# Untitled

October 26, 2020

[404]:
```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

[405]:
```
# Upload data games
df=pd.read_csv('vgsales.csv')
```

### 0.1 Assessing data

[406]:
```
df.head(20)
```

[406]:

| | Rank | Name | Platform | Year | Genre | Publisher | NA_Sales | EU_Sales | JP_Sales |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | Wii Sports | Wii | 2006.0 | Sports | Nintendo | 41.49 | 29.02 | 3.77 |
| 1 | 2 | Super Mario Bros. | NES | 1985.0 | Platform | Nintendo | 29.08 | 3.58 | 6.81 |
| 2 | 3 | Mario Kart Wii | Wii | 2008.0 | Racing | Nintendo | 15.85 | 12.88 | 3.79 |
| 3 | 4 | Wii Sports Resort | Wii | 2009.0 | | | | | |
| 4 | 5 | Pokemon Red/Pokemon Blue | GB | 1996.0 | | | | | |
| 5 | 6 | Tetris | GB | 1989.0 | | | | | |
| 6 | 7 | New Super Mario Bros. | DS | 2006.0 | | | | | |
| 7 | 8 | Wii Play | Wii | 2006.0 | | | | | |
| 8 | 9 | New Super Mario Bros. Wii | Wii | 2009.0 | | | | | |
| 9 | 10 | Duck Hunt | NES | 1984.0 | | | | | |
| 10 | 11 | Nintendogs | DS | 2005.0 | | | | | |
| 11 | 12 | Mario Kart DS | DS | 2005.0 | | | | | |
| 12 | 13 | Pokemon Gold/Pokemon Silver | GB | 1999.0 | | | | | |
| 13 | 14 | Wii Fit | Wii | 2007.0 | | | | | |
| 14 | 15 | Wii Fit Plus | Wii | 2009.0 | | | | | |
| 15 | 16 | Kinect Adventures! | X360 | 2010.0 | | | | | |
| 16 | 17 | Grand Theft Auto V | PS3 | 2013.0 | | | | | |
| 17 | 18 | Grand Theft Auto: San Andreas | PS2 | 2004.0 | | | | | |
| 18 | 19 | Super Mario World | SNES | 1990.0 | | | | | |
| 19 | 20 | Brain Age: Train Your Brain in Minutes a Day | DS | 2005.0 | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| 3 | Sports | Nintendo | 15.75 | 11.01 | 3.28 |
| 4 | Role-Playing | Nintendo | 11.27 | 8.89 | 10.22 |
| 5 | Puzzle | Nintendo | 23.20 | 2.26 | 4.22 |
| 6 | Platform | Nintendo | 11.38 | 9.23 | 6.50 |
| 7 | Misc | Nintendo | 14.03 | 9.20 | 2.93 |
| 8 | Platform | Nintendo | 14.59 | 7.06 | 4.70 |
| 9 | Shooter | Nintendo | 26.93 | 0.63 | 0.28 |
| 10 | Simulation | Nintendo | 9.07 | 11.00 | 1.93 |
| 11 | Racing | Nintendo | 9.81 | 7.57 | 4.13 |
| 12 | Role-Playing | Nintendo | 9.00 | 6.18 | 7.20 |
| 13 | Sports | Nintendo | 8.94 | 8.03 | 3.60 |
| 14 | Sports | Nintendo | 9.09 | 8.59 | 2.53 |
| 15 | Misc | Microsoft Game Studios | 14.97 | 4.94 | 0.24 |
| 16 | Action | Take-Two Interactive | 7.01 | 9.27 | 0.97 |
| 17 | Action | Take-Two Interactive | 9.43 | 0.40 | 0.41 |
| 18 | Platform | Nintendo | 12.78 | 3.75 | 3.54 |
| 19 | Misc | Nintendo | 4.75 | 9.26 | 4.16 |

| | Other_Sales | Global_Sales |
|---|---|---|
| 0 | 8.46 | 82.74 |
| 1 | 0.77 | 40.24 |
| 2 | 3.31 | 35.82 |
| 3 | 2.96 | 33.00 |
| 4 | 1.00 | 31.37 |
| 5 | 0.58 | 30.26 |
| 6 | 2.90 | 30.01 |
| 7 | 2.85 | 29.02 |
| 8 | 2.26 | 28.62 |
| 9 | 0.47 | 28.31 |
| 10 | 2.75 | 24.76 |
| 11 | 1.92 | 23.42 |
| 12 | 0.71 | 23.10 |
| 13 | 2.15 | 22.72 |
| 14 | 1.79 | 22.00 |
| 15 | 1.67 | 21.82 |
| 16 | 4.14 | 21.40 |
| 17 | 10.57 | 20.81 |
| 18 | 0.55 | 20.61 |
| 19 | 2.05 | 20.22 |

[407]:
```
#check for any null values in each column
list_col=list(df.columns)
for col in list_col:
    n=df[col].isnull().sum()
    print(str(col)+':'+str(n))
```

```
Rank:0
```

```
Name:0
Platform:0
Year:271
Genre:0
Publisher:58
NA_Sales:0
EU_Sales:0
JP_Sales:0
Other_Sales:0
Global_Sales:0
```

[408]:
```
df.shape
```

[408]:
```
(16598, 11)
```

[409]:
```
df.dtypes
```

[409]:
```
Rank             int64
Name            object
Platform        object
Year           float64
Genre           object
Publisher       object
NA_Sales       float64
EU_Sales       float64
JP_Sales       float64
Other_Sales    float64
Global_Sales   float64
dtype: object
```

[410]:
```
df.duplicated().sum()
```

[410]:
```
0
```

[411]:
```
name_df=df.Name.duplicated().sum()
name_df
```

[411]:
```
5105
```

[412]:
```
# total unique games
df.shape[0]-name_df
```

[412]:
```
11493
```

## 0.2 Cleaning time

[413]:
```
# make sure that global sales is sum of those columns
condition=(df.Global_Sales==df.NA_Sales +df.EU_Sales +df.JP_Sales)
if condition.any():
    df.Global_Sales=df.NA_Sales +df.EU_Sales +df.JP_Sales
```

[414]:
```
# test the code
condition=(df.Global_Sales==df.NA_Sales +df.EU_Sales +df.JP_Sales)
condition
```

[414]:
```
0        True
1        True
2        True
3        True
4        True
         …
16593    True
16594    True
16595    True
16596    True
16597    True
Length: 16598, dtype: bool
```

[415]:
```
# to convert year column to int type , first i have to deal with all nan values␣
 ↪which contain that column.
# will replace NAN values with 0
df.Year=df.Year.fillna(0)
```

[416]:
```
# convert year column type from float to int.
df.Year=pd.to_numeric(df.Year)
df.Year=df.Year.astype(int)
```

[417]:
```
#to test the code
df.dtypes
```

[417]:
```
Rank             int64
Name            object
Platform        object
Year             int32
Genre           object
Publisher       object
NA_Sales       float64
EU_Sales       float64
JP_Sales       float64
Other_Sales    float64
Global_Sales   float64
```

```
dtype: object
```

### 0.3 Analysis time

- **Questions**
- Total sales for each platform [the top three , and the lowest one]
- Line chart for the top platform to show the distribution of years for sales
- for each platform the most popular genre and it's total sales
- the top genre in total sales
- *IN GENERAL* top and lowest year in sales
- the most popular game
- the order of games
- for each year the most popular game

[418]:
```
df.head(2)
```

[418]:
```
   Rank              Name Platform  Year     Genre Publisher  NA_Sales  \
0     1         Wii Sports      Wii  2006    Sports  Nintendo     41.49
1     2  Super Mario Bros.      NES  1985  Platform  Nintendo     29.08

   EU_Sales  JP_Sales  Other_Sales  Global_Sales
0     29.02      3.77         8.46         74.28
1      3.58      6.81         0.77         39.47
```

### 0.4 Total sales for each platform [the top three , and the lowest one]

[419]:
```
# Total sales for each platform [the top three , and the lowest one]
df_total=df.groupby('Platform')['Global_Sales'].sum().
 ↪sort_values(ascending=False)
df_total
```

[419]:
```
Platform
PS2     1062.33
X360     894.06
Wii      845.44
PS3      815.96
DS       760.93
PS       689.93
GBA      310.12
PSP      254.03
XB       249.02
GB       247.26
NES      245.74
PS4      234.80
3DS      234.74
PC       233.13
N64      214.30
```

```
SNES     196.82
GC       193.75
XOne     129.18
2600      96.07
WiiU      75.34
PSV       53.49
SAT       33.52
GEN       27.46
DC        15.68
SCD        1.81
NG         1.44
WS         1.42
TG16       0.16
3DO        0.10
GG         0.04
PCFX       0.03
Name: Global_Sales, dtype: float64
```

#### 0.4.1 Top three are [PS2, X360 ,Wii]

#### 0.4.2 Lowest one is [PCFX]

### 0.5 _____

### 0.6 second:

### 0.7 scatter chart for the top platform to show the distribution of years for sales

[420]:
```
df_ps2=df[df['Platform']== 'PS2']
ps2=df_ps2.groupby('Year')['Global_Sales'].sum()
ps2
```

[420]:
```
Year
0        19.24
2000     35.59
2001    149.65
2002    183.66
2003    163.62
2004    171.96
2005    141.51
2006     89.18
2007     52.30
2008     34.59
2009     16.24
2010      4.44
2011      0.35
Name: Global_Sales, dtype: float64
```

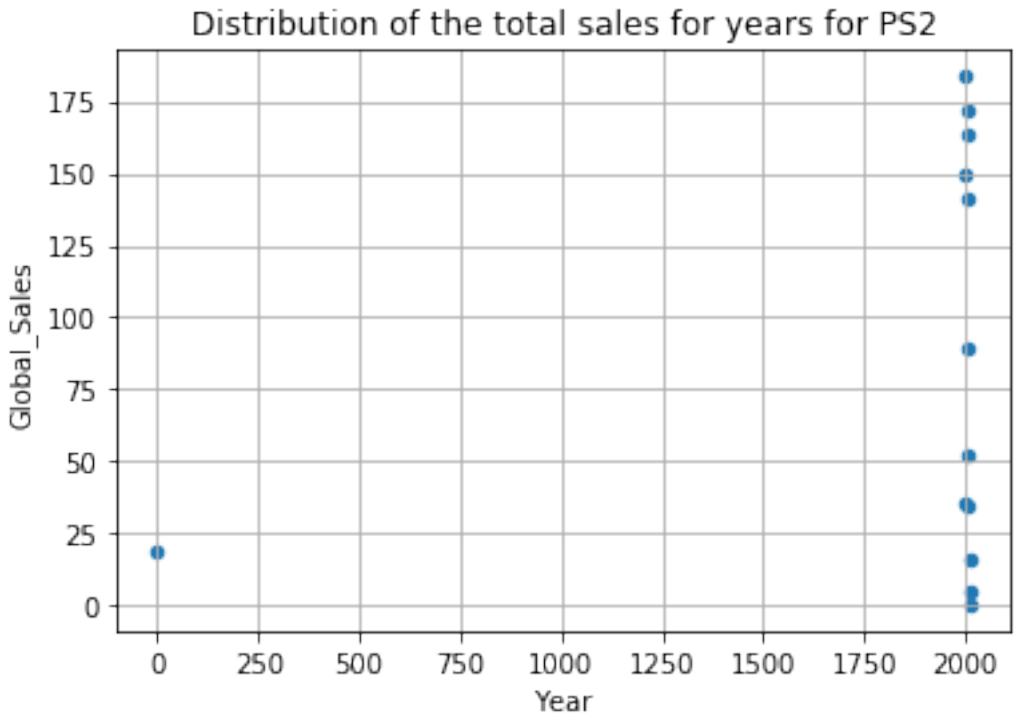
[421]:
```
ps2=pd.DataFrame(ps2.reset_index())
ps2
```

[421]:

| | Year | Global_Sales |
|---|---|---|
| 0 | 0 | 19.24 |
| 1 | 2000 | 35.59 |
| 2 | 2001 | 149.65 |
| 3 | 2002 | 183.66 |
| 4 | 2003 | 163.62 |
| 5 | 2004 | 171.96 |
| 6 | 2005 | 141.51 |
| 7 | 2006 | 89.18 |
| 8 | 2007 | 52.30 |
| 9 | 2008 | 34.59 |
| 10 | 2009 | 16.24 |
| 11 | 2010 | 4.44 |
| 12 | 2011 | 0.35 |

[422]:
```
ps2.plot(x ='Year', y='Global_Sales', kind = 'scatter')
#plt.xticks(ps2.Year)
#plt.yticks(ps2.Global_Sales)
plt.grid(True)
plt.title("Distribution of the total sales for years for PS2")
```

[422]: Text(0.5, 1.0, 'Distribution of the total sales for years for PS2')



### 0.8 3- for each paltform the most popular genre and the total sales for it

[423]:
```
# for each paltform the most popular genre and the total sales for it
platform_genre=df.groupby(['Platform','Genre'])['Global_Sales'].sum()
platform_genre=pd.DataFrame(platform_genre.reset_index())
platform_genre
```

[423]:

| | Platform | Genre | Global_Sales |
|---|---|---|---|
| 0 | 2600 | Action | 29.03 |
| 1 | 2600 | Adventure | 1.69 |
| 2 | 2600 | Fighting | 1.23 |
| 3 | 2600 | Misc | 3.54 |
| 4 | 2600 | Platform | 13.10 |
| .. | … | … | … |
| 288 | XOne | Role-Playing | 8.63 |
| 289 | XOne | Shooter | 47.21 |
| 290 | XOne | Simulation | 0.49 |
| 291 | XOne | Sports | 21.95 |
| 292 | XOne | Strategy | 0.36 |

[293 rows x 3 columns]

[424]:
```
# From this query we see the most popular genre for each platform according to␣
 ↪the total sales from it .
highest_in=platform_genre.groupby('Platform')['Genre','Global_Sales'].max()
highest_in=pd.DataFrame(highest_in.reset_index())
highest_in
```

```
C:\Users\ENTER\anaconda3\lib\site-packages\ipykernel_launcher.py:2:
FutureWarning: Indexing with multiple keys (implicitly converted to a tuple of
keys) will be deprecated, use a list instead.
```

[424]:

| | Platform | Genre | Global_Sales |
|---|---|---|---|
| 0 | 2600 | Sports | 29.03 |
| 1 | 3DO | Simulation | 0.06 |
| 2 | 3DS | Strategy | 72.79 |
| 3 | DC | Sports | 3.61 |
| 4 | DS | Strategy | 127.97 |
| 5 | GB | Strategy | 84.93 |
| 6 | GBA | Strategy | 75.94 |
| 7 | GC | Strategy | 36.71 |
| 8 | GEN | Strategy | 14.88 |
| 9 | GG | Platform | 0.04 |

```
10      N64      Strategy         39.43
11      NES        Sports         94.09
12       NG        Sports          1.42
13       PC      Strategy         47.31
14     PCFX  Role-Playing          0.03
15       PS      Strategy        119.47
16      PS2      Strategy        228.82
17      PS3      Strategy        261.37
18      PS4      Strategy         73.30
19      PSP      Strategy         53.66
20      PSV      Strategy         17.17
21      SAT      Strategy          8.48
22      SCD      Strategy          1.45
23     SNES      Strategy         63.87
24     TG16       Shooter          0.14
25       WS      Strategy          1.22
26      Wii      Strategy        265.01
27     WiiU      Strategy         19.78
28     X360      Strategy        253.91
29       XB      Strategy         61.65
30     XOne      Strategy         47.21
```

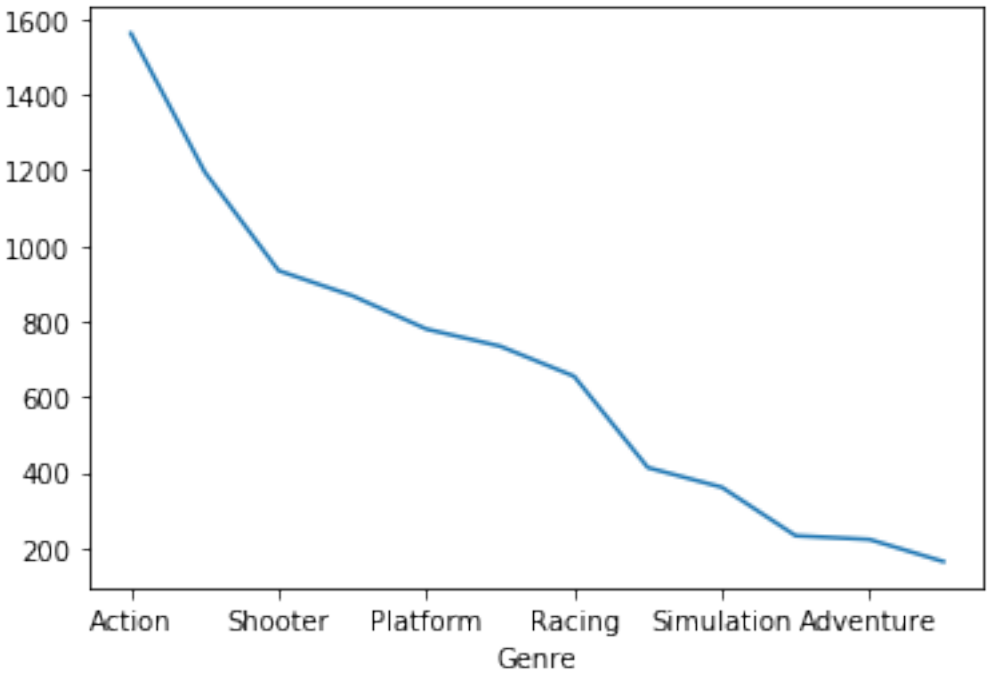
### 0.9 4-The top genre in total global sales

[425]:
```
# The top genre in total global sales
general_top_genre=df.groupby('Genre')['Global_Sales'].sum().
 ↪sort_values(ascending=False)
general_top_genre
```

[425]:
```
Genre
Action          1562.78
Sports          1195.57
Shooter          934.15
Role-Playing     867.65
Platform         779.45
Misc             733.98
Racing           654.50
Fighting         412.26
Simulation       360.39
Puzzle           231.87
Adventure        222.00
Strategy         163.50
Name: Global_Sales, dtype: float64
```

[426]:
```
# top Genre [action, sports, shooter]
# the lowest [strategy]
general_top_genre.plot()
```

[426]: <matplotlib.axes._subplots.AxesSubplot at 0xd77b4b0>




### 0.10 5-top year and lowest one in sales

[427]:
```
# top year and lowest one
# top year in sales [2008]
# the lowest two years in sales [2020 , 2017]
df.groupby('Year')['Global_Sales'].sum().sort_values(ascending=False)
```

[427]:
```
Year
2008    596.10
2009    592.33
2010    540.46
2007    532.84
2006    466.09
2011    461.54
2005    418.83
2004    371.56
2002    367.69
2003    331.60
2013    328.16
2012    325.48
2001    308.73
2014    297.08
```

```
1998    245.30
1999    241.07
2015    234.25
1997    191.94
1996    191.46
2000    190.01
0        91.25
1995     85.47
1994     77.02
1992     74.49
1989     71.95
2016     63.12
1985     53.03
1984     49.65
1990     47.97
1988     46.22
1993     45.10
1981     35.36
1986     35.15
1991     31.49
1982     28.57
1987     21.50
1983     16.66
1980     11.26
2020      0.27
2017      0.05
Name: Global_Sales, dtype: float64
```

### 0.11 6-the most popular game

### 0.12 7-the order of games

[428]:
```
#the most popular game
#the order of games
df.head(2)
```

[428]:
```
   Rank              Name Platform  Year     Genre Publisher  NA_Sales  \
0     1         Wii Sports      Wii  2006    Sports  Nintendo     41.49
1     2  Super Mario Bros.      NES  1985  Platform  Nintendo     29.08

   EU_Sales  JP_Sales  Other_Sales  Global_Sales
0     29.02      3.77         8.46         74.28
1      3.58      6.81         0.77         39.47
```

[429]:
```
popular_games=df.groupby('Name')['Global_Sales'].sum()
popular_games=pd.DataFrame(popular_games.reset_index())
```

```
popular_games=df.groupby('Name')['Global_Sales'].max().
 ↪sort_values(ascending=False)
popular_games=pd.DataFrame(popular_games.reset_index())
popular_games.head(5)
```

[429]:

| | Name | Global_Sales |
|---|---|---|
| 0 | Wii Sports | 74.28 |
| 1 | Super Mario Bros. | 39.47 |
| 2 | Mario Kart Wii | 32.52 |
| 3 | Pokemon Red/Pokemon Blue | 30.38 |
| 4 | Wii Sports Resort | 30.04 |

### 0.13 8-for each year the most popular game

[430]:
```
# for each year the most popular game
year_data=df[df['Year']!=0]
game_for_year=year_data.groupby('Year')['Name','Global_Sales'].max()

game_for_year=pd.DataFrame(game_for_year.reset_index())
game_for_year
```

```
C:\Users\ENTER\anaconda3\lib\site-packages\ipykernel_launcher.py:3:
FutureWarning: Indexing with multiple keys (implicitly converted to a tuple of
keys) will be deprecated, use a list instead.
  This is separate from the ipykernel package so we can avoid doing imports
until
```

[430]:

| | Year | Name | Global_Sales |
|---|---|---|---|
| 0 | 1980 | Missile Command | 4.26 |
| 1 | 1981 | Spider-Man | 4.45 |
| 2 | 1982 | Yars' Revenge | 7.73 |
| 3 | 1983 | Popeye | 3.18 |
| 4 | 1984 | Xevious | 27.84 |
| 5 | 1985 | Wrecking Crew | 39.47 |
| 6 | 1986 | World Class Track Meet | 6.36 |
| 7 | 1987 | Zelda II: The Adventure of Link | 4.30 |
| 8 | 1988 | Tetris | 16.82 |
| 9 | 1989 | Yakuman | 29.68 |
| 10 | 1990 | Teenage Mutant Ninja Turtles II: The Arcade Game | 20.07 |
| 11 | 1991 | Yoshi | 4.48 |
| 12 | 1992 | Yoshi's Cookie | 10.89 |
| 13 | 1993 | Yuu Yuu Hakusho | 10.26 |
| 14 | 1994 | Zero4 Champ RR | 9.07 |
| 15 | 1995 | Zoop | 5.04 |
| 16 | 1996 | World Stadium EX | 30.38 |
| 17 | 1997 | Yoshi's Story | 10.43 |
| 18 | 1998 | Yu-Gi-Oh! Monster Capsule Breed & Battle | 14.05 |

| | | | |
|---|---|---|---|
| 19 | 1999 | Zen-Nippon Pro Wrestling: Ouja no Kon | 22.38 |
| 20 | 2000 | Yu-Gi-Oh: Duel Monsters 4 | 5.40 |
| 21 | 2001 | Zoo Tycoon | 13.81 |
| 22 | 2002 | ZooCube | 15.34 |
| 23 | 2003 | Zoo Tycoon: Complete Collection | 6.76 |
| 24 | 2004 | Zoo Tycoon 2 | 10.24 |
| 25 | 2005 | everGirl | 22.00 |
| 26 | 2006 | Zombie Hunters 2 | 74.28 |
| 27 | 2007 | ¡Shin Chan Flipa en colores! | 20.57 |
| 28 | 2008 | futureU: The Prep Game for SAT | 32.52 |
| 29 | 2009 | th!nk Logic Trainer | 30.04 |
| 30 | 2010 | uDraw Studio | 20.15 |
| 31 | 2011 | uDraw Studio: Instant Artist | 13.44 |
| 32 | 2012 | [Prototype 2] | 12.62 |
| 33 | 2013 | Zyuden Sentai Kyoryuger: Game de Gaburincho!! | 17.25 |
| 34 | 2014 | inFAMOUS: Second Son | 10.68 |
| 35 | 2015 | Zombie Army Trilogy | 11.93 |
| 36 | 2016 | ZombiU | 4.09 |
| 37 | 2017 | Phantasy Star Online 2 Episode 4: Deluxe Package | 0.03 |
| 38 | 2020 | Imagine: Makeup Artist | 0.27 |

# 1 The End