-21,

25 2024 .

1

:

```
df = pd.read_csv("D:/University/third_course/MTAD/lab1/Spotify_Youtube.csv")
df = df.drop(columns=["Unnamed: 0", "Url_spotify", "Uri", "Url_youtube"])
df = df.dropna()
```

: 1

```
for column in numeric_df.columns:
    lower_bound = numeric_df[column].quantile(0.01)
    upper_bound = numeric_df[column].quantile(0.99)

    initial_count = df_cleaned.shape[0]
    df_cleaned = df_cleaned[(df_cleaned[column] >= lower_bound) & (df_cleaned[column] <=
        upper_bound)]
    final_count = df_cleaned.shape[0]

    print(f"Column '{column}': removed {initial_count - final_count} rows.")
df_cleaned.reset_index()
print(f"Number of rows before removal: {df.shape[0]}")
print(f"Number of rows after removal: {df_cleaned.shape[0]}")
```

:

Danceability	376
Energy	321
Key	0
Loudness	203
Speechiness	359
Acousticness	230
Instrumentalness	69
Liveness	341
Valence	192
Tempo	266
Duration_ms	243
Views	291
Likes	75
Comments	50
Stream	177

: 19170

: 15977

2

```
def mean(column):
    return sum(column) / len(column)
```

```
def trimmed_mean(column, trim_percent):
    sorted_col = sorted(column)
    trim_count = int(len(sorted_col) * trim_percent)
    trimmed_col = sorted_col[trim_count:-trim_count]
    return sum(trimmed_col) / len(trimmed_col)
```

```python
def median(column):
    sorted_col = sorted(column)
    n = len(sorted_col)
    mid = n // 2
    if n % 2 == 0:
        return (sorted_col[mid - 1] + sorted_col[mid]) / 2
    else:
        return sorted_col[mid]
```

```python
def variance(column):
    column_mean = mean(column)
    return sum((x - column_mean) ** 2 for x in column) / (len(column) - 1)
```

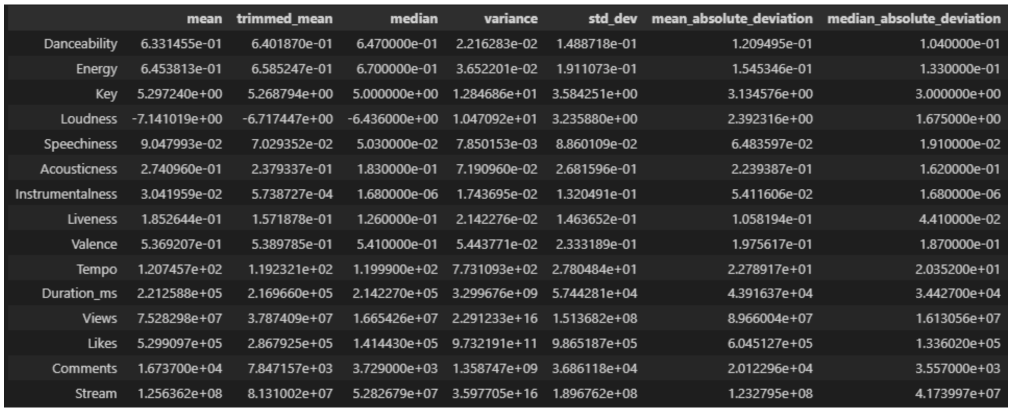
```python
def std_dev(column):
    return variance(column) ** 0.5
```

```python
def mean_absolute_deviation(column):
    column_mean = mean(column)
    return sum(abs(x - column_mean) for x in column) / len(column)
```

```python
def median_absolute_deviation(column):
    col_median = median(column)
    deviations = [abs(x - col_median) for x in column]
    return median(deviations)
```

```python
def calculate_statistics(df, columns = None, trim_percent=0.1):
    if isinstance(columns, str):
        columns = [columns]
    elif columns is None:
        columns = df.columns

    results = {}
    for column in columns:
        col_data = df[column].dropna().tolist()
        stats = {
            'mean': mean(col_data),
            'trimmed_mean': trimmed_mean(col_data, trim_percent),
            'median': median(col_data),
            'variance': variance(col_data),
            'std_dev': std_dev(col_data),
            'mean_absolute_deviation': mean_absolute_deviation(col_data),
            'median_absolute_deviation': median_absolute_deviation(col_data)
        }
        results[column] = stats
    results_df = pd.DataFrame(results).T
    return results_df
```



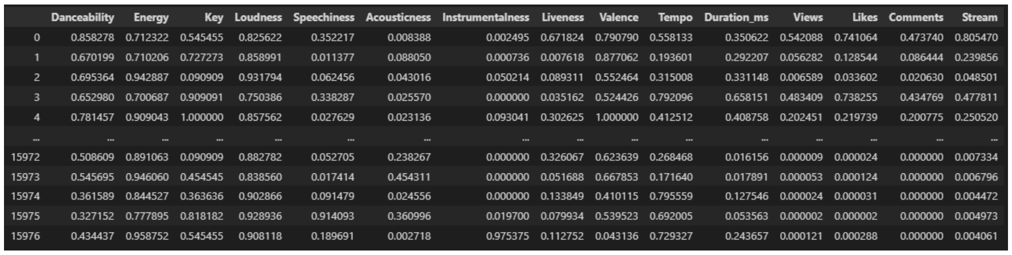
	mean	trimmed_mean	median	variance	std_dev	mean_absolute_deviation	median_absolute_deviation
Danceability	6.331455e-01	6.401870e-01	6.470000e-01	2.216283e-02	1.488718e-01	1.209495e-01	1.040000e-01
Energy	6.453813e-01	6.585247e-01	6.700000e-01	3.652201e-02	1.911073e-01	1.545346e-01	1.330000e-01
Key	5.297240e+00	5.268794e+00	5.000000e+00	1.284686e+01	3.584251e+00	3.134576e+00	3.000000e+00
Loudness	-7.141019e+00	-6.717447e+00	-6.436000e+00	1.047092e+01	3.235880e+00	2.392316e+00	1.675000e+00
Speechiness	9.047993e-02	7.029352e-02	5.030000e-02	7.850153e-03	8.860109e-02	6.483597e-02	1.910000e-02
Acousticness	2.740960e-01	2.379337e-01	1.830000e-01	7.190960e-02	2.681596e-01	2.239387e-01	1.620000e-01
Instrumentalness	3.041959e-02	5.738727e-04	1.680000e-06	1.743695e-02	1.320491e-01	5.411606e-02	1.680000e-06
Liveness	1.852644e-01	1.571878e-01	1.260000e-01	2.142276e-02	1.463652e-01	1.058194e-01	4.410000e-02
Valence	5.369207e-01	5.389785e-01	5.410000e-01	5.443771e-02	2.333189e-01	1.975617e-01	1.870000e-01
Tempo	1.207457e+02	1.192321e+02	1.199900e+02	7.731093e+02	2.780484e+01	2.278917e+01	2.035200e+01
Duration_ms	2.212588e+05	2.169660e+05	2.142270e+05	3.299676e+09	5.744281e+04	4.391637e+04	3.442700e+04
Views	7.528298e+07	3.787409e+07	1.665426e+07	2.291233e+16	1.513682e+08	8.966004e+07	1.613056e+07
Likes	5.299097e+05	2.867925e+05	1.414430e+05	9.732191e+11	9.865187e+05	6.045127e+05	1.336020e+05
Comments	1.673700e+04	7.847157e+03	3.729000e+03	1.358747e+09	3.686118e+04	2.012296e+04	3.557000e+03
Stream	1.256362e+08	8.131002e+07	5.282679e+07	3.597705e+16	1.896762e+08	1.232795e+08	4.173997e+07

. 1:

3

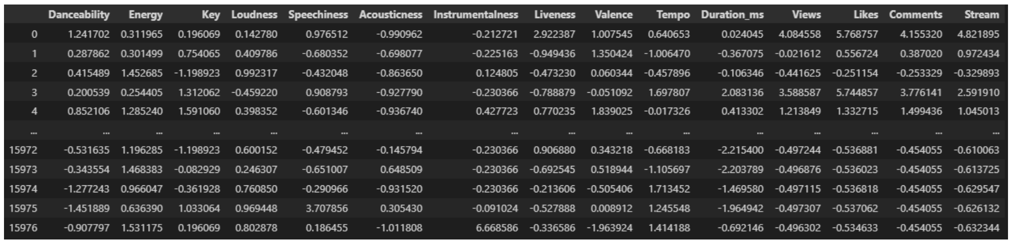
-

```
def min_max_normalization(df):
    normalized_df = df.copy()
    for column in normalized_df.columns:
        min_val = normalized_df[column].min()
        max_val = normalized_df[column].max()
        normalized_df[column] = (normalized_df[column] - min_val) / (max_val - min_val)
    return normalized_df
```



	Danceability	Energy	Key	Loudness	Speechiness	Acousticness	Instrumentalness	Liveness	Valence	Tempo	Duration_ms	Views	Likes	Comments	Stream
0	0.858278	0.712322	0.545455	0.825622	0.352217	0.008388	0.002495	0.671824	0.790790	0.558133	0.350622	0.542088	0.741064	0.473740	0.805470
1	0.670199	0.710206	0.727273	0.858991	0.011377	0.088050	0.000736	0.007618	0.877062	0.193601	0.292207	0.056282	0.128544	0.086444	0.239856
2	0.695364	0.942887	0.090909	0.931794	0.062456	0.043016	0.050214	0.089311	0.552464	0.315008	0.331148	0.006589	0.033602	0.020630	0.048501
3	0.652980	0.700687	0.909091	0.750386	0.338287	0.025570	0.000000	0.035162	0.524426	0.792096	0.658151	0.483409	0.738255	0.434769	0.477811
4	0.781457	0.909043	1.000000	0.857562	0.027629	0.023136	0.093041	0.302625	1.000000	0.412512	0.408758	0.202451	0.219739	0.200775	0.250520
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
15972	0.508609	0.891063	0.090909	0.882782	0.052705	0.238267	0.000000	0.326067	0.623639	0.268468	0.016156	0.000009	0.000024	0.000000	0.007334
15973	0.545695	0.946060	0.454545	0.838560	0.017414	0.454311	0.000000	0.051688	0.667853	0.171640	0.017891	0.000053	0.000124	0.000000	0.006796
15974	0.361589	0.844527	0.363636	0.902866	0.091479	0.024556	0.000000	0.133849	0.410115	0.795559	0.127546	0.000024	0.000031	0.000000	0.004472
15975	0.327152	0.777895	0.818182	0.928936	0.914093	0.360996	0.019700	0.079934	0.539523	0.692005	0.053563	0.000002	0.000002	0.000000	0.004973
15976	0.434437	0.958752	0.545455	0.908118	0.189691	0.002718	0.975375	0.112752	0.043136	0.729327	0.243657	0.000121	0.000288	0.000000	0.004061

. 2: MinMax

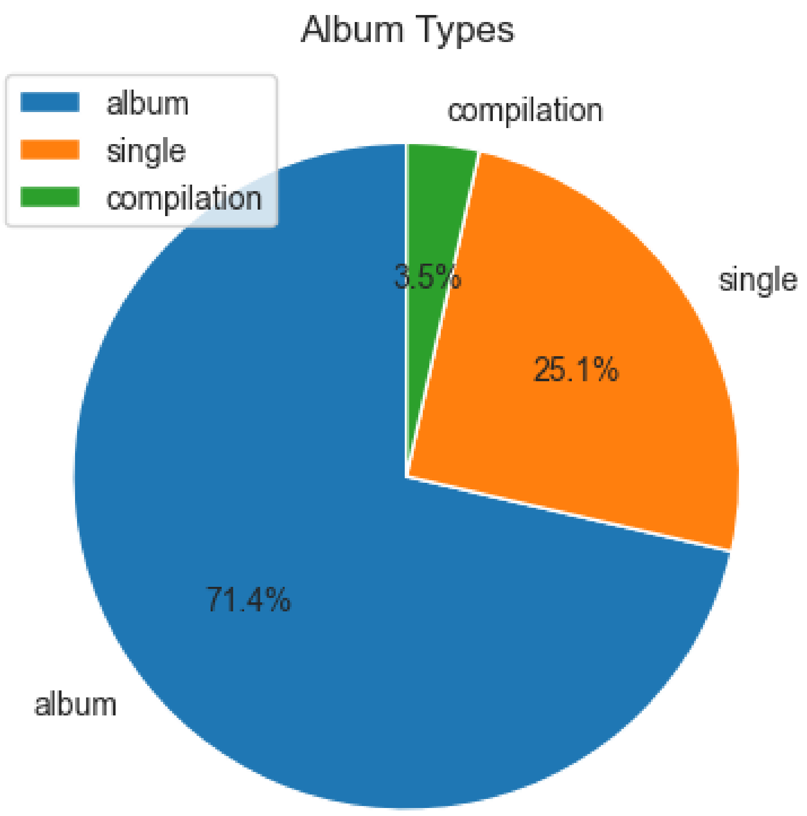


	Danceability	Energy	Key	Loudness	Speechiness	Acousticness	Instrumentalness	Liveness	Valence	Tempo	Duration_ms	Views	Likes	Comments	Stream
0	1.241702	0.311965	0.196069	0.142780	0.976512	-0.990962	-0.212721	2.922387	1.007545	0.640653	0.024045	4.084558	5.768757	4.155320	4.821895
1	0.287862	0.301499	0.754065	0.409786	-0.680352	-0.698077	-0.225163	-0.949436	1.350424	-1.006470	-0.367075	-0.021612	0.556724	0.387020	0.972434
2	0.415489	1.452685	-1.198923	0.992317	-0.432048	-0.863650	0.124805	-0.473230	0.060344	-0.457896	-0.106346	-0.441625	-0.251154	-0.253329	-0.329893
3	0.200539	0.254405	1.312062	-0.459220	0.908793	-0.927790	-0.230366	-0.788879	-0.051092	1.697807	2.083136	3.588587	5.744857	3.776141	2.591910
4	0.852106	1.285240	1.591060	0.398352	-0.601346	-0.936740	0.427723	0.770235	1.839025	-0.017326	0.413302	1.213849	1.332715	1.499436	1.045013
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
15972	-0.531635	1.196285	-1.198923	0.600152	-0.479452	-0.145794	-0.230366	0.906880	0.343218	-0.668183	-2.215400	-0.497244	-0.536881	-0.454055	-0.610063
15973	-0.343554	1.468383	-0.082929	0.246307	-0.651007	0.648509	-0.230366	-0.692545	0.518944	-1.105697	-2.203789	-0.496876	-0.536023	-0.454055	-0.613725
15974	-1.277243	0.966047	-0.361928	0.760850	-0.290966	-0.931520	-0.230366	-0.213606	-0.505406	1.713452	-1.469580	-0.497115	-0.536818	-0.454055	-0.629547
15975	-1.451889	0.636390	1.033064	0.969448	3.707856	0.305430	-0.091024	-0.527888	0.008912	1.245548	-1.964942	-0.497307	-0.537062	-0.454055	-0.626132
15976	-0.907797	1.531175	0.196069	0.802878	0.186455	-1.011808	6.668586	-0.336586	-1.963924	1.414188	-0.692146	-0.496302	-0.534633	-0.454055	-0.632344

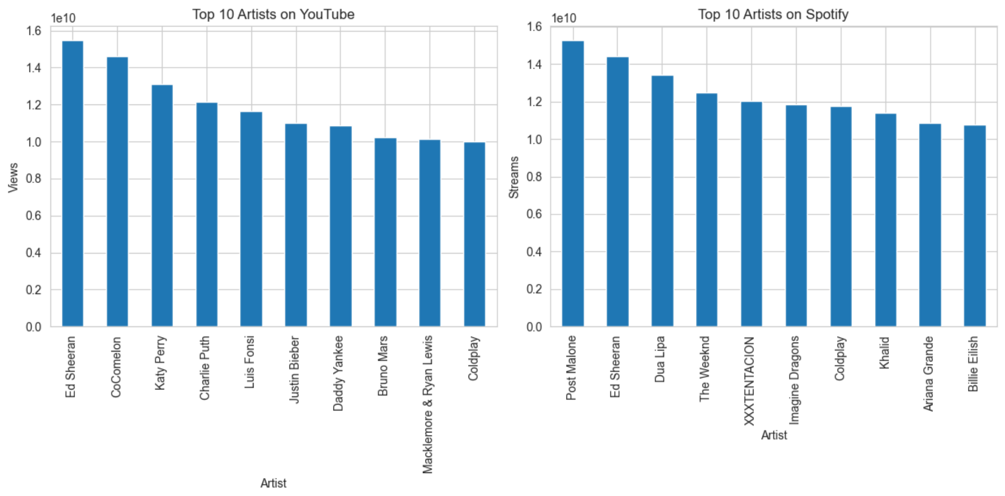
. 3:

```python
def mean_normalization(df):
    normalized_df = df.copy()
    for column in normalized_df.columns:
        mean_val = normalized_df[column].mean()
        std_dev = normalized_df[column].std()
        normalized_df[column] = (normalized_df[column] - mean_val) / std_dev
    return normalized_df
```

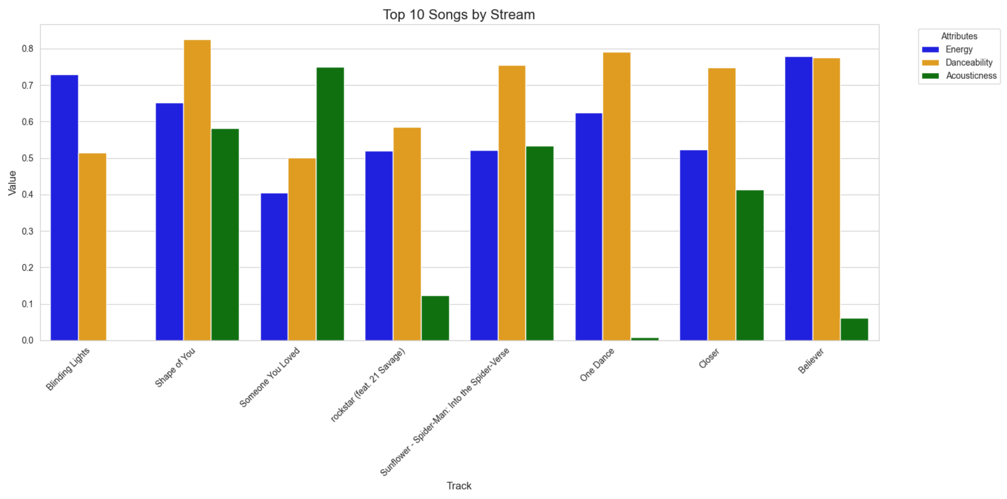
4



. 4


Top 10 Artists on YouTube
1e10
Views
Artist
Ed Sheeran
CoComelon
Katy Perry
Charlie Puth
Luis Fonsi
Justin Bieber
Daddy Yankee
Bruno Mars
Macklemore & Ryan Lewis
Coldplay
Top 10 Artists on Spotify
1e10
Streams
Artist
Post Malone
Ed Sheeran
Dua Lipa
The Weeknd
XXXTENTACION
Imagine Dragons
Coldplay
Khalid
Ariana Grande
Billie Eilish

. 5


Top 10 Songs by Stream
Value
Track
Attributes
Energy
Danceability
Acousticness
Blinding Lights
Shape of You
Someone You Loved
rockstar (feat. 21 Savage)
Sunflower - Spider-Man: Into the Spider-Verse
One Dance
Closer
Believer

. 6