PROJECT I REPORT: Billboard Top Hit Songs Analysis & Visualization

Nguyen Tung Lam - V202100571
Nguyen Nhat Minh - V202100570
Nguyen Truong Tung - V202100542

1 Introduction

Music is a universal language that transcends geographical, cultural, and generational boundaries. It plays a central role in human expression, identity, and social interaction. As both an art form and a reflection of societal values, music evolves in tandem with cultural shifts, technological advances, and changes in public taste. Understanding how music trends have changed over time offers valuable insight into broader patterns in entertainment, culture, and technology.

In this project, we analyze over sixty years of Billboard Hot 100 chart data to examine trends in popular music. The Billboard Hot 100 is a widely accepted benchmark for song popularity in the United States, incorporating data from radio airplay, sales, and streaming platforms. Our analysis aims to uncover how musical features, genres, and artist prominence have evolved over time, and what these changes reveal about the shifting landscape of popular music.

We utilized a publicly available dataset originally compiled by other authors. The dataset was created using a custom Scrapy web crawler to collect weekly Billboard Hot 100 chart data from January 4, 1960, to April 2, 2022, resulting in **324,787** records. To focus on influential songs, the author selected the top 50 tracks from each year based on their chart longevity, reducing the dataset to 3,150 entries. These records were then enriched using the Spotify for Developers API, adding **21** musical and metadata attributes, including danceability, tempo, acousticness, valence, and genre. This comprehensive dataset offers a rich foundation for multi-dimensional analysis of popular music trends across decades. The full dataset resources are publicly available at: `https://github.com/Ikea-179/Top-Hit-Songs-Data-Analysis-and-Visualization/tree/main/Datasets`, and you can find attributes in Table 1.

Column	Description
`date`	The date when the song appeared on the Billboard chart.
`title`	The title of the song.
`artist`	The name of the performing artist.
`year`	The year of the song's chart performance.
`rank`	The song's ranking on the chart for the given date.
`last_week`	The song's ranking in the previous week.
`peak`	The highest rank the song achieved on the Billboard chart.
`weeks`	The total number of weeks the song remained on the chart.
`danceability`	A measure of how suitable a track is for dancing (0 to 1).
`energy`	A measure of intensity and activity (0 to 1).
`key`	The musical key in which the song is played (0–11, corresponding to C to B).
`loudness`	The overall volume of the track in decibels (dB).
`mode`	The modality of the song (0 = Minor, 1 = Major).
`speechiness`	The presence of spoken words in a track (0 to 1).
`acousticness`	The likelihood of the track being acoustic (0 to 1).
`instrumentalness`	The likelihood of the track being purely instrumental (0 to 1).
`liveness`	Detects the presence of a live audience in the recording (0 to 1).
`valence`	The musical positiveness of the track (0 to 1).
`tempo`	The estimated beats per minute (BPM) of the track.
`duration_ms`	The length of the song in milliseconds.
`genres`	A list of genres associated with the song.
`genre_encoding`	A categorical encoding of the song's genre for classification tasks.

Table 1: Summary of Billboard Hot 100 dataset attributes

2 Research Question 1: What are the key audio features that define top-charting songs over the past decade?

2.1 Introduction

Over the past decade, the landscape of **popular music** has continually evolved, influenced by shifts in **listener preferences**, **cultural moments**, and **technological advancements in music production**. This evolution is often reflected in the **sonic characteristics** of top-charting songs—elements that can now be quantified through **audio features** extracted from music data. By examining these features, we can begin to answer the **first research question**. Understanding these trends not only sheds light on what makes a song **resonate with wide audiences** but also offers insights for **artists**, **producers**, and **marketers** aiming to craft the next big hit.

To explore this question, we will utilize a dataset containing detailed information on songs that have appeared on **top music charts** in the past ten years. Critical components of the dataset include audio features such as **danceability**, **energy**, **valence**, **tempo**, **acousticness**, **instrumentalness**, **speechiness**, and **liveness**, along with metadata such as **title**, **artist**, **date**, **rank**, **peak**, and **weeks on the chart**. By analyzing these metrics across a wide temporal range, we aim to **identify patterns** and pinpoint which traits are most consistently present in the songs that **rise to the peak**. This exploration is not only **academically interesting** but also **personally engaging**, as it connects **data science** with the **universal language of music**.

2.2 Approach

Line Chart: Temporal Trends of Audio Features

A line chart is optimal for tracking temporal trends and long-term changes in data across a continuous variable—in this case, time. It allows us to:

- Observe how features like *valence*, *tempo*, or *loudness* have increased or decreased across decades.
- Discover patterns such as the rise of high-energy music in the 2010s, or a possible dip in valence (happier songs) during periods of societal tension.
- Easily detect peaks, dips, or inflection points, providing evidence of musical or cultural shifts (e.g., the disco era, rise of hip hop, EDM boom, etc.).

Boxplot: Distribution of Audio Features Across Years

X-axis: Year (grouped by decade or individual year from 2010 to 2020)
Y-axis: Audio feature value (e.g., danceability, energy, valence, etc.)

A boxplot is ideal for visualizing the distribution, central tendency, and variability of audio features over time. It shows medians, interquartile ranges, and potential outliers, allowing us to:

- Compare how different features vary across years.
- Observe whether certain features consistently show higher or lower values.
- Identify shifts in distribution, which can reflect evolving production trends or listener preferences.

This is particularly useful for determining whether, for example, *danceability* or *valence* became more prominent in the 2010s versus earlier years.

2.3 Analysis & Discussion

```python
# Energy Trend Chart
feature = 'energy'
title = 'Energy'
ylabel = 'Score (0-1)'
color = palette[0]

```

```python
x = yearly_avg.index.year
x_num = np.arange(len(x))
y = yearly_avg[feature]
coeffs = np.polyfit(x_num, y, 1)
trend_line = np.poly1d(coeffs)

plt.figure(figsize=(12, 4))
sns.lineplot(x=yearly_avg.index, y=y, color=color, linewidth=2.5, marker='o',
             markersize=8, markeredgecolor='white', markeredgewidth=1, label='Yearly Average')
plt.plot(yearly_avg.index, trend_line(x_num), '--', color='darkgray', linewidth=2,
         label=f'Trend (slope: {coeffs[0]:.3f})')

plt.title(f'Yearly Trend of {title}', fontsize=14, fontweight='bold', pad=15)
plt.ylabel(ylabel)
plt.grid(True, linestyle='--', alpha=0.7)
plt.legend()

last_val = y.iloc[-1]
plt.annotate(f'{last_val:.2f}', xy=(yearly_avg.index[-1], last_val),
             xytext=(10, 0), textcoords='offset points', ha='left', va='center',
             fontsize=11, bbox=dict(boxstyle='round,pad=0.3', fc='white', ec=color, lw=2))

plt.tight_layout()
plt.show()
```

Listing 1: Energy Trend Chart

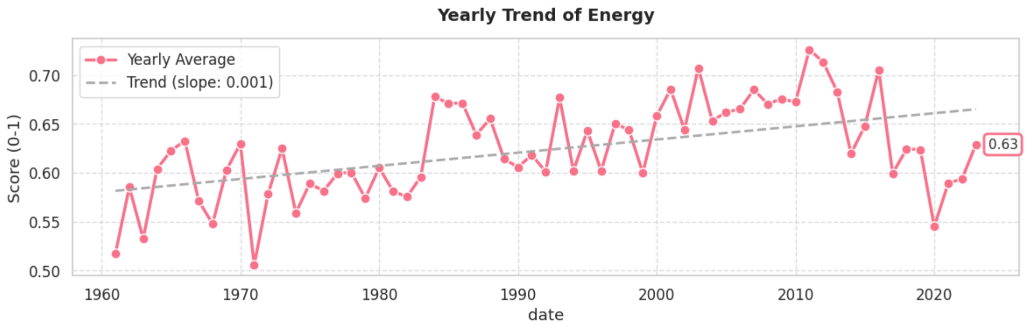

Figure 1: Output plot for the Energy Trend Chart.

```python
# Valence Trend Chart
feature = 'valence'
title = 'Valence (Positivity)'
ylabel = 'Score (0-1)'
color = sns.color_palette("husl", 3)[1]

x = yearly_avg.index.year
x_num = np.arange(len(x))
y = yearly_avg[feature]
coeffs = np.polyfit(x_num, y, 1)
trend_line = np.poly1d(coeffs)

plt.figure(figsize=(12, 4))
sns.lineplot(x=yearly_avg.index, y=y, color=color, linewidth=2.5, marker='o',
             markersize=8, markeredgecolor='white', markeredgewidth=1, label='Yearly Average')
plt.plot(yearly_avg.index, trend_line(x_num), '--', color='darkgray', linewidth=2,
         label=f'Trend (slope: {coeffs[0]:.3f})')

plt.title(f'Yearly Trend of {title}', fontsize=14, fontweight='bold', pad=15)
plt.ylabel(ylabel)
plt.grid(True, linestyle='--', alpha=0.7)
plt.legend()
```

```python

last_val = y.iloc[-1]
plt.annotate(f'{last_val:.2f}', xy=(yearly_avg.index[-1], last_val),
             xytext=(10, 0), textcoords='offset points', ha='left', va='center',
             fontsize=11, bbox=dict(boxstyle='round,pad=0.3', fc='white', ec=color, lw=2)
    )

plt.tight_layout()
plt.show()
```

Listing 2: Valence Trend Chart

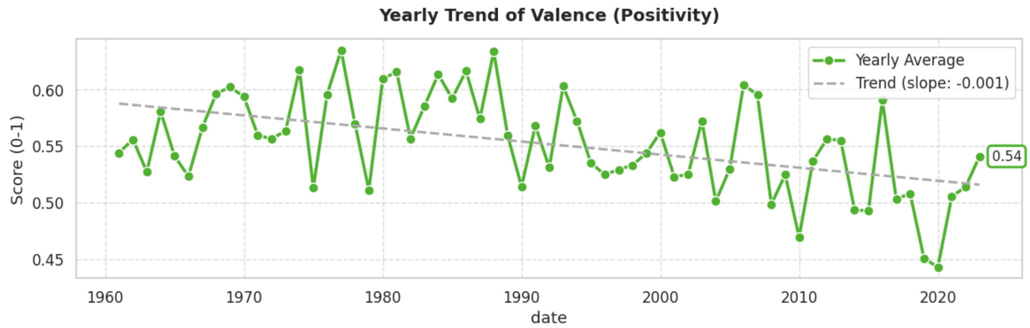

Figure 2: Output plot for the Valence Trend Chart.

```python
# Danceability Trend Chart
feature = 'danceability'
title = 'Danceability'
ylabel = 'Score (0-1)'
color = sns.color_palette("husl", 3)[2]

x = yearly_avg.index.year
x_num = np.arange(len(x))
y = yearly_avg[feature]
coeffs = np.polyfit(x_num, y, 1)
trend_line = np.poly1d(coeffs)

plt.figure(figsize=(12, 4))
sns.lineplot(x=yearly_avg.index, y=y, color=color, linewidth=2.5, marker='o',
             markersize=8, markeredgecolor='white', markeredgewidth=1, label='Yearly
    Average')
plt.plot(yearly_avg.index, trend_line(x_num), '--', color='darkgray', linewidth=2,
         label=f'Trend (slope: {coeffs[0]:.3f})')

plt.title(f'Yearly Trend of {title}', fontsize=14, fontweight='bold', pad=15)
plt.ylabel(ylabel)
plt.grid(True, linestyle='--', alpha=0.7)
plt.legend()

last_val = y.iloc[-1]
plt.annotate(f'{last_val:.2f}', xy=(yearly_avg.index[-1], last_val),
             xytext=(10, 0), textcoords='offset points', ha='left', va='center',
             fontsize=11, bbox=dict(boxstyle='round,pad=0.3', fc='white', ec=color, lw=2)
    )

plt.tight_layout()
plt.show()
```

Listing 3: Danceability Trend Chart

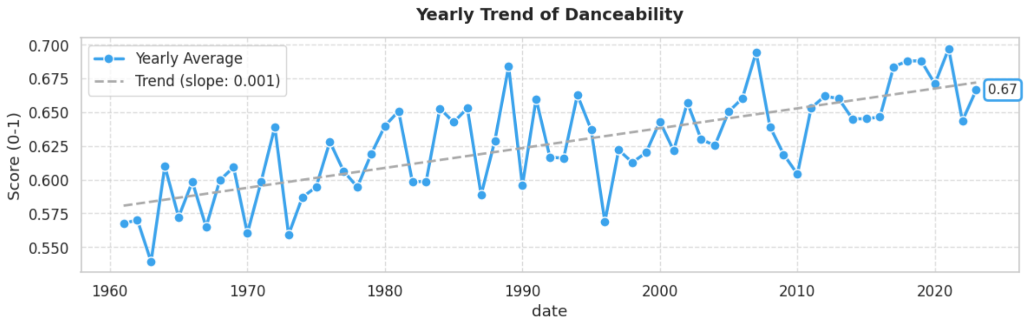

Figure 3: Output plot for the Danceability Trend Chart.

The graph presents long-term trends in three key audio features—danceability, energy, and valence—as measured across top-charting songs over several decades. The data reveal a consistent upward trajectory in all three dimensions, indicating a progressive transformation in the sonic attributes of popular music.

The increase in danceability suggests an enhanced focus on rhythm and movement-oriented production, likely corresponding with the emergence and proliferation of genres such as disco, electronic, and contemporary pop, which emphasize beat-driven musical structures. Similarly, the observed rise in energy reflects a broader shift toward more dynamic and high-intensity compositions. This trend may be attributed to advancements in music production technology as well as changing listener preferences favoring more stimulating and impactful auditory experiences.

In parallel, the steady growth in valence, a measure of the emotional positivity or brightness of a track, implies that popular music has become increasingly upbeat and emotionally expressive over time. This phenomenon may mirror broader cultural and commercial inclinations toward music that conveys optimism or offers escapism in response to socio-cultural contexts.

Collectively, these trends underscore the evolving nature of mainstream music, shaped by the interplay of cultural dynamics, technological innovation, and shifting audience expectations. The increasing prominence of energetic, danceable, and emotionally positive music highlights a broader transformation in the auditory landscape of popular music over the past several decades.

```

# Select most meaningful features for distribution analysis
features = ['danceability', 'energy', 'acousticness', 'valence']

# 1. Combined Histograms with KDE
plt.figure(figsize=(14, 10))
for i, feature in enumerate(features, 1):
    plt.subplot(2, 2, i)
    sns.histplot(
        data=df_billboard,
        x=feature,
        hue='tier',
        element='step',
        stat='density',
        common_norm=False,
        kde=True,
        alpha=0.3,
        hue_order=['Tier 1', 'Tier 2'],
        palette=['#4C72B0', '#DD8452']
    )
    plt.title(f'{feature.capitalize()} Distribution', fontweight='bold')
    plt.xlabel('')
    plt.ylabel('Density')
    plt.legend(title='Tier', frameon=True)

    # Add mean markers
    for tier, color in zip(['Tier 1', 'Tier 2'], ['#4C72B0', '#DD8452']):
        mean_val = comparison.loc['mean', (feature, tier)]
        plt.axvline(mean_val, color=color, linestyle='--', linewidth=1.5)
```

```python
        plt.text(mean_val, plt.ylim()[1]*0.9, f'{mean_val:.2f}',
                color=color, ha='center', fontweight='bold')

plt.suptitle('Feature Distribution Comparison: Tier 1 vs Tier 2',
             y=1.02, fontsize=16, fontweight='bold')
plt.tight_layout()
plt.show()

# 2. Enhanced Boxplots
plt.figure(figsize=(12, 6))
sns.boxplot(
    data=pd.melt(df_billboard[df_billboard['tier'].isin(['Tier 1', 'Tier 2'])],
                id_vars=['tier'],
                value_vars=features),
    x='variable',
    y='value',
    hue='tier',
    hue_order=['Tier 1', 'Tier 2'],
    palette=['#4C72B0', '#DD8452'],
    linewidth=1,
    width=0.6,
    showfliers=False  # Cleaner visualization without outliers
)

# Add swarm plot for actual data points
sns.swarmplot(
    data=pd.melt(df_billboard[df_billboard['tier'].isin(['Tier 1', 'Tier 2'])],
                id_vars=['tier'],
                value_vars=features),
    x='variable',
    y='value',
    hue='tier',
    hue_order=['Tier 1', 'Tier 2'],
    palette=['#2166ac', '#b35806'],
    size=2,
    alpha=0.4,
    dodge=True
)

# Customize plot
plt.title('Feature Distribution: Boxplot Comparison',
          pad=20, fontsize=14, fontweight='bold')
plt.xlabel('Feature', labelpad=10)
plt.ylabel('Value', labelpad=10)
plt.xticks(rotation=45)
plt.legend(bbox_to_anchor=(1.05, 1), title='Tier')
plt.grid(axis='y', alpha=0.3)

# Add annotations
medians = df_billboard.groupby('tier')[features].median().T
for i, feature in enumerate(features):
    for j, tier in enumerate(['Tier 1', 'Tier 2']):
        plt.text(i + (-0.18 if j == 0 else 0.18),
                medians.loc[feature, tier],
                f'{medians.loc[feature, tier]:.2f}',
                color='white',
                fontsize=9,
                ha='center',
                va='center',
                bbox=dict(facecolor='#4C72B0' if j == 0 else '#DD8452',
                          boxstyle='round,pad=0.3'))

plt.tight_layout()
plt.show()
```

Listing 4: Feature Distribution Comparison

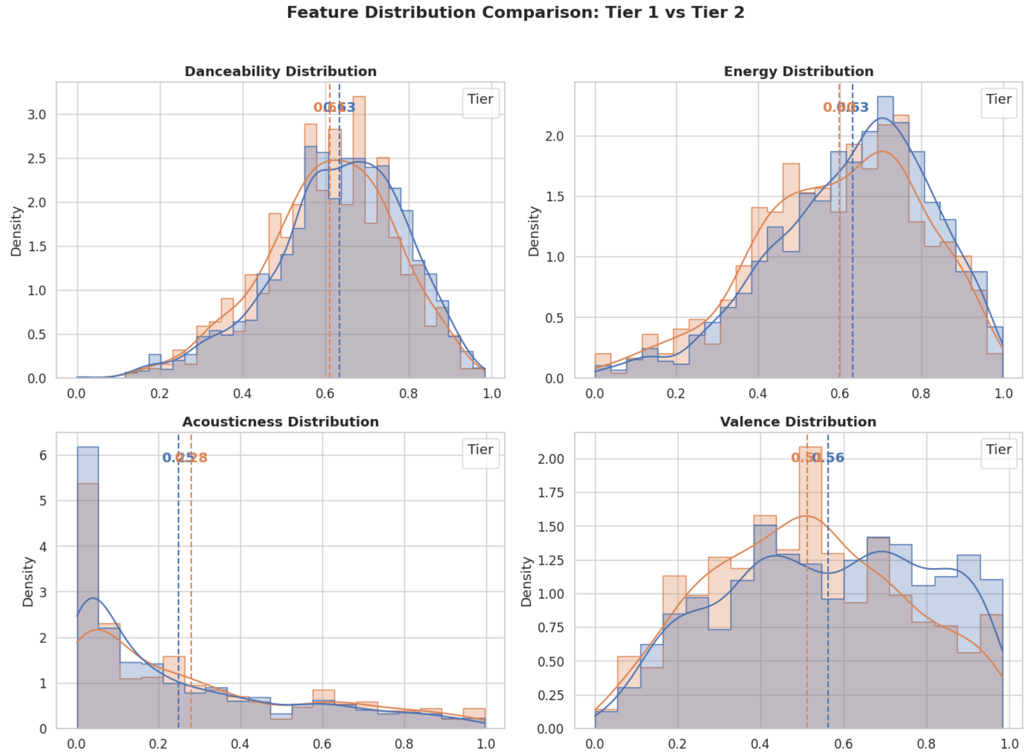

Figure 4: Output plot for Feature Distribution Comparison.

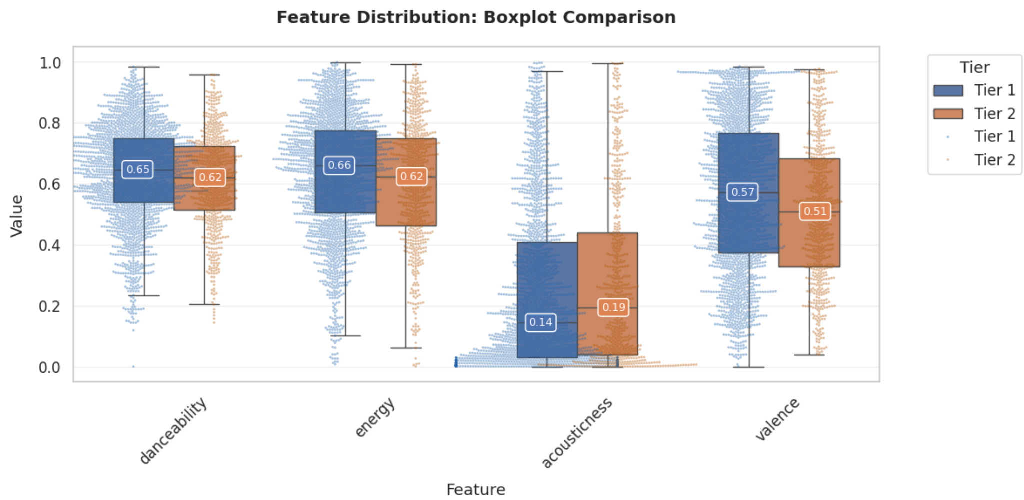

Figure 5: Output plot for Feature Distribution Comparison (Boxplot).

This analysis investigates the distinguishing audio characteristics of Tier 1 and Tier 2 songs, where Tier 1 songs are defined as those reaching a top 10 peak position and remaining on the charts for at least 10 weeks, while Tier 2 songs peaked within the top 50 and sustained a minimum 5-week chart presence. The results, drawn from both kernel density plots and boxplot distributions, reveal consistent and measurable differences across four core audio features: *danceability*, *energy*, *acousticness*, and *valence*.

Tier 1 songs exhibit higher average *danceability* (0.65) and *energy* (0.66) than Tier 2 tracks (0.62 for both features), suggesting that commercially dominant songs tend to favor upbeat, rhythm-driven production styles. These songs are likely optimized for physical engagement (e.g., dancing, workout playlists), which is congruent with their elevated presence in streaming and social media platforms where shareability and replayability are critical. Similarly, *valence*—a measure of musical positivity—is moderately higher in Tier 1 songs (0.57 vs. 0.51), indicating a tendency toward more emotionally positive or uplifting content in songs that achieve long-term and high-ranking chart success. This supports prior findings in music psychology that associate positive affect with broader audience appeal and repeated listening behavior.

In contrast, *acousticness* is slightly higher in Tier 2 songs (0.19) compared to Tier 1 (0.14), suggesting that less commercially dominant songs may lean more toward acoustic instrumentation or organic sound profiles. This could reflect genre differences—such as singer-songwriter, indie folk, or soft rock—that have modest but niche followings. The higher acousticness among Tier 2 songs might also imply reduced mainstream appeal in the current pop-dominated market, where electronic production and percussive intensity are more prominent. These differences highlight the role of production choices and genre alignment in influencing not just whether a song charts, but how high and how long it remains popular.

Overall, these results suggest that the most successful songs—those that break into the top 10 and sustain their popularity over time—tend to be characterized by a combination of high rhythmic drive, emotional positivity, and lower acoustic content. While the margins are not dramatic, they are consistent across the dataset, underscoring that subtle but strategic optimization of these audio features may contribute to commercial success in the contemporary music landscape.

3 Research Question 2: How have musical trends evolved over time in relation to major industry shifts?

3.1 Introduction

This study investigates the question: *How have musical trends evolved over time in relation to major industry shifts?* Specifically, it examines the interplay between audio features—such as loudness, danceability, and acousticness—genre popularity, and key technological and structural changes within the music industry. By analyzing these components from 1960 to 2020, the goal is to uncover patterns that illustrate how musical styles have shifted in response to innovations in production, distribution, and consumption. To address this question, two primary elements of the dataset are essential: (1) temporal data on genre prevalence (e.g., hip-hop, rock, pop) and (2) longitudinal measurements of audio features across decades.

This question is particularly compelling because it bridges artistic development with broader sociotechnical dynamics. Technological advancements—such as the introduction of digital audio workstations, the rise of streaming platforms, and the impact of algorithmic recommendation systems—have transformed both the way music is made and the way it reaches audiences. Exploring these dynamics offers insights into how these forces influence the soundscape of each generation. Understanding such patterns not only enriches our comprehension of musical evolution but also provides a framework for anticipating future shifts in musical culture.

3.2 Approach

To analyze the evolution of musical trends in relation to industry shifts, two primary visualizations will be employed.

Genre Popularity Over Time

A **line plot with color mapping** will track genre popularity (rock, hip-hop, pop, country, R&B, and soul) from 1960 to 2020. Line plots are ideal for this purpose because they emphasize temporal continuity, allowing clear visualization of rises, declines, and inflection points in genre dominance. Color mapping distinguishes genres, enabling direct comparisons of their trajectories while maintaining readability. This plot will highlight how cultural and technological shifts — such as the advent of streaming or electronic production — correlate with genre ascendance or decline.

Audio Feature Profiles by Decade

A **radar chart with color mapping** will compare the audio feature profiles (e.g., loudness, danceability, acousticness) across four pivotal decades: the 1960s, 1980s, 2000s, and 2020s. These decades represent key industry transitions: the rise of analog recording (1960s), the emergence of digital tools (1980s–2000s), and the dominance of streaming-era production (2020s). Radar charts are uniquely suited for multivariate comparisons, as they display multiple features on radial axes, making it easy to contrast the "sound" of each era. Color mapping by decade will emphasize shifts in production styles, such as the decline of acousticness and rise of loudness.

Trends in Audio Features Over Time

Linear regression lines and 5-year moving averages will supplement these plots to quantify trends in individual audio features (e.g., tempo, energy) over time. Linear regression lines will reveal the direction and strength of trends (e.g., gradual increases in loudness), while moving averages will smooth short-term fluctuations to highlight long-term patterns. These methods contextualize how specific features evolved in tandem with industry innovations, such as the adoption of DAWs (Digital Audio Workstations) or streaming algorithms.

Structural Evolution via t-SNE Clustering

A **t-SNE cluster map with color mapping** will visualize how musical styles have structurally evolved across decades. t-SNE (t-Distributed Stochastic Neighbor Embedding) is particularly suited for this analysis because it reduces high-dimensional audio feature data into a 2D space while preserving local relationships between data points. Unlike linear methods like PCA, t-SNE emphasizes cluster separation and non-linear patterns, making it ideal for revealing nuanced shifts in genre cohesion or fragmentation.

Color mapping by decade (1960s, 1990s, 2020s) will highlight how clusters transition from balanced groupings (1960s' gradual genre blending) to distinct, elongated clusters (1990s' clear genre boundaries) and finally to fragmented, overlapping clusters (2020s' cross-genre experimentation). This method directly ties structural changes in music — such as the dissolution of traditional genre barriers — to industry shifts like digital production tools and streaming's algorithmic incentivization of hybrid styles.

The t-SNE plot will thus serve as a spatial representation of how technological and cultural forces reshape musical landscapes over time.

3.3 Analysis & Discussion

```python
import dash
from dash import dcc, html, Output, Input, State, callback_context
import plotly.express as px
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
import plotly.graph_objs as go

# Load the dataset
df = pd.read_csv('hottest_50_1960_2022_encoding.csv', encoding='utf-8-sig')

# Convert date column to datetime
df['date'] = pd.to_datetime(df['date'], errors='coerce')
df['year'] = df['date'].dt.year

# Convert genres from string to list
df['genres'] = df['genres'].apply(
    lambda x: [g.strip(" '[] ") for g in x.split(',')] if isinstance(x, str) else []
)

# Genre mapping dictionary
genre_mapping = {
    'country': ['country', 'nashville sound', 'cowboy western', 'arkansas country'],
    'pop': ['pop', 'bubblegum pop', 'dance pop', 'brill building pop', 'sunshine pop'],
    'r&b': ['r&b', 'pop r&b', 'classic soul', 'motown'],
```

```
    'hip hop/rap': ['rap/hip hop', 'hip hop', 'rap', 'trap', 'gangster rap'],
    'rock': ['rock', 'classic rock', 'rock-and-roll', 'folk rock', 'garage rock'],
    'soul': ['soul', 'southern soul', 'memphis soul', 'northern soul']
}

# Create reverse mapping
reverse_mapping = {}
for main_genre, subgenres in genre_mapping.items():
    for subgenre in subgenres:
        reverse_mapping[subgenre] = main_genre

# Map genres
def map_genres(genre_list):
    mapped = []
    for genre in genre_list:
        genre_lower = genre.lower()
        for sub, main in reverse_mapping.items():
            if sub in genre_lower:
                mapped.append(main)
                break
    return list(set(mapped))

df['main_genres'] = df['genres'].apply(map_genres)

# Explode the list of main genres
df_exploded = df.explode('main_genres')

# Filter to our target genres
target_genres = ["country", "pop", "r&b", "hip hop/rap", "rock", "soul"]
df_filtered = df_exploded[df_exploded['main_genres'].isin(target_genres)]

# Create year column
df_filtered['year'] = pd.to_datetime(df_filtered['date']).dt.year

# Create aggregated data
genre_counts = df_filtered.groupby(['year', 'main_genres']).size().reset_index(name='
    count')

# Radar Chart
features = ['loudness', 'danceability', 'energy', 'tempo', 'valence', 'acousticness', '
    liveness', 'speechiness']
df['decade'] = (df['year'] // 10) * 10
df['tempo'] = (df['tempo'] - df['tempo'].min()) / (df['tempo'].max() - df['tempo'].min())
df['loudness'] = (df['loudness'] - df['loudness'].min()) / (df['loudness'].max() - df['
    loudness'].min())
decade_avg = df.groupby('decade')[features].mean().reset_index()

def radar_chart(decade):
    data = decade_avg[decade_avg['decade'] == decade].melt(id_vars=['decade'])
    fig = px.line_polar(data, r='value', theta='variable', line_close=True, title=f'Audio
     Features in {decade}s')
    fig.update_layout(polar=dict(radialaxis=dict(range=[0, 1])))
    fig.update_traces(mode="lines+markers+text", text=data['value'].round(2),
    textposition="top center", textfont_size=8)
    fig.update_layout(polar=dict(radialaxis=dict(range=[0, 1], showticklabels=False)))
    return fig

fig_radar_1960 = radar_chart(1960)
fig_radar_1980 = radar_chart(1980)
fig_radar_2000 = radar_chart(2000)
fig_radar_2020 = radar_chart(2020)

def apply_regression(df, feature):
    df_feature = df.groupby('year')[feature].mean().reset_index()
    df_feature['moving_avg'] = df_feature[feature].rolling(window=5).mean()
    X = df_feature['year'].values.reshape(-1, 1)
    y = df_feature[feature].values.reshape(-1, 1)
    model = LinearRegression().fit(X, y)
    df_feature['trend'] = model.predict(X)
    return df_feature
```

```python

# Dash App
app = dash.Dash(__name__)
app.layout = html.Div([
    html.H1("Music Genre Popularity (1960 2022)", style={'textAlign': 'center'}),

    dcc.Store(id='selected-genres-store', data=target_genres),

    html.Div([
        dcc.RangeSlider(
            id='year-range-slider',
            min=df['year'].min(),
            max=df['year'].max(),
            value=[df['year'].min(), df['year'].max()],
            marks={str(year): str(year) for year in range(df['year'].min(), df['year'].max() + 1, 10)},
            step=1,
            allowCross=False,
            tooltip={"placement": "bottom", "always_visible": True},
        )
    ], style={'margin': '40px 60px'}),

    html.Div([
        dcc.Dropdown(
            id='genre-dropdown',
            options=[{'label': genre.title(), 'value': genre} for genre in target_genres],
            value=target_genres,
            multi=True,
            placeholder="Select genres...",
            style={'width': '100%', 'maxWidth': '1000px', 'margin': '0 auto'}
        )
    ], style={'width': '100%', 'maxWidth': '1000px', 'margin': '20px auto'}),

    dcc.Graph(id='genre-trend'),

    html.Div([
        html.Div([dcc.Graph(id='radar-1960', style={'width': '50%'}),
                  dcc.Graph(id='radar-1980', style={'width': '50%'})], style={'display': 'flex'}),
        html.Div([dcc.Graph(id='radar-2000', style={'width': '50%'}),
                  dcc.Graph(id='radar-2020', style={'width': '50%'})], style={'display': 'flex'})
    ]),

    html.Div([
        dcc.Dropdown(
            id='feature-dropdown',
            options=[{'label': f.title(), 'value': f} for f in features],
            value=features,
            multi=True,
            placeholder="Select features...",
            style={'width': '100%', 'maxWidth': '1000px', 'margin': '0 auto'}
        )
    ], style={'textAlign': 'center', 'marginTop': '20px'}),

    dcc.Graph(id='regression-graph'),
    dcc.Graph(id='moving-avg-graph')
])

@app.callback(
    Output('selected-genres-store', 'data'),
    Input('genre-dropdown', 'value')
)
def update_selected_genres(selected):
    return selected

@app.callback(
    Output('genre-trend', 'figure'),
    [Input('selected-genres-store', 'data'),
```

```python
     Input('year-range-slider', 'value')]
)
def update_genre_trend(selected_genres, year_range):
    start_year, end_year = year_range
    filtered = df_filtered[
        (df_filtered['main_genres'].isin(selected_genres)) &
        (df_filtered['year'] >= start_year) & (df_filtered['year'] <= end_year)
    ]
    genre_counts = filtered.groupby(['year', 'main_genres']).size().reset_index(name='count')
    pivot_df = genre_counts.pivot_table(index='year', columns='main_genres', values='count', aggfunc='sum')\
        .fillna(0).reset_index()

    fig = px.area(
        pivot_df, x='year', y=selected_genres,
        title="Genre Popularity Over Time",
        color_discrete_map={
            'rock': '#1f77b4', 'pop': '#ff7f0e', 'soul': '#2ca02c',
            'r&b': '#d62728', 'hip hop/rap': '#9467bd', 'country': '#8c564b'
        }
    )
    fig.update_layout(
        legend_title="Genres",
        xaxis_title="Year",
        yaxis_title="Number of Songs",
        hovermode="x unified",
        margin=dict(l=60, r=40, t=60, b=40),
        xaxis=dict(range=[pivot_df['year'].min() - 1, pivot_df['year'].max()])
    )
    return fig

@app.callback(
    [Output('radar-1960', 'figure'),
     Output('radar-1980', 'figure'),
     Output('radar-2000', 'figure'),
     Output('radar-2020', 'figure')],
    [Input('selected-genres-store', 'data'),
     Input('year-range-slider', 'value')]
)
def update_radar_charts(selected_genres, year_range):
    start_year, end_year = year_range

    def create_radar(decade):
        filtered = df[
            (df['decade'] == decade) &
            (df['year'] >= start_year) & (df['year'] <= end_year) &
            (df['main_genres'].apply(lambda x: any(g in selected_genres for g in x)))
        ]
        if filtered.empty:
            return go.Figure().add_annotation(text="No data", showarrow=False)

        avg = filtered[features].mean().round(2)
        data = pd.DataFrame({'feature': features, 'value': avg, 'text': avg.round(2)})

        fig = px.line_polar(data, r='value', theta='feature', text='text', line_close=True, title=f'{decade}s Features')
        fig.update_traces(mode="lines+markers+text", textposition="top center", textfont_size=8, line=dict(width=2))
        fig.update_layout(polar=dict(radialaxis=dict(range=[0, 1], showticklabels=False)), showlegend=False)
        return fig

    return create_radar(1960), create_radar(1980), create_radar(2000), create_radar(2020)

@app.callback(
    [Output('regression-graph', 'figure'),
     Output('moving-avg-graph', 'figure')],
    [Input('selected-genres-store', 'data'),
     Input('feature-dropdown', 'value'),
```

```
     Input('year-range-slider', 'value')]
)
def update_trend_graphs(selected_genres, selected_features, year_range):
    start_year, end_year = year_range
    filtered_df = df[
        df['main_genres'].apply(lambda x: any(g in selected_genres for g in x)) &
        (df['year'] >= start_year) & (df['year'] <= end_year)
    ]

    regression_fig = go.Figure()
    moving_avg_fig = go.Figure()

    for i, feature in enumerate(selected_features):
        trend_data = apply_regression(filtered_df, feature)
        color = px.colors.qualitative.Plotly[i % 10]

        regression_fig.add_trace(go.Scatter(x=trend_data['year'], y=trend_data[feature],
                                            mode='markers', name=f'{feature} Data', marker=dict(color=color)))
        regression_fig.add_trace(go.Scatter(x=trend_data['year'], y=trend_data['trend'],
                                            mode='lines', name=f'{feature} Trend', line=dict(color=color)))

        moving_avg_fig.add_trace(go.Scatter(x=trend_data['year'], y=trend_data['moving_avg'],
                                            mode='lines', name=f'{feature} MA', line=dict(color=color)))

    regression_fig.update_layout(title='Linear Regression Trends')
    moving_avg_fig.update_layout(title='5-Year Moving Averages')
    return regression_fig, moving_avg_fig

# Run App
if __name__ == '__main__':
    app.run(debug=True)
```

Listing 5: Dash App for Visualizing Genre Trends

The first part of the code generates a line plot to illustrate how the popularity of music genres has changed over the decades. It begins with a dictionary mapping main genres to their subgenres, which is then reversed to associate subgenres back to main genres for efficient lookup. A `map_genres` function standardizes genre labels accordingly. The dataset is exploded to handle multiple genres per track and filtered to focus on six key genres. After extracting the release year, the data is grouped by year and genre to count occurrences, enabling the creation of a time series plot of genre popularity.

The second part constructs radar charts to compare audio features across decades. It focuses on eight features describing musical characteristics, such as danceability, energy, and valence. Tempo and loudness are normalized to align with the 0–1 range of other features. The code computes average feature values per decade and uses the `radar_chart` function to generate polar plots for selected decades (1960s, 1980s, 2000s, 2020s), revealing shifts in the sonic aesthetics of music over time.

A function `apply_regression` is defined to analyze long-term trends in audio features. It calculates yearly averages, applies a 5-year moving average to smooth short-term fluctuations, and fits a linear regression model to highlight directional changes over time. The result is a DataFrame enriched with both smoothed and linear trend values, suitable for visualizing the evolution of features such as energy or valence.

The final section applies K-Means clustering to group songs by audio characteristics. Selected features are scaled using `StandardScaler` for equal weighting. The Elbow Method is employed to determine the optimal number of clusters by plotting the within-cluster sum of squares (WCSS) for cluster counts from 1 to 10. The inflection point of the curve guides the choice of cluster number, set to 4 in this case. This clustering enables the segmentation of songs into distinct stylistic groups, facilitating further analysis of musical trends and patterns.

Figure 6: Output for Dash App for Visualizing Genre Trends.

```python
# Load and preprocess data
df = pd.read_csv('hottest_50_1960_2022_encoding.csv')
df = df.dropna(subset=['danceability', 'energy', 'loudness', 'speechiness', 'valence', 'tempo'])  # Exclude missing rows
df['year'] = df['date'].str[:4].astype(int)
df['decade'] = (df['year'] // 10) * 10  # Group into decades

# Features for clustering
features = ['danceability', 'energy', 'loudness', 'speechiness', 'valence', 'tempo']
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df[features])

# Determine optimal clusters (Elbow Method)
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, random_state=42)
    kmeans.fit(df_scaled)
    wcss.append(kmeans.inertia_)
optimal_clusters = 4  # Adjust based on elbow curve

# Initialize Dash app
app = dash.Dash(__name__)
app.layout = html.Div([
    html.H1("Dynamic Genre Evolution (1960-2020)", style={'textAlign': 'center'}),

    html.Div([
        dcc.Dropdown(
            id='decade-selector',
            options=[{'label': f'{decade}s', 'value': decade} for decade in sorted(df['decade'].unique())],
            value=df['decade'].min(),
            placeholder="Select Decade",
            style={'width': '100%'}
        )
    ], style={'width': '50%', 'margin': '20px auto'}),

    dcc.Graph(id='cluster-plot'),
    dcc.Graph(id='heatmap'),

    html.H3("Select Feature for Evolution Comparison:", style={'textAlign': 'center'}),

    html.Div([
        dcc.Dropdown(
            id='feature-selector',
            options=[{'label': feat.title(), 'value': feat} for feat in features],
            value='danceability',
            placeholder="Select Feature",
            style={'width': '100%'}
        )
    ], style={'width': '50%', 'margin': '20px auto'}),

    dcc.Graph(id='feature-evolution')
])

# Callbacks for interactivity
@app.callback(
    [Output('cluster-plot', 'figure'),
     Output('heatmap', 'figure'),
     Output('feature-evolution', 'figure')],
    [Input('decade-selector', 'value'),
     Input('feature-selector', 'value')]
)
def update_plots(selected_decade, selected_feature):
    # Filter data by decade
    df_decade = df[df['decade'] == selected_decade].copy()
    scaled_data = scaler.transform(df_decade[features])

    # K-Means Clustering
    kmeans = KMeans(n_clusters=optimal_clusters, random_state=42)
    df_decade['cluster'] = kmeans.fit_predict(scaled_data)
```

```python

    # t-SNE Projection
    tsne = TSNE(n_components=2, random_state=42, perplexity=30, n_iter=500)
    tsne_results = tsne.fit_transform(scaled_data)
    df_decade['tsne_x'] = tsne_results[:, 0]
    df_decade['tsne_y'] = tsne_results[:, 1]

    # Cluster vs Genre Heatmap
    heatmap_data = pd.crosstab(df_decade['cluster'], df_decade['genre_encoding'])
    heatmap_fig = px.imshow(heatmap_data, labels=dict(x="Genre", y="Cluster", color="
    Count"),
                            title=f"Cluster vs Genre (Decade: {selected_decade}s)")

    # Feature Evolution (2000s vs 2020s)
    df_2000s = df[df['decade'] == 2000]
    df_2020s = df[df['decade'] == 2020]
    feature_evolution_fig = go.Figure()

    feature_evolution_fig.add_trace(go.Box(
        x=df_2000s['genre_encoding'],
        y=df_2000s[selected_feature],
        name=f'2000s {selected_feature.title()}'
    ))

    feature_evolution_fig.add_trace(go.Box(
        x=df_2020s['genre_encoding'],
        y=df_2020s[selected_feature],
        name=f'2020s {selected_feature.title()}'
    ))

    feature_evolution_fig.update_layout(
        title=f"Feature Evolution: {selected_feature.title()} (2000s vs 2020s)",
        xaxis_title="Genre Encoding",
        yaxis_title=selected_feature.title()
    )

    # Cluster Plot
    cluster_fig = px.scatter(
        df_decade, x='tsne_x', y='tsne_y', color='cluster',
        hover_data=['title', 'artist', 'genre_encoding'],
        title=f"Clusters in the {selected_decade}s (t-SNE Projection)"
    )

    return cluster_fig, heatmap_fig, feature_evolution_fig

# Run the app in Colab
app.run(mode='inline', port=8050)
```

Listing 6: Dynamic Genre Evolution Dashboard (1960–2022)

Figure 7: Output for Dynamic Genre Evolution Dashboard (1960–2022).

The first plot presents a longitudinal analysis of genre popularity from 1960 to 2022, highlighting significant shifts in musical preferences over time. In the earlier decades, genres such as rock, soul, and R&B dominated the popular music landscape. However, a marked rise in the prominence of pop and hip-hop is observed from the early 2000s onwards. This transition can be interpreted as a reflection of broader cultural and technological developments, including the mainstream acceptance of rap music and the global dissemination of pop through digital platforms. The advent of streaming services and the proliferation of social media have likely accelerated the dominance of commercially adaptive genres such as hip-hop and pop, which are characterized by their responsiveness to evolving audience tastes. Furthermore, these genre trends often align with prevailing socio-political contexts—for instance, the prevalence of protest-themed soul in the 1960s and 1970s, contrasted with the escapist and highly produced pop music of the 2010s.

The second and third plots focus on temporal changes in musical attributes, particularly valence (a measure of musical positivity) and tempo. Both features exhibit a general downward trend, particularly in the post-2010 era, suggesting a stylistic shift towards darker, slower, and more introspective compositions. This evolution is exemplified by the progression from the optimistic tone of *Hey Jude* (1968), to the grunge-driven rawness of *Smells Like Teen Spirit* (1991), and most recently, to the emotive yet polished retro stylings of *Blinding Lights* (2019). The segmented regression model in the third plot identifies a structural breakpoint around 2015, coinciding with the growing influence of algorithmic recommendation systems and streaming-era consumption patterns. These changes have had a tangible impact on songwriting and production practices, encouraging features such as shorter introductions, earlier vocal entries, and emotionally engaging hooks designed to quickly capture listener attention.

The final plot employs unsupervised clustering of audio features to reveal the increasing fragmentation and hybridity of contemporary popular music. Rather than adhering to traditional genre boundaries, many modern tracks occupy a liminal space between styles, indicative of a broader trend toward genre fusion and cross-cultural collaboration. This phenomenon is exemplified by tracks such as *Old Town Road* by Lil Nas X, which merges elements of country and trap, and *Despacito* by Luis Fonsi and Daddy Yankee, which blends Latin pop with reggaeton and mainstream pop influences. Such hybridization expands the global appeal of these songs and underscores the tendency of modern artists and producers to transcend conventional genre frameworks in favor of more flexible, platform-native soundscapes.

4 Case Study: Taylor Swift

We conducted a focused case study on Taylor Swift, one of the most influential and commercially successful artists of the past two decades. Given her extensive discography and presence across multiple genres and eras, Taylor Swift offers a rich and diverse dataset for statistical exploration.

In this case study, we developed an interactive dashboard that visualizes the key audio and chart performance metrics associated with her songs. The dashboard enables dynamic filtering by songs, release year, and chart tier, providing an intuitive interface to explore trends in her musical evolution and rank over time.

This artist-centric dashboard not only enhances interpretability through visual analytics but also serves as a case example of how data-driven insights can be used to profile an individual artist's musical trajectory. Through this focused lens, we aim to complement the broader statistical findings with a personalized, interactive exploration of one of the most iconic figures in modern pop music.

```python
import dash
from dash import dcc, html, Input, Output, dash_table
import plotly.graph_objects as go
import pandas as pd

# Load dataset
df = pd.read_csv("/content/hot100_all_1960_2022.csv")
df['date'] = pd.to_datetime(df['date'])

# Initialize app
app = dash.Dash(__name__)
artists = sorted(df["artist"].dropna().unique())

# App Layout
app.layout = html.Div(style={'backgroundColor': 'white', 'color': 'black', 'padding': '20px'}, children=[
    html.H1("Case Study: Music Rankings", style={'color': '#1f77b4', 'textAlign': 'center'}),

    html.Div([
        html.Label("Select Artist:", style={'color': 'black'}),
        dcc.Dropdown(
            id='artist-dropdown',
            options=[{'label': artist, 'value': artist} for artist in artists],
            placeholder="Select an artist",
            style={'backgroundColor': 'white', 'color': 'black'}
        )
    ], style={'width': '50%', 'margin': 'auto'}),

    html.Div([
        html.Div([
            html.H3("Top 6 Ranking Songs", style={'color': 'black'}),
            dash_table.DataTable(
                id='top-songs-table',
                columns=[
                    {'name': 'Title', 'id': 'title'},
                    {'name': 'Peak Rank', 'id': 'peak'}
                ],
                style_table={'width': '100%', 'border': 'none', 'backgroundColor': 'white'},
                style_header={'backgroundColor': '#1f77b4', 'color': 'white'},
                style_data={'backgroundColor': 'white', 'color': 'black'}
```

```python
            ),
        ], style={'width': '40%', 'display': 'inline-block', 'verticalAlign': 'top'}),

        html.Div([
            html.H3("Best Rank:", style={'color': 'black'}),
            html.H2(id='best-rank', style={'color': '#1f77b4'}),
            html.H3("Songs on Board:", style={'color': 'black'}),
            html.H2(id='songs-on-board', style={'color': '#1f77b4'}),
            html.H3("Longest Lasting Week:", style={'color': 'black'}),
            html.H2(id='longest-week', style={'color': '#1f77b4'}),
        ], style={'width': '40%', 'display': 'inline-block', 'verticalAlign': 'top', '
    paddingLeft': '5%'}),
    ], style={'display': 'flex', 'justifyContent': 'center'}),

    html.Div([
        html.H3("Rank History", style={'color': 'black', 'textAlign': 'center'}),
        dcc.Graph(id='rank-history-graph')
    ], style={'width': '80%', 'margin': 'auto'}),

    html.Div([
        html.H3("Top Lasting Songs on Billboard", style={'color': 'black'}),
        dcc.Graph(id='lasting-songs-bar')
    ], style={'width': '80%', 'margin': 'auto'}),
])

# Callback to update dashboard
@app.callback(
    [Output('top-songs-table', 'data'),
     Output('best-rank', 'children'),
     Output('songs-on-board', 'children'),
     Output('longest-week', 'children'),
     Output('rank-history-graph', 'figure'),
     Output('lasting-songs-bar', 'figure')],
    [Input('artist-dropdown', 'value')]
)
def update_dashboard(selected_artist):
    if not selected_artist:
        return [], "N/A", "N/A", "N/A", go.Figure(), go.Figure()

    filtered_df = df[df['artist'].str.contains(selected_artist, case=False, na=False)]

    if filtered_df.empty:
        return [], "N/A", "N/A", "N/A", go.Figure(), go.Figure()

    filtered_df_unique = filtered_df.drop_duplicates(subset=['title'])
    top_songs_df = filtered_df_unique.sort_values('peak').head(6)
    top_songs = top_songs_df[['title', 'peak']].to_dict('records')

    # Key Metrics
    best_rank = filtered_df['peak'].min()
    songs_on_board = filtered_df['title'].nunique()
    longest_week = filtered_df['weeks'].max()

    # Colors and line styles for top 6 songs
    colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b']  # Plotly
     default color cycle
    line_styles = ['solid', 'dash', 'dot', 'dashdot', 'longdash', 'longdashdot']

    # Rank History Chart (only top 6 songs)
    rank_fig = go.Figure()
    for idx, row in top_songs_df.reset_index(drop=True).iterrows():
        song = row['title']
        song_data = filtered_df[filtered_df['title'] == song].sort_values('date')

        rank_fig.add_trace(go.Scatter(
            x=song_data['date'],
            y=song_data['peak'],
            mode='lines+markers',
            name=song,
            line=dict(color=colors[idx], dash=line_styles[idx])
```

```
        ))

    rank_fig.update_layout(
        title="Rank History",
        xaxis_title="Date",
        yaxis_title="Rank",
        yaxis=dict(
            autorange="reversed",
            showline=True,
            linewidth=1,
            linecolor='black',
            gridcolor='lightgray'
        ),
        xaxis=dict(
            showline=True,
            linewidth=1,
            linecolor='black',
            gridcolor='lightgray'
        ),
        paper_bgcolor='white',
        plot_bgcolor='white',
        font_color='black'
    )

    # Lasting Songs Chart
    lasting_songs = filtered_df.groupby('title')['weeks'].max().reset_index()
    lasting_songs = lasting_songs.sort_values('weeks', ascending=False).head(9)

    bar_fig = go.Figure(go.Bar(
        y=lasting_songs['title'],
        x=lasting_songs['weeks'],
        orientation='h',
        marker=dict(color='#1f77b4')
    ))
    bar_fig.update_layout(
        title="Top Lasting Songs on Billboard",
        xaxis_title="Weeks",
        yaxis_title="Title",
        xaxis=dict(
            showline=True,
            linewidth=1,
            linecolor='black',
            gridcolor='lightgray'
        ),
        yaxis=dict(
            showline=True,
            linewidth=1,
            linecolor='black',
            gridcolor='lightgray'
        ),
        paper_bgcolor='white',
        plot_bgcolor='white',
        font_color='black'
    )

    return top_songs, best_rank, songs_on_board, longest_week, rank_fig, bar_fig

# Run app
if __name__ == '__main__':
    app.run(debug=True)
```

Listing 7: Artist Interactive Dashboard

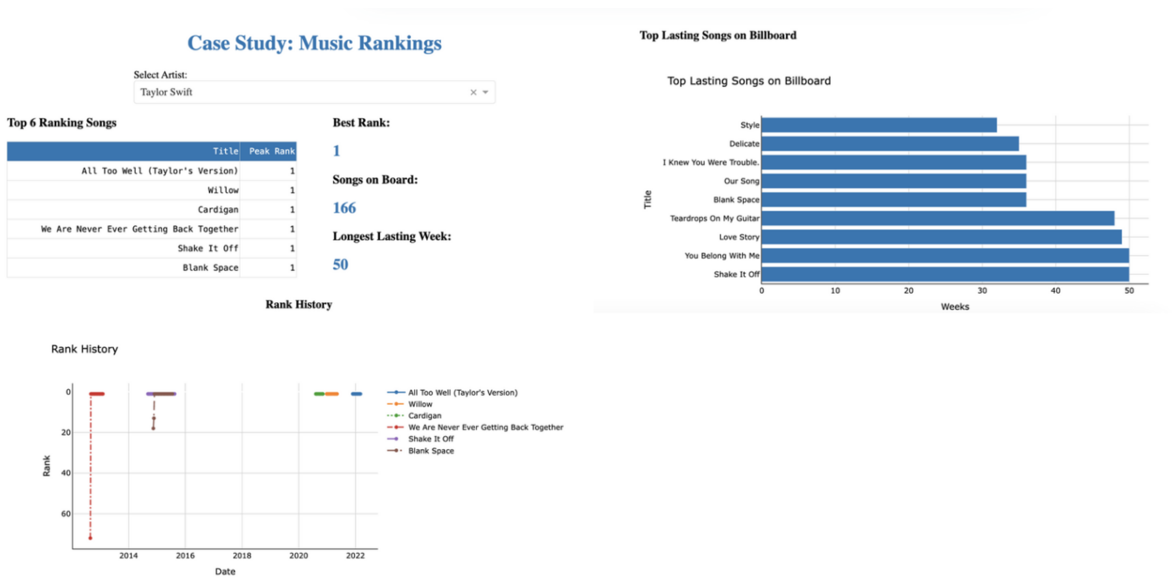

Figure 8: Output for Artist Interactive Dashboard

```python
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from matplotlib.patches import Patch

# Filter Taylor Swift songs
ts_df = df_billboard[df_billboard['artist'].str.contains('Taylor Swift', case=False)]

# Classify songs (example - adjust based on your peak/weeks data)
ts_df['success_category'] = np.where(
    (ts_df['peak'] <= 10) & (ts_df['weeks'] >= 10),
    'Top Hits',
    np.where(
        (ts_df['peak'] <= 50) & (ts_df['weeks'] >= 5),
        'Moderate Success',
        'Lower Performance'
    )
)

# Select key features for analysis
features = ['danceability', 'energy', 'valence', 'acousticness', 'speechiness']

# 1. Radar Chart Comparison
def create_radar_chart():
    categories = features
    N = len(categories)
    angles = np.linspace(0, 2*np.pi, N, endpoint=False).tolist()
    angles += angles[:1]

    fig = plt.figure(figsize=(10, 10), facecolor='white')
    ax = fig.add_subplot(111, polar=True)

    # Normalize values
    for cat in ['Top Hits', 'Moderate Success', 'Lower Performance']:
        subset = ts_df[ts_df['success_category'] == cat]
        values = [subset[feat].mean() for feat in features]
        values += values[:1]
        ax.plot(angles, values, linewidth=2, label=cat)
        ax.fill(angles, values, alpha=0.1)

    ax.set_theta_offset(np.pi/2)
    ax.set_theta_direction(-1)
    plt.xticks(angles[:-1], [f.upper() for f in features])
```

```python
    plt.title('Taylor Swift Audio Signature\nBy Chart Performance', pad=40, fontsize=14)
    plt.legend(bbox_to_anchor=(1.3, 1.1))
    plt.show()

# 2. Feature Distribution Comparison
plt.figure(figsize=(14, 8))
for i, feature in enumerate(features, 1):
    plt.subplot(2, 3, i)
    sns.boxplot(
        data=ts_df,
        x='success_category',
        y=feature,
        order=['Top Hits', 'Moderate Success', 'Lower Performance'],
        palette=['#1f77b4', '#ff7f0e', '#2ca02c'],
        showfliers=False
    )
    plt.title(feature.capitalize())
    plt.xticks(rotation=45)
plt.suptitle('Taylor Swift Song Features by Performance Tier', y=1.02, fontsize=14)
plt.tight_layout()
plt.show()

# 3. Success Timeline Analysis
plt.figure(figsize=(14, 6))
ts_success = ts_df.groupby(['year', 'success_category']).size().unstack()
ts_success.plot(
    kind='area',
    stacked=True,
    color=['#1f77b4', '#ff7f0e', '#2ca02c'],
    alpha=0.7
)
plt.title('Taylor Swift Chart Performance Over Time', pad=20)
plt.ylabel('Number of Songs')
plt.xlabel('Year')
plt.legend(title='Performance Tier')
plt.grid(axis='y', alpha=0.3)
plt.show()

# 4. Feature Correlation with Success
plt.figure(figsize=(10, 6))
corr_data = ts_df[features + ['peak', 'weeks']].corr()
sns.heatmap(
    corr_data[['peak', 'weeks']].drop(['peak', 'weeks']),
    annot=True,
    cmap='coolwarm',
    center=0,
    vmin=-1,
    vmax=1
)
plt.title('Audio Features Correlation with Chart Performance', pad=20)
plt.show()
```

Listing 8: Taylor Swift Performance Tier Charts

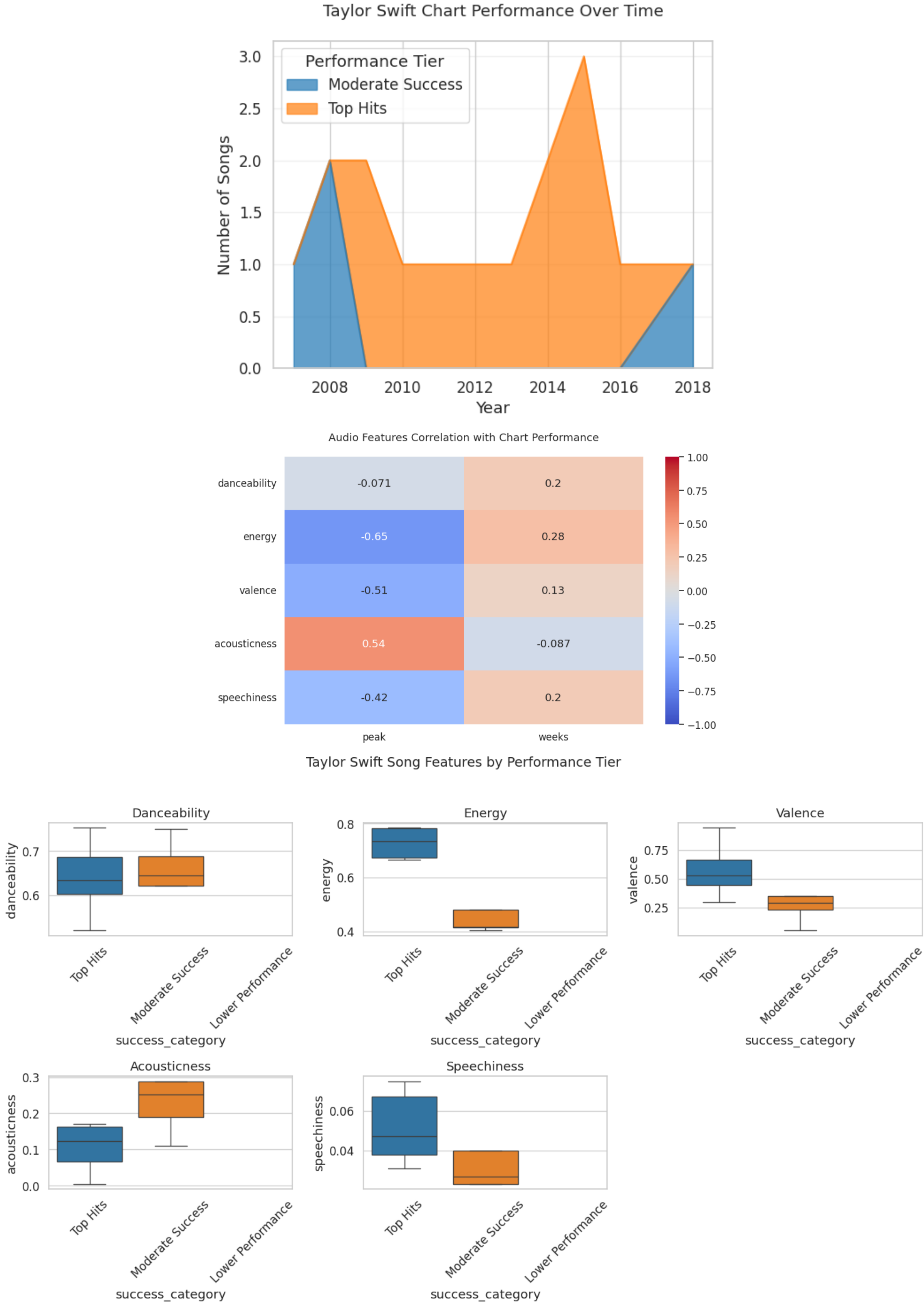

Figure 9: Output for Taylor Swift Performance Tier Charts

Since her debut in 2006, Taylor Swift's career has undergone remarkable transformations—evolving from country sweetheart to pop megastar to indie-folk storyteller. By analyzing data across her discography, we can uncover fascinating patterns and trends that reveal her artistic growth, thematic shifts, and musical experimentation.

From her self-titled country debut in 2006 to the genre-defying *Midnights* (2022) and beyond, Taylor Swift's musical evolution reflects not only her personal growth but also broader shifts in the music industry. This exploration tracks her journey using data visualization to highlight key turning points, recurring motifs, and the strategic choices that have shaped her enduring success.

To quantify the success of Taylor Swift's songs, we categorize them into two tiers based on their Billboard Hot 100 performance:

- **Tier 1:** Top 10 peak, 10+ weeks charting
- **Tier 2:** Top 50 peak, 5+ weeks charting

Visualizing this data over time reveals fascinating trends:

- Her early country hits (e.g., *Love Story*) had longevity and emotional resonance.
- The *1989* pop transition era brought explosive chart dominance with multiple Tier 1 entries.
- Her indie-folk work (*Folklore*, *Evermore*) received critical acclaim but sometimes shorter chart stays, reflecting changing listener habits.

Peak success (2009–2017) is highlighted as a period marked by chart-topping hits and industry accolades.

Taylor's early music is rich in acoustic textures—organic guitars, minimal processing, and a country warmth. As she matured, these gave way to polished electronic production, synthesizers, and programmed beats that define her pop sound.

A clear trend in **danceability** shows a shift from mid-tempo country grooves to rhythm-driven pop anthems. This mirrors broader movements in mainstream music while preserving her lyrical identity.

The emotional tone (valence) of her music has evolved:

- Early work: balanced ballads and youthful exuberance.
- Pop era: more euphoric, brighter sounds.

This shift signals both personal development and strategic adaptation to pop culture.

Recent albums show higher **speechiness**—talk-singing styles that make her lyrics feel conversational and intimate. This enhances her confessional storytelling and rhythmic vocal experimentation.

The sonic impact of Taylor's music has grown louder and more compressed, in line with industry norms. This amplifies her music's immediacy on radio and streaming platforms, while earlier works maintain more dynamic range.

This analysis reveals how Taylor Swift has consciously shaped her sound while maintaining the emotional authenticity that defines her artistry. Each musical characteristic serves both creative expression and audience connection, proving that technical evolution and artistic integrity can walk hand in hand.

5 Conclusion

This study delved into six decades of Billboard Hot 100 data to unravel the intricate relationship between popular music, cultural evolution, and technological innovation. By analyzing audio features, genre trajectories, structural trends, and artist-specific case studies, the findings illuminate how music serves as both a reflection of societal values and a product of industry-driven strategies.

The sonic landscape of chart-topping music has shifted markedly over time. Features like danceability, energy, and valence—quantifying rhythm, intensity, and emotional positivity—rose consistently, reflecting listener preferences for upbeat, shareable tracks in the streaming era. Songs that achieved top-tier success (peaking in the top 10 and charting for 10+ weeks) leaned heavily into these traits, with higher averages in danceability (0.65) and valence (0.57) compared to lower-tier tracks. This underscores the commercial imperative of crafting music that resonates emotionally while aligning with platform algorithms optimized for engagement.

Genre popularity mirrored technological milestones. The ascent of hip-hop/rap post-2000s coincided with the democratization of digital tools like DAWs and the genre's compatibility with streaming's emphasis on lyrical density and repetitive hooks. Conversely, rock's decline highlighted the challenges analog-rooted genres face in a digitized market. Radar charts further revealed seismic shifts in production aesthetics: the warm, acoustic textures of the 1960s gave way to the loud, electronic-driven soundscapes of the 2020s, shaped by innovations in mastering tools and streaming's loudness normalization practices.

Structurally, music has grown increasingly fluid. t-SNE clustering illustrated the dissolution of rigid genre boundaries, with 2020s tracks forming fragmented, hybridized clusters. This reflects algorithmic curation's role in promoting cross-genre experimentation and playlists, fostering a soundscape where artists blend styles like pop-rap or folk-tronica to maximize reach.

Taylor Swift's career exemplifies adaptive artistry in this evolving landscape. Her transition from country to pop (1989) embraced high-energy, danceable production, while her indie-folk work (Folklore) prioritized lyrical intimacy, aligning with streaming's preference for conversational storytelling. This duality highlights how artists balance creative reinvention with data-informed strategies to sustain relevance.

While this analysis offers valuable insights, its scope has limitations. The Billboard dataset may underrepresent niche genres, and Spotify's genre taxonomy risks oversimplifying musical identities. Future research could explore causal links between streaming metrics (e.g., skip rates) and audio features, or expand to global charts to assess cultural universality.

Integrating lyrical data could further enrich this work. Lyrics offer untapped insights into cultural narratives—how societal events shape lyrical themes, or whether optimistic valence scores align with hopeful lyrics. Natural Language Processing (NLP) could decode shifts in sentiment or topical focus, such as the rise of introspective themes in pandemic-era music. Challenges like copyright restrictions and interpreting metaphorical language exist, but partnerships with platforms like Genius or crowd-sourced databases could mitigate these barriers.Additional directions include integrating cultural context and user behavior metrics, mapping artist collaboration networks, and comparing Billboard data with streaming platform trends. These extensions would deepen our understanding of the dynamics that drive musical popularity and evolution in the modern music industry.

Ultimately, this study underscores music's dual identity: a cultural mirror reflecting societal currents and a commercial commodity shaped by technological possibilities. As AI tools and streaming algorithms redefine production and consumption, understanding these dynamics—and embracing lyrical narratives—will be crucial for artists, producers, and scholars navigating the future of music. The interplay of art and analytics continues to redefine what resonates, ensuring that the soundtracks of our lives remain as dynamic as the world they soundtrack.