IRON HACK

Data Analytics

Video Game Market Analysis

Patole Jeremy

December 19th, 2025

Table of content

1. Introduction & Project Objectives

1.1 Introduction

The video game industry is one of the fastest-growing entertainment markets worldwide, generating billions of dollars annually across multiple platforms such as PC, consoles, and mobile devices.

It is a highly competitive market where publishers, developers, and distributors must make data-driven decisions regarding platform availability, genre positioning, monetization strategies, and multiplayer features.

Understanding what factors influence a game's success — measured through user ratings, engagement, and playtime — can provide valuable insights for companies operating in this sector.

This project aims to analyze the video game market using large-scale public data and to apply statistical analysis and machine learning techniques to extract meaningful insights.

1.2 Project Objectives

The main objective of this project is to **analyze the video game market** using real-world data and to **build predictive models** capable of estimating :

- the expected user rating of a game (regression),
- the likeliness of a game to be highly rated (classification).

2. Use Case Definition & Goal

2.1 Business Use Case

The primary use case targets video game publishers and studios seeking to :

- Understand the drivers of high user ratings
- Identify successful genres and platforms
- Evaluate the impact of multiplayer features
- Support strategic decisions related to platform releases and game design

Secondary use cases include :

- Market analysts
- Digital distribution platforms (Steam, Epic Games, PlayStation Store)
- Independent developers

The project answers questions such as :

- Do multi-platform games perform better?
- Does multiplayer gameplay increase player engagement?
- Are some genres more likely to receive high ratings?

2.2 Goal

To accomplish the project objectives, we need to :

- Collect and consolidate video game data from multiple sources
- Clean, normalize, and structure heterogeneous datasets
- Explore market trends through statistical analysis and visualization
- Test business-oriented hypotheses related to game quality and performance
- Build predictive models

3. Project Planning and Methodology

3.1 Project Planning

The project followed a structured workflow inspired by agile methodologies.

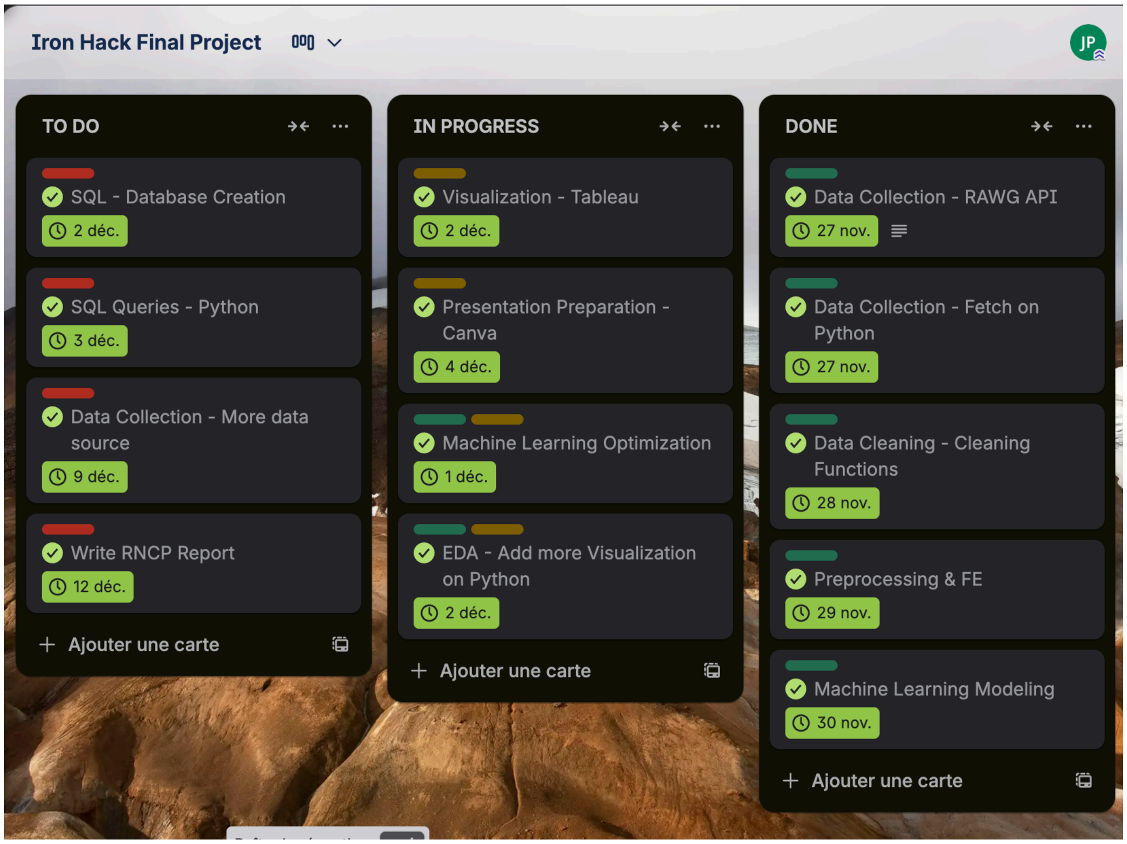
Pipeline :

1. Data collection
2. Data cleaning
3. Exploratory data analysis
4. Hypothesis testing
5. Feature engineering
6. Machine learning modeling
7. Visualization and interpretation
8. Extraction of meaningful business insights

Project planning was managed using **Trello**, allowing task tracking, prioritization, and progress monitoring.

The project management was consistently evolving throughout its entire duration.

Some tasks planned for later were ultimately carried out as a priority, some steps were deemed less essential than others, always with the aim of maximum optimization over a short period of preparation.



Iron Hack Final Project

JP

TO DO

SQL - Database Creation
2 déc.

SQL Queries - Python
3 déc.

Data Collection - More data source
9 déc.

Write RNCP Report
12 déc.

+ Ajouter une carte

IN PROGRESS

Visualization - Tableau
2 déc.

Presentation Preparation - Canva
4 déc.

Machine Learning Optimization
1 déc.

EDA - Add more Visualization on Python
2 déc.

+ Ajouter une carte

DONE

Data Collection - RAWG API
27 nov.

Data Collection - Fetch on Python
27 nov.

Data Cleaning - Cleaning Functions
28 nov.

Preprocessing & FE
29 nov.

Machine Learning Modeling
30 nov.

+ Ajouter une carte

Fig.1 Project planification view on Trello

3.2 Tool Used

- Python (Pandas, NumPy, Scikit-learn, scipy.stats)
- RAWG API
- Steam / SteamSpy data
- MySQL (planned)
- Tableau Public
- Flask (API exposure)
- GitHub (version control)

4. Data Collection and Data Sources

I did explore multiple data sources before making a choice according to the project needs.

4.1 API Data (RAWG API & Steam API)

The primary data source for this project was the **RAWG Video Games Database API** (aprox 800,000 games), which provides structured information on :

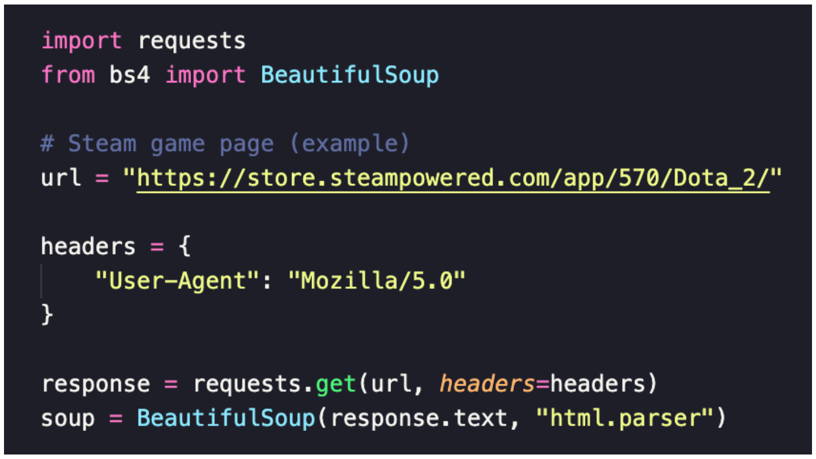
- game metadata
- platforms
- genres
- stores
- ratings
- tags
- user engagement metrics

With 2 API key with 20k requests each (some data was lost because of network errors), I manage to build 2 datasets with almost 38k data before cleaning that I use as raw data.
Those datasets, and those created after cleaning were all saved as .csv files.

I couldn't combine data from the Steam API and those from RAWG API because the SteamApps endpoint and the IStoreService endpoint were deleted or unavailable at the time.

4.2 Web Scraping (Supplementary)

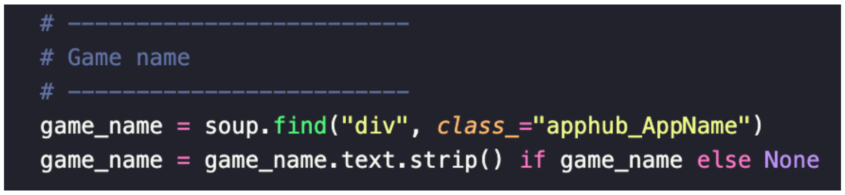
A web scraping step was implemented to collect supplementary data from Steam game pages using BeautifulSoup.



```python
import requests
from bs4 import BeautifulSoup

# Steam game page (example)
url = "https://store.steampowered.com/app/570/Dota_2/"

headers = {
    "User-Agent": "Mozilla/5.0"
}

response = requests.get(url, headers=headers)
soup = BeautifulSoup(response.text, "html.parser")
```

Fig. 2 : Code Snippet of the Web Scraping Step

This approach was used to retrieve publicly available information such as game name, release date, user review summary, tags, and developer information.



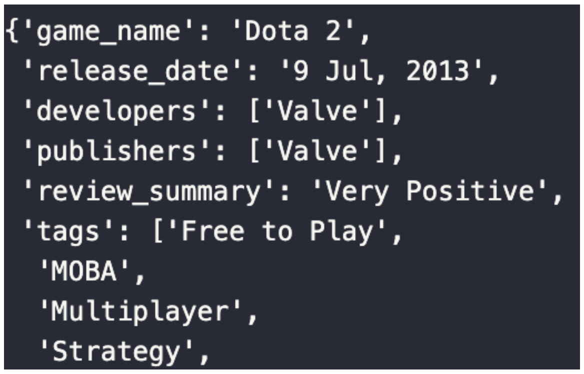
```
# --------------------------
# Game name
# --------------------------
game_name = soup.find("div", class_="apphub_AppName")
game_name = game_name.text.strip() if game_name else None
```

Fig. 3 : Example of infos retrieved with Web Scraping

The scraped data was not used as the main analytical dataset but served as contextual enrichment to better understand how games are presented and perceived on major distribution platforms.

Limitations & Legal aspects **:**

The scraping process was limited to a small number of pages and respected Steam's public access constraints.
No authentication, automation at scale, or personal data collection was involved.
The scraping activity was conducted strictly for educational and analytical purposes.



```
{'game_name': 'Dota 2',
 'release_date': '9 Jul, 2013',
 'developers': ['Valve'],
 'publishers': ['Valve'],
 'review_summary': 'Very Positive',
 'tags': ['Free to Play',
  'MOBA',
  'Multiplayer',
  'Strategy',
```

Fig. 4 : Result after scraping the data

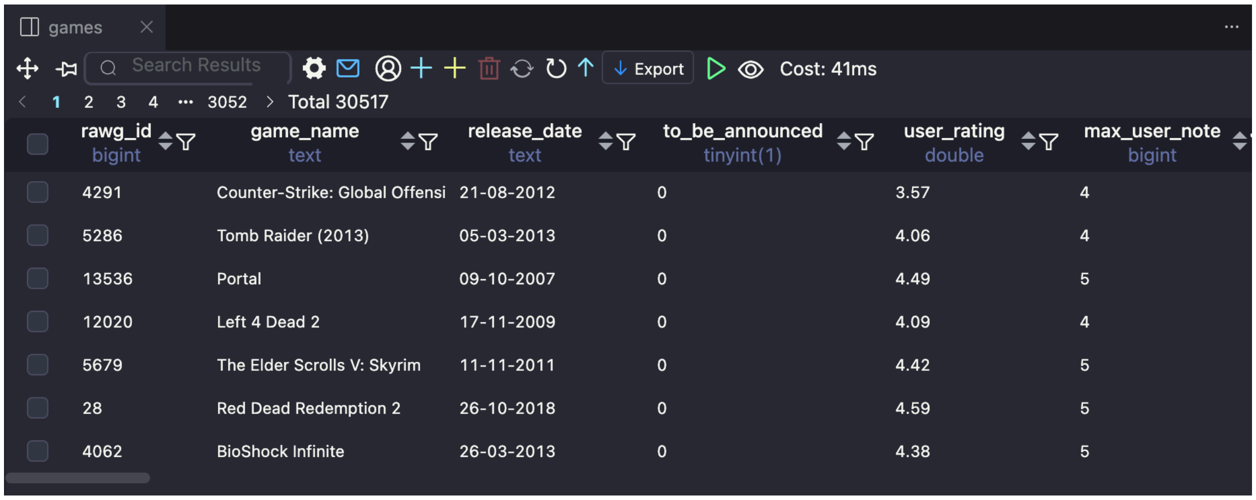
4.3 Flat Files (CSV)

Video Games Sales's CSV were found on Kaggle, and could be use for further comparative analysis with more machine learning models.
Intermediate datasets were stored as CSV files to :

- facilitate debugging
- ensure reproducibility and enable sharing with visualization tools

4.4 Relational Database (MySQL)

Although most analysis was performed in Python, a relational database schema was designed to store cleaned data using MySQL Workbench using Python and SQLAlchemy. This option was chosen over the MySQL Workbench import wizard to avoid common CSV import issues such as encoding errors and type mismatches.



rawg_id bigint	game_name text	release_date text	to_be_announced tinyint(1)	user_rating double	max_user_note bigint
4291	Counter-Strike: Global Offensi	21-08-2012	0	3.57	4
5286	Tomb Raider (2013)	05-03-2013	0	4.06	4
13536	Portal	09-10-2007	0	4.49	5
12020	Left 4 Dead 2	17-11-2009	0	4.09	4
5679	The Elder Scrolls V: Skyrim	11-11-2011	0	4.42	5
28	Red Dead Redemption 2	26-10-2018	0	4.59	5
4062	BioShock Infinite	26-03-2013	0	4.38	5

Fig. 5 : View of the data stored in a Database

4.5 Big Data System (BigQuery)

A denormalized version of the dataset can be uploaded to **Google BigQuery** for :

- large-scale querying
- partitioning by release year
- clustering by platform or genre

4.6 Data Source Choice

This RAWG API was chosen because it offers :

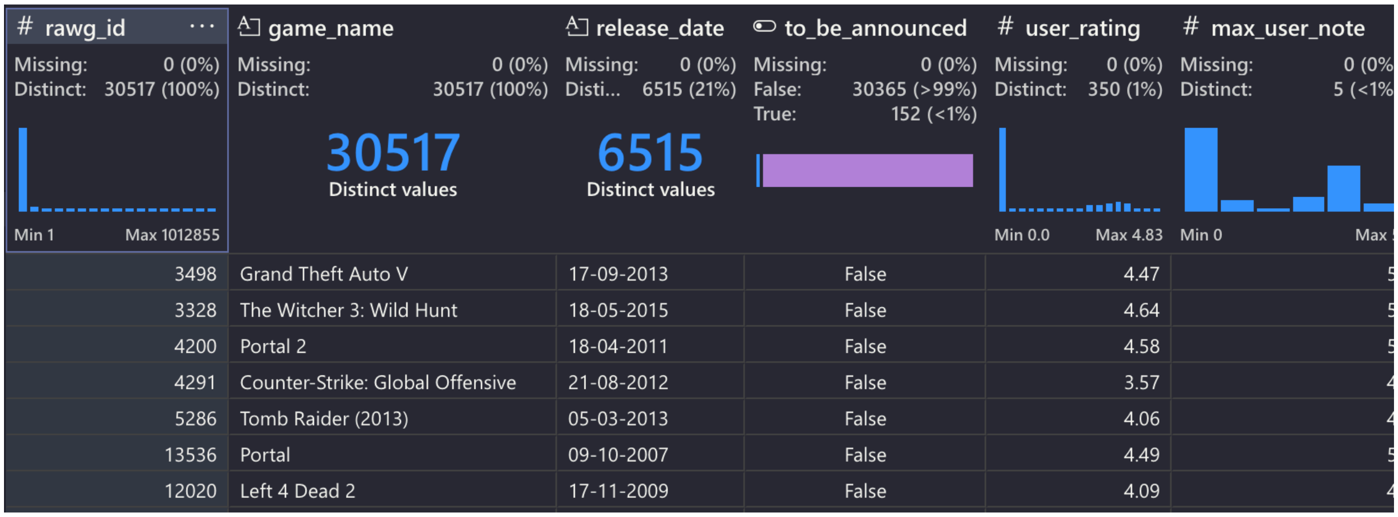
- a large and diverse dataset
- reliable and well-documented endpoints
- up-to-date information

5. METADATA / Data Dictionary

A data dictionary was created to describe each variable, its type, and its meaning.
Lets take a look at the most important columns :

Column Name	Type	Description
rawg_id	Integer	Unique game identifier
game_name	String	Game title
release_date	Datetime	Release date of the game
user_rating	Float	Average user rating (0–5)
rating_count	Integer	Total number of ratings
avg_playtime_hours	Float	Average user's playtime
platforms_count	Integer	Number of supported platforms
platforms_list	String	List of all platform supported by the game
genre_count	Integer	Total number of genre
genres_list	String	List of genres
store_count	Integer	Total number of stores that are distributors
store_list	String	List of stores
….		

Fig. 6 : Data Dictionary (main columns)



# rawg_id	game_name	release_date	to_be_announced	# user_rating	# max_user_note
Missing: 0 (0%)	Missing: 0 (0%)	Missing: 0 (0%)	Missing: 0 (0%)	Missing: 0 (0%)	Missing: 0 (0%
Distinct: 30517 (100%)	Distinct: 30517 (100%)	Disti... 6515 (21%)	False: 30365 (>99%)	Distinct: 350 (1%)	Distinct: 5 (<1%
			True: 152 (<1%)		
	30517 Distinct values	6515 Distinct values			
Min 1 Max 1012855				Min 0.0 Max 4.83	Min 0 Max
3498	Grand Theft Auto V	17-09-2013	False	4.47	5
3328	The Witcher 3: Wild Hunt	18-05-2015	False	4.64	5
4200	Portal 2	18-04-2011	False	4.58	5
4291	Counter-Strike: Global Offensive	21-08-2012	False	3.57	4
5286	Tomb Raider (2013)	05-03-2013	False	4.06	4
13536	Portal	09-10-2007	False	4.49	5
12020	Left 4 Dead 2	17-11-2009	False	4.09	4

Fig. 7 : Final DataFrame first column (after cleaning)

6. Data Cleaning

6.1 Challenges

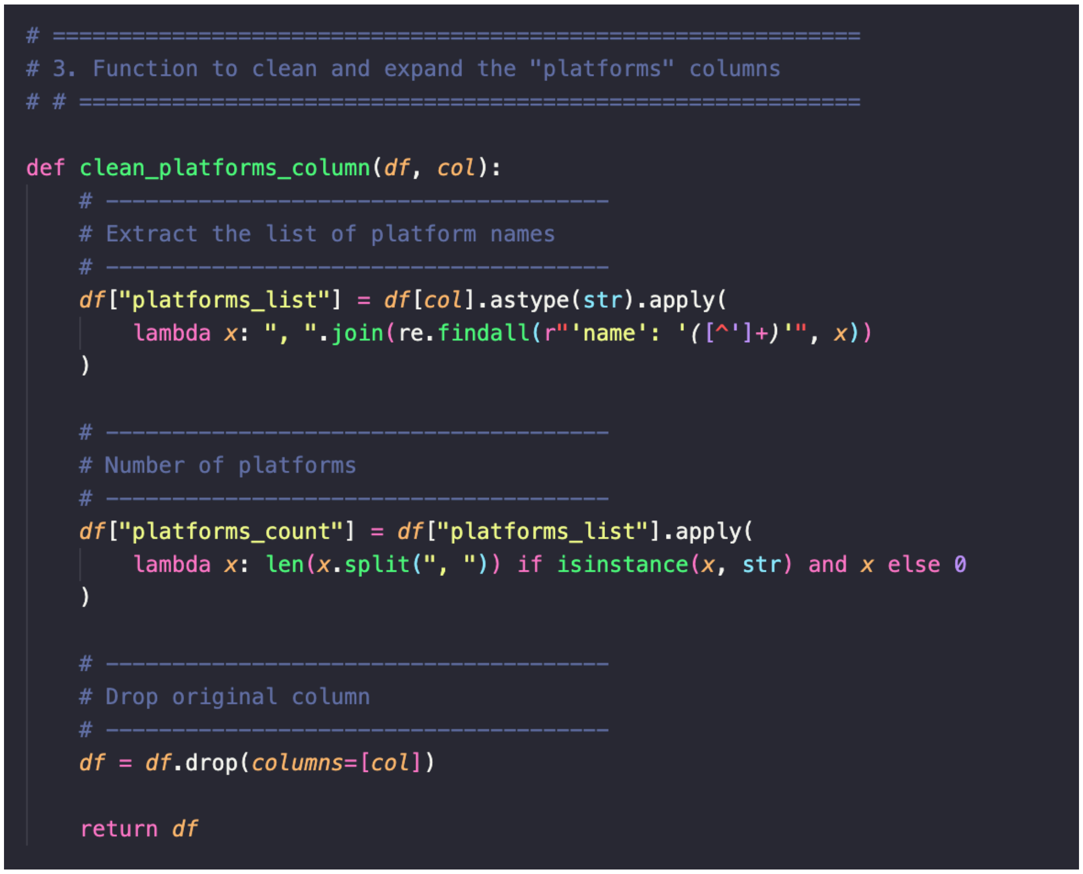
The raw data presented several issues :

- missing values
- nested JSON structures
- inconsistent formats
- duplicated information across sources

6.2 Cleaning Steps

- Conversion of JSON-like columns into structured features using regular expressions
- Removal of high-missing-rate columns
- Standardization of column names and categorical values
- Creation of meaningful numerical indicators

All cleaning steps were implemented in Python using Pandas and .py files to store all the cleaning user's defined functions, DataFrames merging functions.



```python
# ============================================================
# 3. Function to clean and expand the "platforms" columns
# # ==========================================================

def clean_platforms_column(df, col):
    # ---------------------------------------
    # Extract the list of platform names
    # ---------------------------------------
    df["platforms_list"] = df[col].astype(str).apply(
        lambda x: ", ".join(re.findall(r"'name': '([^']+)'", x))
    )

    # ---------------------------------------
    # Number of platforms
    # ---------------------------------------
    df["platforms_count"] = df["platforms_list"].apply(
        lambda x: len(x.split(", ")) if isinstance(x, str) and x else 0
    )

    # ---------------------------------------
    # Drop original column
    # ---------------------------------------
    df = df.drop(columns=[col])

    return df
```

Fig. 8 : Function to clean the "platform" column

7. Exploratory Data Analysis (EDA)

EDA was conducted to :

- identify distributions,
- detect correlations,
- validate assumptions for hypothesis testing.

With the following hypothesis, we explore the influence of multiple key indicators (genre, number of platforms, type of content..) on the player engagement (rating, playtime..)
For each of the hypothesis, we did reject the H0.

H1. Steam games has higher rating on average than non-Steam games
→ The ratings are significantly different

H2. The proportion of high rating is higher for multi-platform games than for mono-platform
→ The proportions are significantly different

H3. There is a difference in genre distribution based on user rating
→ There is a correlation between game genre and high rating probability

H4. ESRB Rating has an influence on high rating probability
→ There is a correlation between ESRB rating and high rating probability

H5. The number of platforms and the rating are correlated
→ There is a correlation between the number of platform and the user rating

H6. Users are more engaged on Multiplayer games
→ There is a difference in playtime between multiplayer and non-multiplayer games

Each hypothesis were tested with the appropriate test (chi-square test, z-test, t-test, spearman correlation, Mann–Whitney test)

Key Insights :

1. **Multi-platform** releases **perform better**.
2. Steam remains a central distribution channel for successful titles even if our test with this specific sample we saw otherwise
3. **Multiplayer increases engagement** and long-term playtime.
4. **Genres matter** - it influence significantly quality perception
5. Review ratios and engagement metrics predict success early.

Visualizations included :

- box plots (ratings on Steam),
- bar charts (genres vs ratings ; mono vs multi platform),
- scatter plot (correlation between high rating games and number of platforms)

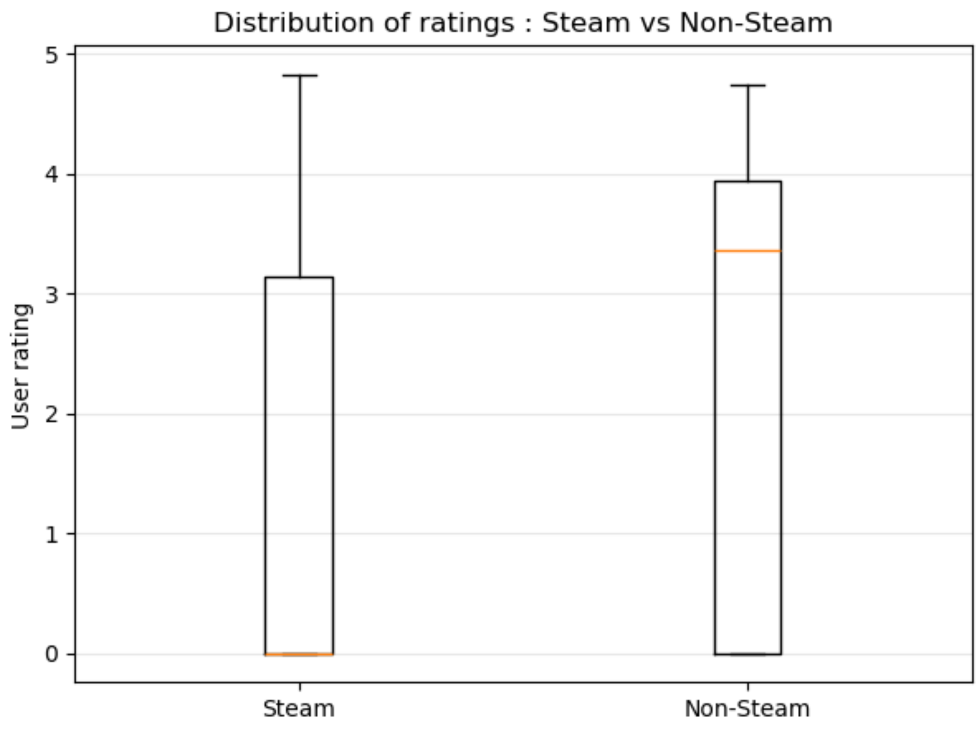


Fig. 9 : Rating distribution on Steam vs non-Steam games

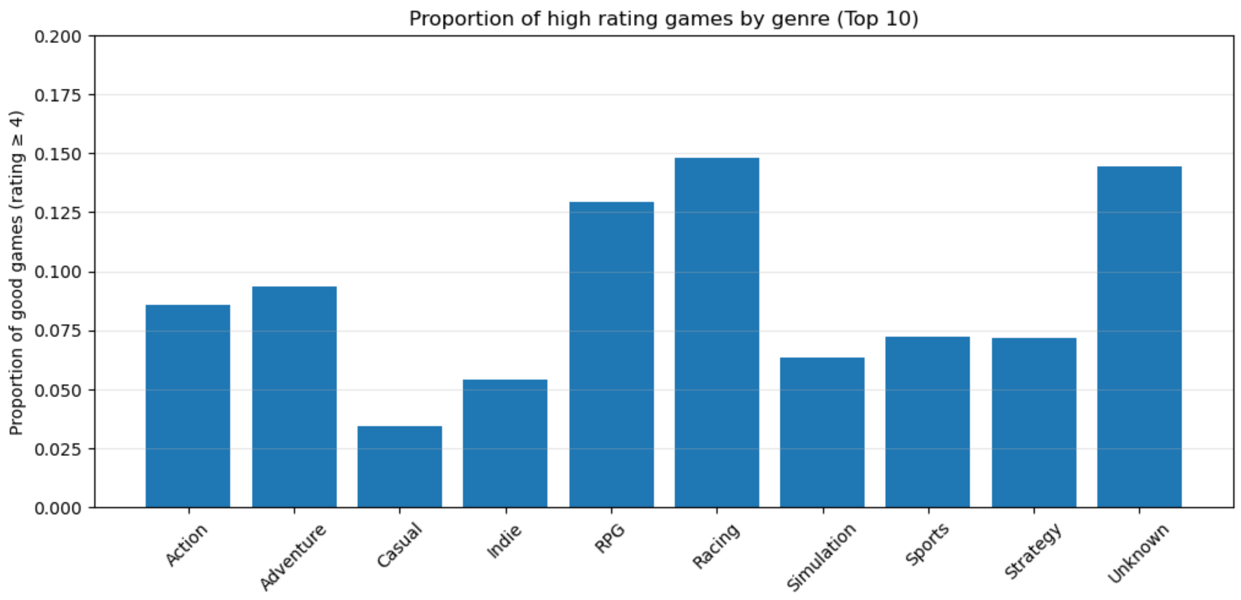


Fig. 10 : Correlation between rating and game genre

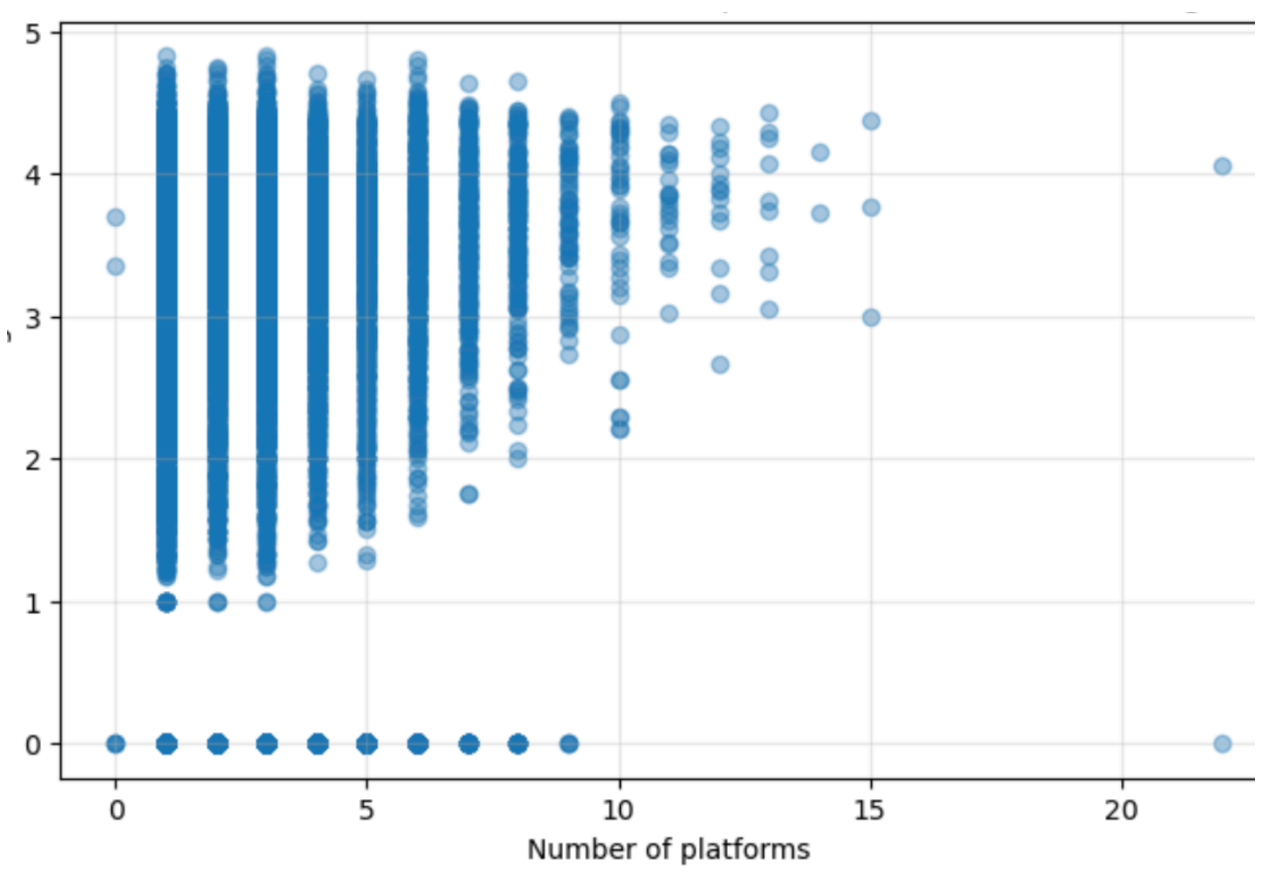


Fig. 11: Correlation between the number of platforms and user rating

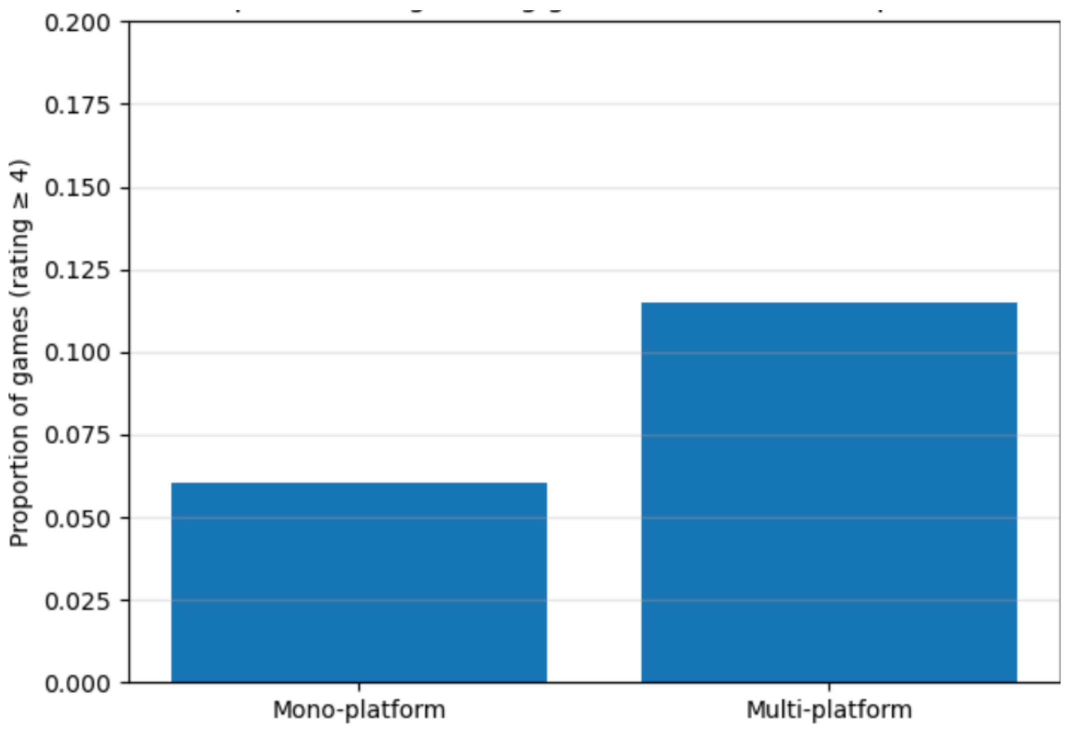


Fig. 12 : High rating games proportion - mono vs multi-platform

8. Feature Engineering

8.1 Feature Engineering Strategy

The primary objective of this step was to transform raw and semi-structured data into meaningful numerical features while strictly avoiding data leakage, which could artificially inflate model performance.

The starting point for this phase was the final consolidated dataset (df_final), which contained game-level information such as user ratings, engagement metrics, platform availability, genres, store presence, and community feedback indicators.

8.2 Leakage Prevention

To ensure that the model learned only from features that would realistically be available prior to observing the target outcome, we exclude the following from the feature set :

- Variables explicitly encoding the target and aggregated review metrics too closely correlated with the final user rating were carefully filtered
- Previously engineered flag DataFrames (e.g. platform or genre binary indicators created before modeling) were deliberately not merged to prevent redundancy and hidden leakage

8.3 Feature Selection and Transformation

The final feature set included the following categories :

- Temporal features
  - Release year (extracted from release date)
- Engagement features
  - Average playtime
  - Number of users playing, owning, or completing the game
- Popularity indicators
  - Number of ratings
  - Number of reviews
- Structural features
  - Number of platforms
  - Number of genres and tags
  - Number of stores

Categorical variables with a limited number of categories were encoded using one-hot encoding when relevant.

9. Machine Learning

9.1 Machine Learning Objectives

Two distinct machine learning tasks were addressed:

1. ***Regression Task***
   Predict the average user rating of a video game on a continuous scale.
2. ***Classification Task***
   Predict whether a game belongs to the "high rating" category (binary classification).

These two tasks were treated independently, with dedicated training pipelines, evaluation metrics, and validation strategies.

9.2 Regression Modeling

Target Variable :

- **User Rating** (continuous value between 0 and 5)

Only features that were independent from the target and available at prediction time were retained.

Models Evaluated

Four regression models were trained and evaluated:

- Random Forest Regressor
- Gradient Boosting Regressor
- LightGBM Regressor
- CatBoost Regressor

Each model was trained using a train-test split strategy, with consistent preprocessing applied to both training and test sets.

Model performance was evaluated using :

- R^2 Score
- Mean Absolute Error (MAE)
- Root Mean Squared Error (RMSE)

Special care was taken to validate that performance scores were realistic and not the result of data leakage. After correcting leakage issues, model performance stabilized at high but plausible levels. It can still be improved by optimizing the feature engineering process.

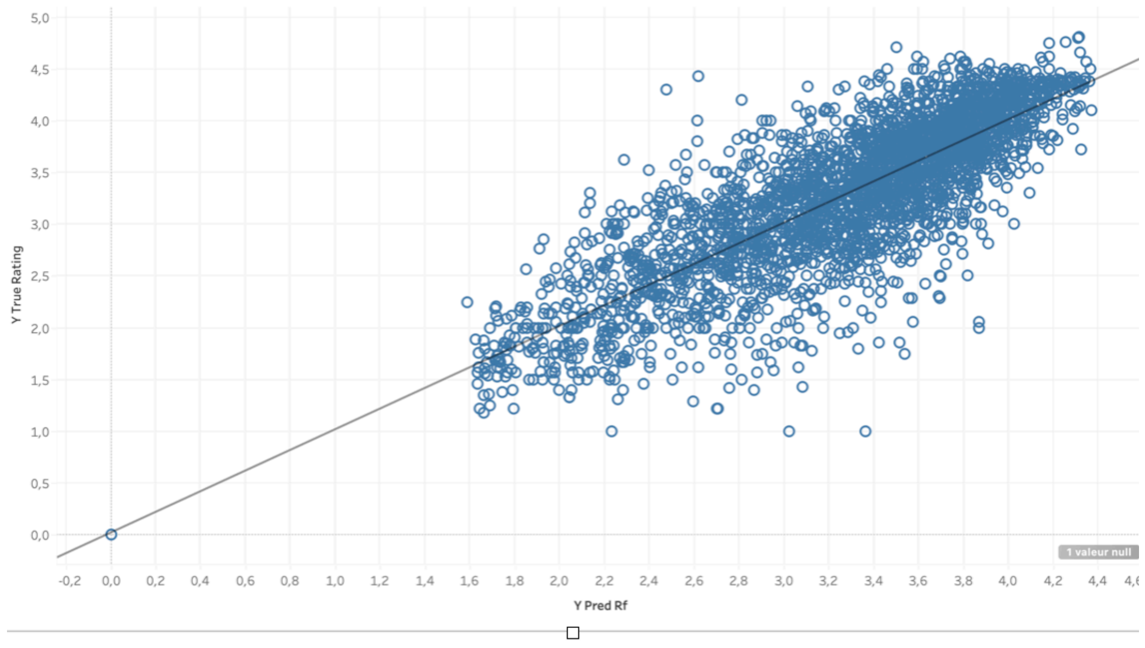


Fig. 13 : Predicted vs Actual Ratings

As can be seen, the majority of our points are concentrated around the trend line, indicating predicted values close to actual values and therefore attesting to the accuracy of our predictive models.

9.3 Classification Modeling

Target Variable

The classification task aimed to predict :

- **High Rating Flag**
  - 1 if the user rating ≥ predefined threshold
  - 0 otherwise

This target was derived from the original rating variable and handled in a separate modeling pipeline to avoid contamination between tasks.

Models Evaluated

The following classification models were tested :

- Logistic Regression
- Random Forest Classifier
- Gradient Boosting Classifie

Evaluation Metrics

Classification performance was assessed using :

- Confusion Matrix

- Accuracy
- Precision
- Recall
- F1-score
- ROC-AUC

9.4 Model Interpretation and Comparison

Model evaluation showed that tree-based ensemble methods achieved the best performance across both tasks, confirming the relevance of non-linear patterns in video game data.

For the **regression task**, the **LightGBM Regressor** delivered the highest predictive accuracy, outperforming other ensemble models. This highlights its ability to efficiently capture complex interactions between engagement metrics, platform exposure, and community activity.

For the **classification task**, the **Random Forest Classifier** proved to be the most effective model for predicting whether a game would achieve a high user rating. Its robustness and interpretability made it particularly suitable for this objective.
These results are coherent with business expectations in the video game industry and reinforce the practical relevance of the modeling approach.

9.5 Limitations and Future Improvements

Despite strong results, several limitations remain :

- The absence of real-time sales data limits commercial inference
- Ratings may be biased by platform-specific communities
- Some features are proxies rather than direct measures of success

Future improvements could include :

- Time-series modeling of ratings evolution
- Integration of external sales or revenue data
- Advanced natural language processing on user reviews

These insights are further contextualized in the following sections through data storage, querying, and API exposure.

10. Data Storage & ERD

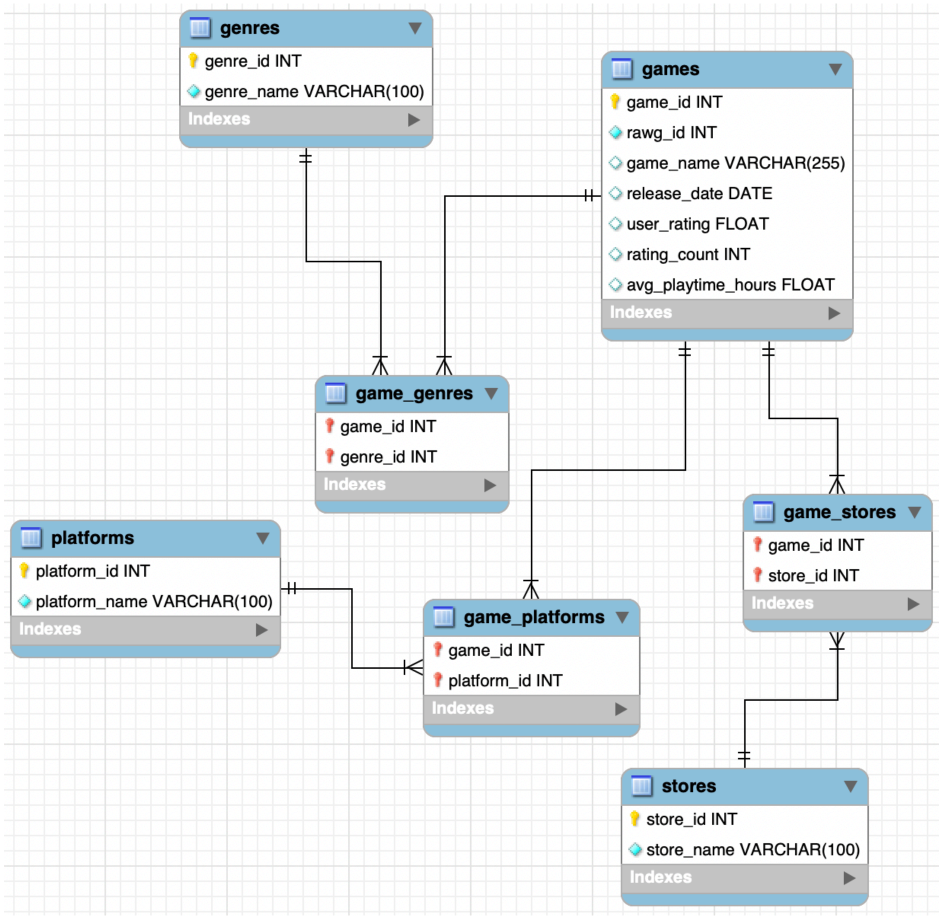
10.1 Relational Database Design

The dataset was imported with SQLalchemy and normalized into multiple tables :

- games
- platforms
- genres
- stores
- ratings

10.2 Entity Relationship Diagram

The relational schema was designed following normalization principles. The initial dataset was denormalized and subsequently split into dimension tables and associative tables to handle many-to-many relationships.
The final schema consists of more than four entities and three relationships.

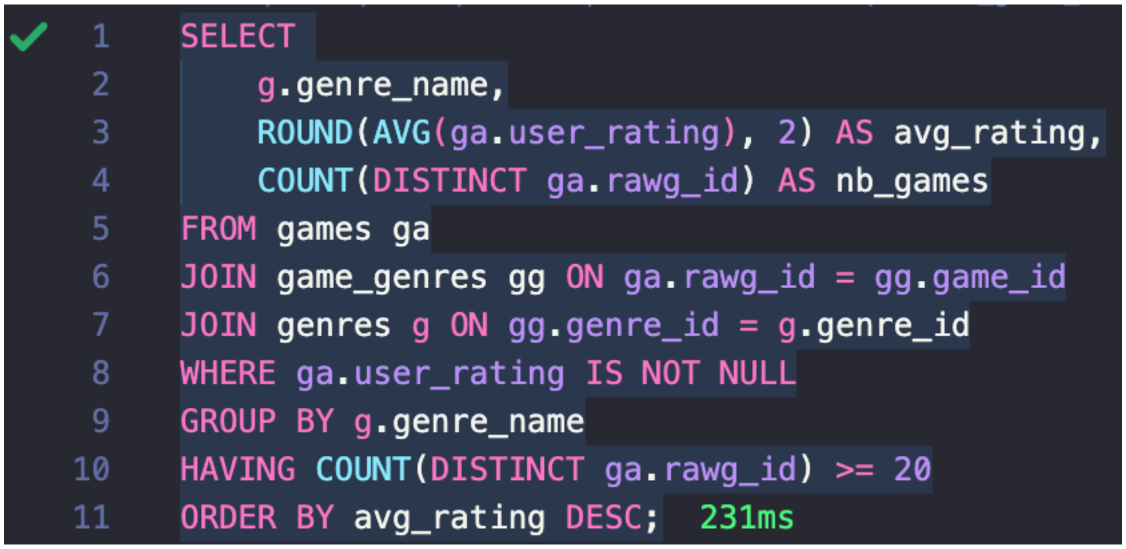


Fig; 14 : Entity Relationship Diagram

10.3 SQL Insights

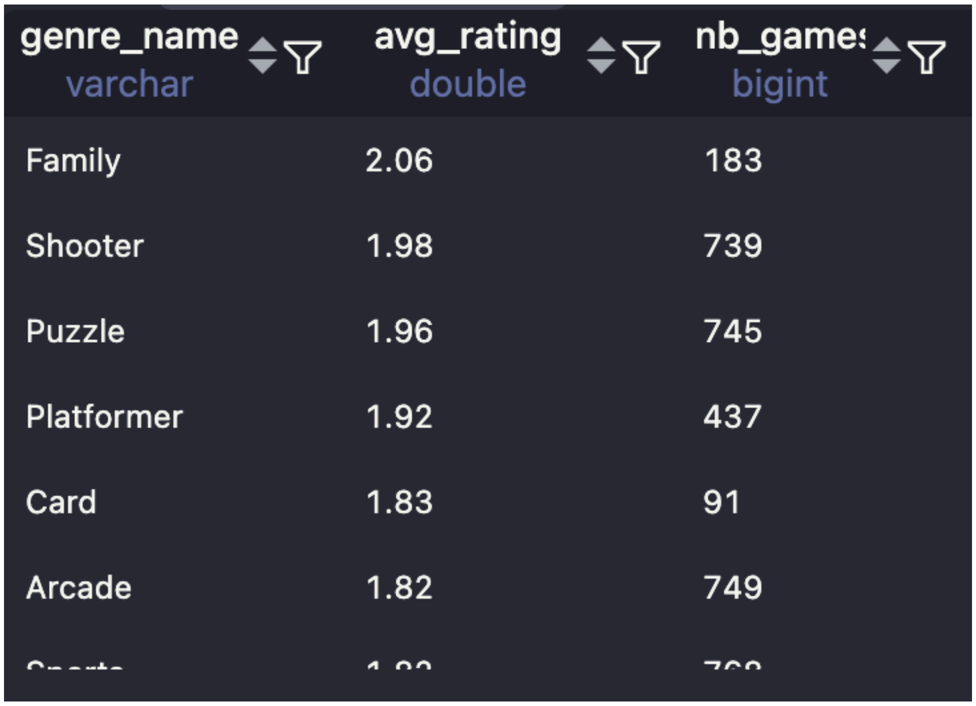
Five SQL queries were created to extract insights such as:

- top-rated genres
- platform popularity
- engagement metrics



```sql
SELECT
    g.genre_name,
    ROUND(AVG(ga.user_rating), 2) AS avg_rating,
    COUNT(DISTINCT ga.rawg_id) AS nb_games
FROM games ga
JOIN game_genres gg ON ga.rawg_id = gg.game_id
JOIN genres g ON gg.genre_id = g.genre_id
WHERE ga.user_rating IS NOT NULL
GROUP BY g.genre_name
HAVING COUNT(DISTINCT ga.rawg_id) >= 20
ORDER BY avg_rating DESC;  231ms
```

Fig. 15 : Example of query for Analysis on SQL



genre_name varchar	avg_rating double	nb_game bigint
Family	2.06	183
Shooter	1.98	739
Puzzle	1.96	745
Platformer	1.92	437
Card	1.83	91
Arcade	1.82	749

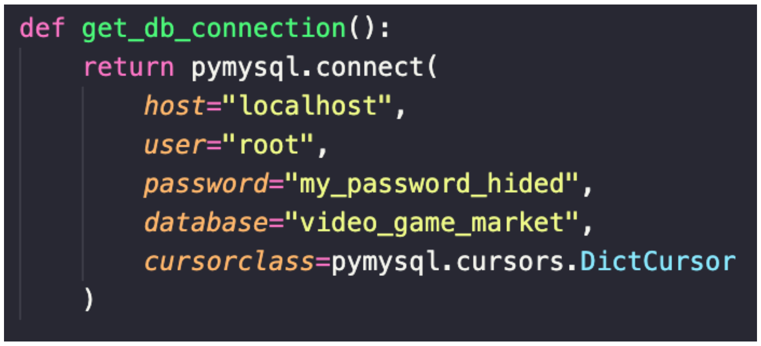
Fig. 16 : Output of the previous query

11. API Exposure (FLASK)

The API was built using Flask and the database introduced in the previous section.
I exposed two main resources :

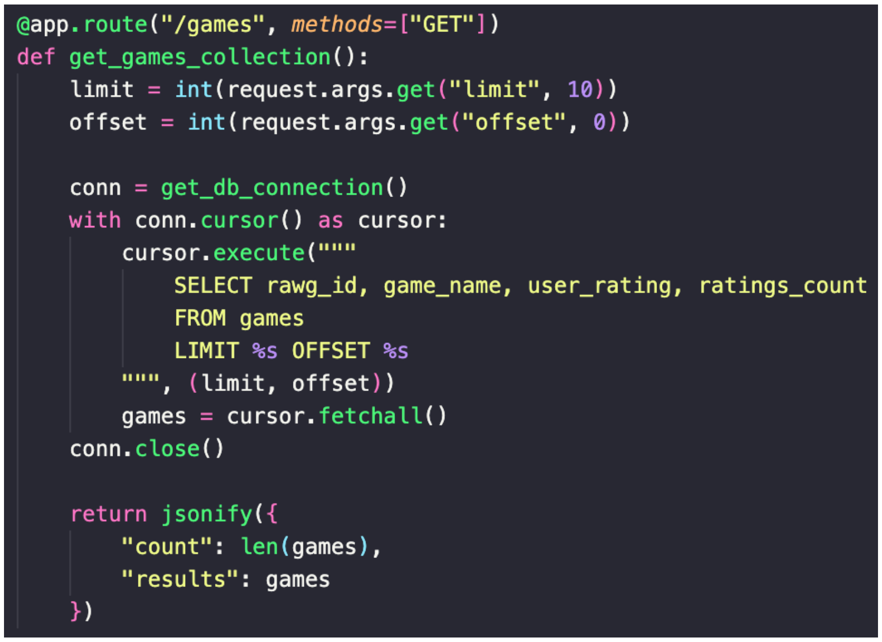
- A collection endpoint with pagination
- A single-resource endpoint by RAWG ID

This architecture follows REST principles and allows external systems to consume the data easily.



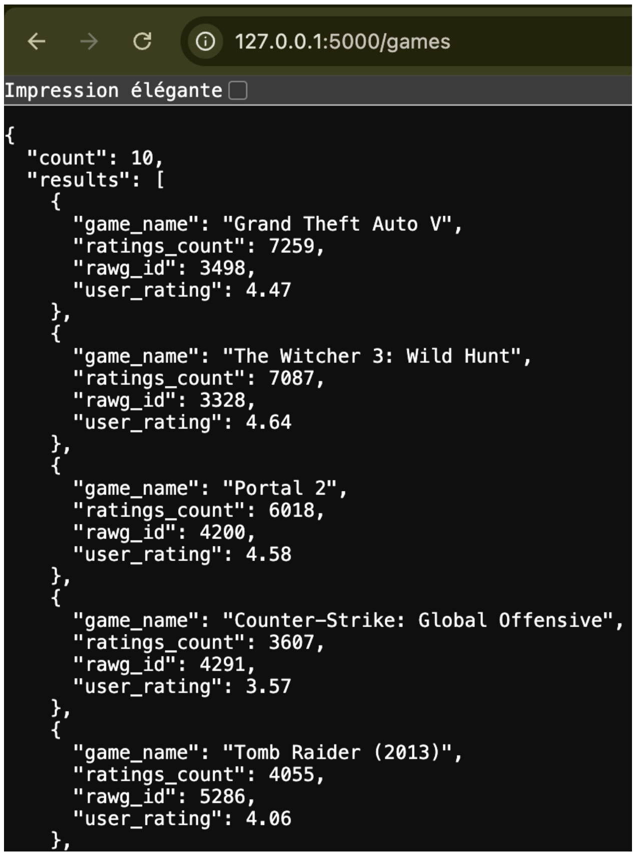
```python
def get_db_connection():
    return pymysql.connect(
        host="localhost",
        user="root",
        password="my_password_hided",
        database="video_game_market",
        cursorclass=pymysql.cursors.DictCursor
    )
```

Fig. 17 : Database connection function



```python
@app.route("/games", methods=["GET"])
def get_games_collection():
    limit = int(request.args.get("limit", 10))
    offset = int(request.args.get("offset", 0))

    conn = get_db_connection()
    with conn.cursor() as cursor:
        cursor.execute("""
            SELECT rawg_id, game_name, user_rating, ratings_count
            FROM games
            LIMIT %s OFFSET %s
        """, (limit, offset))
        games = cursor.fetchall()
    conn.close()

    return jsonify({
        "count": len(games),
        "results": games
    })
```

Fig. 18 : Games Collection Endpoint Function



127.0.0.1:5000/games

Impression élégante ☐

```
{
  "count": 10,
  "results": [
    {
      "game_name": "Grand Theft Auto V",
      "ratings_count": 7259,
      "rawg_id": 3498,
      "user_rating": 4.47
    },
    {
      "game_name": "The Witcher 3: Wild Hunt",
      "ratings_count": 7087,
      "rawg_id": 3328,
      "user_rating": 4.64
    },
    {
      "game_name": "Portal 2",
      "ratings_count": 6018,
      "rawg_id": 4200,
      "user_rating": 4.58
    },
    {
      "game_name": "Counter-Strike: Global Offensive",
      "ratings_count": 3607,
      "rawg_id": 4291,
      "user_rating": 3.57
    },
    {
      "game_name": "Tomb Raider (2013)",
      "ratings_count": 4055,
      "rawg_id": 5286,
      "user_rating": 4.06
    },
```

Fig. 19 : API View

12. Conclusion, Limitations & GDPR

12.1 Conclusions

This project demonstrates how multi-source data can be transformed into actionable insights for the video game industry.

Key outcomes :

- strong predictors of game success were identified,
- machine learning models provided valuable predictive power,
- visual dashboards enhanced interpretability.

The insights derived from the exploratory data analysis can be directly translated into strategic decisions for industry stakeholders :

- Support **market positioning** by identifying the most relevant genre and platform
- Guide p**roduction decisions** by highlighting high-impact features such as multiplayer
- Enable better **marketing performance forecasting** prior to release

Overall, these findings contribute to more informed portfolio optimization for publishers and game studios.

12.2 Limitations

- API rate limits,
- missing ESRB ratings,
- potential data bias toward popular games.

12.3 GDPR Compliance

All data used is publicly available and does not contain personal identifiable information (PII).

12.4 GitHub Repository

https://github.com/jmpbusiness2023-commits/Final-Project--Video-Game-Market-Analysis