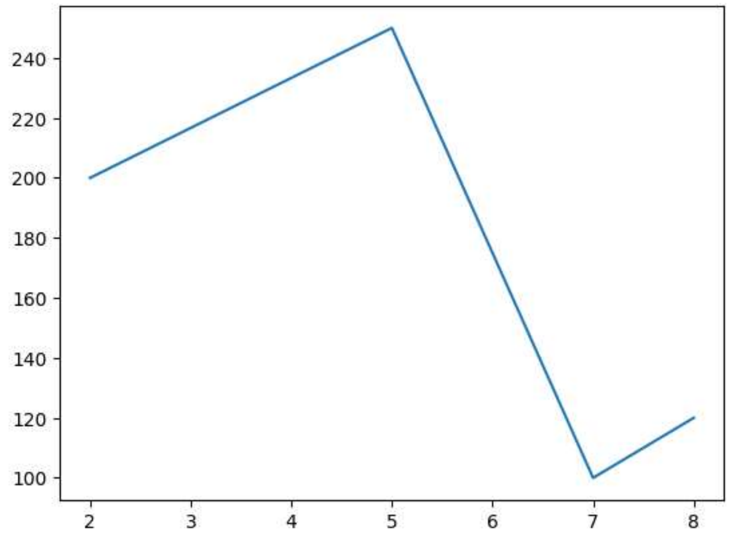
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
```

```
x = [2,5,7,8]
y = [200,250,100,120]
plt.plot(x,y)  #line plot
plt.show() #to not show - [<matplotlib.lines.Line2D at 0x78ee9af363b0>]
```



```
!gdown https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/021/299/original/final_vg1_-_final_vg_%281%29.csv?1670840166 -O final_
```

```
Downloading...
From: https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/021/299/original/final_vg1_-_final_vg_%281%29.csv?1670840166
To: /content/final_vg.csv
100% 2.04M/2.04M [00:00<00:00, 28.5MB/s]
```

```
data = pd.read_csv('final_vg.csv')
data.head()
```

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sa
0	2061	1942	NES	1985.0	Shooter	Capcom	4.569217	3.033887	3.439
1	9137	¡Shin Chan Flipa en colores!	DS	2007.0	Platform	505 Games	2.076955	1.493442	3.033
2	14279	.hack: Sekai no Mukou ni + Versus	PS3	2012.0	Action	Namco Bandai Games	1.145709	1.762339	1.493

```
#to find top 5 genres?
```

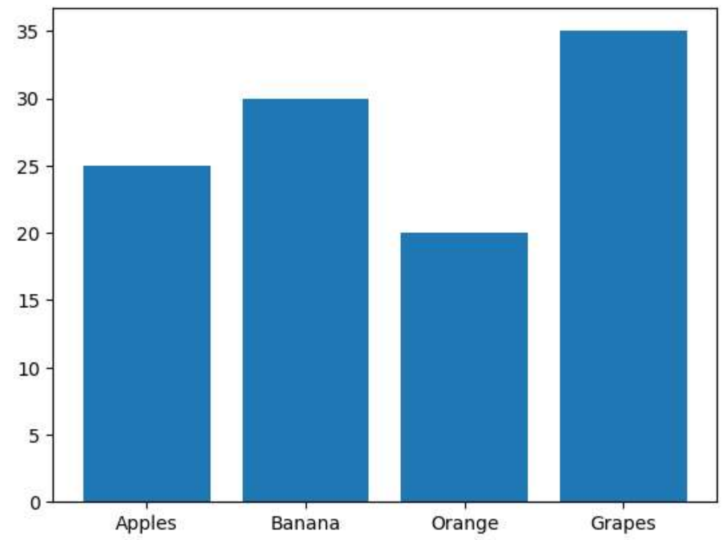
```
df = data
```

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

```
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: Genre, dtype: int64
```

```
#Bar Chart


fruits = ['Apples','Banana','Orange','Grapes']
sales = [25,30,20,35]
plt.bar(fruits,sales)
```

```
<BarContainer object of 4 artists>
```



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

```
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: Genre, dtype: int64
```

```
df['Genre'].value_counts().reset_index()
```

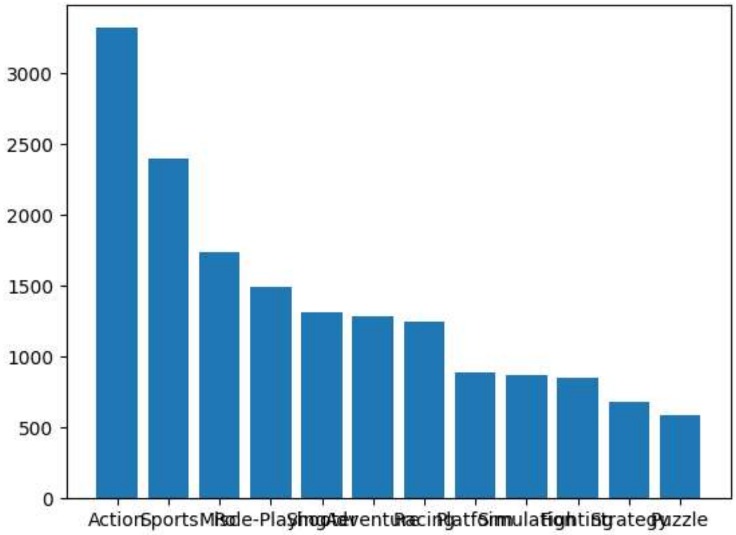
	index	Genre
0	Action	3316
1	Sports	2400
2	Misc	1739
3	Role-Playing	1488
4	Shooter	1310
5	Adventure	1286
6	Racing	1249
7	Platform	886
8	Simulation	867
9	Fighting	848
10	Strategy	681
11	Puzzle	582

```
#instead of reset_index() -- .index() -- you will get index directly
df['Genre'].value_counts().index
```

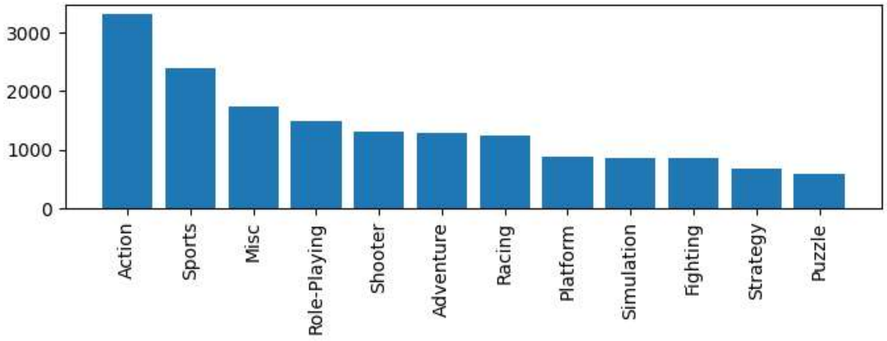
```
Index(['Action', 'Sports', 'Misc', 'Role-Playing', 'Shooter', 'Adventure',
       'Racing', 'Platform', 'Simulation', 'Fighting', 'Strategy', 'Puzzle'],
           dtype='object')
```

```
plt.bar(df['Genre'].value_counts().index,df['Genre'].value_counts())
```

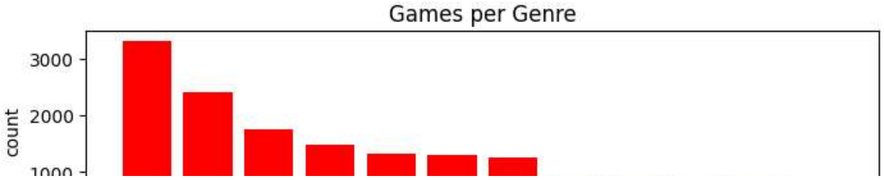
```
<BarContainer object of 12 artists>
```



```
#plt.figure(figsize=(L,H))

plt.figure(figsize=(8,2))
labels  =df['Genre'].value_counts().index
values = df['Genre'].value_counts()
plt.bar(labels,values)
plt.xticks(rotation=90,fontsize=10)
plt.show() #REMOVE EXTRA INFO
```



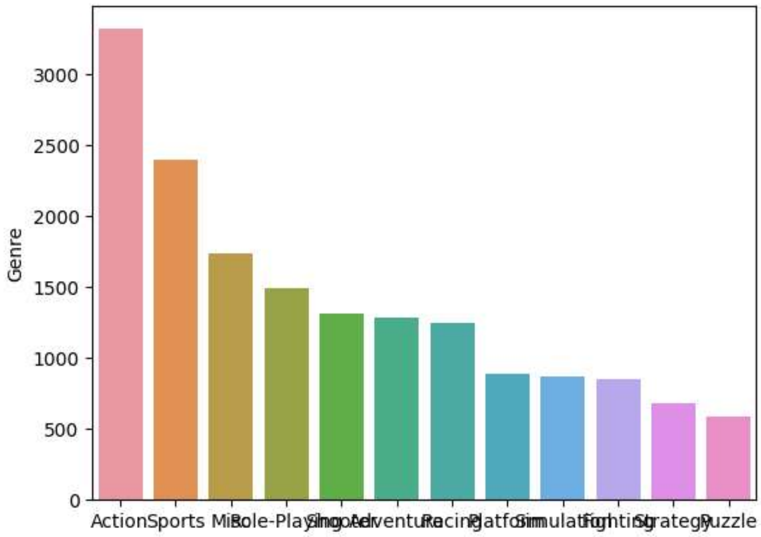
```
#plt.figure(figsize=(L,H))

plt.figure(figsize=(8,2))
labels  =df['Genre'].value_counts().index
values = df['Genre'].value_counts()


plt.bar(labels,values,width=0.8,color='red')
plt.xticks(rotation=90,fontsize=10)
plt.xlabel("Genre")
plt.ylabel("count")
plt.title("Games per Genre")
plt.show()
```



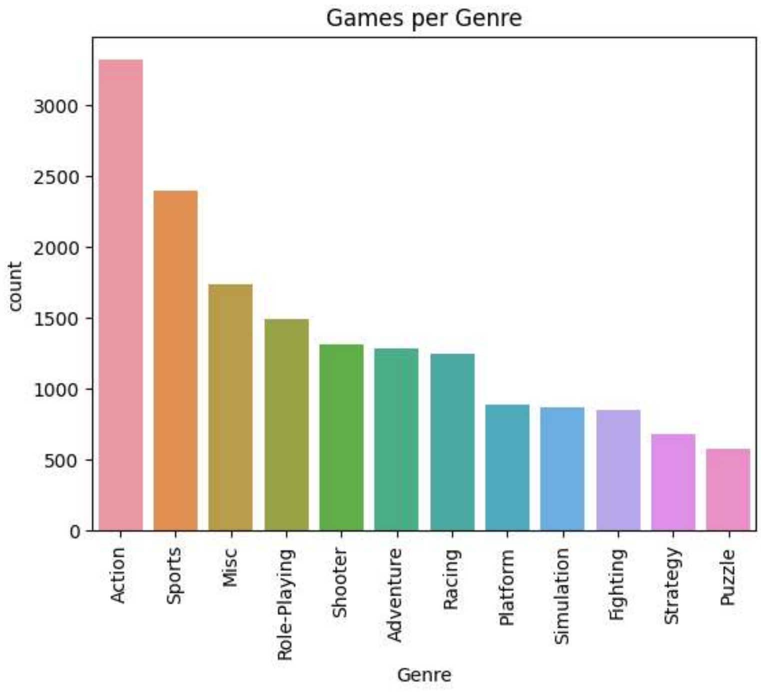
seaborn

```
sns.barplot(x = labels,y = values)
```

```
<Axes: ylabel='Genre'>
```

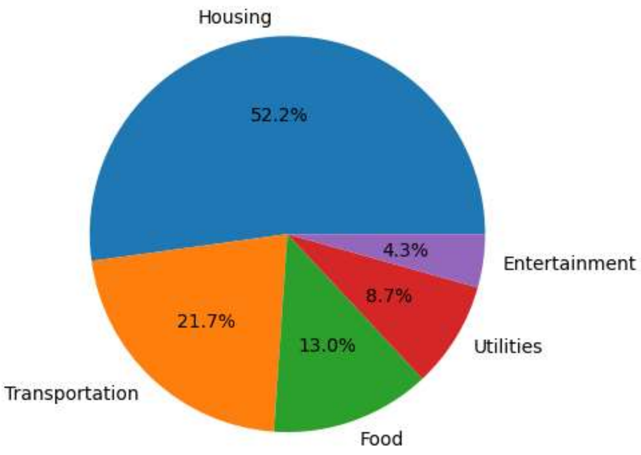


```
sns.barplot(x = labels,y = values)
plt.xticks(rotation=90,fontsize=10)
plt.xlabel("Genre")
plt.ylabel("count")
plt.title("Games per Genre")
plt.show()
```



```
#A bar plot is a graphical representation of data where individual bars represent different categories or groups,
#and the length of each bar corresponds to the quantity or value associated with that category.
```

```
#PIE


# Dummy data for household expenses
categories = ['Housing', 'Transportation', 'Food', 'Utilities', 'Entertainment']
expenses = [1200, 500, 300, 200, 100]
#plt.pie(values,labels)
plt.pie(expenses,labels =categories ,autopct = '%2.1f%%')
plt.show()

#1200/sum([1200, 500, 300, 200, 100]) = 0.52 = 52 %
```



```
#A pie chart is a circular statistical graphic that is divided into slices to illustrate numerical proportions.
#Each slice represents a proportionate part of the whole, and the size of each slice is proportional to the quantity it represents.
#Pie charts are particularly
#useful for displaying the parts of a whole and comparing the contributions of different categories to the total.
```

```
sales_data = df[['NA_Sales', 'EU_Sales', 'JP_Sales', 'Other_Sales']]
```

```
sales_data
```

	NA_Sales	EU_Sales	JP_Sales	Other_Sales
0	4.569217	3.033887	3.439352	1.991671
1	2.076955	1.493442	3.033887	0.394830
2	1.145709	1.762339	1.493442	0.408693
3	2.031986	1.389856	3.228043	0.394830
4	2.792725	2.592054	1.440483	1.493442
...	...	...	...	...
16647	4.409308	3.167419	4.168474	1.087977
16648	3.033887	2.792725	1.596852	1.493442
16649	3.228043	2.004268	1.833151	1.087977
16650	1.087977	0.592445	1.087977	0.394830
16651	1.081046	1.714664	2.004268	0.394830

16652 rows × 4 columns

```
sales_data.sum()
```

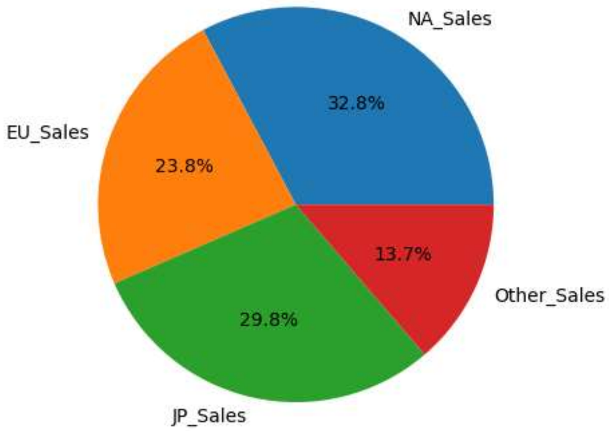
```
NA_Sales       45831.525845
EU_Sales       33251.970702
JP_Sales       41624.625635
Other_Sales    19180.256828
dtype: float64
```

```
sales_data.T # T = Transpose - horizontal to vertical
```

	0	1	2	3	4	5	6	7
NA_Sales	4.569217	2.076955	1.145709	2.031986	2.792725	2.456509	2.031986	4.286650
EU_Sales	3.033887	1.493442	1.762339	1.389856	2.592054	1.435228	1.389856	4.032416
JP_Sales	3.439352	3.033887	1.493442	3.228043	1.440483	3.167419	3.228043	3.652926
Other_Sales	1.991671	0.394830	0.408693	0.394830	1.493442	0.471076	0.394830	2.959779

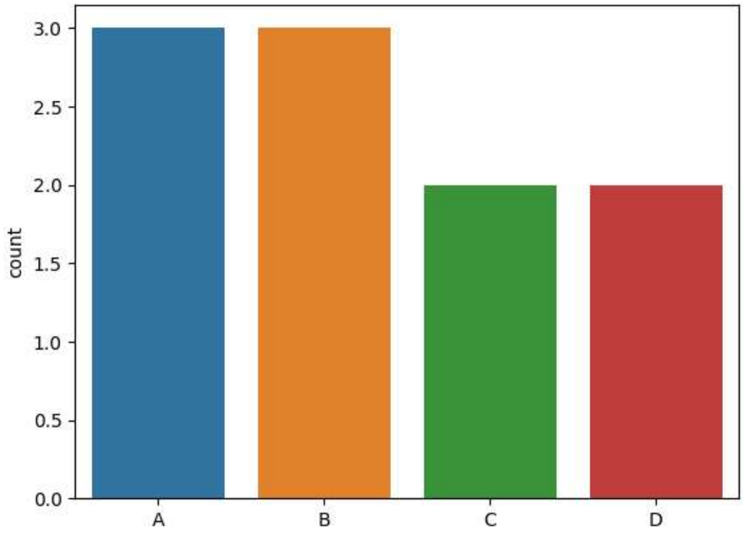
4 rows × 16652 columns

```
plt.pie(sales_data.sum(),labels =sales_data.sum().index ,autopct = '%2.1f%%')
plt.show()
```



```
categories = ['A','B','C','A','B','C','D','A','B','D']
sns.countplot(x=categories) #count how many each "A", "B",'C','D'
```

```
<Axes: ylabel='count'>
```



```
#How can we understand popularity of video games year by year?
```

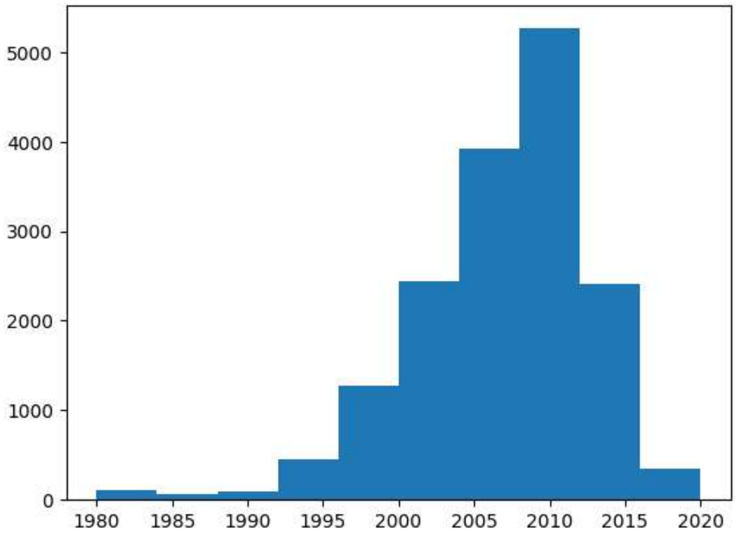
```
df['Year']
```

```
0        1985.0
1        2007.0
2        2012.0
3        2006.0
4        2006.0
          ...
16647    2012.0
16648    2013.0
```

```
16649    2013.0
16650    2008.0
16651    2013.0
Name: Year, Length: 16652, dtype: float64
```

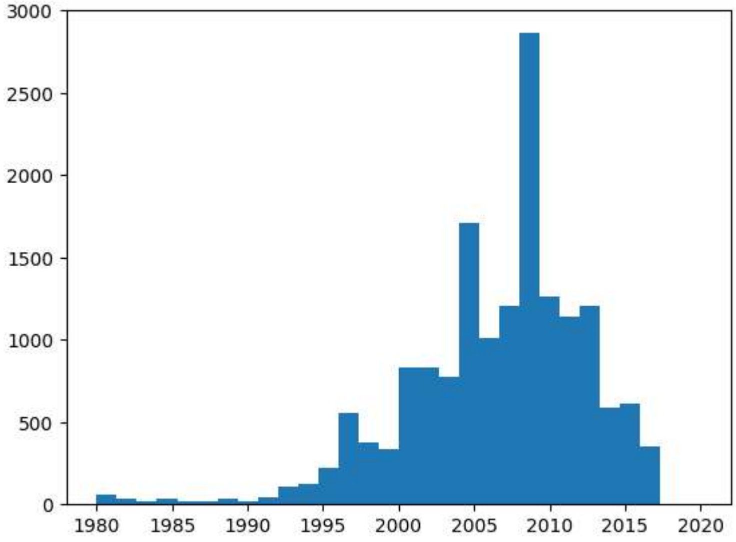
```
plt.hist(df['Year'])
```

```
(array([ 112.,   70.,   92.,  449., 1274., 2440., 3921., 5262., 2406.,
         355.]),
 array([1980., 1984., 1988., 1992., 1996., 2000., 2004., 2008., 2012.,
        2016., 2020.]),
 <BarContainer object of 10 artists>)
```



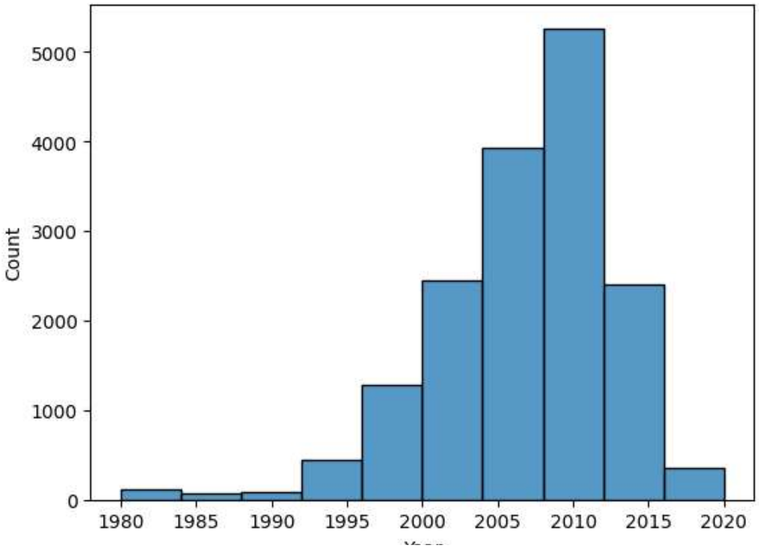
```
plt.hist(df['Year'],bins=30)
```

```
(array([5.700e+01, 3.700e+01, 1.800e+01, 3.100e+01, 2.200e+01, 1.700e+01,
        3.300e+01, 1.700e+01, 4.200e+01, 1.060e+02, 1.220e+02, 2.210e+02,
        5.550e+02, 3.800e+02, 3.390e+02, 8.330e+02, 8.300e+02, 7.770e+02,
        1.708e+03, 1.009e+03, 1.204e+03, 2.861e+03, 1.261e+03, 1.140e+03,
        1.207e+03, 5.840e+02, 6.150e+02, 3.500e+02, 1.000e+00, 4.000e+00]),
 array([1980.        , 1981.33333333, 1982.66666667, 1984.        ,
        1985.33333333, 1986.66666667, 1988.        , 1989.33333333,
        1990.66666667, 1992.        , 1993.33333333, 1994.66666667,
        1996.        , 1997.33333333, 1998.66666667, 2000.        ,
        2001.33333333, 2002.66666667, 2004.        , 2005.33333333,
        2006.66666667, 2008.        , 2009.33333333, 2010.66666667,
        2012.        , 2013.33333333, 2014.66666667, 2016.        ,
        2017.33333333, 2018.66666667, 2020.        ]),
 <BarContainer object of 30 artists>)
```



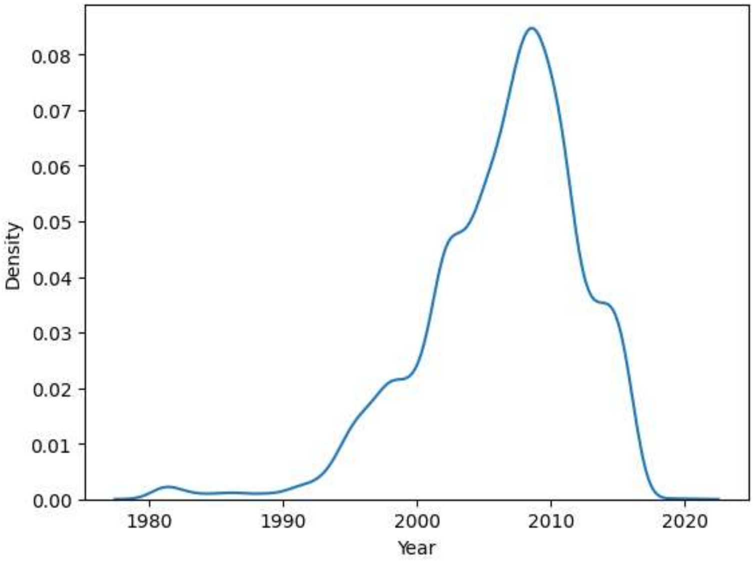
```
sns.histplot(df['Year'],bins=10)
```

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



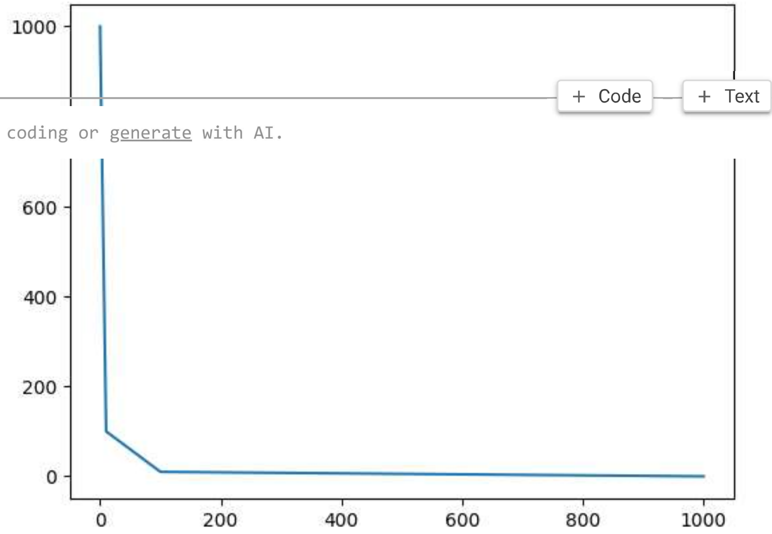
```
#Kernel Density Estimate (KDE) Plot A KDE plot, similar to histrogram, is a method for visualizing the distributions

#But instead of bars, KDE represents data using a continuous probability density curve


sns.kdeplot(data['Year'])
```

```
<Axes: xlabel='Year', ylabel='Density'>
```



```
x = [0,10,100,1000]
y = [1000,100,10,0]
plt.plot(x,y)
```

```
[<matplotlib.lines.Line2D at 0x7e8e733c44c0>]
```



+ Code + Text

Start coding or generate with AI.