IMPACT OF 10-Q FILINGS TONE ON STOCK RETURNS IN 2021

Ayomikun Adebawojo

ICMA Centre, Henley Business School
University of Reading

TABLE OF CONTENT

PART A: Introduction/Literature Review 1

PART B: Descriptive Statistics Summary.....2

PART C: Result of Regression Analysis 4

PART D: Conclusion.....8

References..... 10

Appendix..... 11

PART A: Introduction/Literature Review

Audited financial statements and 10-K Filings, which make up about 80% of annual reports for publicly traded companies, are disclosed in publicly traded firms' annual reports. Due to measurement and reporting norms mandated by Generally Accepted Accounting Principles (GAAP) and Generally Accepted Auditing Standards (GAAS), management's discretion in audited financial statements is constrained by principles like consistency, conservatism, and materiality. However, management is not constrained by these principles when presenting in the 10-K reports. The 10-K reports' subject matter and format are required, but management is free to decide the level of clarity of the content. The ambiguity of management's tone has a significant impact on how stakeholders interpret and make decisions based on this information and how they understand it.

Numerous behavioural finance research contends that market speculation can cause prices to deviate from their underlying values (e.g., De Long et al., 1990; Shefrin, 2008.) According to Loughran and McDonald (2011), businesses that utilise the phrase "uncertain" or weak modal verbs like "possible," "may," and "approximate" or "contingent" in their 10-K filings have higher stock return volatility in the following year.

The Securities and Exchange Commission (SEC) of the United States has rightly demanded narrative clarity and reduced ambiguity in 10-K reports and expressed worry that businesses may be purposefully vague to defend themselves against future accusations (SEC, 2007). In the management discussion and analysis, there is a positive correlation between forward-looking information and future earnings (Li, 2010). The ambiguity of tone has also been linked to higher subsequent ROA (Davis et al., 2012), shareholder litigation (Rogers et al., 2011), and stock return volatility (Kothari et al., 2009).

We, therefore, proceed under the assumption that tone ambiguity is as essential as (or even more significant than) the reports' readability, in part because the latter may be more difficult

to pick up on. We examine whether the textual analysis tone in 30 sample firms' 10K filings is associated with the stock returns of the firm.

PART B: Descriptive Statistics Summary

This paper selects a sample of 30 different stocks from the Standard&Poor's (S&P) 500, based on the index weighting which cuts across different sectors (Fig. 1).

Components of the S&P 500

#	Company	Symbol	Weight	Price	Chg	% Chg
1	Apple Inc.	AAPL	6.083397	134.91	-1.03	(-0.76%)
2	Microsoft Corporation	MSFT	5.36183	235.38	-4.97	(-2.07%)
3	Amazon.com Inc.	AMZN	2.551173	95.34	-0.71	(-0.74%)
4	Berkshire Hathaway Inc. Class B	BRK.B	1.697686	308.20	-6.66	(-2.12%)
5	Alphabet Inc. Class A	GOOGL	1.631808	91.05	-0.24	(-0.26%)
6	Alphabet Inc. Class C	GOOG	1.46031	91.68	-0.48	(-0.52%)
7	Exxon Mobil Corporation	XOM	1.391803	110.59	-2.34	(-2.07%)
8	UnitedHealth Group Incorporated	UNH	1.356356	476.24	-8.84	(-1.82%)
9	Johnson & Johnson	JNJ	1.350054	169.76	-2.60	(-1.51%)
10	NVIDIA Corporation	NVDA	1.320547	173.63	-3.39	(-1.92%)
11	JPMorgan Chase & Co.	JPM	1.235947	136.55	-4.25	(-3.02%)
12	Visa Inc. Class A	V	1.091135	219.46	-3.54	(-1.59%)
13	Procter & Gamble Company	PG	1.066865	146.50	-3.94	(-2.62%)
14	Tesla Inc	TSLA	1.056191	128.70	-2.79	(-2.12%)
15	Home Depot Inc.	HD	1.002862	323.78	-3.75	(-1.14%)
16	Chevron Corporation	CVX	0.960874	177.18	-3.31	(-1.83%)
17	Mastercard Incorporated Class A	MA	0.95317	375.20	0.20	(0.05%)
18	Meta Platforms Inc. Class A	META	0.910896	132.92	-2.44	(-1.80%)
19	Eli Lilly and Company	LLY	0.844319	352.01	-5.73	(-1.60%)
20	Merck & Co. Inc.	MRK	0.838036	108.79	-1.66	(-1.50%)
21	AbbVie Inc.	ABBV	0.808838	149.20	-3.63	(-2.38%)
22	Pfizer Inc.	PFE	0.774079	44.98	-1.10	(-2.39%)
23	PepsiCo Inc.	PEP	0.725887	171.45	-4.61	(-2.62%)
24	Bank of America Corp	BAC	0.721023	33.71	-0.82	(-2.36%)
25	Coca-Cola Company	KO	0.718416	59.80	-1.88	(-3.05%)
26	Broadcom Inc.	AVGO	0.702066	574.00	-5.24	(-0.90%)
27	Thermo Fisher Scientific Inc.	TMO	0.689976	580.00	-7.86	(-1.34%)
28	Costco Wholesale Corporation	COST	0.644766	479.25	-7.53	(-1.55%)
29	Walmart Inc.	WMT	0.609958	140.81	-3.60	(-2.49%)
30	McDonald's Corporation	MCD	0.600663	266.25	-7.86	(-2.87%)

Figure 1: S&P 500 index weighting.
Source: https://www.slickcharts.com/sp500

Python programming was used in downloading all filings in 2021 from the U.S Securities and Exchange Commission Electronic Data Gathering, Analysis and Retrieval system (EDGAR) (appendix 1), creating a Structural Query Language (SQL) database to store all the filings in 2021 (appendix 2), downloaded from EDGAR with about 20,393 with an average of about 3 fillings in the year 2021 from each firm. We then queried the 10-Q fillings from the database on python (appendix 3) and downloaded the stock prices of the selected 30 sample firms after joining the tickers to the 10-Q filings to enable us to call the stock prices and merge the stock prices with the 10-Q filings (appendix 4 & 5). A textual analysis of the sampled firms' 10-Q filings was conducted using the Loughran McDonald dictionary (appendix 6), summary statistics were performed on the stock returns (appendix 7) and the figure below shows some graphs of the stock returns (figure 3), stock trading volume (figure 4) and stock trading characteristics on negative filing dates (figure 2). An econometric procedure to examine the relationship between stock returns and negative tone in the 10-Q filings was then done. From the analysis done, we realized there was no positive tone in the analysis which can be attributed to the fact that the economy was just recovering from covid, and this affected the tone of their filings negatively.

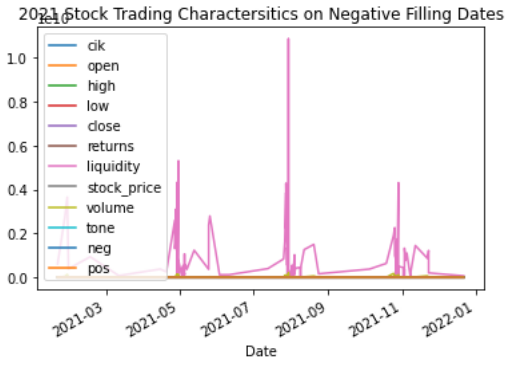
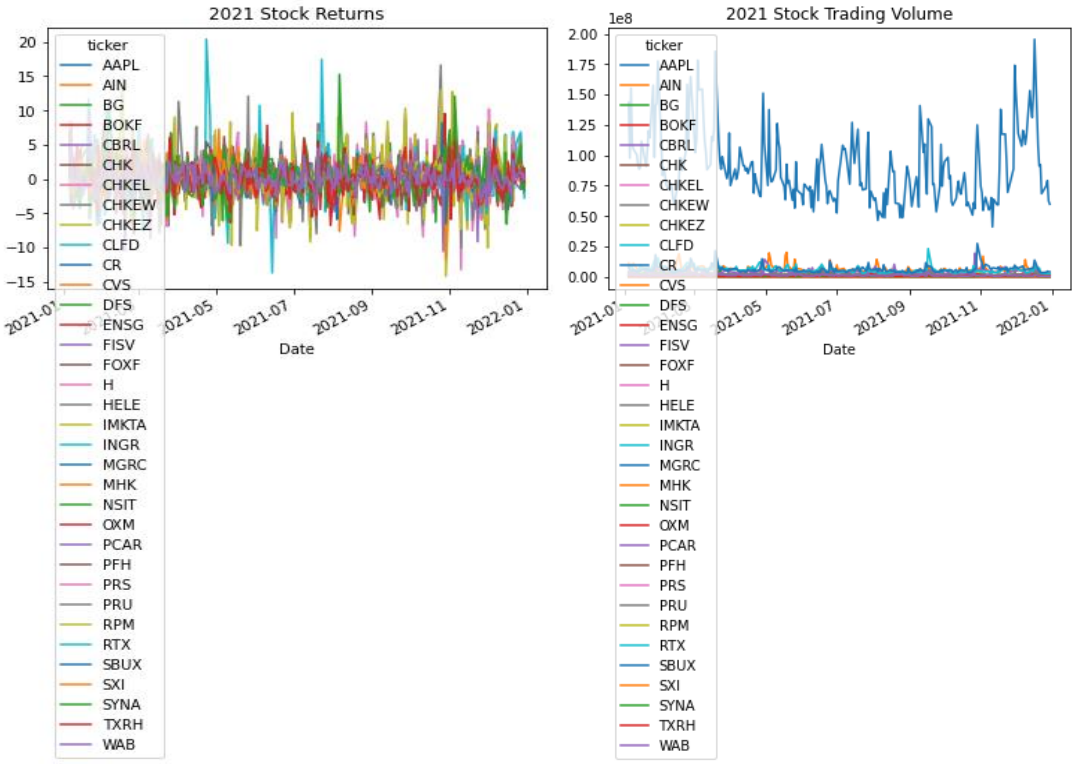


Figure 2: Stock Trading Characteristics on Negative Filing dates
Source: **Author Computation on Python.**



Figures 3 & 4: Stock returns and Stock Trading Volume
Source: **Author Computation on Python.**

PART C: Result of Regression Analysis

The table below is a summary of findings carried out on the stock prices of the top 30 selected samples from the S&P 500 according to their market capitalization for the period January 1st 2021 to December 31st 2021, aimed at finding the impact of textual analysis tones on the returns of the selected samples.

Our independent variables were the market risk premium (Rm-Rf) using the S&P 500 stock prices as the market rate, the risk-free rate (Rf) using the US treasury bill yields and the dummy negative tones textual analysis (tones), performed using python (appendix 6). For the dummy variables, 1 was used to represent dates the firms filed with a negative tone while 0 was used to represent days there was no filing and the returns of all the variables were taken to ensure uniformity. The dependent variables and independent variables were regressed using the Ordinary Least Square (OLS) method.

OLS Regression Results

Dep. Variable:	Returns	R-squared:	0.160
Model:	OLS	Adj. R-squared:	0.160
Method:	Least Squares	F-statistic:	796.1
Date:	Thu, 19 Jan 2023	Prob (F-statistic):	0.00
Time:	18:52:57	Log-Likelihood:	-20051.
No. Observations:	12556	AIC:	4.011e+04
Df Residuals:	12556	BIC:	4.0144e+04
Df Model:	3		
Covariance Type:	nonrobust		

	Coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.0229	0.011	2.117	0.034	0.002	0.044
rm_rf	0.6377	0.013	48.835	0.000	0.612	0.663
rf	0.6719	0.103	6.498	0.000	0.469	0.875
tone	0.0582	0.099	0.589	0.556	-0.135	0.252

Omnibus:	2848.778	Durbin-Watson:	2.033
Prob(Omnibus):	0.000	Jarque-Bera (JB):	89627.204
Skew:	0.409	Prob(JB):	0.00
Kurtosis:	16.063	Cond. No.	9.86

Table 1: Regression Analysis for Textual Analysis (Negative tones)
Source: Author Computation on Python.

Hypothesis Testing

The hypothesis for the theory is as follows:

H_0: Negative tones did not impact the stock price return for sample data.

H_1: Negative tones impacted the stock price returns for sample data.

Model:

$Y = f(X_1, X_2, X_3)$

Returns = f((Rm-Rf), Rf, Tones)

Econometric form:

$Returns = \beta_0 + \beta_1(Rm - Rf) + \beta_2 Rf + \beta_3 Tones + \mu$

Where:

Returns: Stock Price return of selected samples

Rm: Market price (S&P 500 stock price)

Rf: Risk-free rate (US treasury bill yields)

Rm - Rf: Market risk premium

Tones: Negative dummy tones

β_0–β_6: Parameters to be estimated

μ: Stochastic Variable or Error term.

A PRIORI EXPECTATION

The multiple regression model's a priori expression is $\beta_1=\beta_2=\beta_3=0$. A positive sign is anticipated from the coefficient of the association between returns and (Rm – Rf) and Rf, while a negative sign is anticipated from the coefficient of the association between returns and negative tones.

MODEL INTERPRETATION

Returns =	0.0229 +	0.6377(Rm-Rf) +	0.6719Rf +	0.0582Tone + μ
Std. Error	(0.011)	(0.013)	(0.103)	(0.099)
t-statistics	(2.117)	(48.835)	(6.498)	(0.589)
p-value	(0.034)	(0.000)	(0.000)	(0.556)

The output of the regression analysis shows that the 30-sample stock return sustained some level of growth at 0.0229% even at a zero level of influence from the independent variables (market risk premium, risk-free rate and negative tones).

The coefficient β_1 (0.6377) shows a positive relationship between returns and market risk premium. Hence, holding the influence of every other relevant variable constant, a 1% increase

in market risk premium might bring about a 0.64% increase in the returns while a 1% decrease in market risk premium might cause a 0.64% decrease in returns. This explains the fact that if there is an increase in S&P 500 stock, the 30 selected sample firms might also record positive returns. With a p-value of 0.00 < 0.05, it shows that the variable is significant which is in line with the apriori expectation.

The coefficient β_2 (0.6719) shows a positive relationship between returns and the risk-free rate. Hence, holding the influence of every other relevant variable constant, a 1% increase in risk-free rate (US treasury bill yields) might bring about a 0.67% increase in the returns while a 1% decrease in risk-free rate (US treasury bill yields) might bring about a 0.67% decrease in returns. This explains the fact that if there is an increase in US treasury bill yields, the 30 selected sample firms might also record positive returns. With a p-value of 0.00 < 0.05, it shows that the variable is significant which is in line with the apriori expectation.

The coefficient β_3 (0.0582) shows a positive relationship between returns and textual analysis of negative tones. Hence, holding the influence of every other relevant variable constant, a 1% increase in negative tones might bring about a 0.06% increase in the returns while a 1% decrease in negative tones might bring about a 0.06% decrease in returns. This explains that the more negative our tone, the higher our returns. With a p-value of 0.556 > 0.05, it shows that the variable is insignificant which is not in line with the apriori expectation. While this is against our apriori expectation, the stock market bounced back in 2021 and investors' confidence increased meaning that they started to pay less attention to the negative filing tone and the impact that negative filing had on the returns was just temporary and short-lived. Investors' confidence in the market as a result of the ease of lockdown rules, the return of many businesses to normal business activities and the development of vaccines helped make the impact of the negative filing have a little and temporary impact on returns.

The coefficient of determination, R^2, is used to determine the degree of variation in the dependent variable that is explained by the independent variables. From the regression results the R^2 is 0.16. This implies that approximately 16% of the total variation in returns is explained by variations in market risk premium (Rm-Rf), risk-free rate (Rf) and negative tones (tones). The remaining 84% of unexplained variations could be attributed to the stochastic variable (μ) which includes other variables that are not included explicitly in the model. Therefore, it can be concluded that the variables stated above are not good and unreliable in determining the factors that affect the selected stock returns.

PART D: Conclusion

Managers frequently use corporate narrative disclosure to communicate future company prospects to market participants and assess historical performance. According to earlier research such as Miller (2010), Lawrence (2013) and Tan et al. (2015), more complex financial reporting affects investor judgement and raises market participant disagreement.

This study examines the effect of textual analysis tones on the stock returns of 30 selected sample firms in the year 2021. After accounting for some risk indicators, we discover that companies with annual reports that are difficult to read seem to have lower equity financing costs. These results do not support the hypothesis that negative tones in 10-Q filings are likely to affect the stock returns of firms in the S&P 500. Due to Covid in the year 2020, the results indicate that the stock market bounced back in 2021 and investors' confidence might have increased, meaning that they started to pay less attention to the negative filing tone and the impact that negative filing had on the returns was just temporary and short-lived. Investors' confidence in the market as a result of the ease of lockdown rules, the return of many businesses to normal business activities and the development of vaccines helped make the impact of the negative filing have a little and temporary impact on returns.

Our focus on the linguistic characteristics of firm disclosures (10-Q filings) adds new data to the well-established field of research on the factors that affect the returns of firms but not much was gotten to advance our knowledge of how non-quantitative disclosure affects stock price returns.

References

Davis, A.K., Tama-Sweet, I., (2012). Managers' use of language across alternative disclosure outlets: earnings press releases versus MD & A. *Contemp. Account. Res*. 29 (3), 804 837.

De Long, J.B., Shleifer, A., Summers, L.H., Waldmann, R., (1990). Noise trader risk in financial markets. *J. Polit. Econ*. 98, 703–738.

Kothari, S.P., Li, X., Short, J.E., (2009). The effect of disclosures by management, analysts, and business press on cost of capital, return volatility, and analyst forecasts: a study using content analysis. *Account. Rev*. 84 (5), 1639–1670.

Lawrence, A., (2013). Individual investors and financial disclosure. *J. Account. Econ*. 56, 130 147.

Li, F., (2010). The information content of forward-looking statements in corporate filings—a naive Bayesian machine learning approach. *J. Account. Res*. 48 (5), 1049–1102.

Loughran, T., McDonald, B., (2011). When is a Liability not a Liability? Textual analysis, dictionaries, and 10–Ks. *J. Finance* 66 (1), 35–65.

Miller, B.P., (2010). The effects of reporting complexity on small and large investor trading. *Account. Rev*. 85, 2107–2143.

Rogers, J.L., Van Buskirk, A., Zechman, S., (2011). Disclosure tone and shareholder litigation. *Account. Rev*. 86 (6), 2155–2183.

Shefrin, H., (2008). A Behavioral Approach to Asset Pricing, second ed *Elsevier Academic Press, New York.*

Tan, H.-T., Wang, E.Y., Zhou, B., (2015). How does readability influence Investors' judgments? Consistency of benchmark performance matters. *Account. Rev*. 90, 371 393.

Appendix

Appendix 1:
########## QUESTION 1 ##########

```
import pandas as pd
import datetime as dt
import io
import requests
import time

base_url = "https://www.sec.gov/Archives/edgar/full-index/2021/"
filings_name = "/master.idx"
number_of_quarters = 4
filings_start_pos = 11

HEADER = {'Host': 'www.sec.gov', 'Connection': 'close',
        'Accept': 'application/json, text/javascript, */*; q=0.01', 'X-Requested-With':
'XMLHttpRequest',
        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/80.0.3987.163 Safari/537.36',
        }
NO_HEADER = {}

def downloader(url, type=None, f_name=None, f_log=None, number_of_tries=5,
sleep_time=5, header_type=HEADER):
    # type = 1:to file, 2: to string, 3: to list
    print(".....Downloading started for {}......".format(url))
    for i in range(1, number_of_tries):
        try:
            response = requests.get(url, headers=header_type)
            if response.status_code == 200:
                if type == 1:
                    with open(f_name, 'wb') as f:
                        f.write(response.content)
                        print("=====Downloaded completely for {}=====\n".format(url))
                    return True
                elif type == 2:
                    doc = response.content.decode(encoding="UTF-8", errors='ignore' )
                    print("=====Download complete for {}=====\n".format(url))
                    return doc
                elif type == 3:
                    file_list = io.StringIO(response.content.decode(encoding="UTF-8",
errors='ignore' )).readlines()
                    print("=====Downloaded completely for {}=====\n".format(url))

                    return file_list

            else:
```

```
            print(f'  Error in try #{i} downloader : URL = {url} | status_code =
{response.status_code}')
            if i == number_of_tries + 1:
                print(f'  Failed download: URL = {url}')
                if f_log: f_log.write(f'  Failed download: URL = {url}\n')

        except Exception as exc:
            if i == 1:
                print('\n==>response error in downloader')
            print(f'  {i}.url  : {url} \n  exc:  {exc}')
            if '404' in str(exc):
                break
            print(f'     Retry in {sleep_time} seconds')
            time.sleep(sleep_time)
            sleep_time += sleep_time

    print('\n  ERROR:  Download failed for')
    print(f'          url:  {url}')

    if f_log:
        f_log.write('\nERROR:  Download failed=>')
        f_log.write(f'  _url: {url}')
        f_log.write(f'  |  {dt.datetime.now().strftime("%c")}')

    return False

def download_to_list(url, f_log=None, number_of_tries=5, sleep_time=5,
header_type=HEADER):
    # Downloading url content to create list of lines
    file_list = downloader(url, type=3, f_log=f_log, number_of_tries=number_of_tries,
sleep_time=sleep_time, header_type=header_type)
    return file_list

def get_master_index_filings_by_download():
    # Downloading and converting Q1, Q2, Q3, Q4 filings into a dataframe
    df_masterindex = pd.DataFrame()
    print("Downloading Master Index Files for 2021")
    for i in range(1,number_of_quarters+1):
        i_str = str(i)
        url_end_point = base_url+"QTR"+i_str+filings_name
        master_index_list = download_to_list(url_end_point) #
        master_index_list_filing_record = master_index_list[filings_start_pos:]
        df = pd.DataFrame(master_index_list_filing_record)
        df[['cik','company_name','form_type','date_filed','file_name']] =
df[0].str.split('|',expand=True)
        df.drop(0, axis=1, inplace=True)
        df_masterindex = df_masterindex.append(df)

    return df_masterindex
```

```
# Generate master index filings
df_masterindex = get_master_index_filings_by_download()
```

Appendix 2:
########## **QUESTION 2** ##########

```
import sqlite3
from sqlite3 import Error

resources_files_path = ""
db_file = resources_files_path+"newdatabase.db"

def create_connection(db_file):
    conn = None
    try:
        conn = sqlite3.connect(db_file)
        return conn
    except Error as e:
        print(e)

    return conn

def create_table(conn):
    try:
        create_table_sql = """CREATE TABLE IF NOT EXISTS [fillings_2021] (
                            cik          VARCHAR (200),
                        company_name VARCHAR (2000),
                        form_type    VARCHAR (255),
                        date_filed   DATETIME,
                        file_name    VARCHAR (1000)
                    );
                    """
        c = conn.cursor()
        c.execute(create_table_sql)
        print("Table Created")
    except Error as e:
        print(e)

def insert_all_records_into_database(conn, data):
    try:
        data.to_sql("filings_2021",conn,if_exists='replace',index=False)
        conn.commit()
        print("Data Inserted Successfully ")
    except Error as e:
        print(e)

# Creating database connection
db_conn = create_connection(db_file)

# Creating a database table
```

```
create_table(db_conn)

# Loading data into table
insert_all_records_into_database(db_conn, df_masterindex)
```

Appendix 3:
########## QUESTION 3 ##########

```
qry_form_type="10-Q"

def select_filings_by_form_type(conn, form_type):

    cur = conn.cursor()
    rs = cur.execute("SELECT * FROM fillings_2021 where form_type="'+form_type+'"")
    cols = list(map(lambda x:x[0],rs.description))
    df = pd.DataFrame(rs.fetchall(),columns=cols)

    return df

# Selecting 10-Q filings from table
df_10q = select_filings_by_form_type(db_conn,qry_form_type)
df_10q['Date'] = pd.to_datetime(df_10q['date_filed'])
df_10q['cik'] = df_10q["cik"].map(int)
df_10q = df_10q.set_index(['cik'])
```

Appendix 4:
########## QUESTION 4 ##########

```
import yfinance as yf
import numpy as np

sample_cik = ['1800', '1551152', '1652044', '1018724', '320193', '70858', '1067983', '1730168',
'93410', '858877', '21344', '909832', '59478', '34088', '354950', '200406', '19617', '1141391',
'63908', '310158', '1326801', '789019', '1045810', '77476', '78003', '80424', '1318605',
'731766', '1403161', '104169']
all_tickers_url="https://www.sec.gov/include/ticker.txt"
stockprice_start_date=dt.datetime(2021,1,1)
stockprice_end_date=dt.datetime(2021,12,31)


def get_list_of_all_tickers():
    # Downloading list of all tickers and cik from edgar
    all_tickers_list = download_to_list(all_tickers_url)
    df_alltickers = pd.DataFrame(all_tickers_list)
    df_alltickers = df_alltickers.replace('\s+',';', regex=True)
    df_alltickers[['ticker','cik','ex']] = df_alltickers[0].str.split(';',expand=True)
    df_alltickers.drop([0,'ex'], axis=1, inplace=True)
    df_alltickers = df_alltickers.replace(r'\n','', regex=True)
    df_alltickers['cik'] = df_alltickers['cik'].str.strip()
    df_alltickers['cik'] = df_alltickers["cik"].map(int)
```

```
    df_alltickers['ticker'] = df_alltickers['ticker'].str.upper()
    df_alltickers = df_alltickers.set_index('cik')

    return df_alltickers

# Defining returns formula
def LogDiff(x):
    x_diff = 100*np.log(x/x.shift(1))
    x_diff = x_diff.dropna()
    return x_diff

# Defining liquidity formula
def set_liquidity(data):
    data['liquidity'] = (data['Volume']*data['Adj Close'])/(data['High']-data['Low'])

    return data

# Loading sample 30 companies listed in sample_cik into a dataframe
df_sample_cik = pd.DataFrame(sample_cik, columns=['cik'])
df_sample_cik['cik'] = df_sample_cik["cik"].map(int)

# Downloading list of tickers from edgar
df_alltickers = get_list_of_all_tickers()

# Joining sample company cik dataframe and tickers dataframe to get tickers of sample
companies
df_sample_cik_tickers = df_sample_cik.join(df_alltickers, on="cik")

# Downloading stock prices using ticker
df_stockprices = pd.DataFrame()
for index, row in df_sample_cik_tickers.iterrows():
    yf.pdr_override()
    sym = row['ticker']
    cik = row['cik']
    df_onesymstockprices = yf.download(sym,stockprice_start_date, stockprice_end_date)
    df_onesymstockprices['ticker'] = sym
    df_onesymstockprices['cik'] = cik
    df_onesymstockprices['returns'] = LogDiff(df_onesymstockprices['Adj Close'])
    df_stockprices = df_stockprices.append(df_onesymstockprices)

# Creating a new dataframe that contains only one record of each company's filing and drop
columns containing filling information
df_allcompanies = df_10q.drop_duplicates(subset='company_name', keep="first")
df_allcompanies.drop(['form_type','date_filed','file_name','Date'], axis=1, inplace=True)

# Creating a new dataframe by joining (using cik) df_stockprices dataframe and
df_allcompanies dataframe that consists of full company information and stock prices data
df_company_stockprices = df_stockprices.join(df_allcompanies, on=['cik'])
```

Appendix 5:

########## **QUESTION 5** ##########

```
# Joining (using cik and date filed) companystockprices dataframe with 10q filings dataframe to insert company filling information
df_10q = df_10q.set_index(['Date'],append=True)
df_company_stockprices_filings = df_company_stockprices.join(df_10q, on=['cik','Date'], lsuffix='', rsuffix='_right')
print(df_company_stockprices_filings.head(20))
print(df_company_stockprices_filings.tail(20))
```

Appendix 6:
########## **QUESTION 6** ##########

```
import sys
from bs4 import BeautifulSoup

loughran_dictionary_file_path="Loughran-McDonald_MasterDictionary_1993-2021_simplified_ver.csv"
filing_base_url="https://www.sec.gov/Archives/"

def load_lougran_dictionary():
    try:
        lexicon = pd.read_csv(loughran_dictionary_file_path)
        lexicon.head()
        mask = lexicon['negative'] == 1
        negatives = lexicon[mask]['Word']
        neg_words = negatives.tolist()
        mask = lexicon['positive'] ==1
        positives = lexicon[mask]['Word']
        pos_words = positives.tolist()

    except:
        print("Dictionary could not be loaded. Please put 'Loughran-McDonald_MasterDictionary_1993-2021_simplified_ver.csv' in file path")
        sys.exit("Please load dictionary in file path")
    return neg_words, pos_words

def get_filing_content(filing_path):
    filing_url_end_point = filing_base_url+filing_path
    filing_content = download_to_list(filing_url_end_point)
    filing_content = "".join(filing_content)

    return filing_content

def get_filing_content_into_words(filing_path):
    filing_content = get_filing_content(filing_path)
    words = []
    try:
        soup = BeautifulSoup(filing_content,"html.parser")
        content = soup.get_text()
```

```
        words = content.split()
    except:
        print("Unable to parse {}".format(filing_path))
    return words

def get_filing_tone(filing_words,negative_words,positive_words):
    neg_count = 0
    pos_count = 0

    for word in filing_words:
        word = word.upper()
        if word in negative_words:
            neg_count += 1
        if word in positive_words:
            pos_count += 1
    tone=0
    if pos_count+neg_count != 0:
        tone = (pos_count-neg_count)/(pos_count+neg_count)
    return tone,neg_count,pos_count

# Loading Lougran McDonald dictionary
negative_words, positive_words = load_lougran_dictionary()

# Dropping records of days where no fillings were done
df_company_stockprices_filings_rec = df_company_stockprices_filings.dropna()
tone_list = []
neg_list = []
pos_list = []
counter = 0

# Looping through the dataframe, retrieving the file_path from the dataframe and fetching the
filing content from edgar to conduct textual analysis
for index, row in df_company_stockprices_filings_rec.iterrows():
    filing_path = str(row['file_name'])
    filing_path = filing_path.strip()
    filing_words = get_filing_content_into_words(filing_path) # To get filing content and
convert to words
    filing_tone,negative_count,positive_count =
get_filing_tone(filing_words,negative_words,positive_words) # Generating the tone of the
filling
    tone_list.append(filing_tone)
    neg_list.append(negative_count)
    pos_list.append(positive_count)
    counter+=1

# Inserting tone, neg and pos results into dataframe df_company_stockprices_fillings_rec and
formatting dataframe
df_company_stockprices_filings_rec['tone'] = tone_list
df_company_stockprices_filings_rec['neg'] = neg_list
df_company_stockprices_filings_rec['pos'] = pos_list
```

```
df_company_stockprices_filings_rec.reset_index(inplace=True)
df_company_stockprices_filings_rec['cik'] =
df_company_stockprices_filings_rec["cik"].map(int)
df_company_stockprices_filings_rec['ticker'] =
df_company_stockprices_filings_rec["ticker"].map(str)
df_company_stockprices_filings_rec['Date'] =
pd.to_datetime(df_company_stockprices_filings_rec['Date'])
df_company_stockprices_filings_rec =
df_company_stockprices_filings_rec.set_index(['cik','ticker','Date'])

# Formatting df_company_stockprices dataframe
df_company_stockprices.reset_index(inplace=True)
df_company_stockprices['Date'] = pd.to_datetime(df_company_stockprices['Date'])
df_company_stockprices = df_company_stockprices.set_index(['Date'])

# Merging initial 'df_company_stockprices' dataframe with new dataframe
df_company_stockprices_fillings_rec
df_company_stock_filing_tone_raw =
df_company_stockprices.join(df_company_stockprices_filings_rec, on=['cik','ticker','Date'],
lsuffix='', rsuffix='_right')
df_company_stock_filing_tone_raw = set_liquidity(df_company_stock_filing_tone_raw)
df_company_stock_filing_tone_raw.reset_index(inplace=True)
df_company_stock_filing_tone = pd.DataFrame({
                'Date' : df_company_stock_filing_tone_raw['Date'],
                'cik' : df_company_stock_filing_tone_raw['cik'],
                'ticker' : df_company_stock_filing_tone_raw['ticker'],
                'company_name' : df_company_stock_filing_tone_raw['company_name'],
                'open' : df_company_stock_filing_tone_raw['Open'],
                'high' : df_company_stock_filing_tone_raw['High'],
                'low' : df_company_stock_filing_tone_raw['Low'],
                'close' : df_company_stock_filing_tone_raw['Close'],
                'returns' : df_company_stock_filing_tone_raw['returns'],
                'liquidity' : df_company_stock_filing_tone_raw['liquidity'],
                'stock_price' : df_company_stock_filing_tone_raw['Adj Close'],
                'volume' : df_company_stock_filing_tone_raw['Volume'],
                'form_type' : df_company_stock_filing_tone_raw['form_type'],
                'date_filed' : df_company_stock_filing_tone_raw['date_filed'],
                'file_name' : df_company_stock_filing_tone_raw['file_name'],
                'tone' : df_company_stock_filing_tone_raw['tone'],
                'neg' : df_company_stock_filing_tone_raw['neg'],
                'pos' : df_company_stock_filing_tone_raw['pos']})

df_company_stock_filing_tone['Date'] =
pd.to_datetime(df_company_stock_filing_tone['Date'])
df_company_stock_filing_tone = df_company_stock_filing_tone.set_index(['Date'])
```

Appendix 7:

########## **QUESTION 7** ##########

```
# All the information to do the summary statistics is in df_company_stock_filing_tone
Dataframe
df_company_stockprices_returns = df_company_stockprices.pivot(columns="ticker",
values="returns")
df_company_stockprices_volume = df_company_stockprices.pivot(columns="ticker",
values="Volume")
returnssummarystat = df_company_stockprices_returns.describe()
volumesummarystat = df_company_stockprices_volume.describe()
print(returnssummarystat)
print(volumesummarystat)
df_company_stock_filing_tone_negative =
df_company_stock_filing_tone.loc[df_company_stock_filing_tone['tone'] < 0]
df_company_stock_filing_tone_positive =
df_company_stock_filing_tone.loc[df_company_stock_filing_tone['tone'] > 0]
negativesummarystat = df_company_stock_filing_tone_negative.describe()
positivesummarystat = df_company_stock_filing_tone_positive.describe()
print("Returns Summary Statistics for Positive Filing Dates")
print(positivesummarystat)
print("Returns Summary Statistics for Negative Filing Dates")
print(negativesummarystat)

# Plotting Graph
df_company_stockprices_returns.plot(kind = 'line',title = '2021 Stock Returns')
df_company_stockprices_volume.plot(kind = 'line', title = '2021 Stock Trading Volume')
df_company_stock_filing_tone_negative.plot(kind = 'line',title = '2021 Stock Trading
Charactersitics on Negative Filling Dates')
df_company_stock_filing_tone_positive.plot(kind = 'line', title = '2021 Stock Trading
Charactersitics on Positive Filling Dates')
```

Appendix 8:

########## **NUMBER 8** ##########

```
import statsmodels.formula.api as smf

siglevel = 5/100
rm_rf_tickers = [{'cik':", 'ticker':"^GSPC"},{'cik':", 'ticker':"^IRX"}]

# Econometric procedure to show relationship between any of the stock
characteristics(Returns) and tone in 10Q filing
df_rm_rf_tickers = pd.DataFrame(rm_rf_tickers, columns=['ticker'])

# Downloading stock prices for rm and rf using ticker
df_rm_rf_stockprices=pd.DataFrame()
for index, row in df_rm_rf_tickers.iterrows():
    yf.pdr_override()
    sym = row['ticker']
    df_onesymstockprices = yf.download(sym,stockprice_start_date, stockprice_end_date)
    df_onesymstockprices['ticker'] = sym
```

```
    df_onesymstockprices['returns'] = LogDiff(df_onesymstockprices['Adj Close'])
    df_rm_rf_stockprices = df_rm_rf_stockprices.append(df_onesymstockprices)

df_rm_rf_stockprices_returns = df_rm_rf_stockprices.pivot(columns="ticker",
values="returns")
df_company_stock_filing_tone_rf_rm =
df_company_stock_filing_tone.join(df_rm_rf_stockprices_returns, on=['Date'], lsuffix='',
rsuffix='_right')
df_company_stock_filing_tone_rf_rm.loc[df_company_stock_filing_tone_rf_rm["tone"] < 0,
"tone_neg_dummy"] = 1
print("=====Starting Regression Analysis=====\n")
df_reg = pd.DataFrame()
ticker_result = []
intercept_result = []
intercept_conc = []
rf_result = []
rf_conc = []
tone_result = []
tone_conc = []
df_reg = pd.DataFrame({
            'returns' : df_company_stock_filing_tone_rf_rm['returns'],
            'rf' : df_company_stock_filing_tone_rf_rm['^IRX']/250,
            'rm' : df_company_stock_filing_tone_rf_rm['^GSPC'],
            'tone' : df_company_stock_filing_tone_rf_rm["tone_neg_dummy"]})
df_reg['rm_rf'] = df_reg['rm']-df_reg['rf']
df_reg.fillna(0,inplace=True)

formula = 'returns ~ rm_rf + rf + tone'
results = smf.ols(formula, df_reg).fit()
print(results.summary())
print("=====Hypothesis Tests=====\n")
hypothesis01 = "Intercept=0"
hyp_res_01 = results.t_test(hypothesis01)
coef01 = hyp_res_01.effect
pv01 = hyp_res_01.pvalue
tv01 = hyp_res_01.tvalue
conc01 = "Reject H0. Intercept is statistically significant."
if(pv01 > siglevel):
    conc01 = "Do not Reject H0. Intercept is not statistically significant"
it_result = "Value = {}, PValue = {}, TValue = {}".format(coef01,pv01,tv01)
intercept_result.append(it_result)
intercept_conc.append(conc01)
print("====Intercept====")
print(it_result)
hypothesis02 = "rf=1"
hyp_res_02 = results.t_test(hypothesis02)
coef02 = hyp_res_02.effect
pv02 = hyp_res_02.pvalue
tv02 = hyp_res_02.tvalue
conc02 = "Reject H0. rf is statistically significant."
```

```
if(pv02 > siglevel):
    conc02="Do not Reject H0. rf is not statistically significant"
r_result = "Value = {}, PValue = {}, TValue = {}".format(coef02,pv02,tv02)
rf_result.append(r_result)
rf_conc.append(conc02)
print("====RF====")
print(r_result)
hypothesis03 = "tone=1"
hyp_res_03 = results.t_test(hypothesis03)
coef03 = hyp_res_03.effect
pv03 = hyp_res_03.pvalue
tv03 = hyp_res_03.tvalue
conc03 = "Reject H0. tone is statistically significant."
if(pv03 > siglevel):
    conc03 = "Do not Reject H0. tone is not statistically significant"
t_result = "Value = {}, PValue = {}, TValue = {}".format(coef03,pv03,tv03)
tone_result.append(t_result)
tone_conc.append(conc03)
print("====TONE====")
print(t_result)
print("\n\n")
df_reg_result = pd.DataFrame(intercept_result, columns={"Intercept"})
df_reg_result['Intercept_Conclusion'] = intercept_conc
df_reg_result['Rf'] = rf_result
df_reg_result['Rf_Conclusion'] = rf_conc
df_reg_result['Tone'] = tone_result
df_reg_result['Tone_Conclusion'] = tone_conc

print(df_reg_result)
```