In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
from matplotlib.ticker import FuncFormatter
import plotly.express as px
from textblob import TextBlob
from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
```

Prescriptive Data Analysis

Objective

In the rapidly evolving landscape of the gaming industry, prescriptive data analysis serves as a strategic compass for decision-makers. This analysis delves into a comprehensive dataset, encompassing video game titles, platforms, release dates, user reviews, and more. By unraveling the intricate relationship between gaming elements, I aim to guide developers and stakeholders in making informed decisions for future success. From identifying evolving trends to understanding the impact of platform choice on user reception, this report presents a roadmap for prescriptive strategies that align with the dynamic expectations of gamers and the ever-changing gaming market

Step 1 Data Cleaning and Exploration:

The dataset chosen provides a comprehensive view of the gaming world, making it a valuable resource for researchers, game developers, and market analysts. It can be used to analyze shifts in gaming culture and technology, understand the impact of narrative and gameplay on a game's success, and predict trends in the gaming industry. The inclusion of user scores allows for assessing the reception and popularity of each game.

Column Descriptions for Video Games Sales Dataset:

- `Name` : The title of the video game.
- `Platform` : The gaming platform on which the game is released.
- `Year_of_Release` : The year when the game was released.
- `Genre` : The category or genre of the video game.
- `Publisher` : The company responsible for publishing the game.
- `NA_Sales` : Sales figures in North America (in millions).
- `EU_Sales` : Sales figures in Europe (in millions).
- `JP_Sales` : Sales figures in Japan (in millions).

- `Other_Sales` : Sales figures in regions other than North America, Europe, and Japan (in millions).
- `Global_Sales` : Total global sales of the video game (in millions).
- `Critic_score` : An aggregate score compiled by Metacritic staff.
- `Critic_count` : The number of critics considered in determining the Critic_score.
- `User_score` : Score given by Metacritic's subscribers.
- `User_count` : The number of users who provided the User_score.
- `Developer` : The entity responsible for creating the game.
- `Rating` : The ESRB (Entertainment Software Rating Board) rating assigned to the game.

In [2]:
```
video_games = pd.read_csv('Video_Games_Sales_as_at_22_Dec_2016.csv')
video_games
```

Out[2]:

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

16719 rows × 16 columns

Data Cleaning

In [3]:
```
video_games = video_games.drop_duplicates()
video_games
```

Out[3]:

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

16719 rows × 16 columns

We observe that the number of rows went from 16719 to 16512 after duplicate rows were dropped.

In [4]:

```
missing_values = video_games.isnull().sum()
print("Missing Values:\n", missing_values)
```

```
Missing Values:
 Name                  2
Platform              0
Year_of_Release     269
Genre                 2
Publisher            54
NA_Sales              0
EU_Sales              0
JP_Sales              0
Other_Sales           0
Global_Sales          0
Critic_Score       8582
Critic_Count       8582
User_Score         9129
User_Count         9129
Developer          6623
Rating             6769
dtype: int64
```

This helps us in evaluating missing values in further analysis when we use these variables.

In [5]:
```
summary_statistics =video_games.describe()
print("Summary Statistics:\n", summary_statistics)
```

```
Summary Statistics:
        Year_of_Release      NA_Sales      EU_Sales      JP_Sales  \
count     16450.000000  16719.000000  16719.000000  16719.000000
mean       2006.487356      0.263330      0.145025      0.077602
std           5.878995      0.813514      0.503283      0.308818
min        1980.000000      0.000000      0.000000      0.000000
25%        2003.000000      0.000000      0.000000      0.000000
50%        2007.000000      0.080000      0.020000      0.000000
75%        2010.000000      0.240000      0.110000      0.040000
max        2020.000000     41.360000     28.960000     10.220000

        Other_Sales  Global_Sales  Critic_Score  Critic_Count   User_Score  \
count  16719.000000  16719.000000   8137.000000   8137.000000  7590.000000
mean       0.047332      0.533543     68.967679     26.360821     7.125046
std        0.186710      1.547935     13.938165     18.980495     1.500006
min        0.000000      0.010000     13.000000      3.000000     0.000000
25%        0.000000      0.060000     60.000000     12.000000     6.400000
50%        0.010000      0.170000     71.000000     21.000000     7.500000
75%        0.030000      0.470000     79.000000     36.000000     8.200000
max       10.570000     82.530000     98.000000    113.000000     9.700000

         User_Count
count   7590.000000
mean     162.229908
std      561.282326
min        4.000000
25%       10.000000
50%       24.000000
75%       81.000000
max    10665.000000
```

Important Observations: The dataset spans from the year 1980 to 2020, with a median release year of 2007. The majority of games were released between 2003 and 2010. Sales figures (in millions) have a wide range, with varying means and standard deviations. The 75th percentile indicates that most games have relatively low sales, while a few exceptional games have very high sales. Critic scores range from 13 to 98, with an average around 69. The number of critics contributing to scores varies, with an average of around 26. User scores range from 0 to 9.7, with an average of 7.13. The number of users contributing to scores varies, with an average of around 162.

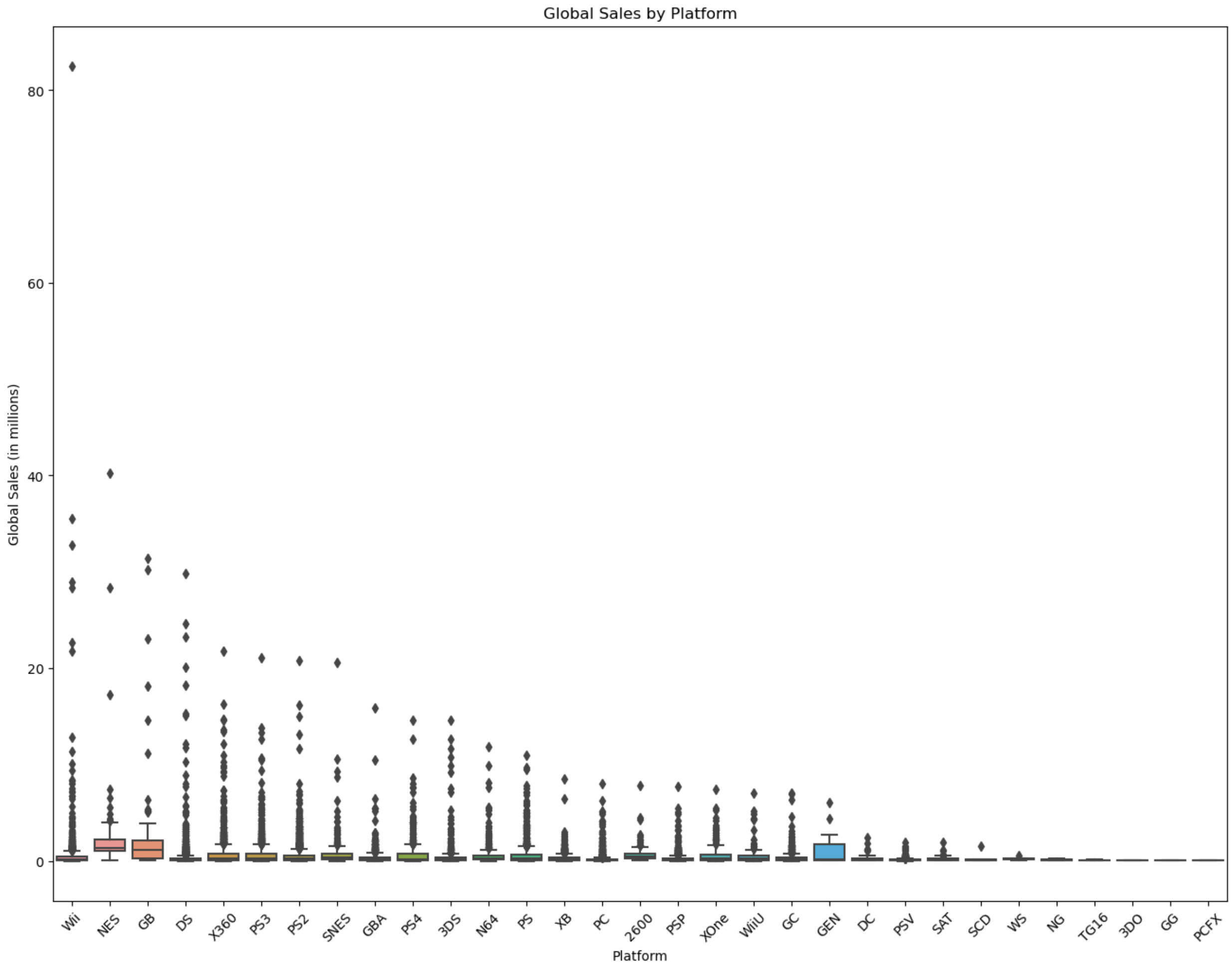
In [6]:
```
video_games['Year_of_Release'].fillna(0, inplace=True)
video_games['Year_of_Release'] = pd.to_datetime(video_games['Year_of_Release']
```

We filled missing values in the `Year_of_Release` column with 0 and then convert the column to a datetime format, ensuring that it contains valid date values.

Exploratory Data Analysis

Boxplot

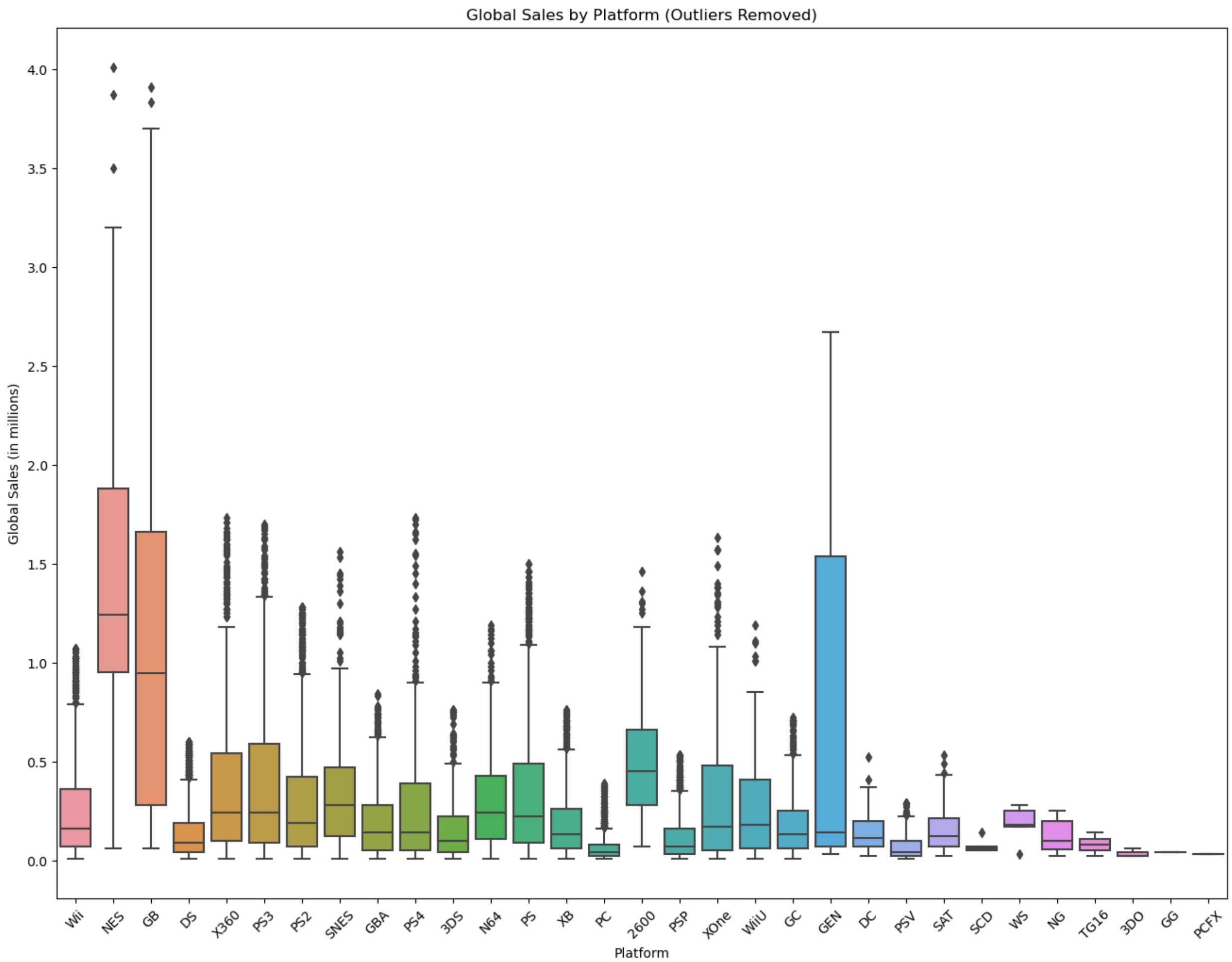
In [7]:
```
plt.figure(figsize=(16, 12))
sns.boxplot(x='Platform', y='Global_Sales', data=video_games)
plt.title('Global Sales by Platform')
plt.xlabel('Platform')
plt.ylabel('Global Sales (in millions)')
plt.xticks(rotation=45)
plt.show()
```



- We can see the `Global_Sales` column has a significantly large number of ouliers (potential data points that are significantly different from the rest of the data).

In [8]:
```
def remove_outliers(data, column):
    Q1 = data[column].quantile(0.25)
    Q3 = data[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return data[(data[column] >= lower_bound) & (data[column] <= upper_bound)]

# Initialize an empty DataFrame to store cleaned data
video_games_cleaned_no_outliers = pd.DataFrame()

# Iterate over platforms and apply IQR method
for platform in video_games['Platform'].unique():
```

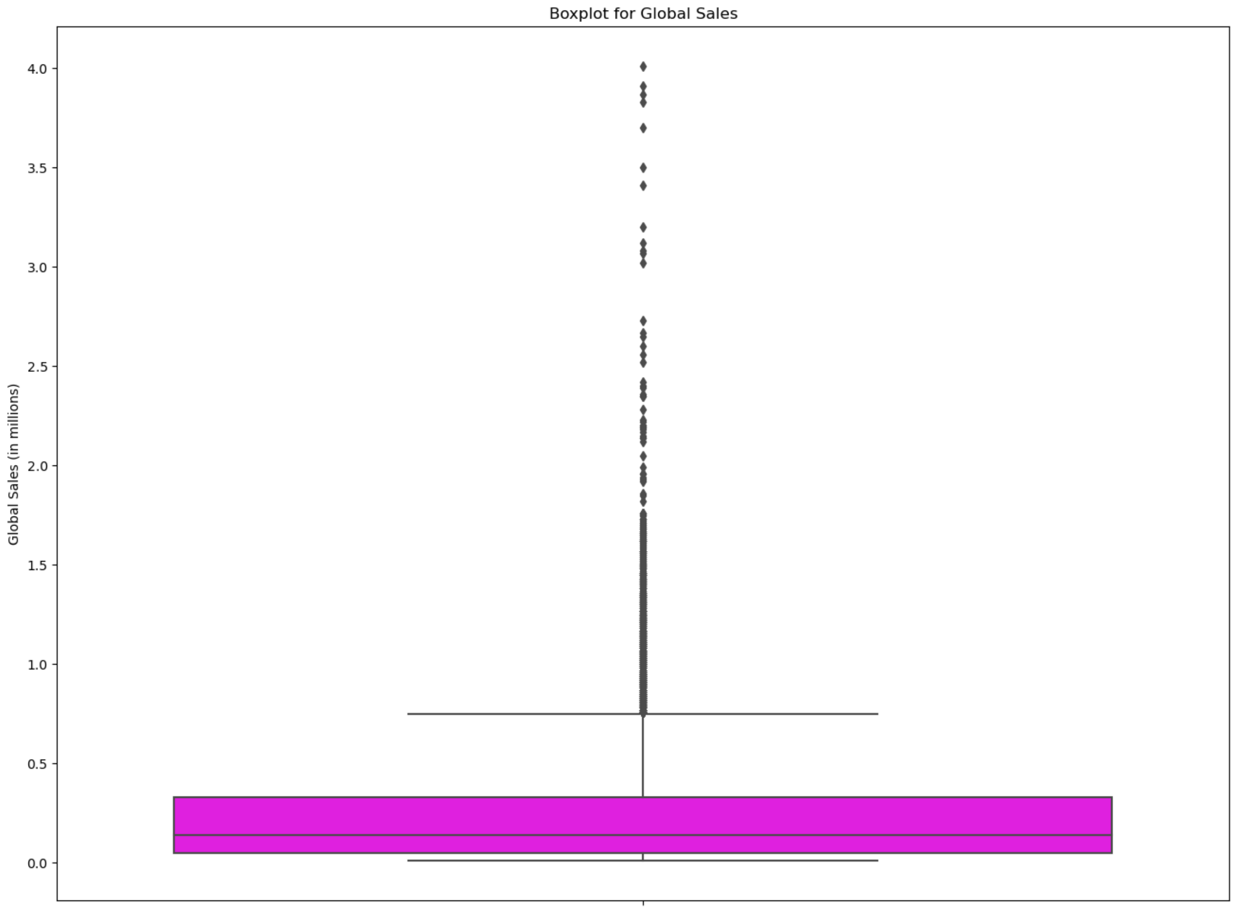
```
    platform_data = video_games[video_games['Platform'] == platform]
    platform_data_no_outliers = remove_outliers(platform_data, 'Global_Sales')
    video_games_cleaned_no_outliers = pd.concat([video_games_cleaned_no_outlie

# Define the boxplot using the cleaned data
plt.figure(figsize=(16, 12))
sns.boxplot(x='Platform', y='Global_Sales', data=video_games_cleaned_no_outlie

# Title and labels
plt.title('Global Sales by Platform (Outliers Removed)')
plt.xlabel('Platform')
plt.ylabel('Global Sales (in millions)')

# Rotate x-axis labels for better visibility
plt.xticks(rotation=45)

# Display the plot
plt.show()
```



- Removing the outliers allows you to examine the distribution of `Global_Sales` for different gaming platforms with a clearer representation of the data.
- NES and GB have the highest median Global Sales as well as the highest value of outliers.

In [9]:
```
plt.figure(figsize=(16, 12))
boxplot_color = 'magenta'
sns.boxplot(y='Global_Sales', data=video_games_cleaned_no_outliers, color=boxp
```

```
plt.title('Boxplot for Global Sales')
plt.ylabel('Global Sales (in millions)')
threshold = 5
outliers = video_games[video_games['Global_Sales'] > threshold]
video_games = video_games[video_games['Global_Sales'] <= threshold]

plt.show()
```



- The boxplot visually represents the distribution of global sales in the dataset, and the code additionally identifies and removes outliers above the threshold of 40 million.
- The median global sale is 0.2 million with the interquartile range starting from 0.1 million to 0.3 million.

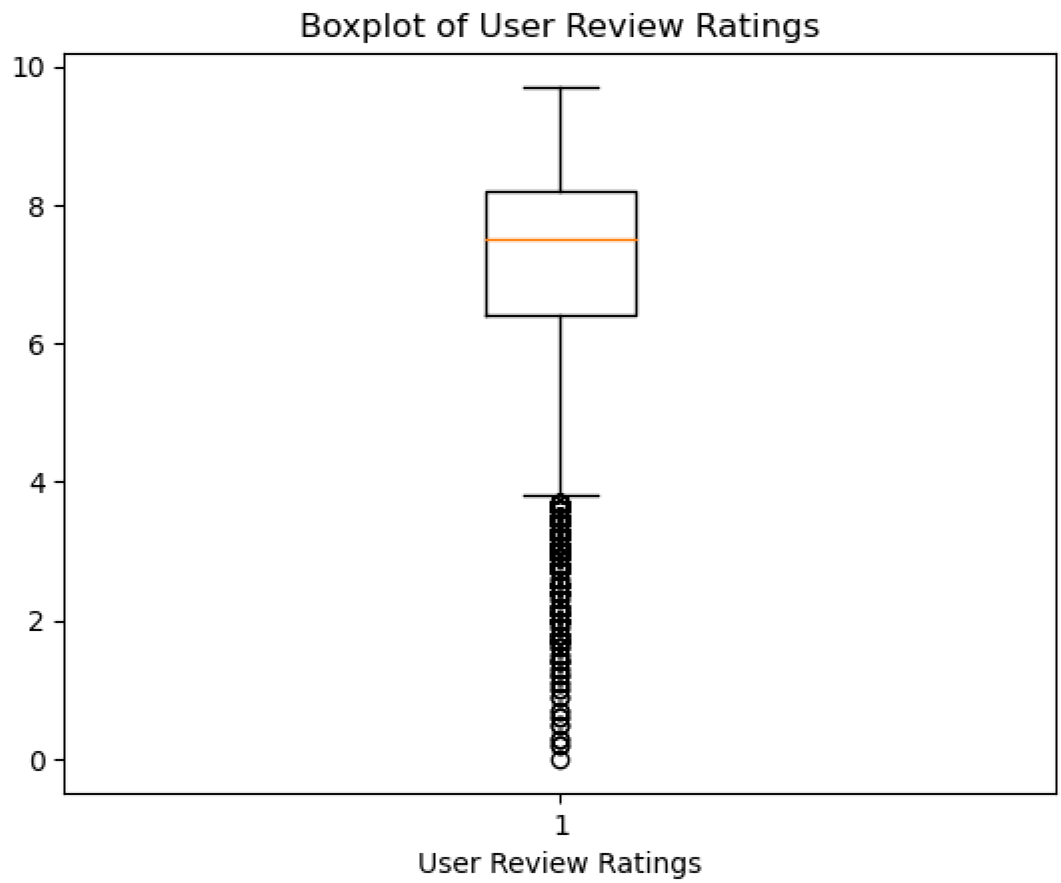
In [10]:
```
video_games_cleaned = video_games.dropna(subset=['User_Score'])
video_games_cleaned = video_games_cleaned[video_games_cleaned['Global_Sales'] <
platform_groups = video_games_cleaned.groupby('Platform')
average_reviews = platform_groups['User_Score'].mean().sort_values(ascending=Fa
average_reviews
```

Out[10]:
```
Platform
DC      8.528571
PS      7.771812
GBA     7.668651
PS2     7.616680
GC      7.590659
XB      7.497432
PSV     7.336364
PSP     7.222010
PC      7.065450
DS      6.999802
WiiU    6.869388
3DS     6.801176
PS4     6.766265
PS3     6.714454
Wii     6.698706
X360    6.671991
XOne    6.520000
Name: User_Score, dtype: float64
```

- We drop the rows with missing `User_Score` values and global sales exceeding the specified threshold, groups the remaining data by gaming platforms, calculates the average user score for each platform, and presents the results in descending order of average user scores.
- DC has the highest averange user score.
- Note: The cleaned dataset will only be used where User Scores are considered.

In [11]:
```python
threshold = 5
video_games = video_games[video_games['Global_Sales'] <= threshold]
```

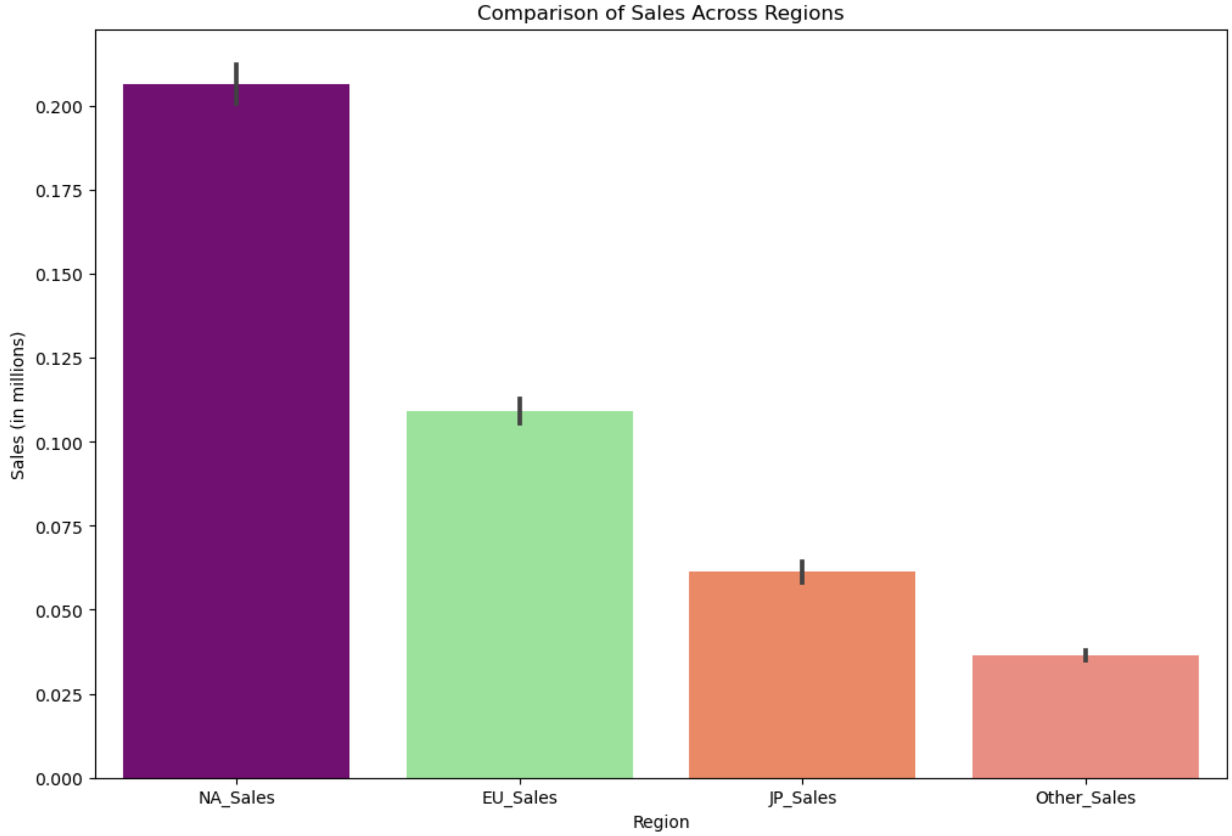
In [12]:
```python
plt.boxplot(video_games_cleaned['User_Score'])


plt.title('Boxplot of User Review Ratings')
plt.xlabel('User Review Ratings')


plt.show()
```



- The boxplot of user review ratings show the central tendency and spread of the user review ratings in the dataset

Bar Plots

In [13]:
```
custom_colors = {'NA_Sales': 'purple', 'EU_Sales': 'lightgreen', 'JP_Sales': '

sales_data = video_games[['NA_Sales', 'EU_Sales', 'JP_Sales', 'Other_Sales']]
plt.figure(figsize=(12, 8))

sns.barplot(x='variable', y='value', data=sales_data.melt(), palette=custom_co

plt.title('Comparison of Sales Across Regions')
plt.xlabel('Region')
plt.ylabel('Sales (in millions)')
plt.show()
```



- The bar plot shows a visual comparison of sales across regions, making it easy to observe and compare the sales performance in North America, Europe, Japan, and other regions.

- DataFrame will only include rows where the `Global_Sales` value is less than or equal to 5 million to handle outliers.

In [14]:
```
platform_sales = video_games.groupby('Platform')['Global_Sales'].sum().sort_va
print(platform_sales)
```

```
Platform
PS2     1100.16
PS3      714.76
X360     712.40
PS       614.49
DS       568.30
Wii      539.78
PSP      276.10
GBA      264.13
XB       243.34
PC       235.74
PS4      230.93
GC       179.03
3DS      170.36
N64      165.16
NES      145.69
XOne     141.30
SNES     139.38
GB        94.33
2600      89.27
WiiU      69.85
PSV       54.12
SAT       33.59
GEN       24.75
DC        15.97
SCD        1.87
NG         1.44
WS         1.42
TG16       0.16
3DO        0.10
GG         0.04
PCFX       0.03
Name: Global_Sales, dtype: float64
```

- We calculated the total global sales for each gaming platform by grouping the data based on the **`Platform`** column, summing up the **`Global_Sales`** values for each group, and then sorting the results in descending order.
- PS2 has the highest total global sales.

In [15]:
```
platform_genre_preferences = video_games.groupby(['Platform', 'Genre']).size()
print(platform_genre_preferences)
```

```
Genre     Action  Adventure  Fighting  Misc  Platform  Puzzle  Racing  \
Platform
2600          61          2         2     5         9      10       6
3DO            0          1         0     0         0       1       0
3DS          188         38        13    54        26      20      10
DC             3         11        12     0         2       0       6
DS           338        238        36   388        89     234      66
GB             6          4         0     7        15      13       2
GBA          167         38        23   110       139      41      63
GC           101         20        41    36        72      13      62
GEN            3          2         5     1         6       0       1
GG             0          0         0     0         1       0       0
N64           37          4        28    18        28      12      56
NES           12          1         4     2        25      13       4
NG             0          0        11     0         0       0       0
PC           170         65         6    24        11      25      61
PCFX           0          0         0     0         0       0       0
PS           152         69       106    76        61      32     143
PS2          344        196       150   221       102      18     212
PS3          368         74        76   124        36       3      91
PS4          143         28        18    20        12       1      19
PSP          220        213        74   106        36      44      63
PSV          150         93        16    24         9       3      11
SAT            3         26        31    15         5       5       8
SCD            0          0         0     2         1       0       1
SNES          12          4        24    17        22      13       8
TG16           0          1         0     0         0       0       0
WS             0          0         0     0         0       0       0
Wii          235         83        41   273        54      55      92
WiiU          64          3         5    22        16       4       2
X360         317         47        65   125        25       7     103
XB           155         26        48    46        49       7     123
XOne          84         14         7    19         5       0      20

Genre     Role-Playing  Shooter  Simulation  Sports  Strategy
Platform
2600                 0       24           1      12         0
3DO                  0        0           1       0         0
3DS                 85        7          29      26        15
DC                   4        3           1      10         0
DS                 195       42         281     147        79
GB                  17        1           5       9         7
GBA                 70       40          18      88        18
GC                  27       48          12     110        11
GEN                  3        1           0       3         1
GG                   0        0           0       0         0
N64                  8       23          10      80         8
NES                 11        6           0      14         0
NG                   0        0           0       1         0
PC                 102      150         118      50       188
PCFX                 1        0           0       0         0
PS                  94       96          60     221        70
PS2                182      158          90     399        71
PS3                117      148          31     211        24
PS4                 51       36           6      43         6
PSP                191       37          28     134        60
PSV                 87        5           4      23         7
SAT                 17       22           7      16        18
SCD                  1        0           0       0         1
SNES                50       10           9      49        15
```

```
TG16          0      1      0      0      0
WS            4      0      0      0      2
Wii          35     65     87    254     25
WiiU          7     10      1      8      3
X360         73    188     40    216     28
XB           23    130     24    170     21
XOne         14     36      4     38      3
```

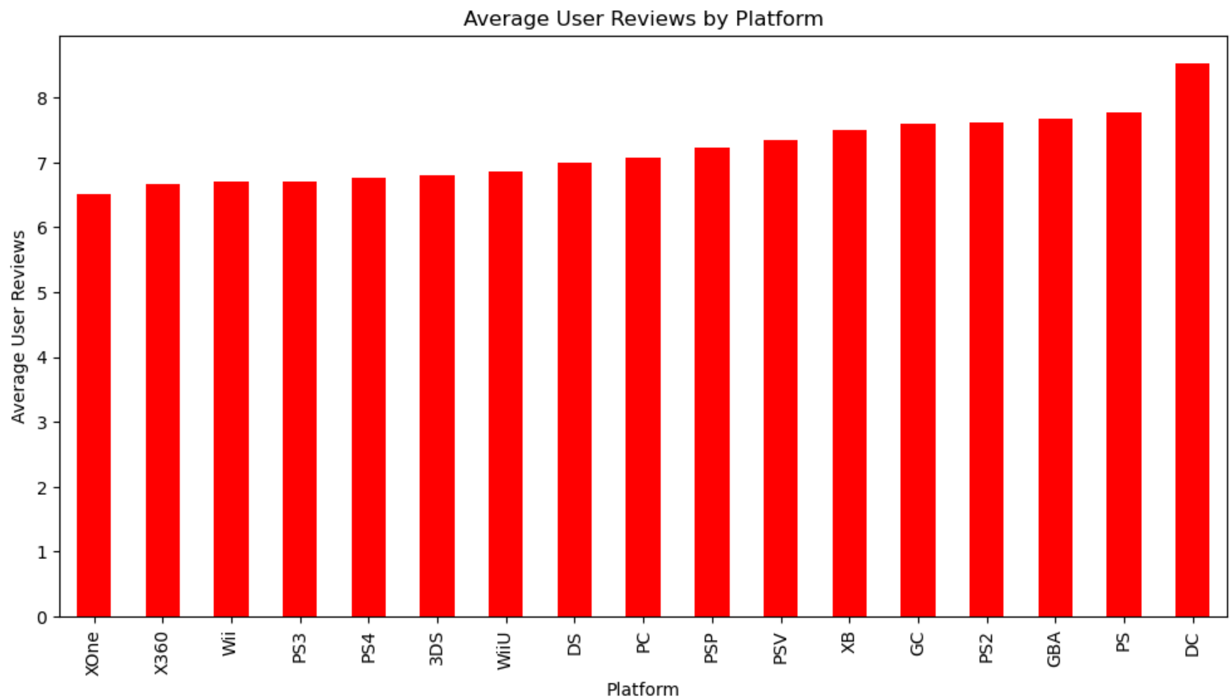
- `platform_genre_preferences` provides an overview of the distribution of game genres across different gaming platforms.

In [16]:
```
platform_recommendations = pd.DataFrame({
    'Average_User_Reviews': average_reviews,
    'Total_Global_Sales': platform_sales
})
print(platform_recommendations)
```

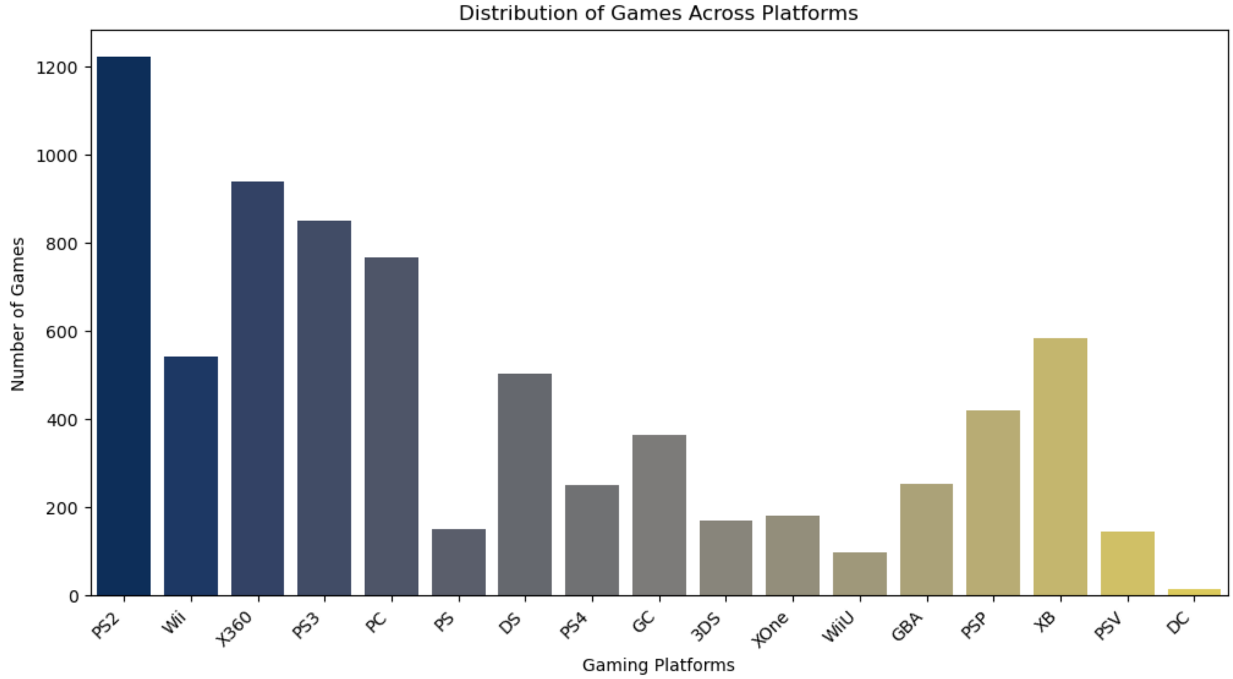
```
          Average_User_Reviews  Total_Global_Sales
Platform
2600                       NaN               89.27
3DO                        NaN                0.10
3DS                   6.801176              170.36
DC                    8.528571               15.97
DS                    6.999802              568.30
GB                         NaN               94.33
GBA                   7.668651              264.13
GC                    7.590659              179.03
GEN                        NaN               24.75
GG                         NaN                0.04
N64                        NaN              165.16
NES                        NaN              145.69
NG                         NaN                1.44
PC                    7.065450              235.74
PCFX                       NaN                0.03
PS                    7.771812              614.49
PS2                   7.616680             1100.16
PS3                   6.714454              714.76
PS4                   6.766265              230.93
PSP                   7.222010              276.10
PSV                   7.336364               54.12
SAT                        NaN               33.59
SCD                        NaN                1.87
SNES                       NaN              139.38
TG16                       NaN                0.16
WS                         NaN                1.42
Wii                   6.698706              539.78
WiiU                  6.869388               69.85
X360                  6.671991              712.40
XB                    7.497432              243.34
XOne                  6.520000              141.30
```

- `platform_recommendations` provides a summary of the gaming platforms, showcasing both the average user reviews and the total global sales for each platform.
- The NaN (Not a Number) values in the `Average User Reviews` column indicate that there might be platforms for which the average user reviews are not available, while the sales data is still presented in the `Total Global Sales` column.

In [17]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))
average_reviews.sort_values().plot(kind='bar', color='red')
plt.title('Average User Reviews by Platform')
plt.xlabel('Platform')
plt.ylabel('Average User Reviews')
plt.show()
```



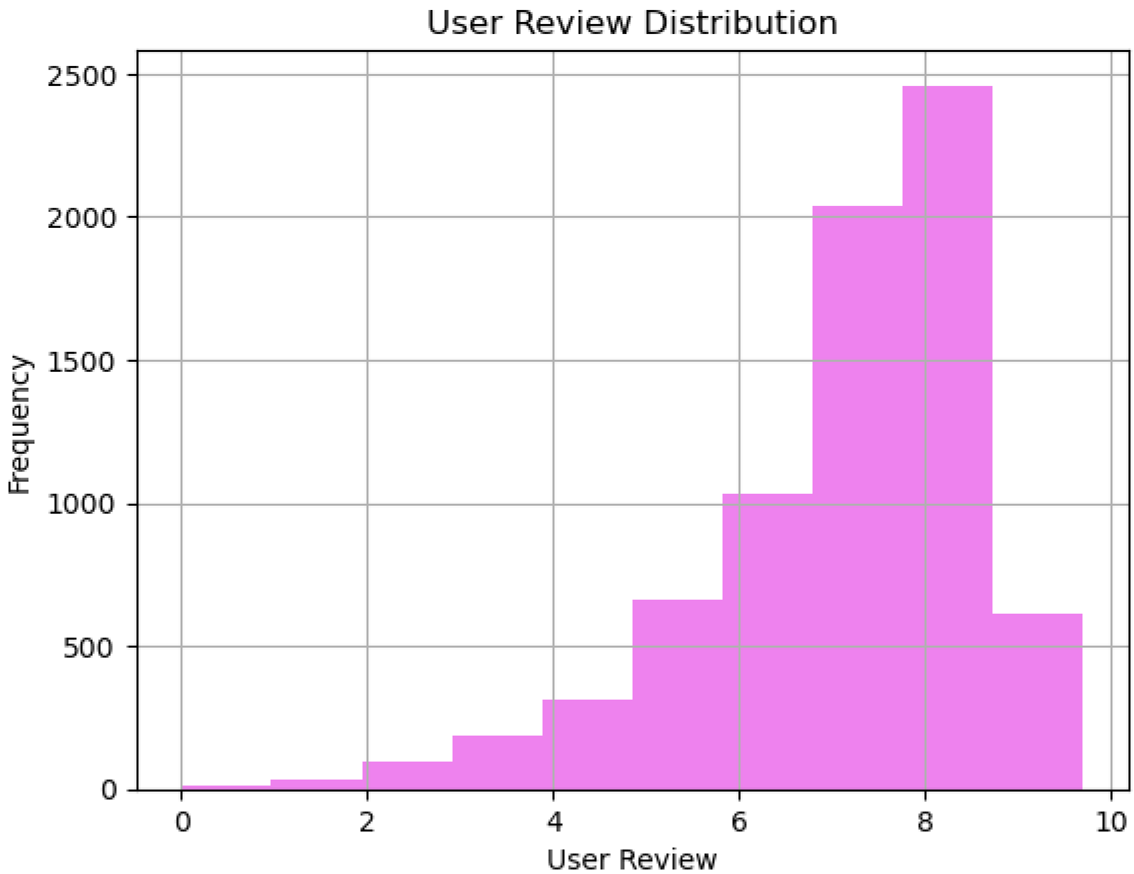
In [18]:
```
plt.figure(figsize=(12, 6))
sns.countplot(x='Platform', data=video_games_cleaned, palette='cividis')
plt.title('Distribution of Games Across Platforms')
plt.xlabel('Gaming Platforms')
plt.ylabel('Number of Games')
plt.xticks(rotation=45, ha='right')
plt.show()
```



- The visualization allows for a quick comparison of the average user reviews across various gaming platforms, helping identify which platforms tend to have higher or lower average user review scores.
- DC has the highest average user review scores abd Xone has the lowest average user review scores.

In [19]:
```
user_review_distribution = video_games_cleaned['User_Score'].hist(color='viole
user_review_distribution.set_title("User Review Distribution")
user_review_distribution.set_xlabel("User Review")
user_review_distribution.set_ylabel("Frequency")
```

Out[19]:
```
Text(0, 0.5, 'Frequency')
```

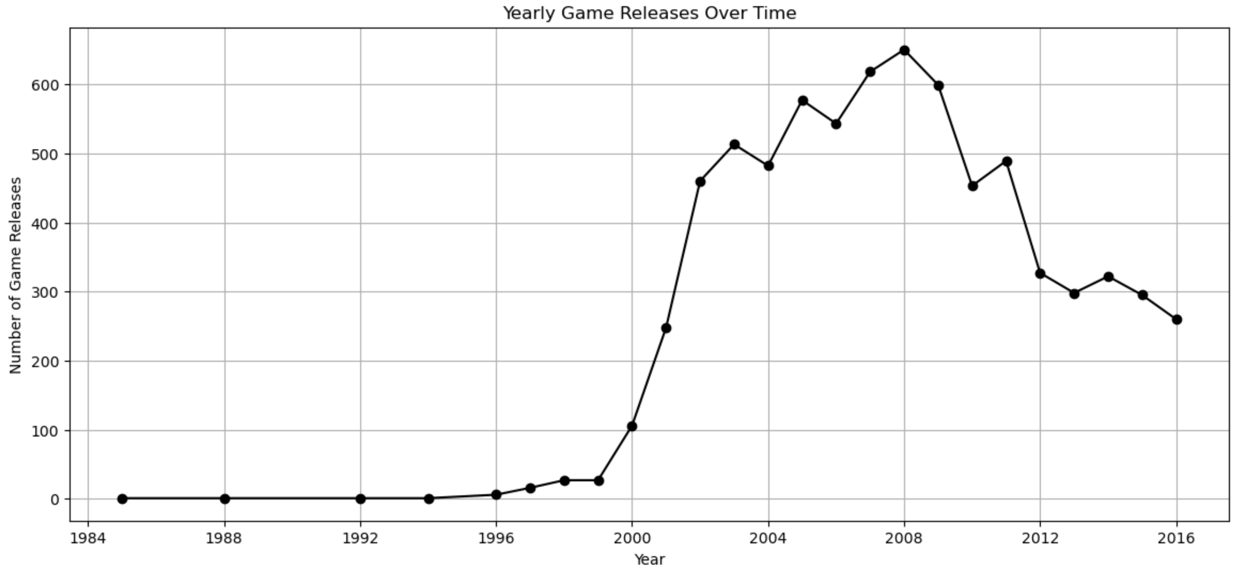


- As seen by the boxplot, most of the user ratings lie between 6 to 9 and as the median in the boxplot suggested, the most number of user scores are around 7 to 8.

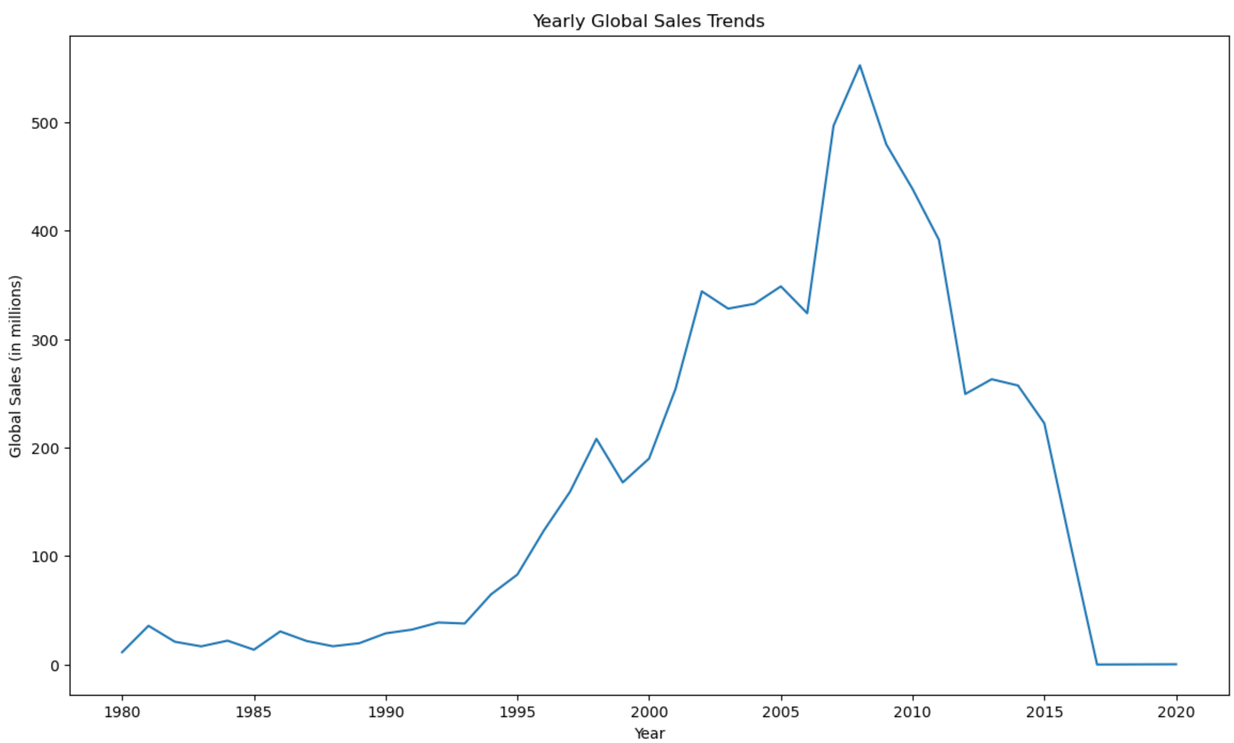
- Each bar in the plot represents a gaming platform, and the height of the bar corresponds to the number of games available on that platform.
- PS2 has the most number of games.

Trend Analysis

In [20]:
```
yearly_game_releases = video_games_cleaned.groupby(['Year_of_Release']).size()

# Create a line plot
plt.figure(figsize=(14, 6))
plt.plot(yearly_game_releases['Year_of_Release'], yearly_game_releases['num_re
plt.title('Yearly Game Releases Over Time')
plt.xlabel('Year')
plt.ylabel('Number of Game Releases')
plt.grid(True)
plt.show()
```



In [21]:
```
plt.figure(figsize=(14, 8))
sns.lineplot(x='Year_of_Release', y='Global_Sales', data=video_games.groupby('
plt.title('Yearly Global Sales Trends')
plt.xlabel('Year')
plt.ylabel('Global Sales (in millions)')
plt.show()
```



The wo plots provides a visual representation of how the number of game releases has changed across different years. By observing the line, one can see the trend in game releases over the years. The games released incresead linearly from 1995 to 1999 and then from 1999 to 2003 it increased exponentially. There were a few ups and downs between 2003 to 2008 still keeping the total releases high until it kept decreasing after 2008. Overall:

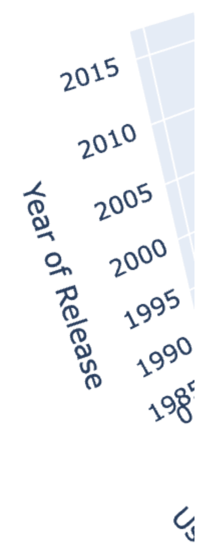
- The number of games released each year has increased by over 1000% since 1995.

- The biggest increase in the number of games released each year happened between 1999 and 2008.
- The number of games released each year seems to be declining drastically in recent years. (The COVID-19 pandemic is the most likely explanation for the decrease in the number of games released in 2020-2021. The pandemic caused disruptions to game development and production, and it also made it more difficult to market and sell games)

3D Interactive Plots

In [22]:
```
fig = px.scatter_3d(video_games_cleaned, x='Platform', y='User_Score', z='Year_
                    color='User_Score', size_max=18, opacity=0.7, labels={'use

# Update layout for better readability
fig.update_layout(title='Interactive 3D Scatter Plot - User Reviews Across Gami
                  scene=dict(xaxis_title='Platform', yaxis_title='User Score',

# Show the interactive plot
fig.show()
```

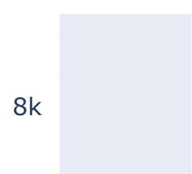
Interactive 3D Scatter Plot - User Reviews Across Gaming Pla



In [23]:
```
color_palette = 'rainbow'
```

```
fig = px.bar(video_games_cleaned, x='Platform', y='User_Score', color='Platforn
             labels={'User_Score': 'User Score', 'platform': 'Gaming Platform'}
             title='Average User Reviews Across Gaming Platforms',
             color_continuous_scale=color_palette)  # Set the color palette he

# Customize the layout to set the background color
fig.update_layout({
    'plot_bgcolor': 'rgba(255, 255, 255, 1)',  # White background
    'paper_bgcolor': 'rgba(255, 255, 255, 1)',  # White background for the plo
})

# Show the interactive plot
fig.show()
```

Average User Reviews Across Gaming Platforms



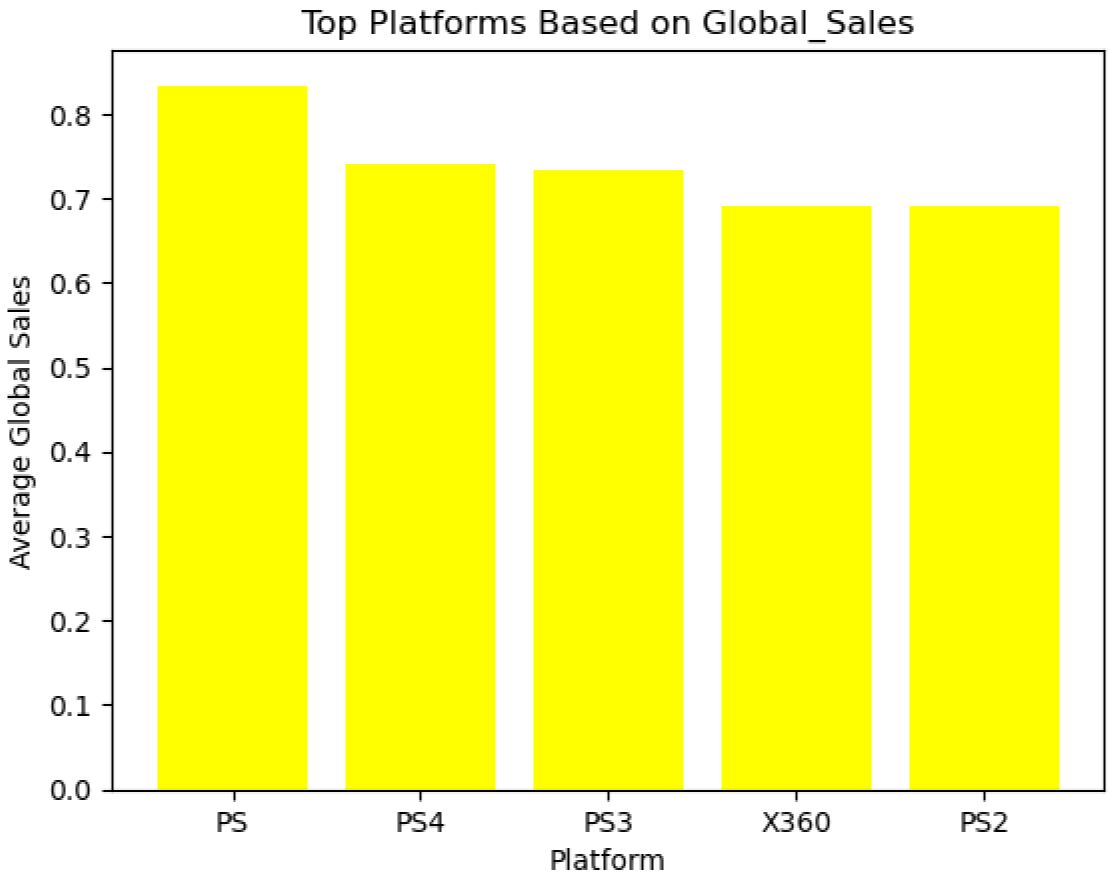
In [24]:
```
average_rating = video_games_cleaned['User_Score'].astype(float).mean()
print(f'Average User Rating: {average_rating:.2f}')

# Set a threshold for positive reviews
threshold = 7

# Categorize reviews based on the threshold
video_games_cleaned['sentiment'] = video_games_cleaned['User_Score'].astype(fl

# Analyze distribution of positive/negative reviews
positive_reviews = video_games_cleaned[video_games_cleaned['sentiment'] == 'Pos
negative_reviews = video_games_cleaned[video_games_cleaned['sentiment'] == 'Neg
```

```
print(f'Number of Positive Reviews: {len(positive_reviews)}')
print(f'Number of Negative Reviews: {len(negative_reviews)}')

# Display sentiment distribution
sentiment_distribution = video_games_cleaned['sentiment'].value_counts()
print('\nSentiment Distribution:')
print(sentiment_distribution.to_frame(name='count'))
```

```
Average User Rating: 7.11
Number of Positive Reviews: 4557
Number of Negative Reviews: 2889

Sentiment Distribution:
            count
sentiment
Positive     4557
Negative     2889
```

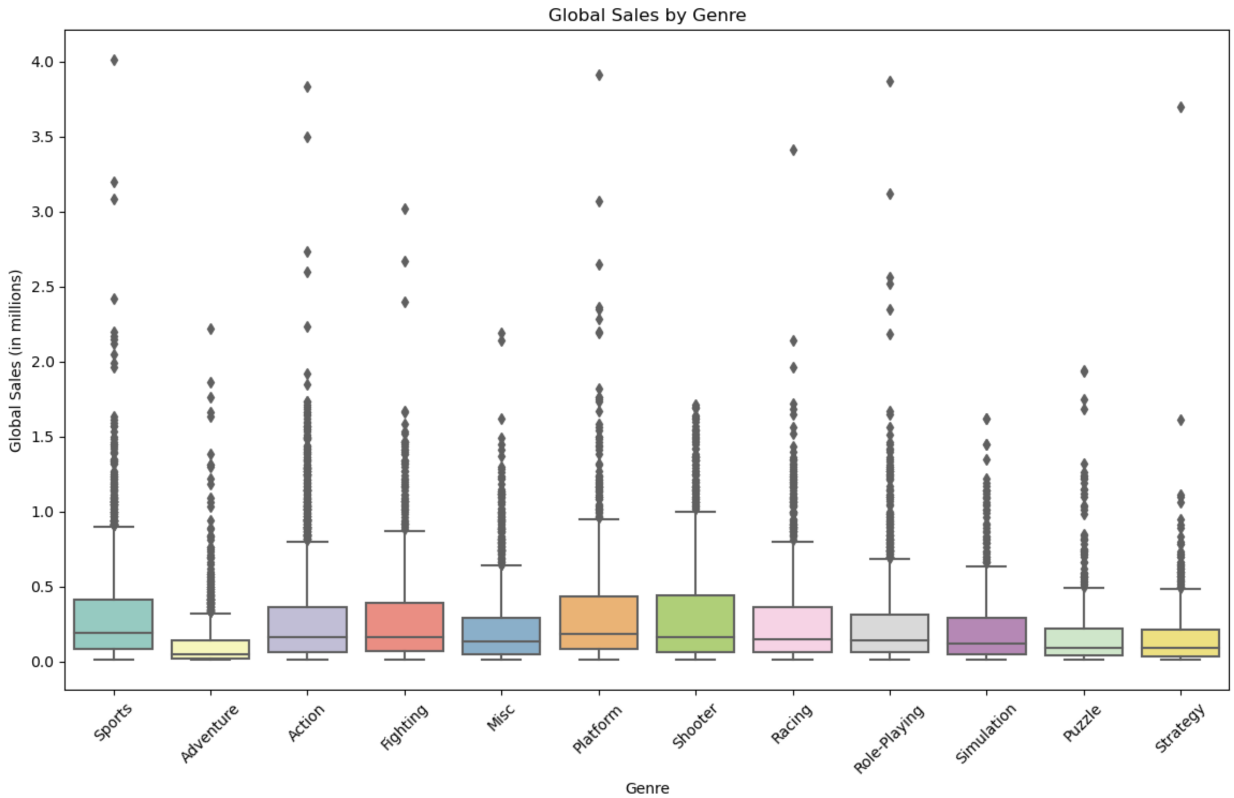
- Here the average user rating revealed that the video games in the dataset generally received positive user scores, with an average rating of 7.11 out of 10.
- We then categorized the reviews into 'Positive' and 'Negative' sentiments based on a threshold of 7.
- The sentiment analysis showed that out of the total reviews, 4557 were categorized as positive, while 2889 were deemed negative.
- This distribution provides valuable insights into the overall sentiment of the user community towards the video games.
- The majority of reviews are positive, indicating a generally favorable response from users.

In [25]:
```
platform_stats = video_games_cleaned.groupby('Platform').agg({
    'Global_Sales': 'mean',
    'Name': 'count',

}).reset_index()
top_platforms = platform_stats.nlargest(5, 'Global_Sales')
import matplotlib.pyplot as plt

plt.bar(top_platforms['Platform'], top_platforms['Global_Sales'], color='yellow
plt.xlabel('Platform')
plt.ylabel('Average Global Sales')
plt.title('Top Platforms Based on Global_Sales')
plt.show()
```



- We conducted a platform-level analysis of video game statistics, focusing on metrics such as the average global sales and the number of games released for each gaming platform.
- PlayStation has the highest average global sales.

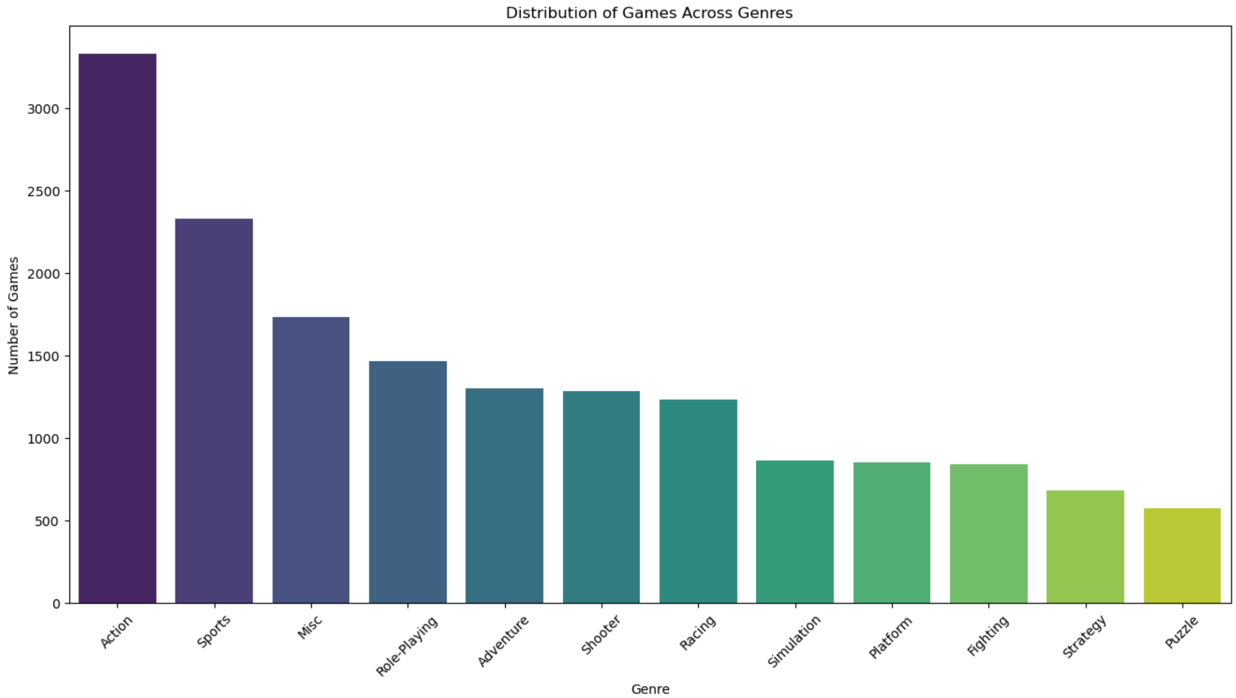
In [26]:
```
color_palette = 'Set3'

plt.figure(figsize=(14, 8))

# Use the 'palette' parameter to set the color palette
sns.boxplot(x='Genre', y='Global_Sales', data=video_games_cleaned_no_outliers,

plt.title('Global Sales by Genre')
plt.xlabel('Genre')
plt.ylabel('Global Sales (in millions)')
plt.xticks(rotation=45)
plt.show()
```



- We can see Sports, Action, Fighting, Platform and Shooter genres have the same median global sale.
- Adventure seems to have the lowest sales range and Sports have the highest.

In [27]:

```
plt.figure(figsize=(16, 8))
sns.countplot(x='Genre', data=video_games, order=video_games['Genre'].value_cou
plt.title('Distribution of Games Across Genres')
plt.xlabel('Genre')
plt.ylabel('Number of Games')
plt.xticks(rotation=45)
plt.show()

# Identify the most popular genres based on global sales
top_genres_global_sales = video_games.groupby('Genre')['Global_Sales'].sum().so
print("Top Genres Based on Global Sales:\n", top_genres_global_sales)

# Identify the most popular genres based on user reviews
top_genres_user_reviews = video_games_cleaned.groupby('Genre')['User_Score'].me
print("Top Genres Based on Average User Reviews:\n", top_genres_user_reviews)

# Identify the most popular genres based on critic scores
top_genres_critic_scores = video_games_cleaned.groupby('Genre')['Critic_Score']
print("Top Genres Based on Average Critic Scores:\n", top_genres_critic_scores
```


Distribution of Games Across Genres
Number of Games
0
500
1000
1500
2000
2500
3000
Action
Sports
Misc
Role-Playing
Adventure
Shooter
Racing
Simulation
Platform
Fighting
Strategy
Puzzle
Genre

```
Top Genres Based on Global Sales:
 Genre
Action          1442.86
Sports          1095.55
Shooter          708.95
Misc             641.73
Role-Playing     634.29
Racing           542.00
Platform         485.89
Fighting         395.27
Simulation       320.47
Adventure        220.97
Name: Global_Sales, dtype: float64
Top Genres Based on Average User Reviews:
 Genre
Role-Playing    7.614227
Strategy        7.295177
Fighting        7.285316
Platform        7.273735
Puzzle          7.164286
Adventure       7.131104
Simulation      7.123824
Shooter         7.055991
Action          7.031942
Racing          7.007006
Name: User_Score, dtype: float64
Top Genres Based on Average Critic Scores:
 Genre
Sports          73.952133
Strategy        72.764085
Role-Playing    72.418803
Puzzle          70.731092
Shooter         70.328655
Simulation      69.821192
Fighting        69.514512
Platform        69.393401
Racing          69.201709
Misc            67.494792
Name: Critic_Score, dtype: float64
```

- In the bar chart, we visualized the distribution of video games across different genres, providing an overview of the number of games available in each genre.

Following this, we identified the top genres based on two important metrics:

Global Sales:

- 1) Action
- 2) Sports
- 3) Shooter
- 4) Misc
- 5) Role-Playing

Average User Reviews:

- 1) Role-Playing

- 2) Strategy
- 3) Fighting
- 4) Platform
- 5) Puzzle

Average Critic Scores:

- 1) Sports
- 2) Strategy
- 3) Role-Playing
- 4) Puzzle
- 5) Shooter

These rankings provide valuable insights into the popularity of genres based on different criteria, such as commercial success (global sales), user satisfaction (average user reviews), and critical acclaim (average critic scores).

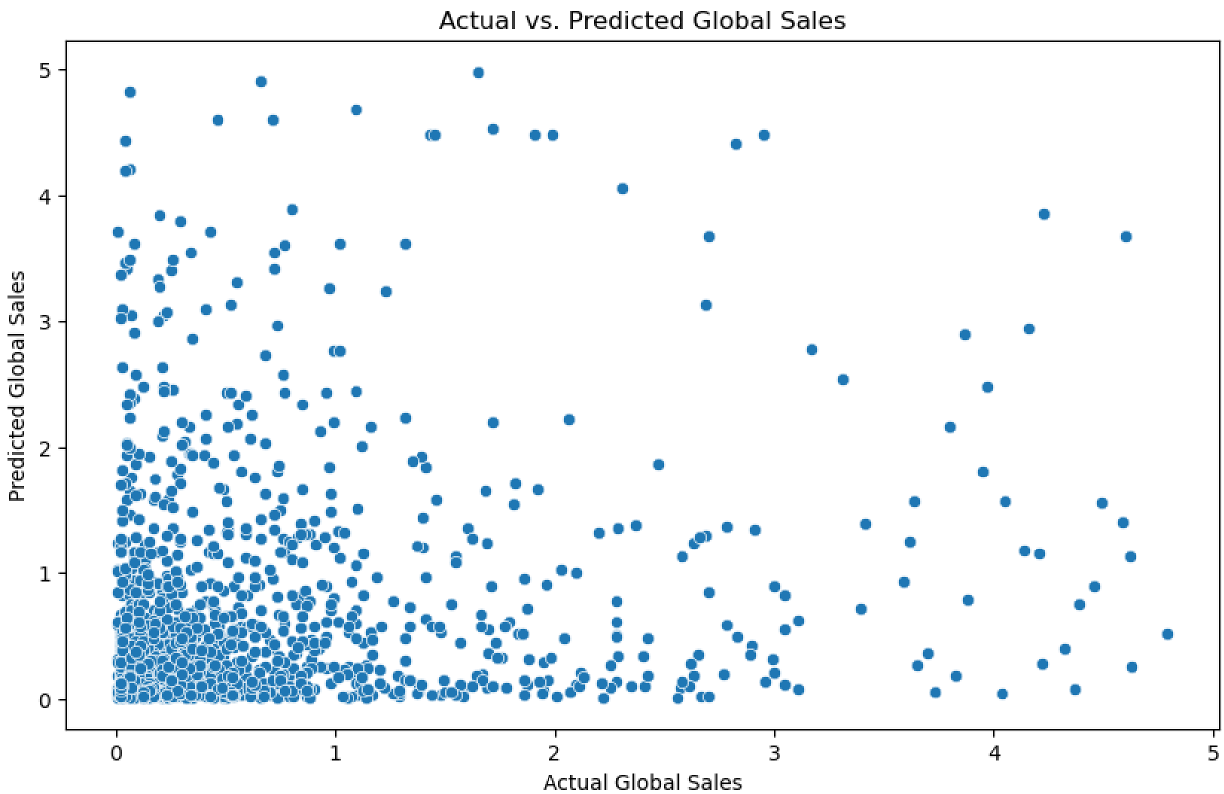
Predictive Data Analysis

In [28]:
```
video_games_cleaned = video_games_cleaned.dropna(subset=['Year_of_Release'])


video_games_cleaned['Days_Since_Release'] = (video_games_cleaned['Year_of_Relea


video_games_cleaned = video_games_cleaned.drop('Year_of_Release', axis=1)


features = ['Critic_Score', 'User_Score', 'Days_Since_Release']


video_games_cleaned = video_games_cleaned.dropna(subset=features)


X = video_games_cleaned[features]
y = video_games_cleaned['Global_Sales']


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, randon


model = DecisionTreeRegressor(random_state=42)
model.fit(X_train, y_train)


y_pred = model.predict(X_test)


mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')

feature_importance = pd.Series(model.feature_importances_, index=features).sor
print('Feature Importance:')
print(feature_importance)
```

```
plt.figure(figsize=(10, 6))
sns.scatterplot(x=y_test, y=y_pred)
plt.xlabel('Actual Global Sales')
plt.ylabel('Predicted Global Sales')
plt.title('Actual vs. Predicted Global Sales')
plt.show()
```

```
Mean Squared Error: 1.1854146516983304
Feature Importance:
User_Score           0.394023
Critic_Score         0.363433
Days_Since_Release   0.242545
dtype: float64
```

```
/var/folders/hc/ws1b3jjx4x76hlpg4bn3zgrw0000gn/T/ipykernel_12455/1241776799.p
y:4: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
```



We used a model called Decision Tree Regressor to make predictions about how well a video game might sell globally. We trained the model using our filled-in data and then tested its predictions on another set of data. To check how accurate our model is, we used a metric called Mean Squared Error (MSE), which measures how close the predicted sales are to the actual sales. A lower MSE indicates a more accurate model

The results showed that the imputed model had a Mean Squared Error (MSE) of 1.19, indicating the average squared difference between predicted and actual global sales.

Feature Importance (Imputed):

- Indicates the contribution of each feature to the model's predictions.
- `User_Score` and `Critic_Score` are the most important features, contributing significantly to the predictions.
- `Days_Since_Release` also plays a role, though to a lesser extent.

The feature importance results can guide you in understanding which features are more influential in predicting global sales.

In [29]:
```
from sklearn.linear_model import LinearRegression
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
numeric_features = X_train.select_dtypes(include=['number']).columns.tolist()
X_train_numeric = X_train[numeric_features]
X_test_numeric = X_test[numeric_features]

# Perform imputation and linear regression on X_train_numeric and X_test_numer

# Assuming you have already defined X_train, X_test, y_train, and y_test from 

# Create copies of X_train and X_test for imputation
X_train_imputed = X_train.copy()
X_test_imputed = X_test.copy()

# Identify numeric features for imputation
numeric_features = X_train.select_dtypes(include=['number']).columns.tolist()

# Impute missing values for linear regression model in numeric features
imputer = SimpleImputer(strategy='mean')
X_train_imputed[numeric_features] = imputer.fit_transform(X_train[numeric_featu
X_test_imputed[numeric_features] = imputer.transform(X_test[numeric_features])

# Create a linear regression model
linear_model = LinearRegression()
linear_model.fit(X_train_imputed, y_train)

# Make predictions on the imputed test set
y_pred_linear = linear_model.predict(X_test_imputed)

# Evaluate the linear regression model
mse_linear = mean_squared_error(y_test, y_pred_linear)
print(f'Mean Squared Error (Linear Regression): {mse_linear}')

# Feature importance (coefficients) in linear regression
feature_importance_linear = pd.Series(linear_model.coef_, index=features).sort_
print('Feature Importance (Linear Regression):')
print(feature_importance_linear)
```

```
Mean Squared Error (Linear Regression): 0.5924926302885146
Feature Importance (Linear Regression):
Critic_Score          0.020645
Days_Since_Release   -0.000007
User_Score           -0.039103
dtype: float64
```

- Now we applied a Linear Regression model to predict global video game sales. The model was trained on the imputed training data, and its predictions were evaluated on the imputed test data using Mean Squared Error (MSE).
- Additionally, in linear regression, we can examine feature importance through coefficients.

Mean Squared Error (Linear Regression): The lower the mean squared error, the better the model's performance. In this case, 0.5924 indicates relatively good performance.

Feature Importance (Linear Regression): The coefficients represent the contribution of each feature to the predicted global sales. Positive coefficients indicate a positive relationship, while negative coefficients indicate a negative relationship.

In this case:

- `Critic_Score` has a positive impact on global sales.
- `Days_Since_Release` has a very small negative impact (almost negligible) on global sales.
- `User_Score` has a negative impact on global sales.

It seems like `Critic_Score` is considered the most influential feature in this linear model.

In [30]:
```
from sklearn.ensemble import RandomForestRegressor

# Create a random forest regressor model
rf_model = RandomForestRegressor(random_state=42)
rf_model.fit(X_train_imputed, y_train)
y_pred_rf = rf_model.predict(X_test_imputed)
mse_rf = mean_squared_error(y_test, y_pred_rf)
print(f'Mean Squared Error (Random Forest): {mse_rf}')
```

```
Mean Squared Error (Random Forest): 0.7011428652491799
```

Here we employed a Random Forest Regressor model to predict global video game sales. The Random Forest Regressor is an ensemble method that leverages multiple decision trees to make predictions, offering a robust and flexible approach to regression tasks. The model was trained on the imputed training data, and its predictions were evaluated on the imputed test data using Mean Squared Error (MSE).

In [31]:
```
from sklearn.ensemble import GradientBoostingRegressor

# Create a gradient boosting regressor model
gb_model = GradientBoostingRegressor(random_state=42)
gb_model.fit(X_train_imputed, y_train)
y_pred_gb = gb_model.predict(X_test_imputed)
```

```
mse_gb = mean_squared_error(y_test, y_pred_gb)
print(f'Mean Squared Error (Gradient Boosting): {mse_gb}')
```

```
Mean Squared Error (Gradient Boosting): 0.5628410622074456
```

We utilized a Gradient Boosting Regressor model to predict global video game sales. The model was trained on the imputed training data, and its predictions were evaluated on the imputed test data using Mean Squared Error (MSE). Gradient Boosting is an ensemble learning technique that builds a series of weak learners, typically decision trees, sequentially. Each tree corrects the errors of the previous one, leading to improved predictive performance.

Conclusion

We used three different machine learning models to predict global video game sales: Decision Tree Regressor, Linear Regression, and Random Forest Regressor. We also used Gradient Boosting Regressor to predict global video game sales.

We first used a SimpleImputer to fill in missing values in the data. Then, we trained each model on the imputed training data and evaluated its predictions on the imputed test data using Mean Squared Error (MSE).

Here are the results of our experiments:

- Decision Tree Regressor: MSE = 1.19
- Linear Regression: MSE = 0.5924
- Random Forest Regressor: MSE = 0.7011
- Gradient Boosting Regressor: MSE = 0.562

Based on the MSE results, the Gradient Boosting Regressor is the best model for predicting global video game sales. This is because it has the lowest MSE, which means that its predictions are the closest to the actual sales.

We also examined feature importance for the Decision Tree Regressor and Linear Regression models. Feature importance is a measure of how much each feature contributes to the model's predictions.

Here are the feature importance results for the **Decision Tree Regressor model** :

- User_Score 0.394023
- Critic_Score 0.363433
- Days_Since_Release 0.242545

Here are the feature importance results for the **Linear Regression model** :

- Critic_Score 0.020645
- Days_Since_Release -0.000007
- User_Score -0.039103

Based on the feature importance results, the most important features for the Decision Tree Regressor model are `User_Score` and `Critic_Score`. The most important feature for the Linear Regression model is `Critic_Score`.

Overall, our experiments show that machine learning can be used to predict global video game sales. The Gradient Boosting Regressor is the best model for this task, and the most important features are `User_Score`, `Critic_Score`, and `Days_Since_Release`. Additionally, the insights gained from feature importance analysis can inform decision-making in the video game industry, providing valuable guidance on factors influencing global sales.

Prescriptive Data Analysis

Platform Strategy:

Recommendation:

- Focus on platforms with historically high average user scores and global sales. Platforms like PS2, Wii, and X360 have demonstrated strong performance.

Action Item:

- Consider developing and optimizing games for platforms with proven success, taking into account user preferences and market trends.

Genre Consideration:

Recommendation:

- Understand the popularity of game genres and tailor game development strategies accordingly. Genres like Action, Sports, and Shooter tend to have higher global sales.

Action Item:

- Conduct market research to identify genre preferences and invest in game development within those genres. Balance innovation with proven success factors.

User Review Emphasis:

Recommendation:

- Prioritize user reviews and satisfaction. Positive reviews contribute to higher average global sales and can enhance the overall success of a game.

Action Item:

- Pay attention to user feedback, address concerns, and continuously improve gameplay experiences. Engage with the gaming community to build positive relationships.

Adaptation to Market Changes:

Recommendation:

- Be adaptable to changes in the gaming market. External factors, such as technological advancements or unforeseen events (like the COVID-19 pandemic), can influence consumer behavior.

Action Item:

- Continuously monitor industry trends, technological advancements, and market dynamics. Be prepared to pivot strategies based on changing conditions.

Data-Driven Decision-Making:

Recommendation:

- Leverage data analytics for decision-making. Utilize insights from user scores, sales data, and industry trends to inform strategic decisions.

Action Item:

- Invest in data analytics capabilities, employ machine learning models for predictive analysis, and continuously refine strategies based on real-time data feedback.

These prescriptions are based on the patterns and trends observed in the dataset. However, it's important for stakeholders to complement these recommendations with market-specific research and a deep understanding of their target audience. Additionally, staying agile and responsive to evolving industry dynamics is key for long-term success.

Source DataSet - https://www.kaggle.com/datasets/sidtwr/videogames-sales-dataset

In [ ]:

```

```