In [2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [3]:
```
import sqlite3
```

In [4]:
```
con = sqlite3.connect(r"C:\Users\usr\Downloads\amazon\database.sqlite")
```

In [5]:
```
type(con)
```

Out[5]:
```
sqlite3.Connection
```

In [9]:
```
df = pd.read_sql_query ("SELECT * FROM REVIEWS", con)
```

In [8]:
```
df.shape
```

Out[8]:
```
(568454, 10)
```

In [10]:
```
type(df)
```

Out[10]:
```
pandas.core.frame.DataFrame
```

In [11]:
```
df.head(4)
```

Out[11]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	130
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0	1	134
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1	4	121
3	4	B000UA0QIQ	A395BORC6FGVXV	Karl	3	3	2	130

In [15]:
```
df.columns
```

Out[15]:
```
Index(['Id', 'ProductId', 'UserId', 'ProfileName', 'HelpfulnessNumerator',
       'HelpfulnessDenominator', 'Score', 'Time', 'Summary', 'Text'],
```

```
      dtype='object')
```

In [16]:
```
df[df['HelpfulnessNumerator'] > df['HelpfulnessDenominator']]
```

Out[16]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Scor
44736	44737	B001EQ55RW	A2V0I904FH7ABY	Ram	3	2	
64421	64422	B000MIDROQ	A161DK06JJMCYF	J. E. Stephens "Jeanne"	3	1	

In [21]:
```
df_valid = df[df['HelpfulnessNumerator'] <= df['HelpfulnessDenominator']]
```

In [22]:
```
df_valid.shape
```

Out[22]:
```
(568452, 10)
```

In [23]:
```
df_valid.columns
```

Out[23]:
```
Index(['Id', 'ProductId', 'UserId', 'ProfileName', 'HelpfulnessNumerator',
       'HelpfulnessDenominator', 'Score', 'Time', 'Summary', 'Text'],
      dtype='object')
```

In [24]:
```
df_valid.duplicated(['UserId', 'ProfileName', 'Time', 'Text'])
```

Out[24]:
```
0         False
1         False
2         False
3         False
4         False
          ...
568449    False
568450    False
568451    False
568452    False
568453    False
Length: 568452, dtype: bool
```

In [27]:
```
df_valid[df_valid.duplicated(['UserId', 'ProfileName', 'Time', 'Text'])]
```

Out[27]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator
29	30	B0001PB9FY	A3HDKO7OW0QNK4	Canadian Fan	1	1
574	575	B000G6RYNE	A3PJZ8TU8FDQ1K	Jared Castle	2	2

1973	1974	B0017165OG	A2EPNS38TTLZYN	tedebear	0	0
2309	2310	B0001VWE0M	AQM74O8Z4FMS0	Sunshine	0	0
2323	2324	B0001VWE0C	AQM74O8Z4FMS0	Sunshine	0	0
...	...	...	...	...	...	...
568409	568410	B0018CLWM4	A2PE0AGWV6OPL7	Dark Water Mermaid	3	3
568410	568411	B0018CLWM4	A88HLWDCU57WG	R28	2	2
568411	568412	B0018CLWM4	AUX1HSY8FX55S	DAW	1	1
568412	568413	B0018CLWM4	AVZ2OZ479Q9E8	Ai Ling Chow	0	0
568413	568414	B0018CLWM4	AI3Y26HLPYW4L	kimosabe	1	2

174521 rows × 10 columns

In [28]:
```
data = df_valid.drop_duplicates(subset = ['UserId', 'ProfileName', 'Time', 'Text'])
```

In [29]:
```
data.shape
```

Out[29]: (393931, 10)

In [30]:
```
data.dtypes
```

Out[30]:
```
Id                      int64
ProductId              object
UserId                 object
ProfileName            object
HelpfulnessNumerator    int64
HelpfulnessDenominator  int64
Score                   int64
Time                    int64
Summary                object
Text                   object
dtype: object
```

In [32]:
```
pd.to_datetime(data['Time'])
```

Out[32]:
```
0        1970-01-01 00:00:01.303862400
1        1970-01-01 00:00:01.346976000
2        1970-01-01 00:00:01.219017600
3        1970-01-01 00:00:01.307923200
4        1970-01-01 00:00:01.350777600
                     ...
568449   1970-01-01 00:00:01.299628800
568450   1970-01-01 00:00:01.331251200
568451   1970-01-01 00:00:01.329782400
568452   1970-01-01 00:00:01.331596800
568453   1970-01-01 00:00:01.338422400
Name: Time, Length: 393931, dtype: datetime64[ns]
```

In [33]:
```
data.columns
```

Out[33]:
```
Index(['Id', 'ProductId', 'UserId', 'ProfileName', 'HelpfulnessNumerator',
       'HelpfulnessDenominator', 'Score', 'Time', 'Summary', 'Text'],
      dtype='object')
```

In [35]:
```
data['ProfileName'].unique()
```

Out[35]:
```
array(['delmartian', 'dll pa', 'Natalia Corres "Natalia Corres"', ...,
       'Lettie D. Carter', 'pksd "pk_007"', 'srfell17'], dtype=object)
```

In [36]:
```
data['ProfileName'].nunique()
```

Out[36]:
```
218418
```

In [37]:
```
data['UserId'].nunique()
```

Out[37]:
```
256059
```

In [40]:
```
recomend_df = data.groupby(['UserId']).agg({'Summary':'count', 'Text':'count','Score':'m
```

In [42]:
```
recomend_df.columns
```

Out[42]:
```
Index(['Summary', 'Text', 'Score', 'ProductId'], dtype='object')
```

In [46]:
```
recomend_df.columns = ['Number_of_summaries', 'num_text', 'avg_score', 'No_of_prods_purc
```

In [47]:
```
recomend_df
```

Out[47]:

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 [48]:
```
recomend_df.index[0:10]
```

Out[48]:
```
Index(['AY12DBB0U420B', 'A3OXHLG6DIBRW8', 'A281NPSIMI1C2R', 'A1YUL9PCJR3JTY',
       'A1Z54EM24Y40LL', 'A2MUGFV2TDQ47K', 'A3D6OI36USYOU1', 'AZV26LP92E6WU',
       'AKMEY1BSHSDG7', 'A2GEZJHBV92EVR'],
      dtype='object', name='UserId')
```

In [50]:
```
recomend_df['No_of_prods_purchased'][0:10].values
```

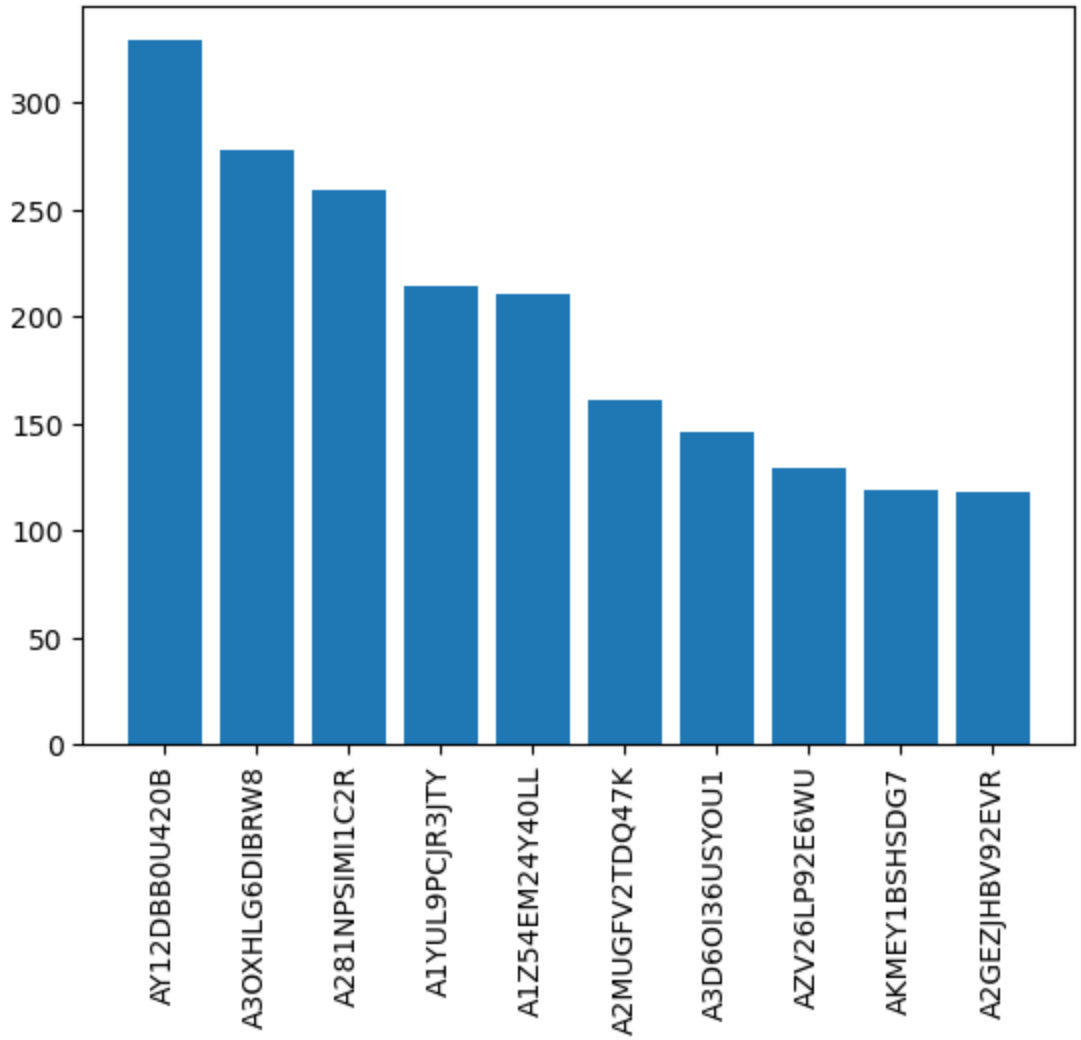
Out[50]:
```
array([329, 278, 259, 214, 211, 161, 146, 129, 119, 118], dtype=int64)
```

In [ ]:
```

```

In [53]:
```
plt.bar(recomend_df.index[0:10], recomend_df['No_of_prods_purchased'][0:10].values )
plt.xticks(rotation = 'vertical')
```

Out[53]:
```
([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 [ ]:
```
# Analysing which product has best number of Reviews ?
```

In [54]:
```
data.columns
```

Out[54]:
```
Index(['Id', 'ProductId', 'UserId', 'ProfileName', 'HelpfulnessNumerator',
       'HelpfulnessDenominator', 'Score', 'Time', 'Summary', 'Text'],
      dtype='object')
```

In [56]:
```
len(data['ProductId'].unique())
```

Out[56]:
```
67624
```

In [58]:
```
prod_count = data['ProductId'].value_counts().to_frame()
```

In [59]:
```
prod_count
```

Out[59]:

	ProductId
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 [63]:
```
prod_count['ProductId'] > 500
```

Out[63]:
```
B007JFMH8M     True
B002QWP89S     True
B003B3OOPA     True
B001EO5Q64     True
B0013NUGDE     True
               ...
B002DNX4GO    False
B000FM2YU2    False
B001M1VA32    False
B009858H6M    False
B001LR2CU2    False
Name: ProductId, Length: 67624, dtype: bool
```

In [66]:
```
freq_prod_ids = prod_count[prod_count['ProductId'] > 500].index
```

In [ ]:
```
freq_prod_ids
```

In [67]:
```
data['ProductId'].isin(freq_prod_ids)
```

Out[67]:
```
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 [69]:
```
fre_prod_df = data[data['ProductId'].isin(freq_prod_ids)]
```

In [70]:
```
fre_prod_df
```

Out[70]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator
20982	20983	B002QWP89S	A21U4DR8M6I9QN	K. M Merrill "justine"	1	1
20983	20984	B002QWP89S	A17TDUBB4Z1PEC	jaded_green	1	1
20984	20985	B002QWP89S	ABQH3WAWMSMBH	tenisbrat87	1	1

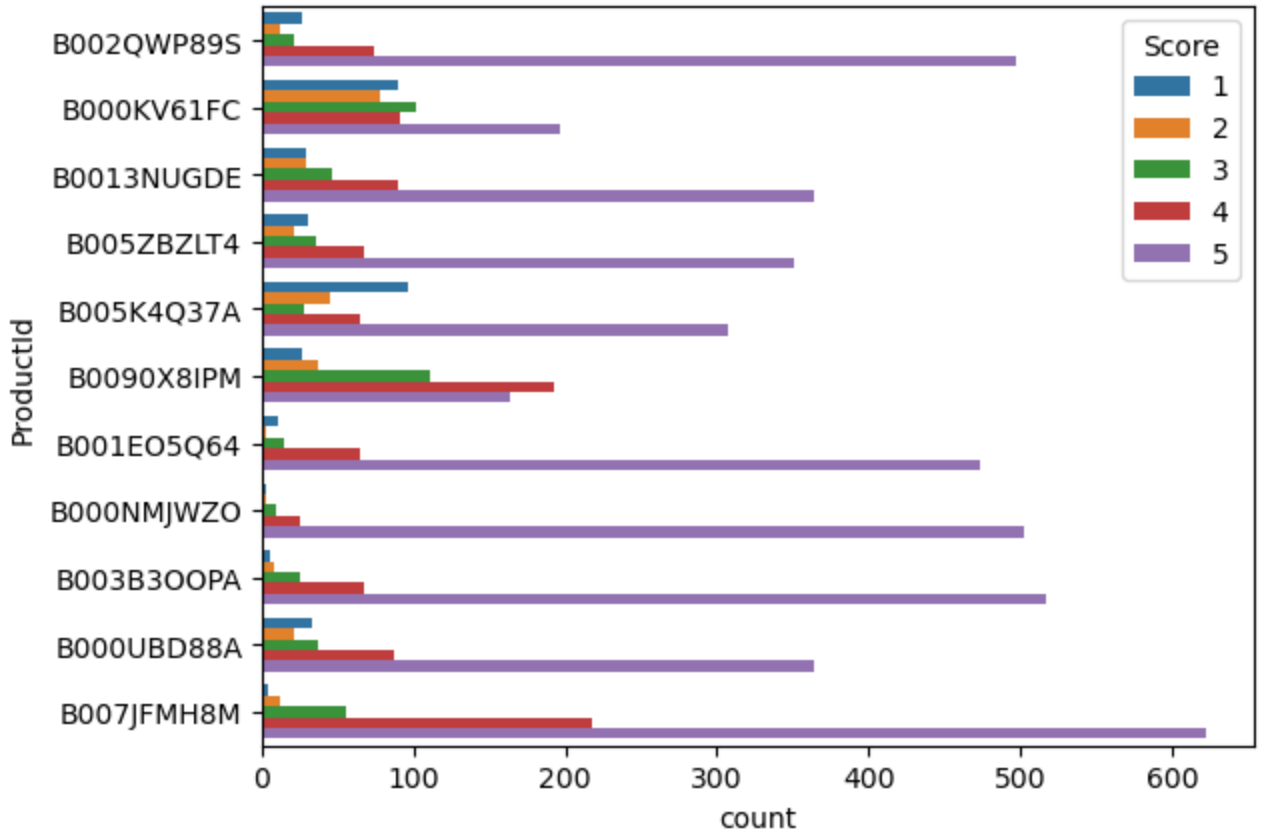
20985	20986	B002QWP89S	AVTY5M74VA1BJ	tarotqueen	1	1
20986	20987	B002QWP89S	A13TNN54ZEAUB1	dcz2221	1	1
...	...	...	...	...	...	...
563878	563879	B007JFMH8M	A366PSH7KFLRPB	TheRosySnail	0	0
563879	563880	B007JFMH8M	A2KV6EYQPKJRR5	Kelley	0	0
563880	563881	B007JFMH8M	A3O7REI0OSV89M	Esme	0	0
563881	563882	B007JFMH8M	A9JS5GQQ6GIQT	Syne	0	0
563882	563883	B007JFMH8M	AMAVEZAGCH52H	Tangela	0	0

6504 rows × 10 columns

In [ ]:
```
#visualize
```

In [73]:
```
sns.countplot(y = 'ProductId', data = fre_prod_df, hue = 'Score')
```

Out[73]:
```
<Axes: xlabel='count', ylabel='ProductId'>
```



In [ ]:

In [ ]:
```
#Understanding Behaviours of Amazon Users !
```

In [74]:
```
data.columns
```

Out[74]:
```
Index(['Id', 'ProductId', 'UserId', 'ProfileName', 'HelpfulnessNumerator',
       'HelpfulnessDenominator', 'Score', 'Time', 'Summary', 'Text'],
      dtype='object')
```

In [79]:
```
x = data['UserId'].value_counts()
```

In [80]:
```
x
```

Out[80]:
```
AY12DBB0U420B     329
A3OXHLG6DIBRW8    278
A281NPSIMI1C2R    259
A1YUL9PCJR3JTY    214
A1Z54EM24Y40LL    211
                 ...
AAQPR1MSRXKTU       1
AGO81Z6PZSF7P       1
ALA84XWMTQBFT       1
A1G9DK8EUR36JC      1
A3LGQPJCZVL9UC      1
Name: UserId, Length: 256059, dtype: int64
```

In [78]:
```
data.head()
```

Out[78]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	130

1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0		0	1	134
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1		1	4	121
3	4	B000UA0QIQ	A395BORC6FGVXV	Karl	3		3	2	130
4	5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	0		0	5	135

In [ ]:

In [82]:
```
data['UserId'].apply(lambda user: "Frequent" if x[user]>50 else "Not frequent" )
```

Out[82]:
```
0         Not frequent
1         Not frequent
2         Not frequent
3         Not frequent
4         Not frequent
             ...
568449    Not frequent
568450    Not frequent
568451    Not frequent
568452    Not frequent
568453    Not frequent
Name: UserId, Length: 393931, dtype: object
```

In [89]:
```
data['viewer_type']
```

Out[89]:
```
0         Not frequent
1         Not frequent
2         Not frequent
3         Not frequent
4         Not frequent
             ...
568449    Not frequent
568450    Not frequent
568451    Not frequent
568452    Not frequent
568453    Not frequent
Name: viewer_type, Length: 393931, dtype: object
```

In [90]:
```
data['viewer_type'] = data['UserId'].apply(lambda user: "Frequent" if x[user]>50 else "N
```

```
C:\Users\usr\AppData\Local\Temp\ipykernel_25328\2371619637.py:1: SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data['viewer_type'] = data['UserId'].apply(lambda user: "Frequent" if x[user]>50 else "Not frequent" )
```

In [91]:
```
data.columns
```

Out[91]:
```
Index(['Id', 'ProductId', 'UserId', 'ProfileName', 'HelpfulnessNumerator',
       'HelpfulnessDenominator', 'Score', 'Time', 'Summary', 'Text',
       'viewer_type'],
      dtype='object')
```

In [92]:
```
data.head(3)
```

Out[92]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	130
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0	1	134
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1	4	121

In [93]:
```
data['viewer_type'].unique()
```

Out[93]:
```
array(['Not frequent', 'Frequent'], dtype=object)
```

In [106…
```
not_freq_df = data[data['viewer_type']=='Not frequent']
freq_df = data[data['viewer_type']=='Frequent']
```

In [102…
```
freq_df['Score'].value_counts()/len(freq_df)*100
```

Out[102]:
```
5    61.605044
4    21.147681
3     9.585381
2     3.932464
1     3.729429
Name: Score, dtype: float64
```

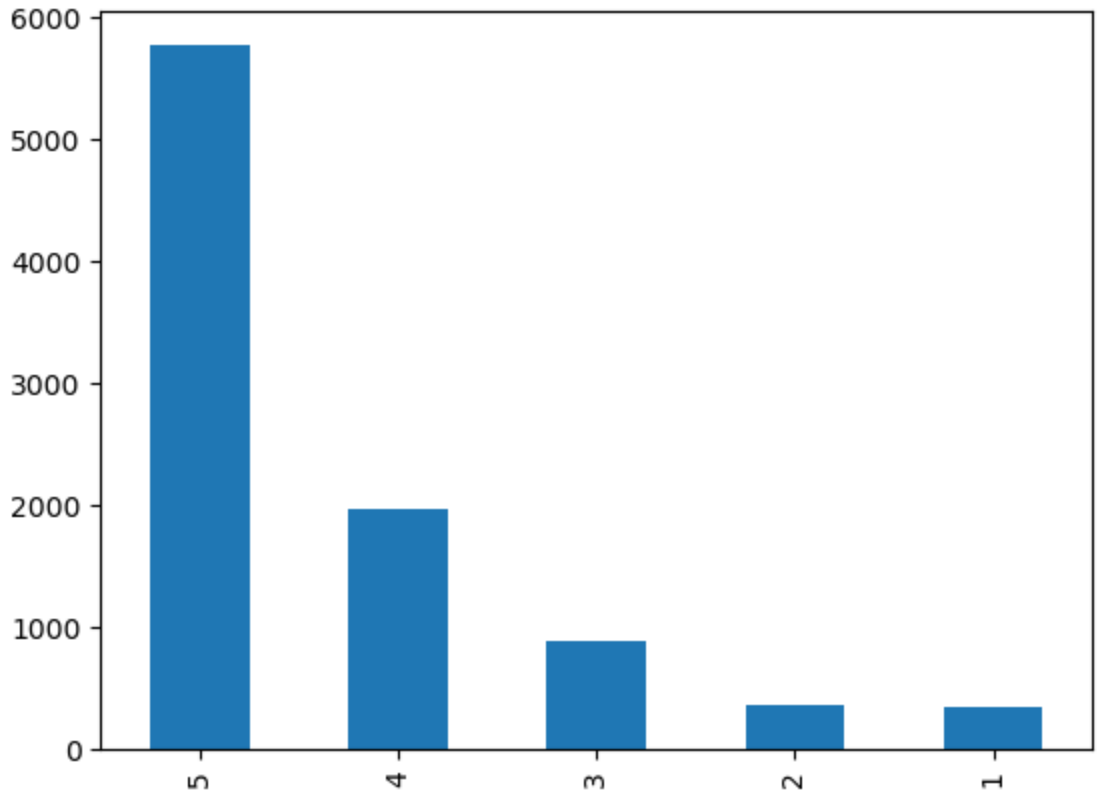
In [107…
```
not_freq_df['Score'].value_counts()/len(not_freq_df)*100
```

Out[107]:
```
5    63.757986
4    14.071191
1     9.349850
3     7.507547
```

```
2      5.313426
Name: Score, dtype: float64
```

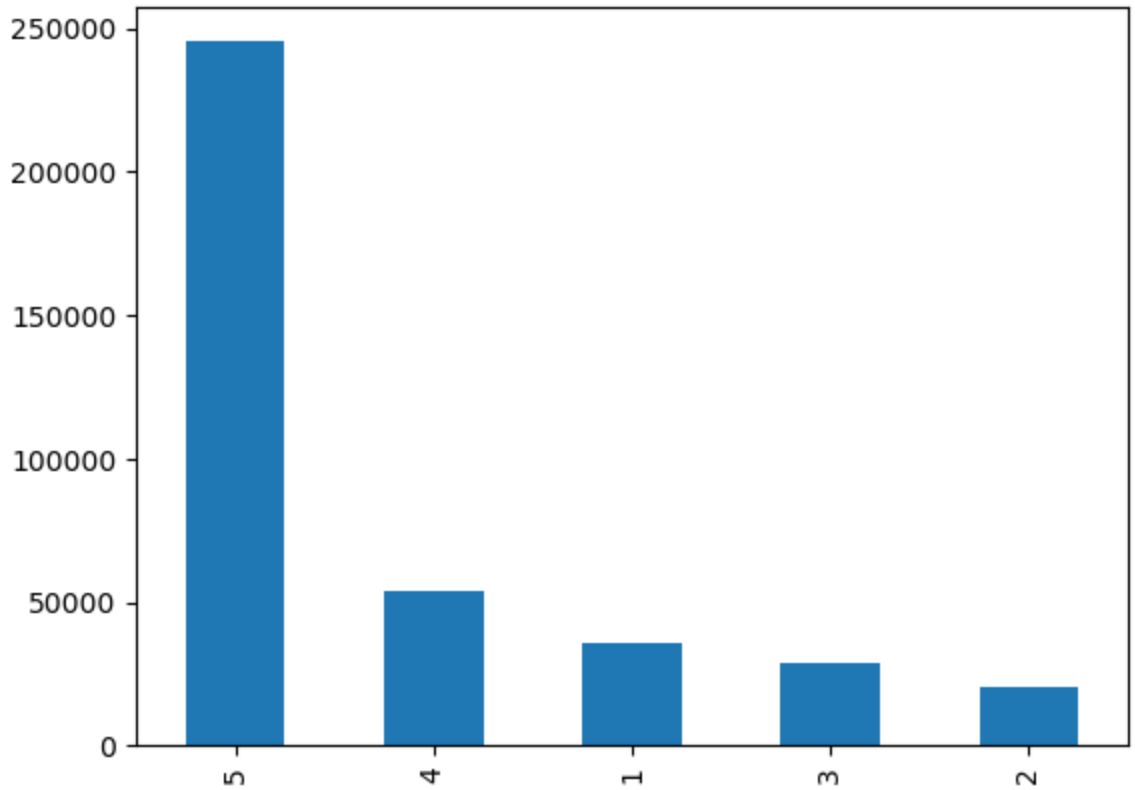
In [103…
```
freq_df['Score'].value_counts().plot(kind="bar")
```

Out[103]:
```
<Axes: >
```



In [108…
```
not_freq_df['Score'].value_counts().plot(kind="bar")
```

Out[108]:
```
<Axes: >
```



In [ ]:
```
#Analysing your frequent Users ! Are frequent users more verbose?
```

In [109…
```
data.columns
```

Out[109]:
```
Index(['Id', 'ProductId', 'UserId', 'ProfileName', 'HelpfulnessNumerator',
       'HelpfulnessDenominator', 'Score', 'Time', 'Summary', 'Text',
       'viewer_type'],
      dtype='object')
```

In [113…
```
data[['UserId','ProductId','Text']]
```

Out[113]:

	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 [114…
```
data['Text'][0]
```

Out[114]:
```
'I have bought several of the Vitality canned dog food products and have found them all
to be of good quality. The product looks more like a stew than a processed meat and it s
mells better. My Labrador is finicky and she appreciates this product better than  mos
t.'
```

In [117…
```
type(data['Text'][0])
```

Out[117]:
```
str
```

In [118…
```
len(data['Text'][0].split(' '))
```

Out[118]:
```
49
```

In [119…
```
def calculate_length(text):
    return len(text.split(' '))
```

In [120…
```
data['Text'].apply(calculate_length)
```

Out[120]:
```
0         49
1         31
2         99
3         43
4         30
          ..
568449    26
568450    46
568451    71
568452    37
```

```
568453    21
Name: Text, Length: 393931, dtype: int64
```

In [122…
```
data['Text_length'] = data['Text'].apply(calculate_length)
```

```
C:\Users\usr\AppData\Local\Temp\ipykernel_25328\1567403007.py:1: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data['Text_length'] = data['Text'].apply(calculate_length)
```

In [121…
```
data['viewer_type'].unique()
```

Out[121]:
```
array(['Not frequent', 'Frequent'], dtype=object)
```

In [123…
```
not_freq_data = data[data['viewer_type']=='Not frequent']
freq_data = data[data['viewer_type']=='Frequent']
```

In [124…
```
not_freq_data
```

Out[124]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1
3	4	B000UA0QIQ	A395BORC6FGVXV	Karl	3	3
4	5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	0	0
...	...	...	...	...	...	...
568449	568450	B001EO7N10	A28KG5XORO54AY	Lettie D. Carter	0	0
568450	568451	B003S1WTCU	A3I8AFVPEE8KI5	R. Sawyer	0	0

568451	568452	B004I613EE	A121AA1GQV751Z	pksd "pk_007"	2	2
568452	568453	B004I613EE	A3IBEVCTXKNOH	Kathy A. Welch "katwel"	1	1
568453	568454	B001LR2CU2	A3LGQPJCZVL9UC	srfell17	0	0

384573 rows × 12 columns

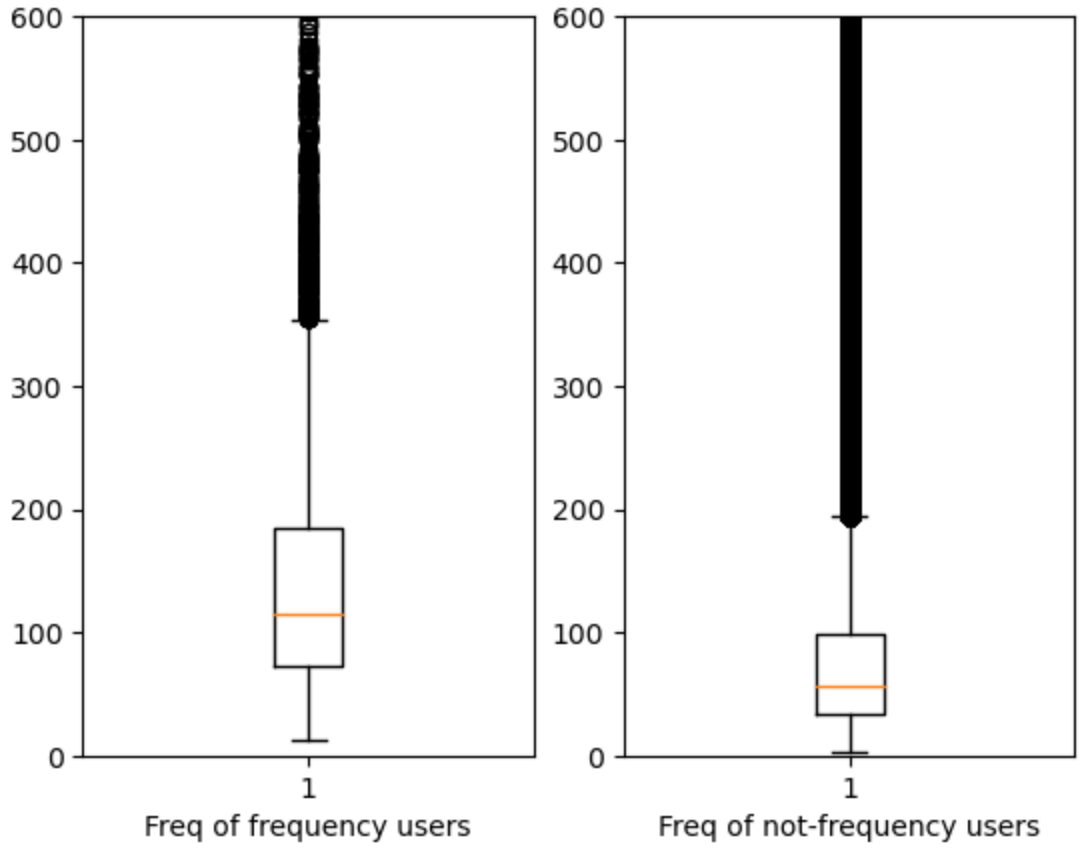
In [ ]:

In [ ]:

In [128…
```
fig = plt.figure()
ax1 = fig.add_subplot(121)
ax1.boxplot(freq_data['Text_length'])
ax1.set_xlabel('Freq of frequency users')
ax1.set_ylim(0,600)


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

Out[128]:
```
(0.0, 600.0)
```



In [ ]:
```
#Sentiment analysis
```

In [134…
```
!pip install textblob
from textblob import TextBlob
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: textblob in c:\users\usr\appdata\roaming\python\python310
\site-packages (0.17.1)
Requirement already satisfied: nltk>=3.1 in c:\programdata\anaconda3\lib\site-packages
(from textblob) (3.7)
Requirement already satisfied: regex>=2021.8.3 in c:\programdata\anaconda3\lib\site-pack
ages (from nltk>=3.1->textblob) (2022.7.9)
Requirement already satisfied: joblib in c:\programdata\anaconda3\lib\site-packages (fro
m nltk>=3.1->textblob) (1.1.1)
Requirement already satisfied: tqdm in c:\programdata\anaconda3\lib\site-packages (from
nltk>=3.1->textblob) (4.64.1)
Requirement already satisfied: click in c:\programdata\anaconda3\lib\site-packages (from
nltk>=3.1->textblob) (8.0.4)
Requirement already satisfied: colorama in c:\programdata\anaconda3\lib\site-packages (f
rom click->nltk>=3.1->textblob) (0.4.6)
```

In [133…
```
data['Summary'][0]
```

Out[133]:
```
'Good Quality Dog Food'
```

In [136…
```
TextBlob('Good Quality Dog Food').sentiment.polarity
```

Out[136]:
```
0.7
```

In [138…
```
data.shape
```

Out[138]:
```
(393931, 12)
```

In [139…
```
sample = data[0:50000]
```

In [ ]:

In [141…
```
polarity =[]
for text in sample['Summary']:
    try:
        polarity.append(TextBlob(text).sentiment.polarity)
    except:
        polarity.append(0)
```

In [142…
```
len(polarity)
```

Out[142]:
```
50000
```

In [144…
```
sample['polarity'] = polarity
```

```
C:\Users\usr\AppData\Local\Temp\ipykernel_25328\4182253960.py:1: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  sample['polarity'] = polarity
```

In [145…
```
sample.head(10)
```

Out[145]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0	1
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1	4
3	4	B000UA0QIQ	A395BORC6FGVXV	Karl	3	3	2
4	5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	0	0	5
5	6	B006K2ZZ7K	ADT0SRK1MGOEU	Twoapennything	0	0	4

6	7	B006K2ZZ7K	A1SP2KVKFXXRU1	David C. Sullivan	0	0	5
7	8	B006K2ZZ7K	A3JRGQVEQN31IQ	Pamela G. Williams	0	0	5
8	9	B000E7L2R4	A1MZYO9TZK0BBI	R. James	1	1	5
9	10	B00171APVA	A21BT40VZCCYT4	Carol A. Reed	0	0	5

In [149…
```
sample_negative = sample[sample['polarity']<0]
sample_positive = sample[sample['polarity']>0]
```

In [151…
```
from collections import Counter
```

In [152…
```
Counter(sample_negative['Summary']).most_common(10)
```

Out[152]:
```
[('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 [153…
```
Counter(sample_positive['Summary']).most_common(10)
```

Out[153]:
```
[('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 [ ]:
```
```