In [1]:
```
#Utilize google_play_scraper package to pull app store reviews from google store
#Import app, Sort and reviews packages
from google_play_scraper import app
from google_play_scraper import Sort, reviews


#Import packages for pandas, numpy, regex, nltk and matplotlib
import pandas as pd
import numpy as np
import re
import nltk
import matplotlib.pyplot as plt
from datetime import datetime

#Install google-play-scraper API if necessary
#!pip install google-play-scraper
```

Google Play Store Robinhood App Reviews

In [2]:
```
#Create result object to search for Robinhood app
result = app(
    'com.robinhood.android',
    lang='en', # defaults to 'en'
    country='us' # defaults to 'us'
)

#View app info to confirm appID
print(result)
```

```
{'title': 'Robinhood - Investment & Trading, Commission-free', 'description':
"Invest in stocks, options, and funds with Robinhood Financial. Buy and sell
crypto like Bitcoin and Dogecoin with Robinhood Crypto. All commission-free w
ith no account minimums. Other fees may apply*.\r\n\r\nWhether you’re new to
the markets or an experienced trader, we have the tools to help you invest wi
th greater confidence. From commission-free trading to award winning design,
investing is now more approachable and affordable. To get you started, you’ll
get your first stock on us. Certain limitations apply.\r\n\r\nHere's what you
get when you join Robinhood:\r\n<b>Finance Explained</b> - We’ll help you bet
ter understand financial markets so you can invest in funds, stocks, and opti
ons, all commission-free.\r\n\r\n<b>Trading Tools</b> - \r\nBefore buying a s
tock, ETF, or cryptocurrency, you can access real-time market data, see analy
st ratings, read relevant news articles, and get notified about important eve
nts.  \r\n\r\n<b>Secure and Trusted</b>\r\nYour security is our priority. Rob
inhood uses cutting-edge security measures to help protect investor assets an
d personal information.\r\n\r\nPlus, we’ve got a full range of products to he
lp make your money work harder for you.\r\n\r\n<b>STOCKS, FUNDS, OPTIONS, AND
CRYPTO</b>\r\nWith Robinhood Financial, you can invest in stocks, funds, and
options. You can also buy and sell cryptocurrencies like Bitcoin (BTC), Ether
```

In [3]:
```
#After confirming app, we can pull 5,000 reviews to analyze
result, continuation_token = reviews(
    'com.robinhood.android',
    lang='en',
    country='us',
    sort=Sort.MOST_RELEVANT,
    count=5000,
    filter_score_with=None
)

#Store reviews in result object
result, _ = reviews(
    'com.robinhood.android',
    continuation_token=continuation_token
)

print(result)
```

```
IOPub data rate exceeded.
The notebook server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--NotebookApp.iopub_data_rate_limit`.

Current values:
NotebookApp.iopub_data_rate_limit=1000000.0 (bytes/sec)
NotebookApp.rate_limit_window=3.0 (secs)
```

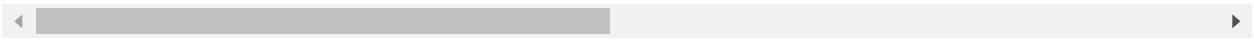
In [4]:
```
#Create pandas dataframe to store data
df = pd.DataFrame (result ,columns=['reviewId','userName','userImage','content',

#Confirm dataframe output
df
```

Out[4]:

	reviewId	userName	
0	gp:AOqpTOHMEtur9u3MbRtH7uVCNIWzs2JHtxj7GknV3FV...	Logan	lh.googleusercontent.com
1	gp:AOqpTOGKL7YwldYR7kqFwXezikvuV85Zq-hyVsuh44c...	alicia Johnson	lh.googleusercontent.com
2	gp:AOqpTOFMekrSUaR2_odwjOE9BJNhlPctEt89PRVOxpR...	Sukrit Ranjan	lh.googleusercontent.com
3	gp:AOqpTOEjggO2SugHUpwCuJ805y7kYEMkQW6xvX9Tag2...	Julian Maldonado	lh.googleusercontent.com/
4	gp:AOqpTOE0m5yZ1YWxeXFFmQilNFr9ysrCRcCvVfsUg4w...	Kenneth Chandler	lh.googleusercontent.com
...	...	...	
4995	gp:AOqpTOEmtxom0XJDA2SpPixoX2nQEfZyC96mDkbTUx7...	Laura Crisan	lh.googleusercontent.com
4996	gp:AOqpTOFlEt1YnpKzj_kEQlGnOcV1410UZpdty44wnOT...	Anonymouse	lh.googleusercontent.com/
4997	gp:AOqpTOFtKEYj4YNx2Iar3opg6232wi-xnex-IIE9_jH...	Allen Naegelen	lh.googleusercontent.com
4998	gp:AOqpTOGzlfKM0o2fdaPM51XMTjuGAZLhzJF293-i5L5...	Jeffrey Eide	lh.googleusercontent.com
4999	gp:AOqpTOGkTOmrfVxPTfUXXOWD_5tv10y5vU8BNrkLNHk...	A. W.	lh.googleusercontent.com/

5000 rows × 10 columns



In [5]:
```
#Create separate dataframe, dropping unecessary columns
review_df = df[['reviewId','content','score','thumbsUpCount','reviewCreatedVersic

#Verify dataframe output
review_df
```

Out[5]:

	reviewId	content	score	thumbsUpCoun
0	gp:AOqpTOHMEtur9u3MbRtH7uVCNIWzs2JHtxj7GknV3FV...	Making the decision to uninstall and post this...	1	
1	gp:AOqpTOGKL7YwldYR7kqFwXezikvuV85Zq-hyVsuh44c...	Confusing. Navigation visuals unclear, very un...	2	
2	gp:AOqpTOFMekrSUaR2_odwjOE9BJNhlPctEt89PRVOxpR...	It's very hard to get helped when you need it....	2	
3	gp:AOqpTOEjggO2SugHUpwCuJ805y7kYEMkQW6xvX9Tag2...	I've lost faith in this companies business pra...	1	
4	gp:AOqpTOE0m5yZ1YWxeXFFmQilNFr9ysrCRcCvVfsUg4w...	I was using other investment apps prior to swi...	5	
...	...	...	...	...
4995	gp:AOqpTOEmtxom0XJDA2SpPixoX2nQEfZyC96mDkbTUx7...	It's a great interface, but can't recommend th...	1	
4996	gp:AOqpTOFlEt1YnpKzj_kEQlGnOcV1410UZpdty44wnOT...	haven't tried pulling out money yet but so far...	5	
4997	gp:AOqpTOFtKEYj4YNx2lar3opg6232wi-xnex-IIE9_jH...	Not a good platform for stocks or cryptocurren...	1	
4998	gp:AOqpTOGzlfKM0o2fdaPM51XMTjuGAZLhzJF293-i5L5...	After multiple attempts to contact the adminis...	1	
4999	gp:AOqpTOGkTOmrfVxPTfUXXOWD_5tv10y5vU8BNrkLNHk...	Many outages, company shuts down trading of lu...	1	

5000 rows × 8 columns

Robinhood Rating Time Series Analysis

In [6]:
```
import warnings
warnings.filterwarnings("ignore")
```

Resampling Data

In [7]:
```
#Create dataframe to hold scores and set datetimeIndex
score_df = review_df[['score','thumbsUpCount']]
score_df['date']= review_df['at']

score_df.set_index('date',inplace=True)
score_df.index
```

Out[7]:
```
DatetimeIndex(['2021-02-08 18:06:41', '2021-03-22 04:51:31',
               '2021-02-18 09:39:06', '2021-03-16 22:33:59',
               '2021-03-21 07:23:13', '2021-04-29 17:55:22',
               '2021-04-27 15:34:12', '2021-05-05 13:31:54',
               '2021-02-14 09:32:29', '2021-03-17 17:49:55',
               ...
               '2021-01-10 14:23:01', '2021-01-29 11:03:52',
               '2021-02-01 18:55:28', '2021-01-31 22:25:14',
               '2021-01-29 13:05:49', '2021-01-28 17:33:50',
               '2021-01-11 10:05:25', '2021-02-02 16:38:36',
               '2021-01-29 13:35:00', '2021-01-30 05:40:22'],
              dtype='datetime64[ns]', name='date', length=5000, freq=None)
```

In [8]:
```
#Observe means from monthly data
score_df.resample(rule = 'M').mean()
```

Out[8]:

date	score	thumbsUpCount
2020-11-30	2.472727	23.418182
2020-12-31	2.333333	35.903704
2021-01-31	1.249169	91.550071
2021-02-28	1.344981	16.240788
2021-03-31	3.283721	0.700000
2021-04-30	2.938272	0.524691
2021-05-31	3.328638	0.286385

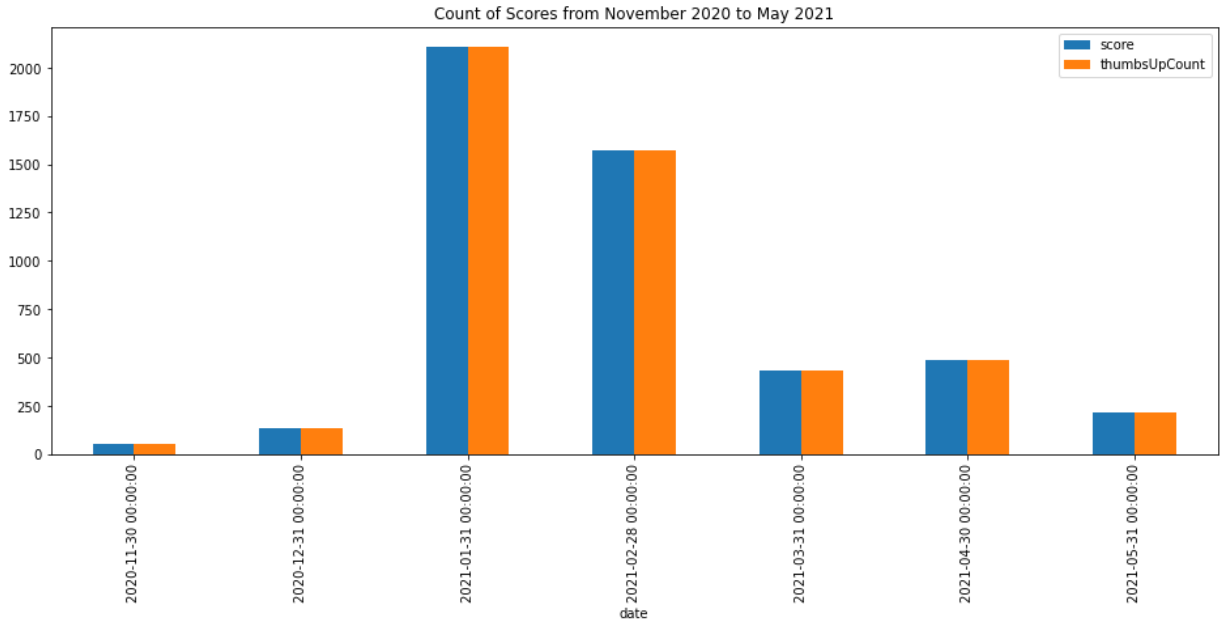
In [9]:
```
#Observe counts of monthly data
score_df.resample(rule = 'M').count()
```

Out[9]:

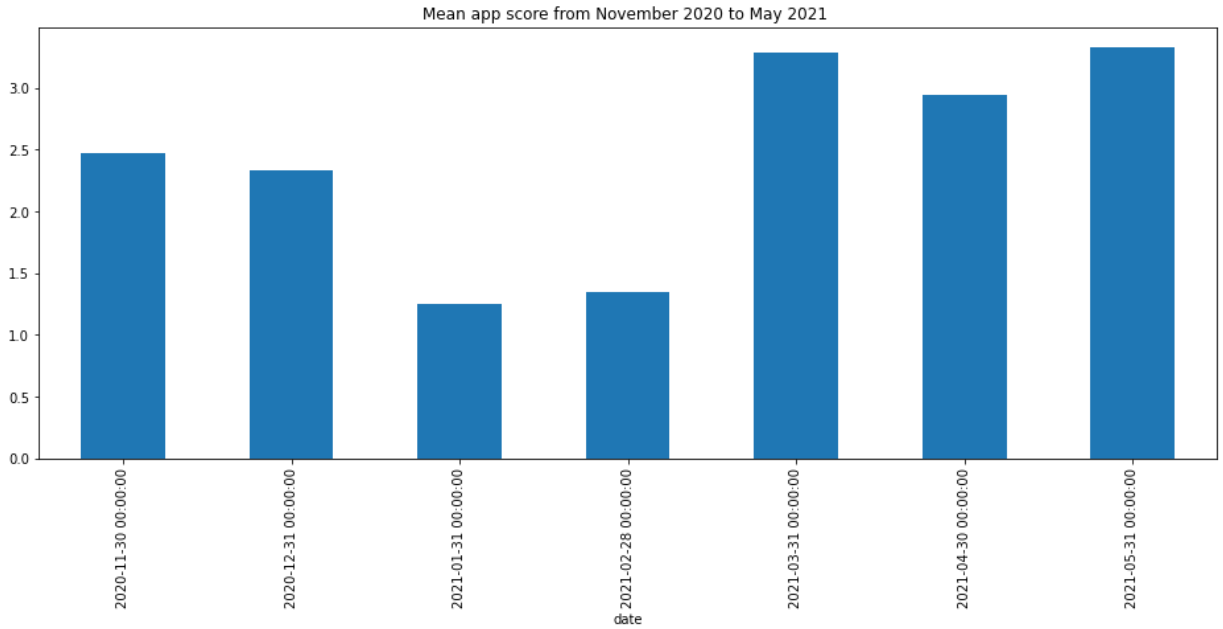
date	score	thumbsUpCount
2020-11-30	55	55
2020-12-31	135	135
2021-01-31	2107	2107
2021-02-28	1574	1574
2021-03-31	430	430
2021-04-30	486	486
2021-05-31	213	213

Monthly Resampling Graphs

In [10]:
```
#Visualize resampling data
%matplotlib inline
title='Count of Scores from November 2020 to May 2021'
score_df.resample (rule='M').count().plot.bar(figsize =[16,6],title=title);
```



In [11]:
```
#Visualize mean scores on a monthly basis
%matplotlib inline
title='Mean app score from November 2020 to May 2021 '
score_df['score'].resample (rule='M').mean().plot.bar(figsize =[16,6],title=title
```



In [12]:
```
#The main trend we see a lare drop in average app scores from January to February
#The drop in mean score correlates to user's sentiment on the app during the GME
#dogpiling and alternate accounts is unknown. Nevertheless, comments from these p
#user sentiment.
```

In [13]:
```
#Create column for 30 day mean scores
score_df['30 Day Mean Score']= score_df['score'].rolling (window = 30).mean()
```

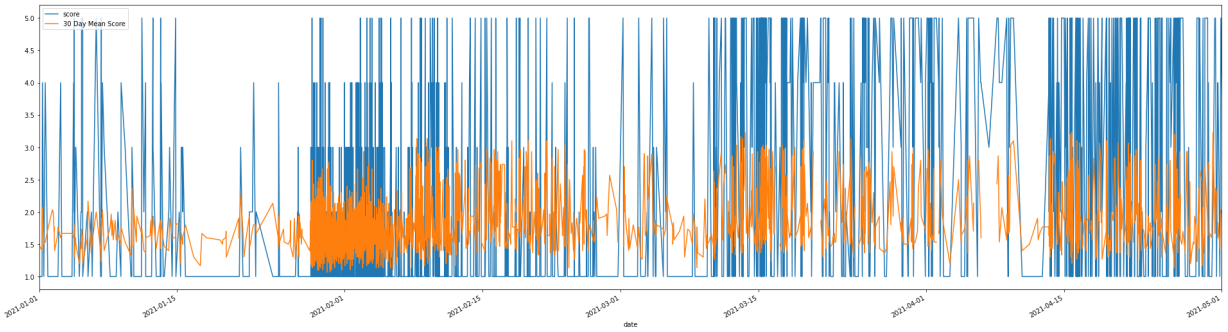
In [14]:
```
#Verify output
score_df
```

Out[14]:

date	score	thumbsUpCount	30 Day Mean Score
2021-02-08 18:06:41	1	1	NaN
2021-03-22 04:51:31	2	0	NaN
2021-02-18 09:39:06	2	1	NaN
2021-03-16 22:33:59	1	0	NaN
2021-03-21 07:23:13	5	0	NaN
...	...	...	...
2021-01-28 17:33:50	1	0	1.733333
2021-01-11 10:05:25	5	2	1.733333
2021-02-02 16:38:36	1	0	1.733333
2021-01-29 13:35:00	1	0	1.733333
2021-01-30 05:40:22	1	0	1.700000

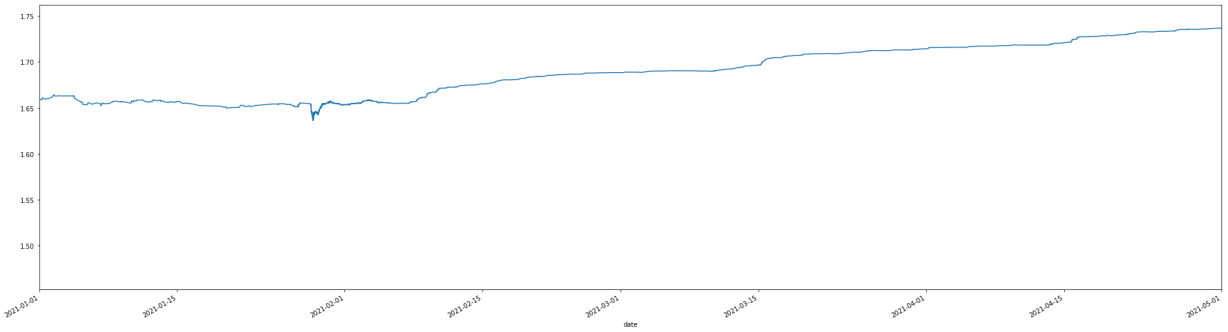
5000 rows × 3 columns

In [15]:
```
#Create plot comparing scores to 30 day mean
score_df[['score','30 Day Mean Score']].plot (figsize=(33,9),xlim=['2021-01-01','
```



In [16]:
```
#While score is much more volatile, due to the reports being sampled, we see a lo
#amount of 0 scores around and after Jan 15th.
```

In [82]:
```
score_df['30 Day Mean Score'].expanding().mean().plot(figsize=(33,9), xlim=['2021
```



In [18]:
```
#Our expanding graph also reveals a low point right at the end of January 2021, w
#deadline of the short squeeze during the GME event.
```

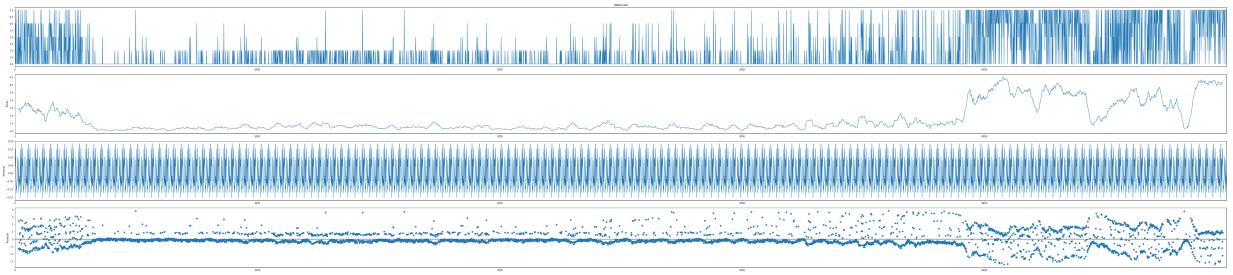
ETS Decomposition

In [19]:
```
from statsmodels.tsa.seasonal import seasonal_decompose
```

In [20]:
```
score_df.sort_index(inplace=True)
```

In [21]:
```
result = seasonal_decompose(score_df['score'].values, model = 'additive', period=
```

In [22]:
```
#change the plot parameters
from pylab import rcParams
rcParams['figure.figsize']=72,16
result.plot();
```



In [23]:
```
#Our observational and trend graphs show an interesting distinction as compared t
#trend graphs show that there was indeed an influx of negative reviews for the ap
#the seasonality graph shows that this did not significatly alter seasonality.

#Interestingly, our residuals graphs shows a relatively consistent residual count
#residuals begin to fluctuate wildly. What this means is that there is a large fl
#compared to the model's values. As this relates to scores later in the sample, i
#anticipated by the model. Whether this is due to the Google Play Store attemptir
#boosted positive reviews, we cannot state, but the low-level residuals for the J
```

Exponential Weighted Moving Average and Exponential Smoothing Analysis

In [ ]:
```
#As seasonality did not see major changes in our ETS composition graph, we decide
#and Double Exponential Smoothing analysis.
```

In [24]:
```
#Simple Exponetial Smoothing
from statsmodels.tsa.holtwinters import SimpleExpSmoothing
```

In [25]:
```
#Set span and alpha parameters
span = 12
alpha = 2/ (span+1)
```

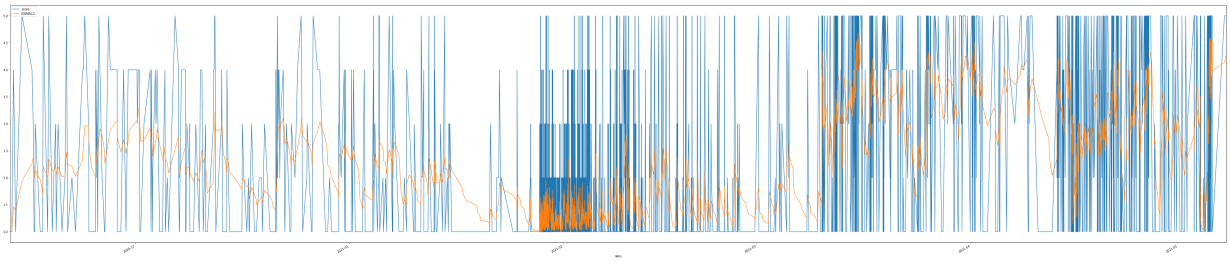
In [99]:
```
#Create column to hold Estimated Weighted Moving Average
score_df['EWMA12'] = score_df['score'].ewm(alpha=alpha, adjust = False).mean()

score_df.head()
```

Out[99]:

date	score	thumbsUpCount	30 Day Mean Score	EWMA12	SES12	DESadd12	TESadd12
2020-11-12 21:18:39	1	4	1.500000	1.000000	1.000000	2.214132	NaN
2020-11-13 02:58:57	1	3	1.500000	1.000000	1.000000	2.135710	NaN
2020-11-13 09:10:01	4	39	1.400000	1.461538	1.461538	2.257098	NaN
2020-11-13 16:25:54	1	5	1.700000	1.390533	1.390533	2.175887	NaN
2020-11-14 14:34:28	5	43	1.633333	1.945835	1.945835	2.359574	NaN

In [88]:
```
#Graph score with Exponential Weighted Moving Average
score_df[['score','EWMA12']].plot(figsize=(72,16)).autoscale(axis='x',tight=True)
```



In [89]:
```
#Apply Simple Exponential Smoothing to our score column
model = SimpleExpSmoothing(score_df['score'])
```

```
D:\Program Files\lib\site-packages\statsmodels\tsa\base\tsa_model.py:216: Value
Warning: A date index has been provided, but it has no associated frequency inf
ormation and so will be ignored when e.g. forecasting.
  warnings.warn('A date index has been provided, but it has no'
```

In [90]:
```
#Fit the model according to our previously set alpha parameter
model.fit(smoothing_level=alpha,optimized=False)
```

Out[90]:
```
<statsmodels.tsa.holtwinters.HoltWintersResultsWrapper at 0x2566cb20>
```

In [91]:
```
#Create fitted_model opbject to store the fitted model
fitted_model = model.fit(smoothing_level=alpha,optimized=False)
```

In [92]:
```
#because values are shifted, we need to shift them back by 1
fitted_model.fittedvalues.shift(-1)
```

Out[92]:
```
date
2020-11-12 21:18:39    1.000000
2020-11-13 02:58:57    1.000000
2020-11-13 09:10:01    1.461538
2020-11-13 16:25:54    1.390533
2020-11-14 14:34:28    1.945835
                           ...
2021-05-05 23:10:45    4.439294
2021-05-05 23:32:08    4.525556
2021-05-06 00:09:26    3.983163
2021-05-07 17:55:47    4.139599
2021-05-07 23:54:44         NaN
Length: 5000, dtype: float64
```

In [93]:
```
#Create a column for our shifted values
score_df['SES12'] = fitted_model.fittedvalues.shift(-1)
score_df.head()
```

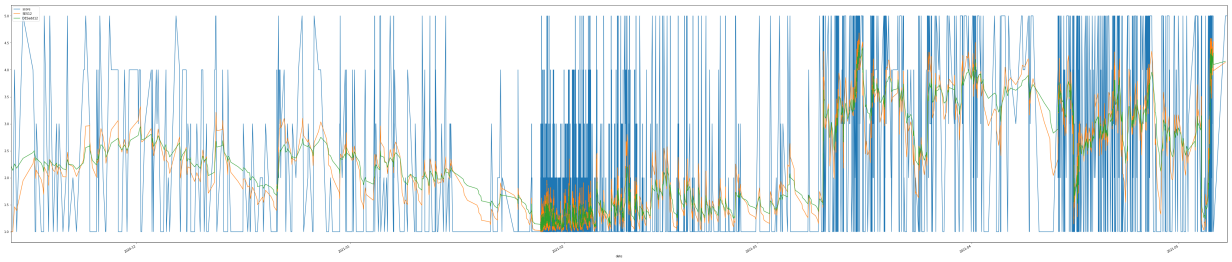
Out[93]:

date	score	thumbsUpCount	30 Day Mean Score	EWMA12	SES12	DESadd12	TESadd12
2020-11-12 21:18:39	1	4	1.500000	1.000000	1.000000	1.519474	NaN
2020-11-13 02:58:57	1	3	1.500000	1.000000	1.000000	1.528449	NaN
2020-11-13 09:10:01	4	39	1.400000	1.461538	1.461538	1.531687	NaN
2020-11-13 16:25:54	1	5	1.700000	1.390533	1.390533	1.550546	NaN
2020-11-14 14:34:28	5	43	1.633333	1.945835	1.945835	1.564904	NaN

In [94]:
```
#import Exponential Smoothing for Exponential Smoothing Analysis
from statsmodels.tsa.holtwinters import ExponentialSmoothing
```

In [ ]:
```
### Create column with Double Exponential Smoothing applied to our score column
score_df['DESadd12'] = ExponentialSmoothing(score_df['score'], trend='add').fit()
```

In [98]:
```
#Graph score with Simple Exponential Smoothing Average and Double Exponential Smo
score_df[['score','SES12','DESadd12']].plot(figsize=(72,16)).autoscale(axis='x',t
```



In [35]:
```
#Our Double Exponential Smoothing following the additive method roughly follows c
#it is forecasting above the average for score, seemingly focusing on the higher
#it captures the end of January-February period well, which is the specific timef

#This could be due to the relatively small amount of data from November to Decemb
#forecast in our interested timeframe is adequate.

#While a triple-exponential smoothing forecast could potentially further enhance
#within a seasonal format, being a random sample of reviews from the Google Play
```

Review Text Analysis

In [39]:
```
#Place text into separate dataframe for cleaning
content_df = review_df['content']
content_df
```

Out[39]:
```
0       Making the decision to uninstall and post this...
1       Confusing. Navigation visuals unclear, very un...
2       It's very hard to get helped when you need it....
3       I've lost faith in this companies business pra...
4       I was using other investment apps prior to swi...
                              ...
4995    It's a great interface, but can't recommend th...
4996    haven't tried pulling out money yet but so far...
4997    Not a good platform for stocks or cryptocurren...
4998    After multiple attempts to contact the adminis...
4999    Many outages, company shuts down trading of lu...
Name: content, Length: 5000, dtype: object
```

In [40]:
```
#Import Beautiful Soup to clean HTML tags
import bs4
from bs4 import BeautifulSoup
```

In [41]:
```
#Use lambda function to strip HTML tags
content_df = content_df.apply(lambda x: bs4.BeautifulSoup(x, 'lxml').get_text())

#Verify corpus_df output
content_df
```

Out[41]:
```
0       Making the decision to uninstall and post this...
1       Confusing. Navigation visuals unclear, very un...
2       It's very hard to get helped when you need it....
3       I've lost faith in this companies business pra...
4       I was using other investment apps prior to swi...
                              ...
4995    It's a great interface, but can't recommend th...
4996    haven't tried pulling out money yet but so far...
4997    Not a good platform for stocks or cryptocurren...
4998    After multiple attempts to contact the adminis...
4999    Many outages, company shuts down trading of lu...
Name: content, Length: 5000, dtype: object
```

In [42]:
```
#Place text into specific corpus object for cleaning
cleaned_corpus = content_df
cleaned_corpus
```

Out[42]:
```
0       Making the decision to uninstall and post this...
1       Confusing. Navigation visuals unclear, very un...
2       It's very hard to get helped when you need it....
3       I've lost faith in this companies business pra...
4       I was using other investment apps prior to swi...
                              ...
4995    It's a great interface, but can't recommend th...
4996    haven't tried pulling out money yet but so far...
4997    Not a good platform for stocks or cryptocurren...
4998    After multiple attempts to contact the adminis...
4999    Many outages, company shuts down trading of lu...
Name: content, Length: 5000, dtype: object
```

Review Text Preprocessing

In [43]:
```
#define WordPunctTokenizer and stopwords objects
wpt=nltk.WordPunctTokenizer()
stop_words=nltk.corpus.stopwords.words('english')
```

In [44]:
```
#Define normalize_corpus method to apply preprocessing, including regex formattir

def normalize_doc (doc):
    doc=re.sub('[^a-zA-Z\s0-9]+',"", doc, re.I|re.A)
    doc=doc.lower()
    doc=doc.strip()
    tokens=wpt.tokenize(doc)
    filtered_tokens=[token for token in tokens if token not in stop_words]
    doc=' '.join(filtered_tokens)
    return doc
```

In [45]:
```
#Create pipeline to normalize corpus
normalize_corpus=np.vectorize(normalize_doc)
```

In [46]:
```
#Create normalized corpus dataframe and apply normalization pipeline to corpus
normalized_corpus = normalize_corpus(cleaned_corpus)

#Verify corpus output
normalized_corpus
```

Out[46]:
```
array(['making decision uninstall post easy one loyal follower since 2015 watch
ed rh become phenomenal app difficult find well developed trustworthy app thoug
ht appalled beyond words handling recent incidents last time waste duplicity da
nte alighieri wrote darkest places hell reserved times great moral crisis maint
ain neutrality',
       'confusing navigation visuals unclear unattractive app',
       'hard get helped need got locked account respond emails week request twi
tter finally got sammi solved problem day without intervention would 1 star rev
iew would stinker poor support ok game something service keep thousands dollars
ok need minimum live chat phone option without cant recommend',
       ...,
       'good platform stocks cryptocurrencies lock buying certain stocks missed
ome really good investments also important dont allow transfer cryptocurrencies
wallets makes basically useless overall bad app disappointed',
       'multiple attempts contact administrator last year given trying access o
ld account service unbearable must clear consumers consumers product details so
ld wall street funds order profit us tune billions dollars wanted seat table in
accessible use robinhood',
       'many outages company shuts trading lucrative stocks protect hours tradi
ng hours better platforms wouldnt create taxable event move id already gone sin
ce wont let directly transfer owned crypto another service stay till im long le
ast skip one point many apps reliable arent run companies liquidity problems'],
      dtype='<U916')
```

Create CV Matrix

In [47]:
```
#Import count vectorizer from sklearn
from sklearn.feature_extraction.text import CountVectorizer

#Create cv object to vectorize corpus
cv = CountVectorizer(min_df=0., max_df=1.)

#Create cv_matrix object to create matrix from corpus
cv_matrix=cv.fit_transform(normalized_corpus)
cv_matrix
```

Out[47]:
```
<5000x8503 sparse matrix of type '<class 'numpy.int64'>'
        with 115991 stored elements in Compressed Sparse Row format>
```

In [48]:
```
#Create array representation to verify output
cv_matrix=cv_matrix.toarray()
cv_matrix
```

Out[48]:
```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

In [49]:
```
#Create vocab object to hold important feature names, and create dataframe to dis
vocab=cv.get_feature_names()
pd.DataFrame(cv_matrix,columns=vocab)
```

Out[49]:

	000	001	005	0061	008	009	01	010	0100	01012020	...	youtube	youtubers	youve	y
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4995	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
4996	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
4997	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
4998	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
4999	0	0	0	0	0	0	0	0	0	0	...	0	0	0	

5000 rows × 8503 columns

TF-IDF

In [50]:
```
#Import TfidfTransformer from sklearn
from sklearn.feature_extraction.text import TfidfTransformer
```

In [51]:
```
#Apply TFidf to matrix

tft=TfidfTransformer(norm='l2', use_idf=True)
tft_matrix=tft.fit_transform(cv_matrix)
tft_matrix=tft_matrix.toarray()
vocab=cv.get_feature_names()
pd.DataFrame(np.round(tft_matrix,2), columns=vocab)
```

Out[51]:

	000	001	005	0061	008	009	01	010	0100	01012020	...	youtube	youtubers	youve	
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4995	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0
4996	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0
4997	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0
4998	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0
4999	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0

5000 rows × 8503 columns

In [52]:
```
#Import cosine_similarity for document similarity
from sklearn.metrics.pairwise import cosine_similarity
```

In [53]:
```
#Create similarity_matrix object to calculate cosine similarity, passing tf-idf m
similarity_matrix = cosine_similarity(tft_matrix)

#Create dataframe to store similarity matrix data
similarity_df = pd.DataFrame(similarity_matrix)


#Confirm dataframe output
similarity_df
```

Out[53]:

	0	1	2	3	4	5	6	7	8	
0	1.000000	0.007247	0.000000	0.004608	0.011926	0.004413	0.000000	0.015871	0.000000	0
1	0.007247	1.000000	0.000000	0.006176	0.000000	0.005915	0.000000	0.000000	0.000000	0
2	0.000000	0.000000	1.000000	0.016532	0.058959	0.023375	0.054533	0.000000	0.000000	0
3	0.004608	0.006176	0.016532	1.000000	0.014725	0.118622	0.016511	0.102544	0.000000	0
4	0.011926	0.000000	0.058959	0.014725	1.000000	0.014101	0.023011	0.000000	0.000000	0
...	...	...	...	...	...	...	...	...	...	
4995	0.022881	0.010802	0.046929	0.047642	0.000000	0.045623	0.000000	0.000000	0.000000	0
4996	0.031352	0.006689	0.032556	0.013507	0.000000	0.014360	0.000000	0.051222	0.000000	0
4997	0.004277	0.005732	0.000000	0.086879	0.000000	0.025581	0.030648	0.038320	0.000000	0
4998	0.012705	0.000000	0.031535	0.009229	0.041257	0.052339	0.000000	0.014579	0.000000	0
4999	0.018832	0.000000	0.006973	0.053526	0.037163	0.045546	0.011941	0.014930	0.023248	0

5000 rows × 5000 columns

In [54]:
```
#Import dendogram and linkage packages from scipy
from scipy.cluster.hierarchy import dendrogram, linkage
```

In [55]:
```
#Create object to create linkage array
linkage_array = linkage (similarity_matrix, 'ward')
linkage_array
```

Out[55]:
```
array([[2.62000000e+02, 1.05600000e+03, 0.00000000e+00, 2.00000000e+00],
       [3.56200000e+03, 5.00000000e+03, 0.00000000e+00, 3.00000000e+00],
       [2.70000000e+01, 4.49900000e+03, 0.00000000e+00, 2.00000000e+00],
       ...,
       [9.99100000e+03, 9.99500000e+03, 3.57381531e+01, 3.57100000e+03],
       [9.99200000e+03, 9.99600000e+03, 4.26120397e+01, 4.77300000e+03],
       [9.99400000e+03, 9.99700000e+03, 7.59261055e+01, 5.00000000e+03]])
```

In [56]:
```
#Create document cluster dataframe
pd.DataFrame(linkage_array, columns=['Documents\Cluster 1','Documents\Cluster 2',
                          'Distance', 'Cluster Size'], dtype='object')
```

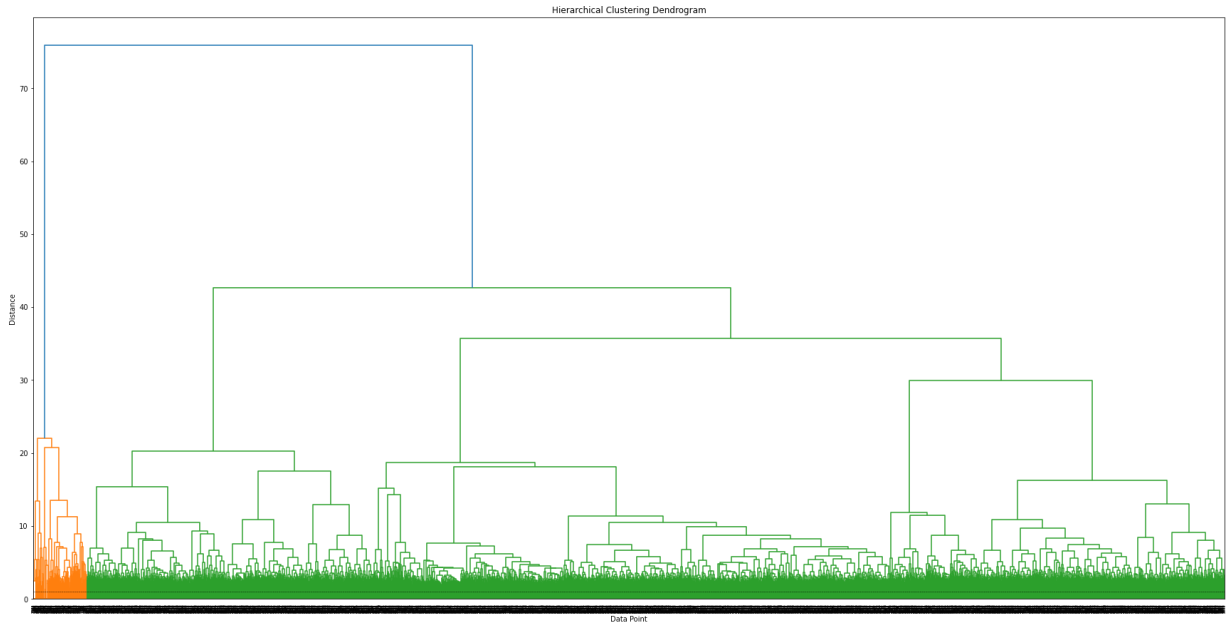
Out[56]:

	Documents\Cluster 1	Documents\Cluster 2	Distance	Cluster Size
0	262	1056	0	2
1	3562	5000	0	3
2	27	4499	0	2
3	3685	3790	0	2
4	1392	1741	0	2
...	...	...	...	...
4994	9983	9993	22.0452	227
4995	9980	9988	29.9619	1404
4996	9991	9995	35.7382	3571
4997	9992	9996	42.612	4773
4998	9994	9997	75.9261	5000

4999 rows × 4 columns

In [57]:
```
#Define figsize, title, and labels for dendogram
plt.figure (figsize = (30,15))
plt.title ('Hierarchical Clustering Dendrogram')
plt.xlabel('Data Point')
plt.ylabel('Distance')


#print dendogram with similarity matrix as the argument
dendrogram(linkage_array)
plt.axhline(y=1.0,c='k',ls='--',lw=0.5);
```



In [58]:
```
#Our dendogram shows two distinct categories, showing a high amount of similarity

#This is expected, and much less diversity was expected between this and the redo
#commentary, which is focused on the assessment of the apps.
```

In [59]:
```
#Import fcluster package from scipy
from scipy.cluster.hierarchy import fcluster
```

In [60]:
```
#Create cluster label dataframe
max_dist = 1.0
cluster_lables = fcluster(linkage_array,max_dist, criterion='distance')
cluster_lables = pd.DataFrame(cluster_lables,columns=['ClusterLabel'])
pd.concat([content_df,cluster_lables],axis=1)
```

Out[60]:

	content	ClusterLabel
0	Making the decision to uninstall and post this...	2099
1	Confusing. Navigation visuals unclear, very un...	1739
2	It's very hard to get helped when you need it....	330
3	I've lost faith in this companies business pra...	3968
4	I was using other investment apps prior to swi...	2570
...	...	...
4995	It's a great interface, but can't recommend th...	1559
4996	haven't tried pulling out money yet but so far...	3872
4997	Not a good platform for stocks or cryptocurren...	3995
4998	After multiple attempts to contact the adminis...	585
4999	Many outages, company shuts down trading of lu...	4527

5000 rows × 2 columns

LSA Topic Modelling

In [61]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import TruncatedSVD
```

In [62]:
```
#Create an instance of vectorizer and fit corpus data into object X
vectorizer = TfidfVectorizer ()
X = vectorizer.fit_transform (normalized_corpus)
```

In [63]:
```
#Create lsa object to apply SVD to vectorized corpus
lsa = TruncatedSVD (n_components=20, n_iter=750)
lsa.fit(X)
```

Out[63]:
```
TruncatedSVD(n_components=20, n_iter=750)
```

In [64]:
```
#check first row of lsa object to confirm output
row1 = lsa.components_[0]
row1
```

Out[64]:
```
array([0.00086823, 0.00037376, 0.00021279, ..., 0.00055804, 0.00046552,
       0.00127111])
```

In [65]:
```
#Create terms object to browse feature names
terms = vectorizer.get_feature_names()
terms
```

Out[65]:
```
['000',
 '001',
 '005',
 '0061',
 '008',
 '009',
 '01',
 '010',
 '0100',
 '01012020',
 '01272021',
 '012721',
 '0128',
 '01282021',
 '012821',
 '01292021',
 '02',
 '0203',
 '020921',
```

In [66]:
```
#Use a loop to find concepts and terms associated with them

#enumerate returns index and row tuples in lsa object
for i,comp in enumerate(lsa.components_):
    componentTerms = zip(terms,comp)
    #sort component terms, by concept value,lambda x (x correpsnding tuples) and
    sortedTerms = sorted(componentTerms, key=lambda x:x[1], reverse=True) #decend
    sortedTerms = sortedTerms[:15] #select 15 most important terms within a conce
    print ("\nConcept",i,":")
    for term in sortedTerms:
        print (term)
```

```
Concept 0 :
('app', 0.3144863850177459)
('use', 0.23142005950658243)
('easy', 0.22907331881239587)
('money', 0.18986239186117615)
('stocks', 0.1596200739851341)
('account', 0.14825150806501033)
('buy', 0.14625758361322955)
('trading', 0.1328804378542925)
('great', 0.13202843117925825)
('stock', 0.12387783153976112)
('get', 0.120175862849023)
('robinhood', 0.1194372419608814)
('market', 0.113457393844317)
('dont', 0.1091759224886768)
('service', 0.1080695364114291)

Concept 1 :
```

In [67]:
```
#create an empty dictionary to hold concepts and associated words
concept_words = {}
```

In [68]:
```
#Loop through again to store concepts and associated words in concept_words dicti
for i,comp in enumerate(lsa.components_):
    componentTerms = zip(terms,comp)

    sortedTerms = sorted(componentTerms, key=lambda x:x[1], reverse=True)
    sortedTerms = sortedTerms[:15]
    concept_words["Concept " + str(i)] = sortedTerms
```

In [69]:
```
#View concepts
concept_words
```

Out[69]:
```
{'Concept 0': [('app', 0.3144863850177459),
  ('use', 0.23142005950658243),
  ('easy', 0.22907331881239587),
  ('money', 0.18986239186117615),
  ('stocks', 0.1596200739851341),
  ('account', 0.14825150806501033),
  ('buy', 0.14625758361322955),
  ('trading', 0.1328804378542925),
  ('great', 0.13202843117925825),
  ('stock', 0.12387783153976112),
  ('get', 0.120175862849023),
  ('robinhood', 0.1194372419608814),
  ('market', 0.113457393844317),
  ('dont', 0.1091759224886768),
  ('service', 0.1080695364114291)],
 'Concept 1': [('easy', 0.7151592842858204),
  ('use', 0.4766948769075384),
  ('navigate', 0.15235606613013025),
  ('understand', 0.15015459459591302),
```

In [70]:
```
#Loop through concepts to find if a particular sentence is featured in that conce
for key in concept_words.keys():
    sentence_scores = [] #a list to store all scores for sentences and concepts
    for sentence in normalized_corpus:
        words = nltk.word_tokenize(sentence)
        score = 0
        for word  in words:
            for word_with_score in concept_words[key]:
                if word  == word_with_score[0]:
                    score +=word_with_score[1]
        sentence_scores.append(score)
    print("\n"+key+":")
    for sentence_scores in sentence_scores:
        print (sentence_scores)
```

```
Concept 0:
0.9900745200271458
0.3144863850177459
0.3764969073254625
0.91628653067923
0.2328946358051984
1.092733216017742
0.2730774678294511
0.6201134523041124
0
0.5668040648329198
0.38957265711387207
0.7749797633367241
0
0.6203640426161096
0.40821423683852975
0.120175862849023
0.4604933783189783
```

Word2Vec Modelling

In [71]:
```
#!pip install gensim
#!pip install python-Levenshtein
```

In [72]:
```
#Import packages necessary for Word2Vec

import urllib
from gensim.models import Word2Vec
from nltk.corpus import stopwords
```

In [73]:
```
#Create a list to hold elements of corpus for tokenization

normalized_corpusList = normalized_corpus.tolist()
```

In [74]:
```
#Verify list output

normalized_corpusList
```

Out[74]:
```
['making decision uninstall post easy one loyal follower since 2015 watched r
h become phenomenal app difficult find well developed trustworthy app thought
appalled beyond words handling recent incidents last time waste duplicity dan
te alighieri wrote darkest places hell reserved times great moral crisis main
tain neutrality',
 'confusing navigation visuals unclear unattractive app',
 'hard get helped need got locked account respond emails week request twitter
finally got sammi solved problem day without intervention would 1 star review
would stinker poor support ok game something service keep thousands dollars o
k need minimum live chat phone option without cant recommend',
 'ive lost faith companies business practices claim free trading advertisemen
ts fine print robinhood could bar trade certain stocks even affect portfolio
money invested important could recommend app',
 'using investment apps prior switching robinhood one month paid fees 200 dol
lars several trades hard make difference market bearish',
 'please use broker robinhood often stop trading markets volatile business pr
actices completely opposite name working retail investors currently trying tr
ansfer funds app frozen account cannot buy sell complete garbage company reco
mmend fidelity sofi instead',
```

In [75]:
```
#Import tokenizer from nltk

from nltk.tokenize import word_tokenize


#Loop through corpus list and tokenize words

tokenized_sents = [word_tokenize(i) for i in normalized_corpusList]
for i in tokenized_sents:
        print (i)
```

```
['making', 'decision', 'uninstall', 'post', 'easy', 'one', 'loyal', 'followe
r', 'since', '2015', 'watched', 'rh', 'become', 'phenomenal', 'app', 'difficu
lt', 'find', 'well', 'developed', 'trustworthy', 'app', 'thought', 'appalle
d', 'beyond', 'words', 'handling', 'recent', 'incidents', 'last', 'time', 'wa
ste', 'duplicity', 'dante', 'alighieri', 'wrote', 'darkest', 'places', 'hel
l', 'reserved', 'times', 'great', 'moral', 'crisis', 'maintain', 'neutralit
y']
['confusing', 'navigation', 'visuals', 'unclear', 'unattractive', 'app']
['hard', 'get', 'helped', 'need', 'got', 'locked', 'account', 'respond', 'ema
ils', 'week', 'request', 'twitter', 'finally', 'got', 'sammi', 'solved', 'pro
blem', 'day', 'without', 'intervention', 'would', '1', 'star', 'review', 'wou
ld', 'stinker', 'poor', 'support', 'ok', 'game', 'something', 'service', 'kee
p', 'thousands', 'dollars', 'ok', 'need', 'minimum', 'live', 'chat', 'phone',
'option', 'without', 'cant', 'recommend']
['ive', 'lost', 'faith', 'companies', 'business', 'practices', 'claim', 'fre
e', 'trading', 'advertisements', 'fine', 'print', 'robinhood', 'could', 'ba
r', 'trade', 'certain', 'stocks', 'even', 'affect', 'portfolio', 'money', 'in
vested', 'important', 'could', 'recommend', 'app']
['using', 'investment', 'apps', 'prior', 'switching', 'robinhood', 'one', 'mo
```

In [76]:
```
#Apply Word2Vec to tokenized sentence corpus

model = Word2Vec(tokenized_sents)
```

In [77]:
```
#Create vocabulary of words stored in "words"

words = model.wv.key_to_index
```

In [78]:
```
#Create object to find words with similar vectors to "robinhood"

robinhoodSimilar = model.wv.most_similar('robinhood', topn = 100)
robinhoodSimilar
```

Out[78]:
```
[('research', 0.9986995458602905),
 ('found', 0.9986862540245056),
 ('things', 0.9986231327056885),
 ('trader', 0.998466432094574),
 ('behind', 0.9984070658683777),
 ('shame', 0.9983689785003662),
 ('made', 0.9983610510826111),
 ('graphs', 0.998245358467102),
 ('investment', 0.9982070922851562),
 ('lots', 0.9981845021247864),
 ('okay', 0.9981842637062073),
 ('broker', 0.9981728792190552),
 ('think', 0.9981148838996887),
 ('hate', 0.9981047511100769),
 ('definitely', 0.9980905652046204),
 ('thought', 0.9980822205543518),
 ('investments', 0.9980626702308655),
 ('bit', 0.9980413317680359),
 ('shady', 0.9979721903800964),
 ('look', 0.9978983998298645),
 ('ok', 0.9978892803192139),
 ('start', 0.9978692531585693),
 ('confusing', 0.9978416562080383),
 ('investor', 0.9977931976318359),
 ('platforms', 0.9977831244468689),
 ('features', 0.9977579712867737),
 ('little', 0.9977553486824036),
 ('actively', 0.997746467590332),
 ('thing', 0.997730553150177),
 ('feel', 0.9977051615715027),
 ('fine', 0.9977037906646729),
 ('analysis', 0.9976663589477539),
 ('learning', 0.997644072059631),
 ('ultimately', 0.99761962890625),
 ('convenient', 0.997602641582489),
 ('events', 0.9975854754447937),
 ('useful', 0.9975832104682922),
 ('halts', 0.9975786209106445),
 ('fidelity', 0.9974715113639832),
 ('seems', 0.9974596500396729),
 ('looking', 0.997456967830658),
 ('larger', 0.9974343180656433),
 ('huge', 0.9974316954612732),
 ('quite', 0.9974055886268616),
 ('webull', 0.997397780418396),
 ('fast', 0.9973866939544678),
 ('technical', 0.9973503351211548),
 ('entry', 0.9973334074020386),
 ('fairly', 0.9973317384719849),
 ('trust', 0.9972484111785889),
 ('decent', 0.9972453713417053),
 ('term', 0.9972132444381714),
```

```
('basic', 0.9971501231193542),
('truly', 0.9971479177474976),
('compared', 0.9971408843994141),
('beginning', 0.9971138834953308),
('cool', 0.99711012840271),
('offer', 0.9970889091491699),
('realize', 0.9970718622207642),
('charts', 0.9970675706863403),
('starting', 0.9970642924308777),
('sad', 0.9970589876174927),
('policies', 0.9970449805259705),
('jump', 0.9970315098762512),
('news', 0.9970235824584961),
('seeing', 0.9970104098320007),
('brokerages', 0.9969995617866516),
('matters', 0.9969889521598816),
('alternative', 0.9969704151153564),
('stuff', 0.9969326853752136),
('job', 0.9969012141227722),
('costs', 0.9968847632408142),
('lists', 0.9968844652175903),
('especially', 0.9968730807304382),
('appreciate', 0.9968593716621399),
('helps', 0.9968581199645996),
('runs', 0.9968558549880981),
('hedgefunds', 0.9968535304069519),
('scam', 0.9968469738960266),
('small', 0.9968175888061523),
('traders', 0.9967871904373169),
('fees', 0.9967703819274902),
('favor', 0.9967601299285889),
('tracking', 0.9967511892318726),
('decisions', 0.9967294335365295),
('everyday', 0.996726393699646),
('tools', 0.9967041611671448),
('game', 0.9966681599617004),
('trash', 0.9966633319854736),
('gives', 0.9966428875923157),
('freezes', 0.9965994358062744),
('details', 0.9965986013412476),
('miss', 0.9965876936912537),
('functionality', 0.9965875148773193),
('mess', 0.9965826869010925),
('serious', 0.9965628981590271),
('financial', 0.9965551495552063),
('easier', 0.9965126514434814),
('ill', 0.9964811205863953),
('competitors', 0.9964807629585266)]
```

In [79]:
```
#Create object to find words with similar vectors to "restrict"

restrictSimilar = model.wv.most_similar('restrict', topn = 100)
restrictSimilar
```

Out[79]:
```
[('allowing', 0.9986704587936401),
 ('purchases', 0.9974376559257507),
 ('stop', 0.9972401261329651),
 ('power', 0.9969561696052551),
 ('allowed', 0.9968785047531128),
 ('allow', 0.996862530708313),
 ('blocked', 0.9965463280677795),
 ('limited', 0.9964619278907776),
 ('amount', 0.9953200817108154),
 ('people', 0.9952033162117004),
 ('trades', 0.9951580762863159),
 ('also', 0.9946290254592896),
 ('manipulate', 0.9944225549697876),
 ('wanted', 0.9943750500679016),
 ('volatile', 0.9940676689147949),
 ('blocking', 0.9936851263046265),
 ('limiting', 0.9933123588562012),
 ('make', 0.9932377934455872),
 ('block', 0.9930251240730286),
 ('higher', 0.9928088188171387),
 ('place', 0.9924230575561523),
 ('orders', 0.9922546148300171),
 ('limits', 0.9921815991401672),
 ('share', 0.9920714497566223),
 ('lower', 0.9918916821479797),
 ('hot', 0.9918306469917297),
 ('users', 0.9916638731956482),
 ('buys', 0.9914857745170593),
 ('fractional', 0.9910605549812317),
 ('amc', 0.9908007979393005),
 ('specific', 0.9904986619949341),
 ('purchasing', 0.9904882907867432),
 ('forcing', 0.9903927445411682),
 ('gme', 0.9902638792991638),
 ('shares', 0.9894583225250244),
 ('securities', 0.9891369342803955),
 ('letting', 0.9890860319137573),
 ('sales', 0.9885898232460022),
 ('currency', 0.9883893132209778),
 ('manipulation', 0.9882313013076782),
 ('prevented', 0.9882082939147949),
 ('volatility', 0.9875568151473999),
 ('restricting', 0.9872667789459229),
 ('bought', 0.9872110486030579),
 ('trade', 0.9871701598167419),
 ('rising', 0.9864186644554138),
 ('high', 0.9863712787628174),
 ('big', 0.986304759979248),
 ('placed', 0.9860326051712036),
 ('restricted', 0.9853887557983398),
 ('lock', 0.9850821495056152),
 ('protect', 0.9849144816398621),
```

```
('particular', 0.9847851991653442),
('preventing', 0.9847779870033264),
('able', 0.9846456050872803),
('making', 0.9845349192619324),
('prices', 0.983408510684967),
('arbitrarily', 0.9833951592445374),
('cost', 0.9833701848983765),
('crypto', 0.983267068862915),
('execute', 0.9831953048706055),
('prevent', 0.9829428791999817),
('sells', 0.9828115105628967),
('control', 0.9824835658073425),
('freeze', 0.9824809432029724),
('manipulating', 0.9824328422546387),
('restrictions', 0.9823839068412781),
('partial', 0.9822335839271545),
('choose', 0.9818054437637329),
('disable', 0.9816427230834961),
('partners', 0.9815371632575989),
('popular', 0.9815130233764648),
('tickers', 0.9814853072166443),
('ability', 0.9814514517784119),
('retail', 0.9812290668487549),
('lose', 0.9812086820602417),
('equities', 0.9810499548912048),
('without', 0.9807297587394714),
('choosing', 0.9804406762123108),
('sale', 0.9802643060684204),
('consent', 0.9801327586174011),
('profit', 0.9799356460571289),
('sold', 0.9798386693000793),
('cryptos', 0.9797217845916748),
('decide', 0.9795334339141846),
('holds', 0.979525625705719),
('outright', 0.9793714880943298),
('value', 0.9791641235351562),
('randomly', 0.9788937568664551),
('selling', 0.9784526228904724),
('freely', 0.9781168699264526),
('hedge', 0.9781138896942139),
('cryptocurrency', 0.9780079126358032),
('gamestop', 0.9779344201087952),
('enough', 0.9777572154998779),
('dogecoin', 0.9776965975761414),
('permission', 0.97762131690979),
('decided', 0.9774969220161438),
('rise', 0.9774318337440491),
('dropping', 0.9774126410484314)]
```

In [80]:
```
#Create object to find words with similar vectors to "legal"

legalSimilar = model.wv.most_similar('legal', topn = 100)
legalSimilar
```

Out[80]:
```
[('servers', 0.9981858134269714),
 ('feature', 0.9981064796447754),
 ('rh', 0.9980984926223755),
 ('hand', 0.9980801343917847),
 ('literally', 0.9980082511901855),
 ('mention', 0.9980055689811707),
 ('cause', 0.9980000257492065),
 ('traffic', 0.997971773147583),
 ('pick', 0.9979649186134338),
 ('man', 0.9979546070098877),
 ('seem', 0.9979474544525146),
 ('plus', 0.9979433417320251),
 ('profits', 0.9979382753372192),
 ('kind', 0.9979217052459717),
 ('whole', 0.9979144930839539),
 ('quickly', 0.9979003667831421),
 ('theyve', 0.9978978633880615),
 ('run', 0.9978746771812439),
 ('gold', 0.997871994972229),
 ('youll', 0.9978672862052917),
 ('show', 0.9978612065315247),
 ('action', 0.9978599548339844),
 ('name', 0.9978439211845398),
 ('50', 0.9978196024894714),
 ('friends', 0.9978094696998596),
 ('happen', 0.9977745413780212),
 ('opportunities', 0.9977638125419617),
 ('actions', 0.997753083705902l),
 ('means', 0.9977161884307861),
 ('rules', 0.9977092742919922),
 ('believe', 0.9976937770843506),
 ('invested', 0.9976813197135925),
 ('portfolio', 0.9976809024810791),
 ('actual', 0.9976783990859985),
 ('save', 0.997671902179718),
 ('attempt', 0.9976691007614136),
 ('screw', 0.9976548552513123),
 ('point', 0.9976547360420227),
 ('delayed', 0.9976452589035034),
 ('fact', 0.9976361989974976),
 ('claim', 0.9976228475570679),
 ('fee', 0.9976142644882202),
 ('everyone', 0.9976012110710144),
 ('working', 0.9976006150245667),
 ('along', 0.997597873210907),
 ('anymore', 0.9975862503051758),
 ('remove', 0.9975846409797668),
 ('keeping', 0.9975727796554565),
 ('arent', 0.9975715279579163),
 ('lawsuit', 0.9975650906562805),
 ('something', 0.9975603818893433),
 ('illegal', 0.9975472688674927),
```

```
('shouldnt', 0.9975422024726868),
('listed', 0.9975420832633972),
('starts', 0.9975416660308838),
('manage', 0.9975360035896301),
('main', 0.9975354075431824),
('matter', 0.9975350499153137),
('thousands', 0.9975333213806152),
('yes', 0.9975252747535706),
('theyre', 0.9975045919418335),
('completely', 0.9974949359893799),
('common', 0.9974948763847351),
('near', 0.9974948167800903),
('today', 0.9974847435951233),
('values', 0.9974758625030518),
('full', 0.9974733591079712),
('crashing', 0.9974692463874817),
('false', 0.9974662661552429),
('wall', 0.9974613785743713),
('thanks', 0.9974578022956848),
('fiasco', 0.9974513649940491),
('turn', 0.9974512457847595),
('capital', 0.9974423050880432),
('future', 0.9974415302276611),
('watch', 0.997438371181488),
('worth', 0.9974216222763062),
('seriously', 0.9974191784858704),
('exact', 0.9974150657653809),
('ridiculous', 0.9974092245101929),
('screen', 0.9974071979522705),
('clearing', 0.9974068999290466),
('happens', 0.9973936080932617),
('maintenance', 0.9973928332328796),
('entire', 0.9973791837692261),
('everytime', 0.9973751306533813),
('shown', 0.9973697066307068),
('gains', 0.9973661303520203),
('shows', 0.9973592758178711),
('etc', 0.9973587393760681),
('changes', 0.9973504543304443),
('exit', 0.9973450899124146),
('chart', 0.9973320364952087),
('customers', 0.9973261952400208),
('losses', 0.9973200559616089),
('wallet', 0.9973114132881165),
('state', 0.9973061084747314),
('missing', 0.9973049163818359),
('class', 0.9973043203353882),
('yesterday', 0.9972869753837585)]
```

In [81]:
```
#Create object to find words with similar vectors to "manipulation"

manipulationSimilar = model.wv.most_similar('manipulation', topn = 100)
manipulationSimilar
```

Out[81]:
```
[('manipulate', 0.996989905834198),
 ('free', 0.9958982467651367),
 ('users', 0.9932639002799988),
 ('limited', 0.9927501082420349),
 ('people', 0.9925035238265991),
 ('limiting', 0.9920622706413269),
 ('allowing', 0.9907973408699036),
 ('trades', 0.9883812069892883),
 ('restrict', 0.9882313013076782),
 ('fractional', 0.9881280660629272),
 ('options', 0.9860175848007202),
 ('purchases', 0.9856432676315308),
 ('stop', 0.9854462742805481),
 ('restricting', 0.9847769141197205),
 ('volatile', 0.9845373630523682),
 ('trade', 0.9841110706329346),
 ('block', 0.9838327169418335),
 ('blocking', 0.9836580753326416),
 ('allowed', 0.9835979342460632),
 ('make', 0.9833129644393921),
 ('limits', 0.9830360412597656),
 ('blocked', 0.9827589988708496),
 ('selling', 0.9824310541152954),
 ('place', 0.9823421835899353),
 ('securities', 0.9821494221687317),
 ('specific', 0.9815951585769653),
 ('investors', 0.9812241196632385),
 ('hot', 0.980461597442627),
 ('amount', 0.9800972938537598),
 ('orders', 0.980005145072937),
 ('buys', 0.9799315929412842),
 ('power', 0.9797795414924622),
 ('big', 0.9796488285064697),
 ('retail', 0.9793391227722168),
 ('purchasing', 0.9791285395622253),
 ('volatility', 0.9789758324623108),
 ('also', 0.9788615703582764),
 ('manipulating', 0.9788486957550049),
 ('ability', 0.9781705141067505),
 ('allow', 0.9777238965034485),
 ('restrictions', 0.9774269461631775),
 ('gme', 0.977261483669281),
 ('lower', 0.9765686988830566),
 ('preventing', 0.976399838924408),
 ('prevented', 0.9763770699501038),
 ('higher', 0.976116418838501),
 ('shares', 0.9757227301597595),
 ('share', 0.9756860733032227),
 ('amc', 0.974727213382721),
 ('prices', 0.9744450449943542),
 ('trading', 0.9741127490997314),
 ('particular', 0.9741124510765076),
```

```
('wanted', 0.9738233685493469),
('forcing', 0.9723541140556335),
('protect', 0.9721150398254395),
('outright', 0.9719778895378113),
('currency', 0.9718220829963684),
('rising', 0.9717763662338257),
('choosing', 0.9713113903999329),
('sales', 0.971178412437439),
('control', 0.9710844159126282),
('prevent', 0.9707391858100891),
('cost', 0.9706709980964661),
('much', 0.9706673622131348),
('arbitrarily', 0.9705997705459595),
('popular', 0.9705654382705688),
('blatant', 0.9699901938438416),
('equities', 0.9696593880653381),
('making', 0.9696574807167053),
('choose', 0.969510018825531),
('lock', 0.9695019721984863),
('crypto', 0.9691848158836365),
('market', 0.9691298007965088),
('high', 0.9689508080482483),
('partial', 0.9687967896461487),
('youre', 0.9687228202819824),
('placed', 0.9685826301574707),
('sells', 0.9685503244400024),
('letting', 0.9684281945228577),
('enough', 0.9675524830818176),
('profit', 0.966770589351654),
('tickers', 0.9664583802223206),
('fair', 0.9663599729537964),
('freely', 0.9661909341812134),
('consent', 0.9659473896026611),
('bought', 0.9657714366912842),
('decided', 0.9656519889831543),
('hedge', 0.9655478596687317),
('allows', 0.9653589725494385),
('cryptocurrency', 0.9653326272964478),
('activity', 0.9653102159500122),
('execute', 0.9652913212776184),
('holds', 0.9652713537216187),
('data', 0.9647678136825562),
('partners', 0.9647514820098877),
('openly', 0.9645499587059021),
('randomly', 0.9643948674201965),
('value', 0.9641870260238647),
('many', 0.9639808535575867),
('halting', 0.9638957381248474)]
```

In [ ]: