

Analyzing Video Game Sales Data using PySpark and Tableau



Hamzah Asghar Farooqi (13191038)

farooqih@uni.coventry.ac.uk

Word Count

Dataset Analysis and Visualization Using Big Data Programs

Module: 7086CEM_Big Data Analytics and Data Visualisation

Faculty of Engineering Environment and Computing

MSc Data Science And Computational Intelligence

Coventry University

Abstract:

This course work explores the comprehensive analysis of video game sales data using the combined power of PySpark and Tableau. Through a structured approach, the project involves data loading, preprocessing, exploratory data analysis, and model evaluation using PySpark. Techniques such as linear regression, logistic regression, and random forest classification are applied to derive meaningful insights. The integration of Tableau facilitates interactive data visualization, offering a dynamic perspective on sales trends, genres, and platforms. The project's objective is to provide a robust framework for understanding the video game industry's dynamics, enabling data-driven decision-making and informed recommendations.

Table of Content

Introduction	- 3 -
Background And Related Work	- 4 -
Dataset Section: Video Game Sales Dataset	- 5 -
Methodology And Experiments	- 6 -
Software and Tools Used:	- 14 -
Social and Ethical Consideration	- 15 -
Result And Discussion	- 15 -
Conclusion And Future Work	- 15 -
Bibliography and References:	- 15 -
Appendix I: Screenshots of Tasks	- 16 -
Appendix II: Code in PySpark	- 37 -

Table of Figures

Figure 1 : Here We had done PySpark installation and importing the Libraries	- 7 -
Figure 2 : Here We created the Spark Session and Loading the Dataset	- 7 -
Figure 3 : Here we present Data After Cleaning	- 8 -
Figure 4 : Here we Calculate the Descriptive Statistics, Mean, Median, Standard Deviation	- 8 -
Figure 5 : Here we had Done some Data Exploration	- 8 -
Figure 6 : Question 1 data exploration and analysis tasks	- 9 -
Figure 7 : This is Histogram of all region of sales: the Result shows that the Sales in Japan are highest in all region.	- 9 -
Figure 8 : Here we execute the all region sales over year by using Scatter plot.	- 9 -
Figure 9 : Here we use Box Plot to show Global Sales vs Genre Pictorially	- 10 -
Figure 10 : Here we Implement code for Bar Plot bu using Matplotlib of All region by Platform	- 10 -
Figure 11 : Correlation Matrix Heatmap	- 11 -
Figure 12 : the linear regression analysis provides insights into the relationships between the sales attributes and how they contribute to the overall global sales.	- 12 -
Figure 13 : the logistic regression analysis provides insights into the attributes that contribute to a game becoming a top seller in the video game industry.	- 12 -
Figure 14 : Random Forest Classifier	- 12 -
Figure 15 : Tableau Visualization screen 1	- 13 -
Figure 16 : Dashboard Visualization screen 1	- 14 -

Table of Table

Table 1 : Dataset Records and Attributes	- 6 -
Table 2 : Dataset Attributes details	- 6 -

Introduction

The project "Analyzing Video Game Sales Data using PySpark and Tableau" seeks to offer in-depth understanding of video game sales dynamics through a blend of data analysis and interactive visualization. It utilizes the diverse "Video Game Sales" dataset, encompassing attributes like game names, platforms, genres, publishers, and sales figures. Through a hybrid approach, the project aims to reveal trends, patterns, and influential factors driving game sales. In the age of digital entertainment, video games have become a global industry that captivates millions of people with its diverse range of products and services. Video games have complex trends, different genres, different platforms, and different customer preferences that influence the entire industry landscape and go beyond mere enjoyment. Our course work begins with a deep dive into video game sales data. Using PySpark's and Tableau's powerful features, we analyze a dataset with more than 16,000 records each of which represent a video game with over 100,000 copies sold. It is essential for all industry players to understand what factors influence video game sales. Educated decisions are the foundation of game development, marketing, and business strategies from developers, publishers, marketers, and strategists. A deep dive into the sales data reveals unseen trends, creates strong connections, and provides insightful conclusions that paint a full picture of the video games market. With this project, we will be able to better understand the industry's pulse, identify trends, and develop effective tactics.

The dataset that serves as the basis of our investigation was obtained through a scrape of the comprehensive database of video game sales statistics on vgchartz.com. It includes significant elements such as game names, platforms for their release, genres, publishers, and sales data from various geographies. The dataset includes fields for North America (NA) Sales, Europe (EU) Sales, Japan (JP) Sales, Other Sales, and Global Sales, giving it a comprehensive picture of sales success. It is feasible to derive insights that illuminate the dynamics of the industry thanks to the dataset's breadth and diversity. The video game sales dataset is the subject of a detailed investigation in our course work. To identify patterns and correlations between variables, we use Random Forest Classifier, Logistic Regression, and Linear Regression approaches. Through this analytical process, we are able to forecast worldwide sales patterns, pinpoint key variables, and group the best-selling games according to genres and characteristics.

We utilize PySpark, a potent data processing framework, in the Google Colab environment to carry out this investigation. PySpark makes it easier to import data, do preprocessing, and conduct advanced analysis, which promotes a thorough grasp of the dataset. In order to visualize data, we use Tableau, a flexible application that turns raw data into engaging dashboards. We seek to offer a thorough and perceptive examination of video game sales statistics by smoothly combining these software technologies. Our course work essentially serves as a testament to the nexus between data science and the gaming sector, solving the mystery of video game sales through thorough analysis and engaging visualization.

Background And Related Work

The motivations for the sales and success of video games have been the subject of several studies and analysis. The effect of game genres, platforms, release years, and regional preferences on sales figures has been studied. Acting, adventure, playing, and sports video games all have different consumer expectations that have been highlighted through genre-based assessments. Similar to this, platform-centric studies have looked at how personal computers, mobile devices, and game consoles affect sales trends. Some of literature review are as follows,

1. *Vertical Integration, Exclusivity, and Game Sales Performance in the US Video Game Industry*

This research investigates the correlation between vertical integration and game performance within the United States video game industry.. Using a detailed dataset spanning 2000 to 2007, including information about developers and mergers, we differentiate between vertically integrated and independently developed games exclusive to a platform. Results show integrated games have higher sales and prices, primarily due to better release strategies and game quality selection. Interestingly, post-release marketing has minimal impact on integrated games' value. Additionally, exclusivity is associated with lower demand and higher pricing, driven by differences in inherent quality and release strategies. This research highlights how vertical integration, release strategies, and game quality influence game performance dynamics.

2. *Technological tying and the intensity of price competition: An empirical analysis of the video game industry*

This study looks at how technical tying affects the 128-bit video game market, with a particular emphasis on how it affects console pricing competition and hardware manufacturers' motivations to tie software to their hardware. Developing technology that is compatible with competing hardware is known as technological tying. When adopting this linkage, integrated enterprises must choose between two important trade-offs: one that increases console popularity & drives up hardware costs, and other driven by connectivity that lowers prices. We discover that technology coupling increases console pricing competitiveness through a counterfactual analysis. Console manufacturers offer discounts on consumer hardware to encourage the purchase of video games, especially the more lucrative ones. The study also shows that, particularly when costs for developing software are low, technical tying is a common approach for hardware producers.

3. *The Impact Of Platform On Global Video Game Sales*

This study looks into the worldwide sales of video games on various platforms between 2006 and 2011. The research aims to determine the important elements impacting sales in light of the domestic video game industry's rapid expansion and emergence as a vital component of interactive home entertainment. The influence of the vertical and horizontal expansion tactics adopted by significant game publishers & developers is discussed in the study. The Kruskal-Wallis test is used to compare eight gaming systems. The results show that the Nintendo DS is the product with the second-highest sales tier worldwide, after the Nintendo Wii. Xbox 360, Sony PlayStation 3, and personal computers (PCs) are included in the third tier. The sixth-generation consoles with the lowest sales tier are the defunct Nintendo GameCube and the Sony PlayStation 2 and Sony PSP.

Data Analysis

Our effort, which builds on earlier work in video game sales analysis, intends to provide a full and in-depth study of the "Video Game Sales" dataset using PySpark and Tableau. While previous research has focused on particular aspects of the industry, our analysis adopts a more thorough approach to identify a broader range of variables driving game sales. Our analysis of the data will centre on the following crucial elements:

- I. **Genre and Platform Analysis:** Our analysis will investigate the influence of game genres and platforms on worldwide sales. By studying sales patterns within various genres and platforms, we intend to unveil consumer preference trends and ascertain how specific combinations influence elevated sales. This exploration complements prior research that explored the interplay between vertical integration, exclusivity, and game performance, as well as the effect of platforms on global video game sales.
- II. **Regional Sales Distribution:** Our study will undertake an extensive examination of regional sales distribution, akin to previous research that studied the effects of exclusivity and regional demand on game sales. We will analyze sales patterns across different regions (North America, Europe, Japan, and others) to understand how regional preferences, cultural elements, and market dynamics contribute to variations in sales.
- III. **Regression Analysis:** By utilizing PySpark's regression analysis functionalities, our project aims to construct predictive models that unveil the connections between different attributes (such as genre, platform, and regional sales) and the global sales performance of a game. This approach is in line with prior research that employed regression analysis to explore the dynamics of price competition intensity and game performance.
- IV. **Interactive Data Visualization:** Similar to the approach taken in previous research, our project will utilize counterfactual analysis and statistical tests. However, we will enhance our analysis by harnessing the power of Tableau to generate interactive visualizations. These visualizations, encompassing charts, scatter plots, bar plots, and box plots, will provide a multi-dimensional exploration of the dataset, showcasing trends, anomalies, and correlations in a visually engaging manner.
- V. **Model Evaluation:** Continuing in the vein of previous research on game performance analysis, we will assess the accuracy of our regression models using established metrics like RMSE for linear regression and area under ROC curve for logistic regression and random forest classifier. This evaluation will enable us to gauge the predictive prowess of our models effectively.

Dataset Section: Video Game Sales Dataset

Dataset Link: <https://www.kaggle.com/datasets/gregorut/videogamesales>

The central focus of our analysis rests on the "Video Game Sales" dataset, a rich source of information encompassing video game sales data across diverse regions, platforms, genres, and publishers. This section outlines the dataset's attributes, structure, and the steps taken to ensure its technical integrity and appropriateness for our analytical pursuits.

Data-set Details:

Table 1: Dataset Records and Attributes

Total Recrds/ Rows	Total Attributes/ Column
16598	11

Table 2: Dataset Attributes details

Attributes Details	Explanation
Rank	Ranking of Overall sale
Name	The name of Video game
Platform	The Platform of game Release
Year	The Year of game release
Genre	The genre of the game
Publisher	The publisher of the game
NA_ Sales	Sales in North America (In Millions)
EU_ Sales	Sales in Europe (In Millions)
JP_ Sales	Sales in Japan (In Millions)
Other_ Sales	Sales in rest of world (In Millions)
Global_ Sales	Total Worldwide sales.

Dataset Processing - Technical Quality:

The "Video Game Sales" dataset was initially extracted from vgchartz.com and subsequently underwent a series of processing stages to ensure its technical soundness and uniformity. The data processing pipeline encompassed the subsequent pivotal steps:

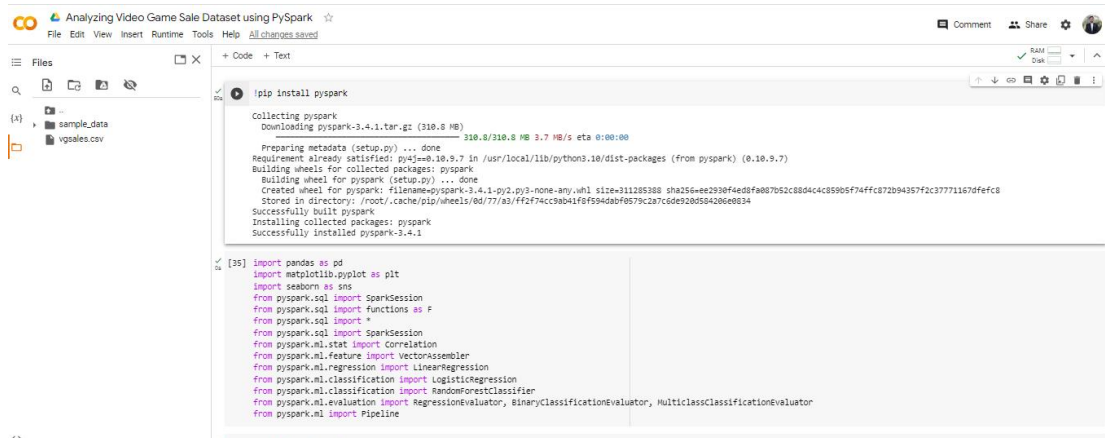
- **Data Collection:** Scrapping vgchartz.com yielded vital video game sales details, including platforms, genres, and more.
- **Data Cleaning:** A thorough process rectified missing values, anomalies, and inconsistencies. Incomplete records were removed, ensuring a reliable dataset.
- **Data Transformation:** Restructuring enabled a structured, analysis-friendly format. Attributes were labeled and formatted, optimizing exploration and modeling.
- **Data Integration:** Merging video game titles, platforms, genres, publishers, and sales data from different regions facilitated comprehensive multi-dimensional analysis.
- **Data Quality Assurance:** Rigorous checks guaranteed data accuracy. Anomalies and outliers were meticulously addressed throughout.
- **Data Enrichment:** While maintaining original integrity, the dataset gained a crucial feature – global sales attribute, offering a consolidated view of worldwide sales.

Methodology And Experiments

Our approach follows a methodical path toward accomplishing the goals of dissecting the "Video Game Sales" dataset. It involves a structured blend of data preprocessing, exploratory data analysis (EDA), regression analysis, and dynamic data visualization. Our toolkit includes the utilization of PySpark for data manipulation and analysis, while Tableau serves as the medium for crafting insightful visualizations. The subsequent sections provide a comprehensive breakdown of the techniques and tools employed for each facet of data analysis:

1) Data Preprocessing

The data preprocessing phase involves several key steps. Initially, the "Video Game Sales" dataset is loaded using PySpark, ensuring accurate data type identification and effective handling of missing values. Subsequently, a series of data cleaning techniques are employed, encompassing the removal of incomplete records and resolution of anomalies, resulting in heightened data quality. Finally, the dataset undergoes transformation into a well-structured format, complete with appropriately labeled and organized attributes, poised for further analysis. Lets get started.

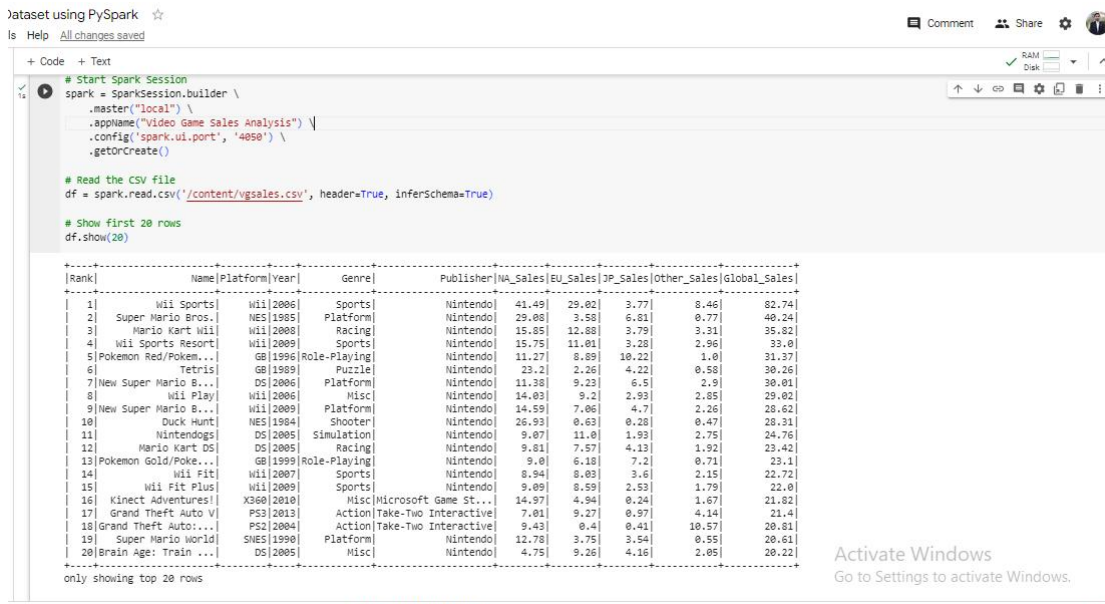


```
!pip install pyspark

collecting pyspark
Downloading pyspark-3.4.1.tar.gz (310.8 MB)
310.8/310.8 MB 3.7 MB/s eta 0:00:00
Preparing metadata (setup.py) ... done
Requirement already satisfied: py4j==4.0.9.0 in /usr/local/lib/python3.10/dist-packages (from pyspark) (4.0.9.0)
Building wheels for collected packages: pyspark
Building wheel for pyspark (setup.py) ... done
Created wheel for pyspark: filename=pyspark-3.4.1-py2.py3-none-any.whl size=311285388 sha256=ee2930f4edf807b52c8804c4c850b5f74ff672094357f2c37711670f6f8
Stored in directory: /root/.cache/pip/wheels/0d/77/03/ff2f74cc90b416f59540b087932a7c6de92050420e08034
Successfully built pyspark
Installing collected packages: pyspark
Successfully installed pyspark-3.4.1

[35]: import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from pyspark.sql import SparkSession
from pyspark.sql import functions as F
from pyspark.sql import *
from pyspark.sql import SparkSession
from pyspark.ml.stat import Correlation
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.regression import LinearRegression
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.evaluation import RegressionEvaluator, BinaryClassificationEvaluator, MulticlassClassificationEvaluator
from pyspark.ml import Pipeline
```

Figure 1: Here We had done PySpark installation and importing the Libraries



```
# Start Spark Session
spark = SparkSession.builder \
    .master("local") \
    .appName("Video Game Sales Analysis") \
    .config("spark.ui.port", "4050") \
    .getOrCreate()

# Read the CSV file
df = spark.read.csv('/content/vgsales.csv', header=True, inferSchema=True)

# Show first 20 rows
df.show(20)
```

[Rank]	[Name]	[Platform]	[Year]	[Genre]	[Publisher]	[NA_Sales]	[EU_Sales]	[JP_Sales]	[Other_Sales]	[Global_Sales]
1	Wii Sports	Wii	2006	Sports	Nintendo	41.49	29.02	3.77	0.46	82.74
2	Super Mario Bros.	NES	1985	Platform	Nintendo	29.08	3.58	6.81	0.77	40.24
3	Mario Kart Wii	Wii	2008	Racing	Nintendo	15.65	12.38	3.79	3.31	35.02
4	Wii Sports Resort	Wii	2009	Sports	Nintendo	15.75	11.01	3.28	2.96	33.00
5	Pokemon Red/Poke...	GB	1996	Role-Playing	Nintendo	11.27	8.89	10.22	1.0	31.37
6	Tetris	GB	1989	Puzzle	Nintendo	23.2	2.26	4.22	0.58	30.26
7	New Super Mario B...	DS	2006	Platform	Nintendo	11.38	9.23	6.5	2.9	30.01
8	Wii Play	Wii	2006	Misc	Nintendo	14.03	9.2	2.93	2.85	29.02
9	New Super Mario B...	Wii	2009	Platform	Nintendo	14.59	7.06	4.7	2.26	28.62
10	Duck Hunt	NES	1984	Shooter	Nintendo	26.93	0.63	0.28	0.47	28.31
11	Nintendogs	DS	2005	Simulation	Nintendo	9.07	11.0	1.93	2.75	24.76
12	Mario Kart DS	DS	2005	Racing	Nintendo	9.81	7.57	4.13	1.92	23.42
13	Pokemon Gold/Poke...	GB	1999	Role-Playing	Nintendo	9.0	6.18	7.2	0.71	23.1
14	Wii Fit	Wii	2007	Sports	Nintendo	8.94	8.03	3.6	2.15	22.72
15	Wii Fit Plus	Wii	2009	Sports	Nintendo	9.09	8.59	2.53	1.79	22.0
16	Kinect Adventures	X360	2010	Misc	Microsoft Game St...	14.97	4.94	0.24	1.67	21.82
17	Grand Theft Auto V	PS3	2013	Action	Take-Two Interactive	7.01	9.27	0.97	4.14	21.4
18	Grand Theft Auto:...	PS2	2004	Action	Take-Two Interactive	9.43	0.4	0.41	10.57	20.81
19	Super Mario World	SNES	1990	Platform	Nintendo	12.78	3.75	3.54	0.55	20.61
20	Brain Age: Train ...	DS	2005	Misc	Nintendo	4.75	9.26	4.16	2.05	20.22

only showing top 20 rows

Figure 2: Here We created the Spark Session and Loading the Dataset


```
df.fillna(value=-99, subset=["NA_Sales", "EU_Sales", "JP_Sales", "Other_Sales", "Global_Sales"]).show()
```

[Rank]	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
1	Wii Sports	Wii	2006	Sports	Nintendo	41.49	29.02	3.77	8.46	82.74
2	Super Mario Bros.	NES	1985	Platform	Nintendo	29.08	3.58	6.81	0.77	40.24
3	Mario Kart Wii	Wii	2008	Racing	Nintendo	15.85	12.88	3.79	3.31	35.82
4	Wii Sports Resort	Wii	2009	Sports	Nintendo	15.75	11.01	3.28	2.96	33.0
5	Pokemon Red/Pokem...	GB	1996	Role-Playing	Nintendo	11.27	8.89	10.22	1.0	31.37
6	Tetris	GB	1989	Puzzle	Nintendo	23.2	2.26	4.22	0.58	30.26
7	New Super Mario B...	DS	2006	Platform	Nintendo	11.38	9.23	6.5	2.9	30.01
8	Wii Play	Wii	2006	Misc	Nintendo	14.03	9.2	2.93	2.85	29.02
9	New Super Mario B...	Wii	2009	Platform	Nintendo	14.59	7.06	4.7	2.26	28.62
10	Duck Hunt	NES	1984	Shooter	Nintendo	26.93	0.63	0.28	0.47	28.31
11	Nintendogs	DS	2005	Simulation	Nintendo	9.07	11.0	1.93	2.75	24.76
12	Mario Kart DS	DS	2005	Racing	Nintendo	9.81	7.57	4.13	1.92	23.42
13	Pokemon Gold/Poke...	GB	1999	Role-Playing	Nintendo	9.0	6.18	7.2	0.71	23.1
14	Wii Fit	Wii	2007	Sports	Nintendo	8.94	8.03	3.6	2.15	22.72
15	Wii Fit Plus	Wii	2009	Sports	Nintendo	9.09	8.59	2.53	1.79	22.0
16	Kinect Adventures!	X360	2010	Misc	Microsoft Game St...	14.97	4.94	0.24	1.67	21.82
17	Grand Theft Auto V	PS3	2013	Action	Take-Two Interactive	7.01	9.27	0.97	4.14	21.4
18	Grand Theft Auto:...	PS2	2004	Action	Take-Two Interactive	9.43	0.4	0.44	10.57	20.81
19	Super Mario World	SNES	1990	Platform	Nintendo	12.78	3.75	3.54	0.55	20.61
20	Brain Age: Train ...	DS	2005	Misc	Nintendo	4.75	9.26	4.16	2.05	20.22

only showing top 20 rows

Figure 3: Here we present Data After Cleaning

2) Exploratory Data Analysis (EDA) using PySpark:

In this exploratory data analysis (EDA) phase:

- Here at first we follow descriptive statistics, such as mean, median, and standard deviation, are calculated for quantitative attributes to understand the central tendencies and variations.

dataset using PySpark ☆

s Help All changes saved

+ Code + Text

```

# Calculate Descriptive Statistics
quantitative_columns = ['NA_Sales', 'EU_Sales', 'JP_Sales', 'Other_Sales', 'Global_Sales']

# Calculate Mean
mean_values = df.select([F.mean(col).alias(f'mean_{col}') for col in quantitative_columns]).collect()[0]

# Calculate Median
median_values = df.select([F.expr(f'percentile_approx({col}, 0.5)').alias(f'median_{col}') for col in quantitative_columns]).collect()[0]

# Calculate Standard Deviation
stddev_values = df.select([F.stddev(col).alias(f'stddev_{col}') for col in quantitative_columns]).collect()[0]

# Display Descriptive Statistics
for col in quantitative_columns:
    print(f'{col} - Mean: {mean_values[f'mean_{col}']:.2f}, Median: {median_values[f'median_{col}']:.2f}, Std Dev: {stddev_values[f'stddev_{col}']:.2f}')

NA_Sales - Mean: 0.26, Median: 0.06, Std Dev: 0.82
EU_Sales - Mean: 0.15, Median: 0.02, Std Dev: 0.51
JP_Sales - Mean: 0.08, Median: 0.00, Std Dev: 0.31
Other_Sales - Mean: 0.05, Median: 0.01, Std Dev: 0.19
Global_Sales - Mean: 0.54, Median: 0.17, Std Dev: 1.56

```

only showing top 20 rows

Activating Windows
Go to Settings to activate Windows.

Executing (0s) <cell line: 2> > show() > __call__() > send_command() > send_command() > readinto()

Figure 4: Here we Calculate the Descriptive Statistics, Mean, Median, Standard Deviation

```

[8] # Data Exploration
# Total number of rows and columns
print('Rows:', df.count())
print('Columns:', len(df.columns))

Rows: 16598
Columns: 11

```

Figure 5: Here we had Done some Data Exploration

There are some question collectively provide insights into the dataset's characteristics, genre distribution, platform distribution, specific sales-related questions, and summary statistics. They contribute to a comprehensive understanding of the video game sales dataset and the video game industry's dynamics. Bellow is Question 1 code and answer, rest see in [Various Data exploration task](#) in Appendix 1.

```
[83] # Question 1: Top 3 Video Games in sports that Sell the most Globally
df.select('Name', 'Global_Sales').where(df.Genre == 'Sports').orderBy('Global_Sales', ascending=False).show(3)
```

Name	Global_Sales
Wii Sports	82.74
Wii Sports Resort	33.0
Wii Fit	22.72

only showing top 3 rows

Figure 6: Question 1 data exploration and analysis tasks

- Now we made Scatter plots, Box Plot, Histogram and Bar Plot by using Matplotlib are generated for categorical attributes of datasets.

- ◆ Histogram of All region of Sales:

Bellow is plot in single row we can find attached separate Figure See [Histogram](#) in [Appendix 1](#).

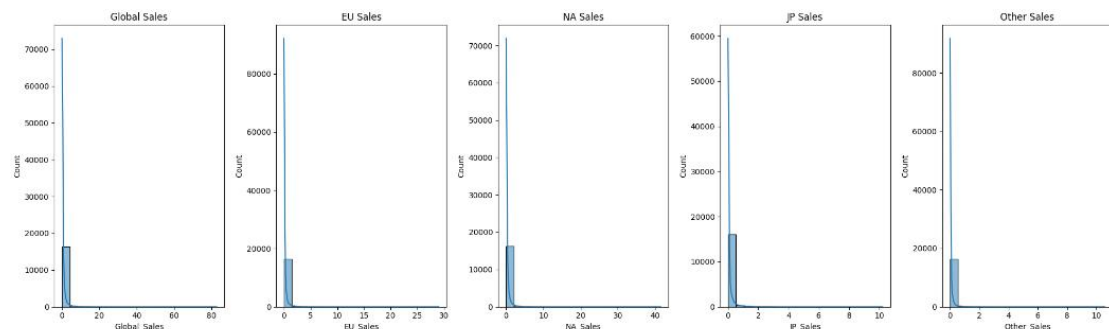


Figure 7: This is Histogram of all region of sales: the Result shows that the Sales in Japan are highest in all region.

- ◆ Scatter Plot of All region of Sales by year

Bellow is plot in a single row we can find attached separate figure .See [Scatter plot](#) [Appendix 1](#).

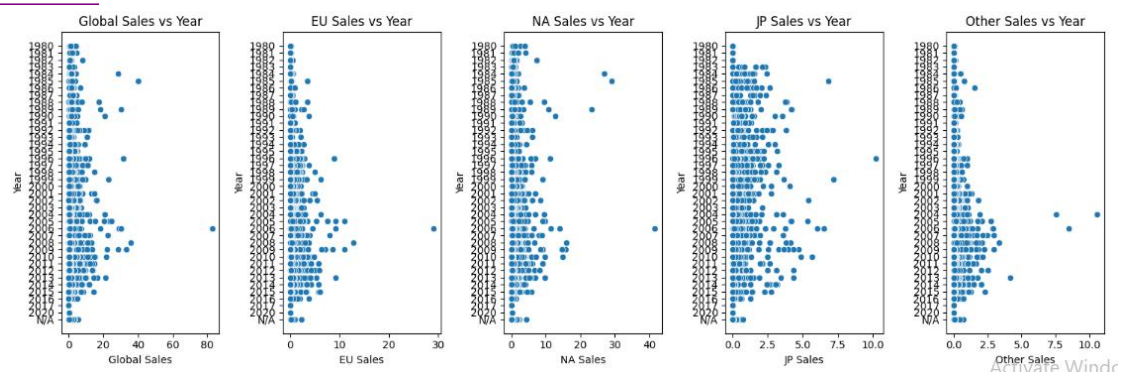


Figure 8: Here we execute the all region sales over year by using Scatter plot.

- ◆ Box Plot of All region of Sales by Genre

Bellow plot is for globe sales we can find attached separate figure See [Box plot](#) [Appendix 1](#).

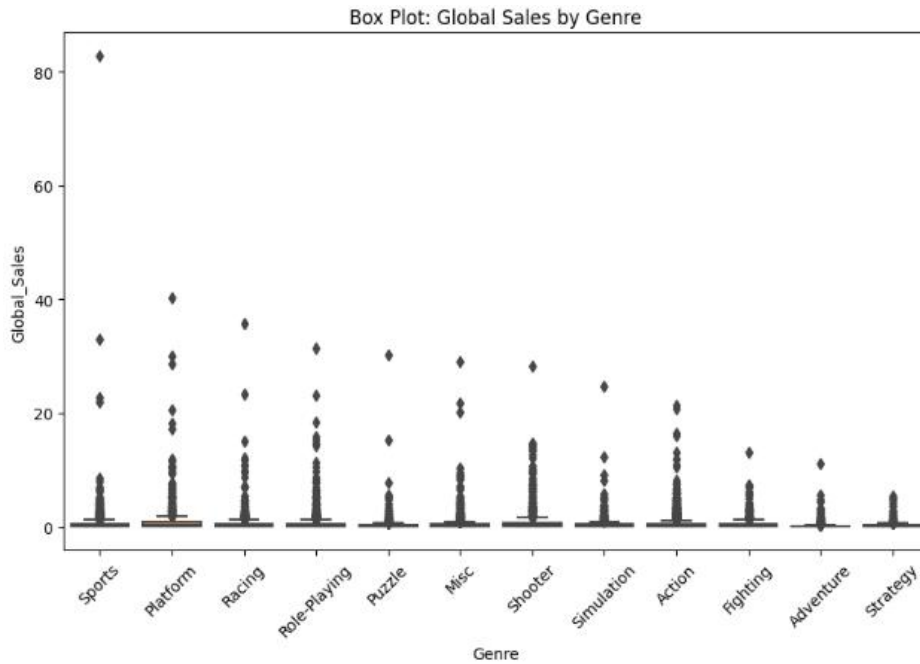


Figure 9: Here we use **Box Plot** to show **Global Sales vs Genre** Pictorially

- ◆ Bar plots using Matplotlib of All region of Sales by Platform
Bellow plot is for globe sales we can find attached separate figure See [Bar Plot Appendix 1](#)

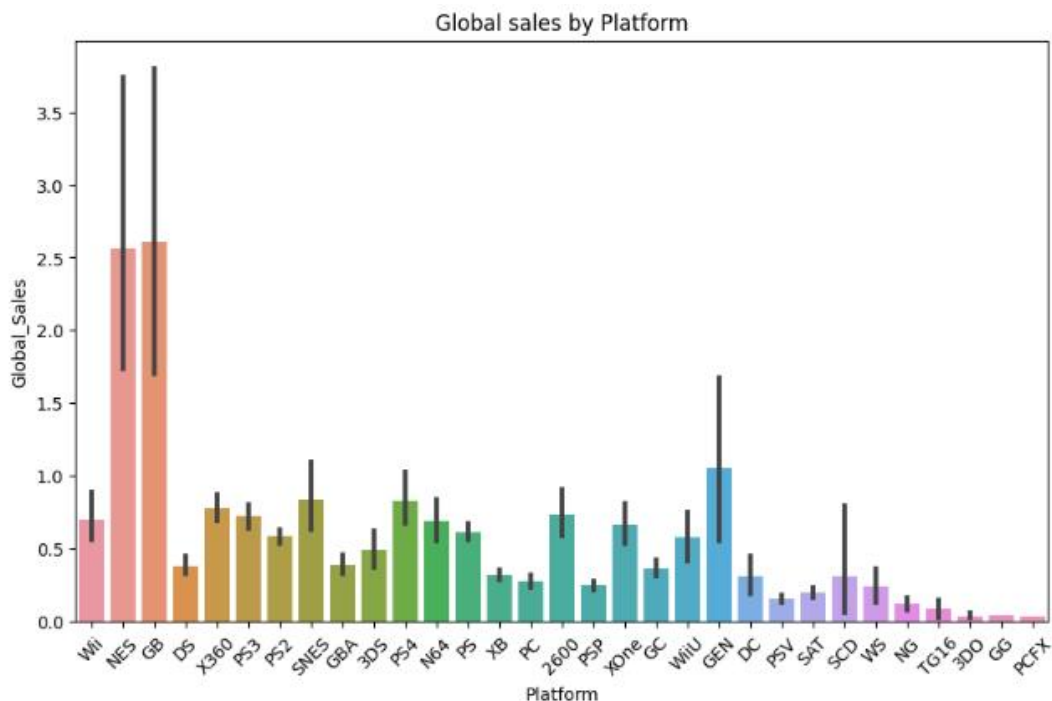


Figure 10: Here we Implement code for Bar Plot bu using Matplotlib of All region by Platform

- Correlation analysis is performed using PySpark's Correlation class to identify relationships between attributes, assisting in uncovering potential patterns.

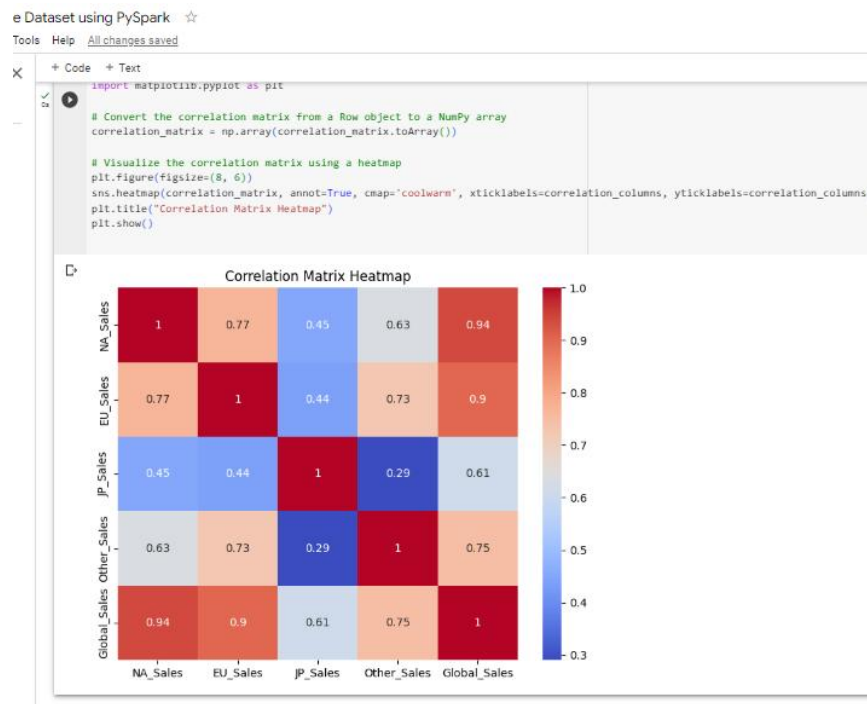


Figure 11: Correlation Matrix Heatmap

Here we calculate the correlation matrix between the specified sales columns ('NA_Sales', 'EU_Sales', 'JP_Sales', 'Other_Sales', 'Global_Sales'). The VectorAssembler is used to transform the selected columns into a feature column named "features," which is required for correlation analysis. The resulting correlation matrix will provide insights into the relationships between these sales attributes as shown in figure 11

3) Regression Analysis using PySpark:

In our regression analysis phase:

- **Linear Regression:** We construct a linear regression model utilizing PySpark's MLlib library to examine the association between chosen attributes (NA_Sales, EU_Sales, JP_Sales, Other_Sales) and the target attribute (Global_Sales).

Here the linear regression model achieved an RMSE of approximately 0.00527, which indicates how well the model's predictions match the actual Global_Sales values. Lower RMSE values indicate better model performance. Additionally, the coefficients for the features (NA_Sales, EU_Sales, JP_Sales, and Other_Sales) indicate the impact of each feature on the target Global_Sales. For example, a coefficient close to 1 suggests a strong positive relationship between that feature and the target.

```

# Linear Regression Analysis
data = df.select("NA_Sales", "EU_Sales", "JP_Sales", "Other_Sales", "Global_Sales").dropna()
feature_columns = ["NA_Sales", "EU_Sales", "JP_Sales", "Other_Sales"]
assembler = VectorAssembler(inputCols=feature_columns, outputCol="features")
data = assembler.transform(data)
train_data, test_data = data.randomSplit([0.8, 0.2], seed=42)
lr = LinearRegression(featuresCol="features", labelCol="Global_Sales")
pipeline = Pipeline(stages=[lr])
lr_model = pipeline.fit(train_data)
predictions = lr_model.transform(test_data)
evaluator = RegressionEvaluator(labelCol="Global_Sales", metricName="rmse")
rmse = evaluator.evaluate(predictions)
print("Linear Regression RMSE:", rmse)
lr_coefficients = lr_model.stages[-1].coefficients
print("Linear Regression Coefficients:", lr_coefficients)

```

Linear Regression RMSE: 0.005268710142527473
 Linear Regression Coefficients: [0.9999544028780193, 1.0000392692552083, 0.9999137696590534, 0.9992546028167025]

Figure 12: the linear regression analysis provides insights into the relationships between the sales attributes and how they contribute to the overall global sales.

- **Logistic Regression:** A logistic regression model is developed to forecast whether a game attains top-selling status based on sales attributes, contributing to an understanding of success determinants.

```
# Logistic Regression Analysis
data = df.select("NA_Sales", "EU_Sales", "JP_Sales", "Other_Sales", "Global_Sales").dropna()
data = data.withColumn("TopSeller", (data["Global_Sales"] > 1).cast("int"))
feature_columns = ["NA_Sales", "EU_Sales", "JP_Sales", "Other_Sales"]
assembler = VectorAssembler(inputCols=feature_columns, outputCol="features")
data = assembler.transform(data)
train_data, test_data = data.randomSplit([0.8, 0.2], seed=42)
lr = LogisticRegression(featuresCol="features", labelCol="TopSeller")
pipeline = Pipeline(stages=[lr])
lr_model = pipeline.fit(train_data)
predictions = lr_model.transform(test_data)
evaluator = BinaryClassificationEvaluator(labelCol="TopSeller")
accuracy = evaluator.evaluate(predictions, {evaluator.metricName: "areaUnderROC"})
print("Logistic Regression Area under ROC:", accuracy)
```

Logistic Regression Area under ROC: 1.0

Figure 13: the logistic regression analysis provides insights into the attributes that contribute to a game becoming a top seller in the video game industry.

In figure 13, An area under the ROC curve (ROC-AUC) of 1.0 indicates that the logistic regression model has achieved perfect separation between the two classes (top sellers and non-top sellers), resulting in a highly accurate prediction. This could be due to a very clear distinction between the attributes of top-selling and non-top-selling games in the dataset.

- **Random Forest Classifier:** Implementation of a random forest classifier aids in predicting top-selling games, incorporating sales metrics and platform data. This ensemble model unveils the significance of various features.

```
# Random Forest Classifier Analysis
data = df.select("NA_Sales", "EU_Sales", "JP_Sales", "Other_Sales", "Global_Sales").dropna()
data = data.withColumn("TopSeller", (data["Global_Sales"] > 1).cast("int"))
feature_columns = ["NA_Sales", "EU_Sales", "JP_Sales", "Other_Sales"]
assembler = VectorAssembler(inputCols=feature_columns, outputCol="features")
data = assembler.transform(data)
train_data, test_data = data.randomSplit([0.8, 0.2], seed=42)
rf = RandomForestClassifier(featuresCol="features", labelCol="TopSeller")
pipeline = Pipeline(stages=[rf])
rf_model = pipeline.fit(train_data)
predictions = rf_model.transform(test_data)
evaluator = BinaryClassificationEvaluator(labelCol="TopSeller")
accuracy = evaluator.evaluate(predictions, {evaluator.metricName: "areaUnderROC"})
print("Random Forest Classifier Area under ROC:", accuracy)
```

Random Forest Classifier Area under ROC: 0.9976716101510846

Figure 14: Random Forest Classifier

From figure 14, An area under the ROC curve (ROC-AUC) close to 1.0 indicates that the random forest classifier model is performing well in predicting top-selling games. This analysis provides valuable insights into the importance of various features in determining a game's status as a top sell

4) Data Preparation for Tableau:

In the "Data Preparation for Tableau" phase, the data that has been processed and analyzed using PySpark is prepared and exported in a format that is well-suited for visualization in Tableau. This phase ensures that the insights and outcomes derived from the data analysis are effectively communicated through interactive and visually appealing visualizations.

Proof of Tableau installation See [Tableau Appendix 1](#)

5) Interactive Data Visualization using Tableau:

In the interactive data visualization phase:

- Tableau is utilized to craft diverse visualizations including scatter plots, bar charts, histograms, and box plots etc. I attached one Visual of Tableau, for more screen of visuals see [Appendix 1](#)

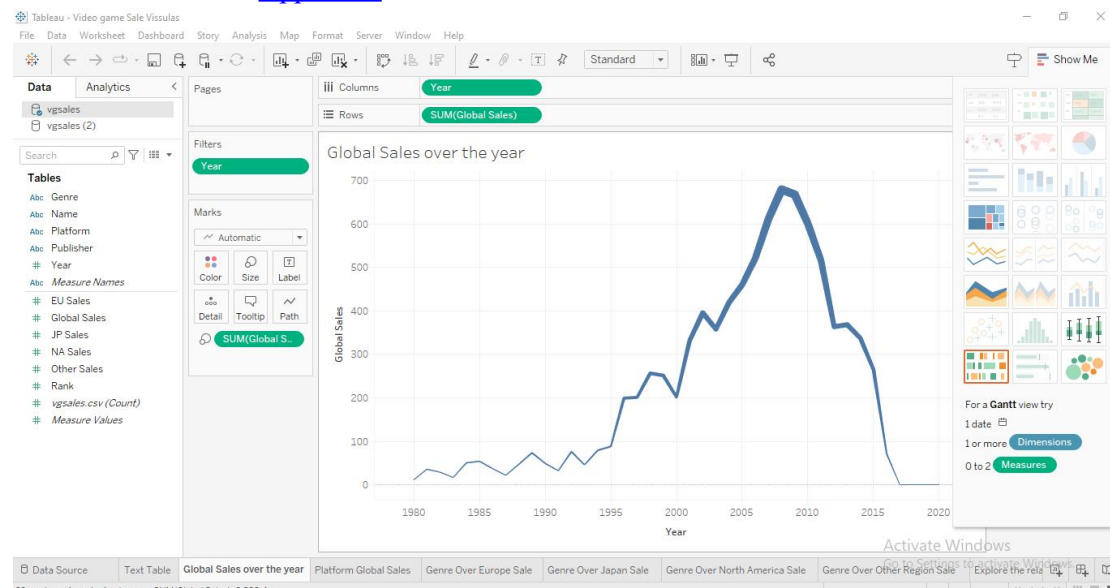


Figure 15: Tableau Visualization screen 1

We Created 14 Tableau screens for Visualization. We can access the screens in [Tableau Visuals](#) in Appendix 1.

Screen 1: Analysis of Text Table

Screen 2: Analysis of Global Sales over the year

Screen 3: Analysis of Platform Global Sales

Screen 4: Analysis of Genre Over Europe Sale

Screen 5: Analysis of Genre Over Japan Sale

Screen 6: Analysis of Genre Over North America Sale

Screen 7: Analysis of Genre Over Other Region Sale

Screen 8: Explore the relationship between Platform and Global sale with tables

Screen 9: Explore all region of Sales with Globally

Screen 10: Analysis of Global Sales over all other Region Sales

Screen 11: Europe and North America Sales Analysis

Screen 12: Analysis of Japan and other sale

Screen 13: Analysis of Heatmap of Sales by Genre and Platform

Screen 14: Analyzing Japan Sales over Year by genre

- Dashboard design is implemented to offer a unified perspective of the discoveries, facilitating interactive exploration & extraction of insights. I attached one Dashboard of Tableau, for more screen of visuals see [Tableau Dashboard Appendix 1](#)

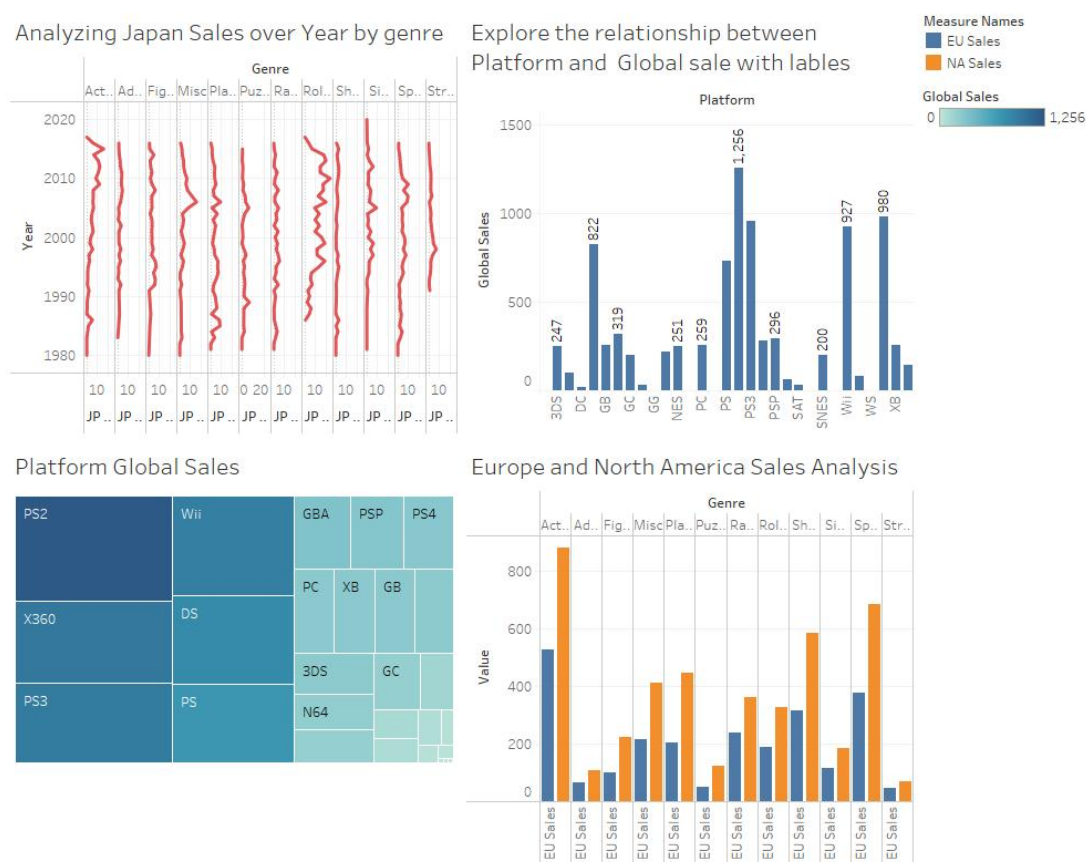


Figure 16: Dashboard Visualization screen 1

We Created 3 Tableau Dashboards to analyze the result via Visualization. We can access the screens in [Tableau Dashboard](#) in Appendix 1.

Dashboard 1: Analyzing the result of Different Region Sales of Video Games with different attributes

Dashboard 2: Analyzing the result of Global sales Sales of Video Games with different attributes.

Dashboard 3: Global Sales with all other regions of dataset.

6) Model Evaluation using PySpark:

For the regression models (linear regression and logistic regression), we evaluate their performance using appropriate metrics such as RMSE (Root Mean Squared Error) and AUC-ROC (Area Under the Receiver Operating Characteristic Curve).

Software and Tools Used:

We all Install Linux in Virtual box For Installation proof see [Linux installation in appendix 1](#)

We used PySpark for data loading, preprocessing, exploratory data analysis, regression analysis, Classification analysis, and model evaluation. For Installation proof see [Installation spark in appendix 1](#)

We used Tableau: For interactive data visualization and dashboard creation. For Installation proof see [Tableau Installation appendix 1](#)
Python: For scripting, data manipulation, and model building.

Social and Ethical Consideration

The project's social impact lies in its potential to inform industry decisions, enabling better game development strategies and resource allocation. Ethical considerations encompass responsible data handling, privacy, and transparency in model interpretation. Ensuring fair representation of diverse gaming communities is vital to avoid perpetuating biases. Striking this balance fosters positive impacts on both the gaming industry and its players.

Result And Discussion

The analysis of the "Video Game Sales" dataset using PySpark and Tableau has yielded several noteworthy findings that provide valuable insights into the video game industry. The discussion of these findings is as follows:

Conclusion And Future Work

In conclusion, our analysis of the "Video Game Sales" dataset using PySpark and Tableau has provided significant insights into the dynamics of the video game industry. We've explored sales trends, identified influential factors, and constructed predictive models. Through descriptive statistics, visualizations, and regression analyses, we've gained a comprehensive understanding of game sales patterns. Future work might involve refining models, exploring new variables like reviews or emerging technologies, and maintaining ethical considerations. This balanced approach aims to enhance industry practices while ensuring positive societal impact.

Bibliography and References:

- [1] Ricard Gil , Frederic Warzynski, Vertical Integration “Exclusivity, and Game Sales Performance in the US Video Game Industry” The Journal of Law, Economics, and Organization, Volume 31, Issue suppl_1, August 2015 <https://doi.org/10.1093/jleo/ewu006>
- [2] Timothy Derdenger, “Technological tying and the intensity of price competition: An empirical analysis of the video game industry”. Quant Mark Econ 12, 127–165 June 2014 <https://doi.org/10.1007/s11129-014-9143-9>
- [3] Babb, J., Terry, N., & Dana, K. (2013). “The Impact Of Platform On Global Video Game Sales.” International Business & Economics Research Journal (IBER), 12(10), 1273–1288. <https://doi.org/10.19030/iber.v12i10.8136>
- [4] Sujala D Shetty, (December 2021). “Sentiment Analysis, Tweet Analysis and Visualization on Big Data Using Apache Spark and Hadoop.”IOP Conf. Ser.: Mater. Sci. Eng
- [5] M. Singh, B. Ghutla, R. Lilo Jnr, A. F. S. Mohammed and M. A. Rashid(2017), "Walmart's Sales Data Analysis - A Big Data Analytics Perspective," 4th Asia-Pacific World Congress on Computer Science and Engineering.
- [6] Claude E. Concolato and Li M. Chen(2017), “ A New Paradigm in the Age of Big-Data Science and Analytics”, New Mathematics and Natural ComputationVol. 13, No. 02, pp. 119-143, <https://doi.org/10.1142/S1793005717400038>

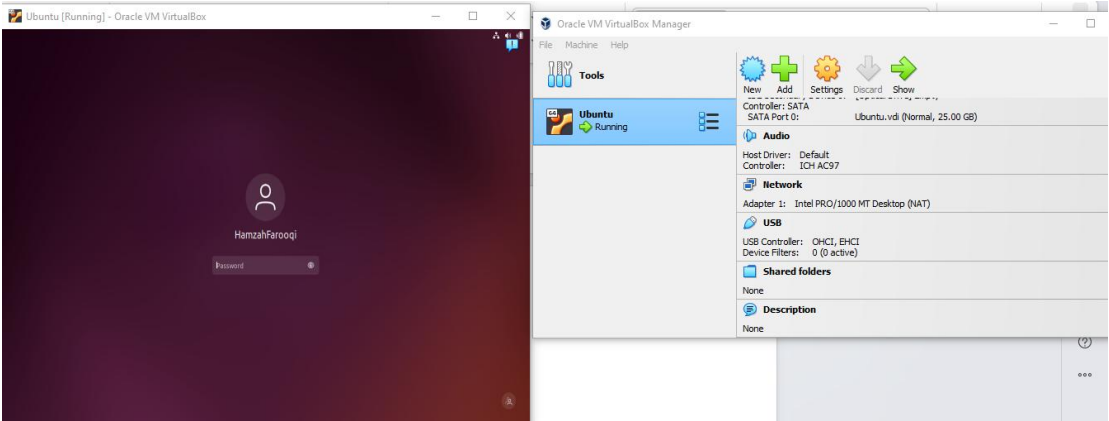
Appendix I: Screenshots of Tasks

Installation Spark

```
hamzahfaroqi@Ubuntu: ~  
hamzahfaroqi@Ubuntu:~$ wget http://dclcdn.apache.org/spark/spark-3.4.0/spark-3.4.0-bin-hadoop3.tgz  
--2023-08-15 22:29:00-- http://dclcdn.apache.org/spark/spark-3.4.0/spark-3.4.0-bin-hadoop3.tgz  
Resolving dclcdn.apache.org (dclcdn.apache.org)... 151.101.2.132, 2a04:4e42::644  
Connecting to dclcdn.apache.org (dclcdn.apache.org)[151.101.2.132]:80... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 388407094 (370M) [application/x-gzip]  
Saving to: 'spark-3.4.0-bin-hadoop3.tgz'  
  
spark-3.4.0-bin-had 100%[=====] 370.41M 773KB/s in 6m 51s  
  
2023-08-15 22:35:58 (924 KB/s) - 'spark-3.4.0-bin-hadoop3.tgz' saved [388407094/388407094]  
  
hamzahfaroqi@Ubuntu:~$ tar xvf spark-3.4.0-bin-hadoop3.tgz  
spark-3.4.0-bin-hadoop3/  
spark-3.4.0-bin-hadoop3/data/  
spark-3.4.0-bin-hadoop3/data/streaming/  
spark-3.4.0-bin-hadoop3/data/streaming/AFINN-111.txt  
spark-3.4.0-bin-hadoop3/data/graphx/  
spark-3.4.0-bin-hadoop3/data/graphx/followers.txt  
spark-3.4.0-bin-hadoop3/data/graphx/users.txt  
spark-3.4.0-bin-hadoop3/data/mllib/  
spark-3.4.0-bin-hadoop3/data/mllib/als/  
spark-3.4.0-bin-hadoop3/data/mllib/als/test.data  
spark-3.4.0-bin-hadoop3/data/mllib/als/sample_movielens_ratings.txt  
spark-3.4.0-bin-hadoop3/data/mllib/gmm_data.txt  
spark-3.4.0-bin-hadoop3/data/mllib/sample_libsvm_data.txt  
spark-3.4.0-bin-hadoop3/data/mllib/sample_linear_regression_data.txt  
spark-3.4.0-bin-hadoop3/data/mllib/sample_fpgrowth.txt  
spark-3.4.0-bin-hadoop3/data/mllib/kmeans_data.txt  
spark-3.4.0-bin-hadoop3/data/mllib/pic_data.txt  
spark-3.4.0-bin-hadoop3/data/mllib/streaming_kmeans_data_test.txt  
spark-3.4.0-bin-hadoop3/data/mllib/sample_linear_regression_data.txt  
spark-3.4.0-bin-hadoop3/data/mllib/ridge_data/  
spark-3.4.0-bin-hadoop3/data/mllib/ridge_data/lpsa.data  
spark-3.4.0-bin-hadoop3/data/mllib/sample_binary_classification_data.txt  
spark-3.4.0-bin-hadoop3/data/mllib/pagerank_data.txt  
spark-3.4.0-bin-hadoop3/data/mllib/sample_isotonic_regression_libsvm_data.txt  
spark-3.4.0-bin-hadoop3/data/mllib/sample_kmeans_data.txt  
spark-3.4.0-bin-hadoop3/data/mllib/sample_lda_libsvm_data.txt
```

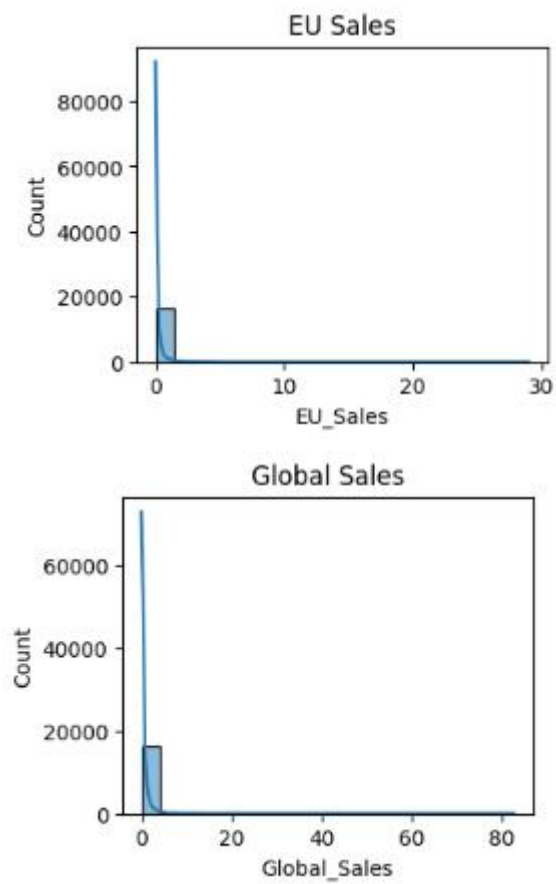
```
hamzahfaroqi@Ubuntu: ~  
spark-3.4.0-bin-hadoop3/bin/spark-connect-shell  
spark-3.4.0-bin-hadoop3/bin/spark-shell2.cmd  
spark-3.4.0-bin-hadoop3/bin/pyspark2.cmd  
spark-3.4.0-bin-hadoop3/bin/spark-submit.cmd  
spark-3.4.0-bin-hadoop3/bin/spark-submit2.cmd  
hamzahfaroqi@Ubuntu:~$ sudo mv spark-3.4.0-bin-hadoop3 /opt/spark  
mv: cannot overwrite non-directory '/opt/spark' with directory 'spark-3.4.0-bin-hadoop3'  
hamzahfaroqi@Ubuntu:~$ cd /opt  
hamzahfaroqi@Ubuntu:/opt$ ls  
spark  
hamzahfaroqi@Ubuntu:/opt$ rm -rf spark  
rm: cannot remove 'spark': Permission denied  
hamzahfaroqi@Ubuntu:/opt$ sudo rm -rf spark  
hamzahfaroqi@Ubuntu:/opt$ cd  
hamzahfaroqi@Ubuntu:~$ sudo mv spark-3.4.0-bin-hadoop3 /opt/spark  
hamzahfaroqi@Ubuntu:~$ nano ~/.bashrc  
hamzahfaroqi@Ubuntu:~$ source ~/.bashrc  
hamzahfaroqi@Ubuntu:~$ spark-shell  
23/08/15 22:43:42 WARN Utils: Your hostname, Ubuntu resolves to a loopback address: 127.0.0.1; using 10.0.2.15 instead (on interface enp0s3)  
23/08/15 22:43:42 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address  
Setting default log level to "WARN".  
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).  
23/08/15 22:44:13 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable  
Spark context Web UI available at http://10.0.2.15:4040  
Spark context available as 'sc' (master = local[*], app id = local-1692135860199).  
Spark session available as 'spark'.  
Welcome to  
  
Spark version 3.4.0  
  
Using Scala version 2.12.17 (OpenJDK 64-Bit Server VM, Java 1.8.0_382)  
Type in expressions to have them evaluated.  
Type :help for more information.
```

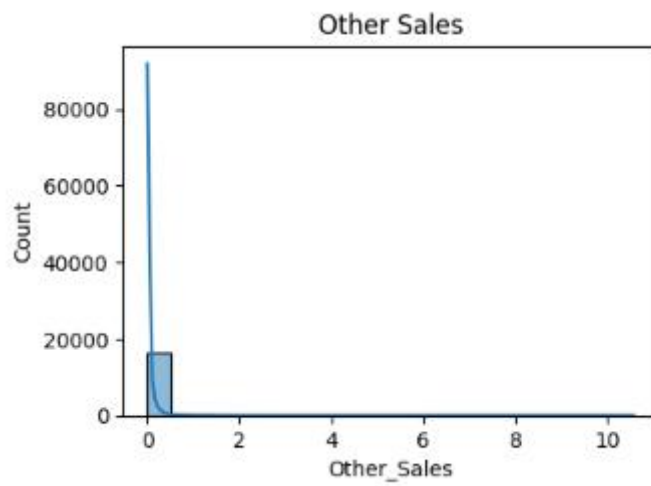
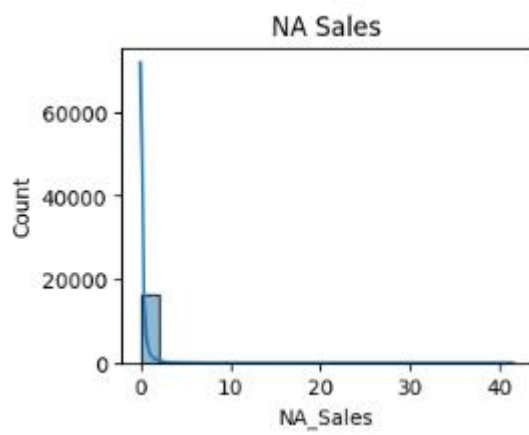
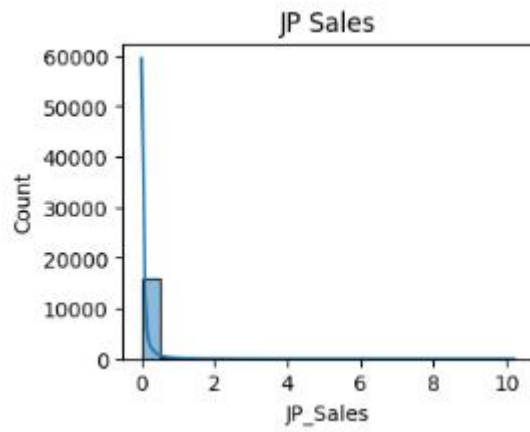
Linux Installation



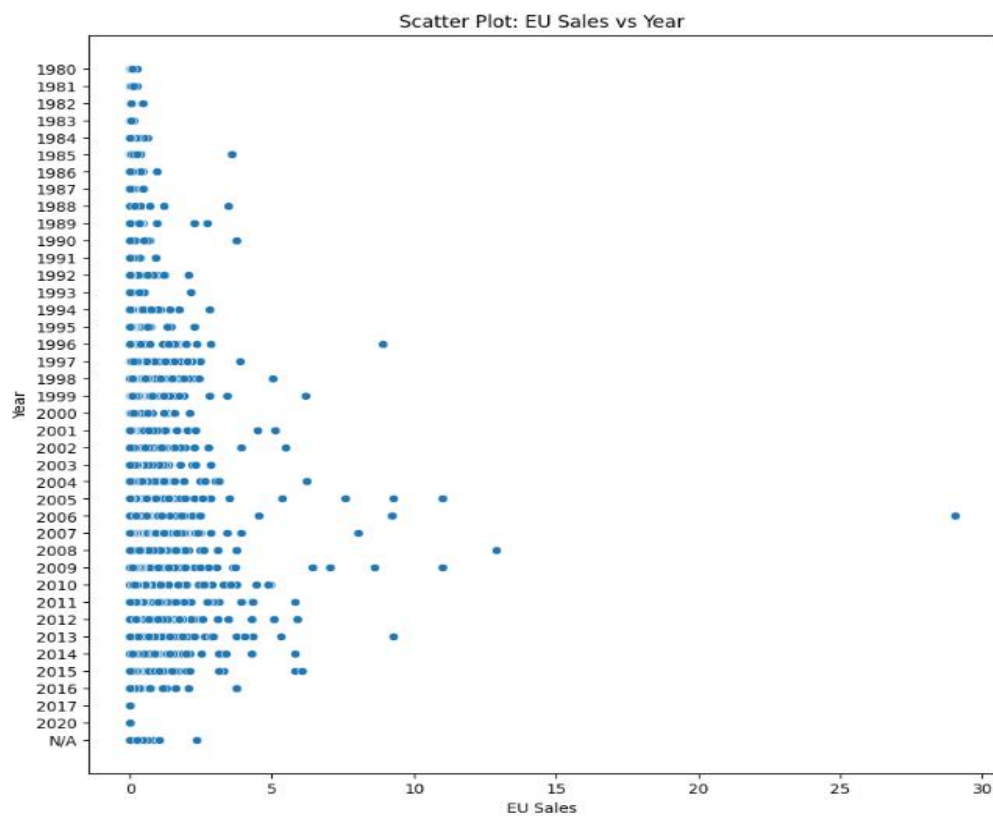
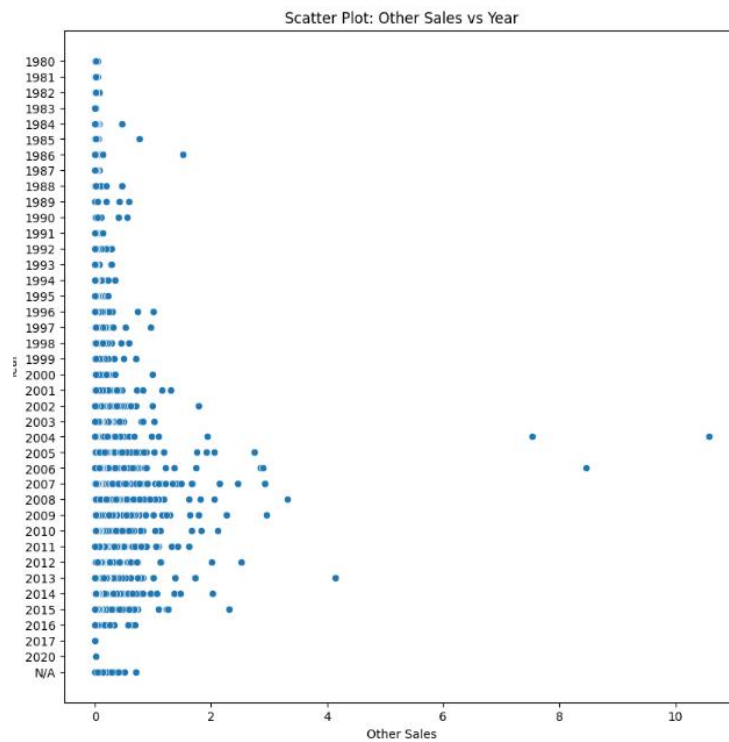
Code of Task 2 Images

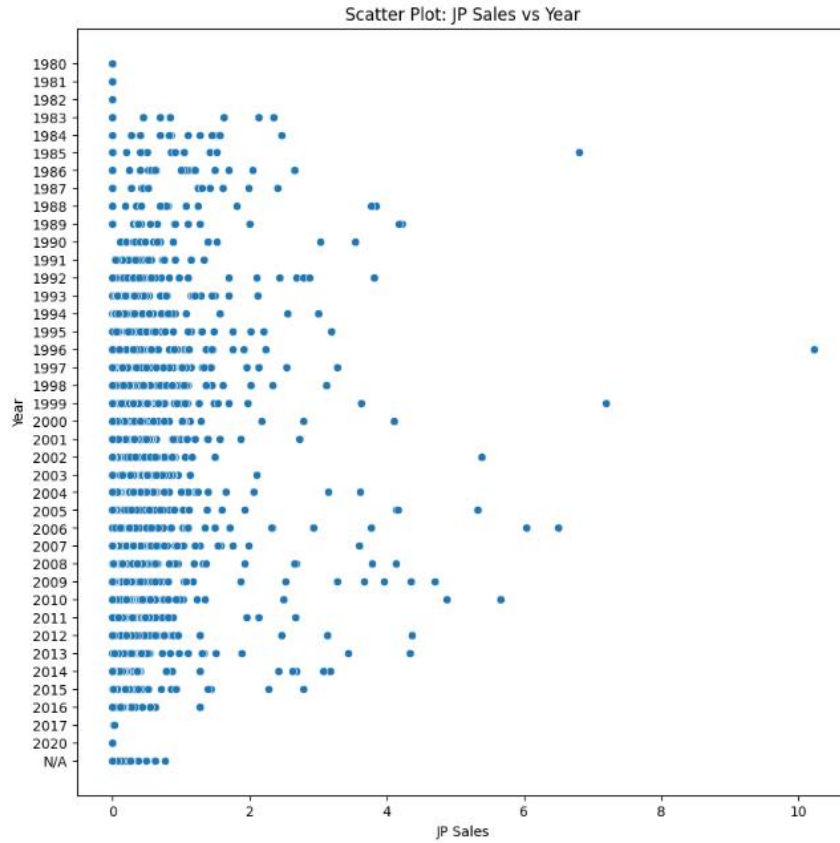
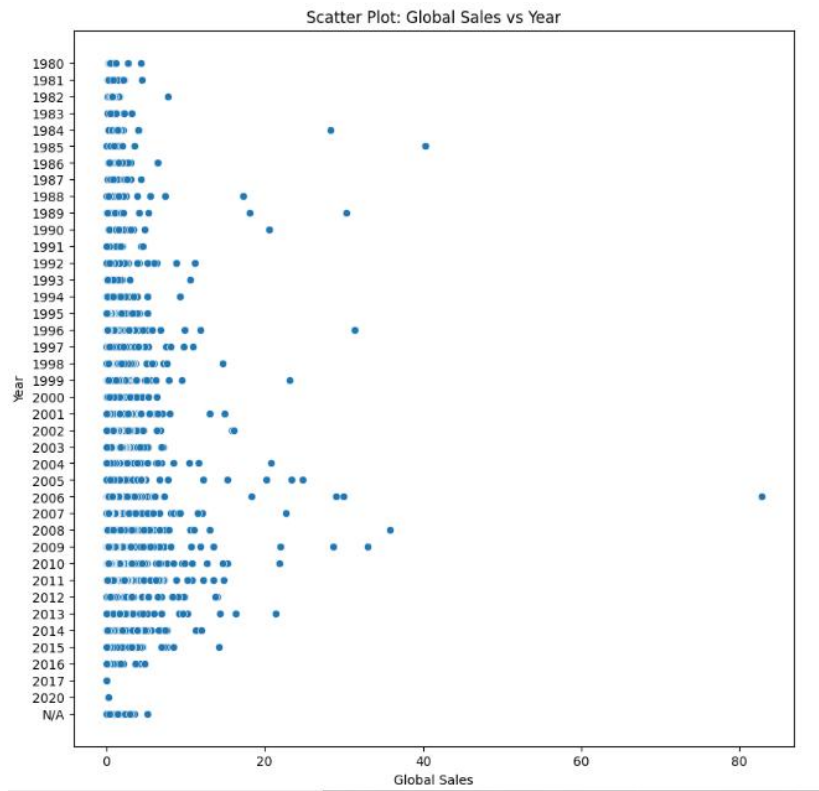
Histogram

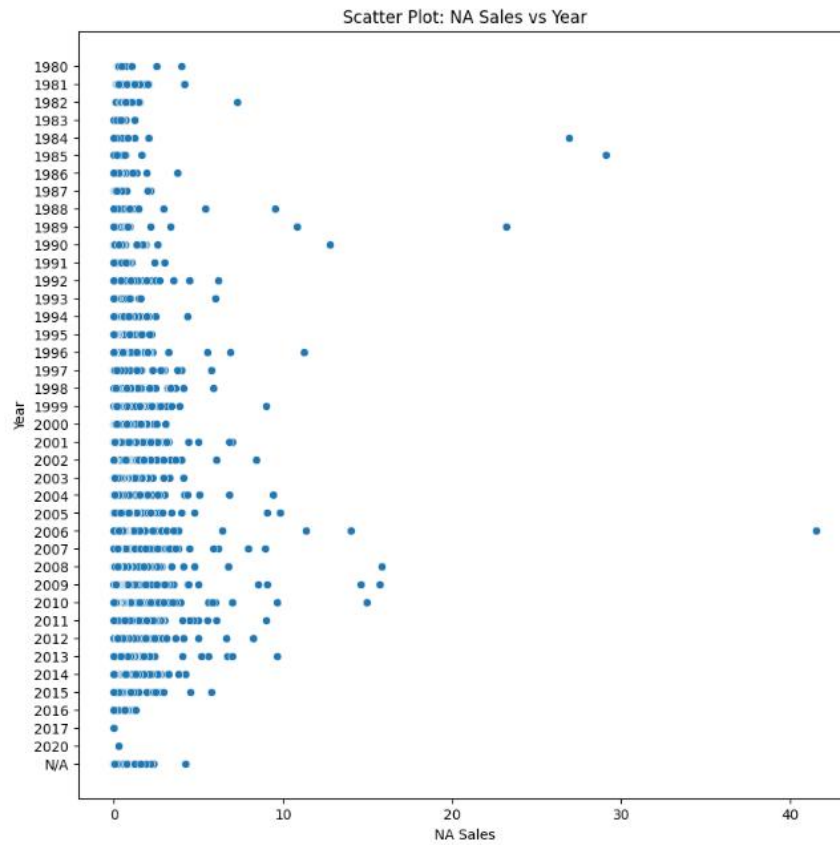




Scatter Plot

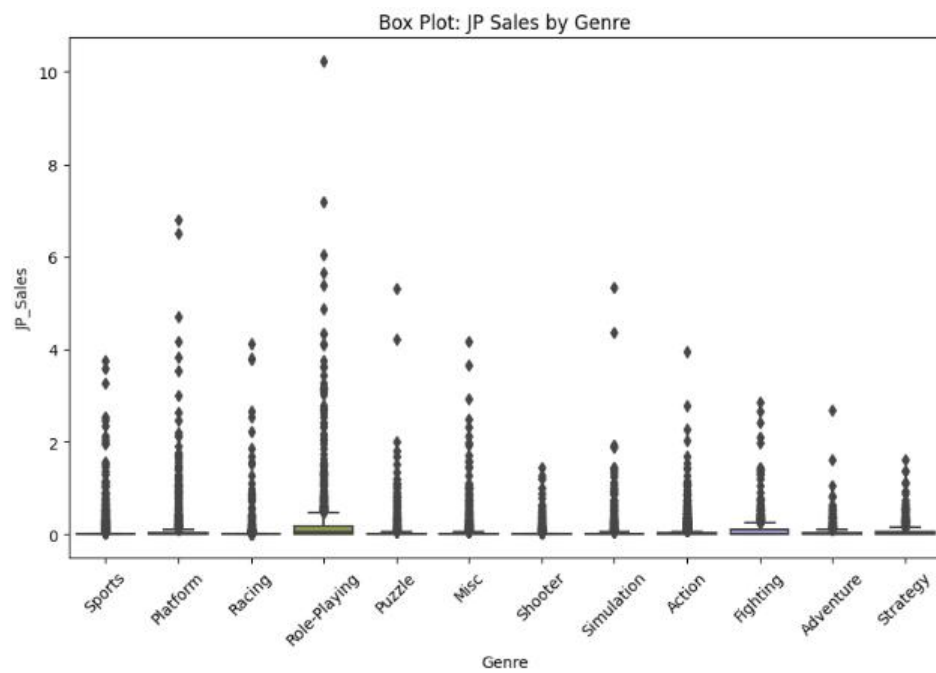


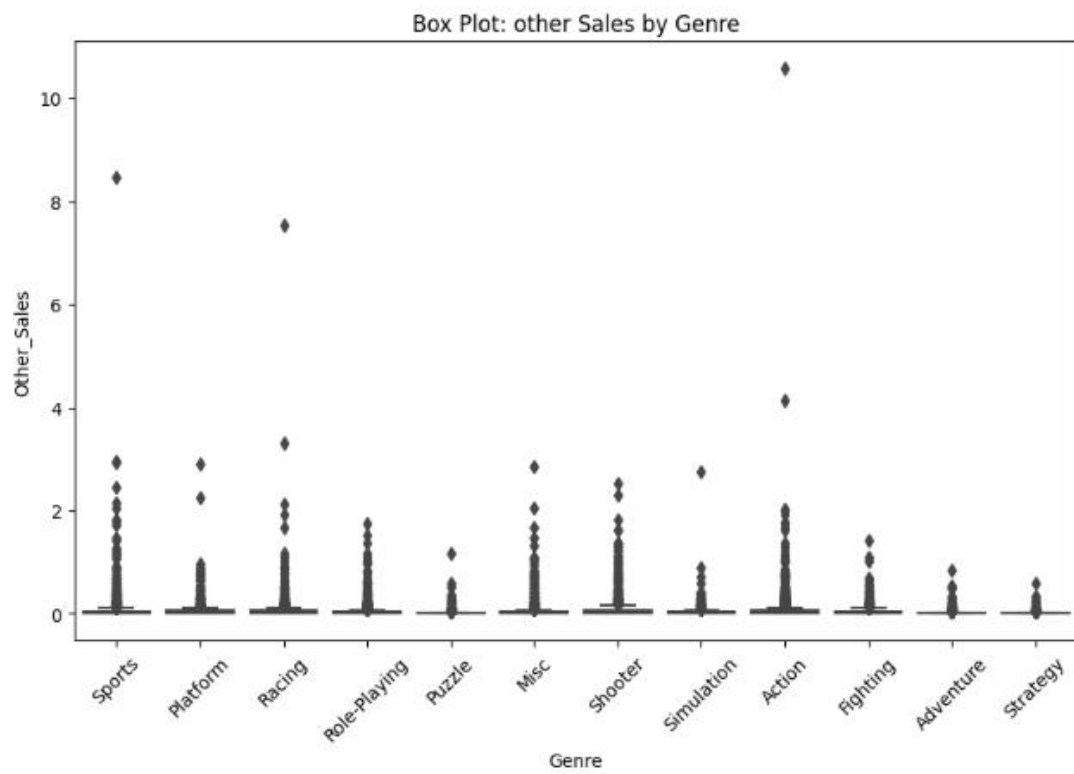
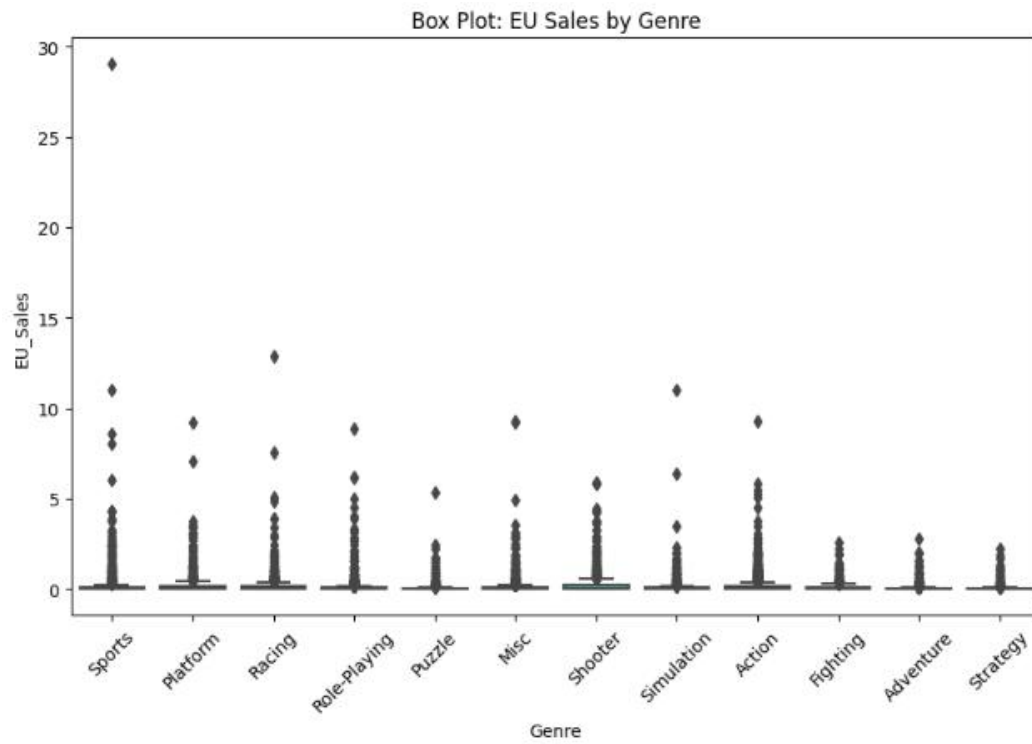


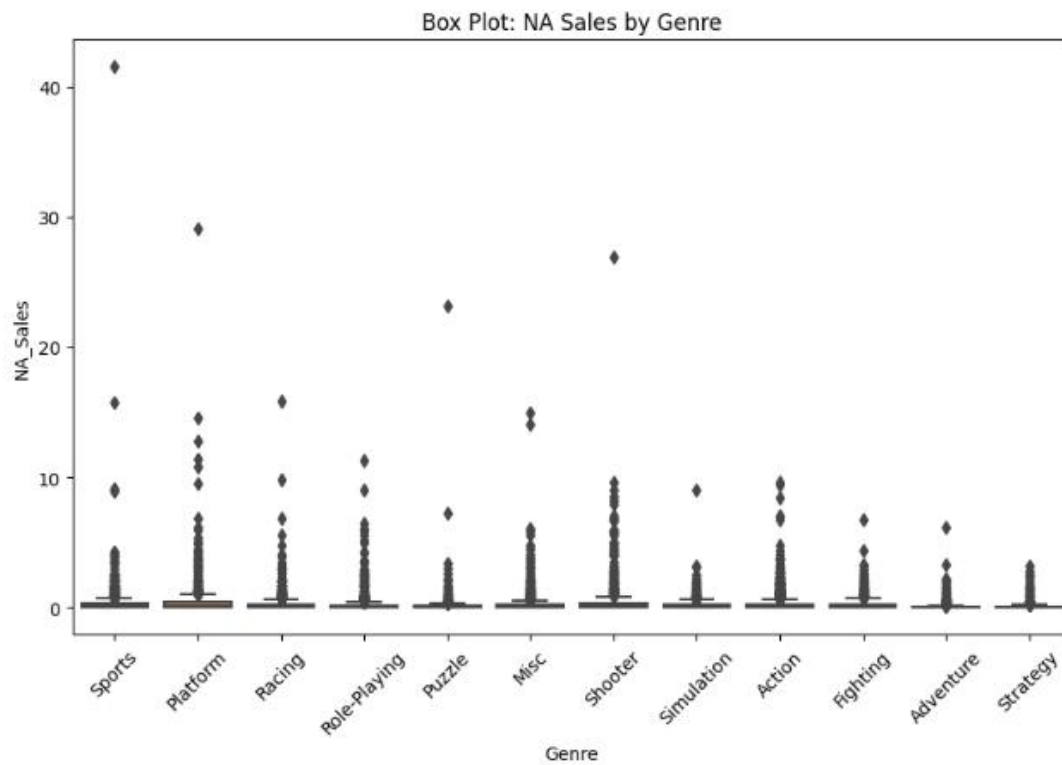


Scatter Plot: JP Sales vs Year

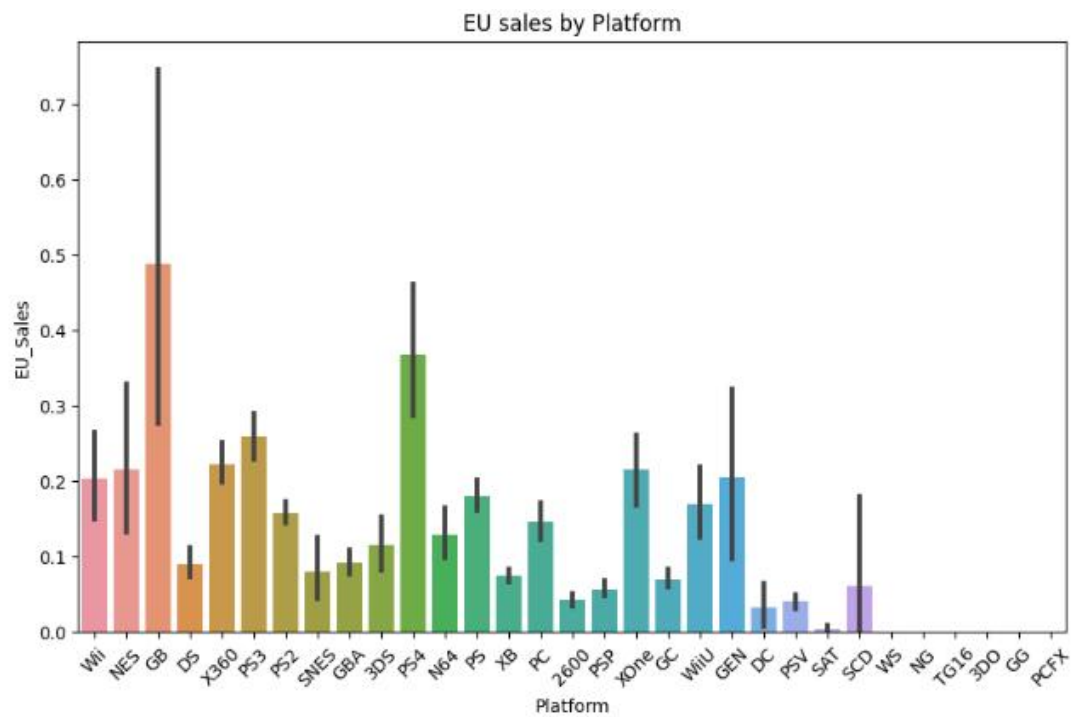
Box Plot

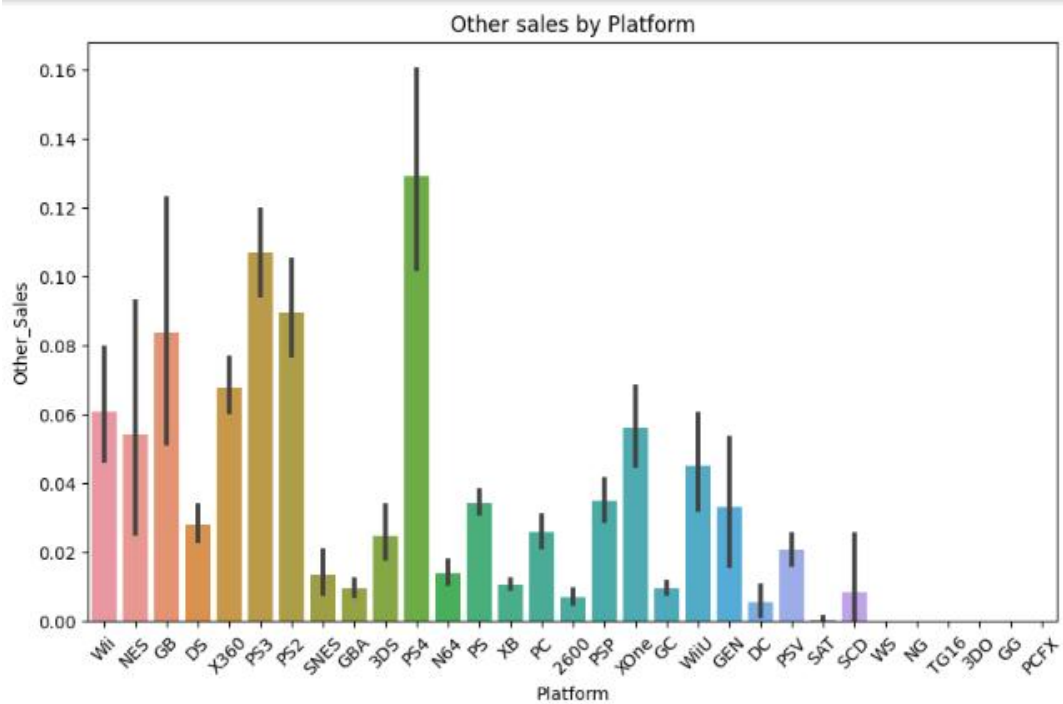
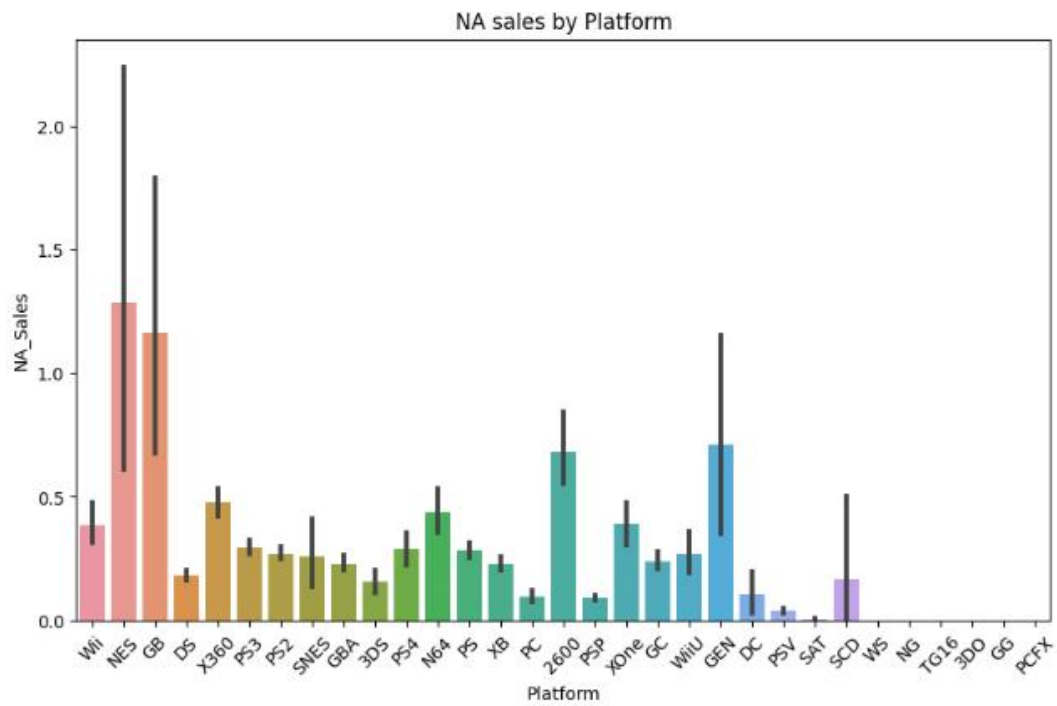


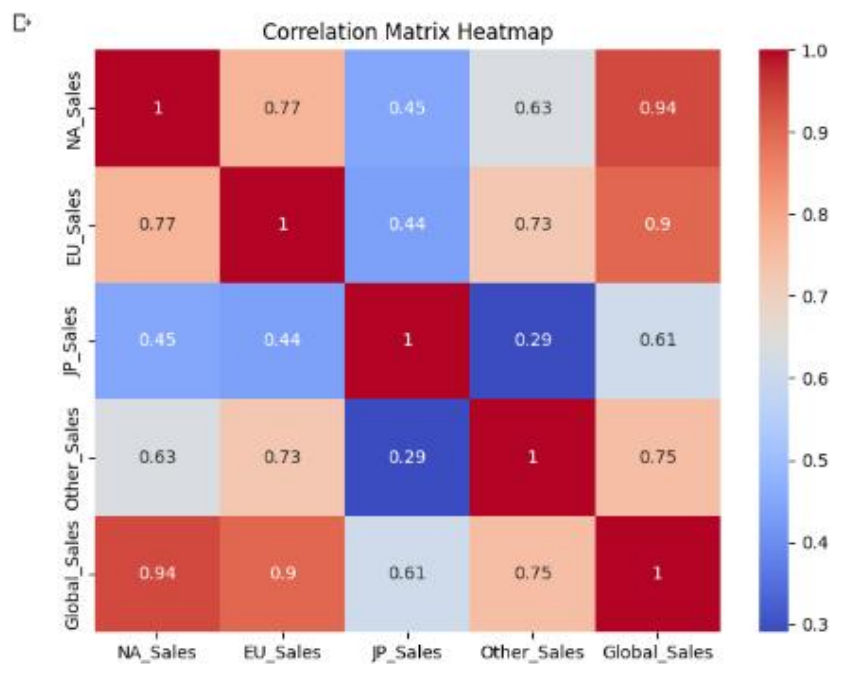
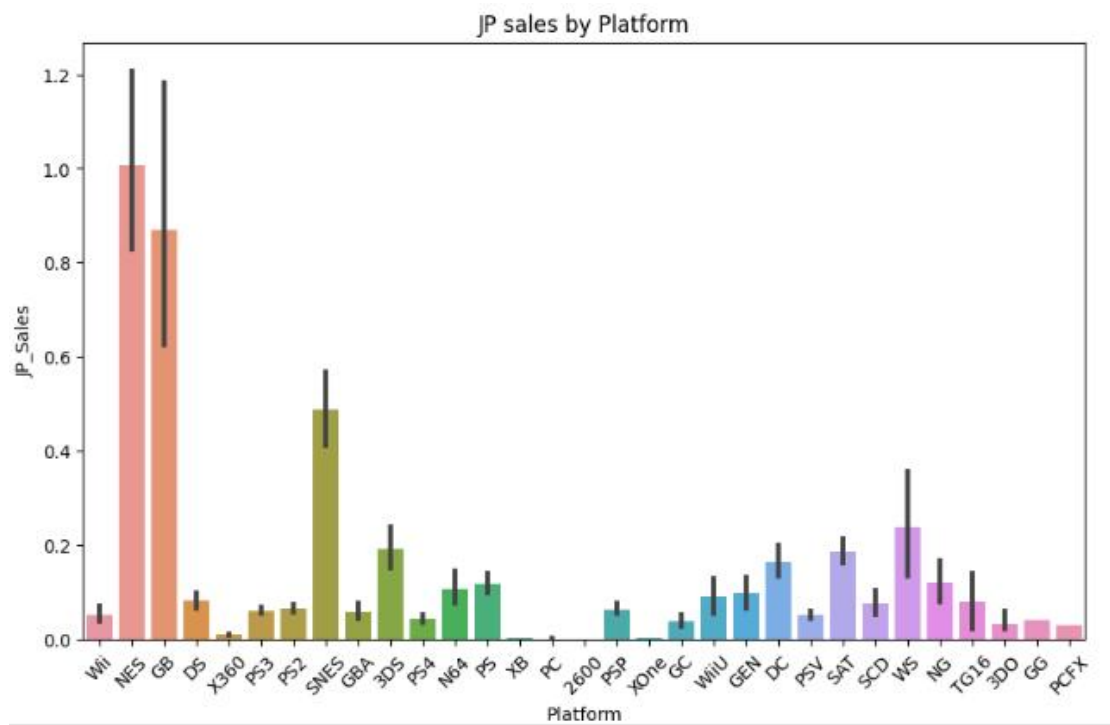




Bar plots using Matplotlib







Various data exploration and analysis tasks

```
✓ [84] # Question 2: Top 5 Publisher in Racing who Have highest sale in Europe  
0s df.select('Publisher', 'EU_Sales').filter(df.Genre == 'Racing').orderBy('EU_Sales', ascending=False).show(5)
```

Publisher	EU_Sales
Nintendo	12.88
Nintendo	7.57
Sony Computer Ent...	5.09
Sony Computer Ent...	4.88
Nintendo	3.91

```
✓ [85] # Question 3: Name 5 lowest Sales Publisher Globally  
2s df.select('Publisher', 'Global_Sales').orderBy('Global_Sales', ascending=True).show(5)
```

Publisher	Global_Sales
Touchstone	0.01
Nobilis	0.01
Nippon Ichi Software	0.01
Namco Bandai Games	0.01
Electronic Arts	0.01

only showing top 5 rows

```
🔍 # Question 4: Average sale for various genres  
df.groupBy('Genre').agg(F.round(F.mean('Global_Sales'), 2).alias('average_sales')).show()
```

Genre	average_sales
Adventure	0.19
Sports	0.57
Racing	0.59
Role-Playing	0.62
Shooter	0.79
Misc	0.47
Platform	0.94
Puzzle	0.42
Fighting	0.53
Action	0.53
Strategy	0.26
Simulation	0.45

```
[87] # Question 5: List of Name with sale in 2015
df.select('Name', 'Publisher', 'Global_Sales').where(df.Year == '2015').orderBy('Global_Sales', ascending=False).show()
```

Name	Publisher	Global_Sales
Call of Duty: Bla...	Activision	14.24
FIFA 16	Electronic Arts	8.49
Star Wars Battlef...	Electronic Arts	7.67
Call of Duty: Bla...	Activision	7.3
Fallout 4	Bethesda Softworks	6.96
Splatoon	Nintendo	4.57
Uncharted: The Na...	Sony Computer Ent...	4.47
Halo 5: Guardians	Microsoft Game St...	4.26
Fallout 4	Bethesda Softworks	4.09
NBA 2K16	Take-Two Interactive	3.85
Batman: Arkham Kn...	Warner Bros. Inte...	3.79
The Witcher 3: Wi...	Namco Bandai Games	3.73
Star Wars Battlef...	Electronic Arts	3.49
Metal Gear Solid ...	Konami Digital En...	3.38
Assassin's Creed ...	Ubisoft	3.28
Monster Hunter X	Capcom	3.26
FIFA 16	Electronic Arts	3.23
Madden NFL 16	Electronic Arts	3.22
Super Mario Maker	Nintendo	3.18
Gears of War: Ult...	Microsoft Game St...	3.0

only showing top 20 rows

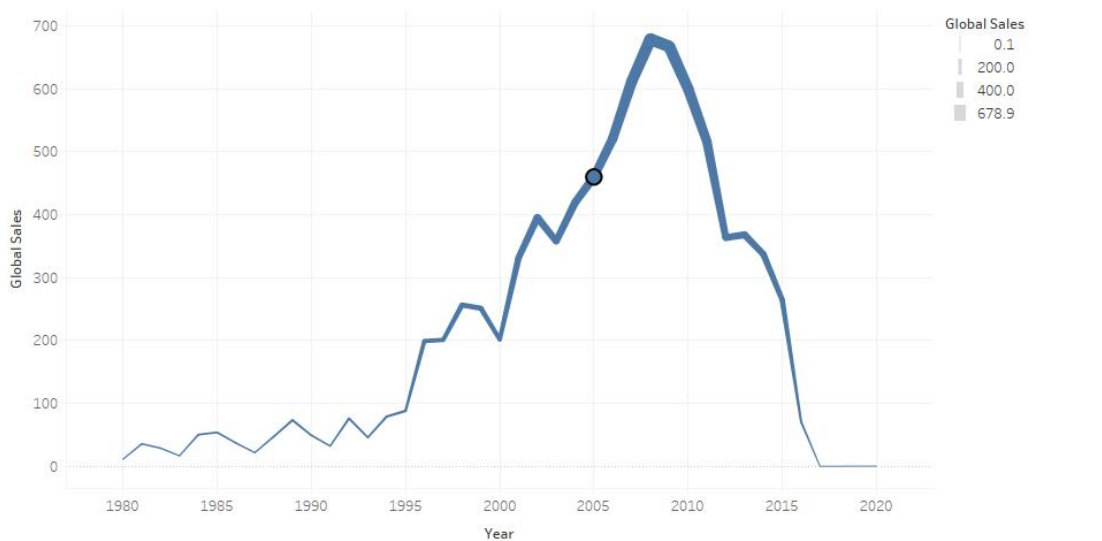
The screenshot displays a Windows 10 desktop environment. In the top-left corner, a File Explorer window shows the path "C:\Users\lenovo\Desktop\ Coventry University \Semester 2\Btg Data Ana & Data Visual...". Below it, a terminal window titled "Ubuntu [Running] - Oracle VM VirtualBox" shows a series of commands being executed to download Hadoop datasets from a public repository and save them locally. The commands include wget, curl, and mv. A large line chart titled "Genre Over Japan Sale" is visible in the background, showing sales trends over time (Year) for various music genres. The chart has a blue line representing the data. On the right side of the chart, there's a sidebar with various visualization options like bar charts, pie charts, etc. At the bottom of the screen, the Windows taskbar is visible with several open applications and the system clock showing 4:10 AM on 8/19/2023.

[illegible]

EU Sales	JP Sales	NA Sales	Other Sales	Global Sales
2,434	1,291	4,393	798	8,920

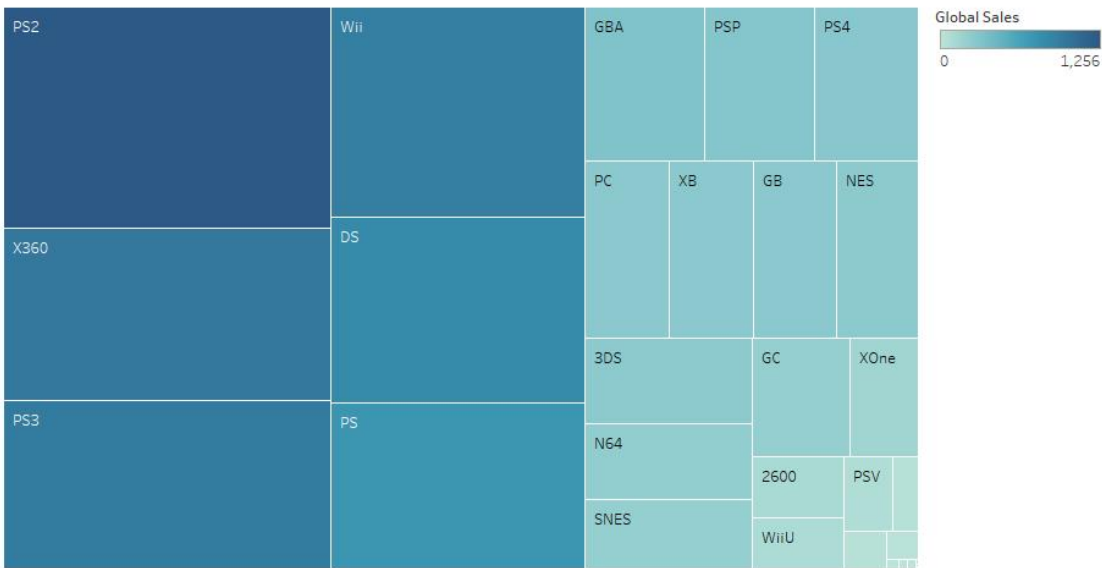
- 28 -

Global Sales over the year



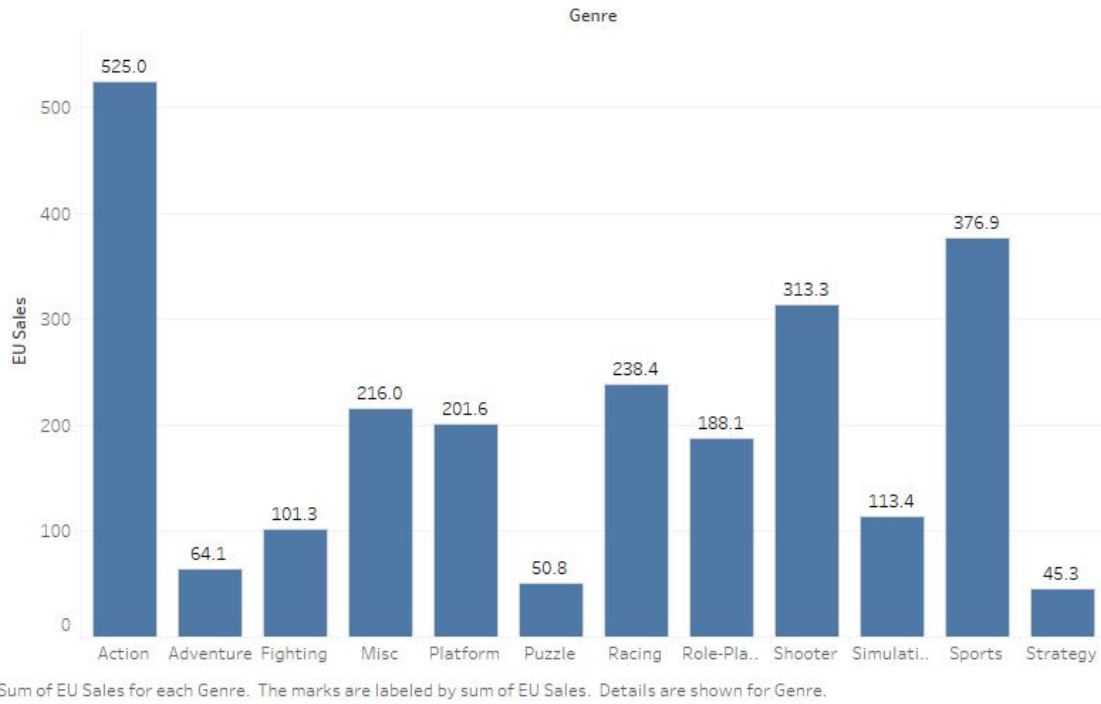
The trend of sum of Global Sales for Year. Size shows sum of Global Sales. The view is filtered on Year, which keeps non-Null values only.

Platform Global Sales

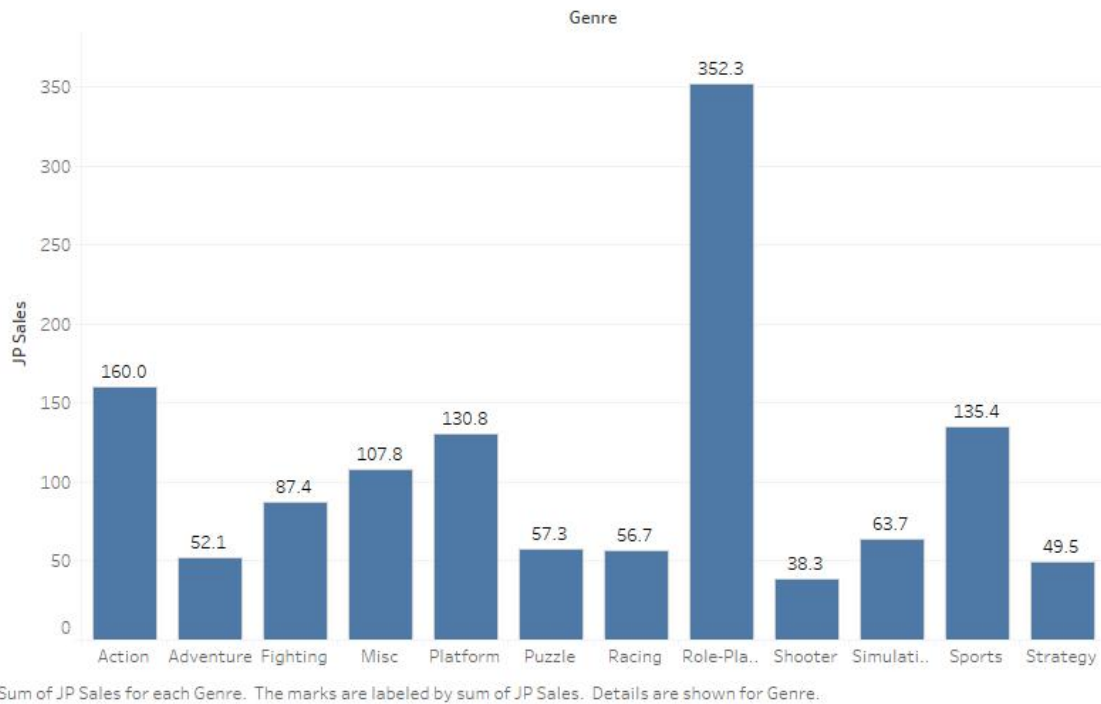


Platform. Color shows sum of Global Sales. Size shows sum of Global Sales. The marks are labeled by Platform.

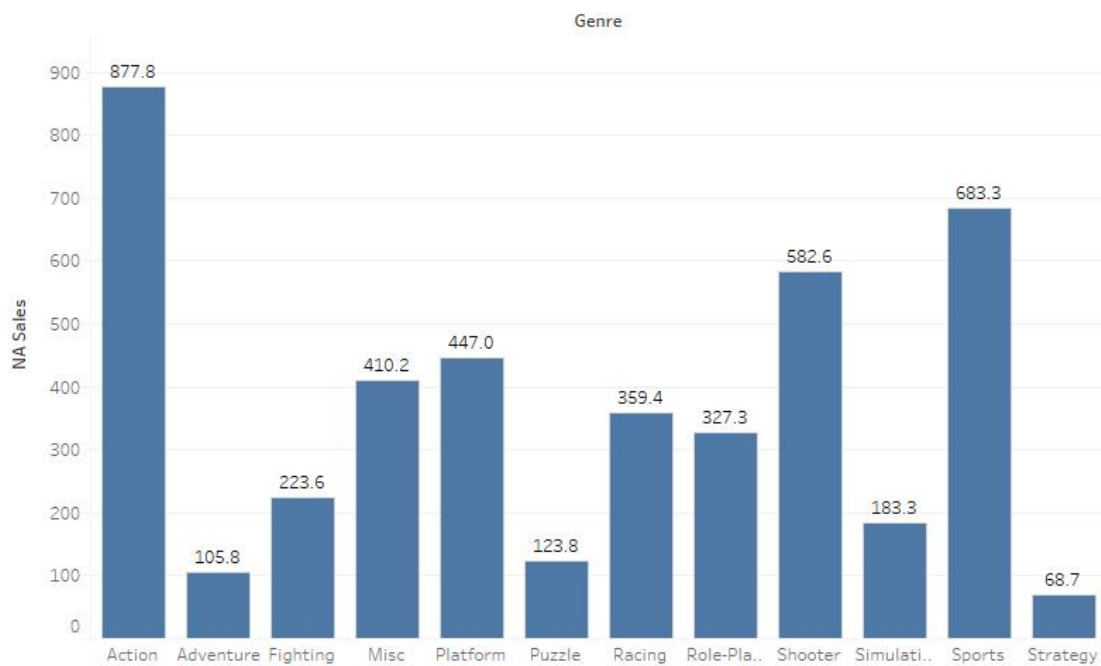
Genre Over Europe Sale



Genre Over Japan Sale

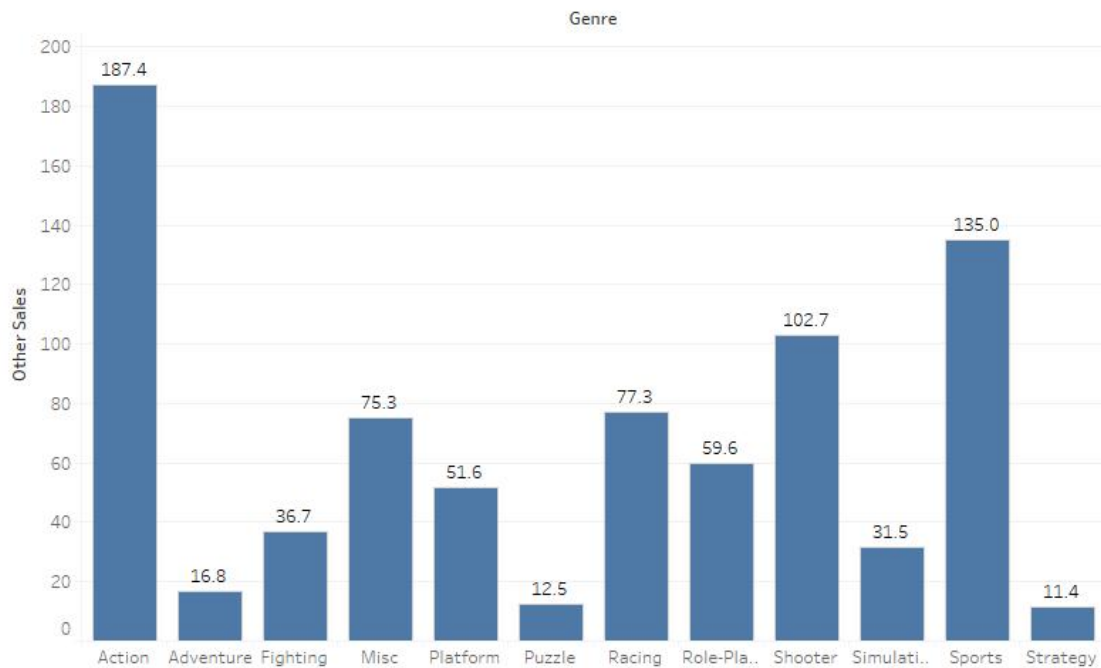


Genre Over North America Sale



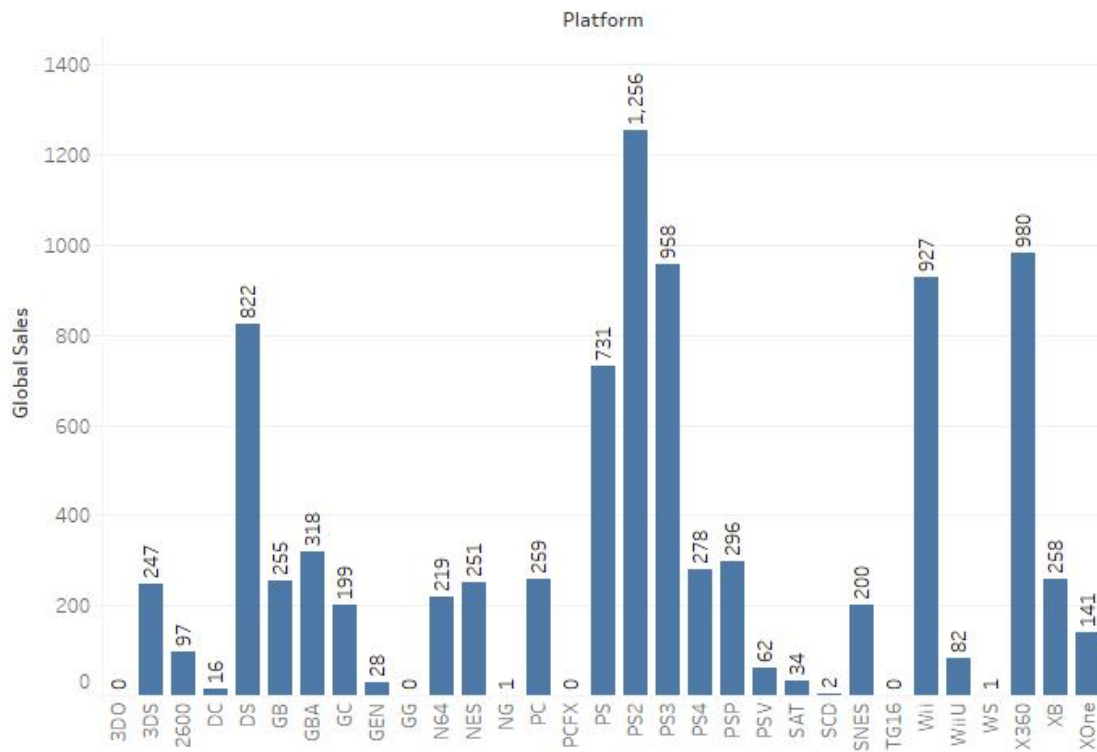
Sum of NA Sales for each Genre. The marks are labeled by sum of NA Sales. Details are shown for Genre.

Genre Over Other Region Sale



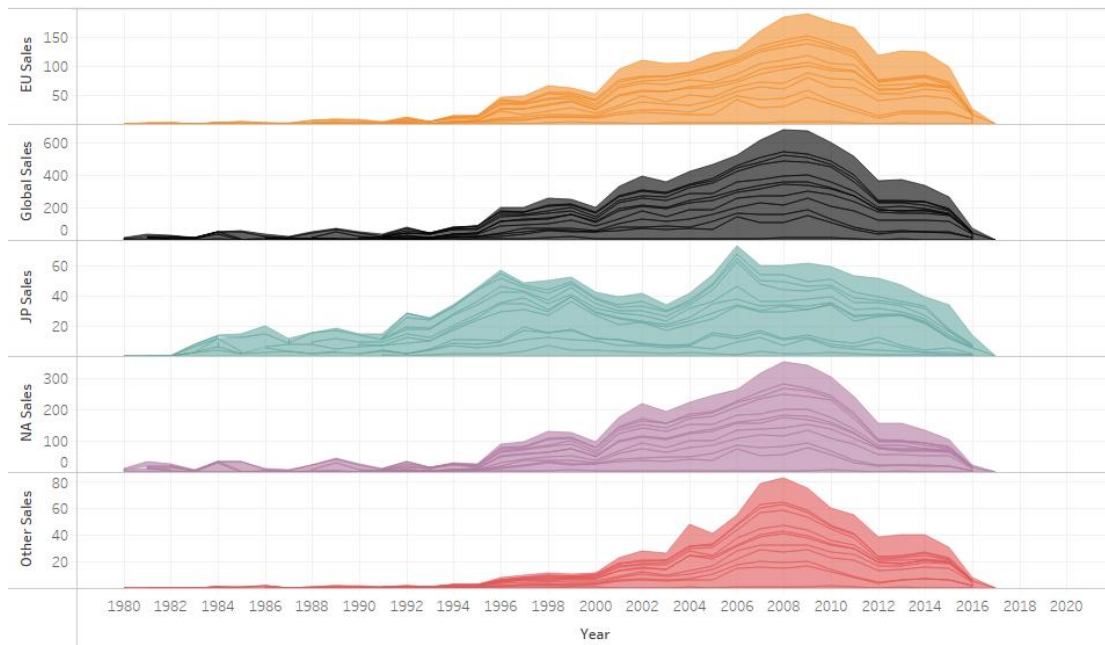
Sum of Other Sales for each Genre. The marks are labeled by sum of Other Sales. Details are shown for Genre.

Explore the relationship between Platform and Global sale with lables



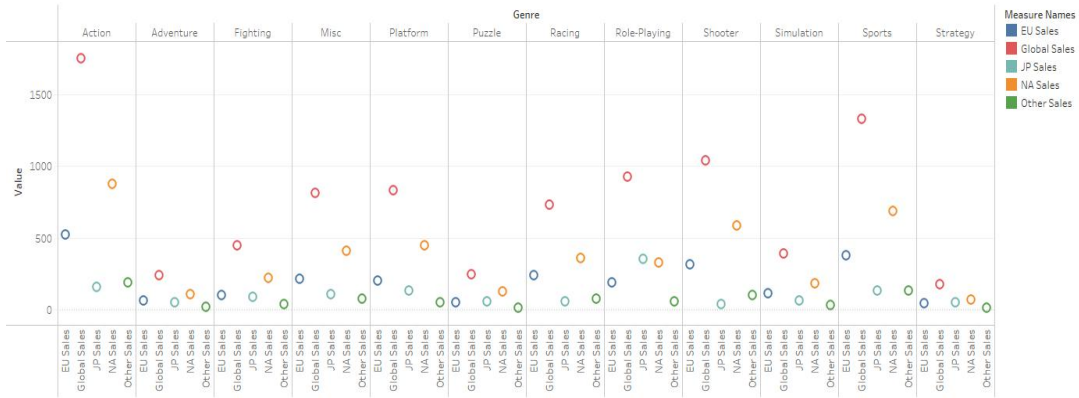
Sum of Global Sales for each Platform. The marks are labeled by sum of Global Sales.

Explore all region of Sales with Globally



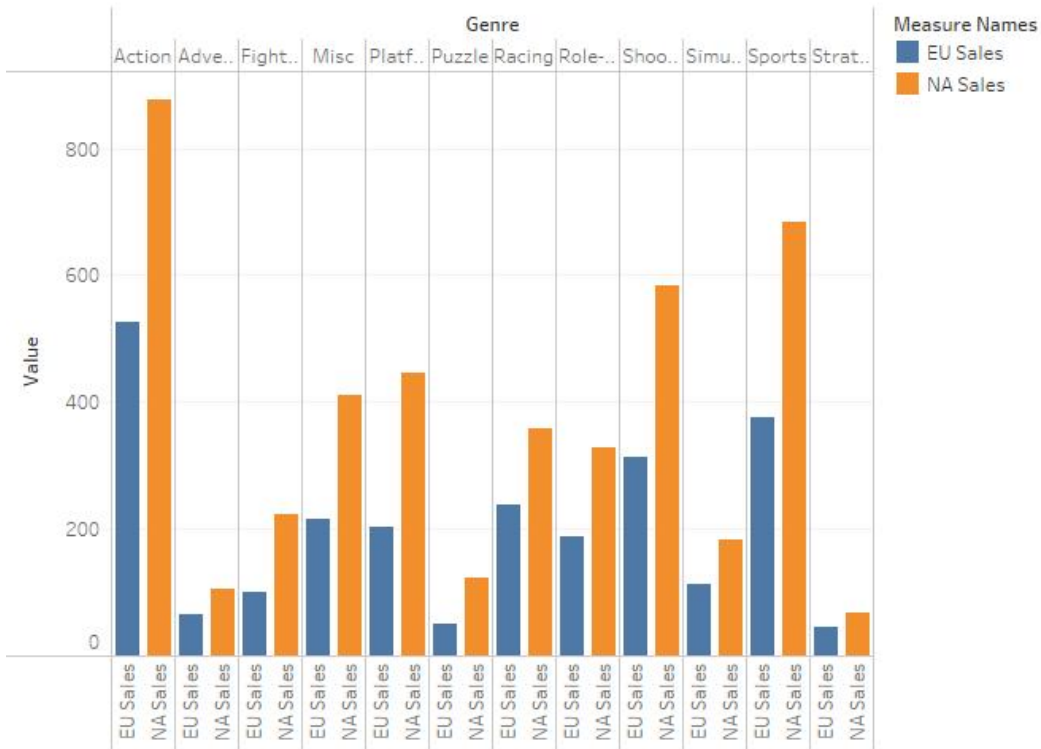
The plots of sum of EU Sales, sum of Global Sales, sum of JP Sales, sum of NA Sales and sum of Other Sales for Year. Details are shown for Genre.

Global Sales over all other Region Sales



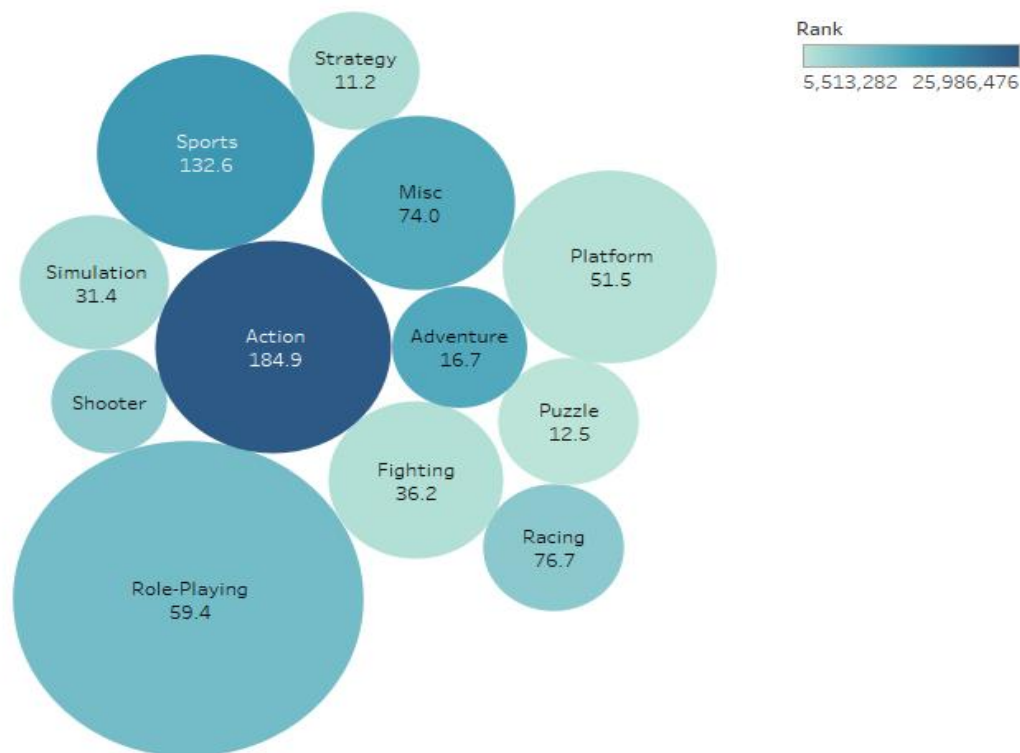
EU Sales, Global Sales, JP Sales, NA Sales and Other Sales for each Genre. Color shows details about EU Sales, Global Sales, JP Sales, NA Sales and Other Sales.

Europe and North America Sales Analysis



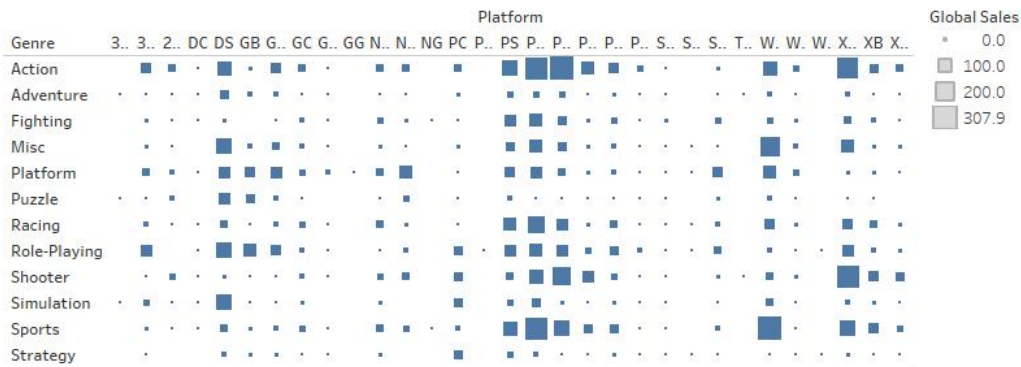
EU Sales and NA Sales for each Genre. Color shows details about EU Sales and NA Sales. The data is filtered on Year, which keeps all values.

Japan and other sale



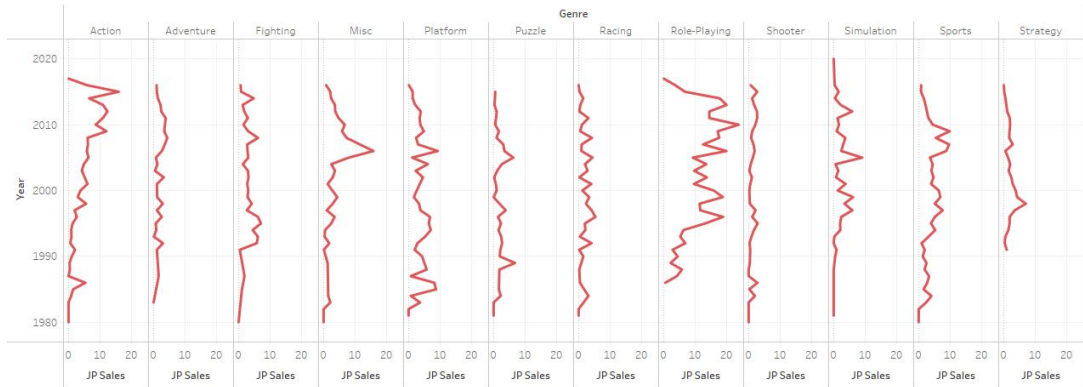
Genre and sum of Other Sales. Color shows sum of Rank. Size shows sum of JP Sales. The marks are labeled by Genre and sum of Other Sales. The data is filtered on Year, which ranges from 1980 to 2020. The view is filtered on Genre, which keeps 12 of 12 members.

Heatmap of Sales by Genre and Platform:



Sum of Global Sales (size) broken down by Platform vs. Genre.

Analyzing Japan Sales over Year by genre



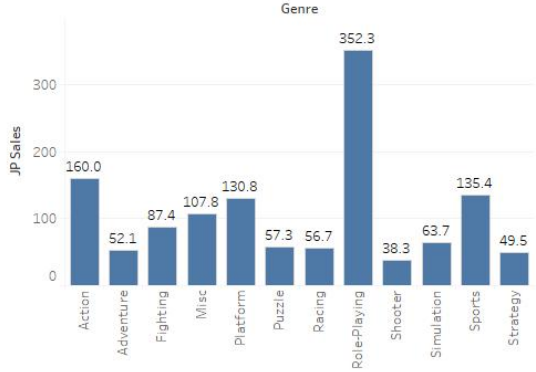
The trend of sum of JP Sales for Year broken down by Genre.

Tableau Dashboard

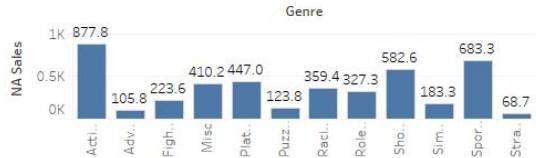
Text Table

EU Sales	JP Sales	NA Sales	Other Sales	Global Sales
2,434	1,291	4,393	798	8,920

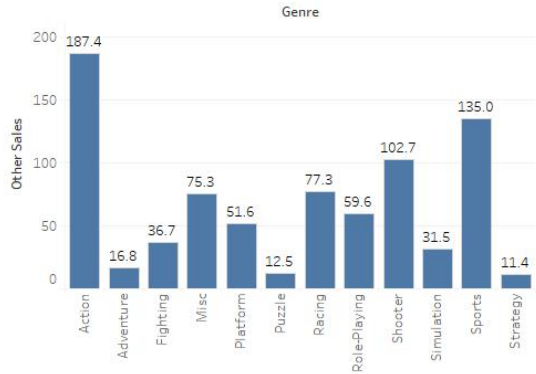
Genre Over Japan Sale



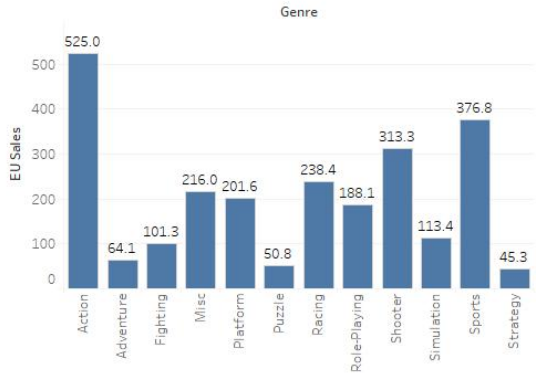
Genre Over North America Sale



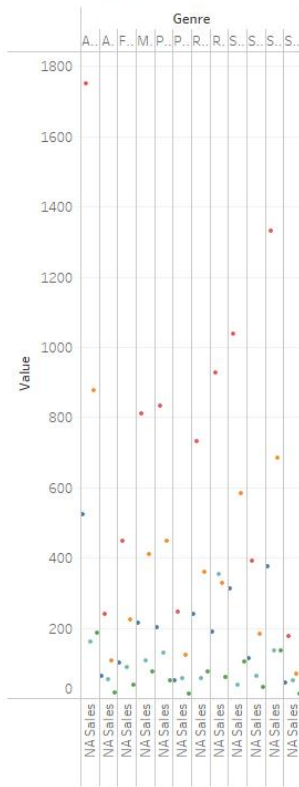
Genre Over Other Region Sale



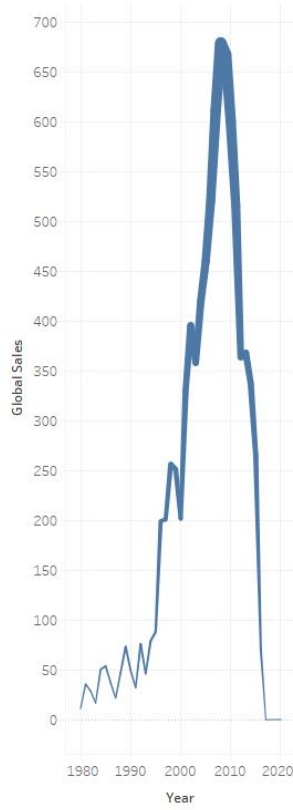
Genre Over Europe Sale



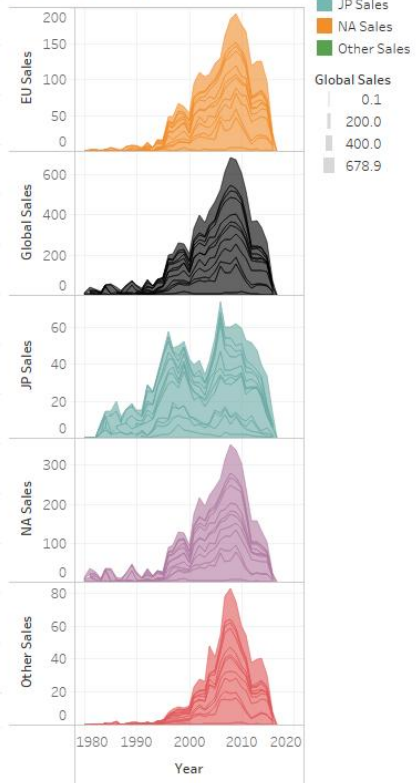
Global Sales over all other Region Sales



Global Sales over the year



Explore all region of Sales with Globally



Appendix II: Code in PySpark

All the code I had done in Google colab. Majority of portion I had done by myself but get some chunk of code from PySpark.com

```
!pip install pyspark
Collecting pyspark
  Downloading pyspark-3.4.1.tar.gz (310.8 MB)
    _____ 310.8/310.8 MB 2.4 MB/s eta
0:00:00
etadatat (setup.py) ... ent already satisfied: py4j==0.10.9.7 in
/usr/local/lib/python3.10/dist-packages (from pyspark) (0.10.9.7)
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) ... e=pyspark-3.4.1-py2.py3-
none-any.whl size=311285388
sha256=411efe0c6c3e1cb307a0807dd047820542656ab50bfdc73b94f76bafbb5b4
8db
  Stored in directory:
/root/.cache/pip/wheels/0d/77/a3/ff2f74cc9ab41f8f594dabf0579c2a7c6de
920d584206e0834
Successfully built pyspark
Installing collected packages: pyspark
Successfully installed pyspark-3.4.1
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from pyspark.sql import SparkSession
from pyspark.sql import functions as F
from pyspark.sql import *
from pyspark.sql import SparkSession
from pyspark.ml.stat import Correlation
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.regression import LinearRegression
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.evaluation import RegressionEvaluator,
BinaryClassificationEvaluator, MulticlassClassificationEvaluator
from pyspark.ml import Pipeline
# Start Spark Session
spark = SparkSession.builder \
    .master("local") \
    .appName("Video Game Sales Analysis") \
    .config('spark.ui.port', '4050') \
    .getOrCreate()
```

```
# Read the CSV file
df = spark.read.csv('/content/vgsales.csv', header=True,
inferSchema=True)

# Show first 20 rows
df.show(20)
```

Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
1	Wii Sports	Wii	2006	Sports	Nintendo	41.49	29.02	3.77	8.46	82.74
2	Super Mario Bros.	NES	1985	Platform	Nintendo	29.08	3.58	6.81	0.77	40.24
3	Mario Kart Wii	Wii	2008	Racing	Nintendo	15.85	12.88	3.79	3.31	35.82
4	Wii Sports Resort	Wii	2009	Sports	Nintendo	15.75	11.01	3.28	2.96	33.0
5	Pokemon Red/Pokem...	GB	1996	Role-Playing	Nintendo	11.27	8.89	10.22	1.0	31.37
6	Tetris	GB	1989	Puzzle	Nintendo	23.2	2.26	4.22	0.58	30.26
7	New Super Mario B...	DS	2006	Platform	Nintendo	11.38	9.23	6.5	2.9	30.01
8	Wii Play	Wii	2006	Misc	Nintendo	14.03	9.2	2.93	2.85	29.02
9	New Super Mario B...	Wii	2009	Platform	Nintendo	14.59	7.06	4.7	2.26	28.62
10	Duck Hunt	NES	1984	Shooter	Nintendo	26.93	0.63	0.28	0.47	28.31
11	Nintendogs	DS	2005	Simulation	Nintendo	9.07	11.0	1.93	2.75	24.76
12	Mario Kart DS	DS	2005	Racing	Nintendo	9.81	7.57	4.13	1.92	23.42
13	Pokemon Gold/Poke...	GB	1999	Role-Playing	Nintendo	9.0	6.18	7.2	0.71	23.1
14	Wii Fit	Wii	2007	Sports	Nintendo	8.94	8.03	3.6	2.15	22.72
15	Wii Fit Plus	Wii	2009	Sports	Nintendo	9.09	8.59	2.53	1.79	22.0
16	Kinect Adventures!	X360	2010	Misc	Microsoft Game St...	14.97	4.94	0.24	1.67	21.82
17	Grand Theft Auto V	PS3	2013	Action	Take-Two Interactive	7.01	9.27	0.97	4.14	21.4
18	Grand Theft Auto:...	PS2	2004	Action	Take-Two					


```

Interactive|    9.43|    0.4|    0.41|    10.57|    20.81|
|    19|    Super Mario World|    SNES|1990|    Platform|
Nintendo|   12.78|    3.75|    3.54|    0.55|    20.61|
|    20|Brain Age: Train ...|    DS|2005|    Misc|
Nintendo|    4.75|    9.26|    4.16|    2.05|    20.22|

```

```

+-----+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+-----+
only showing top 20 rows

```

```

df_1 = spark.read.option("header", "true").option("mode",
"DROPMALFORMED").csv("/content/vgsales.csv")
df.fillna(value=0).show()

```

```

+-----+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+-----+
|Rank|                Name|Platform|Year|                Genre|
Publisher|NA_Sales|EU_Sales|JP_Sales|Other_Sales|Global_Sales|
+-----+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+-----+
|    1|    Wii Sports|    Wii|2006|    Sports|
Nintendo|   41.49|   29.02|    3.77|    8.46|   82.74|
|    2|    Super Mario Bros.|    NES|1985|    Platform|
Nintendo|   29.08|    3.58|    6.81|    0.77|   40.24|
|    3|    Mario Kart Wii|    Wii|2008|    Racing|
Nintendo|   15.85|   12.88|    3.79|    3.31|   35.82|
|    4|    Wii Sports Resort|    Wii|2009|    Sports|
Nintendo|   15.75|   11.01|    3.28|    2.96|   33.0|
|    5|Pokemon Red/Pokem...|    GB|1996|Role-Playing|
Nintendo|   11.27|    8.89|   10.22|    1.0|   31.37|
|    6|    Tetris|    GB|1989|    Puzzle|
Nintendo|    23.2|    2.26|    4.22|    0.58|   30.26|
|    7|New Super Mario B...|    DS|2006|    Platform|
Nintendo|   11.38|    9.23|    6.5|    2.9|   30.01|
|    8|    Wii Play|    Wii|2006|    Misc|
Nintendo|   14.03|    9.2|    2.93|    2.85|   29.02|
|    9|New Super Mario B...|    Wii|2009|    Platform|
Nintendo|   14.59|    7.06|    4.7|    2.26|   28.62|
|   10|    Duck Hunt|    NES|1984|    Shooter|
Nintendo|   26.93|    0.63|    0.28|    0.47|   28.31|
|   11|    Nintendogs|    DS|2005|    Simulation|
Nintendo|    9.07|   11.0|    1.93|    2.75|   24.76|
|   12|    Mario Kart DS|    DS|2005|    Racing|
Nintendo|    9.81|    7.57|    4.13|    1.92|   23.42|
|   13|Pokemon Gold/Poke...|    GB|1999|Role-Playing|
Nintendo|    9.0|    6.18|    7.2|    0.71|   23.1|
|   14|    Wii Fit|    Wii|2007|    Sports|
Nintendo|    8.94|    8.03|    3.6|    2.15|   22.72|
|   15|    Wii Fit Plus|    Wii|2009|    Sports|

```


Nintendo	9.09	8.59	2.53	1.79	22.0
16 Kinect Adventures!			X360 2010		Misc Microsoft Game
St...	14.97	4.94	0.24	1.67	21.82
17 Grand Theft Auto V			PS3 2013		Action Take-Two
Interactive	7.01	9.27	0.97	4.14	21.4
18 Grand Theft Auto:...			PS2 2004		Action Take-Two
Interactive	9.43	0.4	0.41	10.57	20.81
19 Super Mario World			SNES 1990		Platform
Nintendo	12.78	3.75	3.54	0.55	20.61
20 Brain Age: Train ...			DS 2005		Misc
Nintendo	4.75	9.26	4.16	2.05	20.22

```

+---+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+-----+

```

only showing top 20 rows

```
df.fillna(value=-99, subset=["NA_Sales", "EU_Sales", "JP_Sales",
"Other_Sales", "Global_Sales"]).show()
```

```

+---+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+-----+
|Rank|                               Name|Platform|Year|                               Genre|
Publisher|NA_Sales|EU_Sales|JP_Sales|Other_Sales|Global_Sales|
+---+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+-----+
| 1|                               Wii Sports|Wii|2006|                               Sports|
Nintendo| 41.49| 29.02| 3.77| 8.46| 82.74|
| 2| Super Mario Bros.|NES|1985|                               Platform|
Nintendo| 29.08| 3.58| 6.81| 0.77| 40.24|
| 3| Mario Kart Wii|Wii|2008|                               Racing|
Nintendo| 15.85| 12.88| 3.79| 3.31| 35.82|
| 4| Wii Sports Resort|Wii|2009|                               Sports|
Nintendo| 15.75| 11.01| 3.28| 2.96| 33.0|
| 5|Pokemon Red/Pokem...|GB|1996|Role-Playing|
Nintendo| 11.27| 8.89| 10.22| 1.0| 31.37|
| 6| Tetris|GB|1989|                               Puzzle|
Nintendo| 23.2| 2.26| 4.22| 0.58| 30.26|
| 7|New Super Mario B...|DS|2006|                               Platform|
Nintendo| 11.38| 9.23| 6.5| 2.9| 30.01|
| 8| Wii Play|Wii|2006|                               Misc|
Nintendo| 14.03| 9.2| 2.93| 2.85| 29.02|
| 9|New Super Mario B...|Wii|2009|                               Platform|
Nintendo| 14.59| 7.06| 4.7| 2.26| 28.62|
| 10| Duck Hunt|NES|1984|                               Shooter|
Nintendo| 26.93| 0.63| 0.28| 0.47| 28.31|
| 11| NintendoDogs|DS|2005|                               Simulation|
Nintendo| 9.07| 11.0| 1.93| 2.75| 24.76|
| 12| Mario Kart DS|DS|2005|                               Racing|
Nintendo| 9.81| 7.57| 4.13| 1.92| 23.42|

```

13	Pokemon Gold/Poke...	GB	1999	Role-Playing
Nintendo	9.0	6.18	7.2	0.71
14	Wii Fit	Wii	2007	Sports
Nintendo	8.94	8.03	3.6	2.15
15	Wii Fit Plus	Wii	2009	Sports
Nintendo	9.09	8.59	2.53	1.79
16	Kinect Adventures!	X360	2010	Misc
St...	14.97	4.94	0.24	1.67
17	Grand Theft Auto V	PS3	2013	Action
Interactive	7.01	9.27	0.97	4.14
18	Grand Theft Auto:...	PS2	2004	Action
Interactive	9.43	0.4	0.41	10.57
19	Super Mario World	SNES	1990	Platform
Nintendo	12.78	3.75	3.54	0.55
20	Brain Age: Train ...	DS	2005	Misc
Nintendo	4.75	9.26	4.16	2.05

```

+---+-----+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+-----+

```

only showing top 20 rows

Calculate Descriptive Statistics

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

Calculate Mean

```
mean_values = df.select(*[F.mean(col).alias(f'mean_{col}') for col
in quantitative_columns]).collect()[0]
```

Calculate Median

```
median_values = df.select(*[F.expr(f'percentile_approx({col},
0.5)').alias(f'median_{col}') for col in
quantitative_columns]).collect()[0]
```

Calculate Standard Deviation

```
stddev_values = df.select(*[F.stddev(col).alias(f'stddev_{col}') for
col in quantitative_columns]).collect()[0]
```

Display Descriptive Statistics

```
for col in quantitative_columns:
    print(f"{col} - Mean: {mean_values[f'mean_{col}']:.2f}, Median:
{median_values[f'median_{col}']:.2f}, Std Dev:
{stddev_values[f'stddev_{col}']:.2f}")
NA_Sales - Mean: 0.26, Median: 0.08, Std Dev: 0.82
EU_Sales - Mean: 0.15, Median: 0.02, Std Dev: 0.51
JP_Sales - Mean: 0.08, Median: 0.00, Std Dev: 0.31
Other_Sales - Mean: 0.05, Median: 0.01, Std Dev: 0.19
Global_Sales - Mean: 0.54, Median: 0.17, Std Dev: 1.56
```

```

# Data Exploration
# Total number of rows and columns
print('Rows:', df.count())
print('Columns:', len(df.columns))
Rows: 16598
Columns: 11
# Creating a Pandas DataFrame for plotting
pandas_df = df.toPandas()

# Plotting histograms using Seaborn in the same row
plt.figure(figsize=(20, 6))

plt.subplot(1, 5, 1)
sns.histplot(pandas_df['Global_Sales'], bins=20, kde=True)
plt.title("Global Sales")

plt.subplot(1, 5, 2)
sns.histplot(pandas_df['EU_Sales'], bins=20, kde=True)
plt.title("EU Sales")

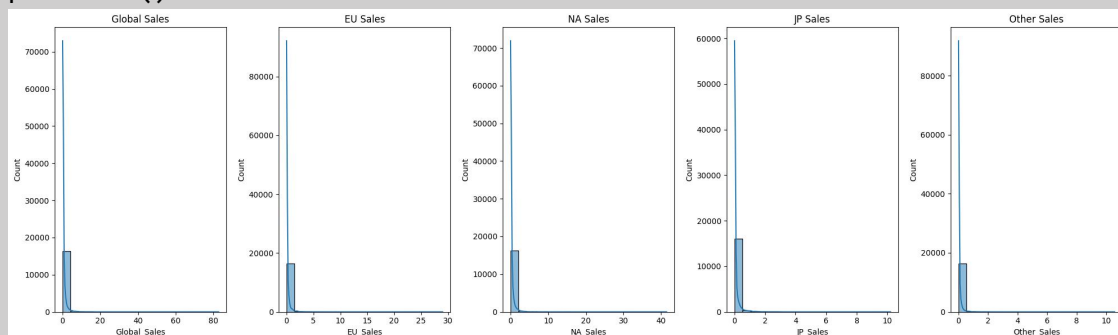
plt.subplot(1, 5, 3)
sns.histplot(pandas_df['NA_Sales'], bins=20, kde=True)
plt.title("NA Sales")

plt.subplot(1, 5, 4)
sns.histplot(pandas_df['JP_Sales'], bins=20, kde=True)
plt.title("JP Sales")

plt.subplot(1, 5, 5)
sns.histplot(pandas_df['Other_Sales'], bins=20, kde=True)
plt.title("Other Sales")

plt.tight_layout()
plt.show()

```



```

# Scatter plots using Seaborn in the same row
plt.figure(figsize=(15, 5))

# Global Sales vs Year

```

```

plt.subplot(1, 5, 1)
sns.scatterplot(x='Global_Sales',
data=pandas_df.sort_values('Year'))
plt.title("Global Sales vs Year")
plt.xlabel("Global Sales")
plt.ylabel("Year")

# EU Sales vs Year
plt.subplot(1, 5, 2)
sns.scatterplot(x='EU_Sales',
data=pandas_df.sort_values('Year'))
plt.title("EU Sales vs Year")
plt.xlabel("EU Sales")
plt.ylabel("Year")

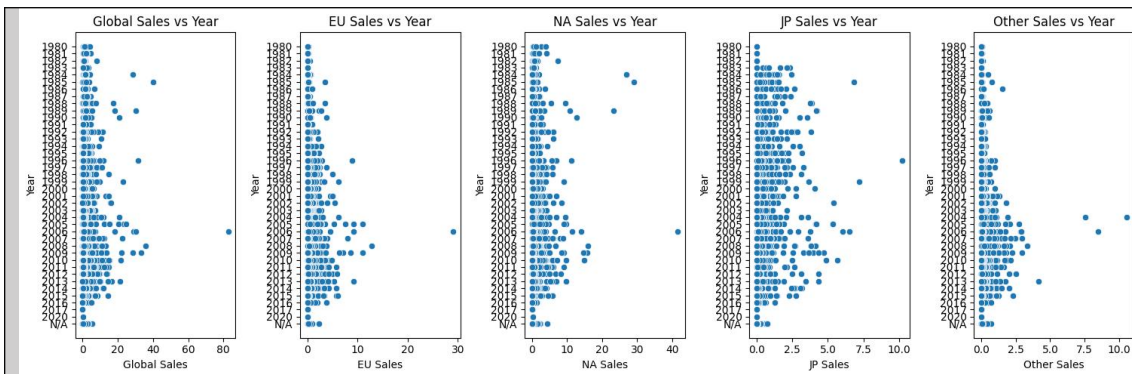
# NA Sales vs Year
plt.subplot(1, 5, 3)
sns.scatterplot(x='NA_Sales',
data=pandas_df.sort_values('Year'))
plt.title("NA Sales vs Year")
plt.xlabel("NA Sales")
plt.ylabel("Year")

# JP Sales vs Year
plt.subplot(1, 5, 4)
sns.scatterplot(x='JP_Sales',
data=pandas_df.sort_values('Year'))
plt.title("JP Sales vs Year")
plt.xlabel("JP Sales")
plt.ylabel("Year")

# Other Sales vs Year
plt.subplot(1, 5, 5)
sns.scatterplot(x='Other_Sales',
data=pandas_df.sort_values('Year'))
plt.title("Other Sales vs Year")
plt.xlabel("Other Sales")
plt.ylabel("Year")

plt.tight_layout()
plt.show()

```



Box plot using Seaborn

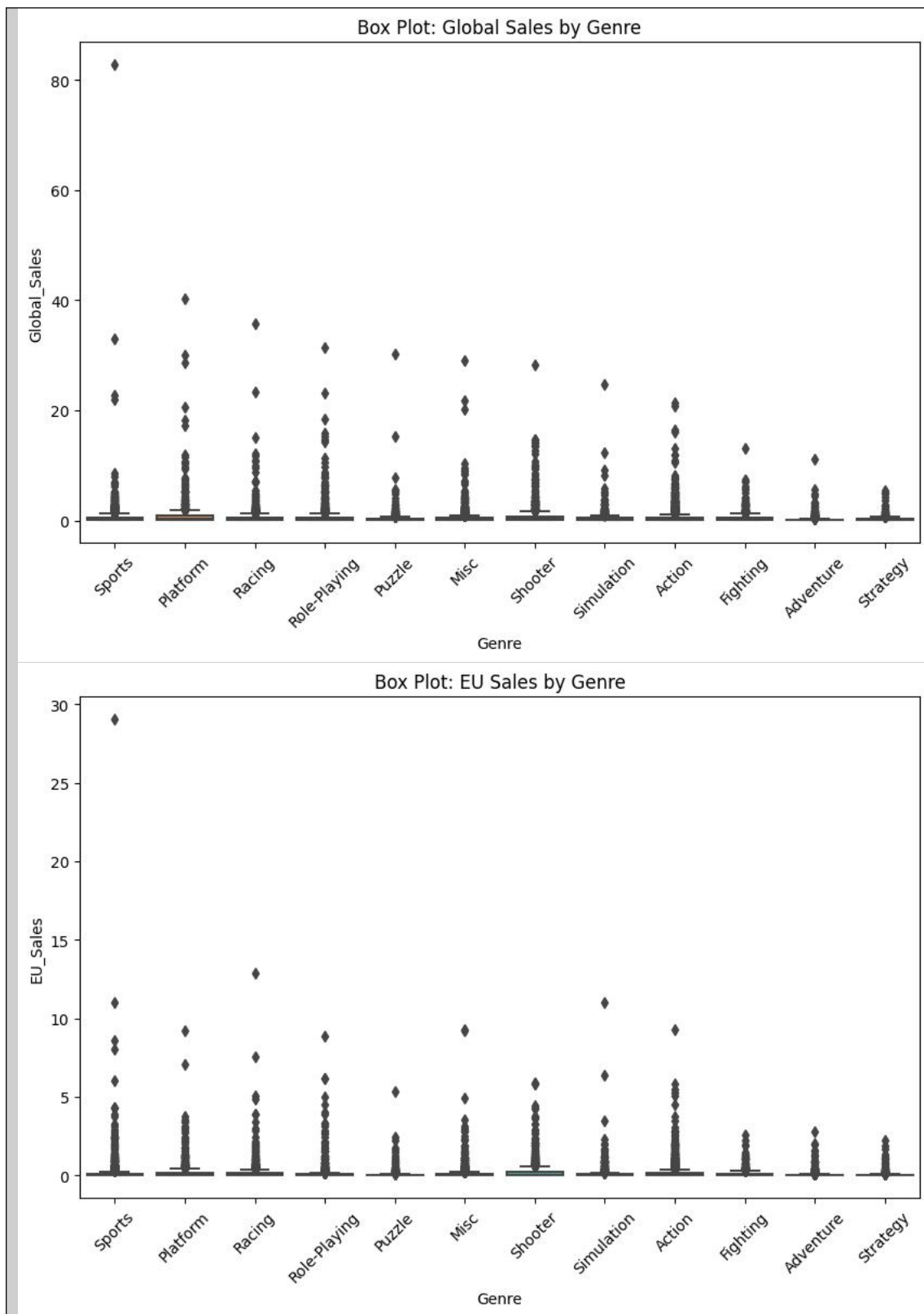
```
plt.figure(figsize=(10, 6))
sns.boxplot(x='Genre', y='Global_Sales', data=pandas_df)
plt.xticks(rotation=45)
plt.title("Box Plot: Global Sales by Genre")
plt.show()
```

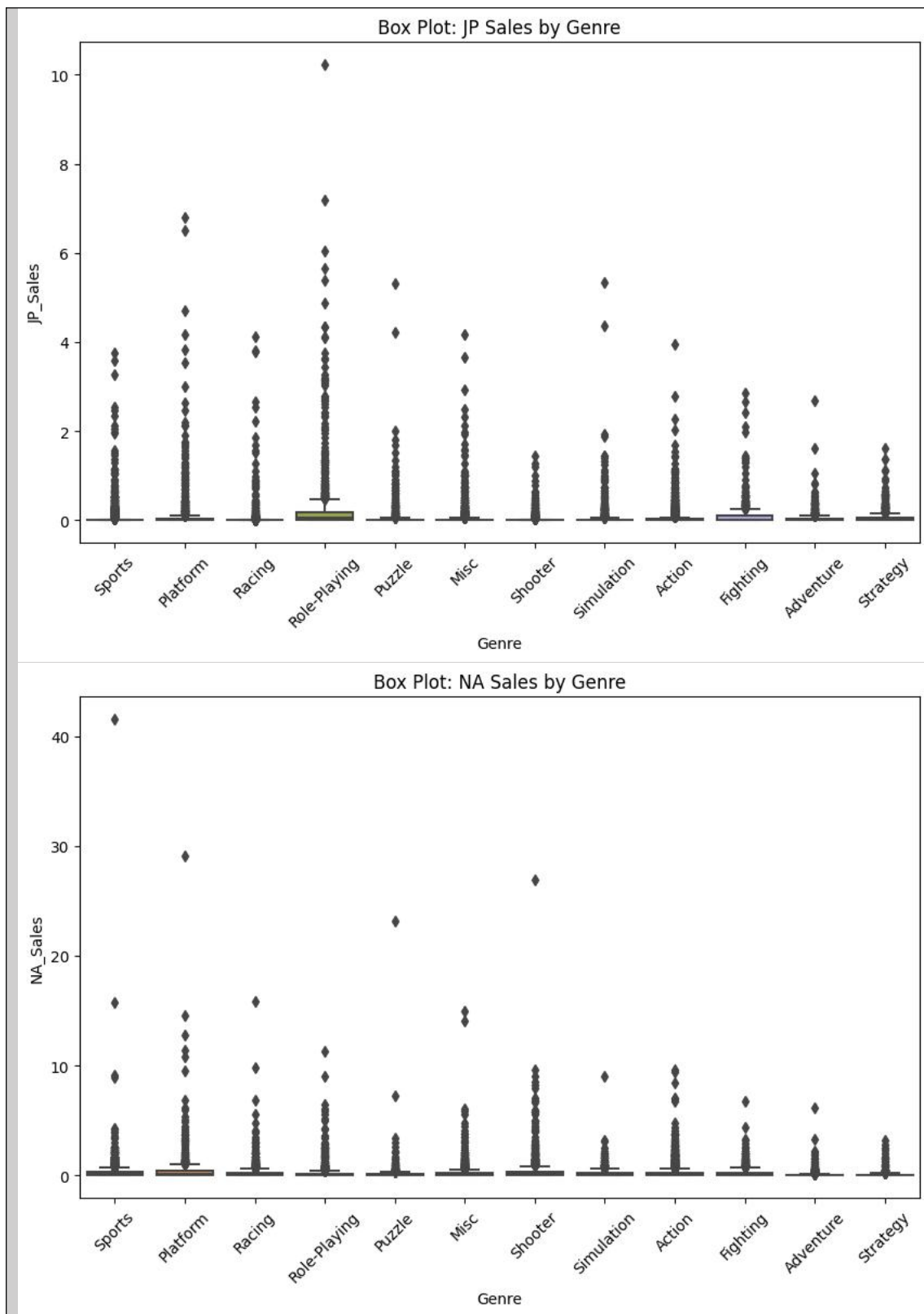
```
plt.figure(figsize=(10, 6))
sns.boxplot(x='Genre', y='EU_Sales', data=pandas_df)
plt.xticks(rotation=45)
plt.title("Box Plot: EU Sales by Genre")
plt.show()
```

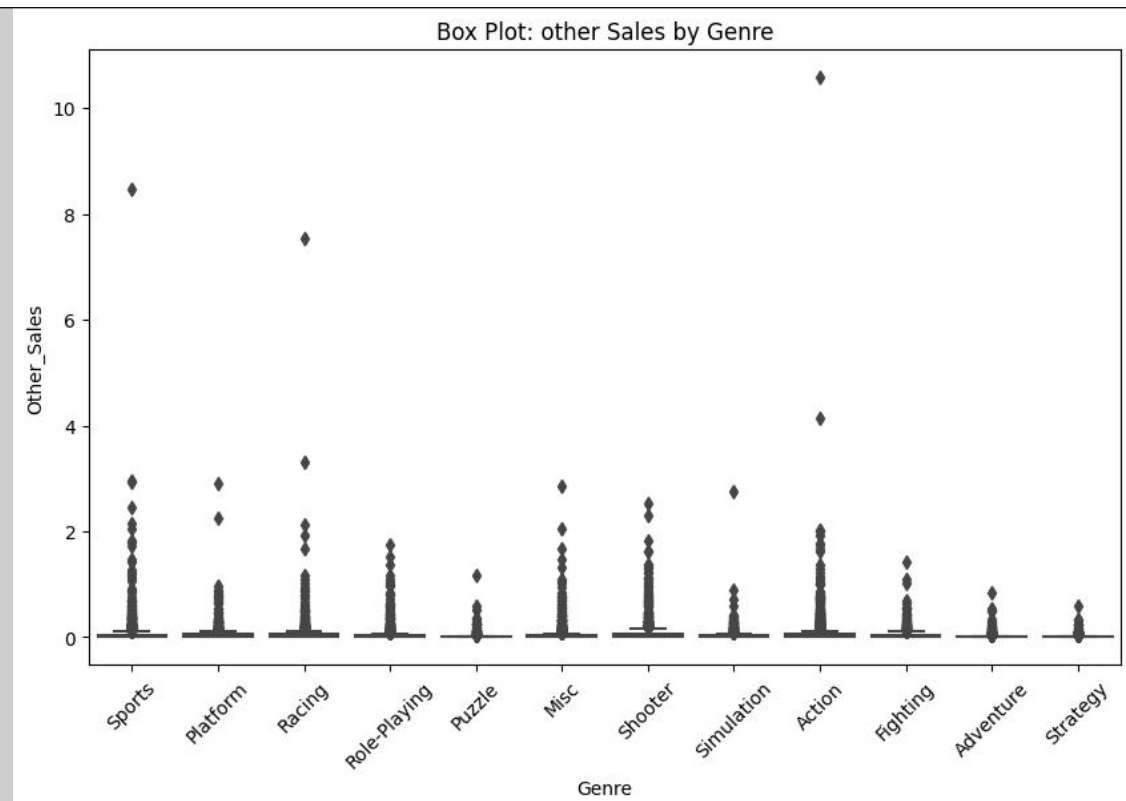
```
plt.figure(figsize=(10, 6))
sns.boxplot(x='Genre', y='JP_Sales', data=pandas_df)
plt.xticks(rotation=45)
plt.title("Box Plot: JP Sales by Genre")
plt.show()
```

```
plt.figure(figsize=(10, 6))
sns.boxplot(x='Genre', y='NA_Sales', data=pandas_df)
plt.xticks(rotation=45)
plt.title("Box Plot: NA Sales by Genre")
plt.show()
```

```
plt.figure(figsize=(10, 6))
sns.boxplot(x='Genre', y='Other_Sales', data=pandas_df)
plt.xticks(rotation=45)
plt.title("Box Plot: other Sales by Genre")
plt.show()
```







Creating bar plots using Matplotlib

```
plt.figure(figsize=(10, 6))
sns.barplot(x='Platform', y='EU_Sales', data=pandas_df)
plt.xticks(rotation=45)
plt.title("EU sales by Platform")
plt.show()
```

```
plt.figure(figsize=(10, 6))
sns.barplot(x='Platform', y='Global_Sales', data=pandas_df)
plt.xticks(rotation=45)
plt.title("Global sales by Platform")
plt.show()
```

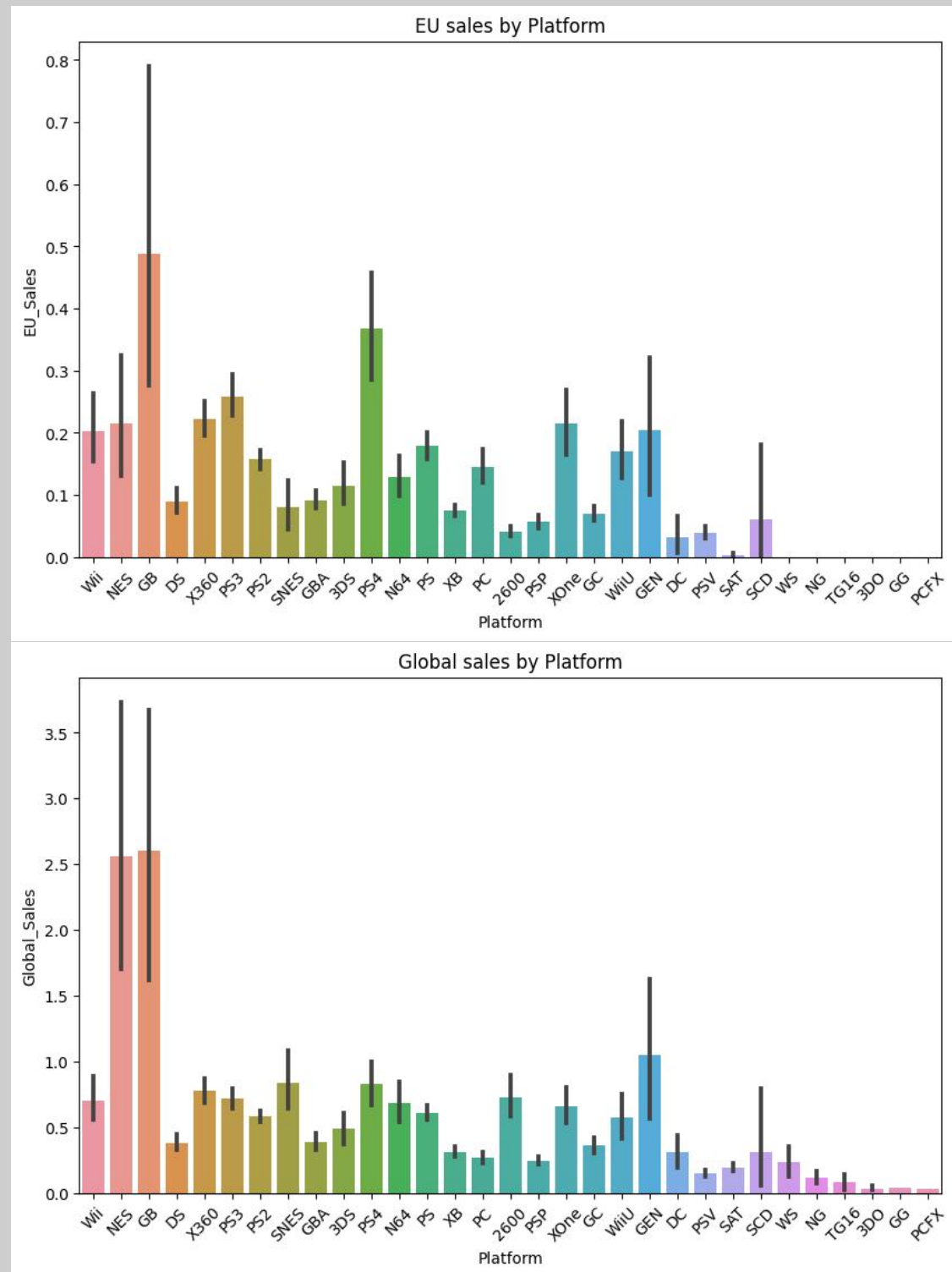
```
plt.figure(figsize=(10, 6))
sns.barplot(x='Platform', y='NA_Sales', data=pandas_df)
plt.xticks(rotation=45)
plt.title("NA sales by Platform")
plt.show()
```

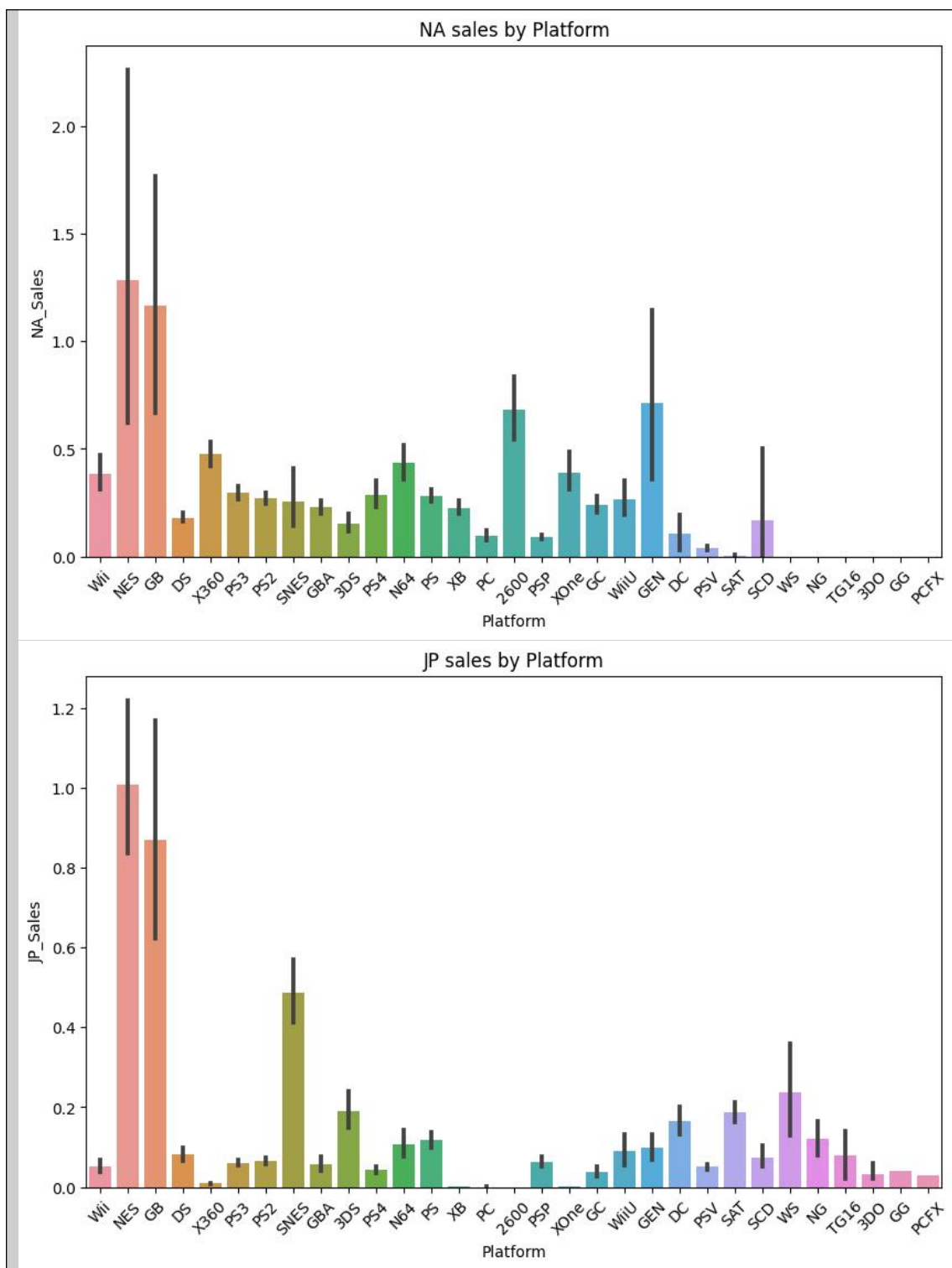
```
plt.figure(figsize=(10, 6))
sns.barplot(x='Platform', y='JP_Sales', data=pandas_df)
plt.xticks(rotation=45)
plt.title("JP sales by Platform")
plt.show()
```

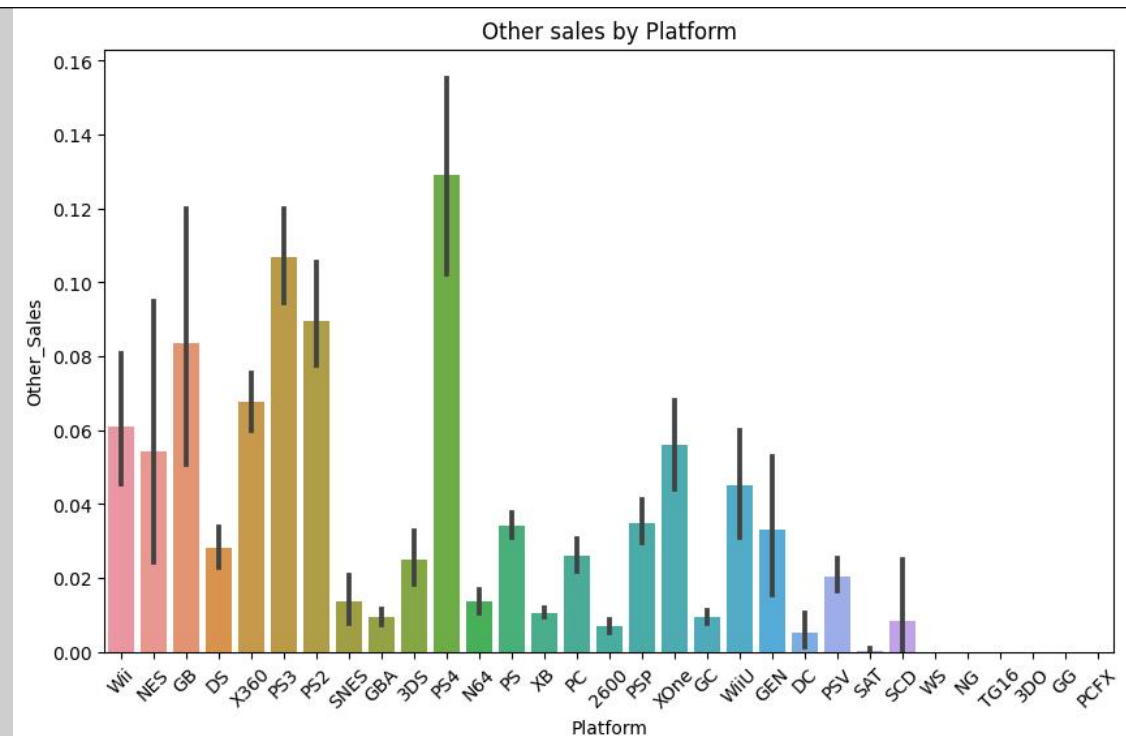
```
plt.figure(figsize=(10, 6))
```



```
sns.barplot(x='Platform', y='Other_Sales', data=pandas_df)
plt.xticks(rotation=45)
plt.title("Other sales by Platform")
plt.show()
```







```

from pyspark.ml.stat import Correlation
from pyspark.ml.feature import VectorAssembler

# Select relevant columns for correlation analysis
correlation_columns = ['NA_Sales', 'EU_Sales', 'JP_Sales',
'Other_Sales', 'Global_Sales']

# Create a VectorAssembler to combine the columns into a single
feature column
assembler = VectorAssembler(inputCols=correlation_columns,
outputCol="features")
data_for_correlation = assembler.transform(df).select("features")

# Calculate correlation matrix
correlation_matrix = Correlation.corr(data_for_correlation,
"features").collect()[0][0]

# Display the correlation matrix
print("Correlation Matrix:")
print(correlation_matrix)
Correlation Matrix:
DenseMatrix([[1.0, 0.76772675, 0.44978741, 0.63473727,
0.94104736],
[0.76772675, 1.0, 0.43558445, 0.72638489,
0.90283581],
[0.44978741, 0.43558445, 1.0, 0.29018625,
0.61181552],
[0.63473727, 0.72638489, 0.29018625, 1.0,
0.89981552],
[0.94104736, 0.90283581, 0.61181552, 0.89981552,
1.0]])

```

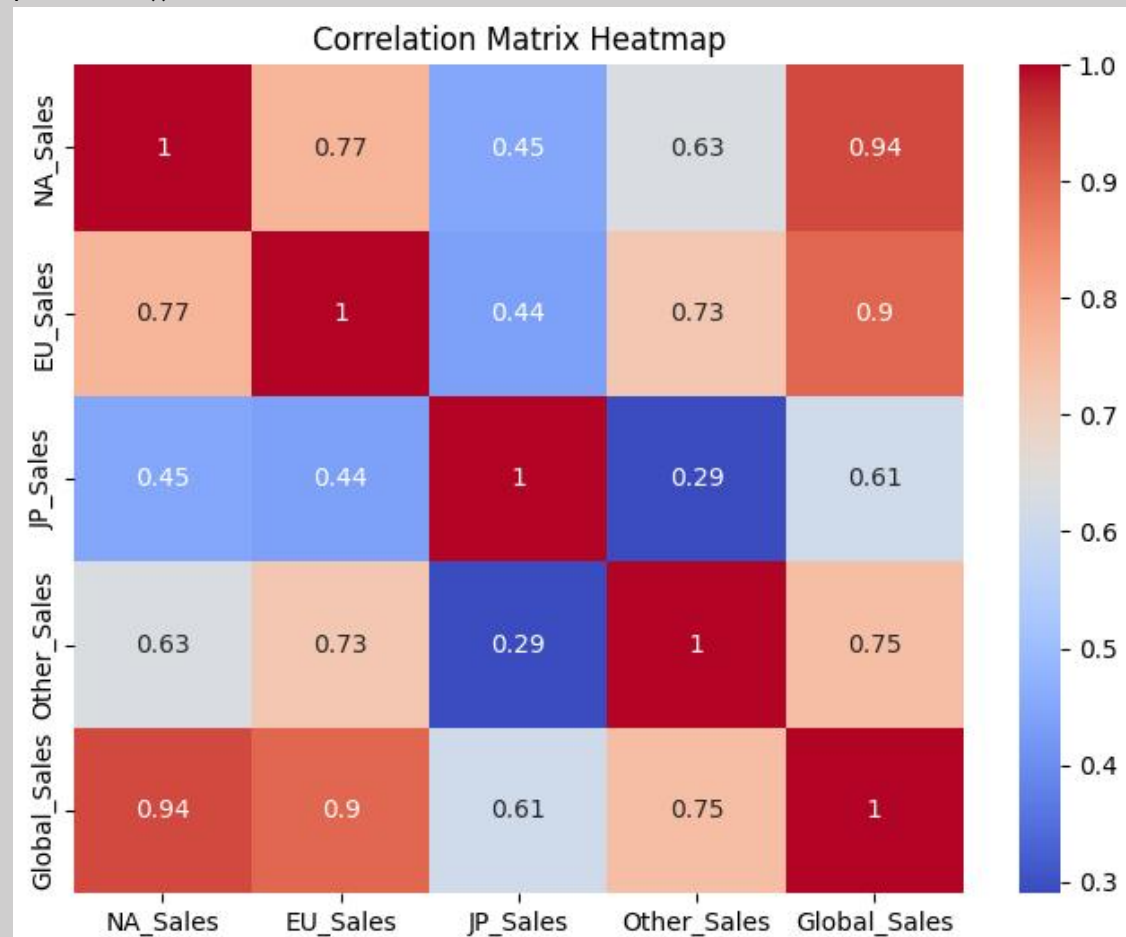
```

0.74833085],
      [0.94104736,  0.90283581,  0.61181552,  0.74833085,
1.      ]])
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# Convert the correlation matrix from a Row object to a NumPy array
correlation_matrix = np.array(correlation_matrix.toArray())

# Visualize the correlation matrix using a heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
            xticklabels=correlation_columns, yticklabels=correlation_columns)
plt.title("Correlation Matrix Heatmap")
plt.show()

```



```

# Unique Values in 'genre' Column
df.select('Genre').distinct().show()
+-----+
|   Genre|
+-----+
| Adventure|
|   Sports|

```

```

|      Racing|
|Role-Playing|
|      Shooter|
|      Misc|
|    Platform|
|    Puzzle|
|    Fighting|
|    Action|
|    Strategy|
|  Simulation|
+-----+

```

Distribution of values inside column 'genre'

```
df.groupBy('genre').count().show()
```

```

+-----+-----+
|      genre|count|
+-----+-----+
|  Adventure| 1286|
|    Sports| 2346|
|    Racing| 1249|
|Role-Playing| 1488|
|    Shooter| 1310|
|    Misc| 1739|
|  Platform|   886|
|    Puzzle|   582|
|  Fighting|   848|
|    Action| 3316|
|  Strategy|   681|
|  Simulation|  867|
+-----+-----+

```

Unique Values in 'Platform' Column

```
df.select('Platform').distinct().show()
```

```

+-----+
|Platform|
+-----+
|    3DO|
|    PC|
|   PS3|
|   NES|
|    PS|
|    DC|
|   GEN|
|   PS2|
|   3DS|
|  PCFX|
|    GG|

```

```

|    WiiU |
|    SNES |
|     GB |
|    SCD |
|    N64 |
|    PS4 |
|    PSP |
|   2600 |
|   XOne |
+-----+
only showing top 20 rows

# Distribution of values inside column 'Platform'
df.groupby('Platform').count().show()
+-----+-----+
|Platform|count|
+-----+-----+
|    3DO |    3 |
|     PC |   960 |
|    PS3 |  1329 |
|    NES |   98 |
|     PS |  1196 |
|     DC |   52 |
|    GEN |   27 |
|    PS2 |  2161 |
|    3DS |   509 |
|  PCFX |    1 |
|     GG |    1 |
|   WiiU |   143 |
|    SNES |   239 |
|     GB |   98 |
|    SCD |    6 |
|    N64 |   319 |
|    PS4 |   336 |
|    PSP |  1213 |
|   2600 |   133 |
|   XOne |   213 |
+-----+-----+
only showing top 20 rows

# Question 1: Top 3 Video Games in sports that Sell the most Globally
df.select('Name', 'Global_Sales').where(df.Genre == 'Sports').orderBy('Global_Sales', ascending=False).show(3)
+-----+-----+
|          Name|Global_Sales|
+-----+-----+

```

```

|      Wii Sports|      82.74|
|Wii Sports Resort|      33.0|
|      Wii Fit|      22.72|
+-----+
only showing top 3 rows

# Question 2: Top 5 Publisher in Racing who Have highest sale in Europe
df.select('Publisher', 'EU_Sales').filter(df.Genre == 'Racing').orderBy('EU_Sales', ascending=False).show(5)
+-----+
|      Publisher|EU_Sales|
+-----+
|      Nintendo|    12.88|
|      Nintendo|     7.57|
|Sony Computer Ent...|     5.09|
|Sony Computer Ent...|     4.88|
|      Nintendo|     3.91|
+-----+
only showing top 5 rows

# Question 3: Name 5 Lowest Sales Publisher Globally
df.select('Publisher', 'Global_Sales').orderBy('Global_Sales', ascending=True).show(5)
+-----+
|      Publisher|Global_Sales|
+-----+
|      Touchstone|      0.01|
|      Nobilis|      0.01|
|Nippon Ichi Software|      0.01|
|  Namco Bandai Games|      0.01|
|      Electronic Arts|      0.01|
+-----+
only showing top 5 rows

# Question 4: Average sale for various genres
df.groupBy('Genre').agg(F.round(F.mean('Global_Sales'), 2).alias('average_sales')).show()
+-----+
|      Genre|average_sales|
+-----+
|  Adventure|      0.19|
|    Sports|      0.57|
|    Racing|      0.59|
|Role-Playing|      0.62|
|    Shooter|      0.79|
|      Misc|      0.47|

```



```

| Platform|      0.94|
| Puzzle|      0.42|
| Fighting|      0.53|
| Action|      0.53|
| Strategy|      0.26|
| Simulation|      0.45|
+-----+

```

Question 5: List of Name with sale in 2015

```

df.select('Name', 'Publisher', 'Global_Sales').where(df.Year ==
'2015').orderBy('Global_Sales', ascending=False).show()

```

```

+-----+-----+-----+
| Name| Publisher| Global_Sales|
+-----+-----+-----+
| Call of Duty: Bla...| Activision| 14.24|
| FIFA 16| Electronic Arts| 8.49|
| Star Wars Battlef...| Electronic Arts| 7.67|
| Call of Duty: Bla...| Activision| 7.3|
| Fallout 4| Bethesda Softworks| 6.96|
| Splatoon| Nintendo| 4.57|
| Uncharted: The Na...| Sony Computer Ent...| 4.47|
| Halo 5: Guardians| Microsoft Game St...| 4.26|
| Fallout 4| Bethesda Softworks| 4.09|
| NBA 2K16| Take-Two Interactive| 3.85|
| Batman: Arkham Kn...| Warner Bros. Inte...| 3.79|
| The Witcher 3: Wi...| Namco Bandai Games| 3.73|
| Star Wars Battlef...| Electronic Arts| 3.49|
| Metal Gear Solid ...| Konami Digital En...| 3.38|
| Assassin's Creed ...| Ubisoft| 3.28|
| Monster Hunter X| Capcom| 3.26|
| FIFA 16| Electronic Arts| 3.23|
| Madden NFL 16| Electronic Arts| 3.22|
| Super Mario Maker| Nintendo| 3.18|
| Gears of War: Ult...| Microsoft Game St...| 3.0|
+-----+-----+-----+

```

only showing top 20 rows

Describe the DataFrame

```

df.describe().show()

```

```

+-----+-----+-----+-----+-----+
| summary| Rank| Name| Platform|
| Year| Genre| Publisher| NA_Sales|
| EU_Sales| JP_Sales| Other_Sales|
| Global_Sales|
+-----+-----+-----+-----+

```

```

-----+-----+-----+-----+-----+-----+
---+-----+-----+-----+-----+-----+-----+
|  count|          16598|          16598|          16598|
16598|    16598|          16598|          16598|
16598|          16598|          16598|          16598|
|      mean|8300.605253645017|          1942.0|
2600.0|2006.4064433147546|          null|
null|0.26466742981084057|0.1466520062658483|0.07778166044101108|0.04
8063019640913515|  0.53744065550074|
|  stddev|  4791.8539328964|          null|          0.0|
5.828981114713253|          null|          null|
0.8166830292988798|0.5053512312869136|          0.3092906480822022|
0.18858840291271395|1.5550279355699066|
|      min|          1|          '98 Koshien|          2600|
1980|    Action|10TACLE Studios|          0.0|
0.0|          0.0|          0.0|          0.01|
|      max|          16600|iShin Chan Flipa ...|          XOne|
N/A|Strategy|  responDESIGN|          41.49|          29.02|
10.22|          10.57|          82.74|
+-----+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+-----+
---+-----+-----+-----+-----+-----+

```

Linear Regression Analysis

```

data = df.select("NA_Sales", "EU_Sales", "JP_Sales", "Other_Sales",
"Global_Sales").dropna()
feature_columns = ["NA_Sales", "EU_Sales", "JP_Sales",
"Other_Sales"]
assembler = VectorAssembler(inputCols=feature_columns,
outputCol="features")
data = assembler.transform(data)
train_data, test_data = data.randomSplit([0.8, 0.2], seed=42)
lr = LinearRegression(featuresCol="features",
labelCol="Global_Sales")
pipeline = Pipeline(stages=[lr])
lr_model = pipeline.fit(train_data)
predictions = lr_model.transform(test_data)
evaluator = RegressionEvaluator(labelCol="Global_Sales",
metricName="rmse")
rmse = evaluator.evaluate(predictions)
print("Linear Regression RMSE:", rmse)
lr_coefficients = lr_model.stages[-1].coefficients
print("Linear Regression Coefficients:", lr_coefficients)
Linear Regression RMSE: 0.005268710142527473
Linear Regression Coefficients:
[0.9999544028780193,1.0000392692552083,0.9999137696590534,0.99925460
28167025]

```

```

# Logistic Regression Analysis
data = df.select("NA_Sales", "EU_Sales", "JP_Sales", "Other_Sales",
"Global_Sales").dropna()
data = data.withColumn("TopSeller", (data["Global_Sales"] >
1).cast("int"))
feature_columns = ["NA_Sales", "EU_Sales", "JP_Sales",
"Other_Sales"]
assembler = VectorAssembler(inputCols=feature_columns,
outputCol="features")
data = assembler.transform(data)
train_data, test_data = data.randomSplit([0.8, 0.2], seed=42)
lr = LogisticRegression(featuresCol="features",
labelCol="TopSeller")
pipeline = Pipeline(stages=[lr])
lr_model = pipeline.fit(train_data)
predictions = lr_model.transform(test_data)
evaluator = BinaryClassificationEvaluator(labelCol="TopSeller")
accuracy = evaluator.evaluate(predictions, {evaluator.metricName:
"areaUnderROC"})
print("Logistic Regression Area under ROC:", accuracy)
Logistic Regression Area under ROC: 1.0

# Random Forest Classifier Analysis
data = df.select("NA_Sales", "EU_Sales", "JP_Sales", "Other_Sales",
"Global_Sales").dropna()
data = data.withColumn("TopSeller", (data["Global_Sales"] >
1).cast("int"))
feature_columns = ["NA_Sales", "EU_Sales", "JP_Sales",
"Other_Sales"]
assembler = VectorAssembler(inputCols=feature_columns,
outputCol="features")
data = assembler.transform(data)
train_data, test_data = data.randomSplit([0.8, 0.2], seed=42)
rf = RandomForestClassifier(featuresCol="features",
labelCol="TopSeller")
pipeline = Pipeline(stages=[rf])
rf_model = pipeline.fit(train_data)
predictions = rf_model.transform(test_data)
evaluator = BinaryClassificationEvaluator(labelCol="TopSeller")
accuracy = evaluator.evaluate(predictions, {evaluator.metricName:
"areaUnderROC"})
print("Random Forest Classifier Area under ROC:", accuracy)

Random Forest Classifier Area under ROC: 0.997757187166054

# Stop the Spark session
spark.stop()

```

