

*Loading needed libraries*

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

```
In [ ]: video_games_df = pd.read_csv(r'C:\Users\User\PycharmProjects\PythonProject2\Other Projects\Video Games Project\Video_games_v2.csv')
video_games_df
```

*Content Dataset checks*

```
In [ ]: #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(' ')
```

*Duplicate rows check & NaN rows check*

```
In [ ]: #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_count}')
print(' ')
#NaN values check
df_NaN = video_games_df.isna().sum()
print(' ')
```

```
print('The analysis for NaN values is:')
print(df_NaN)
```

### Dropping duplicates rows

```
In [ ]: #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(' ')
```

### Handling NaN values

```
In [ ]: #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)
```

### Cleaning NA\_Sales column

```
In [ ]: #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 [ ]: # 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)
```

### Making last 10 row values of column NA\_Sales NaN to after fill with average of column

```
In [ ]: #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

```

*Standardizing categorical values in column Country*

```

In [ ]: 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())

```

*Renaming Columns*

```

In [ ]: #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)

```

*Handling Outliers in National Sales (prior NA\_Sales) Column*

```

In [ ]: 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

```

*Visualizations*

```

In [ ]: # 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)
```

```
In [ ]: # 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 [ ]: # 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 [ ]: # 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)
```

```
In [ ]: # 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 [ ]: # 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 [ ]: # 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()
```