Daniel Ferreira dos Santos junior 2019233

HDIP Data Analytics - Data Visualization Techniques - David McQuaid

In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
df=pd.read_csv("board_games.csv")
```

In [3]:
```
df.head()
```

Out[3]:

	game_id	description	image	max_players	max_playtime	min_age	mi
0	1	Die Macher is a game about seven sequential po...	//cf.geekdo-images.com/images/pic159509.jpg	5	240	14	
1	2	Dragonmaster is a trick-taking card game based...	//cf.geekdo-images.com/images/pic184174.jpg	4	30	12	
2	3	Part of the Knizia tile-laying trilogy, Samura...	//cf.geekdo-images.com/images/pic3211873.jpg	4	60	10	
3	4	When you see the triangular box and the luxuri...	//cf.geekdo-images.com/images/pic285299.jpg	4	60	12	
4	5	In Acquire, each player strategically invests ...	//cf.geekdo-images.com/images/pic342163.jpg	6	90	12	

5 rows × 22 columns

In [4]:
```
df.tail()
```

Out[4]:

	game_id	description	image	max_players	max_playti
10527	214996	Description from the publisher:

Silve... | //cf.geekdo-images.com/images/pic3093082.png | 2 | |
10528	215437	Codex: Card-Time Strategy is a customizable, n...	//cf.geekdo-images.com/images/pic3290122.jpg	5	
10529	215471	Time to walk about town and take some pictures...	//cf.geekdo-images.com/images/pic3290975.png	4	
10530	216201	The race is on for the robots of the Robo Rall...	//cf.geekdo-images.com/images/pic3374227.jpg	6	
10531	216725	The deluxe edition comes in a double tall box ...	//cf.geekdo-images.com/images/pic3308211.jpg	5	

5 rows × 22 columns

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10532 entries, 0 to 10531
Data columns (total 22 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   game_id         10532 non-null  int64  
 1   description     10532 non-null  object 
 2   image           10531 non-null  object 
 3   max_players     10532 non-null  int64  
 4   max_playtime    10532 non-null  int64  
 5   min_age         10532 non-null  int64  
 6   min_players     10532 non-null  int64  
 7   min_playtime    10532 non-null  int64  
 8   name            10532 non-null  object 
 9   playing_time    10532 non-null  int64  
 10  thumbnail       10531 non-null  object 
 11  year_published  10532 non-null  int64  
 12  artist          7759 non-null   object 
 13  category        10438 non-null  object 
 14  compilation     410 non-null    object 
 15  designer        10406 non-null  object 
 16  expansion       2752 non-null   object 
 17  family          7724 non-null   object 
 18  mechanic        9582 non-null   object 
 19  publisher       10529 non-null  object 
 20  average_rating  10532 non-null  float64
 21  users_rated     10532 non-null  int64  
dtypes: float64(1), int64(9), object(12)
memory usage: 1.8+ MB
```

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

Out[6]:
```
(10532, 22)
```

In [7]:
```
df.describe()
```

Out[7]:

	game_id	max_players	max_playtime	min_age	min_players	min_playtime	playing
count	10532.000000	10532.000000	10532.000000	10532.000000	10532.000000	10532.000000	10532.0
mean	62059.203095	5.657330	91.341436	9.714964	2.070547	80.882738	91.3
std	66223.716828	18.884403	659.754400	3.451226	0.664394	637.873893	659.7
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
25%	5444.500000	4.000000	30.000000	8.000000	2.000000	25.000000	30.0
50%	28822.500000	4.000000	45.000000	10.000000	2.000000	45.000000	45.0
75%	126409.500000	6.000000	90.000000	12.000000	2.000000	90.000000	90.0
max	216725.000000	999.000000	60000.000000	42.000000	9.000000	60000.000000	60000.0

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
game_id               0
description           0
image                 1
max_players           0
max_playtime          0
min_age               0
min_players           0
min_playtime          0
name                  0
playing_time          0
thumbnail             1
year_published        0
artist             2773
category             94
compilation       10122
designer            126
expansion          7780
family             2808
mechanic            950
publisher             3
average_rating        0
users_rated           0
dtype: int64
```

In [9]:
```
df.columns
```

Out[9]:
```
Index(['game_id', 'description', 'image', 'max_players', 'max_playtime',
       'min_age', 'min_players', 'min_playtime', 'name', 'playing_time',
       'thumbnail', 'year_published', 'artist', 'category', 'compilation',
       'designer', 'expansion', 'family', 'mechanic', 'publisher',
       'average_rating', 'users_rated'],
      dtype='object')
```

Exploring the Top 5 average rated games.

In [10]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [11]:
```
avg_rating = df[["name","average_rating"]].sort_values(by="average_rating", ascending=
```

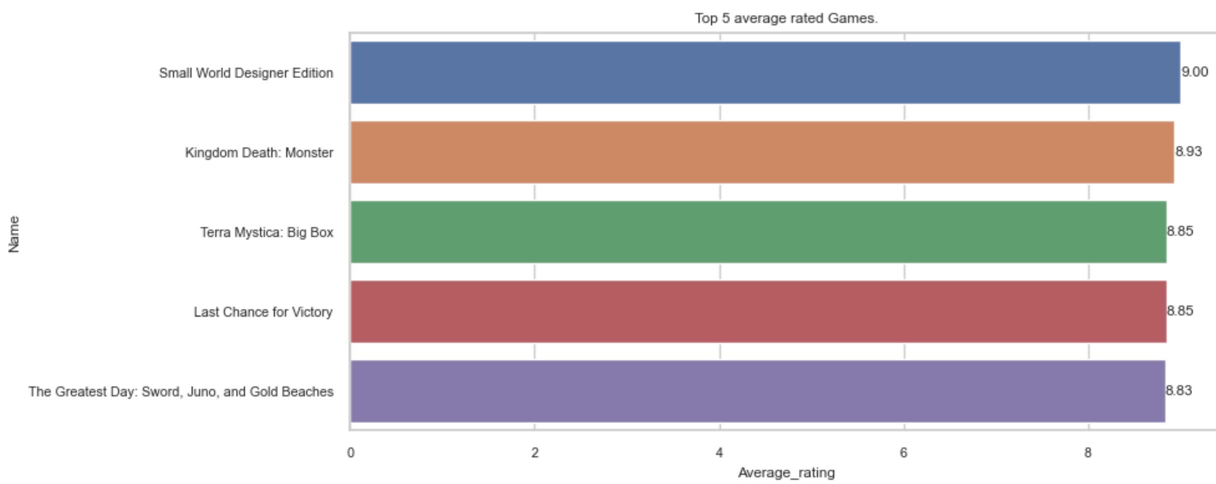
In [12]:
```
avg_rating
```

Out[12]:

	name	average_rating
8348	Small World Designer Edition	9.00392
6392	Kingdom Death: Monster	8.93184
9964	Terra Mystica: Big Box	8.84862
8526	Last Chance for Victory	8.84603
9675	The Greatest Day: Sword, Juno, and Gold Beaches	8.83081

In [13]:
```
#seabornstyle
sns.set(style="whitegrid", context= "notebook", font_scale=0.7)
```

In [14]:
```
#barplot
plt.figure(figsize=(10,4))
ax = sns.barplot(x="average_rating", y= "name", data=avg_rating, orient="h")
#labeling
plt.xlabel("Average_rating")
plt.ylabel("Name")
plt.title("Top 5 average rated Games.");

#adding more details to the graphics as seen in class week 4
for p in ax.patches:
    ax.annotate(f'{p.get_width():.2f}', (p.get_width(),p.get_y()+ p.get_height() / 2),
    plt.tight_layout()
    #plt.show()
```



Above graphic we can see that the Game "Small world Desinger edition" is the higher average rated game with a beautifull 9pts. Others games in top 5 are also well ranked example second and third with 8.83. A bar horizontal blot was to show clearly apresentation how the ratings are distributed with all the diferents games on ranking, that way been able to show the values on that datset.

Correlation between the "user_rated" and "max_playtime"?

could that impact in the final results?

In [15]:
```
df_correlation = df[["users_rated", "max_playtime"]]
```

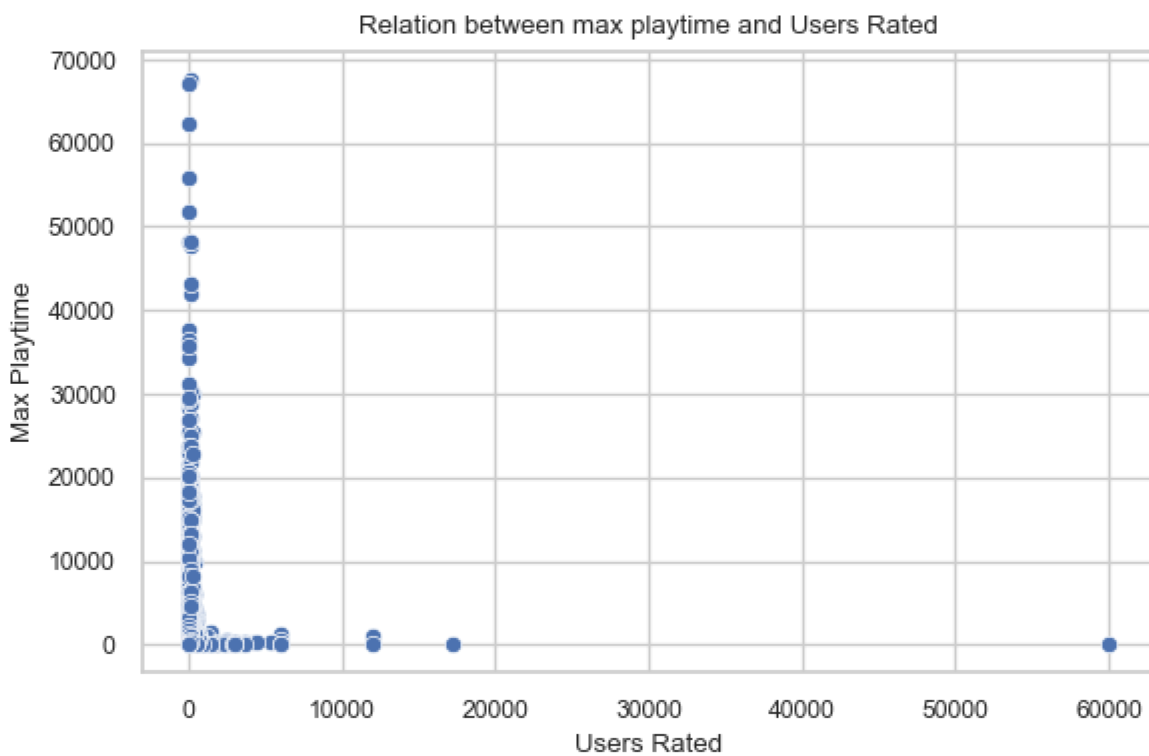
In [16]:
```
df_correlation.head()
```

Out[16]:

	users_rated	max_playtime
0	4498	240
1	478	30
2	12019	60
3	314	60
4	15195	90

In [17]:

```
#same style from beforehand
sns.set(style="whitegrid", context="notebook", font_scale=0.8)
plt.figure(figsize= (6, 4))
sns.scatterplot(x = 'max_playtime', y='users_rated',data = df_correlation);
#labeling
#labeling
plt.xlabel("Users Rated")
plt.ylabel("Max Playtime")
plt.title("Relation between max playtime and Users Rated");

#adding more details to the graphics as seen in class week 4
for p in ax.patches:
    ax.annotate(f'{p.get_width():.2f}', (p.get_width(),p.get_y()+ p.get_height() / 2),
    plt.tight_layout()
    #plt.show()
```



performing a Scatter Plot we were abble to see the correlation between the two variables. Using that tool we can clearly see a vertical linear relation, mainly where max playtime information is showed, scatterplot give usa powerfull instrumment to visually see individuals points, allowing us for expploration of underlying patterns with the dataset, reconizing the inportance of precision in our analysis we conducted a statical test fro correlation. the test was between the numbe of users whose rated the game and game maximum time played, proving a quantitative examination.

In [18]:
```
from scipy.stats import pearsonr
```

In [19]:
```
# Calculate Pearson's correlation coefficient and p-value
correlation_coefficient, p_value = pearsonr(df_correlation['users_rated'], df_correlat

# Create a DataFrame
data = {
    'Variable 1': ['users_rated'],
    'Variable 2': ['max_playtime'],
    'Pearson Correlation Coefficient': [correlation_coefficient],  # Update the columr
    'P-Value': [p_value]
}

df_correlation = pd.DataFrame(data)

# Display the DataFrame
df_correlation
```

Out[19]:

	Variable 1	Variable 2	Pearson Correlation Coefficient	P-Value
0	users_rated	max_playtime	-0.004342	0.655949

The relation between the numbers of users that rated a specific game and the maximum playtime for a specific game

in that case was chosen the pearson correlation coefficient to quantitativelyexamine the relationship between two continues variables: the numbers of users who evaluated a specific game and the maximum playtime allowed for that game. The choice of pearson correlation was motivated by the continuos nature of these variables and the observed vertical linear relation seen in the graphic before, sugesting a potential linear association between them.

To evaluate this association was conducted a hypothesis test with a significance level set as 0.05, was formulated the null and alternative hypothesis as:

H0:The correlation between the maximum playtime of a game and the number of users who rated different from 0 H1:The correlation between the maximum playtime of a game and the number of users who rated not different from 0

Reading the table above we got p-value of 0.655949 which exceeded our chosen siginificance level of0.05, we do not have anough points to reject the null hypothesis,Was conclued the correlation between them is not siginificant, the test shows within the scope of this test and at 0.05 siginificance level, the numbers of users who evaluiated a game is not siginificantly correlated with the maximum playtime os the game

What is the distribution of game categories?

In [20]:
```
value_counts_df = df['category'].value_counts().reset_index()
value_counts_df.columns = ['category', 'count']
value_counts_df = value_counts_df.sort_values(by='count', ascending=False)
total_count = value_counts_df['count'].sum()
value_counts_df['Percentage'] = (value_counts_df['count'] / total_count) * 100
```

In [21]:
```
value_counts_df.head(10)
```

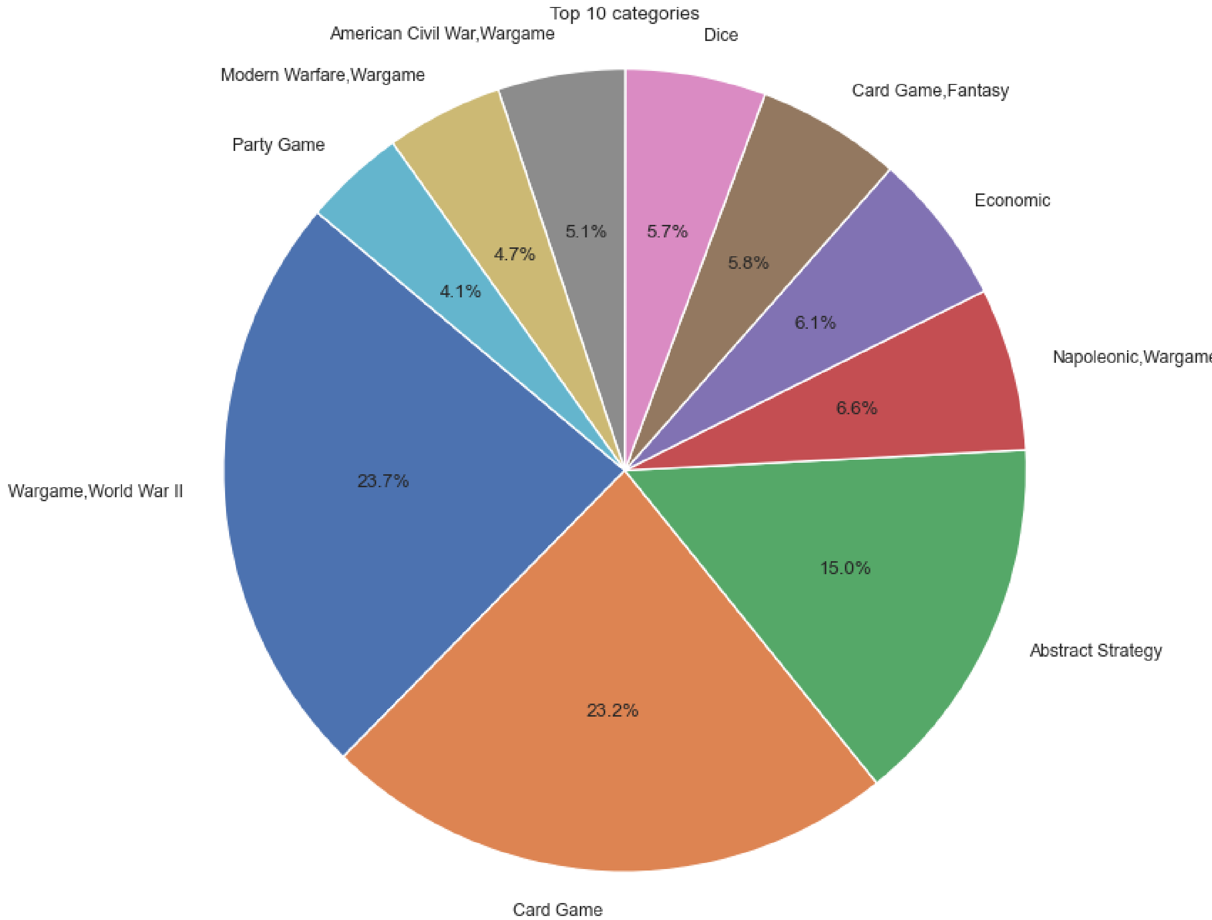
Out[21]:

	category	count	Percentage
0	Wargame,World War II	449	4.301590
1	Card Game	438	4.196206
2	Abstract Strategy	284	2.720828
3	Napoleonic,Wargame	124	1.187967
4	Economic	116	1.111324
5	Card Game,Fantasy	110	1.053842
6	Dice	107	1.025101
7	American Civil War,Wargame	97	0.929297
8	Modern Warfare,Wargame	89	0.852654
9	Party Game	77	0.737689

In [22]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

picking only top 10 of the categories

In [23]:
```
top_10_categories = value_counts_df.head(10)
#pie chart is the easyiest way to see it
plt.figure(figsize=(8, 8))
plt.pie(top_10_categories['count'], labels = top_10_categories['category'], autopct= '
plt.title('Top 10 categories')
plt.axis('equal')
plt.show()
```



The above show the top 10 categories between the whole Dataset which was larger.

Those top 10 provide a valuable ideia for a sale strategy organisation as they represent the most popular games in terms of engagement. The categories"Card wargame" and "World war 2" are the most popular been almost 50% of the total, following by the others where "Party Games" is the one which lower porcentage, only based on the top 10, no the whole dataset, once was choseen only 10 of the total piechart was fully capable for a good visualisation.

Do older games (1992 and earlier) have a higher MEDIAN “average rating” than newer games (after 1992)?

In [24]:
```
df_comparison= df[['name', 'year_published', 'average_rating']]
```

In [25]:
```
df_comparison.head()
```

Out[25]:

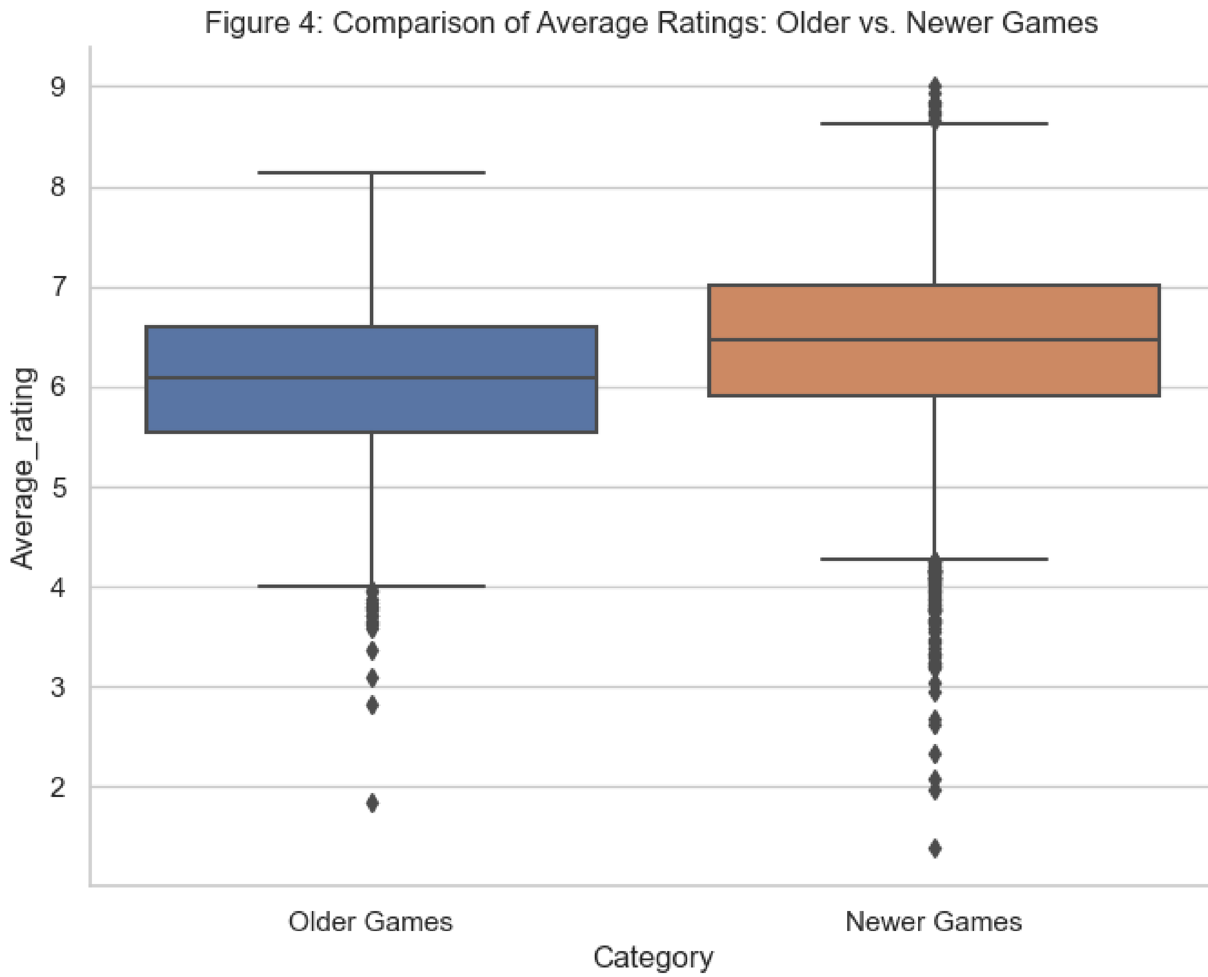
	name	year_published	average_rating
0	Die Macher	1986	7.66508
1	Dragonmaster	1981	6.60815
2	Samurai	1998	7.44119
3	Tal der Könige	1992	6.60675
4	Acquire	1964	7.35830

older Games(1992 and before) have a higher MEDIAN "average rating" than newer games afterm1992?

In [26]:
```
# Creating a new column 'category' based on 'year_published'
df_comparison.loc[df_comparison['year_published'] <= 1992, 'category'] = 'Older Games'
df_comparison.loc[df_comparison['year_published'] > 1992, 'category'] = 'Newer Games'
```

```
C:\Users\dansa\AppData\Local\Temp\ipykernel_10352\1238132925.py:2: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  df_comparison.loc[df_comparison['year_published'] <= 1992, 'category'] = 'Older Gam
es'
```

In [27]:
```
def create_comparison_boxplot(dataframe, x_column, y_column, title):

    # Set the Seaborn style to mimic The Economist's style
    sns.set(style="whitegrid")

    #Creating a boxplot
    plt.figure(figsize=(8, 6))
    ax = sns.boxplot(x=x_column, y=y_column, data=dataframe)
    sns.despine(right=True, top=True)

    #Adding labels and title
    plt.xlabel(x_column.capitalize())
    plt.ylabel(y_column.capitalize())
    plt.title(title)
    plt.show()
create_comparison_boxplot(df_comparison, 'category', 'average_rating', 'Figure 4: Comp
```



the above provides a detailed comparison of the average ratings between two distinct sets of games: the older games (published prior to 1992) and the newer games (published in 1992 and after 1992). To explore the difference present in these two groups, we used boxplots. These boxplots offer a clear distribution of ratings for each group, evaluating the difference in their respective medians (represented by the line within each box). The boxplots clearly show a slight difference in the median ratings between the two groups, raising the initial possibility of a distinction in player preferences. In order to confirm this observed difference and draw statistically significant conclusions regarding the medians of these two groups, we carried out a quantitative statistical hypothesis.

In [28]:
```
df_comparison.head()
```

Out[28]:

	name	year_published	average_rating	category
0	Die Macher	1986	7.66508	Older Games
1	Dragonmaster	1981	6.60815	Older Games
2	Samurai	1998	7.44119	Newer Games
3	Tal der Könige	1992	6.60675	Older Games
4	Acquire	1964	7.35830	Older Games

In [29]:
```
import scipy.stats as stats
import numpy as np
import pandas as pd
```

In [30]:
```
# two groups for the Mann-Whitney U test
older_games_ratings = df_comparison[df_comparison['category'] == 'Older Games']['avera
newer_games_ratings = df_comparison[df_comparison['category'] == 'Newer Games']['avera
```

In [31]:
```
# Making the size of the smaller group
min_group_size = min(len(older_games_ratings), len(newer_games_ratings))

# Subsampling the larger group to have the same size as the smaller group
np.random.seed(42)
if len(older_games_ratings) > len(newer_games_ratings):
    subsample_old = older_games_ratings.sample(n=min_group_size, replace=False)
    subsample_new = newer_games_ratings
else:
    subsample_old = older_games_ratings
    subsample_new = newer_games_ratings.sample(n=min_group_size, replace=False)
```

In [32]:
```
# Mann-Whitney U test
u_statistic, p_value = stats.mannwhitneyu(subsample_old, subsample_new, alternative='g
```

In [33]:
```
#summary table
results_summary = pd.DataFrame({
    'Category': ['Older Games', 'Newer Games'],
    'Sample Size': [len(subsample_old), len(subsample_new)],
    'U Statistic': [u_statistic, None],
    'P-Value': [p_value, None]
})
results_summary
```

Out[33]:

	Category	Sample Size	U Statistic	P-Value
0	Older Games	1934	1332223.5	1.0
1	Newer Games	1934	NaN	NaN

After a Mann-Whitney U test to test the following hypotheses:

H0: There is no significant difference in the median "average rating" between older games (published in 1992 or earlier) and newer games (published after 1992).

H1: Older games (published in 1992 or earlier) have a higher median "average rating" than newer games (published after 1992).

The choice of the Mann-Whitney U test was appropriate for this hypothesis test due to its non-parametric nature, making it robust against distributional assumptions. Due to the disparity in the sizes of the two samples, 1,934 observations in the "older_games_ratings" group and 8,598 in the "newer_games_ratings" group we did subsampling which involves taking a smaller sample from newer_games_rAatings, without replacement, to obtain a distribution of test statistics (such as U-statistics) or p-values, This helped in reducing potential for bias in the Mann-Whitney U test results resulting from imbalanced sample sizes, the larger group may have more diverse characteristics, and the smaller group may be less representative. By subsampling, you can make the groups more comparable, potentially reducing the impact of these biases.

Based on the Table , the p-value(1.0) exceeded our chosen significance level (0.05). This implies that the data did not provide strong evidence against the null hypothesis. Thus, there is insufficient statistical evidence to conclude that older games (published in 1992 or earlier) have a higher median "average rating" than newer games (published after 1992).

What are the 5 most common “mechanics” in the dataset?

In [34]:
```
df_mechanic = df ['mechanic'].value_counts().reset_index()
df_mechanic.columns = ['Mechanic', 'Count']
```

In [35]:
```
#reindex the df
df_mechanic = df_mechanic.reset_index(drop= True)
df_mechanic = df_mechanic.sort_values(by='Count', ascending=False)
```

In [36]:
```
df_mechanic.head()
```

Out[36]:

	Mechanic	Count
0	Hex-and-Counter	523
1	Hand Management	297
2	Dice Rolling	222
3	Roll / Spin and Move	199
4	Tile Placement	170

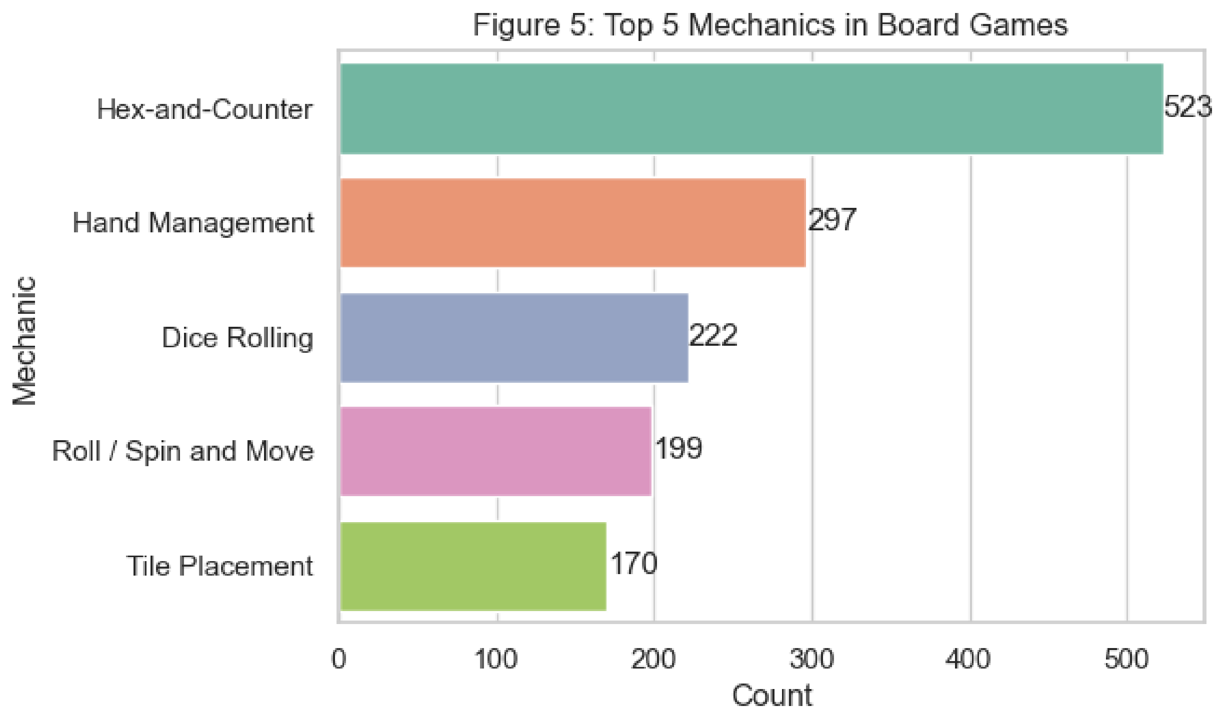
Ploting the top 5

In [37]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [38]:
```
top_5_mechanics = df_mechanic.head(5)
```

In [39]:
```
sns.set(style="whitegrid")

# Creating a bar plot
plt.figure(figsize=(6, 4))
ax = sns.barplot(x='Count', y='Mechanic', data=top_5_mechanics, palette='Set2')

# Add labels,title and values to the bar
plt.xlabel('Count')
plt.ylabel('Mechanic')
plt.title('Figure 5: Top 5 Mechanics in Board Games')
for p in ax.patches:
```

```
    ax.annotate(f'{int(p.get_width())}', (p.get_width() + 0.1, p.get_y() + p.get_heigh
plt.show()
```



illustrating the distribution of game mechanics, with Hex-and-Counter ranking as the most frequently employed mechanic, appearing in 523 times. In contrast, Hand Management comes in second with a considerable gap at 297 occurrences. The top 5 list concludes with the Tile Placement mechanic, featured in 170 instances. The selection of a horizontal bar chart, set against a whitegrid backdrop, was chosen for its ability to convey this information with visual clarity and aesthetic appeal, facilitating easy understanding.

Is there greater diference in the median of the numbers of users that rated newer games after 1999 and before that year?

In order to improve on our gaming sales strategy, we recognized the importance of measuring the willingness of gamers to express their sentiments in both the 21st and 20th centuries. This comparison serves as a valuable indicator to measure whether advancements in sentiment-sharing platforms over time have influenced gamers' engagement, Specifically, it helps us determine whether there is a compelling reason for enhancing the platforms used by gamers to voice their sentiments in the 21st century, as compared to those in the 20th century. Our decision-making process, including potential platform scaling initiatives to accommodate increased sentiment sharing, depends on a key metric: the comparison of medians between the two time periods, The use of median values is preferred in this context due to their robustness against the influence of outliers, ensuring that our analysis reflects the central tendencies of gamers' sentiments accurately.

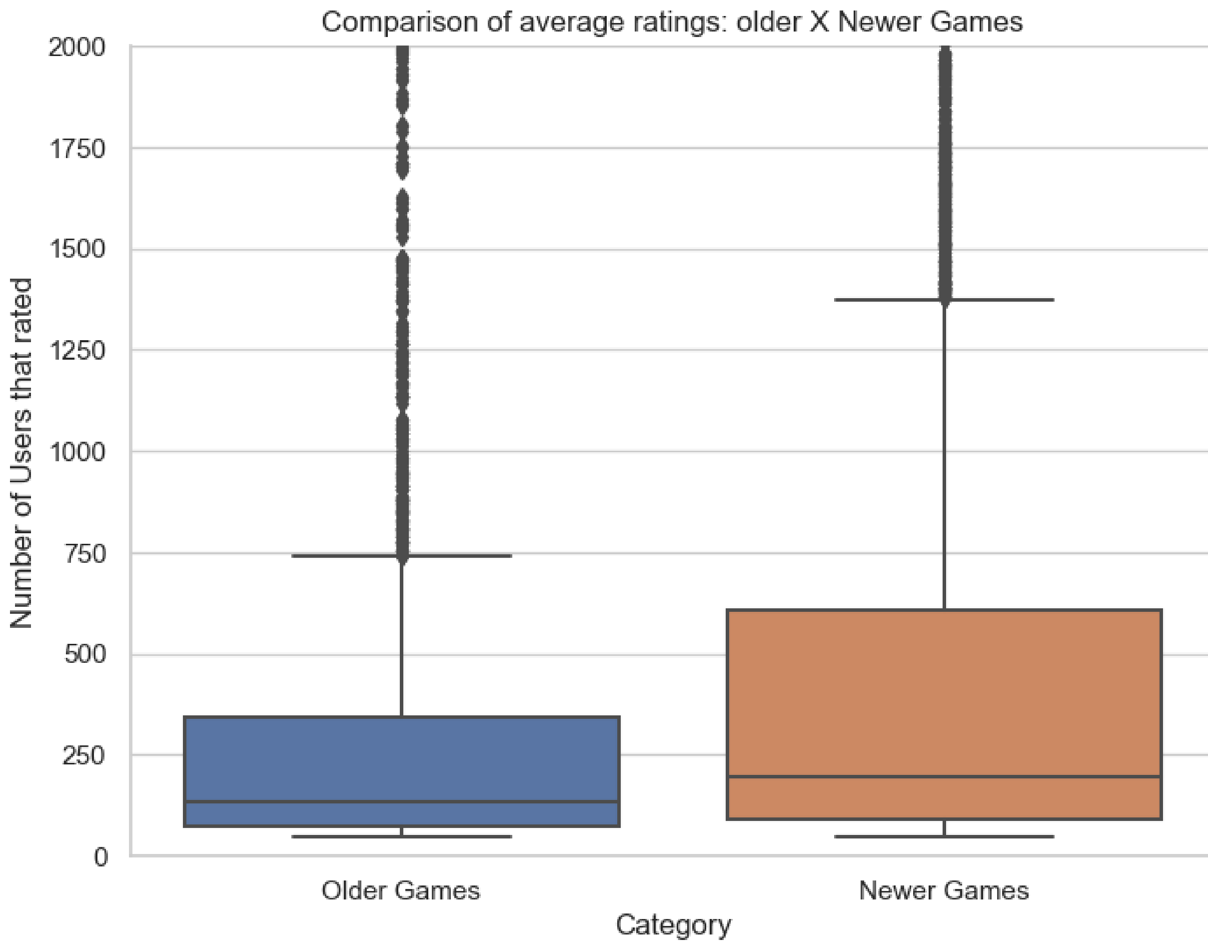
In [40]:
```
df_user_rating = df[['name', 'year_published', 'users_rated']]
```

In [41]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [42]:
```
#Creating a new collumn Category based on year_published
df_user_rating.loc[df_user_rating['year_published'] <= 1999, 'category'] = 'Older Game
df_user_rating.loc[df_user_rating['year_published'] > 1999, 'category'] = 'Newer Games
```

```
C:\Users\dansa\AppData\Local\Temp\ipykernel_10352\1099949560.py:2: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  df_user_rating.loc[df_user_rating['year_published'] <= 1999, 'category'] = 'Older G
ames'
```

In [43]:
```
sns.set(style="whitegrid")
plt.figure(figsize=(8, 6))
ax = sns.boxplot(x= 'category', y= 'users_rated', data=df_user_rating)
#removing the top and right spines
sns.despine(right=True, top=True)
#adding labeling and title
plt.xlabel('Category')
plt.ylabel('Number of Users that rated')
plt.title('Comparison of average ratings: older X Newer Games');
plt.ylim(0, 2000);
plt.show()
```



Above shows boxplots used to visualize summary statistics for both older(1999 and before) and newer(after 1999) games, enhancing comparison with distinct color shades, To improve clarity, the number of user ratings was capped at 2000, allowing for better visibility of the medians (represented by the lines within each box). The utilization of various color shades and a whitegrid theme not only improves the aesthetics but also helps in improving overall visibility, Notably, newer games exhibit a slightly higher median in terms of user ratings, To confirm the significance of this difference, a quantitative hypothesis test was employed.

In [44]:
```
import scipy.stats as stats
import pandas as pd
import numpy as np
```

In [45]:
```
#two groups for the Mann-whitney U test
older_games_ratings = df_user_rating[df_user_rating['category'] == 'Older Games']['use
newer_games_ratings = df_user_rating[df_user_rating['category'] == 'Newer Games']['use
```

In [46]:
```
#groupsize
min_group_size = min(len(older_games_ratings), len(newer_games_ratings))
#tunring into same size
np.random.seed(42)
if len(older_games_ratings) > len(newer_games_ratings):
    subsample_old = older_games_ratings.sample(n=min_group_size, replace=False)
    subsample_new = newer_games_ratings
else:
    subsample_old = older_games_ratings
    subsample_new = newer_games_ratings.sample(n=min_group_size, replace=False)
```

In [47]:
```
#Mann-whitney U test
u_static, p_value = stats.mannwhitneyu(subsample_old, subsample_new, alternative="two-
```

In [48]:
```
#creating a table and showing results
results_summary = pd.DataFrame({
    'Category': ['Older Games', 'Newer Games'],
    'Sample Size': [len(subsample_old), len(subsample_new)],
    'U Statistic': [u_statistic, None],
    'P-Value': [p_value, None]})
results_summary
```

Out[48]:

	Category	Sample Size	U Statistic	P-Value
0	Older Games	2980	1332223.5	6.383455e-26
1	Newer Games	2980	NaN	NaN

Results of Mann-whitney u test for game comparision

"The above presents the results of our hypothesis testing, where we examined the following assertions:

H0: There is no significant difference in the number of users who rated games before the year 2000 compared to those rated in the years 2000 and beyond. H1: A significant difference exists in the number of users who rated games before 2000 compared to games rated from 2000 onward. Due to challenges posed by uncertain data distribution assumptions, we opted for a non-parametric method, the Mann-Whitney test, to test our hypotheses. Our analysis yielded a p-value of 6.383455e-26, which is less than our chosen level of significance (0.05). As a result, we reject the null hypothesis, suggesting that there is no statistically significant distinction between newer and older games in terms of the number of users who rated them." In conclusion, there exists asignificant difference in the median of the number of users that expressed their ratings in 1999 and prior(20th century) compared to 2000 and after(21st century).

References

P, B. (2020). Making Plots in Jupyter Notebook Beautiful & More Meaningful. [online] Medium. Available at: https://towardsdatascience.com/making-plots-in-jupyter-notebook-beautiful-more-meaningful-23c8a35c0d5d [Accessed 14 Oct. 2023].

Waskom, M. (n.d.). Overview of seaborn plotting functions — seaborn 0.12.0 documentation. [online] seaborn.pydata.org. Available at: https://seaborn.pydata.org/tutorial/function_overview.html [Accessed 12 Oct. 2023].

Glen, S. (2022). Correlation coefficient: simple definition, formula, easy steps. [online] Statistics How to. Available at: https://www.statisticshowto.com/probability-and-statistics/correlation-coefficient-formula/ [Accessed 12 Oct. 2023].

Statistics Solutions. (n.d.). Mann-Whitney U Test. [online] Available at: https://www.statisticssolutions.com/free-resources/directory-of-statistical-analyses/mann-whitney-u-test/#:~:text=Mann%2DWhitney%20U%20test%20is [Accessed 13 Oct. 2023].

https://github.com/DanielFerreirajr/Data-Visualization

```
In [ ]:
```