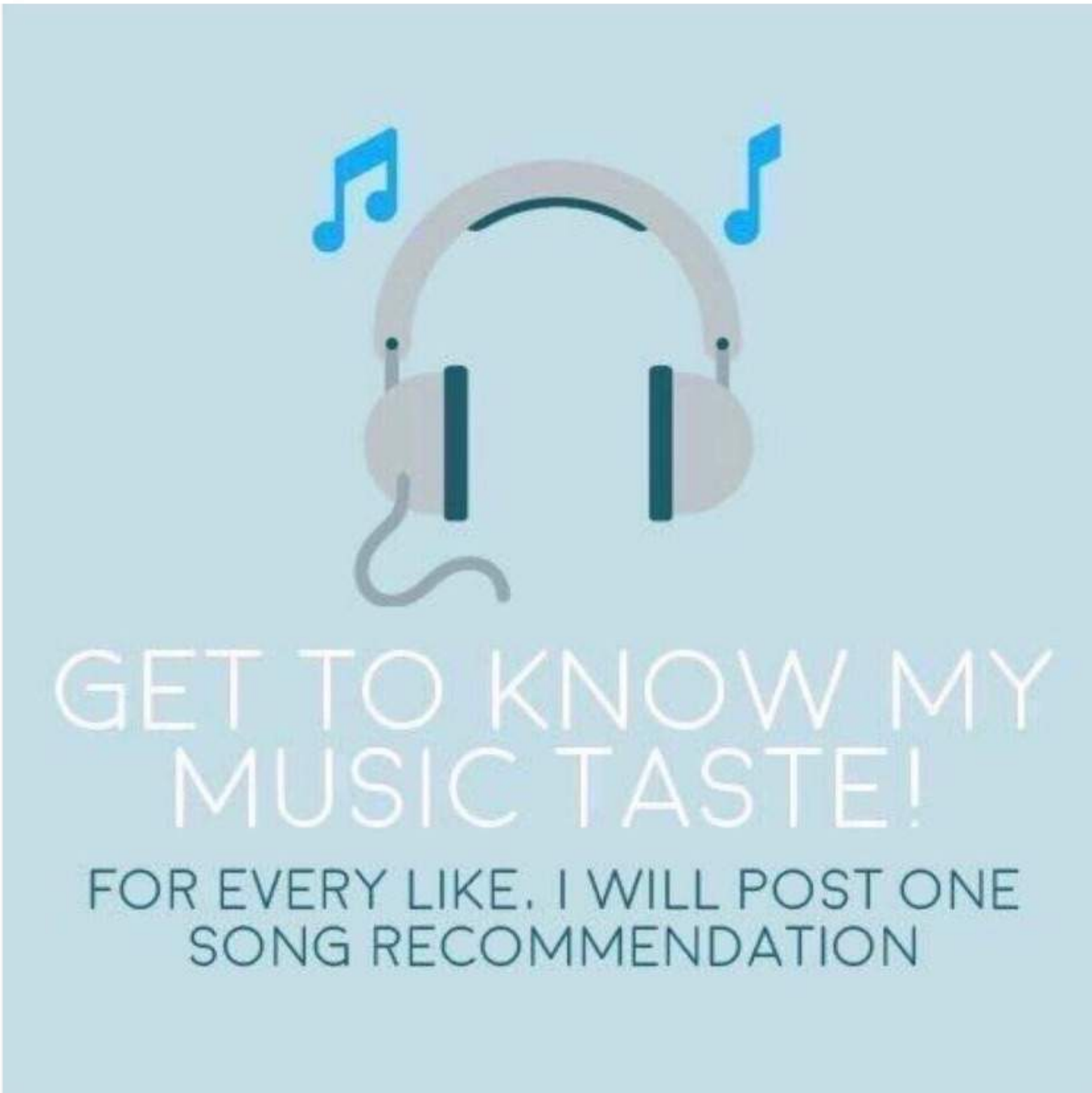
Song-Recommendation-ML




```
~~~Image has been taken from Google Image.
```

Recommendation of a song for the listener nased on gender, age, region, artist they like and many more.

Let's connect our jupyter notebook to jovian.

Problem Statement

I selected the 15th data set from the resources tab in Jovian. Link from where I downloaded the dataset: https://www.kaggle.com/c/MusicHackathon/data

This data has ratings given by the listeners, qualitative feedback, answers to the question on music and listeners demographics. We will use this dataset to get the rating of the test dataset.

It is a Regression type problem.

Installing the required libraries for making the model

```
!pip install plotly==5.11.0
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/publ
Collecting plotly==5.11.0
  Downloading plotly-5.11.0-py2.py3-none-any.whl (15.3 MB)
     |████████████████████████████████| 15.3 MB 4.8 MB/s
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.8/dist-packages
Installing collected packages: plotly
  Attempting uninstall: plotly
    Found existing installation: plotly 5.5.0
    Uninstalling plotly-5.5.0:
      Successfully uninstalled plotly-5.5.0
Successfully installed plotly-5.11.0
```

```
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
%matplotlib inline

sns.set_style('darkgrid')
matplotlib.rcParams['font.size'] = 16
matplotlib.rcParams['figure.figsize'] = (14, 10)
matplotlib.rcParams['figure.facecolor'] = '#00000000'
```

```
!pip install opendatasets
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/publ
Collecting opendatasets
  Downloading opendatasets-0.1.22-py3-none-any.whl (15 kB)
Requirement already satisfied: tqdm in /usr/local/lib/python3.8/dist-packages (from oper
Requirement already satisfied: kaggle in /usr/local/lib/python3.8/dist-packages (from op
Requirement already satisfied: click in /usr/local/lib/python3.8/dist-packages (from ope
Requirement already satisfied: python-slugify in /usr/local/lib/python3.8/dist-packages
```

```
Requirement already satisfied: certifi in /usr/local/lib/python3.8/dist-packages (from k
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.8/dist-packages (fron
Requirement already satisfied: requests in /usr/local/lib/python3.8/dist-packages (from
Requirement already satisfied: urllib3 in /usr/local/lib/python3.8/dist-packages (from k
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.8/dist-packages
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.8/dist-pack
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.8/dist-packag
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.8/dist-packages (f
Installing collected packages: opendatasets
Successfully installed opendatasets-0.1.22
```

```
import os
import opendatasets as od
import pandas as pd
import numpy as np
pd.set_option("display.max_columns", 120)
pd.set_option("display.max_rows", 120)
```

Downloading data set from Kaggle in the notebook

```
od.download('https://www.kaggle.com/c/MusicHackathon/data')
```

```
Downloading MusicHackathon.zip to ./MusicHackathon
100%|██████████| 6.62M/6.62M [00:00<00:00, 47.9MB/s]
Extracting archive ./MusicHackathon/MusicHackathon.zip to ./MusicHackathon
```

```
os.listdir('MusicHackathon')
```

```
['UserKey.csv',
 'global_mean_benchmark.csv',
 'words.csv',
 'tracks_mean_benchmark.csv',
 'sample.r',
 'artists_mean_benchmark.csv',
 'users_mean_benchmark.csv',
 'test.csv',
 'logo_greenplum_main.png',
 'users.csv',
 'train.csv']
```

Converting the dataset to dataframe

```
train_df = pd.read_csv('./MusicHackathon/train.csv')
test_df = pd.read_csv('./MusicHackathon/test.csv')
words_df = pd.read_csv('./MusicHackathon/words.csv', encoding = "ISO-8859-1")
users_df = pd.read_csv('./MusicHackathon/users.csv')
```

```
train_df
```

	Artist	Track	User	Rating	Time
0	40	179	47994	9	17
1	9	23	8575	58	7
2	46	168	45475	13	16
3	11	153	39508	42	15
4	14	32	11565	54	19
...	...	...	...	...	...
188685	0	3	1278	29	6
188686	1	6	2839	30	18
188687	10	142	35756	61	12
188688	22	54	20163	46	21
188689	47	171	45580	12	4

188690 rows × 5 columns

```
test_df
```

	Artist	Track	User	Time
0	1	6	3475	18
1	6	149	39210	15
2	40	177	47861	17
3	31	79	27413	11
4	26	66	23232	22
...	...	...	...	...
125789	14	95	30004	23
125790	10	25	8186	7
125791	40	146	38180	13
125792	22	113	32918	0
125793	2	70	24231	22

125794 rows × 4 columns

```
words_df
```

	Artist	User	HEARD_OF	OWN_ARTIST_MUSIC	LIKE_ARTIST	Uninspired	Sophistica
0	47	45969	Heard of	NaN	NaN	NaN	
1	35	29118	Never heard of	NaN	NaN	0.0	N
2	14	31544	Heard of	NaN	NaN	0.0	N
3	23	18085	Never heard of	NaN	NaN	NaN	N
4	23	18084	Never heard of	NaN	NaN	NaN	N
...	...	...	...	...	...	...	
118296	4	3932	Heard of and listened to music EVER	Own a little of their music	26.0	NaN	N
118297	4	3935	Heard of and listened to music EVER	Own a little of their music	30.0	NaN	N
118298	12	11216	Heard of and listened to music RECENTLY	Own none of their music	71.0	NaN	N
118299	33	35142	Heard of and listened to music EVER	Own none of their music	31.0	NaN	N
118300	4	3915	Heard of and listened to music EVER	Own a little of their music	46.0	NaN	N

118301 rows × 88 columns

```
users_df
```

RESPID	GENDER	AGE	WORKING	REGION	MUSIC	LIST_OWN	LIST_BACK	Q1
					Music is			

```
words_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 118301 entries, 0 to 118300
Data columns (total 88 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   Artist            118301 non-null  int64  
 1   User              118301 non-null  int64  
 2   HEARD_OF          118277 non-null  object 
 3   OWN_ARTIST_MUSIC  33507 non-null   object 
 4   LIKE_ARTIST       33308 non-null   float64
 5   Uninspired        26154 non-null   float64
 6   Sophisticated     20724 non-null   float64
 7   Aggressive        97577 non-null   float64
 8   Edgy              118301 non-null  int64  
 9   Sociable          20724 non-null   float64
 10  Laid back         20724 non-null   float64
 11  Wholesome         1040 non-null    float64
 12  Uplifting         20724 non-null   float64
 13  Intriguing        20724 non-null   float64
 14  Legendary         1040 non-null    float64
 15  Free              20724 non-null   float64
 16  Thoughtful        118301 non-null  int64  
 17  Outspoken         20724 non-null   float64
 18  Serious           97577 non-null   float64
 19  Good lyrics       97577 non-null   float64
 20  Unattractive      97577 non-null   float64
 21  Confident         97577 non-null   float64
 22  Old               1040 non-null    float64
 23  Youthful          117261 non-null  float64
 24  Boring            87080 non-null   float64
 25  Current           118301 non-null  int64  
 26  Colourful         20724 non-null   float64
 27  Stylish           118301 non-null  int64  
 28  Cheap             97577 non-null   float64
 29  Irrelevant        26154 non-null   float64
 30  Heartfelt         20724 non-null   float64
 31  Calm              97577 non-null   float64
 32  Pioneer           1040 non-null    float64
 33  Outgoing          97577 non-null   float64
 34  Inspiring         97577 non-null   float64
 35  Beautiful         118301 non-null  int64  
 36  Fun               118301 non-null  int64  
 37  Authentic         118301 non-null  int64  
 38  Credible          118301 non-null  int64  
 39  Way out           20724 non-null   float64
 40  Cool              118301 non-null  int64  
 41  Catchy            117261 non-null  float64
 42  Sensitive         97577 non-null   float64
 43  Mainstream        46254 non-null   float64
 44  Superficial       97577 non-null   float64
```

```
 45  Annoying          26154 non-null   float64
 46  Dark              1040 non-null    float64
 47  Passionate        118301 non-null  int64
 48  Not authentic     26154 non-null   float64
 49  Good Lyrics       20724 non-null   float64
 50  Background        20724 non-null   float64
 51  Timeless          118301 non-null  int64
 52  Depressing        97577 non-null   float64
```

Score to words DF

Now i will be giving score to 'words_df' by preprocessing the df.

The score system works like this:

- For each value 1 in the positive columns, we **add 1 point to the total score**
- For each value 1 in the negative columns, we **subtract 1 point to the total score**
- Any 0 and NaN value we **ignore as they are neutral**

```
positive_score = ['Sophisticated', 'Sociable', 'Laid back', 'Wholesome', 'Uplifting', 'Intrig
```

```
negative_score = ['Uninspired', 'Unattractive', 'Boring', 'Cheap', 'Irrelevant', 'Superficial
```

```
words_df['plus_score'] = words_df[positive_score].sum(axis=1)
words_df['minus_score'] = words_df[negative_score].sum(axis=1)
words_df['words_score'] = words_df['plus_score'] - words_df['minus_score']
```

```
words_df[words_df.LIKE_ARTIST > 90].sample(15)
```

	Artist	User	HEARD_OF	OWN_ARTIST_MUSIC	LIKE_ARTIST	Uninspired	Sophistica
109843	4	38089	Heard of and listened to music RECENTLY	Own all or most of their music	100.0	NaN	N
28247	41	42540	Heard of and listened to music RECENTLY	Own all or most of their music	95.0	NaN	N
104792	4	36390	Heard of and listened to music RECENTLY	Own all or most of their music	91.0	NaN	N
57761	17	14331	Heard of and listened to music RECENTLY	Own a lot of their music	91.0	NaN	
20595	32	25628	Heard of and listened to music RECENTLY	Own a lot of their music	100.0	NaN	N
62408	40	36255	Heard of and listened to music RECENTLY	Own all or most of their music	93.0	NaN	N

As now we gave the word score we don't need the words columns in the words_df dataframe. Now we will create a dateframe where the columns will be the **word score of above 90**

music

```
words_red_df = words_df[['Artist', 'User', 'HEARD_OF', 'OWN_ARTIST_MUSIC', 'LIKE_ARTIST', 'wo
```

Heard of

```
words_red_df
```

	Artist	User	HEARD_OF	OWN_ARTIST_MUSIC	LIKE_ARTIST	words_score
0	47	45969	Heard of	NaN	NaN	-1.0
1	35	29118	Never heard of	NaN	NaN	3.0
2	14	31544	Heard of	NaN	NaN	2.0
3	23	18085	Never heard of	NaN	NaN	-1.0
4	23	18084	Never heard of	NaN	NaN	0.0
...	...	...	...	...	...	...
118296	4	3932	Heard of and listened to music EVER	Own a little of their music	26.0	-1.0
[illegible]	[illegible]	[illegible]	Heard of and listened	Own a little of their	[illegible]	[illegible]

```
words_red_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 118301 entries, 0 to 118300
Data columns (total 6 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   Artist            118301 non-null  int64  
 1   User              118301 non-null  int64  
 2   HEARD_OF          118277 non-null  object 
 3   OWN_ARTIST_MUSIC  33507 non-null   object 
 4   LIKE_ARTIST       33308 non-null   float64
 5   words_score       118301 non-null  float64
dtypes: float64(2), int64(2), object(2)
memory usage: 5.4+ MB
```

Merging

Now we will merge words_red_df & users_df into training_merge_df dataframe

```
users_df.rename(columns={'RESPID': 'User'}, inplace=True)
```

```
training_merge_df = train_df.merge(words_red_df, how='left', on=['Artist', 'User'])
```

```
users_df
```

	User	GENDER	AGE	WORKING	REGION	MUSIC	LIST_OWN	LIST_BACK	Q1
0	36927	Female	60.0	Other	South	Music is important to me but not necessarily m...	1 hour	NaN	49.0
1	3566	Female	36.0	Full-time housewife / househusband	South	Music is important to me but not necessarily m...	1 hour	1 hour	55.0
2	20054	Female	52.0	Employed 30+ hours a week	Midlands	I like music but it does not feature heavily i...	1 hour	Less than an hour	11.0
3	41749	Female	40.0	Employed 8-29 hours per week	South	Music means a lot to me and is a passion of mine	2 hours	3 hours	81.0
4	23108	Female	16.0	Full-time student	North	Music means a lot to me and is a passion of mine	3 hours	6 hours	76.0
...	...	...	...	...	...	...	...	...	...
48640	19361	Male	48.0	Self-employed	Midlands	I like music but it does not feature heavily i...	Less than an hour	2 hours	9.0
48641	17639	Female	60.0	Full-time housewife / househusband	Midlands	Music means a lot to me and is a passion of mine	2 hours	1 hour	26.0
				Employed 30+		Music means a lot to me			

```
training_merge_df
```

	Artist	Track	User	Rating	Time	HEARD_OF	OWN_ARTIST_MUSIC	LIKE_ARTIST	w
0	40	179	47994	9	17	Never heard of	NaN	NaN	
1	9	23	8575	58	7	Never heard of	NaN	NaN	
2	46	168	45475	13	16	Never heard of	NaN	NaN	
3	11	153	39508	42	15	Heard of and listened to music EVER	Own none of their music	28.0	
4	14	32	11565	54	19	Heard of and listened to music EVER	Own none of their music	18.0	

```
training_merge_df = training_merge_df.merge(users_df, how='left', on=['User'])
```

```
training_merge_df
```

	Artist	Track	User	Rating	Time	HEARD_OF	OWN_ARTIST_MUSIC	LIKE_ARTIST
0	40	179	47994	9	17	Never heard of	NaN	NaN
1	9	23	8575	58	7	Never heard of	NaN	NaN
2	46	168	45475	13	16	Never heard of	NaN	NaN
3	11	153	39508	42	15	Heard of and listened to music EVER	Own none of their music	28.0

```
training_merge_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 188690 entries, 0 to 188689
Data columns (total 35 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   Artist            188690 non-null  int64  
 1   Track             188690 non-null  int64  
 2   User              188690 non-null  int64  
 3   Rating            188690 non-null  int64  
 4   Time              188690 non-null  int64  
 5   HEARD_OF          186418 non-null  object 
 6   OWN_ARTIST_MUSIC  56835 non-null   object 
 7   LIKE_ARTIST       55028 non-null   float64
 8   words_score       186636 non-null  float64
 9   GENDER            176833 non-null  object 
 10  AGE               174982 non-null  float64
 11  WORKING           140545 non-null  object 
 12  REGION            167481 non-null  object 
 13  MUSIC             176833 non-null  object 
 14  LIST_OWN          158651 non-null  object 
 15  LIST_BACK         158790 non-null  object 
 16  Q1                176833 non-null  float64
 17  Q2                176833 non-null  float64
 18  Q3                176833 non-null  float64
```

```
 19  Q4                 176833 non-null  float64
 20  Q5                 176833 non-null  float64
 21  Q6                 176833 non-null  float64
 22  Q7                 176833 non-null  float64
 23  Q8                 176833 non-null  float64
 24  Q9                 176833 non-null  float64
 25  Q10                176833 non-null  float64
 26  Q11                176833 non-null  float64
 27  Q12                176833 non-null  float64
 28  Q13                176833 non-null  float64
 29  Q14                176833 non-null  float64
 30  Q15                176833 non-null  float64
 31  Q16                142754 non-null  float64
 32  Q17                176833 non-null  float64
 33  Q18                140545 non-null  float64
 34  Q19                140545 non-null  float64
dtypes: float64(22), int64(5), object(8)
memory usage: 51.8+ MB
```

```
training_merge_df.sample(15)
```

	Artist	Track	User	Rating	Time	HEARD_OF	OWN_ARTIST_MUSIC	LIKE_ARTIST
52030	35	91	30678	58	23	Never heard of	NaN	NaN
175760	4	12	5536	89	18	Heard of and listened to music EVER	Own a lot of their music	72.0
164836	11	29	9928	12	7	Never heard of	NaN	NaN

Merging the test dataset

```
test_merge_df = test_df.merge(words_red_df, how='left', on=['Artist', 'User'])
test_merge_df = test_merge_df.merge(users_df, how='left', on=['User'])
```

```
test_merge_df
```

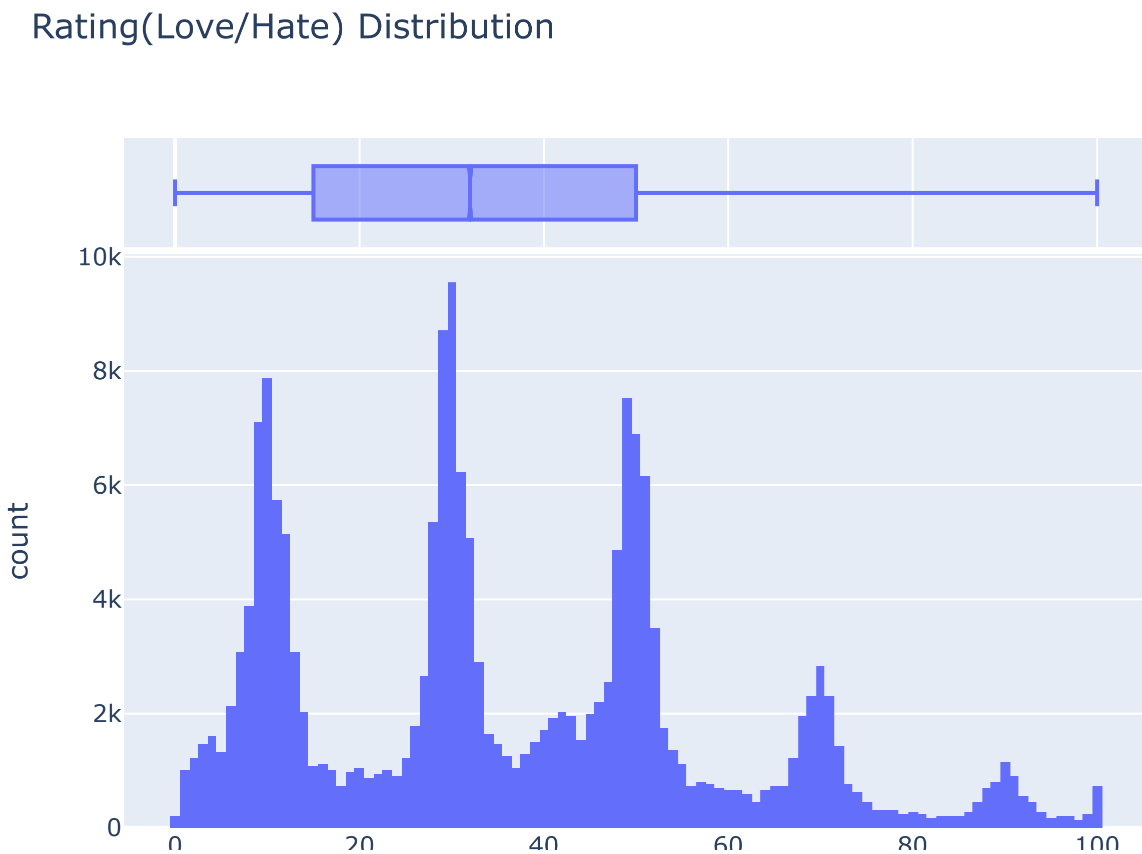
	Artist	Track	User	Time	HEARD_OF	OWN_ARTIST_MUSIC	LIKE_ARTIST	words_sco
0	1	6	3475	18	Heard of and listened to music EVER	Own none of their music	3.0	
1	6	149	39210	15	NaN	NaN	NaN	N
2	40	177	47861	17	Never heard of	NaN	NaN	-
3	31	79	27413	11	Never heard of	NaN	NaN	
4	26	66	23232	22	Never heard of	NaN	NaN	
...	...	...	...	...	...	...	...	
125789	14	95	30004	23	Heard of	NaN	NaN	1

Data Analysis

Now we will try to get the insights from the dataset and see if there is any relationship between the columns. We must also check if any of the columns are interdependent. We ask Question and then we visualize the dataset to get the Answer.

We can do this by plotting the graphs for various columns and observing the relation between the two or more columns depending on the plot we choose.

```
# Do you love or hate the the song?
px.histogram(training_merge_df, x='Rating', nbins=101, marginal='box', title='Rating(Love/Hat
```

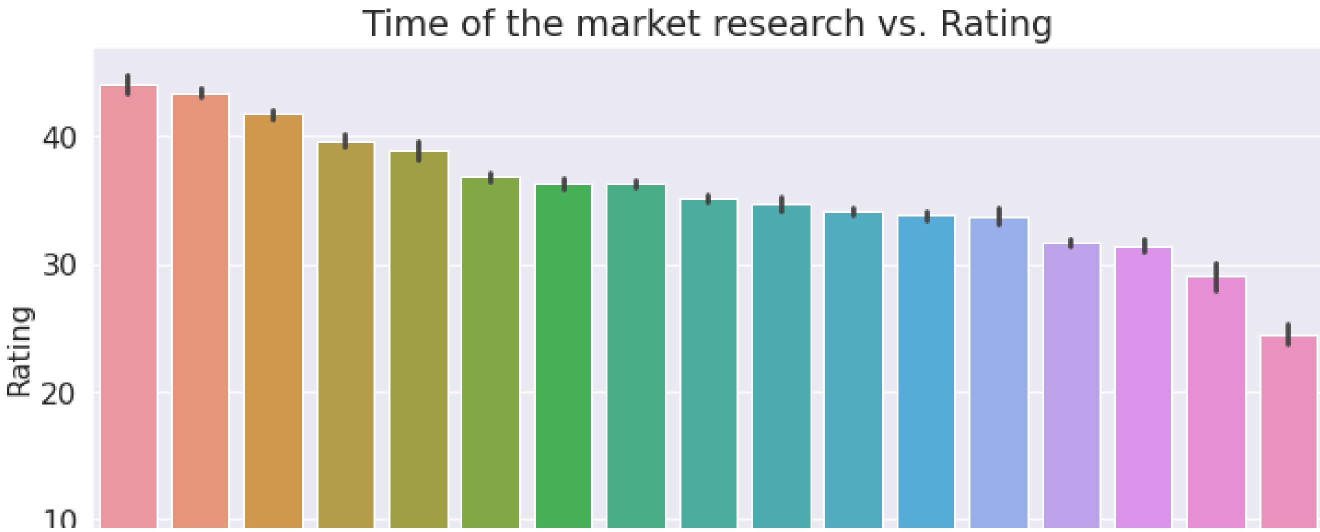



```
training_merge_df.columns
```

```
Index(['Artist', 'Track', 'User', 'Rating', 'Time', 'HEARD_OF',
       'OWN_ARTIST_MUSIC', 'LIKE_ARTIST', 'words_score', 'GENDER', 'AGE',
       'WORKING', 'REGION', 'MUSIC', 'LIST_OWN', 'LIST_BACK', 'Q1', 'Q2', 'Q3',
       'Q4', 'Q5', 'Q6', 'Q7', 'Q8', 'Q9', 'Q10', 'Q11', 'Q12', 'Q13', 'Q14',
       'Q15', 'Q16', 'Q17', 'Q18', 'Q19'],
      dtype='object')
```

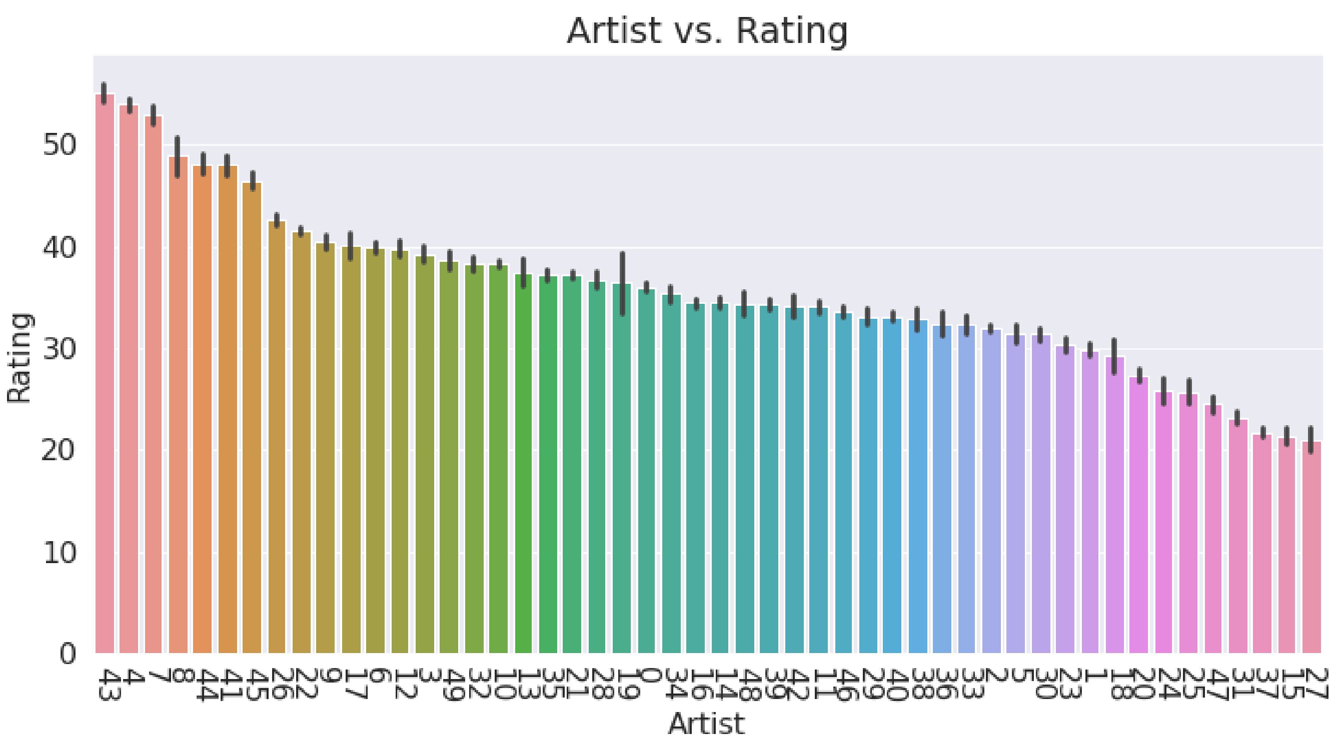
```
plot_order= training_merge_df.groupby('Time')['Rating'].mean().sort_values(ascending=False).i
```

```
fig, ax = plt.subplots(figsize=(12,6))

plt.title('Time of the market research vs. Rating')
sns.barplot(x='Time', y='Rating', data=training_merge_df, order=plot_order)
plt.xticks(rotation=0, ha='center')
plt.show();
```



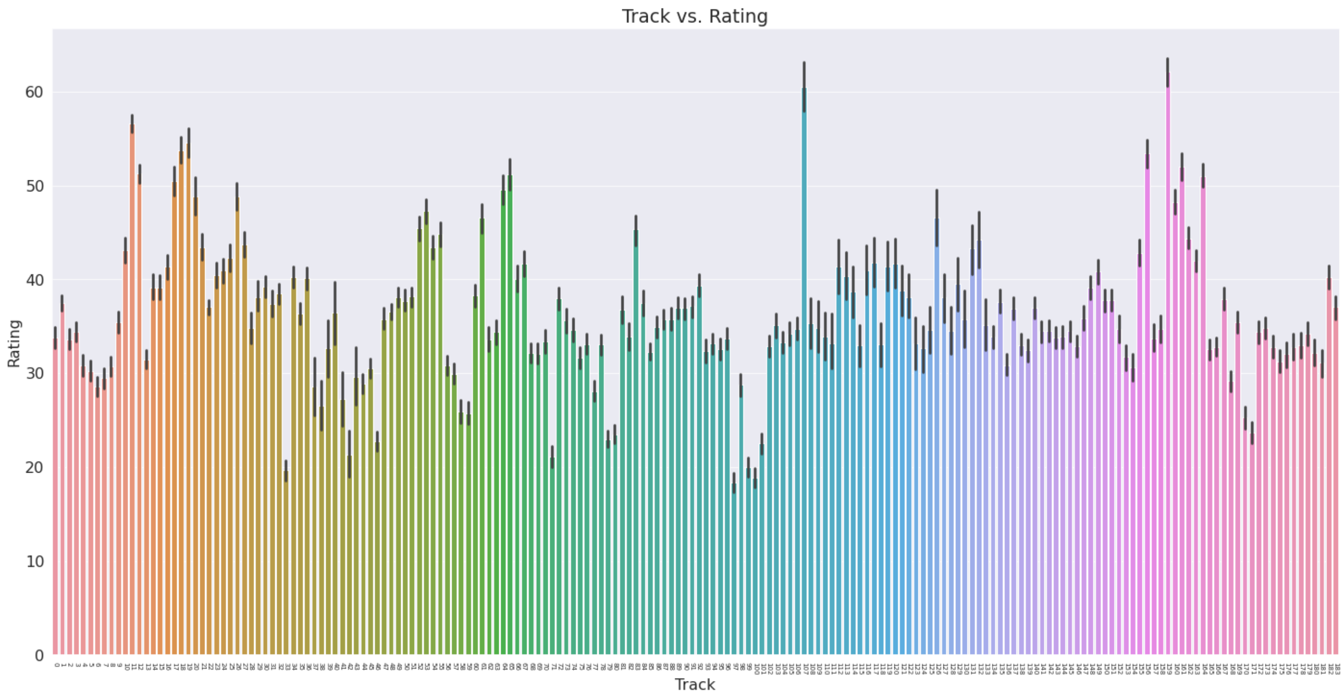

```
plot_order= training_merge_df.groupby('Artist')['Rating'].mean().sort_values(ascending=False)
```

```
fig, ax = plt.subplots(figsize=(12,6))

plt.title('Artist vs. Rating')
sns.barplot(x='Artist', y='Rating', data=training_merge_df, order=plot_order)
plt.xticks(rotation=270, ha='center')
plt.show();
```




```
fig, ax = plt.subplots(figsize=(24,12))

plt.title('Track vs. Rating')
```

```
sns.barplot(x='Track', y='Rating', data=training_merge_df)
plt.xticks(rotation=-90, fontsize=7, ha='center')
plt.show();
```



Change of columns

```
training_merge_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 188690 entries, 0 to 188689
Data columns (total 35 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   Artist           188690 non-null  int64
 1   Track            188690 non-null  int64
```

```
 2   User              188690 non-null  int64
 3   Rating            188690 non-null  int64
 4   Time              188690 non-null  int64
 5   HEARD_OF          186418 non-null  object
 6   OWN_ARTIST_MUSIC  56835 non-null   object
 7   LIKE_ARTIST       55028 non-null   float64
 8   words_score       186636 non-null  float64
 9   GENDER            176833 non-null  object
 10  AGE               174982 non-null  float64
 11  WORKING           140545 non-null  object
 12  REGION            167481 non-null  object
 13  MUSIC             176833 non-null  object
 14  LIST_OWN          158651 non-null  object
 15  LIST_BACK         158790 non-null  object
 16  Q1                176833 non-null  float64
 17  Q2                176833 non-null  float64
 18  Q3                176833 non-null  float64
 19  Q4                176833 non-null  float64
 20  Q5                176833 non-null  float64
 21  Q6                176833 non-null  float64
 22  Q7                176833 non-null  float64
 23  Q8                176833 non-null  float64
 24  Q9                176833 non-null  float64
 25  Q10               176833 non-null  float64
 26  Q11               176833 non-null  float64
 27  Q12               176833 non-null  float64
 28  Q13               176833 non-null  float64
 29  Q14               176833 non-null  float64
 30  Q15               176833 non-null  float64
 31  Q16               142754 non-null  float64
 32  Q17               176833 non-null  float64
 33  Q18               140545 non-null  float64
 34  Q19               140545 non-null  float64
dtypes: float64(22), int64(5), object(8)
memory usage: 55.9+ MB
```

```
training_merge_df['HEARD_OF'].value_counts()
```

```
Never heard of                              94090
Heard of                                    35493
Heard of and listened to music EVER         29854
Heard of and listened to music RECENTLY     17847
Ever heard music by                          5136
Listened to recently                         2191
Ever heard of                                1807
Name: HEARD_OF, dtype: int64
```

```
print('Missing values in HEARD_OF column {}'.format(training_merge_df['HEARD_OF'].isna().sum(
```

```
Missing values in HEARD_OF column 2272
```

```
training_merge_df['HEARD_OF'].replace(['Ever heard of'], 'Never heard of', inplace=True)
training_merge_df['HEARD_OF'].replace(['Ever heard music by'], 'Heard of and listened to musi
```

```
training_merge_df['HEARD_OF'].replace(['Listened to recently'], 'Heard of and listened to mus
training_merge_df['HEARD_OF'].fillna('Never heard of', inplace=True)
```

```
training_merge_df['HEARD_OF'].unique()
```

```
array(['Never heard of', 'Heard of and listened to music EVER',
       'Heard of', 'Heard of and listened to music RECENTLY'],
      dtype=object)
```

```
test_merge_df['HEARD_OF'].replace(['Ever heard of'], 'Never heard of', inplace=True)
test_merge_df['HEARD_OF'].replace(['Ever heard music by'], 'Heard of and listened to music EV
test_merge_df['HEARD_OF'].replace(['Listened to recently'], 'Heard of and listened to music R
test_merge_df['HEARD_OF'].fillna('Never heard of', inplace=True)
```

```
test_merge_df['HEARD_OF'].unique()
```

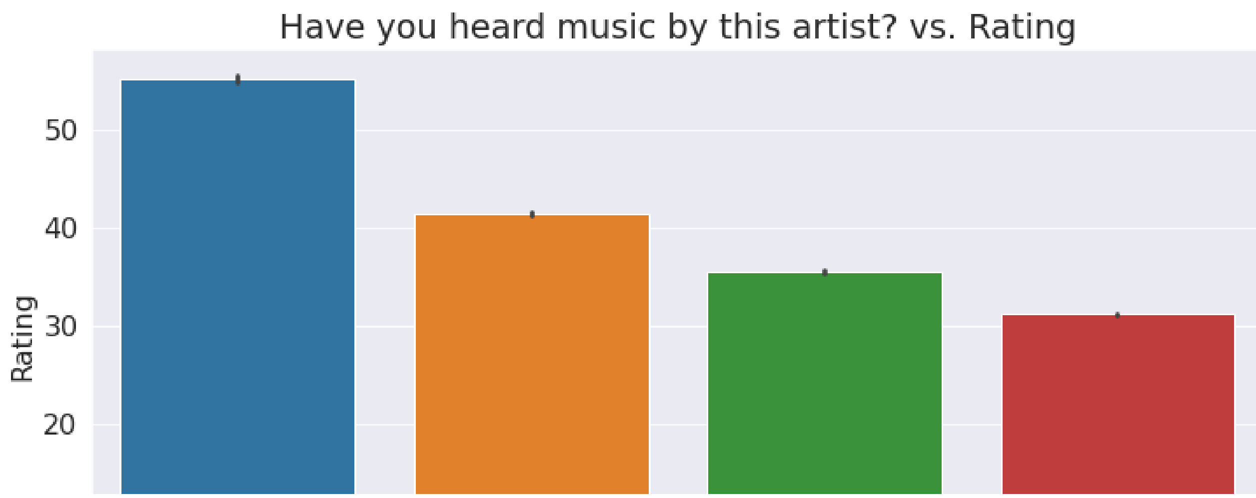
```
array(['Heard of and listened to music EVER', 'Never heard of',
       'Heard of', 'Heard of and listened to music RECENTLY'],
      dtype=object)
```

```
training_merge_df['HEARD_OF'].value_counts()
```

```
Never heard of                             98169
Heard of                                   35493
Heard of and listened to music EVER        34990
Heard of and listened to music RECENTLY    20038
Name: HEARD_OF, dtype: int64
```

```
plot_order= training_merge_df.groupby('HEARD_OF')['Rating'].mean().sort_values(ascending=Fals
```

```
fig, ax = plt.subplots(figsize=(12,6))

plt.title('Have you heard music by this artist? vs. Rating')
sns.barplot(x='HEARD_OF', y='Rating', data=training_merge_df, order=plot_order)
plt.xticks(rotation=350, ha='left')
plt.show();
```



Own_Artist_Music


```
training_merge_df['OWN_ARTIST_MUSIC'].unique()
```

```
array([nan, 'Own none of their music', 'Own a little of their music',
       'Own all or most of their music', 'DonÕt know',
       'Own a lot of their music', 'DonÍt know', 'don`t know'],
      dtype=object)
```

```
training_merge_df['OWN_ARTIST_MUSIC'].value_counts()
```

```
Own none of their music           26810
Own a little of their music       18721
Own a lot of their music           7263
Own all or most of their music     2593
DonÕt know                         1265
DonÍt know                          147
don`t know                           36
Name: OWN_ARTIST_MUSIC, dtype: int64
```

```
training_merge_df['OWN_ARTIST_MUSIC'].replace(['DonÕt know'], 'Own none of their music', inpl
training_merge_df['OWN_ARTIST_MUSIC'].replace(['DonÍt know'], 'Own none of their music', inpl
training_merge_df['OWN_ARTIST_MUSIC'].replace(['don`t know'], 'Own none of their music', inpl
training_merge_df['OWN_ARTIST_MUSIC'].fillna('Own none of their music', inplace=True)
```

```
test_merge_df['OWN_ARTIST_MUSIC'].replace(['DonÕt know'], 'Own none of their music', inplace=
test_merge_df['OWN_ARTIST_MUSIC'].replace(['DonÍt know'], 'Own none of their music', inplace=
test_merge_df['OWN_ARTIST_MUSIC'].replace(['don`t know'], 'Own none of their music', inplace=
test_merge_df['OWN_ARTIST_MUSIC'].fillna('Own none of their music', inplace=True)
```

```
training_merge_df['OWN_ARTIST_MUSIC'].unique()
```

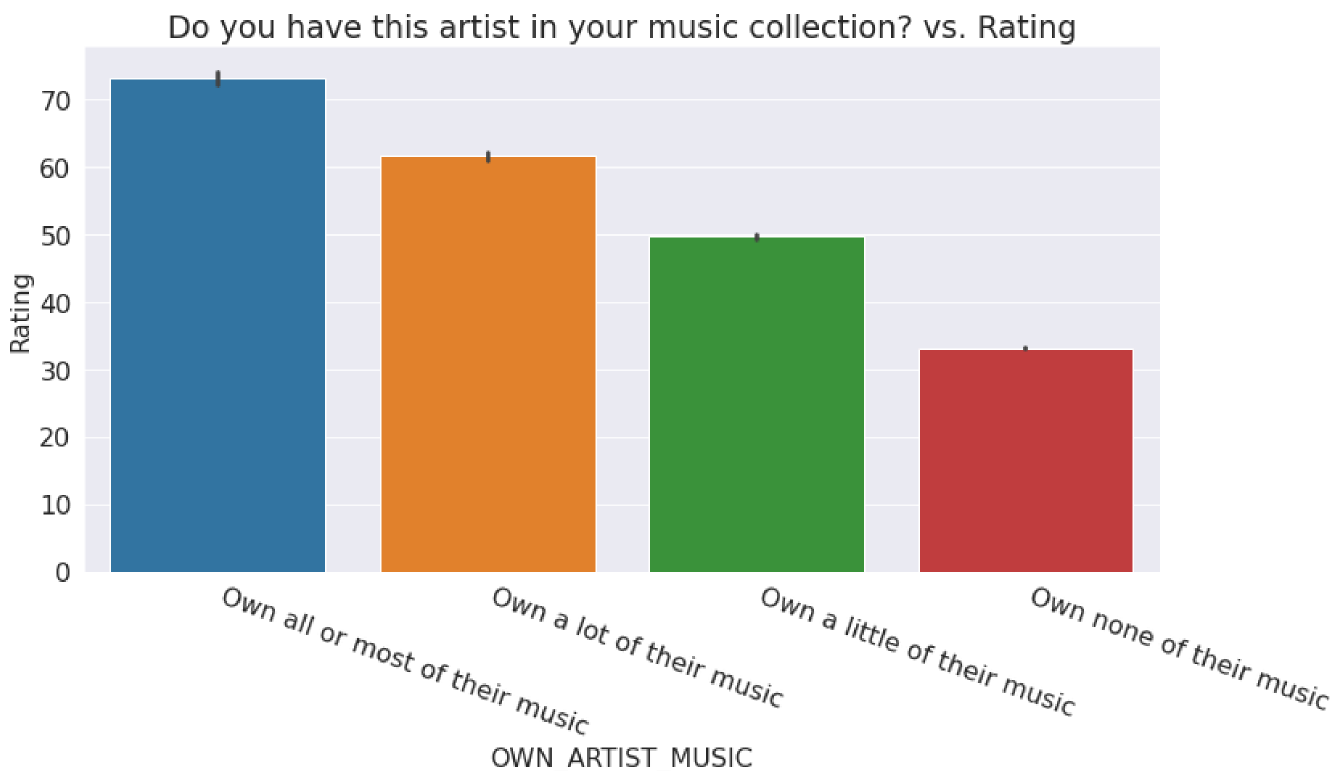
```
array(['Own none of their music', 'Own a little of their music',
       'Own all or most of their music', 'Own a lot of their music'],
      dtype=object)
```

```
training_merge_df['OWN_ARTIST_MUSIC'].value_counts()
```

```
Own none of their music           160113
Own a little of their music        18721
Own a lot of their music            7263
Own all or most of their music      2593
Name: OWN_ARTIST_MUSIC, dtype: int64
```

```
plot_order= training_merge_df.groupby('OWN_ARTIST_MUSIC')['Rating'].mean().sort_values(ascend
```

```
fig, ax = plt.subplots(figsize=(12,6))

plt.title('Do you have this artist in your music collection? vs. Rating')
sns.barplot(x='OWN_ARTIST_MUSIC', y='Rating', data=training_merge_df, order=plot_order)
plt.xticks(rotation=340, ha='left')
plt.show();
```



LIKE_ARTIST

```
training_merge_df['LIKE_ARTIST'].unique()
```

```
array([   nan,  28.  ,  18.  ,  33.  ,  36.  ,  53.  ,  50.  ,  63.  ,
        68.  ,  56.  ,  74.  ,  51.  ,  38.  ,  29.  ,  71.  ,  90.  ,
        70.  ,  30.  ,  52.  ,  84.  ,  59.  ,  66.  ,  42.  ,  48.  ,
        32.  ,  49.  ,  81.  , 100.  ,  45.  ,  87.  ,  57.  ,  83.  ,
        92.  ,  75.  ,  47.  ,  13.  ,  41.  ,  17.  ,  12.  ,   1.  ,
         4.  ,  55.  ,  65.  ,  16.  ,  58.  ,  99.  ,  69.  ,  15.  ,
        27.  ,  46.  ,  10.  ,  44.  ,  35.  ,   6.  ,  31.  ,  73.  ,
        26.  ,   2.  ,  43.  ,  54.  ,  61.  ,   9.  ,  14.  ,  62.  ,
        67.  ,  89.  ,  72.  ,  39.  ,   7.  ,   5.  ,  31.34,  20.  ,
        88.  ,  25.  ,  94.  ,  77.  ,  82.  ,  64.  ,  80.  ,  22.  ,
        23.  ,  86.  ,  40.  ,  37.  ,  34.  ,  21.  ,  93.  ,  11.  ,
        91.  ,  30.92,  98.  ,  79.  ,   8.  ,  33.05,   3.  ,  76.  ,
        85.  ,  78.  ,  60.  ,  24.  ,  97.  ,  19.  ,  95.  ,  29.21,
        28.14,  96.  ,  62.47,  48.83,  54.58,  23.24,  39.45,   0.  ,
        23.88,  32.84,  82.73,  78.25,  55.01,  78.68,  39.02,  65.88,
        13.01,   8.53,  38.59,  49.04,  22.6 ,  70.15,  18.34,  45.84,
        21.32,  55.44,  28.57,  37.74,  75.48,  38.17,  60.34,  32.41,
        27.51,  56.72,  80.81,  26.23,  51.81,  44.99,  57.57,  60.55,
        98.08,  16.63,  66.74,  20.9 ,  27.72,  46.91,  86.78,  62.69,
        72.92,  61.19,  29.85,  47.76,  69.72,  71.64,  84.01,  75.91,
        52.24,  29.64,  51.06,  43.28,  47.55,  25.37,   2.99,  50.32,
        80.38])
```

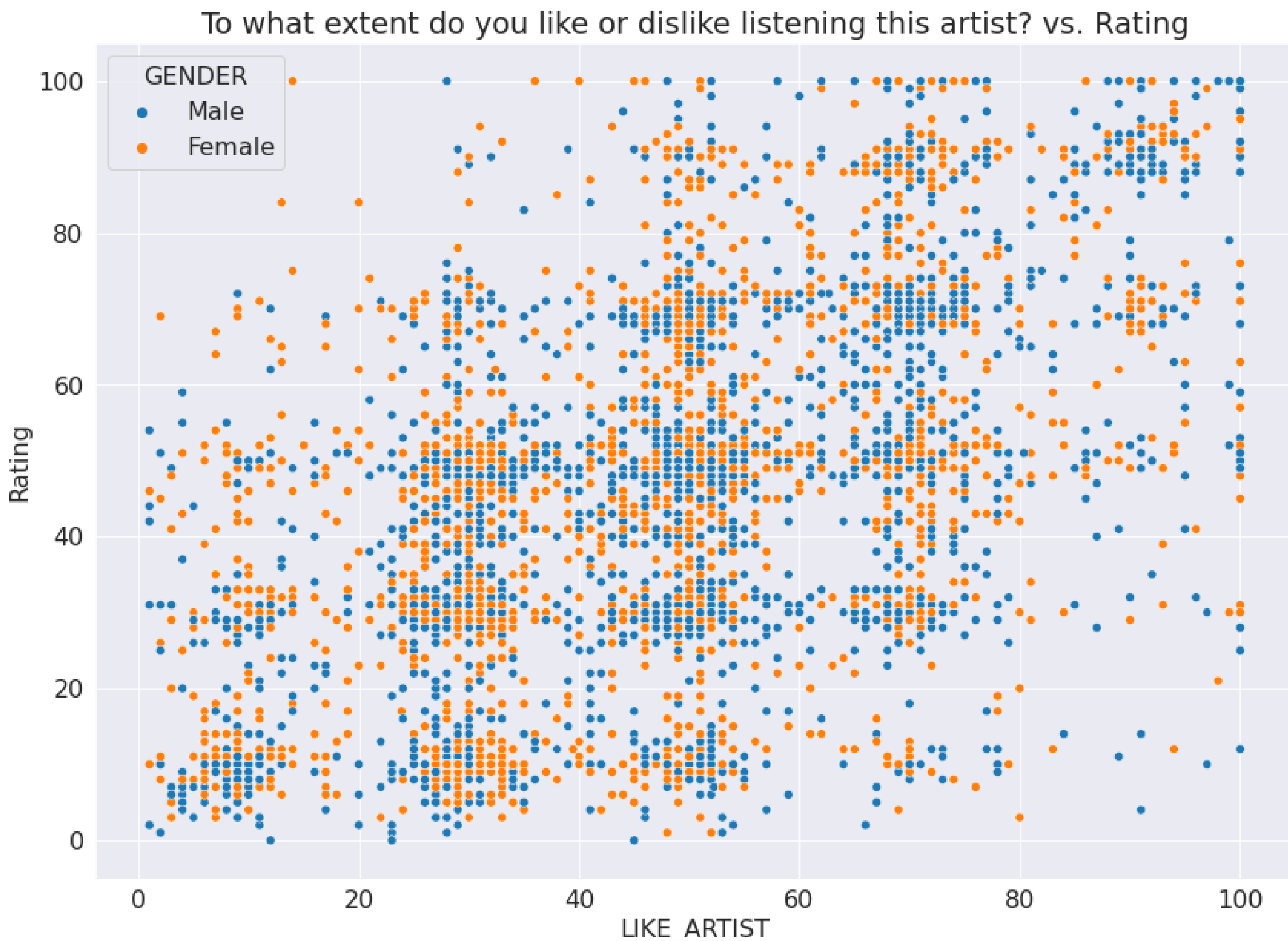
```
training_merge_df['LIKE_ARTIST'].value_counts()
```

```
49.00    2707
51.00    2463
30.00    2425
50.00    2218
29.00    2114
         ...
44.99       1
57.57       1
60.55       1
98.08       1
80.38       1
Name: LIKE_ARTIST, Length: 168, dtype: int64
```

```
training_merge_df
```

	Artist	Track	User	Rating	Time	HEARD_OF	OWN_ARTIST_MUSIC	LIKE_ARTIST
0	40	179	47994	9	17	Never heard of	Own none of their music	NaN
1	9	23	8575	58	7	Never heard of	Own none of their music	NaN
2	46	168	45475	13	16	Never heard of	Own none of their music	NaN
3	11	153	39508	42	15	Heard of and listened to music EVER	Own none of their music	28.0
4	14	32	11565	54	19	Heard of and listened to music EVER	Own none of their music	18.0
...	...	...	...	...	...	...	...	...
188685	0	3	1278	29	6	Never heard of	Own none of their music	NaN
188686	1	6	2839	30	18	Heard of	Own none of their music	NaN

```
plt.title('To what extent do you like or dislike listening this artist? vs. Rating')
sns.scatterplot(x='LIKE_ARTIST', y='Rating', hue='GENDER', data=training_merge_df.sample(1500
```




```
training_merge_df[training_merge_df['LIKE_ARTIST'].isna()].Rating.describe()
```

```
count    133662.000000
mean         32.326353
std          20.782582
min           0.000000
25%          12.000000
50%          30.000000
75%          48.000000
max         100.000000
Name: Rating, dtype: float64
```

```
training_merge_df.Rating.describe()
```

```
count    188690.000000
mean         36.435391
std          22.586036
min           0.000000
25%          15.000000
```

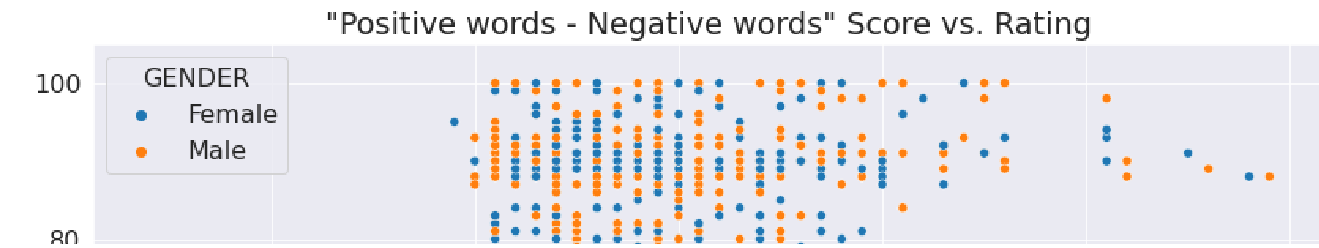
```
50%          32.000000
75%          50.000000
max         100.000000
Name: Rating, dtype: float64
```

```
training_merge_df[~training_merge_df['LIKE_ARTIST'].isna()].Rating.describe()
```

```
count    55028.000000
mean        46.416170
std         23.653523
min          0.000000
25%         30.000000
50%         48.000000
75%         64.250000
max        100.000000
Name: Rating, dtype: float64
```

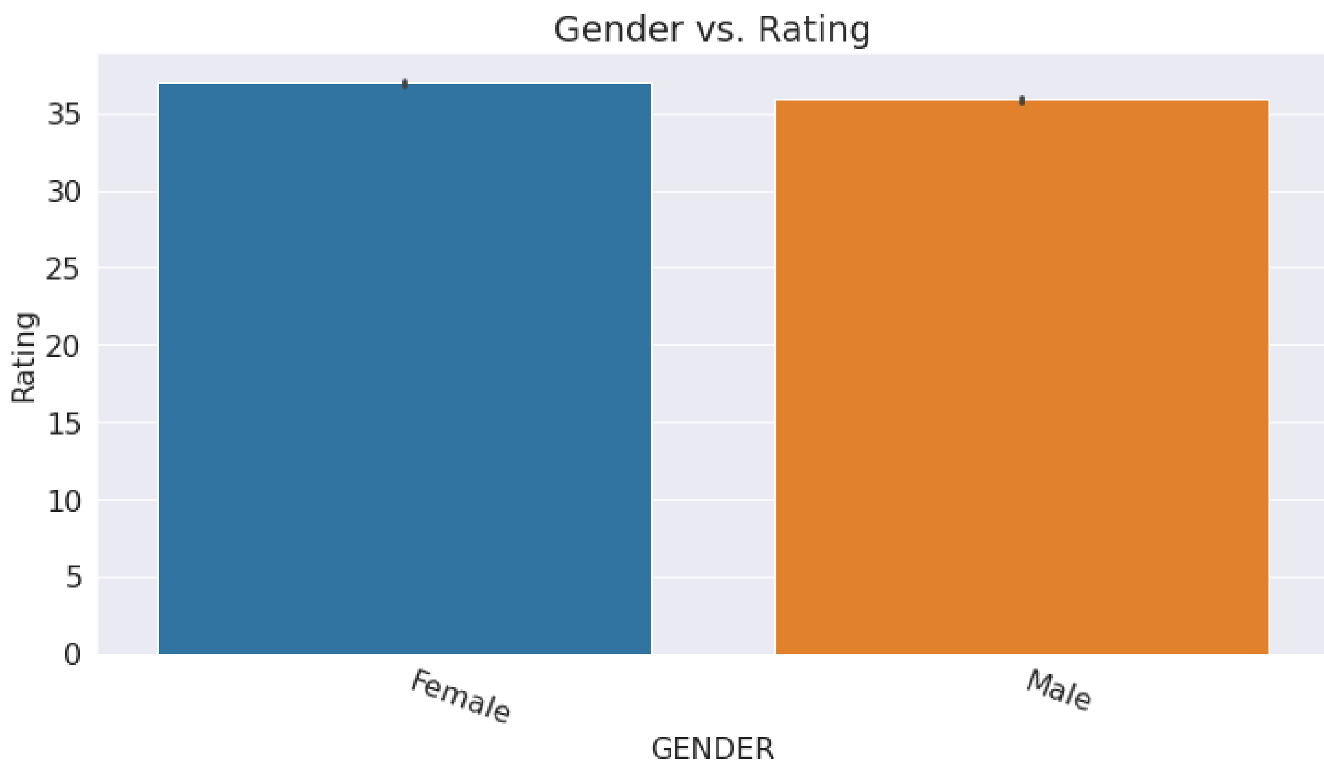
Words_Score

```
plt.title('"Positive words - Negative words" Score vs. Rating')
sns.scatterplot(x='words_score', y='Rating', hue='GENDER', data=training_merge_df.sample(1000
```



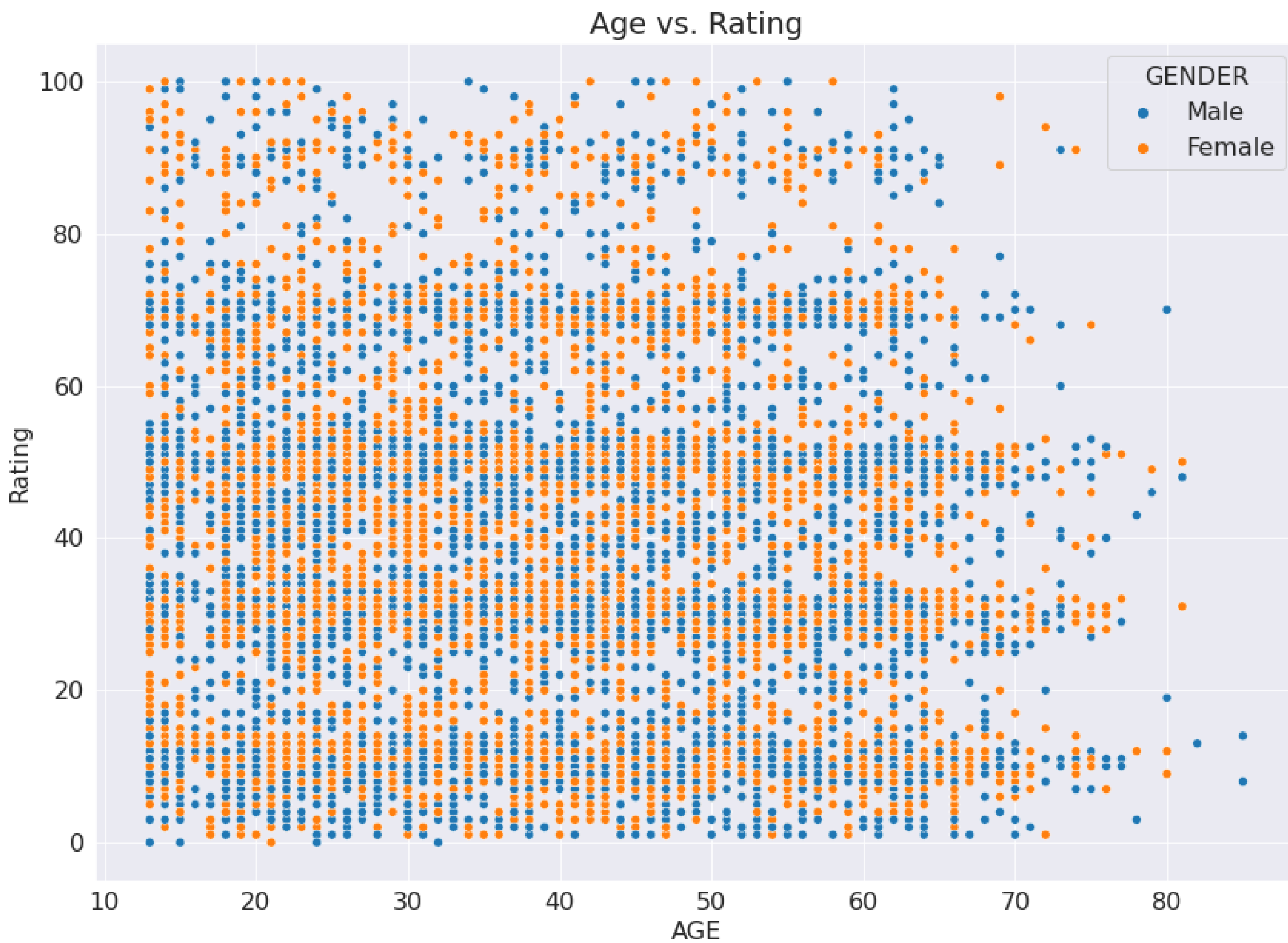
GENDER


```
fig, ax = plt.subplots(figsize=(12,6))

plt.title('Gender vs. Rating')
sns.barplot(x='GENDER', y='Rating', data=training_merge_df)
plt.xticks(rotation=340, ha='left')
plt.show();
```



AGE

```
plt.title('Age vs. Rating')
sns.scatterplot(x='AGE', y='Rating', hue='GENDER', data=training_merge_df.sample(10000));
```



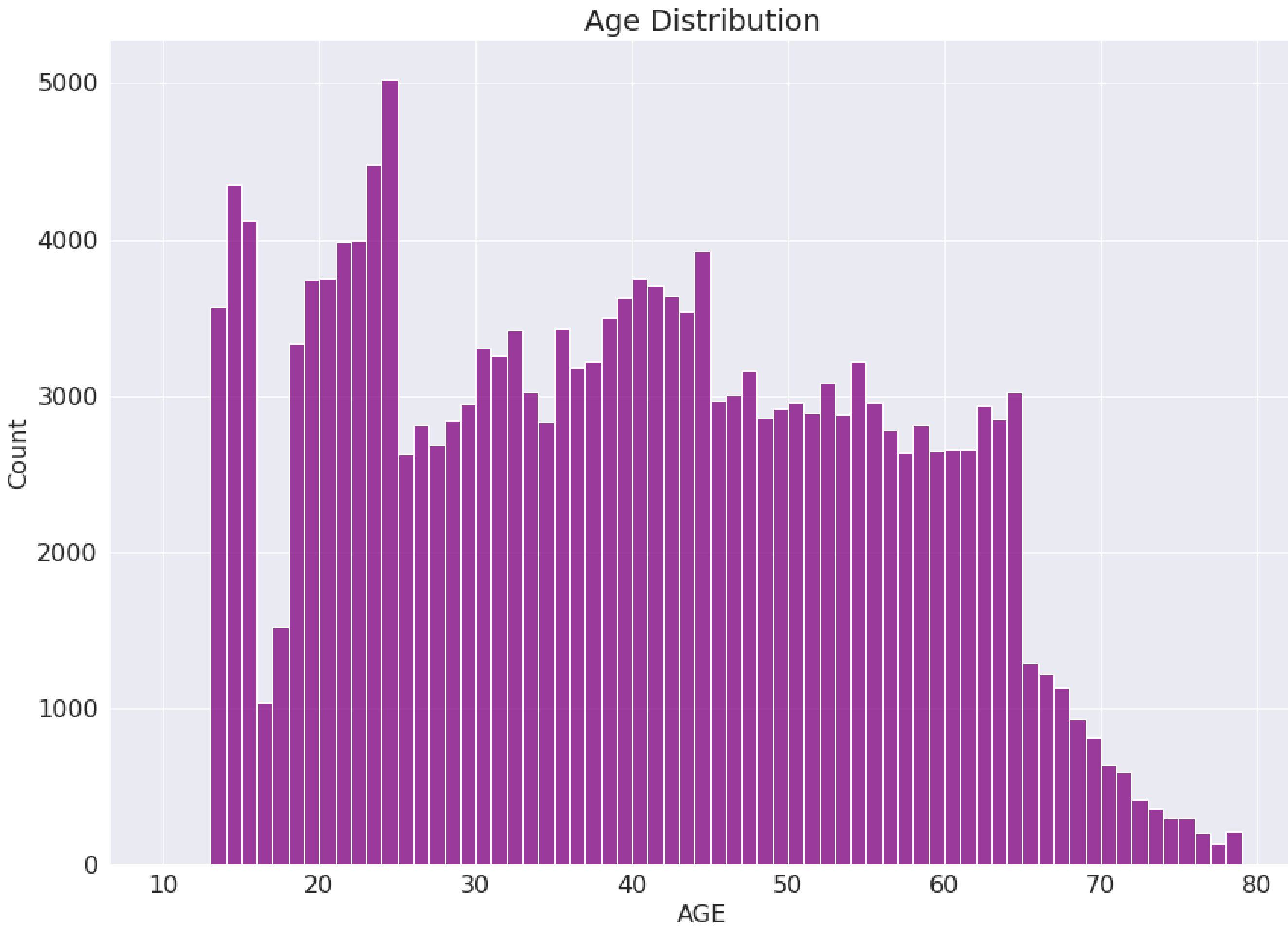

```
training_merge_df['AGE'].describe()
```

```
count    174982.000000
mean         39.246923
std          16.035515
min          13.000000
25%          25.000000
50%          39.000000
75%          52.000000
max          94.000000
Name: AGE, dtype: float64
```

```
print('Nan cells in the training_merge_df table {}'.format(training_merge_df['AGE'].isna().su
```

```
Nan cells in the training_merge_df table 13708
```

```
plt.title('Age Distribution')
sns.histplot(training_merge_df.AGE, bins=np.arange(10,80,1), color='purple');
```




```
training_merge_df[training_merge_df['AGE'] > 50].AGE.count()
```

```
48897
```

```
def age_to_categorical(x):
  try:
    if int(x) <= 17:
      return '13-17'
    elif 17< int(x) <= 25:
      return '18-25'
    elif 25< int(x) <= 35:
      return '26-35'
    elif 35< int(x) <= 50:
      return '36-50'
    elif 50< int(x) <= 65:
      return '51-65'
```

```
    else:
      return 'older than 65'
  except:
    return np.nan

training_merge_df['AGE_GROUP'] = training_merge_df['AGE'].apply(lambda x: age_to_categorical(
```

```
training_merge_df['AGE_GROUP'].value_counts()
```

```
36-50            49963
51-65            41315
18-25            30944
26-35            30573
13-17            14605
older than 65     7582
Name: AGE_GROUP, dtype: int64
```

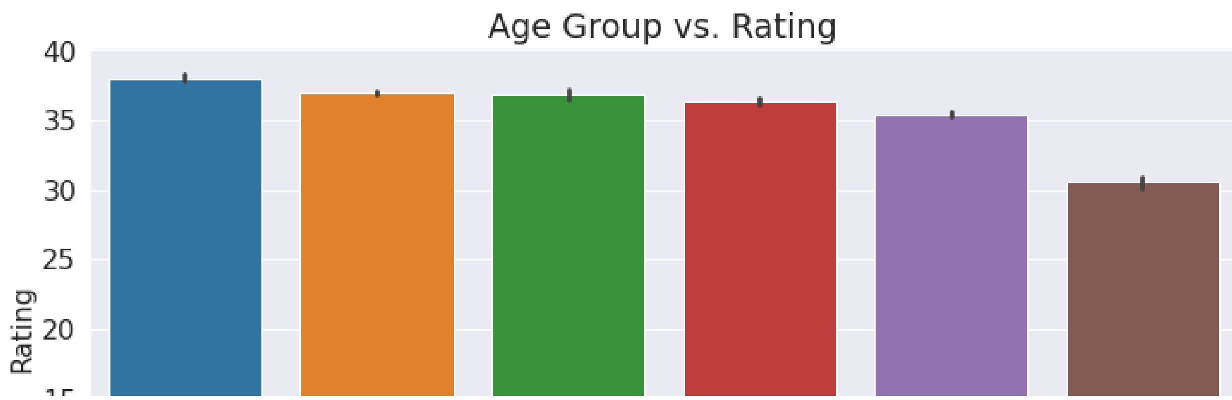
```
training_merge_df['AGE_GROUP'].fillna('36-50', inplace=True)
training_merge_df['AGE'].fillna(39, inplace=True)
```

Test DataFrame

```
test_merge_df['AGE_GROUP'] = test_merge_df['AGE'].apply(lambda x: age_to_categorical(x))
test_merge_df['AGE_GROUP'].fillna('36-50', inplace=True)
test_merge_df['AGE'].fillna(39, inplace=True)
```

```
plot_order= training_merge_df.groupby('AGE_GROUP')['Rating'].mean().sort_values(ascending=Fal
```

```
fig, ax = plt.subplots(figsize=(12,6))

plt.title('Age Group vs. Rating')
sns.barplot(x='AGE_GROUP', y='Rating', data=training_merge_df, order=plot_order)
plt.xticks(rotation=350, ha='left')
plt.show();
```

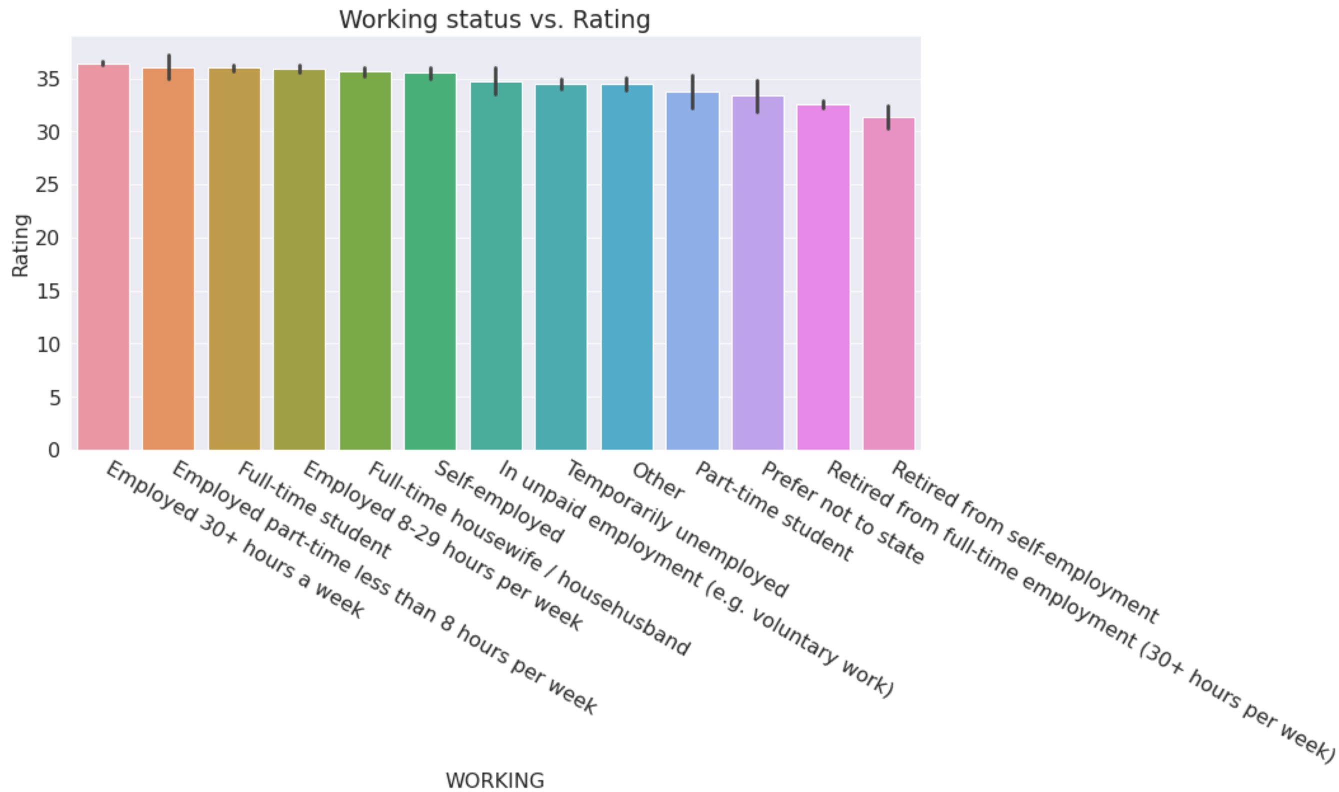


Working

```
training_merge_df['WORKING'].value_counts()
```

```
Employed 30+ hours a week                                 53347
Full-time student                                         20244
Employed 8-29 hours per week                              16284
Retired from full-time employment (30+ hours per week)    13234
Full-time housewife / househusband                        10367
Self-employed                                              7629
Temporarily unemployed                                     7528
Other                                                      5725
Retired from self-employment                               1480
Employed part-time less than 8 hours per week              1480
In unpaid employment (e.g. voluntary work)                 1407
Prefer not to state                                         947
Part-time student                                           873
Name: WORKING, dtype: int64
```

```
plot_order= training_merge_df.groupby('WORKING')['Rating'].mean().sort_values(ascending=False


fig, ax = plt.subplots(figsize=(12,6))

plt.title('Working status vs. Rating')
sns.barplot(x='WORKING', y='Rating', data=training_merge_df, order=plot_order)
plt.xticks(rotation=330, ha='left')
plt.show();
```



Region

```
training_merge_df['REGION'].unique()
```

```
array(['North', 'Centre', 'Midlands', 'South', nan, 'Northern Ireland',
       'North Ireland'], dtype=object)
```

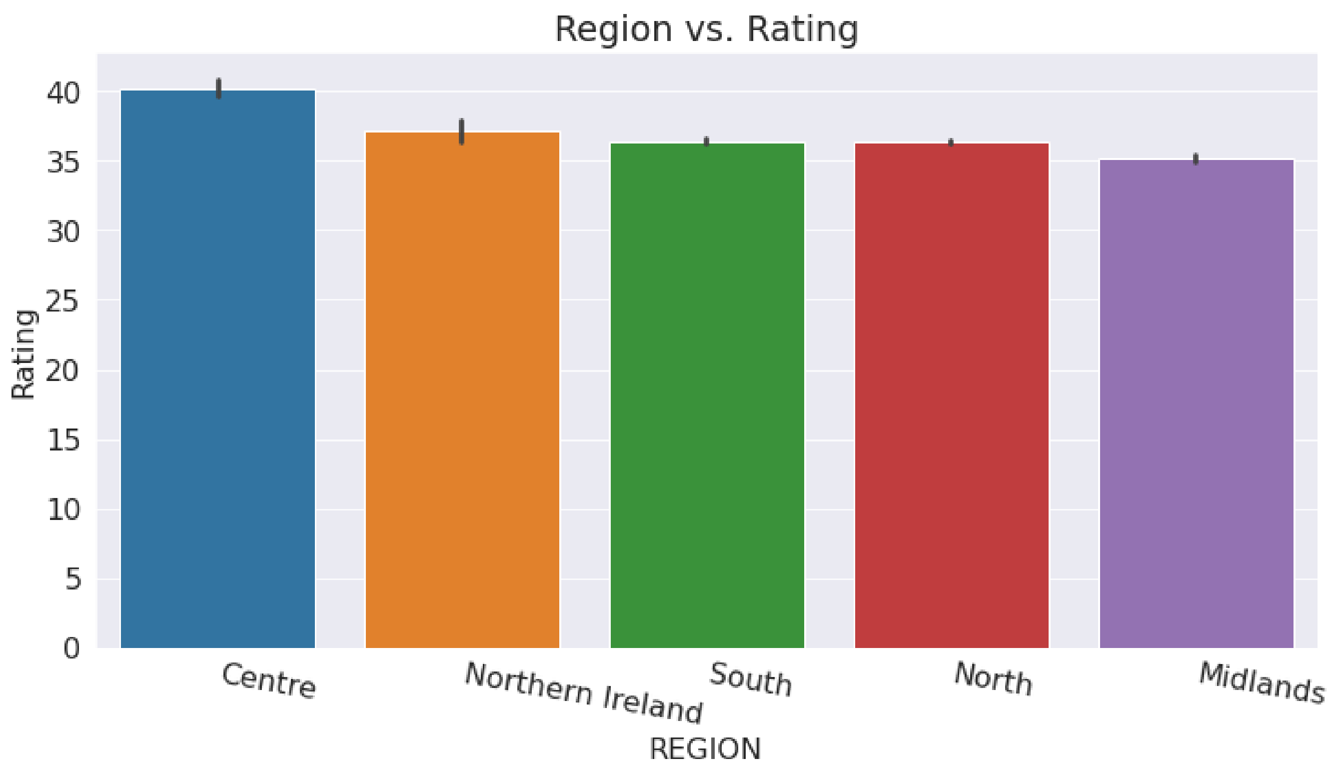
```
training_merge_df['REGION'].value_counts()
```

```
North               58707
South               54005
Midlands            44220
Centre               7284
Northern Ireland     2890
North Ireland         375
Name: REGION, dtype: int64
```

```
training_merge_df['REGION'].replace(['North Ireland'], 'Northern Ireland', inplace=True)
test_merge_df['REGION'].replace(['North Ireland'], 'Northern Ireland', inplace=True)
```

```
plot_order= training_merge_df.groupby('REGION')['Rating'].mean().sort_values(ascending=False)
```

```
fig, ax = plt.subplots(figsize=(12,6))

plt.title('Region vs. Rating')
sns.barplot(x='REGION', y='Rating', data=training_merge_df, order=plot_order)
plt.xticks(rotation=350, ha='left')
plt.show();
```



Music

```
training_merge_df['MUSIC'].unique()
```

```
array(['Music means a lot to me and is a passion of mine',
       'Music is important to me but not necessarily more important',
       'I like music but it does not feature heavily in my life',
       'Music is important to me but not necessarily more important than other hobbies
or interests',
       nan, 'Music has no particular interest for me',
       'Music is no longer as important as it used to be to me'],
      dtype=object)
```

```
training_merge_df['MUSIC'].value_counts()
```

```
Music is important to me but not necessarily more important
56695
Music means a lot to me and is a passion of mine
```

```
54793
I like music but it does not feature heavily in my life
43023
Music is important to me but not necessarily more important than other hobbies or
interests    12977
Music is no longer as important as it used to be to me
5702
Music has no particular interest for me
3643
Name: MUSIC, dtype: int64
```

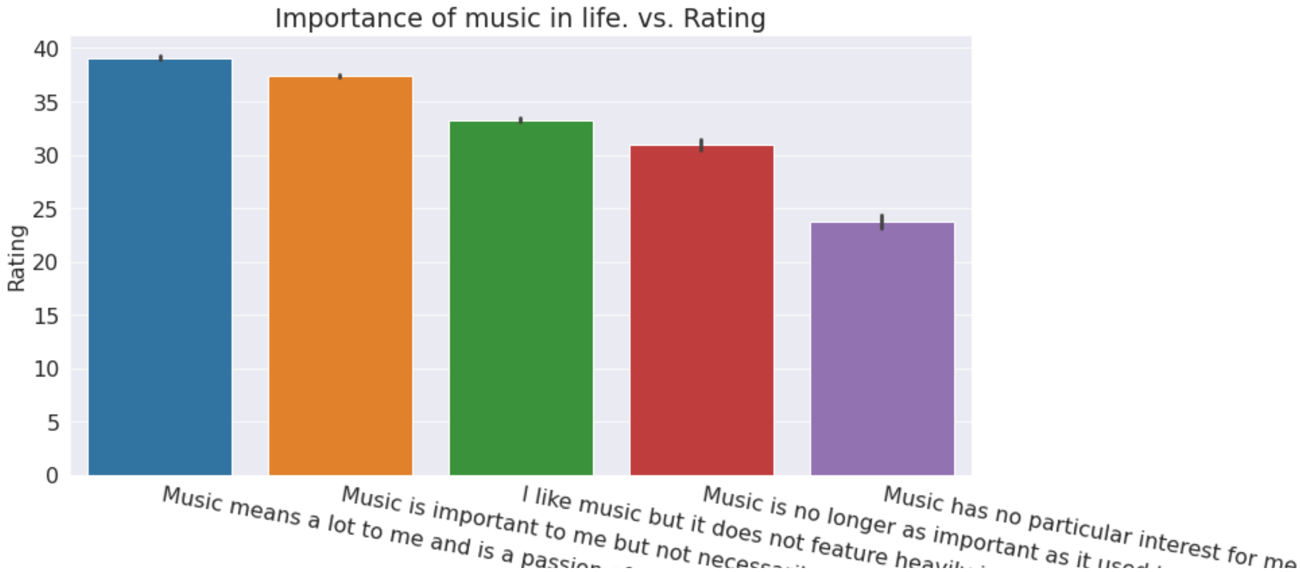
```
training_merge_df['MUSIC'].replace(['Music is important to me but not necessarily more import
test_merge_df['MUSIC'].replace(['Music is important to me but not necessarily more important'
```

```
training_merge_df['MUSIC'].value_counts()
```

```
Music is important to me but not necessarily more important than other hobbies or
interests    69672
Music means a lot to me and is a passion of mine
54793
I like music but it does not feature heavily in my life
43023
Music is no longer as important as it used to be to me
5702
Music has no particular interest for me
3643
Name: MUSIC, dtype: int64
```

```
plot_order= training_merge_df.groupby('MUSIC')['Rating'].mean().sort_values(ascending=False).
```

```
fig, ax = plt.subplots(figsize=(12,6))

plt.title('Importance of music in life. vs. Rating')
sns.barplot(x='MUSIC', y='Rating', data=training_merge_df, order=plot_order)
plt.xticks(rotation=350, ha='left')
plt.show();
```



List own

```
training_merge_df['LIST_OWN'].unique()
```

```
array(['3 hours', '1', '5 hours', '1 hour', 'Less than an hour',
       '0 Hours', nan, '2', '2 hours', '4 hours', '10 hours', '16+ hours',
       '0', '6 hours', '8 hours', '4', '3', '14 hours', '15 hours',
       '7 hours', '13 hours', '12 hours', '5', '6', '8', '10', '12',
       '9 hours', '7', '11 hours', '16 hours', '15', 'More than 16 hours',
       '20', '16', '9', '17', '14', '11', '18', '22', '24', '13'],
      dtype=object)
```

```
training_merge_df['LIST_OWN'].value_counts()
```

```
1 hour               29683
2 hours              27505
Less than an hour    26697
3 hours              13078
0 Hours              12367
1                     8801
4 hours               8116
2                     6937
5 hours               4430
3                     2959
0                     2792
6 hours               2744
16+ hours             1978
8 hours               1874
10 hours              1807
4                     1465
7 hours               1164
5                      940
12 hours               774
9 hours                471
6                      361
```

```
11 hours              235
8                     234
15 hours              231
10                    217
14 hours              193
16 hours              130
7                     121
13 hours              106
12                     94
9                      40
15                     22
14                     20
16                     17
20                     13
More than 16 hours     13
17                      7
11                      6
22                      3
13                      3
24                      2
18                      1
Name: LIST_OWN, dtype: int64
```

```
training_merge_df['LIST_OWN'].isna().sum()
```

```
30039
```

```
training_merge_df['LIST_OWN'].replace(['0 Hours'], '0', inplace=True)
training_merge_df['LIST_OWN'].replace(['Less than an hour'], '0.5', inplace=True)
training_merge_df['LIST_OWN'].replace(['1 hour'], '1', inplace=True)
training_merge_df['LIST_OWN'].replace(['2 hours'], '2', inplace=True)
training_merge_df['LIST_OWN'].replace(['3 hours'], '3', inplace=True)
training_merge_df['LIST_OWN'].replace(['4 hours'], '4', inplace=True)
training_merge_df['LIST_OWN'].replace(['5 hours'], '5', inplace=True)
training_merge_df['LIST_OWN'].replace(['6 hours'], '6', inplace=True)
training_merge_df['LIST_OWN'].replace(['7 hours'], '7', inplace=True)
training_merge_df['LIST_OWN'].replace(['8 hours'], '8', inplace=True)
training_merge_df['LIST_OWN'].replace(['9 hours'], '9', inplace=True)
training_merge_df['LIST_OWN'].replace(['10 hours'], '10', inplace=True)
training_merge_df['LIST_OWN'].replace(['11 hours'], '11', inplace=True)
training_merge_df['LIST_OWN'].replace(['12 hours'], '12', inplace=True)
training_merge_df['LIST_OWN'].replace(['13 hours'], '13', inplace=True)
training_merge_df['LIST_OWN'].replace(['14 hours'], '14', inplace=True)
training_merge_df['LIST_OWN'].replace(['15 hours'], '15', inplace=True)
training_merge_df['LIST_OWN'].replace(['16 hours'], '16', inplace=True)
training_merge_df['LIST_OWN'].replace(['16+ hours'], '16', inplace=True)
training_merge_df['LIST_OWN'].replace(['More than 16 hours'], '16', inplace=True)
```

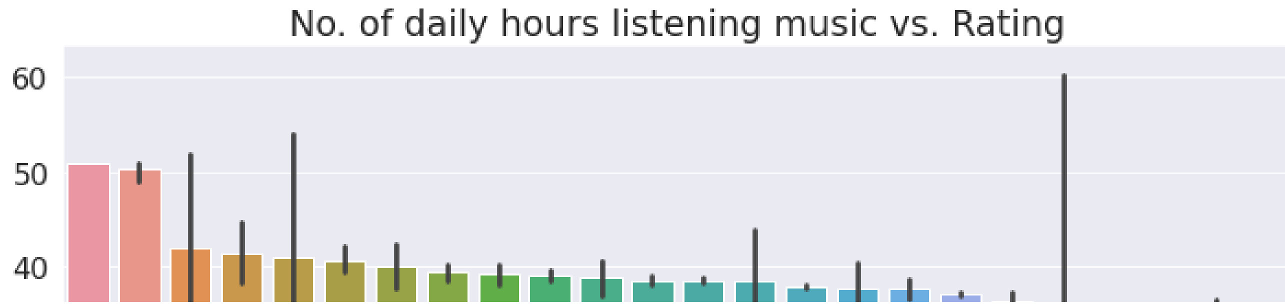
```
training_merge_df['LIST_OWN'].fillna('No Answer', inplace=True)
```

Test DataFrame

```
test_merge_df['LIST_OWN'].replace(['0 Hours'], '0', inplace=True)
test_merge_df['LIST_OWN'].replace(['Less than an hour'], '0.5', inplace=True)
test_merge_df['LIST_OWN'].replace(['1 hour'], '1', inplace=True)
test_merge_df['LIST_OWN'].replace(['2 hours'], '2', inplace=True)
test_merge_df['LIST_OWN'].replace(['3 hours'], '3', inplace=True)
test_merge_df['LIST_OWN'].replace(['4 hours'], '4', inplace=True)
test_merge_df['LIST_OWN'].replace(['5 hours'], '5', inplace=True)
test_merge_df['LIST_OWN'].replace(['6 hours'], '6', inplace=True)
test_merge_df['LIST_OWN'].replace(['7 hours'], '7', inplace=True)
test_merge_df['LIST_OWN'].replace(['8 hours'], '8', inplace=True)
test_merge_df['LIST_OWN'].replace(['9 hours'], '9', inplace=True)
test_merge_df['LIST_OWN'].replace(['10 hours'], '10', inplace=True)
test_merge_df['LIST_OWN'].replace(['11 hours'], '11', inplace=True)
test_merge_df['LIST_OWN'].replace(['12 hours'], '12', inplace=True)
test_merge_df['LIST_OWN'].replace(['13 hours'], '13', inplace=True)
test_merge_df['LIST_OWN'].replace(['14 hours'], '14', inplace=True)
test_merge_df['LIST_OWN'].replace(['15 hours'], '15', inplace=True)
test_merge_df['LIST_OWN'].replace(['16 hours'], '16', inplace=True)
test_merge_df['LIST_OWN'].replace(['16+ hours'], '16', inplace=True)
test_merge_df['LIST_OWN'].replace(['More than 16 hours'], '16', inplace=True)
test_merge_df['LIST_OWN'].fillna('No Answer', inplace=True)
```

```
plot_order= training_merge_df.groupby('LIST_OWN')['Rating'].mean().sort_values(ascending=Fals
```

```
fig, ax = plt.subplots(figsize=(12,6))

plt.title('No. of daily hours listening music vs. Rating')
sns.barplot(x='LIST_OWN', y='Rating', data=training_merge_df, order=plot_order)
plt.xticks(rotation=335, ha='left')
plt.show();
```




```
lo_mapper = {'No Answer': 'No Answer',
          '0': '0',
          '0.5': '0.5',
          '1': '1',
          '2': '2',
          '3': '3-6',
          '4': '3-6',
          '5': '3-6',
          '6': '3-6',
          '7': '7-10',
          '8': '7-10',
          '9': '7-10',
          '10': '7-10',
          '11': '11-14',
          '12':  '11-14',
          '13': '11-14',
          '14': '11-14',
          '15': '15-19',
          '16': '15-19',
          '17': '15-19',
          '18': '15-19',
          '19': '15-19',
          '20': '20 and plus',
          '21': '20 and plus',
          '22': '20 and plus',
          '23': '20 and plus',
          '24': '20 and plus'
          }
```

```
training_merge_df['LIST_OWN'] = training_merge_df['LIST_OWN'].map(lo_mapper)
```

```
training_merge_df['LIST_OWN'].unique()
```

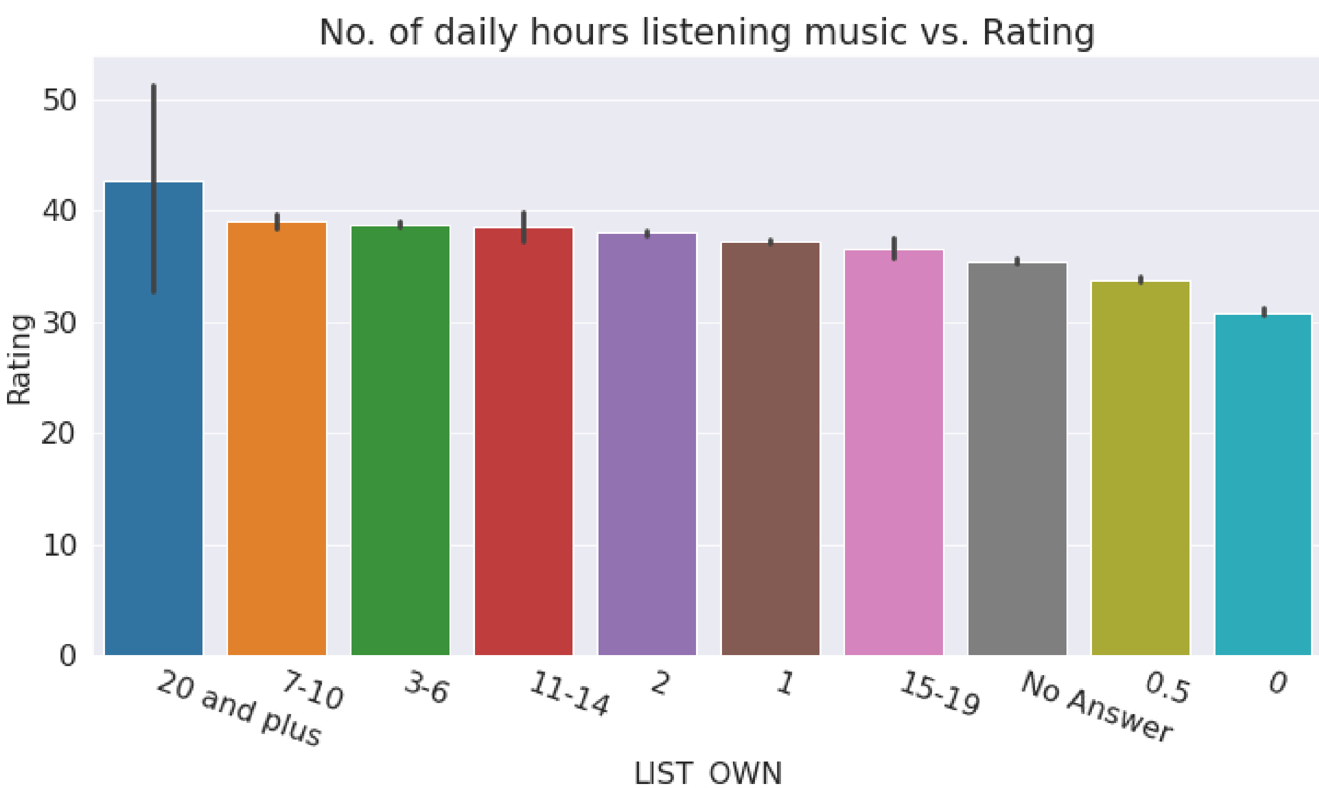
```
array(['3-6', '1', '0.5', '0', 'No Answer', '2', '7-10', '15-19', '11-14',
       '20 and plus'], dtype=object)
```

```
training_merge_df['LIST_OWN'].value_counts()
```

```
1              38484
2              34442
3-6            34093
```

```
No Answer      30039
0.5            26697
0              15159
7-10            5928
15-19           2399
11-14           1431
20 and plus       18
Name: LIST_OWN, dtype: int64
```

```
plot_order= training_merge_df.groupby('LIST_OWN')['Rating'].mean().sort_values(ascending=Fals


fig, ax = plt.subplots(figsize=(12,6))

plt.title('No. of daily hours listening music vs. Rating')
sns.barplot(x='LIST_OWN', y='Rating', data=training_merge_df, order=plot_order)
plt.xticks(rotation=340, ha='left')
plt.show();
```




```
test_merge_df['LIST_OWN'] = test_merge_df['LIST_OWN'].map(lo_mapper)
```

List Back

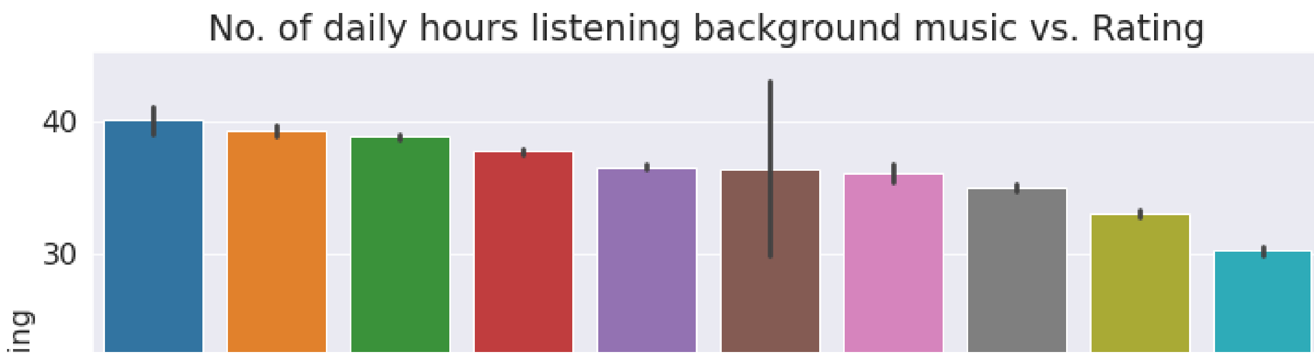
```
training_merge_df['LIST_BACK'].unique()
```

```
array(['0 Hours', '2', nan, '3 hours', 'Less than an hour', '4 hours',
       '8 hours', '5 hours', '4', '3', '1 hour', '2 hours', '5', '1',
       '6 hours', '7 hours', 'More than 16 hours', '0', '9 hours', '6',
       '14 hours', '16+ hours', '8', '10 hours', '9', '16 hours',
       '15 hours', '12', '12 hours', '10', '20', '18', '11 hours',
       '13 hours', '7', '14', '15', '19', '24', '16', '11', '21'],
      dtype=object)
```

```
training_merge_df['LIST_BACK'].value_counts()
```

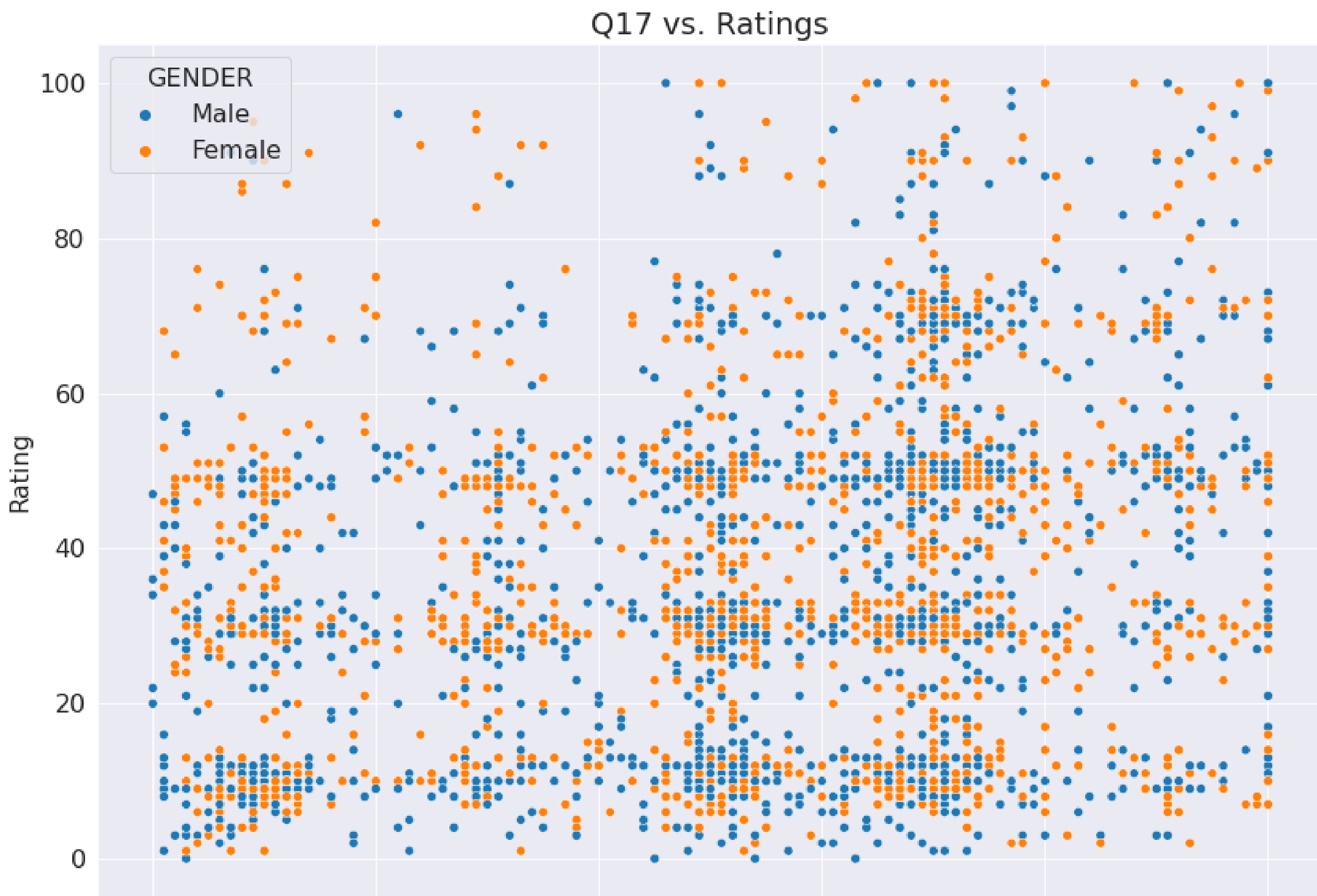
```
2 hours               24663
1 hour                23409
Less than an hour     22232
3 hours               13679
0 Hours               10565
4 hours               10492
1                      6856
5 hours                6170
2                      6027
6 hours                5099
8 hours                4473
3                      3097
16+ hours              2890
0                      2768
7 hours                2572
10 hours               2544
4                      2284
5                      1587
9 hours                1200
12 hours               1171
6                      1119
8                      1013
7                       498
14 hours                369
11 hours                334
15 hours                325
10                      319
16 hours                278
12                      213
13 hours                213
9                       189
20                       36
More than 16 hours       23
15                       17
14                       14
19                       11
24                       11
16                       11
11                       10
18                        8
21                        1
Name: LIST_BACK, dtype: int64
```

```
training_merge_df['LIST_BACK'].replace(['0 Hours'], '0', inplace=True)
training_merge_df['LIST_BACK'].replace(['Less than an hour'], '0.5', inplace=True)
training_merge_df['LIST_BACK'].replace(['1 hour'], '1', inplace=True)
training_merge_df['LIST_BACK'].replace(['2 hours'], '2', inplace=True)
training_merge_df['LIST_BACK'].replace(['3 hours'], '3', inplace=True)
training_merge_df['LIST_BACK'].replace(['4 hours'], '4', inplace=True)
training_merge_df['LIST_BACK'].replace(['5 hours'], '5', inplace=True)
training_merge_df['LIST_BACK'].replace(['6 hours'], '6', inplace=True)
training_merge_df['LIST_BACK'].replace(['7 hours'], '7', inplace=True)
training_merge_df['LIST_BACK'].replace(['8 hours'], '8', inplace=True)
training_merge_df['LIST_BACK'].replace(['9 hours'], '9', inplace=True)
training_merge_df['LIST_BACK'].replace(['10 hours'], '10', inplace=True)
training_merge_df['LIST_BACK'].replace(['11 hours'], '11', inplace=True)
training_merge_df['LIST_BACK'].replace(['12 hours'], '12', inplace=True)
training_merge_df['LIST_BACK'].replace(['13 hours'], '13', inplace=True)
training_merge_df['LIST_BACK'].replace(['14 hours'], '14', inplace=True)
training_merge_df['LIST_BACK'].replace(['15 hours'], '15', inplace=True)
training_merge_df['LIST_BACK'].replace(['16 hours'], '16', inplace=True)
training_merge_df['LIST_BACK'].replace(['16+ hours'], '16', inplace=True)
training_merge_df['LIST_BACK'].replace(['More than 16 hours'], '16', inplace=True)


training_merge_df['LIST_BACK'].fillna('No Answer', inplace=True)


training_merge_df['LIST_BACK'] = training_merge_df['LIST_BACK'].map(lo_mapper)


plot_order= training_merge_df.groupby('LIST_BACK')['Rating'].mean().sort_values(ascending=Fal


fig, ax = plt.subplots(figsize=(12,6))

plt.title('No. of daily hours listening background music vs. Rating')
sns.barplot(x='LIST_BACK', y='Rating', data=training_merge_df, order=plot_order)
plt.xticks(rotation=340, ha='left')
plt.show();
```



Test DataFrame

```
test_merge_df['LIST_BACK'].replace(['0 Hours'], '0', inplace=True)
test_merge_df['LIST_BACK'].replace(['Less than an hour'], '0.5', inplace=True)
test_merge_df['LIST_BACK'].replace(['1 hour'], '1', inplace=True)
test_merge_df['LIST_BACK'].replace(['2 hours'], '2', inplace=True)
test_merge_df['LIST_BACK'].replace(['3 hours'], '3', inplace=True)
test_merge_df['LIST_BACK'].replace(['4 hours'], '4', inplace=True)
test_merge_df['LIST_BACK'].replace(['5 hours'], '5', inplace=True)
test_merge_df['LIST_BACK'].replace(['6 hours'], '6', inplace=True)
test_merge_df['LIST_BACK'].replace(['7 hours'], '7', inplace=True)
test_merge_df['LIST_BACK'].replace(['8 hours'], '8', inplace=True)
test_merge_df['LIST_BACK'].replace(['9 hours'], '9', inplace=True)
test_merge_df['LIST_BACK'].replace(['10 hours'], '10', inplace=True)
test_merge_df['LIST_BACK'].replace(['11 hours'], '11', inplace=True)
test_merge_df['LIST_BACK'].replace(['12 hours'], '12', inplace=True)
test_merge_df['LIST_BACK'].replace(['13 hours'], '13', inplace=True)
test_merge_df['LIST_BACK'].replace(['14 hours'], '14', inplace=True)
test_merge_df['LIST_BACK'].replace(['15 hours'], '15', inplace=True)
test_merge_df['LIST_BACK'].replace(['16 hours'], '16', inplace=True)
test_merge_df['LIST_BACK'].replace(['16+ hours'], '16', inplace=True)
test_merge_df['LIST_BACK'].replace(['More than 16 hours'], '16', inplace=True)
test_merge_df['LIST_BACK'].fillna('No Answer', inplace=True)

test_merge_df['LIST_BACK'] = test_merge_df['LIST_BACK'].map(lo_mapper)


plt.title('Q17 vs. Ratings')
sns.scatterplot(x='Q17', y='Rating', hue='GENDER', data=training_merge_df.sample(3000));
```




```
training_merge_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 188690 entries, 0 to 188689
Data columns (total 36 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   Artist             188690 non-null  int64
 1   Track              188690 non-null  int64
 2   User               188690 non-null  int64
 3   Rating             188690 non-null  int64
 4   Time               188690 non-null  int64
 5   HEARD_OF           188690 non-null  object
 6   OWN_ARTIST_MUSIC   188690 non-null  object
 7   LIKE_ARTIST        55028 non-null   float64
 8   words_score        186636 non-null  float64
 9   GENDER             176833 non-null  object
 10  AGE                188690 non-null  float64
 11  WORKING            140545 non-null  object
 12  REGION             167481 non-null  object
 13  MUSIC              176833 non-null  object
 14  LIST_OWN           188690 non-null  object
 15  LIST_BACK          188690 non-null  object
 16  Q1                 176833 non-null  float64
 17  Q2                 176833 non-null  float64
 18  Q3                 176833 non-null  float64
 19  Q4                 176833 non-null  float64
 20  Q5                 176833 non-null  float64
 21  Q6                 176833 non-null  float64
```

```
 22  Q7                176833 non-null  float64
 23  Q8                176833 non-null  float64
 24  Q9                176833 non-null  float64
 25  Q10               176833 non-null  float64
 26  Q11               176833 non-null  float64
 27  Q12               176833 non-null  float64
 28  Q13               176833 non-null  float64
 29  Q14               176833 non-null  float64
 30  Q15               176833 non-null  float64
 31  Q16               142754 non-null  float64
 32  Q17               176833 non-null  float64
 33  Q18               140545 non-null  float64
 34  Q19               140545 non-null  float64
 35  AGE_GROUP         188690 non-null  object
dtypes: float64(22), int64(5), object(9)
memory usage: 57.3+ MB
```

HEARD_OF

```
mapper = {'Heard of and listened to music RECENTLY': 4,
          'Heard of and listened to music EVER': 3,
          'Heard of': 2,
          'Never heard of': 1}
```

```
training_merge_df['HEARD_OF'] = training_merge_df['HEARD_OF'].map(mapper)
```

Test DataFrame

```
test_merge_df['HEARD_OF'] = test_merge_df['HEARD_OF'].map(mapper)
```

```
training_merge_df['HEARD_OF'].unique()
```

```
array([1, 3, 2, 4])
```

```
training_merge_df['HEARD_OF'].value_counts()
```

```
1    98169
2    35493
3    34990
4    20038
Name: HEARD_OF, dtype: int64
```

Own Art Music

```
oam_mapper = {'Own all or most of their music': 4,
          'Own a lot of their music': 3,
          'Own a little of their music': 2,
          'Own none of their music': 1}
```

```
training_merge_df['OWN_ARTIST_MUSIC'] = training_merge_df['OWN_ARTIST_MUSIC'].map(oam_mapper)
```

Test DataFrame

```
test_merge_df['OWN_ARTIST_MUSIC'] = test_merge_df['OWN_ARTIST_MUSIC'].map(oam_mapper)
```

```
training_merge_df['OWN_ARTIST_MUSIC'].unique()
```

```
array([1, 2, 4, 3])
```

```
training_merge_df['OWN_ARTIST_MUSIC'].value_counts()
```

```
1    160113
2     18721
3      7263
4      2593
Name: OWN_ARTIST_MUSIC, dtype: int64
```

Like Artist

```
training_merge_df['LIKE_ARTIST'].isna().sum()
```

```
133662
```

```
def to_categorical(x):
  try:
    if 1<= int(x) <= 10:
      return '1-10'
    elif 11<= int(x) <= 20:
      return '11-20'
    elif 21<= int(x) <= 30:
      return '21-30'
    elif 31<= int(x) <= 40:
      return '31-40'
    elif 41<= int(x) <= 50:
      return '41-50'
    elif 51<= int(x) <= 60:
      return '51-60'
    elif 61<= int(x) <= 70:
```

```
      return '61-70'
    elif 71<= int(x) <= 80:
      return '71-80'
    elif 81<= int(x) <= 90:
      return '81-90'
    else:
      return '91-100'
  except:
    return np.nan

training_merge_df['LIKE_ARTIST'] = training_merge_df['LIKE_ARTIST'].apply(lambda x: to_catego

test_merge_df['LIKE_ARTIST'] = test_merge_df['LIKE_ARTIST'].apply(lambda x: to_categorical(x)


training_merge_df['LIKE_ARTIST'].fillna('No Answer', inplace=True)

test_merge_df['LIKE_ARTIST'].fillna('No Answer', inplace=True)
```

```
training_merge_df['LIKE_ARTIST'].value_counts()
```

```
No Answer    133662
41-50         11114
21-30          8804
51-60          8244
31-40          6574
61-70          6415
71-80          4976
1-10           2825
91-100         2269
11-20          2126
81-90          1681
Name: LIKE_ARTIST, dtype: int64
```

Music

```
training_merge_df['MUSIC'].unique()
```

```
array(['Music means a lot to me and is a passion of mine',
       'Music is important to me but not necessarily more important than other hobbies
or interests',
       'I like music but it does not feature heavily in my life', nan,
       'Music has no particular interest for me',
       'Music is no longer as important as it used to be to me'],
      dtype=object)
```

```
training_merge_df['MUSIC'].value_counts()
```

```
Music is important to me but not necessarily more important than other hobbies or
interests    69672
Music means a lot to me and is a passion of mine
54793
I like music but it does not feature heavily in my life
43023
Music is no longer as important as it used to be to me
5702
Music has no particular interest for me
3643
Name: MUSIC, dtype: int64
```

```
m_mapper = {'Music means a lot to me and is a passion of mine': 6,
          'Music is important to me but not necessarily more important than other hobbies or
          'No Answer': 4,
          'I like music but it does not feature heavily in my life': 3,
          'Music is no longer as important as it used to be to me': 2,
          'Music has no particular interest for me': 1,
          }
```

```
training_merge_df['MUSIC'] = training_merge_df['MUSIC'].map(m_mapper)
```

Test DataFrame

```
test_merge_df['MUSIC'] = test_merge_df['MUSIC'].map(m_mapper)
```

Missing Values in DF

```
training_merge_df['GENDER'].fillna('No Answer', inplace=True)
training_merge_df['WORKING'].fillna('No Answer', inplace=True)
training_merge_df['REGION'].fillna('No Answer', inplace=True)

test_merge_df['GENDER'].fillna('No Answer', inplace=True)
test_merge_df['WORKING'].fillna('No Answer', inplace=True)
test_merge_df['REGION'].fillna('No Answer', inplace=True)
```

Training & Validation Sets

As test set is already given.

We put 20% of Training test into calidation set.

```
from sklearn.model_selection import train_test_split


training_df, validation_df = train_test_split(training_merge_df, test_size=0.2)


print('training_df.shape :', training_df.shape)
print('validation_df.shape :', validation_df.shape)
```

```
training_df.shape : (150952, 36)
validation_df.shape : (37738, 36)
```

```
training_df
```

	Artist	Track	User	Rating	Time	HEARD_OF	OWN_ARTIST_MUSIC	LIKE_ARTIST	wor
168265	35	88	30594	69	23	1	1	No Answer	
186415	15	41	16939	30	9	2	1	No Answer	
186064	48	172	47900	28	17	1	1	No Answer	
38552	26	63	23658	79	22	1	1	No Answer	
149111	23	57	21142	28	21	1	1	No Answer	
...	...	...	...	...	...	...	...	...	
31991	45	163	45329	31	16	3	1	21-30	
25672	16	134	34029	13	12	1	1	No Answer	
81988	6	14	5979	68	7	1	1	No Answer	
97594	15	33	13060	14	19	1	1	No Answer	
53332	28	72	23225	31	22	1	1	No Answer	

150952 rows × 36 columns

```
validation_df
```

	Artist	Track	User	Rating	Time	HEARD_OF	OWN_ARTIST_MUSIC	LIKE_ARTIST	wor
109566	4	11	5357	74	18	2	1	No Answer	
49742	20	44	17177	53	21	1	1	No Answer	
92733	28	73	23226	49	22	3	2	41-50	
38465	22	122	32594	54	0	4	2	91-100	
20298	31	79	26606	12	11	1	1	No Answer	
...	...	...	...	...	...	...	...	...	
84579	26	64	22187	31	22	1	1	No Answer	
47851	10	145	35978	9	12	3	1	11-20	
165143	37	97	30961	46	23	4	2	71-80	
51210	46	168	44138	11	16	3	1	21-30	
116670	46	165	44448	30	16	1	1	No Answer	

37738 rows × 36 columns

Input and Target Col's

```
input_cols = list(training_df.columns)
input_cols.remove('Rating')
input_cols.remove('AGE')

target_col = 'Rating'


training_inputs = training_df[input_cols].copy()
training_targets = training_df[target_col].copy()


validation_inputs = validation_df[input_cols].copy()
validation_targets = validation_df[target_col].copy()


test_inputs = test_merge_df[input_cols].copy()


training_inputs
```

	Artist	Track	User	Time	HEARD_OF	OWN_ARTIST_MUSIC	LIKE_ARTIST	words_score
168265	35	88	30594	23	1	1	No Answer	-1.0

```
validation_inputs
```

	Artist	Track	User	Time	HEARD_OF	OWN_ARTIST_MUSIC	LIKE_ARTIST	words_score
109566	4	11	5357	18	2	1	No Answer	7.0
49742	20	44	17177	21	1	1	No Answer	7.0
92733	28	73	23226	22	3	2	41-50	2.0
38465	22	122	32594	0	4	2	91-100	21.0
20298	31	79	26606	11	1	1	No Answer	-3.0
...	...	...	...	...	...	...	...	...
84579	26	64	22187	22	1	1	No Answer	5.0
47851	10	145	35978	12	3	1	11-20	-1.0
165143	37	97	30961	23	4	2	71-80	5.0
51210	46	168	44138	16	3	1	21-30	0.0
116670	46	165	44448	16	1	1	No Answer	3.0

37738 rows × 34 columns

```
test_inputs
```

	Artist	Track	User	Time	HEARD_OF	OWN_ARTIST_MUSIC	LIKE_ARTIST	words_score
0	1	6	3475	18	3	1	1-10	2.0
1	6	149	39210	15	1	1	No Answer	NaN
2	40	177	47861	17	1	1	No Answer	-2.0
3	31	79	27413	11	1	1	No Answer	0.0
4	26	66	23232	22	1	1	No Answer	0.0
...	...	...	...	...	...	...	...	...
125789	14	95	30004	23	2	1	No Answer	12.0
125790	10	25	8186	7	1	1	No Answer	6.0
125791	40	146	38180	13	2	1	No Answer	3.0
125792	22	113	32918	0	3	1	41-50	2.0
125793	2	70	24231	22	1	1	No Answer	4.0

125794 rows × 34 columns

Segregation of Numeric and Catego... Cols

```
numeric_cols = ['Artist', 'Track', 'User', 'Time', 'HEARD_OF', 'OWN_ARTIST_MUSIC', 'words_sco
```

```
categorical_cols = ['LIKE_ARTIST', 'GENDER', 'WORKING', 'REGION', 'LIST_OWN', 'LIST_BACK', 'A
```

```
training_inputs[numeric_cols].describe()
```

	Artist	Track	User	Time	HEARD_OF	OWN_ART:
count	150952.000000	150952.000000	150952.000000	150952.000000	150952.000000	1509
mean	22.206688	86.473912	26463.279082	15.656050	1.877252	
std	14.478913	55.988137	13628.240967	6.443697	1.057053	
min	0.000000	0.000000	0.000000	0.000000	1.000000	
25%	10.000000	36.000000	17700.000000	12.000000	1.000000	
50%	22.000000	80.000000	27805.000000	17.000000	1.000000	
75%	35.000000	142.000000	35924.000000	21.000000	3.000000	
max	49.000000	183.000000	50927.000000	23.000000	4.000000	

```
training_inputs[numeric_cols].info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 150952 entries, 168265 to 53332
Data columns (total 27 columns):
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   Artist             150952 non-null  int64  
 1   Track              150952 non-null  int64  
 2   User               150952 non-null  int64  
 3   Time               150952 non-null  int64  
 4   HEARD_OF           150952 non-null  int64  
 5   OWN_ARTIST_MUSIC   150952 non-null  int64  
 6   words_score        149299 non-null  float64
 7   MUSIC              141389 non-null  float64
 8   Q1                 141389 non-null  float64
 9   Q2                 141389 non-null  float64
 10  Q3                 141389 non-null  float64
 11  Q4                 141389 non-null  float64
 12  Q5                 141389 non-null  float64
 13  Q6                 141389 non-null  float64
 14  Q7                 141389 non-null  float64
 15  Q8                 141389 non-null  float64
 16  Q9                 141389 non-null  float64
 17  Q10                141389 non-null  float64
 18  Q11                141389 non-null  float64
 19  Q12                141389 non-null  float64
 20  Q13                141389 non-null  float64
 21  Q14                141389 non-null  float64
 22  Q15                141389 non-null  float64
 23  Q16                114178 non-null  float64
 24  Q17                141389 non-null  float64
 25  Q18                112310 non-null  float64
 26  Q19                112310 non-null  float64
```

```
dtypes: float64(21), int64(6)
memory usage: 32.2 MB
```

```
training_inputs[categorical_cols].nunique()
```

```
LIKE_ARTIST    11
GENDER          3
WORKING        14
REGION          6
LIST_OWN       10
LIST_BACK      10
AGE_GROUP       6
dtype: int64
```

Replacing Missing Data

```
training_merge_df[numeric_cols].isna().sum()
```

```
Artist                  0
Track                   0
User                    0
Time                    0
HEARD_OF                0
OWN_ARTIST_MUSIC        0
words_score          2054
MUSIC               11857
Q1                  11857
Q2                  11857
Q3                  11857
Q4                  11857
Q5                  11857
Q6                  11857
Q7                  11857
Q8                  11857
Q9                  11857
Q10                 11857
Q11                 11857
Q12                 11857
Q13                 11857
Q14                 11857
Q15                 11857
Q16                 45936
Q17                 11857
Q18                 48145
Q19                 48145
dtype: int64
```

```
from sklearn.impute import SimpleImputer
```

```
imputer = SimpleImputer(strategy='mean')
```

```
imputer.fit(training_merge_df[numeric_cols])
```

```
SimpleImputer()
```

```
training_inputs[numeric_cols] = imputer.transform(training_inputs[numeric_cols])
validation_inputs[numeric_cols] = imputer.transform(validation_inputs[numeric_cols])
test_inputs[numeric_cols] = imputer.transform(test_inputs[numeric_cols])
```

```
training_inputs[numeric_cols].isna().sum()
```

```
Artist              0
Track               0
User                0
Time                0
HEARD_OF            0
OWN_ARTIST_MUSIC    0
words_score         0
MUSIC               0
Q1                  0
Q2                  0
Q3                  0
Q4                  0
Q5                  0
Q6                  0
Q7                  0
Q8                  0
Q9                  0
Q10                 0
Q11                 0
Q12                 0
Q13                 0
Q14                 0
Q15                 0
Q16                 0
Q17                 0
Q18                 0
Q19                 0
dtype: int64
```

Scaling of Numeric Col's

```
from sklearn.preprocessing import MinMaxScaler
```

```
scaler = MinMaxScaler()
```

```
scaler.fit(training_merge_df[numeric_cols])
```

```
MinMaxScaler()
```

```
training_inputs[numeric_cols] = scaler.transform(training_inputs[numeric_cols])
validation_inputs[numeric_cols] = scaler.transform(validation_inputs[numeric_cols])
test_inputs[numeric_cols] = scaler.transform(test_inputs[numeric_cols])
```

```
training_inputs[numeric_cols].describe()
```

	Artist	Track	User	Time	HEARD_OF	OWN_ART
count	150952.000000	150952.000000	150952.000000	150952.000000	150952.000000	1509
mean	0.453198	0.472535	0.519632	0.680698	0.292417	
std	0.295488	0.305946	0.267603	0.280161	0.352351	
min	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.204082	0.196721	0.347556	0.521739	0.000000	
50%	0.448980	0.437158	0.545978	0.739130	0.000000	
75%	0.714286	0.775956	0.705402	0.913043	0.666667	
max	1.000000	1.000000	1.000000	1.000000	1.000000	

```
training_inputs[numeric_cols].info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 150952 entries, 168265 to 53332
Data columns (total 27 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   Artist            150952 non-null  float64
 1   Track             150952 non-null  float64
 2   User              150952 non-null  float64
 3   Time              150952 non-null  float64
 4   HEARD_OF          150952 non-null  float64
 5   OWN_ARTIST_MUSIC  150952 non-null  float64
 6   words_score       150952 non-null  float64
 7   MUSIC             150952 non-null  float64
 8   Q1                150952 non-null  float64
 9   Q2                150952 non-null  float64
 10  Q3                150952 non-null  float64
 11  Q4                150952 non-null  float64
 12  Q5                150952 non-null  float64
 13  Q6                150952 non-null  float64
 14  Q7                150952 non-null  float64
```

```
 15  Q8                 150952 non-null  float64
 16  Q9                 150952 non-null  float64
 17  Q10                150952 non-null  float64
 18  Q11                150952 non-null  float64
 19  Q12                150952 non-null  float64
 20  Q13                150952 non-null  float64
 21  Q14                150952 non-null  float64
 22  Q15                150952 non-null  float64
 23  Q16                150952 non-null  float64
 24  Q17                150952 non-null  float64
 25  Q18                150952 non-null  float64
 26  Q19                150952 non-null  float64
dtypes: float64(27)
memory usage: 32.2 MB
```

```
# Encoding Categorical data
```

```
training_merge_df[categorical_cols].isna().sum()
```

```
LIKE_ARTIST    0
GENDER         0
WORKING        0
REGION         0
LIST_OWN       0
LIST_BACK      0
AGE_GROUP      0
dtype: int64
```

```
training_merge_df[categorical_cols].nunique()
```

```
LIKE_ARTIST    11
GENDER          3
WORKING        14
REGION          6
LIST_OWN       10
LIST_BACK      10
AGE_GROUP       6
dtype: int64
```

```
from sklearn.preprocessing import OneHotEncoder
```

```
encoder = OneHotEncoder(sparse=False, handle_unknown='ignore')
```

```
encoder.fit(training_merge_df[categorical_cols])
```

```
OneHotEncoder(handle_unknown='ignore', sparse=False)
```

```
encoded_cols = list(encoder.get_feature_names(categorical_cols));
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning:

Function get_feature_names is deprecated; get_feature_names is deprecated in 1.0 and wi
```

```
training_inputs[encoded_cols] = encoder.transform(training_inputs[categorical_cols])
validation_inputs[encoded_cols] = encoder.transform(validation_inputs[categorical_cols])
test_inputs[encoded_cols] = encoder.transform(test_inputs[categorical_cols])
```

```
training_inputs
```

```
# Saving to Disk
```

```
print('training_inputs:', training_inputs.shape)
print('training_targets:', training_targets.shape)
print('validation_inputs:', validation_inputs.shape)
print('validation_targets:', validation_targets.shape)
print('test_inputs:', test_inputs.shape)
```

```
training_inputs: (150952, 94)
training_targets: (150952,)
validation_inputs: (37738, 94)
validation_targets: (37738,)
test_inputs: (125794, 94)
```

```
!pip install pyarrow --quiet
```

```
training_inputs.to_parquet('training_inputs.parquet')
validation_inputs.to_parquet('validation_inputs.parquet')
test_inputs.to_parquet('test_inputs.parquet')
```

```
pd.DataFrame(training_targets).to_parquet('training_targets.parquet')
pd.DataFrame(validation_targets).to_parquet('validation_targets.parquet')
```

Getting Data Back

97594 0.306122 0.180328 0.256446 0.826087 0.000000 0.0 No Answer

```
training_inputs = pd.read_parquet('training_inputs.parquet')
validation_inputs = pd.read_parquet('validation_inputs.parquet')
test_inputs = pd.read_parquet('test_inputs.parquet')

training_targets = pd.read_parquet('training_targets.parquet')[target_col]
validation_targets = pd.read_parquet('validation_targets.parquet')[target_col]
```

```
print('training_inputs:', training_inputs.shape)
print('training_targets:', training_targets.shape)
print('validation_inputs:', validation_inputs.shape)
print('validation_targets:', validation_targets.shape)
print('test_inputs:', test_inputs.shape)
```

```
training_inputs: (150952, 94)
training_targets: (150952,)
validation_inputs: (37738, 94)
validation_targets: (37738,)
test_inputs: (125794, 94)
```

Starting Modeling

```
X_training = training_inputs[numeric_cols + encoded_cols]

X_validation = validation_inputs[numeric_cols + encoded_cols]

X_test = test_inputs[numeric_cols + encoded_cols]
```

Training

```
from xgboost import XGBRegressor
```

```
model = XGBRegressor(n_jobs=0)
```

```
model.fit(X_training, training_targets)
```

```
[11:34:29] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now de
XGBRegressor(n_jobs=0)
```

```
prediction = model.predict(X_training)
```

```
from sklearn.metrics import mean_squared_error

def rmse(a, b):
  return mean_squared_error(a, b, squared=False)
```

```
rmse(prediction, training_targets)
```

```
15.97886769351449
```

```
impt_df = pd.DataFrame({'feature': X_training.columns,
'importance': model.feature_importances_}).sort_values('importance', ascending=False)
```

```
impt_df.head(10)
```

	feature	importance
6	words_score	0.467572
5	OWN_ARTIST_MUSIC	0.083540
4	HEARD_OF	0.042305
3	Time	0.029398
18	Q11	0.026738
14	Q7	0.025278
36	LIKE_ARTIST_91-100	0.022223
15	Q8	0.016989

Hyperparametre Tuning

```
def test_params(**params):
  model = XGBRegressor(n_jobs=-1, **params)
  model.fit(X_training, training_targets)
  training_rmse = rmse(model.predict(X_training), training_targets)
  validation_rmse = rmse(model.predict(X_validation), validation_targets)
  print('Training RMSE: {}, Validation RMSE: {}'.format(training_rmse, validation_rmse))
```

```
test_params(n_estimators=100)
```

```
[11:35:23] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now de
Training RMSE: 15.97886769351449, Validation RMSE: 15.909823636844786
```

```
test_params(n_estimators=200)
```

```
[11:35:52] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now de
Training RMSE: 15.77299591949312, Validation RMSE: 15.741147592019022
```

```
test_params(n_estimators=400)
```

```
[11:36:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now de
Training RMSE: 15.526375941069244, Validation RMSE: 15.569182444072737
```

```
test_params(n_estimators=800)
```

```
[11:38:41] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now de
Training RMSE: 15.193733025548545, Validation RMSE: 15.370873997573677
```

Tree depth & Learning rate

```
test_params(n_estimators=175, max_depth=8, learning_rate=0.3)
```

```
[11:42:28] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now de
Training RMSE: 10.777195811333101, Validation RMSE: 14.35447977385776
```

```
test_params(n_estimators=175, max_depth=8, learning_rate=0.2)
```

```
[11:44:56] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now de
Training RMSE: 11.705383800905652, Validation RMSE: 14.38046099437259
```

```
test_params(booster='gblinear', n_estimators=400)
```

```
[11:47:14] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now de
Training RMSE: 21.690106222953066, Validation RMSE: 21.679318734157885
```

```
test_params(n_estimators=500, max_depth=9, learning_rate=0.15)
```

```
[11:48:11] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now de
Training RMSE: 8.170350841099964, Validation RMSE: 14.031425644970993
```

```
test_params(n_estimators=1000, max_depth=10, learning_rate=0.10, subsample=0.9, colsample_byt
```

```
[11:56:22] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now de
Training RMSE: 5.912378909481084, Validation RMSE: 14.06289233604193
```

```
from sklearn.model_selection import KFold
```

```
def train_and_evaluate(X_train_k, Y_train_k, X_val_k, Y_val_k, **params):
  model = XGBRegressor(n_jobs=-1, **params)
  model.fit(X_train_k, Y_train_k)
  train_rmse = rmse(model.predict(X_train_k), Y_train_k)
  val_rmse = rmse(model.predict(X_val_k), Y_val_k)
  return model, train_rmse, val_rmse
```

```
kfold = KFold(n_splits=5)
```

```
models = []

for train_idxs, val_idxs in kfold.split(X_training):
  X_train_k, Y_train_k = X_training.iloc[train_idxs], training_targets.iloc[train_idxs]
  X_val_k, Y_val_k = X_training.iloc[val_idxs], training_targets.iloc[val_idxs]
  model, train_rmse, val_rmse = train_and_evaluate(X_train_k, Y_train_k, X_val_k, Y_val_k, n_
  models.append(model)
  print('Train RMSE: {}, Validation RMSE: {}'.format(train_rmse, val_rmse))
```

```
[12:11:43] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now de
Train RMSE: 8.87027198507148, Validation RMSE: 14.309372055537372
[12:17:14] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now de
Train RMSE: 8.930045003488926, Validation RMSE: 14.326546756947248
[12:22:31] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now de
Train RMSE: 8.883624648870502, Validation RMSE: 14.325172790349075
[12:27:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now de
Train RMSE: 8.964643709238226, Validation RMSE: 14.379924947602829
[12:33:01] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now de
Train RMSE: 8.926999279240505, Validation RMSE: 14.242177838373042
```

```
def predict_avg(models, inputs):
    return np.mean([model.predict(inputs) for model in models], axis=0)
```

```
preds_kfold = predict_avg(models, X_validation)
rmse(preds_kfold, validation_targets)
```

```
13.852739686869784
```

```
test_preds = predict_avg(models, X_test)
```

Final Answer

```
test_preds.shape
```

```
(125794,)
```

Model 2

RandomForestRegressor

```
from sklearn.ensemble import RandomForestRegressor
```

```
model_randomForestRegressor = RandomForestRegressor()
```

```
model_randomForestRegressor.fit(X_training, training_targets)
```

```
RandomForestRegressor()
```

```
def test_params(**params):
    model = RandomForestRegressor(random_state=42, n_jobs=-1, **params).fit(X_training, train
    return model.score(X_training, training_targets), model.score(X_validation, validation_ta
```

Hyperparameter Tuning

```
test_params(max_depth=100, max_leaf_nodes=2**4)
```

```
(0.44986499029102844, 0.4560022614096648)
```

```
test_params(max_depth=400, max_leaf_nodes=2**10)
```

```
(0.5880712862974407, 0.5376298342989975)
```

```
test_params(max_depth=600, max_leaf_nodes=2**15)
```

```
(0.9241145573397733, 0.5908789148855196)
```

```
test_params(max_depth=1000, max_leaf_nodes=2**25)
```

```
(0.9419978303117578, 0.5872249941906067)
```

Now we can see that the optimum value occurs at `max_depth=600` & `max_leaf_nodes=2**15`

Now we will use these values to predict the performance of RFR.

Performance of RFR

```
preds_randomForestRegressor = model_randomForestRegressor.predict(X_validation)
rmse(preds_randomForestRegressor, validation_targets)
```

```
14.504533421368059
```

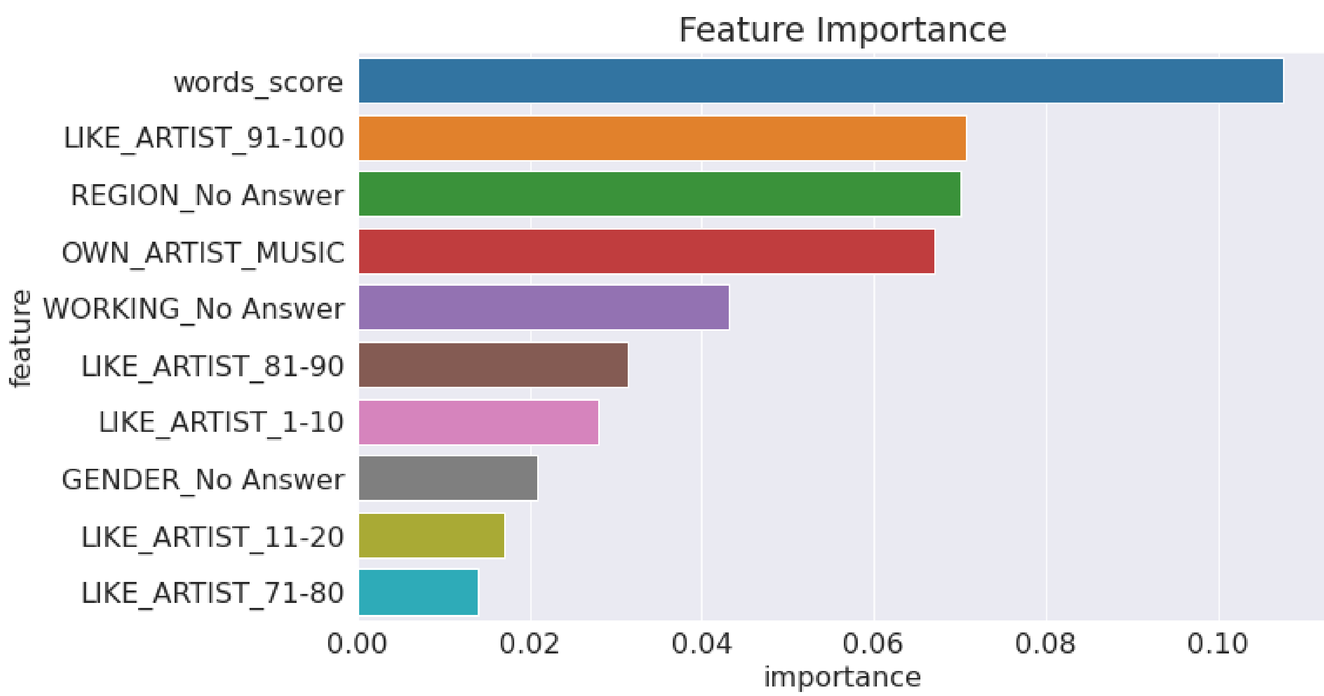
Performance of RFR is less than XGBR

So, we will continue with XGBR Model

Importance of columns

```
importance_df = pd.DataFrame({'feature': X_training.columns,
'importance': model.feature_importances_}).sort_values('importance', ascending=False)
```

```
plt.figure(figsize=(10,6))
plt.title('Feature Importance')
sns.barplot(data=importance_df.head(10), x='importance', y='feature');
```



Saving the model

```
import joblib
```

```
song_recommendation_ml = {
    'model': models,
    'imputer': imputer,
    'scaler': scaler,
    'encoder': encoder,
    'input_cols': input_cols,
```

```
    'target_cols': target_col,
    'numeric_cols': numeric_cols,
    'categorical_cols': categorical_cols,
    'encoded_cols': encoded_cols
}

joblib.dump(song_recommendation_ml, 'song_recommendation_ml')
```

```
['song_recommendation_ml']
```

```
['song_recommendation_ml']
```

```
['song_recommendation_ml']
```

Conclusion

I downloded this dataset from kaggle. Then after I imported the requried python libraries. Now, I started cleaning the dataset like deleting the rows in which data is missing or substituting the average value of the column in the missing data. Then to get the insights from the columns of the dataset, I started to make the visualizations of the dataset. After I got th einsights from the dataset and the relationship betweeen the columns of the dataset I started to make the model. I used XGBoosta and RFR models. After checking the performances of both the models, I had come to a conclusion that XGBoost model had high performance than RFR. So, at last I used the XGBoost model to predict the values of dataset.

References and Future Work

References: The websites that I found useful during this project work are Scikit-learn, Stackoverflow, W3schools, GFG, and many more.

- GFG
- scikit-learn
- GFG
- Stack overflow
- Medium

Future work

Now, I will continue on this project, by adding the songs data and selecting the recommended song for the listner from the dataset. In the dataset of the song we have to differentiate the songs by the lyrics in the song, by the lyrics of the song we can is it a sad, happy or romatic song. By the words in