```
import os
import pandas as pd
import numpy as np
```

```
spotify_df = pd.read_csv("C:\\Users\\DELL\\Downloads\\spotify\\spotify_songs.csv", encoding='latin1')
```

```
spotify_df.head()
```

	Track	Album Name	Artist	Release Date	ISRC	All Time Rank	Track Score	Spotify Streams	Spotify Playlist Count	Spotify Playlist Reach	...	SiriusXM Spins	Deezer Playlist Count	Deezer Playlist Reach	Amazo Playlis Coun
0	MILLION DOLLAR BABY	Million Dollar Baby - Single	Tommy Richman	4/26/2024	QM24S2402528	1	725.4	390,470,936	30,716	196,631,588	...	684	62.0	17,598,718	114.
1	Not Like Us	Not Like Us	Kendrick Lamar	5/4/2024	USUG12400910	2	545.9	323,703,884	28,113	174,597,137	...	3	67.0	10,422,430	111.
2	i like the way you kiss me	I like the way you kiss me	Artemas	3/19/2024	QZJ842400387	3	538.4	601,309,283	54,331	211,607,669	...	536	136.0	36,321,847	172.
3	Flowers	Flowers - Single	Miley Cyrus	1/12/2023	USSM12209777	4	444.9	2,031,280,633	269,802	136,569,078	...	2,182	264.0	24,684,248	210.
4	Houdini	Houdini	Eminem	5/31/2024	USUG12403398	5	423.3	107,034,922	7,223	151,469,874	...	1	82.0	17,660,624	105.

5 rows × 29 columns

```
spotify_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4600 entries, 0 to 4599
Data columns (total 29 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   Track                        4600 non-null   object 
 1   Album Name                   4600 non-null   object 
 2   Artist                       4595 non-null   object 
 3   Release Date                 4600 non-null   object 
 4   ISRC                         4600 non-null   object 
 5   All Time Rank                4600 non-null   object 
 6   Track Score                  4600 non-null   float64
 7   Spotify Streams              4487 non-null   object 
 8   Spotify Playlist Count       4530 non-null   object 
 9   Spotify Playlist Reach       4528 non-null   object 
 10  Spotify Popularity           3796 non-null   float64
 11  YouTube Views                4292 non-null   object 
 12  YouTube Likes                4285 non-null   object 
 13  TikTok Posts                 3427 non-null   object 
 14  TikTok Likes                 3620 non-null   object 
 15  TikTok Views                 3619 non-null   object 
 16  YouTube Playlist Reach       3591 non-null   object 
 17  Apple Music Playlist Count   4039 non-null   float64
 18  AirPlay Spins                4102 non-null   object 
 19  SiriusXM Spins               2477 non-null   object 
 20  Deezer Playlist Count        3679 non-null   float64
 21  Deezer Playlist Reach        3672 non-null   object 
 22  Amazon Playlist Count        3545 non-null   float64
 23  Pandora Streams              3494 non-null   object 
 24  Pandora Track Stations       3332 non-null   object 
 25  Soundcloud Streams           1267 non-null   object 
 26  Shazam Counts                4023 non-null   object 
 27  TIDAL Popularity             0 non-null      float64
 28  Explicit Track               4600 non-null   int64  
dtypes: float64(6), int64(1), object(22)
memory usage: 1.0+ MB
```

```
spotify_df.isna().sum()
```

```
Track                          0
Album Name                     0
Artist                         5
Release Date                   0
ISRC                           0
All Time Rank                  0
Track Score                    0
Spotify Streams              113
Spotify Playlist Count        70
Spotify Playlist Reach        72
Spotify Popularity           804
YouTube Views                308
YouTube Likes                315
TikTok Posts                1173
TikTok Likes                 980
TikTok Views                 981
YouTube Playlist Reach      1009
Apple Music Playlist Count   561
AirPlay Spins                498
SiriusXM Spins              2123
Deezer Playlist Count        921
Deezer Playlist Reach        928
Amazon Playlist Count       1055
Pandora Streams             1106
Pandora Track Stations      1268
Soundcloud Streams          3333
Shazam Counts                577
TIDAL Popularity            4600
Explicit Track                 0
dtype: int64
```

```
catcol = []
numcol = []
for i in spotify_df.columns:
    if spotify_df[i].dtype == 'object':
```

```
        catcol.append(i)
else:
        numcol.append(i)
```

```
print("Categorical Columns:", catcol)
print("Numerical Columns:", numcol)
```

```
Categorical Columns: ['Track', 'Album Name', 'Artist', 'Release Date', 'ISRC', 'All Time Rank', 'Spotify Streams', 'Spotify Playlist Count', 'Spot
Numerical Columns: ['Explicit Track']
```

```
spotify_df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Track Score	4600.0	41.844043	38.543766	19.4	23.3	29.9	44.425	725.4
Spotify Popularity	3796.0	63.501581	16.186438	1.0	61.0	67.0	73.000	96.0
Apple Music Playlist Count	4039.0	54.603120	71.612270	1.0	10.0	28.0	70.000	859.0
Deezer Playlist Count	3679.0	32.310954	54.274538	1.0	5.0	15.0	37.000	632.0
Amazon Playlist Count	3545.0	25.348942	25.989826	1.0	8.0	17.0	34.000	210.0
TIDAL Popularity	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Explicit Track	4600.0	0.358913	0.479734	0.0	0.0	0.0	1.000	1.0

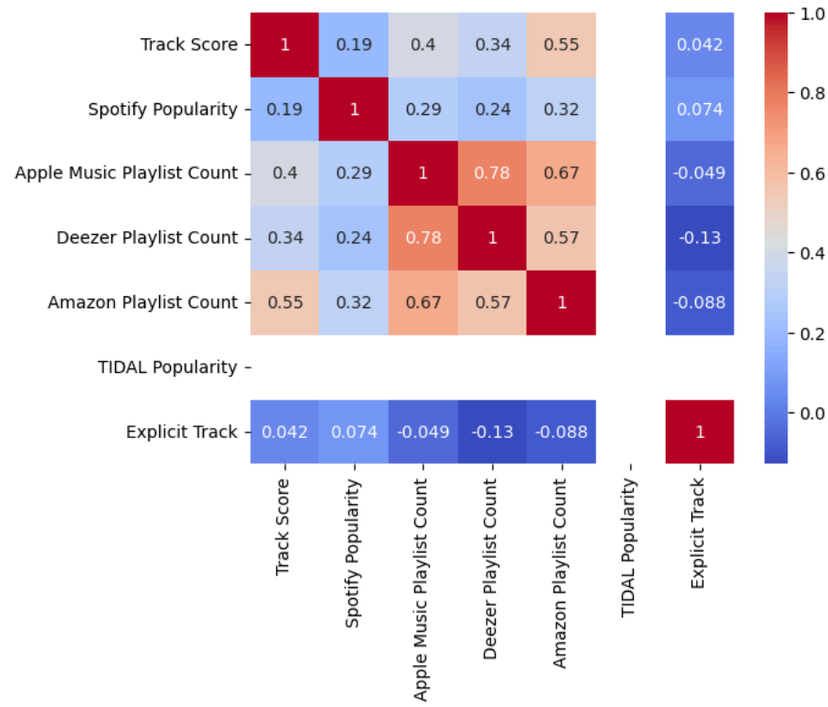
```
for i in catcol:
    print(i)
    print(spotify_df.value_counts())
    print("\n")
```

```
Track
Series([], Name: count, dtype: int64)


Album Name
Series([], Name: count, dtype: int64)


Artist
Series([], Name: count, dtype: int64)


Release Date
Series([], Name: count, dtype: int64)


ISRC
Series([], Name: count, dtype: int64)


All Time Rank
Series([], Name: count, dtype: int64)


Spotify Streams
Series([], Name: count, dtype: int64)


Spotify Playlist Count
Series([], Name: count, dtype: int64)


Spotify Playlist Reach
Series([], Name: count, dtype: int64)


YouTube Views
Series([], Name: count, dtype: int64)


YouTube Likes
Series([], Name: count, dtype: int64)


TikTok Posts
Series([], Name: count, dtype: int64)


TikTok Likes
Series([], Name: count, dtype: int64)


TikTok Views
Series([], Name: count, dtype: int64)


YouTube Playlist Reach
Series([], Name: count, dtype: int64)
```

```
catcol = spotify_df.select_dtypes(include=['object'])
numcol = spotify_df.select_dtypes(include=['float64', 'Int64'])
```

```
import matplotlib.pyplot as plt
import seaborn as sns
sns.heatmap(numcol.corr(), annot=True, cmap='coolwarm')
```

```
<Axes: >
```




```
from sklearn.impute import SimpleImputer
import pandas as pd
catcol_array = np.array(catcol).reshape(-1,1)
```

```
from sklearn.impute import SimpleImputer
imp_median = SimpleImputer(missing_values = np.nan, strategy = 'median')
imp_mode = SimpleImputer(missing_values = np.nan, strategy = 'most_frequent')
```

```
cat_df = pd.DataFrame(imp_mode.fit_transform(catcol_array), columns=['catcol.columns'])
```

```
numcol = numcol.dropna(axis = 1, how = 'all')
num_df = pd.DataFrame(imp_median.fit_transform(numcol), columns=numcol.columns)
```

```
final_df = pd.concat([cat_df, num_df], axis=1)
```

```
final_df.head()
```

	catcol.columns	Track Score	Spotify Popularity	Apple Music Playlist Count	Deezer Playlist Count	Amazon Playlist Count	Explicit Track
0	MILLION DOLLAR BABY	725.4	92.0	210.0	62.0	114.0	0.0
1	Million Dollar Baby - Single	545.9	92.0	188.0	67.0	111.0	1.0
2	Tommy Richman	538.4	92.0	190.0	136.0	172.0	0.0
3	4/26/2024	444.9	85.0	394.0	264.0	210.0	0.0

```
from sklearn.preprocessing import OneHotEncoder
encoder = OneHotEncoder(handle_unknown='ignore', max_categories=100, sparse_output=False)
encoded_cat = encoder.fit_transform(cat_df)
encoded_cat_df = pd.DataFrame(
    encoded_cat,
    columns=encoder.get_feature_names_out(cat_df.columns)
)
```

```
final_df = pd.concat([encoded_cat_df, num_df], axis=1)
```

```
final_df.head()
```

	catcol.columns_1	catcol.columns_1,097	catcol.columns_1/1/2011	catcol.columns_1/1/2012	catcol.columns_1/1/2013	catcol.columns_1/1/2014	cat
0	0.0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	

5 rows × 106 columns

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaled_df = pd.DataFrame(scaler.fit_transform(num_df), columns=num_df.columns)
scaled_df
```

	Track Score	Spotify Popularity	Apple Music Playlist Count	Deezer Playlist Count	Amazon Playlist Count	Explicit Track
0	17.736468	1.889107	2.344766	0.676311	3.923850	-0.748232
1	13.078918	1.889107	2.019599	0.778303	3.793872	1.336484
2	12.884313	1.889107	2.049160	2.185797	6.436754	-0.748232
3	10.458235	1.414916	5.064340	4.796799	8.083139	-0.748232
4	9.897773	1.618141	1.930917	1.084280	3.533917	1.336484
...	...	...	...	...	...	...
4595	-0.582364	0.466534	-0.714756	-0.547597	-0.278765	1.336484
4596	-0.582364	-0.549591	-0.744316	-0.567995	-0.278765	-0.748232
4597	-0.582364	0.060084	-0.478271	-0.567995	-0.755350	1.336484
4598	-0.582364	0.127825	-0.744316	-0.282417	-0.712024	-0.748232
4599	-0.582364	-0.007658	-0.596514	-0.506800	-0.842002	1.336484

4600 rows × 6 columns

```
from sklearn.cluster import KMeans
```

```
k_means = KMeans(n_clusters=5)
k_means.fit(scaled_df)
```

```
KMeans(n_clusters=5)
```

```
k_means.labels_
```

```
array([4, 4, 4, ..., 1, 3, 1], shape=(4600,), dtype=int32)
```

```
k_means.inertia_
```

```
11005.118836530983
```

```
wss = []
```

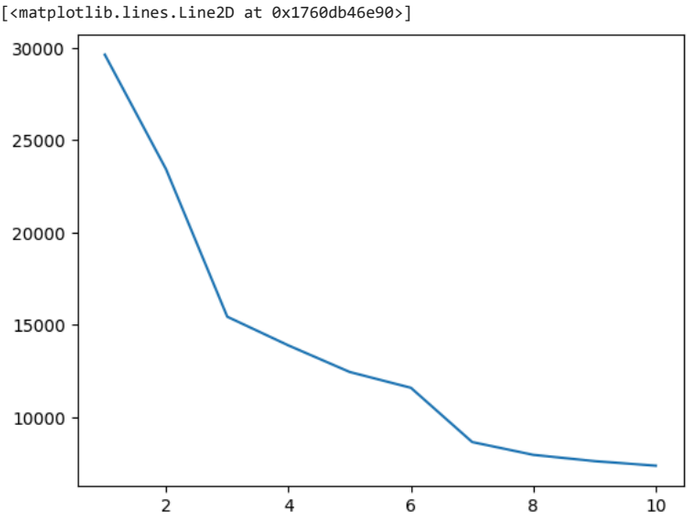
```
for i in range(1,11):
    KM = KMeans(n_clusters=i)
    KM.fit(scaled_df)
    wss.append(KM.inertia_)
```

```
wss
```

```
[29627.471543922205,
 23428.097552120715,
 15446.163198799888,
 13891.742375203605,
 12454.962425764083,
 11599.317251421517,
 8665.220210326313,
 7970.450628290345,
 7630.797142343946,
 7384.523928937054]
```

```
import matplotlib.pyplot as plt
plt.plot(range(1,11), wss)
```

```
[<matplotlib.lines.Line2D at 0x1760db46e90>]
```




```
labels = k_means.labels_
scaled_df["clus_kmeans"] = labels
```

```
scaled_df.head()
```

	Track Score	Spotify Popularity	Apple Music Playlist Count	Deezer Playlist Count	Amazon Playlist Count	Explicit Track	clus_kmeans	sil_samples	silhouette_score
0	17.736468	1.889107	2.344766	0.676311	3.923850	-0.748232	4	0.104011	0.410725

```
from sklearn.metrics import silhouette_score, silhouette_samples
```

```
silhouette_score = silhouette_score(scaled_df, labels)
```

```
silhouette_score
```

```
np.float64(0.5268552206496828)
```

```
sil_samples = silhouette_samples(scaled_df, labels)
```

```
print(sil_samples)
```

```
[0.1252071  0.13576819 0.18835165 ... 0.64839784 0.61687455 0.63573546]
```

```
scaled_df["sil_samples"] = sil_samples
scaled_df["silhouette_score"] = silhouette_score
```

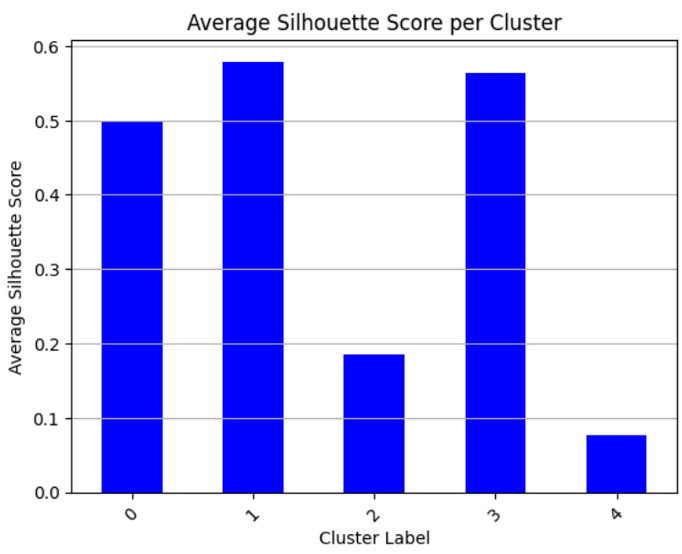
```
scaled_df.head()
```

	Track Score	Spotify Popularity	Apple Music Playlist Count	Deezer Playlist Count	Amazon Playlist Count	Explicit Track	clus_kmeans	sil_samples	silhouette_score
0	17.736468	1.889107	2.344766	0.676311	3.923850	-0.748232	4	0.125207	0.526855
1	13.078918	1.889107	2.019599	0.778303	3.793872	1.336484	4	0.135768	0.526855
2	12.884313	1.889107	2.049160	2.185797	6.436754	-0.748232	4	0.188352	0.526855
3	10.458235	1.414916	5.064340	4.796799	8.083139	-0.748232	4	0.269046	0.526855
4	9.897773	1.618141	1.930917	1.084280	3.533917	1.336484	4	0.121191	0.526855

```
silhouette_samples(scaled_df, labels).min()
```

```
np.float64(-0.19078597355564067)
```

```
import matplotlib.pyplot as plt
cluster_avg_sil = scaled_df.groupby('clus_kmeans')['sil_samples'].mean()
cluster_avg_sil.plot(kind='bar', color='blue')
plt.title("Average Silhouette Score per Cluster")
plt.xlabel("Cluster Label")
plt.ylabel("Average Silhouette Score")
plt.xticks(rotation=45)
plt.grid(axis='y')
plt.show()
```




```
scaled_df.to_csv('kmeansspotify.csv')
```

```
#i used sil score to evaluate clustering quality. i also visualized the avg sil score per cluster using a bar chart-- higher scores means better-
#separated cluster.
```