VIDEO GAME SALES

JAYMEE LIM

CM3010

JAN 2025

Stage 1. Find and critique a dataset

1.Introduction

In recent years, the video game industry has seen an exponential growth, it has evolved into one of the most lucrative and dynamic sectors in the entertainment market. It has generated billions of dollars in annual revenue and also influenced trends in technology and media. As gaming platforms continue to diversify, from consoles and PCs to mobile devices, the scope of game sales expands which offers more opportunities for developers and businesses to capitalize on the market demands.

Shareable link:
https://hub.labs.coursera.org:443/connect/sharedkyxnjfyp?forceRefresh=false&path=%2F%3Ffolder%3D%2Fhome%2Fcoder%2Fproject&isLabVersioning=true

1.1 Dataset Criteria Assessment

Quality

The dataset appears to be well structured and clean with clear column headers. Numerical values are consistent such as the global sales column matching the sum of all columns with region sales.
However the dataset has some missing values under the release year column of the dataset. As there is only a handful of missing data, I manually input the missing values (when the game was released) into the blank cells of the csv file.

Detail

The dataset has a high level of detail with a breakdown of region sales (JP, EU, NA, Other). Dataset also provides game-specific information like platform, year of release, publisher, and genre. However there are also some potentially valuable details not included such as player demographic, and marketing spend.

Documentation

This dataset is taken from kaggle and does not come with any supporting documentation.

Interrelation

The dataset can be interrelated to other datasets such as Platform trends overtime, Genre popularity across the regions, and sales performances of publishers or franchises. This

interrelations can be used for deeper analysis, and the inclusion of both sales as well as other game details can enable diverse analysis such as sales and trend predictions

Use

This dataset holds potential for other uses such as:

- Exploratory data analysis
- Visualization projects showcasing sales trends by region or genre
- Machine learning projects predicting sales based on game attributes

Discoverability

Since this dataset is well labelled and easy to interpret, this makes it accessible even for beginners. The dataset uses simple headers and intuitive data points ensuring that the dataset is straightforward to query or easy to manipulate using tools like Python or SQL.

Terms of use

According to Kaggle's terms of use:
Kaggle is organized under the laws of the State of Delaware, USA, and has offices located at 1600 Amphitheatre Parkway, Mountain View, California 94043 USA.
Kaggle itself does not claim ownership of the datasets uploaded by users. Ownership remains with the uploader.By uploading a dataset, the uploader grants Kaggle and other users the right to access, use, and modify the dataset, subject to the license the uploader chooses.

1.2 Interest in Video Game Sales dataset

I think this topic is interesting as video games reflect consumer preferences, cultural trends, and advancements in technology. Understanding factors that influence game sales such as platform popularity, genres, and time of release can help developers and publishers decide what kind of games to create and when to distribute.

This topic is also particularly interesting to me personally as I enjoy playing video games in my free time. As someone who enjoys playing games on multiple platforms, as well as exploring different genres of games, it makes it more intriguing to analyze the gaming market and understand how it shapes future games. By analyzing video games and its metadata, I might uncover some patterns that might explain the success of certain genres and platforms.

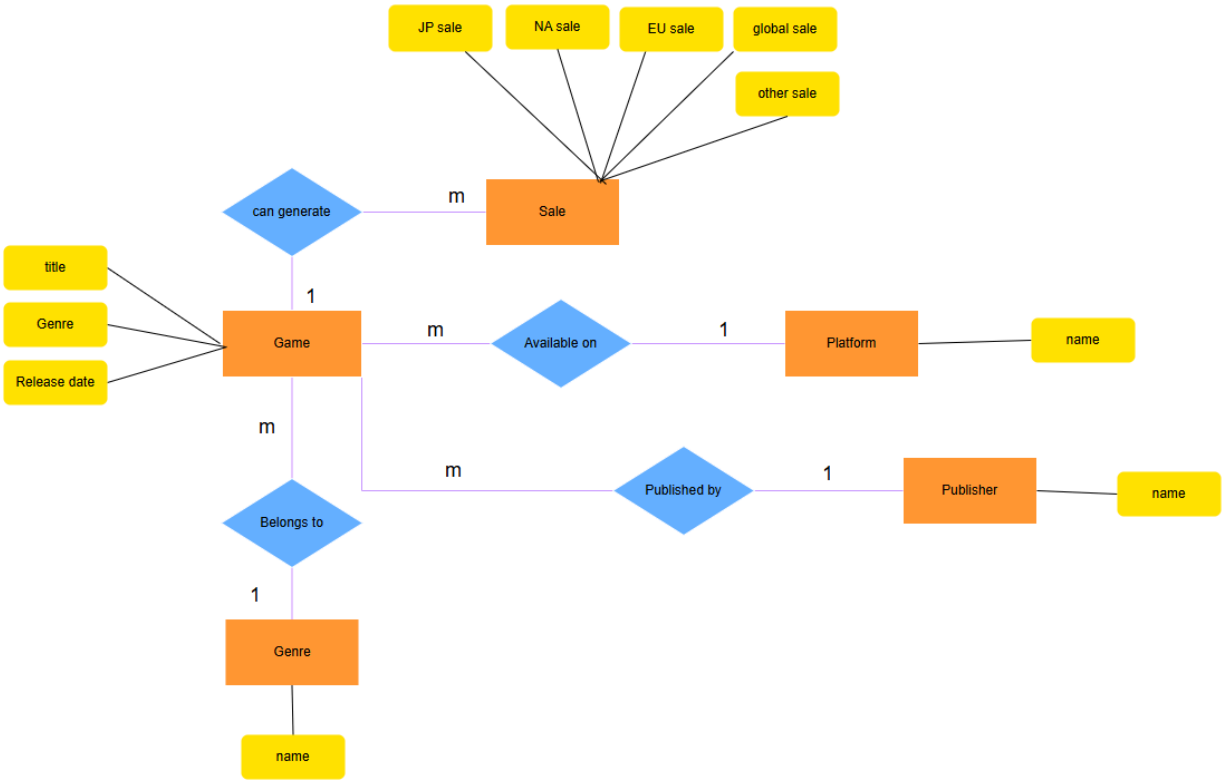
As i was looking through this dataset, several question came to my mind:

- Do certain platforms sell more games compared to other platforms?
- What genres perform best in different regions?
- Do games sell better when they are created by a company originating in their own country/region?
- How have global sales changed over time?
- Do games from specific publishers perform better on certain platforms?
- Which genres are most popular on specific platforms?

Stage 2. Model your data

2.1 E/R model

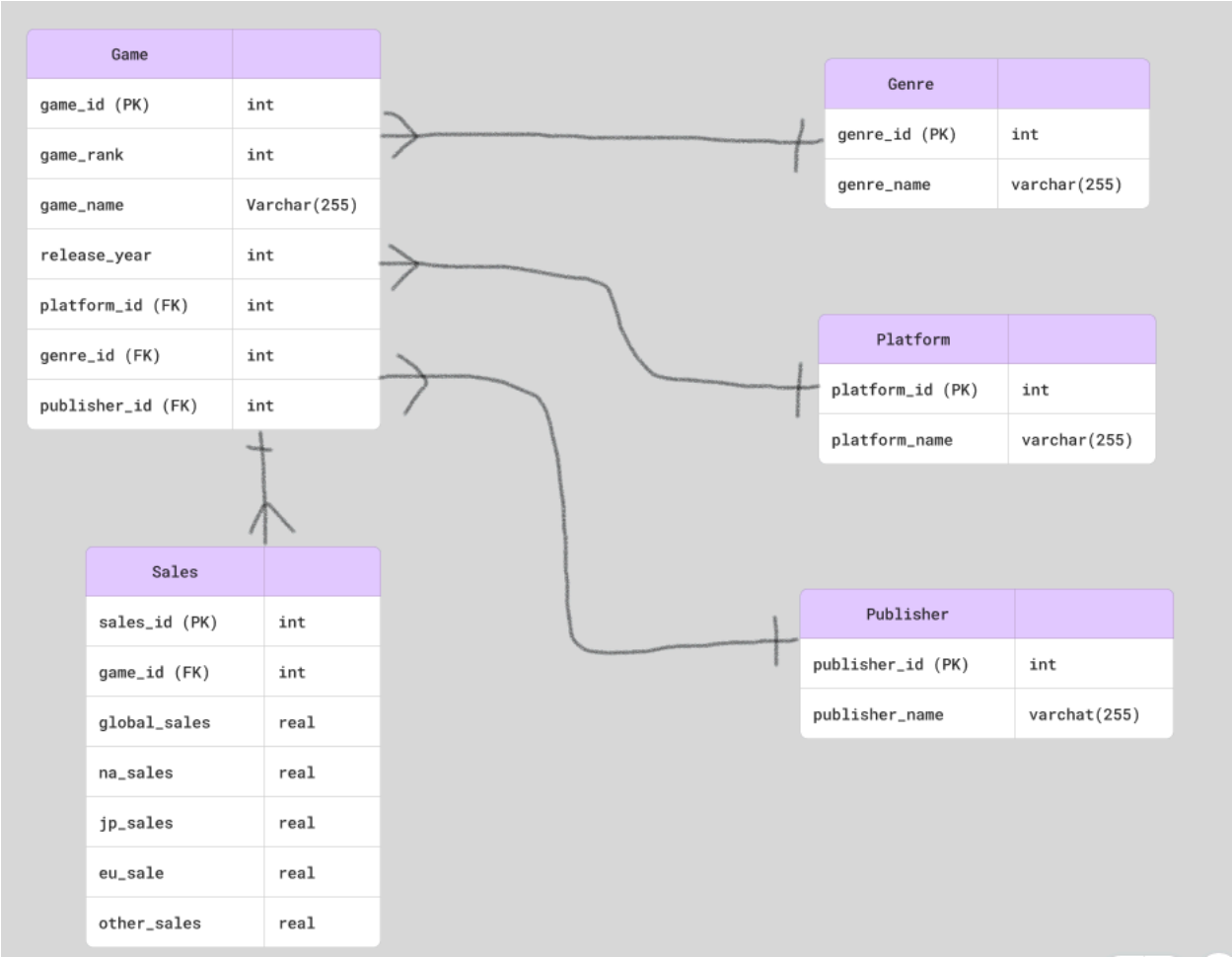
Below is the E/R model. The entire dataset is represented by this ER model. 1 game can generate many sales while many games can appear on 1 platform. Many games can also be published by 1 publisher. Many games can belong to 1 genre.
E/R model is drawn using draw.io [1]



2.2 Database tables and fields

Below are the tables and the relationship structure of the database. The PK symbolises primary key and FK stands for foreign key. I have also stated the type of data for each of the fields.

The tables are generated using Figma [2] and the lines are hand drawn as I could not work out the lines in figma.



2.3 Normalisation

The database is in 3NF as in the Games table, attributes like platform_id, genre_id, and publisher_id are foreign keys referring to separate tables. This avoids transitive dependencies. For example, platform_name is stored in the Platforms table and not repeated in the Games table.
The database is also in BCNF because in all tables, every determinant (e.g., primary key or composite key) is a candidate key and no partial or transitive dependencies exist.

Stage 3. Create the database

3.1 Create Commands

```
-- create database
CREATE DATABASE vgsales;
USE vgsales;
```

```
-- create platform table
CREATE TABLE platform (
    platform_id INT AUTO_INCREMENT PRIMARY KEY,
    platform_name VARCHAR(255) NOT NULL UNIQUE
);
```

```
-- create genre table
CREATE TABLE genre (
    genre_id INT AUTO_INCREMENT PRIMARY KEY,
    genre_name VARCHAR(255) NOT NULL UNIQUE
);
```

```
-- create publisher table
CREATE TABLE publisher (
    publisher_id INT AUTO_INCREMENT PRIMARY KEY,
    publisher_name VARCHAR(255) NOT NULL UNIQUE
);
```

```
-- Create the game table
CREATE TABLE games (
    game_id INT AUTO_INCREMENT PRIMARY KEY,
    game_rank INT,
    game_name VARCHAR(255) NOT NULL,
    platform_id INT NOT NULL,
    release_year INT NULL,
    genre_id INT NOT NULL,
    publisher_id INT NOT NULL,
    FOREIGN KEY (platform_id) REFERENCES platform(platform_id),
    FOREIGN KEY (genre_id) REFERENCES genre(genre_id),
    FOREIGN KEY (publisher_id) REFERENCES publisher(publisher_id)
);
```

```sql
-- Create the sales table
CREATE TABLE sales (
    sales_id INT AUTO_INCREMENT PRIMARY KEY,
    game_id INT NOT NULL,
    na_sales REAL NOT NULL,
    eu_sales REAL NOT NULL,
    jp_sales REAL NOT NULL,
    other_sales REAL NOT NULL,
    global_sales REAL NOT NULL,
    FOREIGN KEY (game_id) REFERENCES games(game_id)
);
```

```sql
CREATE TABLE temp_sales (
    game_name VARCHAR(255),
    na_sales FLOAT,
    eu_sales FLOAT,
    jp_sales FLOAT,
    other_sales FLOAT,
    global_sales FLOAT
);
```

3.2 Enter instance data

To enter the entire dataset of 16,600 entries into the database, I had to manipulate the original vgsales.csv files and break it into multiple csv files in order to use the LOAD DATA INFILE command.

First I used python to help break my csv file into multiple csv files which corresponded to the tables i have created. The csv files include: genre.csv, platform.csv, publisher.csv, games.csv, and sales.csv. I then transferred these files into the uploads folder located in mySQL folder in the programs data folder.

In [1]:
```python
import pandas as pd

# Load the dataset
data = pd.read_csv("vgsales.csv")

# Generate platform.csv
platform_df = pd.DataFrame(data["Platform"].unique(), columns=["platform_name"])
platform_df["platform_id"] = range(1, len(platform_df) + 1)
platform_df.to_csv("platform.csv", index=False)

# Generate genre.csv
genre_df = pd.DataFrame(data["Genre"].unique(), columns=["genre_name"])
genre_df["genre_id"] = range(1, len(genre_df) + 1)
genre_df.to_csv("genre.csv", index=False)

# Generate publisher.csv
publisher_df = pd.DataFrame(data["Publisher"].unique(), columns=["publisher_name"])
publisher_df["publisher_id"] = range(1, len(publisher_df) + 1)
publisher_df.to_csv("publisher.csv", index=False)

# Merge IDs into games.csv
games_df = data.merge(platform_df, left_on="Platform", right_on="platform_name") \
               .merge(genre_df, left_on="Genre", right_on="genre_name") \
               .merge(publisher_df, left_on="Publisher", right_on="publisher_name")
games_df = games_df[["Name", "platform_id", "genre_id", "publisher_id", "Year"]]
games_df.columns = ["game_name", "platform_id", "genre_id", "publisher_id", "release_year"]
games_df.to_csv("games.csv", index=False)

# Generate sales.csv
sales_df = data[["Name", "NA_Sales", "EU_Sales", "JP_Sales", "Other_Sales", "Global_Sales"]]
sales_df = sales_df.merge(games_df, left_on="Name", right_on="game_name")[["game_name", "NA_Sales", "EU_Sales", "JP_Sales", "Othe
sales_df.columns = ["game_name", "na_sales", "eu_sales", "jp_sales", "other_sales", "global_sales"]
sales_df.to_csv("sales.csv", index=False)
```

I then used the code below to import the data into the database. This is just an example of the command used. I repeated this for the other csv files and tables.

```
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server
9.1/Uploads/platform.csv'
INTO TABLE platform
FIELDS TERMINATED BY ',' ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS
(platform_name, platform_id);
```

I included all the raw files in the zip folder.

3.3 Reflection

Overall I feel that the database reflects the information well. I have separated the data into distinct tables which reduces redundancy and ensures that every entity is represented in a single table. Foreign keys are also used to link multiple tables together to ensure data consistency and maintain relationships across tables.

Some things I would have done differently is the sales table. Currently the global_sales column in the sales table is an input by itself. It was included as i just followed the database i found on kaggle and included it. In the future, I would have let global_sales to always be the sum of all the other fields (jp + na + eu + other) to prevent data inconsistencies. I did not change the table due to time constraints.

I also encountered another problem with the sales table. The sales table contains game_id which is the foreign key to the games table. The sales table does not have a field for game_name, but the games csv has a game_name column. So I could properly load my data into the tables. To counter this I created a temporary sales table with game_name as a field. I imported the data from my csv to the temp_sales table. After that I map game_id based on game_name and insert it into the sales table. I did this process by using an INSERT INTO query with a JOIN to map game_name to game_id from the games table.

3.4 List SQL Queries

```sql
--show if certain platform sells more games than others
SELECT
    platform_name,
    SUM(global_sales) AS Total_Sales
FROM sales
JOIN games ON sales.game_id = games.game_id
JOIN platform ON games.platform_id = platform.platform_id
GROUP BY platform_name
ORDER BY Total_Sales DESC;
```

This query calculates the total of global sales from the sales table for each platform and It gives the total global sales for each platform. The query starts by selecting data from the sales table, which likely contains records of game sales, including the game_id and global_sales. It is grouped by platform so we can see the total sales of each platform. Results are ordered in descending order of total sales meaning the platform with the highest sales will appear at the top.

```sql
--show What genres perform best in different regions?
SELECT
    genre_name,
    SUM(na_sales) AS NA_Sales,
    SUM(eu_sales) AS EU_Sales,
    SUM(jp_sales) AS JP_Sales,
    SUM(other_sales) AS Other_Sales,
    SUM(global_sales) AS Global_Sales
FROM sales
JOIN games ON sales.game_id = games.game_id
JOIN genre ON games.genre_id = genre.genre_id
GROUP BY genre_name
ORDER BY Global_Sales DESC;
```

This query is designed to show how different game genres perform across various regions and globally. The total sum of each region's sales is calculated for each genre. The query starts by selecting data from the sales table, which includes the regional sales for each game. sales table is joined with the games table using the game_id column. It ensures each sale is associated with the correct game. games table is joined with the genre table using the genre_id column. It associates each game with its corresponding genre. Results are grouped by genre so that sales can be calculated for each genre across all regions. Results are ordered in descending order of global sales

```sql
--show if games sell better when publisher comes from own region/country
SELECT
    p.publisher_name AS Publisher,
    SUM(s.na_sales) AS NA_Sales,
    SUM(s.eu_sales) AS EU_Sales,
    SUM(s.jp_sales) AS JP_Sales,
    SUM(s.other_sales) AS Other_Sales,
    SUM(s.global_sales) AS Global_Sales
FROM sales s
JOIN games g ON s.game_id = g.game_id
JOIN publisher p ON g.publisher_id = p.publisher_id
GROUP BY p.publisher_name
ORDER BY Global_Sales DESC;
```

The query is designed to show whether games perform better in regions when the publisher comes from the same region or country. It calculates the total sales for each publisher across different regions (North America, Europe, Japan, and Other regions), as well as the total global sales. The query starts by selecting data from the sales table, which contains sales figures for each game, including regional and global sales. Sales and games tables are joined using the game_id column. Results are grouped by the publisher_name from the publisher table, so that sales data can be aggregated for each publisher.

```sql
--show how have gloabl sales changed over time
SELECT
    release_year,
    SUM(global_sales) AS Total_Sales
FROM sales
JOIN games ON sales.game_id = games.game_id
GROUP BY release_year
ORDER BY release_year ASC;
```

This SQL query is designed to show how global game sales have changed over time. Year is selected from the games table and the total global sales for each game in each year is calculated. Results are grouped by release_year from the games table, so that sales can be aggregated for each year.

```sql
--Do games from specific publishers perform better on certain platforms?
SELECT
    publisher_name,
    platform_name,
    SUM(global_sales) AS Total_Sales
FROM sales
JOIN games ON sales.game_id = games.game_id
JOIN publisher ON games.publisher_id = publisher.publisher_id
JOIN platform ON games.platform_id = platform.platform_id
GROUP BY publisher_name, platform_name
ORDER BY publisher_name, Total_Sales DESC;
```

This SQL query is designed to analyze whether games from specific publishers perform better on certain platforms. The query starts by selecting data from the sales table, which contains sales figures, including global_sales for each game. Sales are linked with games tables, games are joined with publisher tables, games and platform tables are also joined. ORDER BY publisher_name, Total_Sales DESC orders the results by publisher name alphabetically, and within each publisher, by total sales (Total_Sales) in descending order.

```sql
--Which genres are most popular on specific platforms?
SELECT
    platform_name,
    genre_name,
    SUM(global_sales) AS Total_Sales
FROM sales
JOIN games ON sales.game_id = games.game_id
JOIN platform ON games.platform_id = platform.platform_id
JOIN genre ON games.genre_id = genre.genre_id
GROUP BY platform_name, genre_name
ORDER BY platform_name, Total_Sales DESC;
```

This SQL query is designed to analyze whether games from specific publishers perform better on certain platforms. It calculates the total global sales for each combination of publisher and platform and orders the results to highlight the best-performing platforms for each publisher.

Stage 4. Create a simple web application

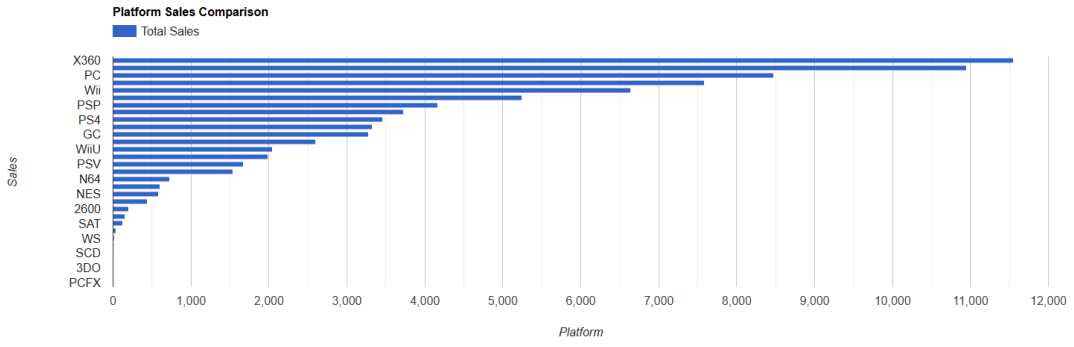
These are some of the screenshots of my simple web application which I am running in my own environment. It is built using express, google charts, and plotly.js.
This is the home page which you will see when you first run the code.

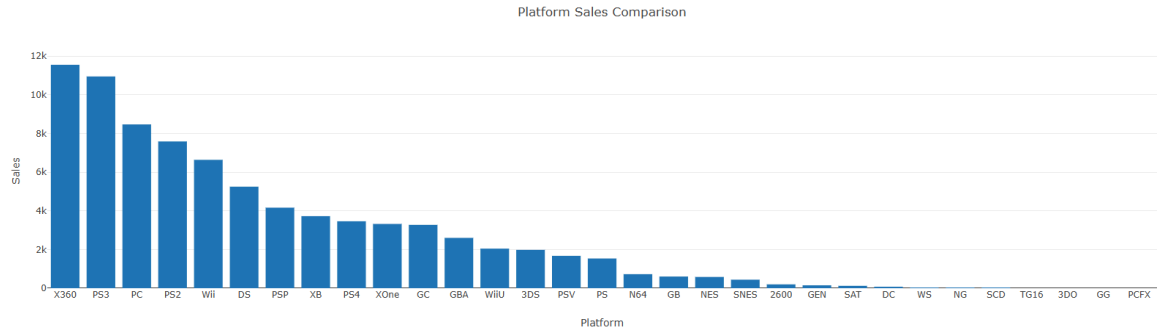
Video Game Sales Analysis

- Platform Sales Comparison
- Genre Performance by Region

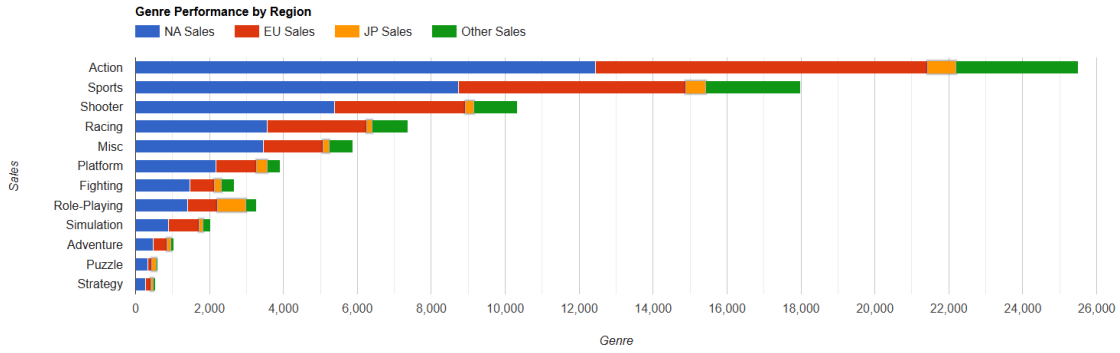
Clicking on the links will lead you to another page where the information is visualised in bar charts.
I chose to visualise bar charts as I think they are the easiest to read and compare the results with. The bar charts are also coloured so that we can see the differences better. The charts are also interactive where if your mouse is hovered over the bar, it will show you the information of that specific bar.

Do certain platforms sell more games compared to other platforms?

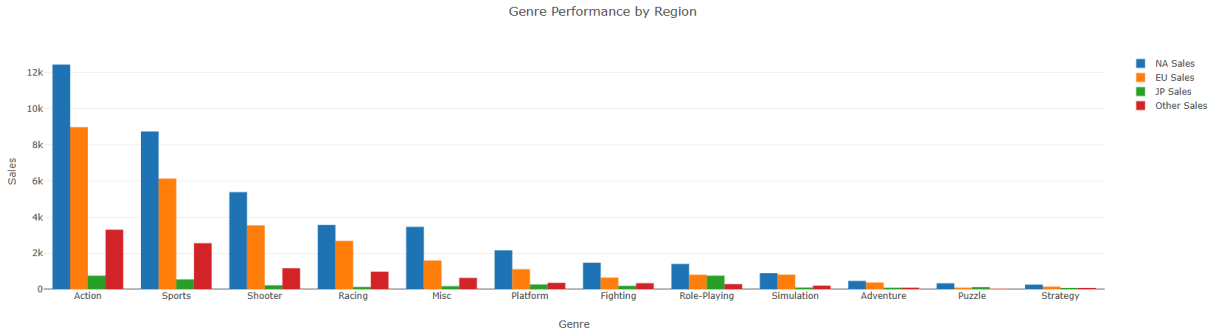




What genres perform best in different regions?



Genre Performance by Region



References

[1] https://app.diagrams.net/
[2]https://www.figma.com/files/team/1376896224599183837/recents-and-sharing?fuid=1376896222196221154

Dataset: https://www.kaggle.com/datasets/gregorut/videogamesales (sales with web scraping)