```
#importing libraries
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import numpy as np
pd.pandas.set_option("display.max_columns",None)
from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn.preprocessing import OneHotEncoder
from sklearn.impute import KNNImputer
from sklearn.impute import SimpleImputer
import xgboost as xgb
from sklearn.metrics import r2_score
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV
```

```
#Read the CSV file "spotify-2023.csv" into a DataFrame and assign it to the variable df1
df1=pd.read_csv("spotify-2023.csv",encoding= 'unicode_escape')
```

```
#Display the first few rows of the DataFrame df1 using the head() function
df1.head()
```

	track_name	artist(s)_name	artist_count	released_year	released_month	released_day	in_spotify_playlists	in_spotify_charts	str
0	Seven (feat. Latto) (Explicit Ver.)	Latto, Jung Kook	2	2023	7	14	553	147	14138
1	LALA	Myke Towers	1	2023	3	23	1474	48	13371
2	vampire	Olivia Rodrigo	1	2023	6	30	1397	113	14000
3	Cruel	Taylor Swift	1	2019	8	23	7858	100	

```
#Get the dimensions (number of rows and columns) of the DataFrame df1 using the shape attribute
df1.shape
```

```
(953, 24)
```

```
# Print a message indicating that you are printing information about input features
print("Input Features in this dataset are:")
count=1
print("")
# Print a header for the feature information table
print("S.no"," ","Feature name"," ","Datatypes")
# Iterate over the columns and their corresponding data types in the DataFrame
for i,j in zip(df1.columns,df1.dtypes):
  # Exclude the "streams" feature from the input features
    if i!="streams":
      # Print the serial number, feature name, and data type
        print(count,'.',i," ",j)
        count=count+1     # Increment the counter
# Print a message indicating the output feature and its data type
print("Output Feature in this dataset is:streams int64")
```

```
Input Features in this dataset are:

S.no   Feature name   Datatypes
1 . track_name   object
2 . artist(s)_name   object
3 . artist_count   int64
4 . released_year   int64
5 . released_month   int64
6 . released_day   int64
7 . in_spotify_playlists   int64
8 . in_spotify_charts   int64
9 . in_apple_playlists   int64
10 . in_apple_charts   int64
11 . in_deezer_playlists   object
12 . in_deezer_charts   int64
13 . in_shazam_charts   object
14 . bpm   int64
15 . key   object
16 . mode   object
```

```
17 . danceability_%   int64
18 . valence_%   int64
19 . energy_%   int64
20 . acousticness_%   int64
21 . instrumentalness_%   int64
22 . liveness_%   int64
23 . speechiness_%   int64
Output Feature in this dataset is:streams int64
```

track_name: Name of the song

artist(s)_name: Name of the artist(s) of the song

artist_count: Number of artists contributing to the song

released_year: Year when the song was released

released_month: Month when the song was released

released_day: Day of the month when the song was released

in_spotify_playlists: Number of Spotify playlists the song is included in

in_spotify_charts: Presence and rank of the song on Spotify charts

streams: Total number of streams on Spotify

in_apple_playlists: Number of Apple Music playlists the song is included in

in_apple_charts: Presence and rank of the song on Apple Music charts

in_deezer_playlists: Number of Deezer playlists the song is included in

in_deezer_charts: Presence and rank of the song on Deezer charts

in_shazam_charts: Presence and rank of the song on Shazam charts

bpm: Beats per minute, a measure of song tempo

key: Key of the song

mode: Mode of the song (major or minor)

danceability_%: Percentage indicating how suitable the song is for dancing

valence_%: Positivity of the song's musical content

energy_%: Perceived energy level of the song

acousticness_%: Amount of acoustic sound in the song

instrumentalness_%: Amount of instrumental content in the song

liveness_%: Presence of live performance elements

speechiness_%: Amount of spoken words in the song

```
# Generate descriptive statistics for the DataFrame df1 using the describe() function
df1.describe()
# This function provides information such as count, mean, std (standard deviation), min, 25th percentile, median (50th percentile), 75th per
# It gives a quick summary of the central tendency and spread of the numerical data in the DataFrame
```

	artist_count	released_year	released_month	released_day	in_spotify_playlists	in_spotify_charts	in_apple_playlists	in_apple_
count	953.000000	953.000000	953.000000	953.000000	953.000000	953.000000	953.000000	953.
mean	1.556139	2018.238195	6.033578	13.930745	5200.124869	12.009444	67.812172	51.
std	0.893044	11.116218	3.566435	9.201949	7897.608990	19.575992	86.441493	50.
min	1.000000	1930.000000	1.000000	1.000000	31.000000	0.000000	0.000000	0.
25%	1.000000	2020.000000	3.000000	6.000000	875.000000	0.000000	13.000000	7.
50%	1.000000	2022.000000	6.000000	13.000000	2224.000000	3.000000	34.000000	38.
75%	2.000000	2022.000000	9.000000	22.000000	5542.000000	16.000000	88.000000	87.
max	8.000000	2023.000000	12.000000	31.000000	52898.000000	147.000000	672.000000	275.

```
df1.isna().sum()
#The missing value features in this dataset are in_shazam_charts and key
```

```
track_name               0
artist(s)_name           0
artist_count             0
released_year            0
released_month           0
released_day             0
in_spotify_playlists     0
in_spotify_charts        0
streams                  0
in_apple_playlists       0
in_apple_charts          0
in_deezer_playlists      0
in_deezer_charts         0
in_shazam_charts        50
bpm                      0
key                     95
mode                     0
danceability_%           0
valence_%                0
energy_%                 0
acousticness_%           0
instrumentalness_%       0
liveness_%               0
speechiness_%            0
dtype: int64
```

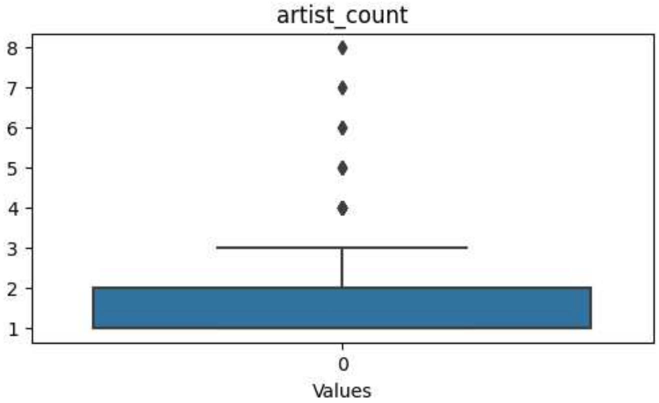
```
# Convert the "in_shazam_charts" column to numeric values, removing commas and handling errors by coercing to NaN
df1["in_shazam_charts"] = pd.to_numeric(df1["in_shazam_charts"].str.replace(',', ''), errors='coerce', downcast='integer')
# Convert the "in_deezer_playlists" column to numeric values, removing commas and handling errors by coercing to NaN
df1["in_deezer_playlists"] = pd.to_numeric(df1["in_deezer_playlists"].str.replace(',', ''), errors='coerce', downcast='integer')
```

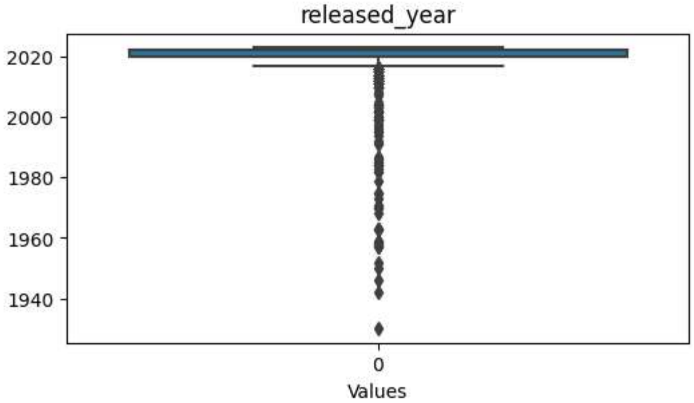
This two columns are numeric values but they were filled in a string format included with commas so we converted them into integer again

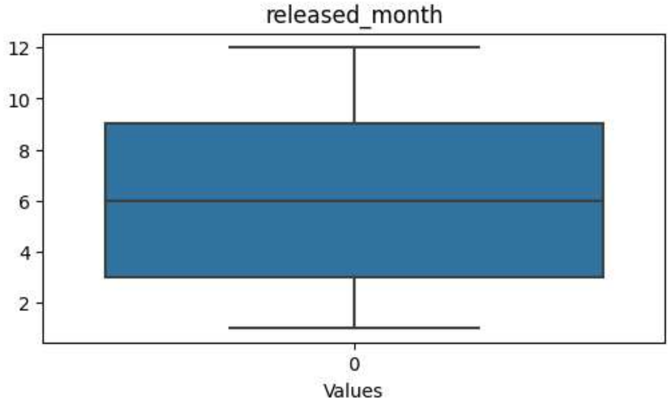
```
# Initialize an empty list to store the names of numerical features
numerical_features=[]
# Iterate through the columns of the DataFrame df1
for feature in df1.columns:
  # Check if the data type of the current column is not 'object' (i.e., not a string)
    if df1[feature].dtypes!='O':
      # If it's not a string, add the feature name to the list of numerical features
        numerical_features.append(feature)
```

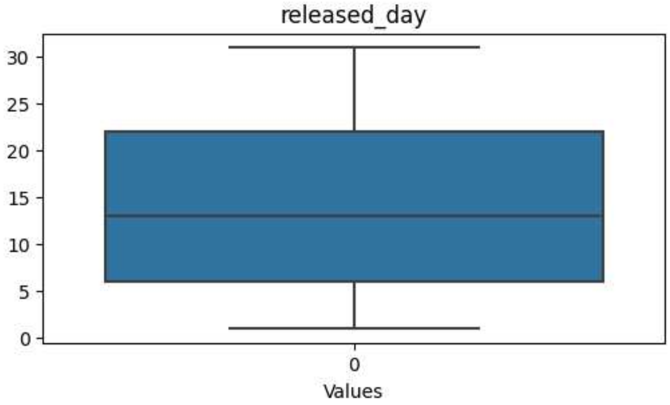
This code will give numerical features from dataframe

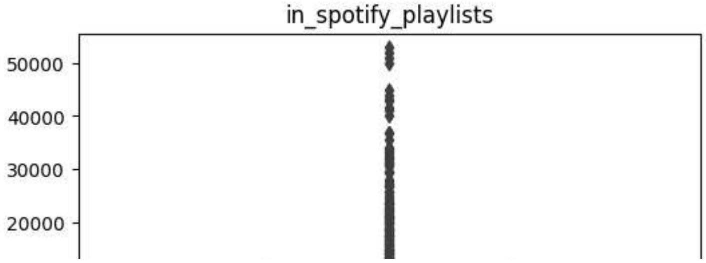
```
# Iterate through each numerical feature in the list numerical_features
for feature in numerical_features:
  # Create a new figure with a specific size
    plt.figure(figsize=(6,3))
    # Use Seaborn to create a vertical boxplot for the current numerical feature
    sns.boxplot(data=df1[feature],orient='v')
    # Set the title of the boxplot to the current feature name
    plt.title(feature)
    # Set the label for the x-axis
    plt.xlabel("Values")
    # Display the boxplot
    plt.show()
```

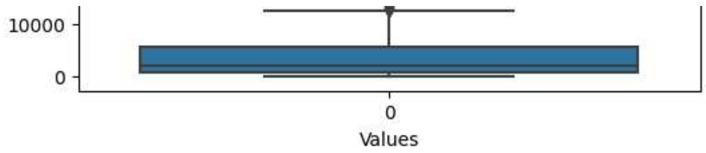

artist_count

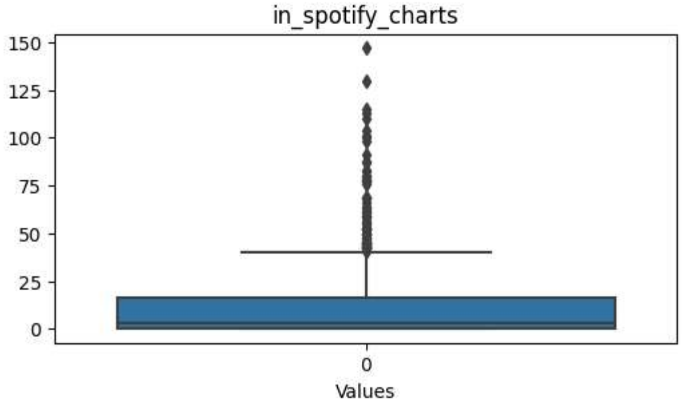

released_year

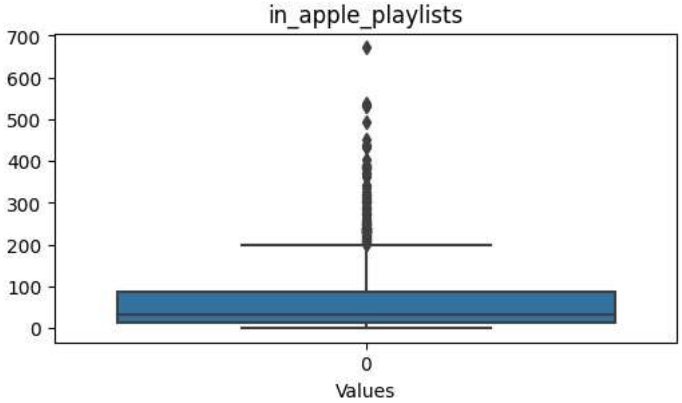

released_month

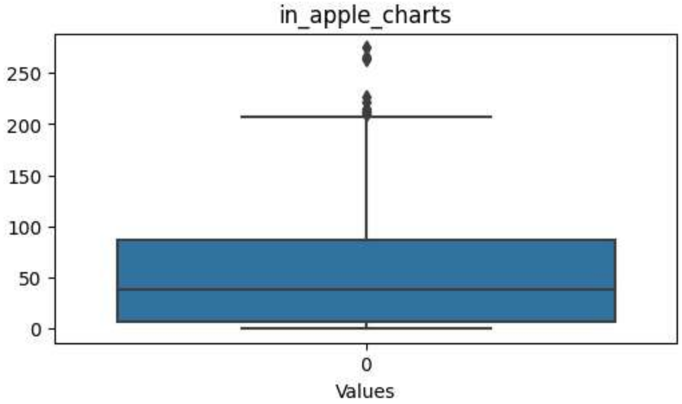

released_day

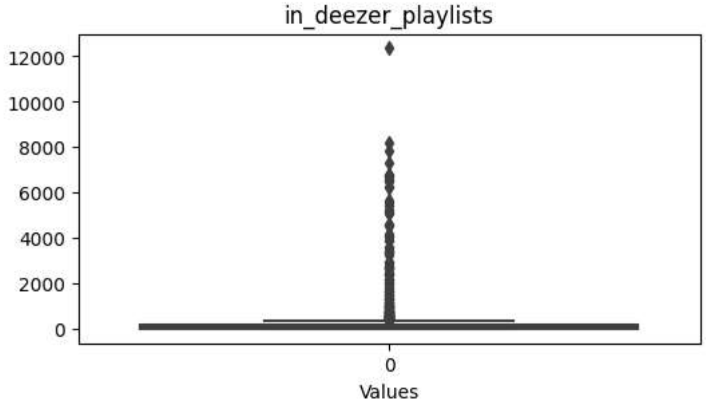

in_spotify_playlists

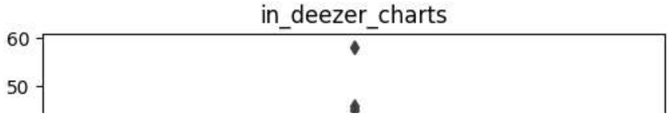

10000
0
0
Values

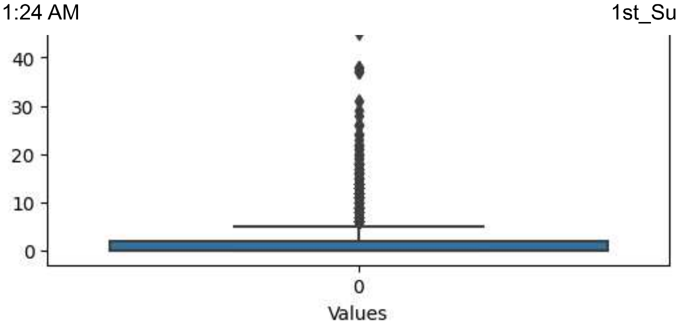

in_spotify_charts
150
125
100
75
50
25
0
0
Values

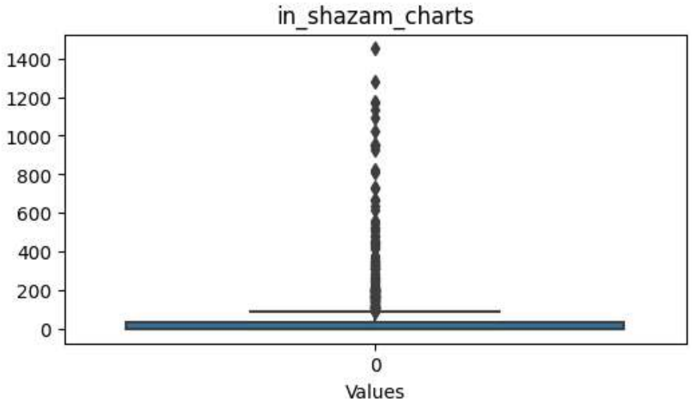

in_apple_playlists
700
600
500
400
300
200
100
0
0
Values

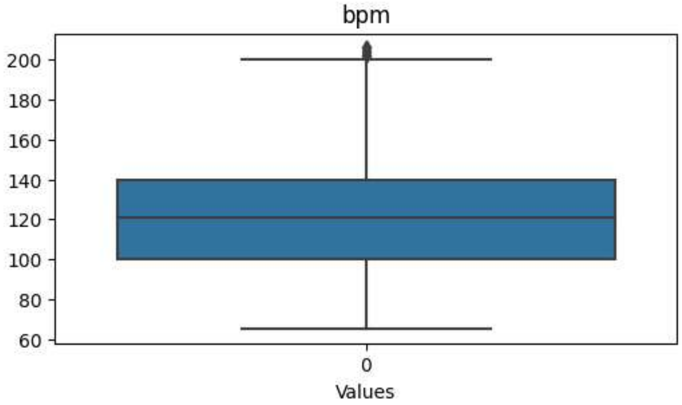

in_apple_charts
250
200
150
100
50
0
0
Values

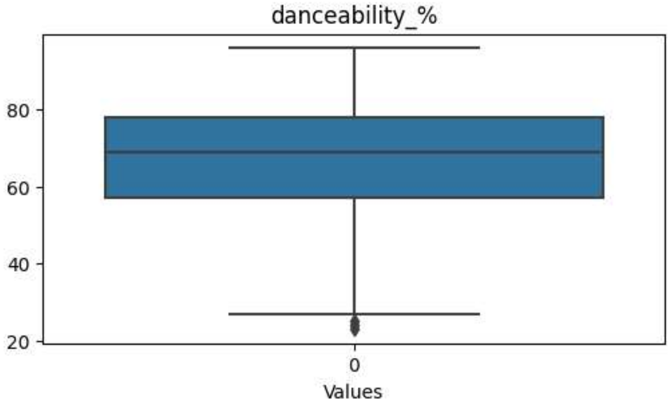

in_deezer_playlists
12000
10000
8000
6000
4000
2000
0
0
Values

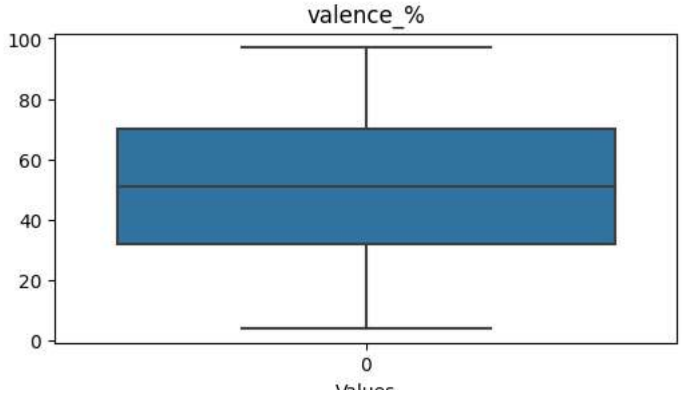

in_deezer_charts
60
50


40
30
20
10
0
0
Values


in_shazam_charts
1400
1200
1000
800
600
400
200
0
0
Values

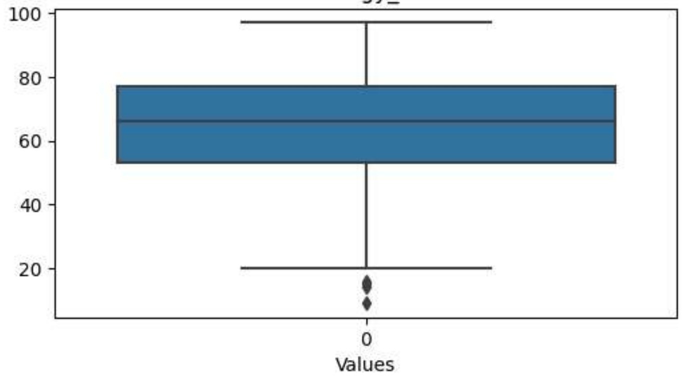

bpm
200
180
160
140
120
100
80
60
0
Values


danceability_%
80
60
40
20
0
Values

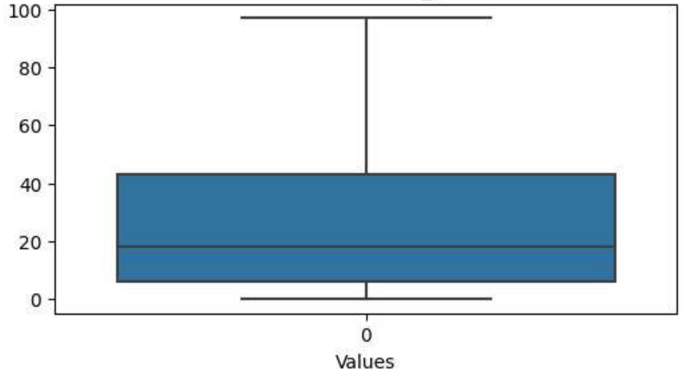

valence_%
100
80
60
40
20
0
0

Values

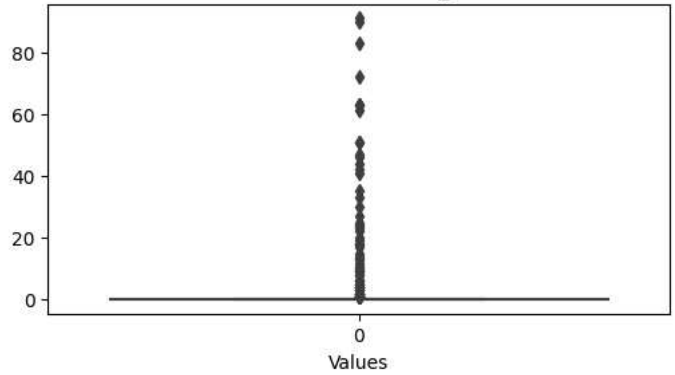
energy_%



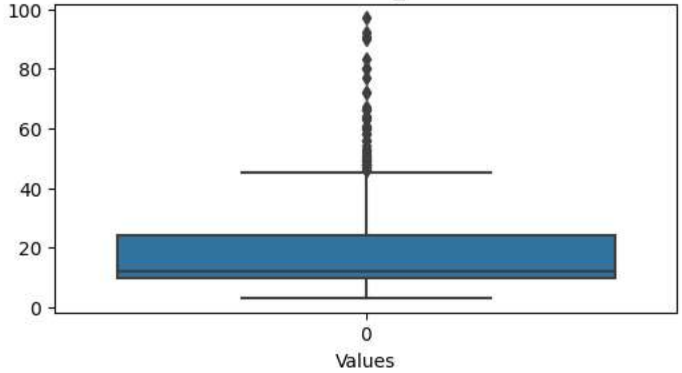
acousticness_%



instrumentalness_%



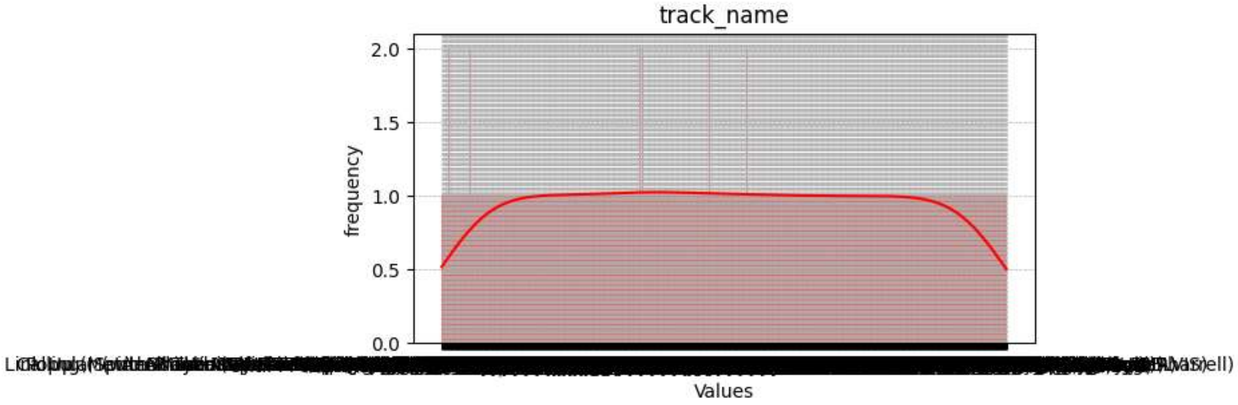
liveness_%

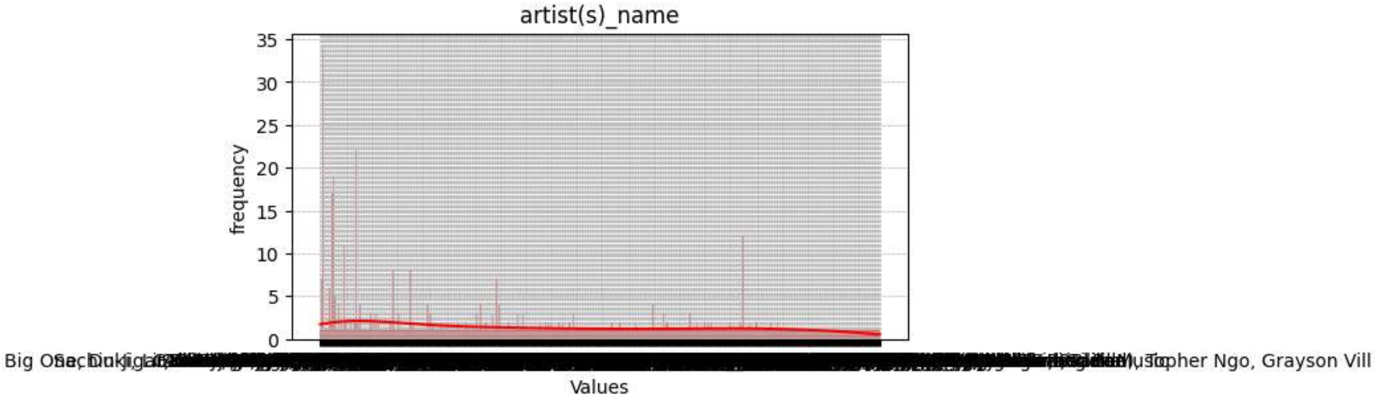


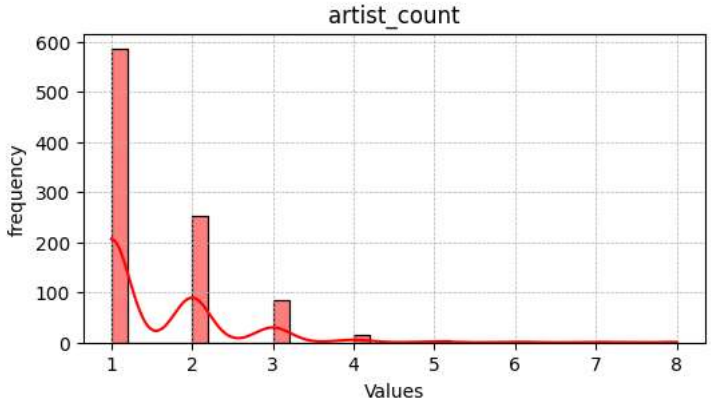
speechiness_%

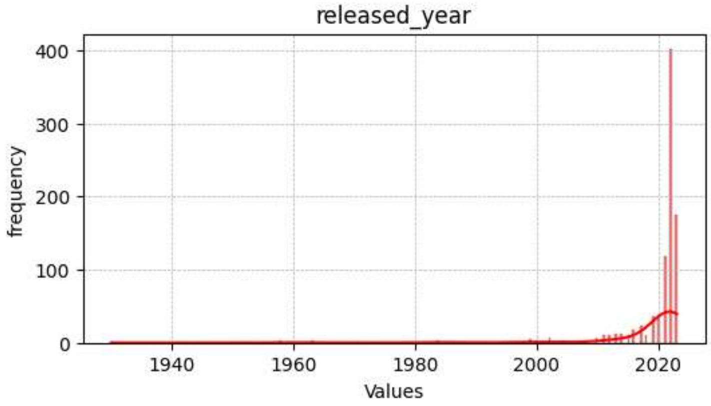
```
# Iterate through each feature in the DataFrame df1
for feature in df1:
  # Create a new figure with a specific size
    plt.figure(figsize=(6,3))
    # Use Seaborn to create a histogram with kernel density estimation (KDE) for the current feature
    sns.histplot(data=df1[feature],kde=True,color='red')
    # Set the title of the histogram to the current feature name
    plt.title(feature)
```

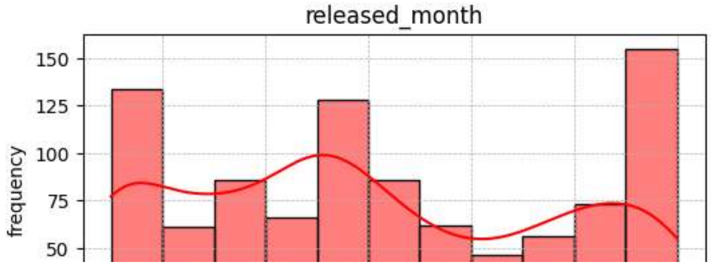
```
# Set the labels for the x-axis and y-axis
plt.xlabel("Values")
plt.ylabel("frequency")
# Add grid lines for better readability
plt.grid(True, which='both', linestyle='--', linewidth=0.5)
# Display the histogram
plt.show()
```

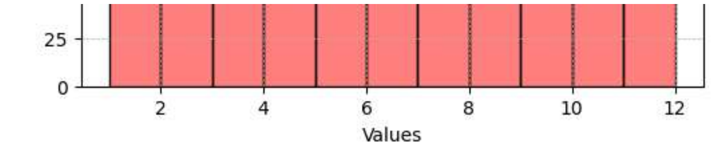

track_name
2.0
1.5
1.0
0.5
0.0
frequency
Values

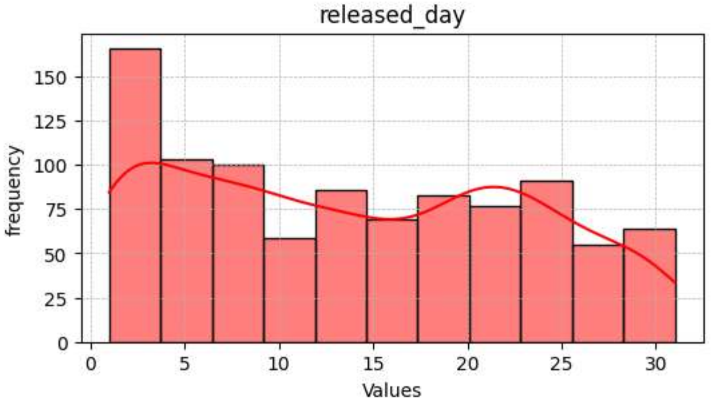

artist(s)_name
35
30
25
20
15
10
5
0
frequency
Values
Topher Ngo, Grayson Vill

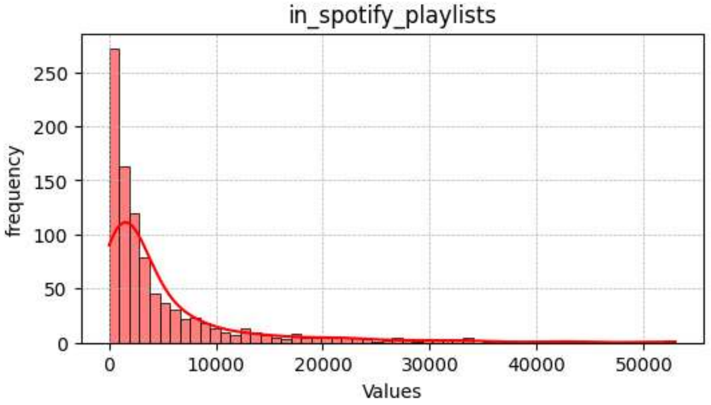

artist_count
600
500
400
300
200
100
0
frequency
1
2
3
4
5
6
7
8
Values

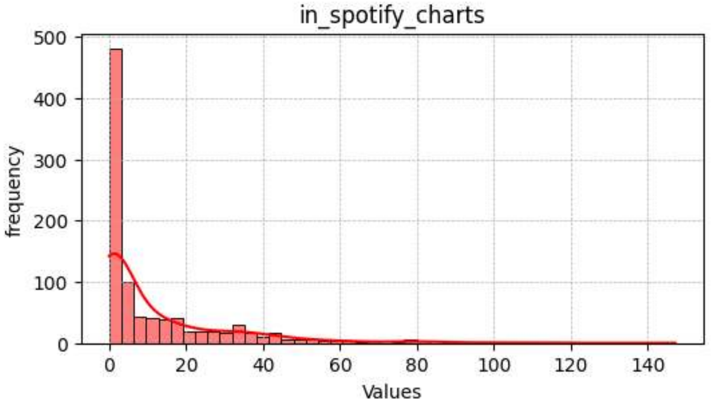

released_year
400
300
200
100
0
frequency
1940
1960
1980
2000
2020
Values

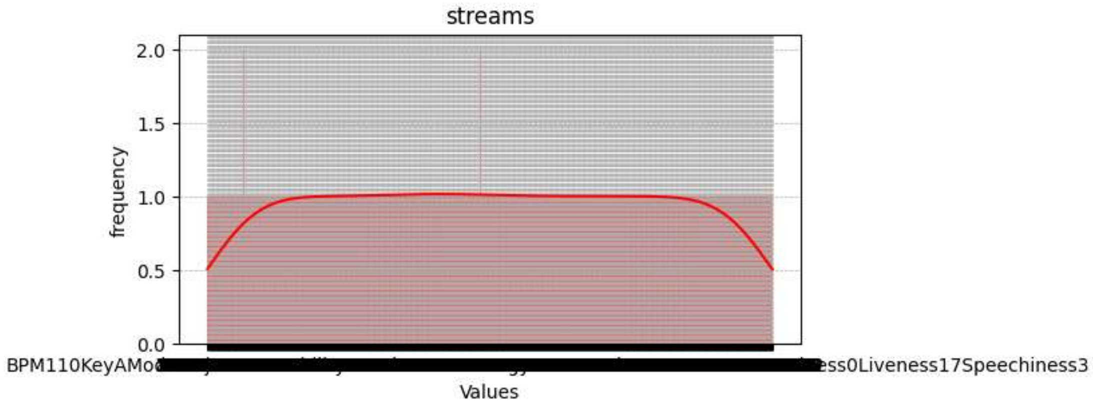

released_month
150
125
100
75
50
frequency

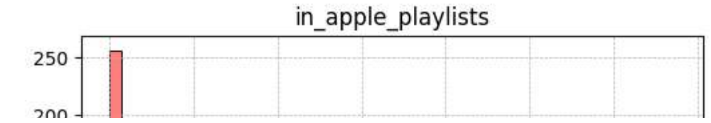

25
0
2
4
6
8
10
12
Values

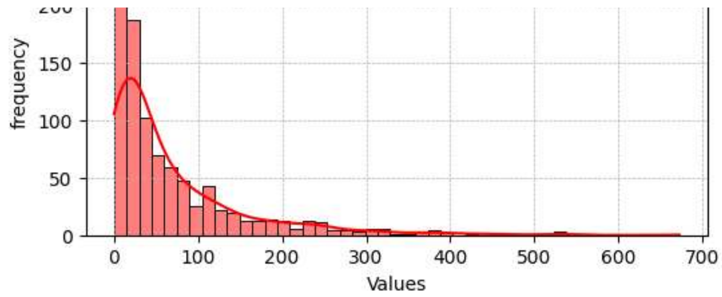

released_day
frequency
150
125
100
75
50
25
0
0
5
10
15
20
25
30
Values


in_spotify_playlists
frequency
250
200
150
100
50
0
0
10000
20000
30000
40000
50000
Values

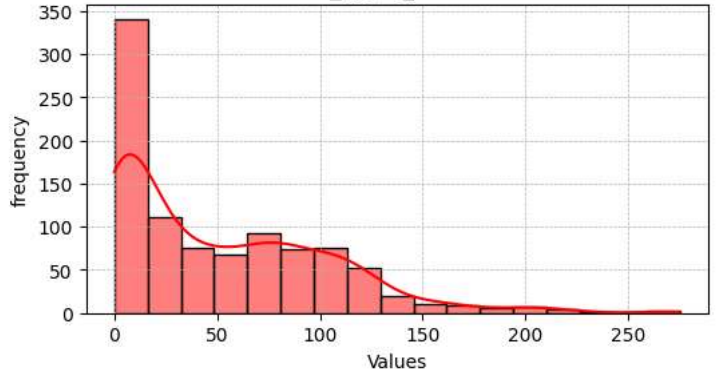

in_spotify_charts
frequency
500
400
300
200
100
0
0
20
40
60
80
100
120
140
Values


streams
frequency
2.0
1.5
1.0
0.5
0.0
BPM110KeyAMod
ess0Liveness17Speechiness3
Values

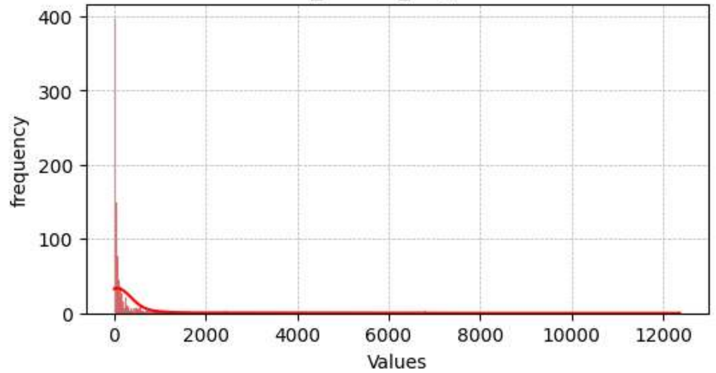

in_apple_playlists
250
200



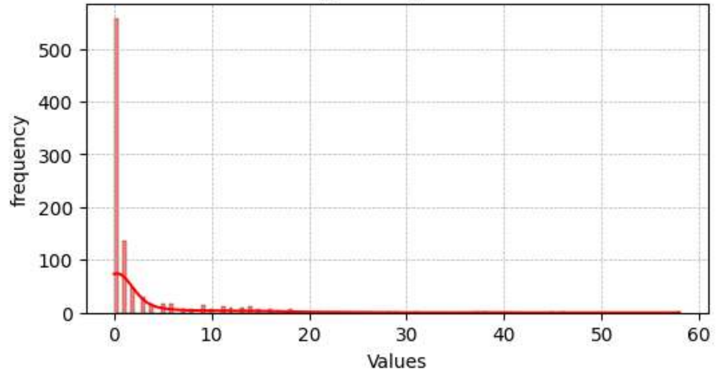
in_apple_charts



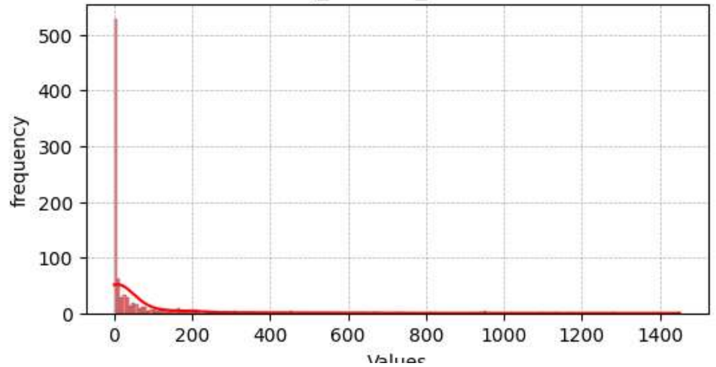
in_deezer_playlists

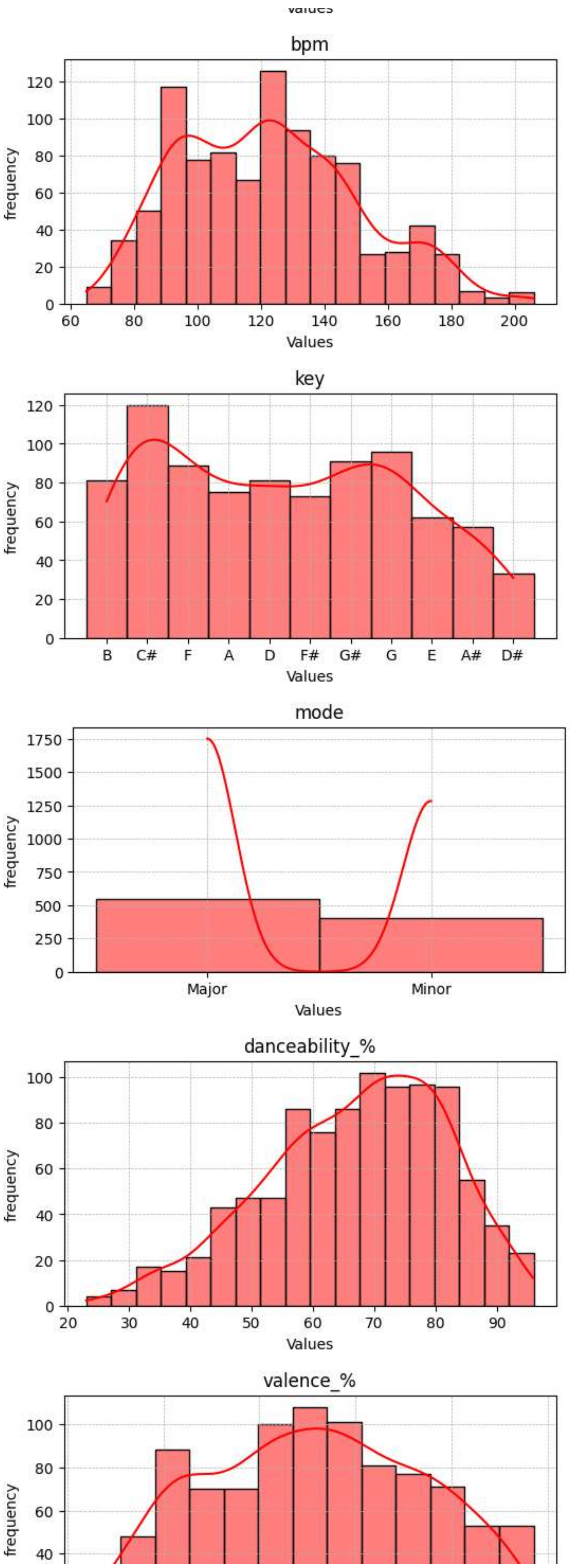


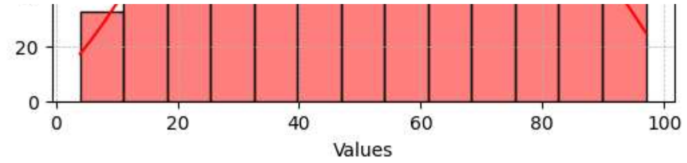
in_deezer_charts

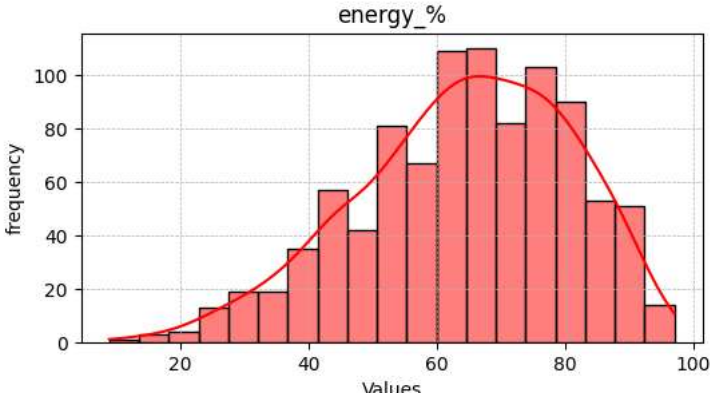


in_shazam_charts



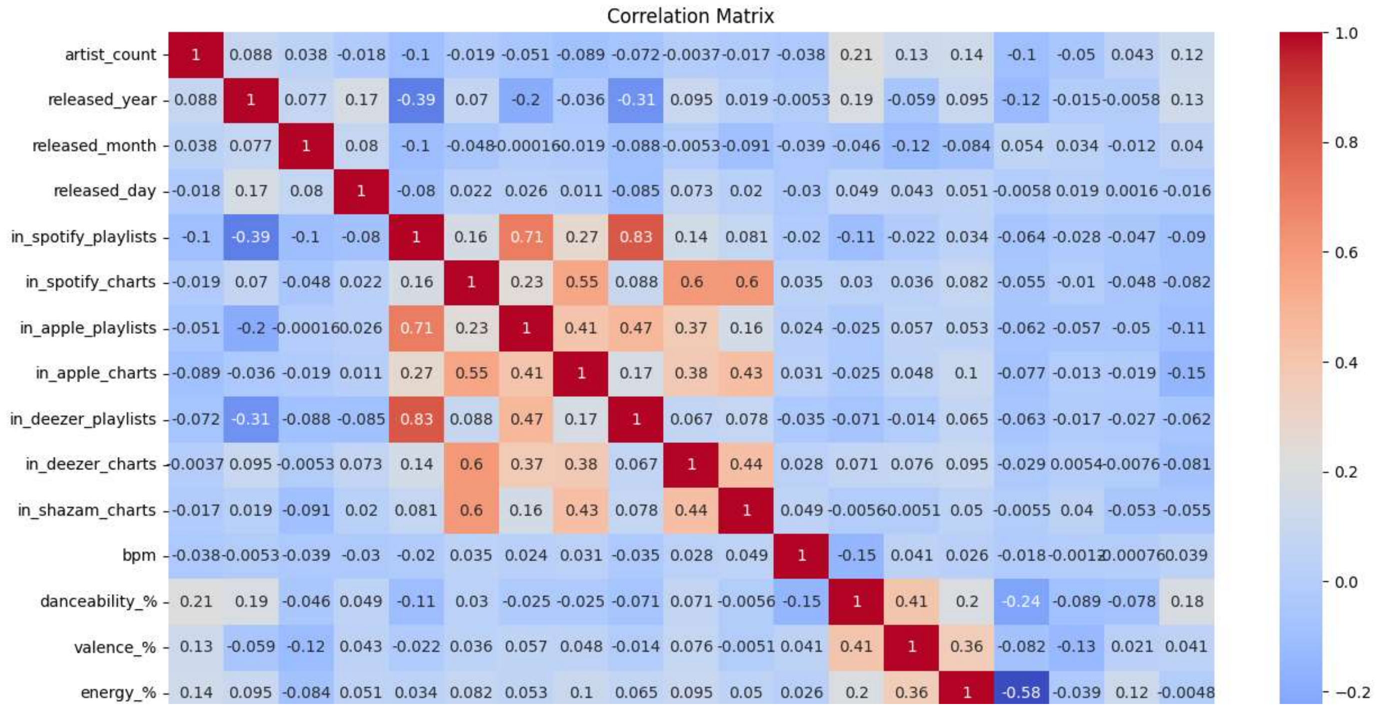

Values
bpm
frequency
0
20
40
60
80
100
120
60
80
100
120
140
160
180
200
Values
key
frequency
0
20
40
60
80
100
120
B
C#
F
A
D
F#
G#
G
E
A#
D#
Values
mode
frequency
0
250
500
750
1000
1250
1500
1750
Major
Minor
Values
danceability_%
frequency
0
20
40
60
80
100
20
30
40
50
60
70
80
90
Values
valence_%
frequency
40
60
80
100





- It looks like most of the songs were create by 1 artist or a group of 2 artists there are very less songs which were created by group more than 2
- It looks like most of the songs in this data are from 2000 there are very less songs in this data which are before 2000
- It looks like most of the songs released in this data are released in between in january-febrauary or in between November-december
- It looks like most of the songs in spotify are there in between 1-10000 playlist after that there are very less songs which we can see where are there in more than 10000 playlists
- The playlist data in apple music is less compared to spotify playlists data and sezzer playlist data
- It looks like all the playlist data are following powerlaw distribution
- It looks like most of the songs in this data are using less than 10 words in their songs
- energy data is very close to fallowing normal distribution

```
# Select numerical columns in the DataFrame df1
numeric_data = df1.select_dtypes(include=[np.number])
# Compute the correlation matrix for the selected numerical columns
correlation_matrix = numeric_data.corr()
# Create a new figure with a specific size
plt.figure(figsize=(15, 10))
# Use Seaborn to create a heatmap of the correlation matrix with annotations
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
# Set the title of the heatmap
plt.title("Correlation Matrix")
# Display the heatmap
plt.show()
```



- All the data related to playlists like spotify playlists ,Apple playlists ,Dezzer playlists are positively corealted which tells that a song which is popular among playlists in one platform is popular in other platform playlists also
- All the data related to ranking like spotify charts,Apple charts ,Dezzer charts ,shazam charts are positively coralated which tells that a song popular in platform is most of the times popular in the other platforms also
- We can see a positive corelation between danceablity,valency,energy also which tells us a song with good valency may have high energy and dancebility value and vice versa
- accousticness and energy are highly negative correalted Which implies songs eith high accousticness have less energy levels in them
- And It looks like accousticness is negatively correalted with many other features in the data
- All the types of playlists data is negatively correlated with released year

```
#timeseries analysis
# List of features to analyze over time
features1 = ['danceability_%', 'valence_%', 'energy_%', 'acousticness_%', 'instrumentalness_%', 'liveness_%', 'speechiness_%']
# Group the data by 'released_year' and calculate the mean for each feature
dp_time_grouped = df1.groupby('released_year')[features1].mean().reset_index()

# Iterate through each feature and create a line plot for its trend over time
for feature in features1:
    plt.figure(figsize=(12, 8))  # Create a new figure with a specific size
     # Use Seaborn to create a line plot for the current feature over the years
    sns.lineplot(data=dp_time_grouped, x='released_year', y=feature)
     # Set the x-axis and y-axis labels
    plt.xlabel("released_year")
    plt.ylabel(feature)
    plt.title(f"trend of {feature} by time")  # Set the title of the line plot
    plt.show() # Display the line plot
```