CLASE DE LOS MIERCOLES
10/21/24

Unidad 2: Procesamiento y Limpieza de Datos

LIMPIEZA DE UNA BASE DE DATOS ENSUCIADA

Rojas Huerta
Rey David

ANÁLISIS INICIAL

Uso del .describe(), .info(), gráficos de pastel, preliminares de los datos de tipo categóricos importantes para el análisis, uso de la función .isnull()

Primeras lineas de la base sin limpiar, notando a primera vista valores NaN.

```
url=('df_sucio .csv')
df=pd.read_csv(url)
df.head(10)
```

✓ 0.2s

	instance_id	artist_name	track_name	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	key	liveness
0	32894.0	Röyksopp	Röyksopp's Night Out	27.0	0.00468	0.652	-1.0	0.941	0.792	A#	0.115
1	46652.0	Thievery Corporation	The Shining Path	31.0	0.0127	0.622	218293.0	0.89	0.95	NaN	0.124
2	30097.0	Dillon Francis	Hurricane	28.0	0.00306	0.62	215613.0	0.755	0.0118	G#	0.534
3	62177.0	Dubloadz	Nitro	34.0	0.0254	0.774	166875.0	0.7	0.00253	C#	0.157
4	24907.0	What So Not	Divide & Conquer	32.0	0.00465	0.638	222369.0	0.5870000000000001	0.909	F#	0.157
5	89064.0	Axel Boman	Hello	47.0	0.0052299999999999	0.755	519468.0	0.731	0.8540000000000001	D	0.216
6	43760.0	Jordan Comolli	NaN	46.0	0.0289	0.5720000000000001	214408.0	0.8029999999999999	7.74e-06	B	0.106
7	30738.0	Hraach	Delirio	43.0	0.0297	0.809	416132.0	0.706	0.903	G	0.0635
8	84950.0	Kayzo	NEVER ALONE	39.0	0.00299	0.509	292800.0	NaN	0.000276	F	0.178
9	56950.0	Shlump	Lazer Beam	22.0	0.00934	0.578	204800.0	0.731	0.0112	A	0.111

Haciendo uso del .describe() y .info() notamos que, a excepción de una columna de datos en formato float (duration_ms) todas las demás están en tipo object.

```
df.describe()
```
✓ 0.0s

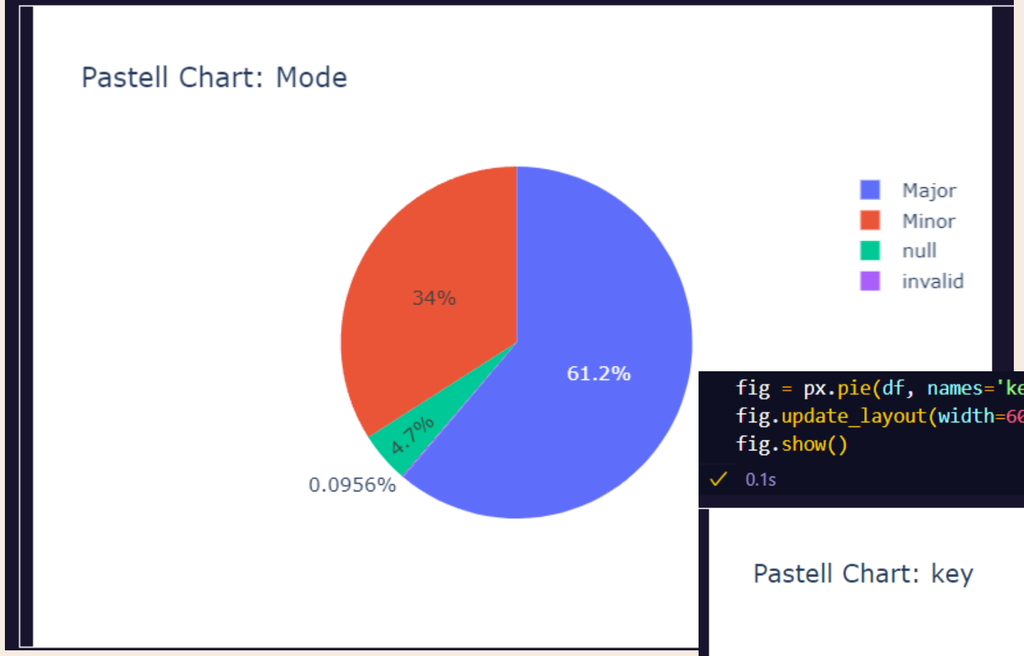
	duration_ms
count	5.077900e+04
mean	2.214766e+05
std	1.280828e+05
min	-1.000000e+00
25%	1.750000e+05
50%	2.194430e+05
75%	2.688270e+05
max	4.830606e+06

```
df.info()
```
✓ 0.0s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 53350 entries, 0 to 53349
Data columns (total 18 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   instance_id       50845 non-null  object
 1   artist_name       50845 non-null  object
 2   track_name        50845 non-null  object
 3   popularity        50846 non-null  object
 4   acousticness      50845 non-null  object
 5   danceability      50846 non-null  object
 6   duration_ms       50779 non-null  float64
 7   energy            50845 non-null  object
 8   instrumentalness  50845 non-null  object
 9   key               50845 non-null  object
 10  liveness          50845 non-null  object
 11  loudness          50845 non-null  object
 12  mode              50845 non-null  object
 13  speechiness       50845 non-null  object
 14  tempo             50845 non-null  object
 15  obtained_date     50845 non-null  object
 16  valence           50845 non-null  object
 17  music_genre       50845 non-null  object
dtypes: float64(1), object(17)
memory usage: 7.3+ MB
```

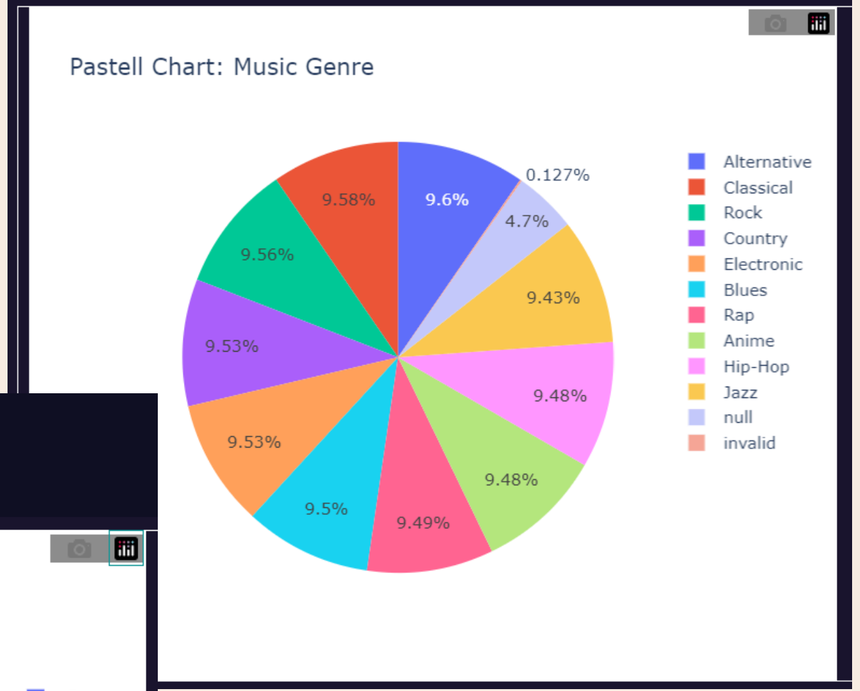
```
fig = px.pie(df, names='mode', title='Pastell Chart: Mode')
fig.update_layout(width=600,height=400)
fig.show()
```

✓ 0.1s



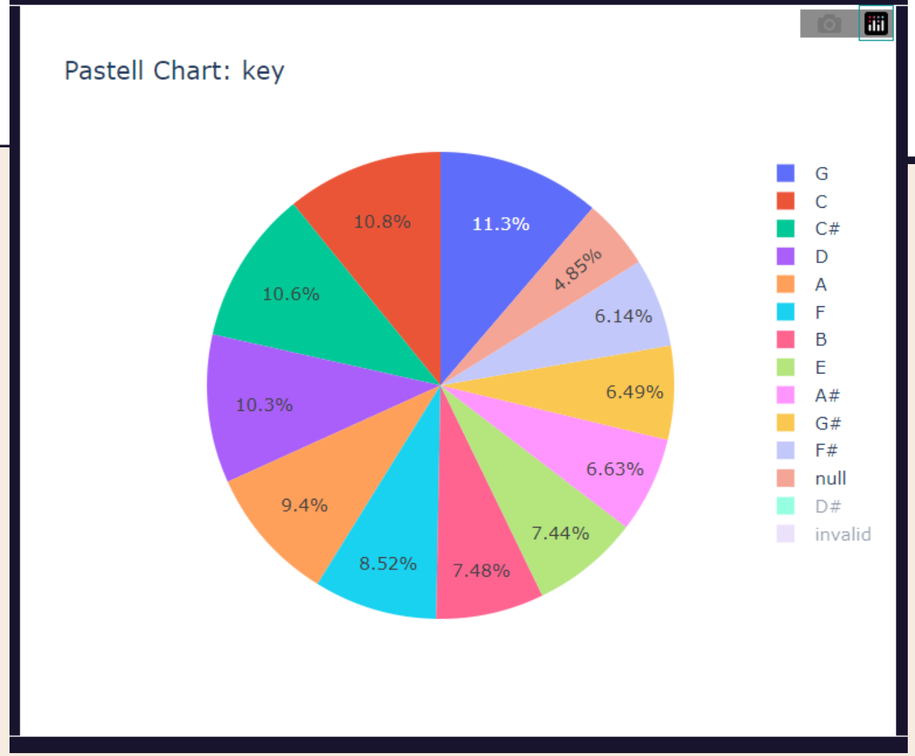

```
fig = px.pie(df, names='music_genre', title='Pastell Chart: Music Genre')
fig.update_layout(width=600,height=500)
fig.show()
```

✓ 0.1s




```
fig = px.pie(df, names='key', title='Pastell Chart: key')
fig.update_layout(width=600,height=500)
fig.show()
```

✓ 0.1s



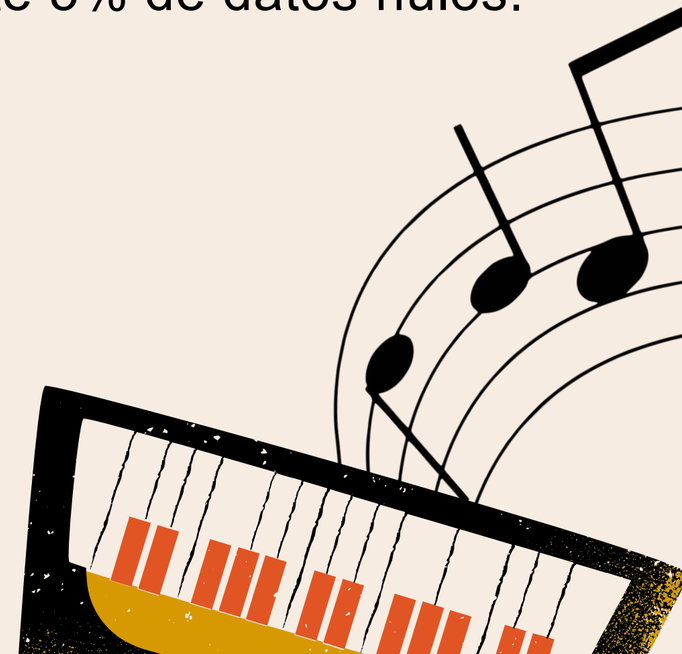
```
df.isnull().mean()*100
```

✓ 0.0s

instance_id	4.695408
artist_name	4.695408
track_name	4.695408
popularity	4.693533
acousticness	4.695408
danceability	4.693533
duration_ms	4.819119
energy	4.695408
instrumentalness	4.695408
key	4.695408
liveness	4.695408
loudness	4.695408
mode	4.695408
speechiness	4.695408
tempo	4.695408
obtained_date	4.695408
valence	4.695408
music_genre	4.695408

dtype: float64

Usando el siguiente código, y basándonos en las gráficas anteriores, notamos que la mayoría de las columnas cuentan con aproximadamente 5% de datos nulos.

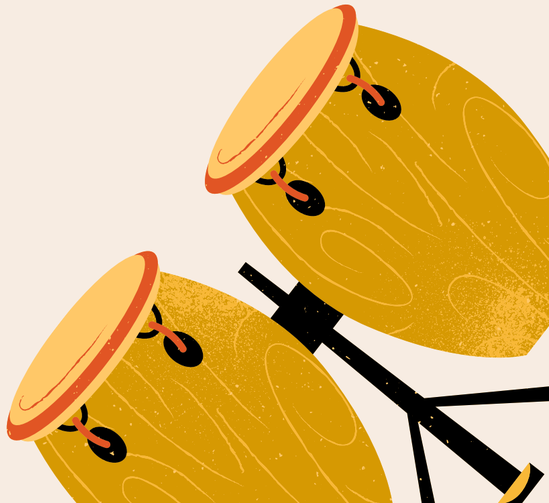


```
for i in lista:
    print(f"Valores únicos en {i}: {df[i].unique()}")
```

✓ 0.0s

```
 "God's Problem Child" 'Slow to Me (feat. Krizz Kaliko, Rittz)'
 'Don’t Play This Song']
Valores únicos en popularity: ['27.0' '31.0' '28.0' '34.0' '32.0' '47.0' '46.0' '43.0' '39.0' '22.0'
 '30.0' nan '50.0' '59.0' '29.0' '35.0' '44.0' '33.0' '56.0' '21.0' '48.0'
 '45.0' '53.0' '63.0' '25.0' '36.0' '37.0' '51.0' '55.0' '49.0' '41.0'
 '38.0' '52.0' '24.0' '26.0' '96.0' '42.0' '40.0' '23.0' '61.0' '54.0'
 '66.0' '70.0' '67.0' '60.0' '58.0' '65.0' '69.0' '72.0' '64.0' '62.0'
 '57.0' '0.0' '76.0' '20.0' '74.0' '71.0' '84.0' '68.0' 'invalid' '18.0'
 '82.0' '3.0' '11.0' '17.0' '15.0' '12.0' '10.0' '13.0' '16.0' '14.0'
 '9.0' '19.0' '8.0' '7.0' '4.0' '2.0' '1.0' '5.0' '6.0' '79.0' '73.0'
 '75.0' '78.0' '83.0' '81.0' '80.0' '77.0' '85.0' '97.0' '88.0' '87.0'
 '86.0' '89.0' '93.0' '90.0' '94.0' '95.0' '92.0' '91.0']
Valores únicos en acousticness: ['0.00468' '0.0127' '0.00306' ... '5.2e-05' '0.000298'
 '0.0003129999999999']
Valores únicos en danceability: ['0.652' '0.622' '0.62' ... '0.974' '0.98' '0.966']
Valores únicos en duration_ms: [-1.00000e+00  2.18293e+05  2.15613e+05 ...  2.22247e+05  3.96520e+05
  5.13502e+05]
Valores únicos en energy: ['0.941' '0.89' '0.755' ... '0.00226' '0.0776' '0.00198']
Valores únicos en instrumentalness: ['0.792' '0.95' '0.0118' ... '0.0914' '0.000926' '3.9299999999999999e-05']
Valores únicos en key: ['A#' nan 'G#' 'C#' 'F#' 'D' 'B' 'G' 'F' 'A' 'C' 'E' 'D#' 'invalid']
Valores únicos en liveness: ['0.115' '0.124' '0.534' ... '0.0352' '0.8029999999999999' '0.0196']
Valores únicos en loudness: ['-5.2010000000000005' '-7.042999999999999' '-4.617' ... '-23.346'
 '-2.527' '-22.884']
Valores únicos en mode: ['Minor' nan 'Major' 'invalid']
Valores únicos en speechiness: ['0.0748' '0.03' '0.0345' ... '0.675' '0.7040000000000001' '0.855']
Valores únicos en tempo: ['100.889' '115.00200000000001' '127.994' ... '81.143' '84.264' '140.407']
Valores únicos en obtained_date: ['4-Apr' '3-Apr' nan 'invalid' '5-Apr' '1-Apr' '0/4']
Valores únicos en valence: ['0.759' '0.531' '0.3329999999999999' ... '0.0292' '0.0633' '0.0755']
Valores únicos en music_genre: ['Electronic' nan 'invalid' 'Anime' 'Jazz' 'Alternative' 'Country' 'Rap'
 'Blues' 'Rock' 'Classical' 'Hip-Hop']
```

Con el siguiente ciclo for obtengo los valores únicos de cada columna, notando así que el valor 'invalid' es otro dato erróneo a tratar.



PROCESO DE LIMPIEZA

Cambiar el tipo de dato de las columnas necesarias para un futuro análisis y realizar una imputación con la media o una cadena de caracteres específicos, por ejemplo: "s/id", eliminacion de duplicados, etc.

Convierto los datos de las columnas con datos cuantitativos a usar en un próximo análisis a flot con el siguiente código, el cual convierte a la par cadenas de texto incorrectas a NaN.

```
df['popularity'] = pd.to_numeric(df['popularity'], errors='coerce')
df['acousticness'] = pd.to_numeric(df['acousticness'], errors='coerce')
df['danceability'] = pd.to_numeric(df['danceability'], errors='coerce')
df['duration_ms'] = pd.to_numeric(df['duration_ms'], errors='coerce')
df['energy'] = pd.to_numeric(df['energy'], errors='coerce')
df['instrumentalness'] = pd.to_numeric(df['instrumentalness'], errors='coerce')
df['liveness'] = pd.to_numeric(df['liveness'], errors='coerce')
df['loudness'] = pd.to_numeric(df['loudness'], errors='coerce')
df['speechiness'] = pd.to_numeric(df['speechiness'], errors='coerce')
df['tempo'] = pd.to_numeric(df['tempo'], errors='coerce')
df['valence'] = pd.to_numeric(df['valence'], errors='coerce')
```

Como lo mencione, este aumento la cantidad de NaN, pero al tratarlos lograré eliminar al mismo tiempo las cadenas de texto incorrectas dentro de los datos.

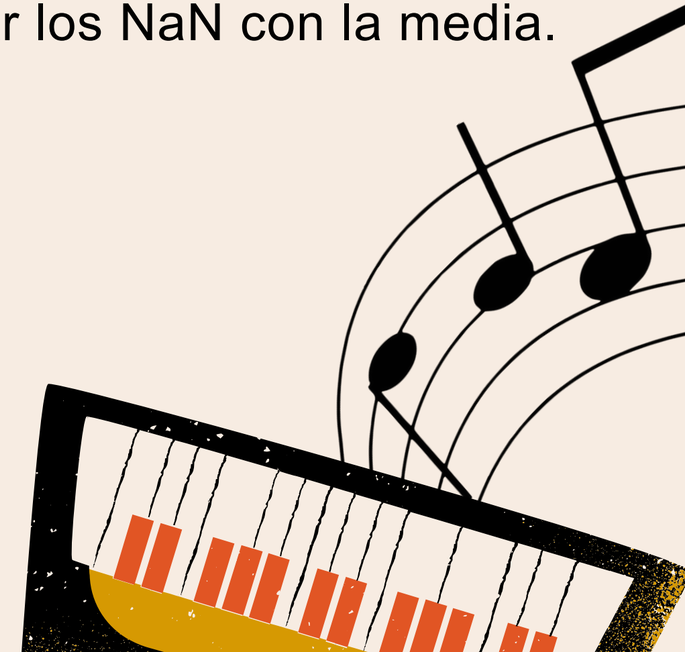
```
df.isnull().mean()*100
```

✓ 0.0s

```
instance_id          4.695408
artist_name          4.695408
track_name           4.695408
popularity           4.783505
acousticness         4.796626
danceability         4.791003
duration_ms          4.819119
energy               4.822868
instrumentalness     4.804124
key                  4.695408
liveness             4.820993
loudness             4.809747
mode                 4.695408
speechiness          4.824742
tempo               14.234302
obtained_date        4.695408
valence              4.787254
music_genre          4.695408
dtype: float64
```

```
df_popularity = df.dropna(subset=['popularity'])
df_acousticness = df.dropna(subset=['acousticness'])
df_danceability  = df.dropna(subset=['danceability'])
df_duration_ms  = df.dropna(subset=['duration_ms'])
df_energy  = df.dropna(subset=['energy'])
df_instrumentalness  = df.dropna(subset=['instrumentalness'])
df_liveness = df.dropna(subset=['liveness'])
df_loudness  = df.dropna(subset=['loudness'])
df_speechiness  = df.dropna(subset=['speechiness'])
df_tempo  = df.dropna(subset=['tempo'])
df_valence  = df.dropna(subset=['valence'])
```

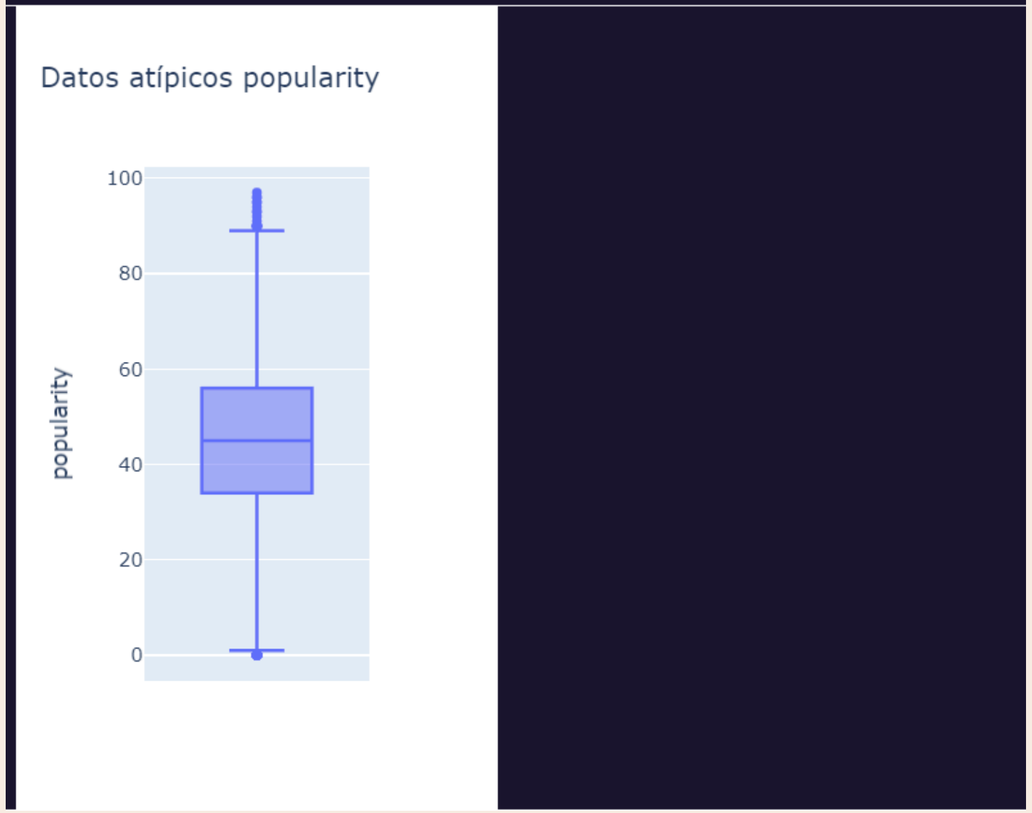
Creo un dataframe usando .dropna () para cada columna cuantitativa, esto es parte de un proceso de rellenar los NaN con la media.



Para que al rellenar con la media de los datos de cada nuevo dataframe, los datos atípicos no la alteren, de forma errónea decido usar bloxpot y el Método del rango inter cuartil (IQR) para eliminarlos.

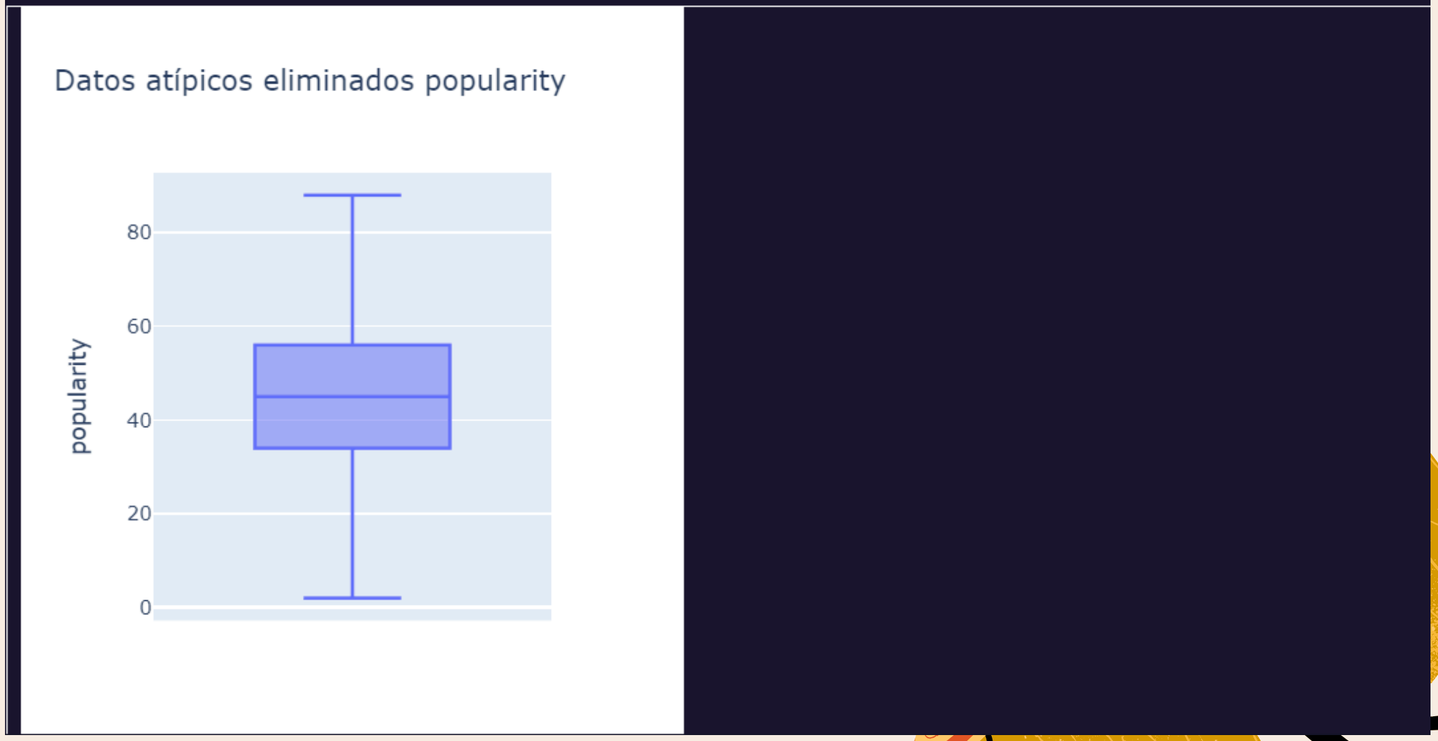
```
fig = px.box(df_popularity, y='popularity', title='Datos atípicos popularity')
fig.update_layout(width=300,height=500)
fig.show()
```

✓ 0.0s




```
iqr = df_popularity['popularity'].quantile(0.75) - df_popularity['popularity'].quantile(0.25)
#Desarrollamos filtros superior o inferior por lo general es el iqr por 1.5 pero se puede manejar el número
filtro_inferior = df_popularity['popularity'] > df_popularity['popularity'].quantile(0.25) - (iqr * 1.5)
filtro_superior = df_popularity['popularity'] < df_popularity['popularity'].quantile(0.75) + (iqr * 1.5)
df2_popularity = df_popularity[filtro_inferior & filtro_superior]
#Graficando el boxplot
fig = px.box(df2_popularity, y='popularity', title='Datos atípicos eliminados popularity')
fig.update_layout(width=400,height=450)
fig.show()
```

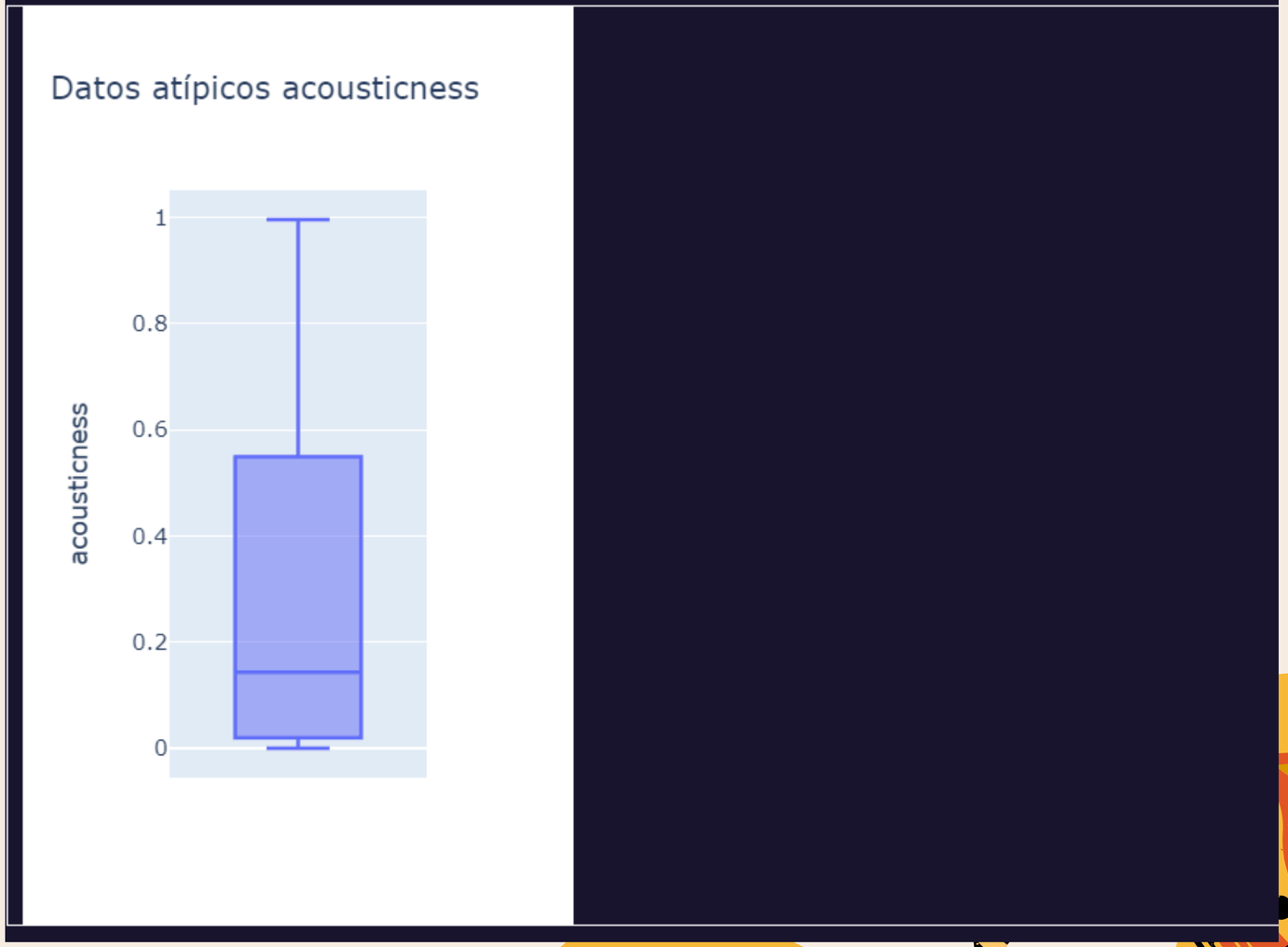
✓ 0.1s



Existieron columnas que no contaban con datos atípicos como la siguiente:

```
fig = px.box(df_acousticness, y='acousticness', title='Datos atípicos acousticness')
fig.update_layout(width=300,height=500)
fig.show()
```

✓ 0.1s



Descubrí columnas que requieren
un análisis profundo en un futuro.

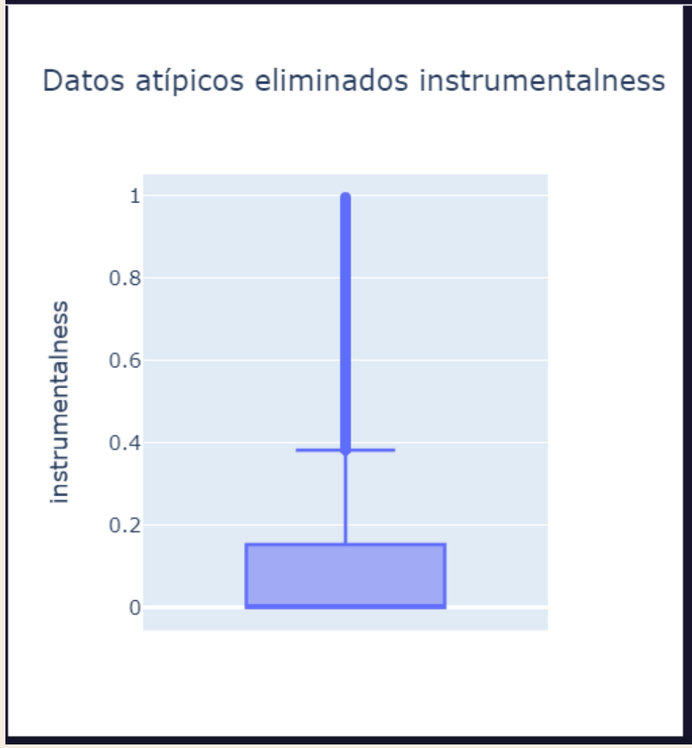
```python
iqr = df_instrumentalness['instrumentalness'].quantile(0.75) - df_instrumentalness['instrumentalness'].quantile(0.25)

filtro_inferior = df_instrumentalness['instrumentalness'] > df_instrumentalness['instrumentalness'].quantile(0.25) - (iqr * 10)
filtro_superior = df_instrumentalness['instrumentalness'] < df_instrumentalness['instrumentalness'].quantile(0.75) + (iqr * 10)

df2_instrumentalness = df_instrumentalness[filtro_inferior & filtro_superior]

fig = px.box(df2_instrumentalness, y='instrumentalness', title='Datos atípicos eliminados instrumentalness')
fig.update_layout(width=400,height=450)
fig.show()
```

✓ 0.1s



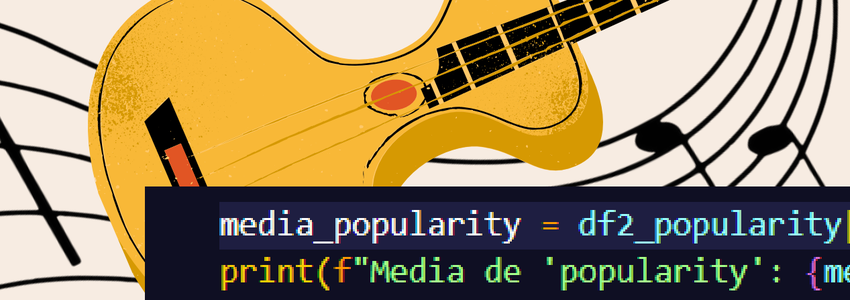

```python
x=df['instrumentalness'].mean()

y=df2_instrumentalness['instrumentalness'].mean()

print(f'1:{x} , 2:{y}')
```

✓ 0.0s

```
1:0.18124572344005355 , 2:0.18124572344005357
```

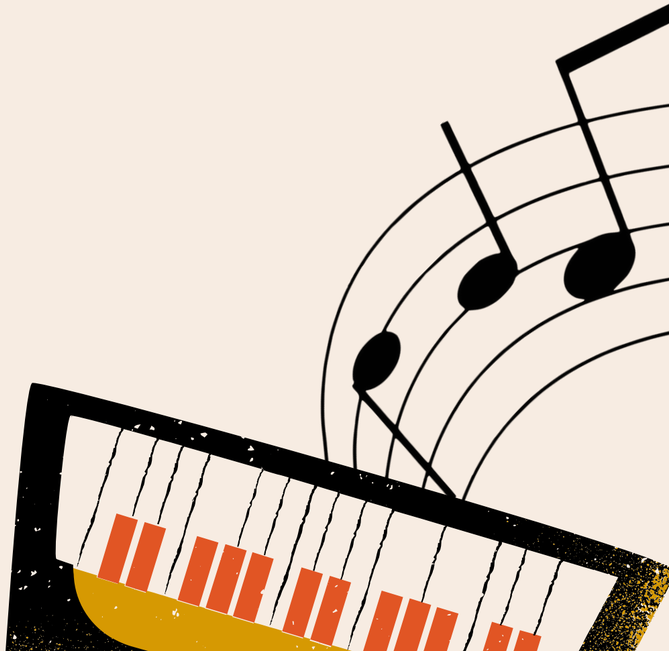


```python
media_popularity = df2_popularity['popularity'].mean()
print(f"Media de 'popularity': {media_popularity}")
media_acousticness = df_acousticness['acousticness'].mean()
print(f"Media de 'acousticness': {media_acousticness}")
media_danceability = df2_danceability['danceability'].mean()
print(f"Media de 'danceability': {media_danceability}")
media_duration_ms = df2_duration_ms['duration_ms'].mean()
print(f"Media de 'duration_ms': {media_duration_ms}")
media_energy = df_energy['energy'].mean()
print(f"Media de 'energy': {media_energy}")
media_loudness = df2_loudness['loudness'].mean()
print(f"Media de 'loudness': {media_loudness}")
media_tempo = df2_tempo['tempo'].mean()
print(f"Media de 'tempo': {media_tempo}")
media_valence = df_valence['valence'].mean()
print(f"Media de 'valence': {media_valence}")
```

✓ 0.0s

```
Media de 'popularity': 44.890298984731
Media de 'acousticness': 0.30546496604792184
Media de 'danceability': 0.5610833878287061
Media de 'duration_ms': 226577.84929778625
Media de 'energy': 0.6001229639009787
Media de 'loudness': -7.227285288202997
Media de 'tempo': 119.78415396237985
Media de 'valence': 0.45633677651783605
```

Obtengo la media de cada nuevo dataframe de la columna específica con el código que se observa.



Creo un nuevo copia del dataframe original y con el siguiente código mediante la función .fillna() rellenas los valores NaN que ya se habían mencionado con la media respectiva de cada columna de los dataframes sin datos atípicos.

```
df_limpio = df
```

✓ 0.0s

```
df_limpio['popularity'] = df_limpio['popularity'].fillna(df2_popularity['popularity'].mean())
df_limpio['acousticness'] = df_limpio['acousticness'].fillna(df_acousticness['acousticness'].mean())
df_limpio['danceability'] = df_limpio['danceability'].fillna(df2_danceability['danceability'].mean())
df_limpio['duration_ms'] = df_limpio['duration_ms'].fillna(df2_duration_ms['duration_ms'].mean())
df_limpio['energy'] = df_limpio['energy'].fillna(df_energy['energy'].mean())
df_limpio['loudness'] = df_limpio['loudness'].fillna(df2_loudness['loudness'].mean())
df_limpio['tempo'] = df_limpio['tempo'].fillna(df_tempo['tempo'].mean())
df_limpio['valence'] = df_limpio['valence'].fillna(df_valence['valence'].mean())
df_limpio['instrumentalness'] = df_limpio['instrumentalness'].fillna(df2_instrumentalness['instrumentalness'].mean())
df_limpio['liveness'] = df_limpio['liveness'].fillna(df_liveness['liveness'].mean())
df_limpio['speechiness'] = df_limpio['speechiness'].fillna(df2_speechiness['speechiness'].mean())
```

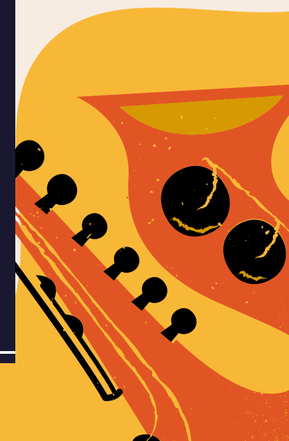
✓ 0.0s

Como se esperaba, los datos NaN de las columnas cuantitativas han sido rellenadas con la media correctamente.

```
df_limpio.isnull().sum()
```

✓ 0.0s

```
instance_id          2505
artist_name          2505
track_name           2505
popularity              0
acousticness            0
danceability            0
duration_ms             0
energy                  0
instrumentalness        0
key                  2505
liveness                0
loudness                0
mode                 2505
speechiness             0
tempo                   0
obtained_date        2505
valence                 0
music_genre          2505
dtype: int64
```



El siguiente código rellena los NaN de las columnas categoricas no significativas para mi análisis con una cadena de caracteres, como ya se había mencionado. Verificando también que no se ha eliminado ni una sola fila.

```
df_limpio.shape
```

✓ 0.0s

```
(53350, 18)
```

```
df_limpio['instance_id'] = df_limpio['instance_id'].fillna('s/id')
df_limpio['artist_name'] = df_limpio['artist_name'].fillna('s/an')
df_limpio['track_name'] = df_limpio['track_name'].fillna('s/tn')
df_limpio['obtained_date'] = df_limpio['obtained_date'].fillna('s/od')
```

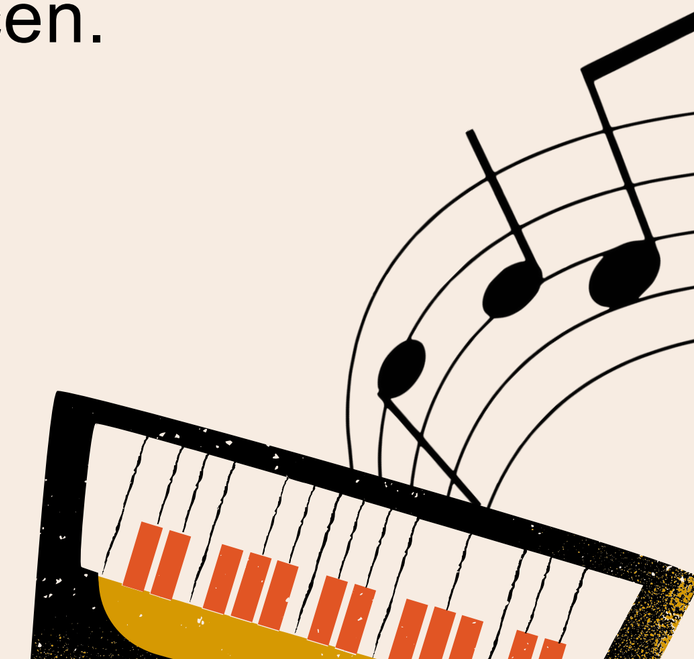
✓ 0.0s

```
df_limpio.isnull().sum()
```

✓ 0.0s

```
instance_id            0
artist_name            0
track_name             0
popularity             0
acousticness           0
danceability           0
duration_ms            0
energy                 0
instrumentalness       0
key                 2505
liveness               0
loudness               0
mode                2505
speechiness            0
tempo                  0
obtained_date          0
valence                0
music_genre         2505
dtype: int64
```

Nuevamente, los valores NaN se reducen.



Esta vez decido usar la función.dropna() con el motivo de limpiar los NaN de las columnas categóricas que serán necesarias para un futuro análisis, mostrando un antes y después de la cantidad de filas del dataframe.

```
df_limpio.shape
```

✓ 0.0s

```
(53350, 18)
```

```
df_limpio = df_limpio.dropna(subset=['key', 'mode', 'music_genre' ])
```

✓ 0.0s

```
df_limpio.shape
```

✓ 0.0s

```
(46184, 18)
```

Verifico la cantidad de duplicados y los elimino.

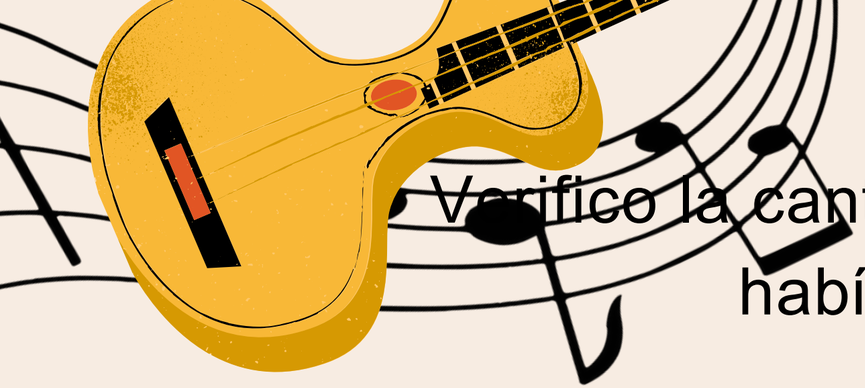
```
df_limpio.duplicated().sum()
```

✓ 0.0s

```
np.int64(619)
```

```
df_limpio2 = df_limpio.drop_duplicates()
```

✓ 0.0s



Verifico la cantidad de valores inválidos 'invalid' como ya lo había nota anteriormente y los elimino.

```python
for i in lista:
    print(f"En la columna {i} los invalid son: {df_limpio2[df_limpio2[i] == 'invalid'].shape[0]}")
```

✓ 0.0s

```
En la columna instance_id los invalid son: 53
En la columna artist_name los invalid son: 41
En la columna track_name los invalid son: 56
En la columna popularity los invalid son: 0
En la columna acousticness los invalid son: 0
En la columna danceability los invalid son: 0
En la columna duration_ms los invalid son: 0
En la columna energy los invalid son: 0
En la columna instrumentalness los invalid son: 0
En la columna key los invalid son: 66
En la columna liveness los invalid son: 0
En la columna loudness los invalid son: 0
En la columna mode los invalid son: 47
En la columna speechiness los invalid son: 0
En la columna tempo los invalid son: 0
En la columna obtained_date los invalid son: 45
En la columna valence los invalid son: 0
En la columna music_genre los invalid son: 62
```

```python
df_limpio2 = df_limpio2
for i in lista:
    df_limpio2=df_limpio2[df_limpio2[i] != 'invalid']
```

✓ 0.1s

RESULTADOS

A continuación se mostraron los resultados y estadísticas finales de la base de datos tras la limpieza.

Se corrobora que los tipos de datos están correctos, y la función describe ya se aplica correctamente sobre los datos cuantitativos. La fecha es un dato sin importancia, por eso no tiene cambios de tipo.

```
df_limpio2.info()
```

✓ 0.0s

```
<class 'pandas.core.frame.DataFrame'>
Index: 45195 entries, 0 to 53349
Data columns (total 18 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   instance_id       45195 non-null  object 
 1   artist_name       45195 non-null  object 
 2   track_name        45195 non-null  object 
 3   popularity        45195 non-null  float64
 4   acousticness      45195 non-null  float64
 5   danceability      45195 non-null  float64
 6   duration_ms       45195 non-null  float64
 7   energy            45195 non-null  float64
 8   instrumentalness  45195 non-null  float64
 9   key               45195 non-null  object 
 10  liveness          45195 non-null  float64
 11  loudness          45195 non-null  float64
 12  mode              45195 non-null  object 
 13  speechiness       45195 non-null  float64
 14  tempo             45195 non-null  float64
 15  obtained_date     45195 non-null  object 
 16  valence           45195 non-null  float64
 17  music_genre       45195 non-null  object 
dtypes: float64(11), object(7)
memory usage: 6.6+ MB
```

```
df_limpio2.describe()
```

✓ 0.0s Python

	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	liveness	loudness	speechiness	tempo	valence
count	45195.000000	45195.000000	45195.000000	4.519500e+04	45195.000000	45195.000000	45195.000000	45195.000000	45195.000000	45195.000000	45195.000000
mean	44.316753	0.305900	0.558362	2.216374e+05	0.600142	0.180900	0.194171	-9.037802	0.091843	119.880067	0.456941
std	15.171509	0.332173	0.173866	1.254725e+05	0.257952	0.316391	0.157867	6.017529	0.099471	28.374292	0.241051
min	0.000000	0.000001	0.059600	-1.000000e+00	0.000792	0.000000	0.009670	-47.046000	0.022300	34.347000	0.000000
25%	34.000000	0.023000	0.451000	1.777680e+05	0.448000	0.000000	0.098500	-10.585500	0.036600	97.953000	0.270000
50%	44.890299	0.169000	0.561083	2.236170e+05	0.625000	0.000315	0.132000	-7.227285	0.051100	119.957332	0.456337
75%	55.000000	0.520000	0.679000	2.657235e+05	0.807000	0.181246	0.233000	-5.279000	0.093250	137.653000	0.638000
max	96.000000	0.996000	0.980000	4.830606e+06	0.999000	0.996000	1.000000	1.949000	0.942000	220.041000	0.992000

```
df_limpio2.duplicated().sum()
```

✓ 0.0s

```
np.int64(0)
```

```
df_limpio2.isnull().sum()
```

✓ 0.0s

```
instance_id         0
artist_name         0
track_name          0
popularity          0
acousticness        0
danceability        0
duration_ms         0
energy              0
instrumentalness    0
key                 0
liveness            0
loudness            0
mode                0
speechiness         0
tempo               0
obtained_date       0
valence             0
music_genre         0
dtype: int64
```

```
for i in lista:
    print(f"En la columna {i} los invalid son: {df_limpio2[df_limpio2[i] == 'invalid'].shape[0]}")
```

✓ 0.0s

```
En la columna instance_id los invalid son: 0
En la columna artist_name los invalid son: 0
En la columna track_name los invalid son: 0
En la columna popularity los invalid son: 0
En la columna acousticness los invalid son: 0
En la columna danceability los invalid son: 0
En la columna duration_ms los invalid son: 0
En la columna energy los invalid son: 0
En la columna instrumentalness los invalid son: 0
En la columna key los invalid son: 0
En la columna liveness los invalid son: 0
En la columna loudness los invalid son: 0
En la columna mode los invalid son: 0
En la columna speechiness los invalid son: 0
En la columna tempo los invalid son: 0
En la columna obtained_date los invalid son: 0
En la columna valence los invalid son: 0
En la columna music_genre los invalid son: 0
```

Como se mostró en las imágenes anteriores, toda la base de datos está limpia, notando que ya no se presencian datos NaN en el dataframe impreso a continuación.

```
df_limpio2 = df_limpio2.reset_index(drop=True)
```
✓ 0.0s Pytho

```
df_limpio2.head(10)
```
✓ 0.0s Pytho

	instance_id	artist_name	track_name	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	key	liveness	loudness	mode	speechiness
0	32894.0	Röyksopp	Röyksopp's Night Out	27.0	0.00468	0.652	-1.0	0.941000	0.792000	A#	0.115	-5.201000	Minor	0.074800
1	30097.0	Dillon Francis	Hurricane	28.0	0.00306	0.620	215613.0	0.755000	0.011800	G#	0.534	-4.617000	Major	0.034500
2	62177.0	Dubloadz	Nitro	34.0	0.02540	0.774	166875.0	0.700000	0.002530	C#	0.157	-4.498000	Major	0.239000
3	24907.0	What So Not	Divide & Conquer	32.0	0.00465	0.638	222369.0	0.587000	0.909000	F#	0.157	-7.227285	Major	0.054473
4	89064.0	Axel Boman	Hello	47.0	0.00523	0.755	519468.0	0.731000	0.854000	D	0.216	-10.517000	Minor	0.041200
5	43760.0	Jordan Comolli	s/tn	46.0	0.02890	0.572	214408.0	0.803000	0.000008	B	0.106	-4.294000	Major	0.054473
6	84950.0	Kayzo	NEVER ALONE	39.0	0.00299	0.509	292800.0	0.600123	0.000276	F	0.178	-3.175000	Minor	0.268000
7	56950.0	Shlump	Lazer Beam	22.0	0.00934	0.578	204800.0	0.731000	0.011200	A	0.111	-7.091000	Minor	0.173000
8	49030.0	Chase & Status	Lost & Not Found - Acoustic	30.0	0.85500	0.607	170463.0	0.158000	0.000000	F#	0.106	-13.787000	Minor	0.034500
9	22654.0	G Jones	Mind	27.0	0.03370	0.513	165132.0	0.828000	0.569000	B	0.109	-5.439000	Minor	0.060900

Finalmente, el paso final, crea una nueva base de datos limpia.

```
df_limpio2.to_csv("DF_Music_Limpio.csv", index=[True])
```
✓ 0.5s

FIN DE LA LIMPIEZA