Bilenkin550Week1_Exercise_1.2

March 15, 2025

1. Load the dataset as a Pandas data frame.

[52]:
```
import pandas as pd

# Loading the dataset
file_path = r"C:\Users\maxim\OneDrive\Desktop\BU\DSC␣
 ↪550\Video_Games_Sales_as_at_22_Dec_2016.csv"
df = pd.read_csv(file_path)
```

2. Display the first ten rows of data.

[53]:
```
# Displaying the first ten rows from the dataset
print("First 10 rows:")
print(df.head(10))
```

```
First 10 rows:
                      Name Platform  Year_of_Release         Genre  \
0                Wii Sports      Wii           2006.0        Sports
1          Super Mario Bros.      NES           1985.0      Platform
2             Mario Kart Wii      Wii           2008.0        Racing
3          Wii Sports Resort      Wii           2009.0        Sports
4   Pokemon Red/Pokemon Blue       GB           1996.0  Role-Playing
5                    Tetris       GB           1989.0        Puzzle
6      New Super Mario Bros.       DS           2006.0      Platform
7                   Wii Play      Wii           2006.0          Misc
8  New Super Mario Bros. Wii      Wii           2009.0      Platform
9                 Duck Hunt      NES           1984.0       Shooter

  Publisher  NA_Sales  EU_Sales  JP_Sales  Other_Sales  Global_Sales  \
0  Nintendo     41.36     28.96      3.77         8.45         82.53
1  Nintendo     29.08      3.58      6.81         0.77         40.24
2  Nintendo     15.68     12.76      3.79         3.29         35.52
3  Nintendo     15.61     10.93      3.28         2.95         32.77
4  Nintendo     11.27      8.89     10.22         1.00         31.37
5  Nintendo     23.20      2.26      4.22         0.58         30.26
6  Nintendo     11.28      9.14      6.50         2.88         29.80
7  Nintendo     13.96      9.18      2.93         2.84         28.92
8  Nintendo     14.44      6.94      4.70         2.24         28.32
9  Nintendo     26.93      0.63      0.28         0.47         28.31
```

	Critic_Score	Critic_Count	User_Score	User_Count	Developer	Rating
0	76.0	51.0	8	322.0	Nintendo	E
1	NaN	NaN	NaN	NaN	NaN	NaN
2	82.0	73.0	8.3	709.0	Nintendo	E
3	80.0	73.0	8	192.0	Nintendo	E
4	NaN	NaN	NaN	NaN	NaN	NaN
5	NaN	NaN	NaN	NaN	NaN	NaN
6	89.0	65.0	8.5	431.0	Nintendo	E
7	58.0	41.0	6.6	129.0	Nintendo	E
8	87.0	80.0	8.4	594.0	Nintendo	E
9	NaN	NaN	NaN	NaN	NaN	NaN

3. Find the dimensions (number of rows and columns) in the data frame. What do these two numbers represent in the context of the data?

[54]:
```
# Finding the dimensions of the dataset
rows, cols = df.shape

# Displaying the number of rows and columns
print(f"\nNumber of rows: {rows}, Number of columns: {cols}")

# Explaining what the rows and columns represent
print("The number of rows represents the total count of video games, while the␣
 ↪columns describe various attributes of each game.")
```

```
Number of rows: 16719, Number of columns: 16
The number of rows represents the total count of video games, while the columns
describe various attributes of each game.
```

4. Find the top five games by critic score.

[55]:
```
# Finding the top five games by critic score
top_5_critic = df.nlargest(5, 'Critic_Score')

# Displaying output
print("\nHere are the top five games listed by Critic Score:")
print(top_5_critic[['Name', 'Critic_Score']])
```

```
Here are the top five games listed by Critic Score:
                          Name  Critic_Score
51          Grand Theft Auto IV          98.0
57          Grand Theft Auto IV          98.0
227   Tony Hawk's Pro Skater 2          98.0
5350                SoulCalibur          98.0
16           Grand Theft Auto V          97.0
```

5. Find the number of video games in the data frame in each genre.

[56]:
```
# Counting of video games in each genre
genre_counts = df['Genre'].value_counts()

# Displaying result
print("\nThe total number of video games in each genre:")
print(genre_counts)
```

```
The total number of video games in each genre:
Genre
Action          3370
Sports          2348
Misc            1750
Role-Playing    1500
Shooter         1323
Adventure       1303
Racing          1249
Platform         888
Simulation       874
Fighting         849
Strategy         683
Puzzle           580
Name: count, dtype: int64
```

6. Find the first five games in the data frame on the SNES platform.

[57]:
```
# Finding the first five games on the SNES platform
snes_games = df[df['Platform'] == 'SNES'].head(5)

# Displaying result
print("\nThe first five games in the data on the SNES platform:")
print(snes_games[['Name', 'Platform']])
```

```
The first five games in the data on the SNES platform:
                                    Name Platform
18                     Super Mario World     SNES
56                 Super Mario All-Stars     SNES
71                   Donkey Kong Country     SNES
76                      Super Mario Kart     SNES
137  Street Fighter II: The World Warrior     SNES
```

7. Find the five publishers with the highest total global sales. Note: You will need to calculate the total global sales for each publisher to do this.

[58]:
```
# Finding the top five publishers by total global sales
top_five_publishers = df.groupby('Publisher')['Global_Sales'].sum().nlargest(5)

# Displaying result
```

```
print("\nTop 5 publishers by total global sales:")
print(top_five_publishers)
```

```
Top 5 publishers by total global sales:
Publisher
Nintendo                         1788.81
Electronic Arts                  1116.96
Activision                        731.16
Sony Computer Entertainment       606.48
Ubisoft                           471.61
Name: Global_Sales, dtype: float64
```

Create a new column in the data frame that calculates the percentage of global sales from North America. Display the first five rows of the new data frame.

[59]:
```
# Creating a new column in the data frame for NA sales percentage
df['NA_Sales_Percentage'] = (df['NA_Sales'] / df['Global_Sales'].replace(0,␣
 ↪float('nan'))) * 100

# Displaying result of the first five rows
print("\nThe first five rows with North America Sales Percentage:")
print(df[['Name', 'NA_Sales_Percentage']].head())
```

```
The first five rows with North America Sales Percentage:
                      Name  NA_Sales_Percentage
0                Wii Sports            50.115110
1         Super Mario Bros.            72.266402
2            Mario Kart Wii            44.144144
3         Wii Sports Resort            47.635032
4  Pokemon Red/Pokemon Blue            35.926044
```

9. Find the number NaN entries (missing data values) in each column.

[60]:
```
# Finding and counting the NaN entries (missing values) in each column
missing_values_in_each_column = df.isna().sum()

# Displaying result
print("\nTotal number of missing values in each column:")
print(missing_values_in_each_column.to_string())
```

```
Total number of missing values in each column:
Name                     2
Platform                 0
Year_of_Release        269
Genre                    2
Publisher               54
NA_Sales                 0
```

```
EU_Sales                  0
JP_Sales                  0
Other_Sales               0
Global_Sales              0
Critic_Score           8582
Critic_Count           8582
User_Score             6704
User_Count             9129
Developer              6623
Rating                 6769
NA_Sales_Percentage       0
```

10. Try to calculate the median user score of all the video games. You will likely run into an error because some of the user score entries are a non-numerical string that cannot be converted to a float. Find and replace this string with NaN and then calculate the median. Then, replace all NaN entries in the user score column with the median value.

[61]:
```
# Converting 'User_Score' to numeric and replacing errors with NaN
df['User_Score'] = pd.to_numeric(df['User_Score'], errors='coerce')

# Checking if there are valid numeric values before computing the median
if df['User_Score'].notna().sum() > 0:
    # Calculating the median of the User Score column
    median_user_score = df['User_Score'].median()

    # Replacing NaN values in User Score with the median (without inplace=True)
    df['User_Score'] = df['User_Score'].fillna(median_user_score)

    print(f"\nThe median user score after handling non-numeric values:␣
 ↪{median_user_score}")
else:
    print("\nNo valid numeric values found in User_Score.")
```

```
The median user score after handling non-numeric values: 7.5
```