Amazon Fine Food Reviews

Dataset: via Kaggle

This dataset consists of reviews of fine foods from amazon.
The data span a period of more than 10 years, including all ~500,000 reviews up to October 2012.
Reviews include product and user information, ratings, and a plain text review. It also includes reviews from all other Amazon categories.

1. Import libraries

In [ ]:
```
import pandas as pd
import numpy as np
from datetime import datetime

import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud

import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from nltk.corpus import stopwords

from tqdm.notebook import tqdm
from pprint import pprint

#Specify type of visualization
plt.style.use('ggplot')
```

2. Read data

But before we getting things started, i want to introduce the data dictionary of the dataset.
According to Kaggle:

- Id: Row Id
- ProductId: Unique identifier for the product
- UserId: Unqiue identifier for the user
- ProfileName: Profile name of the user
- HelpfulnessNumerator: Number of users who found the review helpful

- HelpfulnessDenominator: Number of users who indicated whether they found the review helpful or not
- Score: Rating between 1 and 5
- Time: Timestamp for the review
- Summary: Brief summary of the review
- Text: Text of the review

In [ ]:
```
df = pd.read_csv('Reviews.csv')
df.head(3)
```

Out[ ]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	He
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	

In [ ]:
```
print('Rows:', df.shape[0])
print('Columns:', df.shape[1])
```

```
Rows:  568454
Columns:  10
```

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 568454 entries, 0 to 568453
Data columns (total 10 columns):
 #   Column                  Non-Null Count   Dtype 
---  ------                  --------------   ----- 
 0   Id                      568454 non-null  int64 
 1   ProductId               568454 non-null  object
 2   UserId                  568454 non-null  object
 3   ProfileName             568438 non-null  object
 4   HelpfulnessNumerator    568454 non-null  int64 
 5   HelpfulnessDenominator  568454 non-null  int64 
 6   Score                   568454 non-null  int64 
 7   Time                    568454 non-null  int64 
 8   Summary                 568427 non-null  object
 9   Text                    568454 non-null  object
dtypes: int64(5), object(5)
memory usage: 43.4+ MB
```

3. Clean (if needed)

In [ ]:
```
df.isna().sum()
```

Out[ ]:
```
Id                         0
ProductId                  0
UserId                     0
ProfileName               16
HelpfulnessNumerator       0
HelpfulnessDenominator     0
Score                      0
Time                       0
Summary                   27
Text                       0
dtype: int64
```

In [ ]:
```
df.duplicated().sum()
```

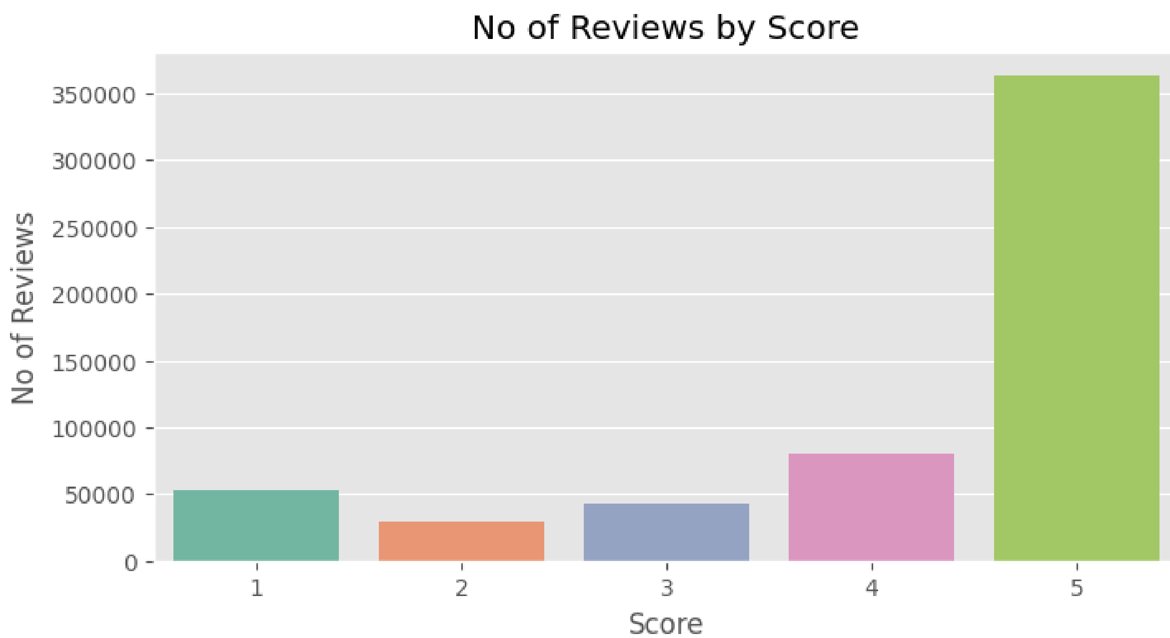
Out[ ]:
```
0
```

The above results are acceptable cuz' I won't use those column 😉

4. EDA

In [ ]:
```
score_grp = df.groupby(by = 'Score').agg(count = ('Score','count')).reset_


plt.figure(figsize=(8,4))
ax = sns.barplot(data = score_grp
                ,x = 'Score'
```

```
                ,y = 'count'
                ,palette='Set2')
ax.set_xlabel('Score')
ax.set_ylabel('No of Reviews')
ax.set_title('No of Reviews by Score')

plt.show()
```



5. Setups for NLTK

In [ ]:
```
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('maxent_ne_chunker')
nltk.download('words')
nltk.download('vader_lexicon')
nltk.download('stopwords')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\MSI\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\MSI\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package maxent_ne_chunker to
[nltk_data]     C:\Users\MSI\AppData\Roaming\nltk_data...
[nltk_data]   Package maxent_ne_chunker is already up-to-date!
[nltk_data] Downloading package words to
[nltk_data]     C:\Users\MSI\AppData\Roaming\nltk_data...
[nltk_data]   Package words is already up-to-date!
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\MSI\AppData\Roaming\nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\MSI\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[ ]:
```
True
```

In [ ]:
```
stop_words = set(stopwords.words('english'))
```

Cuz I ran the polarity_score calculation for some random 'Text'

The score are affected by stopwords, so I decided to remove them

In [ ]:
```
def remove_stopwords(text):
    words = text.split()
    filtered_words = [word for word in words if word.lower() not in stop_w
    return ' '.join(filtered_words)
```

In [ ]:
```
df.columns = ['Id', 'ProductId', 'UserId', 'ProfileName', 'HelpfulnessNume
       'HelpfulnessDenominator', 'Score', 'Time', 'Summary', 'RawText']
```

In [ ]:
```
df['Text'] = df['RawText'].apply(remove_stopwords)
```

I will use a test case to see how it works. I pick the entry along with the index 312.

In [ ]:
```
test = df['Text'][31201]
test
```

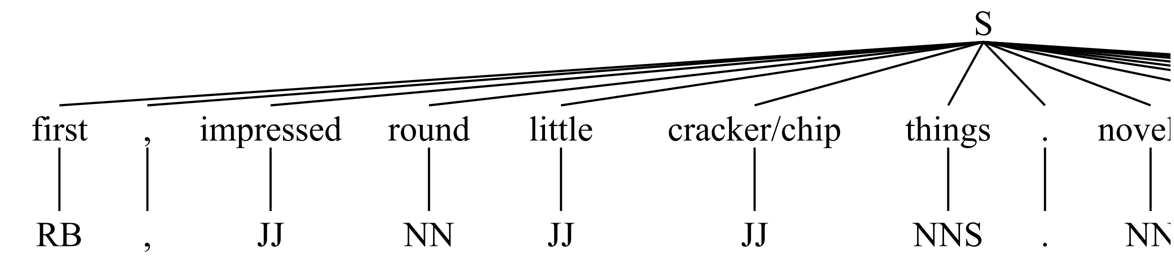
Out[ ]:
```
'first, impressed round little cracker/chip things. novelty wears fast. d
ecided quite gross.'
```

Steps:

1. Lets tokenize it: word_tokenize()
2. Get a tag for each items in the tokens list. Then, represent them as a list of tuple ('word', 'type of word - written in an abbreviation'): pos_tag
3. Visualize list of entities in the tag: ne_chunk()

In [ ]:
```
tokens = nltk.word_tokenize(test)
tags = nltk.pos_tag(tokens)
entities = nltk.ne_chunk(tags)
entities
```

Out[ ]:



In [ ]:
```
pprint(entities)
```

```
Tree('S', [('first', 'RB'), (',', ','), ('impressed', 'JJ'), ('round', 'N
N'), ('little', 'JJ'), ('cracker/chip', 'JJ'), ('things', 'NNS'), ('.',
'.'), ('novelty', 'NN'), ('wears', 'NNS'), ('fast', 'RB'), ('.', '.'), ('de
cided', 'VBD'), ('quite', 'RB'), ('gross', 'JJ'), ('.', '.')])
```

6. VADER Sentiment

In [ ]:
```
sia = SentimentIntensityAnalyzer()
```

In [ ]:
```
pol_list = []

for text in tqdm(df['Text']):
    pol =  sia.polarity_scores(text)
    pol_list.append(pol)
```

```
 0%|          | 0/568454 [00:00<?, ?it/s]
```

☝ Welp, nearly 570k entries takes about 10 mins to complete the loops Pretty slow T.T I think I should run for a smaller sample.

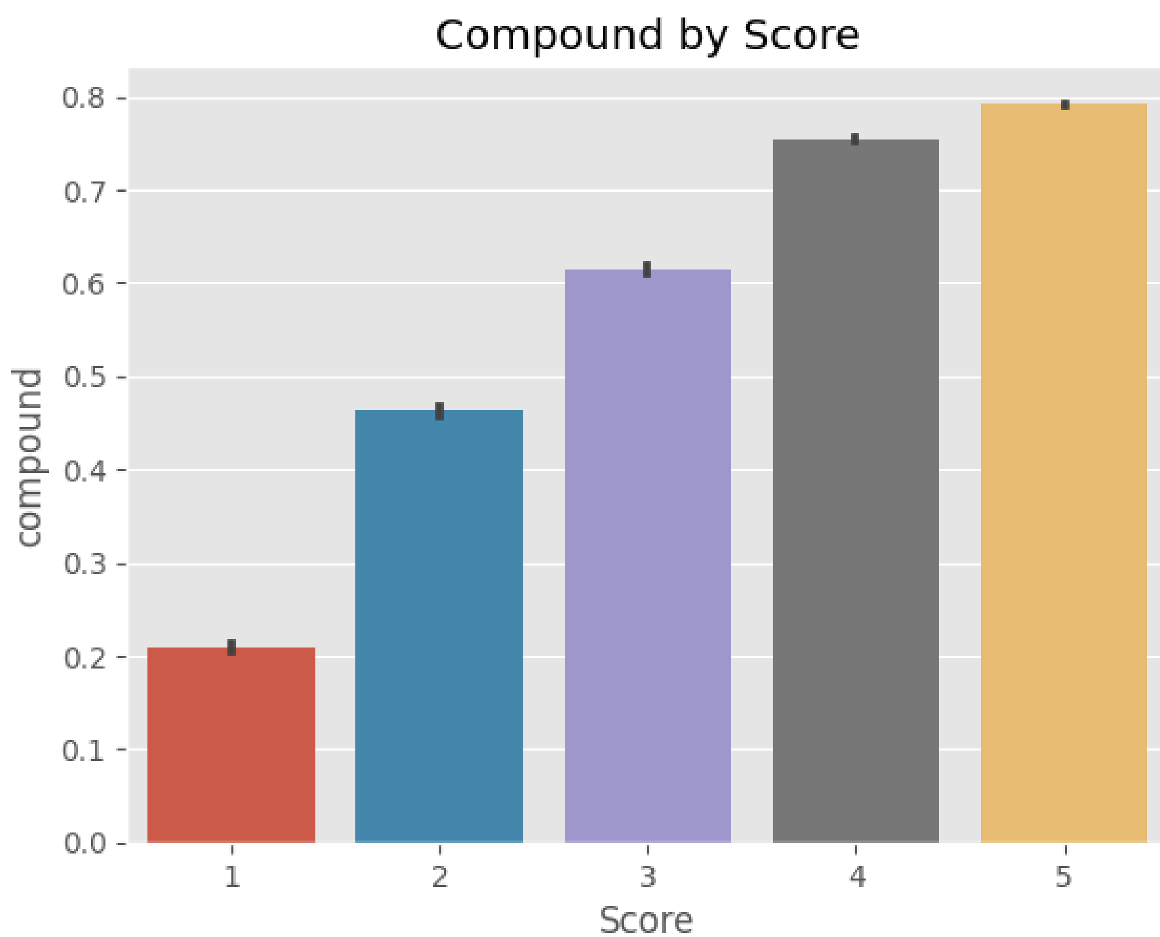
In [ ]:
```
df_pol_score = pd.DataFrame(pol_list)
df_pol_score['Id'] = df['Id']
df_pol_score['Score'] = df['Score']
df_pol_score['Text'] = df['Text']
df_pol_score.head()
```

Out[ ]:

	neg	neu	pos	compound	Id	Score	Text
0	0.000	0.517	0.483	0.9413	1	5	bought several Vitality canned dog food produc...
1	0.129	0.762	0.110	-0.1027	2	1	Product arrived labeled Jumbo Salted Peanuts.....
2	0.165	0.560	0.275	0.8073	3	4	confection around centuries. light, pillowy ci...
3	0.000	1.000	0.000	0.0000	4	2	looking secret ingredient Robitussin believe f...
4	0.000	0.369	0.631	0.9468	5	5	Great taffy great price. wide assortment yummy...

In [ ]:

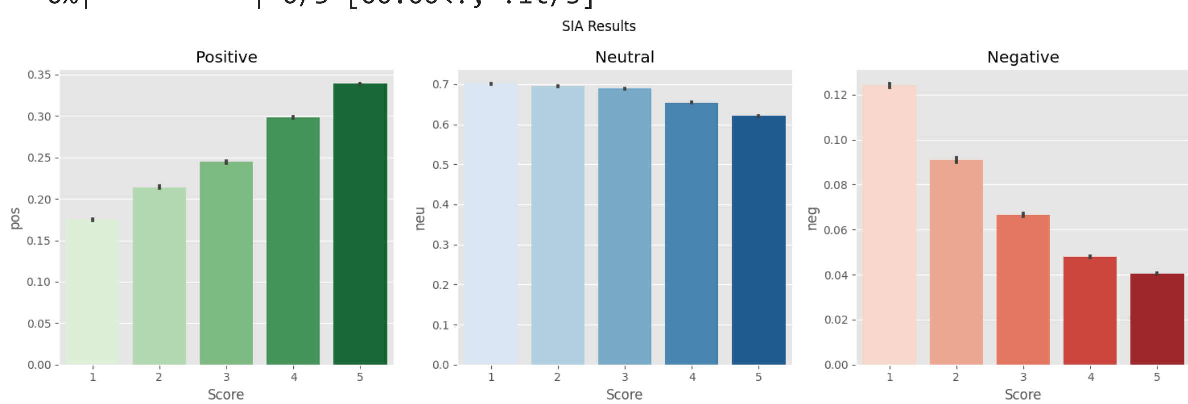
```
ax = sns.barplot(data = df_pol_score
            ,x = 'Score'
            ,y = 'compound')
ax.set_title('Compound by Score')
plt.show()
```



👉 **I suppose that the higher rating score corresponds to a higher compound value.**
Last but not least, I want to check if there is any differentiation in the reviews for each value of 'Score'.

In [ ]:
```
fig, axes = plt.subplots(1,3,figsize = (15,5))

pol_types = ['pos','neu','neg']
pol_names = ['Positive','Neutral','Negative']
color = ['Greens','Blues','Reds']

n = 0

for pol_type in tqdm(pol_types):
    sns.barplot(data = df_pol_score
                ,x = 'Score'
                ,y = pol_type
                ,ax = axes[n]
                ,palette=color[n])
    axes[n].set_title(pol_names[n])
    n+=1

plt.suptitle('SIA Results')

plt.tight_layout()
plt.show()
```

```
 0%|          | 0/3 [00:00<?, ?it/s]
```



👉 **To conclude, VADER gave me results as belows:**

1. For Positive polarity score, the highest value comes to Score = 5.
2. For Neutral polarity score, the values tend to be relatively consistent.
3. For Negative polarity score, the lowest value comes to Score = 1.

7. More EDA after merging VADER into the root dataframe

In [ ]:
```
df_merge_vader = df.merge(df_pol_score, how = 'inner')
df_merge_vader.head(2)
```

Out[ ]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	Hel
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	

In [ ]:
```
df_merge_vader.loc[df_merge_vader['compound'] >= 0.5, 'VaderSegment'] = 'P
df_merge_vader.loc[df_merge_vader['compound'] <= -0.5, 'VaderSegment'] = '
df_merge_vader.loc[(df_merge_vader['compound'] < 0.5) & (df_merge_vader['c
```

In [ ]:
```
df_merge_vader['Date']  = df_merge_vader['Time'].apply(lambda x: datetime.
df_merge_vader['Month'] = df_merge_vader['Date'].dt.month
df_merge_vader['Year']  = df_merge_vader['Date'].dt.year
df_merge_vader.head(2)
```

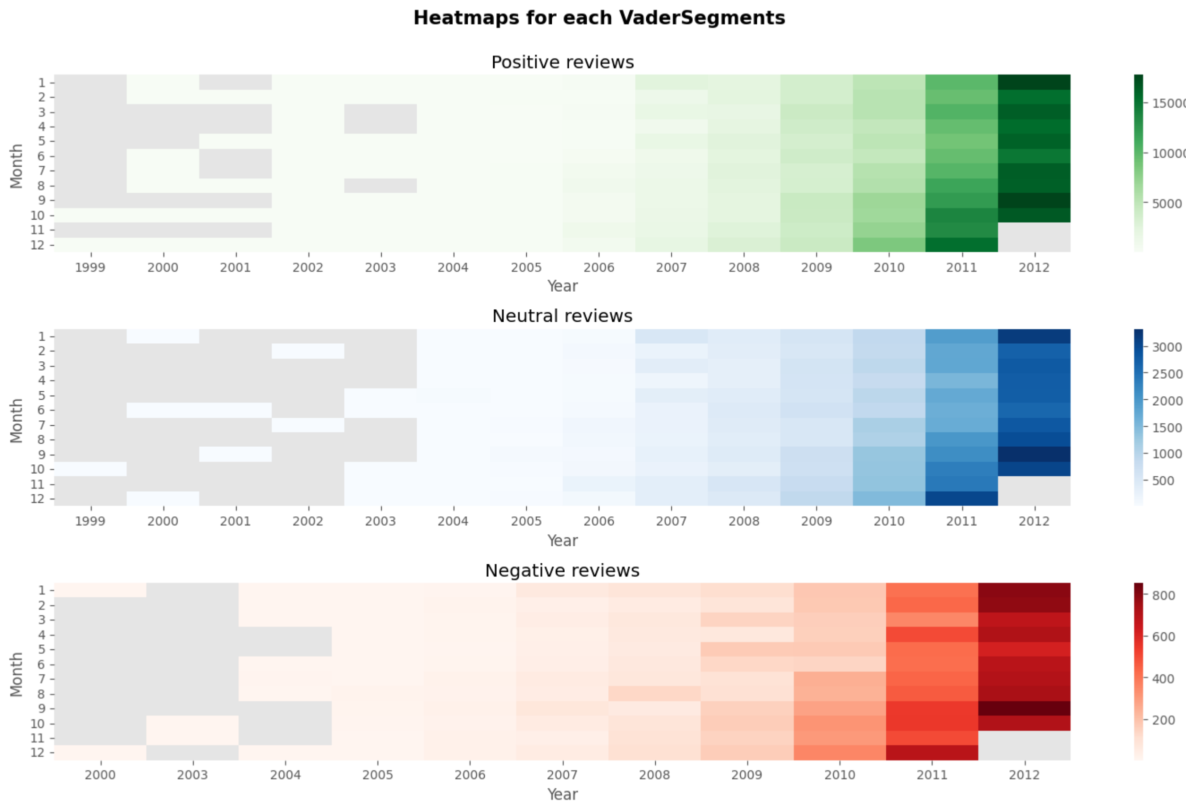
Out[ ]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	Hel
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	

In [ ]:

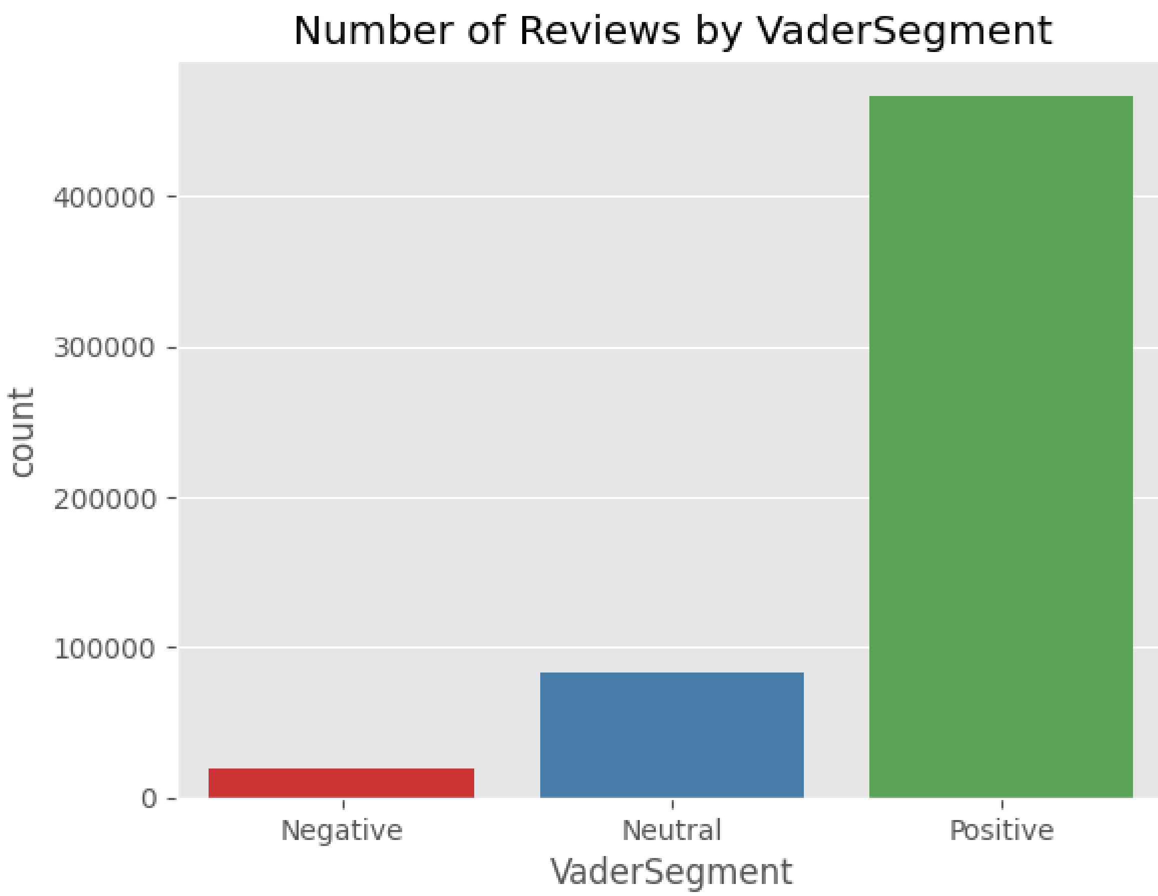
```
fig, axes = plt.subplots(3,1,figsize = (15,9))

plt.suptitle('Heatmaps for each VaderSegments'
             , x = 0.45
             , y = 1
             , fontsize = 15
             , fontweight = 'bold')

segment_list = ['Positive','Neutral','Negative']
cmap_list = ['Greens','Blues','Reds']
name_list = ['Positive reviews','Neutral reviews','Negative reviews']

n = 0

for df_seg in segment_list:
    df_vader_piv = df_merge_vader[df_merge_vader['VaderSegment'] == segmen
                 .pivot_table(values = 'Id'
                             ,index = 'Month'
                             ,columns='Year'
                             ,aggfunc='count')


    sns.heatmap(df_vader_piv
                ,cmap= cmap_list[n]
                ,ax = axes[n])

    axes[n].set_xticklabels(axes[n].get_xticklabels(), rotation=0)
    axes[n].set_yticklabels(axes[n].get_yticklabels(), rotation=0)
    axes[n].set_title(name_list[n])

    n+=1
```

```
plt.tight_layout()
plt.show()
```



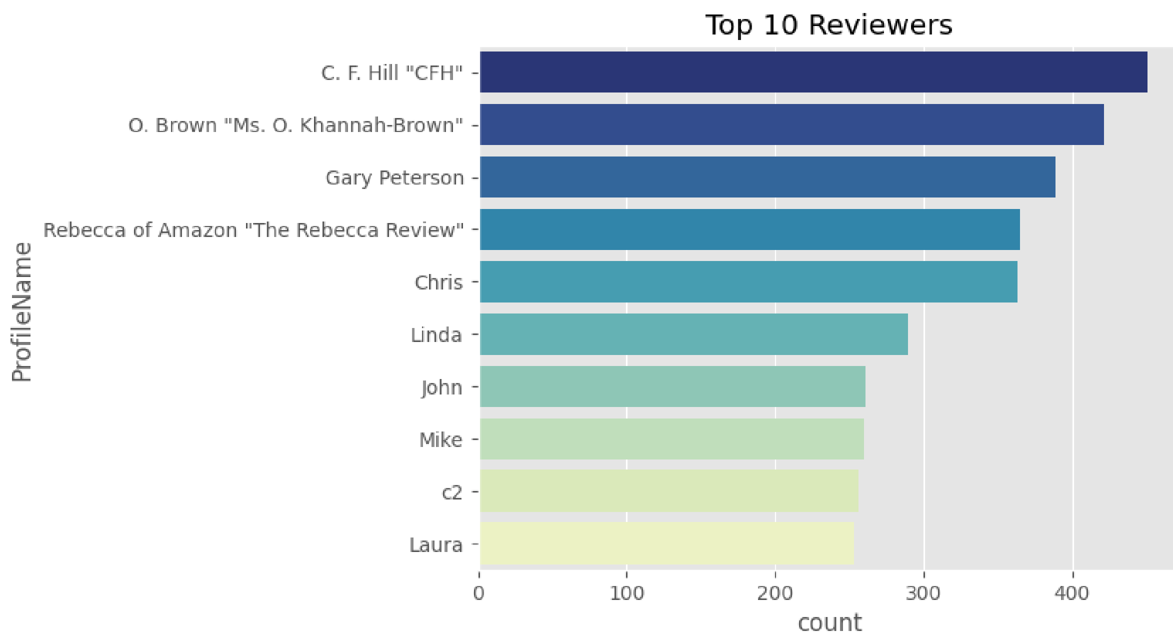
👉 For the most part, reviews exploded in the year 2010 to 2012.

In [ ]:
```
df_vader_count = df_merge_vader.groupby(by = 'VaderSegment').agg(count = (
sns.barplot(data = df_vader_count
            ,x = 'VaderSegment'
            ,y = 'count'
            ,palette='Set1')
plt.title('Number of Reviews by VaderSegment')
plt.show()
```



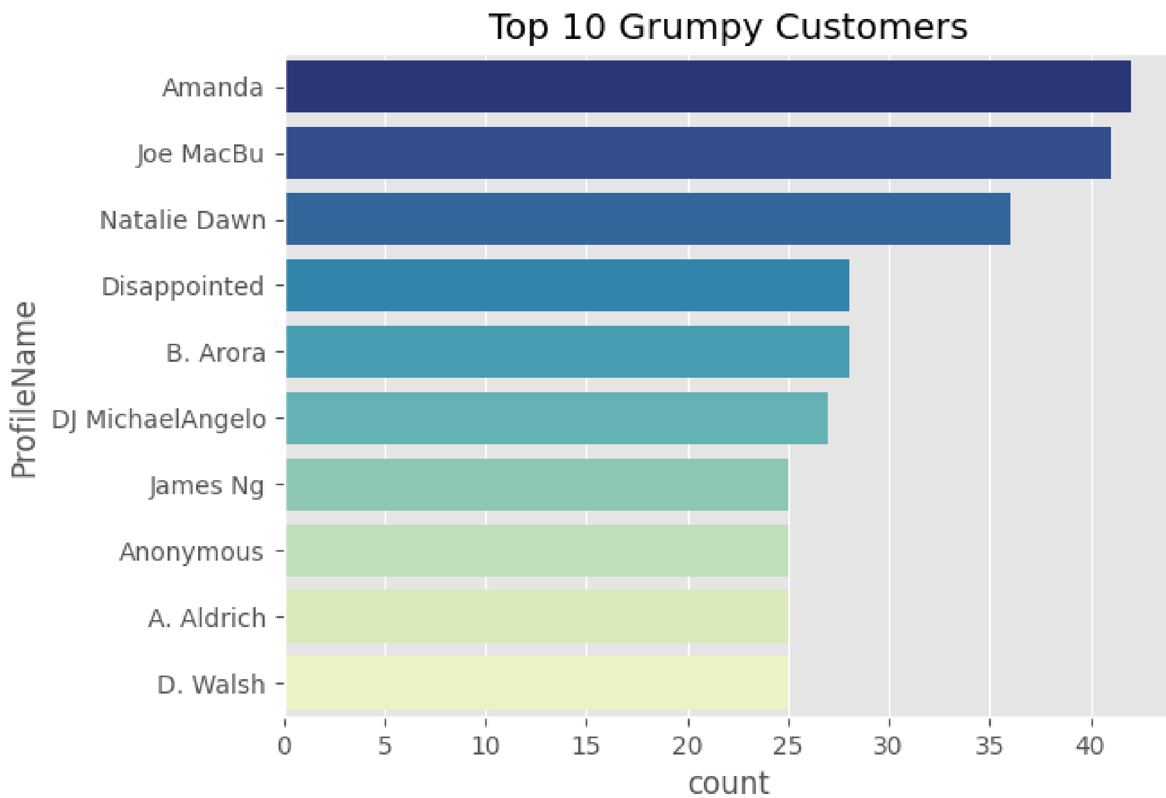
👉 **The plot showed that most of the reviews over the given time periods brought postive vibes.**

In [ ]:
```
top_contributor = df_merge_vader\
            .groupby(by = 'ProfileName').agg(count = ('Id','count'))\
            .reset_index() \
            .sort_values(by = 'count',ascending=False).head(10)

sns.barplot(data = top_contributor
            ,x = 'count'
            ,y = 'ProfileName'
            ,palette='YlGnBu_r')
plt.title('Top 10 Reviewers')
plt.show()
```



👉 **They are top 10 reviewers who wrote the most number of reviews on the platform.**
I think Amazon should honor them with "The reward for the highest valuable customers"

In [ ]:
```
grumpy_customer = df_merge_vader[df_merge_vader['VaderSegment'] == 'Negati
            .groupby(by = 'ProfileName').agg(count = ('Id','count'))\
            .reset_index() \
            .sort_values(by = 'count',ascending=False).head(10)

sns.barplot(data = grumpy_customer
            ,x = 'count'
            ,y = 'ProfileName'
            ,palette='YlGnBu_r')
plt.title('Top 10 Grumpy Customers')
plt.show()
```



👉 **They are top 10 grumpy customers who wrote the most number of negative reviews on the platform.**
CX's Amazon team should take care of them and ask the customer for more details about the issue.
Gathering additional information can help the platform better understand the problem and how to fix it.

In [ ]:

```
vader_type = ['Positive', 'Neutral', 'Negative']
cmap_list = ['Greens', 'Blues', 'Reds']

fig, axes = plt.subplots(3, 1, figsize=(8, 10))

for n, sentiment in enumerate(vader_type):
    text = " ".join(review.strip() for review in df_merge_vader[df_merge_v
                    .str.replace('br',' ')\
                    .str.replace('Br',' ')\
                    )

    wc = WordCloud(width=600, height=200, background_color='white', colorm

    axes[n].imshow(wc, interpolation='bilinear')
    axes[n].set_title(f'Most Frequent Keywords in {sentiment} Reviews')
    axes[n].axis('off')

plt.suptitle('WordClouds for Different Sentiments', fontsize=16)
```

```
plt.tight_layout()
plt.show()
```

👉These Wordclouds showed the most frequently occurring words in each type of review

The End 🤗