Shaun's Spotify Recommender System

By Shaun McKellar Jr

In [1]:
```
import spotipy
from spotipy.oauth2 import SpotifyOAuth
import pandas as pd
import requests
import matplotlib.pyplot as plt
import seaborn as sns
from requests.exceptions import RequestException
from requests.exceptions import HTTPError
from spotipy.exceptions import SpotifyException
import time
import concurrent.futures
```

Set up Spotify authentication

In [2]:
```
sp = spotipy.Spotify(auth_manager=SpotifyOAuth(
    client_id='939af66cdaa845e994b5e256d5cceb5e',
    client_secret='9a04e4e0ab3d498c9aa0394256954f13',
    redirect_uri='http://localhost:8889/callback',
    scope='user-library-read user-top-read playlist-modify-public'
))
```

Fetch liked songs

In [3]:
```
def fetch_all_liked_songs(sp):
    results = sp.current_user_saved_tracks(limit=50)
    liked_songs = results['items']

    while results['next']:
        results = sp.next(results)
        liked_songs.extend(results['items'])

    return liked_songs

liked_songs = fetch_all_liked_songs(sp)
```

Extract relevant features and metadata

In [4]:
```
song_data = []
artist_ids = set()  # To collect unique artist IDs
```

In [5]:
```
for item in liked_songs:
    track = item['track']
    track_info = {
        'track_id': track['id'],
        'track_name': track['name'],
        'artist_id': track['artists'][0]['id'],
        'artist_name': track['artists'][0]['name'],
        'popularity': track['popularity'],
        'album': track['album']['name'],
        'added_at': item['added_at']
    }
    artist_ids.add(track['artists'][0]['id'])
    song_data.append(track_info)
```

Convert to DataFrame

In [6]:
```
df_songs = pd.DataFrame(song_data)
print(df_songs)
```

```
                    track_id              track_name              artist_id  \
0     1fRuRNJVZjDU1yKXvarKqW            I'm Not A Star  1sBkRIssrMs1AbVkOJbc7a
1     7DuVZVdHaFQIYg14VNykXi               Free Mason  1sBkRIssrMs1AbVkOJbc7a
2     1amE1IohObbqkU9UzV8uFl             Tears Of Joy  1sBkRIssrMs1AbVkOJbc7a
3     06aiCjSnK0XXBEERhduZWZ        Maybach Music III  1sBkRIssrMs1AbVkOJbc7a
4     4L0dkLS6mpsf6zRKVSwfWY      Live Fast, Die Young  1sBkRIssrMs1AbVkOJbc7a
...                      ...                      ...                     ...
3709  3BtuIIrQlkujKPuWF2B85z                 Too Good  3TVXtAsR1Inumwj472S9r4
3710  3ppVO2tyWRRznNmONvt7Se   Summers Over Interlude  3TVXtAsR1Inumwj472S9r4
3711  4BhGTc3Cgay2U1QcTS7vQe            Fire & Desire  3TVXtAsR1Inumwj472S9r4
3712  7MjSipTto9QljYzZnloXOn                    Views  3TVXtAsR1Inumwj472S9r4
3713  0wwPcA6wtMf6HUMpIRdeP7             Hotline Bling  3TVXtAsR1Inumwj472S9r4

     artist_name  popularity       album              added_at
0      Rick Ross          47  Teflon Don  2017-03-11T01:23:15Z
1      Rick Ross          44  Teflon Don  2017-03-11T01:23:15Z
2      Rick Ross          38  Teflon Don  2017-03-11T01:23:15Z
3      Rick Ross          38  Teflon Don  2017-03-11T01:23:15Z
4      Rick Ross          39  Teflon Don  2017-03-11T01:23:15Z
...          ...         ...         ...                   ...
3709       Drake          73       Views  2016-05-23T00:30:42Z
3710       Drake          72       Views  2016-05-23T00:30:42Z
3711       Drake          68       Views  2016-05-23T00:30:42Z
3712       Drake          57       Views  2016-05-23T00:30:42Z
3713       Drake          76       Views  2016-05-23T00:30:42Z

[3714 rows x 7 columns]
```

Optimized function to fetch genres for multiple artists using parallel requests

In [7]:
```
def fetch_genres_for_artists(sp, artist_ids):
    artist_genres = {}
    with concurrent.futures.ThreadPoolExecutor() as executor:
        future_to_artist = {executor.submit(sp.artist, artist_id): artist_ic
        for future in concurrent.futures.as_completed(future_to_artist):
            artist_id = future_to_artist[future]
            try:
                artist_data = future.result()
                artist_genres[artist_id] = artist_data['genres']
            except Exception as e:
                print(f"Error fetching genres for artist {artist_id}: {e}")
    return artist_genres
```

Fetch genres for these artists

In [8]:
```
artist_genres = fetch_genres_for_artists(sp, artist_ids)
```

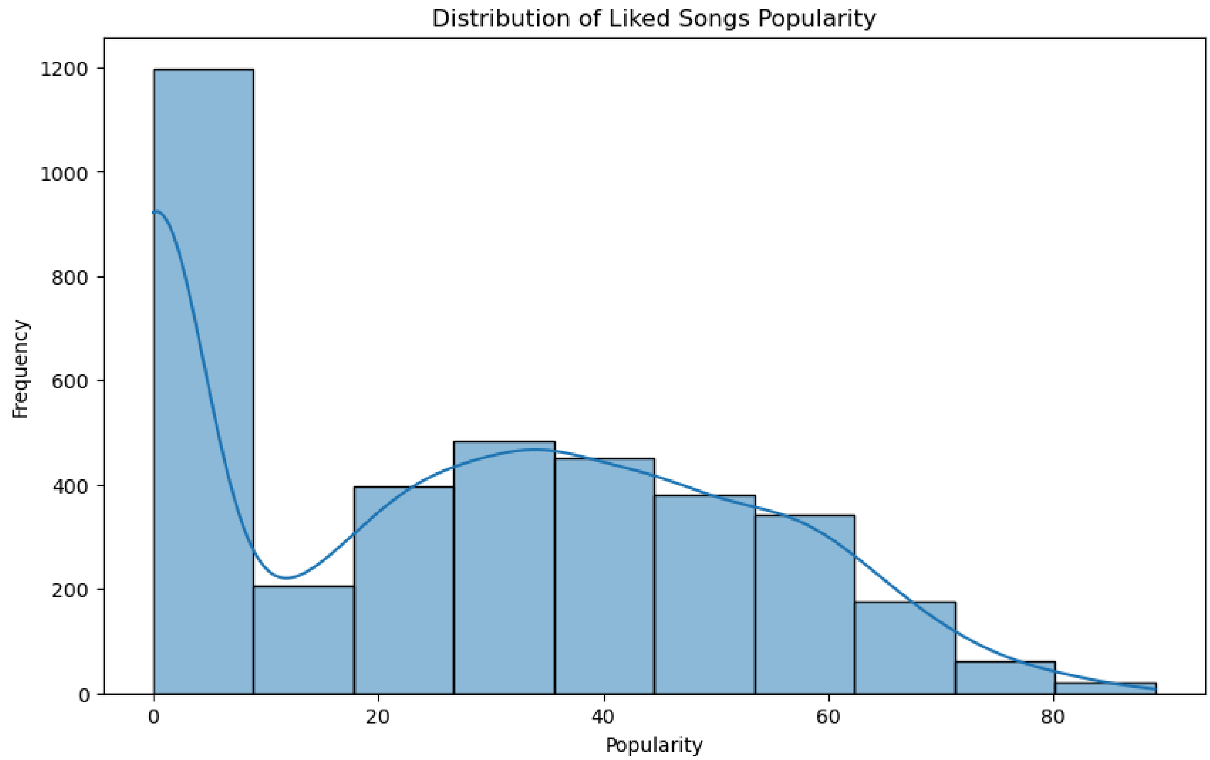
Map genres to the tracks

In [9]:
```
df_songs['genres'] = df_songs['artist_id'].map(lambda x: artist_genres.get(x
```

Plot the distribution of track popularity

In [10]:
```
plt.figure(figsize=(10, 6))
sns.histplot(df_songs['popularity'], kde=True, bins=10)
plt.title('Distribution of Liked Songs Popularity')
plt.xlabel('Popularity')
plt.ylabel('Frequency')
plt.show()
```

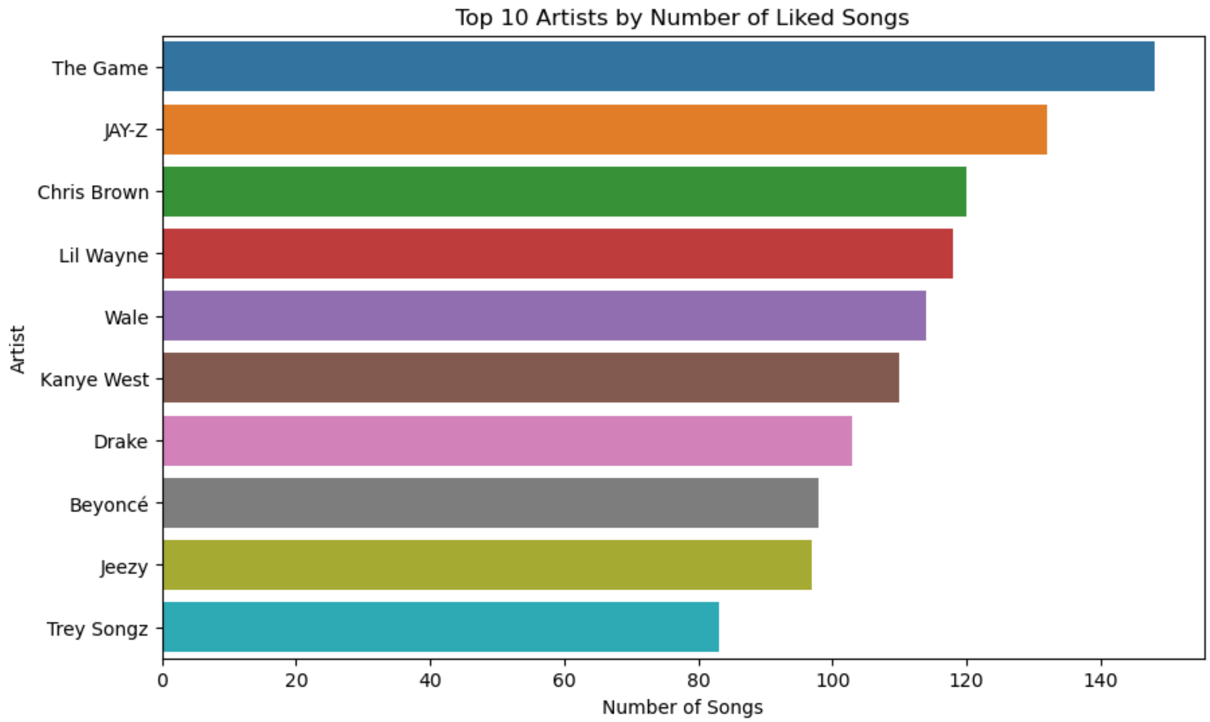
```
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: Future
Warning: use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



Top 10 artists by number of liked songs

In [11]:

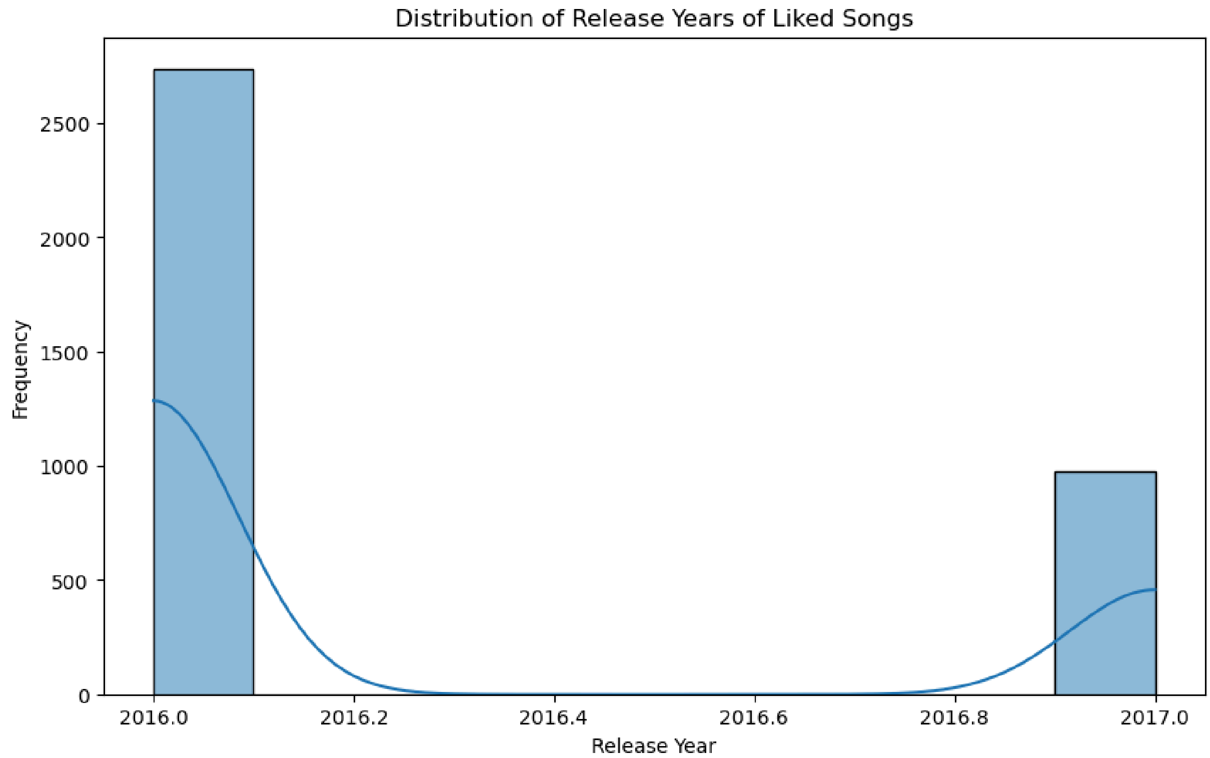
```
top_artists = df_songs['artist_name'].value_counts().head(10)
plt.figure(figsize=(10, 6))
sns.barplot(x=top_artists.values, y=top_artists.index)
plt.title('Top 10 Artists by Number of Liked Songs')
plt.xlabel('Number of Songs')
plt.ylabel('Artist')
plt.show()
```



Distribution of Release Years

In [12]:
```
df_songs['release_year'] = pd.to_datetime(df_songs['added_at']).dt.year
plt.figure(figsize=(10, 6))
sns.histplot(df_songs['release_year'], kde=True, bins=10)
plt.title('Distribution of Release Years of Liked Songs')
plt.xlabel('Release Year')
plt.ylabel('Frequency')
plt.show()
```

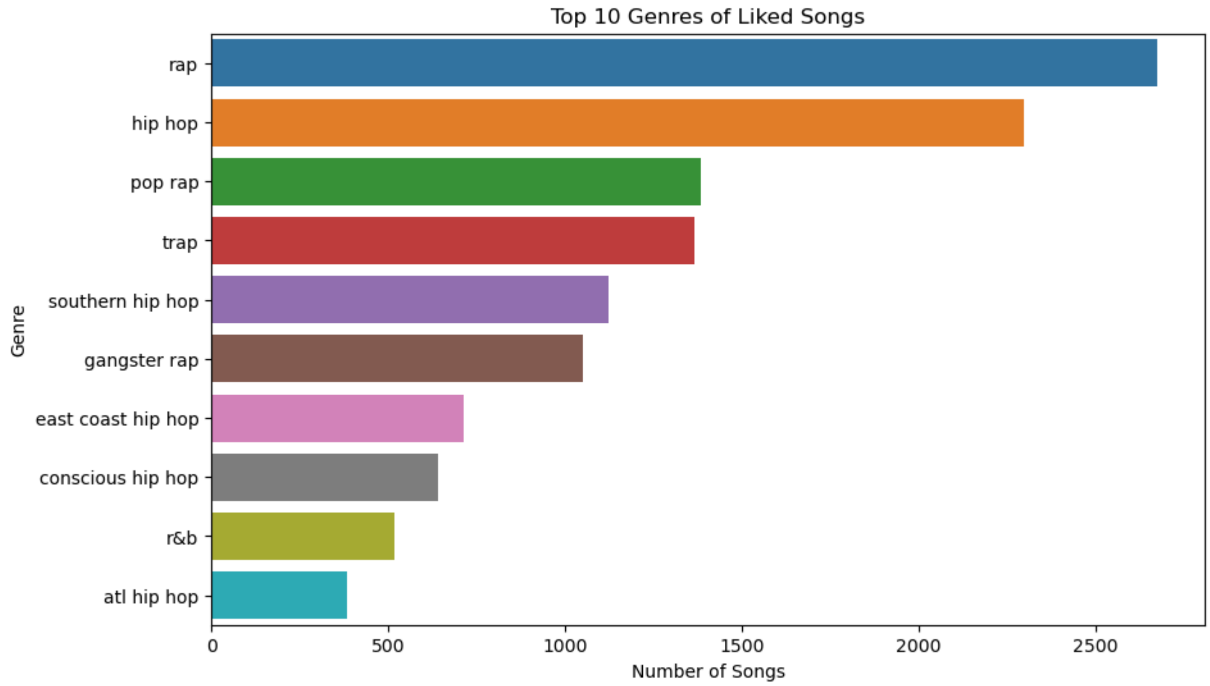
```
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: Future
Warning: use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



Distribution of Genres

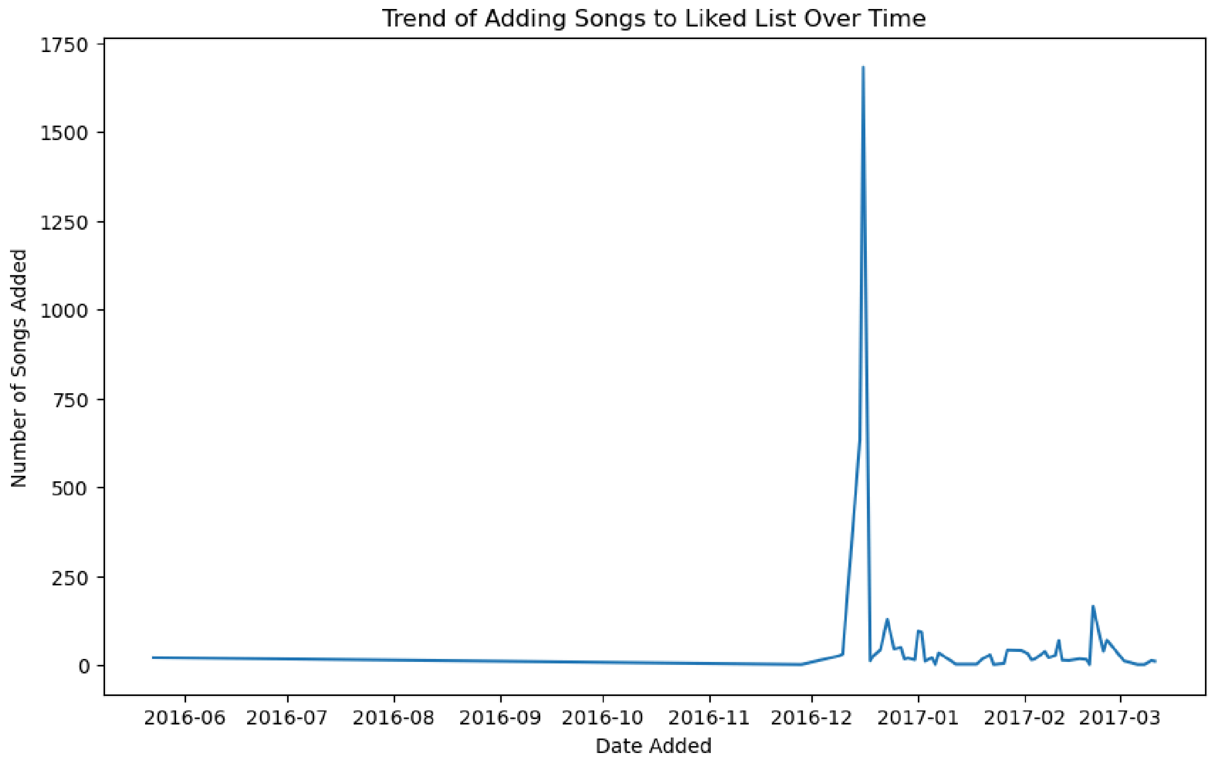
In [13]:
```
# Flatten the list of genres and create a genre DataFrame
genre_data = []
for genres in df_songs['genres']:
    for genre in genres:
        genre_data.append(genre)

df_genres = pd.DataFrame(genre_data, columns=['genre'])
```

In [14]:
```
plt.figure(figsize=(10, 6))
top_genres = df_genres['genre'].value_counts().head(10)
sns.barplot(x=top_genres.values, y=top_genres.index)
plt.title('Top 10 Genres of Liked Songs')
plt.xlabel('Number of Songs')
plt.ylabel('Genre')
plt.show()
```



Analyze Time-Based Trends

In [15]:
```
df_songs['added_date'] = pd.to_datetime(df_songs['added_at']).dt.date
plt.figure(figsize=(10, 6))
added_date_counts = df_songs['added_date'].value_counts().sort_index()
added_date_counts.plot()
plt.title('Trend of Adding Songs to Liked List Over Time')
plt.xlabel('Date Added')
plt.ylabel('Number of Songs Added')
plt.show()
```



Ensure the genres are combined properly

In [16]:
```
df_songs['genres'] = df_songs['genres'].apply(lambda x: ' '.join(x) if isins
```

Combine relevant metadata into a single string for each song

In [17]:
```
df_songs['metadata'] = df_songs[['artist_name', 'genres', 'popularity']].app
```

Calculate TF-IDF for the metadata

In [18]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
# Calculate TF-IDF for the metadata
tfidf_vectorizer = TfidfVectorizer(stop_words='english')
tfidf_matrix = tfidf_vectorizer.fit_transform(df_songs['metadata'])
```

Calculate cosine similarity

In [19]:
```
cosine_sim = cosine_similarity(tfidf_matrix, tfidf_matrix)
```

In [20]:
```
# Create a DataFrame for the similarity matrix
similarity_df = pd.DataFrame(cosine_sim, index=df_songs['track_name'], colum
```

Function to get recommendations for a given track

In [21]:
```
def get_recommendations(track_name, similarity_df, df_songs, num_recommendat
    similar_tracks = similarity_df[track_name].sort_values(ascending=False)[
    similar_track_names = similar_tracks.index
    return df_songs[df_songs['track_name'].isin(similar_track_names)]
```

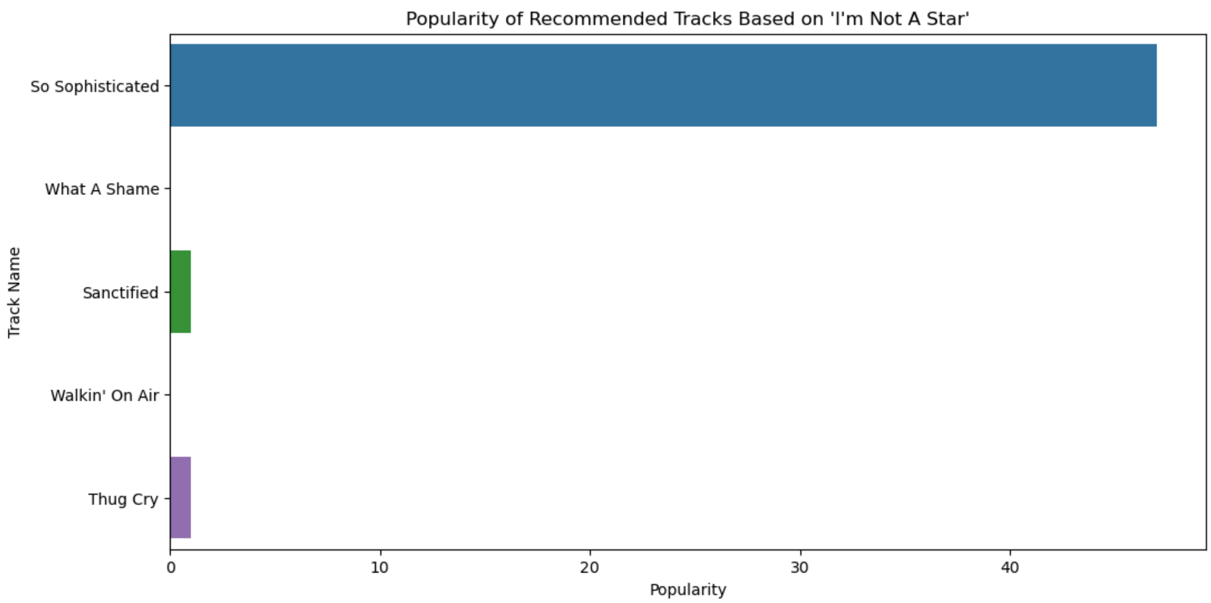
Get recommendations for a specific track

In [22]:
```
track_to_recommend = df_songs['track_name'].iloc[0]
recommendations = get_recommendations(track_to_recommend, similarity_df, df_
print(f"Recommendations based on {track_to_recommend}:")
print(recommendations[['track_name', 'artist_name', 'album', 'popularity', '
```

```
Recommendations based on I'm Not A Star:
           track_name artist_name                                   album  \
490  So Sophisticated   Rick Ross   God Forgives, I Don't (Deluxe Edition)
506      What A Shame   Rick Ross                     Mastermind (Deluxe)
511        Sanctified   Rick Ross                     Mastermind (Deluxe)
512    Walkin' On Air   Rick Ross                     Mastermind (Deluxe)
513          Thug Cry   Rick Ross                     Mastermind (Deluxe)

     popularity                                            genres
490          47  dirty south rap gangster rap hip hop rap south...
506           0  dirty south rap gangster rap hip hop rap south...
511           1  dirty south rap gangster rap hip hop rap south...
512           0  dirty south rap gangster rap hip hop rap south...
513           1  dirty south rap gangster rap hip hop rap south...
```

Visualize the popularity of recommended tracks

In [23]:
```
plt.figure(figsize=(12, 6))
sns.barplot(x='popularity', y='track_name', data=recommendations)
plt.title(f"Popularity of Recommended Tracks Based on '{track_to_recommend}'
plt.xlabel('Popularity')
plt.ylabel('Track Name')
plt.show()
```

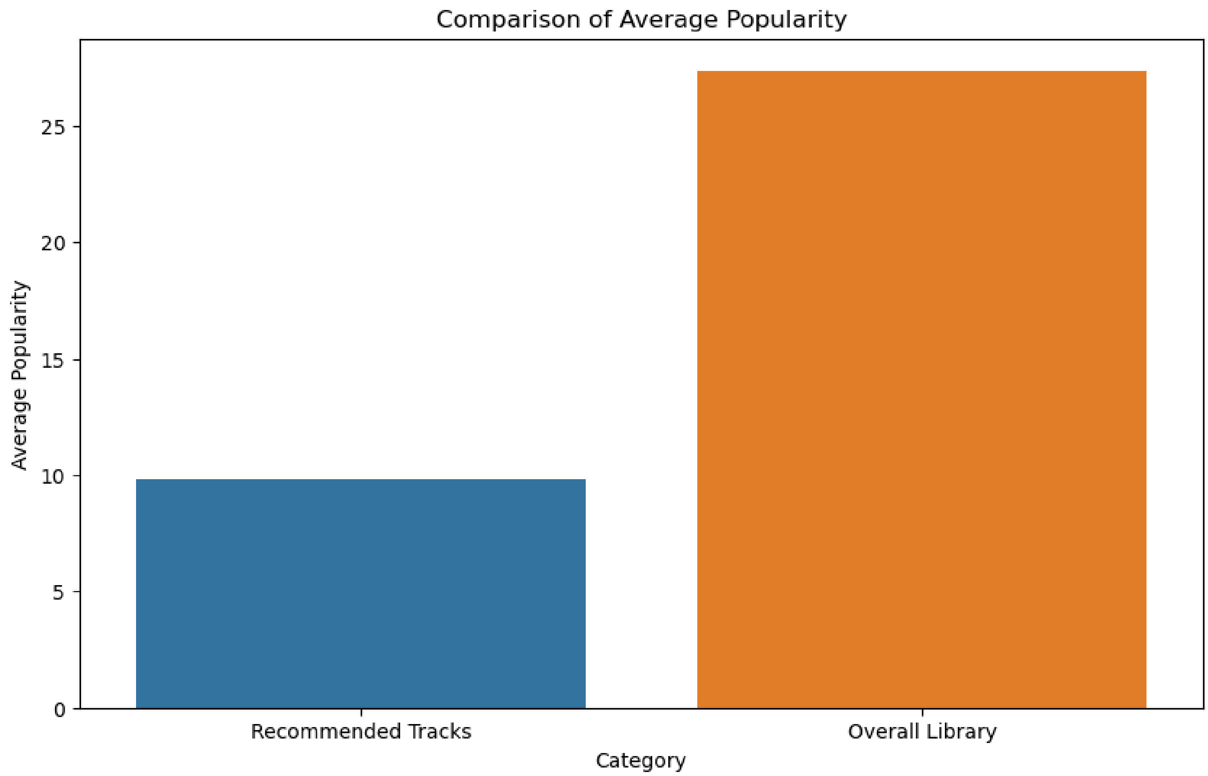


In [24]:
```
# Calculate the average popularity of the recommended tracks
avg_popularity_recommended = recommendations['popularity'].mean()

# Calculate the average popularity of the overall library
avg_popularity_overall = df_songs['popularity'].mean()
```

In [25]:
```
print(f"Average Popularity of Recommended Tracks: {avg_popularity_recommende
print(f"Average Popularity of Overall Library: {avg_popularity_overall}")
```

```
Average Popularity of Recommended Tracks: 9.8
Average Popularity of Overall Library: 27.36133548734518
```

In [26]:
```
# Visualize the comparison
plt.figure(figsize=(10, 6))
popularity_data = pd.DataFrame({
    'Category': ['Recommended Tracks', 'Overall Library'],
    'Average Popularity': [avg_popularity_recommended, avg_popularity_overal
})
sns.barplot(x='Category', y='Average Popularity', data=popularity_data)
plt.title('Comparison of Average Popularity')
plt.ylabel('Average Popularity')
plt.show()
```



In [27]:
```
# Flatten the list of genres for the recommended tracks
genre_data_recommended = []
for genres in recommendations['genres']:
    for genre in genres.split():
        genre_data_recommended.append(genre)

df_genres_recommended = pd.DataFrame(genre_data_recommended, columns=['genre
```

In [28]:
```
# Flatten the list of genres for the overall library
genre_data_overall = []
for genres in df_songs['genres']:
    for genre in genres.split():
        genre_data_overall.append(genre)

df_genres_overall = pd.DataFrame(genre_data_overall, columns=['genre'])
```

Calculate the top genres in both datasets

In [29]:
```
top_genres_recommended = df_genres_recommended['genre'].value_counts().head(
top_genres_overall = df_genres_overall['genre'].value_counts().head(10)
```

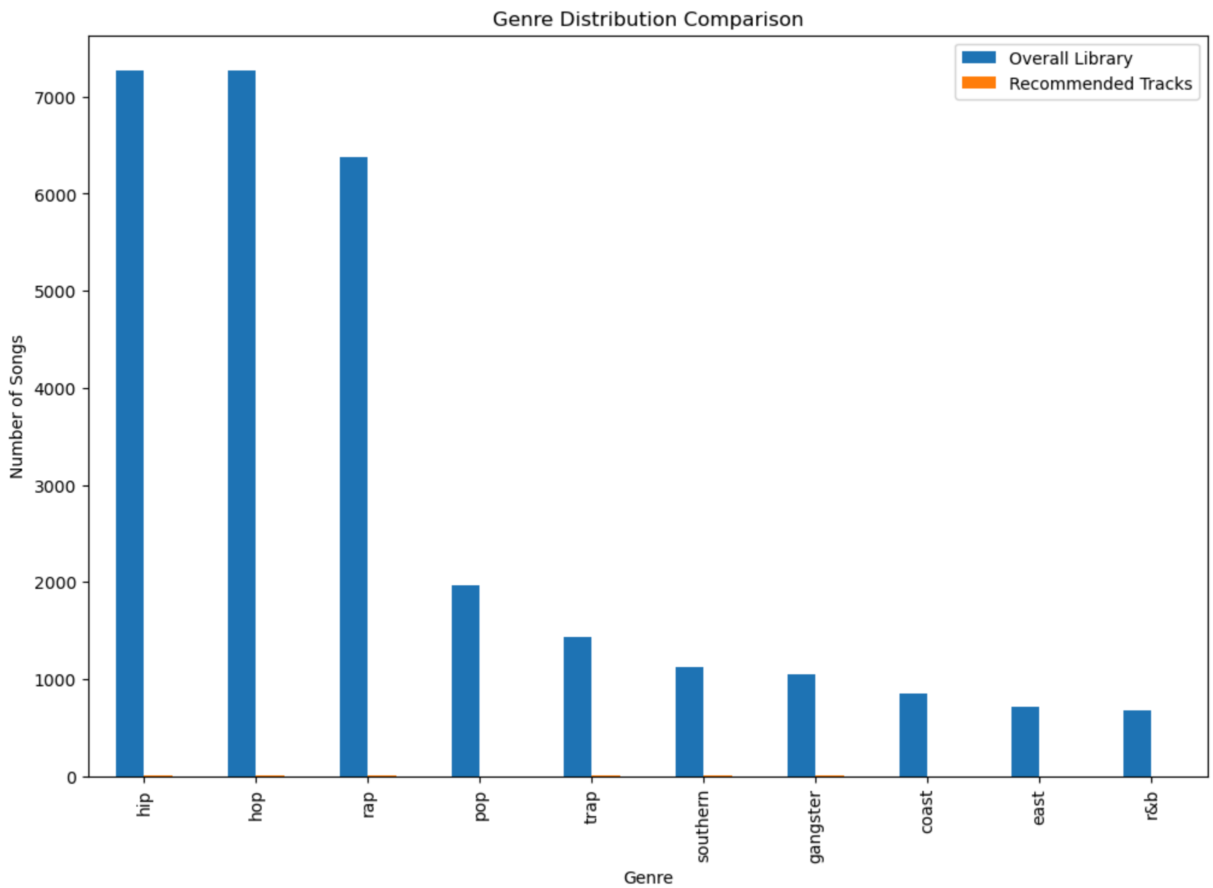
Create a DataFrame for comparison

In [30]:
```
genre_comparison = pd.DataFrame({
    'Genre': top_genres_overall.index,
    'Overall Library': top_genres_overall.values,
```

```
    'Recommended Tracks': top_genres_recommended.reindex(top_genres_overall.
})
```

Visualize the comparison

In [31]:
```
genre_comparison.set_index('Genre').plot(kind='bar', figsize=(12, 8))
plt.title('Genre Distribution Comparison')
plt.xlabel('Genre')
plt.ylabel('Number of Songs')
plt.show()
```



Select the top 5 most popular tracks from my library as seed tracks

In [32]:
```
top_seed_tracks = df_songs.sort_values(by='popularity', ascending=False).hea
```

Function to get recommendations for multiple seed tracks

In [33]:
```
def get_recommendations_multiple_seeds(seed_tracks, similarity_df, df_songs,
    similar_tracks = pd.Series(dtype=float)
```

```
    for track in seed_tracks:
        similar_tracks = pd.concat([similar_tracks, similarity_df[track].sor
    similar_tracks = similar_tracks.groupby(similar_tracks.index).mean().sor
    similar_track_names = similar_tracks.head(num_recommendations).index
    return df_songs[df_songs['track_name'].isin(similar_track_names)]
```

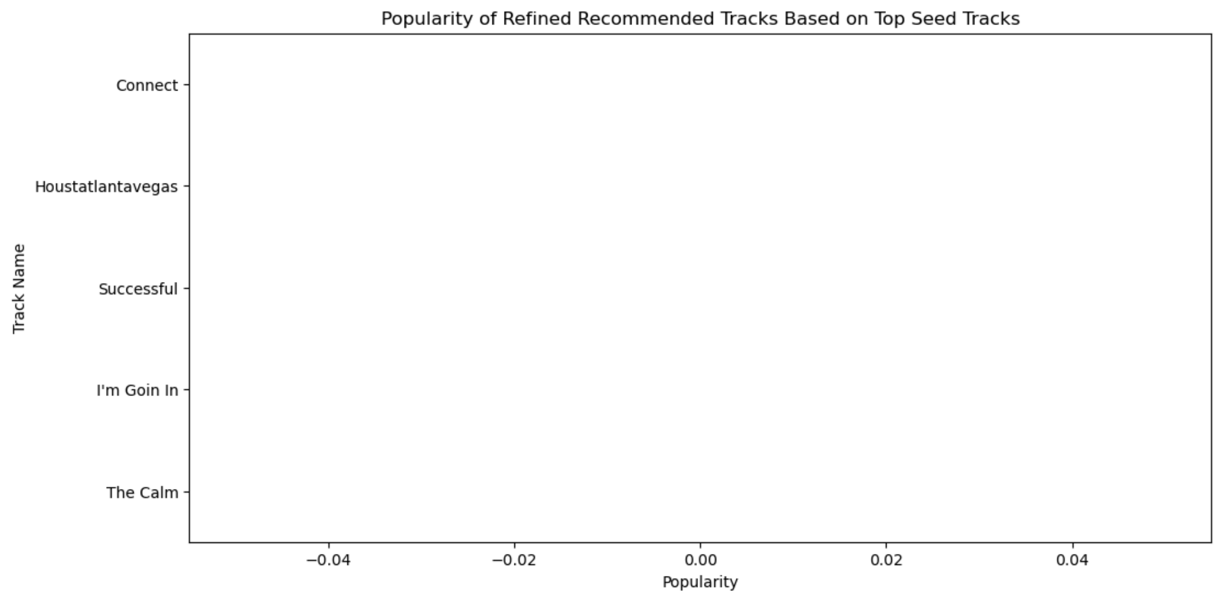
Generate refined recommendations based on multiple seed tracks

In [34]:
```
refined_recommendations = get_recommendations_multiple_seeds(top_seed_tracks
print(f"Refined Recommendations based on top seed tracks:")
print(refined_recommendations[['track_name', 'artist_name', 'album', 'popula
```

```
Refined Recommendations based on top seed tracks:
             track_name artist_name                          album  \
1017            Connect       Drake  Nothing Was The Same (Deluxe)
2897   Houstatlantavegas      Drake                    So Far Gone
2898         Successful       Drake                    So Far Gone
2901          I'm Goin In     Drake                    So Far Gone
2902           The Calm       Drake                    So Far Gone

      popularity                                       genres
1017           0  canadian hip hop canadian pop hip hop pop rap rap
2897           0  canadian hip hop canadian pop hip hop pop rap rap
2898           0  canadian hip hop canadian pop hip hop pop rap rap
2901           0  canadian hip hop canadian pop hip hop pop rap rap
2902           0  canadian hip hop canadian pop hip hop pop rap rap
```

Visualize the popularity of the refined recommended tracks

In [35]:
```
plt.figure(figsize=(12, 6))
sns.barplot(x='popularity', y='track_name', data=refined_recommendations)
plt.title(f"Popularity of Refined Recommended Tracks Based on Top Seed Track
plt.xlabel('Popularity')
plt.ylabel('Track Name')
plt.show()
```



Scale numerical features

In [36]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df_songs[['popularity']] = scaler.fit_transform(df_songs[['popularity']])
```

In [37]:
```
# Combine relevant metadata into a single string for each song, with adjuste
df_songs['metadata'] = df_songs.apply(lambda x: f"{x['artist_name']} {' '.jc
```

In [38]:
```
# Calculate TF-IDF for the adjusted metadata
tfidf_matrix_adjusted = tfidf_vectorizer.fit_transform(df_songs['metadata'])

# Calculate cosine similarity for the adjusted metadata
cosine_sim_adjusted = cosine_similarity(tfidf_matrix_adjusted, tfidf_matrix_
```

In [39]:
```
# Create a DataFrame for the adjusted similarity matrix
similarity_df_adjusted = pd.DataFrame(cosine_sim_adjusted, index=df_songs['t
```

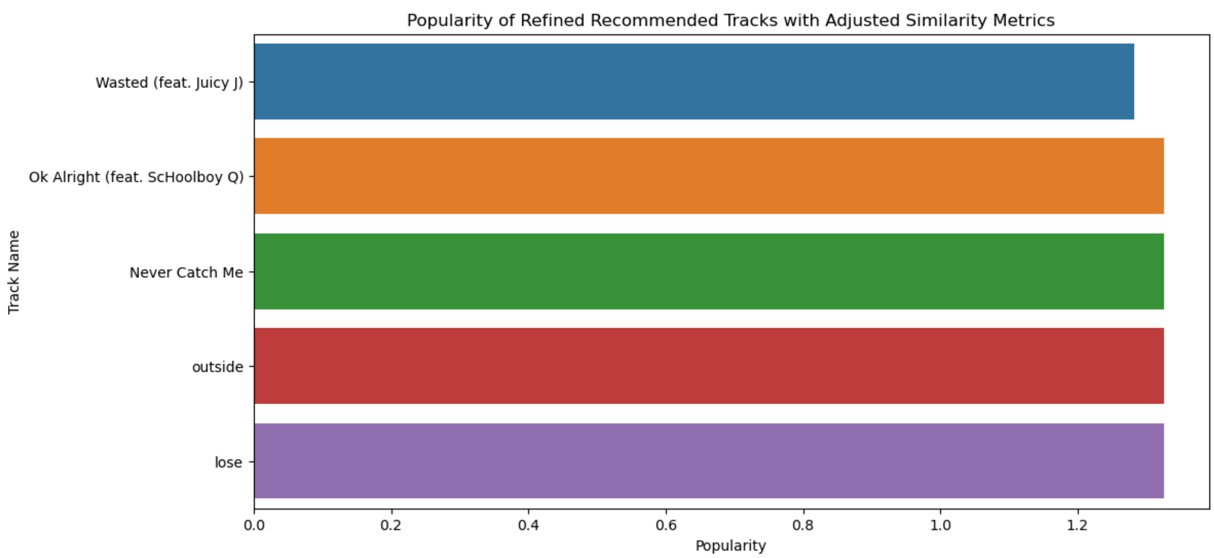
Generate refined recommendations based on the adjusted similarity metrics

In [40]:
```
refined_recommendations_adjusted = get_recommendations_multiple_seeds(top_se
print(f"Refined Recommendations with Adjusted Similarity Metrics:")
print(refined_recommendations_adjusted[['track_name', 'artist_name', 'album'
```

```
Refined Recommendations with Adjusted Similarity Metrics:
                        track_name   artist_name  \
622          Wasted (feat. Juicy J)  Travis Scott
633  Ok Alright (feat. ScHoolboy Q)  Travis Scott
634                  Never Catch Me  Travis Scott
982                         outside  Travis Scott
986                            lose  Travis Scott

                               album  popularity          genres
622                            Rodeo    1.281978  rap slap house
633                            Rodeo    1.325232  rap slap house
634                            Rodeo    1.325232  rap slap house
982  Birds In The Trap Sing McKnight    1.325232  rap slap house
986  Birds In The Trap Sing McKnight    1.325232  rap slap house
```

Visualize the popularity of refined recommended tracks with the adjusted similarity metrics

In [41]:
```
plt.figure(figsize=(12, 6))
sns.barplot(x='popularity', y='track_name', data=refined_recommendations_adj
plt.title(f"Popularity of Refined Recommended Tracks with Adjusted Similarit
plt.xlabel('Popularity')
plt.ylabel('Track Name')
plt.show()
```



Function to fetch audio features for tracks

In [42]:
```
def fetch_audio_features(sp, track_ids):
    audio_features = []
    for i in range(0, len(track_ids), 50):  # Spotify API allows max 100 tra
        batch = track_ids[i:i+50]
```

```
        audio_features.extend(sp.audio_features(batch))
    return [af for af in audio_features if af is not None]
```

Fetch audio features for all liked songs

In [43]:
```
track_ids = df_songs['track_id'].tolist()
audio_features = fetch_audio_features(sp, track_ids)
```

Convert audio features to DataFrame

In [44]:
```
df_audio_features = pd.DataFrame(audio_features)
print(df_audio_features)
```

```
      danceability  energy  key  loudness  mode  speechiness  acousticness  \
0            0.667   0.848    1    -4.406     1       0.3620       0.04510   
1            0.488   0.857    1    -5.148     1       0.4110       0.06030   
2            0.467   0.871   11    -6.484     0       0.4450       0.30100   
3            0.410   0.791    9    -5.703     1       0.1710       0.06690   
4            0.491   0.880    4    -3.985     1       0.3190       0.16800   
...            ...     ...  ...       ...   ...          ...           ...   
3708         0.794   0.653    7    -7.839     1       0.1040       0.04890   
3709         0.699   0.255    4    -8.647     0       0.0303       0.40500   
3710         0.722   0.252    1   -14.411     0       0.0761       0.06710   
3711         0.395   0.852    5    -5.896     1       0.3700       0.06570   
3712         0.891   0.628    2    -7.863     1       0.0551       0.00258   

      instrumentalness  liveness  valence    tempo            type  \
0             0.000006    0.2170    0.196  155.974  audio_features   
1             0.000000    0.1870    0.424   86.426  audio_features   
2             0.000000    0.6820    0.428   92.504  audio_features   
3             0.000000    0.1230    0.203  167.517  audio_features   
4             0.000000    0.4080    0.699  169.781  audio_features   
...                ...       ...      ...      ...             ...   
3708          0.000049    0.1000    0.397  117.996  audio_features   
3709          0.002420    0.0985    0.242  132.031  audio_features   
3710          0.000000    0.0852    0.275   79.923  audio_features   
3711          0.000000    0.2620    0.112   76.428  audio_features   
3712          0.000190    0.0504    0.552  134.966  audio_features   

                          id                                   uri  \
0     1fRuRNJVZjDU1yKXvarKqW  spotify:track:1fRuRNJVZjDU1yKXvarKqW   
1     7DuVZVdHaFQIYg14VNykXi  spotify:track:7DuVZVdHaFQIYg14VNykXi   
2     1amE1IohObbqkU9UzV8uFl  spotify:track:1amE1IohObbqkU9UzV8uFl   
3     06aiCjSnK0XXBEERhduZWZ  spotify:track:06aiCjSnK0XXBEERhduZWZ   
4     4L0dkLS6mpsf6zRKVSwfWY  spotify:track:4L0dkLS6mpsf6zRKVSwfWY   
...                      ...                                   ...   
3708  3BtuIIrQlkujKPuWF2B85z  spotify:track:3BtuIIrQlkujKPuWF2B85z   
3709  3ppVO2tyWRRznNmONvt7Se  spotify:track:3ppVO2tyWRRznNmONvt7Se   
3710  4BhGTc3Cgay2U1QcTS7vQe  spotify:track:4BhGTc3Cgay2U1QcTS7vQe   
3711  7MjSipTto9QljYzZnloXOn  spotify:track:7MjSipTto9QljYzZnloXOn   
3712  0wwPcA6wtMf6HUMpIRdeP7  spotify:track:0wwPcA6wtMf6HUMpIRdeP7   

                                             track_href  \
0     https://api.spotify.com/v1/tracks/1fRuRNJVZjDU...   
1     https://api.spotify.com/v1/tracks/7DuVZVdHaFQI...   
2     https://api.spotify.com/v1/tracks/1amE1IohObbq...   
3     https://api.spotify.com/v1/tracks/06aiCjSnK0XX...   
4     https://api.spotify.com/v1/tracks/4L0dkLS6mpsf...   
...                                                 ...   
3708  https://api.spotify.com/v1/tracks/3BtuIIrQlkuj...   
3709  https://api.spotify.com/v1/tracks/3ppVO2tyWRRz...   
3710  https://api.spotify.com/v1/tracks/4BhGTc3Cgay2...   
3711  https://api.spotify.com/v1/tracks/7MjSipTto9Ql...   
3712  https://api.spotify.com/v1/tracks/0wwPcA6wtMf6...   

                                           analysis_url  duration_ms  \
0     https://api.spotify.com/v1/audio-analysis/1fRu...       180107   
1     https://api.spotify.com/v1/audio-analysis/7DuV...       247213   
```

```
2     https://api.spotify.com/v1/audio-analysis/1amE...       333400
3     https://api.spotify.com/v1/audio-analysis/06ai...       265987
4     https://api.spotify.com/v1/audio-analysis/4L0d...       373800
...                                                ...          ...
3708  https://api.spotify.com/v1/audio-analysis/3Btu...       263373
3709  https://api.spotify.com/v1/audio-analysis/3ppV...       106333
3710  https://api.spotify.com/v1/audio-analysis/4BhG...       238120
3711  https://api.spotify.com/v1/audio-analysis/7MjS...       311960
3712  https://api.spotify.com/v1/audio-analysis/0wwP...       267067

      time_signature
0                  4
1                  4
2                  4
3                  4
4                  4
...              ...
3708               4
3709               3
3710               4
3711               4
3712               4

[3713 rows x 18 columns]
```

In [45]:
```
# Merge audio features with the original song data
df_songs = df_songs.merge(df_audio_features, left_on='track_id', right_on='i
```

Summary stats for audio features

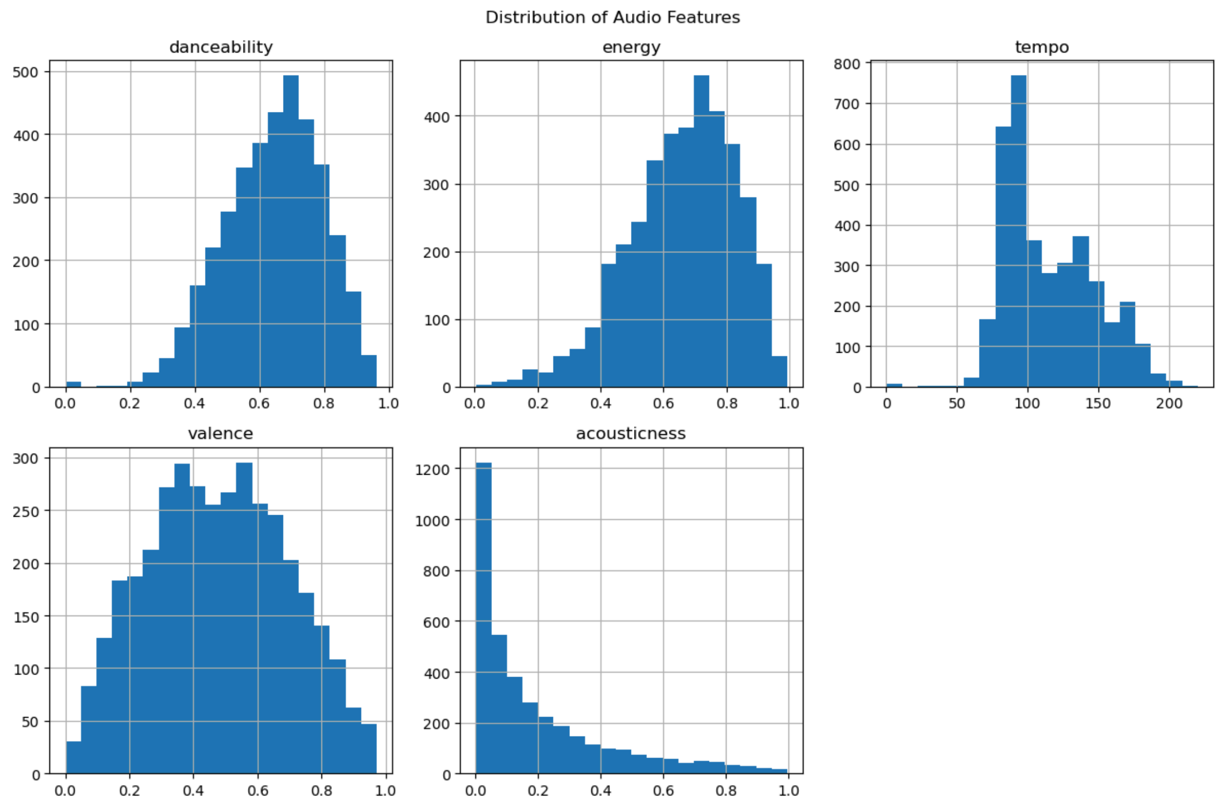
In [46]:
```
summary_stats = df_songs[['danceability', 'energy', 'tempo', 'valence', 'acc
print(summary_stats)
```

```
       danceability       energy        tempo      valence  acousticness
count    3713.00000  3713.000000  3713.000000  3713.000000   3713.000000
mean        0.64415     0.660034   115.230394     0.476729      0.196261
std         0.14914     0.168405    32.176666     0.218086      0.222453
min         0.00000     0.003330     0.000000     0.000000      0.000013
25%         0.54300     0.552000    89.387000     0.311000      0.030300
50%         0.65800     0.682000   106.035000     0.475000      0.110000
75%         0.75400     0.786000   139.948000     0.642000      0.280000
max         0.96300     0.993000   220.251000     0.972000      0.995000
```

Visualize distribution of audio features

In [47]:
```
plt.figure(figsize=(12, 8))
df_songs[['danceability', 'energy', 'tempo', 'valence', 'acousticness']].his
plt.suptitle('Distribution of Audio Features')
plt.tight_layout()
plt.show()
```

```
<Figure size 1200x800 with 0 Axes>
```



Compare audio features between liked songs and recommended tracks

In [48]:
```
recommended_track_ids = refined_recommendations_adjusted['track_id'].tolist(
recommended_audio_features = fetch_audio_features(sp, recommended_track_ids)
```

In [49]:
```
# Convert recommended audio features to DataFrame
df_recommended_audio_features = pd.DataFrame(recommended_audio_features)
```

In [50]:
```
# Merge recommended audio features with the original recommended tracks data
refined_recommendations_adjusted = refined_recommendations_adjusted.merge(df
```

Summary stats for recommended tracks' audio features

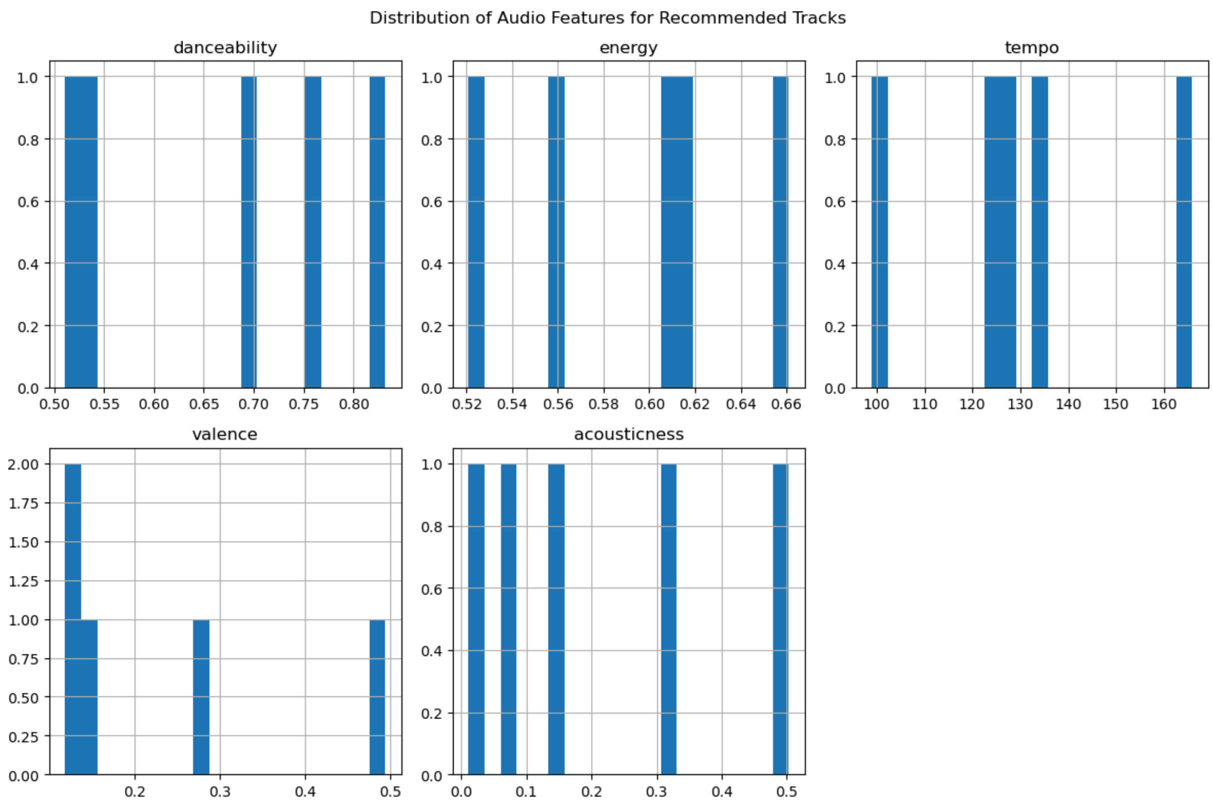
In [51]:
```
summary_stats_recommended = refined_recommendations_adjusted[['danceability'
print(summary_stats_recommended)
```

```
       danceability    energy       tempo   valence  acousticness
count      5.000000  5.000000    5.000000  5.000000      5.000000
mean       0.664000  0.593000  129.675200  0.233400      0.209200
std        0.140579  0.054594   24.202086  0.159481      0.197126
min        0.511000  0.521000   99.058000  0.119000      0.011400
25%        0.530000  0.557000  122.482000  0.126000      0.083600
50%        0.689000  0.610000  125.999000  0.149000      0.141000
75%        0.758000  0.616000  134.931000  0.279000      0.307000
max        0.832000  0.661000  165.906000  0.494000      0.503000
```

Visualize distribution of audio features for the recommended tracks

In [52]:
```python
plt.figure(figsize=(12, 8))
refined_recommendations_adjusted[['danceability', 'energy', 'tempo', 'valenc
plt.suptitle('Distribution of Audio Features for Recommended Tracks')
plt.tight_layout()
plt.show()
```

```
<Figure size 1200x800 with 0 Axes>
```



Select audio features for clustering

In [53]:
```python
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
audio_features = df_songs[['danceability', 'energy', 'tempo', 'valence', 'ac
```

Perform KMeans clustering

In [54]:
```
kmeans = KMeans(n_clusters=5, random_state=42)
df_songs['cluster'] = kmeans.fit_predict(audio_features)
```

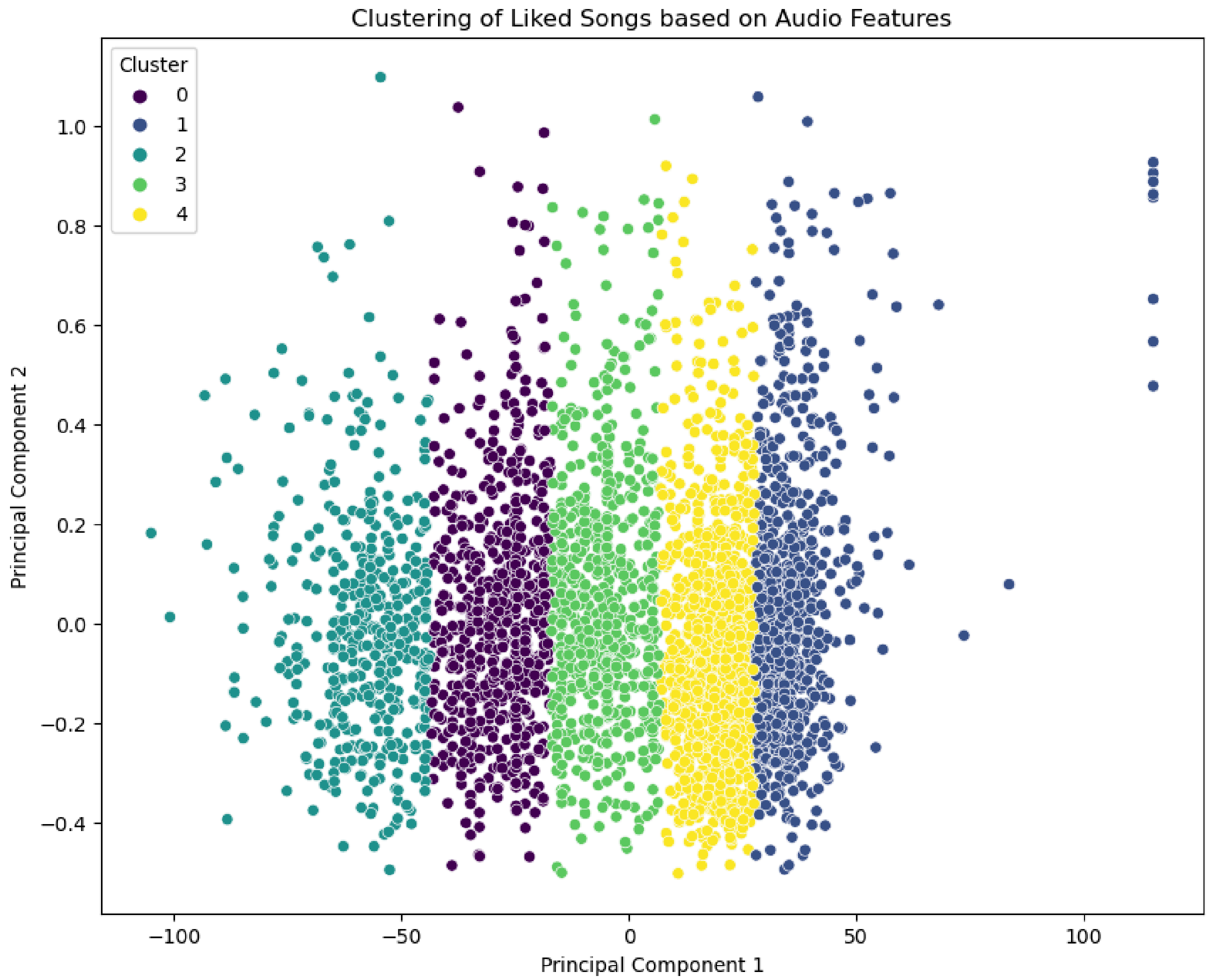
```
/opt/anaconda3/lib/python3.11/site-packages/sklearn/cluster/_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' i
n 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
```

Perform PCA for dimensionality reduction

In [55]:
```
pca = PCA(n_components=2)
principal_components = pca.fit_transform(audio_features)
df_songs['pca1'] = principal_components[:, 0]
df_songs['pca2'] = principal_components[:, 1]
```

Visualize the Liked Songs based on Audio Features clusters

In [56]:
```
plt.figure(figsize=(10, 8))
sns.scatterplot(x='pca1', y='pca2', hue='cluster', data=df_songs, palette='v
plt.title('Clustering of Liked Songs based on Audio Features')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend(title='Cluster')
plt.show()
```



Analyzing the clusters

In [57]:
```
for cluster in df_songs['cluster'].unique():
    cluster_data = df_songs[df_songs['cluster'] == cluster]
    print(f"\nCluster {cluster}:")
    print(cluster_data[['track_name', 'artist_name', 'album', 'genres']].hea
    print(cluster_data[['danceability', 'energy', 'tempo', 'valence', 'acous
```

```
Cluster 0:
                    track_name artist_name            album  \
0               I'm Not A Star   Rick Ross       Teflon Don
7                    MC Hammer   Rick Ross       Teflon Don
8   B.M.F. (Blowin' Money Fast)   Rick Ross       Teflon Don
13                   EdEddnEddy         JID  The Never Story
15                   Hereditary         JID  The Never Story

                                               genres
0   dirty south rap gangster rap hip hop rap south...
7   dirty south rap gangster rap hip hop rap south...
8   dirty south rap gangster rap hip hop rap south...
13            hip hop pop rap rap underground hip hop
15            hip hop pop rap rap underground hip hop
       danceability      energy       tempo     valence  acousticness
count    692.000000  692.000000  692.000000  692.000000    692.000000
mean       0.652312    0.647579  143.683199    0.441629      0.181248
std        0.135835    0.161196    6.969985    0.203624      0.215521
min        0.202000    0.045200  132.525000    0.038200      0.000030
25%        0.556750    0.546000  138.035500    0.281000      0.025975
50%        0.662000    0.660000  142.130000    0.430000      0.099450
75%        0.752250    0.764250  149.841250    0.585000      0.248000
max        0.951000    0.993000  158.528000    0.971000      0.976000

Cluster 1:
    track_name artist_name            album  \
1   Free Mason   Rick Ross       Teflon Don
10     Doo Wop         JID  The Never Story
11     General         JID  The Never Story
14    D/vision         JID  The Never Story
16     All Bad         JID  The Never Story

                                               genres
1   dirty south rap gangster rap hip hop rap south...
10            hip hop pop rap rap underground hip hop
11            hip hop pop rap rap underground hip hop
14            hip hop pop rap rap underground hip hop
16            hip hop pop rap rap underground hip hop
       danceability     energy       tempo     valence  acousticness
count    796.000000  796.00000  796.000000  796.000000    796.000000
mean       0.573366    0.65194   79.199683    0.465380      0.231536
std        0.151418    0.18040   10.110152    0.220636      0.248199
min        0.000000    0.00333    0.000000    0.000000      0.000013
25%        0.469000    0.53850   77.030750    0.304500      0.038875
50%        0.585500    0.68550   81.058000    0.457000      0.138000
75%        0.683250    0.78300   84.360750    0.630250      0.339500
max        0.898000    0.99000   87.439000    0.971000      0.991000

Cluster 4:
      track_name artist_name            album  \
2   Tears Of Joy   Rick Ross       Teflon Don
12         NEVER         JID  The Never Story
17     Underwear         JID  The Never Story
20      Somebody         JID  The Never Story
21        LAUDER         JID  The Never Story
```

```
                                              genres
2   dirty south rap gangster rap hip hop rap south...
12            hip hop pop rap rap underground hip hop
17            hip hop pop rap rap underground hip hop
20            hip hop pop rap rap underground hip hop
21            hip hop pop rap rap underground hip hop
       danceability       energy        tempo      valence  acousticness
count   1113.000000  1113.000000  1113.000000  1113.000000   1113.000000
mean       0.698220     0.677177    95.707966     0.529208      0.182635
std        0.135283     0.160356     5.354725     0.218874      0.206046
min        0.299000     0.124000    87.500000     0.031400      0.000017
25%        0.614000     0.581000    91.376000     0.370000      0.028600
50%        0.718000     0.698000    94.822000     0.547000      0.102000
75%        0.801000     0.798000    99.879000     0.695000      0.263000
max        0.962000     0.984000   108.164000     0.961000      0.985000

Cluster 2:
                   track_name artist_name       album  \
3          Maybach Music III   Rick Ross  Teflon Don
4        Live Fast, Die Young   Rick Ross  Teflon Don
5                  Super High   Rick Ross  Teflon Don
6                       No. 1   Rick Ross  Teflon Don
9  All The Money In The World   Rick Ross  Teflon Don

                                             genres
3  dirty south rap gangster rap hip hop rap south...
4  dirty south rap gangster rap hip hop rap south...
5  dirty south rap gangster rap hip hop rap south...
6  dirty south rap gangster rap hip hop rap south...
9  dirty south rap gangster rap hip hop rap south...
       danceability      energy       tempo     valence  acousticness
count    462.000000  462.000000  462.000000  462.000000    462.000000
mean       0.567935    0.698379  173.619266    0.496287      0.167073
std        0.132841    0.158056   10.554762    0.215260      0.199999
min        0.141000    0.130000  158.935000    0.039900      0.000025
25%        0.472250    0.594750  166.135000    0.331000      0.028025
50%        0.565000    0.720500  172.044000    0.495000      0.085400
75%        0.667250    0.825000  179.004500    0.670750      0.236250
max        0.866000    0.978000  220.251000    0.972000      0.995000

Cluster 3:
                 track_name    artist_name                       album  \
22                    Smile  Isaiah Rashad                       Smile
26                Big Rings          Drake     What A Time To Be Alive
29             Scholarships          Drake     What A Time To Be Alive
41  Smile (feat. Timbaland)       Yo Gotti  The Art of Hustle (Deluxe)
48   Hunnid (feat. Pusha T)       Yo Gotti  The Art of Hustle (Deluxe)

                                              genres
22  hip hop rap tennessee hip hop underground hip hop
26  canadian hip hop canadian pop hip hop pop rap rap
29  canadian hip hop canadian pop hip hop pop rap rap
41  dirty south rap gangster rap memphis hip hop r...
48  dirty south rap gangster rap memphis hip hop r...
       danceability      energy       tempo     valence  acousticness
count    650.000000  650.000000  650.000000  650.000000    650.000000
```

mean	0.683731	0.626599	120.990252	0.424234	0.213121
std	0.136610	0.173084	6.874791	0.209839	0.232353
min	0.184000	0.079200	108.432000	0.000000	0.000032
25%	0.596000	0.512250	115.505500	0.260000	0.033300
50%	0.695000	0.636500	120.263000	0.405000	0.126500
75%	0.782750	0.756000	126.982000	0.572750	0.316000
max	0.963000	0.982000	132.205000	0.970000	0.978000

Fetching my popular tracks

In [58]:
```python
popular_tracks = sp.playlist_tracks('37i9dQZF1DXcBWIGoYBM5M', limit=50)  # S
popular_track_data = []
```

In [59]:
```python
for item in popular_tracks['items']:
    track = item['track']
    track_info = {
        'track_id': track['id'],
        'track_name': track['name'],
        'artist': track['artists'][0]['name'],
        'popularity': track['popularity'],
        'album': track['album']['name'],
    }
    popular_track_data.append(track_info)
```

In [60]:
```python
df_popular_tracks = pd.DataFrame(popular_track_data)
print(df_popular_tracks)
```

```
                  track_id                                       track_name  \
0   6dOtVTDdiauQNBQEDOtlAB                               BIRDS OF A FEATHER   
1   2qSkIjg1o9h3YT9RAgYN75                                         Espresso   
2   4IadxL6BUymXlh8RCJJu7T                                        Too Sweet   
3   7221xIgOnuakPdLqT0F3nP            I Had Some Help (Feat. Morgan Wallen)   
4   2OzhQlSqBEmt7hmkYxfT6m                    Fortnight (feat. Post Malone)   
5   7fzHQizxTqy8wTXwlrgPQQ                              MILLION DOLLAR BABY   
6   629DixmZGHc7ILtEntuiWE                                            LUNCH   
7   46kspZSY3aKmwQe7O77fCC         we can't be friends (wait for your love)   
8   2FQrifJ1N335Ljm3TjTVVf                               A Bar Song (Tipsy)   
9   6tNQ70jh4OwmPGpYy6R2o9                                 Beautiful Things   
10  2HYFX63wP3otVIvopRS99Z                                          Houdini   
11  2GxrNKugF82CnoRFbQfzPf                       i like the way you kiss me   
12  6AI3ezQ4o3HUoP6Dhudph3                                      Not Like Us   
13  3qhlB30KknSejmIvZZLjOD                                 End of Beginning   
14  2uqYupMHANxnwgeiXTZXzd                      Austin (Boots Stop Workin')   
15  4q5YezDOIPcoLr8R81x9qy                  I Can Do It With a Broken Heart   
16  17phhZDn6oGtzMe56NuWvj                                     Lose Control   
17  1bjeWoagtHmUKputLVyDxQ                                           Saturn   
18  0WbMK4wrZ1wFSty9F7FCgu                                 Good Luck, Babe!   
19  5uQ7de4EWjb3rkcFxyEOpu                                  Belong Together   
20  0mflMxspEfB0VbI1kyLiAv                                     Stick Season   
21  51eSHglvG1RJXtL3qI5trr                                     Slow It Down   
22  3rUGC1vUpkDG9CZFHMur1t                                           greedy   
23  0Z7nGFVCLfixWctgePsRk9                                   TEXAS HOLD 'EM   
24  3Vr3zh0r7ALn8VLqCiRR10                                       Stargazing   
25  2Zo1PcszsT9WQ0ANntJbID                                          Feather   
26  3Pbp7cUCx4d3OAkZSCoNvn                                  Scared To Start   
27  4ZJ4vzLQekI0WntDbanNC7                                       Pink Skies   
28  6XjDF6nds4DE2BBbagZol6                                        Gata Only   
29  5aIVCx5tnk0ntmdiinnYvw                                            Water   
30  4pkb8SbRGeHAvdb87v9rpf                                      Miles On It   
31  3SAga35lAPYdjj3qyfEsCF  Feel It - From The Original Series “Invincible”   
32  7BRD7x5pt8Lqa1eGYC4dzj                                          CHIHIRO   
33  7gaA3wERFkFkgivjwbSvkG                                        yes, and?   
34  3lMzT16MjAKKXF7pSZn13B                               Tell Ur Girlfriend   
35  7CyPwkp0oE8Ro9Dd5CUDjW   One Of The Girls (with JENNIE, Lily Rose Depp)   
36  7iabz12vAuVQYyekFIWJxD                       BAND4BAND (feat. Lil Baby)   
37  4xhsWYTOGcal8zt0J161CU                                      Lovin On Me   
38  5bi0gh89wRuH2OgjdAKFsb                                            Santa   
39  59xD5osEFsaNt5PXfIKUnX                                         Illusion   
40  0LMwmV37RCmBO2so0szAFs                                         Whatever   
41  57wp7VFnV8X0pSVnYArGeJ                               Whatever She Wants   
42  4Na2HfNSr58chvfX69fy36                                       one of wun   
43  331l3xABO0HMr1Kkyh2LZq                               I Don't Wanna Wait   
44  0ve0CavjqrUqVmZ605RhTV                                             Jump   
45  4KULAymBBJcPRpk1yO4dOG     I Remember Everything (feat. Kacey Musgraves)   
46  6dpLxbF7lfCAnC9QRTjNLK                                             Home   
47  1aKvZDoLGkNMxoRYgkckZG                                         Magnetic   
48  6tNgRQ0K2NYZ0Rb9l9DzL8                                         obsessed   
49  52eIcoLUM25zbQupAZYoFh                                           redrum   

               artist  popularity  \
0       Billie Eilish          95   
1   Sabrina Carpenter          99   
```

2	Hozier	85
3	Post Malone	96
4	Taylor Swift	94
5	Tommy Richman	98
6	Billie Eilish	96
7	Ariana Grande	88
8	Shaboozey	96
9	Benson Boone	93
10	Eminem	87
11	Artemas	97
12	Kendrick Lamar	97
13	Djo	95
14	Dasha	91
15	Taylor Swift	89
16	Teddy Swims	91
17	SZA	92
18	Chappell Roan	91
19	Mark Ambor	92
20	Noah Kahan	91
21	Benson Boone	89
22	Tate McRae	92
23	Beyoncé	86
24	Myles Smith	90
25	Sabrina Carpenter	89
26	Michael Marcagi	89
27	Zach Bryan	86
28	FloyyMenor	97
29	Tyla	88
30	Marshmello	84
31	d4vd	86
32	Billie Eilish	94
33	Ariana Grande	82
34	Lay Bankz	89
35	The Weeknd	92
36	Central Cee	88
37	Jack Harlow	88
38	Rvssian	91
39	Dua Lipa	83
40	Kygo	88
41	Bryson Tiller	87
42	Gunna	85
43	David Guetta	88
44	Tyla	79
45	Zach Bryan	89
46	Good Neighbours	86
47	ILLIT	91
48	Olivia Rodrigo	85
49	21 Savage	89

	album
0	HIT ME HARD AND SOFT
1	Espresso
2	Unheard
3	I Had Some Help
4	THE TORTURED POETS DEPARTMENT
5	MILLION DOLLAR BABY

```
6                                    HIT ME HARD AND SOFT
7                                        eternal sunshine
8                                      A Bar Song (Tipsy)
9                                        Beautiful Things
10                                                Houdini
11                              i like the way you kiss me
12                                            Not Like Us
13                                                 DECIDE
14                                      What Happens Now?
15                           THE TORTURED POETS DEPARTMENT
16             I've Tried Everything But Therapy (Part 1)
17                                                 Saturn
18                                        Good Luck, Babe!
19                                       Belong Together
20                                           Stick Season
21                               Fireworks & Rollerblades
22                                                 greedy
23                                         TEXAS HOLD 'EM
24                                             Stargazing
25                                emails i can’t send fwd:
26                                        Scared To Start
27                                             Pink Skies
28                                              Gata Only
29                                                  Water
30                                            Miles On It
31     Feel It (From The Original Series “Invincible”)
32                                   HIT ME HARD AND SOFT
33                                              yes, and?
34                                     Tell Ur Girlfriend
35   The Idol Episode 4 (Music from the HBO Origina...
36                               BAND4BAND (feat. Lil Baby)
37                                            Lovin On Me
38                                                  Santa
39                                               Illusion
40                                               Whatever
41                                     Whatever She Wants
42                                             One of Wun
43                                     I Don't Wanna Wait
44                                                   Jump
45                                             Zach Bryan
46                                                   Home
47                                          SUPER REAL ME
48                                         GUTS (spilled)
49                                         american dream
```

In [61]:
```python
# Merge popular tracks audio features
popular_track_ids = df_popular_tracks['track_id'].tolist()
popular_audio_features = fetch_audio_features(sp, popular_track_ids)
df_popular_audio_features = pd.DataFrame(popular_audio_features)
df_popular_tracks = df_popular_tracks.merge(df_popular_audio_features, left_
```

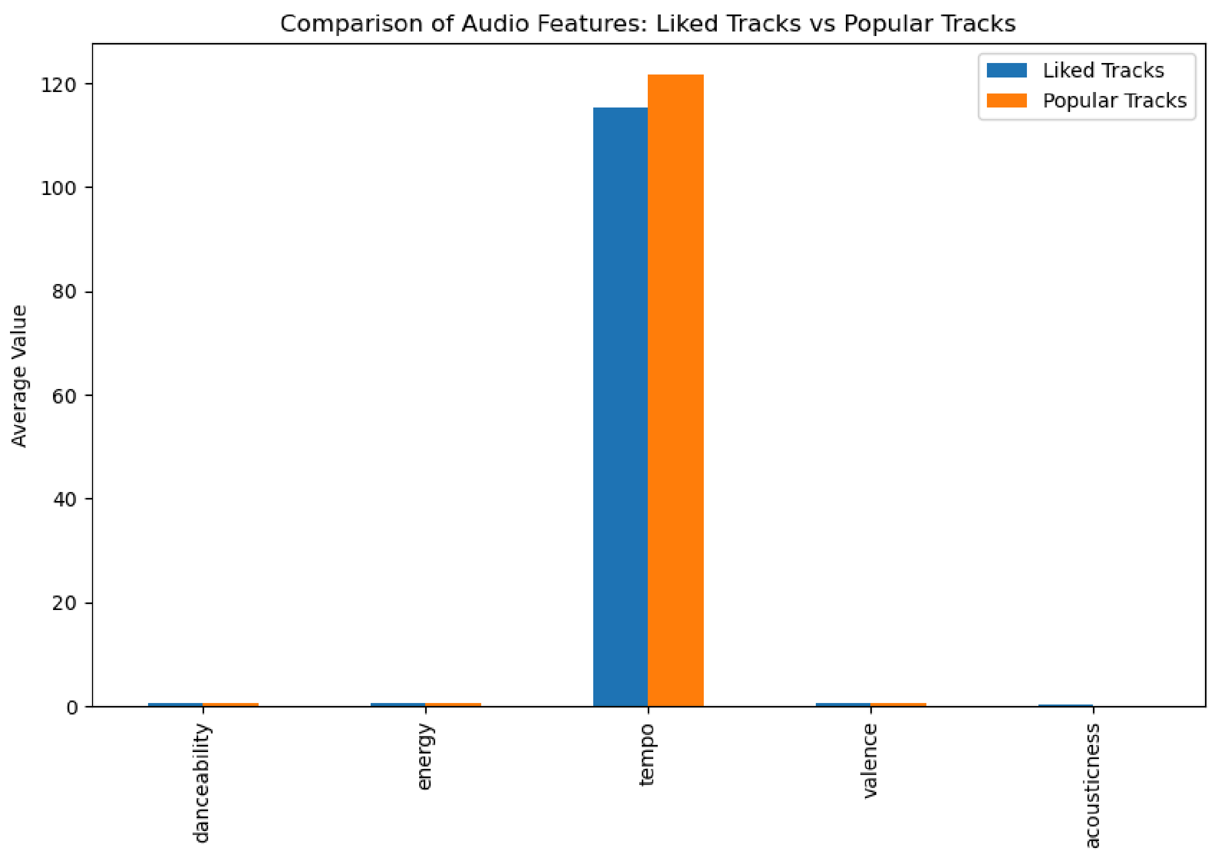
In [62]:
```python
# Compare audio features
liked_audio_features = df_songs[['danceability', 'energy', 'tempo', 'valence
popular_audio_features = df_popular_tracks[['danceability', 'energy', 'tempo
```

In [63]:
```
comparison = pd.DataFrame({'Liked Tracks': liked_audio_features, 'Popular Tr
print(comparison)
```

```
              Liked Tracks  Popular Tracks
danceability      0.644150        0.684980
energy            0.660034        0.635960
tempo           115.230394      121.647440
valence           0.476729        0.557940
acousticness      0.196261        0.179757
```

Visualizing the Comparison of Audio Features: Liked Tracks vs Popular Tracks

In [64]:
```
comparison.plot(kind='bar', figsize=(10, 6))
plt.title('Comparison of Audio Features: Liked Tracks vs Popular Tracks')
plt.ylabel('Average Value')
plt.show()
```



Get my user's ID

In [ ]:
```
user_id = sp.current_user()['id']
```

Create a new playlist called 'Shaun New Recommended Playlist'

In [66]:
```
playlist_name = 'Shaun New Recommended Playlist'
playlist_description = 'Playlist created based on refined recommendations us
new_playlist = sp.user_playlist_create(user_id, playlist_name, public=True,
playlist_id = new_playlist['id']

print(f"Playlist '{playlist_name}' created with ID: {playlist_id}")
```

```
Playlist 'Shaun New Recommended Playlist' created with ID: 3EW3ezaVF4iqacXYE
r0Yqr
```

Fetch track URIs for the recommended tracks

In [68]:
```
track_uris = refined_recommendations_adjusted['track_id'].apply(lambda x: f'
```

Add tracks to my new playlist

In [69]:
```
sp.playlist_add_items(playlist_id, track_uris)
print(f"Added {len(track_uris)} tracks to the playlist '{playlist_name}'.")
```

```
Added 5 tracks to the playlist 'Shaun New Recommended Playlist'.
```

WOO HOO!!