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

In [2]:
```
#import and read data
df=pd.read_excel('1673873388_rolling_stones_spotify.xlsx')
df.head()
```

Out[2]:

	Unnamed: 0	name	album	release_date	track_number	id	
0	0	Concert Intro Music - Live	Licked Live In NYC	2022-06-10	1	2lEkywLJ4ykbhi1yRQvmsT	s
1	1	Street Fighting Man - Live	Licked Live In NYC	2022-06-10	2	6GVgVJBKkGJoRfarYRvGTU	sp
2	2	Start Me Up - Live	Licked Live In NYC	2022-06-10	3	1Lu761pZ0dBTGpzxaQoZNW	spo
3	3	If You Can't Rock Me - Live	Licked Live In NYC	2022-06-10	4	1agTQzOTUnGNggyckEqiDH	sp
4	4	Donâ€™t Stop - Live	Licked Live In NYC	2022-06-10	5	7piGJR8YndQBQWVXv6KtQw	spo

In [3]:
```
#checking null values
pd.isnull(df).sum()
```

Out[3]:
```
Unnamed: 0          0
name                0
album               0
release_date        0
track_number        0
id                  0
uri                 0
acousticness        0
danceability        0
energy              0
instrumentalness    0
liveness            0
loudness            0
speechiness         0
tempo               0
valence             0
popularity          0
duration_ms         0
dtype: int64
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1610 entries, 0 to 1609
Data columns (total 18 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Unnamed: 0        1610 non-null   int64
 1   name              1610 non-null   object
 2   album             1610 non-null   object
 3   release_date      1610 non-null   datetime64[ns]
 4   track_number      1610 non-null   int64
 5   id                1610 non-null   object
 6   uri               1610 non-null   object
 7   acousticness      1610 non-null   float64
 8   danceability      1610 non-null   float64
 9   energy            1610 non-null   float64
 10  instrumentalness  1610 non-null   float64
 11  liveness          1610 non-null   float64
 12  loudness          1610 non-null   float64
 13  speechiness       1610 non-null   float64
 14  tempo             1610 non-null   float64
 15  valence           1610 non-null   float64
 16  popularity        1610 non-null   int64
 17  duration_ms       1610 non-null   int64
dtypes: datetime64[ns](1), float64(9), int64(4), object(4)
memory usage: 226.5+ KB
```

In [6]:
```
df.duplicated()
```

Out[6]:
```
0       False
1       False
2       False
3       False
4       False
        ...
1605    False
1606    False
1607    False
1608    False
1609    False
Length: 1610, dtype: bool
```

In [7]:
```
df[df.duplicated()]
```

Out[7]:

Unnamed: 0	name	album	release_date	track_number	id	uri	acousticness	danceability

In [8]:
```
sort_df=df.sort_values('popularity', ascending=True).head(10)
```

In [9]:
```
sort_df
```

Out[9]:

	Unnamed: 0	name	album	release_date	track_number	
1591	1591	Little By Little - Mono Version	England's Newest Hit Makers	1964-05-30	6	1n3XRfhLcsrpnDpv
587	587	Next Time You See Me - Live	Live At The Checkerboard Lounge	2012-07-09	8	6Y7sXOOb66zFquJ8
1338	1338	All Sold Out	Between The Buttons	1967-01-20	8	73Homv9FozXD85lI
1394	1394	Not Fade Away - Live	Got Live if you want it!	1966-12-10	4	3tdCzIQ5K0jT6Z1F
1396	1396	Fortune Teller - Live	Got Live if you want it!	1966-12-10	6	39UwRqdXwdzOiif›
1397	1397	The Last Time - Live	Got Live if you want it!	1966-12-10	7	3ojj6rJhR7PgYOn7
739	739	Continental Drift - Live	Flashpoint	1991-04-02	1	6fmxr9Wui3uYW7G
1589	1589	Honest I Do	England's Newest Hit Makers	1964-05-30	4	6DhWfyUAX17MrrQ/
1400	1400	I'm Alright - Live	Got Live if you want it!	1966-12-10	10	6G6HUDo8rsJOHgBs
1401	1401	Have You Seen Your Mother, Baby, Standing In T...	Got Live if you want it!	1966-12-10	11	11ALfJyppuVJdqYs

In [10]:
```
leastPopular=(df['popularity']==0).sum()
```

In [11]:
```
leastPopular
```

Out[11]:
```
17
```

In [12]:
```
df.describe().transpose()
```

Out[12]:

	count	mean	std	min	25%	
Unnamed: 0	1610.0	804.500000	464.911282	0.000000	402.250000	8
track_number	1610.0	8.613665	6.560220	1.000000	4.000000	
acousticness	1610.0	0.250475	0.227397	0.000009	0.058350	
danceability	1610.0	0.468860	0.141775	0.104000	0.362250	
energy	1610.0	0.792352	0.179886	0.141000	0.674000	
instrumentalness	1610.0	0.164170	0.276249	0.000000	0.000219	
liveness	1610.0	0.491730	0.349100	0.021900	0.153000	
loudness	1610.0	-6.971615	2.994003	-24.408000	-8.982500	
speechiness	1610.0	0.069512	0.051631	0.023200	0.036500	
tempo	1610.0	126.082033	29.233483	46.525000	107.390750	
valence	1610.0	0.582165	0.231253	0.000000	0.404250	
popularity	1610.0	20.788199	12.426859	0.000000	13.000000	
duration_ms	1610.0	257736.488199	108333.474920	21000.000000	190613.000000	243(

In [23]:
```
popularityValues=df['popularity'].unique()
popularityValues.sort()
popularityValues
#popularityValues=popularityValues.sort_values('popularity')
```

Out[23]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
       34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 48, 49, 50, 51,
       52, 53, 54, 55, 56, 58, 59, 61, 63, 64, 66, 67, 69, 71, 72, 73, 76,
       80])
```

In [25]:
```
mostPopular=df.query('popularity>70', inplace=False).sort_values('popularity
mostPopular[:10]
```

Out[25]:

	Unnamed: 0	name	album	release_date	track_number	
1403	1403	Paint It, Black	Aftermath	1966-04-15	1	63T7DJ1AFDD6Bn8V
862	862	Start Me Up - Remastered 2009	Tattoo You (2009 Re-Mastered)	1981-08-24	1	7HKez549fwJQDzx
1248	1248	Gimme Shelter	Let It Bleed	1969-12-05	1	6H3kDe7CGoWYBabA
1472	1472	(I Can't Get No) Satisfaction - Mono Version	Out Of Our Heads	1965-07-30	7	2PzU4IB8Dr6mxV3I
1257	1257	Sympathy For The Devil - 50th Anniversary Edition	Beggars Banquet (50th Anniversary Edition)	1968-12-06	1	1Ud6moTC0KyXMq1
901	901	Beast Of Burden - Remastered 1994	Some Girls	1978-06-09	9	77oU2rjC5XbjQfNe
1023	1023	Angie	Goats Head Soup (Remastered 2009)	1973-08-31	5	1GcVa4jFySlun4jI

In [26]:
```
df_Cohorts = df
```

In [27]:
```
df_Cohorts["duration"]=df_Cohorts["duration_ms"].apply(lambda x: round(x/100
df_Cohorts.drop("duration_ms", inplace=True, axis =1)
```

In [28]:
```
df_Cohorts
```

Out[28]:

	Unnamed: 0	name	album	release_date	track_number	id
0	0	Concert Intro Music - Live	Licked Live In NYC	2022-06-10	1	2lEkywLJ4ykbhi1yRQvmsT
1	1	Street Fighting Man - Live	Licked Live In NYC	2022-06-10	2	6GVgVJBKkGJoRfarYRvGTU
2	2	Start Me Up - Live	Licked Live In NYC	2022-06-10	3	1Lu761pZ0dBTGpzxaQoZNW
3	3	If You Can't Rock Me - Live	Licked Live In NYC	2022-06-10	4	1agTQzOTUnGNggyckEqiDH
4	4	Donâ€™t Stop - Live	Licked Live In NYC	2022-06-10	5	7piGJR8YndQBQWVXv6KtQw
...	...	...	...	...	...	...
1605	1605	Carol	The Rolling Stones	1964-04-16	8	08l7M5UpRnffGl0FyuRiQZ
1606	1606	Tell Me	The Rolling Stones	1964-04-16	9	3JZllQBsTM6WwoJdzFDLhx
1607	1607	Can I Get A Witness	The Rolling Stones	1964-04-16	10	0t2qvfSBQ3Y08lzRRoVTdb
1608	1608	You Can Make It If You Try	The Rolling Stones	1964-04-16	11	5ivIs5vwSj0RChOIvlY3On
1609	1609	Walking The Dog	The Rolling Stones	1964-04-16	12	43SkTJJ2xleDaeiE4TIM70

1610 rows × 18 columns

In [29]:
```
new_columns = df_Cohorts.columns.values
new_columns[0] = 'Id'
df_Cohorts.columns = new_columns
```

In [30]:
```
df_Cohorts = df_Cohorts.set_index('Id')
```

In [31]:
```
df_Cohorts
```

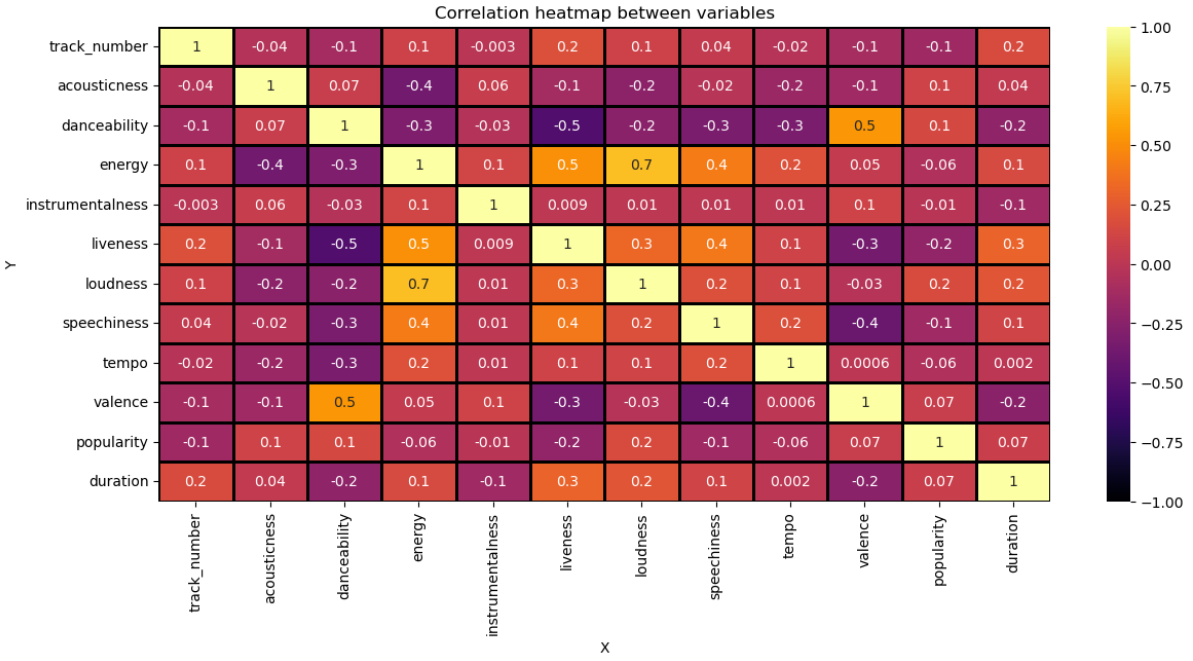
Out[31]:

Id	name	album	release_date	track_number	id	
0	Concert Intro Music - Live	Licked Live In NYC	2022-06-10	1	2IEkywLJ4ykbhi1yRQvmsT	spotify:tr
1	Street Fighting Man - Live	Licked Live In NYC	2022-06-10	2	6GVgVJBKkGJoRfarYRvGTU	spotify:trac
2	Start Me Up - Live	Licked Live In NYC	2022-06-10	3	1Lu761pZ0dBTGpzxaQoZNW	spotify:trac
3	If You Can't Rock Me - Live	Licked Live In NYC	2022-06-10	4	1agTQzOTUnGNggyckEqiDH	spotify:trac
4	Donâ€™t Stop - Live	Licked Live In NYC	2022-06-10	5	7piGJR8YndQBQWVXv6KtQw	spotify:track
...	...	...	...	...	...	
1605	Carol	The Rolling Stones	1964-04-16	8	08l7M5UpRnffGl0FyuRiQZ	spotify:tr
1606	Tell Me	The Rolling Stones	1964-04-16	9	3JZllQBsTM6WwoJdzFDLhx	spotify:trac
1607	Can I Get A Witness	The Rolling Stones	1964-04-16	10	0t2qvfSBQ3Y08lzRRoVTdb	spotify:tra
1608	You Can Make It If You Try	The Rolling Stones	1964-04-16	11	5ivIs5vwSj0RChOIvlY3On	spotify:t
1609	Walking The Dog	The Rolling Stones	1964-04-16	12	43SkTJJ2xleDaeiE4TIM70	spotify:t

1610 rows × 17 columns

In [45]:
```
corr_df=df_Cohorts.corr(method='pearson')
plt.figure(figsize=(14,6))
heatmap=sns.heatmap(corr_df, annot=True, fmt='.1g', vmin=-1, vmax=1, center=
heatmap.set_title('Correlation heatmap between variables')
heatmap.set(xlabel="X", ylabel="Y")
#heatmap.xaxis.tick_top()
#heatmap.set_xticklabels(heatmap.get_xticklables(), rotation=90)
```

Out[45]:
```
[Text(0.5, 36.58159722222221, 'X'), Text(145.58159722222223, 0.5, 'Y')]
```



In [62]:
```
df_Cohorts
```

Out[62]:

Id	name	album	release_date	track_number	id	
0	Concert Intro Music - Live	Licked Live In NYC	2022-06-10	1	2IEkywLJ4ykbhi1yRQvmsT	spotify:tr
1	Street Fighting Man - Live	Licked Live In NYC	2022-06-10	2	6GVgVJBKkGJoRfarYRvGTU	spotify:trac
2	Start Me Up - Live	Licked Live In NYC	2022-06-10	3	1Lu761pZ0dBTGpzxaQoZNW	spotify:trac
3	If You Can't Rock Me - Live	Licked Live In NYC	2022-06-10	4	1agTQzOTUnGNggyckEqiDH	spotify:trac
4	Donâ€™t Stop - Live	Licked Live In NYC	2022-06-10	5	7piGJR8YndQBQWVXv6KtQw	spotify:track
...	...	...	...	...	...	
1605	Carol	The Rolling Stones	1964-04-16	8	08l7M5UpRnffGl0FyuRiQZ	spotify:tr
1606	Tell Me	The Rolling Stones	1964-04-16	9	3JZllQBsTM6WwoJdzFDLhx	spotify:trac
1607	Can I Get A Witness	The Rolling Stones	1964-04-16	10	0t2qvfSBQ3Y08lzRRoVTdb	spotify:tra
1608	You Can Make It If You Try	The Rolling Stones	1964-04-16	11	5ivIs5vwSj0RChOIvlY3On	spotify:t
1609	Walking The Dog	The Rolling Stones	1964-04-16	12	43SkTJJ2xleDaeiE4TIM70	spotify:t

1610 rows × 17 columns

In [56]:
```
new_df=df
```

In [58]:
```
new_columns = new_df.columns.values
new_columns[0] = 'Id'
new_df.columns = new_columns
new_df = new_df.set_index('Id')
new_df
```

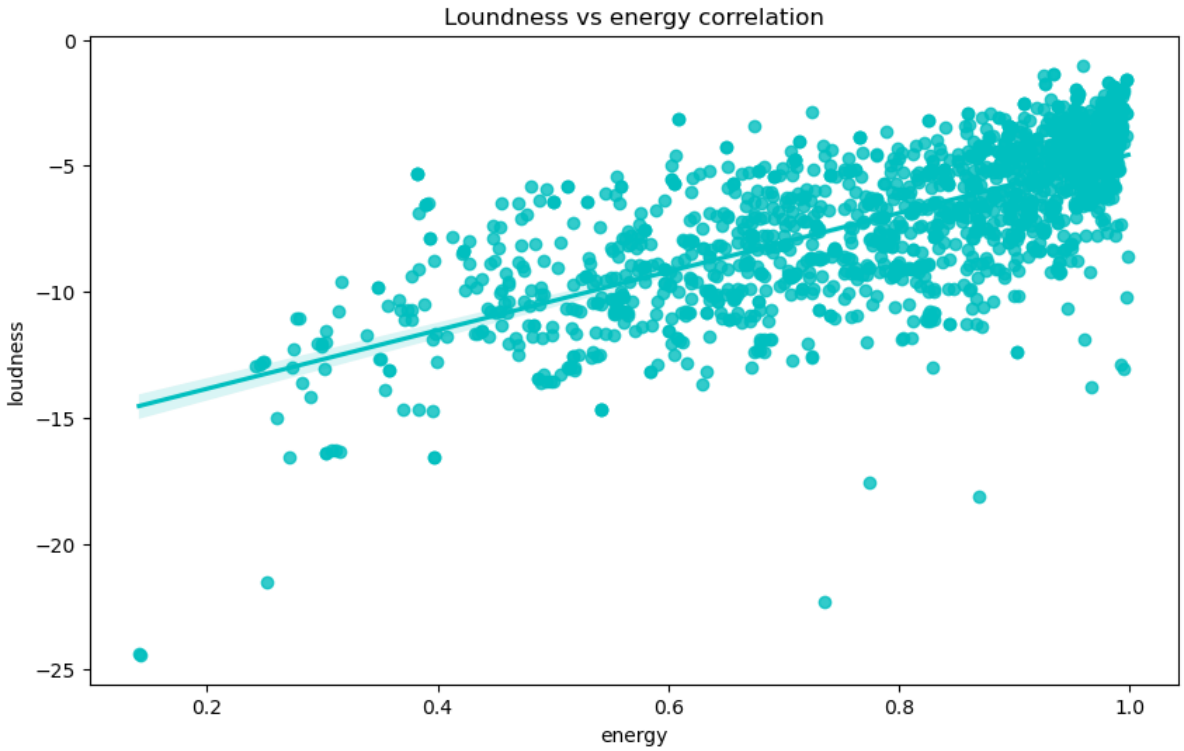
Out[58]:

Id	name	album	release_date	track_number	id	
0	Concert Intro Music - Live	Licked Live In NYC	2022-06-10	1	2IEkywLJ4ykbhi1yRQvmsT	spotify:tr
1	Street Fighting Man - Live	Licked Live In NYC	2022-06-10	2	6GVgVJBKkGJoRfarYRvGTU	spotify:trac
2	Start Me Up - Live	Licked Live In NYC	2022-06-10	3	1Lu761pZ0dBTGpzxaQoZNW	spotify:trac
3	If You Can't Rock Me - Live	Licked Live In NYC	2022-06-10	4	1agTQzOTUnGNggyckEqiDH	spotify:trac
4	Donâ€™t Stop - Live	Licked Live In NYC	2022-06-10	5	7piGJR8YndQBQWVXv6KtQw	spotify:track
...	...	...	...	...	...	
1605	Carol	The Rolling Stones	1964-04-16	8	08l7M5UpRnffGl0FyuRiQZ	spotify:tr
1606	Tell Me	The Rolling Stones	1964-04-16	9	3JZllQBsTM6WwoJdzFDLhx	spotify:trac
1607	Can I Get A Witness	The Rolling Stones	1964-04-16	10	0t2qvfSBQ3Y08lzRRoVTdb	spotify:tra
1608	You Can Make It If You Try	The Rolling Stones	1964-04-16	11	5ivIs5vwSj0RChOIvlY3On	spotify:t
1609	Walking The Dog	The Rolling Stones	1964-04-16	12	43SkTJJ2xleDaeiE4TIM70	spotify:t

1610 rows × 17 columns

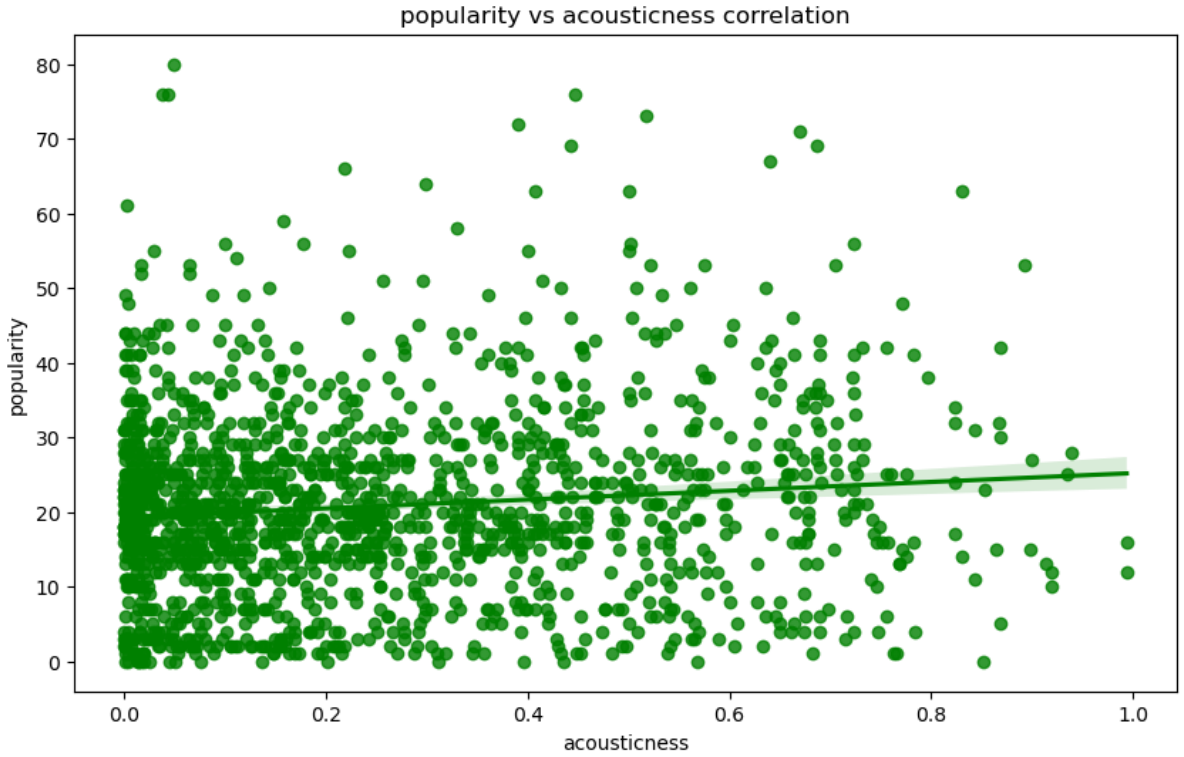
In [63]:
```
plt.figure(figsize=(10,6))
sns.regplot(data=df_Cohorts, y='loudness', x='energy', color="c").set(title=
```

Out[63]:
```
[Text(0.5, 1.0, 'Loundness vs energy correlation')]
```



In [64]:
```
plt.figure(figsize=(10,6))
sns.regplot(data=df_Cohorts, y='popularity', x='acousticness', color="g").se
```

Out[64]:
```
[Text(0.5, 1.0, 'popularity vs acousticness correlation')]
```



In [65]:
```
df_Cohorts.describe
```

Out[65]:

```
<bound method NDFrame.describe of                                 name
album release_date  \
Id
0      Concert Intro Music - Live  Licked Live In NYC   2022-06-10
1      Street Fighting Man - Live  Licked Live In NYC   2022-06-10
2              Start Me Up - Live  Licked Live In NYC   2022-06-10
3     If You Can't Rock Me - Live  Licked Live In NYC   2022-06-10
4             Donâ€™t Stop - Live  Licked Live In NYC   2022-06-10
...                           ...                 ...          ...
1605                      Carol  The Rolling Stones   1964-04-16
1606                   Tell Me  The Rolling Stones   1964-04-16
1607          Can I Get A Witness  The Rolling Stones   1964-04-16
1608   You Can Make It If You Try  The Rolling Stones   1964-04-16
1609              Walking The Dog  The Rolling Stones   1964-04-16

      track_number                      id  \
Id
0                1  2IEkywLJ4ykbhi1yRQvmsT
1                2  6GVgVJBKkGJoRfarYRvGTU
2                3  1Lu761pZ0dBTGpzxaQoZNW
3                4  1agTQzOTUnGNggyckEqiDH
4                5  7piGJR8YndQBQWVXv6KtQw
...            ...                     ...
1605             8  08l7M5UpRnffGl0FyuRiQZ
1606             9  3JZllQBsTM6WwoJdzFDLhx
1607            10  0t2qvfSBQ3Y08lzRRoVTdb
1608            11  5ivIs5vwSj0RChOIvlY3On
1609            12  43SkTJJ2xleDaeiE4TIM70

                                      uri  acousticness  danceability  \
Id
0     spotify:track:2IEkywLJ4ykbhi1yRQvmsT        0.0824         0.463
1     spotify:track:6GVgVJBKkGJoRfarYRvGTU        0.4370         0.326
2     spotify:track:1Lu761pZ0dBTGpzxaQoZNW        0.4160         0.386
3     spotify:track:1agTQzOTUnGNggyckEqiDH        0.5670         0.369
4     spotify:track:7piGJR8YndQBQWVXv6KtQw        0.4000         0.303
...                                    ...           ...           ...
1605  spotify:track:08l7M5UpRnffGl0FyuRiQZ        0.1570         0.466
1606  spotify:track:3JZllQBsTM6WwoJdzFDLhx        0.0576         0.509
1607  spotify:track:0t2qvfSBQ3Y08lzRRoVTdb        0.3710         0.790
1608  spotify:track:5ivIs5vwSj0RChOIvlY3On        0.2170         0.700
1609  spotify:track:43SkTJJ2xleDaeiE4TIM70        0.3830         0.727

      energy  instrumentalness  liveness  loudness  speechiness    tempo  \
Id
0      0.993          0.996000    0.9320   -12.913       0.1100  118.001
1      0.965          0.233000    0.9610    -4.803       0.0759  131.455
2      0.969          0.400000    0.9560    -4.936       0.1150  130.066
3      0.985          0.000107    0.8950    -5.535       0.1930  132.994
4      0.969          0.055900    0.9660    -5.098       0.0930  130.533
...      ...               ...       ...       ...          ...      ...
1605   0.932          0.006170    0.3240    -9.214       0.0429  177.340
1606   0.706          0.000002    0.5160    -9.427       0.0843  122.015
1607   0.774          0.000000    0.0669    -7.961       0.0720   97.035
1608   0.546          0.000070    0.1660    -9.567       0.0622  102.634
1609   0.934          0.068500    0.0965    -8.373       0.0359  125.275

      valence  popularity  duration
Id
0      0.0302          33        49
1      0.3180          34       253
2      0.3130          34       263
3      0.1470          32       306
4      0.2060          32       305
```

```
...       ...          ...       ...
1605   0.9670          39       154
1606   0.4460          36       245
1607   0.8350          30       176
1608   0.5320          27       122
1609   0.9690          35       189

[1610 rows x 17 columns]>
```

In [47]:
```
corr_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 12 entries, track_number to duration
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   track_number      12 non-null     float64
 1   acousticness      12 non-null     float64
 2   danceability      12 non-null     float64
 3   energy            12 non-null     float64
 4   instrumentalness  12 non-null     float64
 5   liveness          12 non-null     float64
 6   loudness          12 non-null     float64
 7   speechiness       12 non-null     float64
 8   tempo             12 non-null     float64
 9   valence           12 non-null     float64
 10  popularity        12 non-null     float64
 11  duration          12 non-null     float64
dtypes: float64(12)
memory usage: 1.5+ KB
```

In [49]:
```
corr_df.head()
```

Out[49]:

	track_number	acousticness	danceability	energy	instrumentalness	
track_number	1.000000	-0.035675	-0.112004	0.096314	-0.002772	(
acousticness	-0.035675	1.000000	0.070017	-0.363819	0.061403	-(
danceability	-0.112004	0.070017	1.000000	-0.300536	-0.031812	-(
energy	0.096314	-0.363819	-0.300536	1.000000	0.120261	
instrumentalness	-0.002772	0.061403	-0.031812	0.120261	1.000000	0

In [52]:
```
corr_df.shape[0]
```

Out[52]:
```
12
```

In [53]:
```
len(corr_df.index)
```

Out[53]:
```
12
```

In [54]:
```
df.shape[0]
```

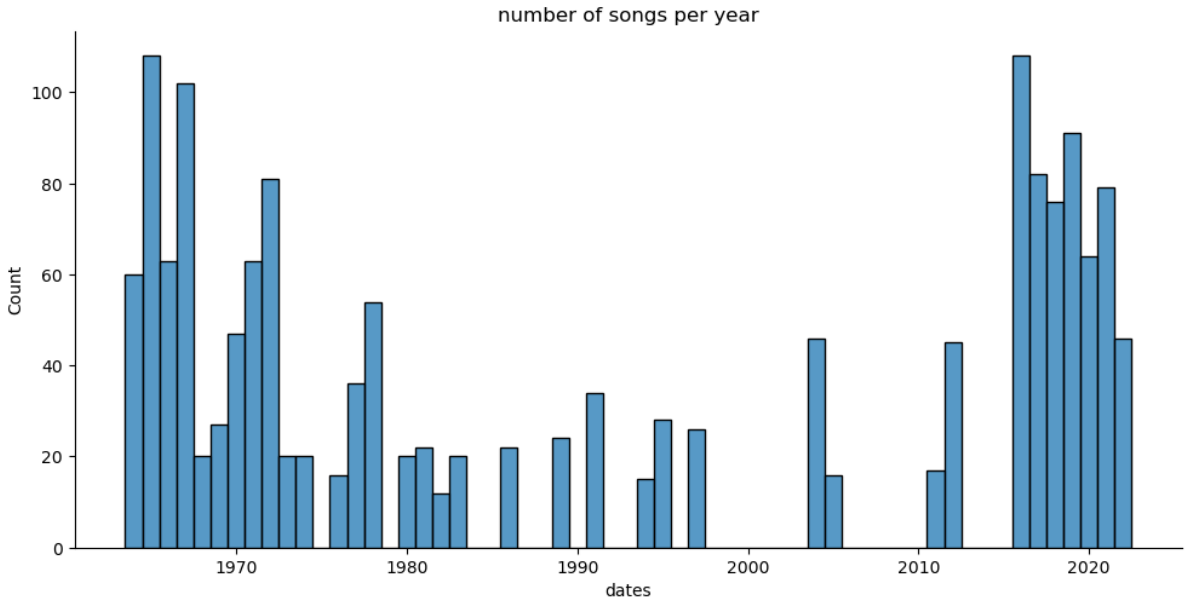
Out[54]:
```
1610
```

In [66]:
```
df_Cohorts.set_index('release_date', inplace=True)
```

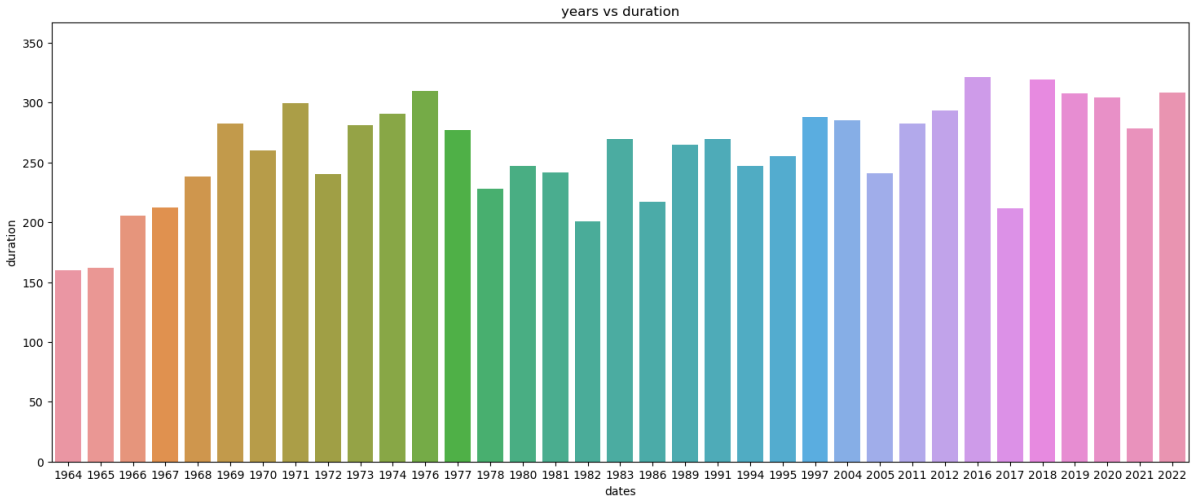
In [68]:
```
df_Cohorts['dates']=df_Cohorts.index.get_level_values('release_date')
df_Cohorts.dates=pd.to_datetime(df_Cohorts.dates)
years=df_Cohorts.dates.dt.year
```

In [69]:
```
sns.displot(years,discrete=True,aspect=2, height=5, kind='hist').set(title='
```

Out[69]:
```
<seaborn.axisgrid.FacetGrid at 0x7fcf4671b370>
```



In [73]:
```
total_dr=df_Cohorts.duration
fig_dims = (18,7)
fig, ax = plt.subplots(figsize = fig_dims)
fig = sns.barplot(x=years, y=total_dr, ax=ax, errwidth=False).set(title= 'ye
#plt.xtics(rotation=90)
```



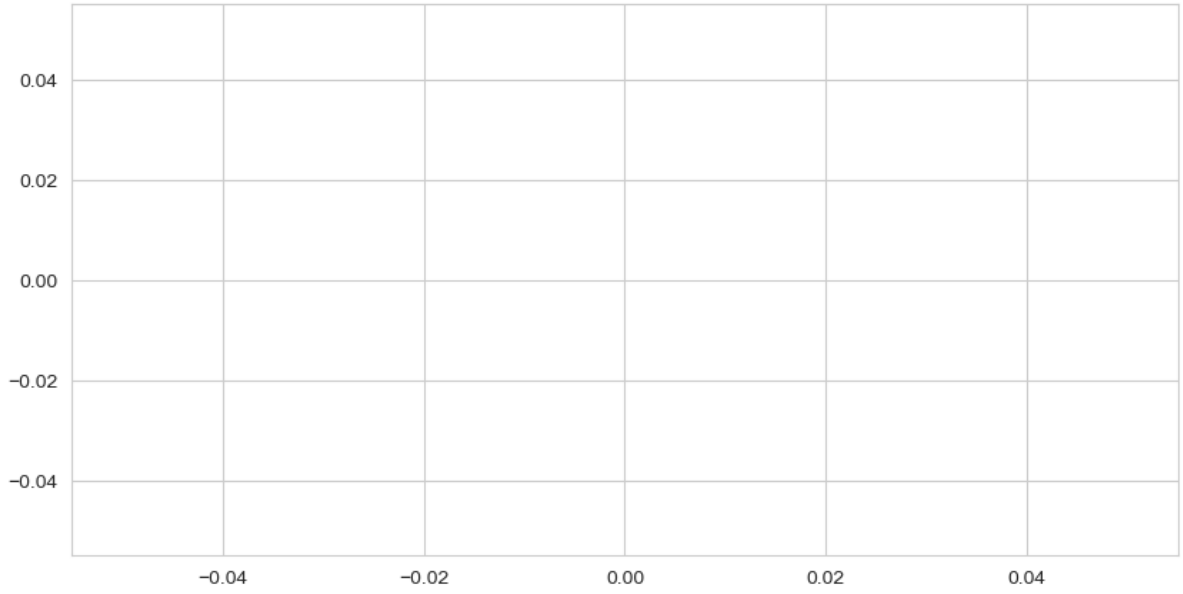
In [75]:
```
total_dr=df_Cohorts.duration
sns.set_style(style='whitegrid')
fig_dims = (10,5)
fig, ax = plt.subplots(figsize = fig_dims)

fig=sns.lineplot(x=years, y=total_dr, ax=ax).set(title='years vs duration')
```

```
/opt/anaconda3/lib/python3.9/site-packages/seaborn/_core.py:1057: FutureWarn
ing: reindexing with a non-unique Index is deprecated and will raise in a fu
ture version.
  comp_col.loc[orig.index] = pd.to_numeric(axis.convert_units(orig))
```

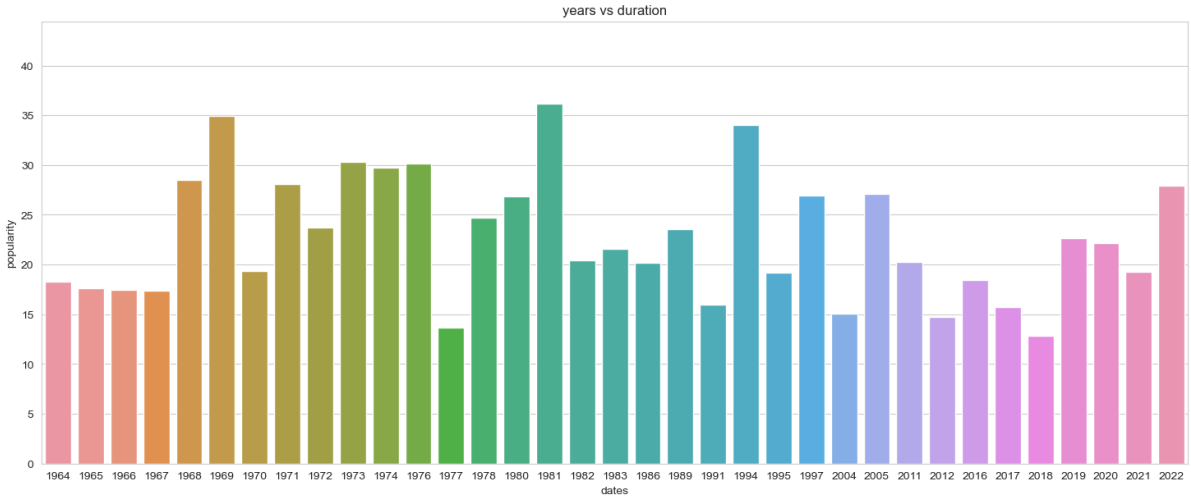
```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/var/folders/vl/ds4939js6cn74f1bkm98d0fm0000gn/T/ipykernel_65196/4263455024.
py in <module>
      4 fig, ax = plt.subplots(figsize = fig_dims)
      5
----> 6 fig=sns.lineplot(x=years, y=total_dr, ax=ax).set(title='years vs du
ration')

/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py in inner_f
(*args, **kwargs)
     44             )
     45         kwargs.update({k: arg for k, arg in zip(sig.parameters, arg
s)})
---> 46         return f(**kwargs)
     47     return inner_f
     48

/opt/anaconda3/lib/python3.9/site-packages/seaborn/relational.py in lineplot
(x, y, hue, size, style, data, palette, hue_order, hue_norm, sizes, size_ord
er, size_norm, dashes, markers, style_order, units, estimator, ci, n_boot, s
eed, sort, err_style, err_kws, legend, ax, **kwargs)
    708     p._attach(ax)
    709
--> 710     p.plot(ax, kwargs)
    711     return ax
    712

/opt/anaconda3/lib/python3.9/site-packages/seaborn/relational.py in plot(sel
f, ax, kws)
    469         # Loop over the semantic subsets and add to the plot
    470         grouping_vars = "hue", "size", "style"
--> 471         for sub_vars, sub_data in self.iter_data(grouping_vars, from
_comp_data=True):
    472
    473             if self.sort:

/opt/anaconda3/lib/python3.9/site-packages/seaborn/_core.py in iter_data(sel
f, grouping_vars, reverse, from_comp_data)
    981
    982         if from_comp_data:
--> 983             data = self.comp_data
    984         else:
    985             data = self.plot_data

/opt/anaconda3/lib/python3.9/site-packages/seaborn/_core.py in comp_data(sel
f)
   1055                     orig = self.plot_data[var].dropna()
   1056                 comp_col = pd.Series(index=orig.index, dtype=float,
name=var)
-> 1057                 comp_col.loc[orig.index] = pd.to_numeric(axis.conve
rt_units(orig))
   1058
   1059                 if axis.get_scale() == "log":

/opt/anaconda3/lib/python3.9/site-packages/pandas/core/indexing.py in __seti
tem__(self, key, value)
    714
    715         iloc = self if self.name == "iloc" else self.obj.iloc
--> 716         iloc._setitem_with_indexer(indexer, value, self.name)
    717
    718     def _validate_key(self, key, axis: int):

/opt/anaconda3/lib/python3.9/site-packages/pandas/core/indexing.py in _setit
```

```
em_with_indexer(self, indexer, value, name)
   1691             self._setitem_with_indexer_split_path(indexer, value, na
me)
   1692         else:
-> 1693             self._setitem_single_block(indexer, value, name)
   1694
   1695     def _setitem_with_indexer_split_path(self, indexer, value, nam
e: str):

/opt/anaconda3/lib/python3.9/site-packages/pandas/core/indexing.py in _setit
em_single_block(self, indexer, value, name)
   1932             # setting for extensionarrays that store dicts. Need to
decide
   1933             # if it's worth supporting that.
-> 1934             value = self._align_series(indexer, Series(value))
   1935
   1936         elif isinstance(value, ABCDataFrame) and name != "iloc":

/opt/anaconda3/lib/python3.9/site-packages/pandas/core/indexing.py in _align
_series(self, indexer, ser, multiindex_indexer)
   2094                 if obj.ndim == 2 and is_empty_indexer(indexer[0], s
er._values):
   2095                     return ser._values.copy()
-> 2096                 ser = ser.reindex(obj.axes[0][indexer[0]], copy=Tru
e)._values
   2097
   2098                 # single indexer

/opt/anaconda3/lib/python3.9/site-packages/pandas/core/series.py in reindex
(self, *args, **kwargs)
   4670                 )
   4671             kwargs.update({"index": index})
-> 4672         return super().reindex(**kwargs)
   4673
   4674     @deprecate_nonkeyword_arguments(version=None, allowed_args=["se
lf", "labels"])

/opt/anaconda3/lib/python3.9/site-packages/pandas/core/generic.py in reindex
(self, *args, **kwargs)
   4964
   4965         # perform the reindex on the axes
-> 4966         return self._reindex_axes(
   4967             axes, level, limit, tolerance, method, fill_value, copy
   4968         ).__finalize__(self, method="reindex")

/opt/anaconda3/lib/python3.9/site-packages/pandas/core/generic.py in _reinde
x_axes(self, axes, level, limit, tolerance, method, fill_value, copy)
   4984
   4985             axis = self._get_axis_number(a)
-> 4986             obj = obj._reindex_with_indexers(
   4987                 {axis: [new_index, indexer]},
   4988                 fill_value=fill_value,

/opt/anaconda3/lib/python3.9/site-packages/pandas/core/generic.py in _reinde
x_with_indexers(self, reindexers, fill_value, copy, allow_dups)
   5030
   5031             # TODO: speed up on homogeneous DataFrame objects (see _
reindex_multi)
-> 5032             new_data = new_data.reindex_indexer(
   5033                 index,
   5034                 indexer,

/opt/anaconda3/lib/python3.9/site-packages/pandas/core/internals/managers.py
in reindex_indexer(self, new_axis, indexer, axis, fill_value, allow_dups, co
```

```
py, consolidate, only_slice, use_na_proxy)
    674         # some axes don't allow reindexing with dups
    675         if not allow_dups:
--> 676             self.axes[axis]._validate_can_reindex(indexer)
    677
    678         if axis >= self.ndim:

/opt/anaconda3/lib/python3.9/site-packages/pandas/core/indexes/base.py in _v
alidate_can_reindex(self, indexer)
   4119         # trying to reindex on an axis with duplicates
   4120         if not self._index_as_unique and len(indexer):
-> 4121             raise ValueError("cannot reindex on an axis with duplica
te labels")
   4122
   4123     def reindex(

ValueError: cannot reindex on an axis with duplicate labels
```



In [76]:
```
total_dr=df_Cohorts.popularity
fig_dims = (18,7)
fig, ax = plt.subplots(figsize = fig_dims)
fig = sns.barplot(x=years, y=total_dr, ax=ax, errwidth=False).set(title= 'ye
#plt.xtics(rotation=90)
```



In [105…
```
df_Cohorts.groupby(['album']).mean('popularity')
```

Out[105]:

album	track_number	acousticness	danceability	energy	instrumentalness	liven
12 X 5	6.5	0.203860	0.489833	0.610583	0.132789	0.199
12 x 5	6.5	0.204471	0.503833	0.620583	0.128532	0.204
A Bigger Bang (2009 Re-Mastered)	8.5	0.124700	0.554625	0.838437	0.000865	0.280
A Bigger Bang (Live)	11.5	0.398091	0.334227	0.948318	0.431331	0.896
Aftermath	6.0	0.271906	0.594182	0.609182	0.084479	0.207
...	...	...	...	...	...	
Undercover	5.5	0.101420	0.590800	0.826700	0.282284	0.322
Undercover (2009 Re-Mastered)	5.5	0.105710	0.573400	0.932900	0.273168	0.298
Voodoo Lounge (Remastered 2009)	8.0	0.187616	0.485733	0.689733	0.060069	0.199
Voodoo Lounge Uncut (Live)	14.5	0.293407	0.416000	0.882000	0.114505	0.887
got LIVE if you want it!	6.5	0.066803	0.313417	0.878667	0.120302	0.646

90 rows × 12 columns

In [87]:
```
df_Cohorts
```

Out[87]:

release_date	name	album	track_number	id	
2022-06-10	Concert Intro Music - Live	Licked Live In NYC	1	2IEkywLJ4ykbhi1yRQvmsT	spotify:track:2IE
2022-06-10	Street Fighting Man - Live	Licked Live In NYC	2	6GVgVJBKkGJoRfarYRvGTU	spotify:track:6GVg
2022-06-10	Start Me Up - Live	Licked Live In NYC	3	1Lu761pZ0dBTGpzxaQoZNW	spotify:track:1Lu76
2022-06-10	If You Can't Rock Me - Live	Licked Live In NYC	4	1agTQzOTUnGNggyckEqiDH	spotify:track:1agT(
2022-06-10	Donâ€™t Stop - Live	Licked Live In NYC	5	7piGJR8YndQBQWVXv6KtQw	spotify:track:7piGJF
...	...	...	...	...	
1964-04-16	Carol	The Rolling Stones	8	08l7M5UpRnffGl0FyuRiQZ	spotify:track:08l
1964-04-16	Tell Me	The Rolling Stones	9	3JZllQBsTM6WwoJdzFDLhx	spotify:track:3JZll
1964-04-16	Can I Get A Witness	The Rolling Stones	10	0t2qvfSBQ3Y08lzRRoVTdb	spotify:track:0t2(
1964-04-16	You Can Make It If You Try	The Rolling Stones	11	5ivIs5vwSj0RChOIvlY3On	spotify:track:5iv
1964-04-16	Walking The Dog	The Rolling Stones	12	43SkTJJ2xleDaeiE4TIM70	spotify:track:43

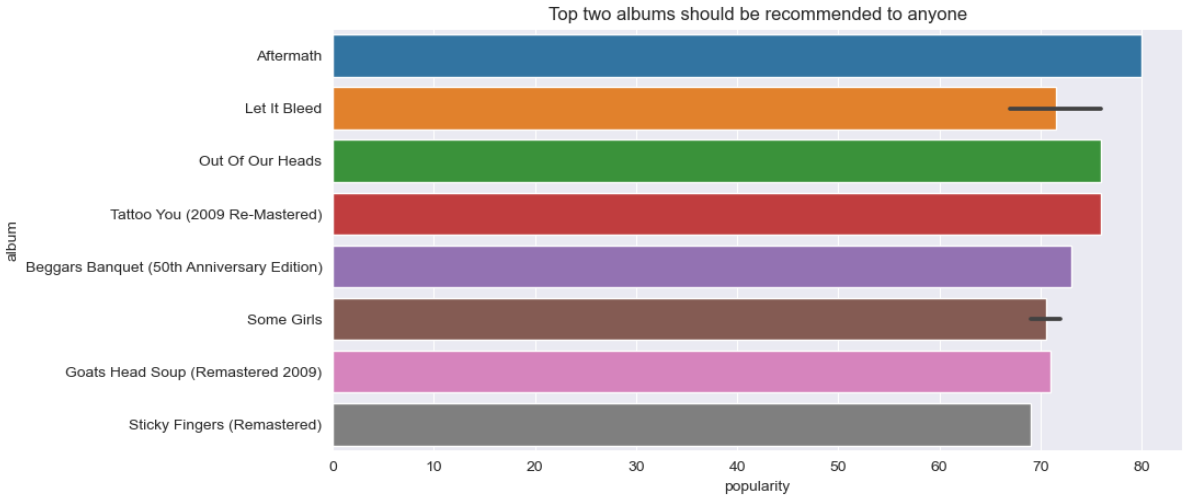
1610 rows × 17 columns

In [85]:
```
uniqueAlbum=df_Cohorts['album'].unique()
len(uniqueAlbum)
```

Out[85]:
```
90
```

In [104…
```
sns.set_style(style='darkgrid')
plt.figure(figsize=(10,5))
famous=df_Cohorts.sort_values("popularity", ascending=False).head(10)
sns.barplot(y='album', x='popularity', data=famous).set(title='Top two album
```

Out[104]:
```
[Text(0.5, 1.0, 'Top two albums should be recommended to anyone')]
```



In [106…
```
new_df_popularity = df_Cohorts[['album','popularity']].copy()
```

In [107…
```
new_df_popularity
```

Out[107]:

release_date	album	popularity
2022-06-10	Licked Live In NYC	33
2022-06-10	Licked Live In NYC	34
2022-06-10	Licked Live In NYC	34
2022-06-10	Licked Live In NYC	32
2022-06-10	Licked Live In NYC	32
...	...	...
1964-04-16	The Rolling Stones	39
1964-04-16	The Rolling Stones	36
1964-04-16	The Rolling Stones	30
1964-04-16	The Rolling Stones	27
1964-04-16	The Rolling Stones	35

1610 rows × 2 columns

In [ ]:
```
sns.set_style(style='darkgrid')
plt.figure(figsize=(10,5))
#famous=df_Cohorts.sort_values("popularity", ascending=False).head(10)
sns.barplot(y='album', x='popularity', data=famous).set(title='Top two album
```

In [117…
```
new_group=df_Cohorts.groupby('album')['popularity'].mean()
```

In [120…
```
new_group
```

Out[120]:
```
album
12 X 5                                32.083333
12 x 5                                 5.000000
A Bigger Bang (2009 Re-Mastered)      27.062500
A Bigger Bang (Live)                  18.181818
Aftermath                             43.090909
                                         ...
Undercover                            18.000000
Undercover (2009 Re-Mastered)         25.100000
Voodoo Lounge (Remastered 2009)       34.000000
Voodoo Lounge Uncut (Live)            11.678571
got LIVE if you want it!              15.333333
Name: popularity, Length: 90, dtype: float64
```

In [129…
```
albums = new_group.index.tolist()
```

In [132…
```
popularityavg = new_group.values
```

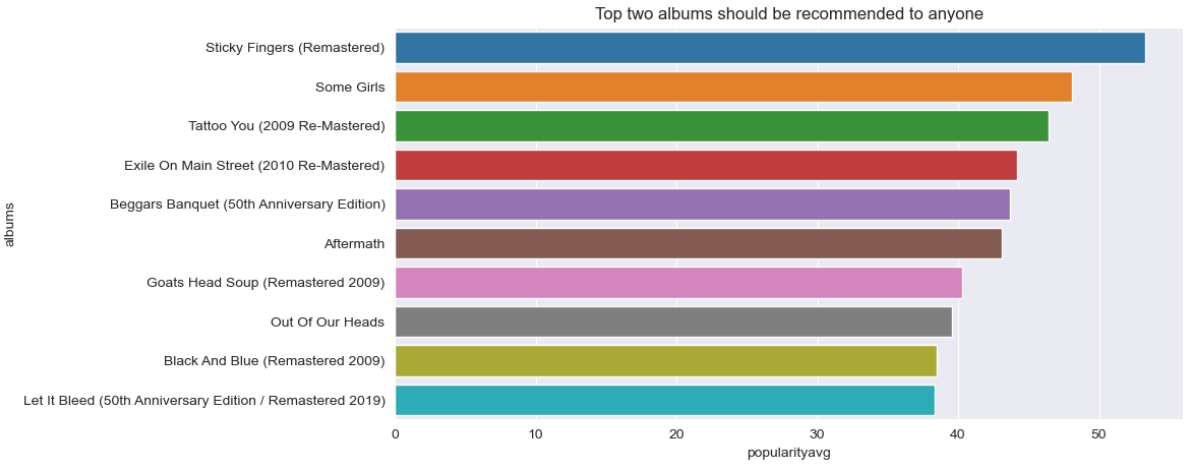
In [135…
```
len(albums)
len(popularityavg)
```

Out[135]:
```
90
```

In [137…
```
albumPopularitydf = pd.DataFrame({'albums':albums, 'popularityavg':popularit
```

In [140…
```
sns.set_style(style='darkgrid')
plt.figure(figsize=(10,5))
famous=albumPopularitydf.sort_values("popularityavg", ascending=False).head(
sns.barplot(y='albums', x='popularityavg', data=famous).set(title='Top two a
```

Out[140]:
```
[Text(0.5, 1.0, 'Top two albums should be recommended to anyone')]
```



In [141…
```
pca_new_data=df_Cohorts
```

In [143…
```
pca_new_data.columns
```

Out[143]:
```
Index(['name', 'album', 'track_number', 'id', 'uri', 'acousticness',
       'danceability', 'energy', 'instrumentalness', 'liveness', 'loudnes
s',
       'speechiness', 'tempo', 'valence', 'popularity', 'duration', 'date
s'],
      dtype='object')
```

In [149…
```
features=['track_number', 'acousticness',
       'danceability', 'energy', 'instrumentalness', 'liveness', 'loudness',
       'speechiness', 'tempo', 'valence', 'duration']
```

```
X=pca_new_data[features]
y=pca_new_data['popularity']
```

In [159…

```
from sklearn.model_selection import train_test_split
# create dataset
#X, y = make_blobs(n_samples=1000)
# split into train test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(1078, 11) (532, 11) (1078,) (532,)
```

In [160…

```
from sklearn.decomposition import PCA
from sklearn.linear_model import LinearRegression

pca = PCA(n_components = 1)

print(X_train.columns);
print(X_test.columns);
X_train = pca.fit_transform(X_train)

#model = LinearRegression()
#model.fit(X_train_pca, y_train)
#X_test = pca.transform(df_test)


X_test = pca.transform(X_test)
```

```
Index(['track_number', 'acousticness', 'danceability', 'energy',
       'instrumentalness', 'liveness', 'loudness', 'speechiness', 'tempo',
       'valence', 'duration'],
      dtype='object')
Index(['track_number', 'acousticness', 'danceability', 'energy',
       'instrumentalness', 'liveness', 'loudness', 'speechiness', 'tempo',
       'valence', 'duration'],
      dtype='object')
```

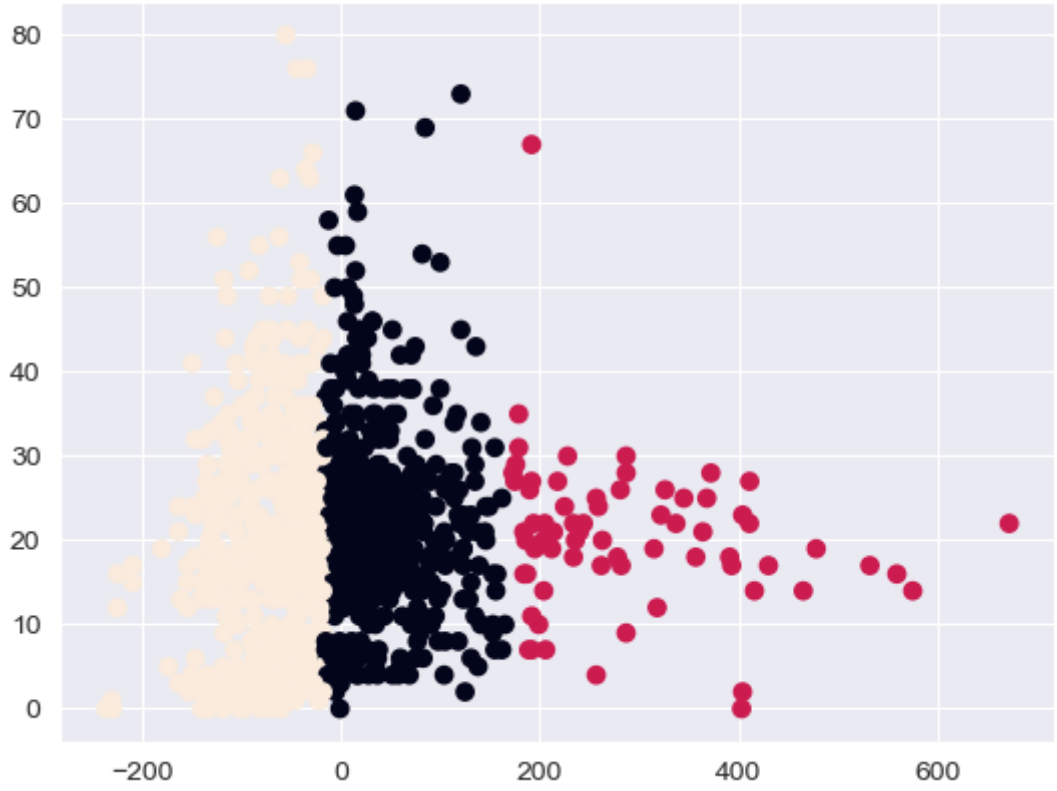
In [162…

```
data_kmeans = list(zip(X_train, y_train))
```

In [169…

```
kmeans = KMeans(n_clusters=3)
kmeans.fit(data_kmeans)

plt.scatter(X_train, y_train, c=kmeans.labels_)
plt.show()
```

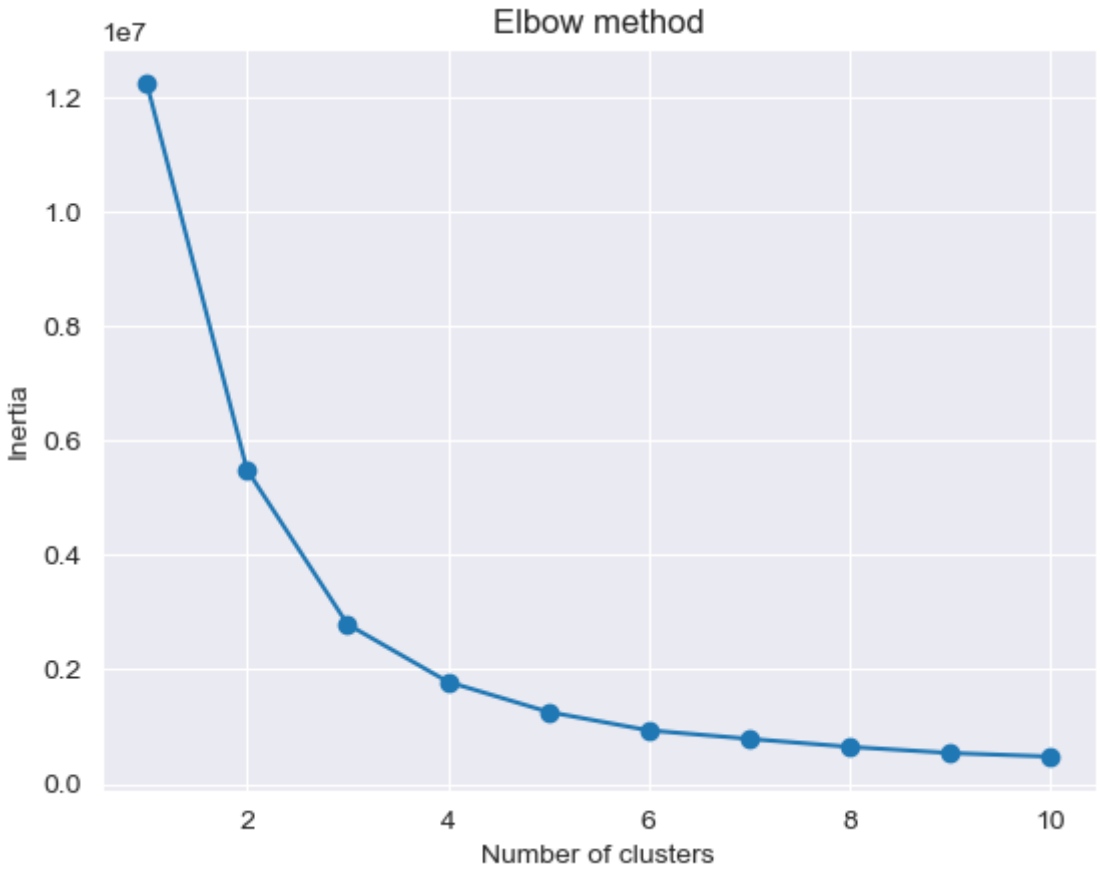
```
<__array_function__ internals>:5: VisibleDeprecationWarning: Creating an nda
rray from ragged nested sequences (which is a list-or-tuple of lists-or-tupl
es-or ndarrays with different lengths or shapes) is deprecated. If you meant
to do this, you must specify 'dtype=object' when creating the ndarray.
```



In [165…

```
inertias = []

for i in range(1,11):
    kmeans = KMeans(n_clusters=i)
    kmeans.fit(data_kmeans)
    inertias.append(kmeans.inertia_)

plt.plot(range(1,11), inertias, marker='o')
plt.title('Elbow method')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
plt.show()
```

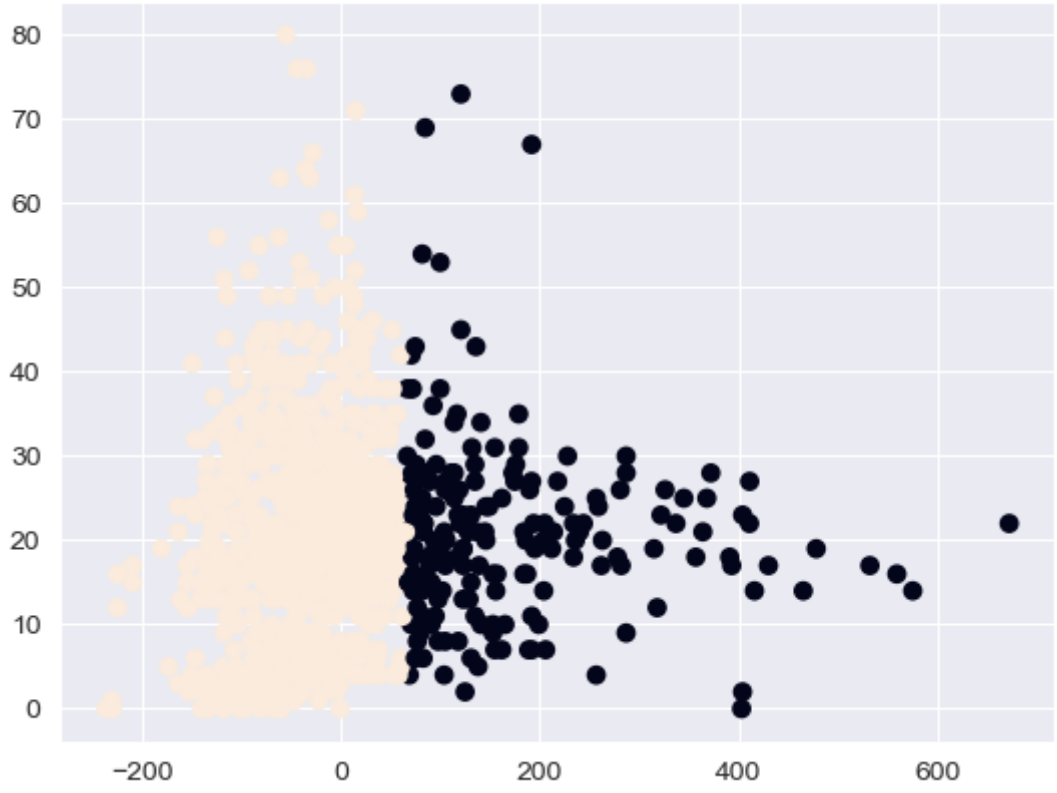
```
<__array_function__ internals>:5: VisibleDeprecationWarning: Creating an nda
rray from ragged nested sequences (which is a list-or-tuple of lists-or-tupl
es-or ndarrays with different lengths or shapes) is deprecated. If you meant
to do this, you must specify 'dtype=object' when creating the ndarray.
<__array_function__ internals>:5: VisibleDeprecationWarning: Creating an nda
rray from ragged nested sequences (which is a list-or-tuple of lists-or-tupl
es-or ndarrays with different lengths or shapes) is deprecated. If you meant
to do this, you must specify 'dtype=object' when creating the ndarray.
<__array_function__ internals>:5: VisibleDeprecationWarning: Creating an nda
rray from ragged nested sequences (which is a list-or-tuple of lists-or-tupl
es-or ndarrays with different lengths or shapes) is deprecated. If you meant
to do this, you must specify 'dtype=object' when creating the ndarray.
<__array_function__ internals>:5: VisibleDeprecationWarning: Creating an nda
rray from ragged nested sequences (which is a list-or-tuple of lists-or-tupl
es-or ndarrays with different lengths or shapes) is deprecated. If you meant
to do this, you must specify 'dtype=object' when creating the ndarray.
<__array_function__ internals>:5: VisibleDeprecationWarning: Creating an nda
rray from ragged nested sequences (which is a list-or-tuple of lists-or-tupl
es-or ndarrays with different lengths or shapes) is deprecated. If you meant
to do this, you must specify 'dtype=object' when creating the ndarray.
<__array_function__ internals>:5: VisibleDeprecationWarning: Creating an nda
rray from ragged nested sequences (which is a list-or-tuple of lists-or-tupl
es-or ndarrays with different lengths or shapes) is deprecated. If you meant
to do this, you must specify 'dtype=object' when creating the ndarray.
<__array_function__ internals>:5: VisibleDeprecationWarning: Creating an nda
rray from ragged nested sequences (which is a list-or-tuple of lists-or-tupl
es-or ndarrays with different lengths or shapes) is deprecated. If you meant
to do this, you must specify 'dtype=object' when creating the ndarray.
<__array_function__ internals>:5: VisibleDeprecationWarning: Creating an nda
rray from ragged nested sequences (which is a list-or-tuple of lists-or-tupl
es-or ndarrays with different lengths or shapes) is deprecated. If you meant
to do this, you must specify 'dtype=object' when creating the ndarray.
<__array_function__ internals>:5: VisibleDeprecationWarning: Creating an nda
rray from ragged nested sequences (which is a list-or-tuple of lists-or-tupl
es-or ndarrays with different lengths or shapes) is deprecated. If you meant
to do this, you must specify 'dtype=object' when creating the ndarray.
<__array_function__ internals>:5: VisibleDeprecationWarning: Creating an nda
rray from ragged nested sequences (which is a list-or-tuple of lists-or-tupl
es-or ndarrays with different lengths or shapes) is deprecated. If you meant
to do this, you must specify 'dtype=object' when creating the ndarray.
```



In [168…

```
kmeans = KMeans(n_clusters=2)
kmeans.fit(data_kmeans)

plt.scatter(X_train, y_train, c=kmeans.labels_)
plt.show()
```

```
<__array_function__ internals>:5: VisibleDeprecationWarning: Creating an nda
rray from ragged nested sequences (which is a list-or-tuple of lists-or-tupl
es-or ndarrays with different lengths or shapes) is deprecated. If you meant
to do this, you must specify 'dtype=object' when creating the ndarray.
```



In [ ]: