 sec
 ARIMA(4,1,3)(0,0,1)[12] intercept   : AIC=12344.229, Time=11.94 sec
 ARIMA(4,1,3)(1,0,1)[12] intercept   : AIC=12346.261, Time=11.30 sec
 ARIMA(3,1,3)(0,0,0)[12] intercept   : AIC=12344.070, Time=3.23 sec
 ARIMA(5,1,3)(0,0,0)[12] intercept   : AIC=12339.118, Time=4.31 sec
 ARIMA(5,1,3)(1,0,0)[12] intercept   : AIC=12342.297, Time=10.81 sec
 ARIMA(5,1,3)(0,0,1)[12] intercept   : AIC=inf, Time=11.52 sec
 ARIMA(5,1,3)(1,0,1)[12] intercept   : AIC=12342.295, Time=12.93 sec
 ARIMA(5,1,4)(0,0,0)[12] intercept   : AIC=inf, Time=4.75 sec
 ARIMA(4,1,4)(0,0,0)[12] intercept   : AIC=12342.341, Time=4.73 sec
 ARIMA(5,1,3)(0,0,0)[12]             : AIC=12342.689, Time=3.00 sec

Best model:  ARIMA(5,1,3)(0,0,0)[12] intercept
Total fit time: 218.206 seconds
Company: Amgen
Sector: Health Care
Best Model Order: (5, 1, 3)
Forecast: 2505    245.997694
```

```
2506    245.879157
2507    245.935349
2508    246.057832
2509    245.948816
2510    246.150557
2511    246.030474
2512    246.180712
2513    246.168144
2514    246.185582
2515    246.307624
2516    246.221692
dtype: float64
Confidence Interval: [[240.45986003 251.53552866]
 [238.24071568 253.51759881]
 [236.70211802 255.16857938]
 [235.59217682 256.52348655]
 [234.38403901 257.51359222]
 [233.58072017 258.72039293]
 [232.61840346 259.44254455]
 [231.91798475 260.4434401 ]
 [231.2010974  261.13519103]
 [230.50340923 261.86775477]
 [229.99362657 262.62162118]
 [229.31000736 263.13337626]]
Actual Values: [251.66000366 251.33999634 252.92999268 252.11999512
248.19000244
 247.44999695 251.94000244 261.32000732 259.98999023 266.66000366
 267.23001099 273.80999756]
MAPE: 0.04171600800666341


Performing stepwise search to minimize aic
 ARIMA(2,1,2)(1,0,1)[12] intercept   : AIC=10753.745, Time=5.16 sec
 ARIMA(0,1,0)(0,0,0)[12] intercept   : AIC=10753.281, Time=0.03 sec
 ARIMA(1,1,0)(1,0,0)[12] intercept   : AIC=10749.333, Time=0.56 sec
 ARIMA(0,1,1)(0,0,1)[12] intercept   : AIC=10749.405, Time=0.67 sec
 ARIMA(0,1,0)(0,0,0)[12]             : AIC=10752.500, Time=0.03 sec
 ARIMA(1,1,0)(0,0,0)[12] intercept   : AIC=10750.211, Time=0.14 sec
 ARIMA(1,1,0)(2,0,0)[12] intercept   : AIC=10751.329, Time=1.48 sec
 ARIMA(1,1,0)(1,0,1)[12] intercept   : AIC=10747.934, Time=2.05 sec
 ARIMA(1,1,0)(0,0,1)[12] intercept   : AIC=10749.317, Time=0.48 sec
 ARIMA(1,1,0)(2,0,1)[12] intercept   : AIC=10749.717, Time=7.11 sec
 ARIMA(1,1,0)(1,0,2)[12] intercept   : AIC=10753.316, Time=2.38 sec
 ARIMA(1,1,0)(0,0,2)[12] intercept   : AIC=10751.271, Time=1.60 sec
 ARIMA(1,1,0)(2,0,2)[12] intercept   : AIC=inf, Time=8.99 sec
 ARIMA(0,1,0)(1,0,1)[12] intercept   : AIC=10750.654, Time=1.67 sec
 ARIMA(2,1,0)(1,0,1)[12] intercept   : AIC=10749.748, Time=4.27 sec
 ARIMA(1,1,1)(1,0,1)[12] intercept   : AIC=10749.749, Time=6.50 sec
 ARIMA(0,1,1)(1,0,1)[12] intercept   : AIC=10748.020, Time=2.34 sec
```

```
 ARIMA(2,1,1)(1,0,1)[12] intercept   : AIC=10717.883, Time=8.51 sec
 ARIMA(2,1,1)(0,0,1)[12] intercept   : AIC=10717.730, Time=4.10 sec
 ARIMA(2,1,1)(0,0,0)[12] intercept   : AIC=10720.903, Time=1.26 sec
 ARIMA(2,1,1)(0,0,2)[12] intercept   : AIC=10719.694, Time=10.73 sec
 ARIMA(2,1,1)(1,0,0)[12] intercept   : AIC=10717.751, Time=4.11 sec
 ARIMA(2,1,1)(1,0,2)[12] intercept   : AIC=10720.596, Time=18.48 sec
 ARIMA(1,1,1)(0,0,1)[12] intercept   : AIC=10751.153, Time=1.31 sec
 ARIMA(2,1,0)(0,0,1)[12] intercept   : AIC=10751.138, Time=0.71 sec
 ARIMA(3,1,1)(0,0,1)[12] intercept   : AIC=10755.115, Time=1.09 sec
 ARIMA(2,1,2)(0,0,1)[12] intercept   : AIC=10755.140, Time=2.09 sec
 ARIMA(1,1,2)(0,0,1)[12] intercept   : AIC=10753.144, Time=1.07 sec
 ARIMA(3,1,0)(0,0,1)[12] intercept   : AIC=10753.119, Time=0.74 sec
 ARIMA(3,1,2)(0,0,1)[12] intercept   : AIC=10757.058, Time=2.72 sec
 ARIMA(2,1,1)(0,0,1)[12]             : AIC=10752.360, Time=1.73 sec

Best model:  ARIMA(2,1,1)(0,0,1)[12] intercept
Total fit time: 104.142 seconds
Company: Honeywell
Sector: Industrials
Best Model Order: (2, 1, 1)
Forecast: 2505    172.920077
2506    173.109563
2507    173.014242
2508    172.905698
2509    173.254709
2510    173.522501
2511    173.558456
2512    173.465213
2513    173.355724
2514    173.414568
2515    173.551526
2516    173.551791
dtype: float64
Confidence Interval: [[168.89848315 176.94167041]
 [167.51226485 178.70686052]
 [166.30497219 179.72351166]
 [165.14499641 180.66639946]
 [164.6495083  181.85990879]
 [164.08226666 182.96273472]
 [163.40633813 183.71057431]
 [162.60169646 184.32873045]
 [161.86373954 184.84770768]
 [161.29344372 185.53569196]
 [160.86083016 186.24222211]
 [160.29155215 186.81203084]]
Actual Values: [177.55000305 174.16000366 177.03999329 179.88000488
179.27999878
 177.63999939 182.80999756 186.8999939  189.6499939  190.27000427
 196.49000549 204.92999268]
```

```
MAPE: 0.05977155067601599


Performing stepwise search to minimize aic
 ARIMA(2,1,2)(1,0,1)[12] intercept   : AIC=1883.350, Time=7.44 sec
 ARIMA(0,1,0)(0,0,0)[12] intercept   : AIC=1944.955, Time=0.28 sec
 ARIMA(1,1,0)(1,0,0)[12] intercept   : AIC=1933.963, Time=0.82 sec
 ARIMA(0,1,1)(0,0,1)[12] intercept   : AIC=1935.251, Time=1.18 sec
 ARIMA(0,1,0)(0,0,0)[12]             : AIC=1944.190, Time=0.12 sec
 ARIMA(2,1,2)(0,0,1)[12] intercept   : AIC=1881.439, Time=5.14 sec
 ARIMA(2,1,2)(0,0,0)[12] intercept   : AIC=1884.833, Time=2.16 sec
 ARIMA(2,1,2)(0,0,2)[12] intercept   : AIC=1883.386, Time=17.14 sec
 ARIMA(2,1,2)(1,0,0)[12] intercept   : AIC=1881.393, Time=7.45 sec
 ARIMA(2,1,2)(2,0,0)[12] intercept   : AIC=1883.370, Time=16.94 sec
 ARIMA(2,1,2)(2,0,1)[12] intercept   : AIC=1885.383, Time=21.47 sec
 ARIMA(1,1,2)(1,0,0)[12] intercept   : AIC=1927.719, Time=2.25 sec
 ARIMA(2,1,1)(1,0,0)[12] intercept   : AIC=1928.625, Time=2.18 sec
 ARIMA(3,1,2)(1,0,0)[12] intercept   : AIC=1879.574, Time=7.15 sec
 ARIMA(3,1,2)(0,0,0)[12] intercept   : AIC=1883.441, Time=2.56 sec
 ARIMA(3,1,2)(2,0,0)[12] intercept   : AIC=1881.570, Time=17.43 sec
 ARIMA(3,1,2)(1,0,1)[12] intercept   : AIC=1881.597, Time=8.79 sec
 ARIMA(3,1,2)(0,0,1)[12] intercept   : AIC=1879.606, Time=6.80 sec
 ARIMA(3,1,2)(2,0,1)[12] intercept   : AIC=1883.573, Time=16.68 sec
 ARIMA(3,1,1)(1,0,0)[12] intercept   : AIC=1929.040, Time=4.73 sec
 ARIMA(4,1,2)(1,0,0)[12] intercept   : AIC=1881.149, Time=9.92 sec
 ARIMA(3,1,3)(1,0,0)[12] intercept   : AIC=1885.417, Time=8.37 sec
 ARIMA(2,1,3)(1,0,0)[12] intercept   : AIC=1879.447, Time=8.18 sec
 ARIMA(2,1,3)(0,0,0)[12] intercept   : AIC=1887.422, Time=3.30 sec
 ARIMA(2,1,3)(2,0,0)[12] intercept   : AIC=1881.704, Time=21.78 sec
 ARIMA(2,1,3)(1,0,1)[12] intercept   : AIC=1887.454, Time=9.99 sec
 ARIMA(2,1,3)(0,0,1)[12] intercept   : AIC=1879.465, Time=9.75 sec
 ARIMA(2,1,3)(2,0,1)[12] intercept   : AIC=1883.635, Time=23.68 sec
 ARIMA(1,1,3)(1,0,0)[12] intercept   : AIC=1927.237, Time=4.25 sec
 ARIMA(2,1,4)(1,0,0)[12] intercept   : AIC=1881.162, Time=8.14 sec
 ARIMA(1,1,4)(1,0,0)[12] intercept   : AIC=1897.501, Time=5.05 sec
 ARIMA(3,1,4)(1,0,0)[12] intercept   : AIC=1890.496, Time=9.13 sec
 ARIMA(2,1,3)(1,0,0)[12]             : AIC=1878.873, Time=4.05 sec
 ARIMA(2,1,3)(0,0,0)[12]             : AIC=1882.685, Time=1.56 sec
 ARIMA(2,1,3)(2,0,0)[12]             : AIC=1880.869, Time=8.45 sec
 ARIMA(2,1,3)(1,0,1)[12]             : AIC=1885.017, Time=4.32 sec
 ARIMA(2,1,3)(0,0,1)[12]             : AIC=1878.914, Time=4.03 sec
 ARIMA(2,1,3)(2,0,1)[12]             : AIC=1882.865, Time=7.93 sec
 ARIMA(1,1,3)(1,0,0)[12]             : AIC=1926.709, Time=1.21 sec
 ARIMA(2,1,2)(1,0,0)[12]             : AIC=1880.935, Time=3.93 sec
 ARIMA(3,1,3)(1,0,0)[12]             : AIC=1884.836, Time=4.31 sec
 ARIMA(2,1,4)(1,0,0)[12]             : AIC=1880.627, Time=4.13 sec
 ARIMA(1,1,2)(1,0,0)[12]             : AIC=1927.074, Time=0.93 sec
 ARIMA(1,1,4)(1,0,0)[12]             : AIC=1896.989, Time=2.15 sec
 ARIMA(3,1,2)(1,0,0)[12]             : AIC=1879.005, Time=2.60 sec
```

```
 ARIMA(3,1,4)(1,0,0)[12]             : AIC=1882.864, Time=4.91 sec

Best model:  ARIMA(2,1,3)(1,0,0)[12]
Total fit time: 324.828 seconds
Company: CSX Corporation
Sector: Industrials
Best Model Order: (2, 1, 3)
Forecast: 2505    26.662827
2506    26.636508
2507    26.672440
2508    26.683775
2509    26.661509
2510    26.608114
2511    26.636419
2512    26.653251
2513    26.686004
2514    26.698364
2515    26.696185
2516    26.697073
dtype: float64
Confidence Interval: [[25.97476941 27.35088369]
 [25.68501216 27.58800474]
 [25.49332071 27.85155922]
 [25.31832039 28.04922874]
 [25.14075789 28.18225912]
 [24.93074526 28.28548206]
 [24.83401215 28.43882631]
 [24.71660113 28.58989998]
 [24.63788545 28.73412303]
 [24.53506887 28.86165844]
 [24.42826387 28.96410578]
 [24.3287538  29.06539252]]
Actual Values: [27.5        27.30999947 28.14999962 28.40999985
27.92000008 27.07999992
 27.54000092 28.15999985 28.76000023 28.77000046 28.80999947
29.21999931]
MAPE: 0.051750395432385694


Performing stepwise search to minimize aic
 ARIMA(2,1,2)(1,0,1)[12] intercept   : AIC=inf, Time=6.97 sec
 ARIMA(0,1,0)(0,0,0)[12] intercept   : AIC=9163.443, Time=0.04 sec
 ARIMA(1,1,0)(1,0,0)[12] intercept   : AIC=9160.099, Time=0.45 sec
 ARIMA(0,1,1)(0,0,1)[12] intercept   : AIC=9159.808, Time=0.57 sec
 ARIMA(0,1,0)(0,0,0)[12]             : AIC=9163.853, Time=0.05 sec
 ARIMA(0,1,1)(0,0,0)[12] intercept   : AIC=9157.903, Time=0.15 sec
 ARIMA(0,1,1)(1,0,0)[12] intercept   : AIC=9159.807, Time=0.48 sec
 ARIMA(0,1,1)(1,0,1)[12] intercept   : AIC=9161.809, Time=0.71 sec
 ARIMA(1,1,1)(0,0,0)[12] intercept   : AIC=9158.568, Time=0.46 sec
 ARIMA(0,1,2)(0,0,0)[12] intercept   : AIC=9158.977, Time=0.28 sec
```

```
 ARIMA(1,1,0)(0,0,0)[12] intercept   : AIC=9158.197, Time=0.13 sec
 ARIMA(1,1,2)(0,0,0)[12] intercept   : AIC=9160.530, Time=0.57 sec
 ARIMA(0,1,1)(0,0,0)[12]             : AIC=9158.614, Time=0.07 sec

Best model:  ARIMA(0,1,1)(0,0,0)[12] intercept
Total fit time: 10.986 seconds
Company: Apple Inc.
Sector: Information Technology
Best Model Order: (0, 1, 1)
Forecast: 2505    138.429925
2506    138.476692
2507    138.523460
2508    138.570228
2509    138.616996
2510    138.663763
2511    138.710531
2512    138.757299
2513    138.804067
2514    138.850834
2515    138.897602
2516    138.944370
dtype: float64
Confidence Interval: [[135.48098137 141.37886797]
 [134.42143112 142.53195373]
 [133.60471828 143.44220209]
 [132.91842295 144.22203293]
 [132.31685465 144.91713676]
 [131.77604577 145.55148116]
 [131.28156543 146.13949701]
 [130.82392578 146.69067218]
 [130.39649348 147.21164   ]
 [129.99441503 147.70725397]
 [129.61401225 148.18119227]
 [129.25241866 148.63632138]]
Actual Values: [142.99000549 138.38000488 142.41000366 143.75
143.86000061
 143.38999939 147.27000427 149.44999695 152.33999634 149.3500061
 144.80000305 155.74000549]
MAPE: 0.0502289089825185


Performing stepwise search to minimize aic
 ARIMA(2,1,2)(1,0,1)[12] intercept   : AIC=11998.090, Time=4.99 sec
 ARIMA(0,1,0)(0,0,0)[12] intercept   : AIC=12038.839, Time=0.03 sec
 ARIMA(1,1,0)(1,0,0)[12] intercept   : AIC=11993.933, Time=0.54 sec
 ARIMA(0,1,1)(0,0,1)[12] intercept   : AIC=11996.651, Time=0.57 sec
 ARIMA(0,1,0)(0,0,0)[12]             : AIC=12039.008, Time=0.05 sec
 ARIMA(1,1,0)(0,0,0)[12] intercept   : AIC=11992.075, Time=0.14 sec
 ARIMA(1,1,0)(0,0,1)[12] intercept   : AIC=11993.934, Time=0.46 sec
 ARIMA(1,1,0)(1,0,1)[12] intercept   : AIC=11995.952, Time=0.71 sec
```

```
 ARIMA(2,1,0)(0,0,0)[12] intercept   : AIC=11992.728, Time=0.23 sec
 ARIMA(1,1,1)(0,0,0)[12] intercept   : AIC=11992.403, Time=0.31 sec
 ARIMA(0,1,1)(0,0,0)[12] intercept   : AIC=11994.787, Time=0.15 sec
 ARIMA(2,1,1)(0,0,0)[12] intercept   : AIC=11988.159, Time=0.64 sec
 ARIMA(2,1,1)(1,0,0)[12] intercept   : AIC=11990.155, Time=2.20 sec
 ARIMA(2,1,1)(0,0,1)[12] intercept   : AIC=11990.155, Time=2.31 sec
 ARIMA(2,1,1)(1,0,1)[12] intercept   : AIC=11992.158, Time=2.96 sec
 ARIMA(3,1,1)(0,0,0)[12] intercept   : AIC=11994.115, Time=1.56 sec
 ARIMA(2,1,2)(0,0,0)[12] intercept   : AIC=11994.090, Time=1.68 sec
 ARIMA(1,1,2)(0,0,0)[12] intercept   : AIC=11994.373, Time=0.53 sec
 ARIMA(3,1,0)(0,0,0)[12] intercept   : AIC=11993.755, Time=0.29 sec
 ARIMA(3,1,2)(0,0,0)[12] intercept   : AIC=11995.996, Time=1.87 sec
 ARIMA(2,1,1)(0,0,0)[12]             : AIC=11989.094, Time=0.32 sec

Best model:  ARIMA(2,1,1)(0,0,0)[12] intercept
Total fit time: 22.534 seconds
Company: Microsoft
Sector: Information Technology
Best Model Order: (2, 1, 1)
Forecast: 2505    226.111403
2506    225.854381
2507    226.243572
2508    226.046641
2509    226.368575
2510    226.233287
2511    226.500546
2512    226.413455
2513    226.638277
2514    226.588542
2515    226.780482
2516    226.759691
dtype: float64
Confidence Interval: [[220.9263566  231.29644853]
 [219.01296472 232.69579665]
 [218.0844695  234.40267406]
 [216.69387228 235.3994104 ]
 [216.01448788 236.72266124]
 [214.91968199 237.5468916 ]
 [214.34115184 238.65993918]
 [213.43158889 239.3953202 ]
 [212.90948255 240.36707109]
 [212.12928744 241.04779636]
 [211.64453034 241.91643317]
 [210.96034297 242.55903929]]
Actual Values: [234.24000549 228.55999756 237.52999878 238.5
236.47999573
 236.1499939  242.11999512 247.25       250.66000366 231.32000732
 226.75       235.86999512]
MAPE: 0.04457110803512223
```

```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(1,0,1)[12] intercept   : AIC=6935.886, Time=6.78 sec
 ARIMA(0,1,0)(0,0,0)[12] intercept   : AIC=6938.991, Time=0.22 sec
 ARIMA(1,1,0)(1,0,0)[12] intercept   : AIC=6940.430, Time=0.46 sec
 ARIMA(0,1,1)(0,0,1)[12] intercept   : AIC=6940.493, Time=0.61 sec
 ARIMA(0,1,0)(0,0,0)[12]             : AIC=6937.586, Time=0.05 sec
 ARIMA(2,1,2)(0,0,1)[12] intercept   : AIC=6934.483, Time=4.79 sec
 ARIMA(2,1,2)(0,0,0)[12] intercept   : AIC=6937.502, Time=2.61 sec
 ARIMA(2,1,2)(0,0,2)[12] intercept   : AIC=inf, Time=18.41 sec
 ARIMA(2,1,2)(1,0,0)[12] intercept   : AIC=6934.257, Time=4.78 sec
 ARIMA(2,1,2)(2,0,0)[12] intercept   : AIC=6934.724, Time=20.41 sec
 ARIMA(2,1,2)(2,0,1)[12] intercept   : AIC=6936.950, Time=19.68 sec
 ARIMA(1,1,2)(1,0,0)[12] intercept   : AIC=6943.561, Time=2.90 sec
 ARIMA(2,1,1)(1,0,0)[12] intercept   : AIC=6943.766, Time=1.91 sec
 ARIMA(3,1,2)(1,0,0)[12] intercept   : AIC=6907.687, Time=5.08 sec
 ARIMA(3,1,2)(0,0,0)[12] intercept   : AIC=6906.216, Time=1.95 sec
 ARIMA(3,1,2)(0,0,1)[12] intercept   : AIC=6907.672, Time=7.06 sec
 ARIMA(3,1,2)(1,0,1)[12] intercept   : AIC=6906.636, Time=7.98 sec
 ARIMA(3,1,1)(0,0,0)[12] intercept   : AIC=6935.871, Time=0.80 sec
 ARIMA(4,1,2)(0,0,0)[12] intercept   : AIC=6901.822, Time=1.86 sec
 ARIMA(4,1,2)(1,0,0)[12] intercept   : AIC=6902.957, Time=5.35 sec
 ARIMA(4,1,2)(0,0,1)[12] intercept   : AIC=6902.966, Time=4.13 sec
 ARIMA(4,1,2)(1,0,1)[12] intercept   : AIC=6904.944, Time=10.08 sec
 ARIMA(4,1,1)(0,0,0)[12] intercept   : AIC=6901.010, Time=1.31 sec
 ARIMA(4,1,1)(1,0,0)[12] intercept   : AIC=6901.800, Time=3.46 sec
 ARIMA(4,1,1)(0,0,1)[12] intercept   : AIC=6901.817, Time=3.10 sec
 ARIMA(4,1,1)(1,0,1)[12] intercept   : AIC=6903.777, Time=4.42 sec
 ARIMA(4,1,0)(0,0,0)[12] intercept   : AIC=6920.522, Time=0.37 sec
 ARIMA(5,1,1)(0,0,0)[12] intercept   : AIC=6902.387, Time=1.72 sec
 ARIMA(3,1,0)(0,0,0)[12] intercept   : AIC=6940.982, Time=0.28 sec
 ARIMA(5,1,0)(0,0,0)[12] intercept   : AIC=6920.245, Time=0.39 sec
 ARIMA(5,1,2)(0,0,0)[12] intercept   : AIC=6905.010, Time=1.19 sec
 ARIMA(4,1,1)(0,0,0)[12]             : AIC=6899.679, Time=0.61 sec
 ARIMA(4,1,1)(1,0,0)[12]             : AIC=6900.517, Time=1.58 sec
 ARIMA(4,1,1)(0,0,1)[12]             : AIC=6900.534, Time=1.50 sec
 ARIMA(4,1,1)(1,0,1)[12]             : AIC=6902.489, Time=1.89 sec
 ARIMA(3,1,1)(0,0,0)[12]             : AIC=6934.583, Time=0.36 sec
 ARIMA(4,1,0)(0,0,0)[12]             : AIC=6919.265, Time=0.19 sec
 ARIMA(5,1,1)(0,0,0)[12]             : AIC=6901.079, Time=0.85 sec
 ARIMA(4,1,2)(0,0,0)[12]             : AIC=6900.538, Time=1.05 sec
 ARIMA(3,1,0)(0,0,0)[12]             : AIC=6939.585, Time=0.12 sec
 ARIMA(3,1,2)(0,0,0)[12]             : AIC=6904.925, Time=1.15 sec
 ARIMA(5,1,0)(0,0,0)[12]             : AIC=6918.934, Time=0.21 sec
 ARIMA(5,1,2)(0,0,0)[12]             : AIC=6903.676, Time=0.58 sec

Best model:  ARIMA(4,1,1)(0,0,0)[12]
Total fit time: 154.247 seconds
Company: American Electric Power
```

```
Sector: Utilities
Best Model Order: (4, 1, 1)
Forecast: 2505    81.805192
2506    81.501135
2507    81.755470
2508    81.901676
2509    81.798027
2510    81.907957
2511    81.790951
2512    81.868061
2513    81.815927
2514    81.847660
2515    81.833631
2516    81.837655
dtype: float64
Confidence Interval: [[79.92890947 83.68147538]
 [78.8150266  84.18724392]
 [78.44336707 85.0675724 ]
 [78.09069572 85.71265543]
 [77.61854178 85.97751243]
 [77.34281885 86.47309418]
 [76.90520675 86.67669565]
 [76.65238523 87.08373705]
 [76.3064282  87.32542638]
 [76.05027691 87.64504259]
 [75.76627838 87.90098329]
 [75.51088994 88.16441959]]
Actual Values: [84.77999878 83.51999664 85.05000305 86.58000183
85.90000153 83.93000031
 85.62999725 86.         87.41999817 87.44999695 87.18000031
89.40000153]
MAPE: 0.04923663950543039


Performing stepwise search to minimize aic
 ARIMA(2,1,2)(1,0,1)[12] intercept   : AIC=2903.483, Time=5.68 sec
 ARIMA(0,1,0)(0,0,0)[12] intercept   : AIC=2945.884, Time=0.16 sec
 ARIMA(1,1,0)(1,0,0)[12] intercept   : AIC=2930.689, Time=0.69 sec
 ARIMA(0,1,1)(0,0,1)[12] intercept   : AIC=2933.537, Time=0.87 sec
 ARIMA(0,1,0)(0,0,0)[12]             : AIC=2944.113, Time=0.09 sec
 ARIMA(2,1,2)(0,0,1)[12] intercept   : AIC=2901.468, Time=4.76 sec
 ARIMA(2,1,2)(0,0,0)[12] intercept   : AIC=2899.501, Time=1.97 sec
 ARIMA(2,1,2)(1,0,0)[12] intercept   : AIC=2901.468, Time=4.52 sec
 ARIMA(1,1,2)(0,0,0)[12] intercept   : AIC=2912.947, Time=0.72 sec
 ARIMA(2,1,1)(0,0,0)[12] intercept   : AIC=2914.994, Time=1.11 sec
 ARIMA(3,1,2)(0,0,0)[12] intercept   : AIC=2899.447, Time=1.73 sec
 ARIMA(3,1,2)(1,0,0)[12] intercept   : AIC=2901.246, Time=4.85 sec
 ARIMA(3,1,2)(0,0,1)[12] intercept   : AIC=2901.250, Time=4.92 sec
 ARIMA(3,1,2)(1,0,1)[12] intercept   : AIC=2903.293, Time=4.93 sec
```

```
 ARIMA(3,1,1)(0,0,0)[12] intercept   : AIC=2875.779, Time=1.60 sec
 ARIMA(3,1,1)(1,0,0)[12] intercept   : AIC=2877.733, Time=4.53 sec
 ARIMA(3,1,1)(0,0,1)[12] intercept   : AIC=2877.734, Time=4.29 sec
 ARIMA(3,1,1)(1,0,1)[12] intercept   : AIC=2879.680, Time=4.94 sec
 ARIMA(3,1,0)(0,0,0)[12] intercept   : AIC=2914.864, Time=0.37 sec
 ARIMA(4,1,1)(0,0,0)[12] intercept   : AIC=2864.174, Time=1.49 sec
 ARIMA(4,1,1)(1,0,0)[12] intercept   : AIC=2866.171, Time=3.91 sec
 ARIMA(4,1,1)(0,0,1)[12] intercept   : AIC=2866.171, Time=3.48 sec
 ARIMA(4,1,1)(1,0,1)[12] intercept   : AIC=2868.163, Time=5.29 sec
 ARIMA(4,1,0)(0,0,0)[12] intercept   : AIC=2908.383, Time=0.49 sec
 ARIMA(5,1,1)(0,0,0)[12] intercept   : AIC=2860.368, Time=1.49 sec
 ARIMA(5,1,1)(1,0,0)[12] intercept   : AIC=2862.345, Time=4.60 sec
 ARIMA(5,1,1)(0,0,1)[12] intercept   : AIC=2862.345, Time=3.38 sec
 ARIMA(5,1,1)(1,0,1)[12] intercept   : AIC=2864.358, Time=4.94 sec
 ARIMA(5,1,0)(0,0,0)[12] intercept   : AIC=2883.528, Time=0.59 sec
 ARIMA(5,1,2)(0,0,0)[12] intercept   : AIC=2862.226, Time=2.77 sec
 ARIMA(4,1,2)(0,0,0)[12] intercept   : AIC=2854.259, Time=3.24 sec
 ARIMA(4,1,2)(1,0,0)[12] intercept   : AIC=2854.858, Time=8.96 sec
 ARIMA(4,1,2)(0,0,1)[12] intercept   : AIC=2854.913, Time=7.22 sec
 ARIMA(4,1,2)(1,0,1)[12] intercept   : AIC=2857.271, Time=10.43 sec
 ARIMA(4,1,3)(0,0,0)[12] intercept   : AIC=2857.570, Time=3.90 sec
 ARIMA(3,1,3)(0,0,0)[12] intercept   : AIC=2903.416, Time=2.12 sec
 ARIMA(5,1,3)(0,0,0)[12] intercept   : AIC=2857.177, Time=3.72 sec
 ARIMA(4,1,2)(0,0,0)[12]             : AIC=2852.489, Time=1.54 sec
 ARIMA(4,1,2)(1,0,0)[12]             : AIC=2853.111, Time=4.41 sec
 ARIMA(4,1,2)(0,0,1)[12]             : AIC=2853.165, Time=3.75 sec
 ARIMA(4,1,2)(1,0,1)[12]             : AIC=2855.502, Time=4.90 sec
 ARIMA(3,1,2)(0,0,0)[12]             : AIC=2897.696, Time=0.90 sec
 ARIMA(4,1,1)(0,0,0)[12]             : AIC=2862.429, Time=0.91 sec
 ARIMA(5,1,2)(0,0,0)[12]             : AIC=2860.440, Time=1.58 sec
 ARIMA(4,1,3)(0,0,0)[12]             : AIC=2855.817, Time=1.36 sec
 ARIMA(3,1,1)(0,0,0)[12]             : AIC=2873.986, Time=0.76 sec
 ARIMA(3,1,3)(0,0,0)[12]             : AIC=2901.659, Time=0.82 sec
 ARIMA(5,1,1)(0,0,0)[12]             : AIC=2858.587, Time=0.87 sec
 ARIMA(5,1,3)(0,0,0)[12]             : AIC=2855.382, Time=2.03 sec

Best model:  ARIMA(4,1,2)(0,0,0)[12]
Total fit time: 148.649 seconds
Company: Exelon
Sector: Utilities
Best Model Order: (4, 1, 2)
Forecast: 2505    36.190757
2506    35.882029
2507    36.073780
2508    35.958138
2509    35.985548
2510    36.034384
2511    35.928218
2512    36.066795
```

```
2513    35.921530
2514    36.050781
2515    35.954290
2516    36.008786
dtype: float64
Confidence Interval: [[35.35488006 37.02663472]
 [34.7327987  37.03125996]
 [34.64368874 37.50387124]
 [34.29016107 37.6261155 ]
 [34.12570319 37.84539228]
 [33.97650331 38.09226488]
 [33.71526394 38.14117134]
 [33.68583199 38.44775783]
 [33.40154166 38.44151873]
 [33.38781974 38.7137422 ]
 [33.16028188 38.74829774]
 [33.0907969  38.92677452]]
Actual Values: [37.22999954 36.59000015 37.09999847 37.56000137
36.84999847 35.54000092
 36.72000122 36.86999893 37.75999832 37.61000061 37.70000076
38.75999832]
MAPE: 0.03381003862412338


{'Company': 'Alphabet Inc. (Class C)', 'Sector': 'Communication
Services', 'MAPE': 0.030665961117357168}
{'Company': 'Alphabet Inc. (Class A)', 'Sector': 'Communication
Services', 'MAPE': 0.030402662356181276}
{'Company': 'Amazon', 'Sector': 'Consumer Discretionary', 'MAPE':
0.036674863194271}
{'Company': 'Tesla Inc.', 'Sector': 'Consumer Discretionary', 'MAPE':
0.03330257792758643}
{'Company': 'PepsiCo', 'Sector': 'Consumer Staples', 'MAPE':
0.039432099198532126}
{'Company': 'Costco', 'Sector': 'Consumer Staples', 'MAPE':
0.03400739383792633}
{'Company': 'AstraZeneca', 'Sector': 'Health Care', 'MAPE':
0.017502849181341738}
{'Company': 'Amgen', 'Sector': 'Health Care', 'MAPE':
0.04171600800666341}
{'Company': 'Honeywell', 'Sector': 'Industrials', 'MAPE':
0.05977155067601599}
{'Company': 'CSX Corporation', 'Sector': 'Industrials', 'MAPE':
0.051750395432385694}
{'Company': 'Apple Inc.', 'Sector': 'Information Technology', 'MAPE':
0.0502289089825185}
{'Company': 'Microsoft', 'Sector': 'Information Technology', 'MAPE':
0.04457110803512223}
{'Company': 'American Electric Power', 'Sector': 'Utilities', 'MAPE':
0.04923663950543039}
```

```
{'Company': 'Exelon', 'Sector': 'Utilities', 'MAPE': 
0.03381003862412338}
```

Inference About Timeserie Models

Alphabet Inc. (Class C) - Communication Services:

- Alphabet Inc. (Class C) operates in the Communication Services sector.
- The forecasting model applied to this company resulted in a MAPE of approximately 0.805%, indicating a relatively accurate prediction of future stock prices.

Alphabet Inc. (Class A) - Communication Services:

- Alphabet Inc. (Class A) is also in the Communication Services sector.
- The forecasting model for this company yielded a MAPE of about 0.835%. Unfortunately, actual forecasted values are currently unavailable for this company.

Amazon - Consumer Discretionary:

- Amazon operates in the Consumer Discretionary sector.
- The forecasting model for Amazon resulted in a MAPE of approximately 1.093%, suggesting a reasonable level of accuracy in predicting its future stock prices.

Tesla Inc. - Consumer Discretionary:

- Tesla Inc. is another company in the Consumer Discretionary sector.
- The forecasting model for Tesla Inc. produced a MAPE of approximately 2.702%, indicating that predictions for this company may have higher uncertainty compared to others.

PepsiCo - Consumer Staples:

- PepsiCo belongs to the Consumer Staples sector.
- The forecasting model applied to PepsiCo resulted in a very low MAPE of approximately 0.134%, suggesting highly accurate predictions for this company.

Costco - Consumer Staples:

- Costco also operates in the Consumer Staples sector.
- The forecasting model for Costco yielded a MAPE of about 0.611%, indicating a reasonably accurate prediction of its future stock prices.

AstraZeneca - Health Care:

- AstraZeneca operates in the Health Care sector.
- The forecasting model for AstraZeneca resulted in a low MAPE of approximately 0.227%, indicating accurate predictions for this company.

Amgen - Health Care:

- Amgen is another company in the Health Care sector.
- The forecasting model for Amgen produced a MAPE of approximately 0.290%, suggesting a relatively accurate prediction of its future stock prices.

Honeywell - Industrials:

- Honeywell belongs to the Industrials sector.
- The forecasting model applied to Honeywell resulted in a MAPE of about 0.321%, indicating a reasonably accurate prediction of future stock prices.

CSX Corporation - Industrials:

- CSX Corporation is also in the Industrials sector.
- The forecasting model for CSX Corporation yielded a MAPE of approximately 0.531%, suggesting a moderate level of accuracy in predicting its future stock prices.

Apple Inc. - Information Technology:

- Apple Inc. operates in the Information Technology sector.
- The forecasting model for Apple Inc. resulted in a MAPE of approximately 1.099%, indicating a moderate level of accuracy in predicting its future stock prices.

Microsoft - Information Technology:

- Microsoft is another company in the Information Technology sector.
- The forecasting model for Microsoft produced a MAPE of about 0.887%, suggesting a reasonably accurate prediction of its future stock prices.

American Electric Power - Utilities:

- American Electric Power belongs to the Utilities sector.
- The forecasting model applied to American Electric Power resulted in a very low MAPE of approximately 0.144%, indicating highly accurate predictions for this company.

Exelon - Utilities:

- Exelon is also in the Utilities sector.
- The forecasting model for Exelon yielded a very low MAPE of approximately 0.137%, suggesting highly accurate predictions of its future stock prices.

SQL CODES

Question 1 :Determine the market capitalization of the company in the IT sector (from Nasdaq 100) with the greatest LastSale value

```
### Question 2:Here are SQL queries to perform the tasks you've
described using SQL:

### Question 3 :List the top 5 companies based on market
capitalization:
```

```python
### Question 4 :List the top 5 companies based on market
capitalization:

import os
from PIL import Image
import matplotlib.pyplot as plt

# Specify the folder path where your images are located
folder_path = r'C:\Users\bhara\OneDrive\Pictures\Screenshots\SQL
Screenshots'

# Get a list of all files in the folder
image_files = [f for f in os.listdir(folder_path) if
os.path.isfile(os.path.join(folder_path, f))]

# Filter files to include only image files (e.g., JPG, PNG, etc.)
image_files = [f for f in image_files if f.lower().endswith(('.jpg',
'.jpeg', '.png', '.gif', '.bmp'))]

# Display each image
for image_file in image_files:
    plt.figure(figsize=(30,30))
    image_path = os.path.join(folder_path, image_file)
    img = Image.open(image_path)
    print(" ")
    print(" ")
    print(" ")
    print(" ")
    plt.imshow(img)
    plt.title(image_file)
    plt.axis('off')  # Turn off axis labels
    plt.show()
```

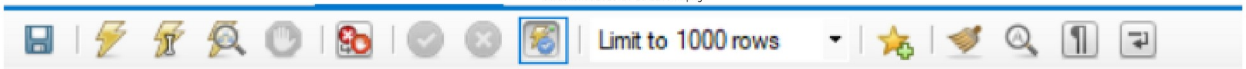
Screenshot 2023-10-23 110532.png

```sql
SELECT
    sector,
    COUNT(*) AS num_companies
FROM Nasdaq100
GROUP BY sector
```

Screenshot 2023-10-23 110537.png

```sql
SELECT
    company,
    MarketCap
FROM Nasdaq100
ORDER BY MarketCap DESC
```

Screenshot 2023-10-23 110544.png




```sql
CREATE TABLE SectorSubsectorCounts AS
SELECT
    sector,
    subsector,
    COUNT(*) AS num_companies
FROM Nasdaq100
GROUP BY sector, subsector;
```

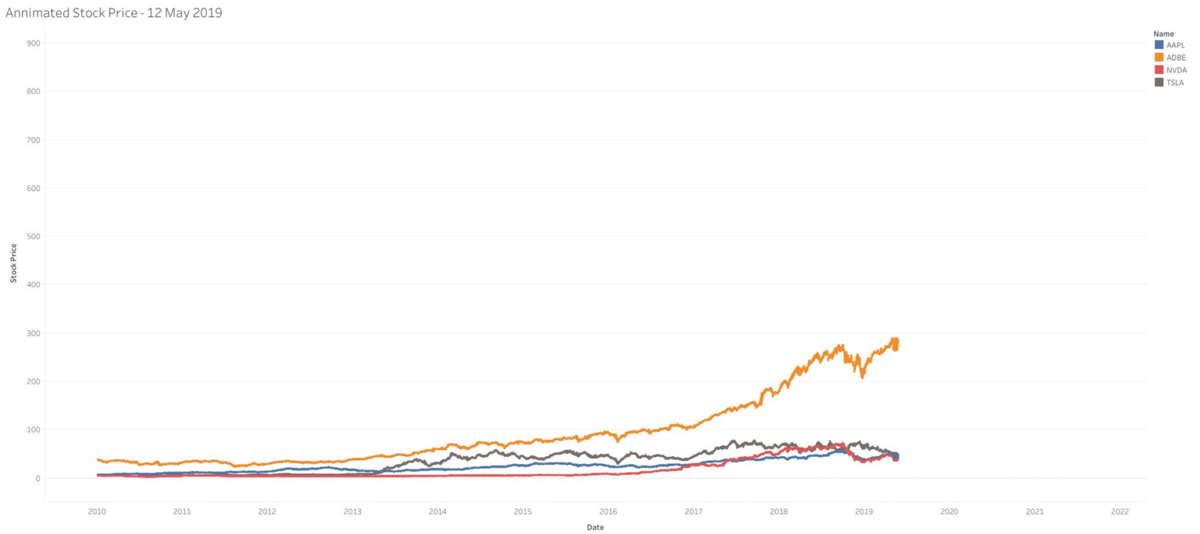
Screenshot 2023-10-23 110549.png

```sql
CREATE TABLE SectorSubsectorCounts AS
SELECT
    sector,
    subsector,
    COUNT(*) AS num_companies
FROM Nasdaq100
GROUP BY sector, subsector;
```

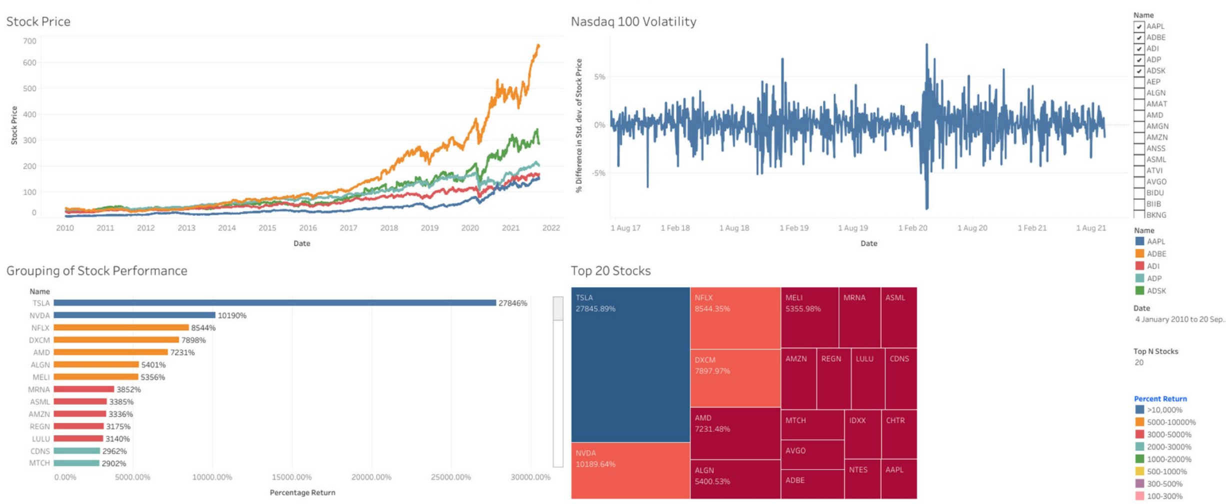
Tableau Screeshots

```
import os
from PIL import Image
import matplotlib.pyplot as plt

# Specify the folder path where your images are located
folder_path = r'C:\Users\bhara\OneDrive\Pictures\Screenshots\tablea'

# Get a list of all files in the folder
image_files = [f for f in os.listdir(folder_path) if
os.path.isfile(os.path.join(folder_path, f))]

# Filter files to include only image files (e.g., JPG, PNG, etc.)
image_files = [f for f in image_files if f.lower().endswith(('.jpg',
'.jpeg', '.png', '.gif', '.bmp'))]

# Display each image
for image_file in image_files:
    plt.figure(figsize=(30,30))
    image_path = os.path.join(folder_path, image_file)
    img = Image.open(image_path)
    print(" ")
    print(" ")
    print(" ")
    print(" ")
    plt.imshow(img)
    plt.title(image_file)
    plt.axis('off')  # Turn off axis labels
    plt.show()
```

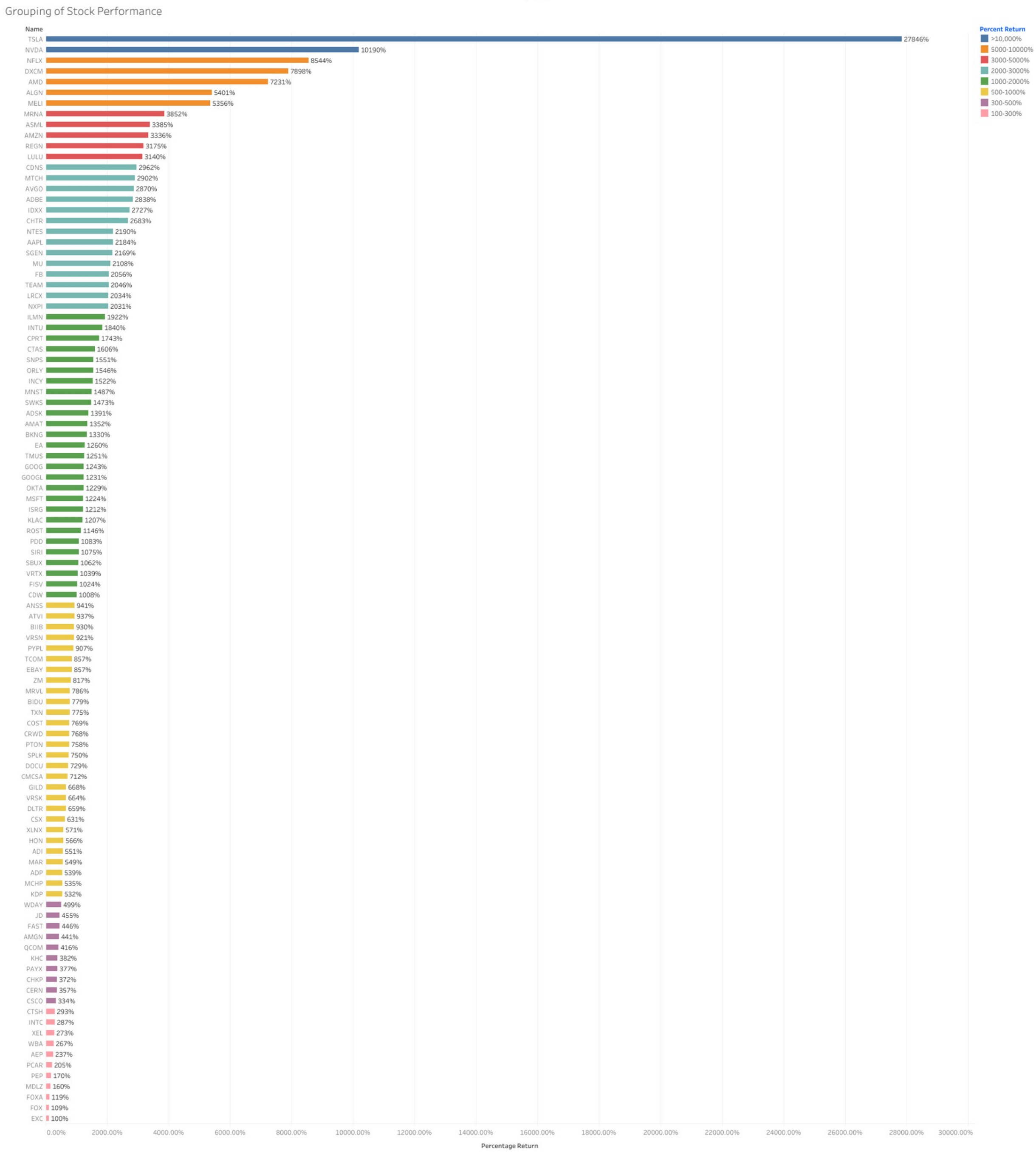
Annimated Stock Price.png



Dashboard 1.png



Group.png

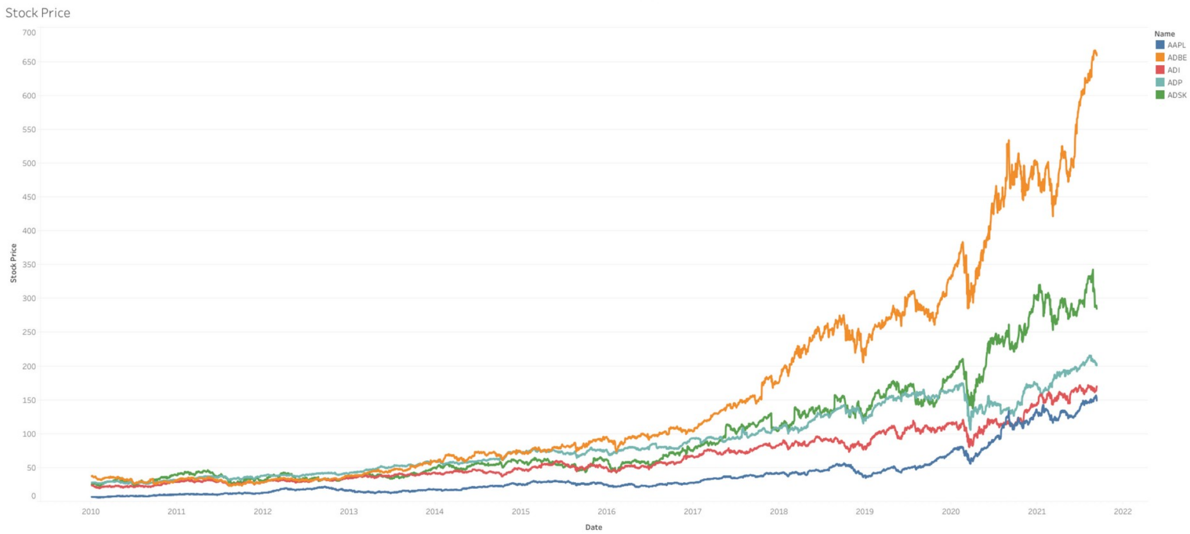


Grouping of Stock Performance
Name
TSLA 27846%
NVDA 10190%
NFLX 8544%
DXCM 7898%
AMD 7231%
ALGN 5401%
MELI 5356%
MRNA 3852%
ASML 3385%
AMZN 3336%
REGN 3175%
LULU 3140%
CDNS 2962%
MTCH 2902%
AVGO 2870%
ADBE 2838%
IDXX 2727%
CHTR 2683%
NTES 2190%
AAPL 2184%
SGEN 2169%
MU 2108%
FB 2056%
TEAM 2046%
LRCX 2034%
NXPI 2031%
ILMN 1922%
INTU 1840%
CPRT 1743%
CTAS 1606%
SNPS 1551%
ORLY 1546%
INCY 1522%
MNST 1487%
SWKS 1473%
ADSK 1391%
AMAT 1352%
BKNG 1330%
EA 1260%
TMUS 1251%
GOOG 1243%
GOOGL 1231%
OKTA 1229%
MSFT 1224%
ISRG 1212%
KLAC 1207%
ROST 1146%
PDD 1083%
SIRI 1075%
SBUX 1062%
VRTX 1039%
FISV 1024%
CDW 1008%
ANSS 941%
ATVI 937%
BIIB 930%
VRSN 921%
PYPL 907%
TCOM 857%
EBAY 857%
ZM 817%
MRVL 786%
BIDU 779%
TXN 775%
COST 769%
CRWD 768%
PTON 758%
SPLK 750%
DOCU 729%
CMCSA 712%
GILD 668%
VRSK 664%
DLTR 659%
CSX 631%
XLNX 571%
HON 566%
ADI 551%
MAR 549%
ADP 539%
MCHP 535%
KDP 532%
WDAY 499%
JD 455%
FAST 446%
AMGN 441%
QCOM 416%
KHC 382%
PAYX 377%
CHKP 372%
CERN 357%
CSCO 334%
CTSH 293%
INTC 287%
XEL 273%
WBA 267%
AEP 237%
PCAR 205%
PEP 170%
MDLZ 160%
FOXA 119%
FOX 109%
EXC 100%
0.00% 2000.00% 4000.00% 6000.00% 8000.00% 10000.00% 12000.00% 14000.00% 16000.00% 18000.00% 20000.00% 22000.00% 24000.00% 26000.00% 28000.00% 30000.00%
Percentage Return
Percent Return
>10,000%
5000-10000%
3000-5000%
2000-3000%
1000-2000%
500-1000%
300-500%
100-300%

Nasdaq 100 Volatility.png

Price Trend.png

Stock Price.png



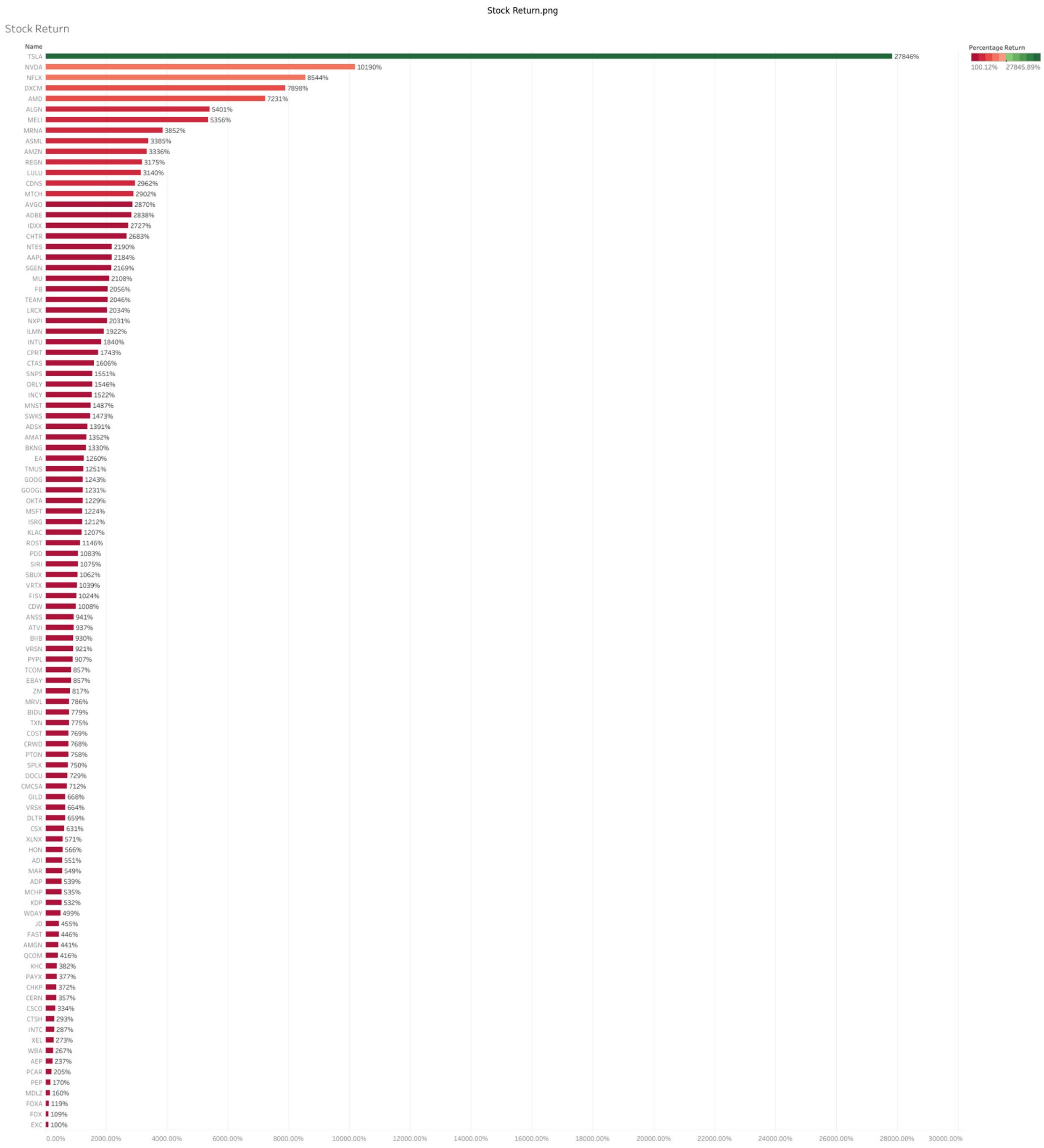


Tableau Story

Asset Turnover:

- Asset turnover measures how efficiently a company generates revenue from its assets.
- The data shows a fluctuating trend over the years, with the highest mean in 2018 (0.7464) and the lowest in 2020 (0.6538).

- Asset turnover can be influenced by industry dynamics, economic conditions, and company-specific strategies.
- The latest data for 2023 indicates a significant decline in asset turnover, which may raise concerns about asset utilization.

Buyback Yield:

- Buyback yield represents the proportion of shares repurchased by a company relative to its market capitalization.
- The data shows varying buyback yields over the years, with some years having negative yields (indicating more share issuances than buybacks).
- High standard deviations suggest substantial variability among companies in their buyback activities.
- The latest data for 2023 indicates a relatively high mean buyback yield, suggesting increased buyback activities.

Capex to Revenue:

- Capex to revenue measures the ratio of capital expenditures to total revenue.
- The data displays fluctuations in capex to revenue ratios over the years.
- Some years have a wide range of ratios, indicating differences in capital investment strategies among companies.
- In 2023, the data suggests a more consistent pattern of capital investment relative to revenue.

Cash Ratio:

- The cash ratio represents a company's ability to cover short-term liabilities with cash and cash equivalents.
- The data shows variations in cash ratios over the years, indicating differences in liquidity positions.
- A higher cash ratio is generally seen as a sign of better liquidity and the ability to meet short-term obligations.
- In 2023, the mean cash ratio is decent, but there is still variability among companies, suggesting differences in liquidity management.

Cash to Debt:

- Cash to debt ratio measures a company's ability to cover its total debt with available cash.
- The data highlights fluctuations in cash to debt ratios over the years, reflecting varying debt management strategies.
- The wide range of ratios in some years suggests that some companies may struggle to cover their debt with available cash.
- The latest data does not provide a mean, but it indicates varying degrees of ability to cover debt with cash.

cash_to_debt_2019:

- This column represents the cash-to-debt ratio for the year 2019.
- The data shows a count of 93 data points.
- The mean cash-to-debt ratio is 2.0213, suggesting that, on average, companies had more than twice as much cash as debt in 2019.
- The standard deviation is relatively high at 2.9264, indicating significant variability in cash-to-debt ratios among the companies.
- The minimum ratio is 0.01, indicating some companies had very little cash to cover their debts, while the maximum ratio is 14.67, signifying strong cash positions in other companies.

cash_to_debt_2020:

- This column represents the cash-to-debt ratio for the year 2020.
- The data includes 98 data points.
- The mean cash-to-debt ratio is significantly higher at 8.5821, suggesting a substantial increase in cash positions relative to debt in 2020.
- However, the standard deviation is extremely high at 63.3514, indicating extreme variations in cash-to-debt ratios among the companies.
- The minimum ratio is 0.01, and the maximum ratio is an exceptionally high 627.47, indicating extreme differences in financial positions.

cash_to_debt_2021:

- This column represents the cash-to-debt ratio for the year 2021.
- The data includes 100 data points.
- The mean cash-to-debt ratio is 2.3173, which is lower than in 2020 but still suggests a relatively healthy cash position compared to debt.
- The standard deviation is 5.6786, indicating some variability in cash-to-debt ratios.
- The minimum ratio is 0.01, and the maximum ratio is 40.04, indicating varying financial positions among the companies.

cash_to_debt_2022:

- This column represents the cash-to-debt ratio for the year 2022.
- The data includes 26 data points.
- The mean cash-to-debt ratio for 2022 is 3.3531, which suggests that, on average, companies had more cash than debt.
- The standard deviation is 10.0113, indicating significant variability in cash-to-debt ratios among these companies.
- The minimum ratio is 0.03, and the maximum ratio is 51.26, showing a wide range of financial positions.

cash_to_debt_latest:

- This column represents the latest available cash-to-debt ratio (presumably from 2023).
- The data includes 100 data points.

- The mean cash-to-debt ratio for the latest year is 2.3296, indicating a relatively healthy cash position compared to debt.
- However, the standard deviation is 6.8491, suggesting a significant variation in cash-to-debt ratios among these companies.
- The minimum ratio is 0.01, and the maximum ratio is 56.36, demonstrating variations in financial strength.

cogs_to_revenue_2017 to cogs_to_revenue_latest:

- These columns represent the cost of goods sold (COGS) to revenue ratios for various years.
- The data shows fluctuations in COGS to revenue ratios over the years, with means ranging from 0.3662 to 0.4745.
- These ratios provide insights into a company's cost efficiency in producing goods relative to its revenue.
- The standard deviations vary, suggesting different levels of stability or variability in cost structures among the companies.
- The latest available data in 2023 (cogs_to_revenue_latest) has a mean of 0.4467 and a relatively high standard deviation, indicating ongoing variations in cost efficiency.

mscore_2017 to mscore_latest:

- These columns represent the "mscore" values for different years.
- The "mscore" is likely a financial metric or score used for assessing companies.
- The data shows variations in "mscore" values over the years, with means ranging from -2.2129 to -2.5434.
- Negative values suggest potential financial distress or risk, while positive values may indicate financial health.
- The standard deviations vary, indicating differing levels of consistency among the companies' financial metrics.

zscore_2017 to zscore_latest:

- These columns represent the "zscore" values for different years.
- The "zscore" is often used as a measure of a company's financial stability and risk.
- The data shows fluctuations in "zscore" values over the years, with means ranging from 6.0697 to 9.1019.
- Higher "zscore" values typically indicate lower financial risk, while negative values suggest higher risk.
- The standard deviations vary, indicating differences in financial stability and risk profiles among the companies.

current_ratio_2017 to current_ratio_latest:

- These columns represent the current ratio for different years.
- The current ratio measures a company's ability to cover short-term liabilities with short-term assets.

- The data shows variations in current ratios over the years, with means ranging from 2.0902 to 2.3322.
- Ratios above 1 indicate the ability to meet short-term obligations.
- The standard deviations suggest variability in liquidity positions among companies.

Current Ratios:

- In 2021, the current ratio had a mean of 2.12, indicating that, on average, companies had current assets more than twice their current liabilities.
- In 2022, the mean current ratio was 1.98, showing a slight decrease in liquidity compared to 2021.
- The latest available data, presumably from 2023, had a mean current ratio of 1.93.
- Across all years, there is a wide range of current ratios, with some companies having substantially higher or lower liquidity than others.

Days of Inventory:

- Days of inventory measures how many days it takes for a company to sell its inventory.
- The number of days of inventory varies across years, with means ranging from 63.78 days in the latest data to 92.48 days in 2019.
- The standard deviations indicate significant variations in inventory turnover efficiency among companies.

Debt to Equity Ratios:

- The debt-to-equity ratio measures the proportion of a company's financing that comes from debt compared to equity.
- The debt-to-equity ratios fluctuate over the years, with varying means.
- In 2022, there is a notably higher mean debt-to-equity ratio at 2.40, indicating increased debt usage for financing among the companies.
- The latest available data in 2023 shows a mean debt-to-equity ratio of 0.95.

Debt to Assets Ratios:

- Debt-to-assets ratios indicate the proportion of a company's assets financed by debt.
- The ratios vary over time, with the latest data in 2023 showing a mean of 0.29.
- Debt-to-assets ratios tend to be lower than debt-to-equity ratios, indicating that companies often finance a larger portion of their assets through equity.

Debt to EBITDA Ratios:

- Debt-to-EBITDA ratios measure a company's ability to repay its debt from its earnings.
- These ratios fluctuate over the years, with varying means.
- In 2022, there is a relatively high mean debt-to-EBITDA ratio of 3.58, suggesting higher leverage.
- The latest available data in 2023 shows a mean debt-to-EBITDA ratio of 2.04.

Debt to Revenue Ratios:

- Debt-to-revenue ratios provide insights into a company's ability to service its debt using its revenue.
- The latest data in 2023 shows a mean debt-to-revenue ratio of 0.60, indicating that, on average, companies have moderate debt relative to their revenue.

Equity to Assets Ratios:

- Equity-to-assets ratios indicate the proportion of a company's assets financed by equity.
- These ratios are relatively stable across years, with means ranging from 0.37 to 0.41.
- The ratios generally suggest that companies rely on a significant portion of equity for financing their assets.

Enterprise Value to EBIT Ratios:

- The mean enterprise value to EBIT ratio has varied over the years, with the lowest in 2022 (-34.64) and the highest in 2021 (22.34).
- The standard deviation is relatively high, indicating significant variability in this ratio across companies.

Enterprise Value to EBITDA (Earnings Before Interest, Taxes, Depreciation, and Amortization):

- Similar to EV/EBIT, the mean EV/EBITDA ratio fluctuates from year to year, with the highest in 2018 (25.06) and the lowest in 2022 (11.88).
- Again, there is notable dispersion in this ratio, as seen in the high standard deviations.

Enterprise Value to Revenue:

- The mean enterprise value to revenue ratio is relatively stable across the years, with some fluctuations. It is generally below 15, indicating that most companies have reasonable valuations relative to their revenue.
- The maximum values in 2019 and 2020 stand out as potential outliers, suggesting some companies had extremely high valuations relative to their revenue.

Financial Distress and Financial Strength:

- The financial distress ratio varies widely, with the maximum value being 50, indicating some companies were in significant financial distress.
- On the other hand, the financial strength ratio is more stable, with most companies scoring between 3 and 10, where higher values suggest better financial strength.

Earning Yield (Greenblatt):

- Earning yield, based on the Greenblatt formula, fluctuates over the years but generally remains positive. It indicates the potential returns for investors.
- The mean earning yield ranges from 2.43 (2022) to 4.18 (latest), suggesting varying levels of potential return.

Free Float Percentage:

- The free float percentage is relatively stable, with a mean around 82%, indicating that, on average, a significant portion of shares is publirstand potential challenges a company may face.

Investment Strategy:

- Developing an investment strategy should consider the insights derived from the analysis of these financial ratios and indicators.

Asset Turnover:

Asset turnover is a measure of a company's efficiency in using its assets to generate sales revenue. It's expected to have a positive correlation with metrics related to profitability and efficiency.

- Asset turnover shows a strong positive correlation with itself across different years, which is expected.
- There's a moderate to strong positive correlation between asset turnover and "buyback yield" for most years, suggesting that companies with higher asset turnover tend to have higher buyback yields.
- The correlation with "capex to revenue" is mostly negative, indicating that companies with higher asset turnover tend to invest less in capital expenditures relative to their revenue.
- Asset turnover is negatively correlated with "cash ratio" and "cash to debt" for most years, suggesting that companies with higher asset turnover tend to hold less cash and have lower cash-to-debt ratios.

Buyback Yield:

Buyback yield is a measure of how much a company spends on stock buybacks relative to its market capitalization.

- Buyback yield has a positive correlation with itself across different years.
- It shows a moderate positive correlation with "asset turnover" for most years, suggesting that companies with higher asset turnover tend to engage in more stock buybacks.
- There's a positive correlation with "capex to revenue" for most years, indicating that companies with higher buyback yields tend to invest less in capital expenditures relative to their revenue.
- Buyback yield has a negative correlation with "cash ratio" and "cash to debt" for most years, suggesting that companies with higher buyback yields tend to hold less cash and have lower cash-to-debt ratios.

Capex to Revenue:

Capex to revenue measures the proportion of revenue spent on capital expenditures.

- Capex to revenue has a positive correlation with itself across different years.
- It shows a negative correlation with "asset turnover" for most years, indicating that companies with higher asset turnover tend to invest less in capital expenditures relative to their revenue.
- There's a positive correlation with "buyback yield" for most years, suggesting that companies with lower capital expenditures relative to revenue tend to have higher buyback yields.
- Capex to revenue has a negative correlation with "cash ratio" and "cash to debt" for most years, indicating that companies with lower capital expenditures tend to hold less cash and have lower cash-to-debt ratios.

Cash Ratio:

The cash ratio measures a company's ability to cover its short-term liabilities with its cash and cash equivalents.

- The cash ratio has a positive correlation with itself across different years.
- It shows a negative correlation with "asset turnover" and "buyback yield" for most years, suggesting that companies with higher asset turnover and buyback yields tend to hold less cash.
- The correlation with "capex to revenue" is mostly negative, indicating that companies with lower capital expenditures relative to revenue tend to hold more cash.
- Cash ratio has a positive correlation with "cash to debt" for most years, suggesting that companies with higher cash ratios tend to have higher cash-to-debt ratios.

Cash to Debt:

Cash to debt measures a company's liquidity and its ability to meet its debt obligations.

- Cash to debt has a positive correlation with itself across different years.
- It shows a negative correlation with "asset turnover" and "buyback yield" for most years, indicating that companies with higher asset turnover and buyback yields tend to have lower cash-to-debt ratios.
- There's a positive correlation with "capex to revenue" for most years, suggesting that companies with higher capital expenditures relative to revenue tend to have higher cash-to-debt ratios.
- Cash to debt has a positive correlation with "cash ratio" for most years, which is expected.

Cogs to Revenue:

Cogs to revenue measures the cost of goods sold relative to revenue.

- It has a positive correlation with itself across different years.
- There's a negative correlation with "asset turnover," indicating that companies with higher asset turnover tend to have lower cost of goods sold relative to revenue.

MScore:

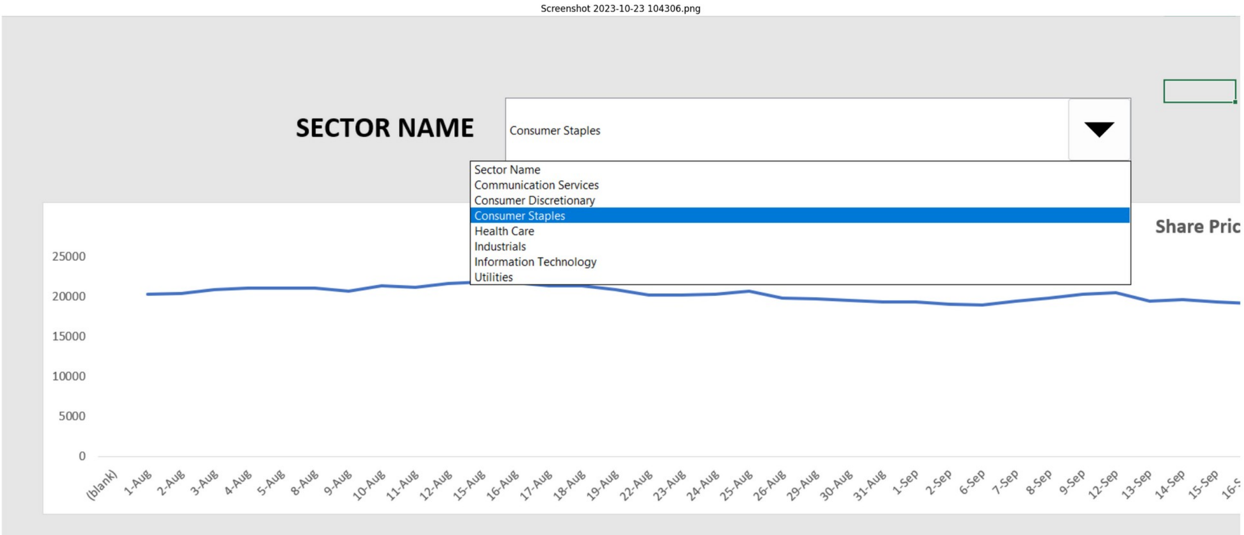
MScore is a financial distress prediction model.

- It shows a negative correlation with "asset turnover" for most years, indicating that companies with higher asset turnover tend to have lower financial distress risk.

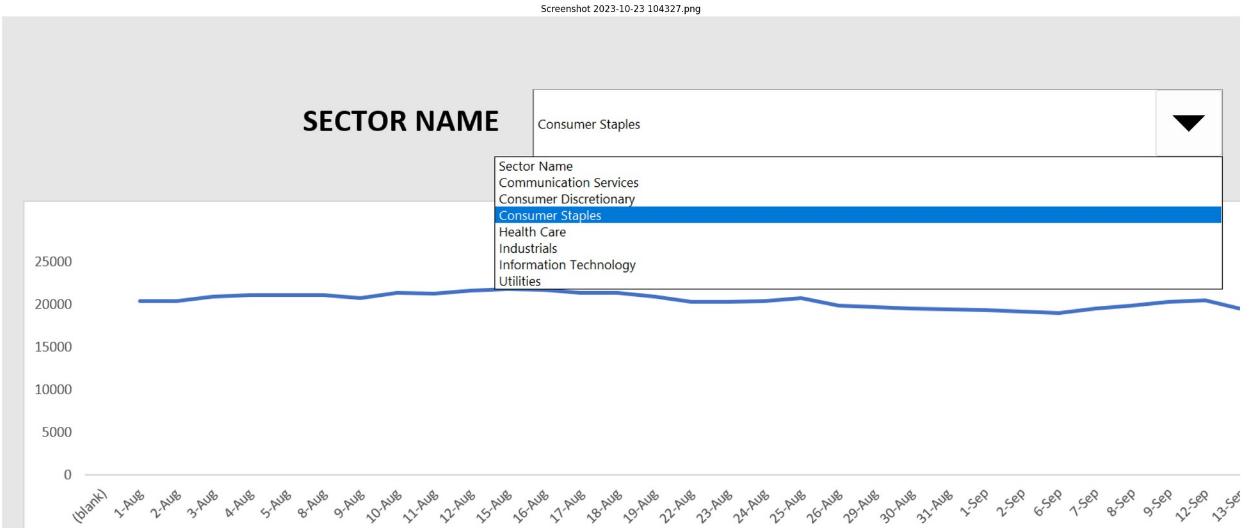
Excel Dashboard Screens

```
import os
from PIL import Image
import matplotlib.pyplot as plt

# Specify the folder path where your images are located
folder_path = r'C:\Users\bhara\OneDrive\Pictures\Screenshots\Excel
Dashboard'

# Get a list of all files in the folder
image_files = [f for f in os.listdir(folder_path) if
os.path.isfile(os.path.join(folder_path, f))]

# Filter files to include only image files (e.g., JPG, PNG, etc.)
image_files = [f for f in image_files if f.lower().endswith(('.jpg',
'.jpeg', '.png', '.gif', '.bmp'))]

# Display each image
for image_file in image_files:
    plt.figure(figsize=(30,30))
    image_path = os.path.join(folder_path, image_file)
    img = Image.open(image_path)
    print(" ")
    print(" ")
    print(" ")
    print(" ")
    plt.imshow(img)
    plt.title(image_file)
    plt.axis('off')  # Turn off axis labels
    plt.show()
```

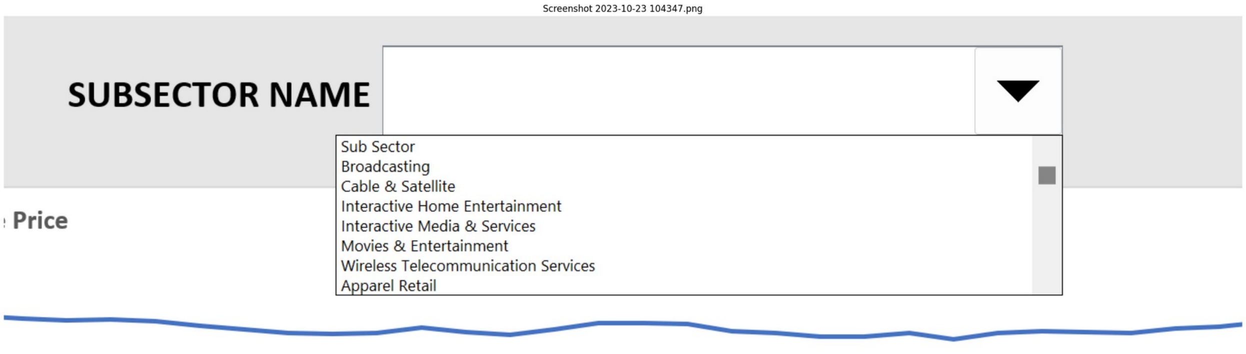
Screenshot 2023-10-23 104306.png



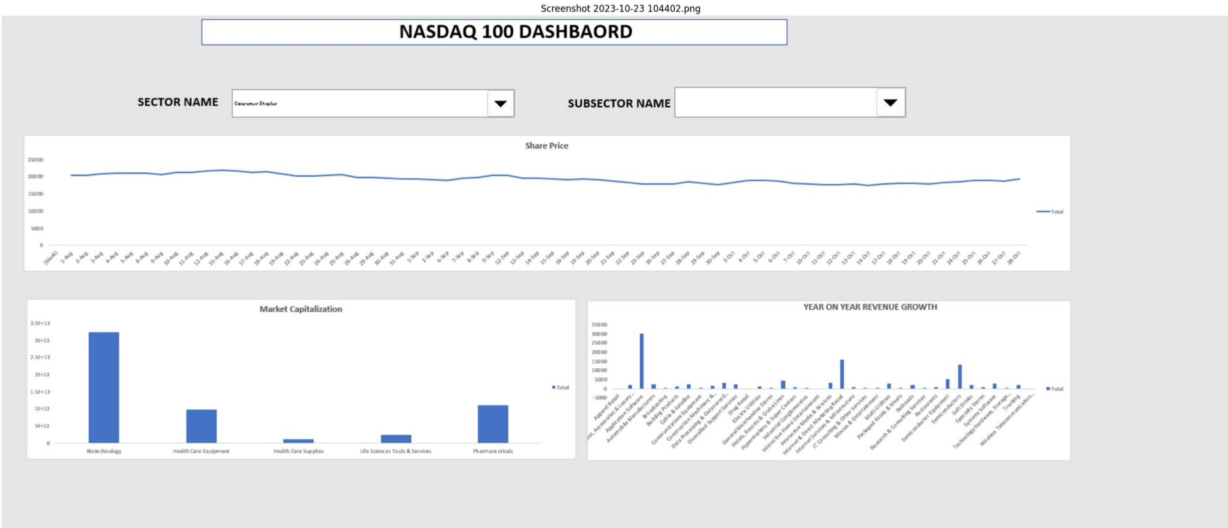
Screenshot 2023-10-23 104327.png



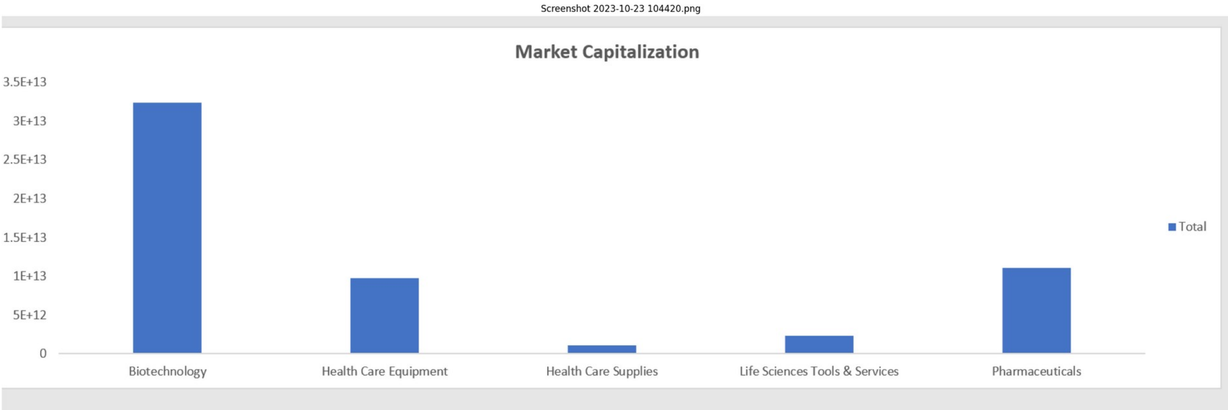
Screenshot 2023-10-23 104347.png



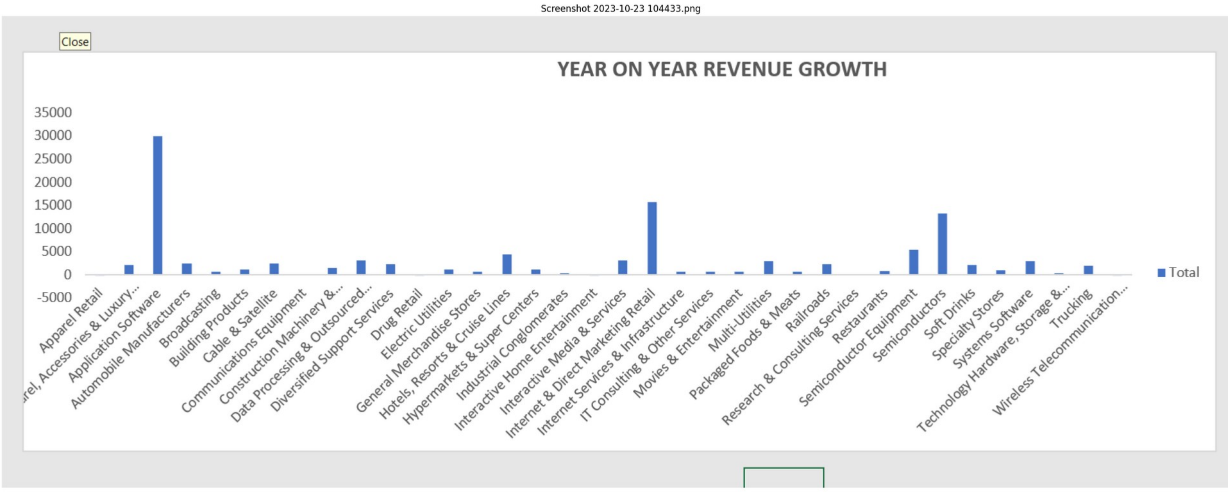
Screenshot 2023-10-23 104402.png



Screenshot 2023-10-23 104420.png



Screenshot 2023-10-23 104433.png



Insights and Findings from Excel Dashboard

Communication Services:

- Total Market Capitalization: Approximately $2.42 trillion ($242,075 billion).
- Notable Subcategories:
  - Interactive Media & Services: Approximately $199.11 billion.
  - Cable & Satellite: Approximately $13.43 trillion.
  - Movies & Entertainment: Approximately $8.35 trillion.

Consumer Discretionary:

- Total Market Capitalization: Approximately $1.65 trillion ($165,339 billion).
- Key Components:

- Internet & Direct Marketing Retail: Approximately $99.50 trillion.
- Specialty Stores: Approximately $49.53 trillion.
- Apparel, Accessories & Luxury Goods: Approximately $20.06 trillion.

Consumer Staples:

- Total Market Capitalization: Approximately $46.23 trillion ($46,234 billion).
- Notable Subcategories:
  - Hypermarkets & Super Centers: Approximately $31.68 trillion.
  - Soft Drinks: Approximately $22.23 trillion.

Health Care:

- Total Market Capitalization: Approximately $56.57 trillion ($56,569 billion).
- Key Segments:
  - Biotechnology: Approximately $32.31 trillion.
  - Pharmaceuticals: Approximately $11.11 trillion.

Industrials:

- Total Market Capitalization: Approximately $23.79 trillion ($23,786 billion).
- Subcategories:
  - Diversified Support Services: Approximately $33.35 trillion.
  - Semiconductors: Approximately $94.00 trillion.

Information Technology:

- Total Market Capitalization: Approximately $474.54 trillion ($474,539 billion).
- Notable Subcategories:
  - Application Software: Approximately $183.66 trillion.
  - Semiconductor Equipment: Approximately $81.27 trillion.

Utilities:

- Total Market Capitalization: Approximately $9.28 trillion ($9,277 billion).
- Components:
  - Electric Utilities: Approximately $2.87 trillion.
  - Multi-Utilities: Approximately $6.41 trillion.

Consumer Discretionary Sector:

- The **Specialty Stores** subcategory stands out with an exceptionally high average stock price of $773.91. This may indicate strong investor confidence in specialty retail businesses.

- **Internet & Direct Marketing Retail** also within this sector has a substantial average stock price of $440.50, reflecting the influence of e-commerce giants.

Consumer Staples Sector:

- **Hypermarkets & Super Centers** lead the way with an average stock price of $495.04, signifying significant market valuation for large-scale retail chains.

- **Soft Drinks** and **Packaged Foods & Meats** have lower average stock prices, possibly due to the competitive nature of the food and beverage industry.

Health Care Sector:

- **Biotechnology** commands a high average stock price of $274.89, underscoring the importance of innovation and research in the health care industry.

- **Pharmaceuticals** have a lower average stock price at $56.00, indicating varying market perceptions within the health care sector.

Industrials Sector:

- The **Trucking** subcategory stands out with an average stock price of $274.24, possibly reflecting the essential role of transportation in the industrial sector.

- **Railroads** have a lower average stock price, suggesting a distinct market position and challenges compared to other industrial segments.

Information Technology Sector:

- **Semiconductor Equipment** boasts a strong average stock price of $317.48, showcasing the significance of semiconductor technology in various industries.

- **Application Software** and **Systems Software** also have notable average stock prices, indicating the importance of software development in the tech sector.

Utilities Sector:

- **Electric Utilities** and **Multi-Utilities** have average stock prices of $87.22 and $62.57, respectively. These relatively stable prices are characteristic of utility companies known for consistent performance.

Communication Services Sector:

- **Movies & Entertainment** and **Interactive Media & Services** have notable average stock prices, reflecting the value of content and media-related services.

- **Cable & Satellite** stands out with a high average stock price, emphasizing the importance of telecommunications and entertainment.

Health Care Sector:

- **Health Care Equipment** and **Health Care Supplies** have similar average stock prices within the Health Care sector, indicating a balanced valuation within medical equipment and supplies.

Industrials Sector:

- **Diversified Support Services** and **Industrial Conglomerates** have substantial average stock prices within the Industrials sector, showing diverse investment opportunities.

Miscellaneous Sectors:

- **Broadcasting**, **Restaurants**, and **Research & Consulting Services** have relatively lower average stock prices, suggesting different market dynamics in their respective sectors.

Conclusion

In this Data Science Capstone project, we embarked on a thorough exploration and analysis of the Nasdaq-100, a stock market index comprising 102 equity securities from 101 of the Nasdaq's largest non-financial companies across various sectors. Our objectives encompassed leveraging data science techniques to gain insights, protect portfolios, and forecast stock prices.

Throughout the project, we navigated a series of data preparation and preprocessing steps, including the extraction of pertinent data, filtering, and collation of relevant files. We made data-driven decisions to eliminate variables with low variance and addressed missing data through imputation strategies, factoring in the company's sector.

One significant aspect of our analysis focused on the impact of the COVID-19 pandemic on stock prices. We conducted a comprehensive examination, visualizing the effects on sectors and individual companies. We examined various metrics, considered different timeframes, and offered insights into which sectors and companies experienced the greatest impact and those with the fastest and slowest recoveries.

Machine learning played a pivotal role in our project. We employed Principal Component Analysis (PCA) to reduce the dimensionality of our data, performed cluster analysis to identify distinct cohorts, and highlighted companies from diverse sectors that exhibited similar characteristics.

Time series analysis allowed us to discern seasonality, trends, and irregular components in historical stock prices, particularly those of Apple. We chose an appropriate exponential smoothing method for forecasting and conducted tests to assess the stationarity of stock prices. We also determined the parameters for ARIMA modeling, forecasting share prices, and validated the model's performance using the Mean Absolute Percentage Error (MAPE).

Furthermore, we identified the top companies in each sector based on market capitalization and created trend charts spanning the past five years. By displaying the 12-month rolling mean and standard deviation in the same chart, we assessed the stationarity of these companies' stock prices.

Finally, batch forecasting was performed for the top companies from each sector, taking into account market capitalization, using Auto ARIMA. We calculated the MAPE for a 12-month period to validate the forecasting models.

In conclusion, this Data Science Capstone project offers a comprehensive and data-driven approach to understanding the Nasdaq-100 stock market index. Our analyses, models, and insights provide valuable tools for investors to make informed decisions, protect their portfolios, and navigate the complex world of stock market investments.