Video Games Analysis

EDA: Exploratory Data Analysis

- we analyze the data
- clean the data
- visualize the data to derive inferences

In [2]:
```
#import the libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [4]:
```
#read the data

df = pd.read_csv('vgsales.csv')
```

In [5]:
```
df.tail(10)
```

Out[5]:

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
16588	16591	Mega Brain Boost	DS	2008.0	Puzzle	Majesco Entertainment	0.01	0.00	0.00	0.0	0.01
16589	16592	Chou Ezaru wa Akai Hana: Koi wa Tsuki ni Shiru...	PSV	2016.0	Action	dramatic create	0.00	0.00	0.01	0.0	0.01
16590	16593	Eiyuu Densetsu: Sora no Kiseki Material Collec...	PSP	2007.0	Role-Playing	Falcom Corporation	0.00	0.00	0.01	0.0	0.01
16591	16594	Myst IV: Revelation	PC	2004.0	Adventure	Ubisoft	0.01	0.00	0.00	0.0	0.01
16592	16595	Plushees	DS	2008.0	Simulation	Destineer	0.01	0.00	0.00	0.0	0.01
16593	16596	Woody Woodpecker in Crazy Castle 5	GBA	2002.0	Platform	Kemco	0.01	0.00	0.00	0.0	0.01
16594	16597	Men in Black II: Alien Escape	GC	2003.0	Shooter	Infogrames	0.01	0.00	0.00	0.0	0.01
16595	16598	SCORE International Baja 1000: The Official Game	PS2	2008.0	Racing	Activision	0.00	0.00	0.00	0.0	0.01
16596	16599	Know How 2	DS	2010.0	Puzzle	7G//AMES	0.00	0.01	0.00	0.0	0.01
16597	16600	Spirits & Spells	GBA	2003.0	Platform	Wanadoo	0.01	0.00	0.00	0.0	0.01

How much data do we have?

In [132…
```
df.shape
```

Out[132… (16598, 11)

In [133…
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16598 entries, 0 to 16597
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Rank          16598 non-null  int64  
 1   Name          16598 non-null  object 
 2   Platform      16598 non-null  object 
 3   Year          16327 non-null  float64
 4   Genre         16598 non-null  object 
 5   Publisher     16540 non-null  object 
 6   NA_Sales      16598 non-null  float64
 7   EU_Sales      16598 non-null  float64
 8   JP_Sales      16598 non-null  float64
 9   Other_Sales   16598 non-null  float64
 10  Global_Sales  16598 non-null  float64
dtypes: float64(6), int64(1), object(4)
memory usage: 1.4+ MB
```

In [134…
```
df.columns
```

Out[134…
```
Index(['Rank', 'Name', 'Platform', 'Year', 'Genre', 'Publisher', 'NA_Sales',
       'EU_Sales', 'JP_Sales', 'Other_Sales', 'Global_Sales'],
      dtype='object')
```

Description of Data

1. '**Rank**' : Position of game in the dataset
2. '**Name**' : Name of the game
3. '**Platform**' : Platform where game was initially launched
4. '**Year**' : Year when game was launched
5. '**Genre**' : Type of the game
6. '**Publisher**' : Who published the game
7. '**NA_Sales**' : Sales of the game in North America
8. '**EU_Sales**' : Sales of the game in Europe
9. '**JP_Sales**' : Sales of the game in Japan
10. '**Other_Sales**' : Sales in other countries
11. '**Global_Sales**' : Sales on global scale(together)

Statistical description of Data

In [135…
```
df.describe().T

# T: convert rows to columns and vice-versa (Transpose)
```

Out[135…

	count	mean	std	min	25%	50%	75%	max
Rank	16598.0	8300.605254	4791.853933	1.00	4151.25	8300.50	12449.75	16600.00
Year	16327.0	2006.406443	5.828981	1980.00	2003.00	2007.00	2010.00	2020.00
NA_Sales	16598.0	0.264667	0.816683	0.00	0.00	0.08	0.24	41.49
EU_Sales	16598.0	0.146652	0.505351	0.00	0.00	0.02	0.11	29.02
JP_Sales	16598.0	0.077782	0.309291	0.00	0.00	0.00	0.04	10.22
Other_Sales	16598.0	0.048063	0.188588	0.00	0.00	0.01	0.04	10.57
Global_Sales	16598.0	0.537441	1.555028	0.01	0.06	0.17	0.47	82.74

In [136…
```
df.describe(include='O')
```

Out[136…

	Name	Platform	Genre	Publisher
count	16598	16598	16598	16540
unique	11493	31	12	578
top	Need for Speed: Most Wanted	DS	Action	Electronic Arts
freq	12	2163	3316	1351

Missing value Analysis

1. if number of rows that contain null values < 30% of entire dataset :

`drop the null rows` 2. if number of rows that contain null values > 30% of entire dataset :

a) `numerical column -> replace the null values with mean/median`

b) `object column -> replace the null values with mode`

In [137…
```
df.isna().sum()
```

Out[137…
```
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
dtype: int64
```

In [138…
```
df.isna().sum().sum()
```

Out[138… 329

In [139…
```
print('percentage of rows containing null values is: ', (329/16598)*100)
```

```
percentage of rows containing null values is:  1.9821665260874803
```

Since it is less than 30% of the eniter dataset -> drop the null rows

In [6]:
```
df.dropna(inplace=True)
```

In [7]:
```
df.isna().sum()
```

Out[7]:
```
Rank            0
Name            0
Platform        0
Year            0
Genre           0
Publisher       0
NA_Sales        0
EU_Sales        0
JP_Sales        0
Other_Sales     0
Global_Sales    0
dtype: int64
```

In [142…
```
df.shape
```

Out[142… (16291, 11)

Duplicate Rows Analysis

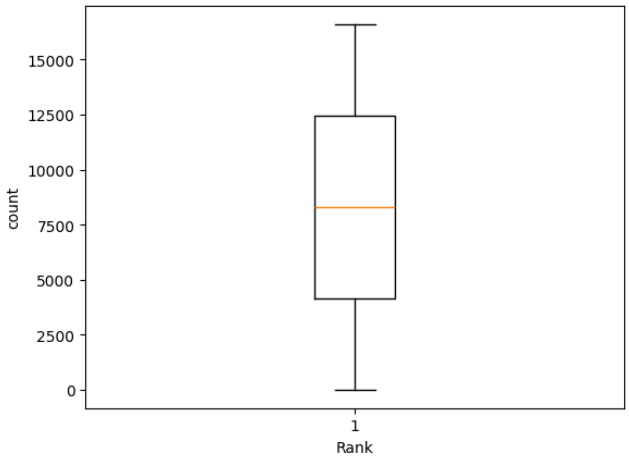
In [8]:
```
df.duplicated().sum()
```

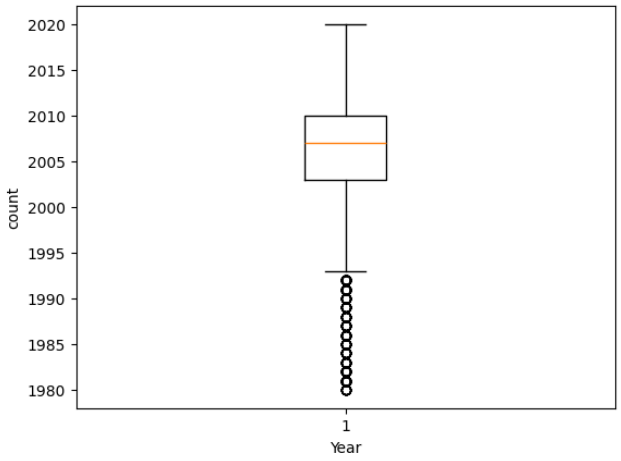
Out[8]: 0

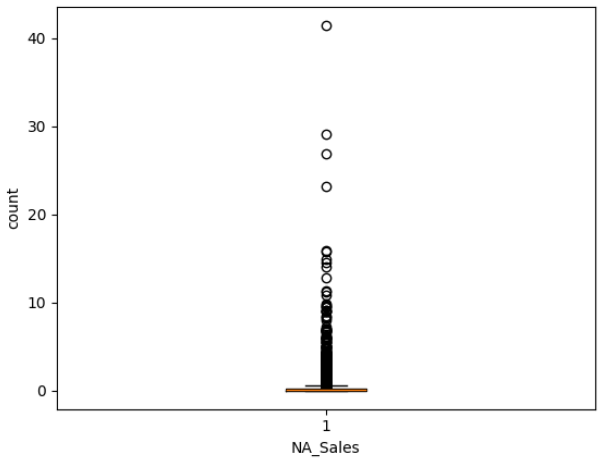
In [144…
```
#in case there are duplicates -> df.drop_duplicated(inplace=True)
```

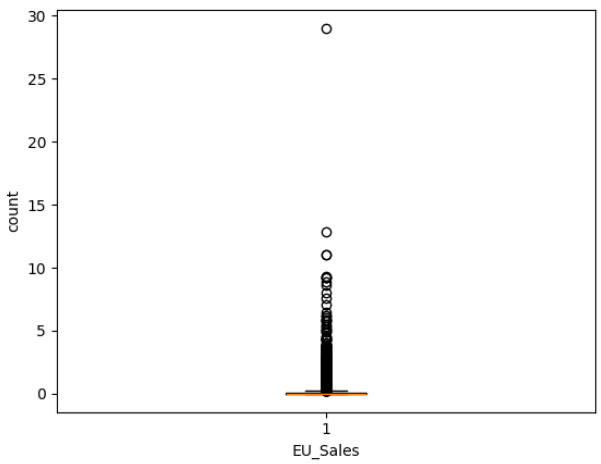
Outliers Detection

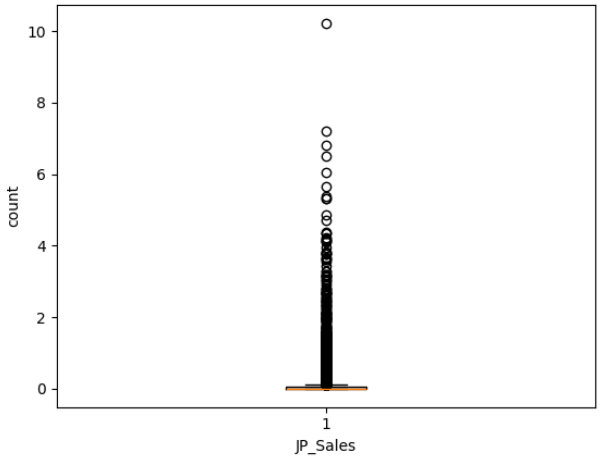
In [9]:
```
for i in df.columns:
  if df[i].dtype != 'object':
    plt.boxplot(df[i])
    plt.xlabel(i)
    plt.ylabel('count')
    plt.show()
```

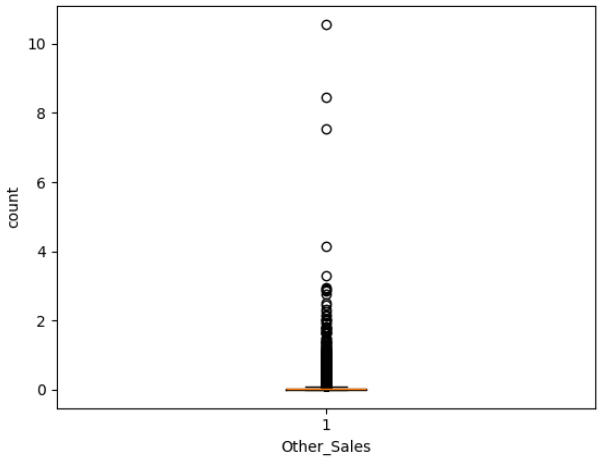


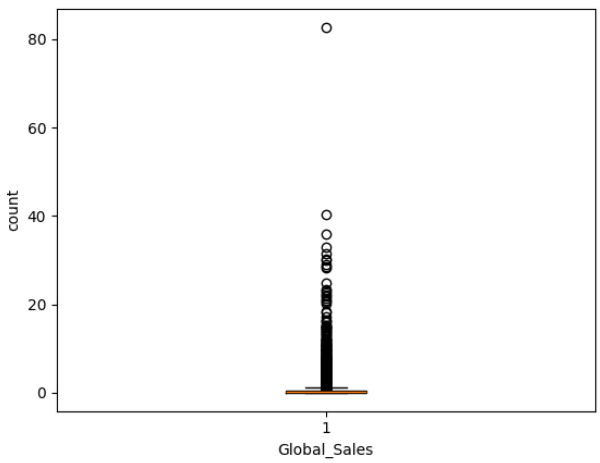










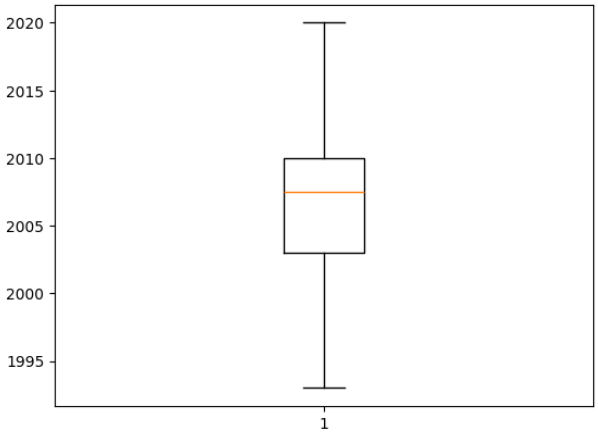


Removal of Outliers : IQR Method

For one column

In [10]:
```
Q1 = df['Year'].quantile(0.25)
Q3 = df['Year'].quantile(0.75)

IQR = Q3 - Q1

LF = Q1 - 1.5*IQR
UF = Q3 + 1.5*IQR

df = df[(df['Year'] > LF) & (df['Year'] < UF)]
```

In [11]:
```
plt.boxplot(df['Year'])
plt.show()
```



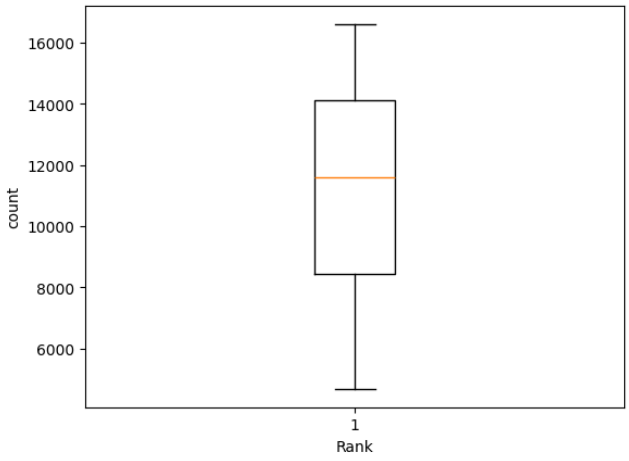
For all columns together

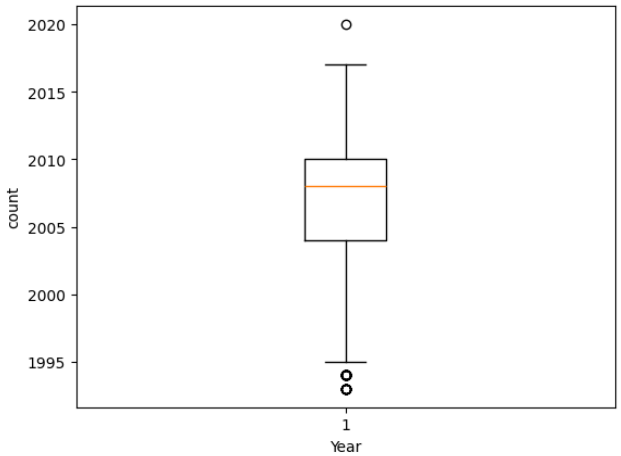
In [12]:
```
col_list = ['NA_Sales','EU_Sales','JP_Sales','Other_Sales','Global_Sales']
```

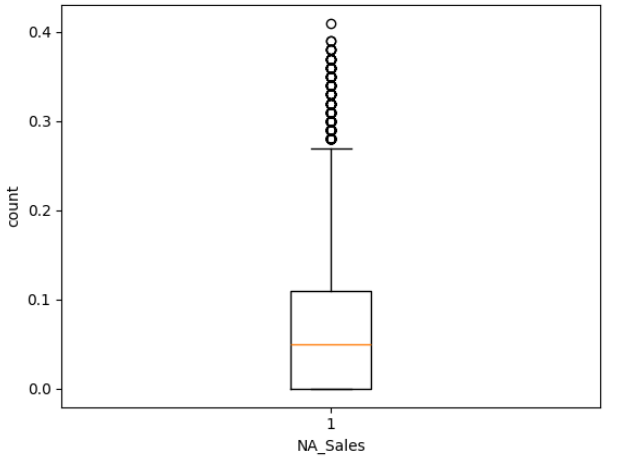
In [13]:
```
for i in col_list:
  Q1 = df[i].quantile(0.25)
  Q3 = df[i].quantile(0.75)
  IQR = Q3 - Q1
  LF = Q1 - 1.5*IQR
  UF = Q3 + 1.5*IQR
  df = df[(df[i] > LF) & (df[i] < UF)]
```

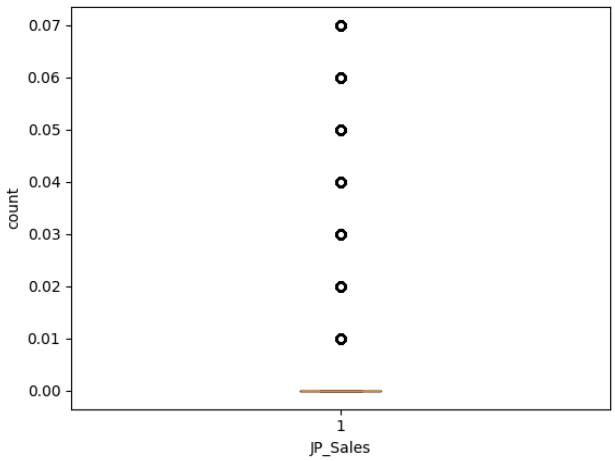
to check how data looks after removal of outlier

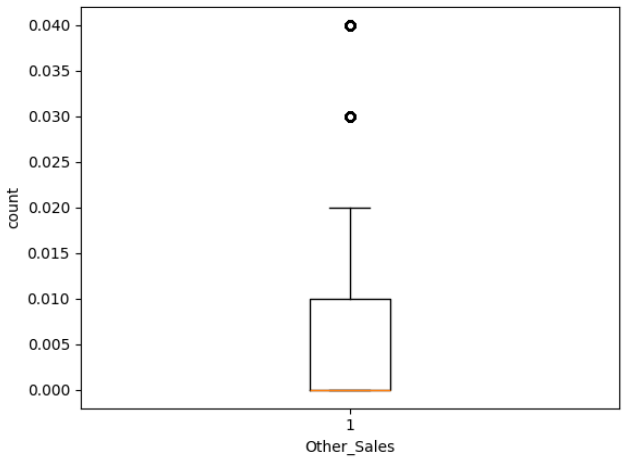
In [14]:
```
for i in df.columns:
  if df[i].dtype != 'object':
    plt.boxplot(df[i])
    plt.xlabel(i)
    plt.ylabel('count')
    plt.show()
```

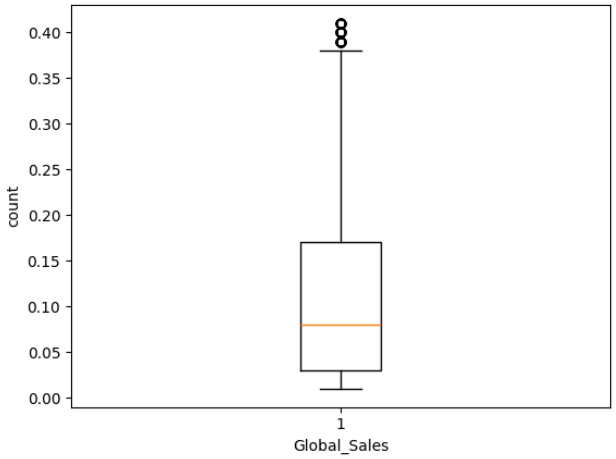












In [151…
```
df.shape
```

Out[151… (9677, 11)

Let's analyze what values our data holds

In [15]:
```
#unique()

#Platform, Year, Genre, Publisher

df['Genre'].unique()
```

Out[15]:
```
array(['Shooter', 'Action', 'Adventure', 'Racing', 'Misc', 'Sports',
       'Simulation', 'Platform', 'Strategy', 'Fighting', 'Role-Playing',
       'Puzzle'], dtype=object)
```

In [153…
```
df['Genre'].nunique()
```

Out[153… 12

In [154…
```
df['Genre'].value_counts()
```

Out[154…
```
Genre
Action          1899
Sports          1253
Misc            1076
Adventure       1014
Racing           780
Role-Playing     702
Shooter          699
Simulation       530
Platform         467
Strategy         442
Fighting         417
Puzzle           398
Name: count, dtype: int64
```

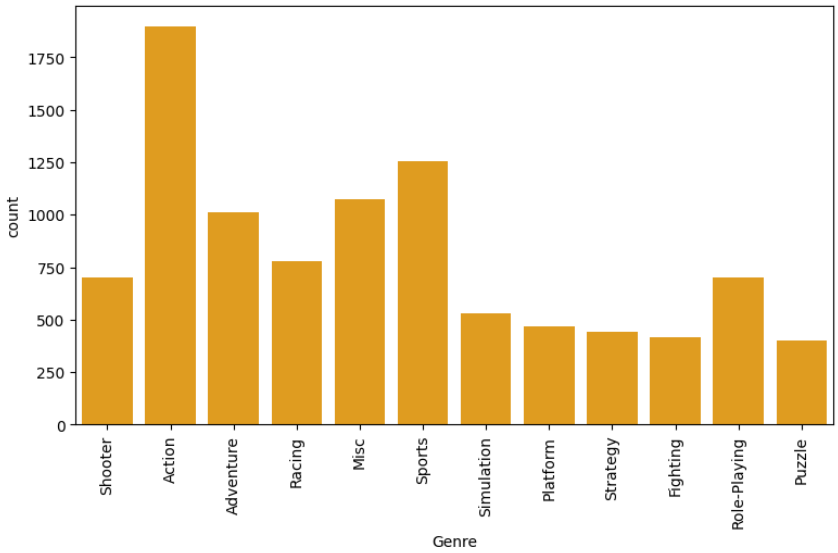
Visualization Analysis of Data

Q1. What type of video game has been the most popular in terms of production volume?

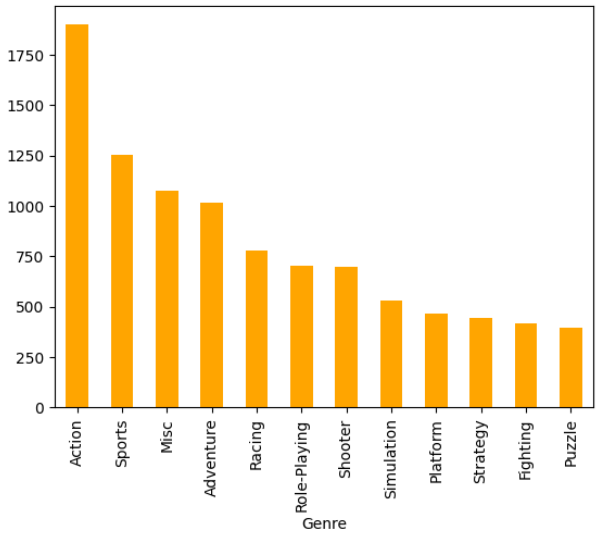
In [16]:
```
top_genre = df['Genre'].value_counts().sort_values(ascending=False)
top_genre.head(3)
```

Out[16]:
```
Genre
Action    1899
Sports    1253
Misc      1076
Name: count, dtype: int64
```

In [24]:
```
plt.figure(figsize=(9,5))
sns.countplot(x='Genre', data=df, color= 'orange')
plt.xticks(rotation=90)
plt.show()
```



In [164…
```
df['Genre'].value_counts().plot(kind='bar', color='orange')
plt.show()
```



Inference:

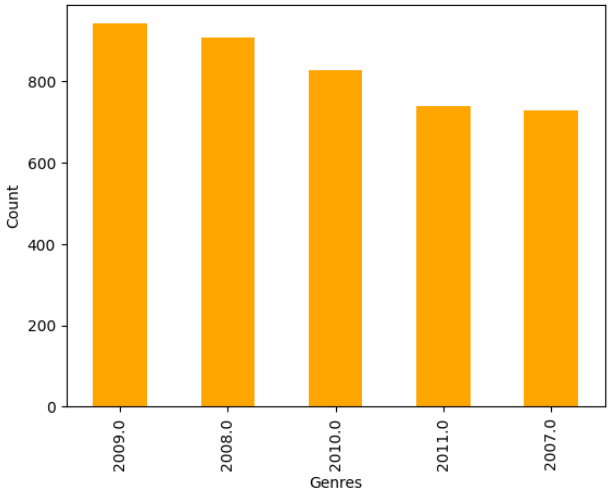
Action and Sports are the two most popular Genres as per production volumns

Q2. What year witnessed the greatest volume of video game launches?

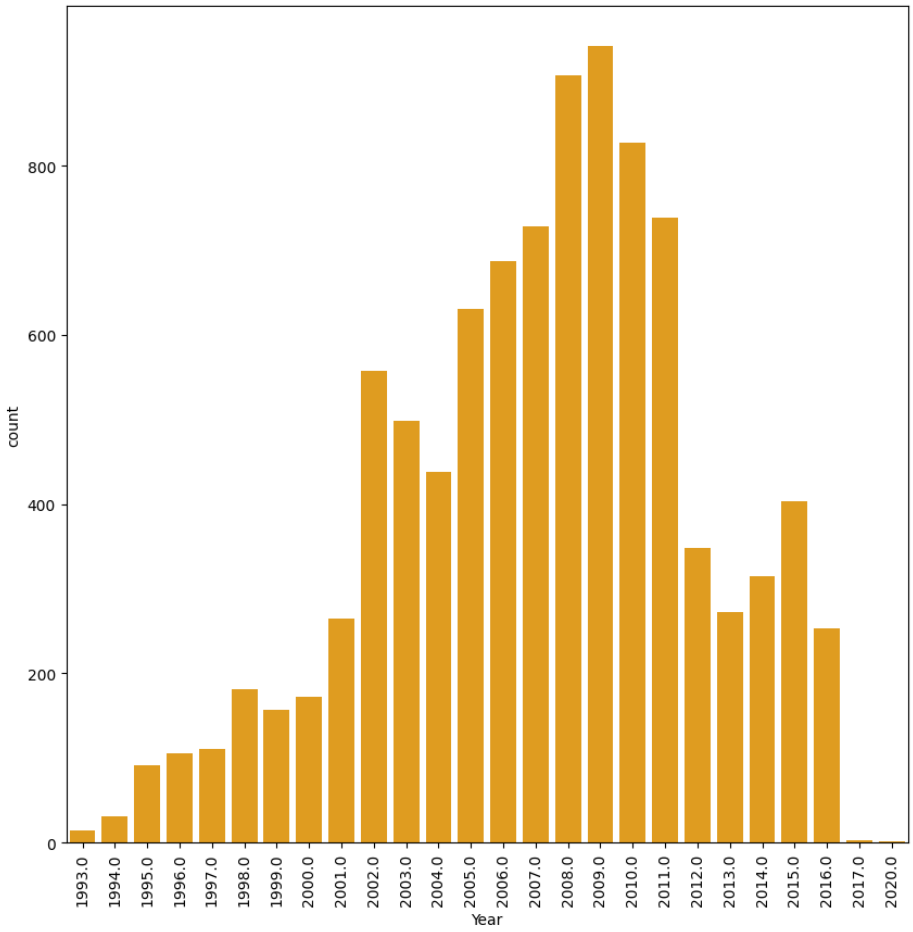
Plot for just top 5

In [187…
```
df['Year'].value_counts().head().plot(kind='bar', color='orange')
plt.xlabel('Genres')
plt.ylabel('Count')
```

Out[187…
```
Text(0, 0.5, 'Count')
```



In [190…
```
plt.figure(figsize=(10,10))
sns.countplot(x='Year', data=df, color='orange')
plt.xticks(rotation=90)
plt.show()
```



Q3. What year experienced the most lucrative sales performance on a global scale?

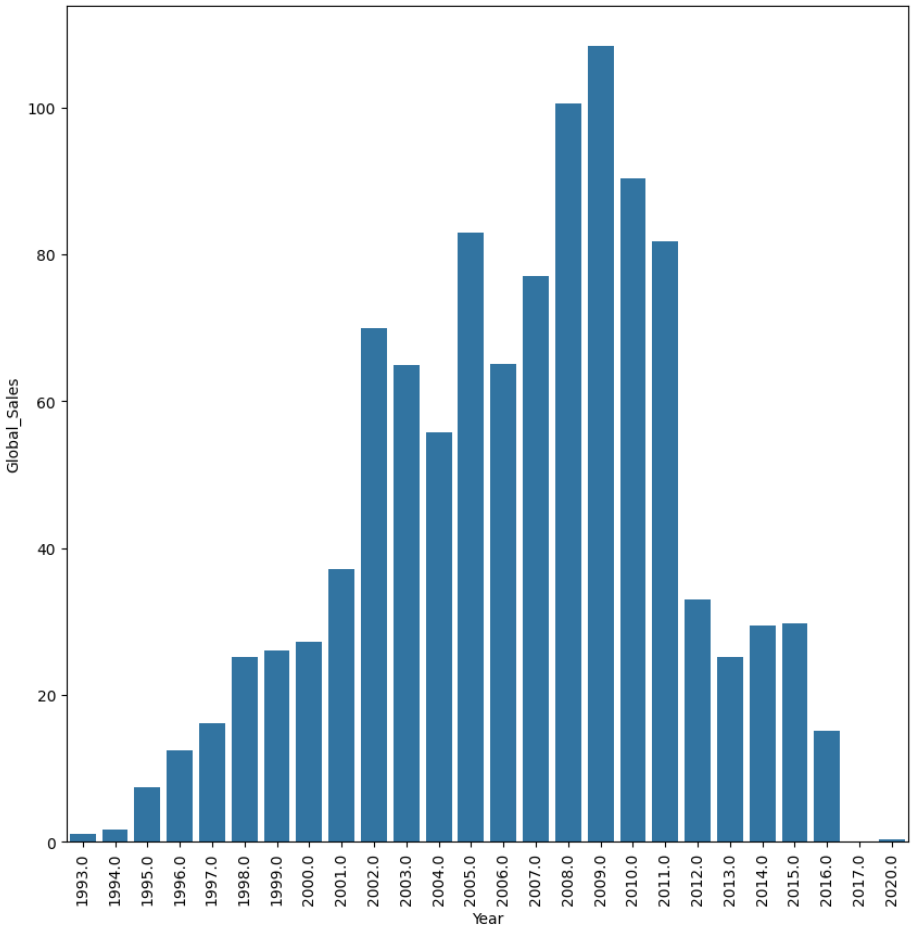
In [212…
```
#year, global_sales

year_sales= df.groupby('Year')['Global_Sales'].sum()
# year_sales
year_sales = year_sales.reset_index()
year_sales
```

Out[212…

	Year	Global_Sales
0	1993.0	1.09
1	1994.0	1.66
2	1995.0	7.46
3	1996.0	12.39
4	1997.0	16.18
5	1998.0	25.14

6	1999.0	26.05
7	2000.0	27.24
8	2001.0	37.16
9	2002.0	70.00
10	2003.0	64.93
11	2004.0	55.76
12	2005.0	82.89
13	2006.0	65.11
14	2007.0	77.09
15	2008.0	100.56
16	2009.0	108.41
17	2010.0	90.41
18	2011.0	81.77
19	2012.0	32.94
20	2013.0	25.20
21	2014.0	29.43
22	2015.0	29.73
23	2016.0	15.04
24	2017.0	0.05
25	2020.0	0.29

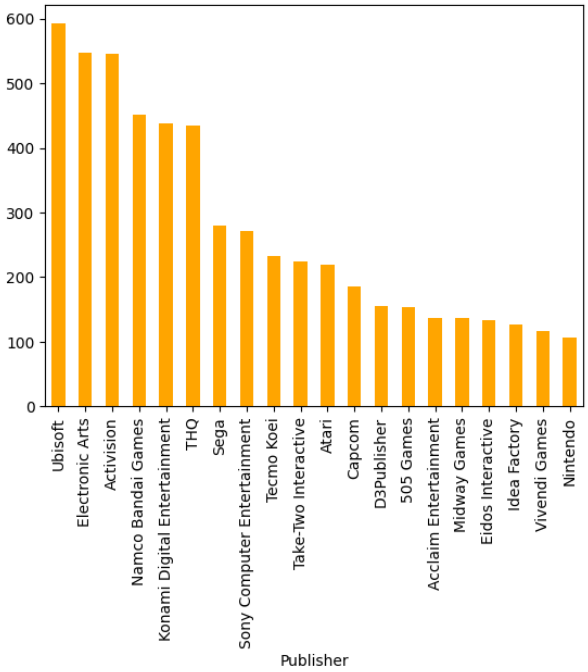
In [211…
```
plt.figure(figsize=(10,10))
sns.barplot(x='Year', y='Global_Sales', data=year_sales)
plt.xticks(rotation=90)
plt.show()
```



Q4. Who are the top publisher of the games in the present dataset. Find the top 20 publishers.

In [197…
```
df['Publisher'].value_counts().head(20).plot(kind='bar', color='orange')
```

Out[197… <Axes: xlabel='Publisher'>



In [ ]: