In [ ]:

```
```

In [6]:

```
# Import libraries
import numpy as np
#import scikit-fuzzy as fuzz
#from scikit-fuzzy import control as ctrl
from urllib.request import urlopen, Request
from bs4 import BeautifulSoup
import os
import pandas as pd
import matplotlib.pyplot as plt
#%matplotlib inline
# NLTK VADER for sentiment analysis
from nltk.sentiment.vader import SentimentIntensityAnalyzer

finwiz_url = 'https://finviz.com/quote.ashx?t=GOOGL&ty=c&ta=1&p=d'
```

In [7]:

```
news_tables = {}
tickers = ['GOOG']

for ticker in tickers:
    url = finwiz_url + ticker
    req = Request(url=url,headers={'user-agent': 'my-app/0.0.1'})
    response = urlopen(req)
    # Read the contents of the file into 'html'
    html = BeautifulSoup(response)
    # Find 'news-table' in the Soup and load it into 'news_table'
    news_table = html.find(id='news-table')
    # Add the table to our dictionary
    news_tables[ticker] = news_table
```

```
/usr/lib/python3/dist-packages/bs4/__init__.py:181: UserWarning: No pa
rser was explicitly specified, so I'm using the best available HTML pa
rser for this system ("lxml"). This usually isn't a problem, but if yo
u run this code on another system, or in a different virtual environme
nt, it may use a different parser and behave differently.

The code that caused this warning is on line 193 of the file /usr/lib/
python3.6/runpy.py. To get rid of this warning, change code that looks
like this:

 BeautifulSoup(YOUR_MARKUP})

to this:

 BeautifulSoup(YOUR_MARKUP, "lxml")

  markup_type=markup_type))
```

In [8]:

```
# Read one single day of headlines for 'AMZN'
goog = news_tables['GOOG']
# Get all the table rows tagged in HTML with <tr> into 'amzn_tr'
goog_tr = goog.findAll('tr')

for i, table_row in enumerate(goog_tr):
    # Read the text of the element 'a' into 'link_text'
    a_text = table_row.a.text
    # Read the text of the element 'td' into 'data_text'
    td_text = table_row.td.text
    # Print the contents of 'link_text' and 'data_text'
    print(a_text)
    print(td_text)
    # Exit after printing 4 rows of data
    if i == 3:
        break
```

```
Coronavirus is reshaping Big Techs battle for city streets
May-14-20 11:00PM
Texas Cases Jump; AMA Warns on Antibody Tests: Virus Update
06:49PM
3 Video Conferencing Apps Benefiting From The COVID Tailwind
05:12PM
Trump is right that rich guys sometimes bet against the stock market
but heres a fuller explanation
04:07PM
```

In [9]:

```
parsed_news = []

# Iterate through the news
for file_name, news_table in news_tables.items():
    # Iterate through all tr tags in 'news_table'
    for x in news_table.findAll('tr'):
        # read the text from each tr tag into text
        # get text from a only
        text = x.a.get_text()
        # splite text in the td tag into a list
        date_scrape = x.td.text.split()
        # if the length of 'date_scrape' is 1, load 'time' as the only element

        if len(date_scrape) == 1:
            time = date_scrape[0]

        # else load 'date' as the 1st element and 'time' as the second
        else:
            date = date_scrape[0]
            time = date_scrape[1]
        # Extract the ticker from the file name, get the string up to the 1st '_'
        ticker = file_name.split('_')[0]

        # Append ticker, date, time and headline as a list to the 'parsed_news' lis
        parsed_news.append([ticker, date, time, text])

parsed_news
```

Out[9]:

```
[['GOOG',
  'May-14-20',
  '11:00PM',
  'Coronavirus is reshaping Big Techs battle for city streets'],
 ['GOOG',
  'May-14-20',
  '06:49PM',
  'Texas Cases Jump; AMA Warns on Antibody Tests: Virus Update'],
 ['GOOG',
  'May-14-20',
  '05:12PM',
  '3 Video Conferencing Apps Benefiting From The COVID Tailwind'],
 ['GOOG',
  'May-14-20',
  '04:07PM',
  'Trump is right that rich guys sometimes bet against the stock mar
ket  but heres a fuller explanation'],
```

In [10]:

```
# Instantiate the sentiment intensity analyzer
vader = SentimentIntensityAnalyzer()

# Set column names
columns = ['ticker', 'date', 'time', 'headline']

# Convert the parsed_news list into a DataFrame called 'parsed_and_scored_news'
parsed_and_scored_news = pd.DataFrame(parsed_news, columns=columns)

# Iterate through the headlines and get the polarity scores using vader
scores = parsed_and_scored_news['headline'].apply(vader.polarity_scores).tolist()

# Convert the 'scores' list of dicts into a DataFrame
scores_df = pd.DataFrame(scores)

# Join the DataFrames of the news and the list of dicts
parsed_and_scored_news = parsed_and_scored_news.join(scores_df, rsuffix='_right')

# Convert the date column from string to datetime
parsed_and_scored_news['date'] = pd.to_datetime(parsed_and_scored_news.date).dt.dat

parsed_and_scored_news.head(50)
```

Out[10]:

	ticker	date	time	headline	neg	neu	pos	compound
0	GOOG	2020-05-14	11:00PM	Coronavirus is reshaping Big Techs battle for ...	0.245	0.755	0.000	-0.3818
1	GOOG	2020-05-14	06:49PM	Texas Cases Jump; AMA Warns on Antibody Tests:...	0.135	0.865	0.000	-0.1027
2	GOOG	2020-05-14	05:12PM	3 Video Conferencing Apps Benefiting From The ...	0.000	1.000	0.000	0.0000
3	GOOG	2020-05-14	04:07PM	Trump is right that rich guys sometimes bet ag...	0.000	0.867	0.133	0.3182
4	GOOG	2020-05-14	03:26PM	How to start a blog in retirement	0.000	1.000	0.000	0.0000
5	GOOG	2020-05-14	02:57PM	Market is 'going to be lower for longer': Stra...	0.216	0.784	0.000	-0.2960
		2020-		Whats happened to value				

In [12]:

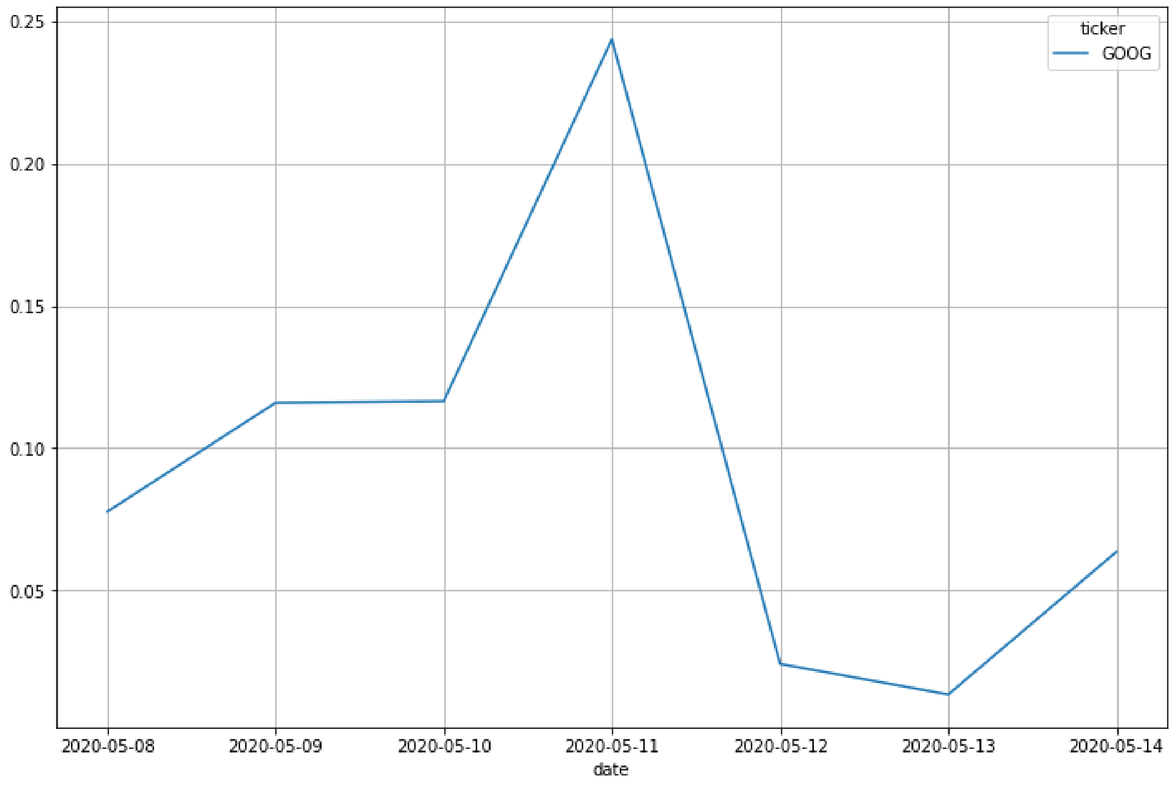
```
plt.rcParams['figure.figsize'] = [12, 8]

# Group by date and ticker columns from scored_news and calculate the mean
mean_scores = parsed_and_scored_news.groupby(['ticker','date']).mean()

# Unstack the column ticker
mean_scores = mean_scores.unstack()

# Get the cross-section of compound in the 'columns' axis
mean_scores = mean_scores.xs('compound', axis="columns").transpose()

# Plot a bar chart with pandas
mean_scores.plot(kind = 'line')
plt.grid()
```



In [ ]:

```

```