BAIT 508 Group Project: Industry Analysis

Submitted by:

Rahul Dhomane	**19801844 [rdhomane@student.ubc.ca]**
Roshan Shetty	**13426960 [roshan77@student.ubc.ca]**
Kunqi (Quinn) Zhao	**62948492 [kqz1437@student.ubc.ca]**

Under the guidance of:

Prof. Gene Lee

SAUDER SCHOOL OF BUSINESS
UNIVERSITY OF BRITISH COLUMBIA

October 2023

PROJECT OVERVIEW:

The goal of this project is to conduct an in-depth analysis of public US firms within selected industry sector(s) using various data analyses and natural language processing (NLP) techniques that we learned in BAIT 508. Each team will choose at least one industry sector to investigate and utilize multiple datasets to extract valuable industry insights from the data.

USED DATASETS:

- public_firms.csv
- major_groups.csv
- 2020_10K_item1_full.csv

ROLES:

Rahul Dhomane - Worked on coding, idea generation, and reporting till Part 2 C.
Roshan Shetty - Worked on coding, debugging, idea generation, and reporting for Part 2 D & E.
Quinn Zhao - Worked on coding, debugging, idea generation, and reporting for Part 3.

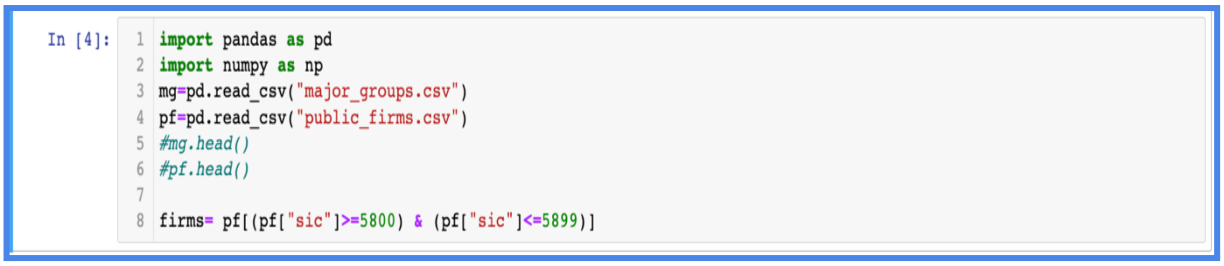
Part 1. Quantitative Analysis of the Industry Sector
A. [Industry Sector Selection and Data Filtering]

1. The file "data/major_groups.csv" contains a list of major industry sectors and their corresponding codes (column "major_group"). Your first task is to choose at least one industry sector that interests your group.

We selected the **Eating and Drinking Places** sector from the major_groups.csv with firm code equivalent to 58.

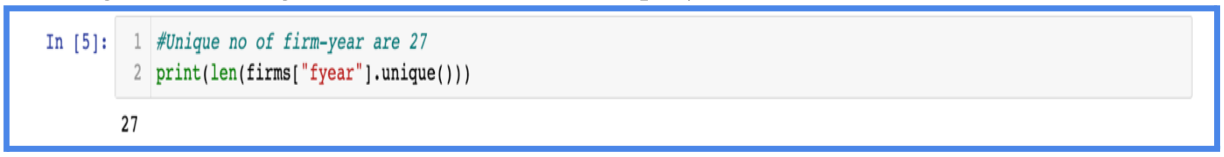
2. Next, filter the data in "data/public_firms.csv" to only include the firms belonging to the industry sector(s) you have selected. You can use the "major_group" value, which corresponds to the first two digits of each firm's SIC code,[1] to identify relevant firms.

We selected the Eating and Drinking sector for which the firm code is 58 and filtered the **sic** column from 5800 to 5899 in public_firms.csv. Then we stored the extracted records in a new dataframe called **firms.**


```
In [4]: import pandas as pd
        import numpy as np
        mg=pd.read_csv("major_groups.csv")
        pf=pd.read_csv("public_firms.csv")
        #mg.head()
        #pf.head()

        firms= pf[(pf["sic"]>=5800) & (pf["sic"]<=5899)]
```

3. Answer the following questions based on the filtered dataset:
a. How many unique firm-year ("fyear") observations are there in the filtered dataset?

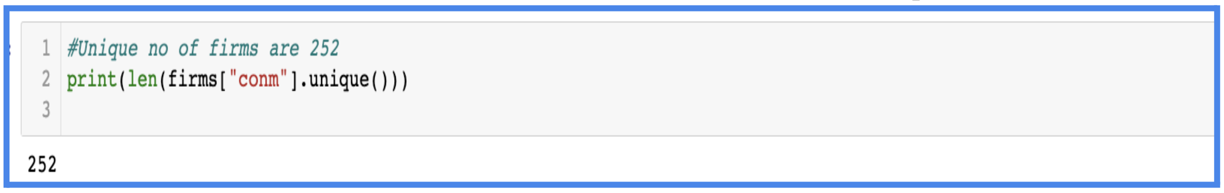
We used the unique function and then the len function on fiscal year **(fyear** column**)** in our created **firms** DataFrame to calculate the number of unique fiscal years among the companies in Eating and Drinking sector. The number of unique years is 27.



```
In [5]: #Unique no of firm-year are 27
        print(len(firms["fyear"].unique()))

27
```

b.How many unique firms are there in the filtered dataset?

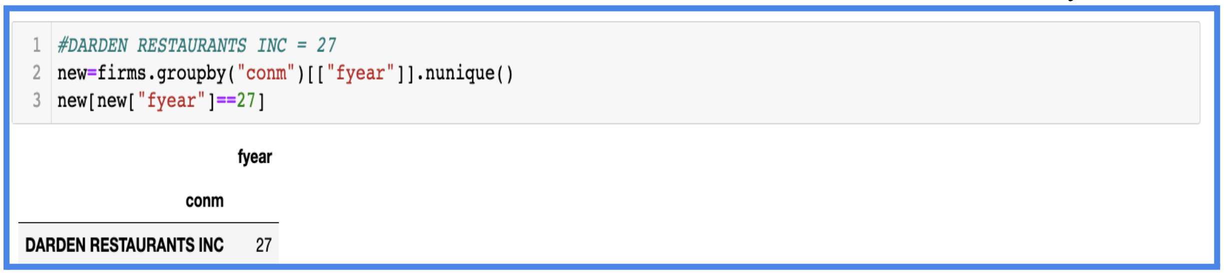
We used the unique function and then the len function on **conm**, which contains company names in the food sector from the **firms** DataFrame**.** There are 252 unique firms.



```
#Unique no of firms are 252
print(len(firms["conm"].unique()))


252
```

c. How many firms in the filtered dataset have records over all 27 years (1994-2020)?

We grouped the **firms** DataFrame by company name **conm** and calculated the number of unique fiscal years- **fyear** for each company. The result is stored in a new DataFrame called **new**. Finally, we filtered the **fyear** column in the **new** dataframe to find out the company with value **27**. The result shows that the Darden Restaurants Inc. has records over all 27 years.

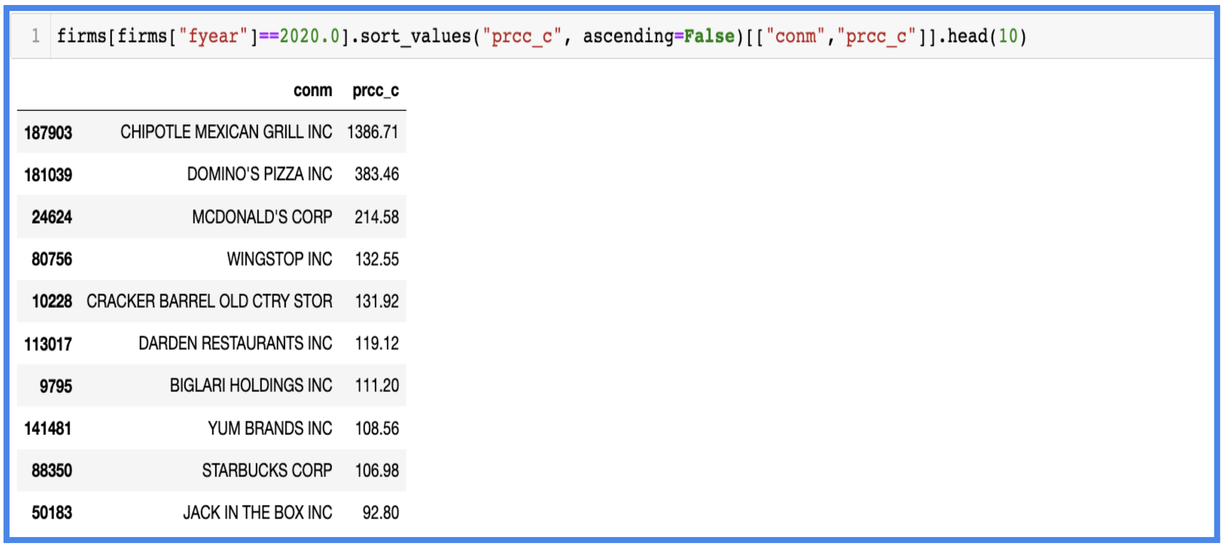


```
#DARDEN RESTAURANTS INC = 27
new=firms.groupby("conm")[["fyear"]].nunique()
new[new["fyear"]==27]
```

	fyear
conm	
DARDEN RESTAURANTS INC	27

B [Preliminary Analysis]
1. What are the top 10 firms with the highest stock price (column "prcc_c") in the year 2020?

We filtered the **firms** DataFrame for the year 2020 and we sorted in the descending order based on **prcc_c** stock prices. From the sorted data, we selected columns **conm** and **prcc_c** . Ultimately, we extracted the top 10 companies with the highest stock prices in 2020 by using head method.

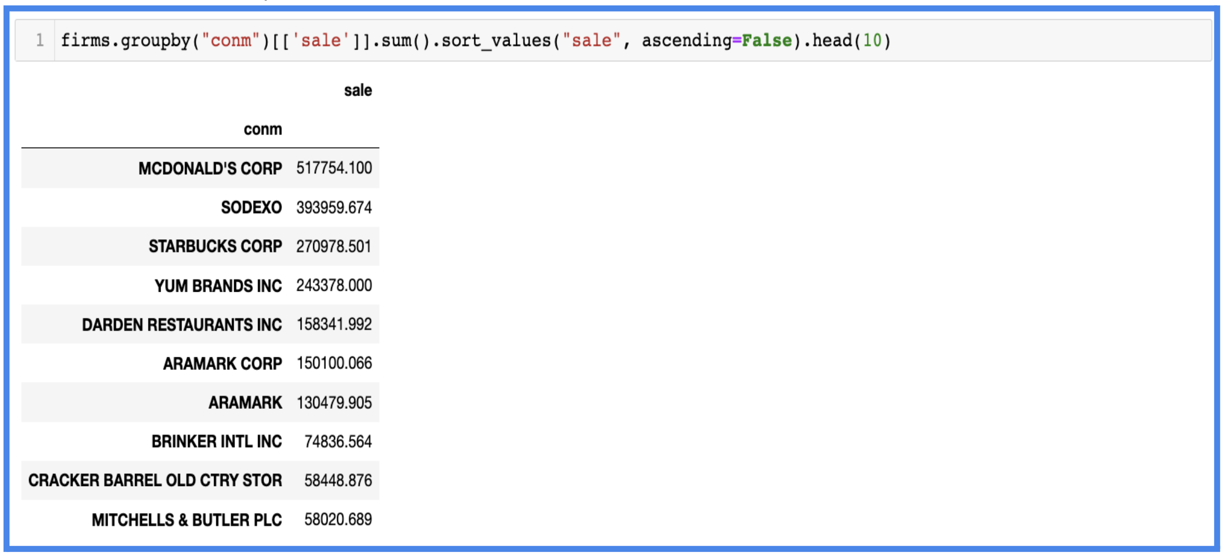


```
firms[firms["fyear"]==2020.0].sort_values("prcc_c", ascending=False)[["conm","prcc_c"]].head(10)
```

	conm	prcc_c
187903	CHIPOTLE MEXICAN GRILL INC	1386.71
181039	DOMINO'S PIZZA INC	383.46
24624	MCDONALD'S CORP	214.58
80756	WINGSTOP INC	132.55
10228	CRACKER BARREL OLD CTRY STOR	131.92
113017	DARDEN RESTAURANTS INC	119.12
9795	BIGLARI HOLDINGS INC	111.20
141481	YUM BRANDS INC	108.56
88350	STARBUCKS CORP	106.98
50183	JACK IN THE BOX INC	92.80

2. What are the top 10 firms with the highest sales (column "sale") in the entire history of the dataset?

We grouped the companies by the sum of sales across all years for every company and sorted the sum of sales in the descending order. Finally, we got the top 10 firms with the highest total sales across all years.

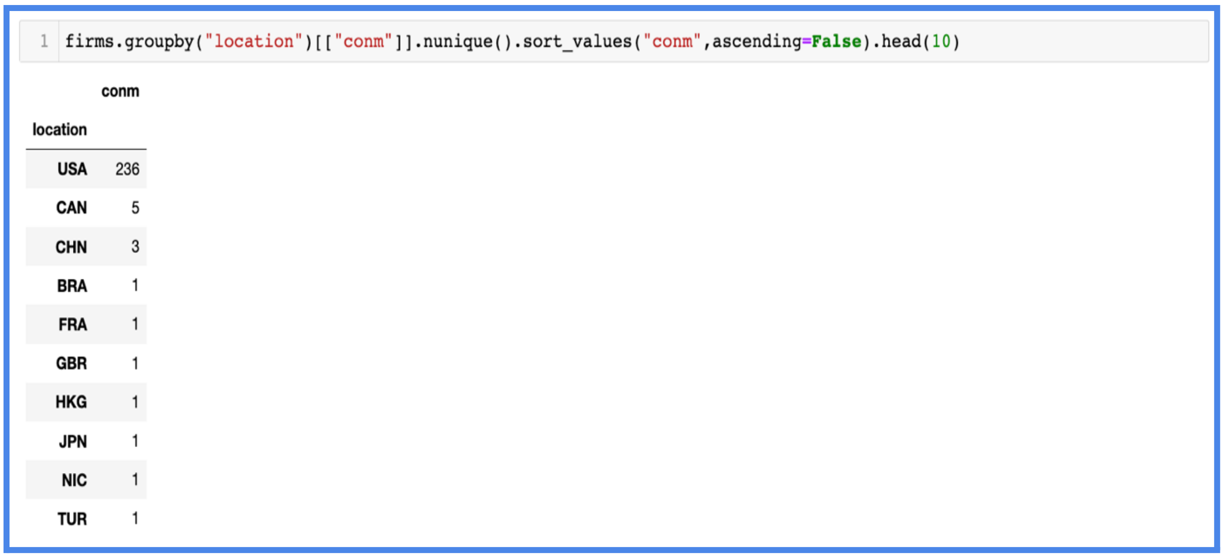


```
firms.groupby("conm")[['sale']].sum().sort_values("sale", ascending=False).head(10)
```

	sale
conm	
MCDONALD'S CORP	517754.100
SODEXO	393959.674
STARBUCKS CORP	270978.501
YUM BRANDS INC	243378.000
DARDEN RESTAURANTS INC	158341.992
ARAMARK CORP	150100.066
ARAMARK	130479.905
BRINKER INTL INC	74836.564
CRACKER BARREL OLD CTRY STOR	58448.876
MITCHELLS & BUTLER PLC	58020.689

3. What is the geographical distribution (column "location") of all the firms? In other words, how many firms are there in each location? Please list the top 10 locations.

We grouped the data by **location** column, and within each location (country) group, we calculated the unique company names. Then we sorted them in descending order by the number of unique firms in that location/country. Ultimately, we displayed the top 10 locations with the most unique companies.



```python
firms.groupby("location")[["conm"]].nunique().sort_values("conm",ascending=False).head(10)
```

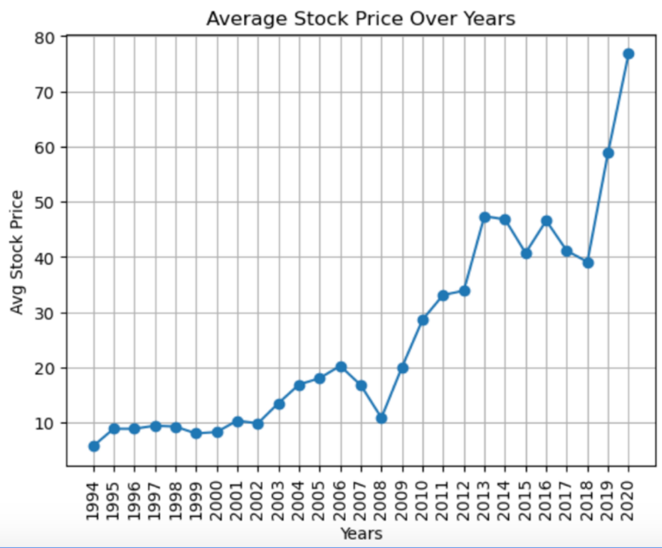
	conm
location	
USA	236
CAN	5
CHN	3
BRA	1
FRA	1
GBR	1
HKG	1
JPN	1
NIC	1
TUR	1

4. Create a line chart to show the average stock price (column "prcc_c") in the selected sector(s) across the years. If you have selected multiple sectors, draw multiple lines to show them separately.

We have imported the matplotlib library to plot the line chart.

a. We calculated the mean values of the stock prices for each fiscal year and stored it in the new DataFrame **line_c.**
b. We assigned the index i.e., year values to variable **x** and the mean values of stock prices to variable **y**.
c. We plotted the line chart using plt.plot() and added formatting applying the functions available to plotting.

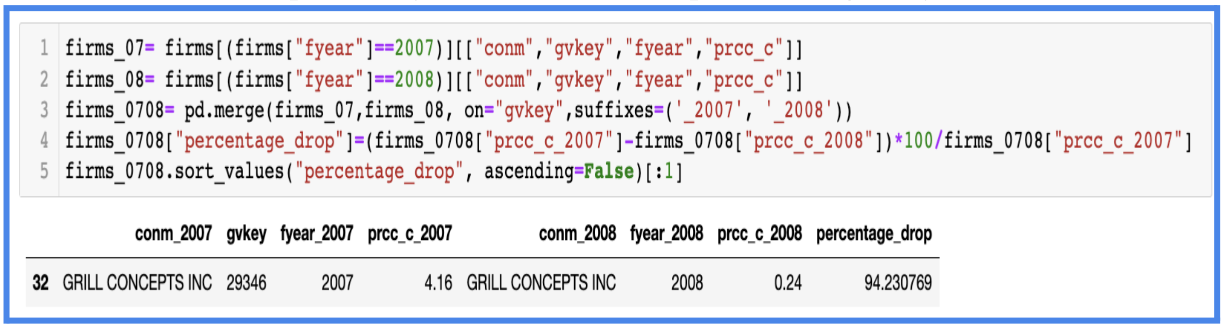
```python
import matplotlib.pyplot as plt
line_c=firms.groupby("fyear")[["prcc_c"]].mean()
x=line_c.index
y=line_c["prcc_c"]
plt.plot(x,y, marker='o')
plt.xticks(x,rotation=90)
plt.title("Average Stock Price Over Years")
plt.xlabel("Years")
plt.ylabel("Avg Stock Price")
plt.grid()
```



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

We used the following approach:

a. We created two new data frames based on **firms** DataFrame for fiscal year 2007 and fiscal year 2008 separately and selected the columns on company names, gvkey, fiscal year, and stock prices.
b. We merged the two data frames on the **gvkey** column to create **firms_0708** DataFrame.
c. We calculated the percentage drop using the formula (the stock price of 2007-the stock price of 2008)*100/the stock price of 2007 and stored it in a new column called **percentage_drop.**
d. We sorted the **firms_0708** DataFrame in descending order based on the **percentage_drop** column to identify the company with the most significant stock price decrease in 2008. Our analysis reveals that **Grill Concepts Inc.** experienced a remarkable drop of nearly 94.23% in its stock prices during that year.

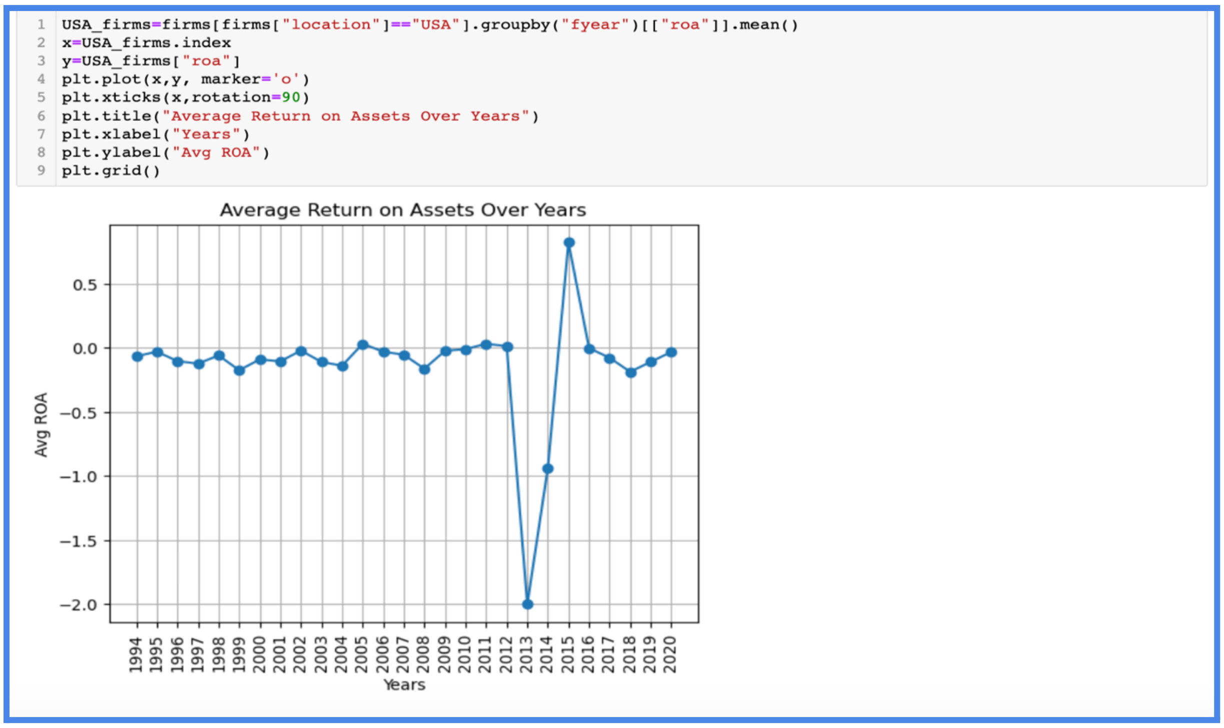


```
firms_07= firms[(firms["fyear"]==2007)][["conm","gvkey","fyear","prcc_c"]]
firms_08= firms[(firms["fyear"]==2008)][["conm","gvkey","fyear","prcc_c"]]
firms_0708= pd.merge(firms_07,firms_08, on="gvkey",suffixes=('_2007', '_2008'))
firms_0708["percentage_drop"]=(firms_0708["prcc_c_2007"]-firms_0708["prcc_c_2008"])*100/firms_0708["prcc_c_2007"]
firms_0708.sort_values("percentage_drop", ascending=False)[:1]
```

	conm_2007	gvkey	fyear_2007	prcc_c_2007	conm_2008	fyear_2008	prcc_c_2008	percentage_drop
32	GRILL CONCEPTS INC	29346	2007	4.16	GRILL CONCEPTS INC	2008	0.24	94.230769

6. Plot the average Return on Assets (ROA) for the firms located in the "USA" across the years. ROA is calculated as ni/asset.

a. We filtered the **location** column as 'USA' to extract records for companies only located in the U.S. Then we grouped the data by fiscal year and calculated the average return on asset (**roa** column) for each year.
b. We assigned the index i.e., **fyear** values to variable x and the mean values of **roa** to variable y.
c. We plotted the line chart using plt.plot() and added formatting using the functions available for plotting.



```python
USA_firms=firms[firms["location"]=="USA"].groupby("fyear")[["roa"]].mean()
x=USA_firms.index
y=USA_firms["roa"]
plt.plot(x,y, marker='o')
plt.xticks(x,rotation=90)
plt.title("Average Return on Assets Over Years")
plt.xlabel("Years")
plt.ylabel("Avg ROA")
plt.grid()
```

C. [Text Cleaning]

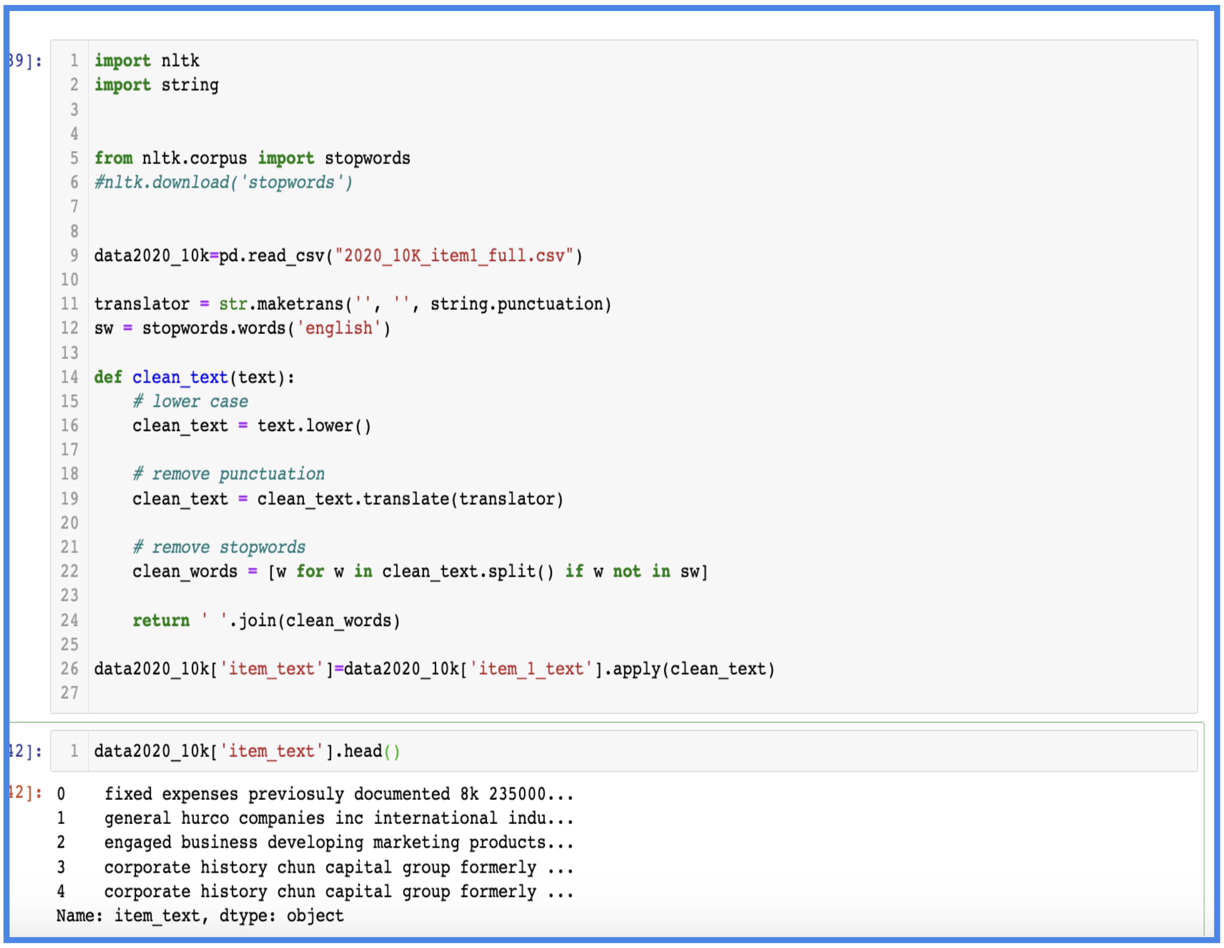
1. We imported the Natural Language Toolkit (NLTK) library. NLTK imports a list of stopwords. Stopwords are common words like the, and, is, etc., that are often removed from text data because they do not carry significant meanings.

2. We stored the records from a csv file named **2020_10K_item1_full.csv** into a new dataframe called **2020_10k**. We then created a translator object using str.maketrans to remove punctuations from text. The string.punctuation variable contains a string of punctuation characters such as ".", ",", "!", etc.

3. We created a function called **clean_text(text)** that takes a text input and performs a simple text preprocessing step. In this function, the input text is converted to lowercase.

4. We removed punctuations from the text using the translator object. Finally, we split the cleaned text into words, removed any stopwords from the words, and joined the remaining words back into a single string.

5. We used the apply function passing the clean_text function to the **item_1_text** column of the **data2020_10k** DataFrame and storing the cleaned text in a new column called **item_text**. We displayed the result of 5 rows from **data2020_10k** having clean data in the **item_text** column.



```
39]:
import nltk
import string


from nltk.corpus import stopwords
#nltk.download('stopwords')


data2020_10k=pd.read_csv("2020_10K_item1_full.csv")

translator = str.maketrans('', '', string.punctuation)
sw = stopwords.words('english')

def clean_text(text):
    # lower case
    clean_text = text.lower()

    # remove punctuation
    clean_text = clean_text.translate(translator)

    # remove stopwords
    clean_words = [w for w in clean_text.split() if w not in sw]

    return ' '.join(clean_words)

data2020_10k['item_text']=data2020_10k['item_1_text'].apply(clean_text)

```

```
42]:
data2020_10k['item_text'].head()
```

```
42]: 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 ...
     Name: item_text, dtype: object
```

D [Keyword Analysis]

1. Inner Join

In our analysis, we initially imported the data from 2020_10K_item1_full.csv and stored it in **data2020_10k**. To prepare the text data, we applied the **clean_text(text)** function to the **item_1_text** column within the **data2020_10k** dataset. This process involved converting text to lowercase, removing punctuation, and eliminating stopwords. The cleaned text was saved in a new column named **item_text**.

Furthermore, we conducted data filtration on the **firms** dataset to isolate information pertaining to companies within our designated industries for the year 2020. This refined dataset was stored in a new variable denoted as **firms_2020**.

Subsequently, an inner join operation was executed to merge the datasets **data2020_10k** and **firms_2020** based on the shared primary key, **gvkey**. The resultant integrated dataset was assigned to the variable **firms_10K**.

2. Steps Implemented for Keyword Distribution Using Word Count

a. The initial step involved filtering the firm data from **public_firms.csv** to extract a comprehensive list of companies within the Eating and Drinking sector, focusing on data specific to the year 2020.

b. Following the filtration, we utilized the Counter class from the collections library. This library encompasses classes akin to the Counter class, facilitating efficient handling of data frames. The Counter class, specifically employed for tallying element occurrences within an iterable (e.g., a list or string), constructs a dictionary-like object where elements serve as keys, and their corresponding counts are denoted as values. This functionality proves valuable for tasks such as identifying prevalent elements or identifying duplicates within a collection.

c. Subsequently, we created a Python function named **get_keyword_wc**. This function accepts text inputs and returns a string comprising the top ten most frequent words from the provided text, separated by spaces.

d. The ensuing code execution involves multiple steps. Initially, it analyzes a given piece of text, counting the frequency of each word by segmenting the text into individual words. The top 10 most common words are stored within an empty list, denoted as **words**. The code then iterates through these top 10 words, incorporating each word into the **words** list. Ultimately, it consolidates these top 10 words into a single text string, delineated by spaces. This process assists in identifying and presenting the most frequently occurring words in the input text, which proves valuable for summarization or text analysis purposes.

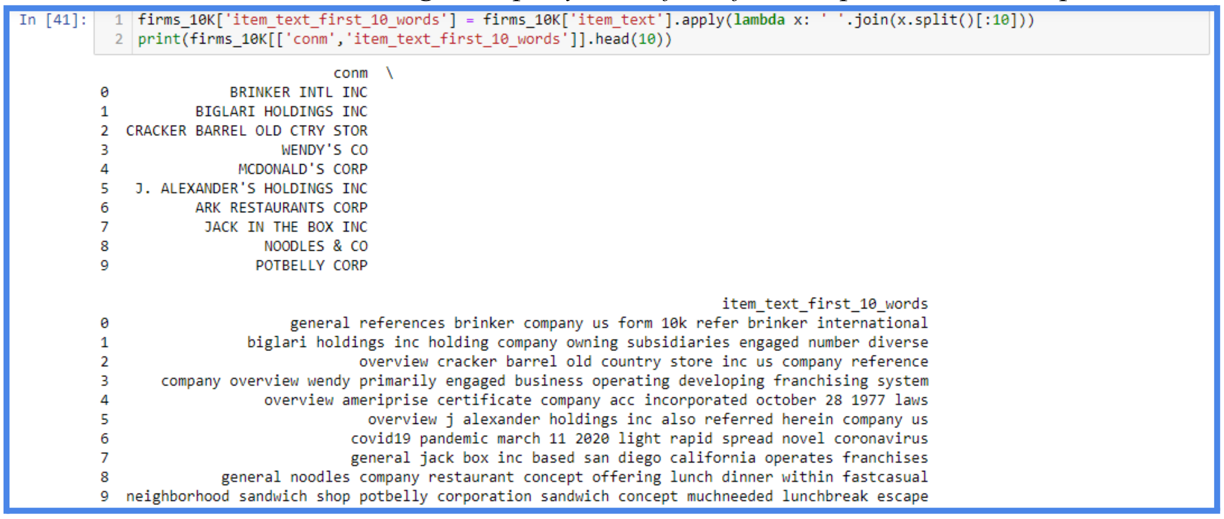
e. Moving forward, we applied the **get_keyword_wc** function to the **item_text** column within the **firms_10K** dataset, consequently yielding the desired outcomes, stored within the **keyword_clean_wc** column of the same dataset.

f. Using the apply function and lambda function, we isolated the first 10 words from the 'item_text' column by splitting the text and rejoining the selected words.

g. The resultant string was stored in a new column named **item_text_first_10_words**, and we showcased the company name (**conm** column) along with these first 10 words for the initial 10 rows.

Results

Results indicating the top keywords for a few companies in our scope



```
In [41]:
firms_10K['item_text_first_10_words'] = firms_10K['item_text'].apply(lambda x: ' '.join(x.split()[:10]))
print(firms_10K[['conm','item_text_first_10_words']].head(10))
```

```
                         conm  \
0            BRINKER INTL INC
1        BIGLARI HOLDINGS INC
2  CRACKER BARREL OLD CTRY STOR
3                   WENDY'S CO
4              MCDONALD'S CORP
5   J. ALEXANDER'S HOLDINGS INC
6        ARK RESTAURANTS CORP
7          JACK IN THE BOX INC
8                 NOODLES & CO
9               POTBELLY CORP

                                             item_text_first_10_words
0              general references brinker company us form 10k refer brinker international
1         biglari holdings inc holding company owning subsidiaries engaged number diverse
2                    overview cracker barrel old country store inc us company reference
3    company overview wendy primarily engaged business operating developing franchising system
4             overview ameriprise certificate company acc incorporated october 28 1977 laws
5                     overview j alexander holdings inc also referred herein company us
6                  covid19 pandemic march 11 2020 light rapid spread novel coronavirus
7                  general jack box inc based san diego california operates franchises
8          general noodles company restaurant concept offering lunch dinner within fastcasual
9  neighborhood sandwich shop potbelly corporation sandwich concept muchneeded lunchbreak escape
```

3. TF-IDF (Term Frequency-Inverse Document Frequency)

TF-IDF, short for Term Frequency-Inverse Document Frequency, is a fundamental technique in natural language processing and information retrieval. It serves to assess the significance of a word within a document concerning a larger collection of documents, known as a corpus.

It comprises two components:

a. Term Frequency (TF): Measures the frequency of a term (word) within a specific document, calculated by counting the occurrences of the term in that document.
b. Inverse Document Frequency (IDF): Evaluates the significance of a term in the entire corpus, computed as the logarithm of the total number of documents in the corpus divided by the number of documents containing the term.

The TF-IDF score is obtained by multiplying the TF and IDF values for each term in a document. This score helps identify and extract top keywords from the dataset based on their importance.

4. Procedure for Keyword Extraction using TF-IDF

a. We imported the **TFIDFVectorizer** from **sklearn.feature_extraction.text.** A function called **get_keywords_tfidf** was created to accept a document as input and return the top keywords in our dataset based on their TF-IDF scores.

b. A **TF-IDF vectorizer** was established to convert text documents into numerical representations, aiding in identifying crucial words.

c. Using the vectorizer, we computed TF-IDF values for all words in our documents, resulting in a matrix illustrating the importance of each word in each document.

d. We extracted the top 10 keywords from our dataset by collecting the words with the highest TF-IDF scores.

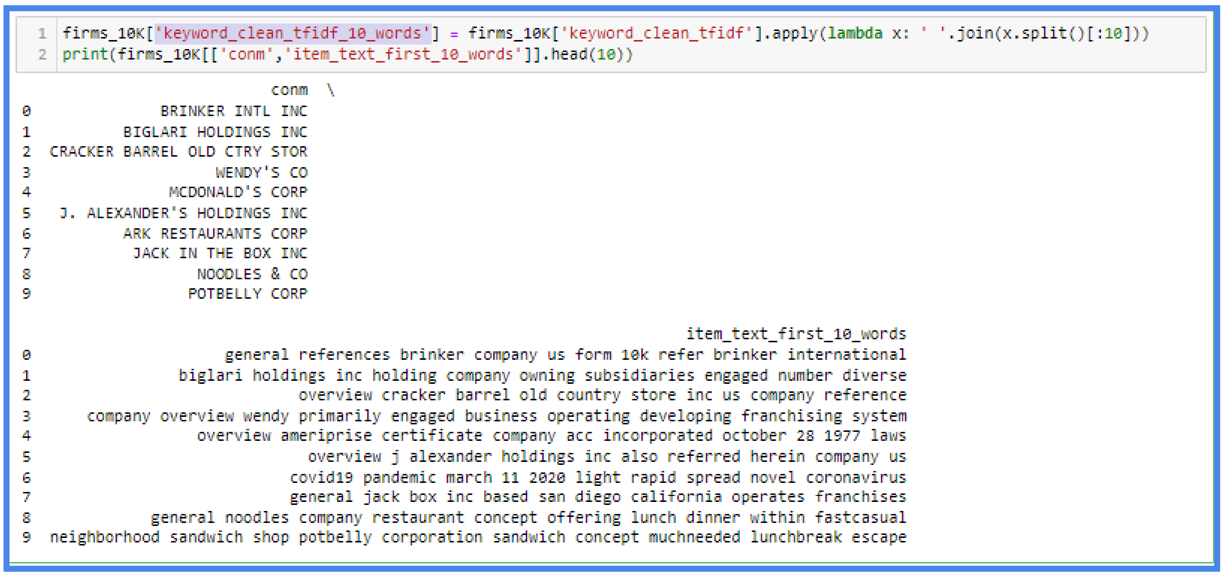
e. The **get_keywords_tfidf** function was applied to the **item_text** column within the **firm2020_10K** dataset. The resulting top keywords were stored in a new column named **keyword_clean_tfidf** within the same dataset.

f. Using the apply function and lambda function, we isolated the first 10 words from the **item_text** column by splitting the text and rejoining the selected words.

g. The resultant string was stored in a new column named **keyword_clean_tfidf_10_words** and we showcased the company name (**conm** column) along with these first 10 words for the initial 10 rows.

Results

Results indicating the top keywords through TF-IDF vector



```
firms_10K['keyword_clean_tfidf_10_words'] = firms_10K['keyword_clean_tfidf'].apply(lambda x: ' '.join(x.split()[:10]))
print(firms_10K[['conm','item_text_first_10_words']].head(10))
```

```
                         conm  \
0            BRINKER INTL INC
1        BIGLARI HOLDINGS INC
2  CRACKER BARREL OLD CTRY STOR
3                  WENDY'S CO
4             MCDONALD'S CORP
5   J. ALEXANDER'S HOLDINGS INC
6        ARK RESTAURANTS CORP
7         JACK IN THE BOX INC
8                NOODLES & CO
9               POTBELLY CORP

                                            item_text_first_10_words
0             general references brinker company us form 10k refer brinker international
1        biglari holdings inc holding company owning subsidiaries engaged number diverse
2                   overview cracker barrel old country store inc us company reference
3     company overview wendy primarily engaged business operating developing franchising system
4          overview ameriprise certificate company acc incorporated october 28 1977 laws
5                    overview j alexander holdings inc also referred herein company us
6                  covid19 pandemic march 11 2020 light rapid spread novel coronavirus
7                  general jack box inc based san diego california operates franchises
8           general noodles company restaurant concept offering lunch dinner within fastcasual
9  neighborhood sandwich shop potbelly corporation sandwich concept muchneeded lunchbreak escape
```

E. [Word Embedding]

1. Word Cloud

a. The code started by installing the **wordcloud** library using Conda, a package manager for Python, ensuring the **wordcloud** library is available for use.

b. Imported the required libraries, **WordCloud** and **matplotlib.pyplot**, for generating and displaying a word cloud visualization.

c. Concatenated the **keyword_clean_wc/keyword_clean_tfidf** column from the **firms_10K** dataset, converting it to a single string using join(). This string will serve as the input for generating the word cloud.

d. Utilized the **WordCloud** class to create a word cloud visualization.

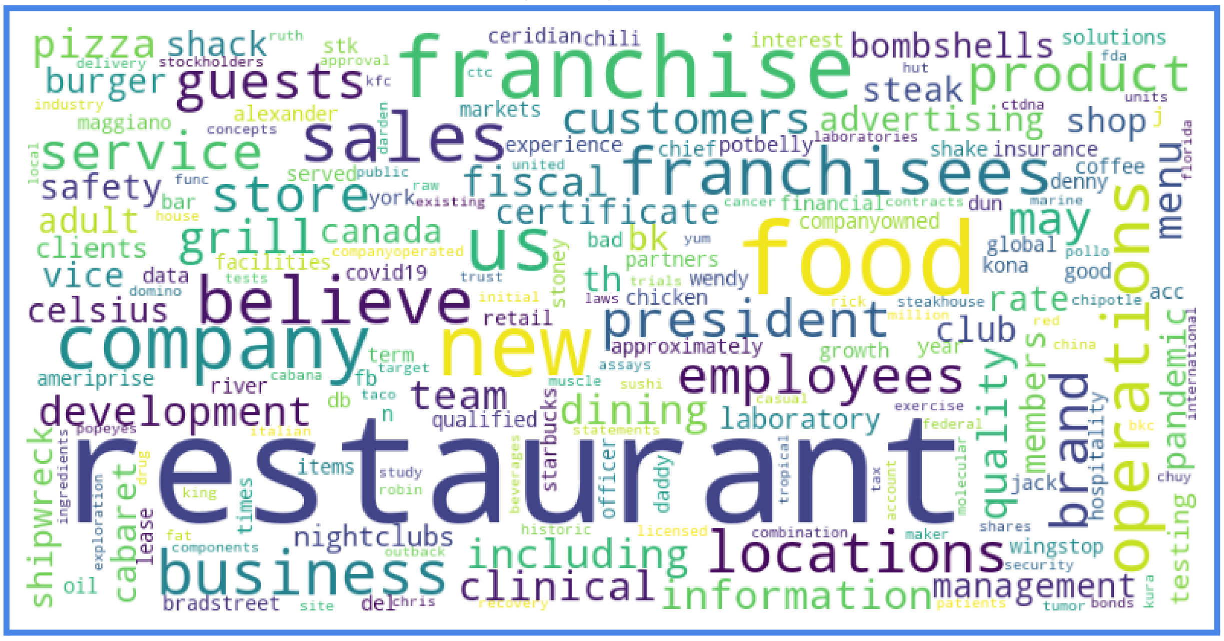
e. Parameters such as width, height, and background color are specified for the word cloud.

f. Configured the size of the plot and displayed the word cloud using imshow() and show() from matplotlib.

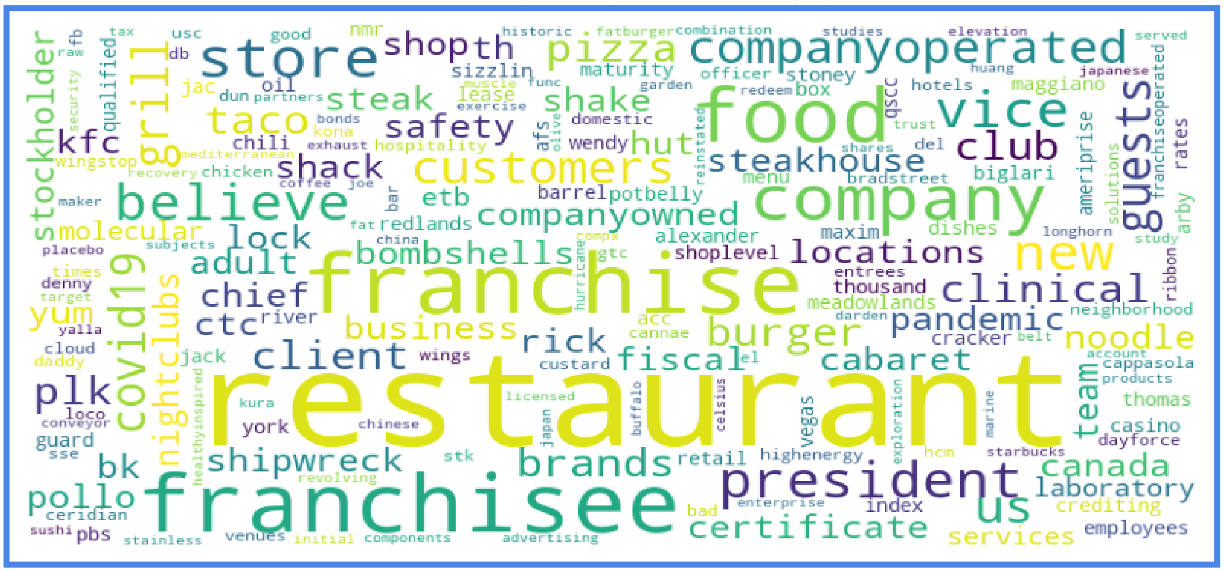
g. Saved the generated word cloud as a PNG file using savefig().

Results

Wordcloud for 'keyword_clean_wc"



Wordcloud for 'keyword_clean_tfidf"



2. Word2Vec

a. Installed the **gensim** library using Conda, a package manager for Python, ensuring the **gensim** library is available for use.

b. Imported the required libraries, **pandas** for data handling and **Word2Vec** from **gensim.models** for training and using **Word2Vec** models.

c. Split the text in the **item_text** column of the **firms_10K** dataset into a list of words. Each document is represented as a list of words.

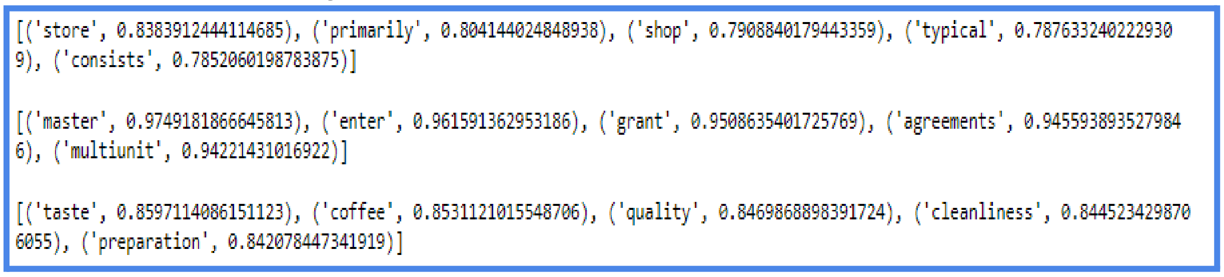
d. Created a **Word2Vec** model using the **Word2Vec** class from **gensim**.

<u>Results</u>

Based on the wordcloud generated in step E.1.c. we selected the following three words to run in the **Word2Vec** model:

a. restaurant - store, primarily, shop, typical, consists
b. franchisee - master, enter, grant, agreements, multi unit
c. food - taste, coffee, quality, cleanliness, preparation

We obtained the following correlations:

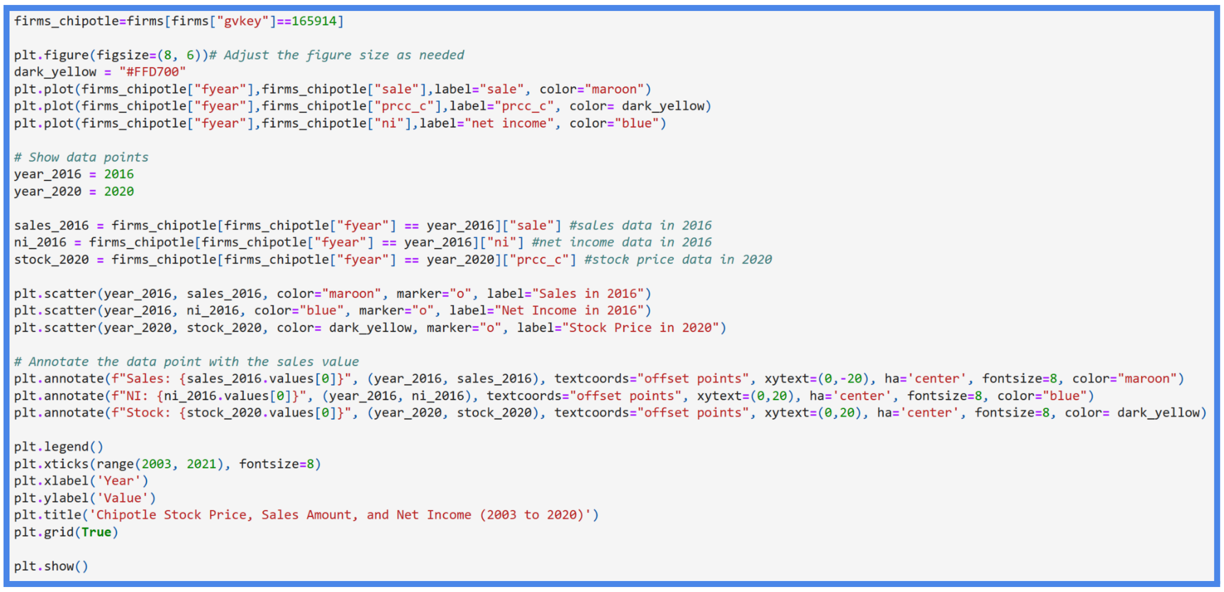


```
[('store', 0.8383912444114685), ('primarily', 0.804144024848938), ('shop', 0.7908840179443359), ('typical', 0.787633240222930
9), ('consists', 0.7852060198783875)]

[('master', 0.9749181866645813), ('enter', 0.961591362953186), ('grant', 0.9508635401725769), ('agreements', 0.945593893527984
6), ('multiunit', 0.94221431016922)]

[('taste', 0.8597114086151123), ('coffee', 0.8531121015548706), ('quality', 0.8469868898391724), ('cleanliness', 0.844523429870
6055), ('preparation', 0.842078447341919)]
```

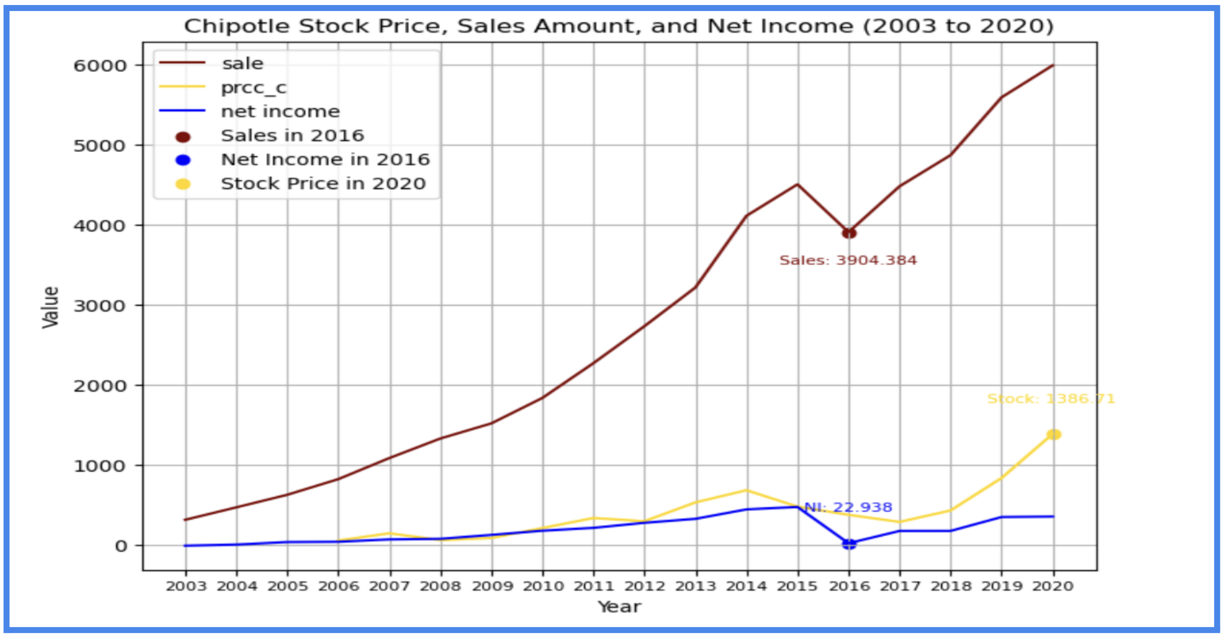
Part 3. Comprehensive Analysis of One Sample Firm

F. [Firm Analysis and Strategy Suggestion]
This is an open question. Pick <u>one</u> firm that you are interested in and try to analyze its market status. The ultimate goal is to provide <u>one valuable suggestion</u> to the firm based on your analysis. Perform an analysis of the historical stock prices, ROA, revenue, and assets of the chosen company. Investigate potential correlations and address noteworthy decreases and increases.

1. The overview of sales, stock prices, and net income of Chipotle from 2003 to 2020.



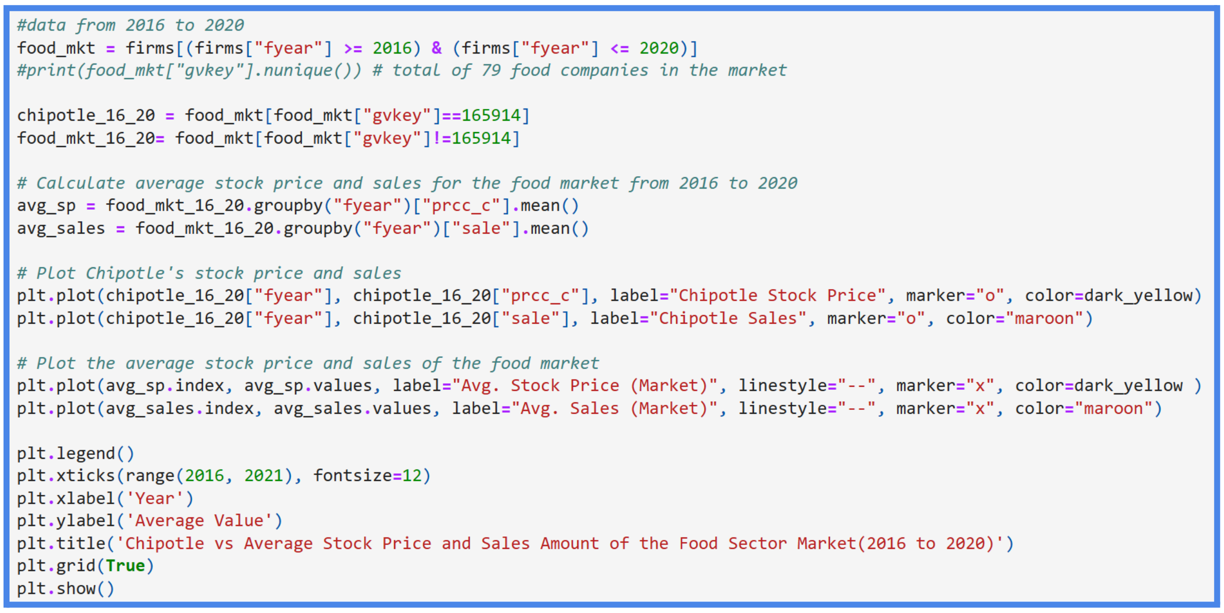
```python
firms_chipotle=firms[firms["gvkey"]==165914]

plt.figure(figsize=(8, 6))# Adjust the figure size as needed
dark_yellow = "#FFD700"
plt.plot(firms_chipotle["fyear"],firms_chipotle["sale"],label="sale", color="maroon")
plt.plot(firms_chipotle["fyear"],firms_chipotle["prcc_c"],label="prcc_c", color= dark_yellow)
plt.plot(firms_chipotle["fyear"],firms_chipotle["ni"],label="net income", color="blue")

# Show data points
year_2016 = 2016
year_2020 = 2020

sales_2016 = firms_chipotle[firms_chipotle["fyear"] == year_2016]["sale"] #sales data in 2016
ni_2016 = firms_chipotle[firms_chipotle["fyear"] == year_2016]["ni"] #net income data in 2016
stock_2020 = firms_chipotle[firms_chipotle["fyear"] == year_2020]["prcc_c"] #stock price data in 2020

plt.scatter(year_2016, sales_2016, color="maroon", marker="o", label="Sales in 2016")
plt.scatter(year_2016, ni_2016, color="blue", marker="o", label="Net Income in 2016")
plt.scatter(year_2020, stock_2020, color= dark_yellow, marker="o", label="Stock Price in 2020")

# Annotate the data point with the sales value
plt.annotate(f"Sales: {sales_2016.values[0]}", (year_2016, sales_2016), textcoords="offset points", xytext=(0,-20), ha='center', fontsize=8, color="maroon")
plt.annotate(f"NI: {ni_2016.values[0]}", (year_2016, ni_2016), textcoords="offset points", xytext=(0,20), ha='center', fontsize=8, color="blue")
plt.annotate(f"Stock: {stock_2020.values[0]}", (year_2020, stock_2020), textcoords="offset points", xytext=(0,20), ha='center', fontsize=8, color= dark_yellow)

plt.legend()
plt.xticks(range(2003, 2021), fontsize=8)
plt.xlabel('Year')
plt.ylabel('Value')
plt.title('Chipotle Stock Price, Sales Amount, and Net Income (2003 to 2020)')
plt.grid(True)

plt.show()
```



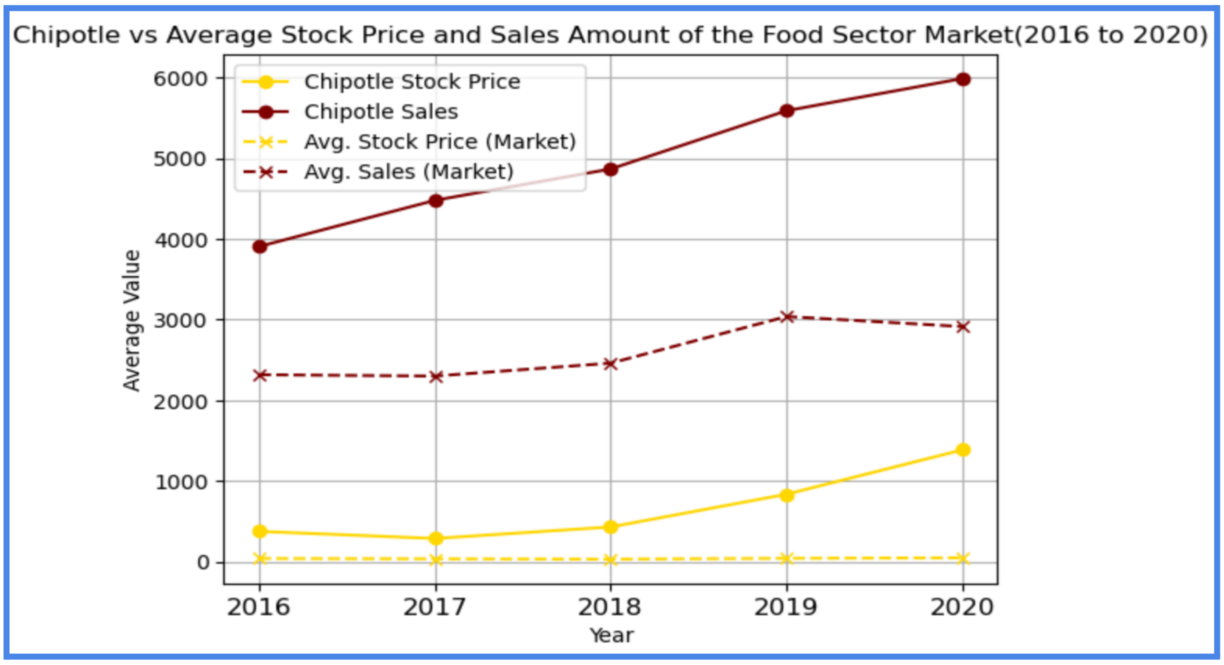
Insights

Chipotle faced declining sales, reaching $3,904.384, and a low net income of $22.938 in 2016. According to some research, Chipotle's stock price experienced a dramatic decline in 2017 due to some food safety concerns. However, from 2018 to 2020, Chipotle's strategic transformation, including enhanced safety measures and digital innovation, rebuilt investor trust. The stock price surged from $289.03 in 2017 to $1,386.71 in 2020. Net income also rebounded, reaching $355.766 in 2020. This reflects Chipotle's impressive financial and market recovery.

2. Comparing the sales amount and stock prices of Chipotle with the average sales and stock prices of the entire food sector market from 2016 to 2020.



```
#data from 2016 to 2020
food_mkt = firms[(firms["fyear"] >= 2016) & (firms["fyear"] <= 2020)]
#print(food_mkt["gvkey"].nunique()) # total of 79 food companies in the market

chipotle_16_20 = food_mkt[food_mkt["gvkey"]==165914]
food_mkt_16_20= food_mkt[food_mkt["gvkey"]!=165914]

# Calculate average stock price and sales for the food market from 2016 to 2020
avg_sp = food_mkt_16_20.groupby("fyear")["prcc_c"].mean()
avg_sales = food_mkt_16_20.groupby("fyear")["sale"].mean()

# Plot Chipotle's stock price and sales
plt.plot(chipotle_16_20["fyear"], chipotle_16_20["prcc_c"], label="Chipotle Stock Price", marker="o", color=dark_yellow)
plt.plot(chipotle_16_20["fyear"], chipotle_16_20["sale"], label="Chipotle Sales", marker="o", color="maroon")

# Plot the average stock price and sales of the food market
plt.plot(avg_sp.index, avg_sp.values, label="Avg. Stock Price (Market)", linestyle="--", marker="x", color=dark_yellow )
plt.plot(avg_sales.index, avg_sales.values, label="Avg. Sales (Market)", linestyle="--", marker="x", color="maroon")

plt.legend()
plt.xticks(range(2016, 2021), fontsize=12)
plt.xlabel('Year')
plt.ylabel('Average Value')
plt.title('Chipotle vs Average Stock Price and Sales Amount of the Food Sector Market(2016 to 2020)')
plt.grid(True)
plt.show()
```

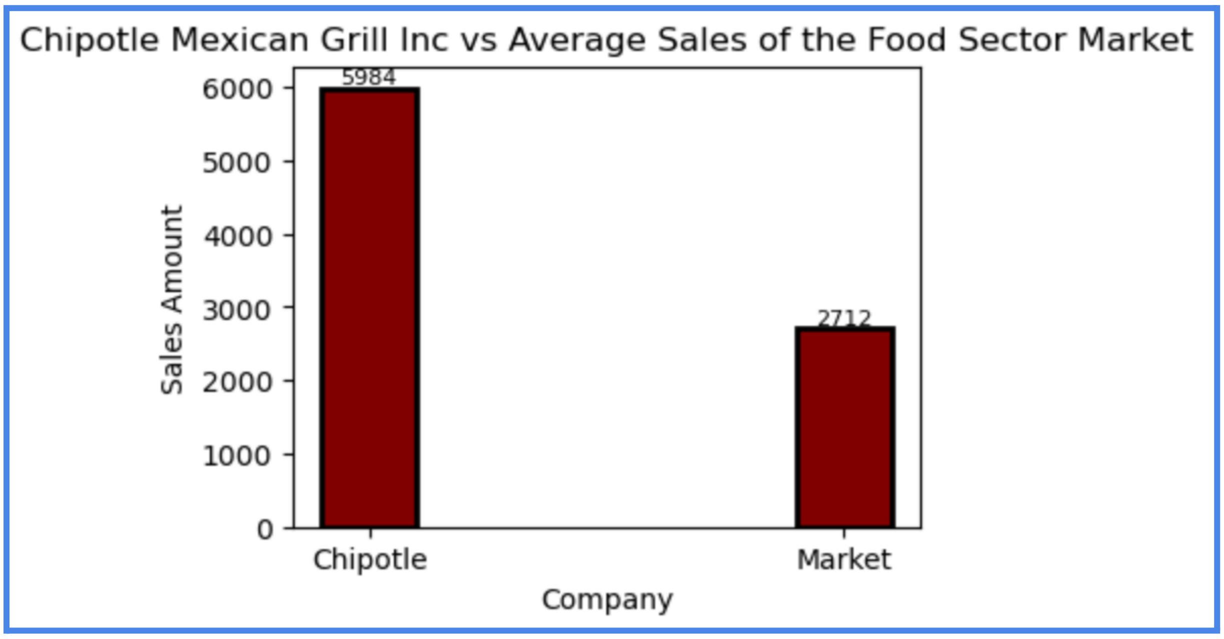


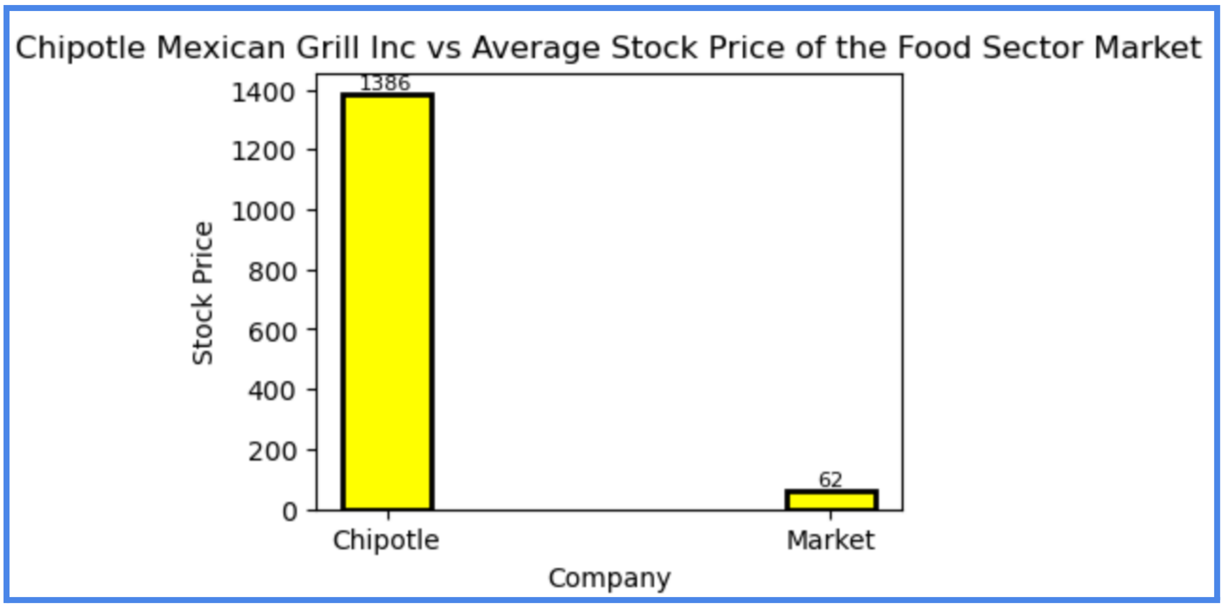
Insights

Despite a significant sales drop in 2016 and a subsequent stock price decline in 2017, Chipotle outperformed the broader food market, comprising 79 companies. In the following years, Chipotle demonstrated remarkable growth, particularly from 2019 to 2020. During this period, while the food market faced challenges, Chipotle's sales and stock price continued to rise, showing the effectiveness of the strategic transformation and innovation from the management.

3. Analyzing and visualizing the sales amount and the stock prices of Chipotle versus the food sector market in 2020 via bar charts.

```python
firm_focal=firms_10K[firms_10K["gvkey"]==165914]
firm_market=firms_10K[firms_10K["gvkey"]!=165914]
print(len(firm_market)) #compared with other 45 companies in the food market

# bar chart for sales amount
ff_x=int(firm_focal["sale"])
fm_x=int(firm_market["sale"].mean())
sale_key=np.array(["Chipotle","Market"])
sale_arr=np.array([ff_x,fm_x])
plt.figure(figsize=(4, 3))# Adjust the figure size as needed
plt.bar(sale_key, sale_arr,width = 0.2,color='maroon', edgecolor='black', linewidth=2, label='Sales')
plt.xlabel('Company')
plt.ylabel('Sales Amount')
plt.title('Chipotle Mexican Grill Inc vs Average Sales of the Food Sector Market')
for i in range(len(sale_key)):
    plt.text(sale_key[i], sale_arr[i], sale_arr[i], ha='center', va='bottom', fontsize=8)

# bar chart for stock price
ff_y=int(firm_focal["prcc_c"])
fm_y=int(firm_market["prcc_c"].mean())
sale_key=np.array(["Chipotle","Market"])
sale_arr=np.array([ff_y,fm_y])
plt.figure(figsize=(4, 3))# Adjust the figure size as needed
plt.bar(sale_key, sale_arr,width = 0.2,color='yellow', edgecolor='black', linewidth=2, label='Sales')
plt.xlabel('Company')
plt.ylabel('Stock Price')
plt.title('Chipotle Mexican Grill Inc vs Average Stock Price of the Food Sector Market')
for i in range(len(sale_key)):
    plt.text(sale_key[i], sale_arr[i], sale_arr[i], ha='center', va='bottom', fontsize=8)

# Show the two separate figures
plt.show()
```





<u>Insights</u>

From the bar graphs, we can tell Chipotle outperformed its peers in the food sector market in 2020. With sales totaling $5,984, it exceeded the average sales of 45 other food companies by 2.2 times. Chipotle's stock price reached $1,386, over 21 times higher than the market average.

References

To copy edit our report and get syntax for merging and matplotlib formatting options

OpenAI. (Year). GPT-3.5 or ChatGPT [Computer software]. Retrieved from https://www.openai.com/chatgpt