TASK 1: Exploratory Data Analysis

Yevgen Petronyuk Pidhaynyy 17/10/2024

Spotify most streamed songs 2023

- Analysis and calculation of statistical features

Mean, also known as average is calculated by adding all the values, and then dividing it by the number of values. The mean provides us with a measure of the **central tendency** of a data set, summarizing the data into a single representative value.

$$\text{Mean} = \frac{\text{Sum of all values}}{\text{Number of values}}$$

Here we have the mean for each of the features (numerical features) of the most streamed **Spotify songs in 2023:**

artist_count	*1.556139*
released_year	*2018.238195*
released_month	*6.033578*
released_day	*13.930745*
in_spotify_playlists	*5200.124869*
in_spotify_charts	*12.009444*
in_apple_playlists	*67.812172*
in_apple_charts	*51.908709*
in_deezer_charts	*2.666317*
bpm	*122.540399*
danceability_%	*66.969570*
valence_%	*51.431270*
energy_%	*64.279119*
acousticness_%	*27.057712*
instrumentalness_%	*1.581322*
liveness_%	*18.213012*
speechiness_%	*10.131165*

Here we can see that the mean of the released month is 6. This information is not very useful because the month of release is not a quantitative variable.

On the other hand, the artist count mean (which is 1,55) gives us an average of how many artists sing a song

Median is another measure of central tendency that represents the middle value of a data set when the values are arranged in order. Unlike the mean, the median is not affected by extreme values (outliers), making it a more robust measure in certain situations.

In order to calculate the median we have to arrange the data in ascending or descending order. If the data set has an odd number of values, the median is the middle value. Otherwise, the median is the average of the two middle values.

Here we have the median for each of the features (numerical features) of the most streamed **Spotify songs in 2023:**

artist_count	*1.0*
released_year	*2022.0*
released_month	*6.0*
released_day	*13.0*
in_spotify_playlists	*2224.0*
in_spotify_charts	*3.0*
in_apple_playlists	*34.0*
in_apple_charts	*38.0*
in_deezer_charts	*0.0*
bpm	*121.0*
danceability_%	*69.0*
valence_%	*51.0*
energy_%	*66.0*
acousticness_%	*18.0*
instrumentalness_%	*0.0*
liveness_%	*12.0*
speechiness_%	*6.0*

Here we can see that the artist count median is 1. With this info we can conclude that most of the songs are sang by 1 single artist (there are no songs with 0 artists, therefore most of the values will be 1)

Variance is a measure of dispersion that gives us information about how spread out the values of the data set are. A higher variance indicates that the data points are more spread out, while a lower variance indicates they are closer to the mean.

$$\sigma^2 = \frac{\sum (x_i - \mu)^2}{N}$$

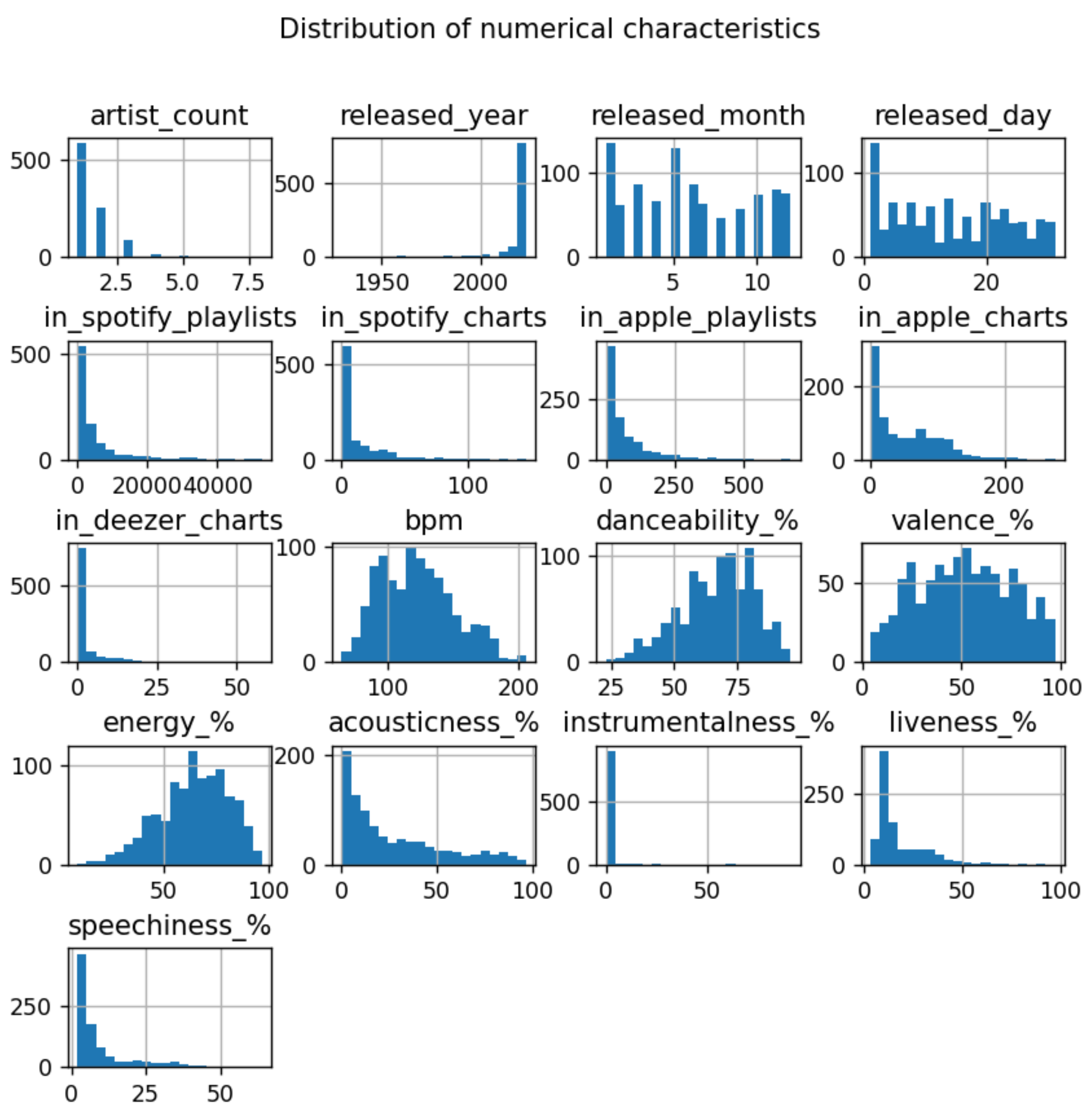
Here we have the median for each of the features (numerical features) of the most streamed **Spotify songs in 2023:**

artist_count	*7.975279 e-01*
released_year	*1.235703 e+02*
released_month	*1.271946 e+01*
released_day	*8.467587 e+01*
in_spotify_playlists	*6.237223 e+07*
in_spotify_charts	*3.832194 e+02*
in_apple_playlists	*7.472132 e+03*
in_apple_charts	*2.563421 e+03*
in_deezer_charts	*3.642845 e+01*
bpm	*7.872402 e+02*
danceability_%	*2.140547 e+02*
valence_%	*5.513401 e+02*
energy_%	*2.739199 e+02*
acousticness_%	*6.757960 e+02*
instrumentalness_%	*7.072473 e+01*
liveness_%	*1.879976 e+02*
speechiness_%	*9.826534 e+01*

The higher the variance is, the more spread out the data is. We can see that there is a very big variance in the spotify playlist variable. This means that there are songs added to very few (or none) playlists, while other have been added to thousands of playlists.

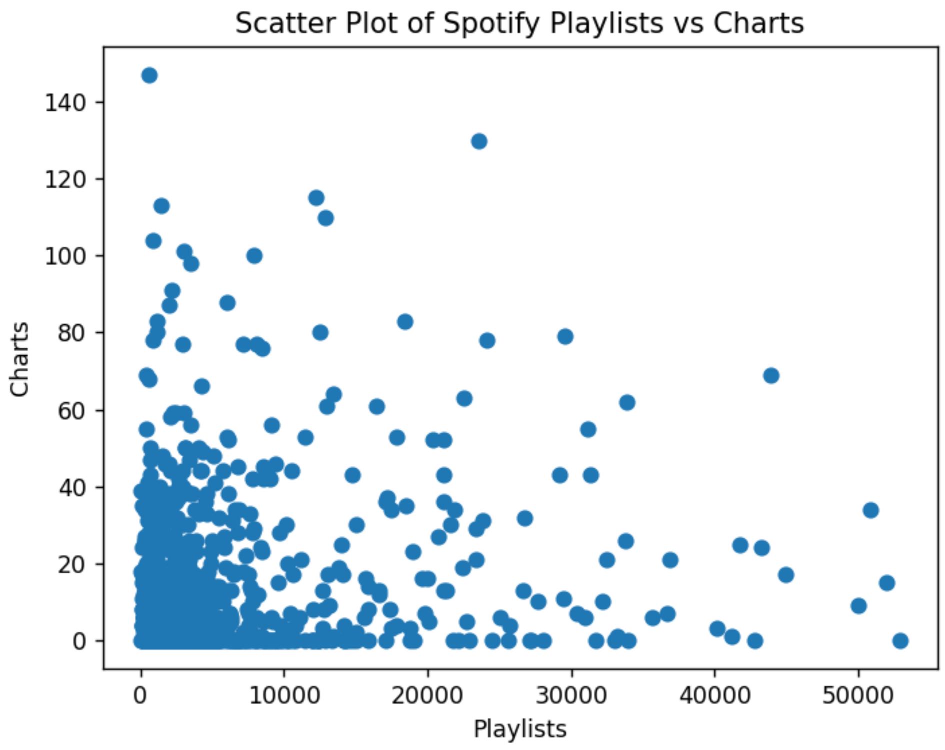
- *Statistic graphical plots*

Histrograms:



Here we can see the histogram for each of the numerical variables we have in the CSV file. We can see that some of them look similar. For example, *in_spotify_playlists* variable has a very similar histogram to the *in_spotify_charts* variable.

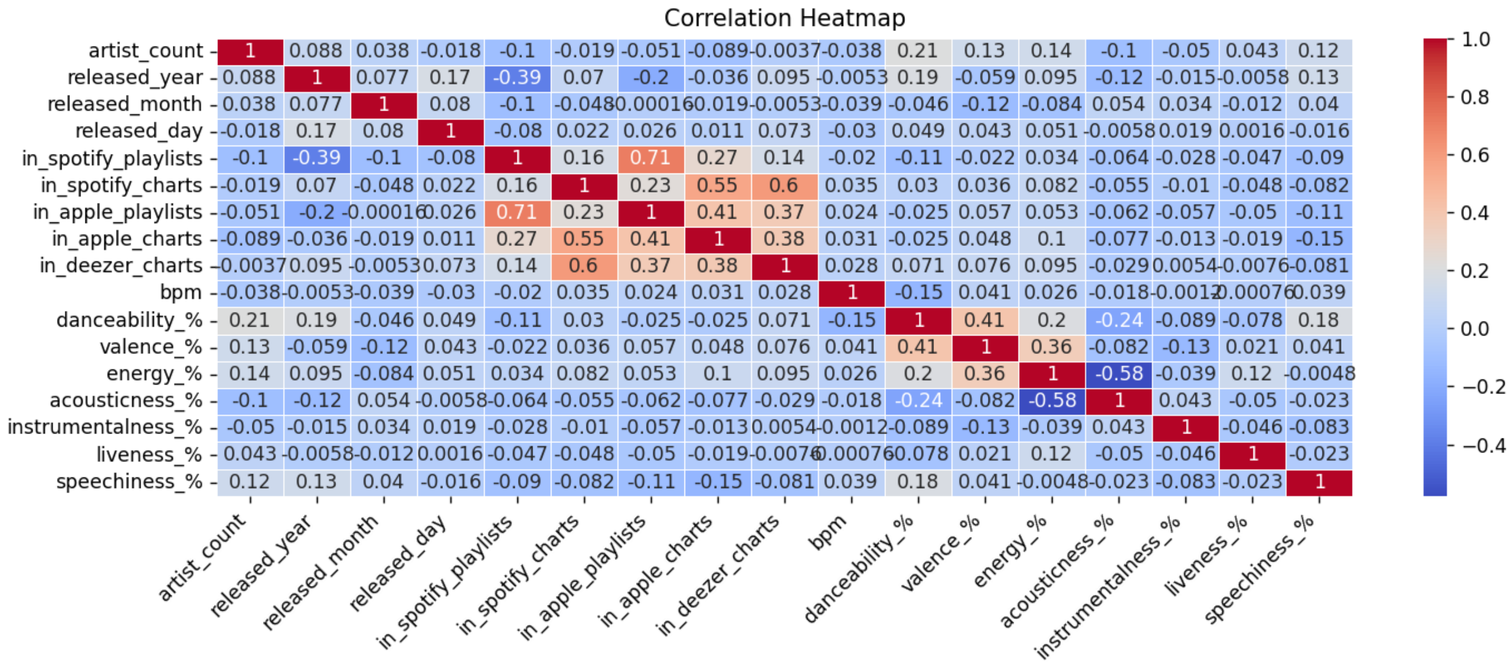
Scatter Plot:



Here we have a scatter plot between spotify playlists variable and spotify charts variable. We can see that the points are distributed down-left. This means that most of the streamed songs don't belong to any (or a few) playlists neither to any (or a few) charts.

We could conclude the same with the variables *in_apple_playlists* and *in_apple_charts*.

Correlation Heatmap:



This heatmap is representing the correlation matrix between the numerical variables we have about the streamed songs. The red – orange values indicate us that both variables are positively correlated (as one grows, the other one does so). On the other hand, dark blue colors indicate negative correlation (as one increases, the other one decreases).

in_apple_playlists and *in_spotify_playlists* are the most positively correlated variables (0,71). *acousticness_%* and *energy_%* are the most negatively correlated variables (-0,58).

- Python code

This is the python program that has extracted the data from the CSV file and calculated every of the features above, including the graphical plots (using libraries such as pandas, matplotlib and seaborn).

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

try:
    # Loads CSV file
    dataframe = pd.read_csv('spotify-2023.csv', encoding = 'latin')

    # Calculates statistics only for numerical columns
    numeric_columns = dataframe.select_dtypes(include=['float64',
'int64']).columns  # Seleccionar solo columnas numéricas

    # Calculates and prints statistics
    mean_values = dataframe[numeric_columns].mean()
    median_values = dataframe[numeric_columns].median()
    variance_values = dataframe[numeric_columns].var()

    print("Mean:\n", mean_values.to_string(), "\n")
    print("Median:\n", median_values.to_string(), "\n")
    print("Variance:\n", variance_values.to_string(), "\n")


    # Histogram of numerical columns
    dataframe.hist(bins=20, figsize=(10, 8))
    plt.suptitle('Distribution of numerical characteristics')

    # Adjusts vertical space between schemes
    plt.subplots_adjust(hspace=0.7)
    plt.show()


    # Calculate covariance matrix for all numerical columns
    cov_matrix = dataframe[numeric_columns].cov()
    print("Covariance Matrix:\n", cov_matrix, "\n")
    # If you want to calculate covariance between two specific columns
    if 'energy_%' in dataframe.columns and 'danceability_%' in
dataframe.columns:
        cov_energy_danceability =
dataframe['energy_%'].cov(dataframe['danceability_%'])
        print(f"Covariance between energy and danceability:
{cov_energy_danceability}\n")

    # Scatter plot between spotify playlists and charts (variables can be
changed as wanted)
    plt.scatter(dataframe['in_spotify_playlists'],
dataframe['in_spotify_charts'])
    plt.title('Scatter Plot of Spotify Playlists vs Charts')
    plt.xlabel('Playlists')
    plt.ylabel('Charts')
    plt.show()

    correlation_matrix = dataframe[numeric_columns].corr()

    # Correlation heatmap
    plt.figure(figsize=(12, 5))
    sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
linewidths=0.5)
    plt.xticks(rotation=45, ha='right')
    plt.title('Correlation Heatmap')
    plt.tight_layout()
    plt.show()


except Exception as e:
    print(f"An error ocurred: {e}")
```

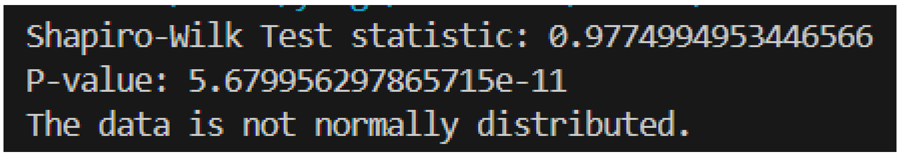
- Statistic tests

Schapiro-Wilk test:

This test is used to know whether a data sample follows a normal distribution or not. This test returns us two values:

- The W-Statistic (between 0 and 1): values closer to 1 suggest that the data sample is closer to a normal distribution.
- The p-value: if p > 0.05, there is no strong evidence against normality, so you fail to reject the null hypothesis (the data could be normally distributed).

Here we have our variable *valence_%* (the one that has the histogram more similar to normal distribution) being tested by the Schapiro-Wilk test:



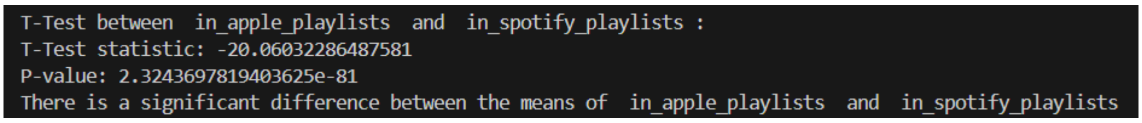
```
Shapiro-Wilk Test statistic: 0.9774994953446566
P-value: 5.679956297865715e-11
The data is not normally distributed.
```

As we see, our data sample is not normally distributed (p-value is almost 0)

T-Test:

The T-Test is used to determine if two variables have significant differences or can be considered to have a similar distribution.

Here we have our variables *in_apple_playlists* and *in_spotify_playlists* being tested:



```
T-Test between  in_apple_playlists  and  in_spotify_playlists :
T-Test statistic: -20.06032286487581
P-value: 2.3243697819403625e-81
There is a significant difference between the means of  in_apple_playlists  and  in_spotify_playlists
```

The T-Test returns us two values:

- The T-Test statistic (in this case -20): if close to 0, it indicates that both variables have similar distributions, in this case they are not.
- The p-value: If bigger than 0,05 then both variables can be considered to have similar distributions. In this case, the p-value is very close to 0, so we reject that hypothesis.

Chi squared Test:

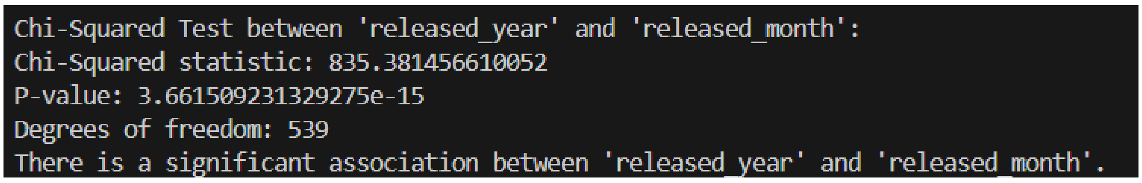
The Chi-Squared test is used for categorical data, to calculate whether two variables have a significant association or not.

The Chi squared Test gives us two values:

- P-value: A Less than 0.05 (common threshold) typically indicates that there is a statistically significant association between the two categorical variables. In this case, you would reject the null hypothesis. A p-value greater than 0.05 suggests that you do not have enough evidence to conclude there is an association, so you fail to reject the null hypothesis.
- Degrees of Freedom: Degrees of freedom are used in determining the critical value of the Chi-Squared distribution for comparison. Higher degrees of freedom can lead to a wider distribution, affecting how we interpret the Chi-Squared statistic.

* We also get the expected frequencies, which are the theoretical counts you would expect for each cell in the contingency table if the null hypothesis were true (i.e., if there were no association between the two categorical variables)

Here we have two variables being tested (*released_year* and *released_month*)



```
Chi-Squared Test between 'released_year' and 'released_month':
Chi-Squared statistic: 835.381456610052
P-value: 3.661509231329275e-15
Degrees of freedom: 539
There is a significant association between 'released_year' and 'released_month'.
```

Here is the code for the three tests:

Schapiro-Wilk & T-Test:

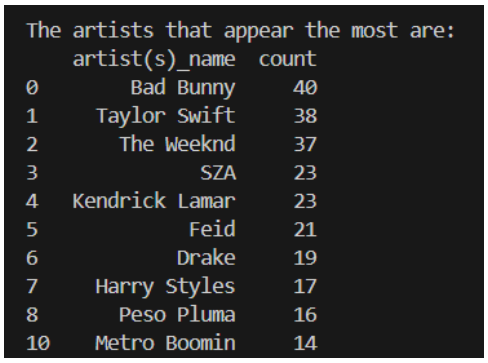
```python
import pandas as pd
from scipy import stats

try:
    # Loads CSV file
    dataframe = pd.read_csv('spotify-2023.csv', encoding = 'latin')


    data1name = "in_apple_playlists";
    data2name = "in_spotify_playlists";


    data1 = dataframe[data1name].dropna()  # dropna() removes any NaN
values
    data2 = dataframe[data2name].dropna()




    # Perform Shapiro-Wilk test
    stat, p_value = stats.shapiro(data1)

    # Display results
    print(f"Shapiro-Wilk Test statistic: {stat}")
    print(f"P-value: {p_value}")

    # Check if the data is normally distributed (typically, p-value >
0.05)
    if p_value > 0.05:
        print("The data is normally distributed.")
    else:
        print("The data is not normally distributed.")





    # Perform two-sample t-test
    t_statistic, t_p_value = stats.ttest_ind(data1, data2)

    # Display T-Test results
    print(f"\nT-Test between ", data1name, " and ", data2name, ":")
    print(f"T-Test statistic: {t_statistic}")
    print(f"P-value: {t_p_value}")

    # Interpret the result
    if t_p_value > 0.05:
        print("There is no significant difference between the means of ",
data1name, " and ", data2name)
    else:
        print("There is a significant difference between the means of ",
data1name, " and ", data2name)







except Exception as e:
    print(f"An error ocurred: {e}")
```

Chi-squared Test:

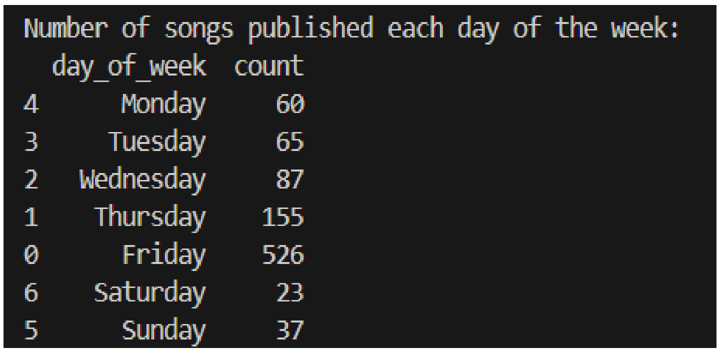
```
import pandas as pd
from scipy import stats

try:
    # Load CSV file
    dataframe = pd.read_csv('spotify-2023.csv', encoding='latin')

    # Display the first few rows of the dataframe (optional)
    print(dataframe.head())

    # Ensure that the relevant columns are treated as categorical
    dataframe['released_year'] = dataframe['released_year'].astype(str)
    dataframe['released_month'] = dataframe['released_month'].astype(str)

    # Create a contingency table for released_year and released_month
    contingency_table = pd.crosstab(dataframe['released_year'],
dataframe['released_month'])

    # Perform Chi-Squared test on the contingency table
    chi2_stat, p_value, dof, expected =
stats.chi2_contingency(contingency_table)

    # Display Chi-Squared Test results
    print(f"\nChi-Squared Test between 'released_year' and
'released_month':")
    print(f"Chi-Squared statistic: {chi2_stat}")
    print(f"P-value: {p_value}")
    print(f"Degrees of freedom: {dof}")
    # Interpret the result
    if p_value > 0.05:
        print("There is no significant association between 
'released_year' and 'released_month'.")
    else:
        print("There is a significant association between 'released_year' 
and 'released_month'.")

except Exception as e:
    print(f"An error occurred: {e}")
```

- Adding new features

I decided to create two small programs that calculate the artists that appear the most in our data sample and how many songs were released each day of the week. Here is the result of each program:



```
The artists that appear the most are:
    artist(s)_name  count
0        Bad Bunny     40
1     Taylor Swift     38
2       The Weeknd     37
3              SZA     23
4   Kendrick Lamar     23
5             Feid     21
6            Drake     19
7     Harry Styles     17
8       Peso Pluma     16
10    Metro Boomin     14
```



```
Number of songs published each day of the week:
  day_of_week  count
4      Monday     60
3     Tuesday     65
2   Wednesday     87
1    Thursday    155
0      Friday    526
6    Saturday     23
5      Sunday     37
```

Here is the code for each program:

```python
import pandas as pd

try:

    dataframe = pd.read_csv('spotify-2023.csv', encoding='latin')


    # We separate different artists
    dataframe['artist(s)_name'] =
dataframe['artist(s)_name'].str.split(', ')
    exploded_df = dataframe.explode('artist(s)_name')

    # Counting each artist apparition
    artist_counts =
exploded_df['artist(s)_name'].value_counts().reset_index()
    artist_counts.columns = ['artist(s)_name', 'count']  # Renombrar
columnas

    # Order each artist by the number of times he appears in our
datasample and only take the first 10
    top_artists = artist_counts.sort_values(by='count',
ascending=False).head(10)

    print("\nThe artists that appear the most are:")
    print(top_artists)

except Exception as e:
    print(f"An error ocurred: {e}")
```

```python
import pandas as pd

try:
    # Cargar el archivo CSV
    dataframe = pd.read_csv('spotify-2023.csv', encoding='latin')  # 
Cambia el nombre del archivo según sea necesario

    # Mostrar las primeras filas del dataframe (opcional)
    print(dataframe.head())


    dataframe['release_date'] = pd.to_datetime(
        dataframe['released_year'].astype(str) + '-' +
        dataframe['released_month'].astype(str) + '-' +
        dataframe['released_day'].astype(str),
        errors='coerce'  # Convierte valores no válidos a NaT
    )

    if dataframe['release_date'].isnull().any():
        print("Not valid date numbers.")

    # Obtaining week day
    dataframe['day_of_week'] = dataframe['release_date'].dt.day_name()

    # Counting songs by week day
    songs_per_day = dataframe['day_of_week'].value_counts().reset_index()
    songs_per_day.columns = ['day_of_week', 'count']  # Renombrar 
columnas

    # Ordering
    days_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 
'Saturday', 'Sunday']
    songs_per_day['day_of_week'] = 
pd.Categorical(songs_per_day['day_of_week'], categories=days_order, 
ordered=True)
    songs_per_day = songs_per_day.sort_values('day_of_week')


    print("\nNumber of songs published each day of the week:")
    print(songs_per_day)


except Exception as e:
    print(f"An error ocurred: {e}")
```

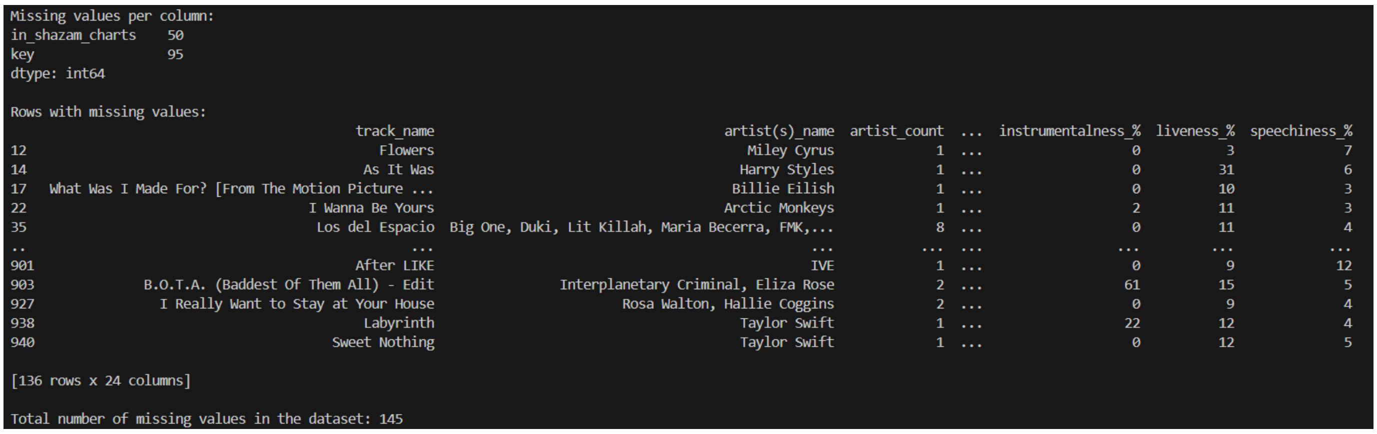
- Missing values

Here is the code for a program that calculates the missing values in the CSV file. Firstly, it calculates the summatory of missing values for each column and prints the ones with more than 0 missing values. Next, it calculates rows that have at least one missing value and prints them.

Finally, it gives us the total number of missing values:

```python
import pandas as pd

# Load the CSV file
df = pd.read_csv('spotify-2023.csv', encoding='latin')

# Check for missing values in each column
missing_per_column = df.isnull().sum()
print("Missing values per column:")
print(missing_per_column[missing_per_column > 0])

# Display rows with any missing values
rows_with_missing_values = df[df.isnull().any(axis=1)]
print("\nRows with missing values:")
print(rows_with_missing_values)

total_missing = df.isnull().sum().sum()
print(f"\nTotal number of missing values in the dataset:
{total_missing}")
```

Result:



```
Missing values per column:
in_shazam_charts    50
key                 95
dtype: int64

Rows with missing values:
                                            track_name                                     artist(s)_name  artist_count  ...  instrumentalness_%  liveness_%  speechiness_%
12                                             Flowers                                        Miley Cyrus             1  ...                   0           3              7
14                                           As It Was                                       Harry Styles             1  ...                   0          31              6
17   What Was I Made For? [From The Motion Picture ...                                      Billie Eilish             1  ...                   0          10              3
22                                    I Wanna Be Yours                                     Arctic Monkeys             1  ...                   2          11              3
35                                     Los del Espacio  Big One, Duki, Lit Killah, Maria Becerra, FMK,...             8  ...                   0          11              4
..                                                 ...                                                ...           ...  ...                 ...         ...            ...
901                                         After LIKE                                                IVE             1  ...                   0           9             12
903              B.O.T.A. (Baddest Of Them All) - Edit                Interplanetary Criminal, Eliza Rose             2  ...                  61          15              5
927                I Really Want to Stay at Your House                        Rosa Walton, Hallie Coggins             2  ...                   0           9              4
938                                          Labyrinth                                       Taylor Swift             1  ...                  22          12              4
940                                      Sweet Nothing                                       Taylor Swift             1  ...                   0          12              5

[136 rows x 24 columns]

Total number of missing values in the dataset: 145
```