cct | College Dublin
Computing • IT • Business

Programme Title: HDIP Data Analytics

Module Title: Data Visualization Techniques

Assessment Title: CA1_DVis_HDip_LvI8

Lecturer Name: David McQuaid

Student Full Name: Laercio Santos Lima

Student Number: 2022055

Assessment Due Date: 01st Nov 2022

Date of Submission: 01st Nov 2022

Declaration

By submitting this assessment, I confirm that I have read the CCT policy on Academic Misconduct and understand the implications of submitting work that is not my own or does not appropriately reference material taken from a third party or other source. I declare it to be my own work and that all material from third parties has been appropriately referenced. I further confirm that this work has not previously been submitted for assessment by myself or someone else in CCT College Dublin or any other higher education institution.

Abstract

Board games can be explained as any game in which pieces are moved in particular ways on a board marked with a pattern. For this project, different data visualization techniques will be explored in order to analyze the dataset called "board_games.csv". As requested in the Assessment Task, different questions related to the dataset will be answered, in order to determine the sales strategy for the company in their upcoming Winter Season. Different approaches for the same questions will also be used, in order to take advantage of different visualization libraries, such as Matplotlib, Seaborn, Plotly Express, Altair, and etc. In order to explain the thinking process, conclusions for each question are also provided. All written work was done using Jupyter Notebook and Python.

Table of Contents

Data Dictionary

variable	class	description
game_id	text	Unique game identifier
description	text	A paragraph of text describing the game
image	text	URL image of the game
max_players	integer	Maximum recommended players
max_playtime	integer	Maximum recommended playtime (min)
min_age	integer	Minimum recommended age
min_players	integer	Minimum recommended players
min_playtime	integer	Minimum recommended playtime (min)
name	text	Name of the game
playing_time	integer	Average playtime
thumbnail	text	URL thumbnail of the game
year_published	integer	Year game was published
artist	text	Artist for game art
category	text	Categories for the game (separated by commas)
compilation	text	If part of a multi-compilation - name of compilation
designer	text	Game designer
expansion	text	If there is an expansion pack - name of expansion
family	text	Family of game - equivalent to a publisher
mechanic	text	Game mechanic - how game is played, separated by comma
publisher	text	Comoany/person who published the game, separated by comma
average_rating	double	Average rating on Board Games Geek (1-10)
users_rated	double	Number of users that rated the game

1. Introduction

The dataset used in this project consists of information about board games. The analysis completed in this project will help to determine the sales strategies for the company in their upcoming Winter Season.

EDA and data engineering will be performed in order to better understand and organize the dataset. The most important steps will be described and explained while every question (requested in the Assessment Task) is answered. In addition to it, every question has its conclusion, which explains better the thinking process and the accomplishments of each graph. In other words, the Part 3 of this project, that is related to explaining decisions, will be included in Parts 1 and 2.

Different approaches were used to answer the questions. In other words, every graph used is able to answer the questions; however, different approaches show different relevant information, adding extra information when necessary. The rationale behind this is the intention to keep showing more and more information if the company decides to spend more time analyzing the graph.

Regarding colors and fonts of the graphs used in this CA, I decided not to change some of them because I understood, after plotting the graph, that the visualization was harmonic and understandable. This decision was taken after studying and analyzing the Basic Color Theory and The Color Wheel. When that was not the case, modifications were implemented.

2. Importing libraries

This CA is a part of a data visualization project. In order to complete it, different libraries were used. According to Project Pro (2022), some libraries used in this project are considered some of the "Top 10 Python Data Visualization Libraries".

In [1]:
```
# Importing some libraries
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.graph_objects as go
import altair as alt
alt.data_transformers.disable_max_rows()
```

Out[1]: DataTransformerRegistry.enable('default')

3. EDA

Exploratory Data Analysis (EDA) is an important step for this project since it is in this part where I am able to identify features patterns. According to Patil (2018), EDA can be defined as a critical process of performing initial investigations in a dataset. As previously mentioned in this report, different data visualization techniques will be performed. Consequently, here in the EDA I intend to discover the necessary patterns and characteristics that will help me to create better graphs in the future. Besides that, any extra anomalies spotted in this stage may be treated in future steps.

- **Info and describe**

In order to have a better idea of the dataset characteristics, I decided to use info() and describe(). As it is possible to see below, the dataset contains 10532 observations and 22 features. After using info(), I get some information about the dataset features, for example, Null Count and Dtype. Besides that, some information about the number of observations and features can also be seen. Regarding describe(), I can see the 5 number summary. In addition to it, the mean and the standard deviation can also be seen. To conclude, all this information may be important during the cleaning and preparation along this CA.

- **Shape, size and number of zeros**

I decided to show some characteristics regarding the dataset's size. In order to do it, I am going to check its size, shape and number of zeros. As it is possible to see, there are values == 0. It may be important because it gives me a better idea and understanding of the dataset. By checking its number of zeros and comparing with its size, I can start thinking about solutions to problems.

- **Duplicated rows**

According to Bhutani (2018), another important part of data analysis is analyzing duplicated values, and subsequently, deciding to remove them or not. Based on that, I am going to look for duplicated rows in the dataset. In order to do it, first I am going to create a new dataframe. After that, I am going to show the amount of rows that are duplicated. Sometimes duplicated rows may interfere in the analysis, since sometimes they do not represent real information collected. However, as it is possible to confirm, there are no duplicated rows in the dataset.

- **NA, NaN and Null**

According to Chandradas (2021), NaN is a short form for Not A Number. In other words, it is a possible form to show a missing value in a dataset. With this information, I decided to count the amount of missing values in the dataset. In order to do this, I am going to use isna() and sum(), since my intention is to better understand the situation with missing values in the dataset. Identifying missing values in a dataset is an important step. Missing data may affect the understanding. Depending on the dataset, different decisions and actions may be taken in order to solve this issue. As it is possible to confirm, some features have lots of missing values.

- **Features: category and mechanic**

As part of this project, these two features need to be analyzed. As it is possible to confirm, the categories and the mechanics are separated by commas. In future steps of this project, this issue will be addressed.

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

# first rows
df.head(2)
```

Out[2]:

	game_id	description	image	max_players	max_playtime	min_age	min_players	min_playtime	name	pla
0	1	Die Macher is a game about seven sequential po...	//cf.geekdo-images.com/images/pic159509.jpg	5	240	14	3	240	Die Macher	
1	2	Dragonmaster is a trick-taking card game based...	//cf.geekdo-images.com/images/pic184174.jpg	4	30	12	3	30	Dragonmaster	

2 rows × 22 columns

In [3]:
```
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 [4]:
```
# Showing some info
df.describe()
```

Out[4]:

	game_id	max_players	max_playtime	min_age	min_players	min_playtime	playing_time	year_published	average_rating
count	10532.000000	10532.000000	10532.000000	10532.000000	10532.000000	10532.000000	10532.000000	10532.000000	10532.000000
mean	62059.203095	5.657330	91.341436	9.714964	2.070547	80.882738	91.341436	2003.070832	6.370856
std	66223.716828	18.884403	659.754400	3.451226	0.664394	637.873893	659.754400	12.278296	0.850364
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1950.000000	1.384210
25%	5444.500000	4.000000	30.000000	8.000000	2.000000	25.000000	30.000000	1998.000000	5.829585
50%	28822.500000	4.000000	45.000000	10.000000	2.000000	45.000000	45.000000	2007.000000	6.392965
75%	126409.500000	6.000000	90.000000	12.000000	2.000000	90.000000	90.000000	2012.000000	6.942675
max	216725.000000	999.000000	60000.000000	42.000000	9.000000	60000.000000	60000.000000	2016.000000	9.003920

In [5]:
```
# looking for objects
df.describe(include = object)
```

Out[5]:

	description	image	name	thumbnail	artist	category	compilation	de
count	10532	10531	10532	10531	7759	10438	410	
unique	10528	10527	10357	10527	4641	3860	336	
top	How could that have happened? Black Stories ar...	//cf.geekdo-images.com/images/pic2410035.png	Robin Hood	//cf.geekdo-images.com/images/pic2410035_t.png	Franz Vohwinkel	Wargame,World War II	Traveller: The Classic Games, Games 1-6+	(Uncr
freq	3	2	5	2	166	449	8	

In [6]:
```
# Checking the shape
df.shape
```

Out[6]: (10532, 22)

In [7]:
```
# Checking the size
df.size
```

Out[7]: 231704

In [8]:
```
# Checking the number of zeros
number_of_zeros = (df.to_numpy() == 0).sum()
print (number_of_zeros)
```

1698

In [9]:
```
# looking for duplicated rows
dup_df = df[df.duplicated()]
```

In [10]:
```
# Showing the number of duplicated rows
print("Number of duplicated rows: ", dup_df.shape)
```

Number of duplicated rows: (0, 22)

In [11]:
```
#Looking for missing values
df.isna().sum()
```

Out[11]:
```
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 [12]:
```
# checking this feature
df['category'].head(10)
```

Out[12]:
```
0    Economic,Negotiation,Political
1                 Card Game,Fantasy
2       Abstract Strategy,Medieval
3                           Ancient
4                          Economic
5             Civilization,Nautical
6                 Abstract Strategy
7              Civilization,Fantasy
8                       Exploration
9                    Fantasy,Travel
Name: category, dtype: object
```

In [13]:
```
# checking this feature
df['mechanic'].head(10)
```

Out[13]:
```
0    Area Control / Area Influence,Auction/Bidding,...
1                                         Trick-taking
2    Area Control / Area Influence,Hand Management,...
3    Action Point Allowance System,Area Control / A...
4         Hand Management,Stock Holding,Tile Placement
5                                         Dice Rolling
6    Area Enclosure,Pattern Building,Pattern Recogn...
7                                        Modular Board
8         Area Control / Area Influence,Tile Placement
9    Card Drafting,Hand Management,Point to Point M...
Name: mechanic, dtype: object
```

Even though I decided not to include it in the final PDF, I have also used the Pandas Profiling library in order to get some extra information regarding the EDA stage and the dataset as a whole. The code I used for that is commented below.

In [14]:
```
#import pandas_profiling
```

In [15]:
```
#pandas_profiling.ProfileReport(df)
```

4. Part 1

4.1 What are the top 5 "average rated" games?

As it was possible to confirm during the EDA, this dataset contains a feature called "name" and another one called "average_rating". Keeping this in mind, I decided to create a variable called "top_5_games". In this variable I used "sort_values", "by=average_rating" and "ascending=False" in order to organize the games by the average rating, from the best to the worst. I also used "head(5)" to keep only the top 5 games.

It is possible to see the results below:

In [16]:
```
# Organizing the top 5 games by average_rating
top_5_games = df.sort_values(by='average_rating', ascending=False).head(5)
top_5_games
```

Out[16]:

	game_id	description	image	max_players	max_playtime	min_age	min_players	min_playtime
8348	140135	Small World's Designer Edition is a spare-no-e...	//cf.geekdo-images.com/images/pic2270432.jpg	6	80	8	2	40
6392	55690	Kingdom Death: Monster is a fully cooperative ...	//cf.geekdo-images.com/images/pic2931007.jpg	6	180	17	1	60
9964	181289	Terra Mystica: Big Box, a Korean crowdfunding ...	//cf.geekdo-images.com/images/pic2602334.jpg	5	150	12	2	60
8526	144574	(from MMP website:)

Last Chance for V... | //cf.geekdo-images.com/images/pic1875530.jpg | 2 | 60 | 15 | 2 | 60 |
| 9675 | 173504 | The Greatest Day: Sword, Juno, and Gold Beache... | //cf.geekdo-images.com/images/pic2422496.jpg | 8 | 6000 | 12 | 2 | 60 |

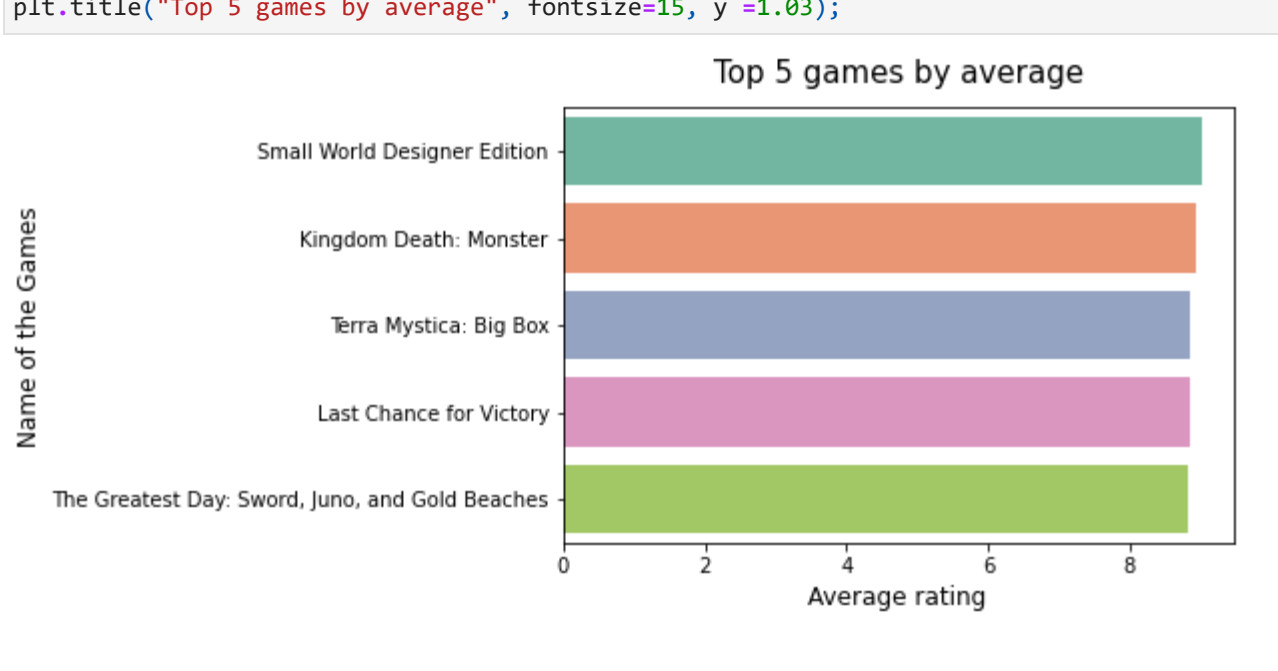
5 rows × 22 columns

4.1.1 First Basic visualization

With the data organized, at first I used "sns.barplot" to visualize the top 5 games. As I'm just checking the name of the game and its rate, this chart facilitates the understanding. As there are only numbers to represent the rates, my X is the average rate. Y is the name of the games.

However, as it is possible to confirm below, it is difficult to see and understand the average rate of each game just with the chart.

In [17]:
```
# Plotting the chart
sns.barplot(data=top_5_games, x="average_rating", y="name", palette='Set2')
plt.xlabel("Average rating", fontsize=12)
plt.ylabel("Name of the Games", fontsize=12)
plt.title("Top 5 games by average", fontsize=15, y =1.03);
```



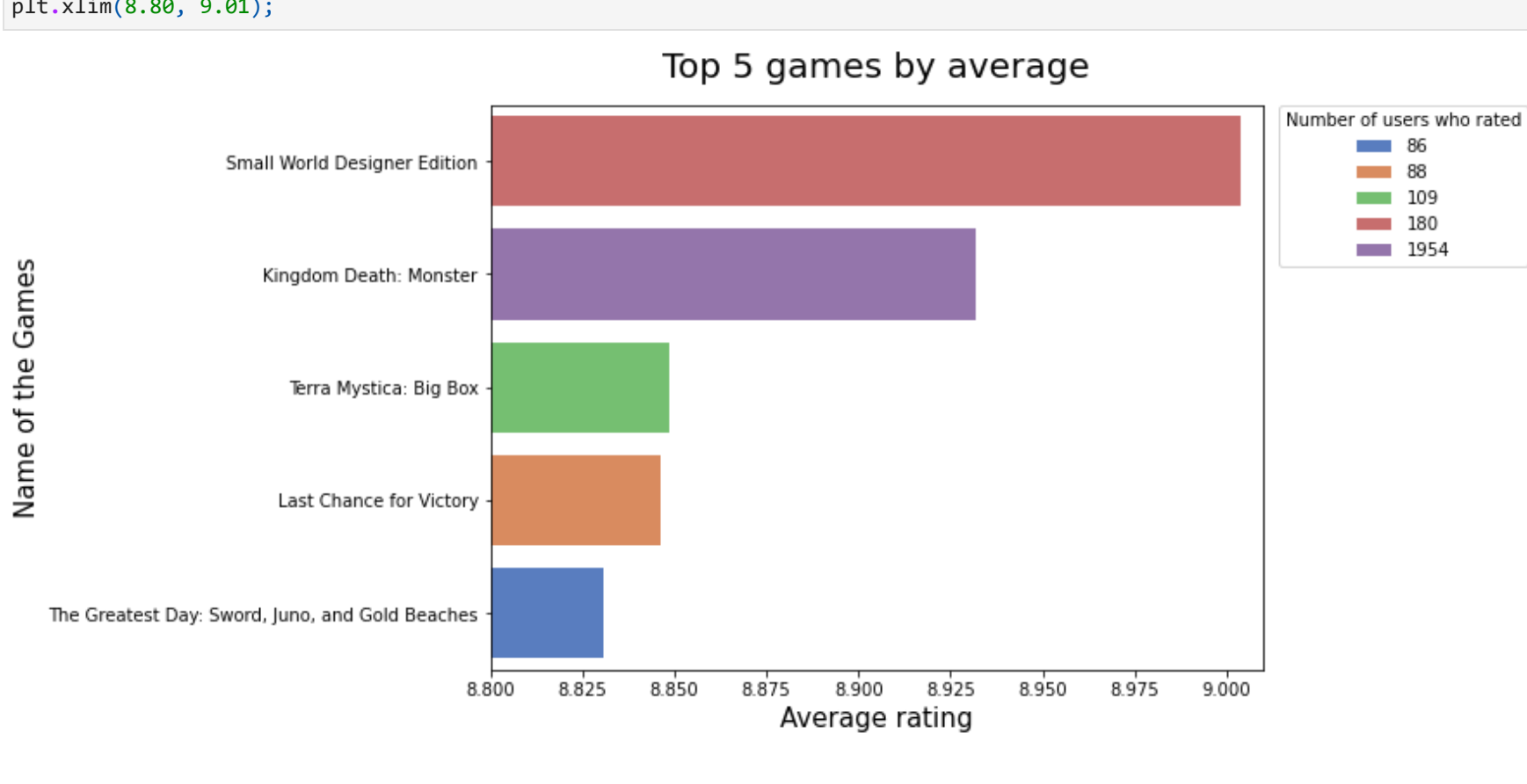
4.1.2 Improving the visualization

In order to make the visualization better, I decided to change and implement the following:

- **Title, X label and Y label:** With this information it is possible to have a general idea of the chart by just looking at it.
- **Title:** I changed the font size to 20 since it is the title. I also moved the title a bit higher, just to create some space between the title and the chart (y =1.03).
- **X label and Y label:** I changed the font size to ease visualization. I also renamed both labels.
- **Figure size: (8, 6)** : a bit bigger to ease the visualization. X larger, since I wanted more distance between the points.
- **Hue and Colors:** different and contrasting colors for different games. I used "hue" and "dodge" in order to give a meaning to the colors in the chart. The colors now represent the number of users who rated the game. I changed the palette for better visualization. I added a legend box with this information and moved this box outside the chart using "plt.legend" and its parameters.
- **X range:** The lowest rate among the top 5 games is 8.83. The highest rate is 9.00. Keeping this in mind, I defined my range in (8.80, 9.01). It helps to visualize the rates, since all of them are pretty close.

This is the result:

In [18]:
```
# Making some changes in the first chart
fig, ax = plt.subplots(figsize=(8, 6))
sns.barplot(data=top_5_games, x="average_rating", y="name", hue='users_rated', dodge=False, palette="muted")

plt.xlabel("Average rating", fontsize=15)
plt.ylabel("Name of the Games", fontsize=15)
plt.title("Top 5 games by average", fontsize=20, y =1.03)

plt.legend(bbox_to_anchor=(1.02, 1), loc='upper left', borderaxespad=0, title='Number of users who rated')

plt.xlim(8.80, 9.01);
```



4.1.3 Showing Different Information

With the main question already answered, I decided to create a few charts to show in different ways the top 5 games.

At first, I decided to compare the top 5 games with the average of all the other games in the dataset. It is important since this chart shows how good the top 5 games are, when taking into consideration the entire dataset. The average of all the games is the sixth bar.

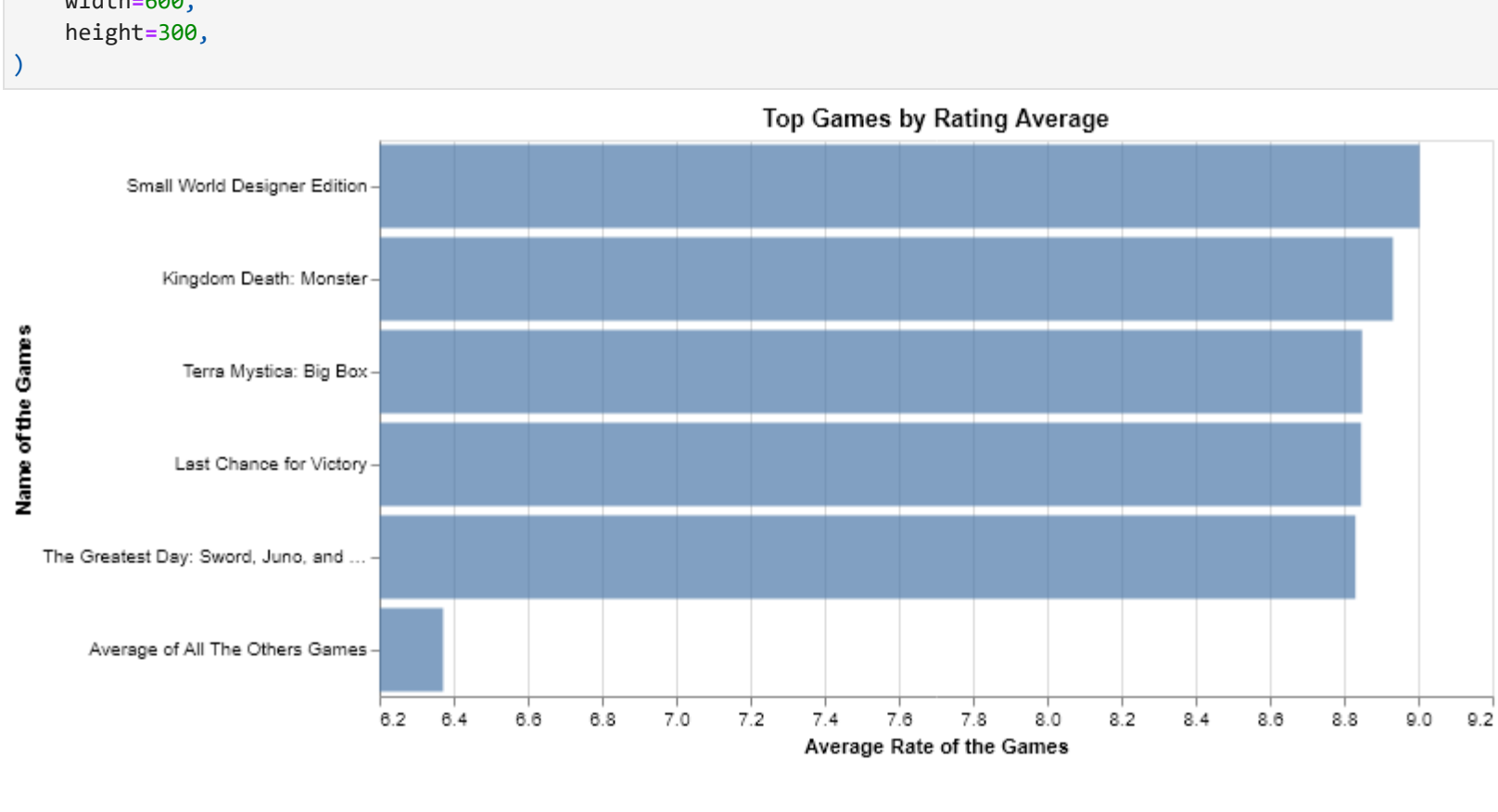
For this chart I decided to use Altair. Those are the main parameters I implemented:

- **Mark_bar:** As I'm just checking the name of the game and its rate. I changed the opacity to 0.7 since I wanted to show what is behind the bars.
- **Transform_aggregate and transform_window:** In order to create the sixth bar, I had to group the games by name and check the general mean of the feature average_rating.
- **alt.X and alt.Y:** I changed the title, the order and the scale in order to have a better visualization.
- **transform_calculate:** by using "datum.rank < 6" I am just showing the first 5 games on the list organized by rank. The average of the rest is shown as the sixth bar.
- **properties:** I included a title and changed the width and height to improve visualization.

This was the final result:

In [19]:
```
# Creating a chart to show the average of the games not in the top 5
alt.Chart(df).mark_bar(opacity=0.7).encode(
    x=alt.X("aggregate_average:Q",
            aggregate="mean",
            title='Average Rate of the Games',
            scale=alt.Scale(zero=False)

            ),
    y=alt.Y("ranked_game:N",
            sort=alt.Sort(op="mean", field="aggregate_average", order="descending"),
            title='Name of the Games'

            ),


).transform_aggregate(
    aggregate_average='mean(average_rating)',
    groupby=['name'],
).transform_window(
    rank='row_number()',
    sort=[alt.SortField("aggregate_average", order="descending")],
).transform_calculate(
    ranked_game="datum.rank < 6 ? datum.name : 'Average of All The Others Games'"
).properties(
    title='Top Games by Rating Average',
    width=600,
    height=300,
)
```

Out[19]:



4.1.4 Changing the approach

A different way of showing the top 5 games is by using points. It may be a good approach since with points, in addition to the X and Y dimension, I can add dimensions such as color and size. This increases the information I can show in a simple chart.

With this in mind, I decided to use one more chart.

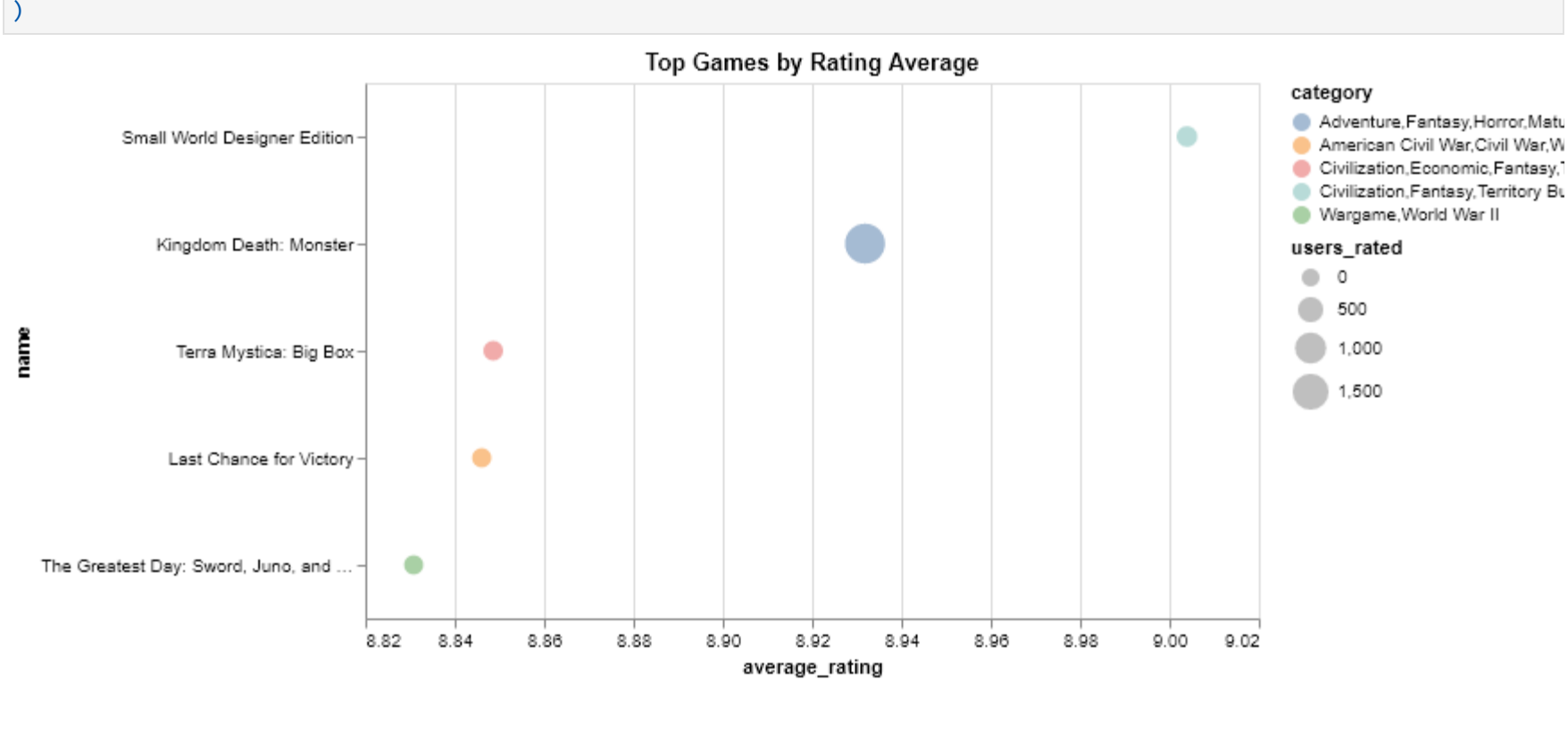
I decided to keep using altair. Those are the main implements I used:

- **Mark_point:** instead of bars, I am using points at this time. The points are filled.
- **alt.X and alt.Y:** I changed alt.Scale in order to zoom in only in the part where there is information. sort='-x' is used to put the best game on top.
- **alt.Size:** I am using users_rated for this. Bigger points means that more users rated the game. I also changed the range of the points size, because I wanted to keep all the points in a good size.
- **alt.Color:** I am using different colors to show different categories of the games.
- **alt.OpacityValue(0.5):** it looks softer and nicer in the chart.
- **Tooltip:** In order to make the chart more interactive, by using tooltip I am creating a "hover" to show extra information, such as the name of the game, its average rate, the number of users who rated the game, its category and its mechanic.
- **properties:** I included a title and changed the width and height to improve visualization.

And this is the result:

In [20]:
```
alt.Chart(top_5_games).mark_point(filled=True).encode(
    alt.X('average_rating:Q', scale = alt.Scale(zero = False)),
    alt.Y('name:N', scale = alt.Scale(zero = False), sort='-x'),
    alt.Size('users_rated:Q', scale=alt.Scale(range=[100, 500])),
    alt.Color('category:N'),
    alt.OpacityValue(0.5),

    tooltip=[alt.Tooltip('name:N'),
             alt.Tooltip('average_rating:Q'),
             alt.Tooltip('category:N'),
             alt.Tooltip('mechanic:N')
             ]

).properties(
    title='Top Games by Rating Average',
    width=500,
    height=300,
)
```

Out[20]:



4.1.5 Conclusion

In this question, the main idea was to find the top 5 games with the highest average rate. My goal to answer this question was to create different plots that would give this information at first glance.

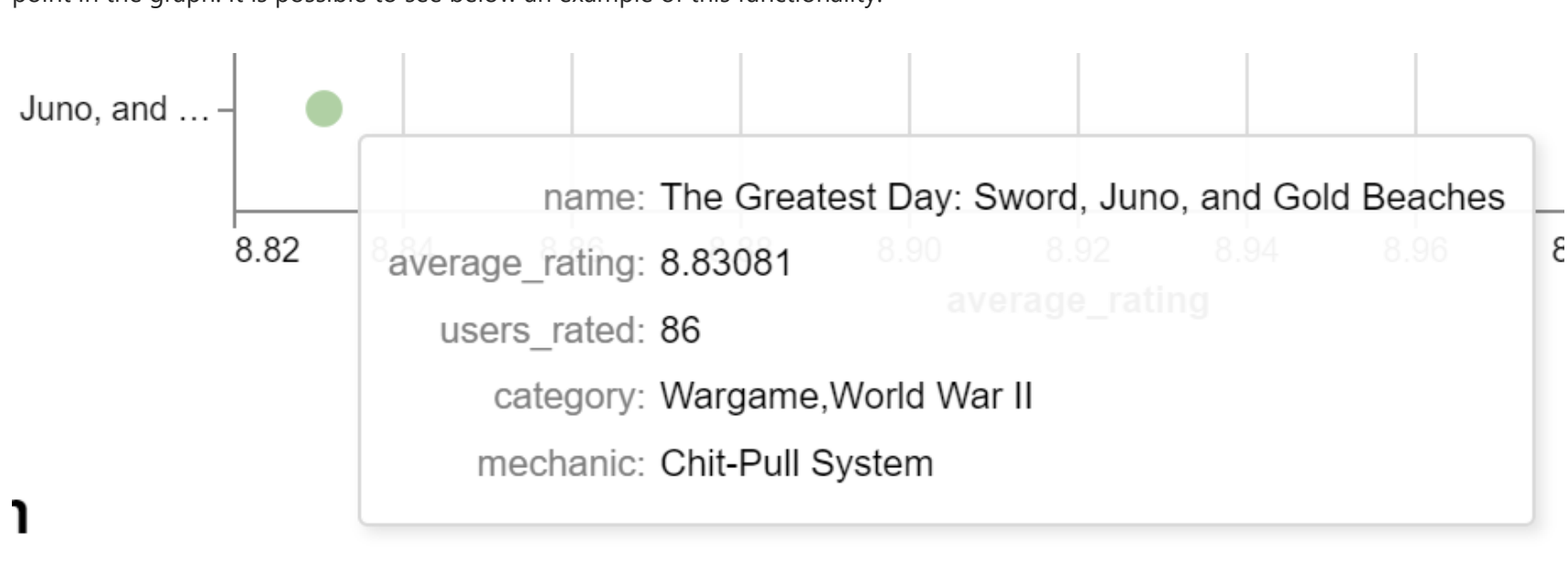
Keeping this in mind I first created a simple barplot using Seaborn. The title, X axis and Y axis were adjusted in order to facilitate understanding. The decision to set the x axis between 8.8 and 9.01 was taken because I wanted to ease the visualization of each rate.

Thinking about different relevant information the company could grab by simply looking at the chart a bit longer, I decided to implement new things. However, I wanted to keep the main question answered at first.

Initially, still using Seaborn, I gave meaning to the bar colors. In that chart the colors represent the number of users who rated each game. A legend box was added on the right.

In order to show different information, I decided to change the library to Altair. In that chart I wanted to show in addition to the top 5 games, the average of all the other games in the dataset. This information may be important to a company to have a general idea of the market of board games.

Finally, I decided to change the approach of how I was showing the top 5 games. By using points, I could use colors to include information about the category of the games. In addition to it, the size of each point was used to show the number of users who rated the game. Complementary, as I am using Altair, I decided to include a "hover functionality" in order to show extra information for each point in the graph. It is possible to see below an example of this functionality:



At the end, by using any of the charts, it is possible to have the question answered fast, and, if wished, it is also possible to have extra relevant information about the board game market.

4.2 Is there a correlation between the "users_rated" and the "max_playtime"?

In order to check the correlation between these two features, first I decided to check if there are any missing values in the columns. As it is possible to see after using info(), there aren't any missing values.

After that, I created a dataframe called correlation_check using only the features users_rated and max_playtime. It is possible to see these two steps below.

In [21]:
```
# Checking extra info, suck as missing values
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 [22]:
```
# creating a df
correlation_check = df.loc[:,['users_rated','max_playtime']]
correlation_check
```

Out[22]:

	users_rated	max_playtime
0	4498	240
1	478	30
2	12019	60
3	314	60
4	15195	90
...	...	...
10527	75	480
10528	82	45
10529	63	20
10530	341	120
10531	119	60

10532 rows × 2 columns

4.2.1 Checking the correlation - Tables

According to Hayes (2022), correlation can be described as a statistical term that explains the degree to which 2 variables move in coordination with each other. Mainly, it is possible to say that if the 2 variables move in the same direction, they have a positive correlation. Oppositely, the variables would have a negative correlation if they move in opposite directions. Regarding numbers, the correlation coefficient must fall between -1.0 and +1.0.

For this project, the function corr() is used to find correlation coefficient. In order to decide which is the best method to be used, I tested Spearman, Kendall and Pearson.

In [23]:
```
# Checking the correlation
correlation_check.corr(method='spearman')
```

Out[23]:

	users_rated	max_playtime
users_rated	1.000000	0.097013
max_playtime	0.097013	1.000000

In [24]:
```
# Checking the correlation
correlation_check.corr(method='kendall')
```

Out[24]:

	users_rated	max_playtime
users_rated	1.000000	0.069239
max_playtime	0.069239	1.000000

In [25]:
```
# Checking the correlation
correlation_check.corr(method='pearson')
```

Out[25]:

	users_rated	max_playtime
users_rated	1.000000	-0.004342
max_playtime	-0.004342	1.000000

4.2.2 Showing the correlation using a heatmap

As it is possible to confirm above, the correlation was not good in all 3 methods. For better visualization, this information will be displayed using a heatmap.

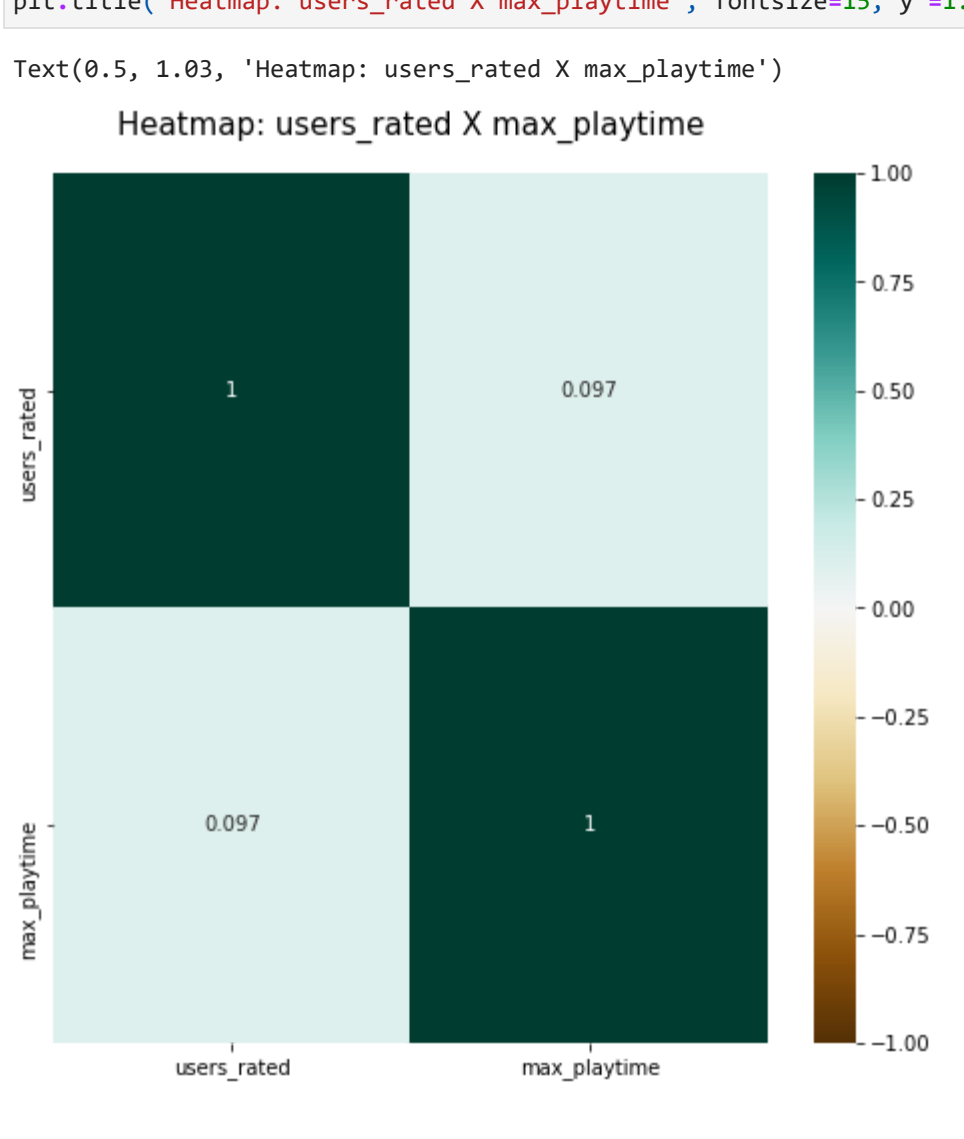
Those are the main implements I used:

- **Correlation method:** I am using the method='spearman' since it was the one that performed better.
- **figsize=(8, 8):** I increased the size in order to improve visualization.
- **vmin=-1, vmax=1:** I wanted to show the entire range of possible values for the correlation coefficient.
- **annot=True:** in order to show the correlation coefficient in the heatmap.
- **cmap='BrBG':** positive and negative correlations are also represented by the colors dark green and dark gold. The point 0.00 is in the middle in white.
- **Title:** I included a title in the heatmap and changed its size and position.

And this is the result:

In [26]:
```
# Checking variables' relation
f, ax = plt.subplots(1, 1, figsize=(8, 8))
ax = sns.heatmap(correlation_check.corr(method='spearman'), vmin=-1, vmax=1, annot=True, cmap='BrBG')
plt.title("Heatmap: users_rated X max_playtime", fontsize=15, y =1.03)
```

Out[26]: `Text(0.5, 1.03, 'Heatmap: users_rated X max_playtime')`



4.2.3 Changing the approach

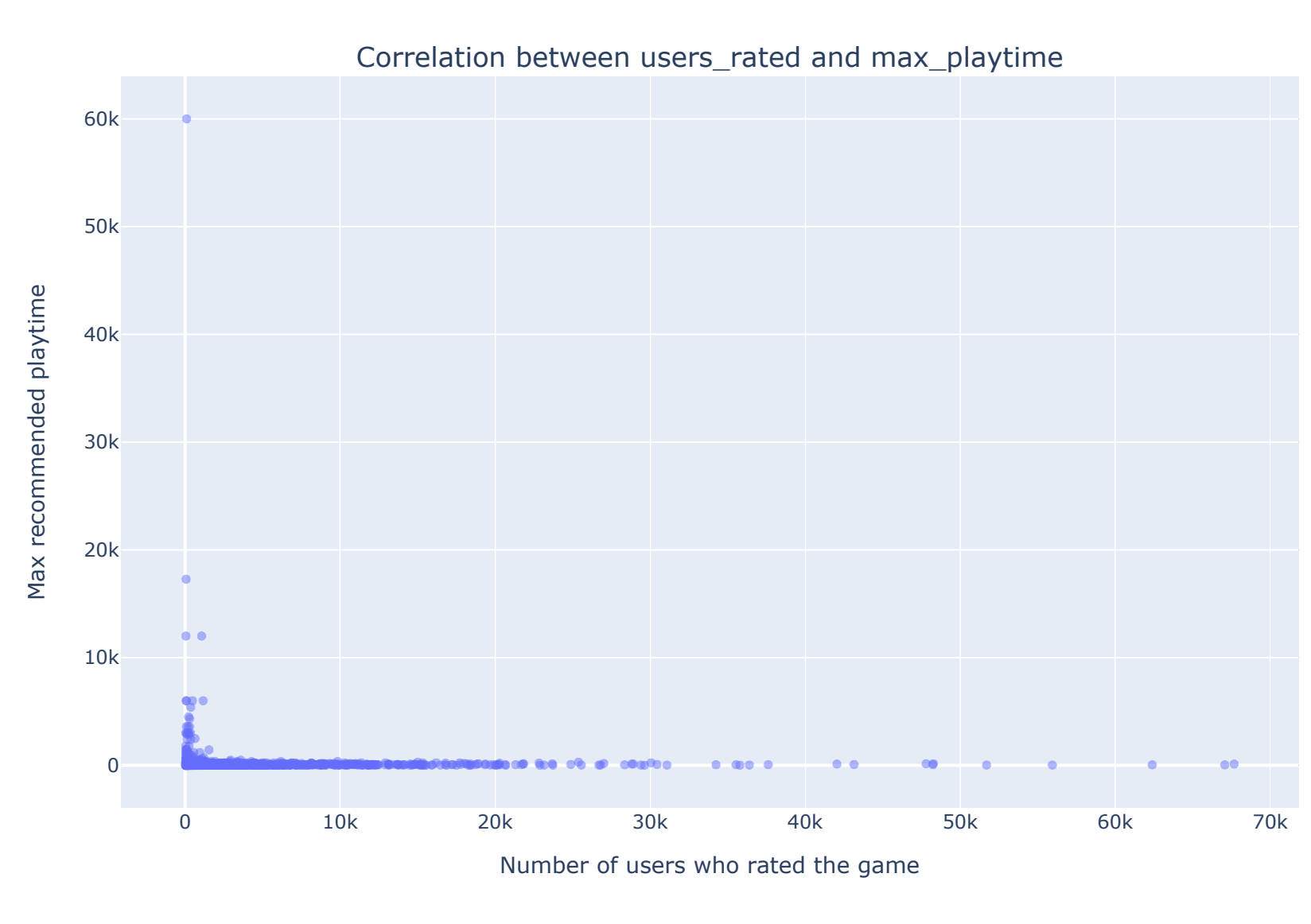
Another way to show correlation is by using scatter plots. Basically, when points move together along the X and Y axis, it may show correlation between these points. As confirmed above, there is not a good correlation between users_rated and max_playtime. It means that by using a scatter plot, the points will remain close to zero on the X or Y axis.

For better visualization, I decided to use Plotly Express for this example and take advantage of some tools. Those are the main implements I used:

- **opacity=0.5:** Since many points tend to be in the same spot or area, this may help to show the concentration of the points.
- **Width and height:** a bit bigger to ease the visualization.
- **Labels:** When hovering the points, it is possible to see this information. I changed the labels to ease the understanding. Also changes X and Y labels.
- **Title:** Title on top and in the center.

And this is the result:

In [27]:
```
# Plotting the Scatter plot
fig = px.scatter(correlation_check,
                 y= 'max_playtime',
                 x='users_rated',
                 opacity=0.5,
                 width=900,
                 height=600,

                 labels={
                     "max_playtime": "Max recommended playtime ",
                     "users_rated": "Number of users who rated the game ",
                        }
                )

fig.update_layout(
                  title={
                        'text': "Correlation between users_rated and max_playtime ",
                        'y':0.93,
                        'x':0.5,
                        'xanchor': 'center',
                        'yanchor': 'top'
                        }

                 )

fig.show()
```



4.2.4 Conclusion

In order to answer this question, it was necessary to statistically test the correlation between users_rated and max_playtime. So, my first decision was to create a variable called correlation_check with only these two features.

After that, I used the function corr() to check the correlation. In this function, I was able to test 3 different methods (Spearman, Kendall and Pearson). However, I could see as a result that users_rated and max_playtime are not correlated.

My next step was to plan a way to show this information. In my first approach, I decided to use a heatmap. I chose different and contrasting colors for +1.00 and -1.00 because I wanted extremes (positive correlation and negative correlation) to be very clear. I also changed the color scale to show +1.00 and -1.00 values.

In the heatmap, it is clear that the more white the color is, the weaker is the correlation. It is clear that the features are not highly correlated because the numbers are close to zero (0.00). In general, my intention with the heatmap was to show in a fast and easy way the correlation, and this can be done by checking the colors and/or the numbers.

After that I decided to try a different approach. In a scatter plot, my intention was to demonstrate that, if a point (or group of points) is far from the zero axis (X and Y), it could represent some correlation. This time I decided to use the Plotly Express library because of the extra functionalities I could implement.

In the scatter plot, as I knew many points would be concentrated on the zero lines (X and Y), I decided to change the opacity in order to ease visualization. I also decided to leave the hover so as to get more information about each point if needed.

At the end, the scatter plot, with all the points concentrated close to the zero lines (X and Y), would give to the company an easy understanding that the features users_rated and max_playtime are not correlated.

4.3 What is the distribution of game categories?

4.3.1 Checking the feature "category"

As it is possible to confirm in the data dictionary and below, the feature category is a string and has each category separated by commas.

The issue in this situation is that it is not possible to count each category without separating the categories in the same row.

In [28]:
```
df['category'].head(10)
```

Out[28]:
```
0    Economic,Negotiation,Political
1                Card Game,Fantasy
2         Abstract Strategy,Medieval
3                          Ancient
4                         Economic
5             Civilization,Nautical
6                Abstract Strategy
7             Civilization,Fantasy
8                      Exploration
9                   Fantasy,Travel
Name: category, dtype: object
```

4.3.2 Data Engineering - category

In order to solve the problem above, these are the following steps:

- **Str.split:** I am splitting each value separated by " , " and saving in a dataset called "cat".
- I created a variable where it is possible to see the name of each category and the number of times the category appears in each column. I am using **"apply(pd.value_counts)"** for this.
- Finally, I get the sum of each column using **"count_cat.sum(axis = 1)"**. With this step it is possible to see the name of each category and the number of times the categories appear in the entire feature.

This is the result after these steps:

In [29]:
```
# Data Engineering
cat = df['category'].str.split(',', expand=True)
cat.head()
```

Out[29]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	Economic	Negotiation	Political	None	None	None	None	None	None	None	None	None	None	None
1	Card Game	Fantasy	None	None	None	None	None	None	None	None	None	None	None	None
2	Abstract Strategy	Medieval	None	None	None	None	None	None	None	None	None	None	None	None
3	Ancient	None	None	None	None	None	None	None	None	None	None	None	None	None
4	Economic	None	None	None	None	None	None	None	None	None	None	None	None	None

In [30]:
```
# Data Engineering
count_cat = cat.apply(pd.value_counts)
count_cat.head(20)
```

Out[30]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Abstract Strategy	710	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Action / Dexterity	492	22.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Adventure	527	14.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Age of Reason	81	2.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
American Civil War	130	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
American Indian Wars	6	9.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
American Revolutionary War	7	23.0	6.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
American West	107	22.0	2.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Ancient	374	48.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Animals	436	132.0	4.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Arabian	44	11.0	2.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Aviation / Flight	159	6.0	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Bluffing	514	131.0	10.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Book	17	14.0	2.0	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Card Game	2143	738.0	96.0	4.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Children's Game	223	335.0	130.0	16.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
City Building	178	106.0	26.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Civil War	25	22.0	1.0	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Civilization	75	95.0	22.0	6.0	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Collectible Components	60	154.0	21.0	2.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [31]:
```
# Data Engineering
sum_count_cat = pd.DataFrame(count_cat.sum(axis = 1))
sum_count_cat
```

Out[31]:

	0
Abstract Strategy	710.0
Action / Dexterity	514.0
Adventure	541.0
Age of Reason	83.0
American Civil War	131.0
...	...
Wargame	2034.0
Word Game	229.0
World War I	117.0
World War II	676.0
Zombies	121.0

83 rows × 1 columns

- To continue, I decided to organize the categories, showing the ones that appear more first. For this step, **sort_values(by=0, ascending=False)** was used.

This is the final result:

In [32]:
```
# Data Engineering
organized_cat = sum_count_cat.sort_values(by=0, ascending=False)
organized_cat
```

Out[32]:

	0
Card Game	2981.0
Wargame	2034.0
Fantasy	1218.0
Fighting	900.0
Economic	878.0
...	...
Pike and Shot	27.0
Game System	22.0
American Indian Wars	15.0
Korean War	14.0
Expansion for Base-game	11.0

83 rows × 1 columns

4.3.3 Showing the distribution

In order to show the chart in the best way I decided to use Plotly Express. The reason for this is that with this library it is possible to select specific parts of the chart to zoom.

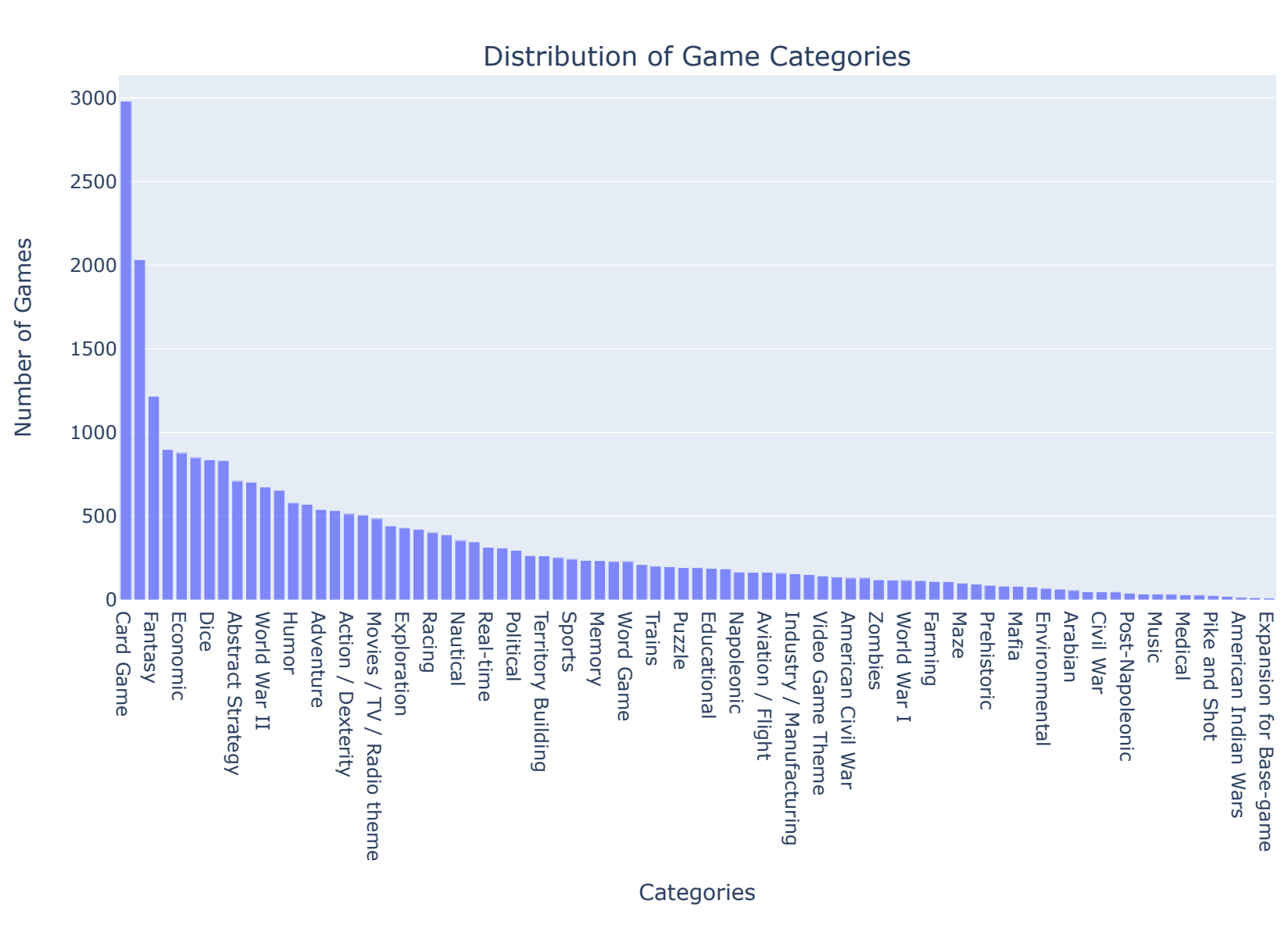
In total, there are 83 different categories. It is unnecessary to show all of them in the bar chart, because the main idea here is to show the distribution of the feature. However, as I am using Plotly Express, if I decided to check the top 5 categories per example, I just need to select that part of the chart and it will zoom in, showing the top categories.

For better visualization, those are the main implements I used:

- **Width and height:** a bit bigger to ease the visualization.
- **Labels:** When hovering the points, it is possible to see this information. I changed the labels to ease the understanding. Also changes X and Y labels.
- **Update_layout:** I am using "total descending" because I decided to show the most frequent category on the left.
- **opacity=0.8:** it looks better than vibrant colors.
- **Title:** Title on top and in the center.

And this is the final result:

In [33]:
```
# distribution of game categories

fig = px.bar(
            x=organized_cat.index,
            y=organized_cat[0],
            width=900,
            height=600,
            opacity=0.8,

            labels={"y":"Number of Games ", "x":"Categories "}
           )


fig.update_layout(xaxis={"categoryorder":"total descending"},

                  title={
                        'text': "Distribution of Game Categories ",
                        'y':0.93,
                        'x':0.5,
                        'xanchor': 'center',
                        'yanchor': 'top'
                        }


                 )

fig.show()
```



4.3.4 Conclusion

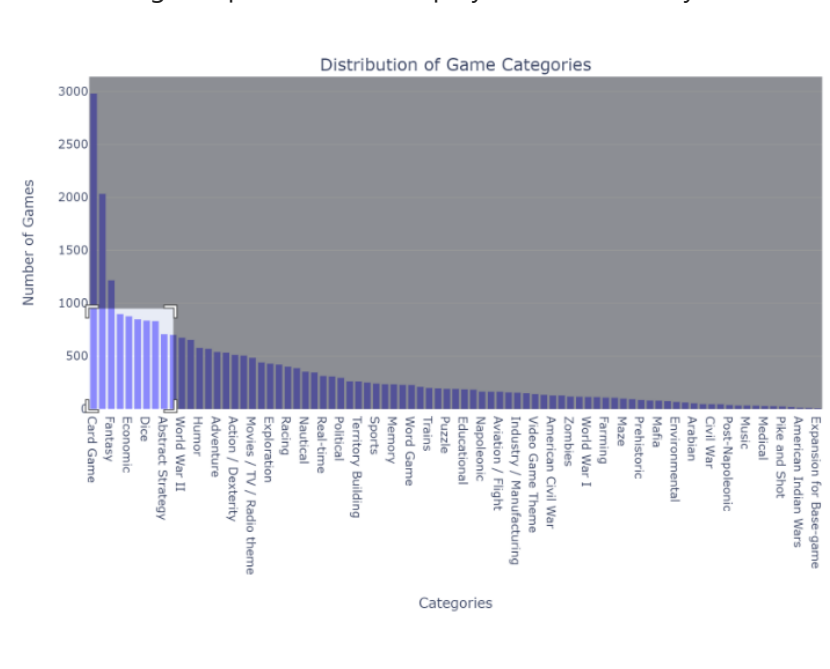
It was possible to see during the EDA and by checking the data dictionary, that the feature "category" had the categories separated by commas. Based on that, my first decision was to separate each category.

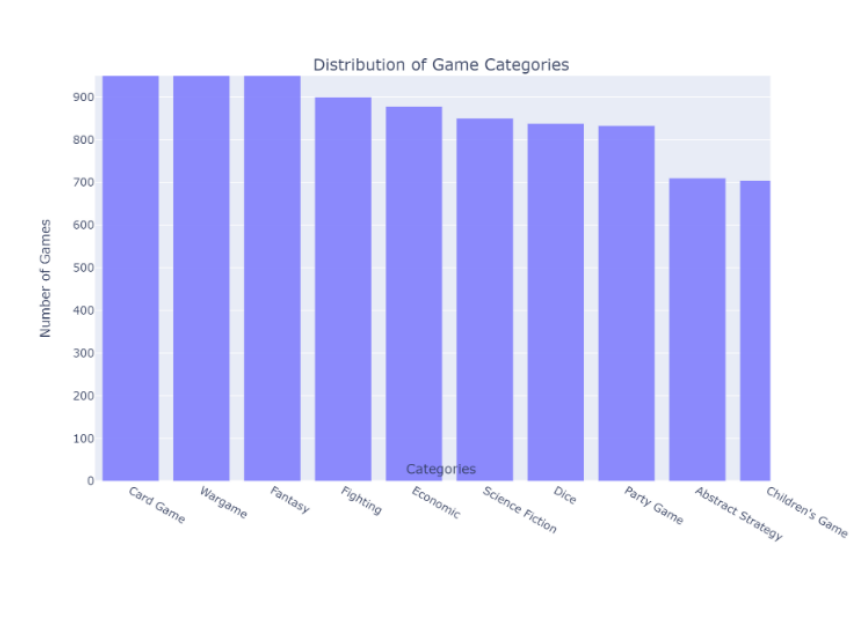
The feature has nominal data. Based on that, I decided to organize the categories by the frequency and the number of times each category is seen. By doing that, I could give to the company the information of the rank of the categories in the dataset. It may be important so as to know what are the most and the least popular categories in the dataset.

Regarding descriptive statistics, as the feature is nominal, I decided to use bar plots. By doing that, besides the distribution of the feature, I could also show the mode.

I chose to use the bar plots in the Plotly Express library. The main reason for that is that it is possible to select and zoom in specific areas of the distribution. In other words, if the company's intention is to have a general idea of the distribution, it is shown in the plot. However, if for some reason it is necessary to check the most frequent categories, it is possible to select just that part of the graph and get this information.

I am showing two pictures to exemplify this functionality below:





4.4 Do older games (1992 and earlier) have a higher MEAN "average rating" than newer games (after 1992)?

4.4.1 Data Engineering

In order to answer this question, I am using the features "year_published" and average_rating. As checked before, there are no missing values in thisse features.

First, let me divide the dataset in two parts, taking into consideration the year:

- **Old_games:** Games published in 1992 or before.
- **New_games:** Games published after 1992.

In [34]:
```
# Checking the dataset
df.head(2)
```

Out[34]:

	game_id	description	image	max_players	max_playtime	min_age	min_players	min_playtime	name	pla
0	1	Die Macher is a game about seven sequential po...	//cf.geekdo-images.com/images/pic159509.jpg	5	240	14	3	240	Die Macher	
1	2	Dragonmaster is a trick-taking card game based...	//cf.geekdo-images.com/images/pic184174.jpg	4	30	12	3	30	Dragonmaster	

2 rows × 22 columns

In [35]:
```
# checking some extra info
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 [36]:
```
# dividing the df
old_games = df.loc[(df['year_published'] <= 1992)]
old_games.head(2)
```

Out[36]:

	game_id	description	image	max_players	max_playtime	min_age	min_players	min_playtime	name	pla
0	1	Die Macher is a game about seven sequential po...	//cf.geekdo-images.com/images/pic159509.jpg	5	240	14	3	240	Die Macher	
1	2	Dragonmaster is a trick-taking card game based...	//cf.geekdo-images.com/images/pic184174.jpg	4	30	12	3	30	Dragonmaster	

2 rows × 22 columns

In [37]:
```
# dividing the df
new_games = df.loc[(df['year_published'] > 1992)]
new_games.head(2)
```

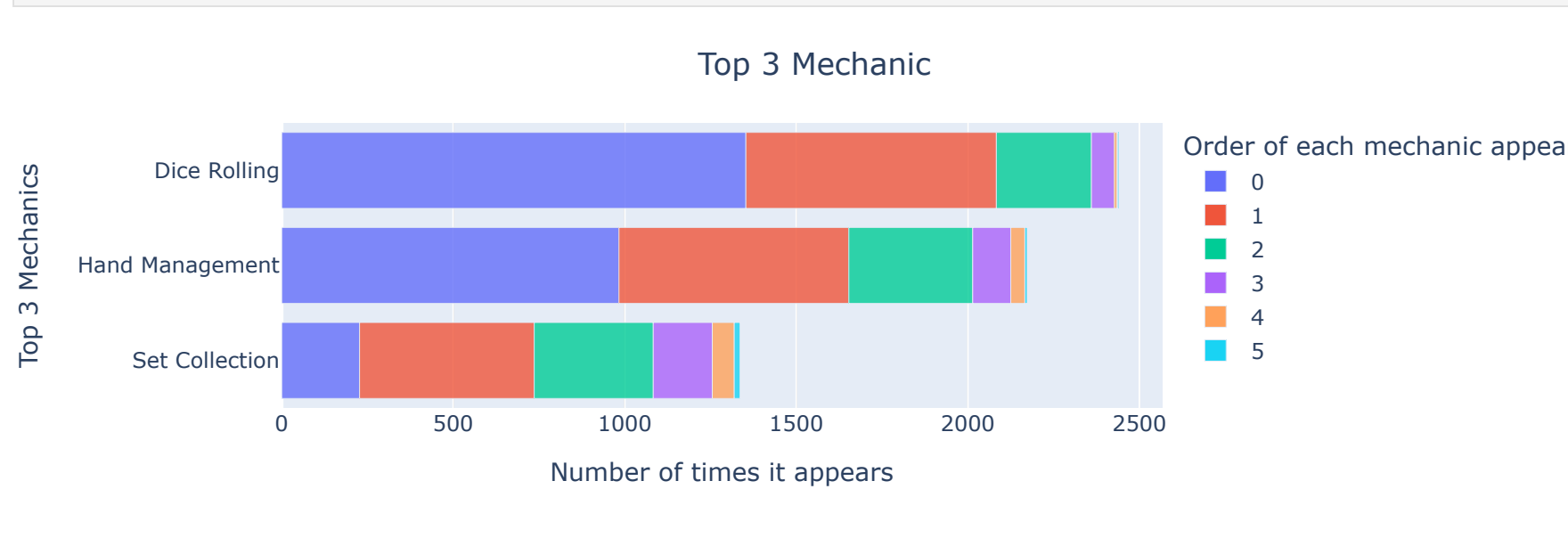
I decided to keep using the Plotly Express library to implement this idea.

For better visualization, those are the main implements I used:

- **Width and height:** a bit bigger to ease the visualization.
- **Hover:** When hovering the bars, it is possible to see the number of times that mechanic appeared in that position.
- **Update_traces:** I decided to change the way the hover is shown, making it simpler and smaller.
- **Update_layout:** I am using "total ascending" because I decided to show the most frequent mechanic on the top. Also changes X and Y labels. I included a title for the legend.
- **Legend:** It is possible to mark and unmark the positions. In case it is necessary to see, for example, just the second position or all of them together.
- **opacity=0.8:** looks better than vibrant colors.
- **Title:** the title is on top and in the center.

This is the final result:

```
In [53]:
# plotting
fig = px.bar(check_pos_top3,
            y=check_pos_top3.index,
            x=[0,1,2,3,4,5],
            opacity=0.8,
            width=900,
            height=300


           )

fig.update_layout(yaxis={"categoryorder":"total ascending"},

                 title={
                      'text': "Top 3 Mechanic ",
                      'y':0.93,
                      'x':0.5,
                      'xanchor': 'center',
                      'yanchor': 'top'
                      },

                 xaxis_title="Number of times it appears",
                 yaxis_title="Top 3 Mechanics",
                 legend_title="Order of each mechanic appears",



                )

fig.update_traces(hovertemplate=None)


fig.show()
```



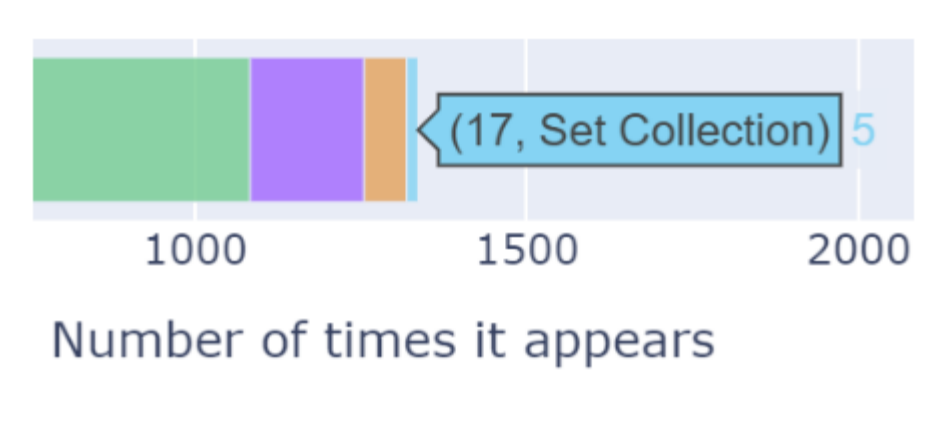
4.5.5 Conclusion

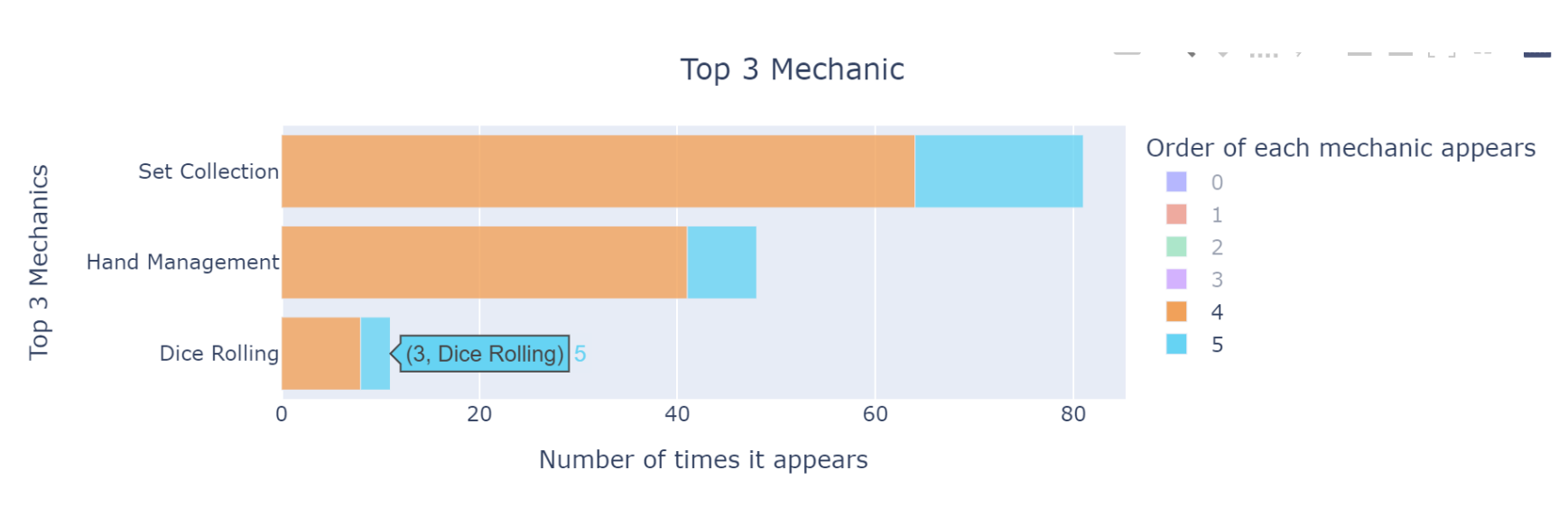
To have this question answered, as my first approach, I tried to use something that would be user friendly. By using a word cloud, the most common mechanics would be highlighted. Another advantage of this technique is that it is not necessary to organize the feature "mechanic" before dealing with it. It means that, with a simple plot, I could get the information needed. However, some implementations were done in the word cloud, such as, include a title, change the size and remove the axis lines.

For the second approach, it was necessary to have the feature "mechanic" organized, since the mechanics were separated by commas. After doing something similar to what was done with the feature "category" previously, I decided to separate only the top 3 mechanics in a variable called "top_3_mec".

For the second graph, I decided to simply plot 3 bars showing the top 3 mechanics. I decided to change the height in order to compact the information, since I am just plotting bars. I decided to include the number of times each mechanic appeared inside the bar because this extra information can be relevant for the company. As I am using Plotly Express, I included hover and I changed the opacity and the color of the bars. The idea was to easily show the top 3 mechanics with a bit of extra information if needed.

For the third graph, I decided to include even more information and interactivity to the bars on the second graph. In addition to the mechanic, I wanted to show the position the mechanic appeared in the description. After analyzing the feature, I could see the first 5 mechanics of each game were the main ones. With that in mind, in this bar plot, it is possible to select which position to show, mark and unmark them. After this, questions like "How many times Dice Rolling is described as the first mechanic of a game?" or "What is the most popular secondary mechanic of board games" can be answered. I also changed the way the hover is displayed, and, with Plotly Express, it is possible to select and zoom in specific parts of the bar plot. Some of this interactivity is displeyed below:





5. Part 2: a "Statistically Relevant" question

5.1 Data Understanding and Data Engineering

It is known that different games are released every year. Considering this, different categories and mechanics take turns as the most popular.

It is also known that even though some games have a really high average rate, the same game may not have been played and rated for many users.

Besides all this, there are really popular mechanics and categories that can not be played for some people because of the minimum age that is recommended to play the game.

Considering this, the statistically relevant question I decided to answer with the following chart is:

Taking into consideration the year and the number of users who rated the game, which are the top mechanics and categories for each recommended minimum age?

In order to answer this question, I decided to use different dimensions and the Altair library. Firstly, let me plan the chart:

- **The X axis:** I am using this to show the minimum recommended age. It is possible to follow the line and check all the categories and mechanics in each particular age.
- **The Y axis:** I am using this to show the average rate. It is easy to understand that a certain category or mechanic is better ranked than the other by simply checking its position.
- **Year selection:** It is possible to select the year using the bar control under the chart.
- **Color:** I am using different colors to show each category in the chart.
- **Shape:** I am using different shapes to show each mechanic in the chart.
- **Size:** I am using different sizes to represent the number of users who rated the game.

In order to show this amount of information in a single chart, first I am going to prepare the features Category and Mechanic. I am going to consider just the first category and mechanic for both of them, respectively.

```
In [54]:
# Preparing the plot
df_part2 = df
df_part2['first_mec'] = mec[0]
df_part2['first_cat'] = cat[0]
df_part2.head()
```

Out[54]:

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

5 rows × 24 columns

```
In [55]:
# Preparing the plot
df_part2["first_cat"].fillna("No Category informed", inplace = True)
df_part2["first_mec"].fillna("No Mechanic informed", inplace = True)
```

5.2 Showing the graph

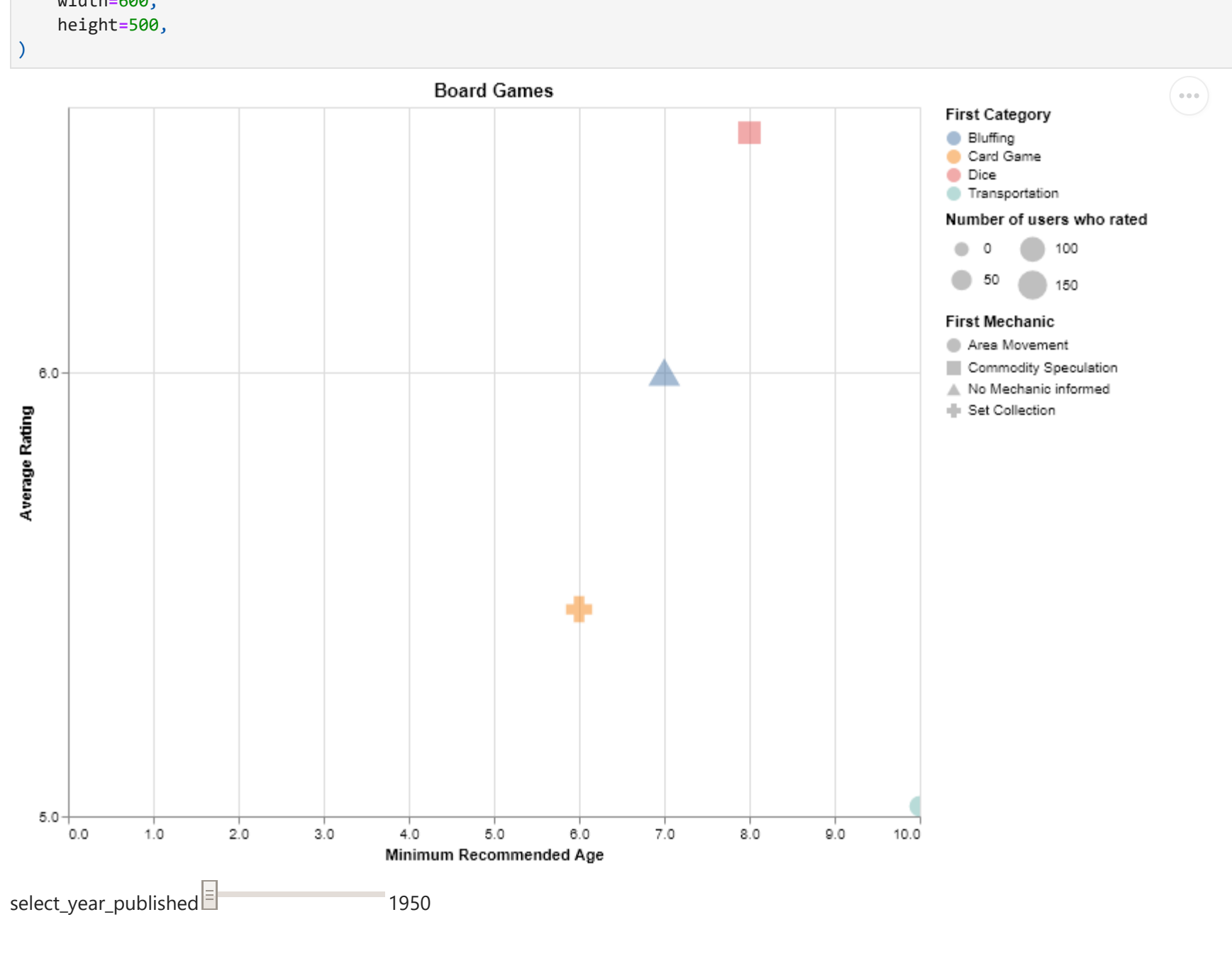
In addition to this, I am also implementing:

- **alt.Tooltip:** In order to be able hover each point and get all its information centralized.
- **interactive():** In order to be able to zoom specific areas in the chart by just scrolling. It is also possible to click and move the areas.
- **Titles:** X axis, Y axis and all the legends will have titles, in order to facilitate the understanding.
- **properties():** I included a title for the chart. I also changed the width and the height for better visualization.
- **Size scale:** I changed the size in because I wanted the games with lower users_rated to have a reasonable size in the chart.
- **Columns for the legends:** For better visualization, some legends will be shown in 1 or 2 columns.
- **symbolLimit:** No more than 8 symbols or color will be shown. This is important because I want to show the information (category, mechanic and users_rated) even if there lots of categories and mechanics.
- **X(alt.Axis):** It is set to the numbers I chose. Even when the chart is zoomed in, the x axis will remain the same. It is a good idea because min_age is integer. This shows the grid from the age zero to age 18.
- **Y(alt.Axis):** The rates go from 0 to 10. The fixed grid helps to visualize the rate of each game.

This is the final result:

```
In [56]:
# Plotting the graph - Part 2
select_year = alt.selection_single(
name='select', fields=['year_published'], init={'year_published': 1950},
bind=alt.binding_range(min=1950, max=2017, step=3)
)
alt.Chart(df_part2).mark_point(filled=True).encode(


alt.X('min_age',
      #scale=alt.Scale(zero=False),
      title='Minimum Recommended Age',
      #scale=alt.Scale(domain=[0, 10], nice=True)
      axis=alt.Axis(values=[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18])
     ),


alt.Y('average_rating',
      scale=alt.Scale(zero=False),
      title='Average Rating',
      axis=alt.Axis(values=[1,2,3,4,5,6,7,8,9,10])
     ),


alt.Size('users_rated:Q',
         scale=alt.Scale(range=[100, 500]),
         legend=alt.Legend(title="Number of users who rated", columns=2)
        ),


alt.Shape('first_mec:N',
          legend=alt.Legend(title="First Mechanic", columns=1, symbolLimit=8)
         ),


alt.Color('first_cat:N',
          legend=alt.Legend(title="First Category", columns=1, symbolLimit=8)
         ),


alt.Order('users_rated:Q',
          sort='descending'
         ),

alt.OpacityValue(0.5),

tooltip=[alt.Tooltip('first_cat:N'),
         alt.Tooltip('first_mec:N'),
         alt.Tooltip('average_rating:Q'),
         alt.Tooltip('min_age:Q'),
         alt.Tooltip('users_rated:Q'),
         alt.Tooltip('name:N'),

        ]


).interactive().add_selection(select_year).transform_filter(select_year).properties(
    title='Board Games',
    width=600,
    height=500,
)
```

Out[56]:



5.3 Conclusion

In this part the main idea was to transmit to the company more and more information as long as more time was spent analyzing the graph. Based on that, I chose to use Altair, because I wanted to take advantage of its functionality and interactivity.

Initially, it is clear that some games get a better average rate than others according to the minimum age. However, after analyzing more carefully the graph, lots of extra information can be seen. I am listing below some of the information this single graph can provide:

- some categories are better than others;
- some mechanics are better than others;
- some games are better than others;
- every year, some age groups have more games than others;
- categories and mechanics perform differently in different years;
- the number of users who rated the game influentiate in its average rate;
- the number of games published has increased year after year;
- the average rate of the games has increased year after year;
- more categories have been created over the years;
- more mechanics have been created over the years;
- the number of people who rate the game has increased over the years;
- proportionally, the number of games for teenagers and adults has increased over the years;
- and etc.

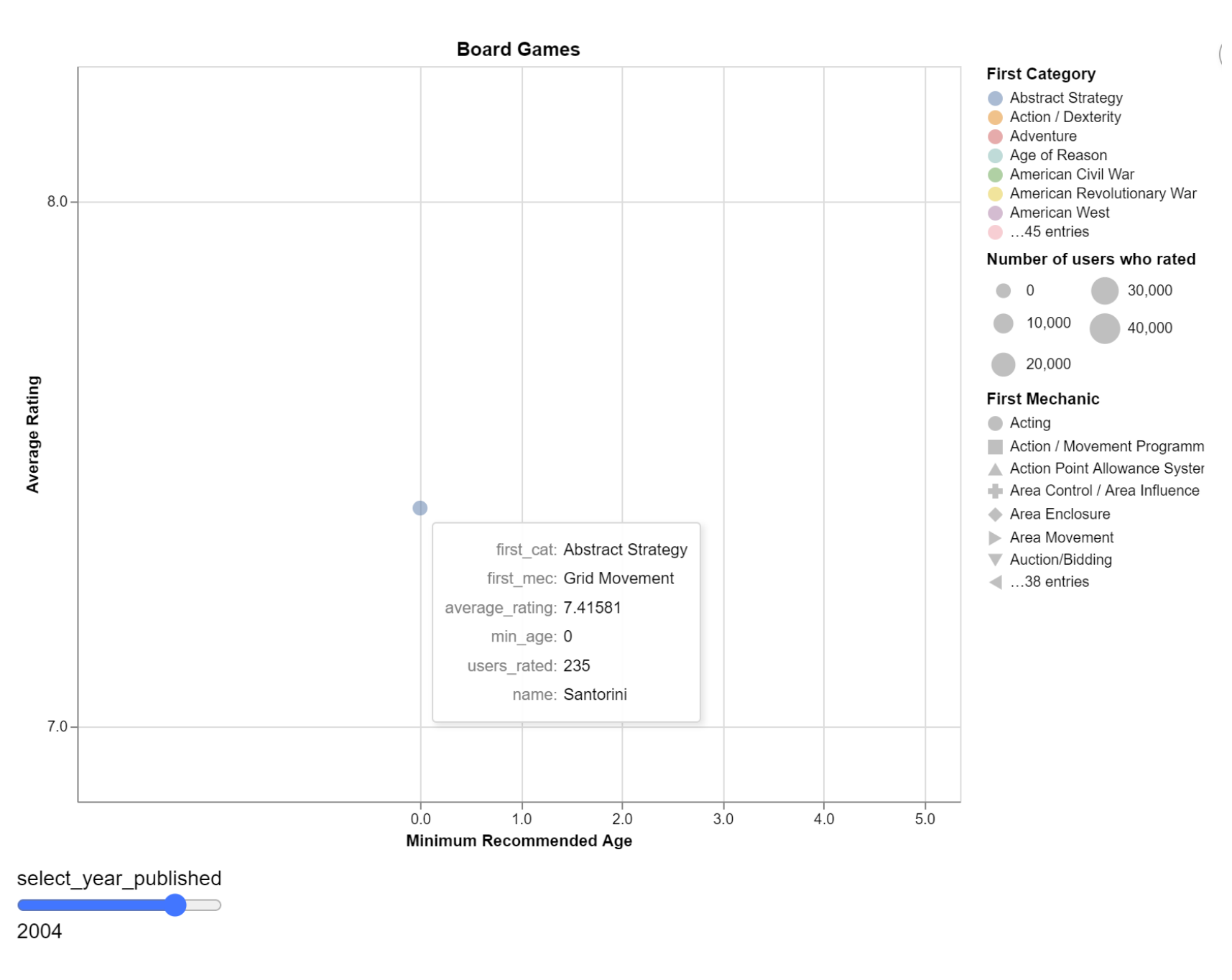
Another reason why I decided to use Altair is because it gives the chance of using interactivity, for example:

- By using hover, it is possible to highlight details of specific games;
- By using zoom, it is possible to highlight specific frames;

For instance, it is possible to frame a very specific game, doing things like:

- Adjust the average rating axis to show only games greater than rate 7;
- Adjust the age axis to show only games recommended to people younger than 5 years old;
- Adjust to see only games published in 2004;
- And etc.

The example above is shown below:



Finally, as described above, not only was it possible to answer the original question, but also, lots of extra relevant information can be seen in a single plot.

6. Part 3 (complement)

As it has been already explained, the part 3 of this project is related to rationalizing the decisions that I have made in each question. During the development of each particular graph, these details were already shown. In addition to it, each question has its own conclusion, which once again explains the thinking process and the rationale.

Here, I decided to explain and comment only about general aspects of data visualization. If the intention is to have a more detailed explanation, it can be found by reviewing Parts 1 and 2.

Types of visualization:

The main idea of data visualization is to be able to provide specific layouts that can transmit a great amount of information as fast as possible. In other words, the idea is to make it possible for the user to understand complicated data in a comfortable way.

According to Evergreen (2020), most of the information can be explained by using one of the types of graphs described below. During this project, several graphs were used. I am going to explain a bit more about the reason of their implementation:

- **Vertical bar chart:** they are a great option when it is necessary to compare things in a dataset since it is possible to visualize the data side by side.
- **Horizontal bar chart:** used to show ranking or performance. The best ones are on top.
- **Stacked bar chart:** used to show part of a whole, but still get the overview picture.
- **Histogram:** used to show distribution.
- **Scatter Plot:** They were used to show the correlation among the points and/or how things were related to each other.
- **Correlation Matrix (Heatmap):** in this project, it was used to show the correlation between two variables.

General design decisions:

Colors:

The palette colors structure was taken into consideration in order to generate better harmony in the charts. Regarding the structure, it is important to highlight their use in this project:

- **Sequential:** it was used when the intention was to show the idea of low and high of the data points. In these cases, the gradiente of the colors was used, where the light colors were used to show the small numbers and the dark colors were used to show the large numbers. In other words, the intention was to give a numerical meaning to the colors.
- **Divergent:** It was used when the intention with the data was to show two different groups. In the heatmap for example, a neutral meet point color was included.
- **Categorical:** in this case, the idea was to show that one thing was different from another thing. For this reason, when it was necessary, I avoided using gradients of the same colors to describe two or more different categories.

Apart from that, the selection of the color palettes for this project were done thinking about being color blind friendly. According to Cravit (2019), there are about 300 million color blind people worldwide.

Titles:

All the graphs have title, X and T labels. It is important because even if the graph is shown by itself, it will be possible to have a general idea of the data.

Size:

The size of the graphs was changed in order to be able to show all the information in an understandable and comfortable way.

Text:

The Typographic Design Systems used to choose the characteristics of the font in this project are called "One font/One Size" and "One font / Big header". According to France (2020), these options are considered good approaches when writing a business report or working with graphs.

7. Final Conclusion

No one can deny that the board games market has been increasing over the years. More and more games, with different mechanics and categories have been developed, making this market very popular among children, teenagers and adults.

In this project, I have cleaned and prepared a database in order to use different visualization libraries and functionalities. At the end of every question, I was able to show in different ways and using different approaches the information that was requested.

By studying and creating graphs, I was able to improve my knowledge regarding different libraries. The rationale behind every detail was carefully thought, in order to satisfy the standards of the good practice of data visualization.

Regarding libraries, in this project I had the opportunity to amplify my knowledge while I was planning and building every graph. During the planning stages, while the information I wanted to pass changed, I had to go deeper in the library's documentation to find specific functionalities.

During part 2, as I have already understood the main characteristics of the dataset, I was able to create and answer an important statistic relevant question. In addition to it, the final graph of this part of the project is able to give lots of extra information. Consequently, I had the opportunity to apply different techniques.

Regarding the scenario that was given in order to have this project completed, all the graphs are able to transmit important statistical information for the company. This information can help in the decision making process since the company can use the graphs to understand the market better, and, consequently, choose better strategies.

Finally, at the end of this project I am able to better understand the business and the dataset characteristics. This knowledge combined with the knowledge acquired during the research period of this project has certainly contributed to my career.

8. Reference List

Altair Developers (2020). Altair: Declarative Visualization in Python — Altair 4.1.0 documentation. [online] altair-viz.github.io. Available at: https://altair-viz.github.io/index.html [Accessed 20 Oct. 2022].

Bhutani, K. (2018). Python | Pandas dataframe.drop_duplicates(). [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/python-pandas-dataframe-drop_duplicates/ [Accessed 15 Oct. 2022].

Chandradas, A. (2021). 5 Methods to Check for NaN Values in Python. [online] Medium. Available at: https://towardsdatascience.com/5-methods-to-check-for-nan-values-in-in-python-3f21ddd17eed#:~:text=NaN%20stands%20for%20Not%20A [Accessed 15 Oct. 2022].

Chappelow, J. (2019). How Statistics Work. [online] Investopedia. Available at: https://www.investopedia.com/terms/s/statistics.asp [Accessed 27 Oct. 2022].

Cravit, R. (2019). How to Use Color Blind Friendly Palettes to Make Your Charts Accessible - Venngage. [online] Venngage. Available at: https://venngage.com/blog/color-blind-friendly-palette/ [Accessed 1 Nov. 2022].

Evergreen, S.D.H. (2020). Effective data visualization : the right chart for the right data. Thousand Oaks, California: Sage Publications, Inc.

France, T. (2020). Choosing Fonts for your Data Visualization. [online] Medium. Available at: https://medium.com/nightingale/choosing-a-font-for-your-data-visualization-2ed37afea637 [Accessed 1 Nov. 2022].

Frost, J. (2021). Five-Number Summary. [online] Statistics By Jim. Available at: https://statisticsbyjim.com/basics/five-number-summary/ [Accessed 28 Oct. 2022].

Hayes, A. (2022). What Is Correlation in Finance? [online] Investopedia. Available at: https://www.investopedia.com/terms/c/correlation.asp#:~:text=Correlation%20is%20a%20statistical%20term [Accessed 23 Oct. 2022].

Pandas (2018). Python Data Analysis Library — pandas: Python Data Analysis Library. [online] Pydata.org. Available at: https://pandas.pydata.org/ [Accessed 15 Oct. 2022].

Patil, P. (2018). What is Exploratory Data Analysis? [online] Towards Data Science. Available at: https://towardsdatascience.com/exploratory-data-analysis-8fc1cb20fd15 [Accessed 15 Oct. 2022].

Pearlman, S. (2018). What is Data Preparation? (+ How to Make It Easier) - Talend. [online] Talend Real-Time Open Source Data Integration Software. Available at: https://www.talend.com/resources/what-is-data-preparation/ [Accessed 21 Oct. 2022].

Plotly (2022). Modern Analytic Apps for the Enterprise - Plotly. [online] plotly.com. Available at: https://plotly.com/ [Accessed 31 Oct. 2022].

Plotly Developers (2022). Bar Charts. [online] plotly.com. Available at: https://plotly.com/python/bar-charts/ [Accessed 27 Oct. 2022].

Project Pro (2022). 10 Python Data Visualization Libraries to Win Over Your Insights. [online] ProjectPro. Available at: https://www.projectpro.io/article/python-data-visualization-libraries/543 [Accessed 20 Oct. 2022].

The Matplotlib development team (2022). Matplotlib: Python plotting — Matplotlib 3.3.4 documentation. [online] matplotlib.org. Available at: https://matplotlib.org/stable/index.html [Accessed 19 Oct. 2022].

Waskom, M. (2021). seaborn: statistical data visualization — seaborn 0.10.1 documentation. [online] seaborn.pydata.org. Available at: https://seaborn.pydata.org/index.html [Accessed 18 Oct. 2022].

Yi, M. (2021). A Complete Guide to Bar Charts. [online] Chartio. Available at: https://chartio.com/learn/charts/bar-chart-complete-guide/ [Accessed 26 Oct. 2022].


```
In [ ]:
```