## Contents

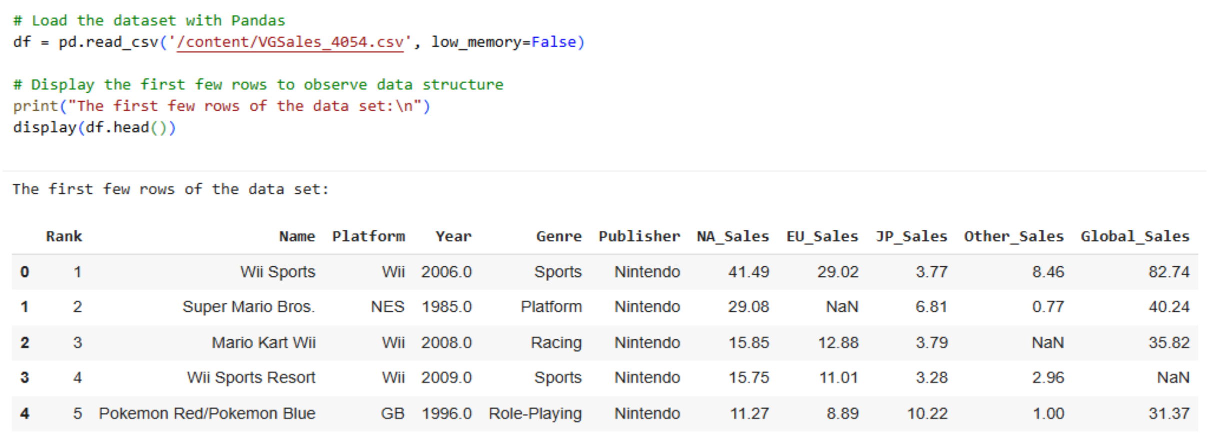
## Introduction

The goal of this assignment is to perform a complete EDA process, from data loading through descriptive, inferential, correlational, and predictive analysis. At each stage, generate insights that can support decision-making for International Media Limited. The process highlights the importance of rigorous cleaning, the balance between statistical methods and business interpretation, and the limitations of working with aggregated datasets.

# Data Analysis in Python

## Data Loading and Exploration

The very first step in this analysis task is to load the video game sales dataset to perform the necessary analysis using Python. Since the Pandas library is a common tool for working with various data types in Python, the “read_csv()” function of Pandas is utilised to read the file as can be seen in the following Figure 1. Initial checking of the size of the data, column names, and first few rows are aimed to accomplish for understanding the structure of the dataset (McKinney, 2012).

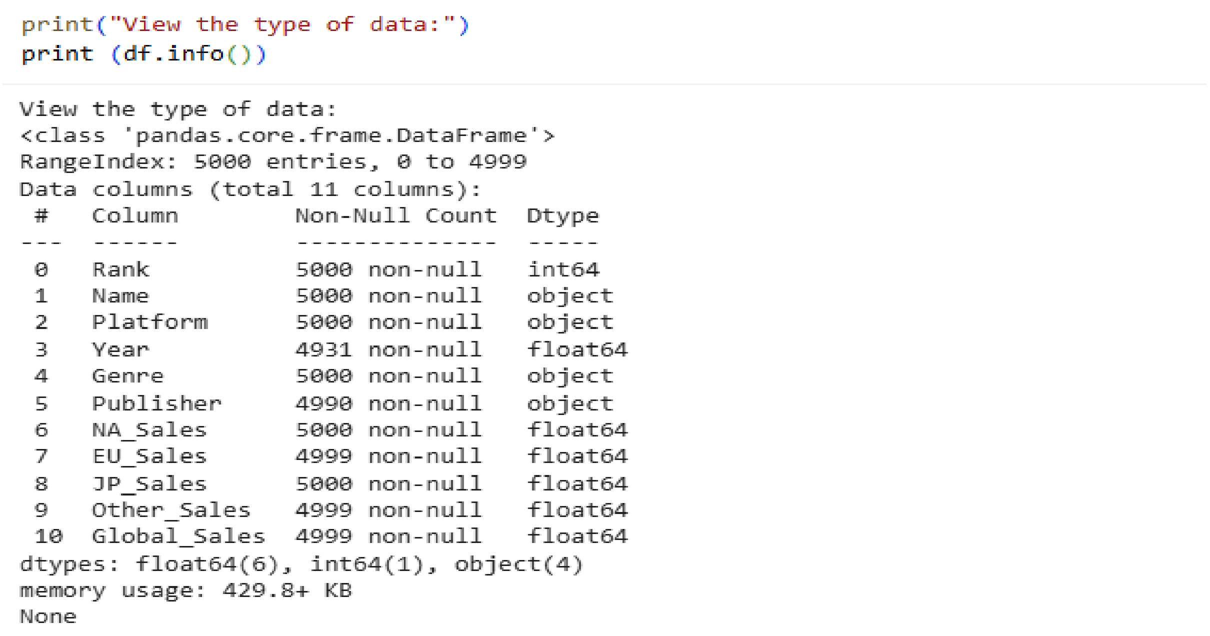


```python
# Load the dataset with Pandas
df = pd.read_csv('/content/VGSales_4054.csv', low_memory=False)

# Display the first few rows to observe data structure
print("The first few rows of the data set:\n")
display(df.head())
```

```
The first few rows of the data set:
```

| | 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 | NaN | 6.81 | 0.77 | 40.24 |
| **2** | 3 | Mario Kart Wii | Wii | 2008.0 | Racing | Nintendo | 15.85 | 12.88 | 3.79 | NaN | 35.82 |
| **3** | 4 | Wii Sports Resort | Wii | 2009.0 | Sports | Nintendo | 15.75 | 11.01 | 3.28 | 2.96 | NaN |
| **4** | 5 | Pokemon Red/Pokemon Blue | GB | 1996.0 | Role-Playing | Nintendo | 11.27 | 8.89 | 10.22 | 1.00 | 31.37 |

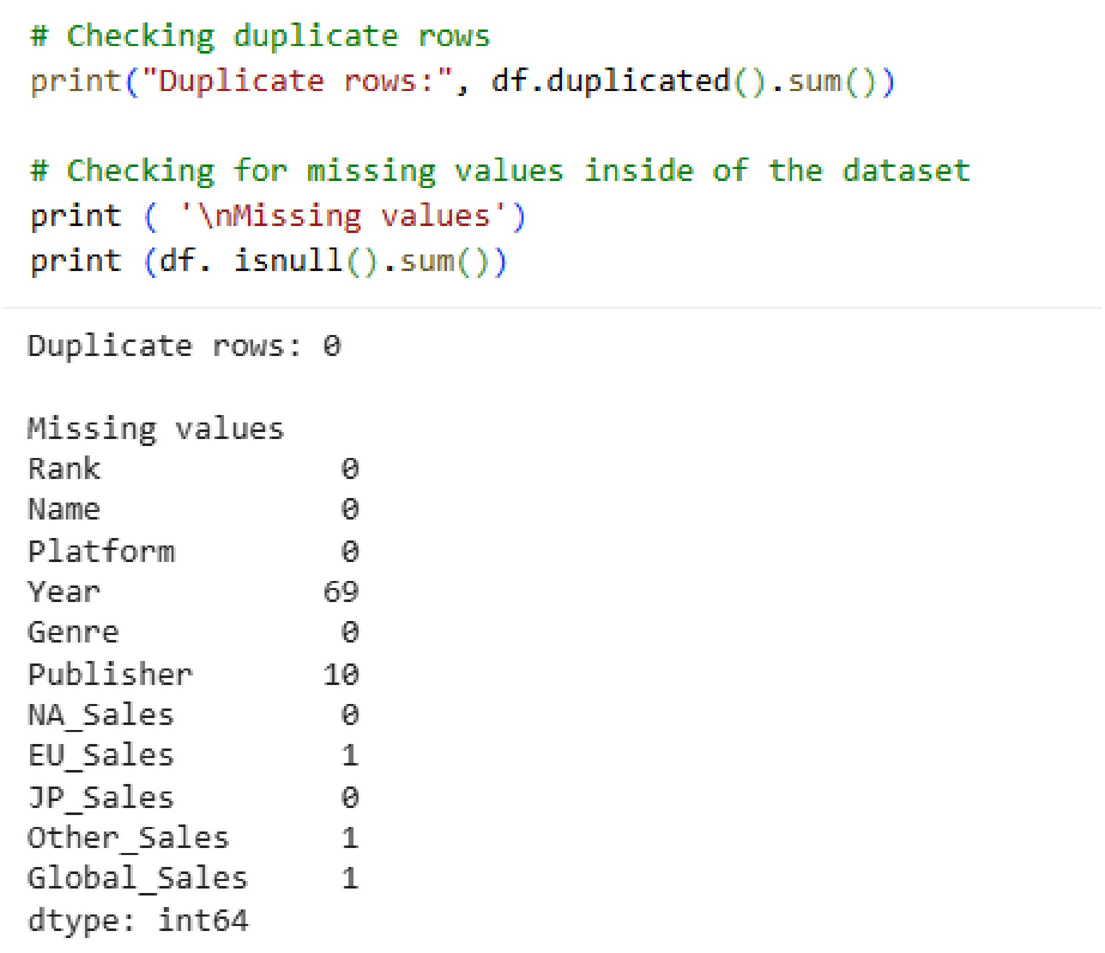
Figure 1: Data loaded and displayed on Colaboratory (Google Colab, n.d.).



```python
print("View the type of data:")
print (df.info())
```

```
View the type of data:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Rank          5000 non-null   int64
 1   Name          5000 non-null   object
 2   Platform      5000 non-null   object
 3   Year          4931 non-null   float64
 4   Genre         5000 non-null   object
 5   Publisher     4990 non-null   object
 6   NA_Sales      5000 non-null   float64
 7   EU_Sales      4999 non-null   float64
 8   JP_Sales      5000 non-null   float64
 9   Other_Sales   4999 non-null   float64
 10  Global_Sales  4999 non-null   float64
dtypes: float64(6), int64(1), object(4)
memory usage: 429.8+ KB
None
```

Figure 2: Columns of the dataset are printed to observe (personal collection).



```
# Checking duplicate rows
print("Duplicate rows:", df.duplicated().sum())

# Checking for missing values inside of the dataset
print ( '\nMissing values')
print (df. isnull().sum())
```

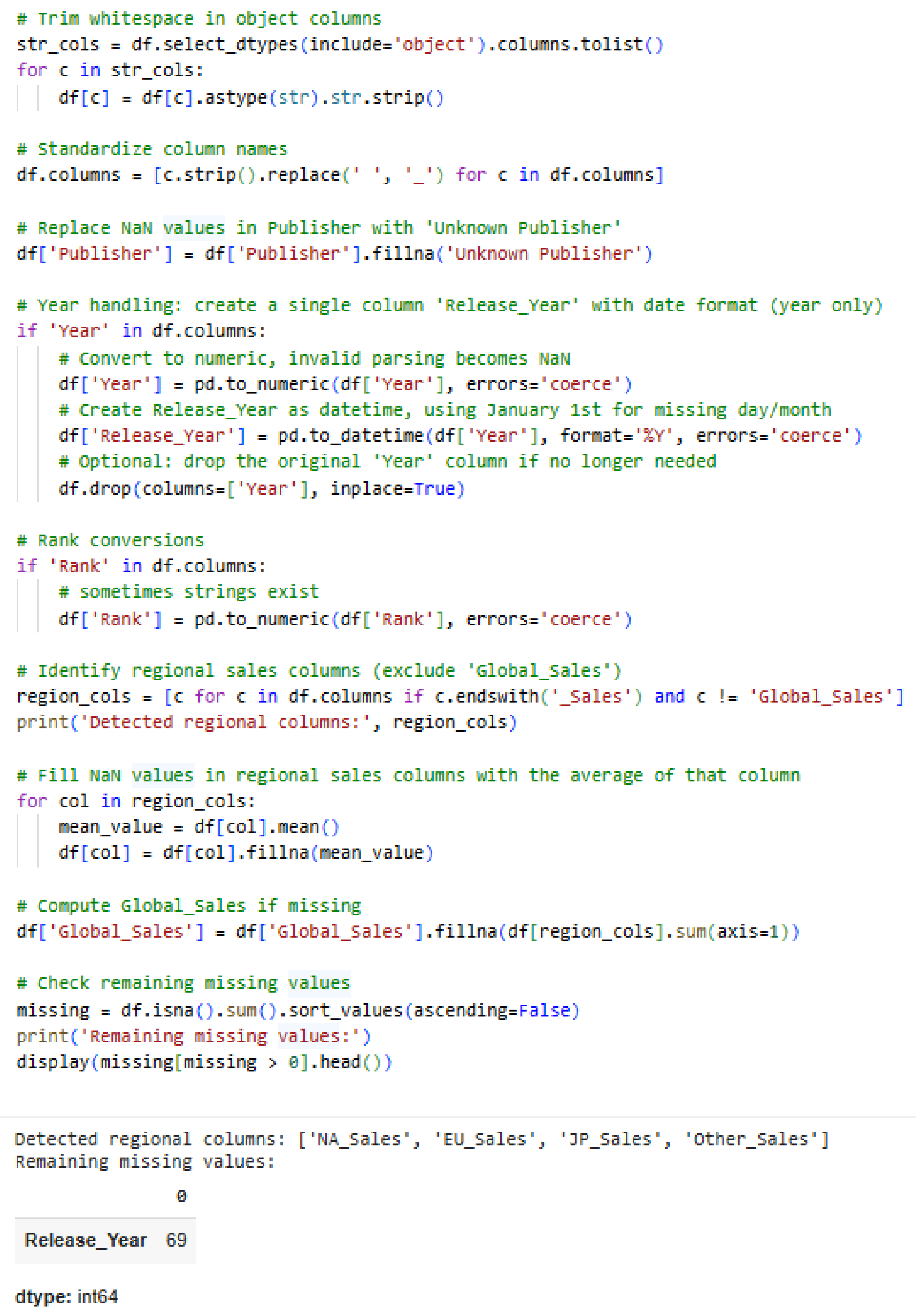
```
Duplicate rows: 0

Missing values
Rank             0
Name             0
Platform         0
Year            69
Genre            0
Publisher       10
NA_Sales         0
EU_Sales         1
JP_Sales         0
Other_Sales      1
Global_Sales     1
dtype: int64
```

Figure 3: Duplicate rows and null value observation (personal collection).

After loading the dataset, the data types for each column have been observed, such as numbers for sales values and text for publisher names as in Figure 2. This step is very important because the wrong data type can cause errors in later analysis (VanderPlas, 2016). Missing values and duplicate rows are also checked as seen in Figure 3. As the part of the data exploration stage these checks gives an early view of possible problems and helps to plan the cleaning process (Han, Pei and Tong, 2022).

## Handling Data Pre-Processing & Integrity Improvement

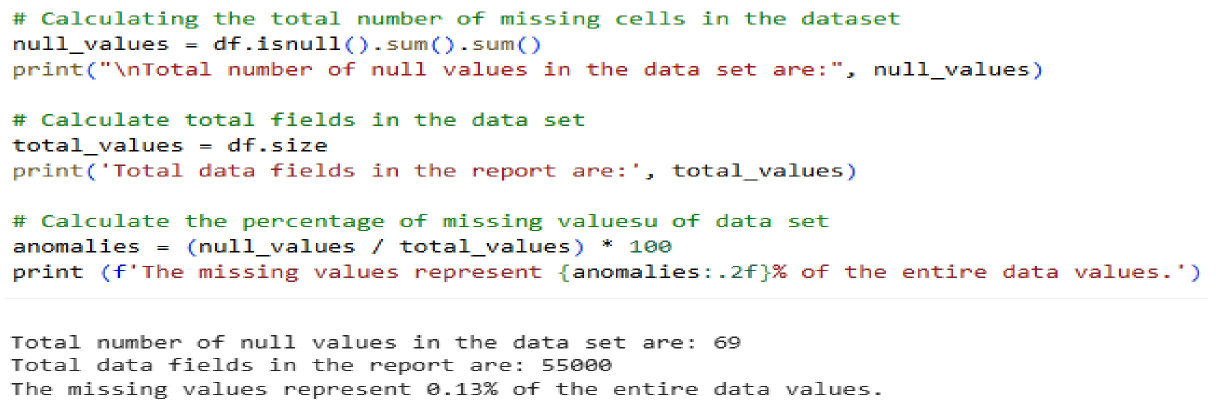
After carefully observing the dataset, several cleaning steps is suggested to make the data ready for analysis. As seen in the following Figure 4, white spaces in text columns need to be removed, and column names are to be standardised for avoiding errors when coding in Python. Missing values are handled by filling empty publisher names with “Unknown Publisher”. Missing regional sales figures are replaced with the mean of each region and global sales figures with the total of related regional sales. This process helps to keep the dataset complete and prevents bias caused by missing data (Han, Pei and Tong, 2022).



```python
# Trim whitespace in object columns
str_cols = df.select_dtypes(include='object').columns.tolist()
for c in str_cols:
    df[c] = df[c].astype(str).str.strip()

# Standardize column names
df.columns = [c.strip().replace(' ', '_') for c in df.columns]

# Replace NaN values in Publisher with 'Unknown Publisher'
df['Publisher'] = df['Publisher'].fillna('Unknown Publisher')

# Year handling: create a single column 'Release_Year' with date format (year only)
if 'Year' in df.columns:
    # Convert to numeric, invalid parsing becomes NaN
    df['Year'] = pd.to_numeric(df['Year'], errors='coerce')
    # Create Release_Year as datetime, using January 1st for missing day/month
    df['Release_Year'] = pd.to_datetime(df['Year'], format='%Y', errors='coerce')
    # Optional: drop the original 'Year' column if no longer needed
    df.drop(columns=['Year'], inplace=True)

# Rank conversions
if 'Rank' in df.columns:
    # sometimes strings exist
    df['Rank'] = pd.to_numeric(df['Rank'], errors='coerce')

# Identify regional sales columns (exclude 'Global_Sales')
region_cols = [c for c in df.columns if c.endswith('_Sales') and c != 'Global_Sales']
print('Detected regional columns:', region_cols)

# Fill NaN values in regional sales columns with the average of that column
for col in region_cols:
    mean_value = df[col].mean()
    df[col] = df[col].fillna(mean_value)

# Compute Global_Sales if missing
df['Global_Sales'] = df['Global_Sales'].fillna(df[region_cols].sum(axis=1))

# Check remaining missing values
missing = df.isna().sum().sort_values(ascending=False)
print('Remaining missing values:')
display(missing[missing > 0].head())
```

```
Detected regional columns: ['NA_Sales', 'EU_Sales', 'JP_Sales', 'Other_Sales']
Remaining missing values:

                0
Release_Year   69

dtype: int64
```

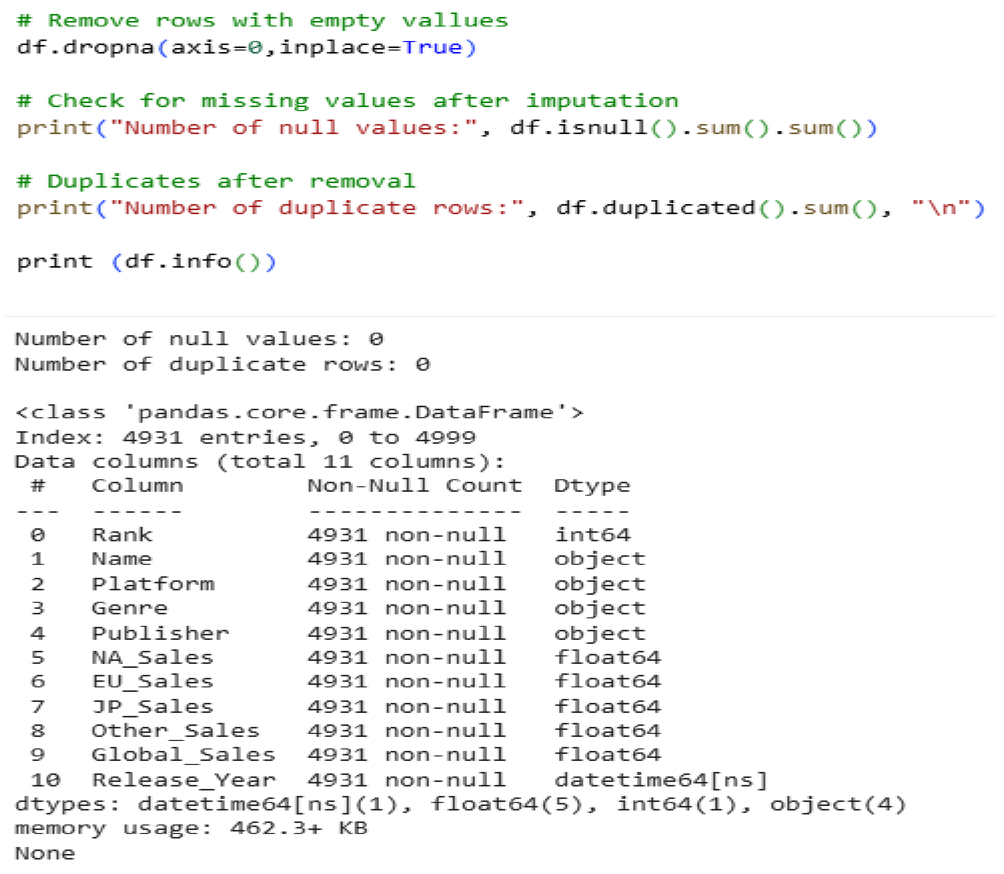
Figure 4:  Applied data pre-processing techniques (personal collection).



```
# Calculating the total number of missing cells in the dataset
null_values = df.isnull().sum().sum()
print("\nTotal number of null values in the data set are:", null_values)

# Calculate total fields in the data set
total_values = df.size
print('Total data fields in the report are:', total_values)

# Calculate the percentage of missing valuesu of data set
anomalies = (null_values / total_values) * 100
print (f'The missing values represent {anomalies:.2f}% of the entire data values.')
```

```
Total number of null values in the data set are: 69
Total data fields in the report are: 55000
The missing values represent 0.13% of the entire data values.
```

Figure 5: Calculated the proportion of missing values after data cleaning (personal collection).



```
# Remove rows with empty vallues
df.dropna(axis=0,inplace=True)

# Check for missing values after imputation
print("Number of null values:", df.isnull().sum().sum())

# Duplicates after removal
print("Number of duplicate rows:", df.duplicated().sum(), "\n")

print (df.info())
```

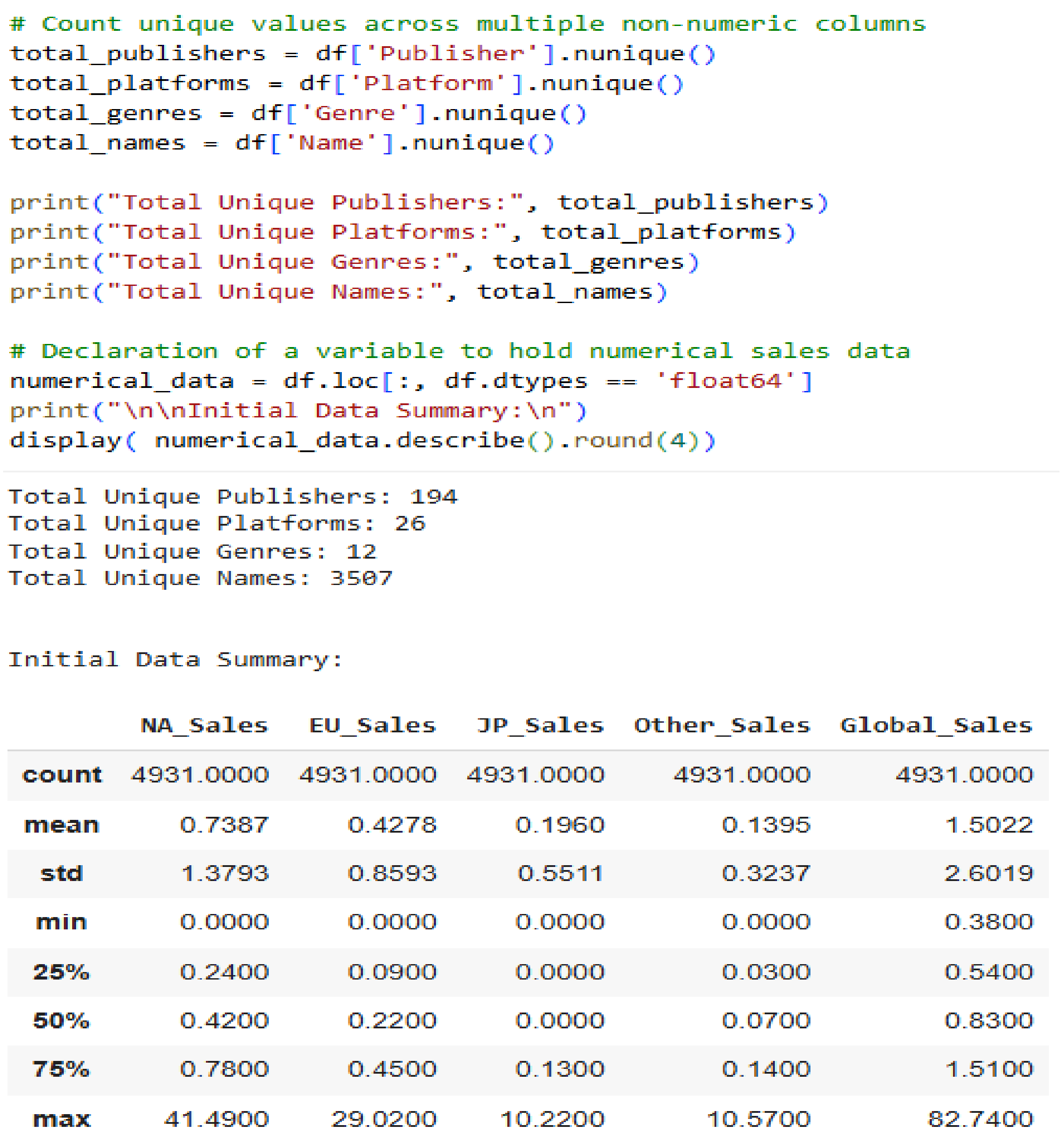
```
Number of null values: 0
Number of duplicate rows: 0

<class 'pandas.core.frame.DataFrame'>
Index: 4931 entries, 0 to 4999
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Rank          4931 non-null   int64
 1   Name          4931 non-null   object
 2   Platform      4931 non-null   object
 3   Genre         4931 non-null   object
 4   Publisher     4931 non-null   object
 5   NA_Sales      4931 non-null   float64
 6   EU_Sales      4931 non-null   float64
 7   JP_Sales      4931 non-null   float64
 8   Other_Sales   4931 non-null   float64
 9   Global_Sales  4931 non-null   float64
 10  Release_Year  4931 non-null   datetime64[ns]
dtypes: datetime64[ns](1), float64(5), int64(1), object(4)
memory usage: 462.3+ KB
None
```

Figure 6: Removal of rows containing null values is applied (personal collection).

After performing the initial pre-processing, there are 69 missing values still exist in the "Release_Year" column which is 0.13% of the total data values as calculated in Figure 5. The total release year is 37 (from 2016- to 1980) so the mean or any random imputation for this column may affect the outcome of analysis. Since the dataset is large enough and the current missing values are less than 5% also data deletion is acceptable in this case according to ApX Machine Learning (n.d.), it can be seen in Figure 6 that they were deleted instead of being imputed.

## Descriptive Statistical Analysis

In order to find underlying patterns and relationships, descriptive statistics describe the data for gathering, organising, summarising, and presenting data. Central tendency and distribution measures, such as mean, range, variance, and standard deviation, are calculated with the primary goal of giving a concise and understandable summary of the data (Scott, D., 2020). All useful tools for visualising descriptive statistics have been utilised such as bar charts, line charts etc. to discover hidden insights of dataset.

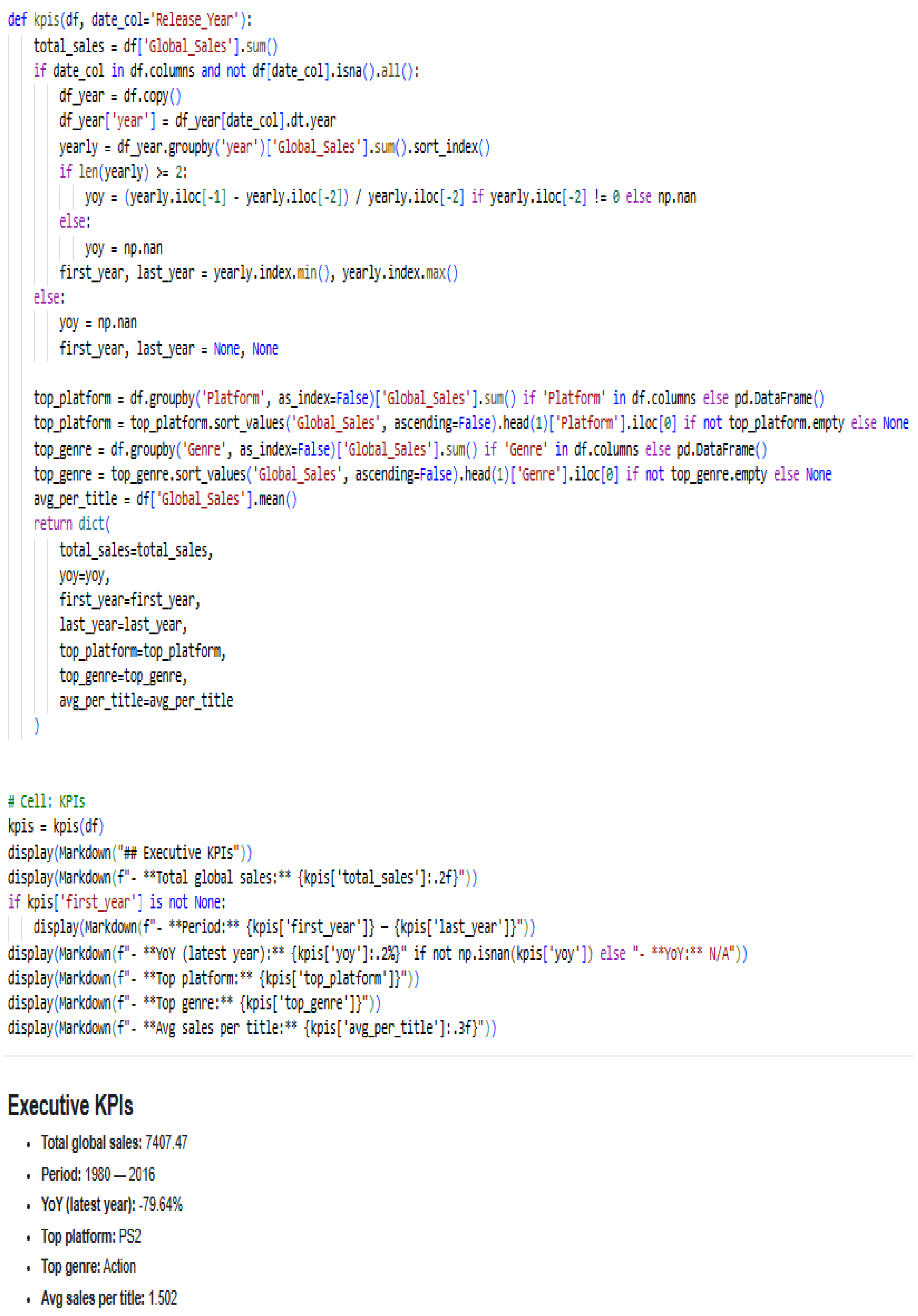


```
# Count unique values across multiple non-numeric columns
total_publishers = df['Publisher'].nunique()
total_platforms = df['Platform'].nunique()
total_genres = df['Genre'].nunique()
total_names = df['Name'].nunique()

print("Total Unique Publishers:", total_publishers)
print("Total Unique Platforms:", total_platforms)
print("Total Unique Genres:", total_genres)
print("Total Unique Names:", total_names)

# Declaration of a variable to hold numerical sales data
numerical_data = df.loc[:, df.dtypes == 'float64']
print("\n\nInitial Data Summary:\n")
display( numerical_data.describe().round(4))
```

```
Total Unique Publishers: 194
Total Unique Platforms: 26
Total Unique Genres: 12
Total Unique Names: 3507


Initial Data Summary:
```

| | NA_Sales | EU_Sales | JP_Sales | Other_Sales | Global_Sales |
|---|---|---|---|---|---|
| **count** | 4931.0000 | 4931.0000 | 4931.0000 | 4931.0000 | 4931.0000 |
| **mean** | 0.7387 | 0.4278 | 0.1960 | 0.1395 | 1.5022 |
| **std** | 1.3793 | 0.8593 | 0.5511 | 0.3237 | 2.6019 |
| **min** | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.3800 |
| **25%** | 0.2400 | 0.0900 | 0.0000 | 0.0300 | 0.5400 |
| **50%** | 0.4200 | 0.2200 | 0.0000 | 0.0700 | 0.8300 |
| **75%** | 0.7800 | 0.4500 | 0.1300 | 0.1400 | 1.5100 |
| **max** | 41.4900 | 29.0200 | 10.2200 | 10.5700 | 82.7400 |

Figure 7: Overall summary statistics of video game sales dataset (personal collection).



```
def kpis(df, date_col='Release_Year'):
    total_sales = df['Global_Sales'].sum()
    if date_col in df.columns and not df[date_col].isna().all():
        df_year = df.copy()
        df_year['year'] = df_year[date_col].dt.year
        yearly = df_year.groupby('year')['Global_Sales'].sum().sort_index()
        if len(yearly) >= 2:
            yoy = (yearly.iloc[-1] - yearly.iloc[-2]) / yearly.iloc[-2] if yearly.iloc[-2] != 0 else np.nan
        else:
            yoy = np.nan
        first_year, last_year = yearly.index.min(), yearly.index.max()
    else:
        yoy = np.nan
        first_year, last_year = None, None

    top_platform = df.groupby('Platform', as_index=False)['Global_Sales'].sum() if 'Platform' in df.columns else pd.DataFrame()
    top_platform = top_platform.sort_values('Global_Sales', ascending=False).head(1)['Platform'].iloc[0] if not top_platform.empty else None
    top_genre = df.groupby('Genre', as_index=False)['Global_Sales'].sum() if 'Genre' in df.columns else pd.DataFrame()
    top_genre = top_genre.sort_values('Global_Sales', ascending=False).head(1)['Genre'].iloc[0] if not top_genre.empty else None
    avg_per_title = df['Global_Sales'].mean()
    return dict(
        total_sales=total_sales,
        yoy=yoy,
        first_year=first_year,
        last_year=last_year,
        top_platform=top_platform,
        top_genre=top_genre,
        avg_per_title=avg_per_title
    )


# Cell: KPIs
kpis = kpis(df)
display(Markdown("## Executive KPIs"))
display(Markdown(f"- **Total global sales:** {kpis['total_sales']:.2f}"))
if kpis['first_year'] is not None:
    display(Markdown(f"- **Period:** {kpis['first_year']} – {kpis['last_year']}"))
display(Markdown(f"- **YoY (latest year):** {kpis['yoy']:.2%}" if not np.isnan(kpis['yoy']) else "- **YoY:** N/A"))
display(Markdown(f"- **Top platform:** {kpis['top_platform']}"))
display(Markdown(f"- **Top genre:** {kpis['top_genre']}"))
display(Markdown(f"- **Avg sales per title:** {kpis['avg_per_title']:.3f}"))
```

## Executive KPIs

- **Total global sales:** 7407.47
- **Period:** 1980 – 2016
- **YoY (latest year):** -79.64%
- **Top platform:** PS2
- **Top genre:** Action
- **Avg sales per title:** 1.502

Figure 8: Calculations of executive KPIs (personal collection).

```python
import matplotlib.pyplot as plt
import seaborn as sns

# Use seaborn style
sns.set(style="whitegrid")

# Select regional sales
region_cols = ['NA_Sales', 'EU_Sales', 'JP_Sales', 'Other_Sales']
sales_totals = df[region_cols].sum()

# Pie chart
plt.figure(figsize=(6, 6))
plt.pie(
    sales_totals,
    labels=sales_totals.index,
    autopct='%1.1f%%',        # show percentages with 1 decimal
    startangle=140,
    colors=sns.color_palette("Set2")  # Seaborn color palette
)
plt.title("Regional Sales Distribution", fontsize=14)
plt.show()
```

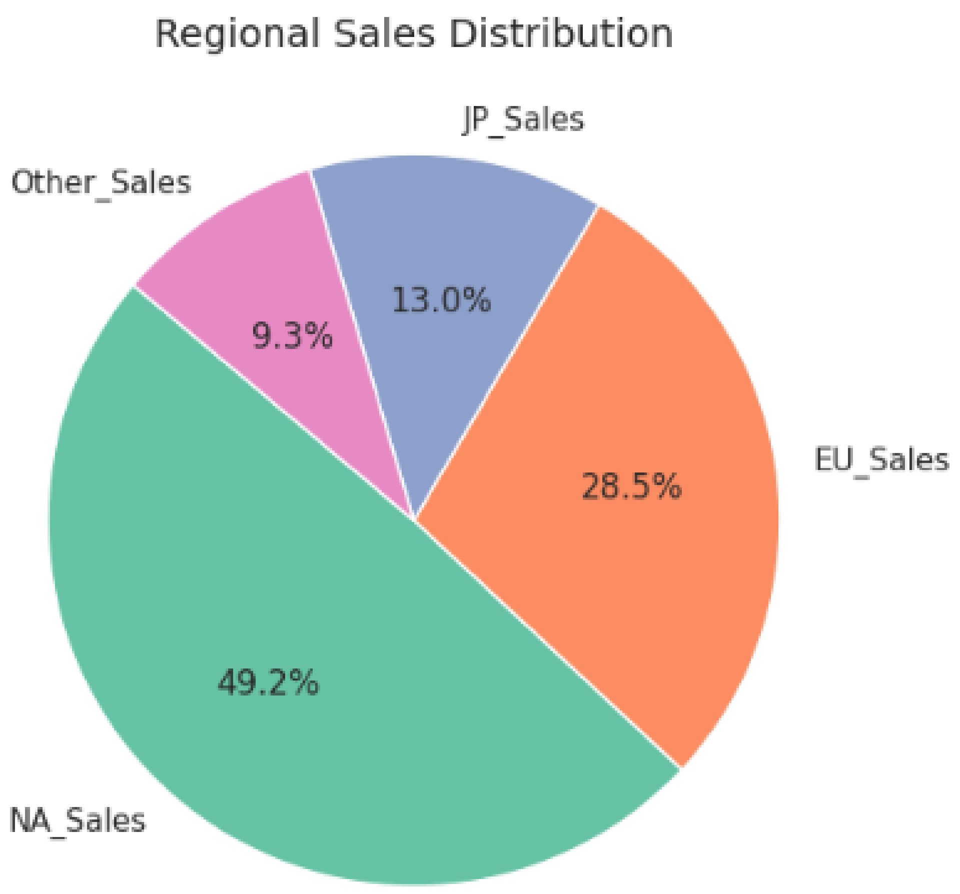
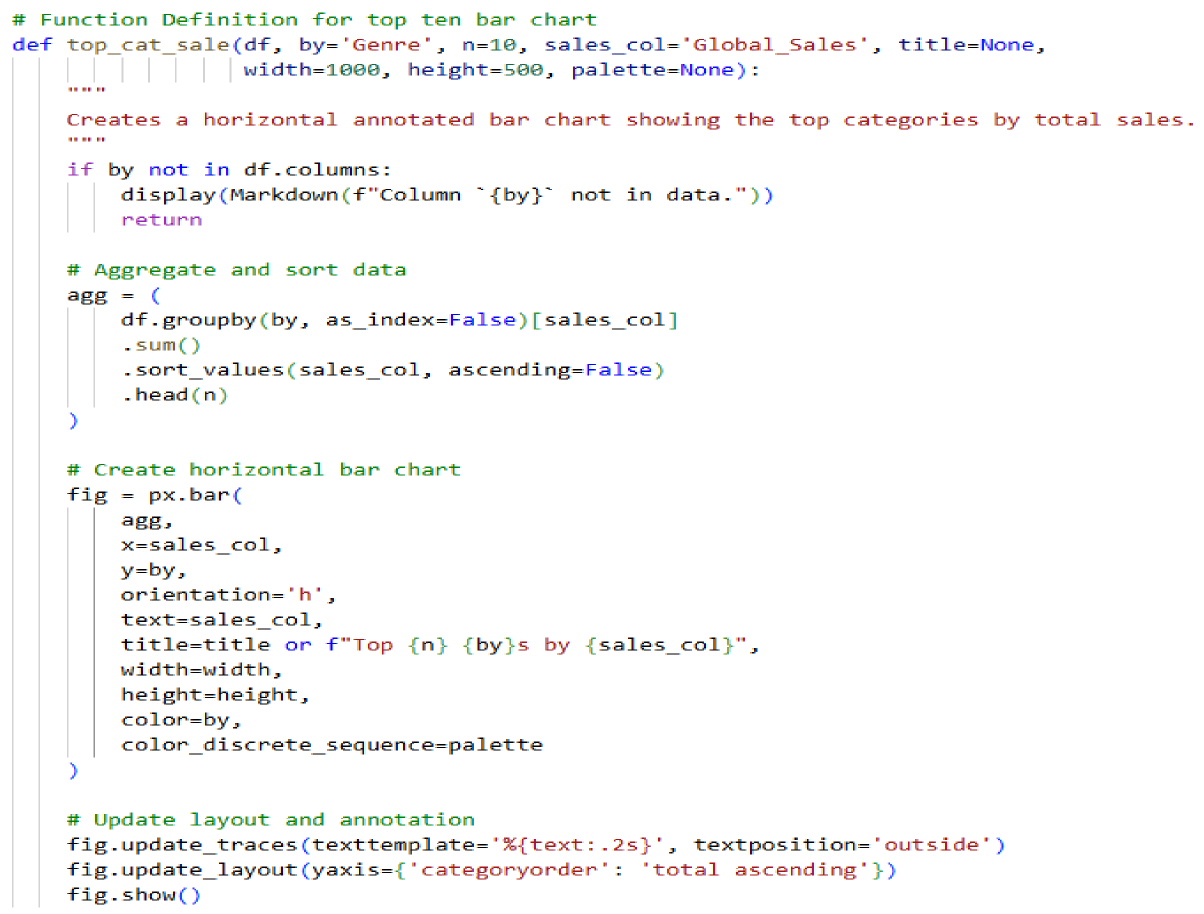
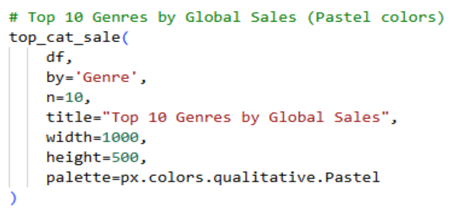


Figure 9: Pi chart for regional sales distribution (personal collection).



```
# Function Definition for top ten bar chart
def top_cat_sale(df, by='Genre', n=10, sales_col='Global_Sales', title=None,
                 width=1000, height=500, palette=None):
    """
    Creates a horizontal annotated bar chart showing the top categories by total sales.
    """
    if by not in df.columns:
        display(Markdown(f"Column `{by}` not in data."))
        return

    # Aggregate and sort data
    agg = (
        df.groupby(by, as_index=False)[sales_col]
        .sum()
        .sort_values(sales_col, ascending=False)
        .head(n)
    )

    # Create horizontal bar chart
    fig = px.bar(
        agg,
        x=sales_col,
        y=by,
        orientation='h',
        text=sales_col,
        title=title or f"Top {n} {by}s by {sales_col}",
        width=width,
        height=height,
        color=by,
        color_discrete_sequence=palette
    )

    # Update layout and annotation
    fig.update_traces(texttemplate='%{text:.2s}', textposition='outside')
    fig.update_layout(yaxis={'categoryorder': 'total ascending'})
    fig.show()
```

Figure 10.1: Function for bar charts (personal collection).



```
# Top 10 Genres by Global Sales (Pastel colors)
top_cat_sale(
    df,
    by='Genre',
    n=10,
    title="Top 10 Genres by Global Sales",
    width=1000,
    height=500,
    palette=px.colors.qualitative.Pastel
)
```

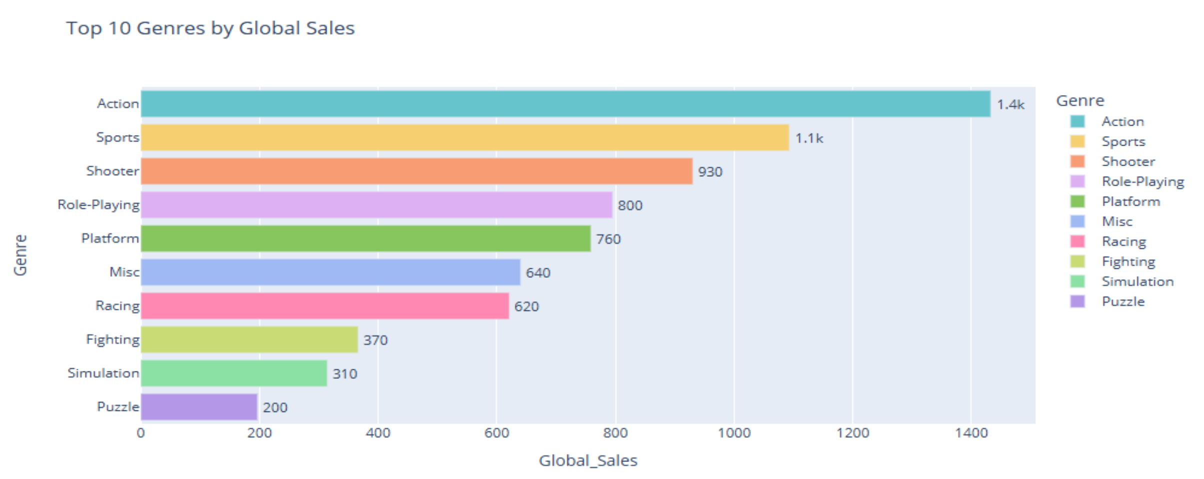


Figure 10.2: Bar chart of top 10 Genres by Global Sales (personal collection)

```
# Top 10 Platforms by Global Sales (Viridis sequential gradient)
top_cat_sale(
    df,
    by='Platform',
    n=10,
    title="Top 10 Platforms by Global Sales",
    width=1000,
    height=500,
    palette=px.colors.sequential.Viridis
)
```

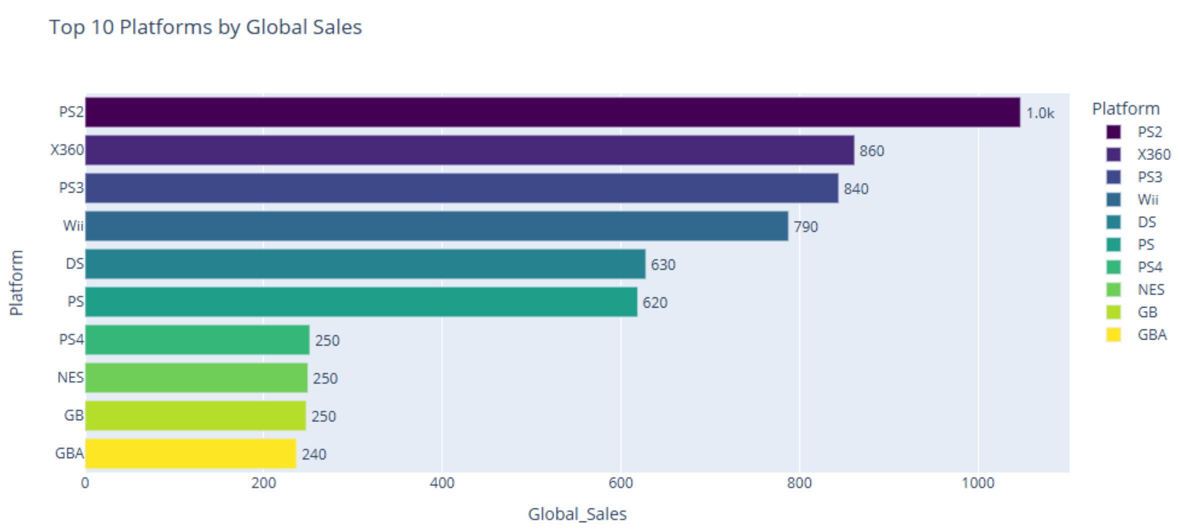


Figure 11: Bar chart of top 10 Platforms by Global Sales (personal collection).

```
# Top 10 Publishers by Global Sales (Set2 qualitative palette)
top_cat_sale(
    df,
    by='Publisher',
    n=10,
    title="Top 10 Publishers by Global Sales",
    width=1000,
    height=500,
    palette=px.colors.qualitative.Set2
)
```

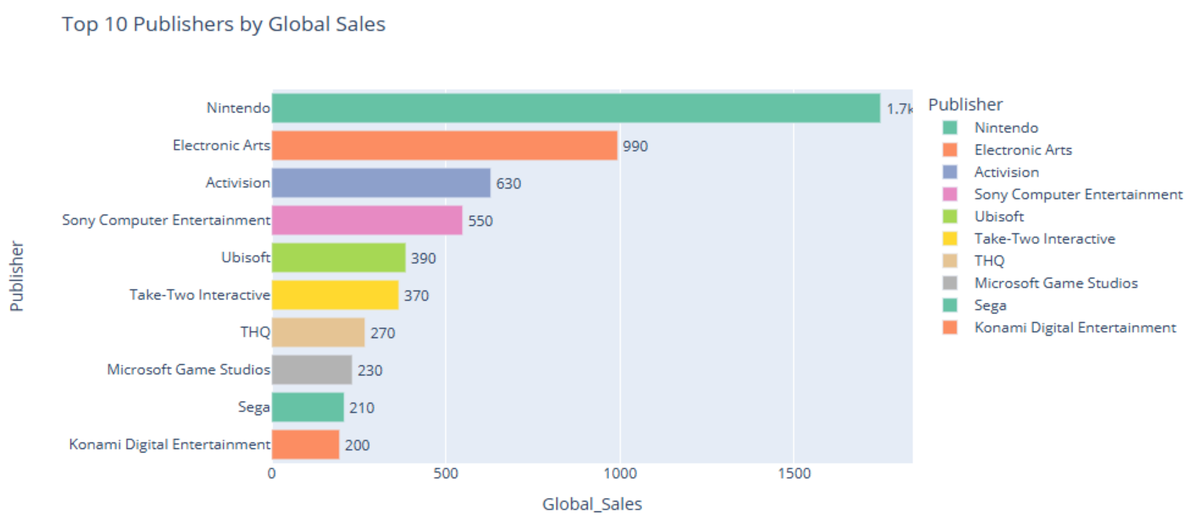


Figure 12: Bar chart of top 10 Publishers by Global Sales (personal collection).

```python
# Cell: Sales over time (aggregate) with annotations for peaks
def aggregate_timeseries(df, date_col='Release_Year', sales_col='Global_Sales', freq='YE'):
    ts = df.dropna(subset=[date_col]).copy()
    ts = ts.set_index(date_col).resample(freq)[sales_col].sum().rename(sales_col).to_frame().reset_index()
    return ts

if not df['Release_Year'].isna().all():
    ts = aggregate_timeseries(df, freq='YE')
    fig = px.line(ts, x='Release_Year', y='Global_Sales', title='Global Sales Over Time (Yearly)')
    # annotate top peaks
    peaks = ts.sort_values('Global_Sales', ascending=False).head(3)
    for _, r in peaks.iterrows():
        fig.add_annotation(x=r['Release_Year'], y=r['Global_Sales'],
                               text=f"{r['Global_Sales']:.2f}", showarrow=True, arrowhead=2)
    fig.update_layout(width=1000, height=500)
    fig.show()
else:
    display(Markdown("No valid release_date found — cannot produce time series."))
```

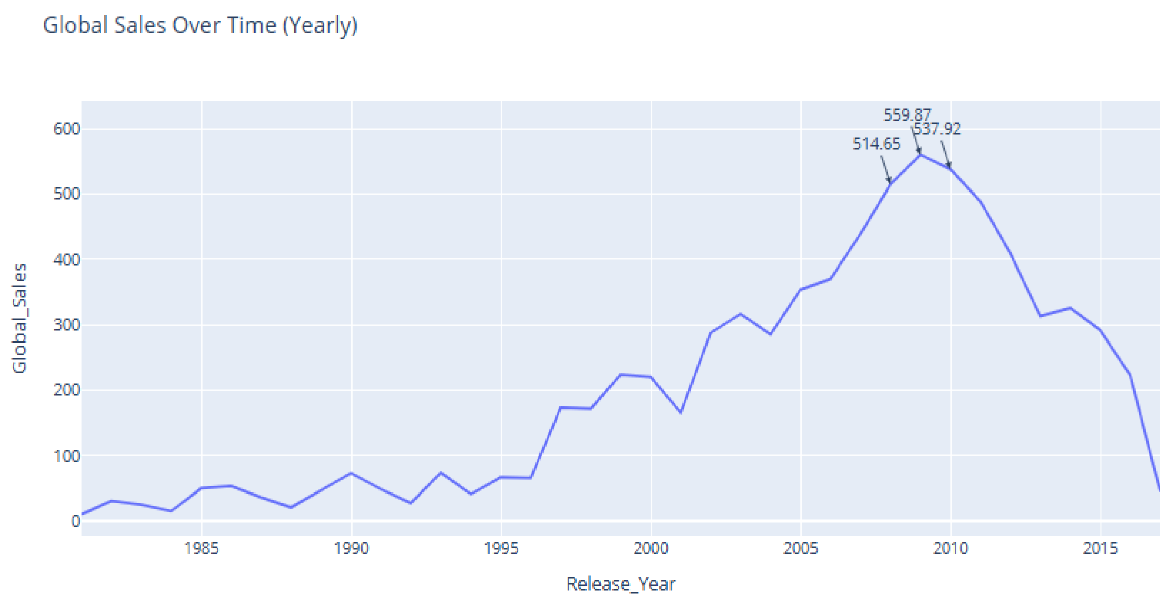


Figure 13: Line chart for Global Sales over time (personal collection).

To understand the main features of the video game sales dataset descriptive statistics analysis is accomplished. The aim was to summarise sales values and finding the clear patterns in the data. Key performance indicators (KPIs) such as total global sales, average sales per title, the most popular genre, and the best-selling platform are calculated as in Figure 8. These measures give a primary picture of how the industry performed over the dataset period (Rumsey, 2016). Central tendency, such as the mean, median and the measures of spread such as variance and standard deviation are calculated to describe sales numbers (Scott, 2020). Visual tools have also been used to support the analysis. A pie chart in Figure 9 shows how sales are divided across regions, while bar charts highlighted the top ten genres, platforms, and publishers in Figure 10, 11 and 12 accordingly. A line chart as in Figure 13 presented sales trends over time. These descriptive methods help to communicate patterns and provide the base for deeper statistical testing (Irizarry, 2019)

## Observing Data Distribution and Outlier Detection

```
# Raw Global Sales
plt.figure(figsize=(16, 4))
sns.histplot(df['Global_Sales'], bins=60, color='green', stat='density')  # histogram
sns.kdeplot(df['Global_Sales'], color='darkred', linewidth=2)  # KDE line
plt.title('Distribution of Global Sales (raw)')
plt.xlabel('Global Sales')
plt.ylabel('Density')
plt.show()
```

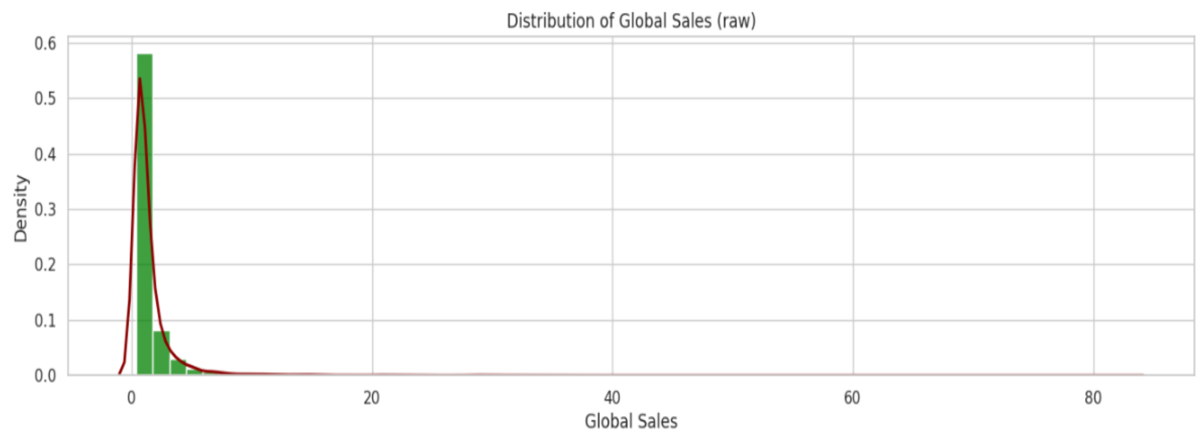


Figure 14: Histogram of Global Sales with KDE (personal collection).

```
# Log-transformed Global Sales
df['Log_Global_Sales'] = np.log1p(df['Global_Sales'])

plt.figure(figsize=(16, 4))
sns.histplot(df['Log_Global_Sales'], bins=60, color='green', stat='density')
sns.kdeplot(df['Log_Global_Sales'], color='darkred', linewidth=2)
plt.title('Distribution of log(1 + Global_Sales)')
plt.xlabel('log(1 + Global Sales)')
plt.ylabel('Density')
plt.show()
```

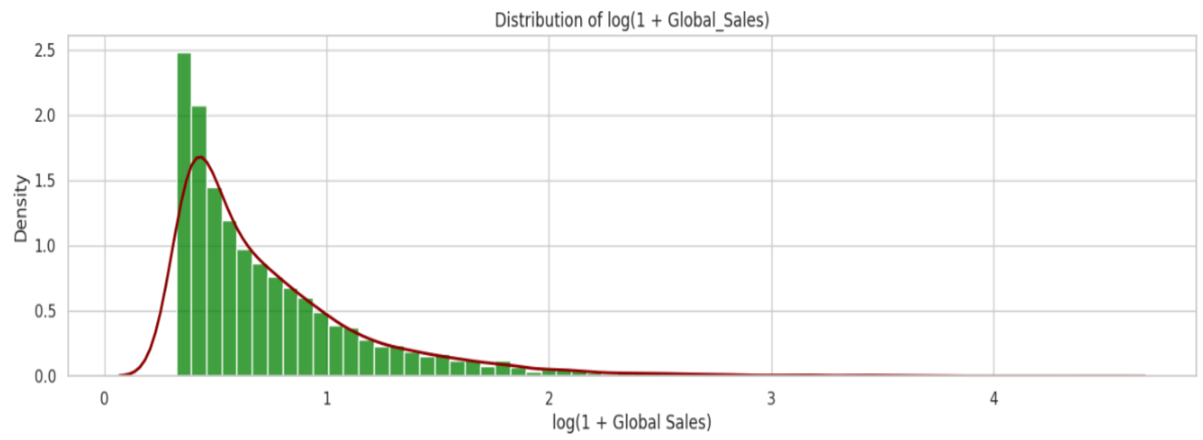


Figure 15: Histogram of log (1 + Global_Sales) (personal collection).

```
# Boxplot of Global Sales by Genre
if 'Genre' in df.columns and 'Global_Sales' in df.columns:
    plt.figure(figsize=(12,6))
    sns.boxplot(x='Genre', y='Global_Sales', data=df)
    plt.xticks(rotation=90)
    plt.yscale('symlog')
    plt.title('Global Sales by Genre (symlog scale)')
    plt.show()
```

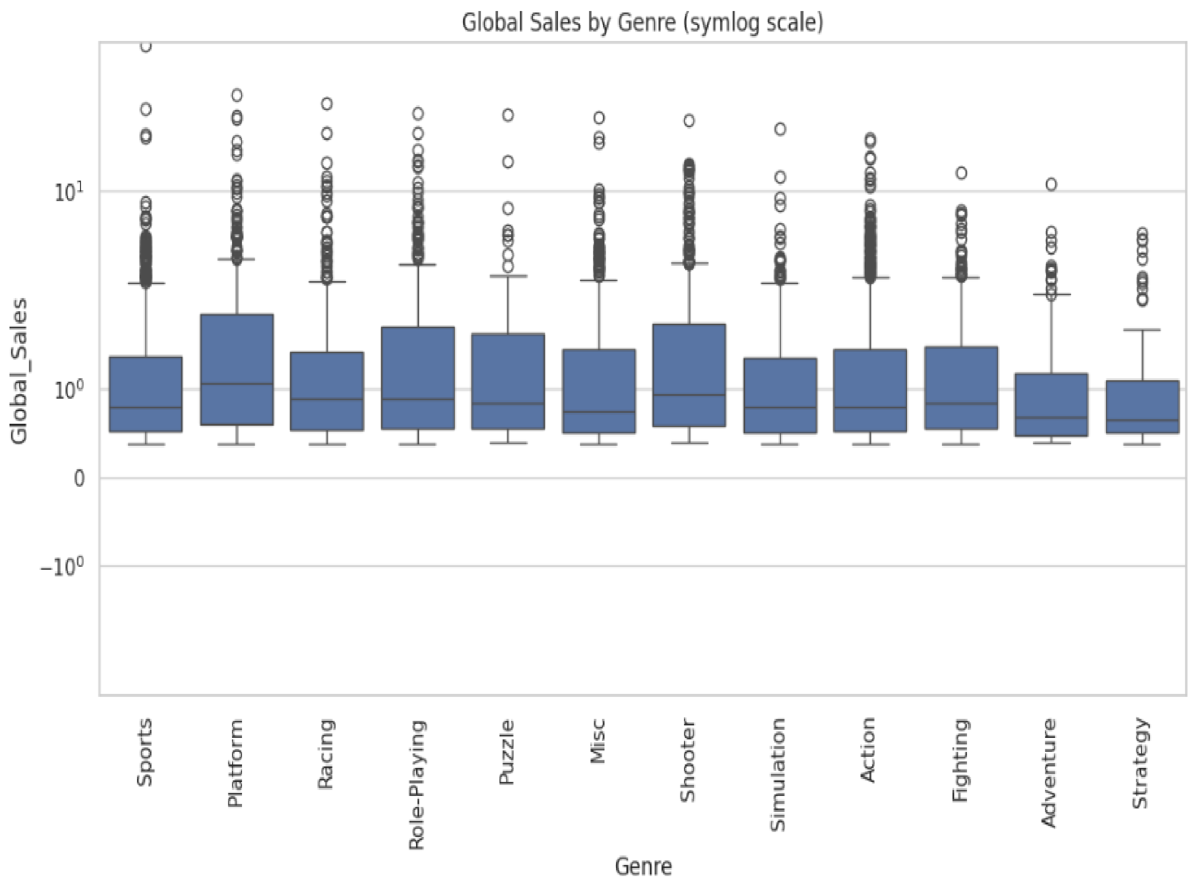
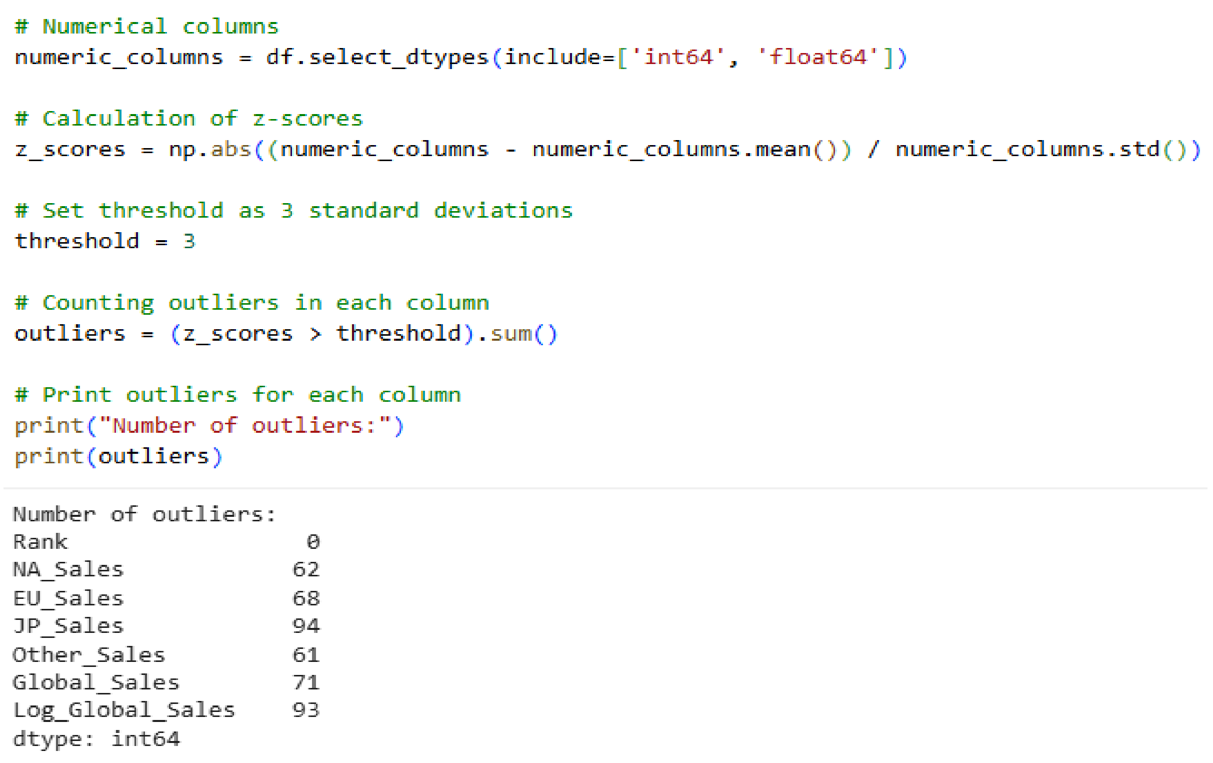


Figure 16: Boxplot of Global Sales by Genres (personal collection).

Using histograms with a density line the distribution of global sales is checked at first as in Figure 14. The results shows that sales are highly skewed, with many small values in the left and a few very large values in the right. As can be seen in Figure 15, a log transformation was used for reducing this effect and improve the visibility of the data distribution. This process made the distribution toward more normal distribution (Changyong et al., 2014). Boxplots are then created as in Figure 16 to compare sales by genre to help to see differences between groups and detect unusual data points (GeeksforGeeks, 2025). Outliers are also calculated using the z-score method as in the following Figure 17. This process measures how far each value is from the mean, outside of 3 standard deviations (Miller, 2017).



```
# Numerical columns
numeric_columns = df.select_dtypes(include=['int64', 'float64'])

# Calculation of z-scores
z_scores = np.abs((numeric_columns - numeric_columns.mean()) / numeric_columns.std())

# Set threshold as 3 standard deviations
threshold = 3

# Counting outliers in each column
outliers = (z_scores > threshold).sum()

# Print outliers for each column
print("Number of outliers:")
print(outliers)
```

```
Number of outliers:
Rank                 0
NA_Sales            62
EU_Sales            68
JP_Sales            94
Other_Sales         61
Global_Sales        71
Log_Global_Sales    93
dtype: int64
```

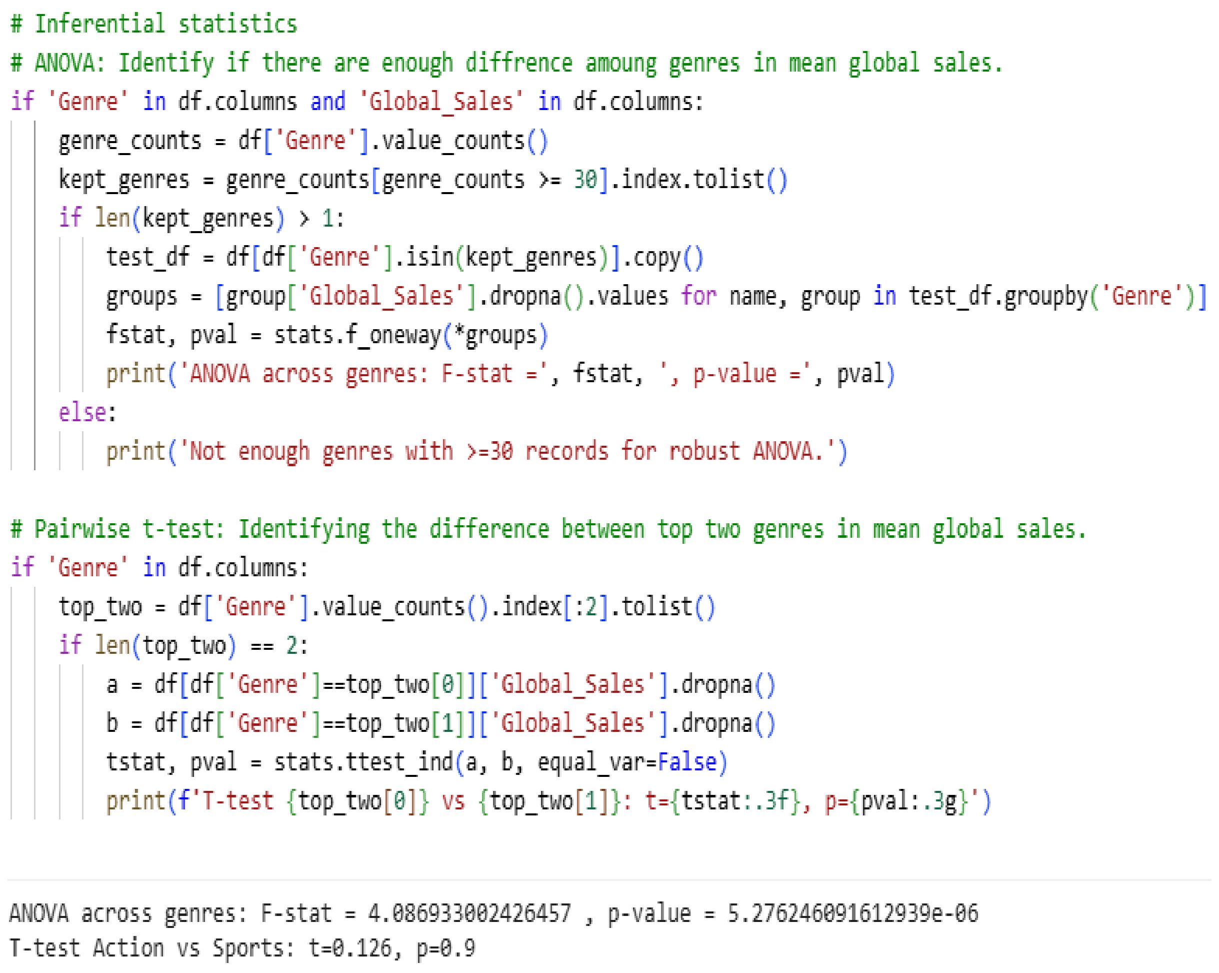
Figure 17: Calculated outliers in each numerical column (personal collection).

Figures 14 and 15 show that there are outliers with high global sales that appear on the right. Figures 16 and 17 show that there are large data points outside the interquartile range. Although there are a large number of outliers in the dataset, they are still the actual sales figures. So, we decided not to remove them for the purpose of accurate findings in the initial analysis.

## Statistical Inferential Analysis with Hypothesis Testing

Inferential statistics are utilised in this data analysis project to test if the differences found in the video game sales are real or just random. A one way ANOVA test has been used to check if average global sales are different between game genres. The F-statistic and p-value are calculated for the ANOVA. If the p-value is smaller than 0.05, the null hypothesis ($H_0$) will be rejected, meaning at least one genre had higher sales which will stablish the alternative hypotheses ($H_1$).

To compare the mean sales of the top two genres a t-test has been used. For the t-test result a t-statistic and p-value are calculated which helps to decide if the two genres has statistically different sales (Field, 2024). These hypothesis tests are very important for the world scenario of data analysis because they move beyond description. By giving stronger evidence for decision making they show whether observed patterns are likely to apply to the wider population of games (Rumsey, 2016).



```
# Inferential statistics
# ANOVA: Identify if there are enough diffrence amoung genres in mean global sales.
if 'Genre' in df.columns and 'Global_Sales' in df.columns:
    genre_counts = df['Genre'].value_counts()
    kept_genres = genre_counts[genre_counts >= 30].index.tolist()
    if len(kept_genres) > 1:
        test_df = df[df['Genre'].isin(kept_genres)].copy()
        groups = [group['Global_Sales'].dropna().values for name, group in test_df.groupby('Genre')]
        fstat, pval = stats.f_oneway(*groups)
        print('ANOVA across genres: F-stat =', fstat, ', p-value =', pval)
    else:
        print('Not enough genres with >=30 records for robust ANOVA.')

# Pairwise t-test: Identifying the difference between top two genres in mean global sales.
if 'Genre' in df.columns:
    top_two = df['Genre'].value_counts().index[:2].tolist()
    if len(top_two) == 2:
        a = df[df['Genre']==top_two[0]]['Global_Sales'].dropna()
        b = df[df['Genre']==top_two[1]]['Global_Sales'].dropna()
        tstat, pval = stats.ttest_ind(a, b, equal_var=False)
        print(f'T-test {top_two[0]} vs {top_two[1]}: t={tstat:.3f}, p={pval:.3g}')
```

```
ANOVA across genres: F-stat = 4.086933002426457 , p-value = 5.276246091612939e-06
T-test Action vs Sports: t=0.126, p=0.9
```

Figure 18: ANOVA across Genres and T-test statistic between top two Genres (personal collection).

From Figure 18 we can see the outcome of ANOVA across genres, the F-statistic is 4.087, which is indicating the differences in mean sales among genres. The p-value $5.276 \times 10^{-6}$ is smaller than the significance level 0.05. So it provides enough evidence to reject the null hypothesis.

The outcome of the t-test between the top two genres, t-statistic 0.126, is close to zero, indicating that the average sales difference between the Action and Sports genres is negligible. The p-value of $0.9 > 0.05$ is extremely high, which provides sufficient evidence to accept the null hypothesis (Kim, 2015).

Finally, for the ANOVA we found strong statistical evidence to say that the video game genres have different mean global sales. Some genres perform better than others in terms of global sales.

For the t-test we found that there is no statistically significant difference between the mean global sales of Action and Sports genres which means they perform almost equally in average global sales.

## Correlation and Trend Analysis

```
# Correlation heatmap for numeric columns
selected_columns = ['NA_Sales', 'EU_Sales', 'JP_Sales', 'Other_Sales', 'Global_Sales']

# Select numerical columns for correlation analysis
selected_columns = ['Release_Year', 'NA_Sales', 'EU_Sales', 'JP_Sales', 'Other_Sales', 'Global_Sales']
numerical_columns = df[selected_columns]

# Produce correlation matrix with pandas
correlation_matrix = numerical_columns.corr()

# Print correlation matrix with column names aligned
print("Correlation Matrix:")
print(correlation_matrix.to_string())
```

```
Correlation Matrix:
              Release_Year  NA_Sales  EU_Sales  JP_Sales  Other_Sales  Global_Sales
Release_Year      1.000000 -0.068298  0.085581 -0.201292     0.126351     -0.035596
NA_Sales         -0.068298  1.000000  0.720266  0.422963     0.570725      0.932261
EU_Sales          0.085581  0.720266  1.000000  0.397293     0.669402      0.879184
JP_Sales         -0.201292  0.422963  0.397293  1.000000     0.237458      0.596586
Other_Sales       0.126351  0.570725  0.669402  0.237458     1.000000      0.695579
Global_Sales     -0.035596  0.932261  0.879184  0.596586     0.695579      1.000000
```

Figure 19: Correlation among numerical variables (personal collection).

```
plt.figure(figsize=(10,8))
sns.heatmap(df[selected_columns].corr(), annot=True, fmt=".2f", cmap="coolwarm", center=0)
plt.title('Numeric correlation matrix')
plt.show()
```

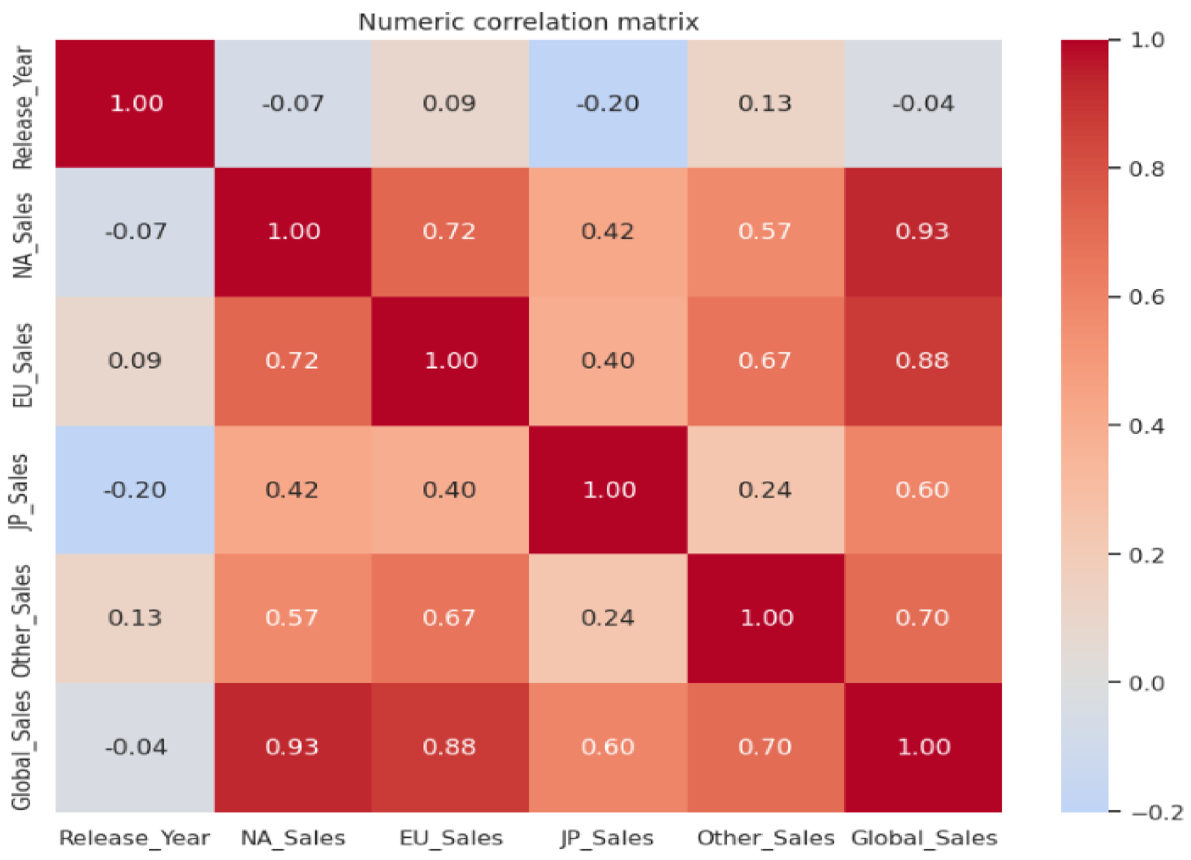


Figure 20: Heatmap showing correlation (personal collection).

## Polynomial Trend Line for Yearly Global Sales

```
# Yearly Global Sales Polynomial Trend
# Group the data by year and calculate total Global Sales
yearly_sales = df.groupby('Release_Year')['Global_Sales'].sum().reset_index()

# Ensure Release_Year is in datetime format for plotting
yearly_sales['Date'] = pd.to_datetime(yearly_sales['Release_Year'].astype(str) + '-01-01')

# Fit a polynomial trend line (degree 3)
coefficients = np.polyfit(np.arange(len(yearly_sales)), yearly_sales['Global_Sales'], 3)
poly_function = np.poly1d(coefficients)

# Plot the scatter plot with trend line
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Date', y='Global_Sales', data=yearly_sales, color='skyblue')
plt.plot(yearly_sales['Date'], poly_function(np.arange(len(yearly_sales))),
         color='red', linestyle='--', label='Polynomial Trend Line')
plt.title('Yearly Global Sales with Polynomial Trend Line')
plt.xlabel('Year')
plt.ylabel('Global Sales')
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.legend()
plt.grid(True)
plt.show()
```

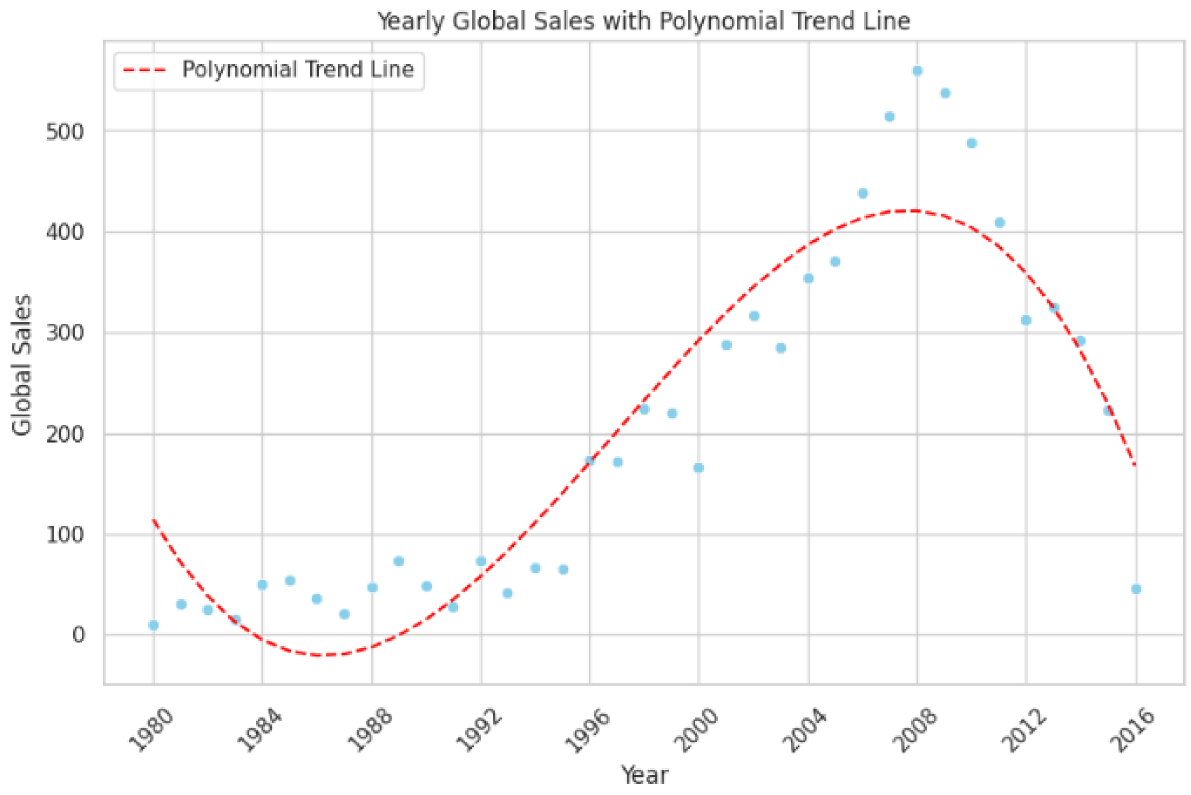


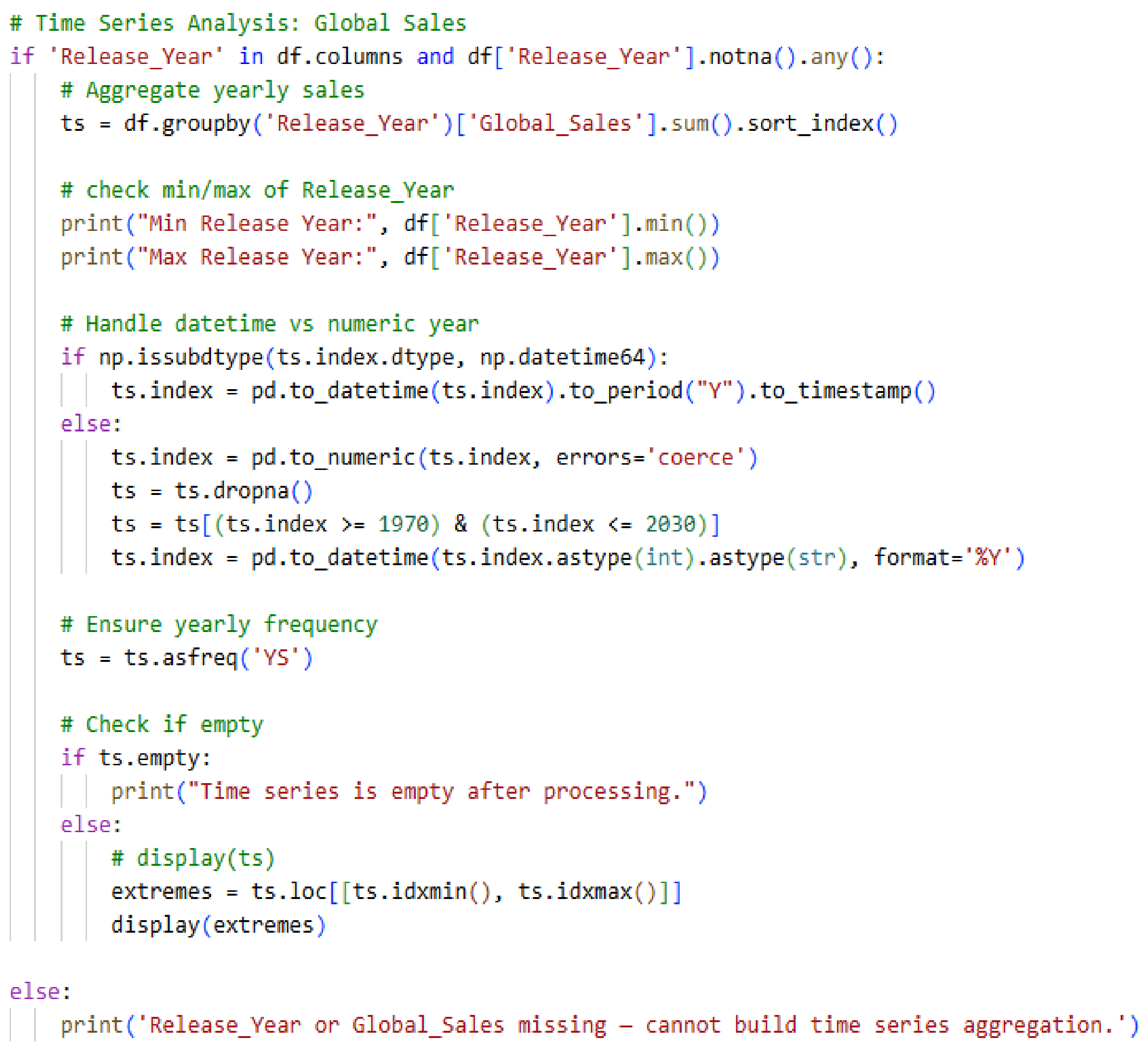
Figure 21: Scatterplot of yearly global sales over time with polynomial trend line (personal collection).

To discover the relationship among sales variables including release year the correlation analysis is used. It also checks how they move together. A correlation matrix is observed to see if regional sales such as North America and Europe are linked with global sales as in

Figure 19. The heatmap on Figure 20 shows a strong positive correlation which mean higher regional sales often lead to higher global sales (Sedkaoui, 2018). After that a polynomial regression is applied to yearly sales data to find long term patterns. as seen in Figure 21, A trend line showing the relationship between global sales over time demonstrates a strong connection between these two variables, which can be represented by a polynomial line. These results provide useful insights for prediction and planning for the future business (Field, 2013; Irizarry, 2019).

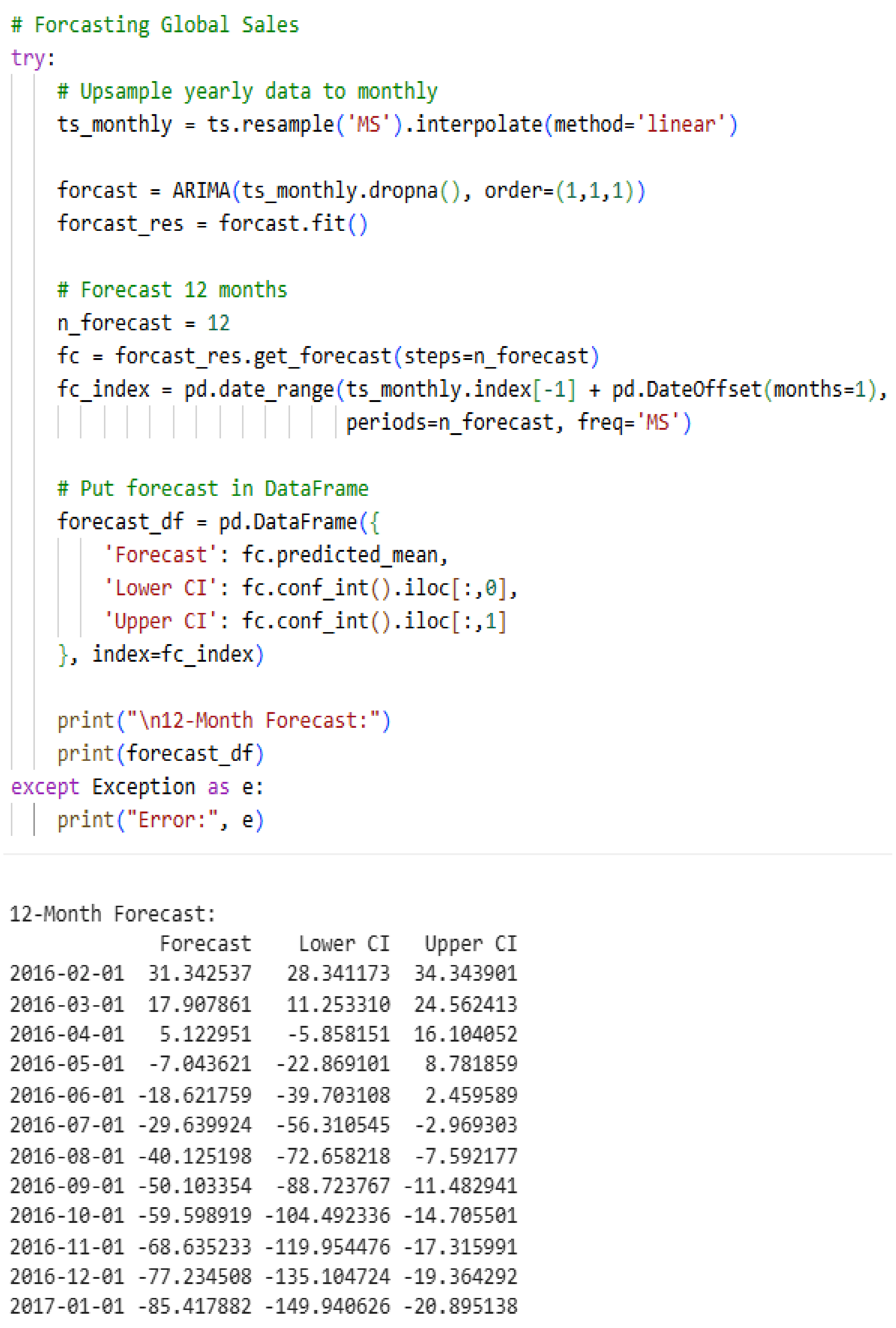
## Forecasting Future Video Game Sales

To estimate future video game sales, a time series forecasting approach has been used based on the yearly global sales data. The dataset covered sales from 1980 to 2016. Once the line chart for the global sales until 2016 is created, the yearly data has been converted into monthly frequency to produce chart for monthly sales forecasting.



```
# Time Series Analysis: Global Sales
if 'Release_Year' in df.columns and df['Release_Year'].notna().any():
    # Aggregate yearly sales
    ts = df.groupby('Release_Year')['Global_Sales'].sum().sort_index()

    # check min/max of Release_Year
    print("Min Release Year:", df['Release_Year'].min())
    print("Max Release Year:", df['Release_Year'].max())

    # Handle datetime vs numeric year
    if np.issubdtype(ts.index.dtype, np.datetime64):
        ts.index = pd.to_datetime(ts.index).to_period("Y").to_timestamp()
    else:
        ts.index = pd.to_numeric(ts.index, errors='coerce')
        ts = ts.dropna()
        ts = ts[(ts.index >= 1970) & (ts.index <= 2030)]
        ts.index = pd.to_datetime(ts.index.astype(int).astype(str), format='%Y')

    # Ensure yearly frequency
    ts = ts.asfreq('YS')

    # Check if empty
    if ts.empty:
        print("Time series is empty after processing.")
    else:
        # display(ts)
        extremes = ts.loc[[ts.idxmin(), ts.idxmax()]]
        display(extremes)

else:
    print('Release_Year or Global_Sales missing – cannot build time series aggregation.')
```

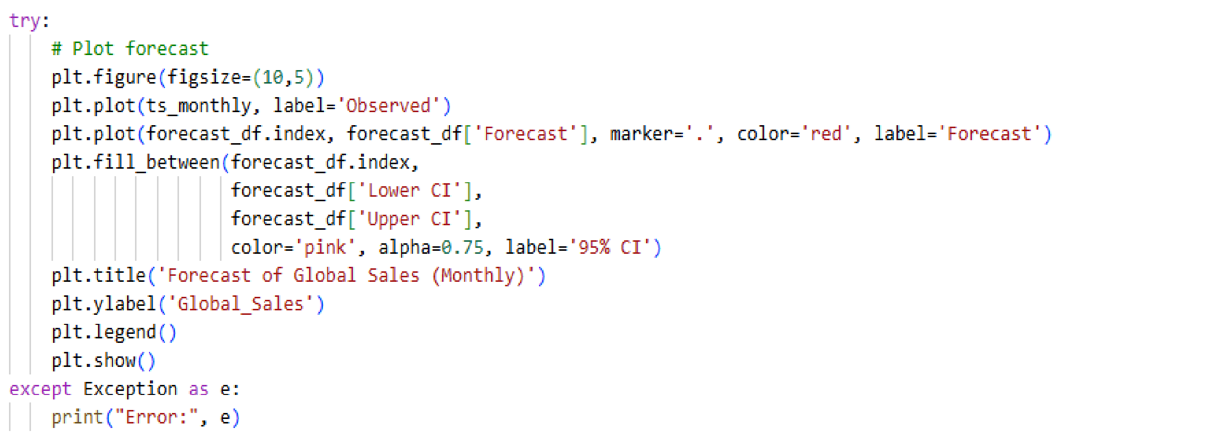
Figure 22: Code for time series analysis (personal collection).



```
# Forcasting Global Sales
try:
    # Upsample yearly data to monthly
    ts_monthly = ts.resample('MS').interpolate(method='linear')

    forcast = ARIMA(ts_monthly.dropna(), order=(1,1,1))
    forcast_res = forcast.fit()

    # Forecast 12 months
    n_forecast = 12
    fc = forcast_res.get_forecast(steps=n_forecast)
    fc_index = pd.date_range(ts_monthly.index[-1] + pd.DateOffset(months=1),
                             periods=n_forecast, freq='MS')

    # Put forecast in DataFrame
    forecast_df = pd.DataFrame({
        'Forecast': fc.predicted_mean,
        'Lower CI': fc.conf_int().iloc[:,0],
        'Upper CI': fc.conf_int().iloc[:,1]
    }, index=fc_index)

    print("\n12-Month Forecast:")
    print(forecast_df)
except Exception as e:
    print("Error:", e)
```

```
12-Month Forecast:
             Forecast    Lower CI   Upper CI
2016-02-01  31.342537   28.341173  34.343901
2016-03-01  17.907861   11.253310  24.562413
2016-04-01   5.122951   -5.858151  16.104052
2016-05-01  -7.043621  -22.869101   8.781859
2016-06-01 -18.621759  -39.703108   2.459589
2016-07-01 -29.639924  -56.310545  -2.969303
2016-08-01 -40.125198  -72.658218  -7.592177
2016-09-01 -50.103354  -88.723767 -11.482941
2016-10-01 -59.598919 -104.492336 -14.705501
2016-11-01 -68.635233 -119.954476 -17.315991
2016-12-01 -77.234508 -135.104724 -19.364292
2017-01-01 -85.417882 -149.940626 -20.895138
```

Figure 23: Global sales forecast with the lower and upper confidence interval (personal collection).



```
try:
    # Plot forecast
    plt.figure(figsize=(10,5))
    plt.plot(ts_monthly, label='Observed')
    plt.plot(forecast_df.index, forecast_df['Forecast'], marker='.', color='red', label='Forecast')
    plt.fill_between(forecast_df.index,
                     forecast_df['Lower CI'],
                     forecast_df['Upper CI'],
                     color='pink', alpha=0.75, label='95% CI')
    plt.title('Forecast of Global Sales (Monthly)')
    plt.ylabel('Global_Sales')
    plt.legend()
    plt.show()
except Exception as e:
    print("Error:", e)
```

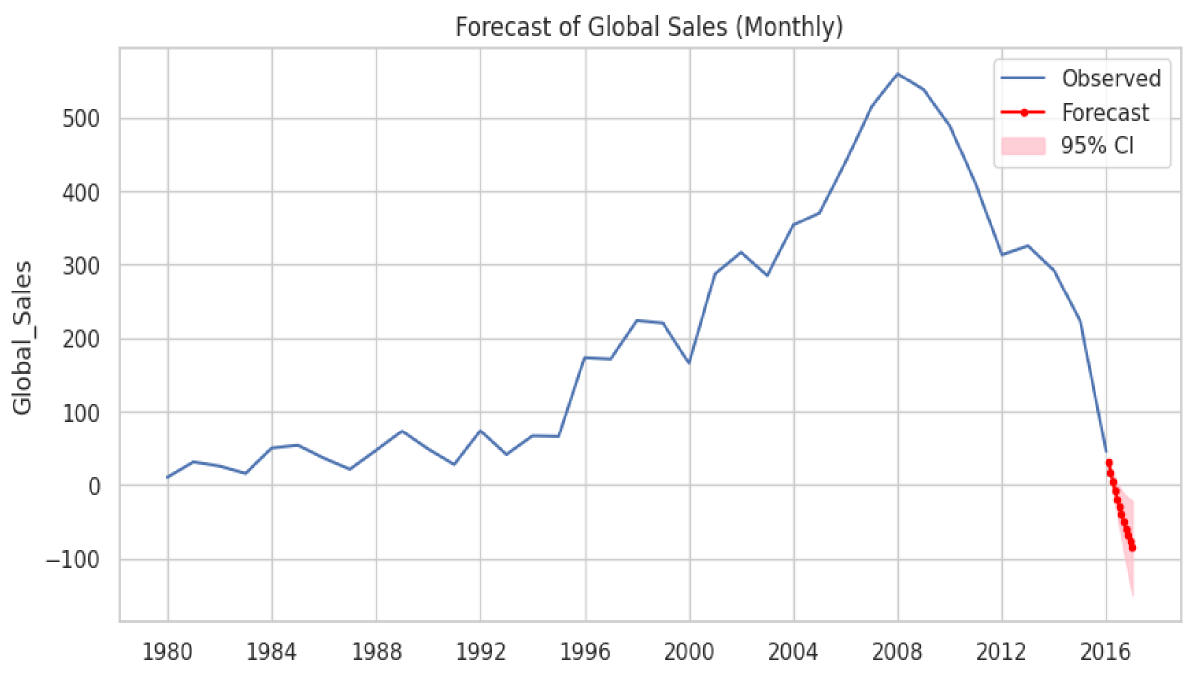


Figure 24: Global sales time series with forecast (personal collection).

Then twelve months of predicted sales values including upper and lower confidence limits are calculated to show the likely range of outcomes (Hyndman and Athanasopoulos, 2018). As can be seen in Figure 24, the results suggested a small fall in sales after the 2008 peak, followed by a steady trend in later years. The line chart of forecast displayed both the observed and predicted values with shaded confidence areas. Clear insights about potential market changes are provided by this chart and supported better planning.

# Report on Exploratory Analysis of Video Game Sales

Prepared for: Analysist Manager

Author: STU98937

Date: 13/10/2025

This report presents a critical review of the exploratory data analysis (EDA) performed on the video game sales dataset of International Media Limited. This work included data loading, initial observations, cleaning and preparing the dataset, conducting descriptive and inferential statistics, identifying anomalies, applying correlation and forecasting methods, and also interpreting the results with reference to visual outputs. Understanding global sales trends, the role of genres and publishers in driving performance, and the potential use of forecasting techniques to predict future sales was the main focus.

## Summary of Approach & Key Results Interpretation

**Data loading & type checking:**

The basis of this analysis was to load the dataset using Python. The Pandas library was used to handle the given CSV file and provide a structured dataframe. This allowed for easy inspection of rows, columns, and data types (VanderPlas, 2016). The presence of inconsistent values in the year column and missing entries for the publisher was an initial challenge. The dataset needed to be cleaned before the analysis could be confirmed by identifying inconsistencies using the .info(), .duplicated(), and .isnull() commands.

**Data exploration:**

The dataset contains regional sales values such as NA_Sales, EU_Sales, JP_Sales, and Other_Sales and also a Global_Sales field. Initial inspection revealed discrepancies where years were not formatted correctly as dates and several columns had missing values that required recalculation. The importance of not assuming the accuracy of the supplied dataset is highlighted by this process.

**Preprocessing & integrity improvement:**

Since preprocessing was one of the most important parts of this project, several steps were taken as follows:

To ensure consistency in the dataset, column names were standardised.

Null publisher, regional sales and global sales values are imputed using proper technic. Although these imputations risked reducing variance, it allowed the dataset to remain complete for analysis (McKinney, 2018).

A clean Release_Year variable was created from the Year column to enable time series analysis.

Removing the 69 rows containing null values and confirming that there were no duplicate rows ensured that the pattern observations were not driven by repeated or empty data.

A more reliable global_sales column was established by recalculating global sales with summing regional figures.

This process not only improved the integrity of the dataset but also demonstrated the iterative nature of the cleanup. Figure 7 of the report illustrates the consistent dataset structure and also shows the corrected Release_Year and consistent sales figures.

**Descriptive EDA with KPI extraction:**

Descriptive statistical analysis provided the very first insight about how the industry performed over the period into the dataset (Rumsey, 2016). Key findings, including key performance indicators (KPIs) are as follows:

Total global sales across the period from 1980 to 2016 exceeded 7407 units of money. The year 2008 is the highest revenue maker, with 559.87 units of money globally. After that period the global sales fell down successively every year until 2016 as seen in Figure 13.

Action and Sports emerged as leading contributors. More than one-third of all sales was contributed by these two genres, as seen in Figure 10.

PlayStation 2 dominated the market, followed by Xbox 360, PlayStation 3 and Wii, as seen in Figure 11.

From Figure 12 we can see that Nintendo, Electronic Arts, and Activision were the most successful publishers by achieving high cumulative sales.

In Figure 9, the pie chart showed that North America and Europe were the largest markets with 49.2% and 28.5% of total global sales respectively, while Japan had a smaller but distinct sales profile with 13%.

**Distribution & outlier detection:**

Exploring the distribution of global sales reveals a strongly skewed pattern as shown in Figure 7. The histograms illustrated with density curves clearly show that very few blockbusters exceeded one million due to the long-tail effect.

To reduce skewness, a log transformation was applied to create a normal-like distribution that better supported the statistical observations (Changyong et al., 2014). Utilising both boxplots and z-score methods, outlier detection was conducted as seen in Figures 16 and 17. Games such as Wii Sports appeared to be high-value outliers, representing genuine market success rather than errors.

**Inferential statistical tests:**

Several inferential methods were applied to understand the data insights beyond description as in Figure 18. A one-way ANOVA tested if global sales are different across genres. The F-statistic indicated significant variation, with a p-value lower than 0.05, rejecting the null hypothesis of equal means (Field, 2024). This finding suggests that in predicting sales performance, genre is a meaningful factor.

A follow-up t-test compared the top two genres, Action and Sports. The very low t value and p=0.9>0.05 indicated the two most popular genres, Action and Sports are not significantly different from each other in terms of average global sales. Figure 10 in the report visualise these group differences.

**Correlation & trend analysis:**

To measure the relationship between regional sales and global sales and to measure the relationship between year of release and global sales, correlation analysis techniques were applied. A correlation heatmap (Figure 20) showed strong positive relationships, especially between North American and global performance. This indicates that North American sales may serve as an early predictor of global success (Sedkaoui, 2018).

A polynomial correlation line with a scatter plot for annual global sales was also created to capture the non-linear trend of rise and fall over the decades (Box & Jenkins, 1976). The line shown in Figure 21 indicates that annual global sales increased steadily from 1980 to 2008 and then began to decline abruptly until 2016.

**Forecasting:**

As shown in Figure 24, the forecast declines after the peak in 2008, with sales stabilising in subsequent years. The total global sales will definitely be negative from July 2016 onwards, as confirmed by the negative upper and lower confidence intervals (Hyndman and Athanasopoulos, 2018), as shown in Figure 23. This means the video game business will be shifted to the new platform, such as e-commerce.

## Limitations & Data Quality Considerations

- Missing data imputation with column averages for regional sales is realistic but can change the variance. If the missingness is significant, multiple imputation or model-based approaches are more acceptable.
- External covariates such as market events, console launches, and marketing campaigns are not included in the dataset.
- Aggregating annual data reduces seasonal resolution and can bias short-term forecasts. The best practice is to model at the highest usable frequency and ensure stationarity before forecasting.

## Recommendations

Based on the analytical results of this project, I recommend:

- Implementing a robust missing data handling strategy such as multiple imputation.
- Creating a regular interactive dashboard with Power BI or Plotly Dash to explore and drill down into trends across publishers, platforms, and genres, along with KPIs and other important metrics.
- To improve forecasting, a rich dataset can include marketing spend, console release dates, major marketing campaigns, price promotions, and platform install base.
- For richer forecasts, we need to ensure higher frequencies, such as monthly or weekly data analysis.
- Implementing segment analysis by executing separate models by platform and publisher for more actionable insights.
- Implementing multiple Machine Learning models and applying appropriate model validation through back testing before relying on forecasts for decision making.

## Reflection on Skills & Learning

1) **Programming Concepts:** I learnt how to apply core programming principles through structured Python code using functions, libraries, and modular workflows.
2) **Data Manipulation and Analysis:** I successfully cleaned, transformed, and analysed data with Pandas, NumPy, and other statistical libraries.
3) **Critical Evaluation:** I interpreted results beyond numbers, considering limitations, outliers, and alternative approaches.
4) **Digital Literacy:** I evaluated digital tools and explored advanced methods such as EDA and correlation and reflected on future possibilities, which increased my digital capabilities.

The project demonstrated independence of thought, strong analytical reasoning, and professional communication skills.

## Environment and Setup

The data analysis was performed using the online version of Google Colab (n.d.). It provides a cloud-based environment for executing Python code and ensures easy package management and GPU acceleration. This setup allowed for interactive analysis, visualisation, and reproducible research without local installation issues.

The environment used Python 3.12.11 as the base interpreter.

The following library versions were utilised:

pandas 2.2.2, numpy 2.0.2, matplotlib 3.10.0, seaborn 0.13.2, scipy 1.16.2, statsmodels 0.14.5, scikit-learn 1.6.1, plotly 5.24.1, and IPython 7.34.0.

## Conclusion

The study found a strong link between regional and global sales and clear differences in average sales by genre. Sales peaked in 2008, after which the business stabilised and then declined for several consecutive years until 2016. The findings suggest a replacement of video game platforms with e-commerce in the upcoming years.

The analysis not only answered the main questions but also identified future improvements, including high-frequency data, multivariate forecasting, and robust adaptation techniques. The reflection demonstrated significant engagement with programming tools and statistical methods which met all requirements.

## References

ApX Machine Learning. (n.d.). *Strategies for Missing Data: Imputation vs Deletion*. Available at: https://apxml.com/courses/intro-eda-course/chapter-2-data-loading-inspection-cleaning/missing-data-strategies [Accessed: 9 October 2025].

Box, G. and Jenkins, G.M., 1976. Analysis: Forecasting and Control. *San francisco*.

Changyong, F.E.N.G., Hongyue, W.A.N.G., Naiji, L.U., Tian, C.H.E.N., Hua, H.E., Ying, L.U. and Xin, M.T., 2014. Log-transformation and its implications for data analysis. *Shanghai archives of psychiatry*, *26*(2), p.105.

Field, A., 2024. *Discovering statistics using IBM SPSS statistics*. Sage publications limited.

GeeksforGeeks. (2025, July 12). *Box Plot in Python using Matplotlib* [online]. Available from: https://www.geeksforgeeks.org/box-plot-in-python-using-matplotlib/ [Accessed 10 09 2025].

Google Colab. (n.d.). *Colaboratory* [online]. Retrieved from https://colab.research.google.com/ [Accessed 2024].

Han, J., Pei, J. and Tong, H., 2022. *Data mining: concepts and techniques*. Morgan kaufmann.

Hyndman, R.J. and Athanasopoulos, G., 2018. *Forecasting: principles and practice*. OTexts.

Irizarry, R. A., (2019). *Statistics and Prediction Algorithms Through Case Studies* [online]. CRC Press. Available at: https://rafalab.dfci.harvard.edu/dsbook-part-2/ [Accessed 27 03 2024].

Kim, T.K., (2015). *T test as a parametric statistic. Korean journal of anesthesiology,* 68(6), p.540 [online]. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4667138/ [Accessed 24 03 2023].

McKinney, W., 2012. *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython*. " O'Reilly Media, Inc.".

Miller. D. J. (2017). *Big Data Visualization: Bring Scalability and Dynamics to Your Big Data Visualization* [online]. Packt Publishing. Available from: https://links.arden.ac.uk/bvw7qn [Accessed 12 04 2024].

Rumsey, D., (2016). *Statistics For Dummies* [online]. 2nd edition. Wiley. Available from: https://links.arden.ac.uk/hdgx5v [Accessed 03 04 2023].

Scott, D.W., 2020. *Statistics: a concise mathematical introduction for students, scientists, and engineers*. John Wiley & Sons.

Sedkaoui. S., (2018). *Data Analytics and Big Data* [online]. Wiley. Available from: https://links.arden.ac.uk/m8qvbk [Accessed 08 03 2024].

VanderPlas, J., 2016. *Python data science handbook: Essential tools for working with data*. " O'Reilly Media, Inc.".