Analysis of Global Musical Artist Listeners and Genres

Data

Source
The original dataset was downloaded from Kaggle and contains data from MusicBrainz and LastFM.
https://www.kaggle.com/datasets/pieca111/music-artists-popularity

Cleaning
Due to the size and crowdsourced nature of the dataset, some cleaning was necessary to make it manageable for our analysis. Steps taken include:

- Deleted artists with less than 100,000 listeners, shrinking the file size from 1.04 million lines to under 8000 lines.
- Reduced the number of columns artist, country, tags (genres), listeners
- Removed any duplicate columns, using all data from LastFM. Any MusicBrainz columns were removed.
- Modified the country column to contain only the first value in instances where more than one country was listed. Countries are listed alphabetically, therefore we acknowledge that this weights countries toward front of alphabet. For example, if an artist is listed as being from both Armenia and the United States, Armenia would be the country left in the cleaned dataset.
- Reduced tags to the first five in separate columns: tags1_lastfm, tags2_lastfm, tags3_lastfm, tags4_lastfm, tags5_lastfm. Tags are user-generated, are not a controlled vocabulary, and appeared to be random in order. As such, tags are inconsistently applied, sometimes repetitive, and not necessarily accurate. They are, however, the closest approximation to music genre in this dataset.

Questions

Main Question:
Given the fields in the principle dataset: artist, country of origin (by artist), number of monthly listeners on LastFM, and user-generated tags, what comparisons can we make across those fields?

Sub Questions:
How many listeners does each top artist have?
How many listeners does each top country have?
What genres have the most listeners?
How many artists are associated with each genre?
What are the top most common genres by country?
Compare ambiguous artists (true/false), sum listeners.

Analysis

Import necessary packages

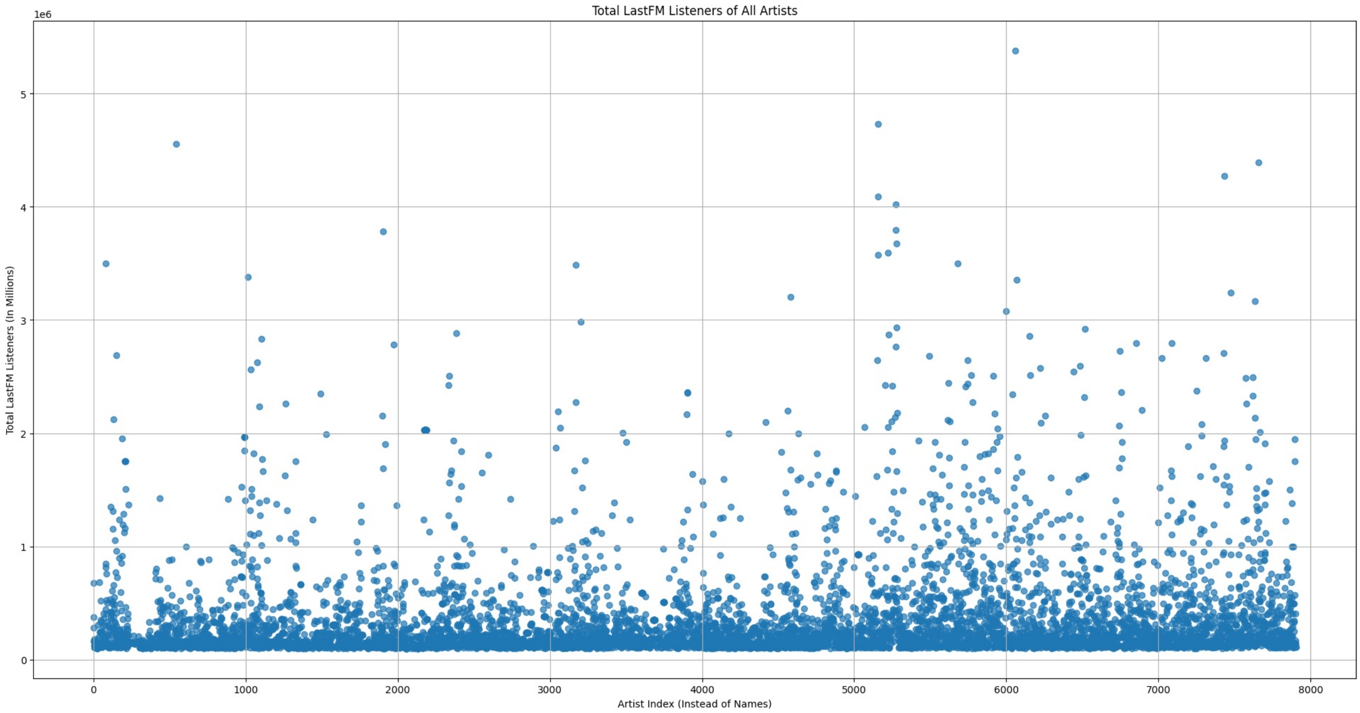
In [1]:
```
# IMPORTS
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
import openpyxl
import seaborn as sns
import ipywidgets as widgets
from ipywidgets import widgets, interactive, Layout
```

Top Artists By Listeners

We use a scatter plot to visualize the total number of listerners per artist (using artist index).

In [2]:
```
data = pd.read_csv('Mapd.csv', low_memory=False) #import data

data['artist_lastfm'] = data['artist_lastfm'].astype(str) #make the artist names consistent

data = data.dropna(subset=['artist_lastfm', 'listeners_lastfm']) #remove n/a

data['artist_lastfm'] = data['artist_lastfm'].str.replace('$', 'S', regex=False) #change dollar signs in artist

plt.figure(figsize=(24, 12))
```

```
plt.scatter(data.index, data['listeners_lastfm'], alpha=0.7) #creating a scatter plot with the artist index pos
plt.xlabel('Artist Index (Instead of Names)')
plt.ylabel('Total LastFM Listeners (In Millions)')
plt.title('Total LastFM Listeners of All Artists') #adding labels and titles
plt.grid(True) #adding grid lines for readability
```



The scatter plot was not helpful in analyzing the data, so we first displayed it as a tabular sorted list.

In [3]:
```
sort_top25_data = data.sort_values(by = "listeners_lastfm", ascending = False) #sorting the data into a list fr
sort_top25_data.head(25) #showing the first 25 artists, by listeners, from the list
```

Out[3]:

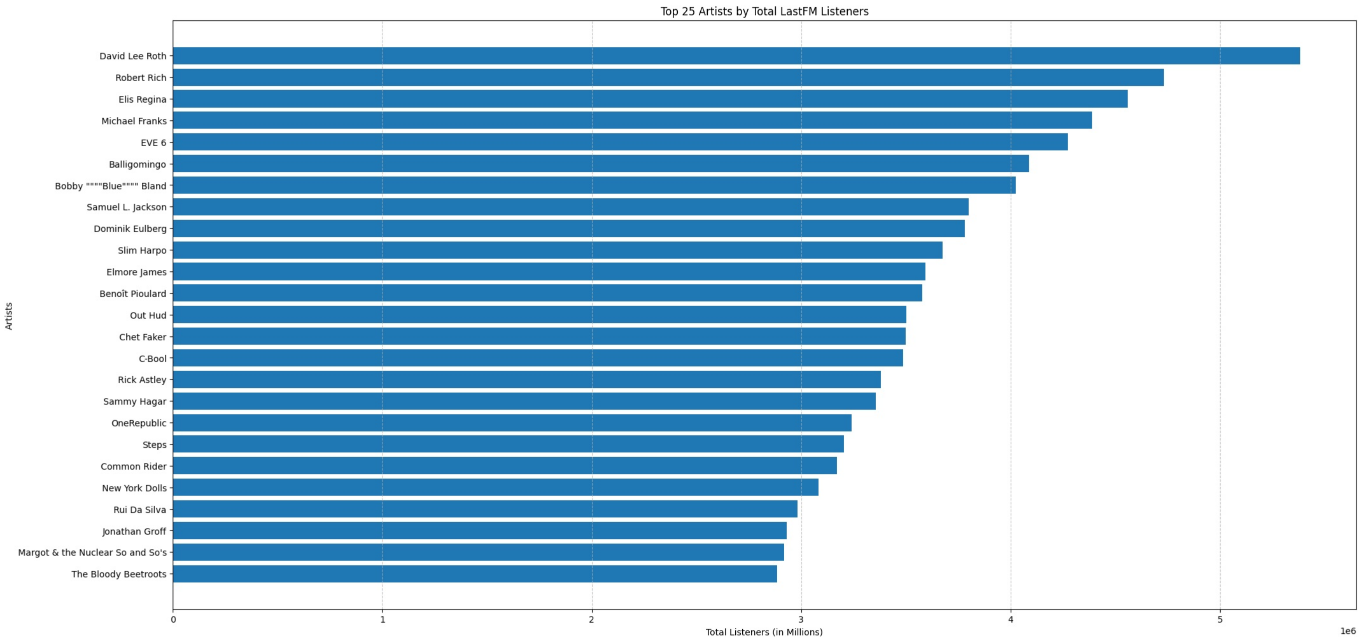
	mbid	artist_lastfm	country_lastfm	tags1_lastfm	tags2_lastfm	tags3_lastfm	tags4_lastfm	tags5_lastfm	tags_lastfm
6060	cc197bad-dc9c-440d-a5b5-d52ba2e14234	David Lee Roth	United Kingdom	rock	alternative	britpop	alternative rock	indie	rock alternative britpop alternative rock indie
5156	a74b1b7f-71a5-4011-9441-d0b5e4122711	Robert Rich	United Kingdom	alternative	alternative rock	rock	indie	electronic	alternative alternative rock rock indie electr..
544	73e5e69d-3554-40d8-8516-00cb38737a1c	Elis Regina	Barbados	pop	rnb	female vocalists	dance	Hip-Hop	pop rnb female vocalists dance Hip-Hop
7659	164f0d73-1234-4e2c-8743-d77bf2191051	Michael Franks	United States	Hip-Hop	rap	hip hop	rnb	Kanye West	Hip-Hop rap hip hop rnb Kanye Wes
7433	5b11f4ce-a62d-471e-81fc-a69a8278c7da	EVE 6	United States	Grunge	rock	alternative	alternative rock	90s	Grunge rock alternative alternative rock 90s
5155	9c9f1380-2516-4fc9-a3e6-f9f61941d090	Balligomingo	United Kingdom	alternative rock	rock	alternative	Progressive rock	seen live	alternative rock rock alternative Progressive ..
5272	0383dadf-2a4e-4d10-a46a-e9e041da8eb3	Bobby """"Blue"""" Bland	United Kingdom	classic rock	rock	80s	hard rock	glam rock	classic rock rock 80s hard rock glam rock
5274	b071f9fa-14b0-4217-8e97-eb41da73f598	Samuel L. Jackson	United Kingdom	classic rock	rock	british	60s	blues	classic rock rock british 60s blues
	056e4f3e-d505-4dad-	Dominik							electronic dance

1903	8ec1-d04f521cbb56	Eulberg	France	electronic	dance	House	electronica	techno	House electronica techno
5276	b10bbbfc-cf9e-42e0-be17-e2c3e1d2600d	Slim Harpo	United Kingdom	classic rock	rock	british	60s	pop	classic rock rock british 60s pop
5224	39ab1aed-75e0-4140-bd47-540276886b60	Elmore James	United Kingdom	britpop	rock	british	alternative	indie	britpop rock british alternative indie
5157	e21857d5-3256-4547-afb3-4b6ded592596	Benoît Pioulard	United Kingdom	alternative	electronic	Hip-Hop	rock	indie	alternative electronic Hip-Hop rock indie
5682	ada7a83c-e3e1-40f1-93f9-3e73dbc9298a	Out Hud	United Kingdom	indie rock	indie	british	rock	alternative	indie rock indie british rock alternative
81	cc0b7089-c08d-4c10-b6b0-873582c17fd6	Chet Faker	Armenia	metal	alternative metal	rock	Nu Metal	alternative	metal alternative metal rock Nu Metal alternative
3170	a3cb23fc-acd3-4ce0-8f36-1e5aa6a18432	C-Bool	Ireland	rock	classic rock	irish	pop	alternative	rock classic rock irish pop alternative
1017	9fff2f8a-21e6-47de-a2b8-7f449929d43f	Rick Astley	Canada	Hip-Hop	rap	rnb	hip hop	Canadian	Hip-Hop rap rnb hip hop Canadian
6068	5441c29d-3602-4898-b1a1-b77fa23b8e50	Sammy Hagar	United Kingdom	rock	glam rock	classic rock	80s	alternative	rock glam rock classic rock 80s alternative
7473	eeb1195b-f213-4ce1-b28c-8565211f8e43	OneRepublic	United States	hard rock	rock	classic rock	80s	metal	hard rock rock classic rock 80s metal
4584	aa7a2827-f74b-473c-bd79-03d065835cf7	Steps	Scotland	indie	indie rock	rock	alternative	seen live	indie indie rock rock alternative seen live
7635	f82bcf78-5b69-4622-a5ef-73800768d9ac	Common Rider	United States	Hip-Hop	rap	hip hop	east coast rap	jay-z	Hip-Hop rap hip hop east coast rap jay-z
5997	83d91898-7763-47d7-b03b-b92132375c47	New York Dolls	United Kingdom	Progressive rock	classic rock	Psychedelic Rock	rock	psychedelic	Progressive rock classic rock Psychedelic Rock..
3201	a66999a7-ae5c-460e-ba94-1a01143ae847	Rui Da Silva	Ireland	indie	alternative	rock	indie rock	britpop	indie alternative rock indie rock britpop
5279	678d88b2-87b0-403b-b63d-5da7465aecc3	Jonathan Groff	United Kingdom	classic rock	rock	hard rock	70s	Progressive rock	classic rock rock hard rock 70s Progressive rock
6515	309c62ba-7a22-4277-9f67-4a162526d18a	Margot & the Nuclear So and So's	United States	alternative	indie	rock	singer-songwriter	indie rock	alternative indie rock singer-songwriter indie..
2385	ea4dfa26-f633-4da6-a52a-f49ea4897b58	The Bloody Beetroots	Georgia	rock	alternative rock	alternative	indie	seen live	rock alternative rock alternative indie seen live

Then we visualized the data using a horizontal bar chart. Instead of artist index, we used the name of the artist.

In [4]:
```
top_25_artists = data.sort_values(by='listeners_lastfm', ascending=False).head(25)
```

```python
plt.figure(figsize=(24, 12))
plt.barh(top_25_artists['artist_lastfm'], top_25_artists['listeners_lastfm']) #creating a horizontal barchart o
plt.xlabel('Total Listeners (in Millions)')
plt.ylabel('Artists')
plt.title('Top 25 Artists by Total LastFM Listeners') #labeling and adding titles
plt.gca().invert_yaxis() #inverting the y-axis so the artist with the highest amount of listeners is up top
plt.grid(axis='x', linestyle='--', alpha=0.7) #adding grid lines

plt.show()
```

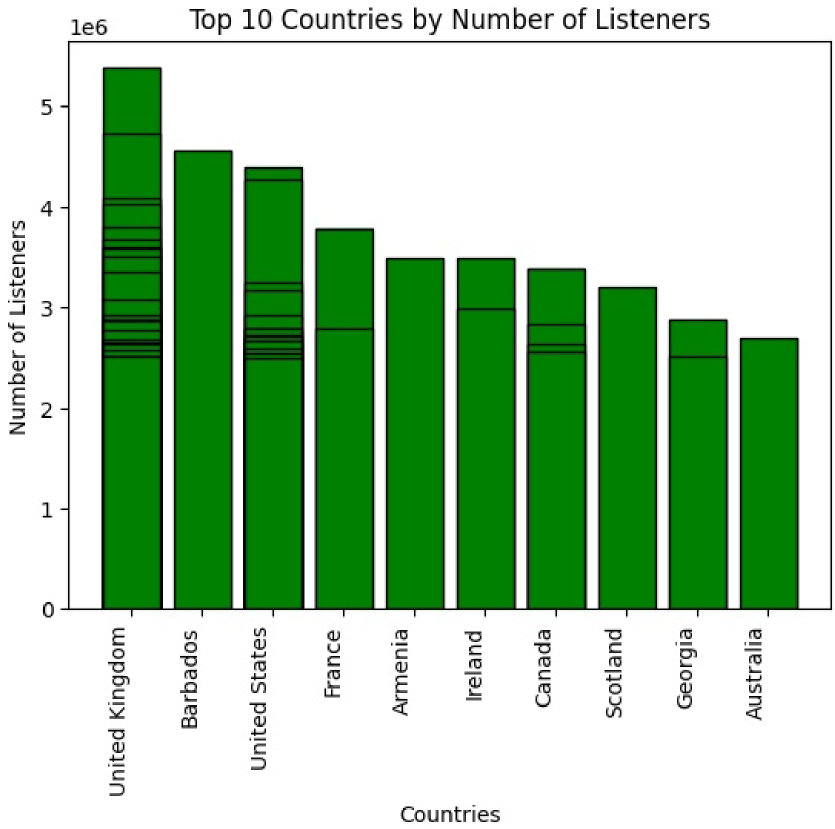


The top artist by LastFM listeners, David Lee Roth, had over 5 million LastFM listeners and had 600,000 and 800,000 more listeners than the 2nd and 3rd top artists, respectively. The top seven aritsts all had over 4 million listeners.

Top 10 Countries By Number of Listener

We use a bar chart to show the top ten countries of artist origin with the highest total number of listerners.

In [5]:
```python
# 'low_memory=False' for better handling of large files
data = pd.read_csv('Mapd.csv', low_memory=False)
# Convert the 'country_lastfm' column to string
data['country_lastfm'] = data['country_lastfm'].astype(str)
# removes any rows or columns with no data
data = data.dropna(subset=['country_lastfm', 'listeners_lastfm'])
#sorts the data descending and selects top 50
top10 = data.sort_values(by='listeners_lastfm', ascending=False).head(50)
# tells the code which columns to look at
countries = top10['country_lastfm']
listeners = top10['listeners_lastfm']
# creates a bar chart
plt.bar(countries, listeners, color='green', edgecolor='black')
# adds detail to the bar chart
plt.title('Top 10 Countries by Number of Listeners')
plt.xlabel('Countries')
plt.ylabel('Number of Listeners')
# adjusts the xaxis label
plt.xticks(rotation=90, ha='right')
#displays the bar chart
plt.show()
```



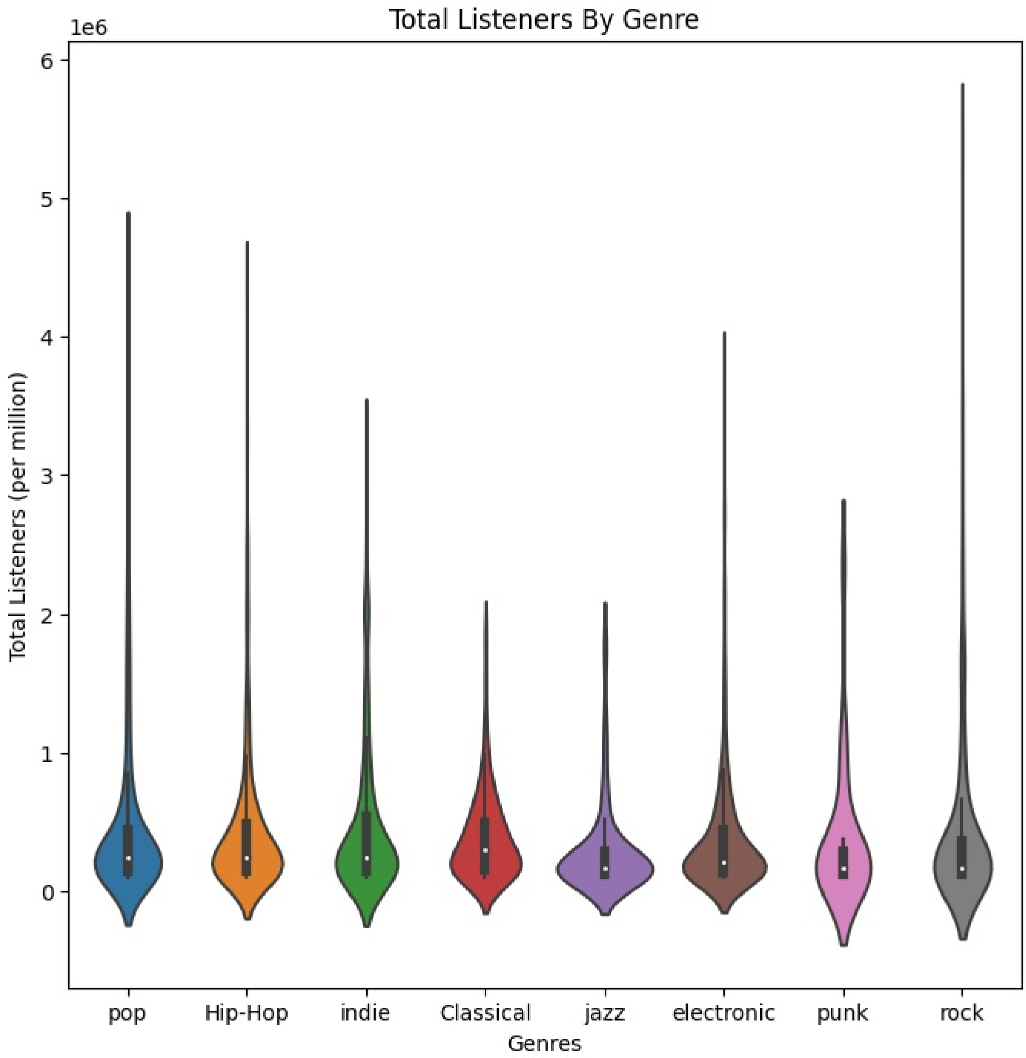
Top Tags By Total Listeners

We display the top genres by total listeners using a violin plot.

In [6]:
```
# Show the count of each tag/genre that appears throughout the dataset
df = pd.DataFrame(data)

counts = df['tags1_lastfm'].value_counts()
print(counts)
```

```
tags1_lastfm
electronic          485
Hip-Hop             467
indie               381
pop                 294
rock                241
                   ...
glitch-hop            1
Christian Rap         1
Elephant 6            1
a capella             1
a cappella metal      1
Name: count, Length: 348, dtype: int64
```

In [7]:
```
df = pd.DataFrame(data)

# Create a new Data Frame only including the most popular Genres in the dataset.

genres = ['rock', 'pop', 'Hip-Hop', 'jazz', 'electronic', 'Classical', 'indie', 'punk']
df_filter = df[df['tags1_lastfm'].isin(genres)]

# Once the counts of the most popular genres are identified, we can create our visualization

plt.figure(figsize=(8,8))
sns.violinplot(data=df_filter, x='tags1_lastfm', y='listeners_lastfm')

# Provide proper Labels and Titles

plt.xlabel('Genres')
plt.ylabel('Total Listeners (per million)')
plt.title('Total Listeners By Genre')

plt.show()
```



Number of Artists by Genre

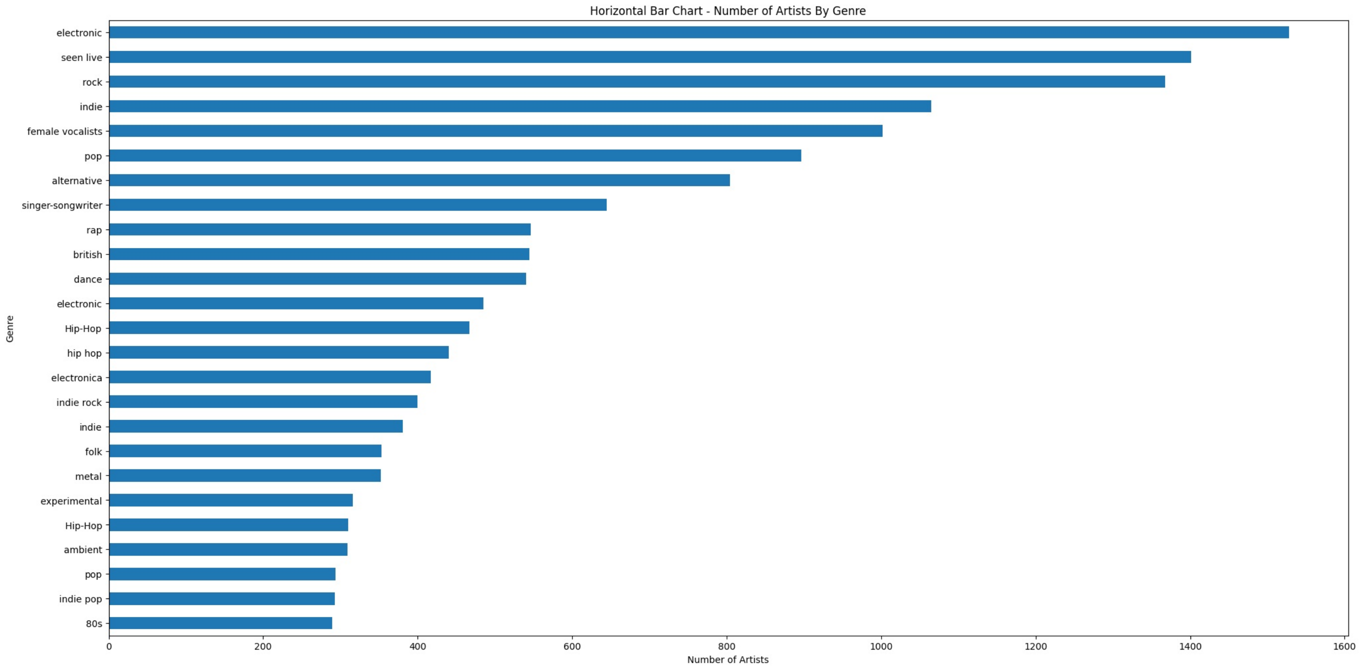
Using a horizontal bar graph, we visual the number of artists associated with top tags.

In [8]:
```
#Load the .csv dataset into a dataframe
df = pd.read_csv('Mapd.csv', low_memory=False)

# Track count of artist genres
artist_counts = {}

for index, row in df.iterrows():
    # Get the genres from the 5 tag columns and get rid of duplicates and blanks
    genres = [row['tags1_lastfm'], row['tags2_lastfm'], row['tags3_lastfm'], row['tags4_lastfm'], row['tags5_la
    genres = [genre for genre in genres if pd.notna(genre) and genre.strip() != '']
    genres = list(set(genres))

    # Count the genres
    for genre in genres:
        if genre in artist_counts:
            artist_counts[genre] += 1
        else:
            artist_counts[genre] = 1

artist_counts_df = pd.DataFrame(list(artist_counts.items()), columns=['Genre', 'Artist Count'])
artist_counts_sorted = artist_counts_df.sort_values('Artist Count', ascending=False).reset_index(drop=True)

print(artist_counts_sorted)
```

```
                 Genre  Artist Count
0           electronic          1528
1            seen live          1401
2                 rock          1368
3                indie          1065
4      female vocalists          1002
...                ...           ...
2116  experimental rap             1
2117     Justus League             1
2118    FUCKING AWESOME            1
2119        electro-pop            1
2120        9th wonder             1

[2121 rows x 2 columns]
```

In [9]:

```python
#Get the sorted list of most popular genres
df_sorted = artist_counts_sorted.sort_values(by='Artist Count', ascending=False).head(25)
df_sorted.plot.barh(x='Genre', y='Artist Count', legend=False, figsize=(24,12))

#Plot the barh
plt.gca().invert_yaxis()
plt.xlabel('Number of Artists')
plt.ylabel('Genre')
plt.title('Horizontal Bar Chart - Number of Artists By Genre')

plt.show()
```



Top 5 Genres Per Country (Interactive Choice)

We use an dropdown selector to choose a country and then visualize the top five musical tags in that country using a bar chart.

In [10]:

```python
df = pd.read_csv("Mapd.csv", low_memory=False)

# User-defined function to get the most popular tags for a specific country
def get_popular_tags_by_country(country_lastfm):
    country_data = df[df['country_lastfm'] == country_lastfm]

    tags_columns = ['tags1_lastfm', 'tags2_lastfm', 'tags3_lastfm', 'tags4_lastfm', 'tags5_lastfm']
    all_tags = country_data[tags_columns].values.flatten() # Flatten tags into list

    all_tags = [tag for tag in all_tags if pd.notna(tag)] # Remove NaN

    tag_counts = pd.Series(all_tags).value_counts()

    return tag_counts.head(5) # Return top 5 tags

# User-defined function to be executed when a country is selected
def on_country_selected(country_lastfm):
    popular_tags = get_popular_tags_by_country(country_lastfm)

    result_text = f"Most Popular Tags for {country_lastfm}:\n"
    for tag, count in popular_tags.items():
        result_text += f"{tag}: {count}\n"

    result_label.value = result_text
```

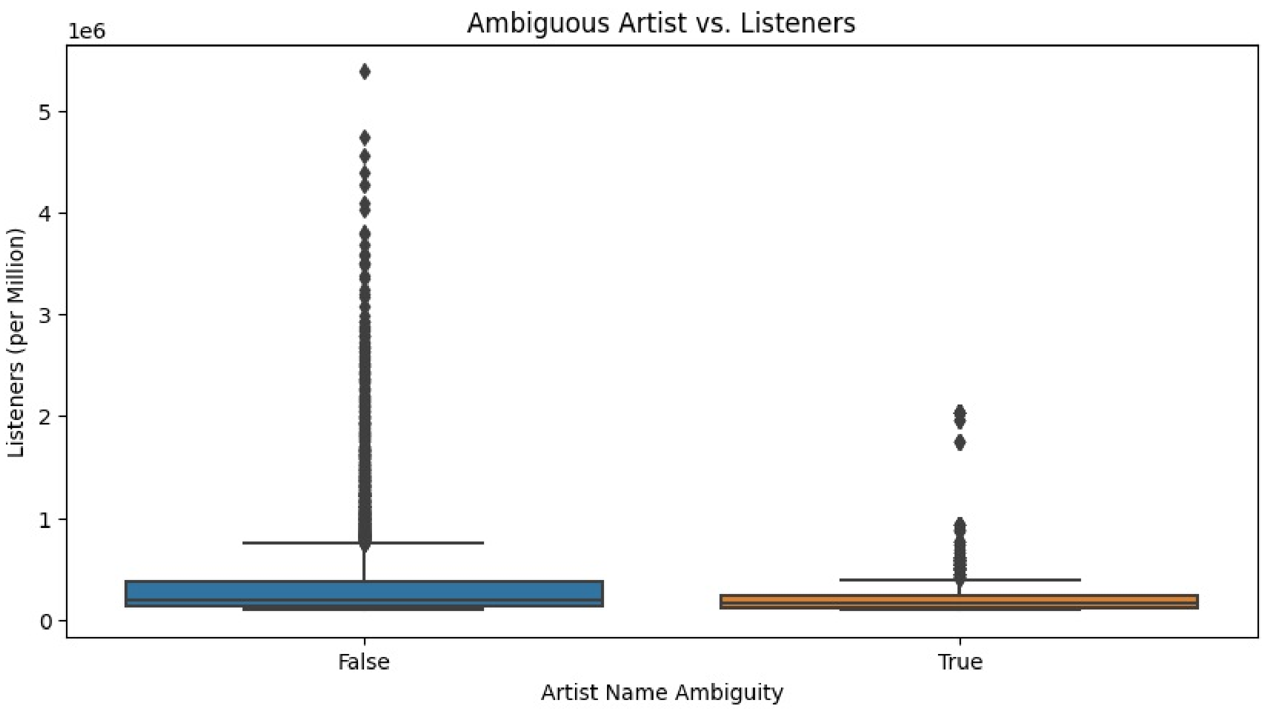
```
    # Create bar plot for Top 5 Tags for selected country
    fig, ax = plt.subplots(figsize=(8, 6))
    popular_tags.plot(kind='bar', ax=ax, color='blue')
    ax.set_title(f"Top 5 Tags for {country_lastfm}")
    ax.set_xlabel("Tags")
    ax.set_ylabel("Frequency")
    plt.xticks(rotation=45, ha="right")

plt.show()

# Create interactive dropdown widget to select country
country_dropdown = widgets.Dropdown(
    options=df['country_lastfm'].unique().tolist(),
    description='Country:',
    disabled=False
)

result_label = widgets.Label(value="Select a country to see the most popular tags.")

interactive_widget = interactive(on_country_selected, country_lastfm=country_dropdown)

display(interactive_widget, result_label)
```

```
interactive(children=(Dropdown(description='Country:', options=('Afghanistan', 'Albania', 'Algeria', 'Andorra'…
Label(value='Most Popular Tags for Afghanistan:\nqawwali: 1\n world: 1\n sufi: 1\n pakistani: 1\n World Music:…
```

Ambiguous Artists vs. Number of Monthly Listeners

Using a box plot we can visualize the difference in listeners when comparing if an artist has an ambiguous name or not.

In [11]:
```
#Set size and type of visualization, including what data will be used.
plt.figure(figsize=(10,5))
sns.boxplot(data=df, x='ambiguous_artist', y='listeners_lastfm')

#Provide a title and proper labels
plt.xlabel('Artist Name Ambiguity')
plt.ylabel('Listeners (per Million)')
plt.title('Ambiguous Artist vs. Listeners')

#Show visualization
plt.show()
```



Machine Learning Attempts

Unsupervised Learning

In [12]:
```
df = pd.read_csv('Mapd.csv', low_memory=False)
df = df[["listeners_lastfm", "ambiguous_artist"]]
df = df.dropna() #of the two coloumns, drop any N/A's
df
```

Out[12]:

	listeners_lastfm	ambiguous_artist
0	163604.0	False
1	678284.0	False
2	279962.0	False
3	375102.0	False
4	127056.0	False
...	...	...
7899	145081.0	False
7900	145081.0	False
7901	406492.0	False
7902	165955.0	False
7903	110502.0	False

7904 rows × 2 columns

In [13]:
```
features = ["listeners_lastfm", "ambiguous_artist"]
X = df[features] #defining the features

X['ambiguous_artist'] = X['ambiguous_artist'].apply(lambda x: 1 if x == "True" else 0) #if the ambiguous artist
X
```

Out[13]:

	listeners_lastfm	ambiguous_artist
0	163604.0	0
1	678284.0	0
2	279962.0	0
3	375102.0	0
4	127056.0	0
...	...	...
7899	145081.0	0
7900	145081.0	0
7901	406492.0	0
7902	165955.0	0
7903	110502.0	0

7904 rows × 2 columns

In [14]:
```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=3, random_state=0) #set the number of clusters to 3 and random state to zero, so we
kmeans
```

Out[14]:
```
▾ KMeans
KMeans(n_clusters=3, random_state=0)
```

In [15]:
```
kmeans.fit(X) #checking the fit of the model to the training data
```

Out[15]:
```
▾ KMeans
KMeans(n_clusters=3, random_state=0)
```

In [16]:
```
kmeans.predict(X) #having the model assign the data to 3 clusters
```

Out[16]:
```
array([0, 1, 0, ..., 0, 0, 0], dtype=int32)
```

In [17]:
```
df["label"] = kmeans.predict(X) #show the assigned clusters in the df rows
df
```

Out[17]:

	listeners_lastfm	ambiguous_artist	label
0	163604.0	False	0
1	678284.0	False	1
2	279962.0	False	0
3	375102.0	False	0
4	127056.0	False	0
...	...	...	...
7899	145081.0	False	0
7900	145081.0	False	0
7901	406492.0	False	0
7902	165955.0	False	0
7903	110502.0	False	0

7904 rows × 3 columns

In [18]:
```
df.label.value_counts() #sum the results of the clusters
```

Out[18]:
```
label
0    6724
1     970
2     210
Name: count, dtype: int64
```

In [19]:
```
cluster_1st, cluster_2nd, cluster_3rd = df.label.value_counts().index #naming all the clusters
cluster_1st, cluster_2nd, cluster_3rd
```

Out[19]:
```
(0, 1, 2)
```

In [20]:
```
df[df.label == cluster_1st].sample(n=10, random_state=0) #calling up 10 random samples from the df of the first
```

Out[20]:

	listeners_lastfm	ambiguous_artist	label
2849	238945.0	True	0
7795	141287.0	False	0
1832	108959.0	True	0
5238	389066.0	True	0
5624	100313.0	False	0
6536	111413.0	True	0
3412	136426.0	True	0
5323	378582.0	False	0
3648	253068.0	True	0
7258	199069.0	False	0

In [21]:
```
df[df.label == cluster_2nd].sample(n=10, random_state=0) #calling up 10 random samples from the df of the secon
```

Out[21]:

	listeners_lastfm	ambiguous_artist	label
7695	1470451.0	False	1
3252	825250.0	False	1
136	550489.0	False	1
6160	952286.0	False	1
5152	561265.0	False	1
6656	625526.0	False	1
6174	844887.0	False	1
7134	676501.0	False	1
4565	593270.0	False	1
5589	829178.0	False	1

In [22]:
```
df[df.label == cluster_3rd].sample(n=10, random_state=0) #calling up 10 random samples from the df of the third
```

Out[22]:

	listeners_lastfm	ambiguous_artist	label
992	1966716.0	True	2
4584	3203026.0	False	2
2175	2030349.0	True	2
209	1754155.0	True	2
7429	2709816.0	False	2
4645	1609221.0	False	2
5532	1924069.0	False	2
5623	1540288.0	False	2
4883	1671502.0	False	2
6223	2577949.0	False	2

Unsupervised Learning Conclusion: The model determined that, of the three clusters formed, first cluster (0) had a mix of ambiguous artists and lower listener numbers, the second cluster had no ambiguous artists and high amounts of listeners, and the third cluster had a low mix of ambiguous artists but high listener counts.

Supervised Learning

In [23]:
```
df = df.iloc[range(7904)]
df
```

Out[23]:

	listeners_lastfm	ambiguous_artist	label
0	163604.0	False	0
1	678284.0	False	1
2	279962.0	False	0
3	375102.0	False	0
4	127056.0	False	0
...	...	...	...
7899	145081.0	False	0
7900	145081.0	False	0
7901	406492.0	False	0
7902	165955.0	False	0
7903	110502.0	False	0

7904 rows × 3 columns

In [24]:
```
#For our first test we are going to run a linear regression.
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr
```

Out[24]:
```
▾ LogisticRegression
LogisticRegression()
```

In [25]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

#Convert True and False to binary code. After running into errors with the .apply function, we decided to use t
df['ambiguous_artist'] = df['ambiguous_artist'].astype(int)

#Set the features and targets to listeners and artist ambiguity to run our predictive models
features = ['listeners_lastfm']
target = ['ambiguous_artist']
X = df[features]
y = df[target]

#will discuss this further in the presentation
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

In [26]:
```
#Set test size, random state and then ultimately fit our regressions.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
```

```
lr = LogisticRegression()
lr.fit(X_train, y_train)
```

```
/home/akerestes/.local/lib/python3.11/site-packages/sklearn/utils/validation.py:1408: DataConversionWarning: A c
olumn-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for examp
le using ravel().
  y = column_or_1d(y, warn=True)
```

Out[26]:

```
▾ LogisticRegression
LogisticRegression()
```

In [27]:

```
#Run the linear regression scores against the training data and test data
lr.score(X_train, y_train), lr.score(X_test, y_test)
```

Out[27]:

```
(0.771255060728745, 0.7677125506072875)
```

In [28]:

```
#Create a dictionary containing the test scores of different models
summary = dict()

summary["LR"] = round(lr.score(X_test, y_test), 3)
summary
```

Out[28]:

```
{'LR': 0.768}
```

In [29]:

```
#Run the closest neighbors test.
from sklearn.neighbors import KNeighborsClassifier
#Fit the model
knc = KNeighborsClassifier(n_neighbors=2)
knc.fit(X_train, y_train)
#Run the scores against the data.
knc.score(X_train, y_train), knc.score(X_test, y_test)
```

```
/home/akerestes/.local/lib/python3.11/site-packages/sklearn/neighbors/_classification.py:239: DataConversionWarn
ing: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), fo
r example using ravel().
  return self._fit(X, y)
```

Out[29]:

```
(0.9875168690958165, 0.9089068825910931)
```

In [30]:

```
#Apply new scores to the dictionary
summary["K-NNs"] = round(knc.score(X_test, y_test), 3)
summary
```

Out[30]:

```
{'LR': 0.768, 'K-NNs': 0.909}
```

In [31]:

```
#Fit and run the Random Forest Classifier model
from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(random_state=10)
rfc.fit(X_train, y_train)
```

```
/home/akerestes/.local/lib/python3.11/site-packages/sklearn/base.py:1389: DataConversionWarning: A column-vector
y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel
().
  return fit_method(estimator, *args, **kwargs)
```

Out[31]:

```
▾ RandomForestClassifier
RandomForestClassifier(random_state=10)
```

In [32]:

```
#Add next score to the current dictionary
rfc.score(X_train, y_train), rfc.score(X_test, y_test)
summary["RF"] = round(rfc.score(X_test, y_test), 3)
```

In [33]:

```
#Fit and run the Linear SVC model
from sklearn.svm import LinearSVC

lsvc = LinearSVC(random_state=0)
lsvc.fit(X_train, y_train)
```

```
/home/akerestes/.local/lib/python3.11/site-packages/sklearn/utils/validation.py:1408: DataConversionWarning: A c
olumn-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for examp
le using ravel().
  y = column_or_1d(y, warn=True)
```

Out[33]:

```
▾ LinearSVC
LinearSVC(random_state=0)
```

In [34]:

```
#Add next score to the current dictionary
```

```
lsvc.score(X_train, y_train), lsvc.score(X_test, y_test)
summary["Linear SVCs"] = round(lsvc.score(X_test, y_test), 3)
```

In [35]:
```
#View all scores to determine which model will work the best. In this case, we went with the Random Forest Clas:
summary
```

Out[35]:
```
{'LR': 0.768, 'K-NNs': 0.909, 'RF': 0.939, 'Linear SVCs': 0.768}
```

In [36]:
```
#Create some random artists with varying listener counts
artist1 = {"listeners_lastfm": 190000,    # Artist with varying listeners
           }
artist2 = {"listeners_lastfm": 1400000,
           }

#Add an empty list
X_new = []

#Extract correct values to add to the empty list
for artist in [artist1, artist2]:
    new_artist = [artist["listeners_lastfm"]]
    X_new.append(new_artist)

#Create a data frame with the correct format
X_new = pd.DataFrame(data=X_new, columns=["listeners_lastfm"])
X_new = X_new[features]

#After realizing that there was an issue with the dataframe shape, I used the scaler function to properly forma
X_new_scaled = scaler.transform(X_new)
```

In [37]:
```
#Use the model to predict whether or not each artist will have a unique or ambiguous name or not
predict_rfc = rfc.predict(X_new)
predict_knn = knc.predict(X_new)
```

In [38]:
```
# The model predicts that both random artists will have non-ambiguous names
predict_rfc
```

Out[38]:
```
array([0, 0])
```

In [39]:
```
#Using the next highest scored model, we get the same result
predict_knn
```

Out[39]:
```
array([0, 0])
```

Conclusion

Our analysis aimed to explore the relationships between artist, country of origin, number of monthly listeners on LastFM, and user-generated tags to identify meaningful comparisons across these fields. We uncovered key insights into how these attributes interact through data cleaning, visualization, and statistical exploration.

Key Comparisons: Country vs. Popularity: Certain countries produce a higher concentration of globally recognized artists, while others have strong regional followings. The distribution of monthly listeners varies significantly across countries, highlighting the global reach of some artists compared to those with more localized appeal.

Genre (Tags) vs. Popularity: User-generated tags provided a diverse look at how artists are categorized. Some genres exhibit widespread popularity across multiple countries, while niche genres show strong regional or cultural significance.

Country vs. Genre Representation: Specific genres are more dominant in certain countries, reflecting cultural and industry trends. For example, some nations have a strong presence in electronic music, while others are well-known for rock or hip-hop.

Artist Popularity Distribution: A small percentage of artists command the majority of monthly listeners, while many others cater to more specialized audiences. This aligns with the "long-tail" distribution often seen in digital music consumption.

By comparing these fields, we better understood how artist popularity, country of origin, and genre are interwoven in global music trends. Future exploration could integrate additional data sources, such as streaming platform engagement, social media activity, or tour locations, to refine these comparisons further and provide a more comprehensive view of artist influence.

In [ ]: