In [2]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
import joblib
import datetime
```

In [7]:
```
game=pd.read_csv("/game sales.csv")
```

In [8]:
```
game
```

Out[8]:

	Rank	Name	Platform	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Oth
0	1.0	Wii Sports	Wii	Sports	Nintendo	41.49	29.02	3.77	
1	NaN	Super Mario Bros.	NES	Platform	NaN	NaN	NaN	NaN	
2	NaN	Mario Kart Wii	Wii	Racing	Nintendo	15.85	12.88	3.79	
3	NaN	Wii Sports Resort	NaN	Sports	Nintendo	15.75	11.01	3.28	
4	NaN	Pokemon Red/Pokemon Blue	NaN	Role-Playing	Nintendo	NaN	NaN	NaN	
...	...	...	...	...	...	...	...	...	
16593	16596.0	Woody Woodpecker in Crazy Castle 5	GBA	Platform	Kemco	0.01	0.00	0.00	
16594	16597.0	Men in Black II: Alien Escape	GC	Shooter	Infogrames	0.01	0.00	0.00	
16595	16598.0	SCORE International Baja 1000: The Official Game	PS2	Racing	Activision	0.00	0.00	0.00	
16596	16599.0	Know How 2	DS	Puzzle	7G//AMES	0.00	0.01	0.00	
16597	16600.0	Spirits & Spells	GBA	Platform	Wanadoo	0.01	0.00	0.00	

16598 rows × 10 columns

In [9]:
```
game.shape
```

Out[9]:
```
(16598, 10)
```

In [10]:
```
game.size
```

Out[10]:
```
165980
```

In [11]:
```
game.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16598 entries, 0 to 16597
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Rank          15581 non-null  float64
 1   Name          16359 non-null  object 
 2   Platform      16173 non-null  object 
 3   Genre         16334 non-null  object 
 4   Publisher     15505 non-null  object 
 5   NA_Sales      15196 non-null  float64
 6   EU_Sales      14039 non-null  float64
 7   JP_Sales      13597 non-null  float64
 8   Other_Sales   13846 non-null  float64
 9   Global_Sales  14866 non-null  float64
dtypes: float64(6), object(4)
memory usage: 1.3+ MB
```

In [12]:
```
game.tail()
```

Out[12]:

	Rank	Name	Platform	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Othe
16593	16596.0	Woody Woodpecker in Crazy Castle 5	GBA	Platform	Kemco	0.01	0.00	0.0	
16594	16597.0	Men in Black II: Alien Escape	GC	Shooter	Infogrames	0.01	0.00	0.0	
16595	16598.0	SCORE International Baja 1000: The Official Game	PS2	Racing	Activision	0.00	0.00	0.0	
16596	16599.0	Know How 2	DS	Puzzle	7G//AMES	0.00	0.01	0.0	
16597	16600.0	Spirits & Spells	GBA	Platform	Wanadoo	0.01	0.00	0.0	

In [13]:
```
game.head()
```

Out[13]:

	Rank	Name	Platform	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales
0	1.0	Wii Sports	Wii	Sports	Nintendo	41.49	29.02	3.77	8.46
1	NaN	Super Mario Bros.	NES	Platform	NaN	NaN	NaN	NaN	0.77
2	NaN	Mario Kart Wii	Wii	Racing	Nintendo	15.85	12.88	3.79	3.31
3	NaN	Wii Sports Resort	NaN	Sports	Nintendo	15.75	11.01	3.28	2.96
4	NaN	Pokemon Red/Pokemon Blue	NaN	Role-Playing	Nintendo	NaN	NaN	NaN	1.00

In [14]:
```
game.isnull()
```

Out[14]:

	Rank	Name	Platform	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	G
0	False	False	False	False	False	False	False	False	False	
1	True	False	False	False	True	True	True	True	False	
2	True	False	False	False	False	False	False	False	False	
3	True	False	True	False	False	False	False	False	False	
4	True	False	True	False	False	True	True	True	False	
...	...	...	...	...	...	...	...	...	...	
16593	False	False	False	False	False	False	False	False	False	
16594	False	False	False	False	False	False	False	False	False	
16595	False	False	False	False	False	False	False	False	False	
16596	False	False	False	False	False	False	False	False	False	
16597	False	False	False	False	False	False	False	False	False	

16598 rows × 10 columns

In [15]:
```
game.notnull()
```

Out[15]:

	Rank	Name	Platform	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	G
0	True	True	True	True	True	True	True	True	True	
1	False	True	True	True	False	False	False	False	True	
2	False	True	True	True	True	True	True	True	True	
3	False	True	False	True	True	True	True	True	True	
4	False	True	False	True	True	False	False	False	True	
...	...	...	...	...	...	...	...	...	...	
16593	True	True	True	True	True	True	True	True	True	
16594	True	True	True	True	True	True	True	True	True	
16595	True	True	True	True	True	True	True	True	True	
16596	True	True	True	True	True	True	True	True	True	
16597	True	True	True	True	True	True	True	True	True	

16598 rows × 10 columns

In [16]:
```
game.isna()
```

Out[16]:

	Rank	Name	Platform	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	G
0	False	False	False	False	False	False	False	False	False	
1	True	False	False	False	True	True	True	True	False	
2	True	False	False	False	False	False	False	False	False	
3	True	False	True	False	False	False	False	False	False	
4	True	False	True	False	False	True	True	True	False	
...	...	...	...	...	...	...	...	...	...	
16593	False	False	False	False	False	False	False	False	False	
16594	False	False	False	False	False	False	False	False	False	
16595	False	False	False	False	False	False	False	False	False	
16596	False	False	False	False	False	False	False	False	False	
16597	False	False	False	False	False	False	False	False	False	

16598 rows × 10 columns

In [17]:
```
game.describe()
```

Out[17]:

	Rank	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
count	15581.000000	15196.000000	14039.000000	13597.000000	13846.000000	14866.000000
mean	8066.450934	0.226803	0.112119	0.059312	0.035376	0.446997
std	4740.104254	0.666912	0.452258	0.232105	0.154902	1.459950
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.010000
25%	4032.000000	0.000000	0.000000	0.000000	0.000000	0.060000
50%	7927.000000	0.070000	0.020000	0.000000	0.010000	0.150000
75%	11865.000000	0.210000	0.080000	0.030000	0.020000	0.390000
max	16600.000000	41.490000	29.020000	6.040000	8.460000	82.740000

In [18]:
```
game.dropna()
```

Out[18]:

	Rank	Name	Platform	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Oth
0	1.0	Wii Sports	Wii	Sports	Nintendo	41.49	29.02	3.77	
11	12.0	Mario Kart DS	DS	Racing	Nintendo	9.81	7.57	4.13	
34	35.0	Call of Duty: Black Ops II	PS3	Shooter	Activision	4.99	5.88	0.65	
35	36.0	Call of Duty: Black Ops II	X360	Shooter	Activision	8.25	4.30	0.07	
41	42.0	Animal Crossing: Wild World	DS	Simulation	Nintendo	2.55	3.52	5.33	
...	...	...	...	...	...	...	...	...	
16593	16596.0	Woody Woodpecker in Crazy Castle 5	GBA	Platform	Kemco	0.01	0.00	0.00	
16594	16597.0	Men in Black II: Alien Escape	GC	Shooter	Infogrames	0.01	0.00	0.00	
16595	16598.0	SCORE International Baja 1000: The Official Game	PS2	Racing	Activision	0.00	0.00	0.00	
16596	16599.0	Know How 2	DS	Puzzle	7G//AMES	0.00	0.01	0.00	
16597	16600.0	Spirits & Spells	GBA	Platform	Wanadoo	0.01	0.00	0.00	

11078 rows × 10 columns

In [19]:
```
game.isna().sum()
```

Out[19]:
```
Rank            1017
Name             239
Platform         425
Genre            264
Publisher       1093
NA_Sales        1402
EU_Sales        2559
JP_Sales        3001
Other_Sales     2752
Global_Sales    1732
dtype: int64
```

In [20]:
```
game.isnull().sum()
```

Out[20]:
```
Rank            1017
Name             239
Platform         425
Genre            264
Publisher       1093
NA_Sales        1402
EU_Sales        2559
JP_Sales        3001
Other_Sales     2752
Global_Sales    1732
dtype: int64
```

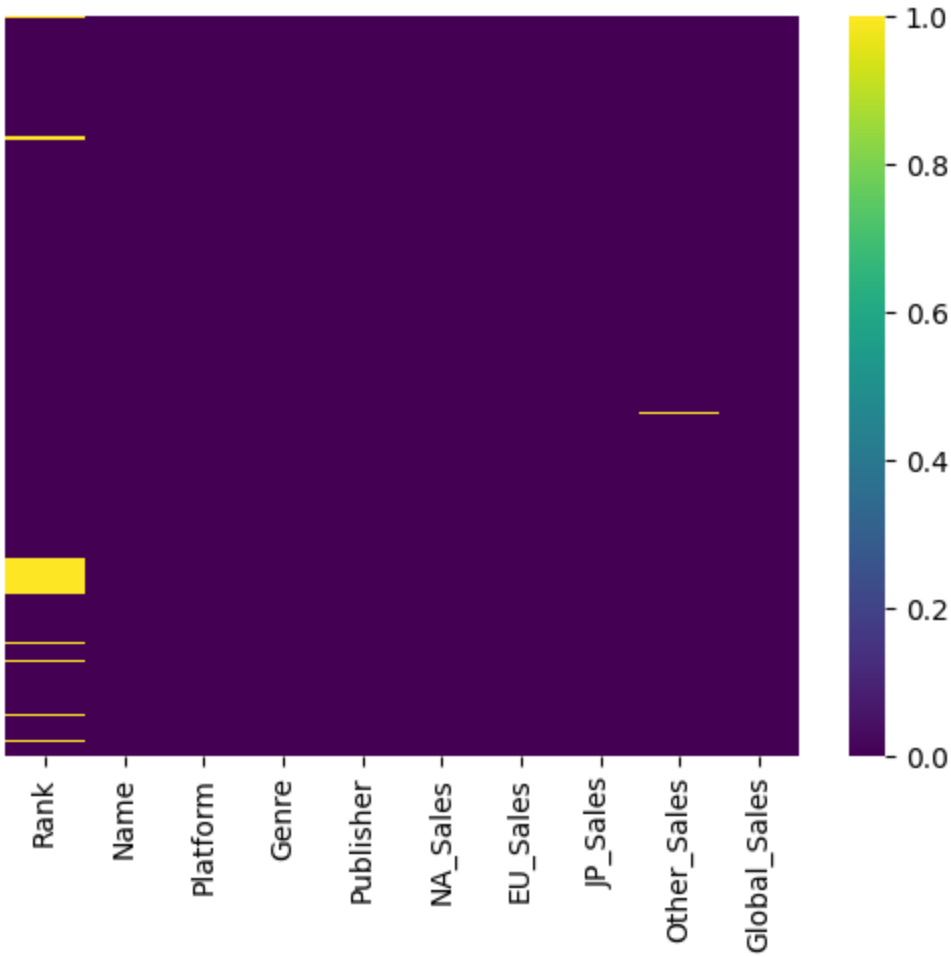
In [21]:
```
(game.isnull().sum()/(len(game)))*100
```

Out[21]:
```
Rank             6.127244
Name             1.439933
Platform         2.560549
Genre            1.590553
Publisher        6.585131
NA_Sales         8.446801
EU_Sales        15.417520
JP_Sales        18.080492
Other_Sales     16.580311
Global_Sales    10.434992
dtype: float64
```

In [22]:
```
game.dropna(subset=["JP_Sales"],inplace=True)
game.dropna(subset=["EU_Sales"],inplace=True)
game.dropna(subset=["Name"],inplace=True)
game.dropna(subset=["Genre"],inplace=True)
game.dropna(subset=["Platform"],inplace=True)
game['Publisher'].fillna(method='bfill' , inplace=True)
game["NA_Sales"].ffill(axis=0,inplace=True)
game['Global_Sales'].fillna(game['Global_Sales'].mean(), inplace=True)
```

In [23]:
```
sns.heatmap(game.isnull(),yticklabels=False,cmap="viridis") #heatmap
```

Out[23]: <Axes: >



In [24]:
```
game.drop(['Genre'],axis=1,inplace=True)#droping the column name (Genre)
```

In [25]:
```
game #the 9 columns will be displayed
```

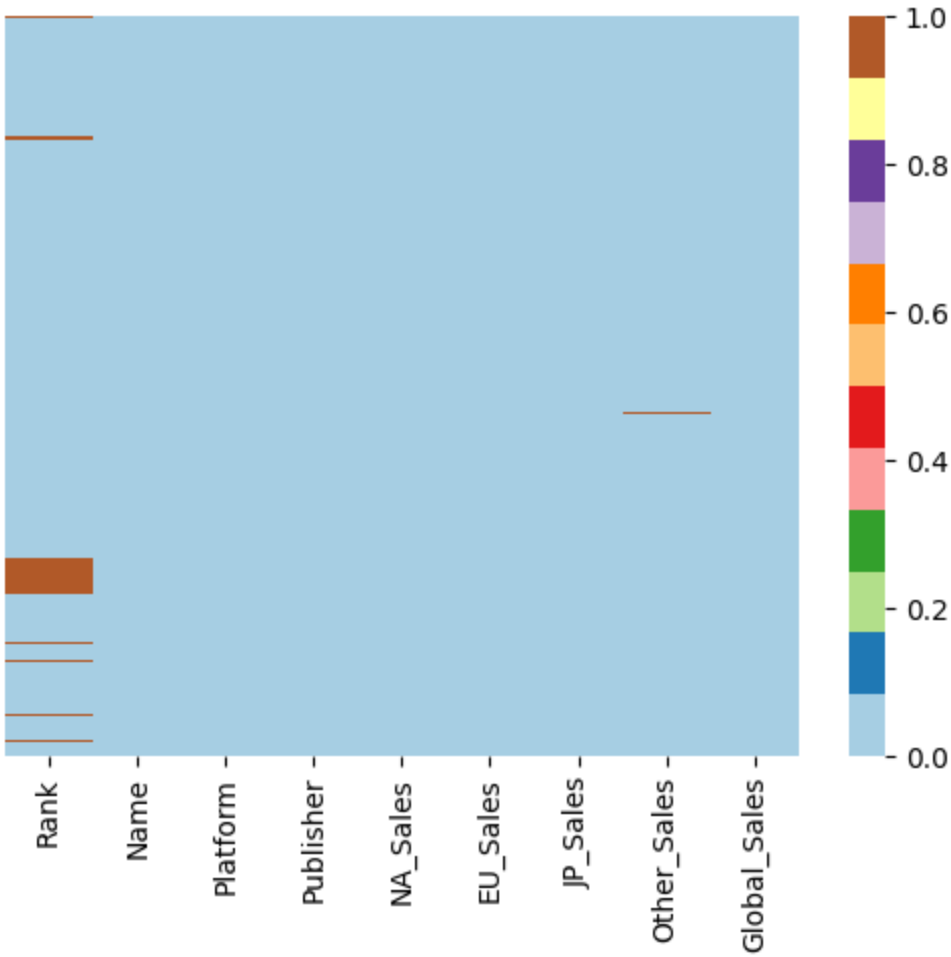
Out[25]:

	Rank	Name	Platform	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sa
0	1.0	Wii Sports	Wii	Nintendo	41.49	29.02	3.77	8
2	NaN	Mario Kart Wii	Wii	Nintendo	15.85	12.88	3.79	3
10	NaN	Nintendogs	DS	Nintendo	9.07	11.00	1.93	2
11	12.0	Mario Kart DS	DS	Nintendo	9.81	7.57	4.13	1
20	21.0	Pokemon Diamond/Pokemon Pearl	DS	Nintendo	9.81	4.52	6.04	1
...	...	...	...	...	...	...	...	
16593	16596.0	Woody Woodpecker in Crazy Castle 5	GBA	Kemco	0.01	0.00	0.00	0
16594	16597.0	Men in Black II: Alien Escape	GC	Infogrames	0.01	0.00	0.00	0
16595	16598.0	SCORE International Baja 1000: The Official Game	PS2	Activision	0.00	0.00	0.00	0
16596	16599.0	Know How 2	DS	7G//AMES	0.00	0.01	0.00	0
16597	16600.0	Spirits & Spells	GBA	Wanadoo	0.01	0.00	0.00	0

12756 rows × 9 columns

In [26]:
```
sns.heatmap(game.isnull(),yticklabels=False,cmap="Paired") #heatmap
```

Out[26]:
```
<Axes: >
```



In [27]:
```
data_set=[['Wii Sports',82.74],['New Super Mario Bros',30.01],['Call of Duty:M
game_test=pd.DataFrame(data_set,columns=['Name','Global_Sales'])#creating data
```

In [28]:
```
game_test
```

Out[28]:

	Name	Global_Sales
0	Wii Sports	82.74
1	New Super Mario Bros	30.01
2	Call of Duty:Modern Warefare 3	14.76
3	Grand Theft Auto 4	11.02

In [29]:
```
game.columns
```

Out[29]:
```
Index(['Rank', 'Name', 'Platform', 'Publisher', 'NA_Sales', 'EU_Sales',
       'JP_Sales', 'Other_Sales', 'Global_Sales'],
      dtype='object')
```

In [30]:
```
game.drop(['Other_Sales'],axis=1,inplace=True) #droping the Other_sales
```

In [31]:
```
game
```

Out[31]:

	Rank	Name	Platform	Publisher	NA_Sales	EU_Sales	JP_Sales	Global_S
0	1.0	Wii Sports	Wii	Nintendo	41.49	29.02	3.77	8
2	NaN	Mario Kart Wii	Wii	Nintendo	15.85	12.88	3.79	3
10	NaN	Nintendogs	DS	Nintendo	9.07	11.00	1.93	2
11	12.0	Mario Kart DS	DS	Nintendo	9.81	7.57	4.13	2
20	21.0	Pokemon Diamond/Pokemon Pearl	DS	Nintendo	9.81	4.52	6.04	1
...	...	...	...	...	...	...	...	
16593	16596.0	Woody Woodpecker in Crazy Castle 5	GBA	Kemco	0.01	0.00	0.00	
16594	16597.0	Men in Black II: Alien Escape	GC	Infogrames	0.01	0.00	0.00	
16595	16598.0	SCORE International Baja 1000: The Official Game	PS2	Activision	0.00	0.00	0.00	
16596	16599.0	Know How 2	DS	7G//AMES	0.00	0.01	0.00	
16597	16600.0	Spirits & Spells	GBA	Wanadoo	0.01	0.00	0.00	

12756 rows × 8 columns

In [32]:
```
game.rename(columns={'JP_Sales':'Japan_Sales'}) #I have change the columns nam
```

Out[32]:

	Rank	Name	Platform	Publisher	NA_Sales	EU_Sales	Japan_Sales	Globa
0	1.0	Wii Sports	Wii	Nintendo	41.49	29.02	3.77	
2	NaN	Mario Kart Wii	Wii	Nintendo	15.85	12.88	3.79	
10	NaN	Nintendogs	DS	Nintendo	9.07	11.00	1.93	
11	12.0	Mario Kart DS	DS	Nintendo	9.81	7.57	4.13	
20	21.0	Pokemon Diamond/Pokemon Pearl	DS	Nintendo	9.81	4.52	6.04	
...	...	...	...	...	...	...	...	
16593	16596.0	Woody Woodpecker in Crazy Castle 5	GBA	Kemco	0.01	0.00	0.00	
16594	16597.0	Men in Black II: Alien Escape	GC	Infogrames	0.01	0.00	0.00	
16595	16598.0	SCORE International Baja 1000: The Official Game	PS2	Activision	0.00	0.00	0.00	
16596	16599.0	Know How 2	DS	7G//AMES	0.00	0.01	0.00	
16597	16600.0	Spirits & Spells	GBA	Wanadoo	0.01	0.00	0.00	

12756 rows × 8 columns

In [33]:
```
game[['Name','Platform','Publisher']].describe()
```

Out[33]:

	Name	Platform	Publisher
count	12756	12756	12756
unique	9544	31	542
top	FIFA 14	DS	Electronic Arts
freq	8	1728	933

In [34]:
```
video=game.sample(10)
```

In [35]:
```
video
```

Out[35]:

	Rank	Name	Platform	Publisher	NA_Sales	EU_Sales	JP_Sales	Global_Sa
14000	14002.0	Smashing Drive	XB	Namco Bandai Games	0.03	0.01	0.00	0
2336	2338.0	Pokemon Card GB2: Here Comes Team GR!	GB	Sony Computer Entertainment	0.00	0.00	0.89	0
11017	11019.0	Junior Mystery Quest	DS	GSP	0.07	0.01	0.00	0
15264	15267.0	Shinken de Watashi ni Koi Shinasai! R	PS3	Minato Station	0.00	0.00	0.02	0
5320	5322.0	Guilty Gear X2	PS2	Sammy Corporation	0.09	0.07	0.16	0
13717	13719.0	Ou to Maou to 7-nin no Himegimitachi: Shin Ous...	PSV	Konami Digital Entertainment	0.00	0.00	0.04	0
12651	12653.0	Brave: A Warrior's Tale	X360	SouthPeak Games	0.05	0.01	0.00	0
15418	15421.0	Super Fruit Fall	Wii	System 3 Arcade Software	0.01	0.00	0.00	0
7872	7874.0	Saints Row IV	PC	Deep Silver	0.11	0.06	0.00	0
8570	8572.0	The King of Fighters XII	PS3	Ignition Entertainment	0.11	0.01	0.03	0

In [36]:
```
cat_data=game.select_dtypes(include=object)
num_data=game.select_dtypes(exclude=object)
```

In [37]:
```
cat_data
```

Out[37]:

	Name	Platform	Publisher
0	Wii Sports	Wii	Nintendo
2	Mario Kart Wii	Wii	Nintendo
10	Nintendogs	DS	Nintendo
11	Mario Kart DS	DS	Nintendo
20	Pokemon Diamond/Pokemon Pearl	DS	Nintendo
...	...	...	...
16593	Woody Woodpecker in Crazy Castle 5	GBA	Kemco
16594	Men in Black II: Alien Escape	GC	Infogrames
16595	SCORE International Baja 1000: The Official Game	PS2	Activision
16596	Know How 2	DS	7G//AMES
16597	Spirits & Spells	GBA	Wanadoo

12756 rows × 3 columns

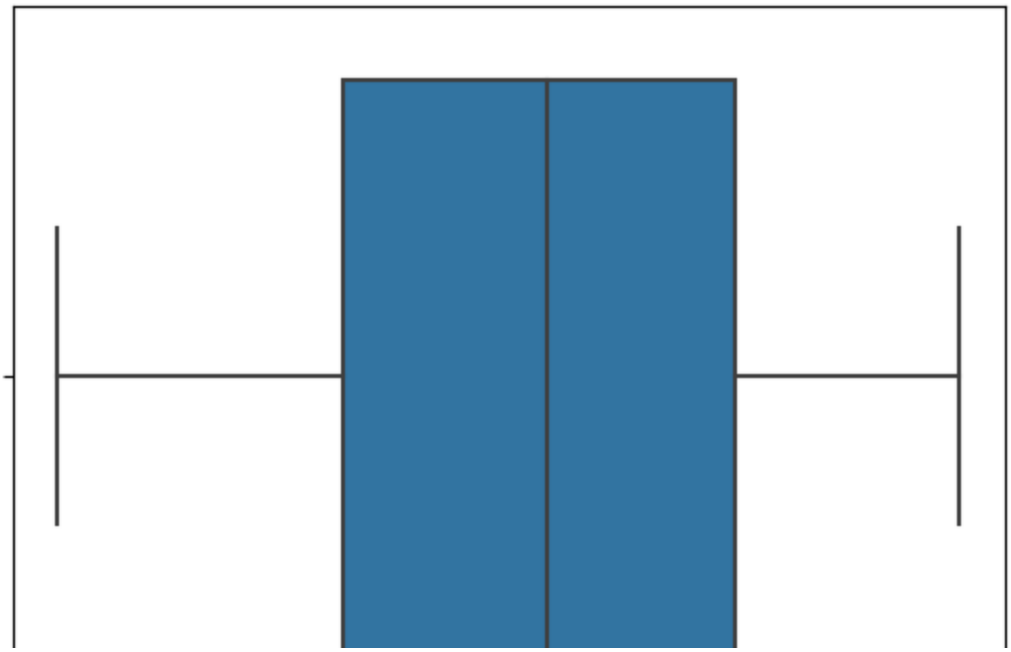
In [38]:
```
num_data
```

Out[38]:

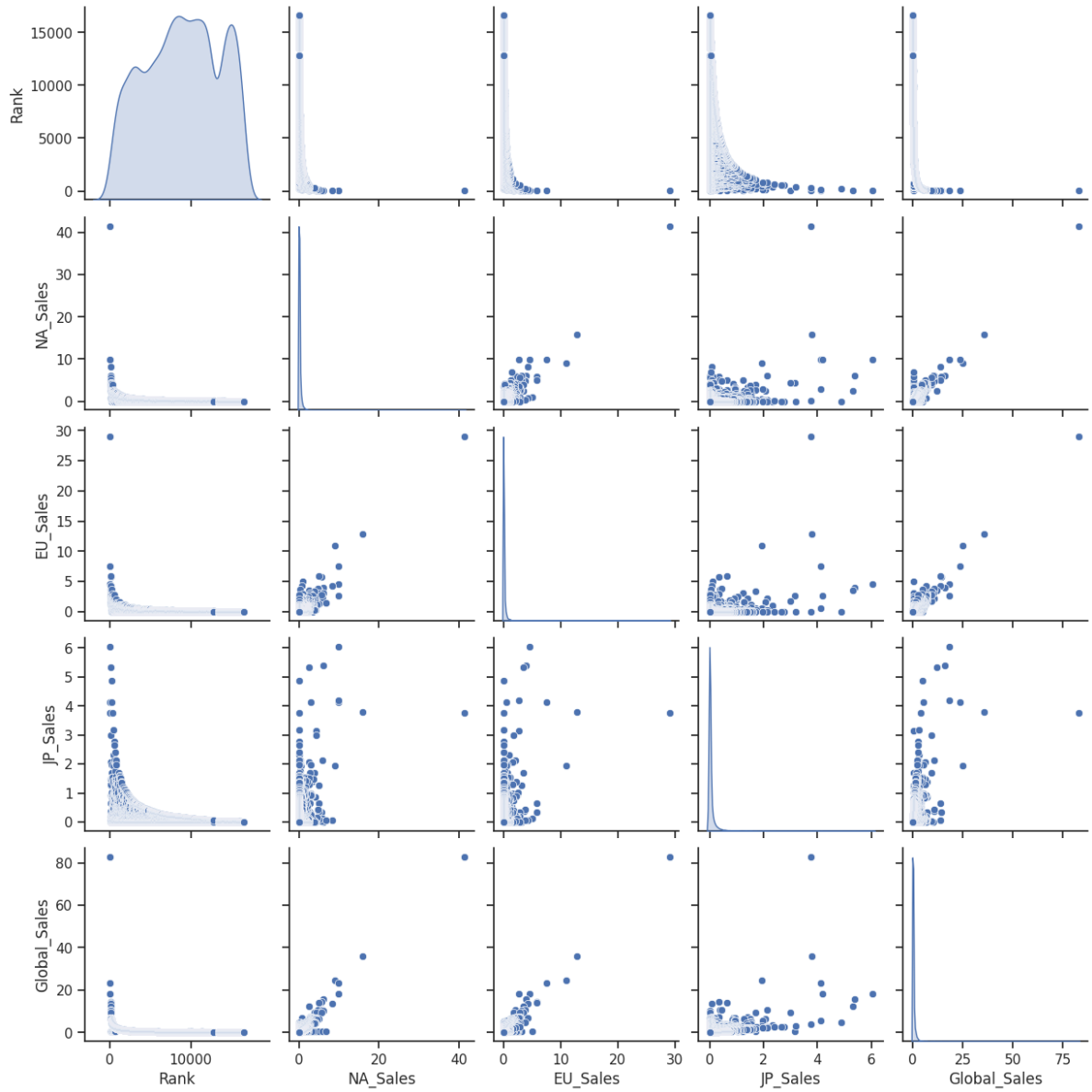
	Rank	NA_Sales	EU_Sales	JP_Sales	Global_Sales
0	1.0	41.49	29.02	3.77	82.74
2	NaN	15.85	12.88	3.79	35.82
10	NaN	9.07	11.00	1.93	24.76
11	12.0	9.81	7.57	4.13	23.42
20	21.0	9.81	4.52	6.04	18.36
...	...	...	...	...	...
16593	16596.0	0.01	0.00	0.00	0.01
16594	16597.0	0.01	0.00	0.00	0.01
16595	16598.0	0.00	0.00	0.00	0.01
16596	16599.0	0.00	0.01	0.00	0.01
16597	16600.0	0.01	0.00	0.00	0.01

12756 rows × 5 columns

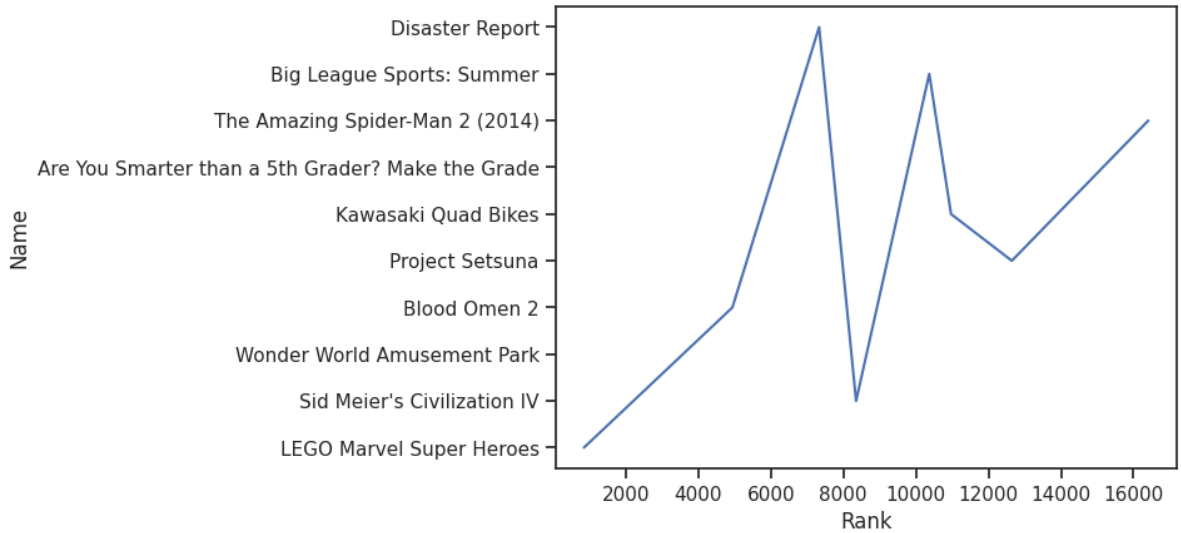
In [39]:
```
for i in num_data.columns:
    sns.boxplot(x=game[i])
    plt.show()
```



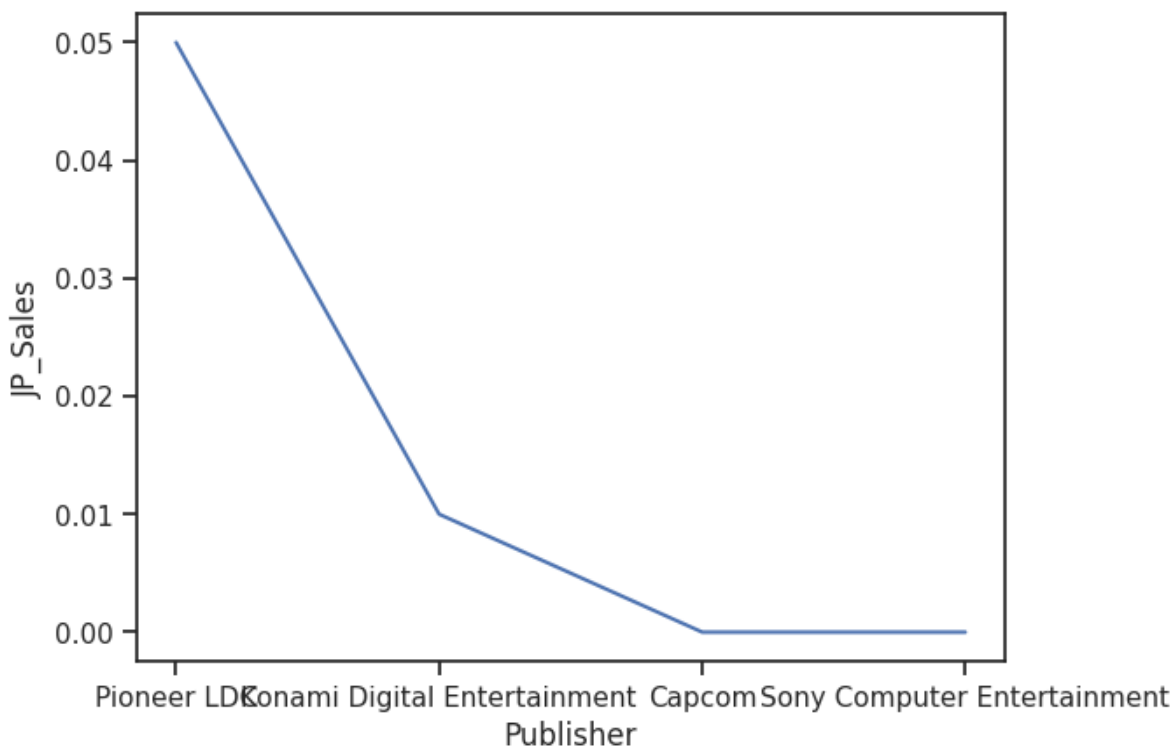
In [40]:
```
sns.set(style="ticks")
sns.pairplot(game,diag_kind="kde",markers="o")
plt.show()
```



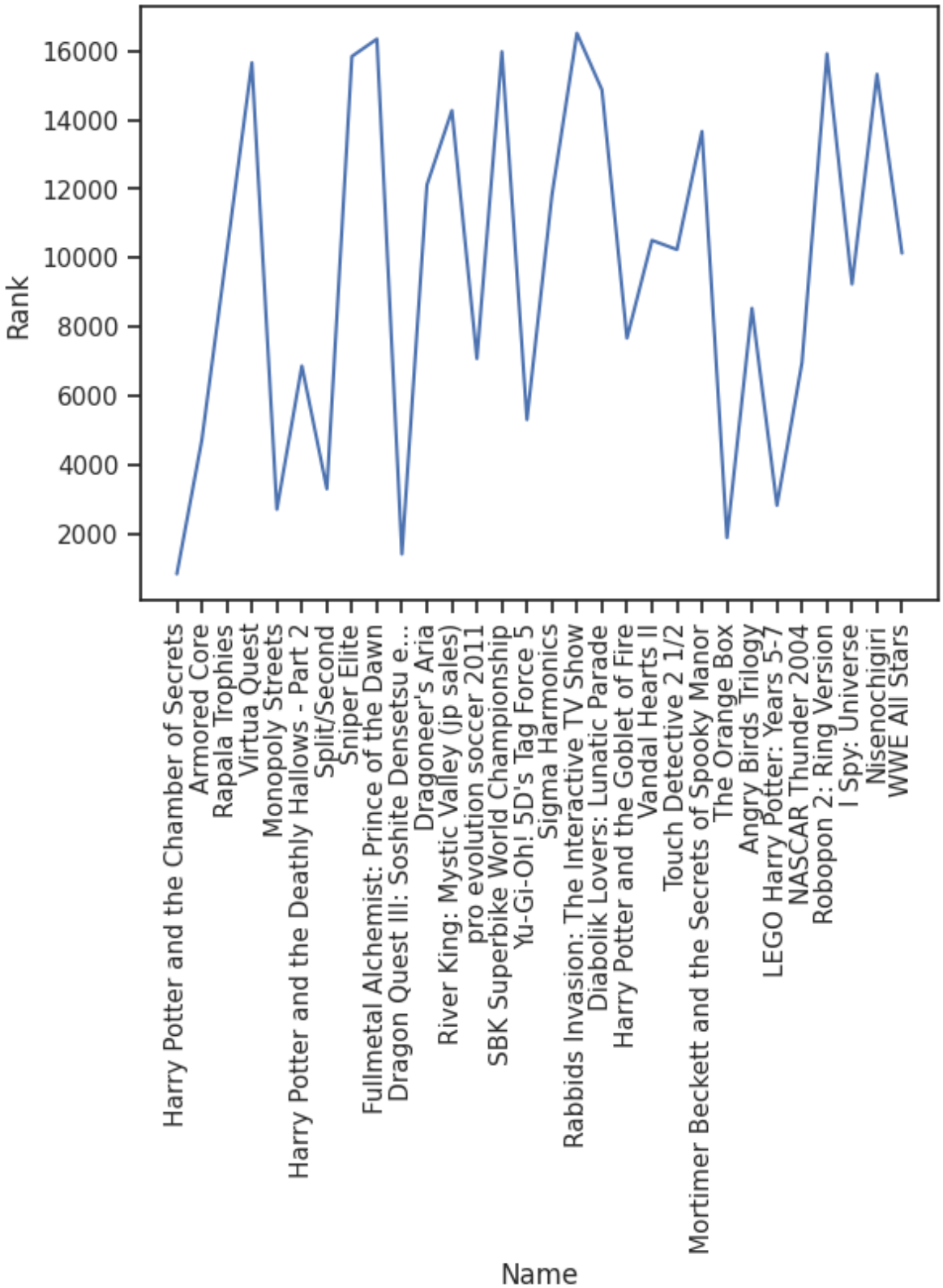
In [41]:
```
a=game.sample(10)
sns.lineplot(x="Rank",y="Name",data=a)
plt.show()
```



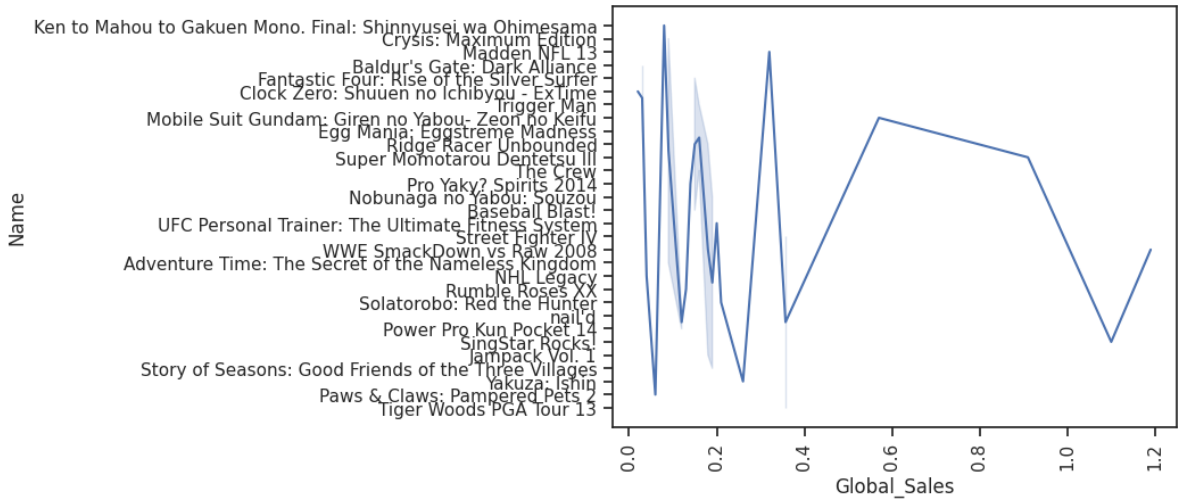
In [42]:
```
a=game.sample(5)
sns.lineplot(x="Publisher",y="JP_Sales",data=a)
plt.show()
```



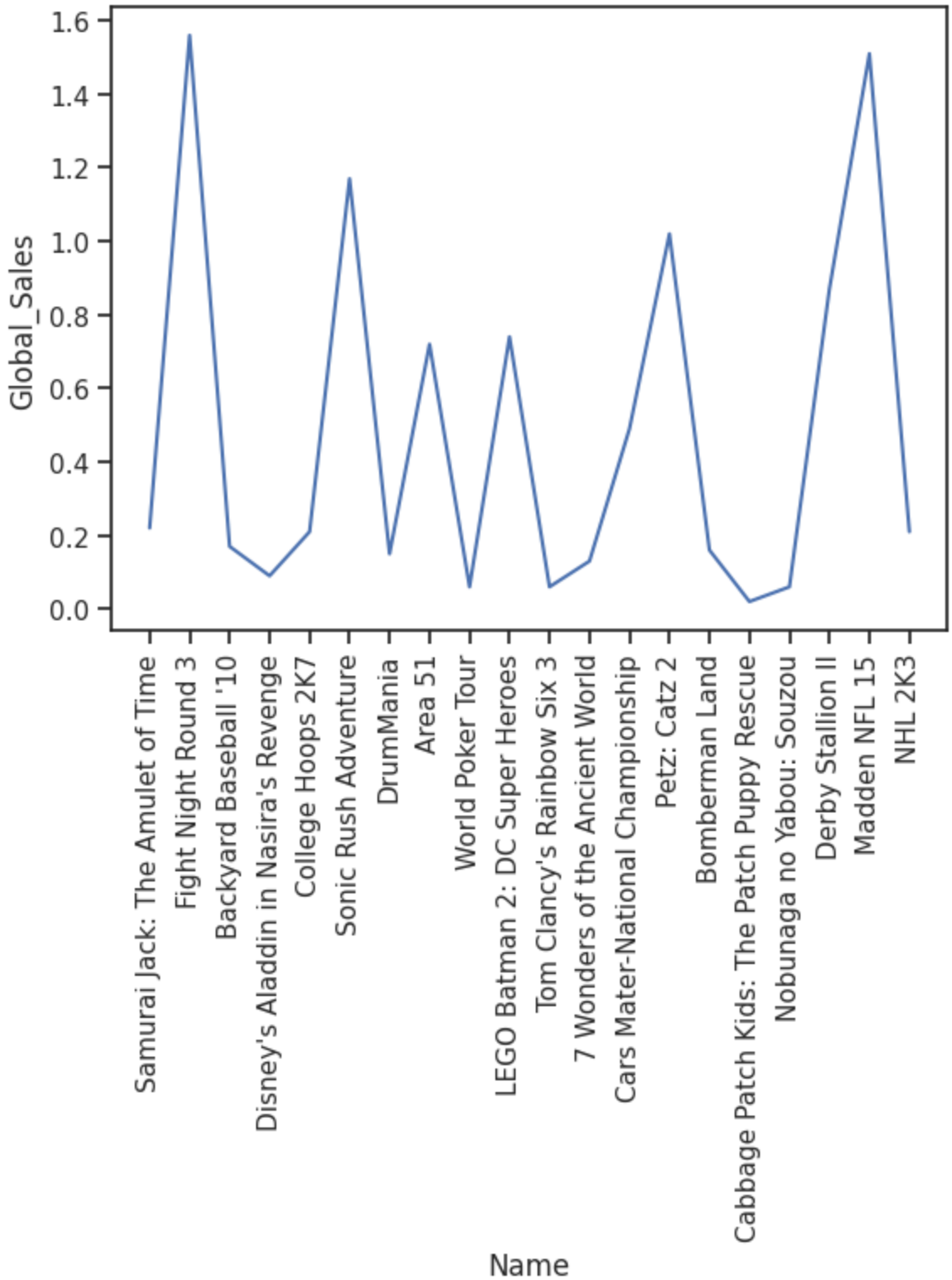
In [43]:
```
a=game.sample(30)
figsize=(20,15)
sns.lineplot(x="Name",y="Rank",data=a)
plt.xticks(rotation=90)
plt.show()
```



In [44]:
```
a=game.sample(30)
sns.lineplot(x="Global_Sales",y="Name",data=a)
plt.xticks(rotation=90)
plt.show()
```

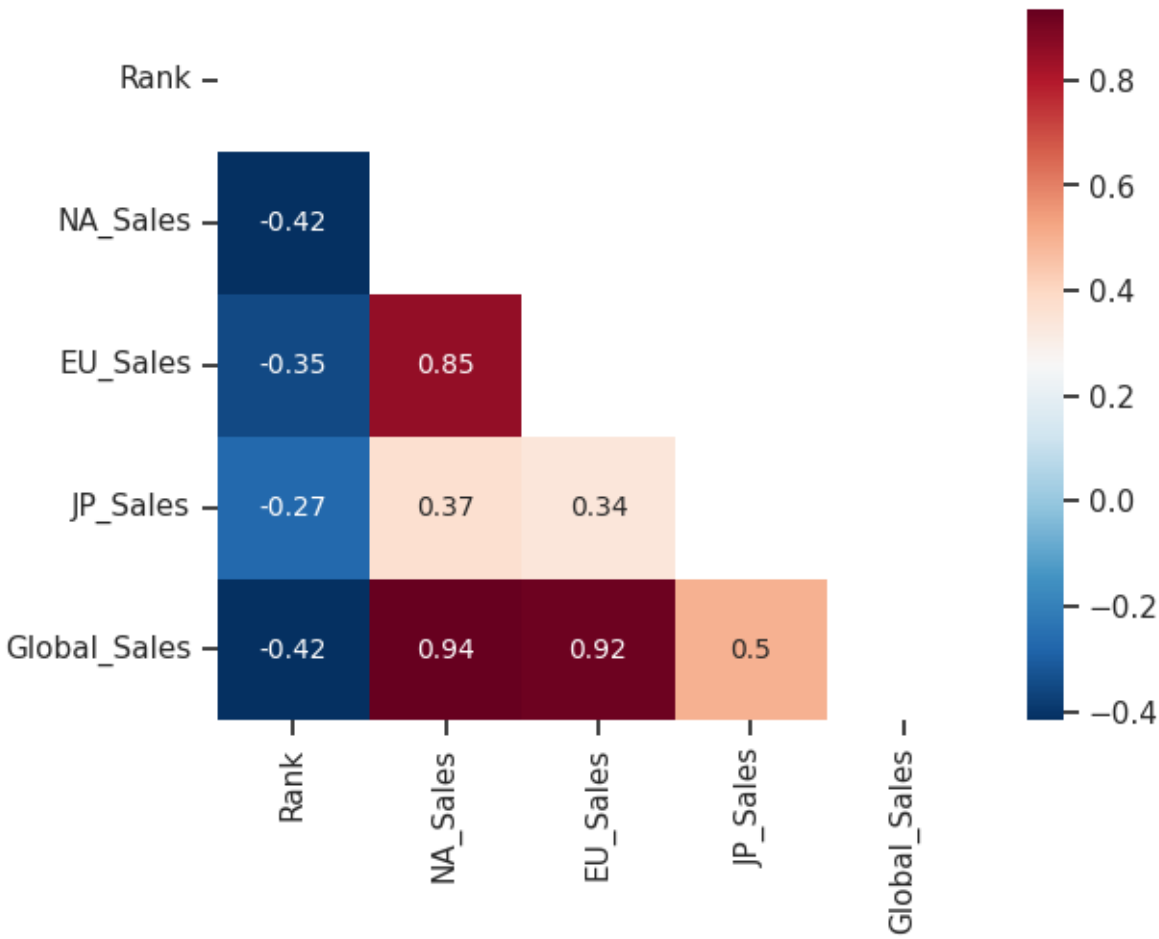


In [45]:
```
a=game.sample(20)
figsize=(10,12)
sns.lineplot(x="Name",y="Global_Sales",data=a)
plt.xticks(rotation=90)
plt.show()
```



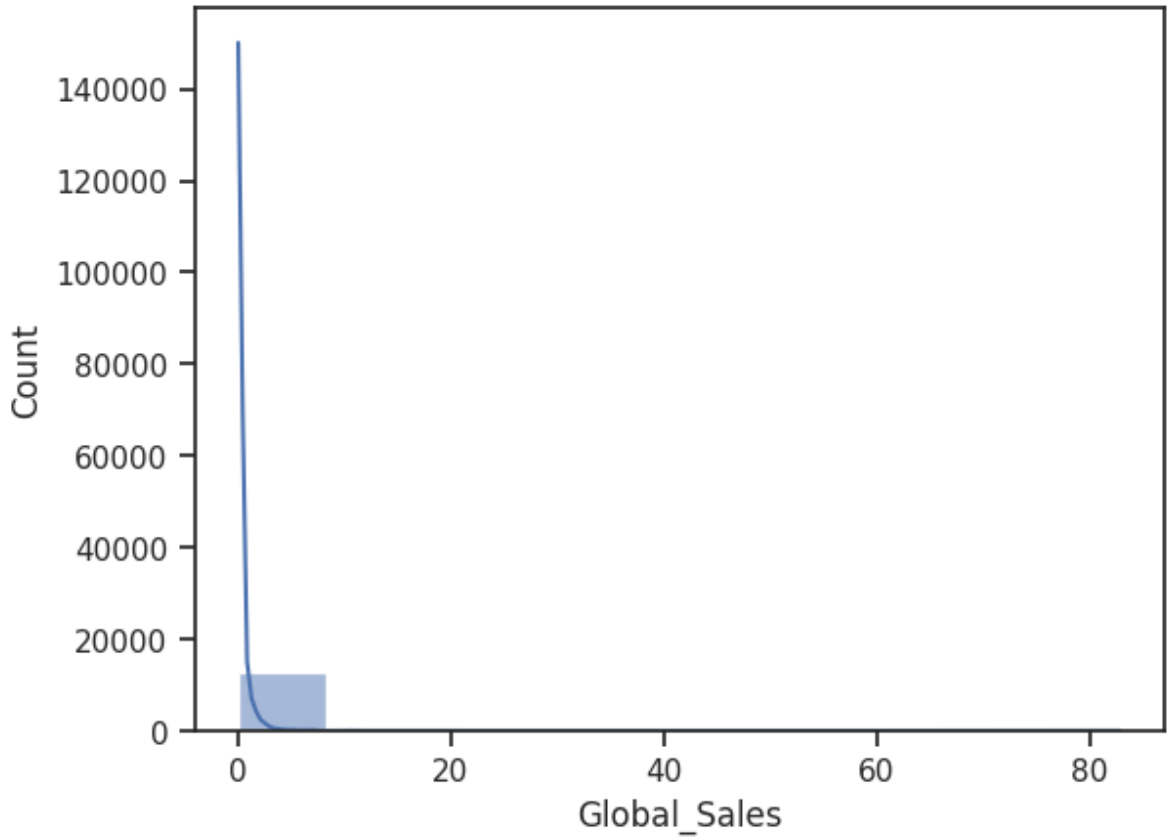
In [46]:
```
corr=num_data.corr()
msk=np.triu(np.ones_like(corr))
sns.heatmap(corr,cmap=plt.cm.RdBu_r,annot=True,annot_kws={'size':10},mask=msk)
```

Out[46]: <Axes: >

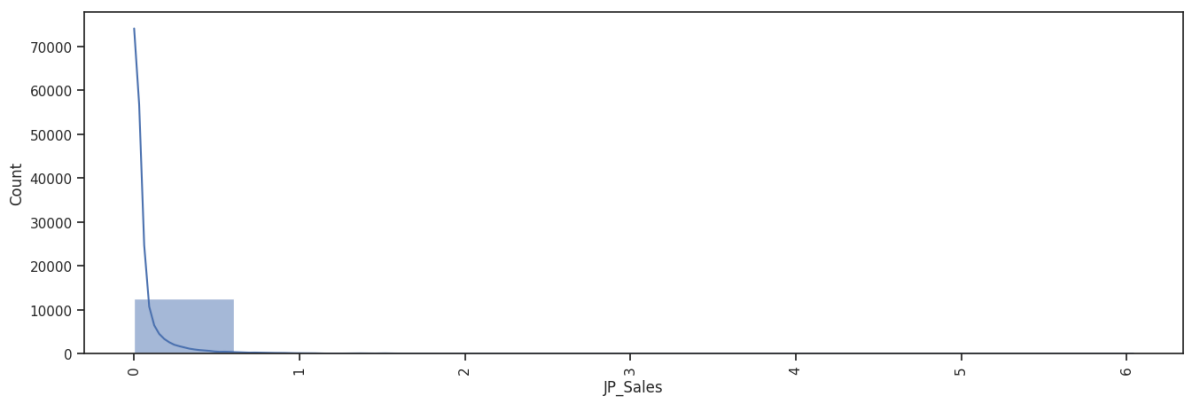


In [47]:
```
sns.histplot(game["Global_Sales"],bins=10,kde=True)
```

Out[47]: <Axes: xlabel='Global_Sales', ylabel='Count'>



In [48]:
```
plt.figure(figsize=(16,5))
sns.histplot(game["JP_Sales"],bins=10,kde=True)
plt.xticks(rotation=90)
plt.show()
```



In [49]:
```
z=game.groupby("Publisher")["Publisher"].count()
```

In [50]:
```
z
```

Out[50]:
```
Publisher
10TACLE Studios                  3
1C Company                       3
20th Century Fox Video Games     4
2D Boy                           1
3DO                             27
                                ..
id Software                      1
imageepoch Inc.                  2
inXile Entertainment             1
mixi, Inc                        1
responDESIGN                     1
Name: Publisher, Length: 542, dtype: int64
```

In [112]:
```
g=game.sample(100)
```

In [113]:
```
g
```

Out[113]:

	Rank	Name	Platform	Publisher	NA_Sales	EU_Sales	JP_Sales	Global_Sales
2628	2630.0	Tom Clancy's Splinter Cell: Double Agent	X360	Ubisoft	0.67	0.05	0.01	0.78
7073	7075.0	Wipeout 2	3DS	Activision	0.22	0.00	0.00	0.23
5969	5971.0	Power Pro Kun Pocket 4	GBA	Konami Digital Entertainment	0.00	0.00	0.29	0.29
6343	6345.0	Jikkyou Powerful Pro Yakyuu 5	N64	Konami Digital Entertainment	0.00	0.00	0.27	0.27
5951	5953.0	Dragon Ball Z: Budokai Tenkaichi 2	Wii	Atari	0.24	0.03	0.00	0.30
...	...	...	...	...	...	...	...	..
8739	8741.0	Dragon Ball Z: Harukanaru Densetsu (JP sales)	DS	Namco Bandai Games	0.00	0.00	0.15	0.15
14418	NaN	Away: Shuffle Dungeon	DS	Majesco Entertainment	0.02	0.00	0.00	0.03
11633	11635.0	WWE Crush Hour	GC	THQ	0.06	0.02	0.00	0.08
15532	15535.0	Kamen Rider: Battride War II	WiiU	Namco Bandai Games	0.00	0.00	0.02	0.02
15007	15010.0	Supermodel Makeover by Lauren Luke	DS	Avanquest	0.00	0.02	0.00	0.02

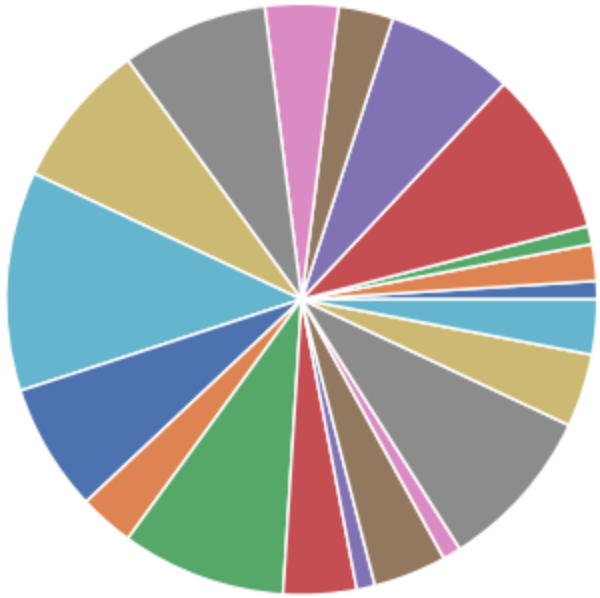
100 rows × 8 columns

In [116]:
```
game["JP_Sales"].sample(10)
```

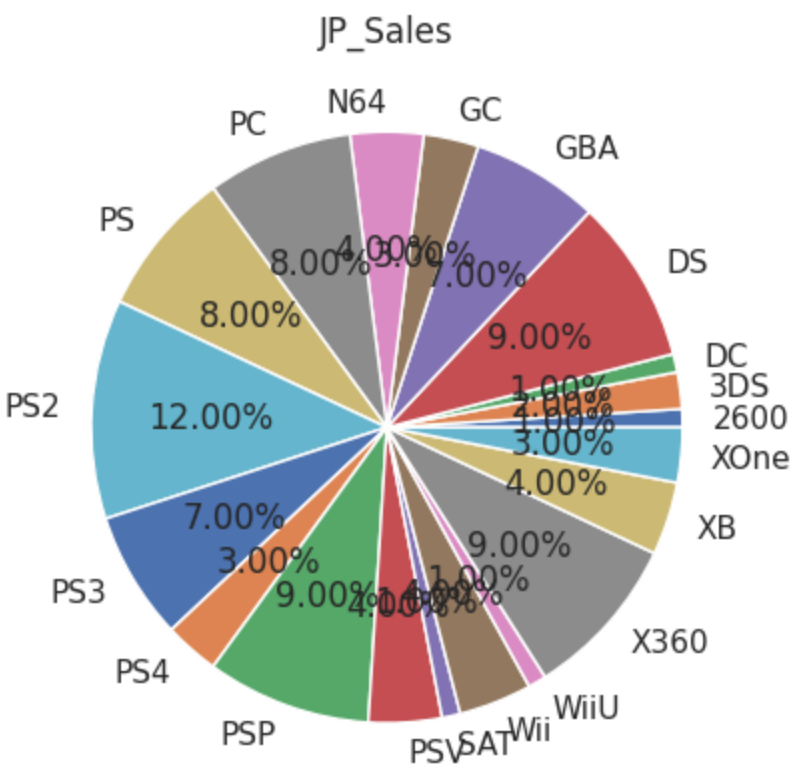
Out[116]:
```
3219     0.00
3138     0.00
11143    0.00
8312     0.00
6140     0.00
12188    0.00
15520    0.00
2409     0.86
10667    0.00
16517    0.00
Name: JP_Sales, dtype: float64
```

In [114]:
```
z=g.groupby("Platform")["Platform"].count()
plt.pie(z)
```

Out[114]:
```
([<matplotlib.patches.Wedge at 0x7d0aeaed9a20>,
  <matplotlib.patches.Wedge at 0x7d0aeaed9750>,
  <matplotlib.patches.Wedge at 0x7d0aeaedac50>,
  <matplotlib.patches.Wedge at 0x7d0aeaedb550>,
  <matplotlib.patches.Wedge at 0x7d0aeaedbe50>,
  <matplotlib.patches.Wedge at 0x7d0ae7943ee0>,
  <matplotlib.patches.Wedge at 0x7d0ae79431f0>,
  <matplotlib.patches.Wedge at 0x7d0ae79420e0>,
  <matplotlib.patches.Wedge at 0x7d0ae7940ee0>,
  <matplotlib.patches.Wedge at 0x7d0ae7940280>,
  <matplotlib.patches.Wedge at 0x7d0aeaed98d0>,
  <matplotlib.patches.Wedge at 0x7d0ae7941510>,
  <matplotlib.patches.Wedge at 0x7d0ae7941e10>,
  <matplotlib.patches.Wedge at 0x7d0ae7942710>,
  <matplotlib.patches.Wedge at 0x7d0ae7942e00>,
  <matplotlib.patches.Wedge at 0x7d0ae7943580>,
  <matplotlib.patches.Wedge at 0x7d0ae7943c40>,
  <matplotlib.patches.Wedge at 0x7d0b61eb64a0>,
  <matplotlib.patches.Wedge at 0x7d0b642bd270>,
  <matplotlib.patches.Wedge at 0x7d0b620ca350>],
 [Text(1.099457216426567, 0.03455183421390152, ''),
  Text(1.0913261718331657, 0.13786655385541477, ''),
  Text(1.0735084393121108, 0.23995756025946832, ''),
  Text(0.9468162265587848, 0.5599455626442484, ''),
  Text(0.5599455404824152, 0.9468162396652564, ''),
  Text(0.23995754769582914, 1.0735084421204166, ''),
  Text(-1.2873679144189123e-08, 1.0999999999999999, ''),
  Text(-0.40493701130464616, 1.0227541331501238, ''),
  Text(-0.8475645614732935, 0.7011663954687103, ''),
  Text(-1.09782939975141, 0.06906959563700359, ''),
  Text(-0.9468162462184919, -0.5599455294014982, ''),
  Text(-0.7274430771565426, -0.8251221542880912, ''),
  Text(-0.372611734556999, -1.03496883782577, ''),
  Text(0.06906955709217517, -1.097829402176445, ''),
  Text(0.23995754769582886, -1.0735084421204169, ''),
  Text(0.40493698736535794, -1.0227541426283582, ''),
  Text(0.5599455349419563, -0.9468162429418744, ''),
  Text(0.8251221670583664, -0.7274430626715094, ''),
  Text(1.0461621663333946, -0.3399186987098812, ''),
  Text(1.0951181600699413, -0.10351915515993293, '')])
```



In [115]:
```
a=game.sample(5)
plt.pie(z,labels=z.index,autopct="%.2f%%")
plt.xticks(rotation=90)
plt.title("JP_Sales")
plt.show()
```



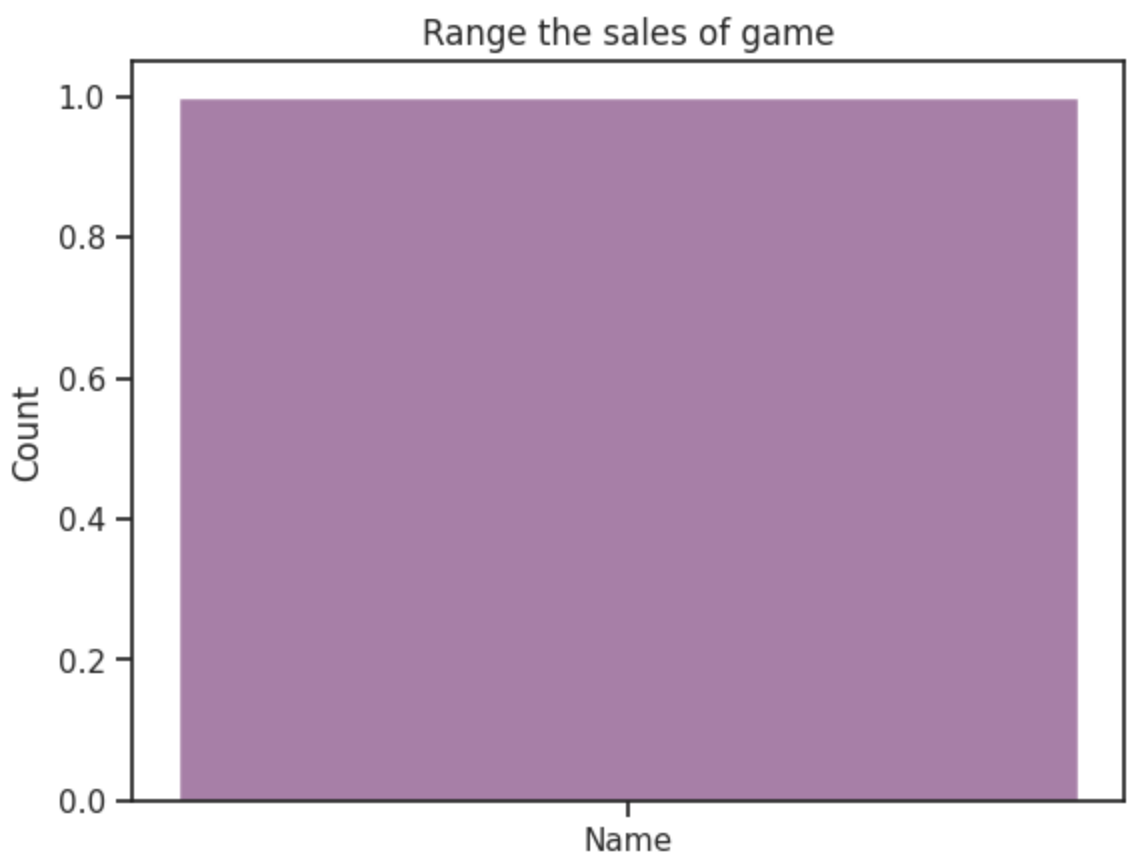
In [110]:
```
z=game.groupby("Name")["Name"].count()
```

In [53]:
```
z
```

Out[53]:
```
Name
.hack//G.U. Vol.1//Rebirth                 1
.hack//G.U. Vol.2//Reminisce               1
.hack//G.U. Vol.2//Reminisce (jp sales)    1
.hack//Link                                1
.hack//Mutation Part 2                     1
                                          ..
thinkSMART: Chess for Kids                 1
uDraw Studio                               1
uDraw Studio: Instant Artist               1
wwe Smackdown vs. Raw 2006                 1
¡Shin Chan Flipa en colores!               1
Name: Name, Length: 9544, dtype: int64
```

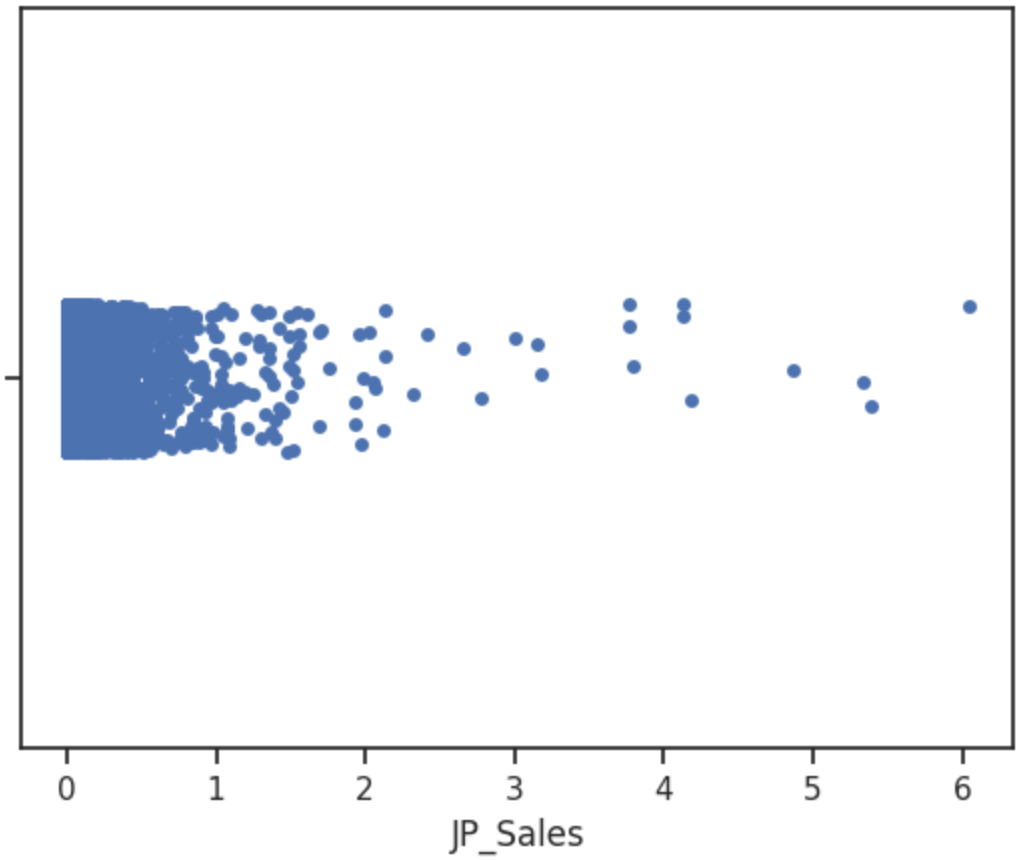
In [54]:
```
sns.histplot(x=['Name'],color='#500050',kde=True)
plt.title(f'Range the sales of game')
```

Out[54]:
```
Text(0.5, 1.0, 'Range the sales of game')
```



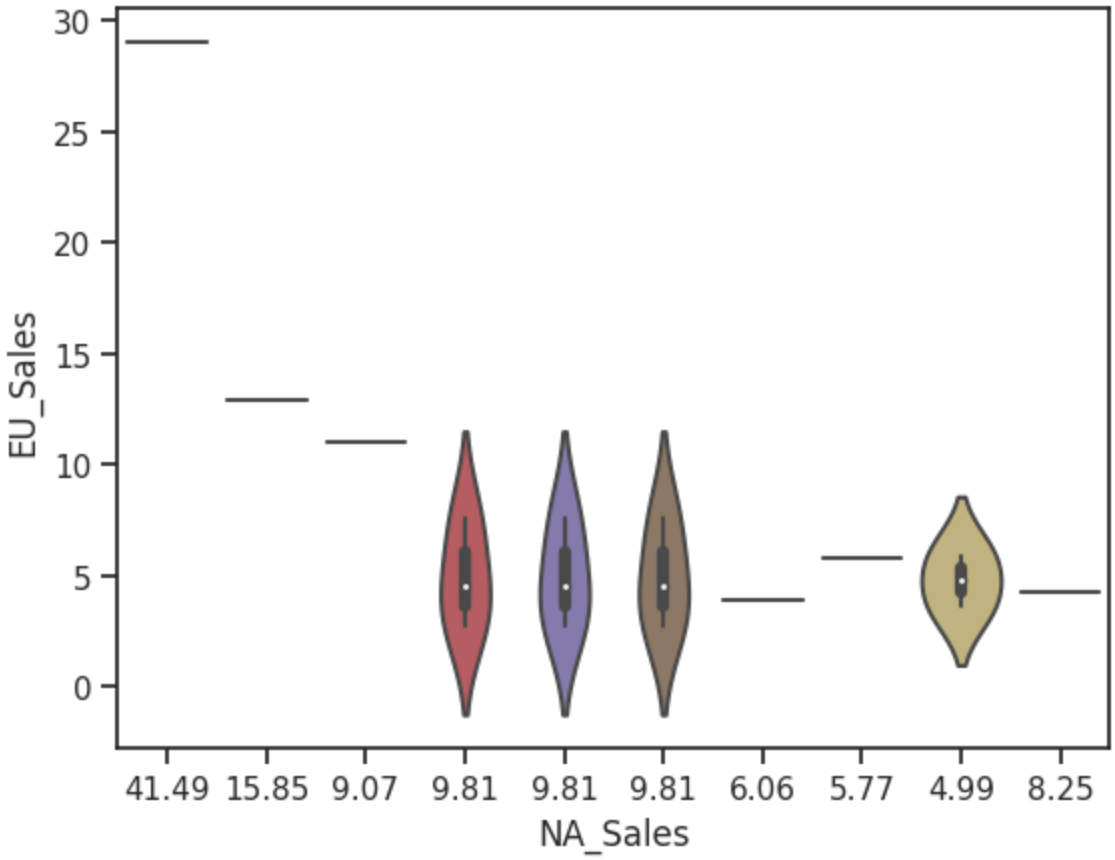
In [55]:
```
sns.stripplot(x="JP_Sales",data=game)
```

Out[55]:
```
<Axes: xlabel='JP_Sales'>
```



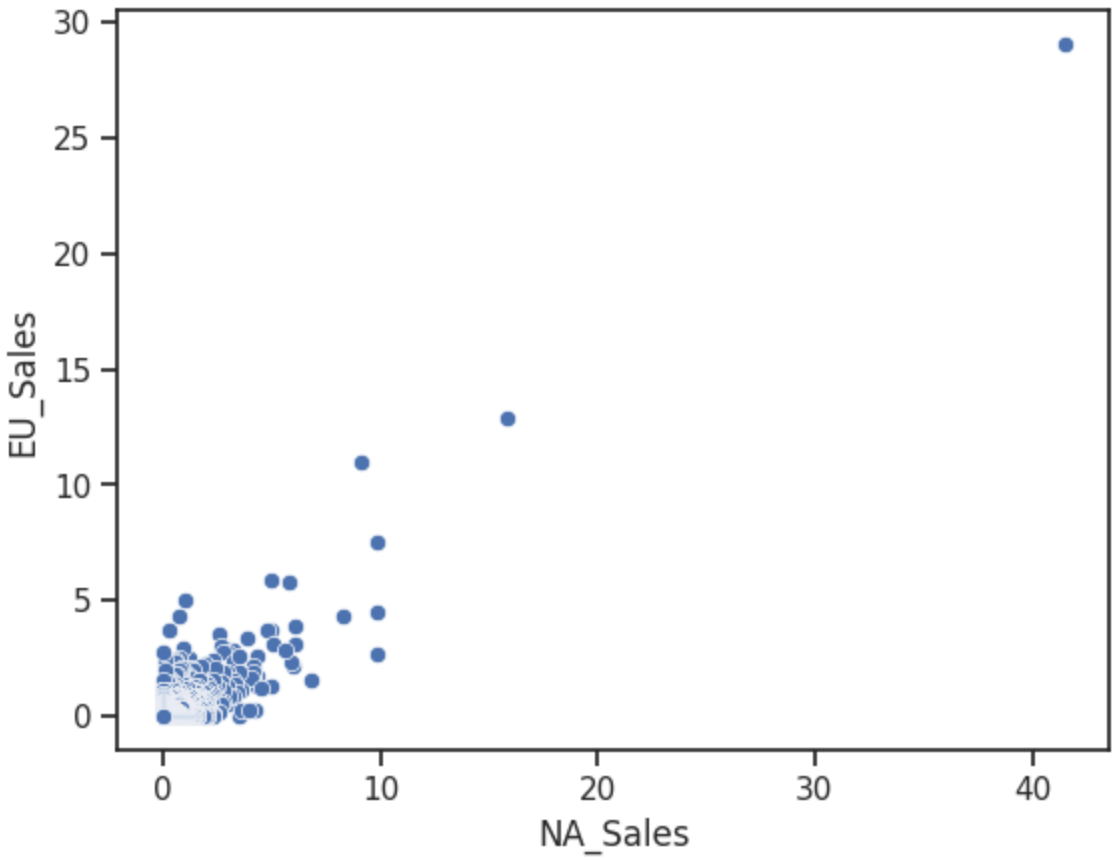
In [56]:
```
sns.violinplot(x="NA_Sales",y="EU_Sales",data=game,order=game.NA_Sales.iloc[:1
```

Out[56]:
```
<Axes: xlabel='NA_Sales', ylabel='EU_Sales'>
```



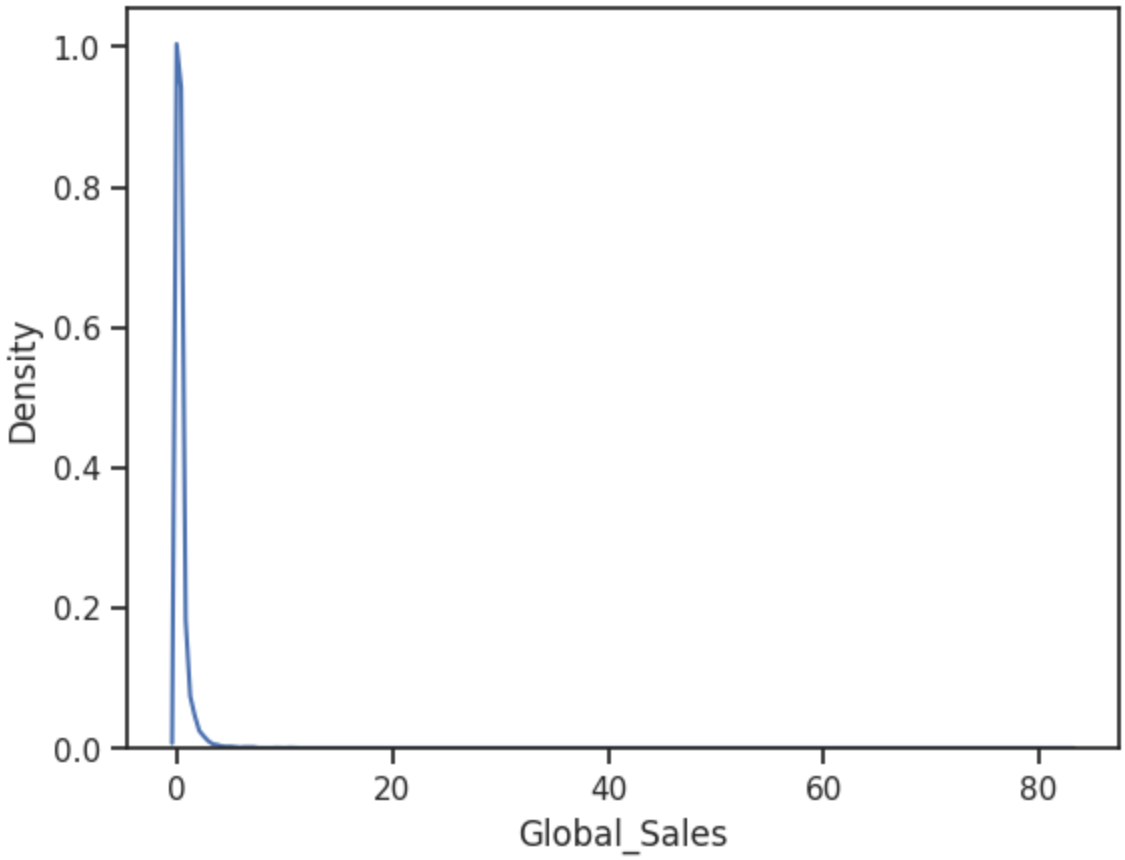
In [57]:
```
sns.scatterplot(x="NA_Sales",y="EU_Sales",data=game)
```

Out[57]:
```
<Axes: xlabel='NA_Sales', ylabel='EU_Sales'>
```



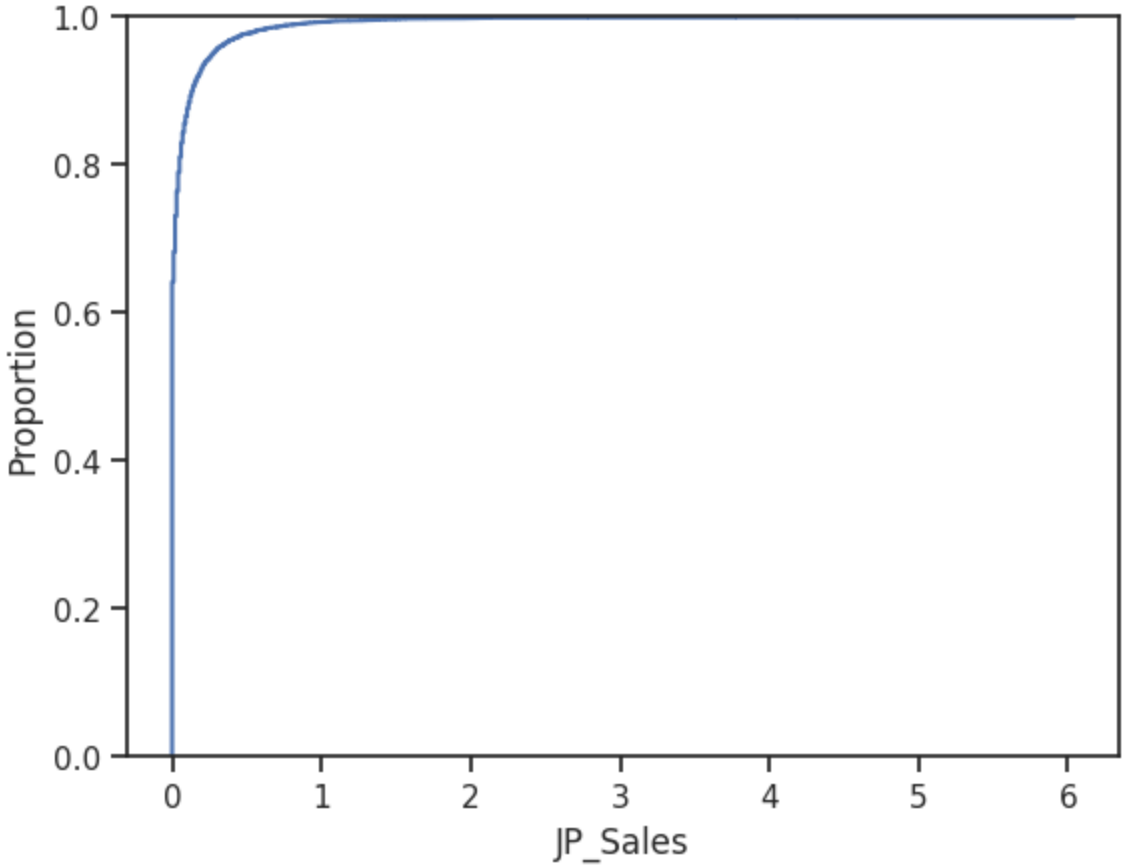
In [58]:
```
sns.kdeplot(game.Global_Sales)
```

Out[58]:
```
<Axes: xlabel='Global_Sales', ylabel='Density'>
```



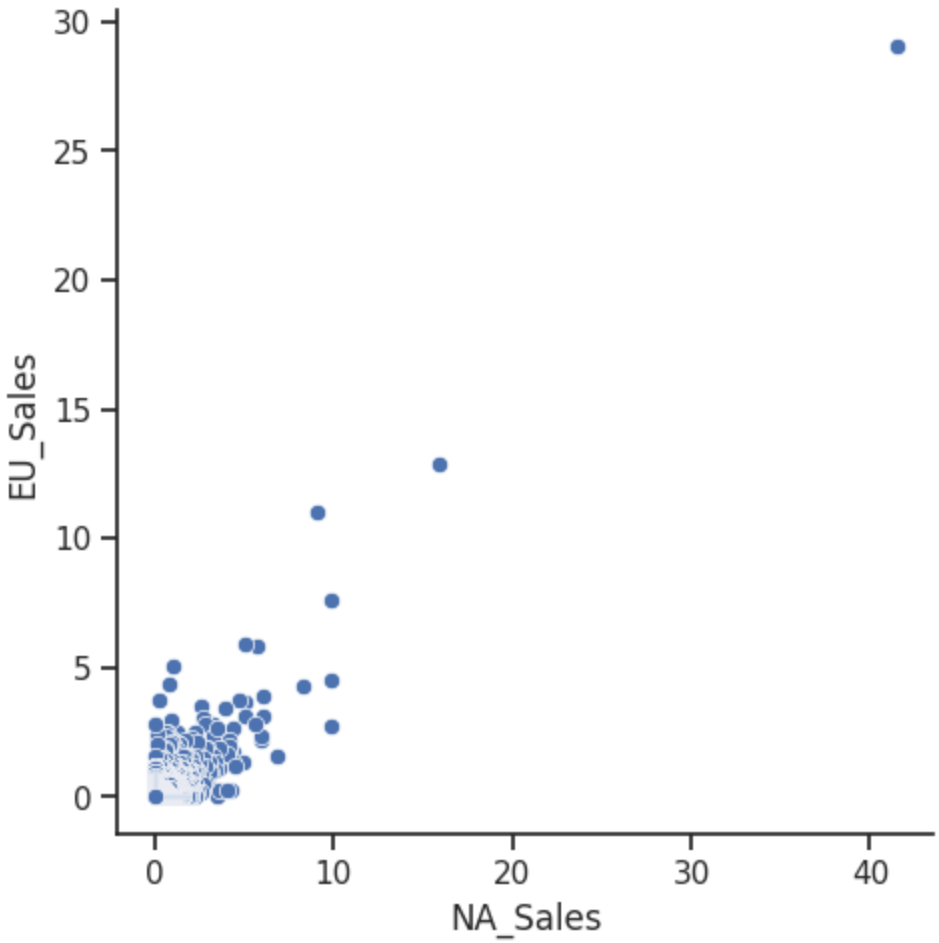
In [59]:
```
sns.ecdfplot(game.JP_Sales)
```

Out[59]:
```
<Axes: xlabel='JP_Sales', ylabel='Proportion'>
```



In [60]:
```
sns.relplot(x="NA_Sales",y="EU_Sales",data=game)
```

Out[60]:
```
<seaborn.axisgrid.FacetGrid at 0x7d0b6925c4f0>
```



In [61]:
```
from scipy import stats
z_score=stats.zscore(game["Global_Sales"])
z_score_outliers=(z_score<-3)|(z_score>3)
```

In [62]:
```
z_score_outlier_rows=game[z_score_outliers]
print("outliers detected by Z-score:",z_score_outlier_rows)
```

```
outliers detected by Z-score:       Rank                                Na
me Platform  \
0      1.0                            Wii Sports      Wii
2      NaN                        Mario Kart Wii      Wii
10     NaN                            Nintendogs       DS
11    12.0                         Mario Kart DS       DS
20    21.0          Pokemon Diamond/Pokemon Pearl       DS
..     ...                                   ...      ...
326  327.0          Ratchet & Clank: Size Matters      PSP
327  328.0                        Just Dance 2014      Wii
329  330.0                      Super Paper Mario      Wii
330  331.0  Harry Potter and the Sorcerer's Stone       PS
331  332.0                The Witcher 3: Wild Hunt      PS4

              Publisher  NA_Sales  EU_Sales  JP_Sales  Global_Sales
0              Nintendo     41.49     29.02      3.77         82.74
2              Nintendo     15.85     12.88      3.79         35.82
10             Nintendo      9.07     11.00      1.93         24.76
11             Nintendo      9.81      7.57      4.13         23.42
20             Nintendo      9.81      4.52      6.04         18.36
..                  ...       ...       ...       ...           ...
326             Ubisoft      1.40      1.40      0.10          3.77
327             Ubisoft      1.98      1.47      0.00          3.76
329            Nintendo      1.98      0.88      0.59          3.76
330     Electronic Arts      1.37      2.00      0.14          3.73
331  Namco Bandai Games      0.96      2.00      0.21          3.73

[93 rows x 8 columns]
```

In [63]:
```
game.shape
```

Out[63]:
```
(12756, 8)
```

In [64]:
```
x=(z_score>-3)&(z_score<3)
```

In [65]:
```
new_df=game[x]
```

In [66]:
```
new_df
```

Out[66]:

	Rank	Name	Platform	Publisher	NA_Sales	EU_Sales	JP_Sales	Global_Sa
58	NaN	Pokemon FireRed/Pokemon LeafGreen	GBA	Nintendo	4.34	2.65	3.15	0.357
59	60.0	Super Mario 64	DS	Nintendo	5.08	3.11	1.25	0.357
70	NaN	Call of Duty 4: Modern Warfare	X360	Activision	5.91	2.38	0.13	0.357
79	NaN	Halo 2	XB	Microsoft Game Studios	6.82	1.53	0.05	0.357
82	NaN	FIFA Soccer 13	PS3	Nintendo	1.06	5.05	0.13	0.357
...	...	...	...	...	...	...	...	
16593	16596.0	Woody Woodpecker in Crazy Castle 5	GBA	Kemco	0.01	0.00	0.00	0.010
16594	16597.0	Men in Black II: Alien Escape	GC	Infogrames	0.01	0.00	0.00	0.010
16595	16598.0	SCORE International Baja 1000: The Official Game	PS2	Activision	0.00	0.00	0.00	0.010
16596	16599.0	Know How 2	DS	7G//AMES	0.00	0.01	0.00	0.010
16597	16600.0	Spirits & Spells	GBA	Wanadoo	0.01	0.00	0.00	0.010

12663 rows × 8 columns

In [67]:
```
z_score=stats.zscore(new_df["NA_Sales"])
z_score_outlier=(z_score<-3)|(z_score>3)
```

In [68]:
```
z_score_outlier_row=new_df[z_score_outlier]
print("outliers detected by Z-score:",z_score_outlier_row)
```

```
outliers detected by Z-score:         Rank                                N
ame Platform  \
58       NaN     Pokemon FireRed/Pokemon LeafGreen      GBA
59      60.0                        Super Mario 64       DS
70       NaN        Call of Duty 4: Modern Warfare     X360
79       NaN                                Halo 2       XB
82       NaN                        FIFA Soccer 13      PS3
...      ...                                   ...      ...
1552  1554.0                 The Sims 2: Nightlife       PC
1593  1595.0                RollerCoaster Tycoon 2       PC
1599  1601.0  Transformers: Autobots / Decepticons       DS
1642  1644.0                           NBA Live 99       PS
1766  1768.0                               Kaboom!     2600

                   Publisher  NA_Sales  EU_Sales  JP_Sales  Global_Sales
58                  Nintendo      4.34      2.65      3.15      0.357022
59                  Nintendo      5.08      3.11      1.25      0.357022
70                Activision      5.91      2.38      0.13      0.357022
79    Microsoft Game Studios      6.82      1.53      0.05      0.357022
82                  Nintendo      1.06      5.05      0.13      0.357022
...                      ...       ...       ...       ...           ...
1552         Electronic Arts      1.22      0.05      0.00      1.270000
1593                   Atari      1.19      0.05      0.00      1.250000
1599              Activision      1.12      0.03      0.00      1.240000
1642         Electronic Arts      1.13      0.05      0.00      1.220000
1766              Activision      1.07      0.07      0.00      1.150000

[289 rows x 8 columns]
```

In [69]:
```
p=(z_score>-3)&(z_score<3)
df_new=new_df[p]
```

In [70]:
```
df_new
```

Out[70]:

	Rank	Name	Platform	Publisher	NA_Sales	EU_Sales	JP_Sales	Global_Sales
344	345.0	Gran Turismo 6	PS3	Activision	0.71	1.80	0.40	3.64
347	348.0	FIFA Soccer 10	PS3	Activision	0.60	2.46	0.05	3.63
348	349.0	Pro Evolution Soccer 2008	PS2	Activision	0.05	0.00	0.64	3.63
376	377.0	Clubhouse Games	DS	Nintendo	0.59	1.83	0.73	3.50
377	378.0	FIFA Soccer 2004	PS2	Nintendo	0.59	2.36	0.04	3.49
...	...	...	...	...	...	...	...	...
16593	16596.0	Woody Woodpecker in Crazy Castle 5	GBA	Kemco	0.01	0.00	0.00	0.01
16594	16597.0	Men in Black II: Alien Escape	GC	Infogrames	0.01	0.00	0.00	0.01
16595	16598.0	SCORE International Baja 1000: The Official Game	PS2	Activision	0.00	0.00	0.00	0.01
16596	16599.0	Know How 2	DS	7G//AMES	0.00	0.01	0.00	0.01
16597	16600.0	Spirits & Spells	GBA	Wanadoo	0.01	0.00	0.00	0.01

12374 rows × 8 columns

In [71]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
import joblib
```

In [72]:
```
categorical_cols=['Name','Platform']
encoder=OneHotEncoder(drop='first',sparse=False)
encoder_cols=pd.DataFrame(encoder.fit_transform(df_new[categorical_cols]),colu
numerical_cols=['NA_Sales','JP_Sales']
scaler=StandardScaler()
scaled_cols=pd.DataFrame(scaler.fit_transform(df_new[numerical_cols]),columns=
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:86
8: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and
will be removed in 1.4. `sparse_output` is ignored unless you leave `sparse`
to its default value.
  warnings.warn(
```

In [73]:
```
encoder_cols
```

Out[73]:

	Name_.hack//G.U. Vol.2//Reminisce	Name_.hack//G.U. Vol.2//Reminisce (jp sales)	Name_.hack//Link	Name_.hack//Mutation Part 2	Name_.hack/
0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	
...	...	...	...	...	
12369	0.0	0.0	0.0	0.0	
12370	0.0	0.0	0.0	0.0	
12371	0.0	0.0	0.0	0.0	
12372	0.0	0.0	0.0	0.0	
12373	0.0	0.0	0.0	0.0	

12374 rows × 9361 columns

In [74]:
```
scaled_cols
```

Out[74]:

	NA_Sales	JP_Sales
0	3.204302	2.364530
1	2.603881	0.014644
2	-0.398227	3.975880
3	2.549297	4.580136
4	2.549297	-0.052495
...	...	...
12369	-0.616562	-0.321054
12370	-0.616562	-0.321054
12371	-0.671146	-0.321054
12372	-0.671146	-0.321054
12373	-0.616562	-0.321054

12374 rows × 2 columns

In [75]:
```
X=pd.concat([encoder_cols,scaled_cols],axis=1)
Y=df_new['EU_Sales']
```

In [76]:
```
X_train,X_test, Y_train, Y_test = train_test_split(X,Y, test_size=0.2,random_s
```

In [77]:
```
model=LinearRegression()
model.fit(X_train,Y_train)
y_pred=model.predict(X_test)
```

In [78]:
```
print(model.intercept_) #y-intercept of the model
```

```
-0.3185384874012706
```

In [79]:
```
print(model.coef_)
```

```
[ 0.05929566  0.00150491 -0.01136081 ...  0.35046864  0.0945518
  0.02285433]
```

In [80]:
```
mae = mean_absolute_error(Y_test,y_pred)
mse= mean_squared_error(Y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(Y_test, y_pred)
print('Mean Absolute Error',mae)
print('Mean Squared Error',mse)
print('Root Mean Absolute Error',rmse)
print('R2 Score',r2)
```

```
Mean Absolute Error 892139634.3411064
Mean Squared Error 1.9429108556130783e+19
Root Mean Absolute Error 4407846249.148305
R2 Score -6.825625078717418e+20
```

In [81]:
```
adjusted_r2=1-((1-0.43205)*(822-1)/(822-11-1))
print('adjusted r2 is :',adjusted_r2)
```

```
adjusted r2 is : 0.42433709876543213
```

In [82]:
```
y_mean=np.mean(Y_test)
SSR = np.sum((y_pred - y_mean) ** 2)
SSR
```

Out[82]:
```
4.8087043676383885e+22
```

In [83]:
```
SST = np.sum((Y_test - y_mean) ** 2)
SST
```

Out[83]:
```
70.45075450505051
```

In [84]:
```
SSE=SST-SSR
SSE
```

Out[84]:
```
-4.8087043676383885e+22
```

In [85]:
```
b=pd.DataFrame({"Actual":Y_test,"Predicted":y_pred})
b
```

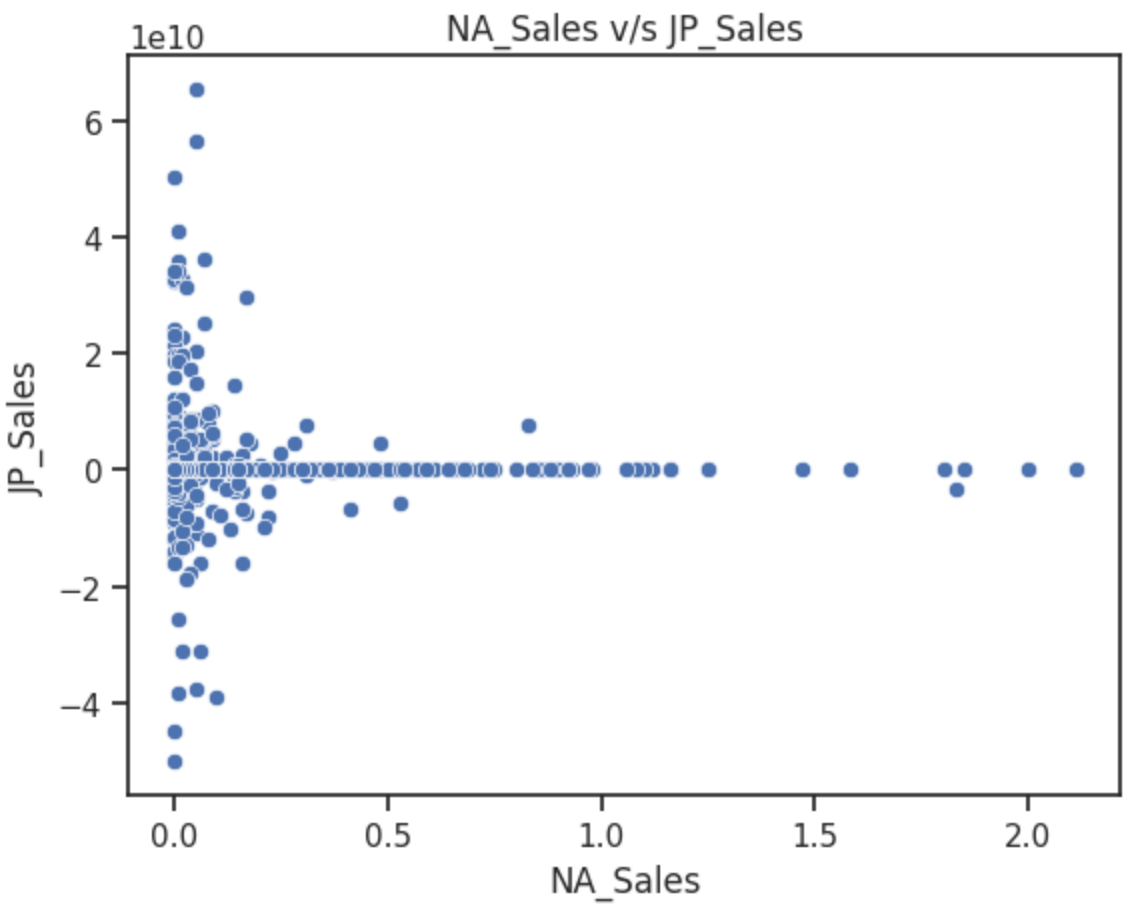
Out[85]:

	Actual	Predicted
6751	0.03	-0.003724
8191	0.00	0.045334
11105	0.01	-0.006276
5959	0.01	0.187873
7634	0.05	0.014078
...	...	...
11449	0.00	-0.030127
15640	0.01	-0.071779
11672	0.00	0.045034
7499	0.00	0.055656
3623	0.36	0.050495

2475 rows × 2 columns

In [86]:
```
sns.scatterplot(x=Y_test,y=y_pred)
plt.xlabel('NA_Sales')
plt.ylabel('JP_Sales')
plt.title('NA_Sales v/s JP_Sales')
```

Out[86]: Text(0.5, 1.0, 'NA_Sales v/s JP_Sales')



In [87]:
```
from sklearn.model_selection import cross_val_score,GridSearchCV
from sklearn.linear_model import Ridge,Lasso
```

In [89]:
```
lr_model=LinearRegression()
lr_scores=cross_val_score(lr_model,X_train,Y_train,cv=5)
```

In [90]:
```
lasso_model=Lasso(alpha=1.0)
lassso_scores=cross_val_score(lasso_model,X_train,Y_train,cv=5)
```

In [91]:
```
ridge_model=Ridge(alpha=1.0)
ridge_scores=cross_val_score(ridge_model,X_train,Y_train,cv=5)
```

In [92]:
```
lr_model.fit(X_train,Y_train)
lr_prediction =lr_model.predict(X_test)
lr_mae =mean_absolute_error(Y_test,lr_prediction)
lr_mse =mean_squared_error(Y_test,lr_prediction)
lr_rmse = np.sqrt(lr_mse)
lr_r2 = r2_score(Y_test,lr_prediction)
print('Linear mae',lr_mae)
print('Linear mse',lr_mse)
print('Linear rmse',lr_rmse)
print('Linear r2',lr_r2)
```

```
Linear mae 892139634.3411064
Linear mse 1.9429108556130783e+19
Linear rmse 4407846249.148305
Linear r2 -6.825625078717418e+20
```

In [93]:
```
lasso_model.fit(X_train,Y_train)
lasso_prediction =lasso_model.predict(X_test)
lasso_mae =mean_absolute_error(Y_test,lasso_prediction)
lasso_mse =mean_squared_error(Y_test,lasso_prediction)
lasso_rmse = np.sqrt(lasso_mse)
lasso_r2 = r2_score(Y_test,lr_prediction)
print('Lasso mae',lasso_mae)
print('Lasso mse',lasso_mse)
print('Lasso rmse',lasso_rmse)
print('Lasso r2',lasso_r2)
```

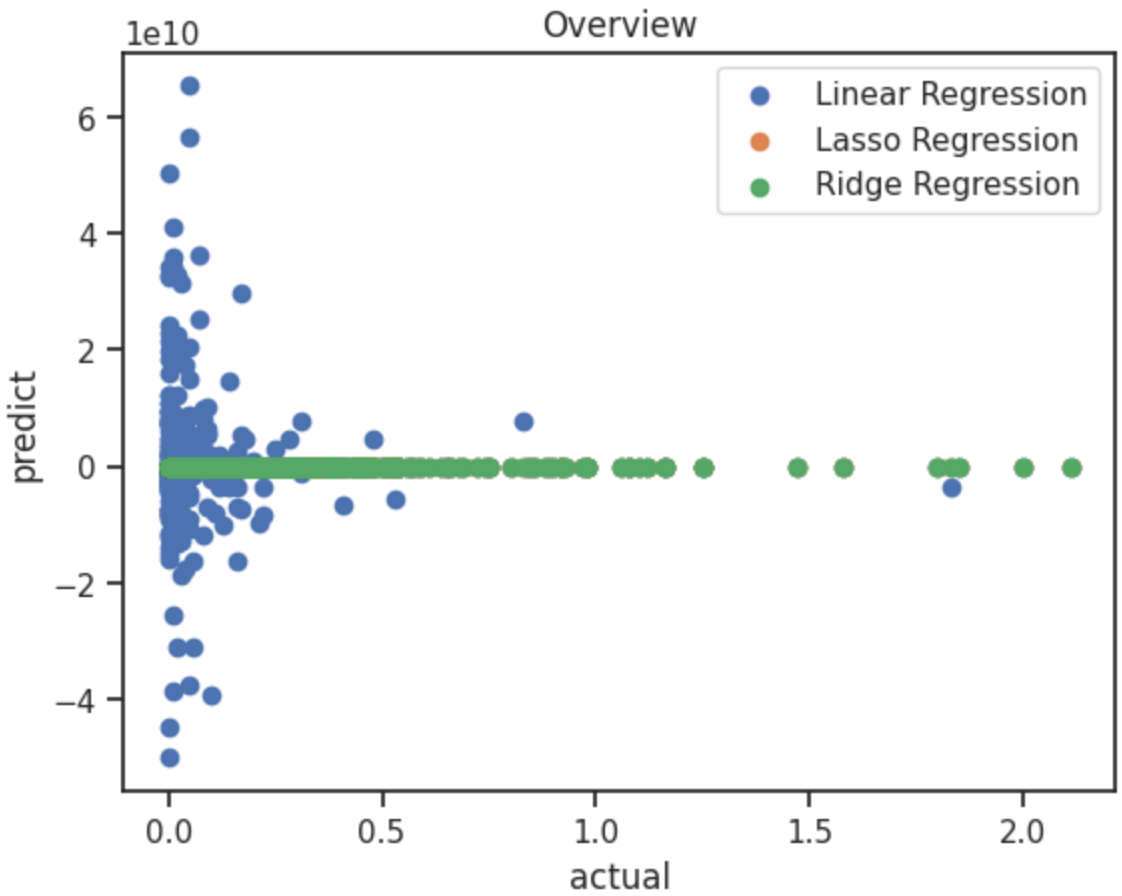
```
Lasso mae 0.08716452289334399
Lasso mse 0.028496984535015295
Lasso rmse 0.16881049888859193
Lasso r2 -6.825625078717418e+20
```

In [94]:
```
ridge_model.fit(X_train,Y_train)
ridge_prediction =ridge_model.predict(X_test)
ridge_mae =mean_absolute_error(Y_test,ridge_prediction)
ridge_mse =mean_squared_error(Y_test,ridge_prediction)
ridge_rmse = np.sqrt(ridge_mse)
ridge_r2 = r2_score(Y_test,ridge_prediction)
print('ridge mae',ridge_mae)
print('ridge mse',ridge_mse)
print('ridge rmse',ridge_rmse)
print('ridge r2',ridge_r2)
```

```
ridge mae 0.05844579083154421
ridge mse 0.01850716346288594
ridge rmse 0.13604103595197276
ridge r2 0.3498262737930081
```

In [95]:
```
plt.scatter(Y_test,lr_prediction,alpha=1.0,label='Linear Regression')
plt.scatter(Y_test,lasso_prediction,alpha=1.0,label='Lasso Regression')
plt.scatter(Y_test,ridge_prediction,alpha=1.0,label='Ridge Regression')
plt.xlabel('actual')
plt.ylabel('predict')
plt.title('Overview')
plt.legend()
```

Out[95]:
```
<matplotlib.legend.Legend at 0x7d0b6000b460>
```



In [96]:
```
from sklearn.linear_model import HuberRegressor
X_scaled = scaler.fit_transform(X_test)
huber = HuberRegressor(epsilon=1.35)
huber.fit(X_scaled, Y_test)
huber_prediction = huber.predict(X_scaled)
huber_mae =mean_absolute_error(Y_test,huber_prediction)
huber_mse =mean_squared_error(Y_test,huber_prediction)
huber_rmse = np.sqrt(huber_mse)
huber_r2 = r2_score(Y_test,huber_prediction)
print('huber mae:',huber_mae)
print('huber mse:',huber_mse)
print('huber rmse:',huber_rmse)
print('huber r2:',huber_r2)
```

```
huber mae: 0.008493172456688364
huber mse: 0.0019215200367379557
huber rmse: 0.0438351461356975
huber r2: 0.9324952284139767
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_huber.py:342: C
onvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sciki
t-learn.org/stable/modules/preprocessing.html)
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

In [99]:
```
from sklearn.linear_model import TheilSenRegressor

# Create a Theil-Sen estimator model
theil_sen = TheilSenRegressor()

# Fit the model to the data
theil_sen.fit(X_test, Y_test)

# Get the Theil-Sen estimate of the coefficients
theil_sen_estimate_intercept = theil_sen.intercept_
theil_sen_estimate_coefficient = theil_sen.coef_[0]
print("Theil-Sen Estimate Intercept:", theil_sen_estimate_intercept)
print("Theil-Sen Estimate Coefficient:", theil_sen_estimate_coefficient)

ts_prediction = theil_sen.predict(X_test)
ts_mae =mean_absolute_error(Y_test,ts_prediction)
ts_mse =mean_squared_error(Y_test,ts_prediction)
ts_rmse = np.sqrt(ts_mse)
ts_r2 = r2_score(Y_test,ts_prediction)
print('ts mae:',ts_mae)
print('ts mse:',ts_mse)
print('ts rmse:',ts_rmse)
print('ts r2:',ts_r2)
```

```
Theil-Sen Estimate Intercept: 27913664473.389503
Theil-Sen Estimate Coefficient: -1015315925584.8302
ts mae: 0.007953445773654514
ts mse: 0.0006315888112754551
ts rmse: 0.025131430744696075
ts r2: 0.9778117023885856
```

In [ ]: