Dataset

Before we start building our application, we need a music dataset. For our dataset, we will use the Spotify and Genius Track Dataset from Kaggle. This dataset contains information on thousands of albums, artists, and songs that are collected from the Spotify platform using its API. In addition, the dataset also contains lower-level audio features of the songs, as well as their lyrics.

	Value
# songs	101938
# albums	75503
# artists	40734
# genres	11
# audio features	9
average song duration	4 min 7 sec

Dataset Statistics

Dataset Preprocessing

The goal of our data preprocessing is that we want a joint dataset that consists of each song with its respective genre information, release year, and audio features, as these will be our inputs to the system. Right now, the dataset is mainly separated into three `csv` files: `spotify_artists.csv`, `spotify_albums.csv`, and `spotify_tracks.csv` . `spotify_artists.csv` contains genre information for each artist, `spotify_albums.csv` contains release date for each album, while `spotify_tracks.csv` contains audio features for each song.

To combine the three datasets, we can start by loading the three datasets using Pandas.

```
import pandas as pd
```

```
data_dir = "SpotGenTrack/Data Sources/"
albums_data = pd.read_csv(data_dir + "spotify_albums.csv")
artists_data = pd.read_csv(data_dir + "spotify_artists.csv")
tracks_data = pd.read_csv(data_dir + "spotify_tracks.csv")
```

Read Data

Here are what the three datasets look like:

```
display(albums_data.head())
albums_data.columns
```

	Unnamed: 0	album_type	artist_id	available_markets	external_urls	href	
0	0	single	3DiDSECUqqY1AuBP8qtaIa	['AD', 'AE', 'AR', 'AT', 'AU', 'BE', 'BG', 'BH...	{'spotify': 'https://open.spotify.com/album/1g...	https://api.spotify.com/v1/albums/1gAM7M4rBwEb...	1gA
1	1	album	6s1pCNXcbdtQJlsnM1hRlA	['AD', 'AE', 'AR', 'AT', 'AU', 'BE', 'BG', 'BH...	{'spotify': 'https://open.spotify.com/album/4K...	https://api.spotify.com/v1/albums/4KfJZV7WfolY...	
2	2	single	5YjfNaHq05WrwldRe1QSBc	['AD', 'AE', 'AR', 'AT', 'AU', 'BE', 'BG', 'BH...	{'spotify': 'https://open.spotify.com/album/7n...	https://api.spotify.com/v1/albums/7nLYY7uAVUb5...	7
3	3	single	2G9Vc16JCpnZmK4uGH46Fa	['AD', 'AE', 'AR', 'AT', 'AU', 'BE', 'BG', 'BH...	{'spotify': 'https://open.spotify.com/album/6p...	https://api.spotify.com/v1/albums/6p20Rt4x2Qn5...	6
4	4	single	2dwM9OcE4c3Ph1UBINSodx	['AD', 'AE', 'AR', 'AT', 'AU', 'BE', 'BG', 'BH...	{'spotify': 'https://open.spotify.com/album/1X...	https://api.spotify.com/v1/albums/1XeoOqC1q7U2...	1

```
Index(['Unnamed: 0', 'album_type', 'artist_id', 'available_markets',
       'external_urls', 'href', 'id', 'images', 'name', 'release_date',
       'release_date_precision', 'total_tracks', 'track_id', 'track_name_prev',
       'uri', 'type'],
      dtype='object')
```

Spotify Albums Data (75511 rows)

```
display(artists_data.head())
artists_data.columns
```

	Unnamed: 0	artist_popularity	followers	genres	id	name	track_id	track_name_prev	type
0	0	44	23230	['sertanejo', 'sertanejo pop', 'sertanejo trad...	4mGnpjhqgx4RUdslJiURdo	Juliano Cezar	0wmDmAILuW9e2aRttkl4aC	track_9	artist
1	1	22	313	[]	1dLnVku4VQUOLswwDFvRc9	The Grenadines	4wqwj0gA8qPZKLI5WVqXml	track_30	artist
2	2	26	1596	['danish pop rock']	6YVY310fjfUzKi8hiqR7iK	Gangway	1bFqWDbvHmZe2f4Nf9qaD8	track_38	artist
3	3	31	149	['uk alternative pop']	2VElyouiCfoYPDJluzwJwK	FADES	3MFSUBAidPzRBblS7BDj1S	track_34	artist
4	4	21	11	['french baroque']	4agVy03qW8juSysCTUOuDI	Jean-Pierre Guignon	2r3q57FhxdsCyYr0kuDq4b	track_26	artist

```
Index(['Unnamed: 0', 'artist_popularity', 'followers', 'genres', 'id', 'name',
       'track_id', 'track_name_prev', 'type'],
      dtype='object')
```

Spotify Artists Data (56129 rows)

```
display(tracks_data.head())
tracks_data.columns
```

	Unnamed: 0	acousticness	album_id	analysis_url	artists_id	available_markets	country	danceability	dis
0	0	0.294	0D3QufeCudpQANOR7luqdr	https://api.spotify.com/v1/audio-analysis/5qlj...	['3mxJuHRn2ZWD5OofvJtDZY']	['AD', 'AE', 'AR', 'AT', 'AU', 'BE', 'BG', 'BH...	BE	0.698	
1	1	0.863	1bcqsH5UyTBzmh9YizdsBE	https://api.spotify.com/v1/audio-analysis/3VAX...	['4xWMewm6CYMstu0sPgd9jJ']	['AD', 'AE', 'AR', 'AT', 'AU', 'BE', 'BG', 'BH...	BE	0.719	
2	2	0.750	4tKijjmxGClg4JOLAyo2qE	https://api.spotify.com/v1/audio-analysis/1L3Y...	['3hYaK5FF3YAglCj5HZgBnP']	['GB']	BE	0.466	
3	3	0.763	6FeJF5r8roonnKraJxr4oB	https://api.spotify.com/v1/audio-analysis/6aCe...	['2KQsUB9DRBcJk17JWX1eXD']	['AD', 'AE', 'AR', 'AT', 'AU', 'BE', 'BG', 'BH...	BE	0.719	
4	4	0.770	4tKijjmxGClg4JOLAyo2qE	https://api.spotify.com/v1/audio-analysis/1Vo8...	['3hYaK5FF3YAglCj5HZgBnP']	['GB']	BE	0.460	

5 rows × 32 columns

```
Index(['Unnamed: 0', 'acousticness', 'album_id', 'analysis_url', 'artists_id',
       'available_markets', 'country', 'danceability', 'disc_number',
       'duration_ms', 'energy', 'href', 'id', 'instrumentalness', 'key',
       'liveness', 'loudness', 'lyrics', 'mode', 'name', 'playlist',
       'popularity', 'preview_url', 'speechiness', 'tempo', 'time_signature',
       'track_href', 'track_name_prev', 'track_number', 'uri', 'valence',
       'type'],
      dtype='object')
```

Spotify Tracks Data (101939 rows)

Now, we can join the `albums` and `artists` with `tracks` data. We need to join the album release year and artist genre information with the track data.

```
## join artist genre information and album release date with track dataset
# drop irrelevant columns
# get only tracks after 1990
def join_genre_and_date(artist_df, album_df, track_df):
  album = album_df.rename(columns={'id':"album_id"}).set_index('album_id')
  artist = artist_df.rename(columns={'id':"artists_id",'name':"artists_name"}).set_index('artists_id')
  track = track_df.set_index('album_id').join(album['release_date'], on='album_id' )
  track.artists_id = track.artists_id.apply(lambda x: x[2:-2])
  track = track.set_index('artists_id').join(artist[['artists_name','genres']], on='artists_id' )
  track.reset_index(drop=False, inplace=True)
  track['release_year'] = pd.to_datetime(track.release_date).dt.year
  track.drop(columns = ['Unnamed: 0','country','track_name_prev','track_number','type'], inplace =
True)

  return track[track.release_year >= 1990]
```

Note that we also drop irrelevant columns from our track dataframe and keep only more recent songs (songs published after 1990) to keep our

dataset small, which will allow faster loading and processing times later when we build our app.

We can further narrow down the size of our dataset by including only the songs belonging to certain genres of our choosing.

```
def get_filtered_track_df(df, genres_to_include):
    df['genres'] = df.genres.apply(lambda x: [i[1:-1] for i in str(x)[1:-1].split(", ")])
    df_exploded = df.explode("genres")[df.explode("genres")["genres"].isin(genres_to_include)]
    df_exploded.loc[df_exploded["genres"]=="korean pop", "genres"] = "k-pop"
    df_exploded_indices = list(df_exploded.index.unique())
    df = df[df.index.isin(df_exploded_indices)]
    df = df.reset_index(drop=True)
    return df
```

The dataframe returned by the `get_filtered_track_df` will remove the songs that do not belong to any genre in `genres_to_include` . We then run the following:

```
genres_to_include = genres = ['dance pop', 'electronic', 'electropop', 'hip hop', 'jazz', 'k-pop', 'latin', 'pop', 'pop rap', 'r&b', 'rock']
track_with_year_and_genre = join_genre_and_date(artists_data, albums_data, tracks_data)
filtered_track_df = get_filtered_track_df(track_with_year_and_genre, genres_to_include)
```

After that, we do some more preprocessing for the `uri` column which will be used later on, and drop irrelevant columns further:

```
filtered_track_df["uri"] = filtered_track_df["uri"].str.replace("spotify:track:", "")
filtered_track_df = filtered_track_df.drop(columns=['analysis_url', 'available_markets'])
```

Here is what the preprocessed data looks like:

```
display(filtered_track_df.head())
filtered_track_df.columns
```

	artists_id	acousticness	danceability	disc_number	duration_ms	energy	href	
0	68WwJXWrpo1yVOOlZjLSeT	0.0268	0.506	1.0	248777.0	0.741	https://api.spotify.com/v1/tracks/0UATU9OJxh4m...	0UATU9OJxh4m3fw
1	09xj0S68Y1OU1vHMCZAlvz	0.5050	0.487	1.0	171573.0	0.297	https://api.spotify.com/v1/tracks/4JH1M62gVDND...	4JH1M62gVDNDhDAL
2	6pSsE5y0uJMwYj83KrPyf9	0.1330	0.629	1.0	207396.0	0.706	https://api.spotify.com/v1/tracks/0h7Ld5CvgzaU...	0h7Ld5CvgzaUN1z
3	7slfeZO9LsJbWgpkIoXBUJ	0.4060	0.590	1.0	279000.0	0.597	https://api.spotify.com/v1/tracks/4S1bYWrLOC8s...	4S1bYWrLOC8smuyl
4	09hVlj6vWgoCDtT03h8ZCa	0.0316	0.727	1.0	218773.0	0.380	https://api.spotify.com/v1/tracks/758mQT4zzlvB...	758mQT4zzlvBhy9P

5 rows × 28 columns

```
Index(['artists_id', 'acousticness', 'danceability', 'disc_number',
       'duration_ms', 'energy', 'href', 'id', 'instrumentalness', 'key',
       'liveness', 'loudness', 'lyrics', 'mode', 'name', 'playlist',
       'popularity', 'preview_url', 'speechiness', 'tempo', 'time_signature',
       'track_href', 'uri', 'valence', 'release_date', 'artists_name',
       'genres', 'release_year'],
      dtype='object')
```

Preprocessed Dataset

And finally, we save this into a csv file:

```
filtered_track_df.to_csv("filtered_track_df.csv", index=False)
```