BANDA SE CAMBIARON LAS VARIABLES EN LAS QUE TUVIMOS QUE LLENAR DATOS, EJ. DE POPULARITY SE CAMBIO A NEW_POPULARITY Y ASI PARA LAS OTRAS

Reto entregable 1

- Guillermo Villegas Morales A01637169
- Adara Luisa Pulido Sánchez A01642450
- Jorge Eduardo Guijarro Márquez A01563113
- Alan Rojas López A01706146
- Gabriel Eduardo Meléndez Zavala A01638293

Introducción

En este entregable realizamos una fase exploratoria de una base de datos de canciones donde cada entrada es una canción y sus atributos constan del nombre del artista/s, nombre de la canción, tonalidad, popularidad, duración entre otros. La base de datos tiene problemas como datos basura o datos faltantes, limpiearemos la base de datos para obtener un análisis propio, además de que realizamos diferentes estadísticas descriptivas.

Objetivos

Nuestro objetivo principal es tener una base de datos limpia y completa. Adicional a esto empezamos con el análisis exploratorio. Realizamos un heatmap para detectar covarianzas dentro los atributos, un boxplot por columna para tener una mejor idea de la distribución de los datos de cada columna y un wordcloud para encontrar las palabras más frecuentes en los nombres de artistas y de canciones

Métodos

Import Libraries

```
import numpy as np # lots of math operations and matrices
import pandas as pd # data structures
import matplotlib.pyplot as plt # plot charts. More on this later
from scipy import stats as st
import seaborn as sns
from wordcloud import WordCloud
import plotly.graph_objects as go

df=pd.read_csv("music.csv")


from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

Información básica de la base de datos

```
df.head()
```

	Artist Name	Track Name	Popularity	danceability	energy	key	loudness	mode	speechiness	acousticne
0	Bruno Mars	That's What I Like (feat. Gucci Mane)	60.0	0.854	0.564	1.0	-4.964	1	0.0485	0.0171
1	Boston	Hitch a Ride	54.0	0.382	0.814	3.0	-7.230	1	0.0406	0.0011
2	The Raincoats	No Side to Fall In	35.0	0.434	0.614	6.0	-8.334	1	0.0525	0.4860
3	Deno	Lingo (feat. J.I & Chunkz)	66.0	0.853	0.597	10.0	-6.528	0	0.0555	0.0212
4	Red Hot Chili Peppers	Nobody Weird Like Me - Remastered	53.0	0.167	0.975	2.0	-4.279	1	0.2160	0.0001

Buscamos los datos nulos dentro de la base de datos

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17996 entries, 0 to 17995
Data columns (total 17 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Artist Name         17996 non-null  object 
 1   Track Name          17996 non-null  object 
 2   Popularity          17568 non-null  float64
 3   danceability        17996 non-null  float64
 4   energy              17996 non-null  float64
 5   key                 15982 non-null  float64
 6   loudness            17996 non-null  float64
 7   mode                17996 non-null  int64  
 8   speechiness         17996 non-null  float64
 9   acousticness        17996 non-null  float64
 10  instrumentalness    13619 non-null  float64
 11  liveness            17996 non-null  float64
 12  valence             17996 non-null  float64
 13  tempo               17996 non-null  float64
 14  duration_in min/ms  17996 non-null  float64
 15  time_signature      17996 non-null  int64  
 16  Class               17996 non-null  int64  
dtypes: float64(12), int64(3), object(2)
memory usage: 2.3+ MB
```

Análisis rápido de cada columna

```
df.describe()
```

```
             Popularity  danceability        energy           key      loudness         mode  speechiness
```

Dimensión de la matríz

```
df.shape
```

```
(17996, 17)
```

```
min        1.000000       0.059600       0.000020       1.000000     -39.952000       0.000000       0.022500
```

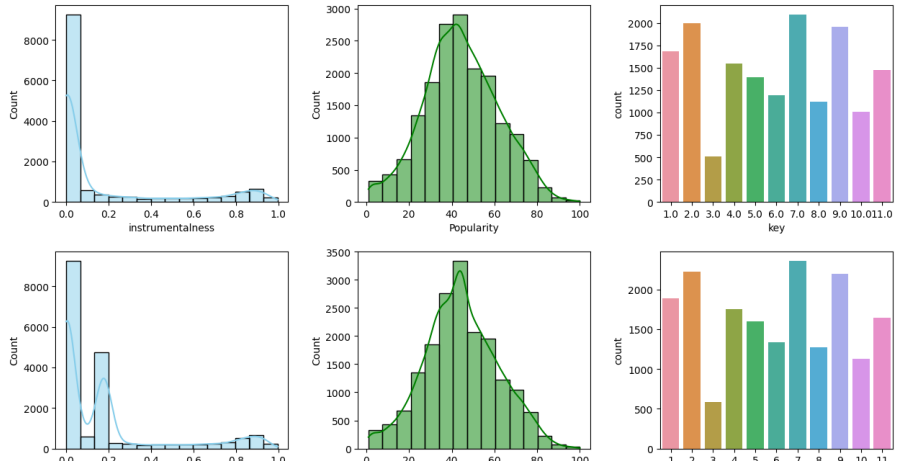
Imputación Simple

Se detectaron valores faltantes en las columnas de "instrumentalness", "key" y "Popularity". Utilizando media, moda y imputacion de k-vecinos más cercanos, se realizó una amputación simple de valores. Para los datos faltantes de 'popularity' introducimos el promedio de la columna ya que no cambia la distribucion. Para la columna de valores discretos 'key' introducimos el 0 donde faltaran valores ya que las columnas sin estos valores estaban en la tonalidad de C. Finalmente para la variable 'instrumentalness' imputamos con el valor anterior para que la distribucion no cambie significativamente. Adicionalmente, no se pueden borrar los datos ya que las variables incluyen un porecentaje significativo de datos, 0.0237, 0.1120, y 0.2432 respectivamente.

```
#Imprime el porcentaje de valores faltantes
print('Porcentaje de valores faltantes "Popularity": ', 1-(df['Popularity'].count()/17996))
print('Porcentaje de valores faltantes "Key": ', 1-(df['key'].count()/17996))
print('Porcentaje de valores faltantes "Instrumentalness": ', 1-(df['instrumentalness'].count()/17996))

#Creamos nuevas variables para mantener las originales
df['new_instrumentalness'] = df['instrumentalness']
df['new_instrumentalness'].ffill()
df['new_instrumentalness'].fillna(np.mean(df.instrumentalness), inplace = True) #Changes the last value

df['new_Popularity'] = df['Popularity']
df['new_Popularity'].fillna(np.mean(df.Popularity),inplace=True)

df['new_key'] = df['key'].ffill()
df['new_key'].fillna(np.mean(df.key), inplace = True) #Changes the last value
df['new_key'] = df['new_key'].astype(int)
```

```
Porcentaje de valores faltantes "Popularity":  0.023783062902867358
Porcentaje de valores faltantes "Key":  0.1119137586130251
Porcentaje de valores faltantes "Instrumentalness":  0.24322071571460324
```

```
fig, axs = plt.subplots(2,3, figsize=(15, 8))

bin_count = int(np.ceil(np.log2(len(df)))) #sturges law to figure out appropiate bin count

sns.histplot(data=df, x="instrumentalness", kde=True, color="skyblue", ax=axs[0, 0], bins = bin_count)
sns.histplot(data=df, x="new_instrumentalness", kde=True, color="skyblue", ax=axs[1, 0], bins = bin_count)

sns.histplot(data=df, x="Popularity", kde=True, color="green", ax=axs[0, 1], bins = bin_count)
sns.histplot(data=df, x="new_Popularity", kde=True, color="green", ax=axs[1, 1], bins = bin_count)

sns.countplot(data=df, x="key", ax=axs[0, 2])
sns.countplot(data=df, x="new_key", ax=axs[1, 2])

fig.subplots_adjust(wspace=0.3, hspace=0.25)
```



Clasificación de variables

- Artist name: categórica
- Track name: categórica
- Popularity: numérica
- danceability: numérica
- energy: numérica
- key: numérica
- loudness: numérica
- mode: numérica
- speechiness: numérica
- acousticness: numérica
- instrumentalness: numérica
- liveness: numérica
- valence: numérica
- tempo: numérica
- duration_in min/ms: numérica
- time_signature:numérica
- Class: categórica

```
df.info() #Show the changes that were made
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17996 entries, 0 to 17995
Data columns (total 20 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Artist Name           17996 non-null  object 
 1   Track Name            17996 non-null  object 
 2   Popularity            17568 non-null  float64
 3   danceability          17996 non-null  float64
 4   energy                17996 non-null  float64
 5   key                   15982 non-null  float64
 6   loudness              17996 non-null  float64
 7   mode                  17996 non-null  int64  
 8   speechiness           17996 non-null  float64
 9   acousticness          17996 non-null  float64
 10  instrumentalness      13619 non-null  float64
 11  liveness              17996 non-null  float64
 12  valence               17996 non-null  float64
 13  tempo                 17996 non-null  float64
 14  duration_in min/ms    17996 non-null  float64
 15  time_signature        17996 non-null  int64  
 16  Class                 17996 non-null  int64  
 17  new_instrumentalness  17996 non-null  float64
 18  new_Popularity        17996 non-null  float64
```

```
 19  new_key              17996 non-null  int64
dtypes: float64(14), int64(4), object(2)
memory usage: 2.7+ MB
```

Como podemos ver, todas las columnas tienen 17996 datos no nulos

Creando nueva clase de Género

Al estar codificada la variable "Class" en números del 1 al 10 es necesario interpretar los números con respecto a cada uno de los géneros musicales. En Base a la tabla proporcionada se crea una nueva variable llamada "Genre" que representa explicítamente el género al que pertenece cada canción.

```
#Create a function that relates the numerical values of class to its corresponding genre
def class_to_genre(row):
    if row == 0:
        return 'Acoustic/Folk'
    elif row == 1:
        return 'Alternative'
    elif row == 2:
       return 'Blues'
    elif row == 3:
       return 'Bollywood'
    elif row == 4:
       return 'Country'
    elif row == 5:
        return 'Hip-Hop'
    elif row == 6:
       return 'Indie'
    elif row == 7:
        return 'Instrumental'
    elif row == 8:
        return 'Metal'
    elif row == 9:
        return 'Pop'
    elif row == 10:
        return 'Rock'

df['Genre'] = df['Class'].apply(class_to_genre)
df.head()
```

	Artist Name	Track Name	Popularity	danceability	energy	key	loudness	mode	speechiness	acousticne
0	Bruno Mars	That's What I Like (feat. Gucci Mane)	60.0	0.854	0.564	1.0	-4.964	1	0.0485	0.0171
1	Boston	Hitch a Ride	54.0	0.382	0.814	3.0	-7.230	1	0.0406	0.0011
2	The Raincoats	No Side to Fall In	35.0	0.434	0.614	6.0	-8.334	1	0.0525	0.4860
3	Deno	Lingo (feat. J.I & Chunkz)	66.0	0.853	0.597	10.0	-6.528	0	0.0555	0.0212
4	Red Hot Chili Peppers	Nobody Weird Like Me - Remastered	53.0	0.167	0.975	2.0	-4.279	1	0.2160	0.0001

5 rows × 21 columns

New class Key

Similar al proceso anterior, decodificamos el atributo 'key' de los registros donde el 0.0 recibe el la calsificación de C, 1.0 de C#, ... y 11.0 de B. dentro de una nueva variable categórica 'Key'.

```
def class_to_Key(row):
    if row == 0.0:
        return 'C'
    elif row == 1.0:
```

```
        return 'C#'
    elif row == 2.0:
       return 'D'
    elif row == 3.0:
       return 'D#'
    elif row == 4.0:
       return 'E'
    elif row == 5.0:
        return 'F'
    elif row == 6.0:
       return 'F#'
    elif row == 7.0:
        return 'G'
    elif row == 8.0:
        return 'G#'
    elif row == 9.0:
        return 'A'
    elif row == 10.0:
        return 'A#'
    elif row == 11.0:
        return 'B'

df['Key'] = df['key'].apply(class_to_Key)
df.head()
```

	Artist Name	Track Name	Popularity	danceability	energy	key	loudness	mode	speechiness	acousticne
0	Bruno Mars	That's What I Like (feat. Gucci Mane)	60.0	0.854	0.564	1.0	-4.964	1	0.0485	0.0171
1	Boston	Hitch a Ride	54.0	0.382	0.814	3.0	-7.230	1	0.0406	0.0011
2	The Raincoats	No Side to Fall In	35.0	0.434	0.614	6.0	-8.334	1	0.0525	0.4860
3	Deno	Lingo (feat. J.I & Chunkz)	66.0	0.853	0.597	10.0	-6.528	0	0.0555	0.0212
4	Red Hot Chili Peppers	Nobody Weird Like Me - Remastered	53.0	0.167	0.975	2.0	-4.279	1	0.2160	0.0001

5 rows × 22 columns

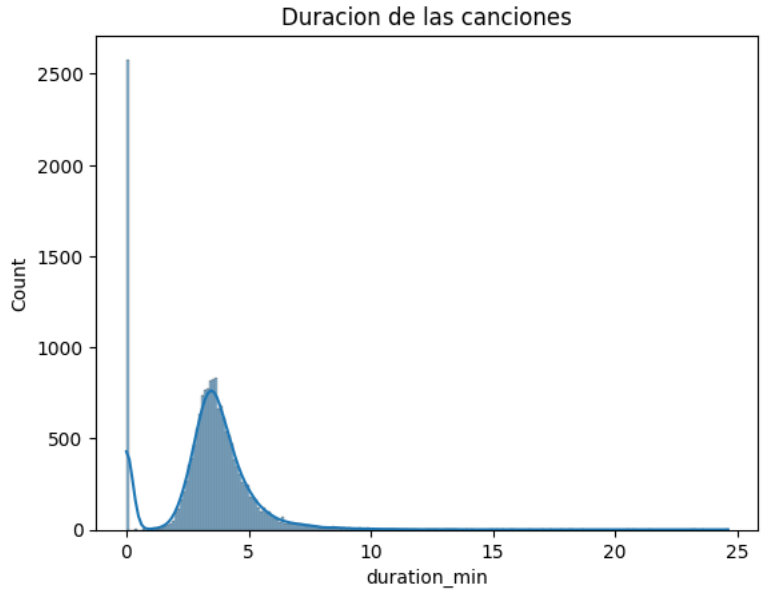
Histogramas

```
sns.histplot(data=df,x="duration_in min/ms")
```

```
<Axes: xlabel='duration_in min/ms', ylabel='Count'>
```

```
df['duration_min'] = df['duration_in min/ms']/60000
sns.histplot(data=df,x="duration_min", kde = True).set_title('Duracion de las canciones')
```

```
Text(0.5, 1.0, 'Duracion de las canciones')
```



Con el fin de observar la distribución de frecuencias del tiempo en cada una de las canciones se genera un histograma de la variable "duration in min/ms". Sin embargo, al revisar la gráfica se observa una gran cantidad de datos en la duración 3 min, al corroborar con la base de datos se encontró que algunas canciones estaban en minutos mientras que otras estaban escritas como milisegundos. Por lo tanto se comprueba la medida de la duración en cada una de las canciones, aquellas con valores menores a 100 se multiplican por 60000 para convertirlos a minutos. Una vez se tiene todos los datos de la duración en minutos se vuelve a generar un histograma de la misma variable con los datos correctamente medidos.

```
df['duration_min'].describe()
```

```
count    17996.000000
mean         3.345741
std          1.866485
min          0.000008
25%          2.772283
50%          3.486000
75%          4.208167
max         24.619783
Name: duration_min, dtype: float64
```

Se observa que al generar esta nueva columna de datos que describe la duracion en minutos de las canciones resalta que la maxima de los datos es de 29.86 mientras la media es de 3.93 con una std de 1.43. Por lo tanto seria importante hacer un analisis para determinar si todos los datos son relevantes para el analisis ya que existen valores extraordinarios

Nueva variable "collab"

Aquí creamos una nueva variable booleana 'collab' donde 1 significa que la canción es una colaboración entre artistas y 0 es que no lo es. Para hacer esto definimos que hay dos posible casos que indiquen esto: cuando la canción tiene una ',' en el atributo 'Artist Name' o cuando contiene la palabra 'feat.' dentro de 'Track Name'. En total encontramos 1202 canciones con colaboración.

```
df['collab'] = df['Artist Name'].str.contains(',') + df['Track Name'].str.contains('feat.')
df['collab']
```

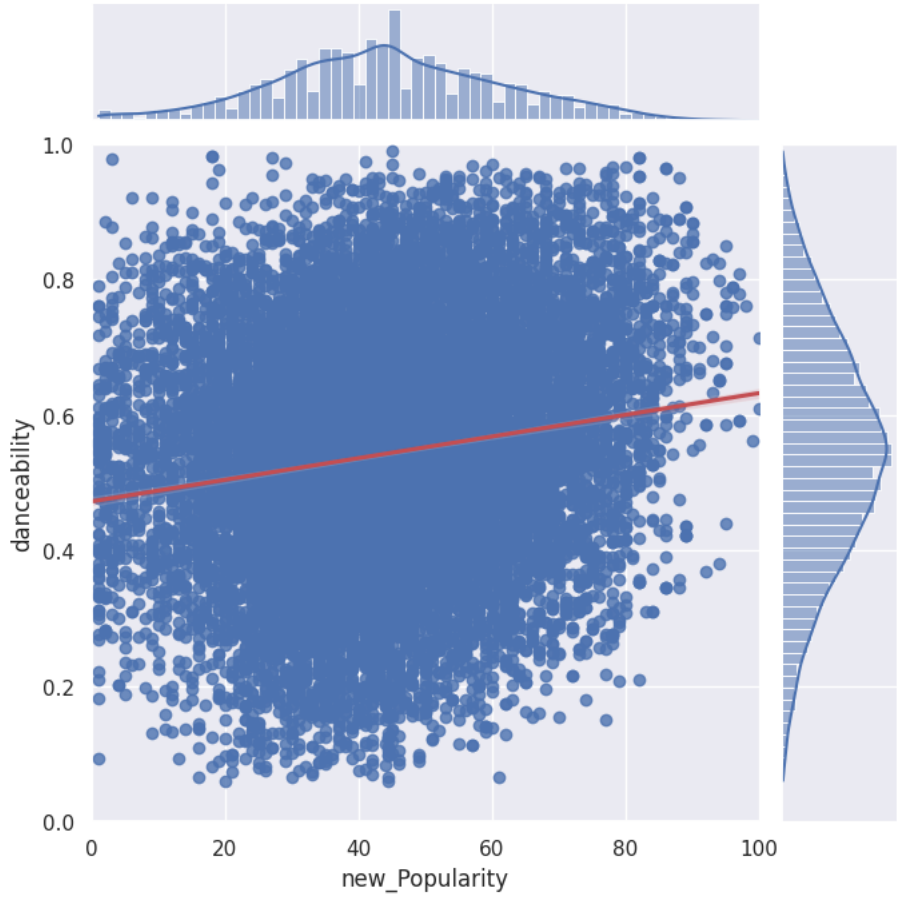
```
0         True
1        False
2        False
3         True
4        False
         ...
17991    False
17992    False
```

```
17993    False
17994    False
17995    False
Name: collab, Length: 17996, dtype: bool
```

Fase 2

Exploración de los datos y Análisis descriptivo

```
sns.set_theme(style="darkgrid")
g = sns.jointplot(x="new_Popularity", y="danceability", data=df,
                  kind="reg", truncate=False,
                  xlim=(0, 100), ylim=(0, 1),
                  color="b", height=7, joint_kws={'line_kws': {'color':'r'}})
```



^^^^^^^^^^AQUI VA TEXTO EXPLICANDO QUE SE NOTA^^^^^^^^^^

```
df2=df[['new_Popularity','danceability','energy','loudness','speechiness','acousticness','new_instrumentalness','liveness','valence','tempo',
df2.head()
```

	new_Popularity	danceability	energy	loudness	speechiness	acousticness	new_instrumentalness	liv
0	60.0	0.854	0.564	-4.964	0.0485	0.017100	0.177562	(
1	54.0	0.382	0.814	-7.230	0.0406	0.001100	0.004010	(
2	35.0	0.434	0.614	-8.334	0.0525	0.486000	0.000196	(
3	66.0	0.853	0.597	-6.528	0.0555	0.021200	0.177562	(
4	53.0	0.167	0.975	-4.279	0.2160	0.000169	0.016100	(

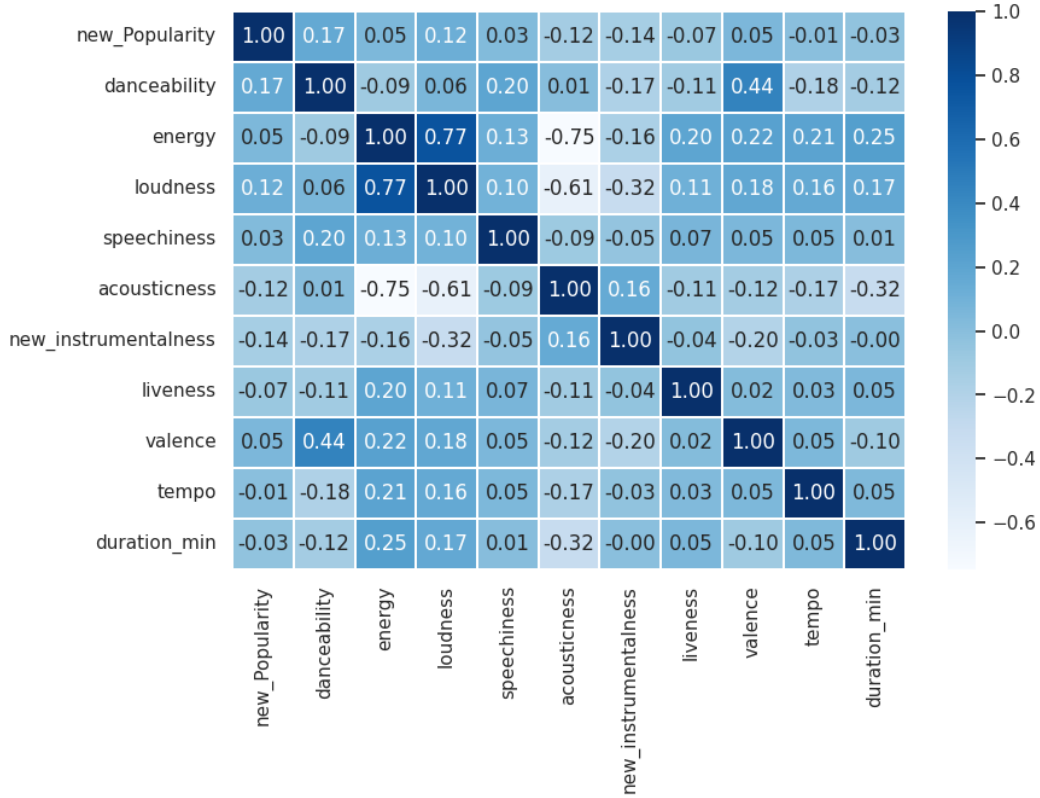
Estos seran las variables cuantitativas que se usaran para el analisis de estadistica descriptiva

Correlation Heatmap

En cuanto a la visualización de la correlación que tienen las variables de "Popularity", "danceability", "energy", "loudness", "speechiness", "acousticness", "instrumentalness", "liveness", "valence", "tempo" y "duration_in min" se produce un mapa de calor de correlación. Se analizó que como resultado las variables que tienen mayor correlación con la variable "Popularity" son "danceability" y "loudness".

```
fig, ax = plt.subplots(figsize = (9, 6))
sns.heatmap(data = df2.corr(), cmap ='Blues', linewidths = 0.30, annot =True,fmt='.2f')
```

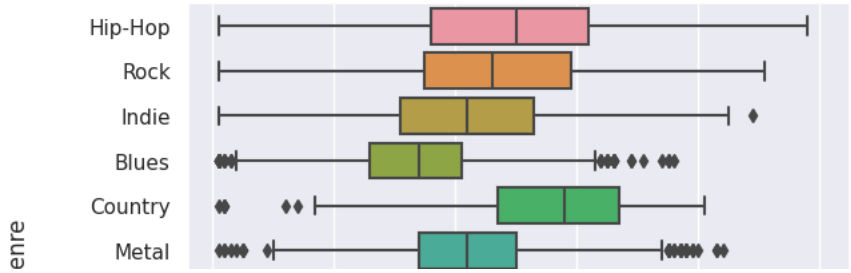
```
<Axes: >
```



Boxplot

Con el objetivo de ver la relación que tienen el género de las canciones con su popularidad se genera un boxplot. En el eje horizontal de la visualización se representa la popularidad de las canciones, mientras que en el eje vertical se observan cada una de las categorías de género. Esta gráfica presenta información acerca del rango intercuartil, la mediana, la cual indica la variabilidad en la popularidad dentro de cada género, así como los valores atípicos. Al examinar los datos proporcionados por el boxplot se resalta que el género "Country" tiende a ser más popular, pues presenta una mediana más alta que el resto, mientras que "Indie" y "Alternative" tienen canciones excepcionalmente populares. Por otro lado, el género con menor popularidad es el de "Bollywood" con una mediana menor.

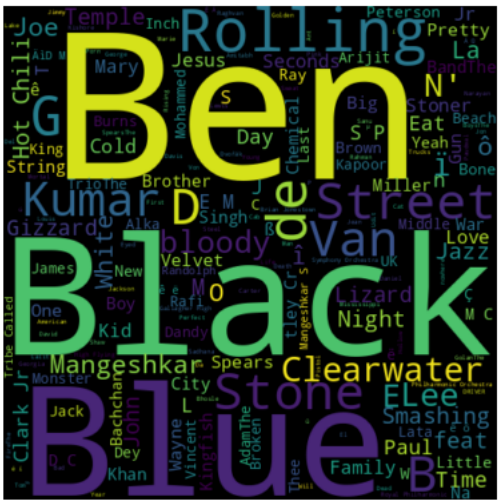
```
sns.boxplot(x=df['new_Popularity'],y=df['Genre']) #Distribución de popularidad por Género musical
plt.show()
```



Worldcloud de artistas

Generamos un Wordcloud de artistas para visualizar las palabras más recurridas.

```
# Create the wordcloud object
artist_array = ''.join(df['Artist Name'])
wordcloud = WordCloud(width=480, height=480, margin=0).generate(artist_array)

# Display the generated image:
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.margins(x=0, y=0)
plt.show()
#sns.violinplot(x=df["species"], y=df["sepal_length"])
```



Palabras más comunes en los nombres de artistas, resalta "Ben", "Blue", "Black", "Rolling", entre otros.

Worldcloud de nombres de canciones

De misma manera se genero un wordcloud para los nombres de canciones y buscamos por patrones o relaciones.

```
# Create the wordcloud object
track_array = ''.join(df['Track Name'])
wordcloud = WordCloud(width=480, height=480, margin=0).generate(track_array)

# Display the generated image:
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.margins(x=0, y=0)
plt.show()
```

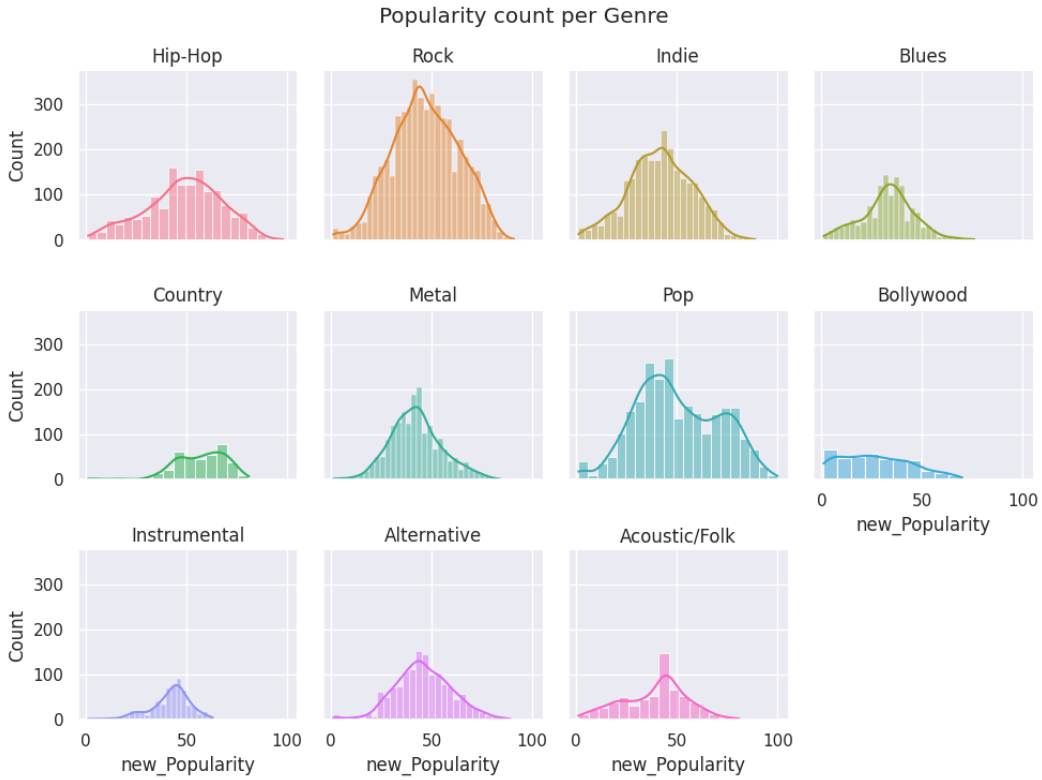
Palabras más comunes en nombres de canciones, resaltan feat, Love ,remastered y live. A su vez, ecnontramos caracteres inusuales que corresponden en buena parte a los datos basura dentro de la columna

Comparativo entre "Popularity"y "Genre"

Se realizaron histogramas de la popularidad de cada genero con fin de visualizar la popularidad de cada genero de musica.

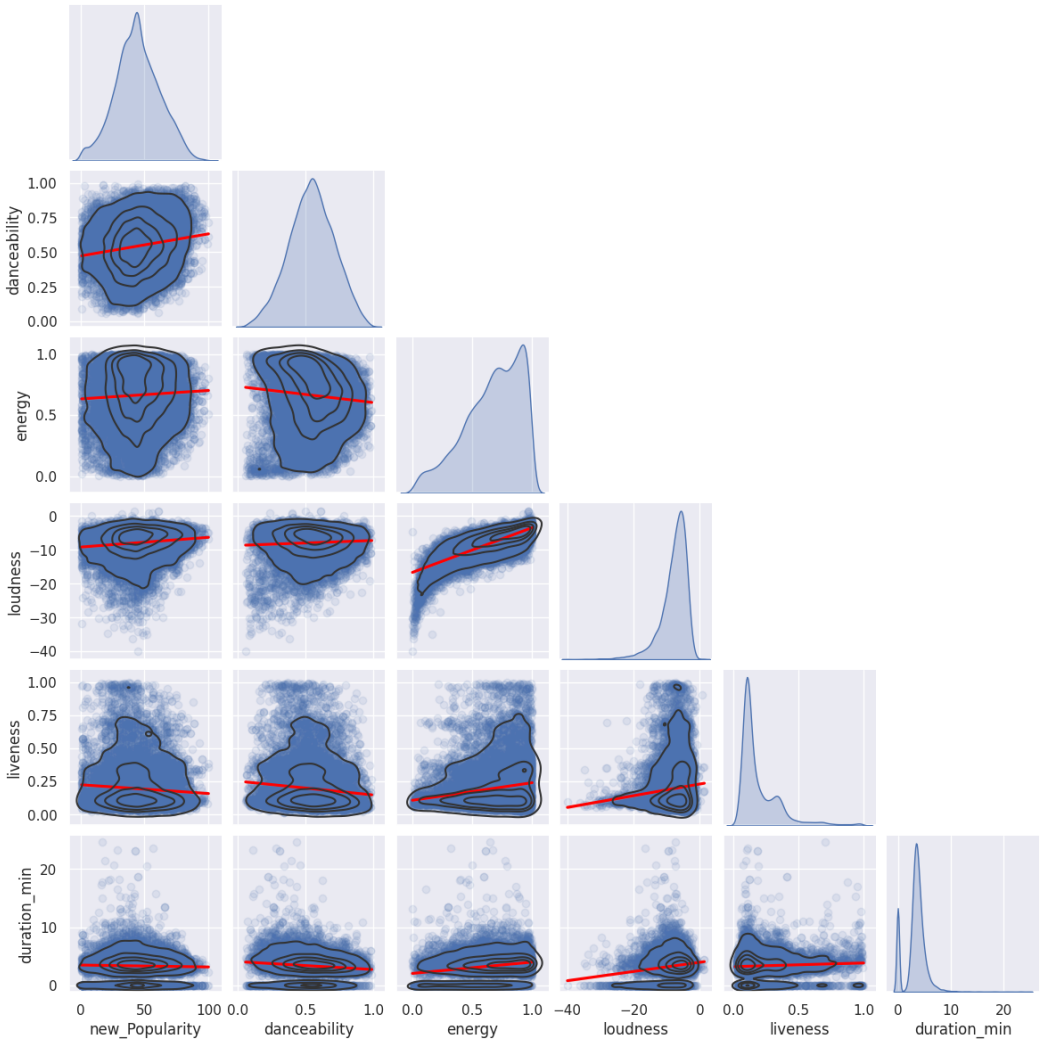
```
g = sns.FacetGrid(df, col='Genre',hue = 'Genre', col_wrap=4, height=2.5)
g.map(sns.histplot,'new_Popularity', kde = True)
g.set_titles("{col_name}")
g.fig.subplots_adjust(top=0.9)
g.fig.suptitle('Popularity count per Genre')

#fig, axs = plt.subplots(4, 3, figsize=(7, 7))
#axs = sns.FacetGrid(data = df, col = 'Genre',hue = 'Genre')
#axs.map(sns.histplot, 'Popularity', kde = True)
```

```
Text(0.5, 0.98, 'Popularity count per Genre')
```

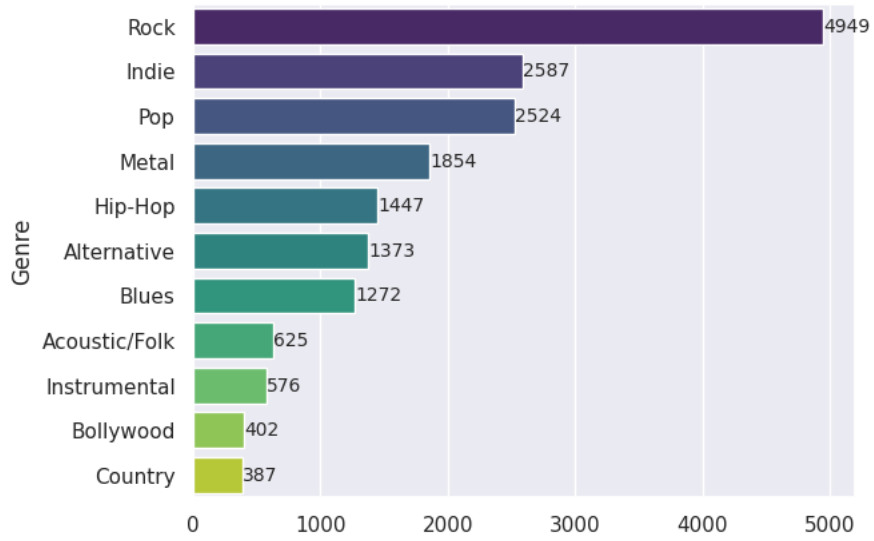


Estadistica basica de las variables relevantes

```
#Pairplot with the most relevant or significant variables
df3=df[['new_Popularity','danceability','energy','loudness','liveness','duration_min']]
g = sns.pairplot(df3, kind="reg", diag_kind = 'kde',height=2,corner = True, plot_kws={'line_kws':{'color':'red'}, 'scatter_kws': {'alpha': 0.
#makes the lower half have a sort of heat map density
g.map_lower(sns.kdeplot, color=".2",levels=5)
plt.show()
```




```
ax = sns.countplot(y = 'Genre', data = df, palette = 'viridis', order = df['Genre'].value_counts().index)
for bars in ax.containers:
  ax.bar_label(bars,size= 10)
```



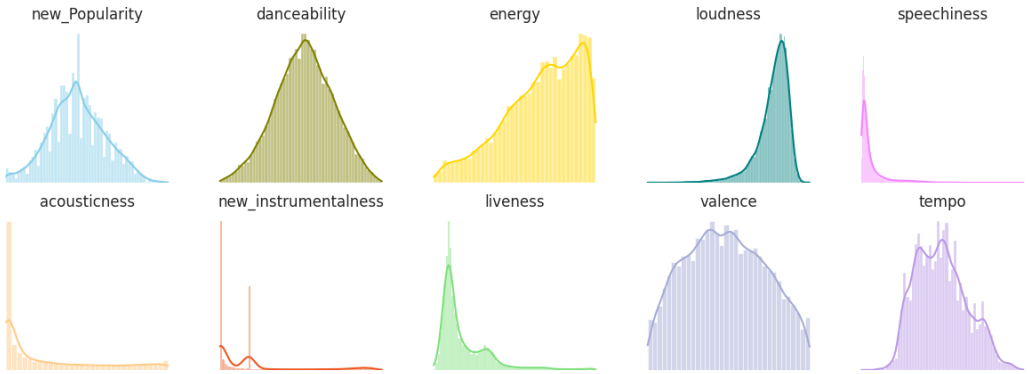

```
#Estadística Básica
columnas_numericas = df.select_dtypes(include=[int, float])

#Coeficiente de asimetria
coeficiente_asimetria_dict = {}

for i in columnas_numericas.columns:
    coeficiente = columnas_numericas[i].skew()
    coeficiente_asimetria_dict[i] = coeficiente

print(f"coeficiente de asimetría: {coeficiente_asimetria_dict}")

#Coeficiente de variación (%)
coeficiente_variacion_dict = {}

for i in columnas_numericas.columns:
    media = columnas_numericas[i].mean()
    desviacion_estandar = columnas_numericas[i].std()
    coeficiente_variacion = (desviacion_estandar / media) * 100
    coeficiente_variacion_dict[i] = coeficiente_variacion

print(f"coeficiente de variación: {coeficiente_variacion_dict}")
```

```
coeficiente de asimetría: {'Popularity': 0.07570521616392309, 'danceability': -0.08352192347287282, 'energy': -0.6611691117532402, 'key
coeficiente de variación: {'Popularity': 39.15096836380726, 'danceability': 30.595935969574263, 'energy': 35.51322160007704, 'key': 53.7
```

Transformación de Box-Cox

```
fig, axs = plt.subplots(2,5, figsize=(15, 5))

sns.histplot(data=df, x="new_Popularity", kde=True, color="skyblue", ax=axs[0, 0]).set_title('new_Popularity')
sns.histplot(data=df, x="danceability", kde=True, color="olive", ax=axs[0, 1]).set_title('danceability')
sns.histplot(data=df, x="energy", kde=True, color="gold", ax=axs[0, 2]).set_title('energy')
sns.histplot(data=df, x="loudness", kde=True, color="teal", ax=axs[0, 3]).set_title('loudness')
sns.histplot(data=df, x="speechiness", kde=True, color="#F387FE", ax=axs[0, 4]).set_title('speechiness')
sns.histplot(data=df, x="acousticness", kde=True, color="#FECB87", ax=axs[1, 0]).set_title('acousticness')
sns.histplot(data=df, x="new_instrumentalness", kde=True, color="#EE5824", ax=axs[1, 1]).set_title('new_instrumentalness')
sns.histplot(data=df, x="liveness", kde=True, color="#7FE07C", ax=axs[1, 2]).set_title('liveness')
sns.histplot(data=df, x="valence", kde=True, color="#A6ADD6", ax=axs[1, 3]).set_title('valence')
sns.histplot(data=df, x="tempo", kde=True, color="#BB99E7", ax=axs[1, 4]).set_title('tempo')

for ax in axs.flat:
    ax.label_outer()

axs = axs.flatten()
for ax in axs:
    ax.set_axis_off()
    ax.set_xticks([])
    ax.set_yticks([])

plt.xticks(visible=False)
```

```
plt.show()
```




```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
data_popularity, lambda_popularity = st.boxcox(df['new_Popularity'])
```

test de Grubbs

```
%%%%%%%%%%%%%
```

```
df.describe()
```

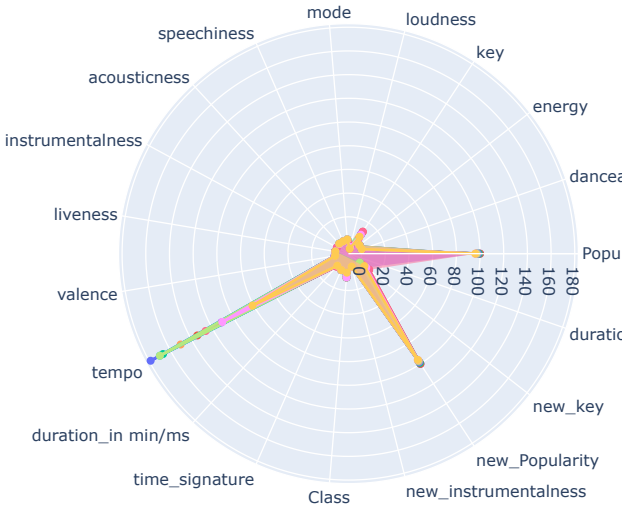
	Popularity	danceability	energy	key	loudness	mode	speechiness
count	17568.000000	17996.000000	17996.000000	15982.000000	17996.000000	17996.000000	17996.000000
mean	44.512124	0.543433	0.662777	5.952447	-7.910660	0.636753	0.079707
std	17.426928	0.166268	0.235373	3.196854	4.049151	0.480949	0.083576
min	1.000000	0.059600	0.000020	1.000000	-39.952000	0.000000	0.022500
25%	33.000000	0.432000	0.509000	3.000000	-9.538000	0.000000	0.034800
50%	44.000000	0.545000	0.700000	6.000000	-7.016000	1.000000	0.047400
75%	56.000000	0.659000	0.860000	9.000000	-5.189000	1.000000	0.083000
max	100.000000	0.989000	1.000000	11.000000	1.355000	1.000000	0.955000

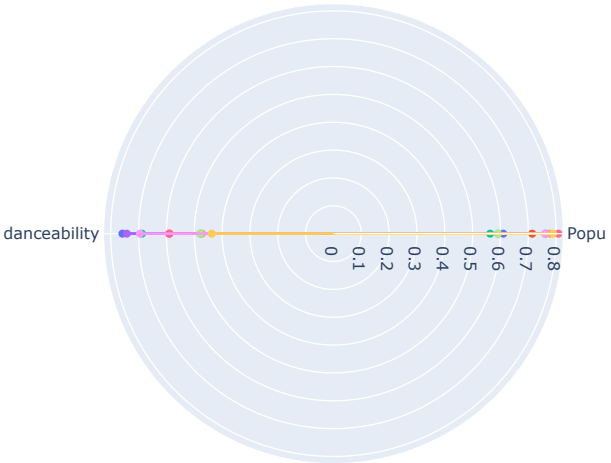
Radar Chart of Top 10 popular songs

```
# Transforming 'duration_in min/ms' to min:
df.loc[df['duration_in min/ms']%1 == 0.0, 'duration_in min/ms']= df['duration_in min/ms']/60000

# parameters we will be using
top10 = pd.DataFrame(df.nlargest(10,'Popularity'))
```

```
top_10_song_names = [] # Create array for song names in Radar Chart
for name in (pd.merge(top10,df)['Track Name']):
  top_10_song_names.append(name)

Radar_Chart1 = go.Figure() #Creating Chart1
```

```
for i in range(0,10):
  Radar_Chart1.add_trace(go.Scatterpolar(
        r= top10.iloc[i,2:],
        theta=columnas_numericas.columns,
        fill='toself',
        name = top_10_song_names[i]
  ))
Radar_Chart1.show()

Radar_Chart2 = go.Figure() #Creating Chart2
for i in range(0,10):
  Radar_Chart2.add_trace(go.Scatterpolar(
        r= top10.iloc[i,[3,12]],
        theta=columnas_numericas.columns,
        fill='toself',
        name = top_10_song_names[i]
  ))
Radar_Chart2.show()
```





```
df.head(10)
```

	Artist Name	Track Name	Popularity	danceability	energy	key	loudness	mode	speechiness	acoustic
0	Bruno Mars	That's What I Like (feat. Gucci Mane)	60.0	0.854	0.564	1.0	-4.964	1	0.0485	0.01
1	Boston	Hitch a Ride	54.0	0.382	0.814	3.0	-7.230	1	0.0406	0.00
2	The Raincoats	No Side to Fall In	35.0	0.434	0.614	6.0	-8.334	1	0.0525	0.48
3	Deno	Lingo (feat. J.I & Chunkz)	66.0	0.853	0.597	10.0	-6.528	0	0.0555	0.02
4	Red Hot Chili Peppers	Nobody Weird Like Me - Remastered	53.0	0.167	0.975	2.0	-4.279	1	0.2160	0.00
5	The Stooges	Search and Destroy - Iggy Pop Mix	53.0	0.235	0.977	6.0	0.878	1	0.1070	0.00
6	Solomon Burke	None Of Us Are Free	48.0	0.674	0.658	5.0	-9.647	0	0.1040	0.40
7	Randy Travis	On the Other Hand	55.0	0.657	0.415	5.0	-9.915	1	0.0250	0.17
8	Professional Murder Music	Slow	29.0	0.431	0.776	10.0	-5.403	1	0.0527	0.00
9	Dudu Aharon	◊ì◊ï◊ì◊ï, ◊ô◊ê◊ú◊ú◊î ◊ô◊ê◊ú◊ú◊î	14.0	0.716	0.885	1.0	-4.348	0	0.0333	0.06

10 rows × 24 columns