Warnings

```
import warnings
warnings.filterwarnings('ignore')
```

Libraries Used

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
from scipy.stats import skew
from scipy.stats import shapiro
```

Importing Dataset

```
d= pd.read_csv('Spotify_Song_Attributes.csv')
d.head()

                                                trackName  \
0                                                "Honest"
1  "In The Hall Of The Mountain King" from Peer G...
2                                      #BrooklynBloodPop!
3                                                     $10
4                               (I Just) Died In Your Arms

                   artistName  msPlayed              genre
danceability  \
0                Nico Collins    191772                NaN
0.476
1  London Symphony Orchestra   1806234  british orchestra
0.475
2                        SyKo    145610         glitchcore
0.691
3                Good Morning     25058   experimental pop
0.624
4                Cutting Crew   5504949         album rock
0.625

   energy   key  loudness  mode  speechiness  ...  liveness  valence
tempo  \
0   0.799   4.0    -4.939   0.0       0.2120  ...    0.2570    0.577
162.139
```

```
1   0.130   7.0   -17.719   1.0       0.0510  ...    0.1010    0.122
112.241
2   0.814   1.0    -3.788   0.0       0.1170  ...    0.3660    0.509
132.012
3   0.596   4.0    -9.804   1.0       0.0314  ...    0.1190    0.896
120.969
4   0.726  11.0   -11.402   0.0       0.0444  ...    0.0625    0.507
124.945

             type                      id  \
0  audio_features  7dTxqsaFGHOXwtzHINjfHv
1  audio_features  14Qcrx6Dfjvcj0H8oV8oUW
2  audio_features  7K9Z3yFNNLv5kwTjQYGjnu
3  audio_features  3koAwrM1RO0TGMeQJ3qt9J
4  audio_features  4ByEFOBuLXpCqvO1kw8Wdm

                                    uri  \
0  spotify:track:7dTxqsaFGHOXwtzHINjfHv
1  spotify:track:14Qcrx6Dfjvcj0H8oV8oUW
2  spotify:track:7K9Z3yFNNLv5kwTjQYGjnu
3  spotify:track:3koAwrM1RO0TGMeQJ3qt9J
4  spotify:track:4ByEFOBuLXpCqvO1kw8Wdm

                                          track_href  \
0  https://api.spotify.com/v1/tracks/7dTxqsaFGHOX...
1  https://api.spotify.com/v1/tracks/14Qcrx6Dfjvc...
2  https://api.spotify.com/v1/tracks/7K9Z3yFNNLv5...
3  https://api.spotify.com/v1/tracks/3koAwrM1RO0T...
4  https://api.spotify.com/v1/tracks/4ByEFOBuLXpC...

                                        analysis_url duration_ms  \
0  https://api.spotify.com/v1/audio-analysis/7dTx...    191948.0
1  https://api.spotify.com/v1/audio-analysis/14Qc...    150827.0
2  https://api.spotify.com/v1/audio-analysis/7K9Z...    145611.0
3  https://api.spotify.com/v1/audio-analysis/3koA...     89509.0
4  https://api.spotify.com/v1/audio-analysis/4ByE...    280400.0

  time_signature
0            4.0
1            4.0
2            4.0
3            4.0
4            4.0

[5 rows x 22 columns]
```

```
d.shape
```

```
(10080, 22)
```

```
d.describe()
```

```
           msPlayed  danceability       energy          key
loudness  \
count  1.008000e+04   9530.000000  9530.000000  9530.000000
9530.000000
mean   1.519657e+06      0.602469     0.563524     5.241973    -
8.685077
std    5.317343e+06      0.157745     0.243548     3.570615
5.414814
min    0.000000e+00      0.000000     0.001080     0.000000   -
42.044000
25%    1.367800e+05      0.509000     0.403000     2.000000   -
10.189000
50%    2.662875e+05      0.623000     0.589000     5.000000    -
7.218000
75%    1.186307e+06      0.714000     0.751000     8.000000    -
5.336000
max    1.583671e+08      0.976000     0.999000    11.000000
3.010000

              mode  speechiness  acousticness  instrumentalness
liveness  \
count  9530.000000  9530.000000   9530.000000       9530.000000
9530.000000
mean      0.612382     0.078468      0.362924          0.153215
0.174589
std       0.487232     0.080101      0.334337          0.313132
0.130749
min       0.000000     0.000000      0.000002          0.000000
0.024900
25%       0.000000     0.036100      0.053800          0.000000
0.096200
50%       1.000000     0.047900      0.245000          0.000025
0.119000
75%       1.000000     0.081900      0.668000          0.027600
0.209000
max       1.000000     0.966000      0.996000          0.993000
0.964000

           valence        tempo   duration_ms  time_signature
count  9530.000000  9530.000000  9.530000e+03     9530.000000
mean      0.434113   119.374474  2.029311e+05        3.917524
std       0.242761    28.993087  9.587253e+04        0.386189
min       0.000000     0.000000  1.002700e+04        0.000000
25%       0.237000    97.568000  1.616970e+05        4.000000
50%       0.409000   119.822000  1.942860e+05        4.000000
75%       0.614000   139.785000  2.295260e+05        4.000000
max       0.986000   236.196000  4.581483e+06        5.000000
```

Finding NULL Values and removing it

```
missing_values = d.isnull().sum()
missing_percentage = (missing_values / len(d)) * 100
missing_info = pd.DataFrame({'Missing Values': missing_values,
'Percentage': missing_percentage})
print(missing_info)
```

```
                  Missing Values  Percentage
trackName                      0    0.000000
artistName                     0    0.000000
msPlayed                       0    0.000000
genre                       1500   14.880952
danceability                 550    5.456349
energy                       550    5.456349
key                          550    5.456349
loudness                     550    5.456349
mode                         550    5.456349
speechiness                  550    5.456349
acousticness                 550    5.456349
instrumentalness             550    5.456349
liveness                     550    5.456349
valence                      550    5.456349
tempo                        550    5.456349
type                         550    5.456349
id                           550    5.456349
uri                          550    5.456349
track_href                   550    5.456349
analysis_url                 550    5.456349
duration_ms                  550    5.456349
time_signature               550    5.456349
```

```
d.isnull().sum()
```

```
trackName              0
artistName             0
msPlayed               0
genre               1500
danceability         550
energy               550
key                  550
loudness             550
mode                 550
speechiness          550
acousticness         550
instrumentalness     550
liveness             550
valence              550
tempo                550
type                 550
```

```
id                     550
uri                    550
track_href             550
analysis_url           550
duration_ms            550
time_signature         550
dtype: int64
```

```
df_cleaned = d.copy()

df_cleaned.dropna(inplace=True)
missing_values = df_cleaned.isnull().sum()
print(missing_values)
```

```
trackName           0
artistName          0
msPlayed            0
genre               0
danceability        0
energy              0
key                 0
loudness            0
mode                0
speechiness         0
acousticness        0
instrumentalness    0
liveness            0
valence             0
tempo               0
type                0
id                  0
uri                 0
track_href          0
analysis_url        0
duration_ms         0
time_signature      0
dtype: int64
```

Finding Duplicate values and removing it

```
duplicate_counts =
df_cleaned[df_cleaned.duplicated(keep=False)].groupby(list(d.columns))
.size().reset_index(name='Count')

# Display the duplicate rows and their counts
print(duplicate_counts)
```

```
                                                trackName  \
0     "In The Hall Of The Mountain King" from Peer G...
1                                      #BrooklynBloodPop!
2                                                     $10
3                                (I Just) Died In Your Arms
4                                           (L)only Child
...                                                   ...
4285                                                  色香水
4286                                                   落日
4287                                                   薄暮
4288                                                記念撮影
4289                                                電光石火

                    artistName   msPlayed                genre
danceability  \
0     London Symphony Orchestra    1806234  british orchestra
0.475
1                          SyKo     145610          glitchcore
0.691
2                  Good Morning      25058   experimental pop
0.624
3                  Cutting Crew    5504949          album rock
0.625
4                   salem ilese    2237969               alt z
0.645
...                         ...        ...                 ...
...
4285               Yoh kamiyama  109652914  japanese teen pop
0.613
4286         Yasuharu Takanashi      12444               anime
0.276
4287         Yasuharu Takanashi      20041               anime
0.150
4288             BUMP OF CHICKEN     437806               j-pop
0.615
4289         Yasuharu Takanashi       8682               anime
0.487

      energy   key  loudness  mode  speechiness  ...  valence    tempo
\
0      0.130   7.0   -17.719   1.0       0.0510  ...   0.1220  112.241

1      0.814   1.0    -3.788   0.0       0.1170  ...   0.5090  132.012

2      0.596   4.0    -9.804   1.0       0.0314  ...   0.8960  120.969

3      0.726  11.0   -11.402   0.0       0.0444  ...   0.5070  124.945

4      0.611   8.0    -5.925   0.0       0.1370  ...   0.6450  157.475
```

```
...      ...   ...       ...   ...          ...  ...      ...      ...
4285   0.857   7.0    -5.272   1.0       0.0271  ...   0.6980  147.974
4286   0.177   2.0   -16.791   1.0       0.0336  ...   0.0394  100.160
4287   0.194   4.0   -15.769   0.0       0.0381  ...   0.1130   93.858
4288   0.738   9.0    -6.883   1.0       0.0291  ...   0.6700  119.924
4289   0.851   0.0   -10.867   1.0       0.0569  ...   0.2760  145.019

                type                      id  \
0     audio_features  14Qcrx6Dfjvcj0H8oV8oUW
1     audio_features  7K9Z3yFNNLv5kwTjQYGjnu
2     audio_features  3koAwrM1RO0TGMeQJ3qt9J
3     audio_features  4ByEFOBuLXpCqvO1kw8Wdm
4     audio_features  22lJaG2yxlSjIwdUIddcFk
...              ...                     ...
4285  audio_features  7FpABRyv5TaZz0llkhjPgc
4286  audio_features  6mktMBAkKKGF9ZHRfn7Ohc
4287  audio_features  0QI75NHBAVISJRrlsujmP7
4288  audio_features  1pfZ0POxTGl7JbQyCdst5Q
4289  audio_features  0vEusE17jAgbBul1lR1o3Y

                                       uri  \
0     spotify:track:14Qcrx6Dfjvcj0H8oV8oUW
1     spotify:track:7K9Z3yFNNLv5kwTjQYGjnu
2     spotify:track:3koAwrM1RO0TGMeQJ3qt9J
3     spotify:track:4ByEFOBuLXpCqvO1kw8Wdm
4     spotify:track:22lJaG2yxlSjIwdUIddcFk
...                                    ...
4285  spotify:track:7FpABRyv5TaZz0llkhjPgc
4286  spotify:track:6mktMBAkKKGF9ZHRfn7Ohc
4287  spotify:track:0QI75NHBAVISJRrlsujmP7
4288  spotify:track:1pfZ0POxTGl7JbQyCdst5Q
4289  spotify:track:0vEusE17jAgbBul1lR1o3Y

                                             track_href  \
0     https://api.spotify.com/v1/tracks/14Qcrx6Dfjvc...
1     https://api.spotify.com/v1/tracks/7K9Z3yFNNLv5...
2     https://api.spotify.com/v1/tracks/3koAwrM1RO0T...
3     https://api.spotify.com/v1/tracks/4ByEFOBuLXpC...
4     https://api.spotify.com/v1/tracks/22lJaG2yxlSj...
...                                                 ...
4285  https://api.spotify.com/v1/tracks/7FpABRyv5TaZ...
4286  https://api.spotify.com/v1/tracks/6mktMBAkKKGF...
4287  https://api.spotify.com/v1/tracks/0QI75NHBAVIS...
4288  https://api.spotify.com/v1/tracks/1pfZ0POxTGl7...
```

```
4289  https://api.spotify.com/v1/tracks/0vEusE17jAgb...

                                         analysis_url duration_ms  \
0     https://api.spotify.com/v1/audio-analysis/14Qc...    150827.0
1     https://api.spotify.com/v1/audio-analysis/7K9Z...    145611.0
2     https://api.spotify.com/v1/audio-analysis/3koA...     89509.0
3     https://api.spotify.com/v1/audio-analysis/4ByE...    280400.0
4     https://api.spotify.com/v1/audio-analysis/22lJ...    144468.0
...                                               ...         ...
4285  https://api.spotify.com/v1/audio-analysis/7FpA...    250792.0
4286  https://api.spotify.com/v1/audio-analysis/6mkt...    115400.0
4287  https://api.spotify.com/v1/audio-analysis/0QI7...    100507.0
4288  https://api.spotify.com/v1/audio-analysis/1pfZ...    282563.0
4289  https://api.spotify.com/v1/audio-analysis/0vEu...    136120.0

     time_signature Count
0               4.0     2
1               4.0     2
2               4.0     2
3               4.0     2
4               3.0     2
...             ...   ...
4285            4.0     2
4286            4.0     2
4287            4.0     2
4288            4.0     2
4289            3.0     2

[4290 rows x 23 columns]
```

```python
total_rows_before = len(d)

# Keep only the first occurrence of each row
df_no_duplicates = df_cleaned.drop_duplicates()

# Count the number of rows after removing duplicates
total_rows_after = len(df_no_duplicates)

# Calculate the number of duplicate rows deleted
duplicate_rows_deleted = total_rows_before - total_rows_after

# Display the DataFrame with duplicates removed
print("Original DataFrame rows:", total_rows_before)
print("Rows after removing duplicates:", total_rows_after)
print("Duplicate rows deleted:", duplicate_rows_deleted)
#print(df_no_duplicates)
df_cleaned = df_no_duplicates.copy()
```

```
Original DataFrame rows: 10080
Rows after removing duplicates: 4290
Duplicate rows deleted: 5790
```

```
d=df_cleaned
d.shape
```

```
(4290, 22)
```

Outliers

```
# Function to identify outliers in a DataFrame
def find_outliers(data_frame):
    numerical_columns = df_cleaned.select_dtypes(include=[np.number])

    # Calculate quartiles
    Q1 = numerical_columns.quantile(0.25)
    Q3 = numerical_columns.quantile(0.75)

    # Calculate IQR
    IQR = Q3 - Q1

    # Define lower and upper bounds for outliers
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Identify outliers
    outliers = ((numerical_columns < lower_bound) | (numerical_columns
> upper_bound)).any(axis=1)

    return df_cleaned[outliers]

# Detect and print outlier records
outlier_records = find_outliers(d)
print("Outlier records:")
print(outlier_records)
```

```
Outlier records:
                                              trackName  \
1     "In The Hall Of The Mountain King" from Peer G...
3                                                   $10
4                              (I Just) Died In Your Arms
5                                           (L)only Child
6                                                   (lol)
...                                                   ...
5032                                           Youngblood
5034                                              Younger
5035                                   Younger with Time.
5037                     Your Love Is My Drug (8 Bit Slowed)
```

```
5039                            Your Voice / Bethel, NY

                     artistName  msPlayed              genre
danceability  \
1     London Symphony Orchestra   1806234  british orchestra
0.475
3                  Good Morning     25058   experimental pop
0.624
4                  Cutting Crew   5504949         album rock
0.625
5                   salem ilese   2237969              alt z
0.645
6                  Eren Cannata    441335        guitar case
0.663
...                         ...       ...                ...
...
5032                  Arc North   1009796         slap house
0.465
5034                       Ruel   5272303              alt z
0.745
5035                  Ben Zaidi    668478           folk-pop
0.537
5037                just valery     97600          sad lo-fi
0.282
5039                      Jaden    213626            pop rap
0.560

      energy   key  loudness  mode  speechiness  ...  liveness
valence  \
1      0.130   7.0   -17.719   1.0       0.0510  ...    0.1010
0.122
3      0.596   4.0    -9.804   1.0       0.0314  ...    0.1190
0.896
4      0.726  11.0   -11.402   0.0       0.0444  ...    0.0625
0.507
5      0.611   8.0    -5.925   0.0       0.1370  ...    0.2370
0.645
6      0.904   7.0    -4.710   1.0       0.0857  ...    0.3410
0.675
...      ...   ...       ...   ...          ...  ...       ...       ..
.
5032   0.891   7.0    -2.239   0.0       0.3680  ...    0.2400
0.407
5034   0.477  11.0    -7.706   0.0       0.0880  ...    0.1200
0.454
5035   0.143   2.0   -16.992   1.0       0.0331  ...    0.1100
0.245
5037   0.158   6.0    -7.783   1.0       0.0311  ...    0.4740
0.248
```

```
5039   0.344   3.0   -12.283   1.0        0.0306  ...    0.1110
0.428

        tempo            type                      id  \
1     112.241  audio_features  14Qcrx6Dfjvcj0H8oV8oUW
3     120.969  audio_features  3koAwrM1RO0TGMeQJ3qt9J
4     124.945  audio_features  4ByEFOBuLXpCqvO1kw8Wdm
5     157.475  audio_features  22lJaG2yxlSjIwdUIddcFk
6     118.024  audio_features  4DS2UXeR2V5W7R9aype6t1
...       ...             ...                     ...
5032  123.466  audio_features  6nLEcwqpVNOGRGtnV2pElc
5034  136.055  audio_features  2qXicQG06oT0ijKBznpgQv
5035  131.118  audio_features  6o8pM5reLgjd5i8gDY3Irt
5037   65.152  audio_features  1EoThnDm6kQfB2idIfR30n
5039  115.393  audio_features  3BcN2Pcy0kTG1zm8Tz9MsB

                                       uri  \
1     spotify:track:14Qcrx6Dfjvcj0H8oV8oUW
3     spotify:track:3koAwrM1RO0TGMeQJ3qt9J
4     spotify:track:4ByEFOBuLXpCqvO1kw8Wdm
5     spotify:track:22lJaG2yxlSjIwdUIddcFk
6     spotify:track:4DS2UXeR2V5W7R9aype6t1
...                                    ...
5032  spotify:track:6nLEcwqpVNOGRGtnV2pElc
5034  spotify:track:2qXicQG06oT0ijKBznpgQv
5035  spotify:track:6o8pM5reLgjd5i8gDY3Irt
5037  spotify:track:1EoThnDm6kQfB2idIfR30n
5039  spotify:track:3BcN2Pcy0kTG1zm8Tz9MsB

                                             track_href  \
1     https://api.spotify.com/v1/tracks/14Qcrx6Dfjvc...
3     https://api.spotify.com/v1/tracks/3koAwrM1RO0T...
4     https://api.spotify.com/v1/tracks/4ByEFOBuLXpC...
5     https://api.spotify.com/v1/tracks/22lJaG2yxlSj...
6     https://api.spotify.com/v1/tracks/4DS2UXeR2V5W...
...                                                 ...
5032  https://api.spotify.com/v1/tracks/6nLEcwqpVNOG...
5034  https://api.spotify.com/v1/tracks/2qXicQG06oT0...
5035  https://api.spotify.com/v1/tracks/6o8pM5reLgjd...
5037  https://api.spotify.com/v1/tracks/1EoThnDm6kQf...
5039  https://api.spotify.com/v1/tracks/3BcN2Pcy0kTG...

                                           analysis_url duration_ms  \
1     https://api.spotify.com/v1/audio-analysis/14Qc...    150827.0
3     https://api.spotify.com/v1/audio-analysis/3koA...     89509.0
4     https://api.spotify.com/v1/audio-analysis/4ByE...    280400.0
5     https://api.spotify.com/v1/audio-analysis/22lJ...    144468.0
6     https://api.spotify.com/v1/audio-analysis/4DS2...    217627.0
...                                                 ...         ...
5032  https://api.spotify.com/v1/audio-analysis/6nLE...    188317.0
```

```
5034  https://api.spotify.com/v1/audio-analysis/2qXi...    222320.0
5035  https://api.spotify.com/v1/audio-analysis/6o8p...    222827.0
5037  https://api.spotify.com/v1/audio-analysis/1EoT...    112582.0
5039  https://api.spotify.com/v1/audio-analysis/3BcN...    213627.0

     time_signature
1               4.0
3               4.0
4               4.0
5               3.0
6               4.0
...             ...
5032            4.0
5034            4.0
5035            3.0
5037            4.0
5039            3.0

[2285 rows x 22 columns]
```

Dataset after cleaning

```
d.shape
```

```
(4290, 22)
```

```
d.describe()
```

```
           msPlayed  danceability       energy          key
loudness  \
count  4.290000e+03   4290.000000  4290.000000  4290.000000
4290.000000
mean   1.535919e+06      0.601829     0.566844     5.246387    -
8.577849
std    5.559026e+06      0.158520     0.241667     3.574820
5.329025
min    0.000000e+00      0.000000     0.001080     0.000000   -
42.044000
25%    1.398910e+05      0.508000     0.407250     2.000000   -
10.008750
50%    2.699110e+05      0.623000     0.592000     5.000000    -
7.129000
75%    1.214759e+06      0.714000     0.753000     8.000000    -
5.312250
max    1.583671e+08      0.976000     0.999000    11.000000
3.010000

              mode  speechiness  acousticness  instrumentalness
```

```
liveness  \
count  4290.000000  4290.000000   4290.000000       4290.000000
4290.000000
mean      0.616317     0.078321      0.357838          0.149342
0.174668
std       0.486339     0.078540      0.332788          0.309789
0.130909
min       0.000000     0.000000      0.000002          0.000000
0.024900
25%       0.000000     0.036200      0.051825          0.000000
0.096125
50%       1.000000     0.048000      0.240000          0.000024
0.120000
75%       1.000000     0.081900      0.657000          0.023500
0.209000
max       1.000000     0.941000      0.996000          0.993000
0.964000

           valence        tempo   duration_ms  time_signature
count  4290.000000  4290.00000  4.290000e+03     4290.000000
mean      0.435603   119.14435  2.037884e+05        3.914452
std       0.242780    28.96881  7.340056e+04        0.391565
min       0.000000     0.00000  1.002700e+04        0.000000
25%       0.238250    97.23800  1.629720e+05        4.000000
50%       0.410000   118.97600  1.959360e+05        4.000000
75%       0.618000   139.47000  2.311830e+05        4.000000
max       0.986000   236.19600  1.847210e+06        5.000000
```

Attributes:

- trackName: The name of the track.
- artistName: The name of the artist or band associated with the track.
- msPlayed: The duration in milliseconds that the track was played.
- genre: The genre or genres associated with the track.
- danceability: A measure of how suitable a track is for dancing.
- energy: The energy level of the track.
- key: The key of the track (e.g., C, D, E).
- loudness: The overall loudness of the track in decibels (dB).
- mode: The modality of the track (1 = major, 0 = minor).
- speechiness: The presence of spoken words in the track.
- acousticness: The acousticness of the track.
- instrumentalness: The probability of the track being instrumental.
- liveness: A measure of the presence of a live audience in the track.
- valence: The musical positiveness or happiness conveyed by the track.
- tempo: The tempo of the track in beats per minute (BPM).
- type: The type of the Spotify track.

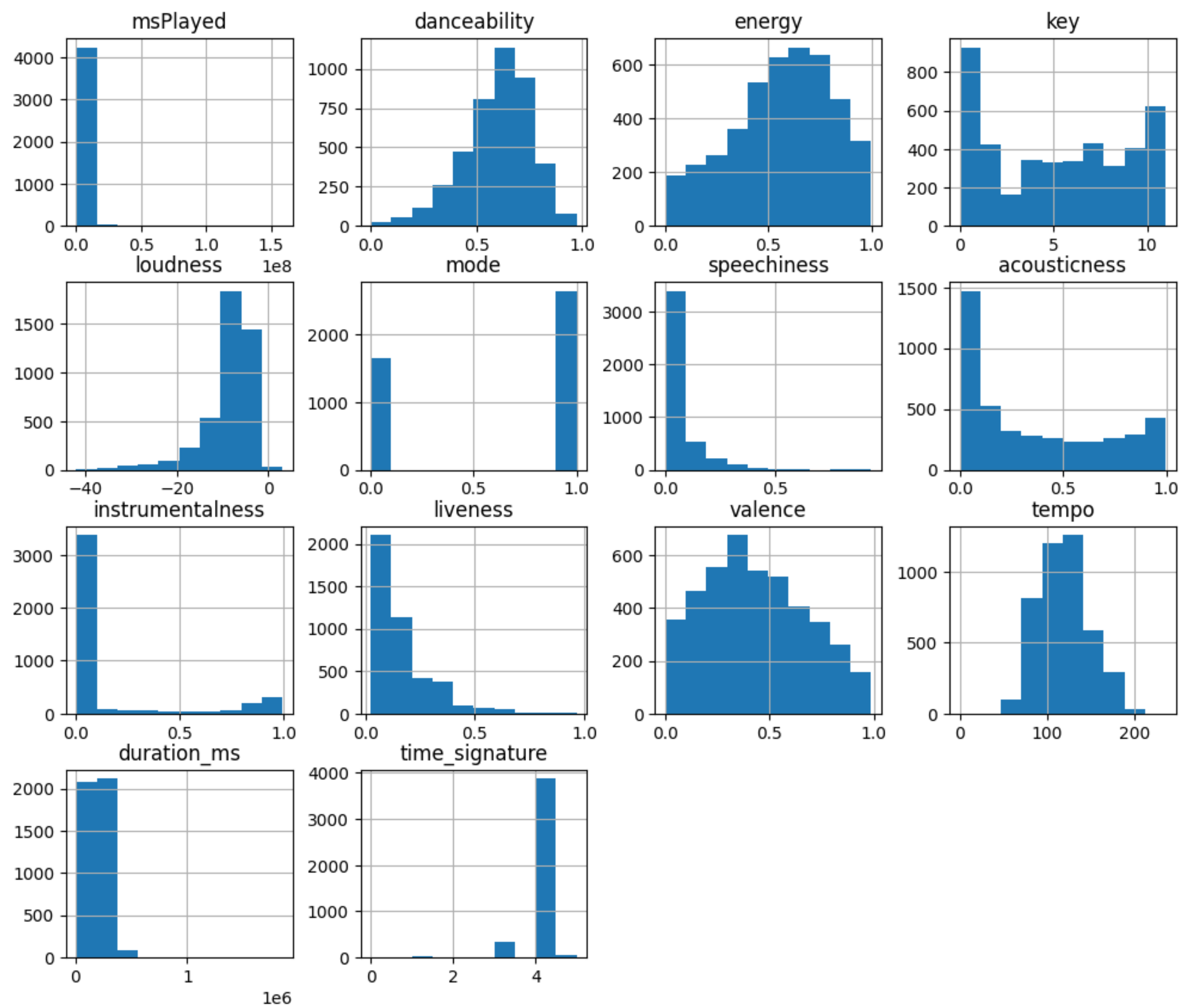
- id: The unique identifier of the track.
- uri: The Spotify URI for the track.
- track_href: A link to the Spotify Web API endpoint for the track.
- analysis_url: A link to the audio analysis of the track.
- duration_ms: The duration of the track in milliseconds.
- time_signature: The time signature of the track.

Unique Values

```
d.nunique()
```

```
trackName           4113
artistName          1828
msPlayed            4240
genre                523
danceability         749
energy              1029
key                   12
loudness            3624
mode                   2
speechiness          963
acousticness        1822
instrumentalness    1792
liveness             951
valence             1099
tempo               3839
type                   1
id                  4261
uri                 4261
track_href          4261
analysis_url        4261
duration_ms         4076
time_signature         5
dtype: int64
```

```
numeric_columns = df_cleaned.select_dtypes(include=['int', 'float'])
numeric_columns.hist(figsize=(12, 10))
plt.show()
```



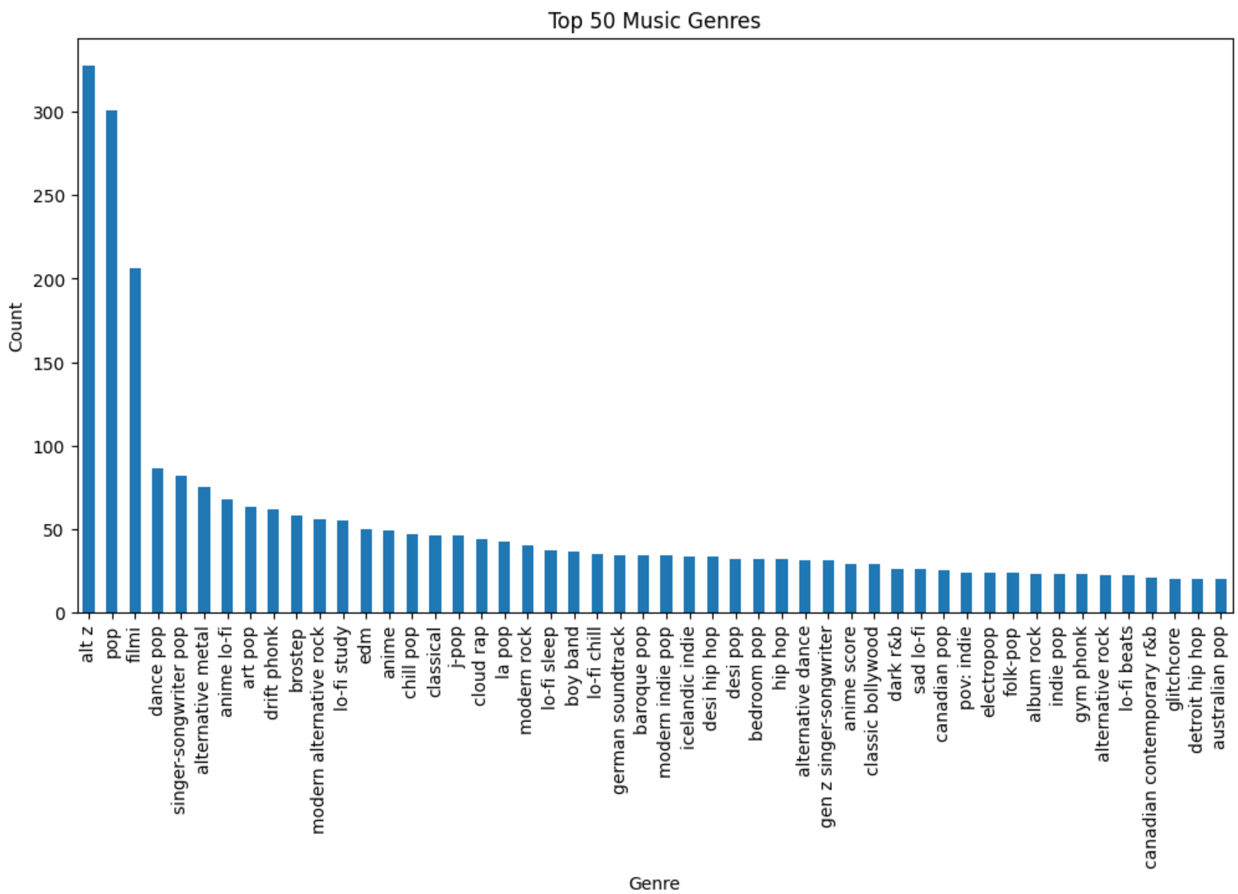
Univariate Analysis

Q1. How diverse are the genres represented in the dataset(Top 50),

and which genres are most prevalent?

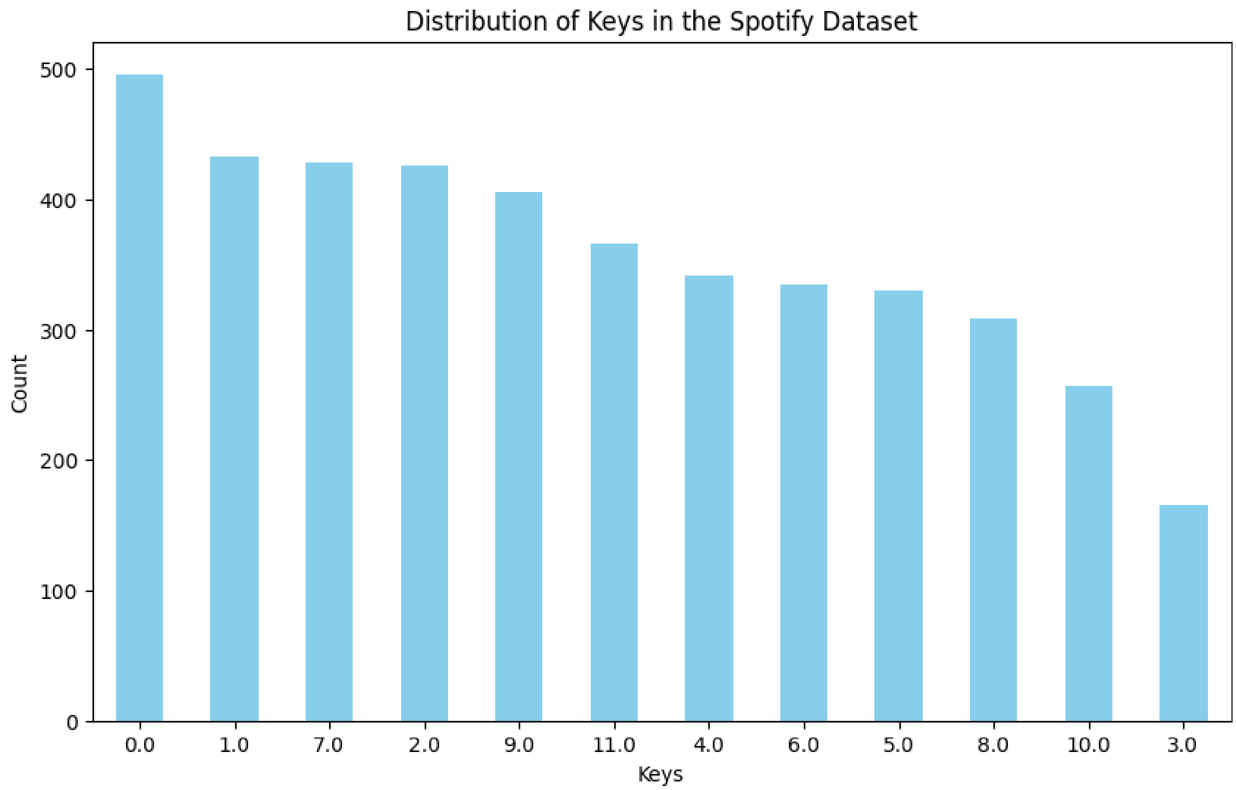
```
top_50_genres = d['genre'].value_counts().head(50)

plt.figure(figsize=(12, 6))
top_50_genres.plot(kind='bar')
plt.xlabel('Genre')
plt.ylabel('Count')
plt.title('Top 50 Music Genres')
```

```
plt.xticks(rotation=90)  # Rotate x-axis labels for better readability
plt.show()
```



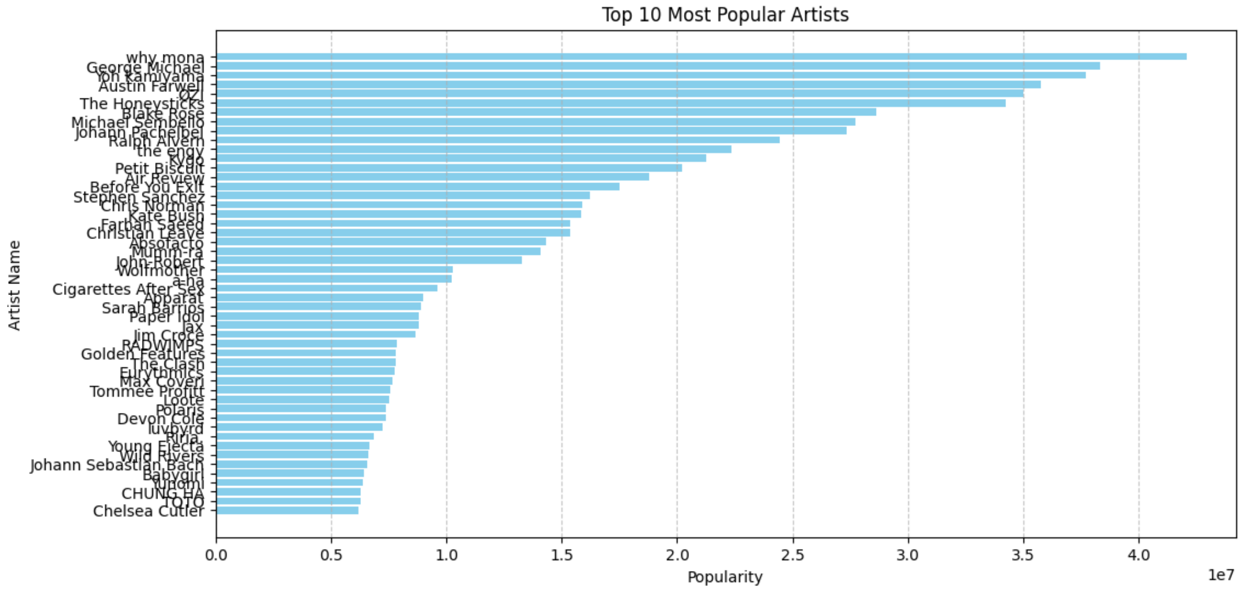
Q2. What are the most common keys for the tracks in the dataset?

```
plt.figure(figsize=(10, 6))
d['key'].value_counts().plot(kind='bar', color='skyblue')
plt.xlabel('Keys')
plt.ylabel('Count')
plt.title('Distribution of Keys in the Spotify Dataset')
plt.xticks(rotation=0)  # Rotate x-axis labels if necessary
plt.show()
```



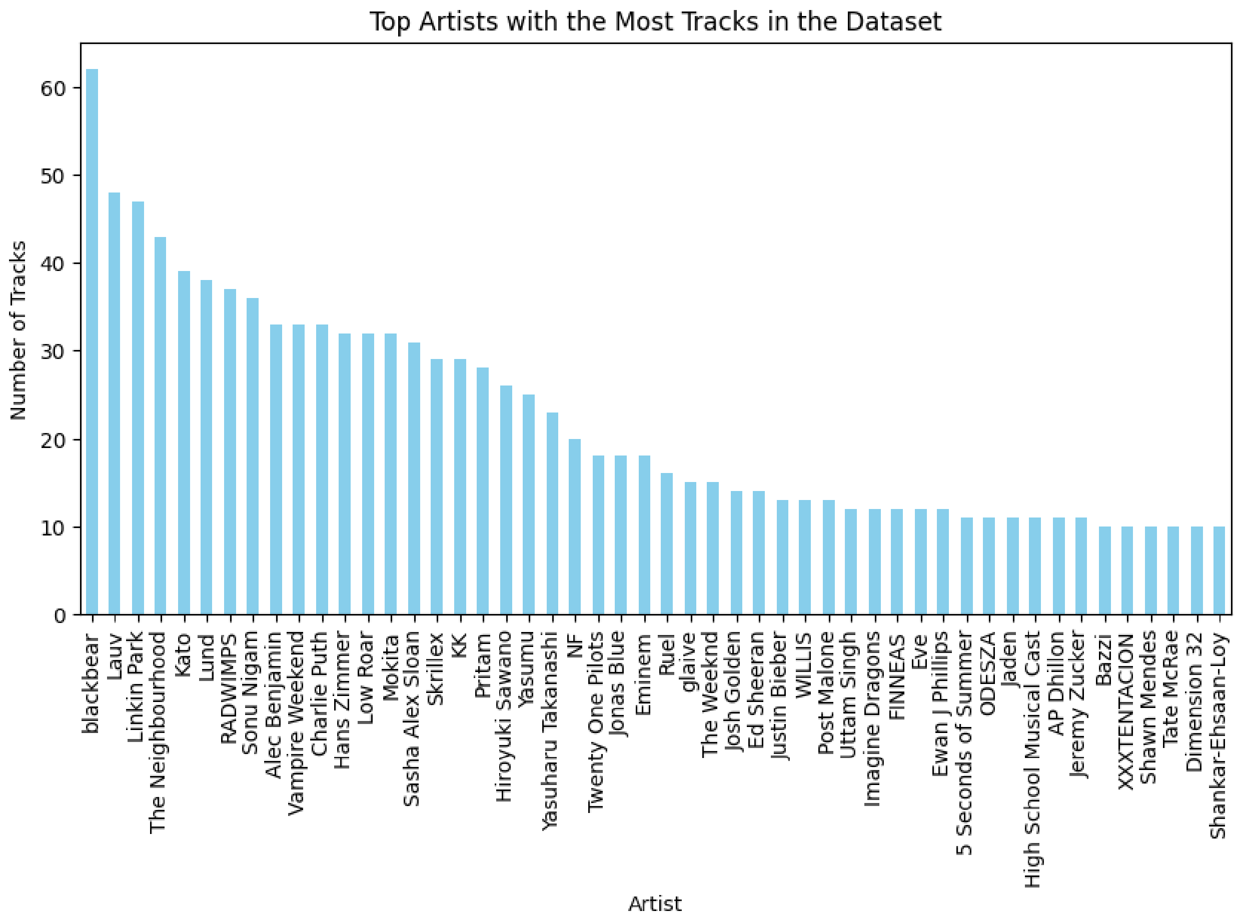
Q3. Top 50 most popular artists in the dataset.

```
# Group the data by artistName and calculate the mean popularity for
each artist
artist_popularity =d.groupby('artistName')
['msPlayed'].mean().reset_index()

# Sort the data by popularity in descending order to find the top 10
artists
top_50_artists = artist_popularity.sort_values(by='msPlayed',
ascending=False).head(50)

# Create a bar chart to visualize the top 10 most popular artists
plt.figure(figsize=(12, 6))
plt.barh(top_50_artists['artistName'], top_50_artists['msPlayed'],
color='skyblue')
plt.xlabel('Popularity')
plt.ylabel('Artist Name')
plt.title('Top 10 Most Popular Artists')
plt.gca().invert_yaxis()  # Invert the y-axis to show the most popular
artist at the top
plt.grid(axis='x', linestyle='--', alpha=0.7)

# Show the bar chart
plt.show()
```



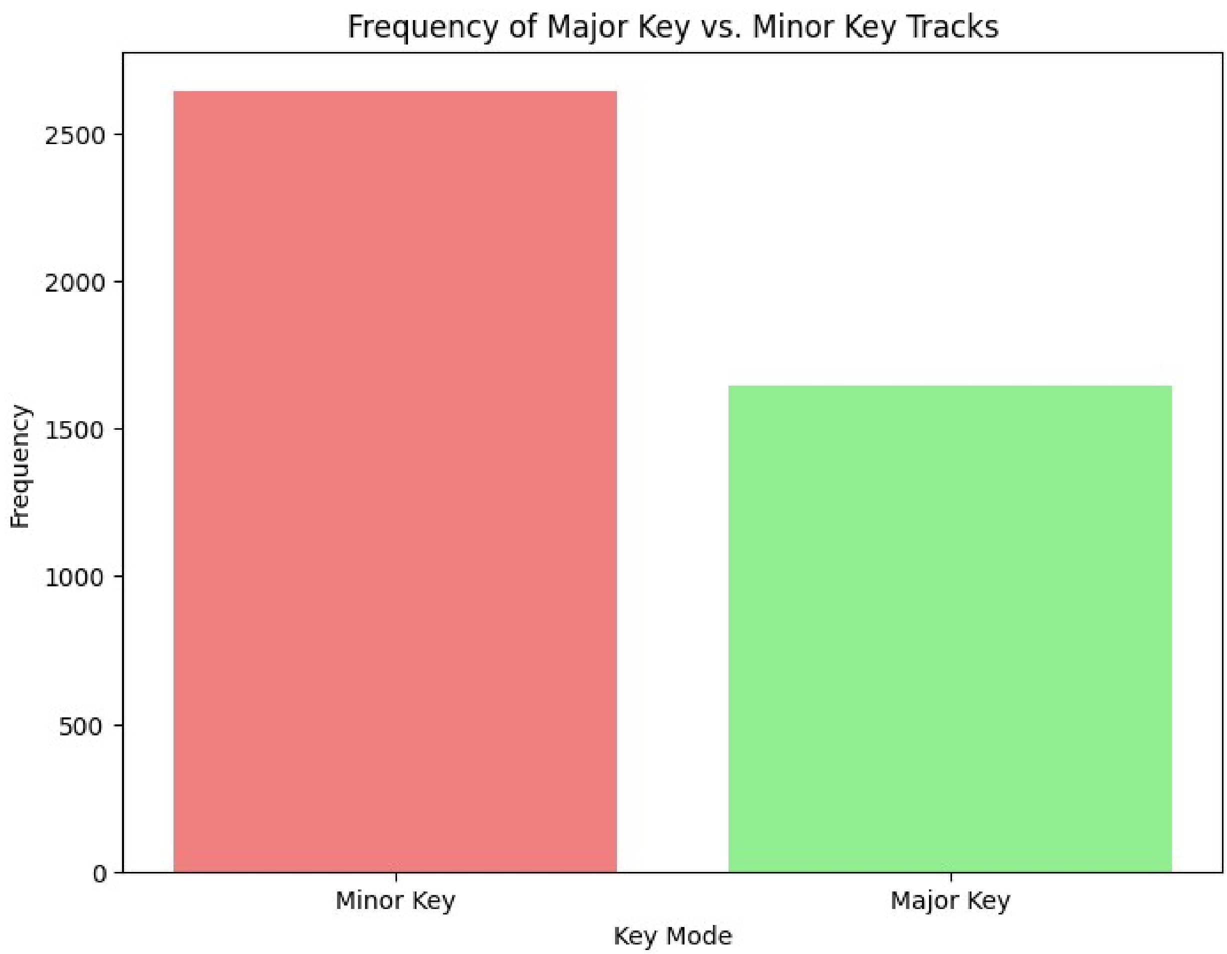
Q4. Which artist has the most tracks in the dataset?

```
# Group by artistName and count the number of tracks for each artist
artist_track_counts = d['artistName'].value_counts()

# Select the top N artists you want to visualize (e.g., top 50)
top_n_artists = artist_track_counts.head(50)

# Create a bar chart to visualize the top N artists and their track
counts
plt.figure(figsize=(10, 5))
top_n_artists.plot(kind='bar', color='skyblue')
plt.xlabel('Artist')
plt.ylabel('Number of Tracks')
plt.title('Top Artists with the Most Tracks in the Dataset')

plt.show()
```



Q5. Are major key tracks more common than minor key tracks?

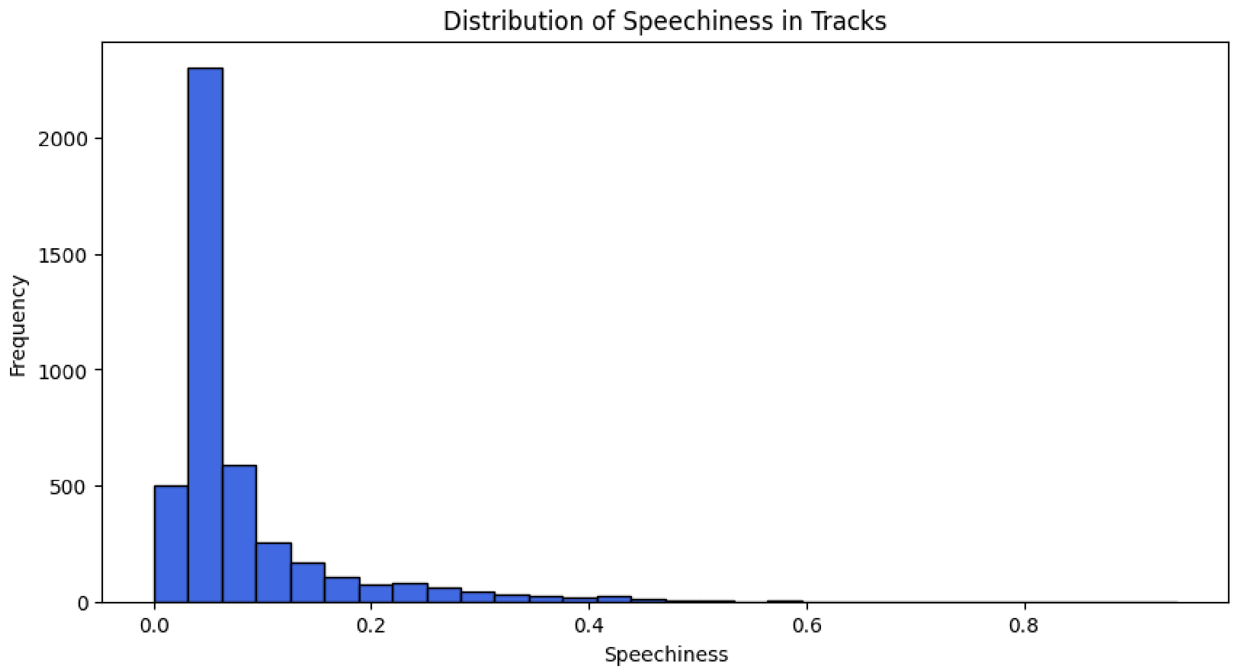
```
# Create a bar chart to compare the frequency of major and minor key
tracks
key_counts = d['mode'].value_counts()
key_labels = ['Minor Key', 'Major Key']

# Create a bar chart
plt.figure(figsize=(8, 6))
plt.bar(key_labels, key_counts, color=['lightcoral', 'lightgreen'])
plt.xlabel('Key Mode')
plt.ylabel('Frequency')
plt.title('Frequency of Major Key vs. Minor Key Tracks')
plt.show()

# Print the counts
print(key_counts)
```




```
mode
1.0    2644
0.0    1646
Name: count, dtype: int64
```

Q6. How speechy are the tracks on average?

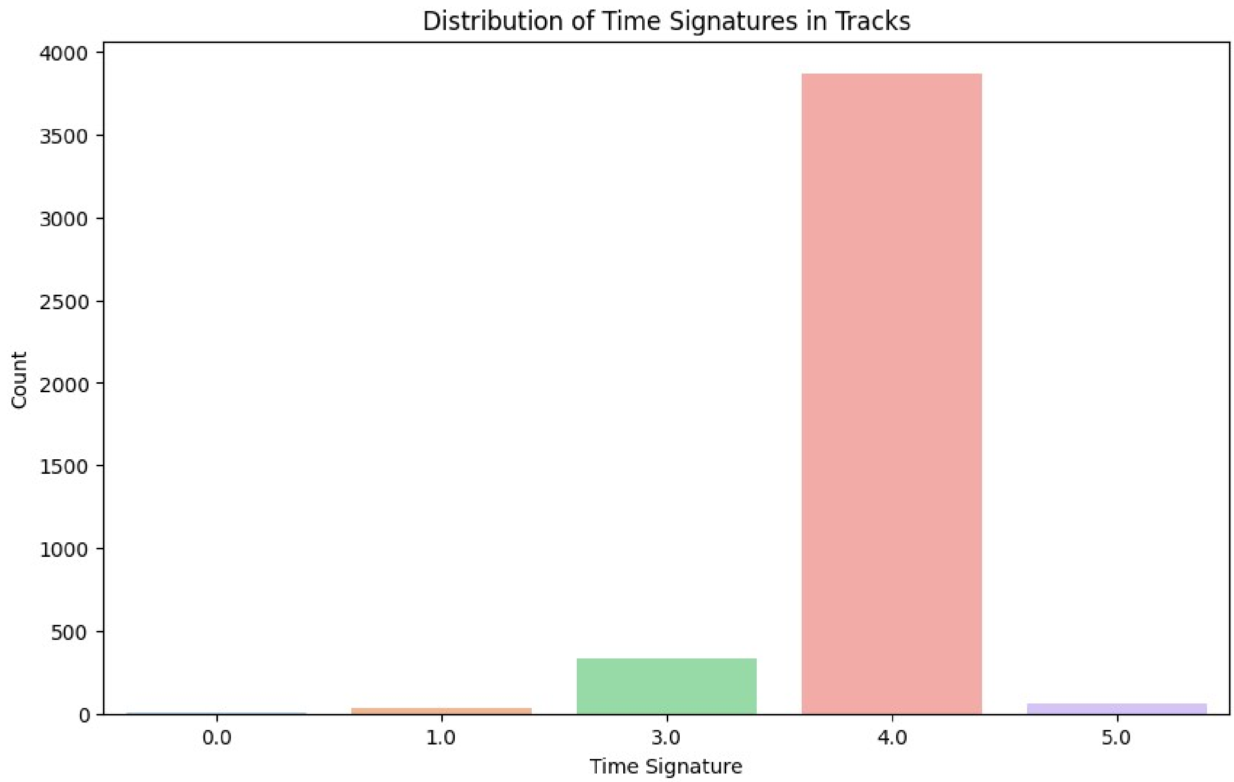
```
# Create a histogram to visualize the distribution of speechiness
scores
plt.figure(figsize=(10, 5))
plt.hist(d['speechiness'], bins=30, color='royalblue',
edgecolor='black')
plt.xlabel('Speechiness')
plt.ylabel('Frequency')
plt.title('Distribution of Speechiness in Tracks')
plt.show()

# Calculate the average speechiness
average_speechiness = d['speechiness'].mean()
print("Average Speechiness:", average_speechiness)
```




```
Average Speechiness: 0.07832058275058275
```

Q7. How does the time signature vary across tracks?

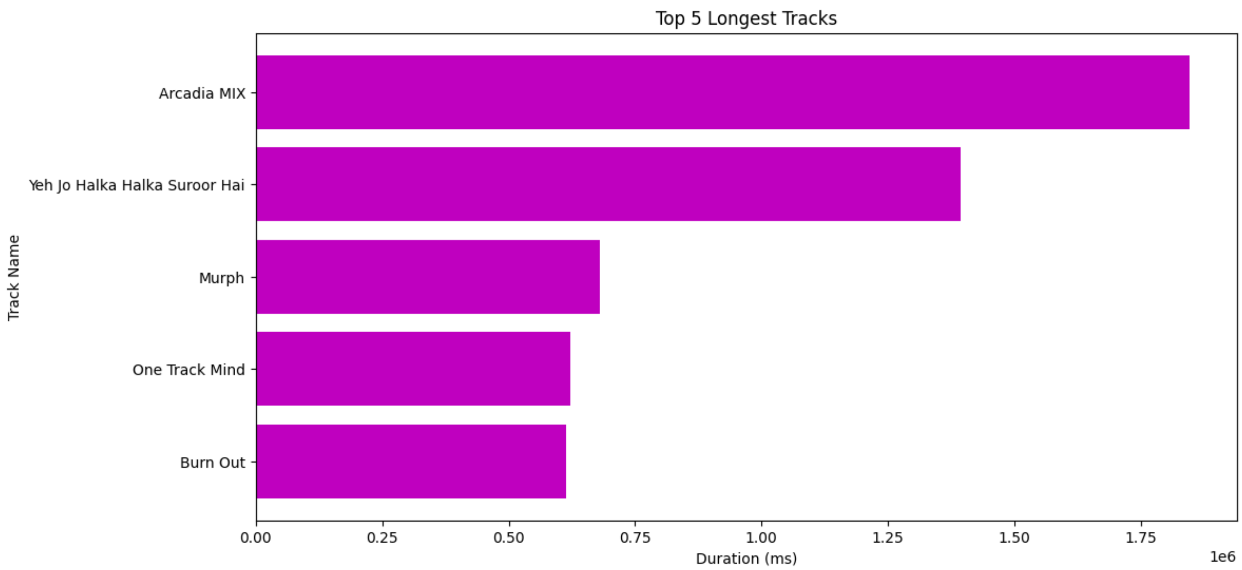
```
# Create a countplot to visualize the distribution of time signatures
plt.figure(figsize=(10, 6))
sns.countplot(data=d, x='time_signature', palette='pastel')
plt.xlabel('Time Signature')
plt.ylabel('Count')
plt.title('Distribution of Time Signatures in Tracks')
plt.show()
```

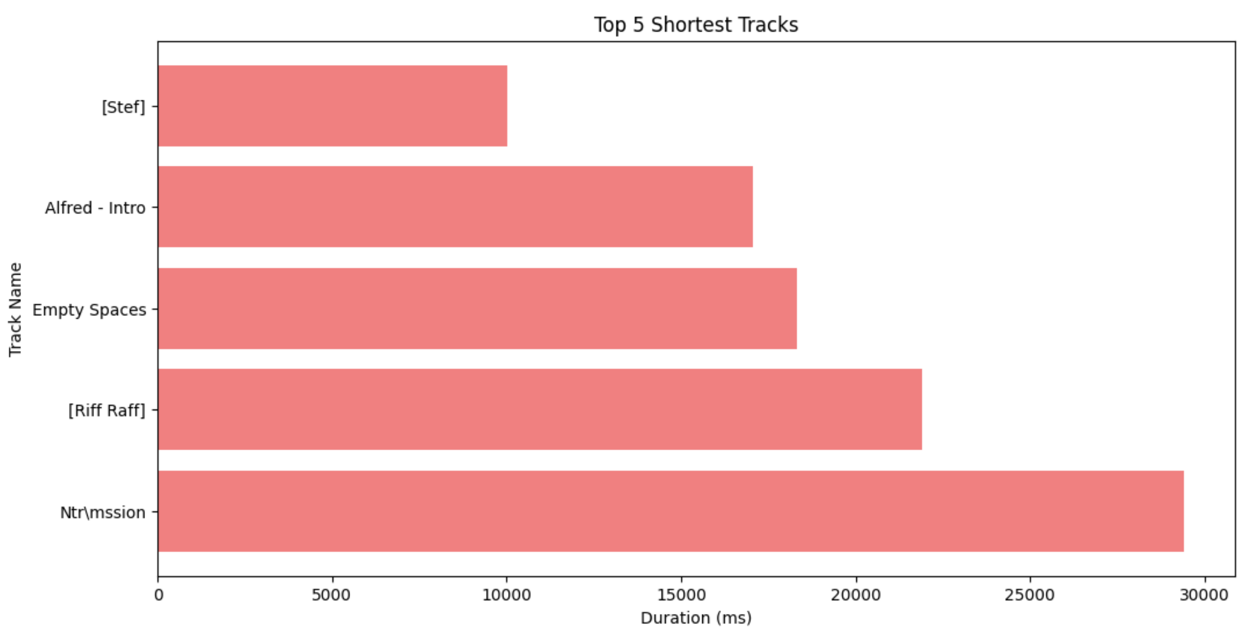


Q8. Which track has the longest duration and which has shortest duration? (Top 5)

```
# Sort the DataFrame by track duration in descending order to get the
longest tracks first
longest_tracks = d.sort_values(by='duration_ms',
ascending=False).head(5)

# Sort the DataFrame by track duration in ascending order to get the
shortest tracks first
shortest_tracks = d.sort_values(by='duration_ms',
ascending=True).head(5)

# Create a bar chart to visualize the top 5 longest tracks
plt.figure(figsize=(12, 6))
plt.barh(longest_tracks['trackName'], longest_tracks['duration_ms'],
color='m')
plt.xlabel('Duration (ms)')
plt.ylabel('Track Name')
plt.title('Top 5 Longest Tracks')
plt.gca().invert_yaxis()  # Reverse the order to display the longest
track at the top
plt.show()

# Create a bar chart to visualize the top 5 shortest tracks
```

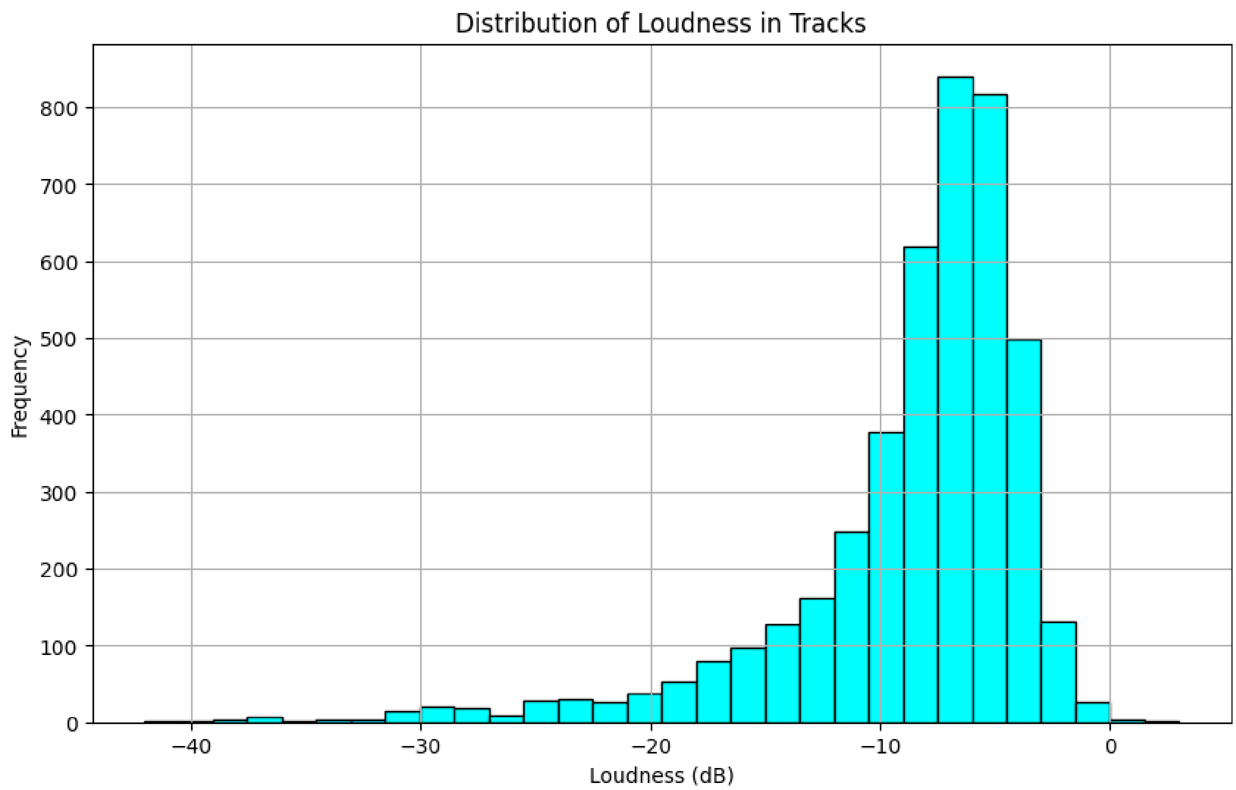
```
plt.figure(figsize=(12, 6))
plt.barh(shortest_tracks['trackName'], shortest_tracks['duration_ms'],
color='lightcoral')
plt.xlabel('Duration (ms)')
plt.ylabel('Track Name')
plt.title('Top 5 Shortest Tracks')
plt.gca().invert_yaxis()  # Reverse the order to display the shortest
track at the top
plt.show()
```





Q9. What is the distribution of loudness across tracks?

```
# Create a histogram to visualize the distribution of loudness
plt.figure(figsize=(10, 6))
plt.hist(d['loudness'], bins=30, color='cyan', edgecolor='black')
plt.xlabel('Loudness (dB)')
plt.ylabel('Frequency')
plt.title('Distribution of Loudness in Tracks')
plt.grid(True)
plt.show()
```



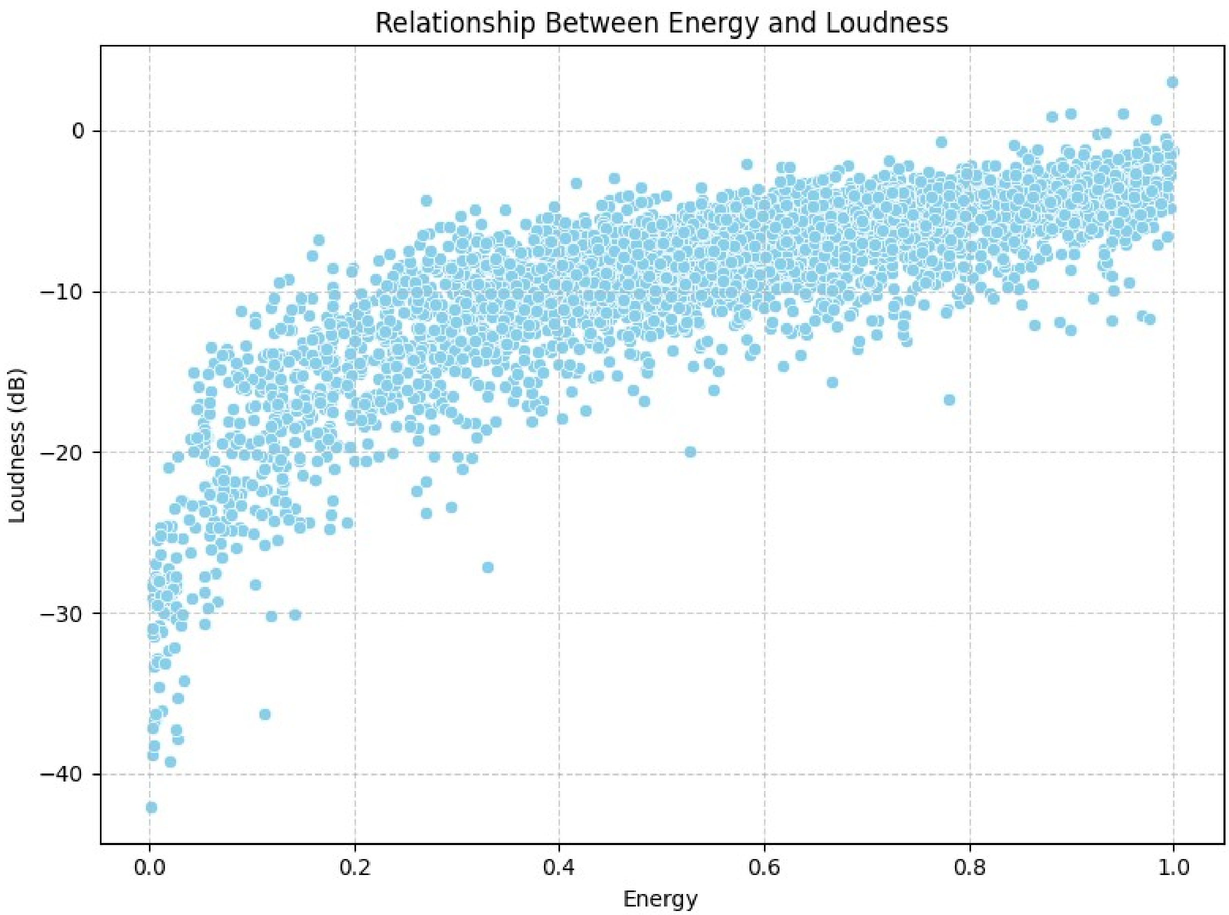
Bivariate Analysis

Q10. Is there a correlation between 'energy' and 'loudness' attributes?

```
# Create a scatter plot to visualize the relationship between 'energy'
and 'loudness'
plt.figure(figsize=(8, 6))
sns.scatterplot(data=d, x='energy', y='loudness', color='skyblue')
plt.xlabel('Energy')
plt.ylabel('Loudness (dB)')
```

```
plt.title('Relationship Between Energy and Loudness')
plt.grid(True, linestyle='--', alpha=0.6)

# Calculate and print the correlation coefficient
correlation_coefficient = d['energy'].corr(d['loudness'])
print(f"Correlation Coefficient: {correlation_coefficient:.2f}")

plt.tight_layout()
plt.show()
```

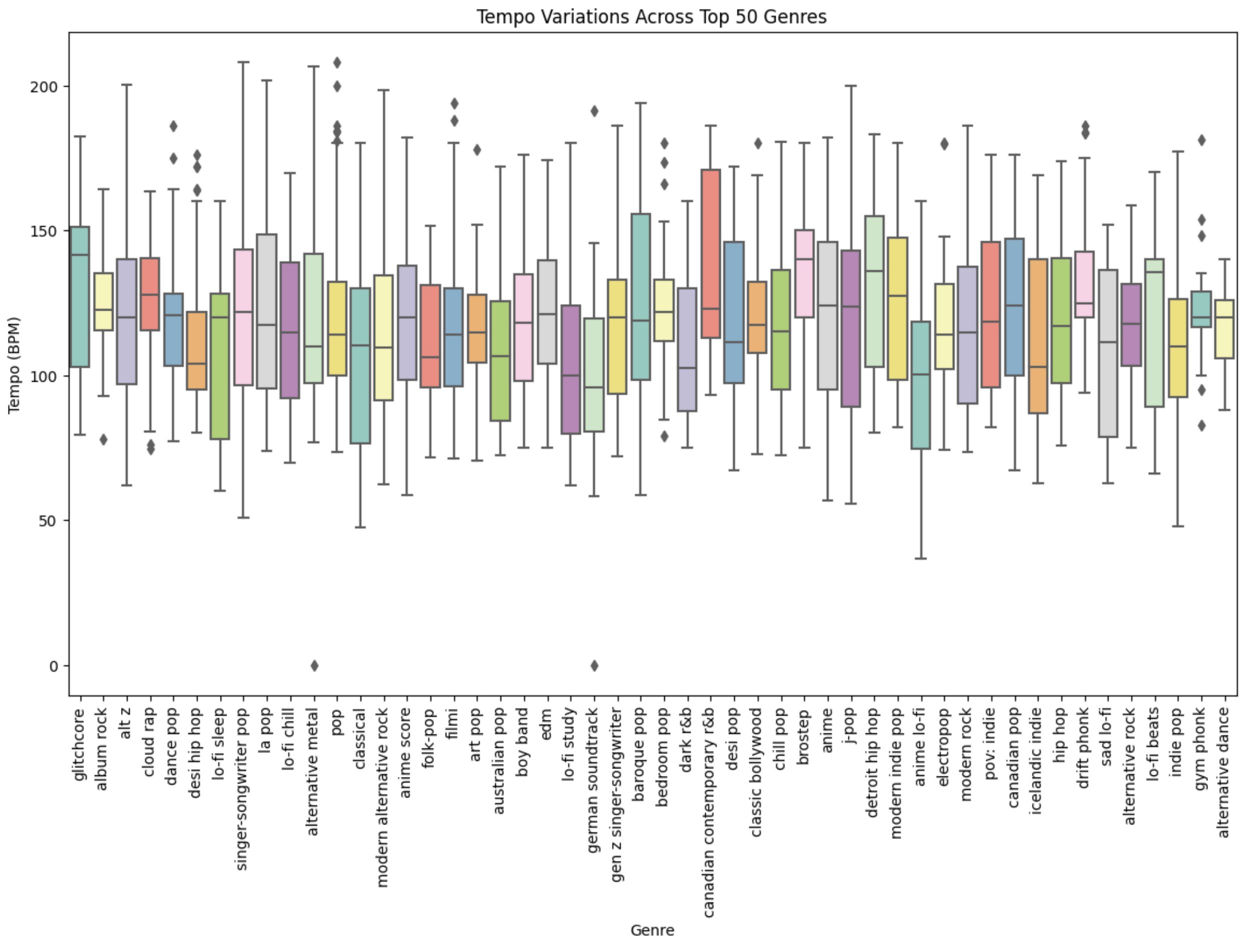
```
Correlation Coefficient: 0.79
```



Q11. How does the 'tempo' attribute vary across different genres?

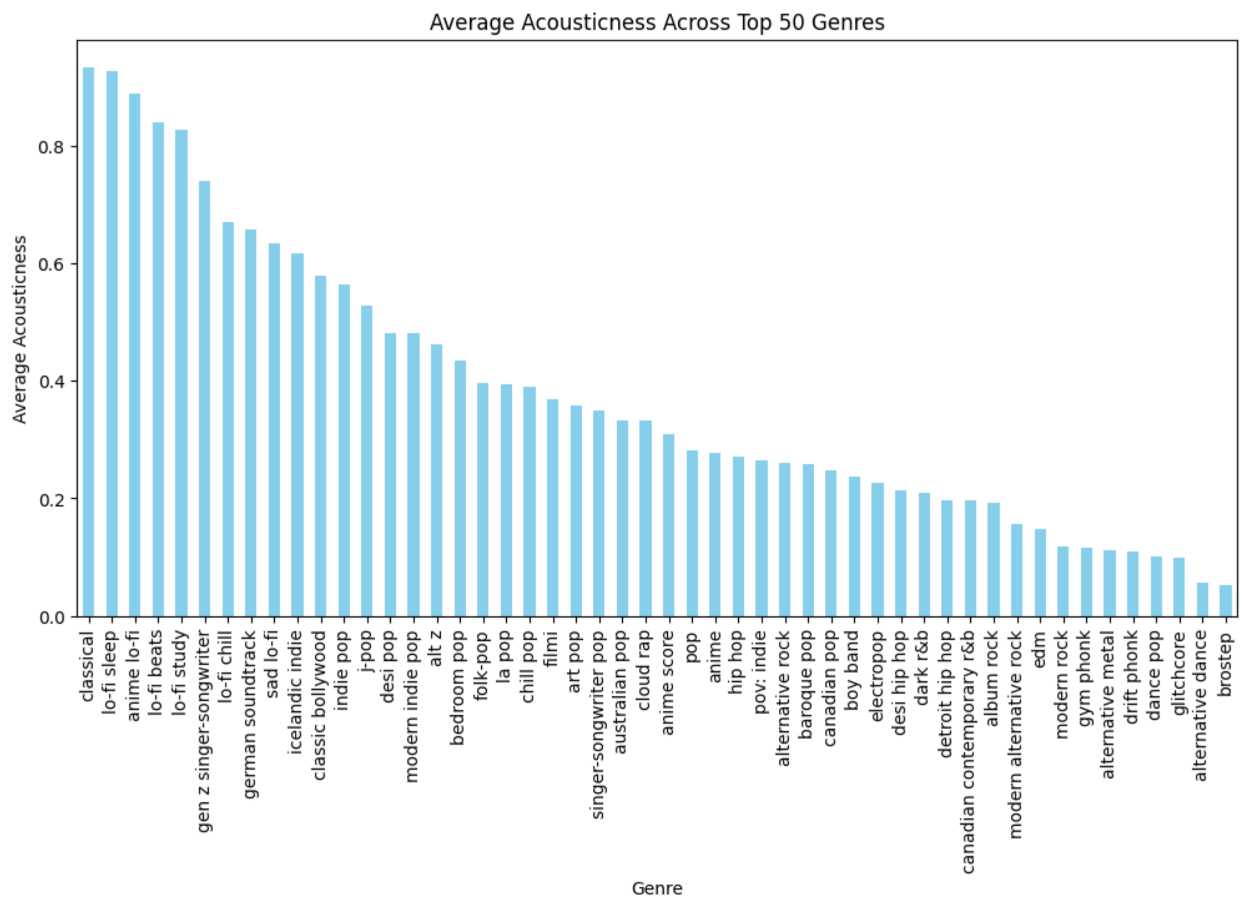
```
top_50_genres = d['genre'].value_counts().head(50).index
df_top_50 = d[d['genre'].isin(top_50_genres)]


plt.figure(figsize=(14, 8))
sns.boxplot(data=df_top_50, x='genre', y='tempo', palette='Set3')
```

```
plt.xlabel('Genre')
plt.ylabel('Tempo (BPM)')
plt.title('Tempo Variations Across Top 50 Genres')
plt.xticks(rotation=90)
plt.show()
```



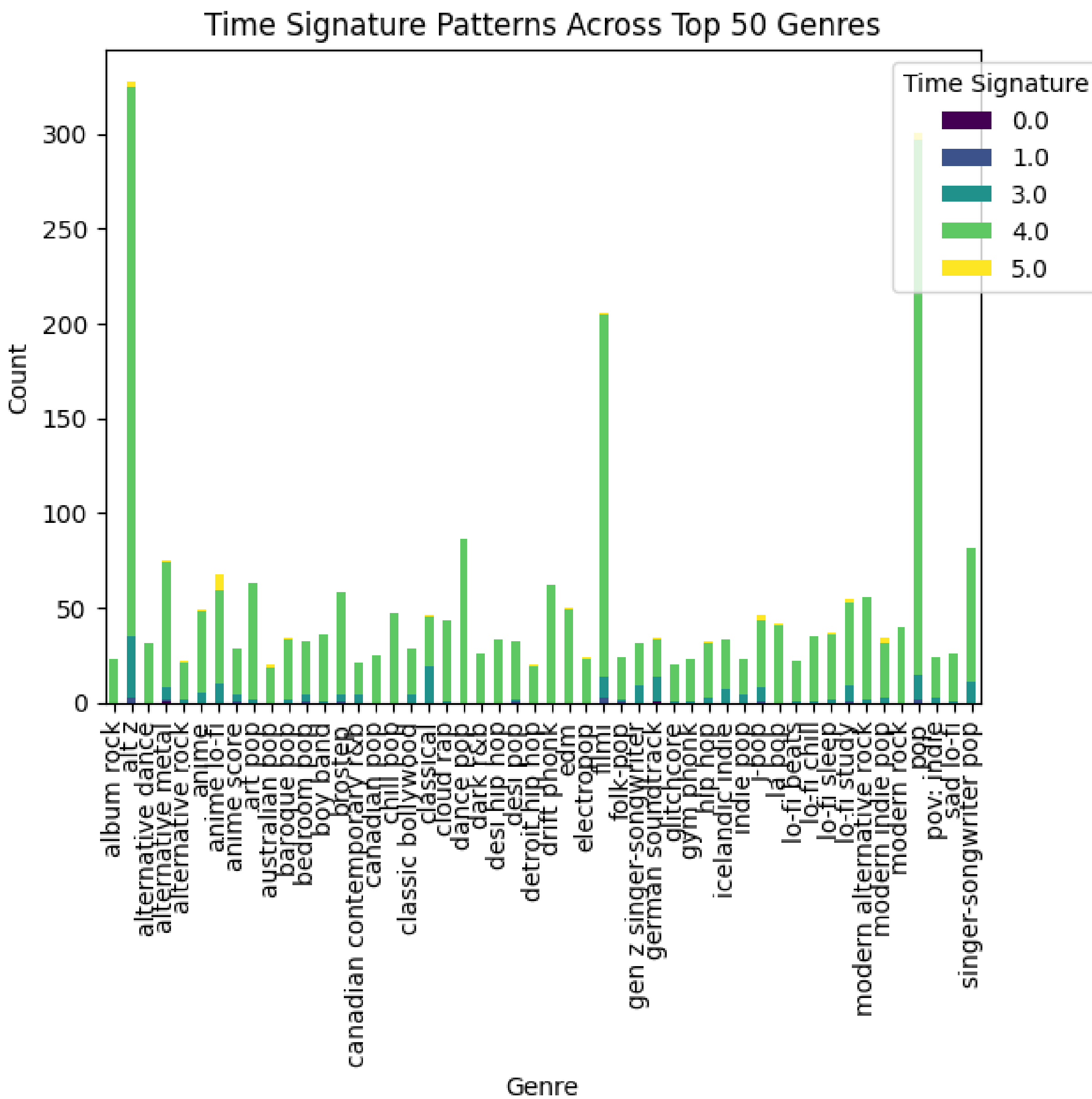
Q12. Trends in 'acousticness' across different genres.

```
top_50_genres = d['genre'].value_counts().head(50).index
df_top_50 = d[d['genre'].isin(top_50_genres)]

genre_acousticness = df_top_50.groupby('genre')
['acousticness'].mean().sort_values(ascending=False)

plt.figure(figsize=(12, 6))
genre_acousticness.plot(kind='bar', color='skyblue')
plt.xlabel('Genre')
plt.ylabel('Average Acousticness')
plt.title('Average Acousticness Across Top 50 Genres')
plt.xticks(rotation=90)
plt.show()
```



Q13. Patterns in the 'time_signature' attribute across different genres.

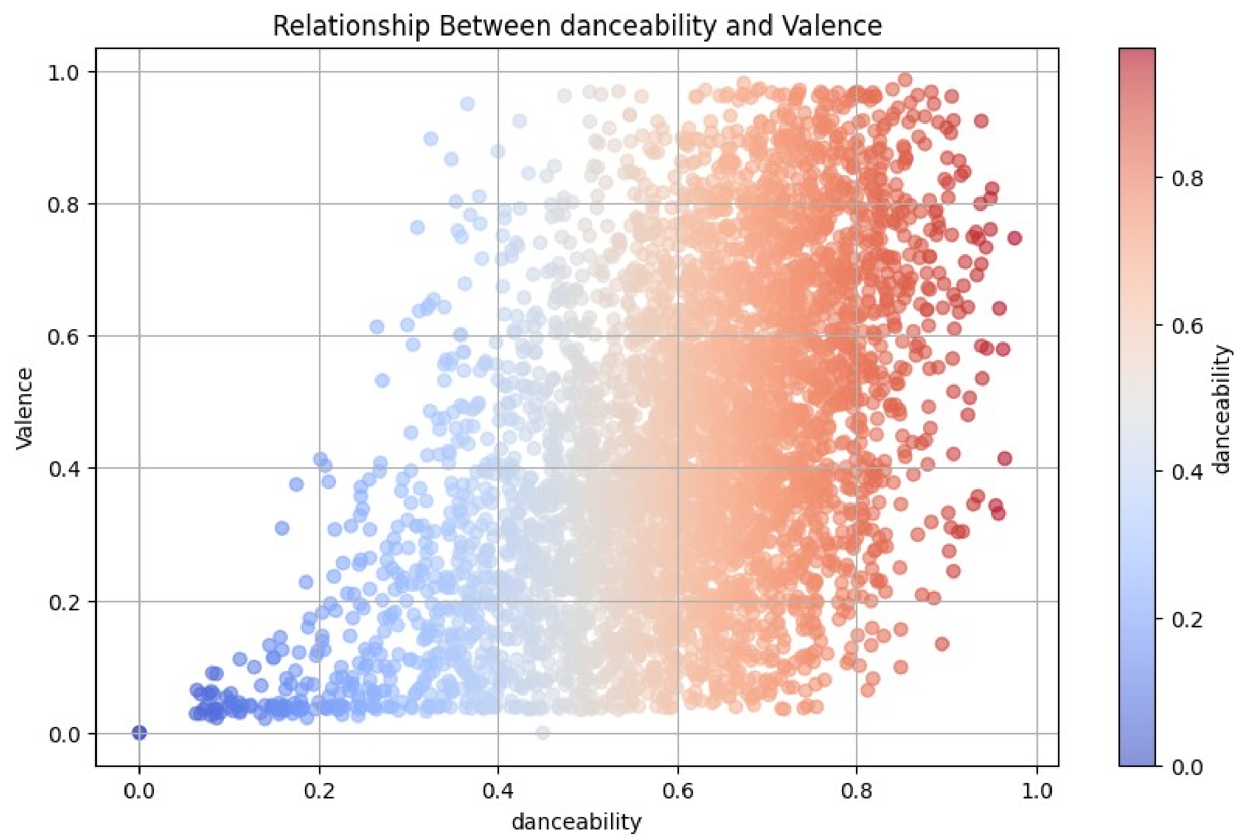
```
top_50_genres = d['genre'].value_counts().head(50).index
df_top_50 = d[d['genre'].isin(top_50_genres)]

time_signature_frequencies = df_top_50.groupby(['genre',
'time_signature'])['time_signature'].count().unstack(fill_value=0)

plt.figure(figsize=(12, 8))
time_signature_frequencies.plot(kind='bar', stacked=True,
cmap='viridis')
plt.xlabel('Genre')
plt.ylabel('Count')
plt.title('Time Signature Patterns Across Top 50 Genres')
plt.legend(title='Time Signature', loc='upper right',
bbox_to_anchor=(1.15, 1))
plt.show()

<Figure size 1200x800 with 0 Axes>
```



Q14. The relationship between danceability and valence (positivity)

```
# Extract tempo and valence columns from the DataFrame
danceability = d['danceability']
valence = d['valence']

# Create a scatter plot with different colors for tempo and valence
plt.figure(figsize=(10, 6))
plt.scatter(danceability, valence, c=danceability, cmap='coolwarm',
alpha=0.6)
```

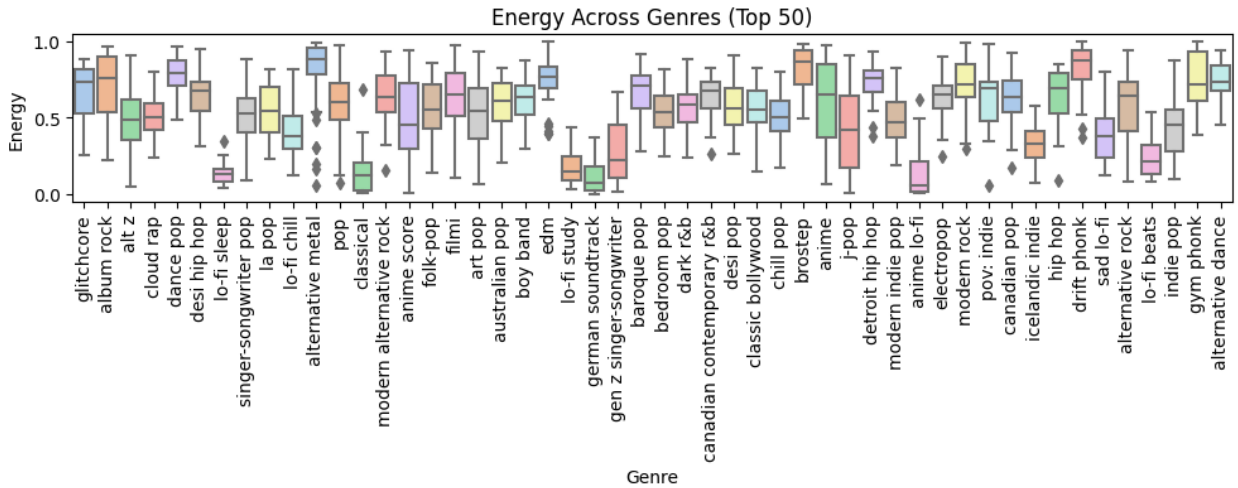
```
# Add labels and a colorbar
plt.xlabel('danceability')
plt.ylabel('Valence')
plt.title('Relationship Between danceability and Valence')
cbar = plt.colorbar()
cbar.set_label('danceability')

# Show the plot
plt.grid(True)
plt.show()
```

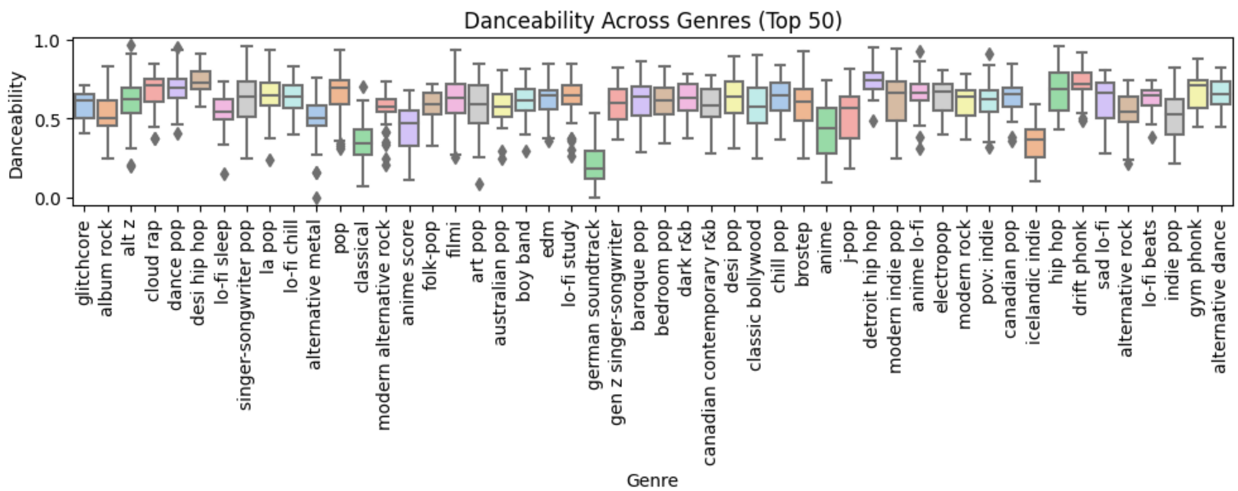


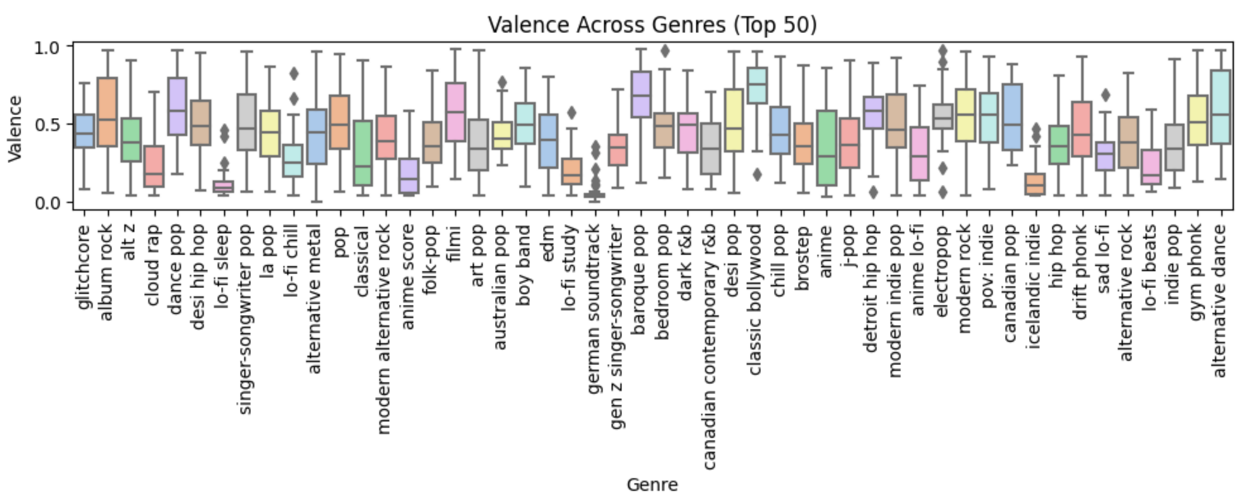
Q15. How do song attributes like energy, danceability, and valence differ across genres(Top 50)

```
# Select the top 50 genres based on frequency
top_50_genres = d['genre'].value_counts().head(50).index.tolist()

# Create a filtered DataFrame containing only the top 50 genres
filtered_df = d[d['genre'].isin(top_50_genres)]

# Define the song attributes you want to compare
attributes = ['energy', 'danceability', 'valence']
```

```
# Create subplots for each attribute
fig, axes = plt.subplots(nrows=len(attributes), ncols=1, figsize=(10,
12))

# Create box plots for each attribute, grouped by genre
for i, attribute in enumerate(attributes):
    sns.boxplot(data=filtered_df, x='genre', y=attribute, ax=axes[i],
palette='pastel')
    axes[i].set_ylabel(attribute.capitalize())
    axes[i].set_xlabel('Genre')
    axes[i].set_xticklabels(axes[i].get_xticklabels(), rotation=90)
    axes[i].set_title(f'{attribute.capitalize()} Across Genres (Top
50)')

plt.tight_layout()
plt.show()
```



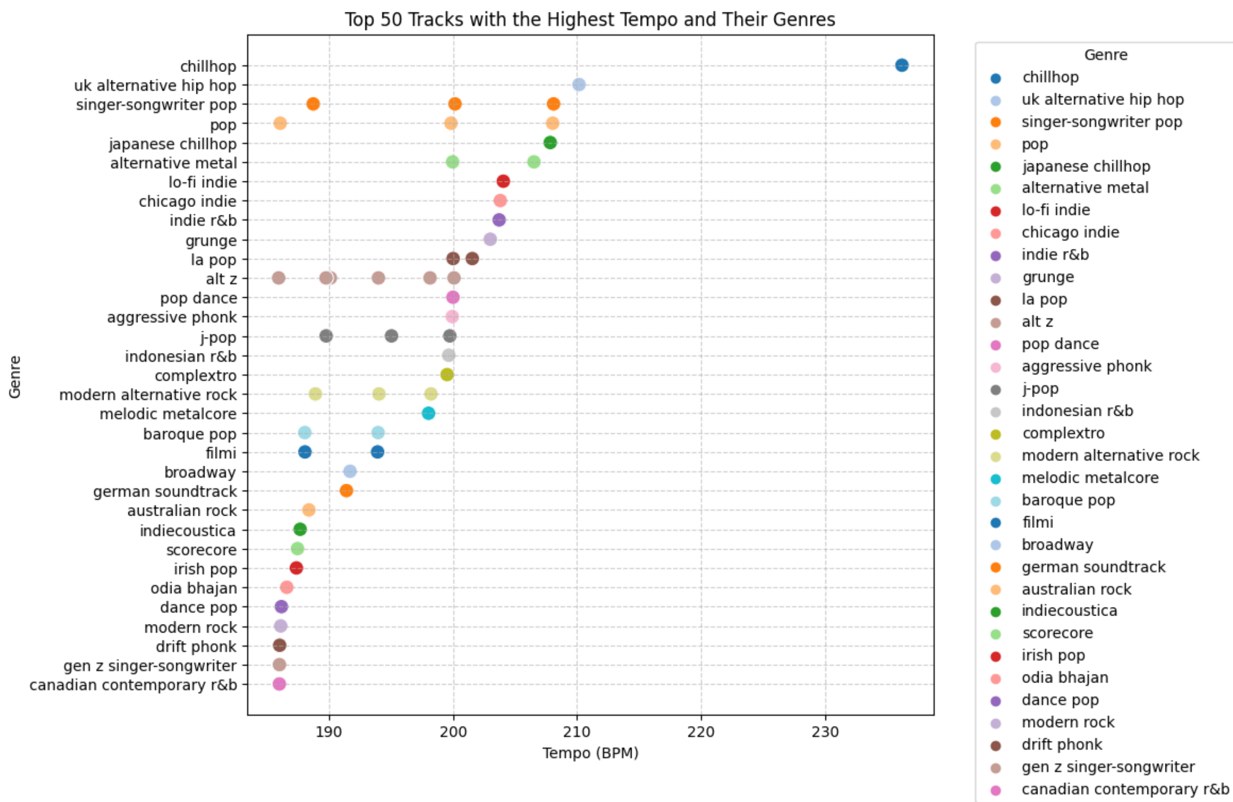




Q16. Are certain genres associated with higher tempos?

```python
# Select the top 50 tracks with the highest tempo
top_50_tempo = d.nlargest(50, 'tempo')
```

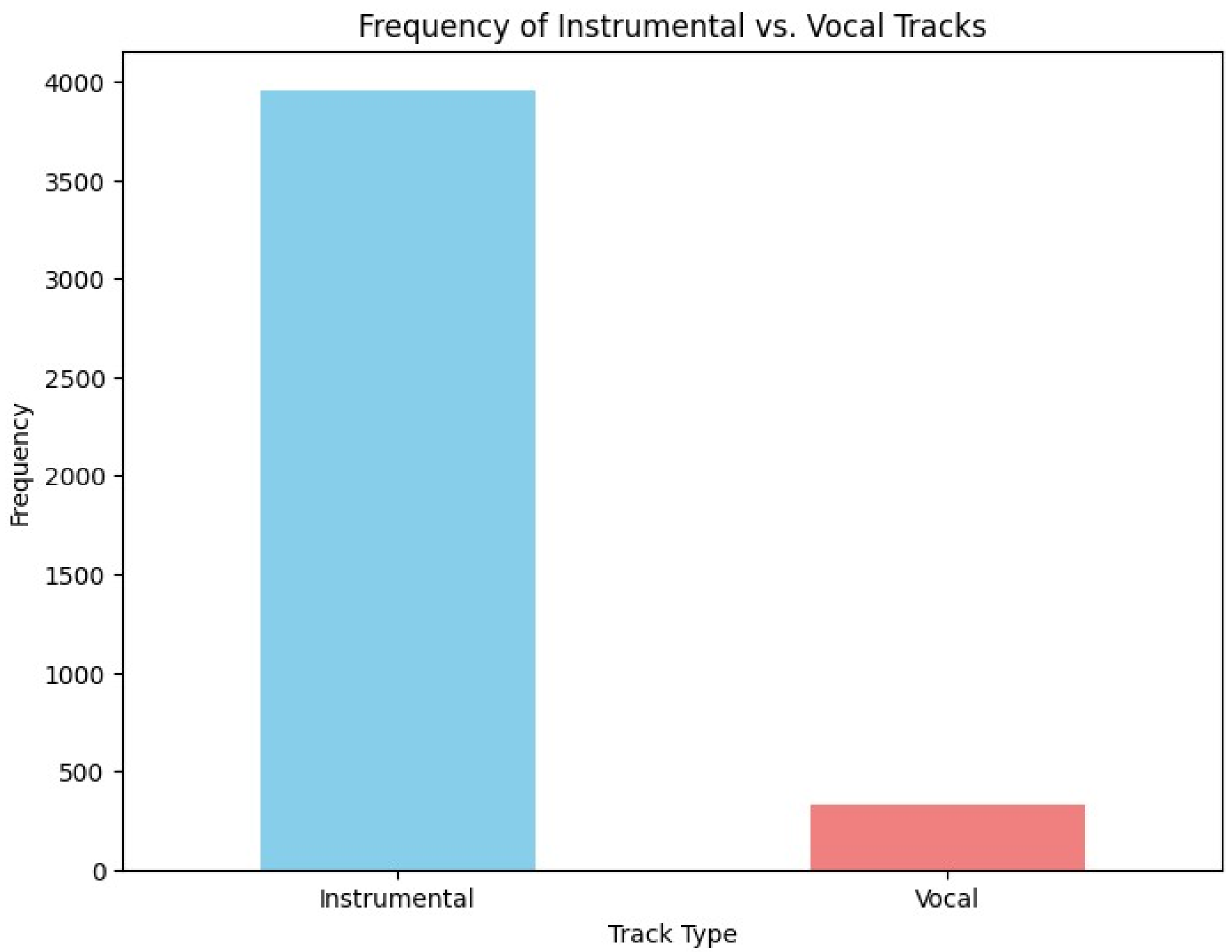
```python
# Create a scatter plot to visualize tempo vs. genre
plt.figure(figsize=(12, 8))
sns.scatterplot(data=top_50_tempo, x='tempo', y='genre', hue='genre',
palette='tab20', s=100)
plt.xlabel('Tempo (BPM)')
plt.ylabel('Genre')
plt.title('Top 50 Tracks with the Highest Tempo and Their Genres')
plt.legend(title='Genre', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.grid(True, linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()
```



Q17. Are instrumental tracks more common than vocal tracks?

```python
d['track_type'] = d['speechiness'].apply(lambda x: 'Instrumental' if x
< 0.2 else 'Vocal')

# Count the number of instrumental and vocal tracks
track_type_counts = d['track_type'].value_counts()

# Create a bar chart to visualize the frequency of track types
plt.figure(figsize=(8, 6))
```

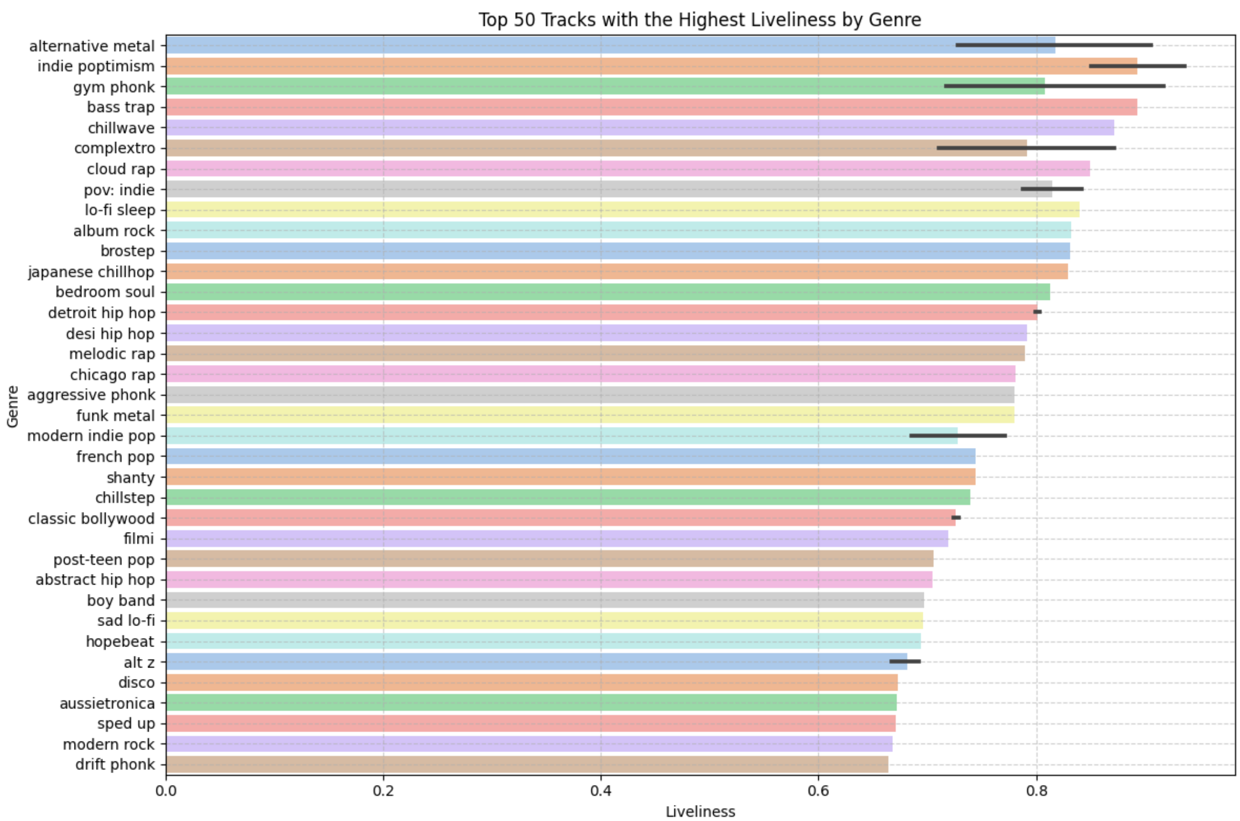
```
track_type_counts.plot(kind='bar', color=['skyblue', 'lightcoral'])
plt.xlabel('Track Type')
plt.ylabel('Frequency')
plt.title('Frequency of Instrumental vs. Vocal Tracks')
plt.xticks(rotation=0)
plt.show()

# Print the counts
print(track_type_counts)
```




```
track_type
Instrumental    3958
Vocal            332
Name: count, dtype: int64
```

Q18. Are live performances more prevalent in certain genres?

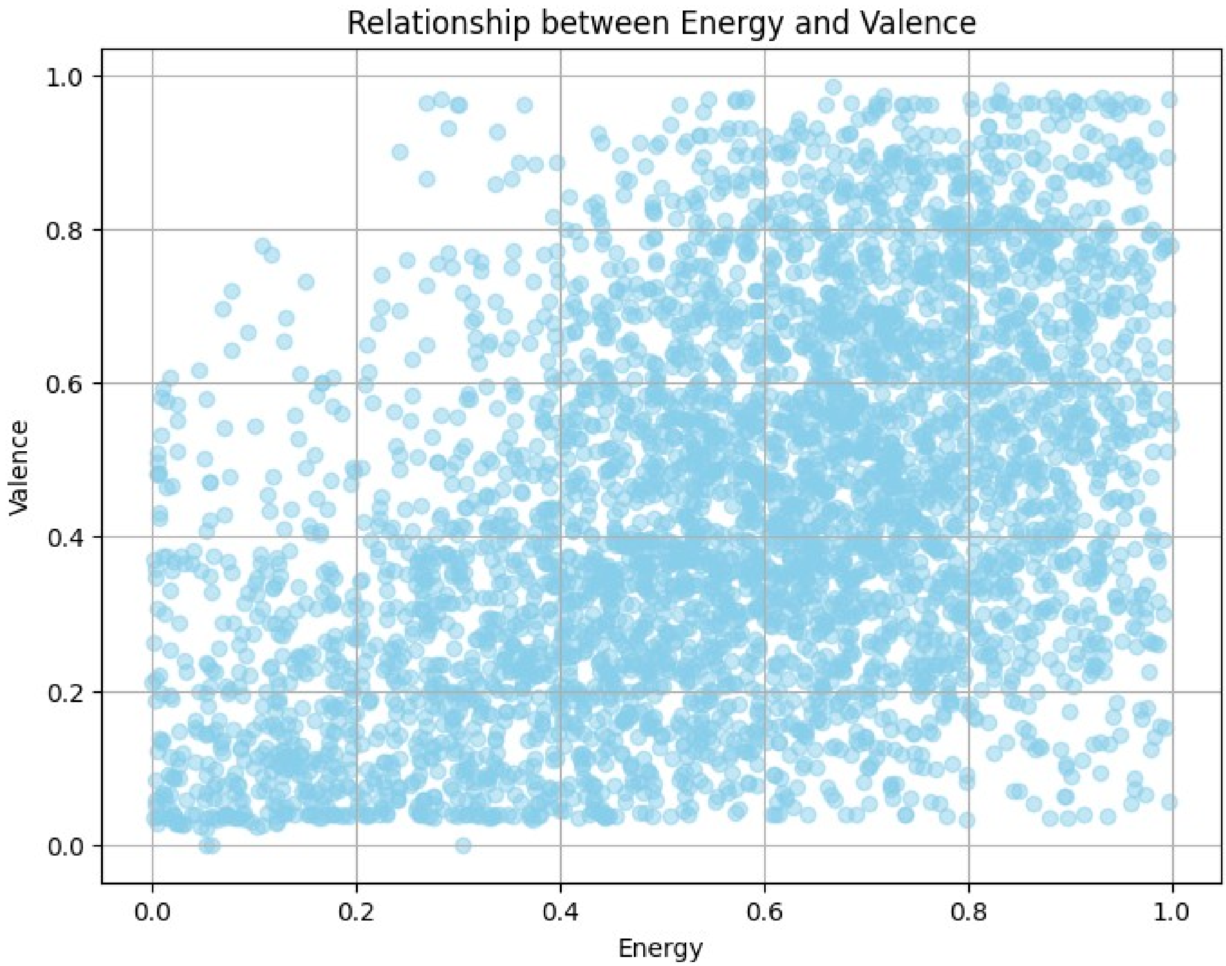
```
# Select the top 50 tracks with the highest liveliness
top_50_liveliness = d.nlargest(50, 'liveness')

# Create a bar chart to visualize the prevalence of live performances
by genre
plt.figure(figsize=(12, 8))
sns.barplot(data=top_50_liveliness, x='liveness', y='genre',
palette='pastel')
plt.xlabel('Liveliness')
plt.ylabel('Genre')
plt.title('Top 50 Tracks with the Highest Liveliness by Genre')
plt.grid(True, linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()
```



Q19. Correlation between 'energy' and 'valence'

```
# Create a scatter plot to visualize the relationship between 'energy'
and 'valence'
plt.figure(figsize=(8, 6))
plt.scatter(d['energy'], d['valence'], alpha=0.5, color='skyblue')
```

```
plt.xlabel('Energy')
plt.ylabel('Valence')
plt.title('Relationship between Energy and Valence')
plt.grid(True)
plt.show()
```

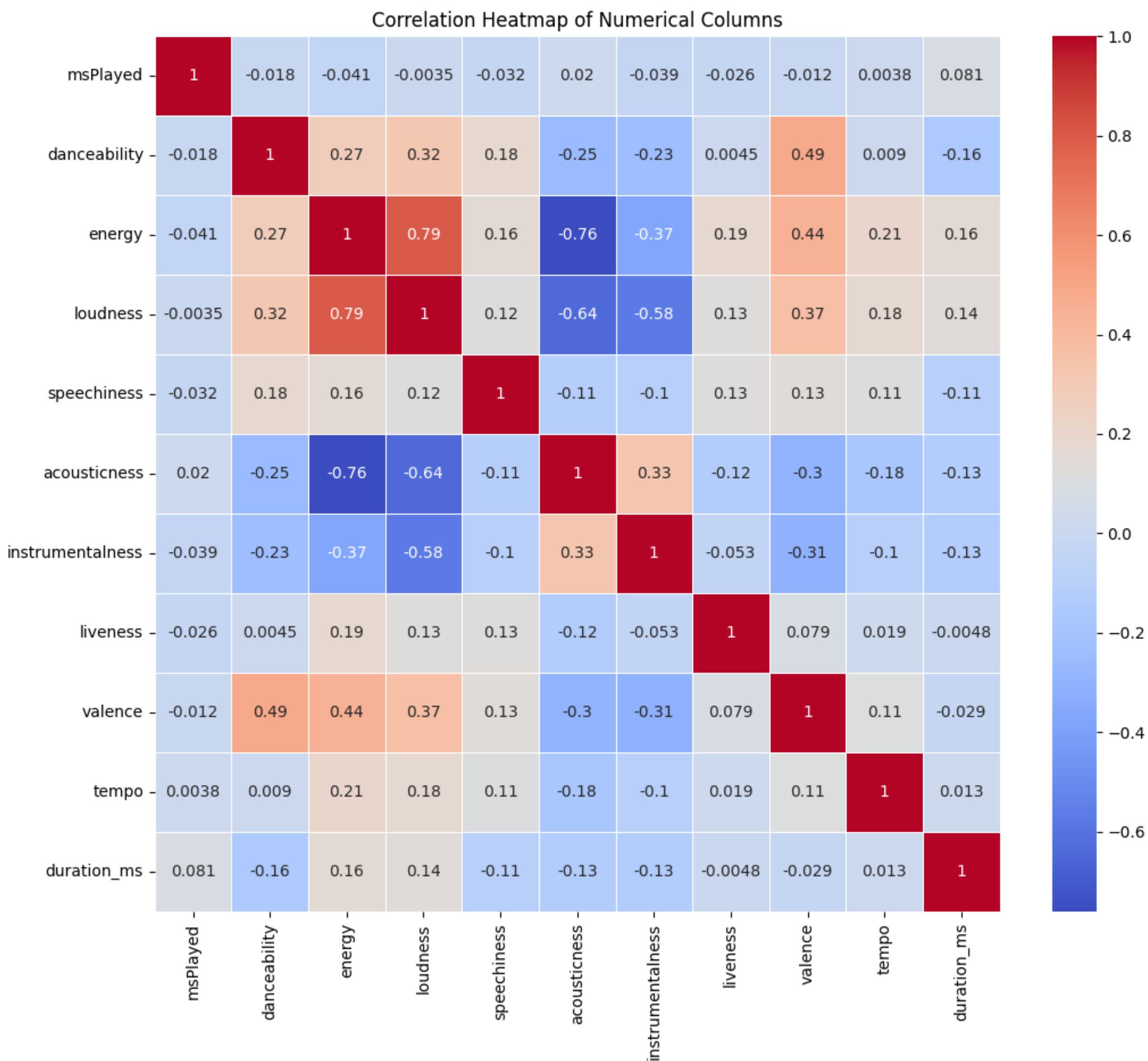


Multivariate Analysis

Q20. Correlation between diffenrent numerical columns.

```
# Selecting only the numerical columns for correlation
numerical_columns = ['msPlayed', 'danceability', 'energy', 'loudness',
'speechiness',
                     'acousticness', 'instrumentalness', 'liveness',
'valence', 'tempo',
                     'duration_ms']
```

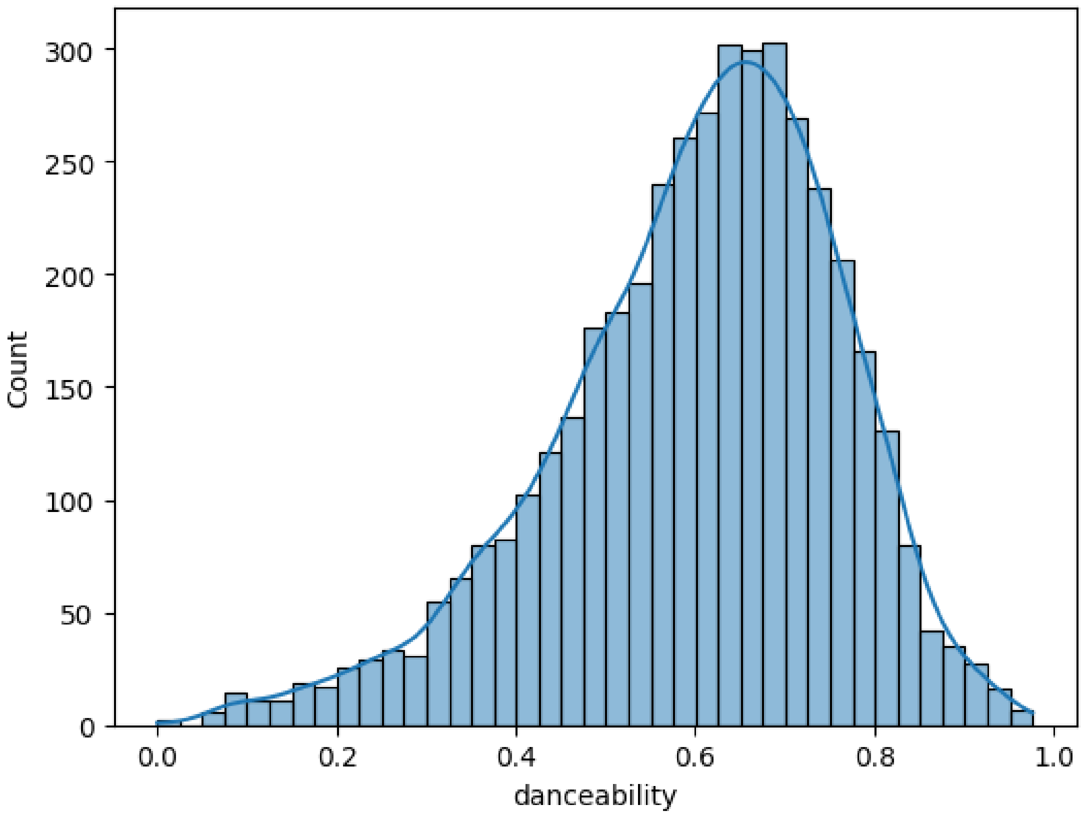
```python
# Creating a correlation matrix
correlation_matrix = d[numerical_columns].corr()

# Creating a heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
linewidths=.5)
plt.title('Correlation Heatmap of Numerical Columns')
plt.show()
```



Distribution

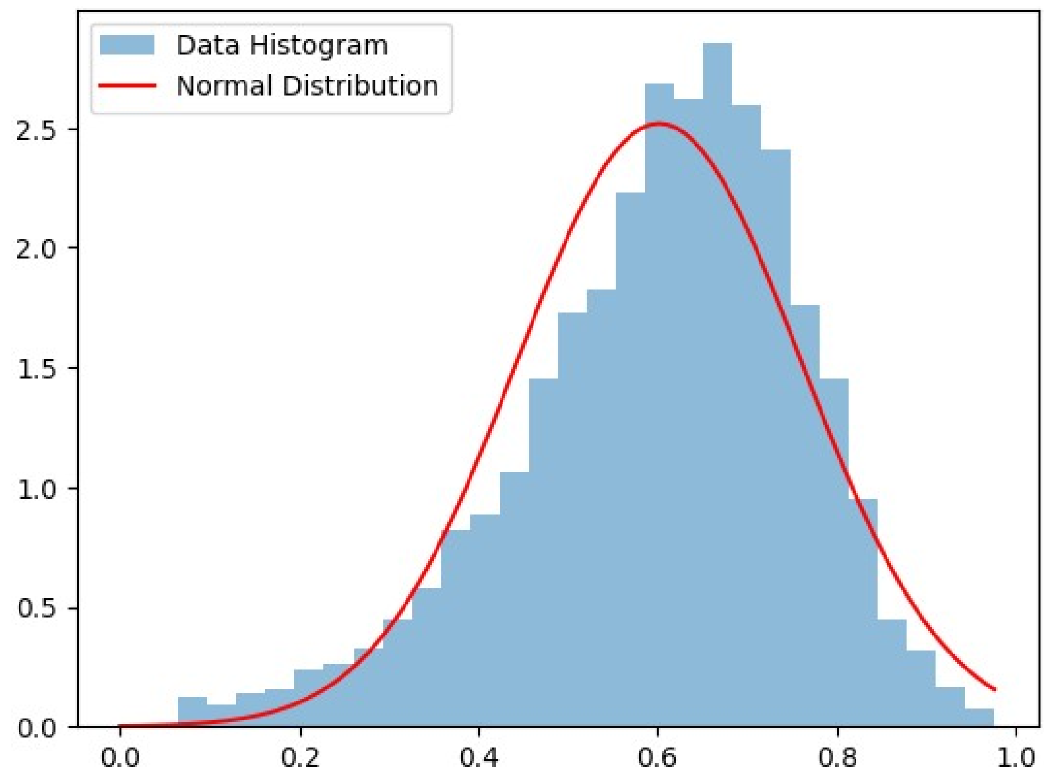
1. Visualzing the distribution of danceability

```
sns.histplot(d['danceability'],kde=True)
plt.show()
```



2. Plotting normal distribution over danceability

```
data = d['danceability']

# Visual inspection
plt.hist(data, bins=30, density=True, alpha=0.5, label='Data
Histogram')
x = np.linspace(min(data), max(data), 100)
plt.plot(x, stats.norm.pdf(x, np.mean(data), np.std(data)), 'r-',
label='Normal Distribution')
plt.legend()
plt.show()
```



As it can be visualized that the data is not following normal distribution

3. Checking skewedness of danceability column

```
# Check skewness
skewness = skew(data)
if skewness > 0:
    print(f"The data is right-skewed (positively skewed). Skewness value: {skewness:.2f}")
elif skewness < 0:
    print(f"The data is left-skewed (negatively skewed). Skewness value: {skewness:.2f}")
else:
    print("The data is approximately symmetric.")

The data is left-skewed (negatively skewed). Skewness value: -0.66
```

Therefore the distribution of data is right skewed distribution

Hypothesis Testing

1.Normality test using Shapiro-Wilk Test : tests If data is normally distributed

Lets assume that the distribution follows normal distribution

```
data = d['danceability']

stat, p = shapiro(data)

print('stat=%.20f, p=%.10f' % (stat, p))

if p > 0.05:
    print('Normal distribution')
else:
    print('Not a normal distribution')
```

```
stat=0.97378325462341308594, p=0.0000000000
Not a normal distribution
```

2.T Test

Lets assume mean age of casualty of both danceability and energy have no major difference

```
# Extract the samples
danceability_sample = d['danceability']  # Drop missing values if any
energy_sample = d['energy']  # Drop missing values if any

# Perform independent samples t-test
t_statistic, p_value = stats.ttest_ind(danceability_sample,
energy_sample)

# Display the results
print(f'T-statistic: {t_statistic}')
print(f'P-value: {p_value}')

# Check for significance
alpha = 0.05  # significance level
if p_value < alpha:
    print('Reject the null hypothesis. There is a significant
difference.')
else:
```

```python
    print('Fail to reject the null hypothesis. There is no significant difference.')
```

```
T-statistic: 7.928391882426873
P-value: 2.4986203431467006e-15
Reject the null hypothesis. There is a significant difference.
```