```python
import pandas as pd

# Load the dataset
df = pd.read_csv('vgsales.csv')

# Display first few rows
df.head()
```

```
   Rank                      Name Platform    Year         Genre
Publisher  \
0     1                Wii Sports      Wii  2006.0        Sports
Nintendo
1     2          Super Mario Bros.      NES  1985.0      Platform
Nintendo
2     3             Mario Kart Wii      Wii  2008.0        Racing
Nintendo
3     4          Wii Sports Resort      Wii  2009.0        Sports
Nintendo
4     5  Pokemon Red/Pokemon Blue       GB  1996.0  Role-Playing
Nintendo

   NA_Sales  EU_Sales  JP_Sales  Other_Sales  Global_Sales
0     41.49     29.02      3.77         8.46         82.74
1     29.08      3.58      6.81         0.77         40.24
2     15.85     12.88      3.79         3.31         35.82
3     15.75     11.01      3.28         2.96         33.00
4     11.27      8.89     10.22         1.00         31.37
```

```python
# Display basic information about the dataset
df.info()

# Statistical overview of numerical columns
df.describe()

# Check for missing values in each column
df.isnull().sum()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16598 entries, 0 to 16597
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Rank          16598 non-null  int64
 1   Name          16598 non-null  object
 2   Platform      16598 non-null  object
 3   Year          16327 non-null  float64
 4   Genre         16598 non-null  object
 5   Publisher     16540 non-null  object
 6   NA_Sales      16598 non-null  float64
 7   EU_Sales      16598 non-null  float64
```

```
 8   JP_Sales       16598 non-null  float64
 9   Other_Sales    16598 non-null  float64
 10  Global_Sales   16598 non-null  float64
dtypes: float64(6), int64(1), object(4)
memory usage: 1.4+ MB
```

```
Rank              0
Name              0
Platform          0
Year            271
Genre             0
Publisher        58
NA_Sales          0
EU_Sales          0
JP_Sales          0
Other_Sales       0
Global_Sales      0
dtype: int64
```

```
# Drop rows with missing values in 'Year' and 'Publisher'
df_cleaned = df.dropna(subset=['Year', 'Publisher'])

# Convert 'Year' column to integer for better handling
df_cleaned['Year'] = df_cleaned['Year'].astype(int)

# Drop duplicates if any
df_cleaned = df_cleaned.drop_duplicates()

# Display cleaned data summary
df_cleaned.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 16291 entries, 0 to 16597
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Rank          16291 non-null  int64
 1   Name          16291 non-null  object
 2   Platform      16291 non-null  object
 3   Year          16291 non-null  int32
 4   Genre         16291 non-null  object
 5   Publisher     16291 non-null  object
 6   NA_Sales      16291 non-null  float64
 7   EU_Sales      16291 non-null  float64
 8   JP_Sales      16291 non-null  float64
 9   Other_Sales   16291 non-null  float64
 10  Global_Sales  16291 non-null  float64
dtypes: float64(5), int32(1), int64(1), object(4)
memory usage: 1.4+ MB
```

```
C:\Users\MIT\AppData\Local\Temp\ipykernel_15448\540113173.py:5:
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/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df_cleaned['Year'] = df_cleaned['Year'].astype(int)
```

```python
#  Find all games with Global Sales > 10 million
high_sales = df_cleaned[df_cleaned['Global_Sales'] > 10]
print(high_sales[['Name', 'Global_Sales']])
```

```
                                  Name  Global_Sales
0                         Wii Sports         82.74
1                  Super Mario Bros.         40.24
2                     Mario Kart Wii         35.82
3                  Wii Sports Resort         33.00
4            Pokemon Red/Pokemon Blue         31.37
..                                ...           ...
57              Super Mario All-Stars         10.55
58  Pokemon FireRed/Pokemon LeafGreen         10.49
59                    Super Mario 64         10.42
60                      Just Dance 3         10.26
61               Call of Duty: Ghosts         10.21

[62 rows x 2 columns]
```

```python
#  Find games released before the year 2000
old_games = df_cleaned[df_cleaned['Year'] < 2000]
print(old_games[['Name', 'Year']])
```

```
                                Name  Year
1                 Super Mario Bros.  1985
4           Pokemon Red/Pokemon Blue  1996
5                             Tetris  1989
9                         Duck Hunt  1984
12       Pokemon Gold/Pokemon Silver  1999
...                              ...   ...
16379             Grand Prix Legends  1997
16436                       Worms 2  1997
16506  Samurai Shodown: Warrios Rage  1999
16554              Psychic Detective  1995
16583                 Carmageddon 64  1999

[1974 rows x 2 columns]
```

```python
#  Find games where NA Sales exactly match EU Sales
equal_sales = df_cleaned[df_cleaned['NA_Sales'] ==
```

```
df_cleaned['EU_Sales']]
print(equal_sales[['Name', 'NA_Sales', 'EU_Sales']])
```

```
                                                    Name  NA_Sales
EU_Sales
214                              Monster Hunter Freedom 3      0.00
0.00
253                               Resistance: Fall of Man      1.73
1.73
326                          Ratchet & Clank: Size Matters      1.40
1.40
338                                      Friend Collection      0.00
0.00
383                                       Monster Hunter 4      0.00
0.00
...                                                    ...       ...
...
16580                                          Real Rode      0.00
0.00
16587                              Mezase!! Tsuri Master DS      0.00
0.00
16589  Chou Ezaru wa Akai Hana: Koi wa Tsuki ni Shiru...      0.00
0.00
16590  Eiyuu Densetsu: Sora no Kiseki Material Collec...      0.00
0.00
16595   SCORE International Baja 1000: The Official Game      0.00
0.00

[3657 rows x 3 columns]
```

```
#  Find games with Global Sales > 10 AND NA Sales > 5
popular_games = df_cleaned[(df_cleaned['Global_Sales'] > 10) &
(df_cleaned['NA_Sales'] > 5)]
print(popular_games[['Name', 'Global_Sales', 'NA_Sales']])
```

```
                                       Name  Global_Sales  NA_Sales
0                                 Wii Sports         82.74     41.49
1                          Super Mario Bros.         40.24     29.08
2                             Mario Kart Wii         35.82     15.85
3                          Wii Sports Resort         33.00     15.75
4                Pokemon Red/Pokemon Blue            31.37     11.27
5                                    Tetris         30.26     23.20
6                   New Super Mario Bros.            30.01     11.38
7                                 Wii Play         29.02     14.03
8               New Super Mario Bros. Wii          28.62     14.59
9                                 Duck Hunt         28.31     26.93
10                               Nintendogs         24.76      9.07
11                            Mario Kart DS         23.42      9.81
12            Pokemon Gold/Pokemon Silver           23.10      9.00
13                                   Wii Fit         22.72      8.94
```

```
14                              Wii Fit Plus          22.00      9.09
15                      Kinect Adventures!            21.82     14.97
16                      Grand Theft Auto V            21.40      7.01
17            Grand Theft Auto: San Andreas           20.81      9.43
18                       Super Mario World            20.61     12.78
20            Pokemon Diamond/Pokemon Pearl           18.36      6.42
21                         Super Mario Land           18.14     10.83
22                    Super Mario Bros. 3             17.28      9.54
23                     Grand Theft Auto V             16.38      9.63
24              Grand Theft Auto: Vice City           16.15      8.41
25            Pokemon Ruby/Pokemon Sapphire           15.85      6.06
26              Pokemon Black/Pokemon White           15.32      5.57
28                   Gran Turismo 3: A-Spec           14.98      6.85
29           Call of Duty: Modern Warfare 3           14.76      9.03
30  Pokémon Yellow: Special Pikachu Edition           14.64      5.89
31                  Call of Duty: Black Ops           14.64      9.67
32                      Pokemon X/Pokemon Y           14.35      5.17
33                Call of Duty: Black Ops 3           14.24      5.77
35               Call of Duty: Black Ops II           13.73      8.25
36           Call of Duty: Modern Warfare 2           13.51      8.52
37           Call of Duty: Modern Warfare 3           13.46      5.54
38                     Grand Theft Auto III           13.10      6.99
39                  Super Smash Bros. Brawl           13.04      6.75
40                  Call of Duty: Black Ops           12.73      5.98
43                                 Halo 3             12.14      7.97
46                            Super Mario 64           11.89      6.91
48                      Super Mario Galaxy             11.52      6.16
50       Super Mario Land 2: 6 Golden Coins            11.18      6.16
51                     Grand Theft Auto IV             11.02      6.76
57                   Super Mario All-Stars             10.55      5.99
59                            Super Mario 64           10.42      5.08
60                              Just Dance 3           10.26      6.05
61                     Call of Duty: Ghosts            10.21      6.72
```

```
#  Find games released after 2010 OR with EU Sales > 5
recent_or_popular = df_cleaned[(df_cleaned['Year'] > 2010) |
(df_cleaned['EU_Sales'] > 5)]
print(recent_or_popular[['Name', 'Year', 'EU_Sales']])
```

```
                                                  Name  Year
EU_Sales
0                                           Wii Sports  2006
29.02
2                                        Mario Kart Wii  2008
12.88
3                                     Wii Sports Resort  2009
11.01
4                                 Pokemon Red/Pokemon Blue  1996
8.89
6                                    New Super Mario Bros.  2006
```

```
9.23
...                                                   ...   ...       .
..
16576                                  Rugby Challenge 3  2016
0.01
16578                       Outdoors Unleashed: Africa 3D  2011
0.00
16581                                          Fit & Fun  2011
0.01
16585                                             Breach  2011
0.00
16589  Chou Ezaru wa Akai Hana: Koi wa Tsuki ni Shiru...  2016
0.00

[3896 rows x 3 columns]
```

```python
# Find all games published by Nintendo
nintendo_games = df_cleaned.loc[df_cleaned['Publisher'] == 'Nintendo']
print(nintendo_games[['Name', 'Publisher']])
```

```
                                                 Name Publisher
0                                          Wii Sports  Nintendo
1                                    Super Mario Bros.  Nintendo
2                                       Mario Kart Wii  Nintendo
3                                    Wii Sports Resort  Nintendo
4                              Pokemon Red/Pokemon Blue  Nintendo
...                                                 ...       ...
16269                          Slide Adventure: Mag Kid  Nintendo
16357          Mario vs. Donkey Kong: Tipping Stars  Nintendo
16456                          Art Academy: Home Studio  Nintendo
16473                                    Captain Rainbow  Nintendo
16542  Mario & Luigi: Paper Jam & Mario Kart 7 Double...  Nintendo

[696 rows x 2 columns]
```

```python
#  Retrieve the first 5 rows using .iloc
first_five_rows = df_cleaned.iloc[:5]
print(first_five_rows)
```

```
   Rank                      Name Platform  Year         Genre
Publisher  \
0     1                Wii Sports      Wii  2006        Sports
Nintendo
1     2          Super Mario Bros.      NES  1985      Platform
Nintendo
2     3             Mario Kart Wii      Wii  2008        Racing
Nintendo
3     4          Wii Sports Resort      Wii  2009        Sports
Nintendo
4     5  Pokemon Red/Pokemon Blue       GB  1996  Role-Playing
```

```
Nintendo
```

```
   NA_Sales  EU_Sales  JP_Sales  Other_Sales  Global_Sales
0     41.49     29.02      3.77         8.46         82.74
1     29.08      3.58      6.81         0.77         40.24
2     15.85     12.88      3.79         3.31         35.82
3     15.75     11.01      3.28         2.96         33.00
4     11.27      8.89     10.22         1.00         31.37
```

```
#  Find all games with NA Sales > 5 using .loc
high_na_sales = df_cleaned.loc[df_cleaned['NA_Sales'] > 5, ['Name',
'NA_Sales']]
print(high_na_sales)
```

```
                                       Name  NA_Sales
0                                Wii Sports     41.49
1                          Super Mario Bros.     29.08
2                             Mario Kart Wii     15.85
3                          Wii Sports Resort     15.75
4                   Pokemon Red/Pokemon Blue     11.27
5                                    Tetris     23.20
6                       New Super Mario Bros.     11.38
7                                    Wii Play     14.03
8                  New Super Mario Bros. Wii     14.59
9                                  Duck Hunt     26.93
10                                 Nintendogs      9.07
11                              Mario Kart DS      9.81
12               Pokemon Gold/Pokemon Silver      9.00
13                                   Wii Fit      8.94
14                               Wii Fit Plus      9.09
15                         Kinect Adventures!     14.97
16                         Grand Theft Auto V      7.01
17              Grand Theft Auto: San Andreas      9.43
18                           Super Mario World     12.78
20              Pokemon Diamond/Pokemon Pearl      6.42
21                            Super Mario Land     10.83
22                      Super Mario Bros. 3      9.54
23                         Grand Theft Auto V      9.63
24               Grand Theft Auto: Vice City      8.41
25              Pokemon Ruby/Pokemon Sapphire      6.06
26               Pokemon Black/Pokemon White      5.57
28                    Gran Turismo 3: A-Spec      6.85
29            Call of Duty: Modern Warfare 3      9.03
30  Pokémon Yellow: Special Pikachu Edition      5.89
31                   Call of Duty: Black Ops      9.67
32                       Pokemon X/Pokemon Y      5.17
33                 Call of Duty: Black Ops 3      5.77
35                Call of Duty: Black Ops II      8.25
36            Call of Duty: Modern Warfare 2      8.52
37            Call of Duty: Modern Warfare 3      5.54
```

```
38                      Grand Theft Auto III      6.99
39                   Super Smash Bros. Brawl      6.75
40                   Call of Duty: Black Ops      5.98
43                                   Halo 3       7.97
46                             Super Mario 64      6.91
48                          Super Mario Galaxy      6.16
50       Super Mario Land 2: 6 Golden Coins      6.16
51                        Grand Theft Auto IV      6.76
57                     Super Mario All-Stars      5.99
59                             Super Mario 64      5.08
60                               Just Dance 3      6.05
61                        Call of Duty: Ghosts      6.72
62                                 Halo: Reach      7.03
63                               Mario Kart 64      5.55
65                                   Halo 4       6.63
68                               Just Dance 2      5.84
70           Call of Duty 4: Modern Warfare      5.91
72                                  Minecraft      5.58
75              The Elder Scrolls V: Skyrim      5.03
79                                   Halo 2       6.82
84                             GoldenEye 007      5.80
89                                   Pac-Man      7.28
96                         Super Mario Bros. 2      5.39
```

```
#  Total Global Sales by Genre
genre_sales = df_cleaned.groupby('Genre')
['Global_Sales'].sum().reset_index()
print(genre_sales)
```

```
           Genre  Global_Sales
0         Action       1722.84
1      Adventure        234.59
2       Fighting        444.05
3           Misc        789.87
4       Platform        829.13
5         Puzzle        242.21
6         Racing        726.76
7   Role-Playing        923.83
8        Shooter       1026.20
9     Simulation        389.98
10        Sports       1309.24
11      Strategy        173.27
```

```
#  Average NA Sales by Platform
platform_na_avg = df_cleaned.groupby('Platform')
['NA_Sales'].mean().reset_index()
print(platform_na_avg)
```

```
   Platform  NA_Sales
0      2600  0.696379
```

```
1        3DO  0.000000
2        3DS  0.156373
3         DC  0.104423
4         DS  0.182323
5         GB  1.171546
6        GBA  0.227010
7         GC  0.243432
8        GEN  0.713704
9         GG  0.000000
10       N64  0.439589
11       NES  1.285102
12        NG  0.000000
13        PC  0.098124
14     PCFX  0.000000
15        PS  0.281505
16       PS2  0.269356
17       PS3  0.298236
18       PS4  0.288095
19       PSP  0.089465
20       PSV  0.039195
21       SAT  0.004162
22       SCD  0.166667
23     SNES  0.256192
24     TG16  0.000000
25        WS  0.000000
26       Wii  0.385558
27     WiiU  0.267972
28     X360  0.481629
29        XB  0.226725
30     XOne  0.390563
```

```
# Count of Games Released per Year
games_per_year =
df_cleaned.groupby('Year').size().reset_index(name='Count')
print(games_per_year)
```

```
    Year  Count
0   1980      9
1   1981     46
2   1982     36
3   1983     17
4   1984     14
5   1985     14
6   1986     21
7   1987     16
8   1988     15
9   1989     17
10  1990     16
11  1991     41
12  1992     43
```

```
13  1993     60
14  1994    121
15  1995    219
16  1996    263
17  1997    289
18  1998    379
19  1999    338
20  2000    349
21  2001    482
22  2002    829
23  2003    775
24  2004    744
25  2005    936
26  2006   1008
27  2007   1201
28  2008   1428
29  2009   1431
30  2010   1257
31  2011   1136
32  2012    655
33  2013    546
34  2014    580
35  2015    614
36  2016    342
37  2017      3
38  2020      1
```

```
# .Find the Platform with the Highest Total Global Sales
platform_sales = df_cleaned.groupby('Platform')['Global_Sales'].sum()
top_platform = platform_sales.idxmax()
print(f'Top platform by global sales: {top_platform}')
```

```
Top platform by global sales: PS2
```

```
#  Top 3 Publishers by Global Sales
top_publishers = df_cleaned.groupby('Publisher')
['Global_Sales'].sum().nlargest(3).reset_index()
print(top_publishers)
```

```
         Publisher  Global_Sales
0         Nintendo       1784.43
1  Electronic Arts       1093.39
2       Activision        721.41
```

```
# Average JP Sales by Genre for Games with JP Sales > 1
jp_sales_avg = df_cleaned[df_cleaned['JP_Sales'] > 1].groupby('Genre')
['JP_Sales'].mean().reset_index()
print(jp_sales_avg)
```

```
           Genre  JP_Sales
0         Action  1.546842
```

```
1      Adventure  1.786667
2       Fighting  1.705714
3           Misc  1.808571
4       Platform  2.088378
5         Puzzle  1.951667
6         Racing  2.349167
7   Role-Playing  2.427632
8        Shooter  1.303333
9     Simulation  1.964167
10        Sports  1.800417
11      Strategy  1.318000
```

```
# Number of Games Released per Platform per Year
platform_year_count = df_cleaned.groupby(['Platform',
'Year']).size().reset_index(name='Count')
print(platform_year_count)
```

```
    Platform  Year  Count
0       2600  1980      9
1       2600  1981     46
2       2600  1982     36
3       2600  1983     11
4       2600  1984      1
..       ...   ...    ...
236       XB  2008      1
237     XOne  2013     19
238     XOne  2014     61
239     XOne  2015     79
240     XOne  2016     54

[241 rows x 3 columns]
```

```
#  Find All Games Released in 2005 with Global Sales > 2
games_2005 = df_cleaned.loc[(df_cleaned['Year'] == 2005) &
(df_cleaned['Global_Sales'] > 2)]
print(games_2005[['Name', 'Global_Sales']])
```

```
                                                  Name  Global_Sales
10                                          Nintendogs         24.76
11                                       Mario Kart DS         23.42
19        Brain Age: Train Your Brain in Minutes a Day         20.22
27         Brain Age 2: More Training in Minutes a Day         15.30
41                           Animal Crossing: Wild World         12.27
90                Grand Theft Auto: Liberty City Stories          7.72
122                                   Big Brain Academy          6.67
211                                       Madden NFL 06          4.91
245                                          God of War          4.45
252                         Need for Speed: Most Wanted          4.37
255                                   Kingdom Hearts II          4.33
269                                      FIFA Soccer 06          4.21
```

```
292                      World Soccer Winning Eleven 9          4.06
302                    Namco Museum: 50th Anniversary           3.98
311                                          Tekken 5           3.87
340                       Midnight Club 3: DUB Edition          3.66
350                                  Resident Evil 4           3.62
357                          Star Wars: Battlefront II         3.59
364                    LEGO Star Wars: The Video Game          3.53
406        Star Wars Episode III: Revenge of the Sith          3.32
437                                        Sonic Rush          3.15
486                         WWE SmackDown! vs. RAW 2006        2.94
565                         Call Of Duty 2: Big Red One        2.67
656                                       Guitar Hero          2.38
788                 Need for Speed: Most Wanted 5-1-0          2.10
790                Devil May Cry 3: Dante's Awakening          2.09
816     Pokemon Mystery Dungeon: Red/Blue Rescue Team          2.06
824         Pokémon Mystery Dungeon: Blue Rescue Team          2.05
833  Peter Jackson's King Kong: The Official Game o...         2.04
841                                    Call of Duty 2          2.02
```

```python
#  Total Sales by Genre and Platform (Using Multiple Grouping)
genre_platform_sales = df_cleaned.groupby(['Genre', 'Platform'])
['Global_Sales'].sum().reset_index()
print(genre_platform_sales)
```

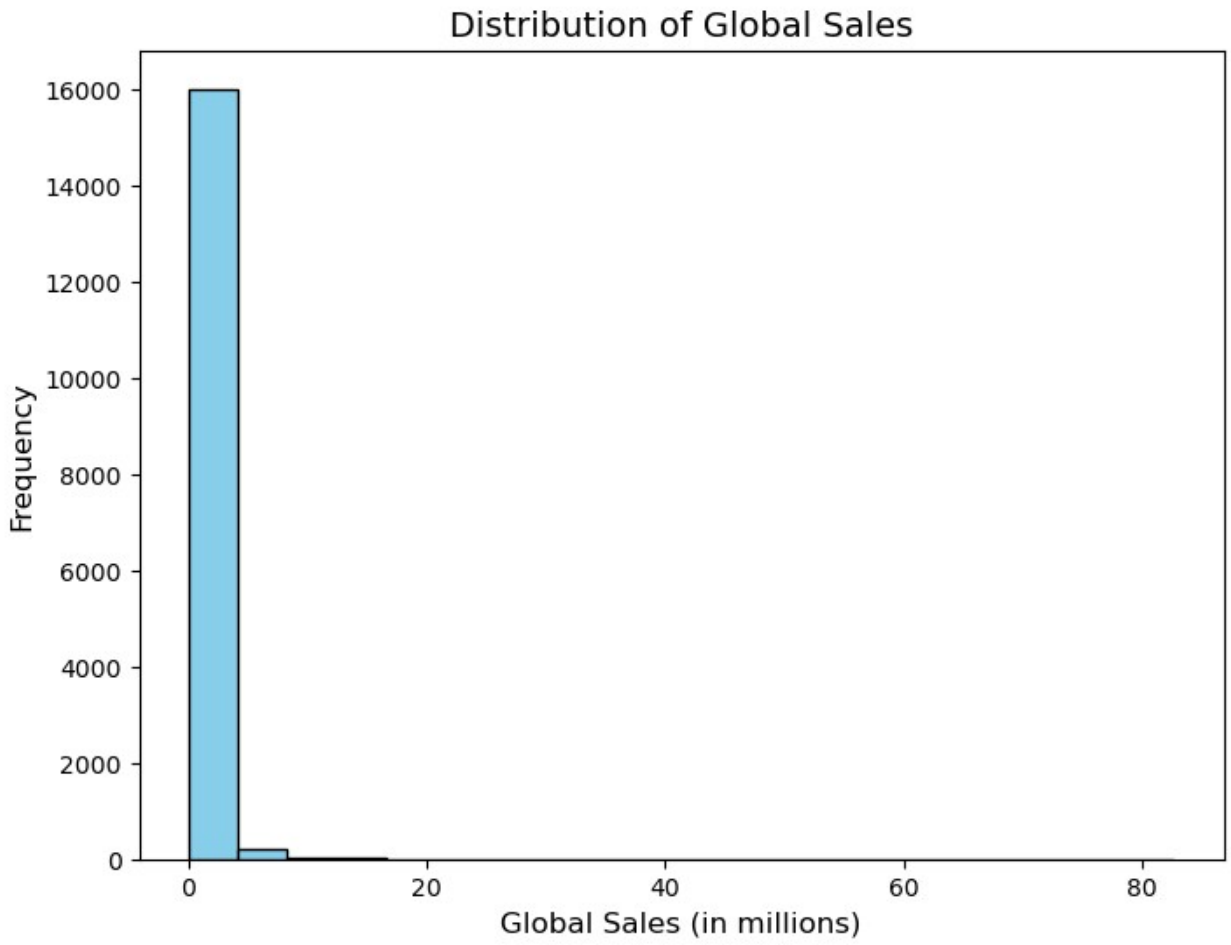
```
        Genre Platform  Global_Sales
0      Action     2600         26.39
1      Action      3DS         56.61
2      Action       DC          1.26
3      Action       DS        114.16
4      Action       GB          7.92
..        ...      ...           ...
288  Strategy      Wii          5.23
289  Strategy     WiiU          1.24
290  Strategy     X360          9.77
291  Strategy       XB          2.78
292  Strategy     XOne          0.38

[293 rows x 3 columns]
```

```python
# Drop the 'Rank' column as it's not necessary for analysis
df_cleaned = df_cleaned.drop(columns=['Rank'])

# Display first few rows of the cleaned dataset
df_cleaned.head()
```

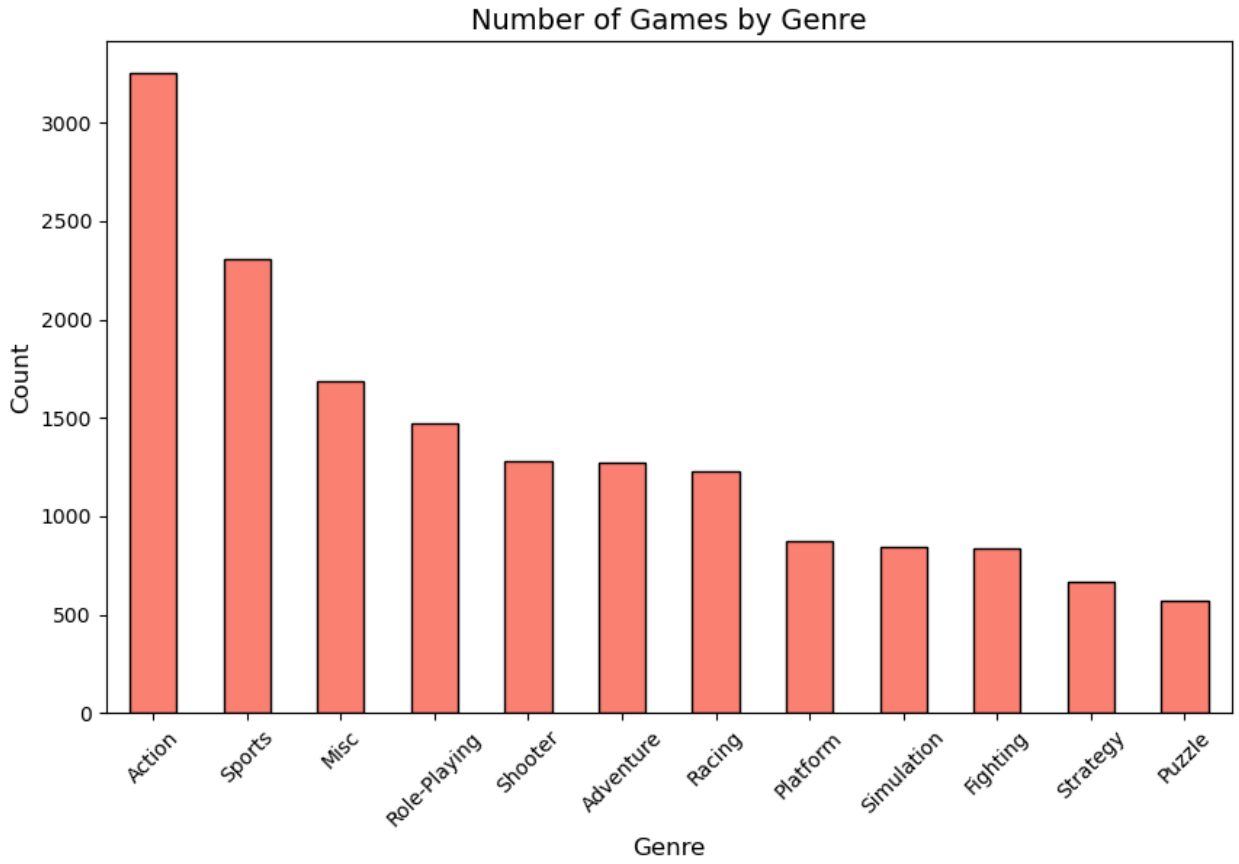
```
                     Name Platform  Year         Genre Publisher
NA_Sales  \
0              Wii Sports      Wii  2006        Sports  Nintendo
41.49
1         Super Mario Bros.      NES  1985      Platform  Nintendo
```

```
29.08
2            Mario Kart Wii      Wii  2008        Racing  Nintendo
15.85
3         Wii Sports Resort      Wii  2009        Sports  Nintendo
15.75
4  Pokemon Red/Pokemon Blue       GB  1996  Role-Playing  Nintendo
11.27

   EU_Sales  JP_Sales  Other_Sales  Global_Sales
0     29.02      3.77         8.46         82.74
1      3.58      6.81         0.77         40.24
2     12.88      3.79         3.31         35.82
3     11.01      3.28         2.96         33.00
4      8.89     10.22         1.00         31.37
```

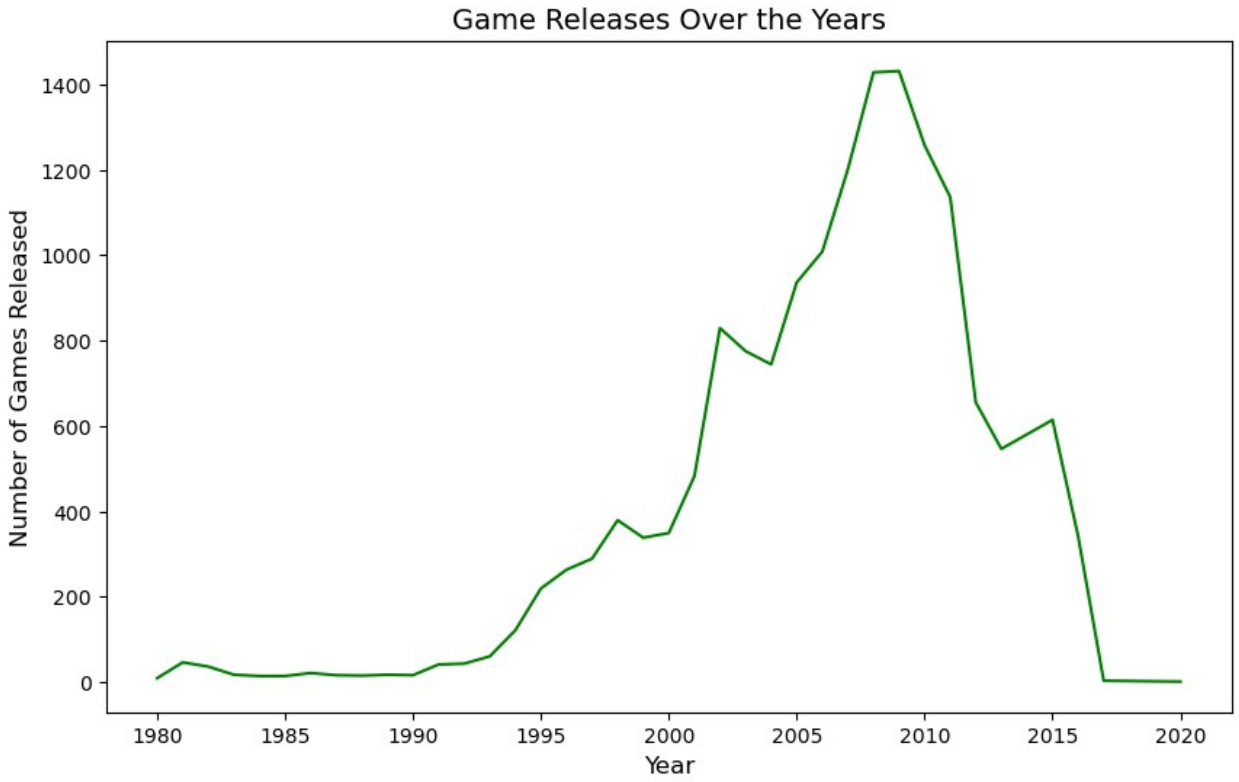
```python
import matplotlib.pyplot as plt
# 1. Distribution of Global Sales(change needed)
plt.figure(figsize=(8, 6))
plt.hist(df_cleaned['Global_Sales'], bins=20, color='skyblue',
edgecolor='black')
plt.title('Distribution of Global Sales', fontsize=14)
plt.xlabel('Global Sales (in millions)', fontsize=12)
plt.ylabel('Frequency', fontsize=12)
plt.show()
#The majority of games have low global sales (under 1 million units),
reflecting the industry's reliance on a small number of blockbuster
titles.
```



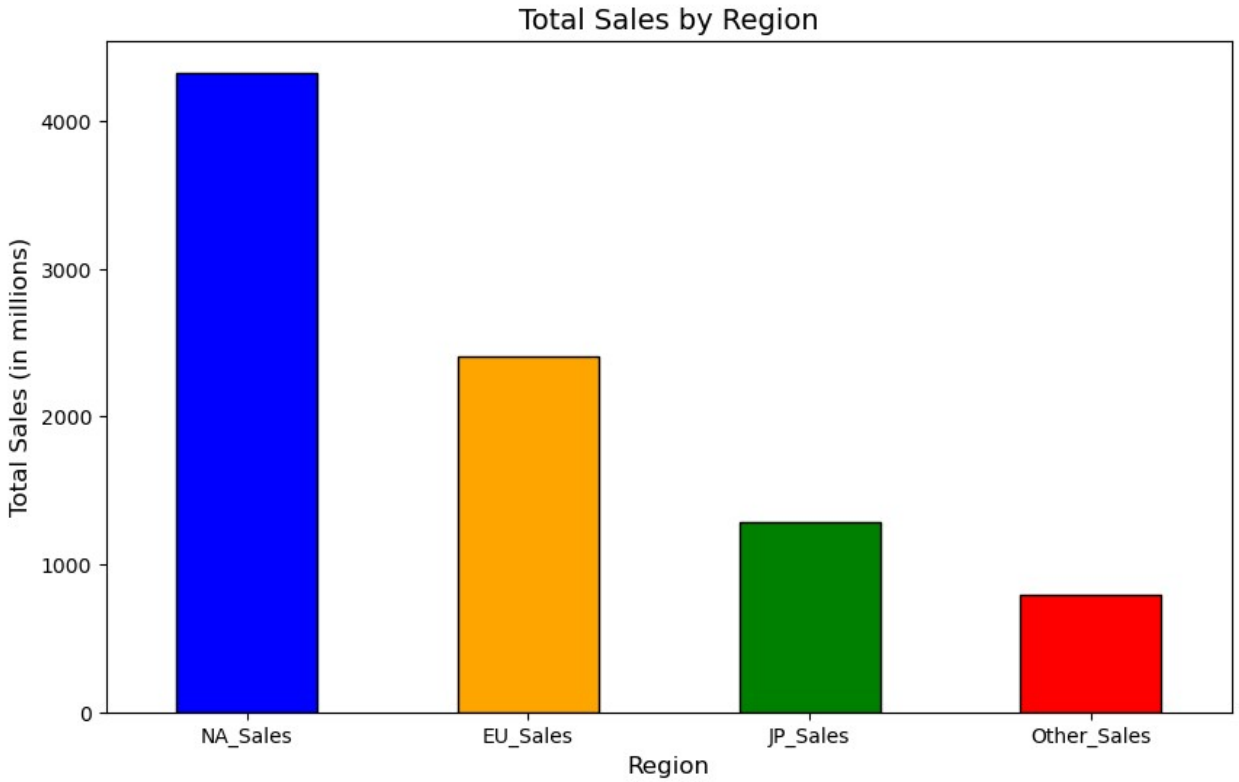

```
# 2. Sales by Genre
plt.figure(figsize=(10, 6))
df_cleaned['Genre'].value_counts().plot(kind='bar', color='salmon',
edgecolor='black')
plt.title('Number of Games by Genre', fontsize=14)
plt.xlabel('Genre', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.xticks(rotation=45)
plt.show()
#Action and Sports genres dominate: These two genres have the highest
number of game releases, reflecting their popularity and profitability
in the gaming market.
```




```
# 3. Game Releases Over the Years(More Analysis Needed for decline
after 2010)
plt.figure(figsize=(10, 6))
df_cleaned.groupby('Year').size().plot(kind='line', color='green')
plt.title('Game Releases Over the Years', fontsize=14)
plt.xlabel('Year', fontsize=12)
plt.ylabel('Number of Games Released', fontsize=12)
plt.show()
#Sales peak between 2005–2010, aligning with the 7th generation
console releases (PS3, Xbox 360, Wii).
```

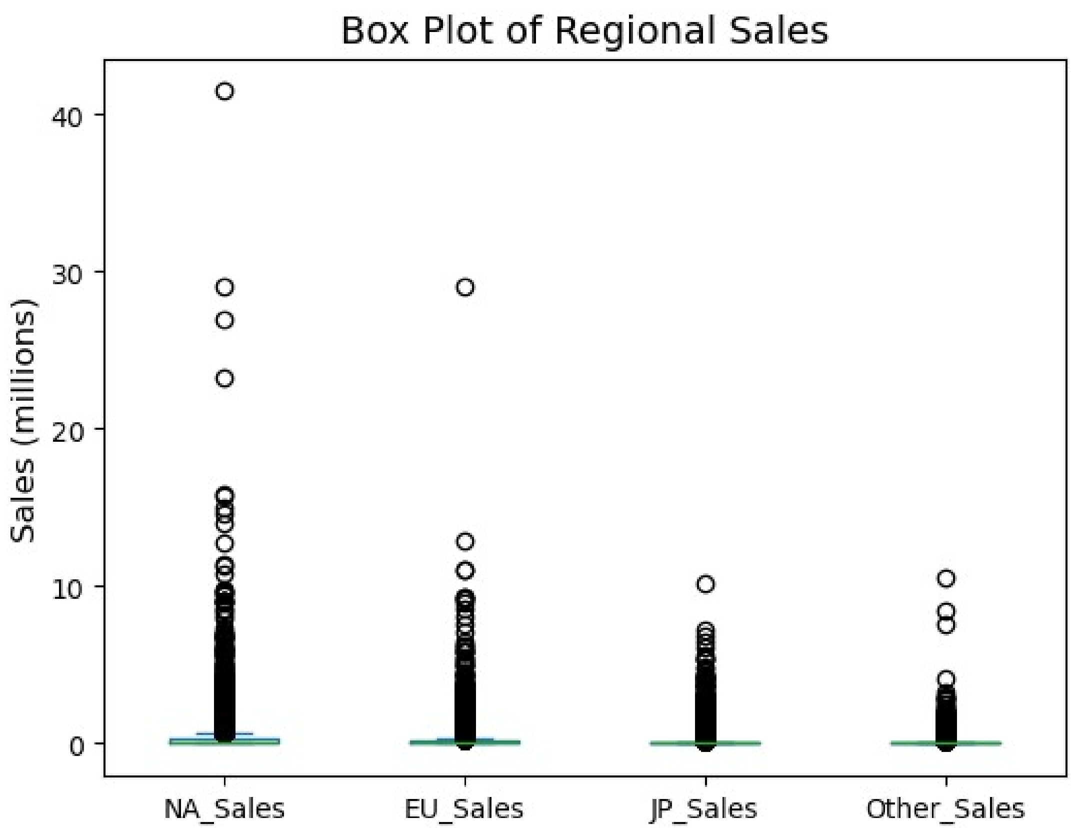



```
# 4. Regional Sales Distribution (NA, EU, JP, Other)
plt.figure(figsize=(10, 6))
regions = ['NA_Sales', 'EU_Sales', 'JP_Sales', 'Other_Sales']
df_cleaned[regions].sum().plot(kind='bar', color=['blue', 'orange',
'green', 'red'], edgecolor='black')
plt.title('Total Sales by Region', fontsize=14)
plt.xlabel('Region', fontsize=12)
plt.ylabel('Total Sales (in millions)', fontsize=12)
plt.xticks(rotation=0)
plt.show()
#NA and EU are the most important markets, with more high-sales
outliers compared to Japan and other regions.
```




```
plt.figure(figsize=(10, 6))
df_cleaned[regions].plot(kind='box')
plt.title('Box Plot of Regional Sales', fontsize=14)
plt.ylabel('Sales (millions)', fontsize=12)
plt.show()
#NA has the highest median sales among all regions, reflecting its
importance in the gaming
```

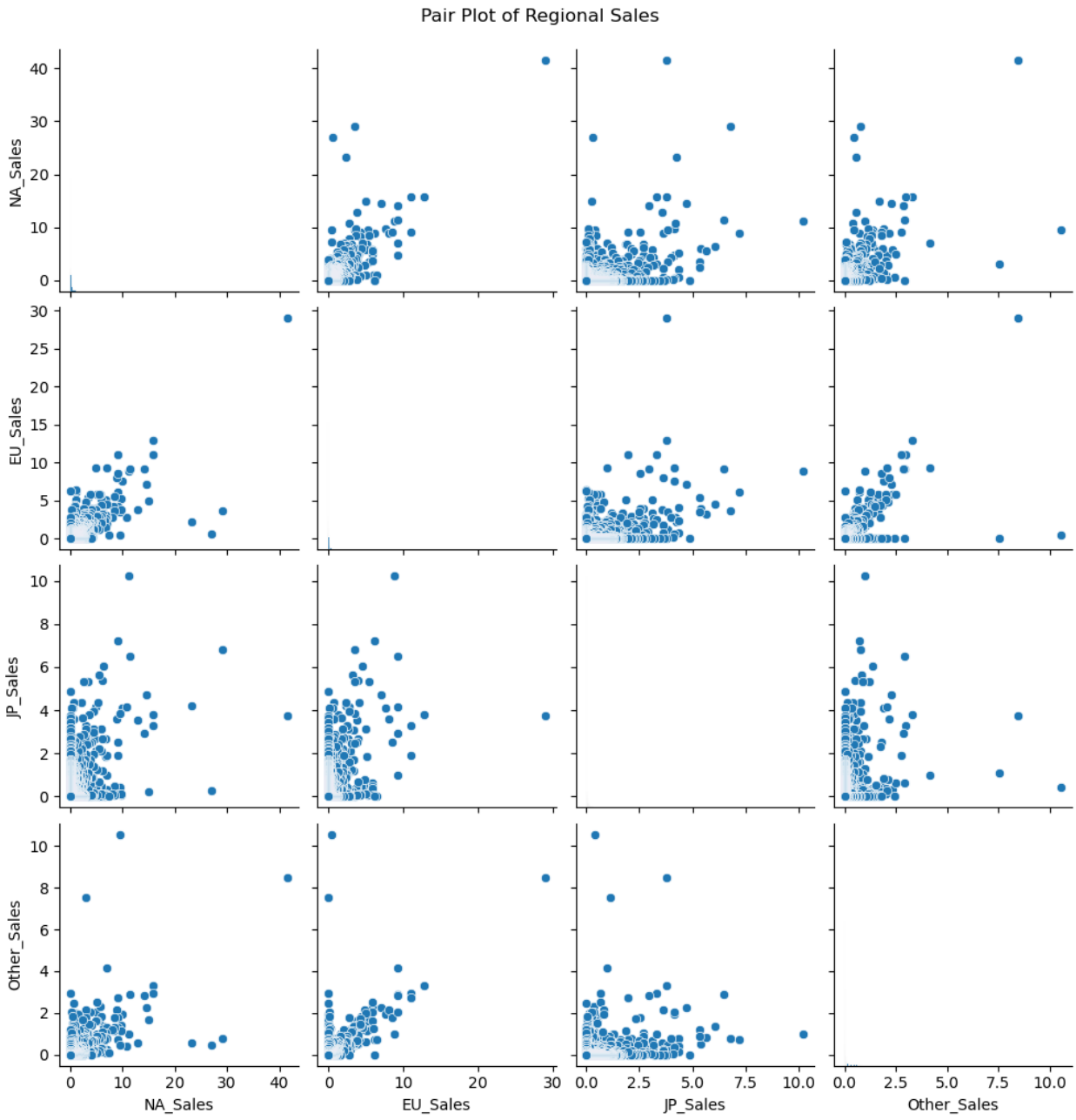
```
<Figure size 1000x600 with 0 Axes>
```




```
import seaborn as sns
sns.pairplot(df_cleaned[regions])
plt.suptitle('Pair Plot of Regional Sales', y=1.02)
plt.show()
#NA Sales vs. EU Sales: Strong positive correlation, suggesting that
games that perform well in North America also tend to perform well in
Europe.
```

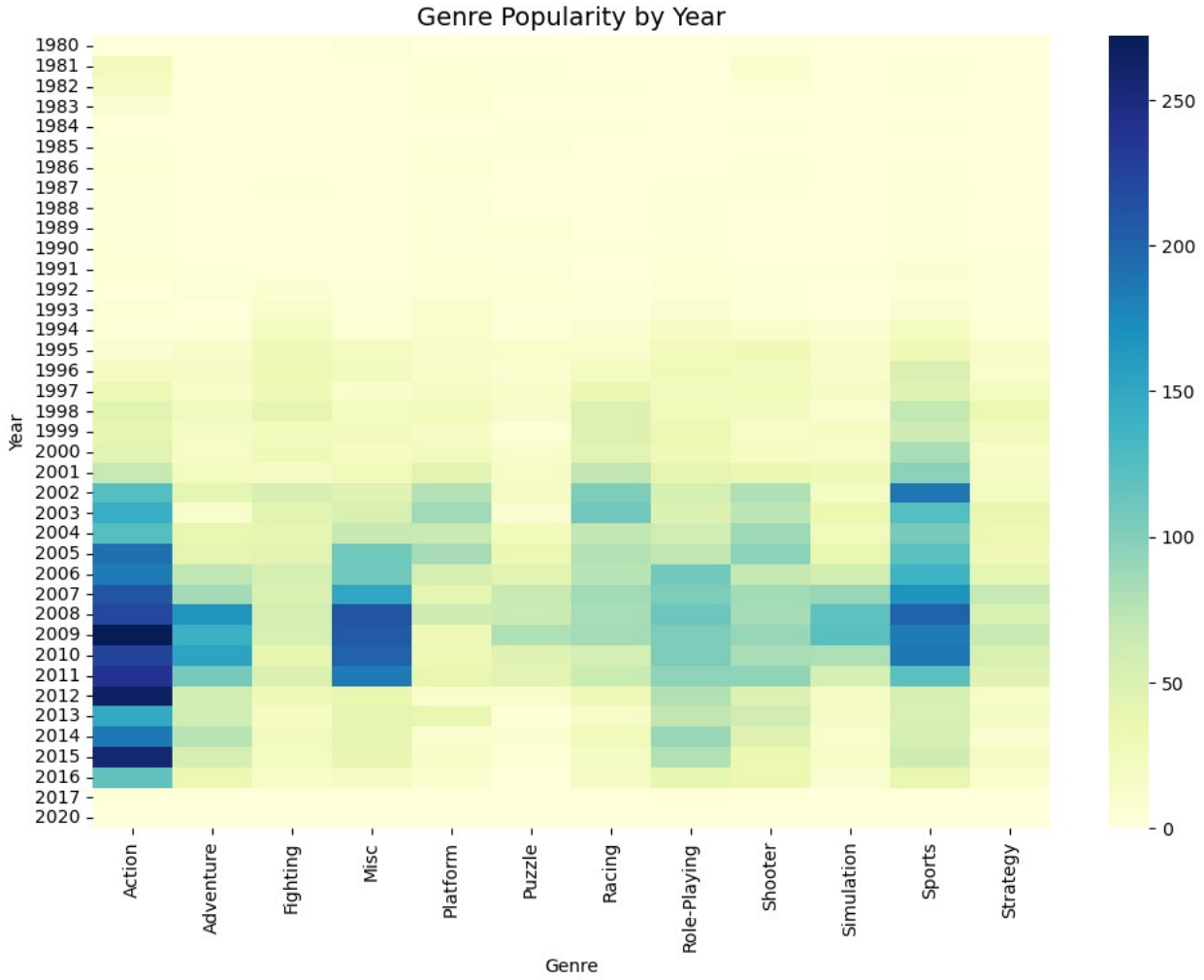
```
C:\Users\MIT\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\MIT\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\MIT\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\MIT\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
```

```
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

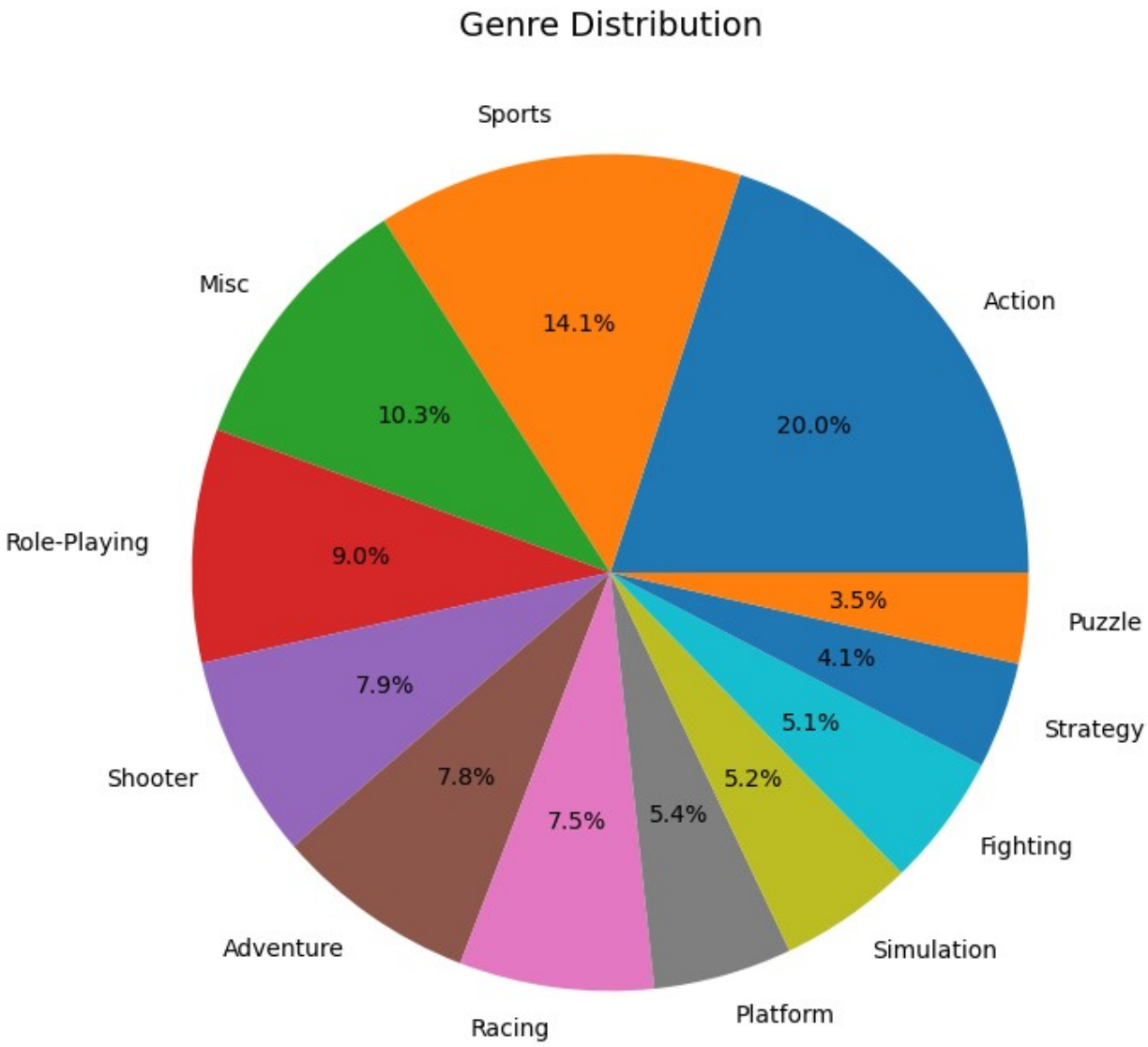



```
genre_year = pd.crosstab(df_cleaned['Year'], df_cleaned['Genre'])
plt.figure(figsize=(12, 8))
sns.heatmap(genre_year, cmap='YlGnBu')
```

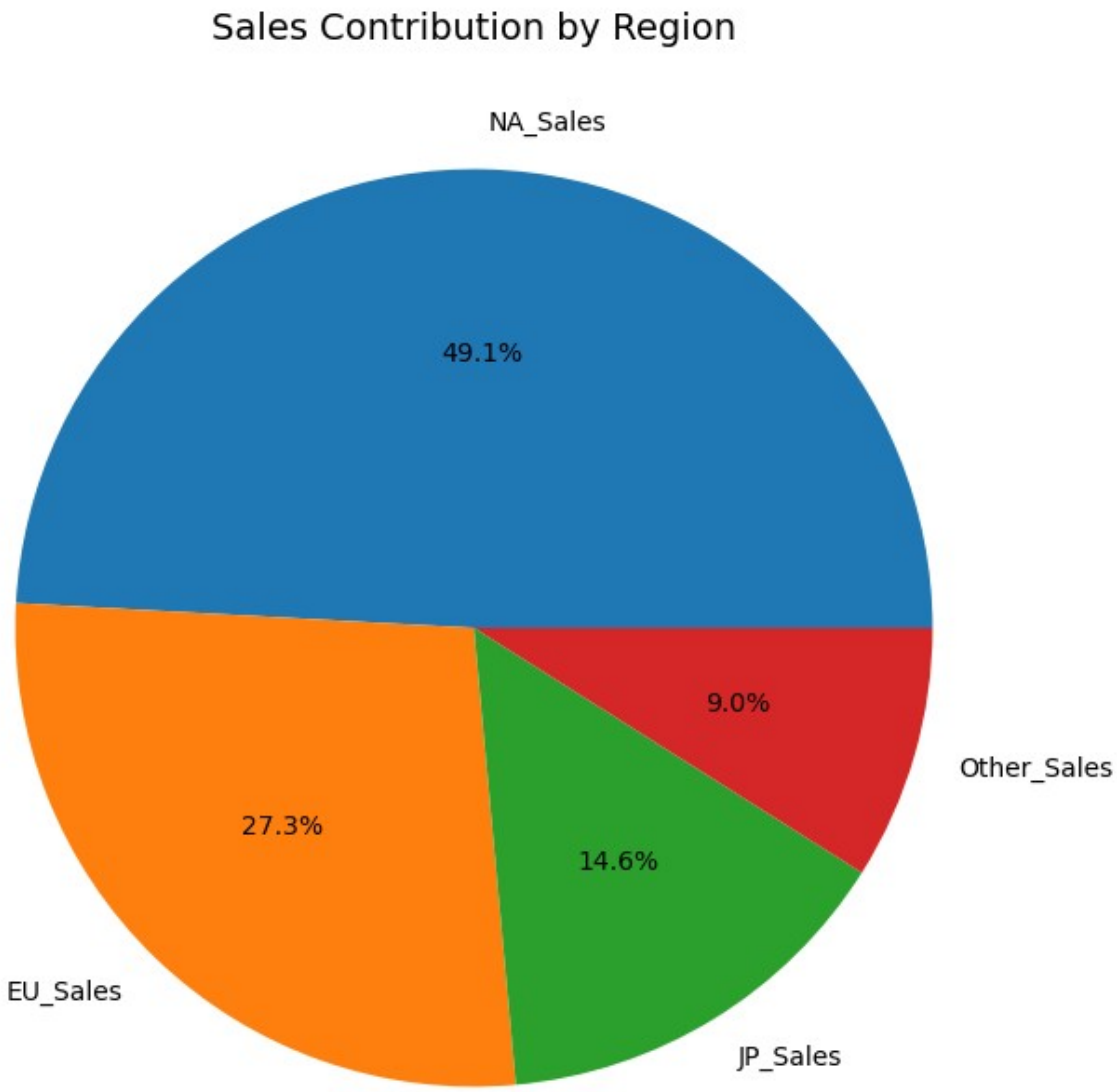
```
plt.title('Genre Popularity by Year', fontsize=14)
plt.show()
#Shooter and Role-Playing (RPG) genres gained prominence in recent
years, likely due to the growth of franchises like Call of Duty and
Final Fantasy.
```



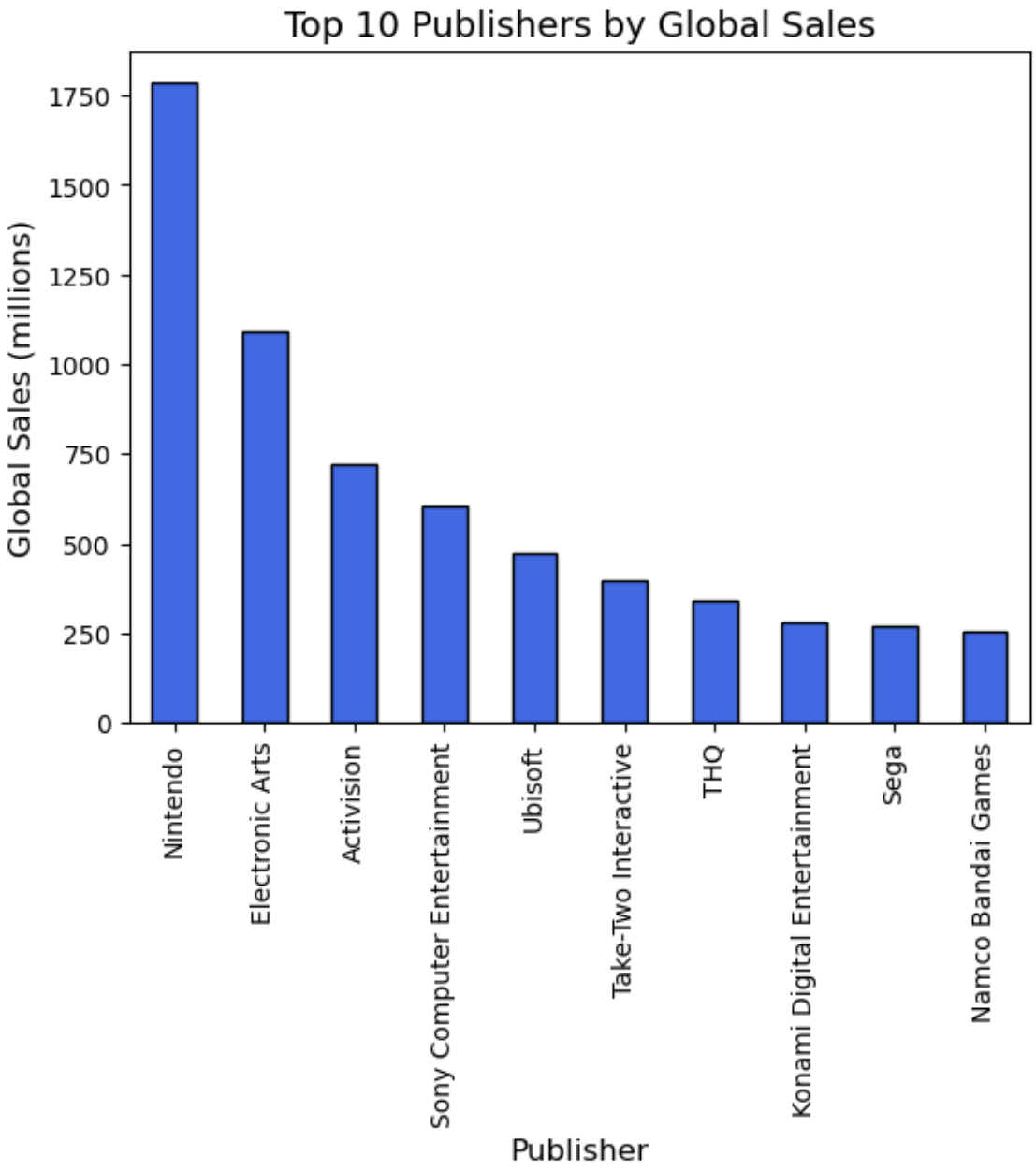

```
df_cleaned['Genre'].value_counts().plot(kind='pie', autopct='%1.1f%%',
figsize=(8, 8))
plt.title('Genre Distribution', fontsize=14)
plt.ylabel('')
plt.show()
```



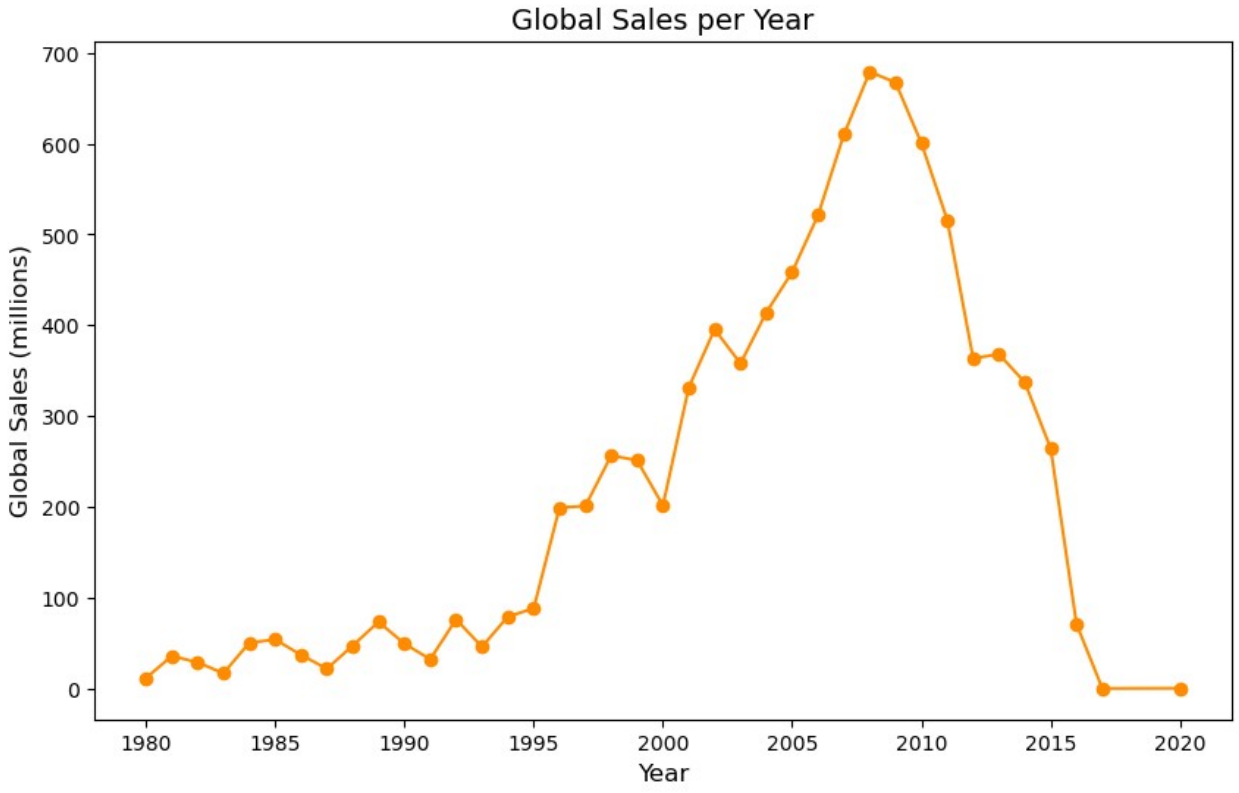

```
df_cleaned[regions].sum().plot(kind='pie', autopct='%1.1f%%',
figsize=(8, 8), labels=regions)
plt.title('Sales Contribution by Region', fontsize=14)
plt.ylabel('')
plt.show()
#North America holds the largest share, indicating the region’s
importance to the gaming industry.
```



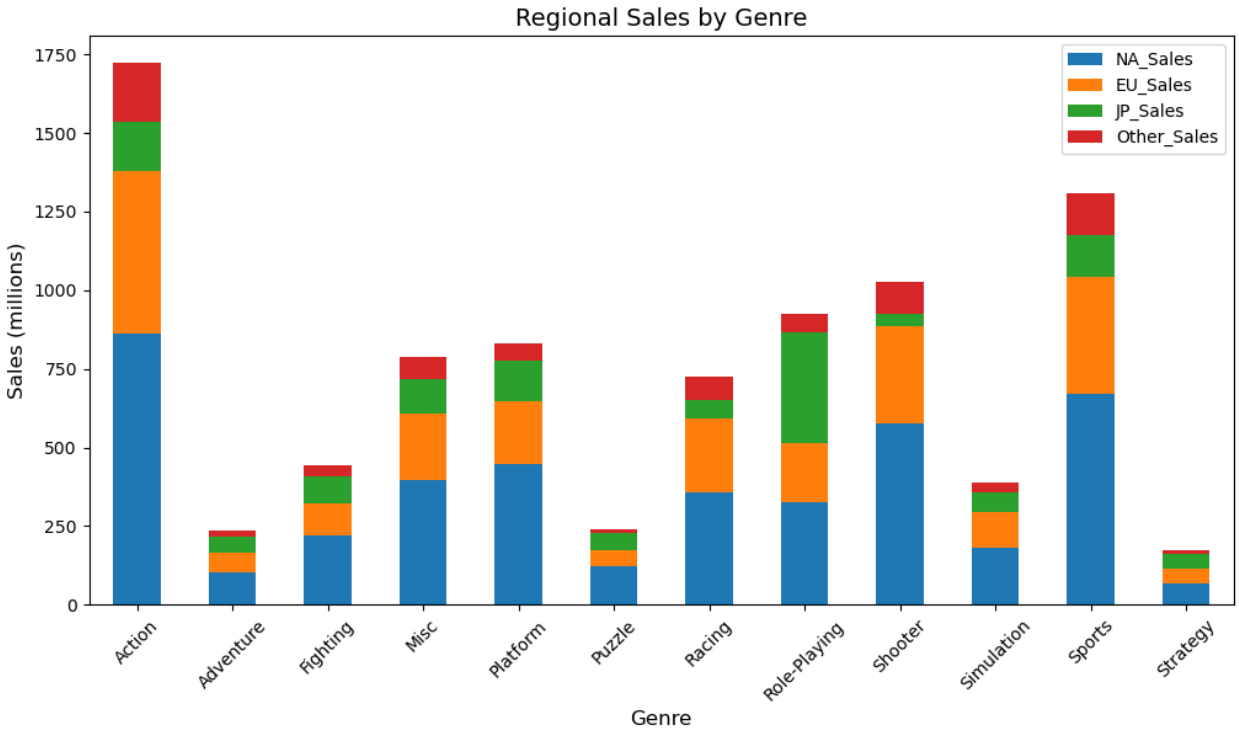

```
top_publishers = df_cleaned.groupby('Publisher')
['Global_Sales'].sum().nlargest(10)
top_publishers.plot(kind='bar', color='royalblue', edgecolor='black')
plt.title('Top 10 Publishers by Global Sales', fontsize=14)
plt.xlabel('Publisher', fontsize=12)
plt.ylabel('Global Sales (millions)', fontsize=12)
plt.show()
```




```
yearly_sales = df_cleaned.groupby('Year')['Global_Sales'].sum()
yearly_sales.plot(kind='line', marker='o', color='darkorange',
figsize=(10, 6))
plt.title('Global Sales per Year', fontsize=14)
plt.xlabel('Year', fontsize=12)
plt.ylabel('Global Sales (millions)', fontsize=12)
plt.show()
```



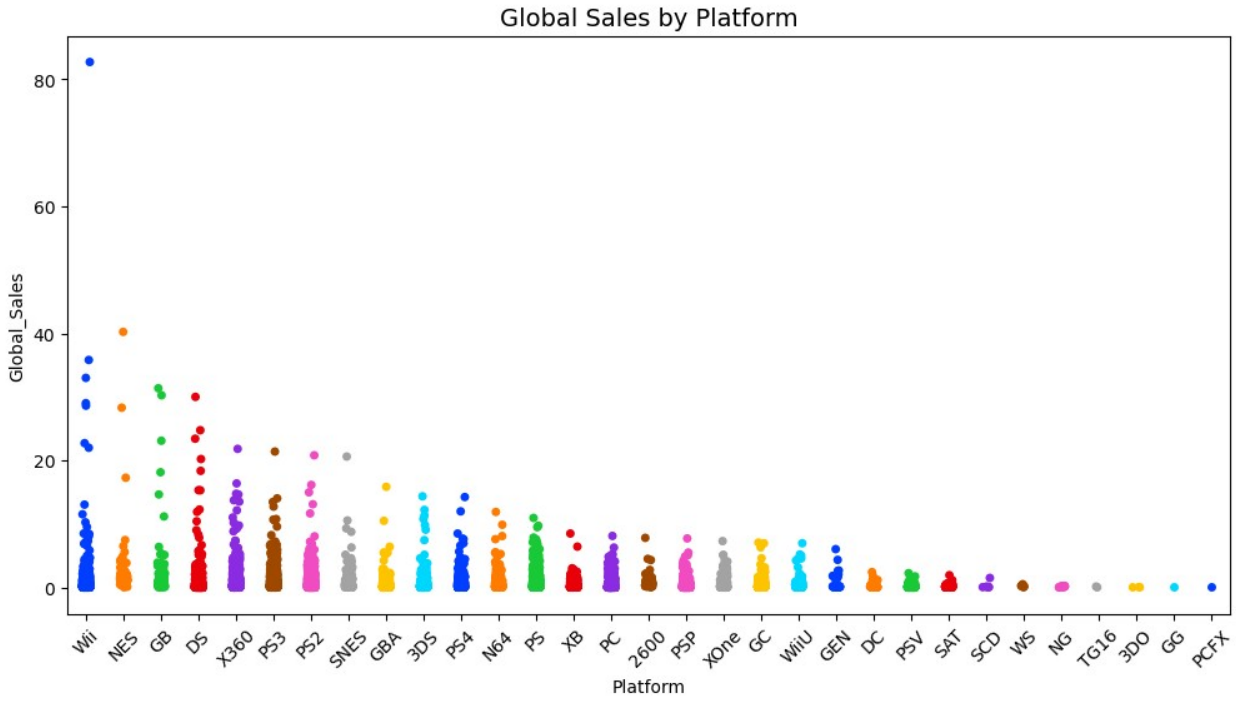

```
genre_sales = df_cleaned.groupby('Genre')[regions].sum()
genre_sales.plot(kind='bar', stacked=True, figsize=(12, 6))
plt.title('Regional Sales by Genre', fontsize=14)
plt.xlabel('Genre', fontsize=12)
plt.ylabel('Sales (millions)', fontsize=12)
plt.xticks(rotation=45)
plt.show()
#Sports games, especially, show a strong presence in EU and NA, driven
by annual franchises like FIFA and Madden.
```



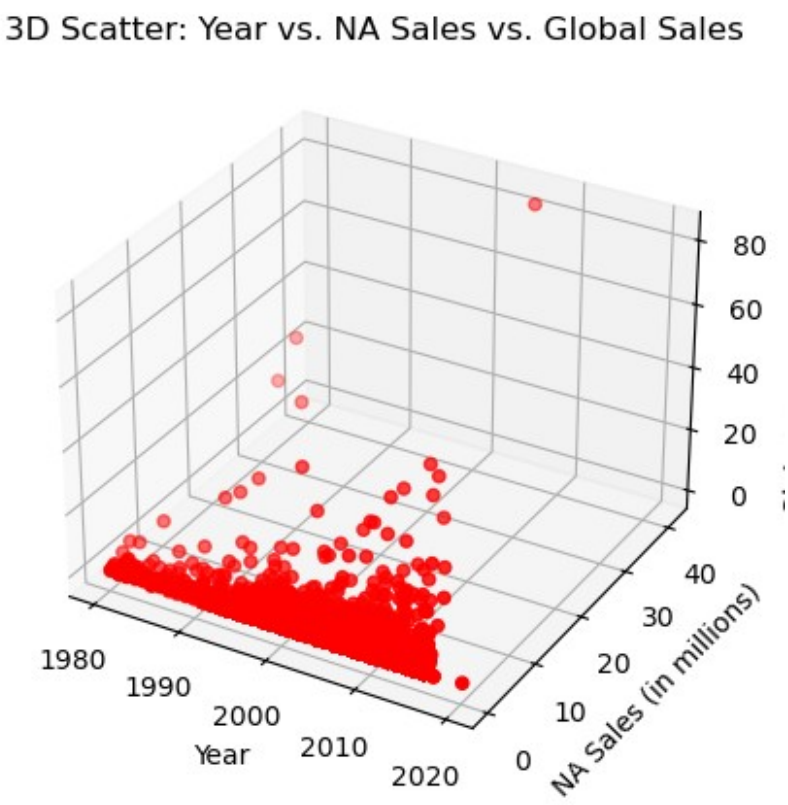

```
from wordcloud import WordCloud

text = ' '.join(df_cleaned['Name'])
wordcloud = WordCloud(width=800, height=400,
background_color='white').generate(text)

plt.figure(figsize=(10, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```


```
plt.figure(figsize=(12, 6))
sns.stripplot(x='Platform', y='Global_Sales', data=df_cleaned,
jitter=True, palette='bright')
plt.title('Global Sales by Platform', fontsize=14)
plt.xticks(rotation=45)
plt.show()
#NES leads global sales, reflecting its long lifespan and extensive
game library.
```

```
C:\Users\MIT\AppData\Local\Temp\ipykernel_15448\3936960473.py:2:
FutureWarning: Passing `palette` without assigning `hue` is
deprecated.
  sns.stripplot(x='Platform', y='Global_Sales', data=df_cleaned,
jitter=True, palette='bright')
C:\Users\MIT\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\MIT\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



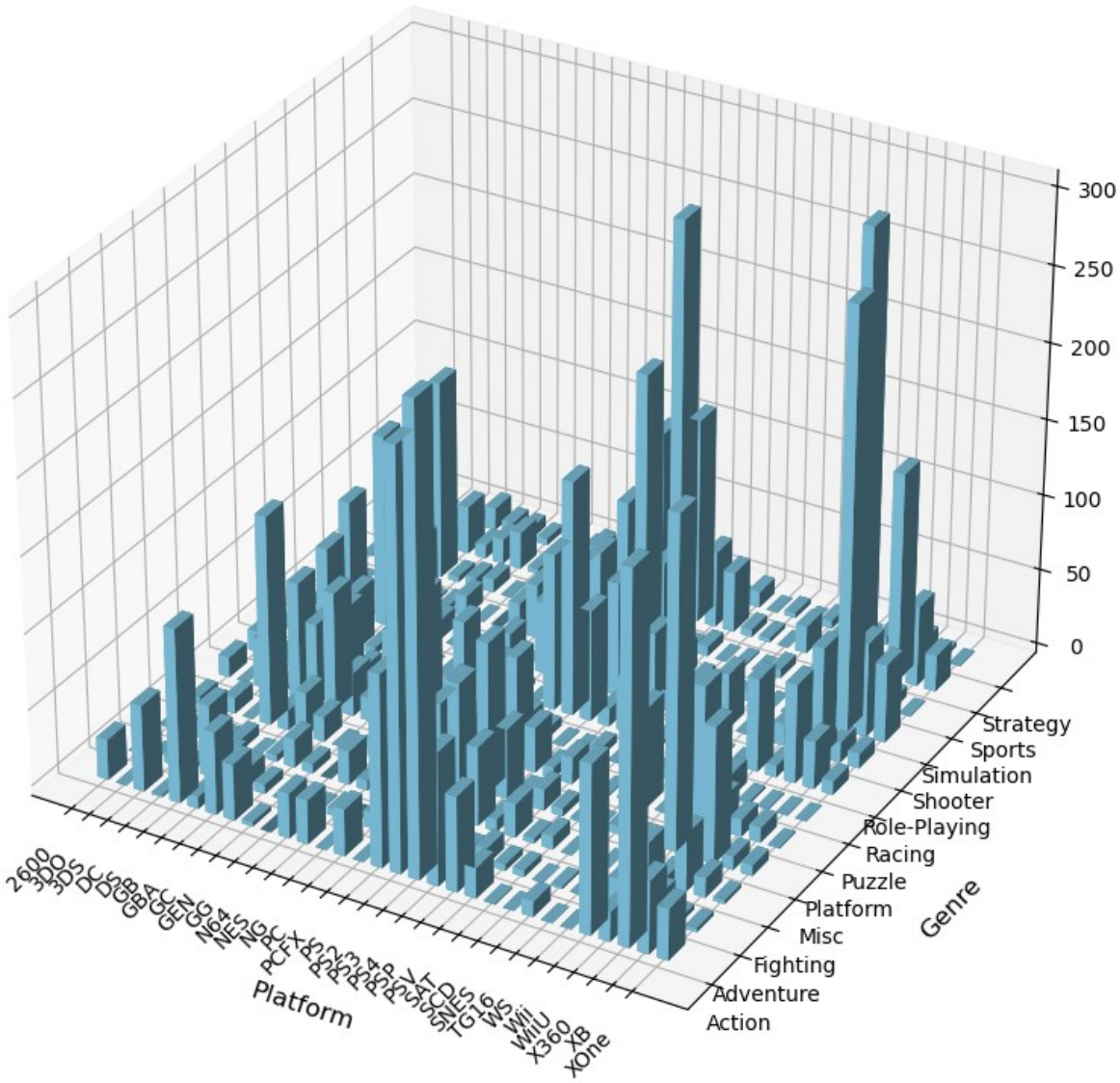

```
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
years = df_cleaned['Year']
na_sales = df_cleaned['NA_Sales']
global_sales = df_cleaned['Global_Sales']
ax.scatter(years, na_sales, global_sales, c='r', marker='o')
ax.set_xlabel('Year')
ax.set_ylabel('NA Sales (in millions)')
ax.set_zlabel('Global Sales (in millions)')
ax.set_title('3D Scatter: Year vs. NA Sales vs. Global Sales')
plt.show()
#Clustered sales in the mid-2000s: The majority of games with high
sales are released between 2005 and 2010, possibly due to the launch
of popular consoles like PlayStation 3, Xbox 360, and Wii
```




```
import numpy as np
import matplotlib.pyplot as plt

# Data Preparation
platform_genre_sales = (
    df_cleaned.groupby(['Platform', 'Genre'])['Global_Sales'].sum()
    .unstack(fill_value=0)
)

platforms = platform_genre_sales.index
genres = platform_genre_sales.columns

# Generate the meshgrid for 3D bar positioning
x = np.arange(len(platforms))
y = np.arange(len(genres))
X, Y = np.meshgrid(x, y)
Z = platform_genre_sales.values.T  # Transpose to align correctly

# Plot Configuration
fig = plt.figure(figsize=(16, 10))
ax = fig.add_subplot(111, projection='3d')

# Plot the 3D bars with optimized spacing
ax.bar3d(
    X.flatten(), Y.flatten(), np.zeros_like(Z.flatten()),
```

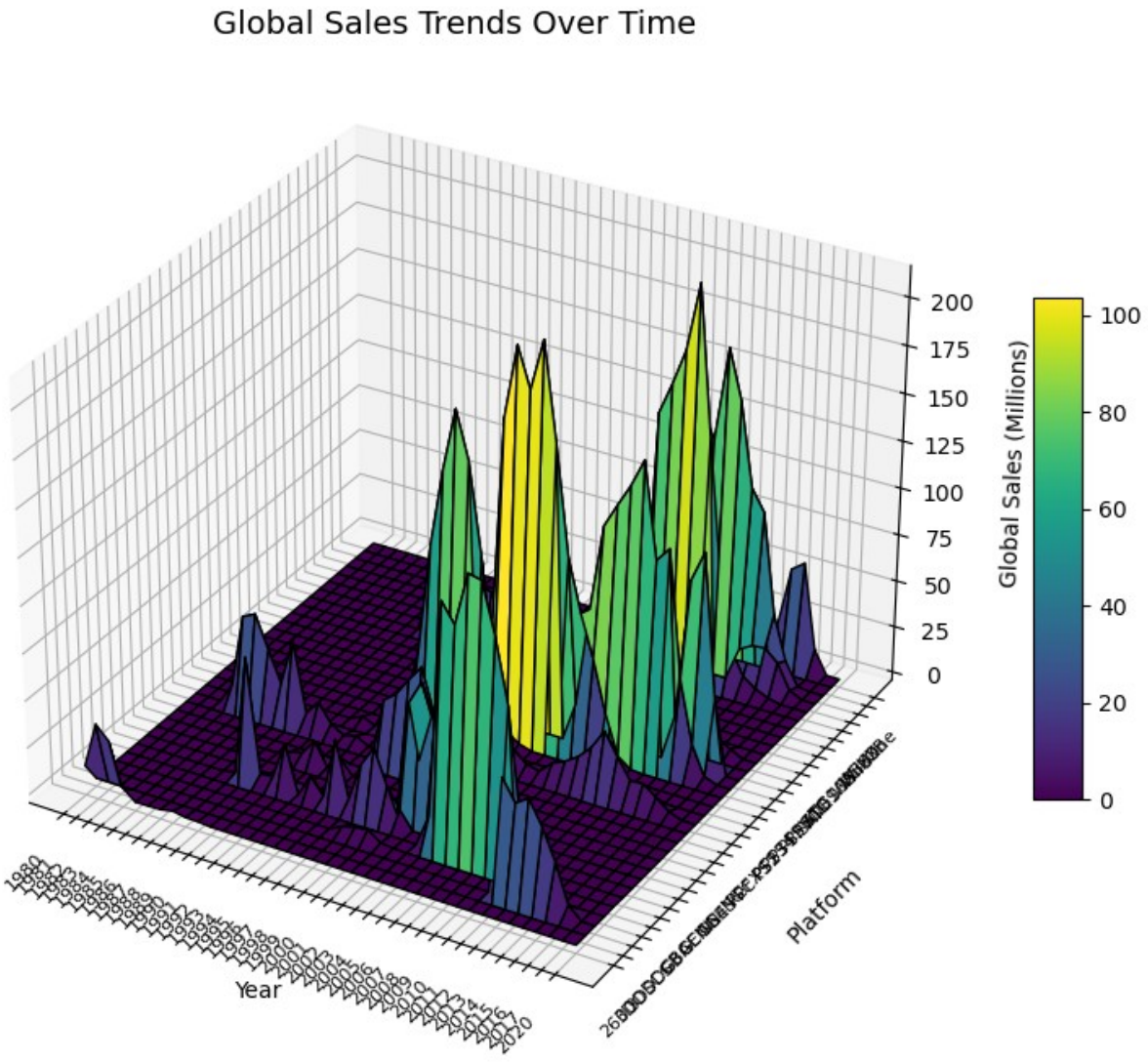
```
    dx=0.6, dy=0.6, dz=Z.flatten(),
    shade=True, color='skyblue'
)

# Adjust Axis Labels
ax.set_xlabel('Platform', labelpad=15, fontsize=12)
ax.set_ylabel('Genre', labelpad=15, fontsize=12)
ax.set_zlabel('Global Sales (Millions)', labelpad=15, fontsize=12)
ax.set_title('Global Sales by Platform and Genre', fontsize=16)

# Configure Tick Labels
ax.set_xticks(x)
ax.set_xticklabels(platforms, rotation=45, ha='right', fontsize=10)

ax.set_yticks(y)
ax.set_yticklabels(genres, rotation=0, ha='center', fontsize=10)

# Add padding to avoid label overlap
fig.tight_layout()

# Display Plot
plt.show()


#Some platforms, such as PS2, Xbox 360, and Wii, have high sales
across multiple genre
```

```
C:\Users\MIT\AppData\Local\Temp\ipykernel_15448\4196763888.py:44:
UserWarning: Tight layout not applied. The left and right margins
cannot be made large enough to accommodate all axes decorations.
  fig.tight_layout()
```



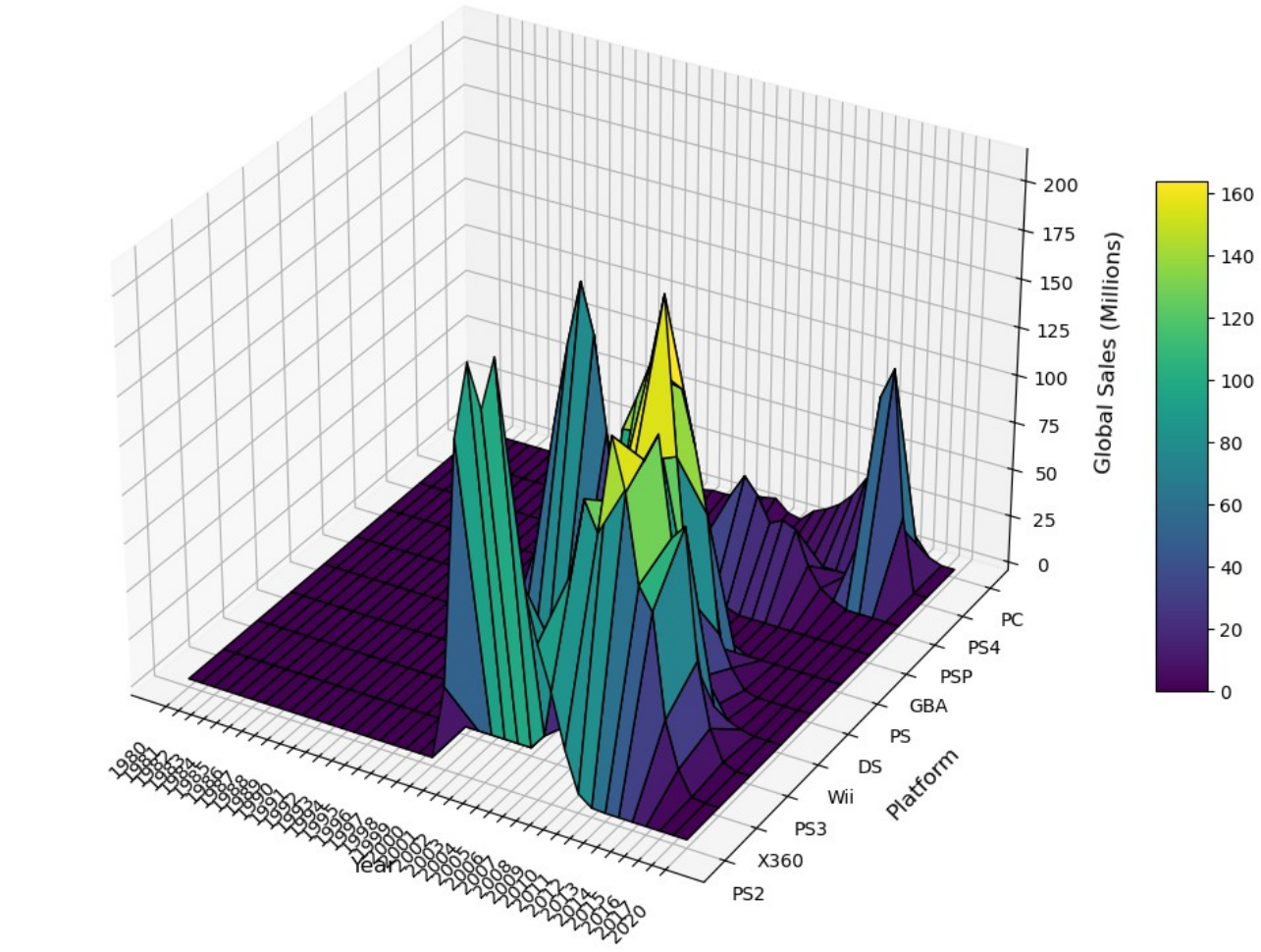

```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm

# Data Preparation
year_sales = df_cleaned.groupby(['Year', 'Platform'])
['Global_Sales'].sum().unstack().fillna(0)
years = year_sales.index
platforms = year_sales.columns
X, Y = np.meshgrid(np.arange(len(years)), np.arange(len(platforms)))
Z = year_sales.values.T
```

```
fig = plt.figure(figsize=(12, 8))
ax = fig.add_subplot(111, projection='3d')

# Surface Plot
surf = ax.plot_surface(X, Y, Z, cmap=cm.viridis, edgecolor='k')

# Axis Labels and Title
ax.set_xlabel('Year', labelpad=10, fontsize=10)
ax.set_ylabel('Platform', labelpad=10, fontsize=10)
ax.set_zlabel('Global Sales (Millions)', labelpad=10, fontsize=10)
ax.set_title('Global Sales Trends Over Time', fontsize=14)

# Adjust Tick Labels Rotation
ax.set_xticks(np.arange(len(years)))
ax.set_xticklabels(years, rotation=45, ha='right', fontsize=8)
ax.set_yticks(np.arange(len(platforms)))
ax.set_yticklabels(platforms, rotation=45, ha='right', fontsize=8)

# Colorbar for Sales Magnitude
fig.colorbar(surf, shrink=0.5, aspect=10)

plt.show()

#Action and Sports dominate sales: These genres generate the highest
global sales, likely due to strong franchises (e.g., FIFA, Call of
Duty).
```




```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
import matplotlib.pyplot as plt
import numpy as np

# Data Preparation
year_sales = df_cleaned.groupby(['Year', 'Platform'])
['Global_Sales'].sum().unstack().fillna(0)

# Limit to top 10 platforms by total global sales
top_platforms = year_sales.sum().nlargest(10).index
year_sales = year_sales[top_platforms]

# Meshgrid for plotting
years = year_sales.index
platforms = year_sales.columns
X, Y = np.meshgrid(np.arange(len(years)), np.arange(len(platforms)))
```

```
Z = year_sales.values.T  # Transpose to align data properly

# Plot Configuration
fig = plt.figure(figsize=(16, 10))
ax = fig.add_subplot(111, projection='3d')

# Surface Plot
surf = ax.plot_surface(X, Y, Z, cmap=cm.viridis, edgecolor='black')

# Axis Labels and Title
ax.set_xlabel('Year', labelpad=10, fontsize=12)
ax.set_ylabel('Platform', labelpad=10, fontsize=12)
ax.set_zlabel('Global Sales (Millions)', labelpad=10, fontsize=12)
ax.set_title('Global Sales Trends Over Time (Top Platforms)',
fontsize=16)

# Adjust Tick Labels
ax.set_xticks(np.arange(len(years)))
ax.set_xticklabels(years, rotation=45, ha='right', fontsize=10)

ax.set_yticks(np.arange(len(platforms)))
ax.set_yticklabels(platforms, rotation=0, ha='center', fontsize=10)

# Colorbar for Sales Magnitude
fig.colorbar(surf, shrink=0.5, aspect=10)

# Tight Layout to Avoid Overlapping
fig.tight_layout()

# Display Plot
plt.show()
#2005–2010 Sales Spike: This period shows the highest sales,
coinciding with the popularity of the 7th generation consoles (e.g.,
PS3, Xbox 360, Wii).

C:\Users\MIT\AppData\Local\Temp\ipykernel_15448\534982345.py:43:
UserWarning: Tight layout not applied. The left and right margins
cannot be made large enough to accommodate all axes decorations.
  fig.tight_layout()
```


Global Sales Trends Over Time (Top Platforms)
Global Sales (Millions)
200
175
150
125
100
75
50
25
0
160
140
120
100
80
60
40
20
0
Platform
PC
PS4
PSP
GBA
PS
DS
Wii
PS3
X360
PS2
Year