```python
import numpy as np
import pandas as pd

import seaborn as sns
import plotly.express as px
import plotly.graph_objs as go
import matplotlib.pyplot as plt

from collections import defaultdict
from scipy.spatial.distance import cdist
from sklearn.preprocessing import MinMaxScaler, StandardScaler

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

```python
data = pd.read_csv("data.csv")
genre_data = pd.read_csv('data_by_genres.csv')
year_data = pd.read_csv('data_by_year.csv')
artist_data = pd.read_csv('data_by_artist.csv')
```

```python
data.sample(5)
```

	valence	year	acousticness	artists	danceability	duration_ms	energy	explicit	id	instrumentalness	key	liveness	loudness	mode	name	pop
147374	0.167	1978	0.506	['Little River Band']	0.511	252947	0.323	0	5ZpRf623O7MaeGHIwNUcZq	0.000114	5	0.3200	-10.028	1	Sanity's Side - 2010 Digital Remaster	
59927	0.338	1944	0.963	['Richard Strauss', 'Franz Szkokan', 'Lisa del...	0.374	753333	0.316	0	6nhH8vIMiyShYsl9RiNbAS	0.000000	10	0.8780	-17.502	0	Arabella, Op.79, TrV 263 / Act 2: Ein Feigling...	
52588	0.296	1996	0.107	['Cake']	0.436	201613	0.579	0	1prNlxt0CJBXEC8Xp0wxYd	0.000730	9	0.0996	-9.795	0	Friend Is a Four Letter Word	
157928	0.744	1956	0.872	['J.J. Johnson', 'Kai Winding']	0.648	175227	0.481	0	1PsRtxEPYMyilBStXWCest	0.871000	0	0.1030	-10.219	1	The Continental (You Kiss While You're Dancing)	
132557	0.634	1978	0.542	['Emmylou Harris']	0.475	189240	0.386	0	5VEUwTxuC4AVsYcYCDMUlk	0.000000	3	0.0783	-11.423	1	Easy from Now On - 2003 Remaster	

```python
genre_data.sample(5)
```

	mode	genres	acousticness	danceability	duration_ms	energy	instrumentalness	liveness	loudness	speechiness	tempo	valence	popularity	key
378	1	british modern classical	0.876342	0.288750	234667.338217	0.184769	0.417125	0.160951	-21.579563	0.051742	106.913516	0.221813	13.468956	2
847	1	delta blues	0.771977	0.579439	200740.886489	0.368764	0.113124	0.179875	-12.740045	0.076983	117.211747	0.599151	24.429914	10
2044	1	oakland hip hop	0.124070	0.729840	236709.893615	0.628087	0.002693	0.164853	-7.680831	0.221095	107.575201	0.540111	47.764114	10
1139	0	gaming edm	0.089090	0.563641	245632.186650	0.810934	0.263486	0.200551	-5.311220	0.072846	132.444843	0.326399	58.918580	7
831	1	deep r&b	0.155000	0.426000	182707.000000	0.953000	0.000000	0.132000	-3.282000	0.106000	169.930000	0.678000	51.000000	9

```python
year_data.sample(5)
```

	mode	year	acousticness	danceability	duration_ms	energy	instrumentalness	liveness	loudness	speechiness	tempo	valence	popularity	key
74	1	1995	0.302512	0.559046	246624.091500	0.578580	0.123269	0.201374	-10.119450	0.083754	119.281978	0.530247	44.801000	0
14	1	1935	0.778386	0.555869	220124.247036	0.246367	0.225873	0.229300	-15.414750	0.353912	110.667957	0.545578	1.501318	7
13	1	1934	0.891149	0.528706	189356.126298	0.262131	0.276382	0.213453	-14.756875	0.102457	115.390846	0.558805	1.257785	0
31	1	1952	0.874301	0.457032	229434.727000	0.253626	0.288413	0.232972	-15.958866	0.137362	108.232280	0.443135	3.338000	0
12	1	1933	0.899898	0.570290	196219.257598	0.279899	0.183949	0.209072	-13.069009	0.091123	112.522000	0.599410	6.898698	7

```python
artist_data.sample(5)
```

	mode	count	acousticness	artists	danceability	duration_ms	energy	instrumentalness	liveness	loudness	speechiness	tempo	valence	popularity	key
21922	0	2	0.170000	Sandra De Sá	0.842000	267267.000000	0.54900	0.001220	0.080400	-10.857000	0.044700	96.56600	0.967000	51.000000	5
7720	1	39	0.949974	Esa-Pekka Salonen	0.274154	196321.179487	0.17727	0.834154	0.233482	-23.138128	0.048528	99.85359	0.133895	7.307692	0
5805	0	2	0.953000	Dave Thomas Junior	0.365000	330484.000000	0.15800	0.039200	0.096100	-14.019000	0.035200	123.53100	0.074000	58.000000	3
243	1	12	0.000521	ASIAN KUNG-FU GENERATION	0.417000	244089.000000	0.91850	0.092434	0.136000	-4.071500	0.065650	156.35350	0.489167	57.000000	4
14215	1	2	0.507000	Lauren O'Connell	0.415000	185263.000000	0.34300	0.818000	0.349000	-13.526000	0.032100	170.90400	0.485000	53.000000	7

```python
datasets = [("data", data), ("genre_data", genre_data), ("year_data", year_data), ("artist_data", artist_data)]
```

```python
data['year'] = pd.to_datetime(data['year'], format='%Y')
data['release_date'] = pd.to_datetime(data['release_date'])
year_data['year'] = pd.to_datetime(year_data['year'], format='%Y')
```

Checking the Data

```python
for name, df in datasets:
    print(f"Info about the dataset: {name}")
    print("-"*30)
    print(df.info())
    print()
```

```
Info about the dataset: data
------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 170653 entries, 0 to 170652
Data columns (total 19 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   valence           170653 non-null  float64
 1   year              170653 non-null  datetime64[ns]
 2   acousticness      170653 non-null  float64
 3   artists           170653 non-null  object
 4   danceability      170653 non-null  float64
 5   duration_ms       170653 non-null  int64
 6   energy            170653 non-null  float64
 7   explicit          170653 non-null  int64
 8   id                170653 non-null  object
 9   instrumentalness  170653 non-null  float64
 10  key               170653 non-null  int64
 11  liveness          170653 non-null  float64
 12  loudness          170653 non-null  float64
 13  mode              170653 non-null  int64
 14  name              170653 non-null  object
 15  popularity        170653 non-null  int64
 16  release_date      170653 non-null  datetime64[ns]
 17  speechiness       170653 non-null  float64
 18  tempo             170653 non-null  float64
dtypes: datetime64[ns](2), float64(9), int64(5), object(3)
memory usage: 24.7+ MB
None

Info about the dataset: genre_data
------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2973 entries, 0 to 2972
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   mode              2973 non-null   int64
 1   genres            2973 non-null   object
 2   acousticness      2973 non-null   float64
 3   danceability      2973 non-null   float64
 4   duration_ms       2973 non-null   float64
 5   energy            2973 non-null   float64
 6   instrumentalness  2973 non-null   float64
 7   liveness          2973 non-null   float64
 8   loudness          2973 non-null   float64
 9   speechiness       2973 non-null   float64
 10  tempo             2973 non-null   float64
 11  valence           2973 non-null   float64
 12  popularity        2973 non-null   float64
 13  key               2973 non-null   int64
dtypes: float64(11), int64(2), object(1)
memory usage: 325.3+ KB
None

Info about the dataset: year_data
------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   mode              100 non-null    int64
 1   year              100 non-null    datetime64[ns]
 2   acousticness      100 non-null    float64
 3   danceability      100 non-null    float64
 4   duration_ms       100 non-null    float64
 5   energy            100 non-null    float64
 6   instrumentalness  100 non-null    float64
 7   liveness          100 non-null    float64
 8   loudness          100 non-null    float64
 9   speechiness       100 non-null    float64
 10  tempo             100 non-null    float64
 11  valence           100 non-null    float64
 12  popularity        100 non-null    float64
 13  key               100 non-null    int64
dtypes: datetime64[ns](1), float64(11), int64(2)
memory usage: 11.1 KB
None

Info about the dataset: artist_data
------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 28680 entries, 0 to 28679
Data columns (total 15 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   mode              28680 non-null  int64
 1   count             28680 non-null  int64
 2   acousticness      28680 non-null  float64
 3   artists           28680 non-null  object
 4   danceability      28680 non-null  float64
 5   duration_ms       28680 non-null  float64
 6   energy            28680 non-null  float64
 7   instrumentalness  28680 non-null  float64
 8   liveness          28680 non-null  float64
 9   loudness          28680 non-null  float64
 10  speechiness       28680 non-null  float64
 11  tempo             28680 non-null  float64
 12  valence           28680 non-null  float64
 13  popularity        28680 non-null  float64
 14  key               28680 non-null  int64
dtypes: float64(11), int64(3), object(1)
memory usage: 3.3+ MB
None
```

```python
for name, df in datasets:
    print(f"Missing Values in: {name}")
    print("-"*30)
    print(df.isnull().sum())
    print()
```

```
Missing Values in: data
------------------------------
valence             0
year                0
acousticness        0
artists             0
danceability        0
duration_ms         0
energy              0
explicit            0
id                  0
instrumentalness    0
key                 0
liveness            0
loudness            0
mode                0
name                0
popularity          0
release_date        0
speechiness         0
tempo               0
dtype: int64

Missing Values in: genre_data
------------------------------
mode                0
genres              0
acousticness        0
danceability        0
duration_ms         0
energy              0
instrumentalness    0
liveness            0
loudness            0
speechiness         0
tempo               0
valence             0
popularity          0
key                 0
dtype: int64

Missing Values in: year_data
------------------------------
mode                0
year                0
acousticness        0
danceability        0
duration_ms         0
energy              0
instrumentalness    0
liveness            0
loudness            0
speechiness         0
tempo               0
valence             0
popularity          0
key                 0
dtype: int64

Missing Values in: artist_data
------------------------------
mode                0
count               0
acousticness        0
artists             0
danceability        0
duration_ms         0
energy              0
instrumentalness    0
liveness            0
loudness            0
speechiness         0
tempo               0
valence             0
popularity          0
key                 0
dtype: int64
```

```python
for name, df in datasets:
    print(f"Duplicates in the dataset: {name}")
    print("-"*30)
    print(df.duplicated(keep=False).sum())
    print()
```

```
Duplicates in the dataset: data
------------------------------
0

Duplicates in the dataset: genre_data
------------------------------
0

Duplicates in the dataset: year_data
------------------------------
0

Duplicates in the dataset: artist_data
------------------------------
0
```

```python
for name, df in datasets:
    print(f"Unique Values in: {name}")
    print("-"*30)
    print(df.nunique())
    print()
```

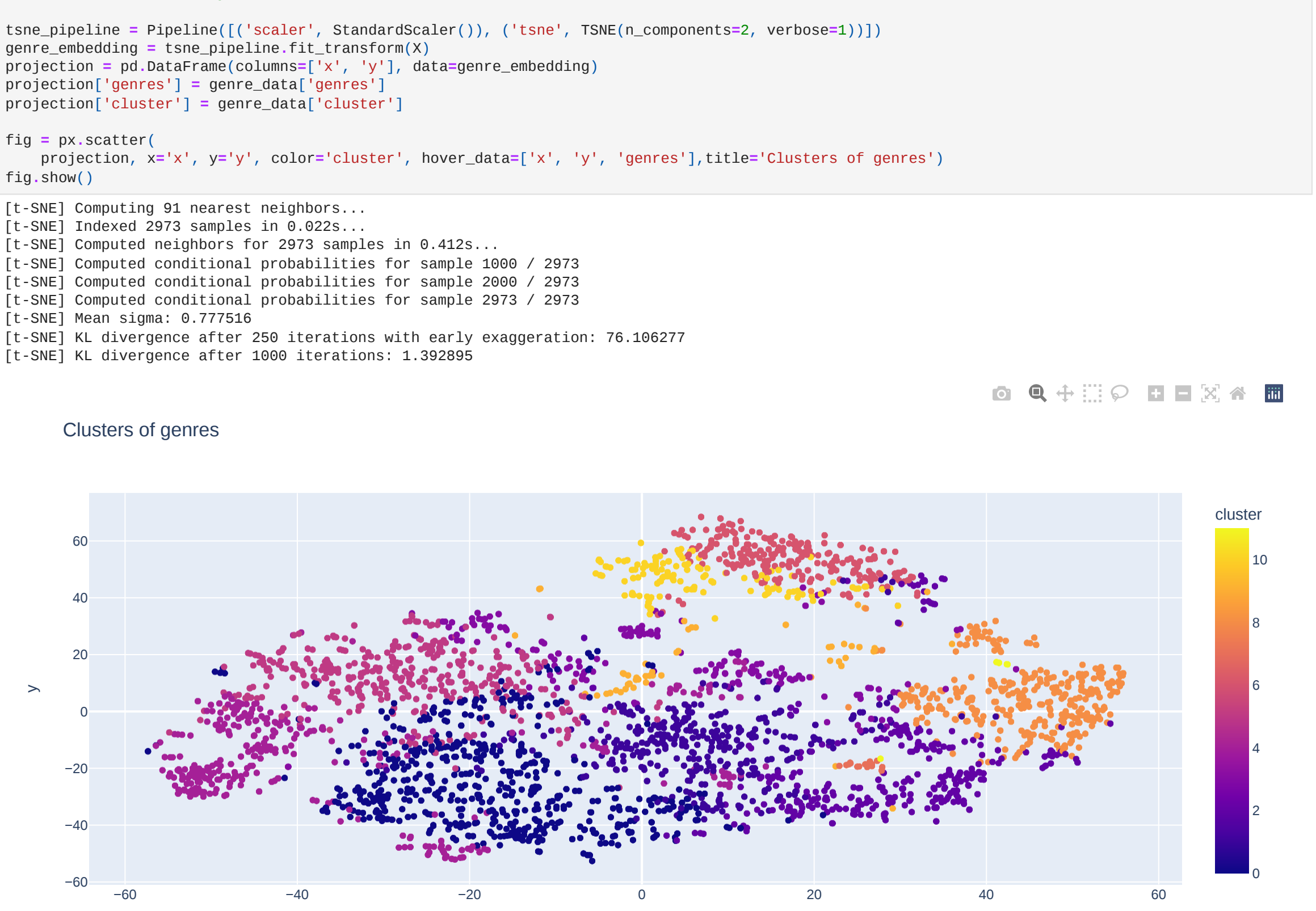
```
Unique Values in: data
------------------------------
valence               1733
year                   100
acousticness          4689
artists              34088
danceability          1240
duration_ms          51755
energy                2332
explicit                 2
id                  170653
instrumentalness      5401
key                     12
liveness              1740
loudness             25410
mode                     2
name                133638
popularity             100
release_date         10968
speechiness           1626
tempo                84694
dtype: int64

Unique Values in: genre_data
------------------------------
mode                   2
genres              2973
acousticness        2798
danceability        2725
duration_ms         2872
energy              2778
instrumentalness    2731
liveness            2709
loudness            2873
speechiness         2707
tempo               2872
valence             2745
popularity          2188
key                   12
dtype: int64

Unique Values in: year_data
------------------------------
mode                  1
year                100
acousticness        100
danceability        100
duration_ms         100
energy              100
instrumentalness    100
liveness            100
loudness            100
speechiness         100
tempo               100
valence             100
popularity          100
key                   7
dtype: int64

Unique Values in: artist_data
------------------------------
mode                    2
count                 379
acousticness        14127
artists             28680
danceability        10650
duration_ms         23960
energy              12126
instrumentalness    15517
liveness            12156
loudness            21862
speechiness         10950
tempo               24801
valence             11882
popularity           4663
key                    12
dtype: int64
```

Clustering

```python
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

cluster_pipeline = Pipeline([('scaler', StandardScaler()), ('kmeans', KMeans(n_clusters=12))])
X = genre_data.select_dtypes(np.number)
cluster_pipeline.fit(X)
genre_data['cluster'] = cluster_pipeline.predict(X)
```

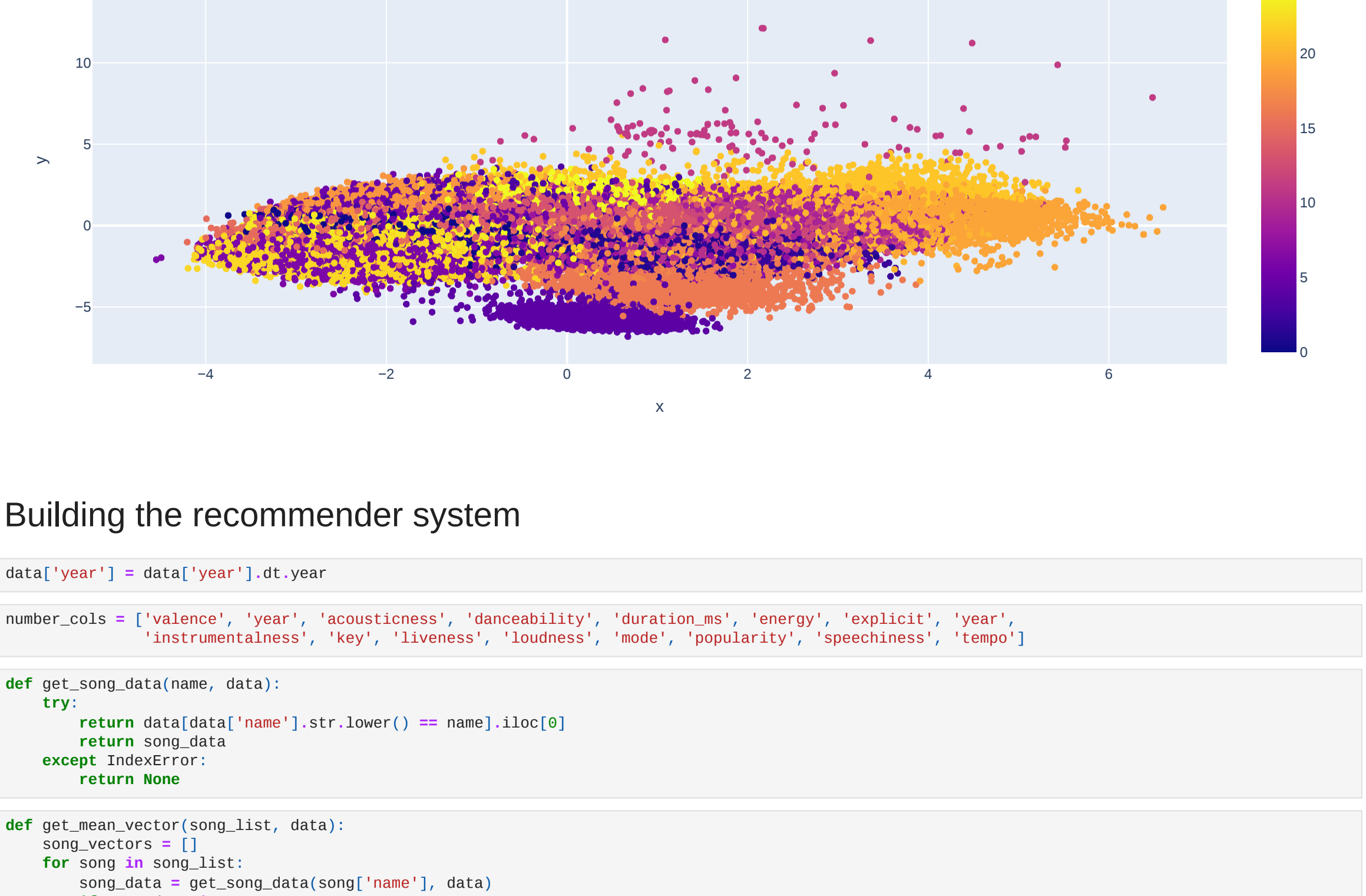
```python
from sklearn.manifold import TSNE

tsne_pipeline = Pipeline([('scaler', StandardScaler()), ('tsne', TSNE(n_components=2, verbose=1))])
genre_embedding = tsne_pipeline.fit_transform(X)
projection = pd.DataFrame(columns=['x', 'y'], data=genre_embedding)
projection['genres'] = genre_data['genres']
projection['cluster'] = genre_data['cluster']

fig = px.scatter(
    projection, x='x', y='y', color='cluster', hover_data=['x', 'y', 'genres'],title='Clusters of genres')
fig.show()
```

```
[t-SNE] Computing 91 nearest neighbors...
[t-SNE] Indexed 2973 samples in 0.022s...
[t-SNE] Computed neighbors for 2973 samples in 0.412s...
[t-SNE] Computed conditional probabilities for sample 1000 / 2973
[t-SNE] Computed conditional probabilities for sample 2000 / 2973
[t-SNE] Computed conditional probabilities for sample 2973 / 2973
[t-SNE] Mean sigma: 0.777516
[t-SNE] KL divergence after 250 iterations with early exaggeration: 76.106277
[t-SNE] KL divergence after 1000 iterations: 1.392895
```




```python
song_cluster_pipeline = Pipeline([('scaler', StandardScaler()),
                                  ('kmeans', KMeans(n_clusters=25,
                                   verbose=False))
                                 ], verbose=False)

X = data.select_dtypes(np.number)
song_cluster_pipeline.fit(X)
song_cluster_labels = song_cluster_pipeline.predict(X)
data['cluster_label'] = song_cluster_labels
```

```python
from sklearn.decomposition import PCA

pca_pipeline = Pipeline([('scaler', StandardScaler()), ('PCA', PCA(n_components=2))])
song_embedding = pca_pipeline.fit_transform(X)
projection = pd.DataFrame(columns=['x', 'y'], data=song_embedding)
projection['title'] = data['name']
projection['cluster'] = data['cluster_label']

fig = px.scatter(
    projection, x='x', y='y', color='cluster', hover_data=['x', 'y', 'title'],title='Clusters of songs')
fig.show()
```



Building the recommender system

```python
data['year'] = data['year'].dt.year
```

```python
number_cols = ['valence', 'year', 'acousticness', 'danceability', 'duration_ms', 'energy', 'explicit', 'year',
               'instrumentalness', 'key', 'liveness', 'loudness', 'mode', 'popularity', 'speechiness', 'tempo']
```

```python
def get_song_data(name, data):
    try:
        return data[data['name'].str.lower() == name].iloc[0]
        return song_data
    except IndexError:
        return None
```

```python
def get_mean_vector(song_list, data):
    song_vectors = []
    for song in song_list:
        song_data = get_song_data(song['name'], data)
        if song_data is None:
            print('Warning: {} does not exist in the dataset'.format(song['name']))
            return None
        song_vector = song_data[number_cols].values
        song_vectors.append(song_vector)
    song_matrix = np.array(list(song_vectors))
    return np.mean(song_matrix, axis=0)
```

```python
def flatten_dict_list(dict_list):
    flattened_dict = defaultdict()
    for key in dict_list[0].keys():
        flattened_dict[key] = []
    for dictionary in dict_list:
        for key, value in dictionary.items():
            flattened_dict[key].append(value)
    return flattened_dict
```

```python
min_max_scaler = MinMaxScaler()
normalized_data = min_max_scaler.fit_transform(data[number_cols])

standard_scaler = StandardScaler()
scaled_normalized_data = standard_scaler.fit_transform(normalized_data)
```

```python
def recommend_songs(seed_songs, data, n_recommendations=10):
    metadata_cols = ['name', 'artists', 'year']
    song_center = get_mean_vector(seed_songs, data)

    if song_center is None:
        return []

    normalized_song_center = min_max_scaler.transform([song_center])

    scaled_normalized_song_center = standard_scaler.transform(normalized_song_center)

    distances = cdist(scaled_normalized_song_center, scaled_normalized_data, 'euclidean')
    index = np.argsort(distances)[0]

    rec_songs = []
    for i in index:
        song_name = data.iloc[i]['name']
        if song_name not in [song['name'] for song in seed_songs] and song_name not in [song['name'] for song in rec_songs]:
            rec_songs.append(data.iloc[i])
            if len(rec_songs) == n_recommendations:
                break

    return pd.DataFrame(rec_songs)[metadata_cols].to_dict(orient='records')
```

```python
my_songs = [
    {'name': 'Stumble'},
    {'name': 'Your Song'},
]
my_songs = [{'name': name['name'].lower()} for name in my_songs]

n_recommendations = 20

recommended_songs = recommend_songs(my_songs, data, n_recommendations)

recommended_df = pd.DataFrame(recommended_songs)

for idx, song in enumerate(recommended_songs, start=1):
    print(f"{idx}. {song['name']} by {song['artists']} ({song['year']})")
```

```
1. Can't Take My Eyes Off You by ['Engelbert Humperdinck'] (1968)
2. Candle In The Wind - Remastered 2014 by ['Elton John'] (1973)
3. New Kid in Town - 2013 Remaster by ['Eagles'] (1976)
4. Reasons by ['Earth, Wind & Fire'] (1975)
5. Skyline Pigeon - Piano Version by ['Elton John'] (1973)
6. Fly Me To The Moon (In Other Words) by ['Bobby Womack'] (1969)
7. It's All Over Now, Baby Blue (feat. Van Morrison) by ['Them', 'Van Morrison'] (1966)
8. Seni Her Gördüğümde by ['Erkin Koray'] (1974)
9. Goodbye Yellow Brick Road - Remastered 2014 by ['Elton John'] (1973)
10. Blinded By The Light - Single edit by ["Manfred Mann's Earth Band"] (1976)
11. Voici les clés by ['Gerard Lenorman'] (1976)
12. I Never Loved a Man (The Way I Love You) by ['Aretha Franklin'] (1967)
13. Beautiful Boy (Darling Boy) - Stripped Down Mix, 2010 by ['John Lennon'] (1980)
14. I Can See Clearly Now - Edit by ['Johnny Nash'] (1972)
15. I Can See Clearly Now by ['Johnny Nash'] (1972)
16. Get Together by ['The Youngbloods'] (1967)
17. Metamorfose Ambulante by ['Raul Seixas'] (1973)
18. Stay by ['Oingo Boingo'] (1985)
19. Que Te Vaya Bonito by ['Vicente Fernández'] (1972)
20. Maggie May by ['Rod Stewart'] (1971)
```