Research Questions

Genre vs Audio Features
Q1 ○ Tash

Top Songs vs Audio Features
Q2 ○ Riona

Album Popularity vs Track Features
Q3 ○ Sri

User Top Tracks vs Daily Mixes
Q4 ○ Mark

Deep Dive

into Spotify®

Research Questions

Genre vs Audio Features
Q1 ○ Tash

Top Songs vs Audio Features
Q2 ○ Riona

Album Popularity vs Track Features
Q3 ○ Sri

User Top Tracks vs Daily Mixes
Q4 ○ Mark

Deep Dive

into

Genre vs Audio Features

Q1 ○ Tash

Genre vs Audio Features: **genre with top audio feature**

6 chosen genres:
Country, Death Metal, Drum and Bass, Hip-Hop, Pop, and **Rock**

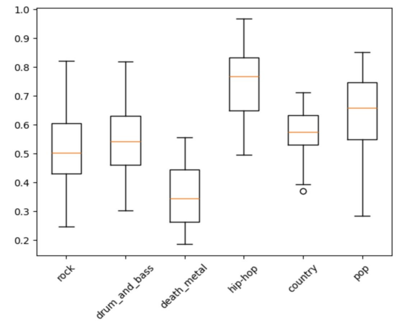
Genre by Mean Audio Feature

Danceability: Hip Hop
Energy: Death Metal
Loudness: Drum and Bass
Acousticness: Country
Instrumentalness: Death Metal
Liveness: Death Metal
Valence: Hip-Hop
Tempo: Drum and Bass

Genre	Danceability	Energy	Loudness	Acousticness	Instrumentalness	Liveness	Valence	Tempo
country	0.56784	0.57468	-7.28798	0.351710	0.001955	0.146506	0.474560	119.95554
death_metal	0.35016	0.93722	-5.76456	0.000344	0.413084	0.228660	0.268878	118.39464
drum_and_bass	0.55230	0.87058	-3.14776	0.049614	0.098785	0.215868	0.336012	139.86554
hip-hop	0.74594	0.66716	-6.66886	0.176523	0.002311	0.215512	0.557440	114.93276
pop	0.63748	0.68478	-5.01192	0.164444	0.000129	0.160110	0.519116	115.83242
rock	0.51538	0.75504	-7.07282	0.101358	0.031590	0.170108	0.532900	128.62016

Distribution for Danceability

Distribution of the Danceability audio feature is much higher for Hip-Hop



Genre vs Audio Features: **genre with top audio feature**

Defining a function to call the API as each request requires us to use credentials AND return an access token to grab playlist data

```
# Setup get requests URL and authentation.
# This auto-generates an access token using your API Client ID and Client Secret ea
api_url = "https://api.spotify.com/v1/"
search_url = api_url + "search"

def authenticate_spotify_client(client_ID, client_secret):
    auth_url = "https://accounts.spotify.com/api/token"
    auth_response = requests.post(auth_url, {
        'grant_type': 'client_credentials',
        'client_id': client_ID,
        'client_secret': client_secret,
      })
    auth_response_data = auth_response.json()
    access_token = auth_response_data['access_token']
    return access_token

access_token = authenticate_spotify_client(client_ID, client_secret)
```

```
# Define an input to generate playlist track data
def search_playlist(playlist_id):
    playlist_api = "https://api.spotify.com/v1/playlists/"
    playlist_url = playlist_api + playlist_id +f"/tracks"
    headers = {
        'Authorization': 'Bearer {token}'.format(token=access_token)
    }
    params = {
        'market': 'AU'
    }
    response = requests.get(playlist_url, headers=headers, params=params)
    data = response.json()
    return data["items"]
```

Defining the chosen playlists using Spotify's playlist IDs

```
# Save a dictionary of playlist ID to call from Spotify API
playlist_id_data = {
    "rock": "37i9dQZF1EQpj7X7UK8OOF",
    "drum_and_bass": "37i9dQZF1EIherXksVvnrN",
    "death_metal": "37i9dQZF1EIf78r65WuXwA",
    "hip-hop": "37i9dQZF1EQnqst5TRi17F",
    "country": "37i9dQZF1EQmPV0vrce2QZ",
    "pop": "37i9dQZF1EQncLwOalG3K7"
}
```

Genre vs Audio Features: **genre with top audio feature**

Iterating through each track in each playlist to grab the audio feature information and appending this to a dataframe.
Then, created a new dataframe that has calculated the mean for each genre.

```python
# Create a variable to get audio features for each track ID  from the Spotify API
track_ID = audio_analysis_summary["Track_ID"]

# Create empty list for audio features data
track_audio_analysis = {"Track_ID": [],
                        "danceability": [],
                        "energy": [],
                        "loudness": [],
                        "acousticness": [],
                        "instrumentalness": [],
                        "liveness": [],
                        "valence": [],
                        "tempo": []
                       }

# Save audio features to the list
for track in track_ID:
    track_audio_data = search_track(track)
    track_audio_analysis["Track_ID"].append(track)
    track_audio_analysis["danceability"].append(track_audio_data["danceability"])
    track_audio_analysis["energy"].append(track_audio_data["energy"])
    track_audio_analysis["loudness"].append(track_audio_data["loudness"])
    track_audio_analysis["acousticness"].append(track_audio_data["acousticness"])
    track_audio_analysis["instrumentalness"].append(track_audio_data["instrumentalness"])
    track_audio_analysis["liveness"].append(track_audio_data["liveness"])
    track_audio_analysis["valence"].append(track_audio_data["valence"])
    track_audio_analysis["tempo"].append(track_audio_data["tempo"])
```

	Genre	Playlist_ID	Track_ID	danceability	energy	loudness	acousticness	instrumentalness
0	rock	37i9dQZF1EQpj7X7UK8OOF	4BP3uh0hFLFRb5cjsgLqDh	0.640	0.663	-7.516	0.20100	0.00806
1	rock	37i9dQZF1EQpj7X7UK8OOF	1JSTJqkT5qHq8MDJnJbRE1	0.820	0.452	-9.796	0.54300	0.00294
2	rock	37i9dQZF1EQpj7X7UK8OOF	60a0Rd6pjrkxjPbaKzXjfq	0.556	0.864	-5.870	0.00958	0.00000

```python
# Average value for tracks of each genre
audio_mean = pd.DataFrame({"Danceability": track_analysis.groupby(["Genre"])["danceability"].mean(),
                           "Energy": track_analysis.groupby(["Genre"])["energy"].mean(),
                           "Loudness": track_analysis.groupby(["Genre"])["loudness"].mean(),
                           "Acousticness": track_analysis.groupby(["Genre"])["acousticness"].mean(),
                           "Instrumentalness": track_analysis.groupby(["Genre"])["instrumentalness"].mean(),
                           "Liveness": track_analysis.groupby(["Genre"])["liveness"].mean(),
                           "Valence": track_analysis.groupby(["Genre"])["valence"].mean(),
                           "Tempo": track_analysis.groupby(["Genre"])["tempo"].mean()
                          })

audio_mean.head(3)
```

Genre	Danceability	Energy	Loudness	Acousticness	Instrumentalness	Liveness	Valence	Tempo
country	0.56784	0.57468	-7.28798	0.351710	0.001955	0.146506	0.474560	119.95554
death_metal	0.35016	0.93722	-5.76456	0.000344	0.413084	0.228660	0.268878	118.39464

Genre vs Audio Features: **genre with the most variance in audio features**

The following analysis identified the genre with the most variance in their audio features; concluding which genre of music is the most diverse.

Genre and Total Variance in Audio Features

Linear Scale (0.00 to 1.00): Danceability, Energy, Acousticness, Instrumentalness, Liveness, Valence.

Db: Loudness

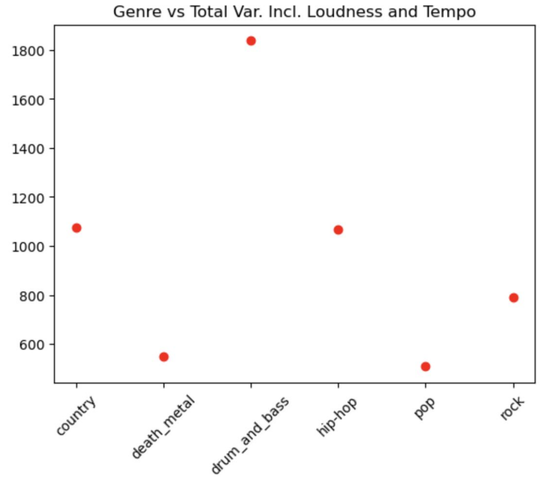
BPM: Tempo

	Genre	Total Var. Incl. Loudness and Tempo	Rank Incl. Loudness and Tempo	Total Var. Excl. Loudness and Tempo	Rank Excl. Loudness and Tempo
0	country	1075.244994	2	0.166542	2
1	death_metal	548.593037	5	0.203888	1
2	drum_and_bass	1836.976728	1	0.140738	4
3	hip-hop	1069.295923	3	0.155203	3
4	pop	507.444116	6	0.119054	6
5	rock	790.422618	4	0.127796	5

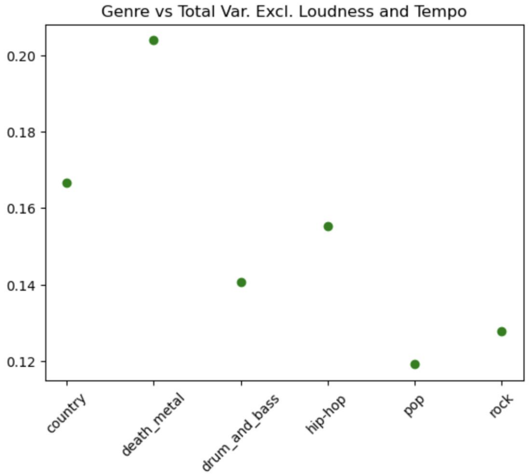
$$Var(X + Y) = Var(X) + Var(Y)$$

Genre vs Audio Features: **genre with the most variance in audio features**

Genre vs. Total Var. Incl. Loudness and Tempo



Genre vs. Total Var. Excl. Loudness and Tempo



Genre vs Audio Features: **genre with the most variance in audio features**

Calculate variance for genre’s audio feature

```python
# Calculating variance of
genre_variance_data = pd.DataFrame({"Danceability": track_analysis.groupby(["Genre"])["danceability"].var(),
                        "Energy": track_analysis.groupby(["Genre"])["energy"].var(),
                        "Loudness": track_analysis.groupby(["Genre"])["loudness"].var(),
                        "Acousticness": track_analysis.groupby(["Genre"])["acousticness"].var(),
                        "Instrumentalness": track_analysis.groupby(["Genre"])["instrumentalness"].var(),
                        "Liveness": track_analysis.groupby(["Genre"])["liveness"].var(),
                        "Valence": track_analysis.groupby(["Genre"])["valence"].var(),
                        "Tempo": track_analysis.groupby(["Genre"])["tempo"].var()
                       })

genre_variance_data.head(3)
```

	Danceability	Energy	Loudness	Acousticness	Instrumentalness	Liveness	Valence	Tempo
Genre								
country	0.006855	0.040638	6.543827	6.653024e-02	0.000065	0.010584	0.041868	1068.534625
death_metal	0.009741	0.005576	7.919430	7.726111e-07	0.139183	0.025490	0.023897	540.469719
drum_and_bass	0.015273	0.009694	3.717555	5.828304e-03	0.040286	0.027252	0.042405	1833.118435

Calculating total variance: Var(X + Y) = Var(X) + Var(Y) and assigning rank

```python
# Calculate total variance for each genre. Note that Var(X + Y) = Var(X) + Var(Y)
n = 0
total = 0
for genre in genre_variance1:
    mask = genre_reset.loc[genre_reset["Genre"] == genre]
    for column in columns1:
        total = total + mask[column][n]
        genre_variance1[genre] = total
    total = 0
    n = n + 1

# create a new dataframe
genre_variance1 = pd.DataFrame(genre_variance1.items(),columns=["Genre", "Total Var. Incl. Loudness and Tempo"])

# assign a rank to the new variance summary
genre_variance1["Rank Incl. Loudness and Tempo"] = genre_variance1.sort_values(by=["Total Var. Incl. Loudness and Tempo"], ascending=False) \
               .reset_index() \
               .sort_values("index") \
               .index + 1
```

Research Questions

Genre vs Audio Features
Q1 ○ Tash

Top Songs vs Audio Features
Q2 ○ Riona

Album Popularity vs Track Features
Q3 ○ Sri

User Top Tracks vs Daily Mixes
Q4 ○ Mark

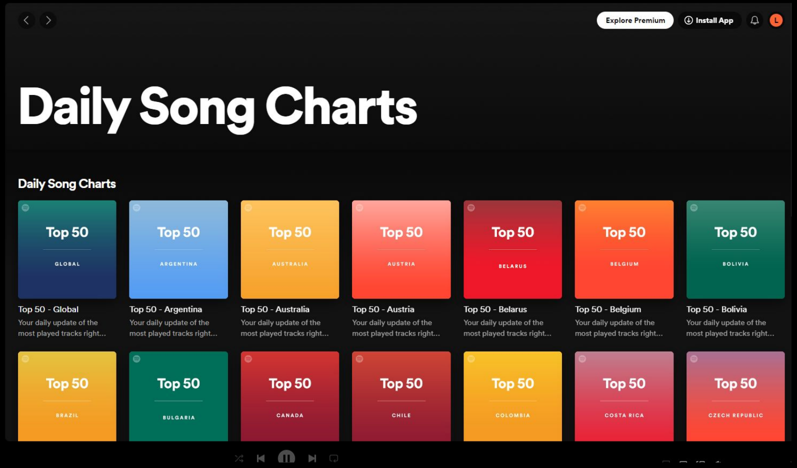
Deep Dive

into

Top Songs vs Audio Features

Q2 ○ Riona

Top Songs vs Audio Features : CODE DEMO




```
#create empty lists to show data
countries = []
tracks = []
artists = []

for country, playlist_id in playlist_ids.items():
    url = f"https://api.spotify.com/v1/playlists/{playlist_id}/tracks"
    headers = {"Authorization": f"Bearer {access_token}"}
    response = requests.get(url, headers=headers)
    if response.status_code == 200:
        # Extract track information from the response JSON
        tracks_data = response.json()["items"]
        for track in tracks_data:
            # Extract the name of the track
            track_name = track["track"]["name"]

            # Extract the name(s) of the artist(s)
            artist_names = [artist["name"] for artist in track["track"]["artists"]]

            # Append the country name, track name, and artist name(s) to their respective lists
            countries.append(country)
            tracks.append(track_name)
            artists.append(artist_names)

#create a dataframe with the columns country, track and , artist
data = {
    'Country': countries,
    'Track': tracks,
    'Artist': artists
}

df = pd.DataFrame(data)
#show dataframe
df
```

Top Songs vs Audio Features : CODE DEMO

```python
#visualisation 1 (print out the each tracks audio features)
import pprint

def get_audio_features(track_ids, access_token):
    headers = {
        'Authorization': f'Bearer {access_token}'
    }
    # Flatten the list of lists
    flat_track_ids = [track_id for sublist in track_ids for track_id in sublist]
    params = {
        'ids': ','.join(flat_track_ids)
    }
    url = 'https://api.spotify.com/v1/audio-features'
    response = requests.get(url, headers=headers, params=params)
    if response.status_code == 200:
        return response.json()['audio_features']
    else:
        return None

#copy track ids
track_ids = [['2GxrNKugF82CnoRFbQfzPf', '6tNQ70jh4OwmPGpYy6R2o9', '3qhlB30KknSejmIvZZLjOD', '3rUGC1vUpkDG9CZFHMur1t', '6XjDI
```

	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	type
0	0.599	0.946	11	-4.263	1	0.0447	0.000938	0.010600	0.0826	0.747	151.647	audio_features
1	0.472	0.471	10	-5.692	1	0.0603	0.151000	0.000000	0.1400	0.219	105.029	audio_features
2	0.689	0.454	2	-7.643	1	0.0584	0.035100	0.002590	0.0707	0.912	159.982	audio_features
3	0.750	0.733	6	-3.180	0	0.0319	0.256000	0.000000	0.1140	0.844	111.018	audio_features
4	0.791	0.499	8	-8.472	0	0.0509	0.446000	0.000024	0.0899	0.669	99.986	audio_features
5	0.741	0.620	10	-5.505	1	0.0412	0.029500	0.000809	0.0398	0.934	117.038	audio_features
6	0.647	0.650	5	-8.278	1	0.0421	0.071600	0.000016	0.0749	0.281	115.853	audio_features
7	0.561	0.604	9	-4.409	1	0.0337	0.199000	0.000019	0.1040	0.242	159.920	audio_features
8	0.679	0.772	8	-4.721	0	0.0472	0.027700	0.000003	0.1060	0.709	126.964	audio_features
9	0.594	0.811	1	-5.746	1	0.1590	0.189000	0.000000	0.3390	0.311	148.144	audio_features

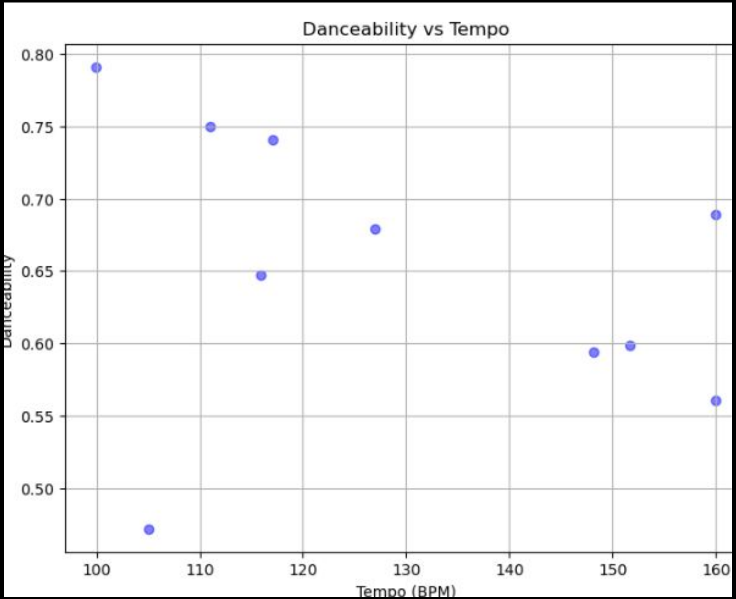
Top Songs vs Audio Features : CODE DEMO

```python
#visualisation 1 (print out the each tracks audio features)
audio_features = [af for af in audio_features if af is not None]
tempos = [af['tempo'] for af in audio_features]
danceabilities = [af['danceability'] for af in audio_features]
```

	Track	Artist	Tempo	Danceability
0	i like the way you kiss me	Artemas	151.647	0.599
1	Beautiful Things	Benson Boone	105.029	0.472
2	End of Beginning	Djo	159.982	0.689
3	greedy	Tate McRae	111.018	0.750
4	Gata Only	FloyyMenor, Cris Mj	99.986	0.791
5	Too Sweet	Hozier	117.038	0.741
6	we can't be friends (wait for your love)	Ariana Grande	115.853	0.647
7	Lose Control	Teddy Swims	159.920	0.561
8	Illusion	Dua Lipa	126.964	0.679
9	CARNIVAL	¥, *KanyeWest*, *TyDolla*ign, Rich The Kid, P...	148.144	0.594

Top Songs vs Audio Features : Analysing The Most Frequent Top 10s Danceability vs Tempo

No obvious Tempo Danceability trend



Different Tempos and Danceability Between the Top 10

	Track	Artist	Tempo	Danceability
0	i like the way you kiss me	Artemas	151.647	0.599
1	Beautiful Things	Benson Boone	105.029	0.472
2	End of Beginning	Djo	159.982	0.689
3	greedy	Tate McRae	111.018	0.750
4	Gata Only	FloyyMenor, Cris Mj	99.986	0.791
5	Too Sweet	Hozier	117.038	0.741
6	we can't be friends (wait for your love)	Ariana Grande	115.853	0.647
7	Lose Control	Teddy Swims	159.920	0.561
8	Illusion	Dua Lipa	126.964	0.679
9	CARNIVAL	¥, KanyeWest, TyDollaign, Rich The Kid, P...	148.144	0.594

Top Songs vs Audio Features : Analysing The Most Frequent Top 10s Danceability vs Tempo

Statistical Analysis

	Tempo	Danceability
count	10.000000	10.000000
mean	129.558100	0.652300
std	23.218879	0.097651
min	99.986000	0.472000
25%	112.226750	0.595250
50%	122.001000	0.663000
75%	150.771250	0.728000
max	159.982000	0.791000

Correlation coefficient between Tempo and Danceability: -0.31921481438170307

Tempo	Danceability
151.647	0.599
105.029	0.472
159.982	0.689
111.018	0.750
99.986	0.791
117.038	0.741
115.853	0.647
159.920	0.561
126.964	0.679
148.144	0.594

- Songs in top 10 already have a high danceability score
- Tempos do not determine a songs danceability
- Other Characteristics such as Songs Musical Structure, Rhythm, Groove ect contribute to the overall Danceability

Top Songs vs Audio Features : Analyse 4 different countries top 10 songs vs Energy

```python
#Visualisation 2 analyse the energy levels from 4 different countries (boxplot)
#use api to get top 10 from the created dataframe
def search_tracks(songs, access_token, country, collected_ids):
    headers = {
        'Authorization': f'Bearer {access_token}'
    }
    track_ids = []
    for song in songs:
        query = f'{song["track"]} {song["artist"]} {country}'
        params = {
            'q': query,
            'type': 'track',
            'limit': 1
        }
        url = 'https://api.spotify.com/v1/search'
        response = requests.get(url, headers=headers, params=params)
        if response.status_code == 200:
            data = response.json()
            if data['tracks']['items']:
                track_id = data['tracks']['items'][0]['id']
                if track_id not in collected_ids[country]:  # Check if track ID is not already collected for this country
                    track_ids.append({'track_id': track_id, 'country': country})
                    collected_ids[country].add(track_id)  # Add track ID to set of collected IDs for this country

    return track_ids

#store the colled ids for each country use set function
collected_ids = {'Australia': set(), 'United Kingdom': set(), 'Malaysia': set(), 'Portugal': set()}

#retrieve the track ids from each song and pretty print for better readability
track_ids_aus = search_tracks(songs[:10], access_token, 'Australia', collected_ids)
print(f'Track IDs (Australia):')
pprint.pprint(track_ids_aus)

track_ids_uk = search_tracks(songs[10:20], access_token, 'United Kingdom', collected_ids)
print(f'Track IDs (United Kingdom):')
pprint.pprint(track_ids_uk)

track_ids_mal = search_tracks(songs[20:30], access_token, 'Malaysia', collected_ids)
print(f'Track IDs (Malaysia):')
pprint.pprint(track_ids_mal)

track_ids_por = search_tracks(songs[30:], access_token, 'Portugal', collected_ids)
print(f'Track IDs (Portugal):')
pprint.pprint(track_ids_por)
```

```
Track IDs (Australia):
[{'country': 'Australia', 'track_id': '0AjmK0Eai4zGrLaJwPvrDp'},
 {'country': 'Australia', 'track_id': '6tNQ70jh4OwmPGpYy6R2o9'},
 {'country': 'Australia', 'track_id': '2GxrNKugF82CnoRFbQfzPf'},
 {'country': 'Australia', 'track_id': '0caJ2wkqp4UmXBwdR2JvB5'},
 {'country': 'Australia', 'track_id': '3qhlB30KknSejmIvZZLjOD'},
 {'country': 'Australia', 'track_id': '17phhZDn6oGtzMe56NuWvj'},
 {'country': 'Australia', 'track_id': '6ClYMObS7f3Nn4AiZHYQt3'},
 {'country': 'Australia', 'track_id': '51ZQ1vr10ffzbwIjDCwqm4'},
 {'country': 'Australia', 'track_id': '3rUGC1vUpkDG9CZFHMur1t'},
 {'country': 'Australia', 'track_id': '0Z7nGFVCLfixWctgePsRk9'}]
Track IDs (United Kingdom):
[{'country': 'United Kingdom', 'track_id': '6XjDF6nds4DE2BBbagZol6'},
 {'country': 'United Kingdom', 'track_id': '5rQSQlZXXjMcevPGoAfE1z'},
 {'country': 'United Kingdom', 'track_id': '4wS0TnQzVkY9ML1BPKpOk1'},
 {'country': 'United Kingdom', 'track_id': '0mflMxspEfB0VbI1kyLiAv'},
 {'country': 'United Kingdom', 'track_id': '4YntVuBqggRsASIbBoWT3F'},
 {'country': 'United Kingdom', 'track_id': '1HfLEcPtq1clzQ5cnU3boo'},
 {'country': 'United Kingdom', 'track_id': '10eRgZUM59q2G5ogpztSeL'},
 {'country': 'United Kingdom', 'track_id': '1kqH58eGh2ZTOHwqBIB2tM'},
 {'country': 'United Kingdom', 'track_id': '28ZVF14CQhyRmBFpJXOpUY'},
 {'country': 'United Kingdom', 'track_id': '7ndALNBDXpk92X7Eh1mdb6'}]
Track IDs (Malaysia):
[{'country': 'Malaysia', 'track_id': '1aKvZDoLGkNMxoRYgkckZG'},
 {'country': 'Malaysia', 'track_id': '51ZQ1vr10ffzbwIjDCwqm4'},
 {'country': 'Malaysia', 'track_id': '6EIMUjQ7Q8Zr2VtIUik4He'},
 {'country': 'Malaysia', 'track_id': '3qhlB30KknSejmIvZZLjOD'},
 {'country': 'Malaysia', 'track_id': '7eaCFseO4fOljCMxLTM5ta'},
 {'country': 'Malaysia', 'track_id': '2VTLUKd4CbW14uKI973vl4'},
 {'country': 'Malaysia', 'track_id': '7F4tV8SiUy6itZTdAzdafO'},
 {'country': 'Malaysia', 'track_id': '2TDIYXFSaesHBw3ZtWpFev'},
 {'country': 'Malaysia', 'track_id': '7CyPwkp0oE8Ro9Dd5CUDjW'},
 {'country': 'Malaysia', 'track_id': '1bjeWoagtHmUKputLVyDxQ'}]
Track IDs (Portugal):
[{'country': 'Portugal', 'track_id': '6XjDF6nds4DE2BBbagZol6'},
 {'country': 'Portugal', 'track_id': '7bywjHOc0wSjGGbj04XbVi'},
 {'country': 'Portugal', 'track_id': '505v13epFXodT9fVAJ6h8k'},
 {'country': 'Portugal', 'track_id': '7iUtQNMRB8ZkKC4AmEuCJC'},
 {'country': 'Portugal', 'track_id': '5gUysq7aO7v7EcV5kQWgeh'},
 {'country': 'Portugal', 'track_id': '1kqH58eGh2ZTOHwqBIB2tM'},
 {'country': 'Portugal', 'track_id': '7iQXYTyuG13aoeHxGG28Nh'},
 {'country': 'Portugal', 'track_id': '54zcJnb3tp9c5OVKREZ1Is'},
 {'country': 'Portugal', 'track_id': '4wS0TnQzVkY9ML1BPKpOk1'},
 {'country': 'Portugal', 'track_id': '3tt9i3Hhzq84dPS8H7iSiJ'}]
```

Top Songs vs Audio Features : VIS2

```python
#Visualisation 2 analyse the energy levels from 4 different countries (boxplot)
def get_audio_features(track_id, access_token):
    headers = {
        'Authorization': f'Bearer {access_token}'
    }
    url = f'https://api.spotify.com/v1/audio-features/{track_id}'
    response = requests.get(url, headers=headers)
    if response.status_code == 200:
        data = response.json()
        return data
    else:
        return None

#get the audio features (specifically energy) for each track ID
audio_features_aus = [get_audio_features(track['track_id'], access_token) for track in track_ids_aus]
audio_features_uk = [get_audio_features(track['track_id'], access_token) for track in track_ids_uk]
audio_features_mal = [get_audio_features(track['track_id'], access_token) for track in track_ids_mal]
audio_features_por = [get_audio_features(track['track_id'], access_token) for track in track_ids_por]

#extract the energy values from the audio features (fror)
energy_aus = [track['energy'] for track in audio_features_aus if track is not None]
energy_uk = [track['energy'] for track in audio_features_uk if track is not None]
energy_mal = [track['energy'] for track in audio_features_mal if track is not None]
energy_por = [track['energy'] for track in audio_features_por if track is not None]

#print all the energy values
print("Energy for Australia:")
pprint.pprint(energy_aus)
print("\nEnergy for United Kingdom:")
pprint.pprint(energy_uk)
print("\nEnergy for Malaysia:")
pprint.pprint(energy_mal)
print("\nEnergy for Portugal:")
pprint.pprint(energy_por)
```

```
Energy for Australia:
[0.62, 0.471, 0.946, 0.595, 0.454, 0.604, 0.907, 0.663, 0.733, 0.709]

Energy for United Kingdom:
[0.499, 0.641, 0.907, 0.488, 0.555, 0.471, 0.812, 0.703, 0.271, 0.597]

Energy for Malaysia:
[0.668, 0.663, 0.493, 0.454, 0.626, 0.558, 0.359, 0.718, 0.641, 0.619]

Energy for Portugal:
[0.499, 0.86, 0.696, 0.738, 0.741, 0.703, 0.778, 0.725, 0.907, 0.679]
```

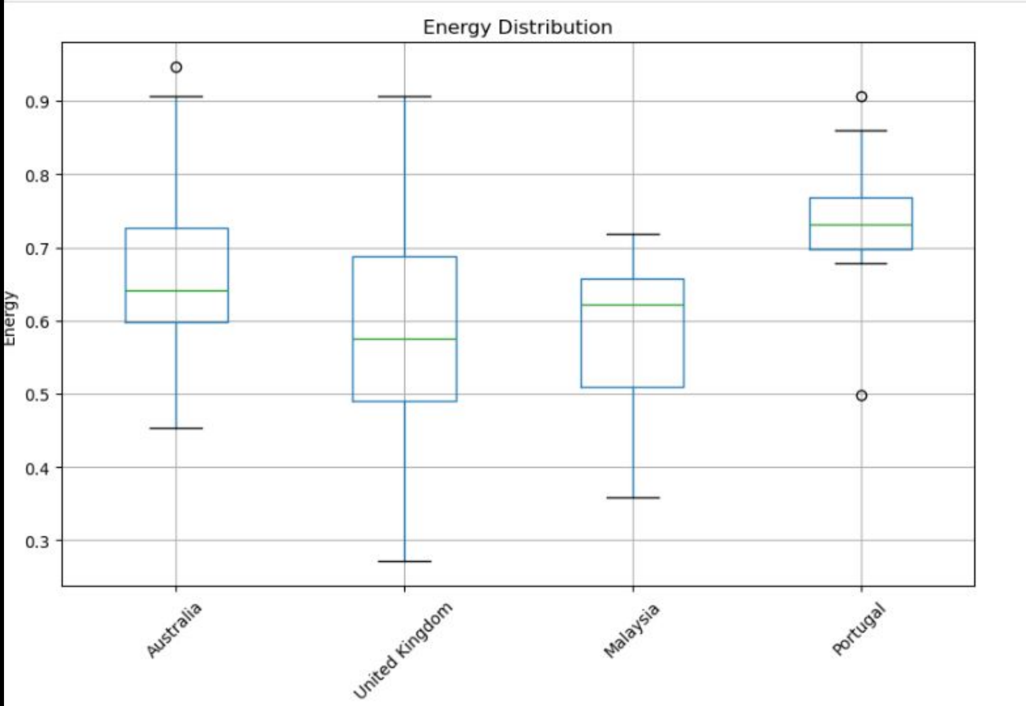
Top Songs vs Audio Features : Analyse 4 different countries top 10 songs vs Energy

Australia:
Median = 0.633
IQR = 0.134

United Kingdom:
Median = 0.584
IQR = 0.187

Malaysia:
Median = 0.596
IQR = 0.124

Portugal:
Median = 0.721
IQR = 0.131



Research Questions

Genre vs Audio Features
Q1 ○ Tash

Top Songs vs Audio Features
Q2 ○ Riona

Album Popularity vs Track Features
Q3 ○ Sri

User Top Tracks vs Daily Mixes
Q4 ○ Mark

Deep Dive into Album Popularity vs Track Features

Q3 ○ Sri

Album Popularity vs Audio Features - Data Analysis and Summary

Using the Spotify API endpoints listed below, data was collected on new music releases, artist information, albums, tracks, and their associated audio features such as danceability, energy, and key:

NEW_RELEASES_URL: Retrieves the latest music releases.

https://api.spotify.com/v1/browse/new-releases

ARTIST_URL: Fetches artist information.

https://api.spotify.com/v1/artists/

ALBUMS_URL: Retrieves albums associated with an artist.

https://api.spotify.com/v1/artists/{artist_id}/albums

TRACKS_URL: Fetches tracks from an album.

https://api.spotify.com/v1/albums/{album_id}/tracks

Individual_TRACK_URL: Provides details about a specific track.

https://api.spotify.com/v1/tracks/{id}

TRACK_FEATURES_URL: Retrieves audio features for a track

https://api.spotify.com/v1/audio-features/{track_id}

•During the data analysis process, the most popular artist, their albums, and tracks were identified. Special attention was given to the most popular album, and a data frame was constructed to analyze the album's popularity and its corresponding audio features.

•Correlation analysis was conducted to determine the relationship between album popularity and track features. To visualize the findings, bar charts, scatter plots, and box plots were generated.

•Additionally, statistical tests, including T-tests and ANOVA, were performed to further explore differences in album popularity across different groups defined by track features such as danceability and key.

Album Popularity vs Audio Features: Data Analysis and conclusions

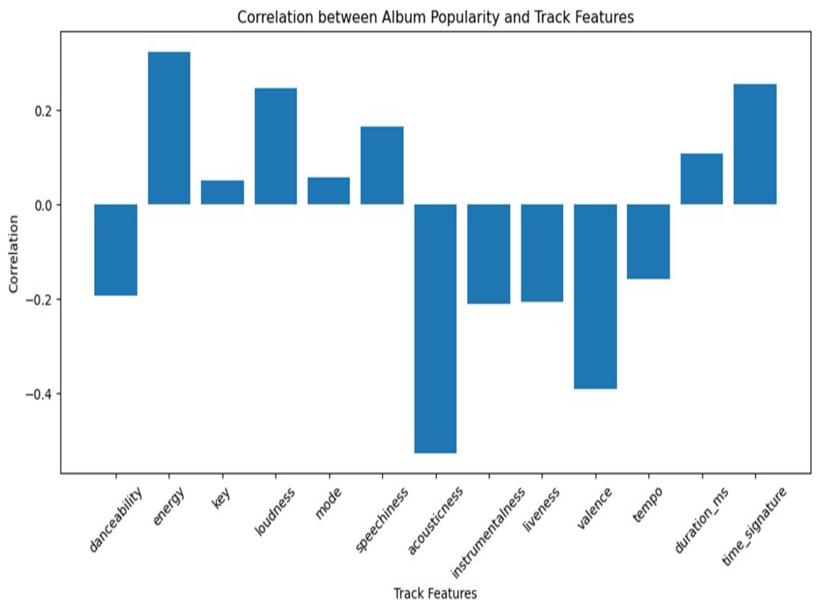
Correlation between Album Popularity and Track Features

The bar graph titled "Correlation between Album Popularity and Track Features" illustrates the relationship between various track features and album popularity.

Positive Correlations:
Energy (0.315), loudness (0.242), speechiness (0.151), and time signature (0.264) exhibit positive correlations with album popularity. Additionally, Key (0.042) and mode (0.045) show weaker positive correlations, suggesting a slight influence on album popularity.

Negative Correlations:
Danceability (-0.188), acousticness (-0.521), instrumentalness (-0.215), liveness (-0.207), and valence (-0.412) have negative correlations with album popularity. Tempo (-0.136) exhibits a weaker negative correlation, suggesting a slight influence.

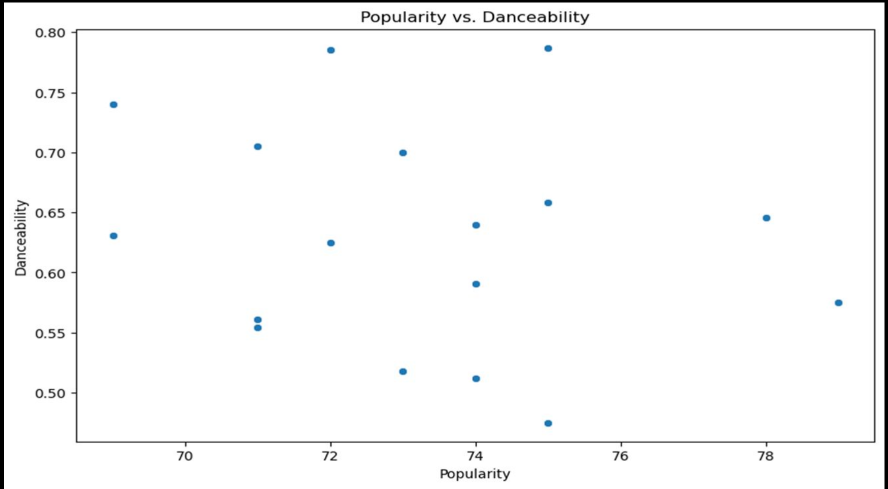
Conclusion:
In summary, certain track features like energy, loudness, speechiness, and certain time signatures tend to correlate positively with album popularity, while factors like acousticness, instrumentalness, liveness, valence, and danceability tend to correlate negatively.



Album Popularity vs Audio Features: Data Analysis and conclusions

The scatter plot graph titled "Popularity vs. Danceability" shows the relationship between popularity and danceability. Here are the key observations:

Popularity (x-axis): The popularity values range from approximately 70 to 78.
Danceability (y-axis): The danceability values range from approximately 0.50 to 0.80.
Data Points: There are twelve blue dots scattered across the graph. These data points do not follow a clear trend or pattern. No strong correlation between popularity and danceability is evident.
Conclusion: Based on this scatter plot, there doesn't appear to be a significant relationship between danceability and album popularity. Each album's popularity seems to vary independently of its danceability score.



Album Popularity vs Audio Features: Data Analysis and conclusions

The bar graph titled “Popularity by Key” displays the popularity levels associated with different musical keys. Here are the key observations:

Popularity by Key:

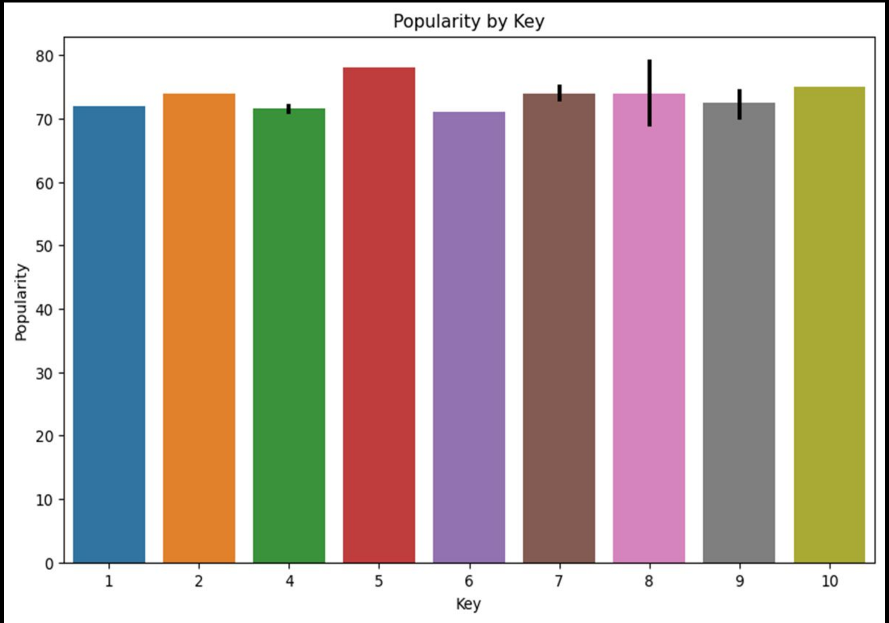
The x-axis represents the keys (numbered from 1 to 10), while the y-axis represents the popularity level (ranging from 0 to 80).

Each key is represented by a colored bar:

All bars extend above a popularity level of approximately 70.

Conclusion:

Each key exhibits a different level of popularity, though all are relatively popular.There are slight variations in popularity levels among the keys.
Overall, the graph indicates that diverse musical keys enjoy popularity, with no key significantly standing out.



Album Popularity vs Audio Features: Data Analysis and conclusions

Box Plot Analysis:

The horizontal line inside each box represents the median (Q2), spanning from the first quartile (Q1) to the third quartile (Q3), containing the middle 50% of the data.
The whiskers extend from the quartiles to the minimum and maximum values. An outlier is marked below the lower whisker in Mode 0.

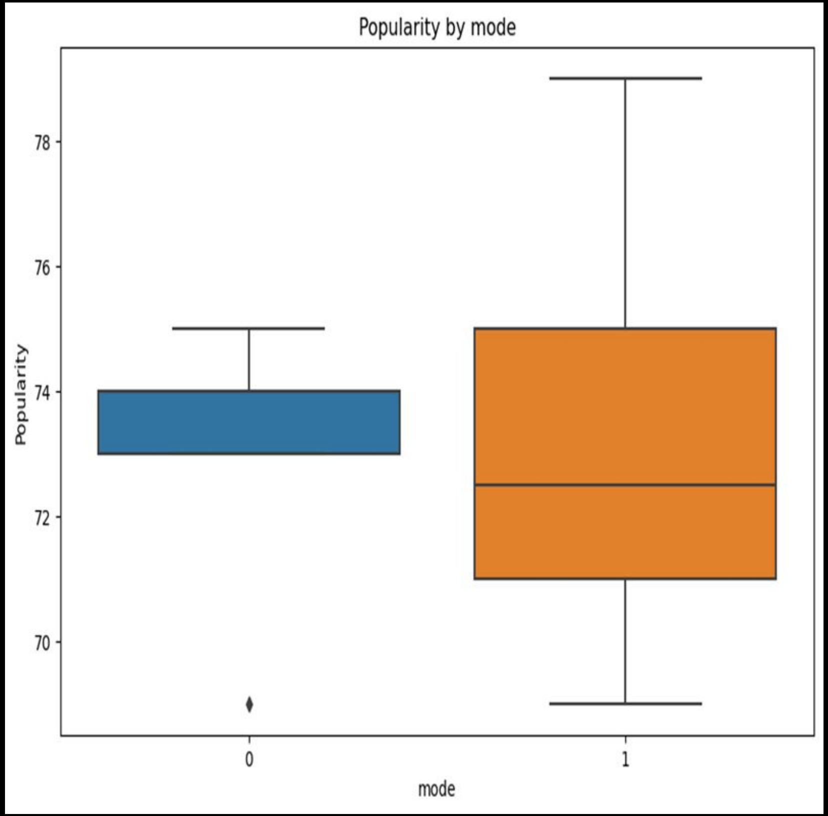
Interpretation:

For mode 0 (blue box):The median is closer to the top of the box, indicating left skewness. The longer whisker on the upper end reinforces this skewness. Therefore, mode 0 is left-skewed (negatively skewed).

For Mode 1 (orange box): The median is closer to the bottom of the box, suggesting right skewness. The shorter whisker on the lower end supports this. Therefore, mode 1 is right-skewed (positively skewed).

Conclusion:

The overall box plot shows a combination of both left-skewed and right-skewed distributions for the two modes. In summary, Mode 1 appears to be more popular than mode 0 based on this graph



Album Popularity vs Audio Features: Data Analysis and conclusions

T-test Analysis and its findings

Comparing Album Popularity Based on Danceability Levels:

Conducted a t-test to compare album popularity between high danceability (danceability >= 0.5) and low danceability (danceability < 0.5) groups.
Using the '**stats.ttest_ind'** function from '**scipy.stats',** the t-statistic and associated p-value were calculated.
With a p-value of 0.526, greater than the significance level of 0.05, the null hypothesis is not rejected, suggesting no significant difference in popularity between the two groups.

Therefore, danceability does not appear to significantly impact album popularity.

```
20]: # T-test comparing popularity of albums with high danceability vs. low danceability
high_danceability = album_data[album_data['danceability'] >= 0.5]['popularity']
low_danceability = album_data[album_data['danceability'] < 0.5]['popularity']

t_statistic, p_value_ttest = stats.ttest_ind(high_danceability, low_danceability)

# Print T-test results
print("T-test results:")
print("T-statistic:", t_statistic)
print("P-value:", p_value_ttest)

# Define significance level (alpha) for T-test
alpha_ttest = 0.05

# Interpret T-test results
if p_value_ttest < alpha_ttest:
    print("There is a statistically significant difference in popularity between albums with high and low danceability.")
else:
    print("There is no statistically significant difference in popularity between albums with high and low danceability.")
```

```
T-test results:
T-statistic: -0.6492350745998484
P-value: 0.5260034381220104
There is no statistically significant difference in popularity between albums with high and low danceability.
```

Album Popularity vs Audio Features: Data Analysis and conclusions

ANOVA analysis and its findings

ANOVA analysis was conducted to compare the popularity of albums across different keys.The popularity data was organized by key, creating separate groups for each unique key. The 'stats.f_oneway' function from the 'scipy.stats' module was utilized to compute the F-statistic and associated p-value for the ANOVA.

The ANOVA results are as follows:
F-statistic: 0.599
P-value: 0.758
With a p-value of 0.758, which exceeds the significance level of 0.05, the null hypothesis is not rejected.
This suggests that there is no statistically significant difference in popularity across different keys. In other words, based on the analysis, the choice of key does not appear to significantly influence the popularity of albums.

20]:
```python
# T-test comparing popularity of albums with high danceability vs. low danceability
high_danceability = album_data[album_data['danceability'] >= 0.5]['popularity']
low_danceability = album_data[album_data['danceability'] < 0.5]['popularity']

t_statistic, p_value_ttest = stats.ttest_ind(high_danceability, low_danceability)

# Print T-test results
print("T-test results:")
print("T-statistic:", t_statistic)
print("P-value:", p_value_ttest)

# Define significance level (alpha) for T-test
alpha_ttest = 0.05

# Interpret T-test results
if p_value_ttest < alpha_ttest:
    print("There is a statistically significant difference in popularity between albums with high and low danceability.")
else:
    print("There is no statistically significant difference in popularity between albums with high and low danceability.")
```

```
T-test results:
T-statistic: -0.6492350745998484
P-value: 0.5260034381220104
There is no statistically significant difference in popularity between albums with high and low danceability.
```

Research Questions

Genre vs Audio Features
Q1 ○ Tash

Top Songs vs Audio Features
Q2 ○ Riona

Album Popularity vs Track Features
Q3 ○ Sri

User Top Tracks vs Daily Mixes
Q4 ○ Mark

Deep Dive into User Top Tracks vs Daily Mixes

Q4 ○ Mark

User Data vs Daily Mix

Data Output

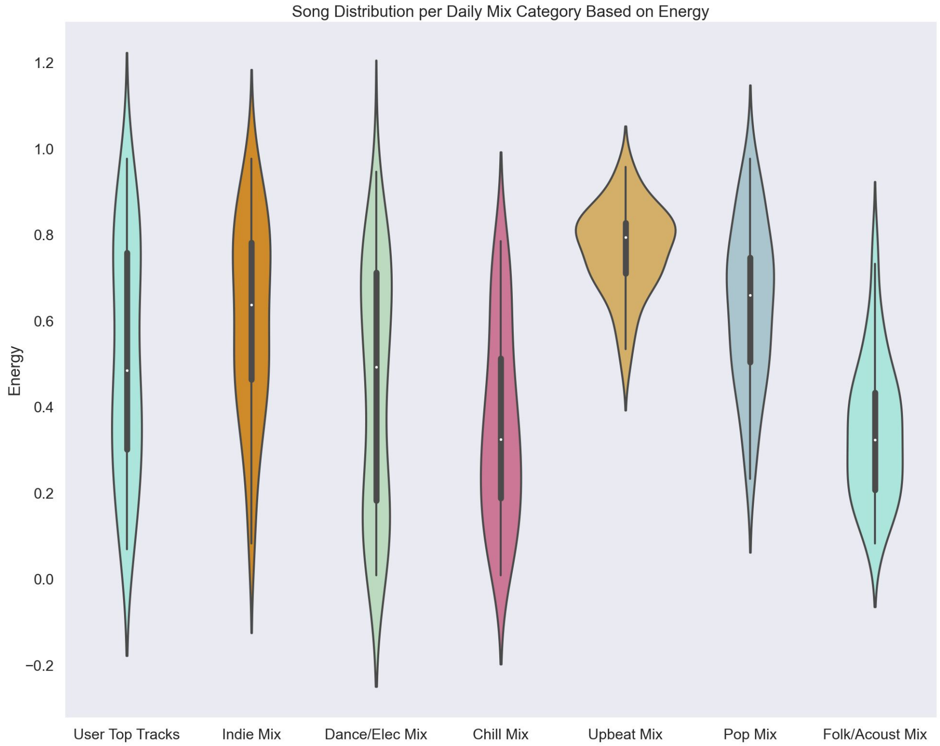
Each track in Spotify contains metadata knowns as “Features”. These features are a numeric representation of the qualities of the track, eg how loud that track is, or how easy it is to dance to.

	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	duration_ms
0	0.399	0.7610	9	-6.318	1	0.0334	0.000105	0.0456	0.0757	0.2430	140.084	304907
1	0.252	0.1880	1	-11.648	1	0.0456	0.329000	0.1210	0.1000	0.0302	86.997	113773
2	0.404	0.4070	5	-11.843	1	0.0299	0.492000	0.0338	0.3910	0.0848	106.547	189293
3	0.430	0.2560	2	-15.737	1	0.0607	0.161000	0.8520	0.1470	0.2080	96.704	330905
4	0.213	0.0695	11	-14.832	1	0.0409	0.018500	0.9580	0.1240	0.0382	112.192	106213

User Data vs Daily Mix

Visualising the Data

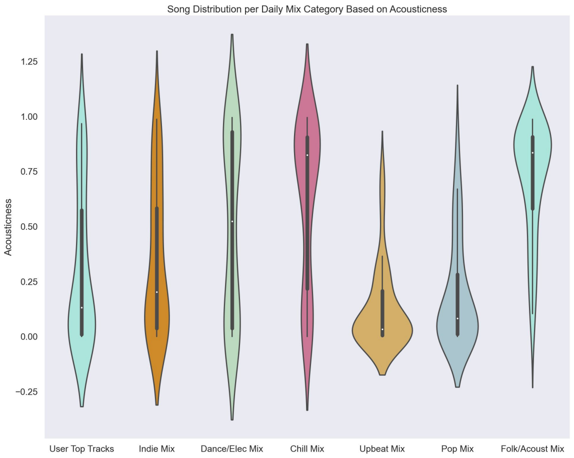
Violin Plot

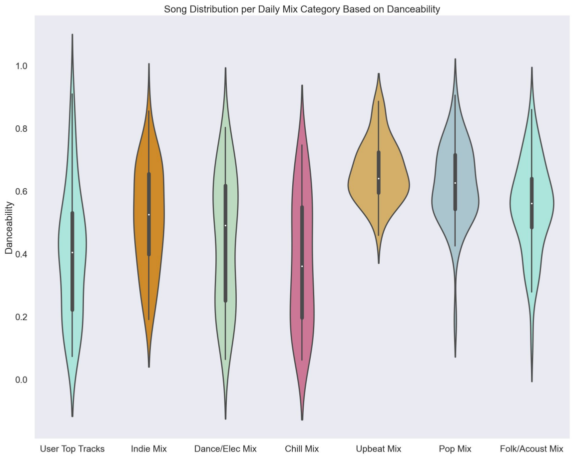


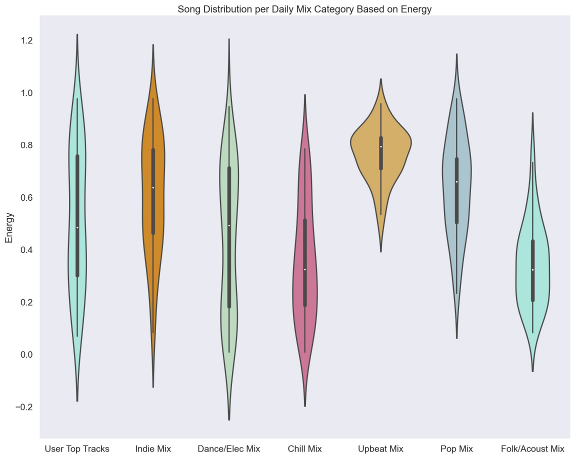
User Data vs Daily Mix

Visualising the Data

Comparing the graphs side by side. Three features (out of twelve) were picked manually to run statistical tests on: danceability, energy, acousticness







User Data vs Daily Mix

Statistical Tests

	ANOVA Score	P Value
Danceability	20.573458	1.376305e-20
Energy	26.219777	1.156735e-25
Acousticness	25.849988	2.431911e-25

	T Score	P Value
Indie Danceability	-2.910622	4.465015e-03
Indie Energy	-1.956052	5.330610e-02
Indie Acousticness	-0.515548	6.073311e-01
Upbeat Danceability	-7.768563	7.881566e-12
Upbeat Energy	-6.256453	1.036904e-08
Upbeat Acousticness	3.276788	1.452878e-03

	Pearson R Score	P Value
Indie Danceability	0.024543	0.865653
Indie Energy	-0.041358	0.775513
Indie Acousticness	-0.048103	0.740095
Upbeat Danceability	-0.036797	0.799726
Upbeat Energy	-0.165771	0.249933
Upbeat Acousticness	-0.189392	0.187742

ANOVA test has the best P values.

T Test has good P values when the test score differs greatly.

Similarity Matrix

Cosine similarity function values stored in a matrix. Generated from Z scores.

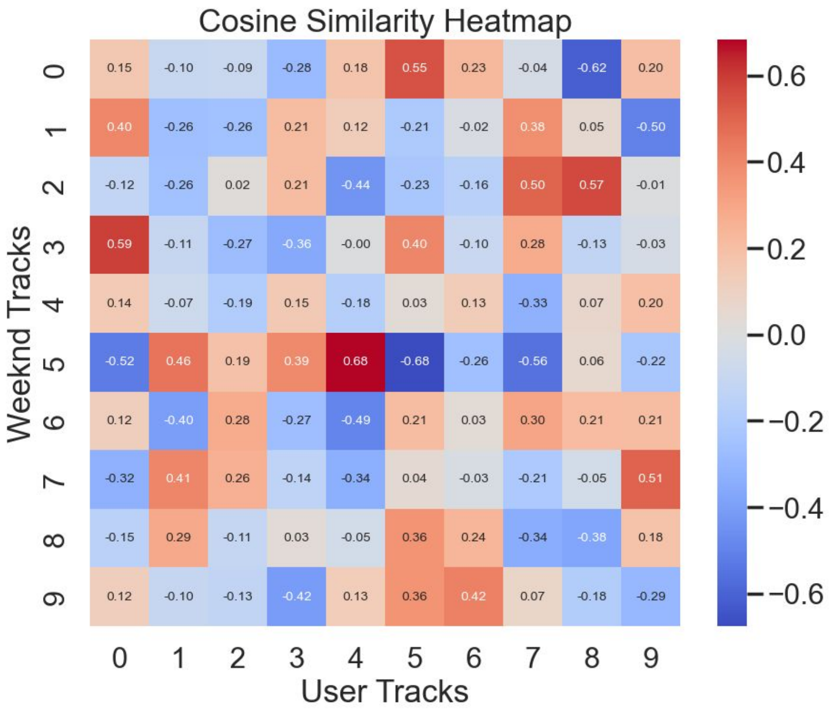
```
# Create matrix using cosine similarity function
similarity_matrix = cosine_similarity(top_weeknd_z_scores.values, features_z_score.values)

# build dataframe for display
similarity_df = pd.DataFrame(similarity_matrix, index=top_weeknd_z_scores.index, columns=features_z_sc
similarity_df
```

	0	1	2	3	4	5	6	7	8	9
0	0.153647	-0.100868	-0.088900	-0.284247	0.177419	0.547879	0.225539	-0.036782	-0.616390	0.204620
1	0.402101	-0.262322	-0.255256	0.210272	0.123630	-0.208126	-0.020619	0.376442	0.049501	-0.502087
2	-0.122630	-0.255833	0.015568	0.206217	-0.440012	-0.225308	-0.162235	0.502341	0.567624	-0.006296
3	0.589635	-0.106175	-0.273008	-0.357188	-0.001530	0.404514	-0.096364	0.276595	-0.126206	-0.033196
4	0.144072	-0.073902	-0.191192	0.149430	-0.176201	0.034012	0.134174	-0.326447	0.066837	0.199747
5	-0.522558	0.457095	0.185955	0.394205	0.683569	-0.675209	-0.256042	-0.557009	0.055456	-0.221833
6	0.119960	-0.401715	0.280017	-0.269796	-0.489835	0.208302	0.034956	0.302678	0.213802	0.206813
7	-0.322536	0.406763	0.263235	-0.135957	-0.337470	0.043581	-0.026239	-0.212431	-0.048910	0.507025
8	-0.150868	0.287781	-0.108203	0.026373	-0.054031	0.362283	0.238673	-0.343011	-0.375437	0.179432
9	0.118062	-0.102270	-0.127394	-0.415042	0.126486	0.360543	0.421124	0.074626	-0.184547	-0.294974

Track Recommendation:
Coding on the Weeknd

Similarity Heatmap



Track Name Reveal

Calling the API to show the name of the recommended track.

	id	best_similarity
0	5gDWsRxpJ2lZAffh5p7K0w	0.683569

```
# Retrieve name from Spotify API
weeknd_track_id = top_weeknd_df_sorted.loc[0:0, "id"]
weeknd_track_info = sp.track(weeknd_track_id[0])
weeknd_track_name = weeknd_track_info['album']['name']
weeknd_track_name
```

```
'Starboy'
```

Thank you!

Any Questions?