Loading needed libraries

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

In [2]:
```
video_games_df = pd.read_csv(r'C:\Users\User\PycharmProjects\PythonProject2\Othe
video_games_df
```

Out[2]:

	Rank	Name	Platform	Year	Month	Genre	Publisher	Country	
0	1	Wii Sports	Wii	2010	Jan	Sports	Nintendo	United States	Fa
1	2	Super Mario Bros.	NES	2010	Feb	Platform	Nintendo	United States	Edr
2	3	Mario Kart Wii	Wii	2010	Mar	Racing	Nintendo	United States	Lou
3	4	Wii Sports Resort	Wii	2010	Apr	Sports	Nintendo	United States	
4	5	Pokemon Red/Pokemon Blue	GB	2010	May	Role-Playing	Nintendo	United States	Na
...	...	...	...	...	...	...	...	...	
5904	5890	Crazy Taxi: Catch a Ride	GBA	2017	Sep	Racing	THQ	Australia	S
5905	5891	MySims Party	DS	2017	Oct	Simulation	Electronic Arts	Australia	S
5906	5892	Harry Potter and the Order of the Phoenix	X360	2017	Nov	Action	Electronic Arts	Australia	S
5907	5893	Skylanders: SuperChargers	PS4	2017	Dec	Action	Activision	Australia	S
5908	5894	Macross Digital Mission VF-X	PS	2017	Jan	Simulation	Namco Bandai Games	Australia	S

5909 rows × 15 columns

Content Dataset checks

In [3]:
```
#Number of rows and columns
total_rows = video_games_df.shape[0]
print(f'Total number of rows in the dataset are: {total_rows}')
total_columns = video_games_df.shape[1]
print(f'Total number of columns in the dataset are: {total_columns}')
print(' ')
```

```
print('--------------------')
#Alternative way for number of rows and columns
shape = video_games_df.shape
print(shape)
print(' ')
#Column Names
columns = video_games_df.columns
print(columns)
print(' ')
print('--------------------')
#First 5 rows
first_5_rows = video_games_df.head()
print(first_5_rows)
print(' ')
#Last 5 rows
last_five_rows = video_games_df.tail()
print(last_five_rows)
print(' ')
print('--------------------')
#Key details of the DataFrame
df_info = video_games_df.info(show_counts=True)
print(df_info)
print(' ')
#Description of the DataFrame
df_describe = video_games_df.describe(include='all').round()
print(df_describe)
print(' ')
```

```
Total number of rows in the dataset are: 5909
Total number of columns in the dataset are: 15

--------------------
(5909, 15)

Index(['Rank', 'Name', 'Platform', 'Year', 'Month', 'Genre', 'Publisher',
       'Country', 'City', 'State', 'Region', 'NA_Sales', 'Global_Sales',
       'NA_Profit', 'Global_Profit'],
      dtype='object')

--------------------
   Rank                      Name Platform  Year Month         Genre  \
0     1                Wii Sports      Wii  2010   Jan        Sports
1     2         Super Mario Bros.      NES  2010   Feb      Platform
2     3            Mario Kart Wii      Wii  2010   Mar        Racing
3     4         Wii Sports Resort      Wii  2010   Apr        Sports
4     5  Pokemon Red/Pokemon Blue       GB  2010   May  Role-Playing

  Publisher        Country        City       State   Region NA_Sales  \
0  Nintendo  United States   Fairfield  California     West   $41.49
1  Nintendo  United States     Edmonds  Washington     West   $29.08
2  Nintendo  United States  Louisville    Kentucky    South   $15.85
3  Nintendo  United States  Round Rock       Texas  Central   $15.75
4  Nintendo  United States   Nashville   Tennessee    South   $11.27

   Global_Sales  NA_Profit  Global_Profit
0         82.74     12.447         24.822
1         40.24      8.724         12.072
2         35.82      4.755         10.746
3         33.00      4.725          9.900
4         31.37      3.381          9.411

      Rank                                      Name Platform  Year Month  \
5904  5890                  Crazy Taxi: Catch a Ride      GBA  2017   Sep
5905  5891                               MySims Party       DS  2017   Oct
5906  5892  Harry Potter and the Order of the Phoenix     X360  2017   Nov
5907  5893                 Skylanders: SuperChargers      PS4  2017   Dec
5908  5894              Macross Digital Mission VF-X       PS  2017   Jan

           Genre           Publisher    Country    City            State  \
5904      Racing                 THQ  Australia  Sydney  New South Wales
5905  Simulation     Electronic Arts  Australia  Sydney  New South Wales
5906      Action     Electronic Arts  Australia  Sydney  New South Wales
5907      Action          Activision  Australia  Sydney  New South Wales
5908  Simulation  Namco Bandai Games  Australia  Sydney  New South Wales

     Region NA_Sales  Global_Sales  NA_Profit  Global_Profit
5904   West     0.21           0.3      0.063           0.09
5905   West     0.15           0.3      0.045           0.09
5906   West     0.24           0.3      0.072           0.09
5907   West     0.17           0.3      0.051           0.09
5908   West        0           0.3      0.000           0.09

--------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5909 entries, 0 to 5908
Data columns (total 15 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
```

```
 0   Rank           5909 non-null   int64
 1   Name           5909 non-null   object
 2   Platform       5909 non-null   object
 3   Year           5909 non-null   int64
 4   Month          5909 non-null   object
 5   Genre          5909 non-null   object
 6   Publisher      5897 non-null   object
 7   Country        5909 non-null   object
 8   City           5909 non-null   object
 9   State          5909 non-null   object
 10  Region         5882 non-null   object
 11  NA_Sales       5909 non-null   object
 12  Global_Sales   5909 non-null   float64
 13  NA_Profit      5909 non-null   float64
 14  Global_Profit  5909 non-null   float64
dtypes: float64(3), int64(2), object(10)
memory usage: 692.6+ KB
None
```

	Rank	Name	Platform	Year	Month	Genre
count	5909.0	5909	5909	5909.0	5909	5909
unique	NaN	4139	26	NaN	14	12
top	NaN	LEGO Marvel Super Heroes	PS2	NaN	Jan	Action
freq	NaN	8	934	NaN	495	1274
mean	2944.0	NaN	NaN	2013.0	NaN	NaN
std	1705.0	NaN	NaN	2.0	NaN	NaN
min	1.0	NaN	NaN	2010.0	NaN	NaN
25%	1469.0	NaN	NaN	2011.0	NaN	NaN
50%	2943.0	NaN	NaN	2013.0	NaN	NaN
75%	4420.0	NaN	NaN	2015.0	NaN	NaN
max	5894.0	NaN	NaN	2017.0	NaN	NaN

	Publisher	Country	City	State	Region	NA_Sales
count	5897	5909	5909	5909	5882	5909
unique	219	3	391	56	6	420
top	Electronic Arts	Australia	Sydney	New South Wales	West	0
freq	865	3111	2956	2980	3841	503
mean	NaN	NaN	NaN	NaN	NaN	NaN
std	NaN	NaN	NaN	NaN	NaN	NaN
min	NaN	NaN	NaN	NaN	NaN	NaN
25%	NaN	NaN	NaN	NaN	NaN	NaN
50%	NaN	NaN	NaN	NaN	NaN	NaN
75%	NaN	NaN	NaN	NaN	NaN	NaN
max	NaN	NaN	NaN	NaN	NaN	NaN

	Global_Sales	NA_Profit	Global_Profit
count	5909.0	5909.0	5909.0
unique	NaN	NaN	NaN
top	NaN	NaN	NaN
freq	NaN	NaN	NaN
mean	1.0	0.0	0.0
std	3.0	0.0	1.0
min	0.0	0.0	0.0
25%	0.0	0.0	0.0
50%	1.0	0.0	0.0
75%	1.0	0.0	0.0
max	83.0	12.0	25.0

Duplicate rows check & NaN rows check

In [4]:
```
#Duplicate rows check
print(video_games_df.duplicated())
duplicated_rows_count = video_games_df.duplicated().sum()
print(f'Total number of duplicated rows in the dataset are: {duplicated_rows_cou
print(' ')
#NaN values check
df_NaN = video_games_df.isna().sum()
print(' ')
print('The analysis for NaN values is:')
print(df_NaN)
```

```
0       False
1       False
2       False
3       False
4       False
        ...
5904    False
5905    False
5906    False
5907    False
5908    False
Length: 5909, dtype: bool
Total number of duplicated rows in the dataset are: 16


The analysis for NaN values is:
Rank              0
Name              0
Platform          0
Year              0
Month             0
Genre             0
Publisher        12
Country           0
City              0
State             0
Region           27
NA_Sales          0
Global_Sales      0
NA_Profit         0
Global_Profit     0
dtype: int64
```

Dropping duplicates rows

In [5]:
```
#Dropping duplicate rows
video_games_df=video_games_df.drop_duplicates()
#Fixing index
video_games_df.reset_index()

#Duplicate Values check again to find 0 for confirming everything is right
print(video_games_df.duplicated())
duplicated_rows_count = video_games_df.duplicated().sum()
print(f'Total number of duplicated rows in the dataset are: {duplicated_rows_cou
print(' ')
```

```
0       False
1       False
2       False
3       False
4       False
        ...
5904    False
5905    False
5906    False
5907    False
5908    False
Length: 5893, dtype: bool
Total number of duplicated rows in the dataset are: 0
```

Handling NaN values

In [6]:
```
#From total 5909 rows we have NaN values per info() in Region column (5909-5882=
#Total number of NaN in both columns is insignificant in terms of dataSet theref
# 'Not available' content

#Filling with re-assigning the old variable name of the DataFrame
video_games_df = video_games_df.fillna({'Region':'Not available'})
video_games_df = video_games_df.fillna({'Publisher':'Not available'})

#NaN values check to confirm our fill is done right
df_NaN = video_games_df.isna().sum()
print(' ')
print('The analysis for NaN values is:')
print(df_NaN)
```

```
The analysis for NaN values is:
Rank             0
Name             0
Platform         0
Year             0
Month            0
Genre            0
Publisher        0
Country          0
City             0
State            0
Region           0
NA_Sales         0
Global_Sales     0
NA_Profit        0
Global_Profit    0
dtype: int64
```

Cleaning NA_Sales column

In [7]:
```
#Using .str.replace() to remove '$' and then .astype(int) to convert
video_games_df['NA_Sales'] = video_games_df['NA_Sales'].str.replace("$", "").ast
```

Converting NA_Sales column values to numeric

In [8]:
```
# Converting NA_Sales column values to numeric and in case it's not possible out
video_games_df['NA_Sales'] = pd.to_numeric ( video_games_df['NA_Sales'] , errors
```

```
print(' ')

#Checking data type of column NA_Sales
print(video_games_df['NA_Sales'].dtype)
print(' ')

#NaN values check
df_NaN = video_games_df.isna().sum()
print(' ')
print('The analysis for NaN values is:')
print(df_NaN)
```

```
float64


The analysis for NaN values is:
Rank             0
Name             0
Platform         0
Year             0
Month            0
Genre            0
Publisher        0
Country          0
City             0
State            0
Region           0
NA_Sales         0
Global_Sales     0
NA_Profit        0
Global_Profit    0
dtype: int64
```

Making last 10 row values of column NA_Sales NaN to after fill with average of column

In [9]:
```
#Making last 10 row values of column NA_Sales NaN to after fill with average of

# Getting the index labels for the last 10 rows
last_10_indices = video_games_df.iloc[-10:].index
# Using .loc to select 10 last rows and the 'NA_Sales' column, then set them to
video_games_df.loc[last_10_indices, 'NA_Sales'] = np.nan
print(f"Set the last 10 'NA_Sales' values to NaN.")
print(f"NaNs in NA_Sales (After):  {video_games_df['NA_Sales'].isna().sum()}")
print(' ')

#Calculating Average of column NA_Sales
average_NA_Sales=int(video_games_df['NA_Sales'].mean())
print(average_NA_Sales)
print(' ')

#Filling with average_NA_Sales the previous NaN made last 10 values of column NA
video_games_df['NA_Sales'] = video_games_df['NA_Sales'] .fillna(average_NA_Sales
video_games_df
```

```
Set the last 10 'NA_Sales' values to NaN.
NaNs in NA_Sales (After):  10

0
```

Out[9]:

	Rank	Name	Platform	Year	Month	Genre	Publisher	Country	
0	1	Wii Sports	Wii	2010	Jan	Sports	Nintendo	United States	Fa
1	2	Super Mario Bros.	NES	2010	Feb	Platform	Nintendo	United States	Edr
2	3	Mario Kart Wii	Wii	2010	Mar	Racing	Nintendo	United States	Lou
3	4	Wii Sports Resort	Wii	2010	Apr	Sports	Nintendo	United States	
4	5	Pokemon Red/Pokemon Blue	GB	2010	May	Role-Playing	Nintendo	United States	Na
...	...	...	...	...	...	...	...	...	
5904	5890	Crazy Taxi: Catch a Ride	GBA	2017	Sep	Racing	THQ	Australia	S
5905	5891	MySims Party	DS	2017	Oct	Simulation	Electronic Arts	Australia	S
5906	5892	Harry Potter and the Order of the Phoenix	X360	2017	Nov	Action	Electronic Arts	Australia	S
5907	5893	Skylanders: SuperChargers	PS4	2017	Dec	Action	Activision	Australia	S
5908	5894	Macross Digital Mission VF-X	PS	2017	Jan	Simulation	Namco Bandai Games	Australia	S

5893 rows × 15 columns

Standardizing categorical values in column Country

In [10]:
```
video_games_df['Country'] = video_games_df['Country'].replace({'USA':'United Sta
video_games_df['Country'] = video_games_df['Country'].replace({'united states':'
print(' ')

#Checking category values of column Country
print(video_games_df['Country'].unique())
print(' ')
#Checking number of values per category for column Country
print(video_games_df['Country'].value_counts())
```

```
['United States' 'Australia']

Country
Australia        3108
United States    2785
Name: count, dtype: int64
```

Renaming Columns

In [11]:
```
#Renaming column NA_Sales to
video_games_df = video_games_df.rename(columns={'NA_Sales':'National Sales','Glo
'Global_Profit':'Global Profit'})

#Column Names Check
columns = video_games_df.columns
print(columns)
```

```
Index(['Rank', 'Name', 'Platform', 'Year', 'Month', 'Genre', 'Publisher',
       'Country', 'City', 'State', 'Region', 'National Sales', 'Global Sales',
       'National Profit', 'Global Profit'],
      dtype='object')
```

Handling Outliers in National Sales (prior NA_Sales) Column

In [12]:
```
sales_cap = video_games_df ['National Sales'].quantile(0.95)
print(f"The 95% of values in column National Sales have values less than:{sales_

print("For values of National Sales that have values > sales_cap, replacing the
video_games_df ['National Sales'] = np.where(video_games_df['National Sales']>sa
video_games_df
```

```
The 95% of values in column National Sales have values less than:2.07
For values of National Sales that have values > sales_cap, replacing the number w
ith sales_cap,else give same number
```

Out[12]:

	Rank	Name	Platform	Year	Month	Genre	Publisher	Country	
0	1	Wii Sports	Wii	2010	Jan	Sports	Nintendo	United States	Fa
1	2	Super Mario Bros.	NES	2010	Feb	Platform	Nintendo	United States	Edr
2	3	Mario Kart Wii	Wii	2010	Mar	Racing	Nintendo	United States	Lou
3	4	Wii Sports Resort	Wii	2010	Apr	Sports	Nintendo	United States	
4	5	Pokemon Red/Pokemon Blue	GB	2010	May	Role-Playing	Nintendo	United States	Na
...	...	...	...	...	...	...	...	...	
5904	5890	Crazy Taxi: Catch a Ride	GBA	2017	Sep	Racing	THQ	Australia	S
5905	5891	MySims Party	DS	2017	Oct	Simulation	Electronic Arts	Australia	S
5906	5892	Harry Potter and the Order of the Phoenix	X360	2017	Nov	Action	Electronic Arts	Australia	S
5907	5893	Skylanders: SuperChargers	PS4	2017	Dec	Action	Activision	Australia	S
5908	5894	Macross Digital Mission VF-X	PS	2017	Jan	Simulation	Namco Bandai Games	Australia	S

5893 rows × 15 columns

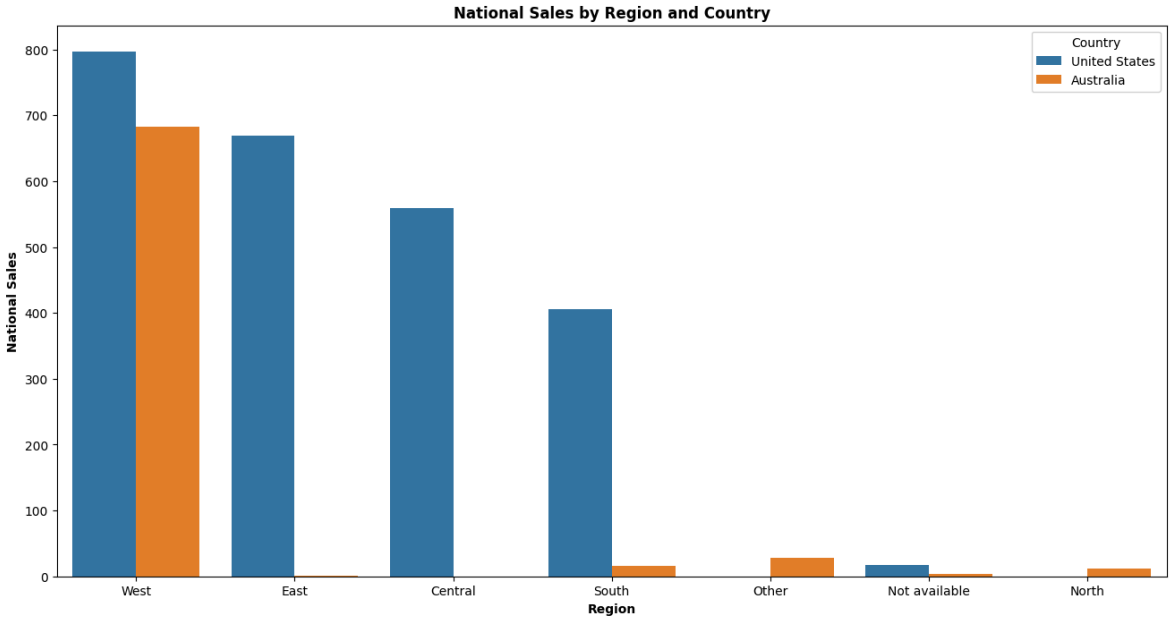
Visualizations

In [13]:
```
# Data preparation for Visualization.

National_Sales = video_games_df.groupby(['Region','Country'])['National Sales'].
print(National_Sales)
print(" ")

#Sorting National_Sales in DESC order
National_Sales = National_Sales.sort_values(by='National Sales', ascending=False
print(National_Sales)
```

	Region	Country	National Sales
0	Central	United States	558.67
1	East	Australia	1.35
2	East	United States	668.92
3	North	Australia	11.68
4	Not available	Australia	3.56
5	Not available	United States	17.69
6	Other	Australia	28.21
7	South	Australia	16.30
8	South	United States	405.99
9	West	Australia	682.02
10	West	United States	796.65

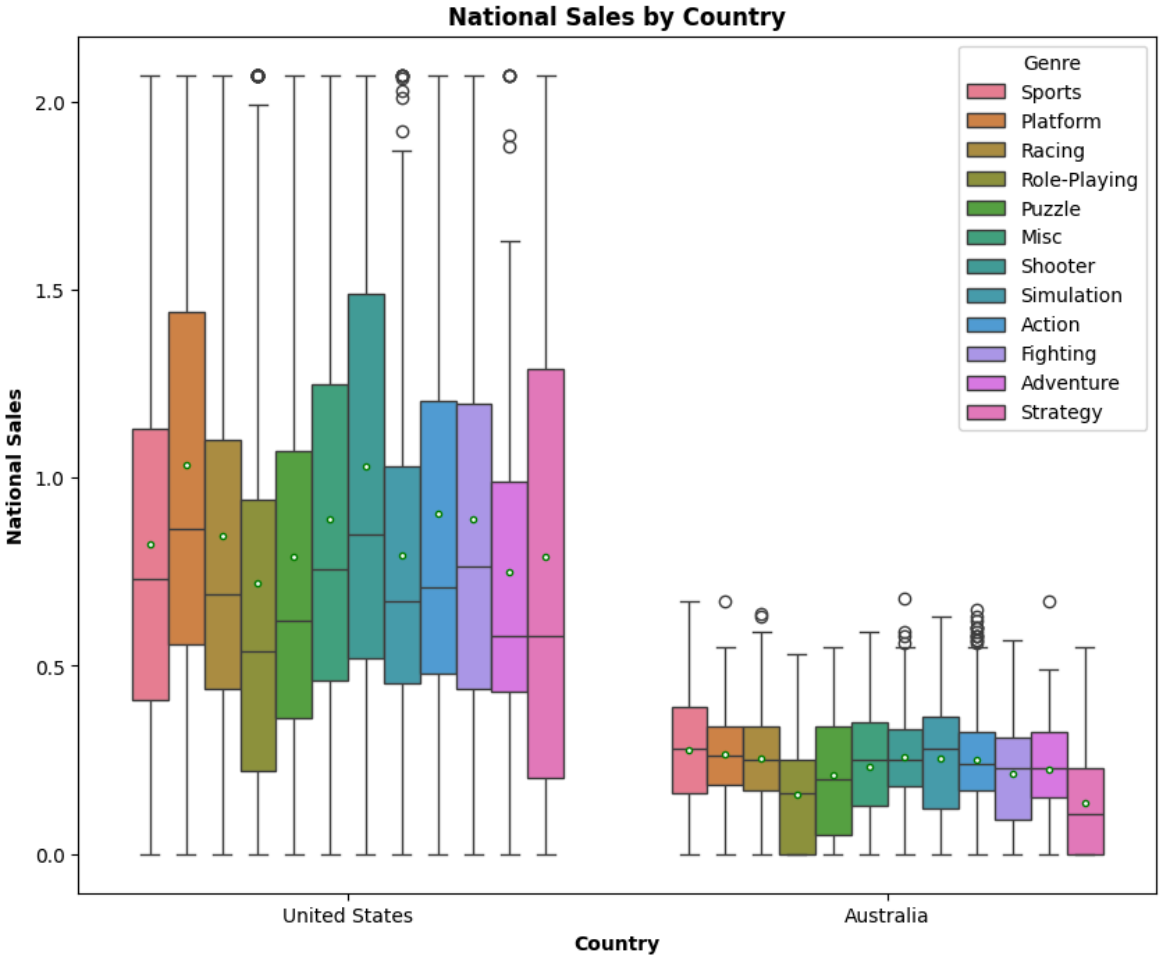
	Region	Country	National Sales
10	West	United States	796.65
9	West	Australia	682.02
2	East	United States	668.92
0	Central	United States	558.67
8	South	United States	405.99
6	Other	Australia	28.21
5	Not available	United States	17.69
7	South	Australia	16.30
3	North	Australia	11.68
4	Not available	Australia	3.56
1	East	Australia	1.35

In [14]:
```
# Plotting a bar chart for National Sales by Region and Country

plt.figure(figsize=(16,8))
sns.barplot(data=National_Sales,x='Region',y='National Sales',hue='Country')
plt.title('National Sales by Region and Country',fontweight='bold')
plt.xlabel('Region',fontweight='bold')
plt.ylabel('National Sales',fontweight='bold')
plt.show()
```



In [15]:
```
# Plotting a box plot for National Sales by Region and Country

plt.figure(figsize=(10,8))
sns.boxplot(x='Country',y='National Sales',data=video_games_df,showmeans=True,hu
```

```
plt.title('National Sales by Country',fontweight='bold')
plt.xlabel('Country',fontweight='bold')
plt.ylabel('National Sales',fontweight='bold')
plt.show()
```

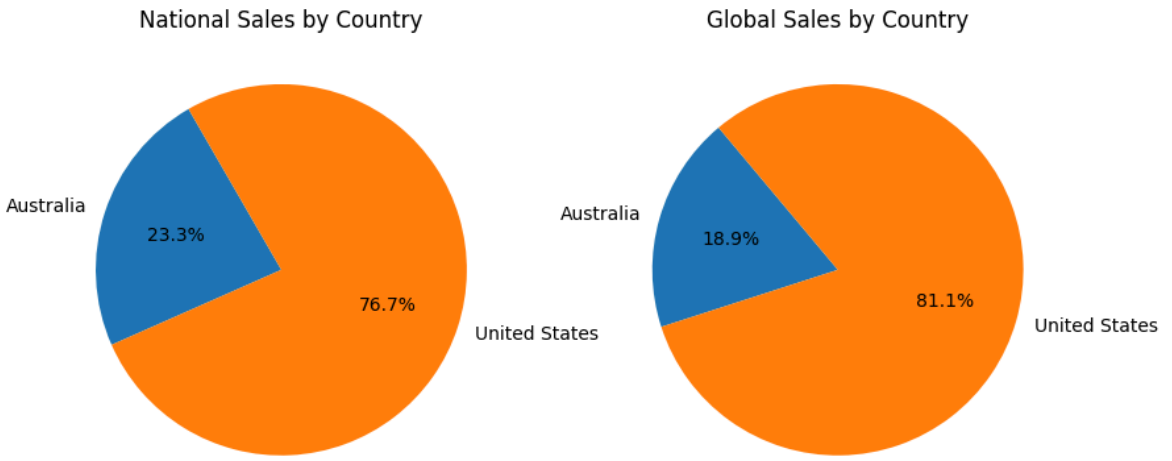


In [16]:
```
# Data preparation for Visualization.

Sales = video_games_df.groupby (['Country']) [['National Sales','Global Sales']]
print(Sales)
print(' ')

#Renaming columns in new DataFrame Sales
Country = Sales['Country']
National__Sales = Sales ['National Sales']
Global_Sales = Sales ['Global Sales']
print(Sales)
```

```
         Country  National Sales  Global Sales
0      Australia          743.12       1468.25
1  United States         2447.92       6315.68

         Country  National Sales  Global Sales
0      Australia          743.12       1468.25
1  United States         2447.92       6315.68
```

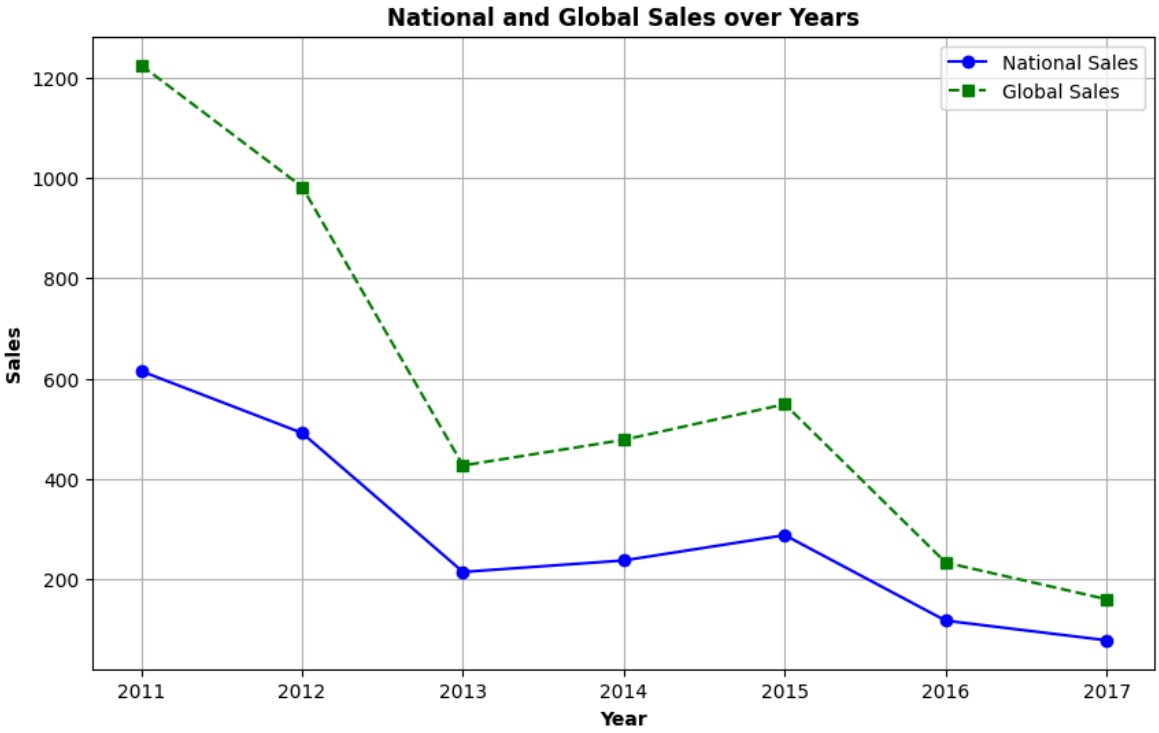
In [17]:
```
# Plotting a pie chart for National Sales by Country and Global Sales
fig, axs = plt.subplots(1,2,figsize=(10,7))
axs[0].pie(National__Sales,labels=Country,autopct='%1.1f%%',startangle=120)
axs[0].set_title('National Sales by Country')
```

```
axs[1].pie(Global_Sales,labels=Country,autopct='%1.1f%%',startangle=130)
axs[1].set_title('Global Sales by Country')
plt.show()
```



In [18]:
```
# Data preparation for Visualization.

sales_by_year = video_games_df.groupby('Year')[['National Sales', 'Global Sales'
sales_by_year_filtered = sales_by_year[sales_by_year['Year'] > 2010]
```

In [19]:
```
# Plotting a line chart for National & Global Sales by Year

plt.figure(figsize=(10,6))

plt.plot(sales_by_year_filtered['Year'], sales_by_year_filtered['National Sales'
         marker='o', linestyle='-', color='b', label='National Sales')

plt.plot(sales_by_year_filtered['Year'], sales_by_year_filtered['Global Sales'],
         marker='s', linestyle='--', color='g', label='Global Sales')

plt.title('National and Global Sales over Years',fontweight='bold')
plt.xlabel('Year',fontweight='bold')
plt.ylabel('Sales',fontweight='bold')
plt.legend()
plt.grid(True)
plt.show()
```


National and Global Sales over Years
National Sales
Global Sales
Sales
Year
1200
1000
800
600
400
200
2011
2012
2013
2014
2015
2016
2017