Project Assignment 3 Report

Introduction:

The main objective of our Data Mining project is to build a recommendation system for Board games. We searched for relevant datasets on various data repository websites and found the Board Games Review dataset. It contains 3 csv files: reviews, detailed_info and Monthly update dataset. Upon further analysis we found out that the Monthly update dataset is just a subset of the detailed_info dataset. As the detailed_info dataset contains all the entries of the monthly update dataset, It is safe to discard the whole dataset.

We have worked on the detailed_info and the reviews dataset for this assignment. The detailed_info dataset contains various unique features like the game ID, game name, rank, max and min playingtime, max and min players, average, etc. It contains roughly 19k unique tuples and 20 feature columns. As it contains a lot of numeric attributes, most of the data understanding and exploration functions can be applied on it. The other dataset is the reviews dataset. It contains 4 features: game ID, user ID, rating and game name. It has roughly 15.8 million tuples. As it contains only one numeric attribute: rating, data exploration and data understanding functionalities are limited.

For this assignment, we have performed data understanding functionalities on our 2 datasets. We have further classified the data understanding into 3 categories. The categories are: Data exploration, Statistical Analysis and Data Visualization. Data exploration contains all the information related to dataset characteristics. Statistical analysis contains numeric functionalities calculations. Data Visualization contains the graphs and visual representation of the data.

Bgg-15m-reviews Dataset:

Data Exploration:

- We loaded the Bgg-15m-reviews in the pandas for data exploration. The dataset can be displayed using head() function which returns a sample number of tuples from the entire dataset.
- The sample of the dataset can be seen in the figure below. The exact count of rows and columns can be found using shape function.
- Detailed information regarding the datatypes of all the attributes can be listed easily using the info() function. Please see the attached images for more information.

```
#Loading in the first dataset
df = pd.read_csv('modified-bgg-15m-reviews.csv')
df.head()
```

	user	rating	ID	name
0	Torsten	10.0	30549	Pandemic
1	mitnachtKAUBO-I	10.0	30549	Pandemic
2	avlawn	10.0	30549	Pandemic
3	Mike Mayer	10.0	30549	Pandemic
4	Mease19	10.0	30549	Pandemic

```
# Basic information of the Dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15823269 entries, 0 to 15823268
Data columns (total 4 columns):
 #   Column  Dtype
---  ------  -----
 0   user    object
 1   rating  float64
 2   ID      int64
 3   name    object
dtypes: float64(1), int64(1), object(2)
memory usage: 482.9+ MB
```

In this dataset, the rating attribute is a float data type. It is the only attribute which can be used for statistical calculations.

Transaction Lists

- Below are transaction lists created for each username on BGG who has rated a game. A “transaction” is considered to be all games that that user has given the same rating. We will be able to use these lists to create a recommendation engine using apriori algorithms.

```
tens.groupby('user').name.apply(set)
```

```
user
 woh                                                    {Arimaa}
- V -                                                {Carcassonne}
-=Yod@=-                                {Gloomhaven, Twilight Struggle}
-DE-                                                {Neuroshima Hex! 3.0}
-Johnny-          {Risk 2210 A.D., Magic: The Gathering, Tigris ...
                                                  ...
zzzkardel                                                {T.I.M.E Stories}
zzzuzu                                             {Space Alert, Endeavor}
zzzxxxyyy            {Modern Art, Great Western Trail, Tic-Tac-Toe}
zzzzzane          {Agricola, Airlines Europe, Agricola (Revised ...
Æleksandr Þræð                                          {Mahjong, Chez Geek}
Name: name, Length: 202691, dtype: object
```

```
eights.groupby('user').name.apply(set)
```

```
user
 beastvol                                                {Domaine, Catan}
 mycroft          {Acquire, Betrayal at House on the Hill, War o...
 woh                                             {Bridge, Chess, Exxit}
-=Yod@=-          {Terraforming Mars, Arkham Horror: The Card Ga...
-Johnny-          {Hamsterrolle, Arkham Horror, 2 de Mayo, Witch...
                                                  ...
zzzuzu            {Agricola, Tikal, Dominion, Ubongo: Duel, Thur...
zzzvone                    {Ticket to Ride: Europe, Ticket to Ride}
zzzxxxyyy         {Scythe, Terraforming Mars, Viticulture Essent...
zzzzzane          {The Great Dalmuti, Shadows over Camelot, Troy...
Æleksandr Þræð                                {Hanafuda, Chez Goth, UNO}
Name: name, Length: 257429, dtype: object
```

```
sixes.groupby('user').name.apply(set)
```

```
user
 beastvol                                      {The Lord of the Rings}
-=Yod@=-          {Exit: The Game – The Secret Lab, Exit: The Ga...
-Johnny-          {Euphrates & Tigris: Contest of Kings, Stone A...
-Loren-           {Forbidden Desert, Trivial Pursuit: Genus Edit...
-LucaS-                                                  {Bohnanza}
                                                  ...
zzzkardel                                                {Loony Quest}
zzzuzu            {Stone Age, Prophecy, Saboteur, Roma, Cockroac...
zzzvone                                          {Tigris & Euphrates}
zzzzzane          {PitchCar, Triplock, Ghost Stories, Blood Bowl...
Æleksandr Þræð                                           {Hnefatafl}
Name: name, Length: 187988, dtype: object
```

```
fours.groupby('user').name.apply(set)
```

```
user
 mycroft                                      {Star Wars: Epic Duels}
-Johnny-     {Crazy Eights, Betrayal at House on the Hill, ...
-mIDE-       {Ocean, Casino Hot Dog, Jenga, Connect Four, C...
-toni-       {Crazy Eights, Spot it!, Risk: The Lord of the...
...Hammer    {Acquire, Glass Road, Power Grid, Splendor, Ca...
                                     ...
zzyzxuk      {Tragedy Looper, Dominion, The Resistance, Dom...
zzzabiss     {Onitama, Aloha, Arcana, Jenga, Monopoly Deal ...
zzzeagle     {Catan, Concept, Diamonds, Power Grid, Keyflower}
zzzuzu                                     {Race for the Galaxy}
zzzzzane                                             {Pick-a-Dog}
Name: name, Length: 93964, dtype: object
```

```
twos.groupby('user').name.apply(set)
```

```
user
-Johnny-     {Braggart, Building An Elder God, Spinergy, Th...
-mIDE-                                        {Skip-Bo, Poker Dice}
-mik-                                                 {Naval War}
...Hammer                          {Monopoly, Dark Darker Darkest}
01hejazi                                        {Monopoly Empire}
                                                  ...
zztap                                          {Super Farmer, Trouble}
zzyzxuk      {Automobile, The Resistance: Avalon, Troyes, D...
zzzabiss     {Don't Panic!, Tortilla de Patatas: The Game, ...
zzzuzu                                               {Dragonheart}
zzzvone                 {Lord of the Rings: The Confrontation}
Name: name, Length: 40254, dtype: object
```

```
nines.groupby('user').name.apply(set)
```

```
user
 Fu_Koios            {Shadowrift, Approaching Dawn: The Witching Hour}
 beastvol                                          {Caylus, Puerto Rico}
 mycroft                    {Descent: Journeys in the Dark, Catan}
(mostly) harmless                                      {Red November}
- V -                                                      {Citadels}
                                                  ...
zzzuzu               {Catan, Puerto Rico, Bohnanza, Battlestar Gala...
zzzvone                                                   {Alhambra}
zzzxxxyyy            {Ticket to Ride: Europe, Cosmic Encounter, The...
zzzzzane             {Viticulture, Lords of Waterdeep, Between Two ...
Æleksandr Þræð                          {Illuminati: New World Order}
Name: name, Length: 246004, dtype: object
```

```
sevens.groupby('user').name.apply(set)
```

```
user
 beastvol              {Catan Card Game, Ticket to Ride, Modern Art}
 mycroft          {Risk: Star Wars – The Clone Wars Edition, The...
- V -                                                      {Blue Moon}
-=Yod@=-          {Shadows over Camelot, Fireball Island, Machi ...
-Johnny-          {Magical Athlete, Ninety-Nine, Gambit Royale, ...
                                                  ...
zzzuzu            {Ribbit, Bananagrams, Ghost Blitz, Bausack, Ci...
zzzvone           {Alhambra: The Dice Game, Catan, Manhattan, Po...
zzzxxxyyy                    {Star Realms, Puerto Rico, Codenames}
zzzzzane          {Onitama, Two Rooms and a Boom, Robinson Cruso...
Æleksandr Þræð                        {Vampire, Mille Bornes, Senet}
Name: name, Length: 225701, dtype: object
```

```
fives.groupby('user').name.apply(set)
```

```
user
 beastvol                        {Candamir: The First Settlers}
-=Yod@=-      {Struggle of Empires, Exit: The Game – The Cat...
-Johnny-      {Witch of Salem, Blokus 3D, Kingdom Builder, S...
-LucaS-                                                 {Medina}
-Mal-         {Stratego: Lord of the Rings Trilogy Edition, ...
                                                  ...
zzzabiss      {Betrayal at House on the Hill, Timebomb, Junk...
zzzeagle      {Star Realms, Roll Through the Ages: The Bronz...
zzzuzu                                                   {Fauna}
zzzvone         {Angkor, The Gardens of the Alhambra, Citadels}
zzzzzane      {Mage Knight Board Game, Smash Up, Morels, Sma...
Name: name, Length: 137604, dtype: object
```

```
threes.groupby('user').name.apply(set)
```

```
user
 woh                                                        {Go}
-Johnny-     {Spot it!, Mixmo, Genesis, Mini Rails, Sid Mei...
-mIDE-       {Pick Up Sticks, Trivial Pursuit: Genus Editio...
-mik-               {G.E.V., Ogre, Twixt, Plague & Pestilence}
-toni-       {Da ist der Wurm drin, Dumm gelaufen!, Edison ...
                                     ...
zztap        {Monopoly, Zapp Zerapp, Choco, Draw Out, Explo...
zzyzxuk      {Escape: The Curse of the Temple, Upwords, Twi...
zzzabiss     {Heroes of Terrinoth, Chupacabra: Survive the ...
zzzeagle                                                {Scythe}
zzzvone                                                   {Risk}
Name: name, Length: 66633, dtype: object
```

```
ones.groupby('user').name.apply(set)
```

```
user
-mIDE-                                          {Tic-Tac-Toe}
0 1 1 2 3 5 8                                {Mice and Mystics}
00McCracken       {Exit: The Game – The Forgotten Island}
0201strong                    {Pandemic Legacy: Season 1}
096959                                        {Rat-a-Tat Cat}
                                                  ...
zzapper0                                                 {Citadels}
zzeroparticle                  {Guess Who?, Tic-Tac-Toe}
zzptichka                                          {Shark Mania}
zzzkardel                                                {Karuba}
zzzzzane                                           {Power Lunch}
Name: name, Length: 26738, dtype: object
```

Statistical Analysis:

Count, std, mean, min, max:

- We have used the describe function to list out the count, std, mean, min, 25%, 50%, 75% and max values for the dataset. The mean and median of the dataset are pretty close to each other and lie roughly around 7.

```
#using desribe function for stats calculation
df.describe()
```

	rating	ID
count	1.582e+07	1.582e+07
mean	7.055e+00	9.508e+04
std	1.600e+00	8.412e+04
min	1.401e-45	1.000e+00
25%	6.000e+00	1.213e+04
50%	7.000e+00	7.644e+04
75%	8.000e+00	1.678e+05
max	1.000e+01	3.140e+05

IQR

- Interquartile range is the next statistical function that we calculated. IQR for rating is 2. Also the 25% quartile value Q1 is 6 and 75% quartile value is 8. Hence we can calculate the minimum and maximum for rating. This will help us in removing the outliers.
- Minimum = Q1 - IQR*1.5 = 3
- Maximum = Q3 + IQR*1.5 = 11
- Hence, Values beyond Minimum and Maximum can be assumed to be Outliers and removed during the Data Cleaning process.

```
#Interquartile range
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1
IQR
```

```
rating          2.0
ID         155657.0
dtype: float64
```

Skew

```
#Skew
skew = df.skew()
skew
```

```
rating    -0.685
ID         0.403
dtype: float64
```

- We also calculated the skew value for the rating attribute. As you can see in the figure above, the value is -0.685. As the value lies between -0.5 to -1, the rating attribute can be considered as moderately skewed.

Median

- Median() function returns the median of the values. For example median rating of 7 indicates half of the dataset has rating values less than 7 and half over 7.

```
1  df2.median()
```

```
rating        7.0
ID        76444.0
dtype: float64
```

Kurtosis

- Kurtosis is a measure of the combined sizes of the two tails. It measures the amount of probability in the tails. The value is often compared to the kurtosis of the normal distribution, which is equal to 3.

```
#Kurtosis
#if the kurt>3 and skew!=0 then data is non normal (positive or negative outliers)
from scipy.stats import kurtosis
kurtosis(df['rating'])
```

```
1.0510481300936956
```

```
import matplotlib.pyplot as plt
import scipy.stats as stats
from scipy.stats import kurtosis
value = stats.shapiro(df['rating'].dropna())[1]
if value <= 0.05:
    print("Null hypothesis of normality is rejected.")
else:
    print("Null hypothesis of normality is accepted")
```

```
Null hypothesis of normality is rejected.
```

Cumsum, cummin, cummax, cumprod:

- Cumsum is a cumulative sum of partial sums of an attribute which is not useful in our data frame. This also can be applied to Cummin and Cummax and cumprod.

- Although we can implement it, it doesn't give out any useful information. Following is the implementation:

```
[13]: reviews_new['rating'].cummin()
      #returns cumulative minimum over row rating. Current value will be compared with previous value and minimum will be foumd
```

```
[13]: 0           1.000000e+01
      1           1.000000e+01
      2           1.000000e+01
      3           1.000000e+01
      4           1.000000e+01
                      ...
      15823264    1.401300e-45
      15823265    1.401300e-45
      15823266    1.401300e-45
      15823267    1.401300e-45
      15823268    1.401300e-45
      Name: rating, Length: 15823269, dtype: float64
```

```
[14]: reviews_new['rating'].cummax()
      #returns cumulative max over row ratings.
```

```
[14]: 0           10.0
      1           10.0
      2           10.0
      3           10.0
      4           10.0
                  ...
      15823264    10.0
      15823265    10.0
      15823266    10.0
      15823267    10.0
      15823268    10.0
      Name: rating, Length: 15823269, dtype: float64
```

```
reviews_new['rating'].cumprod()
#cumulative prof for previous values
```

```
0                10.0
1               100.0
2              1000.0
3             10000.0
4            100000.0
                ...
15823264          inf
15823265          inf
15823266          inf
15823267          inf
15823268          inf
Name: rating, Length: 15823269, dtype: float64
```

Done only on rating here, as it is the only numeric attribute.

Diff and pct_change

- diff() - Calculates the difference of a Dataframe element compared with another element in the Dataframe (default is element in previous row).

pct_change() - Percentage change between the current and a prior element. Computes the percentage change from the immediately previous row by default.

```
reviews_new['rating'].diff()
# diff from previous values,
# first row has NaN because no previous value to diff from
# second is subtracted from first row value and so on
# by default, calculations over row, can be column intstead
```

```
0           NaN
1           0.0
2           0.0
3           0.0
4           0.0
           ...
15823264   -1.0
15823265    0.0
15823266   -1.0
15823267   -1.0
15823268   -5.0
Name: rating, Length: 15823269, dtype: float64
```

```
reviews_new['rating'].pct_change()
#Computes the percentage change from the immediately previous row by default.
#This is useful in comparing the percentage of change in a time series of elements.
```

```
0                NaN
1           0.000000
2           0.000000
3           0.000000
4           0.000000
              ...
15823264   -0.111111
15823265    0.000000
15823266   -0.125000
15823267   -0.142857
15823268   -0.833333
Name: rating, Length: 15823269, dtype: float64
```

Argmin, argmax, idxmin, idxmax

- Argmin and argmax are used to compute index locations at which minimum and maximum values are obtained and idxmin and idxmax are used to compute index labels at which minimum and maximum values are obtained.
- Since the arg and idx functions are almost identical , we will compute idxmin and idxmax. Also argmin and argmax take array values as input and not directly a csv file.
- The first step will be to use only numeric columns from the dataset. We then group these values using Name and User column

```
df2 = pd.read_csv('Final Dataset/modified-bgg-15m-reviews.csv')
df2.head()
```

	user	rating	ID	name
0	Torsten	10.0	30549	Pandemic
1	mitnachtKAUBO-I	10.0	30549	Pandemic
2	avlawn	10.0	30549	Pandemic
3	Mike Mayer	10.0	30549	Pandemic
4	Mease19	10.0	30549	Pandemic

```
df2_numeric = df2.set_index(['name','user']).select_dtypes('number')
df2_numeric
```

		rating	ID
name	user		
Pandemic	Torsten	10.0	30549
	mitnachtKAUBO-I	10.0	30549
	avlawn	10.0	30549
	Mike Mayer	10.0	30549
	Mease19	10.0	30549
	cfarrell	10.0	30549
	katrinacarenne	10.0	30549
	DSpangler	10.0	30549
	gregd	10.0	30549
	calbearfan	10.0	30549
	odustin	10.0	30549
	treece keenes	10.0	30549

- We then calculate the index locations where minimum and maximum values exist. We can also use the highlight option to highlight the biggest minimum values (the lowest values) in the rating column.
- In similar ways we can also calculate the maximum value for the rating column.
- We have grouped those values using the game name and user. So in the below output for minimum , the game ‘Tigris & Euphrates’ is been given the lowest rating by user ‘GeoffMack’
- For maximum values, the game ‘Pandemic’ has been rating highest by user ‘Torsten’

```
idx2_min = df2_numeric.idxmin()
idx2_min
```

```
rating    (Tigris & Euphrates, GeoffMack)
ID                (Die Macher, jackcres)
dtype: object
```

```
df2_idx2_min = df2_numeric.loc[idx2_min]
df2_idx2_min
```

			rating	ID
	name	user		
Tigris & Euphrates		GeoffMack	1.401300e-45	42
Die Macher		jackcres	1.000000e+01	1

```
df2_idx2_min.reset_index(drop=True).style.highlight_min()
```

	rating	ID
0	1.4013e-45	42
1	10	1

```
idx2_max = df2_numeric.idxmax()
idx2_max
```

```
rating                    (Pandemic, Torsten)
ID        (Pandemic Legacy: Season 0, SandroF)
dtype: object
```

```
df2_idx2_max = df2_numeric.loc[idx2_max]
df2_idx2_max
```

		rating	ID
name	user		
Pandemic	Torsten	10.0	30549
Pandemic Legacy: Season 0	SandroF	10.0	314040

```
df2_idx2_max.reset_index(drop=True).style.highlight_max()
```

	rating	ID
0	10	30549
1	10	314040

Chi-Square Test

- Chi-Square test cannot be applied to this dataset since we do not have two categorical attributes in this dataset. We have applied similar tests to the second datasets.

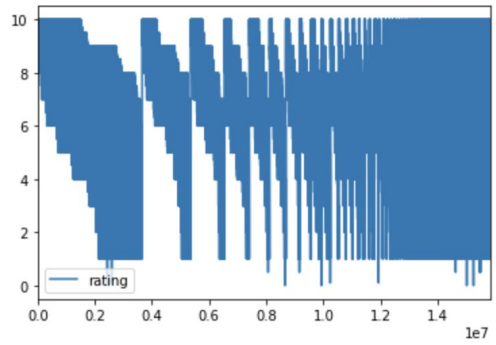
Data Visualization:

Line Plot

- A line graph, also known as a line chart, is a type of chart used to visualize the value of something over the other one (which is usually time).
- Data points are plotted and connected by a line.
- The line graph above indicates the rating values till 10. As you can see in the above graph the values are distributed over 0-10 ratings

```
1 df_temp2.plot(df2.index.name,['rating'],kind='line')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f99719440f0>
```

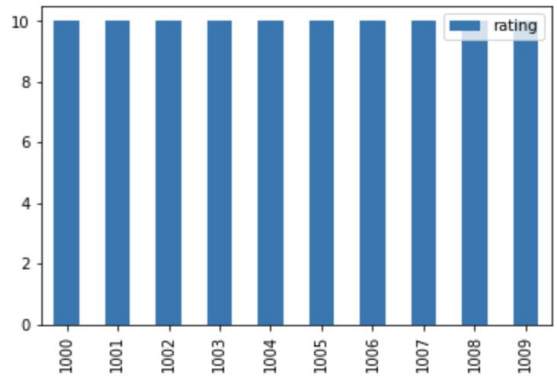


Bar Graph

- A bar graph uses bars to compare data among categories.
- Bar graph presents categorical data with rectangular bars with heights or lengths proportional to the values that they represent.
- This is one of the biggest dataset so the above bar graph just represents 10 values hence all of them have the same values

```
1 df_temp3.plot(df2.index.name,['rating'],kind='bar')
```

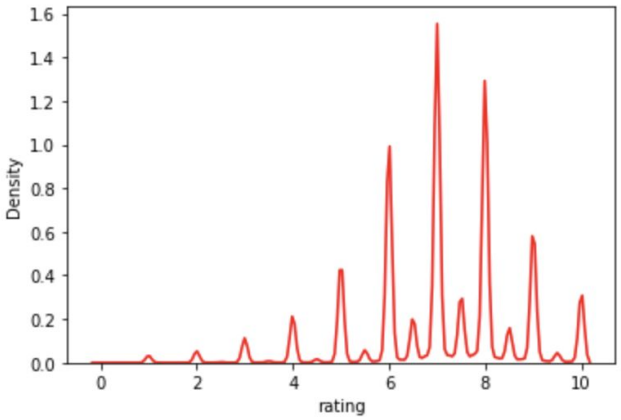
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f997192aeb8>
```



Kernel density estimate

```
#kdeplot skewness graph plot
sns.kdeplot(df['rating'], color = 'r')
```

```
<AxesSubplot:xlabel='rating', ylabel='Density'>
```



- We used the sns library to plot the skewness graph for rating attribute. As the graph has a tail towards the left side, It is negatively skewed.
- The mean of the rating attribute is close to 7 which can be seen in the KDE plot as well.
- Outliers lie beyond 3 and 11 as calculated through IQR. They can be safely removed in the Data cleaning stage.

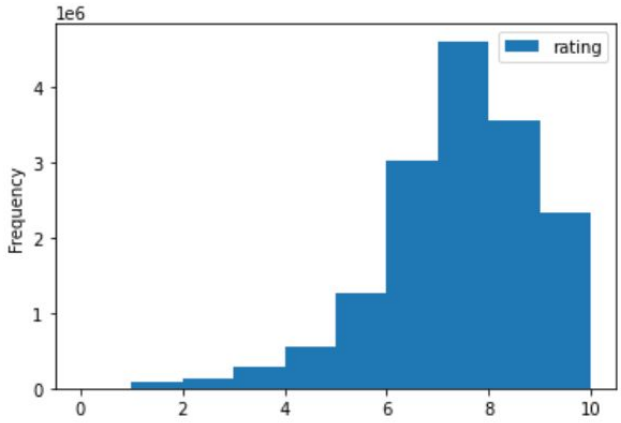
Histogram

- Histograms are a type of bar plot for numeric data that group the data into bins
- A frequency distribution shows how often each different value in a set of data occurs.
- A histogram is the most commonly used graph to show frequency distributions.

```
import matplotlib.pyplot as plt
#Histogram

df.plot(df.index.name,[1],kind="hist")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x138835a57f0>
```

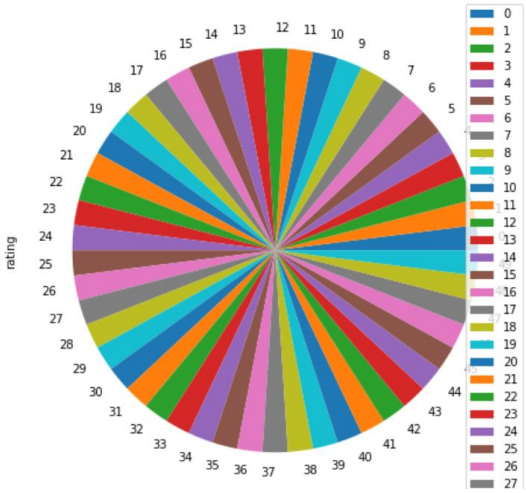


- The Histogram plot indicates the rating values till 10. As you can see in the plot rate 7 has the most frequency and rate 1 has the least frequency

Pie Plot

- Pie charts are generally used to show percentage or proportional data and usually the percentage represented by each category is provided next to the corresponding slice of pie.
- Pie charts are good for displaying data for around 6 categories or fewer.

```
df1= df[:50]
plot = df1.plot.pie(y='rating', figsize=(8,8))
```



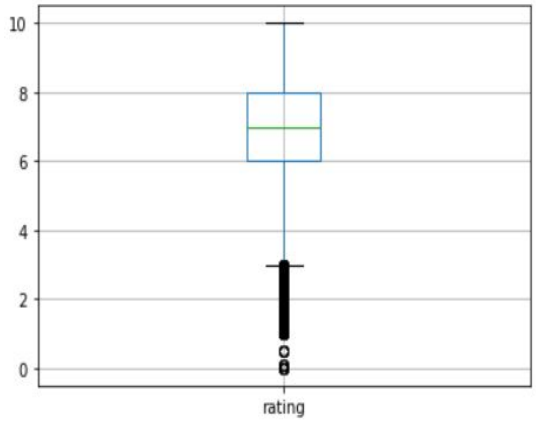
- Because we do not have categorical data, Pie plot is not meaningful here.

Boxplot

- Boxplot is very useful in finding ranges in which data lies, and also find outliers.
- Below graph shows boxplot for this dataset. It is evident that most ratings lie between 6-8 and others that lie around 2 are the outliers.

```
boxplot = reviews_new[:].boxplot(column=['rating'])
boxplot
```

```
<AxesSubplot:>
```



Games_detailed_info Dataset:

Data Exploration:

We loaded the Games_detailed_info Dataset in the pandas library for data exploration. The dataset can be displayed using head() function which returns a sample number of tuples from the entire dataset. The sample of the dataset can be seen in the figure below. The exact count of rows and columns can be found using shape function. Detailed information regarding the datatypes of all the attributes can be listed easily using the info() function. Please see the attached images for more information.

```
#Loading in the second dataset
df = pd.read_csv('modified-games_detailed_info.csv')
df.head()
```

	id	primary	description	yearpublished	minplayers	maxplayers	playingtime	minplaytime	maxplaytime	minage	boardgamecategory	boardgamen
0	30549	Pandemic	In Pandemic, several virulent diseases have br...	2008	2	4	45	45	45	8	['Medical']	['Actio 'Cooperativ 'H
1	822	Carcassonne	Carcassonne is a tile-placement game in which ...	2000	2	5	45	30	45	7	['City Building', 'Medieval', 'Territory Build...	['Area Influen Ad
2	13	Catan	In Catan (formerly The Settlers of Catan), pla...	1995	3	4	120	60	120	10	['Economic', 'Negotiation']	['Dic 'Hexag 'Incom
3	68448	7 Wonders	You are the leader of one of the 7 great citie...	2010	2	7	30	30	30	10	['Ancient', 'Card Game', 'City Building', 'Civ...	['Card 'Draftin Mana
4	36218	Dominion	"You are a monarch, like your parents bef...	2008	2	4	30	30	30	13	['Card Game', 'Medieval']	['Deck, Bag, Building',

```
# Basic information of the Dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19230 entries, 0 to 19229
Data columns (total 20 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   id                  19230 non-null  int64
 1   primary             19230 non-null  object
 2   description         19229 non-null  object
 3   yearpublished       19230 non-null  int64
 4   minplayers          19230 non-null  int64
 5   maxplayers          19230 non-null  int64
 6   playingtime         19230 non-null  int64
 7   minplaytime         19230 non-null  int64
 8   maxplaytime         19230 non-null  int64
 9   minage              19230 non-null  int64
 10  boardgamecategory   19009 non-null  object
 11  boardgamemechanic   17681 non-null  object
 12  boardgamefamily     14743 non-null  object
 13  boardgamedesigner   18800 non-null  object
 14  boardgameartist     13833 non-null  object
 15  boardgamepublisher  19230 non-null  object
 16  usersrated          19230 non-null  int64
 17  average             19230 non-null  float64
 18  bayesaverage        19230 non-null  float64
 19  Board Game Rank     19230 non-null  int64
dtypes: float64(2), int64(10), object(8)
memory usage: 2.9+ MB
```

- The dataset contains 20 attributes as shown in the figure. Info() function gives the non-null count for all the attributes. The function also states the DType of all the attributes.
- Primary and description are the nominal attributes. They contain text information. The same goes for attributes #10 to #15. These attributes cannot be used to perform statistical analysis. We will be working with the other attributes that are numeric in nature.
- Some of the unique attributes in this dataset are yearpublished, max and min players for the game, max and min playing time of the game, max and min age of the players, etc. We will be performing the statistical analysis and data visualization on these chosen attributes.

Statistical Analysis:

Count, std, mean, min,max

- Using the describe function, the 8 main statistical analysis properties can be easily listed. Count, std, mean, min, 25%, 50%, 75% and max values are important in finding other complex statistical attributes.

```
#For only printing upto 3 decimal values
set_option('precision', 3)
#using desribe function for stats calculation
df.describe()
```

	id	yearpublished	minplayers	maxplayers	playingtime	minplaytime	maxplaytime	minage	usersrated	average	bayesaverage	Board Game Rank
count	19230.000	19230.000	19230.000	19230.000	19230.000	19230.000	19230.000	19230.000	19230.000	19230.000	19230.000	19230.000
mean	101482.306	1984.084	2.029	5.618	96.595	66.623	96.595	9.603	822.717	6.393	5.690	9646.835
std	94272.241	212.855	0.686	15.372	1025.842	473.585	1025.842	3.631	3379.272	0.930	0.370	5588.107
min	1.000	-3500.000	0.000	0.000	0.000	0.000	0.000	0.000	30.000	1.048	3.536	1.000
25%	10213.250	2000.000	2.000	4.000	30.000	20.000	30.000	8.000	56.000	5.807	5.511	4808.250
50%	72758.500	2010.000	2.000	4.000	45.000	30.000	45.000	10.000	120.000	6.420	5.550	9618.500
75%	182275.500	2016.000	2.000	6.000	90.000	60.000	90.000	12.000	381.750	7.011	5.688	14485.750
max	314040.000	2021.000	10.000	999.000	120000.000	60000.000	120000.000	25.000	96241.000	9.688	8.569	19345.000

IQR and Skew

```
#Interquartile range
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1
IQR
```

```
id                 172062.250
yearpublished          16.000
minplayers              0.000
maxplayers              2.000
playingtime            60.000
minplaytime            40.000
maxplaytime            60.000
minage                  4.000
usersrated            325.750
average                 1.204
bayesaverage            0.177
Board Game Rank      9677.500
dtype: float64
```

```
#Skew
skew = df.skew()
skew
```

```
id                   0.448
yearpublished      -11.183
minplayers           1.715
maxplayers          44.558
playingtime         94.695
minplaytime        109.102
maxplaytime         94.695
minage              -0.856
usersrated          12.151
average             -0.298
bayesaverage         2.773
Board Game Rank      0.009
dtype: float64
```

- The IQR for this dataset gives a list of values. The Quartile values of 25% and 75% can be taken from the describe function. After the IQR has been calculated, Min and Max values can be calculated to remove Outliers from the dataset.
- This step will be useful in the Data Cleaning stage. The Skew value for all the numeric attributes has also been calculated.
- The range of skew values falls under parameters. Average attribute is slightly skewed as it comes between 0.0 to 0.5 numeric range.
- Minage attribute is moderately skewed as it lies between the 0.5 to 1.0 numeric range. Rest of the attributes are highly skewed as they mostly lie beyond the 1.0 numeric range.

Covariance

```
#Covariance
set_option('precision', 3)
covariance = df.cov()
covariance
```

	id	yearpublished	minplayers	maxplayers	playingtime	minplaytime	maxplaytime	minage	usersrated	average	bayesaverage	
id	8.887e+09	1.953e+06	-6155.694	23790.532	-8.336e+05	-2.339e+06	-8.336e+05	35176.900	-5.275e+06	34541.845	4409.763	-1
yearpublished	1.953e+06	4.531e+04	-0.739	-10.241	1.126e+03	5.024e+02	1.126e+03	65.469	5.907e+03	16.347	4.537	-9
minplayers	-6.156e+03	-7.386e-01	0.470	0.869	6.916e+00	1.477e+01	6.916e+00	0.128	-1.740e+01	-0.097	-0.017	3
maxplayers	2.379e+04	-1.024e+01	0.869	236.300	-5.459e+01	-4.279e+01	-5.459e+01	0.366	-8.687e+01	-0.526	-0.097	1
playingtime	-8.336e+05	1.126e+03	6.916	-54.594	1.052e+06	2.287e+05	1.052e+06	121.880	-1.241e+04	60.283	3.066	-1
minplaytime	-2.339e+06	5.024e+02	14.774	-42.790	2.287e+05	2.243e+05	2.287e+05	59.629	-8.018e+03	17.085	0.857	-5
maxplaytime	-8.336e+05	1.126e+03	6.916	-54.594	1.052e+06	2.287e+05	1.052e+06	121.880	-1.241e+04	60.283	3.066	-1
minage	3.518e+04	6.547e+01	0.128	0.366	1.219e+02	5.963e+01	1.219e+02	13.187	8.830e+02	0.641	0.258	-4
usersrated	-5.275e+06	5.907e+03	-17.405	-86.868	-1.241e+04	-8.018e+03	-1.241e+04	883.016	1.142e+07	544.713	793.848	-5
average	3.454e+04	1.635e+01	-0.097	-0.526	6.028e+01	1.709e+01	6.028e+01	0.641	5.447e+02	0.865	0.168	-3
bayesaverage	4.410e+03	4.537e+00	-0.017	-0.097	3.066e+00	8.570e-01	3.066e+00	0.258	7.938e+02	0.168	0.137	-1
Board Game Rank	-1.236e+08	-9.610e+04	341.289	1747.822	-1.476e+05	-5.013e+04	-1.476e+05	-4305.541	-5.909e+06	-3855.252	-1478.477	3

- The above figure shows the covariance of all the numeric attributes. Covariance provides us a relation between two attributes.
- More precisely, covariance refers to the measure of how two random attributes in a dataset will change together.
- A positive covariance means that the two attributes are positively related, and they move in the same vector direction.
- A negative covariance means that the attributes are inversely related, or that they move in opposite directions.

Correlation

```
#Pairwise Pearson Correlations
set_option('precision', 3)
correlations = df.corr(method='pearson')
correlations
```

	id	yearpublished	minplayers	maxplayers	playingtime	minplaytime	maxplaytime	minage	usersrated	average	bayesaverage	Board Game Rank
id	1.000	0.097	-0.095	0.016	-0.009	-0.052	-0.009	0.103	-0.017	0.394	0.126	-0.235
yearpublished	0.097	1.000	-0.005	-0.003	0.005	0.005	0.005	0.085	0.008	0.083	0.058	-0.081
minplayers	-0.095	-0.005	1.000	0.082	0.010	0.045	0.010	0.051	-0.008	-0.152	-0.068	0.089
maxplayers	0.016	-0.003	0.082	1.000	-0.003	-0.006	-0.003	0.007	-0.002	-0.037	-0.017	0.020
playingtime	-0.009	0.005	0.010	-0.003	1.000	0.471	1.000	0.033	-0.004	0.063	0.008	-0.026
minplaytime	-0.052	0.005	0.045	-0.006	0.471	1.000	0.471	0.035	-0.005	0.039	0.005	-0.019
maxplaytime	-0.009	0.005	0.010	-0.003	1.000	0.471	1.000	0.033	-0.004	0.063	0.008	-0.026
minage	0.103	0.085	0.051	0.007	0.033	0.035	0.033	1.000	0.072	0.190	0.192	-0.212
usersrated	-0.017	0.008	-0.008	-0.002	-0.004	-0.005	-0.004	0.072	1.000	0.173	0.634	-0.313
average	0.394	0.083	-0.152	-0.037	0.063	0.039	0.063	0.190	0.173	1.000	0.487	-0.742
bayesaverage	0.126	0.058	-0.068	-0.017	0.008	0.005	0.008	0.192	0.634	0.487	1.000	-0.714
Board Game Rank	-0.235	-0.081	0.089	0.020	-0.026	-0.019	-0.026	-0.212	-0.313	-0.742	-0.714	1.000

- Correlation is often used as a preliminary technique to discover relations between the numeric attributes of the dataset. More precisely, the correlation is a measure of the linear relationship between two attributes.
- The figure above shows the Pearson's correlation coefficient measure between all the attributes. Pairwise Pearson Correlation is +1 means that there is Total Positive Linear Correlation. Similarly, when Pairwise Pearson Correlation is -1, it means that there is Total Negative Linear Correlation.

Kurtosis

- Kurtosis is a measure of the combined sizes of the two tails. It measures the amount of probability in the tails. The value is often compared to the kurtosis of the normal distribution, which is equal to 3.

```
#if the kurt>3 and skew!=0 then data is non normal (positive or negative outliers)
from scipy.stats import kurtosis
kurtosis(info['yearpublished'])
```

```
148.55803627287017
```

```
kurtosis(info['Board Game Rank'])
```

```
-1.1998666070513748
```

- Kurtosis here indicates there are some outliers in the "yearpublished" column which is because of the vast range of the games. These outliers can be deleted in the data cleaning phase.

Cumsum, cummin, cummax, cumprod

- The cumulative sum, min, max and prod, calculated over the previous row values is shown below. This is performed again on only numeric rows. So games_numeric is the rows extracted from the main dataframe.

- Numeric Data extracted

```
games_info_numeric = games_info[['minplayers','maxplayers','playingtime','minplaytime','maxplaytime','minage','average','bayesaverage','Board G
games_info_numeric
```

	minplayers	maxplayers	playingtime	minplaytime	maxplaytime	minage	average	bayesaverage	Board Game Rank
0	2	4	45	45	45	8	7.61567	7.51795	91
1	2	5	45	30	45	7	7.41884	7.31105	173
2	3	4	120	60	120	10	7.16265	7.00045	381
3	2	7	30	30	30	10	7.76049	7.66214	51
4	2	4	30	30	30	13	7.62671	7.52038	89
...	...	...	...	...	...	...	...	...	...
19225	2	4	0	0	0	7	5.08333	5.48989	16940
19226	2	5	30	15	30	10	6.71667	5.51472	13942
19227	3	8	20	20	20	8	5.16667	5.49304	16652
19228	2	2	25	15	25	8	6.20333	5.51469	13944
19229	2	4	90	60	90	12	9.13333	5.52186	12884

19230 rows × 9 columns

- Cumsum, cummin and cumprod computed

```
games_info_numeric.cumsum()
```

	minplayers	maxplayers	playingtime	minplaytime	maxplaytime	minage	average	bayesaverage	Board Game Rank
0	2	4	45	45	45	8	7.61567	7.51795	91
1	4	9	90	75	90	15	15.03451	14.82900	264
2	7	13	210	135	210	25	22.19716	21.82945	645
3	9	20	240	165	240	35	29.95765	29.49159	696
4	11	24	270	195	270	48	37.58436	37.01197	785
...	...	...	...	...	...	...	...	...	...
19225	39007	108020	1857353	1281049	1857353	184628	122903.49409	109394.45490	185451215
19226	39009	108025	1857383	1281064	1857383	184638	122910.21076	109399.96962	185465157
19227	39012	108033	1857403	1281084	1857403	184646	122915.37743	109405.46266	185481809
19228	39014	108035	1857428	1281099	1857428	184654	122921.58076	109410.97735	185495753
19229	39016	108039	1857518	1281159	1857518	184666	122930.71409	109416.49921	185508637

19230 rows × 9 columns

```
games_info_numeric.cummin()
```

	minplayers	maxplayers	playingtime	minplaytime	maxplaytime	minage	average	bayesaverage	Board Game Rank
0	2	4	45	45	45	8	7.61567	7.51795	91
1	2	4	45	30	45	7	7.41884	7.31105	91
2	2	4	45	30	45	7	7.16265	7.00045	91
3	2	4	30	30	30	7	7.16265	7.00045	51
4	2	4	30	30	30	7	7.16265	7.00045	51
...	...	...	...	...	...	...	...	...	...
19225	0	0	0	0	0	0	1.04844	3.53593	1
19226	0	0	0	0	0	0	1.04844	3.53593	1
19227	0	0	0	0	0	0	1.04844	3.53593	1
19228	0	0	0	0	0	0	1.04844	3.53593	1
19229	0	0	0	0	0	0	1.04844	3.53593	1

19230 rows × 9 columns

```
games_info_numeric.cumprod()
```

	minplayers	maxplayers	playingtime	minplaytime	maxplaytime	minage	average	bayesaverage	Board Game Rank
0	2	4	45	45	45	8	7.615670e+00	7.517950e+00	91
1	4	20	2025	1350	2025	56	5.649944e+01	5.496411e+01	15743
2	12	80	243000	81000	243000	560	4.046857e+02	3.847735e+02	5998083
3	24	560	7290000	2430000	7290000	5600	3.140559e+03	2.948188e+03	305902233
4	48	2240	218700000	72900000	218700000	72800	2.395213e+04	2.217150e+04	27225298737
...	...	...	...	...	...	...	...	...	...
19225	0	0	0	0	0	0	inf	inf	0
19226	0	0	0	0	0	0	inf	inf	0
19227	0	0	0	0	0	0	inf	inf	0
19228	0	0	0	0	0	0	inf	inf	0
19229	0	0	0	0	0	0	inf	inf	0

19230 rows × 9 columns

Diff and pct_change

- Diff performed on consecutive rows

```
games_info_numeric.diff()
```

	minplayers	maxplayers	playingtime	minplaytime	maxplaytime	minage	average	bayesaverage	Board Game Rank
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	0.0	1.0	0.0	-15.0	0.0	-1.0	-0.19683	-0.20690	82.0
2	1.0	-1.0	75.0	30.0	75.0	3.0	-0.25619	-0.31060	208.0
3	-1.0	3.0	-90.0	-30.0	-90.0	0.0	0.59784	0.66169	-330.0
4	0.0	-3.0	0.0	0.0	0.0	3.0	-0.13378	-0.14176	38.0
...	...	...	...	...	...	...	...	...	...
19225	0.0	-4.0	-120.0	-120.0	-120.0	-1.0	0.32333	0.00541	-443.0
19226	0.0	1.0	30.0	15.0	30.0	3.0	1.63334	0.02483	-2998.0
19227	1.0	3.0	-10.0	5.0	-10.0	-2.0	-1.55000	-0.02168	2710.0
19228	-1.0	-6.0	5.0	-5.0	5.0	0.0	1.03666	0.02165	-2708.0
19229	0.0	2.0	65.0	45.0	65.0	4.0	2.93000	0.00717	-1060.0

19230 rows × 9 columns

- Pct_change on consecutive rows

```
games_info_numeric.pct_change()
```

	minplayers	maxplayers	playingtime	minplaytime	maxplaytime	minage	average	bayesaverage	Board Game Rank
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	0.000000	0.250000	0.000000	-0.333333	0.000000	-0.125000	-0.025845	-0.027521	0.901099
2	0.500000	-0.200000	1.666667	1.000000	1.666667	0.428571	-0.034532	-0.042484	1.202312
3	-0.333333	0.750000	-0.750000	-0.500000	-0.750000	0.000000	0.083466	0.094521	-0.866142
4	0.000000	-0.428571	0.000000	0.000000	0.000000	0.300000	-0.017239	-0.018501	0.745098
...	...	...	...	...	...	...	...	...	...
19225	0.000000	-0.500000	-1.000000	-1.000000	-1.000000	-0.125000	0.067926	0.000986	-0.025485
19226	0.000000	0.250000	inf	inf	inf	0.428571	0.321313	0.004523	-0.176978
19227	0.500000	0.600000	-0.333333	0.333333	-0.333333	-0.200000	-0.230769	-0.003931	0.194377
19228	-0.333333	-0.750000	0.250000	-0.250000	0.250000	0.000000	0.200644	0.003941	-0.162623
19229	0.000000	1.000000	2.600000	3.000000	2.600000	0.500000	0.472327	0.001300	-0.076018

19230 rows × 9 columns

Argmin, argmax, idxmin, idxmax

```
df1_numeric = df1.set_index(['primary','yearpublished']).select_dtypes('number')
df1_numeric.head()
```

primary	yearpublished	id	minplayers	maxplayers	playingtime	minplaytime	maxplaytime	minage	usersrated	average	bayesaverage	Board Game Rank
Pandemic	2008	30549	2	4	45	45	45	8	96241	7.61567	7.51795	91
Carcassonne	2000	822	2	5	45	30	45	7	96228	7.41884	7.31105	173
Catan	1995	13	3	4	120	60	120	10	96213	7.16265	7.00045	381
7 Wonders	2010	68448	2	7	30	30	30	10	79873	7.76049	7.66214	51
Dominion	2008	36218	2	4	30	30	30	13	74959	7.62671	7.52038	89

```
idx1_max = df1_numeric.idxmax()
idx1_max
```

```
id                                  (Pandemic Legacy: Season 0, 2020)
minplayers                                          (Haggle, 1963)
maxplayers                                (Scrimish Card Game, 2015)
playingtime                             (1985: Under an Iron Sky, 2018)
minplaytime          (The Campaign for North Africa: The Desert War...
maxplaytime                             (1985: Under an Iron Sky, 2018)
minage                                     (Martinis & Men, 2007)
usersrated                                      (Pandemic, 2008)
average                                          (Aeolis, 2020)
bayesaverage                                  (Gloomhaven, 2017)
Board Game Rank                             (Tic-Tac-Toe, -1300)
dtype: object
```

- The below output highlights the maximum and minimum values for each column.

```
df1_idx1_max.reset_index(drop=True).style.highlight_max()
```

	id	minplayers	maxplayers	playingtime	minplaytime	maxplaytime	minage	usersrated	average	bayesaverage	Board Game Rank
0	314040	2	4	60	45	60	14	98	6.20508	5.53925	10684
1	17529	10	100	300	120	300	10	52	6.67542	5.53648	11001
2	189890	2	999	100	10	100	8	227	5.95132	5.54576	10013
3	236650	2	6	120000	120	120000	16	73	9.16301	5.60165	6808
4	4815	8	10	60000	60000	60000	14	143	6.01476	5.51165	14359
5	236650	2	6	120000	120	120000	16	73	9.16301	5.60165	6808
6	28308	2	8	20	20	20	25	76	5.00066	5.47301	18007
7	30549	2	4	45	45	45	8	96241	7.61567	7.51795	91
8	281257	1	7	120	60	120	18	32	9.6875	5.51875	13367
9	174430	1	4	120	60	120	14	37098	8.8199	8.56858	1
10	11901	2	2	1	1	1	4	3108	2.66461	3.53593	19345

```
1 df1_idx1_min.reset_index(drop=True).style.highlight_min()
```

	id	minplayers	maxplayers	playingtime	minplaytime	maxplaytime	minage	usersrated	average	bayesaverage	Board Game Rank
0	1	3	5	240	240	240	14	5090	7.63112	7.14996	266
1	21804	0	0	0	0	0	0	848	6.59683	5.8906	2898
2	21804	0	0	0	0	0	0	848	6.59683	5.8906	2898
3	21804	0	0	0	0	0	0	848	6.59683	5.8906	2898
4	21804	0	0	0	0	0	0	848	6.59683	5.8906	2898
5	21804	0	0	0	0	0	0	848	6.59683	5.8906	2898
6	38996	2	2	90	90	90	0	2328	7.63725	6.80345	573
7	1624	2	7	60	60	60	10	30	5.47333	5.4977	16174
8	144110	2	4	90	90	90	12	64	1.04844	5.32385	19211
9	11901	2	2	1	1	1	4	3108	2.66461	3.53593	19345
10	174430	1	4	120	60	120	14	37098	8.8199	8.56858	1

Median

- The median indicates the middle point of the column. In the below output we can see the median value for the year published is 2010 , so we can infer that half of the games in the dataset are published before 2010 and half after 2010.
- Similarly the average median is 6.4 , so half of the games are rated less 6.4 and half of them greater than 6.4

```
1 df1.median()
```

```
id                72758.50000
yearpublished      2010.00000
minplayers            2.00000
maxplayers            4.00000
playingtime          45.00000
minplaytime          30.00000
maxplaytime          45.00000
minage               10.00000
usersrated          120.00000
average               6.42035
bayesaverage          5.55031
Board Game Rank    9618.50000
dtype: float64
```

Chi - Square Test, ANOVA Test

- Chi - Square test is used to find correlation between two categorical attributes. Since we do not have any such requirement, we have instead used ANOVA test (Analysis of Variance Test)

- ANOVA ("analysis of variance") compares the means of two or more independent groups in order to determine whether there is statistical evidence that the associated population means are significantly different.
- The ANOVA test requires
    - Dependent variable that is continuous (i.e., interval or ratio level) - Board Game Rank
    - Independent variable that is categorical - Game category

- We first need to install statsmodel module and import it

```
conda install -c conda-forge statsmodels
```

```
Collecting package metadata (current_repodata.json): done
Solving environment: done

# All requested packages already installed.


Note: you may need to restart the kernel to use updated packages.
```

```
import statsmodels.api as sm
from statsmodels.formula.api import ols
```

- The null and alternative hypotheses of one-way ANOVA can be expressed as:
    - H0: $\mu 1 = \mu 2 = \mu 3 = ... = \mu k$ ("all k population means are equal")
    - H1: At least one μi different ("at least one of the k population means is not equal to the others") where μi is the population mean of the ith group (i = 1, 2, ..., k)

```
#One way anova
#Relationship between minimum players and boardgamecategory
mod1 = ols('Q("Board Game Rank")~boardgamecategory',data=df1).fit()
aov1 = sm.stats.anova_lm(mod1,type=2)
print(aov1)
```

```
                      df        sum_sq       mean_sq         F  \
boardgamecategory   6103.0  2.833568e+11  4.642909e+07  1.927502
Residual           12905.0  3.108517e+11  2.408770e+07       NaN

                       PR(>F)
boardgamecategory  6.751757e-209
Residual                    NaN
```

```
#Two way anova
mod2 = ols('Q("Board Game Rank")~minplayers+maxplayers',data=df1).fit()
aov2 = sm.stats.anova_lm(mod2,type=2)
print(aov2)
```

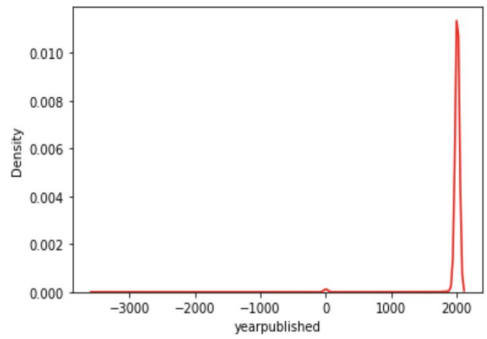
```
                 df        sum_sq       mean_sq           F        PR(>F)
minplayers      1.0  4.763175e+09  4.763175e+09  153.764198  3.554601e-35
maxplayers      1.0  1.023040e+08  1.023040e+08    3.302563  6.918743e-02
Residual    19227.0  5.955974e+11  3.097714e+07         NaN           NaN
```

- You can see the value in the last column is less than < 0.05, hence we reject the null hypothesis that all the data come from populations with the same mean.

Data Visualization:

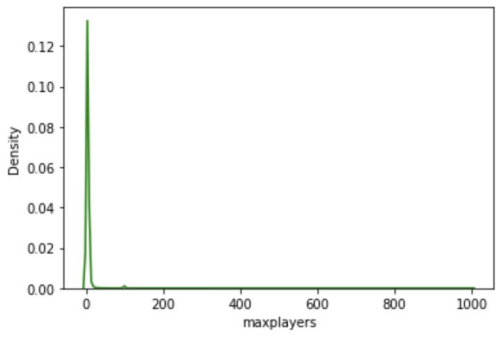
```
#kdeplot skewness graph plot
sns.kdeplot(df['yearpublished'], color = 'r')
```

```
<AxesSubplot:xlabel='yearpublished', ylabel='Density'>
```



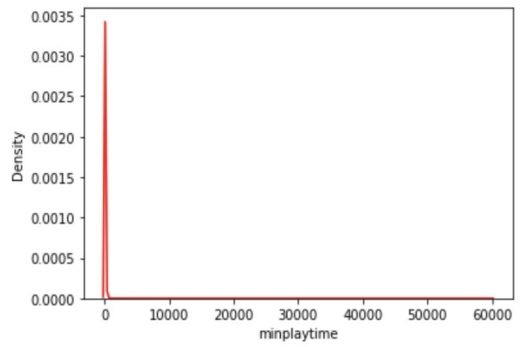

```
#kdeplot skewness graph plot
sns.kdeplot(df['maxplayers'], color = 'g')
```

```
<AxesSubplot:xlabel='maxplayers', ylabel='Density'>
```



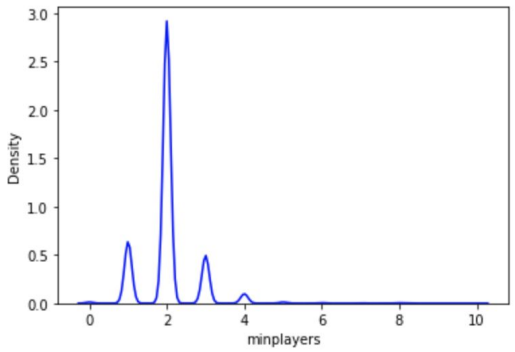
```
#kdeplot skewness graph plot
sns.kdeplot(df['minplaytime'], color = 'r')
```

```
<AxesSubplot:xlabel='minplaytime', ylabel='Density'>
```




```
#kdeplot skewness graph plot
sns.kdeplot(df['minplayers'], color = 'b')
```

```
<AxesSubplot:xlabel='minplayers', ylabel='Density'>
```

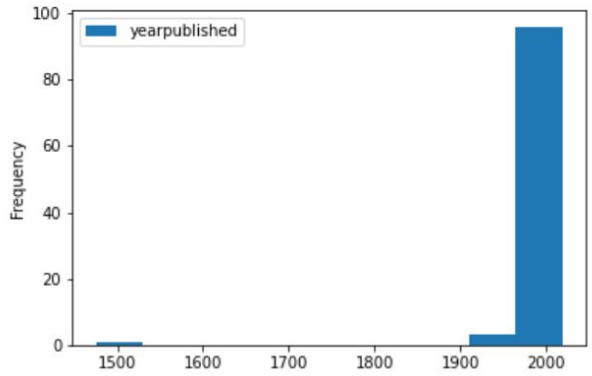


Histogram Plot

```
info1 = info[:100]
#Histogram

info1.plot(info1.index.name,[3],kind="hist")
```

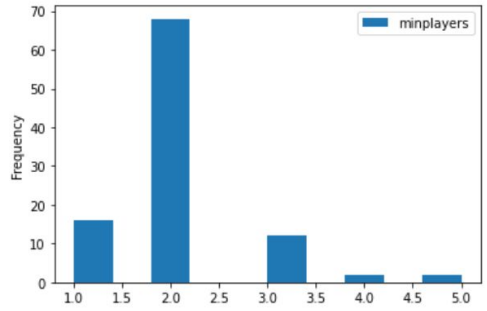
```
<matplotlib.axes._subplots.AxesSubplot at 0x1393cecc640>
```



- This Histogram plot shows that we have some games that were published in around the year 1500 (they can be ancient games) but most of the games have been published around 2000.

```
info1.plot(info1.index.name,[4],kind="hist")
```

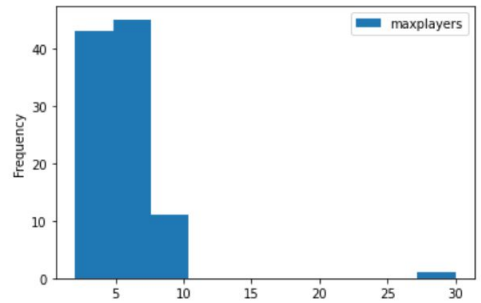
```
<matplotlib.axes._subplots.AxesSubplot at 0x1393cf448e0>
```



- In this Histogram plot of minimum players we can realize most of the games need at least 2 players however, there are some games that need respectively 1, 3, 4, and 5 players at least.

```
info1.plot(info1.index.name,[5],kind="hist")
```

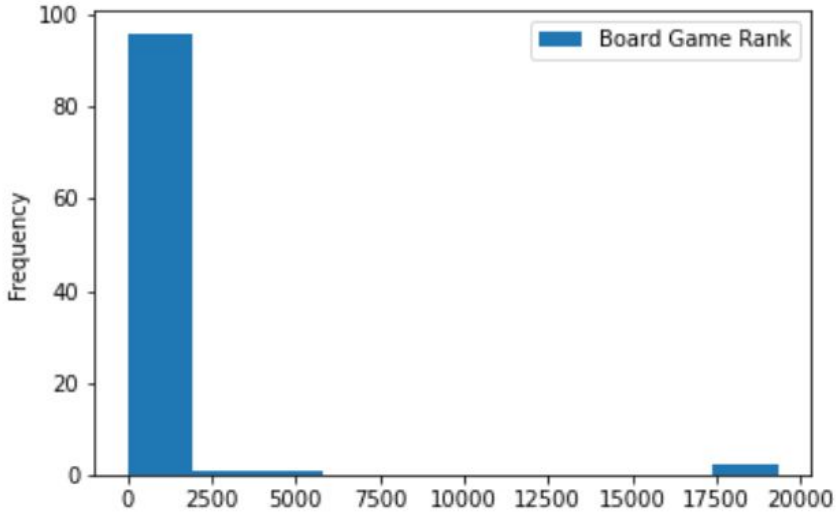
```
<matplotlib.axes._subplots.AxesSubplot at 0x1393cfb4730>
```



- In this Histogram plot of maximum player we can see that most of the games need at the most about 5 to 7 players However, there are some other games that need at the most 10 players and a little number of games that can have about 30 players at the most that means there is no limit for the number of the players.

```
info1.plot(info1.index.name,[19],kind="hist")
```

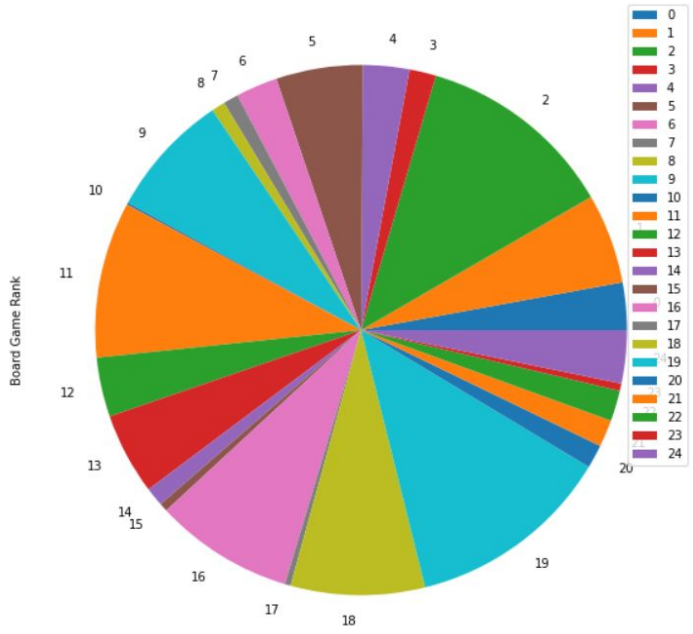
```
<matplotlib.axes._subplots.AxesSubplot at 0x21f0a6c1460>
```



- This Histogram plot shows Board Game Rank which clearly most of the ranks are between 0 to 2500.

Pie Plot

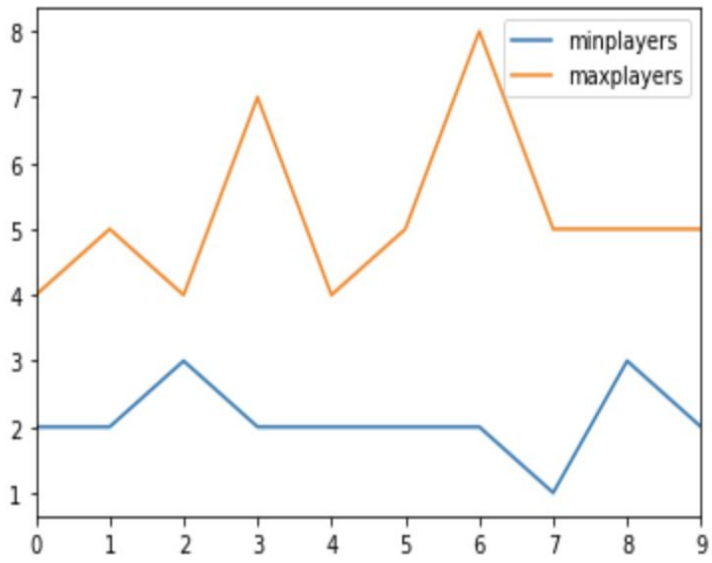
```
plot = info1[:25].plot.pie(y='Board Game Rank', figsize=(10,10))
```



Line Plot

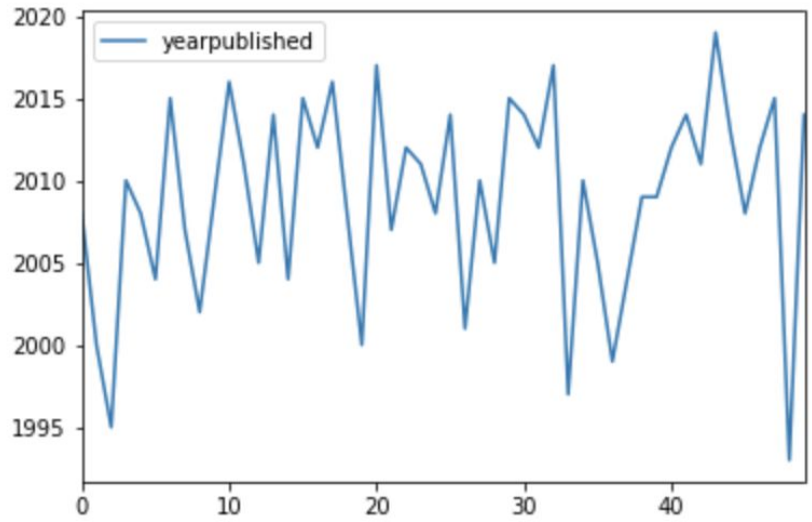
```
1 df_temp.plot(df1.index.name,['minplayers','maxplayers'],kind='line')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f995e21b630>
```



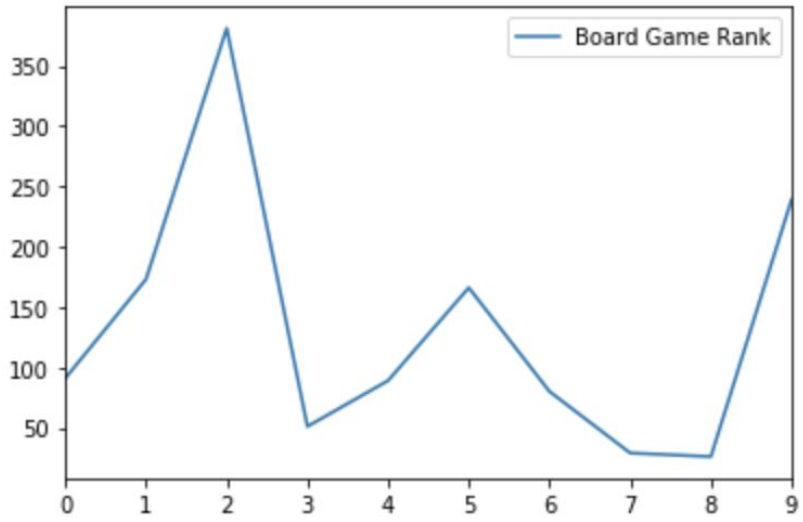

```
1 df_temp.plot(df1.index.name,['yearpublished'],kind='line')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f99710aeb38>
```



```
1 df_temp.plot(df1.index.name,['Board Game Rank'],kind='line')
```

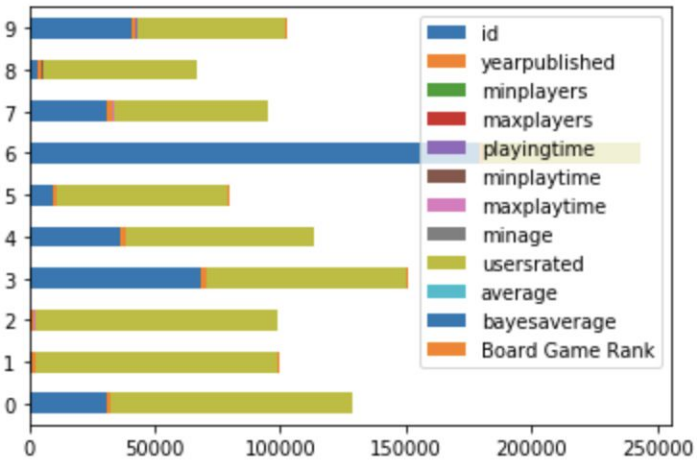
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f995e5c0550>
```



Bar Graph

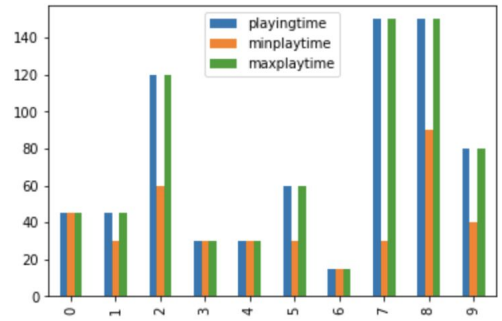
```
1 df_temp.plot.barh(stacked='True')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9a3f0a3ac8>
```



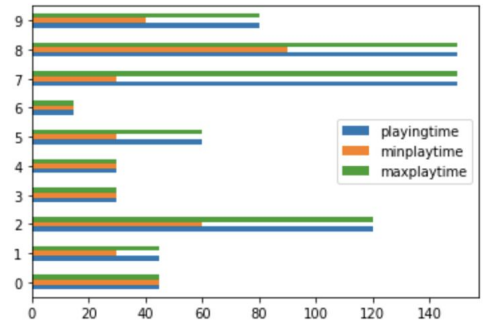
```
1 df_temp.plot(df1.index.name,['playingtime','minplaytime','maxplaytime'],kind='bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f995d95a198>
```



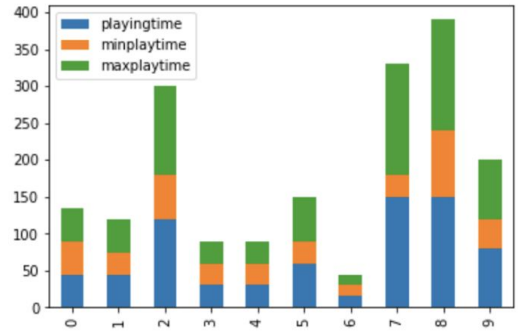

```
1 df_temp.plot(df1.index.name,['playingtime','minplaytime','maxplaytime'],kind='barh')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f995da4c198>
```




```
1 df_temp.plot(df1.index.name,['playingtime','minplaytime','maxplaytime'],kind='bar',stacked='True')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f995db8e860>
```



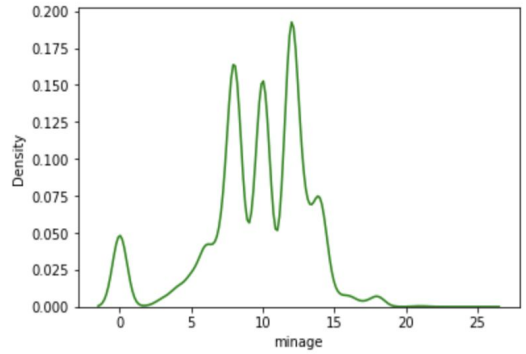
Kernel density estimate

- For performing the Data visualization of the detailed_info dataset, we started with the KDE plot. Please refer to the figures below for more information.
- The yearpublished graph is negatively skewed. We can also see here clearly that it has some garbage values like -3000.

- The maxplayers, minplayers, maxplaytime graphs are positively skewed as they have a tail extending towards the right end of the graph.
- The minplayers and minplaytime graph more than one spikes in the graphs. These spikes represent outliers. This information can be used to remove unnecessary tuples while building the recommendation model.
- The average and bayesaverage graphs have symmetrical distribution, hence they don't contain outliers and all the data can be considered for normal distribution while building the recommendation model.

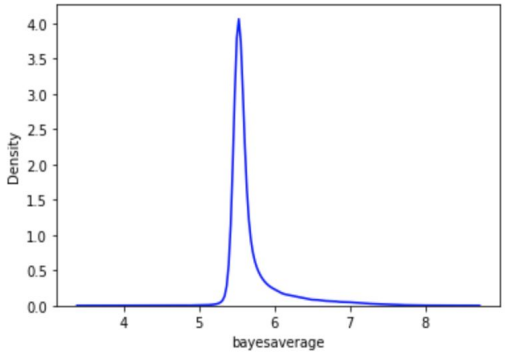
```
#kdeplot skewness graph plot
sns.kdeplot(df['minage'], color = 'g')
```

```
<AxesSubplot:xlabel='minage', ylabel='Density'>
```



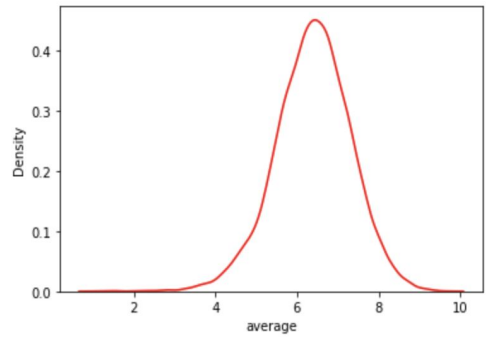

```
#kdeplot skewness graph plot
sns.kdeplot(df['bayesaverage'], color = 'b')
```

```
<AxesSubplot:xlabel='bayesaverage', ylabel='Density'>
```



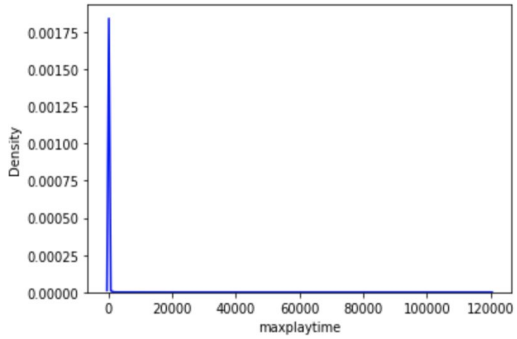

```
#kdeplot skewness graph plot
sns.kdeplot(df['average'], color = 'r')
```

```
<AxesSubplot:xlabel='average', ylabel='Density'>
```




```
#kdeplot skewness graph plot
sns.kdeplot(df['maxplaytime'], color = 'b')
```

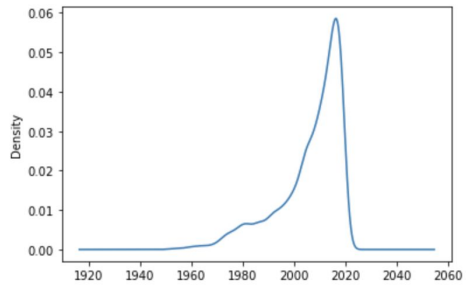
```
<AxesSubplot:xlabel='maxplaytime', ylabel='Density'>
```



- The “yearpublished” column’s density graph shows that more games are being released the later the date is. This is not likely to affect any model we create, though it is worth bearing in mind that more of our data is newer.

```
games_info.yearpublished.plot.density()
```

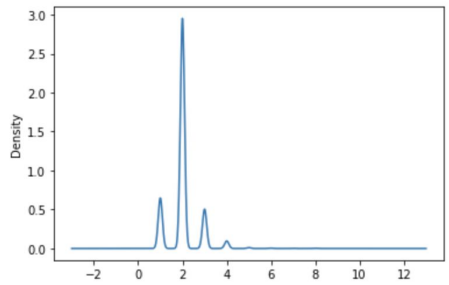
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fed36ffe8e0>
```



- The density graphs for min and max player counts for games show that most games have a minimum player count of two, and if not, then the minimum player count is likely 1 or 3. The maximum has a mode of 4, with smaller peaks at 2 and 5-6.

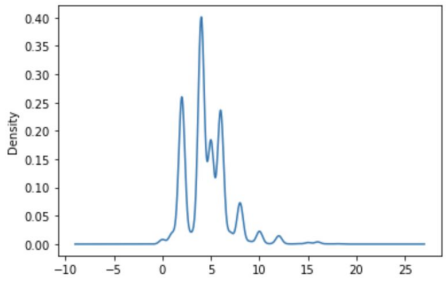
```
games_info.minplayers.plot.density()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fed39309820>
```




```
games_info_maxplayer_constrained.maxplayers.plot.density()
```

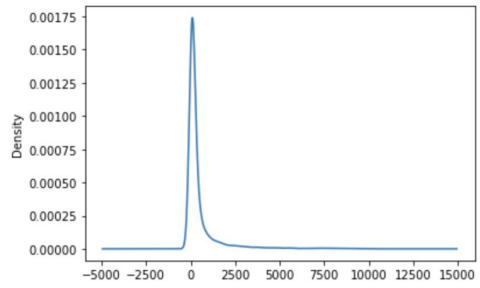
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fed39166ee0>
```



- The “userrated” density graph shows that most games have been rated less than 1,000 times, while a few exceptions have more ratings. From the perspective of transaction lists, this will mean that the lists we will be dealing with will be relatively short, so this may influence what we choose as our target support.

```
games_info.usersrated.plot.density()
```

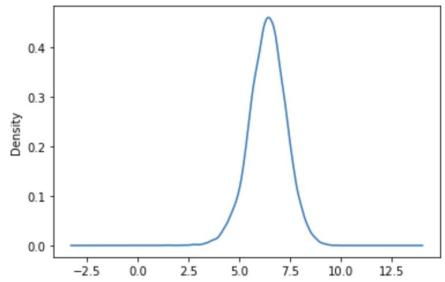
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fed38bae9d0>
```



- The average rating density plot shows that most games’ average rating is about a 7, and few games are outside of the range 5-8.

```
games_info.average.plot.density()
```

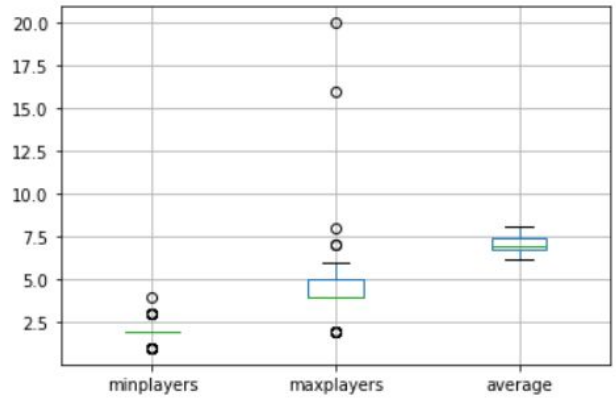
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fed39022e80>
```



Boxplot

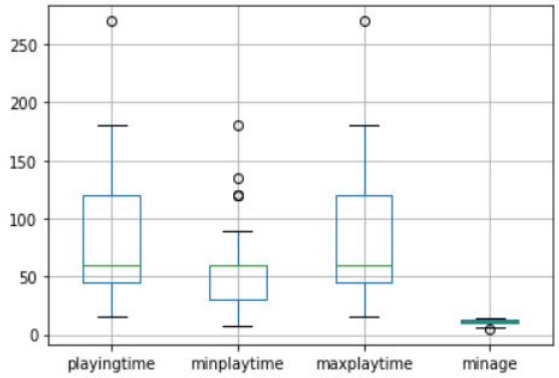
```
boxplot = games_info_numeric[900:950].boxplot(column=['minplayers', 'maxplayers', 'average'])
boxplot
```

```
<AxesSubplot:>
```




```
boxplot = games_info_numeric[700:750].boxplot(column=['playingtime', 'minplaytime', 'maxplaytime', 'minage'])
boxplot
```

```
<AxesSubplot:>
```



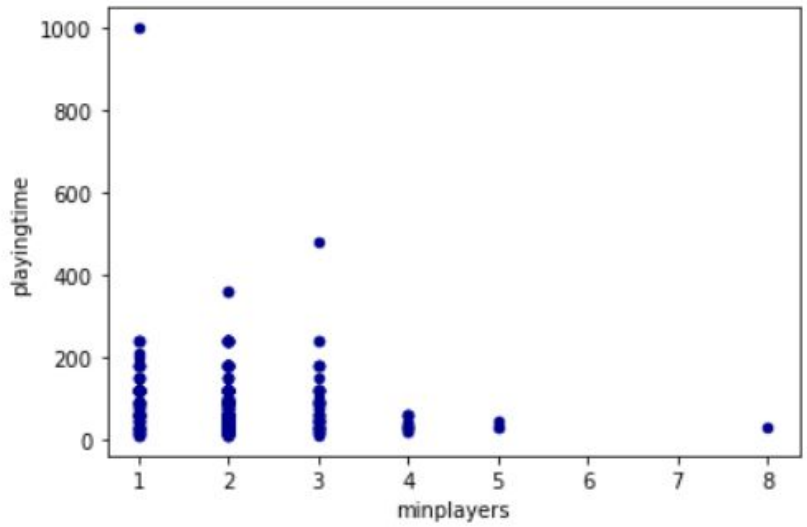
- Boxplot here is taken in 50 datasamples. We calculated it over multiple portions of data and this is the result we get for most of the data, with positions of outliers varying.

- Here we can see for major data attributes, their ideal value ranges and some outliers, which we can remove in the data cleaning process.

Scatter plots

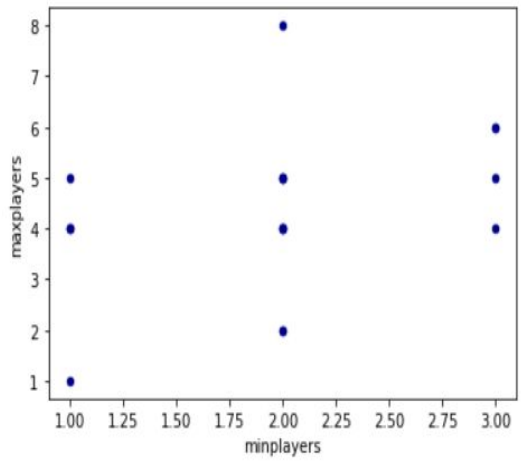
- Scatter plot here is used to visualize dependency of two or more attributes
- First, playingtime and minplayers

```
scatter_2 = games_info_numeric[:550].plot.scatter(x='minplayers',
                      y='playingtime',
                      c='DarkBlue')
#scatter between min players and playingtime
```



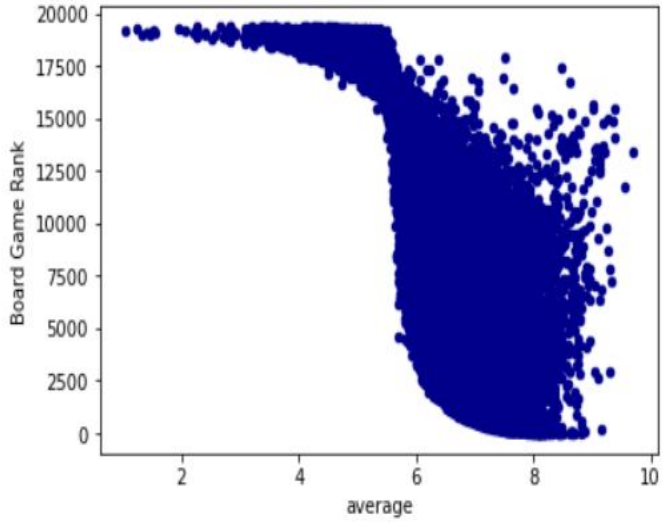
- Analysis between minplayers and maxplayers

```
scatter_1 = games_info_numeric[800:820].plot.scatter(x='minplayers',
                    y='maxplayers',
                    c='DarkBlue')
#scatter between min players and max player
```



- Analysis between average and board game rank

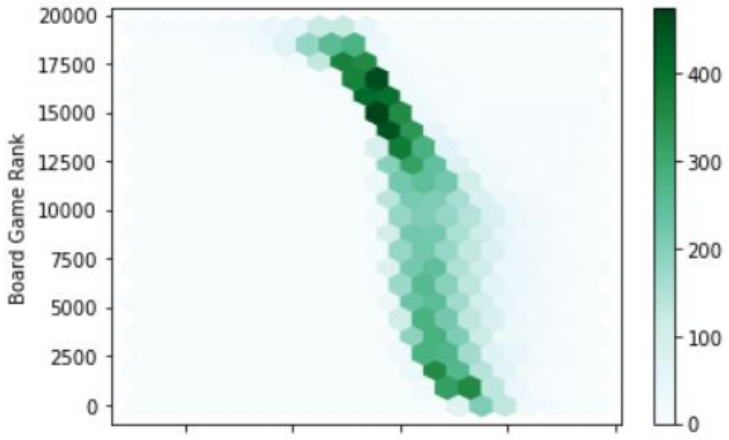
```
scatter_3 = games_info_numeric[:].plot.scatter(x='average',
                      y='Board Game Rank',
                      c='DarkBlue')
```



Hexbin Plot

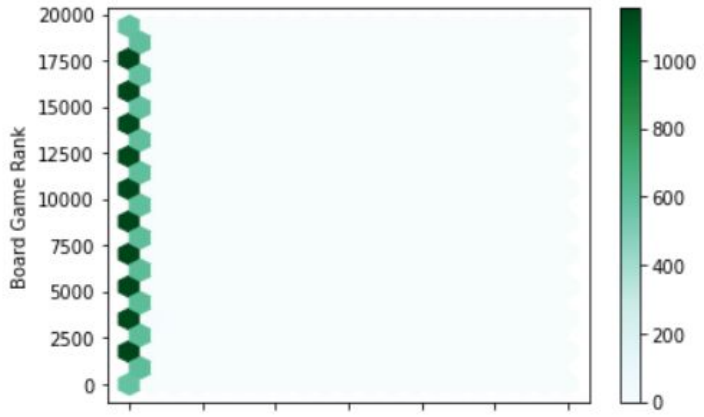
- Hexbin is similar to boxplot, but just the visualization is in the form of hexbins
- Above scatterplot shows some relation between average and boardgame, so the hexbin plot is as shown:

```
hexbin = games_info_numeric.plot.hexbin(x='average', y='Board Game Rank', gridsize=20)
```



- Hexbin plot between playing time and board game rank

```
hexbin = games_info_numeric.plot.hexbin(x='playingtime', y='Board Game Rank', gridsize=20)
```



Conclusion:

Hence, this was our data exploration part in which we explored 2 datasets, did statistical analysis as well as visualization on them. From these points we can conclude that:

- Based on the correlation test, we can conclude that there is no significant relation between the different numeric attributes of the dataset.
- Line and bar graph visualizations show that values for a particular attribute are distributed over the dataset.
- Based on the boxplot, we can see that there are certain data points that fall into the outlier category. This can be very useful in eliminating outlier/noisy data in the data cleaning process. Example in the second dataset the maxplayers lies between 2-5 and hence, the dataset can be sliced based on that criteria. Similarly we can take other attributes and remove noisy data accordingly.
- There are a few outliers that may be due to incorrect data, especially with regards to Year Published. When those were excluded from the data, density graphs and histograms showed that board games are being published at an increasing rate. This should not affect the performance of any models we create, but it is worth keeping in mind that much of our data is relatively new.