In [1]:
```
#Import Packages
import os
import glob
import time

import pandas as pd
import numpy as np
from tqdm.notebook import tqdm

import matplotlib.pyplot as plt
from IPython.display import display, HTML

from langdetect.lang_detect_exception import LangDetectException
from langdetect import detect, DetectorFactory
import spacy
```

A. Combine Playlist CSVs into combined_df

The playlist csv files were gathered using a website that takes a Spotify playlist link and returns a file with all of the songs information / metadata. 17 different public playlists from a myriad of genres and time periods were selected as the basis for this dataset. Upon running these playlists through the website and concatenating the resulting csv files, we were left with a dataset of nearly 9,000 songs that is the basis for the beginning of this notebook.

In [2]:
```
# Set Working Directory
working_dir = r"C:\Users\Ryan\OneDrive\Documents\Final Project\Final_Script\Data"
os.chdir(working_dir)
# Our first dataset consists of a number of popular Spotify Playlists downloaded on
# Set the folder path where your CSVs are
folder_path = "Playlist"

# Use glob to get all CSV files in the folder
csv_files = glob.glob(os.path.join(folder_path, "*.csv"))

# Read and combine them into one DataFrame
df_list = [pd.read_csv(file) for file in csv_files]
combined_df = pd.concat(df_list, ignore_index=True)

combined_df
```

Out[2]:

	#	Song	Artist	Popularity	BPM	Genres	Parent Genres	Album	All
0	1	Murder On The Dancefloor	Sophie Ellis-Bextor	2	117	NaN	NaN	Read My Lips	2(0
1	2	Dog Days Are Over	Florence + The Machine	83	150	baroque pop	Classical, Pop	Lungs (Deluxe Version)	2(0(
2	3	Say It Right	Nelly Furtado	84	117	NaN	NaN	Loose	2(0
3	4	Where Is The Love?	Black Eyed Peas	3	94	NaN	NaN	Elephunk	2(0(
4	5	Walking On A Dream	Empire Of The Sun	87	127	NaN	NaN	Walking On A Dream (10th Anniversary Edition)	2(1(
...	...	...	...	...	...	...	...	...	
8681	334	San Tropez	Pink Floyd	59	122	progressive rock, psychedelic rock, classic ro...	Rock	Meddle	1' 1
8682	335	Wish You Were Here	Pink Floyd	83	123	progressive rock, psychedelic rock, classic ro...	Rock	Wish You Were Here	1' 0'
8683	336	Mrs. Robinson - From The Graduate Soundtrack	Simon & Garfunkel	77	92	folk rock, folk	Folk, Rock	Bookends	1' 0.
8684	337	Subterranean Homesick Blues	Bob Dylan	64	174	folk rock, folk, singer-songwriter, roots rock...	Country, Rock, Folk	Bringing It All Back Home	1' 0.
8685	338	Welcome to Paradise	Green Day	71	89	punk, pop punk	Pop, Rock	Dookie	1' 0.

8686 rows × 26 columns

In [3]:
```
#List of the columns i combined_df
combined_df.columns.tolist()
```

Out[3]:
```
['#',
 'Song',
 'Artist',
 'Popularity',
 'BPM',
 'Genres',
 'Parent Genres',
 'Album',
 'Album Date',
 'Time',
 'Dance',
 'Energy',
 'Acoustic',
 'Instrumental',
 'Happy',
 'Speech',
 'Live',
 'Loud (Db)',
 'Key',
 'Time Signature',
 'Added At',
 'Spotify Track Id',
 'Album Label',
 'Camelot',
 'ISRC',
 'Time ']
```

B. Initial Clean the Dataset (remove duplicates, remixes, unnecessary columns, keep only the first listed artist, and split each song title at '-'¶

The clean_artist function standardizes artist names by splitting them at commas and keeping only the first listed artist. This helps isolate the primary artist when multiple collaborators are listed. However, some artists—like "Tyler, The Creator" and "Earth, Wind & Fire"—have commas in their actual names. To avoid incorrectly splitting these, the function checks for exceptions and preserves those names in full. Example: "Post Malone, Swae Lee" --> "Post Malone" (split at comma)

Songs are also split at "-" in their titles. This helps filter out phrases such as features.

In [4]:
```
# Define columns to drop
columns_to_drop = [
    "Added At", "Spotify Track Id", "Album Label", "#", "ISRC",
    "Album Date", "Time ", "Album"
]
```

```
# Define artist exceptions - these artists have commas in their names, but we don't
exceptions = ["Tyler, The Creator", "Earth, Wind & Fire"]

# Artist cleaning function - standardizes artist names (keeps exceptions intact)
# For non-exceptions: grabs the first artist listed before a comma (the main artist
def clean_artist(artist_name):
    if artist_name in exceptions:
        return artist_name.strip()
    return artist_name.split(",")[0].strip()

# Build final cleaned DataFrame step-by-step
clean_df = (
    combined_df
    .drop_duplicates(subset=["Song", "Artist"], keep="first")
    .query("~Song.str.contains('remix', case=False, na=False)", engine="python")
    .drop(columns=[col for col in columns_to_drop if col in combined_df.columns])
    .assign(
        Artist_Clean=lambda df: df["Artist"].apply(clean_artist),
        Song_Clean=lambda df: df["Song"].str.split(" - ").str[0]
    )
    .reset_index(drop=True)
)
```

In [5]:
```
# Check to make sure the Artists in the exception list were preserved before saving
tyler_songs = clean_df[clean_df["Artist_Clean"] == "Tyler, The Creator"]
print(tyler_songs)
```

	Song	Artist	Popularity	BPM	Genres	\
4594	GONE, GONE / THANK YOU	Tyler, The Creator	79	79	NaN	
4624	Glitter	Tyler, The Creator	70	79	NaN	
4840	ARE WE STILL FRIENDS?	Tyler, The Creator	86	187	NaN	
4946	DOGTOOTH	Tyler, The Creator	12	78	NaN	
5311	LUMBERJACK	Tyler, The Creator	66	171	NaN	
5412	I THINK	Tyler, The Creator	77	121	NaN	
5416	EARFQUAKE	Tyler, The Creator	84	80	NaN	
5485	RUNNING OUT OF TIME	Tyler, The Creator	79	84	NaN	
5583	NEW MAGIC WAND	Tyler, The Creator	82	140	NaN	
5586	BEST INTEREST	Tyler, The Creator	83	98	NaN	
5838	Noid	Tyler, The Creator	75	82	NaN	

	Parent Genres	Time	Dance	Energy	Acoustic	Instrumental	Happy	\
4594	NaN	06:15	52	49	22	0	47	
4624	NaN	03:44	45	43	31	0	37	
4840	NaN	04:25	22	50	13	0	31	
4946	NaN	02:41	71	65	51	0	80	
5311	NaN	02:18	39	76	12	0	43	
5412	NaN	03:32	83	58	1	0	43	
5416	NaN	03:10	55	50	23	0	41	
5485	NaN	02:57	36	39	30	0	16	
5583	NaN	03:15	62	73	10	0	46	
5586	NaN	02:07	60	57	9	0	34	
5838	NaN	04:44	38	78	52	0	42	

	Speech	Live	Loud (Db)	Key	Time Signature	Camelot	\
4594	10	10	-8	A#/B♭ minor	4	3A	
4624	20	60	-12	C♯/D♭	5	3B	
4840	0	10	-8	A#/B♭	3	6B	
4946	30	20	-6	G#/A♭	4	4B	
5311	50	50	-6	A	4	11B	
5412	0	10	-8	G#/A♭ minor	4	1A	
5416	0	80	-9	A	4	11B	
5485	0	10	-11	F#/G♭	4	2B	
5583	10	60	-5	F minor	4	4A	
5586	0	30	-8	B minor	3	10A	
5838	30	40	-6	C♯/D♭	4	3B	

	Artist_Clean	Song_Clean
4594	Tyler, The Creator	GONE, GONE / THANK YOU
4624	Tyler, The Creator	Glitter
4840	Tyler, The Creator	ARE WE STILL FRIENDS?
4946	Tyler, The Creator	DOGTOOTH
5311	Tyler, The Creator	LUMBERJACK
5412	Tyler, The Creator	I THINK
5416	Tyler, The Creator	EARFQUAKE
5485	Tyler, The Creator	RUNNING OUT OF TIME
5583	Tyler, The Creator	NEW MAGIC WAND
5586	Tyler, The Creator	BEST INTEREST
5838	Tyler, The Creator	Noid

In [6]:
```
clean_df.to_csv("initial_songs_master.csv", index=False)
```

C. Genre Scrape and Cleaning

Due to the long runtime of the genre scraping process, we've included a pre-scraped CSV (songs_with_genre_tags_full.csv) for direct import. This ensures faster execution and reproducibility.

The source code used for scraping genres is provided below, but commented out for convenience.

MusicBrainz setup # musicbrainzngs.set_useragent("GenreTagFinder", "1.0", "your_email@example.com") musicbrainzngs.set_useragent("GenreTagFinder", "1.0", "jackmetzger13@gmail.com") # Last.fm setup # LASTFM_API_KEY = "your_lastfm_key" # LASTFM_API_SECRET = "your_lastfm_secret" LASTFM_API_KEY = "c664435134e5705a886a39d77eb9d216" LASTFM_API_SECRET = "0eab85625c09e1f392fd2e1a6cc99452" lastfm_network = pylast.LastFMNetwork(api_key=LASTFM_API_KEY, api_secret=LASTFM_API_SECRET) # Get the MusicBrainz ID (MBID) for an artist based on a song title and artist name def get_artist_mbid(song_title, artist_name): try: # Search MusicBrainz for a recording that matches the song and artist result = musicbrainzngs.search_recordings(recording=song_title, artist=artist_name, limit=1) recording = result['recording-list'][0] # Extract the artist's MBID from the search result artist = recording['artist-credit'][0] ['artist'] return artist['id'] except Exception as e: # Print error if MBID lookup fails print(f"[MBID] Error finding '{song_title}' by '{artist_name}': {e}") return None # Get top genre tags for a specific song using Last.fm def get_song_level_tags(song_title, artist_name): try: # Fetch track object from Last.fm track = lastfm_network.get_track(artist_name, song_title) # Get top 5 tags associated with the track tags = track.get_top_tags(limit=5) if tags: # Return tag names as a list return [tag.item.name for tag in tags] except Exception as e: # Print error if track-level tags can't be retrieved print(f"[Track] No track tags for '{song_title}' by '{artist_name}': {e}") return [] # Get top genre tags for an artist using their MBID def get_artist_level_tags(artist_mbid): try: # Fetch artist object from Last.fm using MBID artist = lastfm_network.get_artist_by_mbid(artist_mbid) # Get top 5 tags associated with the artist tags = artist.get_top_tags(limit=5) # Return tag names as a list return [tag.item.name for tag in tags] except Exception as e: # Print error if artist-level tags can't be retrieved print(f"[Artist] Error getting artist tags: {e}") return [] # Determine genre tags for a song, using track-level first, then artist-level as fallback def get_genre_tags(song_title, artist_name): # Try to get track-level tags track_tags = get_song_level_tags(song_title, artist_name) if track_tags: return track_tags, "track" # If no track tags, try to get artist MBID mbid = get_artist_mbid(song_title, artist_name) if mbid: # Use MBID to get artist-level tags artist_tags = get_artist_level_tags(mbid) if artist_tags: return artist_tags, "artist" # If no tags found, return empty list and source as "none" return [], "none" df = pd.read_csv("initial_songs_master.csv") results = [] for idx, row in tqdm(df.iterrows(), total=len(df), desc="Processing songs"): original_song = row['Song'] original_artist = row['Artist'] song_clean = row['Song_Clean'] artist_clean = row['Artist_Clean'] tags, source = get_genre_tags(song_clean, artist_clean) tags_str = ", ".join(tags) if tags else "" results.append({ "Song": original_song, "Artist": original_artist, "Genre Tags": tags_str, "Tag Source": source }) time.sleep(0.25) results_df = pd.DataFrame(results) results_df.to_csv("songs_with_genre_tags_full.csv", index=False) print("\n ✅ Done! Results saved to 'songs_with_genre_tags_full.csv'.")

In [7]:
```
# Load the pre-scraped genre CSV file created from "initial_songs_master.CSV"
genre_df = pd.read_csv("songs_with_genre_tags_full.csv")

# Get value counts for the 'Tag Source' column
# Track - genre scraped directly from the song
# Artist - inferred from the artist
# None - no genre tag was found
tag_source_counts = genre_df["Tag Source"].value_counts()

# Display the result
print(tag_source_counts)
```

```
Tag Source
track     5468
none       687
artist     505
Name: count, dtype: int64
```

In [8]:
```
# Keep only rows where Tag Source is 'track' - more reliable than artist or no tags
df_track_only = genre_df[genre_df["Tag Source"] == "track"]

# Optionally reset the index
df_track_only = df_track_only.reset_index(drop=True)

df_track_only
```

Out[8]:

	Song	Artist	Genre Tags	Tag Source
0	Sweet Dreams (Are Made of This) - 2005 Remaster	Eurythmics,Annie Lennox,Dave Stewart	80s, pop, new wave, female vocalists, synth pop	track
1	Smalltown Boy	Bronski Beat	80s, new wave, synthpop, pop, synth pop	track
2	I'm Still Standing	Elton John	pop, 80s, elton john, rock, classic rock	track
3	Funky Town	Lipps Inc.	Disco, 80s, pop, 70s, dance	track
4	I'm So Excited	The Pointer Sisters	80s, Disco, pop, dance, soul	track
...	...	...	...	...
5463	You're The First, The Last, My Everything - Edit	Barry White	soul, barry white, Disco, 70s, Love	track
5464	Get It On	T. Rex	glam rock, 70s, classic rock, rock, glam	track
5465	We Are Family - 1995 Remaster	Sister Sledge	Disco, 70s, dance, funk, soul	track
5466	Love Really Hurts Without You	Billy Ocean	80s, pop, soul, 70s, Disco	track
5467	I Can See Clearly Now	Johnny Nash	reggae, 70s, pop, oldies, soul	track

5468 rows × 4 columns

In [9]:
```
# Save to a new CSV
df_track_only.to_csv("songs_genre_clean.csv", index=False)
```

Lyric Scrape + Cleaning

Due to the long runtime of the lyric scraping process, we've included a pre-scraped CSV (week_2_songs_genre_lyrics.csv) for direct import. This CSV was built off

"songs_genre_clean.CSV). This ensures faster execution and reproducibility.

The source code used for scraping lyrics is provided below, but commented out for convenience.

SET YOUR GENIUS API TOKEN GENIUS_API_TOKEN = "lWBc3eivTr7Q3g0DcSl7hjpm-eUT7LaZtwL-cBCVk3wcGb1efmqBhq8mLDiJmutl" # Initialize Genius API using the lyricsgenius Python Library genius = lyricsgenius.Genius(GENIUS_API_TOKEN, #API token defined above skip_non_songs=True, #Skip non songs (interviews, podcasts, etc.) remove_section_headers=True, #Remove "[Verse]", "[Chorus]", etc. timeout=15 #API will wait 15 seconds for a response before timing out) genius.verbose = False def load_songs_from_csv(filepath): df = pd.read_csv(filepath) required_cols = {'Song', 'Artist'} if not required_cols.issubset(df.columns): raise ValueError(f"CSV must contain columns: {required_cols}") return df.to_dict(orient='records') csv_input_path = "songs_genre_clean.csv" tracks = load_songs_from_csv(csv_input_path) def enrich_with_lyrics(tracks): for track in tqdm(tracks, desc="Fetching lyrics"): try: song = genius.search_song(track['Song'], track['Artist']) if song and song.lyrics: raw_lyrics = song.lyrics.strip() # Optional cleaning if "Read More" in raw_lyrics: cleaned = raw_lyrics.split("Read More", 1)[1].strip() elif "Lyrics" in raw_lyrics: cleaned = raw_lyrics.split("Lyrics", 1)[1].strip() else: cleaned = raw_lyrics track['lyrics'] = cleaned else: track['lyrics'] = None except Exception as e: print(f" ❌ Failed for {track['Song']} by {track['Artist']}: {e}") track['lyrics'] = None time.sleep(1) # Respect Genius rate limits return tracks enriched_tracks = enrich_with_lyrics(tracks) csv_output_path = "songs_genre_lyrics.csv" pd.DataFrame(enriched_tracks).to_csv(csv_output_path, index=False) print(f" ✅ Done. Saved to {csv_output_path}")

Removing rows with missing or empty lyrics.

In [10]:
```
# Import pre-scraped (lyrics) CSV file
lyrics_df = pd.read_csv('songs_genre_lyrics.csv')

# Check how many rows have empty or missing lyrics
empty_lyrics = lyrics_df[lyrics_df['lyrics'].isna() | (lyrics_df['lyrics'].str.stri
print(f"Number of rows with empty lyrics: {len(empty_lyrics)}")

# Remove rows with empty or missing lyrics
lyrics_df = lyrics_df[~(lyrics_df['lyrics'].isna() | (lyrics_df['lyrics'].str.strip

lyrics_df
```

```
Number of rows with empty lyrics: 671
```

Out[10]:

	Song	Artist	Genre Tags	Tag Source	lyrics
1	Smalltown Boy	Bronski Beat	80s, new wave, synthpop, pop, synth pop	track	To your soul\nTo your soul\nCry\nCry\nCry\n\nY...
2	I'm Still Standing	Elton John	pop, 80s, elton john, rock, classic rock	track	You could never know what it's like\nYour bloo...
3	Funky Town	Lipps Inc.	Disco, 80s, pop, 70s, dance	track	Gotta make a move to a town that's right for m...
4	I'm So Excited	The Pointer Sisters	80s, Disco, pop, dance, soul	track	Tonight's the night we're gonna make it happen...
5	Cheri Cheri Lady	Modern Talking	80s, Disco, pop, Modern Talking, dance	track	Oh, I cannot explain\nEvery time, it's the sam...
...	...	...	...	...	...
5460	Rock with You - Single Version	Michael Jackson	pop, michael jackson, 80s, Disco, dance	track	Girl, close your eyes\nLet that rhythm get int...
5462	You Sexy Thing	Hot Chocolate	Disco, 70s, funk, soul, pop	track	I believe in miracles\nWhere're you from?\nYou...
5464	Get It On	T. Rex	glam rock, 70s, classic rock, rock, glam	track	Well, you're dirty and sweet\nClad in black, d...
5466	Love Really Hurts Without You	Billy Ocean	80s, pop, soul, 70s, Disco	track	You run around town like a fool and you think ...
5467	I Can See Clearly Now	Johnny Nash	reggae, 70s, pop, oldies, soul	track	I can see clearly now the rain is gone\nI can ...

4797 rows × 5 columns

Dropping rows whose lyrics are not English.

In [11]:
```
#setting the langdetect seed for reproducibility
DetectorFactory.seed = 0

# Uses langdetect to identify the language of text
def detect_language(text):
    try:
        return detect(text)
    except LangDetectException:
        return 'error'

# Adds a new column 'language' to the dataframe and applies the detect_language fun
```

```
# Each row has a detected language code; 'en' - English, 'es' - Spanish, 'fr' - Fre
lyrics_df['language'] = lyrics_df['lyrics'].apply(detect_language)

# Filters out non-English songs and removes them from the dataframe.
non_english_df = lyrics_df[lyrics_df['language'] != 'en']
lyrics_df = lyrics_df[~lyrics_df['Song'].isin(non_english_df['Song'])].reset_index(

# Resets the index of the dataframe.
lyrics_df = lyrics_df.reset_index(drop=True)

lyrics_df.to_csv('lyrics_df')
```

Combine the genre and lyric dataframes with the metadata dataframe for a final dataset.

In [12]:
```
master_df = pd.read_csv('initial_songs_master.csv')

# Merge Genre Tags and lyrics from lyrics_df
# We'll assume the columns to join on are 'artist' and 'title'
columns_to_add = ['Artist', 'Song', 'Genre Tags', 'lyrics']  # adjust names if need

# Ensure lyrics_df has only the necessary columns
df_to_merge = lyrics_df[columns_to_add]

# Perform the merge
merged_df = master_df.merge(df_to_merge, on=['Artist', 'Song'], how='left')


# Remove rows where lyrics are NaN or just whitespace
final_df = merged_df[~(merged_df['lyrics'].isna() | (merged_df['lyrics'].str.strip(

# Reset index for cleanliness
final_df = final_df.reset_index(drop=True)


final_df = final_df.drop(columns=['Genres', 'Parent Genres', 'Time Signature'])
```

In [13]:
```
final_df.to_csv('dataset.csv')
```

In [14]:
```
df = pd.read_csv('dataset.csv')
df.head()
df.info()
df.describe(include='object')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4489 entries, 0 to 4488
Data columns (total 20 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Unnamed: 0     4489 non-null   int64
 1   Song           4489 non-null   object
 2   Artist         4489 non-null   object
 3   Popularity     4489 non-null   int64
 4   BPM            4489 non-null   int64
 5   Time           4125 non-null   object
 6   Dance          4489 non-null   int64
 7   Energy         4489 non-null   int64
 8   Acoustic       4489 non-null   int64
 9   Instrumental   4489 non-null   int64
 10  Happy          4489 non-null   int64
 11  Speech         4489 non-null   int64
 12  Live           4489 non-null   int64
 13  Loud (Db)      4489 non-null   int64
 14  Key            4489 non-null   object
 15  Camelot        4489 non-null   object
 16  Artist_Clean   4489 non-null   object
 17  Song_Clean     4489 non-null   object
 18  Genre Tags     4489 non-null   object
 19  lyrics         4489 non-null   object
dtypes: int64(11), object(9)
memory usage: 701.5+ KB
```

Out[14]:

	Song	Artist	Time	Key	Camelot	Artist_Clean	Song_Clean	Genre Tags	lyrics
count	4489	4489	4125	4489	4489	4489	4489	4489	4489
unique	4316	2009	356	24	24	1632	4284	4180	4471
top	Heaven	Taylor Swift	03:29	C	8B	Drake	Closer	Bryson Tiller	On my way, on my way\nOn my way, on my way\n\n...
freq	4	49	45	430	430	66	4	12	2

Note the duplicate songs and lyrics. We will clean that up next. Also note that Key and Camelot have the same amount of unique values.

Cleaning Duplicate Lyrics and Song Titles.

In [15]:
```
# Step 1: Sort by popularity descending
df = df.sort_values(by='Popularity', ascending=False)

# Step 2: Drop duplicate lyrics, keeping the most popular version
```

```
df = df.drop_duplicates(subset='lyrics', keep='first').reset_index(drop=True)

# Step 3: Find songs with duplicate titles (after lyrics deduplication)
duplicates = (
    df[df['Song'].duplicated(keep=False)]
    [['Song', 'Artist']]
    .sort_values(by='Song')
)

# Step 4: Display in scrollable HTML box
html_box = f"""
<div style="height: 300px; overflow-y: scroll; border: 1px solid #ccc; padding: 10p
    <h4 style="margin-top:0;">Songs with Duplicate Titles</h4>
    {duplicates.to_html(index=False)}
</div>
"""

display(HTML(html_box))
```

Songs with Duplicate Titles

Song	Artist
23	Mike WiLL Made-It,Miley Cyrus,Wiz Khalifa,Juicy J
23	Wallice
23	NLE Choppa
Afraid	Nico
Afraid	The Neighbourhood
After Hours	The Weeknd
After Hours	The Velvet Underground
Ain't No Sunshine	Bill Withers

In [16]:
```
df.describe(include='object')
```

Out[16]:

	Song	Artist	Time	Key	Camelot	Artist_Clean	Song_Clean	Genre Tags	lyrics
count	4471	4471	4109	4471	4471	4471	4471	4471	4471
unique	4299	2008	356	24	24	1632	4271	4175	4471
top	Alive	Taylor Swift	03:29	C	8B	Drake	Alive	Bryson Tiller	(But I wanna stay)\n\nI want you to stay\n'Til...
freq	4	49	45	429	429	66	4	12	1

The difference between count lyrics and unique lyrics is now 0.

The difference between count song and unique song is now much less (172) likely due to common song names.

In [17]:

```
# Group by the cleaned song name
grouped = df.groupby('Song_Clean')['Song'].nunique()

# Filter where more than one distinct 'song' maps to a single 'song_clean'
duplicates = grouped[grouped > 1]

# Now view those entries
df[df['Song_Clean'].isin(duplicates.index)][['Song_Clean', 'Song', 'Artist', 'Popul
```

Out[17]:

	Song_Clean	Song	Artist	Popularity
2527	4 AM	4 AM	2 Chainz,Travis Scott	62
3268	4 AM	4 AM - Adam K & Soha Radio Edit	Kaskade	51
274	Another One Bites The Dust	Another One Bites The Dust - Remastered 2011	Queen	84
4073	Another One Bites The Dust	Another One Bites The Dust	Queen	1
2625	Babooshka	Babooshka	Kate Bush	61
2311	Babooshka	Babooshka - 2018 Remaster	Kate Bush	65
2209	Boys	Boys - Summertime Love	Sabrina	66
4138	Boys	Boys	Hippo Campus	0
637	Dancing In The Flames	Dancing In The Flames	The Weeknd	80
3216	Dancing In The Flames	Dancing In The Flames - Acoustic	The Weeknd	52
4014	Every Little Thing She Does Is Magic	Every Little Thing She Does Is Magic - Remaste...	The Police	2
742	Every Little Thing She Does Is Magic	Every Little Thing She Does Is Magic	The Police	79
2249	Hot Stuff	Hot Stuff - 12 Version	Donna Summer	66
1693	Hot Stuff	Hot Stuff	Donna Summer	71
3302	I'm Not Alone	I'm Not Alone - 2009 Remaster	Calvin Harris	50
2541	I'm Not Alone	I'm Not Alone - Radio Edit	Calvin Harris	62
4029	Lola	Lola - 2020 Stereo Remaster	The Kinks	2
3949	Lola	Lola	The Kinks	4
719	Message In A Bottle	Message In A Bottle	The Police	80
4257	Message In A Bottle	Message In A Bottle - Remastered 2003	The Police	0
2368	Mirrors	Mirrors - Radio Edit	Justin Timberlake	64
254	Mirrors	Mirrors	Justin Timberlake	85
334	On The Floor	On The Floor	Jennifer Lopez,Pitbull	84
937	On The Floor	On The Floor - Radio Edit	Jennifer Lopez,Pitbull	78

	Song_Clean	Song	Artist	Popularity
2186	Out of Touch	Out of Touch	Daryl Hall & John Oates	66
3175	Out of Touch	Out of Touch - Single Version	Daryl Hall & John Oates	53
524	Roxanne	Roxanne	The Police	81
4071	Roxanne	Roxanne - Remastered 2003	The Police	1
3543	Sarah	Sarah - Version 3	Thin Lizzy	42
4343	Sarah	Sarah	Alex G	0
3079	Show Me The Way	Show Me The Way - Live	Peter Frampton	55
1979	Show Me The Way	Show Me The Way	Peter Frampton	68
1919	Silhouettes	Silhouettes - Original Radio Edit	Avicii	69
4229	Silhouettes	Silhouettes	Colony House	0
4222	Sleepless	Sleepless	Flume,Jezzabell Doran	0
4220	Sleepless	Sleepless - Radio Edit	CAZZETTE,The High	0
3410	Sunday Bloody Sunday	Sunday Bloody Sunday	U2	47
1287	Sunday Bloody Sunday	Sunday Bloody Sunday - Remastered 2008	U2	74
1465	Surfin' U.S.A.	Surfin' U.S.A. - Remastered 2001	The Beach Boys	73
2532	Surfin' U.S.A.	Surfin' U.S.A.	The Beach Boys	62
3404	The Letter	The Letter - Single Version	Joe Cocker	47
1829	The Letter	The Letter	The Box Tops	70
4258	The Show Must Go On	The Show Must Go On	Queen	0
1612	The Show Must Go On	The Show Must Go On - Remastered 2011	Queen	72
3774	Wake Me Up	Wake Me Up	Avicii	17
3837	Wake Me Up	Wake Me Up - Radio Edit	Avicii	9
2800	Waves	Waves	Luke Bryan	59
3815	Waves	Waves - Robin Schulz Radio Edit	Mr. Probz,Robin Schulz	11
1637	Waves	Waves	Kanye West	71

	Song_Clean	Song	Artist	Popularity
2824	What's Love Got to Do with It	What's Love Got to Do with It	Tina Turner	59
1627	What's Love Got to Do with It	What's Love Got to Do with It - 2015 Remaster	Tina Turner	72
2463	White Wedding	White Wedding	Billy Idol	63
1091	White Wedding	White Wedding - Pt. 1	Billy Idol	76
2636	Wicked Game	Wicked Game - Remastered	Chris Isaak	61
4436	Wicked Game	Wicked Game	Chris Isaak	0
1899	Wuthering Heights	Wuthering Heights	Kate Bush	69
3567	Wuthering Heights	Wuthering Heights - 2018 Remaster	Kate Bush	41

In [18]:
```
# Step 1: Sort by popularity descending
df = df.sort_values(by='Popularity', ascending=False)

# Step 2: Drop duplicates based on Song_Clean and Artist
# This keeps the most popular version for each (Song_Clean, Artist) pair
df_deduped = df.drop_duplicates(subset=['Song_Clean', 'Artist'], keep='first').rese

removed = pd.merge(df, df_deduped, how='outer', indicator=True)
removed = removed[removed['_merge'] == 'left_only']

# View removed entries
removed[['Song_Clean', 'Song', 'Artist', 'Popularity']].sort_values(by='Song_Clean'
```

Out[18]:

	Song_Clean	Song	Artist	Popularity
4408	Another One Bites The Dust	Another One Bites The Dust	Queen	1
1314	Babooshka	Babooshka	Kate Bush	61
4008	Dancing In The Flames	Dancing In The Flames - Acoustic	The Weeknd	52
888	Every Little Thing She Does Is Magic	Every Little Thing She Does Is Magic - Remaste...	The Police	2
1934	Hot Stuff	Hot Stuff - 12 Version	Donna Summer	66
2858	I'm Not Alone	I'm Not Alone - 2009 Remaster	Calvin Harris	50
724	Lola	Lola - 2020 Stereo Remaster	The Kinks	2
641	Message In A Bottle	Message In A Bottle - Remastered 2003	The Police	0
142	Mirrors	Mirrors - Radio Edit	Justin Timberlake	64
247	On The Floor	On The Floor - Radio Edit	Jennifer Lopez,Pitbull	78
1608	Out of Touch	Out of Touch - Single Version	Daryl Hall & John Oates	53
2098	Roxanne	Roxanne - Remastered 2003	The Police	1
755	Show Me The Way	Show Me The Way - Live	Peter Frampton	55
4403	Sunday Bloody Sunday	Sunday Bloody Sunday	U2	47
2045	Surfin' U.S.A.	Surfin' U.S.A.	The Beach Boys	62
4406	The Show Must Go On	The Show Must Go On	Queen	0
2828	Wake Me Up	Wake Me Up - Radio Edit	Avicii	9
1018	What's Love Got to Do with It	What's Love Got to Do with It	Tina Turner	59
883	White Wedding	White Wedding	Billy Idol	63
1028	Wicked Game	Wicked Game	Chris Isaak	0
989	Wuthering Heights	Wuthering Heights - 2018 Remaster	Kate Bush	41

Upon further investigation, Key and Camelot seem to be the exact same thing just in different notation. Will run Cramer's V categorical correlation to confirm this, and if score is one will remove Key from the dataset

In [19]:
```
import numpy as np
from scipy.stats import chi2_contingency

def cramers_v(x, y):
    confusion_matrix = pd.crosstab(x, y)
    chi2 = chi2_contingency(confusion_matrix)[0]
    n = confusion_matrix.sum().sum()
    phi2 = chi2 / n
    r, k = confusion_matrix.shape
    phi2_corr = max(0, phi2 - ((k-1)*(r-1))/(n-1))  # bias correction
    r_corr = r - ((r-1)**2)/(n-1)
    k_corr = k - ((k-1)**2)/(n-1)
    return np.sqrt(phi2_corr / min((k_corr-1), (r_corr-1)))

cramers_v(df['Key'], df['Camelot'])

# Dropping 'Key' column
df = df.drop(columns=['Key'])

#Remove songs without time
df = df[df['Time'].notnull()]
df = df.reset_index(drop=True)
```

In [20]:
```
df.describe()
```

Out[20]:

	Unnamed: 0	Popularity	BPM	Dance	Energy	Acoustic	Instr
count	4109.000000	4109.000000	4109.000000	4109.000000	4109.000000	4109.000000	410
mean	2415.318326	55.970309	121.808226	58.770260	64.348017	23.628863	
std	1209.713314	26.733076	27.474388	14.840298	20.016864	26.470157	1
min	0.000000	0.000000	49.000000	9.000000	3.000000	0.000000	
25%	1393.000000	48.000000	101.000000	49.000000	51.000000	2.000000	
50%	2426.000000	65.000000	121.000000	59.000000	67.000000	12.000000	
75%	3458.000000	75.000000	138.000000	69.000000	80.000000	38.000000	
max	4488.000000	100.000000	219.000000	96.000000	100.000000	99.000000	9

In [21]:
```
df.to_csv('week2_dataset.csv')
```