Analysis of Spotify tracks

Aim: To examine the audio characteristics of Spotify's Top 50 tracks such as danceability, energy and loudness, to identify the key features that underpin a successful hit.

1 Project Setup and Data Acquisition

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

2 Data Loading and Initial Exploration

1) Preview the first few rows to see what the data looks like:

2.1–2 Data Loading & Initial Exploration

Goal:
Load the `spotifytop50csv`, verify that rows and columns are intact, and get an initial look at data types and sample values.

In [2]:
```
data_path = "../data/spotifytoptracks.csv"
spotify_df = pd.read_csv(data_path)
spotify_df.head()
```

Out[2]:

	Unnamed: 0	artist	album	track_name	track_id	energ
0	0	The Weeknd	After Hours	Blinding Lights	0VjIjW4GlUZAMYd2vXMi3b	0.7
1	1	Tones And I	Dance Monkey	Dance Monkey	1rgnBhdG2JDFTbYkYRZAku	0.5
2	2	Roddy Ricch	Please Excuse Me For Being Antisocial	The Box	0nbXyq5TXYPCO7pr3N8S4I	0.5
3	3	SAINt JHN	Roses (Imanbek Remix)	Roses - Imanbek Remix	2Wo6QQD1KMDWeFkkjLqwx5	0.7
4	4	Dua Lipa	Future Nostalgia	Don't Start Now	3PfIrDoz19wz7qK7tYeu62	0.7

2) This shows the final rows to quickly verify the dataset's end state:

In [3]:
```
spotify_df = spotify_df.drop(columns=['Unnamed: 0'], errors='ignore')
spotify_df.tail()
```

Out[3]:

	artist	album	track_name	track_id	energy	danc
45	Juice WRLD	Goodbye & Good Riddance	Lucid Dreams	285pBltuF7vW8TeWk8hdRR	0.566	
46	Ariana Grande	Stuck with U	Stuck with U (with Justin Bieber)	4HBZA5flZLE435QTztThqH	0.450	
47	JP Saxe	If the World Was Ending (feat. Julia Michaels)	If the World Was Ending - feat. Julia Michaels	2kJwzbxV2ppxnQoYw4GLBZ	0.473	
48	Dua Lipa	Future Nostalgia	Physical	3AzjcOeAmA57TIOr9zF1ZW	0.844	
49	Travis Scott	ASTROWORLD	SICKO MODE	2xLMifQCjDGFmkHkpNLD9h	0.730	

2.1-2 Data Loading & Initial Exploration

- **Complete dataset ready for analysis:** Loaded 50 tracks × 16 audio features with zero missing files.
- **Feature uniformity:** All expected audio metrics are present and properly typed, enabling direct comparison.

- **Index cleanup:** Removed `Unnamed: 0` to avoid skewing summaries or visuals.

---

3) Display summary statistics for numeric columns:

2.3 Summary Statistics for Numeric Audio Features

Goal:
Review each numeric column's basic stats.

In [4]:
```
spotify_df.describe()
```

Out[4]:

	energy	danceability	key	loudness	acousticness	speechine
count	50.000000	50.000000	50.000000	50.000000	50.000000	50.0000
mean	0.609300	0.716720	5.720000	-6.225900	0.256206	0.1241
std	0.154348	0.124975	3.709007	2.349744	0.265250	0.1168
min	0.225000	0.351000	0.000000	-14.454000	0.001460	0.0290
25%	0.494000	0.672500	2.000000	-7.552500	0.052800	0.0483
50%	0.597000	0.746000	6.500000	-5.991500	0.188500	0.0700
75%	0.729750	0.794500	8.750000	-4.285500	0.298750	0.1555
max	0.855000	0.935000	11.000000	-3.280000	0.934000	0.4870

2.3 Summary Statistics for Key Audio Features

- Every numeric column has 50 values (no missing data).
- **Energy:** Ranges from 0.23 to 0.85.
- **Danceability:** Spans 0.35 to 0.94.
- **Loudness:** Goes from -14.45 dB to -3.28 dB.
- **Tempo:** Varies between 76 BPM and 180 BPM.
- **Duration:** Runs from 140 526 ms to 312 820 ms.

---

4) Inspect summary information about the Dataframe (data types and non-null counts):

2.4 Inspect DataFrame Summary

Goal:
View each column’s data type and non-null count to verify structure and completeness.

In [5]:
```
spotify_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 16 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   artist            50 non-null     object 
 1   album             50 non-null     object 
 2   track_name        50 non-null     object 
 3   track_id          50 non-null     object 
 4   energy            50 non-null     float64
 5   danceability      50 non-null     float64
 6   key               50 non-null     int64  
 7   loudness          50 non-null     float64
 8   acousticness      50 non-null     float64
 9   speechiness       50 non-null     float64
 10  instrumentalness  50 non-null     float64
 11  liveness          50 non-null     float64
 12  valence           50 non-null     float64
 13  tempo             50 non-null     float64
 14  duration_ms       50 non-null     int64  
 15  genre             50 non-null     object 
dtypes: float64(9), int64(2), object(5)
memory usage: 6.4+ KB
```

2.4 Key Insights from DataFrame Structure

- **Complete dataset:** 50 rows × 16 columns, zero missing values.
- **Numeric features:** 11 floats/ints ready for analysis.
- **Categorical columns:** 5 fields (artist, album, track_name, track_id, key).
- **Memory usage:** 6.4+ KB

Takeaway: A clean, complete dataset lays the foundation for reliable exploration and downstream analyses.

---

3 Data cleaning

1) Check for missing values:

3.1 Check for Missing Values

Goal:
Determine whether any features in the dataset contain null (missing) entries.

In [6]:
```
spotify_df.isnull().sum()
```

Out[6]:
```
artist              0
album               0
track_name          0
track_id            0
energy              0
danceability        0
key                 0
loudness            0
acousticness        0
speechiness         0
instrumentalness    0
liveness            0
valence             0
tempo               0
duration_ms         0
genre               0
dtype: int64
```

3.1 Check for Missing Values

- **Takeaway:** Every one of the 16 columns has zero missing entries.

---

2) Show number of duplicate rows:

3.2 Duplicate Row Check

Goal:
Find out how many exact duplicate rows are in the dataset.

In [7]:
```
num_duplicates = spotify_df.duplicated().sum()
print("Number of duplicate rows:", num_duplicates)
```

```
Number of duplicate rows: 0
```

3.2 Duplicate Row Check

- **Takeaway:** Zero exact duplicate rows - each entry is a distinct track record.

---

3) Treating Outliers:

3.3 Treating Outliers

Goal:
Identify extreme values in each numeric feature using the IQR rule (1.5 × IQR beyond Q1/Q3).

In [8]:
```
"""
Cell 1: compute IQR bounds and print outlier counts
"""
num_cols = spotify_df.select_dtypes(include=np.number).columns.tolist()

bounds = {}

for col in num_cols:
    q1 = spotify_df[col].quantile(0.25)
    q3 = spotify_df[col].quantile(0.75)
    iqr = q3 - q1
    lower_bound = q1 - 1.5 * iqr
    higher_bound = q3 + 1.5 * iqr

    bounds[col] = (lower_bound, higher_bound)

    out = spotify_df[
        (spotify_df[col] < lower_bound)
        | (spotify_df[col] > higher_bound)
    ]
    print(col, "outliers:", len(out))
```

```
energy outliers: 0
danceability outliers: 3
key outliers: 0
loudness outliers: 1
acousticness outliers: 7
speechiness outliers: 6
instrumentalness outliers: 12
liveness outliers: 3
valence outliers: 0
tempo outliers: 0
duration_ms outliers: 2
```

Goal:
Build a outlier table that flags which tracks fall outside the IQR-based bounds for each numeric feature but also captures the exact outlier values and their cutoff thresholds.

In [9]:
```
"""
Cell 2: build a detailed outlier table
"""

outlier_records = []
for feature_name, (lower_bound, higher_bound) in bounds.items():
    for idx, val in spotify_df[feature_name].items():
        if val < lower_bound or val > higher_bound:
            outlier_records.append({
                'artist'      : spotify_df.at[idx, 'artist'],
                'track_name'  : spotify_df.at[idx, 'track_name'],
```

```
                'feature'     : feature_name,
                'value'       : val,
                'lower_bound' : lower_bound,
                'upper_bound' : higher_bound
            })

outliers_df = (
    pd.DataFrame(outlier_records)
      .drop_duplicates(subset=['artist','track_name','feature'])
      .reset_index(drop=True)
)
outliers_df.index += 1
outliers_df
```

Out[9]:

	artist	track_name	feature	value	lower_bound	upp
1	Lewis Capaldi	Before You Go	danceability	0.459000	0.489500	
2	Billie Eilish	lovely (with Khalid)	danceability	0.351000	0.489500	
3	JP Saxe	If the World Was Ending - feat. Julia Michaels	danceability	0.464000	0.489500	
4	Billie Eilish	everything i wanted	loudness	-14.454000	-12.453000	
5	Tones And I	Dance Monkey	acousticness	0.688000	-0.316125	
6	Powfu	death bed (coffee for your head)	acousticness	0.731000	-0.316125	
7	Lewis Capaldi	Someone You Loved	acousticness	0.751000	-0.316125	
8	Maroon 5	Memories	acousticness	0.837000	-0.316125	
9	Billie Eilish	everything i wanted	acousticness	0.902000	-0.316125	
10	Billie Eilish	lovely (with Khalid)	acousticness	0.934000	-0.316125	
11	JP Saxe	If the World Was Ending - feat. Julia Michaels	acousticness	0.866000	-0.316125	
12	Future	Life Is Good (feat. Drake)	speechiness	0.487000	-0.112437	
13	Billie Eilish	bad guy	speechiness	0.375000	-0.112437	
14	Cardi B	WAP (feat. Megan Thee Stallion)	speechiness	0.375000	-0.112437	
15	Eminem	Godzilla (feat. Juice WRLD)	speechiness	0.342000	-0.112437	
16	Maluma	Hawái	speechiness	0.389000	-0.112437	
17	Bad Bunny	Safaera	speechiness	0.379000	-0.112437	
18	The Weeknd	Blinding Lights	instrumentalness	0.000095	-0.000030	
19	Tones And I	Dance Monkey	instrumentalness	0.000161	-0.000030	

	artist	track_name	feature	value	lower_bound	upp
20	SAINt JHN	Roses - Imanbek Remix	instrumentalness	0.004320	-0.000030	
21	KAROL G	Tusa	instrumentalness	0.000134	-0.000030	
22	Post Malone	Circles	instrumentalness	0.002440	-0.000030	
23	Billie Eilish	everything i wanted	instrumentalness	0.657000	-0.000030	
24	Billie Eilish	bad guy	instrumentalness	0.130000	-0.000030	
25	BENEE	Supalonely (feat. Gus Dapperton)	instrumentalness	0.000209	-0.000030	
26	Surf Mesa	ily (i love you baby) (feat. Emilee)	instrumentalness	0.001880	-0.000030	
27	Regard	Ride It	instrumentalness	0.000064	-0.000030	
28	Black Eyed Peas	RITMO (Bad Boys For Life)	instrumentalness	0.001090	-0.000030	
29	Dua Lipa	Physical	instrumentalness	0.000658	-0.000030	
30	Roddy Ricch	The Box	liveness	0.790000	-0.172000	
31	Powfu	death bed (coffee for your head)	liveness	0.696000	-0.172000	
32	Black Eyed Peas	RITMO (Bad Boys For Life)	liveness	0.792000	-0.172000	
33	Bad Bunny	Safaera	duration_ms	295177.000000	117017.750000	27389
34	Travis Scott	SICKO MODE	duration_ms	312820.000000	117017.750000	27389

3.3 Outlier Handling

- **Detection Method:** Applied the 1.5 × IQR rule to every numeric feature and counted outliers per column.
- **Audit Table:** Compiled a table listing each outlier’s track, feature name, actual value, and cutoff bounds.

- **Handling Decision:** Recognized outliers as valid data points (e.g., long mixes, spoken-word tracks), not errors and retained them in the dataset.
- **Action:** I've logged every outlier in the audit table for my clarity and will include all outlier records unchanged in my further analyses.

Insight: Retaining genuine outliers ensures we capture all defining extremes that make a track a Top 50 hit.

---

4 Exploratory Data Analysis

4.1 Basic Stats & Structure

1) Basic Stats & Structure (Observations & Features):

How many observations are there in this dataset?

How many features this dataset has?

4.1.1 Basic Stats & Structure

Goal:
Determine how many observations (rows) and features (columns) are in our dataset.

In [10]:
```
num_tracks, num_attributes = spotify_df.shape

print(f"Total tracks (observations): {num_tracks}")
print(f"Total attributes (features): {num_attributes}")
```

```
Total tracks (observations): 50
Total attributes (features): 16
```

4.1.1 Basic Stats & Structure

- **Takeaway:** The dataset contains 50 tracks described by 16 attributes (features).

---

2) Basic Stats & Structure (Categorical & Numeric):

Which of the features are categorical?

Which of the features are numeric?

4.1.2 Feature Types Overview

Goal:
Identify which columns in the dataset are categorical vs. numeric.

In [11]:
```
cat_cols = spotify_df.select_dtypes(include=['object']).columns.tolist()

print("Categorical features:", cat_cols)
print("Numeric features:", num_cols)
```

```
Categorical features: ['artist', 'album', 'track_name', 'track_id', 'genre']
Numeric features: ['energy', 'danceability', 'key', 'loudness', 'acousticnes
s', 'speechiness', 'instrumentalness', 'liveness', 'valence', 'tempo', 'dura
tion_ms']
```

4.1.2 Feature Types Overview

Results:

- **Categorical features (5):** `artist`, `album`, `track_name`, `track_id`, `genre`
- **Numeric features (11):** `energy`, `danceability`, `key`, `loudness`, `acousticness`, `speechiness`, `instrumentalness`, `liveness`, `valence`, `tempo`, `duration_ms`
- **Total features:** 16

4.2 Artist & Album Checks

1) Artist & Album Checks (Multi-Track Artists):

Are there any artists that have more than 1 popular track? If yes, which and how many?

4.2.1 Multi-Track Artists

Goal:
Identify which artists have more than one song in the Top 50.

In [12]:
```
artist_counts = spotify_df['artist'].value_counts()
multi_artist = artist_counts[artist_counts > 1]
```

```
print(multi_artist)
```

```
artist
Dua Lipa         3
Billie Eilish    3
Travis Scott     3
Harry Styles     2
Lewis Capaldi    2
Justin Bieber    2
Post Malone      2
Name: count, dtype: int64
```

4.2.1 Multi-Track Artists

Insight: Seven artists place multiple tracks in the Top 50. Dua Lipa, Billie Eilish, and Travis Scott each with three songs - highlighting their dominant presence in 2020's hits. Harry Styles, Lewis Capaldi, Justin Bieber, and Post Malone each contribute two songs.

---

2) Artist & Album Checks (Most Popular Artist):

Who was the most popular artist?

4.2.2 Most Popular Artist

Goal:
Determine which artist appears most frequently among the Top 50 tracks.

In [13]:
```
max_count = artist_counts.max()
most_pop_artist = artist_counts[artist_counts == max_count]
print(most_pop_artist)
```

```
artist
Dua Lipa         3
Billie Eilish    3
Travis Scott     3
Name: count, dtype: int64
```

4.2.2 Most Popular Artist

Insight: Ranked by track count in the Top 50, the most popular artists were Dua Lipa, Billie Eilish, and Travis Scott. Each with three hits - there's no single standout.

Note: This dataset lacks actual stream counts. To truly gauge popularity, we'd need to merge in streaming data.

---

3) Artist & Album Checks (Total Unique Artists):

How many artists in total have their songs in the top 50?

4.2.3 Total Unique Artists

Goal:

Count how many distinct artists have songs in the Top 50.

In [14]:
```
total_artists = spotify_df['artist'].nunique()
print(f"Total unique artists: {total_artists}")
```

```
Total unique artists: 40
```

4.2.3 Total Unique Artists

- **Takeaway:** 40 distinct artists.

---

4) Artist & Album Checks (Multiple Top Tracks):

Are there any albums that have more than 1 popular track? If yes, which and how many?

4.2.4 Albums with Multiple Top 50 Tracks

Goal:

Determine which albums have more than one track in the Top 50 and count them.

In [15]:
```
album_counts = spotify_df['album'].value_counts()
multi_album = album_counts[album_counts > 1]
print(multi_album)
```

```
album
Future Nostalgia        3
Hollywood's Bleeding    2
Fine Line               2
Changes                 2
Name: count, dtype: int64
```

4.2.4 Albums with Multiple Top 50 Tracks

- **Takeaway:** Four albums place multiple tracks in the Top 50: Future Nostalgia leads with three records, while Hollywood's Bleeding, Fine Line, and Changes each contribute two entries.

---

5) Artist & Album Checks (Total Unique Albums):

How many albums in total have their songs in the top 50?

4.2.5 Total Unique Albums

Goal:
Count how many different albums are represented in the Top 50.

In [16]:
```
total_albums = spotify_df['album'].nunique()
print(f"Total unique albums: {total_albums}")
```

```
Total unique albums: 45
```

4.2.5 Total Unique Albums

- **Takeaway:** 45 unique albums appear in the Top 50.

4.3 Track-Level Queries

1) Track-Level Queries (High Danceability):

Which tracks have a danceability score above 0.7?

4.3.1 High Danceability Tracks

Goal:
Identify which tracks have a danceability score above 0.7.

In [17]:
```
high_dance_sorted = (
    spotify_df[spotify_df['danceability'] > 0.7]
      [['artist','track_name','danceability']]
      .sort_values('danceability', ascending=False)
      .reset_index(drop=True)
)
```

```
high_dance_sorted.index = high_dance_sorted.index + 1
high_dance_sorted
```

Out[17]:

	artist	track_name	danceability
1	Cardi B	WAP (feat. Megan Thee Stallion)	0.935
2	Roddy Ricch	The Box	0.896
3	Regard	Ride It	0.880
4	Surfaces	Sunday Best	0.878
5	BENEE	Supalonely (feat. Gus Dapperton)	0.862
6	Travis Scott	goosebumps	0.841
7	Travis Scott	SICKO MODE	0.834
8	Drake	Toosie Slide	0.830
9	Tones And I	Dance Monkey	0.825
10	Eminem	Godzilla (feat. Juice WRLD)	0.808
11	Justin Bieber	Intentions (feat. Quavo)	0.806
12	KAROL G	Tusa	0.803
13	Future	Life Is Good (feat. Drake)	0.795
14	Dua Lipa	Don't Start Now	0.793
15	Topic	Breaking Me	0.789
16	Doja Cat	Say So	0.787
17	SAINt JHN	Roses - Imanbek Remix	0.785
18	Trevor Daniel	Falling	0.784
19	Maluma	Hawái	0.783
20	Lil Mosey	Blueberry Faygo	0.774
21	Jawsh 685	Savage Love (Laxed - Siren Beat)	0.767
22	Maroon 5	Memories	0.764
23	Shawn Mendes	Señorita	0.759
24	Post Malone	Sunflower - Spider-Man: Into the Spider-Verse	0.755
25	BTS	Dynamite	0.746
26	DaBaby	ROCKSTAR (feat. Roddy Ricch)	0.746
27	Dua Lipa	Break My Heart	0.730
28	Powfu	death bed (coffee for your head)	0.726
29	Black Eyed Peas	RITMO (Bad Boys For Life)	0.723
30	THE SCOTTS	THE SCOTTS	0.716
31	Billie Eilish	everything i wanted	0.704
32	Billie Eilish	bad guy	0.701

4.3.1 High Danceability Tracks

- **Takeaway:** 32 tracks exceed a danceability of 0.7. Cardi B's "WAP (feat. Megan Thee Stallion)" tops the chart with the highest danceability score at 0.935. Billie Eilish's "bad guy" marks the lower edge of this group at 0.701.

---

2) Track-Level Queries (Low Danceability):

Which tracks have a danceability score below 0.4?

4.3.2 Low Danceability Tracks

Goal:
Identify which tracks have a danceability score below 0.4.

In [18]:
```
low_dance_sorted= (
    spotify_df[spotify_df['danceability'] < 0.4]
      [['artist','track_name','danceability']]
      .reset_index(drop=True)
)
low_dance_sorted.index = low_dance_sorted.index + 1
low_dance_sorted
```

Out[18]:

	artist	track_name	danceability
1	Billie Eilish	lovely (with Khalid)	0.351

4.3.2 Low Danceability Tracks

- **Takeaway:** Only one track Billie Eilish's "lovely (with Khalid)" has a danceability below 0.4 (0.351).

---

3) Track-Level Queries (Loudest Tracks):

Which tracks have their loudness above -5?

4.3.3 Loudest Tracks

Goal:
Identify which tracks play louder than -5 dB in the dataset.

In [19]:
```
high_loud_sorted = (
    spotify_df[spotify_df['loudness'] > -5]
      [['artist','track_name','loudness']]
      .sort_values('loudness', ascending=False)
      .reset_index(drop=True)
)
high_loud_sorted.index = high_loud_sorted.index + 1
high_loud_sorted
```

Out[19]:

	artist	track_name	loudness
1	KAROL G	Tusa	-3.280
2	Travis Scott	goosebumps	-3.370
3	Dua Lipa	Break My Heart	-3.434
4	Maluma	Hawái	-3.454
5	Post Malone	Circles	-3.497
6	24kGoldn	Mood (feat. iann dior)	-3.558
7	Harry Styles	Adore You	-3.675
8	Travis Scott	SICKO MODE	-3.714
9	Dua Lipa	Physical	-3.756
10	Lady Gaga	Rain On Me (with Ariana Grande)	-3.764
11	Bad Bunny	Safaera	-4.074
12	Harry Styles	Watermelon Sugar	-4.209
13	Regard	Ride It	-4.258
14	Post Malone	Sunflower - Spider-Man: Into the Spider-Verse	-4.368
15	BTS	Dynamite	-4.410
16	Dua Lipa	Don't Start Now	-4.521
17	Doja Cat	Say So	-4.577
18	BENEE	Supalonely (feat. Gus Dapperton)	-4.746
19	Lewis Capaldi	Before You Go	-4.858

4.3.3 Loudest Tracks

- **Takeaway:** 19 tracks exceed the -5 dB threshold. The loudest song is KAROL G's "Tusa" at -3.280 dB.

---

4) Track-Level Queries (Softest Tracks):

Which tracks have their loudness below -8?

4.3.4 Softest Tracks (Loudness < -8 dB)

Goal:
Identify which tracks fall below -8 dB in loudness.

In [20]:
```
low_loud_sorted = (
    spotify_df[spotify_df['loudness'] < -8]
      [['artist','track_name','loudness']]
      .sort_values('loudness', ascending=True)
      .reset_index(drop=True)
)
low_loud_sorted.index = low_loud_sorted.index + 1
low_loud_sorted
```

Out[20]:

	artist	track_name	loudness
1	Billie Eilish	everything i wanted	-14.454
2	Billie Eilish	bad guy	-10.965
3	Billie Eilish	lovely (with Khalid)	-10.109
4	JP Saxe	If the World Was Ending - feat. Julia Michaels	-10.086
5	Drake	Toosie Slide	-8.820
6	Powfu	death bed (coffee for your head)	-8.765
7	Travis Scott	HIGHEST IN THE ROOM	-8.764
8	Trevor Daniel	Falling	-8.756
9	Jawsh 685	Savage Love (Laxed - Siren Beat)	-8.520

4.3.4 Softest Tracks

Takeaway: 9 tracks fall below -8 dB. Softest is Billie Eilish's "everything i wanted" (-14.454 dB).

---

5) Track-Level Queries (Longest Track):

Which track is the longest?

4.3.5 Longest Track

Goal:
Find the longest track and display its duration in minutes : seconds instead of milliseconds.

In [21]:
```
longest = spotify_df.loc[
    spotify_df['duration_ms'].idxmax(),
    ['artist', 'track_name', 'duration_ms']
]

total_seconds = longest['duration_ms'] // 1000
minutes, seconds = divmod(total_seconds, 60)
print(f"Longest track: {longest['artist']} - {longest['track_name']} "
      f"({minutes}:{seconds:02d})")
```

```
Longest track: Travis Scott - SICKO MODE (5:12)
```

4.3.5 Longest Track

- **Takeaway:** The longest track is Travis Scott's "SICKO MODE" at 5 minutes 12 seconds.

---

6) Track-Level Queries (Shortest Track):

Which track is the shortest?

4.3.6 Shortest Track

Goal:
Identify which track in the Top 50 dataset has the shortest duration.

In [22]:
```
shortest = spotify_df.loc[
    spotify_df['duration_ms'].idxmin(),
    ['artist', 'track_name', 'duration_ms']
]

total_seconds = shortest['duration_ms'] // 1000
minutes, seconds = divmod(total_seconds, 60)
print(f"Shortest track: {shortest['artist']} - {shortest['track_name']} "
      f"({minutes}:{seconds:02d})")
```

```
Shortest track: 24kGoldn - Mood (feat. iann dior) (2:20)
```

4.3.6 Shortest Track

- **Takeaway:** The shortest track is 24kGoldn - Mood (feat. iann dior) at 2 minutes 20 seconds.

4.4 Genre Breakdown

1) Genre Breakdown (Most Popular):

Which genre is the most popular?

4.4.1 Most Popular Genre

Goal:

Determine which genre appears most frequently among the Top 50 tracks.

In [23]:
```
genre_counts = spotify_df['genre'].value_counts()
most_pop_genre = genre_counts.idxmax()
count = genre_counts.max()
print(f"Most popular genre: {most_pop_genre} ({count} tracks)")
```

```
Most popular genre: Pop (14 tracks)
```

4.4.1 Most Popular Genre

- **Takeaway:** Ranked by number of enteries in the Top 50, Pop comes out as most popular genre.

---

2) Genre Breakdown (Single-Song):

Which genres have just one song on the top 50?

4.4.2 Single-Song Genres

Goal:

Find all genres that appear exactly once in the Top 50 dataset.

In [24]:
```
genre_counts = spotify_df['genre'].value_counts()
single_song_genres = genre_counts[genre_counts == 1].index.tolist()
print(f"Total single song genres: {single_song_genres}")
```

```
Total single song genres: ['R&B/Hip-Hop alternative', 'Nu-disco', 'Pop/Soft
Rock', 'Pop rap', 'Hip-Hop/Trap', 'Dance-pop/Disco', 'Disco-pop', 'Dreampop/
Hip-Hop/R&B', 'Alternative/reggaeton/experimental', 'Chamber pop']
```

4.4.2 Single-Song Genres

- **Takeaway:** Ten genres each appear exactly once in the Top 50: R&B/Hip-Hop alternative, Nu-disco, Pop/Soft Rock, Pop rap, Hip-Hop/Trap, Dance-pop/Disco, Disco-pop, Dreampop/Hip-Hop/R&B, Alternative/reggaeton/experimental, and Chamber pop.

---

3) Genre Breakdown (Total Unique Genres):

How many distinct genres in total are represented in the top 50?

4.4.3 Total Unique Genres

Goal:
Determine how many distinct genres are represented among the Top 50 tracks.

In [25]:
```
total_genres = spotify_df['genre'].nunique()
print(f"Total unique genres: {total_genres}")
```

```
Total unique genres: 16
```

4.4.3 Total Unique Genres

- **Takeaway:** There are 16 distinct genres represented in the Top 50.

4.5 Correlation Overview

1) Correlation Overview (Positively Correlated):

Which features are strongly positively correlated?

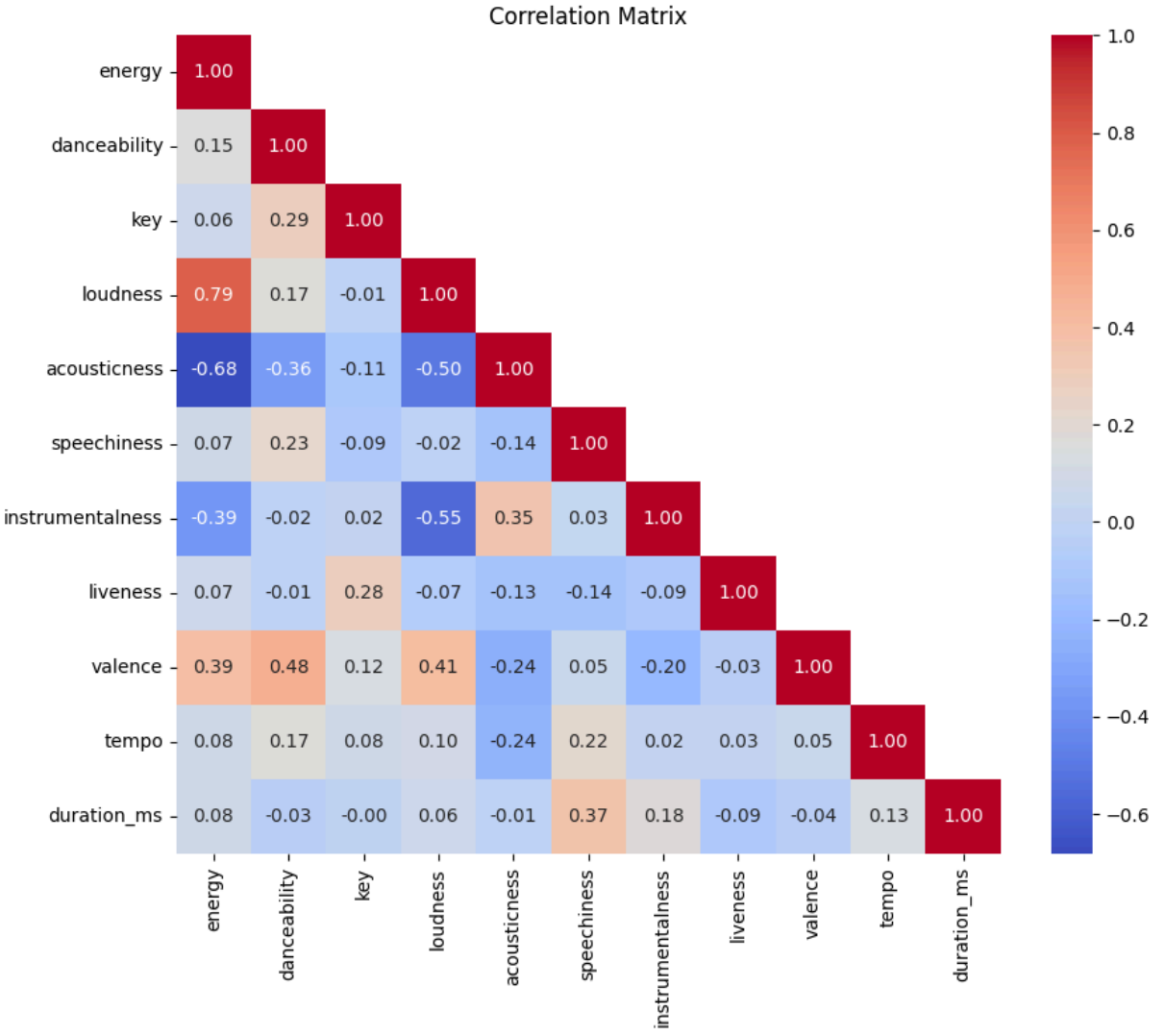
4.5.1 Strongly Positive Correlations

Goal:
Identify which pairs of numeric audio features exhibit a strong positive relationship (correlation r > 0.70).

In [26]:
```
correlation_matrix = spotify_df[num_cols].corr()

upper_triangle_mask = np.triu(
    np.ones_like(correlation_matrix, dtype=bool),
    k=1
)

plt.figure(figsize=(10, 8))
sns.heatmap(
    correlation_matrix,
    mask=upper_triangle_mask,
    annot=True,
    fmt=".2f",
    cmap="coolwarm"
)
plt.title("Correlation Matrix")
plt.show()
```



4.5.1 Strongly Positive Correlations

- **Takeaway:** The only pair of audio features with a correlation above 0.70 is energy and loudness (r ≈ 0.79).
- **Insight:** This shows that louder tracks feel more energetic.

---

2) Correlation Overview (Negatively Correlated):

Which features are strongly negatively correlated?

4.5.2 Strongly Negative Correlations

Goal:
Identify which pairs of numeric audio features exhibit a strong negative relationship (correlation r < -0.70).

4.5.2 Strongly Negative Correlations

- **Takeaway:** No feature pair crosses the -0.70 threshold. The strongest negative correlations are acousticness (vs energy r = -0.68, vs loudness r = -0.50) and instrumentalness & loudness (r = -0.55). We also see moderate negative correlations for instrumentalness & energy (r = -0.39) and acousticness & danceability (r = -0.36).
- **Insight:**
- Higher acousticness → lower energy: Acoustic tracks feel noticeably calmer.
- Higher instrumentalness → lower loudness: Instrumental focused songs play more softly.
- Higher acousticness → lower loudness: Acoustic tracks tend to be softer in volume.
- Higher instrumentalness → lower energy: Instrumental pieces feel less energetic.
- Higher acousticness → lower danceability: Acoustic tracks are less danceable.

---

3) Correlation Overview (Not Correlated):

Which features are not correlated?

4.5.3 Features That Are Not Correlated

Goal:
Identify which pairs of numeric audio features exhibit virtually no linear

relationship (|r| < 0.10).

4.5.3 Uncorrelated Features

- **Takeaway:** Key, speechiness, liveness, tempo and duration are near-zero correlation (|r| < 0.10) with energy, loudness, danceability, acousticness, instrumentalness, and valence.
- **Insight:** The statistical independence of these features means these dimensions add unique, non-redundant information. Knowing a track's key, length, or live/vocal content tells you nothing about its energy, volume, or danceability.

4.6 Genre Comparisons

1) Genre Comparisons (Danceability):

How does the danceability score compare between Pop, Hip-Hop/Rap, Dance/Electronic, and Alternative/Indie genres?

4.6.1 Danceability Comparison by Genre

Goal:
Compare how danceable the Top 50 hits are across four genres. Pop, Hip-Hop/Rap, Dance/Electronic and Alternative/Indie by looking at the median, lowest and highest danceability scores.

In [27]:
```
genres = ['Pop', 'Hip-Hop/Rap', 'Dance/Electronic', 'Alternative/Indie']

genre_subset = spotify_df[spotify_df['genre'].isin(genres)]

dance_stats = (
    genre_subset
      .groupby('genre')['danceability']
      .agg(
          median='median',
          minimum='min',
          maximum='max'
      )
      .loc[genres]
)
dance_stats
```

Out[27]:

genre	median	minimum	maximum
Pop	0.690	0.464	0.806
Hip-Hop/Rap	0.774	0.598	0.896
Dance/Electronic	0.785	0.647	0.880
Alternative/Indie	0.663	0.459	0.862

4.6.1 Danceability Comparison by Genre

- **Takeaway:** Dance/Electronic and Hip-Hop/Rap: Almost every track gets you moving. Pop: Keeps a steady, dance friendly beat without extremes. Alternative/Indie: Some songs are laid-back and mellow, others are full on dance tunes.

---

2) Genre Comparisons (Loudness):

How does the loudness score compare between Pop, Hip-Hop/Rap, Dance/Electronic, and Alternative/Indie genres?

4.6.2 Loudness Comparison by Genre

Goal:
Compare how loud the Top 50 hits are across four genres. Pop, Hip-Hop/Rap, Dance/Electronic, and Alternative/Indie by examining their median, quietest, and loudest scores.

In [28]:
```
loud_stats = (
    genre_subset
      .groupby('genre')['loudness']
      .agg(
          median='median',
          minimum='min',
          maximum='max'
      )
      .loc[genres]
)
loud_stats
```

Out[28]:

genre	median	minimum	maximum
Pop	-6.6445	-14.454	-3.280
Hip-Hop/Rap	-7.6480	-8.820	-3.370
Dance/Electronic	-5.4570	-7.567	-3.756
Alternative/Indie	-5.2685	-6.401	-4.746

4.6.2 Loudness Comparison by Genre

- **Takeaway:** Dance/Electronic: Always loud and punchy. Hip-Hop/Rap: Generally softer overall. Pop: Tracks shift from soft, quiet verses to full blast choruses. Alternative/Indie: Stays loud throughout with hardly any quiet parts.

---

3) Genre Comparisons (Acousticness):

How does the acousticness score compare between Pop, Hip-Hop/Rap, Dance/Electronic, and Alternative/Indie genres?

4.6.3 Acousticness Comparison by Genre

Goal: Compare how acoustic the Top 50 hits are across four genres. Pop, Hip-Hop/Rap, Dance/Electronic, and Alternative/Indie by examining their median, lowest, and highest acousticness scores.

In [29]:
```
acoustic_stats = (
    genre_subset
      .groupby('genre')['acousticness']
      .agg(
          median='median',
          minimum='min',
          maximum='max'
      )
      .loc[genres]
)
acoustic_stats
```

Out[29]:

genre	median	minimum	maximum
Pop	0.2590	0.02100	0.902
Hip-Hop/Rap	0.1450	0.00513	0.731
Dance/Electronic	0.0686	0.01370	0.223
Alternative/Indie	0.6460	0.29100	0.751

4.6.3 Acousticness Comparison by Genre

- **Takeaway:** Alternative/Indie: Songs almost always use real instruments. Dance/Electronic: Songs rarely use live instruments and rely almost entirely on synthetic sounds. Pop: Tracks blend both real, “live” instruments (like guitars or pianos) and electronic sounds (synths, drum machines) in the same song. Hip-Hop/Rap: Songs stick mostly to electronic beats with a few live sounds.

Spotify Tracks Analysis: Final Summary

In line with the stated aim to uncover the audio characteristics that underpin a successful hit here are the four key conclusions:

- **High energy, loud production drives hit potential.** The very strong positive correlation between energy and loudness (r ≈ 0.79) indicates that tracks with punchy, high energy mixes are far more likely to break into the Top 50.
- **Mainstream pop stylings and star power matter.** With Pop accounting for over a quarter of the list and artists like Dua Lipa, Billie Eilish, and Travis Scott each securing three entries, the analysis shows that familiar pop formats and established names consistently translate into chart success.
- **Genre tailored audio profiles align with listener expectations.** Dance/Electronic and Hip-Hop/Rap tracks top the danceability rankings, while Alternative/Indie songs lead in acousticness. Demonstrating that hits are most successful when they accentuate the features fans expect from a given genre.
- **Chart topping tracks span a wide feature spectrum.** Although there are clear “sweet spots,” hits range from low danceability ballads (e.g. 0.351) to high energy anthems (e.g. 0.935) and run from ~2 min 20 sec to ~5 min

12 sec - showing that flexibility within broad feature ranges still supports mass appeal.

Further Improvements & Research Directions:

1. Show how danceability, loudness, and acousticness overlap and differ between genres using ridgeline plots.
2. Analyze the most frequently used musical notes (e.g. C, D, E) in top songs and compare key popularity across genres such as Pop and Hip-Hop via bar charts.
3. For artists with multiple hit songs, create a simple chart comparing the energy, mood, and loudness of their hits to reveal each artist's signature sound traits.
4. Add streaming counts to the dataset to truly measure each song's popularity.