In [15]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

plt.style.use('ggplot')

import nltk
```

In [3]:
```
df=pd.read_csv("C:\Data Analytics\Sediment Analysis\Reviews.csv")
```

In [4]:
```
df.head()
```

Out[4]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDen
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	
3	4	B000UA0QIQ	A395BORC6FGVXV	Karl	3	
4	5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	0	

In [5]:
```
df.columns
```

Out[5]:
```
Index(['Id', 'ProductId', 'UserId', 'ProfileName', 'HelpfulnessNumerator',
       'HelpfulnessDenominator', 'Score', 'Time', 'Summary', 'Text'],
      dtype='object')
```

In [6]:
```
df['Text'].values[0]
```

Out[6]:
```
'I have bought several of the Vitality canned dog food products and have f
ound them all to be of good quality. The product looks more like a stew th
an a processed meat and it smells better. My Labrador is finicky and she a
ppreciates this product better than  most.'
```

In [7]:
```
df.shape
```

Out[7]:
```
(568454, 10)
```

In [8]:
```
df=df.head(500)
df.shape
```

Out[8]:
```
(500, 10)
```

EDA

In [9]:
```
df['Score'].value_counts()
```

Out[9]:
```
5    339
4     70
3     37
1     36
2     18
Name: Score, dtype: int64
```

In [10]:
```
ax=df['Score'].value_counts().sort_index().plot(kind='bar', title='Reviews
ax.set_xlabel('Review Stars')
plt.show()
```

NLTK

In [18]:
```
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\sofia\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt.zip.
```

Out[18]:
```
True
```

In [17]:
```
example=df['Text'][50]
print(example)
```

```
This oatmeal is not good. Its mushy, soft, I don't like it. Quaker Oats is
the way to go.
```

In [19]:
```
tokens=nltk.word_tokenize(example)
tokens
```

Out[19]:
```
['This',
 'oatmeal',
 'is',
 'not',
 'good',
 '.',
 'Its',
 'mushy',
 ',',
 'soft',
 ',',
 'I',
 'do',
 "n't",
 'like',
 'it',
 '.',
 'Quaker',
 'Oats',
 'is',
 'the',
 'way',
 'to',
 'go',
 '.']
```

In [20]:
```
tokens[:10]
```

Out[20]:
```
['This', 'oatmeal', 'is', 'not', 'good', '.', 'Its', 'mushy', ',', 'soft']
```

In [21]:
```
nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\sofia\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping taggers\averaged_perceptron_tagger.zip.
```

Out[21]:
```
True
```

In [22]:
```
tagged=nltk.pos_tag(tokens)
tagged[:10]
```

Out[22]:
```
[('This', 'DT'),
 ('oatmeal', 'NN'),
 ('is', 'VBZ'),
 ('not', 'RB'),
 ('good', 'JJ'),
 ('.', '.'),
 ('Its', 'PRP$'),
 ('mushy', 'NN'),
 (',', ','),
 ('soft', 'JJ')]
```

In [23]:
```
nltk.download('maxent_ne_chunker')
```

```
[nltk_data] Downloading package maxent_ne_chunker to
[nltk_data]     C:\Users\sofia\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping chunkers\maxent_ne_chunker.zip.
```

Out[23]:
```
True
```

In [24]:
```
nltk.download('words')
```

```
[nltk_data] Downloading package words to
[nltk_data]     C:\Users\sofia\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\words.zip.
```

Out[24]:
```
True
```

In [25]:
```
entities=nltk.chunk.ne_chunk(tagged)
entities.pprint()
```

```
(S
  This/DT
  oatmeal/NN
  is/VBZ
  not/RB
  good/JJ
  ./.
  Its/PRP$
  mushy/NN
  ,/,
  soft/JJ
  ,/,
  I/PRP
  do/VBP
  n't/RB
  like/VB
  it/PRP
  ./.
  (ORGANIZATION Quaker/NNP Oats/NNPS)
  is/VBZ
  the/DT
  way/NN
  to/TO
  go/VB
  ./.)
```

VADER Sentiment Scoring

In [26]:
```
from nltk.sentiment import SentimentIntensityAnalyzer
from tqdm.notebook import tqdm
```

In [27]:
```
nltk.download('vader_lexicon')
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\sofia\AppData\Roaming\nltk_data...
```

Out[27]:
```
True
```

In [28]:
```
sia=SentimentIntensityAnalyzer()
```

In [29]:
```
sia.polarity_scores(example)
```

Out[29]:
```
{'neg': 0.22, 'neu': 0.78, 'pos': 0.0, 'compound': -0.5448}
```

In [30]:
```
sia.polarity_scores('I am so happy.')
```

Out[30]:
```
{'neg': 0.0, 'neu': 0.334, 'pos': 0.666, 'compound': 0.6115}
```

In [31]:
```
# Run the polarity score on the entire dataset
res={}
for i, row in tqdm(df.iterrows(),total=len(df)):
    text=row['Text']
    myid=row['Id']
    res[myid]=sia.polarity_scores(text)
```

```
100%                                                 500/500 [00:00<00:00, 1274.81it/s]
```

In [32]:
```
# first 10 values of result
n = list(res.items())[:10]

for key, value in n:
    print(f"{key}: {value}")
```

```
1: {'neg': 0.0, 'neu': 0.695, 'pos': 0.305, 'compound': 0.9441}
2: {'neg': 0.138, 'neu': 0.862, 'pos': 0.0, 'compound': -0.5664}
3: {'neg': 0.091, 'neu': 0.754, 'pos': 0.155, 'compound': 0.8265}
4: {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
5: {'neg': 0.0, 'neu': 0.552, 'pos': 0.448, 'compound': 0.9468}
6: {'neg': 0.029, 'neu': 0.809, 'pos': 0.163, 'compound': 0.883}
7: {'neg': 0.034, 'neu': 0.693, 'pos': 0.273, 'compound': 0.9346}
8: {'neg': 0.0, 'neu': 0.52, 'pos': 0.48, 'compound': 0.9487}
9: {'neg': 0.0, 'neu': 0.851, 'pos': 0.149, 'compound': 0.6369}
10: {'neg': 0.0, 'neu': 0.705, 'pos': 0.295, 'compound': 0.8313}
```

In [33]:
```
# converting results into dataframe
res_df=pd.DataFrame(res)
res_df
```

Out[33]:

	1	2	3	4	5	6	7	8	9	10	...	
neg	0.0000	0.1380	0.0910	0.0	0.0000	0.029	0.0340	0.0000	0.0000	0.0000	...	0.(
neu	0.6950	0.8620	0.7540	1.0	0.5520	0.809	0.6930	0.5200	0.8510	0.7050	...	0.1
pos	0.3050	0.0000	0.1550	0.0	0.4480	0.163	0.2730	0.4800	0.1490	0.2950	...	0.2
compound	0.9441	-0.5664	0.8265	0.0	0.9468	0.883	0.9346	0.9487	0.6369	0.8313	...	0.9

4 rows × 500 columns

In [34]:
```
res_df.T
```

Out[34]:

	neg	neu	pos	compound
1	0.000	0.695	0.305	0.9441
2	0.138	0.862	0.000	-0.5664
3	0.091	0.754	0.155	0.8265
4	0.000	1.000	0.000	0.0000
5	0.000	0.552	0.448	0.9468
...	...	...	...	...
496	0.000	0.554	0.446	0.9725
497	0.059	0.799	0.142	0.7833
498	0.025	0.762	0.212	0.9848
499	0.041	0.904	0.055	0.1280
500	0.000	0.678	0.322	0.9811

500 rows × 4 columns

In [36]:
```
vaders=res_df.T
vaders=vaders.reset_index().rename(columns={'index':'Id'})
vaders=vaders.merge(df,how='left')
```

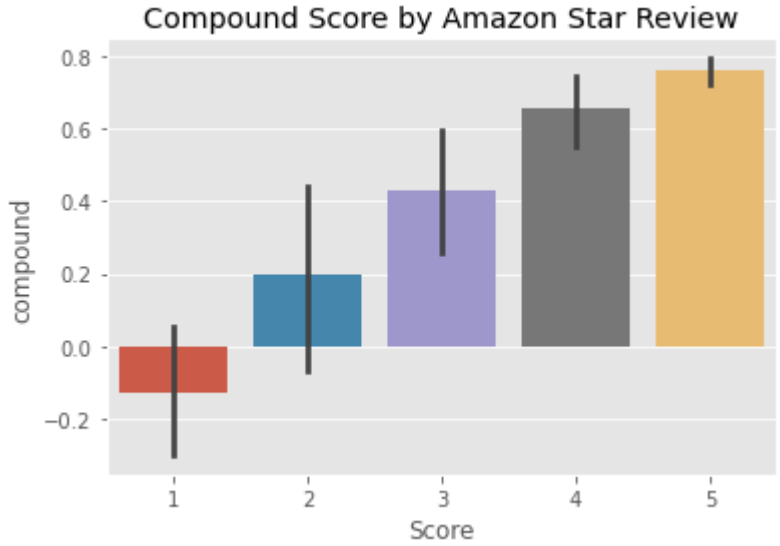
In [37]:
```
# Now we have sentiment score and metadata
vaders.head()
```

Out[37]:

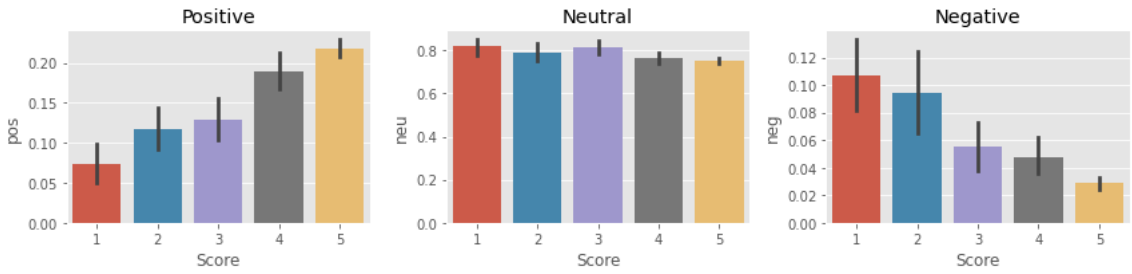
	Id	neg	neu	pos	compound	ProductId	UserId	ProfileName	Helpf
0	1	0.000	0.695	0.305	0.9441	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	
1	2	0.138	0.862	0.000	-0.5664	B00813GRG4	A1D87F6ZCVE5NK	dll pa	
2	3	0.091	0.754	0.155	0.8265	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	
3	4	0.000	1.000	0.000	0.0000	B000UA0QIQ	A395BORC6FGVXV	Karl	
4	5	0.000	0.552	0.448	0.9468	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	

Plot VADER Results

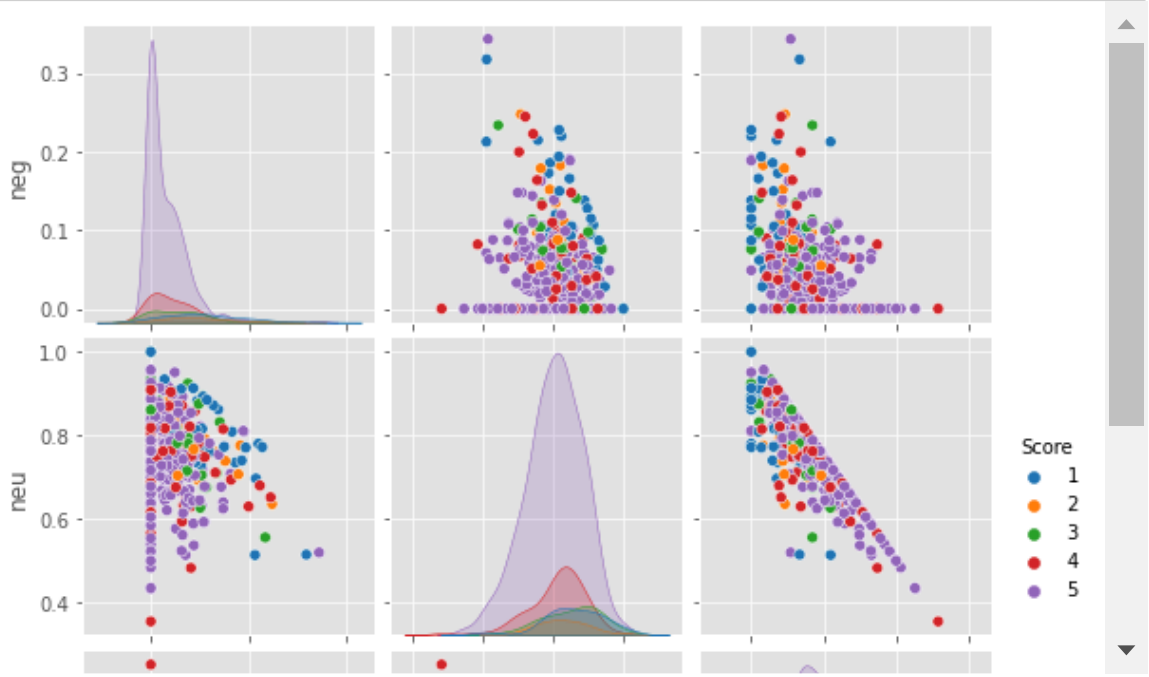
In [38]:
```
ax=sns.barplot(data=vaders,x='Score',y='compound')
ax.set_title('Compound Score by Amazon Star Review')
plt.show()
```



In [39]:
```
fig, axs = plt.subplots(1, 3, figsize=(12, 3))
sns.barplot(data=vaders, x='Score', y='pos', ax=axs[0])
sns.barplot(data=vaders, x='Score', y='neu', ax=axs[1])
sns.barplot(data=vaders, x='Score', y='neg', ax=axs[2])
axs[0].set_title('Positive')
axs[1].set_title('Neutral')
axs[2].set_title('Negative')
plt.tight_layout()
plt.show()
```



In [40]:
```
sns.pairplot(data=vaders,
             vars=['neg', 'neu', 'pos',],
            hue='Score',
            palette='tab10')
plt.show()
```



In [ ]: