Spotify - Recommendations based on Energy levels

The recommendation system presented below was designed thinking about people who have an active life and would like a playlist to boost their workouts.

The libraries used for the project are listed below:

- Pandas
- Numpy
- Matplotlib
- Seaborn
- SKLearn
- SKImage

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import numpy as np
from sklearn.preprocessing import OneHotEncoder
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from skimage import io
```

The DataFrame (`df`) is created from the file 'data_moods.csv' and assigned to the variable `file`.

For better analysis we will display the number of rows columns before further handling of the data

In [2]:
```
file = 'data_moods.csv'
df = pd.read_csv(file)
print(f"Number of rows: {df.shape[0]}\nNumber of columns: {df.shape[1]}")
```

```
Number of rows: 686
Number of columns: 19
```

Now, the number os columns displayed will adjusted and after that, the head of our DataFrame

In [3]:
```
pd.set_option('display.max_columns',19)
df.head(10)
```

Out[3]:

	name	album	artist	id	release_date	popularity	length	danceability	acousticness	energy	inst
0	1999	1999	Prince	2H7PHVdQ3mXqEHXcvclTB0	1982-10-27	68	379266	0.866	0.137000	0.7300	
1	23	23	Blonde Redhead	4HIwL9ii9CcXpTOTzMq0MP	2007-04-16	43	318800	0.381	0.018900	0.8320	
2	9 Crimes	9	Damien Rice	5GZEeowhvSieFDiR8fQ2im	2006-11-06	60	217946	0.346	0.913000	0.1390	
3	99 Luftballons	99 Luftballons	Nena	6HA97v4wEGQ5TUClRM0XLc	1984-08-21	2	233000	0.466	0.089000	0.4380	
4	A Boy Brushed Red Living In Black And White	They're Only Chasing Safety	Underoath	47IWLfIKOKhFnz1FUEUIkE	2004-01-01	60	268000	0.419	0.001710	0.9320	
5	A Burden to Bear	A Burden to Bear	Emmanuelle Rimbaud	67DOFCrkcQaLp5yhzF8Y8N	2020-07-31	27	129410	0.394	0.995000	0.0475	
6	A La Plage	A La Plage	Ron Adelaar	79NmiFAgcXUIVDGfCWDdWF	2020-08-07	29	141888	0.504	0.994000	0.0584	
7	A Little Less Conversation - JXL Radio Edit Remix	Elvis 75 - Good Rockin' Tonight	Elvis Presley	4l2hnfUx0esSbITQa7iJt0	2009-12-28	1	211173	0.586	0.000155	0.9350	
8	A Place for My Head	Hybrid Theory (Bonus Edition)	Linkin Park	5rAxhWcgFng3s570sGO2F8	2000-10-24	68	184640	0.603	0.014400	0.9080	
9	ATTACK	A Beautiful Lie + 30 Seconds To Mars	Thirty Seconds To Mars	6QxTWEvzcJljVZaeTzuHF1	2007-05-15	0	189200	0.331	0.003440	0.8760	

Not every column is relevant for the intended purposes, so the following ones will be dropped: `'release_date'`, `'length'` and `'key'`.

The new number of columns will also be diplayed.

In [4]:
```
df.drop(['release_date', 'length','key'], axis=1, inplace=True)
print(f"Number of columns: {df.shape[1]}")
df.head(10)
```

```
Number of columns: 16
```

Out[4]:

	name	album	artist	id	popularity	danceability	acousticness	energy	instrumentalness	liveness
0	1999	1999	Prince	2H7PHVdQ3mXqEHXcvclTB0	68	0.866	0.137000	0.7300	0.000000	0.0843
1	23	23	Blonde Redhead	4HIwL9ii9CcXpTOTzMq0MP	43	0.381	0.018900	0.8320	0.196000	0.1530
2	9 Crimes	9	Damien Rice	5GZEeowhvSieFDiR8fQ2im	60	0.346	0.913000	0.1390	0.000077	0.0934
3	99 Luftballons	99 Luftballons	Nena	6HA97v4wEGQ5TUClRM0XLc	2	0.466	0.089000	0.4380	0.000006	0.1130
4	A Boy Brushed Red Living In Black And White	They're Only Chasing Safety	Underoath	47IWLfIKOKhFnz1FUEUIkE	60	0.419	0.001710	0.9320	0.000000	0.1370
5	A Burden to Bear	A Burden to Bear	Emmanuelle Rimbaud	67DOFCrkcQaLp5yhzF8Y8N	27	0.394	0.995000	0.0475	0.955000	0.1050
6	A La Plage	A La Plage	Ron Adelaar	79NmiFAgcXUIVDGfCWDdWF	29	0.504	0.994000	0.0584	0.956000	0.1150
7	A Little Less Conversation - JXL Radio Edit Remix	Elvis 75 - Good Rockin' Tonight	Elvis Presley	4l2hnfUx0esSbITQa7iJt0	1	0.586	0.000155	0.9350	0.277000	0.1590
8	A Place for My Head	Hybrid Theory (Bonus Edition)	Linkin Park	5rAxhWcgFng3s570sGO2F8	68	0.603	0.014400	0.9080	0.000000	0.6710
9	ATTACK	A Beautiful Lie + 30 Seconds	Thirty Seconds To Mars	6QxTWEvzcJljVZaeTzuHF1	0	0.331	0.003440	0.8760	0.000835	0.7320

Let's now check the existence of null values and the types of data within the DataFrame

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
name              0
album             0
artist            0
id                0
popularity        0
danceability      0
acousticness      0
energy            0
instrumentalness  0
liveness          0
valence           0
loudness          0
speechiness       0
tempo             0
time_signature    0
mood              0
dtype: int64
```

In [6]:
```
df.dtypes
```

Out[6]:
```
name               object
album              object
artist             object
id                 object
popularity          int64
danceability      float64
acousticness      float64
energy            float64
instrumentalness  float64
liveness          float64
valence           float64
loudness          float64
speechiness       float64
tempo             float64
time_signature      int64
mood               object
dtype: object
```

No `NaN` were found, but the `mood` column presents categorical values that would certainly impact the recommendation system. A new dataframe (`df2`) will be created with the `mood` column dropped from the original dataframe and with the `OneHotEncoder` technique will assign numerical values to the moods of the songs and concatenate the results with `df2` . The resulting DataFrame will be assigned to `df_moods`

In [7]:
```
df2 = df.drop('mood', axis=1)
ohe = OneHotEncoder(dtype=int)
mood_columns = ohe.fit_transform(df[['mood']]).toarray()
```

In [8]:
```
ohe.get_feature_names_out(['mood'])
```

Out[8]:
```
array(['mood_Calm', 'mood_Energetic', 'mood_Happy', 'mood_Sad'],
      dtype=object)
```

In [9]:
```
df_moods = pd.concat([df2,pd.DataFrame(mood_columns, columns = ohe.get_feature_names_out(['mood']))], axis=1)
print(f"Number of columns Mood DataFrame: {df_moods.shape[1]}")
df_moods.head(10)
```

```
Number of columns Mood DataFrame: 19
```

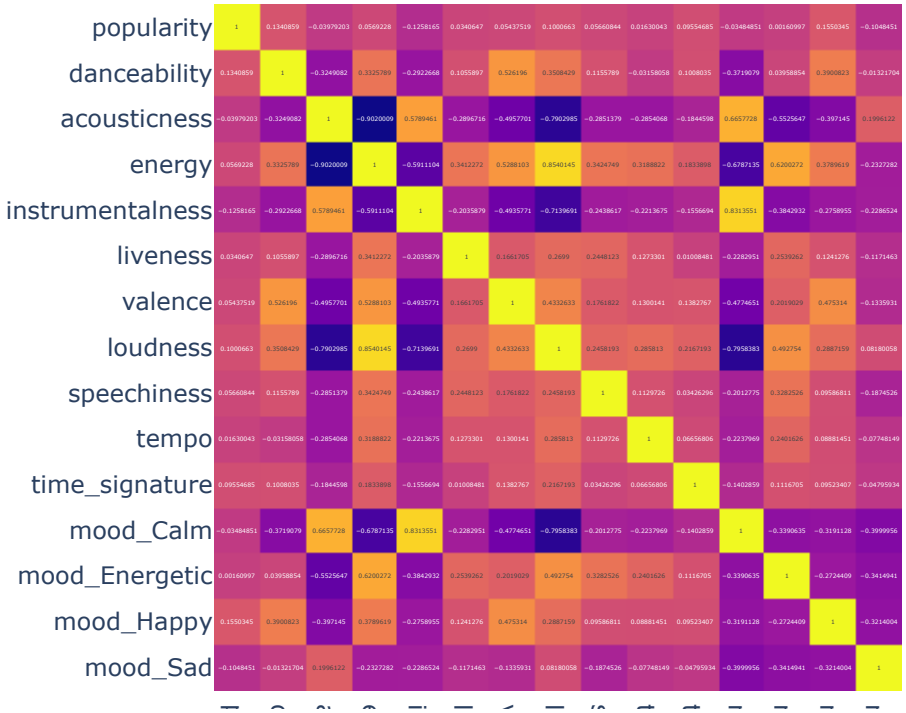
Out[9]:

	name	album	artist	id	popularity	danceability	acousticness	energy	instrumentalness	liveness
0	1999	1999	Prince	2H7PHVdQ3mXqEHXcvclTB0	68	0.866	0.137000	0.7300	0.000000	0.0843
1	23	23	Blonde Redhead	4HIwL9ii9CcXpTOTzMq0MP	43	0.381	0.018900	0.8320	0.196000	0.1530
2	9 Crimes	9	Damien Rice	5GZEeowhvSieFDiR8fQ2im	60	0.346	0.913000	0.1390	0.000077	0.0934
3	99 Luftballons	99 Luftballons	Nena	6HA97v4wEGQ5TUClRM0XLc	2	0.466	0.089000	0.4380	0.000006	0.1130
4	A Boy Brushed Red Living In Black And White	They're Only Chasing Safety	Underoath	47IWLfIKOKhFnz1FUEUIkE	60	0.419	0.001710	0.9320	0.000000	0.1370
5	A Burden to Bear	A Burden to Bear	Emmanuelle Rimbaud	67DOFCrkcQaLp5yhzF8Y8N	27	0.394	0.995000	0.0475	0.955000	0.1050
6	A La Plage	A La Plage	Ron Adelaar	79NmiFAgcXUIVDGfCWDdWF	29	0.504	0.994000	0.0584	0.956000	0.1150
7	A Little Less Conversation - JXL Radio Edit Remix	Elvis 75 - Good Rockin' Tonight	Elvis Presley	4l2hnfUx0esSbITQa7iJt0	1	0.586	0.000155	0.9350	0.277000	0.1590
8	A Place for My Head	Hybrid Theory (Bonus Edition)	Linkin Park	5rAxhWcgFng3s570sGO2F8	68	0.603	0.014400	0.9080	0.000000	0.6710
9	ATTACK	A Beautiful Lie + 30 Seconds	Thirty Seconds To Mars	6QxTWEvzcJljVZaeTzuHF1	0	0.331	0.003440	0.8760	0.000835	0.7320

Now that we have a dataframe, let's make some Exploratory Data Analysis, with the `energy` feature as the guideline, to understand how it relates to other features in the dataframe.

Let's start with a Correlation Matrix

In [10]:
```
corr_matrix = px.imshow(df_moods.corr(), text_auto = True)
corr_matrix.show()
```



My inital assumption was confirmed by the Correlation Matrix. Loudness is the percevied intensity, the amplitude of sound, so the louder a song is, most likely it is to be an energetic song.

Acoustic songs tend to be more mellow or intimistic. Hence, there is a clear **negative linear correlation** between `acousticness` and `energy`, as the the correlation dataframes and `regplot` graphs show below.

In [11]:
```
df[['loudness', 'energy']].corr().iloc[0].to_frame()
```

Out[11]:

	loudness
loudness	1.000000
energy	0.854015

In [12]:
```
df[['acousticness', 'energy']].corr().iloc[0].to_frame()
```

Out[12]:

	acousticness
acousticness	1.000000
energy	-0.902001

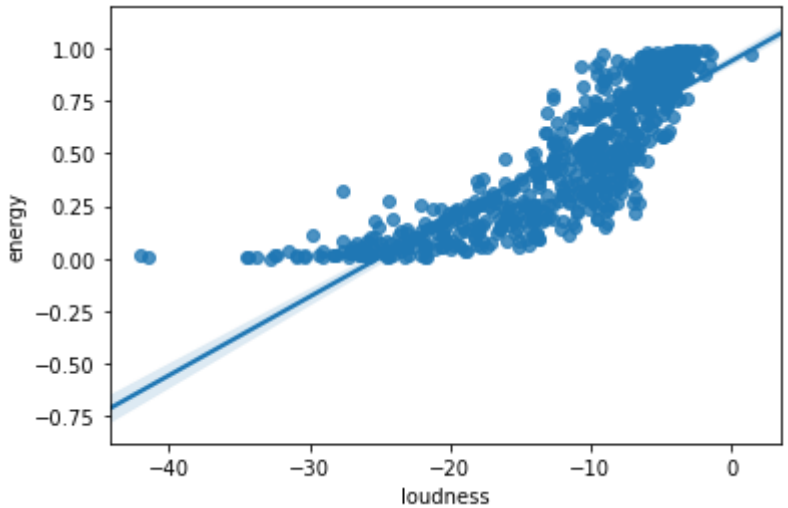
In [13]:
```
df[['loudness', 'acousticness']].corr().iloc[0].to_frame()
```

Out[13]:

	loudness
loudness	1.000000
acousticness	-0.790299

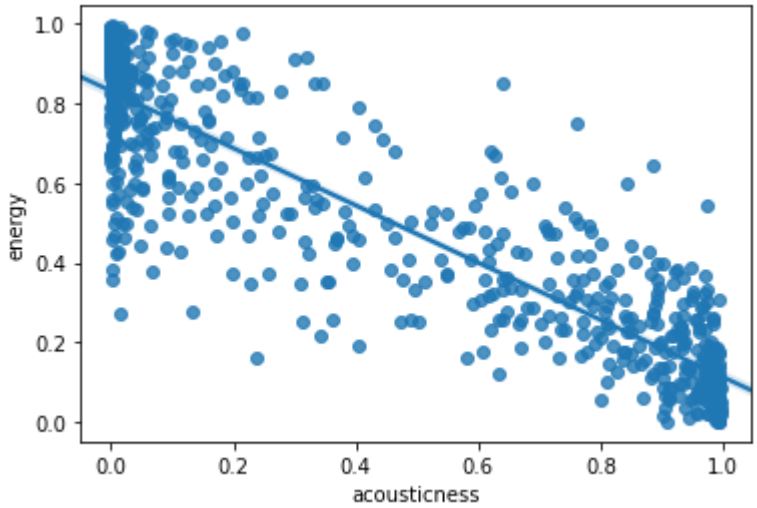
In [14]:
```
sns.regplot(x='loudness', y='energy', data=df2)
```

Out[14]:
```
<AxesSubplot:xlabel='loudness', ylabel='energy'>
```



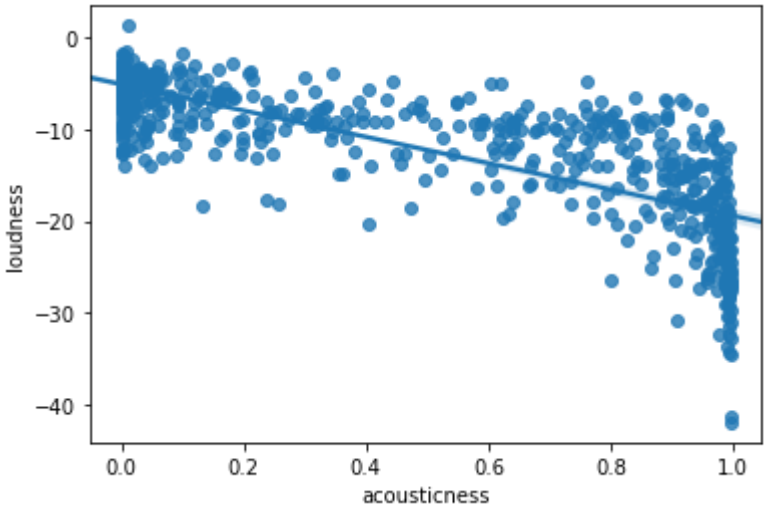
In [15]:
```
sns.regplot(x='acousticness', y='energy', data=df2)
```

Out[15]:
```
<AxesSubplot:xlabel='acousticness', ylabel='energy'>
```



In [16]:
```
sns.regplot(x='acousticness', y='loudness', data=df2)
```

Out[16]:
```
<AxesSubplot:xlabel='acousticness', ylabel='loudness'>
```



After this short data analysis, it is necessary to build a Pipeline to standardize the data through `StandardScaler` and make the Components Analysis through `PCA` technique. We will use 70% of our dataframe in order to avoid overfitting.

For the purposes of reproduction of the DataFrame, the `random_state` parameter will be define as `1234` and assigned to the variable seed

In [17]:
```
SEED = 1234
np.random.seed(1234)
pca_pipeline = Pipeline([('scaler',StandardScaler()),
                         ('PCA', PCA(n_components=0.7, random_state=SEED))])
```

In [18]:
```
df_moods
```

Out[18]:

	name	album	artist	id	popularity	danceability	acousticness	energy	instrumentalness	liveness
0	1999	1999	Prince	2H7PHVdQ3mXqEHXcvclTB0	68	0.866	0.13700	0.7300	0.000000	0.084
1	23	23	Blonde Redhead	4HIwL9ii9CcXpTOTzMq0MP	43	0.381	0.01890	0.8320	0.196000	0.153
2	9 Crimes	9	Damien Rice	5GZEeowhvSieFDiR8fQ2im	60	0.346	0.91300	0.1390	0.000077	0.093
3	99 Luftballons	99 Luftballons	Nena	6HA97v4wEGQ5TUClRM0XLc	2	0.466	0.08900	0.4380	0.000006	0.113
4	A Boy Brushed Red Living In Black And White	They're Only Chasing Safety	Underoath	47IWLfIKOKhFnz1FUEUIkE	60	0.419	0.00171	0.9320	0.000000	0.137
...	...	...	...	...	...	...	...	...	...	...
681	windcatcher	windcatcher	Leo Nocta	59VApBbrS2IADQk4ml5mdo	36	0.402	0.96100	0.2360	0.919000	0.092
682	yellow is the color of her eyes	yellow is the color of her eyes	Soccer Mommy	4D3nttJPU6L0M2epr7sId6	5	0.452	0.75700	0.5150	0.120000	0.140
683	you broke me first	you broke me first	Tate McRae	45bE4HXI0AwGZXfZtMp8JR	87	0.642	0.78600	0.3740	0.000000	0.090
684	you were good to me	brent	Jeremy Zucker	4CxFN5zON70B3VOPBYbd6P	76	0.561	0.91300	0.0848	0.000026	0.112
685	æfre	æfre	praam	2irbT1BSYaIEF44PlyKaoM	41	0.377	0.99400	0.0156	0.881000	0.099

686 rows × 19 columns

The DataFrame `projection` will be the created from the outcome of `df_moods` (`mood_embedding_pca`) through the Pipeline

In [19]:
```
mood_embedding_pca = pca_pipeline.fit_transform(df_moods.drop(['id','artist','album','name'], axis=1))
projection = pd.DataFrame(mood_embedding_pca)
```

In [20]:
```
pca_pipeline[1].n_components_
```

Out[20]: 5

In [21]:
```
mood_embedding_pca
```

Out[21]:
```
array([[-2.39157324, -1.36518263,  2.51029634, -0.25973101,  0.34577779],
       [-0.83401561, -0.78906698, -1.69500695, -0.44808354, -0.02710972],
       [ 1.31937135, -1.40674725, -1.61403318, -1.15679956,  0.55715111],
       ...,
       [ 0.18113985, -1.78669284, -0.83285626, -1.46300036,  1.77506574],
       [ 1.0769633 , -2.03367985, -0.76216052, -1.06680354,  1.69324362],
       [ 3.63169944,  1.02318233,  0.53780978, -0.63010035, -0.44312229]])
```

In [22]:
```
projection
```

Out[22]:

	0	1	2	3	4
0	-2.391573	-1.365183	2.510296	-0.259731	0.345778
1	-0.834016	-0.789067	-1.695007	-0.448084	-0.027110
2	1.319371	-1.406747	-1.614033	-1.156800	0.557151
3	-1.481015	-0.196788	0.733334	0.574310	-2.701284
4	-2.743797	1.896716	-0.969154	-0.973701	0.061559
...	...	...	...	...	...
681	3.055752	1.109257	0.404262	1.035114	-0.051757
682	0.573026	-1.675048	-1.760564	0.542327	-0.412348
683	0.181140	-1.786693	-0.832856	-1.463000	1.775066
684	1.076963	-2.033680	-0.762161	-1.066804	1.693244
685	3.631699	1.023182	0.537810	-0.630100	-0.443122

686 rows × 5 columns

In [23]:
```
print(f"Number of Rows: {projection.shape[0]}\nNumber of columns: {projection.shape[1]}")
```

```
Number of Rows: 686
Number of columns: 5
```

n_cluster validation

Once the DataFrame is ready, we need to set an optimal number of clusters for our recommendantion system. For that, the Sillhouette Score will be used as reference, along with a random dataframe assigned to the variable `random_data`, which contains the same number of rows and columns as `projection`

In [24]:
```
random_data = np.random.rand(686,5)
print(f"Number of Rows: {random_data.shape[0]}\nNumber of columns: {random_data.shape[1]}")
```

```
Number of Rows: 686
Number of columns: 5
```

In [25]:
```
random_data
```

Out[25]:
```
array([[0.19151945, 0.62210877, 0.43772774, 0.78535858, 0.77997581],
       [0.27259261, 0.27646426, 0.80187218, 0.95813935, 0.87593263],
       [0.35781727, 0.50099513, 0.68346294, 0.71270203, 0.37025075],
       ...,
       [0.01314559, 0.20132326, 0.29192121, 0.16010375, 0.48397445],
       [0.64563914, 0.41341197, 0.61103894, 0.58554597, 0.84897027],
       [0.52327112, 0.81164891, 0.51784942, 0.94750604, 0.14352508]])
```

Why Silhouette Score?

The main goal of the project is recommend songs for a specific moment. In this case, physical activities. Hence, the songs recommended from the user's choice should have similiar characteristics. The Sillhouette Score indicates the similiarity of an object compared to its own cluster, compared to other clusters.

Values range from `-1` to `1`, where the higher the score, the better is the clustering

In [26]:
```
from sklearn.metrics import silhouette_score as s_score
```

The function `clustering_validation` will receive as parameter the number os cluster to be tested (`n_clusters`) and the dataset (`dataset`)

In [27]:
```
def clustering_validation(n_clusters, dataset):
    kmeans = KMeans(n_clusters = n_clusters, n_init = 10, max_iter = 300, random_state=SEED)
    labels = kmeans.fit_predict(dataset)
```

```
    s = s_score(dataset, labels, metric='euclidean')
    return s
```

We will assign the following values to `n_clusters` : `5` , `20` and `50` , comparing `projection` to `random_data` .

We're looking for the highest score in order to determine our number of clusters.

In [28]:
```
s_moods_5 = clustering_validation(5,projection)
s_random_5 = clustering_validation(5,random_data)
print(f"Projection Silhouette Score: {s_moods_5}")
print(f"Random Data Sillhouete Score: {s_random_5}")
```

```
Projection Silhouette Score: 0.48615262675615684
Random Data Sillhouete Score: 0.16784310778369008
```

In [29]:
```
s_moods_20 = clustering_validation(20, projection)
s_random_20 = clustering_validation(20,random_data)
print(f"Projection Silhouette Score: {s_moods_20}")
print(f"Random Data Sillhouete Score: {s_random_20}")
```

```
Projection Silhouette Score: 0.25185099192485444
Random Data Sillhouete Score: 0.18842651179332343
```

In [30]:
```
s_moods_50 = clustering_validation(50,projection)
s_random_50 = clustering_validation(50,random_data)
print(f"Projection Silhouette Score: {s_moods_50}")
print(f"Random Data Sillhouete Score: {s_random_50}")
```

```
Projection Silhouette Score: 0.2211845150183364
Random Data Sillhouete Score: 0.18288660707160803
```

The optimal number of clusters is `5` . Important to notice the significant difference between our `random_data` scores and `projection` scores

The Sillhouette Score value is a very good indicator of the stability of our cluster, but lets split our DataFrame in 5 and to each of this sets, let's calculate their own Sillhouette Scores.

In [31]:
```
set1, set2, set3, set4, set5 = np.array_split(projection, 5)
s1 = clustering_validation(5,set1)
s2 = clustering_validation(5,set2)
s3 = clustering_validation(5,set3)
s4 = clustering_validation(5,set4)
```

```
s5 = clustering_validation(5,set5)
print(s1, s2, s3, s4, s5)
```

```
0.44731205557467246 0.49515613652932916 0.5019799234722745 0.4539963586239956 0.4358936075354957
```

As shown above, ou clusters are stable and there's no significant difference between their Sillhoute Scores.

Further along the project, we will group the clusters with `.groupby()` and sum their `energy` levels in order to have visual and numerical understanding of the clustering performed by the K-Means technique

Now that we have the number of clusters, our pipelined DataFrame `projection`, the `kmeans_pca_pipeline` will instatiate `KMeans` which will fit our data to the model. Once, the `random_state` parameter is defined as `SEED`.

In [32]:
```
kmeans_pca_pipeline = KMeans(n_clusters=5, verbose=False, random_state=SEED)
kmeans_pca_pipeline.fit(projection)
```

Out[32]:
```
KMeans(n_clusters=5, random_state=1234, verbose=False)
```

The column `cluster_pca` will be added to `df` and `projection` with the outcome of `kmeans_pca_pipeline.predict(projection)`

In [33]:
```
df['cluster_pca'] = kmeans_pca_pipeline.predict(projection)
projection['cluster_pca'] = kmeans_pca_pipeline.predict(projection)
```

The columns `artist`, `name` and `energy` will be "imported" from `df` to `projectio`, in order to build our recommendation system

In [34]:
```
projection['artist'] = df['artist']
projection['name'] = df['name']
projection['energy'] = df['energy']
projection.head(10)
```

Out[34]:

	0	1	2	3	4	cluster_pca	artist	name	energy
0	-2.391573	-1.365183	2.510296	-0.259731	0.345778	3	Prince	1999	0.7300
1	-0.834016	-0.789067	-1.695007	-0.448084	-0.027110	1	Blonde Redhead	23	0.8320
2	1.319371	-1.406747	-1.614033	-1.156800	0.557151	1	Damien Rice	9 Crimes	0.1390
3	-1.481015	-0.196788	0.733334	0.574310	-2.701284	3	Nena	99 Luftballons	0.4380
4	-2.743797	1.896716	-0.969154	-0.973701	0.061559	4	Underoath	A Boy Brushed Red Living In Black And White	0.9320
5	3.468711	0.755210	0.954910	-1.168354	-0.006014	2	Emmanuelle Rimbaud	A Burden to Bear	0.0475
6	2.529262	0.582623	1.089858	0.037040	-0.715914	2	Ron Adelaar	A La Plage	0.0584
7	-1.862514	-0.554799	1.356120	1.155381	-1.955080	3	Elvis Presley	A Little Less Conversation - JXL Radio Edit Remix	0.9350
8	-3.629038	2.643469	-0.246900	0.708351	3.006919	4	Linkin Park	A Place for My Head	0.9080
9	-2.870962	2.802562	-2.006053	1.305558	-0.668305	4	Thirty Seconds To Mars	ATTACK	0.8760

As mentioned above, here is the sum of the values of `energy` for each cluster contained in our clustering `projection['cluster_pca']`

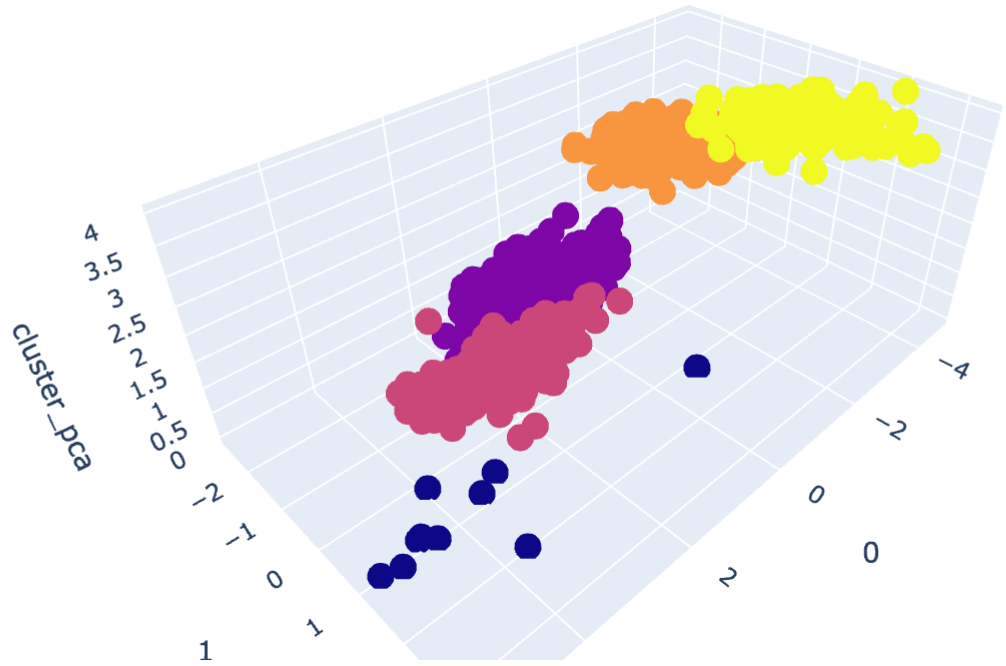
In [35]:
```
projection.groupby("cluster_pca")['energy'].sum().to_frame()
```

Out[35]:

cluster_pca	energy
0	2.31378
1	75.41900
2	29.68494
3	96.33200
4	144.52800

Below, the 3D visualization of the clusters.

In [36]:
```
fig = px.scatter_3d(projection, x=0, y=1, z = 'cluster_pca', color='cluster_pca')
fig.show()
```



The explained variance ratio is the percentage of variance that is attributed by each of the selected components. Below, its values:

In [37]:
```
variance_ratio = pca_pipeline[1].explained_variance_ratio_
for i in variance_ratio:
    print(i)
```

```
0.3603097640251467
0.1139124224840144
0.10838089626548392
0.06980151875734583
0.06501194015679124
```

Now lets sum them, multiply by `100` and print the Explained Variance Ratio

In [38]:
```
variance_ratio_sum = pca_pipeline[1].explained_variance_ratio_.sum()
print(f"Explained Variance Ratio: {variance_ratio_sum*100:.2f}%")
```

```
Explained Variance Ratio: 71.74%
```

And the number of explained components

In [39]:
```
components = pca_pipeline[1].explained_variance_.sum()
print(f"Explained Components: {int(components)}")
```

```
Explained Components: 10
```

What `projection` looks like now

In [40]:
```
projection
```

Out[40]:

	0	1	2	3	4	cluster_pca	artist	name	energy
0	-2.391573	-1.365183	2.510296	-0.259731	0.345778	3	Prince	1999	0.7300
1	-0.834016	-0.789067	-1.695007	-0.448084	-0.027110	1	Blonde Redhead	23	0.8320
2	1.319371	-1.406747	-1.614033	-1.156800	0.557151	1	Damien Rice	9 Crimes	0.1390
3	-1.481015	-0.196788	0.733334	0.574310	-2.701284	3	Nena	99 Luftballons	0.4380
4	-2.743797	1.896716	-0.969154	-0.973701	0.061559	4	Underoath	A Boy Brushed Red Living In Black And White	0.9320
...	...	...	...	...	...	...	...	...	...
681	3.055752	1.109257	0.404262	1.035114	-0.051757	2	Leo Nocta	windcatcher	0.2360
682	0.573026	-1.675048	-1.760564	0.542327	-0.412348	1	Soccer Mommy	yellow is the color of her eyes	0.5150
683	0.181140	-1.786693	-0.832856	-1.463000	1.775066	1	Tate McRae	you broke me first	0.3740
684	1.076963	-2.033680	-0.762161	-1.066804	1.693244	1	Jeremy Zucker	you were good to me	0.0848
685	3.631699	1.023182	0.537810	-0.630100	-0.443122	2	praam	æfre	0.0156

686 rows × 9 columns

Euclidean Distances

In machine learning, Euclidean distance is a common method used to measure the similarity between two data points. When calculating Euclidean distance between two points in a multi-dimensional space, we need to compare the values of each dimension or feature between the two points.

We will use the method to build the recommendation system

In [41]:
```
from sklearn.metrics import euclidean_distances
```

We are looking for energtic song the cluster with the highest level of energy is number `4`. Let's now check the songs contained in that cluster

In [42]:
```
projection[projection['cluster_pca']==4]['name'].unique()
```

Out[42]:
```
array(['A Boy Brushed Red Living In Black And White',
       'A Place for My Head', 'ATTACK', 'Adagio For Strings',
       'Afraid of Heights', 'Afterlife', 'Alive',
       'All These Things I Hate (Revolve Around Me)',
       'All in the Suit That You Wear', 'Always', 'Angry Chair',
       'Anthem of Our Dying Day', 'As I Sleep (feat. Charlee)',
       'Away From Me', 'Back Where I Belong (feat. Avicii)',
       'Bat Country', 'Beautiful Night', 'Billinghurst', 'Blackout',
       'Blame', 'Blaster', 'Bleed American', 'Body Rock', 'Boiler',
       'Bottom of a Bottle', 'Break Stuff', 'Car Underwater',
       'Children Of Today', 'Chop Suey!', 'Click Click Boom',
       'Cold as Stone (feat. Charlotte Lawrence) - Lipless Remix',
       'Come Out And Play (Keep ‘Em Separated)', 'Coming Undone',
       'Count That - R3HAB Edit', 'Crawling In The Dark',
       'Crystal Baller - 2006 Remaster', 'Dancing Through Sunday',
       'Dani California', 'Deep Dark Jungle', 'Devotion And Desire',
       'Do Bad Well (feat. Nevve)', 'Do It Again',
       "Don't Stop Me Now - 2011 Mix", 'Dragula', 'Duality',
       'Duck And Run', 'EDM Sux',
       'Electronic Memories (feat. Mickey Kojak)', 'Enemy', 'Every Way',
       'Faces', 'Falling Away from Me', 'Fantasy - R3HAB Remix',
       'Feeling This', 'Flex',
       'Flying by Candlelight (ABGT300) - Above & Beyond Club Mix',
       'From Yesterday', 'Gasoline', 'Get Up Offa That Thing',
       'Getting Away With Murder', 'Go With The Flow',
       'Gone My Way - KhoMha Remix', 'Groovin', 'Guernica',
       'Halfway There (feat. Lena Leon)', 'Handful Of Redemption',
       'Hash Pipe', 'Heavy', 'Holding On', 'Hollowman', 'Holy Water',
       'Home', "Honey, This Mirror Isn't Big Enough for the Two of Us",
       'I Stand Alone', "I'm Not Okay (I Promise)", 'ID8', 'Indigo',
       'Inside Of You', 'Inside the Fire',
       "It's Goin' Down (feat. Mike Shinoda & Mr. Hahn)",
       'Killing In The Name', 'Kiss You All Over - Single Edit',
       "Let's Go Crazy", 'Lift Me Up (feat. Carla Monroe)',
       'Lights Go Down', 'Like The First Time - House Mix', 'Like We Do',
       'Listening', 'Little Things', 'Machinehead', 'Make Believe',
       'Midnight Sun', 'Mudshovel', 'My Favourite Game', 'My Hero',
       'My Own Worst Enemy', 'My Way - Marcus Santoro Remix',
       'Never Be Lonely (feat. Envy Monroe)', 'No More 54',
       'No One Knows', 'Nowhere Kids', 'One More Night',
       'One Night - D.O.D Remix', 'One Step Closer',
       'Our Lady of Sorrows', 'Out Of Control',
       'Out Of Control - Pat Vocal Mix', 'Over My Head (Better Off Dead)',
       'Papercut', 'Paperthin Hymn', 'Paralyzer',
```

```
'People Are Strange - Festival Mix', 'Poem', 'Points of Authority',
'Polyamorous', 'Prayer Of The Refugee', 'Pressure', 'Promise',
'Reinventing Your Exit', 'Renegades Of Funk', 'Rescue Me',
'Ring The Alarm - Stadiumx Remix',
'Rise (ft. Matluck) - Thomas Gold Remix', 'Rusted From the Rain',
'Sahara Love - Seven Lions Remix', 'Same Direction', 'Savior',
'Say Days Ago', 'Secrets', 'Shimmer', 'Shine', 'Smooth Criminal',
'South Beach', 'Stay Here', 'Still Alive', 'Story Of A Violin',
'Stricken', 'Stupify', 'Supersonic (feat. Christian Burns)',
'Swerve City', "Tears Don't Fall", 'Thank You for the Venom',
'The Hardest Button To Button', 'The Hell Song',
'The King (with Dzeko)', 'The Wall', 'There For You', 'Toxic',
'Tubthumping', 'Under A Killing Moon', 'Up Against (Blackout)',
'Uprising', 'Vamos', 'Voices',
'Waiting For Tomorrow (feat. Mike Shinoda)', 'Wash It All Away',
'Welcome Home', 'Whiskey In The Jar', 'White Lies',
'White Lies - Guy Arthur Remix', 'Wild Wild Son - Club Mix',
'Without', 'Wrong Way', "You're Gonna Go Far, Kid"], dtype=object)
```

So, working out to the sound of Rage Against The Machine is always a good call. Let's assing the song `'Renegades Of Funk'` to the variable `song`

In [43]:
```
song = 'Renegades Of Funk'
```

In [44]:
```
projection[projection['name']==song]
```

Out[44]:

	0	1	2	3	4	cluster_pca	artist	name	energy
446	-3.224087	0.998701	0.111385	-0.205318	0.94058	4	Rage Against The Machine	Renegades Of Funk	0.908

We will build the dataframe `recommendations` with the names of the songs and contained in `projection['cluster_pca']==4` and their respective values in columns `0` and `1`.

In [45]:
```
recommendations = projection[projection['cluster_pca']==4][[0,1,'name']]
```

In [46]:
```
recommendations
```

Out[46]:

	0	1	name
4	-2.743797	1.896716	A Boy Brushed Red Living In Black And White
8	-3.629038	2.643469	A Place for My Head
9	-2.870962	2.802562	ATTACK
10	-1.222704	2.052526	Adagio For Strings
13	-3.341989	1.482998	Afraid of Heights
...	...	...	...
639	-2.789877	1.175616	White Lies - Guy Arthur Remix
644	-1.940914	1.345136	Wild Wild Son - Club Mix
650	-2.151457	0.840846	Without
658	-2.198107	-0.004079	Wrong Way
664	-2.547729	1.246958	You're Gonna Go Far, Kid

164 rows × 3 columns

The variables `x_song` and `y_song` will hold the values of `0` and `1` for the song we chose in order to calculate the euclidean distances between it and the other songs

In [47]:
```
x_song = list(projection[projection['name']==song][0])[0]
y_song = list(projection[projection['name']==song][1])[0]
```

In [48]:
```
print(x_song, y_song)
```

```
-3.2240866227337612 0.998701176958259
```

The distance between the song we chose and the others in the cluster will be assigned to the variable `distances` which will be futher added to `recommendations`, along with `df['id']`.

`recommendations_sorted` will be sort the values by distance.

In [49]:
```
distances = euclidean_distances(recommendations[[0,1]], [[x_song, y_song]])
#distances
```

In [50]:
```
recommendations['id'] =  df['id']
recommendations['distances'] = distances
recommendations_sorted = recommendations.sort_values('distances').head(30)
recommendations_sorted
```

Out[50]:

	0	1	name	id	distances
446	-3.224087	0.998701	Renegades Of Funk	5YBVDvTSSSiqv7KZDeUlXA	0.000000
309	-3.249514	1.117968	Like The First Time - House Mix	5ABnK6bOqbsIRVsrc1UBOt	0.121947
362	-3.050675	1.226869	My Own Worst Enemy	33iv3wnGMrrDugd7GBso1z	0.286587
489	-2.994669	0.737384	Smooth Criminal	2T3blUc9Jnui2a0ZhuNOag	0.347734
129	-2.915992	0.829588	Dani California	10Nmj3JCNoMeBQ87uw5j8k	0.351457
126	-3.039267	1.340155	Dancing Through Sunday	4O5GKVbnQ8U9BxYWu0hlug	0.388264
172	-3.617261	0.977242	Every Way	4xxyCh7xkK30LVBO81poif	0.393760
164	-2.766716	1.074843	Electronic Memories (feat. Mickey Kojak)	2viJufnGlsjR1X51CEHpeg	0.463665
587	-2.807817	1.210457	Tubthumping	6dL46HEj2hzaSach8EFyCi	0.467034
639	-2.789877	1.175616	White Lies - Guy Arthur Remix	1FoMWISq2KNqKappkrG5Yo	0.468868
191	-2.871466	1.319940	Flex	43p1llKF0PBgwkqzS2EdCk	0.477007
206	-2.945186	1.393519	Gasoline	4e7yuouxvWPbK1q2kVlgz8	0.483390
146	-3.703493	1.086199	Don't Stop Me Now - 2011 Mix	7hQJA50XrCWABAu5v6QZ4i	0.487325
13	-3.341989	1.482998	Afraid of Heights	0P44AUPRzaIAHNWquN0fGR	0.498442
611	-2.761503	1.212269	Voices	47H4oc7Zkihwae7ST7F4zp	0.509505
110	-2.707614	1.106659	Count That - R3HAB Edit	1J1wnDwBjkFvDVOxNOLVjY	0.527636
268	-2.833946	1.372676	Inside Of You	6okaxit4V1bU6yIVVsP836	0.540432
404	-2.937617	1.470351	Over My Head (Better Off Dead)	3SO0vfryYv381w1ImgWONG	0.551832
143	-2.650562	1.004015	Do It Again	5yzzy7Hevww3RDntQM5mG1	0.573549
400	-2.687435	0.783430	Out Of Control - Pat Vocal Mix	6wCsrMyI3DavGvodBx8A7h	0.578219
49	-2.769430	1.372771	Bat Country	6irNz2hSXvd2QJG0vXyBFR	0.588762
411	-2.802922	0.578377	Paralyzer	6k1njNBYUIQgWJLG4ed26y	0.595022
113	-2.607342	0.920336	Crawling In The Dark	1eK4KsyjAzr2xJ86Tj5SWa	0.621703
305	-2.616891	1.173081	Lift Me Up (feat. Carla Monroe)	0vK80mtYypq4dAwQelZ2Dh	0.631739
599	-2.620119	1.196179	Up Against (Blackout)	7dx3zyB8aQHd2LQm0u1lwS	0.635432

	0	1	name	id	distances
42	-2.819961	1.548055	Back Where I Belong (feat. Avicii)	78W8wiUIlQ2SnWY9TVowKZ	0.681988
260	-2.650529	0.625078	ID8	2jxfHzOYVLmGKejNRHvqoc	0.684516
142	-2.727330	1.483088	Do Bad Well (feat. Nevve)	7zHrHnVSQwI95FGHewXDI8	0.693828
150	-2.553212	0.765712	Dragula	6Nm8h73ycDG2saCnZV8poF	0.710181
23	-2.724320	1.505448	All in the Suit That You Wear	17Nk1aji6gd26vabky7fMi	0.711729

The libray `spotipy` will connect us to Spotify using `client_id` and `client_secret`.

The credentials are created with `SpotifyClientCredentials` and `sp` receive these parameters in order to extract information such as album covers, names of the songs, etc.

In [51]:
```
import spotipy
from spotipy.oauth2 import SpotifyOAuth
from spotipy.oauth2 import SpotifyClientCredentials
```

In [52]:
```
scope = 'user-library-read playlist-modify-private'
client_id = '23e8bc89c69342b0a45315c4ecc57919'
client_secret = '5461de2cb2ac42adb55f2fac095cc718'
OAuth = SpotifyOAuth(
        scope = scope,
    redirect_uri = 'http://localhost:5000/callback',
    client_id = client_id,
    client_secret = client_secret
        )
client_credentials_manager = SpotifyClientCredentials(client_id = client_id, client_secret = client_secret)
sp = spotipy.Spotify(client_credentials_manager = client_credentials_manager)
```

Song ID is responsible for identifying the song on Spotify database and return its information. Both `df` and `recommendations_sorted` have this information. We will use in order to get the ID for `df`

In [53]:
```
song_id = df[df['name']==song]['id'].iloc[0]
print(song_id)
```

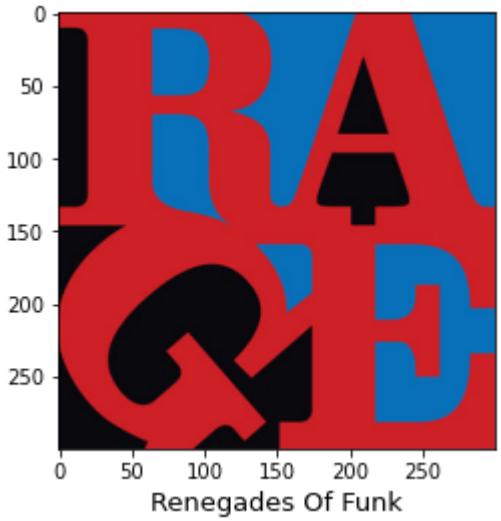
```
5YBVDvTSSSiqv7KZDeUlXA
```

The variable `track` will receive `sp.track()` with `song_id` as parameter and `url` and `name` the link for the album/single cover and the song's name, respectively

In [54]:
```
track = sp.track(song_id)
#track.keys()
url = track["album"]["images"][1]["url"]
name = track["name"]
```

Below, the album cover, displayed using `matplotlib.pyplot`

In [55]:
```
image = io.imread(url)
plt.imshow(image)
plt.xlabel(name, fontsize=13)
plt.show()
```



Now, we need to create a list containing the names and urls of the songs in `recommendations_sorted`, in order to display visually what's been recommended to us.

The function `recommendations` will loop through `recommendations_sorted` and return two lists containing names and urls, which will laber be assigned to the variables `name` and `url` respectively

In [56]:
```
def recommendations(playlist_id):
    url = []
```

```
    name = []
    for i in playlist_id:
        track = sp.track(i)
        url.append(track["album"]["images"][1]["url"])
        name.append(track["name"])
    return name, url
```

In [57]:
```
name, url = recommendations(recommendations_sorted['id'])
```

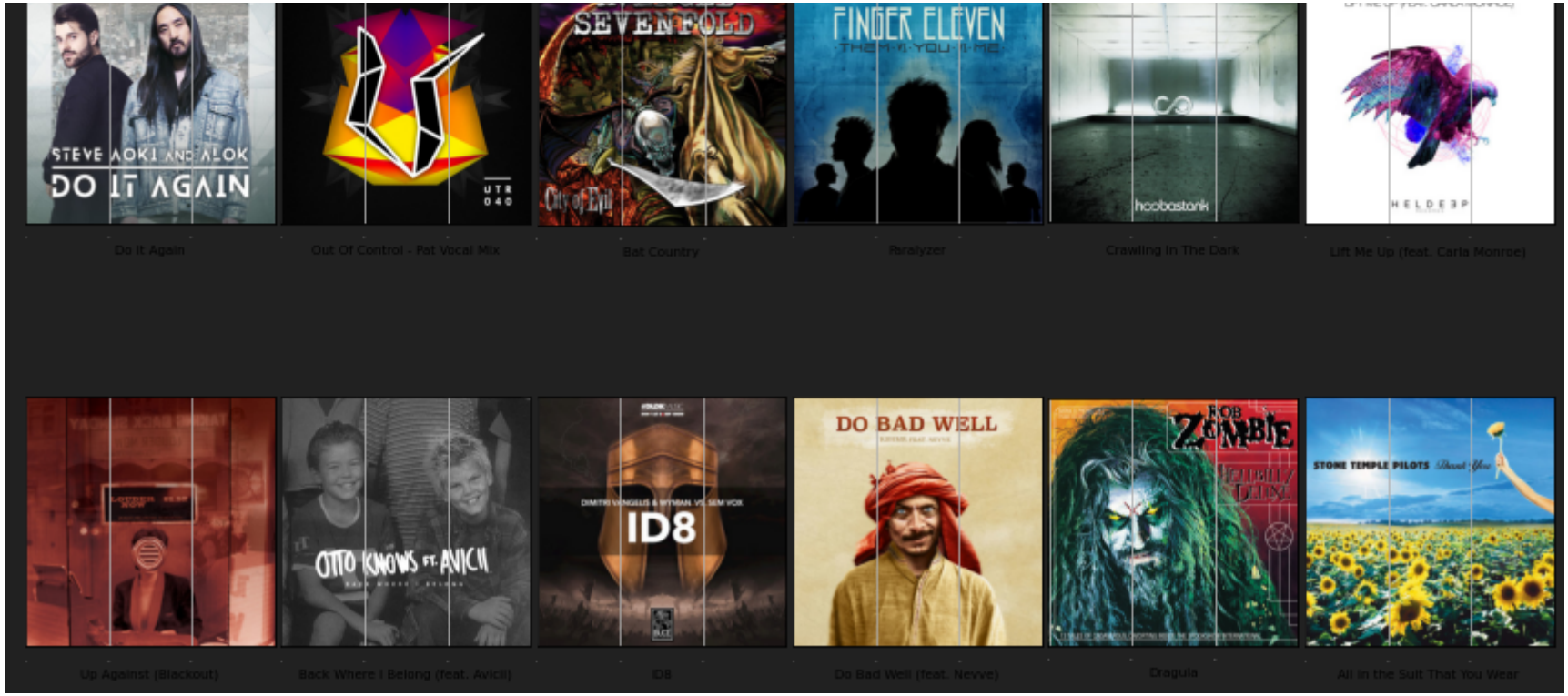
The function `album_cover` will receive these two variables as parameter and using using `matplotlib.pypolt` will return an image with the album covers for the 30 songs contained in `recommendations_sorted`

In [58]:
```
def album_cover(name, url):
    plt.figure(figsize=(12,20))
    columns = 6

    for i, u in enumerate(url):
        ax = plt.subplot(len(url) // columns + 1, columns, i + 1)
        image = io.imread(u)
        plt.imshow(image)
        ax.get_yaxis().set_visible(False)
        plt.xticks(color = 'w', fontsize = 0.1)
        plt.yticks(color = 'w', fontsize = 0.1)
        plt.xlabel(name[i], fontsize = 7)
        plt.tight_layout(h_pad = 0.5, w_pad = 0)
        plt.subplots_adjust(wspace = None, hspace = None)
        plt.tick_params(bottom = False)
        plt.grid(visible = None)
    plt.show()
```

In [59]:
```
album_cover(name,url)
```


RAGE
Renegades Of Funk
Like The First Time - House Mix
Lit
My Own Worst Enemy
ALIEN ANT FARM
Smooth Criminal
RED HOT CHILI PEPPERS
STADIUM ARCADIUM
Dani California
afi
Dancing Through Sunday
ANTONIO GIACCA
Every Way
F*ND PARAD*SE
Electronic Memories (feat. Mickey Kojak)
CHUMBAWAMBA
Tubthumping
White Lies - Guy Arthur Remix
FEDDE LE GRAND
AND FUNK MACHINE FEAT. GEMEDAL LEVY
FLEX
Rex
SEETHER
DISCLAIMER II
Gasoline
QUEEN QUEEN QUEEN QUEEN QUEEN
Don't Stop Me Now - Remastered 2011
BILLY TALENT
AFRAID OF HEIGHTS
Afraid of Heights
DISTURBED
Voices
STEFF DA CAMPO & SMACK
COUNT THAT (FT. KYDSHO)
R3HAB EDIT
Count That - R3HAB Edit
Inside Of You
SUM 41
DOES THIS
LOOK INFECTED?
Over My Head (Better Off Dead)



The function below (`song_choice`)summarizes all the process. It receives the song name as a parameter and returns a list 30 songs based on the song you chose

For the purposes of demonstration all the songs in cluster 4 will be assigned to `random_choice` which will randomly choose one so it can be passed though `song_choice` and test the function

In [60]:
```
from random import choice
```

In [61]:
```
random_choice = choice(projection[projection['cluster_pca']==4]['name'].unique())
print(random_choice)
```

```
Poem
```

In [62]:
```
def song_choice(song_name):
    recommendation = projection[projection['cluster_pca']==4][[0,1,'name']]
    song = projection[projection['name']==song_name]
    x_song = list(projection[projection['name']==song_name][0])[0]
    y_song = list(projection[projection['name']==song_name][1])[0]
```

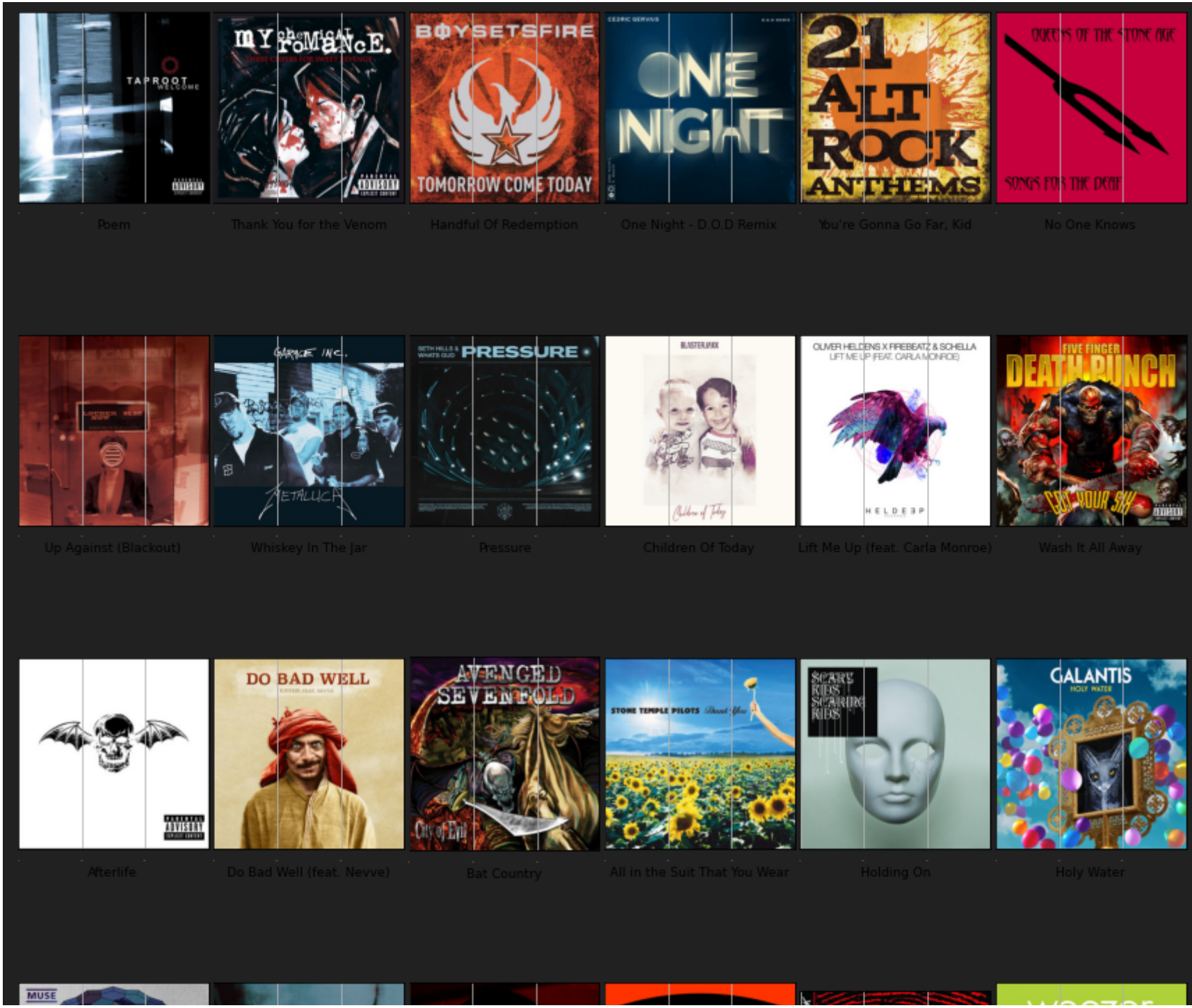
```
    distances = euclidean_distances(recommendation[[0,1]],[[x_song, y_song]])
    recommendation['id'] = df['id']
    recommendation['distances'] = distances
    recommendation_sorted = recommendation.sort_values('distances').head(30)
    #return recommendation_sorted

    playlist_id = recommendation_sorted['id']

    url = []
    name = []
    for i in playlist_id:
        track = sp.track(i)
        url.append(track["album"]["images"][1]["url"])
        name.append(track["name"])

    plt.figure(figsize = (12,20))
    columns = 6
    for i, u in enumerate(url):
        ax = plt.subplot(len(url) // columns + 1, columns, i + 1)
        image = io.imread(u)
        plt.imshow(image)
        ax.get_yaxis().set_visible(False)
        plt.xticks(color = 'w', fontsize = 0.1)
        plt.yticks(color = 'w', fontsize = 0.1)
        plt.xlabel(name[i], fontsize = 9)
        plt.tight_layout(h_pad = 0.3, w_pad =0)
        plt.subplots_adjust(wspace = None, hspace = None)
        plt.grid(visible = None)
        plt.tick_params(bottom = False)
    plt.show()
```

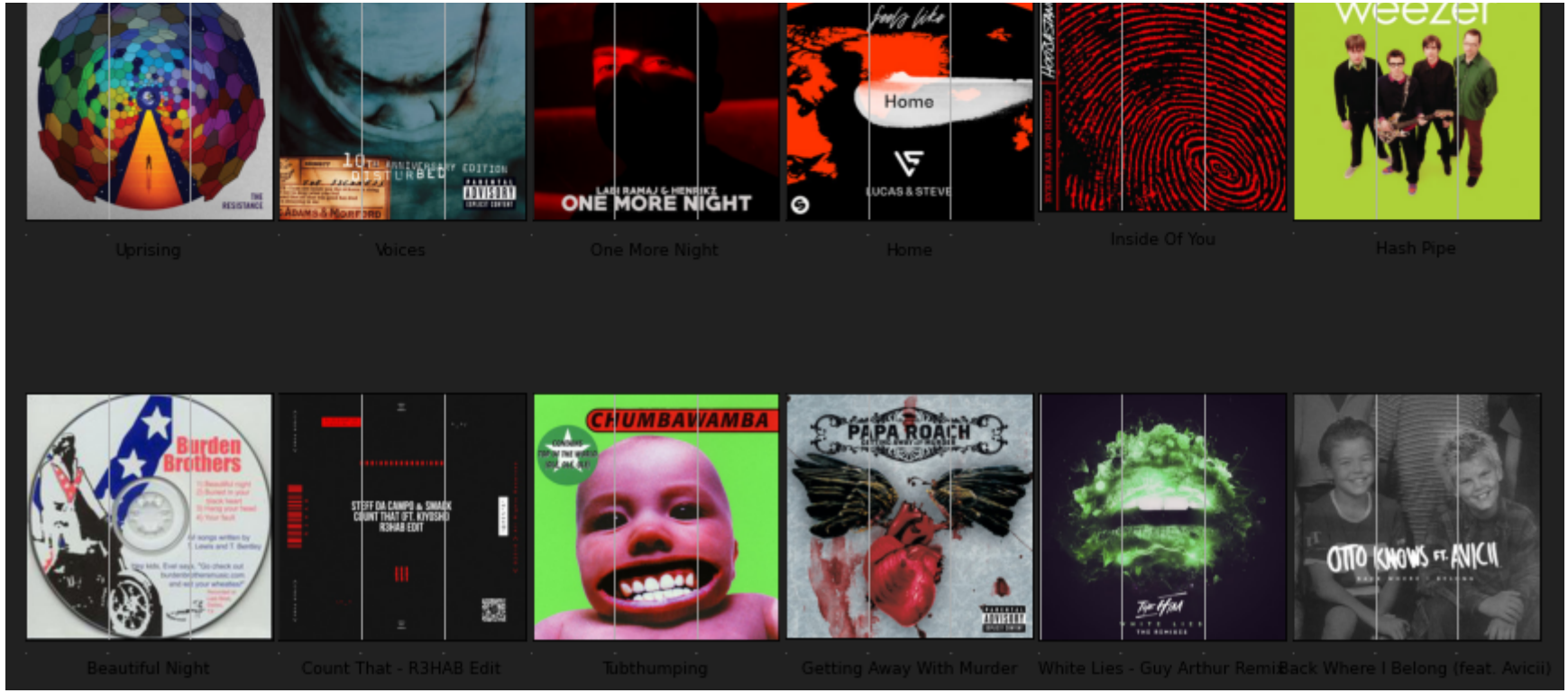
In [63]:
```
song_choice(random_choice)
```



Poem | Thank You for the Venom | Handful Of Redemption | One Night - D.O.D Remix | You're Gonna Go Far, Kid | No One Knows

Up Against (Blackout) | Whiskey In The Jar | Pressure | Children Of Today | Lift Me Up (feat. Carla Monroe) | Wash It All Away

Afterlife | Do Bad Well (feat. Nevve) | Bat Country | All in the Suit That You Wear | Holding On | Holy Water


Uprising
Voices
One More Night
Home
Inside Of You
Hash Pipe
Beautiful Night
Count That - R3HAB Edit
Tubthumping
Getting Away With Murder
White Lies - Guy Arthur Remi
Back Where I Belong (feat. Avicii)

In [ ]: