```
"""
Analysis of Popular Songs on Spotify and Youtube
Renee Singh and Nila Ragu
"""
```

```
'\nAnalysis of Popular Songs on Spotify and Youtube\nRenee Singh and Nila Ragu\n'
```

Double-click (or enter) to edit

Step 0: Imports and Data Manipulation

```
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
from sklearn.preprocessing import MinMaxScaler
pd.options.display.float_format = '{:.1f}'.format
```

```
spotify_2000 = pd.read_csv('Spotify_2000.csv')
spotify_yt = pd.read_csv('Spotify_Youtube.csv')
spotify_features = pd.read_csv('Spotify_Features.csv')

# Visualization dataset
spotify = spotify_2000.merge(spotify_yt, left_on=['Title', 'Artist'],
                             right_on=['Track', 'Artist'], how='inner')
pd.set_option('display.max_columns', None)

# ML dataset
spotify_ml = spotify_yt.merge(spotify_features, left_on=['Artist', 'Track'],
                              right_on=['artist_name', 'track_name'],
                              how='inner')
```

Show hidden output

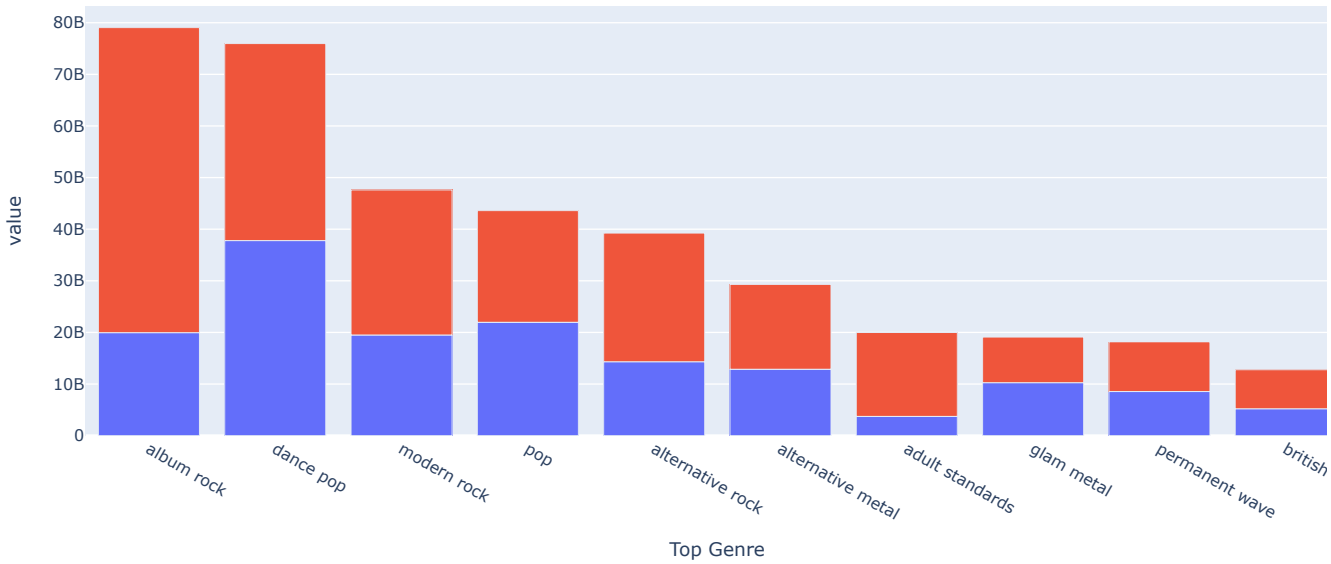
Step 1: Platform analysis

Questions to answer:

For the platform analysis, we decided to focus on the top 10 most popular genres. This was calculated by summing up all the views and streams of songs in that genre.

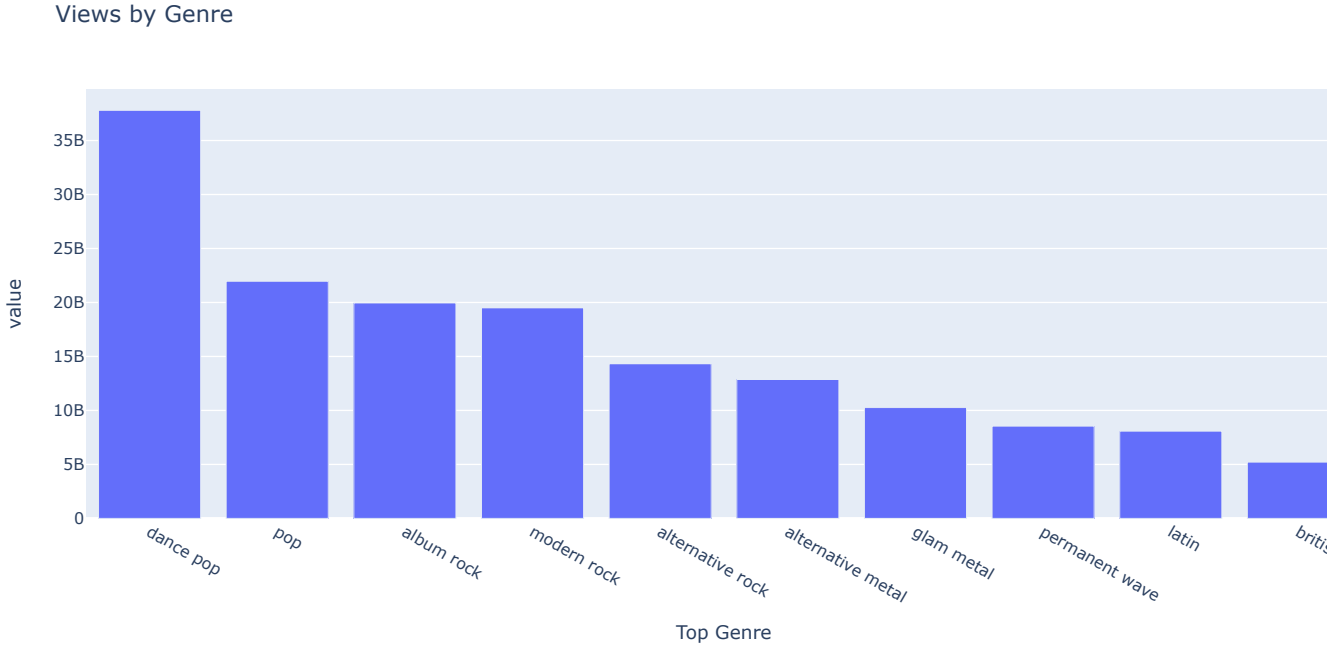
1. What platform are songs more popular on?

```
# Getting top 10 most viewed/streamed genres
genre = spotify.groupby('Top Genre')[['Views', 'Stream']].sum().reset_index()
genre['Total Views/Streams'] = genre['Views'] + genre['Stream']

# Graph
genre['Difference'] = genre['Views'] - genre['Stream']
genre_views_and_streams = genre.nlargest(10, 'Total Views/Streams')

genre_total_fig = px.bar(genre_views_and_streams,
                         x='Top Genre', y=['Views', 'Stream'],
                         title='Views and Streams by Genre')
genre_total_fig
```

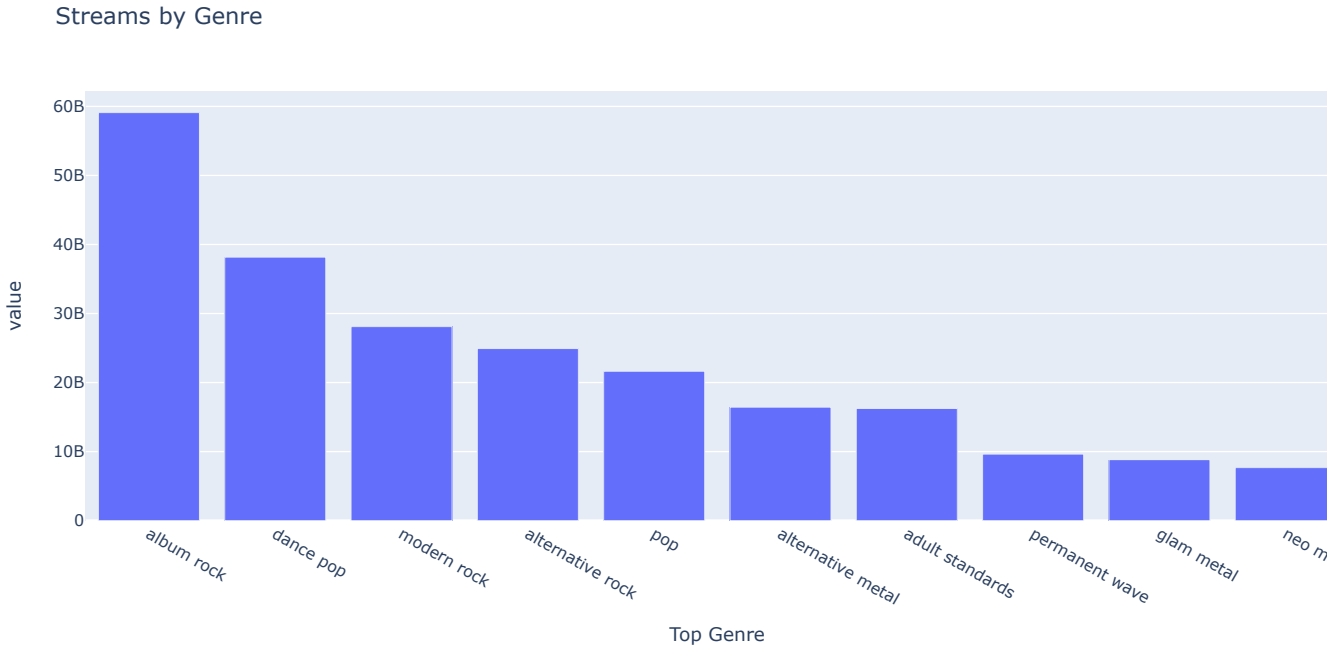


2. What is the difference in popularity for genres, depending on the platform?

```
genre_views = genre.nlargest(10, 'Views')
genre_views_fig = px.bar(genre_views,
                         x='Top Genre', y=['Views'],
                         title='Views by Genre')
genre_views_fig
```

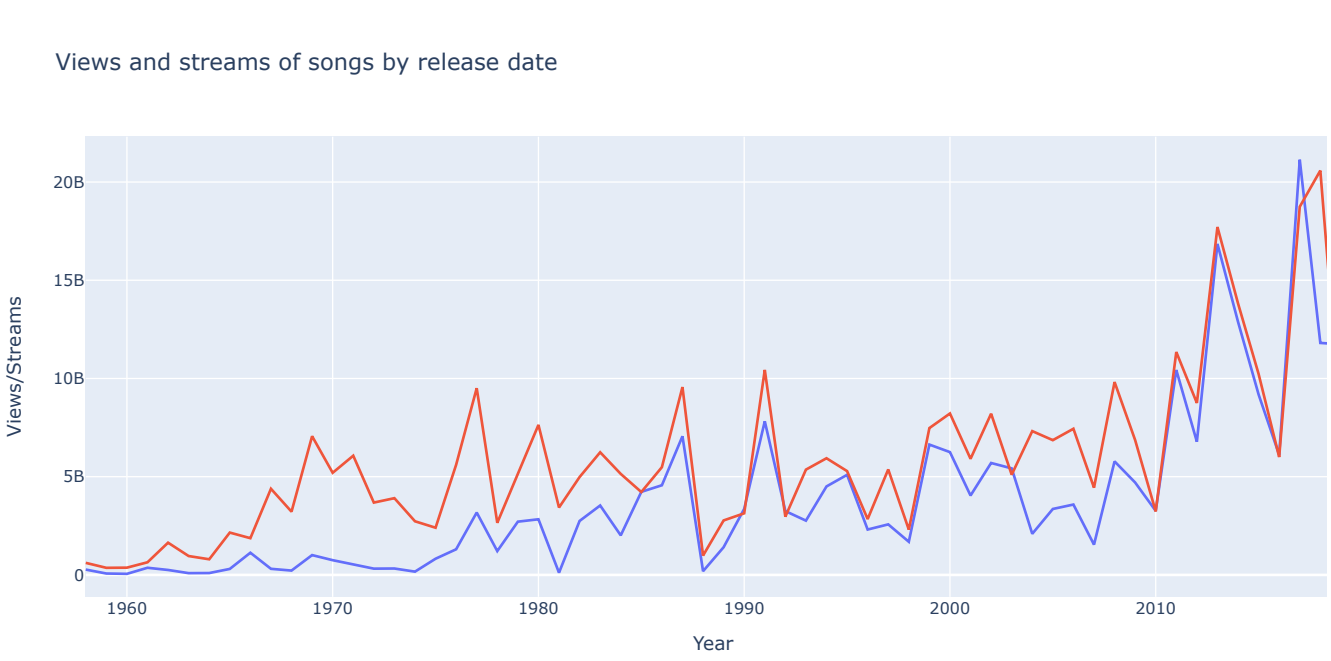



```
genre_streams = genre.nlargest(10, 'Stream')
genre_streams_fig = px.bar(genre_streams,
                           x='Top Genre', y=['Stream'],
                           title='Streams by Genre')
genre_streams_fig
```



3. How do the total amount of streams and views vary depending on release date?

```
platform_year = spotify.groupby('Year')[['Views',
                                         'Stream']].sum().reset_index()

platform_year_fig = go.Figure()
platform_year_fig.add_trace(go.Scatter(x=platform_year['Year'],
                                       y=platform_year['Views'],
                                       name='Views'))
platform_year_fig.add_trace(go.Scatter(x=platform_year['Year'],
                                       y=platform_year['Stream'],
                                       name='Streams'))
platform_year_fig.update_layout(title='Views and streams of songs by'
                                ' release date',
                                xaxis_title='Year',
                                yaxis_title='Views/Streams')

platform_year_fig
```



Step 2: Genre and Auditory Quality Analysis

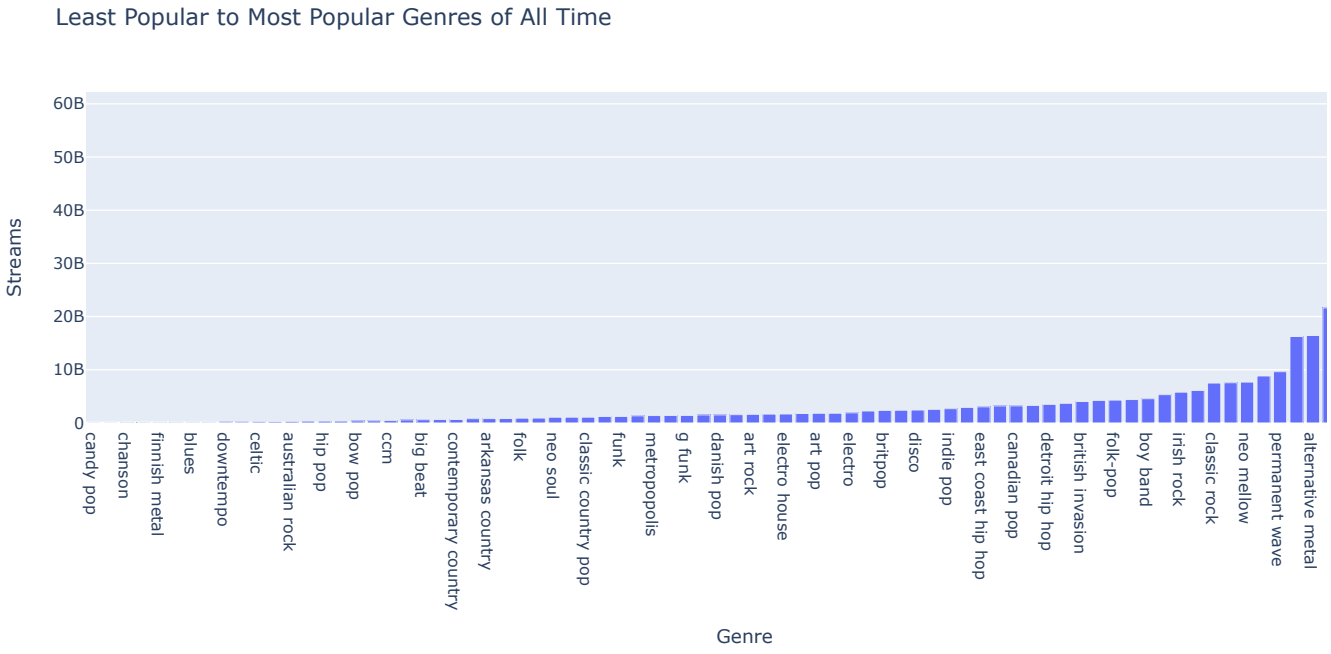
Questions to answer:

1. What are the popular genres, from least to greatest?
2. What are the auditory quality averages by genre?
3. How do the auditory qualities of a song impact its popularity?
4. How did the popularity of genres rise and fall over time?
5. How did various auditory qualities change over time?
6. What are the average auditory qualities depending on artist?
7. How many songs from the top 10 genres were released every year?

These questions will be looking at the streams of songs.

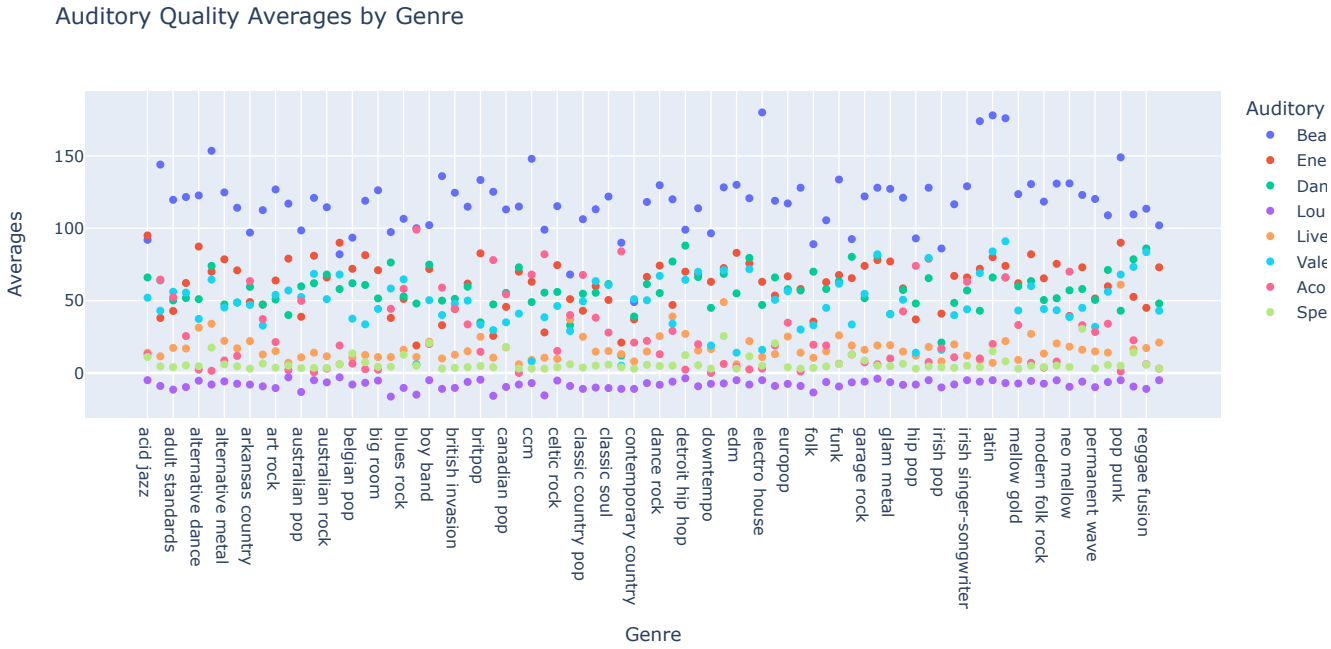
1. What are the popular genres, from least popular to most popular?

```
pop_genre = spotify.sort_values(by='Stream', ascending=True)
popular = pop_genre.groupby('Top Genre')['Stream'].sum().reset_index()
fig0 = px.bar(popular, x='Top Genre',
              y=popular['Stream'],
              title='Least Popular to Most Popular Genres of All Time')
fig0.update_layout(
  xaxis_title='Genre',
  yaxis_title='Streams',
  xaxis={'categoryorder': 'total ascending'})
fig0.show()
```



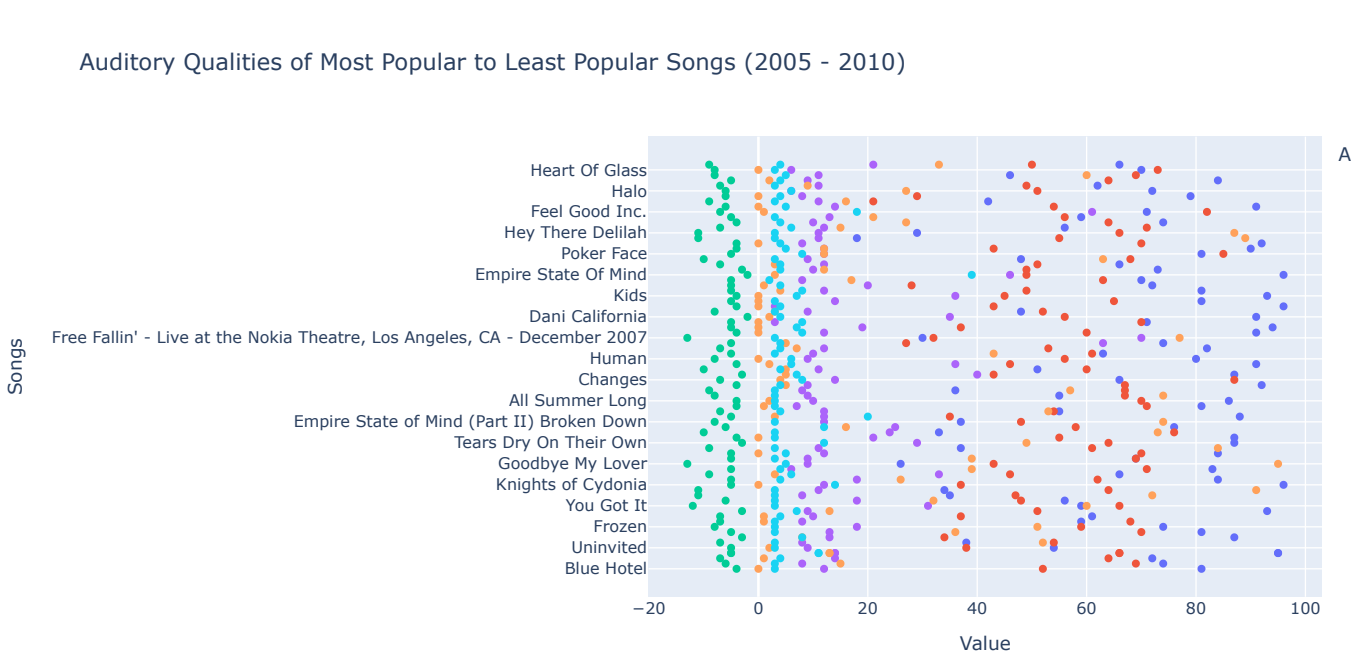
2. What are the auditory quality averages by genre?

```
avg = spotify.groupby('Top Genre')[['Beats Per Minute (BPM)', 'Energy_x',
                                    'Danceability_x', 'Loudness (dB)',
                                    'Liveness_x', 'Valence_x',
                                    'Acousticness_x',
                                    'Speechiness_x']].mean().reset_index()
fig = px.scatter(avg, x='Top Genre', y=['Beats Per Minute (BPM)', 'Energy_x',
                                        'Danceability_x', 'Loudness (dB)',
                                        'Liveness_x', 'Valence_x',
                                        'Acousticness_x', 'Speechiness_x'],
                          title='Auditory Quality Averages by Genre')
fig.update_layout(
  xaxis_title='Genre',
  yaxis_title='Averages',
  legend_title='Auditory Quality')
fig.show()
```



3. How do the auditory qualities of a song impact its popularity?

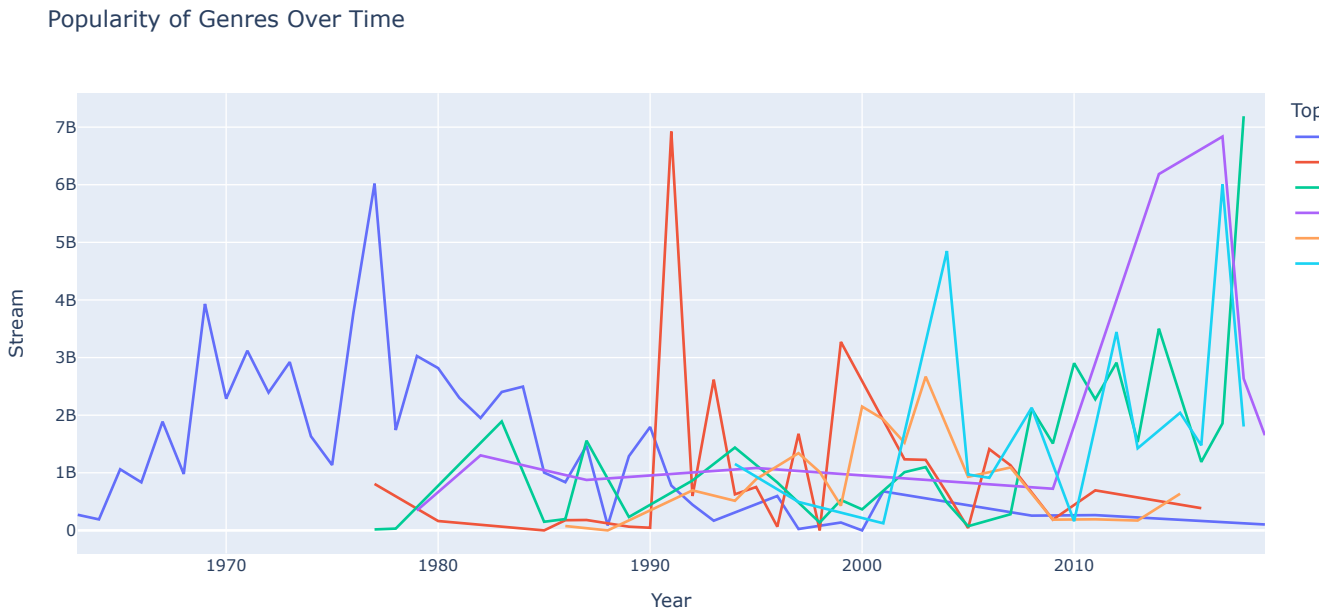
```
years = spotify.loc[(spotify['Year'] >= 2005) & (spotify["Year"] <= 2010)]
sorted_pop = years.sort_values(by='Stream', ascending=True)
fig2 = px.scatter(sorted_pop, y='Title_x', x=['Energy_x',   'Danceability_x',
                                              'Loudness (dB)', 'Liveness_x',
                                              'Acousticness_x',
                                              'Speechiness_x'],
                  title='Auditory Qualities of Most Popular to Least'
                        ' Popular Songs (2005 - 2010)')
fig2.update_layout(
  xaxis_title='Value',
  yaxis_title='Songs',
  legend_title='Auditory Quality')
fig2.show()
```



4. How did the popularity of the top 6 genres of all time rise and fall over the years?

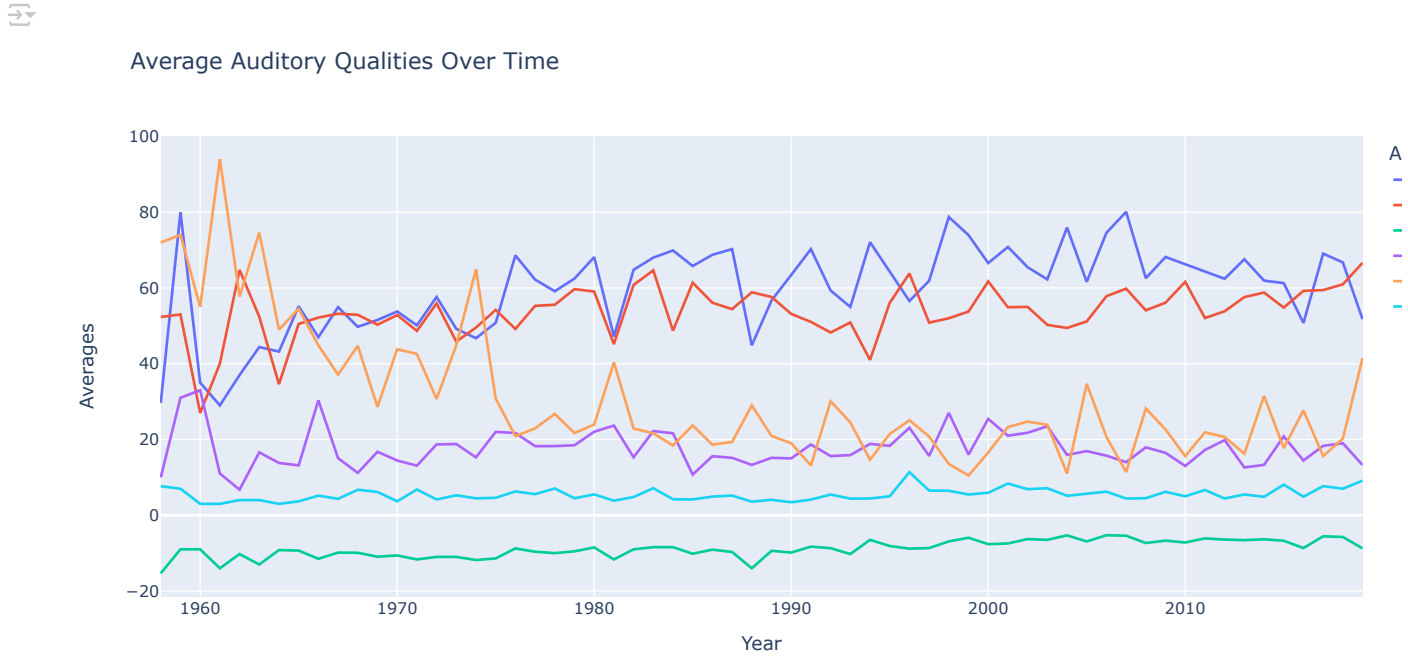
```
filtered_genres = spotify[(spotify['Top Genre'] == 'pop') |
                          (spotify['Top Genre'] == 'alternative rock') |
                          (spotify['Top Genre'] == 'modern rock') |
                          (spotify['Top Genre'] == 'album rock') |
```

```
                          (spotify['Top Genre'] == 'alternative metal') |
                          (spotify['Top Genre'] == 'dance pop')]
genre_groups = filtered_genres.groupby(['Year', 'Top Genre'])['Stream']\
                                    .sum().reset_index()
fig4 = px.line(genre_groups, y='Stream', x='Year', color='Top Genre',
               title='Popularity of Genres Over Time')
fig4.show()
```



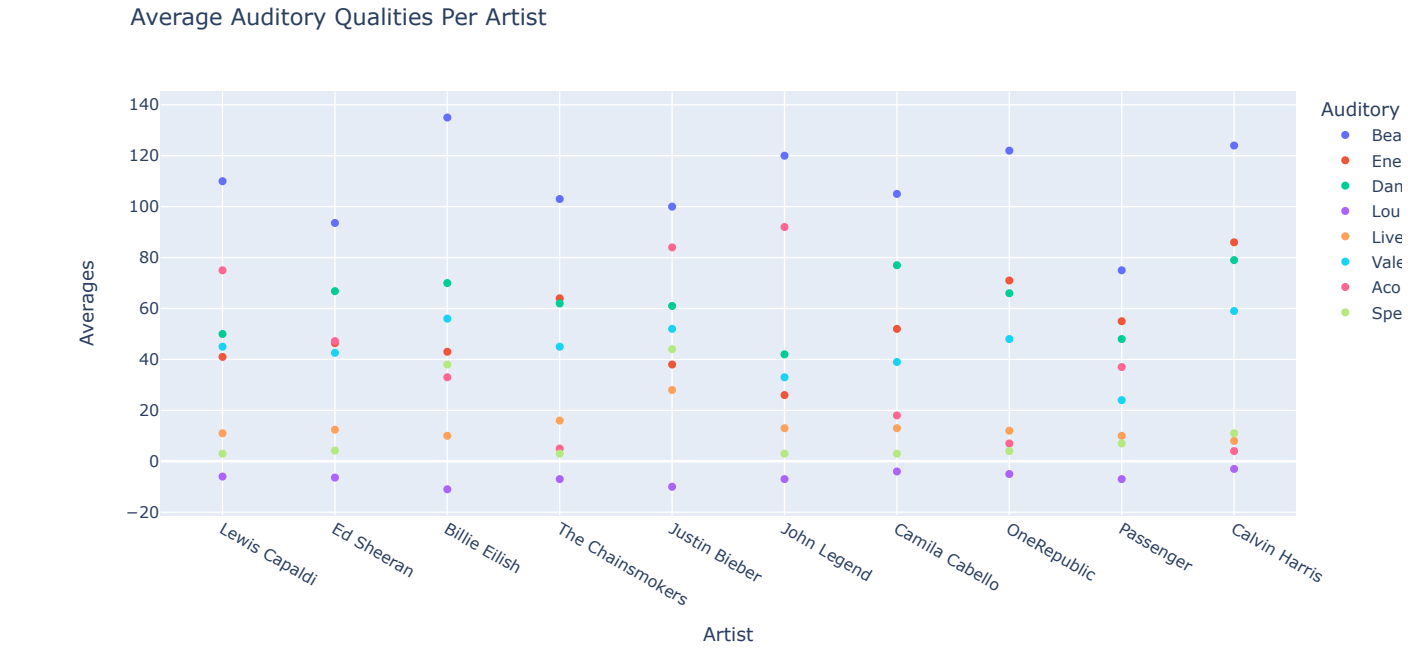
5. How did various auditory qualities change over time?

```
audio = spotify.groupby('Year')[['Beats Per Minute (BPM)', 'Energy_x',
                                 'Danceability_x', 'Loudness (dB)',
                                 'Liveness_x', 'Valence_x',
                                 'Acousticness_x',
                                 'Speechiness_x']].mean().reset_index()
fig6 = px.line(audio, x="Year", y=['Energy_x', 'Danceability_x',
                                   'Loudness (dB)', 'Liveness_x',
                                   'Acousticness_x',
                                   'Speechiness_x'],
               title='Average Auditory Qualities Over Time')
fig6.update_layout(
  yaxis_title='Averages',
  legend_title='Auditory Quality')
fig6.show()
```



6. What are the average auditory qualities for the top 10 artists of all time?

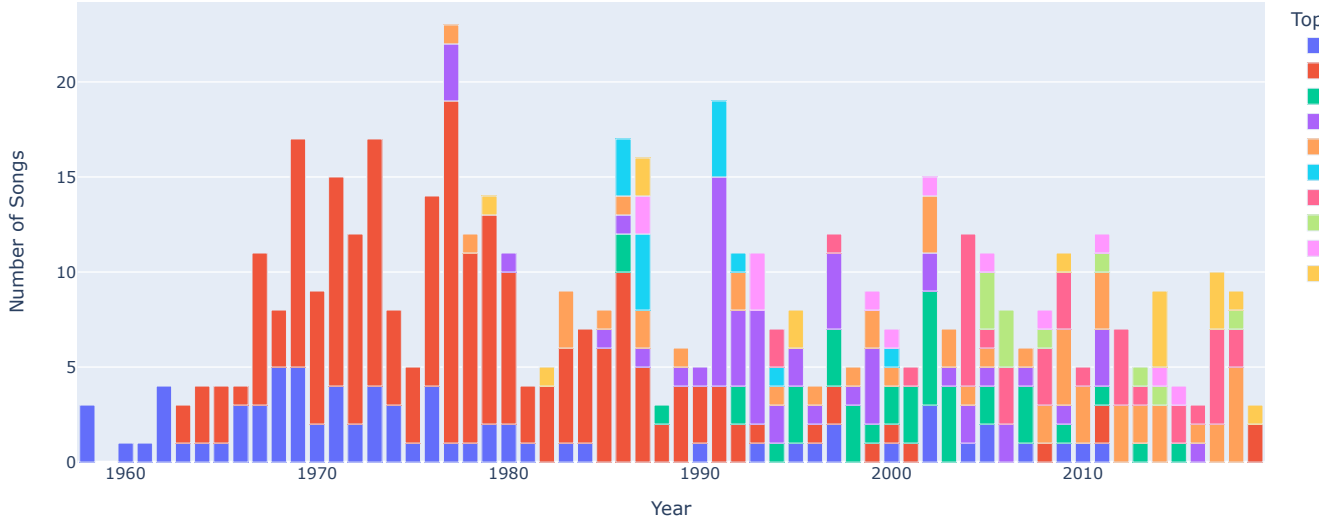
```
top = spotify.groupby('Artist')[['Beats Per Minute (BPM)', 'Energy_x',
                                 'Danceability_x', 'Loudness (dB)',
                                 'Liveness_x', 'Valence_x',
                                 'Acousticness_x', 'Speechiness_x',
                                 'Stream']].mean().reset_index()
top10 = top.nlargest(10, 'Stream')
fig7 = px.scatter(top10, x='Artist', y=['Beats Per Minute (BPM)', 'Energy_x',
                                       'Danceability_x',   'Loudness (dB)',
                                       'Liveness_x', 'Valence_x',
                                       'Acousticness_x', 'Speechiness_x'],
                  title='Average Auditory Qualities Per Artist')
fig7.update_layout(
  yaxis_title='Averages',
  legend_title='Auditory Quality')
fig7.show()
```



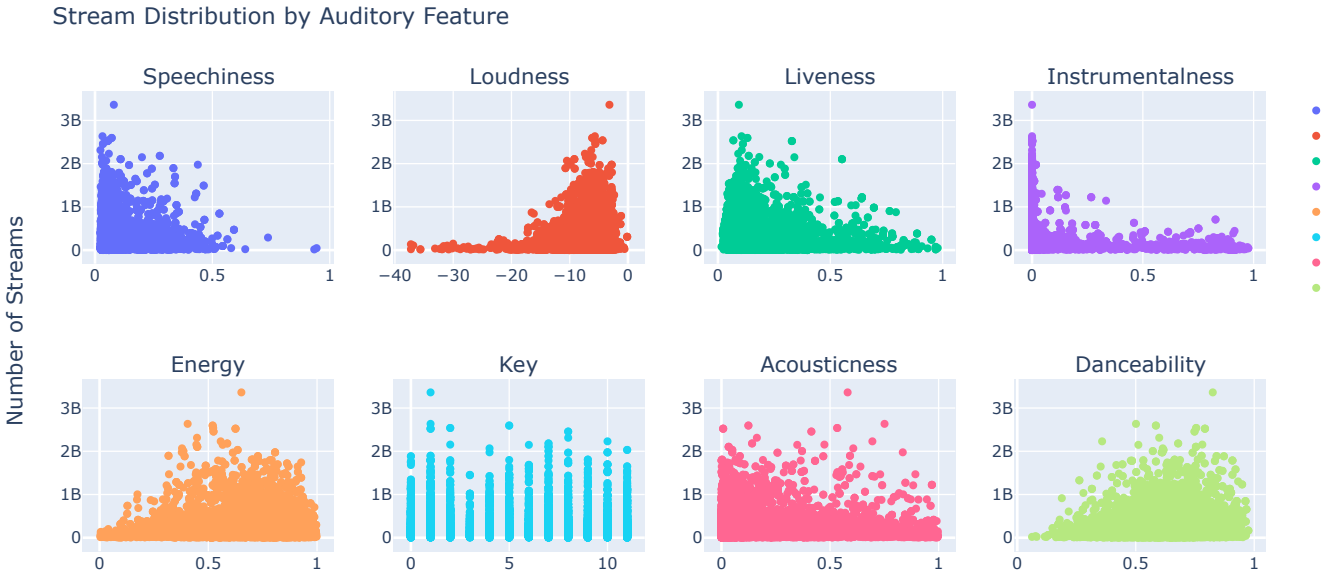
7. How many songs from the top 10 genres were released every year?

```
genre_streams_list = list(genre_streams['Top Genre'])
genre_count_df = spotify[spotify['Top Genre'].isin(genre_streams_list)]
```

```
genre_count_df = genre_count_df.groupby(['Top Genre', 'Year'])[['Title_x']]\
                                    .count().reset_index()
genre_count_df_fig = px.bar(genre_count_df, x='Year',
                            y='Title_x', color='Top Genre',
                            labels={'Title_x': 'Number of Songs Released'},
                            title='Number of Popular Songs Released Per Year'
                                  ' In Top 10 Most Streamed Genres')
genre_count_df_fig.update_layout(yaxis_title='Number of Songs',)
genre_count_df_fig
```




```
# Stream distributions by Auditory Features Graph
ml_fig = go.Figure()
ml_fig = make_subplots(rows=2, cols=4,
                       y_title='Number of Streams',
                       subplot_titles=('Speechiness', 'Loudness', 'Liveness',
                                       'Instrumentalness',
                                       'Energy', 'Key', 'Acousticness',
                                       'Danceability'))
ml_fig.add_trace(go.Scatter(x=spotify_ml['Speechiness'],
                 y=spotify_ml['Stream'],
                 mode='markers', name='Speechiness'),
                 row=1, col=1)

ml_fig.add_trace(go.Scatter(x=spotify_ml['Loudness'], y=spotify_ml['Stream'],
                            mode='markers', name='Loudness'),
                 row=1, col=2)
ml_fig.add_trace(go.Scatter(x=spotify_ml['Liveness'], y=spotify_ml['Stream'],
                            mode='markers', name='Liveness'),
                 row=1, col=3)
ml_fig.add_trace(go.Scatter(x=spotify_ml['Instrumentalness'],
                            y=spotify_ml['Stream'],
                            mode='markers', name='Instrumentalness'),
                 row=1, col=4)
ml_fig.add_trace(go.Scatter(x=spotify_ml['Energy'], y=spotify_ml['Stream'],
                            mode='markers', name='Energy'),
                 row=2, col=1)
ml_fig.add_trace(go.Scatter(x=spotify_ml['Key'], y=spotify_ml['Stream'],
                            mode='markers', name='Key'),
                 row=2, col=2)
ml_fig.add_trace(go.Scatter(x=spotify_ml['Acousticness'], y=spotify_ml['Stream'],
                            mode='markers', name='Acousticness'),
                 row=2, col=3)
ml_fig.add_trace(go.Scatter(x=spotify_ml['Danceability'],
                            y=spotify_ml['Stream'],
                            mode='markers', name='Danceability'),
                 row=2, col=4)

ml_fig.update_layout(title_text='Stream Distribution by Auditory Feature')

ml_fig
```



Step 3: Machine Learning

Questions to answer:

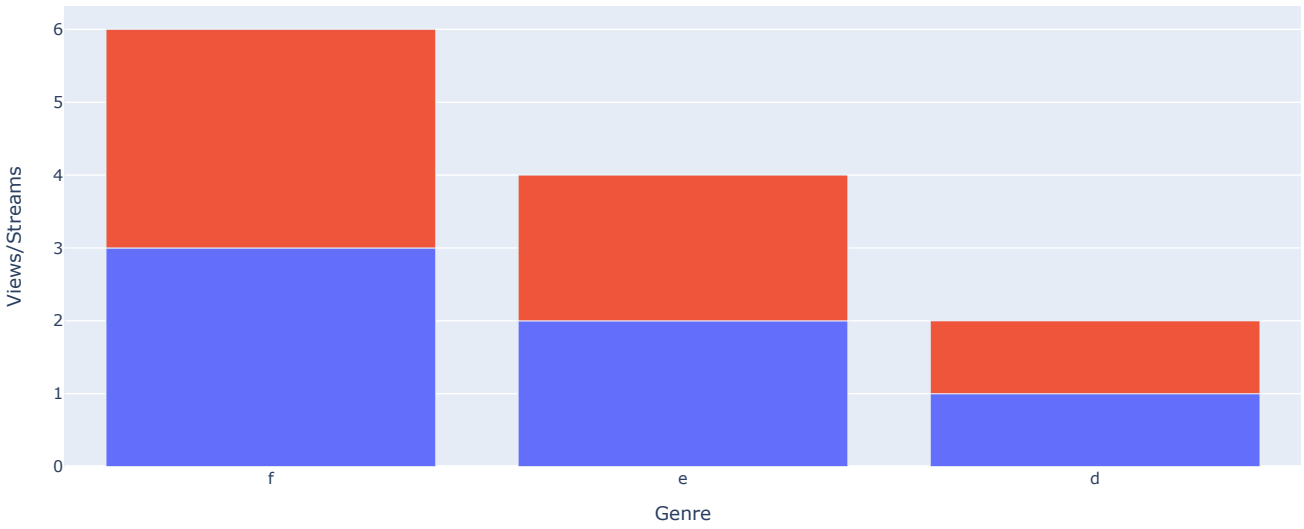
1. Based on this model, can the amount of streams be predicted given speechiness, loudness, liveness, and instrumentalness?

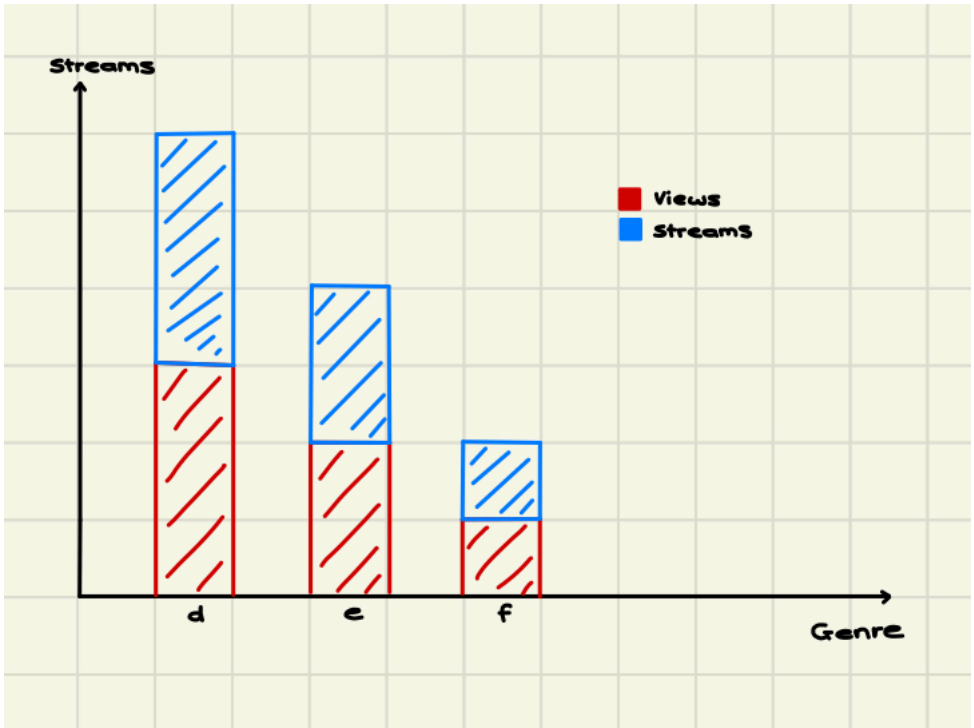
```
learning_model = spotify_ml.dropna()
labels = learning_model['Stream']
cols = ['Speechiness', 'Loudness', 'Liveness', 'Instrumentalness']
features = learning_model[cols]

# Normalizing the features
scaler = MinMaxScaler(feature_range=(0, 2))
scaled_features = scaler.fit_transform(features)
normalized_features = pd.DataFrame(scaled_features, columns=cols)

# Splitting the features
features = pd.get_dummies(normalized_features)
features_train, features_test, labels_train, labels_test = \
  train_test_split(features, labels, test_size=0.3)

# Training the model
model = KNeighborsRegressor()
model.fit(features_train, labels_train)

train_predictions = model.predict(features_train)
test_predictions = model.predict(features_test)

r2_score_train = r2_score(labels_train, train_predictions)
print('r2_score_train: ', r2_score_train)
r2_score_test = r2_score(labels_test, test_predictions)
print('r2_score_test: ', r2_score_test)

print(kneighbors_graph(X, mode='connectivity'))
```

```
r2_score_train:  0.40837171617231094
r2_score_test:  0.07527043596506822
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-58-60c6020e043d> in <cell line: 28>()
     26 print('r2_score_test: ', r2_score_test)
     27
---> 28 print(kneighbors_graph(X, mode='connectivity'))

NameError: name 'kneighbors_graph' is not defined
```

Step 4: Testing

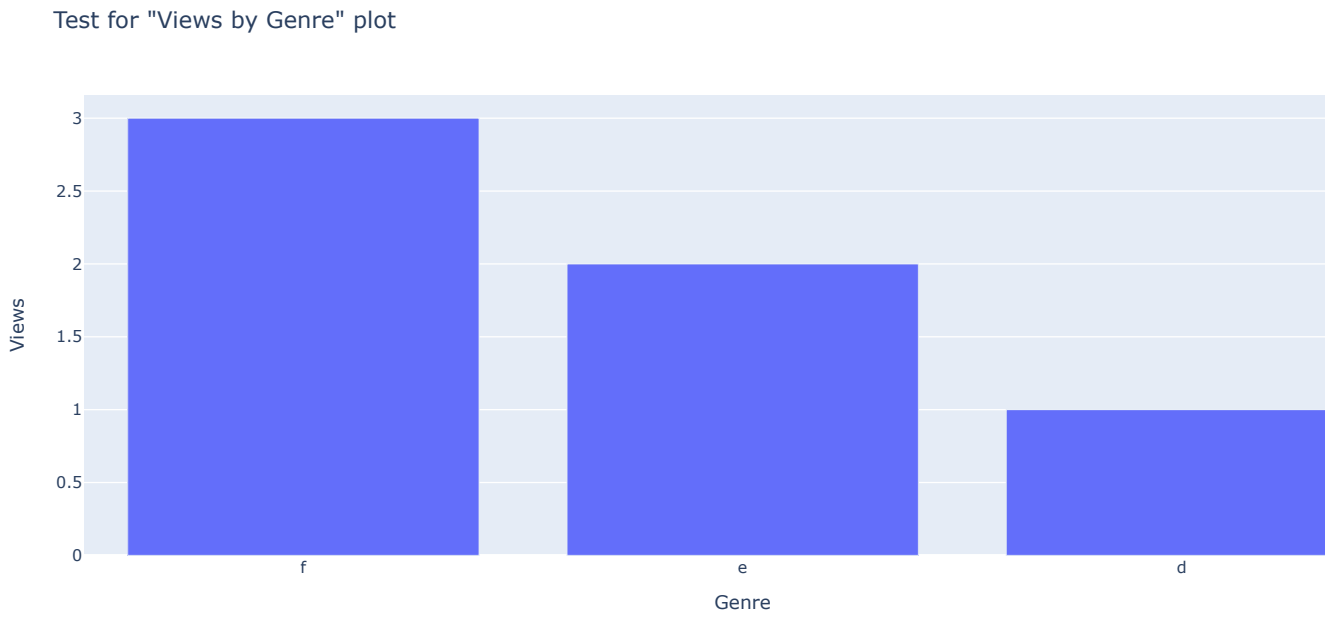
```
testfile = pd.read_csv('testfile.csv')
testfilea = pd.read_csv('testfilea.csv')
testfileb = pd.read_csv('testfileb.csv')


# Test for "Views and Streams by Genre"
testfile1 = testfile.groupby('genre')[['views', 'streams']].sum().reset_index()
testfile1['Total Views/Streams'] = testfile1['views'] + testfile1['streams']
testfile1 = testfile1.nlargest(3, 'Total Views/Streams')
testfig1 = px.bar(testfile1, x='genre', y=['views', 'streams'],
                  title='Test for "Views and Streams by Genre" plot')
testfig1.update_layout(xaxis_title='Genre',
                       yaxis_title='Views/Streams',
                       legend_title='Views/Streams')
testfig1
```

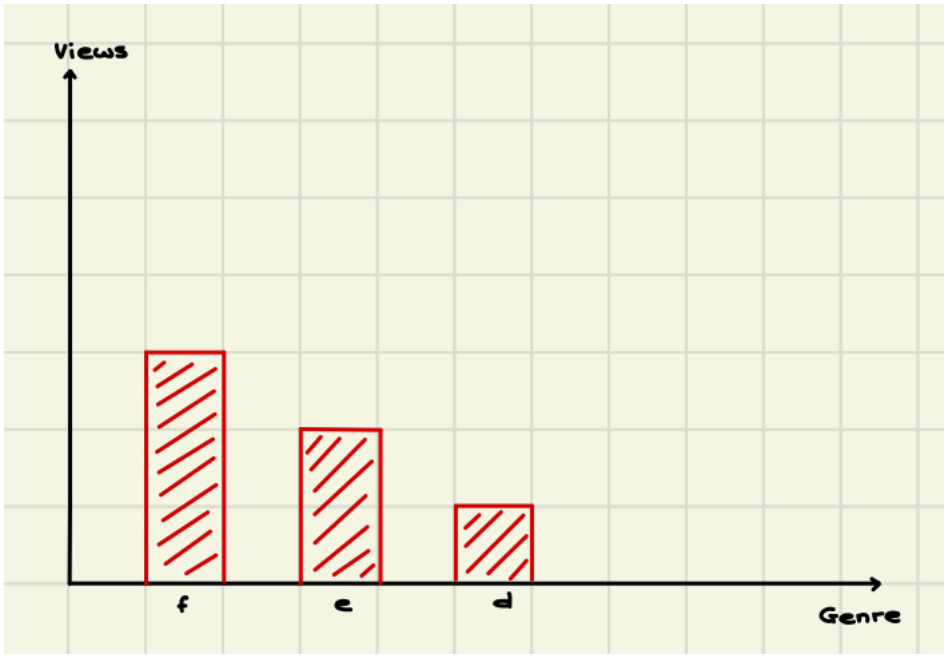




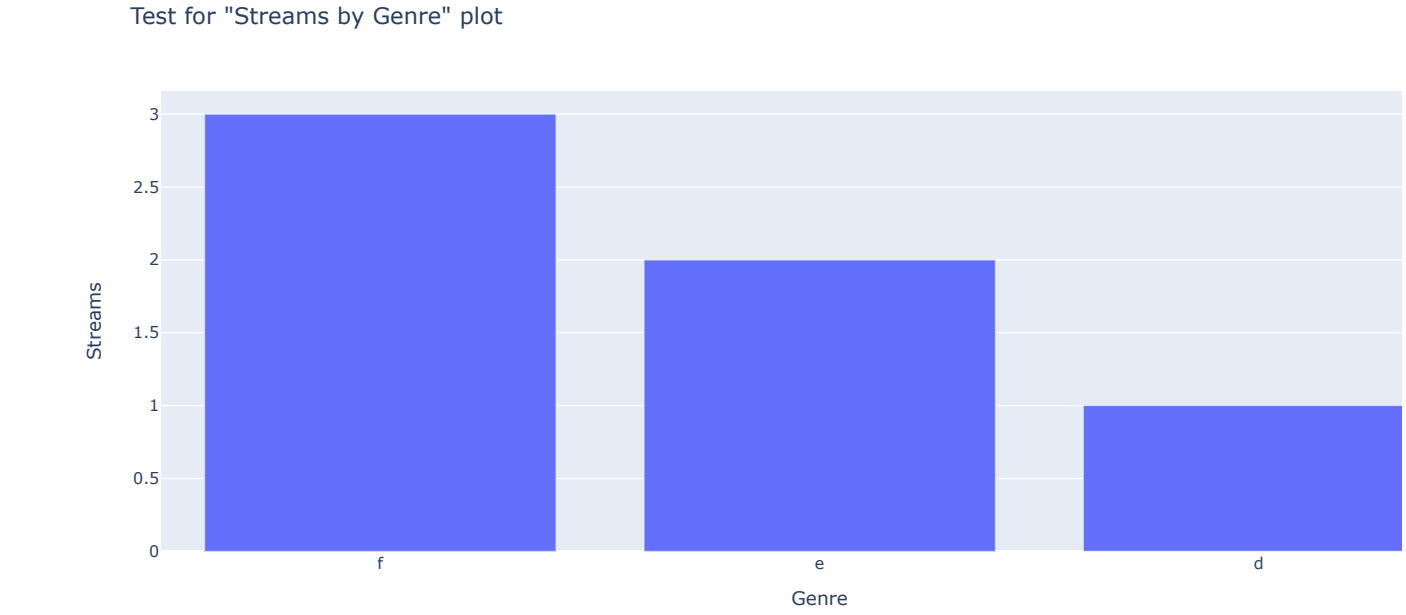

```
# Test for "Views by Genre"
testfile2 = testfile.nlargest(3, 'views')
testfig2 = px.bar(testfile2, x='genre', y='views',
```

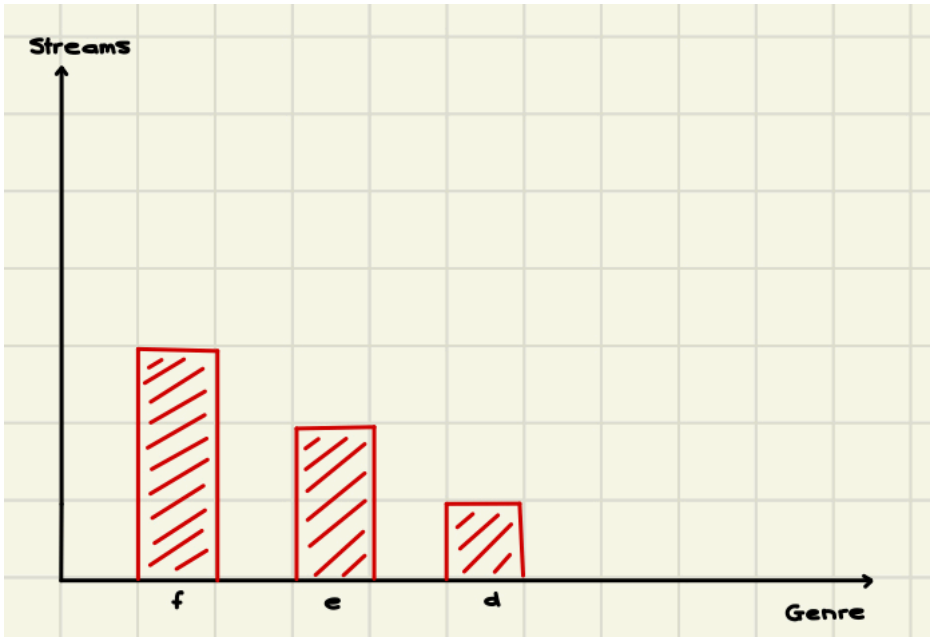
```
                  title='Test for "Views by Genre" plot')
testfig2.update_layout(xaxis_title='Genre',
                       yaxis_title='Views')
testfig2
```



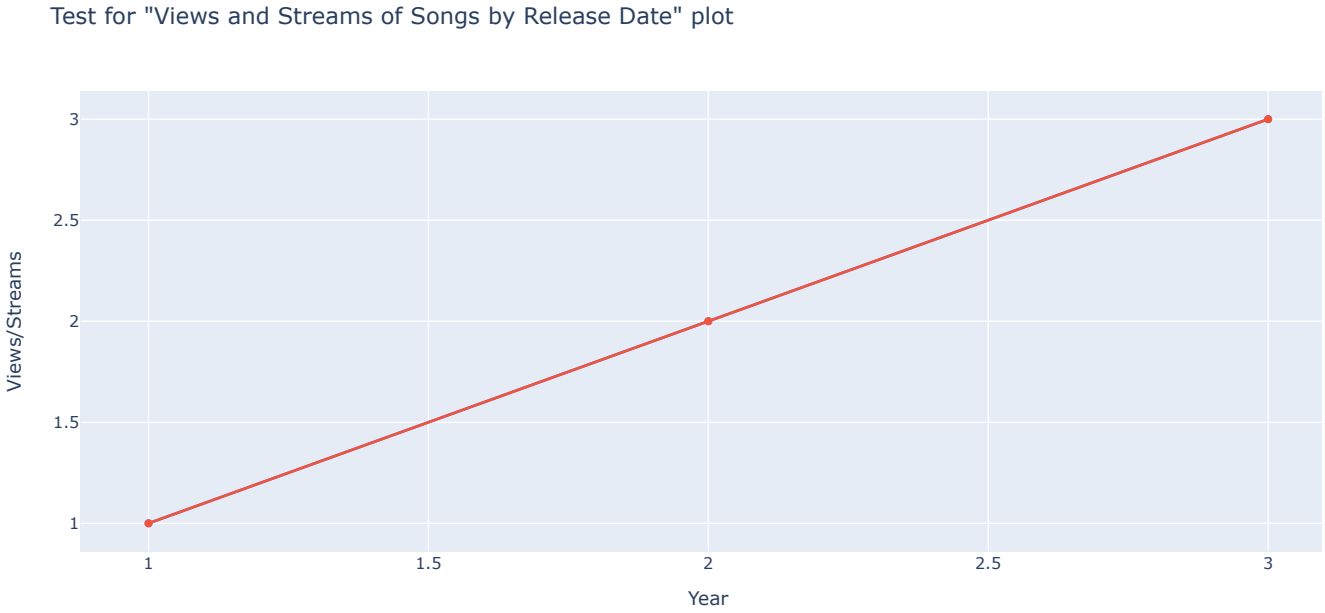


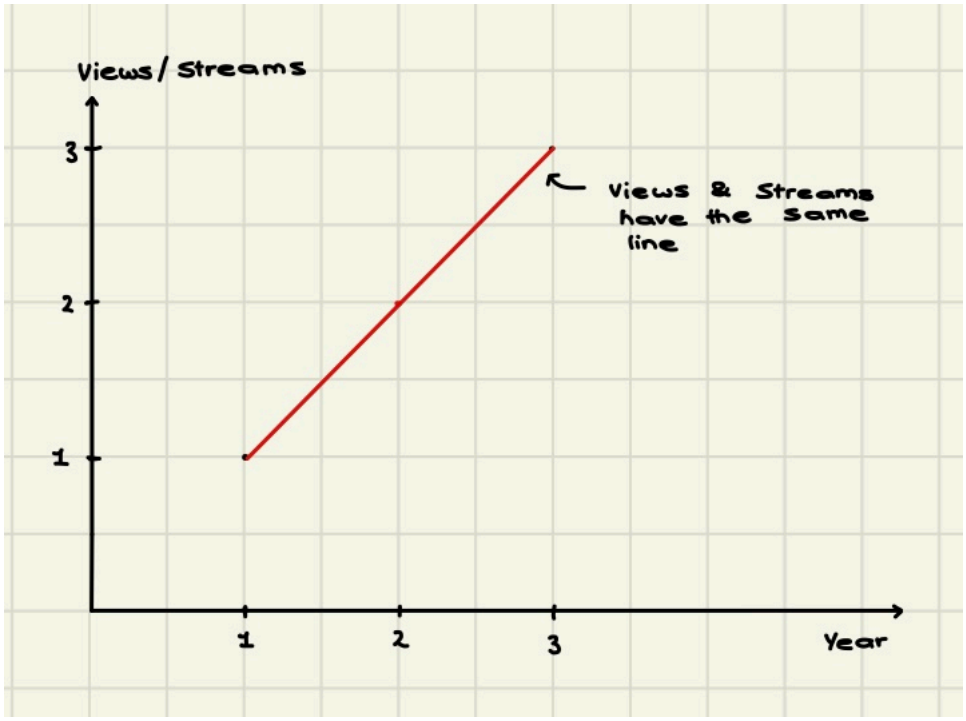

```
# Test for "Streams by Genre"
testfig3 = px.bar(testfile2, x='genre', y='streams',
                  title='Test for "Streams by Genre" plot')
testfig3.update_layout(xaxis_title='Genre',
                       yaxis_title='Streams')
testfig3
```



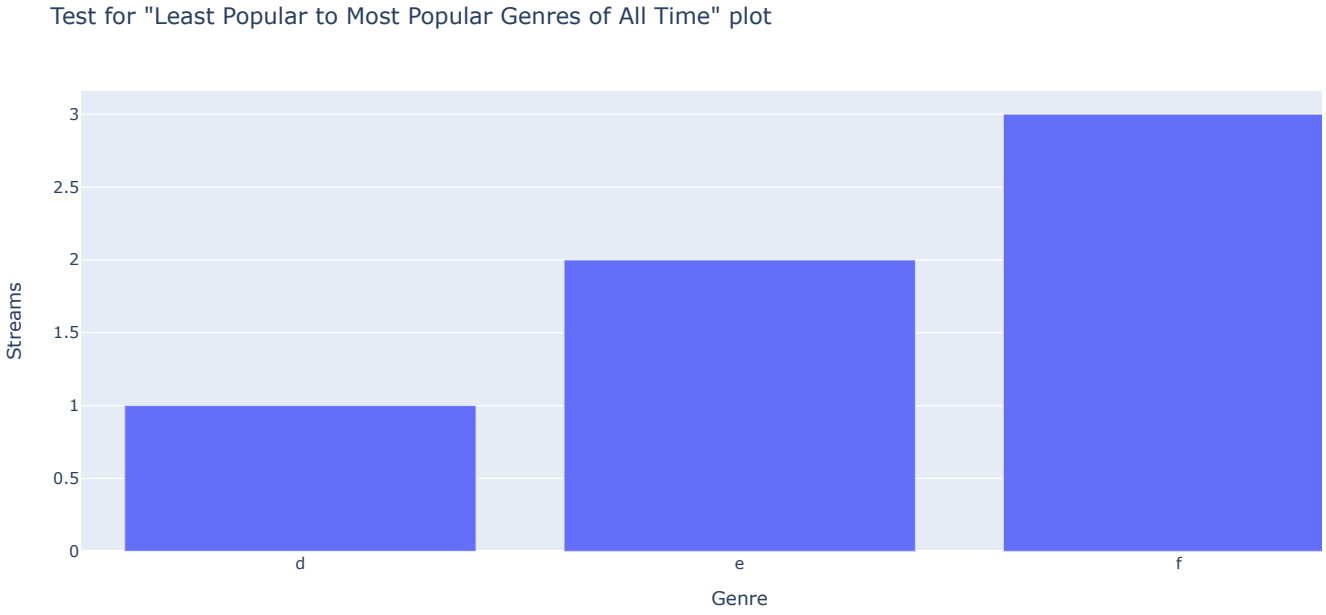


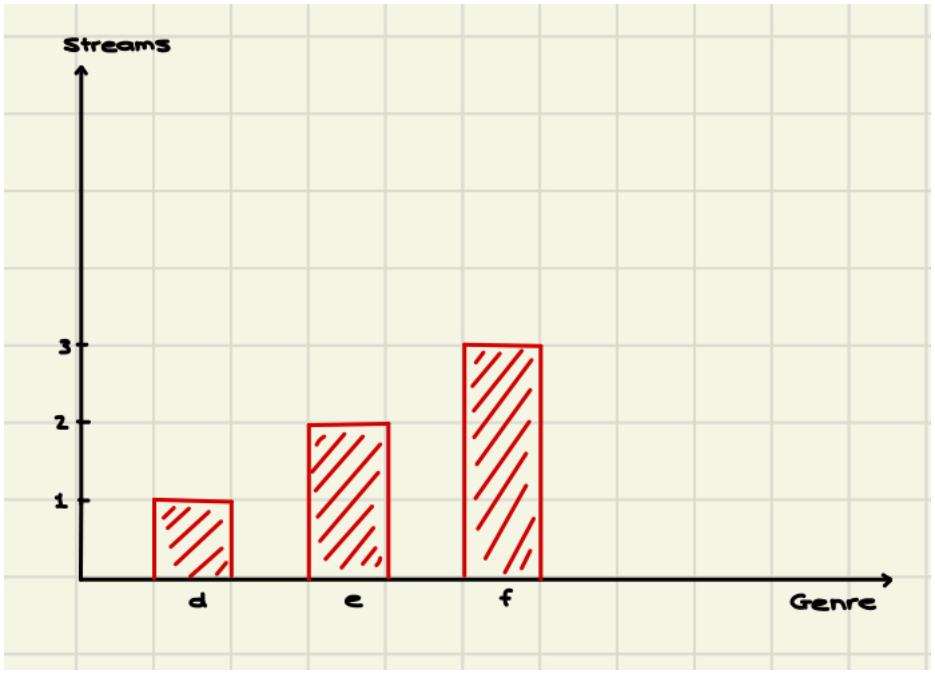

```
# Test for "Views and Streams of Songs by Release Date"
testfile4 = testfile.groupby('year')[['views', 'streams']].sum().reset_index()

testfig4 = go.Figure()
testfig4.add_trace(go.Scatter(x=testfile4['year'],
                              y=testfile4['views'],
                              name='Views'))
testfig4.add_trace(go.Scatter(x=testfile4['year'],
                              y=testfile4['streams'],
                              name='Streams'))
testfig4.update_layout(title='Test for "Views and Streams of Songs by'
                       ' Release Date" plot',
                       xaxis_title='Year',
                       yaxis_title='Views/Streams')

testfig4
```



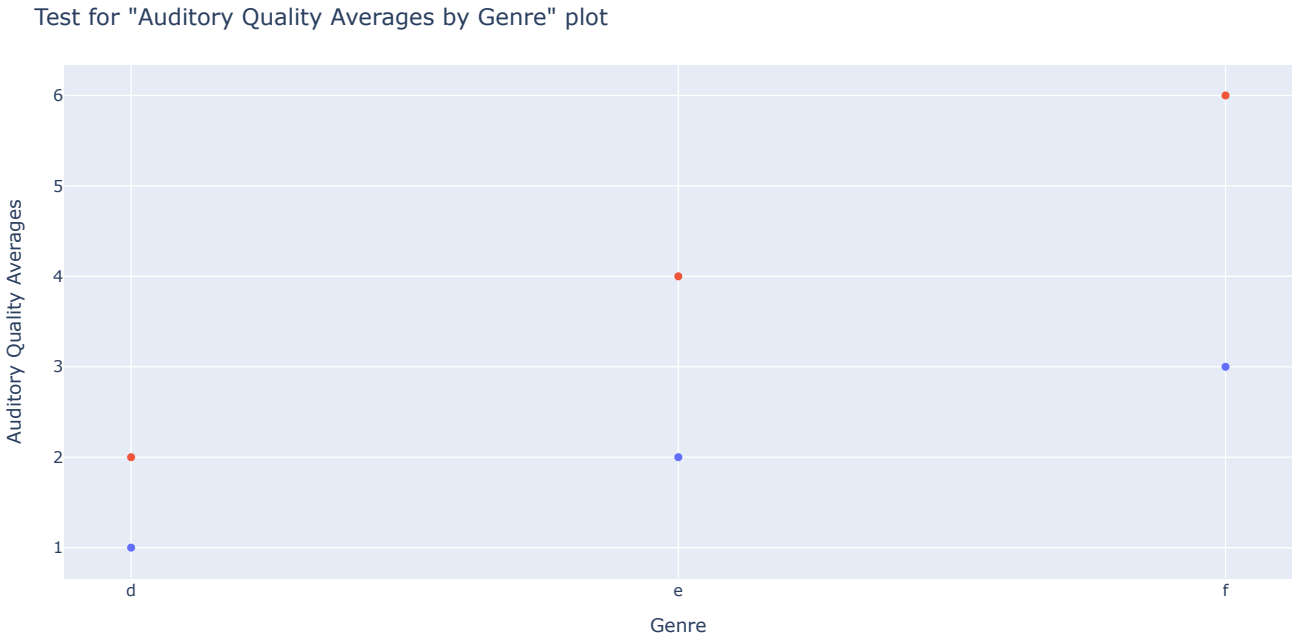


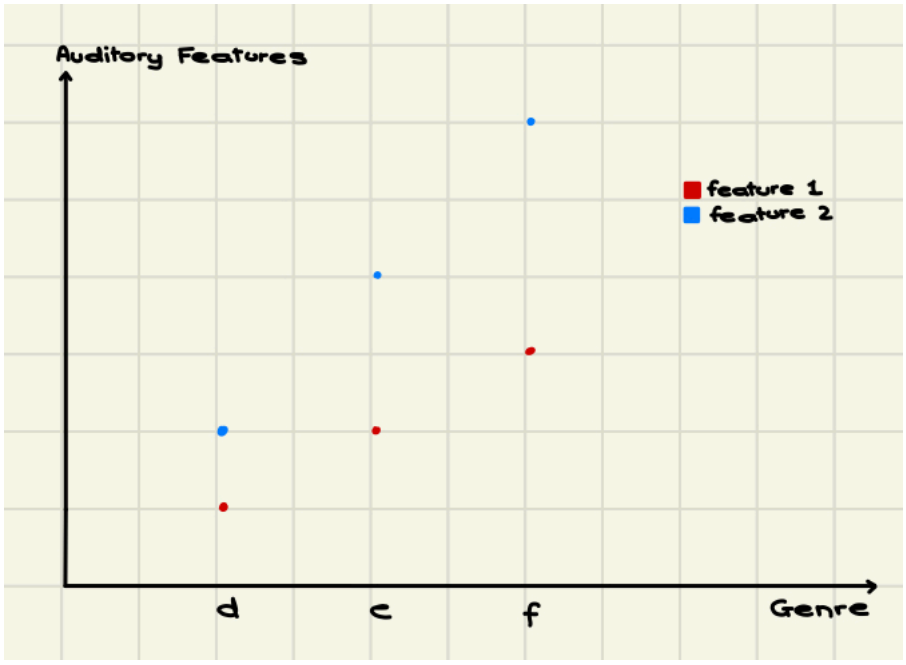

```
# Test for "Least Popular to Most Popular Genres of All Time"
testfile5 = testfile.sort_values(by='streams', ascending=True)
testfile5 = testfile5.groupby('genre')['streams'].sum().reset_index()
testfig5 = px.bar(testfile5, x='genre',
                  y=testfile5['streams'],
                  title='Test for "Least Popular to Most Popular Genres of All'
                  ' Time" plot')
testfig5.update_layout(xaxis_title='Genre', yaxis_title='Streams',
                       xaxis={'categoryorder': 'total ascending'})

testfig5
```



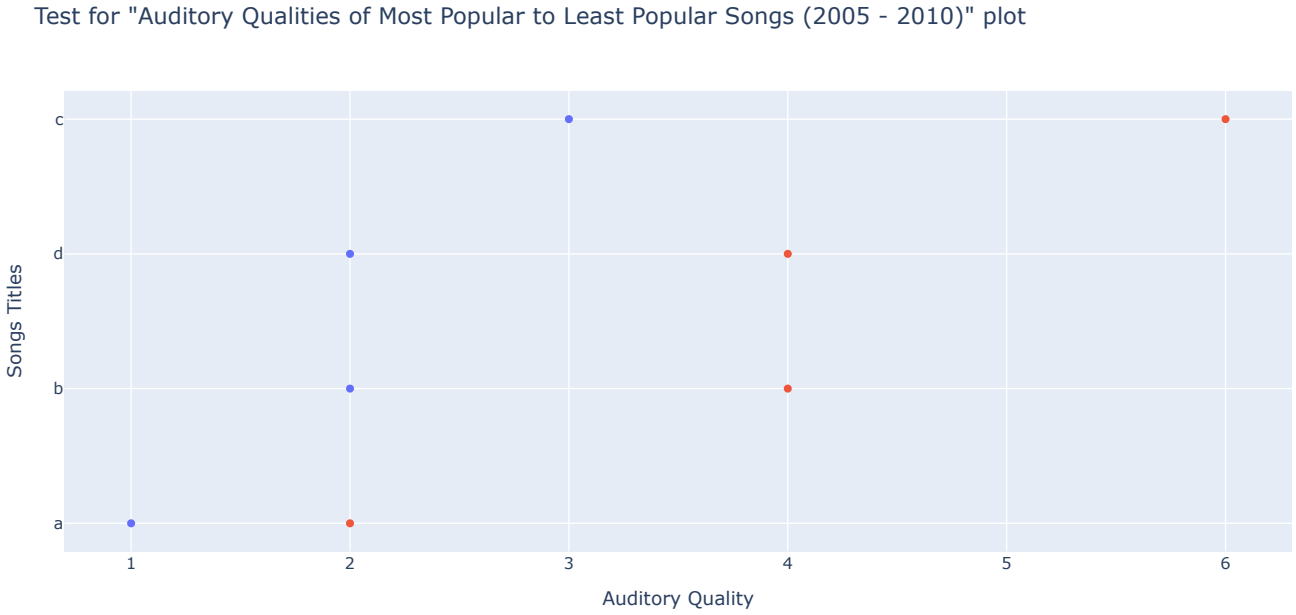


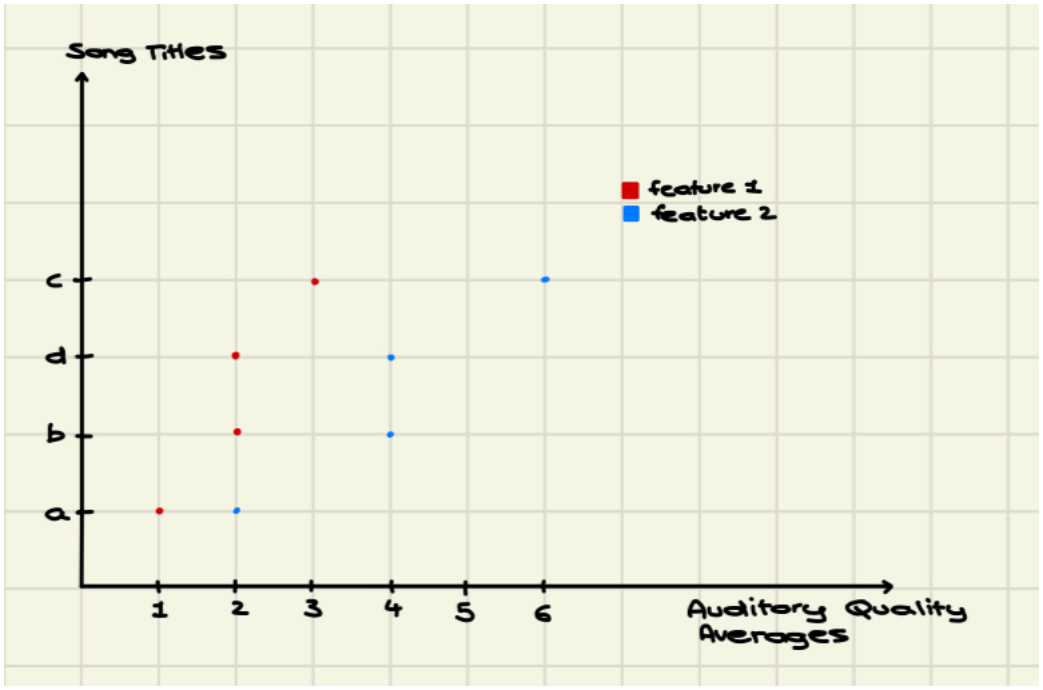

```
# Test for "Auditory Quality Averages by Genre"
testfile6 = testfile.groupby('genre')[['feature1',
                                       'feature2']].mean().reset_index()

testfig6 = px.scatter(testfile6, x="genre", y=['feature1', 'feature2'])

testfig6.update_layout(title='Test for "Auditory Quality Averages by Genre"'
                       ' plot',
                       xaxis_title='Genre',
                       yaxis_title='Auditory Quality Averages',
                       legend_title='Auditory Quality')

testfig6
```



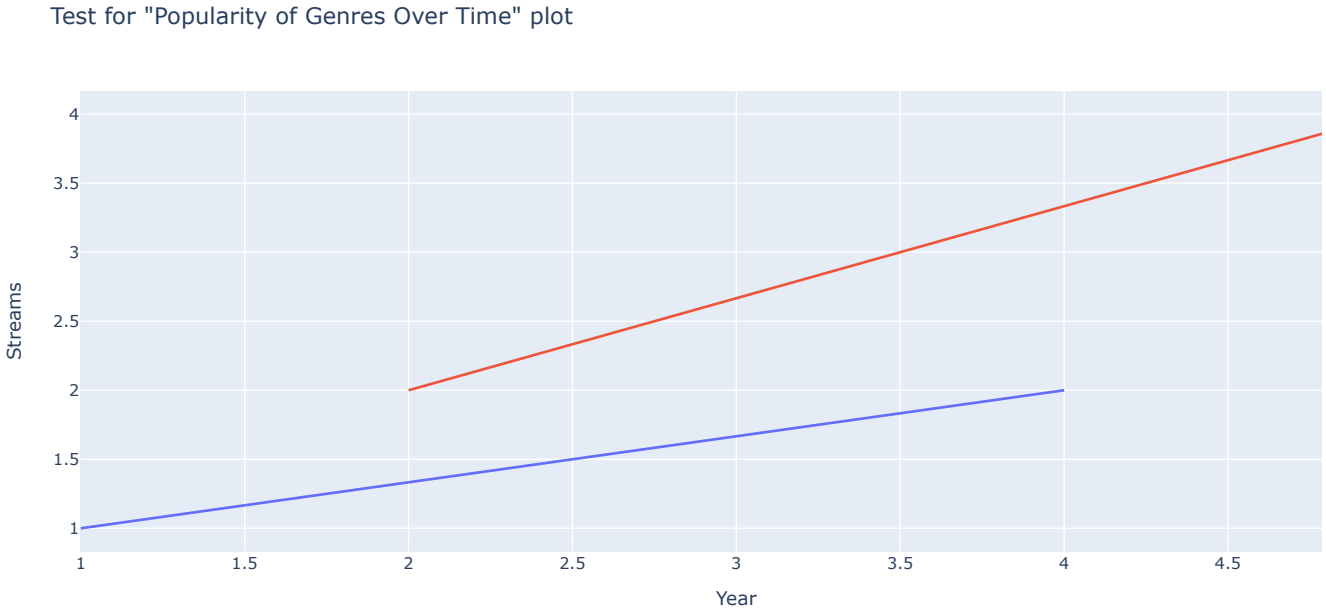


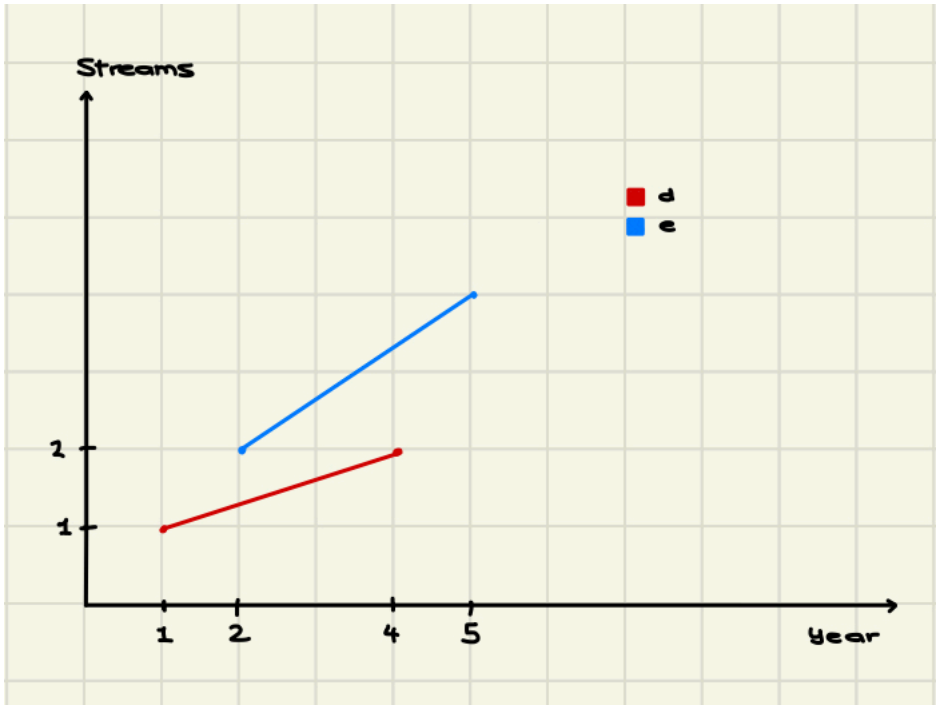

```
# Test for "Auditory Qualitites of Most Popular to Least Popular Songs"
testfile7 = testfileb.loc[(testfileb["year"] >= 1) & (testfileb["year"] <= 4)]
testfile7 = testfile7.sort_values(by='streams', ascending=True)
testfig7 = px.scatter(testfile7, y="title", x=['feature1', 'feature2'],
                      title='Test for "Auditory Qualities of Most Popular'
                      ' to Least Popular Songs (2005 - 2010)" plot')
testfig7.update_layout(xaxis_title='Auditory Quality',
                       yaxis_title='Songs Titles',
                       legend_title='Auditory Quality')

testfig7
```



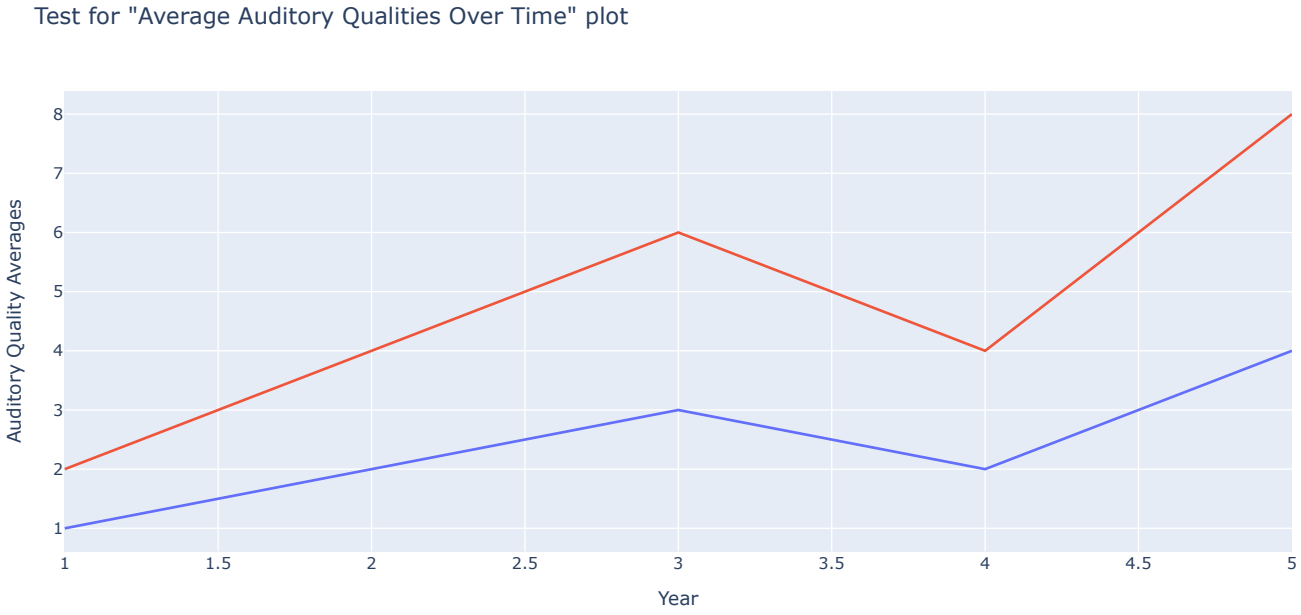


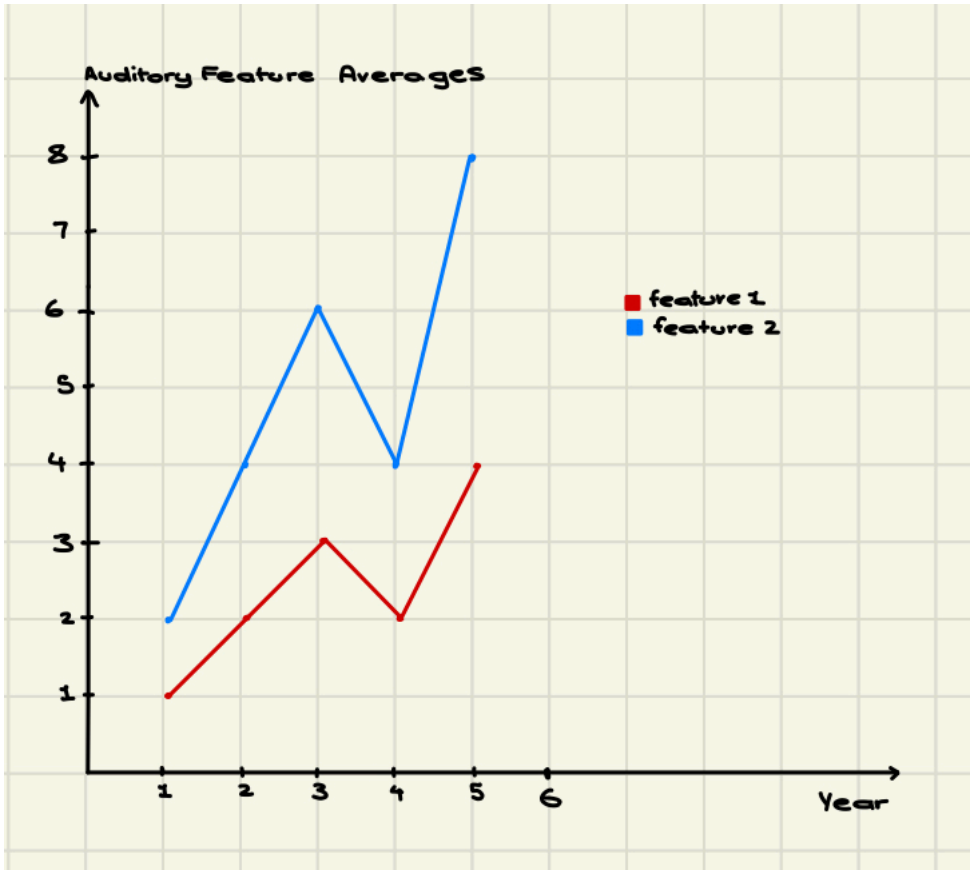

```
# Test for "Popularity of Genres Over Time"
testfile8 = testfilea[(testfilea['genre'] == 'd') |
                      (testfilea['genre'] == 'e')]
testfile8 = testfile8.groupby(['year', 'genre'])['streams'].sum().reset_index()
testfig8 = px.line(testfile8, x='year', y='streams', color='genre',
                   title='Test for "Popularity of Genres Over Time" plot')
testfig8.update_layout(xaxis_title='Year',
                       yaxis_title='Streams',
                       legend_title='Genre')

testfig8
```



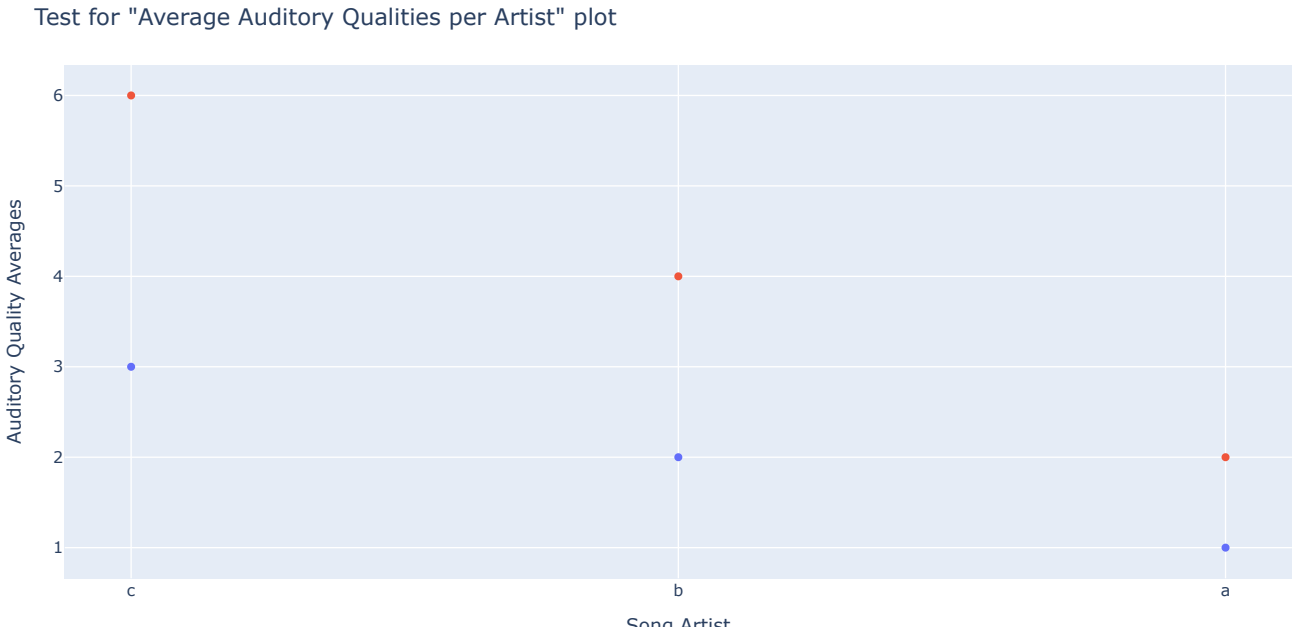


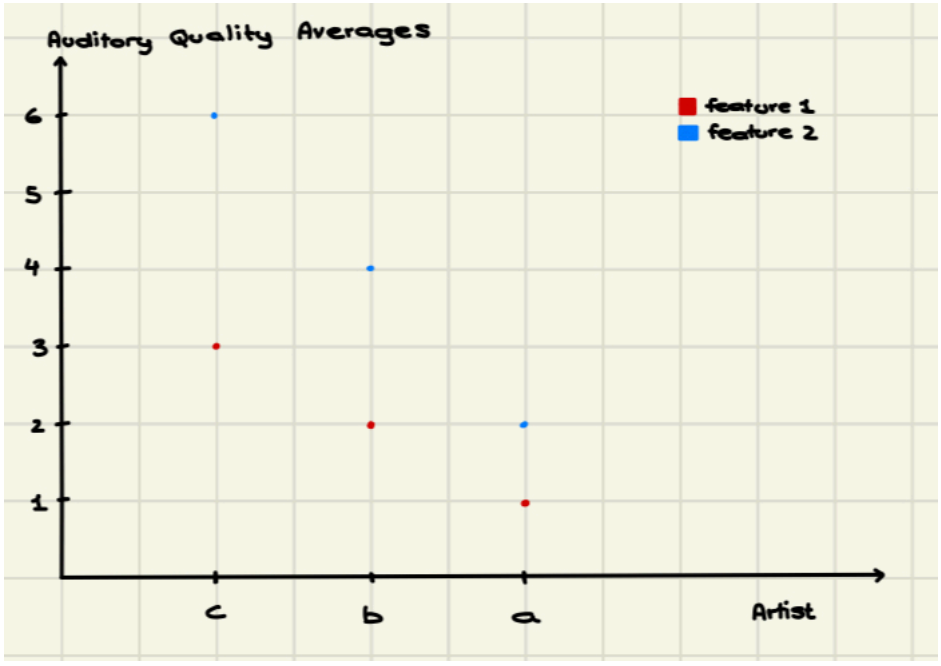

```
# Test for "Average Auditory Qualities Over Time"
testfile9 = testfilea.groupby('year')[['feature1',
                                       'feature2']].mean().reset_index()
testfig9 = px.line(testfile9, x="year", y=['feature1', 'feature2'],
                   title='Test for "Average Auditory Qualities'
                   ' Over Time" plot')
testfig9.update_layout(xaxis_title='Year',
                       yaxis_title='Auditory Quality Averages',
                       legend_title='Auditory Quality')
testfig9
```






```
# Test for "Average Auditory Qualities per Artist"
testfile10 = testfile.groupby('artist')[['feature1',
                                         'feature2',
                                         'streams']].mean().reset_index()
testfile10 = testfile10.nlargest(3, 'streams')
testfig10 = px.scatter(testfile10, x="artist", y=['feature1', 'feature2'])
testfig10.update_layout(
    title='Test for "Average Auditory Qualities per Artist" plot',
    xaxis_title='Song Artist',
    yaxis_title='Auditory Quality Averages',
    legend_title='Auditory Quality')

testfig10
```






```
# Test for "Number of Popular Songs Released Per Year in Top 10 Most Streamed
# Genres"
testfig11_list = list(testfileb['genre'])
testfile11 = testfileb[testfileb['genre'].isin(testfig11_list)]
testfile11 = testfile11.groupby(['genre',
                                 'year'])['title'].count().reset_index()
testfig11 = px.bar(testfile11, x='year',
                   y='title', color='genre',
                   labels={"title": "Number of Songs Released"},
                   title='Test for "Number of Popular Songs Released Per'
                   ' Year In Top 10 Most Streamed Genres" plot')
testfig11.update_layout(xaxis_title='Year',
```