Introduction to Data Analysis:

Before training any machine learning models, I spent time analyzing and improving the dataset to make it more suitable for prediction. This included checking for missing values, understanding how each feature is distributed, and identifying any outliers or unusual patterns. I also worked on scaling the data and transforming it when needed to ensure the models could learn effectively. This step helped me get a better understanding of the data I was working with and made the modeling results more accurate and meaningful.

Step 1: Loading the Dataset:

import pandas as pd

df = pd.read_csv('Most Streamed Spotify Songs 2024.csv', encoding='ISO-8859-1')

df.head()

	Track	Album Name	Artist	Release Date	ISRC	All Time Rank	Track Score	Spotify Streams	Spotify Playlist Count	Spotify Playlist Reach	...	SiriusXM Spins	Deezer Playlist Count	Deezer Playlist Reach	Amazon Playlist Count	Pandora Streams	Pandora Track Stations	Soundcloud Streams	
0	MILLION DOLLAR BABY	Million Dollar Baby - Single	Tommy Richman	4/26/2024	QM24S2402528	1	725.4	390,470,936	30,716	196,631,588	...	684	62.0	17,598,718	114.0	18,004,655	22,931	4,818,457	2,6
1	Not Like Us	Not Like Us	Kendrick Lamar	5/4/2024	USUG12400910	2	545.9	323,703,884	28,113	174,597,137	...	3	67.0	10,422,430	111.0	7,780,028	28,444	6,623,075	1,1
2	i like the way you kiss me	I like the way you kiss me	Artemas	3/19/2024	QZJ842400387	3	538.4	601,309,283	54,331	211,607,669	...	536	136.0	36,321,847	172.0	5,022,621	5,639	7,208,651	5,2

Step 2: Exploring the Structure of the Dataset

After loading the dataset, I used df.info() to check the overall structure of the data. This command showed me how many rows and columns the dataset had, what types of data were in each column (e.g., integers, floats, or objects), and whether any columns contained missing values

```
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Track                    4600 non-null   object
 1   Album Name               4600 non-null   object
 2   Artist                   4595 non-null   object
 3   Release Date             4600 non-null   object
 4   ISRC                     4600 non-null   object
 5   All Time Rank            4600 non-null   object
 6   Track Score              4600 non-null   float64
 7   Spotify Streams          4487 non-null   object
 8   Spotify Playlist Count   4530 non-null   object
 9   Spotify Playlist Reach   4528 non-null   object
 10  Spotify Popularity       3796 non-null   float64
 11  YouTube Views            4292 non-null   object
 12  YouTube Likes            4285 non-null   object
 13  TikTok Posts             3427 non-null   object
 14  TikTok Likes             3620 non-null   object
 15  TikTok Views             3619 non-null   object
 16  YouTube Playlist Reach   3591 non-null   object
 17  Apple Music Playlist Count  4039 non-null  float64
 18  AirPlay Spins            4102 non-null   object
 19  SiriusXM Spins           2477 non-null   object
 20  Deezer Playlist Count    3679 non-null   float64
 21  Deezer Playlist Reach    3672 non-null   object
 22  Amazon Playlist Count    3545 non-null   float64
 23  Pandora Streams          3494 non-null   object
 24  Pandora Track Stations   3332 non-null   object
 25  Soundcloud Streams       1267 non-null   object
 26  Shazam Counts            4023 non-null   object
 27  TIDAL Popularity         0 non-null      float64
 28  Explicit Track           4600 non-null   int64
dtypes: float64(6), int64(1), object(22)
```

Step 3: Identifying and Removing Fully Empty Columns:

To make sure the dataset was clean and complete, I checked for missing values using .isnull().sum(). This printed out the number of missing values in each column. It's an important step because missing data can cause errors or reduce model performance later on. By identifying any columns with gaps early on, I could decide whether to fill in the missing values, drop them, or handle them in another way depending on the situation.

Step 4: Dropping the Empty IDAL_POPULATIRY Column:

During the cleaning process, I found that the column IDAL_POPULATIRY was completely empty — it didn't contain any data in any row. To keep the dataset clean and focused only on useful features, I removed this column using the drop() function. Since it didn't provide any information, removing it helped simplify the dataset and avoid confusion during the analysis and modeling stages.

Step 5: Filling Missing Values:

After cleaning out fully empty columns, I checked for any remaining missing values in the dataset. To handle them, I used different strategies depending on the type of data. For categorical columns (like song names or artists), I filled missing values with the most frequent value (the mode) in that column. For numerical columns (like view counts or popularity scores), I filled missing values using the median, which helps avoid the influence of extreme outliers.

```
Filled 'Artist' with mode: Drake
Filled 'Spotify Streams' with mode: 1,655,575,417
Filled 'Spotify Playlist Count' with mode: 1
Filled 'Spotify Playlist Reach' with mode: 3
Filled 'YouTube Views' with mode: 30,913,276
Filled 'YouTube Likes' with mode: 159,791
Filled 'TikTok Posts' with mode: 1,100,000
Filled 'TikTok Likes' with mode: 1,800,000
Filled 'TikTok Views' with mode: 1,200,000
Filled 'YouTube Playlist Reach' with mode: 381,728
Filled 'AirPlay Spins' with mode: 1
Filled 'SiriusXM Spins' with mode: 1
Filled 'Deezer Playlist Reach' with mode: 1,097
Filled 'Pandora Streams' with mode: 2,829
Filled 'Pandora Track Stations' with mode: 9
Filled 'Soundcloud Streams' with mode: 1,336,043
Filled 'Shazam Counts' with mode: 1
Filled 'Spotify Popularity' with median: 67.0
Filled 'Apple Music Playlist Count' with median: 28.0
Filled 'Deezer Playlist Count' with median: 15.0
Filled 'Amazon Playlist Count' with median: 17.0
```

Step 6: Verifying That All Missing Values Are Handled

After filling in the missing values, I ran one last check using .isnull().sum() to confirm that there were no remaining null values in the dataset. This step helped me make sure the dataset was fully cleaned and ready for analysis and modeling.

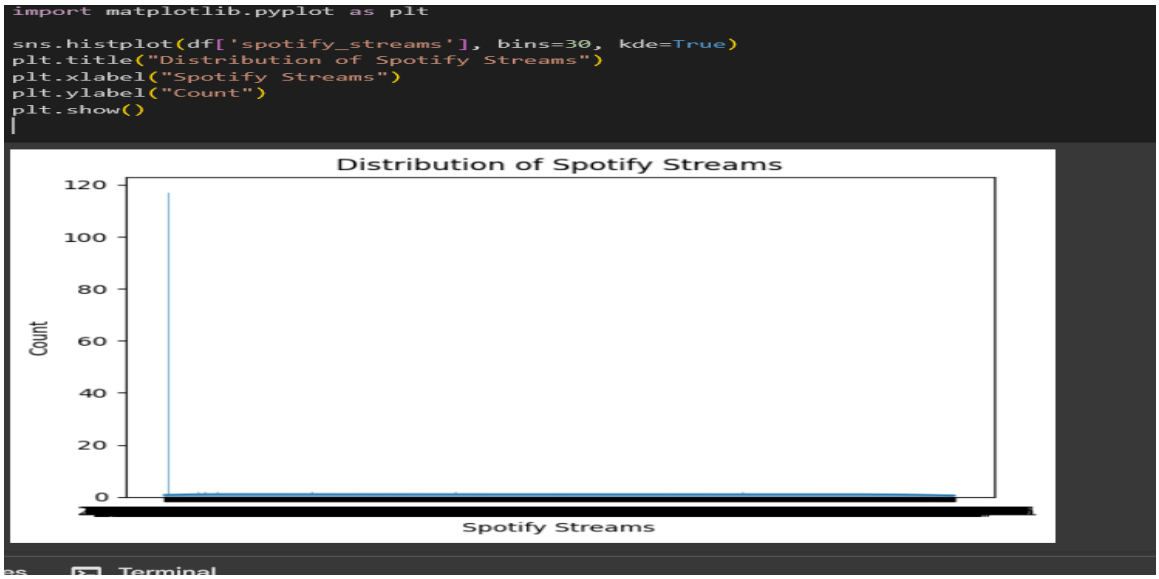
Step 7: Cleaning Column Names

To make the column names easier to work with, I cleaned them up by removing extra spaces, converting all letters to lowercase, and replacing spaces with underscores. This made the column names more consistent and compatible with coding practices, especially when referencing them in later steps. For example, a column like "Spotify Popularity" became "spotify_popularity", which is cleaner and less error-prone in Python.

df.columns = df.columns.str.strip().str.lower().str.replace(" ", "_")

Step 8: Visualizing the Original Distribution of Spotify Streams

Before applying any normalization or transformation, I created a histogram to visualize the distribution of the spotify_streams column. The plot showed that most songs had very low stream counts, while a few songs had extremely high numbers, creating a strong right skew (long tail). This kind of imbalance can negatively affect machine learning models because the extreme values dominate the scale. Visualizing the data in this way helped me realize that normalization or outlier handling would be necessary before training any models.



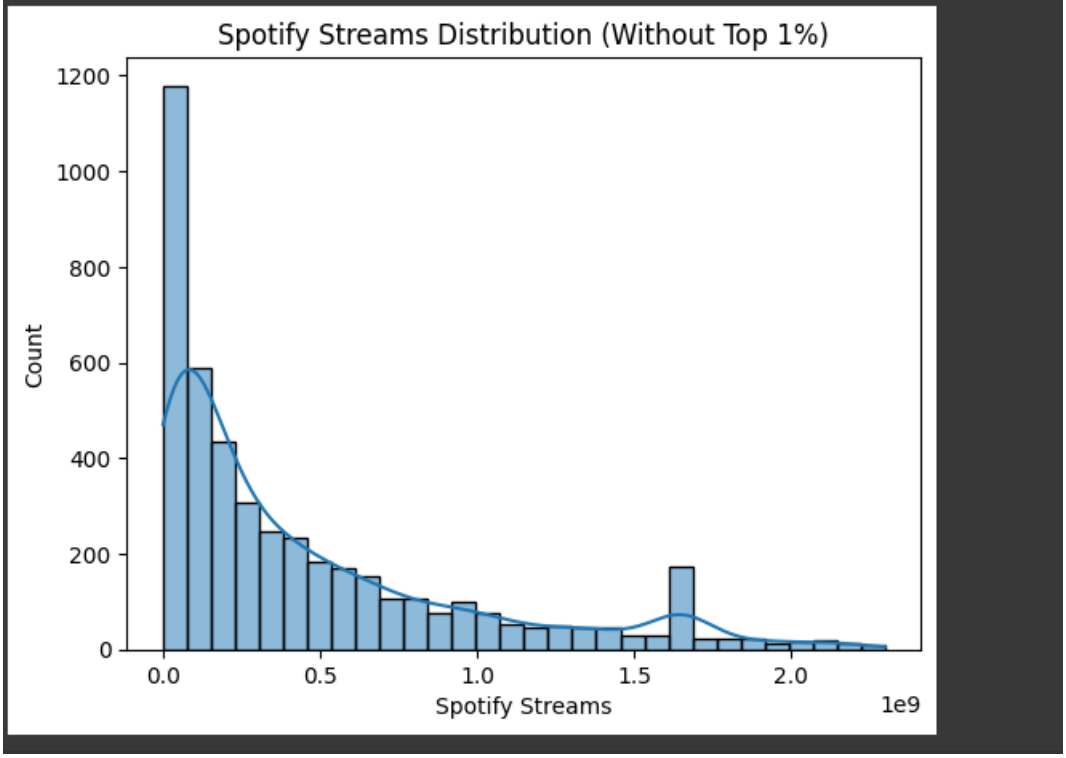
Step 9: Removing Outliers from Spotify Streams

After seeing how skewed the original distribution of spotify_streams was, I decided to remove the top 1% of values to reduce the impact of extreme outliers. I calculated the 99th percentile and filtered out any songs above that threshold. Then, I visualized the new distribution using a histogram. The updated plot looked much more balanced, with a clearer view of how most songs were streamed. This step helped make the data more suitable for machine learning by preventing the models from being biased toward just the few most-streamed songs.

```
q99 = df['spotify_streams'].quantile(0.99)
df_filtered = df[df['spotify_streams'] <= q99]

import seaborn as sns
import matplotlib.pyplot as plt

sns.histplot(df_filtered['spotify_streams'], bins=30, kde=True)
plt.title("Spotify Streams Distribution (Without Top 1%)")
plt.xlabel("Spotify Streams")
plt.ylabel("Count")
plt.show()
```



Step 10: Normalizing Numeric Features

After handling outliers, I normalized all the numeric columns using MinMaxScaler, which scales each value to a range between 0 and 1. This was important because the dataset included features with very different scales — for example, TikTok views and playlist counts could be in totally different ranges. By bringing everything into the same scale, I made sure that no single feature would dominate the model just because of its larger numbers.

Step 11: Verifying Normalization with Descriptive Statistics

To confirm that normalization worked correctly, I used the .describe() function on the numeric columns. This summary showed the minimum, maximum, mean, and other statistics for each feature. Since I applied MinMaxScaler, I expected all minimum values to be close to 0 and maximum values close to 1. This verification step helped ensure that the scaling process was successful and that the numeric data was now properly prepared for machine learning.

	track_score	spotify_streams	spotify_playlist_reach	spotify_popularity	apple_music_playlist_count	deezer_playlist_count	amazon_playlist_count	explicit_track
count	4600.000000	4600.000000	4600.000000	4600.000000	4600.000000	4600.000000	4600.000000	4600.000000
mean	0.031790	0.111426	0.087600	0.664348	0.058693	0.044128	0.107340	0.358913
std	0.054595	0.131666	0.112814	0.155406	0.078864	0.077700	0.110447	0.479734
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.005524	0.017092	0.017154	0.642105	0.012821	0.009509	0.043062	0.000000
50%	0.014873	0.058694	0.049087	0.694737	0.031469	0.022187	0.076555	0.000000
75%	0.035446	0.156494	0.111706	0.736842	0.068765	0.045959	0.129187	1.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

Step 12: Creating a Correlation Matrix:

To understand how the numeric features in the dataset relate to one another, I created a correlation matrix using seaborn. First, I removed non-numeric columns like track names or rankings, since they would interfere with the calculations. Then, I used .corr() to generate the correlation values and visualized them with a heatmap. This plot helped me identify which features were strongly related to each other — for example, some playlist counts and streaming numbers were highly correlated. Understanding these relationships helped guide my decisions for feature selection and gave insight into which variables might have the most influence on a song's popularity.

```
import seaborn as sns
import matplotlib.pyplot as plt


df_corr = df.drop(columns=['track', 'all_time_rank'], errors='ignore')


plt.figure(figsize=(15, 10))
sns.heatmap(df_corr.corr(), annot=True, cmap='coolwarm')
plt.title("Correlation Matrix")
plt.show()
```

Conclusion:

This data analysis and cleaning phase was a critical part of the project. It helped transform a raw dataset into something ready for machine learning. By removing empty columns, filling in missing values, and normalizing numerical data, I ensured that the dataset was clean, balanced, and fair. Visualizations helped me understand the shape and distribution of the data, while the correlation matrix gave me insight into how the features were related. These steps made the modeling process smoother and more accurate later on.