quUniversity of British Columbia

Sauder School of Business

BAIT 508 Business Analytics Programming

Group Project: Industry Analysis

Oil and Gas Extraction Analysis using Python

‘

Team Members:

Novar Qin BA2	**Hanlin Zhao BA2**	**Bohan Yue BA2**
Student Number: 92408798	Student Number: 54357223	Student Number: 37095650
Email: novarqin@gmail.com	Email: zhl18053223226@gmail.com	Email: yuebohan8859@gmail.com

Role and Responsibilities:

Novar:

- Python: Write code for Part F; Code check for previous parts
- Project Report: draft the content for report (Part A-C/Part E); formating of project report

Hanlin:

- Python: Write code for Part B, E
- Project Report: draft the content for report (Part F, E)

Allen:

- Python: Write code for Part A, C&D
- Project Report: draft the content for report (Part C)

PART 1. QUANTITATIVE ANALYSIS OF THE INDUSTRY SECTOR

A. Industry Sector Selection and Data Filtering

To start, we read the two files, “major_group.csv” and “public_firm.csv”[1]. We also import pandas to conduct data analysis.

As the world shifts towards more sustainable and renewable energy sources, the oil and gas industry is adapting to new challenges and opportunities. Therefore, we have selected “Oil and Gas Extraction” (major_group = 13) as our target industry to explore its evolving role in this energy transition and to discover innovative strategies that companies are adopting to to transformate their business while maintaining economic relevance.

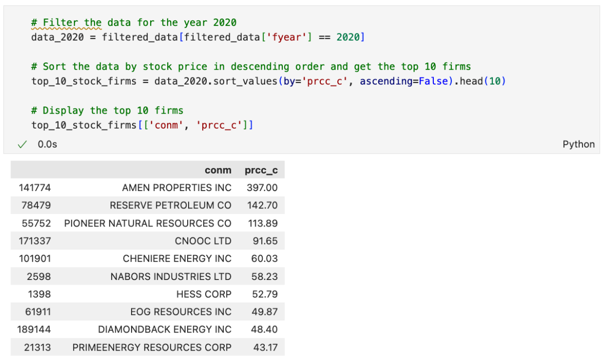
We use “gvkey” (Global Company Key) as the unique identifier. After filtering the dataset using the first two digits of SIC code as “13”, here are some basic information we have obtained for the Oil and Gas industry:

- There are 27 unique firm-years.
- There are 793 unique firms.
- 4 firms have records over all 27 years (1994-2020)

B. Preliminary Analysis

1. ***Top 10 firms with the highest stock price in 2020*:**

   The “filtered data” contains only firms in the Oil and Gas Extraction industry. By filtering on the year 2020, sort the values by stock price (col “prcc_c”) in descending order and use the .head() function to get the top 10 firms with the highest stock price. The result is recorded in the new dataframe, named “tops_10_stock_firms”.

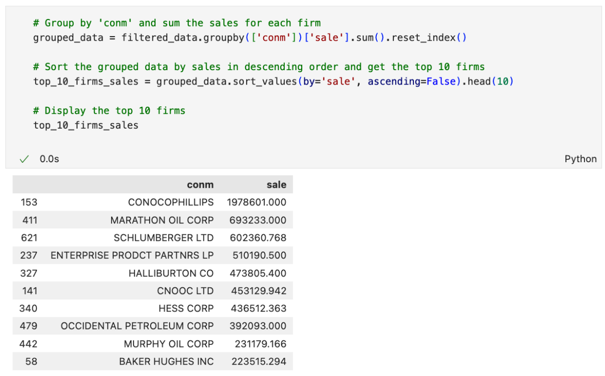


```
# Filter the data for the year 2020
data_2020 = filtered_data[filtered_data['fyear'] == 2020]

# Sort the data by stock price in descending order and get the top 10 firms
top_10_stock_firms = data_2020.sort_values(by='prcc_c', ascending=False).head(10)

# Display the top 10 firms
top_10_stock_firms[['conm', 'prcc_c']]
```
✓ 0.0s Python

	conm	prcc_c
141774	AMEN PROPERTIES INC	397.00
78479	RESERVE PETROLEUM CO	142.70
55752	PIONEER NATURAL RESOURCES CO	113.89
171337	CNOOC LTD	91.65
101901	CHENIERE ENERGY INC	60.03
2598	NABORS INDUSTRIES LTD	58.23
1398	HESS CORP	52.79
61911	EOG RESOURCES INC	49.87
189144	DIAMONDBACK ENERGY INC	48.40
21313	PRIMEENERGY RESOURCES CORP	43.17

2. ***Top 10 firms with highest sales:***

   We use “groupby” to group the same company (column “conm”) together, and sum the sales for each firm. Then we use *“sorted_values()”* to sort the grouped data by sales, and use *head()* to get the top 10 firms. The result is recorded in the new dataframe, named “tops_10_firms_sales”.



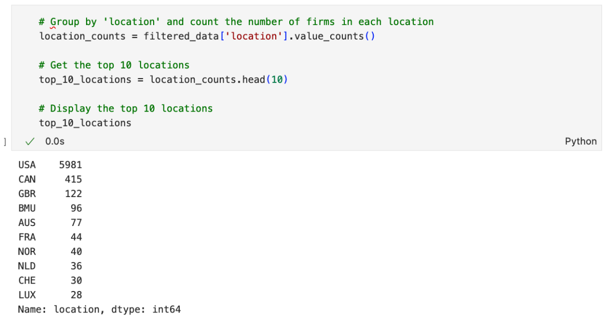
```
# Group by 'conm' and sum the sales for each firm
grouped_data = filtered_data.groupby(['conm'])['sale'].sum().reset_index()

# Sort the grouped data by sales in descending order and get the top 10 firms
top_10_firms_sales = grouped_data.sort_values(by='sale', ascending=False).head(10)

# Display the top 10 firms
top_10_firms_sales
```
✓ 0.0s Python

	conm	sale
153	CONOCOPHILLIPS	1978601.000
411	MARATHON OIL CORP	693233.000
621	SCHLUMBERGER LTD	602360.768
237	ENTERPRISE PRODCT PARTNRS LP	510190.500
327	HALLIBURTON CO	473805.400
141	CNOOC LTD	453129.942
340	HESS CORP	436512.363
479	OCCIDENTAL PETROLEUM CORP	392093.000
442	MURPHY OIL CORP	231179.166
58	BAKER HUGHES INC	223515.294

[1] The two files are saved in the same Jupyter notebook

3. ***Geographic distribution:***

   Based on the “filtered_data”, group by “location” and use *.value_count()* method to count the number of firms in each location. Then use *.head(10)* to get the top 10 locations for companies in the Oil and Gas Extraction Industry.

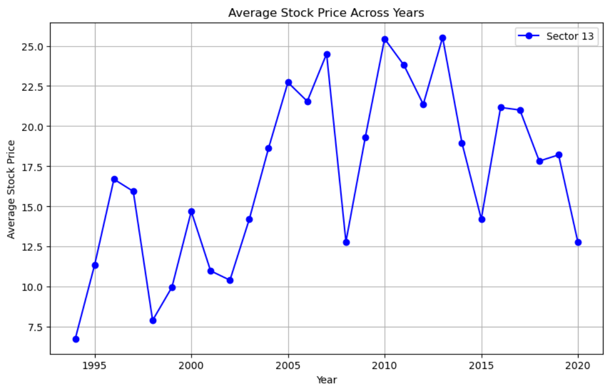


```
# Group by 'location' and count the number of firms in each location
location_counts = filtered_data['location'].value_counts()

# Get the top 10 locations
top_10_locations = location_counts.head(10)

# Display the top 10 locations
top_10_locations
```
```
✓ 0.0s                                                    Python
```
```
USA    5981
CAN     415
GBR     122
BMU      96
AUS      77
FRA      44
NOR      40
NLD      36
CHE      30
LUX      28
Name: location, dtype: int64
```

4. ***Line chart for average stock price of Oil and Gas Extraction Industry:***

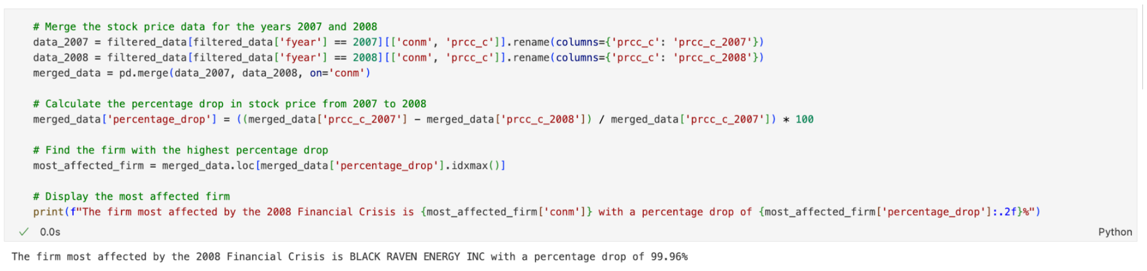
   Import *matplotlib.pyplot* package to create line chat. We calculate the average_stock_price for each year by grouping years and take the mean. We set the x-axis as “Year”, which is the index for average_stock_price, and y axis as “Average Stock Price”.

   Insights: The line chart highlights the inherent volatility of the oil and gas industry, characterized by several significant price fluctuations over the years. Notably, the sharp drops in 2008, 2015, and 2020 coincide with key global economic crises including 2008 Financial Crisis, 2014-2015 Oil Price Crash, and 2020 Pandemic.



5. ***Firm affected most by 2008 Financial Crisis:***

   First, filter the dataset to get two sub-dataset for 2007 and 2008, separately. Then merge the two sub-dataset based on company names (“conm”). Then calculate the percentage drop in stock price for each company, and use idxmax() to identify the company with the highest percentage drop, which is **Black Raven Energy Inc**.

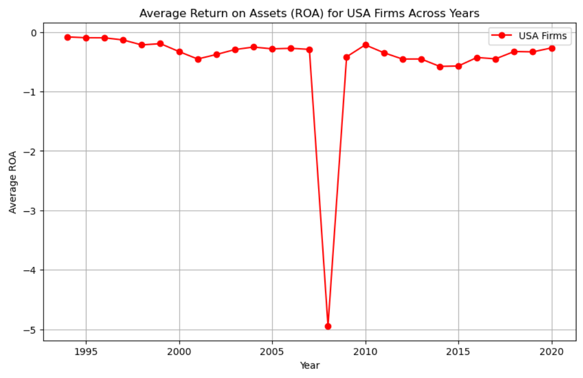


```
# Merge the stock price data for the years 2007 and 2008
data_2007 = filtered_data[filtered_data['fyear'] == 2007][['conm', 'prcc_c']].rename(columns={'prcc_c': 'prcc_c_2007'})
data_2008 = filtered_data[filtered_data['fyear'] == 2008][['conm', 'prcc_c']].rename(columns={'prcc_c': 'prcc_c_2008'})
merged_data = pd.merge(data_2007, data_2008, on='conm')

# Calculate the percentage drop in stock price from 2007 to 2008
merged_data['percentage_drop'] = ((merged_data['prcc_c_2007'] - merged_data['prcc_c_2008']) / merged_data['prcc_c_2007']) * 100

# Find the firm with the highest percentage drop
most_affected_firm = merged_data.loc[merged_data['percentage_drop'].idxmax()]

# Display the most affected firm
print(f"The firm most affected by the 2008 Financial Crisis is {most_affected_firm['conm']} with a percentage drop of {most_affected_firm['percentage_drop']:.2f}%")
```
```
✓ 0.0s                                                    Python
```
```
The firm most affected by the 2008 Financial Crisis is BLACK RAVEN ENERGY INC with a percentage drop of 99.96%
```

6. ***Plot ROA for US companies***

   First, we calculate ROA as ni/asset. Then we only filter firms where location is "USA". We then use *groupby* and calculate average, similar to the previous line chart to get average_roa_usa. Lastly, we plot the line chart.

   Insights: Prior to the 2008 Financial Crisis, the Oil and Gas industry's ROA was at around the break-even point. Oil and Gas is an asset-heavy industry, which is particularly vulnerable to external market shock, that is why we saw a dramatic drop in ROA in 2008. However, the industry quickly recovered afterward, while the ROA went down during the 2014-2015 oil price crash, ROA was not severely hurt as it was in 2008.



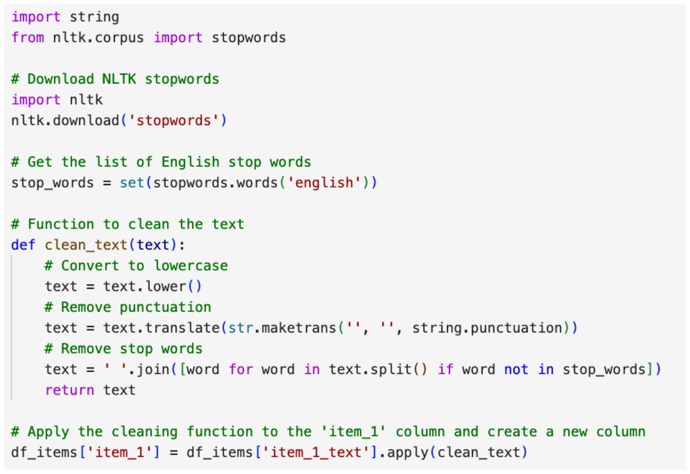
<u>PART 2. TEXT ANALYSIS ON THE INDUSTRY SECTOR</u>

C. Text Cleaning

First, we download the NLTK stopwords and retrieve a list of common English stop words. To streamline the text preprocessing process, we define a custom function called *clean_text()*, which performs the following tasks:

1. ***Convert all words into lowercase:***
   This can be easily achieved by *.low()*.
2. ***Remove punctuations:***
   The line text = text.translate(str.maketrans(' ', ' ', string.punctuation)) is used to remove punctuation from the text. The first two arguments of maketrans() specify that there are no characters to replace, and the third argument, string.punctuation, is defined from the string module that contains all common punctuation characters. The.translate() method scans through the text and removes all punctuation characters.
3. ***Remove stop words based on the list of English stop words in NLTK:***
   If the word is not a stop word, then we will add that word in a for loop and combine all of the non-stop word words.

   Lastly, we apply the clean_text() function to the text data, and create a new column, "item_1". This prepares the data ready for late keyword analysis and word embedding.



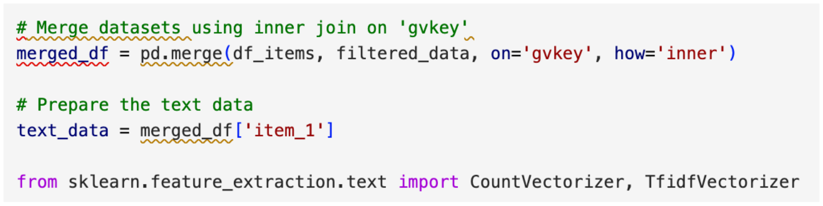
```python
import string
from nltk.corpus import stopwords

# Download NLTK stopwords
import nltk
nltk.download('stopwords')

# Get the list of English stop words
stop_words = set(stopwords.words('english'))

# Function to clean the text
def clean_text(text):
    # Convert to lowercase
    text = text.lower()
    # Remove punctuation
    text = text.translate(str.maketrans('', '', string.punctuation))
    # Remove stop words
    text = ' '.join([word for word in text.split() if word not in stop_words])
    return text

# Apply the cleaning function to the 'item_1' column and create a new column
df_items['item_1'] = df_items['item_1_text'].apply(clean_text)
```

D. Keyword Analysis

In this section, we perform a keyword analysis on the cleaned text data from our dataset, aiming to identify the most significant words based on both raw word counts and TF-IDF scores.

From previous analysis, we created the dataframe *filter_data*, which only contains firms from the Oil and Gas industry. Then we merge the *df_items,* which is the dataframe for "2020_10K_item1_full.csv", with *filter_data* using inner join on '*gckey*'. The merged dataframe is named as *merged_df.* Since the *item_1* column is the clean data that we processed from the previous question, we select column *item_1* from *merged_df* , renamed as *text_data,* for keyword analysis.

Then we import *CountVectorizer, TfidfVectorizer* from the *sklearn* to set up for analysis.



```python
# Merge datasets using inner join on 'gvkey'
merged_df = pd.merge(df_items, filtered_data, on='gvkey', how='inner')

# Prepare the text data
text_data = merged_df['item_1']

from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
```

Method 1: Word Counts

For method 1 word count, we use *Count Vectorization* to calculate the frequency of words in the cleaned text data. The *fit_transform* method on *text_data*, which converts the collection of documents into a sparse matrix where each row represents a document and each column corresponds to a unique word. This matrix reflects the frequency of each word in the respective documents. We then convert the sparse matrix into an array, create a dataframe, named *count_df* to sum the occurrences of each word across all documents, and identify the top 10 most frequently occurring words by sorting in descending order.

```python
# Method 1: Word counts
count_vectorizer = CountVectorizer()
count_matrix = count_vectorizer.fit_transform(text_data)
count_array = count_matrix.toarray()

# Create a DataFrame for word counts
count_df = pd.DataFrame(count_array, columns=count_vectorizer.get_feature_names_out())
top_words_count = count_df.sum().sort_values(ascending=False)
```

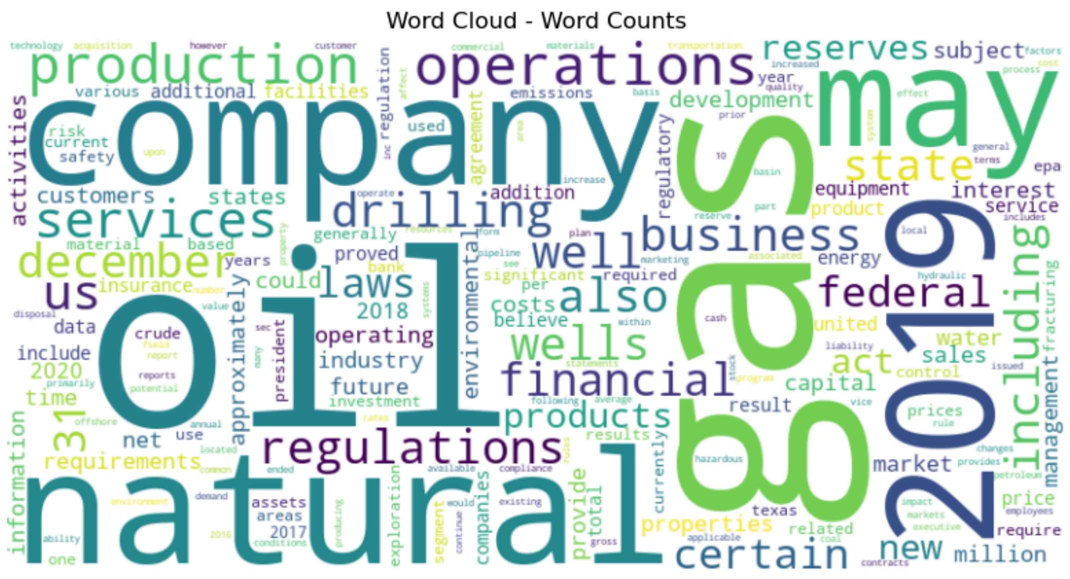
Method 2: TF-IDF Scores

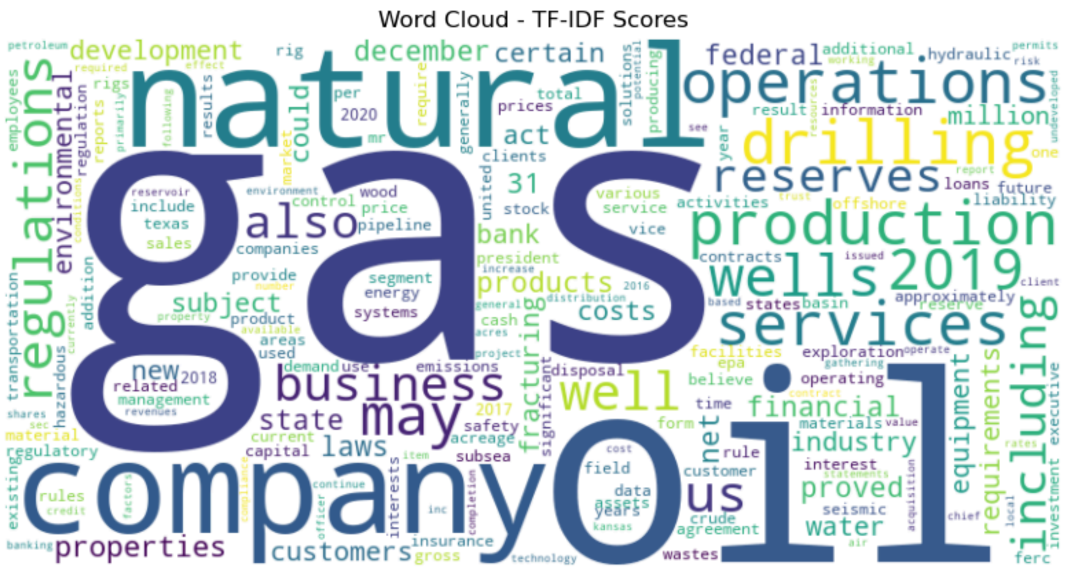
For method 2, we compute the TF-IDF scores for the words. This metric helps to determine the importance of each word in relation to the entire dataset, emphasizing terms that are significant in the context of the documents. The steps are similar to the previous methods.

```python
# Method 2:TF-IDF scores
tfidf_vectorizer = TfidfVectorizer()
tfidf_matrix = tfidf_vectorizer.fit_transform(text_data)
tfidf_array = tfidf_matrix.toarray()

# Create a DataFrame for TF-IDF
tfidf_df = pd.DataFrame(tfidf_array, columns=tfidf_vectorizer.get_feature_names_out())
top_words_tfidf = tfidf_df.mean().sort_values(ascending=False)
```

Create Word Clouds

To visualize the results of our analysis, we create word clouds for both the word counts and the TF-IDF scores methods. The two methods produce similar results with few difference, which is show as below:





E. Word Embedding

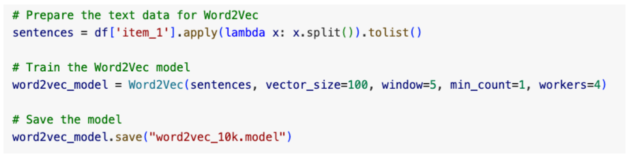
In this section, we train a Word2Vec model using textual data and analyze word similarities within the dataset. The goal is to uncover relationships between industry-related terms based on their semantic context in the text.

1. *Word2Vec Model Training*

We began by importing necessary libraries such as gensim for Word2Vec, pandas for data handling, and nltk for natural language processing. The dataset used for training is the cleaned text from a 10-K report. Dataset Preparation: The dataset was first loaded using pandas. If the text was not previously cleaned, we applied a function clean_text() that:

- Converted the text to lowercase.
- Removed punctuation using Python's str.translate() method.
- Filtered out stop words from the text.

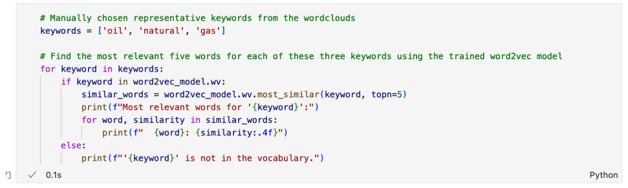
After preparing the text data, we split the cleaned text into sentences, which formed the basis for training the Word2Vec model. Lastly, we saved the Word2Vec model for future analysis.



```python
# Prepare the text data for Word2Vec
sentences = df['item_1'].apply(lambda x: x.split()).tolist()

# Train the Word2Vec model
word2vec_model = Word2Vec(sentences, vector_size=100, window=5, min_count=1, workers=4)

# Save the model
word2vec_model.save("word2vec_10k.model")
```

2. *Word Similarity Analysis*

We chose three representative keywords of the Oil and Gas Extraction Industry from the word clouds generated earlier: “Oil”, “Gas”, and “Natural”, Using the trained Word2Vec model, we calculated the five most similar words for each keyword.



```python
# Manually chosen representative keywords from the wordclouds
keywords = ['oil', 'natural', 'gas']

# Find the most relevant five words for each of these three keywords using the trained word2vec model
for keyword in keywords:
    if keyword in word2vec_model.wv:
        similar_words = word2vec_model.wv.most_similar(keyword, topn=5)
        print(f"Most relevant words for '{keyword}':")
        for word, similarity in similar_words:
            print(f"  {word}: {similarity:.4f}")
    else:
        print(f"'{keyword}' is not in the vocabulary.")
```
✓ 0.1s Python

Output:

- **Oil**: oil resource: 0.6771, stearin: 0.6744, activities oil: 0.6684, natural: 0.6626, non oil: 0.6553
- **Gas**: Basrah: 0.7278, power July: 0.6783, oil: 0.6626, flared: 0.6618, Kleberg: 0.6609
- **Natural**: gas fueled: 0.7618, gas related: 0.7349, gas producing: 0.7172, gas powered: 0.7084, gas fired: 0.6921

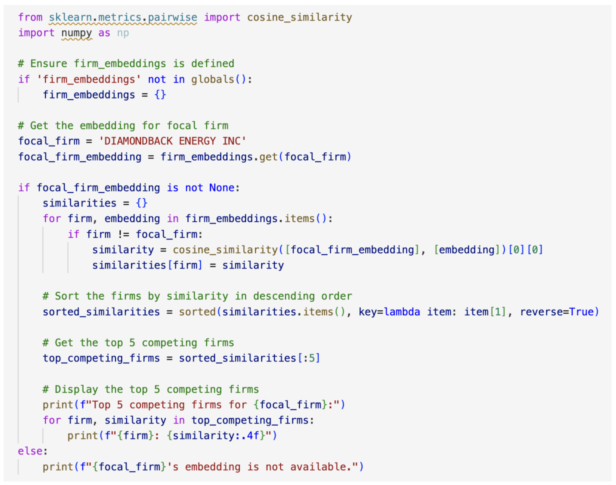
Insights: "Oil" appears as a relevant word for "nature", which indicates that "Nature" and "Oil" are two highly relevant topics. It is also interesting to see that "non oil" appears as a highly relevant word for oil. This indicates that the industry in under the energy transition trend to shift to more environmentally friendly energy sources. Additionally, there are also geographic locations (i.e. Basrah, Kleberg), and energy processing words (i.e. flared, power july) appeared as relevant words, which indicates the oil and gas industry is highly influenced by seasonality and geographic location.

PART 3. COMPREHENSIVE ANALYSIS OF ONE SAMPLE FIRM

Our focal company that we pick is Diamondback Energy Inc. It is an independent oil and natural gas company headquartered in Midland, Texas.

Analysis Process

For our firm analysis, our goal is to compare similar companies with our focal company, and look into their historical stock price, ROA, revenue, and assets to investigate opportunities for potential growth.



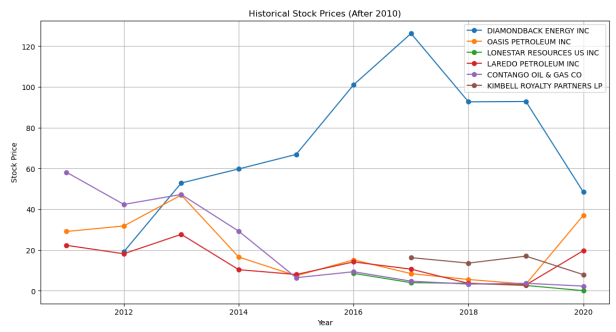
```python
from sklearn.metrics.pairwise import cosine_similarity
import numpy as np

# Ensure firm_embeddings is defined
if 'firm_embeddings' not in globals():
    firm_embeddings = {}

# Get the embedding for focal firm
focal_firm = 'DIAMONDBACK ENERGY INC'
focal_firm_embedding = firm_embeddings.get(focal_firm)

if focal_firm_embedding is not None:
    similarities = {}
    for firm, embedding in firm_embeddings.items():
        if firm != focal_firm:
            similarity = cosine_similarity([focal_firm_embedding], [embedding])[0][0]
            similarities[firm] = similarity

    # Sort the firms by similarity in descending order
    sorted_similarities = sorted(similarities.items(), key=lambda item: item[1], reverse=True)

    # Get the top 5 competing firms
    top_competing_firms = sorted_similarities[:5]

    # Display the top 5 competing firms
    print(f"Top 5 competing firms for {focal_firm}:")
    for firm, similarity in top_competing_firms:
        print(f"{firm}: {similarity:.4f}")
else:
    print(f"{focal_firm}'s embedding is not available.")
```

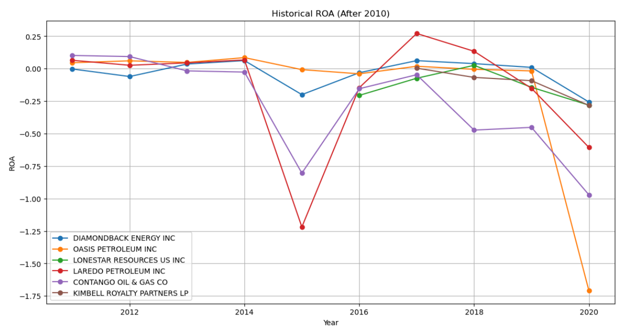
As a first step, we use the previous trained word2vec model to get word embedding. Then we sum up the embedding for each firm's document to create firm-level embedding. For the second steps, we need to compare the similarity of firm-level embedding to find our comparable companies. We achieve this by importing *cosine_similarity* from *sklearn.metrics.pairwise.* This part of the code we rely on copilot and applied new packages. Details and citations can be found in the Jupyter notebook.

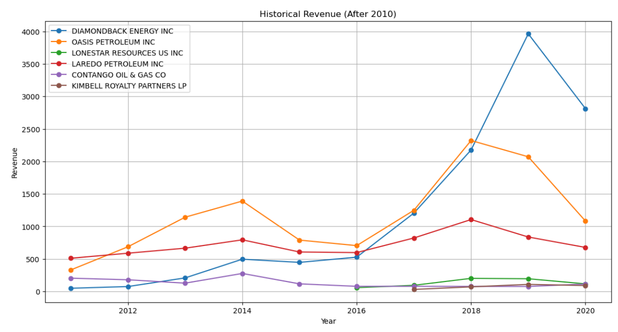
The five competing firms (with similarity scores) are:

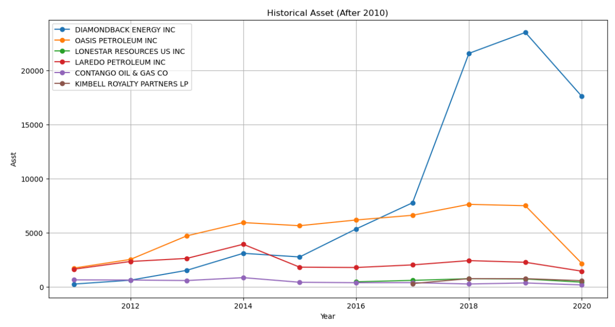
1. OASIS PETROLEUM INC: 0.9928
2. LONESTAR RESOURCES US INC: 0.9879
3. LAREDO PETROLEUM INC: 0.9866
4. CONTANGO OIL & GAS CO: 0.9843
5. KIMBELL ROYALTY PARTNERS LP: 0.9818

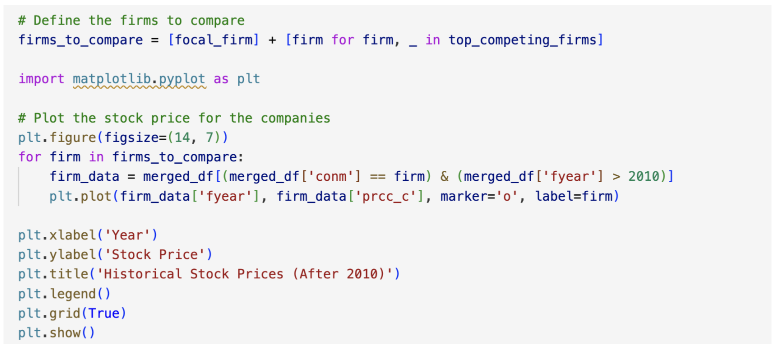
The third step is to compare the historical stock price, ROA, revenue, and assets for our focal company and its competitors. We plot the charts using *matplotlib*.











```
# Define the firms to compare
firms_to_compare = [focal_firm] + [firm for firm, _ in top_competing_firms]

import matplotlib.pyplot as plt

# Plot the stock price for the companies
plt.figure(figsize=(14, 7))
for firm in firms_to_compare:
    firm_data = merged_df[(merged_df['conm'] == firm) & (merged_df['fyear'] > 2010)]
    plt.plot(firm_data['fyear'], firm_data['prcc_c'], marker='o', label=firm)

plt.xlabel('Year')
plt.ylabel('Stock Price')
plt.title('Historical Stock Prices (After 2010)')
plt.legend()
plt.grid(True)
plt.show()
```

Analysis on Diamondback: Acquisition lead to Expansion Post 2015 Oil Price Crash

From our previous industry analysis (Part 1 - B), we found that the Oil and Gas industry is highly sensitive to macroeconomic shocks. The 2014-2015 Oil Price Crash had a substantial impact on the stock prices and Return on Assets (ROA) of most companies in the sector. However, the negative effect on Diamondback was moderate compared to its peers, illustrated by the charts above.

Since 2016, when oil prices began recovering from the downturn caused by the crash, Diamondback experienced a period of significant expansion. This was primarily driven by its strategic acquisitions of

Energen and Ajax in 2018[2][3], which solidified Diamondback's position as a dominant player in the Permian Basin. The charts show a marked increase in revenue and assets following these acquisitions.

Acquisitions are a critical growth strategy for oil and gas companies, as they allow for efficient entry into specific geographic markets. However, sustaining the growth post-acquisition can be challenging. The charts also show a decline in Diamondback's stock price, revenue, and assets in 2020, partly due to the impact of COVID-19 on the global economy. While the pandemic-induced recession has since subsided, new opportunities have emerged with the energy transition.

Suggestion: Integrating renewable energy production into a portfolio.

As the global economy recovers, the energy sector is shifting toward cleaner, more sustainable sources. With Diamondback already holding significant land assets in the Permian Basin from its past acquisitions, there is an opportunity for the company to diversify its energy portfolio.

We recommend that Diamondback Energy consider renewable energy investments in solar and wind projects within the Permian Basin, an area already central to its oil and gas operations. This move would allow the company to maximize the use of its existing land and infrastructure, creating operational synergies by integrating renewable energy production into its portfolio.

Solar and wind power offer a significant opportunity for diversification, as both are abundant resources in Texas. This shift would align Diamondback with the growing demand for cleaner, renewable energy sources. By leveraging its expertise in energy production and infrastructure management, Diamondback can develop large-scale renewable energy projects that not only help reduce its carbon footprint but also create a stable and scalable revenue stream in a future increasingly dominated by low-carbon energy sources.This will allow Diamondback to establish long-term resilience, not only as an oil and gas producer but as a key player in the broader energy transition, positioning itself as a forward-thinking, sustainable energy leader in the industry.

[2] Diamondback Energy, Inc. (2018, August 14). Diamondback Energy, Inc. to acquire Energen Corporation in all-stock transaction. GlobeNewswire. https://ir.diamondbackenergy.com/news-releases/news-release-details/diamondback-energy-inc-acquire-energen-corporation-all-stock

[3] Diamondback Energy, Inc. (2018, August 8). Diamondback Energy, Inc. announces second quarter 2018 financial and operating results and announces accretive acquisition. GlobeNewswire. https://ir.diamondbackenergy.com/news-releases/news-release-details/diamondback-energy-inc-announces-second-quarter-2018-financial

Appendix

Abbreviations	**Description**
df_groups	Dataframe for “major_group.csv”
df_firms	Dataframe for “public_firms.csv”
filtered_data	Firms in the Oil and Gas Extraction Industry
df_items	Dataframe for “2020_10K_item1_full.csv”, contains the updated cleaned text.
top_10_stock_firms	Top 10 firms with the highest stock price in the year 2020
top_10_firms_sales	Top 10 firms with the highest sales in the entire history of the dataset
top_10_locations	Top 10 locations for Oil and Gas firms
average_stock_price	Average stock price for Oil and Gas firms each year
data_2007	Stock price for year 2007
data_2008	Stock price for year 2008
merged_data	Combined data for 2007 and 2008 stock price for each Oil and Gas firm
usa_firms	Oil and Gas firms which are located in US
average_roa_usa	Average return to asset for Oil and Gas firms in US
item_1	Clean text data from for all industry
merged_df	The merged dateframe of “2020_10K_item1_full.csv” and “public_firms.csv” that only constrained firms in the Oil and Gas Extraction industry.
text_data	Clean text data from for only Oil and Gas Extraction firms
count_df	Dataframe that sum the occurrence of words from Oil and Gas firms’ 10K report
top_words_count	Top 10 keywords for Oil and Gas firms using word count methods
tfidf_df	Dataframe that sum the TF-IDF scores of words from Oil and Gas firms’ 10K report
top_words_tfidf	Top 10 keywords for Oil and Gas firms using word TF-IDF methods
clean_text	Function that clean text data by lowering cases, removing punctuations, and taking out top words
word2vec_10k.mode	Word2vec model trained with the full 10-K sample
get_text_embedding	Function that get the embedding for a give text
firm_embeddings	The tuple that sum up with embedding for each firm’s 10K documents