In [1]:
```
#Import the relevant libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
file_path = "Spotify Data.csv"
sdata = pd.read_csv(file_path)

# Display basic information about the dataset
df_info = sdata.info()
df_head = sdata.head()

df_info, df_head
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 114000 entries, 0 to 113999
Data columns (total 20 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   track_id          114000 non-null  object 
 1   artists           113999 non-null  object 
 2   album_name        113999 non-null  object 
 3   track_name        113999 non-null  object 
 4   popularity        114000 non-null  int64  
 5   duration_ms       114000 non-null  int64  
 6   explicit          114000 non-null  bool   
 7   danceability      114000 non-null  float64
 8   energy            114000 non-null  float64
 9   key               114000 non-null  int64  
 10  loudness          114000 non-null  float64
 11  mode              114000 non-null  int64  
 12  speechiness       114000 non-null  float64
 13  acousticness      114000 non-null  float64
 14  instrumentalness  114000 non-null  float64
 15  liveness          114000 non-null  float64
 16  valence           114000 non-null  float64
 17  tempo             114000 non-null  float64
 18  time_signature    114000 non-null  int64  
 19  track_genre       114000 non-null  object 
dtypes: bool(1), float64(9), int64(5), object(5)
memory usage: 16.6+ MB
```

Out[1]:
(None,

	track_id	artists	album_name	track_name	popularity	duration_ms	explicit	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	time_signature	track_genre
0	5SuOikwiRyPMVoIQDJUgSV	Gen Hoshino	Comedy	Comedy	73	230666	False	0.676	0.4610	1	-6.746	0	0.1430	0.0322	0.000001	0.3580	0.715	87.917	4	acoustic
1	4qPNDBW1i3p13qLCt0Ki3A	Ben Woodward	Ghost (Acoustic)	Ghost - Acoustic	55	149610	False	0.420	0.1660	1	-17.235	1	0.0763	0.9240	0.000006	0.1010	0.267	77.489	4	acoustic
2	1iJBSr7s7jYXzM8EGcbK5b	Ingrid Michaelson;ZAYN	To Begin Again	To Begin Again	57	210826	False	0.438	0.3590	0	-9.734	1	0.0557	0.2100	0.000000	0.1170	0.120	76.332	4	acoustic
3	6lfxq3CG4xtTiEg7opyCyx	Kina Grannis	Crazy Rich Asians (Original Motion Picture Sou...	Can't Help Falling In Love	71	201933	False	0.266	0.0596	0	-18.515	1	0.0363	0.9050	0.000071	0.1320	0.143	181.740	3	acoustic
4	5vjLSffimiIP26QG5WcN2K	Chord Overstreet	Hold On	Hold On	82	198853	False	0.618	0.4430	2	-9.681	1	0.0526	0.4690	0.000000	0.0829	0.167	119.949	4	acoustic

)

In [2]:
```
# 1. Standardizing simple text columns, not names (strip spaces, lowercase)
sdata['track_genre'] = sdata['track_genre'].str.strip().str.lower()

#2. Convert popularity to binary labels (1 = Popular, 0 = Not Popular)
sdata["popularity_label"] = (sdata["popularity"] > 80).astype(int)

#3 Handle missing values (drop rows with missing artist, album_name, or track_name)
sdata = sdata.dropna()

#4. Removing duplicates
sdata = sdata.drop_duplicates()

#5. Normalize loudness (since it's in negative decibels)
```

```
sdata["loudness"] = (sdata["loudness"] - sdata["loudness"].min()) / (sdata["loudness"].max() - sdata["loudness"].min())

#6. Normalize tempo
sdata["tempo"] = (sdata["tempo"] - sdata["tempo"].min()) / (sdata["tempo"].max() - sdata["tempo"].min())

#7. Removing duplicates
sdata_clean = sdata.drop_duplicates()

# 8. Confirming dataset cleanup
sdata_clean.info()
sdata_clean.head()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 113549 entries, 0 to 113999
Data columns (total 21 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   track_id          113549 non-null  object 
 1   artists           113549 non-null  object 
 2   album_name        113549 non-null  object 
 3   track_name        113549 non-null  object 
 4   popularity        113549 non-null  int64  
 5   duration_ms       113549 non-null  int64  
 6   explicit          113549 non-null  bool   
 7   danceability      113549 non-null  float64
 8   energy            113549 non-null  float64
 9   key               113549 non-null  int64  
 10  loudness          113549 non-null  float64
 11  mode              113549 non-null  int64  
 12  speechiness       113549 non-null  float64
 13  acousticness      113549 non-null  float64
 14  instrumentalness  113549 non-null  float64
 15  liveness          113549 non-null  float64
 16  valence           113549 non-null  float64
 17  tempo             113549 non-null  float64
 18  time_signature    113549 non-null  int64  
 19  track_genre       113549 non-null  object 
 20  popularity_label  113549 non-null  int32  
dtypes: bool(1), float64(9), int32(1), int64(5), object(5)
memory usage: 17.9+ MB
```

Out[2]:

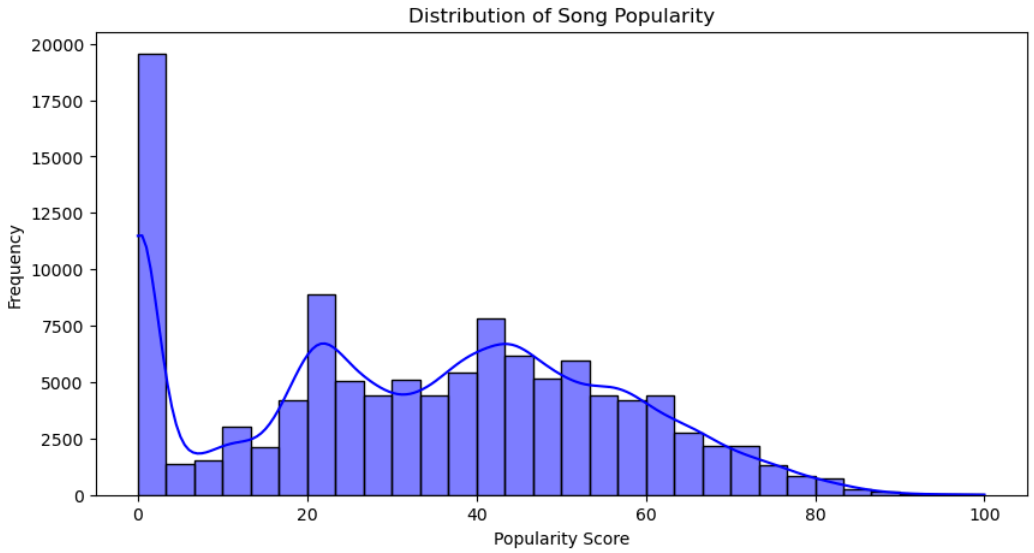
	track_id	artists	album_name	track_name	popularity	duration_ms	explicit	danceability	energy	key	...	mode	speechiness	acous
0	5SuOikwiRyPMVoIQDJUgSV	Gen Hoshino	Comedy	Comedy	73	230666	False	0.676	0.4610	1	...	0	0.1430	
1	4qPNDBW1i3p13qLCt0Ki3A	Ben Woodward	Ghost (Acoustic)	Ghost - Acoustic	55	149610	False	0.420	0.1660	1	...	1	0.0763	
2	1iJBSr7s7jYXzM8EGcbK5b	Ingrid Michaelson;ZAYN	To Begin Again	To Begin Again	57	210826	False	0.438	0.3590	0	...	1	0.0557	
3	6lfxq3CG4xtTiEg7opyCyx	Kina Grannis	Crazy Rich Asians (Original Motion Picture Sou...	Can't Help Falling In Love	71	201933	False	0.266	0.0596	0	...	1	0.0363	
4	5vjLSffimiIP26QG5WcN2K	Chord Overstreet	Hold On	Hold On	82	198853	False	0.618	0.4430	2	...	1	0.0526	

5 rows × 21 columns

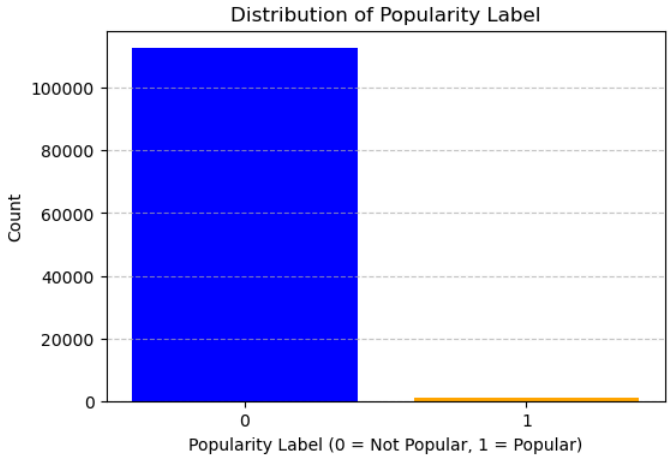
In [3]:
```
# Features to analyze (song metadata that is universal)
features = ["danceability", "energy", "valence", "loudness", "acousticness", "tempo"]

# Finding Min and Max for each feature
high_low_values = {}
for feature in features:
    high_low_values[feature] = {
        "Minimum": sdata_clean[feature].min(),
        "Maximum": sdata_clean[feature].max()
    }

# Convert to DataFrame for better readability
high_low_df = pd.DataFrame(high_low_values)

# Display the results
print(high_low_df)
```

```
         danceability  energy  valence  loudness  acousticness  tempo
Minimum         0.000     0.0    0.000       0.0         0.000    0.0
Maximum         0.985     1.0    0.995       1.0         0.996    1.0
```

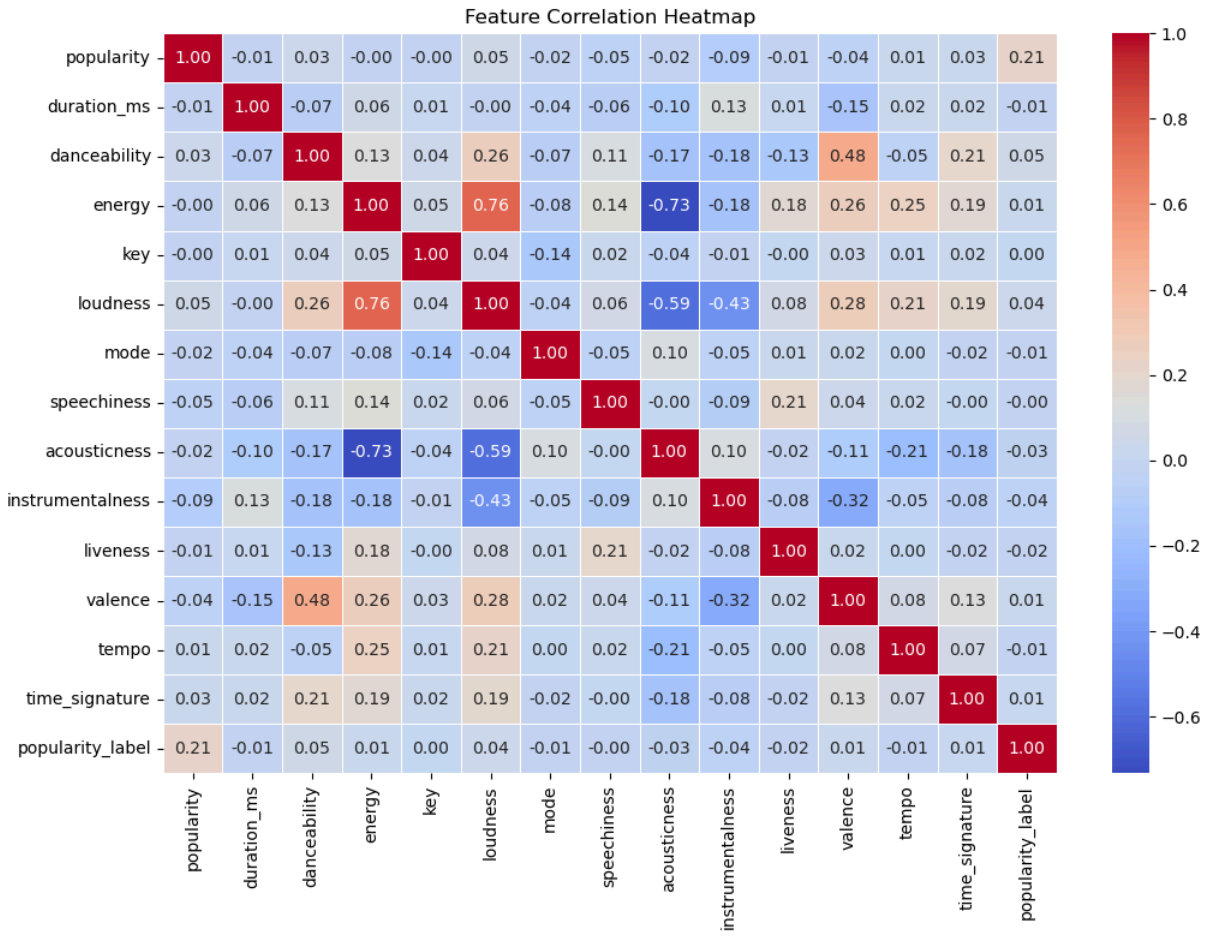
In [4]:
```
# Plot distribution of popularity scores to show how many are
plt.figure(figsize=(10, 5))
sns.histplot(sdata_clean["popularity"], bins=30, kde=True, color="blue")
plt.title("Distribution of Song Popularity")
plt.xlabel("Popularity Score")
plt.ylabel("Frequency")
plt.show()
```



In [5]:
```
# Assuming 'popularity_code' is a binary variable (0 or 1)
if 'popularity_label' in sdata_clean.columns:
    # Count occurrences of each category (0 and 1)
    popularity_counts = sdata_clean['popularity_label'].value_counts()

    # Create bar plot
    plt.figure(figsize=(6, 4))
    plt.bar(popularity_counts.index.astype(str), popularity_counts.values, color=['blue', 'orange'])
    plt.xlabel("Popularity Label (0 = Not Popular, 1 = Popular)")
    plt.ylabel("Count")
    plt.title("Distribution of Popularity Label")
    plt.grid(axis='y', linestyle='--', alpha=0.7)
    plt.show()
else:
    print("Column 'popularity_label' not found in the dataset. Please ensure it exists.")
```



In [6]:
```
# Drop non-numeric columns
numeric_data = sdata_clean.select_dtypes(include=['number'])

# Compute correlation matrix
corr_matrix = numeric_data.corr()

# Generate heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, annot=True, fmt=".2f", cmap="coolwarm", linewidths=0.5)
plt.title("Feature Correlation Heatmap")
plt.show()
```


Feature Correlation Heatmap
popularity 1.00 -0.01 0.03 -0.00 -0.00 0.05 -0.02 -0.05 -0.02 -0.09 -0.01 -0.04 0.01 0.03 0.21
duration_ms -0.01 1.00 -0.07 0.06 0.01 -0.00 -0.04 -0.06 -0.10 0.13 0.01 -0.15 0.02 0.02 -0.01
danceability 0.03 -0.07 1.00 0.13 0.04 0.26 -0.07 0.11 -0.17 -0.18 -0.13 0.48 -0.05 0.21 0.05
energy -0.00 0.06 0.13 1.00 0.05 0.76 -0.08 0.14 -0.73 -0.18 0.18 0.26 0.25 0.19 0.01
key -0.00 0.01 0.04 0.05 1.00 0.04 -0.14 0.02 -0.04 -0.01 -0.00 0.03 0.01 0.02 0.00
loudness 0.05 -0.00 0.26 0.76 0.04 1.00 -0.04 0.06 -0.59 -0.43 0.08 0.28 0.21 0.19 0.04
mode -0.02 -0.04 -0.07 -0.08 -0.14 -0.04 1.00 -0.05 0.10 -0.05 0.01 0.02 0.00 -0.02 -0.01
speechiness -0.05 -0.06 0.11 0.14 0.02 0.06 -0.05 1.00 -0.00 -0.09 0.21 0.04 0.02 -0.00 -0.00
acousticness -0.02 -0.10 -0.17 -0.73 -0.04 -0.59 0.10 -0.00 1.00 0.10 -0.02 -0.11 -0.21 -0.18 -0.03
instrumentalness -0.09 0.13 -0.18 -0.18 -0.01 -0.43 -0.05 -0.09 0.10 1.00 -0.08 -0.32 -0.05 -0.08 -0.04
liveness -0.01 0.01 -0.13 0.18 -0.00 0.08 0.01 0.21 -0.02 -0.08 1.00 0.02 0.00 -0.02 -0.02
valence -0.04 -0.15 0.48 0.26 0.03 0.28 0.02 0.04 -0.11 -0.32 0.02 1.00 0.08 0.13 0.01
tempo 0.01 0.02 -0.05 0.25 0.01 0.21 0.00 0.02 -0.21 -0.05 0.00 0.08 1.00 0.07 -0.01
time_signature 0.03 0.02 0.21 0.19 0.02 0.19 -0.02 -0.00 -0.18 -0.08 -0.02 0.13 0.07 1.00 0.01
popularity_label 0.21 -0.01 0.05 0.01 0.00 0.04 -0.01 -0.00 -0.03 -0.04 -0.02 0.01 -0.01 0.01 1.00
popularity
duration_ms
danceability
energy
key
loudness
mode
speechiness
acousticness
instrumentalness
liveness
valence
tempo
time_signature
popularity_label
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

In [ ]: