ADS 509 Sentiment Assignment

This notebook holds the Sentiment Assignment for Module 6 in ADS 509, Applied Text Mining. Work through this notebook, writing code and answering questions where required.

In a previous assignment you put together Twitter data and lyrics data on two artists. In this assignment we apply sentiment analysis to those data sets. If, for some reason, you did not complete that previous assignment, data to use for this assignment can be found in the assignment materials section of Blackboard.

General Assignment Instructions

These instructions are included in every assignment, to remind you of the coding standards for the class. Feel free to delete this cell after reading it.

One sign of mature code is conforming to a style guide. We recommend the Google Python Style Guide. If you use a different style guide, please include a cell with a link.

Your code should be relatively easy-to-read, sensibly commented, and clean. Writing code is a messy process, so please be sure to edit your final submission. Remove any cells that are not needed or parts of cells that contain unnecessary code. Remove inessential `import` statements and make sure that all such statements are moved into the designated cell.

Make use of non-code cells for written commentary. These cells should be grammatical and clearly written. In some of these cells you will have questions to answer. The questions will be marked by a "Q:" and will have a corresponding "A:" spot for you. *Make sure to answer every question marked with a* `Q:` *for full credit.*

In [2]:
```
import os
import re
import emoji
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

from collections import Counter, defaultdict
from string import punctuation

from nltk.corpus import stopwords

sw = stopwords.words("english")
```

In [3]:
```
# Add any additional import statements you need here
from pathlib import Path
```

In [4]:
```
# change `data_location` to the location of the folder on your machine.
data_location = Path("../datasets")

# These subfolders should still work if you correctly stored the
# data from the Module 1 assignment
twitter_folder = "twitter/"
lyrics_folder = "lyrics/"

positive_words_file = "positive-words.txt"
negative_words_file = "negative-words.txt"
tidy_text_file = "tidytext_sentiments.txt"
```

Data Input

Now read in each of the corpora. For the lyrics data, it may be convenient to store the entire contents of the file to make it easier to inspect the titles individually, as you'll do in the last part of the assignment. In the solution, I stored the lyrics data in a dictionary with two dimensions of keys: artist and song. The value was the file contents. A Pandas data frame would work equally well.

For the Twitter data, we only need the description field for this assignment. Feel free all the descriptions read it into a data structure. In the solution, I stored the descriptions as a dictionary of lists, with the key being the artist.

Read in Lyrics data

In [5]:
```
# Read in the lyrics data
df_lyrics = pd.read_pickle(data_location / "df_lyrics_cleaned.pkl")
```

Read in Twitter data

In [6]:
```
# Read in the twitter data
df_twitter = pd.read_pickle(data_location / "df_twitter_cleaned.pkl")
```

Custom Functions for Sentiment Dictionary

In [ ]:
```
def check_sentiment_value(word, current_value, new_value, verbose=False):
    """
    Resolve a word's sentiment score when encountering potential conflicts.

    Given an existing sentiment score (`current_value`) and a proposed new
    score (`new_value`) for the same word, this function applies the
    following rules:

    1) If the current value is already 0 (previously marked conflicting),
       keep it as 0.
    2) If the current value differs from the new value (e.g., +1 vs -1),
       mark as conflicting by setting the score to 0.
```

```
    3) Otherwise (values match), keep the new value.

    When `verbose=True`, a human-readable message describing the decision
    is printed.

    Parameters
    ----------
    word : str
        The token/term being evaluated.
    current_value : int
        The word's existing sentiment score in the dictionary. Expected
        values are typically {-1, 0, 1}, where 0 denotes a conflict found
        earlier.
    new_value : int
        The proposed sentiment score for the word (usually -1 or 1).
    verbose : bool, optional
        If True, prints a message describing the resolution. Defaults to False.

    Returns
    -------
    int
        The resolved sentiment score for the word:
        - 0 if a conflict is detected or already present,
        - `new_value` if it matches `current_value`,
        - `current_value` if it is already 0.

    Notes
    -----
    - Use this helper during dictionary construction to ensure that once a
      word is marked as conflicting (0), it remains 0 on subsequent passes.
    - ChatGPT assisted in creating this docstring.

    Examples
    --------
    >>> check_sentiment_value("happy", 1, 1)
    1
    >>> check_sentiment_value("sick", 1, -1)
    0
    >>> check_sentiment_value("wicked", 0, 1)
    0
    """
    message = ""
    score_value = None
    if current_value == 0:
        message = "Conflicting sentiment value already determined!"
        score_value = current_value
    elif current_value != new_value:
        message = f"Conflict: {current_value} vs {new_value}."
        score_value = 0
    else:
        message = "Exists but no conflict in sentiment value."
        score_value = new_value

    if verbose:
        message_0 = f"{word}: "
        message_value = f"Value set to {score_value}."
```

```
        print(message_0 + message + message_value)

    return score_value


def create_scoring_dict(dictionary_name, text_file_of_words, base_value, verbose=Fa
    """
    Populate a sentiment scoring dictionary from a plain-text word list.

    Reads a text file (one word per line) and inserts each cleaned, lowercased
    word into `dictionary_name` with the given `base_value` (e.g., +1 for a
    positive lexicon, -1 for a negative lexicon). Lines starting with ';'
    are treated as comments and skipped. If a word already exists in the
    dictionary, the final score is resolved via `check_sentiment_value`
    to handle conflicts (e.g., a word appearing in both positive and negative lists
    once a word has been marked conflicting (score 0), it remains 0.

    Parameters
    ----------
    dictionary_name : dict[str, int]
        The mapping of words to sentiment scores. Modified in place.
        Typical scores are {-1, 0, 1}.
    text_file_of_words : str
        Path to a newline-delimited word list. Each non-empty, non-comment
        line should contain a single token.
    base_value : int
        The base sentiment score to assign to each word from this file
        (commonly +1 for positive or -1 for negative).
    verbose : bool, optional
        If True, passes through to `check_sentiment_value` to print
        per-word resolution messages. Default is False.

    Returns
    -------
    None
        This function modifies `dictionary_name` in place and returns None.

    Notes
    -----
    - Words are cleaned with `strip().lower()` before insertion.
    - Comment lines (prefixed with ';') and blank lines are ignored.
    - Conflict resolution is delegated to `check_sentiment_value`.
    - ChatGPT assisted in creating this docstring.

    Examples
    --------
    >>> word_dict = {}
    >>> create_scoring_dict(word_dict, "negative-words.txt", base_value=-1)
    >>> create_scoring_dict(word_dict, "positive-words.txt", base_value=+1)
    >>> word_dict.get("terrible"), word_dict.get("great")
    (-1, 1)
    """
    score_value = base_value
    with open(text_file_of_words, "r", encoding="utf-8") as f:
        for line in f:
            word = line.strip().lower()
```

```
            if word and not word.startswith(";"):
                word_score = score_value
                if word in dictionary_name:
                    word_score = check_sentiment_value(
                        word, dictionary_name[word], score_value, verbose
                    )
                dictionary_name[word] = word_score

    return None


def tidytext_scoring(
    dictionary_name, text_file_of_words="tidytext_sentiments.txt", verbose=False
):
    """
    Augment a sentiment dictionary using a Tidytext-style lexicon file.

    Reads a whitespace-delimited file whose first line is a header and whose
    rows contain at least two columns: the token (column 0) and its sentiment
    label (column 1). The label "negative" is mapped to -1; any other label
    is mapped to +1. Each token is lowercased before lookup/insertion. If a
    token already exists in `dictionary_name`, conflicts are resolved via
    `check_sentiment_value` (e.g., positive vs. negative → set to 0).

    Parameters
    ----------
    dictionary_name : dict[str, int]
        Mapping from token to sentiment score. Modified in place. Typical
        values are {-1, 0, 1}, where 0 indicates a previously detected conflict.
    text_file_of_words : str, optional
        Path to the Tidytext-style sentiment file. Must contain a header line
        followed by rows with at least "word" and "sentiment" columns.
        Defaults to "tidytext_sentiments.txt".
    verbose : bool, optional
        If True, passes through to `check_sentiment_value` to print per-token
        resolution messages. Defaults to False.

    Returns
    -------
    None
        The function updates `dictionary_name` in place and returns None.

    Notes
    -----
    - Assumes whitespace-delimited columns and a single header line (skipped).
    - Tokens are normalized with `lower()`.
    - Conflict handling is delegated to `check_sentiment_value`.
    - ChatGPT assisted in creating this docstring.

    Examples
    --------
    >>> d = {"great": 1}
    >>> tidytext_scoring(d, "tidytext_sentiments.txt")
    >>> d.get("awful"), d.get("great")
    (-1, 1)
    """
```

```
    with open(text_file_of_words, "r", encoding="utf-8") as f:
        next(f)  # skip header
        for line in f:
            words = line.split()
            if words[1] == "negative":
                score_value = -1
            else:
                score_value = 1
            word = words[0].lower()
            if word in dictionary_name:
                score_value = check_sentiment_value(
                    word, dictionary_name[word], score_value, verbose
                )
            dictionary_name[word] = score_value

    return None


def clean_sentiment_dictionary(dictionary_name, conflicting_words_list):
    """
    Remove ambiguous (conflicting) entries from a sentiment dictionary.

    This function scans `dictionary_name` for words whose sentiment score is
    `0` (used to mark conflicts, e.g., words that appeared in both positive
    and negative lists). It removes those words from the dictionary and
    appends each removed word to `conflicting_words_list` for auditing or
    later review.

    Parameters
    ----------
    dictionary_name : dict[str, int]
        Mapping of words to sentiment scores. This dictionary is modified
        in place; any key with value `0` is deleted.
    conflicting_words_list : list[str]
        A list that will be extended with the words removed due to conflict.

    Returns
    -------
    None
        The function performs in-place modifications and prints a short
        summary of how many words were removed.

    Notes
    -----
    - Iterates over a snapshot of keys (`list(dictionary_name.keys())`) to
      safely delete items during traversal.
    - Use this after building your lexicon to ensure that ambiguous words
      don't contribute noise to sentiment scoring.
    - ChatGPT assisted in creating this docstring.

    Examples
    --------
    >>> d = {"happy": 1, "sad": -1, "sick": 0}
    >>> removed = []
    >>> clean_sentiment_dictionary(d, removed)
    Total words removed from: 1
```

```
    >>> d
    {'happy': 1, 'sad': -1}
    >>> removed
    ['sick']
    """
    for key in list(dictionary_name.keys()):
        if dictionary_name[key] == 0:
            conflicting_words_list.append(key)
            del dictionary_name[key]
    print(f"Total words removed from: {len(conflicting_words_list)}")


def sentiment_score(tokens):
    """
    Compute the average lexicon sentiment score for a sequence of tokens.

    Each token found in the global sentiment dictionary ``word_dict`` contributes
    its score (e.g., -1 for negative, +1 for positive). Tokens not present in
    ``word_dict`` contribute 0 implicitly (i.e., are ignored). The final score is
    the sum of matched token scores divided by the total number of input tokens.

    Parameters
    ----------
    tokens : Sequence[str]
        Iterable of tokenized words to score.

    Returns
    -------
    float
        The average sentiment score over ``tokens``:
        ``sum(word_dict[w] for w in tokens if w in word_dict) / len(tokens)``.

    Raises
    ------
    ZeroDivisionError
        If ``tokens`` is empty. Guard with ``if not tokens: return 0.0`` if desired

    Notes
    -----
    - Relies on a global ``word_dict`` mapping tokens to integer scores.
    - Out-of-vocabulary tokens (not in ``word_dict``) do not change the sum.
    - ChatGPT assisted in creating this docstring.

    Examples
    --------
    >>> word_dict = {"great": 1, "bad": -1}
    >>> sentiment_score(["great", "movie"])
    0.5
    >>> sentiment_score(["bad", "bad", "plot"])
    -0.6666666666666666
    """
    sentiment_score = 0
    for word in tokens:
        if word in word_dict:
            sentiment_score += word_dict[word]
    return sentiment_score / len(tokens)
```

```
def display_lyrics(dataframe):
    """
    Print each song's title and lyrics from a DataFrame.

    Expects a DataFrame with columns **'song_title'** and **'lyrics'**.
    For each row, prints the title, then the lyrics, followed by a
    60-character separator line. If the lyrics are stored as a list of
    tokens, they are joined with spaces; otherwise they are converted to
    string as-is.

    Parameters
    ----------
    dataframe : pandas.DataFrame
        Input table containing at least the columns 'song_title' and 'lyrics'.

    Returns
    -------
    None
        Outputs formatted text to stdout for each row.

    Notes
    -----
    - Long lyrics may be truncated by the notebook/console display settings.
      To avoid truncation in pandas display contexts, you can set:
      ``pd.set_option('display.max_colwidth', None)``.
    - ChatGPT assisted in creating this docstring.

    Examples
    --------
    >>> display_lyrics(df_cher.head(3))
    My Love
    when i go away i know my heart...
    ------------------------------------------------------------
    """
    for title, tokens, score in dataframe[
        ["song_title", "lyrics", "sentiment_score"]
    ].itertuples(index=False):
        text = " ".join(tokens) if isinstance(tokens, list) else str(tokens)
        data = display_counter(tokens)
        print(f"{title} (Sentiment score: {score})\n")
        print(f"{text}\n")
        print(data)
        print("-" * 60)


def display_counter(tokens, top=5, header="words", header2="count", show=False):
    """
    Summarize token frequencies and optionally display them as a DataFrame.

    Uses ``collections.Counter`` to compute the most common tokens, creates a
    pandas ``DataFrame`` with customizable column headers, and either displays
    it (pretty table) or returns the raw list of (token, count) pairs.

    Parameters
```

```
    ----------
    tokens : Sequence[str]
        Iterable of tokens to count.
    top : int, optional
        Number of most frequent tokens to include (default: 10).
    header : str, optional
        Column name for the token column in the output DataFrame (default: "words")
    header2 : str, optional
        Column name for the frequency column in the output DataFrame (default: "cou
    show : bool, optional
        If True, prints a header line and displays the DataFrame, returning ``None`
        If False, returns the list of (token, count) tuples instead. Default is Tru

    Returns
    -------
    list[tuple[str, int]] or None
        The top-``top`` (token, count) pairs when ``show=False``; otherwise ``None`

    Notes
    -----
    - Requires ``import pandas as pd`` and, for pretty display in notebooks,
      ``from IPython.display import display``.
    - If ``tokens`` is empty, the resulting summary is empty.
    - ChatGPT assisted in creating this docstring.

    Examples
    --------
    >>> tokens = ["a", "b", "a", "c", "b", "a"]
    >>> display_counter(tokens, top=2, show=False)
    [('a', 3), ('b', 2)]
    >>> display_counter(tokens, top=2, header="token", header2="freq", show=True)
    # prints: "The top 2 most common token:" and displays a 2-row DataFrame
    """
    data = Counter(tokens).most_common(top)
    df = pd.DataFrame(data, columns=[header, header2])

    if show:
        print(f"\nThe top {top} most common {header}:")
        display(df)
        return None
    else:
        return data
```

In [8]:

```
# Read in the positive and negative words and the
# tidytext sentiment. Store these so that the positive
# words are associated with a score of +1 and negative words
# are associated with a score of -1. You can use a dataframe or a
# dictionary for this.
pos_score = 1
neg_score = -1
word_dict = {}
conflict_words = []
set_verbose = False

create_scoring_dict(word_dict, "negative-words.txt", neg_score, verbose=set_verbose
```

```
create_scoring_dict(word_dict, "positive-words.txt", pos_score, verbose=set_verbose
tidytext_scoring(word_dict, verbose=set_verbose)
```

In [9]:
```
clean_sentiment_dictionary(word_dict, conflict_words)
```

```
Total words removed from: 131
```

Sentiment Analysis on Songs

In this section, score the sentiment for all the songs for both artists in your data set. Score the sentiment by manually calculating the sentiment using the combined lexicons provided in this repository.

After you have calculated these sentiments, answer the questions at the end of this section.

In [10]:
```
# your code here
df_lyrics["sentiment_score"] = df_lyrics["lyrics"].apply(sentiment_score).round(5)

df_lyrics.sample(5)
```

Out[10]:

	artist	song_title	lyrics	no_sw	sentiment_score
309	cher	Don't Come Around Tonight	[dont, come, around, tonight, i, bring, a, pap...	[dont, come, around, tonight, bring, paper, ca...	0.02727
68	cher	I Want You	[guilty, undertaker, sighs, lonesome, organ, g...	[guilty, undertaker, sighs, lonesome, organ, g...	-0.02966
27	cher	The Same Mistake	[crash, the, car, this, road, is, going, nowhe...	[crash, car, road, going, nowhere, fast, burn,...	0.02479
135	cher	Favourite Scars	[best, jump, like, do, we, feel, were, going, ...	[best, jump, like, feel, going, nowhere, unles...	0.06490
51	cher	Chastity's Song (Band Of Thieves)	[wind, up, when, a, band, of, thieves, making,...	[wind, band, thieves, making, way, across, tow...	0.06926

In [11]:
```
# Calculate average sentiment score
avg_cher = df_lyrics.loc[df_lyrics["artist"] == "cher", "sentiment_score"].mean()
avg_robyn = df_lyrics.loc[df_lyrics["artist"] == "robyn", "sentiment_score"].mean()

print(f"Artist: Cher \nAverage sentiment score: {avg_cher:0.5f}")
print("\n")
print(f"Artist: Robyn \nAverage sentiment score: {avg_robyn:0.5f}")
```

```
Artist: Cher
Average sentiment score: 0.02701


Artist: Robyn
Average sentiment score: 0.03341
```

Cher's Song Lyrics

In [12]:
```python
df_cher = df_lyrics.loc[df_lyrics["artist"] == "cher"].sort_values(
    "sentiment_score", ascending=False
)
cher_t3 = df_cher.head(3).copy()
cher_b3 = df_cher.tail(3).copy()

print("Top 3 Sentiment")
print("=" * 60)
display_lyrics(cher_t3)

print("Bottom 3 Sentiment")
print("=" * 60)
display_lyrics(cher_b3)
```

```
Top 3 Sentiment
============================================================
My Love (Sentiment score: 0.23889)

when i go away i know my heart can stay with my love its understood everywhere with
my love my love does it good whoa my love oh only my love my love does it good and w
hen the cupboards bare ill still find something there with my love its understood ev
erywhere with my love my love does it so good whoa my love oh only my love my love d
oes it good only my love oh only my love only my love hold the other things to me oh
only my love oh only my love only my love does it good to me everywhere with my love
dont you see my love it does it so good oh only my love only my love my love does it
good dont ever ask me why i never say goodbye to my love its understood cause everyw
here with my love my love does it good whoa only my love oh only my love my love doe
s it good oh only my love only my love does it good to mewhoa

[('my', 32), ('love', 31), ('only', 14), ('it', 11), ('does', 10)]
------------------------------------------------------------
Love And Understanding (Sentiment score: 0.20863)

here here in this world where do we go where can we turn when we need some love it s
eems that love just cant be found where where do we stand when loves supply dont mee
t loves demand we got enough stars to light the sky at night enough sun to make to m
ake the whole world bright we got more than enough but theres one thing theres just
not enough of not enough love and understanding we could use some love to ease these
troubled times not enough love and understanding why oh why spend all of our time bu
ilding buildings up to the sky reaching everywhere but where we need to reach the mo
st hearts never can win oh in this race this race that were in weve got enough cars
to drive around the world enough planes to take us anywhere we got more than enough
but theres one thing theres just not enough of not enough love and understanding we
could use some love to ease these troubled times not enough love and understanding w
hy oh why not enough love and understanding we could use some love to ease these tro
ubled times not enough love and understanding why oh why we need some understandin w
e need a little more love some love and understandin enough stars to light the sky a
t night enough sun to make the whole world bright enough hearts to find some love in
side we got more than enough but theres one thing theres just not enough of not enou
gh love and understanding we could use some love to ease these troubled times not en
ough love and understanding why oh why

[('enough', 21), ('love', 17), ('we', 15), ('to', 14), ('not', 11)]
------------------------------------------------------------
Sunny (Sentiment score: 0.1989)

sunny yesterday my life was filled with rain sunny you smiled at me and then it ease
d my pain now the dark days are gone and bright days are here my sunny once shined s
o sincere sunny once so true i love you i love you sunny thank you for the sunshine
you gave to me sunny thank you for the love you brought my way you gave to me your a
ll and all and now i feel that im 10 feet tall sunny once so true i love you sunny t
hank you for the truth that you let me see sunny thank you for the facts from a to z
somehow i was torn like a wind blown sail then our love was borned when you held my
hand sunny once so true i love you sunny thank you for the sunshine you gave to me s
unny thank you for the love you brought my way you gave to me your all and all and n
ow i feel that im 10 feet tall sunny once so true i love you

[('you', 20), ('sunny', 13), ('i', 8), ('love', 8), ('the', 7)]
------------------------------------------------------------
Bottom 3 Sentiment
```

```
============================================================
Outrageous (Sentiment score: -0.12821)

outrageous outrageous they say im outrageous its the rage im gonna wear what i will
and spend some and i will be dress to kill dontcha know and when the lights come up
im ready im ready to put on a show with class and if i clash its cause i want to wha
t a show and i want everyone to know theyre gonna fly up get an eyeful everything th
ats craved from me im gonna be im gonna be outrageous outrageous they say im outrage
ous its the rage its the rage with my long black hair hanging way down to my ask me
no questions and ill tell you no lies dont tell me what to do dont tell me what to b
e see i dont trust anybody elses traits about makeup and me well in my show i let ev
erything go is what you want is whatcha wanna see from me but when the curtain comes
down and youre on your way back home i change into my jeans that are split at the se
am i grab my funky black jacket and make quite a racket you drive like youre an outl
aw cause everything thats craved from me im gonna be im gonna be outrageous so outra
geous im outrageous honey yes a rage its the rage outrageous outrageous im outrageou
s its the rage its a rage outrageous outrageous they say im outrageous

[('outrageous', 15), ('im', 12), ('i', 9), ('and', 9), ('the', 8)]
------------------------------------------------------------
Bang Bang (My Baby Shot Me Down) (Sentiment score: -0.20122)

i was five and he was six we rode on horses made of sticks he wore black and i wore
white he would always win the fight bang bang he shot me down bang bang i hit the gr
ound bang bang that awful sound bang bang my baby shot me down seasons came and chan
ged the time when i grew up i called him mine he would always laugh and say remember
when we used to play bang bang i shot you down bang bang you hit the ground bang ban
g that awful sound bang bang i used to shoot you down music played and people sang j
ust for me the church bells rang now hes gone i dont know why and til this day somet
imes i cry he didnt even say goodbye he didnt take the time to lie bang bang he shot
me down bang bang i hit the ground bang bang that awful sound bang bang my baby shot
me down

[('bang', 24), ('i', 10), ('he', 8), ('the', 7), ('and', 6)]
------------------------------------------------------------
Bang-Bang (Sentiment score: -0.25912)

bang bang you shot me down bang bang i hit the ground bang bang that awful sound ban
g bang my baby shot me down i was five and you were six we rode on horses made of st
icks i wore black you wore white you would always win the fight bang bang you shot m
e down bang bang i hit the ground bang bang that awful sound bang bang my baby shot
me down seasons came and changed the time i grew up i called you mine you would alwa
ys laugh and say remember when we used to play bang bang you shot me down bang bang
and i hit the ground bang bang that awful sound bang bang my baby shot me down music
played and people sang just for me the church bells rang after echoes from a gun we
both vowed that wed be one now youre gone i dont know why sometimes i cry you didnt
say goodbye you didnt take the time to lie bang bang you shot me down bang bang i hi
t the ground bang bang that awful sound bang bang my baby shot me down bang bang you
shot me right between my eyes bang bang you cant go paralyzed bang bang bang bang ba
ng bang bang bang oh baby im laying on the ground bang bang ill never come around ba
ng bang bang bang oh baby bang bang oh baby come and wrap me bang bang you see how s
weet its gonna be bang bang bang bang bang bang oh my baby my baby shot me down bang
bang im up on the ground now

[('bang', 62), ('you', 14), ('me', 12), ('shot', 10), ('i', 10)]
------------------------------------------------------------
```

Robyn's Song Lyrics

In [13]:
```
df_robyn = df_lyrics.loc[df_lyrics["artist"] == "robyn"].sort_values(
    "sentiment_score", ascending=False
)
robyn_t3 = df_robyn.head(3).copy()
robyn_b3 = df_robyn.tail(4).copy()

print("Top 3 Sentiment")
print("=" * 60)
display_lyrics(robyn_t3)

print("Bottom 3 Sentiment")
print("=" * 60)
display_lyrics(robyn_b3)
```

Top 3 Sentiment
==========
Baby Forgive Me (Sentiment score: 0.27083)

here come the night in your eyes baby be brave be wise its up to you you do what you like mmm wont you give it a chance baby just one more try baby forgive me baby forgive me baby forgive me baby forgive me wont you give it a chance baby wont you give it a chance baby just one more try one more try yeah eh you got the power you set the price but baby be fair be nice you say you want to be happy then you got to put your love on the line just let me make you smile again baby i know we can work it out yes i know we can baby forgive me yeah baby forgive me baby forgive me baby forgive me baby forgive me baby forgive me baby forgive me baby forgive me

[('baby', 18), ('me', 13), ('you', 12), ('forgive', 12), ('be', 5)]

Love Is Free (Sentiment score: 0.25773)

free love is free baby free love is free baby boom boom boom boom boom chica boom let me give it to you baby chica boom chica boom chica boom chica boom chica boom chica boom free love is free baby free love is free baby boom boom boom boom boom chica boom let me give it to you baby ima give it to you baby ima give it when im ready some lose some steady some light some heavy its all over this city sometimes in the ugly sometimes in the pretty you never know where you get it cause you cant control it and you cant unfold it slow down ima give it to you baby ima give it when im ready some lose some steady some light some heavy its all over this city sometimes in the ugly sometimes in the pretty you never know where you get it cause you cant control it and you cant unfold it slow down free love is free baby free love is free baby boom boom boom boom boom chica boom let me give it to you baby yeah yeah yeah yeah love is free baby love is free baby ima give it to you baby ima give it like a motha safe like a rubber mutter like a stutter its all over this city sometimes in the nitty sometimes in the gritty you know ima give it to you better ima give you love forever no you cant control it slow down ima give it to you baby ima give it like a motha safe like a rubber mutter like a stutter its all over this city sometimes in the nitty sometimes in the gritty you know ima give it to you better ima give you love forever no you cant control it slow down free love is free baby free love is free baby boom boom boom boom boom chica boom let me give it to you baby boom boom boom boom boom chica boom chica boom chica boom chica boom chica boom free love is free baby free love is free baby aprã©ndelo aprã©ndelo te lo digo right now sã³lo entiã©ndelo aprã©ndelo aprã©ndelo te lo digo right now sã³lo entiã©ndelo slow down aprã©ndelo aprã©ndelo te lo digo right now sã³lo entiã©ndelo aprã©ndelo aprã©ndelo te lo digo right now sã³lo entiã©ndelo slow down ima give it to you baby ima give it when im ready some lose some steady some light some heavy its all over this city sometimes in the ugly sometimes in the pretty you never know where you get it cause you cant control it and you cant unfold it slow down slow down x6 free love is free baby love is free baby so free x4 you know love is free so free free â love is free baby x8 so free

[('boom', 46), ('free', 31), ('you', 30), ('it', 27), ('baby', 24)]

Between The Lines (Sentiment score: 0.14846)

even reading in between the lines every day you hit my phone up every time you hit my phone up it makes my heart jump i want you to say it baby it makes my heart jump say it like you mean it its right there on the tip of your tongue you hit my phone up and i can feel it and i like it baby dont you stop dont stop what youre doing baby you know i like it you stretch it out you know i like it you give massages its not yo

ur words you know i like it its whats in between them theres no need to spell out it makes my heart jump i know you want it and i like it baby im reading in between the lines im reading in between the lines you got me reading in between the lines im rea ding in between the lines even reading in between the lines im reading in between th e lines you got me reading in between the lines and i like it baby and i like it bab y even reading in between the lines i dont mind i dont really mind it baby when we g et silent pressures rising its so intense when we get silent were making diamonds th eres no need to say it baby were making diamonds cause i can tell you mean it all ov er the tip of your tongue were making diamonds you got me spaced out and i like it b aby im reading in between the lines im reading in between the lines you got me readi ng in between the lines im reading in between the lines even reading in between the lines im reading in between the lines you got me reading in between the lines and i like it baby and i like it baby even reading in between the lines and i like it baby and i like it and i like it baby and i like it relax thats amazing you got me readin g in between the lines that shits so good relax you got me reading in between the li nes amazing

[('it', 26), ('the', 21), ('in', 20), ('between', 20), ('you', 20)]

Bottom 3 Sentiment
==========
Robotboy (Sentiment score: -0.08235)

hey now boy where you been smashed up toy are you lost again your circuits blown wil l you find your coordinates home your batterys low did you crash again robot boy do you need a friend hey little droid is your head on wrong hey little prince youve los t control calendar boy are you growing old your radars jammed shut your lasers down while you can hush now boy please give in robot boy youve reached the end hey little droid let your xray shine

[('you', 7), ('your', 7), ('boy', 5), ('hey', 4), ('little', 3)]

Criminal Intent (Sentiment score: -0.1167)

somebody alert the authorities i got criminal intent conspiracy to engage in lewd an d indecent acts and events imma wind it grind it oh my imma say it again somebody al ert the authorities shes got criminal intent somebody alert the authorities i got cr iminal intent conspiracy to engage in lewd and indecent acts and events imma wind it grind it oh my imma say it again somebody alert the authorities shes got criminal in tent somebody alert the authorities i got criminal intent conspiracy to engage in le wd and indecent acts and events imma wind it grind it oh my imma say it again somebo dy alert the authorities i got criminal intent will the defendant please rise state your full name for the record robyn do you wish to say anything before the sentence is imposed i do your honor you know from time to time i need to get down unwind and just bump and grind get my shot on have some fun a little dirty never hurt anyone i admit i can get somewhat xrated on the floor but your honor hows that something you get incarcerated for iii done nothing thats wrong something thats frowned upon i obj ect most strongly judge they played my song somebody alert the authorities i got cri minal intent conspiracy to engage in lewd and indecent acts and events imma wind it grind it oh my imma say it again somebody alert the authorities shes got criminal in tent somebody alert the authorities i got criminal intent conspiracy to engage in le wd and indecent acts and events imma wind it grind it oh my imma say it again somebo dy alert the authorities i got criminal intent somebody alert the authorities shes g ot criminal intent yyoyo judge may the record reflect the fact i dont have any prior s besides would you pardon me for being inappropriately attired but yo listen them c uffs down at county totally ripped up my pantyhose and some snitch punk at legal aid

```
stole my voucher for court clothes i done nothing thats wrong something thats frowne
d upon i object most strongly judge they played my song somebody alert the authoriti
es i got criminal intent conspiracy to engage in lewd and indecent acts and events i
mma wind it grind it oh my imma say it again somebody alert the authorities shes got
criminal intent somebody alert the authorities i got criminal intent conspiracy to e
ngage in lewd and indecent acts and events imma wind it grind it oh my imma say it a
gain somebody alert the authorities i got criminal intent somebody alert the authori
ties shes got criminal intent

[('the', 22), ('it', 21), ('i', 18), ('and', 17), ('somebody', 16)]
------------------------------------------------------------
Don't Fucking Tell Me What To Do (Sentiment score: -0.22901)

my drinking is killing me my drinking is killing me my drinking is killing me my dri
nking is killing me my drinking is killing me my drinking is killing me my drinking
is killing me my drinking is killing me my drinking is killing me my drinking is kil
ling me my drinking is killing me my drinking is killing me my drinking is killing m
e my drinking is killing me my drinking is killing me my smoking is killing me my di
et is killing me my heels are killing me my shoppings killing me my ego is killing m
e cant sleep its killing me my labels killing me kickdrum my phone is killing me my
email is killing me these hours are killing me my tour is killing me this flight is
killing me my managers killing me my mothers killing me my landlords killing me my b
oss is killing me the tv is killing me your nagging is killing me my boyfriends kill
ing me my talkings killing me killing me killing me cant sleep its killing me my dre
ams are killing me tv is killing me my talkings killing me let go youre killing me e
ase up youre killing me calm down youre killing me my god youre killing me my drinki
ng is killing me my smoking is killing me my head is killing me my mind is killing m
e my back is killing me my neck is killing me your nagging is killing me my gut is k
illing me my pms is killing me my email is killing me these hours are killing me my
tour is killing me this flight is killing me my managers killing me my mothers killi
ng me my landlords killing me my smoking is killing me the tv is killing me your nag
ging is killing me ease up youre killing me let go youre killing me calm down youre
killing me my god youre killing me dont fucking tell me what to do do dont fucking t
ell me what to do do do do do dont fucking tell me what to do dont fucking tell me w
hat to do do dont fucking tell me what to do do dont fucking tell me what to do dont
fucking tell me what to do dont fucking tell me what to do dont fucking tell me what
to do

[('me', 77), ('killing', 68), ('my', 48), ('is', 41), ('drinking', 16)]
------------------------------------------------------------
Don't Fucking Tell Me What To Do (Sentiment score: -0.22901)

my drinking is killing me my drinking is killing me my drinking is killing me my dri
nking is killing me my drinking is killing me my drinking is killing me my drinking
is killing me my drinking is killing me my drinking is killing me my drinking is kil
ling me my drinking is killing me my drinking is killing me my drinking is killing m
e my drinking is killing me my drinking is killing me my smoking is killing me my di
et is killing me my heels are killing me my shoppings killing me my ego is killing m
e cant sleep its killing me my labels killing me kickdrum my phone is killing me my
email is killing me these hours are killing me my tour is killing me this flight is
killing me my managers killing me my mothers killing me my landlords killing me my b
oss is killing me the tv is killing me your nagging is killing me my boyfriends kill
ing me my talkings killing me killing me killing me cant sleep its killing me my dre
ams are killing me tv is killing me my talkings killing me let go youre killing me e
ase up youre killing me calm down youre killing me my god youre killing me my drinki
ng is killing me my smoking is killing me my head is killing me my mind is killing m
```

```
e my back is killing me my neck is killing me your nagging is killing me my gut is k
illing me my pms is killing me my email is killing me these hours are killing me my
tour is killing me this flight is killing me my managers killing me my mothers killi
ng me my landlords killing me my smoking is killing me the tv is killing me your nag
ging is killing me ease up youre killing me let go youre killing me calm down youre
killing me my god youre killing me dont fucking tell me what to do do dont fucking t
ell me what to do do do do do dont fucking tell me what to do dont fucking tell me w
hat to do do dont fucking tell me what to do do dont fucking tell me what to do dont
fucking tell me what to do dont fucking tell me what to do dont fucking tell me what
to do

[('me', 77), ('killing', 68), ('my', 48), ('is', 41), ('drinking', 16)]
------------------------------------------------------------
```

Questions

Q: Overall, which artist has the higher average sentiment per song?

A: Robyn has a higher average sentiment per song at 0.03341. Cher's lyrics are 0.02701.

Q: For your first artist, what are the three songs that have the highest and lowest sentiments? Print the lyrics of those songs to the screen. What do you think is driving the sentiment score?

A: Cher's Song Lyrics
Top 3 Sentiment:

- My Love (0.23889)
- Love and Understanding (0.20863)
- Sunny (0.19890)

Bottom 3 Sentiment:

- Outrageous (-0.12821)
- Bang Bang (My Baby Shot Me Down) (-0.20122)
- Bang-Bang (-0.25912)

There are a lot of repeated words in these song lyrics which contribute to the sentiment score such as the following: love, enough, sunny, outrageous.

Q: For your second artist, what are the three songs that have the highest and lowest sentiments? Print the lyrics of those songs to the screen. What do you think is driving the sentiment score?

A: Robyn's Song Lyrics
Top 3 Sentiment:

- Baby Forgive Me (0.27083)
- Love Is Free (0.25773)

- Between The Lines (0.14846)

Bottom 3 Sentiment:

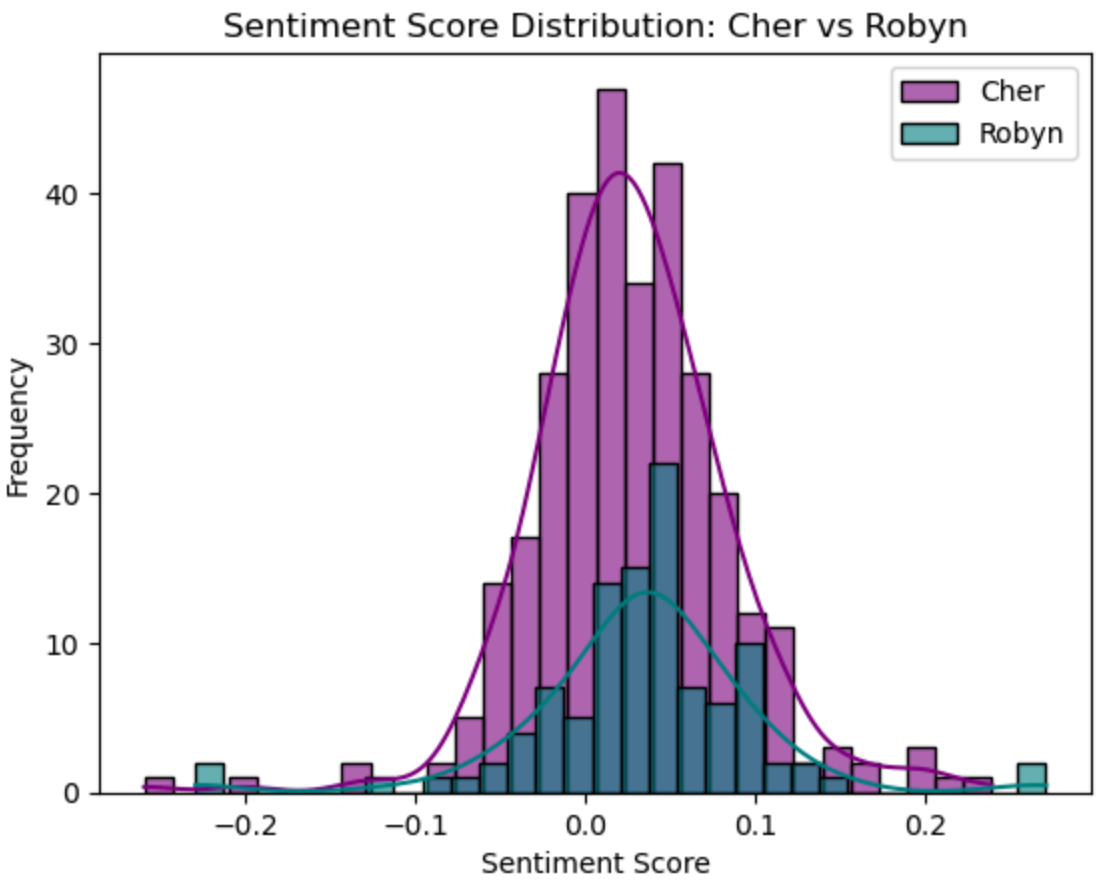
- Robotboy (-0.08235)
- Criminal Intent (-0.1167)
- Don't Fucking Tell Me What To Do (-0.22901)

Similar to the Cher's lyrics, words are repeated over and over in the lyrics which contribute to the sentiment score.

Q: Plot the distributions of the sentiment scores for both artists. You can use `seaborn` to plot densities or plot histograms in matplotlib.

In [14]:
```
# plot cher
sns.histplot(
    df_lyrics[df_lyrics["artist"] == "cher"]["sentiment_score"],
    bins=30,
    kde=True,
    color="purple",
    label="Cher",
    alpha=0.6,
)
# plot robyn
sns.histplot(
    df_lyrics[df_lyrics["artist"] == "robyn"]["sentiment_score"],
    bins=30,
    kde=True,
    color="teal",
    label="Robyn",
    alpha=0.6,
)

plt.title("Sentiment Score Distribution: Cher vs Robyn")
plt.xlabel("Sentiment Score")
plt.ylabel("Frequency")
plt.legend()
plt.show()
```



Sentiment Analysis on Twitter Descriptions

In this section, define two sets of emojis you designate as positive and negative. Make sure to have at least 10 emojis per set. You can learn about the most popular emojis on Twitter at the emojitracker.

Associate your positive emojis with a score of +1, negative with -1. Score the average sentiment of your two artists based on the Twitter descriptions of their followers. The average sentiment can just be the total score divided by number of followers. You do not need to calculate sentiment on non-emoji content for this section.

In [ ]:
```
# The code for the positive/negative emoji lists was create with the
# assistance of ChatGPT.
positive_keywords = [
    "smile",
    "heart",
    "grin",
    "laugh",
    "sparkle",
    "star",
    "hug",
    "clap",
    "fire",
    "party",
    "rainbow",
```

```
    "sun",
    "double exclamation mark",
    "bullseye",
]
negative_keywords = [
    "cry",
    "sad",
    "angry",
    "fear",
    "skull",
    "frown",
    "cold",
    "tired",
    "sick",
    "eyes",
    "cross mark",
    "broken heart",
    "question mark",
    "pile of poo",
]

positive_emojis = [
    e
    for e, meta in emoji.EMOJI_DATA.items()
    if any(k in meta["en"].lower() for k in positive_keywords)
]

negative_emojis = [
    e
    for e, meta in emoji.EMOJI_DATA.items()
    if any(k in meta["en"].lower() for k in negative_keywords)
]


def create_emoji_scoring_dict(
    dictionary_name, list_of_emojis, base_value, verbose=False
):
    score_value = base_value
    for emoji in list_of_emojis:
        emoji_score = score_value
        if emoji in dictionary_name:
            emoji_score = check_sentiment_value(
                emoji, dictionary_name[emoji], score_value, verbose
            )
        dictionary_name[emoji] = emoji_score

    return None


def emoji_sentiment_score(tokens):
    if len(tokens) == 0:
        return None

    sentiment_score = 0
    for word in tokens:
        if len(tokens) == 0:
```

```
            print(word, len(tokens))
        if word in emoji_dict:
            sentiment_score += emoji_dict[word]

    return sentiment_score / len(tokens)


# ChatGPT was used to assist in creating this function
def total_emoji_counts(
    dataframe, emojis_list, tokens_col="description", artist_col="artist"
):

    emoji_counts = defaultdict(Counter)

    for _, row in dataframe.iterrows():
        artist = row[artist_col]
        tokens = row[tokens_col]

        for tok in tokens:
            if tok in emojis_list:
                emoji_counts[artist][tok] += 1

    return emoji_counts
```

In [76]:
```
# your code here
emoji_dict = {}
conflicting_emojis_list = []

create_emoji_scoring_dict(emoji_dict, positive_emojis, pos_score)
create_emoji_scoring_dict(emoji_dict, negative_emojis, neg_score)

clean_sentiment_dictionary(emoji_dict, conflicting_emojis_list)

emoji_dict
```

```
Total words removed from: 5
```

```
Out[76]: {'🫀': 1,
 '💓': 1,
 '🖤': 1,
 '💙': 1,
 '💔': 1,
 '🤎': 1,
 '🎯': 1,
 '😼': 1,
 '🎬': 1,
 '👏': 1,
 '👏🏿': 1,
 '👏🏻': 1,
 '👏🏾': 1,
 '👏🏼': 1,
 '👏🏽': 1,
 '💑': 1,
 '💑🏿': 1,
 '💑🏻': 1,
 '👨\u200d❤️\u200d👨': 1,
 '👨\u200d❤\u200d👨': 1,
 '👨🏿\u200d❤️\u200d👨🏿': 1,
 '👨🏿\u200d❤\u200d👨🏿': 1,
 '👨🏿\u200d❤️\u200d👨🏻': 1,
 '👨🏿\u200d❤\u200d👨🏻': 1,
 '👨🏿\u200d❤️\u200d👨🏾': 1,
 '👨🏿\u200d❤\u200d👨🏾': 1,
 '👨🏿\u200d❤️\u200d👨🏼': 1,
 '👨🏿\u200d❤\u200d👨🏼': 1,
 '👨🏿\u200d❤️\u200d👨🏽': 1,
 '👨🏿\u200d❤\u200d👨🏽': 1,
 '👨🏻\u200d❤️\u200d👨🏻': 1,
 '👨🏻\u200d❤\u200d👨🏻': 1,
 '👨🏻\u200d❤️\u200d👨🏿': 1,
 '👨🏻\u200d❤\u200d👨🏿': 1,
 '👨🏻\u200d❤️\u200d👨🏾': 1,
 '👨🏻\u200d❤\u200d👨🏾': 1,
 '👨🏻\u200d❤️\u200d👨🏼': 1,
 '👨🏻\u200d❤\u200d👨🏼': 1,
 '👨🏻\u200d❤️\u200d👨🏽': 1,
 '👨🏻\u200d❤\u200d👨🏽': 1,
 '👨🏾\u200d❤️\u200d👨🏾': 1,
 '👨🏾\u200d❤\u200d👨🏾': 1,
 '👨🏾\u200d❤️\u200d👨🏿': 1,
 '👨🏾\u200d❤\u200d👨🏿': 1,
 '👨🏾\u200d❤️\u200d👨🏻': 1,
 '👨🏾\u200d❤\u200d👨🏻': 1,
 '👨🏾\u200d❤️\u200d👨🏼': 1,
 '👨🏾\u200d❤\u200d👨🏼': 1,
 '👨🏾\u200d❤️\u200d👨🏽': 1,
 '👨🏾\u200d❤\u200d👨🏽': 1,
 '👨🏼\u200d❤️\u200d👨🏼': 1,
 '👨🏼\u200d❤\u200d👨🏼': 1,
 '👨🏼\u200d❤️\u200d👨🏿': 1,
 '👨🏼\u200d❤\u200d👨🏿': 1,
 '👨🏼\u200d❤️\u200d👨🏻': 1,
 '👨🏼\u200d❤\u200d👨🏻': 1,
```

```
'👨🏼\u200d❤️\u200d👨🏾': 1,
'👨🏼\u200d🖤\u200d👨🏾': 1,
'👨🏼\u200d❤️\u200d👨🏽': 1,
'👨🏼\u200d🖤\u200d👨🏽': 1,
'👨🏽\u200d❤️\u200d👨🏽': 1,
'👨🏽\u200d🖤\u200d👨🏽': 1,
'👨🏽\u200d❤️\u200d👨🏿': 1,
'👨🏽\u200d🖤\u200d👨🏿': 1,
'👨🏽\u200d❤️\u200d👨🏻': 1,
'👨🏽\u200d🖤\u200d👨🏻': 1,
'👨🏽\u200d❤️\u200d👨🏾': 1,
'👨🏽\u200d🖤\u200d👨🏾': 1,
'👨🏽\u200d❤️\u200d👨🏼': 1,
'👨🏽\u200d🖤\u200d👨🏼': 1,
'👩🏿': 1,
'👩🏼': 1,
'👩🏾': 1,
'👨🏿\u200d❤️\u200d👨🏻': 1,
'👨🏿\u200d🖤\u200d👨🏻': 1,
'👨🏿\u200d❤️\u200d👨🏾': 1,
'👨🏿\u200d🖤\u200d👨🏾': 1,
'👨🏿\u200d❤️\u200d👨🏼': 1,
'👨🏿\u200d🖤\u200d👨🏼': 1,
'👨🏿\u200d❤️\u200d👨🏽': 1,
'👨🏿\u200d🖤\u200d👨🏽': 1,
'👨🏻\u200d❤️\u200d👨🏿': 1,
'👨🏻\u200d🖤\u200d👨🏿': 1,
'👨🏻\u200d❤️\u200d👨🏾': 1,
'👨🏻\u200d🖤\u200d👨🏾': 1,
'👨🏻\u200d❤️\u200d👨🏼': 1,
'👨🏻\u200d🖤\u200d👨🏼': 1,
'👨🏻\u200d❤️\u200d👨🏽': 1,
'👨🏻\u200d🖤\u200d👨🏽': 1,
'👨🏾\u200d❤️\u200d👨🏿': 1,
'👨🏾\u200d🖤\u200d👨🏿': 1,
'👨🏾\u200d❤️\u200d👨🏻': 1,
'👨🏾\u200d🖤\u200d👨🏻': 1,
'👨🏾\u200d❤️\u200d👨🏼': 1,
'👨🏾\u200d🖤\u200d👨🏼': 1,
'👨🏾\u200d❤️\u200d👨🏽': 1,
'👨🏾\u200d🖤\u200d👨🏽': 1,
'👨🏼\u200d❤️\u200d👨🏿': 1,
'👨🏼\u200d🖤\u200d👨🏿': 1,
'👨🏼\u200d❤️\u200d👨🏻': 1,
'👨🏼\u200d🖤\u200d👨🏻': 1,
'👨🏼\u200d❤️\u200d👨🏽': 1,
'👨🏼\u200d🖤\u200d👨🏽': 1,
'👨🏼\u200d❤️\u200d👨🏾': 1,
'👨🏼\u200d🖤\u200d👨🏾': 1,
'👨🏽\u200d❤️\u200d👨🏿': 1,
'👨🏽\u200d🖤\u200d👨🏿': 1,
'👨🏽\u200d❤️\u200d👨🏻': 1,
'👨🏽\u200d🖤\u200d👨🏻': 1,
'👨🏽\u200d❤️\u200d👨🏾': 1,
'👨🏽\u200d🖤\u200d👨🏾': 1,
'👨🏽\u200d❤️\u200d👨🏼': 1,
```

```
'👩🏾\u200d🖤\u200d👨🏼': 1,
'👩\u200d❤️\u200d👴': 1,
'👩\u200d🖤\u200d👴': 1,
'👧🏿\u200d❤️\u200d👦🏿': 1,
'👧🏿\u200d🖤\u200d👦🏿': 1,
'👧🏿\u200d❤️\u200d👨🏻': 1,
'👧🏿\u200d🖤\u200d👨🏻': 1,
'👧🏿\u200d❤️\u200d👦🏾': 1,
'👧🏿\u200d🖤\u200d👦🏾': 1,
'👧🏿\u200d❤️\u200d👨🏼': 1,
'👧🏿\u200d🖤\u200d👨🏼': 1,
'👧🏿\u200d❤️\u200d👦🏽': 1,
'👧🏿\u200d🖤\u200d👦🏽': 1,
'👩🏽\u200d❤️\u200d👨🏻': 1,
'👩🏽\u200d🖤\u200d👨🏻': 1,
'👩🏽\u200d❤️\u200d👦🏿': 1,
'👩🏽\u200d🖤\u200d👦🏿': 1,
'👩🏽\u200d❤️\u200d👦🏾': 1,
'👩🏽\u200d🖤\u200d👦🏾': 1,
'👩🏽\u200d❤️\u200d👨🏼': 1,
'👩🏽\u200d🖤\u200d👨🏼': 1,
'👩🏽\u200d❤️\u200d👦🏽': 1,
'👩🏽\u200d🖤\u200d👦🏽': 1,
'👧🏾\u200d❤️\u200d👧🏿': 1,
'👧🏾\u200d🖤\u200d👧🏿': 1,
'👧🏾\u200d❤️\u200d👦🏿': 1,
'👧🏾\u200d🖤\u200d👦🏿': 1,
'👧🏾\u200d❤️\u200d👨🏻': 1,
'👧🏾\u200d🖤\u200d👨🏻': 1,
'👧🏾\u200d❤️\u200d👨🏼': 1,
'👧🏾\u200d🖤\u200d👨🏼': 1,
'👧🏾\u200d❤️\u200d👦🏽': 1,
'👧🏾\u200d🖤\u200d👦🏽': 1,
'👧🏽\u200d❤️\u200d👦🏿': 1,
'👧🏽\u200d🖤\u200d👦🏿': 1,
'👧🏽\u200d❤️\u200d👦🏾': 1,
'👧🏽\u200d🖤\u200d👦🏾': 1,
'👧🏽\u200d❤️\u200d👨🏻': 1,
'👧🏽\u200d🖤\u200d👨🏻': 1,
'👧🏽\u200d❤️\u200d👨🏼': 1,
'👧🏽\u200d🖤\u200d👨🏼': 1,
'👧🏽\u200d❤️\u200d👦🏽': 1,
'👧🏽\u200d🖤\u200d👦🏽': 1,
'👩🏼\u200d❤️\u200d👨🏼': 1,
'👩🏼\u200d🖤\u200d👨🏼': 1,
'👩🏼\u200d❤️\u200d👦🏿': 1,
'👩🏼\u200d🖤\u200d👦🏿': 1,
'👩🏼\u200d❤️\u200d👨🏻': 1,
'👩🏼\u200d🖤\u200d👨🏻': 1,
'👩🏼\u200d❤️\u200d👦🏾': 1,
'👩🏼\u200d🖤\u200d👦🏾': 1,
'👩🏽\u200d❤️\u200d👦🏽': 1,
'👩🏽\u200d🖤\u200d👦🏽': 1,
'👧🏿\u200d❤️\u200d👦🏿': 1,
'👧🏿\u200d🖤\u200d👦🏿': 1,
'👧🏿\u200d❤️\u200d👨🏻': 1,
'👧🏿\u200d🖤\u200d👨🏻': 1,
'👧🏿\u200d❤️\u200d👦🏽': 1,
'👧🏿\u200d🖤\u200d👦🏽': 1,
'👧🏿\u200d❤️\u200d👨🏼': 1,
'👧🏿\u200d🖤\u200d👨🏼': 1,
'👩\u200d❤️\u200d👩': 1,
'👩\u200d🖤\u200d👩': 1,
'👧🏿\u200d❤️\u200d👧🏿': 1,
```

```
'👩🏿\u200d❤\u200d👩🏿': 1,
'👩🏿\u200d❤️\u200d👩🏻': 1,
'👩🏿\u200d❤\u200d👩🏻': 1,
'👩🏿\u200d❤️\u200d👩🏾': 1,
'👩🏿\u200d❤\u200d👩🏾': 1,
'👩🏿\u200d❤️\u200d👶🏻': 1,
'👩🏿\u200d❤\u200d👶🏻': 1,
'👩🏿\u200d❤️\u200d👩🏽': 1,
'👩🏿\u200d❤\u200d👩🏽': 1,
'👩🏻\u200d❤️\u200d👨🏻': 1,
'👩🏻\u200d❤\u200d👩🏻': 1,
'👩🏻\u200d❤️\u200d👩🏿': 1,
'👩🏻\u200d❤\u200d👨🏿': 1,
'👩🏻\u200d❤️\u200d👩🏾': 1,
'👩🏻\u200d❤\u200d👩🏾': 1,
'👩🏻\u200d❤️\u200d👶🏻': 1,
'👩🏻\u200d❤\u200d👶🏻': 1,
'👩🏻\u200d❤️\u200d👩🏽': 1,
'👩🏻\u200d❤\u200d👩🏽': 1,
'👩🏾\u200d❤️\u200d👩🏾': 1,
'👩🏾\u200d❤\u200d👩🏾': 1,
'👩🏾\u200d❤️\u200d👩🏿': 1,
'👩🏾\u200d❤\u200d👩🏿': 1,
'👩🏾\u200d❤️\u200d👨🏻': 1,
'👩🏾\u200d❤\u200d👩🏻': 1,
'👩🏾\u200d❤️\u200d👶🏻': 1,
'👩🏾\u200d❤\u200d👶🏻': 1,
'👩🏾\u200d❤️\u200d👩🏽': 1,
'👩🏾\u200d❤\u200d👩🏽': 1,
'👩🏽\u200d❤️\u200d👩🏽': 1,
'👩🏽\u200d❤\u200d👩🏽': 1,
'👩🏽\u200d❤️\u200d👩🏿': 1,
'👩🏽\u200d❤\u200d👩🏿': 1,
'👩🏽\u200d❤️\u200d👨🏻': 1,
'👩🏽\u200d❤\u200d👩🏻': 1,
'👩🏽\u200d❤️\u200d👩🏾': 1,
'👩🏽\u200d❤\u200d👩🏾': 1,
'👩🏽\u200d❤️\u200d👶🏻': 1,
'👩🏽\u200d❤\u200d👶🏻': 1,
'🎩': 1,
'✡️': 1,
'✴️': 1,
'✴': 1,
'🔥': 1,
'🚘': 1,
'🧯': 1,
```

```
'🧨': 1,
'👱\u200d🚒': 1,
'👦\u200d🚒': 1,
'👨\u200d🚒': 1,
'👨\u200d🚒': 1,
'👩\u200d🚒': 1,
'🧑\u200d🚒': 1,
'🎆': 1,
'🌟': 1,
'💚': 1,
'\U0001fa76': 1,
'😻': 1,
'😀': 1,
'😅': 1,
'😆': 1,
'💗': 1,
'💟': 1,
'❣': 1,
'❣': 1,
'🫶': 1,
'🫶🏿': 1,
'🫶🏻': 1,
'🫶🏾': 1,
'🫶🏼': 1,
'🫶🏽': 1,
'❤\u200d🔥': 1,
'🖤\u200d🔥': 1,
'❤': 1,
'♥': 1,
'💘': 1,
'💝': 1,
'\U0001fa75': 1,
'👱\u200d🚒': 1,
'👦\u200d🚒': 1,
'👨\u200d🚒': 1,
'👨\u200d🚒': 1,
'👩\u200d🚒': 1,
'🧑\u200d🚒': 1,
'❤\u200d🩹': 1,
'🖤\u200d🩹': 1,
'🎇': 1,
'🧡': 1,
'🎉': 1,
'🥳': 1,
'🪩': 1,
'\U0001fa77': 1,
'💜': 1,
'🌈': 1,
'🏳\u200d🌈': 1,
'🏳\u200d🌈': 1,
'❤': 1,
'🖤': 1,
'💞': 1,
'🤣': 1,
'🌄': 1,
'😘': 1,
```

```
'😎': 1,
'❇️': 1,
'✳': 1,
'🎇': 1,
'✨': 1,
'💖': 1,
'⭐': 1,
'🤩': 1,
'☪️': 1,
'☪': 1,
'✡️': 1,
'✡': 1,
'🌞': 1,
'☀': 1,
'🌤️': 1,
'🌥️': 1,
'🌦': 1,
'🌧️': 1,
'🌦️': 1,
'🌤': 1,
'⛅': 1,
'🤗': 1,
'🌻': 1,
'🕶️': 1,
'🕶': 1,
'🌅': 1,
'🌄': 1,
'🌇': 1,
'💕': 1,
'🤍': 1,
'👶\u200d🚒': 1,
'👧🏿\u200d🚒': 1,
'👩🏻\u200d🚒': 1,
'👧🏾\u200d🚒': 1,
'👶🏻\u200d🚒': 1,
'👩🏾\u200d🚒': 1,
'🧡': 1,
'😠': -1,
'😈': -1,
'😁': -1,
'🥶': -1,
'😾': -1,
'😥': -1,
'🌑': -1,
'👀': -1,
'😱': -1,
'\U0001fae9': -1,
'😵': -1,
'🙄': -1,
'😒': -1,
'😵\u200d💫': -1,
'😨': -1,
'😩': -1,
'😟': -1,
'😳': -1,
'😚': -1,
```

```
 '😙': -1,
 '👩‍❤️‍👩': -1,
 '👱\u200d♂️': -1,
 '👱\u200d♂': -1,
 '👨🏿\u200d♂️': -1,
 '👨🏿\u200d♂': -1,
 '👨🏻\u200d♂️': -1,
 '👨🏻\u200d♂': -1,
 '👨🏿\u200d♂️': -1,
 '👨🏿\u200d♂': -1,
 '👨🏼\u200d♂️': -1,
 '👨🏼\u200d♂': -1,
 '👨🏾\u200d♂️': -1,
 '👨🏾\u200d♂': -1,
 '👱': -1,
 '👨🏿': -1,
 '👨🏻': -1,
 '👨🏾': -1,
 '👨🏼': -1,
 '👨🏽': -1,
 '👨‍🦰': -1,
 '👩🏿‍🦰': -1,
 '👩🏻‍🦰': -1,
 '👩🏾‍🦰': -1,
 '👩🏽‍🦰': -1,
 '🥵': -1,
 '💀': -1,
 '☠️': -1,
'☠': -1,
 '😟': -1,
 '😊': -1,
 '😫': -1,
 '👱\u200d♀️': -1,
 '👱\u200d♀': -1,
 '👨🏿\u200d♀️': -1,
 '👨🏿\u200d♀': -1,
 '👨🏻\u200d♀️': -1,
 '👨🏻\u200d♀': -1,
 '👨🏿\u200d♀️': -1,
 '👨🏿\u200d♀': -1,
 '👨🏼\u200d♀️': -1,
 '👨🏼\u200d♀': -1,
 '👨🏾\u200d♀️': -1,
 '👨🏾\u200d♀': -1}
```

In [78]:
```
df_twitter["sentiment_score"] = df_twitter["description"].apply(emoji_sentiment_sco
```

In [79]:
```
# Calculate average sentiment score
avgt_cher = df_twitter.loc[df_twitter["artist"] == "cher", "sentiment_score"].mean(
avgt_robyn = df_twitter.loc[df_twitter["artist"] == "robyn", "sentiment_score"].mea

print(f"Artist: Cher \nAverage sentiment score: {avgt_cher:0.5f}")
print("\n")
print(f"Artist: Robyn \nAverage sentiment score: {avgt_robyn:0.5f}")
```

```
Artist: Cher
Average sentiment score: 0.00694


Artist: Robyn
Average sentiment score: 0.00570
```

In [80]:
```
pos_emoji_counts = total_emoji_counts(df_twitter, positive_emojis)
neg_emoji_counts = total_emoji_counts(df_twitter, negative_emojis)
```

In [81]:
```
for artist, counts in pos_emoji_counts.items():
    print(f"\n{artist}:")
    for e, c in counts.most_common(3):
        print(f"  {e}: {c}")
```

```
robyn:
  🏳‍🌈: 1701
  ♥: 1167
  ❤️: 987

cher:
  ❤️: 14715
  🏳‍🌈: 14122
  ♥: 10154
```

In [82]:
```
for artist, counts in neg_emoji_counts.items():
    print(f"\n{artist}:")
    for e, c in counts.most_common(3):
        print(f"  {e}: {c}")
```

```
robyn:
  😊: 75
  👀: 57
  😍: 55

cher:
  😊: 1766
  😍: 1541
  😁: 941
```

Q: What is the average sentiment of your two artists?

A: Cher has an average sentiment score of 0.00694 while Robyn has an average sentiment score of 0.00570.

Q: Which positive emoji is the most popular for each artist? Which negative emoji?

A: The most popular positive emoji for Robyn is 🏳‍🌈 (rainbow flag) while Cher's is the ❤️ (heart).

For both Cher and Robyn, the top most negative emoji is 😊 (smiling face with smiling eyes). I do not agree with this answer. The reason for this smiling face emoji is because my negative and positive emoji lists were not properly cleaned to remove conflicting emojis.