All Data EDA

Description of data cleaning before this file:

1. Load and combine credit rating datasets to get unique ratings, company/tickers, and rating issue dates (unique key is credit rating, rating agency, and issuance date). Limit to S&P ratings from 2010-2016.
2. Load dataset of earnings call transcripts, dates, year + quarter of statement releases, companies, and sectors (unique key is company by earnings call date).
3. Load tabular financial statement datasets (unique key is company, year, quarter).
4. Use earnings call dates to transform dataset of credit ratings so there is one rating at each earnings call date. The key assumption is that a rating stays the same until a new rating is issued. Use leads to get rating at next earnings call date, date of next earnings call, rating 2 earnings call dates ahead, and date of that earnings call.
5. Inner join earnings call data with credit rating data. Use year and quarter from earnings calls to inner join with financial statement data. This produces the `all_data` dataset.

Setup - Sample Path and Packages

In [ ]:
```
# Flag for if you are running this on the sample dataset
# Sample comprises 100 earnings calls (transcripts included)
# Full data comprises 4532 earnings calls (transcripts included)
sample = False
# Modify this path as needed to run on your machine
sample_path = r'all_data_sample.csv'
```

In [ ]:
```
# Packages
import pandas as pd
import matplotlib.pyplot as plt
```

Code

In [ ]:
```
# Load in sample csv, or full parquet file
# Use inputted sample path, or ~\Box\STAT 222 Capstone\Intermediate Data\al
if sample:
    df = pd.read_csv(sample_path)
else:
    df = pd.read_parquet(r'all_data.parquet')
df
```

Out[ ]:

	ticker	earnings_call_date	next_earnings_call_date	rating_on_next_earnings_call_date
0	ABBV	2014-07-25	2014-10-31	A
1	ABBV	2014-10-31	2015-01-30	A
2	ABBV	2015-01-30	2015-04-23	A
3	ABBV	2015-04-23	2015-07-24	A
4	ABBV	2015-07-24	2015-10-30	A
...	...	...	...	...
4527	ZTS	2015-11-03	2016-02-16	BBE
4528	ZTS	2016-02-16	2016-05-04	BBE
4529	ZTS	2016-05-04	2016-08-03	BBE
4530	ZTS	2016-08-03	2016-11-02	BBE
4531	ZTS	2016-11-02	None	None

4532 rows × 158 columns

In [ ]:

```
## Because the many units in the financial documents are different(in the ur
# We try to deal with extrem values (caused by different units in webstrachi
def deal_with_invalid_numbers(x,lower_bound, upper_bound):
    if str(x).endswith("000.0") and (x < lower_bound or x > upper_bound):
        #Divide the value by 1000 and check if it becomes more reasonable
        return x / 1000
    else:
        return x

# Check invalid data for every quantitative attribute
```

```
for column in df.columns:
    if df[column].dtype == float:
        lower_bound = df[column].quantile(0.025)  #2.5% quantile
        upper_bound = df[column].quantile(0.975)  #97.5% quantile
        df[column] = df[column].apply(deal_with_invalid_numbers, args=(lowe
```

In [ ]:

```
# Summarize all numeric columns
# use describe method, transpose, and print all rows
# round to two decimal places, no scientific notation, commas for thousands
pd.options.display.float_format = '{:,.2f}'.format
# pandas setting to display all rows
pd.set_option('display.max_rows', None)
df.describe().T
```

Out[ ]:

	count	mean	std
days_until_next_earnings_call	4,213.00	93.27	24.64
Rating Rank AAA is 10	4,532.00	6.75	1.28
Change in Rating	3,587.00	0.02	0.53
Year	4,532.00	2,013.28	1.64
year	4,532.00	2,014.29	1.57
quarter	4,532.00	2.52	1.12
cik	4,532.00	709,311.50	549,506.65
calendarYear	4,532.00	2,014.29	1.57
period	4,532.00	2.52	1.12
cashAndCashEquivalents	4,532.00	985,253,582.30	1,451,800,272.86
shortTermInvestments	4,532.00	150,561,295.73	518,833,141.63
cashAndShortTermInvestments	4,532.00	1,190,587,028.28	1,969,108,522.52
netReceivables	4,532.00	1,451,577,435.85	1,958,588,578.39
inventory	4,532.00	1,165,226,967.14	1,671,491,564.07
otherCurrentAssets	4,532.00	403,768,403.34	719,646,206.07
totalCurrentAssets	4,532.00	4,524,714,275.22	6,110,803,188.70
propertyPlantEquipmentNet	4,532.00	4,059,484,008.24	5,946,923,865.33
goodwill	4,532.00	2,345,663,120.45	3,813,808,482.06
intangibleAssets	4,532.00	1,192,766,112.01	2,453,711,774.88
goodwillAndIntangibleAssets	4,532.00	3,624,906,666.81	6,109,024,442.62
longTermInvestments	4,532.00	347,778,885.95	1,104,399,562.57
taxAssets	4,532.00	364,391,146.31	892,191,702.39
otherNonCurrentAssets	4,532.00	495,857,791.11	904,694,502.51
totalNonCurrentAssets	4,532.00	9,832,802,186.74	13,528,700,382.83
otherAssets	4,532.00	37,279.29	644,947.87
totalAssets	4,532.00	14,769,298,164.73	19,655,411,852.55
accountPayables	4,532.00	1,113,651,226.01	1,857,295,908.91
shortTermDebt	4,532.00	444,021,590.96	843,842,257.41
taxPayables	4,532.00	58,994,399.10	157,188,844.45
deferredRevenue	4,532.00	259,315,848.01	539,629,980.01
otherCurrentLiabilities	4,532.00	987,846,918.75	1,647,432,119.36
totalCurrentLiabilities	4,532.00	3,017,282,001.70	4,294,465,779.33
longTermDebt	4,532.00	3,805,920,761.66	4,819,130,864.58
deferredRevenueNonCurrent	4,532.00	191,797,557.85	618,142,475.36
deferredTaxLiabilitiesNonCurrent	4,532.00	606,892,100.95	1,142,419,959.07
otherNonCurrentLiabilities	4,532.00	880,573,344.00	1,349,565,059.60
totalNonCurrentLiabilities	4,532.00	5,706,062,661.34	7,380,375,542.44

	count	mean	std
otherLiabilities	4,532.00	234,123.41	3,268,113.31
capitalLeaseObligations	4,532.00	8,314,283.92	183,413,475.48
totalLiabilities	4,532.00	8,948,292,950.39	11,402,104,528.25
preferredStock	4,532.00	5,638,677.71	27,537,832.78
commonStock	4,532.00	241,241,216.63	688,514,234.44
retainedEarnings	4,532.00	4,504,462,554.63	7,976,913,271.32
accumulatedOtherComprehensiveIncomeLoss	4,532.00	-473,535,983.34	973,796,914.19
othertotalStockholdersEquity	4,532.00	815,895,740.82	3,975,062,744.89
totalStockholdersEquity	4,532.00	5,481,339,822.95	8,379,406,279.69
totalEquity	4,532.00	5,504,107,742.93	8,412,309,763.15
totalLiabilitiesAndStockholdersEquity	4,532.00	14,759,523,463.12	19,653,580,040.11
minorityInterest	4,532.00	151,934,542.12	461,477,535.29
totalLiabilitiesAndTotalEquity	4,532.00	14,759,523,463.12	19,653,580,040.11
totalInvestments	4,532.00	678,587,687.26	2,097,536,898.01
totalDebt	4,532.00	4,282,681,415.25	5,552,348,712.97
netDebt	4,532.00	3,109,879,783.36	4,467,260,449.20
cik_cash_flow_statement	4,532.00	709,311.50	549,506.65
calendarYear_cash_flow_statement	4,532.00	2,014.29	1.57
period_cash_flow_statement	4,532.00	2.52	1.12
netIncome	4,532.00	213,486,295.48	376,727,531.33
depreciationAndAmortization	4,532.00	143,223,503.73	221,757,457.29
deferredIncomeTax	4,532.00	1,086,999.26	62,657,393.57
stockBasedCompensation	4,532.00	13,834,210.80	21,886,482.74
changeInWorkingCapital	4,532.00	-10,084,181.15	196,047,628.97
accountsReceivables	4,532.00	-15,934,685.00	108,935,378.92
inventory_cash_flow_statement	4,532.00	-9,366,691.67	81,943,172.27
accountsPayables	4,532.00	6,912,841.33	102,572,202.25
otherWorkingCapital	4,532.00	14,588,297.85	165,188,830.93
otherNonCashItems	4,532.00	14,381,260.79	129,913,145.30
netCashProvidedByOperatingActivities	4,532.00	393,588,098.32	624,765,236.00
investmentsInPropertyPlantAndEquipment	4,532.00	-176,454,100.65	293,524,458.28
acquisitionsNet	4,532.00	-30,545,301.43	124,394,268.18
purchasesOfInvestments	4,532.00	-77,753,259.41	331,617,521.48
salesMaturitiesOfInvestments	4,532.00	81,725,760.62	322,263,168.52
otherInvestingActivites	4,532.00	3,446,595.97	100,024,035.40
netCashUsedForInvestingActivites	4,532.00	-260,170,843.28	500,000,583.96
debtRepayment	4,532.00	-260,091,666.09	493,889,583.83

	count	mean	st
commonStockIssued	4,532.00	61,872,601.70	190,020,004.3
commonStockRepurchased	4,532.00	-89,550,419.10	218,921,046.5
dividendsPaid	4,532.00	-94,915,530.19	193,387,072.8
otherFinancingActivites	4,532.00	226,432,866.79	551,841,405.1
netCashUsedProvidedByFinancingActivities	4,532.00	-126,861,615.65	402,579,554.6
effectOfForexChangesOnCash	4,532.00	-3,771,817.30	17,291,314.2
netChangeInCash	4,532.00	4,173,421.97	320,938,625.6
cashAtEndOfPeriod	4,532.00	988,719,244.91	1,458,351,579.2
cashAtBeginningOfPeriod	4,532.00	979,151,201.11	1,438,463,844.2
operatingCashFlow	4,532.00	393,588,098.32	624,765,236.0
capitalExpenditure	4,532.00	-176,454,100.65	293,524,458.2
freeCashFlow	4,532.00	191,329,878.15	413,756,231.3
cik_income_statement	4,532.00	709,311.50	549,506.6
calendarYear_income_statement	4,532.00	2,014.29	1.5
period_income_statement	4,532.00	2.52	1.1
revenue	4,532.00	3,023,056,727.51	4,563,317,466.0
costOfRevenue	4,532.00	1,958,081,957.15	3,354,652,665.7
grossProfit	4,532.00	957,147,665.21	1,462,705,593.6
grossProfitRatio	4,532.00	0.35	0.3
researchAndDevelopmentExpenses	4,532.00	36,911,267.10	134,690,097.4
generalAndAdministrativeExpenses	4,532.00	185,544,028.85	338,125,442.8
sellingAndMarketingExpenses	4,532.00	46,446,664.18	171,813,138.4
sellingGeneralAndAdministrativeExpenses	4,532.00	401,956,329.19	619,916,421.1
otherExpenses	4,532.00	35,707,031.03	349,087,984.4
operatingExpenses	4,532.00	610,655,901.07	1,029,605,668.0
costAndExpenses	4,532.00	2,644,062,022.45	4,134,578,114.7
interestIncome	4,532.00	2,012,344.37	5,098,695.2
interestExpense	4,532.00	40,462,556.42	49,835,766.4
depreciationAndAmortization_income_statement	4,532.00	139,275,821.82	205,060,981.1
ebitda	4,532.00	484,215,183.16	707,473,572.8
ebitdaratio	4,532.00	0.18	0.7
operatingIncome	4,532.00	340,515,503.16	543,365,993.9
operatingIncomeRatio	4,532.00	0.09	0.5
totalOtherIncomeExpensesNet	4,532.00	-14,499,209.66	83,476,560.5
incomeBeforeTax	4,532.00	294,085,415.69	508,755,665.2
incomeBeforeTaxRatio	4,532.00	0.07	0.4
incomeTaxExpense	4,532.00	81,102,792.16	145,523,275.3

	count	mean	std
netIncome_income_statement	4,532.00	213,008,372.24	374,067,843.4
netIncomeRatio	4,532.00	0.05	0.5
eps	4,532.00	0.54	3.9
epsdiluted	4,532.00	0.55	3.8
weightedAverageShsOut	4,532.00	433,847,585.01	1,094,548,597.0
weightedAverageShsOutDil	4,532.00	387,954,384.79	953,474,982.9

Note: currently investigating issues with year variables being outside of the desired range of the data.

Also need to investigate financial statement variables in quadrillions, trillions, etc. and how to handle them.

In [ ]:
```
# Revert to default settings
pd.reset_option('display.float_format')
pd.reset_option('display.max_rows')
```

In [ ]:
```
# Check data is unique on ticker by earnings_call_date
df['ticker_earnings_date'] = df['ticker'] + '_' + df['earnings_call_date'].a
df['ticker_earnings_date'].value_counts().max()
```

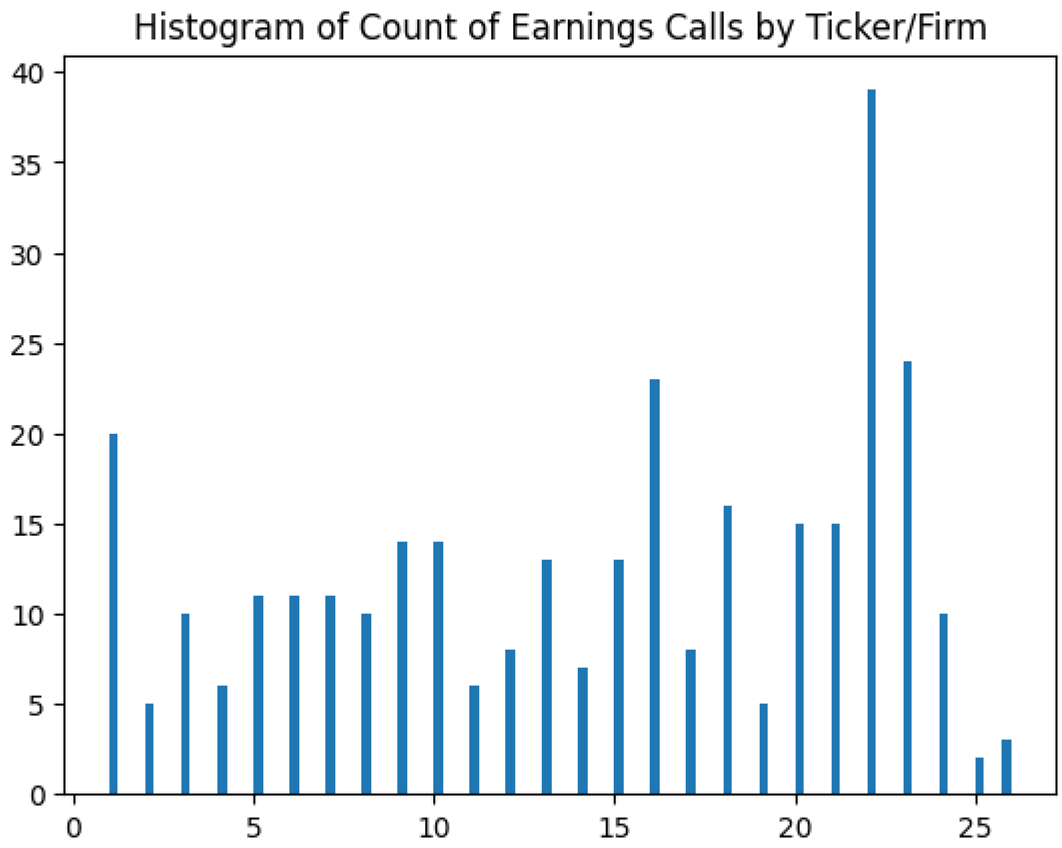
```
/var/folders/2v/664l7ccj4vn7kztqkgv6y7mh0000gn/T/ipykernel_92792/318712792
9.py:2: PerformanceWarning: DataFrame is highly fragmented.  This is usuall
y the result of calling `frame.insert` many times, which has poor performan
ce.  Consider joining all columns at once using pd.concat(axis=1) instead.
To get a de-fragmented frame, use `newframe = frame.copy()`
  df['ticker_earnings_date'] = df['ticker'] + '_' + df['earnings_call_dat
e'].astype(str)
```

Out[ ]: 1

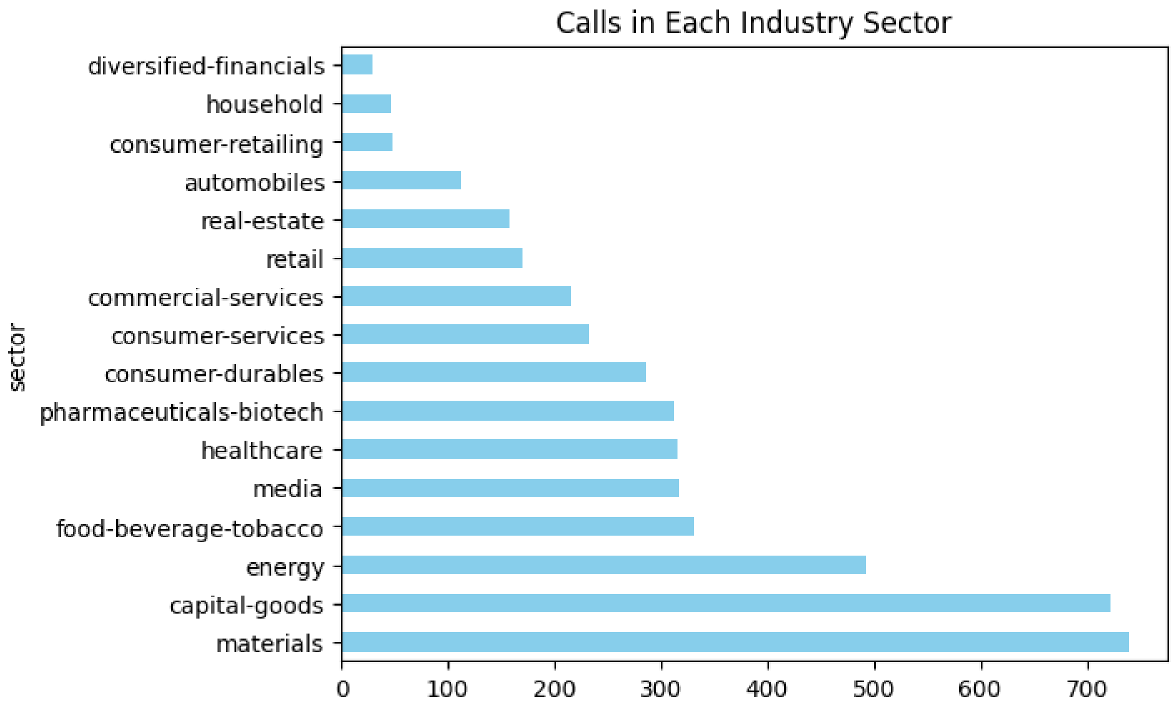
In [ ]:
```
# Number of unique firms (identified by ticker)
df['ticker'].nunique()
```

Out[ ]: 319

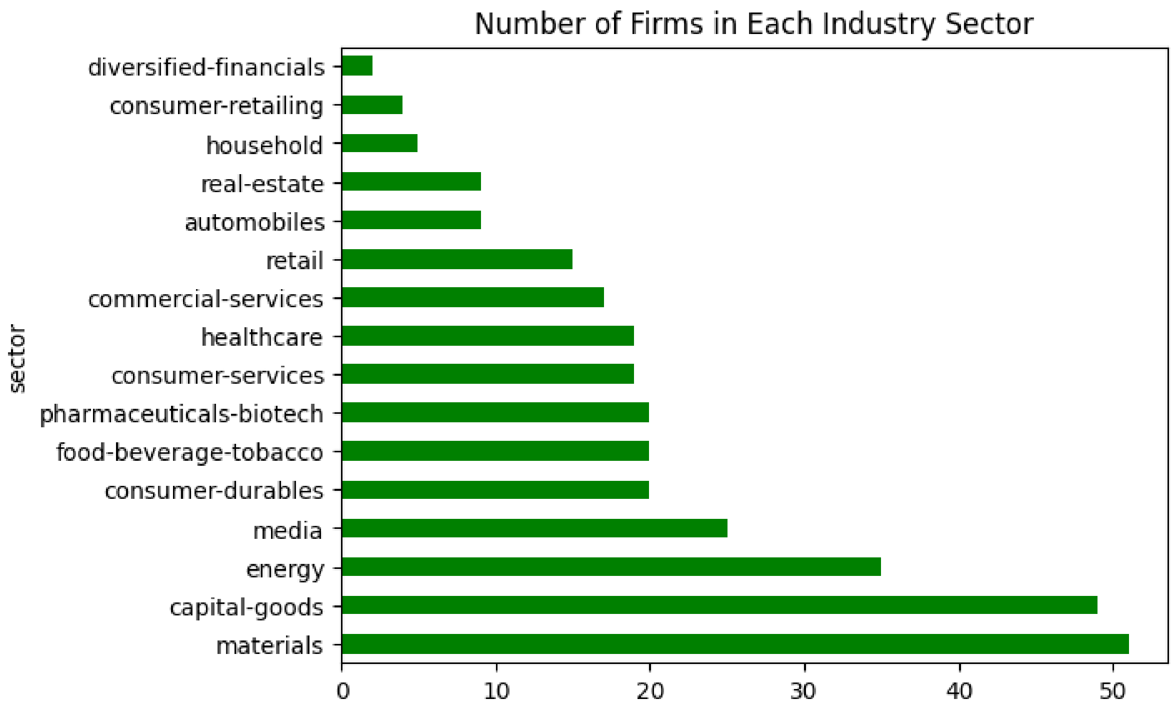
In [ ]:
```
# Histogram of count of earnings calls by ticker/firm
# Title: Histogram of Count of Earnings Calls by Ticker
plt.hist(df['ticker'].value_counts(), bins = 100)
plt.title('Histogram of Count of Earnings Calls by Ticker/Firm')
plt.show()
```



In [ ]:
```
# Distribution of industry sectors
# variable sector
# Title: Calls in Each Industry Sector
df['sector'].value_counts().plot(kind = 'barh', color = 'skyblue')
plt.title('Calls in Each Industry Sector')
plt.show()
```

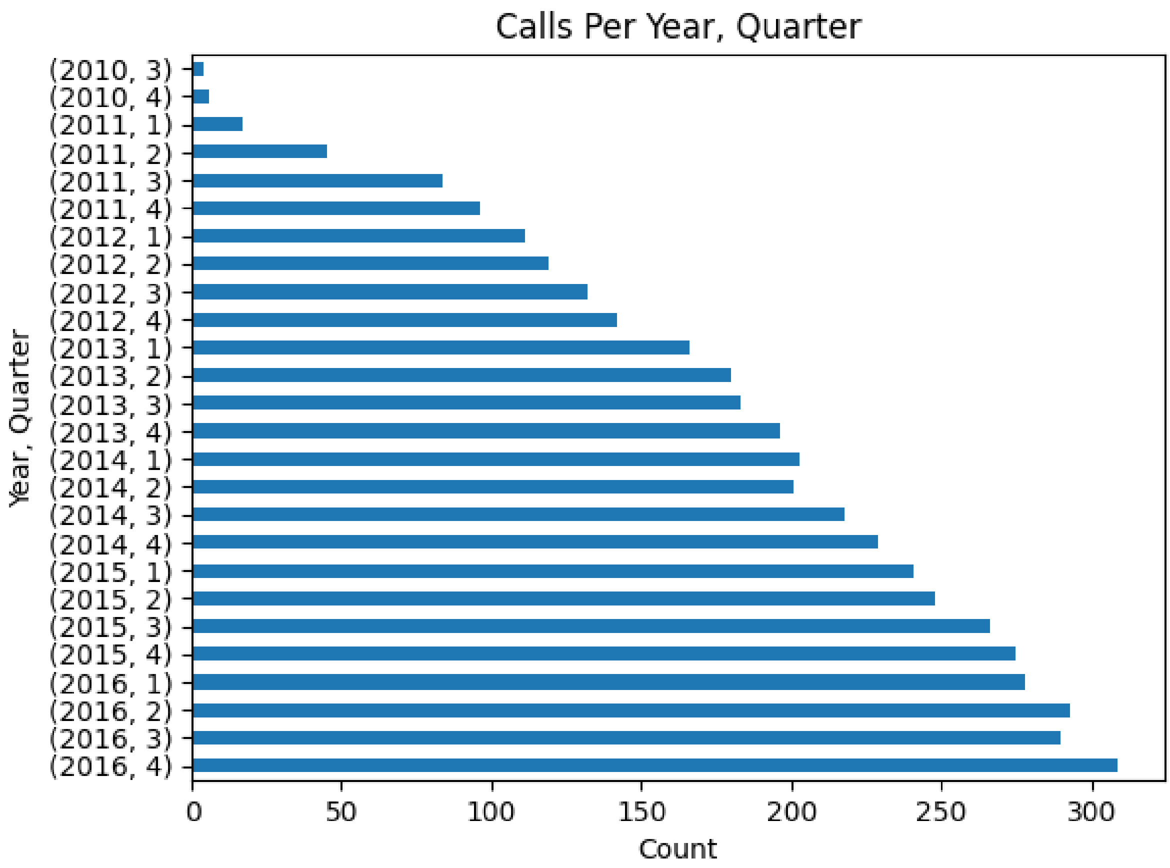


In [ ]:
```
# Number of firms in each industry sector
# Title: Number of Firms in Each Industry Sector
# Unique ticker by sector, sort by number of firms
df.groupby('sector')['ticker'].nunique().sort_values(ascending=False).plot(k
plt.title('Number of Firms in Each Industry Sector')
plt.show()
```



In [ ]:
```
# Distribution of earnings_call_date
# Create call_year and call_quarter columns after converting earnings_call_d
df['call_year'] = pd.to_datetime(df['earnings_call_date']).dt.year
df['call_quarter'] = pd.to_datetime(df['earnings_call_date']).dt.quarter
# Group by year and quarter
data_grouped = df.groupby([df.call_year, df.call_quarter]).size().sort_inde
# Plot horizontal bar chart
# 2010 at the top
data_grouped.plot(kind='barh')
plt.title('Calls Per Year, Quarter')
plt.xlabel('Count')
plt.ylabel('Year, Quarter')
plt.show()
```

```
/var/folders/2v/664l7ccj4vn7kztqkgv6y7mh0000gn/T/ipykernel_92792/102451380
9.py:3: PerformanceWarning: DataFrame is highly fragmented.  This is usuall
y the result of calling `frame.insert` many times, which has poor performan
ce.  Consider joining all columns at once using pd.concat(axis=1) instead.
To get a de-fragmented frame, use `newframe = frame.copy()`
  df['call_year'] = pd.to_datetime(df['earnings_call_date']).dt.year
/var/folders/2v/664l7ccj4vn7kztqkgv6y7mh0000gn/T/ipykernel_92792/102451380
9.py:4: PerformanceWarning: DataFrame is highly fragmented.  This is usuall
y the result of calling `frame.insert` many times, which has poor performan
ce.  Consider joining all columns at once using pd.concat(axis=1) instead.
To get a de-fragmented frame, use `newframe = frame.copy()`
  df['call_quarter'] = pd.to_datetime(df['earnings_call_date']).dt.quarter
```



In [ ]:

```
# Company dropout
# For each ticker, get the max value of earnings_call_date, and print out i
# Convert earnings_call_date to datetime
df['earnings_call_date_dt'] = pd.to_datetime(df['earnings_call_date'])
# Add column max_date to df
df['max_date'] = df.groupby('ticker')['earnings_call_date_dt'].transform('ma
# Display rows where max_date is not in the last quarter of 2016
df[df['max_date'] < '2016-10-01']
```

```
/var/folders/2v/664l7ccj4vn7kztqkgv6y7mh0000gn/T/ipykernel_92792/202515821
3.py:4: PerformanceWarning: DataFrame is highly fragmented.  This is usuall
y the result of calling `frame.insert` many times, which has poor performan
ce.  Consider joining all columns at once using pd.concat(axis=1) instead.
To get a de-fragmented frame, use `newframe = frame.copy()`
  df['earnings_call_date_dt'] = pd.to_datetime(df['earnings_call_date'])
/var/folders/2v/664l7ccj4vn7kztqkgv6y7mh0000gn/T/ipykernel_92792/202515821
3.py:6: PerformanceWarning: DataFrame is highly fragmented.  This is usuall
y the result of calling `frame.insert` many times, which has poor performan
ce.  Consider joining all columns at once using pd.concat(axis=1) instead.
To get a de-fragmented frame, use `newframe = frame.copy()`
  df['max_date'] = df.groupby('ticker')['earnings_call_date_dt'].transform
('max')
```

Out[ ]:

	ticker	earnings_call_date	next_earnings_call_date	rating_on_next_earnings_call_date
143	AMCR	2016-08-25	None	None
495	BTU	2011-10-25	2012-04-19	BE
496	BTU	2012-04-19	2012-07-24	BE
497	BTU	2012-07-24	2012-10-22	E
498	BTU	2012-10-22	2013-01-29	E
...	...	...	...	..
3887	STON	2013-08-07	2013-11-08	E
3888	STON	2013-11-08	2014-03-14	E
3889	STON	2014-03-14	2014-05-08	E
3890	STON	2014-05-08	2015-05-08	E
3891	STON	2015-05-08	None	None

177 rows × 163 columns

In [ ]:

```
# Unique firms where this is the case
df[df['max_date'] < '2016-10-01'][['ticker', 'max_date']].drop_duplicates()
```

Out[ ]:

	ticker	max_date
143	AMCR	2016-08-25
495	BTU	2016-02-11
1598	FTI	2016-04-30
2257	KBH	2016-09-21
2687	MKC	2016-09-30
2742	MOS	2016-08-02
3056	NUE	2016-07-22
3267	PEP	2016-09-29
3318	PKG	2016-07-21
3496	RFP	2016-08-06
3887	STON	2015-05-08

AMCR is amcor, should still exist but it's date is kind of close to the end of 2016

BTU, peabody energy, seems to have gone bankrupt April 13, 2016

FTI underwent a merger in 2016-2017

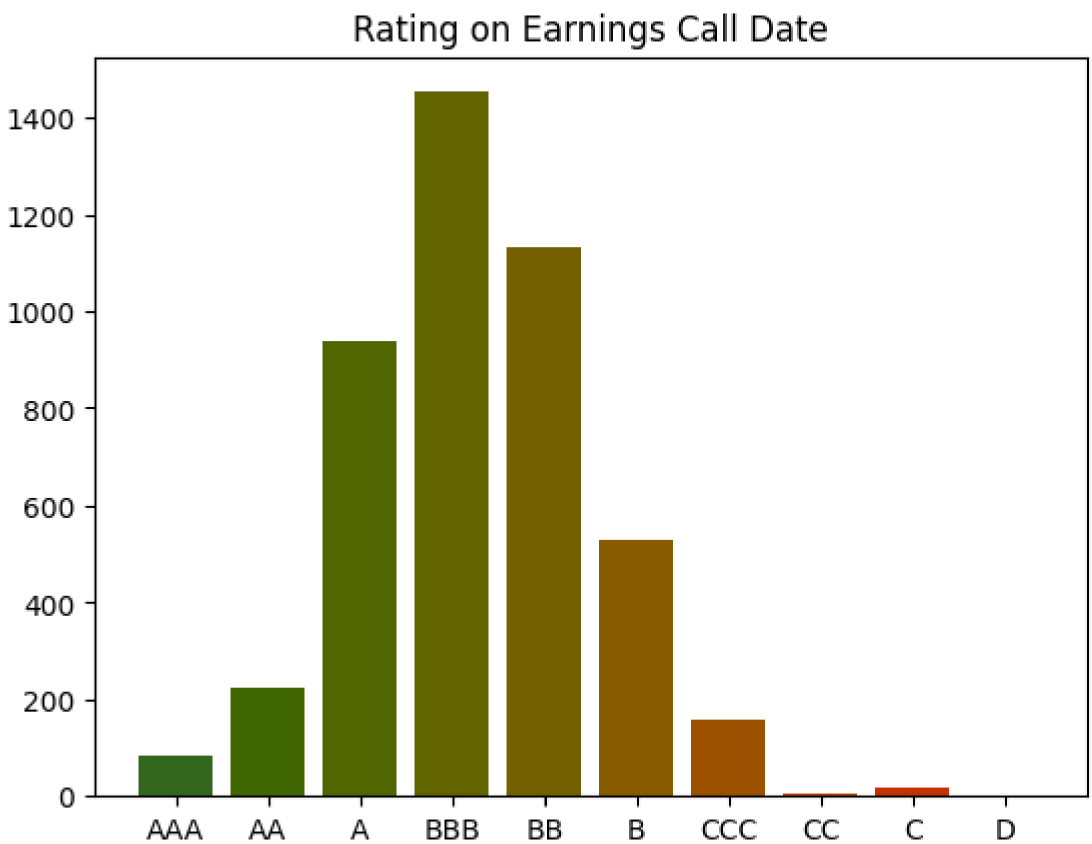
KBH still exists, but again the date is pretty close...

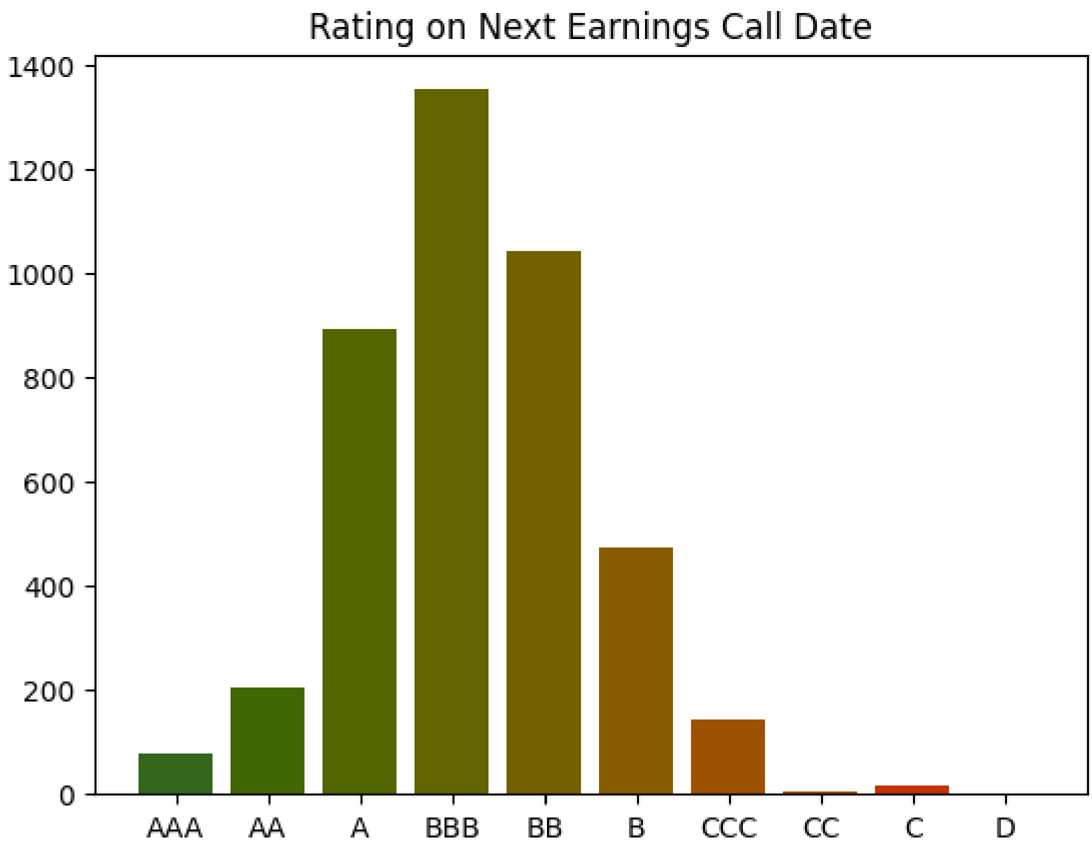
The other non-close one is STON. Notably, StonMor Partners had some issues with delayed SEC filings https://seekingalpha.com/article/4056108-prelude-to-bankruptcy-saving-grace-stonemor-partners-delays-10-k-again

In [ ]:

```
# Distribution of Rating and rating_on_next_earnings_call_date

# Colored with gradient and ordered

# Colors AAA through D
# Used https://colordesigner.io/gradient-generator#google_vignette
# Assign hex codes from green to red
#32671d
#416703
#516600
#626400
#756000
#885b00
#9c5200
#af4500
#c33200
#d60000
hex_code_mapper = {'AAA': '#32671d', 'AA': '#416703', 'A': '#516600', 'BBB'

# Ordering of bars - keys from hex_code_mapper
bar_order = ['AAA', 'AA', 'A', 'BBB', 'BB', 'B', 'CCC', 'CC', 'C', 'D']
# Assign values of Rating to this ordering
df['Rating'] = pd.Categorical(df['Rating'], categories=bar_order, ordered=Tr

# Create plot
# Save to "../Output/Distribution of Ratings.png"
plt.bar(df['Rating'].value_counts().sort_index().index, df['Rating'].value_c
plt.title('Rating on Earnings Call Date')
```

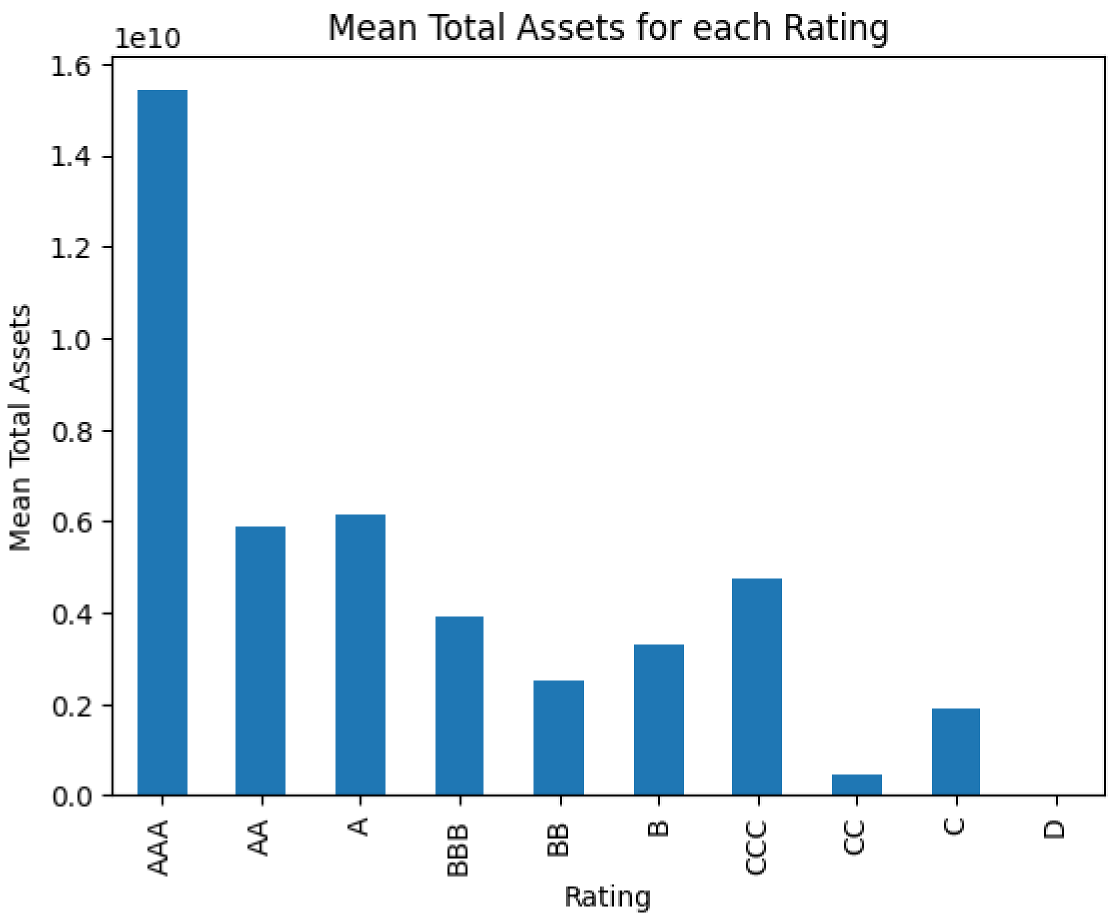
```
#plt.savefig('../../Output/Distribution of Rating Issuances.png')
plt.show()

# Rating on next earnings call date
df['rating_on_next_earnings_call_date'] = pd.Categorical(df['rating_on_next_
plt.bar(df['rating_on_next_earnings_call_date'].value_counts().sort_index()
plt.title('Rating on Next Earnings Call Date')
#plt.savefig('../../Output/Distribution of Rating on Next Earnings Call Date
plt.show()
```





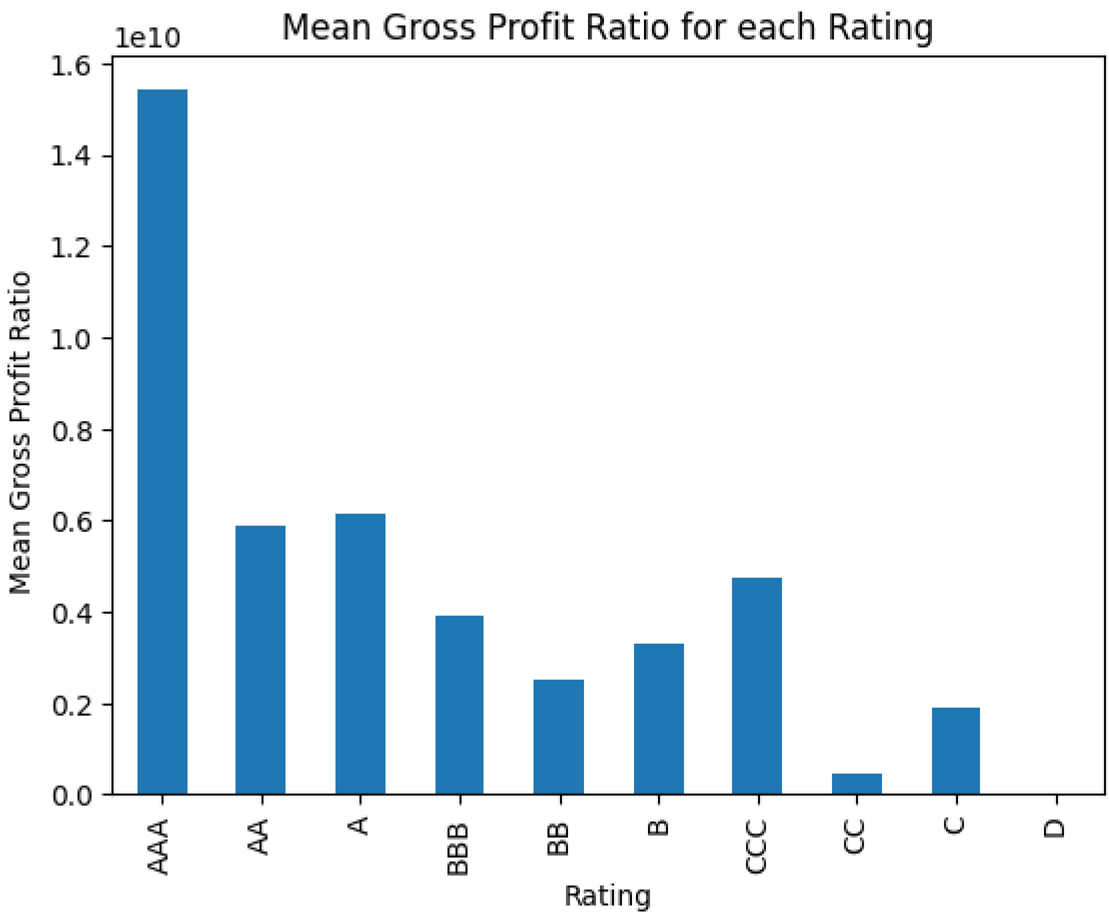
In [ ]:
```
# Calculate the mean of "totalAssets" for each kind of "Rating"
mean_assets_by_rating = df.groupby('Rating')['totalDebt'].mean()

# Plotting
mean_assets_by_rating.plot(kind='bar')
plt.title('Mean Total Assets for each Rating')
plt.xlabel('Rating')
plt.ylabel('Mean Total Assets')
plt.show()
```

```
/var/folders/2v/664l7ccj4vn7kztqkgv6y7mh0000gn/T/ipykernel_92792/311983169
4.py:2: FutureWarning: The default of observed=False is deprecated and will
be changed to True in a future version of pandas. Pass observed=False to re
tain current behavior or observed=True to adopt the future default and sile
nce this warning.
  mean_assets_by_rating = df.groupby('Rating')['totalDebt'].mean()
```



In [ ]:
```
# Calculate the mean of "grossProfitRatio" for each kind of "Rating"
mean_gross_profit_by_rating = df.groupby('Rating')["grossProfitRatio"].mean

# Plotting
mean_assets_by_rating.plot(kind='bar')
plt.title('Mean Gross Profit Ratio for each Rating')
plt.xlabel('Rating')
plt.ylabel('Mean Gross Profit Ratio ')
plt.show()
```

```
/var/folders/2v/664l7ccj4vn7kztqkgv6y7mh0000gn/T/ipykernel_92792/796684613.
py:2: FutureWarning: The default of observed=False is deprecated and will b
e changed to True in a future version of pandas. Pass observed=False to ret
ain current behavior or observed=True to adopt the future default and silen
ce this warning.
  mean_gross_profit_by_rating = df.groupby('Rating')["grossProfitRatio"].me
an()
```



We can see the relationship between rating and total assets; rating and gross profit ratio. High rating like AAA will have high mean total assets and high mean gross profit ratio.

NLP EDA

In [ ]:

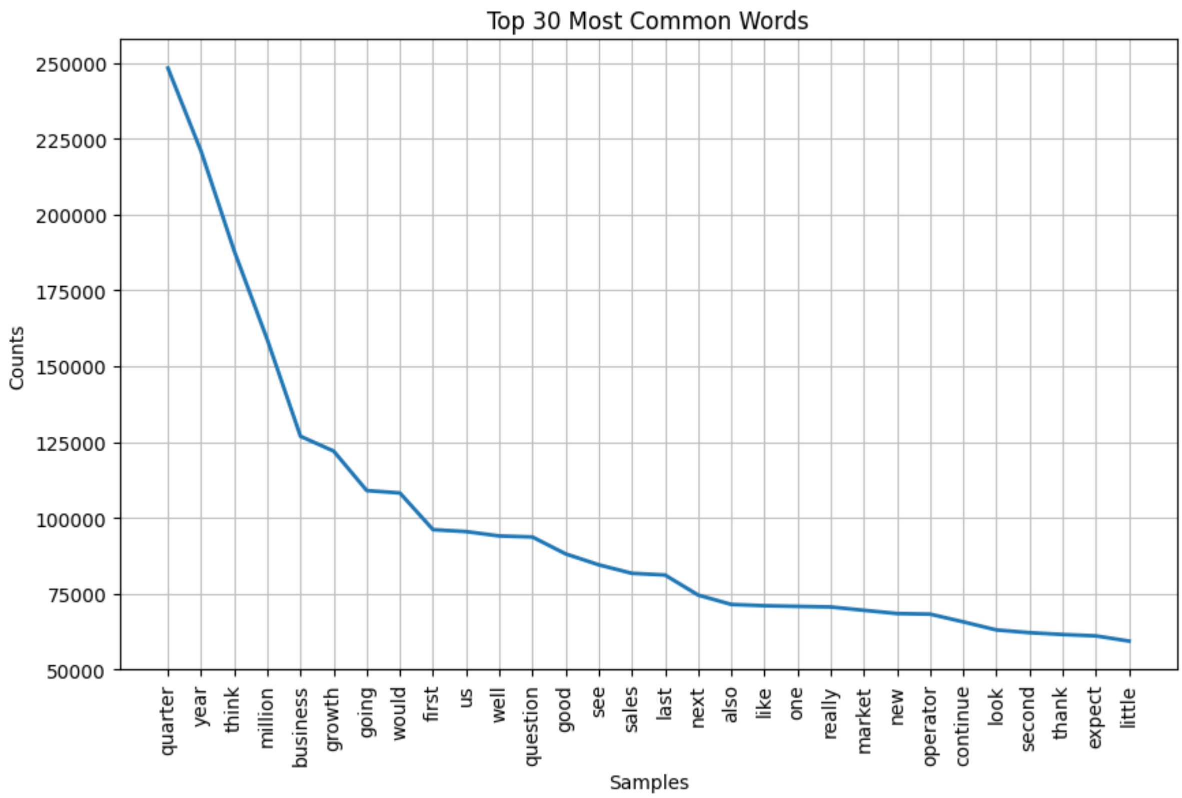
```
import pandas as pd
import matplotlib.pyplot as plt
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.probability import FreqDist


# Basic Statistics
num_records = len(df)
avg_length = df['transcript'].str.len().mean()

print(f"Number of records: {num_records}")
print(f"Average transcript length: {avg_length:.2f} characters")

# Tokenize the text
nltk.download('punkt')  # Download NLTK tokenizer data
tokens = df['transcript'].apply(word_tokenize)

# Remove stop words
nltk.download('stopwords')  # Download NLTK stop words data
stop_words = set(stopwords.words('english'))
tokens = tokens.apply(lambda tokens: [word for word in tokens if word.lower

# Word Frequency Analysis
all_words = [word.lower() for token_list in tokens for word in token_list]
```

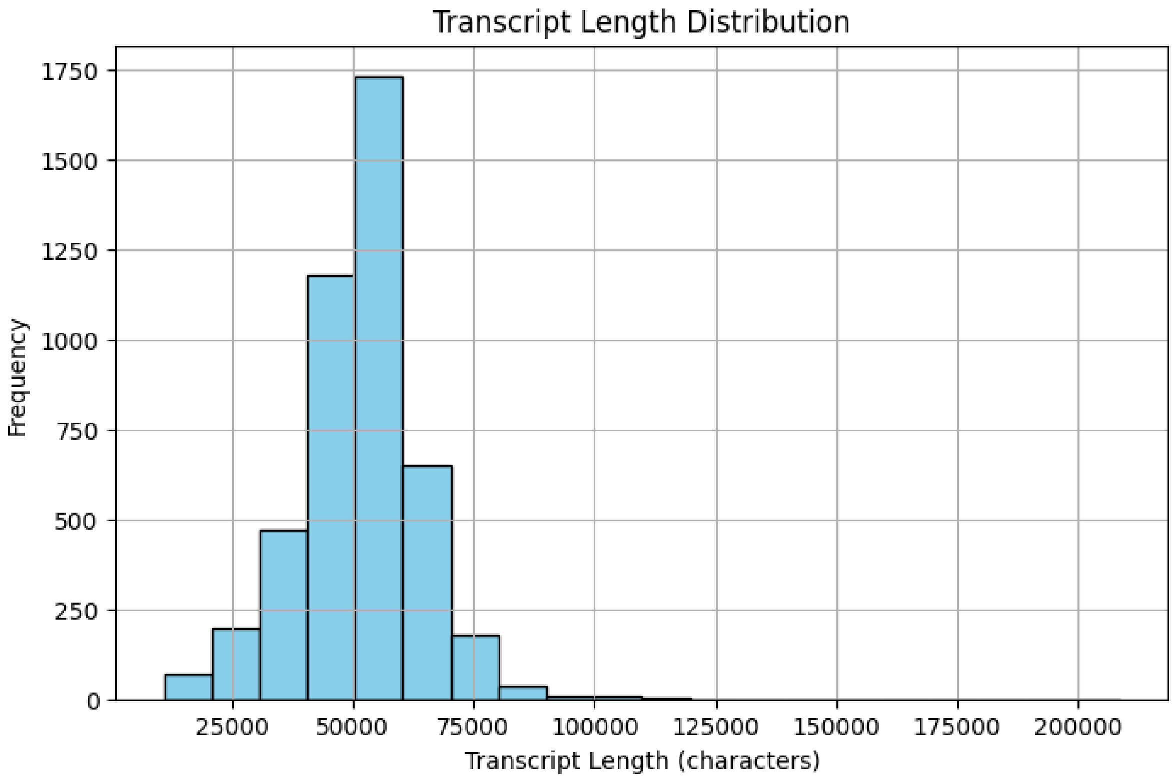
```
fdist = FreqDist(all_words)
top_words = fdist.most_common(10)
print("Top 10 most common words:")
for word, freq in top_words:
    print(f"{word}: {freq}")

# Plot Word Frequency Distribution
plt.figure(figsize=(10, 6))
fdist.plot(30, title='Top 30 Most Common Words')

# Text Length Distribution
transcript_lengths = df['transcript'].str.len()
plt.figure(figsize=(8, 5))
plt.hist(transcript_lengths, bins=20, color='skyblue', edgecolor='black')
plt.title('Transcript Length Distribution')
plt.xlabel('Transcript Length (characters)')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```

```
Number of records: 4532
Average transcript length: 51412.94 characters
```

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

```
Top 10 most common words:
quarter: 248253
year: 220870
think: 187990
million: 158704
business: 126889
growth: 122016
going: 108995
would: 108204
first: 96107
us: 95476
```





In [ ]:
```
from wordcloud import WordCloud

# Generate word clouds for each credit rating
ratings = ["AAA","BBB","CCC"]
for rating in ratings:

    rating_df = df[df['Rating'] == rating]
    tokens = rating_df['transcript'].apply(word_tokenize)
    stop_words = set(stopwords.words('english'))
    tokens = tokens.apply(lambda tokens: [word for word in tokens if word.lo

    all_words = [word.lower() for token_list in tokens for word in token_lis
    fdist = FreqDist(all_words)

    # Generate Word Cloud
    wordcloud = WordCloud(width=800, height=400, background_color='white').g

    # Plot Word Cloud
    plt.figure(figsize=(10, 6))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis('off')
    plt.title(f'Word Cloud for Credit Rating {rating}')
    plt.show()
```

```
<FreqDist with 12575 samples and 454618 outcomes>
```

Word Cloud for Credit Rating AAA

<FreqDist with 37055 samples and 6342100 outcomes>

Word Cloud for Credit Rating BBB

<FreqDist with 14221 samples and 622346 outcomes>

Word Cloud for Credit Rating CCC

We can see from word cloud for credit rating AAA, postive words like "growth" are bigger.