Sentiment Analysis

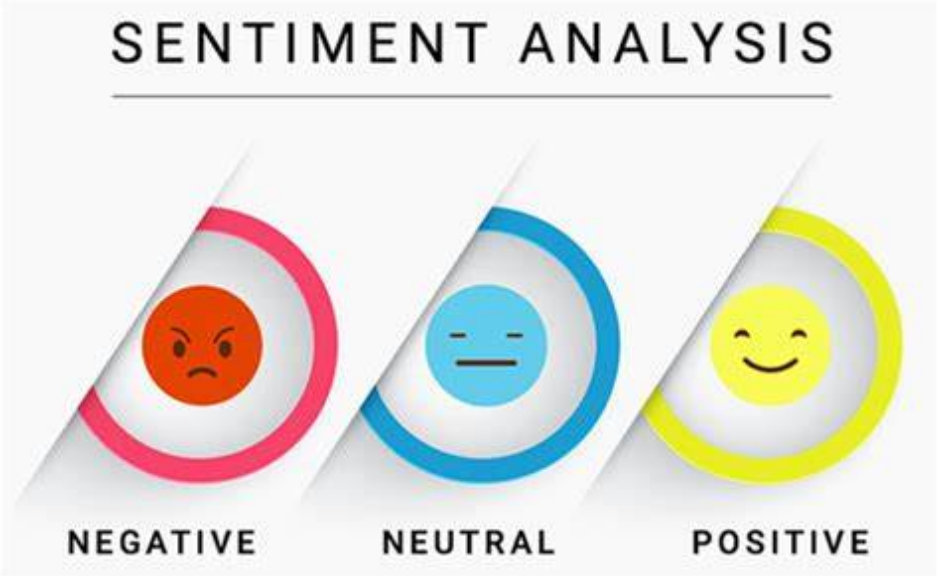


Table of Contents

Introduction

back to top

This project aims to develop a Python-based social media sentiment analysis that allows clients to analyze comments or users' interactions with social media accounts of musicians before and after they release their music. The project will collect data from various social media platforms, clean and preprocess the data, perform sentiment analysis, and display the collected data in a visually appealing and easy-to-read format. The project will provide insights on frequently used words, hashtags, users, and comment sentiment. The project will ensure that the analysis is performed on high-quality data, leading to more accurate results.

Problem Statement

back to top

This project aims to address the need for a comprehensive social media sentiment analysis tool for the client. By collecting and analyzing data from various social media platforms, the tool will provide insights on frequently used words, hashtags, users, and comment sentiment before and after a music release.

Objectives

back to top

- Collect data from various social media platforms within a specific time frame before and after the music release date.
- Preprocess the collected data by removing unnecessary characters, handling missing data, removing stop words, and performing other transformations to prepare the data for sentiment analysis.
- Use NLTK's Vader to perform sentiment analysis on each comment and classify them as positive, negative or neutral.
- Provide insights on frequently used words, hashtags, users, and comment sentiment to help the client analyze social media posts of musicians.
- Develop a visually appealing and easy-to-read notebook that displays the collected data in the form of graphs, charts, and other visualizations to enable the client to quickly analyze the data and gain insights.

Importing Libraries

back to top

In [1]:
```
# Libraries for data loading, data manipulation and data visulisation
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
%matplotlib inline
import seaborn as sns
sns.set() # plot style
from wordcloud import WordCloud
pd.options.display.float_format = '{:.0f}'.format

# Libraries for Natural Language Processing
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import TreebankWordTokenizer
from nltk import SnowballStemmer, PorterStemmer
from nltk.stem import WordNetLemmatizer
from nltk.sentiment import SentimentIntensityAnalyzer
#from nltk.util import ngrams
import re
import contractions # for word contractions
import string
import emoji

import warnings
warnings.filterwarnings('ignore')
```

Functions

back to top

Here are the functions that have been defined for this project:

- drop_columns(df): This function drops unnecessary columns from a given DataFrame.
- drop_null_tweets(df): This function removes any rows with null values in the Tweet_Content column of a given DataFrame.
- add_artist(df, name): This function adds an Artist column to a given DataFrame and fills it with the specified name.
- combine_dataframes(dfs): This function combines a list of DataFrames into one DataFrame.
- clean_column(column): This function cleans text data by converting to lowercase, removing non-ASCII characters, removing URLs, removing punctuation, and removing numbers.
- text_preproc(x): This function standardizes text data by expanding contractions, removing extra spaces and newlines.
- emoji_converter(x): This function converts emojis to words, removes hashtags, user names, and stop words from text data.
- extract_hashtags(df, column): This function extracts hashtags from a given column of a DataFrame and returns a DataFrame containing the hashtags and their occurrences.
- extract_users(df, column): This function extracts user names from a given column of a DataFrame and returns a DataFrame containing the user names and their occurrences.
- plot_hashtags(df, name): This function plots the hashtags extracted from a DataFrame using extract_hashtags.

- get_sentiment_label(tweet): This function uses the VADER Sentiment Analyzer to determine the sentiment of a given tweet and returns a label of 'Positive', 'Negative', or 'Neutral'.

In [2]:
```
def drop_columns(df):
    """The drop_columns function takes a DataFrame as input and drops sev
    The function returns the updated DataFrame with the specified columns

    df = df.drop(['Category', 'Keyword', 'Web_Page_URL', 'Tweet_Timestamp
       'Retweet_Original_Tweet_Content', 'Retweet_Original_Tweet_Poster',
       'Retweet_Original_Tweet_Time', 'Retweet_Original_Tweet_PosterID',
    return df
```

In [3]:
```
def drop_null_tweets(df):
    """Drop rows with missing values in the 'Tweet_Content' column of the

    Args:
    df (pandas.DataFrame): Input DataFrame.

    Returns:
    pandas.DataFrame: DataFrame with missing values in 'Tweet_Content' co
    """
    df = df.dropna(subset=['Tweet_Content'], axis=0)
    return df
```

In [4]:
```
def add_artist(df, name):
    """This function takes in a pandas DataFrame and an artist name as in
    It then adds a new column named 'Artist' to the DataFrame with the pr
    Finally, the updated DataFrame is returned."""
    df['Artist'] = name
    return df
```

In [5]:
```
def combine_dataframes(dfs):
    """The combine_dataframes function takes a list of dataframes as inpu

    Args:

    dfs: a list of pandas dataframes to be combined
    Returns:

    combined_df: a single pandas dataframe with all the rows from the inp
    The function iterates through each dataframe in the input list,
    and uses the append() method to concatenate it to the previously comb
    It also sets ignore_index=True to ensure that the row indexes are res
    Finally, it returns the concatenated dataframe."""
    combined_df = pd.DataFrame()  # create an empty dataframe to start wi
    for df in dfs:
        combined_df = combined_df.append(df, ignore_index=True)  # append
    return combined_df
```

In [6]:
```
#function to remove noise from dataframe

def clean_column(column):
    """The function clean_column(column) removes noise from a given colum
    converting the text to lowercase,
    removing web URLs, removing punctuation,
    removing numbers and converting unicode characters into binary string
```

```
    The cleaned column is returned."""
    #convert to lowercase()
    column = column.str.lower()
    #convert unicode characters into binary string
    column = column.str.encode('ascii', 'ignore').str.decode("utf-8")
    #removes web URL from text
    column = column.str.replace(r'https*\S+', 'url ', regex=True)
    #removes "\" from string
    column = column.str.replace(r'\'\w+', ' ', regex=True)

    column = column.str.replace(r'([A-Za-z])\1{2,}', r'\1', regex=True)
    # removes punctuation from string
    column = column.str.replace('[%s]' % re.escape(string.punctuation), '
    # removes numbers from string
    column = column.str.replace(r'\w*\d+\w*', ' ', regex=True)
    return column
```

In [7]:
```
def text_preproc(x):
    """The function text_preproc takes an input and performs the followin
    It expands contractions using the contractions package.
    It replaces multiple spaces with a single space using regular express
    It replaces newline characters with a single space using regular expr
    """
    x = ' '.join([contractions.fix(word) for word in x.split()])
    x = re.sub(' +', ' ', x)
    x = re.sub('\n', ' ', x)
    return x
```

The ***text_preproc*** function changes all the contradictions (*e.g don't **to** do not **and** there's **to** there is*), and removes extra whitespaces

In [8]:
```
def emoji_converter(x):
    """The function emoji_converter takes in a string of text as input,
    converts emojis to words,
    removes hashtags and user names,
    and then removes stopwords.
    The function returns the preprocessed string.
    """
    # Convert emojis to words
    x = ' '.join([emoji.demojize(word) for word in x.split()])
    # Remove hashtags
    x = re.sub(r'#\w+', '', x)
    # Remove user names
    x = re.sub(r'@\w+', '', x)
    # Remove stopwords
    stop_words = set(stopwords.words('english'))
    x = ' '.join([word for word in x.split() if word.lower() not in stop_
    return x
```

In [9]:
```
def plot_sentiment_distribution(df, column):
    sns.countplot(data=df, x=column)
```

In [10]:
```
def extract_hashtags(df, column):
    """
    This function takes a DataFrame and a column name as input and extrac
    It returns a DataFrame with two columns - 'hashtag' and 'occurrences'
    The 'hashtag' column contains the hashtag text and
    the 'occurrences' column contains the number of times that hashtag ap
```

```
    """
    hashtags = df[column].apply(lambda x: pd.value_counts(re.findall('(#\
                    .sum(axis=0)\
                    .to_frame()\
                    .reset_index()\
                    .sort_values(by=0,ascending=False)
    hashtags.columns = ['hashtag','occurences']
    return hashtags
```

In [11]:
```
def extract_users(df, column):
    """
    The function extract_users takes a dataframe and a column name as inp
    and returns a dataframe with two columns - the username
    and the number of times it occurs in the specified column of the inpu
    It uses regular expressions to find all mentions of usernames (starti
    and then counts the number of occurrences for each unique username.
    The resulting dataframe is sorted in descending order of occurrence c
    """
    users= df['Tweet_Content'].apply(lambda x: pd.value_counts(re.findall
                    .sum(axis=0)\
                    .to_frame()\
                    .reset_index()\
                    .sort_values(by=0,ascending=False)
    users.columns = ['user','occurences']
    return users
```

In [12]:
```
def plot_hashtags(df, name):
    df.plot(fontsize=10,figsize  = (10,5), kind='bar',y='occurences',x='h
    plt.xticks(rotation = 75)
    plt.tight_layout()
    plt.grid(False)
    plt.suptitle(f'Hashtags for {name} Tweets', fontsize=14);
```

In [13]:
```
def plot_users(df, name):
    df[:50].plot(fontsize=10,figsize  = (10,5), kind='bar',y='occurences'
    plt.xticks(rotation = 75, fontsize=14)
    plt.tight_layout()
    plt.grid(False)
    plt.suptitle(f'Top Users mention in {name} Tweets', fontsize=14);
```

In [14]:
```
def plot_wordcloud(df, name):
    all_tweets = " ".join(word for word in df.Tweet_Content)
    fig, ax = plt.subplots(1, 1, figsize  = (10,10))
    # Create and generate a word cloud image:
    wordcloud_all = WordCloud(max_font_size=100, max_words=100, backgroun
    # Display the generated image:
    ax.imshow(wordcloud_all, interpolation='bilinear')
    ax.set_title(f'WordCloud for {name} Tweets', fontsize=14)
    ax.axis('off');
```

In [15]:
```
def get_sentiment_label(tweet):
    """
    This function takes in a tweet as input,
    performs sentiment analysis on it using the VADER
    (Valence Aware Dictionary and Sentiment Reasoner) sentiment analysis
    and returns a sentiment label for the tweet (Positive, Negative or Ne
    based on the compound score calculated by VADER.
    """
```

```
    sid = SentimentIntensityAnalyzer()
    scores = sid.polarity_scores(tweet)
    if scores['compound'] >= 0.05:
        return 'Positive'
    elif scores['compound'] <= -0.05:
        return 'Negative'
    else:
        return 'Neutral'
```

Data

back to top

Data was scraped from Twitter and YouTube using Octoparse software

In [16]:
```
davido_announcement = pd.read_csv('Data/Davido-Timeless-Announcement.csv'
davido_release = pd.read_csv('Data/Davido-Timeless-release-tweet-data.csv
dualipa_announcement = pd.read_csv('Data/dua lipa - future nostalgia - an
dualipa_release = pd.read_csv('Data/dua lipa - future nostalgia - release
halsey_announcement = pd.read_csv('Data/halsey - manic - after release.cs
halsey_release = pd.read_csv('Data/halsey - manic - announcement.csv')
killers_announcement = pd.read_csv('Data/the killers imploding the mirage
killers_release = pd.read_csv('Data/the killers imploding the mirage rele
weeknd_announcement = pd.read_csv('Data/the weeknd after hours announceme
weeknd_release = pd.read_csv('Data/the weeknd after hours release.csv')
```

The above code reads in several CSV files containing data related to music album announcements and releases. The data includes information such as the artist, the content of the tweet, the date and time of the tweet, and other relevant details. This data will be used for analysis and visualization in the project.

Exploratory Data Analysis

back to top

In [17]:
```
davido_announcement.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 183 entries, 0 to 182
Data columns (total 18 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   Category                        183 non-null    object 
 1   Keyword                         0 non-null      float64
 2   Web_Page_URL                    183 non-null    object 
 3   Tweet_Website                   182 non-null    object 
 4   Author_Name                     34 non-null     object 
 5   Author_Web_Page_URL             182 non-null    object 
 6   Tweet_Timestamp                 182 non-null    float64
 7   Tweet_Time                      183 non-null    object 
 8   Tweet_Content                   152 non-null    object 
 9   Tweet_Image_URL                 19 non-null     object 
 10  Tweet_Number_of_Likes           168 non-null    float64
 11  Tweet_Number_of_Retweets        36 non-null     float64
 12  Tweet_Number_of_Reviews         121 non-null    float64
 13  Retweet_or_not                  183 non-null    object 
 14  Retweet_Original_Tweet_Content  2 non-null      object 
 15  Retweet_Original_Tweet_Poster   2 non-null      object 
 16  Retweet_Original_Tweet_Time     2 non-null      object 
 17  Retweet_Original_Tweet_PosterID  2 non-null      object 
dtypes: float64(5), object(13)
memory usage: 25.9+ KB
```

In [18]:
```
davido_announcement.head()
```

Out[18]:

	Category	Keyword	Web_Page_URL	
0	Post	NaN	https://twitter.com/davido/status/164113897954...	https://twitter.com/d
1	Post	NaN	https://twitter.com/davido/status/164113897954...	https://twitter.com/I
2	Post	NaN	https://twitter.com/davido/status/164113897954...	https://twitter.com/RE
3	Post	NaN	https://twitter.com/davido/status/164113897954...	https://twitter.com/D
4	Post	NaN	https://twitter.com/davido/status/164113897954...	https://twitter.com/Da

In [19]:
```
davido_announcement.describe(include='O')
```

Out[19]:

	Category	Web_Page_URL	
count	183	183	
unique	1	1	
top	Post	https://twitter.com/davido/status/164113897954...	https://twitter.com/davido/s
freq	183	183	

Data Cleaning

back to top

In [20]:
```
davido_announcement = drop_columns(davido_announcement)
davido_release = drop_columns(davido_release)
dualipa_announcement = drop_columns(dualipa_announcement)
dualipa_release = drop_columns(dualipa_release)
halsey_announcement = drop_columns(halsey_announcement)
halsey_release = drop_columns(halsey_release)
killers_announcement = drop_columns(killers_announcement)
killers_release = drop_columns(killers_release)
weeknd_announcement = drop_columns(weeknd_announcement)
weeknd_release = drop_columns(weeknd_release)
```

The code above drops some unnecessary columns from each of the dataframes

In [21]:
```
davido_announcement = drop_null_tweets(davido_announcement)
davido_release = drop_null_tweets(davido_release)
dualipa_announcement = drop_null_tweets(dualipa_announcement)
dualipa_release = drop_null_tweets(dualipa_release)
halsey_announcement = drop_null_tweets(halsey_announcement)
halsey_release = drop_null_tweets(halsey_release)
killers_announcement = drop_null_tweets(killers_announcement)
killers_release = drop_null_tweets(killers_release)
weeknd_announcement = drop_null_tweets(weeknd_announcement)
weeknd_release = drop_null_tweets(weeknd_release)
```

The code above drops all the rows that have empty tweets from each of the dataframes

In [22]:
```
davido_announcement = add_artist(davido_announcement, 'Davido')
davido_release = add_artist(davido_release, 'Davido')
dualipa_announcement = add_artist(dualipa_announcement, 'Dua Lipa')
dualipa_release = add_artist(dualipa_release, 'Dua Lipa')
halsey_announcement = add_artist(halsey_announcement, 'Halsey')
halsey_release = add_artist(halsey_release, 'Halsey')
killers_announcement = add_artist(killers_announcement, 'The Killers')
killers_release = add_artist(killers_release, 'The Killers')
weeknd_announcement = add_artist(weeknd_announcement, 'The Weeknd')
weeknd_release = add_artist(weeknd_release, 'The Weeknd')
```

The code above adds artist name to all the rows from each of the dataframes related to the artist

Data Preparation

back to top

In [23]:
```
davido_df = combine_dataframes([davido_announcement, davido_release])
```

In [24]:
```
dualipa_df = combine_dataframes([dualipa_announcement, dualipa_release])
```

In [25]:
```
halsey_df = combine_dataframes([halsey_announcement, halsey_release])
```

In [26]:
```
killers_df = combine_dataframes([killers_announcement, killers_release])
```

In [27]:
```
weeknd_df = combine_dataframes([weeknd_announcement, weeknd_release])
```

In [28]:
```
announcement_df = combine_dataframes([davido_announcement, dualipa_announ
                                      halsey_announcement, killers_announ
                                      weeknd_announcement])
```

In [29]:
```
release_df = combine_dataframes([davido_release, dualipa_release,
                                 halsey_release, killers_release,
                                 weeknd_release])
```

In [30]:
```
all_df = combine_dataframes([davido_announcement, dualipa_announcement,
                             halsey_announcement, killers_announcement,
                             weeknd_announcement, davido_release,
                             dualipa_release, halsey_release,
                             killers_release, weeknd_release])
```

Data Exploration & Visualization

back to top

Pre-release

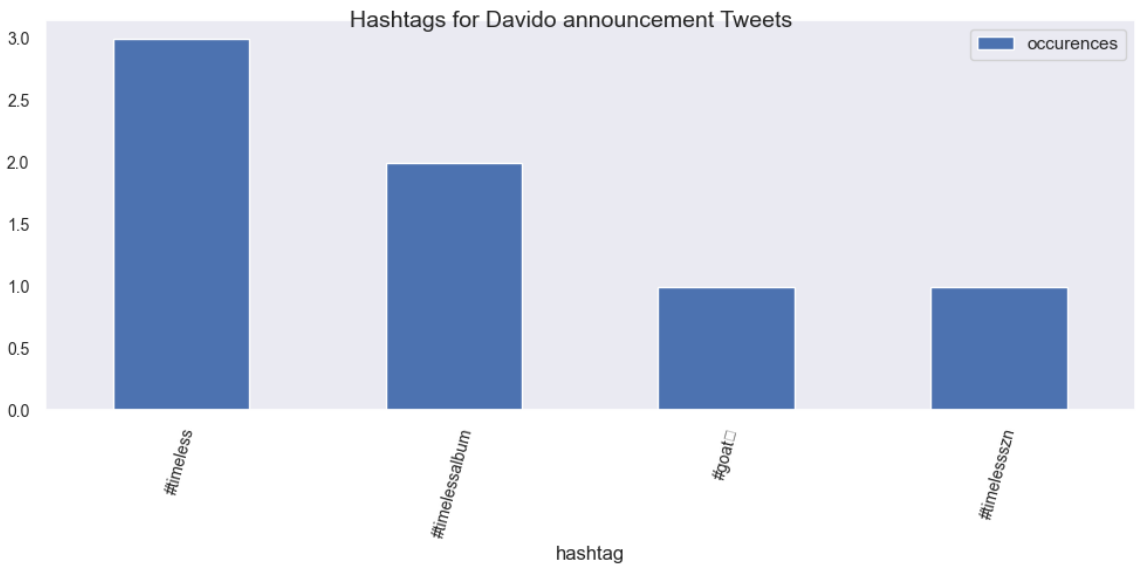
back to top

Davido

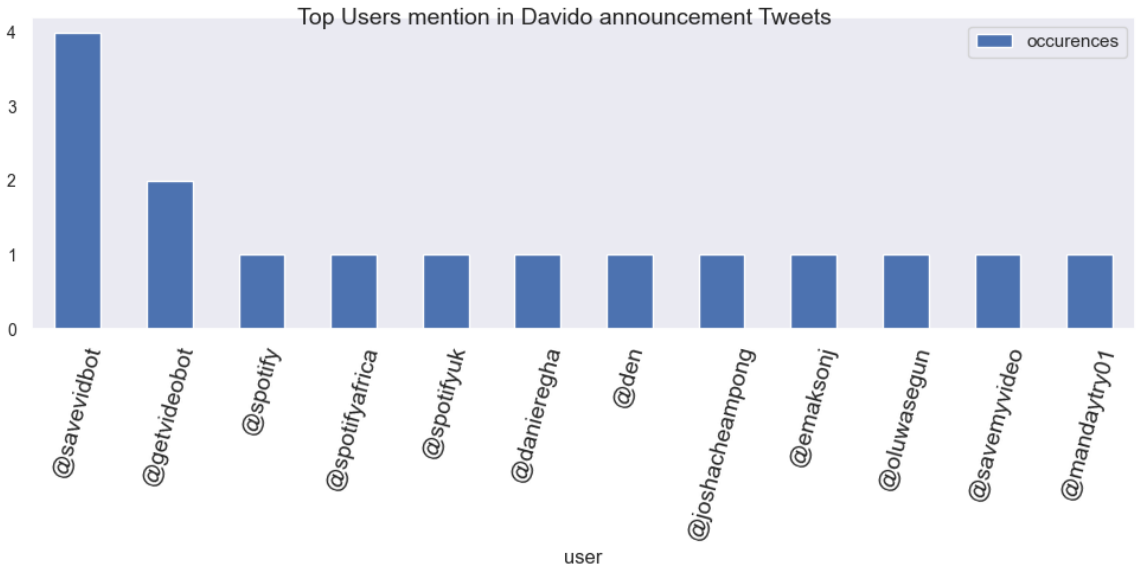
In [31]:
```
davido_announcement_hashtags = extract_hashtags(davido_announcement, 'Twe
davido_announcement_users = extract_users(davido_announcement, 'Tweet_Con
```

This code block extracted hashtags and users from the pre-release tweet of the artist

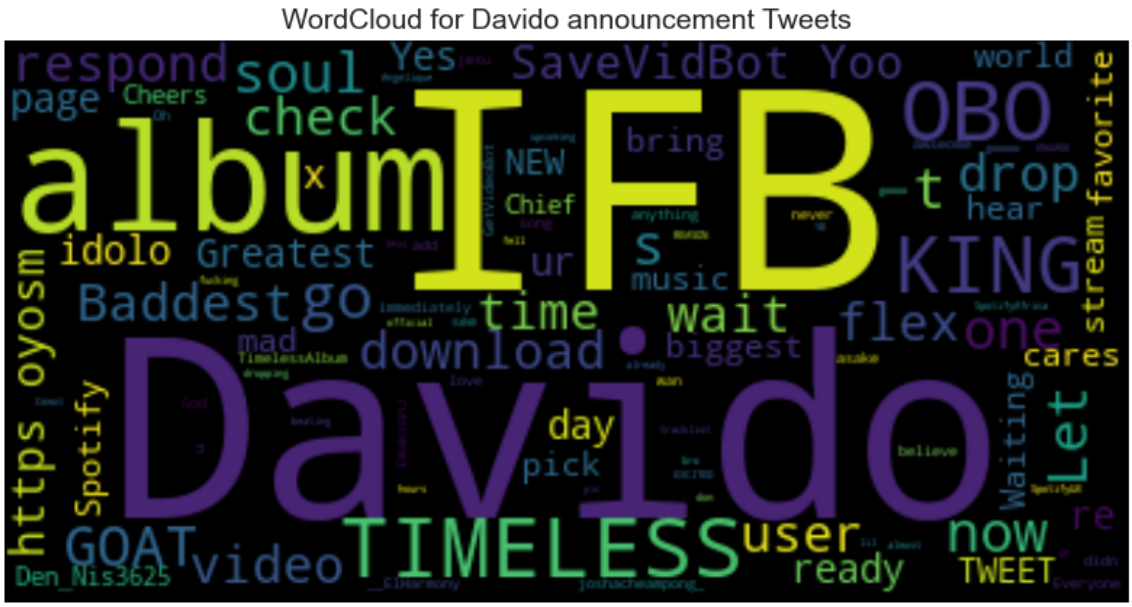
In [32]:
```
plot_hashtags(davido_announcement_hashtags, "Davido announcement")
```



In [33]:
```
plot_users(davido_announcement_users, "Davido announcement")
```



In [34]:
```
plot_wordcloud(davido_announcement, "Davido announcement")
```

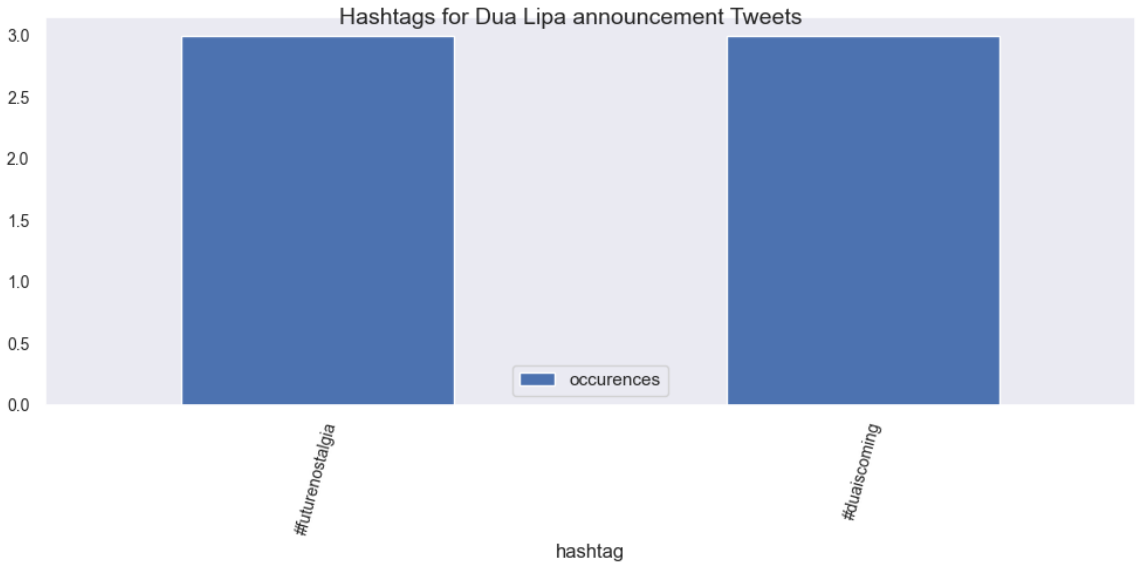


Dua Lipa

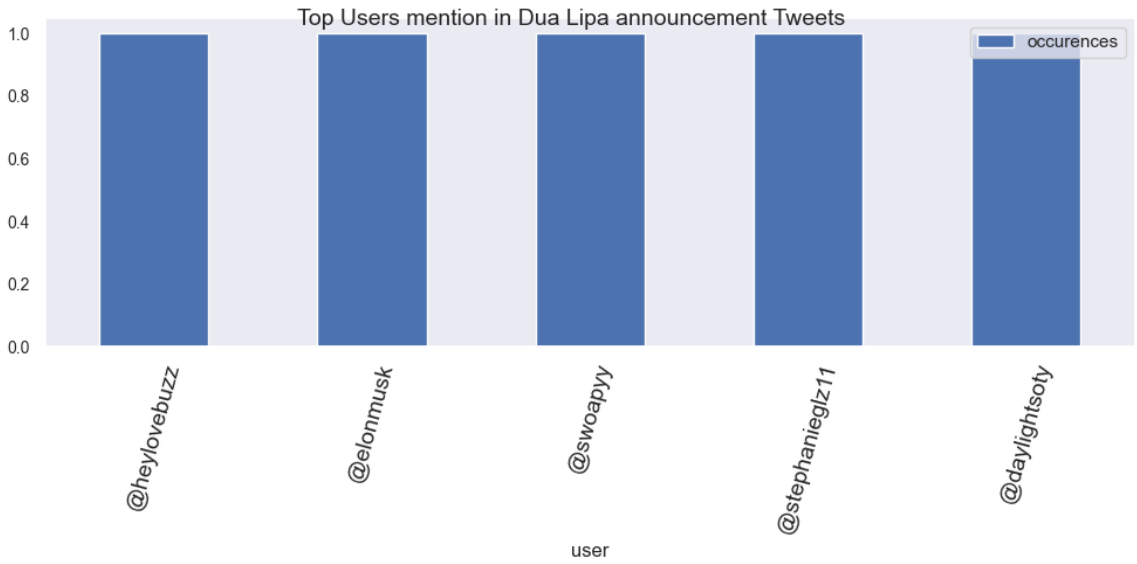
In [35]:
```
dualipa_announcement_hashtags = extract_hashtags(dualipa_announcement, 'T
dualipa_announcement_users = extract_users(dualipa_announcement, 'Tweet_C
```

This code block extracted hashtags and users from the pre-release tweet of the artist

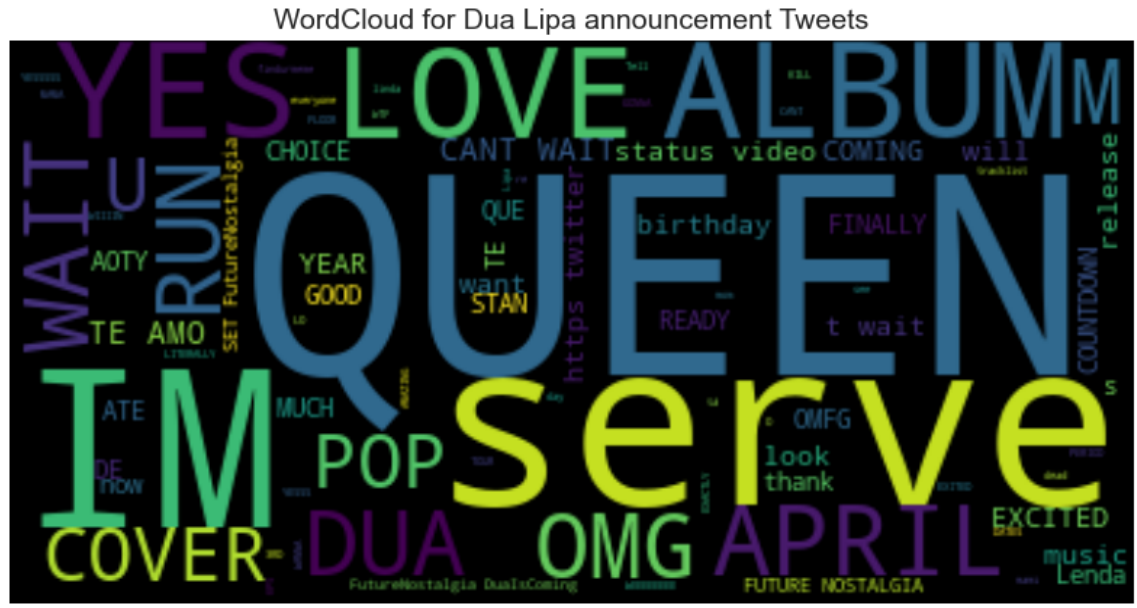
In [36]:
```
plot_hashtags(dualipa_announcement_hashtags, "Dua Lipa announcement")
```



In [37]:
```
plot_users(dualipa_announcement_users, "Dua Lipa announcement")
```



In [38]:
```
plot_wordcloud(dualipa_announcement, "Dua Lipa announcement")
```

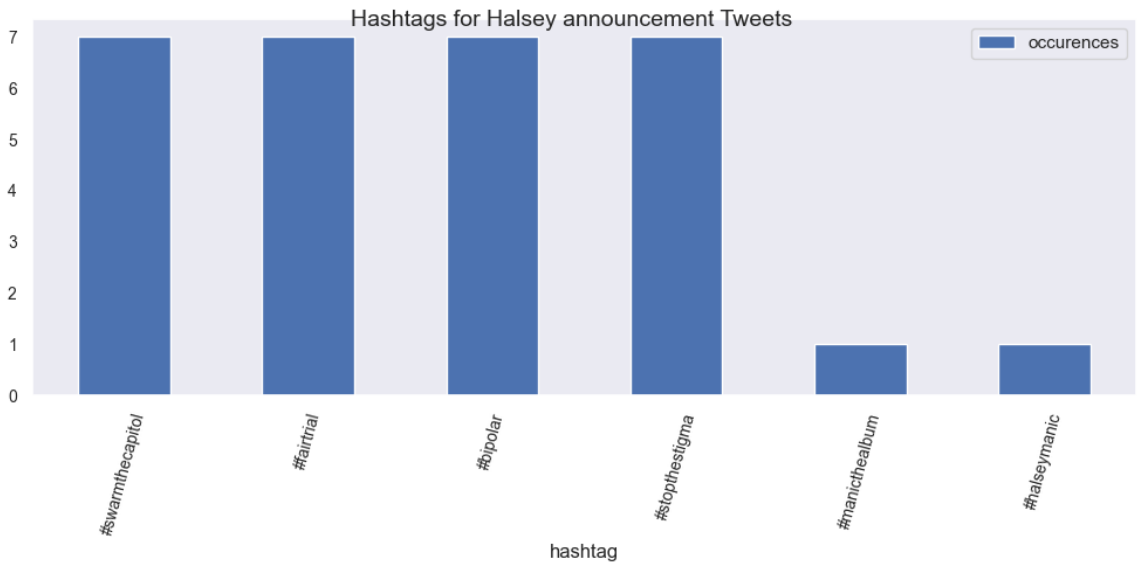


Halsey

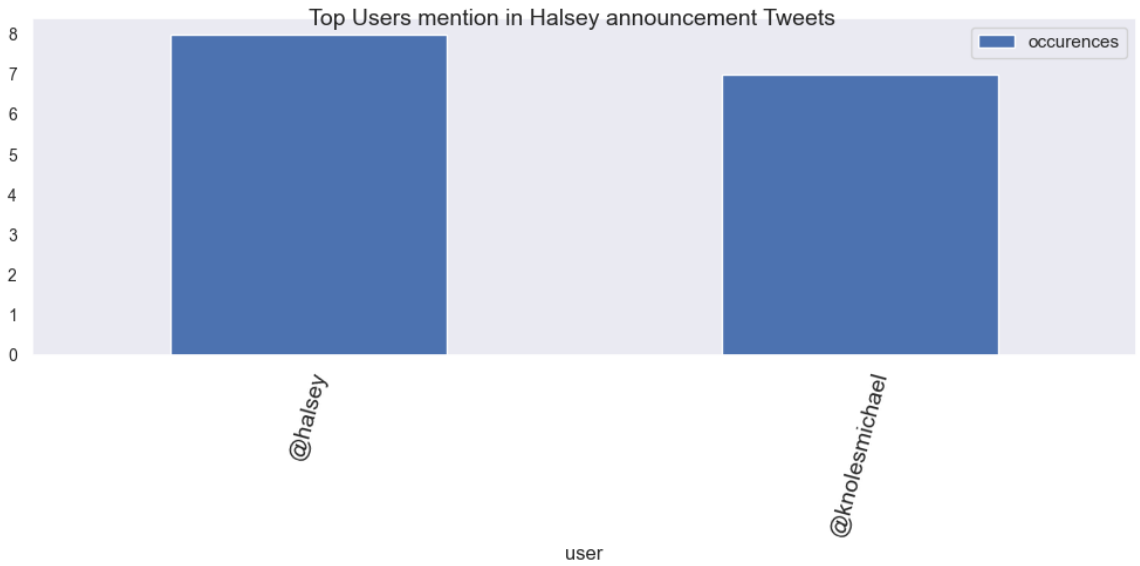
In [39]:
```
halsey_announcement_hashtags = extract_hashtags(halsey_announcement, 'Twe
halsey_announcement_users = extract_users(halsey_announcement, 'Tweet_Con
```

This code block extracted hashtags and users from the pre-release tweet of the artist

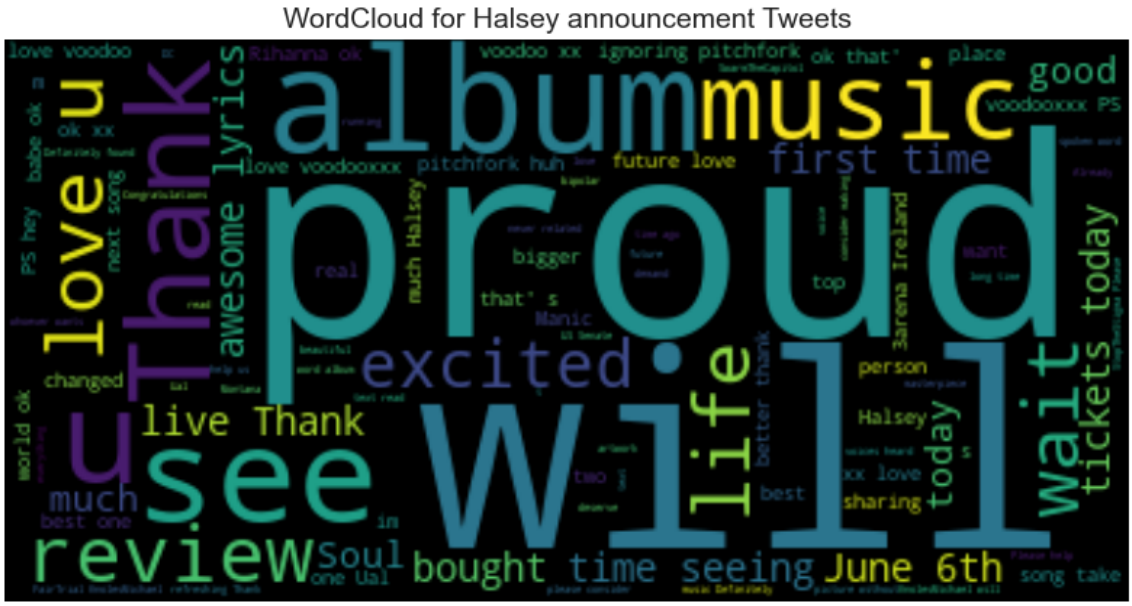
In [40]:
```
plot_hashtags(halsey_announcement_hashtags, 'Halsey announcement')
```



In [41]:
```
plot_users(halsey_announcement_users, 'Halsey announcement')
```



In [42]:
```
plot_wordcloud(halsey_announcement, 'Halsey announcement')
```

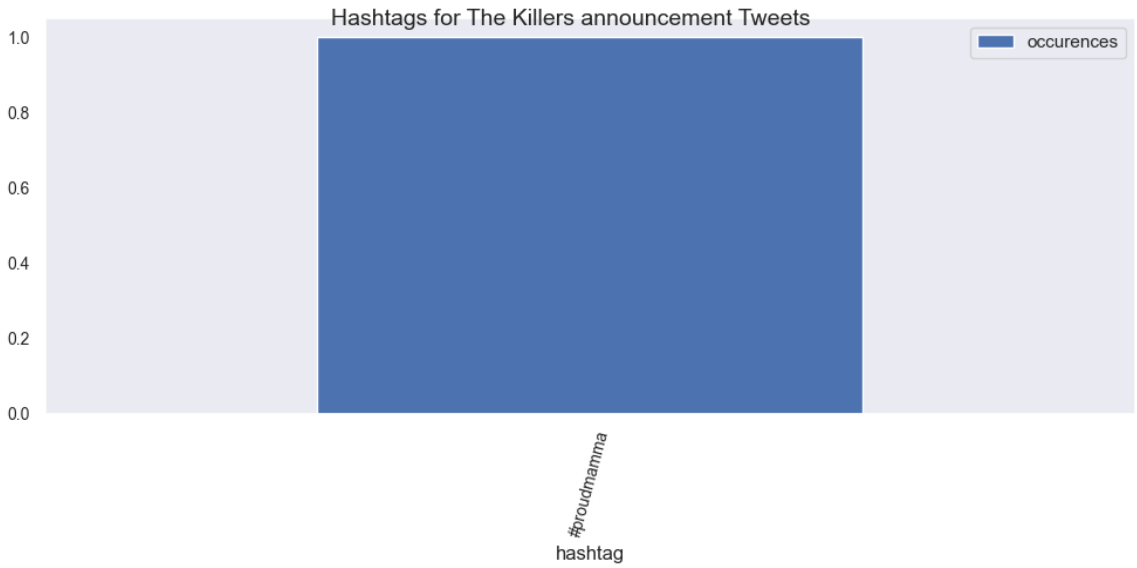


The Killers

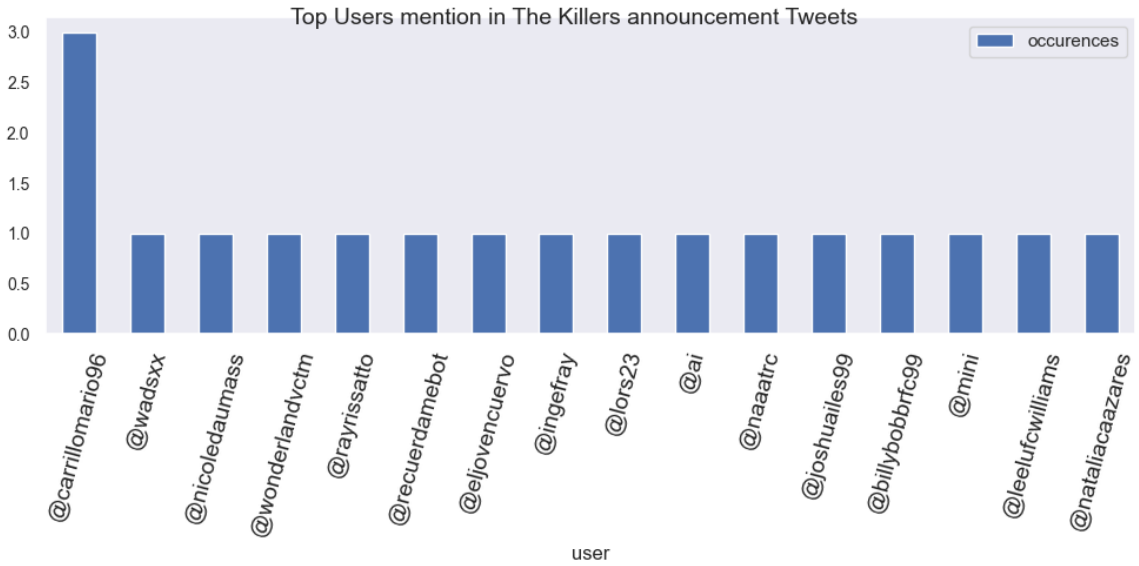
In [43]:
```
killers_announcement_hashtags = extract_hashtags(killers_announcement, 'T
killers_announcement_users = extract_users(killers_announcement, 'Tweet_C
```

This code block extracted hashtags and users from the pre-release tweet of the artist

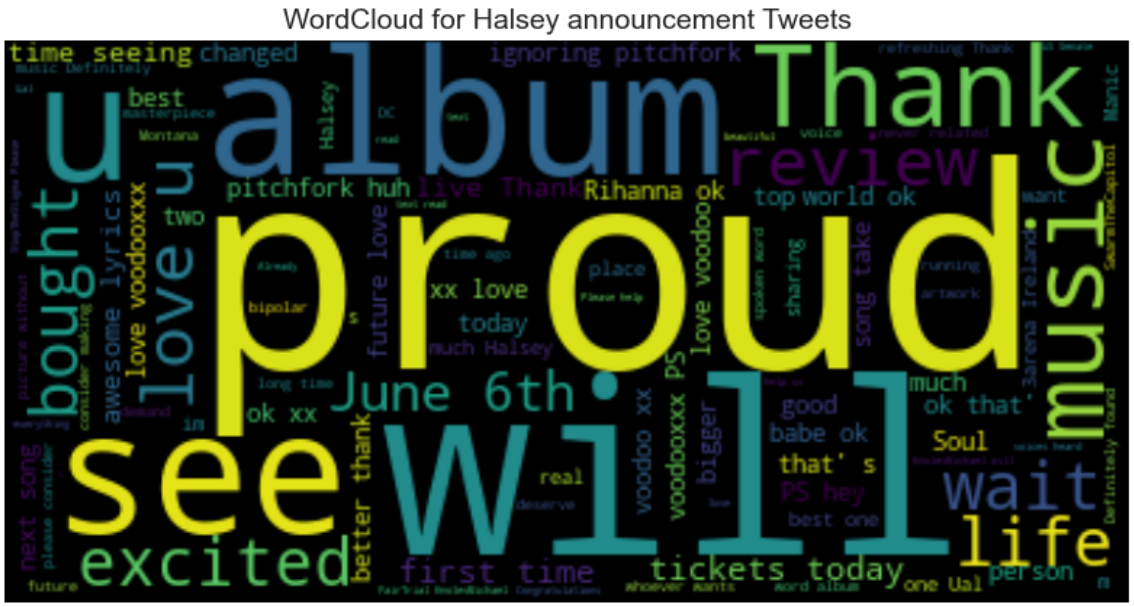
In [44]:
```
plot_hashtags(killers_announcement_hashtags, 'The Killers announcement')
```



In [45]:
```
plot_users(killers_announcement_users, 'The Killers announcement')
```



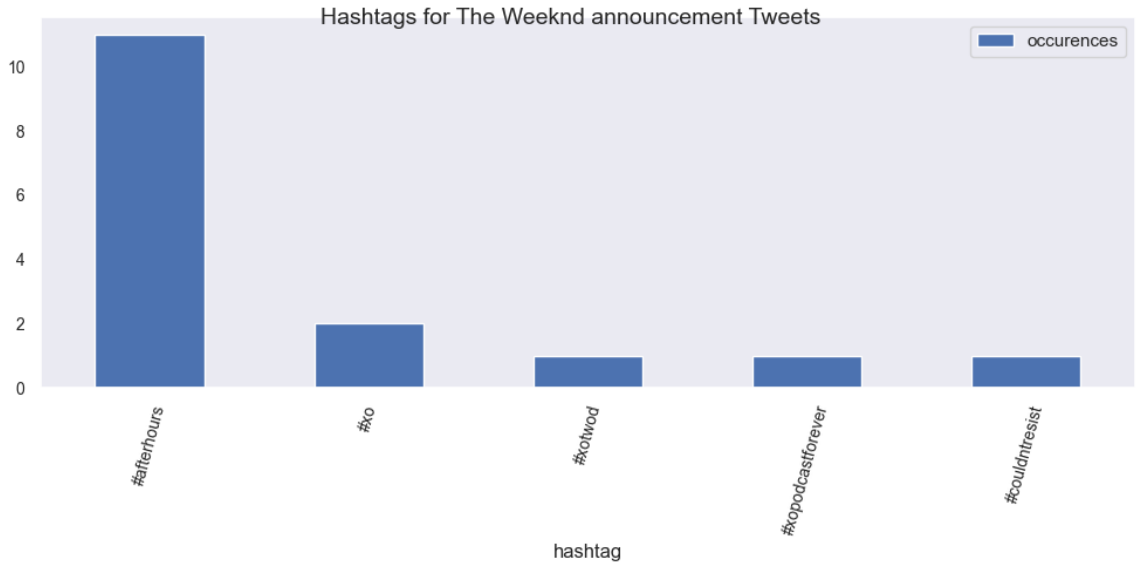
In [46]:
```
plot_wordcloud(halsey_announcement, 'Halsey announcement')
```



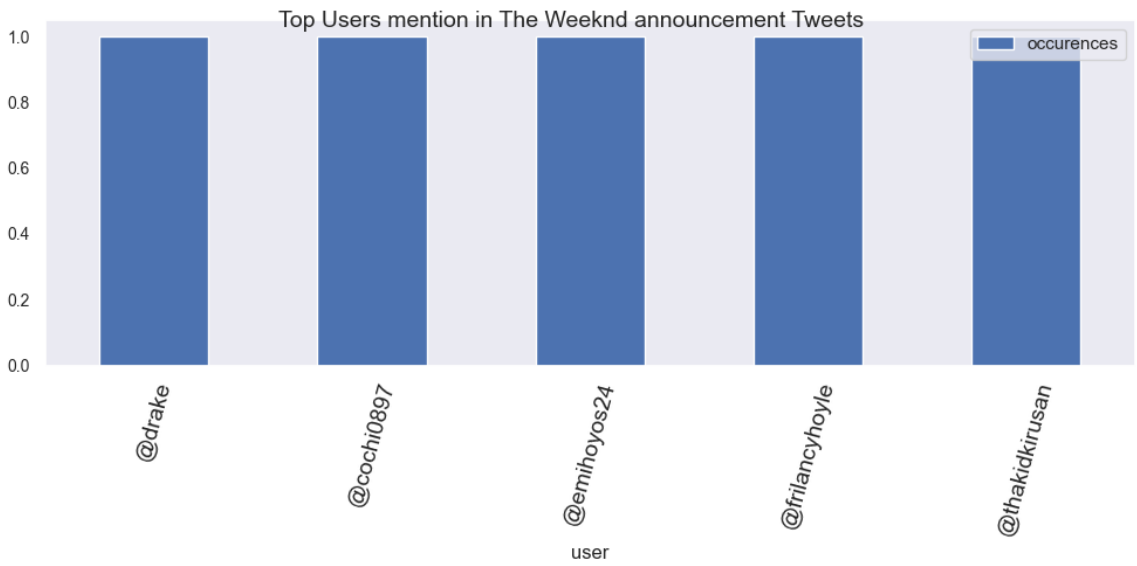
The Weeknd

In [47]:
```
weeknd_announcement_hashtags = extract_hashtags(weeknd_announcement, 'Twe
weeknd_announcement_users = extract_users(weeknd_announcement, 'Tweet_Con
```

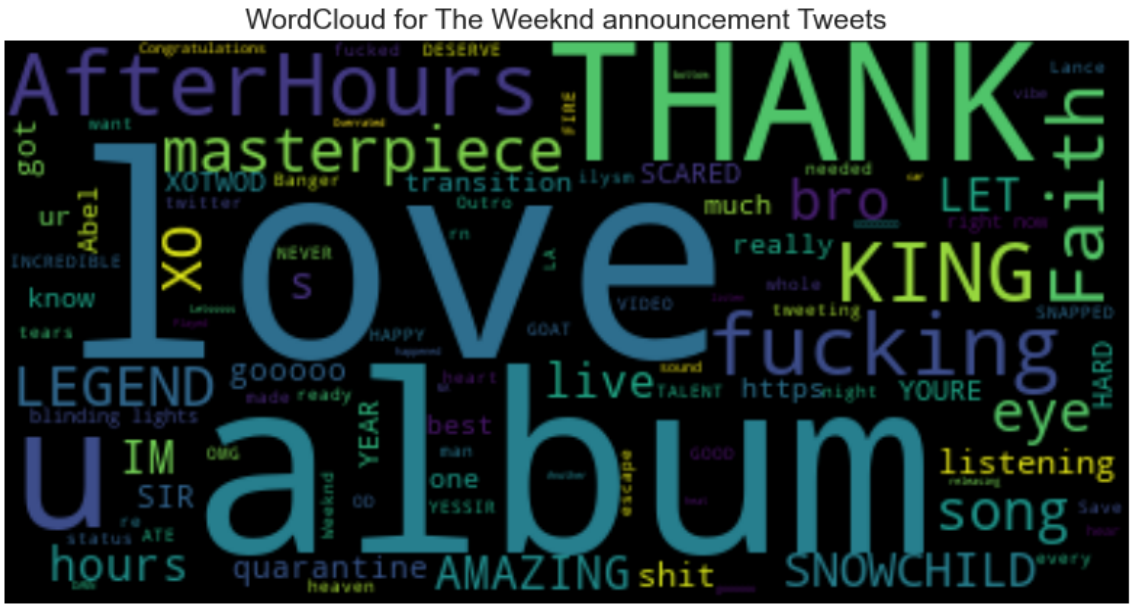
In [48]:
```
plot_hashtags(weeknd_announcement_hashtags, 'The Weeknd announcement')
```



In [49]:
```
plot_users(weeknd_announcement_users, 'The Weeknd announcement')
```



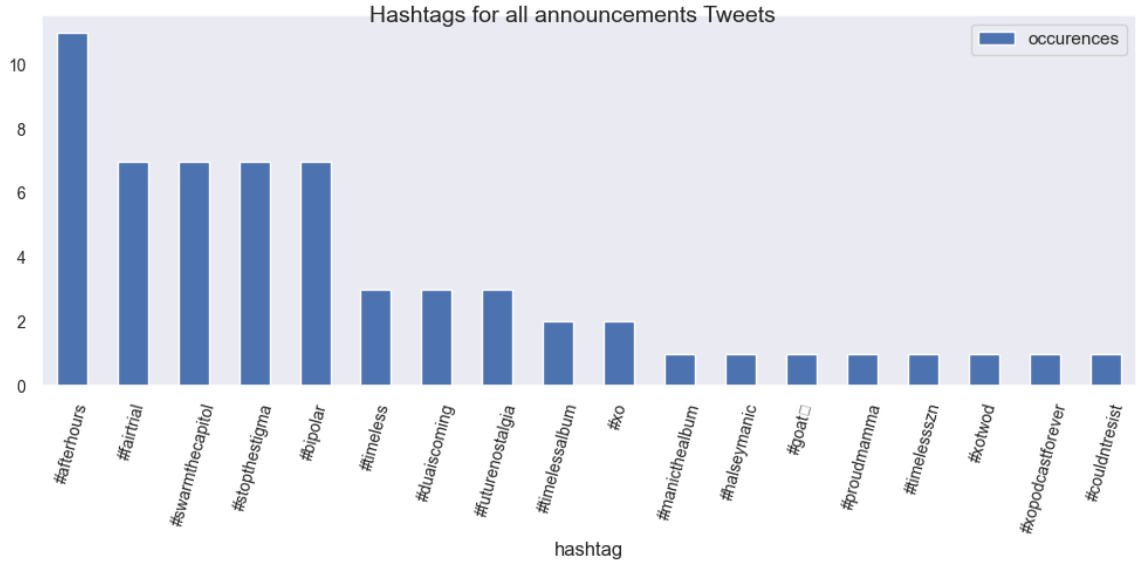
In [50]:
```
plot_wordcloud(weeknd_announcement, 'The Weeknd announcement')
```



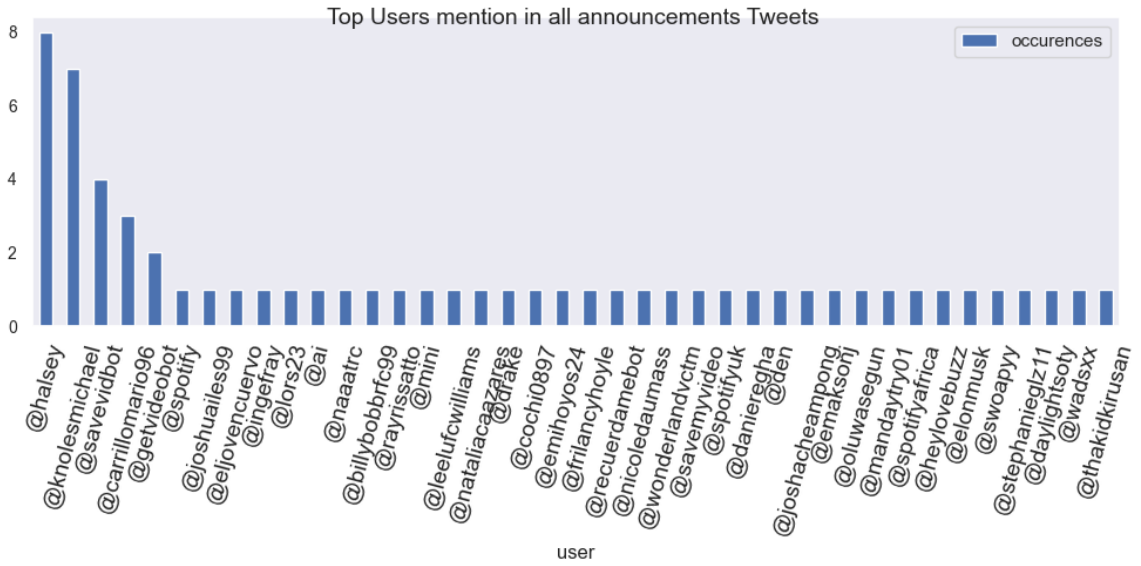
All Pre-release tweets

In [51]:
```
announcement_df_hashtags = extract_hashtags(announcement_df, 'Tweet_Conte
announcement_df_users = extract_users(announcement_df, 'Tweet_Content')
```

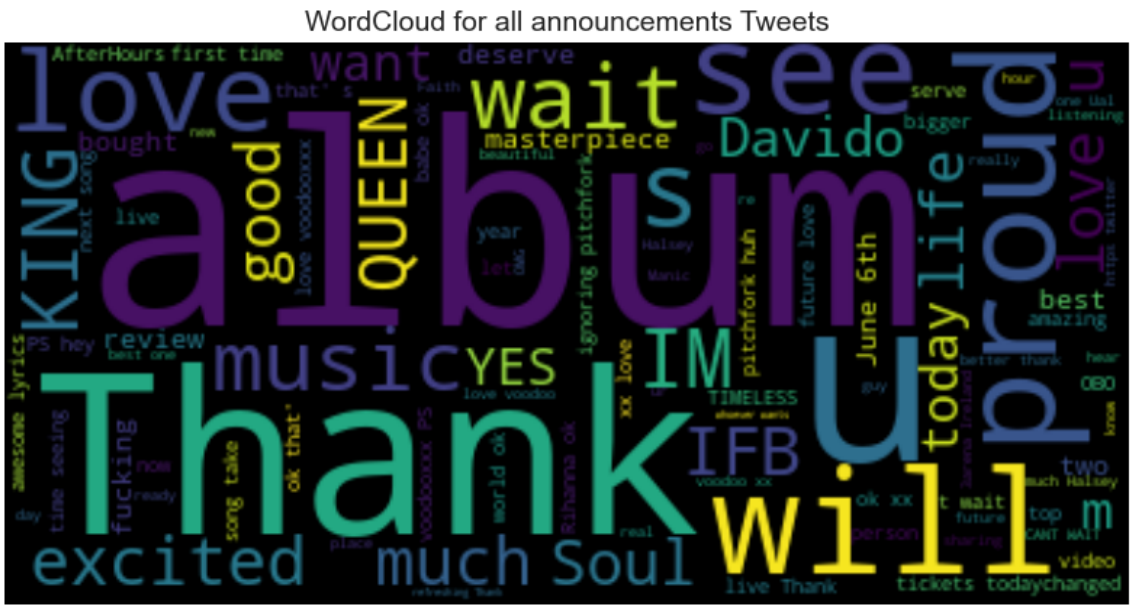
In [52]:
```
plot_hashtags(announcement_df_hashtags, 'all announcements')
```



In [53]:
```
plot_users(announcement_df_users, 'all announcements')
```



In [54]:
```
plot_wordcloud(announcement_df, 'all announcements')
```



Post-release

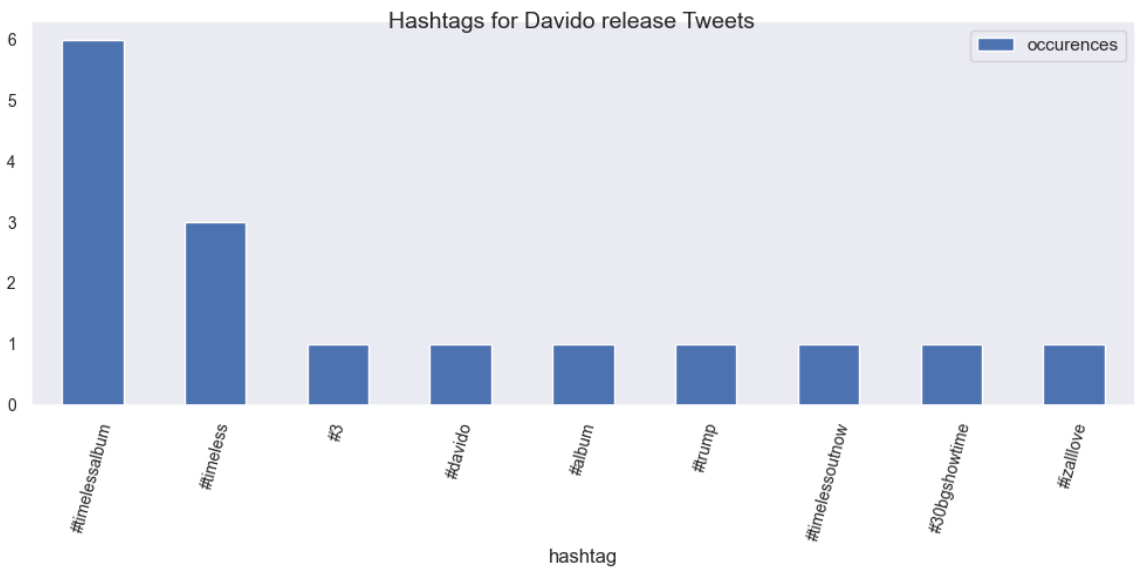
back to top

Davido

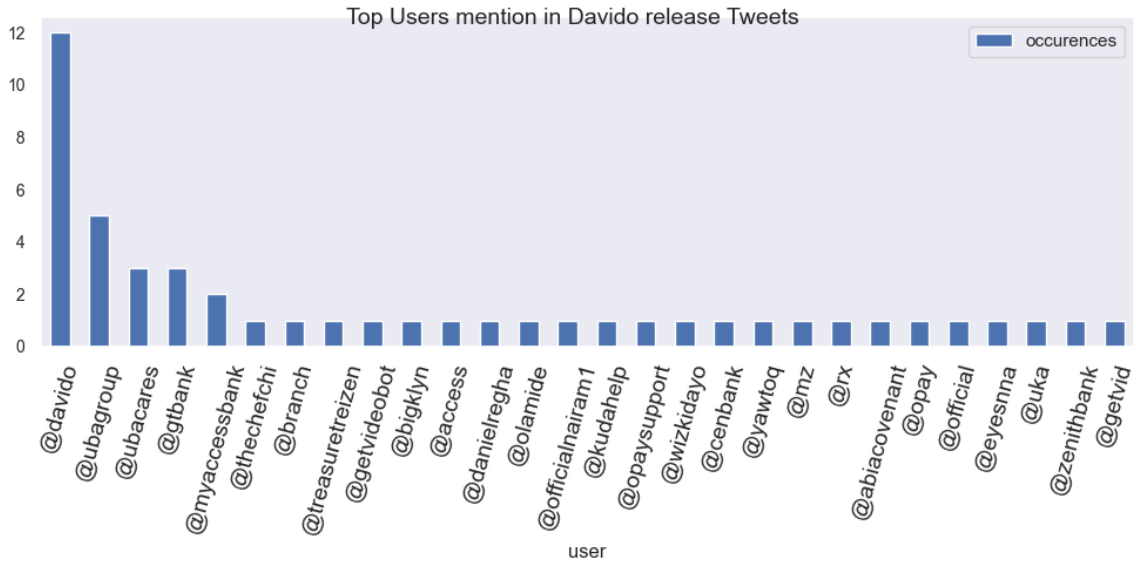
In [55]:
```
davido_release_hashtags = extract_hashtags(davido_release, 'Tweet_Content
```

In [56]:
```
davido_release_users = extract_users(davido_release, 'Tweet_Content')
```

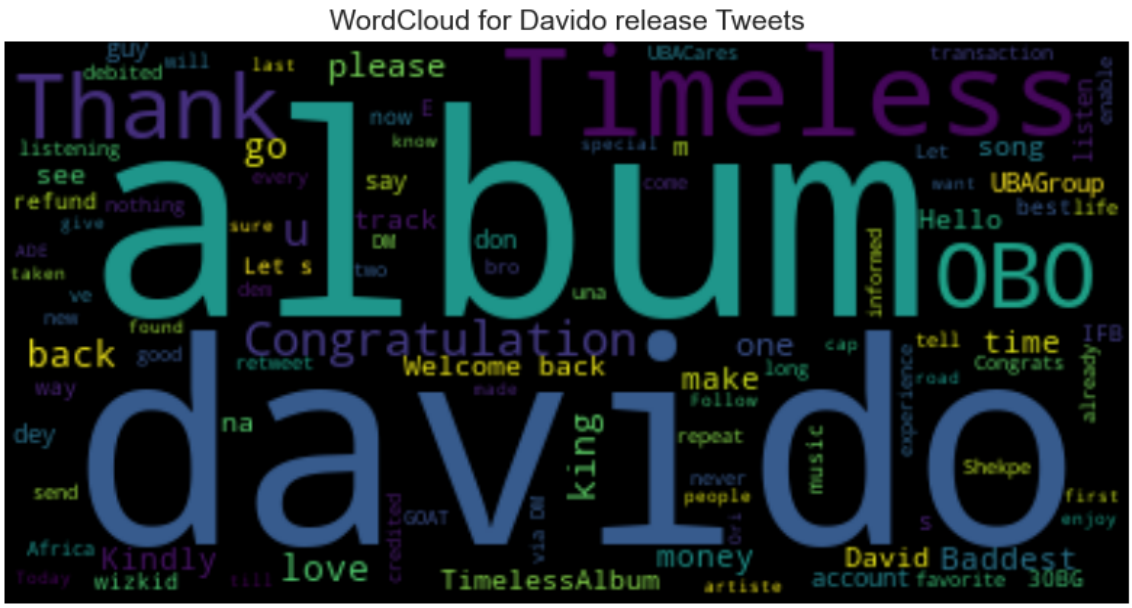
In [57]:
```
plot_hashtags(davido_release_hashtags, "Davido release")
```



In [58]:
```
plot_users(davido_release_users, "Davido release")
```



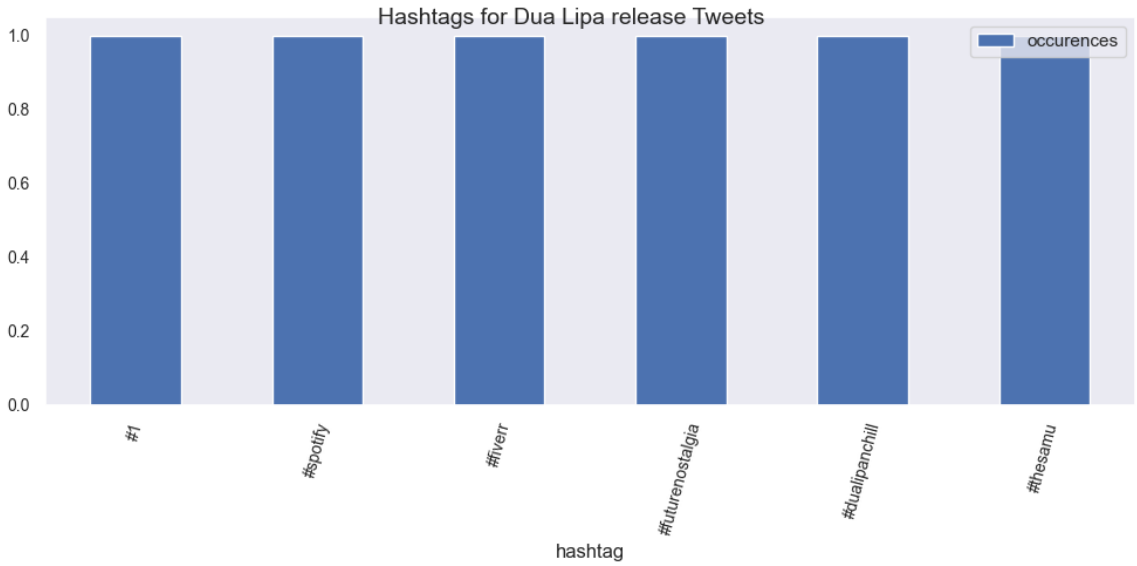
In [59]:
```
plot_wordcloud(davido_release, "Davido release")
```



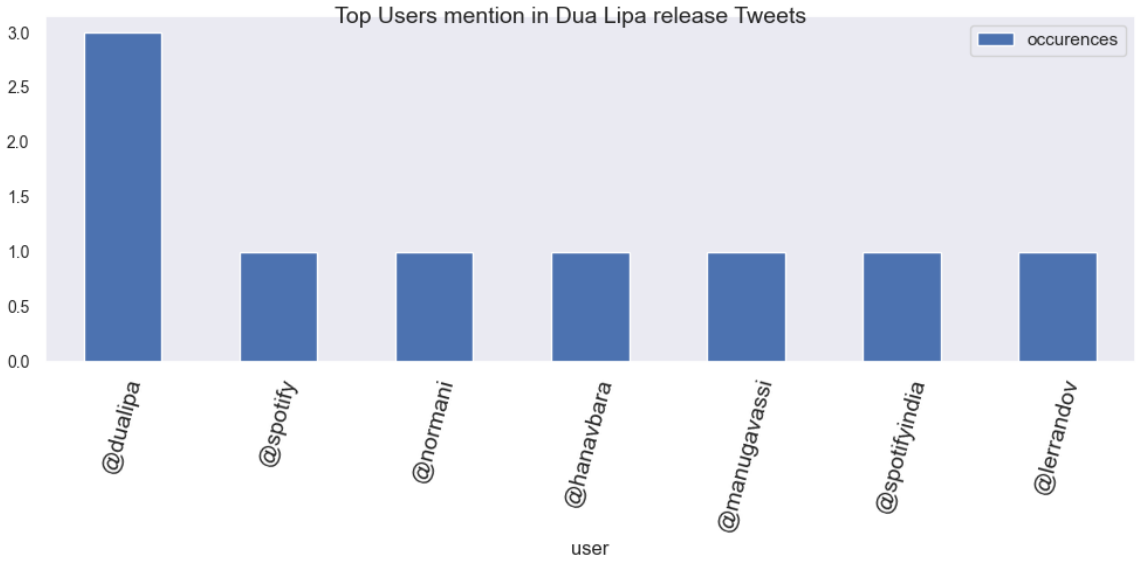
Dua Lipa

In [60]:
```
dualipa_release_hashtags = extract_hashtags(dualipa_release, 'Tweet_Conte
dualipa_release_users = extract_users(dualipa_release, 'Tweet_Content')
```

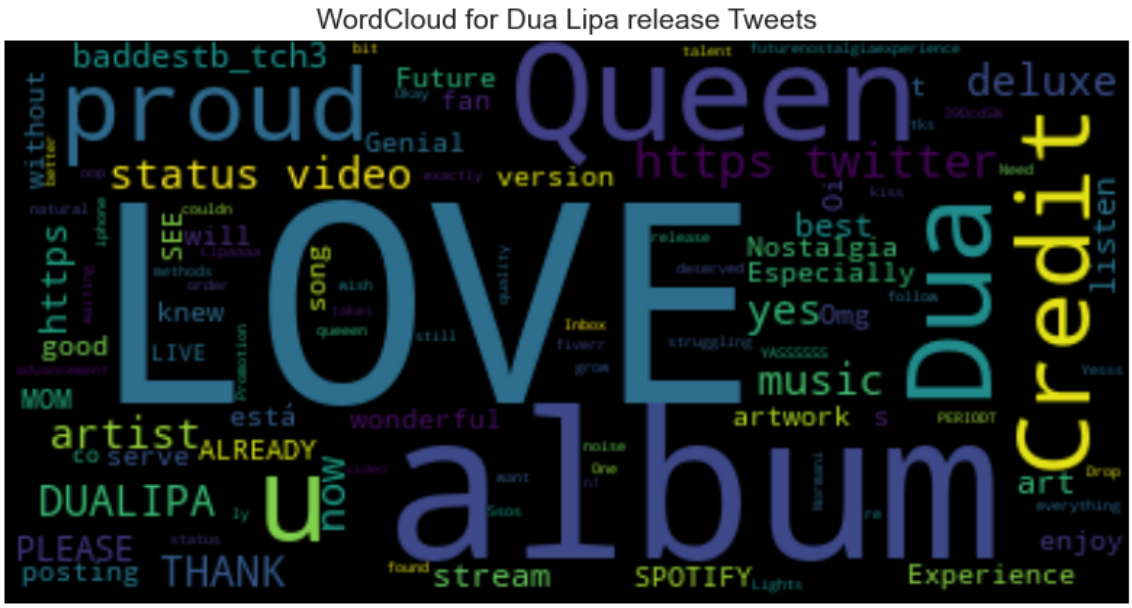
In [61]:
```
plot_hashtags(dualipa_release_hashtags, "Dua Lipa release")
```



In [62]:
```
plot_users(dualipa_release_users, "Dua Lipa release")
```



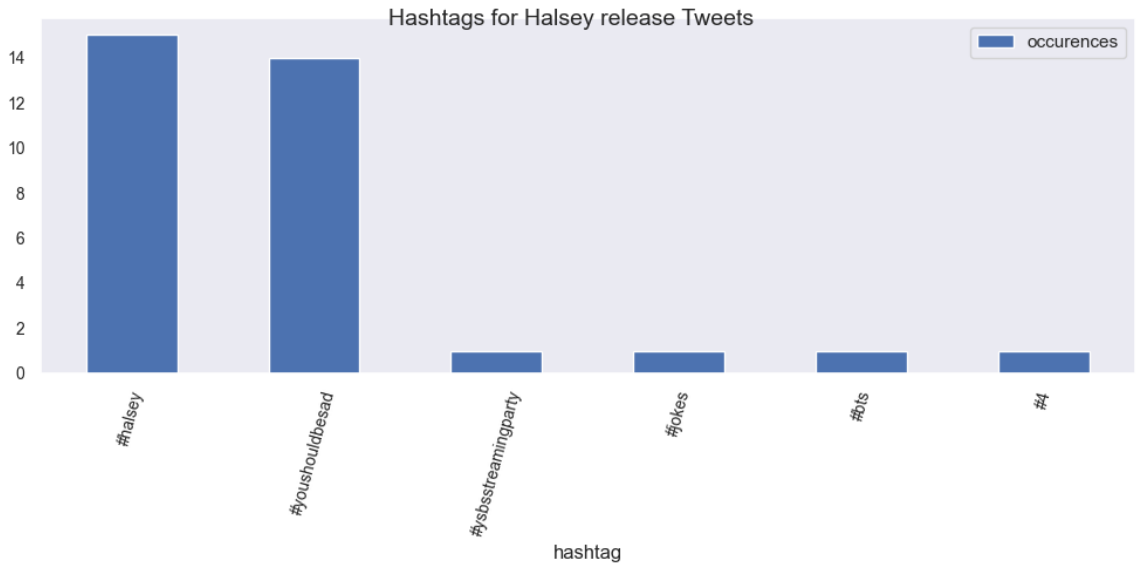
In [63]:
```
plot_wordcloud(dualipa_release, "Dua Lipa release")
```



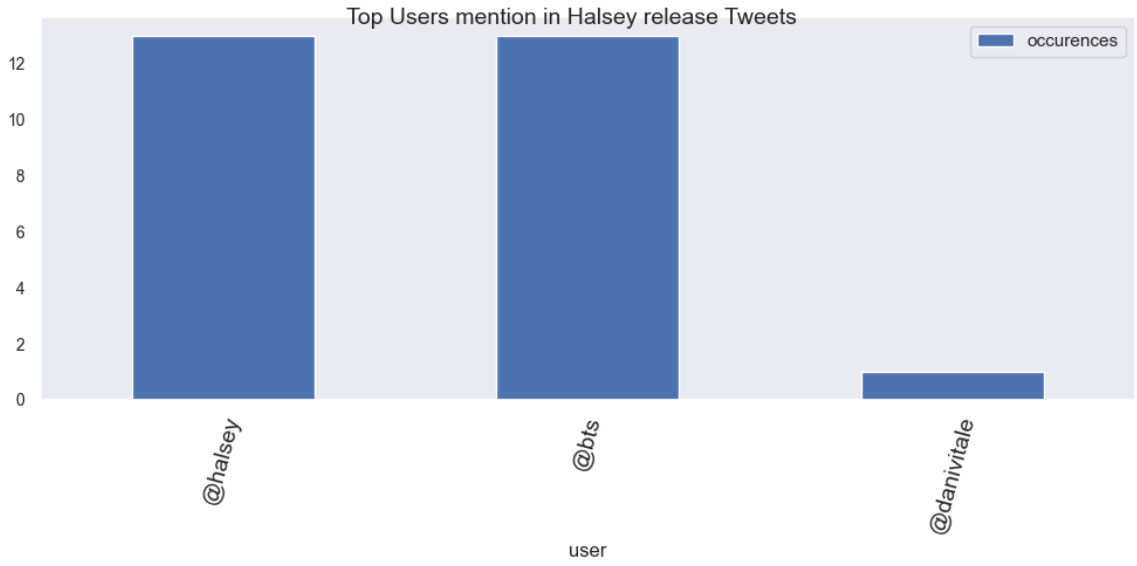
Halsey

In [64]:
```
halsey_release_hashtags = extract_hashtags(halsey_release, 'Tweet_Content
halsey_release_users = extract_users(halsey_release, 'Tweet_Content')
```

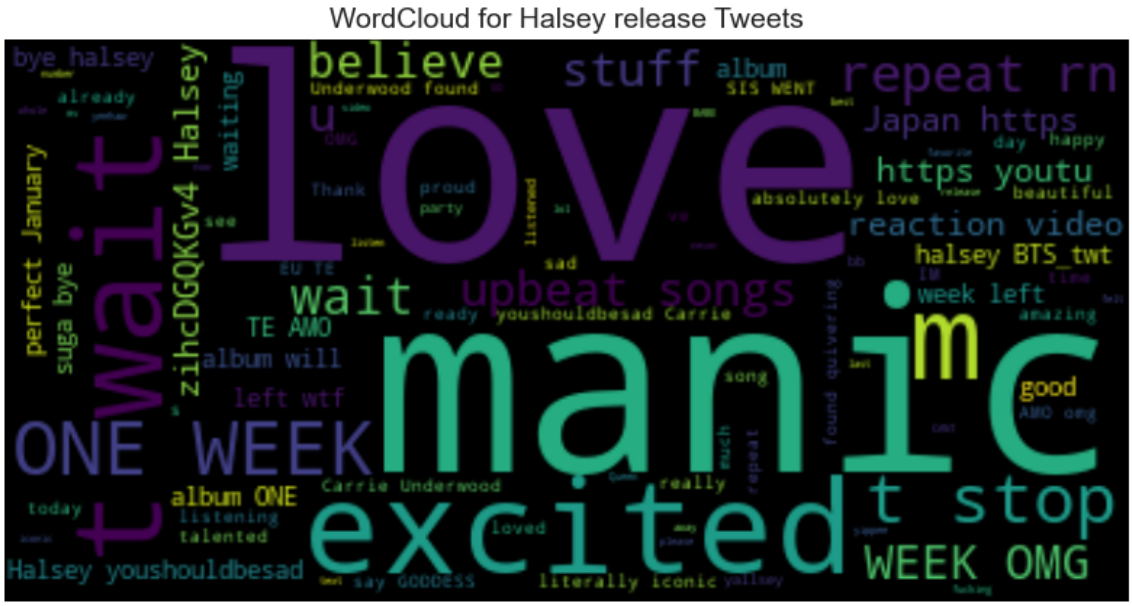
In [65]:
```
plot_hashtags(halsey_release_hashtags, 'Halsey release')
```



In [66]:
```
plot_users(halsey_release_users, 'Halsey release')
```



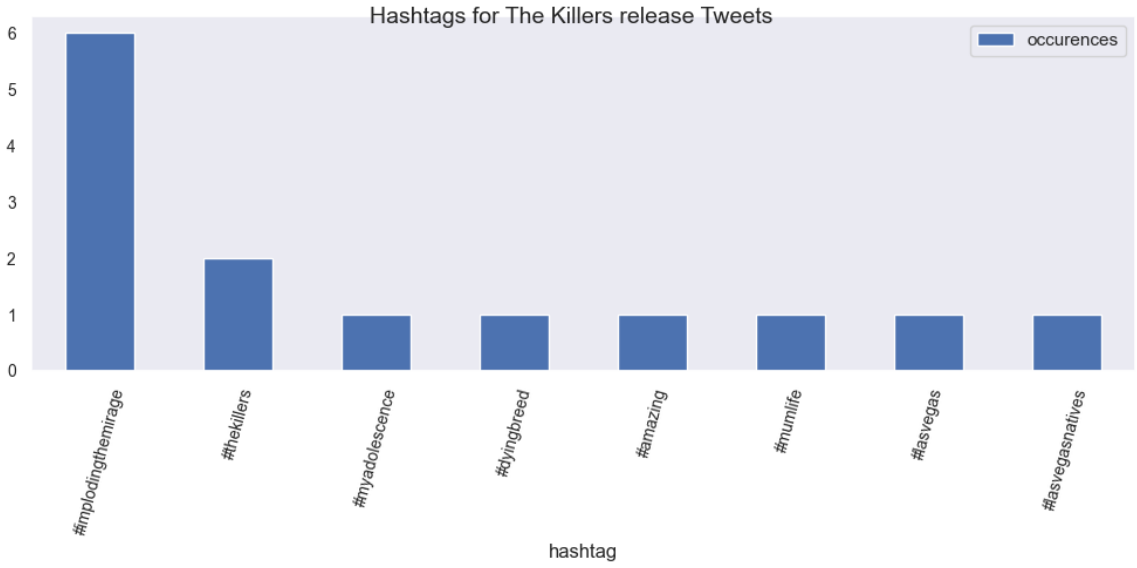
In [67]:
```
plot_wordcloud(halsey_release, 'Halsey release')
```



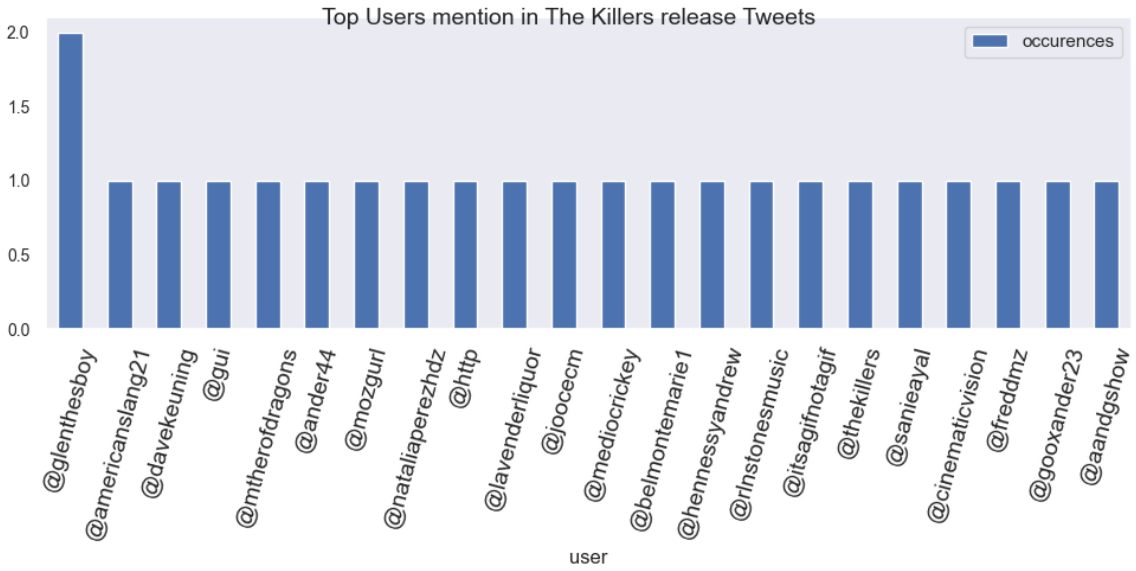
The Killers

In [68]:
```
killers_release_hashtags = extract_hashtags(killers_release, 'Tweet_Conte
killers_release_users = extract_users(killers_release, 'Tweet_Content')
```

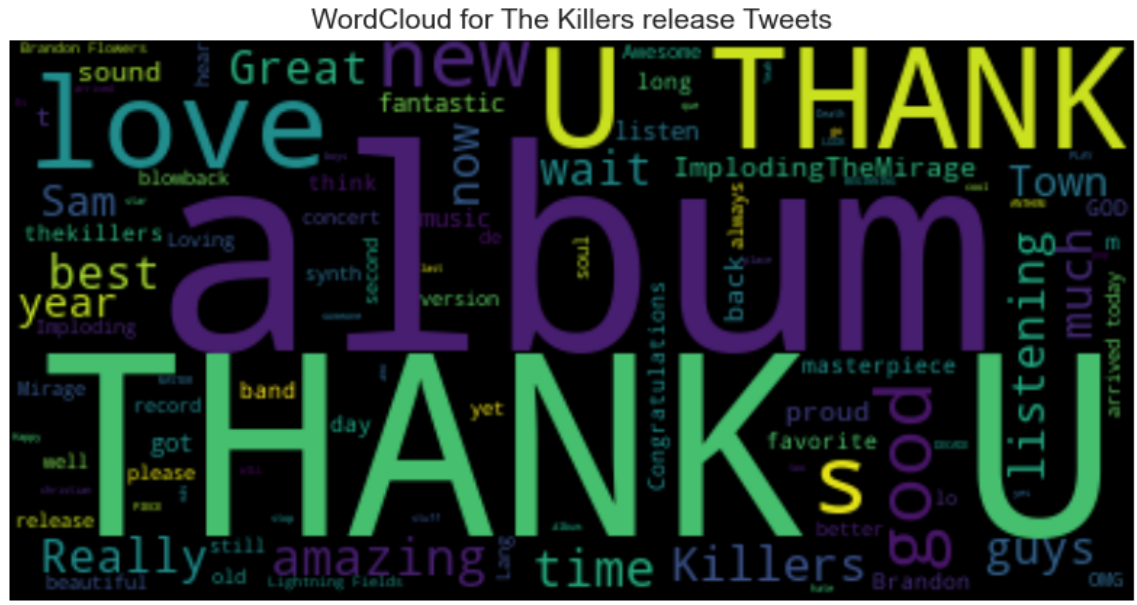
In [69]:
```
plot_hashtags(killers_release_hashtags, 'The Killers release')
```



In [70]:
```
plot_users(killers_release_users, 'The Killers release')
```



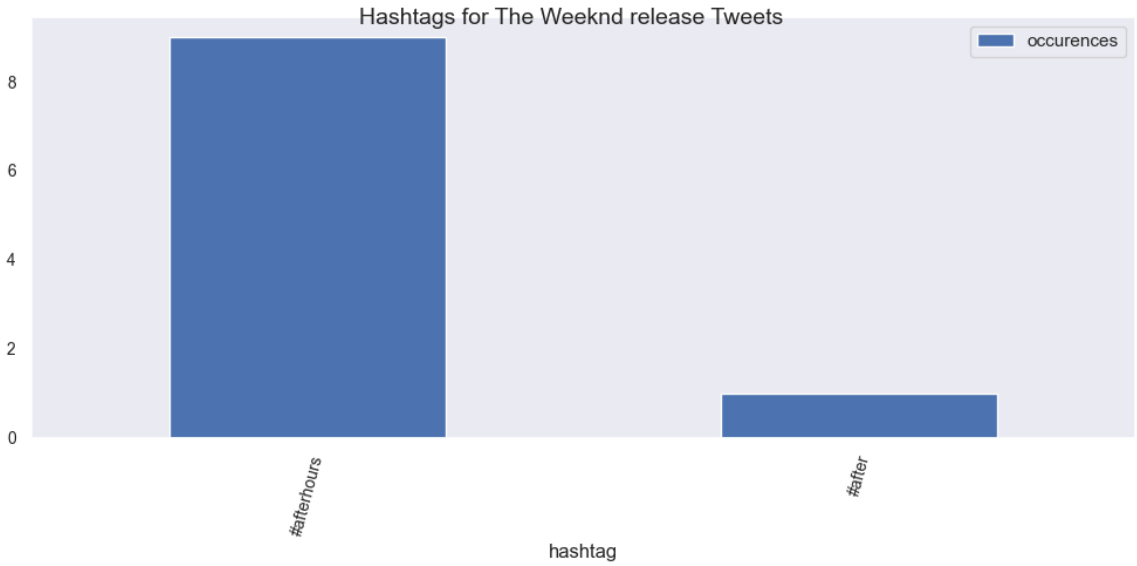
In [71]:
```
plot_wordcloud(killers_release, 'The Killers release')
```



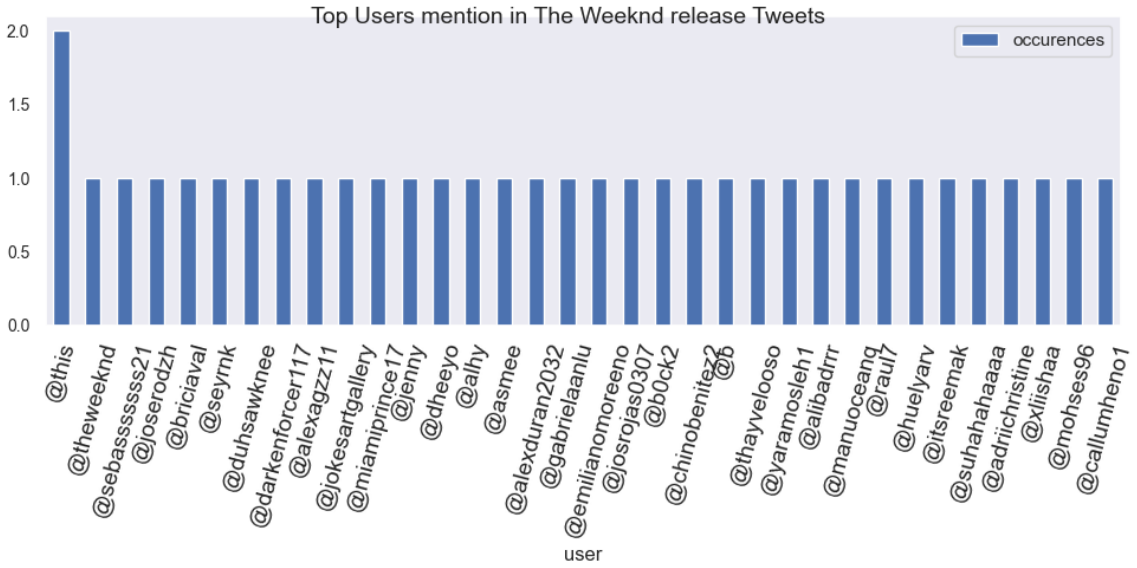
The Weeknd

In [72]:
```
weeknd_release_hashtags = extract_hashtags(weeknd_release, 'Tweet_Content
weeknd_release_users = extract_users(weeknd_release, 'Tweet_Content')
```

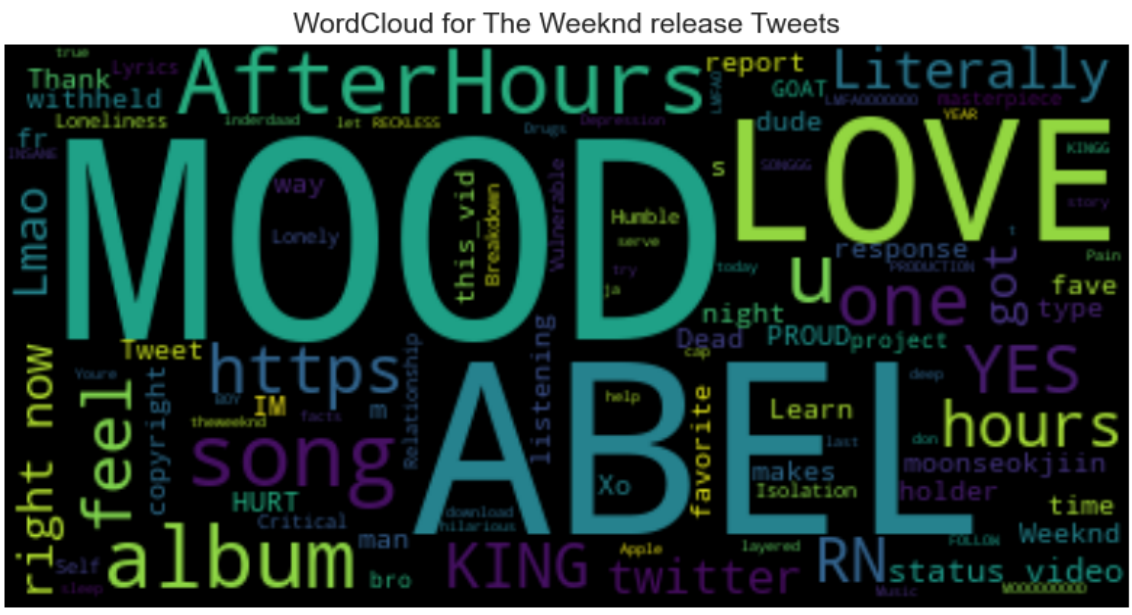
In [73]:
```
plot_hashtags(weeknd_release_hashtags, 'The Weeknd release')
```



In [74]:
```
plot_users(weeknd_release_users, 'The Weeknd release')
```



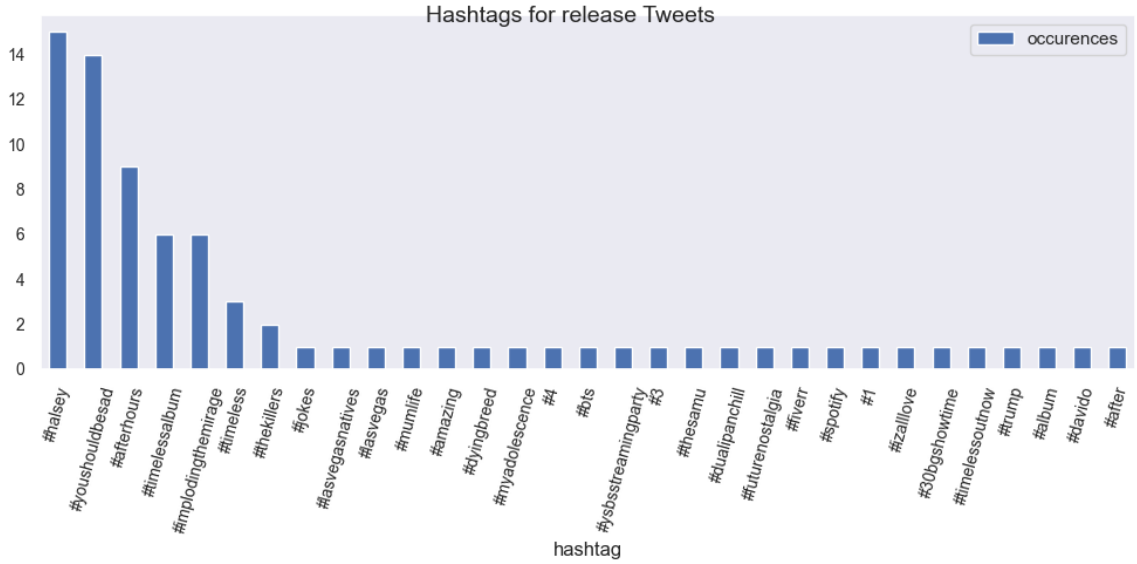
In [75]:
```
plot_wordcloud(weeknd_release, 'The Weeknd release')
```



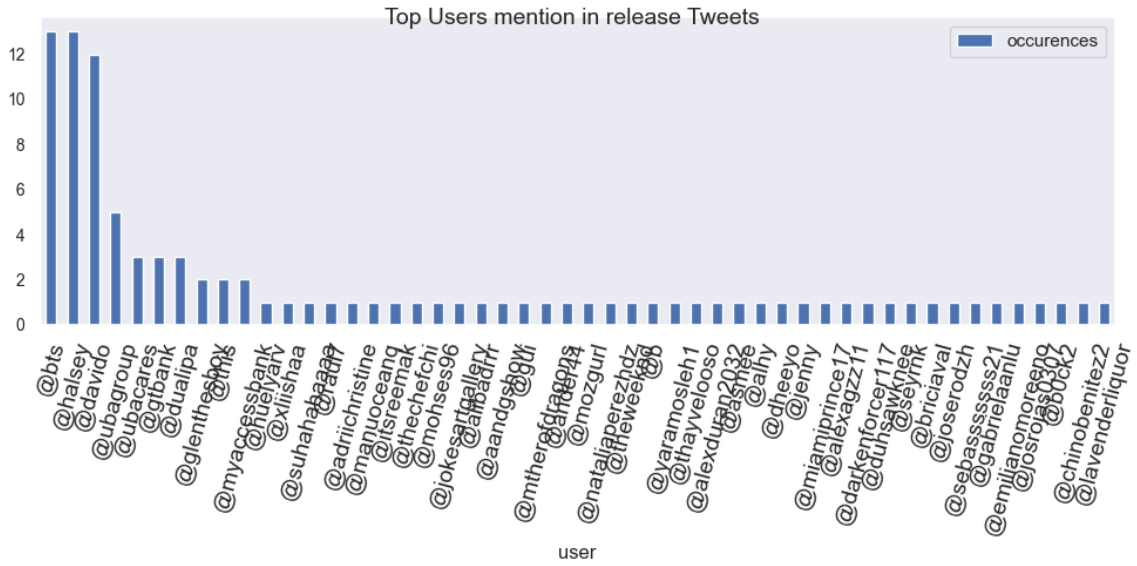
All Post-release tweets

In [76]:
```
release_df_hashtags = extract_hashtags(release_df, 'Tweet_Content')
release_df_users = extract_users(release_df, 'Tweet_Content')
```

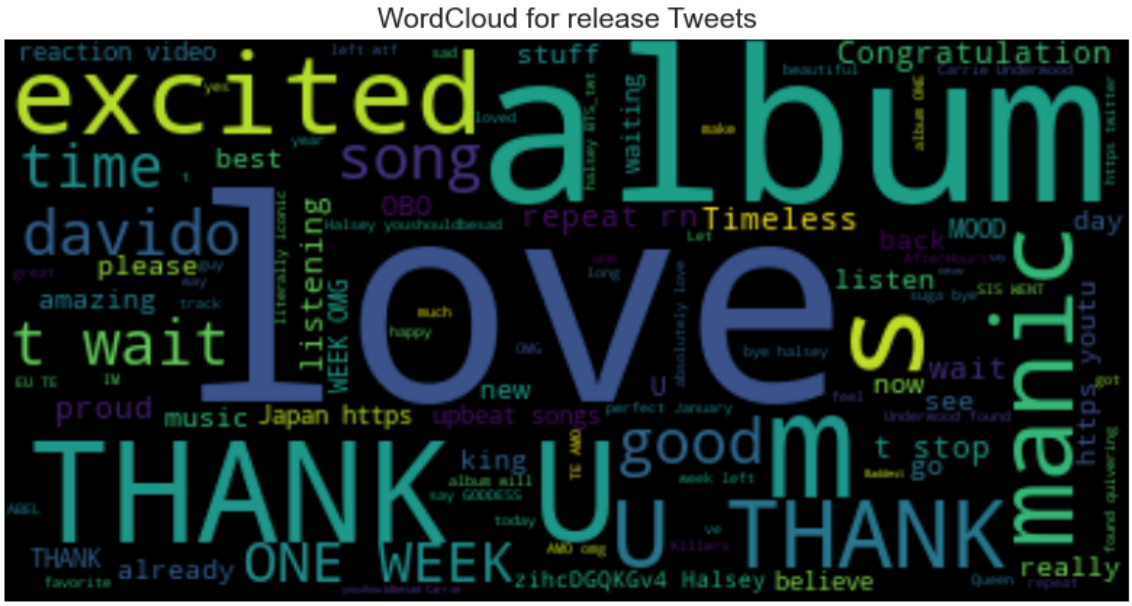
In [77]:
```
plot_hashtags(release_df_hashtags, 'release')
```



In [78]:
```
plot_users(release_df_users, 'release')
```



In [79]:
```
plot_wordcloud(release_df, 'release')
```



Artists Analysis: Pre-release & Post-release combined

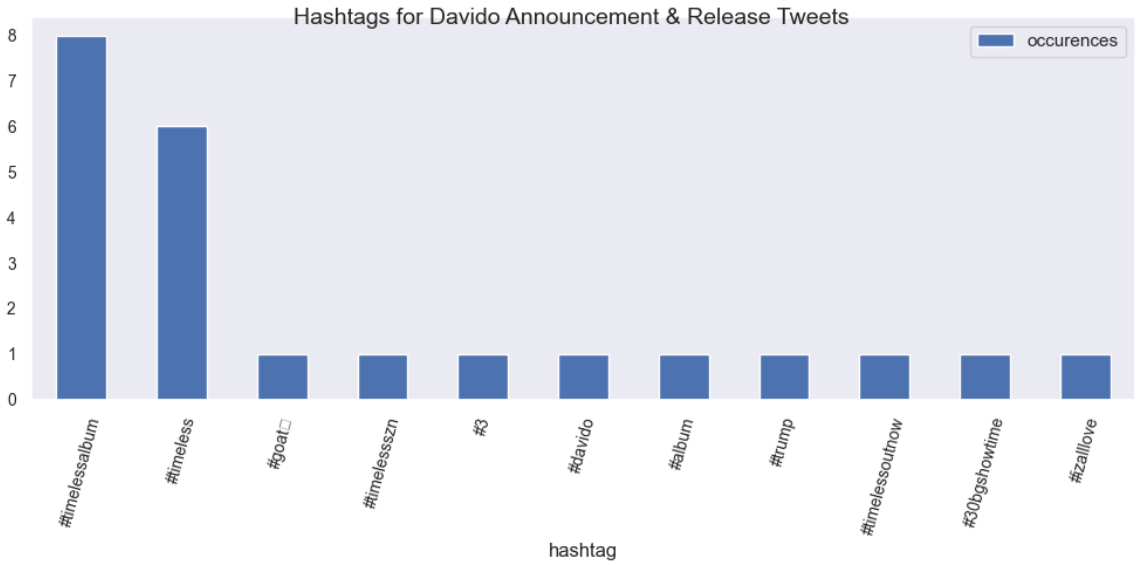
back to top

Davido

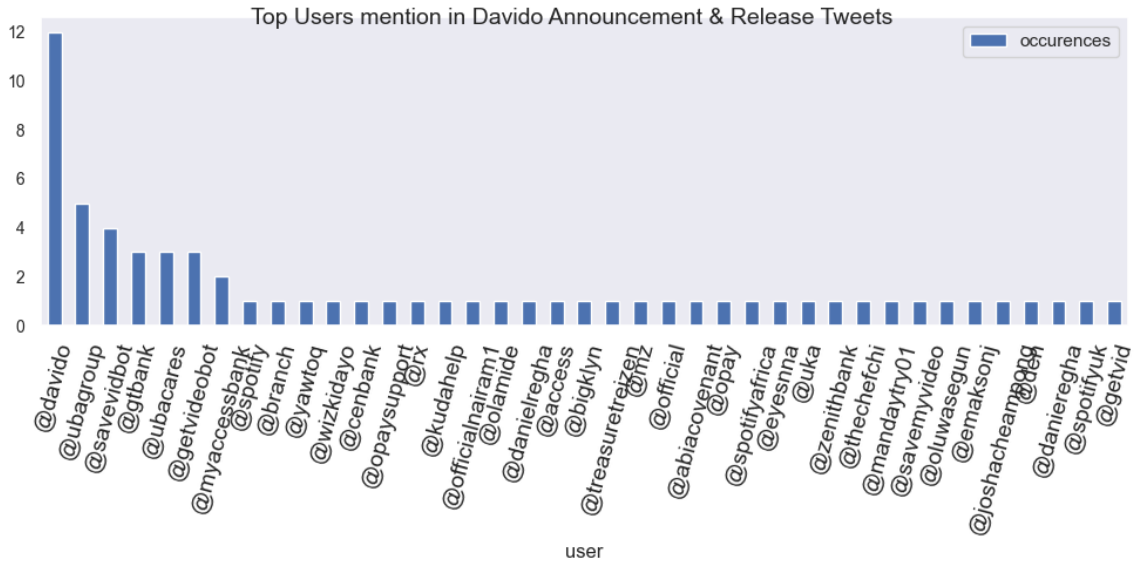
In [80]:
```
davido_df_hashtags = extract_hashtags(davido_df, 'Tweet_Content')
```

In [81]:
```
davido_df_users = extract_users(davido_df, 'Tweet_Content')
```

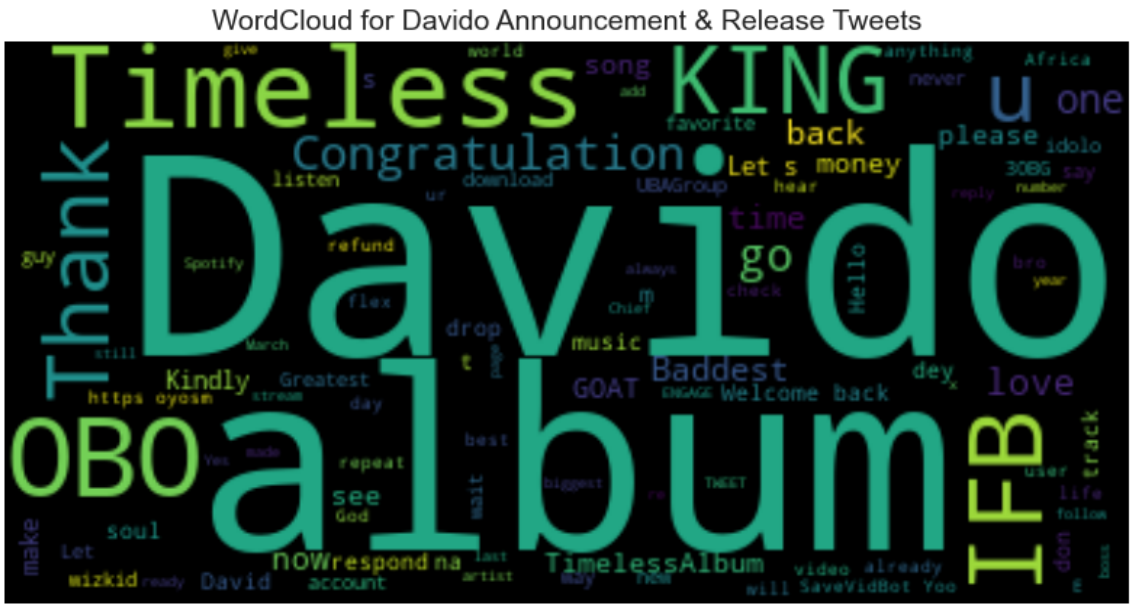
In [82]:
```
plot_hashtags(davido_df_hashtags, "Davido Announcement & Release")
```



In [83]:
```
plot_users(davido_df_users, "Davido Announcement & Release")
```



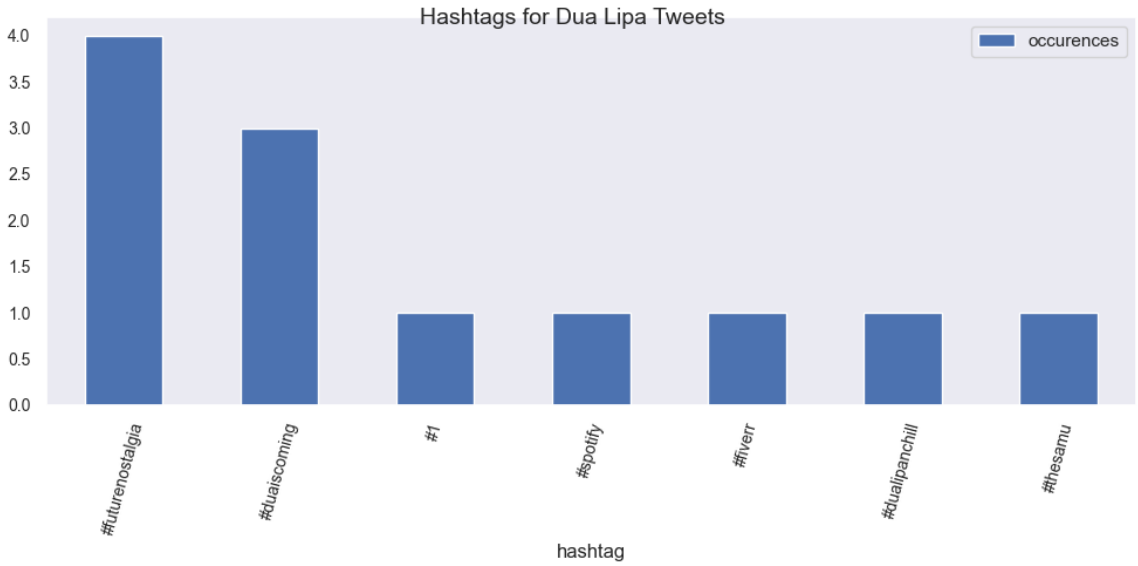
In [84]:
```
plot_wordcloud(davido_df, "Davido Announcement & Release")
```



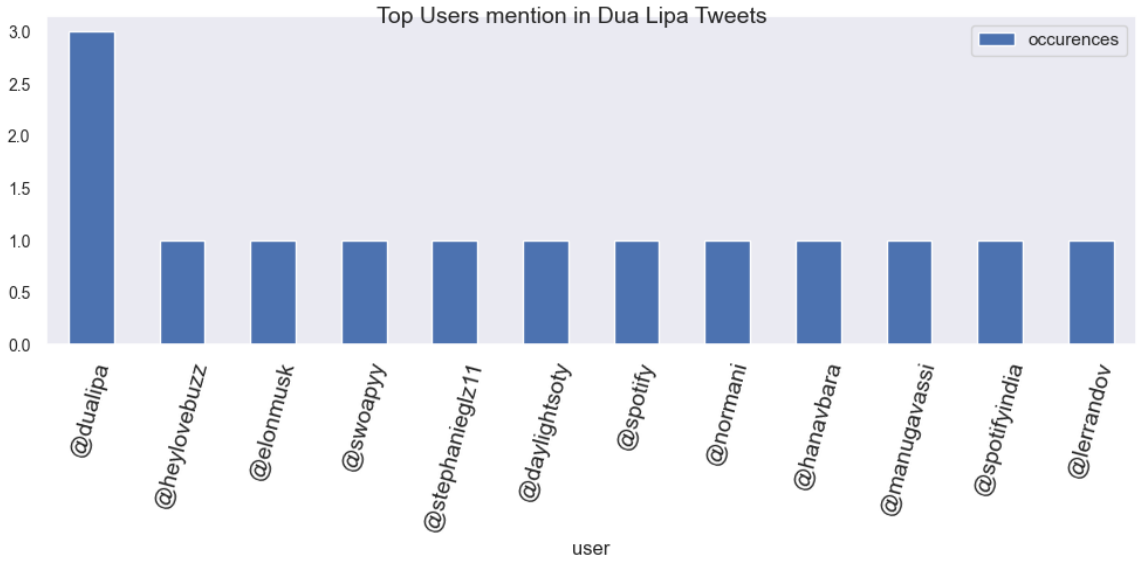
Dua Lipa

In [85]:
```
dualipa_df_hashtags = extract_hashtags(dualipa_df, 'Tweet_Content')
dualipa_df_users = extract_users(dualipa_df, 'Tweet_Content')
```

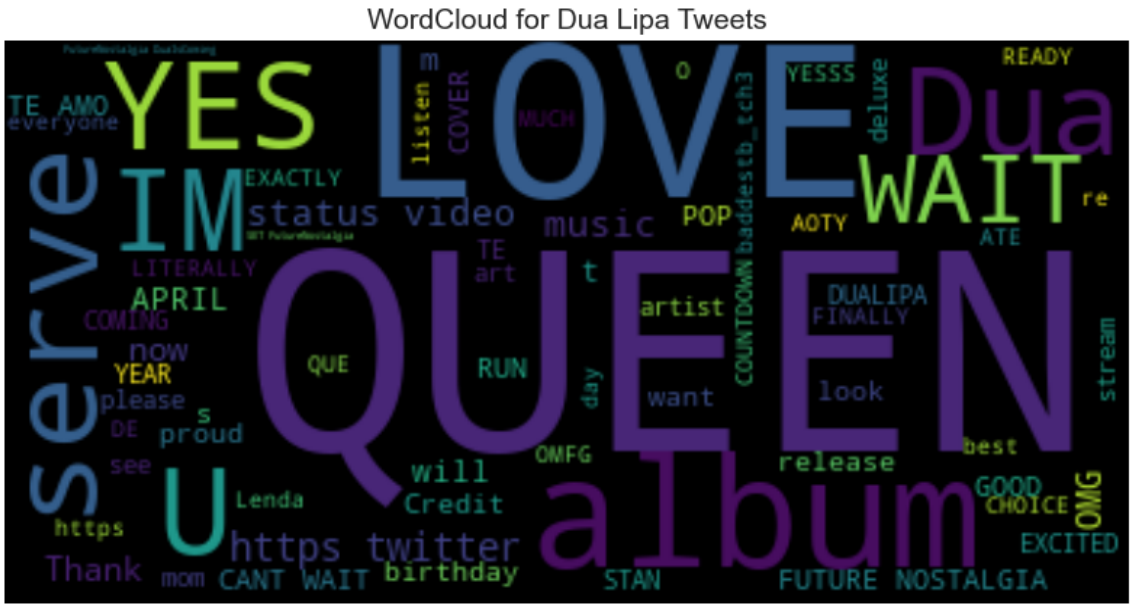
In [86]:
```
plot_hashtags(dualipa_df_hashtags, 'Dua Lipa')
```



In [87]:
```
plot_users(dualipa_df_users, 'Dua Lipa')
```



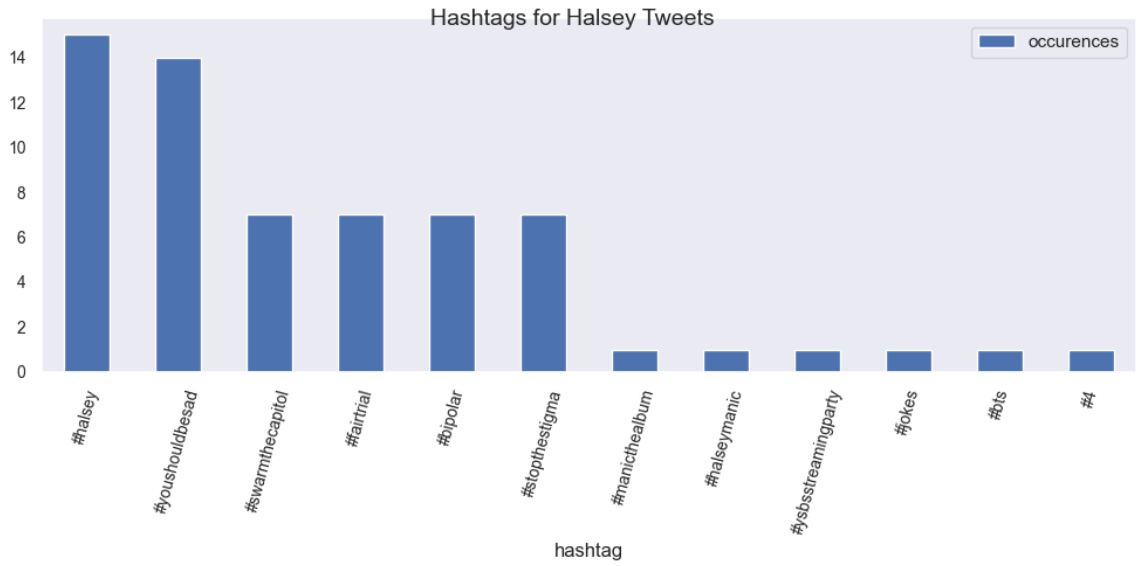
In [88]:
```
plot_wordcloud(dualipa_df, 'Dua Lipa')
```



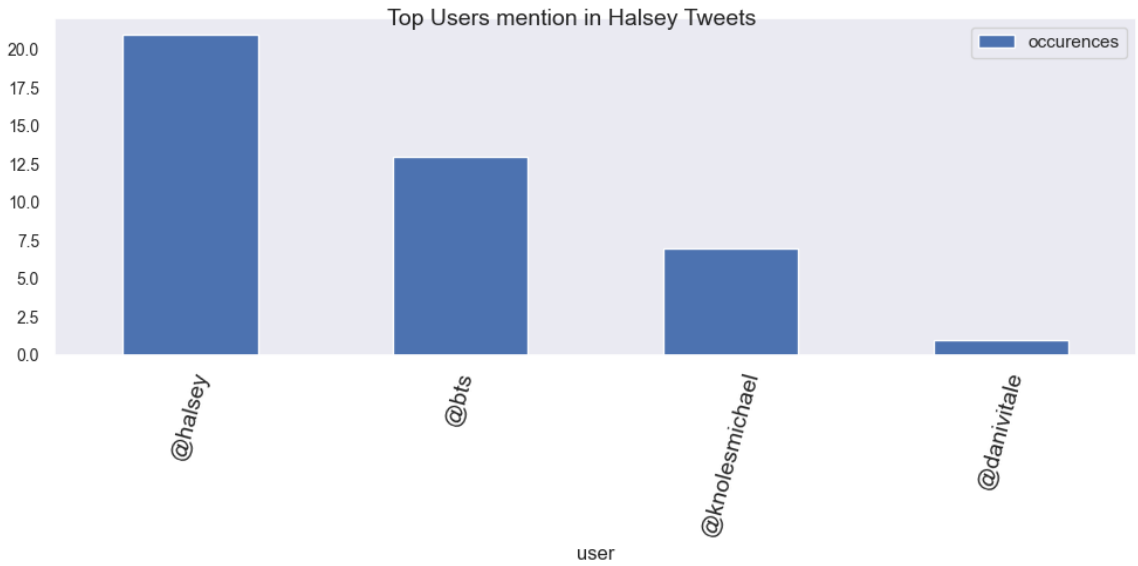
Halsey

In [89]:
```
halsey_df_hashtags = extract_hashtags(halsey_df, 'Tweet_Content')
halsey_df_users = extract_users(halsey_df, 'Tweet_Content')
```

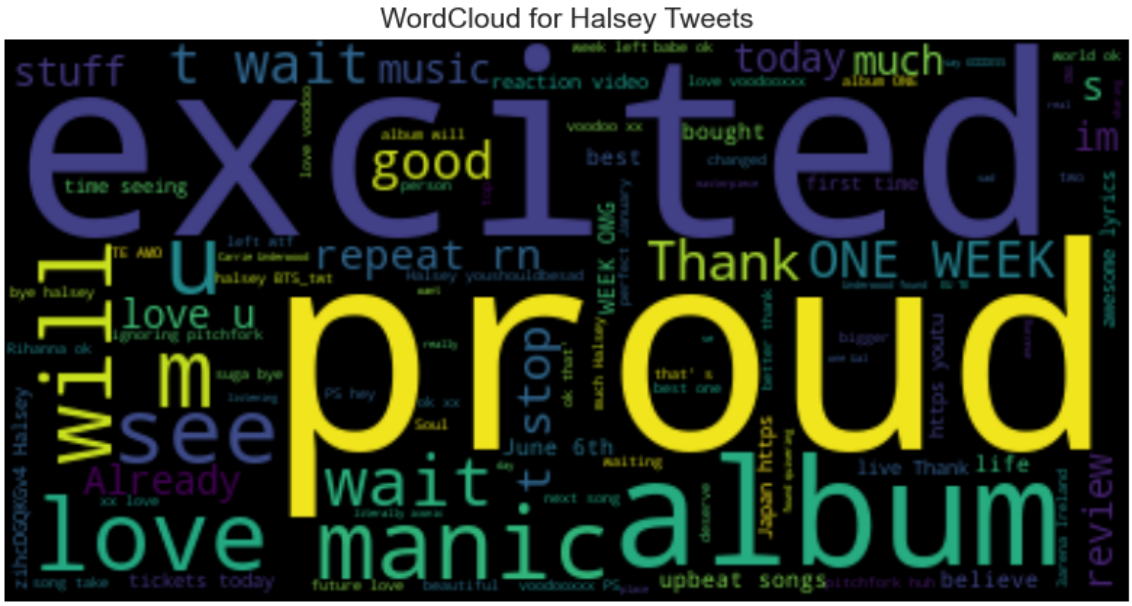
In [90]:
```
plot_hashtags(halsey_df_hashtags, "Halsey")
```



In [91]:
```
plot_users(halsey_df_users, 'Halsey')
```



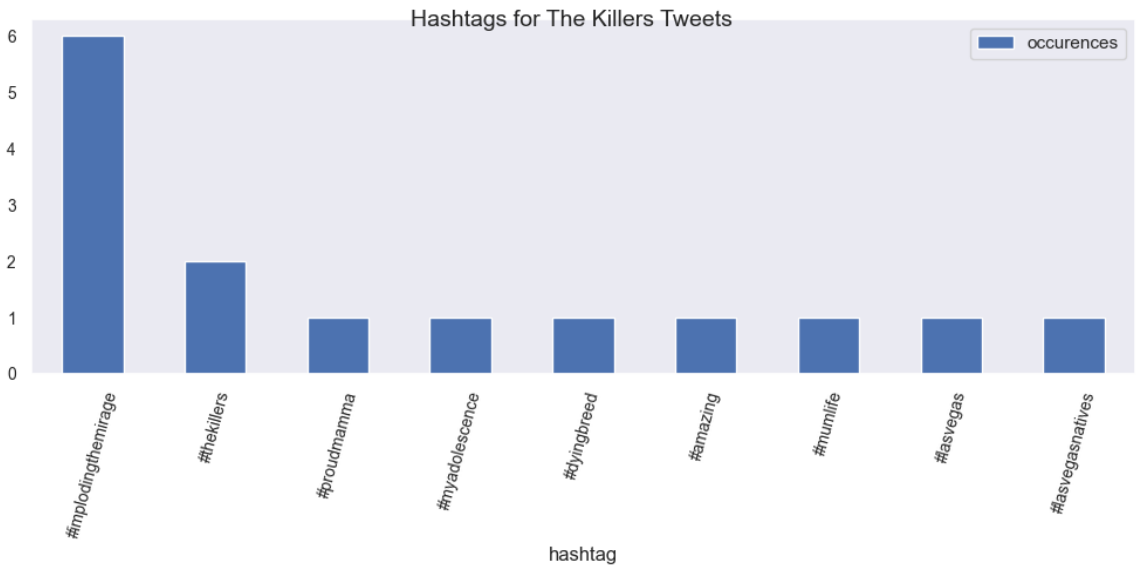
In [92]:
```
plot_wordcloud(halsey_df, 'Halsey')
```



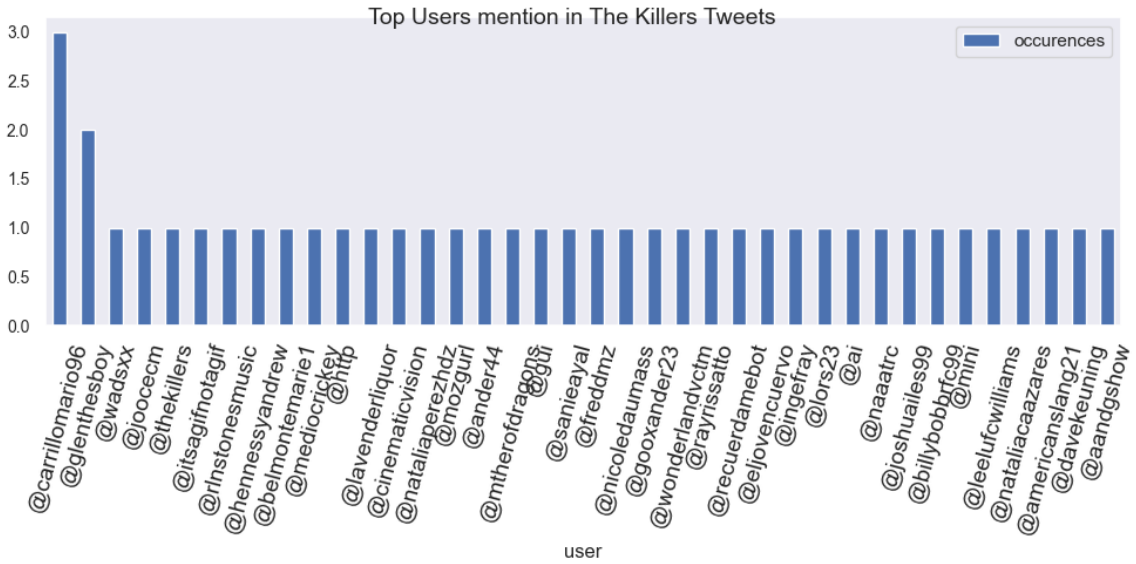
The Killers

In [93]:
```
killers_df_hashtags = extract_hashtags(killers_df, 'Tweet_Content')
killers_df_users = extract_users(killers_df, 'Tweet_Content')
```

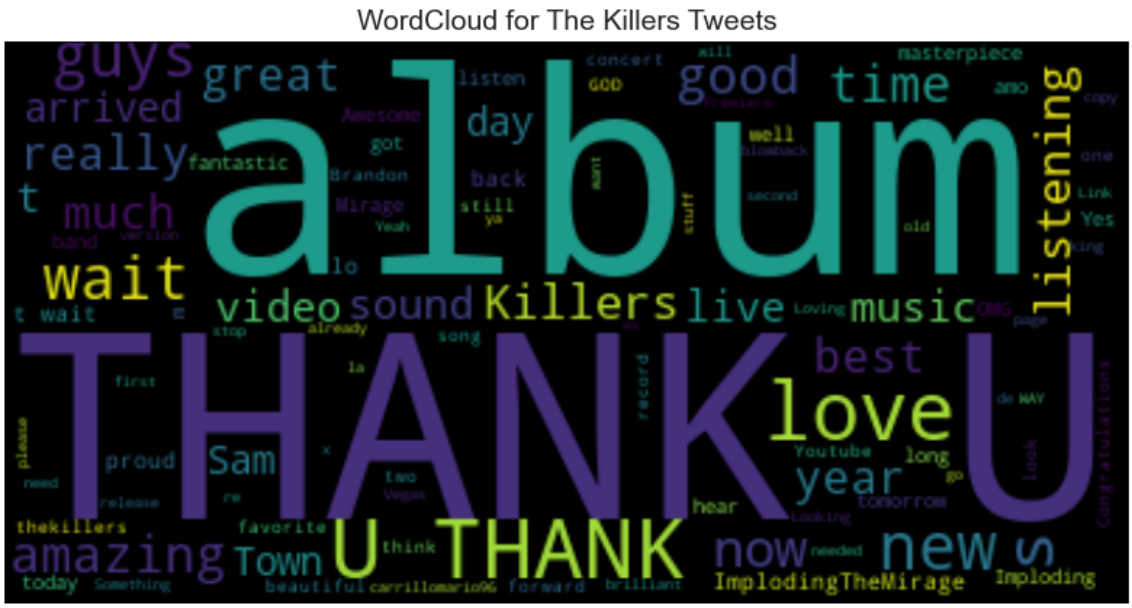
In [94]:
```
plot_hashtags(killers_df_hashtags, 'The Killers')
```



In [95]:
```
plot_users(killers_df_users, 'The Killers')
```



In [96]:
```
plot_wordcloud(killers_df, 'The Killers')
```

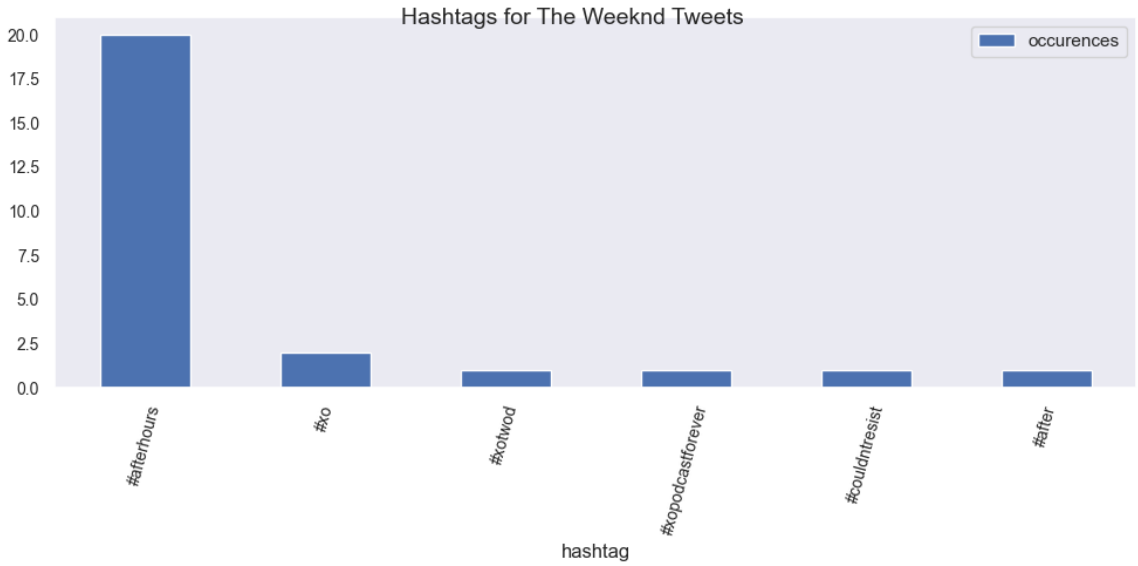


The Weeknd

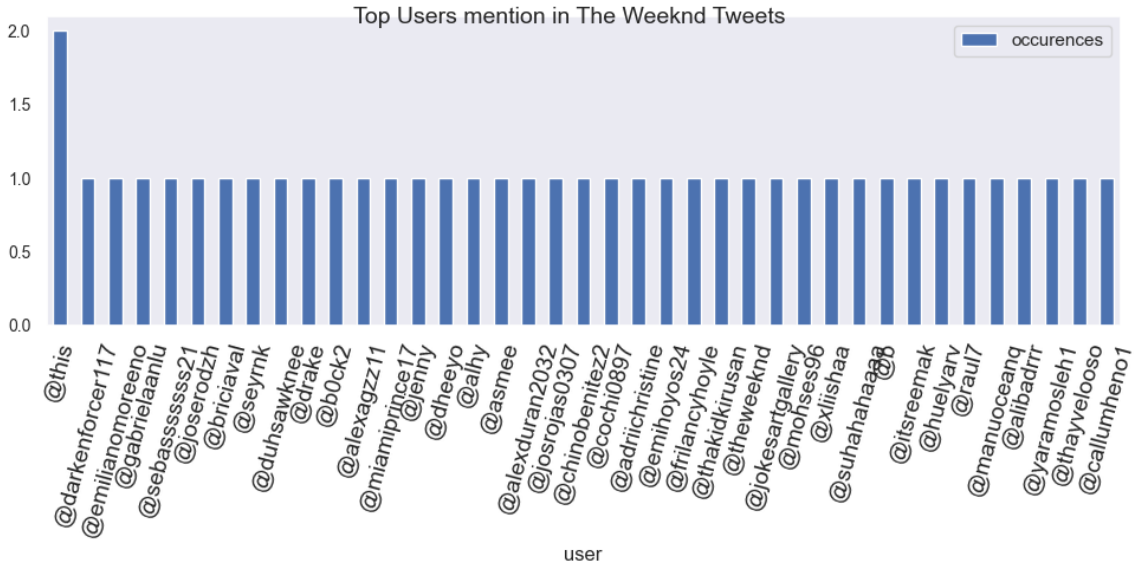
In [97]:
```
weeknd_df = combine_dataframes([weeknd_announcement, weeknd_release])
```

In [98]:
```
weeknd_df_hashtags = extract_hashtags(weeknd_df, 'Tweet_Content')
weeknd_df_users = extract_users(weeknd_df, 'Tweet_Content')
```

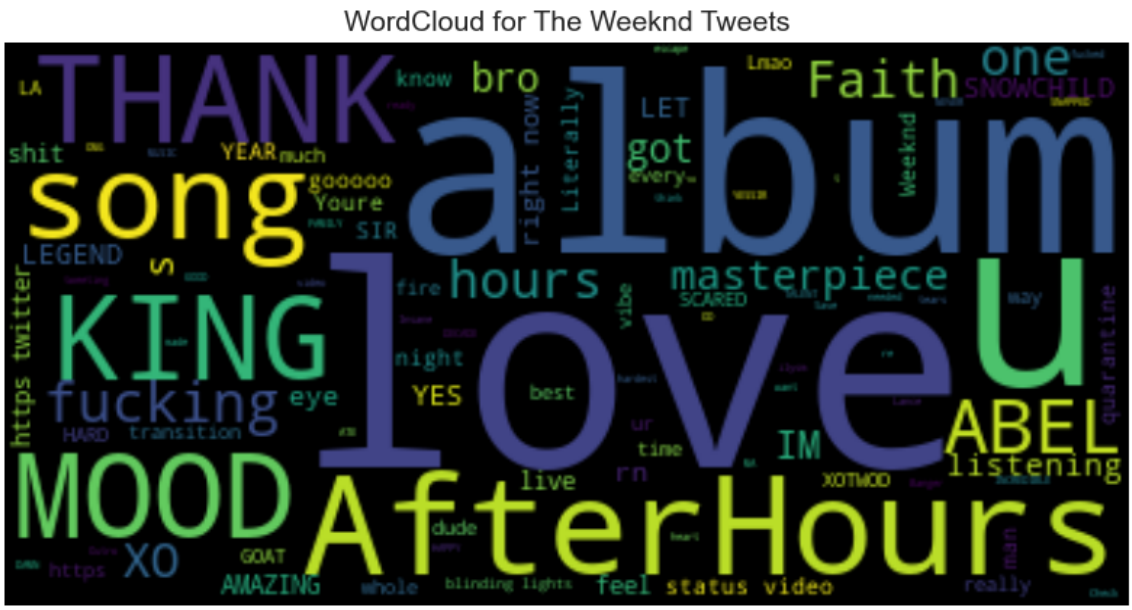
In [99]:
```
plot_hashtags(weeknd_df_hashtags, 'The Weeknd')
```



In [100…
```
plot_users(weeknd_df_users, 'The Weeknd')
```



In [101…
```
plot_wordcloud(weeknd_df, 'The Weeknd')
```

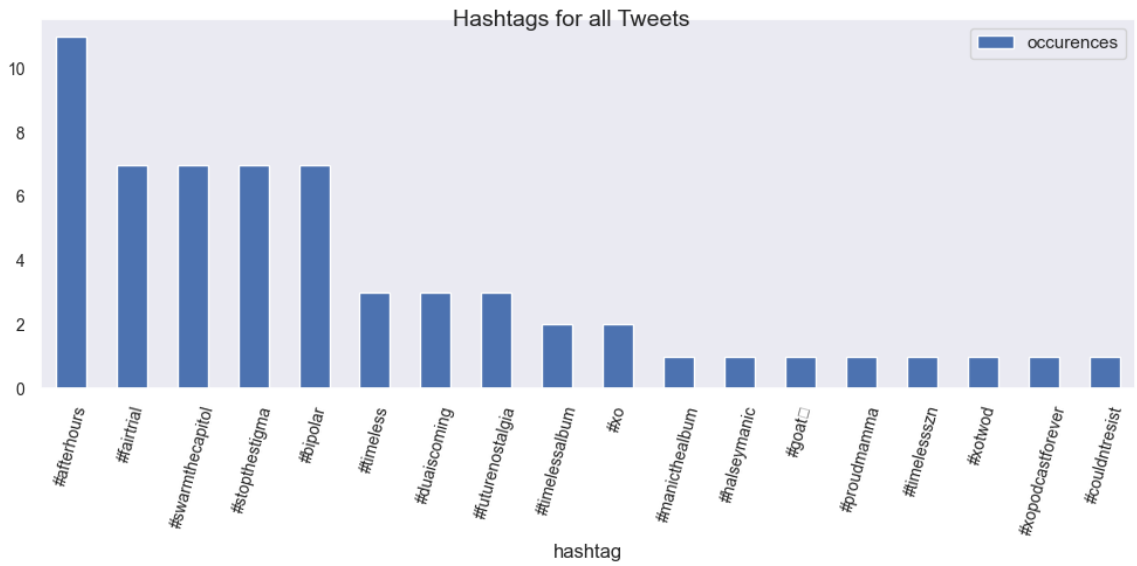


All Tweets Analysis

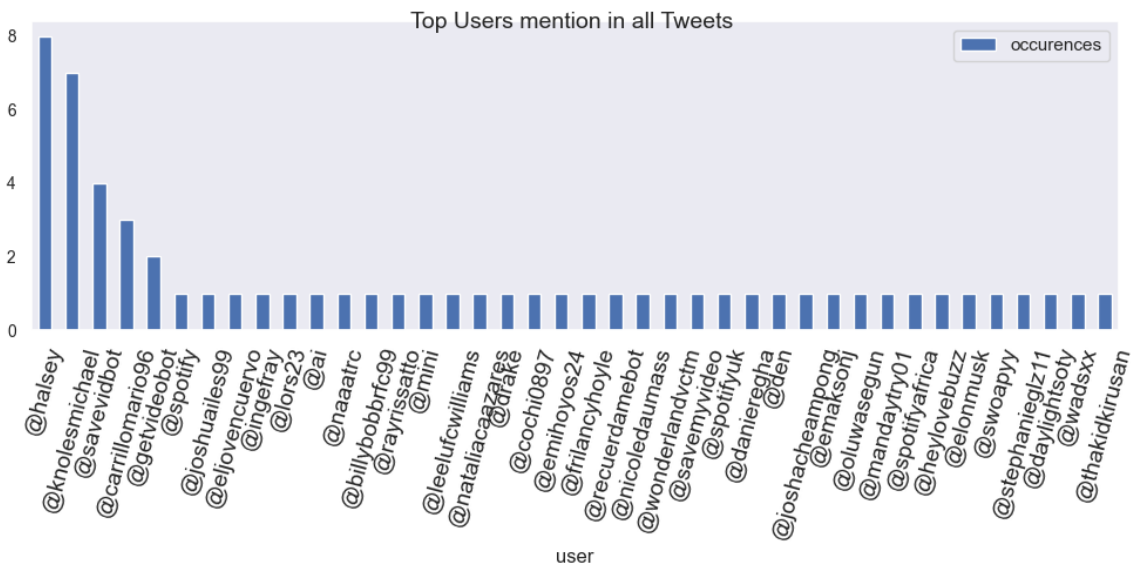
In [102…
```
all_df = combine_dataframes([davido_announcement, dualipa_announcement,
                             halsey_announcement, killers_announcement,
                             weeknd_announcement, davido_release,
                             dualipa_release, halsey_release,
                             killers_release, weeknd_release])
```

In [103…
```
all_df_hashtags = extract_hashtags(announcement_df, 'Tweet_Content')
all_df_users = extract_users(announcement_df, 'Tweet_Content')
```

In [104…
```
plot_hashtags(all_df_hashtags, 'all')
```

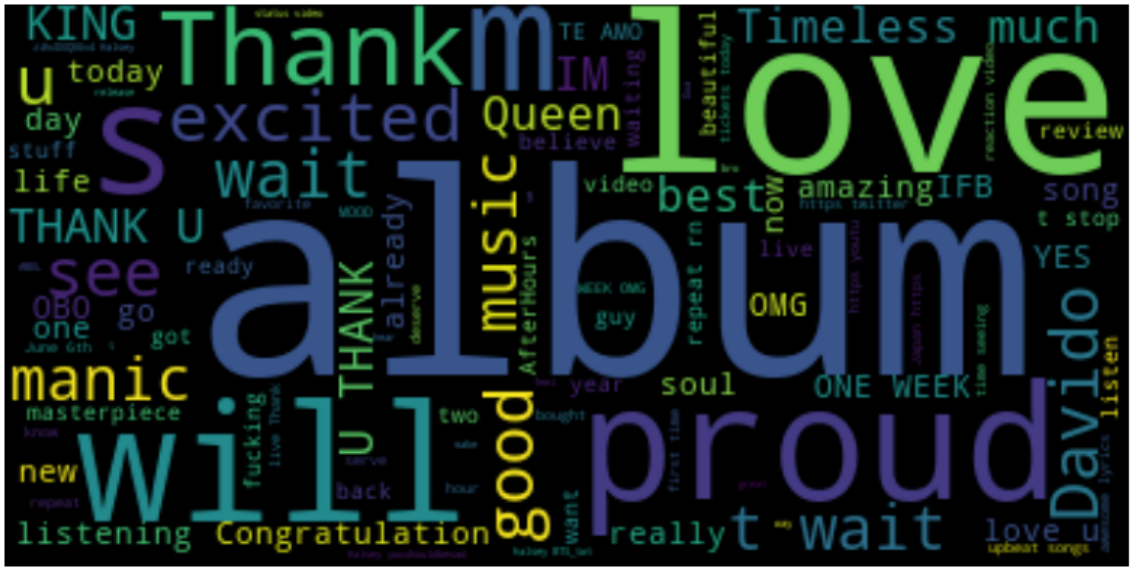


In [105…
```
plot_users(all_df_users, 'all')
```



In [106…
```
plot_wordcloud(all_df, 'all')
```

WordCloud for all Tweets



Sentiment Analysis using Vader

back to top

In [107…
```
all_df['Tweet_Number_of_Likes'] = all_df['Tweet_Number_of_Likes'].str.rep
all_df['Tweet_Number_of_Retweets'] = all_df['Tweet_Number_of_Retweets'].s
all_df['Tweet_Number_of_Reviews'] = all_df['Tweet_Number_of_Reviews'].fil
```

Artists

back to top

Davido

In [108…
```
davido_announcement['Tweet_Content'] = davido_announcement.Tweet_Content.
davido_announcement['Tweet_Content'] = clean_column(davido_announcement['
davido_announcement['Tweet_Content'] = davido_announcement.Tweet_Content.
davido_announcement['Sentiment'] = davido_announcement['Tweet_Content'].a

davido_release['Tweet_Content'] = davido_release.Tweet_Content.apply(text
davido_release['Tweet_Content'] = clean_column(davido_release['Tweet_Cont
davido_release['Tweet_Content'] = davido_release.Tweet_Content.apply(emoj
davido_release['Sentiment'] = davido_release['Tweet_Content'].apply(get_s

davido_df['Tweet_Content'] = davido_df.Tweet_Content.apply(text_preproc)
davido_df['Tweet_Content'] = clean_column(davido_df['Tweet_Content'])
davido_df['Tweet_Content'] = davido_df.Tweet_Content.apply(emoji_converte
davido_df['Sentiment'] = davido_df['Tweet_Content'].apply(get_sentiment_l
```

In [132…
```
davido_announcement['Sentiment'].value_counts()
```

Out[132]:
```
Neutral     114
Positive     33
Negative      5
Name: Sentiment, dtype: int64
```

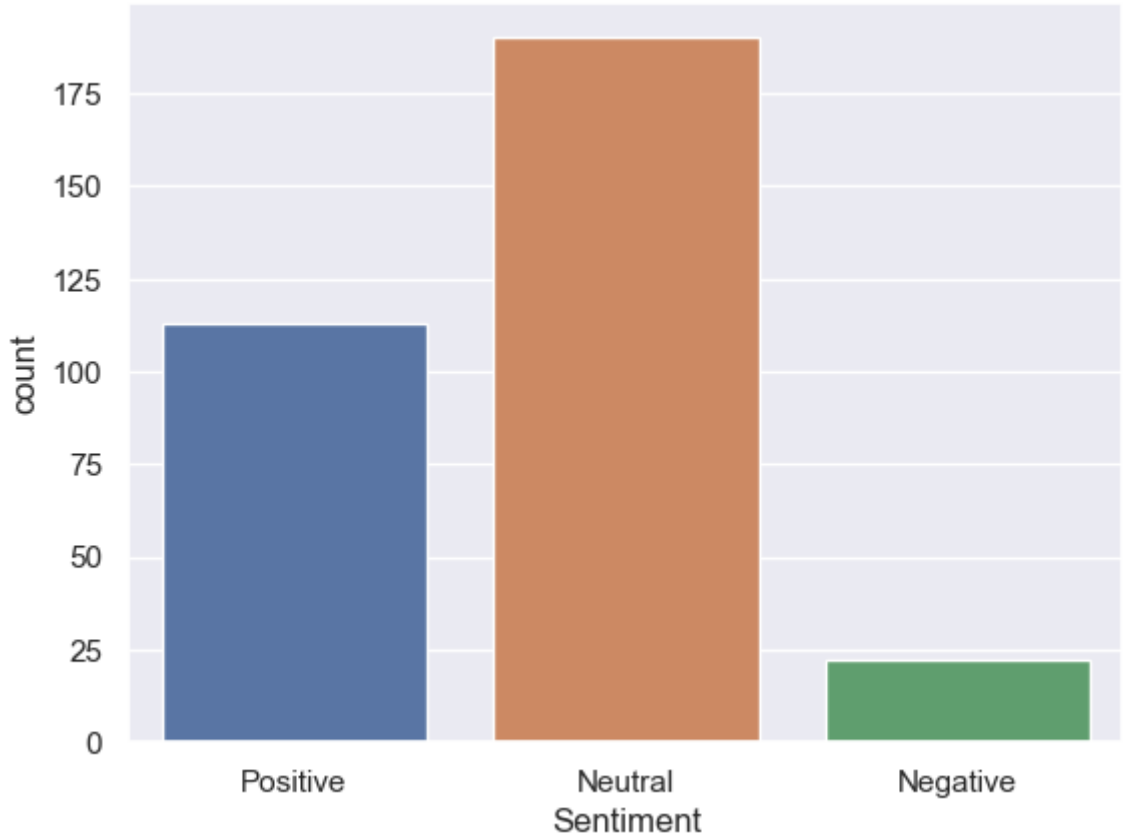
In [133…
```
davido_release['Sentiment'].value_counts()
```

Out[133]:
```
Positive    80
Neutral     76
Negative    17
Name: Sentiment, dtype: int64
```

In [109…
```
davido_df['Sentiment'].value_counts()
```

Out[109]:
```
Neutral     190
Positive    113
Negative     22
Name: Sentiment, dtype: int64
```

In [110…
```
plot_sentiment_distribution(davido_df, 'Sentiment')
```



Dua Lipa

In [111…
```
dualipa_announcement['Tweet_Content'] = dualipa_announcement.Tweet_Conten
dualipa_announcement['Tweet_Content'] = clean_column(dualipa_announcement
dualipa_announcement['Tweet_Content'] = dualipa_announcement.Tweet_Conten
dualipa_announcement['Sentiment'] = dualipa_announcement['Tweet_Content']

dualipa_release['Tweet_Content'] = dualipa_release.Tweet_Content.apply(te
dualipa_release['Tweet_Content'] = clean_column(dualipa_release['Tweet_Co
dualipa_release['Tweet_Content'] = dualipa_release.Tweet_Content.apply(em
dualipa_release['Sentiment'] = dualipa_release['Tweet_Content'].apply(get

dualipa_df['Tweet_Content'] = dualipa_df.Tweet_Content.apply(text_preproc
dualipa_df['Tweet_Content'] = clean_column(dualipa_df['Tweet_Content'])
dualipa_df['Tweet_Content'] = dualipa_df.Tweet_Content.apply(emoji_conver
dualipa_df['Sentiment'] = dualipa_df['Tweet_Content'].apply(get_sentiment
```

In [134…
```
dualipa_announcement['Sentiment'].value_counts()
```

Out[134]:
```
Neutral     131
Positive     54
Negative     16
Name: Sentiment, dtype: int64
```

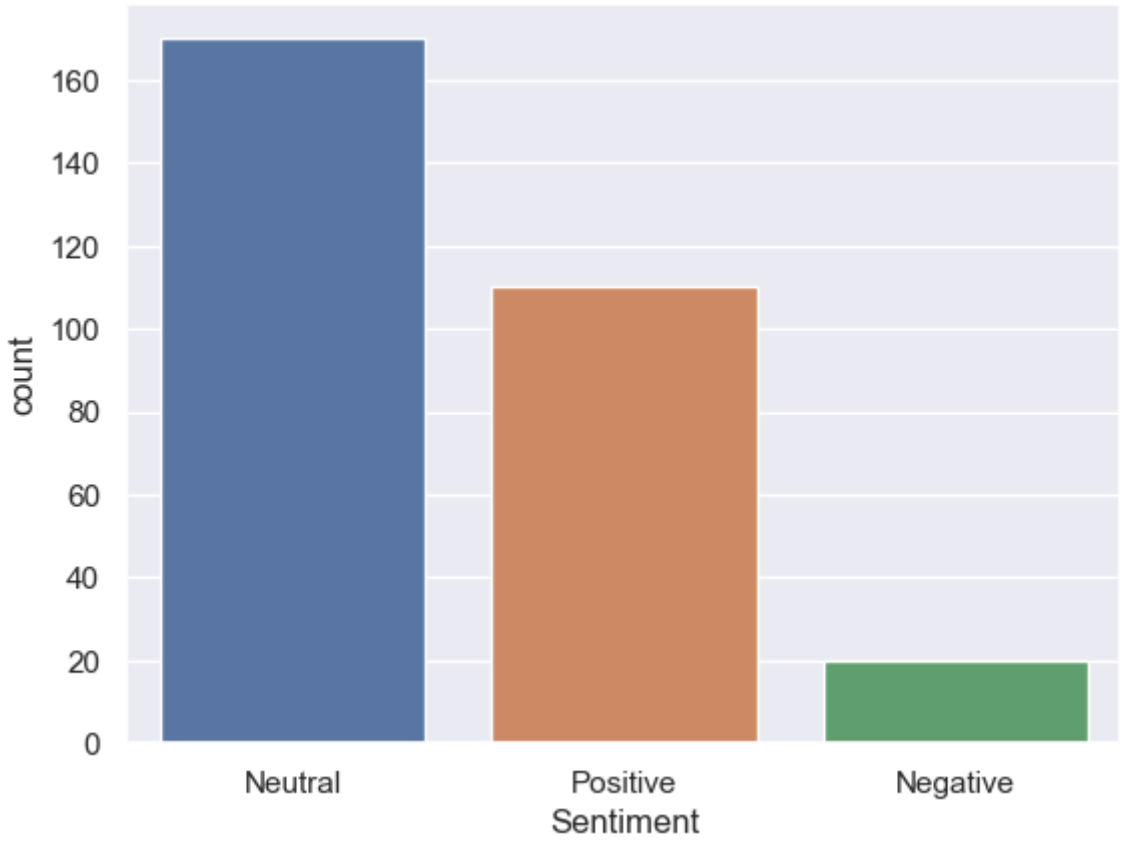
In [135…
```
dualipa_release['Sentiment'].value_counts()
```

Out[135]:
```
Positive    56
Neutral     39
Negative     4
Name: Sentiment, dtype: int64
```

In [112…
```
dualipa_df['Sentiment'].value_counts()
```

Out[112]:
```
Neutral     170
Positive    110
Negative     20
Name: Sentiment, dtype: int64
```

In [113…
```
plot_sentiment_distribution(dualipa_df, 'Sentiment')
```



Halsey

In [114…
```
halsey_announcement['Tweet_Content'] = halsey_announcement.Tweet_Content.
halsey_announcement['Tweet_Content'] = clean_column(halsey_announcement['
halsey_announcement['Tweet_Content'] = halsey_announcement.Tweet_Content.
halsey_announcement['Sentiment'] = halsey_announcement['Tweet_Content'].a

halsey_release['Tweet_Content'] = halsey_release.Tweet_Content.apply(text
halsey_release['Tweet_Content'] = clean_column(halsey_release['Tweet_Cont
halsey_release['Tweet_Content'] = halsey_release.Tweet_Content.apply(emoj
halsey_release['Sentiment'] = halsey_release['Tweet_Content'].apply(get_s

halsey_df['Tweet_Content'] = halsey_df.Tweet_Content.apply(text_preproc)
halsey_df['Tweet_Content'] = clean_column(halsey_df['Tweet_Content'])
halsey_df['Tweet_Content'] = halsey_df.Tweet_Content.apply(emoji_converte
halsey_df['Sentiment'] = halsey_df['Tweet_Content'].apply(get_sentiment_l
```

In [136…
```
halsey_announcement['Sentiment'].value_counts()
```

Out[136]:
```
Positive    176
Neutral      56
Negative     15
Name: Sentiment, dtype: int64
```

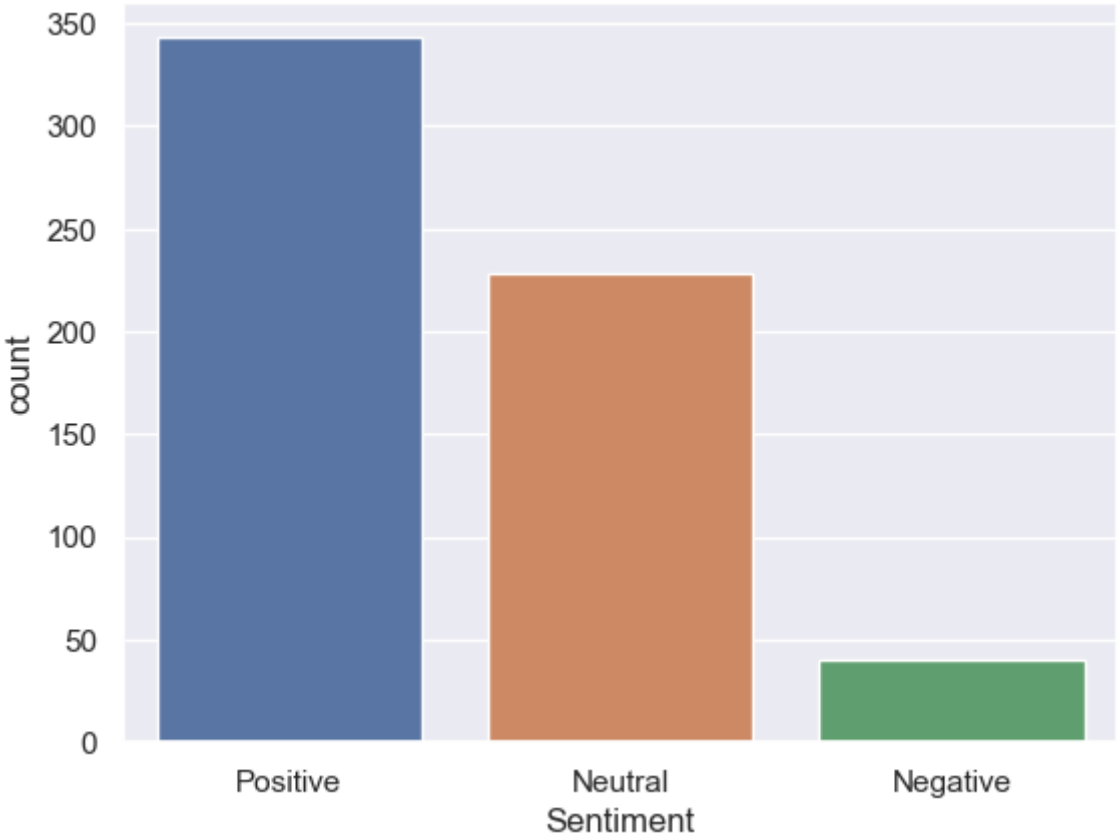
In [137…
```
halsey_release['Sentiment'].value_counts()
```

Out[137]:
```
Neutral     172
Positive    167
Negative     25
Name: Sentiment, dtype: int64
```

In [115…
```
halsey_df['Sentiment'].value_counts()
```

Out[115]:
```
Positive    343
Neutral     228
Negative     40
Name: Sentiment, dtype: int64
```

In [116…
```
plot_sentiment_distribution(halsey_df, 'Sentiment')
```



The Killers

In [117…
```
killers_announcement['Tweet_Content'] = killers_announcement.Tweet_Conten
killers_announcement['Tweet_Content'] = clean_column(killers_announcement
killers_announcement['Tweet_Content'] = killers_announcement.Tweet_Conten
killers_announcement['Sentiment'] = killers_announcement['Tweet_Content']

killers_release['Tweet_Content'] = killers_release.Tweet_Content.apply(te
killers_release['Tweet_Content'] = clean_column(killers_release['Tweet_Co
killers_release['Tweet_Content'] = killers_release.Tweet_Content.apply(em
killers_release['Sentiment'] = killers_release['Tweet_Content'].apply(get

killers_df['Tweet_Content'] = killers_df.Tweet_Content.apply(text_preproc
killers_df['Tweet_Content'] = clean_column(killers_df['Tweet_Content'])
killers_df['Tweet_Content'] = killers_df.Tweet_Content.apply(emoji_conver
killers_df['Sentiment'] = killers_df['Tweet_Content'].apply(get_sentiment
```

In [138…
```
killers_announcement['Sentiment'].value_counts()
```

Out[138]:
```
Neutral     45
Positive    36
Negative     5
Name: Sentiment, dtype: int64
```

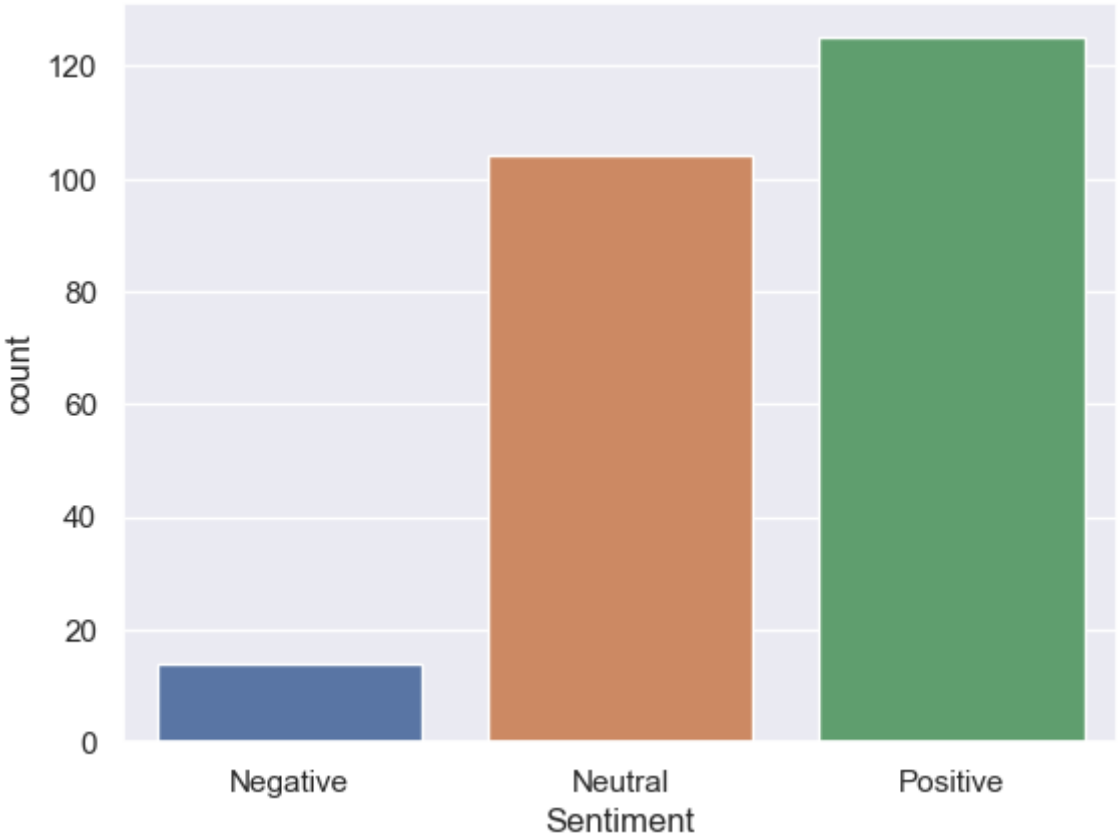
In [139…
```
killers_release['Sentiment'].value_counts()
```

Out[139]:
```
Positive    89
Neutral     59
Negative     9
Name: Sentiment, dtype: int64
```

In [118…
```
killers_df['Sentiment'].value_counts()
```

Out[118]:
```
Positive    125
Neutral     104
Negative     14
Name: Sentiment, dtype: int64
```

In [119…
```
plot_sentiment_distribution(killers_df, 'Sentiment')
```



The Weeknd

In [120…
```
weeknd_announcement['Tweet_Content'] = weeknd_announcement.Tweet_Content.
weeknd_announcement['Tweet_Content'] = clean_column(weeknd_announcement['
weeknd_announcement['Tweet_Content'] = weeknd_announcement.Tweet_Content.
weeknd_announcement['Sentiment'] = weeknd_announcement['Tweet_Content'].a

weeknd_release['Tweet_Content'] = weeknd_release.Tweet_Content.apply(text
weeknd_release['Tweet_Content'] = clean_column(weeknd_release['Tweet_Cont
weeknd_release['Tweet_Content'] = weeknd_release.Tweet_Content.apply(emoj
weeknd_release['Sentiment'] = weeknd_release['Tweet_Content'].apply(get_s

weeknd_df['Tweet_Content'] = weeknd_df.Tweet_Content.apply(text_preproc)
weeknd_df['Tweet_Content'] = clean_column(weeknd_df['Tweet_Content'])
weeknd_df['Tweet_Content'] = weeknd_df.Tweet_Content.apply(emoji_converte
weeknd_df['Sentiment'] = weeknd_df['Tweet_Content'].apply(get_sentiment_l
```

In [140…
```
weeknd_announcement['Sentiment'].value_counts()
```

Out[140]:
```
Neutral     73
Positive    70
Negative    21
Name: Sentiment, dtype: int64
```

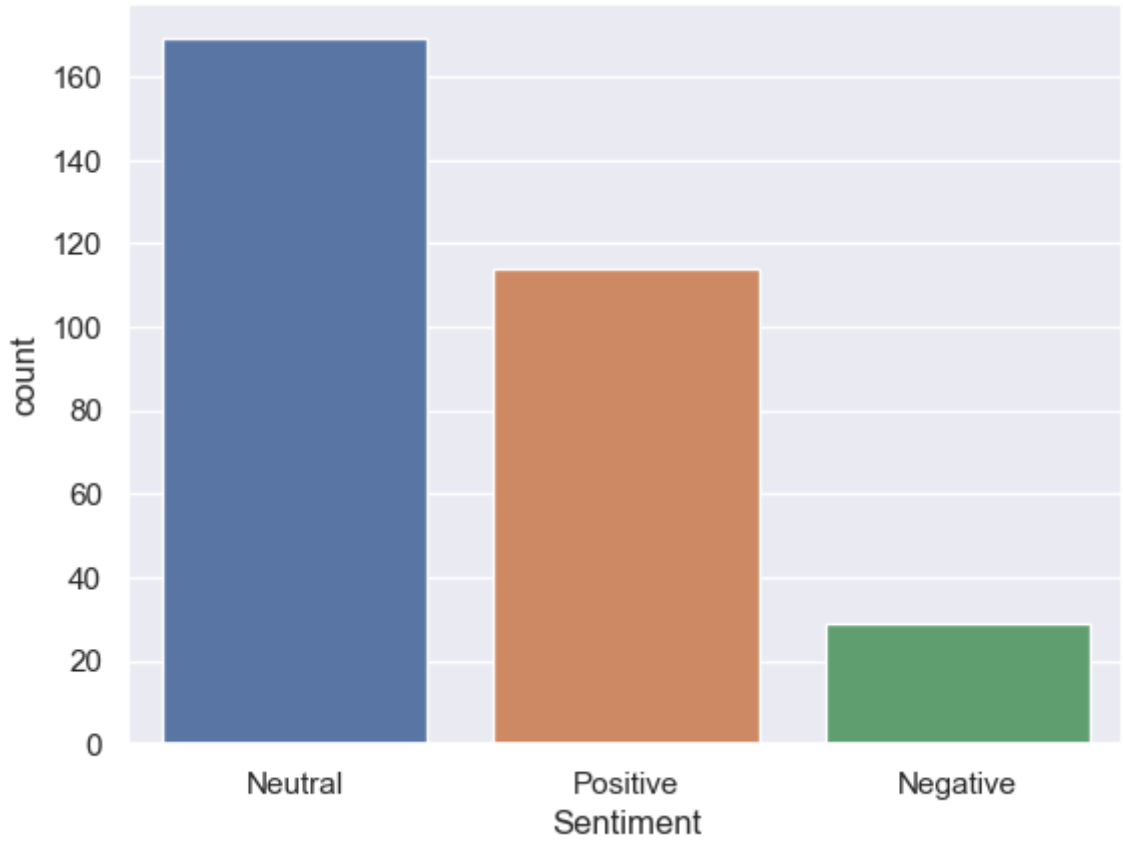
In [141…
```
weeknd_release['Sentiment'].value_counts()
```

Out[141]:
```
Neutral     96
Positive    44
Negative     8
Name: Sentiment, dtype: int64
```

In [121…
```
weeknd_df['Sentiment'].value_counts()
```

Out[121]:
```
Neutral     169
Positive    114
Negative     29
Name: Sentiment, dtype: int64
```

In [122…
```
plot_sentiment_distribution(weeknd_df, 'Sentiment')
```



Pre-release

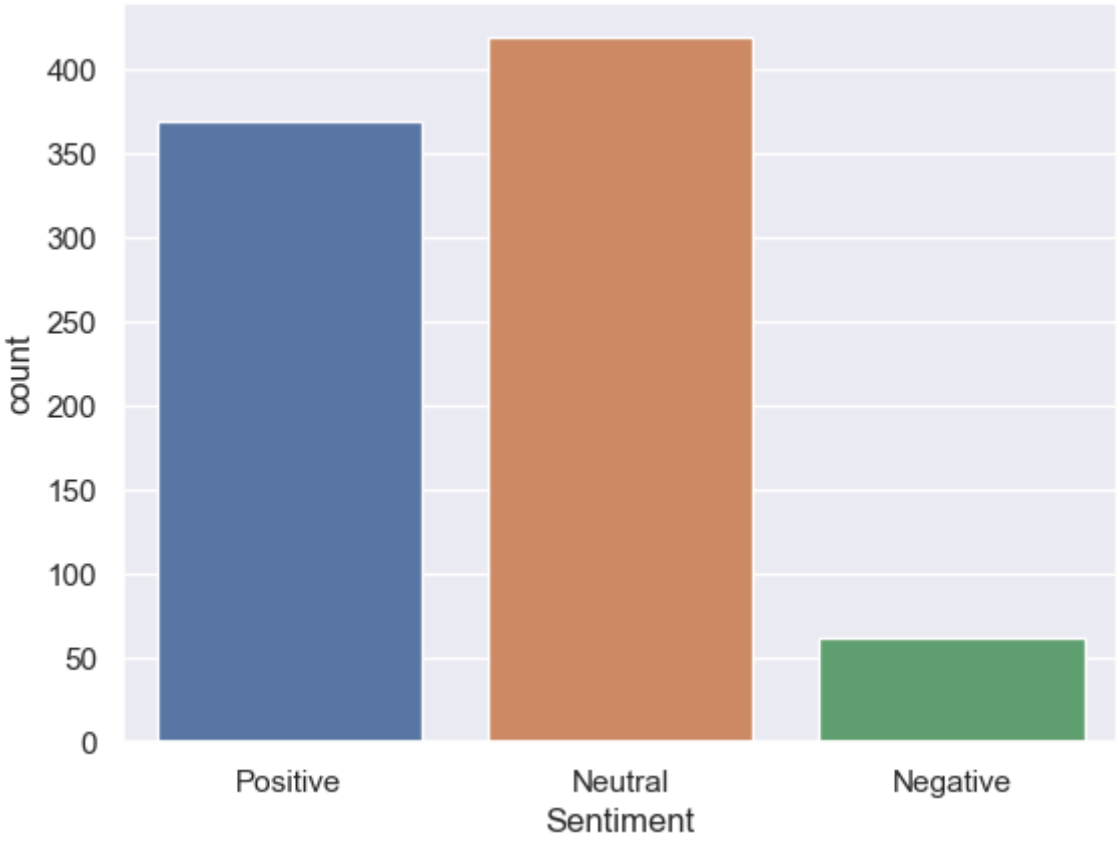
back to top

In [123…
```
announcement_df['Tweet_Content'] = announcement_df.Tweet_Content.apply(te
announcement_df['Tweet_Content'] = clean_column(announcement_df['Tweet_Co
announcement_df['Tweet_Content'] = announcement_df.Tweet_Content.apply(em
announcement_df['Sentiment'] = announcement_df['Tweet_Content'].apply(get
```

In [124…
```
announcement_df['Sentiment'].value_counts()
```

Out[124]:
```
Neutral     419
Positive    369
Negative     62
Name: Sentiment, dtype: int64
```

In [125…
```
plot_sentiment_distribution(announcement_df, 'Sentiment')
```



Post-release

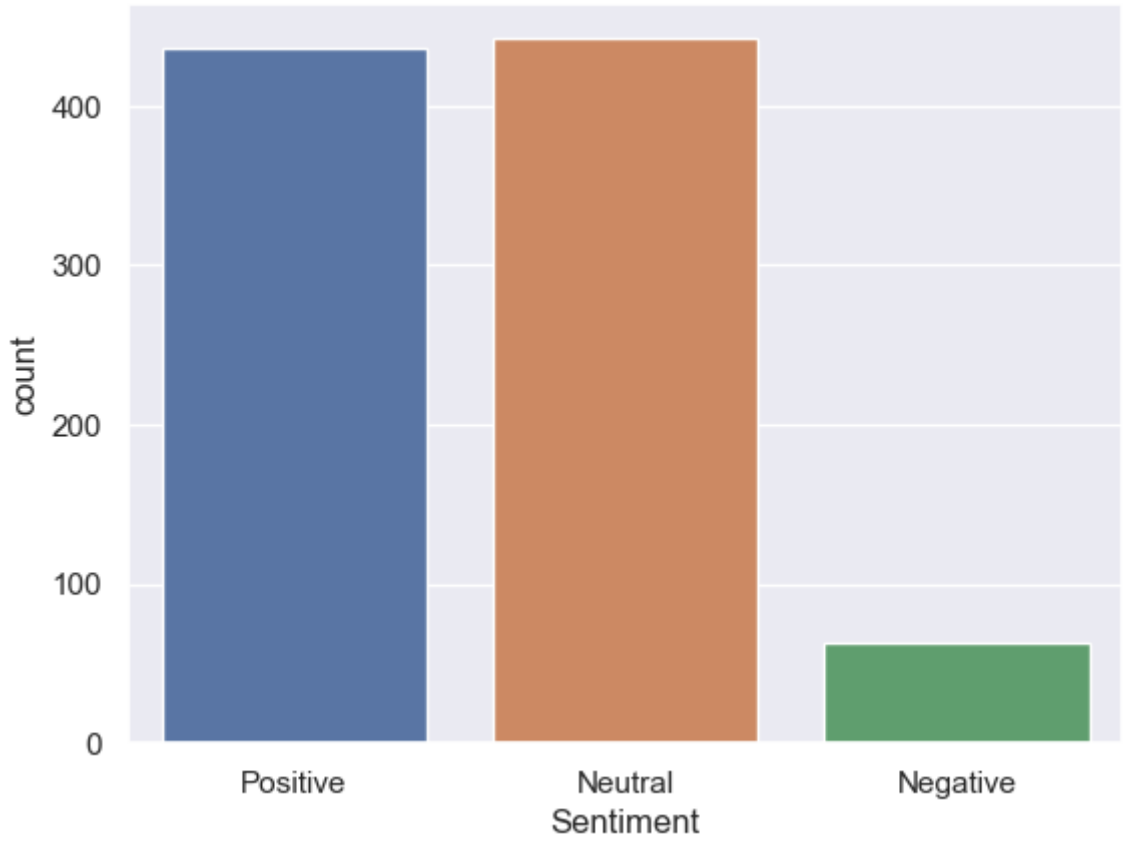
back to top

In [126…
```
release_df['Tweet_Content'] = release_df.Tweet_Content.apply(text_preproc
release_df['Tweet_Content'] = clean_column(release_df['Tweet_Content'])
release_df['Tweet_Content'] = release_df.Tweet_Content.apply(emoji_conver
release_df['Sentiment'] = release_df['Tweet_Content'].apply(get_sentiment
```

In [127…
```
release_df['Sentiment'].value_counts()
```

Out[127]:
```
Neutral     442
Positive    436
Negative     63
Name: Sentiment, dtype: int64
```

In [128…
```
plot_sentiment_distribution(release_df, 'Sentiment')
```



All Tweets

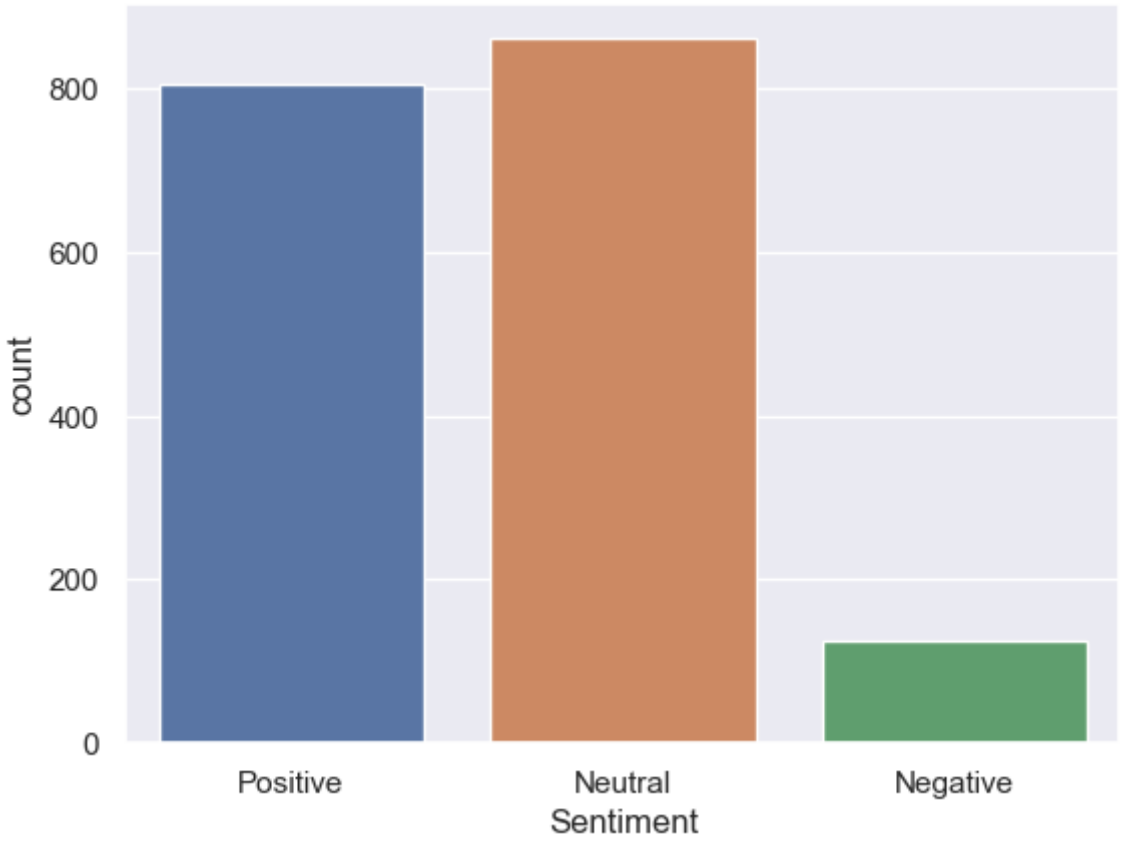
back to top

In [129…
```
all_df['Tweet_Content'] = all_df.Tweet_Content.apply(text_preproc)
all_df['Tweet_Content'] = clean_column(all_df['Tweet_Content'])
all_df['Tweet_Content'] = all_df.Tweet_Content.apply(emoji_converter)
all_df['Sentiment'] = all_df['Tweet_Content'].apply(get_sentiment_label)
```

In [130…
```
all_df['Sentiment'].value_counts()
```

Out[130]:
```
Neutral     861
Positive    805
Negative    125
Name: Sentiment, dtype: int64
```

In [131…
```
plot_sentiment_distribution(all_df, 'Sentiment')
```


800
600
400
200
0
count
Positive
Neutral
Negative
Sentiment