Spotify Exploratory Data Analysis using Python

By: Rakshit Bhadoria

Introduction

Spotify is a proprietary Swedish audio streaming and media services provider founded on April 23, 2006 by Daniel Ek and Martin Lorentzon. It is one of the largest music streaming service providers, with over 456 million monthly active users, including 195 million paying subscribers, as of September 2022.

In this project we will be exploring and quantifying data about music from Spotify, and drawing valuable insights.

This includes performing exploratory data analysis (EDA) and creating data visualizations using data scraped from Spotify.

This project will be done in Python, utilizing multiple packages including NumPy, Pandas, Matplotlib, and Seaborn.

Download Spotify datasets from Kaggle:

https://www.kaggle.com/datasets/lehaknarnauli/spotify-datasets?select=tracks.csv

https://www.kaggle.com/datasets/zaheenhamidani/ultimate-spotify-tracks-db?select=SpotifyFeatures.csv

Import libraries for data analysis

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Import the Spotify datasets

```
df_tracks = pd.read_csv('/Users/mattabruzzeseott/Documents/Portfolio Projects/Jupyter/tracks.csv')
df_genres = pd.read_csv('/Users/mattabruzzeseott/Documents/Portfolio Projects/Jupyter/SpotifyFeatures.csv')
```

Spotify Tracks Dataset Analysis

Preview the Dataset

```
df_tracks.head()
```

	id	name	popularity	duration_ms	explicit	artists	
0	35iwgR4jXetI318WEWsa1Q	Carve	6	126903	0	['Uli']	['45tIt0
1	021ht4sdgPcrDgSk7JTbKY	Capítulo 2.16 - Banquero Anarquista	0	98200	0	['Fernando Pessoa']	['14jtPCOoN
2	07A5yehtSnoedViJAZkNnc	Vivo para Quererte - Remasterizado	0	181640	0	['Ignacio Corsini']	['5LiOoJbxVS
3	08FmqUhxtyLTn6pAh6bk45	El Prisionero - Remasterizado	0	176907	0	['Ignacio Corsini']	['5LiOoJbxVS
4	08y9GfoqCWfOGsKdwojr5e	Lady of the Evening	0	163080	0	['Dick Haymes']	['3BiJGZsyX

Check for any null values present in the dataset.

```
pd.isnull(df_tracks).sum()
```

```
id                  0
name               71
popularity          0
duration_ms         0
explicit            0
```

```
artists              0
id_artists           0
release_date         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
dtype: int64
```

It can be seen that the song name column has 71 null values.

```
pd.isnull(df_tracks).sum().sum()
```

```
71
```

```
len(df_tracks)
```

```
586672
```

There are just 71 missing values in the entire dataset consisting of ~586k rows. Therefore the dataset is good, and our conclusions will be valid.

Inspect and identify the number of rows and columns in the dataset, and check the data type for each variable.

```
df_tracks.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 586672 entries, 0 to 586671
Data columns (total 20 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   id                586672 non-null  object 
 1   name              586601 non-null  object 
 2   popularity        586672 non-null  int64  
 3   duration_ms       586672 non-null  int64  
 4   explicit          586672 non-null  int64  
 5   artists           586672 non-null  object 
 6   id_artists        586672 non-null  object 
 7   release_date      586672 non-null  object 
 8   danceability      586672 non-null  float64
 9   energy            586672 non-null  float64
 10  key               586672 non-null  int64  
 11  loudness          586672 non-null  float64
 12  mode              586672 non-null  int64  
 13  speechiness       586672 non-null  float64
 14  acousticness      586672 non-null  float64
 15  instrumentalness  586672 non-null  float64
 16  liveness          586672 non-null  float64
 17  valence           586672 non-null  float64
 18  tempo             586672 non-null  float64
 19  time_signature    586672 non-null  int64  
dtypes: float64(9), int64(6), object(5)
memory usage: 89.5+ MB
```

Descriptive statistics for numerical values in the dataset.

```
df_tracks.describe().transpose()
```

	count	mean	std	min	25%	50%	75%	max
popularity	586672.0	27.570053	18.370642	0.0	13.0000	27.000000	41.00000	100.000
duration_ms	586672.0	230051.167286	126526.087418	3344.0	175093.0000	214893.000000	263867.00000	5621218.000
danceability	586672.0	0.563594	0.166103	0.0	0.4530	0.577000	0.68600	0.991

Identify the ten least popular songs in the dataset.

To obtain a list of the ten least popular songs, we will sort the "popularity" column in ascending order.

```
sorted_df = df_tracks.sort_values('popularity', ascending = True).head(10)
sorted_df
```

	id	name	popularity	duration_ms	explicit	artists	id_artists	release_date	danceability	energ
546130	181rTRhCcggZPwP2TUcVqm	Newspaper Reports On Abner, 20 February 1935	0	896575	0	['Norris Goff', 'Chester Lauck', 'Carlton Bric...	['3WCwCPDMpGzrt0Qz6quumy', '7vk8UqABg0Sga78GI3...	1935-02-20	0.595	0.26
546222	0yOCz3V5KMm8l1T8EFc60i	恋は水の上で	0	188440	0	['Hibari Misora']	['1m5pMY5blqJwdxJ7vqQtuN']	1949	0.418	0.38
546221	0y48Hhwe52099UqYjegRCO	私の誕生日	0	173467	0	['Hibari Misora']	['1m5pMY5blqJwdxJ7vqQtuN']	1949	0.642	0.17
546220	0xCmgtf9ka07hkZg3D6PaV	エル・チョクロ (EL CHOCLO)	0	205280	0	['Hibari Misora']	['1m5pMY5blqJwdxJ7vqQtuN']	1949	0.695	0.46
546219	0tBXS3VuCPX7KWUFH2nros	恋は不思議なもの	0	185733	0	['Hibari Misora']	['1m5pMY5blqJwdxJ7vqQtuN']	1949	0.389	0.38
546218	0qrKnQtYDVJhKFAXTHYVS9	ゆうべはどうしたの (WHATSA MALLA U)	0	183427	0	['Hibari Misora']	['1m5pMY5blqJwdxJ7vqQtuN']	1949	0.631	0.24
546217	0nqsDxOeKSwEzp3AUQAAqS	Screen Director's Playhouse, Music For Million...	0	1767071	0	['Wilms Herbert', 'June Allyson', 'Joseph Kear...	['2rbm8QWvmnVwxFo84EVM1h', '4yW5adMgyIfHFzaL9i...	1949-04-10	0.533	0.31
546216	0kGEdsxVLYjCdfxM9tbezd	ブルーマンボ	0	162147	0	['Hibari Misora']	['1m5pMY5blqJwdxJ7vqQtuN']	1949	0.529	0.54
546215	0bc3PUZurUUXrY7yqoOxjq	Screen Director's Playhouse, Trade Winds direc...	0	1776652	0	['Wally Maher', 'Tay Garnett', 'Lurene Tuttle'...	['7hkhJTTI3VnUGVWUt8SJXT', '3kYeeIpRCgJz4fQYDv...	1949-05-19	0.599	0.32
546214	0Wwm0ruSjYMIiWG0nyAI1F	Screen Director's Playhouse, It's A Wonderful ...	0	1767576	0	['Joseph Granby', 'Jimmy Stewart', 'Irene Tedr...	['6GK59BC4LJzqR0OpHAX2S3', '58BzBaExrnrx898sby...	1949-05-08	0.645	0.34

Identify the ten most popular songs in the dataset with "popularity" greater than 90.

```
most_popular = df_tracks.query('popularity > 90', inplace = False).sort_values('popularity', ascending = False)
most_popular[:10]
```

	id	name	popularity	duration_ms	explicit	artists	id_artists	release_date	danceability	energy
93802	4iJyoBOLtHqaGxP12qzhQI	Peaches (feat. Daniel Caesar & Giveon)	100	198082	1	['Justin Bieber', 'Daniel Caesar', 'Giveon']	['1uNFoZAHBGtllmzznpCI3s', '20wkVLutqVOYrc0kxF...	2021-03-19	0.677	0.696
93803	7lPN2DXiMsVn7XUKtOW1CS	drivers license	99	242014	1	['Olivia Rodrigo']	['1McMsnEElThX1knmY4oliG']	2021-01-08	0.585	0.436
93804	3Ofmpyhv5UAQ70mENzB277	Astronaut In The Ocean	98	132780	0	['Masked Wolf']	['1uU7g3DNSbsu0QjSEqZtEd']	2021-01-06	0.778	0.695

Make the "release_date" column the new index column.

```
df_tracks.set_index('release_date', inplace = True)
df_tracks.index = pd.to_datetime(df_tracks.index)
df_tracks.head()
```

release_date	id	name	popularity	duration_ms	explicit	artists	id_artists	danceability	energy	key
1922-02-22	35iwgR4jXetI318WEWsa1Q	Carve	6	126903	0	['Uli']	['45tIt06XoI0Iio4LBEVpls']	0.645	0.4450	0
1922-06-01	021ht4sdgPcrDgSk7JTbKY	Capítulo 2.16 - Banquero Anarquista	0	98200	0	['Fernando Pessoa']	['14jtPCOoNZwquk5wd9DxrY']	0.695	0.2630	0
1922-03-21	07A5yehtSnoedViJAZkNnc	Vivo para Quererte - Remasterizado	0	181640	0	['Ignacio Corsini']	['5LiOoJbxVSAMkBS2fUm3X2']	0.434	0.1770	1
1922-03-21	08FmqUhxtyLTn6pAh6bk45	El Prisionero - Remasterizado	0	176907	0	['Ignacio Corsini']	['5LiOoJbxVSAMkBS2fUm3X2']	0.321	0.0946	7
1922-01-01	08y9GfoqCWfOGsKdwojr5e	Lady of the Evening	0	163080	0	['Dick Haymes']	['3BiJGZsyX9sJchTqcSA7Su']	0.402	0.1580	3

Find the name of the artist in the 17th row of the dataset.

We can filter any specific information from the dataset using the index location method iloc[].

```
df_tracks[['artists']].iloc[17]
```

```
artists    ['Fernando Pessoa']
Name: 1922-06-01 00:00:00, dtype: object
```

Querying and sorting examples.

Example 1: Inspect the ten most popular songs in the updated dataset (new index).

```
most_popular = df_tracks.query('popularity > 90', inplace = False).sort_values('popularity', ascending = False)
most_popular[:10]
```

release_date	id	name	popularity	duration_ms	explicit	artists	id_artists	danceability	energy	key
2021-03-19	4iJyoBOLtHqaGxP12qzhQI	Peaches (feat. Daniel Caesar &	100	198082	1	['Justin Bieber', 'Daniel Caesar',	['1uNFoZAHBGtllmzznpCI3s', '20wkVLutqVOYrc0kxF...	0.677	0.696	0

Example 2: Sort the most popular songs by release date, and include only specific columns.

2021-01-08	7lPN2DXiMsVn7XUKtOW1CS	drivers	99	242014	1	['Olivia	['1McMsnEElThX1knmY4oliG']	0.585	0.436	10

```
pop_date = most_popular.sort_values('release_date', ascending = False)
pop_date[['name', 'artists', 'popularity', 'explicit']][:10]
```

release_date	name	artists	popularity	explicit
2021-03-19	Peaches (feat. Daniel Caesar & Giveon)	['Justin Bieber', 'Daniel Caesar', 'Giveon']	100	1
2021-03-05	Leave The Door Open	['Bruno Mars', 'Anderson .Paak', 'Silk Sonic']	96	0
2021-03-05	What's Next	['Drake']	91	1
2021-03-05	Hold On	['Justin Bieber']	92	0
2021-02-11	We're Good	['Dua Lipa']	91	0
2021-02-05	911	['Sech']	91	1
2021-02-05	Up	['Cardi B']	92	1
2021-02-04	Fiel	['Los Legendarios', 'Wisin', 'Jhay Cortez']	94	0
2021-02-03	Ella No Es Tuya - Remix	['Rochy RD', 'Myke Towers', 'Nicki Nicole']	92	0
2021-01-21	Wellerman - Sea Shanty / 220 KID x Billen Ted ...	['Nathan Evans', '220 KID', 'Billen Ted']	92	0
	Heartbreak			

Example 3: Sort the top ten non-explicit songs with "popularity" greater than 85.

```
most_pop_non_ex = df_tracks.query('(popularity > 85) and (explicit == 0)').sort_values('popularity',
                  ascending = False)
most_pop_non_ex[:10]
```

release_date	id	name	popularity	duration_ms	explicit	artists	id_artists	danceability	energy
2021-01-06	3Ofmpyhv5UAQ70mENzB277	Astronaut In The Ocean	98	132780	0	['Masked Wolf']	['1uU7g3DNSbsu0QjSEqZtEd']	0.778	0.695
2020-12-04	6tDDoYIxWvMLTdKpjFkc1B	telepatía	97	160191	0	['Kali Uchis']	['1U1el3k54VvEUzo3ybLPlM']	0.653	0.524
2021-03-05	7MAibcTli4IisCtbHKrGMh	Leave The Door Open	96	242096	0	['Bruno Mars', 'Anderson .Paak', 'Silk Sonic']	['0du5cEVh5yTK9QJze8zA0C', '3jK9MiCrA42lLAdMGU...	0.586	0.616
2020-03-20	0VjIjW4GlUZAMYd2vXMi3b	Blinding Lights	96	200040	0	['The Weeknd']	['1Xyo4u8uXC1ZmMpatF05PJ']	0.514	0.730
2020-09-16	6f3Slt0GbA2bPZlz0aIFXN	The Business	95	164000	0	['Tiësto']	['2o5jDhtHVPhrJdv3cEQ99Z']	0.798	0.620
2021-02-04	7Bk0uXKk1uPT0XuQbpFzvs	Fiel	94	261667	0	['Los Legendarios', 'Wisin', 'Jhay Cortez']	['0n6sKrG0xKAf8xmdqeNGke', '3E6xrwgnVfYCrCs0eP...	0.849	0.701
2021-01-15	4cG7HUWYHBV6R6tHn1gxrl	Friday (feat. Mufasa & Hypeman) - Dopamine Re-...	94	169153	0	['Riton', 'Nightcrawlers', 'Mufasa & Hypeman',...	['7i9j813KFoSBMldGqlh2Z1', '1gALaWbNDnwS2ECV09...	0.824	0.862
2020-12-10	1xK1Gg9SxG8fy2Ya373oqb	Bandido	94	232853	0	['Myke Towers', 'Juhn']	['7iK8PXO48WeuP03g8YR51W', '2LmcxBak1alK1bf7d1...	0.713	0.617
2020-03-27	3FAJ6O0NOHQV8Mc5Ri6ENp	Heartbreak Anniversary	94	198371	0	['Giveon']	['4fxd5Ee7UefO4CUXgwJ7IP']	0.449	0.465
2020-11-27	2XIc1pqjXV3Cr2BQUGNBck	LA NOCHE DE ANOCHE	93	203201	0	['Bad Bunny', 'ROSALÍA']	['4q3ewBCX7sLwd24euuV69X', '7ltDVBr6mKbRvohxhe...	0.856	0.618

Convert the song duration column in the Tracks dataframe from Milliseconds to Seconds.

```
df_tracks['duration'] = df_tracks['duration_ms'].apply(lambda x: round(x/1000))
df_tracks.drop('duration_ms', inplace = True, axis = 1)
df_tracks.duration.head()
```

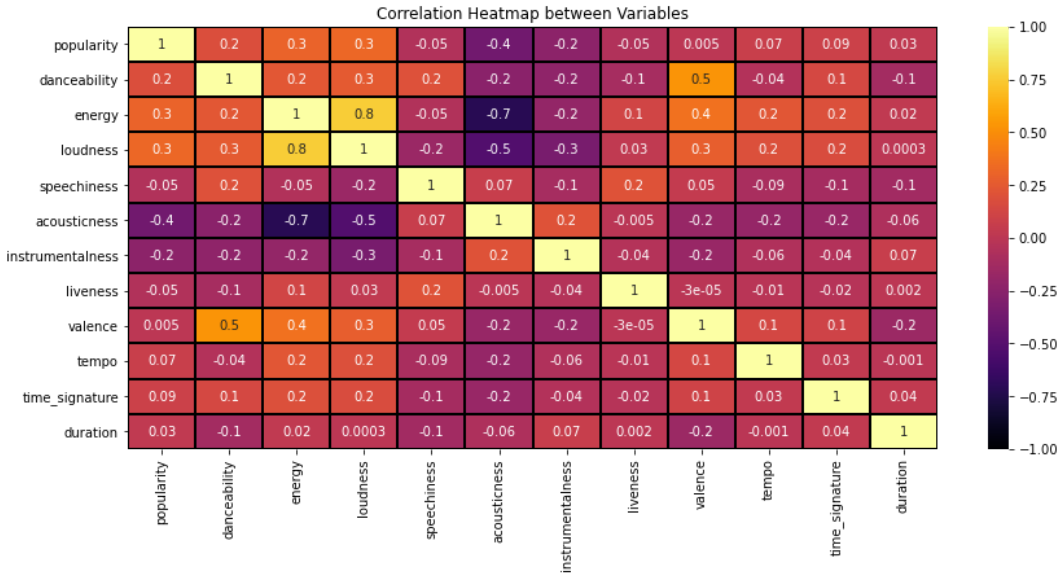
```
release_date
1922-02-22    127
1922-06-01     98
1922-03-21    182
1922-03-21    177
1922-01-01    163
Name: duration, dtype: int64
```

Data Visualizations

Correlation Heatmap between Variables using Pearson Correlation

We will drop three unwanted variables, key, mode, and explicit. We will set the figure size of the correlation map to (14,6), and use the heatmap() function to create the correlation map. Setting the "annotation = True" parameter will place the data value in each cell on the map.

```
corr_df = df_tracks.drop(['key', 'mode', 'explicit'], axis = 1).corr(method = 'pearson')
plt.figure(figsize = (14,6))
heatmap = sns.heatmap(corr_df, annot = True, fmt = '.1g', vmin = -1, vmax = 1, center = 0, cmap = 'inferno',
          linewidths = 1, linecolor = 'Black')
heatmap.set_title('Correlation Heatmap between Variables')
heatmap.set_xticklabels(heatmap.get_xticklabels(), rotation = 90)
ticks = heatmap.set_xticklabels(heatmap.get_xticklabels(), rotation = 90)
```



The scale for this correlation map ranges from -1 to 1. Values between -1 and 0 denote variables that have a negative correlation (least correlated). Values between 0 and 1 denote variables that have a positive correlation (most correlated).

r > 0 implies positive correlation
r < 0 implies negative correlation
r = 0 implies no correlation

The Spotify Tracks dataset is very large (586672 rows), so we will sample a small percentage of the entire dataset (0.4%) for future calculations and visualizations.

```
print(len(df_tracks))
```

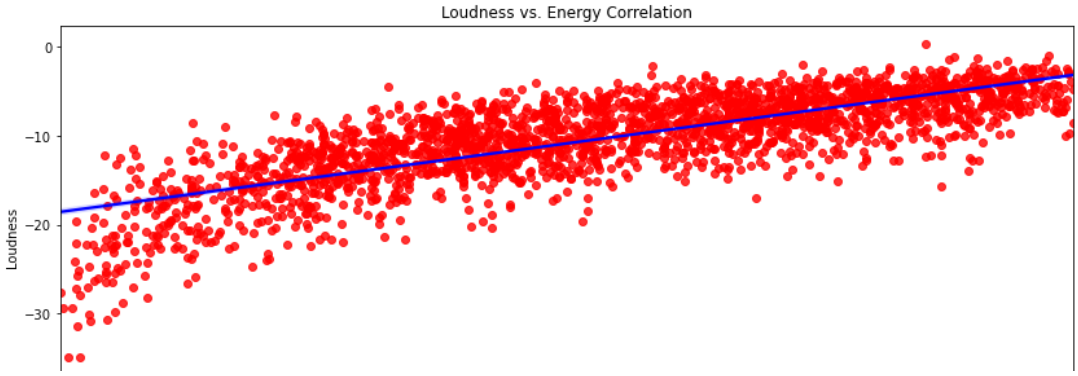
```
586672
```

```
sample = df_tracks.sample(int(0.004 * len(df_tracks)))
print(len(sample))
```

```
2346
```

Linear Regression Plot - Correlation between Loudness and Energy

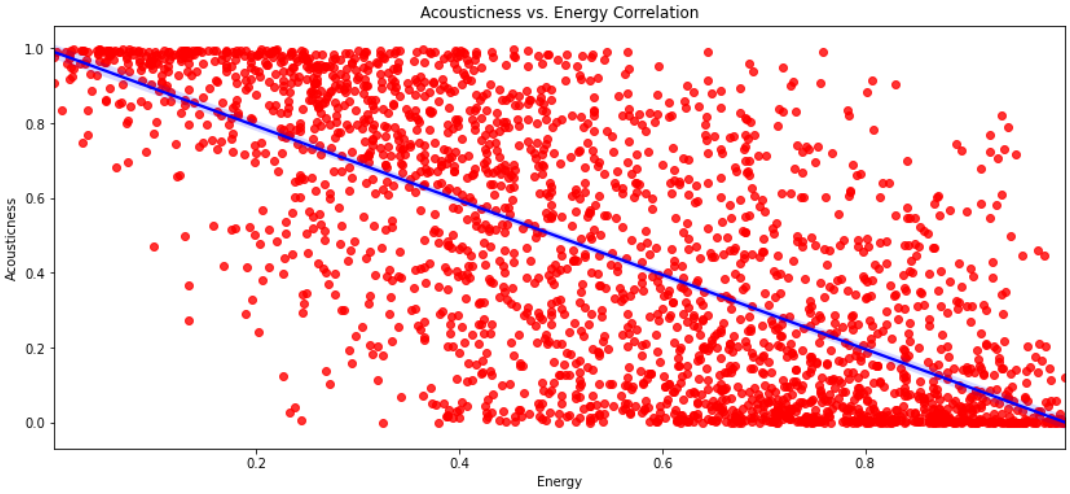
```
plt.figure(figsize = (14,6))
sns.regplot(data = sample, y = 'loudness', x = 'energy', scatter_kws = {'color':'red'},
            line_kws = {'color':'blue'}).set(title = 'Loudness vs. Energy Correlation', xlabel = 'Energy',
            ylabel = 'Loudness');
```



This regression plot visualizes the positive correlation between the "loudness" and "energy" variables (r = 0.8).

Linear Regression Plot - Correlation between Acousticness and Energy

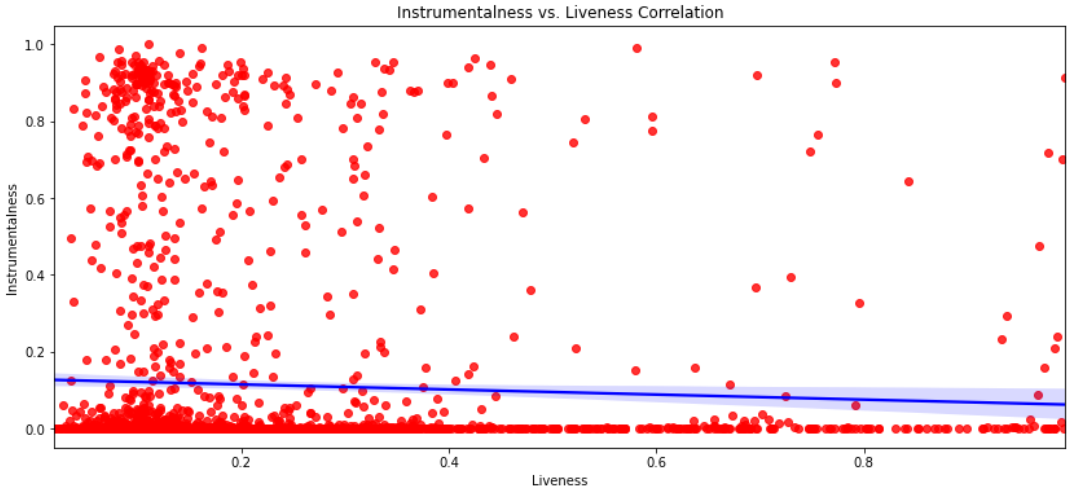
```
plt.figure(figsize = (14,6))
sns.regplot(data = sample, y = 'acousticness', x = 'energy', scatter_kws = {'color':'red'},
            line_kws = {'color':'blue'}).set(title = 'Acousticness vs. Energy Correlation', xlabel = 'Energy',
            ylabel = 'Acousticness');
```



This regression plot visualizes the negative correlation between the "acousticness" and "energy" variables (r = -0.7).

Linear Regression Plot - Correlation between Instrumentalness and Liveness

```
plt.figure(figsize = (14,6))
sns.regplot(data = sample, y = 'instrumentalness', x = 'liveness', scatter_kws = {'color':'red'},
            line_kws = {'color':'blue'}).set(title = 'Instrumentalness vs. Liveness Correlation', xlabel = 'Liveness',
            ylabel = 'Instrumentalness');
```



This regression plot visualizes the lack of correlation between the "instrumentalness" and "liveness" variables (r = -0.04).

Extract the "year" value from each song's release date and store these "year" values in a new variable.

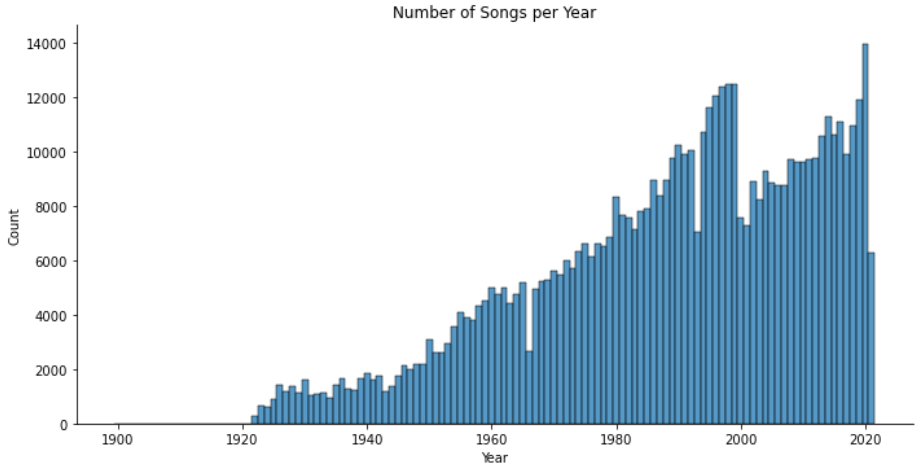
```
df_tracks['date'] = df_tracks.index.get_level_values('release_date')
df_tracks.date = pd.to_datetime(df_tracks.date)
df_tracks['year'] = df_tracks.date.dt.year
df_tracks.head()
```

release_date	id	name	popularity	explicit	artists	id_artists	danceability	energy	key	loudness	..
1922-02-22	35iwgR4jXetI318WEWsa1Q	Carve	6	0	['Uli']	['45tIt06XoI0Iio4LBEVpls']	0.645	0.4450	0	-13.338	
1922-06-01	021ht4sdgPcrDgSk7JTbKY	Capítulo 2.16 - Banquero Anarquista	0	0	['Fernando Pessoa']	['14jtPCOoNZwquk5wd9DxrY']	0.695	0.2630	0	-22.136	
1922-03-21	07A5yehtSnoedViJAZkNnc	Vivo para Quererte - Remasterizado	0	0	['Ignacio Corsini']	['5LiOoJbxVSAMkBS2fUm3X2']	0.434	0.1770	1	-21.180	
1922-03-21	08FmqUhxtyLTn6pAh6bk45	El Prisionero - Remasterizado	0	0	['Ignacio Corsini']	['5LiOoJbxVSAMkBS2fUm3X2']	0.321	0.0946	7	-27.961	
1922-01-01	08y9GfoqCWfOGsKdwojr5e	Lady of the Evening	0	0	['Dick Haymes']	['3BiJGZsyX9sJchTqcSA7Su']	0.402	0.1580	3	-16.900	

5 rows × 21 columns

Distribution Plot - Visualize the Total Number of Songs on Spotify by Year

```
dist_plot = sns.displot(data = df_tracks, x = 'year', discrete = True, aspect = 2, height = 5, kind = 'hist')
dist_plot.set(title = 'Number of Songs per Year');
dist_plot.set_axis_labels('Year', 'Count');
```




```
song_count = df_tracks['year'].value_counts()
song_count
```

```
2020    13937
1998    12485
1999    12484
1997    12349
1996    12058
        ...
1925      903
1923      657
1924      633
1922      294
1900        1
Name: year, Length: 101, dtype: int64
```

It can be visualized through the distribution plot, and confirmed by the song count above that the year 1900 only has one song in the dataset, and can be viewed as an outlier. For the next part of the analysis, we will be working with the means of the variables in the dataset. This single song would skew our results when analyzing years prior to 1922, therefore we will remove this song and its values.

Remove Outlier

```
df_tracks = df_tracks.drop(df_tracks[df_tracks.year == 1900].index)
df_tracks.sort_values('year', ascending = True).head()
```

release_date	id	name	popularity	explicit	artists	id_artists	danceability	energy	key	loudness	...
1922-02-22	35iwgR4jXetI318WEWsa1Q	Carve	6	0	['Uli']	['45tIt06XoI0Iio4LBEVpls']	0.645	0.445	0	-13.338	...
1922-01-01	3AwlEhAkDDwKuTaNlgmMNQ	Nola	0	0	['Vincent Lopez and his Orchestra', 'Vincent L...	['1NElogFmaZxxGVsKS6hvl2', '3wxzXhMAoYbpJDXtBx...	0.567	0.663	2	-5.334	...
1922-01-01	32Y9PU9JqxYFqzFaIdCQOs	Midnight rose	0	0	['Abe Lyman's Orchestra', 'Abe Lyman']	['6LxnbCQ3ZrKj1lvC1lylS5', '3cNzWID6yZ1HN8qj4g...	0.483	0.060	1	-9.499	...
1922-01-01	2zRV6Vk6ZQYDokmiv5QEoP	California blues	0	0	['Abe Lyman's Orchestra', 'Abe Lyman']	['6LxnbCQ3ZrKj1lvC1lylS5', '3cNzWID6yZ1HN8qj4g...	0.578	0.462	8	-7.217	...

```
print(len(df_tracks))
```

```
586671
```

The row of the outlier song from 1900 has been removed from the dataset.

Create a new dataframe with the mean of each column grouped by year.

```
mean_df = df_tracks.groupby(['year']).mean()
mean_df
```

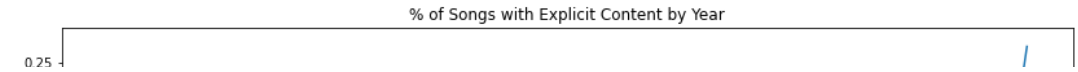
year	popularity	explicit	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	t
1922	0.054422	0.000000	0.533320	0.324054	4.850340	-13.953241	0.738095	0.246295	0.898857	0.324971	0.250670	0.563605	109.12
1923	1.575342	0.001522	0.637332	0.266977	3.651446	-16.351921	0.797565	0.552072	0.859965	0.157659	0.225396	0.671967	109.55
1924	0.612954	0.001580	0.593344	0.356725	4.723539	-13.290367	0.693523	0.375208	0.866266	0.339628	0.203447	0.554935	119.45
1925	1.414175	0.000000	0.617391	0.263749	5.069767	-14.977595	0.764120	0.305693	0.912170	0.275384	0.255157	0.635196	113.90
1926	1.938776	0.000000	0.622113	0.263075	5.746657	-15.929906	0.705841	0.356952	0.785739	0.323416	0.211447	0.539702	113.38
...	...	...	...	...	...	...	...	...	...	...	...	...	
2017	42.236222	0.170998	0.624972	0.659730	5.378805	-7.215648	0.583274	0.100517	0.278547	0.089228	0.200736	0.493965	121.33
2018	42.139539	0.189923	0.634510	0.652202	5.371617	-7.390117	0.566843	0.108212	0.278446	0.098398	0.198216	0.483251	121.68
2019	44.920551	0.214916	0.649724	0.637066	5.369615	-7.434020	0.567565	0.111168	0.293902	0.081428	0.189971	0.494212	122.02
2020	44.686948	0.217981	0.657129	0.639669	5.306809	-7.533985	0.562531	0.112279	0.271750	0.113089	0.192968	0.501712	122.36
2021	35.191848	0.260627	0.671531	0.617998	5.269543	-7.898172	0.520936	0.130416	0.278645	0.119411	0.172572	0.506228	120.75

100 rows × 15 columns

Has music become more explicit over time?

The "explicit" variable is binary, showing a "1" for songs that contain explicit content, and a "0" for songs that do not contain explicit content.

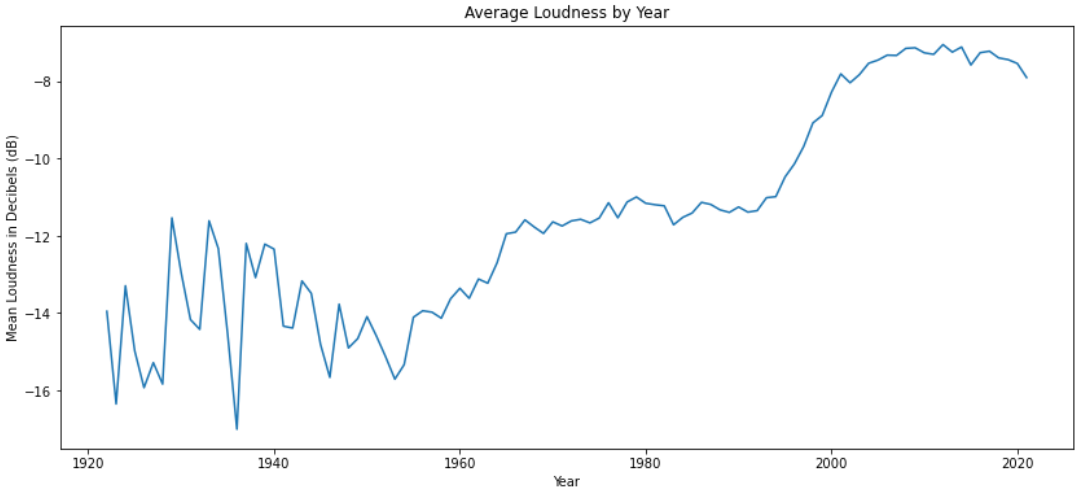
```
explicit_trend = mean_df['explicit'].plot.line(figsize = (14,6))
explicit_trend.set_xlabel('Year');
explicit_trend.set_ylabel('Mean Explicit Value');
explicit_trend.set_title('% of Songs with Explicit Content by Year');
```



This line chart shows that music is much more explicit today, and there was a large increase in explicit content from 1990 to 2020.

Has music become louder over time?

```
loudness_trend = mean_df['loudness'].plot.line(figsize = (14,6))
loudness_trend.set_xlabel('Year');
loudness_trend.set_ylabel('Mean Loudness in Decibels (dB)');
loudness_trend.set_title('Average Loudness by Year');
```

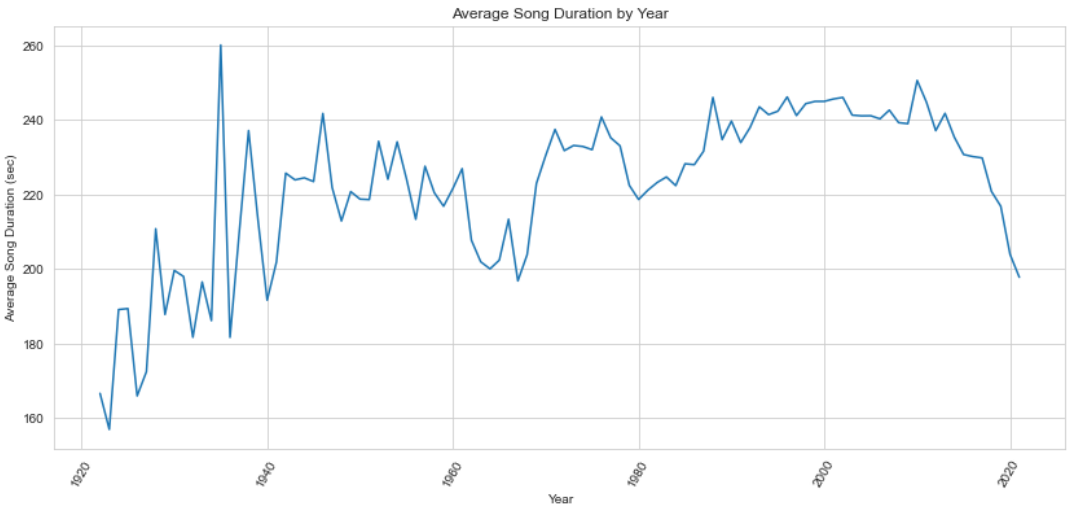


According to the line chart, there is a clear trend that the average loudness of songs has increased dramatically since the 1980's.

How long do songs last on average per year?

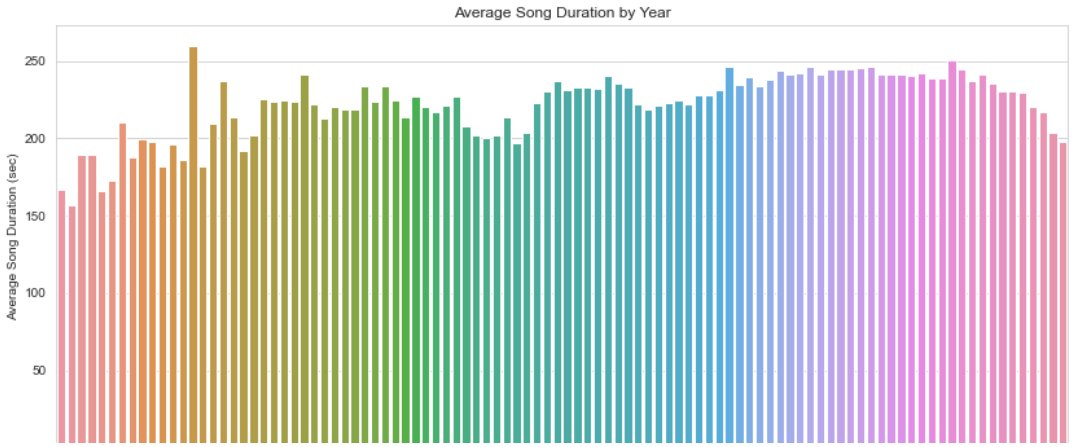
Line Graph - Visualize the Change in Average Duration of Songs per Year

```
avg_dur = mean_df.duration
sns.set_style(style = 'whitegrid')
plot_dims = (14,6)
plot, ax = plt.subplots(figsize = plot_dims)
plot = sns.lineplot(x = mean_df.index, y = avg_dur, ax = ax).set(title = 'Average Song Duration by Year',
                    xlabel = 'Year', ylabel = 'Average Song Duration (sec)')
plt.xticks(rotation = 60)
ticks = plt.xticks(rotation = 60)
```



Bar Graph - Visualize the Change in Average Duration of Songs per Year

```
sns.set_style(style = 'whitegrid')
bar_dims = (14,6)
bar, ax = plt.subplots(figsize = bar_dims)
bar = sns.barplot(x = mean_df.index, y = avg_dur, ax = ax).set(title = 'Average Song Duration by Year',
                  xlabel = 'Year', ylabel = 'Average Song Duration (sec)')
plt.xticks(rotation = 90)
ticks = plt.xticks(rotation = 90)
```



Both of these visualizations show that on average, songs from the 1920's to the 1960's were shorter. However, their average duration was increasing. From the 1970's to the 2000's, there was a gradual increase in average song duration. Finally, from 2010 and on, the average song duration began to see a decline.

Spotify Genres Dataset Analysis

Preview the Dataset

```
df_genres.head()
```

	genre	artist_name	track_name	track_id	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	key	liveness
0	Movie	Henri Salvador	C'est beau de faire un Show	0BRjO6ga9RKCKjfDqeFgWV	0	0.611	0.389	99373	0.910	0.000	C#	0.346
1	Movie	Martin & les fées	Perdu d'avance (par Gad Elmaleh)	0BjC1NfoEOOusryehmNudP	1	0.246	0.590	137373	0.737	0.000	F#	0.151

Check for any null values present in the dataset.

```
pd.isnull(df_genres).sum()
```

```
genre               0
artist_name         0
track_name          0
track_id            0
popularity          0
acousticness        0
danceability        0
duration_ms         0
energy              0
instrumentalness    0
key                 0
liveness            0
loudness            0
mode                0
speechiness         0
tempo               0
time_signature      0
valence             0
dtype: int64
```

```
pd.isnull(df_genres).sum().sum()
```

```
0
```

```
len(df_genres)
```

```
232725
```

There are no missing values in the entire dataset consisting of ~232k rows. Therefore the dataset is good, and our conclusions will be valid.

Inspect and identify the number of rows and columns in the dataset, and check the data type for each variable.

```
df_genres.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 232725 entries, 0 to 232724
Data columns (total 18 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
```

```
 0   genre             232725 non-null  object
 1   artist_name       232725 non-null  object
 2   track_name        232725 non-null  object
 3   track_id          232725 non-null  object
 4   popularity        232725 non-null  int64
 5   acousticness      232725 non-null  float64
 6   danceability      232725 non-null  float64
 7   duration_ms       232725 non-null  int64
 8   energy            232725 non-null  float64
 9   instrumentalness  232725 non-null  float64
 10  key               232725 non-null  object
 11  liveness          232725 non-null  float64
 12  loudness          232725 non-null  float64
 13  mode              232725 non-null  object
 14  speechiness       232725 non-null  float64
 15  tempo             232725 non-null  float64
 16  time_signature    232725 non-null  object
 17  valence           232725 non-null  float64
dtypes: float64(9), int64(2), object(7)
memory usage: 32.0+ MB
```

Descriptive statistics for numerical values in the dataset.

```
df_genres.describe().transpose()
```

	count	mean	std	min	25%	50%	75%	max
popularity	232725.0	41.127502	18.189948	0.00000	29.0000	43.000000	55.0000	100.000
acousticness	232725.0	0.368560	0.354768	0.00000	0.0376	0.232000	0.7220	0.996
danceability	232725.0	0.554364	0.185608	0.05690	0.4350	0.571000	0.6920	0.989
duration_ms	232725.0	235122.339306	118935.909299	15387.00000	182857.0000	220427.000000	265768.0000	5552917.000
energy	232725.0	0.570958	0.263456	0.00002	0.3850	0.605000	0.7870	0.999
instrumentalness	232725.0	0.148301	0.302768	0.00000	0.0000	0.000044	0.0358	0.999
liveness	232725.0	0.215009	0.198273	0.00967	0.0974	0.128000	0.2640	1.000
loudness	232725.0	-9.569885	5.998204	-52.45700	-11.7710	-7.762000	-5.5010	3.744
speechiness	232725.0	0.120765	0.185518	0.02220	0.0367	0.050100	0.1050	0.967
tempo	232725.0	117.666585	30.898907	30.37900	92.9590	115.778000	139.0540	242.903
valence	232725.0	0.454917	0.260065	0.00000	0.2370	0.444000	0.6600	1.000

Inspect and explore the dataset.

Let's look at all of the unique genres in the dataset.

```
dif_genres = df_genres.genre.unique()
len(dif_genres)
```

```
27
```

```
print(dif_genres)
```

```
['Movie' 'R&B' 'A Capella' 'Alternative' 'Country' 'Dance' 'Electronic'
 'Anime' 'Folk' 'Blues' 'Opera' 'Hip-Hop' "Children's Music"
 'Children’s Music' 'Rap' 'Indie' 'Classical' 'Pop' 'Reggae' 'Reggaeton'
 'Jazz' 'Rock' 'Ska' 'Comedy' 'Soul' 'Soundtrack' 'World']
```

Here we see that there are 27 genres in the dataset, however there is an error because "Children's Music" is listed twice as two separate genres when it should be one genre. This was a simple error made when constructing the datset, one of the "Children's Music" genres uses an apostrophe while the other uses a single comma quotation mark.

Clean the dataset for future visualizations.

We can use Unicode to look for these specific characters in the genre names, and adjust them accordingly.

```
print('Single comma quotation mark = ' + '\u2019')
```

```
Single comma quotation mark = ’
```

```
print('Apostrophe = ' + '\u0027')
```

```
Apostrophe = '
```

Rows in dataframe where genre = Children's Music (containing a single comma quotation mark)

```
child_genre_1 = df_genres[(df_genres['genre'] == "Children" + "\u2019" + "s " + "Music")]
child_genre_1.head()
```

	genre	artist_name	track_name	track_id	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	key
77052	Children's Music	Joji	SLOW DANCING IN THE DARK	0rKtyWc8bvkriBthvHKY8d	84	0.5440	0.515	209274	0.479	0.005980	D#
77053	Children's Music	YUNGBLUD	11 Minutes (with Halsey feat. Travis Barker)	4mGdjNMo0RonTlOEb7cYg4	86	0.0116	0.464	239507	0.852	0.000000	B
77054	Children's Music	H.E.R.	Best Part (feat. Daniel Caesar)	4OBZT9EnhYIV17t4pGw7ig	84	0.7950	0.473	209400	0.371	0.000000	E

```
print(len(child_genre_1))
```

```
9353
```

Rows in dataframe where genre = Children's Music (containing an apostrophe)

```
child_genre_2 = df_genres[(df_genres['genre'] == "Children" + "\u0027" + "s " + "Music")]
child_genre_2.head()
```

	genre	artist_name	track_name	track_id	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	key
71649	Children's Music	Toddler Tunes	Itsy Bitsy Spider	749a3IktgztaKcRg3zmQvv	47	0.302	0.901	137454	0.526	0.000000	C#
71650	Children's Music	Spongebob Squarepants	The Best Day Ever	3L3dVggx061Wx5y4d2L0mQ	47	0.202	0.565	182173	0.758	0.000000	B
71651	Children's Music	Blippi	The Excavator Song	3X6fJLY0KvjLQSDHTLwvDn	46	0.528	0.737	175187	0.549	0.000087	A
71652	Children's Music	Pinkfong	Baby Shark Music Box	07enmUOMmpuy7ZVUXHAPIk	44	0.688	0.686	110523	0.121	0.885000	G

Head.

```
print(len(child_genre_2))
```

```
5403
```

Option 1: Incorrect

We could combine the rows associated with child_genre_1 and child_genre_2 to make one "Children's Music" category in the genre column.

```
new_df_genres = pd.read_csv('/Users/mattabruzzeseott/Documents/Portfolio Projects/Jupyter/SpotifyFeatures.csv')
new_df_genres['genre'] = np.where(new_df_genres['genre'] == "Children" + "\u2019" + "s " + "Music", "Children" +
                                  "\u0027" + "s " + "Music", df_genres['genre'])
print(new_df_genres.genre.unique())
new_unique_genre = new_df_genres.groupby(['genre'])
len(new_unique_genre)
```

```
['Movie' 'R&B' 'A Capella' 'Alternative' 'Country' 'Dance' 'Electronic'
 'Anime' 'Folk' 'Blues' 'Opera' 'Hip-Hop' "Children's Music" 'Rap' 'Indie'
 'Classical' 'Pop' 'Reggae' 'Reggaeton' 'Jazz' 'Rock' 'Ska' 'Comedy'
 'Soul' 'Soundtrack' 'World']
26
```

Number of rows in "new_df_genres" dataframe.

```
len(new_df_genres)
```

```
232725
```

```
new_df_genres.groupby(['genre']).count()
```

genre	artist_name	track_name	track_id	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	key	liveness	loudness
A Capella	119	119	119	119	119	119	119	119	119	119	119	119
Alternative	9263	9263	9263	9263	9263	9263	9263	9263	9263	9263	9263	9263
Anime	8936	8936	8936	8936	8936	8936	8936	8936	8936	8936	8936	8936
Blues	9023	9023	9023	9023	9023	9023	9023	9023	9023	9023	9023	9023
Children's Music	14756	14756	14756	14756	14756	14756	14756	14756	14756	14756	14756	14756
Classical	9256	9256	9256	9256	9256	9256	9256	9256	9256	9256	9256	9256
Comedy	9681	9681	9681	9681	9681	9681	9681	9681	9681	9681	9681	9681
Country	8664	8664	8664	8664	8664	8664	8664	8664	8664	8664	8664	8664
Dance	8701	8701	8701	8701	8701	8701	8701	8701	8701	8701	8701	8701
Electronic	9377	9377	9377	9377	9377	9377	9377	9377	9377	9377	9377	9377
Folk	9299	9299	9299	9299	9299	9299	9299	9299	9299	9299	9299	9299
Hip-Hop	9295	9295	9295	9295	9295	9295	9295	9295	9295	9295	9295	9295
Indie	9543	9543	9543	9543	9543	9543	9543	9543	9543	9543	9543	9543
Jazz	9441	9441	9441	9441	9441	9441	9441	9441	9441	9441	9441	9441
Movie	7806	7806	7806	7806	7806	7806	7806	7806	7806	7806	7806	7806
Opera	8280	8280	8280	8280	8280	8280	8280	8280	8280	8280	8280	8280
Pop	9386	9386	9386	9386	9386	9386	9386	9386	9386	9386	9386	9386
Rap	9232	9232	9232	9232	9232	9232	9232	9232	9232	9232	9232	9232

In theory this is a viable solution.

Option 2: Correct

However, upon further inspection of the dataset, it can be observed that the rows contained in the "child_genre_1" dataframe are miscategorized. Therefore, the data is incorrect and should be removed from the original Genres dataset for proper analysis moving forward.

```
df_genres = df_genres.drop(df_genres[df_genres.genre == "Children" + "\u2019" + "s " + "Music"].index)
df_genres
```

	genre	artist_name	track_name	track_id	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	key	li
0	Movie	Henri Salvador	C'est beau de faire un Show	0BRjO6ga9RKCKjfDqeFgWV	0	0.61100	0.389	99373	0.910	0.000000	C#	
1	Movie	Martin & les fées	Perdu d'avance (par Gad Elmaleh)	0BjC1NfoEOOusryehmNudP	1	0.24600	0.590	137373	0.737	0.000000	F#	
2	Movie	Joseph Williams	Don't Let Me Be Lonely Tonight	0CoSDzoNIKCRs124s9uTVy	3	0.95200	0.663	170267	0.131	0.000000	C	
3	Movie	Henri Salvador	Dis-moi Monsieur Gordon Cooper	0Gc6TVm52BwZD07Ki6tIvf	0	0.70300	0.240	152427	0.326	0.000000	C#	
4	Movie	Fabien Nataf	Ouverture	0IusIXpMROHdEPvSl1fTQK	4	0.95000	0.331	82625	0.225	0.123000	F	

```
print(df_genres.genre.unique())
len(df_genres.groupby(['genre']))
```

```
['Movie' 'R&B' 'A Capella' 'Alternative' 'Country' 'Dance' 'Electronic'
 'Anime' 'Folk' 'Blues' 'Opera' 'Hip-Hop' "Children's Music" 'Rap' 'Indie'
 'Classical' 'Pop' 'Reggae' 'Reggaeton' 'Jazz' 'Rock' 'Ska' 'Comedy'
 'Soul' 'Soundtrack' 'World']
26
```

Number of rows in updated "df_genres" dataframe.

```
len(df_genres)
```

```
223372
```

```
df_genres.groupby(['genre']).count()
```

genre	artist_name	track_name	track_id	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	key	liveness	loudness
A Capella	119	119	119	119	119	119	119	119	119	119	119	119
Alternative	9263	9263	9263	9263	9263	9263	9263	9263	9263	9263	9263	9263
Anime	8936	8936	8936	8936	8936	8936	8936	8936	8936	8936	8936	8936
Blues	9023	9023	9023	9023	9023	9023	9023	9023	9023	9023	9023	9023
Children's Music	5403	5403	5403	5403	5403	5403	5403	5403	5403	5403	5403	5403
Classical	9256	9256	9256	9256	9256	9256	9256	9256	9256	9256	9256	9256
Comedy	9681	9681	9681	9681	9681	9681	9681	9681	9681	9681	9681	9681
Country	8664	8664	8664	8664	8664	8664	8664	8664	8664	8664	8664	8664
Dance	8701	8701	8701	8701	8701	8701	8701	8701	8701	8701	8701	8701
Electronic	9377	9377	9377	9377	9377	9377	9377	9377	9377	9377	9377	9377
Folk	9299	9299	9299	9299	9299	9299	9299	9299	9299	9299	9299	9299
Hip-Hop	9295	9295	9295	9295	9295	9295	9295	9295	9295	9295	9295	9295
Indie	9543	9543	9543	9543	9543	9543	9543	9543	9543	9543	9543	9543
Jazz	9441	9441	9441	9441	9441	9441	9441	9441	9441	9441	9441	9441
Movie	7806	7806	7806	7806	7806	7806	7806	7806	7806	7806	7806	7806
Opera	8280	8280	8280	8280	8280	8280	8280	8280	8280	8280	8280	8280
Pop	9386	9386	9386	9386	9386	9386	9386	9386	9386	9386	9386	9386
R&B	8992	8992	8992	8992	8992	8992	8992	8992	8992	8992	8992	8992
Rap	9232	9232	9232	9232	9232	9232	9232	9232	9232	9232	9232	9232
Reggae	8771	8771	8771	8771	8771	8771	8771	8771	8771	8771	8771	8771
Reggaeton	8927	8927	8927	8927	8927	8927	8927	8927	8927	8927	8927	8927
Rock	9272	9272	9272	9272	9272	9272	9272	9272	9272	9272	9272	9272
Ska	8874	8874	8874	8874	8874	8874	8874	8874	8874	8874	8874	8874
Soul	9089	9089	9089	9089	9089	9089	9089	9089	9089	9089	9089	9089

Now that our dataset has been cleaned and the rows with miscategorized data have been removed (~9k), we can move forward with our visualizations.

Data Visualizations

Analyze song duration with respect to genre.

First we need to convert the song duration column in the Genres dataframe from Milliseconds to Seconds.

```
df_genres['duration'] = df_genres['duration_ms'].apply(lambda x: round(x/1000))
df_genres.drop('duration_ms', inplace = True, axis = 1)
df_genres.duration.head()
```

```
0     99
1    137
2    170
3    152
4     83
Name: duration, dtype: int64
```

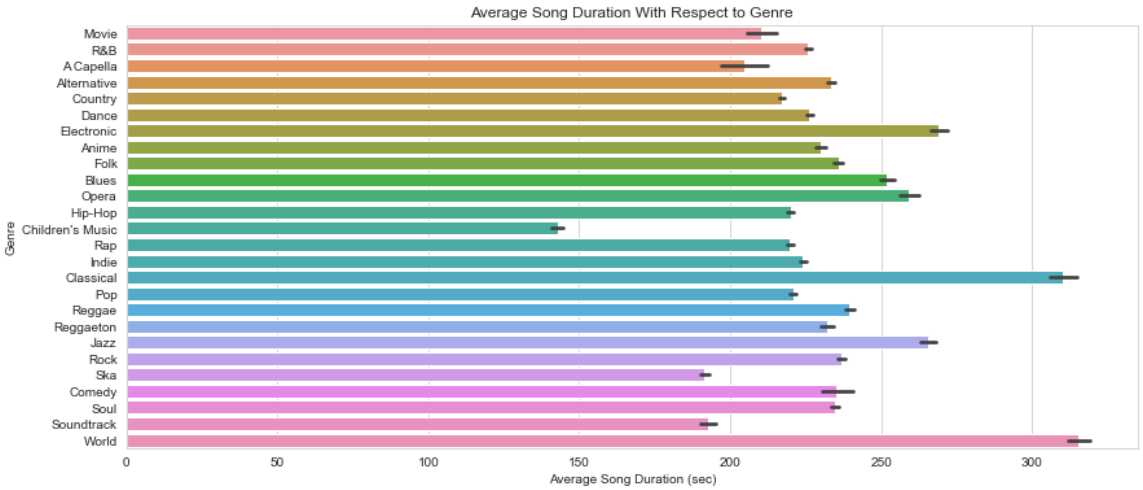
```
df_genres.groupby(['genre']).duration.mean()
```

```
genre
A Capella           204.529412
Alternative         233.241283
Anime               229.931625
Blues               251.934501
Children's Music    142.728114
Classical           310.337619
Comedy              235.307923
Country             217.230379
Dance               226.263878
Electronic          269.198358
Folk                235.803420
Hip-Hop             219.981495
Indie               224.150791
Jazz                265.642729
Movie               210.407891
Opera               259.160386
Pop                 220.859685
R&B                 225.746330
Rap                 219.858752
Reggae              239.498575
```

```
Reggaeton           232.034390
Rock                237.000108
Ska                 191.547780
Soul                234.719441
Soundtrack          192.609475
World               315.322779
Name: duration, dtype: float64
```

Bar Graph - Visualize the Average Song Duration With Respect to Genre

```
sns.set_style(style = 'whitegrid')
song_bar_dims = (14,6)
song_bar, ax = plt.subplots(figsize = song_bar_dims)
sns.color_palette('rocket', as_cmap = True)
song_bar = sns.barplot(data = df_genres, x = 'duration', y = 'genre', ax = ax).set(title =
                       'Average Song Duration With Respect to Genre', xlabel = 'Average Song Duration (sec)',
                       ylabel = 'Genre')
```



The bar graph above visualizes the average song duration per genre. Each bar in the graph displays the mean song duration according to genre as confirmed by the dataframe above. The Seaborn barplot function uses the mean as the default estimator for each bin in the graph.

This visualization shows that the "World" and "Classical" genres have the longest songs on average, while the "Children's Music" genre has the shortest songs on average by a significant amount.

Are shorter songs more popular than longer songs?

First we must sort the songs in the dataset into different bins based on their respective durations; Short (<2 min), Medium (2-4 min), Long (4-6 min), and Very Long (>6 min).

```
pd.options.mode.chained_assignment = None
song_length = df_genres[['popularity', 'duration']]
bin_names = ['Short', 'Medium', 'Long', 'Very Long']
bin_numbers = [0, 119, 239, 359, 999999999]
song_length['length'] = pd.cut(song_length['duration'], bins = bin_numbers, labels = bin_names)
```

Bar Graph - Visualize the Average Song Popularity With Respect to Length

```
length_graph = song_length[['popularity', 'length']].groupby('length').mean().plot.bar(figsize = (14,6))
length_graph.set_xlabel('Length');
length_graph.set_ylabel('Popularity');
length_graph.set_title('Song Popularity by Length');
plt.xticks(rotation = 0);
```

This analysis shows that songs between two and six minutes long are noticebly more popular than songs that are shorter than two minutes and longer than six minutes.

Determine the top 5 genres by popularity.

First we need to determine the average popularity per genre in the dataset through grouping and sorting.

```
pop = df_genres[['genre', 'popularity']].copy()
most_pop = pop.groupby(['genre']).popularity.mean()
most_pop_sort = most_pop.sort_values(ascending = False).head()
most_pop_sort
```

```
genre
Pop        66.590667
Rap        60.533795
Rock       59.619392
Hip-Hop    58.423131
Dance      57.275256
Name: popularity, dtype: float64
```

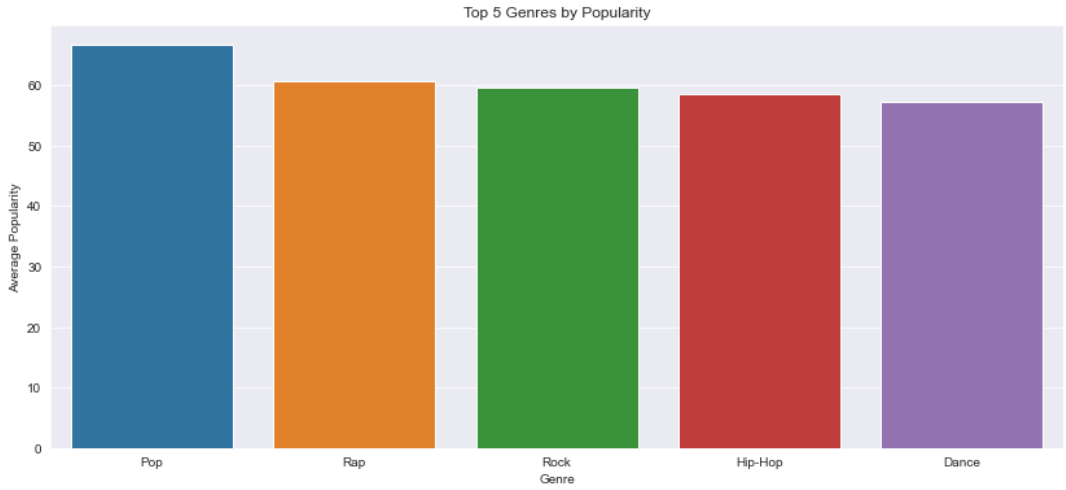
Create a new dataframe to use for our bar graph from the resulting series above.

```
index_names = ['first', 'second', 'third', 'fourth', 'fifth']
column_1 = ['Pop', 'Rap', 'Rock', 'Hip-Hop', 'Dance']
column_2 = [66.590667, 60.533795, 59.619392, 58.423131, 57.275256]
column_names = ['genre', 'mean_popularity']
most_pop_df = pd.DataFrame(list(zip(column_1, column_2)), columns = column_names, index = index_names)
most_pop_df
```

	genre	mean_popularity
first	Pop	66.590667
second	Rap	60.533795
third	Rock	59.619392
fourth	Hip-Hop	58.423131
fifth	Dance	57.275256

Bar Graph - Visualize the Top 5 Most Popular Genres

```
sns.set_style(style = 'darkgrid')
plt.figure(figsize = (14,6))
sns.barplot(data = most_pop_df, x = 'genre', y = 'mean_popularity').set(title = 'Top 5 Genres by Popularity',
            xlabel = 'Genre', ylabel = 'Average Popularity');
```



This bar graph clearly visualizes the top 5 genres and their respective popularity ratings.

Determine the percentage of each genre in the dataset.

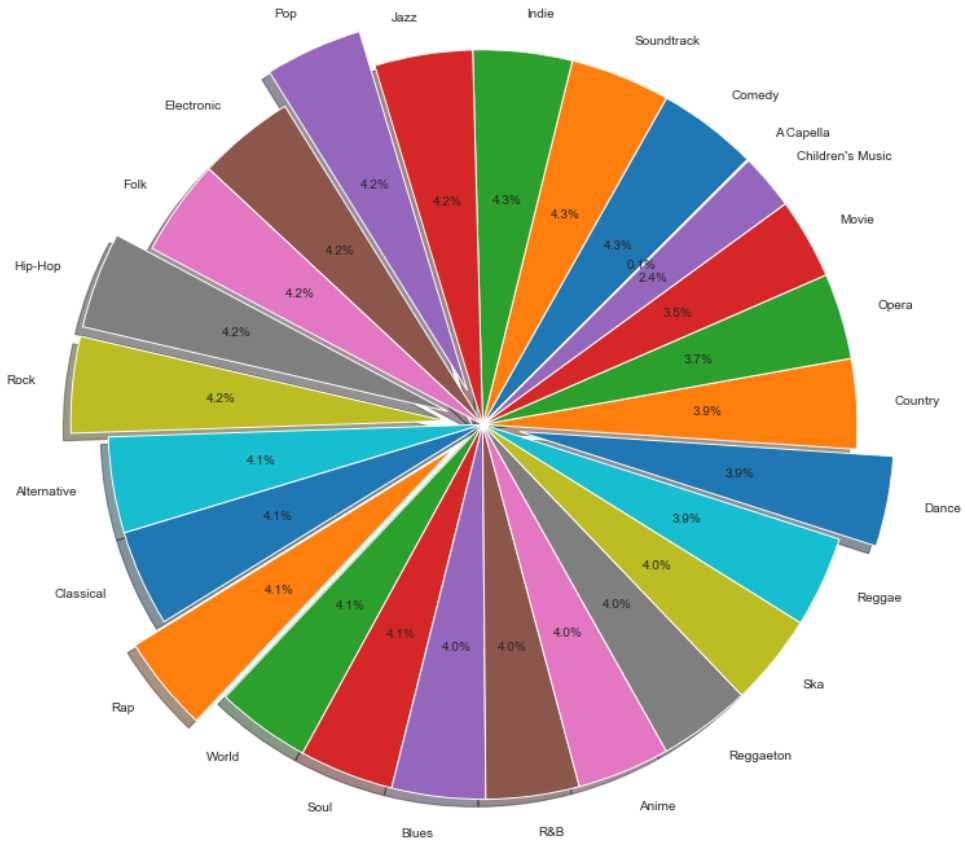
Create a new dataframe for the different genres and their respective song counts in the dataset.

```
pie_genres = pd.DataFrame(df_genres['genre'].value_counts()).reset_index()
pie_genres.rename(columns = {'index':'genres', 'genre':'total_count'}, inplace = True)
pie_genres
```

	genres	total_count
0	Comedy	9681
1	Soundtrack	9646
2	Indie	9543
3	Jazz	9441
4	Pop	9386
5	Electronic	9377
6	Folk	9299
7	Hip-Hop	9295
8	Rock	9272
9	Alternative	9263
10	Classical	9256
11	Rap	9232
12	World	9096
13	Soul	9089
14	Blues	9023
15	R&B	8992
16	Anime	8936
17	Reggaeton	8927
18	Ska	8874
19	Reggae	8771
20	Dance	8701
21	Country	8664
22	Opera	8280
23	Movie	7806
24	Children's Music	5403
25	A Capella	110

Pie Chart - Visual Representation of Each Genre by Percentage

```
labels = pie_genres.genres
sizes = pie_genres.total_count
explode = [0,0,0,0,0.1,0,0,0.1,0.1,0,0,0.1,0,0,0,0,0,0,0,0,0.1,0,0,0,0]
fig = plt.figure(figsize = (14,12))
ax = fig.subplots()
ax.pie(sizes, explode = explode, labels = labels, autopct='%1.1f%%', shadow = True, startangle = 45)
ax.axis('equal');
```



This pie chart visualizes the percentages of each genre in the dataset. From the pie chart it can be easily observed that each of the genres except "A Capella" and "Children's Music" make up ~4% of the dataset. Therefore, there is a fairly even representation of all of the genres within the entire dataset. The three genres with the most songs are "Indie," "Soundtrack," and "Comedy" with 4.3% of the dataset. The genre with the least songs is "A Capella" with 0.1% of the dataset. The pie chart also has the slices of the top 5 most popular genres exploded for easy visualization.

Determine the top 15 most popular artists.

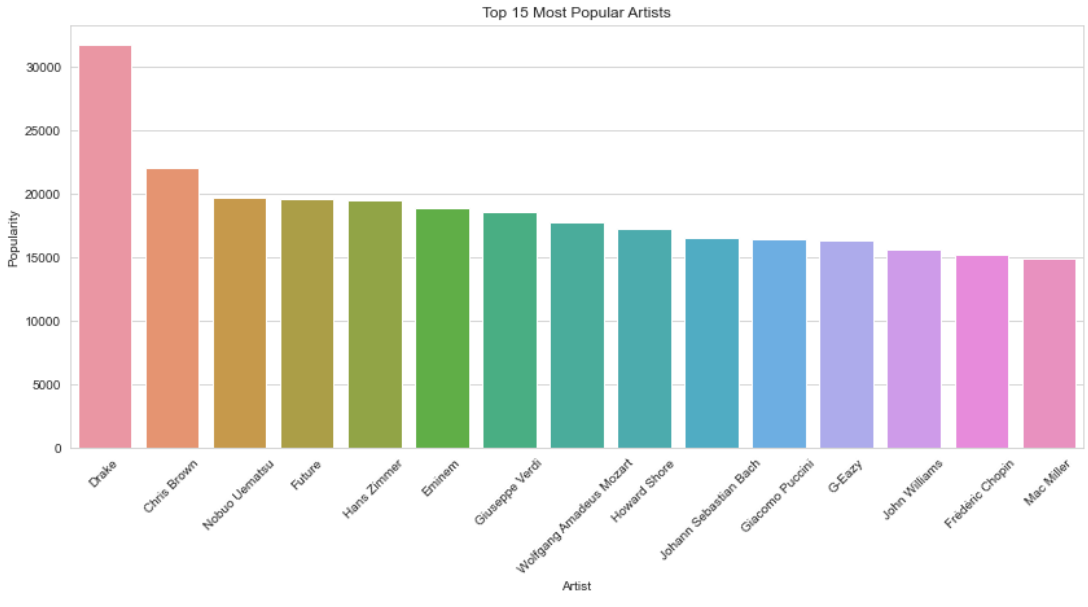
Create a new dataframe that ranks the top 15 artists in the dataset by their respective popularities.

```
artist_pop = df_genres.groupby('artist_name').sum().sort_values('popularity', ascending = False)[:15]
artist_pop = artist_pop.reset_index()
artist_pop
```

	artist_name	popularity	acousticness	danceability	energy	instrumentalness	liveness	loudness	speechiness	tempo	valence	duration
0	Drake	31703	93.260066	316.0960	266.82000	3.099511	99.6669	-3856.222	113.3354	57529.231	171.6347	116148
1	Chris Brown	22047	41.894164	251.4070	238.32800	0.021477	58.4486	-2481.568	44.4065	45572.893	188.2914	89375
2	Nobuo Uematsu	19710	525.223346	338.5755	279.79289	601.255151	139.4140	-12039.046	37.1339	98771.076	268.9002	166362
3	Future	19590	39.376227	240.8530	179.97300	3.487802	61.3051	-2254.880	69.0511	42430.948	114.0344	66701
4	Hans Zimmer	19439	329.527433	147.1366	141.18172	437.113536	90.8699	-10669.538	25.2366	58485.140	59.4124	133739
5	Eminem	18876	38.632000	217.5780	228.33300	0.269124	78.2914	-1359.224	80.0316	34010.032	157.2370	81120
6	Giuseppe Verdi	18580	1330.979000	425.7761	221.33948	169.783522	362.7316	-28175.900	96.1431	140096.253	256.1542	344129
7	Wolfgang Amadeus Mozart	17785	769.482000	257.6289	96.05924	439.144784	161.7496	-17812.061	46.1827	86223.997	214.4025	271882
8	Howard Shore	17283	362.502090	70.1809	79.16198	404.511136	66.9259	-9910.852	19.9603	45231.992	29.5684	122290
9	Johann Sebastian Bach	16508	599.294000	206.5148	73.26695	378.931669	96.1632	-14934.723	31.6374	65928.364	250.8826	140056
10	Giacomo Puccini	16376	1092.427000	293.7235	209.16981	164.622330	281.0961	-20412.311	61.9506	112237.008	155.7211	258659
11	G-Eazy	16300	28.928076	174.1500	171.42400	0.172957	48.2954	-1977.804	64.6429	31863.944	98.1171	59689
12	John Williams	15585	374.532300	104.1749	80.97053	352.071471	68.5215	-9178.677	18.7514	45442.770	50.1524	114278

Bar Graph - Visualize the Top 15 Artists by Popularity

```
sns.set_style(style = 'whitegrid')
plt.figure(figsize = (14,6))
sns.barplot(data = artist_pop, x = 'artist_name', y = 'popularity').set(title = 'Top 15 Most Popular Artists',
            xlabel = 'Artist', ylabel = 'Popularity');
plt.xticks(rotation = 45);
```



The bar graph clearly visualizes the data from the "artist_pop" dataframe above. According to the visualization, Drake is clearly the most popular artist in the entire dataset, followed by Chris Brown, Nobuo Uematsu, Future, etc.

Conclusion

Through the use of Python and its various libraries and functions, one can analyze extensive data and draw useful insights from it. This Spotify EDA project allowed us to analyze large music datasets, create interesting visualizations, find correlations within the data, and extract useful insights.