Name: Balakrishna Mupparaju

Assignment: Week 6 & Project Milestone 2

In [40]:

```python
import pandas as pd
import numpy as np
from fuzzywuzzy import process
import requests
from bs4 import BeautifulSoup
import time
import warnings
# Suppress all warnings
warnings.filterwarnings("ignore")




# Load Kaggle dataset
kaggle_data = pd.read_csv("/Users/balakrishnamupparaju/Downloads/financials.
```

In [41]:

```python
kaggle_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 505 entries, 0 to 504
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Symbol          505 non-null    object 
 1   Name            505 non-null    object 
 2   Sector          505 non-null    object 
 3   Price           505 non-null    float64
 4   Price/Earnings  503 non-null    float64
 5   Dividend Yield  505 non-null    float64
 6   Earnings/Share  505 non-null    float64
 7   52 Week Low     505 non-null    float64
 8   52 Week High    505 non-null    float64
 9   Market Cap      505 non-null    float64
 10  EBITDA          505 non-null    float64
 11  Price/Sales     505 non-null    float64
 12  Price/Book      497 non-null    float64
 13  SEC Filings     505 non-null    object 
dtypes: float64(10), object(4)
memory usage: 55.4+ KB
```

In [42]:

```python
# Replace headers in Kaggle dataset for clarity and consistency
#Converted original headers into more descriptive and consistent names.
kaggle_data.rename(columns={
    'Symbol': 'Ticker',
    'Name': 'Company_Name',
    'Sector': 'Industry_Sector',
    'Price': 'Stock_Price',
    'Price/Earnings': 'PE_Ratio',
    'Dividend Yield': 'Dividend_Yield',
    'Earnings/Share': 'Earnings_Per_Share',
    '52 Week Low': '52_Week_Low',
    '52 Week High': '52_Week_High',
    'Market Cap': 'Market_Cap',
    'EBITDA': 'EBITDA_Value',
    'Price/Sales': 'Price_to_Sales_Ratio',
    'Price/Book': 'Price_to_Book_Ratio',
    'SEC Filings': 'SEC_Filings_Link'
}, inplace=True)

# Verify header replacements
print("Step: Replaced Headers")
print(kaggle_data.head())
```

```
Step: Replaced Headers
  Ticker         Company_Name         Industry_Sector  Stock_Price  PE_Ratio
\
0    MMM           3M Company             Industrials       222.89     24.31
1    AOS      A.O. Smith Corp             Industrials        60.24     27.76
2    ABT  Abbott Laboratories             Health Care        56.27     22.51
3   ABBV          AbbVie Inc.             Health Care       108.48     19.41
4    ACN        Accenture plc  Information Technology       150.51     25.47

   Dividend_Yield  Earnings_Per_Share  52_Week_Low  52_Week_High  \
0            2.33                7.92       259.77        175.49
1            1.15                1.70        68.39         48.92
2            1.91                0.26        64.60         42.28
3            2.50                3.29       125.86         60.05
4            1.71                5.44       162.60        114.82

          Market_Cap      EBITDA_Value  Price_to_Sales_Ratio  \
0 138,721,055,226.00  9,048,000,000.00                  4.39
1  10,783,419,933.00    601,000,000.00                  3.58
2 102,121,042,306.00  5,744,000,000.00                  3.74
3 181,386,347,059.00 10,310,000,000.00                  6.29
4  98,765,855,553.00  5,643,228,000.00                  2.60

   Price_to_Book_Ratio                                   SEC_Filings_Link
0                11.34  http://www.sec.gov/cgi-bin/browse-edgar?action...
1                 6.35  http://www.sec.gov/cgi-bin/browse-edgar?action...
2                 3.19  http://www.sec.gov/cgi-bin/browse-edgar?action...
3                26.14  http://www.sec.gov/cgi-bin/browse-edgar?action...
4                10.62  http://www.sec.gov/cgi-bin/browse-edgar?action...
```

In [43]:

```
# Scrape ticker data from Wikipedia
wiki_url = "https://en.wikipedia.org/wiki/List_of_S%26P_500_companies"
response = requests.get(wiki_url)
soup = BeautifulSoup(response.content, 'html.parser')

# Extract Ticker, Company, Sector, and Industry
rows = soup.find('table', {'id': 'constituents'}).find_all('tr')
tickers = []
companies = []
sectors = []
industries = []
```

```
date_added=[]
#print(rows)
for row in rows[1:]:
    cols = row.find_all('td')
    tickers.append(cols[0].text.strip())
    companies.append(cols[1].text.strip())
    sectors.append(cols[3].text.strip())
    industries.append(cols[4].text.strip())
    date_added.append(cols[5].text.strip())

wiki_data = pd.DataFrame({
    "Ticker": tickers,
    "Company": companies,
    "Sector": sectors,
    "Industry": industries,
    "Date_Added":date_added
})
print("Step 2: Extracted Wikipedia dataset.")
print(wiki_data.head())
```

```
Step 2: Extracted Wikipedia dataset.
  Ticker              Company                          Sector  \
0    MMM                   3M        Industrial Conglomerates
1    AOS          A. O. Smith               Building Products
2    ABT  Abbott Laboratories           Health Care Equipment
3   ABBV               AbbVie                   Biotechnology
4    ACN            Accenture  IT Consulting & Other Services

                 Industry  Date_Added
0    Saint Paul, Minnesota  1957-03-04
1     Milwaukee, Wisconsin  2017-07-26
2  North Chicago, Illinois  1957-03-04
3  North Chicago, Illinois  2012-12-31
4           Dublin, Ireland  2011-07-06
```

In [44]:

```
# Replace headers in Wikipedia dataset for consistency with Kaggle dataset
wiki_data.rename(columns={
    'Ticker': 'Ticker',  # Already aligned
    'Company': 'Company_Name',  # Aligns with Kaggle's 'Company_Name'
    'Sector': 'Industry_Sector',  # Matches Kaggle's 'Industry_Sector'
    'Industry': 'Headquarters'  # Assuming this column reflects headquarters
}, inplace=True)

# Verify header replacements in Wikipedia dataset
print("Step: Replaced Headers in Wikipedia Dataset")
print(wiki_data.head())
```

```
Step: Replaced Headers in Wikipedia Dataset
  Ticker         Company_Name                 Industry_Sector  \
0    MMM                   3M        Industrial Conglomerates   
1    AOS          A. O. Smith               Building Products   
2    ABT  Abbott Laboratories           Health Care Equipment   
3   ABBV               AbbVie                   Biotechnology   
4    ACN            Accenture  IT Consulting & Other Services   

              Headquarters  Date_Added  
0    Saint Paul, Minnesota  1957-03-04  
1     Milwaukee, Wisconsin  2017-07-26  
2  North Chicago, Illinois  1957-03-04  
3  North Chicago, Illinois  2012-12-31  
4          Dublin, Ireland  2011-07-06  
```

In [45]:

```
import requests
import time
import pandas as pd

# Placeholder for storing data
api_data = []

# Fetch the full list of tickers from the Wikipedia dataset
tickers = wiki_data['Ticker'].tolist()  # Use all available tickers
print(f"Total tickers to process: {len(tickers)}")

# Initialize ticker counter
ticker_count = 0

# Loop through each ticker and fetch data
for ticker in tickers:
    # Define API parameters
    params = {
        "function": "TIME_SERIES_DAILY",
        "symbol": ticker,
        "apikey": "your_alpha_vantage_api_key"  # Replace with your API key
    }

    # API call
    response = requests.get("https://www.alphavantage.co/query", params=para
    data = response.json()

    # Process time series data if available
    if "Time Series (Daily)" in data:
        time_series = data["Time Series (Daily)"]
        for date, values in time_series.items():
            api_data.append({
                "Ticker": ticker,
                "Date": date,
                "Open": float(values.get("1. open", 0)),
                "High": float(values.get("2. high", 0)),
                "Low": float(values.get("3. low", 0)),
                "Close": float(values.get("4. close", 0)),
                "Volume": int(values.get("5. volume", 0))
            })
    else:
        print(f"No data for {ticker}: {data.get('Note', 'Unknown error')}")

    # Respect API rate limits
    time.sleep(12)

    # Increment ticker counter
    ticker_count += 1

    # Break the loop after processing 24 tickers
    if ticker_count >= 24:
        print(f"Processed data for {ticker_count} tickers. Exiting loop.")
        break

# Convert API data to DataFrame
api_df = pd.DataFrame(api_data)
print(f"Total rows fetched from API: {len(api_df)}")

# Save to CSV (optional)
api_df.to_csv("partial_api_data.csv", index=False)
print("Partial API data saved.")
```

```
Total tickers to process: 503
Processed data for 24 tickers. Exiting loop.
Total rows fetched from API: 2400
Partial API data saved.
```

In [48]:

```
# Standardize Ticker and Industry_Sector casing
wiki_data['Ticker'] = wiki_data['Ticker'].str.upper()
wiki_data['Industry_Sector'] = wiki_data['Industry_Sector'].str.title()  # F
wiki_data['Headquarters'] = wiki_data['Headquarters'].str.title()  # Assumin
kaggle_data['Industry_Sector'] = kaggle_data['Industry_Sector'].str.title()

print("Step 4 Part 1: Fixed casing inconsistencies in Wiki and Kaggle datase
```

```
Step 4 Part 1: Fixed casing inconsistencies in Wiki and Kaggle datasets.
```

In [49]:

```
# Fill missing Kaggle data with median/mean values
kaggle_data['Market_Cap'] = kaggle_data['Market_Cap'].fillna(kaggle_data['Ma
kaggle_data['PE_Ratio'] = kaggle_data['PE_Ratio'].fillna(kaggle_data['PE_Rat
print("Step 4.2: Handled missing values in Kaggle data.")
```

```
Step 4.2: Handled missing values in Kaggle data.
```

In [50]:

```
# Remove duplicate rows from Wikipedia data
wiki_data.drop_duplicates(subset=['Ticker'], inplace=True)
print("Step 4.3: Removed duplicate entries in Wikipedia data.")
```

```
Step 4.3: Removed duplicate entries in Wikipedia data.
```

In [51]:

```
# Match company names between Kaggle and Wikipedia
"""Corrected the column reference from Company to Company_Name to match the
The Matched_Company column will now store the closest company name match bet

wiki_data['Matched_Company'] = wiki_data['Company_Name'].apply(
    lambda x: process.extractOne(x, kaggle_data['Company_Name'].tolist())[0]
)
```

```
print("Step 4.4: Performed Fuzzy Matching for company names.")
print(wiki_data[['Company_Name', 'Matched_Company']].head())
```

```
Step 4.4: Performed Fuzzy Matching for company names.
          Company_Name      Matched_Company
0                   3M           3M Company
1          A. O. Smith      A.O. Smith Corp
2  Abbott Laboratories  Abbott Laboratories
3               AbbVie          AbbVie Inc.
4            Accenture        Accenture plc
```

In [52]:

```
"""First Merge:

Kaggle and Wikipedia datasets are merged using the Ticker column to align fi

Second Merge:

The resulting merged dataset is further combined with the API dataset using 

Output:

The cleaned and unified dataset is saved as final_dataset.csv for further ar

# Merge Kaggle and Wikipedia datasets on Ticker
merged_data = pd.merge(kaggle_data, wiki_data, on='Ticker', how='inner')

# Merge the result with API data on Ticker
final_data = pd.merge(merged_data, api_df, on='Ticker', how='inner')

# Save the final cleaned dataset to a CSV file
final_data.to_csv("/Users/balakrishnamupparaju/Downloads/final_dataset.csv",

print("\nStep 5: Final cleaned dataset saved as 'cleaned_final_dataset.csv'.
print(final_data.head())
```

```
Step 5: Final cleaned dataset saved as 'cleaned_final_dataset.csv'.
  Ticker Company_Name_x Industry_Sector_x  Stock_Price  PE_Ratio  \
0    MMM     3M Company       Industrials       222.89     24.31   
1    MMM     3M Company       Industrials       222.89     24.31   
2    MMM     3M Company       Industrials       222.89     24.31   
3    MMM     3M Company       Industrials       222.89     24.31   
4    MMM     3M Company       Industrials       222.89     24.31   

   Dividend_Yield  Earnings_Per_Share  52_Week_Low  52_Week_High  \
0            2.33                7.92       259.77        175.49   
1            2.33                7.92       259.77        175.49   
2            2.33                7.92       259.77        175.49   
3            2.33                7.92       259.77        175.49   
4            2.33                7.92       259.77        175.49   

          Market_Cap  ...          Industry_Sector_y           Headquarters  
\
0 138,721,055,226.00  ...  Industrial Conglomerates  Saint Paul, Minnesota   
1 138,721,055,226.00  ...  Industrial Conglomerates  Saint Paul, Minnesota   
2 138,721,055,226.00  ...  Industrial Conglomerates  Saint Paul, Minnesota   
3 138,721,055,226.00  ...  Industrial Conglomerates  Saint Paul, Minnesota   
4 138,721,055,226.00  ...  Industrial Conglomerates  Saint Paul, Minnesota   

   Date_Added Matched_Company        Date   Open   High    Low  Close   Volu
me  
0  1957-03-04      3M Company  2025-04-17 130.34 132.95 130.08 130.21  49520
15  
1  1957-03-04      3M Company  2025-04-16 133.51 134.48 129.87 130.46  56358
29  
2  1957-03-04      3M Company  2025-04-15 136.01 137.47 135.14 135.26  25418
40  
3  1957-03-04      3M Company  2025-04-14 138.11 138.29 134.43 136.01  38158
06  
4  1957-03-04      3M Company  2025-04-11 133.13 136.49 131.66 135.95  33378
40  

[5 rows x 25 columns]
```

In [53]:

```python
# Ensure the necessary columns exist before deriving new columns
if 'High_Price' in final_data.columns and 'Low_Price' in final_data.columns:
    # Daily Price Range: Difference between High and Low prices
    final_data['Daily_Price_Range'] = final_data['High_Price'] - final_data[

if 'Close_Price' in final_data.columns and 'Open_Price' in final_data.column
    # Price Performance Index: Ratio of Close Price to Open Price
    final_data['Price_Performance_Index'] = final_data['Close_Price'] / fina

if 'Dividend_Yield' in merged_data.columns and 'Earnings_Per_Share' in merge
    # Dividend to Earnings Ratio: Ratio of Dividend Yield to Earnings per Sh
    merged_data['Dividend_to_Earnings_Ratio'] = merged_data['Dividend_Yield'

print("Step: Derived new columns successfully.")
print(final_data.head())
```

```
Step: Derived new columns successfully.
  Ticker Company_Name_x Industry_Sector_x  Stock_Price  PE_Ratio  \
0    MMM     3M Company       Industrials       222.89     24.31   
1    MMM     3M Company       Industrials       222.89     24.31   
2    MMM     3M Company       Industrials       222.89     24.31   
3    MMM     3M Company       Industrials       222.89     24.31   
4    MMM     3M Company       Industrials       222.89     24.31   

   Dividend_Yield  Earnings_Per_Share  52_Week_Low  52_Week_High  \
0            2.33                7.92       259.77        175.49   
1            2.33                7.92       259.77        175.49   
2            2.33                7.92       259.77        175.49   
3            2.33                7.92       259.77        175.49   
4            2.33                7.92       259.77        175.49   

          Market_Cap  ...         Industry_Sector_y           Headquarters  
\
0 138,721,055,226.00  ...  Industrial Conglomerates  Saint Paul, Minnesota   
1 138,721,055,226.00  ...  Industrial Conglomerates  Saint Paul, Minnesota   
2 138,721,055,226.00  ...  Industrial Conglomerates  Saint Paul, Minnesota   
3 138,721,055,226.00  ...  Industrial Conglomerates  Saint Paul, Minnesota   
4 138,721,055,226.00  ...  Industrial Conglomerates  Saint Paul, Minnesota   

   Date_Added Matched_Company        Date   Open   High    Low  Close   Volu
me  
0  1957-03-04      3M Company  2025-04-17 130.34 132.95 130.08 130.21  49520
15  
1  1957-03-04      3M Company  2025-04-16 133.51 134.48 129.87 130.46  56358
29  
2  1957-03-04      3M Company  2025-04-15 136.01 137.47 135.14 135.26  25418
40  
3  1957-03-04      3M Company  2025-04-14 138.11 138.29 134.43 136.01  38158
06  
4  1957-03-04      3M Company  2025-04-11 133.13 136.49 131.66 135.95  33378
40  

[5 rows x 25 columns]
```

In [54]:

```
#Earnings_to_MarketCap_Ratio: Assess the company’s earnings relative to its
final_data['Earnings_to_MarketCap_Ratio'] = final_data['Earnings_Per_Share']
#Debt_Equity_Calculation: Use Industry_Sector and financial metrics to deriv
sector_avg_de_ratio = final_data.groupby('Industry_Sector_x')['PE_Ratio'].me
final_data['Sector_Avg_PE_Ratio'] = final_data['Industry_Sector_x'].map(sect
#Volume_to_MarketCap: Ratio of traded volume to market capitalization. This 
final_data['Volume_to_MarketCap'] = final_data['Volume'] / final_data['Marke

#Daily_Market_Movement: Aggregate daily price movement (High - Low), providi
final_data['Daily_Market_Movement'] = final_data['High'] - final_data['Low']

#Region_Sector_Combo: Combine Headquarters and Industry_Sector_y for geograp
final_data['Region_Sector_Combo'] = final_data['Headquarters'] + ' - ' + fir

#Time_Trend: Create a rolling average of daily Close prices for a 30-day per
#final_data['30_Day_Rolling_Avg'] = final_data['Close'].rolling(window=30).m
final_data['30_Day_Rolling_Avg'] = final_data['Close'].rolling(window=30).me
```

```
final_data['30_Day_Rolling_Avg'].fillna(method='ffill', inplace=True)
print(final_data[['Date', 'Close', '30_Day_Rolling_Avg']].head(40))
```

```
          Date  Close  30_Day_Rolling_Avg
0   2025-04-17 130.21                 NaN
1   2025-04-16 130.46                 NaN
2   2025-04-15 135.26                 NaN
3   2025-04-14 136.01                 NaN
4   2025-04-11 135.95                 NaN
5   2025-04-10 132.97                 NaN
6   2025-04-09 138.32                 NaN
7   2025-04-08 127.16                 NaN
8   2025-04-07 128.55                 NaN
9   2025-04-04 126.91                 NaN
10  2025-04-03 139.74                 NaN
11  2025-04-02 147.76                 NaN
12  2025-04-01 147.67                 NaN
13  2025-03-31 146.86                 NaN
14  2025-03-28 144.84                 NaN
15  2025-03-27 148.44                 NaN
16  2025-03-26 152.68                 NaN
17  2025-03-25 153.50                 NaN
18  2025-03-24 153.15                 NaN
19  2025-03-21 150.36                 NaN
20  2025-03-20 151.27                 NaN
21  2025-03-19 153.21                 NaN
22  2025-03-18 150.92                 NaN
23  2025-03-17 153.21                 NaN
24  2025-03-14 150.41                 NaN
25  2025-03-13 146.10                 NaN
26  2025-03-12 150.24                 NaN
27  2025-03-11 147.54                 NaN
28  2025-03-10 147.62                 NaN
29  2025-03-07 146.30              143.45
30  2025-03-06 146.94              144.01
31  2025-03-05 147.61              144.58
32  2025-03-04 145.86              144.94
33  2025-03-03 153.42              145.52
34  2025-02-28 155.12              146.16
35  2025-02-27 150.52              146.74
36  2025-02-26 147.43              147.04
37  2025-02-25 146.54              147.69
38  2025-02-24 145.48              148.25
39  2025-02-21 144.98              148.86
```

In [55]:
```
print(f"Total columns after adding new features: {len(final_data.columns)}")
```

```
Total columns after adding new features: 31
```

In [56]:
```
final_data.head()
```

Out[56]:

	Ticker	Company_Name_x	Industry_Sector_x	Stock_Price	PE_Ratio	Dividend_Yiel
0	MMM	3M Company	Industrials	222.89	24.31	2.3
1	MMM	3M Company	Industrials	222.89	24.31	2.3
2	MMM	3M Company	Industrials	222.89	24.31	2.3
3	MMM	3M Company	Industrials	222.89	24.31	2.3
4	MMM	3M Company	Industrials	222.89	24.31	2.3

5 rows × 31 columns

In [57]:

```python
# Get the number of rows
total_rows = final_data.shape[0]
print(f"Total number of rows: {total_rows}")
```

```
Total number of rows: 2200
```

In [58]:

```python
# Convert Market_Cap to a readable format
final_data['Market_Cap'] = final_data['Market_Cap'].apply(lambda x: f"{x:,.0

print("Formatted Market_Cap values:")
print(final_data[['Ticker', 'Market_Cap']].head())


# Disable scientific notation globally
pd.options.display.float_format = '{:,.2f}'.format

print("Disabled scientific notation for all float columns.")
print(final_data.head())


# Save the dataset with formatted Market_Cap
final_data.to_csv("final_formatted_dataset.csv", index=False)
print("Final dataset with formatted Market_Cap saved as 'final_formatted_dat
Formatted Market_Cap values:
  Ticker       Market_Cap
0    MMM  138,721,055,226
1    MMM  138,721,055,226
2    MMM  138,721,055,226
3    MMM  138,721,055,226
4    MMM  138,721,055,226
Disabled scientific notation for all float columns.
  Ticker Company_Name_x Industry_Sector_x  Stock_Price  PE_Ratio  \
0    MMM     3M Company       Industrials       222.89     24.31
1    MMM     3M Company       Industrials       222.89     24.31
2    MMM     3M Company       Industrials       222.89     24.31
3    MMM     3M Company       Industrials       222.89     24.31
4    MMM     3M Company       Industrials       222.89     24.31

   Dividend_Yield  Earnings_Per_Share  52_Week_Low  52_Week_High  \
0            2.33                7.92       259.77        175.49
1            2.33                7.92       259.77        175.49
2            2.33                7.92       259.77        175.49
3            2.33                7.92       259.77        175.49
4            2.33                7.92       259.77        175.49

        Market_Cap  ...   High    Low  Close   Volume  \
0  138,721,055,226  ... 132.95 130.08 130.21  4952015
1  138,721,055,226  ... 134.48 129.87 130.46  5635829
2  138,721,055,226  ... 137.47 135.14 135.26  2541840
3  138,721,055,226  ... 138.29 134.43 136.01  3815806
4  138,721,055,226  ... 136.49 131.66 135.95  3337840

  Earnings_to_MarketCap_Ratio Sector_Avg_PE_Ratio Volume_to_MarketCap  \
0                        0.00               24.38                0.00
1                        0.00               24.38                0.00
2                        0.00               24.38                0.00
3                        0.00               24.38                0.00
4                        0.00               24.38                0.00

  Daily_Market_Movement                              Region_Sector_Combo  \
0                  2.87  Saint Paul, Minnesota - Industrial Conglomerates
1                  4.61  Saint Paul, Minnesota - Industrial Conglomerates
2                  2.33  Saint Paul, Minnesota - Industrial Conglomerates
3                  3.86  Saint Paul, Minnesota - Industrial Conglomerates
4                  4.82  Saint Paul, Minnesota - Industrial Conglomerates

  30_Day_Rolling_Avg
0                NaN
1                NaN
2                NaN
3                NaN
4                NaN

[5 rows x 31 columns]
Final dataset with formatted Market_Cap saved as 'final_formatted_dataset.cs
v'.
```

Ethical Implications:

Changes Made: Headers replaced, casing standardized, duplicates removed, missing values handled, and fuzzy matching conducted for consistency.

Legal Guidelines: All sources—Kaggle, Wikipedia, and Alpha Vantage API—were used in accordance with their terms of service.

Risks: Imputation of missing values could introduce biases, and fuzzy matching may result in slight inaccuracies in matching company names.

Assumptions: Assumed missing financial values could be reasonably approximated using median/mean values. Fuzzy matching accuracy relies on string similarity.

Data Credibility: All data is from credible public sources and API services validated for analysis purposes.

Mitigation: Detailed documentation of all transformations ensures transparency, minimizing risks of misrepresentation or inaccuracies.

This workflow completes the tasks of reading all three datasets, performing five transformations, and outputting a clean, formatted dataset for analysis

In [ ]: