Data Viz 1

Content

- Intro to Matplotlib & Seaborn
- Tencent Use Case
- Anatomy of Matplotlib
- Components of a Matplotlib plot
- Univariate Data Visualization
  - Categorical
    - Bar chart
    - Countplot
    - Pie chart
  - Continous
    - Histogram
    - KDE plot
    - Box and Whiskers plot

Plots Presentation:

https://docs.google.com/presentation/d/1DkLTjTe6YmGbDHtr4v9Jso553DlCuP3cfSnwvUN1ı
usp=sharing

Importing Matplotlib & Seaborn

In case of `matplotlib`,

- We don't need to import the entire library but just its sub-module `pyplot`.
- We'll use the alias name `plt`.

What is `pyplot`?

- `pyplot` is a **sub-module for visualization** in `matplotlib`.
- Think of it as a **high-level API** which **makes plotting an easy task**.
- Data Scientists stick to using `pyplot` only unless they want to create **something totally new.

For `seaborn`,

- We will be importing the whole seaborn library as alias `sns`.

What is seaborn?

Seaborn is another visualization library which uses Matplotlib in the backend for plotting.

What is the major difference then between both matplotlib and seaborn?

- `Seaborn` is built on the top of `Pandas` and `Matplotlib`.
- Seaborn uses **fascinating themes** and **reduces number of code lines** by doing a lot of work in the backend.
- While matplotlib is used to **plot basic plots and add more functionlaity** on top of that

As we proceed through the lecture, we will see the difference between both the libraries.

In [1]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

Before we dive into learning these libraries, lets answer some general questions.

Why do even we need to visualize data?

- **Exploratory** - I can't see certain patterns just by crunching numbers (avg, rates, %ages).
- **Explanatory** - I have the numbers crunches and insights ready, but I'd like a visual art for storytelling.

What do we already know?

Data

- Rows: Samples, Data-points, Records
- Columns: Features, Variables

At the fundamental level, we have two types of data:

- Numerical/Continous
- Categorical

Categorical can be further divided into:

- **Ordinal:** Categorical data with an order (e.g. low, medium, high)
- **Non-ordinal/nominal:** Categorical data without any order (e.g. Male/Female)

Video Games Analysis

You are a Data Scientist at "Tencent Games".

You need to analyze what kind of games the company should create in order to perform better in the market.

In [2]:
```
import pandas as pd
import numpy as np
```

In [3]:
```
data = pd.read_csv('final.csv')
data.head()
```

Out[3]:

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sa
0	2061	1942	NES	1985.0	Shooter	Capcom	4.569217	3.033887	3.439
1	9137	¡Shin Chan Flipa en colores!	DS	2007.0	Platform	505 Games	2.076955	1.493442	3.033
2	14279	.hack: Sekai no Mukou ni + Versus	PS3	2012.0	Action	Namco Bandai Games	1.145709	1.762339	1.493
3	8359	.hack//G.U. Vol.1//Rebirth	PS2	2006.0	Role-Playing	Namco Bandai Games	2.031986	1.389856	3.228
4	7109	.hack//G.U. Vol.2//Reminisce	PS2	2006.0	Role-Playing	Namco Bandai Games	2.792725	2.592054	1.440

Notice that,

- columns like `Platform`, `Genre` are Categorical
- columns like `NA_Sales`, `Global_Sales`, `Rank` are Continuous

Furthermore,

- `Platform` is of nominal type (no proper order between the categories)
- `Year` is of ordinal type (an order exists between the categories)

Introduction to Matplotlib

How to create a basic plot using `plt`?

Let's say we want to draw a curve passing through 3 points:

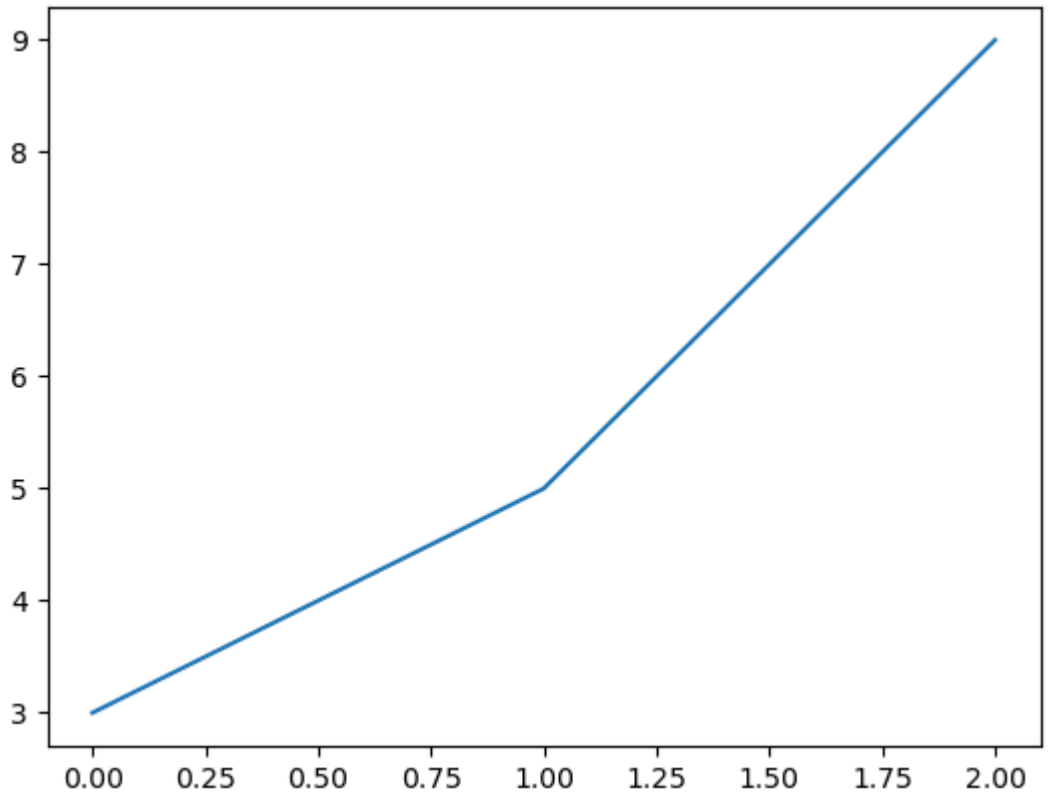
- (0, 3)
- (1, 5)
- (2, 9)

How can we draw a curve using `matplotlib`?

- by using the `plt.plot()` function

In [4]:
```
x_val = [0, 1, 2]
y_val = [3, 5, 9]
plt.plot(x_val, y_val)
```

Out[4]:
```
[<matplotlib.lines.Line2D at 0x159767d10>]
```

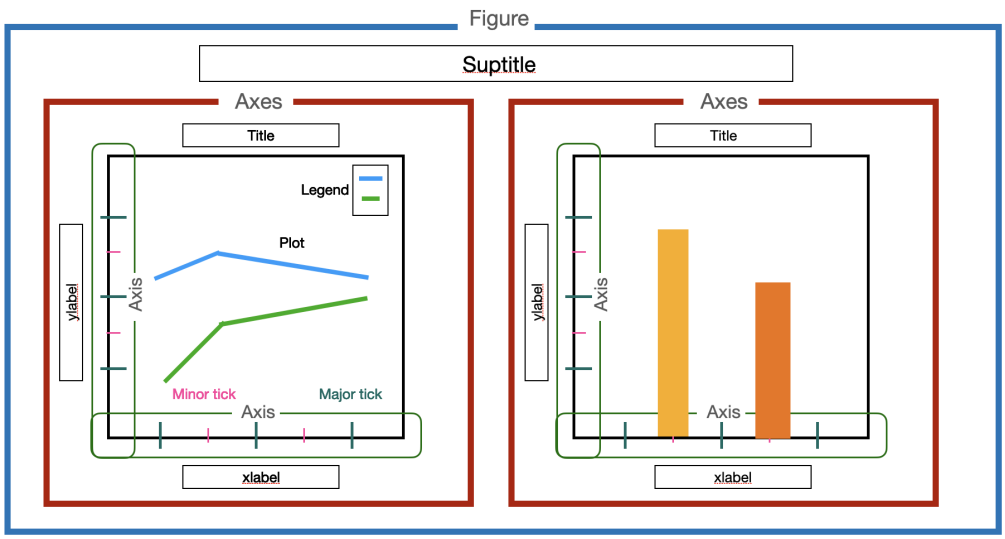


What can we observe from this plot?

- `plt.plot()` automatically decided the scale of the plot.
- It also prints the **type of object** i.e. `matplotlib.lines.Line2D`

While this command decided a lot of things for you, you can customise each of these by understanding the **components of a matplotlib plot**.

Anatomy of Matplotlib



Woah! There is a lot of information in this image. Let's understand them one at a time.

Components of a Matplotlib plot

- Figure: The **overall window** or page that everything is drawn on.
  - You can create multiple independent Figures in Jupyter.
  - If you run the code in terminal, separate windows will pop-up.
- Axes: You can add multiple **Axes** to the Figure, which represents a plot.
- **Axis**: Simply the `x-axis` and `y-axis`
- **Axes**: It is the **area** on which the **data is plotted** with functions such as `plot()`.
  - **x-label**: Name of x-axis
  - **y-label**: Name of y-axis
- **Major ticks**:
  - Subdivides the axis into major units.
  - They appear by default during plotting.
- **Minor ticks**:
  - Subdivides the major tick units.
  - They are by default hidden and can be toggled on.
- **Title**: Title of each plot **(Axes)**
- **Subtitle**: The common title of all the plots.
- **Legend**:
  - Describes the elements in the plot.
  - Blue and Green curves in this case.

These are the major components of a matplotlib plot.

How to choose the right plot?

Firstly, it depends on the what is your question of interest.

When the question is clear

- How many variables are involved?
- Whether the variable(s) are numerical or categorical?

How many variables are involved?

- 1 Variable $\rightarrow$ Univariate Analysis
- 2 Variables $\rightarrow$ Bivariate Analysis
- 3+ Variables $\rightarrow$ Multivariate Analysis

What are the possible cases?

Univariate

- Numerical

- Categorical

Bivariate

- Numerical-Numerical
- Numerical-Categorical
- Categorical-Categorical

Multivariate

Let's start with these and then we can generalize.

- Numerical-Numerical-Categorical
- Categorical-Categorical-Numerical
- Categorical-Categorical-Categorical
- Numerical-Numerical-Numerical

Univariate Data Visualization - Categorical Data

What kind of questions we may want to ask for a categorical variable?

- What is the Distribution/Frequency of the data across different categories?
- What proportion does a particular category constitutes?

...and so on

Let's take the categorical column "Genre".

How can we find the top-5 genres?

In [5]:
```
cat_counts = data['Genre'].value_counts()
cat_counts
```

Out[5]:
```
Action          3316
Sports          2400
Misc            1739
Role-Playing    1488
Shooter         1310
Adventure       1286
Racing          1249
Platform         886
Simulation       867
Fighting         848
Strategy         681
Puzzle           582
Name: Genre, dtype: int64
```

What kind of plot can we use to visualize this information?

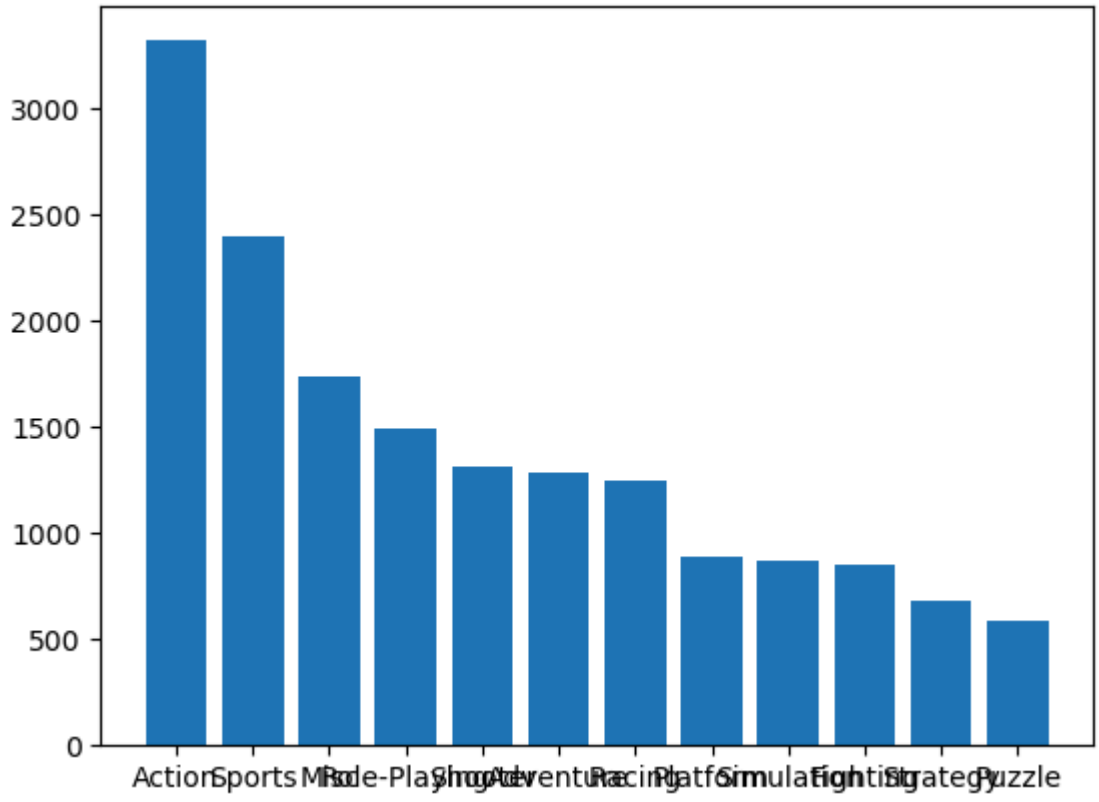
- We can perhaps plot categories on X-axis and their corresponding frequencies on Y-axis.
- This is called a `Bar Chart` or a `Count Plot`.

Bar Chart

- We can draw a bar plot using `plt.bar()`.
- The data is binned here into categories.

In [6]:
```
x_bar=cat_counts.index
y_bar=cat_counts
plt.bar(x_bar,y_bar)
```

Out[6]:
```
<BarContainer object of 12 artists>
```



The names seem to be overlapping.

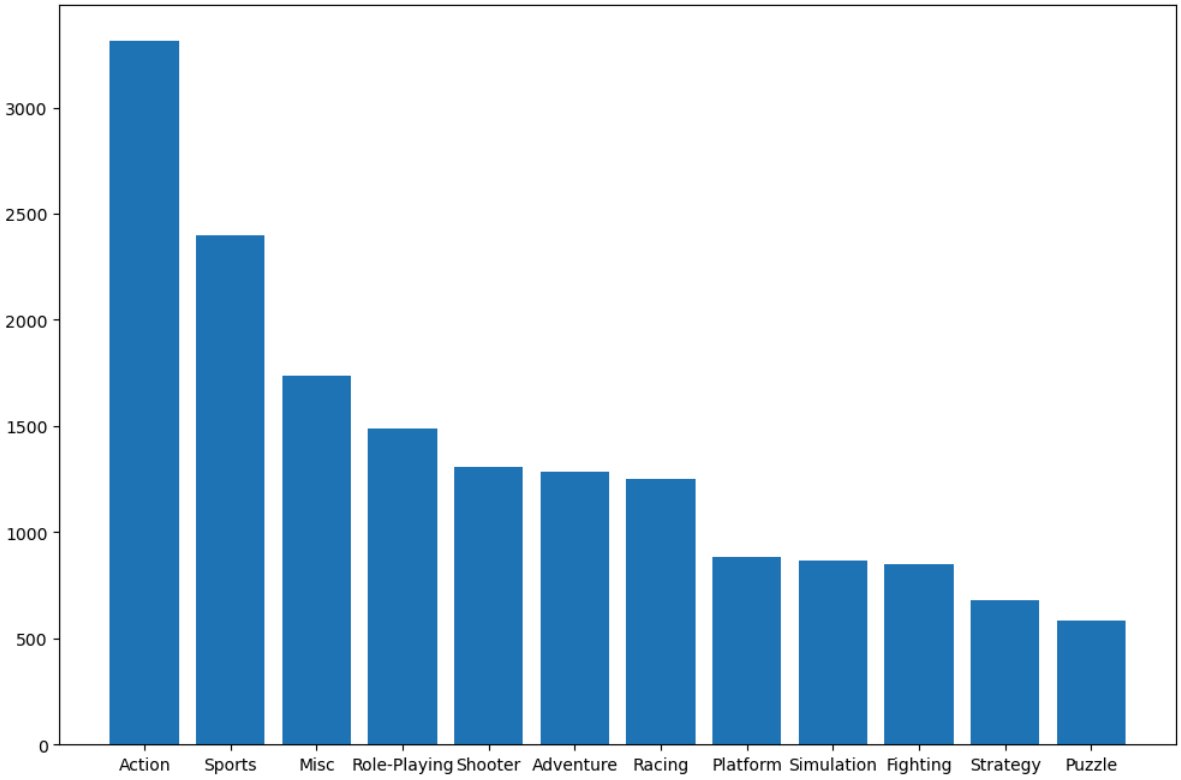
How can we handle overlapping labels?

1. Decrease the font size (not preferred)
2. Increase the figure size
3. Rotate the labels

How can we change the plot size?

In [8]:
```
plt.figure(figsize=(12,8))
plt.bar(x_bar,y_bar)
```

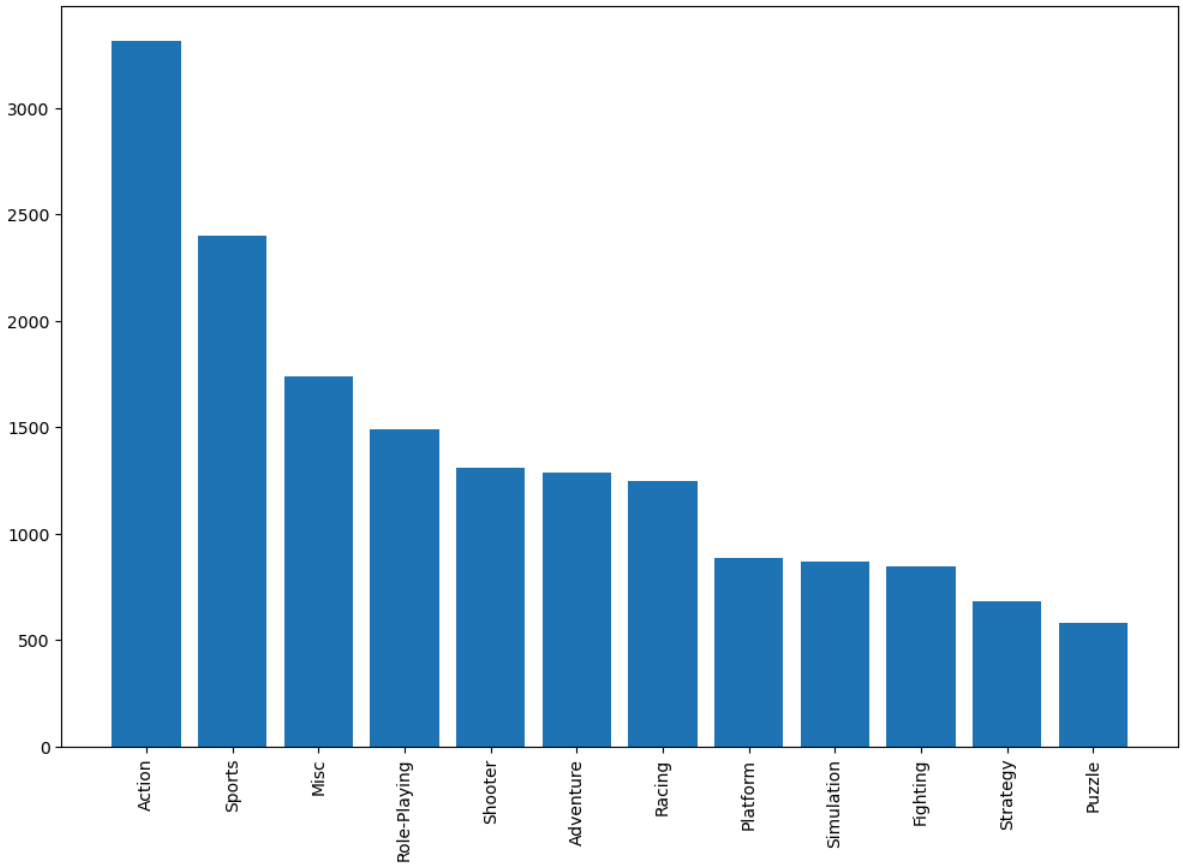
Out[8]:
```
<BarContainer object of 12 artists>
```



How can we rotate the tick labels, also increase the fontsize of the same?

In [9]:
```
plt.figure(figsize=(12,8))
plt.bar(x_bar,y_bar)
plt.xticks(rotation=90, fontsize=10)
```

Out[9]:
```
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11],
 [Text(0, 0, 'Action'),
  Text(1, 0, 'Sports'),
  Text(2, 0, 'Misc'),
  Text(3, 0, 'Role-Playing'),
  Text(4, 0, 'Shooter'),
  Text(5, 0, 'Adventure'),
  Text(6, 0, 'Racing'),
  Text(7, 0, 'Platform'),
  Text(8, 0, 'Simulation'),
  Text(9, 0, 'Fighting'),
  Text(10, 0, 'Strategy'),
  Text(11, 0, 'Puzzle')])
```

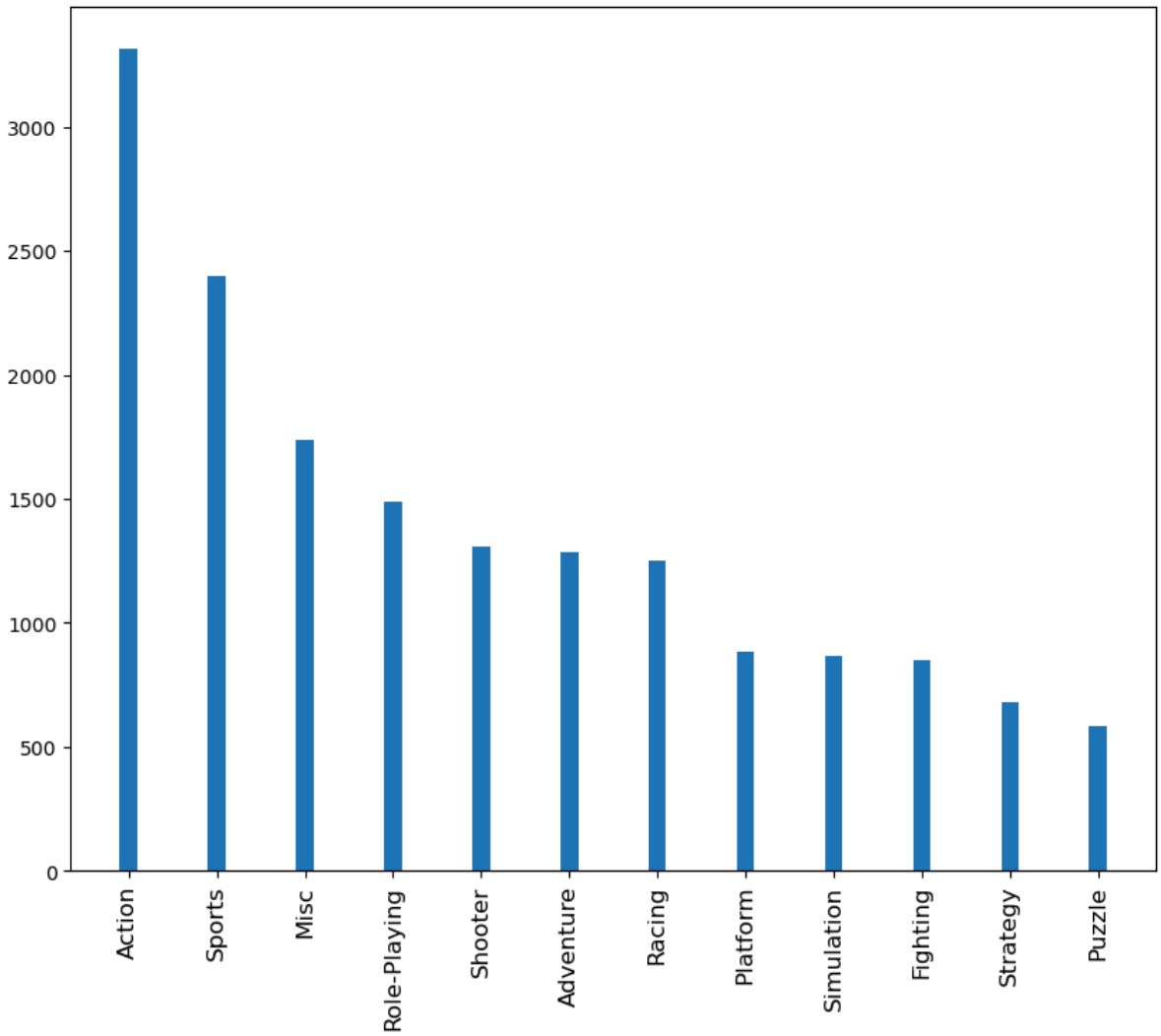


If you notice, the width of each bar is **1**.

Can we also change the width of these bars?

In [10]:
```
plt.figure(figsize=(10,8))
plt.bar(x_bar,y_bar,width=0.2)
plt.xticks(rotation=90, fontsize=12)
```

Out[10]:
```
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11],
 [Text(0, 0, 'Action'),
  Text(1, 0, 'Sports'),
  Text(2, 0, 'Misc'),
  Text(3, 0, 'Role-Playing'),
  Text(4, 0, 'Shooter'),
  Text(5, 0, 'Adventure'),
  Text(6, 0, 'Racing'),
  Text(7, 0, 'Platform'),
  Text(8, 0, 'Simulation'),
  Text(9, 0, 'Fighting'),
  Text(10, 0, 'Strategy'),
  Text(11, 0, 'Puzzle')])
```

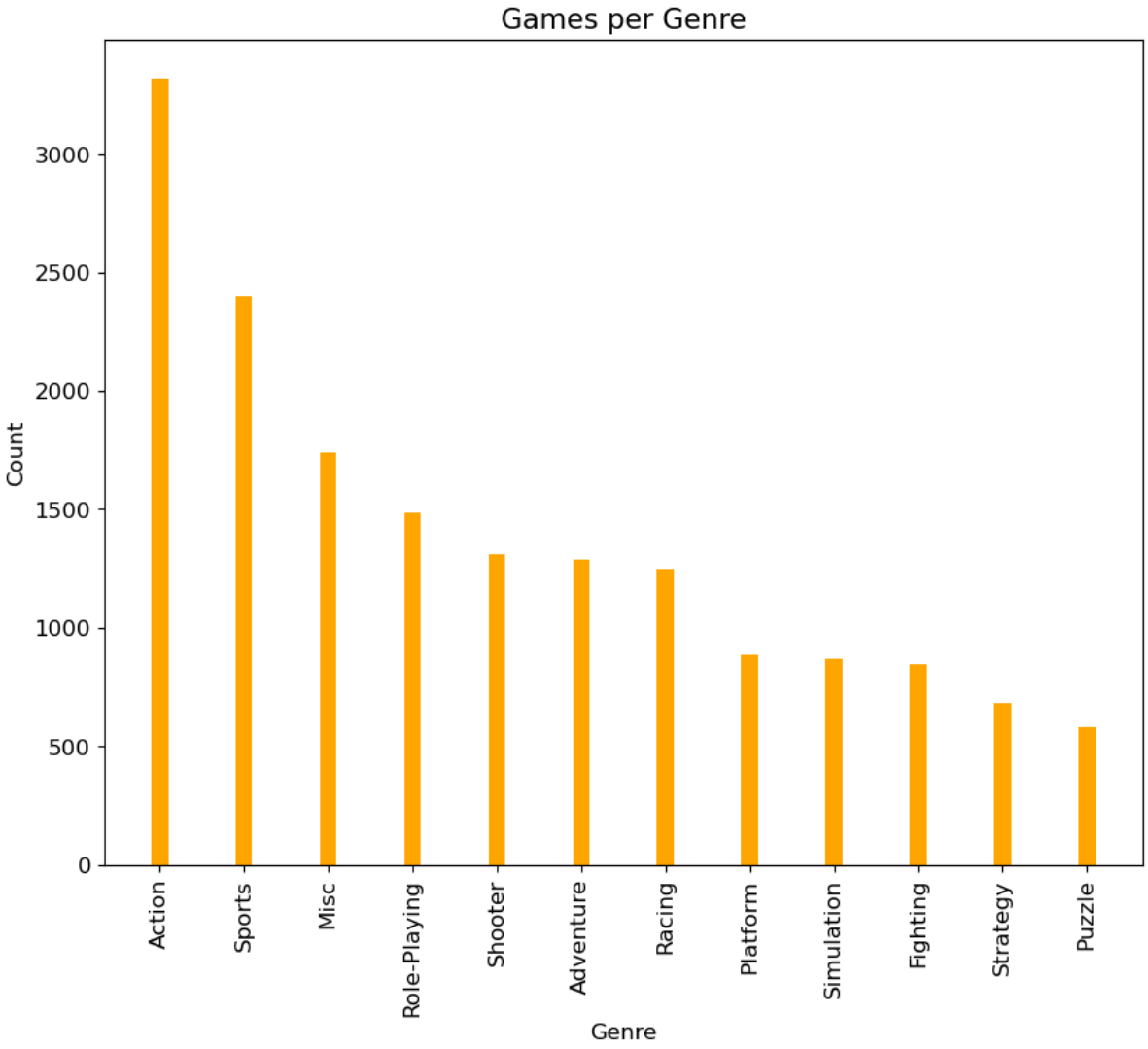


What about any adding some styling to the bars?

- We can **change the colour of bars**
- We can add a **title to the axes**
- We can also **add x and y labels**

In [11]:
```
plt.figure(figsize=(10,8))
plt.bar(x_bar,y_bar,width=0.2,color='orange')
plt.title('Games per Genre',fontsize=15)
plt.xlabel('Genre',fontsize=12)
plt.ylabel('Count',fontsize=12)
plt.xticks(rotation = 90, fontsize=12)
plt.yticks(fontsize=12)
```

Out[11]:
```
(array([   0.,  500., 1000., 1500., 2000., 2500., 3000., 3500.]),
 [Text(0, 0.0, '0'),
  Text(0, 500.0, '500'),
  Text(0, 1000.0, '1000'),
  Text(0, 1500.0, '1500'),
  Text(0, 2000.0, '2000'),
  Text(0, 2500.0, '2500'),
  Text(0, 3000.0, '3000'),
  Text(0, 3500.0, '3500')])
```



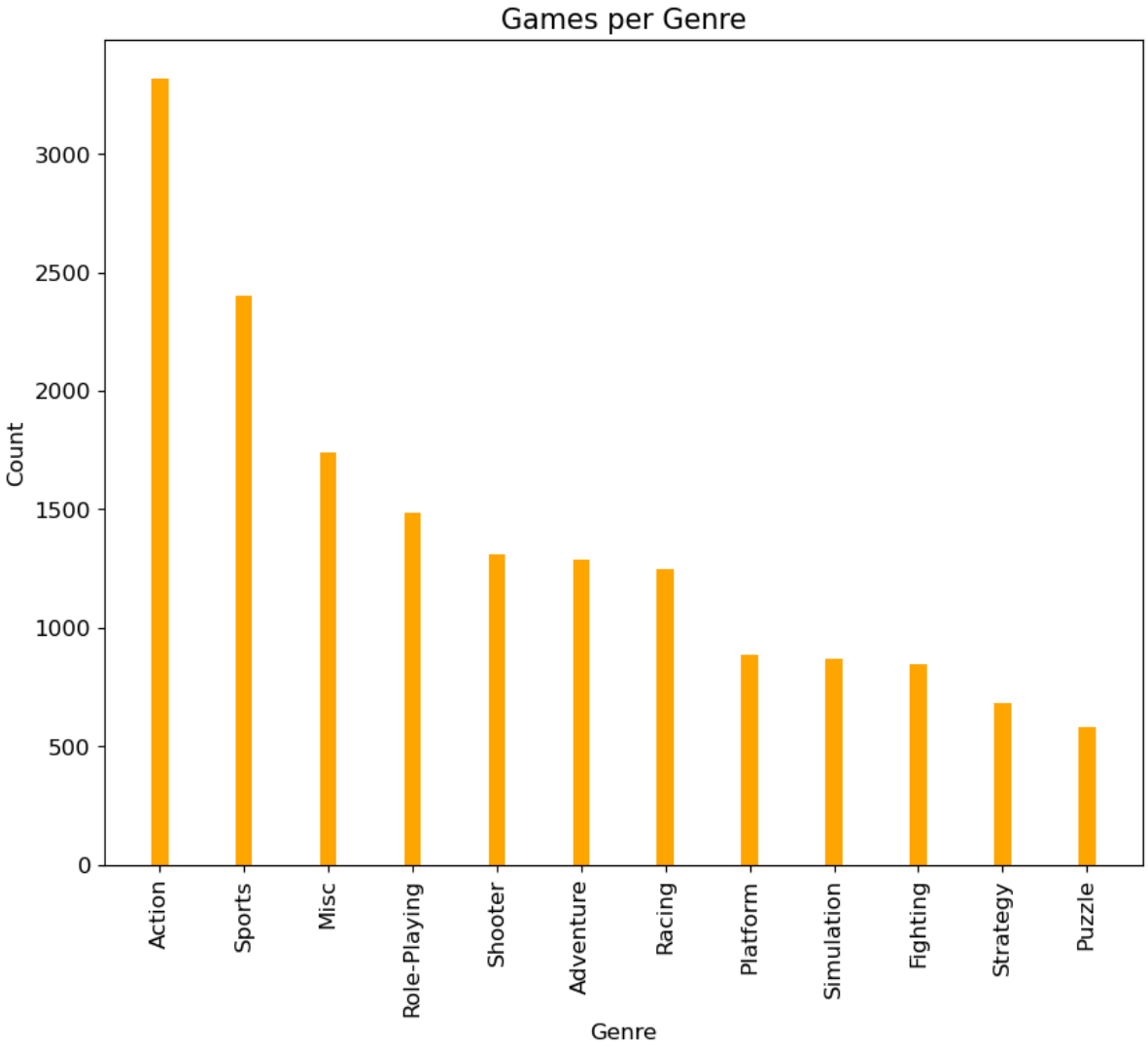
If you notice, there's some text which is always printed before the plots.

This contains the data information of the plot.

How can we remove the text printed before the plot?

- by using `plt.show()` at the end

In [12]:
```
plt.figure(figsize=(10,8))
plt.bar(x_bar,y_bar,width=0.2,color='orange')
plt.title('Games per Genre',fontsize=15)
plt.xlabel('Genre',fontsize=12)
plt.ylabel('Count',fontsize=12)
plt.xticks(rotation = 90, fontsize=12)
plt.yticks(fontsize=12)
plt.show()
```



How can we draw a bar chart in Seaborn?

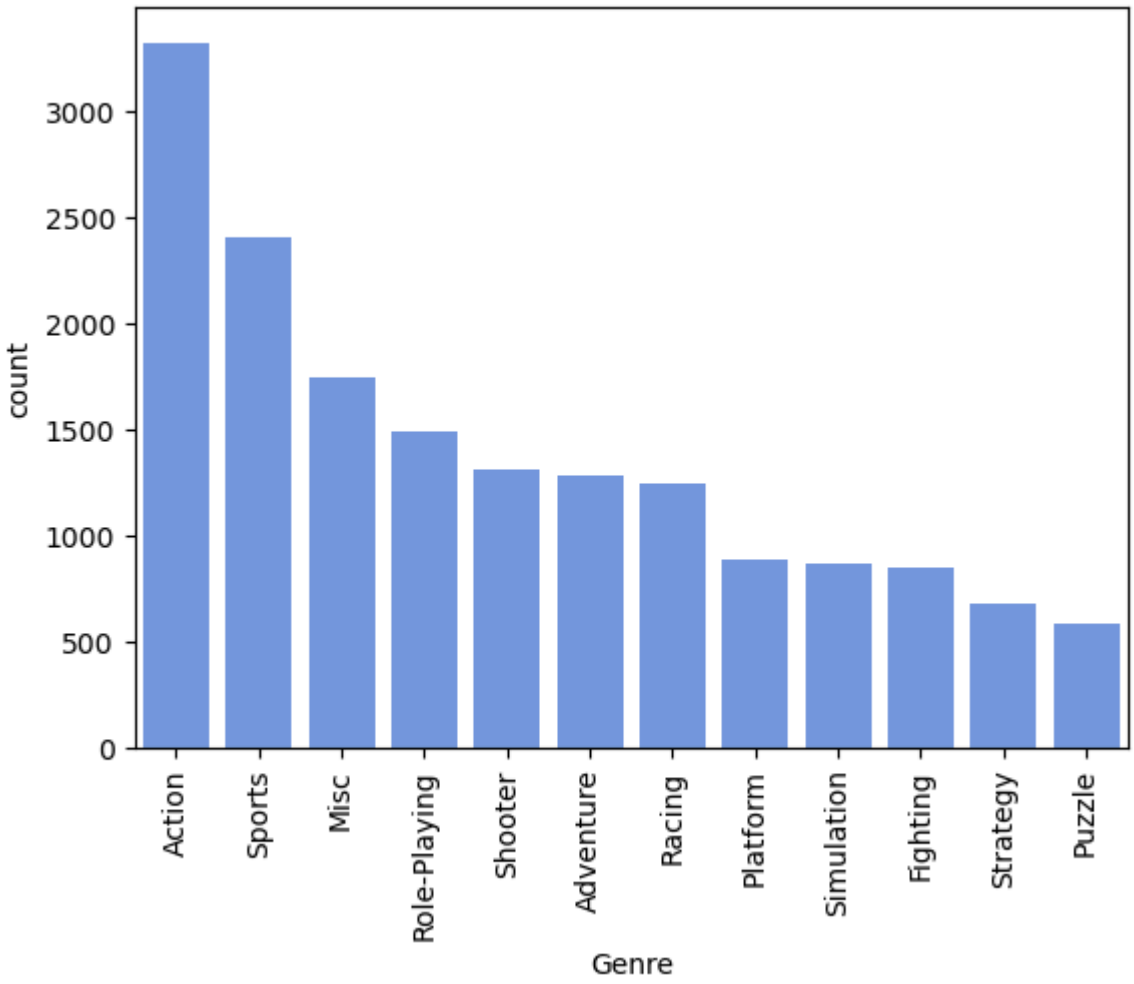
- In Seaborn, the same plot is called a **countplot**.
- It automatically does the counting of frequencies for you.

Why not just call it a barplot?

There is another function in Seaborn called a **barplot** which has some other purpose. We'll discuss this later.

In [13]:
```
sns.countplot(x='Genre', data=data, order=data['Genre'].value_counts().index
plt.xticks(rotation=90)
```

Out[13]:
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11]),
 [Text(0, 0, 'Action'),
  Text(1, 0, 'Sports'),
  Text(2, 0, 'Misc'),
  Text(3, 0, 'Role-Playing'),
  Text(4, 0, 'Shooter'),
  Text(5, 0, 'Adventure'),
  Text(6, 0, 'Racing'),
  Text(7, 0, 'Platform'),
  Text(8, 0, 'Simulation'),
  Text(9, 0, 'Fighting'),
  Text(10, 0, 'Strategy'),
  Text(11, 0, 'Puzzle')])
```



The top 5 genres are action, sports, misc, role-playing, and shooter.

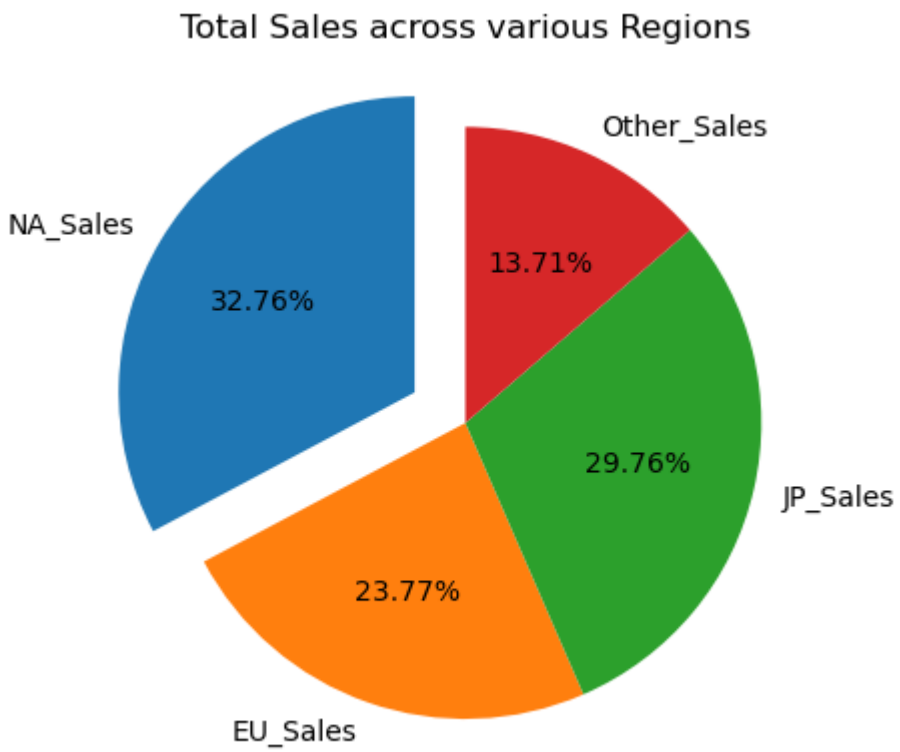
Pie Chart

What if instead of actual frequencues, we want see the proportion of the categories?

Say, we want to compare the distrubution/proportion of sales across different regions?

Which plot can we use for this? A pie-chart!

In [15]:
```
sales_data = data[['NA_Sales', 'EU_Sales', 'JP_Sales', 'Other_Sales']]
region_sales = sales_data.T.sum(axis='columns')

plt.pie(region_sales,
        labels=region_sales.index,
        startangle=90,
        explode=(0.2,0,0,0),
        autopct = '%.2f%%') # label the wedges with their numeric value

plt.title('Total Sales across various Regions')
plt.show()
```



In [17]:
```
region_sales = sales_data.T.sum(axis='columns')
region_sales
```

Out[17]:
```
NA_Sales       45831.525845
EU_Sales       33251.970702
JP_Sales       41624.625635
Other_Sales    19180.256828
dtype: float64
```

Univariate Data Visualisation - Numerical Data

What kind of questions we may have regarding a numerical variable?

- How is the data distributed?
- Is the data skewed? Are there any outliers?
- How much percentage of data is below/above a certain number?
- Statistics like - Min, Max, Mean, Median, etc.

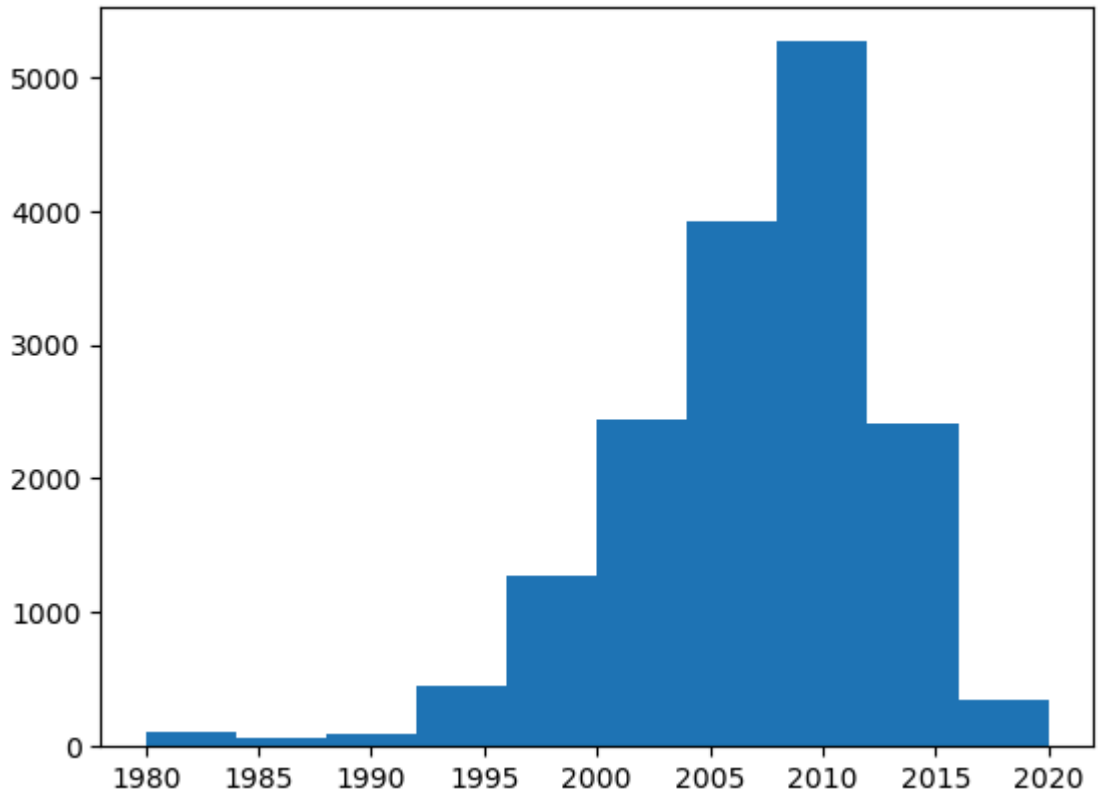
Now say you want to find the distribution of games released every year.

Unlike barplot, to see the distribution here we will have to `bin` the data.

How can we understand popularity of video games year by year?

Histogram

In [18]:
```
plt.hist(data['Year'])
plt.show()
```



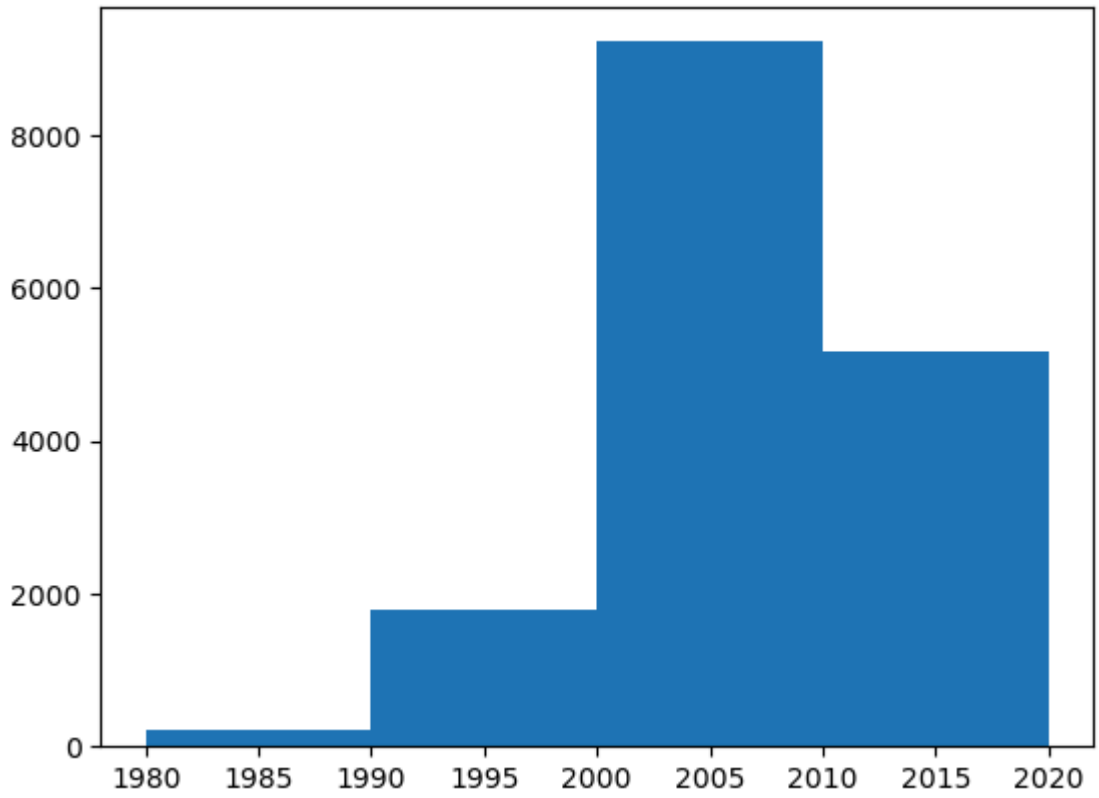
- The curve is left skewed, with a lot more games being published in 2005-2015.
- This shows that games started becoming highly popular in the last 1-2 decades.
- Maybe could point to increased usage of internet worldwide.

If you notice, histograms are basically frequency charts.

We can also vary the number of bins. **The default number of bins is 10**

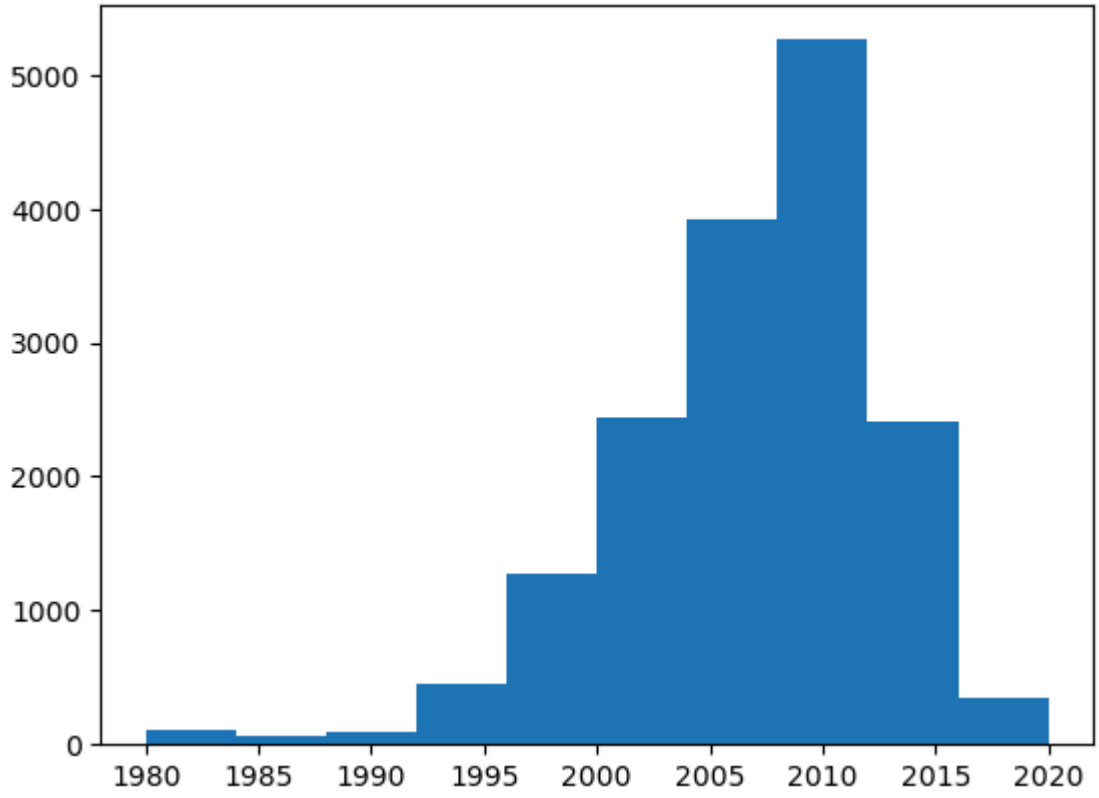
If we want to see this data per decade, we would need 40 years in 4 bins.

In [19]:
```
plt.hist(data['Year'], bins=4)
plt.show()
```



We can also get the data of each bin, such as range of the boundaries, values, etc.

In [20]:
```
count, bins, _ = plt.hist(data['Year'])
```



In [21]:
```
count
```

Out[21]:
```
array([ 112.,   70.,   92.,  449., 1274., 2440., 3921., 5262., 2406.,
        355.])
```

In [22]:
```
bins
```

Out[22]:
```
array([1980., 1984., 1988., 1992., 1996., 2000., 2004., 2008., 2012.,
       2016., 2020.])
```

What do these `count` and `bins` mean?

- **bins** provides bin edges
- **counts** provides it corresponding counts

What is the length of `count`?

- 10

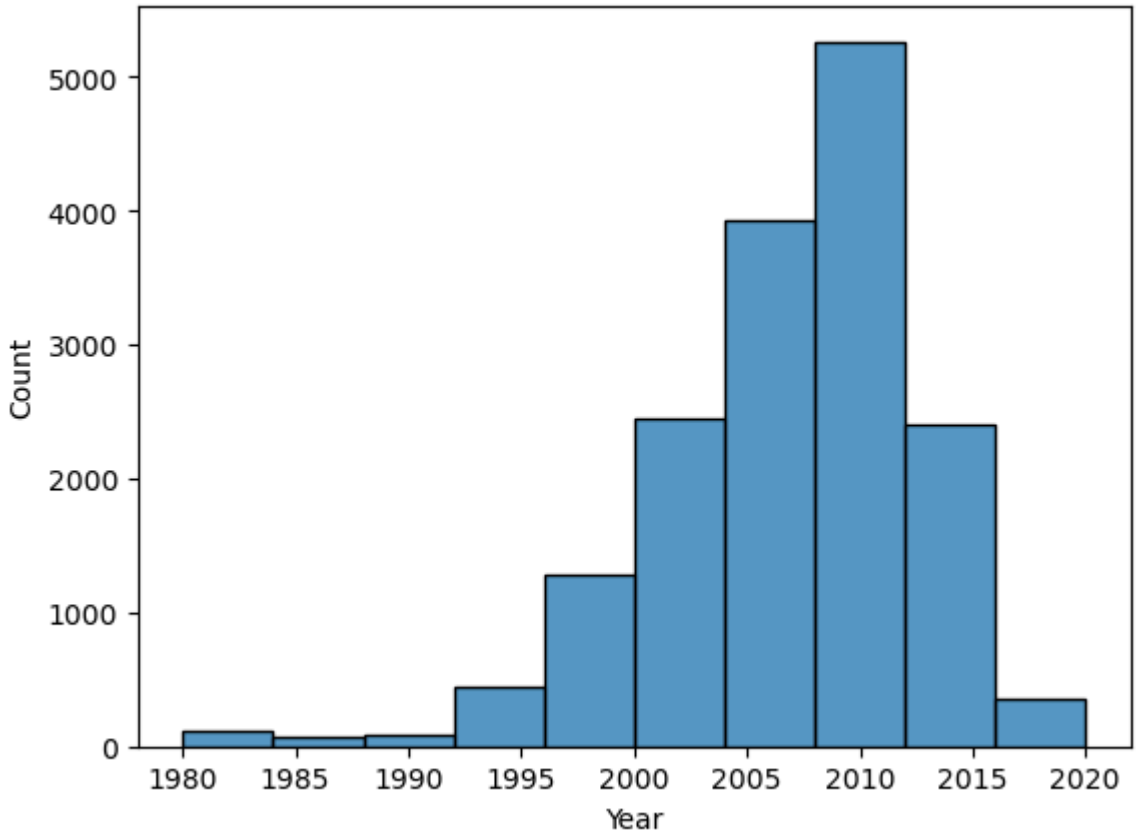
What shoud be the length of `bins`?

- 10 + 1 = 11

How can we plot a histogram in Seaborn?

In [23]:
```
sns.histplot(data['Year'], bins=10)
```

Out[23]:
```
<Axes: xlabel='Year', ylabel='Count'>
```



Notice that,

- The boundaries are more defined than matplotlib's plotting.
- The x and y axis are labelled automatically.

Kernel Density Estimate (KDE) Plot

- A KDE plot, similar to histrogram, is a method for visualizing the distributions.
- But instead of bars, KDE represents data using a **continuous probability density curve**.

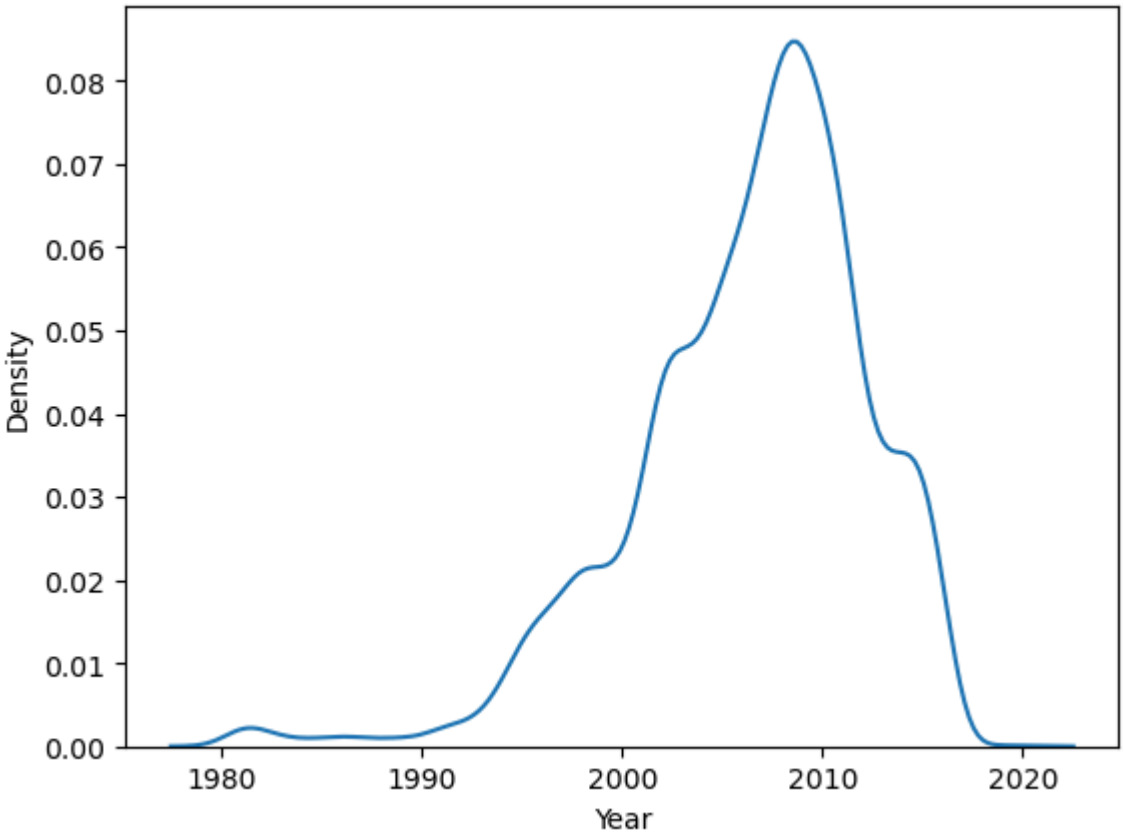
Why do we even need KDE plots?

- Compared to histogram, KDE produces a plot which is **less cluttered** and **more interpretable**.
- Think of it as a **smoothened version** of a histogram.

\ Let's plot KDE using `seaborn`'s `kdeplot`.

In [24]:
```
sns.kdeplot(data['Year'])
```

Out[24]:
```
<Axes: xlabel='Year', ylabel='Density'>
```



Can you notice the difference between KDE plot and histogram?

The Y-axis has **probability density estimation** instead of count.

You can read more about this on:

- https://en.wikipedia.org/wiki/Kernel_density_estimation
- https://www.youtube.com/watch?v=DCgPRaIDYXA

Boxplot

What if we want to find the aggregates like median, min, max and percentiles of the data.

Say I want the typical earnings of a game when it is published.

What kind of plot can we use here? Boxplot

What exactly is a Box plot?

- A box plot or **box and whiskers plot** shows the **distribution of quantitative data**.
- It facilitates comparisons between
  - attributes
  - across levels

of a categorical attribute.

- The **box** shows the **quartiles** of the dataset.
- The **whiskers** show the **rest of the distribution**.
- Except for points that are determined to be **"outliers"** using a method that is a function of the **inter-quartile range**.

Let's go through the terminology one-by-one.

Box plots show the five-number summary of data:

1. Minimum score
2. First (lower) quartile
3. Median
4. Third (upper) quartile
5. Maximum score

1. Minimum Score

- It is the **lowest value**, excluding outliers.
- It is shown at the **end of bottom whisker**.

2. Lower Quartile

- **25% of values** fall below the lower quartile value.
- It is also known as the **first quartile**.

3. Median

- Median marks the **mid-point of the data**.
- It is shown by the **line that divides the box into two parts**.
- **Half the scores are greater than or equal to this value and half are less**.
- It is sometimes known as the **second quartile**.

4. Upper Quartile

- **75% of values** fall below the upper quartile value.
- It is also known as the **third quartile**.

Maximum Score

- It is the **highest value**, excluding outliers.
- It is shown at the **end of upper whisker**.

Whiskers

- The upper and lower whiskers represent **values outside the middle 50%**.

- That is, the **lower 25% of values** and the **upper 25% of values**.

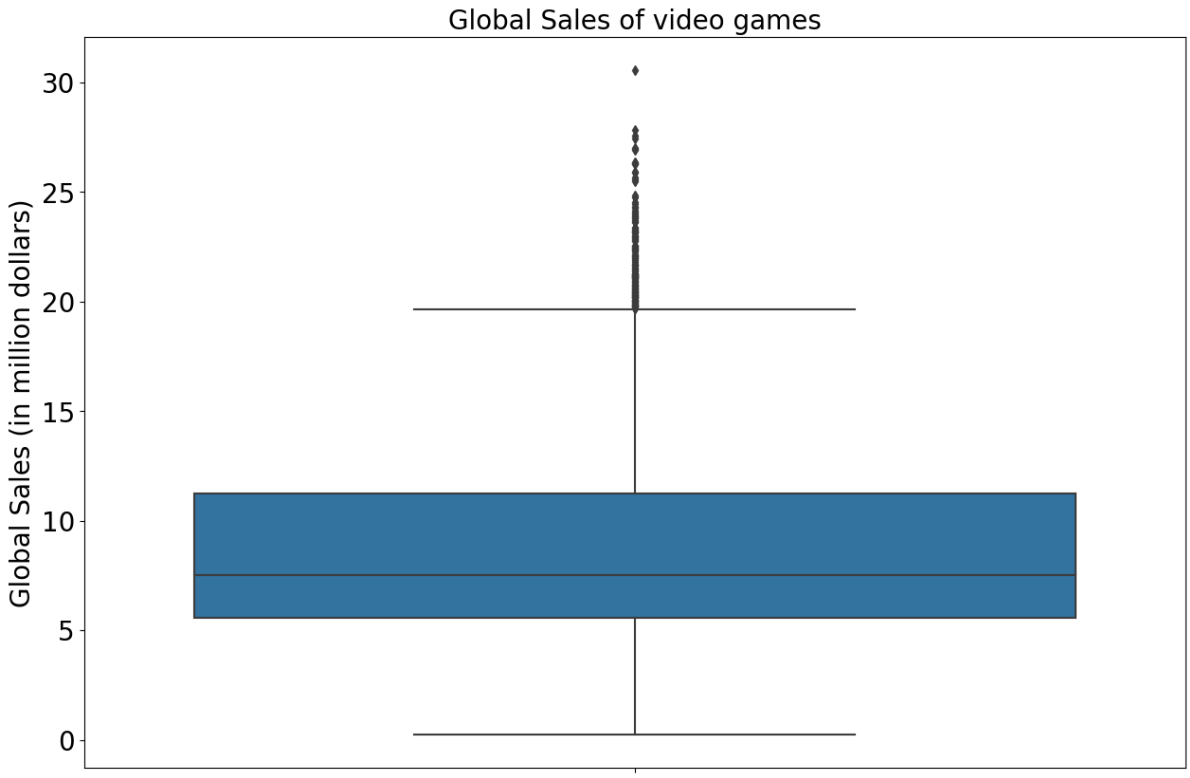
Interquartile Range (or IQR)

- This is the box plot showing the **middle 50% of scores**.
- It is the **range between the 25th and 75th percentile**.

\ Let's plot a box plot to find the average typical earnings for a game.

In [25]:
```
plt.figure(figsize=(15,10))
sns.boxplot(y = data["Global_Sales"])
plt.yticks(fontsize=20)
plt.ylabel('Global Sales (in million dollars)', fontsize=20)
plt.title('Global Sales of video games', fontsize=20)
```

Out[25]:
```
Text(0.5, 1.0, 'Global Sales of video games')
```



What can we infer from this?

The 5 point estimates here are:

- Minimum, excluding outliers: 0
- Maximum, excluding outliers: 20 (in million dollars)
- 25th Quantile: 6 million
- Median: around 7 million
- 75th Quantile: 12 million

Note:

- The outliers always will appear either below the minimum or above the maximum.
- There are quite a few outliers above 20 million dollars, represented by black colored circles.

Question-1

Q. For analyzing marks given by an edtech company, we want to find the range in which the most number of students have scored.

Which will be the best suited plot for this?

a. Pie Chart

b. Histogram

c. Countplot

d. Line Plot

Answer: Histogram

Explanation:

A histogram provides a visual representation of the distribution of numerical data, divided into intervals called bins. It allows you to see the frequency or count of data points within each bin, making it ideal for identifying the range in which the most number of students have scored. By observing the peak(s) in the histogram, you can easily determine the range(s) with the highest frequency of scores.

Question-2

Q. The telecom company "Airtel" wants to find the count customers across different payment modes opted by the customer.

Which will be the best suited plot for this?

a. Pie Chart

b. Countplot

c. Line Plot

d. Boxplot

Answer: Count Plot

Explanation:

A countplot is specifically designed for visualizing the count of observations in each category of a categorical variable. In this case, the payment modes are categorical variables, and the countplot will provide a clear representation of the number of customers using each payment mode. This makes it ideal for comparing the distribution of customers across different payment modes.

Question-3

Q. In the state “Haryana”, we want to find the proportion of people who smoke.

Which will be the best suited plot for this?

a. Pie Chart

b. Bar Chart

c. Countplot

d. Boxplot

Answer: Pie Chart

Explanation:

A pie chart is an effective way to represent proportions or percentages within a whole. In this case, the proportion of people who smoke can be represented as a fraction of the entire population of "Haryana". Each section of the pie chart would represent a different category, such as smokers and non-smokers, allowing for a clear visual comparison of the proportion of smokers in the population.

Data Viz 2

Content

- Bivariate Data Visualization
- Continous-Continuous
  - Line plot
  - Styling and Labelling
  - Scatter plot
- Categorical-Categorical
  - Dodged countplot
  - Stacked countplot
- Categorical-Continuous
  - Boxplot
  - Bar plot
- Subplots

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
data = pd.read_csv('final.csv')
data.head()
```

Out[3]:

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sa
0	2061	1942	NES	1985.0	Shooter	Capcom	4.569217	3.033887	3.439
1	9137	¡Shin Chan Flipa en colores!	DS	2007.0	Platform	505 Games	2.076955	1.493442	3.033
2	14279	.hack: Sekai no Mukou ni + Versus	PS3	2012.0	Action	Namco Bandai Games	1.145709	1.762339	1.493
3	8359	.hack//G.U. Vol.1//Rebirth	PS2	2006.0	Role-Playing	Namco Bandai Games	2.031986	1.389856	3.228
4	7109	.hack//G.U. Vol.2//Reminisce	PS2	2006.0	Role-Playing	Namco Bandai Games	2.792725	2.592054	1.440

In [4]:
```
data.describe()
```

Out[4]:

	Rank	Year	NA_Sales	EU_Sales	JP_Sales	Other_Sa
count	16652.000000	16381.000000	16652.000000	16652.000000	16652.000000	16652.0000
mean	8283.409620	2006.390513	2.752314	1.996875	2.499677	1.1518
std	4794.471477	5.863261	1.327002	1.322972	1.164023	1.0548
min	1.000000	1980.000000	0.140000	0.010000	0.000000	-0.4742
25%	4129.750000	2003.000000	1.781124	1.087977	1.781124	0.3948
50%	8273.500000	2007.000000	2.697415	1.714664	2.480356	0.4918
75%	12436.250000	2010.000000	3.677290	2.795123	3.176299	1.7811
max	16600.000000	2020.000000	8.725452	8.367985	12.722984	7.3580

Continuous-Continuous

So far we have been analyzing only a single feature.

But what if we want to visualize two features at once?

What kind of questions can we ask regarding a continuous-continuous pair of features?

- Show relation between two features, like **how does the sales vary over the years**?
- Show **how are the features associated, positively or negatively**?

...and so on

Line Plot

How can we plot the sales trend over the years for the longest running game?

First, let's find the longest running game first.

In [5]:
```
game_life = data.groupby('Name').agg(min_year = ('Year', 'min'), max_year =
game_life['range'] = game_life['max_year'] - game_life['min_year']
game_life.sort_values(['range'], ascending = False)[:5]
```

Out[5]:

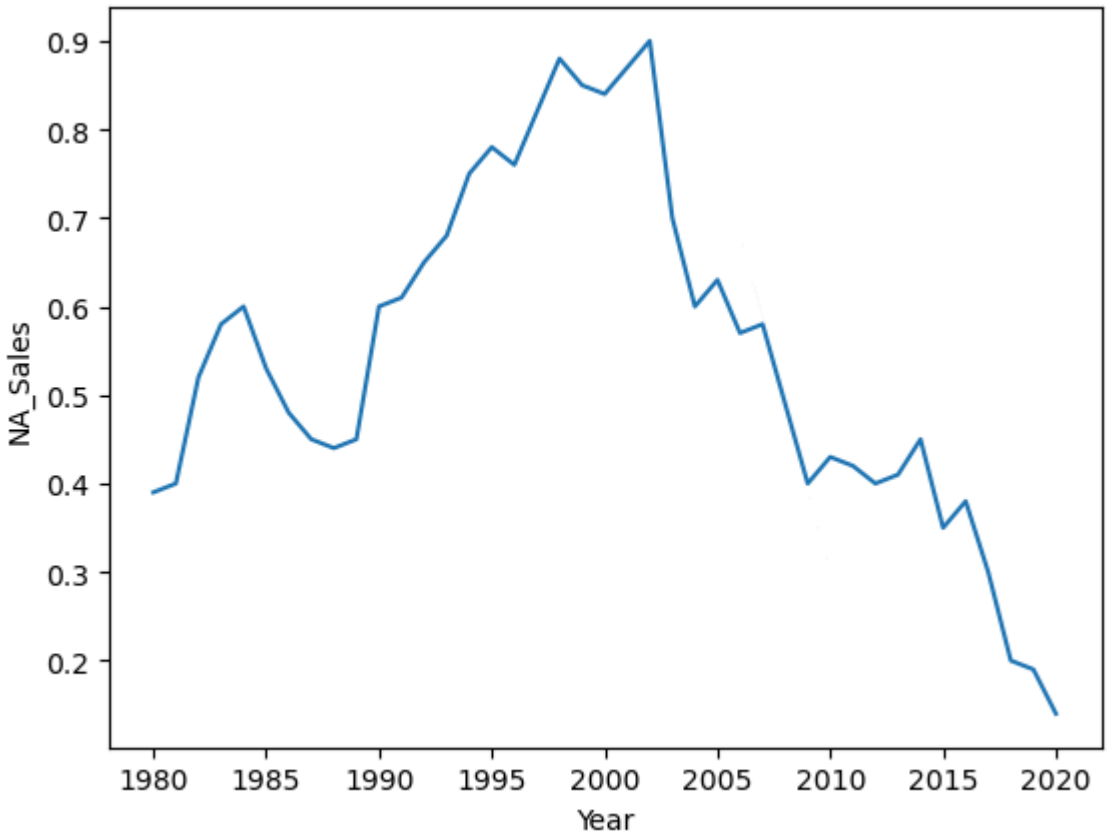
Name	min_year	max_year	range
Ice Hockey	1980.0	2020.0	40.0
Baseball	1980.0	2019.0	39.0
Battlezone	1982.0	2006.0	24.0
Romance of the Three Kingdoms II	1991.0	2015.0	24.0
Bomberman	1985.0	2008.0	23.0

Great! So `Ice Hockey` is longer running than most of the games.

Let's try to find the sales trend in North America of the same across the years.

In [6]:
```
ih = data.loc[data['Name']=='Ice Hockey']
sns.lineplot(x='Year', y='NA_Sales', data=ih)
```

Out[6]:
```
<Axes: xlabel='Year', ylabel='NA_Sales'>
```



What can we infer from this graph?

- The sales across North America seem to have been boosted in the years of 1995-2005.
- Post 2010 though, the sales seem to have taken a dip.

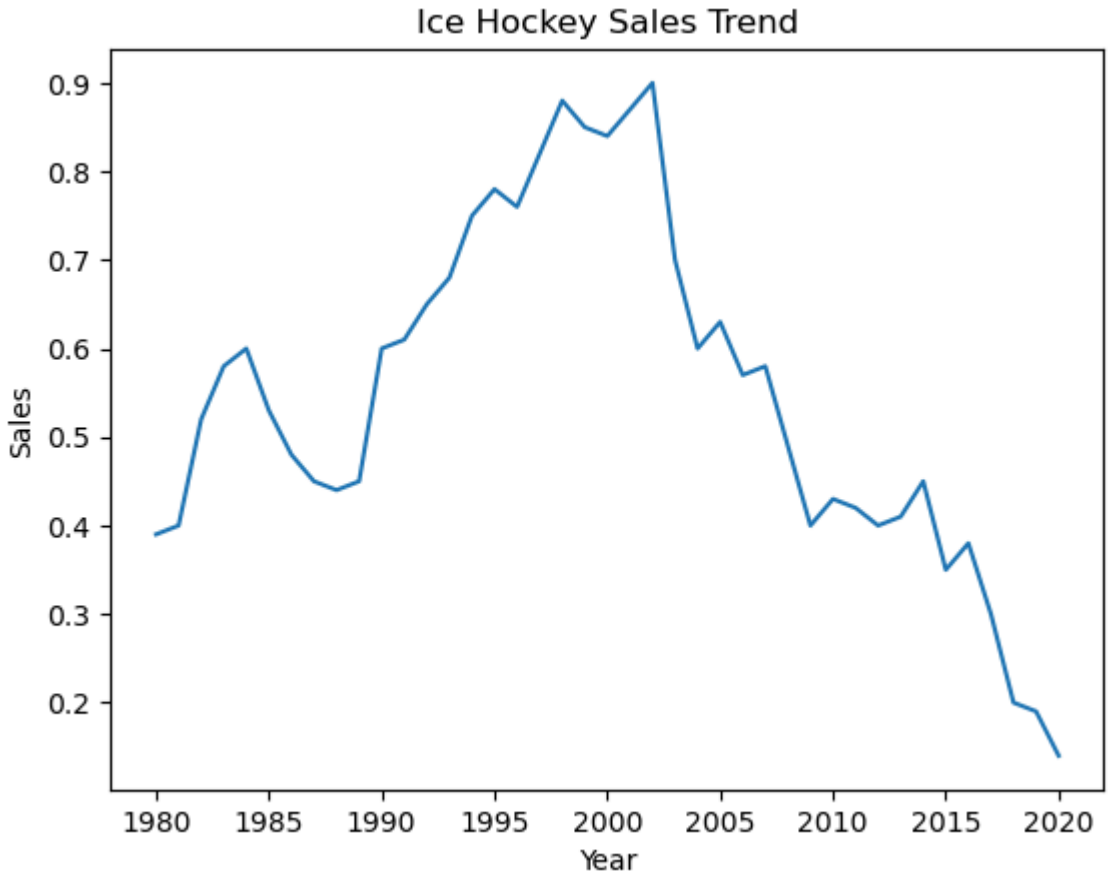
Line plots are great for representing trends such as above, over time.

Style and Labelling

We already learnt how to add **titles, x-label and y-label** in barplot.

Let's do the same here.

In [7]:
```
plt.title('Ice Hockey Sales Trend')
plt.xlabel('Year')
plt.ylabel('Sales')
sns.lineplot(x='Year', y='NA_Sales', data=ih)
plt.show()
```



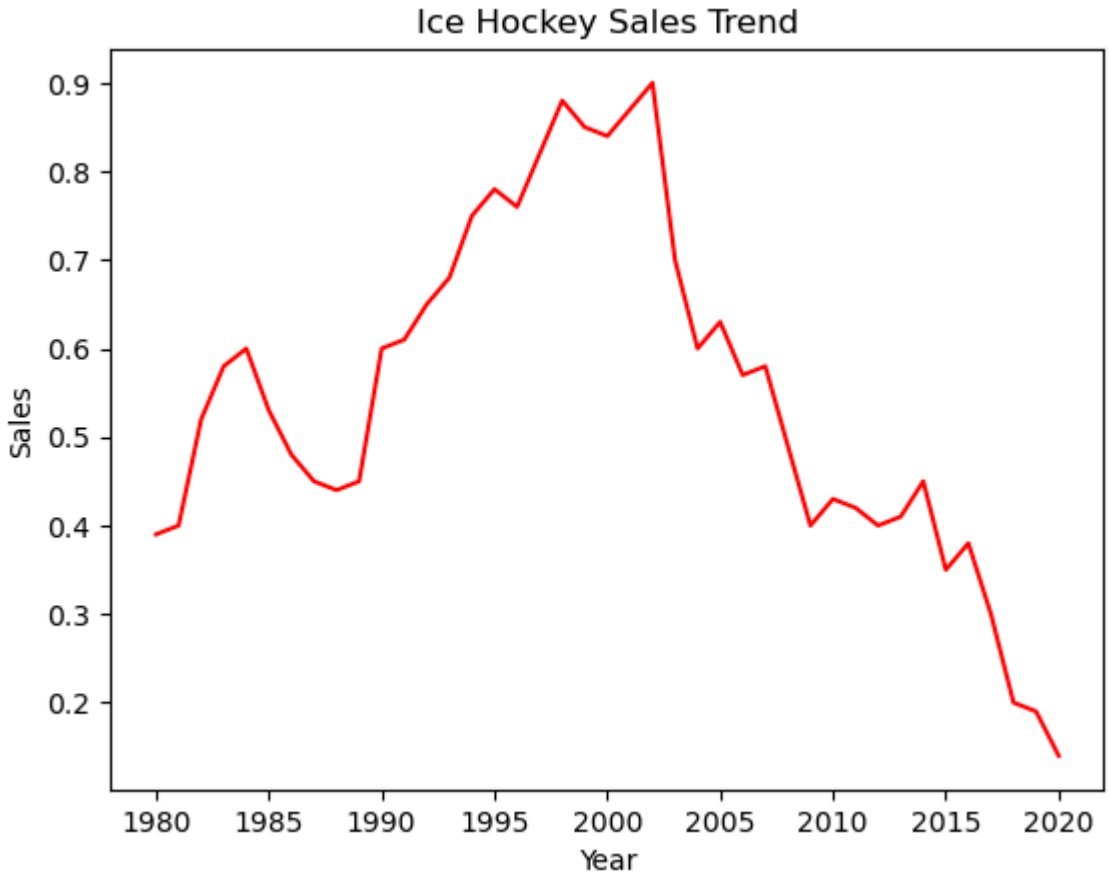
What if we want to change the colour of the curve?

- `sns.lineplot()` contains an argument **color**
- It takes a matplotlib color or string for some defined colours.
  - black: `k` / `black`
  - red: `r` / `red`

But what all colours can we use?

- Matplotlib provides a variety of colors.
- Check the documentation for more - https://matplotlib.org/2.0.2/api/colors_api.html

In [8]:
```
plt.title('Ice Hockey Sales Trend')
plt.xlabel('Year')
plt.ylabel('Sales')
sns.lineplot(x='Year', y='NA_Sales', data=ih, color='r')
plt.show()
```



Now, let's say we only want to show the values from years 1990-2000.

How can we limit our plot to only the last decade of 20th century?

- This requires changing the range of x-axis.

But how can we change the range of an axis in matplotlib?

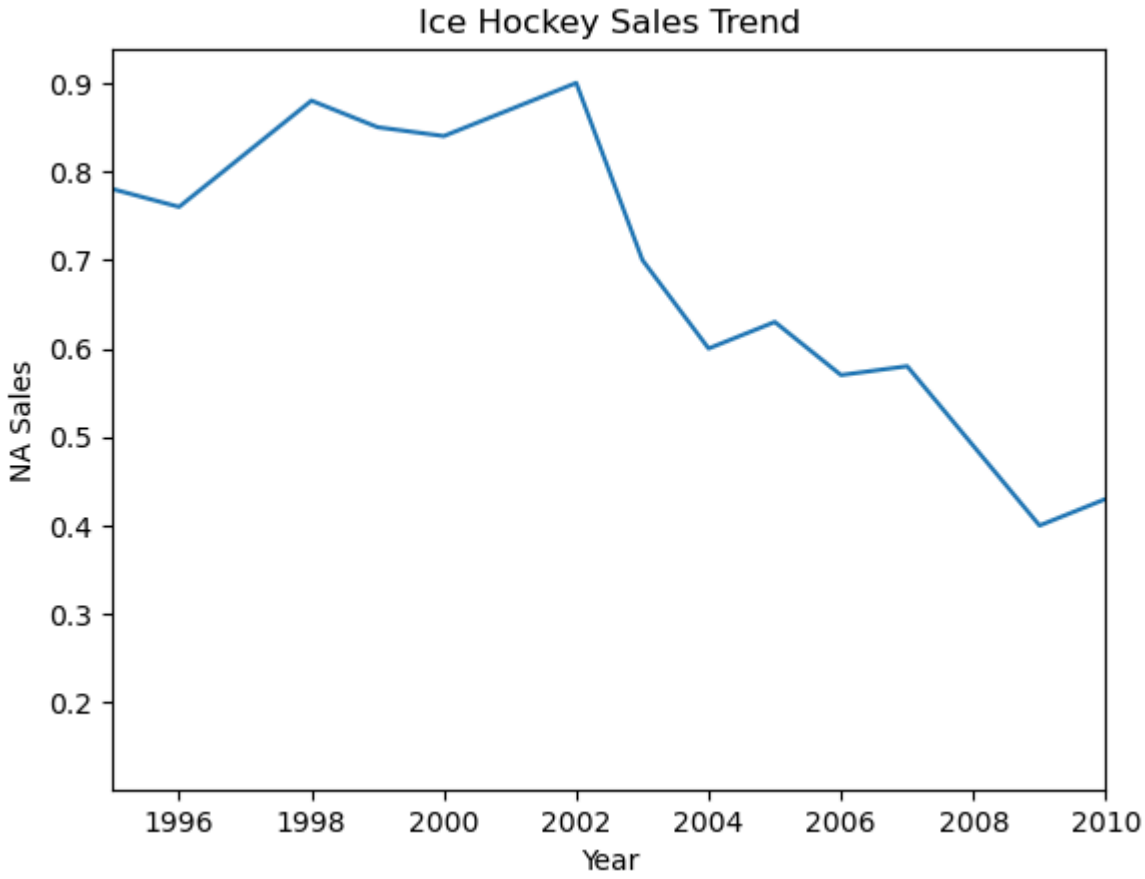
We can use:

- `plt.xlim()` : x-axis
- `plt.ylim()` : y-axis

These functions take 2 arguments:

1. `left` : Starting point of range
2. `right` : End point of range

In [9]:
```
plt.title('Ice Hockey Sales Trend')
plt.xlabel('Year')
plt.ylabel('NA Sales')
plt.xlim(left=1995,right=2010)
sns.lineplot(x='Year', y='NA_Sales', data=ih)
plt.show()
```



So far we have visualised a single plot to understand it.

What if we want to compare it with some other plot?

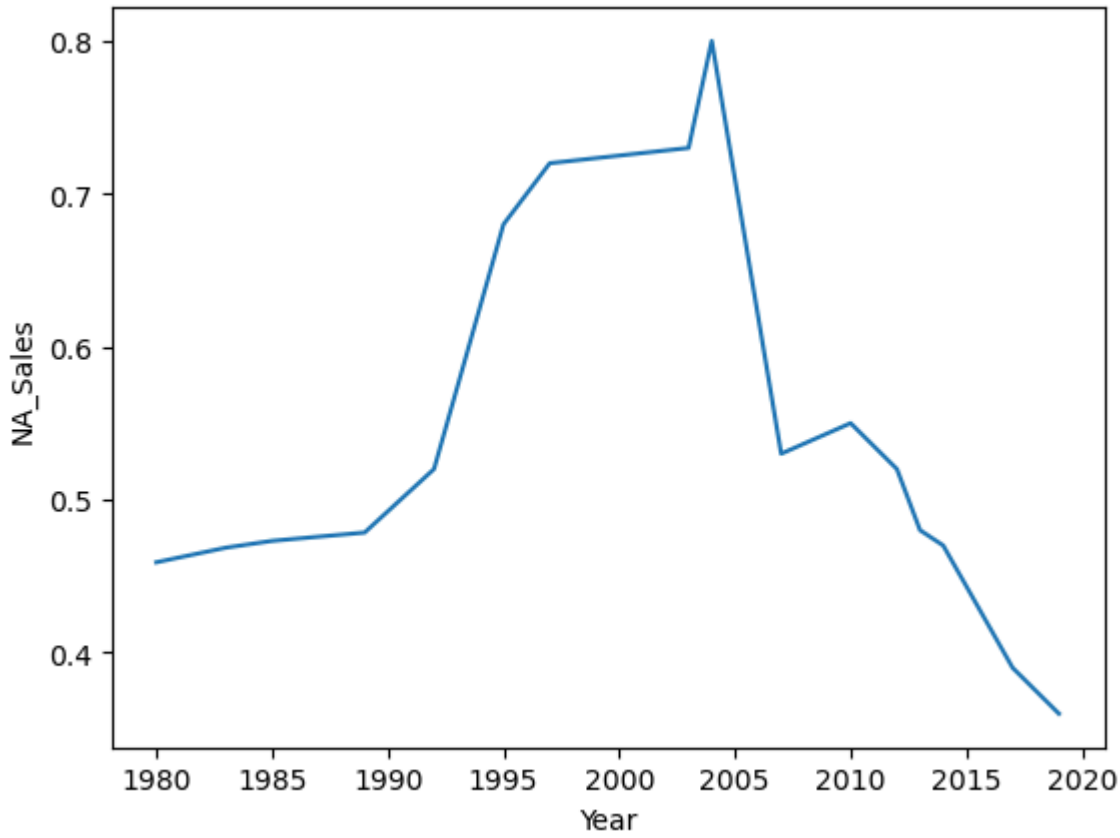
Say, we want to compare the same sales trend between two games.

- Ice Hockey
- Baseball

Let's first plot the trend for "Baseball".

In [10]:
```
baseball = data.loc[data['Name']=='Baseball']
sns.lineplot(x='Year', y='NA_Sales', data=baseball)
```

Out[10]:
```
<Axes: xlabel='Year', ylabel='NA_Sales'>
```

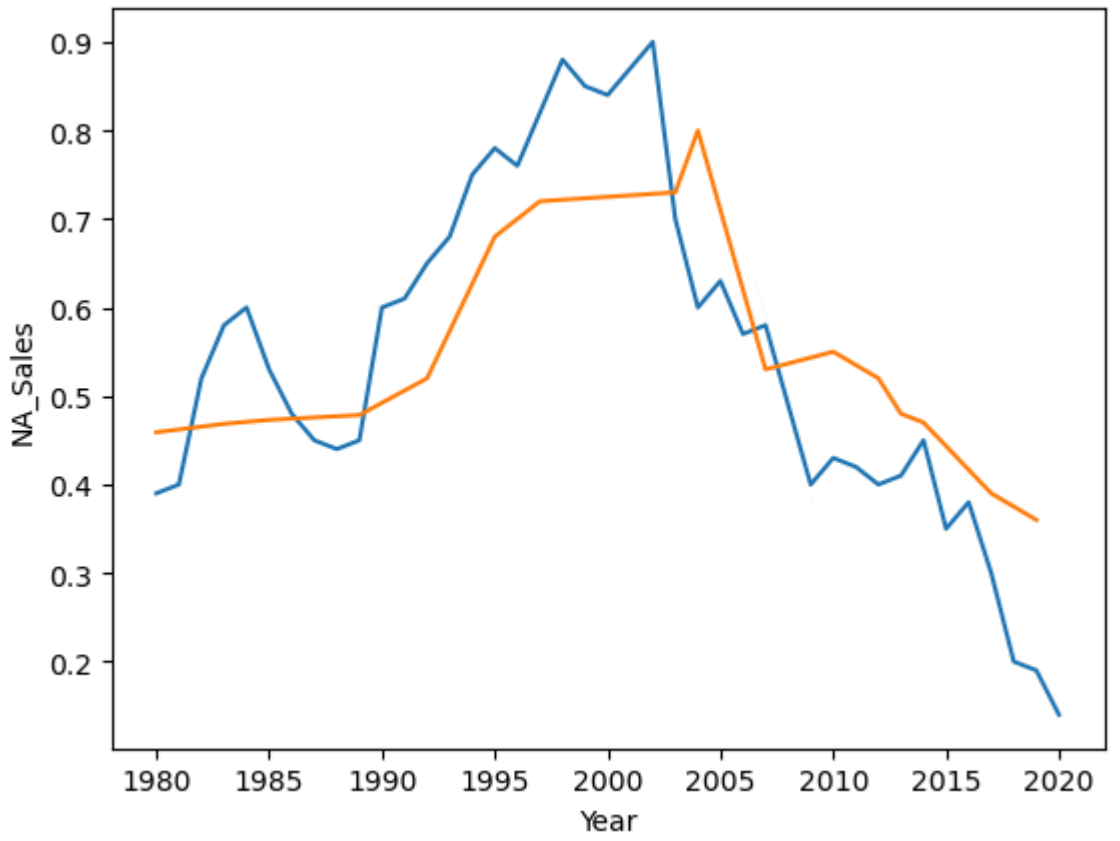


Now for the comparison, we will have to draw these plots in the same figure.

How can we plot multiple plots in the same figure?

In [11]:
```
sns.lineplot(x='Year', y='NA_Sales', data=ih)
sns.lineplot(x='Year', y='NA_Sales', data=baseball)
```

Out[11]:
```
<Axes: xlabel='Year', ylabel='NA_Sales'>
```



We can use multiple `sns.lineplot()` functions.

Observe that Seaborn automatically created 2 plots with **different colors**.

But how can we know which colour is of which plot?

- We can simply set the label of each plot.
- `sns.lineplot()` has another argument **label** to do so.

In [12]:
```
sns.lineplot(x='Year', y='NA_Sales', data=ih, label='Ice Hockey')
sns.lineplot(x='Year', y='NA_Sales', data=baseball, label='Baseball')
```

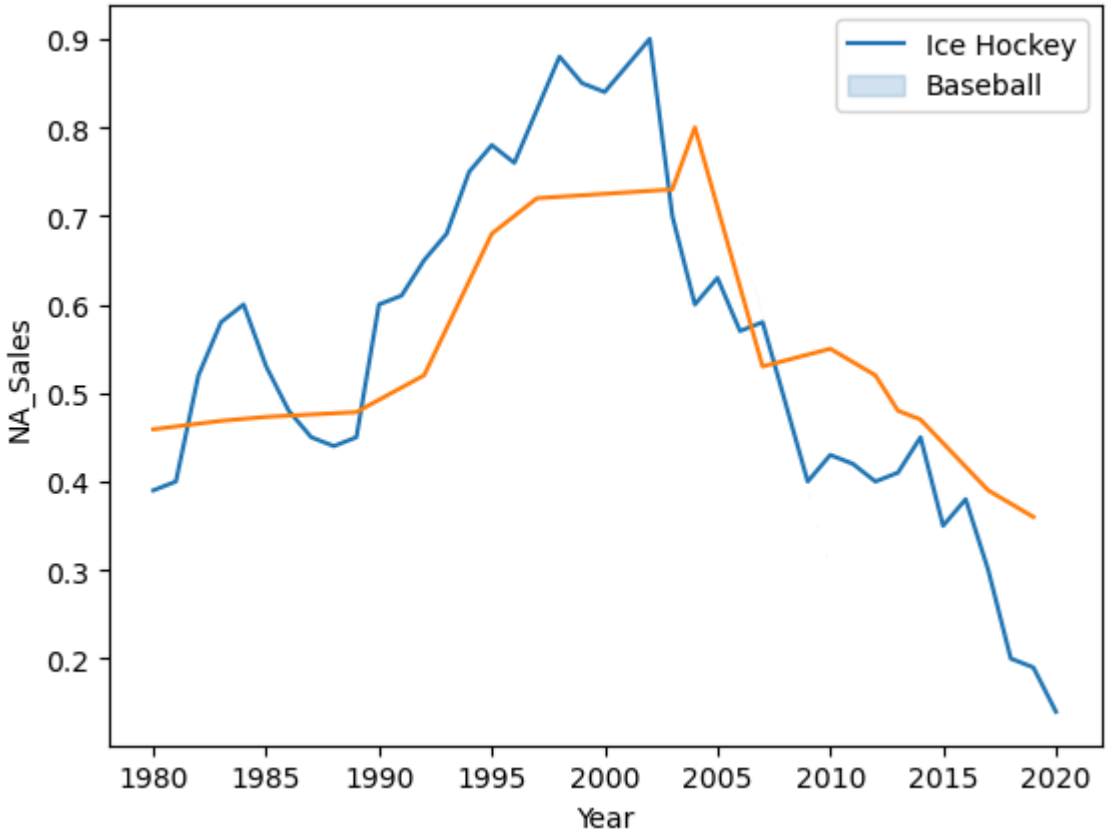
Out[12]:
```
<Axes: xlabel='Year', ylabel='NA_Sales'>
```

We can also pass these labels in `plt.legend()` as a list in the order the plots are created.

Please note that using labels in `plt.legend()` gives unexpected result in **Google Colab**. However, it works absolutely fine on other platforms.

Here, we will use `plt.legend()` for its other functionalities.

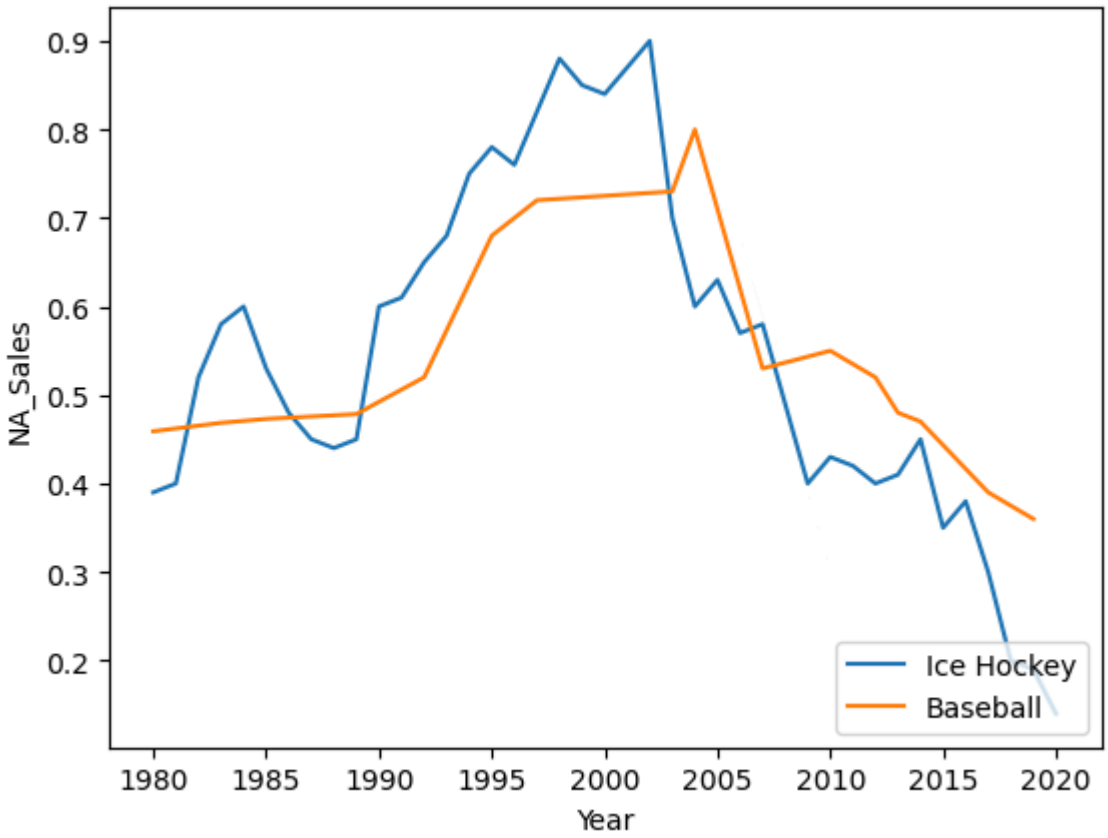
In [13]:
```
sns.lineplot(x='Year', y='NA_Sales', data=ih)
sns.lineplot(x='Year', y='NA_Sales', data=baseball)
plt.legend(['Ice Hockey','Baseball'])
plt.show()
```



Can we change the position of the legend, say to bottom-right corner?

- Matplotlib automatically decides the best position for the legends.
- But we can also change it using the `loc` parameter.
- `loc` takes input as 1 of following strings:
  - upper center
  - upper left
  - upper right
  - lower right

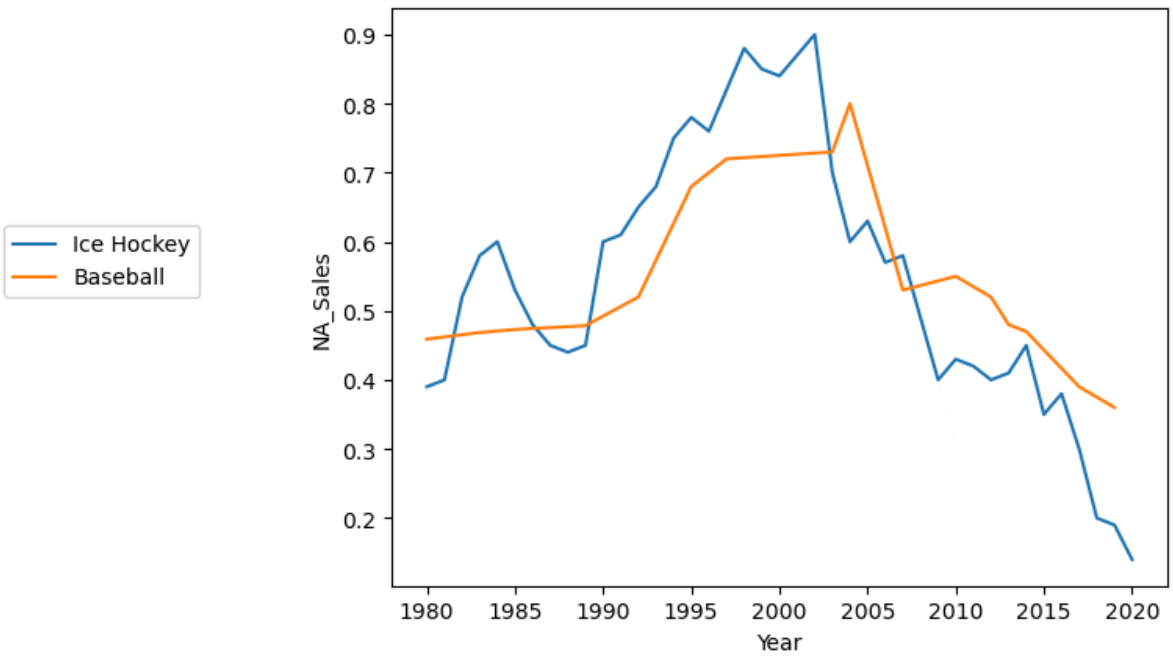
In [14]:
```
sns.lineplot(x='Year', y='NA_Sales', data=ih, label='Ice Hockey')
sns.lineplot(x='Year', y='NA_Sales', data=baseball, label='Baseball')
plt.legend(loc='lower right')
plt.show()
```



What if we want the legend to be outside the plot?

- Maybe the plot is too congested to show the legend.
- We can use the same `loc` parameter for this too.

In [15]:
```
sns.lineplot(x='Year', y='NA_Sales', data=ih, label='Ice Hockey')
sns.lineplot(x='Year', y='NA_Sales', data=baseball, label='Baseball')
plt.legend(loc=(-0.5,0.5))
plt.show()
```



The pair of floats signify the `(x,y)` coordinates for the legend.

From this we can conclude that `loc` takes **two types of arguments**:

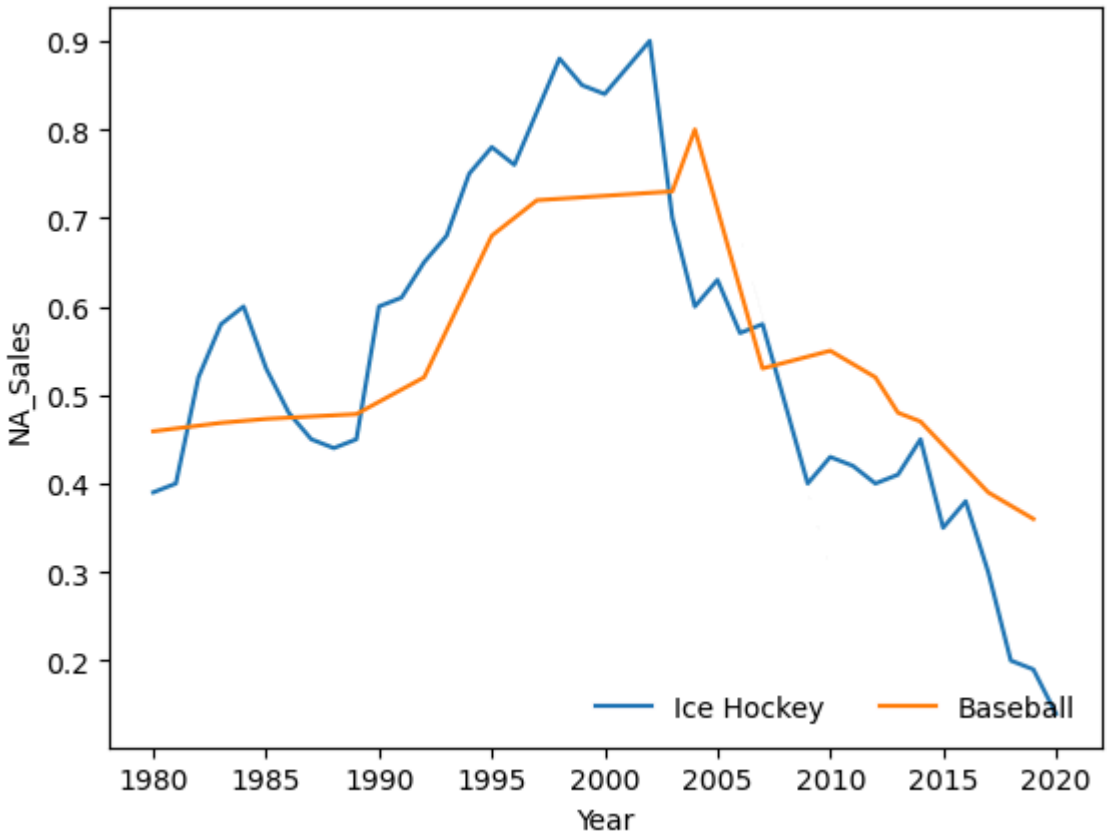
- The location in the **form of string**.
- The location in the **form of coordinates**.

\ **What if we want to add other stylings to legends?**

- Specify the **number of rows/cols**
  - Uses parameter `ncols` for this.
  - The number of **rows are decided automatically**.
- Decide if we want the box of legends to be displayed
  - Use the bool parameter `frameon`.

...and so on.

In [16]:
```
sns.lineplot(x='Year', y='NA_Sales', data=ih, label='Ice Hockey')
sns.lineplot(x='Year', y='NA_Sales', data=baseball, label='Baseball')
plt.legend(loc='lower right', ncol=2, frameon=False)
plt.show()
```



Now say we want to highlight a point on our curve.

How can we highlight the maximum "Ice Hockey" sales across all years?

Let's first find this point.

In [17]:
```
print(max(ih['NA_Sales']))
```

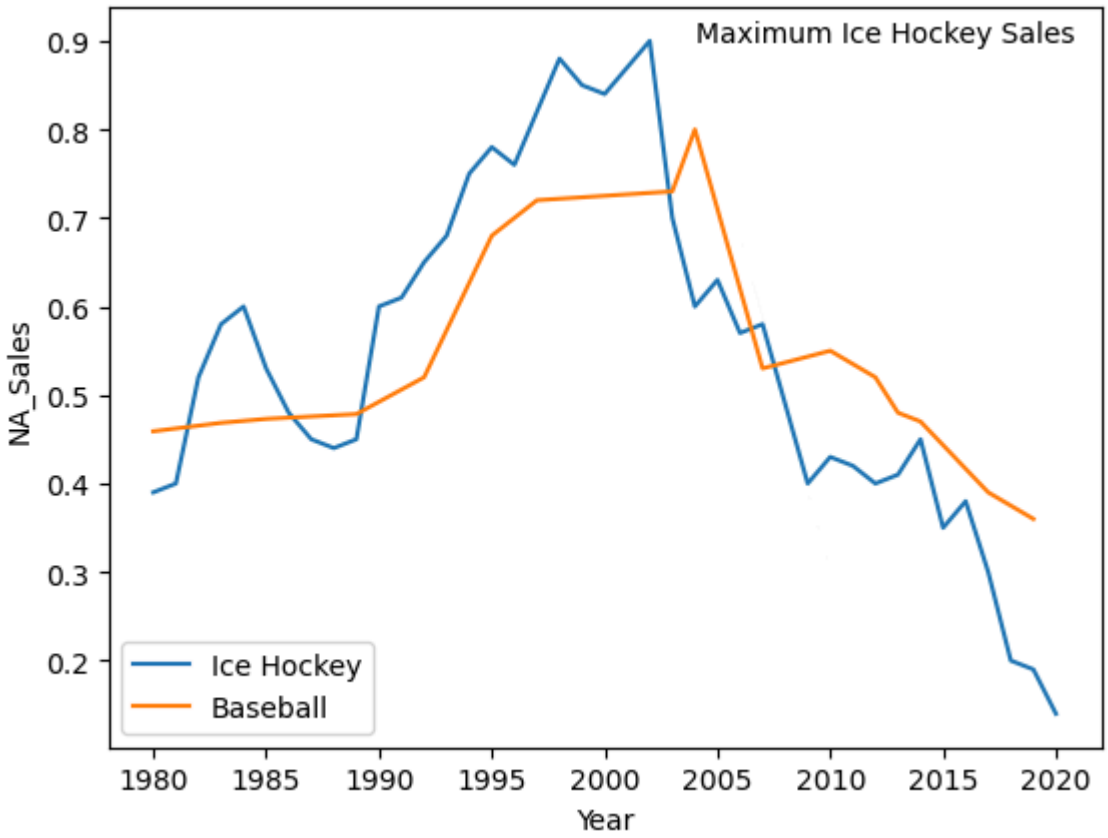
```
0.9
```

If we observe, this point lies in the year 2004-2005.

Now we need to add text to this point (2004,0.9)

How can we add text to a point in a figure?

In [18]:
```
sns.lineplot(x='Year', y='NA_Sales', data=ih, label='Ice Hockey')
sns.lineplot(x='Year', y='NA_Sales', data=baseball, label='Baseball')
plt.legend(loc='lower left')
plt.text(2004,max(ih['NA_Sales']), 'Maximum Ice Hockey Sales')
plt.show()
```

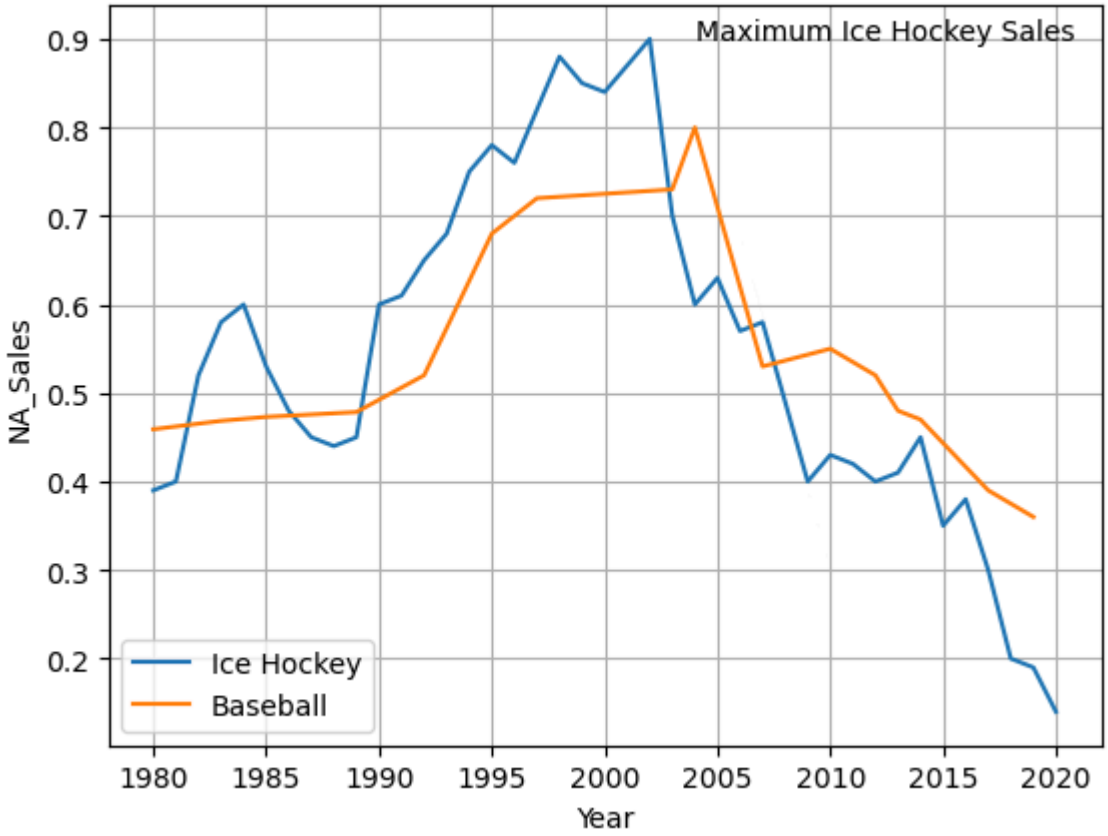


By using `plt.text()`

- Pass in the **x and y coordinates** where we want the text to appear.
- Pass in the **text string**.

We can also use `plt.grid()` to show the grid layout in the background.

In [19]:
```
sns.lineplot(x='Year', y='NA_Sales', data=ih, label='Ice Hockey')
sns.lineplot(x='Year', y='NA_Sales', data=baseball, label='Baseball')
plt.legend(loc='lower left')
plt.text(2004, max(ih['NA_Sales']), 'Maximum Ice Hockey Sales')
plt.grid()
plt.show()
```



Note: We can pass in parameters inside `plt.grid()` to control its density, colour of grid lines, etc.

You can look it up later on how to customize the grid - https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.grid.html

Scatter Plot

Suppose we want to find the relation between `Rank` and `Sales` of all games.

Are `Rank` and `Sales` positively or negatively correlated?

In this case, unlike line plot, there maybe multiple points in y-axis for each point in x-axis.

In [20]:
```
data.head()
```

Out[20]:

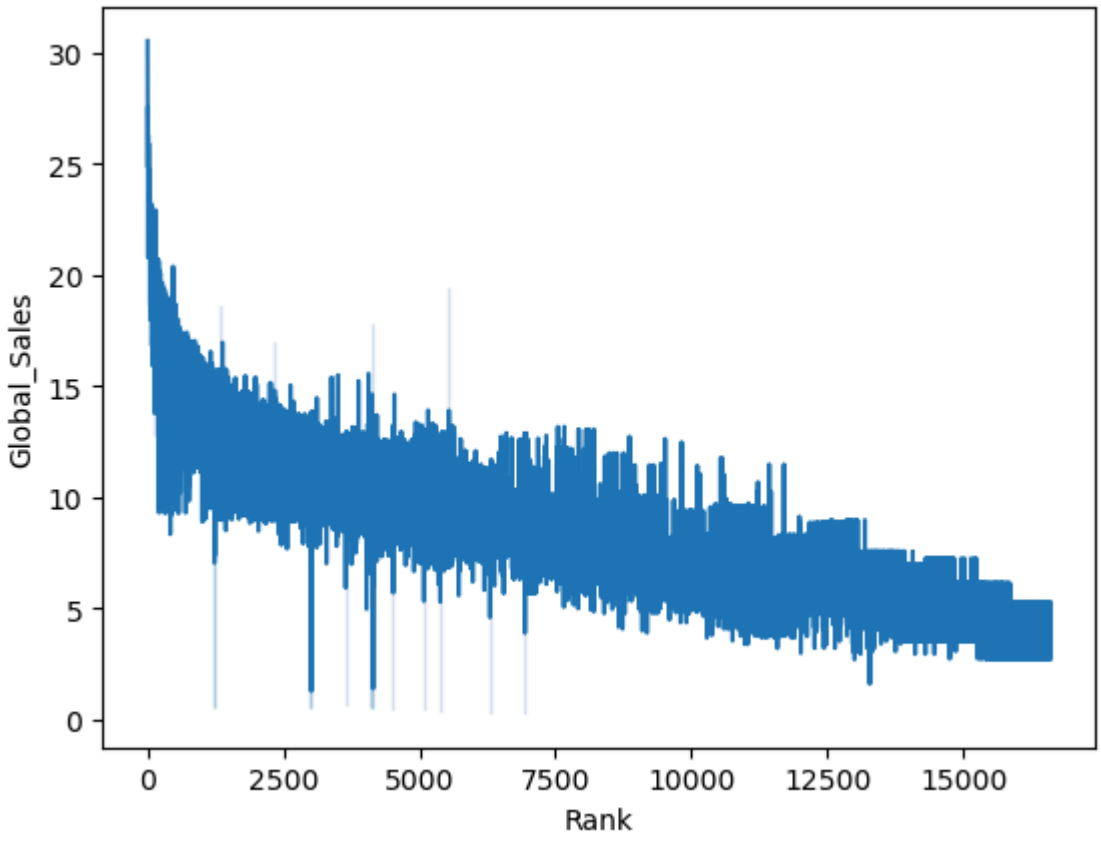
	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sa
0	2061	1942	NES	1985.0	Shooter	Capcom	4.569217	3.033887	3.439
1	9137	¡Shin Chan Flipa en colores!	DS	2007.0	Platform	505 Games	2.076955	1.493442	3.033
2	14279	.hack: Sekai no Mukou ni + Versus	PS3	2012.0	Action	Namco Bandai Games	1.145709	1.762339	1.493
3	8359	.hack//G.U. Vol.1//Rebirth	PS2	2006.0	Role-Playing	Namco Bandai Games	2.031986	1.389856	3.228
4	7109	.hack//G.U. Vol.2//Reminisce	PS2	2006.0	Role-Playing	Namco Bandai Games	2.792725	2.592054	1.440

How can we plot the relation between `Rank` and `Global Sales`?

Can we use lineplot? Let's try it out.

In [21]:
```
sns.lineplot(data=data, x='Rank', y='Global_Sales')
```

Out[21]:
```
<Axes: xlabel='Rank', ylabel='Global_Sales'>
```



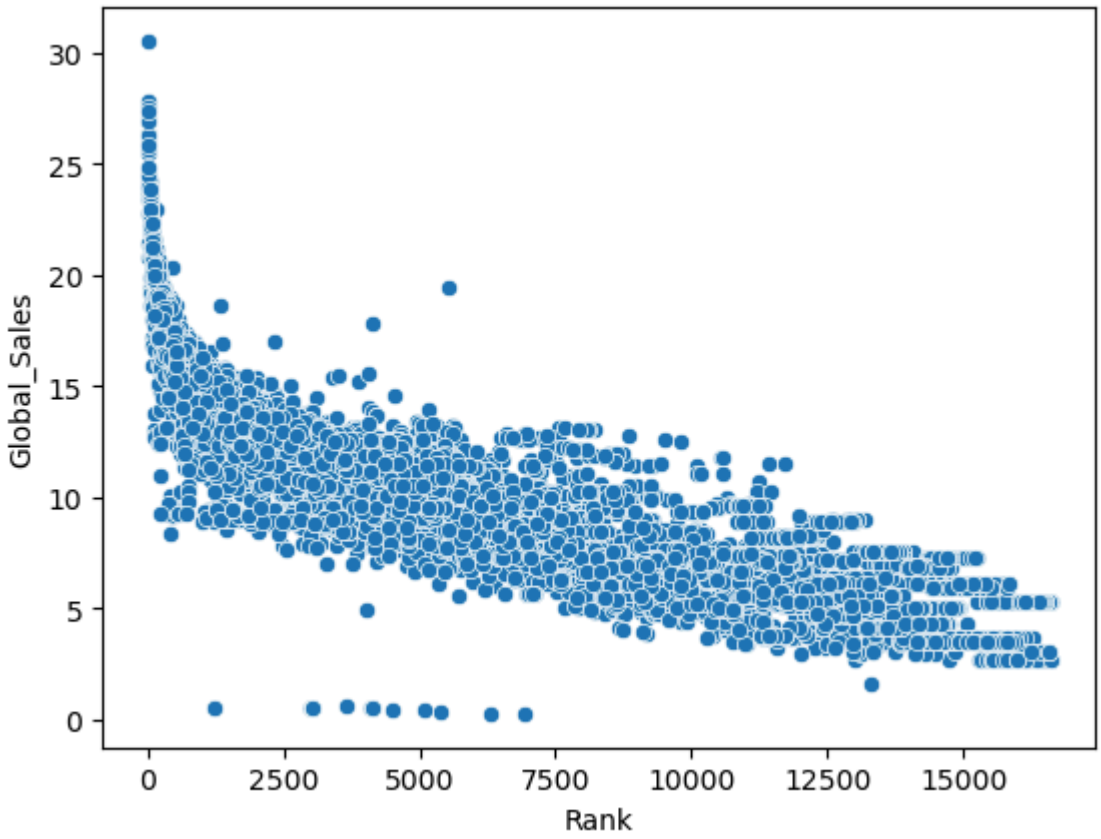
The plot itself looks very messy and it's hard to find any patterns from it.

Is there any other way we can visualize this relation?

- using `scatterplot()`

In [22]:
```
sns.scatterplot(data=data, x='Rank', y='Global_Sales')
```

Out[22]: <Axes: xlabel='Rank', ylabel='Global_Sales'>



Compared to lineplot, we are able to see the patterns and points more distinctly now!

Notice,

- The two variables are negatively correlated with each other.
- With increase in ranks, the sales tend to go down, implying lower ranked games have higher sales overall!

Scatter plots help us visualize these relations and find any patterns in the data.

Sometimes, people also like to display the linear trend between two variables - **Regression Plot**.

If you notice, `Genres`, `Publisher` and `Platform` are categorical values.

Since we have a lot of categories of each of them, we will use top 3 of each to make our analysis easier.

In [23]:
```
top3_pub = data['Publisher'].value_counts().index[:3]
top3_gen = data['Genre'].value_counts().index[:3]
top3_plat = data['Platform'].value_counts().index[:3]
top3_data = data.loc[(data["Publisher"].isin(top3_pub)) & (data["Platform"]
top3_data
```

Out[23]:

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sal
2	14279	.hack: Sekai no Mukou ni + Versus	PS3	2012.0	Action	Namco Bandai Games	1.145709	1.762339	1.4934
13	2742	[Prototype 2]	PS3	2012.0	Action	Activision	3.978349	3.727034	0.8488
16	1604	[Prototype]	PS3	2009.0	Action	Activision	4.569217	4.108402	1.1872
19	1741	007: Quantum of Solace	PS3	2008.0	Action	Activision	4.156030	4.346074	1.0879
21	4501	007: Quantum of Solace	PS2	2008.0	Action	Activision	3.228043	2.738800	2.5855
...	...	...	...	...	...	...	...	...	
16438	14938	Yes! Precure 5 Go Go Zenin Shu Go! Dream Festival	DS	2008.0	Action	Namco Bandai Games	1.087977	0.592445	1.0879
16479	10979	Young Justice: Legacy	PS3	2013.0	Action	Namco Bandai Games	2.186589	1.087977	3.4090
16601	11802	ZhuZhu Pets: Quest for Zhu	DS	2011.0	Misc	Activision	2.340740	1.525543	3.1038
16636	9196	Zoobles! Spring to Life!	DS	2011.0	Misc	Activision	2.697415	1.087977	2.7607
16640	9816	Zubo	DS	2008.0	Misc	Electronic Arts	2.592054	1.493442	1.4934

617 rows × 11 columns

Categorical-Categorical

Earlier we saw how to work with continous-continuous pair of variables.

Now let's come to the second type of pair of i.e. **Categorical-Categorical**.

What questions comes to your mind when we say categorical-categorical pair?

Questions related to distribution of a category within another category.

- What is the **distribution of genres for top-3 publishers**?
- Which **platforms do these top publishers use?**

Which plot can we use to show distribution of one category with respect to another?

We can have can **have multiple bars for each category**

These multiple bars can be

- stacked together - **Stacked Countplot**
- placed next to each other - **Dodged Countplot**

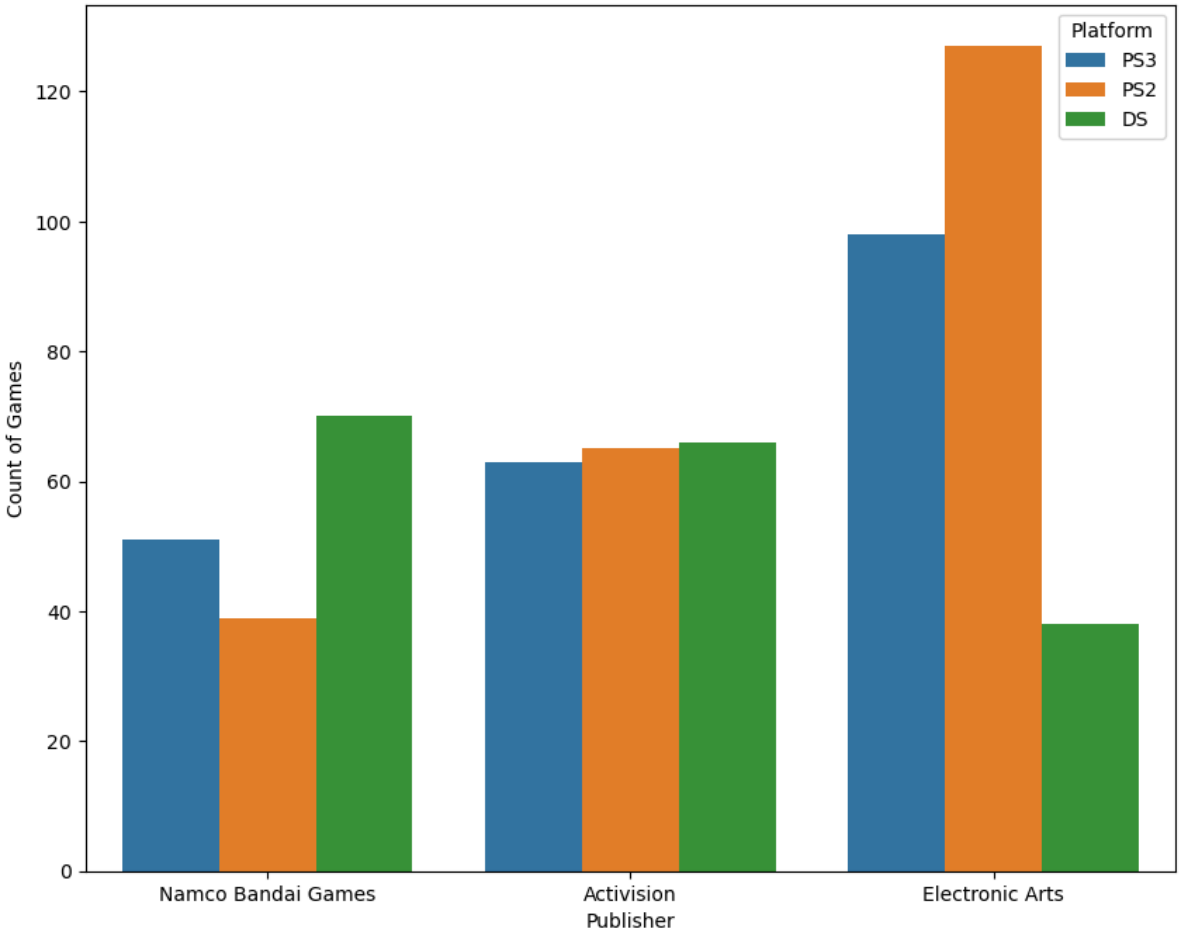
Dodged Countplot

How can we compare the top 3 platforms these publishers use?

- We can use a **dodged countplot** in this case.

In [24]:
```
plt.figure(figsize=(10,8))
sns.countplot(x='Publisher',hue='Platform',data=top3_data)
plt.ylabel('Count of Games')
```

Out[24]:
```
Text(0, 0.5, 'Count of Games')
```



What can we infer from the dodged countplot?

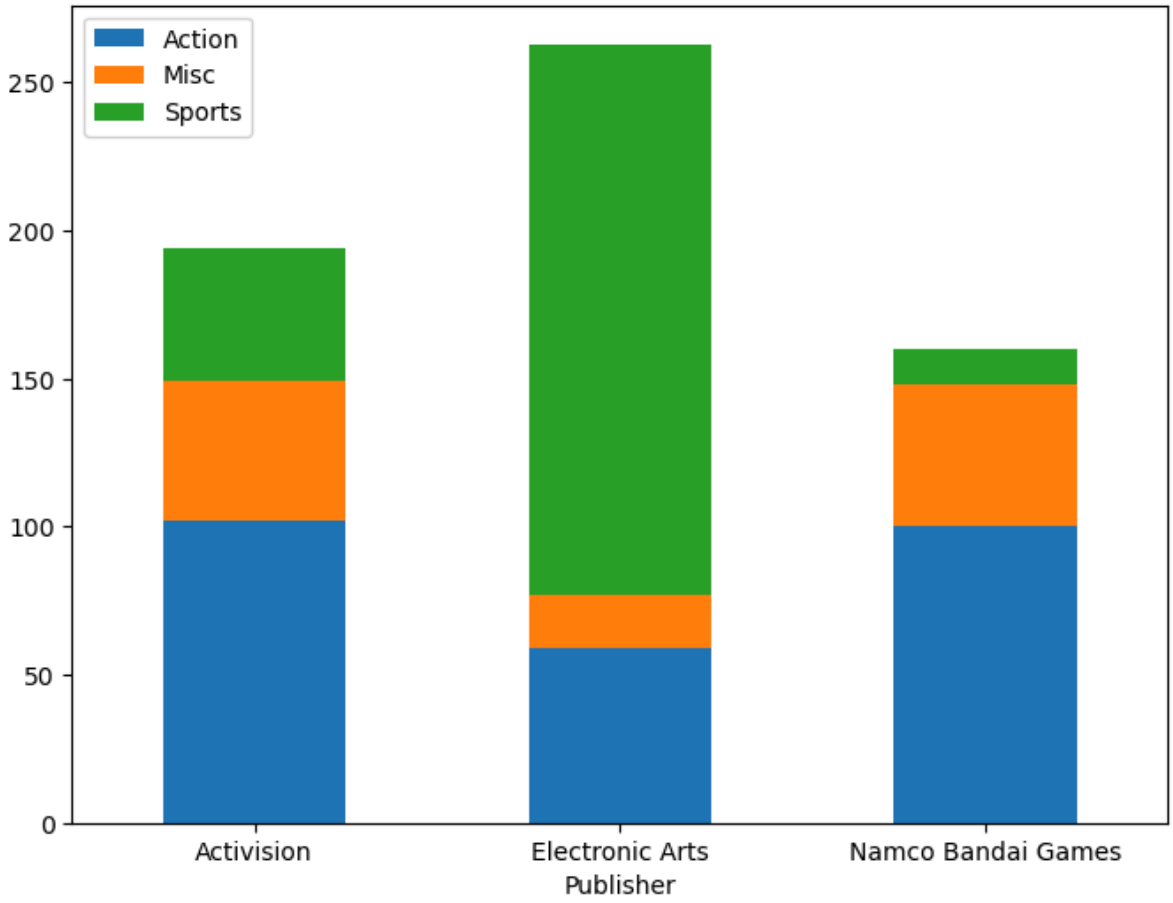
- EA releases PS2 games way more than any other publisher, or even platform!
- Activision has almost the same count of games for all 3 platforms.
- EA is leading in PS3 and PS2, but Namco leads when it comes to DS platform.

Stacked Countplot

How can we visualize the distribution of genres for top-3 publishers?

- We can use a **stacked countplot** for this.

In [25]:
```
df_stacked_plot = pd.crosstab(index=top3_data['Publisher'], columns=top3_dat

df_stacked_plot.plot(kind='bar', stacked=True, figsize=(8, 6))
plt.xticks(rotation=0)
plt.legend(loc='upper left')
plt.show()
```



But stacked countplots can be misleading.

Some may find it difficult to understand if it starts from baseline or from on top of the bottom area.

How do we decide between a Stacked countplot and Dodged countlot?

- Stacked countplots are a good way to represent totals.
- Dodged countplots helps us to comapare values between various categories, and within the category itself too.

Continous-Categorical

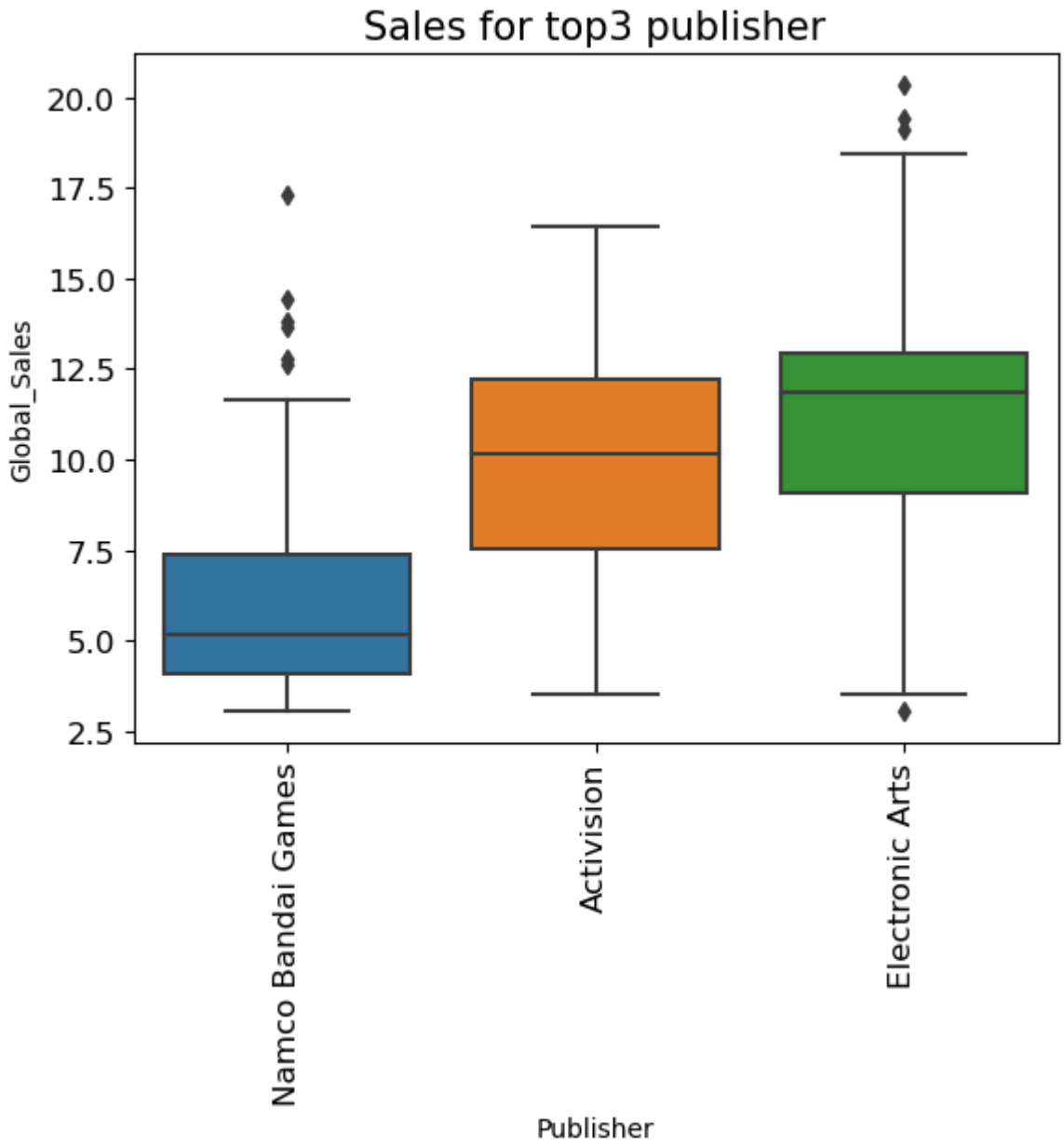
What kind of questions we may have regarding a continuous-categorical pair?

- We might to want calculate some numbers category-wise.
  - **What is the average sales for every genre?**
- We might be interested in checking the distribution of the data category-wise.
  - **What is the distribution of sales for the top3 publishers?**

What kind of plot can we make for every category?

- Either `KDE plot` or `Boxplot` per category.

In [27]:
```
sns.boxplot(x='Publisher', y='Global_Sales', data=top3_data)
plt.xticks(rotation=90,fontsize=12)
plt.yticks(fontsize=12)
plt.title('Sales for top3 publisher', fontsize=15)
plt.show()
```



What can we infer from this plot?

- The overall sales of EA is higher, with a much larger spread than other publishers.
- Activision doesn't have many outliers.
  - If you notice, even though the spread is lesser than EA, the median is almost the same.

Bar Plot

What if we want to compare the sales between the genres?

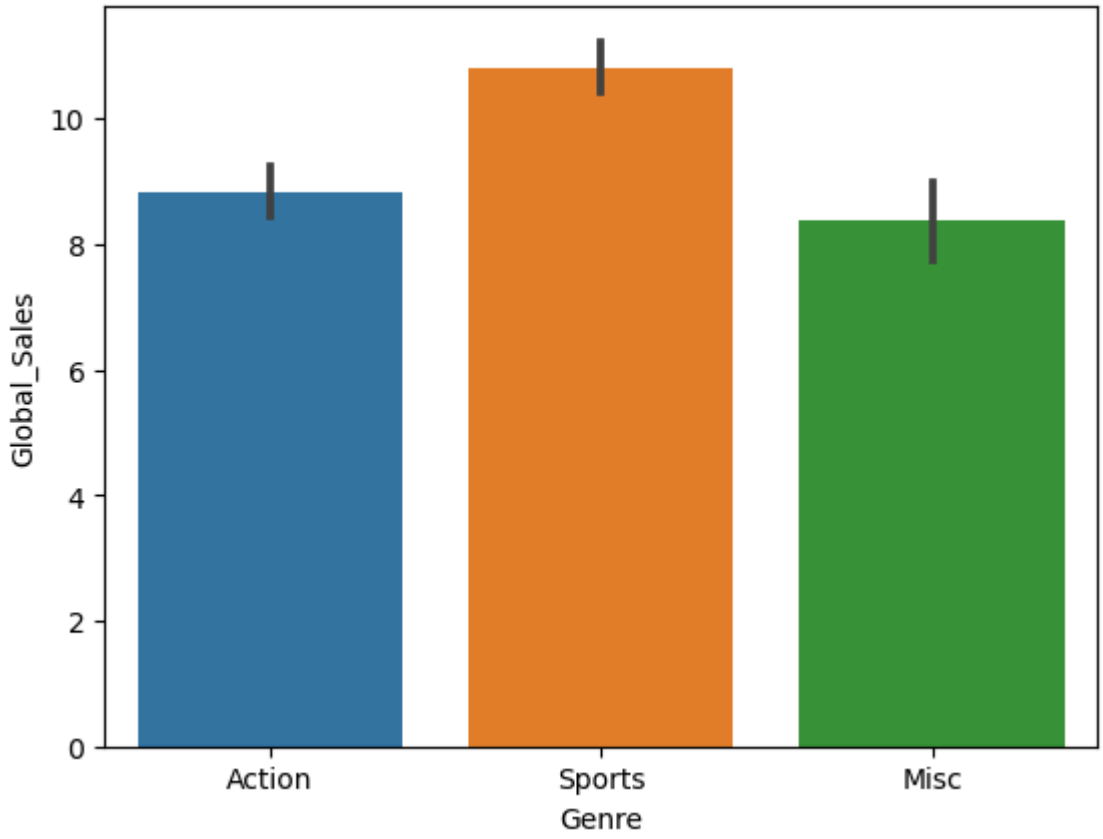
We have to use:

- Genre (categorical)
- Mean of global sales per genre (numerical)

How to visualize which genres bring higher average global sales?

In [28]:
```
sns.barplot(data=top3_data, x="Genre", y="Global_Sales", estimator=np.mean)
```

Out[28]:
```
<Axes: xlabel='Genre', ylabel='Global_Sales'>
```



What can we infer from this plot?

- If you remember, we had earlier seen EA had a larger market share of sales.
- Along with this fact, majority of games EA made was sports.
- This ultimately proves the fact that Sports has a high market share in the industry, as shown in the bar chart.

Subplots

So far we have **shown only 1 plot** using `plt.show()`.

Let's say we want to plot the trend of NA and every other region separately in a single figure.

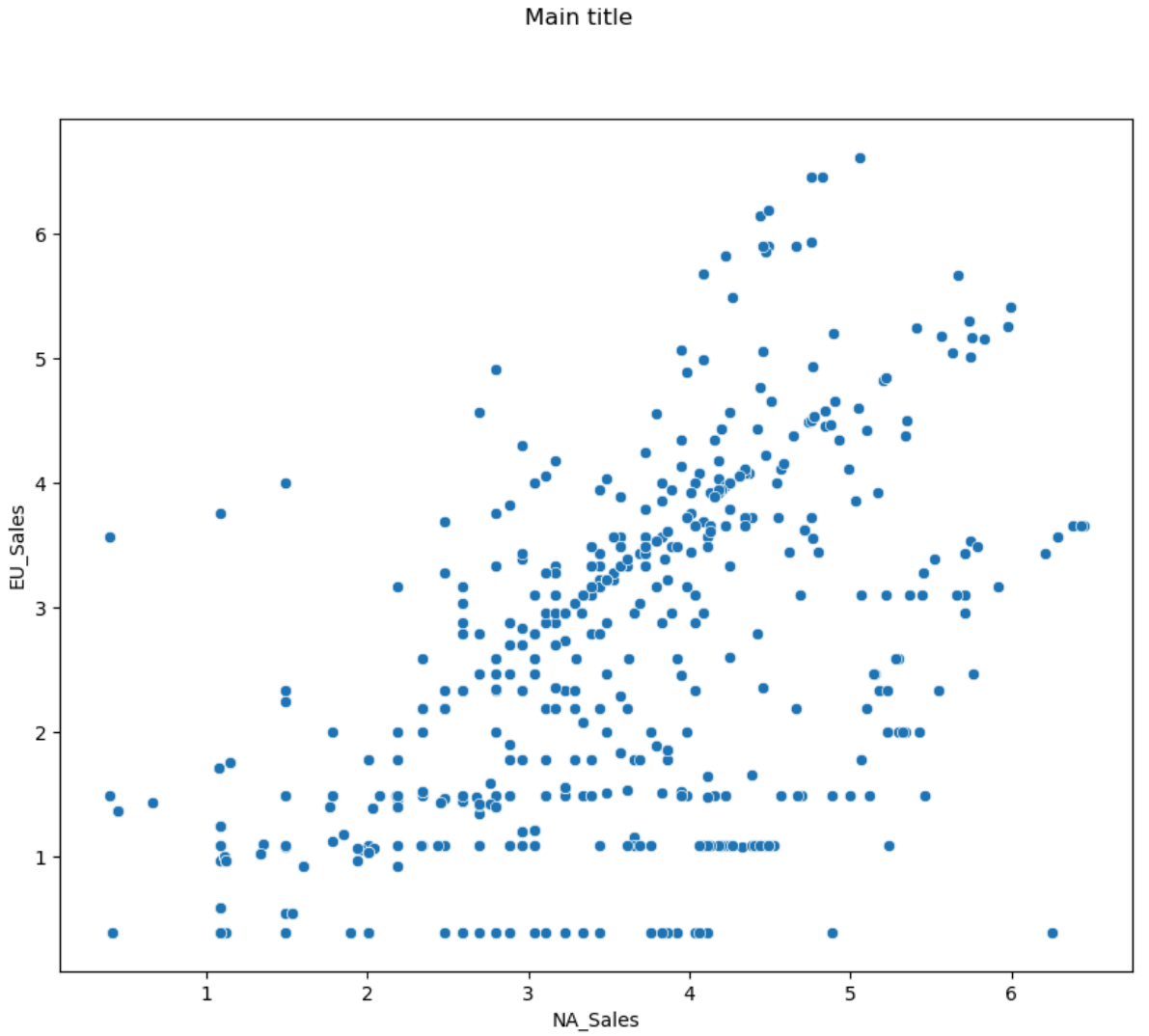
How can we plot multiple smaller plots at the same time?

We will use **subplots**, i.e., **divide the figure into smaller plots**.

- We will be using `plt.subplots()`

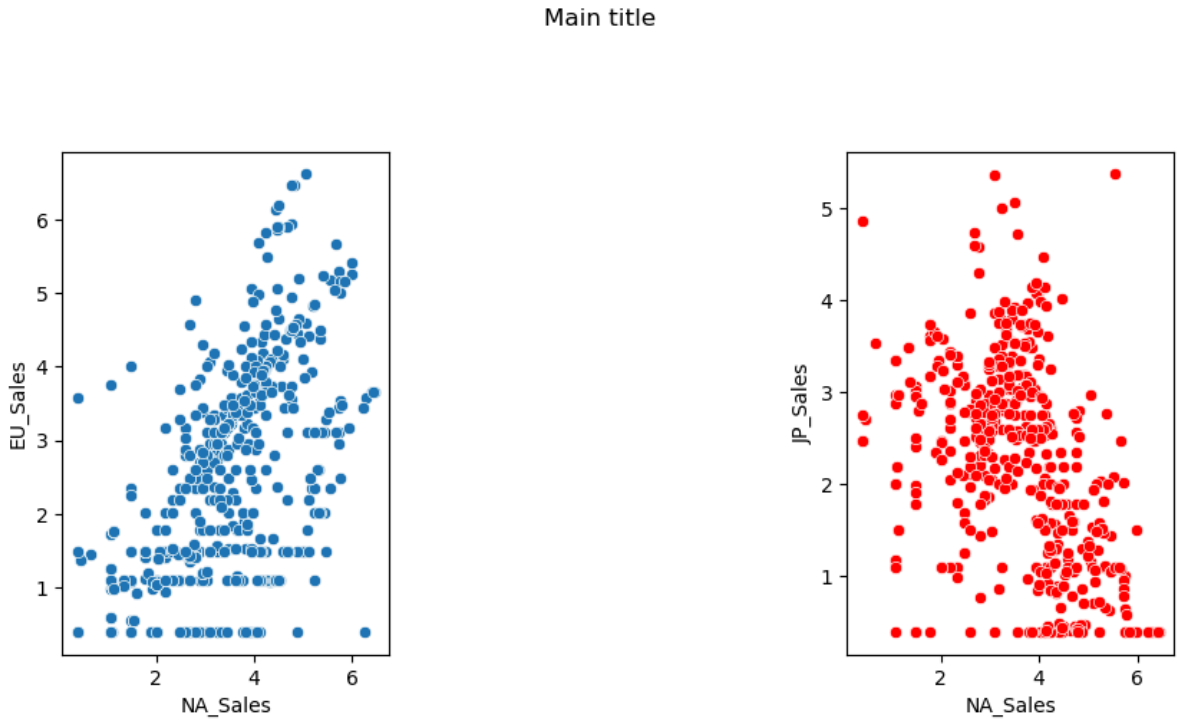
- It mainly takes 2 arguments -
  1. **No. of rows** we want to **divide our figure** into.
  2. **No. of columns** we want to **divide our figure** into.
- It returns
  - Figure
  - Numpy matrix of subplots

In [29]:
```
fig = plt.figure(figsize=(10,8))
sns.scatterplot(x=top3_data['NA_Sales'], y=top3_data['EU_Sales'])
fig.suptitle('Main title')
plt.show()
```



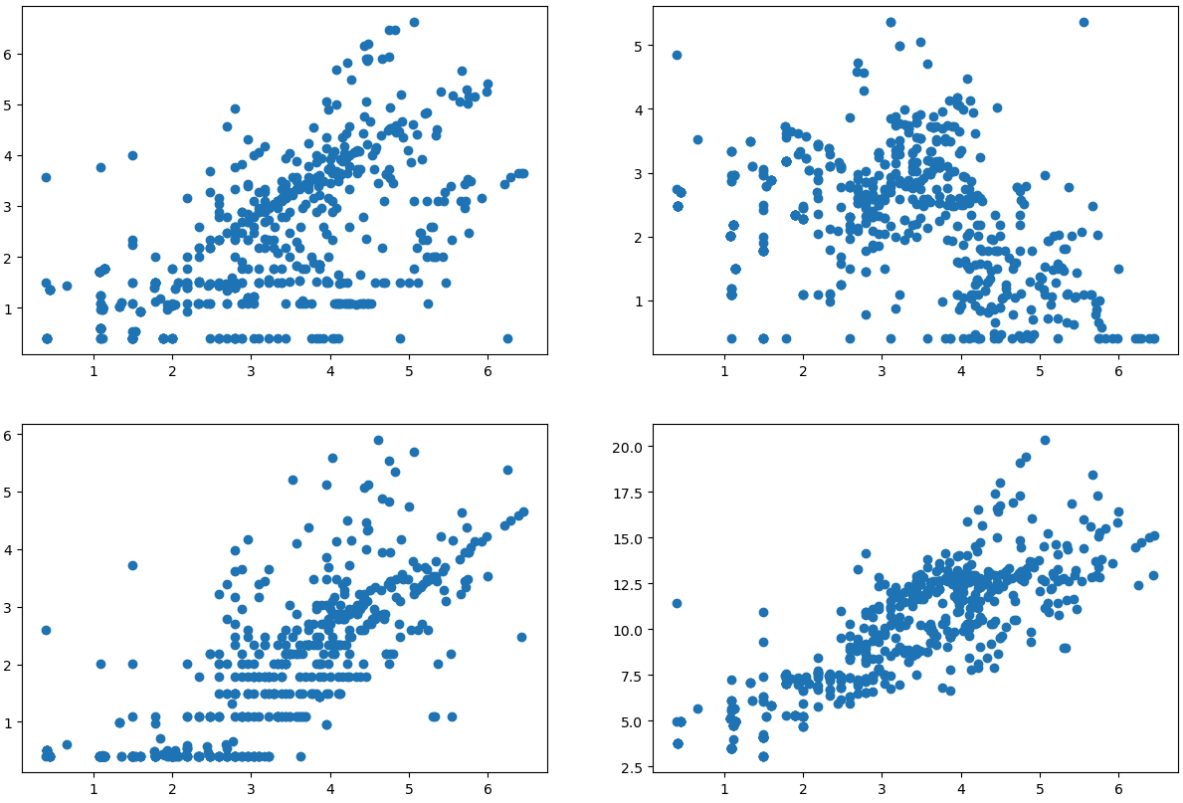
In [30]:
```
fig = plt.figure(figsize=(10,10))

plt.subplot(2, 3, 1)
sns.scatterplot(x='NA_Sales', y='EU_Sales', data=top3_data)

plt.subplot(2, 3, 3)
sns.scatterplot(x='NA_Sales', y='JP_Sales', data=top3_data, color='red')

fig.suptitle('Main title')
```

Out[30]:
```
Text(0.5, 0.98, 'Main title')
```



In [31]:
```
fig, ax = plt.subplots(2, 2, figsize=(15,10))

ax[0,0].scatter(top3_data['NA_Sales'], top3_data['EU_Sales'])
ax[0,1].scatter(top3_data['NA_Sales'], top3_data['JP_Sales'])
ax[1,0].scatter(top3_data['NA_Sales'], top3_data['Other_Sales'])
ax[1,1].scatter(top3_data['NA_Sales'], top3_data['Global_Sales'])
```

Out[31]:
```
<matplotlib.collections.PathCollection at 0x16189fb10>
```



Notice that we are using 2 numbers during each plotting.

Think of subplots as a `2x2 grids`, with the two numbers denoting `x,y` / `row,column` coordinate of each subplot.

What is this `ax` parameter exactly?

In [32]:
```
print(ax)
```

```
[[<Axes: > <Axes: >]
 [<Axes: > <Axes: >]]
```

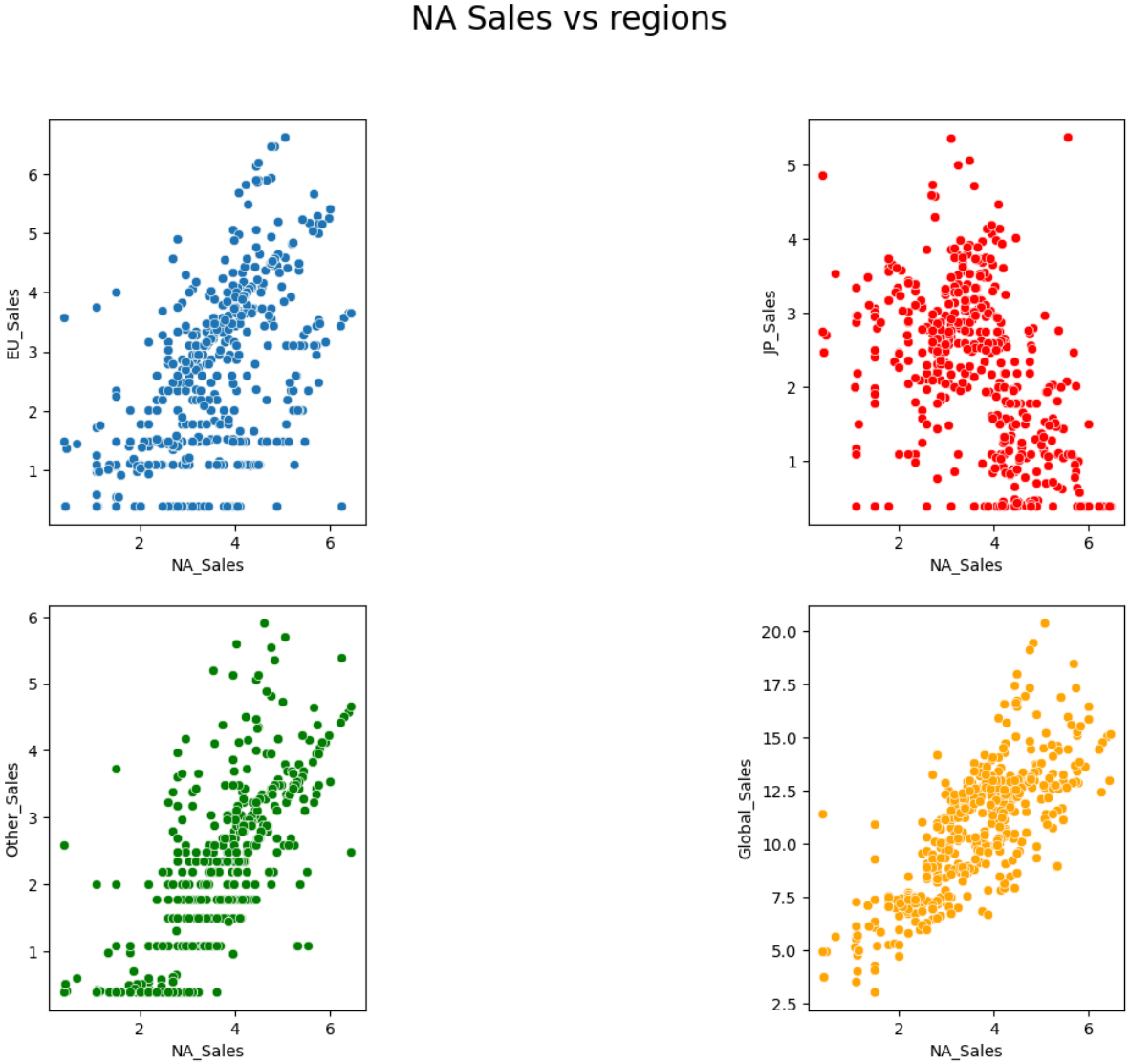
It's a `2x2 matrix` of multiple axes objects.

We are plotting each plot on a single `axes` object.

Hence, we are using a 2D notation to access each grid/axes object of the subplot.

Instead of accesing the individual axes using `ax[0, 0]` , `ax[1, 0]` , there is another method we can use too.

In [33]:
```
plt.figure(figsize=(12,10)).suptitle("NA Sales vs regions",fontsize=20)

# using a 2x3 subplot
plt.subplot(2, 3, 1)
sns.scatterplot(x='NA_Sales', y='EU_Sales', data=top3_data)

plt.subplot(2, 3, 3)
sns.scatterplot(x='NA_Sales', y='JP_Sales', data=top3_data, color='red')

plt.subplot(2, 3, 4)
sns.scatterplot(x='NA_Sales', y='Other_Sales', data=top3_data, color='green

plt.subplot(2, 3, 6)
sns.scatterplot(x='NA_Sales', y='Global_Sales', data=top3_data, color='orang

plt.show()
```



`Suptitle` adds a title to the whole figure.

We need to observe a few things here -

1. The 3rd paramter defines the position of the plot.
2. The position/numbering starts from 1.
3. It goes on row-wise from start of row to its finish.
4. Empty subplots don't show any axes.

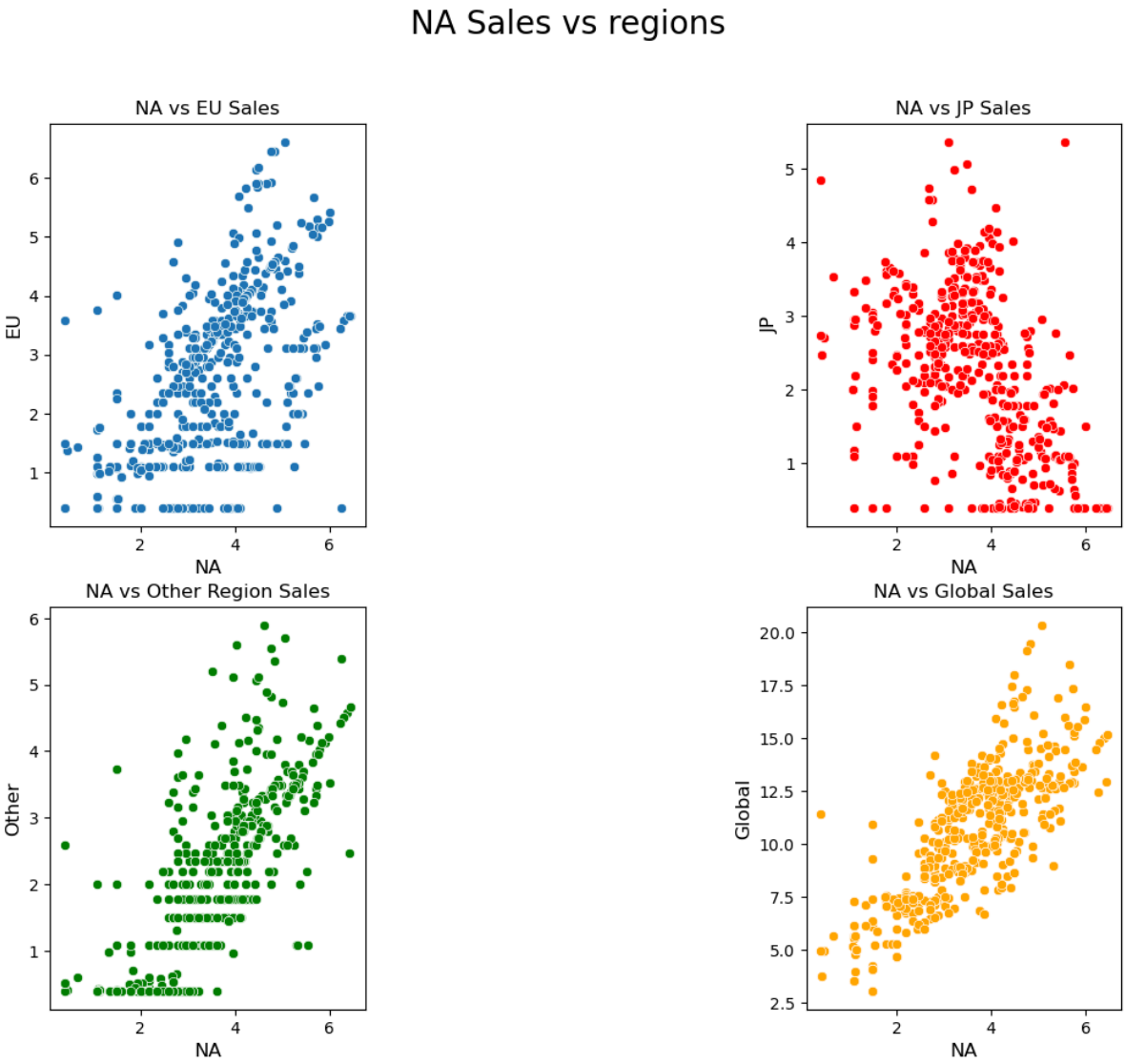
But how do we know which plot belongs to which category?

- Basically the context of each plot.

We can use `title`, `x/y label` and every other functionality for the subplots too.

In [34]:
```
plt.figure(figsize=(12,10)).suptitle("NA Sales vs regions",fontsize=20)

# using a 2x3 subplot
plt.subplot(2, 3, 1)
sns.scatterplot(x='NA_Sales', y='EU_Sales', data=top3_data)
plt.title('NA vs EU Sales', fontsize=12)
plt.xlabel('NA', fontsize=12)
plt.ylabel('EU', fontsize=12)

plt.subplot(2, 3, 3)
```

```
sns.scatterplot(x='NA_Sales', y='JP_Sales', data=top3_data, color='red')
plt.title('NA vs JP Sales', fontsize=12)
plt.xlabel('NA', fontsize=12)
plt.ylabel('JP', fontsize=12)

plt.subplot(2, 3, 4)
sns.scatterplot(x='NA_Sales', y='Other_Sales', data=top3_data, color='green
plt.title('NA vs Other Region Sales', fontsize=12)
plt.xlabel('NA', fontsize=12)
plt.ylabel('Other', fontsize=12)

plt.subplot(2, 3, 6)
sns.scatterplot(x='NA_Sales', y='Global_Sales', data=top3_data, color='orang
plt.title('NA vs Global Sales', fontsize=12)
plt.xlabel('NA', fontsize=12)
plt.ylabel('Global', fontsize=12)

plt.show()
```



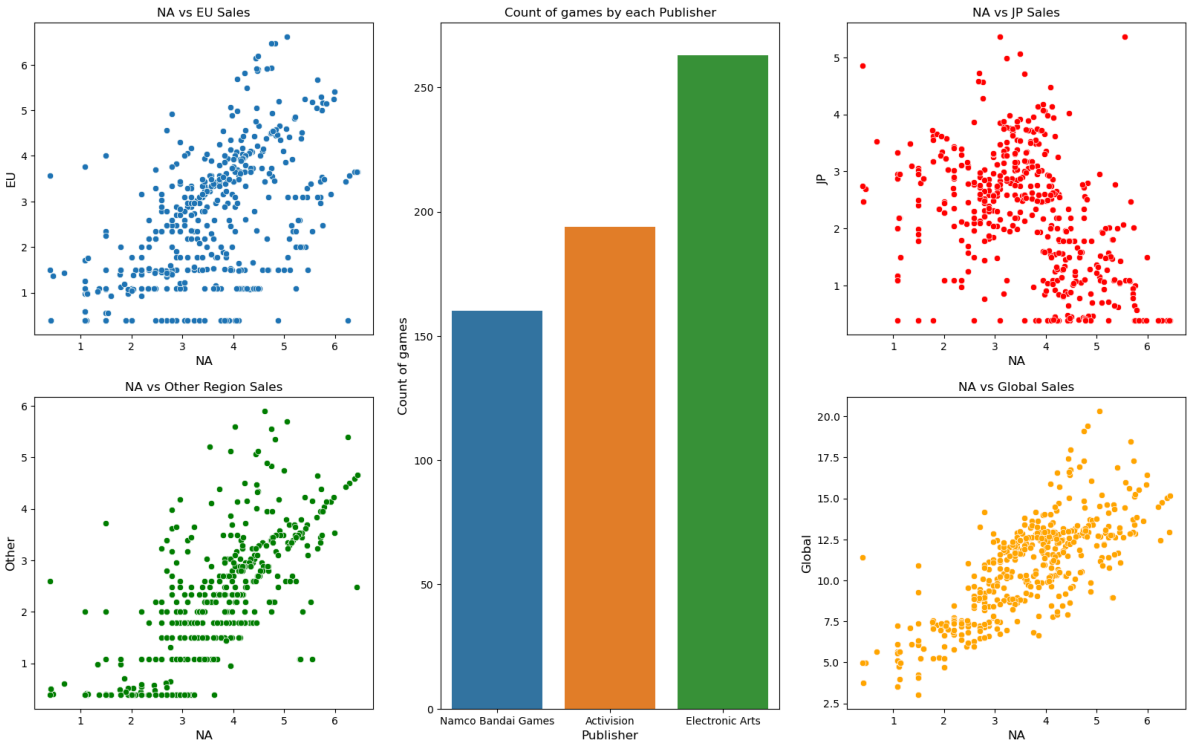
What if we want to span a plot across the full length of the plot?

- Think of this in **terms of a grid**.
- Currently we are **dividing our plot into 2 rows and 3 columns**.
- But we want our plot to be across the middle column, with **grids 2 and 5**.
- This can be said as a **single column**.

So this problem can be simplified to plotting the plot across **second column in a 1 row 3 column subplot**.

In [35]:
```
plt.figure(figsize=(20,12)).suptitle("Video Games Sales Dashboard",fontsize=

# using a 2x3 subplot
plt.subplot(2, 3, 1)
sns.scatterplot(x='NA_Sales', y='EU_Sales', data=top3_data)
plt.title('NA vs EU Sales', fontsize=12)
plt.xlabel('NA', fontsize=12)
plt.ylabel('EU', fontsize=12)

plt.subplot(2, 3, 3)
sns.scatterplot(x='NA_Sales', y='JP_Sales', data=top3_data, color='red')
plt.title('NA vs JP Sales', fontsize=12)
plt.xlabel('NA', fontsize=12)
plt.ylabel('JP', fontsize=12)

# Countplot of publishers
plt.subplot(1,3,2)
sns.countplot(x='Publisher', data=top3_data)
plt.title('Count of games by each Publisher', fontsize=12)
plt.xlabel('Publisher', fontsize=12)
plt.ylabel('Count of games', fontsize=12)

plt.subplot(2, 3, 4)
sns.scatterplot(x='NA_Sales', y='Other_Sales', data=top3_data, color='green
plt.title('NA vs Other Region Sales', fontsize=12)
plt.xlabel('NA', fontsize=12)
plt.ylabel('Other', fontsize=12)

plt.subplot(2, 3, 6)
sns.scatterplot(x='NA_Sales', y='Global_Sales', data=top3_data, color='orang
plt.title('NA vs Global Sales', fontsize=12)
plt.xlabel('NA', fontsize=12)
plt.ylabel('Global', fontsize=12)

plt.show()
```

Video Games Sales Dashboard



Question-1

Q. For the company "Toyota", we want to find which type of vehicle has made the maximum sales.

Which plot we will prefer to use here?

a. Bar Plot
b. Pie Chart
c. Boxplot
d. Line Plot

Answer: Bar Plot

Explanation:

A bar plot is ideal for comparing the quantities of different categories or groups, making it suitable for visualizing sales data across different types of vehicles. Each type of vehicle can be represented on the x-axis, while the sales quantity (or revenue) can be represented on the y-axis. This allows for a clear comparison of sales performance among different vehicle types, making it easy to identify which type has made the maximum sales.

Question-2

Q. Apple wanted to conduct an analysis and find the relationship between price and number of units sold for it's products.

Which of the following plots would you prefer?

a. Scatter Plot
b. Pie Chart
c. Boxplot
d. Line Plot

Answer: Scatter Plot

Explanation:

A scatter plot is specifically designed for visualizing the relationship between two continuous variables. In this case, you can plot the price of the products on the x-axis and the corresponding number of units sold on the y-axis. Each data point represents a product, with its price and the corresponding number of units sold. By examining the distribution of points on the plot, you can discern any patterns or trends in the relationship between price and sales volume for Apple products.

In [ ]:

Data Viz 3

Content

- Multivariate Data Visualization
  - CCN
  - CNN
  - NNN
- Joint plot
- Pairplot
- Correlation Heatmap

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
data = pd.read_csv('final.csv')
data.head()
```

Out[2]:

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sa
0	2061	1942	NES	1985.0	Shooter	Capcom	4.569217	3.033887	3.439
1	9137	¡Shin Chan Flipa en colores!	DS	2007.0	Platform	505 Games	2.076955	1.493442	3.033
2	14279	.hack: Sekai no Mukou ni + Versus	PS3	2012.0	Action	Namco Bandai Games	1.145709	1.762339	1.493
3	8359	.hack//G.U. Vol.1//Rebirth	PS2	2006.0	Role-Playing	Namco Bandai Games	2.031986	1.389856	3.228
4	7109	.hack//G.U. Vol.2//Reminisce	PS2	2006.0	Role-Playing	Namco Bandai Games	2.792725	2.592054	1.440

If you remember, `Genres`, `Publisher` and `Platform` were categorical values.

In [3]:
```
top3_pub = data['Publisher'].value_counts().index[:3]
top3_gen = data['Genre'].value_counts().index[:3]
top3_plat = data['Platform'].value_counts().index[:3]
top3_data = data.loc[(data["Publisher"].isin(top3_pub)) & (data["Platform"]
top3_data
```

Out[3]:

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sal
2	14279	.hack: Sekai no Mukou ni + Versus	PS3	2012.0	Action	Namco Bandai Games	1.145709	1.762339	1.4934
13	2742	[Prototype 2]	PS3	2012.0	Action	Activision	3.978349	3.727034	0.8488
16	1604	[Prototype]	PS3	2009.0	Action	Activision	4.569217	4.108402	1.1872
19	1741	007: Quantum of Solace	PS3	2008.0	Action	Activision	4.156030	4.346074	1.0879
21	4501	007: Quantum of Solace	PS2	2008.0	Action	Activision	3.228043	2.738800	2.5855
...	...	...	...	...	...	...	...	...	
16438	14938	Yes! Precure 5 Go Go Zenin Shu Go! Dream Festival	DS	2008.0	Action	Namco Bandai Games	1.087977	0.592445	1.0879
16479	10979	Young Justice: Legacy	PS3	2013.0	Action	Namco Bandai Games	2.186589	1.087977	3.4090
16601	11802	ZhuZhu Pets: Quest for Zhu	DS	2011.0	Misc	Activision	2.340740	1.525543	3.1038
16636	9196	Zoobles! Spring to Life!	DS	2011.0	Misc	Activision	2.697415	1.087977	2.7607
16640	9816	Zubo	DS	2008.0	Misc	Electronic Arts	2.592054	1.493442	1.4934

617 rows × 11 columns

Multivariate Data Visualization

Let's try to add a 3rd variable on the top of the plots that we have seen so far.

NNC

How can we visualize the correlation between NA and EU, but for different genres?

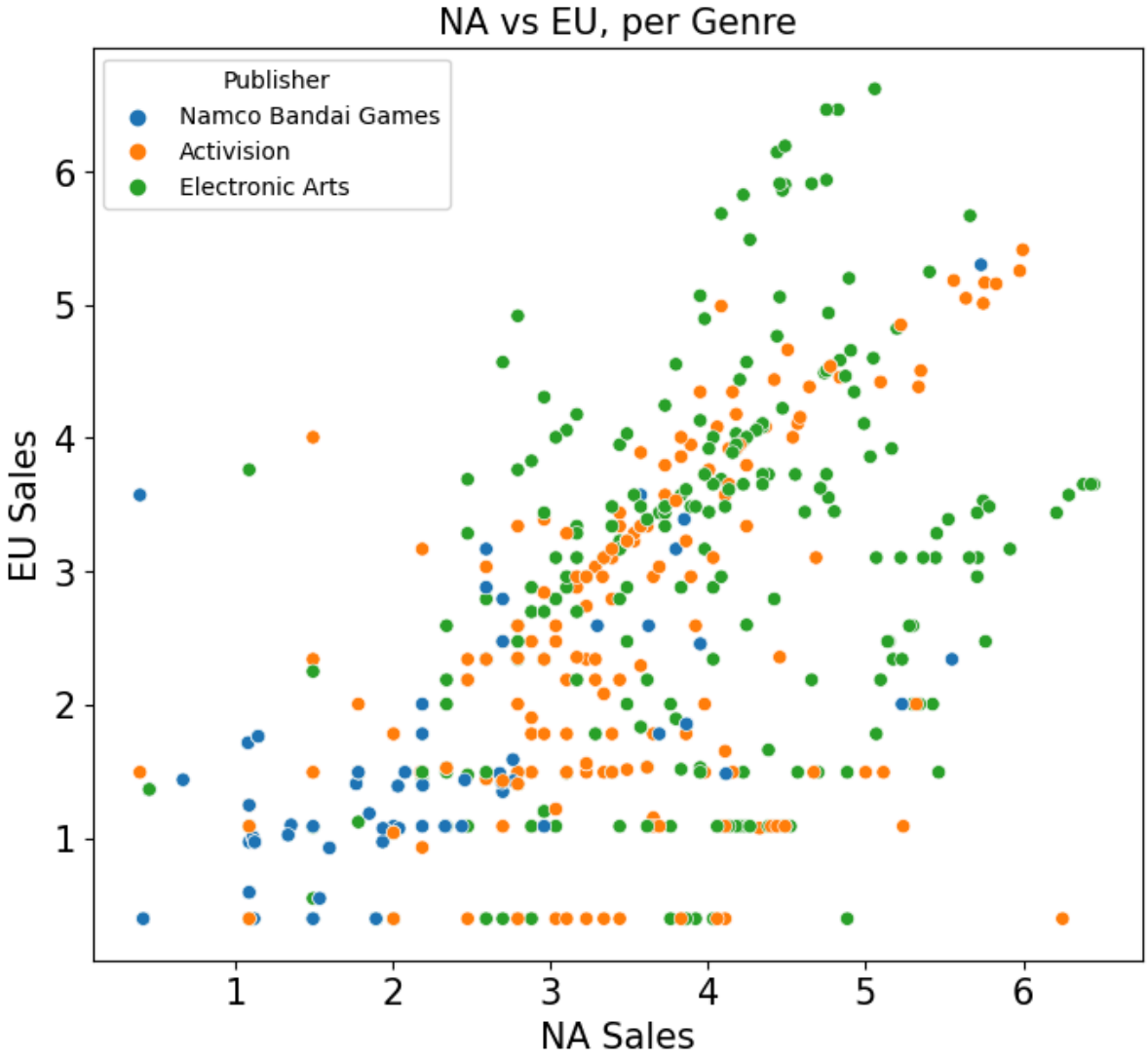
Here, we have two numerical and one categorical variable!

- Numerical-Numerical → Scatter plot, need to add info about one categorical variable.
- Numerical-Categorical → Boxplot, need to add info about one numerical variable.

Let's ask two questions.

- Is it possible to add information about a continuous variable upon boxplots?
  - No
- Is it possible to add information about a categorical variable on scatterplot?
  - Yes (using colors)"

In [4]:
```
plt.figure(figsize=(8,7))
sns.scatterplot(x='NA_Sales', y='EU_Sales',hue='Publisher',data=top3_data)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.xlabel('NA Sales',fontsize=15)
plt.ylabel('EU Sales',fontsize=15)
plt.title('NA vs EU, per Genre', fontsize=15)
plt.show()
```



Inferences:

- If we see this plot, we can notice now that Namco has lower sales correlation, while Activision has a concentrated positivee correlation.
- EA also has positive correlation, but it's more spread compared to Activision.

CCN

How will you visualize the global sales for each publisher, but separated by genres?

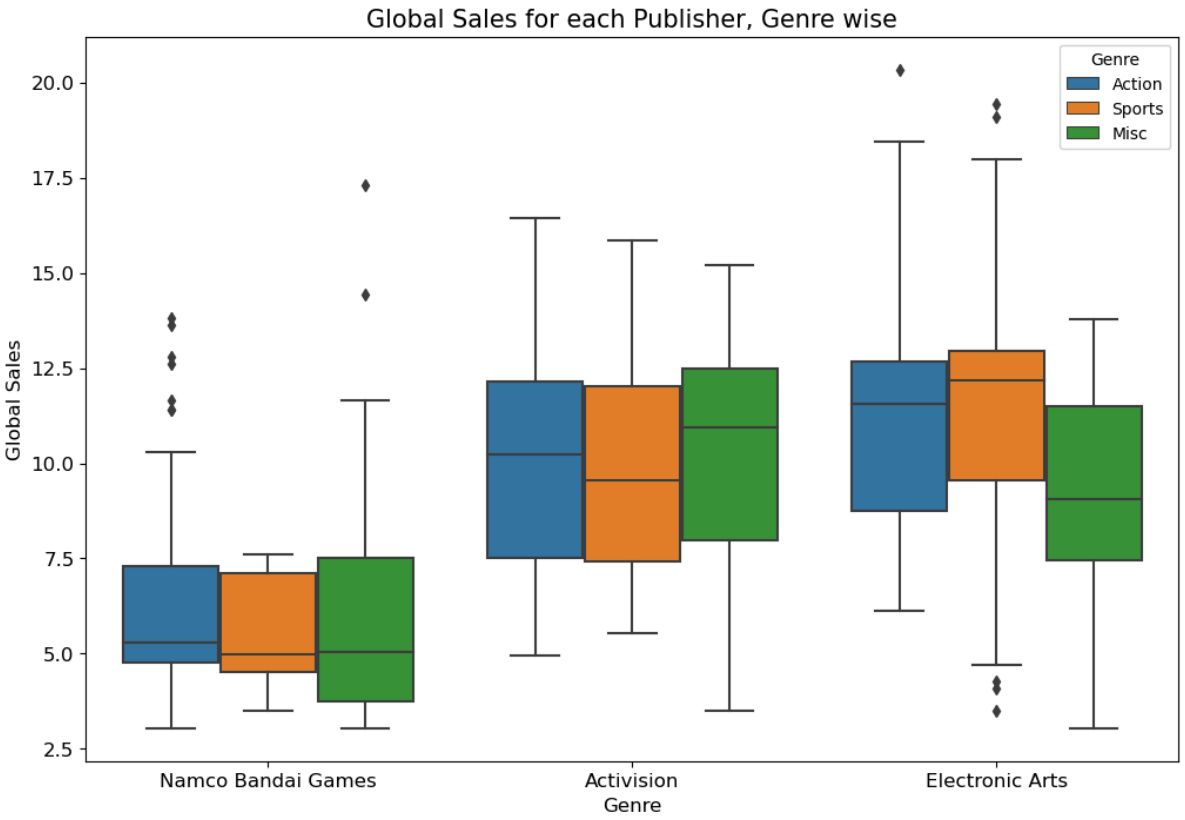
We have two categorical and one numerical data here!

- Categorical-Categorical → Stacked Bar plot, need to add info about one continuous feature.
- Categorical-Numerical → Boxplot, need to add categorical variable.

Which one is easier and possible?

We can add one categorical variable by "dodging" multiple boxplots.

In [5]:
```
plt.figure(figsize=(12,8))
sns.boxplot(x='Publisher',y='Global_Sales',hue='Genre',data=top3_data)
plt.xlabel('Genre', fontsize=12)
plt.ylabel('Global Sales', fontsize=12)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.title('Global Sales for each Publisher, Genre wise', fontsize=15)
plt.show()
```



Inferences:

- Namco has lower median sales in every genre as compared to all publishers.
- Looking at the Action genre, even though EA and Activision have almost similar medians, Action is more spread in EA.
- An interesting thing to notice here is that for each of the three publishers, three different genre of games have higher sales median.
  - Namco: Action
  - Activision: Misc
  - EA: Sports

NNN

So far we have seen how NA and EU are correlated with each other.

But how can we compare the data when we have 3 numerical variables?

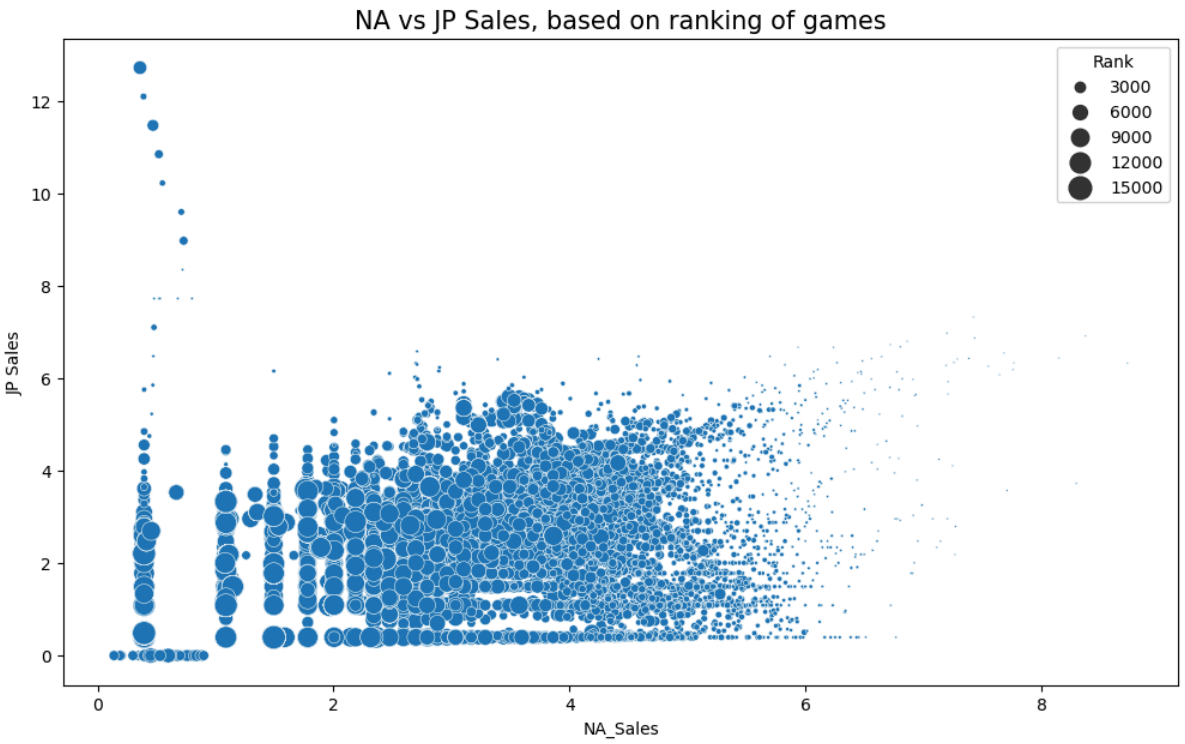
How does rank affect the correlation between NA and EU Sales?

We have used scatter plot for two numerical features.

We have two options here -

- Make a 3D Scatterplot
  - Good for 3D visualization, but tough to report/show in static setting.
- Add info about the 3rd feature on the 2D scatter plot itself.
  - Bubble Chart

In [6]:
```
plt.figure(figsize=(12,7))
sns.scatterplot(x='NA_Sales', y='JP_Sales', size='Rank', sizes=(1, 200), dat
plt.xlabel('NA_Sales',fontsize=10)
plt.ylabel('JP Sales', fontsize=10)
plt.title('NA vs JP Sales, based on ranking of games', fontsize=15)
plt.show()
```



Inferences:

- Interestingly, we can notice that higher ranking games are actually on the lower scale of sales, while lower ranking games are high on the sales side.

Joint Plot

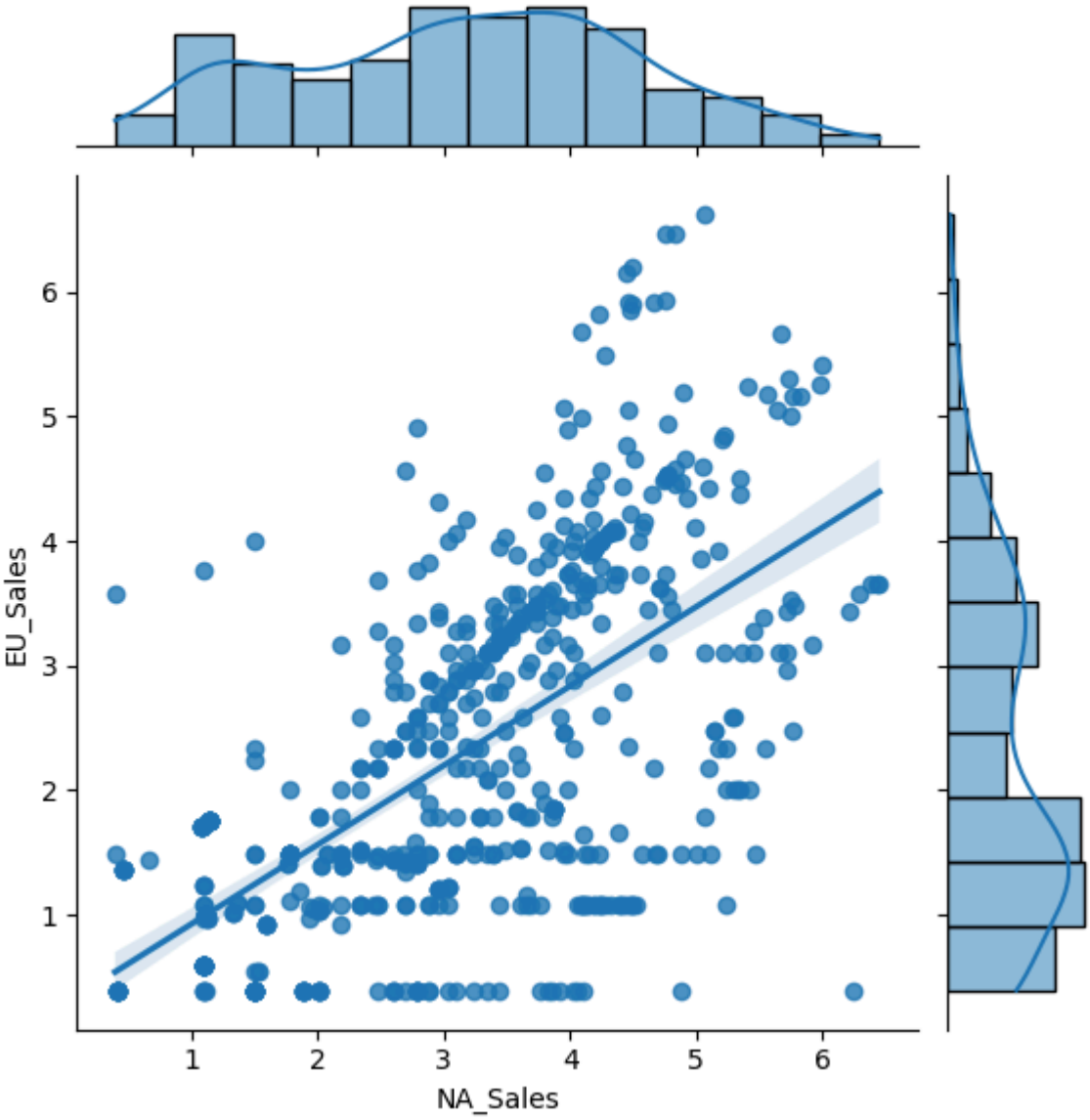
- `jointplot()` draws a plot between two variables.

- It shows scatter plot, histogram and KDE plot in the same plot.

Let's check it out -

- We will take **`NA_Sales` as x-coordinates** and **`EU_Sales` as y-coordinates**.
- We can select from different values for parameter `kind and it will plot accordingly.
  - "scatter" | "kde" | "hist" | "hex" | "reg" | "resid"
- We will set the `kind` parameter to **'reg'** here.

In [7]:
```
sns.jointplot(x='NA_Sales', y='EU_Sales', kind='reg', data=top3_data)
plt.show()
```



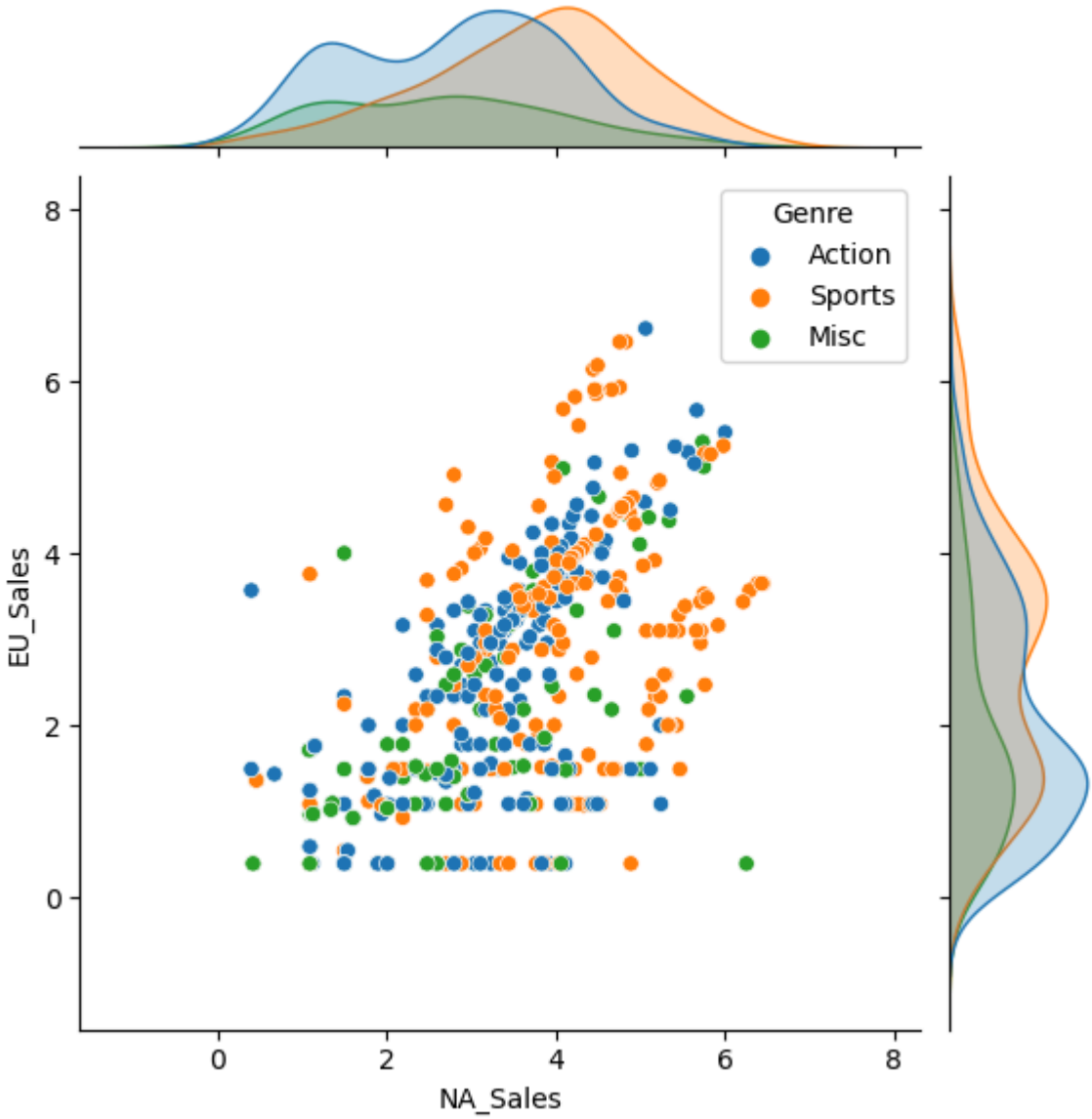
As we can see here,

- `jointplot` plots scatter plot, histogram and KDE plot in the same graph, when we set `kind=reg`.
- Scatter plot shows the **scattering of (`NA_Sales`, `EU_Sales`) pairs as (x, y) points**.
- Histogram and KDE plot show the separate distributions of `NA_Sales` and `EU_Sales` in the data.

We can also add hue to Joint plot.

Let's check how the 3 genres of games are distributed in terms of `NA_Sales` and `EU_Sales`.

In [8]:
```
sns.jointplot(x='NA_Sales', y='EU_Sales', data=top3_data, hue='Genre')
plt.show()
```



Pair Plot

- `pairplot()` creates a **grid of Axes by default**.
- Each numeric attribute in `data` is shared across **the y-axes across a single row** and the **x-axes across a single column**.
- It displays a **scatterplot between each pair of attributes in the data** with different **hue** for each category.

Since the diagonal plots belong to same attribute at both x and y axis, they are treated differently.

A univariate distribution plot is drawn to show the marginal distribution of the data in each column.

In [9]:
```
sns.pairplot(data=top3_data, hue='Genre')
plt.show()
```

Notice that,

- It is **like a scatter plot of video games with `hue='Genre'`**
- But it is **plotted between every pair of attributes**.
- **Color Legends** for each genre category are given on the **right side**.

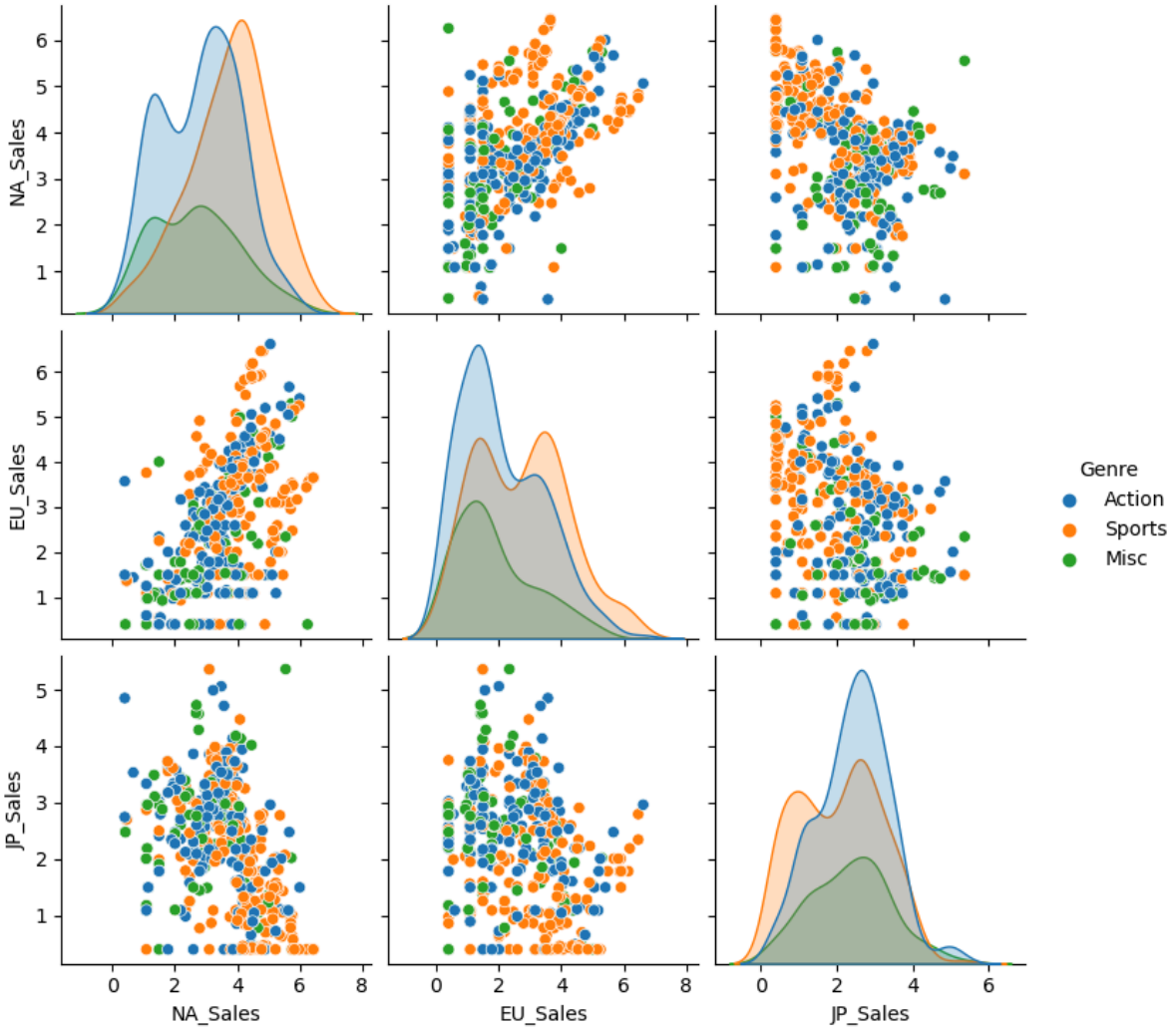
Diagonal plots are different from scatter plots because x and y axis have same attribute.

Diagonal plots show a univariate curve category-wise for each attribute.

You can also customize the pairplot to display only a selected subset of variables.

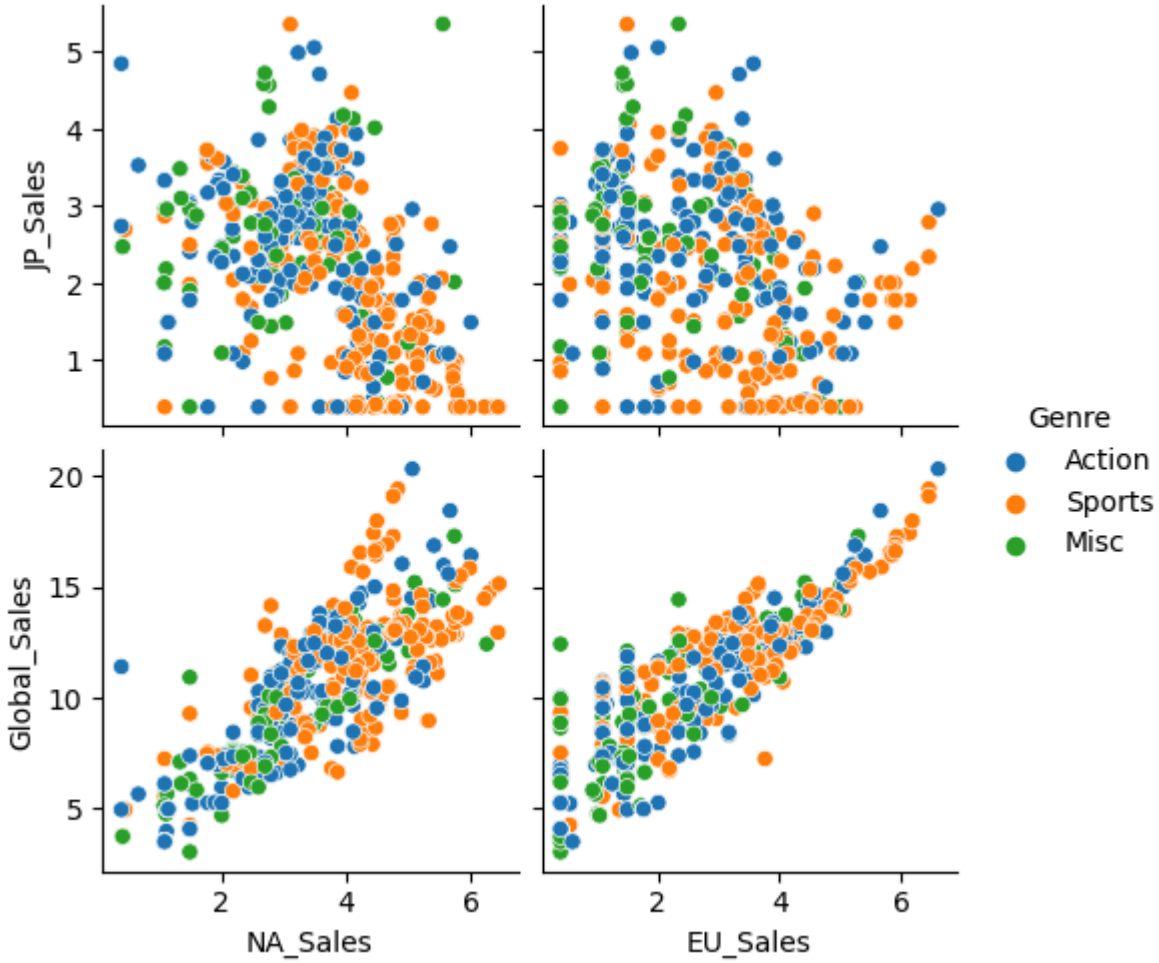
In [10]:
```
sns.pairplot(data=top3_data, vars=['NA_Sales', 'EU_Sales', 'JP_Sales'], hue=
```

Out[10]:
```
<seaborn.axisgrid.PairGrid at 0x16c54db90>
```



In [11]:
```
sns.pairplot(data=top3_data, x_vars=['NA_Sales', 'EU_Sales'], y_vars=['JP_Sa
```

Out[11]:
```
<seaborn.axisgrid.PairGrid at 0x178c1ae90>
```



Correlation Matrix

We can find the level of correlation b/w different attributes (variables).

But what exactly is a correlation?

- Two variables are said to be correlated when **they change in the same/opposite direction**.

We can check the **correlation coefficient** using `corr()` method.

In [12]:
```
num_df = top3_data.select_dtypes(include=[float,int])
num_df.corr()
```

Out[12]:

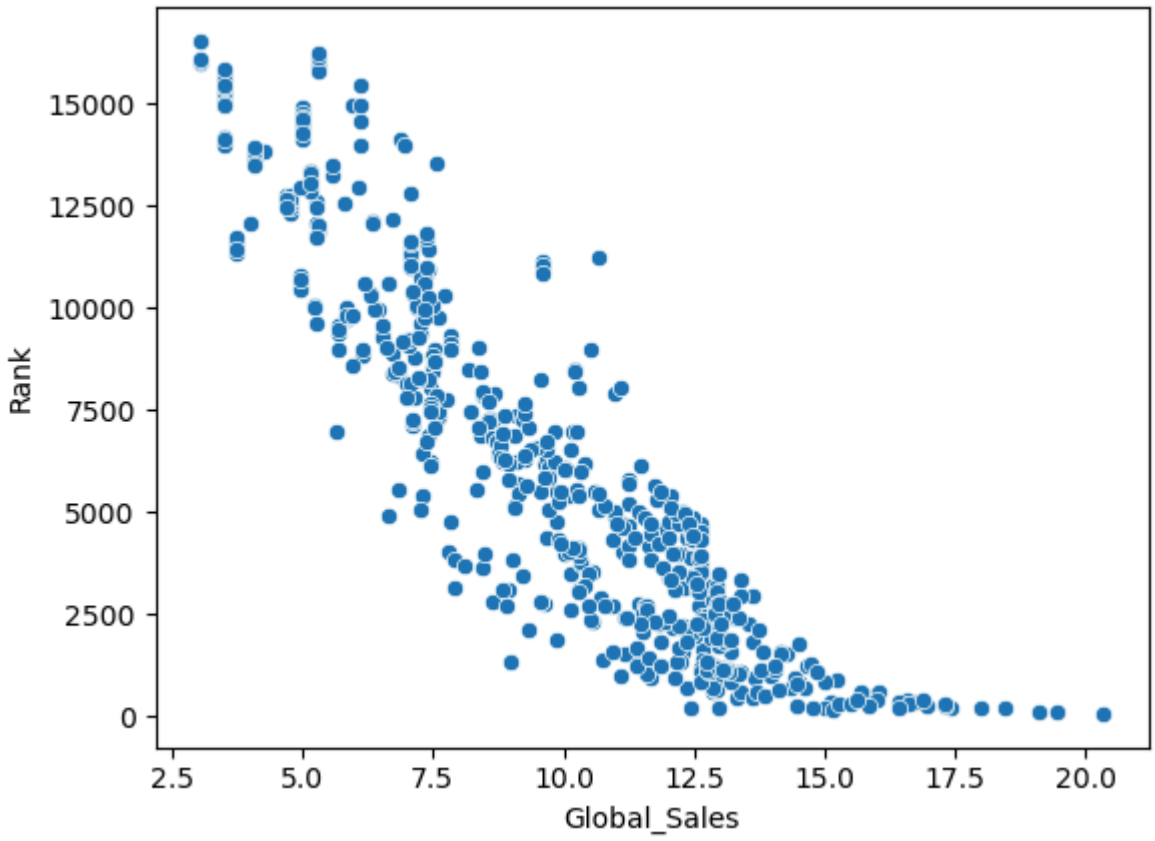
	Rank	Year	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_S
Rank	1.000000	0.328705	-0.873726	-0.735711	0.115459	-0.857567	-0.91
Year	0.328705	1.000000	-0.354256	-0.178026	0.055864	-0.239876	-0.28
NA_Sales	-0.873726	-0.354256	1.000000	0.617483	-0.233315	0.794353	0.856
EU_Sales	-0.735711	-0.178026	0.617483	1.000000	-0.208249	0.771105	0.86
JP_Sales	0.115459	0.055864	-0.233315	-0.208249	1.000000	-0.355825	-0.01
Other_Sales	-0.857567	-0.239876	0.794353	0.771105	-0.355825	1.000000	0.87
Global_Sales	-0.911721	-0.280351	0.856300	0.864147	-0.014193	0.878816	1.000

- Higher the **magnitude** of coefficient of correlation, more the variables are **correlated**.
- Note that the **sign just determines the direction of change**.
  - `+` means increase in value of one variable causes increase in value of other variable.
  - `-` means increase in value of one variable causes decrease in value of other variable, and vice versa.

As you can see, `Global Sales` and `Rank` have the highest correlation coefficient of -0.91.

Let's plot it using a scatter plot.

In [13]:
```
sns.scatterplot(x= 'Global_Sales', y= 'Rank', data = num_df)
plt.show()
```



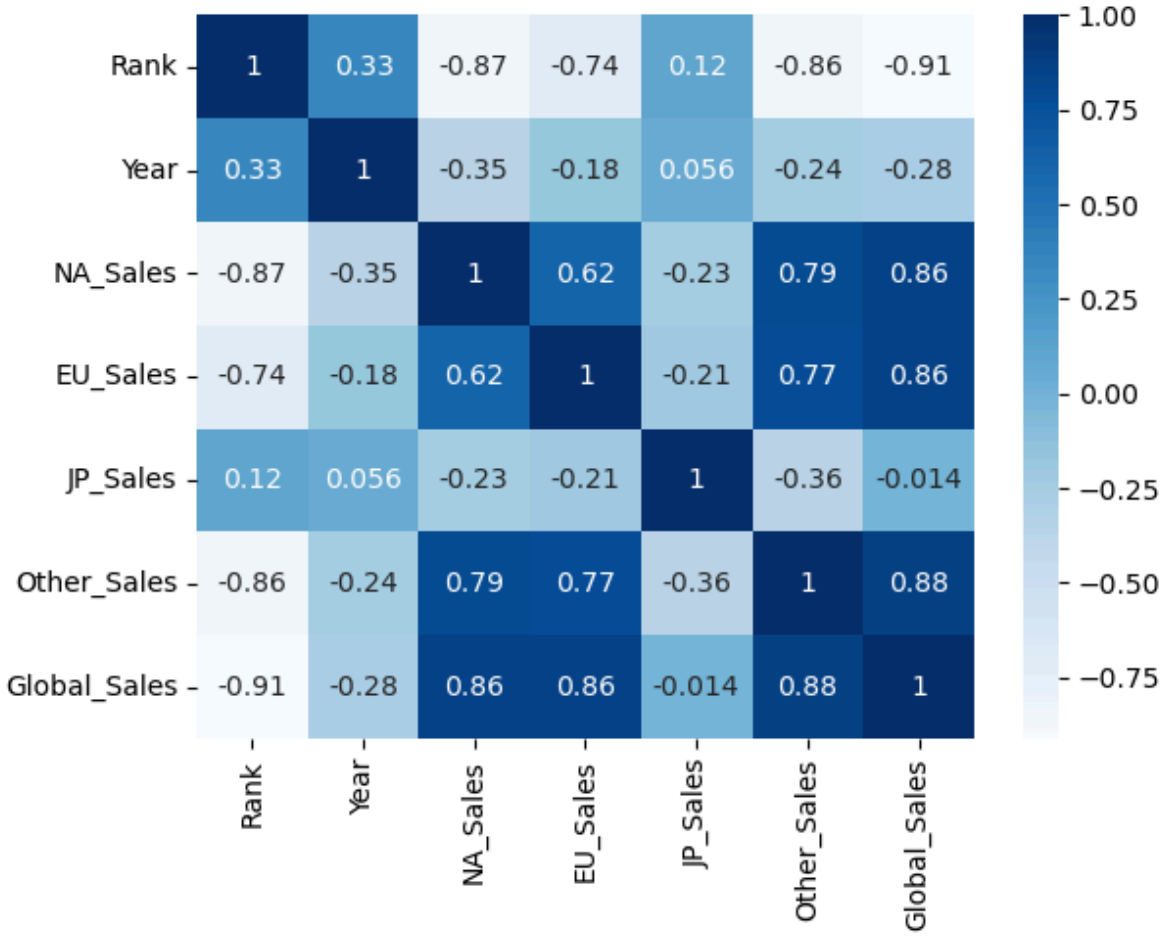
Now let's look at a way to visualize correlation among variables.

Heat Map

- A heat map plots rectangular data as a color-encoded matrix.
- The **more intense the color, the stronger the correlation** between the variables.

Let's plot a Heat Map using the correlation matrix generated using `corr()`.

In [14]:
```
sns.heatmap(num_df.corr(), cmap= "Blues", annot=True)
plt.show()
```

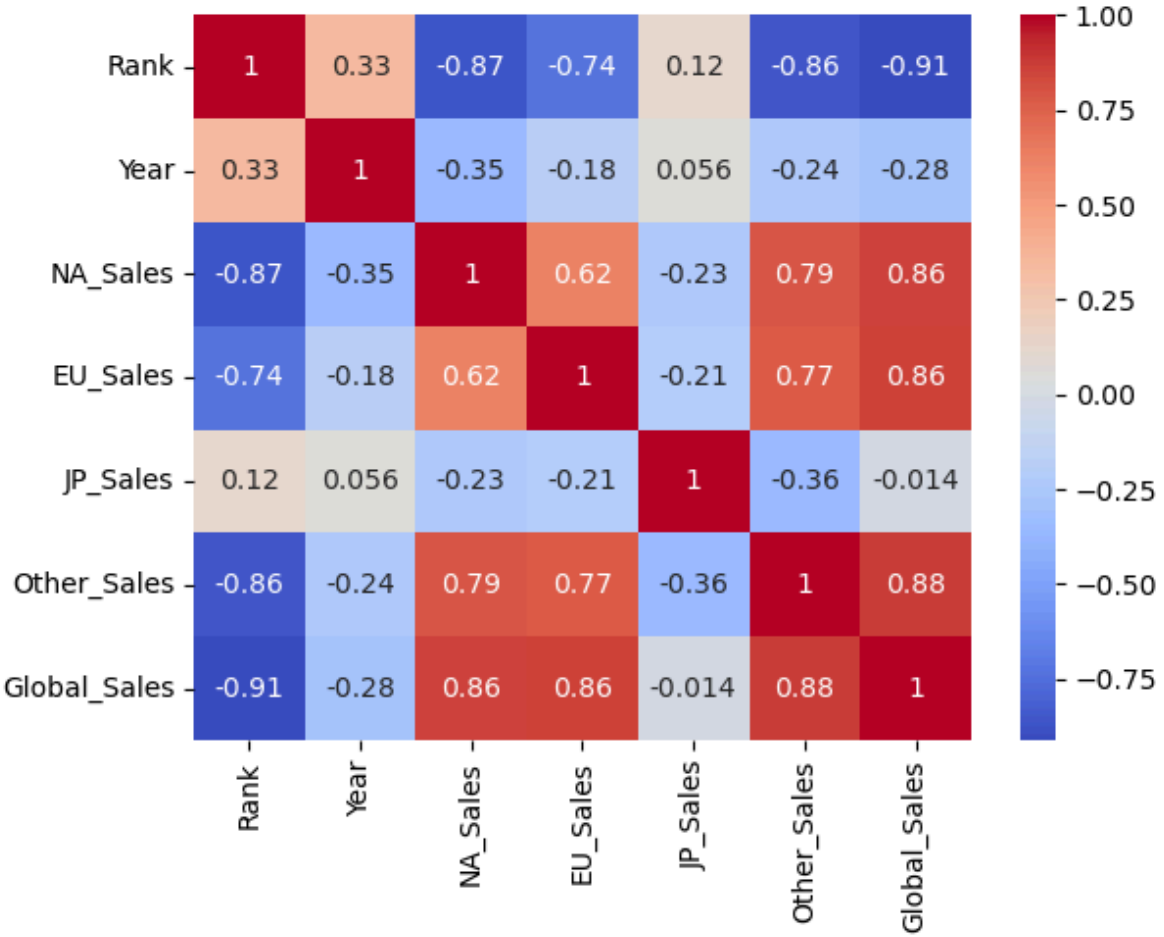


- **`annot=True`** is for writing the correlation coefficient inside each cell.
- You can change the colours of cells in heat map if you like.
  - There are a lot of options available!

In [15]:
```
print(plt.colormaps())
```

```
['magma', 'inferno', 'plasma', 'viridis', 'cividis', 'twilight', 'twilight_
shifted', 'turbo', 'Blues', 'BrBG', 'BuGn', 'BuPu', 'CMRmap', 'GnBu', 'Gree
ns', 'Greys', 'OrRd', 'Oranges', 'PRGn', 'PiYG', 'PuBu', 'PuBuGn', 'PuOr',
'PuRd', 'Purples', 'RdBu', 'RdGy', 'RdPu', 'RdYlBu', 'RdYlGn', 'Reds', 'Spe
ctral', 'Wistia', 'YlGn', 'YlGnBu', 'YlOrBr', 'YlOrRd', 'afmhot', 'autumn',
'binary', 'bone', 'brg', 'bwr', 'cool', 'coolwarm', 'copper', 'cubehelix',
'flag', 'gist_earth', 'gist_gray', 'gist_heat', 'gist_ncar', 'gist_rainbo
w', 'gist_stern', 'gist_yarg', 'gnuplot', 'gnuplot2', 'gray', 'hot', 'hsv',
'jet', 'nipy_spectral', 'ocean', 'pink', 'prism', 'rainbow', 'seismic', 'sp
ring', 'summer', 'terrain', 'winter', 'Accent', 'Dark2', 'Paired', 'Pastel
1', 'Pastel2', 'Set1', 'Set2', 'Set3', 'tab10', 'tab20', 'tab20b', 'tab20
c', 'magma_r', 'inferno_r', 'plasma_r', 'viridis_r', 'cividis_r', 'twilight
_r', 'twilight_shifted_r', 'turbo_r', 'Blues_r', 'BrBG_r', 'BuGn_r', 'BuPu_
r', 'CMRmap_r', 'GnBu_r', 'Greens_r', 'Greys_r', 'OrRd_r', 'Oranges_r', 'PR
Gn_r', 'PiYG_r', 'PuBu_r', 'PuBuGn_r', 'PuOr_r', 'PuRd_r', 'Purples_r', 'Rd
Bu_r', 'RdGy_r', 'RdPu_r', 'RdYlBu_r', 'RdYlGn_r', 'Reds_r', 'Spectral_r',
'Wistia_r', 'YlGn_r', 'YlGnBu_r', 'YlOrBr_r', 'YlOrRd_r', 'afmhot_r', 'autu
mn_r', 'binary_r', 'bone_r', 'brg_r', 'bwr_r', 'cool_r', 'coolwarm_r', 'cop
per_r', 'cubehelix_r', 'flag_r', 'gist_earth_r', 'gist_gray_r', 'gist_heat_
r', 'gist_ncar_r', 'gist_rainbow_r', 'gist_stern_r', 'gist_yarg_r', 'gnuplo
t_r', 'gnuplot2_r', 'gray_r', 'hot_r', 'hsv_r', 'jet_r', 'nipy_spectral_r',
'ocean_r', 'pink_r', 'prism_r', 'rainbow_r', 'seismic_r', 'spring_r', 'summ
er_r', 'terrain_r', 'winter_r', 'Accent_r', 'Dark2_r', 'Paired_r', 'Pastel1
_r', 'Pastel2_r', 'Set1_r', 'Set2_r', 'Set3_r', 'tab10_r', 'tab20_r', 'tab2
0b_r', 'tab20c_r', 'rocket', 'rocket_r', 'mako', 'mako_r', 'icefire', 'icef
ire_r', 'vlag', 'vlag_r', 'flare', 'flare_r', 'crest', 'crest_r']
```

In [16]:
```
sns.heatmap(num_df.corr(), cmap= "coolwarm", annot=True)
plt.show()
```



Quiz-1

Q. We are analyzing the results of the Olympics, and want to find the count of gold, silver, and bronze medals won by each country.

Which will be the best suited plot for this?

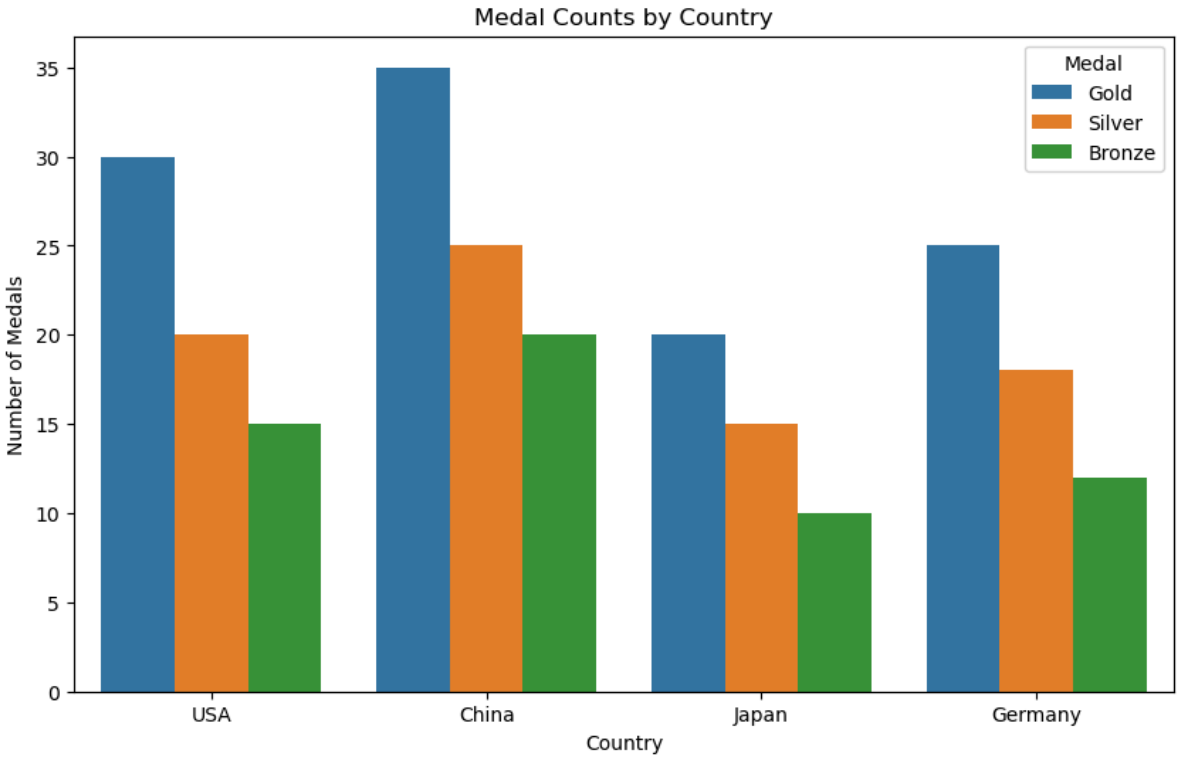
a. Dodged Bar Plot
b. Pie Chart
c. Scatter Plot
d. Line Plot

Answer: Dodged Bar Plot

Explanation:

Bar plots are effective for comparing the quantities of different categories, such as medal counts for different countries, making them ideal for this scenario. Each country can be represented by a bar, with the height of the bar corresponding to the total number of medals won (separately for gold, silver, and bronze). This allows for easy comparison between countries and their respective medal counts.

In [17]:
```
# Example DataFrame
data = {
    'Country': ['USA', 'China', 'Japan', 'Germany'],
    'Gold': [30, 35, 20, 25],
    'Silver': [20, 25, 15, 18],
    'Bronze': [15, 20, 10, 12]
}

df = pd.DataFrame(data)

# Melt the DataFrame to long format for easier plotting
df_melted = df.melt(id_vars='Country', var_name='Medal', value_name='Count')

# Plotting
plt.figure(figsize=(10, 6))
sns.barplot(x='Country', y='Count', hue='Medal', data=df_melted)
plt.xlabel('Country')
plt.ylabel('Number of Medals')
plt.title('Medal Counts by Country')
plt.show()
```



Quiz-2

Q. Suppose in a `2x3 subplot` (2 rows 3 columns), we want to create a plot to span across the first row.

What would be the right code for this?

```
a. plt.subplot(2,1,1)
b. plt.subplot(1,2,(1,1))
c. plt.subplot(2,2,(1,3))
d. plt.subplot(2,3,3)
```

Answer: plt.subplot(2,2,(1,3))

Explanation:

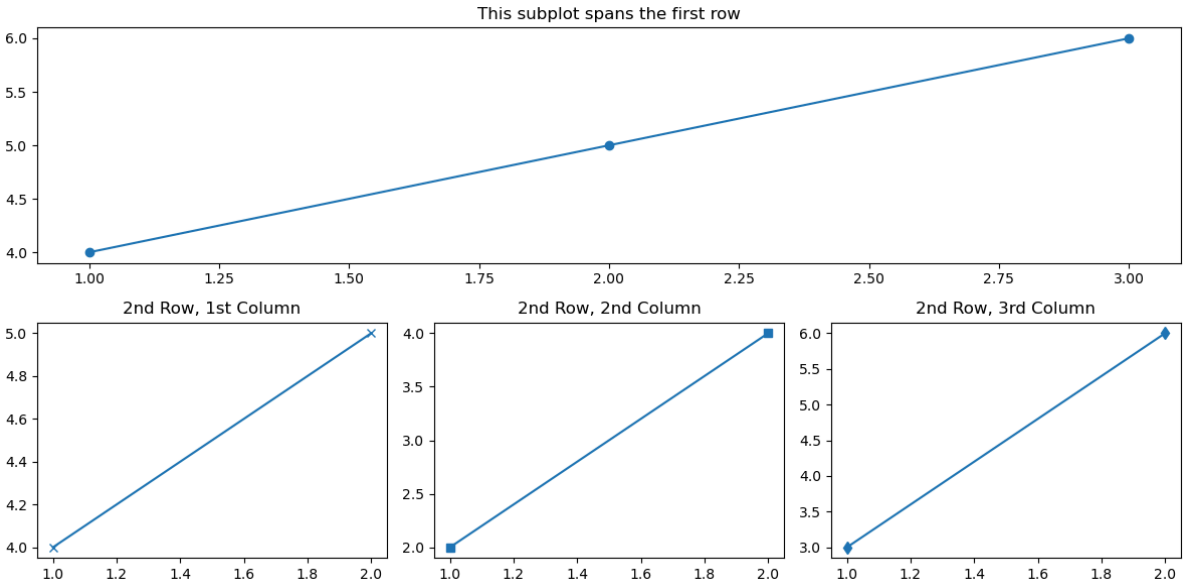
In `plt.subplot(nrows, ncols, index)`, the function creates subplots in a grid format with `nrows` rows and `ncols` columns, and `index` indicates the position of the subplot in the grid.

- `nrows=2` : Specifies that the subplot grid has 2 rows.
- `ncols=3` : Specifies that the grid has 3 columns.
- `index=(1,3)` : The index 1 refers to the first position in the grid, and 3 refers to the last position in the first row.

By specifying (1,3), you're telling Matplotlib to span the plot across the entire first row, i.e., over columns 1, 2, and 3.

In [18]:
```
plt.figure(figsize=(12, 6))

# Subplot spanning the entire first row (1st to 3rd index)
plt.subplot(2, 3, (1, 3))
```

```
plt.title('This subplot spans the first row')
plt.plot([1, 2, 3], [4, 5, 6], marker='o')

# Create individual subplots for the remaining cells
plt.subplot(2, 3, 4)
plt.title('2nd Row, 1st Column')
plt.plot([1, 2], [4, 5], marker='x')

plt.subplot(2, 3, 5)
plt.title('2nd Row, 2nd Column')
plt.plot([1, 2], [2, 4], marker='s')

plt.subplot(2, 3, 6)
plt.title('2nd Row, 3rd Column')
plt.plot([1, 2], [3, 6], marker='d')

# Adjust layout to avoid overlap
plt.tight_layout()

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



In [ ]: