In [1]:
```
# Import standard packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
import statsmodels.formula.api as smf
from statsmodels.formula.api import ols
from scipy import stats
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import KFold
from sklearn.metrics import mean_squared_error, r2_score
import re

%matplotlib inline
```

In [2]:
```
#Importing data
anime = pd.read_csv("data/anime_cleaned.csv")
anime
```

Out[2]:

	anime_id	title	title_english	title_japanese	title_synonyms	imag
0	11013	Inu x Boku SS	Inu X Boku Secret Service	妖狐×僕SS	Youko x Boku SS	https://myanimelist dena.com/images/ani
1	2104	Seto no Hanayome	My Bride is a Mermaid	瀬戸の花嫁	The Inland Sea Bride	https://myanimelist dena.com/images/ani
2	5262	Shugo Chara!! Doki	Shugo Chara!! Doki	しゅごキャラ！！ どきっ	Shugo Chara Ninenme, Shugo Chara! Second Year	https://myanimelist dena.com/images/ani
3	721	Princess Tutu	Princess Tutu	プリンセスチュチュ	NaN	https://myanimelist dena.com/images/ani
4	12365	Bakuman. 3rd Season	Bakuman.	バクマン。	Bakuman Season 3	https://myanimelist dena.com/images/ani
...	...	...	...	...	...	
6663	37405	Dokidoki Little Ooyasan	NaN	dokidokiりとる大家さん	NaN	https://myanimelist dena.com/images/ani
6664	37886	Wo Shi Jiang Xiaobai (2018)	I'm Joybo OVA	我是江小白 小剧场	Wo Shi Jiang Xiao Bai: Xiao Ju Chang	https://myanimelist dena.com/images/ani
6665	37255	Genki Genki Non-tan: Obake Mura Meiro	NaN	げんきげんきノンタン　おばけむらめいろ	NaN	https://myanimelist dena.com/images/ani
6666	35229	Mr. Men Little Miss	Mr. Men Little Miss	Mr. Men Little Miss / ミスターメン リトルミス	NaN	https://myanimelist dena.com/images/ani
6667	36315	Mushi Mushi Mura no Nakama-tachi: Minna Ii Tok...	NaN	むしむし村の仲間たち　みんないいとこあるんだよ	NaN	https://myanimelist dena.com/images/ani

6668 rows × 33 columns

In [3]:
```
#Checking for nulls
print(anime.isna().sum())
```

```
anime_id              0
title                 0
title_english      3230
title_japanese        5
title_synonyms     2187
image_url             2
type                  0
source                0
episodes              0
status                0
airing                0
aired_string          0
aired                 0
duration              0
rating                0
score                 0
scored_by             0
rank                356
popularity            0
members               0
favorites             0
background         5855
premiered          3702
broadcast          3688
related               0
producer           2266
licensor           3881
studio                0
genre                 4
opening_theme         0
ending_theme          0
duration_min          0
aired_from_year       0
dtype: int64
```

In [4]:

```
#Data Cleaning

#Dropping nulls from genre
anime = anime.dropna(subset=['genre'])

#duplicates banished
anime = anime.drop_duplicates()

#dropping irrelevant or incomplete data
anime = anime.drop(['title_english', 'title_japanese', 'title_synonyms', 'im
anime = anime[~anime['genre'].str.contains('hentai|yaoi|yuri', case=False)]

# drop all rows that contain a cell with the value of zero
anime = anime[(anime != 0).all(1)]

# create a new column named "watch_length" by multiplying "duration_min" and
anime['watch_length'] = anime['duration_min'] * anime['episodes']

anime
```

Out[4]:

	anime_id	title	type	source	episodes	score	scored_by	members	stu
0	11013	Inu x Boku SS	TV	Manga	12	7.63	139250	283882	Da Product
1	2104	Seto no Hanayome	TV	Manga	26	7.89	91206	204003	Gor
2	5262	Shugo Chara!! Doki	TV	Manga	51	7.55	37129	70127	Sateli
3	721	Princess Tutu	TV	Original	38	8.21	36501	93312	Hal F Ma
4	12365	Bakuman. 3rd Season	TV	Manga	25	8.67	107767	182765	J.C.St
...	...	...	...	...	...	...	...	...	
6657	37090	Hitori no Shita: The Outcast Recap	Special	Manga	1	6.04	181	1013	Haolin Animat Leag
6658	36913	Inazuma Eleven x Kaitou Gru no Tsuki Dorobou	Special	Original	4	5.87	128	560	O
6662	37894	Ling Yu 4th Season	ONA	Novel	12	7.53	17	126	Haolin Animat Leag
6665	37255	Genki Genki Non-tan: Obake Mura Meiro	OVA	Original	1	4.20	5	37	Polyg Pictu
6667	36315	Mushi Mushi Mura no Nakama-tachi: Minna Ii Tok...	OVA	Original	1	7.00	4	40	T Animat

6151 rows × 13 columns

In [5]:
```
#Importing data
#users = pd.read_csv("data/animelists_cleaned.csv")
#users
```

In [6]:
```
#Checking for correlation/multicolinearity

# Selecting columns with numerical values
numerical_columns = anime.select_dtypes(include=[np.number]).columns.tolist(

# Selecting the score column and calculating correlations with other columns
features = []
correlations = []
corr = anime[numerical_columns].corr()
for idx, correlation in corr['score'].iteritems():
    if correlation >= 0.3 and idx != 'score':
        features.append(idx)
        correlations.append(correlation)
corr_score_df = pd.DataFrame({'Features': features, 'Correlations': correlat

# Checking for multicollinearity among the selected features
multicollinear_features = []
multicollinear_corr = []
def check_multicollinearity(feature):
    for idx, correlation in corr[feature].iteritems():
        if correlation >= 0.8 and idx != feature:
            multicollinear_features.append([feature, idx])
            multicollinear_corr.append(correlation)

for feature in numerical_columns:
    check_multicollinearity(feature)
mc_df = pd.DataFrame({'Features': [f[0] for f in multicollinear_features],
                      'Multicollinear_Features': [f[1] for f in multicolline
                      'Correlations': multicollinear_corr})

# Displaying the results
print('Features with Correlations to score')
display(corr_score_df)

print('Multicollinear Features')
display(mc_df)
```

```
Features with Correlations to score
```

	Features	Correlations
0	scored_by	0.394409
1	members	0.428485

```
Multicollinear Features
```

	Features	Multicollinear_Features	Correlations
0	episodes	watch_length	0.834427
1	scored_by	members	0.987091
2	members	scored_by	0.987091
3	watch_length	episodes	0.834427

In [8]:
```
#Dropping multicolinear faetures
anime = anime.drop('scored_by', axis=1)

#Dropping non-correlated variables
anime = anime.drop(['episodes', 'aired_from_year', 'duration_min', 'watch_le
```

In [9]:
```
#Creating a list with the columns I'm using
num_cols = list(anime.drop(['anime_id', 'score'], axis=1).select_dtypes(incl
```

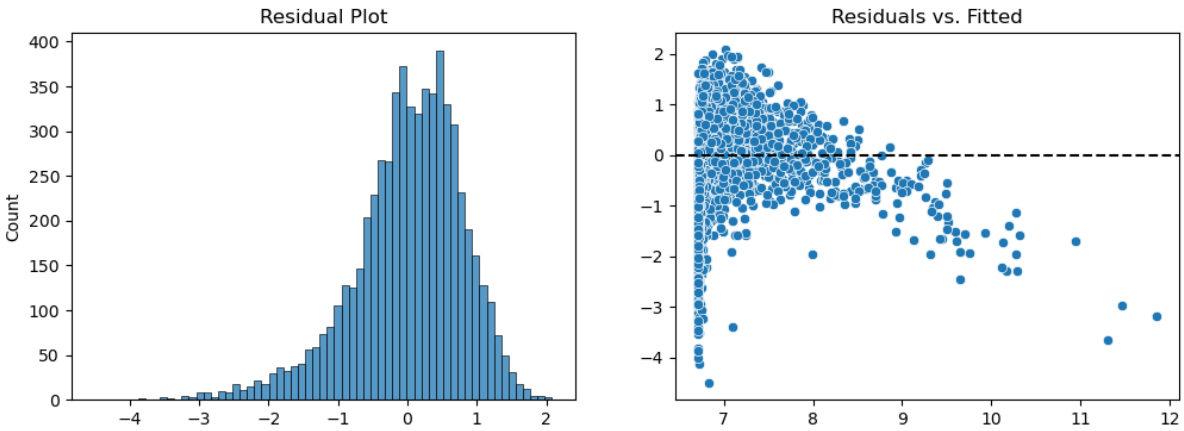
```
predictors = pd.DataFrame({'Features': num_cols})
predictors
```

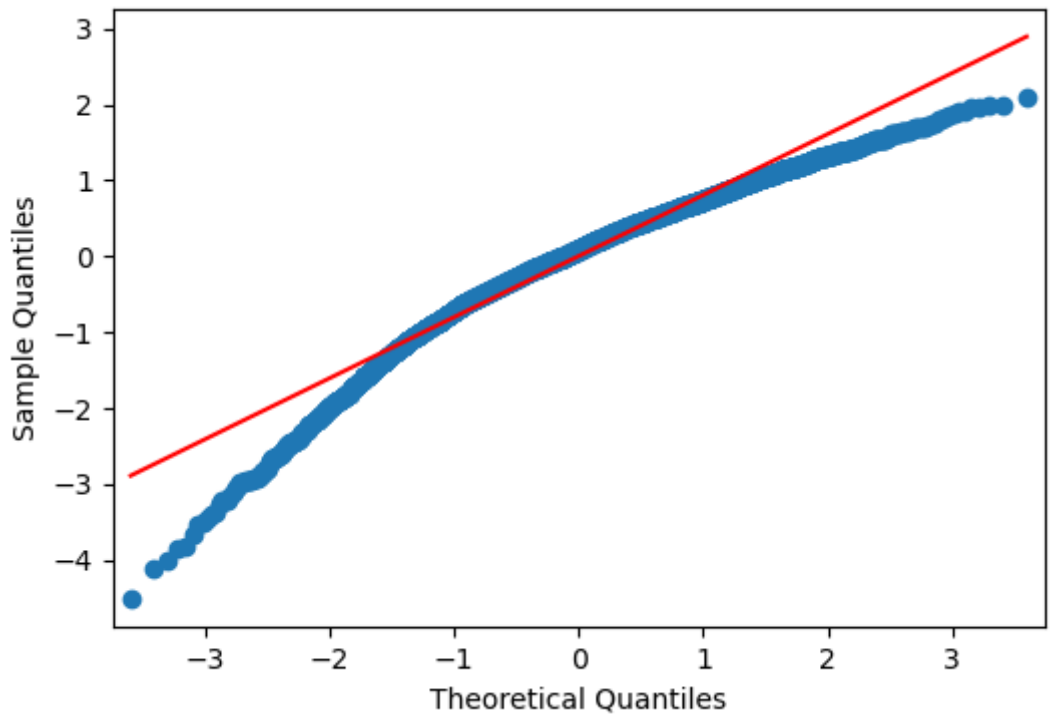
Out[9]:

	Features
0	members

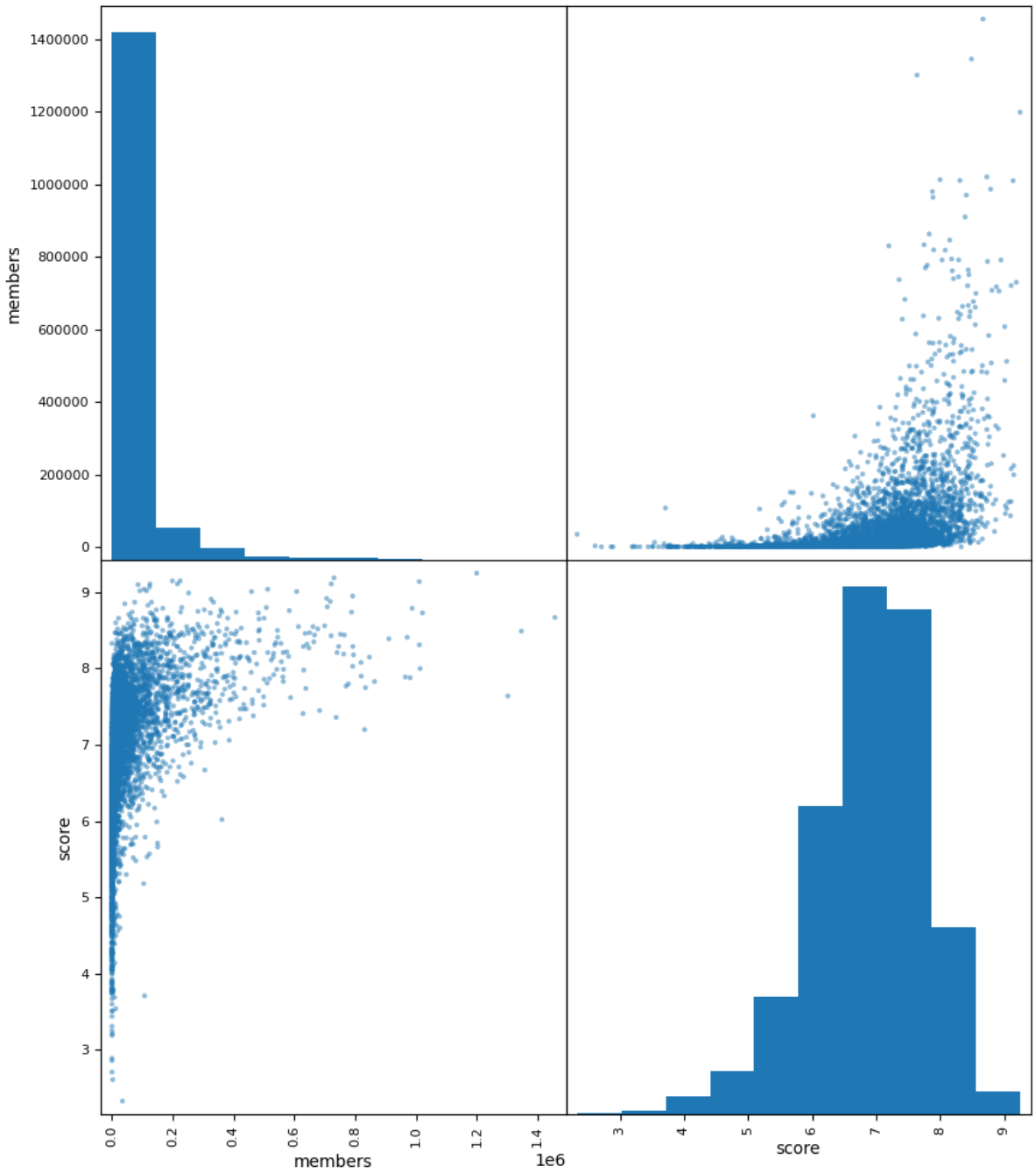
In [10]:

```
#Normality and homoscedacity check
# Fit simple linear regression model
model = smf.ols('score ~ ' + ' + '.join(predictors['Features']), data=anime)

#Residual plot
resid = model.resid
fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(12, 4))
sns.histplot(resid, ax=ax1)
ax1.set_title('Residual Plot')
#Scatter Plot
sns.scatterplot(x=model.predict(), y=resid, ax=ax2)
ax2.axhline(0, color='k', linestyle='--')
ax2.set_title('Residuals vs. Fitted')
plt.show()
#QQ plot
fig, ax = plt.subplots(figsize=(6,4))
sm.qqplot(resid, line='s', ax=ax)
plt.show()
```



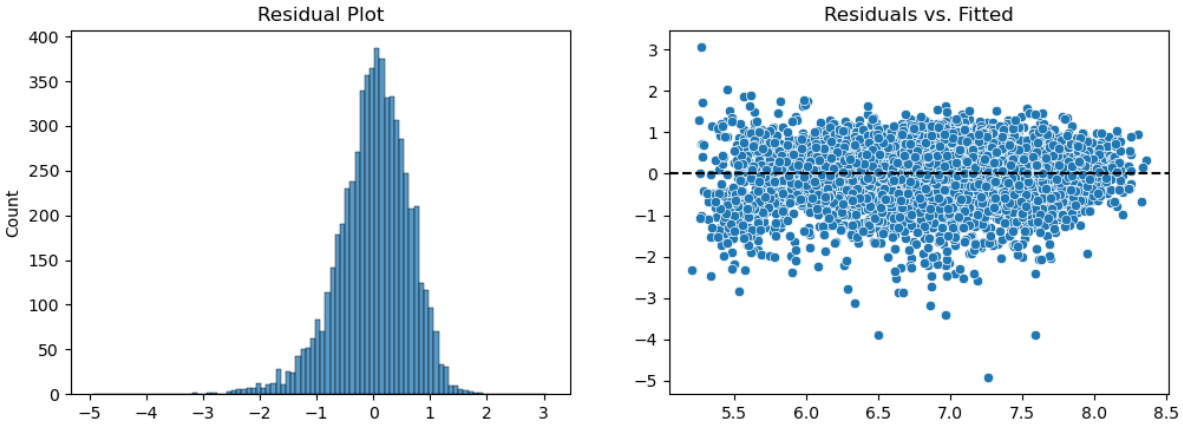


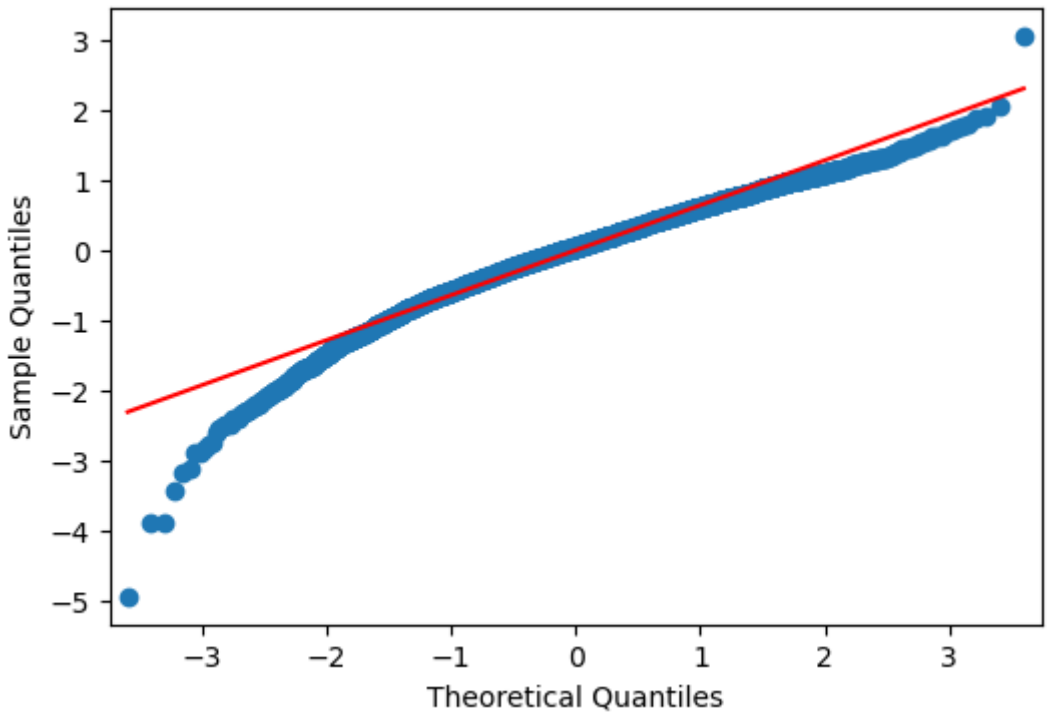
In [11]:
```
#Normal check of relevant features

x_cols = list(predictors['Features']) + ['score']
pd.plotting.scatter_matrix(anime[x_cols], figsize=(10, 12))
plt.show()
```



In [12]:
```
#Taking the log of scored_by and rechecking

anime['members'] = np.log(anime['members'])

#Normality and homoscedacity check
# Fit simple linear regression model
model = smf.ols('score ~ ' + ' + '.join(predictors['Features']), data=anime)

#Residual plot
resid = model.resid
fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(12, 4))
sns.histplot(resid, ax=ax1)
ax1.set_title('Residual Plot')
#Scatter Plot
sns.scatterplot(x=model.predict(), y=resid, ax=ax2)
```

```
ax2.axhline(0, color='k', linestyle='--')
ax2.set_title('Residuals vs. Fitted')
plt.show()
#QQ plot
fig, ax = plt.subplots(figsize=(6,4))
sm.qqplot(resid, line='s', ax=ax)
plt.show()
```





In [13]:
```
#running multiple regression on relevant features
#"relevant features" ended up being only scored_by but that's fine

outcome = 'score'
pred_sum = '+'.join(predictors["Features"])
formula = outcome + '~' + pred_sum

model = ols(formula=formula, data=anime).fit()
model.summary()
```

Out[13]:

OLS Regression Results

Dep. Variable:	score	**R-squared:**	0.482
Model:	OLS	**Adj. R-squared:**	0.482
Method:	Least Squares	**F-statistic:**	5730.
Date:	Wed, 01 Mar 2023	**Prob (F-statistic):**	0.00
Time:	20:28:15	**Log-Likelihood:**	-5993.2
No. Observations:	6151	**AIC:**	1.199e+04
Df Residuals:	6149	**BIC:**	1.200e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	4.2073	0.036	115.478	0.000	4.136	4.279
members	0.2923	0.004	75.698	0.000	0.285	0.300

Omnibus:	768.373	**Durbin-Watson:**	1.991
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1714.301
Skew:	-0.750	**Prob(JB):**	0.00
Kurtosis:	5.107	**Cond. No.**	42.5

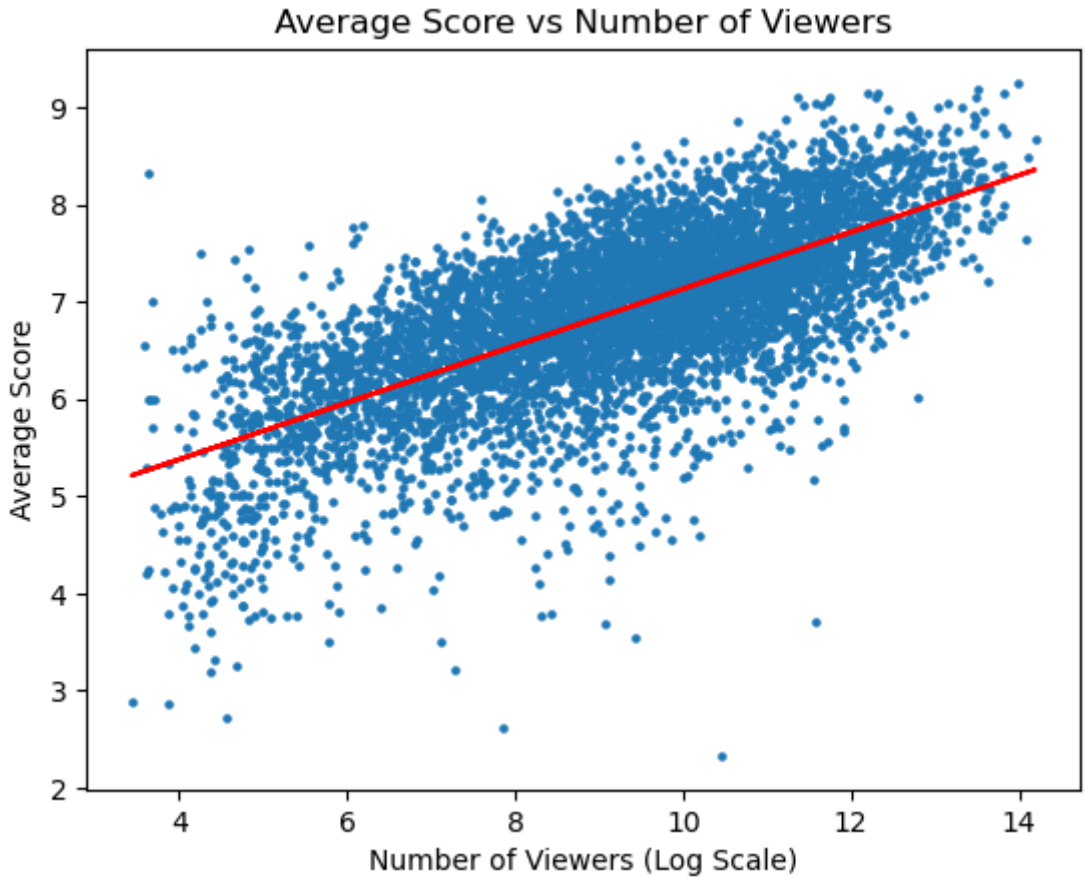
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [14]:

```
#Running linear regression analysis on scored_by and making a graph

# Extract the "scored_by" and "score" columns as X and y
X = anime["members"].values.reshape(-1, 1)
y = anime["score"].values.reshape(-1, 1)

# Initialize the linear regression model
model = LinearRegression()

# Set up the KFold cross-validation
kf = KFold(n_splits=5, shuffle=True, random_state=42)
mse_scores = []

# Loop over the folds
for train_index, test_index in kf.split(X):
    # Split the data into training and test sets
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]

    # Fit the model on the training data
    model.fit(X_train, y_train)

    # Predict on the test data and calculate mean squared error
    y_pred = model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    mse_scores.append(mse)

# Print the mean and standard deviation of the mean squared error
```

```
print("Mean MSE:", round(sum(mse_scores)/len(mse_scores), 4))
print("Std. Dev. of MSE:", round(np.std(mse_scores), 4))

# Print the model coefficients
print("Intercept:", model.intercept_)
print("Slope:", model.coef_[0])

# Plot the data points and regression line
plt.scatter(X, y, s=5)
plt.plot(X, model.predict(X), color="red")
plt.title('Average Score vs Number of Viewers')
plt.xlabel("Number of Viewers (Log Scale)")
plt.ylabel("Average Score")
plt.show()
```

```
Mean MSE: 0.4113
Std. Dev. of MSE: 0.0249
Intercept: [4.20890272]
Slope: [0.29235258]
```



In [15]:
```
#Making dummies for categorical values

#First converting the categorical columns into lists

# Define a pattern to extract words from each category
pattern = re.compile(r'[a-zA-Z0-9\- ]+')

# Modify the 'studio', 'genre', 'type', and 'source' columns using regular e
anime['studio'] = anime['studio'].apply(lambda x: [word.strip().title() for
anime['genre'] = anime['genre'].apply(lambda x: [word.strip().title() for wo
anime['type'] = anime['type'].apply(lambda x: [word.strip().title() for word
anime['source'] = anime['source'].apply(lambda x: [word.strip().title() for

# Create dummy variables for the modified columns
anime_studio_dummies = anime['studio'].apply(lambda x: '|'.join(x)).str.get_
anime_genre_dummies = anime['genre'].apply(lambda x: '|'.join(x)).str.get_du
```

```
anime_type_dummies = anime['type'].apply(lambda x: '|'.join(x)).str.get_dumm
anime_source_dummies = anime['source'].apply(lambda x: '|'.join(x)).str.get_

# Replace spaces with underscores in column names
anime_studio_dummies.columns = anime_studio_dummies.columns.str.replace(' ',
anime_genre_dummies.columns = anime_genre_dummies.columns.str.replace(' ', '
anime_type_dummies.columns = anime_type_dummies.columns.str.replace(' ', '_'
anime_source_dummies.columns = anime_source_dummies.columns.str.replace(' ',
```

In [16]:

```
#Running multiple regression analysis on genres

# Set the dependent variable 'score' and drop the first column of the datafr
y = anime['score']
X = anime_genre_dummies.iloc[:, 1:]

# Add a constant column to the predictor variables
#X = sm.add_constant(X)

# Fit the multiple regression model
model = sm.OLS(y, X).fit()

# Print the model summary
print(model.summary())
```

OLS Regression Results

Dep. Variable:	score	R-squared (uncentered):	0.874
Model:	OLS	Adj. R-squared (uncentered):	0.873
Method:	Least Squares	F-statistic:	1085.
Date:	Wed, 01 Mar 2023	Prob (F-statistic):	0.00
Time:	20:28:16	Log-Likelihood:	-14288.
No. Observations:	6151	AIC:	2.865e+04
Df Residuals:	6112	BIC:	2.892e+04
Df Model:	39		
Covariance Type:	nonrobust		

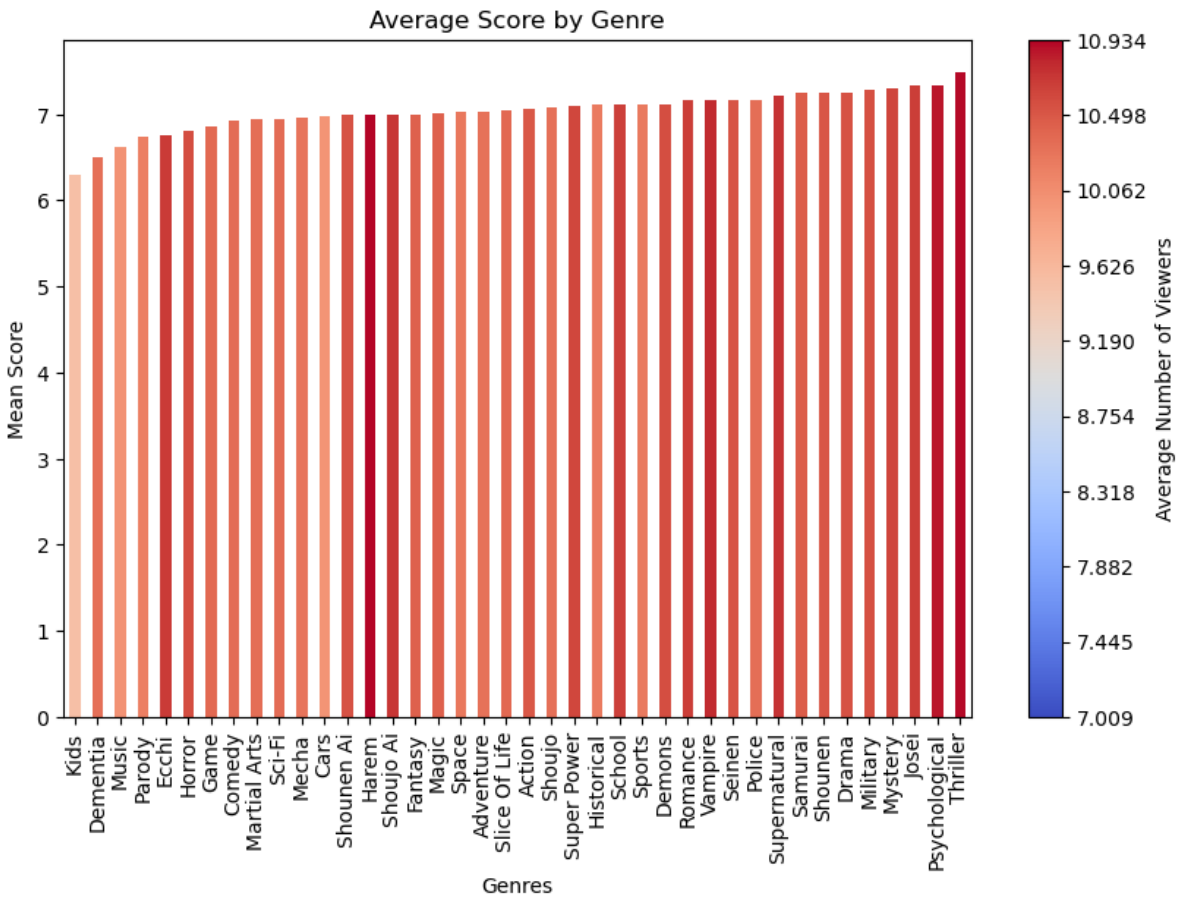
	coef	std err	t	P>|t|	[0.025	0.975]
Adventure	1.3416	0.085	15.769	0.000	1.175	1.508
Cars	1.9947	0.402	4.957	0.000	1.206	2.783
Comedy	2.6514	0.063	41.985	0.000	2.528	2.775
Dementia	1.4078	0.380	3.706	0.000	0.663	2.153
Demons	0.6186	0.187	3.314	0.001	0.253	0.984
Drama	1.6237	0.082	19.822	0.000	1.463	1.784
Ecchi	0.9517	0.128	7.450	0.000	0.701	1.202
Fantasy	1.9708	0.084	23.497	0.000	1.806	2.135
Game	2.7405	0.197	13.929	0.000	2.355	3.126
Harem	0.2110	0.175	1.206	0.228	-0.132	0.554
Historical	2.1105	0.138	15.286	0.000	1.840	2.381
Horror	1.0890	0.185	5.882	0.000	0.726	1.452
Josei	1.1203	0.319	3.509	0.000	0.494	1.746
Kids	2.0523	0.136	15.063	0.000	1.785	2.319
Magic	1.2743	0.115	11.037	0.000	1.048	1.501
Martial_Arts	1.3606	0.193	7.048	0.000	0.982	1.739
Mecha	1.5174	0.130	11.668	0.000	1.262	1.772
Military	1.7676	0.160	11.032	0.000	1.453	2.082
Music	3.3446	0.142	23.474	0.000	3.065	3.624
Mystery	2.0872	0.133	15.728	0.000	1.827	

```
2.347
Parody             1.4971      0.165      9.091      0.000       1.174
1.820
Police             1.3963      0.213      6.559      0.000       0.979
1.814
Psychological      1.1946      0.195      6.116      0.000       0.812
1.578
Romance            1.0733      0.092     11.644      0.000       0.893
1.254
Samurai            1.3291      0.272      4.883      0.000       0.796
1.863
School             1.4319      0.094     15.284      0.000       1.248
1.616
Sci-Fi             2.1998      0.090     24.558      0.000       2.024
2.375
Seinen             1.6910      0.124     13.664      0.000       1.448
1.934
Shoujo             1.5685      0.132     11.911      0.000       1.310
1.827
Shoujo_Ai          1.5640      0.342      4.571      0.000       0.893
2.235
Shounen            1.1208      0.090     12.505      0.000       0.945
1.297
Shounen_Ai         1.4869      0.343      4.338      0.000       0.815
2.159
Slice_Of_Life      2.4245      0.094     25.919      0.000       2.241
2.608
Space              0.9606      0.183      5.260      0.000       0.603
1.319
Sports             3.0324      0.151     20.109      0.000       2.737
3.328
Super_Power        1.6392      0.140     11.686      0.000       1.364
1.914
Supernatural       1.7249      0.103     16.795      0.000       1.524
1.926
Thriller           2.1912      0.294      7.449      0.000       1.615
2.768
Vampire            1.0463      0.264      3.965      0.000       0.529
1.564
==============================================================================
==
Omnibus:                      396.194   Durbin-Watson:                   1.7
02
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              587.5
29
Skew:                          -0.543   Prob(JB):                     2.63e-1
28
Kurtosis:                       4.056   Cond. No.                         1
1.4
==============================================================================
==

Notes:
[1] R² is computed without centering (uncentered) since the model does not c
ontain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is corre
ctly specified.
```

In [17]:
```
#Making a bar chart of the average score vs genres, coloured by average numb

# Explode the genres column to create a new row for each genre in the list
anime_exploded = anime.explode('genre')

# Group the data by genre and calculate the mean score and scored by for eac
```

```
mean_scores_by_genre = anime_exploded.groupby('genre')['score', 'members'].m

# Sort the mean scores in ascending order
mean_scores_by_genre = mean_scores_by_genre.sort_values('score')

# Define a colormap
cmap = plt.cm.get_cmap('coolwarm')

# Create a bar plot of the mean scores
ax = mean_scores_by_genre['score'].plot(kind='bar', figsize=(10, 6), color=c

# Add labels and a title to the plot
plt.xlabel('Genres')
plt.ylabel('Mean Score')
plt.title('Average Score by Genre')

# Add a colorbar to the plot
smplot = plt.cm.ScalarMappable(cmap=cmap, norm=plt.Normalize(vmin=mean_score
smplot.set_array([])  # Remove colorbar's axes
cbar = plt.colorbar(smplot, ticks=np.linspace(mean_scores_by_genre['members'
cbar.ax.set_ylabel('Average Number of Viewers')

plt.show()
```

```
/var/folders/3l/csf1hx2d4x9306d48st_8n700000gn/T/ipykernel_50858/2730547821.
py:7: FutureWarning: Indexing with multiple keys (implicitly converted to a
tuple of keys) will be deprecated, use a list instead.
  mean_scores_by_genre = anime_exploded.groupby('genre')['score', 'member
s'].mean()
```



In [18]:
```
#Repeating for anime sources

# Set the dependent variable 'score' and drop the first column of the datafr
y = anime['score']
X = anime_source_dummies.iloc[:, 1:]
```

```
# Add a constant column to the predictor variables
#X = sm.add_constant(X)

# Fit the multiple regression model
model = sm.OLS(y, X).fit()

# Print the model summary
print(model.summary())
```

OLS Regression Results

Dep. Variable:	score	R-squared (uncentered):	0.955
Model:	OLS	Adj. R-squared (uncentered):	0.955
Method:	Least Squares	F-statistic:	9376.
Date:	Wed, 01 Mar 2023	Prob (F-statistic):	0.00
Time:	20:28:16	Log-Likelihood:	-11095.
No. Observations:	6151	AIC:	2.222e+04
Df Residuals:	6137	BIC:	2.231e+04
Df Model:	14		
Covariance Type:	nonrobust		

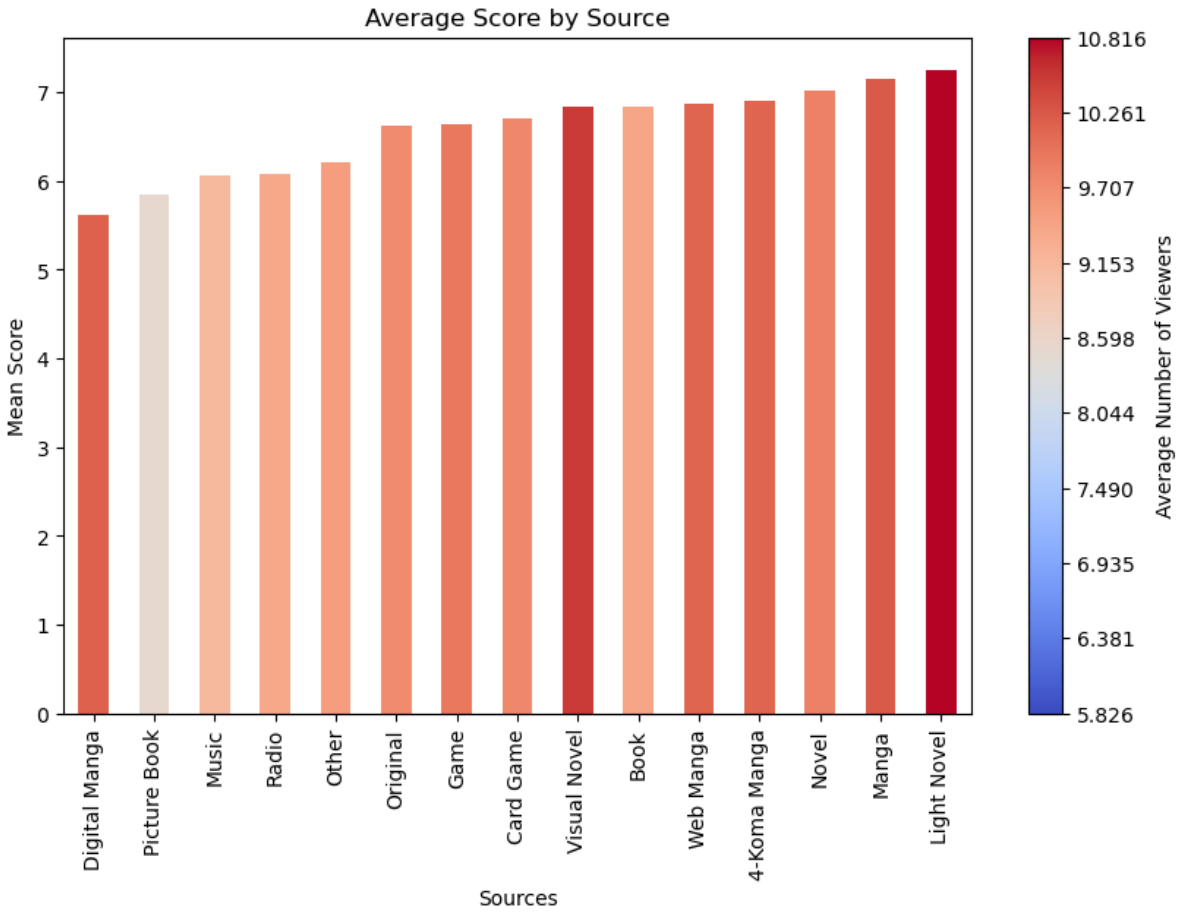
	coef	std err	t	P>|t|	[0.025	0.975]
Book	6.8389	0.222	30.839	0.000	6.404	7.274
Card_Game	6.7133	0.219	30.615	0.000	6.283	7.143
Digital_Manga	5.6100	0.556	10.090	0.000	4.520	6.700
Game	6.6424	0.073	91.434	0.000	6.500	6.785
Light_Novel	7.2538	0.065	110.815	0.000	7.125	7.382
Manga	7.1504	0.030	236.243	0.000	7.091	7.210
Music	6.0696	0.198	30.601	0.000	5.681	6.458
Novel	7.0103	0.089	79.173	0.000	6.837	7.184
Original	6.6155	0.036	184.663	0.000	6.545	6.686
Other	6.2142	0.116	53.769	0.000	5.988	6.441
Picture_Book	5.8434	0.222	26.350	0.000	5.409	6.278
Radio	6.0800	0.658	9.242	0.000	4.790	7.370
Visual_Novel	6.8354	0.092	74.494	0.000	6.656	7.015
Web_Manga	6.8731	0.144	47.878	0.000	6.592	7.155

Omnibus:	3852.210	Durbin-Watson:	1.951
Prob(Omnibus):	0.000	Jarque-Bera (JB):	46998.652
Skew:	2.851	Prob(JB):	0.00
Kurtosis:	15.283	Cond. No.	21.7

```
==

Notes:
[1] R² is computed without centering (uncentered) since the model does not c
ontain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is corre
ctly specified.
```

In [19]:
```
# Explode the sources column to create a new row for each source in the list
anime_exploded = anime.explode('source')

# Group the data by source and calculate the mean score and scored by for ea
mean_scores_by_source = anime_exploded.groupby('source')['score', 'members']

# Sort the mean scores in ascending order
mean_scores_by_source = mean_scores_by_source.sort_values('score')

# Define a colormap
cmap = plt.cm.get_cmap('coolwarm')

# Create a bar plot of the mean scores
ax = mean_scores_by_source['score'].plot(kind='bar', figsize=(10, 6), color=

# Add labels and a title to the plot
plt.xlabel('Sources')
plt.ylabel('Mean Score')
plt.title('Average Score by Source')

# Add a colorbar to the plot
smplot = plt.cm.ScalarMappable(cmap=cmap, norm=plt.Normalize(vmin=mean_score
smplot.set_array([])  # Remove colorbar's axes
cbar = plt.colorbar(smplot, ticks=np.linspace(mean_scores_by_source['members
cbar.ax.set_ylabel('Average Number of Viewers')

plt.show()
```

```
/var/folders/3l/csf1hx2d4x9306d48st_8n700000gn/T/ipykernel_50858/1681968551.
py:5: FutureWarning: Indexing with multiple keys (implicitly converted to a
tuple of keys) will be deprecated, use a list instead.
  mean_scores_by_source = anime_exploded.groupby('source')['score', 'member
s'].mean()
```



In [20]:
```
#Repeating for anime types

# Set the dependent variable 'score' and drop the first column of the datafr
y = anime['score']
X = anime_type_dummies.iloc[:, 1:]

# Add a constant column to the predictor variables
#X = sm.add_constant(X)

# Fit the multiple regression model
model = sm.OLS(y, X).fit()

# Print the model summary
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
===========
Dep. Variable:                   score   R-squared (uncentered):
0.834
Model:                             OLS   Adj. R-squared (uncentered):
0.834
Method:                  Least Squares   F-statistic:
6162.
Date:                 Wed, 01 Mar 2023   Prob (F-statistic):
0.00
Time:                         20:28:16   Log-Likelihood:
-15138.
No. Observations:                6151   AIC:
3.029e+04
Df Residuals:                    6146   BIC:
3.032e+04
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
==
                 coef    std err          t      P>|t|      [0.025      0.97
5]
------------------------------------------------------------------------------
--
Music          5.9858      0.288     20.785      0.000       5.421       6.5
50
Ona            6.1242      0.148     41.420      0.000       5.834       6.4
14
Ova            6.7416      0.090     74.824      0.000       6.565       6.9
18
Special        6.7381      0.094     71.742      0.000       6.554       6.9
22
Tv             7.0475      0.053    133.852      0.000       6.944       7.1
51
==============================================================================
==
Omnibus:                     1752.592   Durbin-Watson:                   1.7
04
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3747.5
80
Skew:                           1.708   Prob(JB):                         0.
00
Kurtosis:                       4.718   Cond. No.                         5.
47
==============================================================================
==

Notes:
[1] R² is computed without centering (uncentered) since the model does not c
ontain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is corre
ctly specified.
```

In [21]:
```
# Explode the types column to create a new row for each type in the list
anime_exploded = anime.explode('type')

# Group the data by type and calculate the mean score and scored by for each
mean_scores_by_type = anime_exploded.groupby('type')['score', 'members'].mea

# Sort the mean scores in ascending order
mean_scores_by_type = mean_scores_by_type.sort_values('score')

# Define a colormap
```

```
cmap = plt.cm.get_cmap('coolwarm')

# Create a bar plot of the mean scores
ax = mean_scores_by_type['score'].plot(kind='bar', figsize=(10, 6), color=cm

# Add labels and a title to the plot
plt.xlabel('Types')
plt.ylabel('Mean Score')
plt.title('Average Score by Type')

# Add a colorbar to the plot
smplot = plt.cm.ScalarMappable(cmap=cmap, norm=plt.Normalize(vmin=mean_score
smplot.set_array([])  # Remove colorbar's axes
cbar = plt.colorbar(smplot, ticks=np.linspace(mean_scores_by_type['members']
cbar.ax.set_ylabel('Average Number of Viewers')

plt.show()
```

```
/var/folders/3l/csf1hx2d4x9306d48st_8n700000gn/T/ipykernel_50858/14216106.p
y:5: FutureWarning: Indexing with multiple keys (implicitly converted to a t
uple of keys) will be deprecated, use a list instead.
  mean_scores_by_type = anime_exploded.groupby('type')['score', 'members'].m
ean()
```