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?

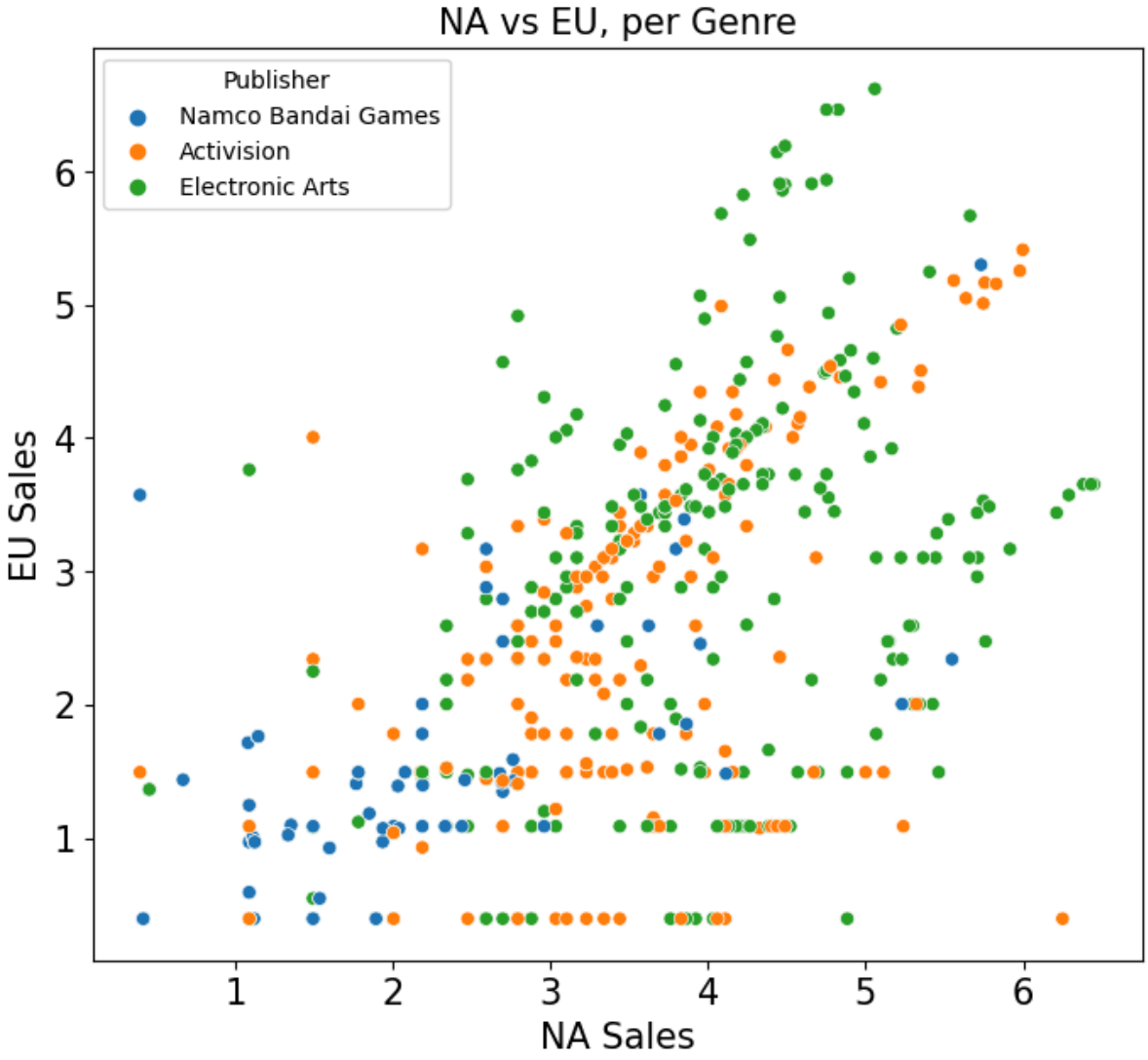
Here, we have two numerical and one categorical variable!

- 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?

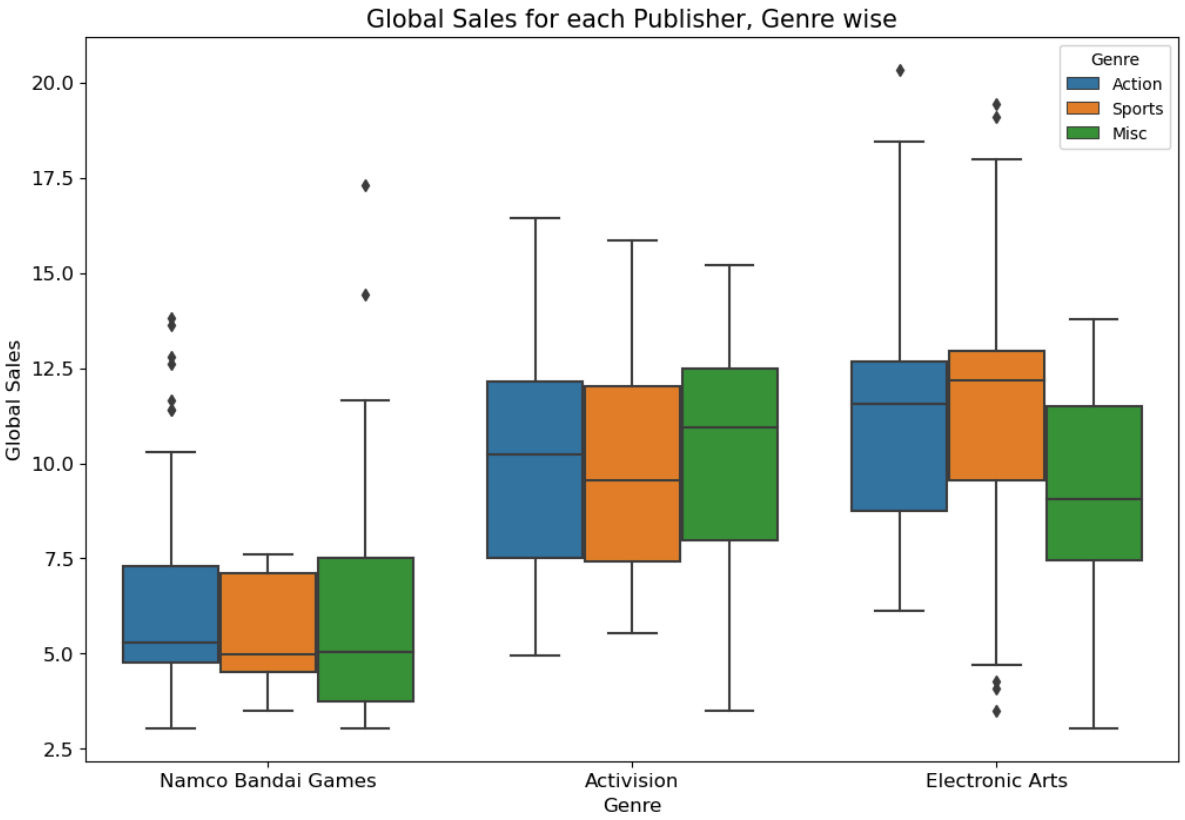
We have two categorical and one numerical data here!

- 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?

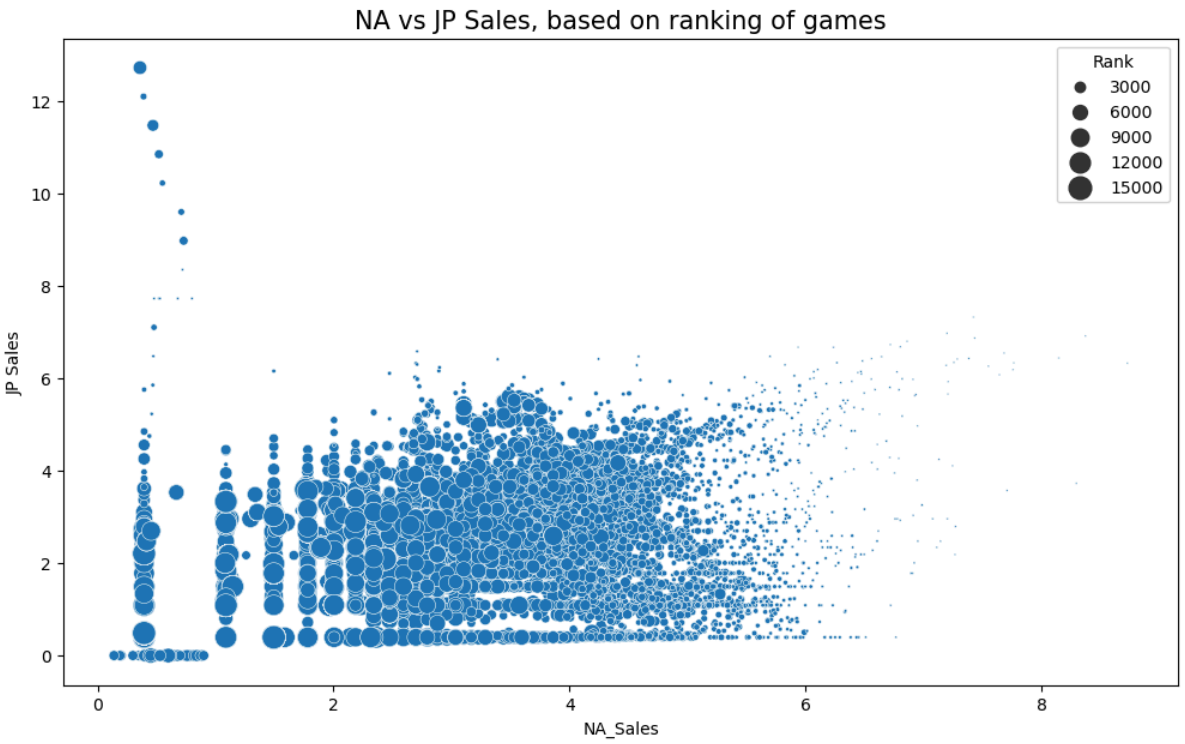
How does rank affect the correlation between NA and EU Sales?

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

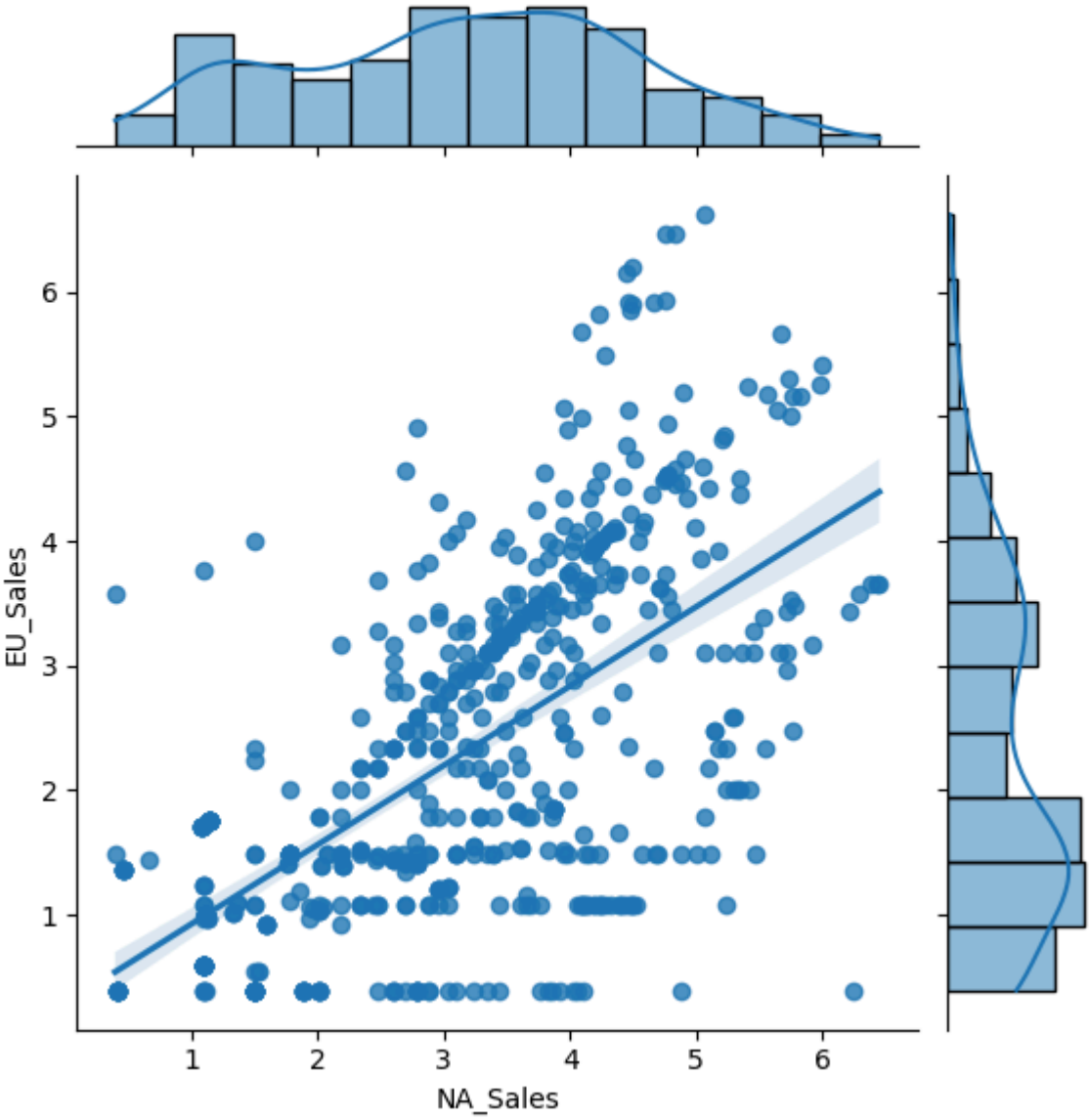
- `jointplot()` draws a plot between two variables.

- 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()
```



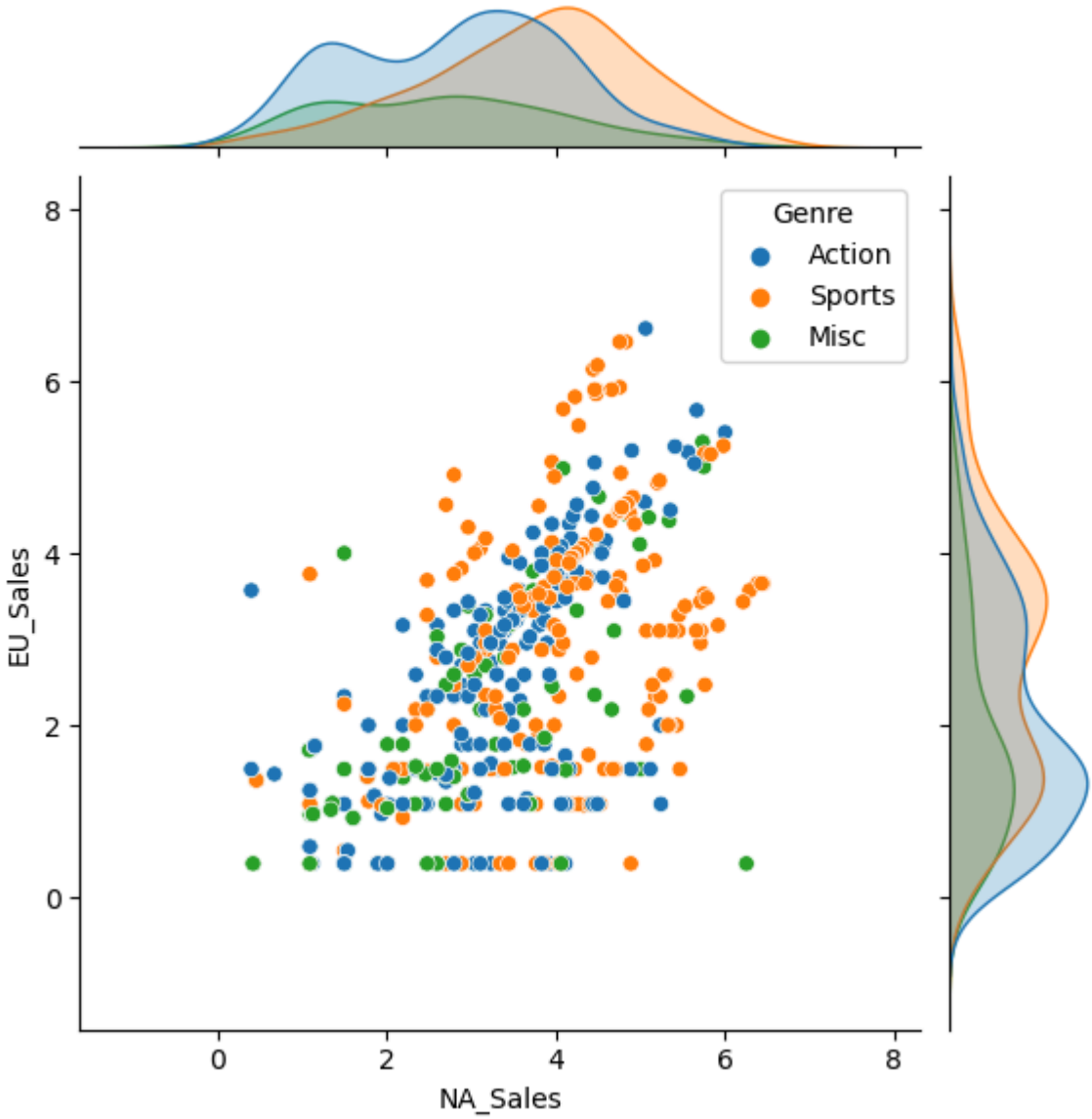
As we can see here,

- `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**.

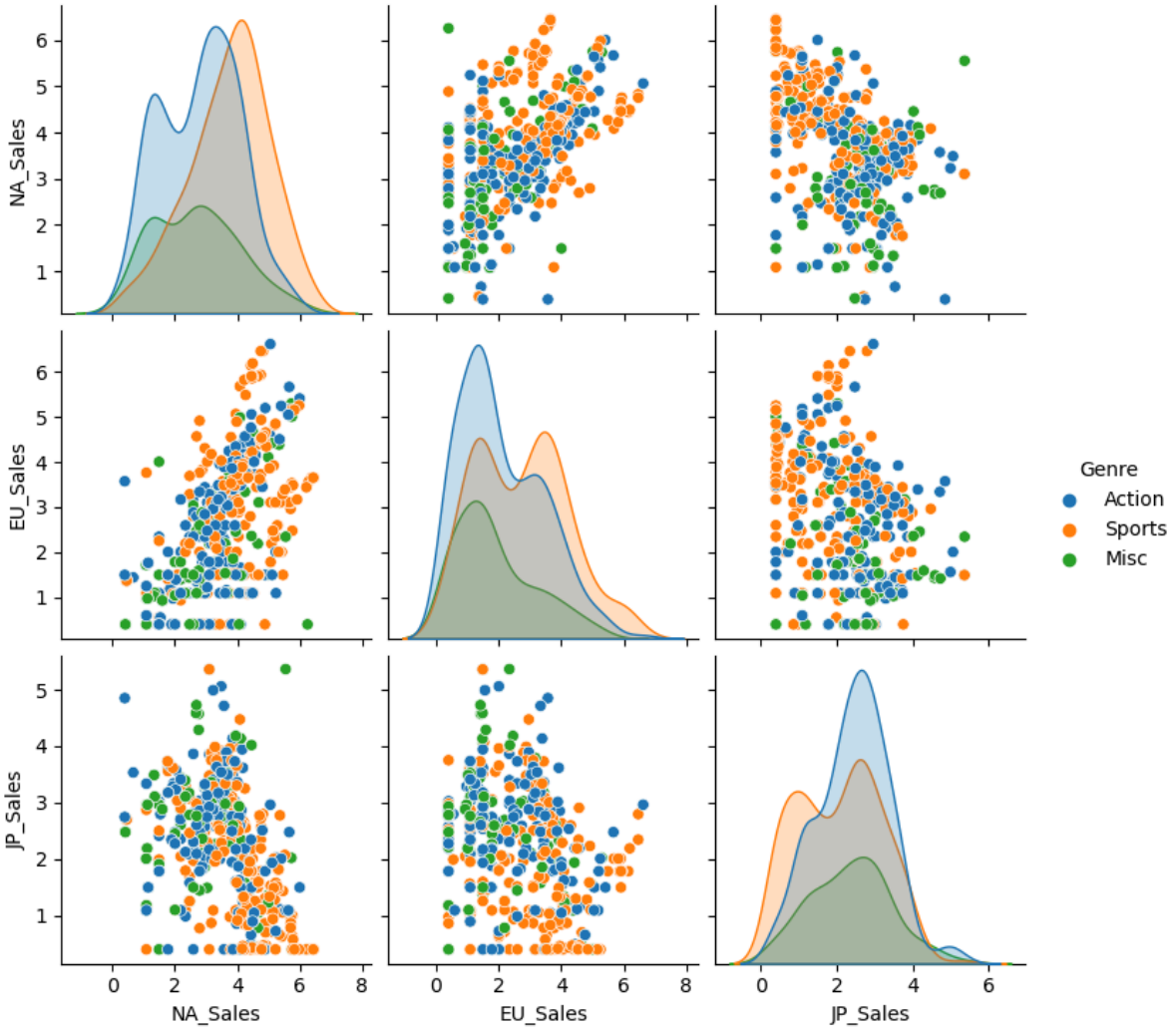
Diagonal plots are different from scatter plots because x and y axis have same attribute.

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.

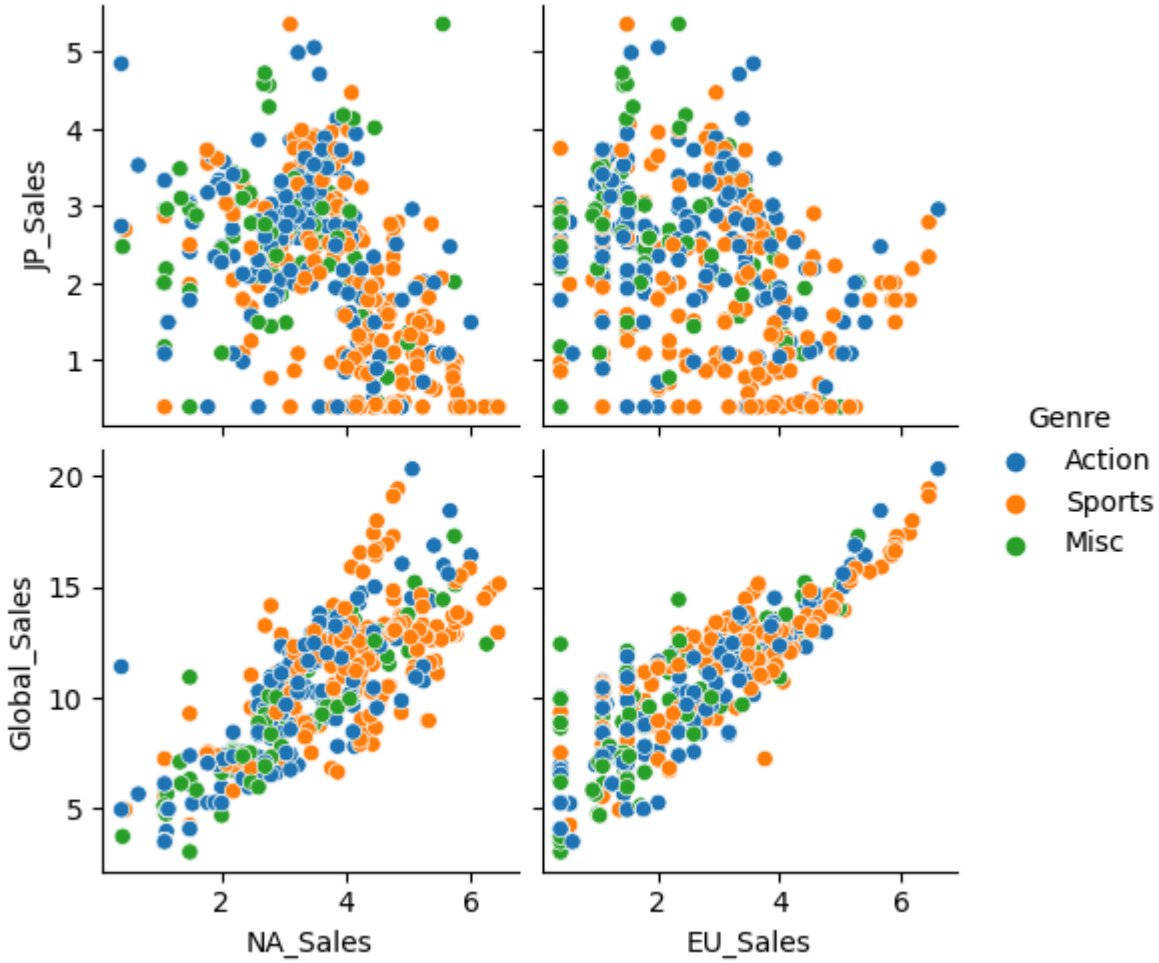
In [10]:
```
sns.pairplot(data=top3_data, vars=['NA_Sales', 'EU_Sales', 'JP_Sales'], hue=
```

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]:

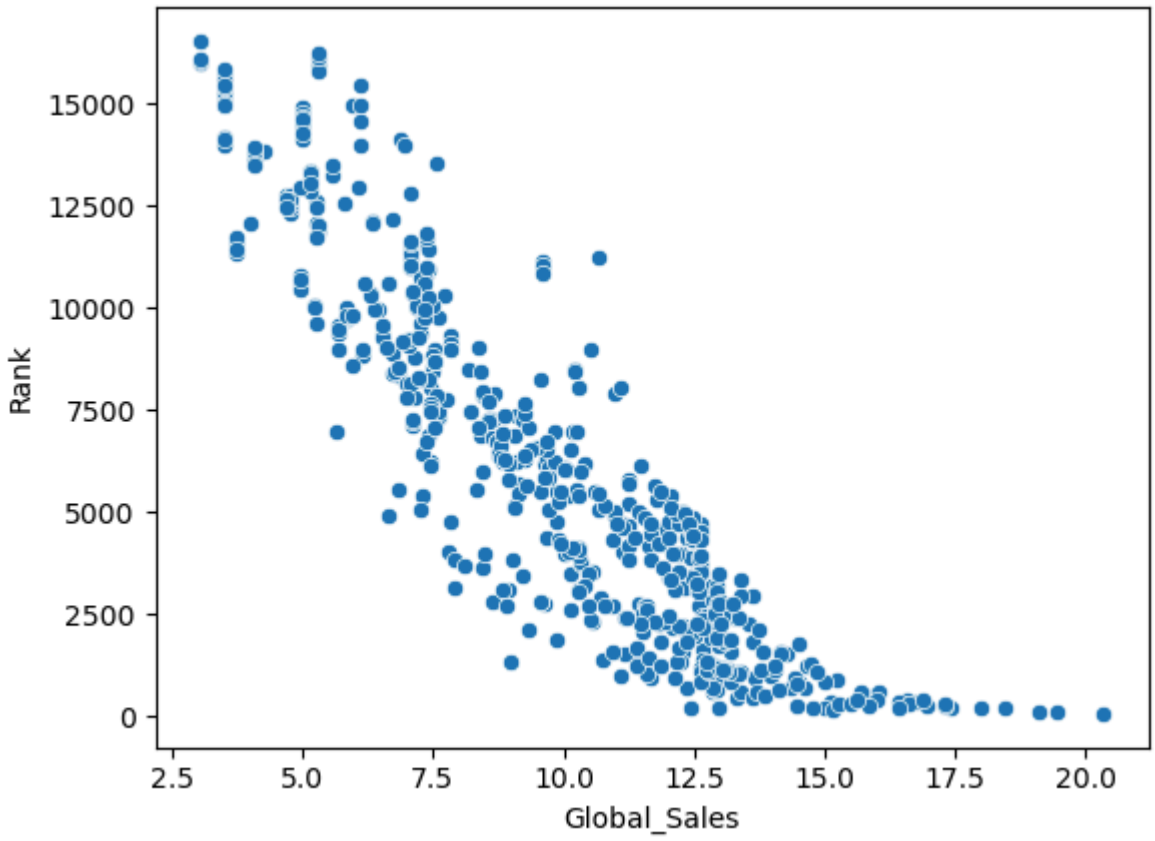
	Rank	Year	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_S
Rank	1.000000	0.328705	-0.873726	-0.735711	0.115459	-0.857567	-0.91
Year	0.328705	1.000000	-0.354256	-0.178026	0.055864	-0.239876	-0.28
NA_Sales	-0.873726	-0.354256	1.000000	0.617483	-0.233315	0.794353	0.856
EU_Sales	-0.735711	-0.178026	0.617483	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()
```



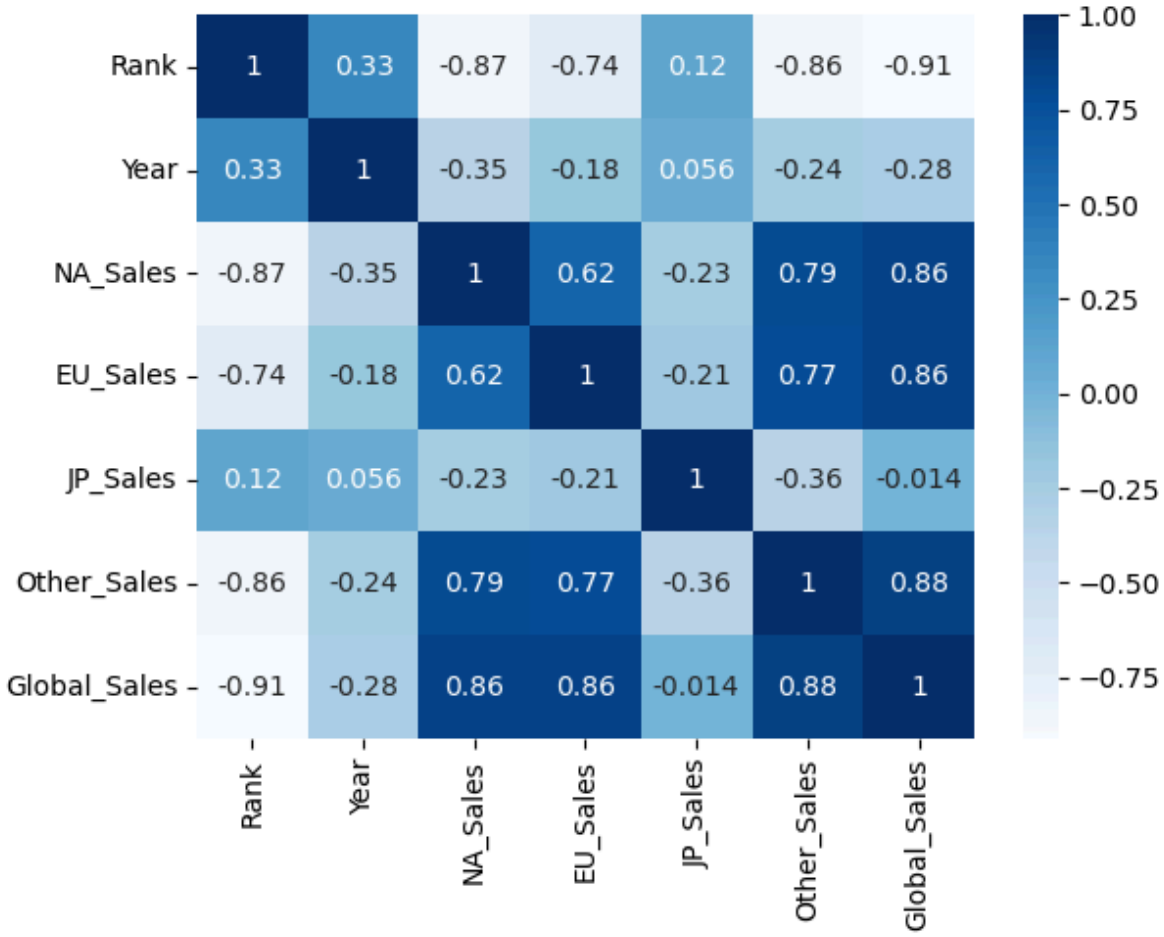
Now let's look at a way to visualize correlation among variables.

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()
```

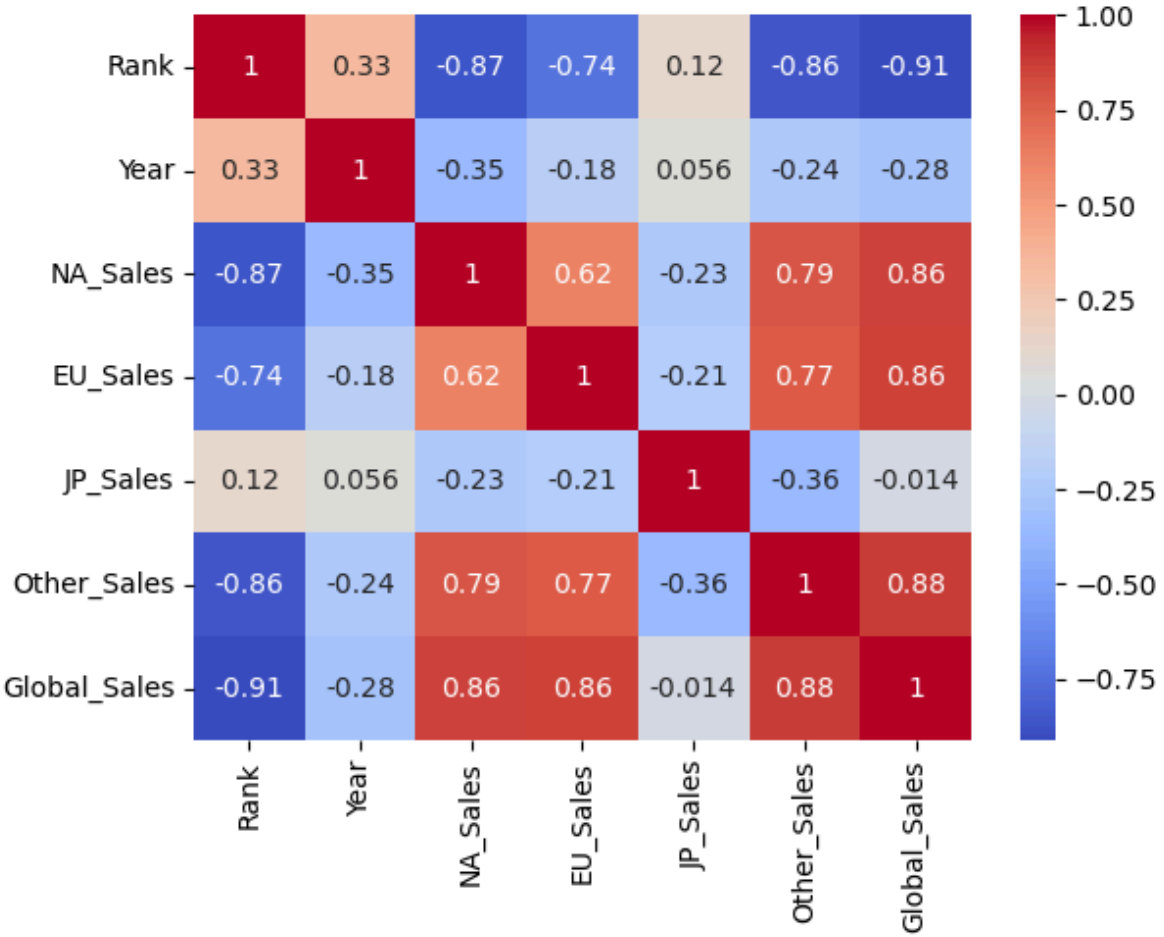


- **`annot=True`** is for writing the correlation coefficient inside each cell.
- You can change the colours of cells in heat map if you like.
  - There are a lot of options available!

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?

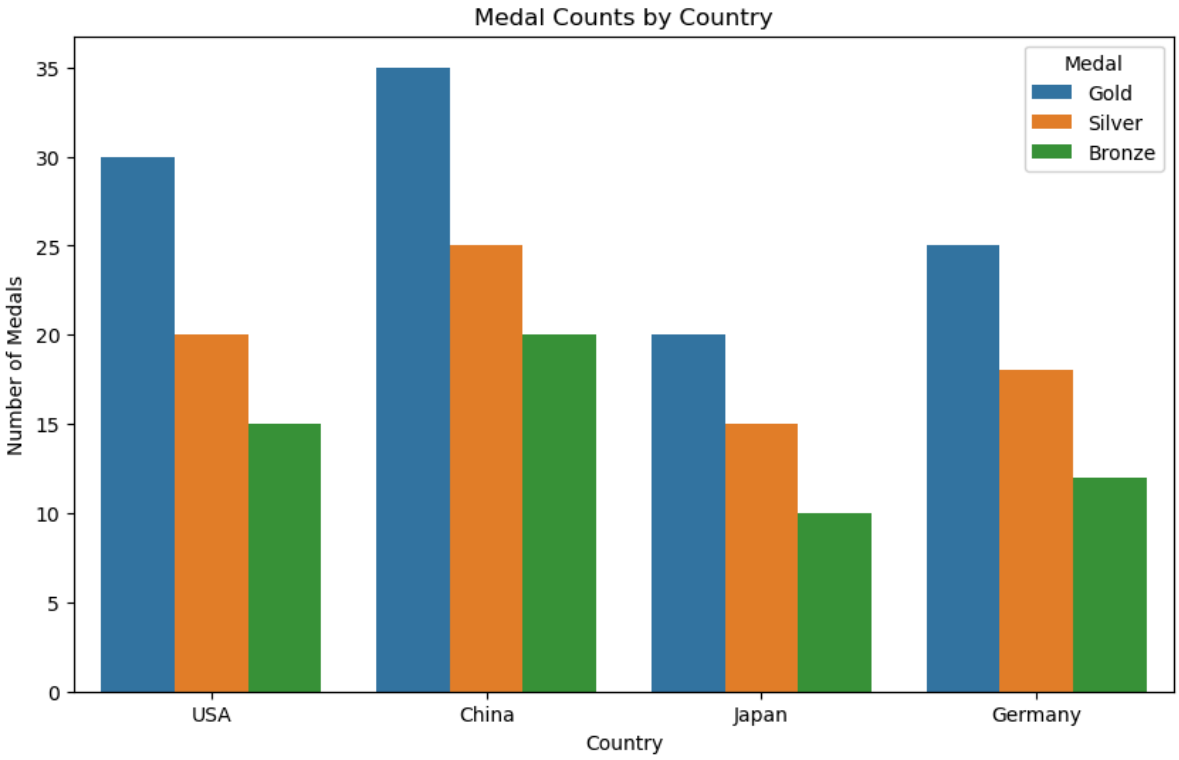
a. Dodged Bar Plot
b. Pie Chart
c. Scatter Plot
d. Line Plot

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:

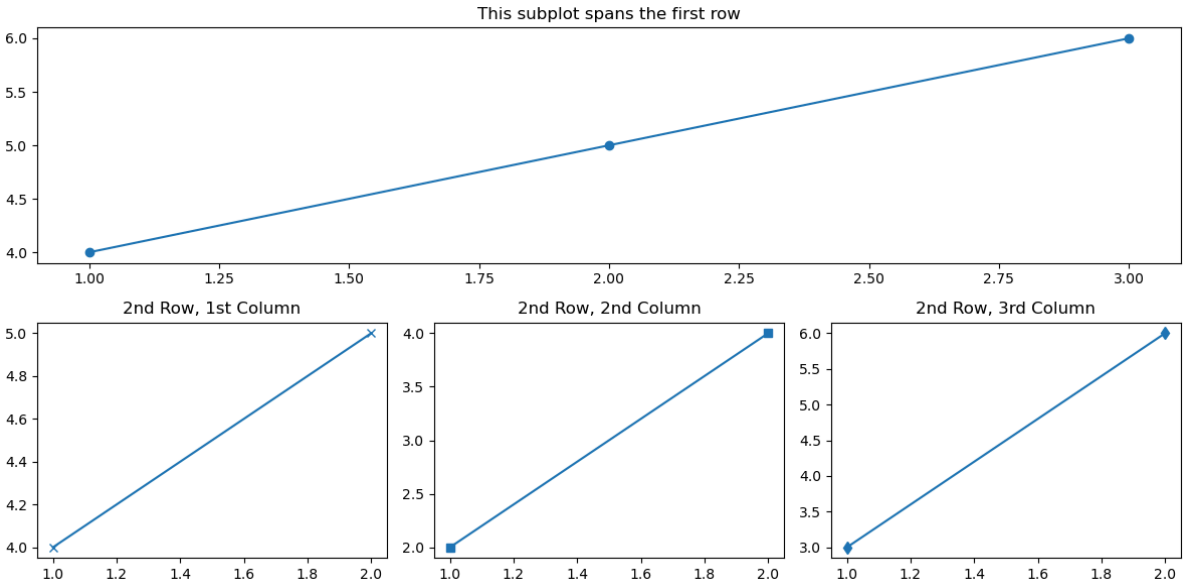
In `plt.subplot(nrows, ncols, index)`, the function creates subplots in a grid format with `nrows` rows and `ncols` columns, and `index` indicates the position of the subplot in the grid.

- `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 [ ]: