Day_40_061223

January 23, 2024

1 Data Visualization

[1]:
```
import pandas as pd
import numpy as np
```

2 Importing matplotlib and seaborn libraries

[2]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

3 Downloading the CSV File using URL

[3]:
```
!gdown 15I3g3TBZvN6-WxLWMwFi1_h8oeT6gA7G
```

```
Downloading…
From: https://drive.google.com/uc?id=15I3g3TBZvN6-WxLWMwFi1_h8oeT6gA7G
To: C:\Data\Data_science\Data Science RIA\3 Python\3 Data Visualization -
Matplotlib and Seaborn\Codes\final_vg.csv

  0%|          | 0.00/2.15M [00:00<?, ?B/s]
 24%|##4       | 524k/2.15M [00:00<00:00, 2.27MB/s]
 98%|#########7| 2.10M/2.15M [00:00<00:00, 7.08MB/s]
100%|##########| 2.15M/2.15M [00:00<00:00, 6.13MB/s]
```

4 Reading the CSV File

[4]:
```
data = pd.read_csv("final_vg.csv")
```

[5]:
```
data.head()
```

[5]:

	Unnamed: 0	Rank	Name	Platform	Year	\
0	0	2061	1942	NES	1985.0	
1	1	9137	¡Shin Chan Flipa en colores!	DS	2007.0	
2	2	14279	.hack: Sekai no Mukou ni + Versus	PS3	2012.0	
3	3	8359	.hack//G.U. Vol.1//Rebirth	PS2	2006.0	
4	4	7109	.hack//G.U. Vol.2//Reminisce	PS2	2006.0	

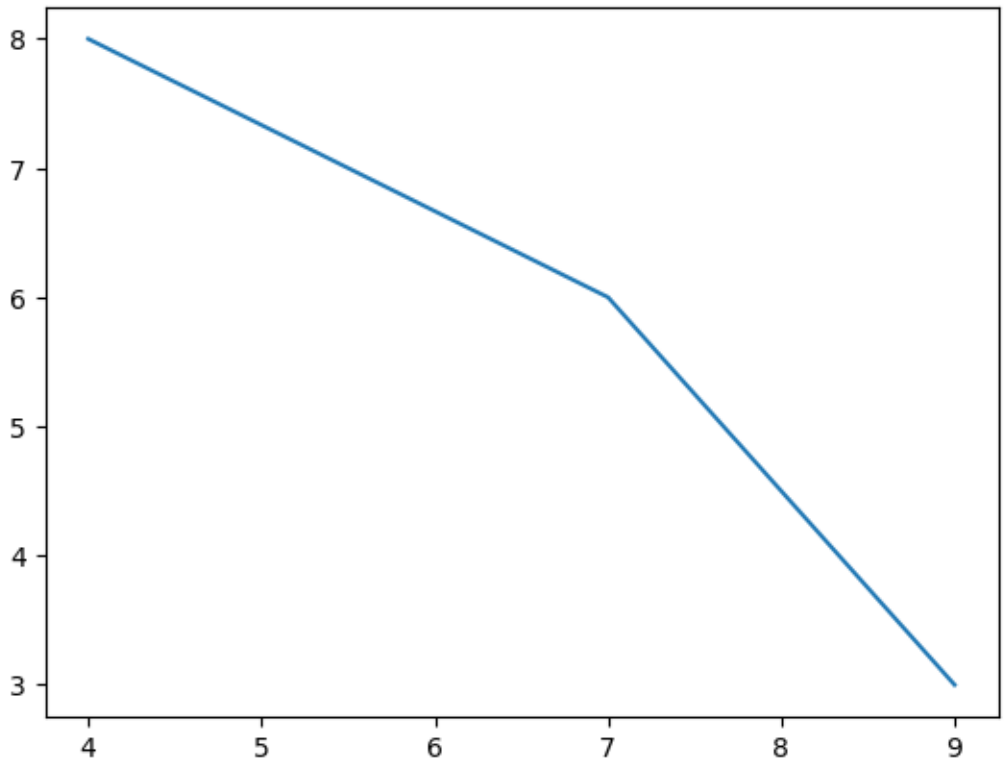
```
          Genre           Publisher  NA_Sales  EU_Sales  JP_Sales  \
0       Shooter              Capcom  4.569217  3.033887  3.439352
1      Platform           505 Games  2.076955  1.493442  3.033887
2        Action  Namco Bandai Games  1.145709  1.762339  1.493442
3  Role-Playing  Namco Bandai Games  2.031986  1.389856  3.228043
4  Role-Playing  Namco Bandai Games  2.792725  2.592054  1.440483

   Other_Sales  Global_Sales
0     1.991671     12.802935
1     0.394830      7.034163
2     0.408693      4.982552
3     0.394830      7.226880
4     1.493442      8.363113
```

5 Plotting general points

[6]:
```
x = [4,7,9]
y = [8,6,3]
plt.plot(x,y)
```

[6]:
```
[<matplotlib.lines.Line2D at 0x25e8e668b00>]
```



6 Find the top 5 geners of video games

7 Univariate Analysis - Categorical

- Distribution of Each category
- What proportion each category has on the total

[7]:
```
data['Genre']
```

[7]:
```
0             Shooter
1            Platform
2              Action
3        Role-Playing
4        Role-Playing
             …
16647          Sports
16648            Misc
16649            Misc
16650    Role-Playing
16651          Action
Name: Genre, Length: 16652, dtype: object
```

[8]:
```
cat_count = data['Genre'].value_counts().sort_values(ascending=False)
cat_count
```

[8]:
```
Genre
Action          3316
Sports          2400
Misc            1739
Role-Playing    1488
Shooter         1310
Adventure       1286
Racing          1249
Platform         886
Simulation       867
Fighting         848
Strategy         681
Puzzle           582
Name: count, dtype: int64
```

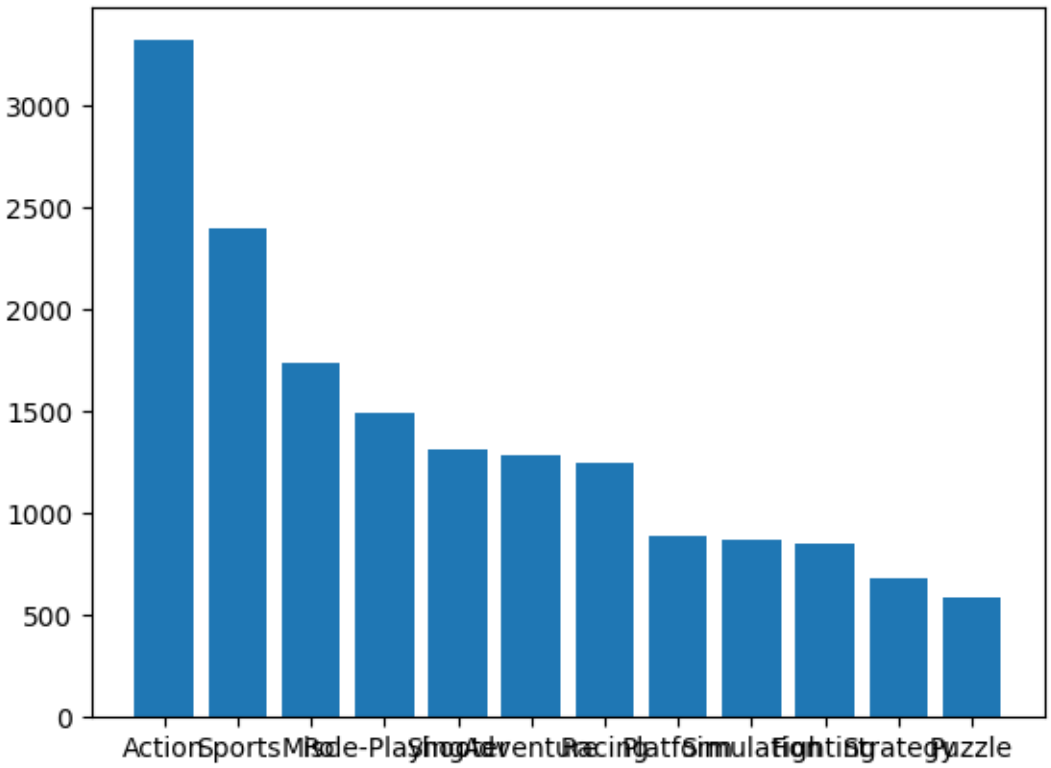
[9]:
```
cat_count.index
```

[9]:
```
Index(['Action', 'Sports', 'Misc', 'Role-Playing', 'Shooter', 'Adventure',
       'Racing', 'Platform', 'Simulation', 'Fighting', 'Strategy', 'Puzzle'],
      dtype='object', name='Genre')
```

7.1 Bar chart to visualize the distribution

[10]:
```
x_bar = cat_count.index
y_bar = cat_count
plt.bar(x_bar,y_bar)
```

[10]:
```
<BarContainer object of 12 artists>
```



7.1.1 Here the names on the x axis is bit messy so

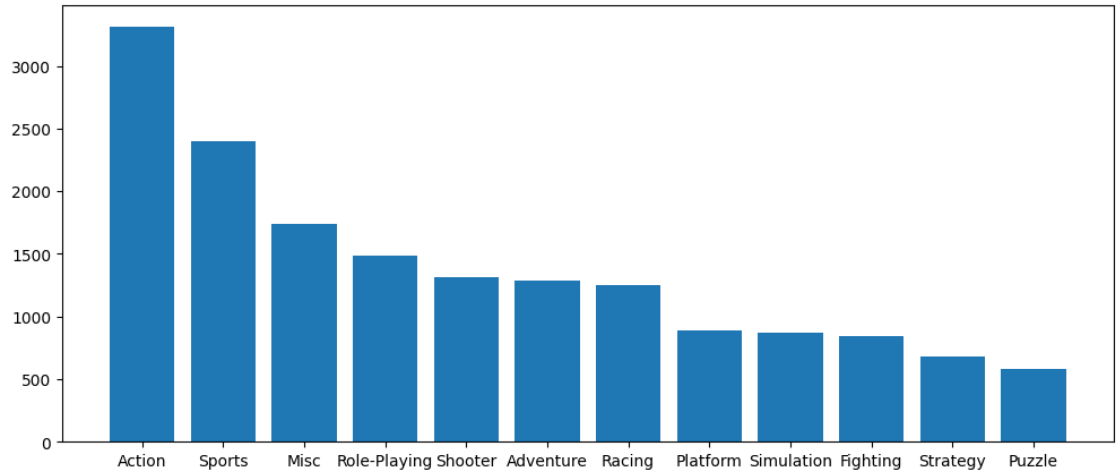
So there are two way

- Increasing the size of the plot
- Rotating the xlabels to certain angles

7.1.2 Figure size

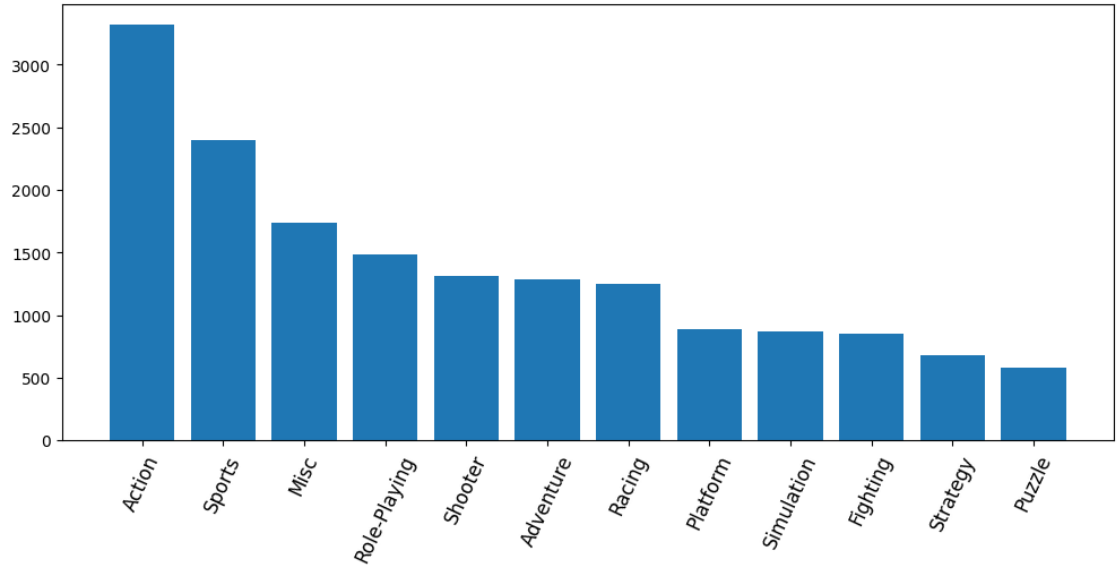
[11]:
```
plt.figure(figsize=(12,5))
x_bar = cat_count.index
y_bar = cat_count
plt.bar(x_bar,y_bar)
```

[11]:
```
<BarContainer object of 12 artists>
```



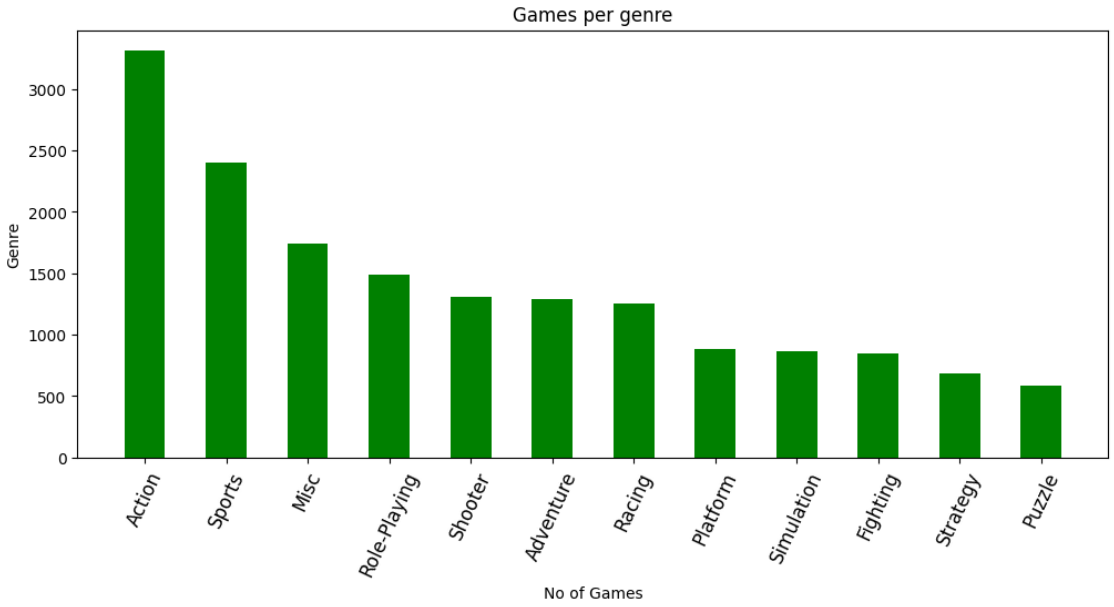
7.1.3 Rotating the label with angle

[12]:
```
plt.figure(figsize=(12,5))
x_bar = cat_count.index
y_bar = cat_count
plt.bar(x_bar,y_bar)
plt.xticks(rotation=65,fontsize=12)
plt.show()
```



7.2 Adding title, x and y lables

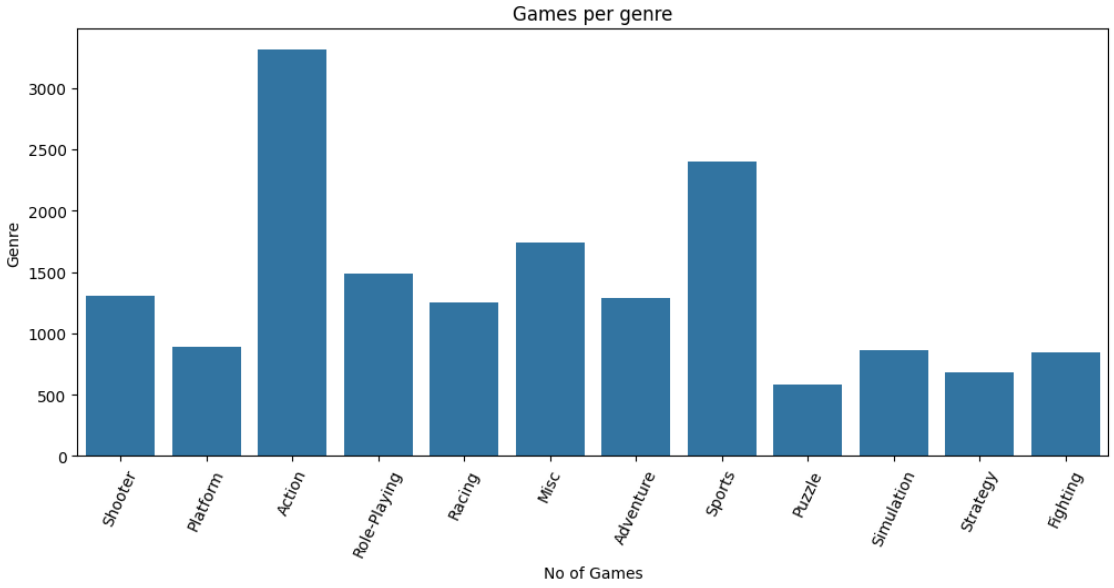
[13]:
```
plt.figure(figsize=(12,5))
x_bar = cat_count.index
y_bar = cat_count
# Here in bar plot we can add color, width of bar
plt.bar(x_bar,y_bar,color='green',width=0.5)
plt.title("Games per genre")
plt.xlabel("No of Games")
plt.ylabel("Genre")
plt.xticks(rotation=65,fontsize=12)
plt.show()
```



Upto here we wrote a lot of code using matplotlib library but using seaborn is much more simple

8 Plotting distribution in seaborn

[14]:
```
plt.figure(figsize=(12,5))
plt.title("Games per genre")
plt.xlabel("No of Games")
plt.ylabel("Genre")
sns.countplot(x='Genre',data=data)
plt.xticks(rotation=65)
plt.savefig("Plots/GamesPerGenre.jpg") # Saving the figure to local directory
plt.show()
```



9 Contribution to the total

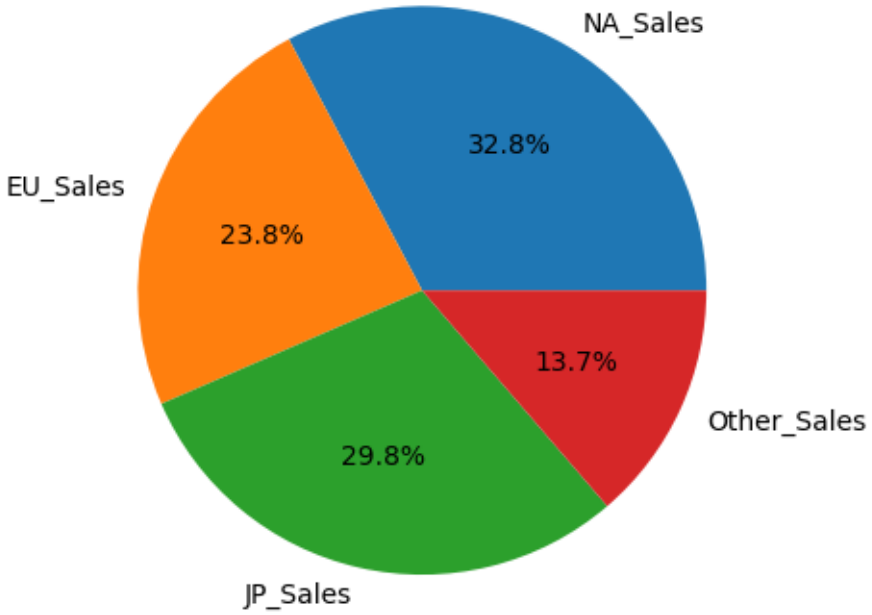
[15]:
```
data.columns
```

[15]:
```
Index(['Unnamed: 0', 'Rank', 'Name', 'Platform', 'Year', 'Genre', 'Publisher',
       'NA_Sales', 'EU_Sales', 'JP_Sales', 'Other_Sales', 'Global_Sales'],
      dtype='object')
```

[16]:
```
sales_data = data[['NA_Sales','EU_Sales','JP_Sales','Other_Sales']]
total_sales = sales_data.sum(axis=0)
total_sales
```

[16]:
```
NA_Sales       45831.525845
EU_Sales       33251.970702
JP_Sales       41624.625635
Other_Sales    19180.256828
dtype: float64
```

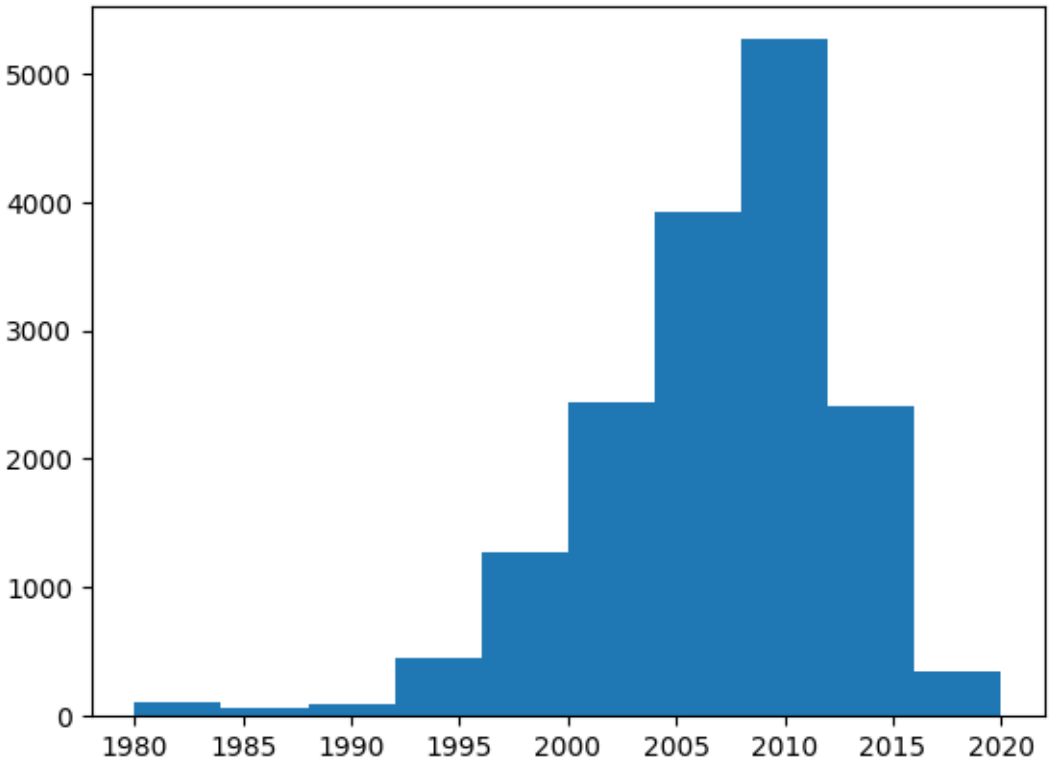
[17]:
```
plt.
 ↪pie(total_sales,labels=['NA_Sales','EU_Sales','JP_Sales','Other_Sales'],autopct='%1.
 ↪1f%%')
plt.savefig("Plots/Contributiontosales.jpg")
plt.show()
```



10 Univariate Analysis - Numerical

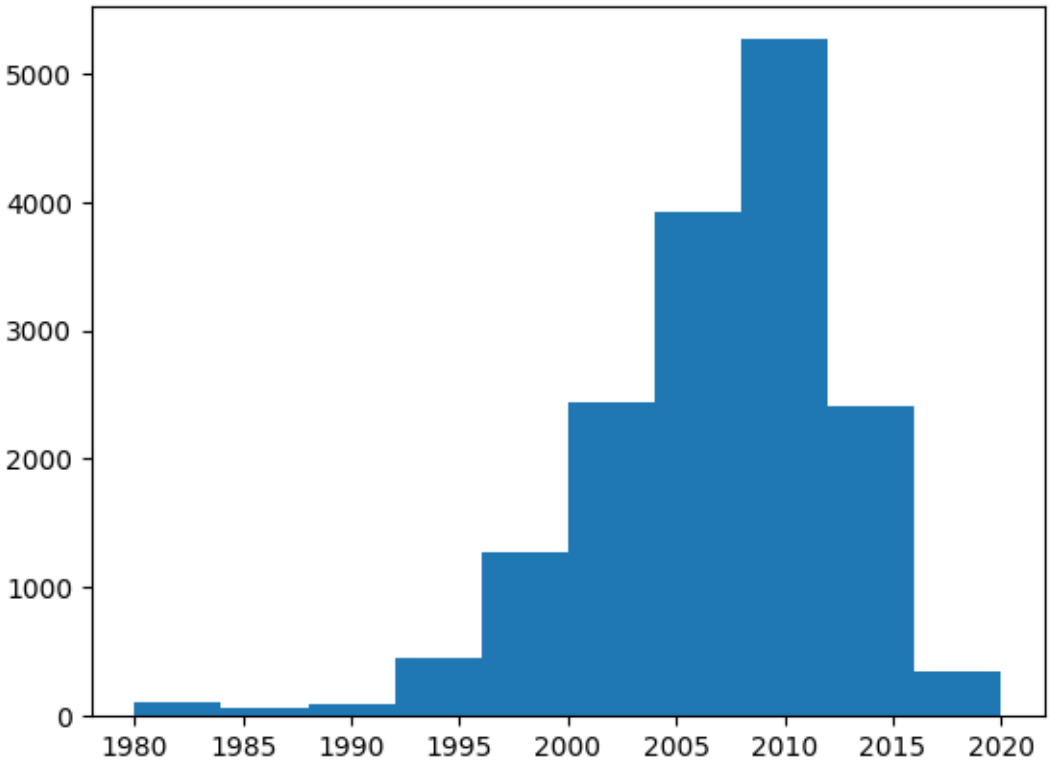
10.1 How to identify the popularity of a video game year by year

[18]:
```
plt.hist(data['Year'])
plt.show()
```



10.2 We can reduce or increase the bars by using bins

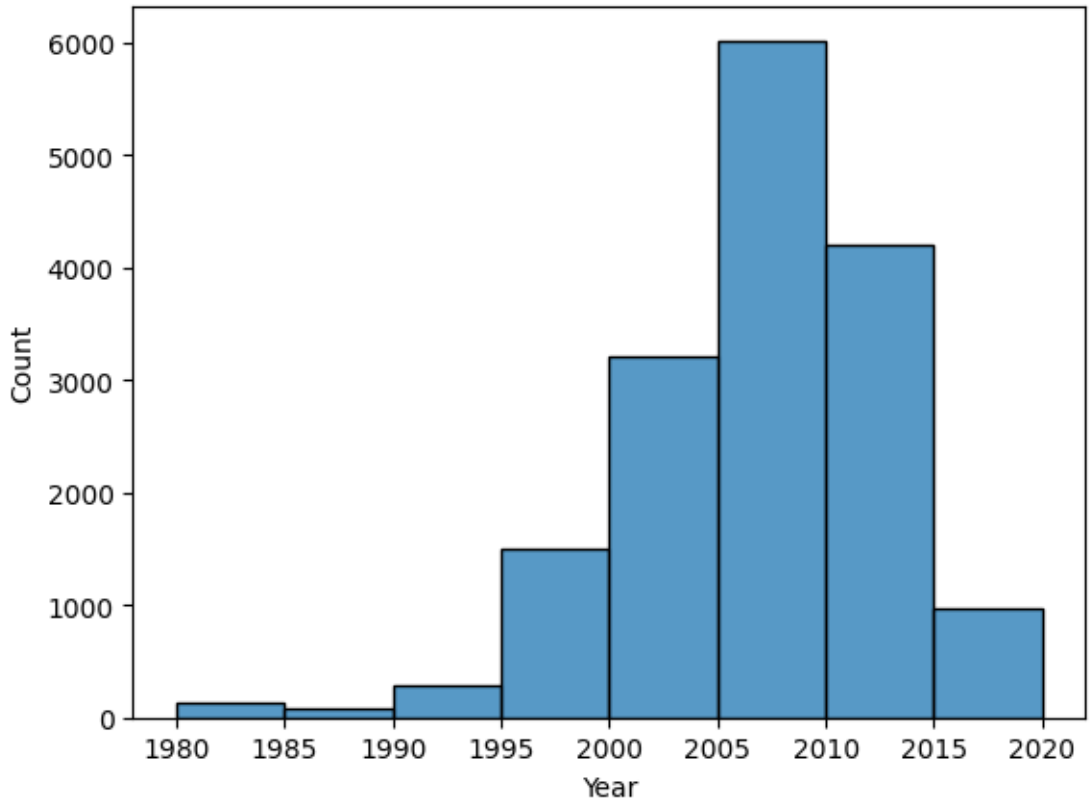
[19]:
```
plt.hist(data['Year'],bins=10)
plt.show()
```



11 Using seaborn

[20]:
```
sns.histplot(data['Year'],bins=8)
```

[20]:
```
<Axes: xlabel='Year', ylabel='Count'>
```



12 Bi variate Analysis

- It has 2 types of data points

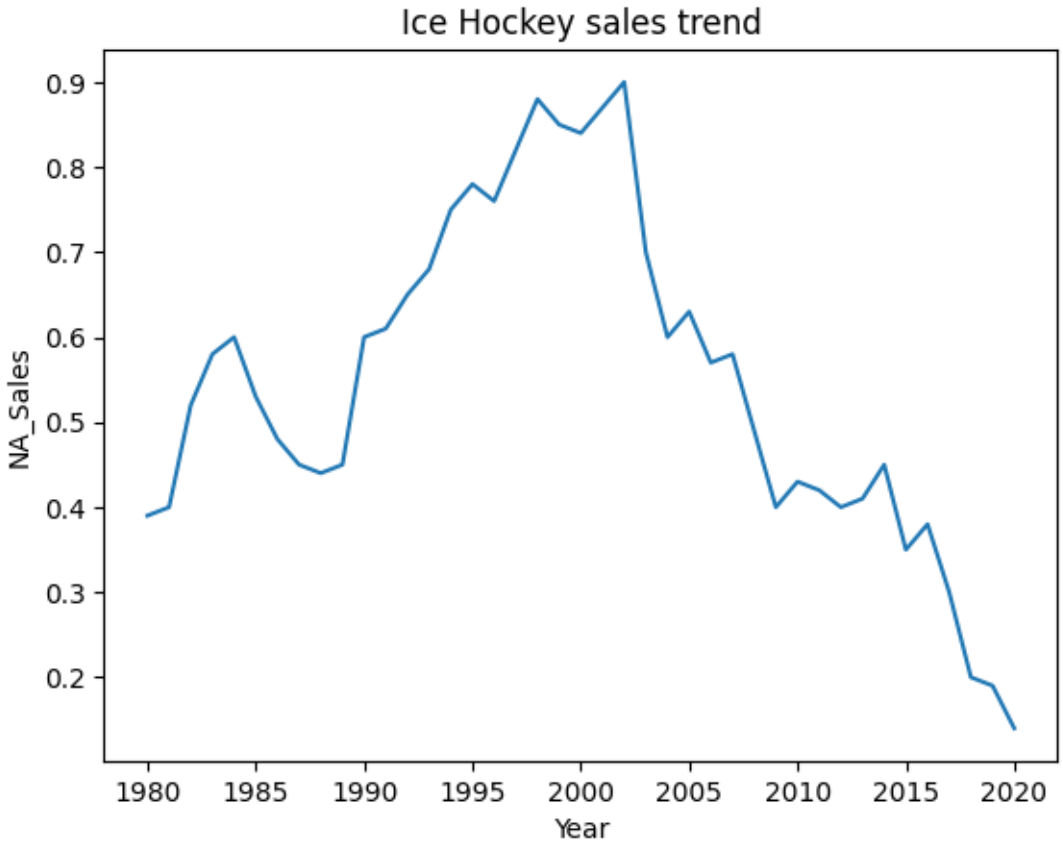
[21]:
```
data.loc[data['Name']=='Ice Hockey'].head()
```

[21]:
```
      Unnamed: 0  Rank         Name Platform    Year   Genre   Publisher  \
6073        6073   639  Ice Hockey      NES  1988.0  Sports    Nintendo
6074        6074  4027  Ice Hockey     2600  1980.0  Sports  Activision
6075        6075  4149  Ice Hockey     2600  1991.0  Sports  Activision
6076        6076  4149  Ice Hockey     2600  1992.0  Sports  Activision
6077        6077  4149  Ice Hockey     SNES  1993.0  Sports  Activision

      NA_Sales  EU_Sales  JP_Sales  Other_Sales  Global_Sales
6073      0.44  3.860566  4.751539     2.004268     15.855389
6074      0.39  1.493442  2.741701     0.394830      4.956249
6075      0.61  0.020000  0.000000     0.010000      0.470000
6076      0.65  0.020000  0.000000     0.010000      0.470000
6077      0.68  0.020000  0.000000     0.010000      0.470000
```

13 Getting the Sales trend of Ice Hockey

[22]:
```
plt.title("Ice Hockey sales trend")
ih = data.loc[data['Name']=='Ice Hockey']
sns.lineplot(x='Year',y='NA_Sales',data=ih)
plt.savefig("Plots/Ice Hockey sales trend.jpg")
plt.show()
```



14 Including two or more trends in one plot

[23]:
```
baseball = data.loc[data['Name']=='Baseball']
baseball.head()
```

[23]:

	Unnamed: 0	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	\
941	941	324	Baseball	NES	1980.0	Sports	Nintendo	0.459000	
942	942	422	Baseball	NES	1983.0	Sports	Nintendo	0.468529	
943	943	231	Baseball	GB	1985.0	Sports	Nintendo	0.473000	
944	944	1144	Baseball	GB	1989.0	Sports	Nintendo	0.478448	
945	945	134	Baseball	GB	1992.0	Sports	Nintendo	0.520000	

	EU_Sales	JP_Sales	Other_Sales	Global_Sales
941	2.320000	5.230000	1.230000	9.239000
942	2.697415	5.854415	1.087977	10.108336
943	3.074830	6.478831	0.945954	10.972614
944	3.452245	7.103246	0.803931	11.837870
945	3.829660	7.727661	0.661908	12.739229

[24]:
```
sns.lineplot(x='Year',y='NA_Sales',data=baseball,color='r',label='Base ball NA␣
 ↪Sales')
sns.lineplot(x='Year',y='NA_Sales',data=ih,color='b',label='Ice Hockey NA␣
 ↪Sales')
plt.legend(loc=(-0.5,0.8)) # Changing the location of legend
plt.show()
```