Imports

In [1]:
```
import pandas as pd
import json
from typing import List
from os import listdir
import spotipy
import spotipy.util as util
import plotly.express as px
import plotly.io as pio
from spotipy.oauth2 import SpotifyClientCredentials
```

Getting the Data

To retrieve my Spotify data, I navigated to the privacy settings on Spotify's website at https://www.spotify.com/ (https://www.spotify.com/). I initiated the data request process, and it took approximately 15 days for me to receive a zip file containing my data via email.

In [2]:
```
def get_streamings(path: str = 'MyData') -> List[dict]:
    """
    Retrieves Spotify streaming history from the specified directory.

    Parameters:
        path (str): The directory path where Spotify streaming history data files

    Returns:
        List[dict]: A list of dictionaries containing streaming history data.
    """

    # List all files in the specified directory with the correct naming scheme
    files = [f'{path}/{x}' for x in listdir(path) if x.startswith('Streaming_Histo

    all_streamings = []

    for file in files:
        # Open and read the file's content
        with open(file, 'r', encoding='UTF-8') as f:
            # Load JSON content from the file
            new_streamings = json.load(f)
            all_streamings += new_streamings

    return all_streamings

# dir path in my machine
spotify_data_directory = '/Users/adrianrodriguez/Downloads/MyData 3/'

# Call the function to retrieve streaming history data
streaming_data = get_streamings(spotify_data_directory)
```

Converted 'streaming_data' into DataFrame and used the 'ts' (timestamp) column to create 'year' and 'month' columns.

In [3]:
```
# Loads streaming data into a Pandas DataFrame
streaming_data = pd.DataFrame(streaming_data)

# Converts the 'ts' (timestamp) column to a datetime format
streaming_data['ts'] = pd.to_datetime(streaming_data['ts'])

# Creates a new column 'year' to extract and store the year from the 'ts' column
streaming_data['year'] = streaming_data['ts'].dt.year

# Creates a new column 'month' to extract and store the month as a string
# using the strftime method to format it to the full month name
streaming_data['month'] = streaming_data['ts'].dt.strftime('%B')

# Creates DataFrame
streaming_data = pd.DataFrame(streaming_data)

# Drop the unwanted columns
streaming_data.drop(columns=['ip_addr_decrypted', 'username'], inplace=True)

# Displays the updated DataFrame with the new 'year' and 'month' columns and remov
streaming_data
```

Out[3]:

ılbum_name	spotify_track_uri	episode_name	...	spotify_episode_uri	reason_start	reason_
ʻst Nightmare	spotify:track:7e8utCy2JlSB8dRHKi49xM	None	...	None	trackdone	track
ʻst Nightmare	spotify:track:7ABWRukVQcXrlrDKDx5Gek	None	...	None	trackdone	track
ʻst Nightmare	spotify:track:13NCxLOlvQ4Tnexgfp03Gs	None	...	None	trackdone	track
ʻst Nightmare	spotify:track:7gPd55hW5pVjTm3H9S1Wbv	None	...	None	trackdone	track
ʻst Nightmare	spotify:track:0ZUXj43fteJjwvGoLMntte	None	...	None	trackdone	track
...	...	...	...	...	...	
City Club	spotify:track:4WmkSPSFkXMpWrzPJOiz1H	None	...	None	fwdbtn	fw
City Club	spotify:track:2iCzZEPF8GTfQ1v4WCi7De	None	...	None	fwdbtn	fw
City Club	spotify:track:3UL1VvlrSYjTKbpgGAyyUx	None	...	None	fwdbtn	fw
City Club	spotify:track:63KqjQQ4ofFpQkmPXKTyEX	None	...	None	fwdbtn	fw
City Club	spotify:track:0EHGQN0jJ0oDtDGDTKWCBm	None	...	None	fwdbtn	fw

Data Cleaning: streaming_data:

'streaming_data' offers an extensive log of each individual streaming session, including session durations initially measured in milliseconds. To enhance clarity, these durations will be converted into minutes.

In [4]:
```
# Calculates and adds a new column "minutes_played" by converting "ms_played" from
streaming_data['minutes_played'] = streaming_data['ms_played'] / (1000 * 60)

# Prints the updated DataFrame, which includes the original "ms_played" and the ne
print(streaming_data[['ms_played', 'minutes_played']])
```

```
        ms_played  minutes_played
0          173493        2.891550
1          181480        3.024667
2          205240        3.420667
3          189680        3.161333
4          274200        4.570000
...           ...             ...
299339      2600        0.043333
299340    140341        2.339017
299341      3436        0.057267
299342     69022        1.150367
299343    138251        2.304183

[299344 rows x 2 columns]
```

In [5]:
```
# Sorts the DataFrame from greatest to lowest based on 'minutes_played'
streaming_data_sorted = streaming_data.sort_values(by='minutes_played', ascending=

# Displays the sorted DataFrame
streaming_data_sorted
```

Out[5]:

ıntry	user_agent_decrypted	master_metadata_track_name	master_metadata_album_artist_name	master_metadata_
US	unknown	White Noise 3 Hour Long	Erik Eriksson	Whi
US	unknown	White Noise 3 Hour Long	Erik Eriksson	Whi
US	unknown	White Noise 2 Hour Long	Erik Eriksson	Whi
US	unknown	White Noise 3 Hour Long	Erik Eriksson	Whi
US	unknown	None	None	
...	...	...	...	
US	unknown	I Think I Smell A Rat	The White Stripes	
US	unknown	Pretend I'm Gay	The Growlers	
US	unknown	delicate creature	iogi	
US	unknown	I'm In The Band	The Hellacopters	
US	unknown	I'm Not Making out With You	Surf Curse	

Most Played Tracks:

When sorted in descending order, the tracks with the highest 'minutes_played' values include 'White Noise 3 Hour Long.' It's essential to focus on actual songs and exclude sound machines.

Filtering out tracks with durations exceeding 20 minutes is crucial in limiting the analysis to songs. This approach helps identify the songs that were listened to the most.

In [6]:
```
import pandas as pd

# Creates a new DataFrame, songs_df, by filtering rows where the song duration are
# This helps narrow down the analysis to shorter songs for further examination.

songs_df = streaming_data[streaming_data['minutes_played'] <= 20]

# Print the updated DataFrame
songs_df
```

Out[6]:

	ts	platform	ms_played	conn_country	user_agent_decrypted	master_metadata_track_na
0	2020-02-04 01:52:32+00:00	iOS 13.3 (iPhone11,8)	173493	US	unknown	Fluorescent Adolesc
1	2020-02-04 01:55:37+00:00	iOS 13.3 (iPhone11,8)	181480	US	unknown	Only Ones Who Kn
2	2020-02-04 01:59:02+00:00	iOS 13.3 (iPhone11,8)	205240	US	unknown	Do Me a Fav
3	2020-02-04 02:02:13+00:00	iOS 13.3 (iPhone11,8)	189680	US	unknown	This House Is a Circ
4	2020-02-04 02:06:56+00:00	iOS 13.3 (iPhone11,8)	274200	US	unknown	If You Were There, Bew
...	...	...	...	...	...	
299339	2018-04-18 21:14:55+00:00	iOS 11.3 (iPhone8,2)	2600	US	unknown	Rubber & Bo
299340	2018-04-18	iOS 11.3	140341	US	unknown	The Daisy Ch

Data Aggregation:

Summarizes the cumulative minutes played for individual songs while preserving the initial data set for each column group and adding a 'month' and 'year' column.

In [7]:
```
import pandas as pd

# Removed rows where 'master_metadata_track_name' was equal to 'None'
songs_df = songs_df[songs_df['master_metadata_track_name'] != 'None']

# Grouped the data by 'master_metadata_track_name' and calculated the sum of 'minu
grouped_df = songs_df.groupby('master_metadata_track_name').agg({
    'minutes_played': 'sum',  # Sum of minutes_played
    'ts': 'count',  # Count of rows (number of plays)
    'platform': 'first',  # Kept the first platform in the group
    'conn_country': 'first',  # Kept the first conn_country in the group
    'user_agent_decrypted': 'first',  # Kept the first user_agent_decrypted in the
    'master_metadata_album_artist_name': 'first',  # Kept the first artist name in
    'master_metadata_album_album_name': 'first',  # Kept the first album name in t
    'spotify_track_uri': 'first',  # Kept the first track URI in the group
}).reset_index()

# Sorted the grouped DataFrame by 'minutes_played' in descending order
songs_grouped_df = grouped_df.sort_values(by='minutes_played', ascending=False)

# Convert 'ts' column to datetime format
songs_grouped_df['ts'] = pd.to_datetime(streaming_data['ts'])

# Create 'year' column
songs_grouped_df['year'] = streaming_data['ts'].dt.year

# Create 'month' column using strftime
songs_grouped_df['month'] = streaming_data['ts'].dt.strftime('%B')

# Display the updated DataFrame
songs_grouped_df
```

Out[7]:

	master_metadata_track_name	minutes_played	ts	platform	conn_country	user_agent_dec
24773	Star Treatment	1483.573367	2021-04-29 22:44:32+00:00	iOS 13.3 (iPhone11,8)	US	ur
12651	I'll Come Too	1482.494117	2020-04-25 06:15:19+00:00	iOS 13.3.1 (iPhone11,8)	US	ur
24281	Someday	1176.526867	2021-04-27 21:53:35+00:00	iOS 13.3.1 (iPhone11,8)	US	ur
2589	Batphone	963.282333	2020-02-28 22:30:08+00:00	iOS 13.3.1 (iPhone11,8)	US	ur
14947	Leave It In My Dreams	949.748350	2020-05-08 22:30:26+00:00	iOS 13.3 (iPhone11,8)	US	ur
...	...	...	...	...	...	
4637	Changing	0.000000	2020-03-10 03:14:01+00:00	iOS 14.6 (iPhone11,8)	US	ur
12638	I'll Be Somewhere	0.000000	2020-04-25 06:07:40+00:00	ios	US	ur
1168	All About That Bass	0.000000	2020-02-14 22:55:42+00:00	iOS 14.6 (iPhone11,8)	US	ur
27736	Three Times A Lady	0.000000	2021-05-24 17:42:59+00:00	ios	US	ur
3407	Blueberry Hill	0.000000	2020-03-04 21:21:47+00:00	iOS 14.2 (iPhone11,8)	US	ur

32279 rows × 11 columns

Top Songs Played:

In [8]:
```
# Rank the rows based on the 'count' column
songs_grouped_df['rank'] = songs_grouped_df['minutes_played'].rank(ascending=False

# Reorder columns to make 'rank' the first column
column_order = ['rank'] + [col for col in songs_grouped_df.columns if col != 'rank
songs_grouped_df = songs_grouped_df[column_order]

# Sort the DataFrame based on the rank
songs_grouped_df = songs_grouped_df.sort_values(by='rank')
songs_grouped_df
```

Out[8]:

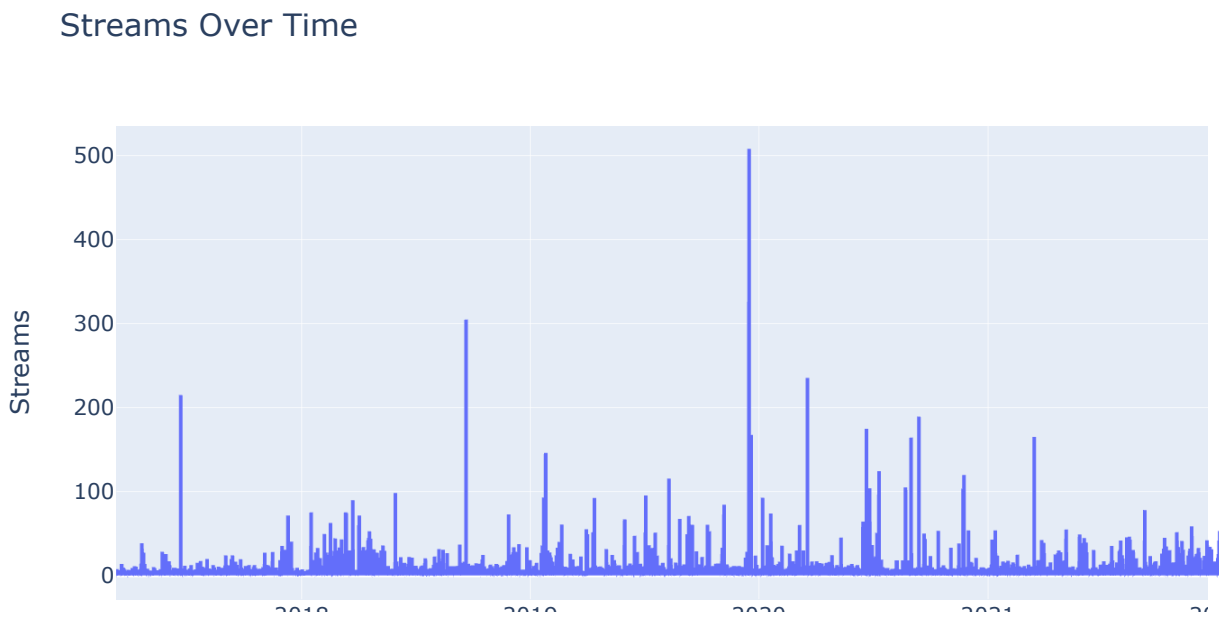
	rank	master_metadata_track_name	minutes_played	ts	platform	conn_country	user_a
24773	1.0	Star Treatment	1483.573367	2021-04-29 22:44:32+00:00	iOS 13.3 (iPhone11,8)	US	
12651	2.0	I'll Come Too	1482.494117	2020-04-25 06:15:19+00:00	iOS 13.3.1 (iPhone11,8)	US	
24281	3.0	Someday	1176.526867	2021-04-27 21:53:35+00:00	iOS 13.3.1 (iPhone11,8)	US	
2589	4.0	Batphone	963.282333	2020-02-28 22:30:08+00:00	iOS 13.3.1 (iPhone11,8)	US	
14947	5.0	Leave It In My Dreams	949.748350	2020-05-08 22:30:26+00:00	iOS 13.3 (iPhone11,8)	US	
...	...	...	...	...	...	...	
5217	31395.0	Come Sail Away	0.000000	2020-03-12 15:38:43+00:00	iOS 14.8.1 (iPhone11,8)	US	
13951	31395.0	Joseph, Better You Than Me	0.000000	2020-05-01 16:58:56+00:00	OS X 10.12.5 [x86 8]	US	
6896	31395.0	Don't Carry It All	0.000000	2020-03-21 23:45:46+00:00	iOS 14.6 (iPhone11,8)	US	
25593	31395.0	Swamp Fruit	0.000000	2021-05-03 18:01:40+00:00	iOS 12.3.1 (iPhone11,8)	US	
3407	31395.0	Blueberry Hill	0.000000	2020-03-04 21:21:47+00:00	iOS 14.2 (iPhone11,8)	US	

32279 rows × 12 columns

Vizualizations:

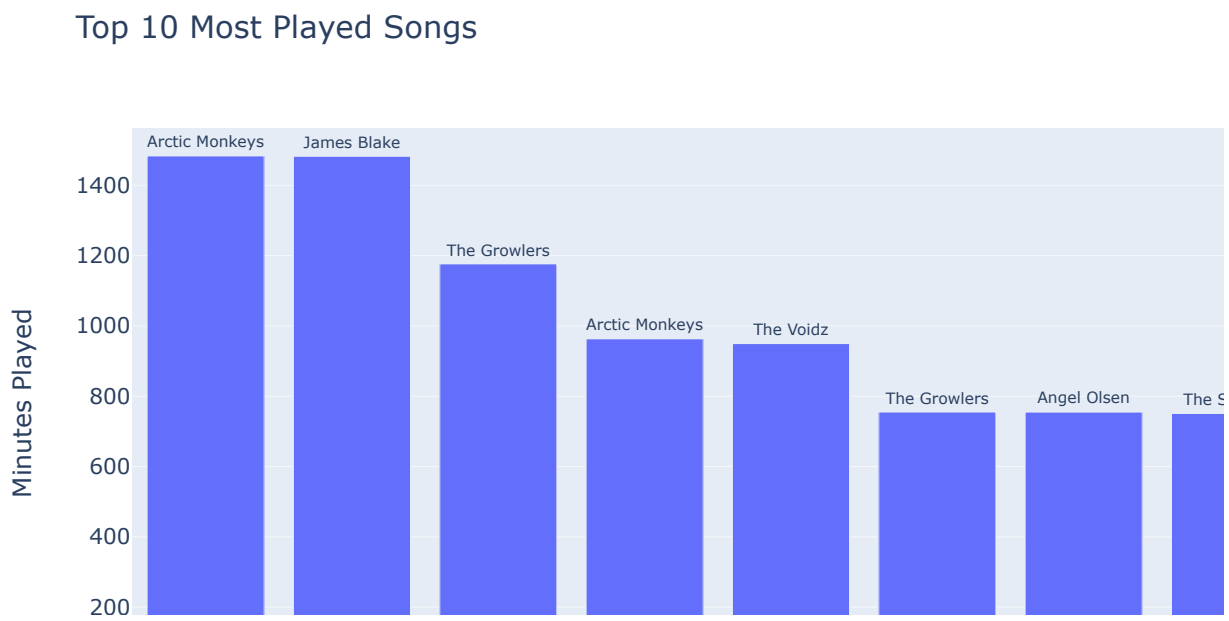
Streams Over Time

In [9]:
```
pio.renderers.default = 'notebook'

# Assuming your DataFrame is named streaming_data
# Convert the 'ts' column to a datetime object
streaming_data['ts'] = pd.to_datetime(streaming_data['ts'])

# Group the data by 'ts' and sum the 'minutes_played' for each timestamp
streams_over_time = streaming_data.groupby('ts')['minutes_played'].sum().reset_ind

# Create an interactive line plot for streams over time with Plotly
fig = px.line(streams_over_time, x='ts', y='minutes_played', title='Streams Over T
fig.update_xaxes(title_text='Date')
fig.update_yaxes(title_text='Streams')

# Enable zooming and panning
fig.update_xaxes(rangeslider_visible=True)

# Show the plot
fig.show()
```



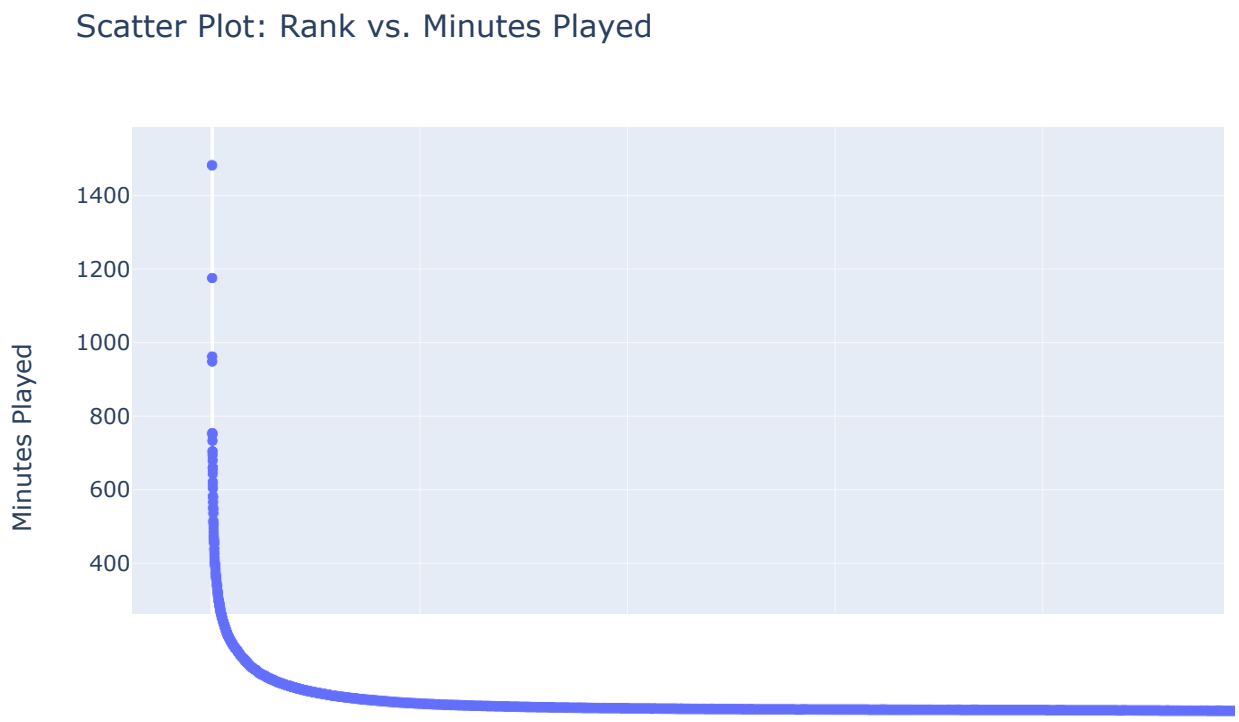
Top Played Songs

In [10]:
```
# Assuming songs_grouped_df is your actual DataFrame with the columns mentioned
# Sort the DataFrame by 'minutes_played' to get the most played songs and limit to
top_songs = songs_grouped_df.sort_values('minutes_played', ascending=False).head(1

# Create the bar chart using Plotly Express
fig = px.bar(top_songs,
             x='master_metadata_track_name',
             y='minutes_played',
             text='master_metadata_album_artist_name',
             title='Top 10 Most Played Songs')

# Customize the layout to include the artist names in the bar chart
fig.update_traces(texttemplate='%{text}', textposition='outside')
fig.update_layout(uniformtext_minsize=8, uniformtext_mode='hide',
                  xaxis_title='Song Name',
                  yaxis_title='Minutes Played')

# Show the figure
fig.show()
```



Showing Plot Between 'rank' and 'minutes_played'

In [11]:
```
# Creates an interactive scatter plot between 'rank' and 'minutes_played' using Pl
fig = px.scatter(
    songs_grouped_df,
    x='rank',
    y='minutes_played',
    title='Scatter Plot: Rank vs. Minutes Played',
    hover_name='master_metadata_track_name',  # Track name shown on hover
    hover_data=['master_metadata_album_artist_name'],  # Artist name shown on hove
)

fig.update_xaxes(title_text='Rank')
fig.update_yaxes(title_text='Minutes Played')

# Enable zooming and panning
fig.update_xaxes(type='linear')
fig.update_yaxes(type='linear')

# Show the plot
fig.show()
```



Accessing Spotify API

In [12]:
```
username = 'juliorod323'
client_id ='3d2dd78837114f7ab9c5bb16c14ee9ba'
client_secret = '706144568b26419d90d293a14904396b'
redirect_uri = 'http://localhost:7777/callback'
scope = 'user-read-recently-played'

token = util.prompt_for_user_token(username=username,
                                   scope=scope,
                                   client_id=client_id,
                                   client_secret=client_secret,
                                   redirect_uri=redirect_uri)
```

In [13]:
```
import requests

def get_id(track_name: str, token: str) -> str:
    headers = {
        'Accept': 'application/json',
        'Content-Type': 'application/json',
        'Authorization': f'Bearer {token}',
    }
    params = {
        'q': track_name,
        'type': 'track',
    }
    try:
        response = requests.get('https://api.spotify.com/v1/search', headers=heade
        data = response.json()
        first_result = data['tracks']['items'][0]
        track_id = first_result['id']
        return track_id
    except:
        return None
```

In [14]:
```
# Create a Spotify API client
client_credentials_manager = SpotifyClientCredentials(client_id=client_id, client_
sp = spotipy.Spotify(client_credentials_manager=client_credentials_manager)

def get_features(track_id: str, token: str) -> dict:
    """
    Get Spotify track audio features.

    Args:
        track_id (str): The Spotify track ID.
        token (str): Your Spotify access token.

    Returns:
        dict: Dictionary containing track audio features.
    """
    try:
        features = sp.audio_features(track_id)
        return features[0] if features else None
    except:
        return None
```

In [15]:
```
songs_grouped_df.columns
```

Out[15]:
```
Index(['rank', 'master_metadata_track_name', 'minutes_played', 'ts',
       'platform', 'conn_country', 'user_agent_decrypted',
       'master_metadata_album_artist_name', 'master_metadata_album_album_name',
       'spotify_track_uri', 'year', 'month'],
      dtype='object')
```

API Obstacles

I have run into request limitation with the Spotify API, so as a precaution, I will create the top_song_100 df that contains the top 100 most played songs. I want to see the attributes for my top 100 songs played.

In [16]:
```
# Creates a condensed version of the songs_grouped_df DataFrame
top_songs_100 = songs_grouped_df.sort_values('minutes_played', ascending=False).he
top_songs_100
```

Out[16]:

	rank	master_metadata_track_name	minutes_played	ts	platform	conn_country	user_ag
24773	1.0	Star Treatment	1483.573367	2021-04-29 22:44:32+00:00	iOS 13.3 (iPhone11,8)	US	
12651	2.0	I'll Come Too	1482.494117	2020-04-25 06:15:19+00:00	iOS 13.3.1 (iPhone11,8)	US	
24281	3.0	Someday	1176.526867	2021-04-27 21:53:35+00:00	iOS 13.3.1 (iPhone11,8)	US	
2589	4.0	Batphone	963.282333	2020-02-28 22:30:08+00:00	iOS 13.3.1 (iPhone11,8)	US	
14947	5.0	Leave It In My Dreams	949.748350	2020-05-08 22:30:26+00:00	iOS 13.3 (iPhone11,8)	US	
...	...	...	...	...	...	...	
28359	96.0	Trouble	364.247700	2021-05-31 00:16:11+00:00	iOS 13.3.1 (iPhone11,8)	US	
10079	97.0	Going Gets Tough	362.715700	2020-04-09 02:04:29+00:00	iOS 13.3.1 (iPhone11,8)	US	
3762	98.0	British Bombs	362.609383	2020-03-05 23:58:30+00:00	OS X 10.14.6 [x86 8]	US	
30484	99.0	Why'd You Only Call Me When You're High?	359.919717	2021-06-10 00:38:03+00:00	iOS 13.3.1 (iPhone11,8)	US	
18403	100.0	Nights	359.622617	2021-03-13 08:41:57+00:00	iOS 13.3.1 (iPhone11,8)	US	

100 rows × 12 columns

In [17]:

```
import pandas as pd

# Defines a list to store the audio features for each track
audio_features_list = []

# Loops through the 'master_metadata_track_name' column
for track_name in top_songs_100['master_metadata_track_name']:
    # Get the Spotify track ID for the current track
    track_id = get_id(track_name, token)

    # If a track ID is found, retrieves audio features
    if track_id:
        track_features = get_features(track_id, token)

        # If audio features are found, add them to the list
        if track_features:
            audio_features_list.append(track_features)

# Creates a DataFrame from the list of audio features
audio_features_df = pd.DataFrame(audio_features_list)

# Displays the resulting DataFrame with audio features
print(audio_features_df)
```

```
    danceability  energy  key  loudness  mode  speechiness  acousticness  \
0          0.581   0.767    7    -5.026     0       0.0527       0.24300
1          0.661   0.472    7    -7.889     1       0.1640       0.75700
2          0.539   0.521    9    -7.460     1       0.0299       0.00425
3          0.675   0.681    5    -5.728     0       0.0356       0.31200
4          0.600   0.746    4    -4.696     0       0.0355       0.02260
..           ...     ...  ...       ...   ...          ...           ...
95         0.470   0.623    0    -5.655     1       0.0302       0.39200
96         0.501   0.769    8    -6.257     1       0.0326       0.22500
97         0.674   0.865    6    -6.167     1       0.0412       0.04620
98         0.691   0.631    2    -6.478     1       0.0368       0.04830
99         0.457   0.551    5    -9.360     0       0.1670       0.42700

    instrumentalness  liveness  valence    tempo            type  \
0           0.001310    0.1410    0.673  120.070  audio_features
1           0.002750    0.1920    0.206   73.986  audio_features
2           0.000000    0.3670    0.679  106.035  audio_features
3           0.000000    0.3040    0.675  106.521  audio_features
4           0.011900    0.1090    0.614  118.551  audio_features
..               ...       ...      ...      ...             ...
95          0.000439    0.0992    0.298   77.861  audio_features
96          0.000000    0.1090    0.737  162.047  audio_features
97          0.000000    0.3810    0.694  131.975  audio_features
98          0.000011    0.1040    0.800   92.004  audio_features
99          0.000001    0.1130    0.428   89.870  audio_features

                        id                                  uri  \
0   0b93tWwuoAC0nXe1CfR30I  spotify:track:0b93tWwuoAC0nXe1CfR30I
1   6EvUAsfncetT0RcWshHCbl  spotify:track:6EvUAsfncetT0RcWshHCbl
2   7hm4HTk9encxT0LYC0J6oI  spotify:track:7hm4HTk9encxT0LYC0J6oI
3   7aiKdAM9WYW3GzWSA9OXIl  spotify:track:7aiKdAM9WYW3GzWSA9OXIl
4   31u6rUeIEXGrYVoh10U7eu  spotify:track:31u6rUeIEXGrYVoh10U7eu
..                     ...                                 ...
95  5n0CTysih20NYdT2S0Wpe8  spotify:track:5n0CTysih20NYdT2S0Wpe8
96  6Et9MYUuK2OkYE3ce2F0oK  spotify:track:6Et9MYUuK2OkYE3ce2F0oK
97  4ZdmTNaBTErD8n9AQE0YaX  spotify:track:4ZdmTNaBTErD8n9AQE0YaX
98  086myS9r57YsLbJpU0TgK9  spotify:track:086myS9r57YsLbJpU0TgK9
99  7eqoqGkKwgOaWNNHx90uEZ  spotify:track:7eqoqGkKwgOaWNNHx90uEZ

                                           track_href  \
0   https://api.spotify.com/v1/tracks/0b93tWwuoAC0... (https://api.spotify.com/v
1/tracks/0b93tWwuoAC0...)
1   https://api.spotify.com/v1/tracks/6EvUAsfncetT... (https://api.spotify.com/v
1/tracks/6EvUAsfncetT...)
2   https://api.spotify.com/v1/tracks/7hm4HTk9encx... (https://api.spotify.com/v
1/tracks/7hm4HTk9encx...)
3   https://api.spotify.com/v1/tracks/7aiKdAM9WYW3... (https://api.spotify.com/v
1/tracks/7aiKdAM9WYW3...)
4   https://api.spotify.com/v1/tracks/31u6rUeIEXGr... (https://api.spotify.com/v
1/tracks/31u6rUeIEXGr...)
..                                                ...
95  https://api.spotify.com/v1/tracks/5n0CTysih20N... (https://api.spotify.com/v
1/tracks/5n0CTysih20N...)
96  https://api.spotify.com/v1/tracks/6Et9MYUuK2Ok... (https://api.spotify.com/v
1/tracks/6Et9MYUuK2Ok...)
97  https://api.spotify.com/v1/tracks/4ZdmTNaBTErD... (https://api.spotify.com/v
1/tracks/4ZdmTNaBTErD...)
98  https://api.spotify.com/v1/tracks/086myS9r57Ys... (https://api.spotify.com/v
1/tracks/086myS9r57Ys...)
99  https://api.spotify.com/v1/tracks/7eqoqGkKwgOa... (https://api.spotify.com/v
1/tracks/7eqoqGkKwgOa...)

                                        analysis_url  duration_ms  \
```

```
0   https://api.spotify.com/v1/audio-analysis/0b93... (https://api.spotify.com/v
1/audio-analysis/0b93...)       354640
1   https://api.spotify.com/v1/audio-analysis/6EvU... (https://api.spotify.com/v
1/audio-analysis/6EvU...)       222400
2   https://api.spotify.com/v1/audio-analysis/7hm4... (https://api.spotify.com/v
1/audio-analysis/7hm4...)       183440
3   https://api.spotify.com/v1/audio-analysis/7aiK... (https://api.spotify.com/v
1/audio-analysis/7aiK...)       271613
4   https://api.spotify.com/v1/audio-analysis/31u6... (https://api.spotify.com/v
1/audio-analysis/31u6...)       239442
..                                                ...          ...
95  https://api.spotify.com/v1/audio-analysis/5n0C... (https://api.spotify.com/v
1/audio-analysis/5n0C...)       225973
96  https://api.spotify.com/v1/audio-analysis/6Et9... (https://api.spotify.com/v
1/audio-analysis/6Et9...)       208613
97  https://api.spotify.com/v1/audio-analysis/4Zdm... (https://api.spotify.com/v
1/audio-analysis/4Zdm...)       297715
98  https://api.spotify.com/v1/audio-analysis/086m... (https://api.spotify.com/v
1/audio-analysis/086m...)       161124
99  https://api.spotify.com/v1/audio-analysis/7eqo... (https://api.spotify.com/v
1/audio-analysis/7eqo...)       307151

    time_signature
0                4
1                4
2                4
3                4
4                4
..             ...
95               4
96               4
97               4
98               4
99               4

[100 rows x 18 columns]
```

Limiting "Features" in Analysis:

In [18]:
```
# Columns to drop
columns_to_drop = ['mode', 'valence', 'analysis_url', 'time_signature', 'uri', 'tr

# Drop the specified columns from the DataFrame, ignoring columns that do not exis
audio_features_df = audio_features_df.drop(columns=columns_to_drop, errors='ignore

# Display the resulting DataFrame
print(audio_features_df)
```

```
    danceability  energy  loudness  speechiness  acousticness  \
0          0.581   0.767    -5.026       0.0527       0.24300
1          0.661   0.472    -7.889       0.1640       0.75700
2          0.539   0.521    -7.460       0.0299       0.00425
3          0.675   0.681    -5.728       0.0356       0.31200
4          0.600   0.746    -4.696       0.0355       0.02260
..           ...     ...       ...          ...           ...
95         0.470   0.623    -5.655       0.0302       0.39200
96         0.501   0.769    -6.257       0.0326       0.22500
97         0.674   0.865    -6.167       0.0412       0.04620
98         0.691   0.631    -6.478       0.0368       0.04830
99         0.457   0.551    -9.360       0.1670       0.42700

    instrumentalness  liveness    tempo  duration_ms
0           0.001310    0.1410  120.070       354640
1           0.002750    0.1920   73.986       222400
2           0.000000    0.3670  106.035       183440
3           0.000000    0.3040  106.521       271613
4           0.011900    0.1090  118.551       239442
..               ...       ...      ...          ...
95          0.000439    0.0992   77.861       225973
96          0.000000    0.1090  162.047       208613
97          0.000000    0.3810  131.975       297715
98          0.000011    0.1040   92.004       161124
99          0.000001    0.1130   89.870       307151

[100 rows x 9 columns]
```

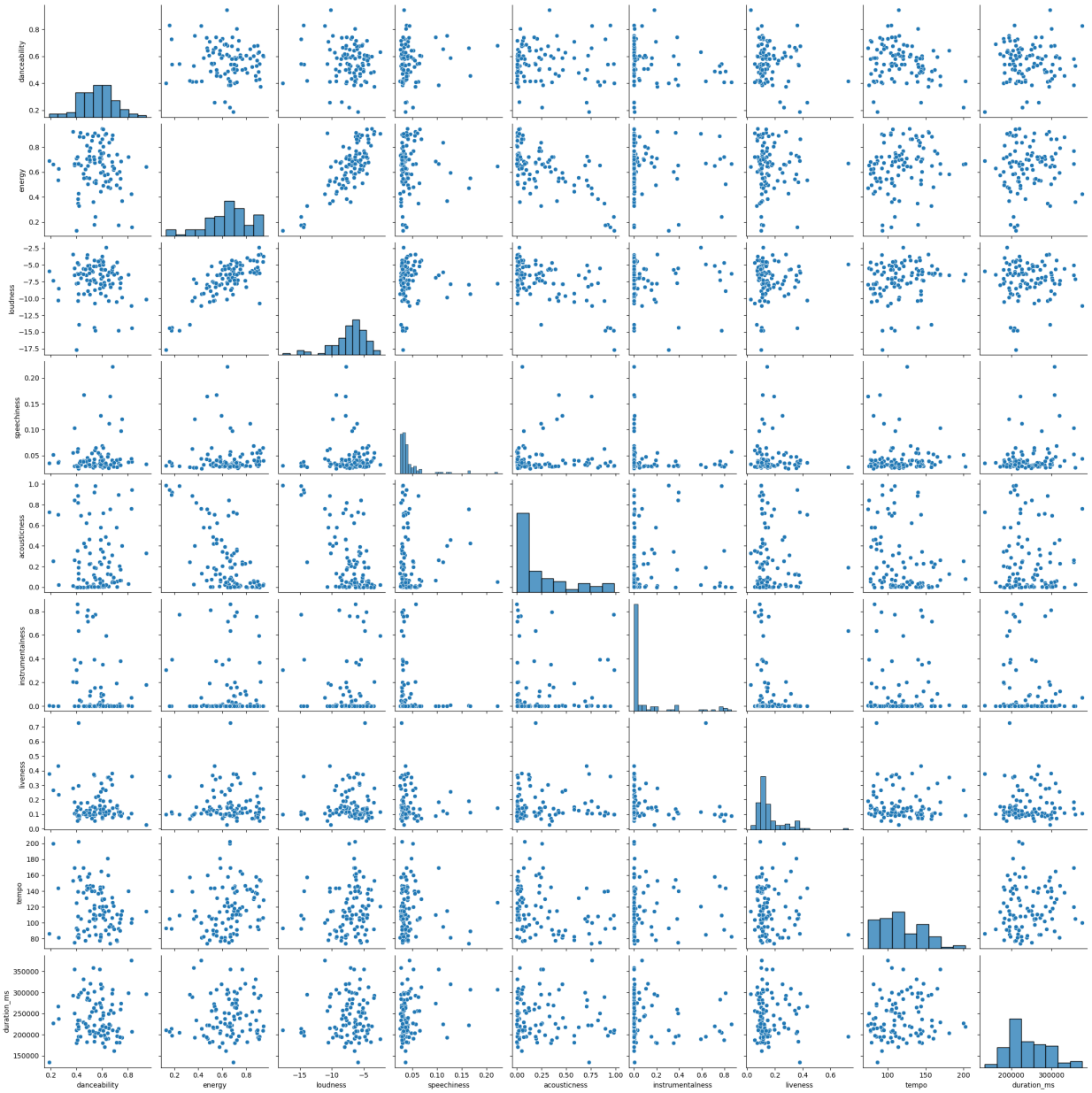
In [19]:
```
audio_features_df.columns
```

Out[19]:
```
Index(['danceability', 'energy', 'loudness', 'speechiness', 'acousticness',
       'instrumentalness', 'liveness', 'tempo', 'duration_ms'],
      dtype='object')
```

Pair Plot

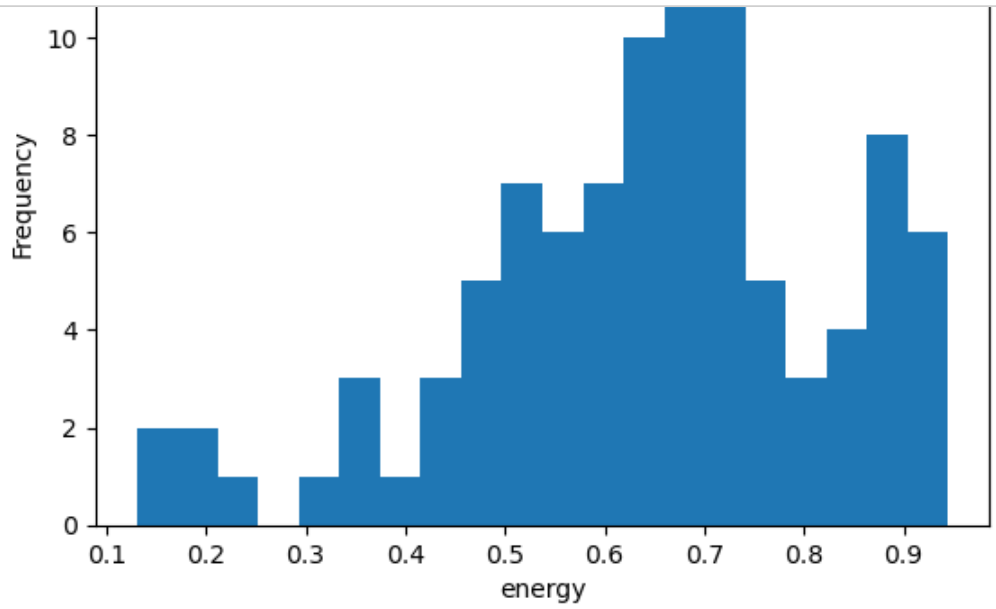
In [20]:
```
import seaborn as sns
sns.pairplot(audio_features_df)
```

Out[20]:
```
<seaborn.axisgrid.PairGrid at 0x17776f410>
```



Spotify Track "Features" Plotted Along Frequency

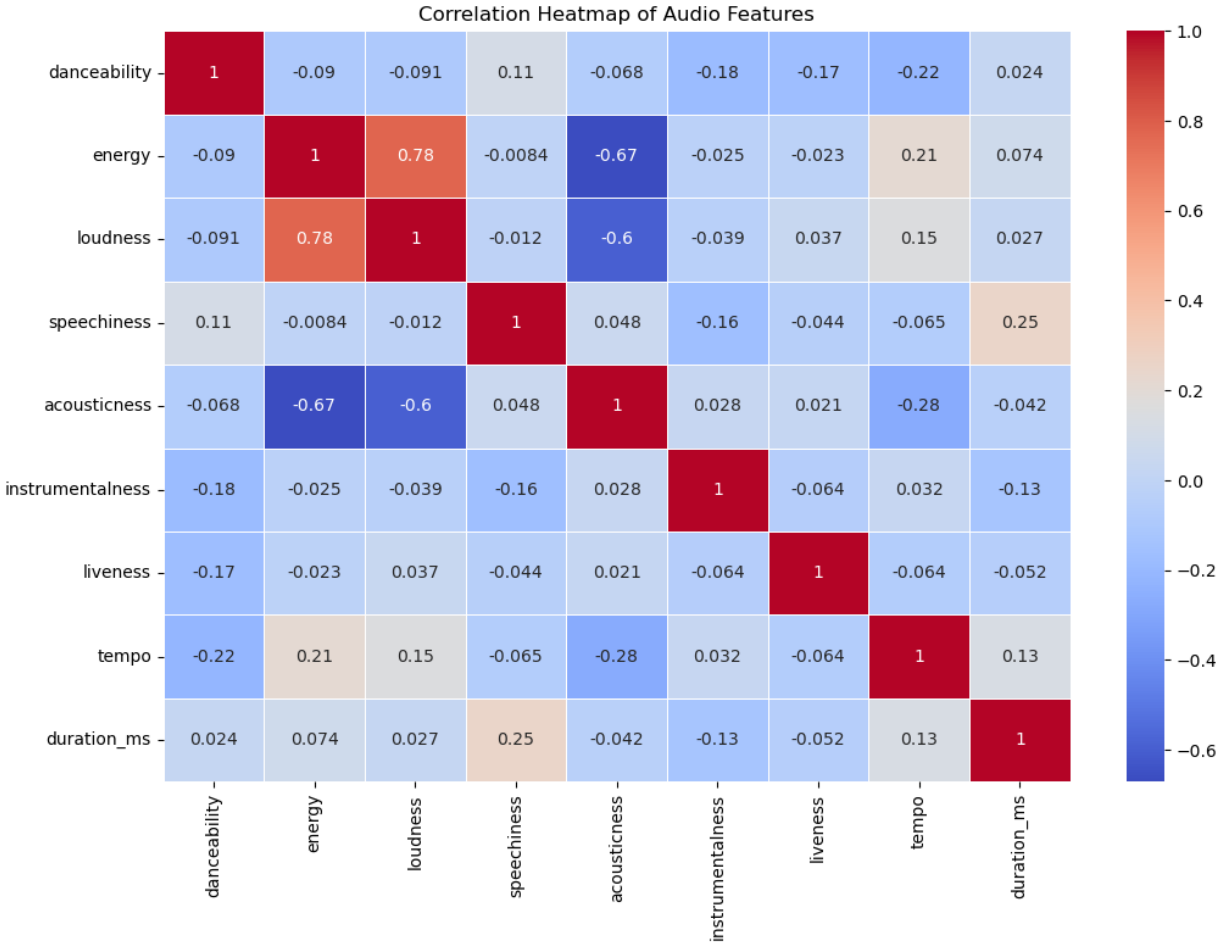
In [21]:
```
import matplotlib.pyplot as plt
for feature in audio_features_df.columns:
    plt.hist(audio_features_df[feature], bins=20)
    plt.xlabel(feature)
    plt.ylabel('Frequency')
    plt.title(f'Distribution of {feature}')
    plt.show()
```



Coorelation Heatmap:

In [22]:
```
import seaborn as sns
import matplotlib.pyplot as plt
corr_matrix = audio_features_df.corr()
plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=.5)
plt.title('Correlation Heatmap of Audio Features')
```

Out[22]: Text(0.5, 1.0, 'Correlation Heatmap of Audio Features')



In [ ]: