Exploratory Data Analysis

Data Cleaning and Preparation:

```
import pandas as pd
# Load the dataset to begin the process
file_path = '/mnt/data/vgsales.csv'
data = pd.read_csv(file_path)
```

```
# Display the first few rows of the dataset to understand its structure and variables
data.head()
```

Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
0	1	Wii Sports	Wii	2006.0	Sports	Nintendo	41.49	29.02	3.77	8.46	82.74
1	2	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08	3.58	6.81	0.77	40.24
2	3	Mario Kart Wii	Wii	2008.0	Racing	Nintendo	15.85	12.88	3.79	3.31	35.82
3	4	Wii Sports Resort	Wii	2009.0	Sports	Nintendo	15.75	11.01	3.28	2.96	33.00
4	5	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	Nintendo	11.27	8.89	10.22	1.00	31.37

```
# Check for missing values in the dataset
missing_values = data.isnull().sum()
```

```
# Display the columns with missing values and the count of missing data
```

```
Year         271
Publisher     58
dtype: int64
```

```
# Fill missing 'Year' values with the median year
data['Year'].fillna(data['Year'].median(), inplace=True)
```

```
# Fill missing 'Publisher' values with 'Unknown'
data['Publisher'].fillna('Unknown', inplace=True)
```

```
# Verify if all missing values have been addressed
data.isnull().sum()
```

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

```
# Define a function to remove outliers based on the IQR method
def remove_outliers(df, column_list):
    for column in column list:
        Q1 = df[column].quantile(0.25)
        Q3 = df[column].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR

# Filter out outliers from the dataframe
df = df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]
return df
```

```
# Columns to check for outliers
```

```
sales_columns = ['NA_Sales', 'EU_Sales', 'JP_Sales', 'Other_Sales',
'Global_Sales']
```

```
# Remove outliers from the dataset
cleaned_data = remove_outliers(data, sales_columns)
```

```
# Display the shape of the dataset before and after outlier removal for
comparison
original_shape = data.shape
cleaned_shape = cleaned_data.shape
original_shape, cleaned_shape
```

```
((16598, 11), (10240, 11))
```

```
# Define a function to cap outliers based on the IQR method
def cap_outliers(df, column_list):
for column in column_list:
        Q1 = df[column].quantile(0.25)
        Q3 = df[column].quantile(0.75)
        IQR = Q3 - Q1
        upper_bound = Q3 + 1.5 * IQR
```

```
# Cap outliers in the dataframe
        df[column] = df[column].clip(upper=upper_bound)
        return df
```

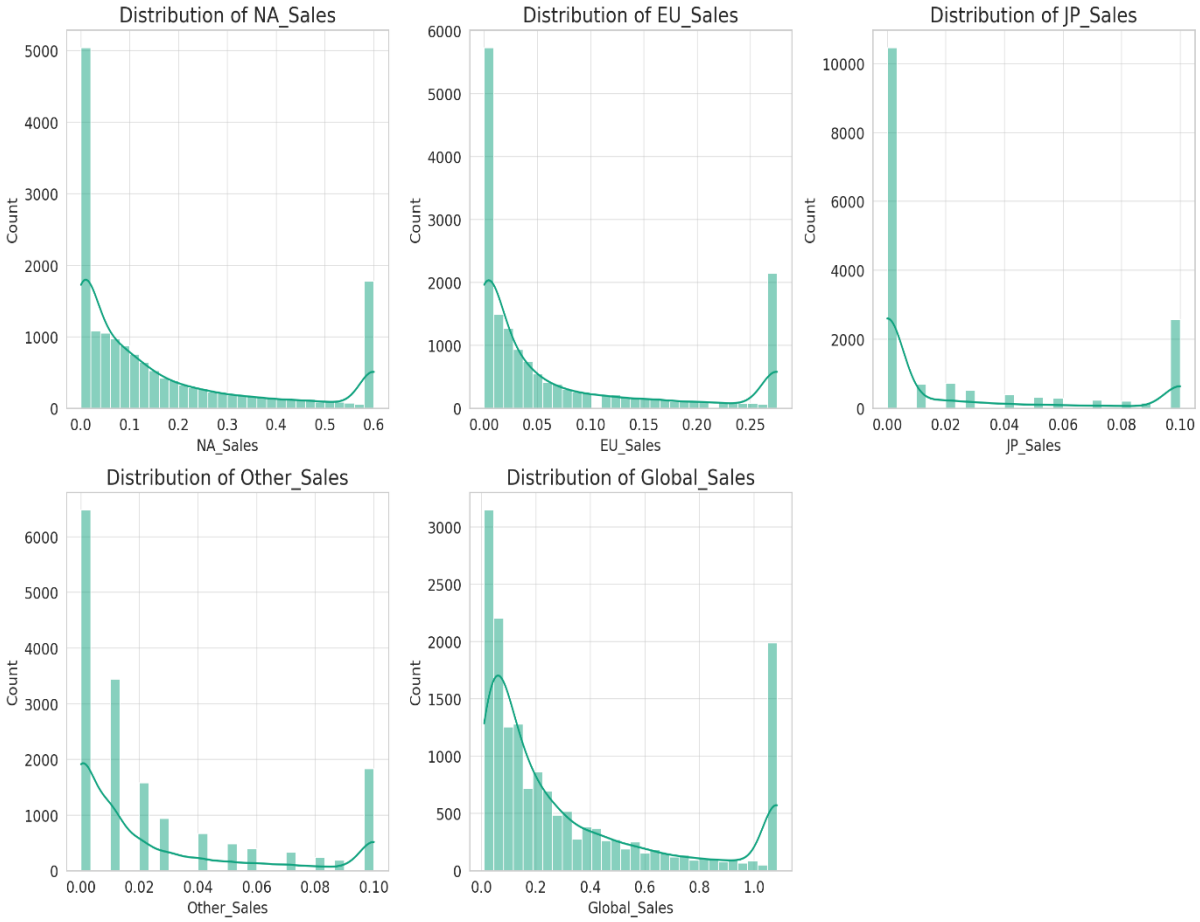
```
# Cap outliers in the dataset
capped_data = cap_outliers(data.copy(), sales_columns)
```

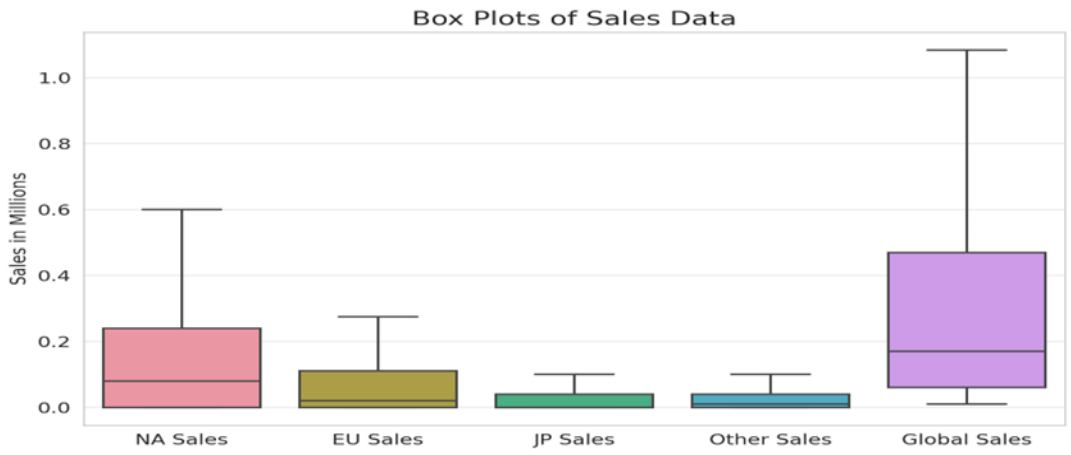
```
# Verify the changes by displaying the first few rows of the capped data
capped_data.head()
```

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

	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
0	0.6	0.275	0.1	0.1	1.085
1	0.6	0.275	0.1	0.1	1.085
2	0.6	0.275	0.1	0.1	1.085
3	0.6	0.275	0.1	0.1	1.085
4	0.6	0.275	0.1	0.1	1.085

Descriptive Analysis:






```
# Format the 'Year' column as integer

capped_data['Year'] = capped_data['Year'].astype(int)
```

```
# Perform summary statistics for key business metrics

summary_stats = capped_data[['NA_Sales', 'EU_Sales', 'JP_Sales',
'Other_Sales', 'Global_Sales']].describe()
```

```
# Add mode to the summary statistics (mode may return multiple values, so
we take the first one if that's the case)

modes = capped_data[['NA_Sales', 'EU_Sales', 'JP_Sales', 'Other_Sales',
'Global_Sales']].mode().iloc[0]

summary_stats.loc['mode'] = modes
```

```
# Visualizations: Histograms and Box Plots for Sales Data

import matplotlib.pyplot as plt

import seaborn as sns
```

```
# Set the style of seaborn

sns.set_style("whitegrid")
```

```
# Histograms for sales data

plt.figure(figsize=(15, 10))

for i, column in enumerate(['NA_Sales', 'EU_Sales', 'JP_Sales',
'Other_Sales', 'Global_Sales'], start=1):

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

```
    sns.histplot(capped_data[column], kde=True, bins=30)
    plt.title(f'Distribution of {column}')
    plt.tight_layout()
```

```
# Box plots for sales data
plt.figure(figsize=(10, 6))
sns.boxplot(data=capped_data[['NA_Sales', 'EU_Sales', 'JP_Sales', 'Other_Sales', 'Global_Sales']])
plt.title('Box Plots of Sales Data')
plt.xticks(ticks=[0, 1, 2, 3, 4], labels=['NA Sales', 'EU Sales', 'JP Sales', 'Other Sales', 'Global Sales'])
plt.ylabel('Sales in Millions')
summary_stats
```

	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
count	16598.000000	16598.000000	16598.000000	16598.000000	16598.000000
mean	0.161686	0.072202	0.023591	0.024857	0.328819
std	0.197668	0.096450	0.037706	0.033256	0.354541
min	0.000000	0.000000	0.000000	0.000000	0.010000
25%	0.000000	0.000000	0.000000	0.000000	0.060000
50%	0.080000	0.020000	0.000000	0.010000	0.170000
75%	0.240000	0.110000	0.040000	0.040000	0.470000
max	0.600000	0.275000	0.100000	0.100000	1.085000
mode	0.000000	0.000000	0.000000	0.000000	1.085000

Segmentation and Profiling:

```
# Segmentation and profiling by Genre
genre_segmentation = data.groupby('Genre').agg({
    'Global_Sales': ['mean', 'median', 'sum'],
    'NA_Sales': 'sum',
    'EU_Sales': 'sum',
    'JP_Sales': 'sum',
    'Other_Sales': 'sum'
}).sort_values(by=('Global_Sales', 'mean'), ascending=False)
```

```
# Segmentation and profiling by Platform
platform_segmentation = data.groupby('Platform').agg({

    'Global_Sales': ['mean', 'median', 'sum'],
    'NA_Sales': 'sum',
    'EU_Sales': 'sum',
    'JP_Sales': 'sum',
    'Other_Sales': 'sum'

}).sort_values(by=('Global_Sales', 'mean'), ascending=False)
   genre_segmentation, platform_segmentation
```

(

	Global_Sales			NA_Sales	EU_Sales	JP_Sales
	mean	median	sum	sum	sum	sum
Genre						
Platform	0.938341	0.280	831.37	447.05	201.63	130.77
Shooter	0.791885	0.230	1037.37	582.60	313.27	38.28
Role-Playing	0.623233	0.185	927.37	327.28	188.06	352.31
Racing	0.586101	0.190	732.04	359.42	238.39	56.69
Sports	0.567319	0.220	1330.93	683.35	376.85	135.37
Fighting	0.529375	0.210	448.91	223.59	101.32	87.35
Action	0.528100	0.190	1751.18	877.83	525.00	159.95
Misc	0.465762	0.160	809.96	410.24	215.98	107.76
Simulation	0.452364	0.160	392.20	183.31	113.38	63.70
Puzzle	0.420876	0.110	244.95	123.78	50.78	57.31
Strategy	0.257151	0.090	175.12	68.70	45.34	49.46
Adventure	0.185879	0.060	239.04	105.80	64.13	52.07

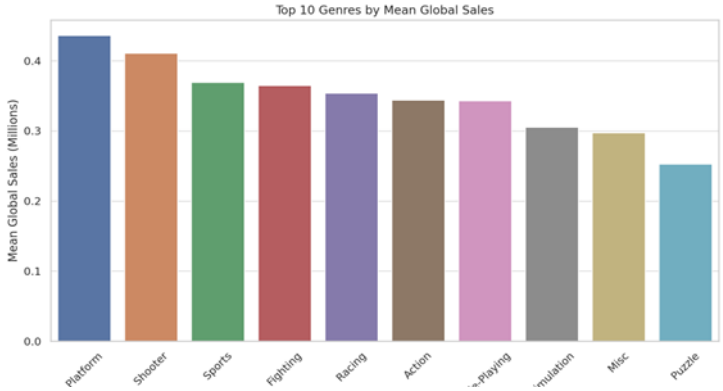
	Other_Sales
	sum
Genre	
Platform	51.59
Shooter	102.69
Role-Playing	59.61
Racing	77.27
Sports	134.97
Fighting	36.68
Action	187.38
Misc	75.32
Simulation	31.52
Puzzle	12.55
Strategy	11.36
Adventure	16.81

,

	Global_Sales		NA_Sales	EU_Sales	JP_Sales	Other_Sales
	mean	median	sum	sum	sum	sum
sum						
Platform						
GB	2.606633	1.165	255.45	114.32	47.82	85.12	8.20
NES	2.561939	1.375	251.07	125.94	21.15	98.65	5.31
GEN	1.050370	0.150	28.36	19.27	5.52	2.67	0.89
SNES	0.837029	0.320	200.05	61.23	19.04	116.55	3.22
PS4	0.827679	0.220	278.10	96.80	123.70	14.30	43.36
X360	0.774672	0.280	979.96	601.05	280.58	12.43	85.54
2600	0.729925	0.460	97.08	90.60	5.47	0.00	0.91
PS3	0.720722	0.280	957.84	392.26	343.71	79.99	141.93
Wii	0.699404	0.200	926.71	507.71	268.38	69.35	80.61
N64	0.686144	0.270	218.88	139.02	41.06	34.22	4.38
XOne	0.662254	0.240	141.06	83.19	45.65	0.34	11.92
PS	0.610920	0.260	730.66	336.51	213.60	139.82	40.91
PS2	0.581046	0.230	1255.64	583.84	339.29	139.20	193.44
WiiU	0.572448	0.230	81.86	38.32	24.23	12.79	6.45
3DS	0.486169	0.120	247.46	78.87	58.52	97.35	12.63
GBA	0.387470	0.165	318.50	187.54	75.25	47.33	7.73

DS	0.380254	0.110	822.49	390.71	194.65	175.57	60.53
GC	0.358561	0.150	199.36	133.46	38.71	21.58	5.18
XB	0.313422	0.140	258.26	186.69	60.95	1.38	8.72
SCD	0.311667	0.065	1.87	1.00	0.36	0.45	0.05
DC	0.307115	0.135	15.97	5.43	1.69	8.56	0.27
PC	0.269604	0.040	258.82	93.28	139.68	0.17	24.86
PSP	0.244254	0.090	296.28	108.99	68.25	76.79	42.19
WS	0.236667	0.215	1.42	0.00	0.00	1.42	0.00
SAT	0.194162	0.120	33.59	0.72	0.54	32.26	0.07
PSV	0.149952	0.060	61.93	16.20	16.33	20.96	8.45
NG	0.120000	0.100	1.44	0.00	0.00	1.44	0.00
TG16	0.080000	0.080	0.16	0.00	0.00	0.16	0.00
GG	0.040000	0.040	0.04	0.00	0.00	0.04	0.00
3DO	0.033333	0.020	0.10	0.00	0.00	0.10	0.00
PCFX	0.030000	0.030	0.03	0.00	0.00	0.03	0.00)

Segmentation and Profiling Analysis:



Correlation and Trends:

```
# Correlation analysis between different sales regions
correlation_matrix = capped_data[['NA_Sales', 'EU_Sales', 'JP_Sales',
'Other_Sales', 'Global_Sales']].corr()
```

```
# Trends analysis: Sales trends over the years
# Calculate total global sales per year
yearly_global_sales = capped_data.groupby('Year')['Global_Sales'].sum()
```

```
# Visualizing the trend of global sales over the years
plt.figure(figsize=(12, 6))
yearly_global_sales.plot(kind='line', marker='o', linestyle='-')
plt.title('Trend of Global Video Game Sales Over the Years')
plt.ylabel('Total Global Sales (Millions)')
plt.xlabel('Year')
plt.grid(True)
correlation_matrix, plt.show()
```

```
(              NA_Sales  EU_Sales  JP_Sales  Other_Sales  Global_Sales
 NA_Sales       1.000000  0.706698 -0.011368     0.758732      0.877403
 EU_Sales       0.706698  1.000000  0.011690     0.842957      0.809262
 JP_Sales      -0.011368  0.011690  1.000000     0.049137      0.232470
 Other_Sales    0.758732  0.842957  0.049137     1.000000      0.841219
 Global_Sales   0.877403  0.809262  0.232470     0.841219      1.000000,
 None)
```