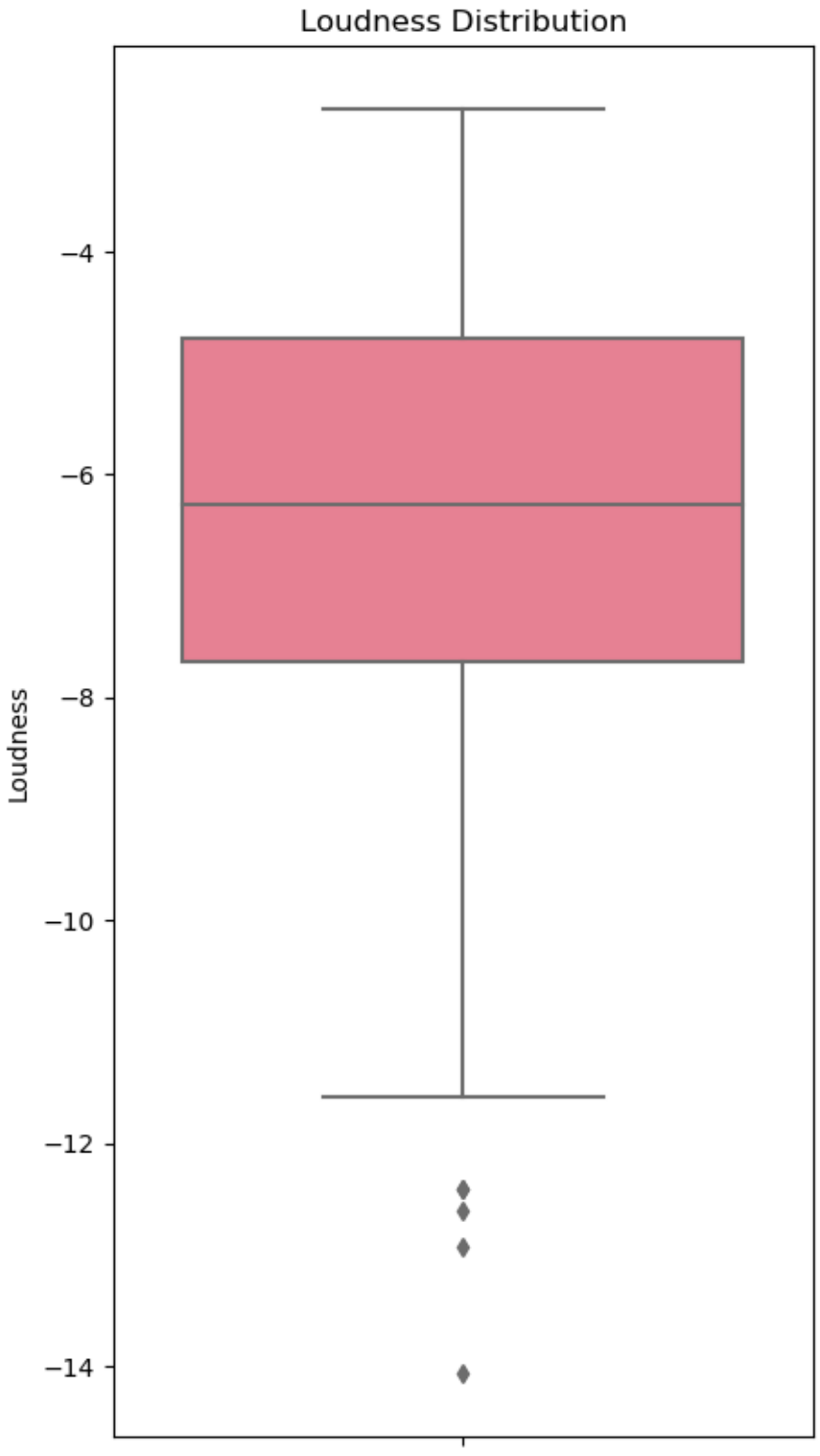
Python EL

March 13, 2023

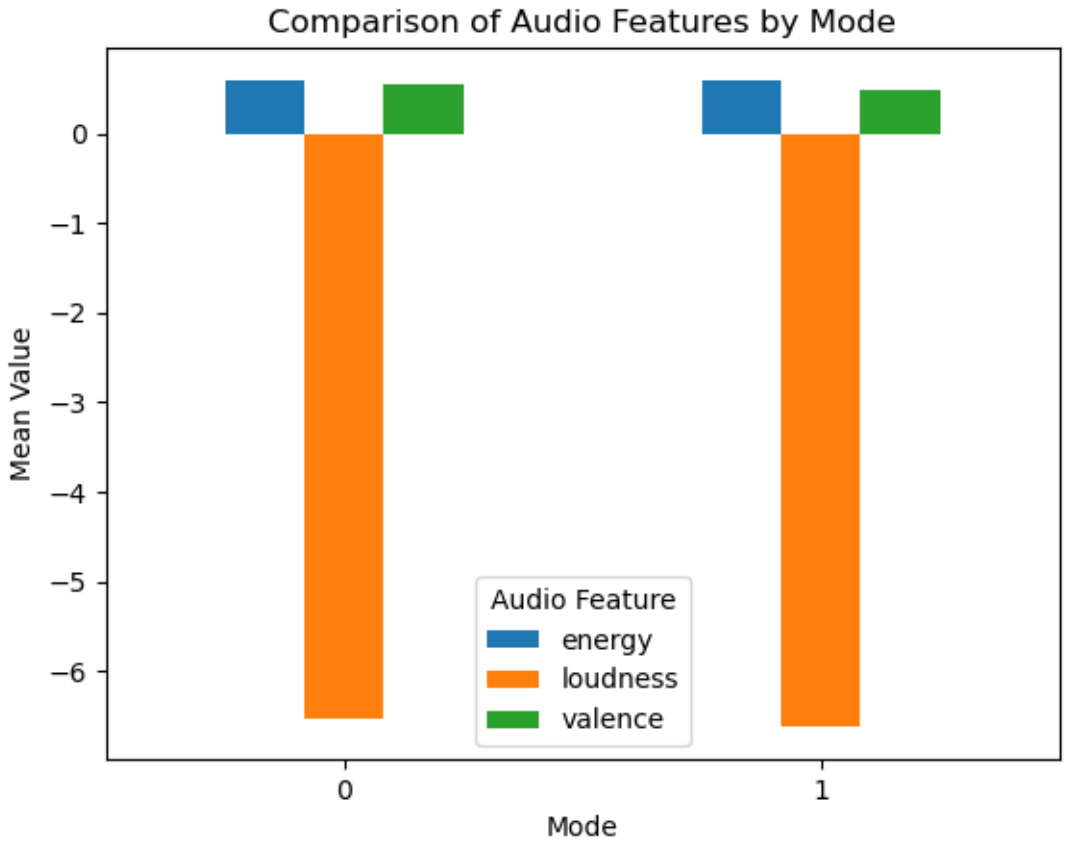
[18]:
```
from bs4 import BeautifulSoup
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from IPython.display import IFrame


# Load the CSV file into a Pandas DataFrame
df = pd.read_csv('top 100 streamed songs 2 - top 100 streamed songs 2.csv')
```

[19]:
```
# Create a new HTML template using BeautifulSoup
soup = BeautifulSoup('<html><head><title>Spotify Data</title></head><body></
 ↪body></html>', 'html.parser')
```

[60]:
```
# Plot a box plot of loudness
plt.figure(figsize=(5, 10))
sns.boxplot(y='loudness', data=df)
plt.title("Loudness Distribution")
plt.ylabel("Loudness")

plt.savefig('loudness box plot.png', bbox_inches='tight')
graph1_div = soup.new_tag("div", id="graph0")
plt.show()
img_tag1 = soup.new_tag('img', src='loudness box plot.png')
graph1_div.append(img_tag1)
soup.body.append(graph1_div)
```

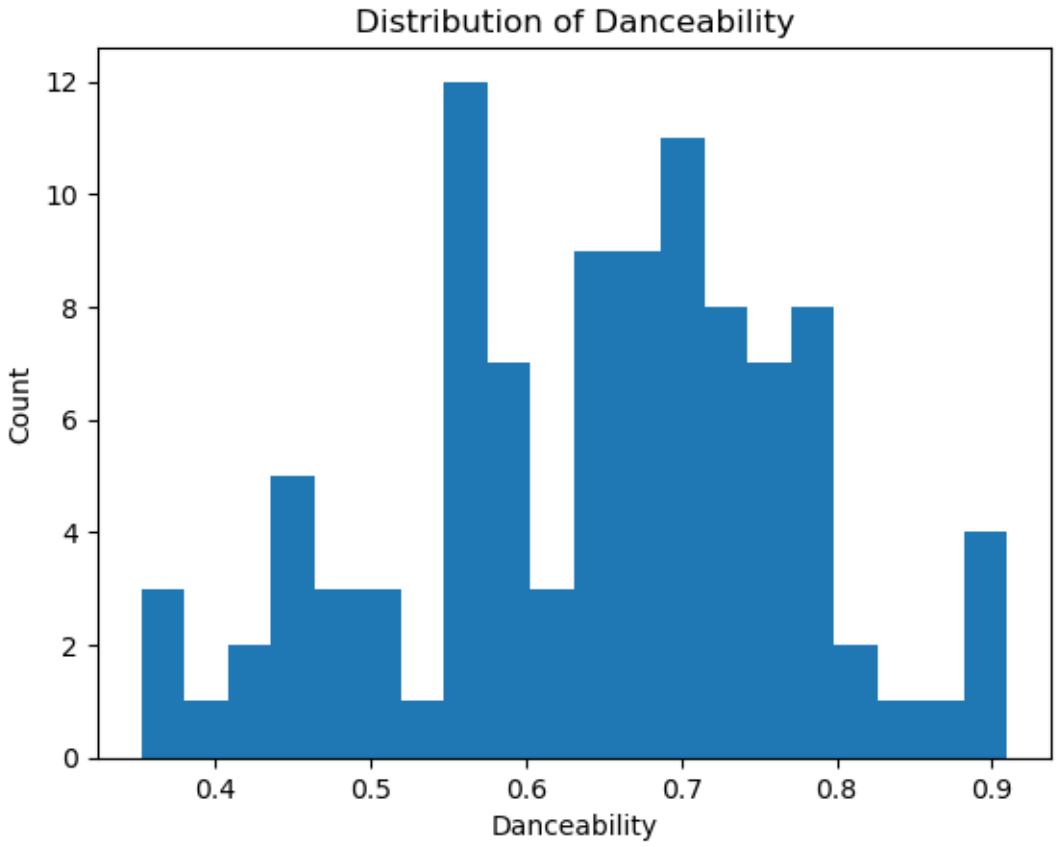

Loudness Distribution
Loudness
−4
−6
−8
−10
−12
−14

[41]:
```
# Group the data by mode and calculate the mean values for energy, loudness,␣
↪and valence
grouped_data = df.groupby('mode')[['energy', 'loudness', 'valence']].mean()

# Create a column chart to compare the mean values of different audio features
grouped_data.plot(kind='bar', rot=0)
plt.xlabel('Mode')
plt.ylabel('Mean Value')
plt.title('Comparison of Audio Features by Mode')
plt.legend(title='Audio Feature')
plt.savefig('energy_valence_loudness.png', bbox_inches='tight')
plt.show()
graph1_div = soup.new_tag("div", id="graph2")
img_tag1 = soup.new_tag('img', src='energy_valence_loudness.png')
graph1_div.append(img_tag1)
soup.body.append(graph1_div)
```



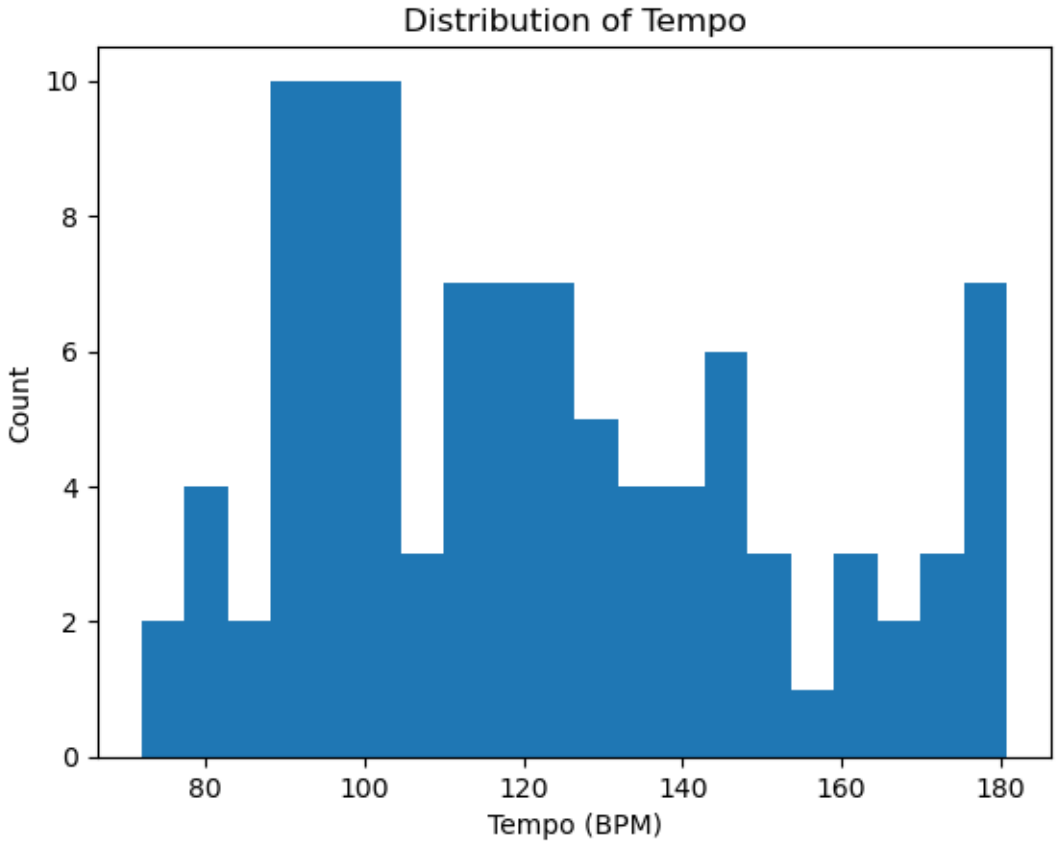
[42]:
```
# Plot the distribution of danceability using a histogram
plt.hist(df['danceability'], bins=20)
```

```
plt.xlabel('Danceability')
plt.ylabel('Count')
plt.title('Distribution of Danceability')
plt.savefig('danceability histogram.png', bbox_inches='tight')
plt.show()
graph1_div = soup.new_tag("div", id="graph3")
img_tag1 = soup.new_tag('img', src='danceability histogram.png')
graph1_div.append(img_tag1)
soup.body.append(graph1_div)
```

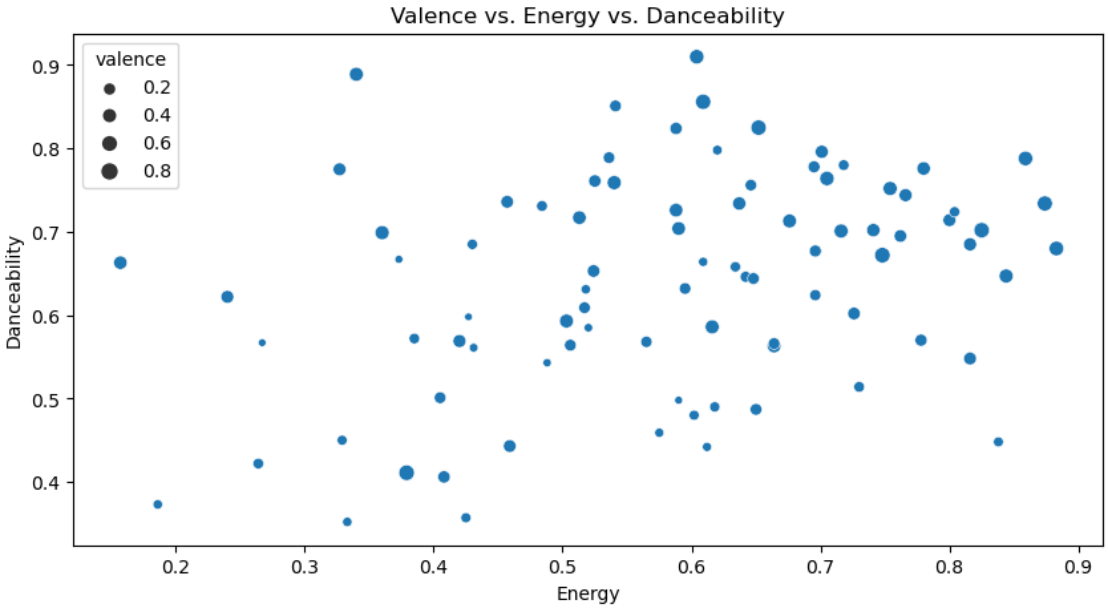


[43]:
```
# Plot the distribution of tempo using a histogram
plt.hist(df['tempo'], bins=20)
plt.xlabel('Tempo (BPM)')
plt.ylabel('Count')
plt.title('Distribution of Tempo')

plt.savefig('tempo histogram.png', bbox_inches='tight')
plt.show()
graph1_div = soup.new_tag("div", id="graph4")
img_tag1 = soup.new_tag('img', src='tempo histogram.png')
```

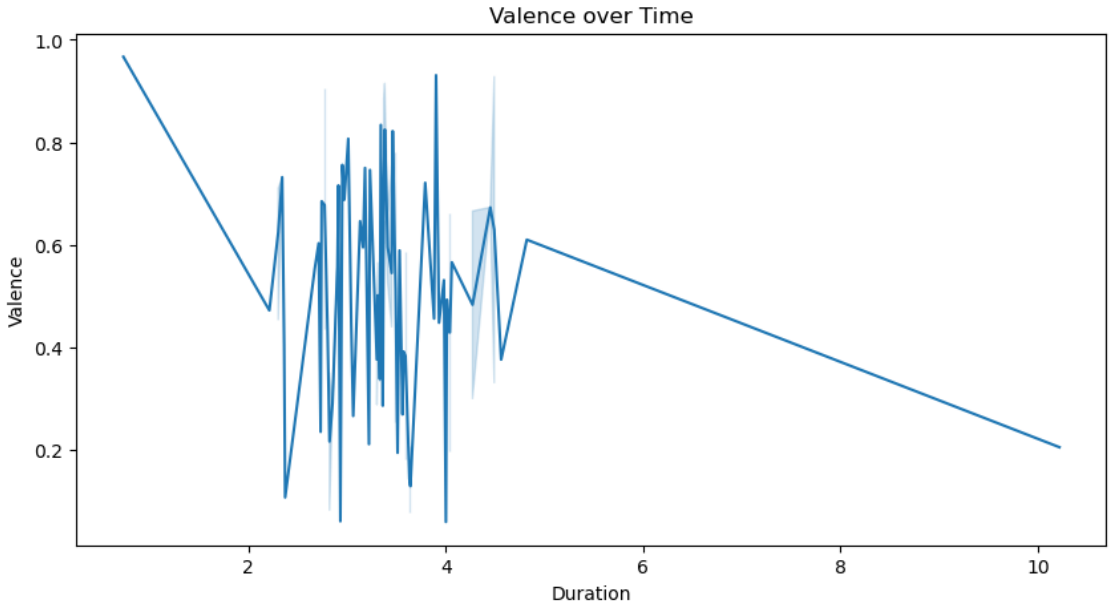
```
graph1_div.append(img_tag1)
soup.body.append(graph1_div)
```



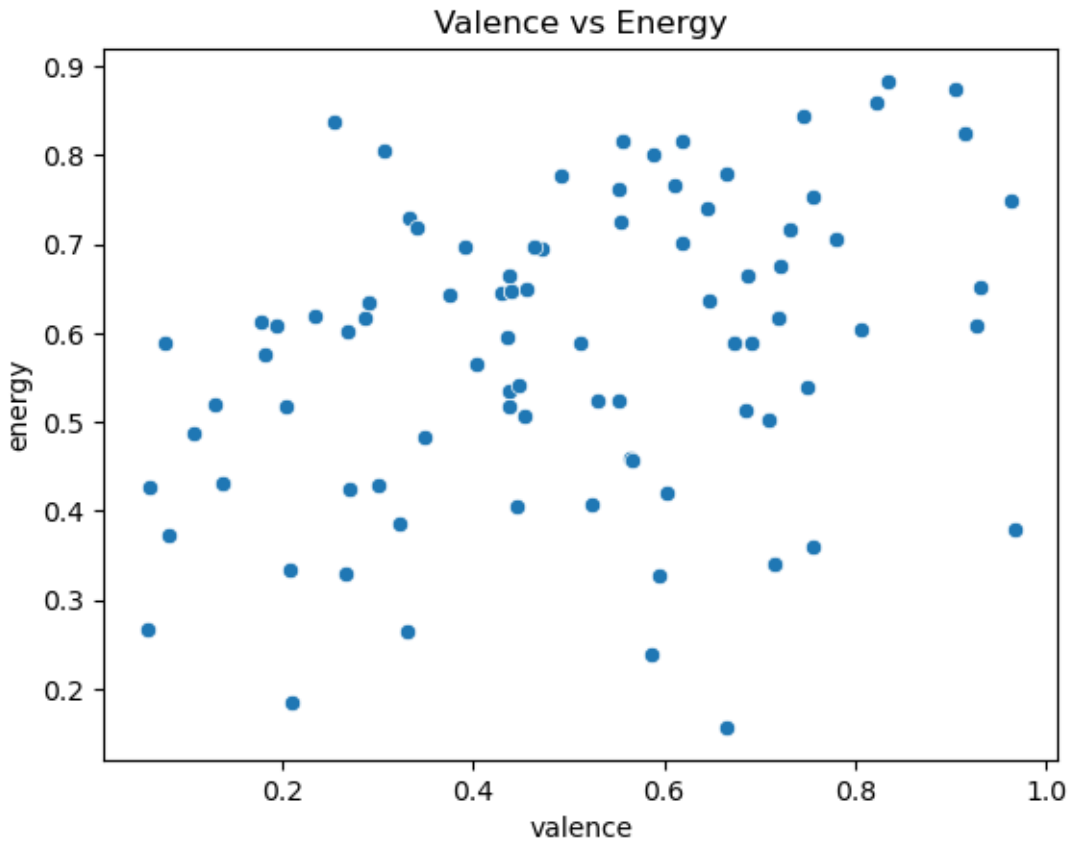
[44]:
```
# Plot a bubble chart of valence vs. energy vs. danceability
plt.figure(figsize=(10, 5))
sns.scatterplot(x='energy', y='danceability', size='valence', data=df)
plt.title("Valence vs. Energy vs. Danceability")
plt.xlabel("Energy")
plt.ylabel("Danceability")

plt.savefig('bubble_valence_energy_danceability.png', bbox_inches='tight')
plt.show()
graph1_div = soup.new_tag("div", id="graph5")
img_tag1 = soup.new_tag('img', src='bubble_valence_energy_danceability.png')
graph1_div.append(img_tag1)
soup.body.append(graph1_div)
```



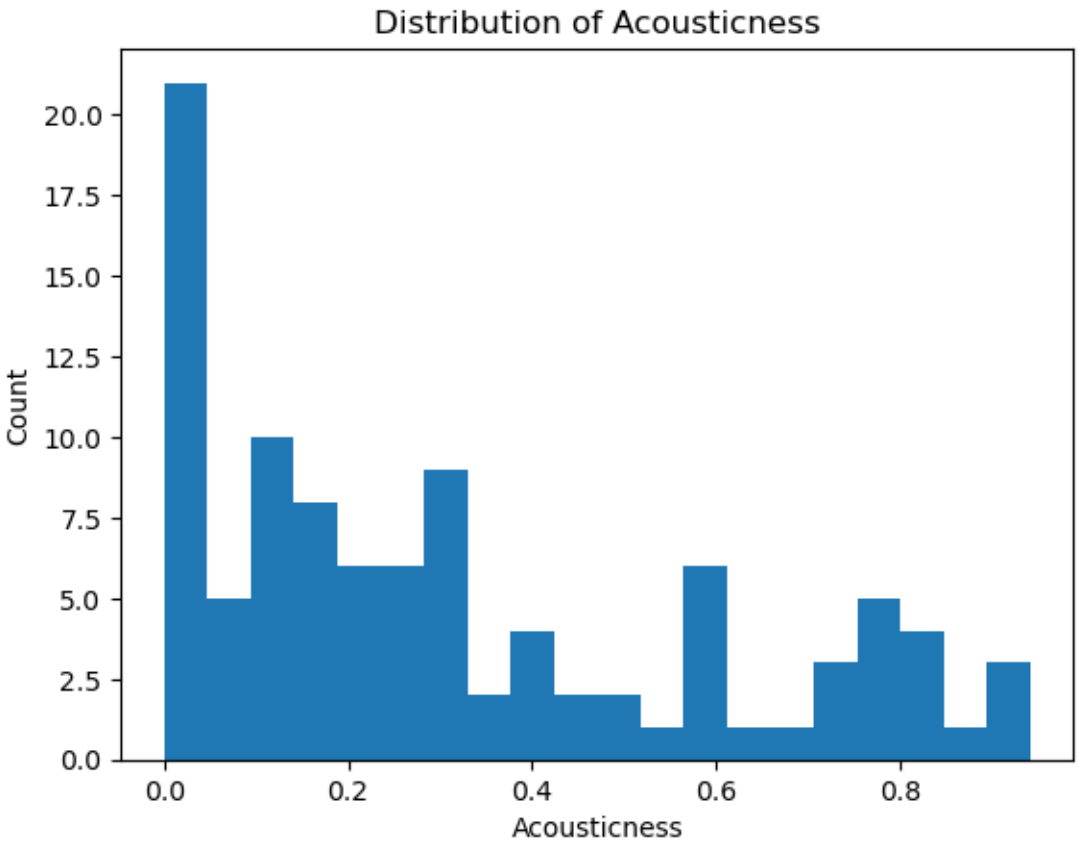
[45]:
```
# Plot a line chart of valence over time
plt.figure(figsize=(10, 5))
sns.lineplot(x='duration', y='valence', data=df)
plt.title("Valence over Time")
plt.xlabel("Duration")
plt.ylabel("Valence")
plt.savefig('line chart_valence_time.png', bbox_inches='tight')
plt.show()
graph1_div = soup.new_tag("div", id="graph6")
img_tag1 = soup.new_tag('img', src='line chart_valence_time.png')
graph1_div.append(img_tag1)
soup.body.append(graph1_div)
```



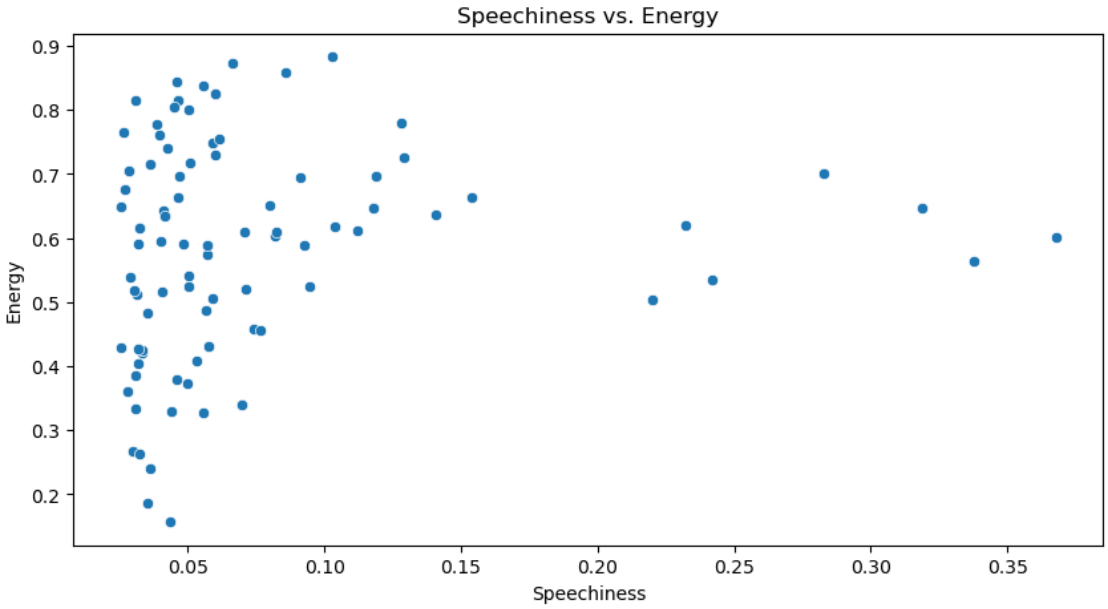
[46]:
```
# Plot the correlation between valence and energy using a scatter plot

sns.scatterplot(data=df, x='valence', y='energy')
plt.title('Valence vs Energy')

plt.savefig('scatter_valence_energy.png', bbox_inches='tight')
plt.show()
graph1_div = soup.new_tag("div", id="graph7")
img_tag1 = soup.new_tag('img', src='scatter_valence_energy.png')
graph1_div.append(img_tag1)
soup.body.append(graph1_div)
```



[47]:
```
# Plot the distribution of acousticness using a histogram
plt.hist(df['acousticness'], bins=20)
plt.xlabel('Acousticness')
plt.ylabel('Count')
plt.title('Distribution of Acousticness')

plt.savefig('Histogram_acousticness.png', bbox_inches='tight')
plt.show()
graph1_div = soup.new_tag("div", id="graph8")
img_tag1 = soup.new_tag('img', src='Histogram_acousticness.png')
graph1_div.append(img_tag1)
soup.body.append(graph1_div)
```

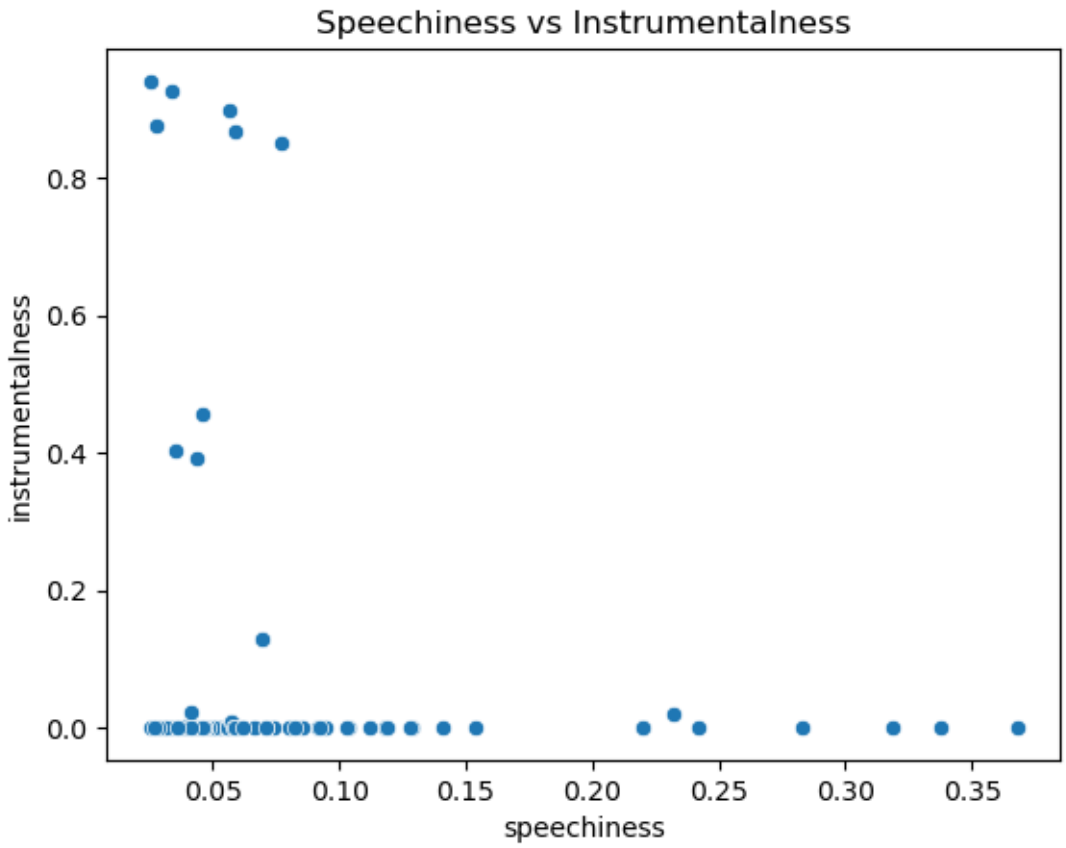


[49]:
```
# Plot a scatter plot of speechiness vs. energy
plt.figure(figsize=(10, 5))
sns.scatterplot(x='speechiness', y='energy', data=df)
plt.title("Speechiness vs. Energy")
plt.xlabel("Speechiness")
plt.ylabel("Energy")
plt.show()
plt.savefig('scatter_speechiness_energy.png', bbox_inches='tight')
graph1_div = soup.new_tag("div", id="graph9")
img_tag1 = soup.new_tag('img', src='scatter_speechiness_energy.png')
graph1_div.append(img_tag1)
soup.body.append(graph1_div)
```

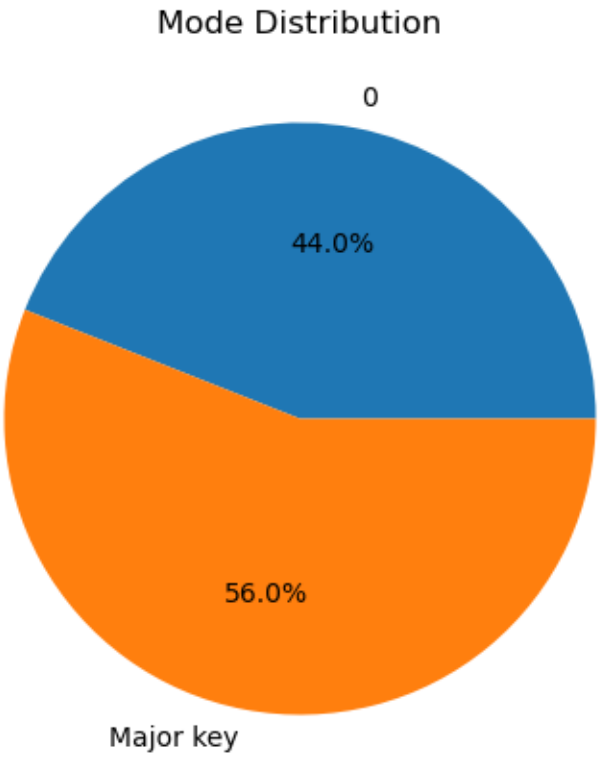


<Figure size 640x480 with 0 Axes>

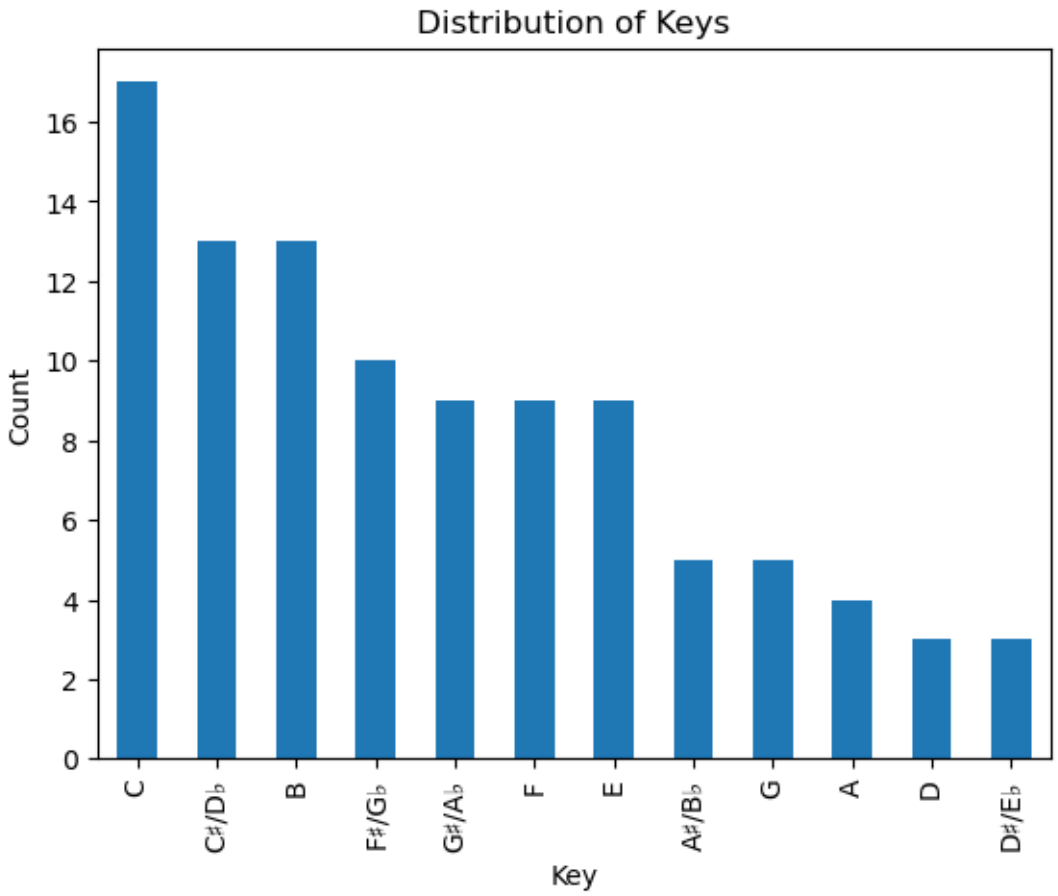
[50]:
```
# Plot the relationship between speechiness and instrumentalness using a␣
 ↪scatter plot
sns.scatterplot(data=df, x='speechiness', y='instrumentalness')
plt.title('Speechiness vs Instrumentalness')
plt.savefig('scatter_speechiness_instrumentalness.png', bbox_inches='tight')
plt.show()
graph1_div = soup.new_tag("div", id="graph10")
img_tag1 = soup.new_tag('img', src='scatter_speechiness_instrumentalness.png')
graph1_div.append(img_tag1)
soup.body.append(graph1_div)
```



[53]:
```
# Plot the distribution of mode using a pie chart
mode_count = df.groupby('mode').size().rename({1: 'Major key', 2: 'Minor key'})
plt.figure(figsize=(5, 5))
plt.pie(mode_count.values, labels=mode_count.index, autopct='%1.1f%%')
plt.title("Mode Distribution")

plt.savefig('pie chart_mode.png', bbox_inches='tight')
plt.show()
graph1_div = soup.new_tag("div", id="graph11")
img_tag1 = soup.new_tag('img', src='pie chart_mode.png')
graph1_div.append(img_tag1)
soup.body.append(graph1_div)
```

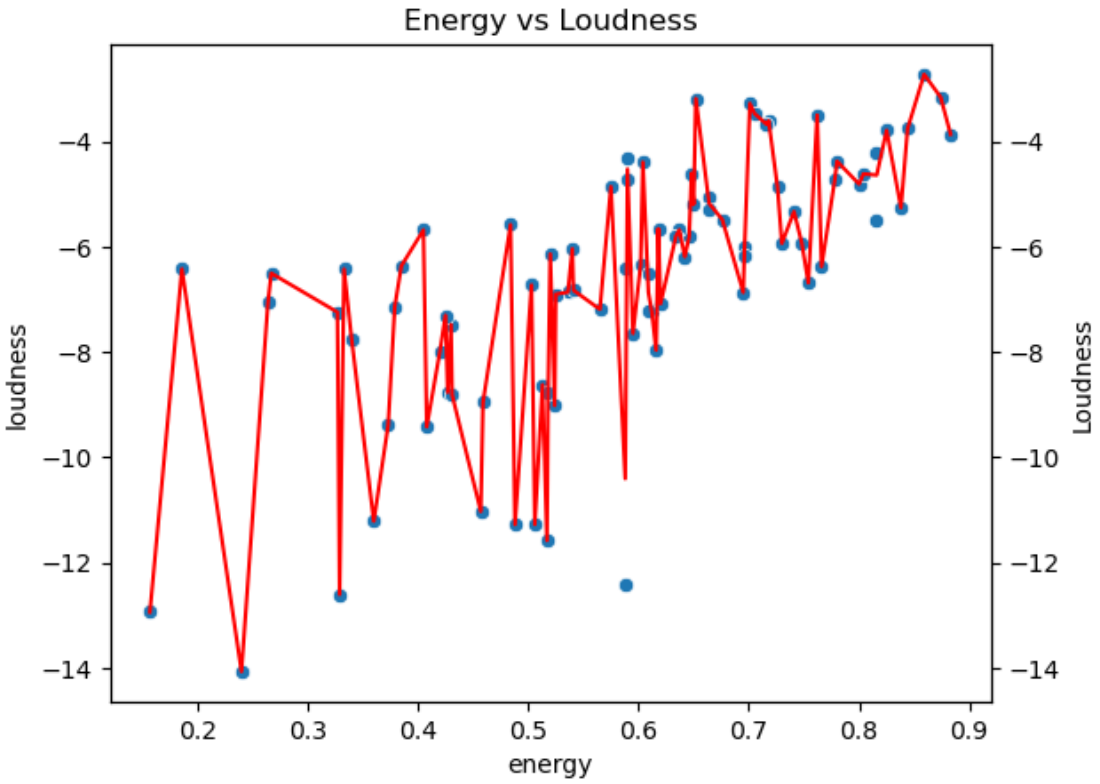


[54]:
```
# Plot the distribution of keys using a bar plot
# create a dictionary that maps integers to pitch names
pitch_map = {0: 'C', 1: 'C /D ', 2: 'D', 3: 'D /E ', 4: 'E', 5: 'F', 6: 'F /G ', 7:
 ↪ 'G', 8: 'G /A ', 9: 'A', 10: 'A /B ', 11: 'B'}

# replace integers in 'key' column with pitch names
df['key'] = df['key'].replace(pitch_map)

# plot distribution of keys
df['key'].value_counts().plot(kind='bar')
plt.xlabel('Key')
plt.ylabel('Count')
plt.title('Distribution of Keys')

plt.savefig('bar plot_keys.png', bbox_inches='tight')
plt.show()
graph1_div = soup.new_tag("div", id="graph12")
img_tag1 = soup.new_tag('img', src='bar plot_keys.png')
graph1_div.append(img_tag1)
soup.body.append(graph1_div)
```



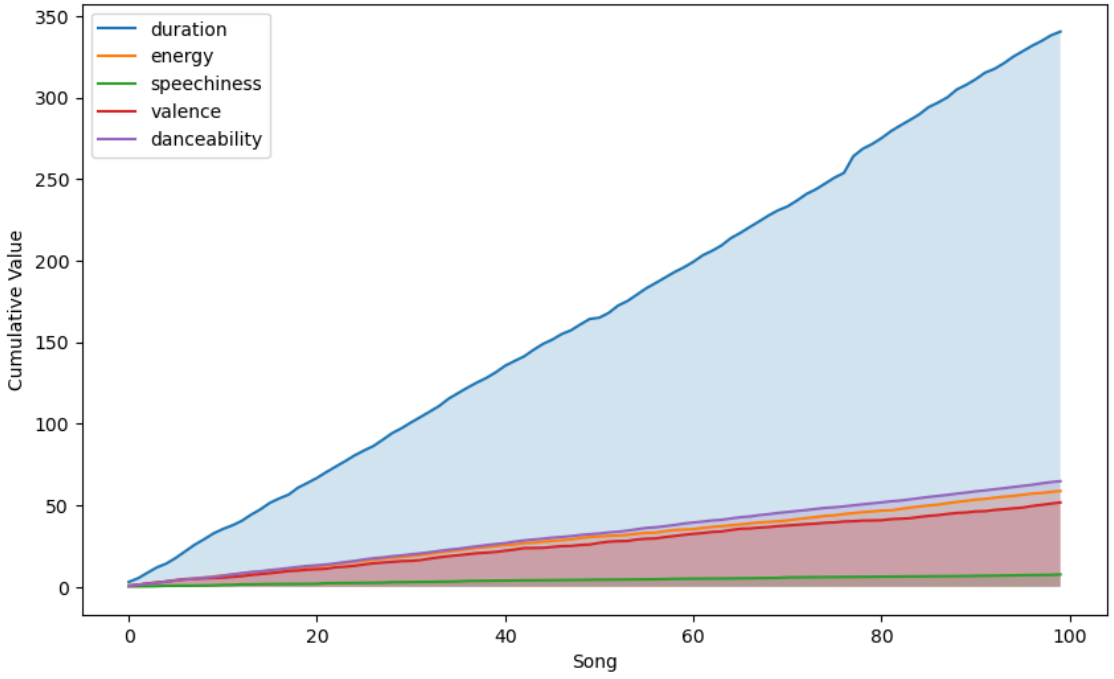
[55]:
```
# Create a scatter plot of energy vs loudness using seaborn
sns.scatterplot(x='energy', y='loudness', data=df)

# Add a secondary y-axis on the right side of the plot
ax2 = plt.twinx()

# Create a line plot of the mean loudness for each energy value
sns.lineplot(x='energy', y='loudness', data=df.groupby('energy')['loudness'].
 ↪mean().reset_index(), ax=ax2, color='red')

# Set the x and y axis labels and titles
plt.xlabel('Energy')
plt.ylabel('Loudness')
plt.title('Energy vs Loudness')

plt.savefig('double axis_energy_loudness.png', bbox_inches='tight')

plt.show()
```

```
graph1_div = soup.new_tag("div", id="graph13")
img_tag1 = soup.new_tag('img', src='double axis_energy_loudness.png')
graph1_div.append(img_tag1)
soup.body.append(graph1_div)
#dual axis
```

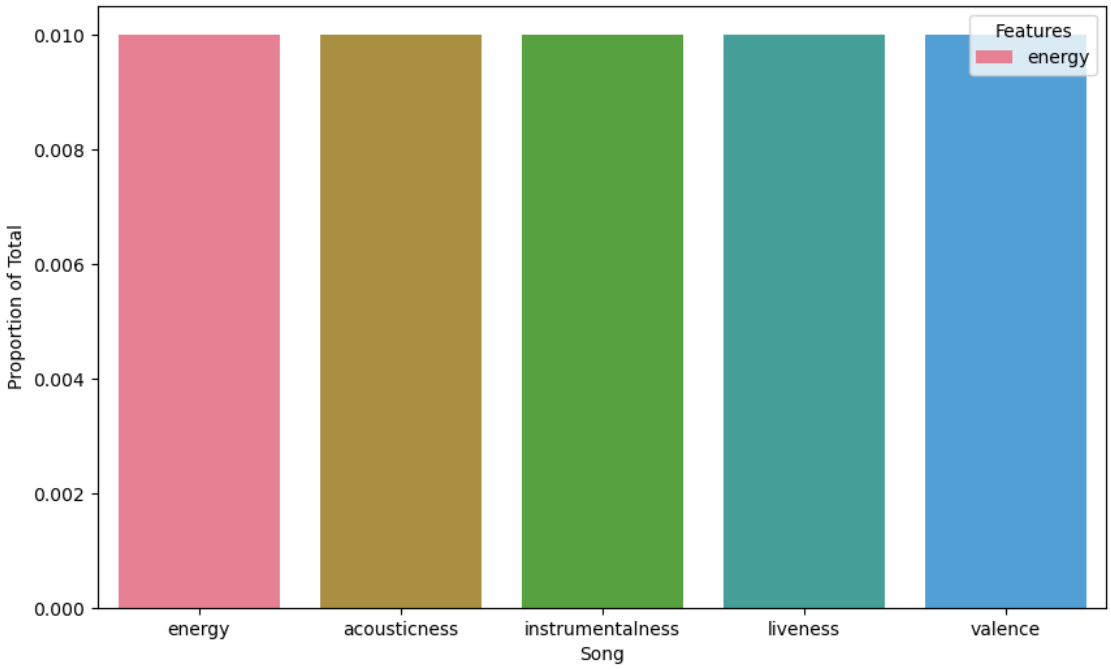


[56]:
```
#area chart between, duration, energy, speechiness, valence, daneability
columns = ["duration", "energy", "speechiness", "valence", "danceability"]
data = df[columns]
cumulative_data = np.cumsum(data, axis=0)
plt.figure(figsize=(10, 6))
for column in columns:
    sns.lineplot(data=cumulative_data[column], label=column)
    plt.fill_between(cumulative_data.index, cumulative_data[column], alpha=0.2)
plt.legend()
plt.xlabel("Song")
plt.ylabel("Cumulative Value")
plt.savefig('area_duration_etc.png', bbox_inches='tight')
plt.show()
graph1_div = soup.new_tag("div", id="graph14")
img_tag1 = soup.new_tag('img', src='area_duration_etc.png')
```

```
graph1_div.append(img_tag1)
soup.body.append(graph1_div)
```

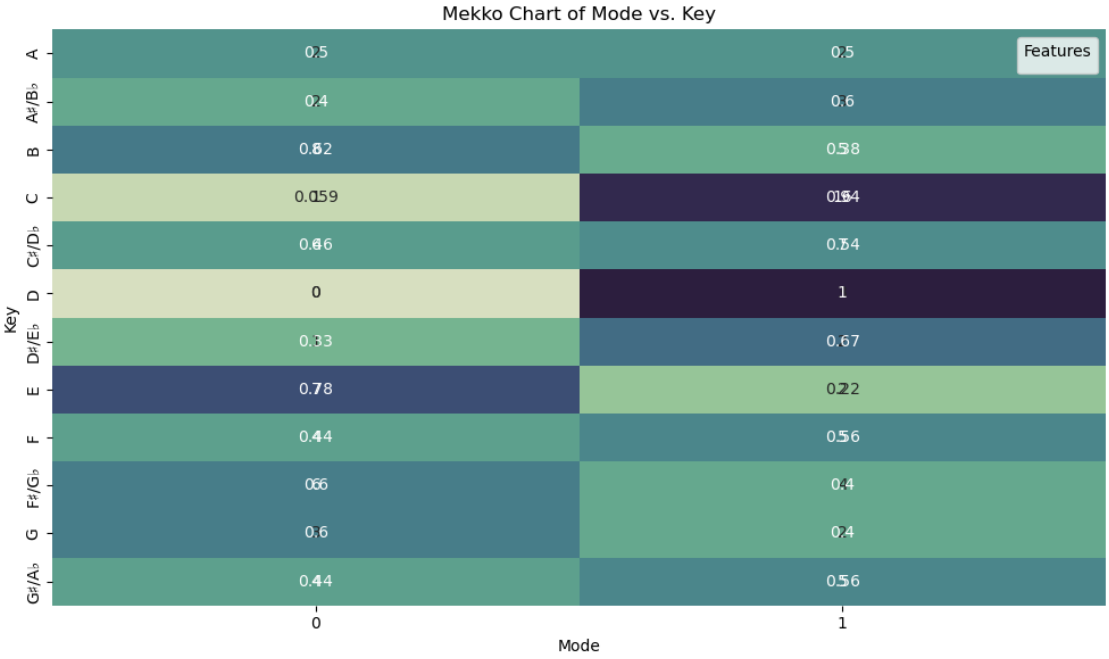


[57]:
```
#stacked bar graph energy, acousticness, instrumentalness, liveness, valence
columns = ["energy", "acousticness", "instrumentalness", "liveness", "valence"]
data = df[columns]
sums = data.sum(axis=0)
proportions = data.divide(sums, axis=1)
plt.figure(figsize=(10, 6))
sns.set_palette("husl")
sns.barplot(data=proportions, ci=None)
plt.legend(title="Features", labels=columns)
plt.xlabel("Song")
plt.ylabel("Proportion of Total")
plt.savefig('stacked bar_energy_etc.png', bbox_inches='tight')
plt.show()
graph1_div = soup.new_tag("div", id="graph15")
img_tag1 = soup.new_tag('img', src='stacked bar_energy_etc.png')
graph1_div.append(img_tag1)
soup.body.append(graph1_div)
```

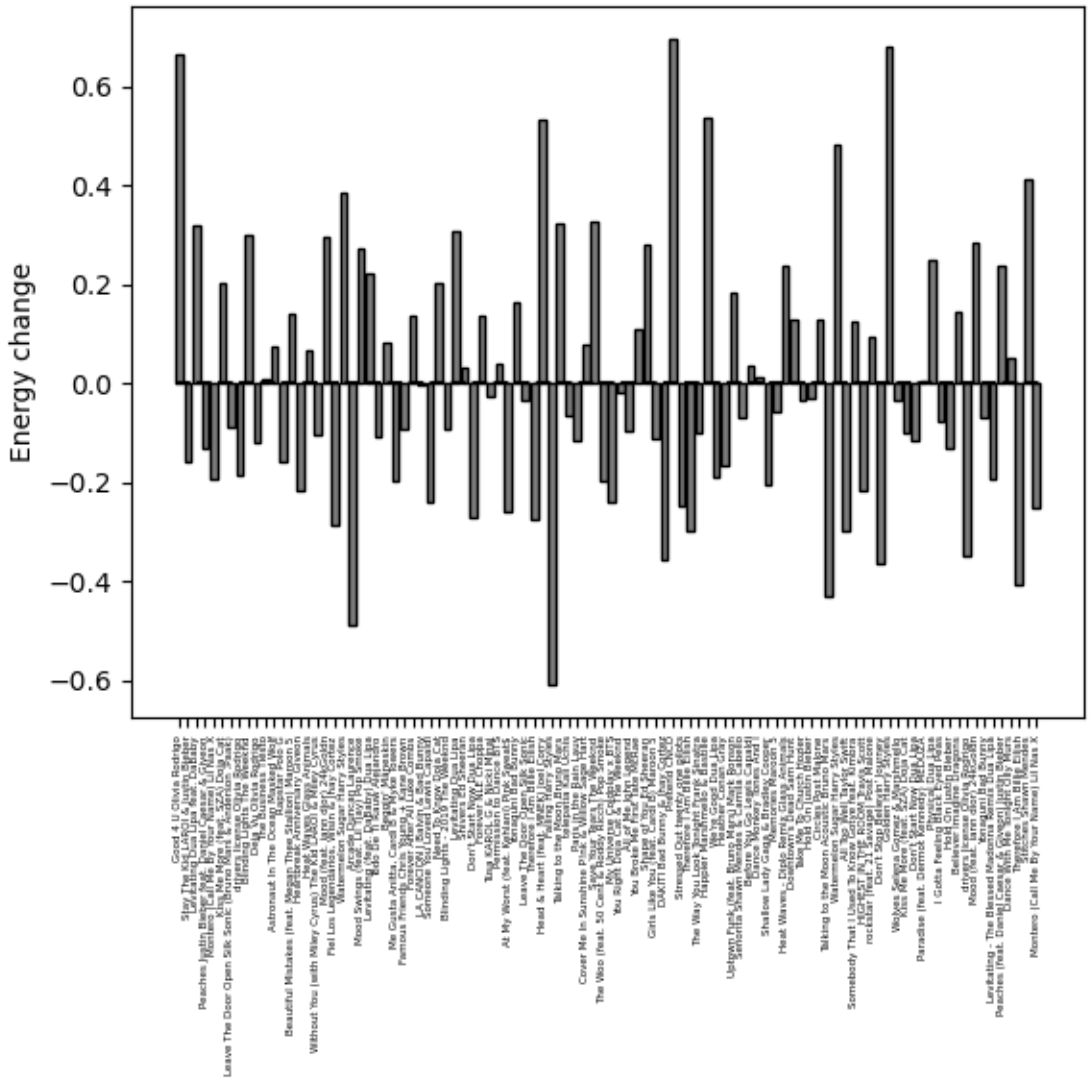


[65]:
```
#mekko chart between mode and key
#mekko chart between mode and key
columns = ["mode", "key"]
data = df[columns]
table = pd.crosstab(data["key"], data["mode"])
row_totals = table.sum(axis=1)
col_totals = table.sum(axis=0)
row_proportions = table.div(row_totals, axis=0)
col_proportions = table.div(col_totals, axis=1)

plt.figure(figsize=(10, 6))
cmap = sns.cubehelix_palette(8, start=.5, rot=-.75, as_cmap=True) # choose a␣
 ↪colormap for the heatmap
sns.heatmap(col_proportions, cmap=cmap, annot=table, fmt="d",␣
 ↪cbar_kws={'format': '%d%%'}, cbar=False) # remove the color bar on the right␣
 ↪and modify the annot parameter
sns.heatmap(row_proportions, cmap=cmap, annot=True, cbar=False)
plt.legend(title="Features", labels=columns)
plt.xlabel("Mode")
plt.ylabel("Key")
plt.title("Mekko Chart of Mode vs. Key")
plt.tight_layout()
plt.savefig('mekko_mode_key.png', dpi=300, bbox_inches='tight') # increase dpi␣
 ↪for better image quality
plt.show()
```

```
graph1_div = soup.new_tag("div", id="graph16")
img_tag1 = soup.new_tag('img', src='mekko_mode_key.png')
graph1_div.append(img_tag1)
soup.body.append(graph1_div)
```

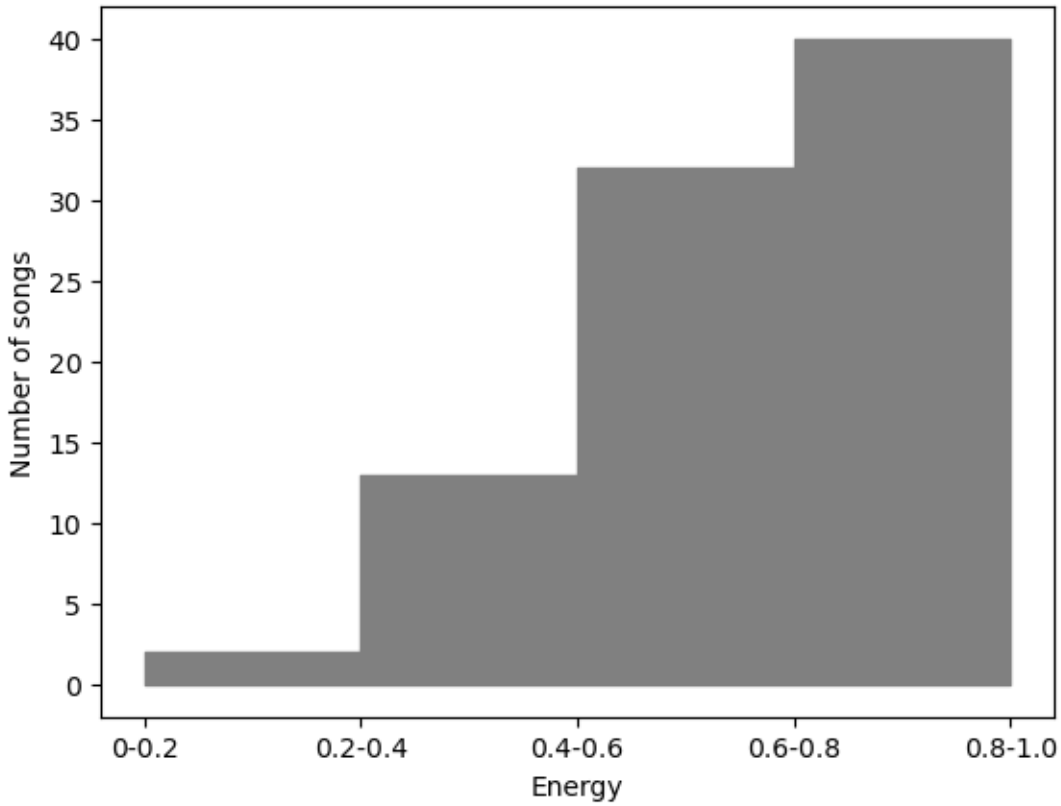


[39]:
```
#waterfall chart for energy
data = df["energy"]
changes = [data.iloc[0]] + [(data.iloc[i] - data.iloc[i-1]) for i in range(1,␣
 ↪len(data))]
plt.bar(range(len(data)), changes, color="gray", edgecolor="black")
plt.plot([-0.5, len(data)-0.5], [0, 0], "k--")
plt.xticks(range(len(data)), df["name"], rotation=90,fontsize=4)
plt.ylabel("Energy change")
plt.savefig('energy_change.png', bbox_inches='tight')
plt.show()
graph1_div = soup.new_tag("div", id="graph1")
img_tag1 = soup.new_tag('img', src='energy_change.png')
graph1_div.append(img_tag1)
soup.body.append(graph1_div)
```



[61]:
```
#funnel chart for energy
data = df["energy"]
bins = np.linspace(0, 1, 6)
counts, _ = np.histogram(data, bins=bins)
plt.fill_between(range(len(counts)), counts, step="post", color="gray")
plt.xticks(range(len(bins)-1), ["0-0.2", "0.2-0.4", "0.4-0.6", "0.6-0.8", "0.
↪8-1.0"])
plt.xlabel("Energy")
plt.ylabel("Number of songs")
plt.savefig('funnel_energy.png', bbox_inches='tight')
plt.show()
graph1_div = soup.new_tag("div", id="graph17")
img_tag1 = soup.new_tag('img', src='funnel_energy.png')
```

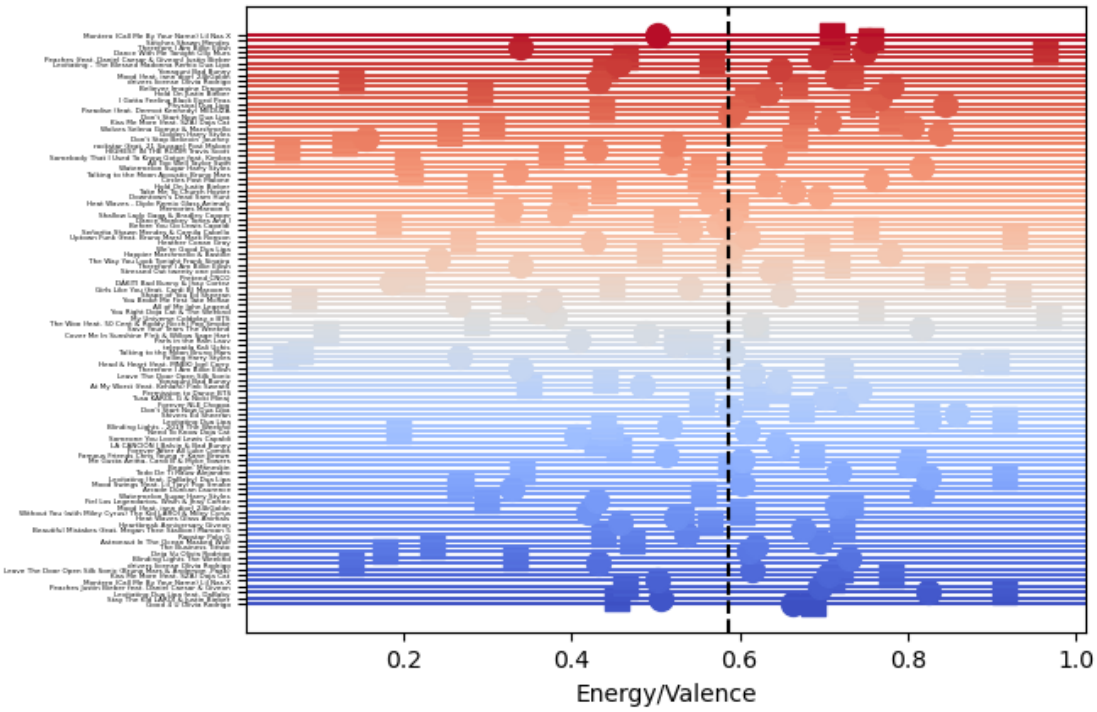
```
graph1_div.append(img_tag1)
soup.body.append(graph1_div)
```




```
<Figure size 640x480 with 0 Axes>
```

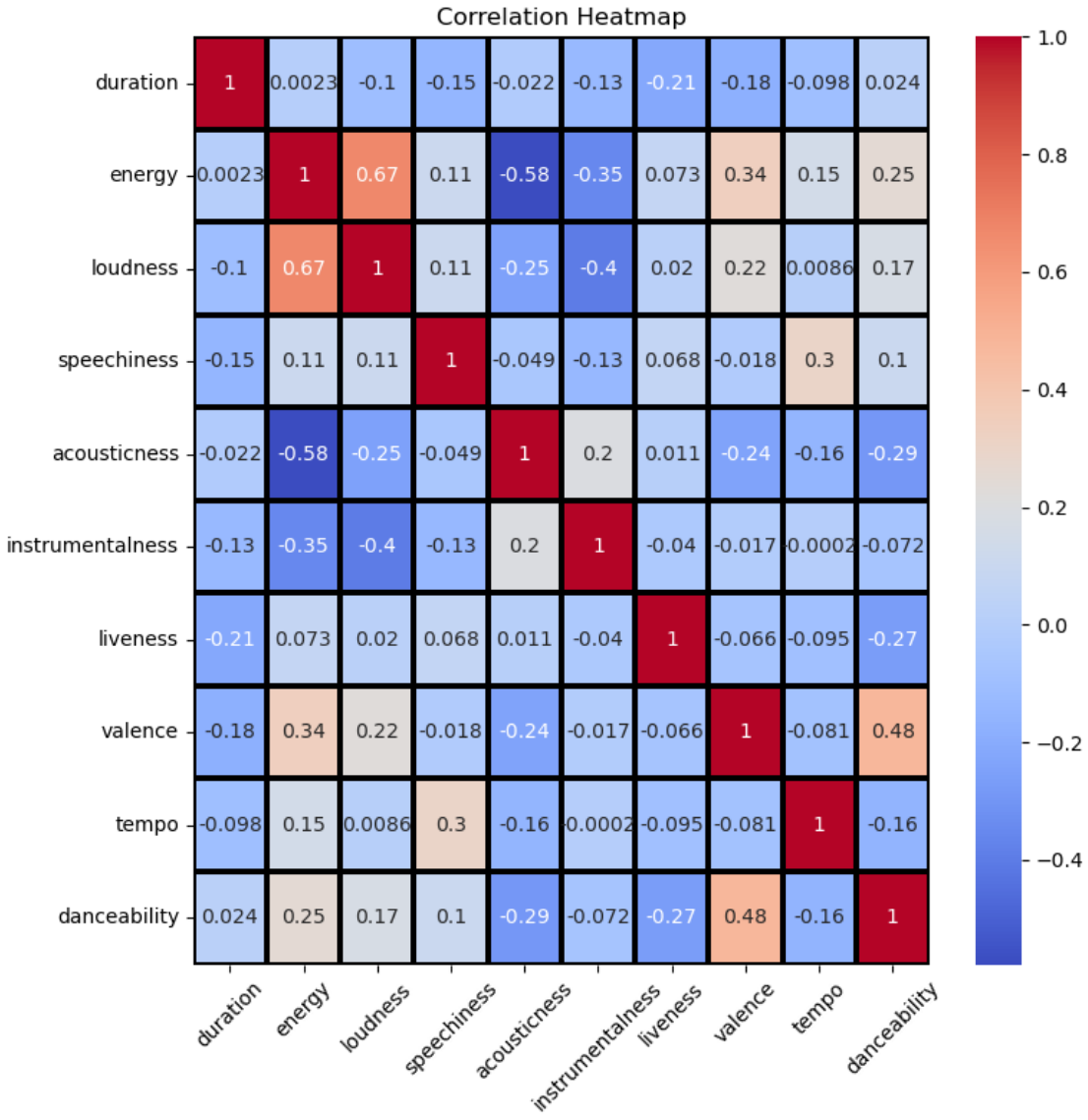
[62]:
```
#bullet chart between energy and valence
data = df[["name", "energy", "valence"]]
target_energy = data["energy"].mean()
colors = sns.color_palette("coolwarm", len(data))
for i, row in data.iterrows():
    plt.axhline(y=i, color=colors[i])
    plt.scatter(row["energy"], i, color=colors[i], s=100, zorder=2)
    plt.scatter(row["valence"], i, marker="s", color=colors[i], s=100, zorder=1)
plt.axvline(x=target_energy, color="black", linestyle="--")
plt.yticks(range(len(data)), data["name"],fontsize=3)
plt.xlabel("Energy/Valence")
plt.savefig('bullet_energy_valence.png', bbox_inches='tight')
plt.show()
graph1_div = soup.new_tag("div", id="graph18")
img_tag1 = soup.new_tag('img', src='bullet_energy_valence.png')
graph1_div.append(img_tag1)
```

```
soup.body.append(graph1_div)
```

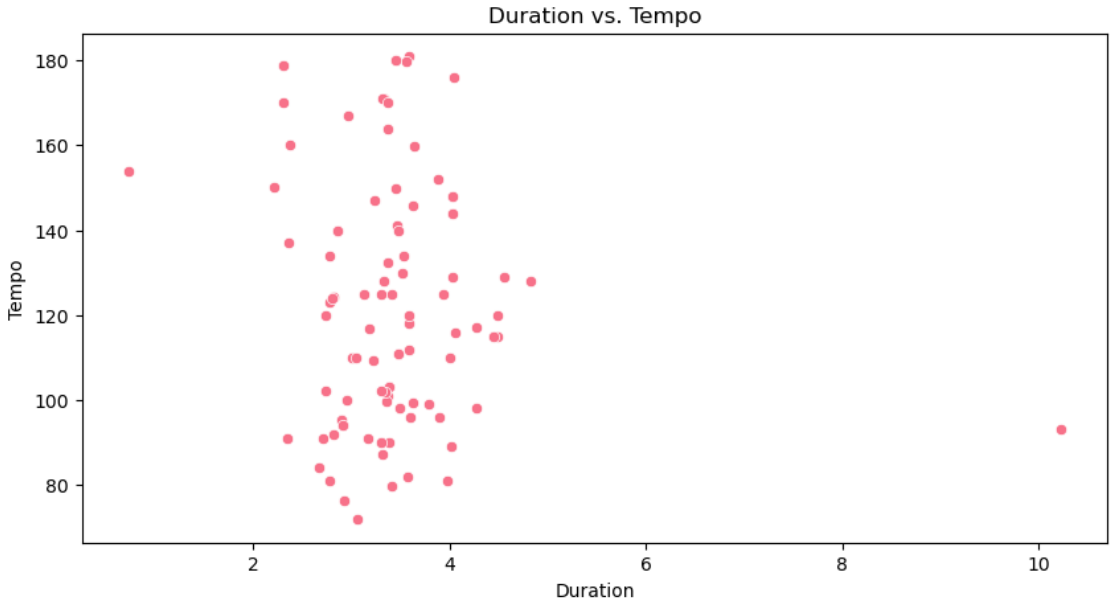



```
<Figure size 640x480 with 0 Axes>
```

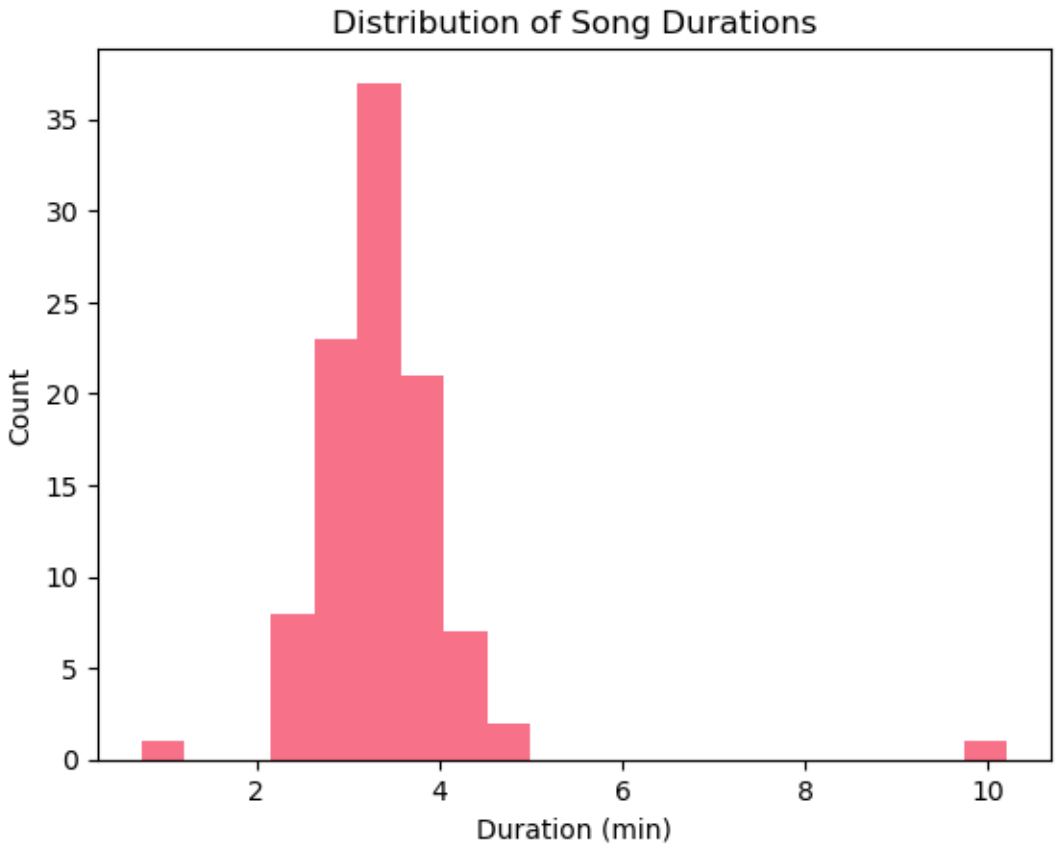
[63]:
```
#heatmap between duration, energy, loudness, speechniess, acousticness,␣
 ↪instrumentalness, liveness, valence, tempo, danceability
data = df[["duration", "energy", "loudness", "speechiness", "acousticness",␣
 ↪"instrumentalness", "liveness", "valence", "tempo", "danceability"]]
corr = data.corr()

plt.figure(figsize=(8,8))  # adjust the figure size
sns.heatmap(corr, annot=True, cmap="coolwarm", linewidths=1.5,␣
 ↪linecolor="black")  # adjust the linewidths and linecolor
plt.title("Correlation Heatmap")
plt.xticks(rotation=45)  # rotate x-axis labels for better visibility
plt.tight_layout()  # fix layout to prevent labels from being cut off
plt.savefig('heatmap.png', bbox_inches='tight')
plt.show()
graph1_div = soup.new_tag("div", id="graph19")
img_tag1 = soup.new_tag('img', src='heatmap.png')
graph1_div.append(img_tag1)
soup.body.append(graph1_div)
```



[67]:
```
# Plot a scatter plot of duration vs. tempo
plt.figure(figsize=(10, 5))
sns.scatterplot(x='duration', y='tempo', data=df)
plt.title("Duration vs. Tempo")
plt.xlabel("Duration")
plt.ylabel("Tempo")
plt.savefig('scatter_duration_tempo.png', bbox_inches='tight')
plt.show()
graph1_div = soup.new_tag("div", id="graph20")
img_tag1 = soup.new_tag('img', src='scatter_duration_tempo.png')
graph1_div.append(img_tag1)
soup.body.append(graph1_div)
```



[70]:
```
# Plot the distribution of song durations using a histogram
plt.hist(df['duration'], bins=20)
plt.xlabel('Duration (min)')
plt.ylabel('Count')
plt.title('Distribution of Song Durations')
plt.savefig('histogram_duration.png', bbox_inches='tight')
plt.show()
graph1_div = soup.new_tag("div", id="graph1")
img_tag1 = soup.new_tag('img', src='histogram_duration.png')
graph1_div.append(img_tag1)
soup.body.append(graph1_div)
```



[71]:
```
with open('python_EL.html', 'w') as f:
    f.write(str(soup))
IFrame(src='python_EL.html', width=700, height=600)
```

[71]: <IPython.lib.display.IFrame at 0x7d9774ddf340>

[ ]: