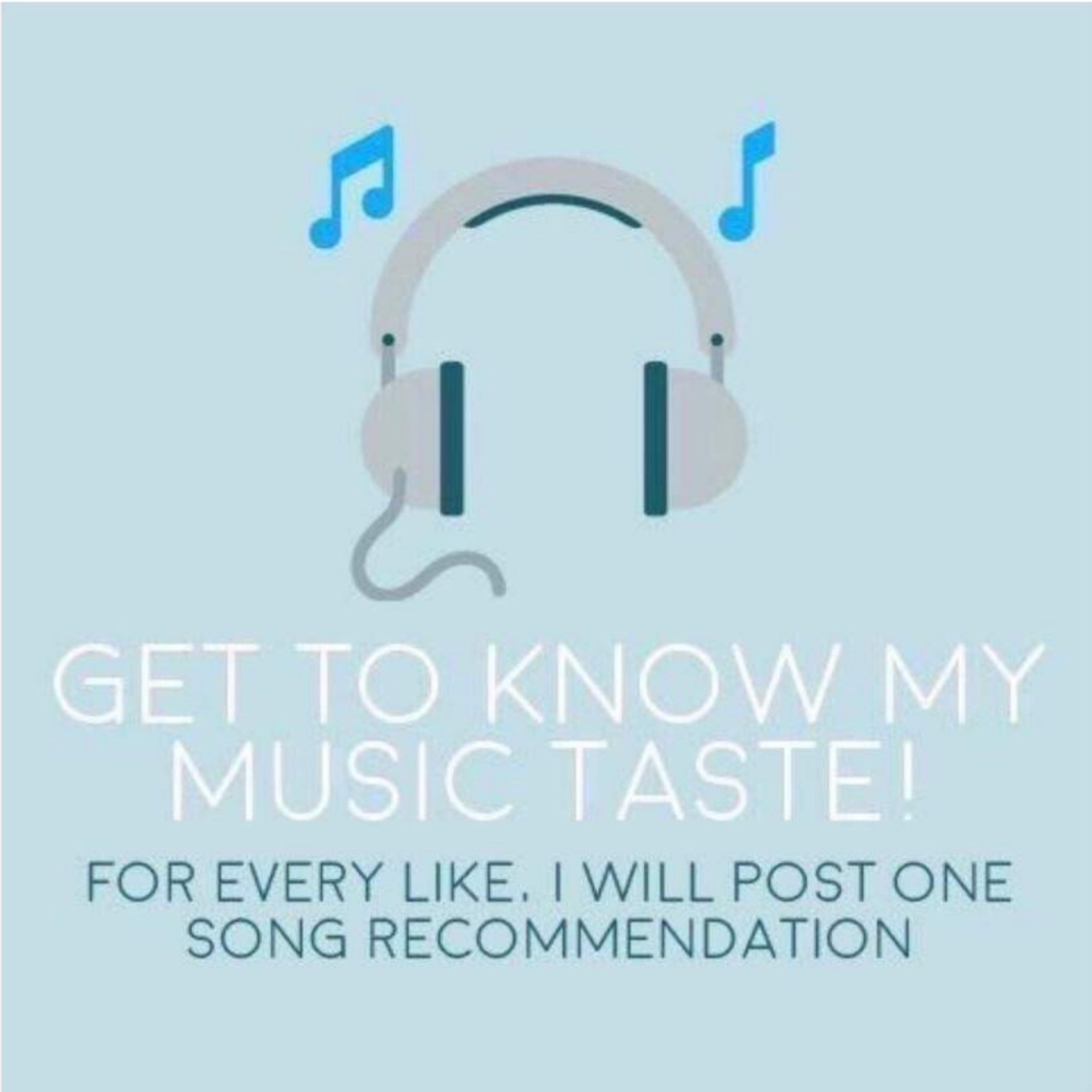
Song-Recommendation-ML



`~~~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/public/simple/
Collecting plotly==5.11.0
  Downloading plotly-5.11.0-py2.py3-none-any.whl (15.3 MB)
     |████████████████████████████████| 15.3 MB 5.1 MB/s
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.8/dist-packages (from plotly==5.11.0) (8.1.0)
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/public/simple/
Collecting opendatasets
  Downloading opendatasets-0.1.22-py3-none-any.whl (15 kB)
Requirement already satisfied: click in /usr/local/lib/python3.8/dist-packages (from opendatasets) (7.1.2)
Requirement already satisfied: kaggle in /usr/local/lib/python3.8/dist-packages (from opendatasets) (1.5.12)
Requirement already satisfied: tqdm in /usr/local/lib/python3.8/dist-packages (from opendatasets) (4.64.1)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.8/dist-packages (from kaggle->opendatasets) (7.0.0)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.8/dist-
```

```
packages (from kaggle->opendatasets) (2.8.2)
Requirement already satisfied: certifi in /usr/local/lib/python3.8/dist-packages (from
kaggle->opendatasets) (2022.12.7)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.8/dist-packages
(from kaggle->opendatasets) (1.15.0)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.8/dist-packages (from
kaggle->opendatasets) (1.24.3)
Requirement already satisfied: requests in /usr/local/lib/python3.8/dist-packages (from
kaggle->opendatasets) (2.23.0)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.8/dist-
packages (from python-slugify->kaggle->opendatasets) (1.3)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.8/dist-
packages (from requests->kaggle->opendatasets) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.8/dist-packages
(from requests->kaggle->opendatasets) (2.10)
Installing collected packages: opendatasets
Successfully installed opendatasets-0.1.22
```

```python
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

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

```
Please provide your Kaggle credentials to download this dataset. Learn more:
http://bit.ly/kaggle-creds
Your Kaggle username: pdheeraj2002
Your Kaggle Key: ··········
Downloading MusicHackathon.zip to ./MusicHackathon

100%|██████████| 6.62M/6.62M [00:00<00:00, 49.3MB/s]


Extracting archive ./MusicHackathon/MusicHackathon.zip to ./MusicHackathon
```

```python
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

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

118301 rows × 88 columns

```
users_df
```

	RESPID	GENDER	AGE	WORKING	REGION	MUSIC	LIST_OWN	LIST_BACK	Q1	Q2	Q3	Q4

	RESPID	GENDER	AGE	WORKING	REGION	MUSIC	LIST_OWN	LIST_BACK	Q1	Q2	Q3	Q4
0	36927	Female	60.0	Other	South	Music is important to me but not necessarily m...	1 hour	NaN	49.0	50.0	49.0	50.(
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	55.0	62.0	9.(
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	50.0	9.0	8.(
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	80.0	88.0	88.(
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	79.0	78.0	73.(
...	...	...	...	...	...	...	...	...	...	...	...	..
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	73.0	33.0	6.(
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	50.0	49.0	58.(
48642	28753	Female	25.0	Employed 30+ hours a week	Midlands	Music means a lot to me and is a passion of mine	2 hours	6 hours	89.0	89.0	89.0	6.(
48643	26197	Male	44.0	Employed 30+ hours a week	Midlands	Music means a lot to me and is a passion of mine	2 hours	4 hours	95.0	97.0	97.0	98.(
48644	16225	Female	43.0	NaN	North	I like music but it does not feature heavily i...	NaN	2	49.0	48.0	50.0	51.(

48645 rows × 27 columns

```
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
53  Original          118301 non-null  int64
54  Talented          118301 non-null  int64
55  Worldly           1040 non-null    float64
56  Distinctive       118301 non-null  int64
57  Approachable      118301 non-null  int64
58  Genius            20724 non-null   float64
59  Trendsetter       118301 non-null  int64
60  Noisy             97577 non-null   float64
61  Upbeat            117261 non-null  float64
62  Relatable         46254 non-null   float64
63  Energetic         118301 non-null  int64
64  Exciting          20724 non-null   float64
65  Emotional         20724 non-null   float64
66  Nostalgic         1040 non-null    float64
67  None of these     118301 non-null  int64
68  Progressive       1040 non-null    float64
69  Sexy              118301 non-null  int64
70  Over              90157 non-null   float64
71  Rebellious        20724 non-null   float64
72  Fake              97577 non-null   float64
73  Cheesy            97577 non-null   float64
74  Popular           19684 non-null   float64
75  Superstar         46254 non-null   float64
76  Relaxed           20724 non-null   float64
77  Intrusive         26154 non-null   float64
78  Unoriginal        97577 non-null   float64
79  Dated             117261 non-null  float64
80  Iconic            1040 non-null    float64
```

```
 81  Unapproachable    97577 non-null   float64
 82  Classic           105235 non-null  float64
 83  Playful           97577 non-null   float64
 84  Arrogant          97577 non-null   float64
 85  Warm              118301 non-null  int64
 86  Soulful           19684 non-null   float64
 87  Unnamed: 87       0 non-null       float64
dtypes: float64(64), int64(22), object(2)
memory usage: 79.4+ MB
```

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', '
```

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

```
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	Sophisticated	Aggressive	Edgy	S
61341	17	14077	Heard of and listened to music EVER	Own none of their music	97.0	NaN	1.0	NaN	0	
112317	22	32417	Listened to recently	Own a lot of their music	92.0	NaN	0.0	NaN	0	
63273	8	9197	Heard of and listened to music EVER	Own a little of their music	93.0	NaN	1.0	NaN	1	

	Artist	User	HEARD_OF	OWN_ARTIST_MUSIC	LIKE_ARTIST	Uninspired	Sophisticated	Aggressive	Edgy	S
60559	22	17873	Heard of and listened to music RECENTLY	Own a lot of their music	92.0	NaN	NaN	0.0	0	
8108	44	43174	Heard of and listened to music RECENTLY	Own a lot of their music	94.0	NaN	NaN	0.0	0	
20171	22	20534	Heard of and listened to music RECENTLY	Own a lot of their music	92.0	NaN	NaN	0.0	1	
54272	40	36860	Heard of and listened to music RECENTLY	Own a lot of their music	99.0	NaN	NaN	0.0	0	
4944	44	41784	Heard of and listened to music RECENTLY	Own all or most of their music	93.0	NaN	NaN	0.0	0	
36830	3	3016	Heard of and listened to music EVER	Own a lot of their music	92.0	NaN	NaN	0.0	0	
104076	4	36477	Heard of and listened to music RECENTLY	Own all or most of their music	95.0	NaN	NaN	0.0	0	
59164	17	15832	Heard of and listened to music RECENTLY	Own all or most of their music	91.0	NaN	0.0	NaN	0	
11566	41	41534	Heard of and listened to music RECENTLY	Own all or most of their music	94.0	NaN	NaN	0.0	0	
71176	15	11608	Heard of and listened to music RECENTLY	Own all or most of their music	94.0	NaN	NaN	0.0	0	
108436	4	37208	Heard of and listened to music RECENTLY	Own a lot of their music	100.0	NaN	NaN	0.0	0	
8423	17	14195	Heard of and listened to music EVER	Own a lot of their music	96.0	NaN	1.0	NaN	0	

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**

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

```
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
118297	4	3935	Heard of and listened to music EVER	Own a little of their music	30.0	1.0
118298	12	11216	Heard of and listened to music RECENTLY	Own none of their music	71.0	6.0
118299	33	35142	Heard of and listened to music EVER	Own none of their music	31.0	3.0
118300	4	3915	Heard of and listened to music EVER	Own a little of their music	46.0	4.0

118301 rows × 6 columns

```
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	Q2	Q3	Q4
0	36927	Female	60.0	Other	South	Music is important to me but not necessarily m...	1 hour	NaN	49.0	50.0	49.0	50.0
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	55.0	62.0	9.0
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	50.0	9.0	8.0
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	80.0	88.0	88.0
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	79.0	78.0	73.0
...	...	...	...	...	...	...	...	...	...	...	...	...
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	73.0	33.0	6.0
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	50.0	49.0	58.0
48642	28753	Female	25.0	Employed 30+ hours a week	Midlands	Music means a lot to me and is a passion of mine	2 hours	6 hours	89.0	89.0	89.0	6.0
48643	26197	Male	44.0	Employed 30+ hours a week	Midlands	Music means a lot to me and is a passion of mine	2 hours	4 hours	95.0	97.0	97.0	98.0
48644	16225	Female	43.0	NaN	North	I like music but it does not feature heavily i...	NaN	2	49.0	48.0	50.0	51.0

48645 rows × 27 columns

```
training_merge_df
```

	Artist	Track	User	Rating	Time	HEARD_OF	OWN_ARTIST_MUSIC	LIKE_ARTIST	words_score
0	40	179	47994	9	17	Never heard of	NaN	NaN	-2.0
1	9	23	8575	58	7	Never heard of	NaN	NaN	5.0
2	46	168	45475	13	16	Never heard of	NaN	NaN	1.0
3	11	153	39508	42	15	Heard of and listened to music EVER	Own none of their music	28.0	4.0
4	14	32	11565	54	19	Heard of and listened to music EVER	Own none of their music	18.0	2.0
...	...	...	...	...	...	...	...	...	...
188685	0	3	1278	29	6	Never heard of	NaN	NaN	3.0
188686	1	6	2839	30	18	Heard of	NaN	NaN	-1.0
188687	10	142	35756	61	12	Heard of	NaN	NaN	3.0
188688	22	54	20163	46	21	Heard of and listened to music RECENTLY	Own a lot of their music	74.0	10.0
188689	47	171	45580	12	4	Heard of and listened to music RECENTLY	Own none of their music	7.0	1.0

188690 rows × 9 columns

```
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	words_score	GENDER	A
0	40	179	47994	9	17	Never heard of	NaN	NaN	-2.0	Female	4
1	9	23	8575	58	7	Never heard of	NaN	NaN	5.0	Female	4
2	46	168	45475	13	16	Never heard of	NaN	NaN	1.0	Male	2
3	11	153	39508	42	15	Heard of and listened to music EVER	Own none of their music	28.0	4.0	Female	6

	Artist	Track	User	Rating	Time	HEARD_OF	OWN_ARTIST_MUSIC	LIKE_ARTIST	words_score	GENDER	A
4	14	32	11565	54	19	Heard of and listened to music EVER	Own none of their music	18.0	2.0	Female	2
...	...	...	...	...	...	...	...	...	...	...	
188685	0	3	1278	29	6	Never heard of	NaN	NaN	3.0	Female	5
188686	1	6	2839	30	18	Heard of	NaN	NaN	-1.0	Male	5
188687	10	142	35756	61	12	Heard of	NaN	NaN	3.0	Female	2
188688	22	54	20163	46	21	Heard of and listened to music RECENTLY	Own a lot of their music	74.0	10.0	Female	3
188689	47	171	45580	12	4	Heard of and listened to music RECENTLY	Own none of their music	7.0	1.0	Female	8

188690 rows × 35 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: 51.8+ MB
```

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

	Artist	Track	User	Rating	Time	HEARD_OF	OWN_ARTIST_MUSIC	LIKE_ARTIST	words_score	GENDER	A
157823	37	101	28680	10	23	Heard of and listened to music EVER	Own none of their music	57.0	-8.0	Male	3
51032	27	71	21320	9	22	Never heard of	NaN	NaN	-1.0	Male	2
57817	42	158	42805	11	16	Heard of	NaN	NaN	-2.0	Female	2

	Artist	Track	User	Rating	Time	HEARD_OF	OWN_ARTIST_MUSIC	LIKE_ARTIST	words_score	GENDER	A
153619	1	4	3026	32	18	Heard of	NaN	NaN	-4.0	Male	5
126819	1	5	2859	6	18	Never heard of	NaN	NaN	-5.0	Male	6
156686	22	55	18557	70	21	Heard of	NaN	NaN	3.0	Male	6
179944	6	148	39101	32	15	NaN	NaN	NaN	NaN	Male	3
31382	6	14	6061	37	7	Heard of and listened to music EVER	Own none of their music	33.0	5.0	Male	2
179943	26	64	22977	51	22	Never heard of	NaN	NaN	1.0	NaN	N
10229	1	6	2450	8	18	Heard of	NaN	NaN	1.0	Female	4
57818	4	11	36352	28	13	Heard of and listened to music EVER	Own none of their music	32.0	0.0	Male	6
67454	46	167	43350	51	16	Heard of and listened to music EVER	DonÕt know	47.0	2.0	Female	4
107446	49	182	50454	27	17	Never heard of	NaN	NaN	-1.0	Female	5
136711	28	73	23570	9	22	Never heard of	NaN	NaN	-2.0	NaN	N
110552	6	16	7398	38	7	Heard of and listened to music RECENTLY	Own a little of their music	46.0	4.0	Male	2

Merging the test dataset

```python
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'])
```

```python
test_merge_df
```

	Artist	Track	User	Time	HEARD_OF	OWN_ARTIST_MUSIC	LIKE_ARTIST	words_score	GENDER	AGE	
0	1	6	3475	18	Heard of and listened to music EVER	Own none of their music	3.0	2.0	Female	48.0	En h
1	6	149	39210	15	NaN	NaN	NaN	NaN	Male	28.0	En h
2	40	177	47861	17	Never heard of	NaN	NaN	-2.0	Female	59.0	
3	31	79	27413	11	Never heard of	NaN	NaN	0.0	Female	25.0	p t
4	26	66	23232	22	Never heard of	NaN	NaN	0.0	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	...	
125789	14	95	30004	23	Heard of	NaN	NaN	12.0	Male	36.0	En h
125790	10	25	8186	7	Never heard of	NaN	NaN	6.0	Male	49.0	
125791	40	146	38180	13	Heard of	NaN	NaN	3.0	Female	40.0	hou
125792	22	113	32918	0	Ever heard music by	Own none of their music	48.0	2.0	Female	48.0	
125793	2	70	24231	22	Never heard of	NaN	NaN	4.0	Male	43.0	En h

125794 rows × 34 columns

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.

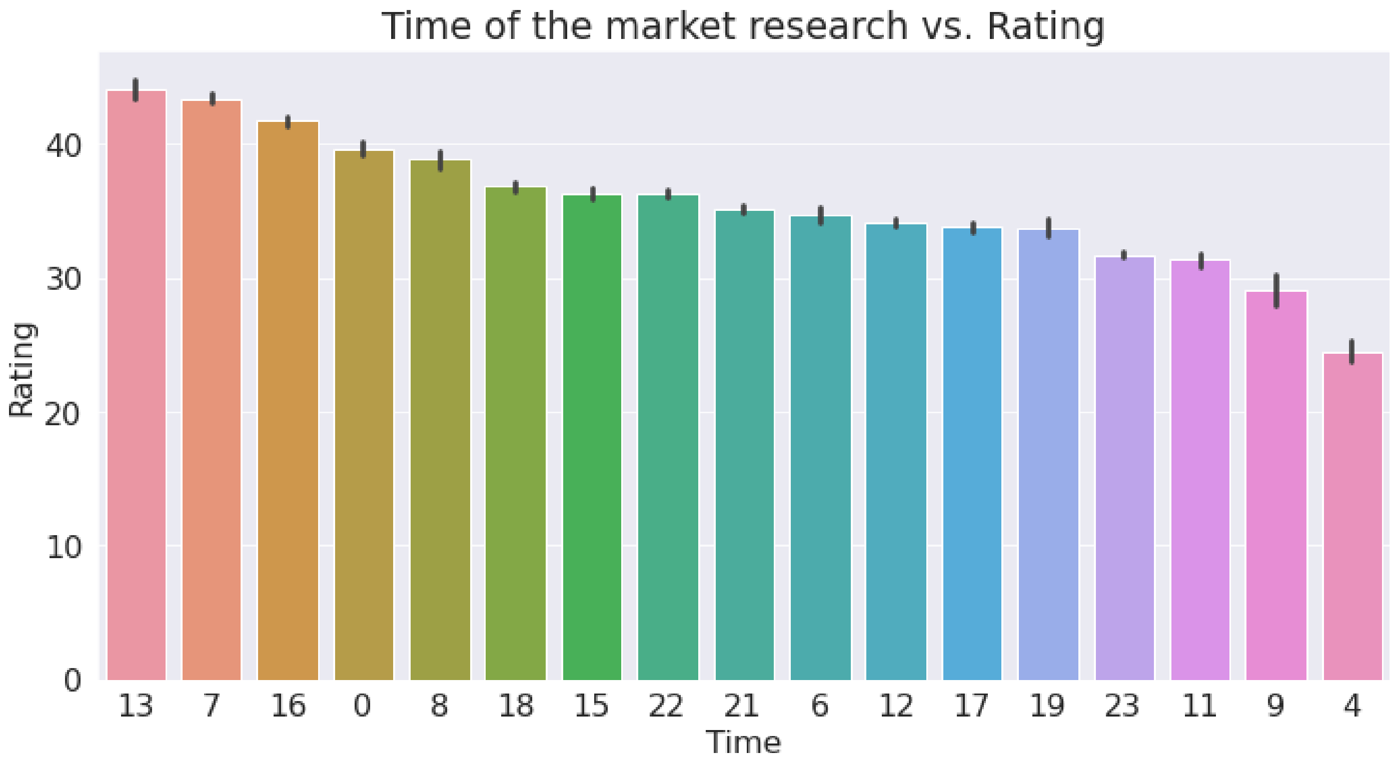
```
# Do you love or hate the the song?
px.histogram(training_merge_df, x='Rating', nbins=101, marginal='box', title='Rating(Lo
```

```
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')
```

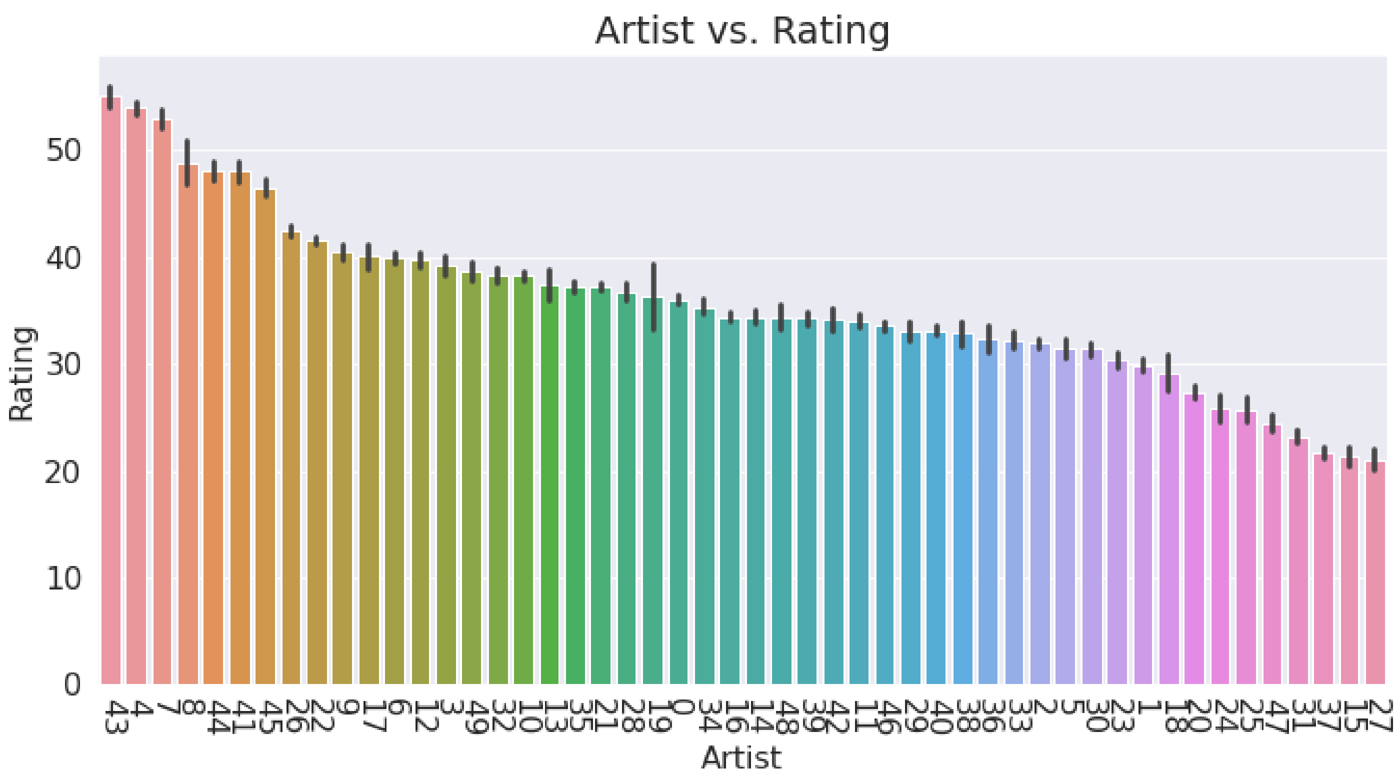
```
plot_order= training_merge_df.groupby('Time')['Rating'].mean().sort_values(ascending=Fa
```

```
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();
```

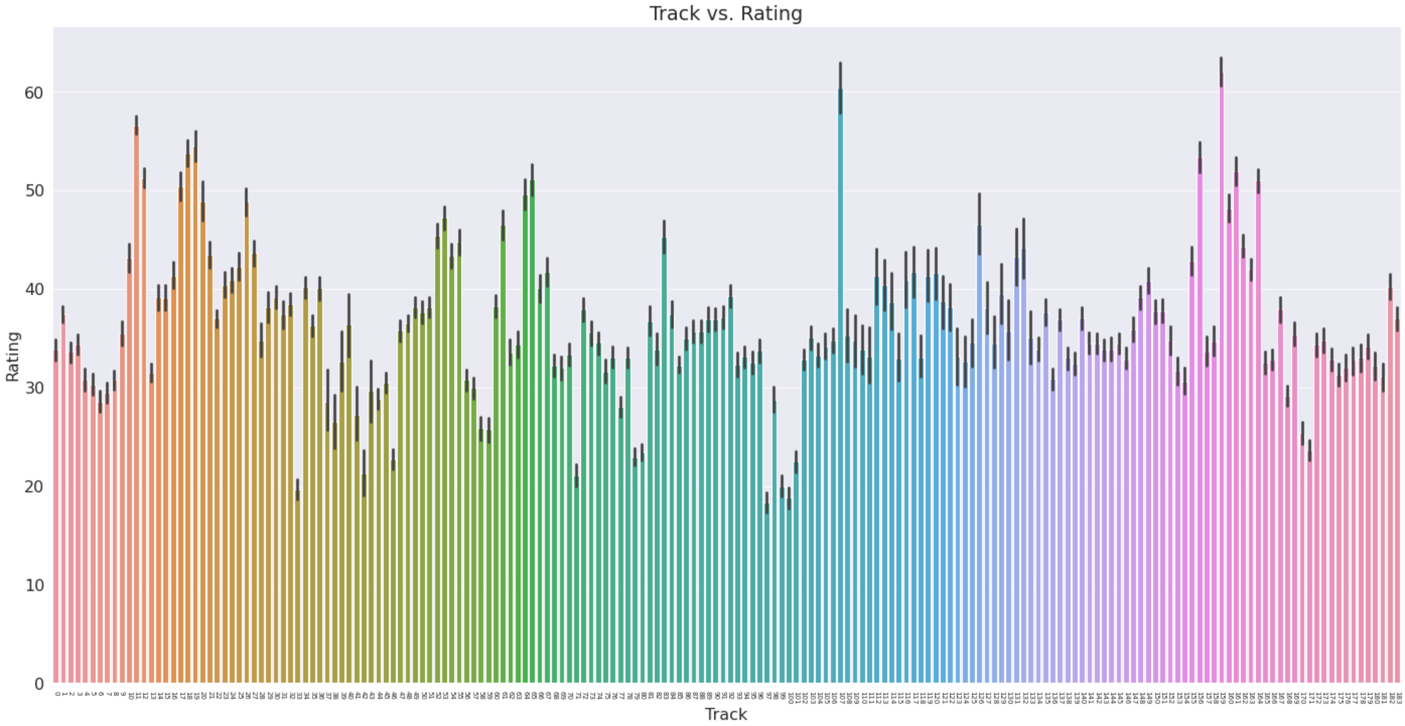



```
plot_order= training_merge_df.groupby('Artist')['Rating'].mean().sort_values(ascending=
```

```
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(
```

```
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 t
training_merge_df['HEARD_OF'].replace(['Listened to recently'], 'Heard of and listened 
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 mu
test_merge_df['HEARD_OF'].replace(['Listened to recently'], 'Heard of and listened to m
test_merge_df['HEARD_OF'].fillna('Never heard of', inplace=True)
```

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

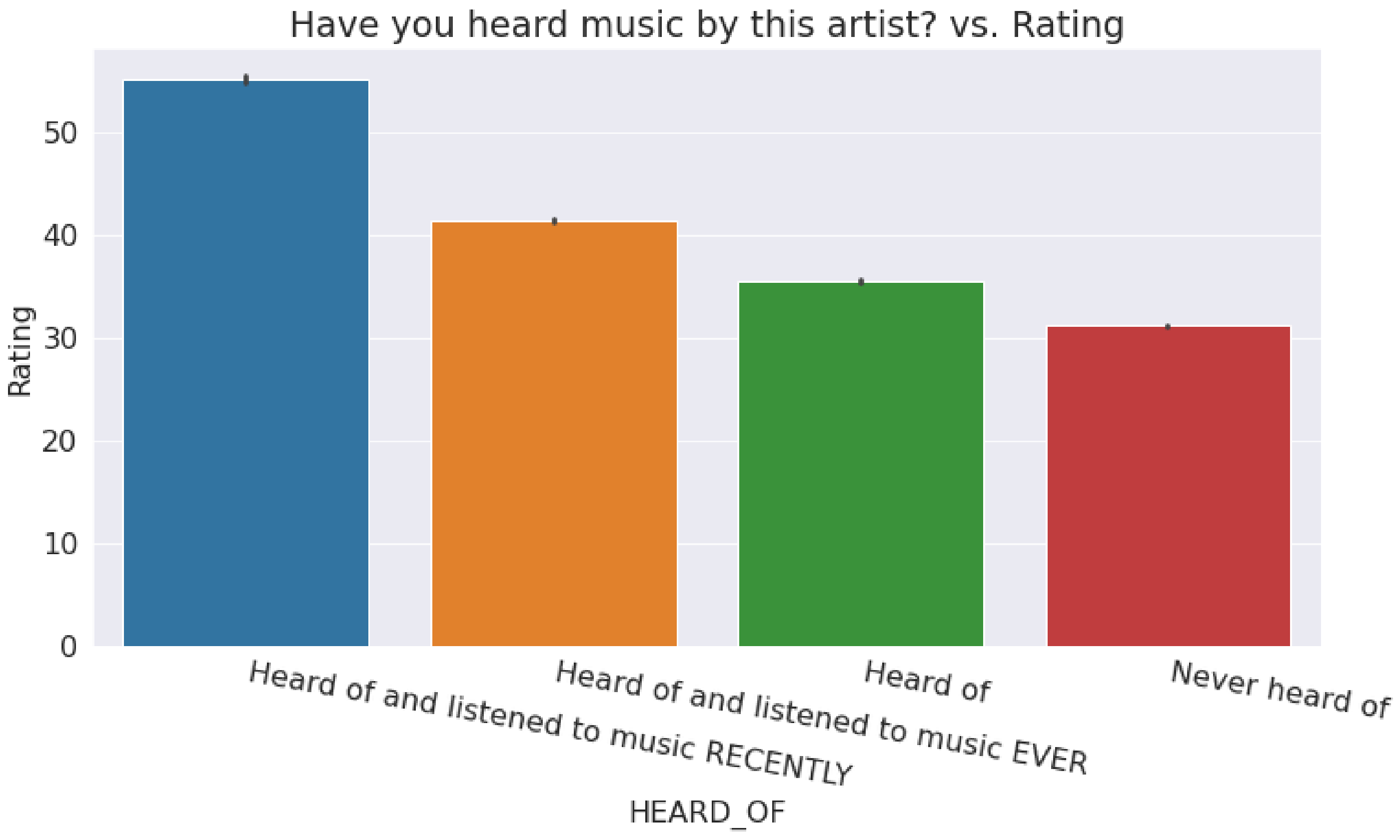
```
array(['Heard of and listened to music EVER', 'Never heard of',
       'Heard of', 'Heard of and listened to music RECENTLY'],
      dtype=object)
```

```
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(ascendin
```

```
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'
training_merge_df['OWN_ARTIST_MUSIC'].replace(['DonÍt know'], 'Own none of their music'
training_merge_df['OWN_ARTIST_MUSIC'].replace(['don`t know'], 'Own none of their music'
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', ir
test_merge_df['OWN_ARTIST_MUSIC'].replace(['DonÍt know'], 'Own none of their music', ir
```

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

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

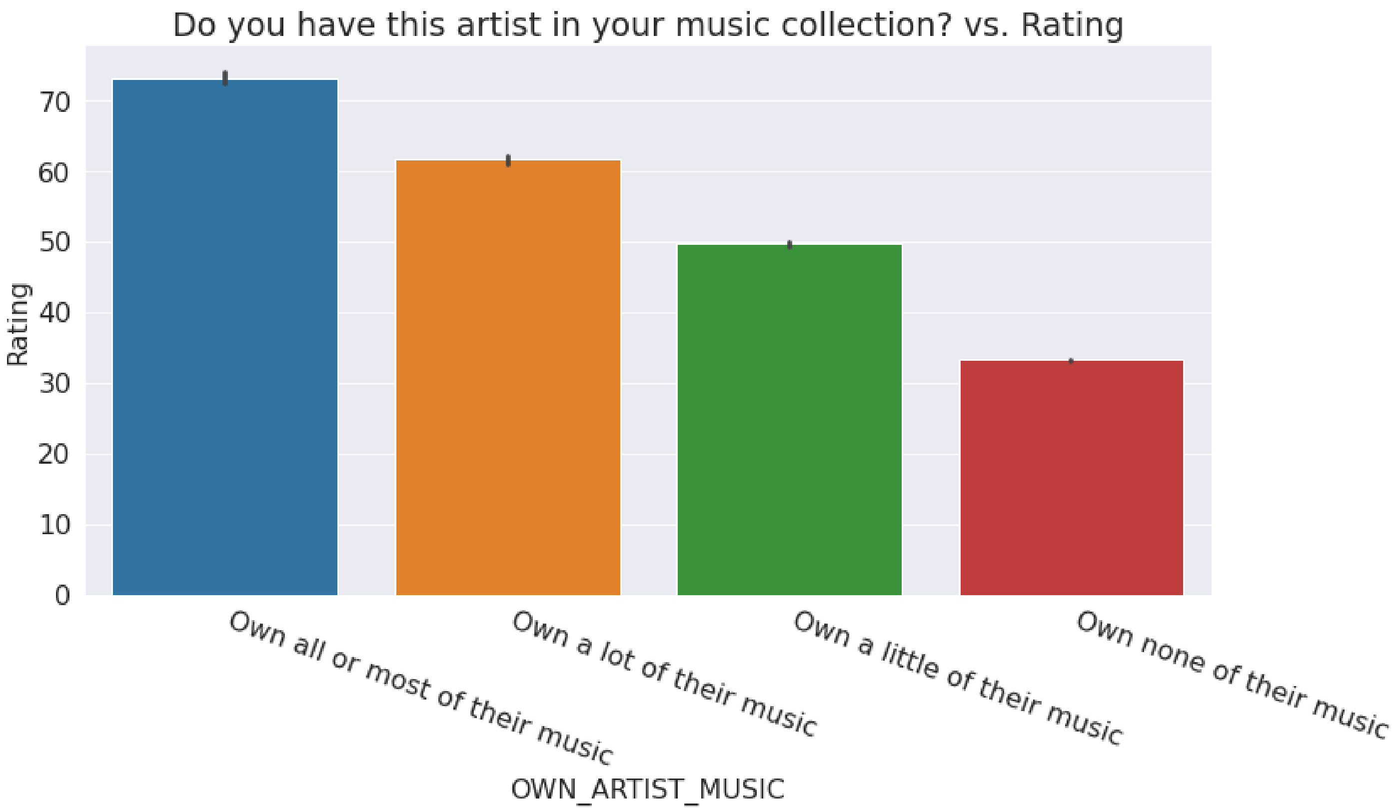
```
array(['Own none of their music', 'Own a little of their music',
       'Own all or most of their music', 'Own a lot of their music'],
      dtype=object)
```

```
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(
```

```
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
```

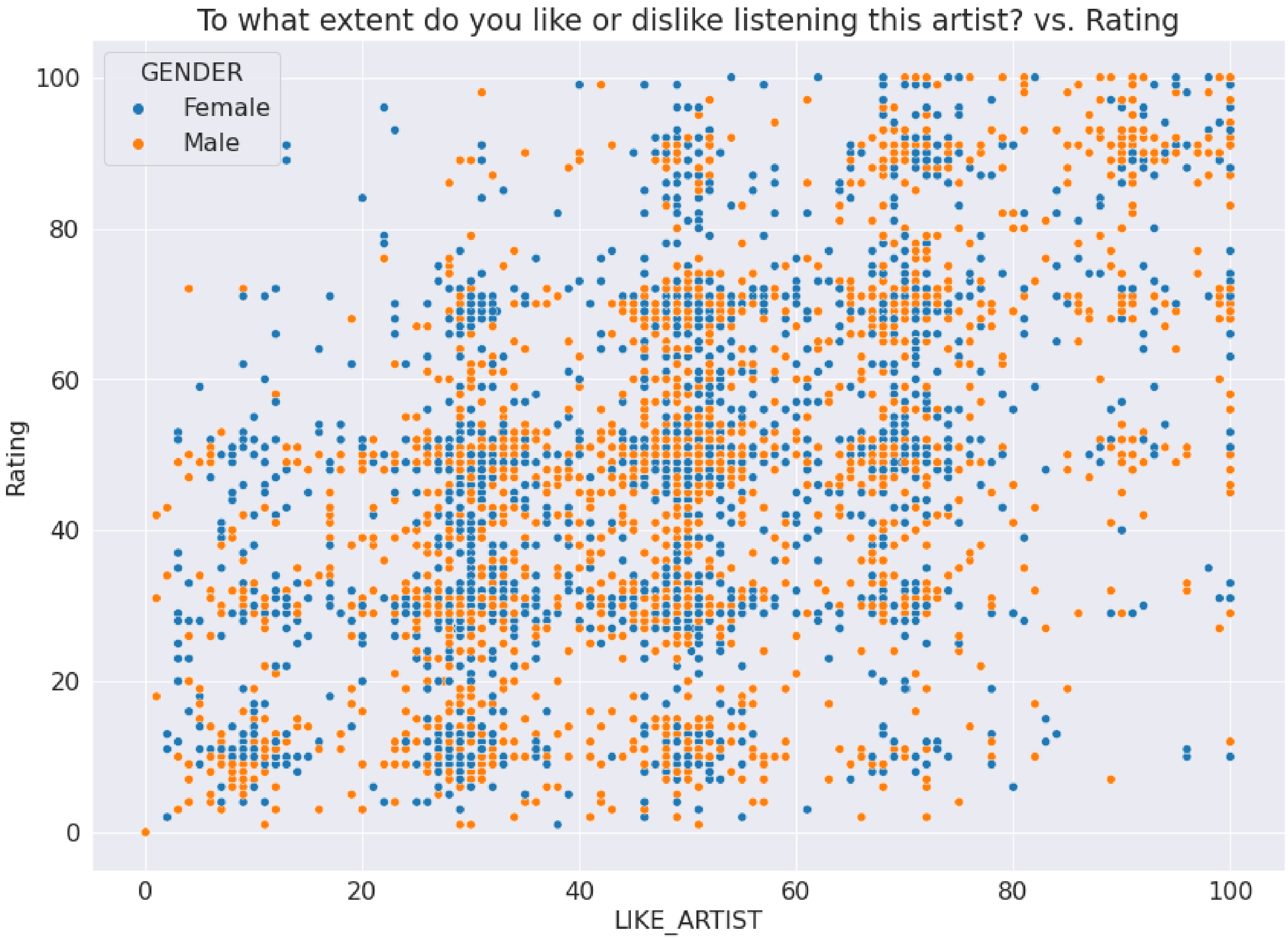
```
training_merge_df
```

	Artist	Track	User	Rating	Time	HEARD_OF	OWN_ARTIST_MUSIC	LIKE_ARTIST	words_score	GENDER	A
0	40	179	47994	9	17	Never heard of	Own none of their music	NaN	-2.0	Female	4

	Artist	Track	User	Rating	Time	HEARD_OF	OWN_ARTIST_MUSIC	LIKE_ARTIST	words_score	GENDER	A
1	9	23	8575	58	7	Never heard of	Own none of their music	NaN	5.0	Female	4
2	46	168	45475	13	16	Never heard of	Own none of their music	NaN	1.0	Male	2
3	11	153	39508	42	15	Heard of and listened to music EVER	Own none of their music	28.0	4.0	Female	6
4	14	32	11565	54	19	Heard of and listened to music EVER	Own none of their music	18.0	2.0	Female	2
...	...	...	...	...	...	...	...	...	...	...	
188685	0	3	1278	29	6	Never heard of	Own none of their music	NaN	3.0	Female	5
188686	1	6	2839	30	18	Heard of	Own none of their music	NaN	-1.0	Male	5
188687	10	142	35756	61	12	Heard of	Own none of their music	NaN	3.0	Female	2
188688	22	54	20163	46	21	Heard of and listened to music RECENTLY	Own a lot of their music	74.0	10.0	Female	3
188689	47	171	45580	12	4	Heard of and listened to music RECENTLY	Own none of their music	7.0	1.0	Female	8

188690 rows × 35 columns

```
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.sampl
```




```
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()
```

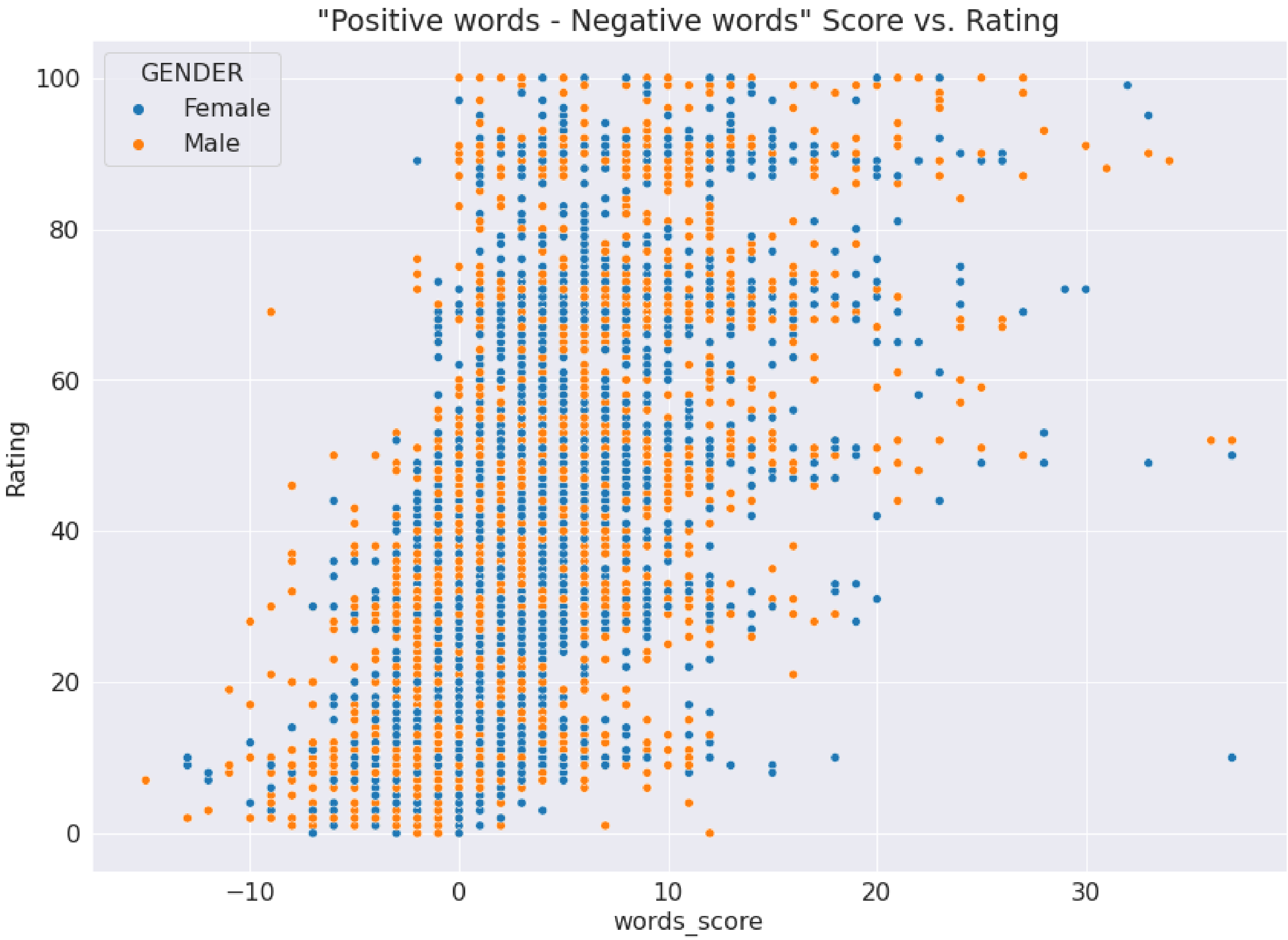
```
count    188690.000000
mean         36.435391
std          22.586036
min           0.000000
25%          15.000000
50%          32.000000
75%          50.000000
max         100.000000
Name: Rating, dtype: float64
```

```
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

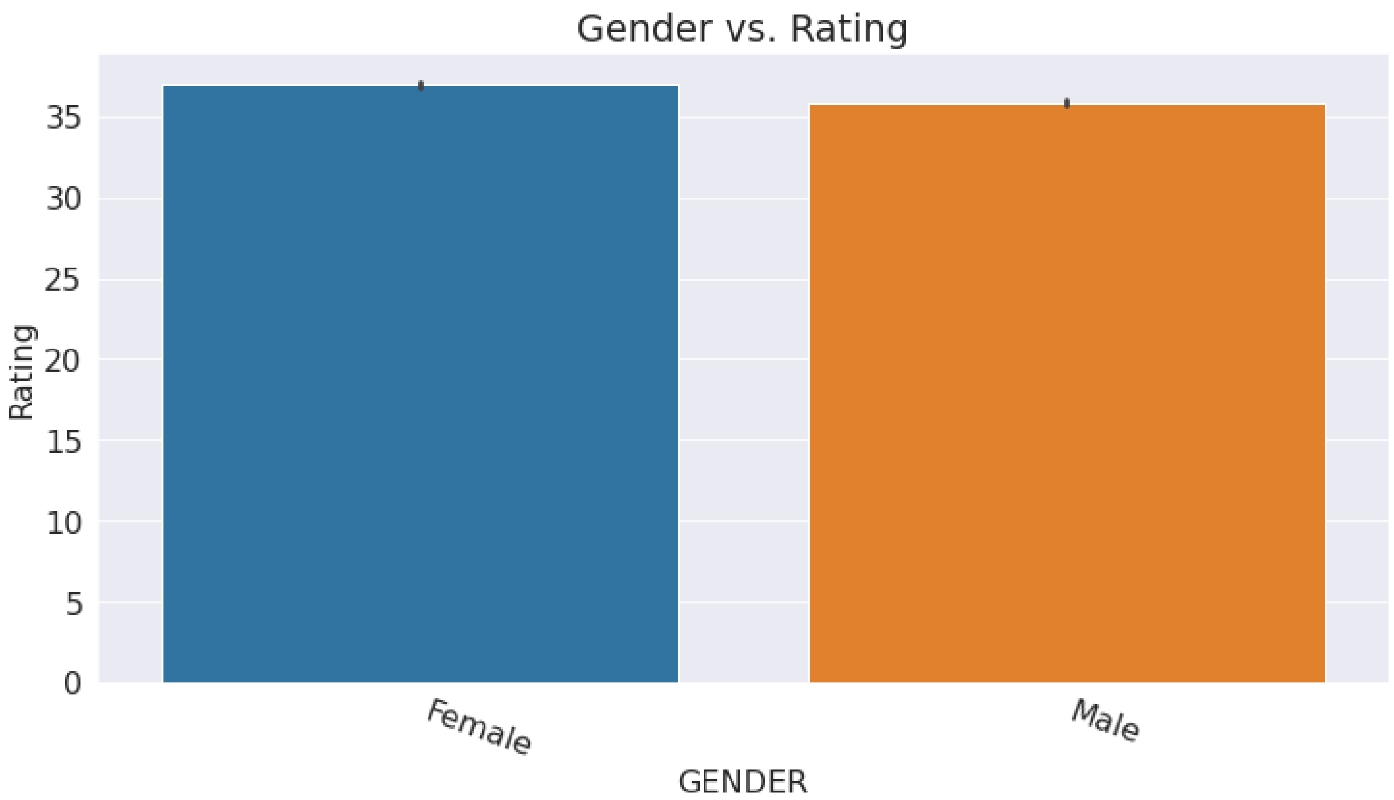
```
plt.title('"Positive words - Negative words" Score vs. Rating')
sns.scatterplot(x='words_score', y='Rating', hue='GENDER', data=training_merge_df.sampl
```



GENDER

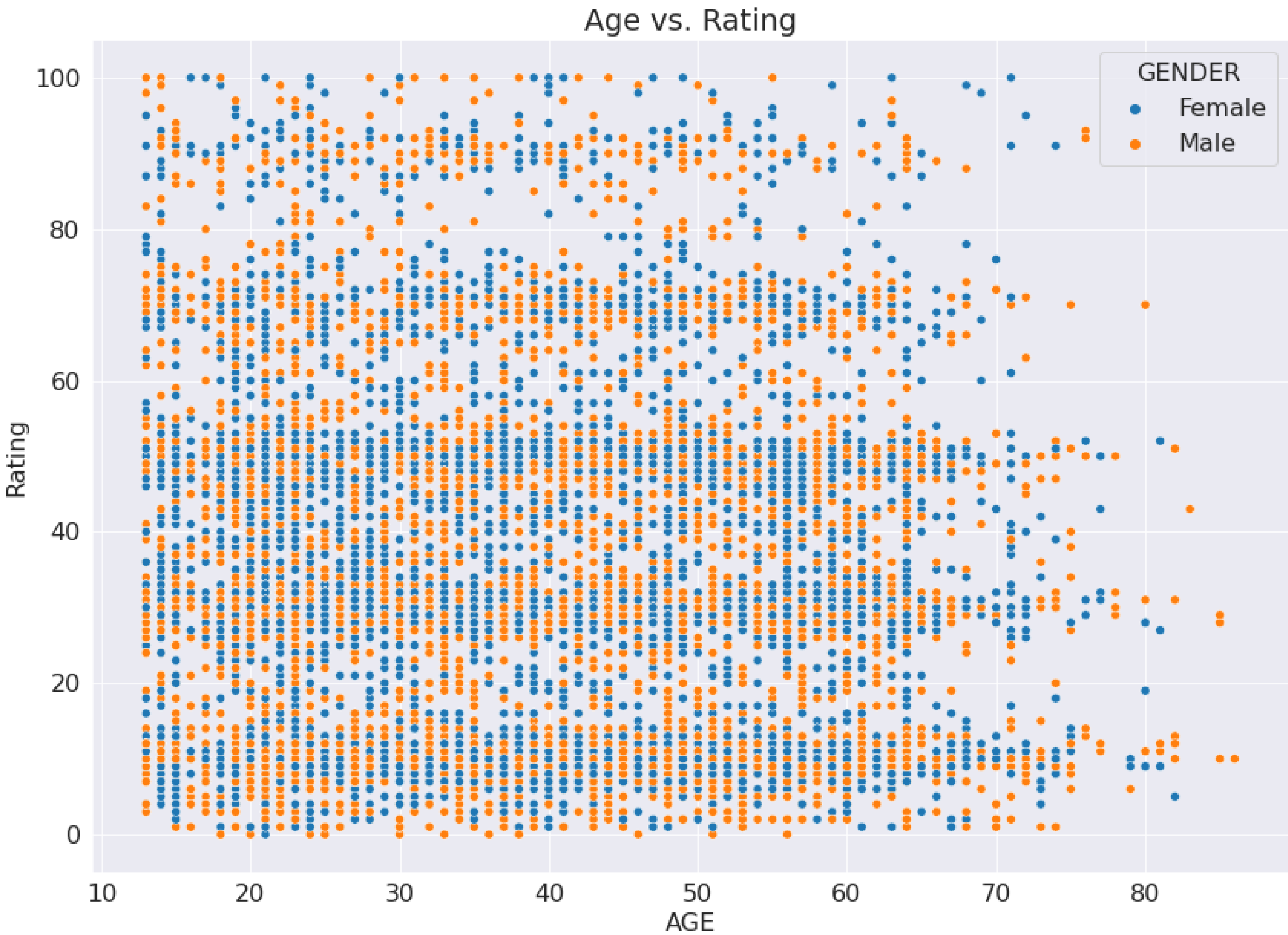
```
fig, ax = plt.subplots(figsize=(12,6))

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)
```



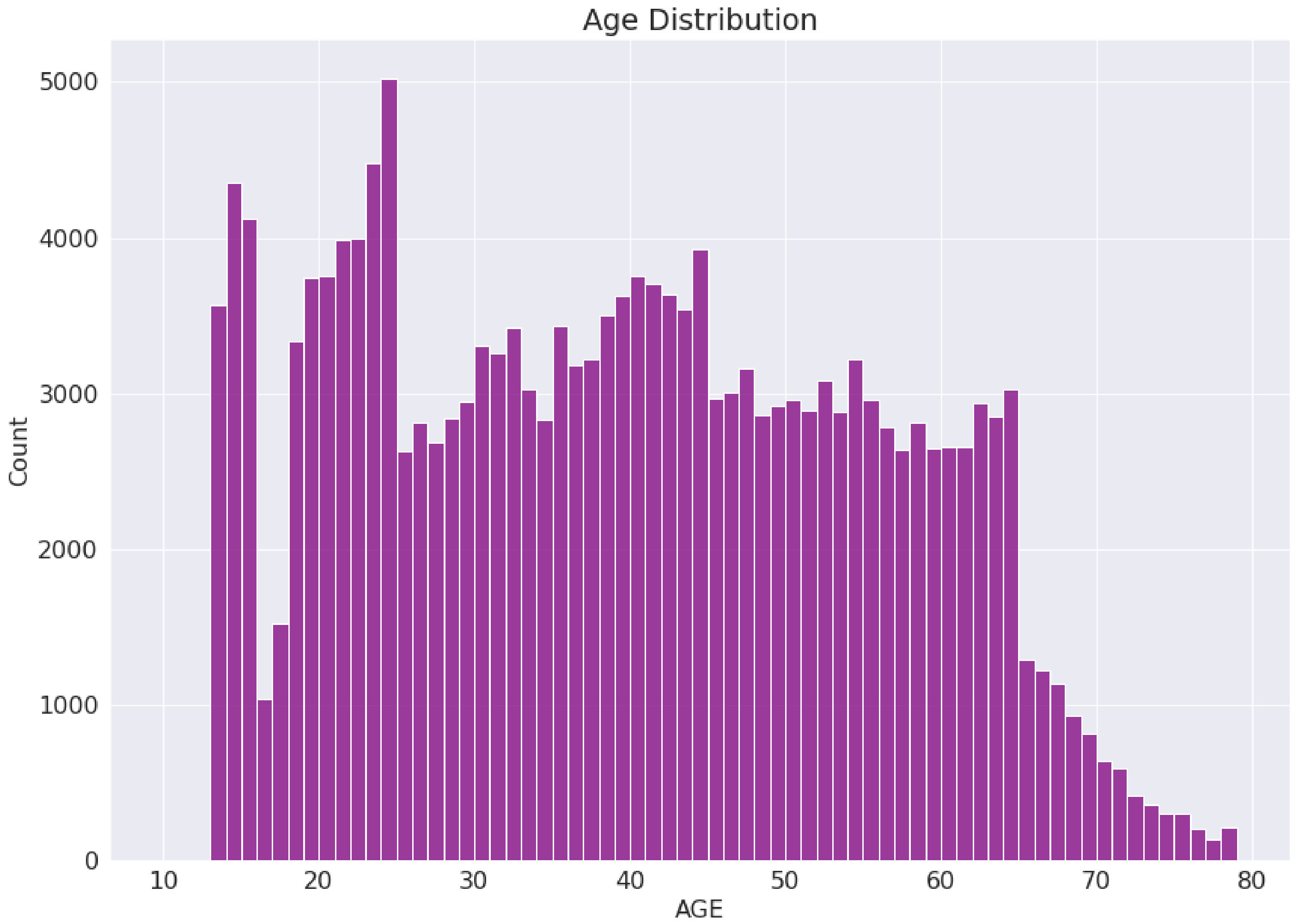

```
training_merge_df['AGE'].describe()
```

```
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'].isn
```

```
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_catego
```

```
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
```

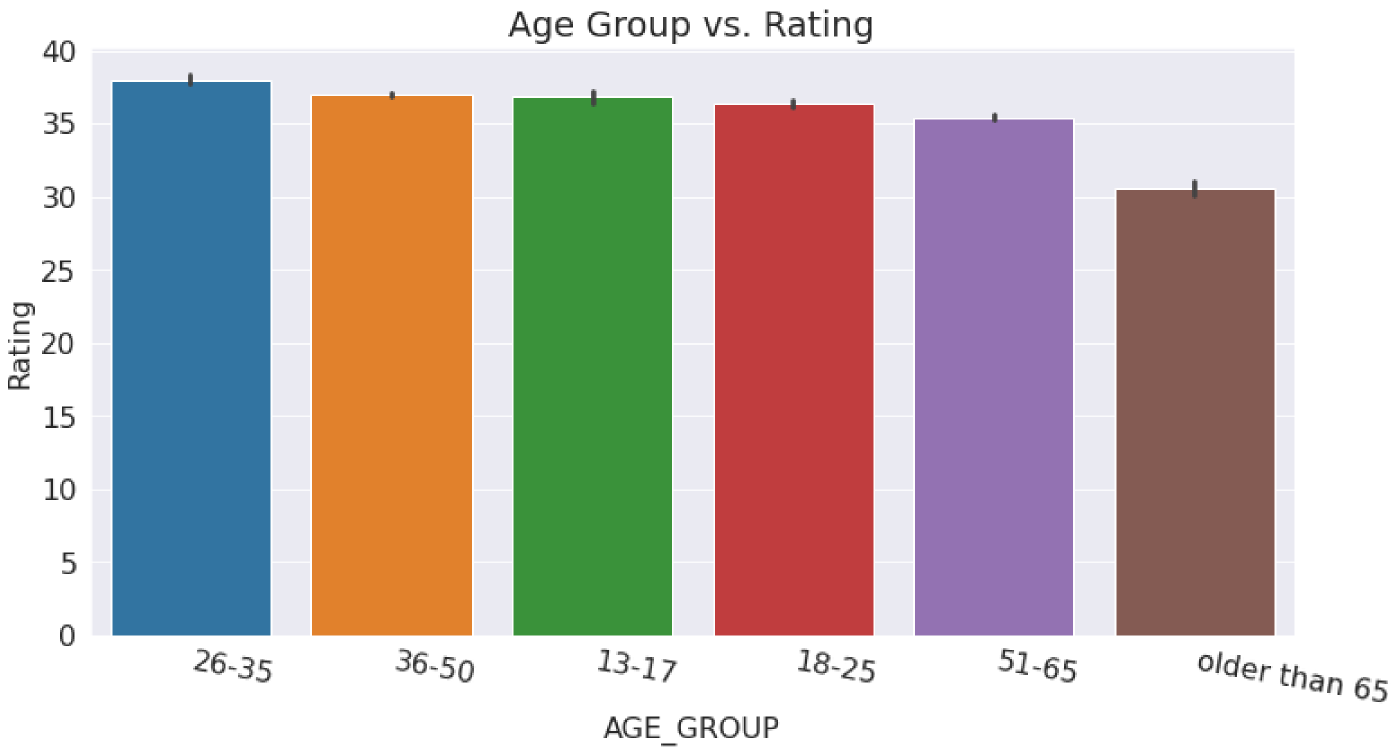
```
training_merge_df['AGE_GROUP'].fillna('36-50', inplace=True)
training_merge_df['AGE'].fillna(39, inplace=True)
```

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(ascendi
```

```
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();
```



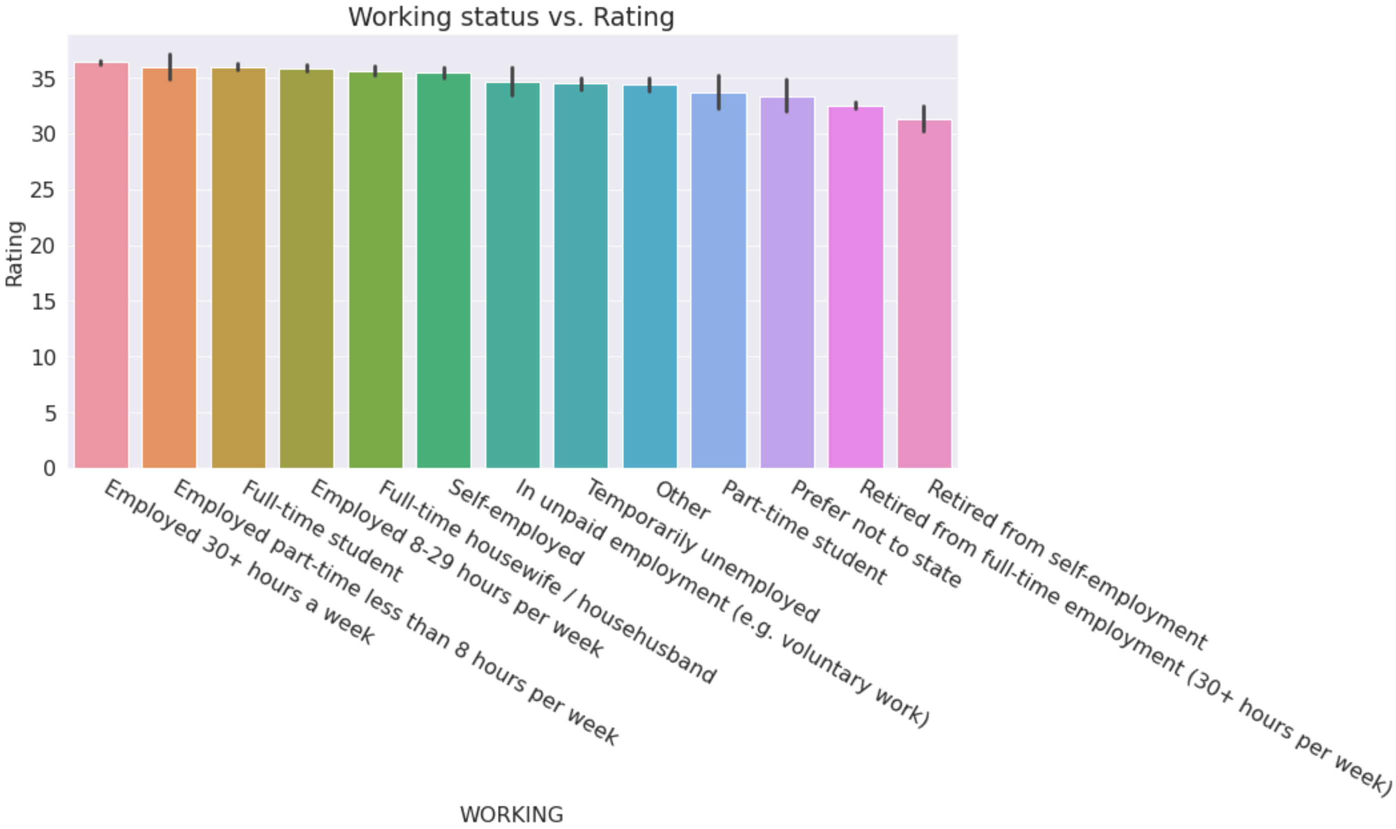
Working

```
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
```

```
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)
```

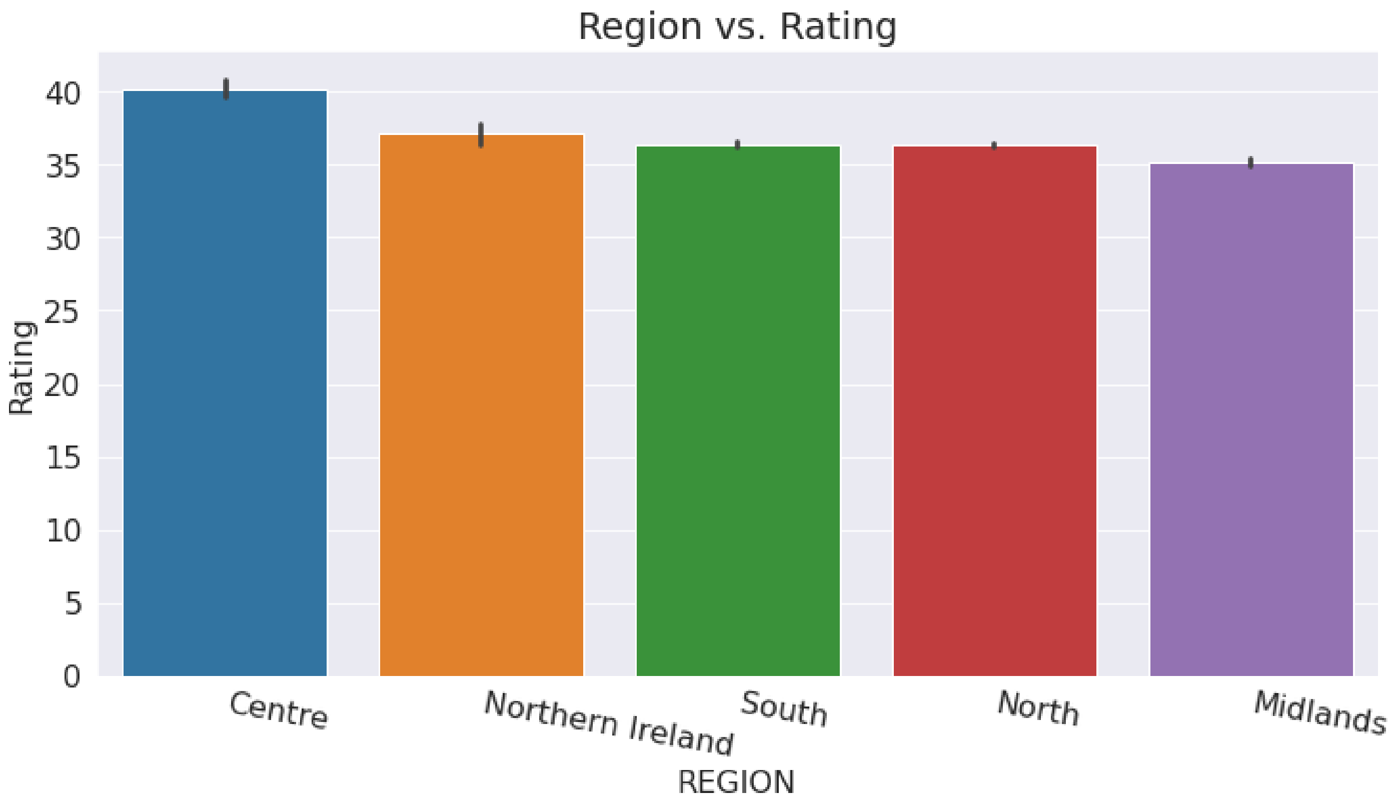
```
training_merge_df['REGION'].value_counts()
```

```
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=
```

```
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
```

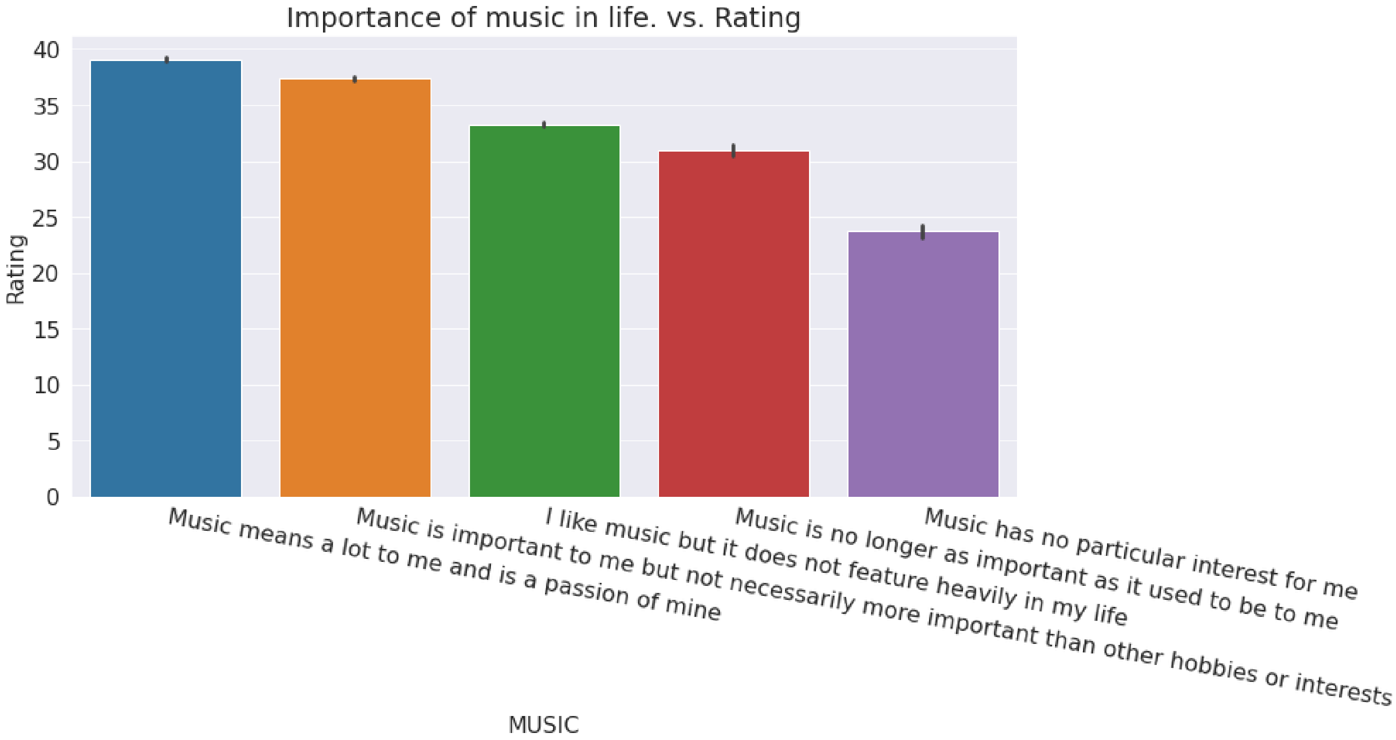
```
training_merge_df['MUSIC'].replace(['Music is important to me but not necessarily more
test_merge_df['MUSIC'].replace(['Music is important to me but not necessarily more impo
```

```
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=F
```

```
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)
```

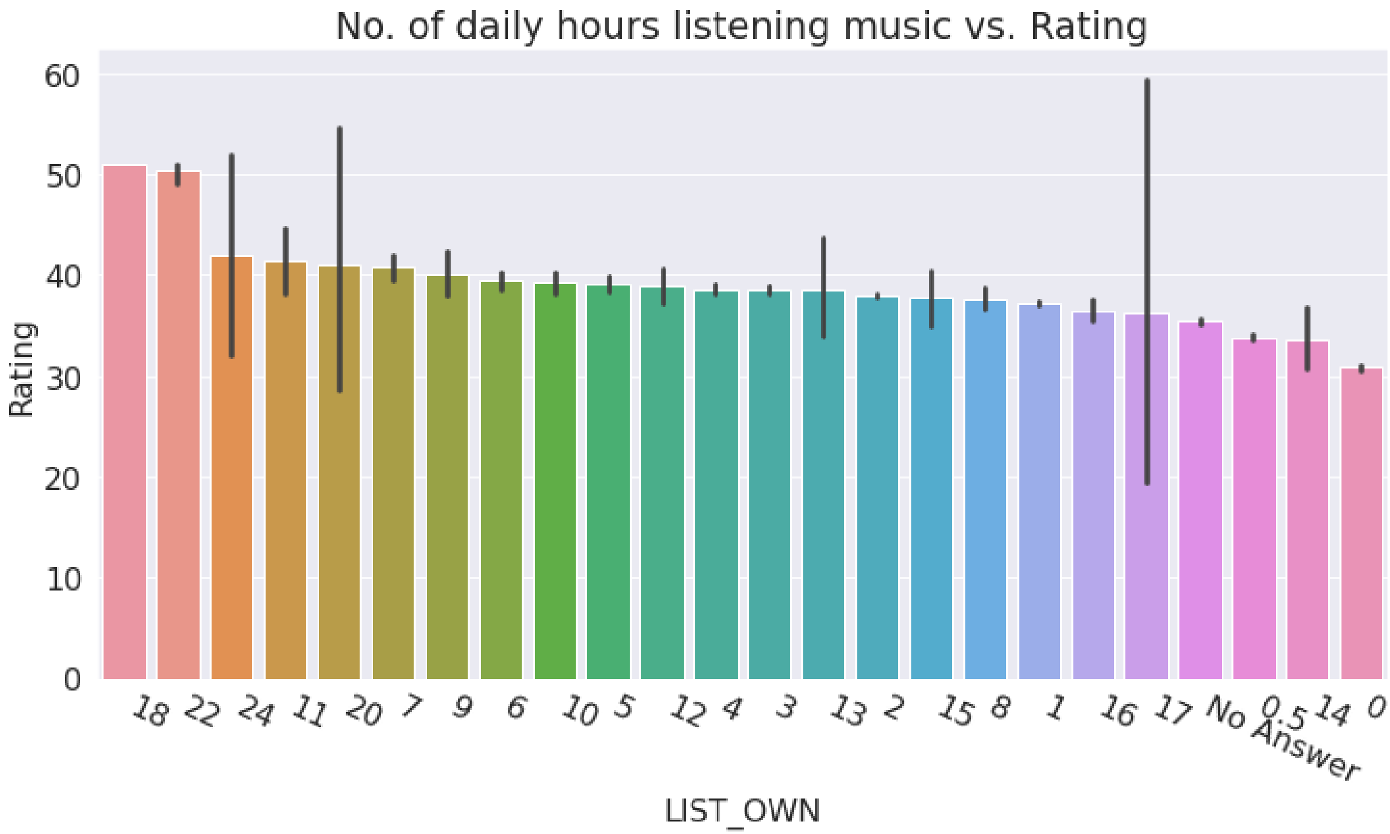
```
training_merge_df['LIST_OWN'].fillna('No Answer', inplace=True)
```

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(ascendin
```

```
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'
          }
```

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

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

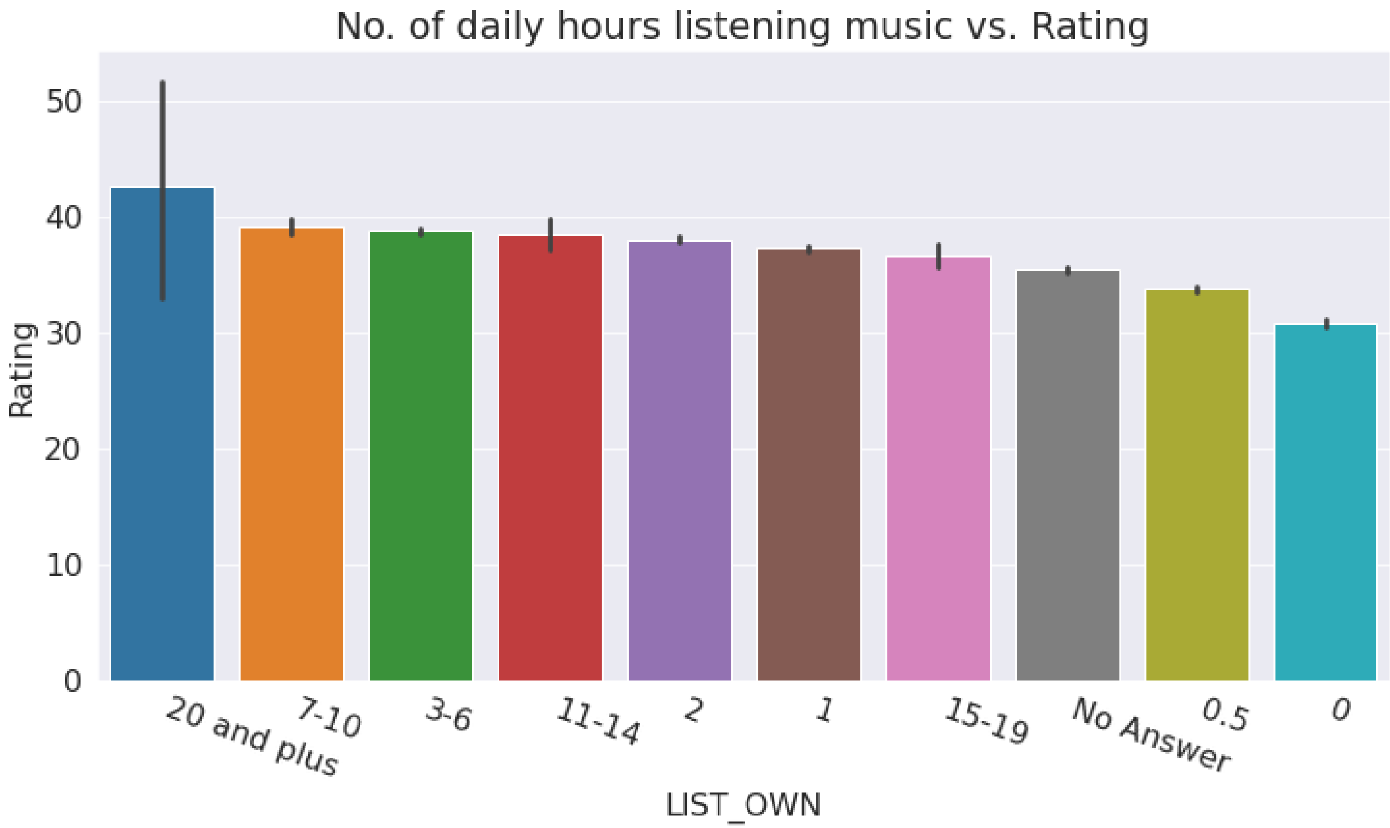
```
array(['3-6', '1', '0.5', '0', 'No Answer', '2', '7-10', '15-19', '11-14',
       '20 and plus'], dtype=object)
```

```python
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
```

```python
plot_order= training_merge_df.groupby('LIST_OWN')['Rating'].mean().sort_values(ascendin
```

```python
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)
```

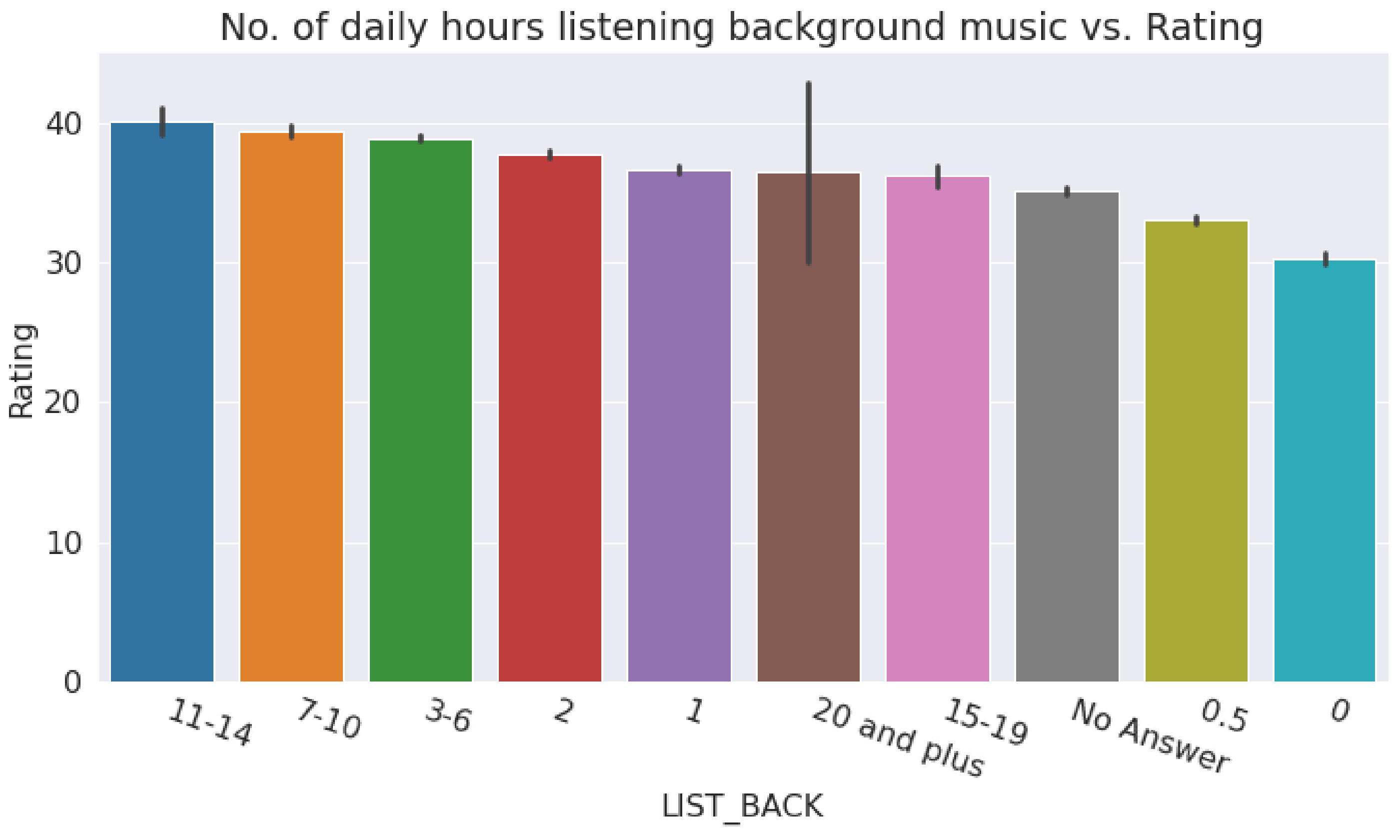
```
training_merge_df['LIST_BACK'].replace(['16+ hours'], '16', inplace=True)
training_merge_df['LIST_BACK'].replace(['More than 16 hours'], '16', inplace=True)
```

```
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(ascendi
```

```
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();
```

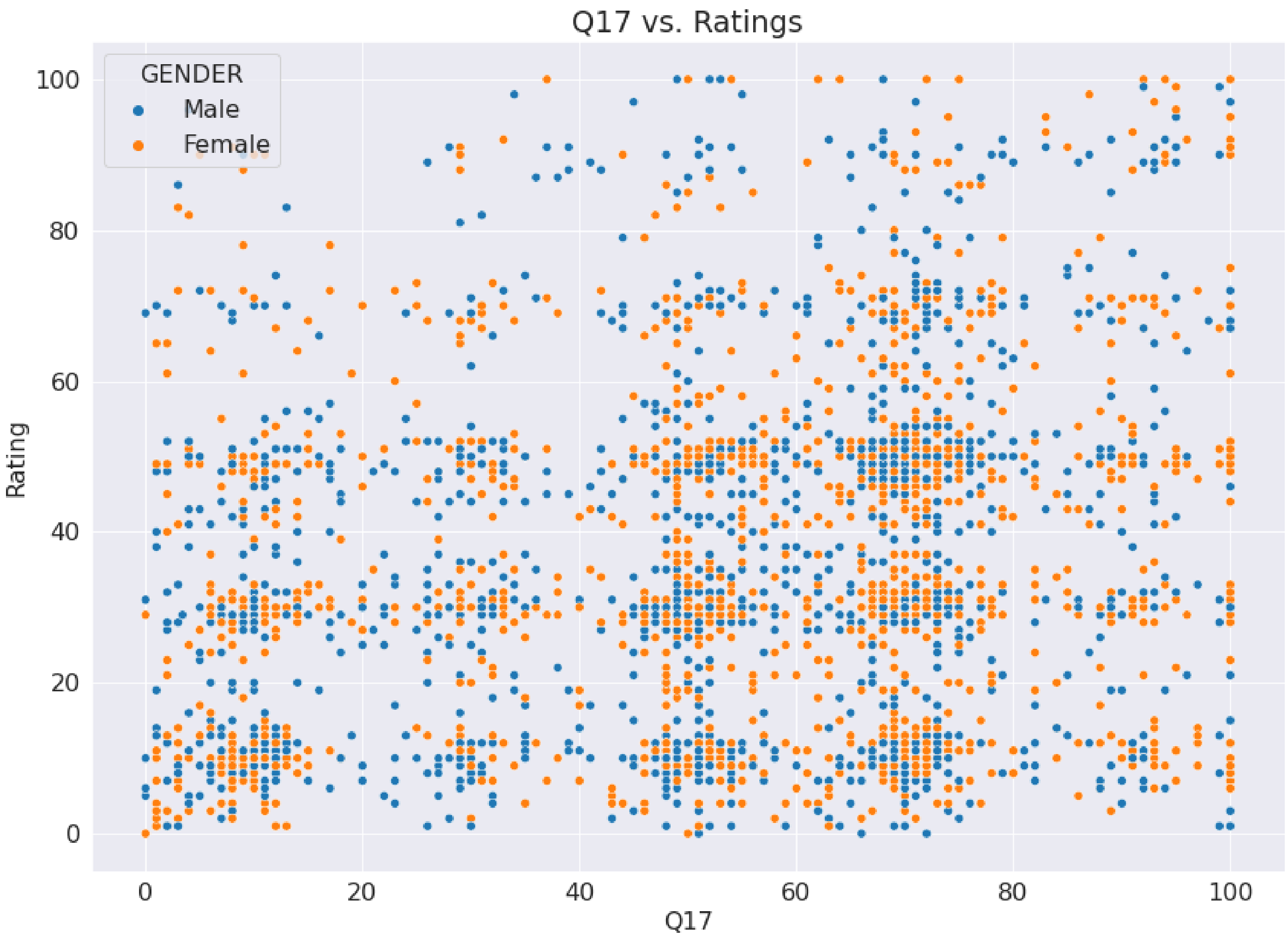


Test DataFrame

```
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_m
```

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_

test_merge_df['LIKE_ARTIST'] = test_merge_df['LIKE_ARTIST'].apply(lambda x: to_categori
```

```
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 hobbi
          '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	words_score	GENDER	A
94459	26	66	23839	33	22	1	1	No Answer	6.0	Male	5
107256	39	105	30043	28	23	2	1	No Answer	0.0	Male	5
70068	38	102	28429	11	23	1	1	No Answer	-6.0	Male	4
65626	40	178	47357	10	17	1	1	No Answer	3.0	Male	6
178573	4	12	36730	87	13	2	1	No Answer	8.0	Female	2
...	...	...	...	...	...	...	...	...	...	...	
132836	49	182	50591	42	17	1	1	No Answer	7.0	Male	6

	Artist	Track	User	Rating	Time	HEARD_OF	OWN_ARTIST_MUSIC	LIKE_ARTIST	words_score	GENDER	A
63620	21	50	19101	68	21	1	1	No Answer	8.0	Female	2
99278	26	62	22913	68	22	1	1	No Answer	4.0	No Answer	3
97059	8	20	9151	31	7	2	1	No Answer	9.0	Male	4
76601	2	175	48309	31	17	1	1	No Answer	2.0	Female	5

150952 rows × 36 columns

```
validation_df
```

	Artist	Track	User	Rating	Time	HEARD_OF	OWN_ARTIST_MUSIC	LIKE_ARTIST	words_score	GENDER	A
181143	37	101	30087	10	23	3	1	41-50	-6.0	Female	2
90132	2	68	22370	8	22	1	1	No Answer	-4.0	Female	6
106194	42	157	41769	12	16	2	1	No Answer	0.0	Female	2
145815	4	11	36416	52	13	3	2	51-60	2.0	Female	3
3393	33	85	26003	26	11	4	2	51-60	1.0	Male	3
...	...	...	...	...	...	...	...	...	...	...	
52303	26	65	22243	10	22	1	1	No Answer	-2.0	Female	4
152525	34	86	29201	10	23	1	1	No Answer	-8.0	Male	5
46901	22	126	32627	72	0	3	2	71-80	8.0	Female	4
109675	40	178	50478	31	17	1	1	No Answer	-2.0	Female	2
49200	22	132	32204	14	0	3	1	1-10	0.0	Male	5

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	GENDER	WORKII
94459	26	66	23839	22	1	1	No Answer	6.0	Male	Employ 30+ hour: we
107256	39	105	30043	23	2	1	No Answer	0.0	Male	Employ 30+ hour: we
70068	38	102	28429	23	1	1	No Answer	-6.0	Male	Temporal unemploy
65626	40	178	47357	17	1	1	No Answer	3.0	Male	Retired frc full-tir employme (30+ hou
178573	4	12	36730	13	2	1	No Answer	8.0	Female	Temporal unemploy
...	...	...	...	...	...	...	...	...	...	
132836	49	182	50591	17	1	1	No Answer	7.0	Male	Retired frc s employme
63620	21	50	19101	21	1	1	No Answer	8.0	Female	Employ 30+ hour: we
99278	26	62	22913	22	1	1	No Answer	4.0	No Answer	No Answv
97059	8	20	9151	7	2	1	No Answer	9.0	Male	No Answv
76601	2	175	48309	17	1	1	No Answer	2.0	Female	Employ 30+ hour: we

150952 rows × 34 columns

```
validation_inputs
```

	Artist	Track	User	Time	HEARD_OF	OWN_ARTIST_MUSIC	LIKE_ARTIST	words_score	GENDER	WOR
181143	37	101	30087	23	3	1	41-50	-6.0	Female	Tempo unempl
90132	2	68	22370	22	1	1	No Answer	-4.0	Female	Retired full employ (30+ hour

	Artist	Track	User	Time	HEARD_OF	OWN_ARTIST_MUSIC	LIKE_ARTIST	words_score	GENDER	WOR
106194	42	157	41769	16	2	1	No Answer	0.0	Female	Full stu
145815	4	11	36416	13	3	2	51-60	2.0	Female	Employed hours per
3393	33	85	26003	11	4	2	51-60	1.0	Male	Employec hours a
...	...	...	...	...	...	...	...	...	...	
52303	26	65	22243	22	1	1	No Answer	-2.0	Female	Full housev househus
152525	34	86	29201	23	1	1	No Answer	-8.0	Male	Employec hours a
46901	22	126	32627	0	3	2	71-80	8.0	Female	No Ar
109675	40	178	50478	17	1	1	No Answer	-2.0	Female	Employed hours per
49200	22	132	32204	0	3	1	1-10	0.0	Male	No Ar

37738 rows × 34 columns

```
test_inputs
```

	Artist	Track	User	Time	HEARD_OF	OWN_ARTIST_MUSIC	LIKE_ARTIST	words_score	GENDER	WOR
0	1	6	3475	18	3	1	1-10	2.0	Female	Employec hours a
1	6	149	39210	15	1	1	No Answer	NaN	Male	Employec hours a
2	40	177	47861	17	1	1	No Answer	-2.0	Female	(
3	31	79	27413	11	1	1	No Answer	0.0	Female	Empl part-time than 8 h per
4	26	66	23232	22	1	1	No Answer	0.0	No Answer	No Ar
...	...	...	...	...	...	...	...	...	...	
125789	14	95	30004	23	2	1	No Answer	12.0	Male	Employec hours a
125790	10	25	8186	7	1	1	No Answer	6.0	Male	No Ar
125791	40	146	38180	13	2	1	No Answer	3.0	Female	Full housev househus
125792	22	113	32918	0	3	1	41-50	2.0	Female	No Ar
125793	2	70	24231	22	1	1	No Answer	4.0	Male	Employec hours a

125794 rows × 34 columns

Segregation of Numeric and Catego... Cols

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

categorical_cols = ['LIKE_ARTIST', 'GENDER', 'WORKING', 'REGION', 'LIST_OWN', 'LIST_BAC
```

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

	Artist	Track	User	Time	HEARD_OF	OWN_ARTIST_MUSIC	w
count	150952.000000	150952.000000	150952.000000	150952.000000	150952.000000	150952.000000	1493
mean	22.186914	86.589671	26495.799652	15.647636	1.878928	1.218261	
std	14.483465	56.014475	13631.312898	6.442953	1.056842	0.575563	
min	0.000000	0.000000	0.000000	0.000000	1.000000	1.000000	·
25%	10.000000	36.000000	17723.750000	12.000000	1.000000	1.000000	
50%	22.000000	81.000000	27873.500000	17.000000	1.000000	1.000000	
75%	35.000000	142.000000	35952.250000	21.000000	3.000000	1.000000	
max	49.000000	183.000000	50927.000000	23.000000	4.000000	4.000000	

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 150952 entries, 94459 to 76601
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        149321 non-null  float64
 7   MUSIC              141491 non-null  float64
 8   Q1                 141491 non-null  float64
 9   Q2                 141491 non-null  float64
 10  Q3                 141491 non-null  float64
 11  Q4                 141491 non-null  float64
 12  Q5                 141491 non-null  float64
 13  Q6                 141491 non-null  float64
 14  Q7                 141491 non-null  float64
 15  Q8                 141491 non-null  float64
 16  Q9                 141491 non-null  float64
 17  Q10                141491 non-null  float64
```

```
 18  Q11                141491 non-null  float64
 19  Q12                141491 non-null  float64
 20  Q13                141491 non-null  float64
 21  Q14                141491 non-null  float64
 22  Q15                141491 non-null  float64
 23  Q16                114250 non-null  float64
 24  Q17                141491 non-null  float64
 25  Q18                112419 non-null  float64
 26  Q19                112419 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_ARTIST_MUSIC	w
count	150952.000000	150952.000000	150952.000000	150952.000000	150952.000000	150952.000000	1509
mean	0.452794	0.473168	0.520270	0.680332	0.292976	0.072754	
std	0.295581	0.306090	0.267664	0.280128	0.352281	0.191854	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.204082	0.196721	0.348023	0.521739	0.000000	0.000000	
50%	0.448980	0.442623	0.547323	0.739130	0.000000	0.000000	
75%	0.714286	0.775956	0.705957	0.913043	0.666667	0.000000	
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 150952 entries, 94459 to 76601
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
```

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

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

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

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

```python
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
will be removed in 1.2. Please use get_feature_names_out instead.
```

```python
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])
```

```python
training_inputs
```

	Artist	Track	User	Time	HEARD_OF	OWN_ARTIST_MUSIC	LIKE_ARTIST	words_score	GEN
94459	0.530612	0.360656	0.468101	0.956522	0.000000	0.0	No Answer	0.400000	I
107256	0.795918	0.573770	0.589923	1.000000	0.333333	0.0	No Answer	0.290909	I
70068	0.775510	0.557377	0.558230	1.000000	0.000000	0.0	No Answer	0.181818	I
65626	0.816327	0.972678	0.929900	0.739130	0.000000	0.0	No Answer	0.345455	I
178573	0.081633	0.065574	0.721228	0.565217	0.333333	0.0	No Answer	0.436364	Fer
...	...	...	...	...	...	...	...	...	
132836	1.000000	0.994536	0.993402	0.739130	0.000000	0.0	No Answer	0.418182	I

	Artist	Track	User	Time	HEARD_OF	OWN_ARTIST_MUSIC	LIKE_ARTIST	words_score	GEN
63620	0.428571	0.273224	0.375066	0.913043	0.000000	0.0	No Answer	0.436364	Fer
99278	0.530612	0.338798	0.449919	0.956522	0.000000	0.0	No Answer	0.363636	Ans
97059	0.163265	0.109290	0.179689	0.304348	0.333333	0.0	No Answer	0.454545	I
76601	0.040816	0.956284	0.948593	0.739130	0.000000	0.0	No Answer	0.327273	Fer

150952 rows × 94 columns

```
# 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

```
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)
```

```
[05:27:44] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.
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)
```

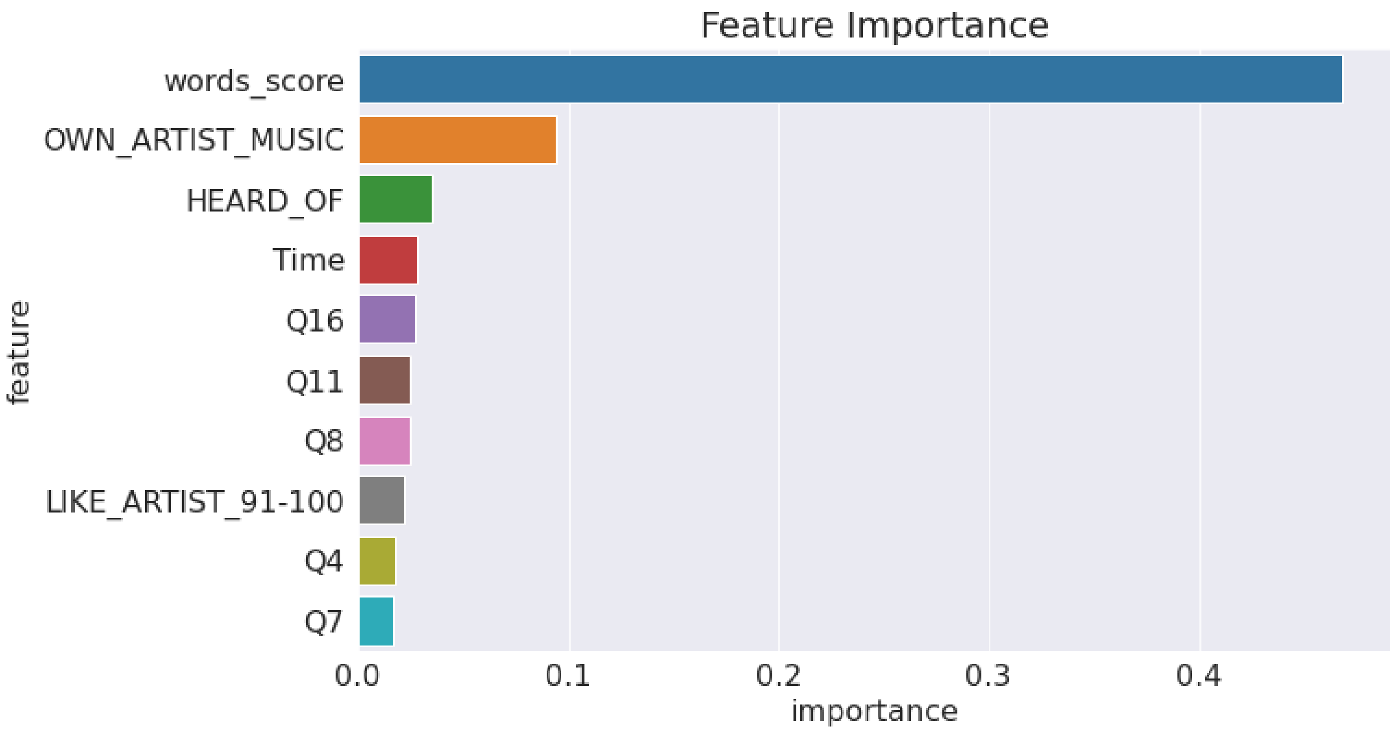
```
15.95594644858887
```

```
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	0.467643
5	OWN_ARTIST_MUSIC	0.094419
4	HEARD_OF	0.035286
3	Time	0.027736
23	Q16	0.027494
18	Q11	0.024733
15	Q8	0.024590
36	LIKE_ARTIST_91-100	0.022184
11	Q4	0.017584
14	Q7	0.016818

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



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)
```

```
[05:40:03] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now
deprecated in favor of reg:squarederror.
Training RMSE: 15.95594644858887, Validation RMSE: 16.012146067743867
```

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

```
[05:38:08] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now
deprecated in favor of reg:squarederror.
Training RMSE: 15.608608855342386, Validation RMSE: 15.739599633365968
```

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

```
[05:40:39] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now
deprecated in favor of reg:squarederror.
Training RMSE: 15.4972850265345, Validation RMSE: 15.663096963239324
```

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

```
[05:43:22] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now
deprecated in favor of reg:squarederror.
Training RMSE: 15.164389073947351, Validation RMSE: 15.462918969733828
```

Tree depth & Learning rate

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

```
[05:48:03] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now
deprecated in favor of reg:squarederror.
Training RMSE: 10.761732362133515, Validation RMSE: 14.483297295263416
```

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

```
[05:51:01] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now
deprecated in favor of reg:squarederror.
Training RMSE: 11.735802155463258, Validation RMSE: 14.475093637694114
```

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

```
[06:11:12] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now
deprecated in favor of reg:squarederror.
Training RMSE: 21.67496231755449, Validation RMSE: 21.733228417983653
```

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

[06:12:32] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.
Training RMSE: 8.142226331280742, Validation RMSE: 14.07542497669014

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

[06:23:06] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.
Training RMSE: 5.822519262858865, Validation RMSE: 14.082742854087666

```
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
  models.append(model)
  print('Train RMSE: {}, Validation RMSE: {}'.format(train_rmse, val_rmse))
```

[06:56:15] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.
Train RMSE: 8.87857799015854, Validation RMSE: 14.232260924487084
[07:03:08] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.
Train RMSE: 8.921200180590178, Validation RMSE: 14.264136056361622
[07:09:28] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.
Train RMSE: 8.870303235221652, Validation RMSE: 14.350492495024087
[07:16:00] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.
Train RMSE: 8.86326892581729, Validation RMSE: 14.419535640174853
[07:22:26] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now

```
deprecated in favor of reg:squarederror.
Train RMSE: 8.877630772144759, Validation RMSE: 14.329955985260652
```

```
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)
```

```
13.89967225314936
```

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

Final Answer

```
test_preds.shape
```

```
(125794,)
```

```

```