I. Text Extraction and Cleaning

Import libraries

In [1]:
```
import pandas as pd
import pdfkit
import PyPDF2
import re
from nltk.tokenize import word_tokenize
import spacy
from string import punctuation
from IPython.display import display_html
```

A) Creating the Dataset

Compile all URLs

In [2]:
```
base_url = 'http://www.sec.gov/Archives/edgar/data'
cik_ls = ['0000078003', '1682852','200406','59478','0001551152']
filings_num_ls = ['000007800322000027','000168285222000012','000020040622000022','000005947822000068',
                  '000155115222000007']
company_ls=["Pfizer Inc.", "Moderna, Inc.", "Johnson & Johnson", "Eli Lilly and Company", "AbbVie Inc."]
ticker_ls =['pfe','mrna','jnj','lly','abbv']
fye_ls = ['20211231','20211231','20220102','20211231','20211231']
```

In [3]:
```
url_ls=[]
for i in range(0,5):
    url= base_url+'/'+cik_ls[i]+'/'+filings_num_ls[i]+'/'+ticker_ls[i]+'-'+fye_ls[i]+'.htm'
    url_ls.append(url)
url_ls
```

Out[3]:
```
['http://www.sec.gov/Archives/edgar/data/0000078003/000007800322000027/pfe-20211231.htm',
 'http://www.sec.gov/Archives/edgar/data/1682852/000168285222000012/mrna-20211231.htm',
 'http://www.sec.gov/Archives/edgar/data/200406/000020040622000022/jnj-20220102.htm',
 'http://www.sec.gov/Archives/edgar/data/59478/000005947822000068/lly-20211231.htm',
 'http://www.sec.gov/Archives/edgar/data/0001551152/000155115222000007/abbv-20211231.htm']
```

Converting HTML to PDFs

In [4]:
```
for i in range(0,5):
    pdfkit.from_url(url_ls[i], ticker_ls[i]+'.pdf')
```

Retrieve Items 1A and 7 from all PDFs

Create lookup dictionaries for function parse_section_text:

In [5]:
```
toc_dict = {"pfe": 1, "mrna": 1, "jnj": 2, "lly": 1, "abbv": 2}
```

In [6]:
```
toc_start_dict = {"1A": "risk factors ",
                  "7": "management's discussion and analysis of financial condition and results of operations ",
                 "7alt":"management's discussion and analysis of results of operations and financial condition "}
```

In [7]:
```
toc_end_dict = {"1A": "properties ",
                "7": "quantitative and qualitative disclosures about market risk ",
                "7alt": "quantitative and qualitative disclosures about market risk "}
```

In [8]:
```
end_section_dict = {"1A": "item 2",
                    "7": "item 7a",
                   "7alt":"item 7a"}
```

Create function parse_section_text

In [9]:
```
def parse_section_text(ticker, section):
    with open(ticker+'.pdf','rb') as pdf_file:
    read_pdf = PyPDF2.PdfFileReader(pdf_file)
    toc_content = read_pdf.getPage(toc_dict[ticker]).extractText().replace("\t"," ").replace("\n"," ")
                  .lower().replace("'","’")

    start_page = int(toc_content[re.search(toc_start_dict[section], toc_content).end()
                                 -1:re.search(toc_start_dict[section], toc_content).end()+3].split()[0])
    end_page = int(toc_content[re.search(toc_end_dict[section], toc_content).end()
                                 -1:re.search(toc_end_dict[section], toc_content).end()+3].split()[0])

    pages_to_extract=[]
    for i in range(0, read_pdf.numPages):
        try:
            last3_pg_end=read_pdf.getPage(i).extractText().replace("\t"," ").replace("\n"," ").lower().
            replace("   |    2021 form 10-k","")[-3:]
            if int(re.sub('\D', '', last3_pg_end)) in [start_page, end_page]:
                pages_to_extract.append(i)
        except Exception:
            pass

    first_page=read_pdf.getPage(pages_to_extract[0]).extractText().replace("\t"," ").replace("\n"," ").lower()
               .replace("   |    2021 form 10-k","").replace("\"","").replace("'","’")
    first_page_keep=first_page[re.search(toc_start_dict[section], first_page).end():]

    last_page=read_pdf.getPage(pages_to_extract[1]).extractText().replace("\t"," ").replace("\n"," ").lower()
              .replace("   |    2021 form 10-k","").replace("\"","").replace("'","’")
    last_page_keep=last_page[:re.search(end_section_dict[section], last_page).start()]

    content_str=first_page_keep
    for i in range(pages_to_extract[0]+1,pages_to_extract[1]):
        page_content=read_pdf.getPage(i).extractText().replace("\t"," ").replace("\n"," ").lower()
                     .replace("   |    2021 form 10-k","").replace("\"","").replace("'","’")
        content_str+=page_content
    content_str+=last_page_keep
    return content_str
```

Create dictionaries for item 1A and 7

In [10]:
```
item1A_dict={}
for i in range(0,5):
    item1A_dict[ticker_ls[i]]= parse_section_text(ticker_ls[i],'1A')
```

In [11]:
```
item7_dict={}
for i in range(0,len(ticker_ls)):
    if ticker_ls[i] in ['jnj','lly']:
        item7_dict[ticker_ls[i]] = parse_section_text(ticker_ls[i],'7alt')
    else:
        item7_dict[ticker_ls[i]] = parse_section_text(ticker_ls[i],'7')
```

Taking a look at first 100 words of items 1A and 7 for Pfizer Inc. (PFE)

In [12]:
```
print("Item 1A: "+' '.join(item1A_dict['pfe'].split()[0:100]))
```

```
Item 1A: this section describes the material risks to our business, which should be considered carefully in addition to the other information in this report and our other filings with the sec. investors should be aware that it is not possible to predict or identify all such factors and that the following is pfizer inc. 2021 form 10-k 13not meant to be a co mplete discussion of all potential risks or uncertainties. additionally, our business is subject to general risks applicable to any company, such as economic conditions, geopoliti cal events, extreme weather and natural disasters. if known or unknown risks or uncertainties materialize,
```

In [13]:
```
print("Item 7: "+' '.join(item7_dict['pfe'].split()[0:100]))
```

```
Item 7: overview of our performance, operating environment, strategy and outlook financial highlights the following is a summary of certain financial performance metrics (in billi
```

```
ons, except per share data): 2021 total revenues——$81.3 billion 2021 net cash flow from operations——$32.6 billion an increase of 95% compared to 2020 an increase of 126% compared
to 2020 2021 reported diluted eps——$3.85 2021 adjusted diluted eps (non-gaap)——$4.42* an increase of 137% compared to 2020 an increase of 96% compared to 2020 * for additional inf
ormation regarding adjusted diluted eps (which is a non-gaap financial measure), including reconciliations of certain gaap reported to non-gaap adjusted information, see
```

B) Text Cleaning

Stop word and punctuation removal

Create list of stop words

In [14]:
```
nlp = spacy.load('en_core_web_lg')
```

In [15]:
```
stop_words_ls=list(nlp.Defaults.stop_words)
```

In [16]:
```
#Taking a look at first 10 elements
print(stop_words_ls[0:9])
```

```
['along', 'few', 'amongst', 'is', 'except', 'these', 'over', 'noone', 'hereupon']
```

Display list of punctuation

In [17]:
```
punctuation
```

Out[17]:
```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

Remove stop words and punctuation from dictionaries *item1A_dict* and *item7_dict*

In [18]:
```
def create_nsw_dictionaries(section_dict):
    nsw_dict={}
    nsw_str_ls=[]
    for t in range(0, len(ticker_ls)):
        all_words=word_tokenize(section_dict[ticker_ls[t]])
        nsw_str=''
        for i in range(0, len(all_words)):
            if all_words[i] not in stop_words_ls and all_words[i] not in punctuation:
                nsw_str+=all_words[i]+' '
        nsw_str_ls.append(nsw_str)
        nsw_dict[ticker_ls[t]]=nsw_str_ls[t]
    return nsw_dict
```

In [19]:
```
item1A_nsw_dict=create_nsw_dictionaries(item1A_dict)
item7_nsw_dict=create_nsw_dictionaries(item7_dict)
```

Table for stop words and punctuations removed:

In [20]:
```
def compare_length(dict):
    len_ls=[]
    for t in range(0, len(ticker_ls)):
        len_ls.append(len(dict[ticker_ls[t]].split()))
    return len_ls
```

In [56]:
```
company=pd.Series(company_ls)
item1A_orig=pd.Series(compare_length(item1A_dict))
item1A_nsw=pd.Series(compare_length(item1A_nsw_dict))
item7_orig=pd.Series(compare_length(item7_dict))
item7_nsw=pd.Series(compare_length(item7_nsw_dict))
length_df=pd.DataFrame({'Company': company, "Item 1A Original": item1A_orig, 'Item 1A Shortened': item1A_nsw,
          'Item 1A Stop Words & Punctuation': item1A_orig-item1A_nsw,
          'Item 7 Original': item7_orig, 'Item 7 Shortened': item7_nsw,
          'Item 7 Stop Words & Punctuation': item7_orig-item7_nsw,
          'Total Stop Words & Punctuation': item1A_orig-item1A_nsw+item7_orig-item7_nsw,
           '% of Words Removed': (item1A_orig-item1A_nsw+item7_orig-item7_nsw)/(item1A_orig+item7_orig)})
```

```
length_df['% of Words Removed']=length_df['% of Words Removed'].map('{:.1%}'.format)
title_style = [dict(selector="caption",
            props=[("text-align", "center"),
                   ("font-size", "100%"),
                   ("color", 'black'),
                   ("font-weight", "bold")])]


length_df.style.hide_index().set_properties(subset=['Item 1A Stop Words & Punctuation',
                                                    'Item 7 Stop Words & Punctuation',
                                                   'Total Stop Words & Punctuation', '% of Words Removed'],
                                                    **{'font-weight': 'bold'})
                                                    .set_caption('Text Cleaning Word Count Comparison')
                                                    .set_table_styles(title_style)
```

Out[56]:

Text Cleaning Word Count Comparison

Company	Item 1A Original	Item 1A Shortened	Item 1A Stop Words & Punctuation	Item 7 Original	Item 7 Shortened	Item 7 Stop Words & Punctuation	Total Stop Words & Punctuation	% of Words Removed
Pfizer Inc.	10668	6045	**4623**	16942	10640	**6302**	**10925**	**39.6%**
Moderna, Inc.	32508	17472	**15036**	8484	5096	**3388**	**18424**	**44.9%**
Johnson & Johnson	5972	3613	**2359**	10238	6703	**3535**	**5894**	**36.4%**
Eli Lilly and Company	5894	3353	**2541**	10483	6497	**3986**	**6527**	**39.9%**
AbbVie Inc.	7759	4942	**2817**	9415	6123	**3292**	**6109**	**35.6%**

II. Comparison of Sentiment Word Frequency

A) Create lists for sentiment words

For this analysis, I will use the Master Dictionary created by Loughran and McDonald, which includes 354 positive and 2355 negative words that have frequently appeared in 10-k's.

Import Master Dictionary

In [22]:
```
master_dict=pd.read_csv(
            "/Users/chenmouse/Desktop/NLP/Final Project - 10K Sentiment Analysis/Loughran-McDonald_MasterDict.csv")
```

Taking a look at first 10 rows

In [23]:
```
master_dict.head(10)
```

Out[23]:

	Word	Seq_num	Word Count	Word Proportion	Average Proportion	Std Dev	Doc Count	Negative	Positive	Uncertainty	Litigious	Strong_Modal	Weak_Modal	Constraining	Syllables	Source
0	AARDVARK	1	354	1.550000e-08	1.420000e-08	3.820000e-06	99	0	0	0	0	0	0	0	2	12of12inf
1	AARDVARKS	2	3	1.310000e-10	8.650000e-12	9.240000e-09	1	0	0	0	0	0	0	0	2	12of12inf
2	ABACI	3	9	3.940000e-10	1.170000e-10	5.290000e-08	7	0	0	0	0	0	0	0	3	12of12inf
3	ABACK	4	29	1.270000e-09	6.650000e-10	1.600000e-07	28	0	0	0	0	0	0	0	2	12of12inf
4	ABACUS	5	8570	3.750000e-07	3.810000e-07	3.530000e-05	1108	0	0	0	0	0	0	0	3	12of12inf
5	ABACUSES	6	0	0.000000e+00	0.000000e+00	0.000000e+00	0	0	0	0	0	0	0	0	4	12of12inf
6	ABAFT	7	4	1.750000e-10	2.300000e-11	2.460000e-08	1	0	0	0	0	0	0	0	2	12of12inf
7	ABALONE	8	142	6.220000e-09	4.970000e-09	1.070000e-06	48	0	0	0	0	0	0	0	4	12of12inf
8	ABALONES	9	1	4.380000e-11	8.280000e-11	8.850000e-08	1	0	0	0	0	0	0	0	4	12of12inf
9	ABANDON	10	127090	5.560000e-06	4.700000e-06	3.310000e-05	66312	2009	0	0	0	0	0	0	3	12of12inf

Filter for sentiment words, which are indicated by the year added to the master dictionary under "Negative" and "Positive" columns

List of negative words

In [24]:
```
neg_words_ls=master_dict[(master_dict['Negative']!=0)]['Word'].tolist()
neg_words_ls=[i.lower() for i in neg_words_ls]
print(neg_words_ls[0:10])
```

```
['abandon', 'abandoned', 'abandoning', 'abandonment', 'abandonments', 'abandons', 'abdicated', 'abdicates', 'abdicating', 'abdication']
```

List of positive words

In [25]:
```
pos_words_ls=master_dict[(master_dict['Positive']!=0)]['Word'].tolist()
pos_words_ls=[i.lower() for i in pos_words_ls]
print(pos_words_ls[0:10])
```

```
['able', 'abundance', 'abundant', 'acclaimed', 'accomplish', 'accomplished', 'accomplishes', 'accomplishing', 'accomplishment', 'accomplishments']
```

In [26]:
```
len(neg_words_ls), len(pos_words_ls)
```

Out[26]:
```
(2355, 354)
```

Notice that the dictionary is comprehensive of all word forms (i.e. all forms of "accomplish"), thus there is no need to lemmatize item 1A and item 7 dictionaries with no stop words.

B) Count number of positive and negative words in 10-k's

Count number of positive words

In [27]:
```
def count_positive(dict, ticker):
    pos_count=0
    for i in range(0,len(word_tokenize(dict[ticker]))):
        if word_tokenize(dict[ticker])[i] in pos_words_ls:
            pos_count+=1
    return pos_count
```

In [28]:
```
def count_negative(dict, ticker):
    neg_count=0
    for i in range(0,len(word_tokenize(dict[ticker]))):
        if word_tokenize(dict[ticker])[i] in neg_words_ls:
            neg_count+=1
    return neg_count
```

Count number of positive and negative words

In [29]:
```
item1A_pos_count={}
item7_pos_count={}
item1A_neg_count={}
item7_neg_count={}
for i in range(0, len(ticker_ls)):
    item1A_pos_count[ticker_ls[i]]=count_positive(item1A_nsw_dict, ticker_ls[i])
    item7_pos_count[ticker_ls[i]]=count_positive(item7_nsw_dict, ticker_ls[i])
    item1A_neg_count[ticker_ls[i]]=count_negative(item1A_nsw_dict, ticker_ls[i])
    item7_neg_count[ticker_ls[i]]=count_negative(item7_nsw_dict, ticker_ls[i])
```

In [30]:
```
item1A_pos_count, item1A_neg_count, item7_pos_count, item7_neg_count
```

Out[30]:
```
({'pfe': 111, 'mrna': 453, 'jnj': 75, 'lly': 70, 'abbv': 86},
 {'pfe': 417, 'mrna': 1291, 'jnj': 290, 'lly': 338, 'abbv': 304},
 {'pfe': 179, 'mrna': 58, 'jnj': 116, 'lly': 92, 'abbv': 108},
 {'pfe': 226, 'mrna': 46, 'jnj': 138, 'lly': 164, 'abbv': 78})
```

In [31]:
```
def word_count_table(item_count_pos, item_count_neg, item_nsw):
    pos=pd.Series(item_count_pos.values())
```

```
    neg=pd.Series(item_count_neg.values())
    wordcount=pd.DataFrame({'Company': company,'Total w/o Stop Words': item_nsw,
              'Positive Words': pos,
              'Positive Words Frequency': (pos/item_nsw),
              'Negative Words': neg,
              'Negative Words Frequency': (neg/item_nsw)})
    wordcount=wordcount.sort_values(by=['Negative Words Frequency'],ascending=False)
    wordcount['Positive Words Frequency']=wordcount['Positive Words Frequency'].map('{:.1%}'.format)
    wordcount['Negative Words Frequency']=wordcount['Negative Words Frequency'].map('{:.1%}'.format)
    return wordcount
```

In [32]:
```
sentiment1a=word_count_table(item1A_pos_count, item1A_neg_count, item1A_nsw)
sentiment7=word_count_table(item7_pos_count, item7_neg_count, item7_nsw)
```

In [33]:
```
styles = [dict(selector="caption",props=[("text-align", "center"), ("font-size", "100%"), ("color", 'black'),
                  ("font-weight", "bold")])]
```

In [34]:
```
sentiment1a.style.hide_index().set_properties(subset=['Positive Words Frequency','Negative Words Frequency'],
**{'font-weight': 'bold'}).set_caption('Item 1A Sentiment Word Count').set_table_styles(styles)
```

Out[34]:

Item 1A Sentiment Word Count

Company	Total w/o Stop Words	Positive Words	Positive Words Frequency	Negative Words	Negative Words Frequency
Eli Lilly and Company	3353	70	**2.1%**	338	**10.1%**
Johnson & Johnson	3613	75	**2.1%**	290	**8.0%**
Moderna, Inc.	17472	453	**2.6%**	1291	**7.4%**
Pfizer Inc.	6045	111	**1.8%**	417	**6.9%**
AbbVie Inc.	4942	86	**1.7%**	304	**6.2%**

In [35]:
```
sentiment7.style.hide_index().set_properties(subset=['Positive Words Frequency','Negative Words Frequency'],
**{'font-weight': 'bold'}).set_caption('Item 7 Sentiment Word Count').set_table_styles(styles)
```

Out[35]:

Item 7 Sentiment Word Count

Company	Total w/o Stop Words	Positive Words	Positive Words Frequency	Negative Words	Negative Words Frequency
Eli Lilly and Company	6497	92	**1.4%**	164	**2.5%**
Pfizer Inc.	10640	179	**1.7%**	226	**2.1%**
Johnson & Johnson	6703	116	**1.7%**	138	**2.1%**
AbbVie Inc.	6123	108	**1.8%**	78	**1.3%**
Moderna, Inc.	5096	58	**1.1%**	46	**0.9%**

In [36]:
```
sentiment_total=sentiment1a.add(sentiment7)
sentiment_total['Company']=company
sentiment_total['Positive Words Frequency']=(sentiment_total['Positive Words']
                                             /sentiment_total['Total w/o Stop Words']).map('{:.1%}'.format)
sentiment_total['Negative Words Frequency']=(sentiment_total['Negative Words']
                                             /sentiment_total['Total w/o Stop Words']).map('{:.1%}'.format)
```

In [37]:
```
sentiment_total.sort_values(by=['Negative Words Frequency'], ascending=False).style.hide_index().set_properties(
                            subset=['Positive Words Frequency','Negative Words Frequency'],
**{'font-weight': 'bold'}).set_caption('Total Sentiment Word Count').set_table_styles(styles)
```

Out[37]:

Total Sentiment Word Count					
Company	**Total w/o Stop Words**	**Positive Words**	**Positive Words Frequency**	**Negative Words**	**Negative Words Frequency**
Moderna, Inc.	22568	511	**2.3%**	1337	**5.9%**
Eli Lilly and Company	9850	162	**1.6%**	502	**5.1%**
Johnson & Johnson	10316	191	**1.9%**	428	**4.1%**
Pfizer Inc.	16685	290	**1.7%**	643	**3.9%**
AbbVie Inc.	11065	194	**1.8%**	382	**3.5%**

C) Conclusion

With the range of Positive Words Frequency being much narrower than that of Negative Words Frequency among the five companies, Negative Words Frequency serves as a differentiator in the tones of the 10-k financial reports.

Overall, Moderna, Inc. ranks highest in frequency of negative words, followed by Eli Lilly and Company, Johnson & Johnson, Pfizer Inc. and AbbVie Inc. However, for the individual items, Eli Lilly and Company ranks highest instead. It appears Moderna's high overall ranking is mostly attributable to high negative words frequency in item 1A.

Zooming in on item 7, which discusses the company's annual performance in greater detail than item 1A, Eli Lilly and Company, Pfizer Inc. and Johnson and Johnson's reports are worth a more in-depth read as their tones indicate more negative sentiments than the rest.

We will shed more light on this analysis in section II, higlighting the sentiment words that appear most frequently and the sentences they belong to.

III. Identification of Most Frequent Sentiment Words

Combine positive and negative words into one sentiment word list

In [38]:
```
sent_word_ls=neg_words_ls+pos_words_ls
```

Retrieve 10 most frequently appearing sentiment words in item 1A and 7 as dictionaries

In [39]:
```
def top_10_sent_words(item_dict, ticker):
    sent_word_dict={}
    for i in range(0, len(sent_word_ls)):
        sent_word_dict[sent_word_ls[i]]=word_tokenize(item_dict[ticker]).count(sent_word_ls[i])
    sent_word_dict={k: v for k, v in sorted(sent_word_dict.items(), key=lambda item: item[1])}
    top10ls=list(sent_word_dict)[-10:]
    top10={}
    for i in range(0, len(top10ls)):
        top10[top10ls[i]]=sent_word_dict[top10ls[i]]
    return top10
```

In [40]:
```
top10pfe_1a=top_10_sent_words(item1A_nsw_dict, 'pfe')
top10mrna_1a=top_10_sent_words(item1A_nsw_dict, 'mrna')
top10jnj_1a=top_10_sent_words(item1A_nsw_dict, 'jnj')
top10lly_1a=top_10_sent_words(item1A_nsw_dict, 'lly')
top10abbv_1a=top_10_sent_words(item1A_nsw_dict, 'abbv')
top10pfe_7=top_10_sent_words(item7_nsw_dict, 'pfe')
top10mrna_7=top_10_sent_words(item7_nsw_dict, 'mrna')
top10jnj_7=top_10_sent_words(item7_nsw_dict, 'jnj')
top10lly_7=top_10_sent_words(item7_nsw_dict, 'lly')
top10abbv_7=top_10_sent_words(item7_nsw_dict, 'abbv')
```

Compile all dictionaries into a Dataframe

In [41]:
```
top101adict_ls=[top10pfe_1a, top10mrna_1a, top10jnj_1a, top10lly_1a, top10abbv_1a]
top107dict_ls=[top10pfe_7, top10mrna_7, top10jnj_7, top10lly_7, top10abbv_7]
```

In [42]:
```
def top_10_dict(company_name, dict):
    name_sr=pd.Series([company_name]*10)
    top10df=pd.DataFrame(dict.items(), columns=['Sentiment Word', 'Frequency'])
    top10df['Sentiment Word']=top10df['Sentiment Word'].str.capitalize()
    top10df=top10df.sort_values(['Frequency'], ascending=False).reset_index(drop=True)
    return top10df
```

In [43]:
```
def top10_df(item_dict_ls,item_str):
    df1 = top_10_dict(company_ls[0], item_dict_ls[0])
    df2 = top_10_dict(company_ls[1], item_dict_ls[1])
    df3 = top_10_dict(company_ls[2], item_dict_ls[2])
    df4 = top_10_dict(company_ls[3], item_dict_ls[3])
    df5 = top_10_dict(company_ls[4], item_dict_ls[4])
    df1_style = df1.style.hide_index().set_table_attributes("style='display:inline; margin-right:20px;'")
                .set_caption(item_str+str(company_ls[0])).set_table_styles(styles)
    df2_style = df2.style.hide_index().set_table_attributes("style='display:inline; margin-right:20px;'")
                .set_caption(item_str+str(company_ls[1])).set_table_styles(styles)
    df3_style = df3.style.hide_index().set_table_attributes("style='display:inline; margin-right:20px;'")
                .set_caption(item_str+str(company_ls[2])).set_table_styles(styles)
    df4_style = df4.style.hide_index().set_table_attributes("style='display:inline; margin-right:20px;'")
                .set_caption(item_str+str(company_ls[3])).set_table_styles(styles)
    df5_style = df5.style.hide_index().set_table_attributes("style='display:inline'")
                .set_caption(item_str+ str(company_ls[4])).set_table_styles(styles)
    return display_html(df1_style._repr_html_() + df2_style._repr_html_() + df3_style._repr_html_()
                 + df4_style._repr_html_() + df5_style._repr_html_(), raw=True)
```

Top 10 Most Frequent Sentiment Words

In [44]:
```
top10_df(top101adict_ls,"Item 1A - ")
```

Item 1A - Pfizer Inc.		Item 1A - Moderna, Inc.		Item 1A - Johnson & Johnson		Item 1A - Eli Lilly and Company		Item 1A - AbbVie Inc.	
Sentiment Word	Frequency	Sentiment Word	Frequency	Sentiment Word	Frequency	Sentiment Word	Frequency	Sentiment Word	Frequency
Challenges	26	Adversely	71	Adversely	23	Adversely	21	Adverse	27
Adversely	20	Collaborators	55	Litigation	11	Litigation	13	Adversely	23
Claims	18	Adverse	47	Delays	9	Failure	12	Failure	16
Litigation	12	Delays	45	Challenges	8	Unauthorized	9	Successful	10
Adverse	11	Claims	43	Negatively	8	Claims	8	Impairment	9
Disruptions	11	Fail	40	Damage	7	Successful	8	Profitability	9
Delays	10	Delay	36	Investigations	7	Disruption	7	Loss	8
Fail	10	Failure	34	Loss	7	Failures	7	Negatively	7
Loss	10	Able	32	Successful	7	Investigations	7	Restated	7
Able	10	Litigation	28	Effective	6	Loss	7	Successfully	7

In [45]:
```
top10_df(top107dict_ls,"Item 7 - ")
```

Item 7 - Pfizer Inc.		Item 7 - Moderna, Inc.		Item 7 - Johnson & Johnson		Item 7 - Eli Lilly and Company		Item 7 - AbbVie Inc.	
Sentiment Word	Frequency	Sentiment Word	Frequency	Sentiment Word	Frequency	Sentiment Word	Frequency	Sentiment Word	Frequency
Benefit	25	Loss	8	Loss	16	Impairment	17	Benefit	18
Collaboration	23	Collaboration	8	Gains	13	Benefit	17	Collaboration	10
Decline	19	Alliances	4	Losses	12	Loss	10	Strong	10
Discontinued	19	Benefit	4	Achieved	12	Litigation	9	Loss	9
Gains	19	Enable	4	Litigation	11	Adversely	8	Severe	9
Impairment	18	Termination	3	Benefit	11	Failure	8	Impairment	7
Restructuring	15	Advances	3	Positive	10	Exclusivity	7	Favorable	7
Losses	12	Collaborators	3	Doubtful	8	Severe	6	Inadequate	6
Greater	10	Effective	3	Negative	7	Favorable	6	Favorably	6
Unfavorable	9	Progress	3	Strong	7	Gains	6	Effective	5

You may notice one or two of the top 10 most frequent sentiment words being of the same root word (i.e. "adverse" and "adversely"). This is due to lemmatization being skipped to save significant time on running this notebook. Regardless, the output still produces top 8-9 words.

Conclusion

Item 1A tends to include more negative words because it discusses risk factors. This is consistent with the five companies. </mark> Item 7 shows a more diverse range of sentiments, with a more even split between positive and negative words. In conjunction with the quantitative comparison of sentiment words frequency in section I, searching for the most frequent sentiment words can lead to informative insight, with the following examples:

AbbVie Inc., Item 1A

"Successful(ly)", "Adversely"

The **successful** discovery, development, manufacturing and sale of biologics is a long, expensive and uncertain process. There are unique risks and uncertainties with biologics. For example, access to and supply of necessary biological materials, such as cell lines, may be limited and governmental regulations restrict access to and regulate the transport and use of such materials. In addition, the development, manufacturing and sale of biologics is subject to regulations that are often more complex and extensive than the regulations applicable to other pharmaceutical products...Biologics are also frequently costly to manufacture because production inputs are derived from living animal or plant material, and some biologics cannot be made synthetically. Failure to **successfully** discover, develop, manufacture and sell biologics—including Humira—could **adversely** impact AbbVie's business and results of operations.

Pfizer Inc., Item 1A

"Litigation", "Loss", "Adversely"

We recorded direct product and/or Alliance revenues of more than $1 billion for each of nine products that collectively accounted for 75% of our total revenues in 2021. In particular, Comirnaty/BNT162b2 accounted for 45% of our total revenues in 2021. For additional information, see Notes 1 and 17. If these products or any of our other major products were to experience **loss** of patent protection (if applicable), changes in prescription or vaccination growth rates, material product liability **litigation**, unexpected side effects or safety concerns, regulatory proceedings, negative publicity affecting doctor or patient confidence, pressure from existing competitive products, changes in labeling, pricing and access pressures or supply shortages or if a new, more effective product should be introduced, the **adverse** impact on our revenues could be significant.

Eli Lilly and Company, Item 7

"Exclusivity", "Favorable", "Loss", "Severe(ly)"

Revenue of Alimta, a treatment for various cancers, decreased 2 percent in the U.S., driven by decreased volume, partially offset by higher realized prices. Revenue outside the U.S. decreased 22 percent, primarily driven by decreased volume due to the entry of generic competition in certain markets and, to a lesser extent, lower realized prices, partially offset by the **favorable** impact of foreign exchange rates. Following the **loss** of **exclusivity** in major European countries and Japan in June 2021, we faced, and remain exposed to, generic competition which has eroded revenue and is likely to continue to rapidly and **severely** erode revenue from current levels. In the U.S., we expect the limited entry of generic competition starting February 2022 and subsequent unlimited entry starting April 2022. We expect that the entry of generic competition following the **loss** of **exclusivity** in the U.S. will cause a rapid and **severe** decline in revenue.

Moderna Inc., Item 7

"Advance", "Progress"

We expect that research and development expenses will increase in 2022 as we continue to **progress** our indication expansion of mRNA-1273, and continue to develop our pipeline and **advance** our product candidates into later-stage development. In addition, we also expect to incur significant costs related to the development of variantspecific COVID-19 candidates and our next-generation COVID-19 vaccine candidate (mRNA-1283).

Johnson & Johnson, Item 7

"Achieved", "Positive"

In 2021, sales by companies in Europe **achieved** growth of 24.3% as compared to the prior year, which included operational growth of 20.7% and a **positive** currency impact of 3.6%. Sales by companies in the Western Hemisphere (excluding the U.S.) **achieved** growth of 7.8% as compared to the prior year, which included operational growth of 7.3% and a **positive** currency impact of 0.5%. Sales by companies in the Asia-Pacific, Africa region **achieved** growth of 14.1% as compared to the prior year, including operational growth of 11.4% and a **currency** impact of 2.7%.