The Spotify Tracks Dataset:

This is a dataset of Spotify tracks over a range of 114 different genres. Each track is associated with the following features including the artist, track name, and various audio features. We decided to focus on predicting `track_genre` given the dataset's numerical features while setting aside the categorical features like `artists`, `album_name`, and `track_name` which may not be as informative.

</br>

`track_id` : The Spotify ID for the track

`artists` : The artists' names who performed the track. If there is more than one artist, they are separated by a ;

`album_name` : The album name in which the track appears

`track_name` : Name of the track

`popularity` : The popularity of a track is a value between 0 and 100, with 100 being the most popular. The popularity is calculated by algorithm and is based, in the most part, on the total number of plays the track has had and how recent those plays are. Generally speaking, songs that are being played a lot now will have a higher popularity than songs that were played a lot in the past. Duplicate tracks (e.g. the same track from a single and an album) are rated independently. Artist and album popularity is derived mathematically from track popularity.

`duration_ms` : The track length in milliseconds

`explicit` : Whether or not the track has explicit lyrics (true = yes it does; false = no it does not OR unknown)

`danceability` : Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable

`energy` : Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. For example, death metal has high energy, while a Bach prelude scores low on the scale

`key` : The key the track is in. Integers map to pitches using standard Pitch Class notation. E.g. 0 = C, 1 = C♯/D♭, 2 = D, and so on. If no key was detected, the value is -1

`loudness` : The overall loudness of a track in decibels (dB)

`mode` : Mode indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived. Major is represented by 1 and minor is 0

`speechiness` : Speechiness detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value. Values above 0.66 describe tracks that are probably made entirely of spoken words. Values between 0.33 and 0.66 describe tracks that may contain both music and speech, either in sections or layered, including such cases as rap music. Values below 0.33 most likely represent music and other non-speech-like tracks

`acousticness` : A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic instrumentalness: Predicts whether a track contains no vocals. "Ooh" and "aah" sounds are treated as instrumental in this context. Rap or spoken word tracks are clearly "vocal". The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content `liveness` : Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. A value above 0.8 provides strong likelihood that the track is live

`valence` : A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry)

`tempo` : The overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece and derives directly from the average beat duration

`time_signature` : An estimated time signature. The time signature (meter) is a notational convention to specify how many beats are in each bar (or measure). The time signature ranges from 3 to 7 indicating time signatures of 3/4, to 7/4.

`track_genre` : The genre in which the track belongs

I. Data Pre-Processing

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

pd.set_option('display.max_columns', None)
pd.options.display.max_colwidth = 500
pd.options.display.max_rows = 100
```

```
/Users/peiyuanlee/miniforge3/envs/myenv/lib/python3.11/site-packages/pandas/core/arrays/masked.py:60: UserWarning: Pandas requires version '1.3.6' or newer of 'bottleneck' (version '1.3.5' currently installed).
  from pandas.core import (
```

In [2]:
```
# Load the dataset
df_raw = pd.read_csv("hf://datasets/maharshipandya/spotify-tracks-dataset/dataset.csv")
```

In [ ]:
```
# Examine the first 10 rows (song tracks)
df_raw.head(10)
```

Out[ ]:

	Unnamed: 0	track_id	artists	album_name	track_name	popularity	duration_ms	explicit	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	vale
0	0	5SuOikwiRyPMVoIQDJUgSV	Gen Hoshino	Comedy	Comedy	73	230666	False	0.676	0.4610	1	-6.746	0	0.1430	0.0322	0.000001	0.3580	0.7
1	1	4qPNDBW1i3p13qLCt0Ki3A	Ben Woodward	Ghost (Acoustic)	Ghost - Acoustic	55	149610	False	0.420	0.1660	1	-17.235	1	0.0763	0.9240	0.000006	0.1010	0.2
2	2	1iJBSr7s7jYXzM8EGcbK5b	Ingrid Michaelson;ZAYN	To Begin Again	To Begin Again	57	210826	False	0.438	0.3590	0	-9.734	1	0.0557	0.2100	0.000000	0.1170	0.1
3	3	6lfxq3CG4xtTiEg7opyCyx	Kina Grannis	Crazy Rich Asians (Original Motion Picture Soundtrack)	Can't Help Falling In Love	71	201933	False	0.266	0.0596	0	-18.515	1	0.0363	0.9050	0.000071	0.1320	0.1
4	4	5vjLSffimiIP26QG5WcN2K	Chord Overstreet	Hold On	Hold On	82	198853	False	0.618	0.4430	2	-9.681	1	0.0526	0.4690	0.000000	0.0829	0.1
5	5	01MVOl9KtVTNfFiBU9I7dc	Tyrone Wells	Days I Will Remember	Days I Will Remember	58	214240	False	0.688	0.4810	6	-8.807	1	0.1050	0.2890	0.000000	0.1890	0.6
6	6	6Vc5wAMmXdKIAM7WUoEb7N	A Great Big World;Christina Aguilera	Is There Anybody Out There?	Say Something	74	229400	False	0.407	0.1470	2	-8.822	1	0.0355	0.8570	0.000003	0.0913	0.0
7	7	1EzrEOXmMH3G43AXT1y7pA	Jason Mraz	We Sing. We Dance. We Steal Things.	I'm Yours	80	242946	False	0.703	0.4440	11	-9.331	1	0.0417	0.5590	0.000000	0.0973	0.7
8	8	0IktbUcnAGrvD03AWnz3Q8	Jason Mraz;Colbie Caillat	We Sing. We Dance. We Steal Things.	Lucky	74	189613	False	0.625	0.4140	0	-8.700	1	0.0369	0.2940	0.000000	0.1510	0.6
9	9	7k9GuJYLp2AzqokyEdwEw2	Ross Copperman	Hunger	Hunger	56	205594	False	0.442	0.6320	1	-6.770	1	0.0295	0.4260	0.004190	0.0735	0.1

In [ ]:

```
# Examine a random sample of 10 rows (song tracks)
df_raw.sample(n=10)
```

Out[ ]:

	Unnamed: 0	track_id	artists	album_name	track_name	popularity	duration_ms	explicit	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness
3647	3647	27sytaeEm6TDVMpdExyVfd	O Rappa	Só as Melhores do Pop Rock Brasileiro	A feira	0	239533	False	0.713	0.886	7	-7.750	1	0.0411	0.0012	0.000008
56533	56533	0w2emroZUEoocjdVbK5F69	Claire Rosinkranz	Die For You	Boy In A Billion	0	207678	False	0.513	0.458	10	-7.300	1	0.1420	0.4370	0.000000
63240	63240	0qjhMbCmCSZC2f4qohgcxa	Eikichi Yazawa	the Name Is... (50th Anniversary Remastered)	アリよさらば - Remastered 2022	38	255893	False	0.682	0.828	11	-8.009	0	0.0571	0.2410	0.000000
17958	17958	1bYscy2XwW0fKJoNrALWP7	Deathpact	SPLIT // PERSONALITY	SONG SIX	37	290666	False	0.541	0.891	4	-4.399	0	0.0595	0.0703	0.381000
54714	54714	6ApQUXDhO9qcqVz6Q4OCJY	Datassette	Existenzmaximum - EP	Holiday 88	10	282125	False	0.713	0.707	4	-14.870	0	0.0511	0.0827	0.879000
78606	78606	5wOby0SgajxMIOFce5HLyh	voXXclub	Hitmedley	Hitmedley	27	358653	False	0.407	0.936	3	-4.503	1	0.1950	0.0280	0.000000
97310	97310	4nzbkdiJ0GzxNUhVFz43j4	Atitude 67	Atitude 67 (Ao Vivo)	Casal Do Ano (Plutão) - Ao Vivo	51	207853	False	0.704	0.865	4	-5.494	1	0.0678	0.4890	0.000000
106326	106326	3tXgGYRE0spiBVxaa9Xr79	Lars Winnerbäck	Hosianna	Utkast till ett brev	41	278106	False	0.662	0.897	4	-4.742	0	0.0356	0.0173	0.005950
81986	81986	4tjLYTXFqZhkUDga4bQ0yl	Neha Kakkar;Dhvani Bhanushali;Ikka;Tanishk Bagchi	Dilbar (From "Satyameva Jayate")	Dilbar (From "Satyameva Jayate")	66	184432	False	0.725	0.912	9	-3.665	0	0.0851	0.1550	0.000077
84776	84776	5gOnivVq0hLxPvIPC00ZhF	T. Rex	Electric Warrior	Cosmic Dancer	58	266533	False	0.363	0.803	0	-8.089	1	0.0600	0.0117	0.006010

In [ ]:
```
# Examine the number of features and observations
df_raw.shape
```

Out[ ]:
```
(114000, 21)
```

Checking for feature relevance, duplicates, and missing data:

In [ ]:
```
# Examine features given by the dataset
df_raw.columns
```

Out[ ]:
```
Index(['Unnamed: 0', 'track_id', 'artists', 'album_name', 'track_name',
       'popularity', 'duration_ms', 'explicit', 'danceability', 'energy',
       'key', 'loudness', 'mode', 'speechiness', 'acousticness',
       'instrumentalness', 'liveness', 'valence', 'tempo', 'time_signature',
       'track_genre'],
      dtype='object')
```

In [ ]:
```
# Examine feature data types and missingness
df_raw.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 114000 entries, 0 to 113999
Data columns (total 21 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   Unnamed: 0        114000 non-null  int64  
 1   track_id          114000 non-null  object 
 2   artists           113999 non-null  object 
 3   album_name        113999 non-null  object 
 4   track_name        113999 non-null  object 
 5   popularity        114000 non-null  int64  
 6   duration_ms       114000 non-null  int64  
 7   explicit          114000 non-null  bool   
 8   danceability      114000 non-null  float64
 9   energy            114000 non-null  float64
 10  key               114000 non-null  int64  
 11  loudness          114000 non-null  float64
 12  mode              114000 non-null  int64  
 13  speechiness       114000 non-null  float64
 14  acousticness      114000 non-null  float64
 15  instrumentalness  114000 non-null  float64
 16  liveness          114000 non-null  float64
 17  valence           114000 non-null  float64
 18  tempo             114000 non-null  float64
 19  time_signature    114000 non-null  int64  
 20  track_genre       114000 non-null  object 
dtypes: bool(1), float64(9), int64(6), object(5)
memory usage: 17.5+ MB
```

We see that the datset has 114000 samples for 21 features with the only missing data being `artists`, `album_name`, and `track_name` each having 1 null observation. Since this is not many observations, it is safe to just drop them.

In [3]:
```
df_raw.dropna(axis=0,inplace=True)
df_raw.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 113999 entries, 0 to 113999
Data columns (total 21 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   Unnamed: 0        113999 non-null  int64  
 1   track_id          113999 non-null  object 
 2   artists           113999 non-null  object 
 3   album_name        113999 non-null  object 
 4   track_name        113999 non-null  object 
 5   popularity        113999 non-null  int64  
 6   duration_ms       113999 non-null  int64  
 7   explicit          113999 non-null  bool   
 8   danceability      113999 non-null  float64
 9   energy            113999 non-null  float64
 10  key               113999 non-null  int64  
 11  loudness          113999 non-null  float64
 12  mode              113999 non-null  int64  
 13  speechiness       113999 non-null  float64
 14  acousticness      113999 non-null  float64
 15  instrumentalness  113999 non-null  float64
 16  liveness          113999 non-null  float64
 17  valence           113999 non-null  float64
 18  tempo             113999 non-null  float64
 19  time_signature    113999 non-null  int64  
 20  track_genre       113999 non-null  object 
dtypes: bool(1), float64(9), int64(6), object(5)
memory usage: 18.4+ MB
```

In [ ]:
```
# Ensuring that 'track_id' and 'Unnamed: 0" are entirely arbitrary
print("track_id: ",df_raw.track_id.nunique(), "/",113999)
print("Unnamed: 0: ",df_raw['Unnamed: 0'].nunique(), "/",113999)
```

```
track_id:  89740 / 113999
Unnamed: 0:  113999 / 113999
```

The feature `Unnamed: 0` is unique per track, thus, can be removed. However, `track_id` seems to have duplicates, perhaps in terms of nominal variables like `explicit`, `mode`, `key`, or `track_genre` since there can be different versions of the same song in terms of these variables. We will isolate each feature as a potential explanation for the duplicates.

In [4]:
```
# Dropping the feature 'Unnamed: 0"
df_raw.drop('Unnamed: 0', axis=1, inplace=True)
```

In [5]:
```
# Sample the first 20 rows that have duplicated track IDs
dup_tracks = df_raw[df_raw.duplicated(subset=['track_id'], keep=False)].sort_values(by='track_id')
dup_tracks.head(20)
```

Out[5]:

	track_id	artists	album_name	track_name	popularity	duration_ms	explicit	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence
15028	001APMDOl3qtx1526T11n1	Pink Sweat$;Kirby	New RnB	Better	0	176320	False	0.613	0.471	1	-6.644	0	0.1070	0.31600	0.000001	0.1170	0.406
103211	001APMDOl3qtx1526T11n1	Pink Sweat$;Kirby	New RnB	Better	0	176320	False	0.613	0.471	1	-6.644	0	0.1070	0.31600	0.000001	0.1170	0.406
85578	001YQlnDSduXd5LgBd66gT	Soda Stereo	Soda Stereo (Remastered)	El Tiempo Es Dinero - Remasterizado 2007	38	177266	False	0.554	0.921	2	-4.589	1	0.0758	0.01940	0.088100	0.3290	0.700
100420	001YQlnDSduXd5LgBd66gT	Soda Stereo	Soda Stereo (Remastered)	El Tiempo Es Dinero - Remasterizado 2007	38	177266	False	0.554	0.921	2	-4.589	1	0.0758	0.01940	0.088100	0.3290	0.700
91801	003vvx7Niy0yvhvHt4a68B	The Killers	Hot Fuss	Mr. Brightside	86	222973	False	0.352	0.911	1	-5.230	1	0.0747	0.00121	0.000000	0.0995	0.236
3257	003vvx7Niy0yvhvHt4a68B	The Killers	Hot Fuss	Mr. Brightside	86	222973	False	0.352	0.911	1	-5.230	1	0.0747	0.00121	0.000000	0.0995	0.236
2106	003vvx7Niy0yvhvHt4a68B	The Killers	Hot Fuss	Mr. Brightside	86	222973	False	0.352	0.911	1	-5.230	1	0.0747	0.00121	0.000000	0.0995	0.236
33178	004h8smbIoAkUNDJvVKwkG	Ouse;Powfu	Loners Diary	Lovemark	58	219482	True	0.808	0.331	5	-13.457	1	0.0557	0.13100	0.000000	0.2250	0.337
94239	004h8smbIoAkUNDJvVKwkG	Ouse;Powfu	Loners Diary	Lovemark	58	219482	True	0.808	0.331	5	-13.457	1	0.0557	0.13100	0.000000	0.2250	0.337
97533	006rHBBNLJMpQs8fRC2GDe	Calcinha Preta;Gusttavo Lima	CP 25 Anos (Ao Vivo em Aracaju)	Agora Estou Sofrendo - Ao Vivo	47	260510	False	0.605	0.678	0	-3.257	1	0.0311	0.64200	0.000000	0.1570	0.439
77391	006rHBBNLJMpQs8fRC2GDe	Calcinha Preta;Gusttavo Lima	CP 25 Anos (Ao Vivo em Aracaju)	Agora Estou Sofrendo - Ao Vivo	47	260510	False	0.605	0.678	0	-3.257	1	0.0311	0.64200	0.000000	0.1570	0.439
35138	006rHBBNLJMpQs8fRC2GDe	Calcinha Preta;Gusttavo Lima	CP 25 Anos (Ao Vivo em Aracaju)	Agora Estou Sofrendo - Ao Vivo	47	260510	False	0.605	0.678	0	-3.257	1	0.0311	0.64200	0.000000	0.1570	0.439
112131	006tmNZLXEXPqdb23wwSN1	İlhan İrem	Bezginin Gizli Mektupları	Yemyeşil Bir Deniz	44	358173	False	0.486	0.568	9	-9.199	0	0.0417	0.65200	0.000000	0.8340	0.650
64662	006tmNZLXEXPqdb23wwSN1	İlhan İrem	Bezginin Gizli Mektupları	Yemyeşil Bir Deniz	44	358173	False	0.486	0.568	9	-9.199	0	0.0417	0.65200	0.000000	0.8340	0.650
62346	006tmNZLXEXPqdb23wwSN1	İlhan İrem	Bezginin Gizli Mektupları	Yemyeşil Bir Deniz	44	358173	False	0.486	0.568	9	-9.199	0	0.0417	0.65200	0.000000	0.8340	0.650
63142	006tmNZLXEXPqdb23wwSN1	İlhan İrem	Bezginin Gizli Mektupları	Yemyeşil Bir Deniz	44	358173	False	0.486	0.568	9	-9.199	0	0.0417	0.65200	0.000000	0.8340	0.650
64246	00970cTs7LnxWt0d5Qk08m	Ella Fitzgerald	Weihnachtslieder 2022	Sleigh Ride	0	175986	False	0.593	0.287	1	-12.472	1	0.0469	0.76400	0.000000	0.1530	0.639
8095	00970cTs7LnxWt0d5Qk08m	Ella Fitzgerald	Weihnachtslieder 2022	Sleigh Ride	0	175986	False	0.593	0.287	1	-12.472	1	0.0469	0.76400	0.000000	0.1530	0.639
71588	00B7SBwrjbycLMOgAmeIU8	Red Hot Chili Peppers	Return of the Dream Canteen	Reach Out	66	251588	False	0.663	0.710	11	-5.550	0	0.0599	0.00745	0.005590	0.1470	0.487
2870	00B7SBwrjbycLMOgAmeIU8	Red Hot Chili Peppers	Return of the Dream Canteen	Reach Out	66	251588	False	0.663	0.710	11	-5.550	0	0.0599	0.00745	0.005590	0.1470	0.487

In [6]:
```
# number of complete duplicates
dup_num = dup_tracks[dup_tracks.duplicated(keep=False)].shape[0]
dup_num
```

Out[6]: 894

In [7]:
```
# Dropping the feature 'track_id"
df_raw.drop('track_id', axis=1, inplace=True)

# Remove the duplicates
df_raw.drop_duplicates(inplace=True)
```

In [8]:
```
print("number of duplicates in terms of all other features except for...")
cols_to_check = list(dup_tracks.columns)
dup_cols = list(dup_tracks.columns)
for i  in cols_to_check:
  dup_cols.remove(i)
  print(i,': ', dup_tracks[dup_tracks.duplicated(subset=dup_cols, keep=False)].shape[0]-dup_num)
  dup_cols.append(i)
```

```
number of duplicates in terms of all other features except for...
track_id :  151
artists :  0
album_name :  0
track_name :  0
popularity :  0
duration_ms :  0
explicit :  0
danceability :  0
energy :  0
key :  0
loudness :  0
mode :  0
speechiness :  0
acousticness :  0
instrumentalness :  0
liveness :  0
valence :  0
tempo :  0
time_signature :  0
track_genre :  38948
```

The duplicates seems to be the exact same tracks listed under either multiple genres (38948 of these) or listed under different track IDs (151). We will remove the tracks with duplicated track_IDs but keep the tracks listed under multiple genres since `track_genre` is our response variable.

In [9]:
```
print("number of duplicates in terms of all other features except for...")
cols_to_check = list(df_raw.columns)
dup_cols = list(df_raw.columns)
for i  in cols_to_check:
  dup_cols.remove(i)
  print(i,': ', df_raw[df_raw.duplicated(subset=dup_cols, keep=False)].shape[0])
  dup_cols.append(i)
```

```
number of duplicates in terms of all other features except for...
artists :  0
album_name :  9238
track_name :  2
popularity :  293
duration_ms :  0
explicit :  0
danceability :  0
energy :  0
key :  0
loudness :  0
mode :  0
speechiness :  0
acousticness :  0
instrumentalness :  0
liveness :  0
valence :  0
tempo :  2
time_signature :  0
track_genre :  38967
```

There are tracks that are exactly the same but received different `popularity` ratings. According to the features documentation, `popularity` is "calculated by algorithm and is based, in the most part, on the total number of plays the track has had and how recent those plays are." In order to reflect the performance of the tracks, we will keep the observation with the highest popularity rating and remove the remaining duplicates.

In [10]:
```
# sort in descending order by popularity
df_raw.sort_values(by='popularity',ascending=False).head(20)

# keep only the first occurance of the duplicate (i.e. observation with the max popularity value)
dup_cols = list(df_raw.columns)
dup_cols.remove('popularity')
df_raw.drop_duplicates(subset=dup_cols, keep='first', inplace=True)
```

There are also tracks with the exact same values for all features except `track_name`, `album_name`, and `tempo`. We will examine these duplications to ensure it is fair to remove their duplicates.

In [11]:
```
# same track under different names??
dup_cols = list(df_raw.columns)
dup_cols.remove('track_name')
df_raw[df_raw.duplicated(subset=dup_cols, keep=False)].head(10)
```

Out[11]:

	artists	album_name	track_name	popularity	duration_ms	explicit	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	time_signature	track_genre
49371	UVIQUE	InfeXious Euphoric - Chapter One	Falling	0	178374	False	0.43	0.781	3	-5.601	1	0.0334	0.0108	0.734	0.0818	0.206	75.017	4	hardstyle
49376	UVIQUE	InfeXious Euphoric - Chapter One	Falling - Radio Mix	0	178374	False	0.43	0.781	3	-5.601	1	0.0334	0.0108	0.734	0.0818	0.206	75.017	4	hardstyle

In [12]:
```
# same track under different album names??
dup_cols = list(df_raw.columns)
dup_cols.remove('album_name')
df_raw[df_raw.duplicated(subset=dup_cols, keep=False)].head(10)
```

Out[12]:

	artists	album_name	track_name	popularity	duration_ms	explicit	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	time_signature	track_genre
26	Jason Mraz	Holly Jolly Christmas	Winter Wonderland	0	131760	False	0.620	0.309	5	-9.209	1	0.0495	0.788	0.000000	0.1460	0.664	145.363	4	acoustic
28	Jason Mraz	Christmas Time	Winter Wonderland	0	131760	False	0.620	0.309	5	-9.209	1	0.0495	0.788	0.000000	0.1460	0.664	145.363	4	acoustic
29	Jason Mraz	Perfect Christmas Hits	Winter Wonderland	0	131760	False	0.620	0.309	5	-9.209	1	0.0495	0.788	0.000000	0.1460	0.664	145.363	4	acoustic
30	Jason Mraz	Merry Christmas	Winter Wonderland	0	131760	False	0.620	0.309	5	-9.209	1	0.0495	0.788	0.000000	0.1460	0.664	145.363	4	acoustic
31	Jason Mraz	Christmas Music - Holiday Hits	Winter Wonderland	0	131760	False	0.620	0.309	5	-9.209	1	0.0495	0.788	0.000000	0.1460	0.664	145.363	4	acoustic
33	Brandi Carlile;Sam Smith	Human - Best Adult Pop Tunes	Party of One	0	259558	False	0.296	0.206	0	-11.799	1	0.0412	0.782	0.000225	0.0959	0.202	165.400	4	acoustic
34	Brandi Carlile;Sam Smith	Feeling Good - Adult Pop Favorites	Party of One	0	259558	False	0.296	0.206	0	-11.799	1	0.0412	0.782	0.000225	0.0959	0.202	165.400	4	acoustic
35	Brandi Carlile;Sam Smith	Mellow Bars R'n'B	Party of One	0	259558	False	0.296	0.206	0	-11.799	1	0.0412	0.782	0.000225	0.0959	0.202	165.400	4	acoustic
36	KT Tunstall	Chill Christmas Dinner	Lonely This Christmas	0	257493	False	0.409	0.153	6	-10.740	0	0.0306	0.939	0.000026	0.1080	0.180	85.262	4	acoustic
39	KT Tunstall	sadsadchristmas	Lonely This Christmas	0	257493	False	0.409	0.153	6	-10.740	0	0.0306	0.939	0.000026	0.1080	0.180	85.262	4	acoustic

In [13]:

```
# same track with different tempo??
dup_cols = list(df_raw.columns)
dup_cols.remove('tempo')
df_raw[df_raw.duplicated(subset=dup_cols, keep=False)].head(10)
```

Out[13]:

	artists	album_name	track_name	popularity	duration_ms	explicit	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	time_signature	track_genre
59208	AMONGST THE ASHES	Agonizing Awakening	Exordium of Sickness	0	80948	False	0.423	0.853	1	-10.133	1	0.0382	0.00044	0.69	0.145	0.107	89.980	4	iranian
59916	AMONGST THE ASHES	Agonizing Awakening	Exordium of Sickness	0	80948	False	0.423	0.853	1	-10.133	1	0.0382	0.00044	0.69	0.145	0.107	89.977	4	iranian

In [14]:

```
# it is fair to remove their duplicates
dup_cols = list(df_raw.columns)
dup_cols.remove('track_name')
df_raw.drop_duplicates(subset=dup_cols, keep='first', inplace=True)

dup_cols = list(df_raw.columns)
dup_cols.remove('album_name')
df_raw.drop_duplicates(subset=dup_cols, keep='first', inplace=True)
```

```
dup_cols = list(df_raw.columns)
dup_cols.remove('tempo')
df_raw.drop_duplicates(subset=dup_cols, keep='first', inplace=True)
```

In [15]:
```
# No negative values except for "loudness" which is reasonable since decibels can be negative
(df_raw.select_dtypes(exclude='object')<0).any()
```

Out[15]:
```
popularity          False
duration_ms         False
explicit            False
danceability        False
energy              False
key                 False
loudness             True
mode                False
speechiness         False
acousticness        False
instrumentalness    False
liveness            False
valence             False
tempo               False
time_signature      False
dtype: bool
```

We are now left with 106811 observations after removing duplicates and obervations with missing values.

In [16]:
```
df_raw.shape
```

Out[16]:
```
(106811, 19)
```

Handling the categorical features:

The features `artists` , `album_name` , and `track_name` cannot be encoded by category in a meaningful way. Since analyzing text is out of scope for this project, we will not be considering these features.

In [17]:
```
# Unique name values
print("artists: ",df_raw.artists.nunique(), "/",106811)
print("album_name: ",df_raw.album_name.nunique(), "/",106811)
print("track_name: ",df_raw.track_name.nunique(), "/",106811)
```

```
artists:  31437 / 106811
album_name:  46512 / 106811
track_name:  73607 / 106811
```

The features `popularity` , `explicit` , `key` , `mode` , and `time_signature` are categorical variables given by numerical values. We will one-hot encode the features `explicit` and `mode` since they are nominal, meaning, their categories lack a natural order (`mode` is already encoded). On the other hand, `popularity` , `key` , and `time_signature` are quasi-interval variables so we will leave them alone to preserve their natural order.

In [18]:
```
# Encode 'explicit' as 0 or 1
label_encoder = preprocessing.LabelEncoder()
df_raw['explicit']=label_encoder.fit_transform(df_raw['explicit'])
df_raw.sample(n=20)
```

Out[18]:

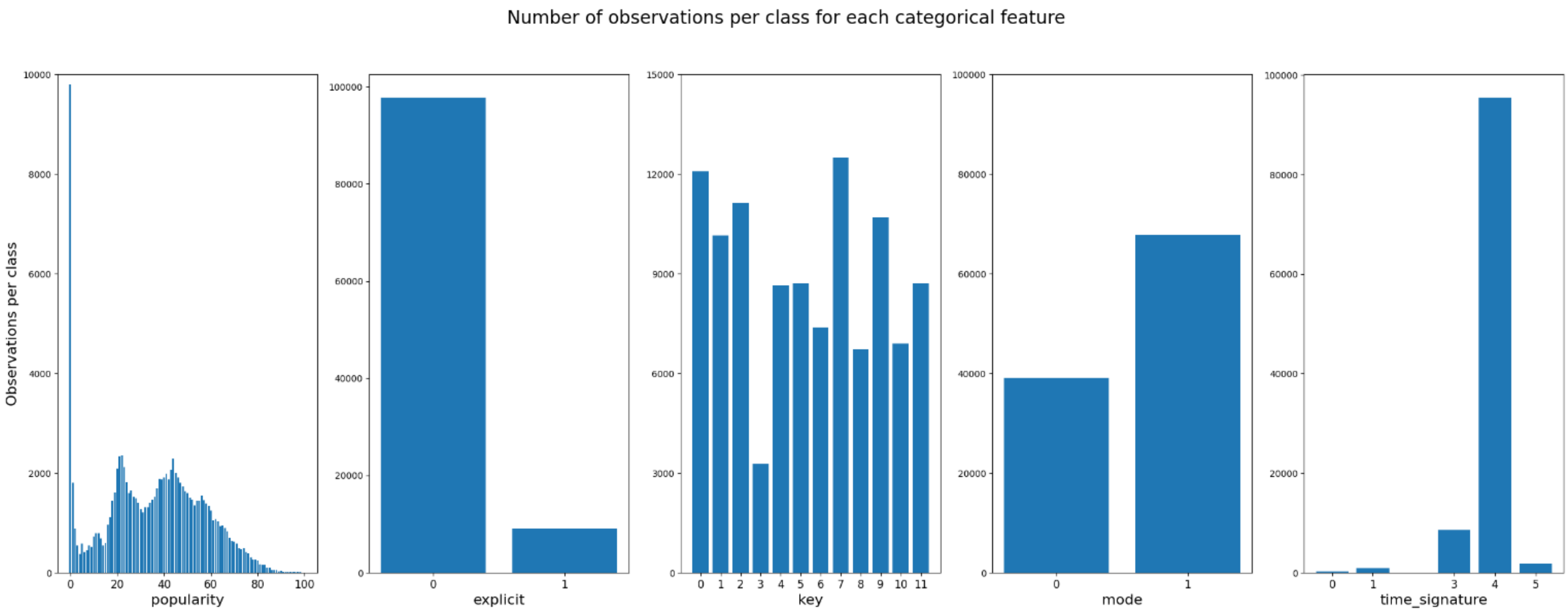
	artists	album_name	track_name	popularity	duration_ms	explicit	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	time_signature	tra
23084	Mark Knight;Armand Van Helden	Toolroom Amsterdam 2022	The Music Began To Play	5	145984	0	0.738	0.936	4	-3.451	0	0.2190	0.001510	0.707000	0.6170	0.2700	127.022	4	d
54092	Rival Consoles	Now Is	Running	39	251188	0	0.584	0.613	11	-13.823	1	0.0418	0.113000	0.855000	0.1080	0.0511	118.883	4	
35937	Canindé	Ao Vivo	Borbulhas de Amor (Tenho um Coração) - Burbujas de Amor	37	232266	0	0.815	0.571	0	-5.032	1	0.0302	0.308000	0.000000	0.9130	0.6150	120.093	4	
75741	Paul Cardall	Sacred Piano	Redeemer	20	352373	0	0.218	0.107	7	-18.676	1	0.0384	0.912000	0.398000	0.1180	0.0392	134.568	4	
57833	Maisie Peters	All Bops	This Is on You	0	195253	1	0.715	0.508	7	-6.899	1	0.0889	0.560000	0.000000	0.1240	0.2740	95.981	4	
9628	Aline Barros;Comunidade Evangélica Internacional da Zona Sul	Rompendo em Fé	Rompendo em Fé	44	289459	0	0.436	0.673	9	-4.640	1	0.0311	0.348000	0.000000	0.1100	0.2320	133.844	4	
48257	A Tribe Called Quest	People's Instinctive Travels and the Paths of Rhythm (25th Anniversary Edition)	Can I Kick It?	70	251573	0	0.848	0.666	0	-6.547	1	0.2740	0.173000	0.000699	0.1290	0.7440	96.662	4	
85659	ELLEGARDEN	ジターバグ	Cakes And Ale And Everlasting Laugh	36	184173	0	0.382	0.945	2	-2.897	1	0.0670	0.000486	0.000000	0.0481	0.3140	154.960	4	
87555	Terno Rei	Violeta	São Paulo	43	171000	0	0.657	0.436	4	-9.036	0	0.0306	0.267000	0.000258	0.1110	0.2650	122.000	4	
7483	The Infamous Stringdusters	Silver Sky	Rockets	23	183686	0	0.526	0.576	4	-6.392	1	0.0286	0.233000	0.000056	0.1020	0.4240	119.924	3	
59809	From The Vastland	Mar-Tiya-Khvara	Mar-Tiya-Khvara	3	375500	0	0.273	0.923	8	-3.873	1	0.0955	0.000069	0.003300	0.0877	0.0416	119.302	4	
27957	Sub Focus;Alice Gold	Torus	Out The Blue	48	277840	0	0.423	0.912	8	-5.271	0	0.0478	0.003200	0.003430	0.2090	0.2610	174.021	4	
70809	Joker Xue	渡	像風一樣	47	255111	0	0.514	0.418	2	-9.053	1	0.0619	0.355000	0.000000	0.0604	0.1470	117.705	4	
32031	DJ Snake;Selena Gomez;Ozuna;Cardi B	Carte Blanche	Taki Taki (feat. Selena Gomez, Ozuna & Cardi B)	73	212500	1	0.842	0.801	8	-4.167	0	0.2280	0.157000	0.000005	0.0642	0.6170	95.881	4	
61219	Nogizaka46	帰り道は遠回りしたくなる	帰り道は遠回りしたくなる	26	269706	0	0.510	0.857	1	-2.715	1	0.0403	0.457000	0.000000	0.0794	0.6930	138.064	4	

	artists	album_name	track_name	popularity	duration_ms	explicit	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	time_signature	tra
96655	Tim Maia	Sufocante	Bons Momentos	38	292322	0	0.583	0.360	11	-13.415	0	0.0330	0.601000	0.000054	0.1140	0.2230	126.218	4	
56817	Nikhil D'Souza	Boss	Har Kisi Ko	39	304025	0	0.463	0.788	3	-6.523	1	0.0518	0.336000	0.000000	0.3370	0.3820	179.968	4	
85260	Joan Jett & the Blackhearts	Bad Reputation (Expanded Edition)	Bad Reputation	68	169186	0	0.378	0.974	6	-4.055	1	0.1940	0.001920	0.013900	0.0588	0.8240	203.715	4	
103988	Anita Baker	Jazz Ballads Classics	Body and Soul	42	342000	0	0.532	0.432	8	-10.481	0	0.0387	0.550000	0.000000	0.0754	0.2190	109.112	3	
43956	JADED;MIRAMAR	Overtime (Remixes)	Overtime (MIRAMAR Remix)	39	212957	0	0.773	0.897	2	-4.855	1	0.1390	0.003010	0.612000	0.1180	0.3720	123.978	4	

Additionally, we will check for class imbalance in these categorical features.

In [19]:
```
# Visualize the class imbalance in the categorical features
fig, axs = plt.subplots(1,5)
features = ['popularity','explicit','key', 'mode', 'time_signature']
col = 0
for i in features:
  val_count = df_raw[i].value_counts().rename_axis(i).reset_index(name='count')
  axs[col].bar(val_count[i], val_count['count'])
  axs[col].set_xlabel(i, fontsize=16)
  axs[col].set_xticks(val_count[i])
  axs[col].set_yticks(np.arange(0,106811,20000))
  axs[col].tick_params(axis='x', which='major', labelsize=12)

  if i=='popularity':
    axs[col].set_xticks(np.arange(0,101,20))
    axs[col].set_yticks(np.arange(0, 10001, 2000))
    axs[col].set_ylim(0, 10000)

  if i == 'key':
    axs[col].set_yticks(np.arange(0, 15001, 3000))
    axs[col].set_ylim(0, 15000)

  col += 1


axs[0].set_ylabel('Observations per class', fontsize=16)
plt.suptitle("Number of observations per class for each categorical feature", fontsize=20)
fig.set_figwidth(30)
fig.set_figheight(10)
fig.show()
```

```
/var/folders/h8/frp0f1bd0v32l04p93kbg3f00000gn/T/ipykernel_12674/2397168134.py:29: UserWarning: Matplotlib is currently using module://matplotlib_inline.backend_inline, which is a non-G
UI backend, so cannot show the figure.
  fig.show()
```



Summary of class imbalances:

`popularity` : On a scale from 0 to 100, majority of the tracks were labeled as 0.

`explicit` : The non-explicit (`0`) class significantly dominates explicit (`1`) class.

`key` : Relatively balanced.

`mode` : The major scale (`1`) class somewhat dominates the minor scale (`0`) class.

`time_signature` : Tracks with time signature 4/4 (`4`) significantly outnumbers rest, followed by the time signature 3/4 (`3`).

Examine the reponse variable `track_genre` :

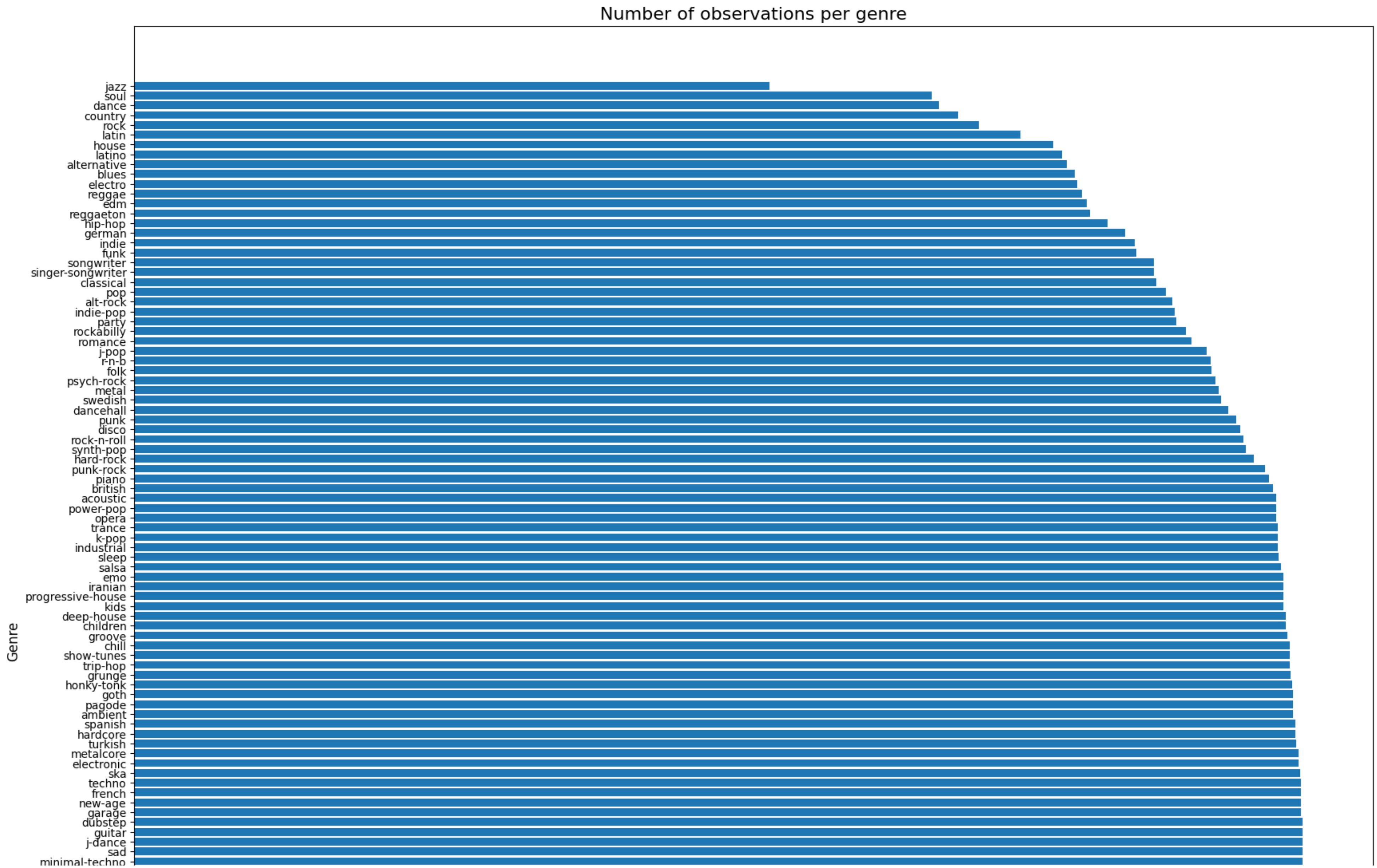
In [ ]:
```
df_raw.track_genre.nunique()
```

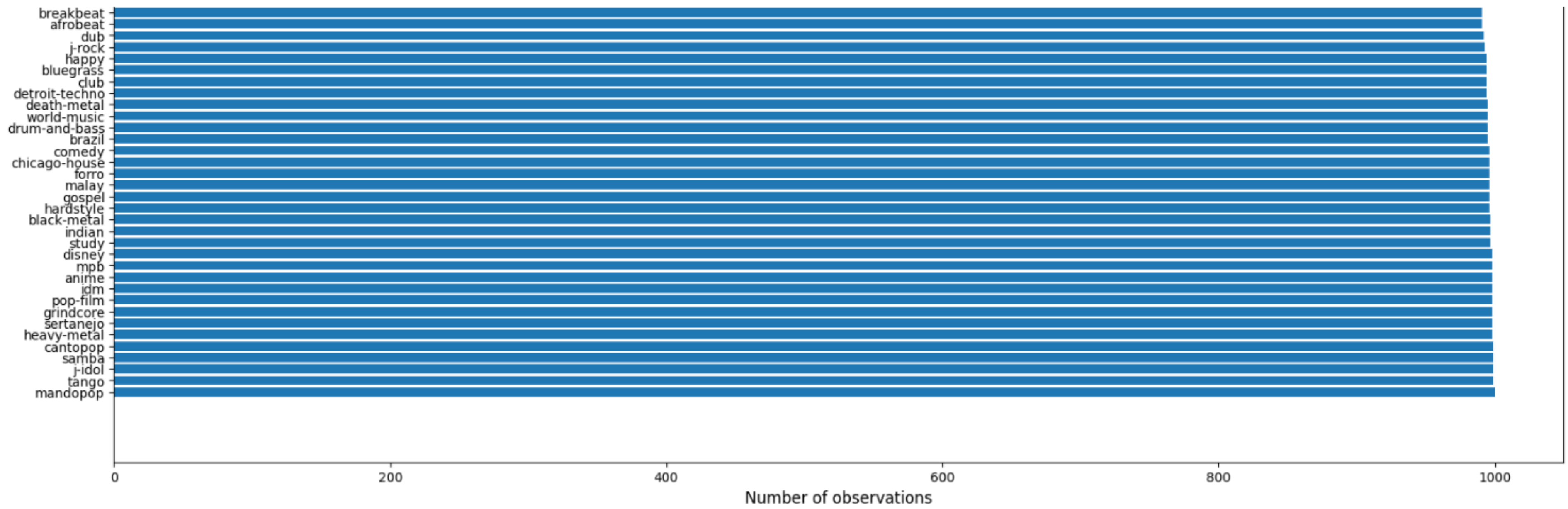
Out[ ]: 114

In [ ]:
```
df_raw.track_genre.unique()
```

Out[ ]:
```
array(['acoustic', 'afrobeat', 'alt-rock', 'alternative', 'ambient',
       'anime', 'black-metal', 'bluegrass', 'blues', 'brazil',
       'breakbeat', 'british', 'cantopop', 'chicago-house', 'children',
       'chill', 'classical', 'club', 'comedy', 'country', 'dance',
       'dancehall', 'death-metal', 'deep-house', 'detroit-techno',
       'disco', 'disney', 'drum-and-bass', 'dub', 'dubstep', 'edm',
       'electro', 'electronic', 'emo', 'folk', 'forro', 'french', 'funk',
       'garage', 'german', 'gospel', 'goth', 'grindcore', 'groove',
       'grunge', 'guitar', 'happy', 'hard-rock', 'hardcore', 'hardstyle',
       'heavy-metal', 'hip-hop', 'honky-tonk', 'house', 'idm', 'indian',
       'indie-pop', 'indie', 'industrial', 'iranian', 'j-dance', 'j-idol',
       'j-pop', 'j-rock', 'jazz', 'k-pop', 'kids', 'latin', 'latino',
       'malay', 'mandopop', 'metal', 'metalcore', 'minimal-techno', 'mpb',
       'new-age', 'opera', 'pagode', 'party', 'piano', 'pop-film', 'pop',
       'power-pop', 'progressive-house', 'psych-rock', 'punk-rock',
       'punk', 'r-n-b', 'reggae', 'reggaeton', 'rock-n-roll', 'rock',
       'rockabilly', 'romance', 'sad', 'salsa', 'samba', 'sertanejo',
       'show-tunes', 'singer-songwriter', 'ska', 'sleep', 'songwriter',
       'soul', 'spanish', 'study', 'swedish', 'synth-pop', 'tango',
       'techno', 'trance', 'trip-hop', 'turkish', 'world-music'],
      dtype=object)
```

In [ ]:
```
# Visualize any class imbalance in the response variable 'track_genre
from matplotlib import container

fig, ax = plt.subplots()

val_count = df_raw['track_genre'].value_counts().rename_axis('track_genre').reset_index(name='count')
ax.barh(val_count['track_genre'], val_count['count'])
ax.set_ylabel('Genre', fontsize=12)
ax.set_xlabel('Number of observations', fontsize=12)
ax.tick_params(axis='both', which='major', labelsize=10)

ax.set_title("Number of observations per genre", fontsize=16)
fig.set_figwidth(20)
fig.set_figheight(20)
fig.show()
```


Number of observations per genre
Genre
jazz
soul
dance
country
rock
latin
house
latino
alternative
blues
electro
reggae
edm
reggaeton
hip-hop
german
indie
funk
songwriter
singer-songwriter
classical
pop
alt-rock
indie-pop
party
rockabilly
romance
j-pop
r-n-b
folk
psych-rock
metal
swedish
dancehall
punk
disco
rock-n-roll
synth-pop
hard-rock
punk-rock
piano
british
acoustic
power-pop
opera
trance
k-pop
industrial
sleep
salsa
emo
iranian
progressive-house
kids
deep-house
children
groove
chill
show-tunes
trip-hop
grunge
honky-tonk
goth
pagode
ambient
spanish
hardcore
turkish
metalcore
electronic
ska
techno
french
new-age
garage
dubstep
guitar
j-dance
sad
minimal-techno



In [ ]:

```
pd.set_option('display.max_rows', None)
df_raw['track_genre'].value_counts()
```

Out[ ]:

track_genre	count
mandopop	1000
tango	999
j-idol	999
samba	999
cantopop	999
heavy-metal	998
sertanejo	998
grindcore	998
pop-film	998
idm	998
anime	998
mpb	998
disney	998
study	997
indian	997
black-metal	997
hardstyle	996
gospel	996
malay	996
forro	996
chicago-house	996
comedy	996
brazil	995
drum-and-bass	995
world-music	995
death-metal	995
detroit-techno	994
club	994
bluegrass	994
happy	994
j-rock	993

track_genre	count
dub	992
afrobeat	991
breakbeat	991
minimal-techno	990
sad	990
j-dance	990
guitar	990
dubstep	990
garage	989
new-age	989
french	989
techno	989
ska	988
electronic	987
metalcore	987
turkish	985
hardcore	984
spanish	984
ambient	982
pagode	982
goth	982
honky-tonk	981
grunge	980
trip-hop	979
show-tunes	979
chill	979
groove	977
children	976
deep-house	976
kids	974
progressive-house	974

	count
track_genre	
iranian	974
emo	974
salsa	972
sleep	970
industrial	969
k-pop	969
trance	969
opera	968
power-pop	968
acoustic	968
british	965
piano	962
punk-rock	958
hard-rock	949
synth-pop	942
rock-n-roll	940
disco	937
punk	934
dancehall	927
swedish	921
metal	919
psych-rock	916
folk	913
r-n-b	912
j-pop	909
romance	896
rockabilly	891
party	883
indie-pop	882
alt-rock	880
pop	874

track_genre	count
classical	866
singer-songwriter	864
songwriter	864
funk	849
indie	848
german	840
hip-hop	825
reggaeton	810
edm	807
reggae	803
electro	799
blues	797
alternative	790
latino	786
house	779
latin	751
rock	716
country	698
dance	682
soul	676
jazz	538

dtype: int64

The data is somewhat imbalanced in terms of the response variable `track_genre`. However, the ratio between the smallest class ("jazz") and largest class ("mandopop") is 538:1000, which is relatively acceptable rate.

In preparation for EDA, we'll make a new copy of the data set with the columns `album_name`, `track_name`, and `artists` dropped and will keep only observations belonging to the top 20 genres by count in the data in order to preserve the amount of data we have to work with and allow for more robust and effective classification of track genre. With 114 classes and 15 features, most models are unable to accurately classify observations.

In [20]:
```
# Isolate the numerical data (i.e. drop the album, track, and artist names) and the response variable 'track_genre'
df_raw = df_raw.drop(columns=['album_name','track_name','artists'])
top20 = df_raw['track_genre'].value_counts(ascending=False)[:20].index
df_raw = df_raw[df_raw['track_genre'].isin(top20)]
df_raw.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 19955 entries, 5000 to 108999
Data columns (total 16 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   popularity        19955 non-null  int64
 1   duration_ms       19955 non-null  int64
 2   explicit          19955 non-null  int64
 3   danceability      19955 non-null  float64
 4   energy            19955 non-null  float64
 5   key               19955 non-null  int64
 6   loudness          19955 non-null  float64
 7   mode              19955 non-null  int64
 8   speechiness       19955 non-null  float64
 9   acousticness      19955 non-null  float64
 10  instrumentalness  19955 non-null  float64
 11  liveness          19955 non-null  float64
 12  valence           19955 non-null  float64
 13  tempo             19955 non-null  float64
 14  time_signature    19955 non-null  int64
 15  track_genre       19955 non-null  object
dtypes: float64(9), int64(6), object(1)
memory usage: 2.6+ MB
```

In [21]:

```
# Create a standardized version of the data for modeling purposes after EDA
ss = preprocessing.StandardScaler()
df = df_raw.copy()
num_cols = df.drop(columns='track_genre').columns
df[num_cols] = ss.fit_transform(df.drop(columns='track_genre'))
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 19955 entries, 5000 to 108999
Data columns (total 16 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   popularity        19955 non-null  float64
 1   duration_ms       19955 non-null  float64
 2   explicit          19955 non-null  float64
 3   danceability      19955 non-null  float64
 4   energy            19955 non-null  float64
 5   key               19955 non-null  float64
 6   loudness          19955 non-null  float64
 7   mode              19955 non-null  float64
 8   speechiness       19955 non-null  float64
 9   acousticness      19955 non-null  float64
 10  instrumentalness  19955 non-null  float64
 11  liveness          19955 non-null  float64
 12  valence           19955 non-null  float64
 13  tempo             19955 non-null  float64
 14  time_signature    19955 non-null  float64
 15  track_genre       19955 non-null  object
dtypes: float64(15), object(1)
memory usage: 2.6+ MB
```

In [22]:
```
# Create a scaled version of the data so that all values are between -1 and 1
mm = preprocessing.MinMaxScaler()
df_mm = df_raw.copy()
num_cols = df_mm.drop(columns='track_genre').columns
df_mm[num_cols] = mm.fit_transform(df_mm.drop(columns='track_genre'))
df_mm.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 19955 entries, 5000 to 108999
Data columns (total 16 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   popularity        19955 non-null  float64
 1   duration_ms       19955 non-null  float64
 2   explicit          19955 non-null  float64
 3   danceability      19955 non-null  float64
 4   energy            19955 non-null  float64
 5   key               19955 non-null  float64
 6   loudness          19955 non-null  float64
 7   mode              19955 non-null  float64
 8   speechiness       19955 non-null  float64
 9   acousticness      19955 non-null  float64
 10  instrumentalness  19955 non-null  float64
 11  liveness          19955 non-null  float64
 12  valence           19955 non-null  float64
 13  tempo             19955 non-null  float64
 14  time_signature    19955 non-null  float64
 15  track_genre       19955 non-null  object 
dtypes: float64(15), object(1)
memory usage: 2.6+ MB
```

II. Exploratory Data Analysis

In [24]:
```
df.sample(10)
```