# Assignment: 1.2 Exercise: Exploring a Pandas Data Frame (Veera Reddy Koppula)

In [1]:
```
#Load the necessary libraries
import numpy as np
import pandas as pd
pd.options.mode.chained_assignment = None
```

Download the Video Game Sales with Ratings dataset from this link: Video Game Sales with Ratings

**1.Load the dataset as a Pandas data frame.**

In [2]:
```
#loading Vide Games sales CSV into a pandas data frame
df=pd.read_csv("Video_Games_Sales_as_at_22_Dec_2016.csv")
```

**2.Display the first ten rows of data.**

In [3]:
```
#Displaying top 10 rows
df.head(10)
```

Out[3]:

| | Name | Platform | Year_of_Release | Genre | Publisher | NA_Sales | EU_Sales | JP_Sales | Other_Sales | Global_Sales | Critic_Sco |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Wii Sports | Wii | 2006.0 | Sports | Nintendo | 41.36 | 28.96 | 3.77 | 8.45 | 82.53 | 76 |
| 1 | Super Mario Bros. | NES | 1985.0 | Platform | Nintendo | 29.08 | 3.58 | 6.81 | 0.77 | 40.24 | Nа |
| 2 | Mario Kart Wii | Wii | 2008.0 | Racing | Nintendo | 15.68 | 12.76 | 3.79 | 3.29 | 35.52 | 82 |
| 3 | Wii Sports Resort | Wii | 2009.0 | Sports | Nintendo | 15.61 | 10.93 | 3.28 | 2.95 | 32.77 | 80 |
| 4 | Pokemon Red/Pokemon Blue | GB | 1996.0 | Role-Playing | Nintendo | 11.27 | 8.89 | 10.22 | 1.00 | 31.37 | Nа |
| 5 | Tetris | GB | 1989.0 | Puzzle | Nintendo | 23.20 | 2.26 | 4.22 | 0.58 | 30.26 | Nа |
| 6 | New Super Mario Bros. | DS | 2006.0 | Platform | Nintendo | 11.28 | 9.14 | 6.50 | 2.88 | 29.80 | 89 |
| 7 | Wii Play | Wii | 2006.0 | Misc | Nintendo | 13.96 | 9.18 | 2.93 | 2.84 | 28.92 | 58 |
| 8 | New Super Mario Bros. Wii | Wii | 2009.0 | Platform | Nintendo | 14.44 | 6.94 | 4.70 | 2.24 | 28.32 | 87 |
| 9 | Duck Hunt | NES | 1984.0 | Shooter | Nintendo | 26.93 | 0.63 | 0.28 | 0.47 | 28.31 | Nа |

**3.Find the dimensions (number of rows and columns) in the data frame. What do these two numbers represent in the context of the data?**

In [4]:
```
#Displaying Size of the data frame
df.shape
```

Out[4]:
```
(16719, 16)
```

first number 16719 represents the number of rows in the dataset

second number 16 represents the number of columns in the dataset

**4.Find the top five games by critic score.**

In [5]:
```
#Sorting data by top 5 rows with highest critic Score
df.sort_values('Critic_Score',ascending = False).head(5)
```

Out[5]:

| | Name | Platform | Year_of_Release | Genre | Publisher | NA_Sales | EU_Sales | JP_Sales | Other_Sales | Global_Sales | Critic_Sc |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 227 | Tony Hawk's Pro Skater 2 | PS | 2000.0 | Sports | Activision | 3.05 | 1.41 | 0.02 | 0.20 | 4.68 | 9 |
| 57 | Grand Theft Auto IV | PS3 | 2008.0 | Action | Take-Two Interactive | 4.76 | 3.69 | 0.44 | 1.61 | 10.50 | 9 |
| 51 | Grand Theft Auto IV | X360 | 2008.0 | Action | Take-Two Interactive | 6.76 | 3.07 | 0.14 | 1.03 | 11.01 | 9 |
| 5350 | SoulCalibur | DC | 1999.0 | Fighting | Namco Bandai Games | 0.00 | 0.00 | 0.34 | 0.00 | 0.34 | 9 |
| 165 | Grand Theft Auto V | XOne | 2014.0 | Action | Take-Two Interactive | 2.81 | 2.19 | 0.00 | 0.47 | 5.48 | 9 |

**5.Find the number of video games in the data frame in each genre**

In [6]:
```
#Displaying data of number of games in eache Genre
df.Genre.value_counts()
```

Out[6]:
```
Action          3370
Sports          2348
Misc            1750
Role-Playing    1500
Shooter         1323
Adventure       1303
Racing          1249
Platform         888
Simulation       874
Fighting         849
Strategy         683
Puzzle           580
Name: Genre, dtype: int64
```

**6.Find the first five games in the data frame on the SNES platform.**

In [7]:
```
#Displaying first 5 rows with game on platform SNES
df.loc[df['Platform'] == "SNES"].head(5)
```

Out[7]:

| | Name | Platform | Year_of_Release | Genre | Publisher | NA_Sales | EU_Sales | JP_Sales | Other_Sales | Global_Sales | Critic_Score |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 18 | Super Mario World | SNES | 1990.0 | Platform | Nintendo | 12.78 | 3.75 | 3.54 | 0.55 | 20.61 | NaN |
| 56 | Super Mario All-Stars | SNES | 1993.0 | Platform | Nintendo | 5.99 | 2.15 | 2.12 | 0.29 | 10.55 | NaN |
| 71 | Donkey Kong Country | SNES | 1994.0 | Platform | Nintendo | 4.36 | 1.71 | 3.00 | 0.23 | 9.30 | NaN |
| 76 | Super Mario Kart | SNES | 1992.0 | Racing | Nintendo | 3.54 | 1.24 | 3.81 | 0.18 | 8.76 | NaN |
| 137 | Street Fighter II: The World Warrior | SNES | 1992.0 | Fighting | Capcom | 2.47 | 0.83 | 2.87 | 0.12 | 6.30 | NaN |

**7.Find the five publishers with the highest total global sales. Note: You will need to calculate the total global sales for each publisher to do this.**

In [8]:
```
#grouping data by publisher
grouped_df = df.groupby(["Publisher"])
#Creating sum of Global sales by publisher
grouped_and_summed = grouped_df.Global_Sales.sum()
#Printing top 5 rows with highest global sales by publisher
grouped_and_summed.sort_values(ascending = False).head(5)
```

Out[8]:
```
Publisher
Nintendo                      1788.81
Electronic Arts               1116.96
Activision                     731.16
Sony Computer Entertainment    606.48
Ubisoft                        471.61
Name: Global_Sales, dtype: float64
```

**8.Create a new column in the data frame that calculates the percentage of global sales from North America. Display the first five rows of the new data frame.**

In [9]:
```
#Creating a new column in exisiting data frame 'NA_Sales_percent' representing the NA Sales as percentage of to
df['NA_Sales_percent']=(df['NA_Sales'] /
                        df['Global_Sales']) * 100
df.head(5)
```

Out[9]:

| | Name | Platform | Year_of_Release | Genre | Publisher | NA_Sales | EU_Sales | JP_Sales | Other_Sales | Global_Sales | Critic_Sco |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Wii Sports | Wii | 2006.0 | Sports | Nintendo | 41.36 | 28.96 | 3.77 | 8.45 | 82.53 | 76 |
| 1 | Super Mario Bros. | NES | 1985.0 | Platform | Nintendo | 29.08 | 3.58 | 6.81 | 0.77 | 40.24 | Nа |
| 2 | Mario Kart Wii | Wii | 2008.0 | Racing | Nintendo | 15.68 | 12.76 | 3.79 | 3.29 | 35.52 | 82 |
| 3 | Wii Sports Resort | Wii | 2009.0 | Sports | Nintendo | 15.61 | 10.93 | 3.28 | 2.95 | 32.77 | 80 |
| 4 | Pokemon Red/Pokemon Blue | GB | 1996.0 | Role-Playing | Nintendo | 11.27 | 8.89 | 10.22 | 1.00 | 31.37 | Nа |

**9.Find the number NaN entries (missing data values) in each column.**

In [10]:
```
#Printing number of NaN entries in each column
df.isnull().sum()
```

Out[10]:
```
Name                  2
Platform              0
Year_of_Release     269
Genre                 2
Publisher            54
NA_Sales              0
EU_Sales              0
JP_Sales              0
Other_Sales           0
Global_Sales          0
Critic_Score       8582
Critic_Count       8582
User_Score         6704
User_Count         9129
Developer          6623
Rating             6769
NA_Sales_percent      0
dtype: int64
```

**10.Try to calculate the median user score of all the video games. You will likely run into an error because some of the user score entries are a non-numerical string that cannot be converted to a float. Find and replace this string with NaN and then calculate the median. Then, replace all NaN entries in the user score column with the median value.**

In [11]:
```
#Finding all the unique values in User Score column
User_Score_values = df.User_Score.unique()

#Display all the values
User_Score_values
```

Out[11]:
```
array(['8', nan, '8.3', '8.5', '6.6', '8.4', '8.6', '7.7', '6.3', '7.4',
       '8.2', '9', '7.9', '8.1', '8.7', '7.1', '3.4', '5.3', '4.8', '3.2',
       '8.9', '6.4', '7.8', '7.5', '2.6', '7.2', '9.1', '7', '7.3', '4.3',
       '7.6', '5.7', '5', '9.1', '6.5', 'tbd', '8.8', '6.9', '9.4', '6.8',
       '6.1', '6.7', '5.4', '4', '4.9', '4.5', '9.3', '6.2', '4.2', '6',
       '3.7', '4.1', '5.8', '5.6', '5.5', '4.4', '4.6', '5.9', '3.9',
       '3.1', '2.9', '5.2', '3.3', '4.7', '5.1', '3.5', '2.5', '1.9', '3',
       '2.7', '2.2', '2', '1.5', '2.1', '3.6', '2.8', '1.8', '3.8', '0',
       '1.6', '9.6', '2.4', '1.7', '1.1', '0.3', '1.3', '0.7', '1.2',
       '2.3', '0.5', '1.3', '0.2', '0.6', '1.4', '0.9', '1', '9.7'],
      dtype=object)
```

Based on above here is one string 'tbd' to be replaced by null values

In [12]:
```
#Replacing tbd string with NaN values
df['User_Score'] = df['User_Score'].replace(r'tbd', np.NaN, regex=True)
```

In [13]:
```
#Converting User_Score column to float type from object
df['User_Score'] = df['User_Score'].astype(float)
```

In [14]:
```
#Finding all the unique values in User_Score column - after the update to type of column to float
User_Score_update_values = df.User_Score.unique()
#Display all the values
User_Score_update_values
```

Out[14]:
```
array([8. , nan, 8.3, 8.5, 6.6, 8.4, 8.6, 7.7, 6.3, 7.4, 8.2, 9. , 7.9,
       8.1, 8.7, 7.1, 3.4, 5.3, 4.8, 3.2, 8.9, 6.4, 7.8, 7.5, 2.6, 7.2,
       9.2, 7. , 7.3, 4.3, 7.6, 5.7, 5. , 9.1, 6.5, 8.8, 6.9, 9.4, 6.8,
       6.1, 6.7, 5.4, 4. , 4.9, 4.5, 9.3, 6.2, 4.2, 6. , 3.7, 4.1, 5.8,
       5.6, 5.5, 4.4, 4.6, 5.9, 3.9, 3.1, 2.9, 5.2, 3.3, 4.7, 5.1, 3.5,
       2.5, 1.9, 3. , 2.7, 2.2, 2. , 9.5, 2.1, 3.6, 2.8, 1.8, 3.8, 0. ,
       1.6, 9.6, 2.4, 1.7, 1.1, 0.3, 1.3, 0.7, 1.2, 2.3, 0.5, 1.3, 0.2,
       0.6, 1.4, 0.9, 1. , 9.7])
```

In [15]:
```
#Calcululating the Median of User_score Column
median=df['User_Score'].median()
median
```

Out[15]:
```
7.5
```

In [16]:
```
#replacing Null(NaN) values with median value 7.5
df['User_Score'].fillna(value=median, inplace=True)
```

In [17]:
```
#Finding all the unique values in User Score column - to validate that null values are replaced by median value
User_Score_update_med = df.User_Score.unique()

#Display all the values
User_Score_update_med
```

Out[17]:
```
array([8. , 7.5, 8.3, 8.5, 6.6, 8.4, 8.6, 7.7, 6.3, 7.4, 8.2, 9. , 7.9,
       8.1, 8.7, 7.1, 3.4, 5.3, 4.8, 3.2, 8.9, 6.4, 7.8, 7.5, 2.6, 7.2, 9.2,
       7. , 7.3, 4.3, 7.6, 5.7, 5. , 9.1, 6.5, 8.8, 6.9, 9.4, 6.8, 6.1,
       6.7, 5.4, 4. , 4.9, 4.5, 9.3, 6.2, 4.2, 6. , 3.7, 4.1, 5.8, 5.6,
       5.5, 4.4, 4.6, 5.9, 3.9, 3.1, 2.9, 5.2, 3.3, 4.7, 5.1, 3.5, 2.5,
       1.9, 3. , 2.7, 2.2, 2. , 9.5, 2.1, 3.6, 2.8, 1.8, 3.8, 0. , 1.6,
       9.6, 2.4, 1.7, 1.1, 0.3, 1.5, 0.7, 1.2, 2.3, 0.5, 1.3, 0.2, 0.6,
       1.4, 0.9, 1. , 9.7])
```

In [ ]:
```

```