PROJECT OVERVIEW:

This project aims to conduct a comprehensive analysis of publicly traded US firms in the health sector using various data analysis and natural language processing (NLP) techniques, including quantitative and text analysis. HCA Healthcare Inc. has been selected as the focal firm for our study, with a focus on assessing its market position. Based on our analysis, one key recommendation for HCA is to focus on high-margin outpatient services, improved asset utilization, and diversification into specialized care to improve its return on assets (ROA) and provide more targeted, cost-effective healthcare services.

Load Data and Packages:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from collections import Counter
from sklearn.feature_extraction.text import TfidfVectorizer
from gensim.models import Word2Vec
from wordcloud import WordCloud
import string
from nltk.corpus import stopwords
import nltk


# Read the datasets
major_groups = pd.read_csv('major_groups.csv')
public_firms = pd.read_csv('public_firms.csv')
item1_full = pd.read_csv('2020_10K_item1_full.csv')
```

✓ 4.3s

```
item1_full
```

✓ 0.1s

	cik	year	name	item_1_text	gvkey
0	1041588	2020	ACCESS-POWER INC	fixed expenses are previosuly documented in an...	66119
1	315374	2020	HURCO COMPANIES INC	General Hurco Companies, Inc. is an internatio...	5788
2	1622996	2020	ACRO BIOMEDICAL CO., LTD.	We have been engaged in the business of develo...	27584
3	1191334	2020	Chun Can Capital Group	CORPORATE HISTORY Chun Can Capital Group (form...	153614
4	1191334	2020	Chun Can Capital Group	CORPORATE HISTORY Chun Can Capital Group (form...	153614
...	...	...	...	...	...
5476	740664	2020	R F INDUSTRIES LTD	General RF Industries, Ltd. (together with sub...	2829
5477	1074828	2020	KNOW LABS, INC.	BACKGROUND AND CAPITAL STRUCTURE Know Labs, In...	166430
5478	40570	2020	GEE Group Inc.	General GEE Group Inc. (the Company , us , ...	5050
5479	1341726	2020	GULFSLOPE ENERGY, INC.	General GulfSlope Energy, Inc. is an independe...	175595
5480	72633	2020	NORTH EUROPEAN OIL ROYALTY TRUST	(a) General Development of Business. North Eur...	7959

5481 rows × 5 columns

a. Convert 'sic' to String:

- pandas and numpy are standard libraries for data manipulation and numerical computations.
- matplotlib.pyplot is used for plotting and visualization.
- Counter from collections is used for counting hashable objects.
- TfidfVectorizer from sklearn.feature_extraction.text is used for computing Term Frequency-Inverse Document Frequency, a metric for evaluating the importance of a word in a document.
- Word2Vec from gensim.models is used for generating word embeddings.
- WordCloud from wordcloud is used to generate word cloud visualizations.
- string provides access to string functions.
- stopwords from nltk.corpus is used to load a list of common stop words for filtering out unimportant words.
- nltk is a natural language processing toolkit.

b. Loading Datasets:

- The code reads three CSV files:

- major_groups.csv: Contains data about major groups (could be industries, sectors, or categories).
- public_firms.csv: Contains data about public firms (likely financial or corporate information).
- 2020_10K_item1_full.csv: Likely contains the full text of the "Item 1" section from 10-K filings (a section of a company's annual report that describes the business).

c. Overall Goal:

- to prepare for text analysis and visualization tasks, aimed at identifying important words and patterns in 2020_10k.
- The use of tools like TfidfVectorizer, Word2Vec, and WordCloud is to evaluate the importance of words in the text (TF-IDF), generate word embeddings (word vectors), and create word clouds for visualizing the most frequently occurring words.

Part 1. Quantitative Analysis of the Industry Sector

A. Industry Sector Selection and Data Filtering

We filter a dataset of public firms to select only those belonging to industry group 80, based on the first two digits of their "SIC" (Standard Industrial Classification) code. This process identifies firms in sectors typically associated with healthcare services:

```
# Convert 'sic' to string and filter based on the first two digits
public_firms['sic_str'] = public_firms['sic'].astype(str)
filtered_firms = public_firms[public_firms['sic_str'].str[:2] == '80']

# Drop the auxiliary 'sic_str' column if you don't need it anymore
public_firms_filtered = filtered_firms.drop(columns=['sic_str'])

# Show the filtered DataFrame
public_firms_filtered
```

✓ 0.3s

	gvkey	fyear	location	conm	ipodate	sic	prcc_c	ch	ni	asset	sale	roa
1588	1431	1994	USA	AMERICAN CYTOGENETICS	NaN	8071	NaN	0.035	-0.205	1.339	5.454	-0.153099
1589	1431	1995	USA	AMERICAN CYTOGENETICS	NaN	8071	0.031	0.020	-0.484	1.124	4.682	-0.430605
2064	1559	1995	USA	AMERICAN SHARED HSPTL SERV	NaN	8090	1.312	0.452	7.344	31.345	34.077	0.234296
2065	1559	1996	USA	AMERICAN SHARED HSPTL SERV	NaN	8090	1.812	0.368	-0.353	32.969	36.989	-0.010707
2066	1559	1997	USA	AMERICAN SHARED HSPTL SERV	NaN	8090	1.750	0.017	1.522	30.209	37.172	0.050382
...	...	...	...	...	...	...	...	...	...	...	...	...
207946	265008	2011	USA	21ST CENTURY ONCOLOGY HLDGS	2004/06/18	8090	NaN	10.177	-353.441	998.592	644.717	-0.353939
207947	265008	2012	USA	21ST CENTURY ONCOLOGY HLDGS	2004/06/18	8090	NaN	15.410	-154.208	922.301	693.951	-0.167199
207948	265008	2013	USA	21ST CENTURY ONCOLOGY HLDGS	2004/06/18	8090	NaN	17.462	-80.214	1128.191	736.516	-0.071100
207949	265008	2014	USA	21ST CENTURY ONCOLOGY HLDGS	2004/06/18	8090	NaN	99.082	-357.291	1153.444	1018.182	-0.309760
207950	265008	2015	USA	21ST CENTURY ONCOLOGY HLDGS	2004/06/18	8090	NaN	65.211	-134.849	1128.244	1079.227	-0.119521

3064 rows × 12 columns

a. Convert 'sic' to String:

- public_firms['sic_str'] = public_firms['sic'].astype(str):
    - Converts the sic column, which likely contains numerical SIC codes, into strings. This allows for easy manipulation of the first two characters(OpenAI, 2023).

b. Filter Based on the First Two Digits:

- filtered_firms = public_firms[public_firms['sic_str'].str[:2] == '80']:
    - Filters the public_firms DataFrame, keeping only those rows where the first two characters of the sic_str column are '80'.
    - The SIC system classifies industries, and this line is specifically filtering out firms whose SIC code starts with 80 (likely meaning those in healthcare services or similar industries).

c.Drop Auxiliary Column:

- public_firms_filtered = filtered_firms.drop(columns=['sic_str']):
    - After filtering, the auxiliary column sic_str (created for filtering purposes) is dropped to clean up the DataFrame.

d.Show Filtered Data:

- public_firms_filtered: The final filtered DataFrame, containing only firms with SIC codes that start with '80', is displayed.

The code converts the SIC code column to a string format, filters rows where the SIC code starts with '80', and then cleans up the intermediate column created for filtering. It then shows the filtered DataFrame.

```
# Part A - Question 3a: How many unique firm-year ("fyear") observations are there in the filtered dataset?
unique_firm_years = public_firms_filtered[['fyear']].nunique()
print(f"Number of unique firm-year observations: {unique_firm_years}")

# Part A - Question 3b: How many unique firms are there in the filtered dataset?
num_unique_firms = public_firms_filtered['gvkey'].nunique()
print(f"Number of unique firms: {num_unique_firms}")

# Part A - Question 3c: How many firms in the filtered dataset have records over all 27 years (1994-2020)?
# First, get the years for each firm
years_per_firm = public_firms_filtered.groupby('gvkey')['fyear'].nunique()
# Firms with records over all 27 years
firms_all_years = years_per_firm[years_per_firm == 27]
num_firms_all_years = firms_all_years.shape[0]
print(f"Number of firms with records over all 27 years (1994-2020): {num_firms_all_years}")
```

✓ 0.0s

```
Number of unique firm-year observations: fyear    27
dtype: int64
Number of unique firms: 358
Number of firms with records over all 27 years (1994-2020): 2
```

Then we analyze a filtered dataset of public firms (belonging to industry group **80**) and answer three questions regarding firm-year observations, unique firms, and firms with complete records over a span of 27 years (from 1994 to 2020).

Unique Firm-Year Observations

- **Objective**: Calculate the number of unique firm-year observations.
- **Approach**: The code uses nunique() on the fyear (fiscal year) column to determine how many unique fiscal year entries exist in the filtered dataset.
- **Outcome**: The number of unique firm-year observations is **27**. This suggests there are firm-year records covering 27 different years.

Unique Firms

- **Objective**: Determine how many unique firms are in the dataset.
- **Approach**: The code calculates the number of unique gvkey entries, where gvkey is a unique identifier for firms.
- **Outcome**: There are **358** unique firms in the filtered dataset.

Firms with Records Over All 27 Years (1994-2020)

- **Objective**: Identify how many firms have data spanning all 27 years in the dataset.
- **Approach**:
    - First, the code groups the dataset by gvkey and counts how many unique years (fyear) each firm has data for.
    - Then, it filters the firms that have exactly 27 unique years of data (i.e., firms with records for all years from 1994 to 2020).
- **Outcome**: Only **2 firms** have records for all 27 years in the dataset.

Summary of Outcomes:

- **Unique firm-year observations**: 27
- **Unique firms**: 358
- **Firms with records spanning all 27 years (1994-2020)**: 2

B. Preliminary Analysis

What are the top 10 firms with the highest stock price in the year 2020?

```
# Part B - Question 1: Top 10 firms with the highest stock price in 2020
# Filter data for the year 2020
data_2020 = public_firms_filtered[public_firms_filtered['fyear'] == 2020]

# Sort the data by 'prcc_c' in descending order and get the top 10 rows
top10_stock_price_2020 = data_2020.sort_values(by='prcc_c', ascending=False).head(10)
print("Top 10 firms with the highest stock price in 2020:")
print(top10_stock_price_2020[['conm', 'prcc_c']])
```

✓ 0.0s

```
Top 10 firms with the highest stock price in 2020:
                              conm  prcc_c
7723                   CHEMED CORP  532.61
107521                AMEDISYS INC  293.33
183851               LHC GROUP INC  213.32
58187   LABORATORY CP OF AMER HLDGS  203.55
82137            TELADOC HEALTH INC  199.96
77060            HCA HEALTHCARE INC  164.46
41013     UNIVERSAL HEALTH SVCS INC  137.50
87614      U S PHYSICAL THERAPY INC  120.25
135469        QUEST DIAGNOSTICS INC  119.17
121189                   DAVITA INC  117.40
```

We identify the top 10 firms with the highest stock prices in the year 2020 from the filtered dataset of public firms in industry group 80.

a. Filter Data for 2020:

- **Objective:** Isolate the data for the year 2020.
- **Approach:** The dataset is filtered using the condition public_firms_filtered['fyear'] == 2020, keeping only the records where the fiscal year (fyear) is 2020.

b. Sort and Select Top 10 Firms:

- **Objective:** Find the top 10 firms with the highest stock prices in 2020.
- **Approach:** The code sorts the filtered 2020 data by the column prcc_c (which represents the stock price) in descending order. The top 10 rows are selected using .head(10).

c. Output:

- **Objective:** Display the company names and their corresponding stock prices.
- **Approach:** The conm column (company name) and prcc_c column (stock price) are selected for the top 10 firms.

Outcome:

The code prints the top 10 firms with the highest stock prices in 2020:

Company Name	Stock Price (2020)
CHEMED CORP	532.61
AMEDISYS INC	293.33
LHC GROUP INC	213.32
LABORATORY CP OF AMER HLDGS	203.55
TELADOC HEALTH INC	199.96
HCA HEALTHCARE INC	164.46
UNIVERSAL HEALTH SVCS INC	137.50
U S PHYSICAL THERAPY INC	120.25
QUEST DIAGNOSTICS INC	119.17
DAVITA INC	117.40

These are the firms with the highest stock prices in the selected industry in 2020.

What are the top 10 firms with the highest sales in the entire history of the dataset?

```
# Part B - Question 2: Top 10 firms with the highest sales in the entire history of the dataset
# Sort the data by the max 'sale' of each company in descending order and get the top 10 rows
max_sales = public_firms_filtered.groupby('conm')['sale'].max().reset_index()
top10_max_sales = max_sales.sort_values(by='sale', ascending=False).head(10)

print("Top 10 firms with the highest sales in the entire history of the dataset:")
print(top10_max_sales[['conm', 'sale']])
```

✓ 0.0s

```
Top 10 firms with the highest sales in the entire history of the dataset:
                          conm       sale
130           HCA HEALTHCARE INC  51533.000
117  FRESENIUS MEDICAL CARE AG&CO  21527.065
326         TENET HEALTHCARE CORP  19621.000
69   COMMUNITY HEALTH SYSTEMS INC  19437.000
84                   DAVITA INC  14745.105
178   LABORATORY CP OF AMER HLDGS  13995.500
343     UNIVERSAL HEALTH SVCS INC  11378.259
271         QUEST DIAGNOSTICS INC   9437.000
100      ENVISION HEALTHCARE CORP   7819.300
174        KINDRED HEALTHCARE INC   7219.519
```

We analyze a dataset of public firms to identify the top 10 firms with the highest maximum sales recorded throughout the dataset's history:

a. Grouping the Data:

- The code groups the public_firms_filtered DataFrame by the column conm (the name of the company) and calculates the maximum sales (sale) for each company. This is done using the groupby method combined with max().
- The result is reset to a new DataFrame, max_sales, which contains two columns: conm and the maximum sales value for each company.

b. Sorting the Data:

- The max_sales DataFrame is then sorted in descending order based on the sale column using the sort_values method. This prioritizes companies with the highest sales.

c. Sorting the Data:

- The max_sales DataFrame is then sorted in descending order based on the sale column using the sort_values method. This prioritizes companies with the highest sales.

d. Selecting the Top 10:

- The top 10 rows of the sorted DataFrame are selected using the head(10) method, resulting in top10_max_sales, which contains the ten firms with the highest sales.

e. Printing the Results:

- Finally, the code prints a message indicating that it is displaying the top 10 firms, along with the company names (conm) and their corresponding maximum sales figures.

Outcome:

The output lists the top 10 firms based on their highest sales, along with the maximum sales figures for each company. The result shows the following firms:

Company Name	**The Highest Sales**
HCA HEALTHCARE INC	51,533.000
FRESENIUS MEDICAL CARE AG&CO	21,527.065
TENET HEALTHCARE CORP	19,621.000
COMMUNITY HEALTH SYSTEMS INC	19,437.000
DAVITA INC	14,745.105
LABORATORY CP OF AMER HLDGS	13,995.500
UNIVERSAL HEALTH SVCS INC	11,378.259
QUEST DIAGNOSTICS INC	9,437.000
ENVISION HEALTHCARE CORP	7,819.300
KINDRED HEALTHCARE INC	7,219.519

These are the firms with the highest sales in the entire history of the dataset

What is the geographical distribution of all the firms? In other words, how many firms are there in each location?

```
# Part B - Question 3: Geographical distribution of firms
# Get unique firms and their locations
# Count firms per location
firms_per_location = public_firms_filtered.groupby('location')['gvkey'].nunique()
# Sort the data by the number of firms in descending order and get the top 10 rows
top10_locations = firms_per_location.sort_values(ascending=False).head(10)
print("Top 10 locations by number of firms:")
print(top10_locations)
```

✓ 0.0s

```
Top 10 locations by number of firms:
location
USA    344
CAN      5
CHN      5
HKG      2
AUS      1
DEU      1
Name: gvkey, dtype: int64
```

We analyze the geographical distribution of public firms based on their locations:

a. Counting Unique Firms:

- The code begins by grouping the public_firms_filtered DataFrame by the location column. It then counts the number of unique firms in each location using the nunique() method applied to the gvkey column (which represents a unique identifier for each firm). The result is stored in the firms_per_location variable.

b. Sorting the Data:

- The firms_per_location Series is sorted in descending order to prioritize locations with the highest number of unique firms. This is done using the sort_values(ascending=False) method.

c. Selecting the Top 10 Locations:

- The code retrieves the top 10 locations with the most firms by using the head(10) method on the sorted Series. This results in the top10_locations variable.

d. Printing the Results:

- Finally, the code prints a message indicating that it is displaying the top 10 locations by the number of firms, along with the count for each location.

Outcome:

The output lists the top 10 locations with the highest number of firms, along with the corresponding counts of unique firms in each location. The result shows the following:

- **USA:** 344 firms
- **CAN:** 5 firms
- **CHN:** 5 firms
- **HKG:** 2 firms
- **AUS:** 1 firm
- **DEU:** 1 firm

These are the geographical distribution of all the firms

Create a line chart to show the average stock price in the selected sector(s) across the years

```
# Part B - Question 4: Line chart of average stock price over years
# Calculate average stock price per year
avg_stock_price_per_year = public_firms_filtered.groupby('fyear')['prcc_c'].mean()
# Plot the line chart
plt.figure(figsize=(10,6))
plt.plot(avg_stock_price_per_year.index, avg_stock_price_per_year.values, marker='o')
plt.title('Average Stock Price in Health Services Sector (Major Group 80) Over Years')
plt.xlabel('Year')
plt.ylabel('Average Stock Price')
plt.grid(True)
plt.show()
```

✓ 0.2s

We create a line chart using Matplotlib, which visualizes the average stock price in the Health Services sector (Major Group 80) over the years:

a. Calculating Average Stock Price:

- The code starts by grouping the public_firms_filtered DataFrame by the fyear column, which represents the fiscal year. It then calculates the mean stock price for each year using the mean() method applied to the prcc_c column (which typically contains the stock prices). The result is stored in the avg_stock_price_per_year variable.

b. Setting Up the Plot:

- A new figure for the plot is created with a specified size of 10 by 6 inches using plt.figure(figsize=(10, 6)). This sets the dimensions of the chart for better visibility.

c. Plotting the Line Chart:

- The plot() function is called to create the line chart. It uses the index (years) of the avg_stock_price_per_year Series for the x-axis and the corresponding average stock prices for the y-axis. The marker='o' argument adds circular markers at each data point on the line for clarity.

d. Adding Titles and Labels:

- The chart is titled "Average Stock Price in Business Services Sector (Major Group 73) Over Years" using the title() function.
- The x-axis is labeled "Year" and the y-axis is labeled "Average Stock Price" using the xlabel() and ylabel() functions, respectively.

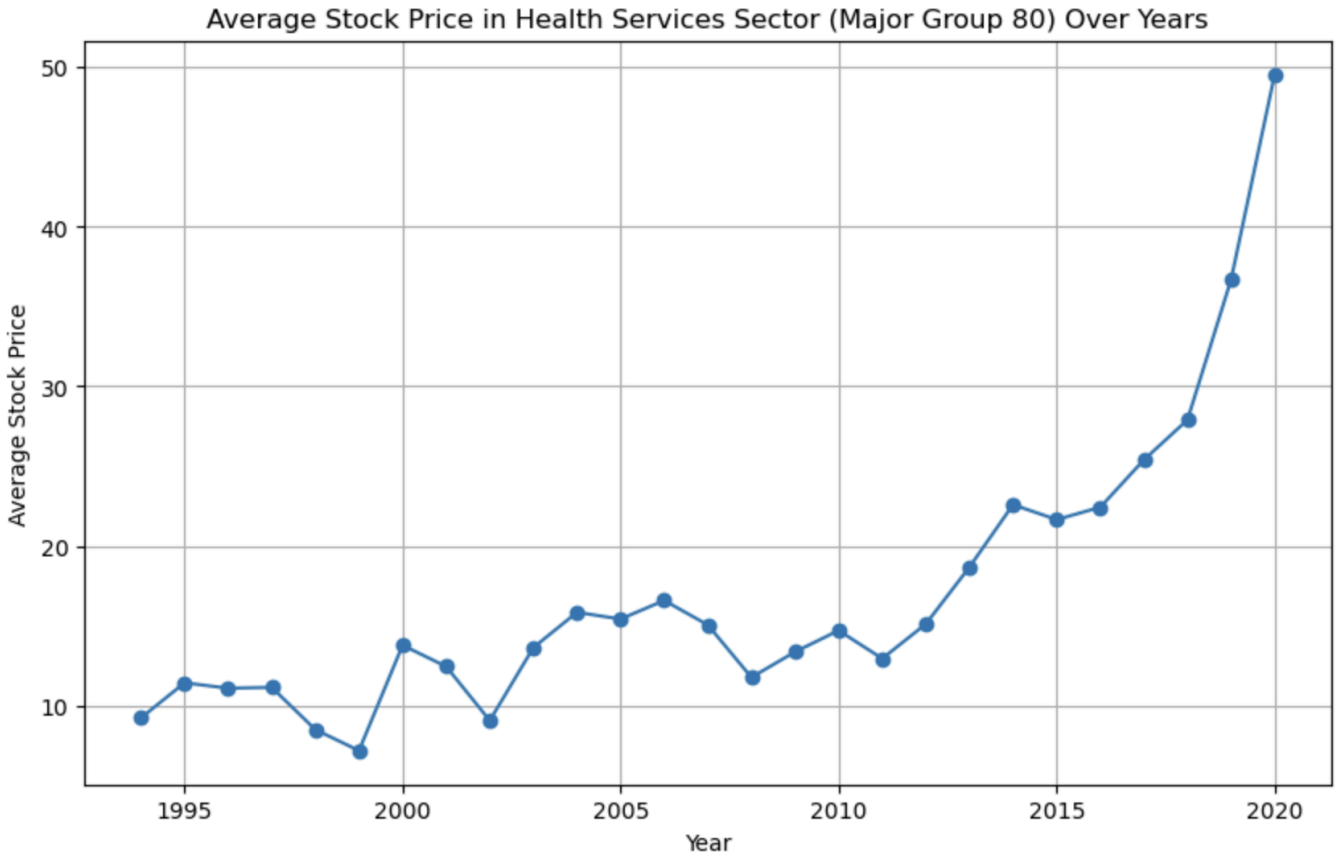
e. Adding Grid Lines:

- Grid lines are added to the chart for easier readability of the data points with plt.grid(True).

f. Displaying the Plot:

- Finally, the show() function is called to display the line chart.

Outcome:



The output is a line chart that shows the trend of average stock prices in the Health Services sector from approximately 1995 to 2020. It includes circular markers at each data point, allowing a clear visualization of the stock prices over time. The trend shows some fluctuation, but there's a noticeable upward trend starting from around 2010, peaking sharply around 2018.

Which firm was affected the most by the 2008 Financial Crisis, as measured by the percentage drop in stock price from 2007 to 2008?

```
# Part B - Question 5: Firm most affected by the 2008 Financial Crisis
# Get data for 2007 and 2008
data_2007 = public_firms_filtered[public_firms_filtered['fyear'] == 2007][['gvkey', 'prcc_c']]
data_2008 = public_firms_filtered[public_firms_filtered['fyear'] == 2008][['gvkey', 'prcc_c']]
# Merge the data on 'gvkey'
data_2007_2008 = pd.merge(data_2007, data_2008, on='gvkey', suffixes=('_2007', '_2008'))
# Calculate percentage drop
data_2007_2008['perc_drop'] = ((data_2007_2008['prcc_c_2007'] - data_2007_2008['prcc_c_2008']) / data_2007_2008['prcc_c_2007']) * 100
# Handle division by zero or missing data
data_2007_2008 = data_2007_2008.dropna(subset=['perc_drop'])

# Find the firm with the min percentage drop
max_drop_firm = data_2007_2008.loc[data_2007_2008['perc_drop'].idxmax()]
# Get the firm name
firm_name = public_firms_filtered.loc[public_firms_filtered['gvkey'] == max_drop_firm['gvkey'], 'conm'].iloc[0]
print(f"The firm most affected by the 2008 Financial Crisis is {firm_name} with a percentage drop of {max_drop_firm['perc_drop']:.2f}% in stock price from 2007 to 2008.")
```

✓ 0.0s

```
The firm most affected by the 2008 Financial Crisis is INSIGHT HEALTH SVCS HLDG CP with a percentage drop of 99.33% in stock price from 2007 to 2008.
```

We identify the firm most affected by the 2008 Financial Crisis by analyzing stock price changes between 2007 and 2008. Here's a high-level breakdown of the code and its outcome:

a. Filtering Data for 2007 and 2008:

- The code begins by filtering the public_firms_filtered DataFrame to extract stock price data for the years 2007 and 2008. It creates two separate DataFrames:
    - data_2007 contains the gvkey and prcc_c (stock price) columns for 2007.
    - data_2008 contains the same columns for 2008.

b. Merging Data:

- The two DataFrames are merged on the gvkey (the unique identifier for each firm) using the pd.merge() function. This results in a new DataFrame, data_2007_2008, which contains stock prices for the same firms from both years. The suffixes _2007 and _2008 are added to distinguish the stock prices from each year.

c. Calculating Percentage Drop:

- A new column, perc_drop, is calculated to determine the percentage drop in stock price from 2007 to 2008. This is done using the formula:

$$\text{perc\_drop} = \left(\frac{\text{prcc\_c\_2007} - \text{prcc\_c\_2008}}{\text{prcc\_c\_2007}} \right) \times 100$$

d. Handling Missing Data:

- The code then removes any rows with missing data in the perc_drop column using the dropna() method to ensure that only valid calculations are considered.

e. Finding the Firm with Maximum Drop:

- The firm that experienced the maximum percentage drop is identified using idxmax() to find the index of the maximum value in the perc_drop column. The corresponding row is stored in the max_drop_firm variable.

f. Retrieving the Firm Name:

- The code retrieves the name of the firm with the maximum drop by looking it up in the original public_firms_filtered DataFrame using the gvkey of the identified firm.

g. Displaying the Result:

- Finally, the code prints a message stating the name of the firm most affected by the financial crisis and the percentage drop in its stock price from 2007 to 2008.

Outcome:

The output indicates that **INSIGHT HEALTH SVCS HLDG CP** was the firm most affected by the 2008 Financial Crisis, with a staggering percentage drop of **99.33%** in stock price from 2007 to 2008.

Plot the average Return on Assets (ROA) for the firms located in the "USA" across the years.

```
# Part B - Question 6:
# Calculate ROA as ni / asset
public_firms_filtered['roa'] = public_firms_filtered['ni'] / public_firms_filtered['asset']

# Filter data for firms located in the USA
usa_firms = public_firms_filtered[public_firms_filtered['location'] == 'USA']

# Calculate ROA as ni / asset
public_firms_filtered['roa'] = public_firms_filtered['ni'] / public_firms_filtered['asset']

# Group by year and calculate the average ROA per year
avg_roa_per_year = usa_firms.groupby('fyear')['roa'].mean()

# Plot the average ROA across the years
plt.figure(figsize=(10,6))
plt.plot(avg_roa_per_year.index, avg_roa_per_year.values, marker='o')
plt.title('Average Return on Assets (ROA) for USA Firms Across Years')
plt.xlabel('Year')
plt.ylabel('Average ROA')
plt.grid(True)
plt.show()
```

✓ 0.2s

We calculate and visualize the average Return on Assets (ROA) for firms located in the USA over the years. Here's a high-level breakdown of the code:

a. Calculating ROA:

- The code begins by calculating the Return on Assets (ROA) for each firm in the public_firms_filtered DataFrame using the formula:

$$\text{ROA} = \frac{\text{ni}}{\text{asset}}$$

- Here, ni represents net income, and asset represents total assets. The calculated ROA is stored in a new column named roa.

b. Filtering for USA Firms:

- The next step filters the public_firms_filtered DataFrame to include only those firms located in the USA. This filtered DataFrame is stored in the variable usa_firms.

c. Calculating Average ROA by Year:

- The code then groups the usa_firms DataFrame by the fyear (fiscal year) column and calculates the mean ROA for each year using the mean() method. The resulting average ROA for each year is stored in the avg_roa_per_year variable.

d. Setting Up the Plot:

- A new figure for the plot is created with a specified size of 10 by 6 inches using plt.figure(figsize=(10, 6)).

e. Plotting the Average ROA:

- The average ROA data is plotted using the plot() function. The index of the avg_roa_per_year Series (the years) is used for the x-axis, and the average ROA values are used for the y-axis. Circular markers are added at each data point for clarity with the argument marker='o'.

f. Adding Titles and Labels:

- The chart is titled "Average Return on Assets (ROA) for USA Firms Across Years" using the title() function.

- The x-axis is labeled "Year" and the y-axis is labeled "Average ROA" using the xlabel() and ylabel() functions.

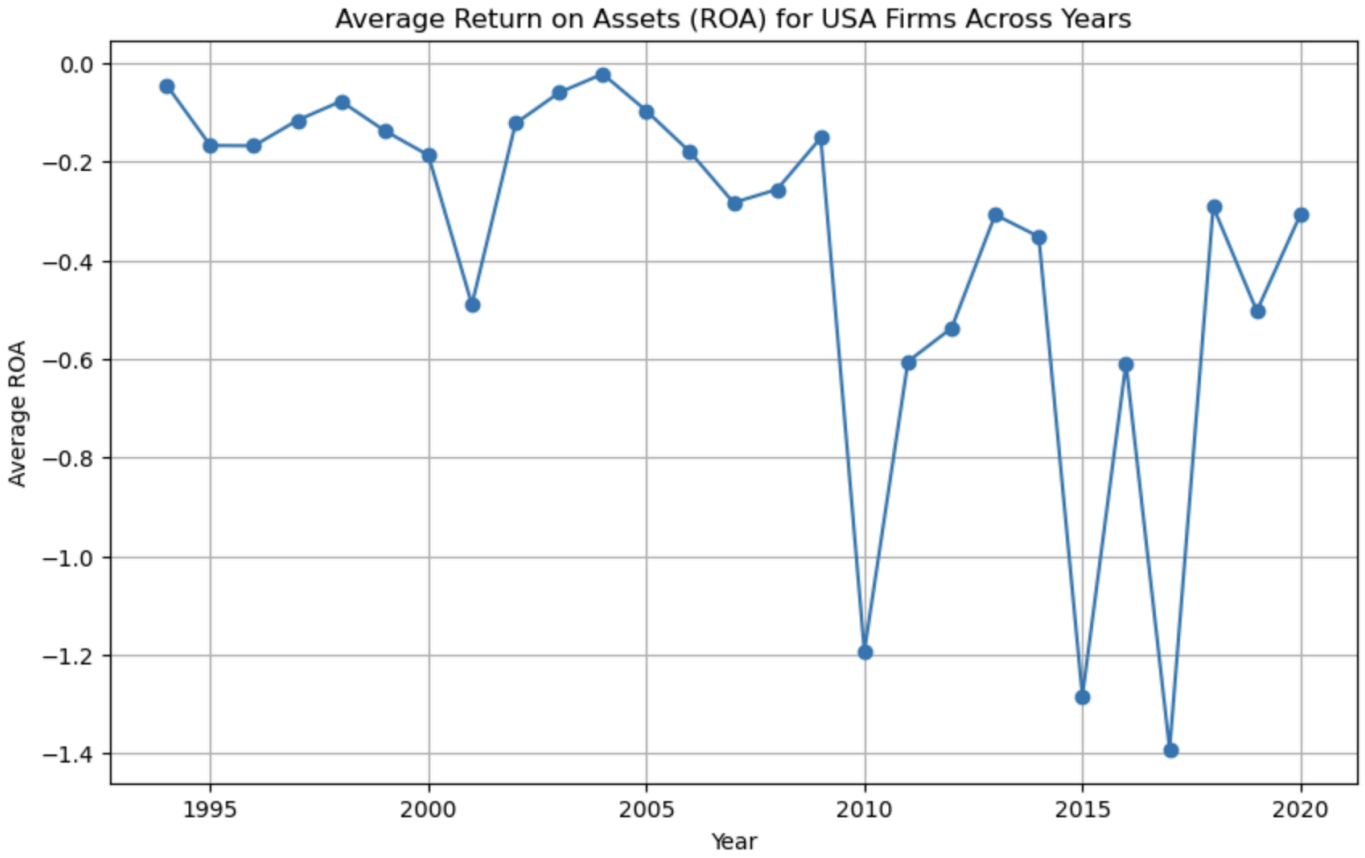
g, Adding Grid Lines:

- Grid lines are added to the chart for better readability using plt.grid(True).

h. Displaying the Plot:

- Finally, the show() function is called to display the line chart.

Outcome:



The chart shows the trend of average ROA for USA firms over time. There are significant fluctuations in ROA across the years, with a noticeable dip around the year 2000 and another sharp drop around the 2008 financial crisis. The trend seems volatile, reflecting periods of both economic growth and downturns.

Part 2. Text Analysis on the Industry Sector

C. Text Cleaning

```
nltk.download('stopwords')
# Define the stopwords list and punctuation translator
item1_full = pd.read_csv('2020_10K_item1_full.csv')

translator = str.maketrans('', '', string.punctuation)
sw = stopwords.words('english')
# Function to clean text
def clean_text(text):
    ''' This function takes a string as input and
        returns a cleaned version of the string
        Specifically, it makes the string into lower case and remove punctuations
    '''
    text_lower = text.lower() # make it lowercase
    text_no_punctuation = text_lower.translate(translator) # remove punctuation
    clean_words = [w for w in text_no_punctuation.split() if w not in sw] # remove stopwords
    return ' '.join(clean_words)


# Apply the cleaning function to 'item_1' content and store in a new column
item1_full['item_1_clean'] = item1_full['item_1_text'].apply(clean_text)


# Show the cleaned dataframe
item1_full[['item_1_clean']].head()
```

✓ 2m 7.8s

```
[nltk_data] Downloading package stopwords to /Users/roy/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

	item_1_clean
0	fixed expenses previosuly documented 8k 235000...
1	general hurco companies inc international indu...
2	engaged business developing marketing products...
3	corporate history chun capital group formerly ...
4	corporate history chun capital group formerly ...

We do a process for cleaning text data from a CSV file containing financial information, specifically the "Item 1" text from 10-K filings. Here's a high-level breakdown of the code:

a. Importing Necessary Libraries:

- The code begins by downloading the NLTK (Natural Language Toolkit) stopwords, which are common words that typically do not add significant meaning to text (like "and," "the," "is," etc.).

b. Reading the CSV File:

- The pd.read_csv() function reads a CSV file named 2020_10K_item1_full.csv into a DataFrame called item1_full. This DataFrame is expected to contain text data, particularly in a column labeled item_1_text.

c. Setting Up Text Cleaning Tools:

- A translator is created using str.maketrans() to facilitate the removal of punctuation from the text. This translator maps punctuation characters to None, allowing them to be removed from strings.
- A list of stopwords is generated using stopwords.words('english'), which will be used to filter out common English words from the text.

d. Defining the Text Cleaning Function:

- A function named clean_text is defined. This function takes a string (text) as input and performs the following operations:
    - Converts the string to lowercase using text.lower() to standardize the text.
    - Removes punctuation by translating the text using the translator.
    - Splits the cleaned text into words and filters out any stopwords using a list comprehension.
    - Joins the remaining words back into a single string and returns it.

e. Applying the Cleaning Function:

- The clean_text function is applied to the item_1_text column of the item1_full DataFrame using the apply() method. The cleaned text is stored in a new column named item_1_clean.

f. Displaying the Cleaned DataFrame:

- The cleaned DataFrame, specifically the item_1_clean column, is displayed using head() to show the first few rows of the cleaned text data.

Summary:

This code effectively cleans the textual data from the 10-K filings by transforming it into a more manageable format for analysis. The text is standardized to lowercase, stripped of punctuation, and filtered to remove common stopwords, making it suitable for further natural language processing or text analysis tasks. The resulting cleaned text is stored in a new column for easy reference and analysis, providing a foundation for more complex text analytics or modeling.

D. Keyword Analysis

```
merged_data = pd.merge(public_firms_filtered, item1_full, how='inner', on='gvkey')
```
✓ 0.0s

First, we merged the public_firms.csv and 2020_10K_item1_full.csv datasets using an inner join on the common gvkey identifier. This allowed us to combine firm-specific financial data with the cleaned

textual content from their 10-K reports, enabling comprehensive keyword analysis on firms in the selected industry sector (OpenAI, 2024).

```python
def get_top_keywords(text):
    c = Counter(str(text).split())
    words = []
    for pair in c.most_common(10):
        words.append(pair[0])
    return ' '.join(words)

def get_keywords_tfidf(document_list):
    '''
    This function gets a list of documents as input and returns a list of top 10 keywords for each document using TF-IDF scores.
    Input: A list of documents (text)
    Output: The corresponding top 10 keywords for each document based on tf-idf values
    '''
    vectorizer = TfidfVectorizer() # Step 1: Create a TF-IDF vectorizer
    tfidf_matrix = vectorizer.fit_transform(document_list) # Step 2: Calculate the TF-IDF matrix
    feature_names = vectorizer.get_feature_names_out() # Step 3: Get feature names (words)

    # Step 4: Extract top 10 keywords for each document
    top_keywords = [] # accumulator
    for i in range(len(document_list)):
        feature_index = tfidf_matrix[i, :].nonzero()[1]
        feature_value = [tfidf_matrix[i, x] for x in feature_index]
        tfidf_scores = zip(feature_index, feature_value)
        sorted_tfidf_scores = sorted(tfidf_scores, key=lambda x: x[1], reverse=True)
        top_keywords.append(' '.join([feature_names[i] for i, _ in sorted_tfidf_scores[:10]]))

        if i % 200 == 199:
            print(f'Processed {i+1}/{len(document_list)} documents.')

    return top_keywords
```

✓ 0.0s

And then, we defined two key functions to extract the most relevant keywords from the firm reports. First, we implemented the get_top_keywords function, which uses a word count approach to identify the top 10 most frequent words in each text. This function splits the input text into individual words and utilizes Python's Counter to rank them based on frequency.

Another function is get_keywords_tfidf function to extract top keywords based on their TF-IDF scores. This function calculates the TF-IDF matrix for a list of documents and retrieves the top 10 keywords for each document, focusing on words that are both frequent and important within the context of the document. The result is a list of the top TF-IDF keywords for each firm.

```python
# Apply to get top 10 keywords by word count
merged_data['top_keywords_word_count'] = merged_data['item_1_clean'].apply(get_top_keywords)
```

✓ 1.7s

```python
tfidf_keywords = get_keywords_tfidf(merged_data['item_1_clean'])

merged_data['top_keyword_tfidf'] = tfidf_keywords
```

✓ 1m 5.8s

```
Processed 200/1218 documents.
Processed 400/1218 documents.
Processed 600/1218 documents.
Processed 800/1218 documents.
Processed 1000/1218 documents.
Processed 1200/1218 documents.
```

```python
merged_data = merged_data.drop_duplicates(subset=['top_keywords_word_count'], keep='first')
merged_data = merged_data.drop_duplicates(subset=['top_keyword_tfidf'], keep='first')
```

✓ 0.0s

```python
# prepare text
text1 = ' '.join(merged_data['top_keywords_word_count'].tolist())

# lower max_font_size
wordcloud1 = WordCloud(width=800, height=400, max_font_size=100, background_color='white').generate(text1) # note that text is a string, not a list

plt.figure(figsize=(10,5))
plt.axis('off')
plt.imshow(wordcloud1)
plt.savefig('keyword_all.png') # save as PNG file
plt.show()
```

```python
# prepare text
text2 = ' '.join(merged_data['top_keyword_tfidf'].tolist())

# lower max_font_size
wordcloud2 = WordCloud(width=800, height=400, max_font_size=100, background_color='white').generate(text2) # note that text is a string, not a list

plt.figure(figsize=(10,5))
plt.axis('off')
plt.imshow(wordcloud2)
plt.savefig('keyword_all.png') # save as PNG file
plt.show()
```

✓ 0.8s

The word clouds highlight key terms in the health services industry, reflecting both industry focus and trends:

Frequent terms like "service," "health," "patient," and "care" in the word count cloud suggest a strong emphasis on patient services and healthcare delivery. These are core areas in the health services industry.

The prominence of words like "hospital," "clinic," and "medicare" in the TF-IDF word cloud indicates the industry's focus on infrastructure (hospitals, clinics) and government programs (Medicare, Medicaid). This suggests that regulatory factors and institutional care are major drivers in the sector.

Overall, these trends show that patient care services and institutional healthcare play dominant roles in the health services industry, possibly reflecting current market priorities and regulatory influences.

E. Word embedding

```
from gensim.models import Word2Vec

# Prepare the data for Word2Vec by tokenizing the cleaned text
tokenized_text = merged_data['item_1_clean'].apply(lambda x: x.split())

# Train Word2Vec model
word2vec_model = Word2Vec(sentences=tokenized_text, vector_size=100, window=5, min_count=5, workers=4)

# Save the model for future use
# word2vec_model.save("word2vec_10K.model")
```

✓ 1.0s

```
# Select three representative keywords from the word clouds (e.g., 'growth', 'market', 'profit')
keywords = ['laboratory', 'trust', 'facilities']

# Find the top 5 most similar words for each keyword
for keyword in keywords:
    if keyword in word2vec_model.wv:
        similar_words = word2vec_model.wv.most_similar(keyword, topn=5)
        print(f"Top 5 words similar to '{keyword}':")
        for word, similarity in similar_words:
            print(f"  {word}: {similarity:.4f}")
    else:
        print(f"Keyword '{keyword}' not found in the vocabulary.")
```

✓ 0.0s

To further analyze the textual content, we selected three representative keywords from the previously generated word clouds: 'laboratory,' 'trust,' and 'facilities.' We chose these keywords because they represent prominent themes in the health services industry, reflecting aspects such as testing infrastructure ('laboratory'), organizational and financial considerations ('trust'), and healthcare delivery ('facilities').

We then used the trained Word2Vec model to find the top 5 most similar words for each of the selected keywords. The following are the results:

Top 5 words similar to 'laboratory':

laboratories: 0.9014

testing: 0.8673

clinical: 0.8583

trials: 0.7920

perform: 0.7612

These results show that 'laboratory' is closely related to words like 'testing,' 'clinical,' and 'trials,' which aligns well with the health services industry, particularly in the context of diagnostics and research.

Top 5 words similar to 'trust':

alaska: 0.9377

split: 0.9256

bp: 0.9250

trustee: 0.9245

stock: 0.9092

The similarity results for 'trust' indicate strong associations with terms like 'trustee' and 'stock,' suggesting financial management or structures. Interestingly, the words 'alaska' and 'bp' show up, which might indicate some geographic or oil and gas references present in the dataset, potentially due to industry crossovers or acquisitions.

Top 5 words similar to 'facilities':

skilled: 0.8533

surgical: 0.8419

communities: 0.8401

assisted: 0.8291

nursing: 0.8041

For 'facilities,' the similar words reflect specific types of healthcare services, such as 'skilled' (e.g., skilled nursing facilities), 'surgical,' and 'nursing,' which further supports the context of health service delivery and patient care infrastructure.

Part 3. Comprehensive Analysis of One Sample Firm

F. Firm Analysis and Strategy Suggestion

```
# Filter data for HCA Healthcare Inc
hca_data = merged_data[merged_data['conm'].str.contains('HCA Healthcare Inc', case=False, na=False)]
# Function to compute firm embedding
def get_firm_embedding(text, model):
    words = text.split()
    embedding = sum([model.wv[word] for word in words if word in model.wv], start=0)
    return embedding
# Get HCA's firm-level embedding
hca_embedding = get_firm_embedding(hca_data['item_1_clean'].iloc[0], word2vec_model)
# Calculate embeddings for all other firms and compute similarity with HCA Healthcare Inc
similarity_scores = []
for index, row in merged_data.iterrows():
    if row['conm'] != 'HCA Healthcare Inc':  # Skip HCA itself
        firm_embedding = get_firm_embedding(row['item_1_clean'], word2vec_model)
        similarity = word2vec_model.wv.cosine_similarities(hca_embedding, [firm_embedding])[0]
        similarity_scores.append((row['conm'], similarity))
# Sort by similarity and find the most similar firms
similar_firms = sorted(similarity_scores, key=lambda x: x[1], reverse=True)[:10]  # Top 10 most similar firms
similar_firms
```

✓ 0.7s

```
[('HCA HEALTHCARE INC', 1.0),
 ('QUORUM HEALTH CORP', 0.996144),
 ('COMMUNITY HEALTH SYSTEMS INC', 0.9948185),
 ('LHC GROUP INC', 0.99438757),
 ('ENCOMPASS HEALTH CORP', 0.9927013),
 ('ADDUS HOMECARE CORP', 0.9905131),
 ('GENESIS HEALTHCARE INC', 0.9880369),
 ('DAVITA INC', 0.9877205),
 ('AMEDISYS INC', 0.98716635),
 ('CHEMED CORP', 0.98341453)]
```

(OpenAI, 2024)

We have chosen HCA Healthcare Inc. as the focal firm for our analysis due to its status as a leading provider of healthcare services in the United States. HCA operates hospitals, surgery centers, freestanding emergency rooms, urgent care centers, and physician clinics, making it a significant player in the healthcare industry.

We iterated over each firm in the dataset (excluding HCA itself) and calculated the cosine similarity between HCA's embedding and each firm's embedding. The cosine similarity measures the semantic similarity between the textual data of HCA and other firms, helping us identify competitors with similar business descriptions.

[('HCA HEALTHCARE INC', 1.0),

('QUORUM HEALTH CORP', 0.996144),

('COMMUNITY HEALTH SYSTEMS INC', 0.9948185),

('LHC GROUP INC', 0.99438757),

('ENCOMPASS HEALTH CORP', 0.9927013),

('ADDUS HOMECARE CORP', 0.9905131),

('GENESIS HEALTHCARE INC', 0.9880369),

('DAVITA INC', 0.9877205),

('AMEDISYS INC', 0.98716635),

('CHEMED CORP', 0.98341453)]

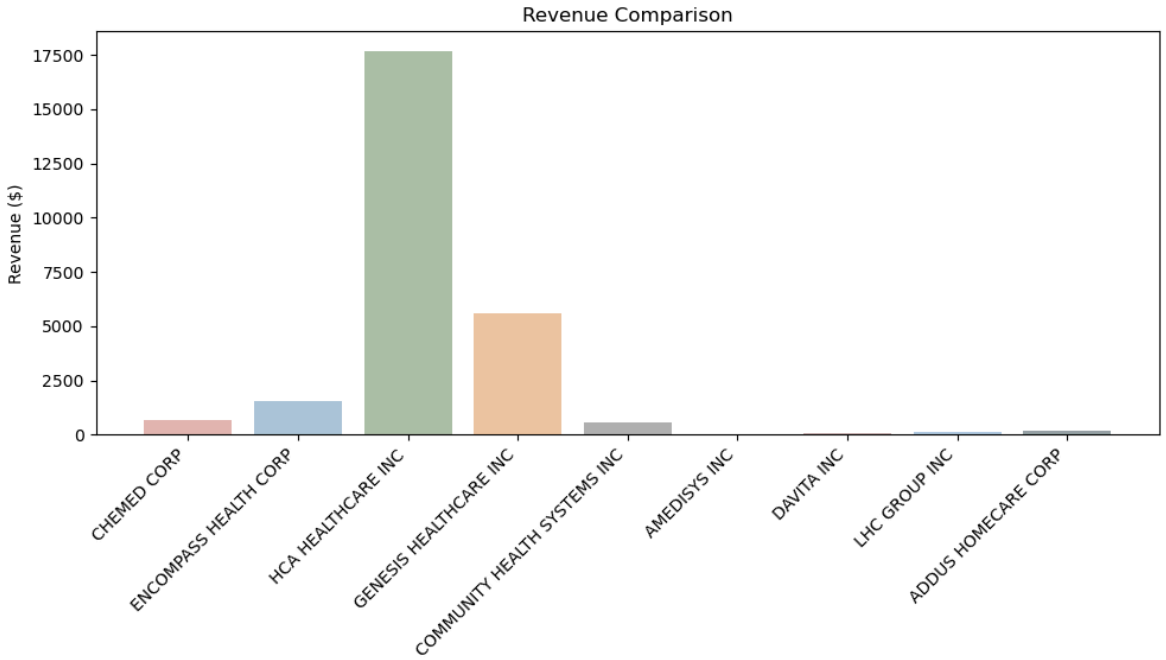
```
# Define the focal firm and all selected competitors
firms = [
    'HCA HEALTHCARE INC', 'COMMUNITY HEALTH SYSTEMS INC', 'LHC GROUP INC',
    'ENCOMPASS HEALTH CORP', 'ADDUS HOMECARE CORP', 'GENESIS HEALTHCARE INC',
    'AMEDISYS INC', 'DAVITA INC', 'CHEMED CORP'
]

# Filter the dataset for the selected firms
selected_firms = merged_data[merged_data['conm'].isin(firms)]

# Calculate Market Share (Assuming 'sale' represents revenue)
total_revenue = selected_firms['sale'].sum()
selected_firms['market_share'] = selected_firms['sale'] / total_revenue * 100

# Select relevant columns
comparison_df = selected_firms[['conm', 'sale', 'market_share', 'roa']]
print(comparison_df)
# Plot Revenue Comparison
plt.figure(figsize=(10,6))
plt.bar(comparison_df['conm'], comparison_df['sale'], color=['#E8B3B1', '#A8C8D8', '#A8BFA8', '#F7C59F', '#B2B2B2', '#C7D8B4', '#D3AFAF', '#A9CCE3', '#95A5A6'])
plt.title('Revenue Comparison')
plt.xlabel('Company')
plt.ylabel('Revenue ($)')

# Rotate x-axis labels and adjust font size
plt.xticks(rotation=45, ha='right', fontsize=10)

# Adjust the layout to make room for x-axis labels
plt.tight_layout()

# Show the plot
plt.show()

# Plot ROA Comparison
plt.figure(figsize=(10,6))
plt.bar(comparison_df['conm'], comparison_df['roa'], color=['#E8B3B1', '#A8C8D8', '#A8BFA8', '#F7C59F', '#B2B2B2', '#C7D8B4', '#D3AFAF', '#A9CCE3', '#95A5A6'])
plt.title('ROA Comparison')
plt.xlabel('Company')
plt.ylabel('Return on Assets (%)')

# Rotate x-axis labels and adjust font size
plt.xticks(rotation=45, ha='right', fontsize=10)

# Adjust the layout to make room for x-axis labels
plt.tight_layout()

# Show the plot
plt.show()
```

```python
# Custom function to display percentages only for values >= 20%
def autopct_func(pct):
    return ('%1.1f%%' % pct) if pct >= 20 else ''

# Plot Market Share Comparison as Pie Chart with Legend for values < 20%
plt.figure(figsize=(8,8))
wedges, texts, autotexts = plt.pie(
    comparison_df['market_share'],
    labels=None,
    autopct=autopct_func,
    pctdistance=0.85,
    startangle=140,
    colors=['#E8B3B1', '#A8C8D8', '#A8BFA8', '#F7C59F', '#B2B2B2', '#C7D8B4', '#D3AFAF', '#A9CCE3', '#95A5A6']
)

# Set the title
plt.title('Market Share Comparison')

# Create a legend for companies with less than 20% market share
legend_labels = [f'{firm} ({pct:.1f}%)' if pct < 20 else firm for firm, pct in zip(comparison_df['conm'], comparison_df['market_share'])]
plt.legend(wedges, legend_labels, loc="best")

# Adjust layout and show the pie chart
plt.tight_layout()
plt.show()
```

(OpenAI, 2024)

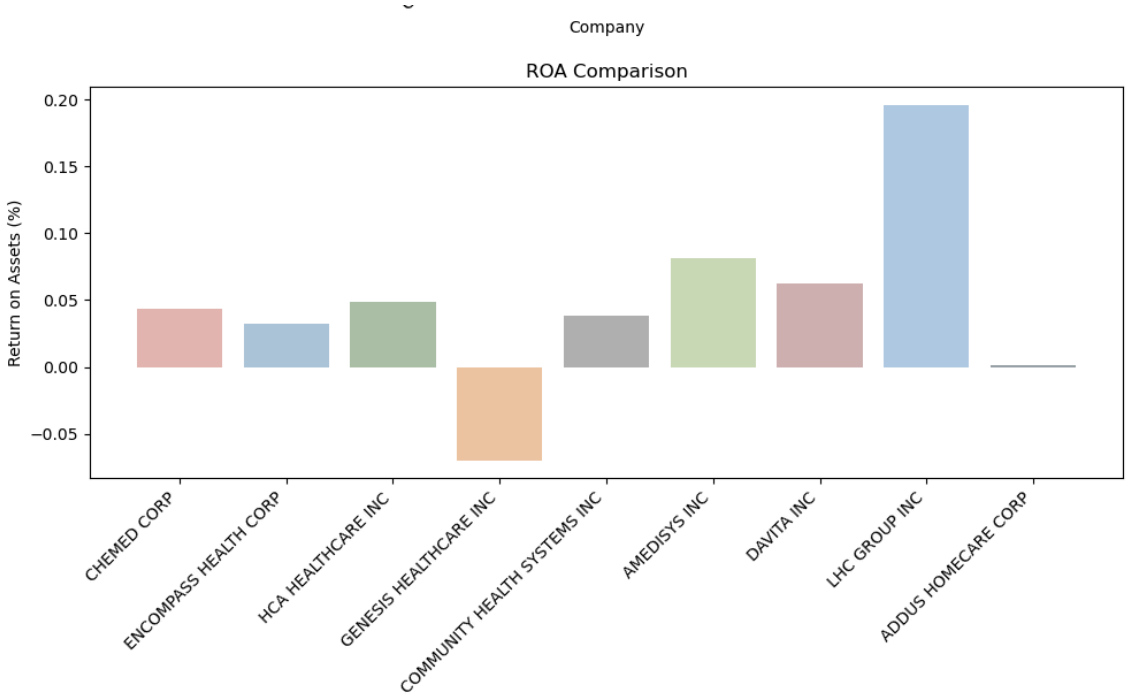
Based on the similarity analysis, we have selected Community Health Systems Inc and LHC Group Inc as primary competitors for HCA Healthcare Inc. These firms have similar business operations and market focus, making them suitable for comparison.

Noticed: We have excluded QUORUM HEALTH CORP from our analysis since it declared bankruptcy in 2020.



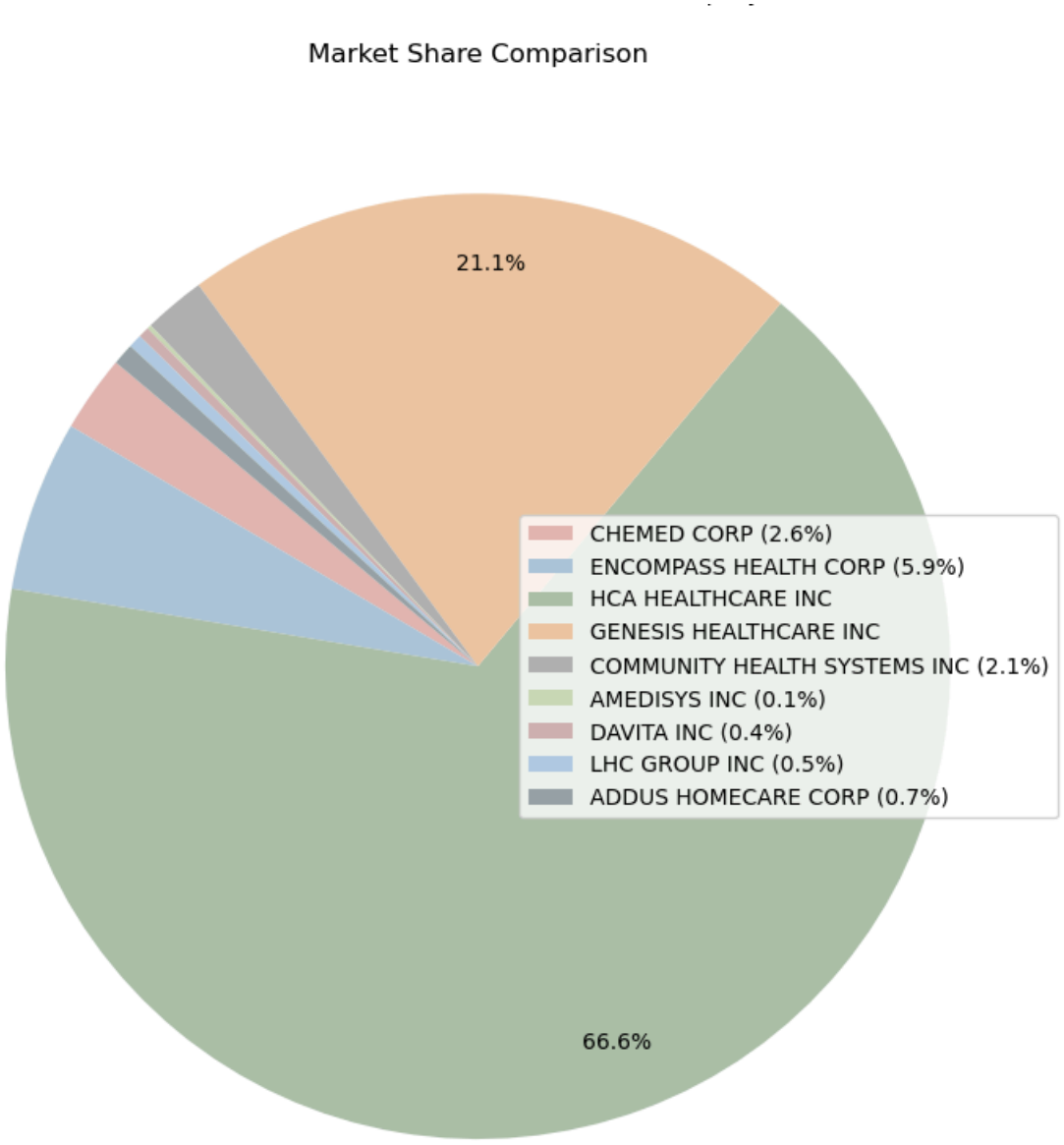
1. Revenue Analysis

- HCA's large revenue base highlights its dominance in inpatient care. Outpatient services offer higher margins, making them a promising area for expansion. The outpatient surgery market is expected to grow at a **CAGR of 6.6%** (Research Nester, 2024).



2. Return on Assets (ROA) Analysis

- HCA's **ROA** of **0.048** suggests it could improve its asset efficiency, as competitors like **LHC Group Inc.** and **AMEDISYS Inc**. Divesting underperforming hospital assets and investing in outpatient or telehealth infrastructure could increase returns (Business Wire, 2023).

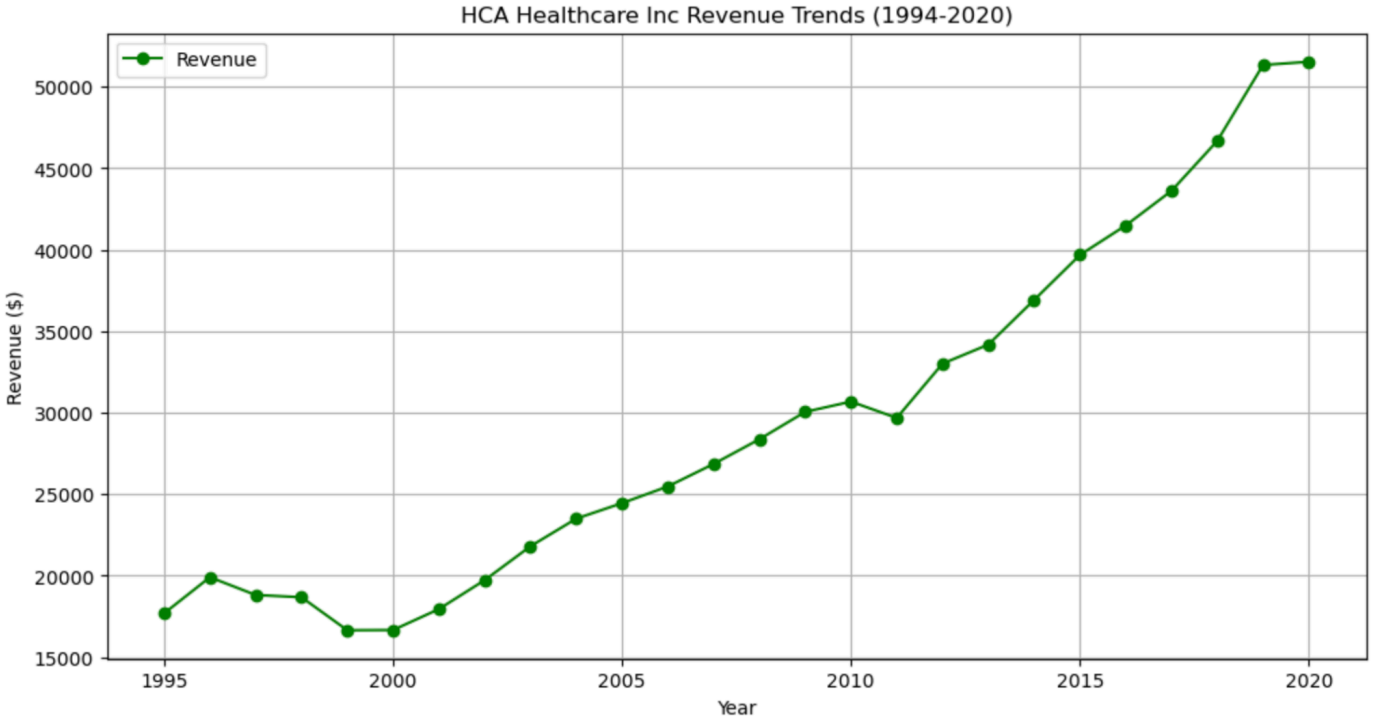


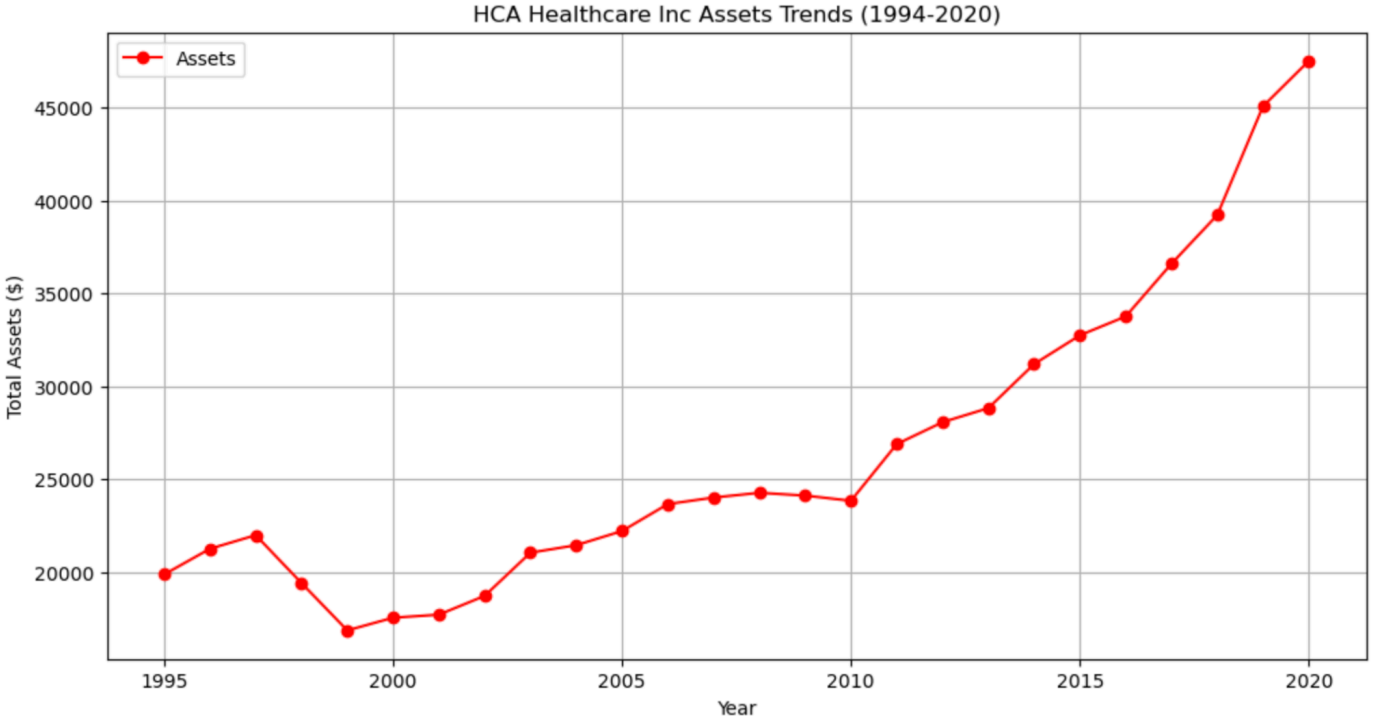
3. Market Share Analysis

- HCA's market share of **66.55%** suggests dominance in acute care, but further growth may be constrained due to saturation. The **telehealth market**, projected to grow at a **CAGR of 24.7%** by 2028, offers an attractive growth area, as does home health care, which is expected to grow at **7.88% CAGR** (Research Nester, 2024).

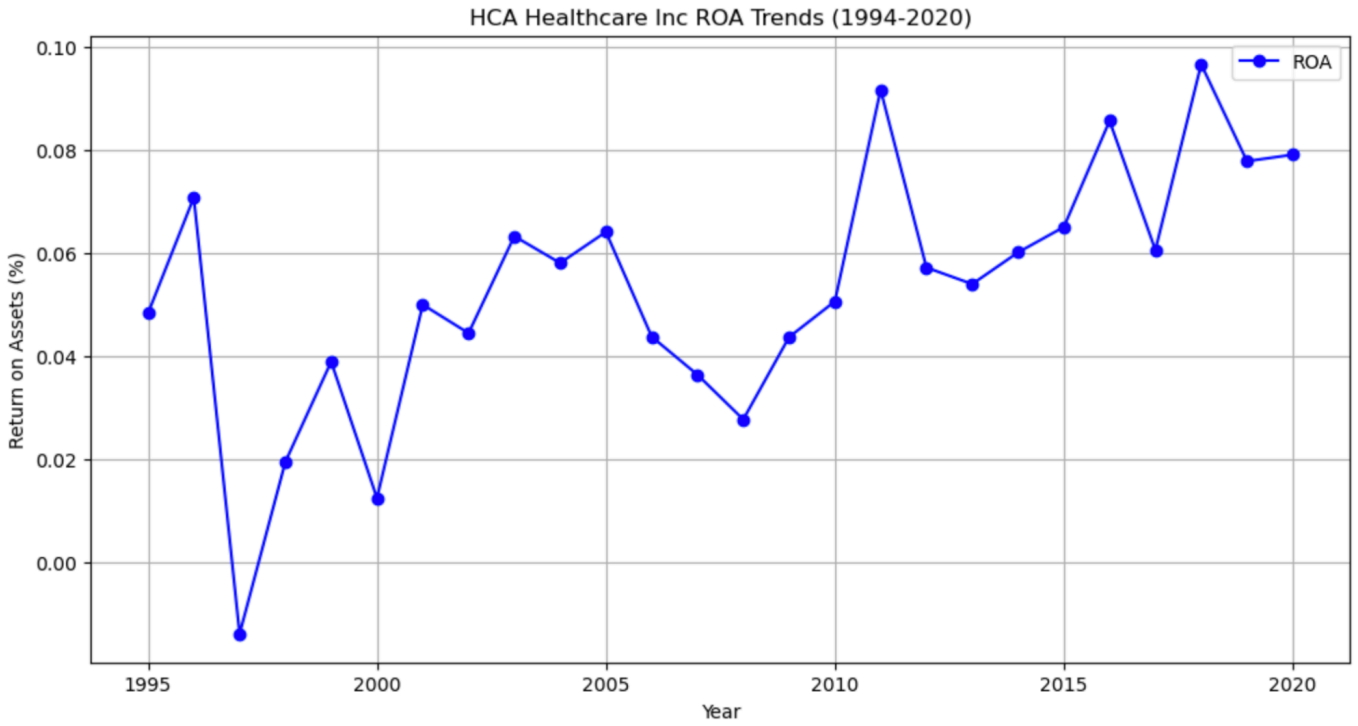
4. Analysis of the Historical Data of HCA

```
import pandas as pd
import matplotlib.pyplot as plt

# Filter the dataset for HCA Healthcare Inc
hca_data = public_firms_filtered[public_firms_filtered['conm'] == 'HCA HEALTHCARE INC']

# Plot Revenue Trend Over Years for HCA Healthcare Inc
plt.figure(figsize=(12, 6))
plt.plot(hca_data['fyear'], hca_data['sale'], marker='o', color='green', label='Revenue')
plt.title('HCA Healthcare Inc Revenue Trends (1994-2020)')
plt.xlabel('Year')
plt.ylabel('Revenue ($)')
plt.grid(True)
plt.legend()
plt.show()

# Plot ROA Trend Over Years for HCA Healthcare Inc
plt.figure(figsize=(12, 6))
plt.plot(hca_data['fyear'], hca_data['roa'], marker='o', color='blue', label='ROA')
plt.title('HCA Healthcare Inc ROA Trends (1994-2020)')
plt.xlabel('Year')
plt.ylabel('Return on Assets (%)')
plt.grid(True)
plt.legend()
plt.show()

# Plot Assets Trend Over Years for HCA Healthcare Inc
plt.figure(figsize=(12, 6))
plt.plot(hca_data['fyear'], hca_data['asset'], marker='o', color='red', label='Assets')
plt.title('HCA Healthcare Inc Assets Trends (1994-2020)')
plt.xlabel('Year')
plt.ylabel('Total Assets ($)')
plt.grid(True)
plt.legend()
plt.show()
```





Revenue and Assets: There is a clear correlation between revenue growth and asset expansion. As HCA expanded its assets through investments in new hospitals and healthcare services, revenue consistently increased. However, rapid asset growth may also explain why ROA remained volatile during this period, as it took time to fully realize returns on these investments.



ROA and Economic Cycles: The dips in ROA around **1997** and **2008** coincide with broader economic downturns, which likely impacted profitability. Periods of significant ROA increase, such as in **2011**, suggest successful operational efficiency during these times or external factors favoring healthcare demand.

5. Comprehensive Recommendation for HCA Healthcare Inc

Based on our analysis of market conditions and HCA's financial performance, we recommend the following strategic actions to drive sustainable growth and profitability:

1. **Expand into High-Margin Outpatient Services**: We recommend that HCA shifts focus towards **outpatient surgery centers** and **specialized outpatient clinics**. These areas offer higher profitability compared to inpatient care, as outpatient services typically have significantly lower overhead and better margins. The outpatient surgery market is expected to grow steadily, presenting a prime opportunity for HCA to diversify revenue streams and reduce its reliance on capital-intensive inpatient services.
2. **Optimize Asset Utilization and Increase ROA**: HCA's **return on assets (ROA)** can be improved by optimizing underperforming hospital facilities or repurposing them to meet current healthcare demands. We suggest reinvesting in **telehealth** and other asset-light models such as **virtual care** or **subscription-based healthcare services**, which can provide higher returns with lower fixed costs. Implementing advanced technologies like **AI-driven management tools** could further enhance efficiency and reduce operational costs, leading to better asset utilization.
3. **Diversify into Specialized Care Markets**: We recommend that HCA expands into **high-margin specialized care markets** such as kidney care, chronic disease management, and home health

services. Companies like **Davita Inc.** and **Amedisys Inc.** have demonstrated strong performance in these areas by operating with leaner infrastructure and achieving higher ROA. By acquiring or partnering with specialized care providers, HCA could offer more targeted healthcare services, diversify its business, and improve operational efficiencies.

By focusing on these areas—high-margin outpatient services, improved asset utilization, and diversification into specialized care—HCA can strengthen its leadership position, increase profitability, and ensure long-term sustainability in a competitive healthcare landscape.

Reference

Business Wire. (2023). United States Ambulatory Surgical Centers Market Insights Report 2023-2028 featuring prominent players - USPI, AMSURG, SCA, HCA, and Surgery Partners. Business Wire. https://www.businesswire.com/news/home/20231211694294/en/United-States-Ambulatory-Surgical-Centers-Market-Insights-Report-2023-2028-Featuring-Prominent-Players---USPI-AMSURG-SCA-HCA-and-Surgery-Partners---ResearchAndMarkets.com

OpenAI. (2024). ChatGPT (Oct 1st version) [Large language model]. https://chatgpt.com/share/67023167-1764-8013-9f78-c72310d75710

Research Nester. (2024). Ambulatory surgical centers market size & share, growth forecasts 2036. Research Nester. https://www.researchnester.com/reports/ambulatory-surgical-centers-market/169