In [2]:
```
import pandas as pd
import pickle
import warnings
warnings.filterwarnings("ignore")
```

In [3]:
```
a=pd.read_csv("C:\\Users\\reshma_koduri\\OneDrive\\Documents\\Video Games Sales.csv"
a
```

Out[3]:

	index	Rank	Game Title	Platform	Year	Genre	Publisher	North America	Europe	Japan
0	0	1	Wii Sports	Wii	2006.0	Sports	Nintendo	40.43	28.39	3.77
1	1	2	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08	3.58	6.81
2	2	3	Mario Kart Wii	Wii	2008.0	Racing	Nintendo	14.50	12.22	3.63
3	3	4	Wii Sports Resort	Wii	2009.0	Sports	Nintendo	14.82	10.51	3.18
4	4	5	Tetris	GB	1989.0	Puzzle	Nintendo	23.20	2.26	4.22
...	...	...	...	...	...	...	...	...	...	...
1902	1902	1903	Lizzie McGuire 2: Lizzie Diaries	GBA	2004.0	Action	Disney Interactive Studios	0.60	0.22	0.00
1903	1903	1904	Xenoblade Chronicles	Wii	2010.0	Role-Playing	Nintendo	0.39	0.22	0.16
1904	1904	1905	SingStar Abba	PS3	2008.0	Misc	Sony Computer Entertainment	0.25	0.44	0.00
1905	1905	1906	FIFA Soccer World Championship	PS2	2000.0	Sports	Electronic Arts	0.27	0.21	0.28
1906	1906	1907	WWE SmackDown vs. Raw 2011	X360	2010.0	Fighting	THQ	0.42	0.32	0.00

1907 rows × 13 columns

In [4]:
```
a.head(10)
```

Out[4]:

	index	Rank	Game Title	Platform	Year	Genre	Publisher	North America	Europe	Japan	Rest of World
0	0	1	Wii Sports	Wii	2006.0	Sports	Nintendo	40.43	28.39	3.77	8.54
1	1	2	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08	3.58	6.81	0.77
2	2	3	Mario Kart Wii	Wii	2008.0	Racing	Nintendo	14.50	12.22	3.63	3.21

	index	Rank	Game Title	Platform	Year	Genre	Publisher	North America	Europe	Japan	Rest of World
3	3	4	Wii Sports Resort	Wii	2009.0	Sports	Nintendo	14.82	10.51	3.18	3.01
4	4	5	Tetris	GB	1989.0	Puzzle	Nintendo	23.20	2.26	4.22	0.58
5	5	6	New Super Mario Bros.	DS	2006.0	Platform	Nintendo	10.85	8.87	6.48	2.88
6	6	7	Wii Play	Wii	2006.0	Misc	Nintendo	13.83	9.11	2.93	2.84
7	7	8	Duck Hunt	NES	1984.0	Shooter	Nintendo	26.93	0.63	0.28	0.47
8	8	9	New Super Mario Bros. Wii	Wii	2009.0	Platform	Nintendo	13.35	6.48	4.66	2.25
9	9	10	Nintendogs	DS	2005.0	Simulation	Nintendo	9.02	10.81	1.93	2.73

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

Out[5]:

	index	Rank	Game Title	Platform	Year	Genre	Publisher	North America	Europe	Japa
1897	1897	1898	Ace Combat 3: Electrosphere	PS	1999.0	Simulation	Sony Computer Entertainment	0.22	0.15	0.4
1898	1898	1899	Dynasty Warriors 2	PS2	2000.0	Action	THQ	0.24	0.19	0.3
1899	1899	1900	Madden NFL 07	PSP	NaN	Sports	Unknown	0.77	0.03	0.0
1900	1900	1901	Army of Two: The 40th Day	X360	2010.0	Shooter	Electronic Arts	0.52	0.22	0.0
1901	1901	1902	Medal of Honor: Warfighter	X360	2012.0	Shooter	Electronic Arts	0.42	0.32	0.0
1902	1902	1903	Lizzie McGuire 2: Lizzie Diaries	GBA	2004.0	Action	Disney Interactive Studios	0.60	0.22	0.0
1903	1903	1904	Xenoblade Chronicles	Wii	2010.0	Role-Playing	Nintendo	0.39	0.22	0.1
1904	1904	1905	SingStar Abba	PS3	2008.0	Misc	Sony Computer Entertainment	0.25	0.44	0.0
1905	1905	1906	FIFA Soccer World Championship	PS2	2000.0	Sports	Electronic Arts	0.27	0.21	0.2
1906	1906	1907	WWE SmackDown vs. Raw 2011	X360	2010.0	Fighting	THQ	0.42	0.32	0.0

In [6]:
```
a.describe()
```

Out[6]:

	index	Rank	Year	North America	Europe	Japan	Rest of World	
count	1907.0000	1907.0000	1878.000000	1907.000000	1907.000000	1907.000000	1907.000000	1907.0
mean	953.0000	954.0000	2003.766773	1.258789	0.706675	0.317493	0.206471	2.4
std	550.6478	550.6478	5.895369	1.956560	1.148904	0.724945	0.343093	3.5
min	0.0000	1.0000	1983.000000	0.000000	0.000000	0.000000	0.000000	0.8
25%	476.5000	477.5000	2000.000000	0.510000	0.230000	0.000000	0.060000	1.1
50%	953.0000	954.0000	2005.000000	0.810000	0.440000	0.020000	0.130000	1.5
75%	1429.5000	1430.5000	2008.000000	1.375000	0.810000	0.300000	0.220000	2.5
max	1906.0000	1907.0000	2012.000000	40.430000	28.390000	7.200000	8.540000	81.1

In [7]:
```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1907 entries, 0 to 1906
Data columns (total 13 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   index          1907 non-null   int64  
 1   Rank           1907 non-null   int64  
 2   Game Title     1907 non-null   object 
 3   Platform       1907 non-null   object 
 4   Year           1878 non-null   float64
 5   Genre          1907 non-null   object 
 6   Publisher      1905 non-null   object 
 7   North America  1907 non-null   float64
 8   Europe         1907 non-null   float64
 9   Japan          1907 non-null   float64
 10  Rest of World  1907 non-null   float64
 11  Global         1907 non-null   float64
 12  Review         1907 non-null   float64
dtypes: float64(7), int64(2), object(4)
memory usage: 193.8+ KB
```

In [8]:
```
a['Genre'].unique()
```

Out[8]:
```
array(['Sports', 'Platform', 'Racing', 'Puzzle', 'Misc', 'Shooter',
       'Simulation', 'Role-Playing', 'Action', 'Fighting', 'Adventure',
       'Strategy'], dtype=object)
```

In [9]:
```
a.isna().sum()
```

Out[9]:
```
index             0
Rank              0
Game Title        0
Platform          0
Year             29
Genre             0
Publisher         2
North America     0
```

```
Europe            0
Japan             0
Rest of World     0
Global            0
Review            0
dtype: int64
```

In [10]:
```
b=a.drop(['Publisher','Game Title','Platform'],axis=1)
b
```

Out[10]:

	index	Rank	Year	Genre	North America	Europe	Japan	Rest of World	Global	Review
0	0	1	2006.0	Sports	40.43	28.39	3.77	8.54	81.12	76.28
1	1	2	1985.0	Platform	29.08	3.58	6.81	0.77	40.24	91.00
2	2	3	2008.0	Racing	14.50	12.22	3.63	3.21	33.55	82.07
3	3	4	2009.0	Sports	14.82	10.51	3.18	3.01	31.52	82.65
4	4	5	1989.0	Puzzle	23.20	2.26	4.22	0.58	30.26	88.00
...	...	...	...	...	...	...	...	...	...	...
1902	1902	1903	2004.0	Action	0.60	0.22	0.00	0.01	0.83	55.00
1903	1903	1904	2010.0	Role-Playing	0.39	0.22	0.16	0.07	0.83	91.74
1904	1904	1905	2008.0	Misc	0.25	0.44	0.00	0.14	0.83	73.00
1905	1905	1906	2000.0	Sports	0.27	0.21	0.28	0.07	0.83	73.00
1906	1906	1907	2010.0	Fighting	0.42	0.32	0.00	0.09	0.83	82.00

1907 rows × 10 columns

In [11]:
```
c=pd.get_dummies(b,dtype=int)
c
```

Out[11]:

	index	Rank	Year	North America	Europe	Japan	Rest of World	Global	Review	Genre_Action	...	Genr
0	0	1	2006.0	40.43	28.39	3.77	8.54	81.12	76.28	0	...	
1	1	2	1985.0	29.08	3.58	6.81	0.77	40.24	91.00	0	...	
2	2	3	2008.0	14.50	12.22	3.63	3.21	33.55	82.07	0	...	
3	3	4	2009.0	14.82	10.51	3.18	3.01	31.52	82.65	0	...	
4	4	5	1989.0	23.20	2.26	4.22	0.58	30.26	88.00	0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
1902	1902	1903	2004.0	0.60	0.22	0.00	0.01	0.83	55.00	1	...	
1903	1903	1904	2010.0	0.39	0.22	0.16	0.07	0.83	91.74	0	...	
1904	1904	1905	2008.0	0.25	0.44	0.00	0.14	0.83	73.00	0	...	
1905	1905	1906	2000.0	0.27	0.21	0.28	0.07	0.83	73.00	0	...	
1906	1906	1907	2010.0	0.42	0.32	0.00	0.09	0.83	82.00	0	...	

1907 rows × 21 columns

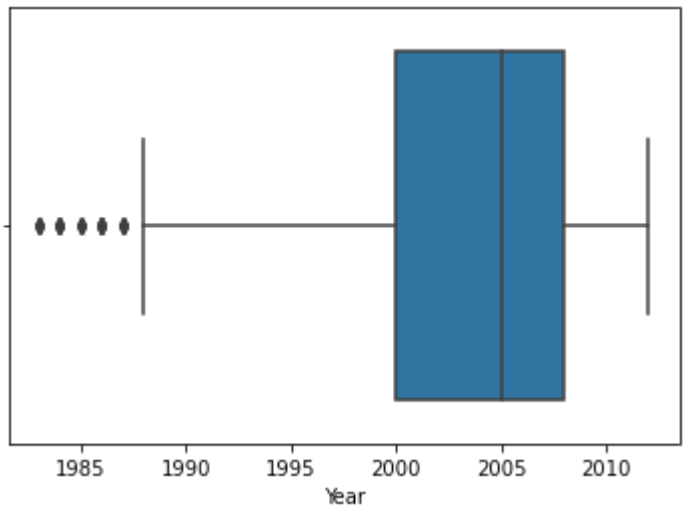
In [12]:
```
colnames=list(c)
colnames
```

Out[12]:
```
['index',
 'Rank',
 'Year',
 'North America',
 'Europe',
 'Japan',
 'Rest of World',
 'Global',
 'Review',
 'Genre_Action',
 'Genre_Adventure',
 'Genre_Fighting',
 'Genre_Misc',
 'Genre_Platform',
 'Genre_Puzzle',
 'Genre_Racing',
 'Genre_Role-Playing',
 'Genre_Shooter',
 'Genre_Simulation',
 'Genre_Sports',
 'Genre_Strategy']
```

In [13]:
```
from sklearn.impute import KNNImputer
imputer=KNNImputer(n_neighbors=3)
data_filled=imputer.fit_transform(c)
c=pd.DataFrame(data=data_filled,columns=colnames)
```

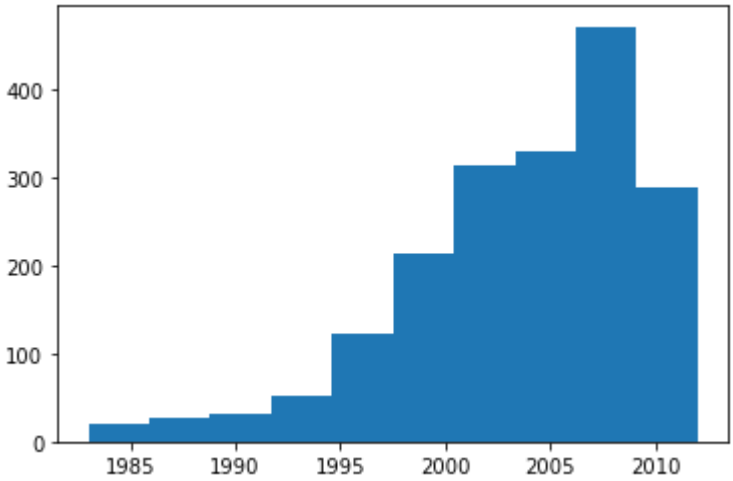
In [14]:
```
import seaborn as sb
import matplotlib.pyplot as plt
sb.boxplot(a.Year)
```

Out[14]:
```
<AxesSubplot:xlabel='Year'>
```

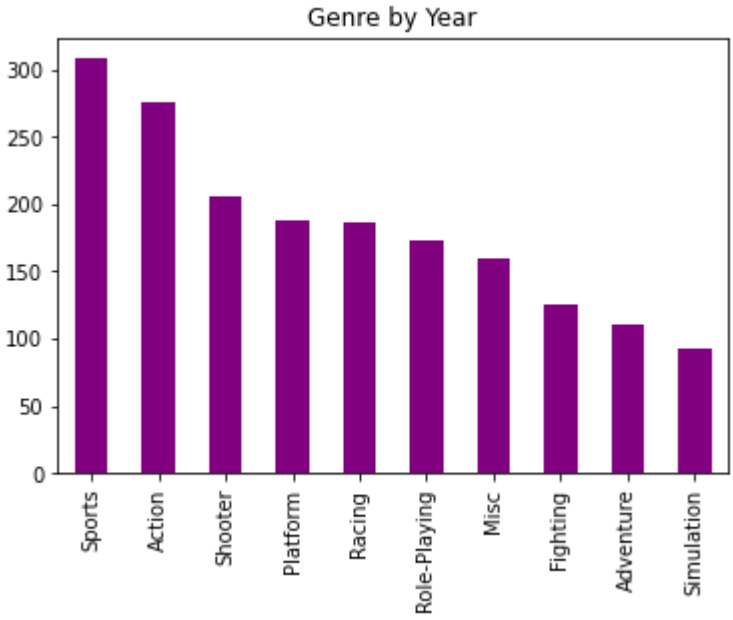


In [15]:
```
import seaborn as sb
import matplotlib.pyplot as plt
plt.hist(a['Year'])
```

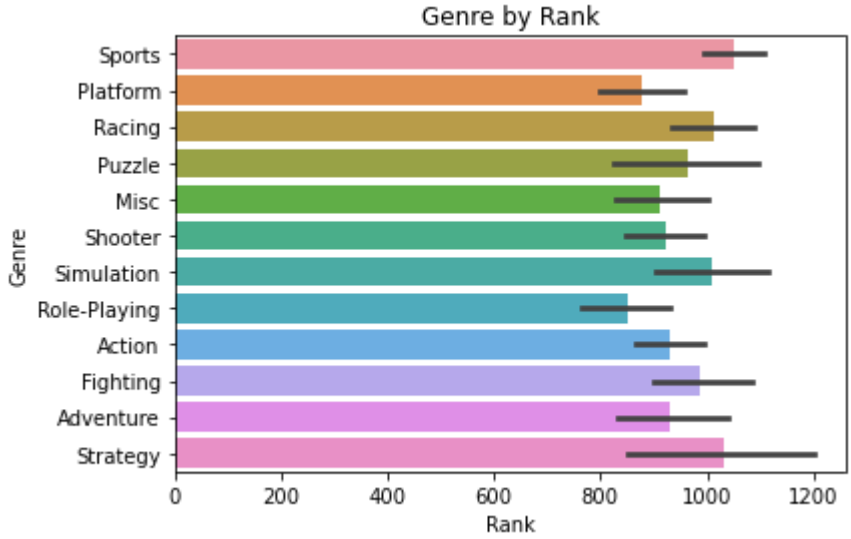
Out[15]:
```
(array([ 21.,  28.,  32.,  53., 123., 214., 315., 330., 472., 290.]),
 array([1983. , 1985.9, 1988.8, 1991.7, 1994.6, 1997.5, 2000.4, 2003.3,
        2006.2, 2009.1, 2012. ]),
 <BarContainer object of 10 artists>)
```



In [16]:
```
import seaborn as sns
import matplotlib.pyplot as plt
a['Genre'].value_counts().head(10).plot(kind='bar',color='purple')
plt.title('Genre by Year');
```

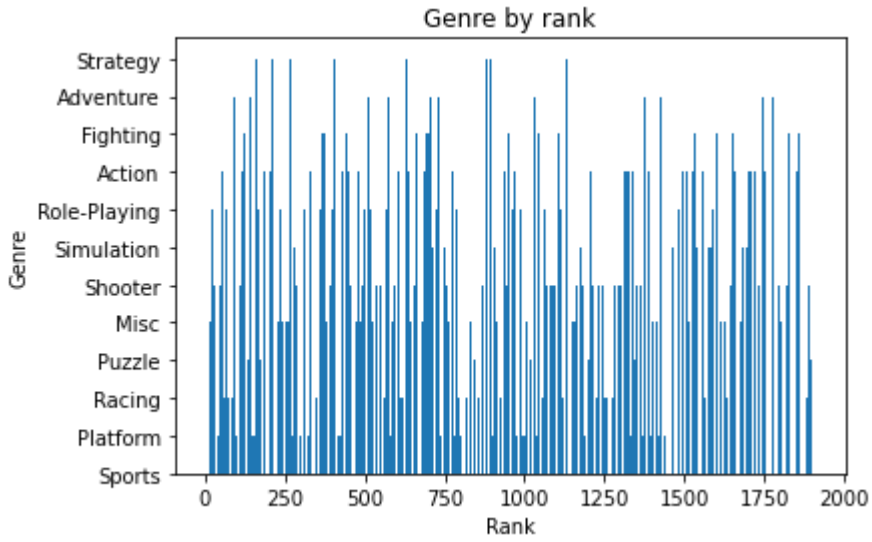


In [17]:
```
sb.barplot(x='Rank', y='Genre', data=a)
plt.xlabel('Rank')
plt.ylabel('Genre')
plt.title('Genre by Rank')
plt.show()
```



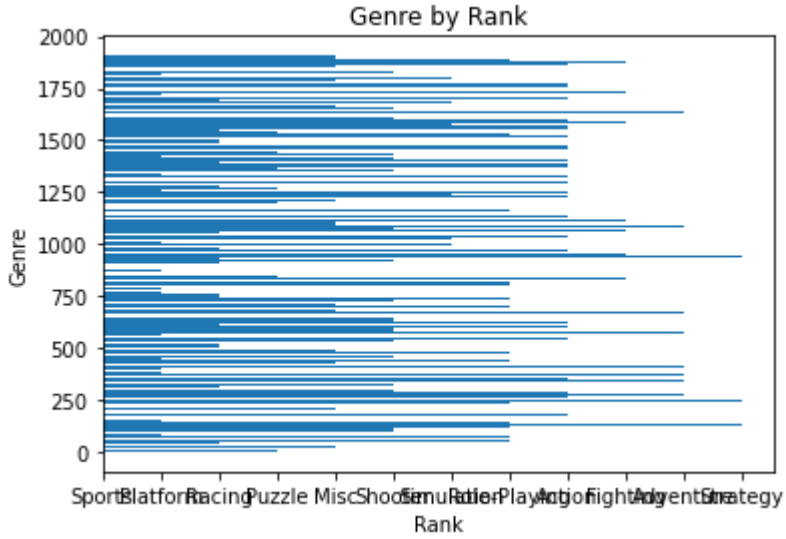
In [18]:

```
plt.bar(a['Rank'],a['Genre'])#vertical barplot
plt.xlabel('Rank')
plt.ylabel('Genre')
plt.title('Genre by rank')
plt.show()
```



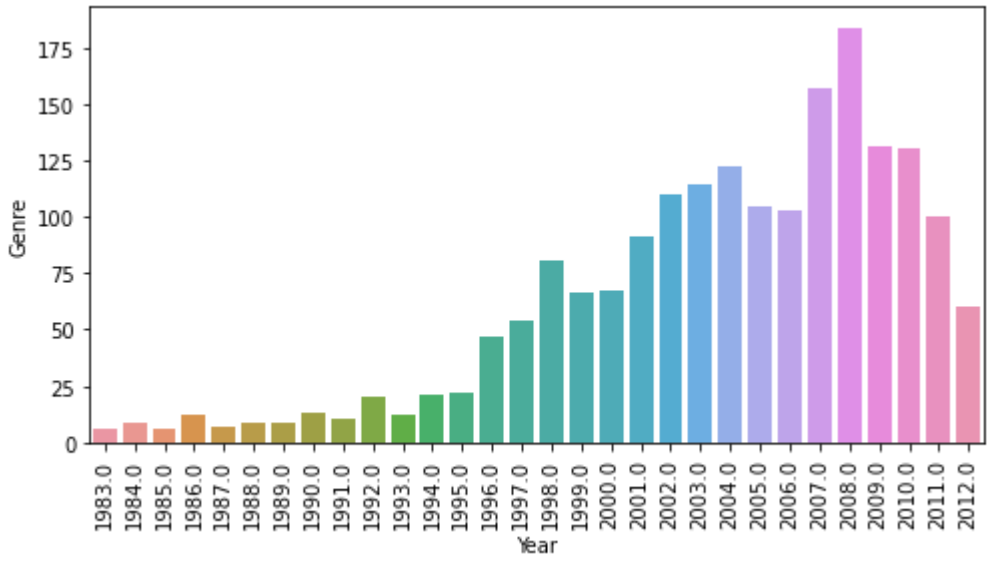
In [19]:

```
plt.barh(a['Rank'],a['Genre'])#horizontal barplot
plt.xlabel('Rank')
plt.ylabel('Genre')
plt.title('Genre by Rank')
plt.show()
```



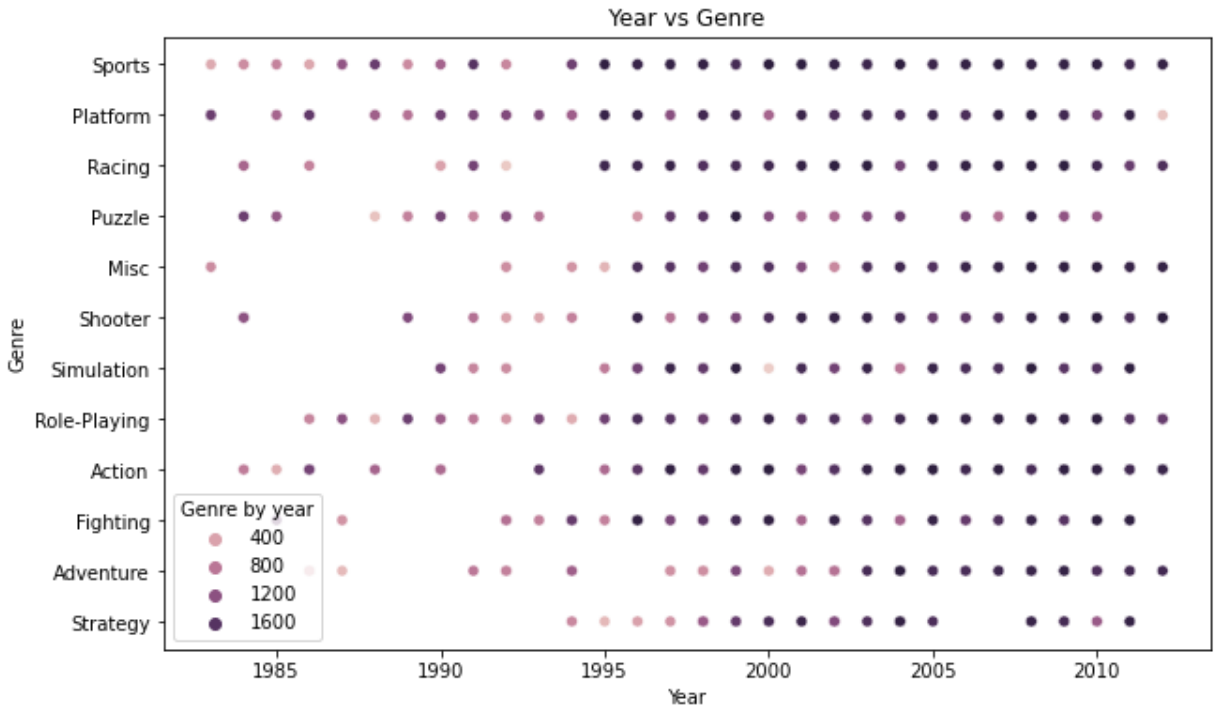
In [20]:

```
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(8,4))
sns.countplot(data=a,x='Year')
plt.xticks(rotation=90)
plt.xlabel('Year')
plt.ylabel('Genre')
plt.show()
```



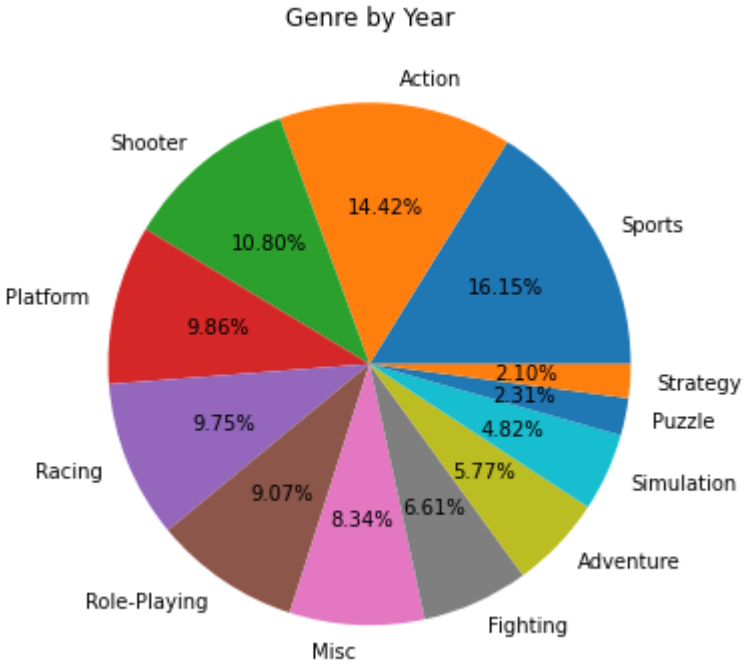
In [21]:

```
plt.figure(figsize=(10, 6))
sns.scatterplot(data=a, x='Year', y='Genre', hue='Rank')
plt.title('Year vs Genre')
plt.xlabel('Year')
plt.ylabel('Genre')
plt.legend(title='Genre by year')
plt.show()
```



In [22]:

```
plt.figure(figsize=(8, 6))
a['Genre'].value_counts().plot.pie(autopct='%2.2f%%')
plt.title('Genre by Year')
plt.ylabel('')
plt.show()
```



In [23]:

```
cor=a.corr()
cor
```

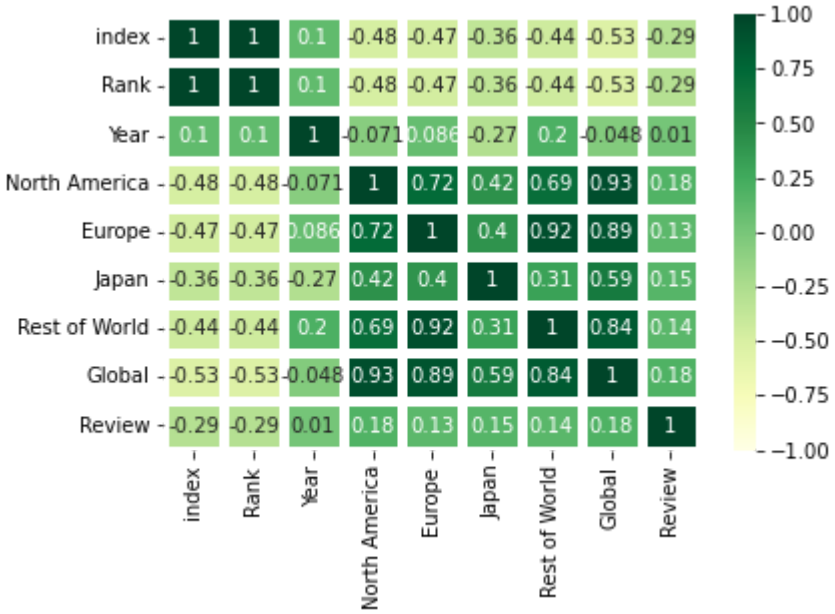
Out[23]:

	index	Rank	Year	North America	Europe	Japan	Rest of World	Global	Rev
index	1.000000	1.000000	0.101943	-0.480582	-0.466451	-0.358849	-0.436750	-0.529373	-0.292
Rank	1.000000	1.000000	0.101943	-0.480582	-0.466451	-0.358849	-0.436750	-0.529373	-0.292

	index	Rank	Year	North America	Europe	Japan	Rest of World	Global	Rev
Year	0.101943	0.101943	1.000000	-0.071347	0.085549	-0.274221	0.201768	-0.047886	0.010
North America	-0.480582	-0.480582	-0.071347	1.000000	0.720766	0.416743	0.693662	0.933073	0.175
Europe	-0.466451	-0.466451	0.085549	0.720766	1.000000	0.402289	0.922623	0.888902	0.129
Japan	-0.358849	-0.358849	-0.274221	0.416743	0.402289	1.000000	0.308785	0.591751	0.148
Rest of World	-0.436750	-0.436750	0.201768	0.693662	0.922623	0.308785	1.000000	0.837469	0.138
Global	-0.529373	-0.529373	-0.047886	0.933073	0.888902	0.591751	0.837469	1.000000	0.181
Review	-0.292892	-0.292892	0.010387	0.175684	0.129741	0.148584	0.138467	0.181881	1.000

In [24]:
```
sb.heatmap(cor,vmax=1,vmin=-1,annot=True,linewidth=5,cmap='YlGn')
```

Out[24]:
```
<AxesSubplot:>
```



In [27]:
```
y=c['Global']
y
```

Out[27]:
```
0       81.12
1       40.24
2       33.55
3       31.52
4       30.26
        ...
1902     0.83
1903     0.83
1904     0.83
1905     0.83
1906     0.83
Name: Global, Length: 1907, dtype: float64
```

In [28]:
```
x=c.drop(['Global'],axis=1)
```

```
x
```

Out[28]:

	index	Rank	Year	North America	Europe	Japan	Rest of World	Review	Genre_Action	Genre_Adventu
0	0.0	1.0	2006.0	40.43	28.39	3.77	8.54	76.28	0.0	0
1	1.0	2.0	1985.0	29.08	3.58	6.81	0.77	91.00	0.0	0
2	2.0	3.0	2008.0	14.50	12.22	3.63	3.21	82.07	0.0	0
3	3.0	4.0	2009.0	14.82	10.51	3.18	3.01	82.65	0.0	0
4	4.0	5.0	1989.0	23.20	2.26	4.22	0.58	88.00	0.0	0
...	...	...	...	...	...	...	...	...	...	
1902	1902.0	1903.0	2004.0	0.60	0.22	0.00	0.01	55.00	1.0	0
1903	1903.0	1904.0	2010.0	0.39	0.22	0.16	0.07	91.74	0.0	0
1904	1904.0	1905.0	2008.0	0.25	0.44	0.00	0.14	73.00	0.0	0
1905	1905.0	1906.0	2000.0	0.27	0.21	0.28	0.07	73.00	0.0	0
1906	1906.0	1907.0	2010.0	0.42	0.32	0.00	0.09	82.00	0.0	0

1907 rows × 20 columns

In [29]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.33,random_state=42)
```

In [30]:

```
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
reg.fit(x_train, y_train)
```

Out[30]:

```
LinearRegression()
```

In [31]:

```
ypred=reg.predict(x_test)
ypred
```

Out[31]:

```
array([ 1.24964738,  5.93843642,  0.86053727,  0.95001128,  2.79115906,
        1.15033839,  7.86980917,  2.37959972,  2.09948858,  1.0297089 ,
        0.880642  ,  1.53060938,  1.36942647,  5.45104981,  3.02062395,
        0.90894156,  1.19950693,  1.87941947,  2.98958906,  1.17936967,
        3.63154741,  1.64962594,  1.24909515,  1.41928645,  4.2096421 ,
        1.76921856,  5.33959613,  1.0797274 ,  0.97051   ,  0.88946745,
        2.02955241,  1.66925718,  1.03958029,  1.01963279,  4.03983666,
        0.95824203,  1.50929158,  1.42026723,  2.14909992,  0.91935363,
        2.17896445,  3.31964254,  3.01956982,  2.29845612,  1.24074351,
        1.32954407,  2.78895936,  0.93898048,  6.3094137 ,  0.94867452,
        1.0599472 ,  0.87962955,  1.10959617,  3.96974059,  3.52948073,
        0.84048153,  0.92070291,  2.94152792,  1.72963146,  1.1086103 ,
        1.35989466,  1.04973331,  1.68012946,  0.85052628,  3.53093245,
        1.26967434,  1.5105184 ,  1.12990357,  1.97051266,  1.07075932,
        1.07928739,  0.84884151,  0.84039095,  8.0905761 ,  1.39982608,
        1.24984027,  1.0598107 ,  1.56975896, 13.09954872,  2.11948896,
        1.2705795 ,  1.91925112,  0.86064709,  1.04961701,  2.21928359,
        1.17898056,  3.25932577,  1.02954345,  1.03051816,  2.49975017,
        0.87981018,  2.80050573,  0.90038398,  1.21055757,  2.61928528,
```

```
 2.69899849,  1.51919586,  1.42002186,  0.96118079,  1.94956054,
 3.31922294,  0.99035528,  0.95046122,  4.44975586,  1.56020138,
 2.1402406 ,  5.64080837,  1.10895557,  0.99971016,  1.81137974,
 6.03045271,  3.4494393 ,  1.09887541,  1.13869593,  0.8584445 ,
 1.82936522,  3.1485521 ,  2.748891  ,  2.27019206,  1.32990079,
 1.67972073,  1.15961906,  0.85979773,  0.99957351,  1.18911883,
 2.10924837,  1.49926928,  1.15137961,  1.76061109,  1.91966969,
 2.11917434,  1.17981042,  2.41027062,  1.28876037,  3.82932823,
 0.8488928 ,  0.82114369,  2.52956123,  2.38083679,  3.92965726,
 0.98949824,  1.01026741,  1.54982048,  3.79027497,  0.84038373,
 1.23945819,  1.41068974,  1.43028485,  1.4995751 ,  2.40943093,
 1.25957295,  1.3800122 ,  1.12024172,  1.93066847, 15.23071614,
 3.44011887,  8.23079934,  2.77146593,  3.4892231 ,  0.92982528,
 1.91031559,  2.74976932,  0.91939603,  1.56016662,  4.52011834,
 0.83938983,  1.27945671,  0.85954221,  1.12968107,  1.15098661,
 1.66034034,  2.30915807,  2.79925297,  1.32981503,  1.17969213,
 2.13985451,  1.08878531,  5.78942874,  1.0695536 ,  1.80987489,
 2.38070786,  1.65068925,  1.03922447,  2.15005643,  1.90035381,
 1.26995533,  2.78861053,  1.08970767,  4.84929705,  1.00962733,
 1.25948238,  0.86943248,  2.50874098,  1.77909259,  1.51927696,
 1.46966772, 11.90186404,  1.64045419,  4.36954587,  3.63950837,
 5.49084701,  1.03933608,  4.98008005,  1.08949366,  2.0393701 ,
 1.28970259,  1.23974553,  9.87030324,  1.66065585,  3.99929773,
 1.19034729,  1.7797659 ,  1.38968825,  0.99916672,  1.01919789,
 3.75888055,  3.8686404 ,  3.08902507,  4.11067906,  3.02039711,
 0.99045841,  3.08061753,  0.90984374,  0.83940235,  4.11965044,
 3.50060242,  1.02920714,  7.18146375,  8.81827195,  1.9599634 ,
 3.11866821,  1.48058263,  1.29943984,  3.3596479 ,  7.60023328,
 1.26974541,  1.00007252,  4.19928671,  7.88976012,  4.98886065,
 1.58965269,  2.09958944,  0.87996925,  0.91918971,  2.11038342,
 2.47920346,  1.44944104,  0.89962329,  1.50951048,  2.35010488,
 1.13995177, 10.48979694,  1.20953166,  1.07065461,  0.90941262,
 1.24036405,  1.33970286,  7.46063527,  0.86937814,  1.17055291,
 1.09937002,  0.82947668,  1.47962034,  1.94051959,  1.06993133,
12.99887999,  1.15973915,  1.53967387,  1.6794728 ,  1.23058661,
 2.63927527,  2.72905589,  1.07963635,  1.05915033,  0.82961282,
 1.06959721, 10.55119724,  1.47039347,  4.90921635,  2.20935216,
 3.0887711 ,  0.86956643,  2.29934849,  1.17932484,  1.14067111,
 0.88920489,  2.20976611,  2.36996933,  0.97927756,  1.24973658,
 4.5293779 ,  1.53883631,  0.98941133,  3.44075952,  0.85955256,
 2.5290605 ,  0.84965568,  1.14865844,  1.19852721,  1.18036072,
 0.93922046,  1.08027415,  1.04126888,  5.53942296,  2.81992576,
 2.50977461,  1.4899699 ,  1.29991307,  1.21962216,  1.56955153,
 0.8714657 ,  4.45882864,  2.40987204,  1.22882832,  0.97950681,
 1.26949865,  3.69850321,  1.22070323,  8.03018847,  2.55864464,
 1.48971114,  2.36976826,  2.68987463,  1.15987939,  2.93971957,
 0.94893242,  1.72940436,  3.03089556,  1.0195571 ,  1.51855501,
 3.17131173,  0.96984666,  1.06041273,  0.87864972,  1.31956702,
 2.25064921,  2.58061298,  2.07990819,  0.89034192,  1.44050763,
 3.71873211,  1.24981428,  3.90972615,  0.91960801,  2.07939501,
 1.97959223,  1.43974381,  1.23001515,  1.56030959,  2.35866008,
 1.99120767,  1.20945429,  1.93936258,  7.85940671,  2.11959147,
 3.71851016,  9.78005911,  1.22052984,  0.9108599 ,  2.15040687,
 3.2407676 ,  1.20898449,  0.92045826,  1.26924881,  2.85908465,
 1.00073326,  1.70964474,  2.31948206,  2.1414365 ,  0.93964754,
 1.07969382,  1.62966991,  0.83919007,  2.48028069,  1.16953082,
 4.48015728,  2.27975304,  1.23970377,  0.85889194,  1.85960114,
 2.16986663,  1.70929057,  5.2691056 ,  1.1998695 ,  1.34074344,
 0.89065608,  3.17996435,  1.13094747,  2.58060135,  1.2211871 ,
 1.00938126,  2.87975528,  3.6499225 ,  4.54917996,  1.16036843,
 3.88946531,  2.51035265,  2.34071107,  1.84914239,  2.75933726,
 1.21034498,  0.93957406,  2.09878204,  1.47950323,  1.40948587,
 1.10959185,  1.13990202,  3.82966625,  0.96065532,  1.14945136,
 1.92979664,  0.9605386 ,  2.95954751,  2.71850375,  1.45001925,
```

```
0.91951739,  1.27953316,  1.21036016,  2.83043257,  0.93861982,
1.00927849,  1.66958456,  0.84120524,  1.02940525,  1.0098816 ,
4.60950497,  1.26904825,  4.87880232,  3.54041581,  0.86966212,
2.35947866,  3.4995341 ,  0.89067201,  1.58943754,  1.00959854,
1.11957078,  0.85981222,  1.49879858,  0.88964478,  1.71963354,
2.78932727,  3.0006822 ,  1.07928517,  1.69971262,  2.13881305,
1.53033645,  1.34901008,  4.75827465,  2.32971899,  1.53930733,
1.54836254,  1.91957073,  3.20082826,  1.68046729,  1.51900725,
1.33978677,  1.49992265,  1.03857011,  1.88866185,  1.09956054,
4.41936736, 20.62159841,  3.68980991,  1.18997946,  1.80958769,
1.58926667,  1.4205504 ,  1.61827781,  2.17943669,  1.72990704,
1.34075862,  1.01933633,  2.56865682,  1.27885385,  4.3991112 ,
1.14982422,  0.98962446,  1.12133056,  1.9604832 ,  1.34963508,
1.57077656,  2.52950209,  1.45030093,  1.23929738,  1.79041994,
1.25967791,  1.63960559,  2.16921278,  1.45088274,  3.4287552 ,
9.36035877,  2.11956961,  1.54972421,  1.2493029 ,  1.91928462,
1.49068175, 10.75098202,  6.29868798,  3.62946726,  1.17919654,
3.08055703,  6.2787367 ,  1.13952814,  1.7398461 ,  1.63966969,
6.04940385,  1.29068371,  2.44930921,  0.82963519,  1.09961071,
1.74940648,  1.03973043,  1.03910486,  1.41039773,  1.23949747,
1.62968512,  1.37953762,  7.70984516,  0.85025851,  1.03027882,
1.33896099,  1.02129929,  2.52024382,  1.29912106,  2.43935114,
2.1086582 ,  0.87841561,  0.83954957,  1.97938435,  1.78074544,
1.13972836,  1.82046458,  2.84949325,  2.04903973,  1.23896529,
3.41992725,  4.06983323,  0.98039819,  1.80075839,  1.07958271,
1.82989478,  1.68952994,  5.15070346,  2.24886759,  2.03957403,
1.81951127,  1.84884281,  1.16914111,  1.71958675,  1.36012742,
1.21964761,  1.35058918,  1.01952597,  3.58111026,  1.04069303,
1.63970083,  1.16968989,  1.02984419,  1.05004675,  1.50963261,
1.28980446,  1.06958718,  1.5492622 ,  2.17927425,  1.06896623,
0.86966934,  0.83964982,  0.99933632,  1.15882938,  1.5799968 ,
2.48008916,  0.94127482,  3.62915711,  0.90959786,  3.05828918,
1.04063271,  3.99006076,  2.3088128 ,  0.88949847,  0.96999583,
4.22989113,  1.17980477,  1.07011785,  8.87908209,  1.04839926,
1.05984178,  3.03231923,  1.49076743,  1.90910883,  1.39971262,
1.04046831,  1.28923917,  1.19953257,  1.62915009,  1.85086778,
5.74124599,  1.23973688,  1.11908511,  2.40041495,  2.41950144,
7.01056465,  1.21956942,  1.57073477,  2.96008125,  1.00005974,
6.91921617,  1.00939291,  3.28952252,  1.47037915,  1.95999378,
1.25944939,  0.86014629,  0.89963879,  2.62932798,  2.48066344,
1.11960403,  3.57824533,  1.24964745,  0.83880352,  1.18970408,
1.34850359,  0.87025262,  0.96950175,  0.99015868,  1.87950565,
0.90015441,  0.8296633 ,  1.42977539,  0.91964214,  3.2997315 ])
```

In [32]:

```
from sklearn.metrics import r2_score
r2_score(y_test,ypred)
```

Out[32]:

```
0.9999918154554228
```

In [33]:

```
results=pd.DataFrame(columns=['Price','Predicted'])
results['Price']=y_test
results["Predicted"]=ypred
results=results.reset_index()
results['Id']=results.index
results.head(5)
```

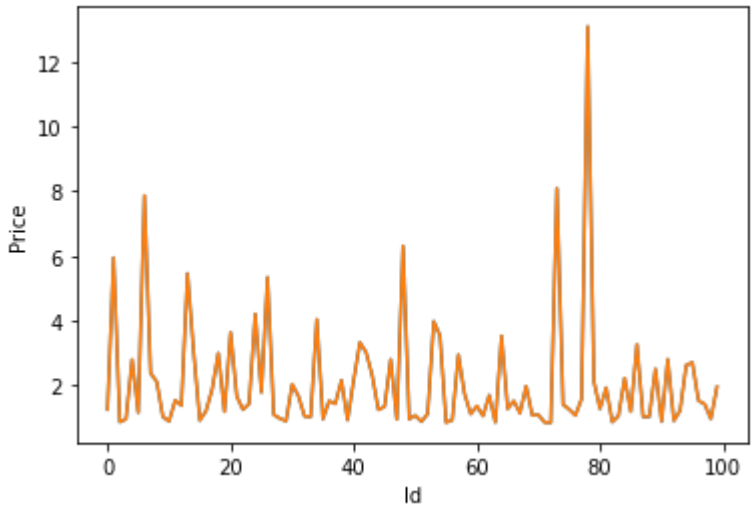
Out[33]:

	index	Price	Predicted	Id
0	1226	1.25	1.249647	0
1	111	5.95	5.938436	1

	index	Price	Predicted	Id
2	1840	0.86	0.860537	2
3	1676	0.95	0.950011	3
4	415	2.79	2.791159	4

In [34]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.lineplot(x='Id',y='Price',data=results.head(100))
sns.lineplot(x='Id',y='Predicted',data=results.head(100))
plt.plot()
```

Out[34]:
```
[]
```



In [ ]:
```
```