In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
import sqlite3
```

In [3]:
```
con=sqlite3.connect(r'C:\Users\naila.iram\Downloads\Amazon Food Review Analysis\dat
```

In [4]:
```
type(con)
```

Out[4]:
```
sqlite3.Connection
```

In [5]:
```
data=pd.read_sql_query("SELECT * FROM REVIEWS",con)
```

In [6]:
```
data.head(2)
```

Out[6]:

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

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

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

In [8]:
```
data.dtypes
```

Out[8]:
```
Id                       int64
ProductId               object
UserId                  object
ProfileName             object
HelpfulnessNumerator     int64
HelpfulnessDenominator   int64
Score                    int64
Time                     int64
Summary                 object
Text                    object
dtype: object
```

In [9]:
```
data.columns
```

Out[9]:
```
Index(['Id', 'ProductId', 'UserId', 'ProfileName', 'HelpfulnessNumerator',
       'HelpfulnessDenominator', 'Score', 'Time', 'Summary', 'Text'],
      dtype='object')
```

In [10]:
```
data_valid = data[data['HelpfulnessNumerator'] <= data['HelpfulnessDenominator']]
```

In [11]:
```
data_valid.shape
```

Out[11]:
```
(568452, 10)
```

In [12]:
```
data_valid[data_valid.duplicated(['UserId', 'ProfileName','Time','Text'])]
```

Out[12]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	H
29	30	B0001PB9FY	A3HDKO7OW0QNK4	Canadian Fan	1	
574	575	B000G6RYNE	A3PJZ8TU8FDQ1K	Jared Castle	2	
1973	1974	B0017165OG	A2EPNS38TTLZYN	tedebear	0	
2309	2310	B0001VWE0M	AQM74O8Z4FMS0	Sunshine	0	
2323	2324	B0001VWE0C	AQM74O8Z4FMS0	Sunshine	0	
...	...	...	...	...	...	
568409	568410	B0018CLWM4	A2PE0AGWV6OPL7	Dark Water Mermaid	3	
568410	568411	B0018CLWM4	A88HLWDCU57WG	R28	2	

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	H
568411	568412	B0018CLWM4	AUX1HSY8FX55S	DAW	1	
568412	568413	B0018CLWM4	AVZ2OZ479Q9E8	Ai Ling Chow	0	
568413	568414	B0018CLWM4	AI3Y26HLPYW4L	kimosabe	1	

174521 rows × 10 columns

In [13]:
```
new_data=data_valid.drop_duplicates(subset=['UserId', 'ProfileName','Time','Text'])
```

In [14]:
```
new_data.shape
```

Out[14]:
```
(393931, 10)
```

In [15]:
```
import warnings
from warnings import filterwarnings
filterwarnings('ignore')
```

In [16]:
```
new_data['Time']=pd.to_datetime(new_data['Time'],unit='s')
```

In [17]:
```
### How Amazon Recommend Products (to what user amazon can recommend more products?
```

In [18]:
```
new_data['UserId'].nunique()
```

Out[18]:
```
256059
```

In [19]:
```
new_data.columns
```

Out[19]:
```
Index(['Id', 'ProductId', 'UserId', 'ProfileName', 'HelpfulnessNumerator',
       'HelpfulnessDenominator', 'Score', 'Time', 'Summary', 'Text'],
      dtype='object')
```

In [20]:
```
recommend_data=new_data.groupby(['UserId']).agg({'Summary':'count','Text':'count','
```

In [21]:
```
recommend_data.columns=['Number_of_summaries','num_text','avg_score','No_of_prods_p
```

In [22]:
```
recommend_data
```

Out[22]:

UserId	Number_of_summaries	num_text	avg_score	No_of_prods_purchased
AY12DBB0U420B	329	329	4.659574	329
A3OXHLG6DIBRW8	278	278	4.546763	278
A281NPSIMI1C2R	259	259	4.787645	259
A1YUL9PCJR3JTY	214	214	4.621495	214
A1Z54EM24Y40LL	211	211	4.383886	211
...	...	...	...	...
A2E80MDB9TCNGW	1	1	3.000000	1
A2E80RT3HOR35T	1	1	5.000000	1
A2E816C5N51F6X	1	1	5.000000	1
A2E81TVIUZI1IC	1	1	5.000000	1
AZZZOVIBXHGDR	1	1	2.000000	1

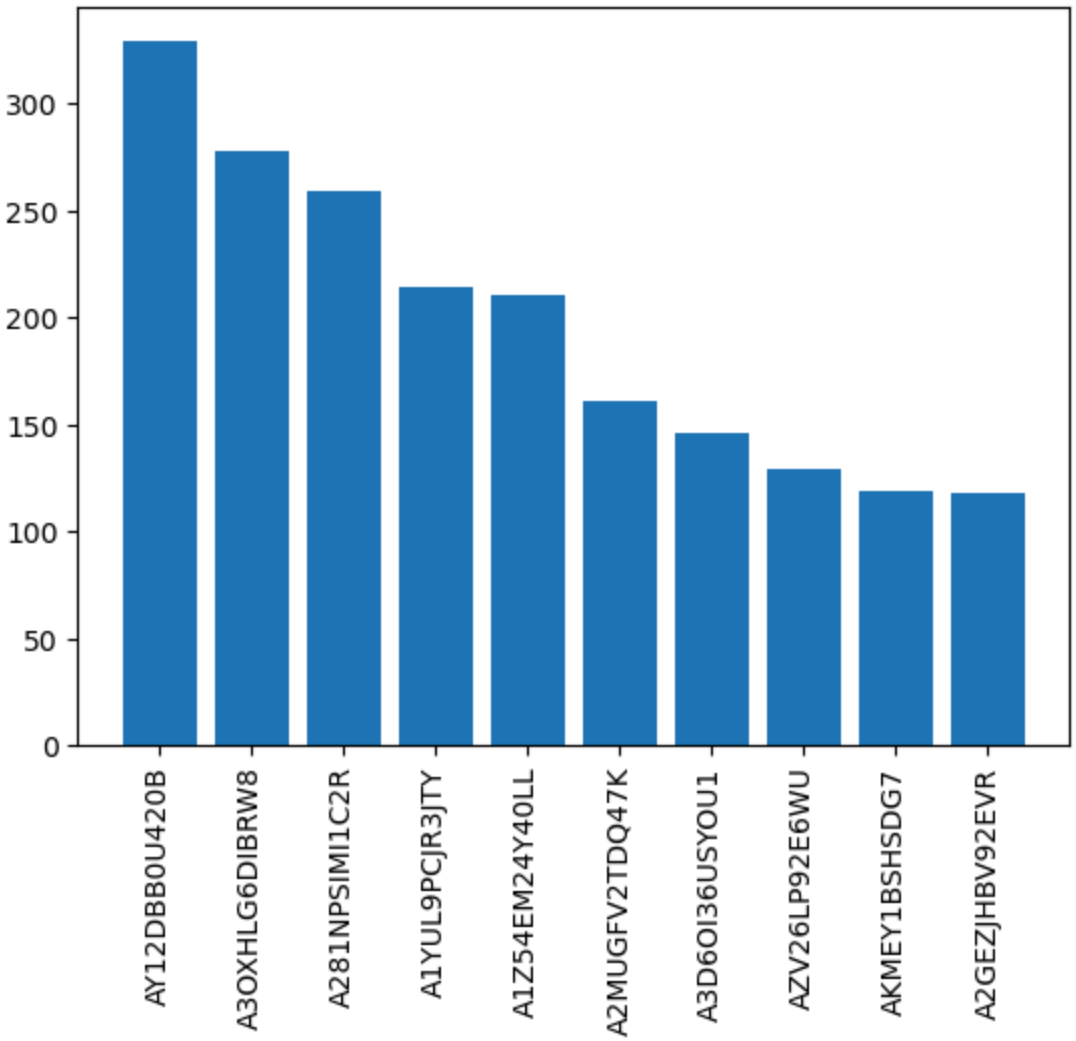
256059 rows × 4 columns

In [23]:
```
recommend_data['No_of_prods_purchased'][0:10].values
```

Out[23]:
```
array([329, 278, 259, 214, 211, 161, 146, 129, 119, 118], dtype=int64)
```

In [24]:
```
plt.bar(recommend_data.index[0:10],recommend_data['No_of_prods_purchased'][0:10].va
plt.xticks(rotation='vertical')
```

Out[24]:
```
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9],
 [Text(0, 0, 'AY12DBB0U420B'),
  Text(1, 0, 'A3OXHLG6DIBRW8'),
  Text(2, 0, 'A281NPSIMI1C2R'),
  Text(3, 0, 'A1YUL9PCJR3JTY'),
  Text(4, 0, 'A1Z54EM24Y40LL'),
  Text(5, 0, 'A2MUGFV2TDQ47K'),
  Text(6, 0, 'A3D6OI36USYOU1'),
  Text(7, 0, 'AZV26LP92E6WU'),
  Text(8, 0, 'AKMEY1BSHSDG7'),
  Text(9, 0, 'A2GEZJHBV92EVR')])
```



In [25]:
```
new_data.columns
```

Out[25]:
```
Index(['Id', 'ProductId', 'UserId', 'ProfileName', 'HelpfulnessNumerator',
       'HelpfulnessDenominator', 'Score', 'Time', 'Summary', 'Text'],
      dtype='object')
```

In [26]:
```
len(new_data['ProductId'].unique())
```

Out[26]:
```
67624
```

In [27]:
```
### which Product has good number of Reviews
prod_count=new_data['ProductId'].value_counts().to_frame()
```

In [28]:
```
prod_count
```

Out[28]:

ProductId	count
B007JFMH8M	912
B002QWP89S	630
B003B3OOPA	622
B001EO5Q64	566
B0013NUGDE	558
...	...
B002DNX4GO	1
B000FM2YU2	1
B001M1VA32	1
B009858H6M	1
B001LR2CU2	1

67624 rows × 1 columns

In [29]:
```
prod_count.columns
```

Out[29]:
```
Index(['count'], dtype='object')
```

In [30]:
```
freq_product_ids=prod_count[prod_count['count'] > 500].index
```

In [31]:
```
freq_product_ids
```

Out[31]:
```
Index(['B007JFMH8M', 'B002QWP89S', 'B003B3OOPA', 'B001EO5Q64', 'B0013NUGDE',
       'B000KV61FC', 'B000UBD88A', 'B000NMJWZO', 'B005K4Q37A', 'B0090X8IPM',
       'B005ZBZLT4'],
      dtype='object', name='ProductId')
```

In [32]:
```
new_data['ProductId'].isin(freq_product_ids)
```

Out[32]:
```
0         False
1         False
2         False
3         False
4         False
          ...
568449    False
568450    False
568451    False
568452    False
568453    False
Name: ProductId, Length: 393931, dtype: bool
```

In [33]:
```
freq_prod=new_data[new_data['ProductId'].isin(freq_product_ids)]
```

In [34]:
```
freq_prod
```

Out[34]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	He
20982	20983	B002QWP89S	A21U4DR8M6I9QN	K. M Merrill "justine"	1	
20983	20984	B002QWP89S	A17TDUBB4Z1PEC	jaded_green	1	
20984	20985	B002QWP89S	ABQH3WAWMSMBH	tenisbrat87	1	
20985	20986	B002QWP89S	AVTY5M74VA1BJ	tarotqueen	1	
20986	20987	B002QWP89S	A13TNN54ZEAUB1	dcz2221	1	
...	...	...	...	...	...	
563878	563879	B007JFMH8M	A366PSH7KFLRPB	TheRosySnail	0	
563879	563880	B007JFMH8M	A2KV6EYQPKJRR5	Kelley	0	
563880	563881	B007JFMH8M	A3O7REI0OSV89M	Esme	0	
563881	563882	B007JFMH8M	A9JS5GQQ6GIQT	Syne	0	

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	He
563882	563883	B007JFMH8M	AMAVEZAGCH52H	Tangela	0	

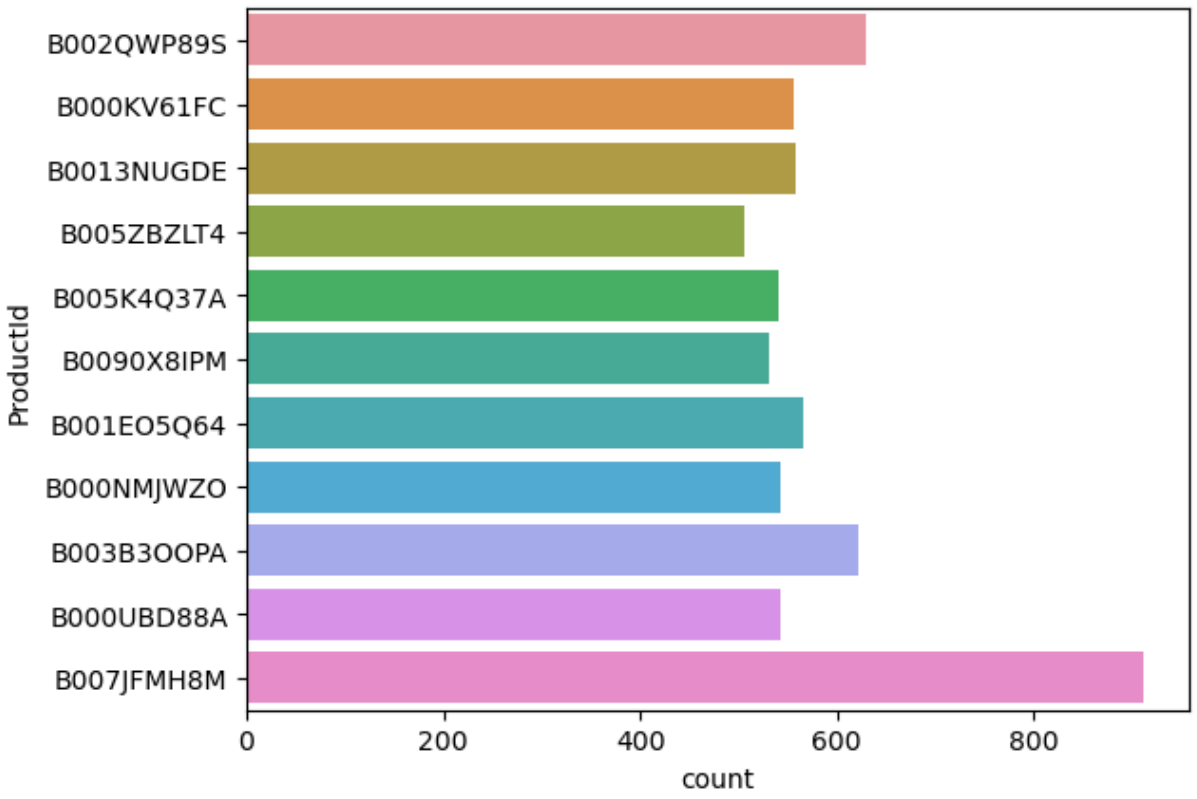
6504 rows × 10 columns

In [35]:
```
freq_prod.columns
```

Out[35]:
```
Index(['Id', 'ProductId', 'UserId', 'ProfileName', 'HelpfulnessNumerator',
       'HelpfulnessDenominator', 'Score', 'Time', 'Summary', 'Text'],
      dtype='object')
```

In [46]:
```
sns.countplot(y='ProductId' , data=freq_prod)
```

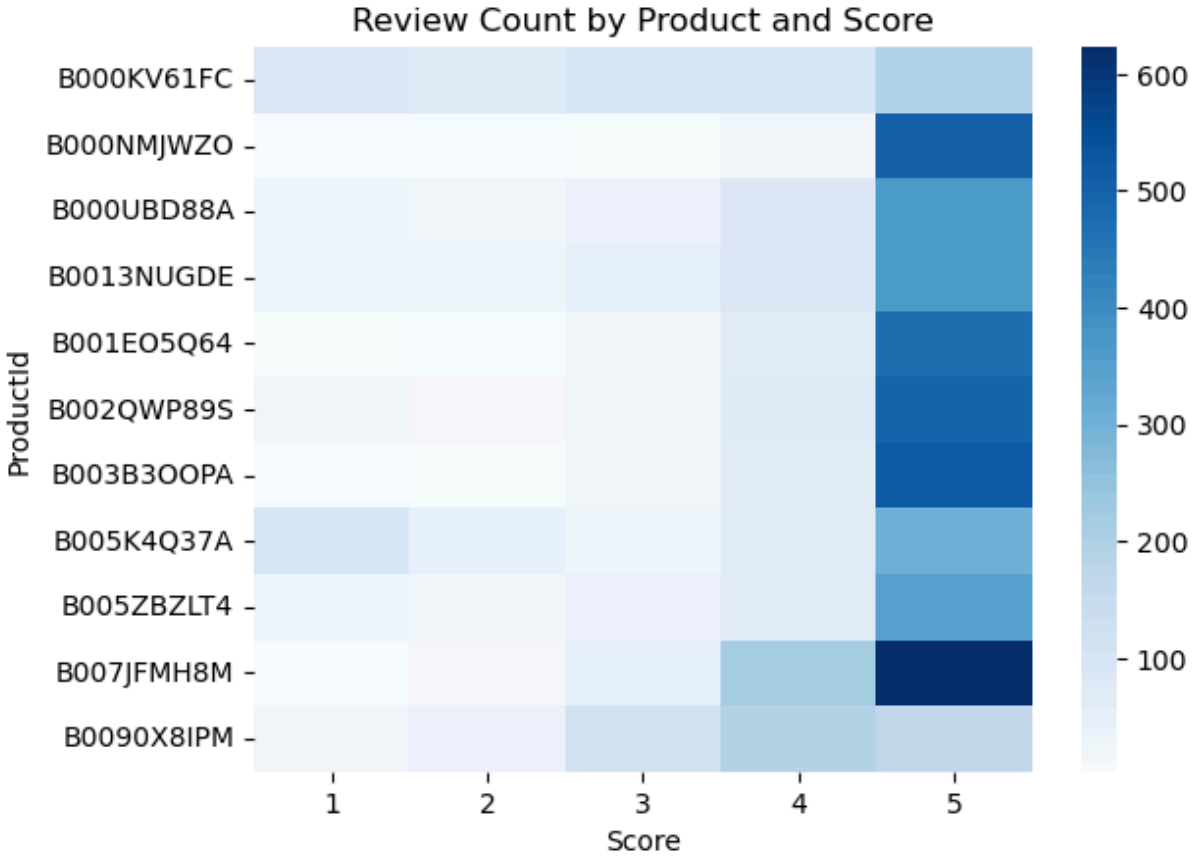
Out[46]:
```
<Axes: xlabel='count', ylabel='ProductId'>
```



In [47]:
```
pivot = (
    freq_prod
    .groupby(['ProductId', 'Score'])
    .size()
    .unstack(fill_value=0)
)

sns.heatmap(pivot, cmap='Blues')
plt.title('Review Count by Product and Score')
```

```
plt.xlabel('Score')
plt.ylabel('ProductId')
plt.show()
```



In [52]:
```
### Is there any difference between behaviour of frequent viewers & not frequent vi
x=new_data['UserId'].value_counts()
```

In [55]:
```
new_data['viewer_type']= new_data['UserId'].apply(lambda user : "Frequent" if x[use
```

In [56]:
```
new_data.head(2)
```

Out[56]:

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

In [58]:
```
new_data['viewer_type'].unique()
```

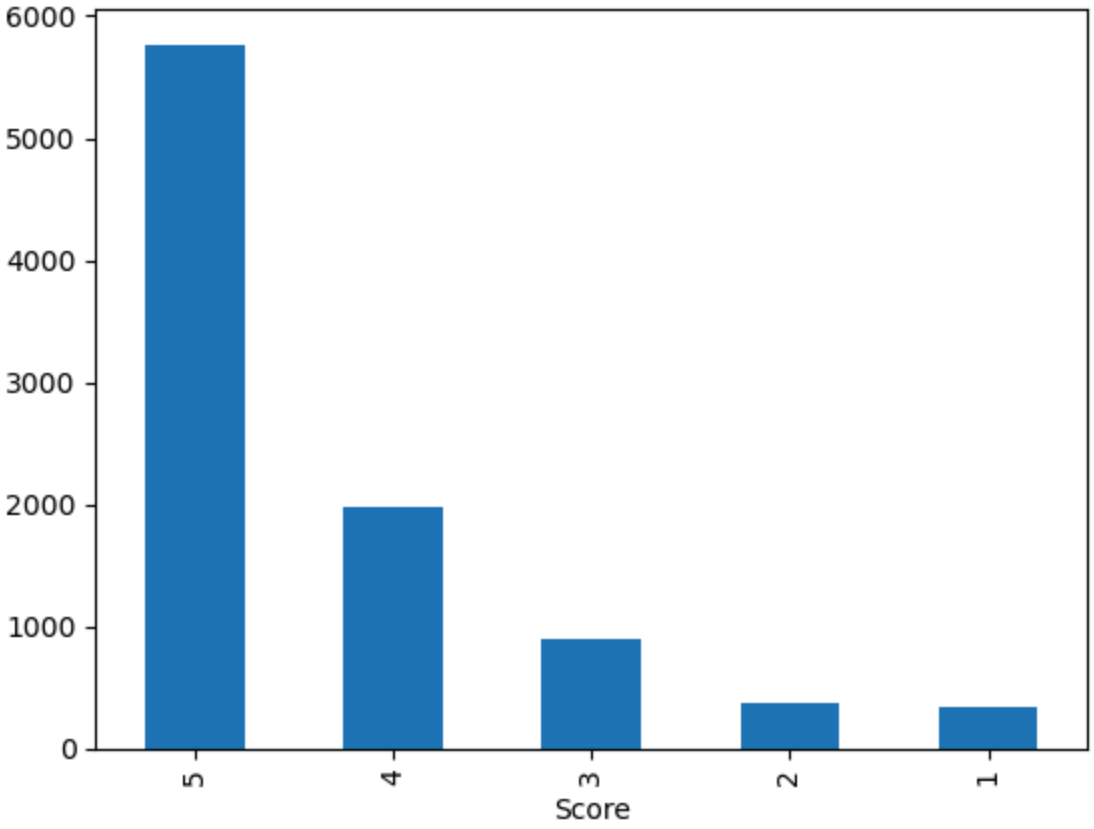
Out[58]:
```
array(['Not Frequent', 'Frequent'], dtype=object)
```

In [59]:
```
not_freq_viewers=new_data[new_data['viewer_type']=='Not Frequent']
```

In [60]:
```
freq_viewers=new_data[new_data['viewer_type']=='Frequent']
```

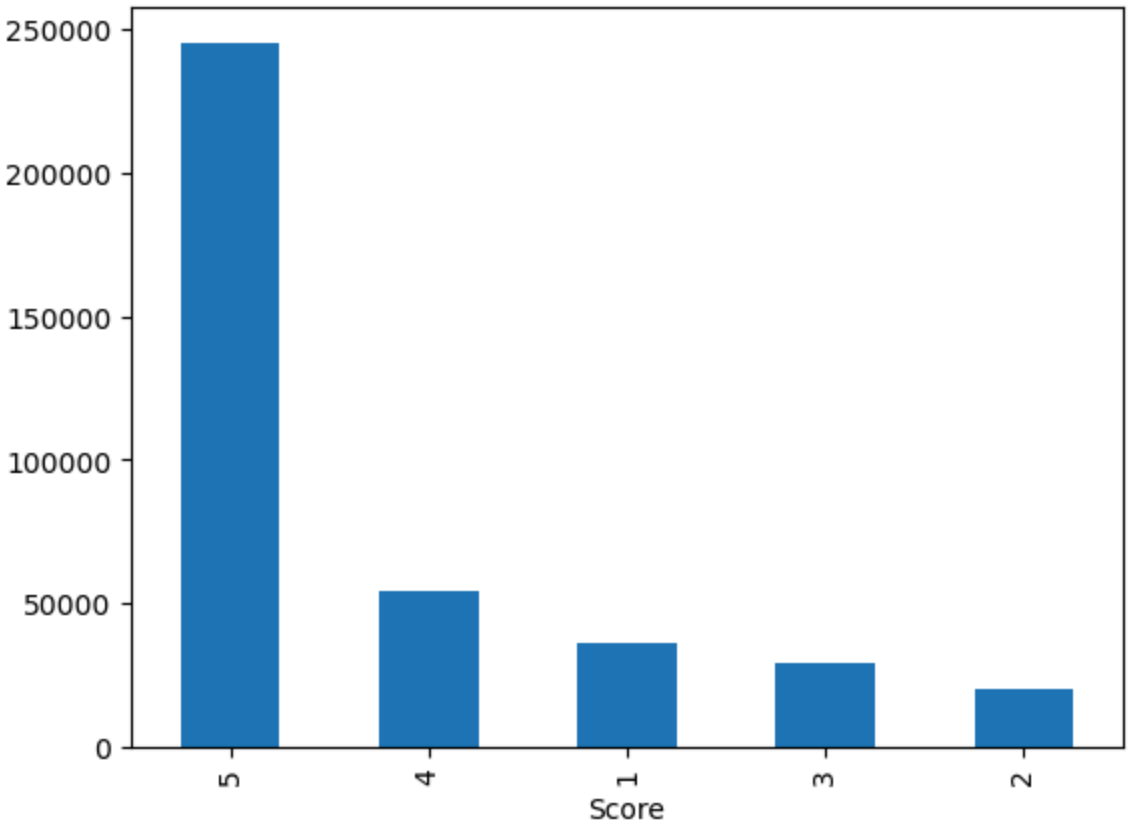
In [63]:
```
freq_viewers['Score'].value_counts().plot(kind='bar')
```

Out[63]:
```
<Axes: xlabel='Score'>
```



In [65]:
```
not_freq_viewers['Score'].value_counts().plot(kind='bar')
```

Out[65]: <Axes: xlabel='Score'>



In [66]:
```
new_data[['UserId','ProductId','Text']]
```

Out[66]:

	UserId	ProductId	Text
0	A3SGXH7AUHU8GW	B001E4KFG0	I have bought several of the Vitality canned d...
1	A1D87F6ZCVE5NK	B00813GRG4	Product arrived labeled as Jumbo Salted Peanut...
2	ABXLMWJIXXAIN	B000LQOCH0	This is a confection that has been around a fe...
3	A395BORC6FGVXV	B000UA0QIQ	If you are looking for the secret ingredient i...
4	A1UQRSCLF8GW1T	B006K2ZZ7K	Great taffy at a great price. There was a wid...
...	...	...	...
568449	A28KG5XORO54AY	B001EO7N10	Great for sesame chicken..this is a good if no...
568450	A3I8AFVPEE8KI5	B003S1WTCU	I'm disappointed with the flavor. The chocolat...
568451	A121AA1GQV751Z	B004I613EE	These stars are small, so you can give 10-15 o...
568452	A3IBEVCTXKNOH	B004I613EE	These are the BEST treats for training and rew...
568453	A3LGQPJCZVL9UC	B001LR2CU2	I am very satisfied ,product is as advertised,...

393931 rows × 3 columns

In [68]:
```
def calculate_length(text):
    return len(text.split(' '))
```

In [69]:
```
new_data['Text_length']= new_data['Text'].apply(calculate_length)
```

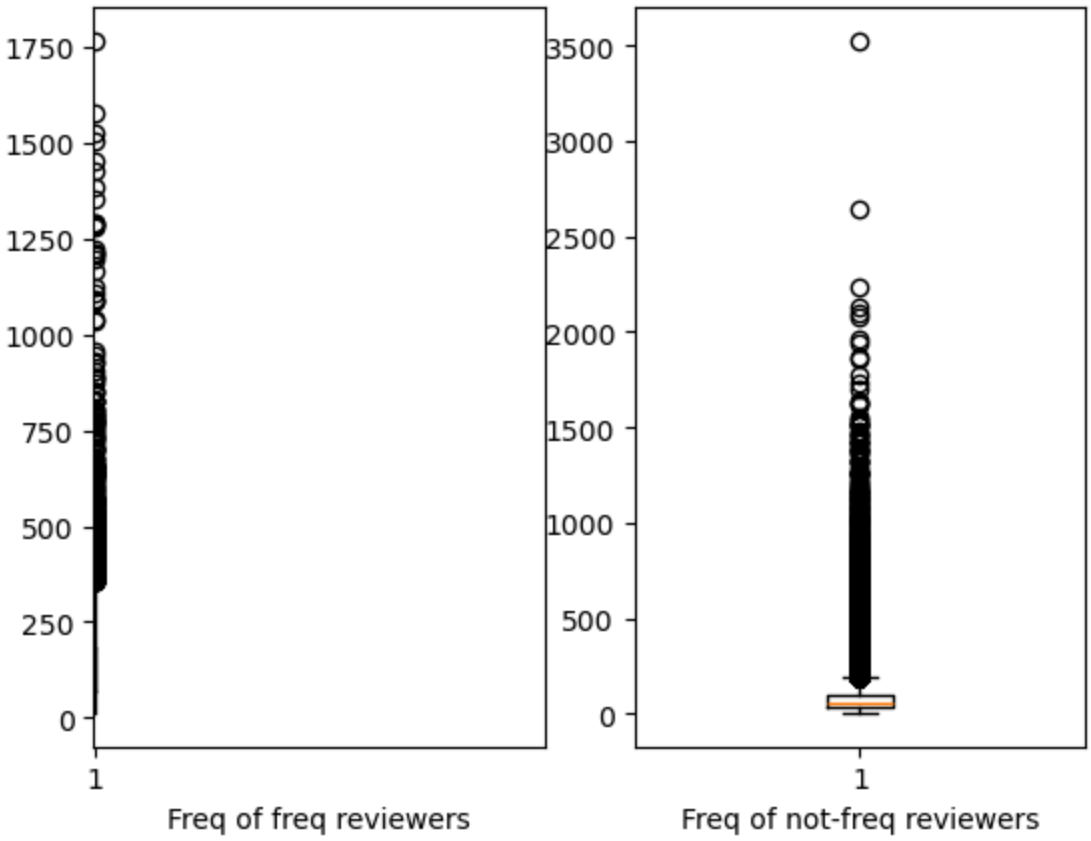
In [71]:
```
not_freq_data=new_data[new_data['viewer_type']=='Not Frequent']
freq_data=new_data[new_data['viewer_type']=='Frequent']
```

In [83]:
```
fig=plt.figure()

ax1=fig.add_subplot(121)
ax1.boxplot(freq_data['Text_length'])
ax1.set_xlabel('Freq of freq reviewers')
ax1.set_xlim(0,600)

ax2=fig.add_subplot(122)
ax2.boxplot(not_freq_data['Text_length'])
ax2.set_xlabel('Freq of not-freq reviewers')
```

Out[83]:
```
Text(0.5, 0, 'Freq of not-freq reviewers')
```

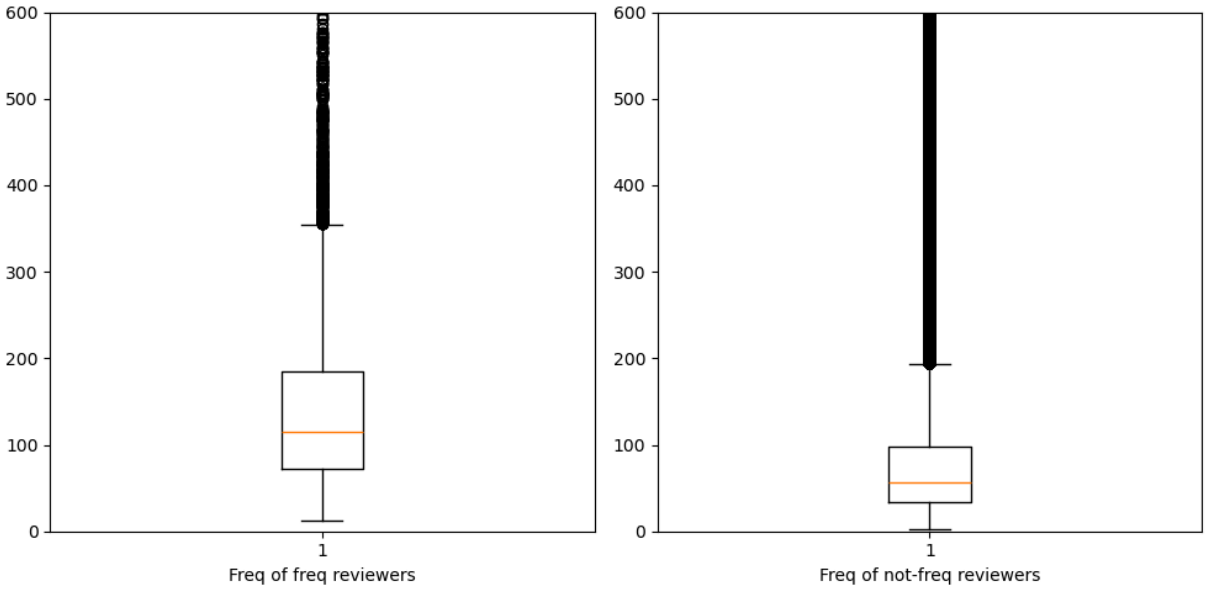


In [85]:
```
fig = plt.figure(figsize=(10, 5))

ax1 = fig.add_subplot(121)
ax1.boxplot(freq_data['Text_length'])
ax1.set_xlabel('Freq of freq reviewers')
ax1.set_ylim(0, 600)

ax2 = fig.add_subplot(122)
ax2.boxplot(not_freq_data['Text_length'])
ax2.set_xlabel('Freq of not-freq reviewers')
ax2.set_ylim(0, 600)

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



In [ ]:
```
### Sentiment Amazon_food_review_analysis.ipynb
```

In [86]:
```
!pip install textblob
```

```
Collecting textblob
  Downloading textblob-0.19.0-py3-none-any.whl.metadata (4.4 kB)
Collecting nltk>=3.9 (from textblob)
  Downloading nltk-3.9.2-py3-none-any.whl.metadata (3.2 kB)
Requirement already satisfied: click in c:\users\naila.iram\appdata\local\anaconda3
\lib\site-packages (from nltk>=3.9->textblob) (8.2.1)
Requirement already satisfied: joblib in c:\users\naila.iram\appdata\local\anaconda3
\lib\site-packages (from nltk>=3.9->textblob) (1.2.0)
Requirement already satisfied: regex>=2021.8.3 in c:\users\naila.iram\appdata\local
\anaconda3\lib\site-packages (from nltk>=3.9->textblob) (2023.10.3)
Requirement already satisfied: tqdm in c:\users\naila.iram\appdata\local\anaconda3\l
ib\site-packages (from nltk>=3.9->textblob) (4.65.0)
Requirement already satisfied: colorama in c:\users\naila.iram\appdata\local\anacond
a3\lib\site-packages (from click->nltk>=3.9->textblob) (0.4.6)
Downloading textblob-0.19.0-py3-none-any.whl (624 kB)
   ---------------------------------------- 0.0/624.3 kB ? eta -:--:--
   - -------------------------------------- 30.7/624.3 kB 640.0 kB/s eta 0:00:01
   --------------- ----------------------- 256.0/624.3 kB 3.2 MB/s eta 0:00:01
   ---------------------------------------- 624.3/624.3 kB 4.9 MB/s eta 0:00:00
Downloading nltk-3.9.2-py3-none-any.whl (1.5 MB)
   ---------------------------------------- 0.0/1.5 MB ? eta -:--:--
   ------------- ------------------------- 0.6/1.5 MB 17.8 MB/s eta 0:00:01
   ---------------------------------- --- 1.4/1.5 MB 17.5 MB/s eta 0:00:01
   ---------------------------------------- 1.5/1.5 MB 13.8 MB/s eta 0:00:00
Installing collected packages: nltk, textblob
  Attempting uninstall: nltk
    Found existing installation: nltk 3.8.1
    Uninstalling nltk-3.8.1:
      Successfully uninstalled nltk-3.8.1
Successfully installed nltk-3.9.2 textblob-0.19.0
```

In [87]:
```
from textblob import TextBlob
```

In [88]:
```
TextBlob(new_data['Summary'][0]).sentiment.polarity
```

Out[88]: 0.7

In [89]:
```
sample= new_data[0:50000]
```

In [90]:
```
polarity=[]
for text in sample['Summary']:
    try:
        polarity.append(TextBlob(text).sentiment.polarity)
    except:
        polarity.append(0)
```

In [91]:
```
len(polarity)
```

Out[91]: 50000

In [92]:
```
sample['polarity']=polarity
```

In [93]:
```
sample.head(3)
```

Out[93]:

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

In [94]:
```
negetive_polarity=sample[sample['polarity']<0]
positive_polarity=sample[sample['polarity']>0]
```

In [95]:
```
negetive_polarity['Summary']
```

Out[95]:
```
16                  poor taste
26              Nasty No flavor
57       How can you go wrong!
61             pretty expensive
62               stale product.
                  ...
54185                 Horrible
54186            Horrible idea
54191                    STALE
54212             Disappointed
54251      Just not very good!
Name: Summary, Length: 4659, dtype: object
```

In [96]:
```
from collections import Counter
```

In [97]:
```
Counter(negetive_polarity['Summary']).most_common(10)
```

Out[97]:
```
[('Disappointed', 44),
 ('Disappointing', 32),
 ('Bland', 18),
 ('Awful', 17),
 ('Not what I expected', 17),
 ('Terrible', 15),
 ('Horrible', 15),
 ('disappointed', 15),
 ('Disgusting', 12),
 ('not good', 11)]
```

In [98]:
```
Counter(positive_polarity['Summary']).most_common(10)
```

Out[98]:
```
[('Delicious!', 208),
 ('Delicious', 204),
 ('Great product', 100),
 ('Excellent', 85),
 ('Love it!', 81),
 ('Great', 81),
 ('Great Product', 77),
 ('Great!', 70),
 ('Good stuff', 51),
 ('Awesome', 50)]
```

In [ ]: