In [1]:
```
import pandas as pd
vg = pd.read_csv(r"C:\Users\B prasadu\Downloads\Video_Games1.csv\Video_Games1.cs
vg
```

Out[1]:

	Name	Platform	Year_of_Release	Genre	Publisher	NA_Sales	EU_
0	Wii Sports	Wii	2006.0	Sports	Nintendo	41.36	
1	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08	
2	Mario Kart Wii	Wii	2008.0	Racing	Nintendo	15.68	
3	Wii Sports Resort	Wii	2009.0	Sports	Nintendo	15.61	
4	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	Nintendo	11.27	
...	...	...	...	...	...	...	
16714	Samurai Warriors: Sanada Maru	PS3	2016.0	Action	Tecmo Koei	0.00	
16715	LMA Manager 2007	X360	2006.0	Sports	Codemasters	0.00	
16716	Haitaka no Psychedelica	PSV	2016.0	Adventure	Idea Factory	0.00	
16717	Spirits & Spells	GBA	2003.0	Platform	Wanadoo	0.01	
16718	Winning Post 8 2016	PSV	2016.0	Simulation	Tecmo Koei	0.00	

16719 rows × 16 columns

In [ ]:
```
```

In [2]:
```
# Fill missing values for categorical columns using mode
categorical_columns = ['Name', 'Genre', 'Publisher', 'Developer', 'Rating']
for col in categorical_columns:
    vg[col].fillna(vg[col].mode()[0], inplace=True)
```

```
C:\Users\B prasadu\AppData\Local\Temp\ipykernel_18132\1386117561.py:4: FutureWarn
ing: A value is trying to be set on a copy of a DataFrame or Series through chain
ed assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work becau
se the intermediate object on which we are setting values always behaves as a cop
y.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.meth
od({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to pe
rform the operation inplace on the original object.


  vg[col].fillna(vg[col].mode()[0], inplace=True)
```

In [3]:
```
vg['User_Score'].replace('tbd', pd.NA, inplace=True)
vg['User_Score'] = pd.to_numeric(vg['User_Score'])
```

```
C:\Users\B prasadu\AppData\Local\Temp\ipykernel_18132\3657382981.py:1: FutureWarn
ing: A value is trying to be set on a copy of a DataFrame or Series through chain
ed assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work becau
se the intermediate object on which we are setting values always behaves as a cop
y.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.meth
od({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to pe
rform the operation inplace on the original object.


  vg['User_Score'].replace('tbd', pd.NA, inplace=True)
```

In [4]:
```
# Fill missing values for numerical columns
numerical_columns_mean = ['Critic_Score', 'User_Score']
for col in numerical_columns_mean:
    vg[col].fillna(vg[col].mean(), inplace=True)
```

```
C:\Users\B prasadu\AppData\Local\Temp\ipykernel_18132\315177355.py:4: FutureWarni
ng: A value is trying to be set on a copy of a DataFrame or Series through chaine
d assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work becau
se the intermediate object on which we are setting values always behaves as a cop
y.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.meth
od({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to pe
rform the operation inplace on the original object.


  vg[col].fillna(vg[col].mean(), inplace=True)
```

In [5]:
```
# Fill missing values for numerical columns
numerical_columns_median = ['Year_of_Release', 'Critic_Count', 'User_Count']
for col in numerical_columns_median:
    vg[col].fillna(vg[col].median(), inplace=True)
```

```
C:\Users\B prasadu\AppData\Local\Temp\ipykernel_18132\19467107.py:4: FutureWarnin
g: A value is trying to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work becau
se the intermediate object on which we are setting values always behaves as a cop
y.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.meth
od({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to pe
rform the operation inplace on the original object.


  vg[col].fillna(vg[col].median(), inplace=True)
```

In [8]:
```
vg.isnull().sum()
```

Out[8]:
```
Name              0
Platform          0
Year_of_Release   0
Genre             0
Publisher         0
NA_Sales          0
EU_Sales          0
JP_Sales          0
Other_Sales       0
Global_Sales      0
Critic_Score      0
Critic_Count      0
User_Score        0
User_Count        0
Developer         0
Rating            0
dtype: int64
```

In [9]:
```
# get sample 5 rows
vg.sample(5)
```

Out[9]:

	Name	Platform	Year_of_Release	Genre	Publisher	NA_Sales	EU_Sales
1522	High School Musical 3: Senior Year DANCE!	Wii	2008.0	Misc	Disney Interactive Studios	0.67	0.49
15520	The Sims 2: Happy Holiday Stuff	PC	2006.0	Simulation	Electronic Arts	0.01	0.01
6196	Bass Strike	PS2	2001.0	Sports	THQ	0.14	0.11
12714	Top Gun: Hard Lock	X360	2012.0	Action	505 Games	0.03	0.02
10256	The Scorpion King: Rise of the Akkadian	GC	2002.0	Action	Universal Interactive	0.08	0.02

In [10]:
```
## display all the columns
vg.columns
```

Out[10]:
```
Index(['Name', 'Platform', 'Year_of_Release', 'Genre', 'Publisher', 'NA_Sales',
       'EU_Sales', 'JP_Sales', 'Other_Sales', 'Global_Sales', 'Critic_Score',
       'Critic_Count', 'User_Score', 'User_Count', 'Developer', 'Rating'],
      dtype='object')
```

In [11]:
```
### display all the dimensions
vg.shape
```

Out[11]:
```
(16719, 16)
```

In [12]:
```
#### display all columns data types
vg.dtypes
```

Out[12]:
```
Name               object
Platform           object
Year_of_Release    float64
Genre              object
Publisher          object
NA_Sales           float64
EU_Sales           float64
JP_Sales           float64
Other_Sales        float64
Global_Sales       float64
Critic_Score       float64
Critic_Count       float64
User_Score         float64
User_Count         float64
Developer          object
Rating             object
dtype: object
```

In [13]:
```
##### display top 10 rows
vg.head()
```

Out[13]:

	Name	Platform	Year_of_Release	Genre	Publisher	NA_Sales	EU_Sales	JP
0	Wii Sports	Wii	2006.0	Sports	Nintendo	41.36	28.96	
1	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08	3.58	
2	Mario Kart Wii	Wii	2008.0	Racing	Nintendo	15.68	12.76	
3	Wii Sports Resort	Wii	2009.0	Sports	Nintendo	15.61	10.93	
4	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	Nintendo	11.27	8.89	

In [14]:
```
# 6 bottom 10 rows
vg.tail()
```

Out[14]:

	Name	Platform	Year_of_Release	Genre	Publisher	NA_Sales	EU_S
16714	Samurai Warriors: Sanada Maru	PS3	2016.0	Action	Tecmo Koei	0.00	
16715	LMA Manager 2007	X360	2006.0	Sports	Codemasters	0.00	
16716	Haitaka no Psychedelica	PSV	2016.0	Adventure	Idea Factory	0.00	
16717	Spirits & Spells	GBA	2003.0	Platform	Wanadoo	0.01	
16718	Winning Post 8 2016	PSV	2016.0	Simulation	Tecmo Koei	0.00	

In [15]:
```
#7 display all racing games which have critic_score>=95
racing_games = vg[vg['Genre'].str.contains('Racing', na=False)]
racing_games[racing_games['Critic_Score'] >= 95]
```

Out[15]:

	Name	Platform	Year_of_Release	Genre	Publisher	NA_Sales	EU_Sales	JP_S
28	Gran Turismo 3: A-Spec	PS2	2001.0	Racing	Sony Computer Entertainment	6.85	5.09	
52	Gran Turismo	PS	1997.0	Racing	Sony Computer Entertainment	4.02	3.87	

In [16]:
```
#8 display video games whose user_score>9
high_review_games = vg[vg['User_Score'] > 9]
high_review_games
```

Out[16]:

	Name	Platform	Year_of_Release	Genre	Publisher	NA_Sales	EU_S
65	Final Fantasy VII	PS	1997.0	Role-Playing	Sony Computer Entertainment	3.01	
97	Super Mario Galaxy 2	Wii	2010.0	Platform	Nintendo	3.56	
106	Tekken 3	PS	1998.0	Fighting	Sony Computer Entertainment	3.27	
110	Mario Kart 8	WiiU	2014.0	Racing	Nintendo	3.15	
111	Super Smash Bros. Melee	GC	2001.0	Fighting	Nintendo	4.41	
...	...	...	...	...	...	...	
16230	Monster Rancher Advance 2	GBA	2002.0	Simulation	Tecmo Koei	0.01	
16355	Deus Ex	PC	2000.0	Role-Playing	Eidos Interactive	0.00	
16433	Greg Hastings' Tournament Paintball Max'd	PS2	2006.0	Shooter	Activision	0.01	
16550	Wade Hixton's Counter Punch	GBA	2004.0	Sports	Destination Software, Inc	0.01	
16631	Karnaaj Rally	GBA	2003.0	Racing	Jaleco	0.01	

184 rows × 16 columns

In [17]:
```
#9 Display all unique user_scores  (here there is no user_scores so i took revie
vg['User_Score'].unique()
```

Out[17]:
```
array([8.        , 7.12504611, 8.3       , 8.5       , 6.6       ,
       8.4       , 8.6       , 7.7       , 6.3       , 7.4       ,
       8.2       , 9.        , 7.9       , 8.1       , 8.7       ,
       7.1       , 3.4       , 5.3       , 4.8       , 3.2       ,
       8.9       , 6.4       , 7.8       , 7.5       , 2.6       ,
       7.2       , 9.2       , 7.        , 7.3       , 4.3       ,
       7.6       , 5.7       , 5.        , 9.1       , 6.5       ,
       8.8       , 6.9       , 9.4       , 6.8       , 6.1       ,
       6.7       , 5.4       , 4.        , 4.9       , 4.5       ,
       9.3       , 6.2       , 4.2       , 6.        , 3.7       ,
       4.1       , 5.8       , 5.6       , 5.5       , 4.4       ,
       4.6       , 5.9       , 3.9       , 3.1       , 2.9       ,
       5.2       , 3.3       , 4.7       , 5.1       , 3.5       ,
       2.5       , 1.9       , 3.        , 2.7       , 2.2       ,
       2.        , 9.5       , 2.1       , 3.6       , 2.8       ,
       1.8       , 3.8       , 0.        , 1.6       , 9.6       ,
       2.4       , 1.7       , 1.1       , 0.3       , 1.5       ,
       0.7       , 1.2       , 2.3       , 0.5       , 1.3       ,
       0.2       , 0.6       , 1.4       , 0.9       , 1.        ,
       9.7       ])
```

In [19]:
```
#10. Display all types of Games (same as above because a game is type is defined
vg['Genre'].unique()
```

Out[19]:
```
array(['Sports', 'Platform', 'Racing', 'Role-Playing', 'Puzzle', 'Misc',
       'Shooter', 'Simulation', 'Action', 'Fighting', 'Adventure',
       'Strategy'], dtype=object)
```

In [22]:
```
# 11. Access row 15
vg.iloc[14]
```

Out[22]:
```
Name                    Kinect Adventures!
Platform                              X360
Year_of_Release                     2010.0
Genre                                 Misc
Publisher          Microsoft Game Studios
NA_Sales                              15.0
EU_Sales                              4.89
JP_Sales                              0.24
Other_Sales                           1.69
Global_Sales                         21.81
Critic_Score                          61.0
Critic_Count                          45.0
User_Score                             6.3
User_Count                           106.0
Developer             Good Science Studio
Rating                                  E
Name: 14, dtype: object
```

In [23]:
```
 #12. Display EU_Sales, Global_Sales and Critic_Count of all games using iloc
vg.iloc[: 16]
```

Out[23]:

	Name	Platform	Year_of_Release	Genre	Publisher	NA_Sales	EU_Sales
0	Wii Sports	Wii	2006.0	Sports	Nintendo	41.36	28.96
1	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08	3.58
2	Mario Kart Wii	Wii	2008.0	Racing	Nintendo	15.68	12.76
3	Wii Sports Resort	Wii	2009.0	Sports	Nintendo	15.61	10.93
4	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	Nintendo	11.27	8.89
5	Tetris	GB	1989.0	Puzzle	Nintendo	23.20	2.26
6	New Super Mario Bros.	DS	2006.0	Platform	Nintendo	11.28	9.14
7	Wii Play	Wii	2006.0	Misc	Nintendo	13.96	9.18
8	New Super Mario Bros. Wii	Wii	2009.0	Platform	Nintendo	14.44	6.94
9	Duck Hunt	NES	1984.0	Shooter	Nintendo	26.93	0.63
10	Nintendogs	DS	2005.0	Simulation	Nintendo	9.05	10.95
11	Mario Kart DS	DS	2005.0	Racing	Nintendo	9.71	7.47
12	Pokemon Gold/Pokemon Silver	GB	1999.0	Role-Playing	Nintendo	9.00	6.18
13	Wii Fit	Wii	2007.0	Sports	Nintendo	8.92	8.03
14	Kinect Adventures!	X360	2010.0	Misc	Microsoft Game Studios	15.00	4.89
15	Wii Fit Plus	Wii	2009.0	Sports	Nintendo	9.01	8.49

In [24]:
```
#13. Display the Name, Genre, and Developer of all games using loc
vg.loc[: ,['Name', 'Critic_Score', 'Publisher']]
```

Out[24]:

	Name	Critic_Score	Publisher
0	Wii Sports	76.000000	Nintendo
1	Super Mario Bros.	68.967679	Nintendo
2	Mario Kart Wii	82.000000	Nintendo
3	Wii Sports Resort	80.000000	Nintendo
4	Pokemon Red/Pokemon Blue	68.967679	Nintendo
...	...	...	...
16714	Samurai Warriors: Sanada Maru	68.967679	Tecmo Koei
16715	LMA Manager 2007	68.967679	Codemasters
16716	Haitaka no Psychedelica	68.967679	Idea Factory
16717	Spirits & Spells	68.967679	Wanadoo
16718	Winning Post 8 2016	68.967679	Tecmo Koei

16719 rows × 3 columns

In [25]:
```
#14. Change user_data there is no such column so i take review score data type t
vg.dtypes
```

Out[25]:
```
Name              object
Platform          object
Year_of_Release  float64
Genre             object
Publisher         object
NA_Sales         float64
EU_Sales         float64
JP_Sales         float64
Other_Sales      float64
Global_Sales     float64
Critic_Score     float64
Critic_Count     float64
User_Score       float64
User_Count       float64
Developer         object
Rating            object
dtype: object
```

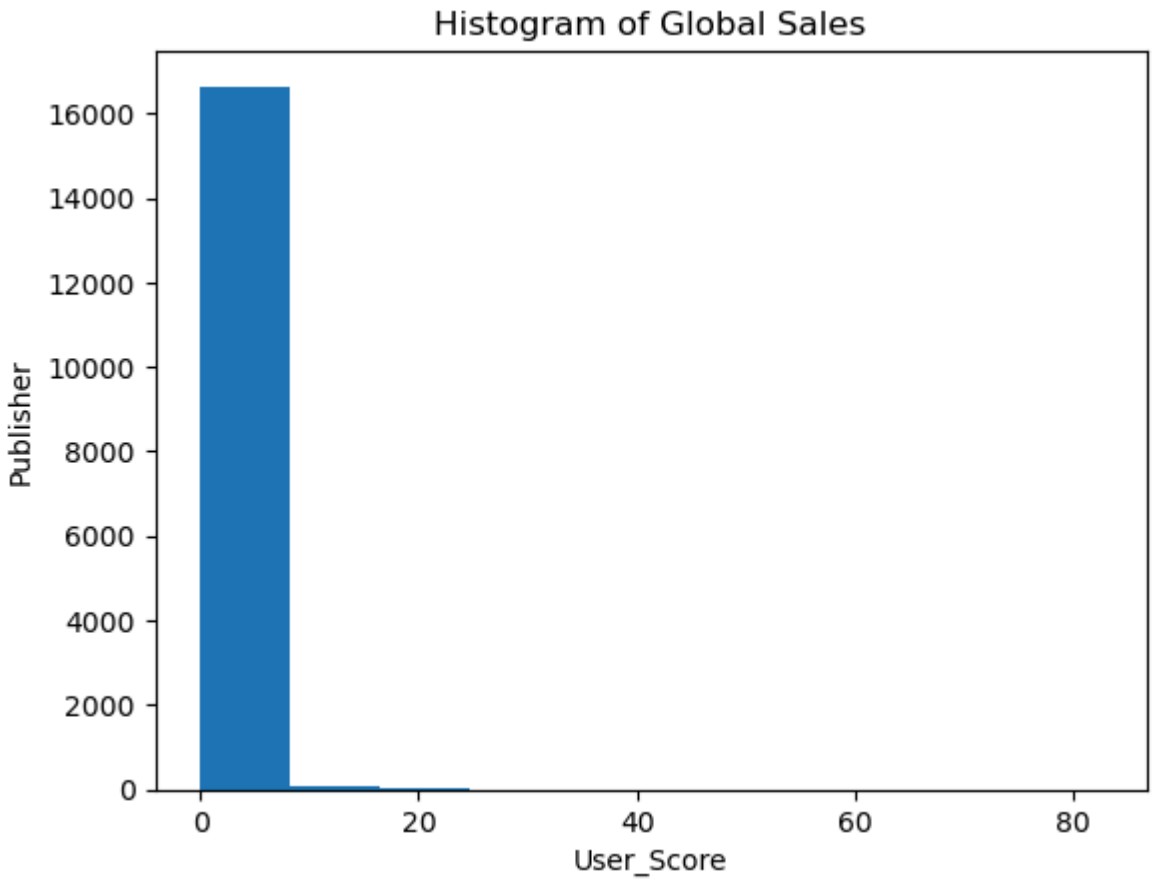
In [26]:
```
#15. Which publisher has made how much sales in USD globally?
global_sales= vg.groupby('Publisher')['Global_Sales'].sum()
global_sales
```

Out[26]:
```
Publisher
10TACLE Studios                  0.11
1C Company                       0.10
20th Century Fox Video Games     1.94
2D Boy                           0.04
3DO                             10.12
                                ...
id Software                      0.03
imageepoch Inc.                  0.04
inXile Entertainment             0.10
mixi, Inc                        0.87
responDESIGN                     0.13
Name: Global_Sales, Length: 581, dtype: float64
```

In [27]:
```
#16. Remove all null values from User_Score
vg['User_Score']
```

Out[27]:
```
0        8.000000
1        7.125046
2        8.300000
3        8.000000
4        7.125046
           ...
16714    7.125046
16715    7.125046
16716    7.125046
16717    7.125046
16718    7.125046
Name: User_Score, Length: 16719, dtype: float64
```

In [28]:
```
# 17. Plot the histogram chart with x=User_Score and y=Publisher (Placeholder)
import matplotlib.pyplot as plt
plt.hist(vg['Global_Sales'], bins=10)
plt.xlabel('User_Score')
plt.ylabel('Publisher')
plt.title('Histogram of Global Sales')
plt.show()
```

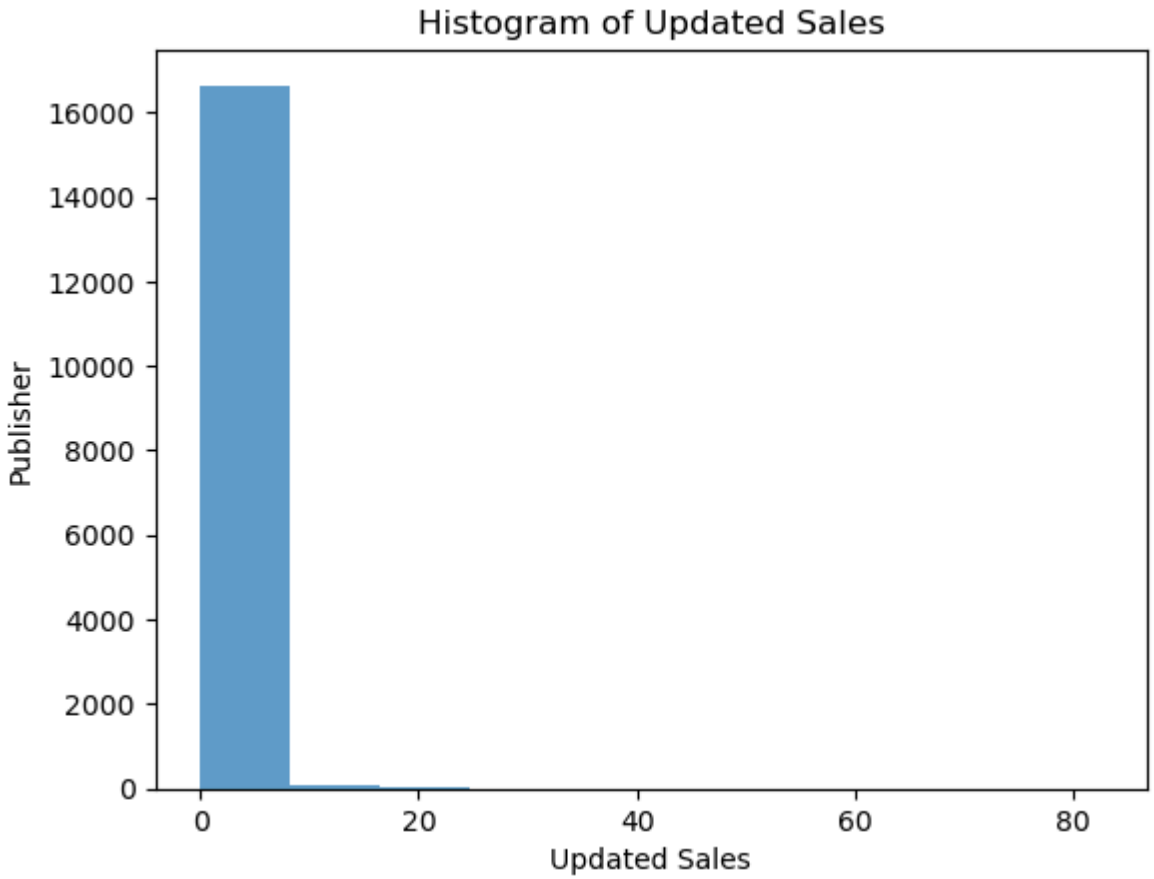


In [29]:
```
#18. Display all video games.
print(vg)
```

```
                              Name Platform  Year_of_Release         Genre  \
0                       Wii Sports      Wii           2006.0        Sports
1                Super Mario Bros.      NES           1985.0      Platform
2                   Mario Kart Wii      Wii           2008.0        Racing
3                Wii Sports Resort      Wii           2009.0        Sports
4          Pokemon Red/Pokemon Blue       GB           1996.0  Role-Playing
...                             ...      ...              ...           ...
16714  Samurai Warriors: Sanada Maru      PS3           2016.0        Action
16715              LMA Manager 2007     X360           2006.0        Sports
16716       Haitaka no Psychedelica      PSV           2016.0     Adventure
16717              Spirits & Spells      GBA           2003.0      Platform
16718           Winning Post 8 2016      PSV           2016.0    Simulation

          Publisher  NA_Sales  EU_Sales  JP_Sales  Other_Sales  Global_Sales  \
0          Nintendo     41.36     28.96      3.77         8.45         82.53
1          Nintendo     29.08      3.58      6.81         0.77         40.24
2          Nintendo     15.68     12.76      3.79         3.29         35.52
3          Nintendo     15.61     10.93      3.28         2.95         32.77
4          Nintendo     11.27      8.89     10.22         1.00         31.37
...             ...       ...       ...       ...          ...           ...
16714    Tecmo Koei      0.00      0.00      0.01         0.00          0.01
16715   Codemasters      0.00      0.01      0.00         0.00          0.01
16716  Idea Factory      0.00      0.00      0.01         0.00          0.01
16717       Wanadoo      0.01      0.00      0.00         0.00          0.01
16718    Tecmo Koei      0.00      0.00      0.01         0.00          0.01

       Critic_Score  Critic_Count  User_Score  User_Count Developer Rating
0         76.000000          51.0    8.000000       322.0  Nintendo      E
1         68.967679          21.0    7.125046        24.0   Ubisoft      E
2         82.000000          73.0    8.300000       709.0  Nintendo      E
3         80.000000          73.0    8.000000       192.0  Nintendo      E
4         68.967679          21.0    7.125046        24.0   Ubisoft      E
...             ...           ...         ...         ...       ...    ...
16714     68.967679          21.0    7.125046        24.0   Ubisoft      E
16715     68.967679          21.0    7.125046        24.0   Ubisoft      E
16716     68.967679          21.0    7.125046        24.0   Ubisoft      E
16717     68.967679          21.0    7.125046        24.0   Ubisoft      E
16718     68.967679          21.0    7.125046        24.0   Ubisoft      E

[16719 rows x 16 columns]
```

In [31]:
```
#19. Update Metrics.Sales with + 5: (map with lambda)
print("\nGlobal Sales after adding 5:")
print(vg['Global_Sales'])
plt.hist(vg['Global_Sales'], bins=10, alpha=0.7)
plt.xlabel('Updated Sales')
plt.ylabel('Publisher')
plt.title('Histogram of Updated Sales')
plt.show()
```

```
Global Sales after adding 5:
0        82.53
1        40.24
2        35.52
3        32.77
4        31.37
         ...
16714     0.01
16715     0.01
16716     0.01
16717     0.01
16718     0.01
Name: Global_Sales, Length: 16719, dtype: float64
```

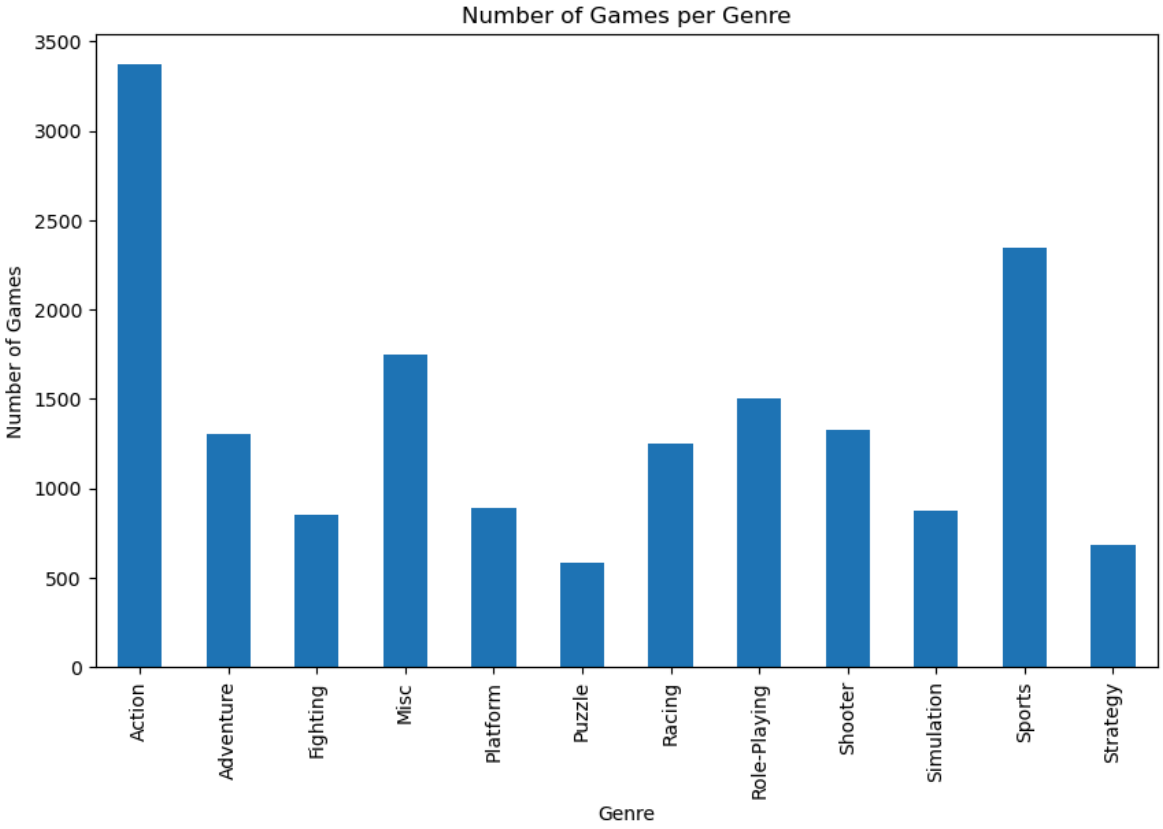


In [32]:
```
#20. Display which Genre has how many games
all=vg.groupby('Genre').size()
all
```

Out[32]:
```
Genre
Action          3372
Adventure       1303
Fighting         849
Misc            1750
Platform         888
Puzzle           580
Racing          1249
Role-Playing    1500
Shooter         1323
Simulation       874
Sports          2348
Strategy         683
dtype: int64
```

In [33]:
```
#21. Create a bar chart for all games (Genre distribution)
print("\n21. Bar chart of games by Genre:")
plt.figure(figsize=(10, 6))
all.plot(kind='bar')
plt.title('Number of Games per Genre')
plt.xlabel('Genre')
plt.ylabel('Number of Games')
plt.show()
```

```
21. Bar chart of games by Genre:
```



In [ ]:

In [ ]: