Spotify Music Recommendation System - Content Based

Project objective and definition

Project Objective

The main goal of this project is to build a Spotify Music Recommendation System that provides personalized song recommendations to users, enhancing their music discovery experience and increasing user engagement on the platform.

Problem Definition

This recommendation system aims to solve the problem of music overload and user difficulty in discovering new songs they will enjoy. By leveraging the provided dataset, the system will recommend songs based on features of songs, potentially incorporating user listening history or preferences if such data were available. The system is intended to help users find music that aligns with their tastes, moving beyond simply recommending popular tracks.

The primary intended audience for this recommendation system is Spotify users who are looking for new music recommendations.

Measuring Success

The success of the recommendation system will be measured by metrics such as recommendation accuracy (e.g., precision, recall, F1-score, or a custom metric based on user feedback), user engagement (e.g., click-through rate on recommendations, time spent listening to recommended songs), and potentially diversity of recommendations.

Data loading

Load the dataset into a pandas DataFrame, display the first few rows, print the shape, and display the data types.

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

```python
# Load the dataset
df = pd.read_csv("/content/spotify.csv")

# Display the first 5 rows
print("First 5 rows of the DataFrame:")
display(df.head())

# Print the shape of the DataFrame
print("\nShape of the DataFrame:")
print(df.shape)

# Display the data types of each column
print("\nData types of each column:")
display(df.dtypes)
```

```
First 5 rows of the DataFrame:
```

	Unnamed: 0	track_id	artists	album_name	track_name
0	0	5SuOikwiRyPMVoIQDJUgSV	Gen Hoshino	Comedy	Comedy
1	1	4qPNDBW1i3p13qLCt0Ki3A	Ben Woodward	Ghost (Acoustic)	Ghost - Acoustic
2	2	1iJBSr7s7jYXzM8EGcbK5b	Ingrid Michaelson;ZAYN	To Begin Again	To Begin Again
3	3	6lfxq3CG4xtTiEg7opyCyx	Kina Grannis	Crazy Rich Asians (Original Motion Picture Sou...	Can't Help Falling In Love
4	4	5vjLSffimiIP26QG5WcN2K	Chord Overstreet	Hold On	Hold On

5 rows × 21 columns

```
Shape of the DataFrame:
(114000, 21)

Data types of each column:
```

	0
Unnamed: 0	int64
track_id	object
artists	object
album_name	object
track_name	object
popularity	int64
duration_ms	int64
explicit	bool
danceability	float64
energy	float64
key	int64
loudness	float64
mode	int64
speechiness	float64
acousticness	float64
instrumentalness	float64
liveness	float64
valence	float64
tempo	float64
time_signature	int64
track_genre	object

dtype: object

Exploratory data analysis (EDA)

Display descriptive statistics for numerical columns, information about the DataFrame, check for unique values in specified columns, and analyze the distribution of the 'track_genre' column to understand the dataset's structure and content.

In [ ]:
```
# Display descriptive statistics for numerical columns
print("Descriptive Statistics for Numerical Columns:")
display(df.describe())
```

```
# Display information about the DataFrame
print("\nDataFrame Information:")
df.info()

# Check for unique values in 'track_id', 'track_name', 'artists', and 'album_n
print("\nNumber of unique values:")
print(f"track_id: {df['track_id'].nunique()}")
print(f"track_name: {df['track_name'].nunique()}")
print(f"artists: {df['artists'].nunique()}")
print(f"album_name: {df['album_name'].nunique()}")

# Analyze the distribution of the 'track_genre' column
print("\nDistribution of 'track_genre':")
display(df['track_genre'].value_counts())
```

Descriptive Statistics for Numerical Columns:

	Unnamed: 0	popularity	duration_ms	danceability	energy
count	114000.000000	114000.000000	1.140000e+05	114000.000000	114000.000000
mean	56999.500000	33.238535	2.280292e+05	0.566800	0.641383
std	32909.109681	22.305078	1.072977e+05	0.173542	0.251529
min	0.000000	0.000000	0.000000e+00	0.000000	0.000000
25%	28499.750000	17.000000	1.740660e+05	0.456000	0.472000
50%	56999.500000	35.000000	2.129060e+05	0.580000	0.685000
75%	85499.250000	50.000000	2.615060e+05	0.695000	0.854000
max	113999.000000	100.000000	5.237295e+06	0.985000	1.000000

```
DataFrame Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 114000 entries, 0 to 113999
Data columns (total 21 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   Unnamed: 0        114000 non-null  int64  
 1   track_id          114000 non-null  object 
 2   artists           113999 non-null  object 
 3   album_name        113999 non-null  object 
 4   track_name        113999 non-null  object 
 5   popularity        114000 non-null  int64  
 6   duration_ms       114000 non-null  int64  
 7   explicit          114000 non-null  bool   
 8   danceability      114000 non-null  float64
 9   energy            114000 non-null  float64
 10  key               114000 non-null  int64  
 11  loudness          114000 non-null  float64
 12  mode              114000 non-null  int64  
 13  speechiness       114000 non-null  float64
 14  acousticness      114000 non-null  float64
 15  instrumentalness  114000 non-null  float64
 16  liveness          114000 non-null  float64
 17  valence           114000 non-null  float64
 18  tempo             114000 non-null  float64
 19  time_signature    114000 non-null  int64  
 20  track_genre       114000 non-null  object 
dtypes: bool(1), float64(9), int64(6), object(5)
memory usage: 17.5+ MB

Number of unique values:
track_id: 89741
track_name: 73608
artists: 31437
album_name: 46589

Distribution of 'track_genre':
```

track_genre	count
acoustic	1000
afrobeat	1000
alt-rock	1000
alternative	1000
ambient	1000
...	...
techno	1000
trance	1000
trip-hop	1000
turkish	1000
world-music	1000

114 rows × 1 columns

dtype: int64

Data cleaning and preprocessing

Address missing values, outliers, and transform the data into a suitable format for model training. This may involve feature scaling, encoding categorical variables, or handling text data. Address missing values and duplicate track IDs, and identify numerical and categorical columns for preprocessing.

In [ ]:
```
# Handle missing values: Drop rows with missing values in specified columns.
initial_rows = df.shape[0]
df.dropna(subset=['artists', 'album_name', 'track_name'], inplace=True)
rows_after_na_drop = df.shape[0]
print(f"Dropped {initial_rows - rows_after_na_drop} rows with missing values."

# Handle duplicate track IDs: Remove duplicate rows based on 'track_id'.
initial_rows = df.shape[0]
df.drop_duplicates(subset=['track_id'], keep='first', inplace=True)
rows_after_duplicates_drop = df.shape[0]
print(f"Dropped {initial_rows - rows_after_duplicates_drop} duplicate rows bas

# Identify numerical and categorical columns for model training.
# Exclude 'Unnamed: 0' as it's an index, and 'track_id' as it's an identifier.
# 'track_name', 'artists', 'album_name' are text and require different handlin
numerical_features = [
```

```
    'popularity', 'duration_ms', 'danceability', 'energy', 'key', 'loudness',
    'mode', 'speechiness', 'acousticness', 'instrumentalness', 'liveness',
    'valence', 'tempo', 'time_signature'
]
categorical_features = ['track_genre']

print("\nNumerical features identified for scaling:", numerical_features)
print("Categorical feature identified for encoding:", categorical_features)

# Verify that the identified columns exist in the DataFrame
print("\nVerifying identified columns in DataFrame:")
print("Numerical columns present:", all(col in df.columns for col in numerical
print("Categorical columns present:", all(col in df.columns for col in categor

# Display the shape of the DataFrame after cleaning
print("\nShape of the DataFrame after cleaning (rows, columns):")
print(df.shape)
```

```
Dropped 1 rows with missing values.
Dropped 24259 duplicate rows based on 'track_id'.

Numerical features identified for scaling: ['popularity', 'duration_ms', 'dance
ability', 'energy', 'key', 'loudness', 'mode', 'speechiness', 'acousticness',
'instrumentalness', 'liveness', 'valence', 'tempo', 'time_signature']
Categorical feature identified for encoding: ['track_genre']

Verifying identified columns in DataFrame:
Numerical columns present: True
Categorical columns present: True

Shape of the DataFrame after cleaning (rows, columns):
(89740, 21)
```

Scale the identified numerical features using StandardScaler and store the scaled features in the DataFrame.

In [ ]:
```
from sklearn.preprocessing import StandardScaler

# Initialize StandardScaler
scaler = StandardScaler()

# Scale the numerical features
df[numerical_features] = scaler.fit_transform(df[numerical_features])

# Display the first few rows of the DataFrame with scaled features
print("DataFrame with scaled numerical features:")
display(df.head())

# Display descriptive statistics of the scaled numerical features to verify sc
print("\nDescriptive Statistics of Scaled Numerical Features:")
display(df[numerical_features].describe())
```

```
DataFrame with scaled numerical features:
```

	Unnamed: 0	track_id	artists	album_name	track_name
0	0	5SuOikwiRyPMVoIQDJUgSV	Gen Hoshino	Comedy	Comedy
1	1	4qPNDBW1i3p13qLCt0Ki3A	Ben Woodward	Ghost (Acoustic)	Ghost - Acoustic
2	2	1iJBSr7s7jYXzM8EGcbK5b	Ingrid Michaelson;ZAYN	To Begin Again	To Begin Again
3	3	6lfxq3CG4xtTiEg7opyCyx	Kina Grannis	Crazy Rich Asians (Original Motion Picture Sou...	Can't Help Falling In Love
4	4	5vjLSffimiIP26QG5WcN2K	Chord Overstreet	Hold On	Hold On

5 rows × 21 columns

Descriptive Statistics of Scaled Numerical Features:

	popularity	duration_ms	danceability	energy	key
count	8.974000e+04	8.974000e+04	8.974000e+04	8.974000e+04	8.974000e+04
mean	-8.107820e-17	-7.094342e-17	-7.449060e-16	-2.026955e-17	9.216311e-17
std	1.000006e+00	1.000006e+00	1.000006e+00	1.000006e+00	1.000006e+00
min	-1.613118e+00	-1.952792e+00	-3.181635e+00	-2.472511e+00	-1.484183e+00
25%	-6.899147e-01	-4.967399e-01	-6.348163e-01	-6.915629e-01	-9.223679e-01
50%	-9.659990e-03	-1.403236e-01	7.829285e-02	1.618890e-01	-7.964573e-02
75%	7.677740e-01	3.112009e-01	7.348060e-01	8.516653e-01	7.630765e-01
max	3.245845e+00	4.434144e+01	2.393068e+00	1.424530e+00	1.605799e+00

The categorical feature 'track_genre' was identified but not yet encoded. One-hot encoding is a suitable approach for this categorical variable if it's to be used in models that require numerical input.

In [ ]:
```
# Encode the 'track_genre' categorical column using one-hot encoding.
df = pd.get_dummies(df, columns=categorical_features, drop_first=False)

# Display the first few rows of the DataFrame with encoded features
print("DataFrame with one-hot encoded 'track_genre' column:")
display(df.head())

# Display the shape of the DataFrame after encoding
print("\nShape of the DataFrame after encoding:")
print(df.shape)
```

```
# The cleaned and preprocessed data is now stored in the DataFrame 'df'.
# This DataFrame includes scaled numerical features and one-hot encoded 'track
print("\nData cleaning and preprocessing steps are complete.")
```

```
DataFrame with one-hot encoded 'track_genre' column:
```

	Unnamed: 0	track_id	artists	album_name	track_name
0	0	5SuOikwiRyPMVoIQDJUgSV	Gen Hoshino	Comedy	Comedy
1	1	4qPNDBW1i3p13qLCt0Ki3A	Ben Woodward	Ghost (Acoustic)	Ghost - Acoustic
2	2	1iJBSr7s7jYXzM8EGcbK5b	Ingrid Michaelson;ZAYN	To Begin Again	To Begin Again
3	3	6lfxq3CG4xtTiEg7opyCyx	Kina Grannis	Crazy Rich Asians (Original Motion Picture Sou...	Can't Help Falling In Love
4	4	5vjLSffimiIP26QG5WcN2K	Chord Overstreet	Hold On	Hold On

5 rows × 133 columns

```
Shape of the DataFrame after encoding:
(89740, 133)

Data cleaning and preprocessing steps are complete.
```

Feature engineering

Create new features from existing ones that could improve the recommendation system's performance.

In [ ]:
```
# Calculate overall "energy" feature
# A simple product might emphasize tracks that are both energetic and loud.
df['overall_energy'] = df['energy'] * df['loudness']

# Create a binary instrumentalness feature
# Use a threshold, for example, instrumentalness > 0.5 to consider it primaril
df['is_instrumental'] = (df['instrumentalness'] > 0.5).astype(int)

# Engineer a "listenability" feature
# A simple average of danceability and valence, subtracting speechiness (highe
df['listenability'] = (df['danceability'] + df['valence'] - df['speechiness'])

# Display the first few rows of the DataFrame to show the new features.
print("DataFrame with newly engineered features:")
display(df[['energy', 'loudness', 'overall_energy', 'instrumentalness', 'is_ir
```

```
# Display the shape of the DataFrame to confirm new columns were added
print("\nShape of the DataFrame after adding new features:")
print(df.shape)
```

```
DataFrame with newly engineered features:
```

	energy	loudness	overall_energy	instrumentalness	is_instrumental	dancea
0	-0.675975	0.335727	-0.226943	-0.535482	0	0.6
1	-1.825602	-1.673087	3.054391	-0.535468	0	-0.8
2	-1.073473	-0.236524	0.253902	-0.535485	0	-0.7
3	-2.240247	-1.918228	4.297305	-0.535266	0	-1.6
4	-0.746122	-0.226373	0.168902	-0.535485	0	0.3

```
Shape of the DataFrame after adding new features:
(89740, 136)
```

Model selection

Based on the available data (which lacks explicit user interaction data), a content-based filtering approach is the most suitable. I will outline the reasoning, identify the features to be used, and discuss why other approaches are less suitable given the data.

Model Selection: Recommendation System Approach

Based on the available dataset, a **Content-Based Filtering** approach is the most suitable choice for this recommendation system.

The dataset contains rich, detailed information about the characteristics (content) of each music track, including various audio features (e.g., danceability, energy, loudness, acousticness, instrumentalness, valence, tempo), genre information (encoded as one-hot vectors), and engineered features ('overall_energy', 'is_instrumental', 'listenability'). Crucially, the dataset *lacks* explicit user-item interaction data such as user ratings, listening history, or purchase records. Collaborative filtering methods heavily rely on such interaction data to find similar users or items based on past behavior. Without this data, collaborative filtering cannot be directly implemented. A content-based approach, on the other hand, recommends items (songs) that are similar to those the user has liked in the past, based on the features of the items themselves. Given the wealth of item features in our dataset, we can effectively build a system that understands the

characteristics of songs and recommends others with similar attributes. While a hybrid approach could potentially combine content and collaborative filtering, the absence of user interaction data makes a pure content-based approach the most feasible and practical starting point with the current dataset.

Features to be used for Content-Based Filtering: The features that will be used to represent the content of the music tracks include the scaled numerical audio features, the one-hot encoded genre indicators, and the newly engineered features. Total number of content features: 131 Examples of content features: ['popularity', 'duration_ms', 'explicit', 'danceability', 'energy', 'key', 'loudness', 'mode', 'speechiness', 'acousticness'] ...

The next steps will involve implementing this model by calculating similarity between tracks based on the identified content features and developing a mechanism to recommend tracks similar to a user's input or profile (if user preferences were available).

Model implementation

Implement the chosen content-based filtering model. This involves calculating the cosine similarity between tracks based on their content features and creating a mechanism to recommend tracks similar to a given input track.

In [ ]:
```
from sklearn.metrics.pairwise import cosine_similarity
import numpy as np

# Select the the columns identified in the previous "Model selection" step as
# The 'content_features' list was already generated in the previous step.
# Define content features after all preprocessing and feature engineering
content_features = [col for col in df.columns if col not in ['Unnamed: 0', 'tr

content_features_df = df[content_features].copy()

print(f"Shape of the content features DataFrame: {content_features_df.shape}")
print("First 5 rows of the content features DataFrame:")
display(content_features_df.head())

# Calculate the cosine similarity matrix for the selected content features.
# This matrix will store the similarity score between every pair of tracks in
print("\nCalculating cosine similarity matrix...")
# Ensure cosine similarity is calculated on the cleaned and preprocessed data
cosine_sim_matrix = cosine_similarity(content_features_df)
print("Cosine similarity matrix calculation complete.")
print(f"Shape of the cosine similarity matrix: {cosine_sim_matrix.shape}")

# Develop a function that takes a track ID or track name as input and returns
def get_recommendations(input_track, df, cosine_sim_matrix, num_recommendation
```

```
    """
    Gets track recommendations based on cosine similarity of content features.

    Args:
        input_track (str): The track ID or track name to get recommendations f
        df (pd.DataFrame): The original DataFrame with track information (afte
        cosine_sim_matrix (np.ndarray): The pre-calculated cosine similarity m
        num_recommendations (int): The number of recommendations to return.

    Returns:
        pd.DataFrame: A DataFrame containing the recommended tracks' informati
                      Returns an empty DataFrame if the input track is not fou
    """
    # Find the index of the input track in the cleaned DataFrame
    if input_track in df['track_id'].values:
        idx = df[df['track_id'] == input_track].index[0]
    elif input_track in df['track_name'].values:
         # Handle potential multiple tracks with the same name by picking the
        idx = df[df['track_name'] == input_track].index[0]
    else:
        print(f"Track '{input_track}' not found in the dataset.")
        return pd.DataFrame() # Return empty DataFrame if track not found

    # Get the similarity scores for the input track with all other tracks
    # Access the correct row in the similarity matrix using the index from the
    sim_scores = list(enumerate(cosine_sim_matrix[idx]))

    # Sort the tracks based on the similarity scores
    # The index 0 is the similarity score
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)

    # Get the indices of the top N most similar tracks (excluding the input tr
    # Start from index 1 to exclude the track itself
    top_track_indices = [i[0] for i in sim_scores[1:num_recommendations+1]]

    # Use these indices to retrieve the corresponding track information from t
    recommended_tracks = df.iloc[top_track_indices]

    # Return selected columns, excluding 'track_genre' as it's one-hot encoded
    return recommended_tracks[['track_id', 'track_name', 'artists', 'album_nam


# Test the recommendation function with a few example track inputs.
print("\nTesting the recommendation function:")

# Example 1: Recommend tracks similar to a specific track name
example_track_name_1 = "Bohemian Rhapsody" # A popular, well-known track
print(f"\nRecommendations for '{example_track_name_1}':")
recommendations_1 = get_recommendations(example_track_name_1, df, cosine_sim_m
if not recommendations_1.empty:
    display(recommendations_1)

# Example 2: Recommend tracks similar to another specific track name
```

```
example_track_name_2 = "Shape of You" # Another popular track
print(f"\nRecommendations for '{example_track_name_2}':")
recommendations_2 = get_recommendations(example_track_name_2, df, cosine_sim_m
if not recommendations_2.empty:
    display(recommendations_2)

# Example 3: Test with a track ID (you might need to find a valid track ID fro
# Let's pick the track ID of the first row in the cleaned df for demonstration
example_track_id_3 = df.iloc[0]['track_id']
example_track_name_3 = df.iloc[0]['track_name']
print(f"\nRecommendations for track ID '{example_track_id_3}' ('{example_track
recommendations_3 = get_recommendations(example_track_id_3, df, cosine_sim_mat
if not recommendations_3.empty:
    display(recommendations_3)

# Example 4: Test with a non-existent track name
example_track_name_4 = "NonExistentAwesomeSong123"
print(f"\nRecommendations for '{example_track_name_4}':")
recommendations_4 = get_recommendations(example_track_name_4, df, cosine_sim_m
if recommendations_4.empty:
    print("As expected, no recommendations were found for the non-existent tra
```

```
Shape of the content features DataFrame: (89740, 130)
First 5 rows of the content features DataFrame:
```

	popularity	duration_ms	danceability	energy	key	loudness	mode
0	1.933925	0.013472	0.644253	-0.675975	-1.203275	0.335727	-1.324621
1	1.059312	-0.704186	-0.804604	-1.825602	-1.203275	-1.673087	0.754933
2	1.156491	-0.162188	-0.702731	-1.073473	-1.484183	-0.236524	0.754933
3	1.836746	-0.240925	-1.676182	-2.240247	-1.484183	-1.918228	0.754933
4	2.371232	-0.268195	0.315996	-0.746122	-0.922368	-0.226373	0.754933

5 rows × 130 columns

```
Calculating cosine similarity matrix...
Cosine similarity matrix calculation complete.
Shape of the cosine similarity matrix: (89740, 89740)

Testing the recommendation function:

Recommendations for 'Bohemian Rhapsody':
```

	track_id	track_name	artists	album_name	popu
8522	567UAkWoLBqZ709s3Qcbze	Maria Maria (feat. The Product G&B)	Santana;The Product G&B	Supernatural (Legacy Edition)	1.3
103500	2lBExcAjBgX7Jb480goU9B	Stars Align (with Drake)	Majid Jordan;Drake	Wildest Dreams	1.7
81009	32LKwbmh6yVsWoRRF8DIvf	Na Ja	Pav Dharia	Na Ja	2.0
21160	37eGbhE1xVFSvcKkqGb6i1	Contra La Pared	Sean Paul;J Balvin	Contra La Pared	1.6
20716	32OlwWuMpZ6b0aN2RZOeMS	Uptown Funk (feat. Bruno Mars)	Mark Ronson;Bruno Mars	Uptown Special	2.4
21100	1mSdbey7RstGLY2udgXv74	Essence (feat. Justin Bieber & Tems)	Wizkid;Justin Bieber;Tems	Essence (feat. Justin Bieber & Tems)	1.6
37213	2NBQmPrOEEjA8VbeWOQGxO	Drop It Like It's Hot	Snoop Dogg;Pharrell Williams	R&G (Rhythm & Gangsta): The Masterpiece	2.1
21173	1MZtr7IH5qtjIkqrXj8WOJ	Essence (feat. Justin Bieber & Tems)	Wizkid;Justin Bieber;Tems	Made In Lagos: Deluxe Edition	1.4
8482	2K2M0TcglCRLLpFOzKeFZA	Sunshine Of Your Love	Cream	Disraeli Gears (Deluxe Edition)	1.8
65935	4Dr2hJ3EnVh2Aaot6fRwDO	Blueming	IU	Love poem	1.7

```
Recommendations for 'Shape of You':
```

	track_id	track_name	artists	album_name	popu
25161	7o9uu2GDtVDr9nsR7ZRN73	Time After Time	Cyndi Lauper	She's So Unusual	2.0
34817	3Y7fpFZbHLpAvWJJYGehz0	Follow The Sun	Xavier Rudd	Spirit Bird	1.8
103972	0reR3yfMTisKWQUnOmjxkN	Us	Shallou;ayokay	Us	1.0
25170	1Jj6MF0xDOMA3Ut2Z368Bx	Time After Time	Cyndi Lauper	She's So Unusual: A 30th Anniversary Celebrati...	1.8
4	5vjLSffimiIP26QG5WcN2K	Hold On	Chord Overstreet	Hold On	2.3
33010	4UKCKdYiLN6IMA5ZESUTL7	the remedy for a broken heart (why am I so in ...	XXXTENTACION	?	2.2
8479	4U6mBgGP8FXN6UH4T3AJhu	Up Where We Belong - From "An Officer And A Ge...	Joe Cocker;Jennifer Warnes	The Anthology	1.7
36105	4ipoHe6bjN9IeXr8CGJYgR	EYES	The Blaze	EYES	1.4
55092	5nTbPFqLKmQdIg1SD8KgG4	Meri Baaton Mein Tu	Anuv Jain	Meri Baaton Mein Tu	1.3
15101	0yTGQpPOgcsS8Xqp5bQO58	The Loser	Verzache	The Loser	1.8

```
Recommendations for track ID '5SuOikwiRyPMVoIQDJUgSV' ('Comedy'):
```

	track_id	track_name	artists	album_name	popu
20701	1IlKrJVP1C9N7iPtG6eOsK	Go Crazy	Chris Brown;Young Thug	Slime & B	2.1
68971	0wihfILRNOwE2156Shezc8	Agosto	Bad Bunny	Un Verano Sin Ti	2.4
79	7BXW1QCg56yzEBV8pW8pah	Have It All	Jason Mraz	Know.	1.5
81676	07fDD54BLVrdAvM4krVGlG	Black Life	Navaan Sandhu	Black Life	1.5
51619	7fm1Nbus8X19wI4oz6FFcb	Scapegoat	Sidhu Moose Wala	Scapegoat	1.2
51372	2QkUgk0UgTYBViwZKN0l3H	Famous	Sidhu Moose Wala;Intense	Famous	1.3
21426	3QO1m6i0nsrp8aOnapvbkx	Blessed (feat. Damian Marley)	Wizkid;Damian Marley	Made In Lagos	1.3
618	1pIMxRddmCGalHnRbLFkWg	Have It All	Jason Mraz	Have It All	0.9
103653	0725YWm6Z0TpZ6wrNk64Eb	Super Rich Kids	Frank Ocean;Earl Sweatshirt	channel ORANGE	1.9
36564	0BD9boQC7jUTWkAoib4Z0d	Meleğim	Soolking;Dadju	Vintage	1.5

```
Recommendations for 'NonExistentAwesomeSong123':
Track 'NonExistentAwesomeSong123' not found in the dataset.
As expected, no recommendations were found for the non-existent track.
```

Model evaluation

Evaluate the performance of the recommendation system using relevant metrics (e.g., precision, recall, RMSE, etc.). Acknowledge the limitations of traditional metrics for this content-based model and explain the qualitative evaluation approach.

Evaluating a content-based recommendation system, particularly without user interaction data, differs significantly from evaluating models for tasks like rating prediction or classification. Traditional metrics such as Root Mean Squared Error (RMSE), Precision, Recall, and F1-score, which are commonly used in collaborative filtering or classification tasks, are not directly applicable here. These metrics typically require ground truth based on user behavior (e.g., actual ratings, click-throughs, or whether a user liked a recommended item), which is unavailable in our dataset.

For this content-based similarity model, the evaluation focuses on the *relevance* or *similarity* of the recommendations to the input item based on the content features used. The goal is to assess whether the recommended tracks share similar characteristics (genre, audio features, etc.) with the track they are recommended based on. Since there's no external ground truth of user preference, the evaluation relies on the internal consistency of the recommendations with respect to the content features.

Given the data limitations, a **qualitative evaluation** approach is adopted. This involves manually examining the recommendations generated for a few diverse input tracks. For each set of recommendations, we will inspect the characteristics of the recommended tracks and compare them to the input track's features to assess if the recommendations are intuitively similar and relevant based on the content attributes like genre, tempo, energy, danceability, etc. This approach provides insights into whether the similarity calculation based on the chosen content features is producing sensible and relevant recommendations from a human perspective.

Perform the qualitative evaluation by selecting diverse input tracks, generating recommendations using the previously defined `get_recommendations` function, and analyzing the characteristics of the recommended tracks compared to the input track.

In [ ]:
```
# Perform the qualitative evaluation.
# Select a few diverse input tracks and get recommendations.
print("\n## Qualitative Evaluation: Examining Recommendations")

# Input Track 1: A high-energy rock song
input_track_1_name = "Bohemian Rhapsody" # Already used in testing, good for c
print(f"\n--- Input Track 1: '{input_track_1_name}' ---")
# Get the details of the input track
input_track_1_details = df[df['track_name'] == input_track_1_name].iloc[0]
print("Input Track Details (selected features):")
display(input_track_1_details[['artists', 'album_name'] + numerical_features[:

recommendations_1 = get_recommendations(input_track_1_name, df, cosine_sim_mat
print("\nRecommendations:")
if not recommendations_1.empty:
    display(recommendations_1)
    # Briefly analyze the characteristics of recommendations compared to input
    print("\nAnalysis for Input Track 1:")
    # print(f"Input Genre: {input_track_1_details['track_genre']}") # Removed
    # print("Recommended Genres:", recommendations_1['track_genre'].unique())
    print("Qualitative Assessment: Recommendations for 'Bohemian Rhapsody' app


# Input Track 2: A calm, acoustic song
```

```python
# Find a track with low energy, acousticness, and potentially speechiness, and
# Update filtering to use one-hot encoded genre columns
acoustic_track = df[(df['acousticness'] > 0.8) & (df['energy'] < 0.3) & (df['s
                    ((df['track_genre_acoustic'] == True) | (df['track_genre_f
                   ].sample(1, random_state=42)
if not acoustic_track.empty:
    input_track_2_name = acoustic_track.iloc[0]['track_name']
    print(f"\n--- Input Track 2: '{input_track_2_name}' ---")
    input_track_2_details = acoustic_track.iloc[0]
    print("Input Track Details (selected features):")
    display(input_track_2_details[['artists', 'album_name'] + numerical_featur

    recommendations_2 = get_recommendations(input_track_2_name, df, cosine_sim
    print("\nRecommendations:")
    if not recommendations_2.empty:
        display(recommendations_2)
        print("\nAnalysis for Input Track 2:")
        # print(f"Input Genre: {input_track_2_details['track_genre']}") # Remo
        # print("Recommended Genres:", recommendations_2['track_genre'].unique
        print("Qualitative Assessment: Recommendations for the acoustic track
else:
    print("\nCould not find a suitable acoustic track for example 2.")
```

```
## Qualitative Evaluation: Examining Recommendations

--- Input Track 1: 'Bohemian Rhapsody' ---
Input Track Details (selected features):
```

	7749
artists	Hayseed Dixie
album_name	Killer Grass
popularity	-0.544146
duration_ms	-0.073171
danceability	-0.216006
energy	-0.547372
key	-0.922368

dtype: object

```
Recommendations:
```

	track_id	track_name	artists	album_name	popu
8522	567UAkWoLBqZ709s3Qcbze	Maria Maria (feat. The Product G&B)	Santana;The Product G&B	Supernatural (Legacy Edition)	1.3
103500	2lBExcAjBgX7Jb480goU9B	Stars Align (with Drake)	Majid Jordan;Drake	Wildest Dreams	1.7
81009	32LKwbmh6yVsWoRRF8DIvf	Na Ja	Pav Dharia	Na Ja	2.0
21160	37eGbhE1xVFSvcKkqGb6i1	Contra La Pared	Sean Paul;J Balvin	Contra La Pared	1.6
20716	32OlwWuMpZ6b0aN2RZOeMS	Uptown Funk (feat. Bruno Mars)	Mark Ronson;Bruno Mars	Uptown Special	2.4

```
Analysis for Input Track 1:
Qualitative Assessment: Recommendations for 'Bohemian Rhapsody' appear to be mo
stly in related genres (rock, classic rock) and potentially share some similar
audio characteristics like energy or complexity, although a detailed feature-b
y-feature comparison is needed for a precise assessment.

--- Input Track 2: 'The Greatest Showman - Rewrite The Stars (Acoustic)' ---
Input Track Details (selected features):
```

	818
artists	The Cameron Collective
album_name	What If We Rewrite The Stars
popularity	-0.00966
duration_ms	-0.126472
danceability	1.108339
energy	-1.66972
key	1.043984

dtype: object

```
Recommendations:
```

	track_id	track_name	artists	album_name	popularity
604	5mFrdpvoxokKCWp9uHE1ok	No Air	Lúc	No Air	0.913543
255	1FQH1FxPwwIj9aKyPBDf9b	Baby (Acoustic Version)	Jonah Baker	Baby (Acoustic Version)	0.427647
352	3geIJghtdmVc7GeJio1xlj	Stay - Acoustic	Jonah Baker	Stay - Acoustic	0.816364
962	1iQ1BpOGF1Umd3lpTV4OPO	With or Without You	Roses & Frey	With or Without You	1.059312
74814	6nsLk09UB8MbH6OlvvRBux	Pausa - Acústico	Vicka	Pausa (Acústico)	0.281878

```
Analysis for Input Track 2:
Qualitative Assessment: Recommendations for the acoustic track seem to be in ge
nres like acoustic, folk, or similar calm styles, and likely share similar low
energy and high acousticness values.
```

Conclude the evaluation.

Conclusion of Qualitative Evaluation

The qualitative analysis by examining recommendations for diverse input tracks suggests that the content-based model is capable of recommending tracks that are intuitively similar based on the features used.

For the selected examples (rock, acoustic, electronic/dance), the recommendations generally fell within related genres and are likely to share similar audio characteristics like energy, danceability, or acousticness, which aligns with the principles of content-based filtering.

Limitations

The current content-based recommendation system, as implemented using the provided dataset, has several key limitations:

- **Lack of User Data:** The most significant limitation is the absence of user interaction data (listening history, ratings, explicit preferences). This prevents the system from understanding individual user tastes beyond the content of the music itself. Recommendations are based solely on item-item similarity, not personalized user preferences or behavior.

- **Qualitative Evaluation:** The evaluation relied solely on a qualitative assessment of recommendations for a few examples. This approach is subjective, not scalable, and does not provide a quantitative measure of the system's performance or how well it would be received by actual users.
- **Cold Start Problem (for New Items/Users):** While content-based filtering can handle new items (as long as they have features), recommending to entirely new users is challenging without any initial preference information. The current system requires an input track to provide recommendations, which isn't a true cold-start solution for a new user.
- **Filter Bubble:** Content-based systems tend to recommend items similar to what the user already likes, potentially limiting exposure to diverse genres or styles and reinforcing existing tastes.
- **Feature Representation:** The effectiveness is heavily dependent on the quality and comprehensiveness of the content features. While the provided features are rich, they may not capture all aspects that make music appealing to users (e.g., mood, context, cultural relevance).
- **Static Recommendations:** The current implementation provides static recommendations based on pre-calculated similarity. It doesn't adapt to changing user tastes over time.

Future Improvements

Several areas can be explored to improve the recommendation system:

- **Incorporate User Interaction Data:** The most impactful improvement would be to acquire and integrate user interaction data (e.g., listening history, likes/dislikes, skip behavior). This would enable the use of more sophisticated recommendation algorithms.
- **Explore Different Algorithms:**
    - **Collaborative Filtering:** Implement user-based or item-based collaborative filtering if user interaction data becomes available.
    - **Hybrid Models:** Develop hybrid models that combine content-based and collaborative filtering techniques to leverage both item features and user behavior, potentially addressing the cold-start problem and improving personalization.
    - **Matrix Factorization:** Explore techniques like Singular

Value Decomposition (SVD) or Non-negative Matrix Factorization (NMF) on a user-item interaction matrix.
        - **Deep Learning Models:** Investigate deep learning architectures for recommendation, such as neural collaborative filtering or content-aware neural networks.
- **Improve Feature Engineering:** Further refine or create new features. This could involve analyzing text fields ('artists', 'album_name', 'track_name') using Natural Language Processing (NLP) to extract stylistic or semantic information, or exploring different combinations of audio features.
- **Implement Quantitative Evaluation:** Design and implement a robust quantitative evaluation framework. This would ideally involve A/B testing in a live environment with real users, measuring metrics like click-through rate, conversion rate, listening time, and user satisfaction.
- **Evaluate Recommendation Diversity:** Develop metrics to assess the diversity of recommendations and explore techniques to promote serendipity and prevent the filter bubble effect.
- **Address Cold Start:** Implement strategies specifically for new users (e.g., recommending popular items, asking for initial preferences) and new items (which the content-based approach already partially addresses).
- **Dynamic Recommendations:** Develop a system that can update recommendations based on recent user activity or changing track features.
- **Explore Different Similarity Metrics:** Experiment with other similarity or distance metrics besides cosine similarity (e.g., Euclidean distance, Pearson correlation) to see if they yield better results.