

**Data Analytics**

April-June, 2023

“Brocommender”

Robin DEBESSEL

**Table of content**

Table of content………………………………….………………………………………………………………………………….2

Introduction.………………….…………………..………………………………………………………………………………….3

Project Plan, made with Trello……….……………………………………………………………………………………….4

Data collection & Data sources…..…..……………………………………………………………………………………..4

Overall Process……………………………………..……………………………………………………………………….………5

Data Cleaning, EDA and Visualization……………..……………………………………………………………..….….9

To SQL or not to SQL…………………………………………………………………………………………………………….18

Entity Relationship Diagram (ERD) ……………………………………..……………………………………………….18

MySQL Queries ……………………………………..………………………………………………………………………..….21

Machine Learning ……………………………………..……………………………………………………………………….23

Conclusion………………..………………………………………………………………………………………………………..24

**Introduction**

"**Brocommender**" aims to develop a recommendation algorithm that fits to the individual preferences of a gamer using their Steam ID.

The end goal of this enterprise is to generate a short list of game recommendations that aligns with the genres that the user has spent most of his time playing on the Steam platform.

The recommendations offered by the algorithm are intended to be personalized, hopefully leading the users to a higher likelihood of discovering new games they would enjoy and might have missed or last track of, enhancing their engagement with the platform while respecting their real motivations.

*Context that explains my choice*:

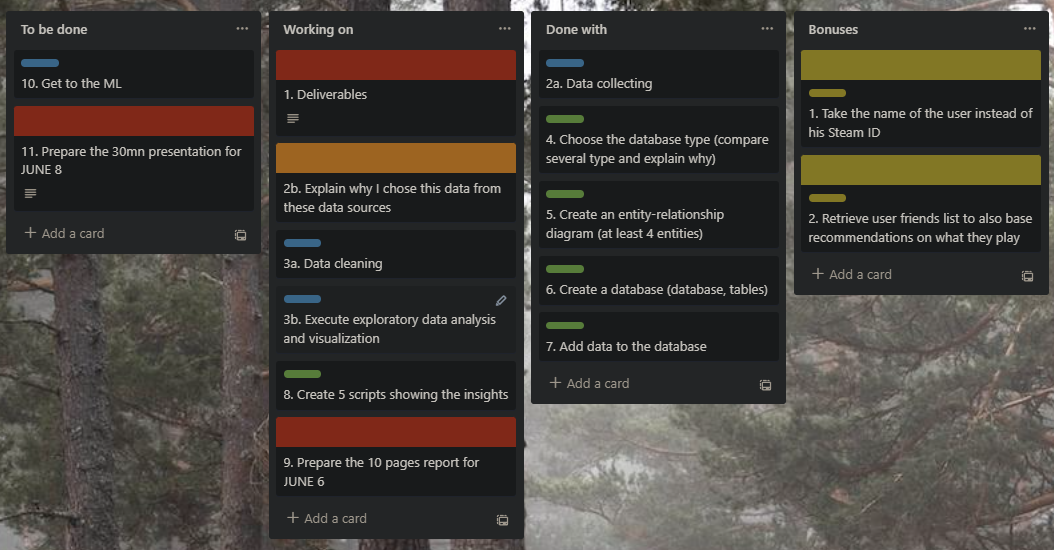
So often we nowadays find ourselves “trapped”, if not directly through plain manipulation, at least through not-always-wanted redirection by algorithms tailored for our satisfaction but also accompanied by an ever present incentive : being the most profitable.  
  
My will was to tackle this issue, through a minimalist and unoppressive interface that doesn’t contain hundreds of references on one page, most of them being featured and displayed for commercial purpose, with sales tags from top to bottom.

The goal for it was, and will stay, to be interactive enough to keep engagement with the user while getting rid of the things they did not ask to see in order to stay on track and really chase and find what they want and not what “we” want them to find and buy.

*A small disclaimer:*

My original ambitions for this project were high and I had to tone them down due to the complexity of tasks.  
“Brocommender” asks to be updated (quite a lot) for it to fit my vision but this will be addressed later in development.

**Project Plan, made with Trello**



*This snapshot is dated from Tuesday the 6th @12:30pm*

**Data collection & Data Sources**

*Data Sources:*

* The primary data source for the project was a dataset downloaded from [Kaggle](https://www.kaggle.com/datasets/nikdavis/steam-store-games?resource=download&select=steam_support_info.csv), a well-known online community of data scientists and machine learning practitioners. The dataset, although four years old, provides a comprehensive snapshot of various game features, which is crucial for the project.

The main attributes considered from the data include the game's genre, the amount of time played, and user ratings.  
  
This dataset is 27000+ rows long and has 18 different columns.

* The second data source we are going to use is the user’s Steam games library.  
  This will be taken care of using the Steam API to dynamically retrieve a user’s account information in order for us to gather the required data.

*The biggest challenges I faced:*

One of the significant challenges during the data collection stage was the use of the Steam API wrapper for Python. As a new concept, I misunderstood it to be the API itself, which led to a few obstacles, during the implementation process.

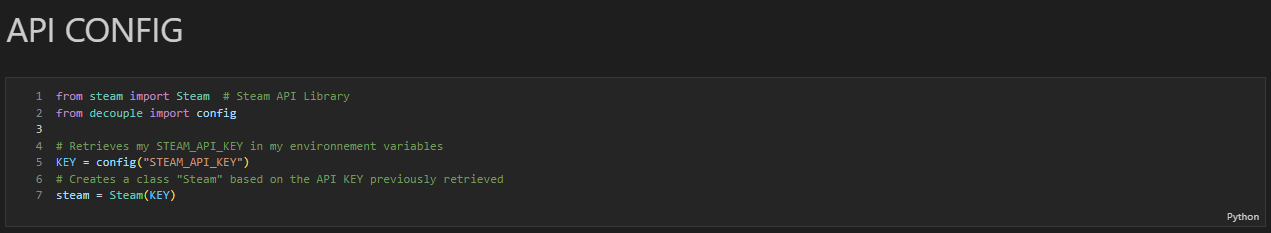
However, once tinkered with, the Steam API wrapper turned out to be beneficial and helped me out with the process of extracting vital information about the games.  
It provided a simple and efficient way to gather essential data elements necessary for our algorithm without having to collect all the data manually from the Steam platform through their basic API, which would have been less practical and probably more time consuming.

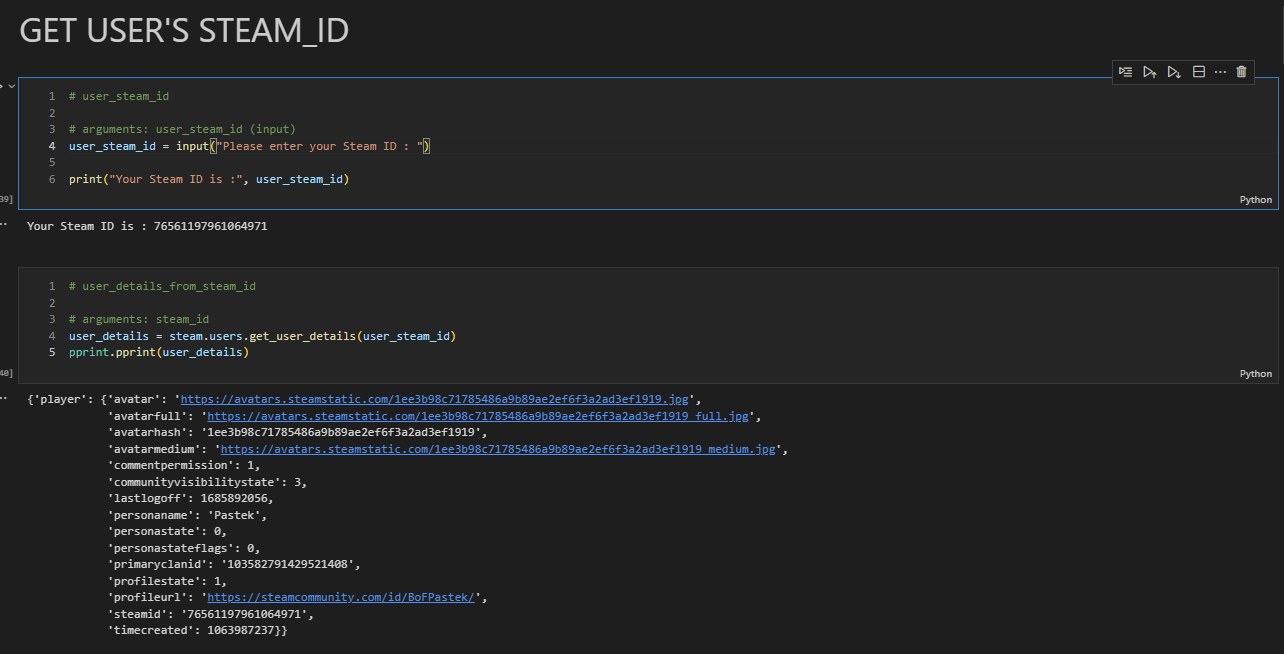
The other big difficulty was managing the complex data structure that the genres and other crucial information were embedded in.  
Extracting them proved to be a tedious task that took numerous hours.

**Overall Process**

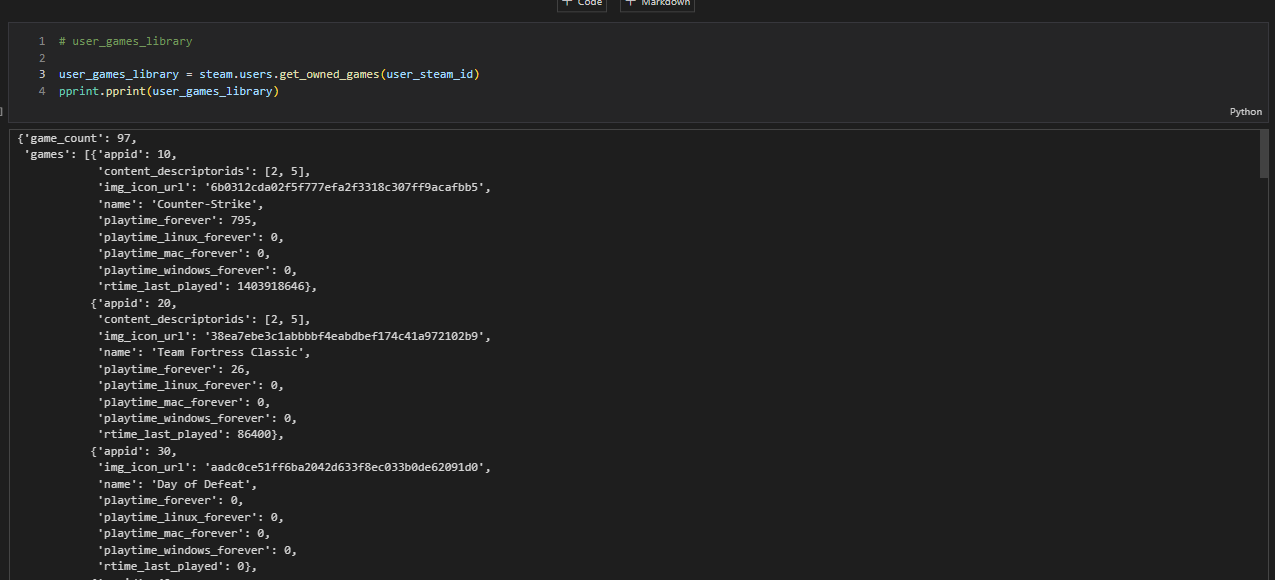
The development process for the "Brocommender" project was organized into three separate notebooks, each serving a distinct purpose for safety and clarity measures.

The first notebook was designed as a platform to install and utilize the Steam API and its Python wrapper. This notebook was instrumental in making queries to the Steam API, gathering the required data that is fundamental to the recommendation algorithm. In the upcoming project demonstration, this notebook will serve as the primary showcase, highlighting how we interact with the Steam API to extract game and user data:

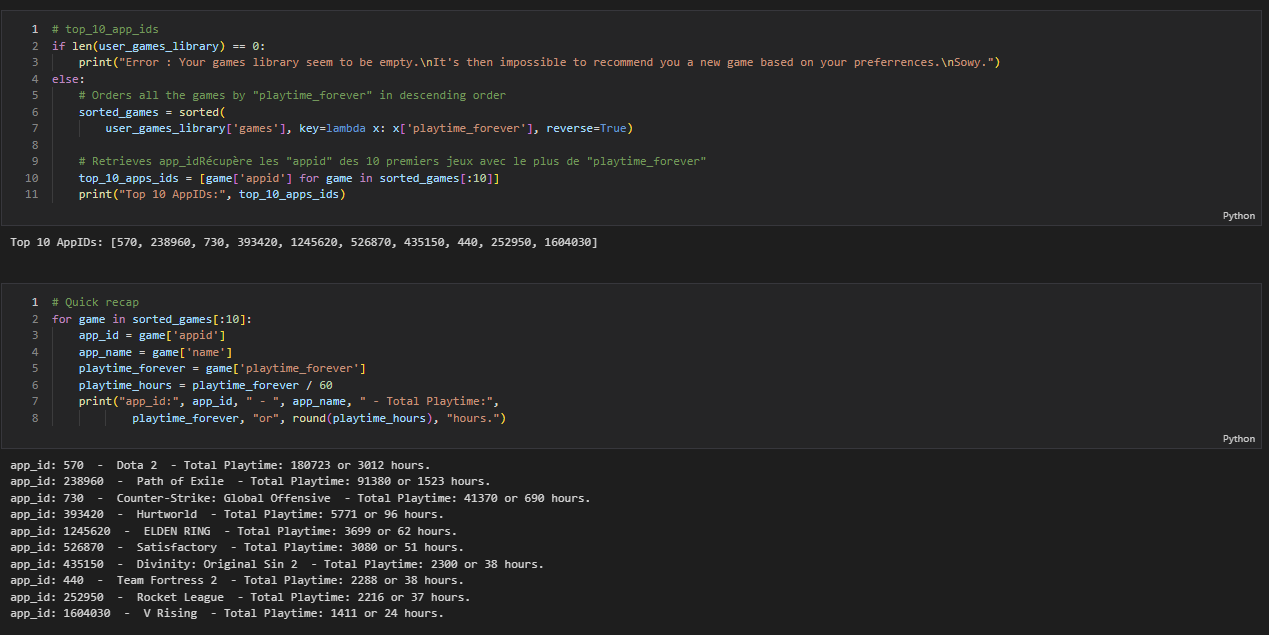




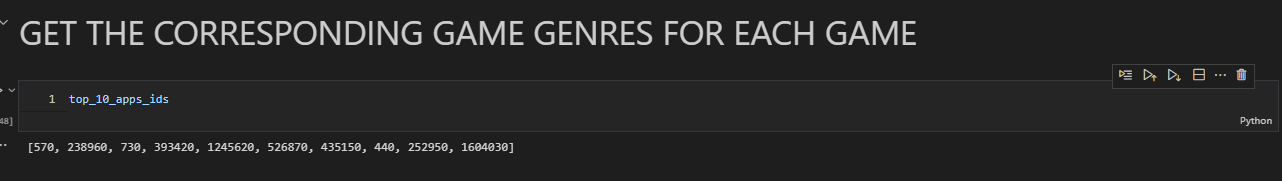
From the user’s data, we can then request to know about his entire games library:

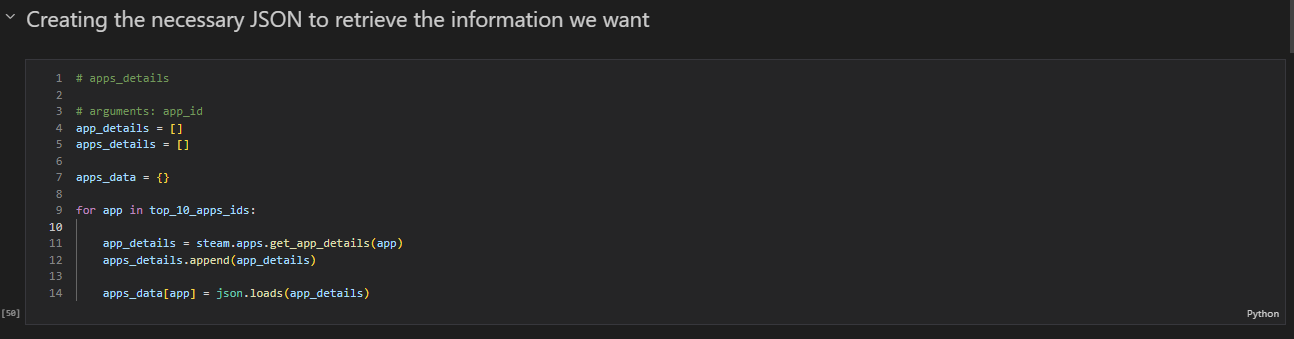


And from this library we are going to retain the 10 most played games, based on the time they have been running:

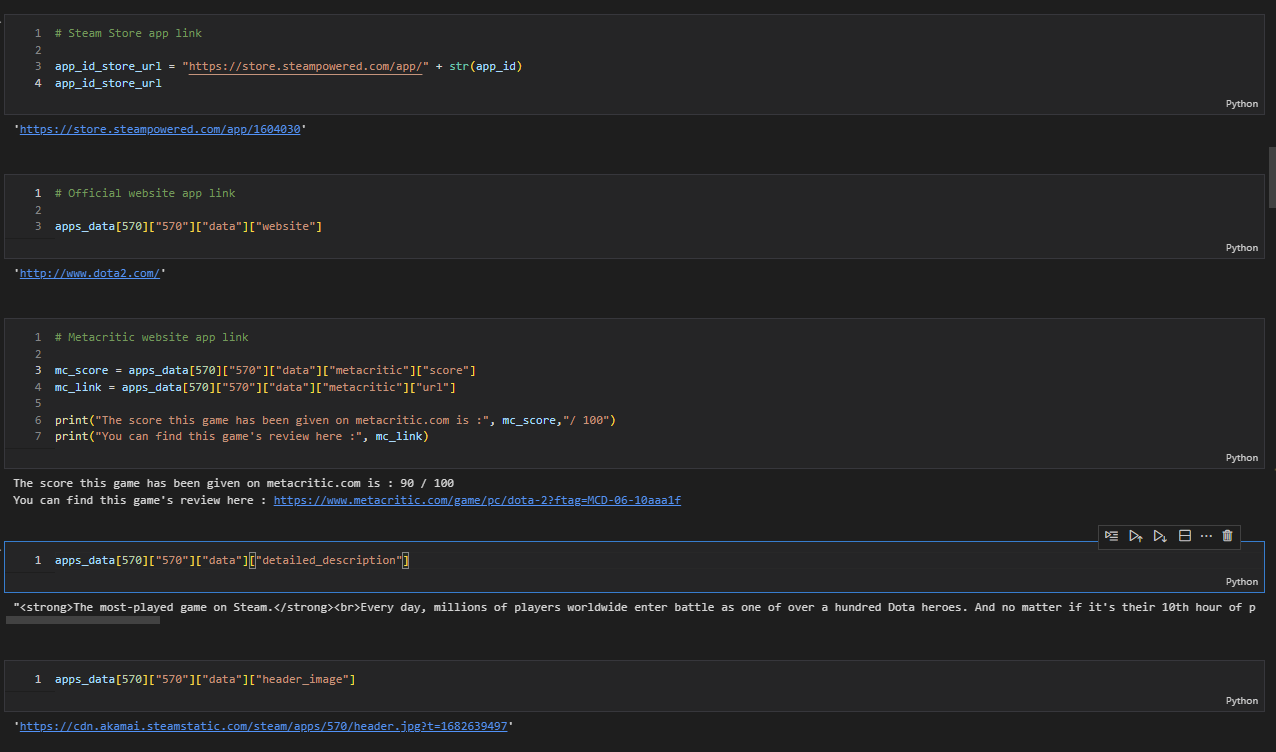


After this we will use the games IDs (app\_id here) to get the information relative to each game or app:

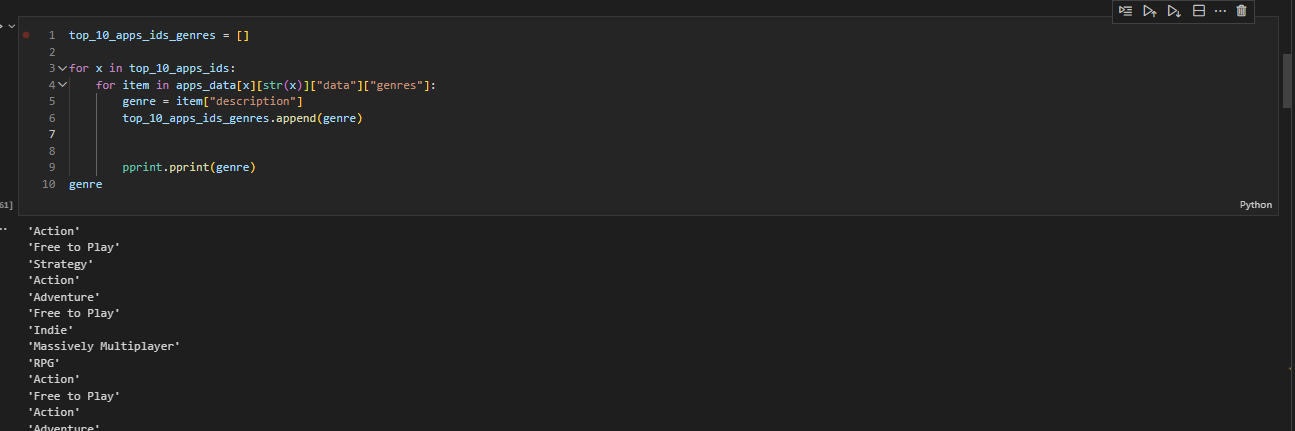


We then use a JSON to hold the retrieved information

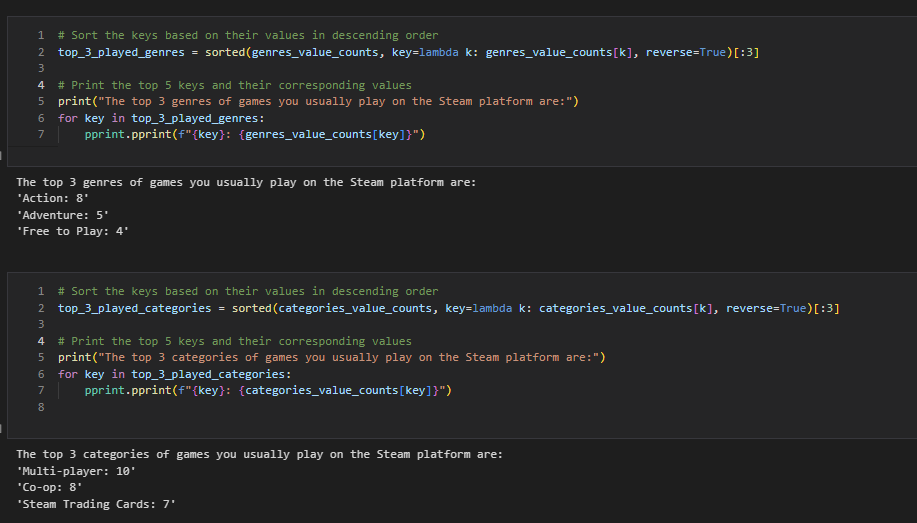
A few fun features we can now work with to offer different links to different websites for the games we want:



Afterwards, we can finally gather all our different genres for each games



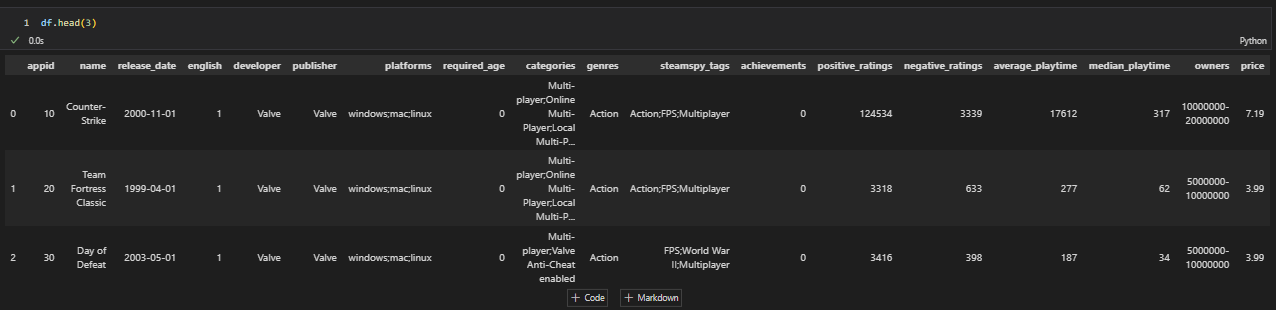
We also do this for the game’s categories in case we need later to build a better recommender.  
  
Then we have to count our different genres but not before translating them to english as sometimes the info we get from the API is not in english. It’s hard to tell why this happens as it really seems to be randomized, especially for the biggest and most played games of the platform.

This is what we end up with!

**Data cleaning, Exploratory Data Analysis and Visualization**

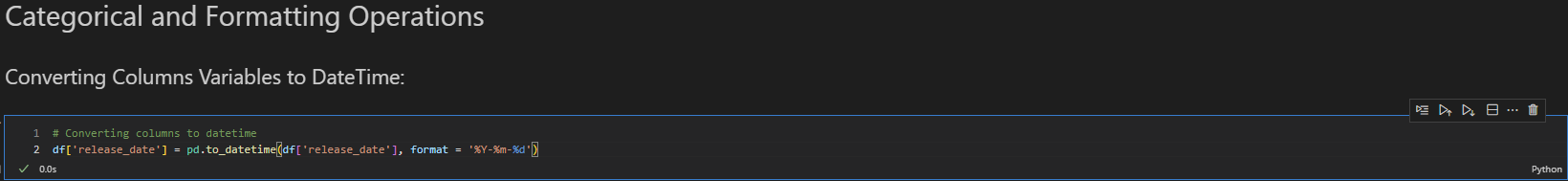
The second notebook was employed to import, clean, and visualize our primary dataset obtained from Kaggle. It has a vital role in the data preprocessing stage of the project. It includes a few steps to clean the data, ensure its consistency, perform some exploratory data analysis through various visualizations.

We then have to export the cleaned data as SQL tables via appropriate data frames. This process will ensure the data fed into our machine learning model is qualitative as we can make it in order to improve the reliability of our recommendation algorithm.  
We will detail this in the following parts of our report.

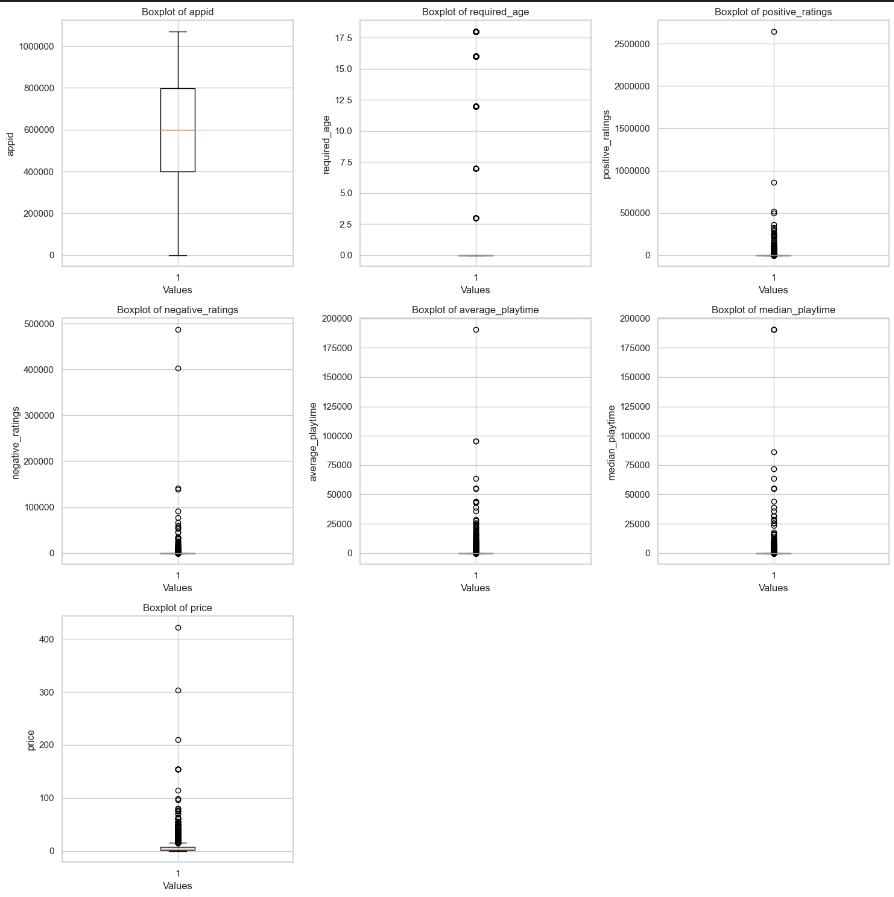
Our data being very clean from the get go, we are quickly going to go through this part of the report. At least when it comes to coding snippets.

We can however meet and greet our first and biggest frustration for this project:  
As you can see in the screenshot, a very interesting column we could use would be the “steamspy\_tags”. They would complement our game genres data greatly.  
  
Unfortunately, after spending a few hours on this matter, I could not find a way to retrieve said tags from any of the game information responses to our request.  
Which means that for this version of “**Brocommender**”, we will have to do without…

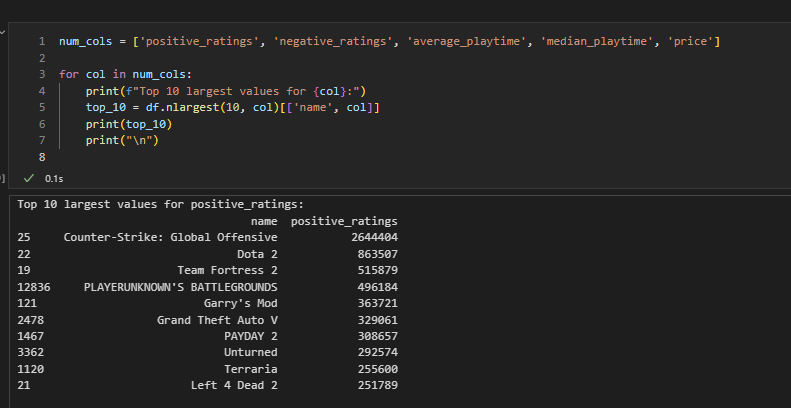
*We can then move onto our next important operation:*

****

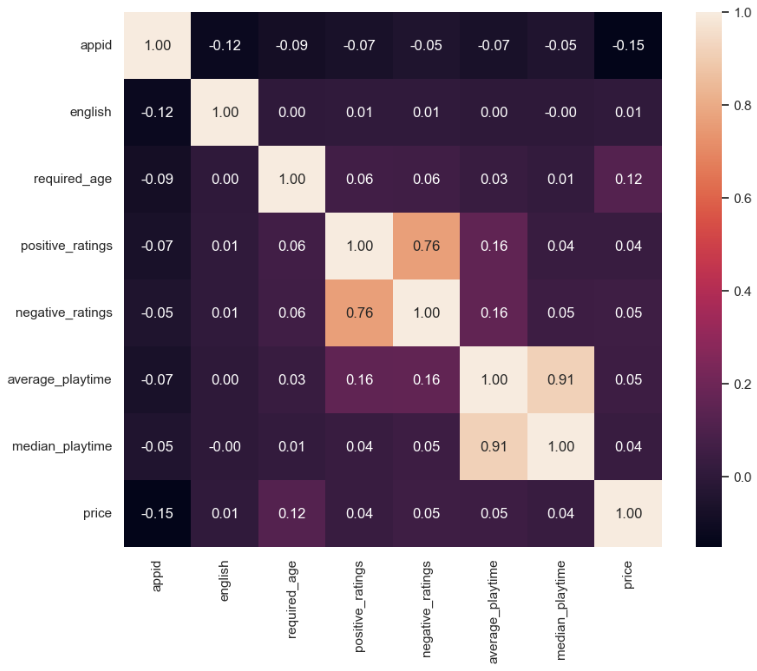
*To begin with visualization, a few whiskers boxes to quickly see what’s what is always good. In a glance we can gather a lot of useful information through this process:*

****

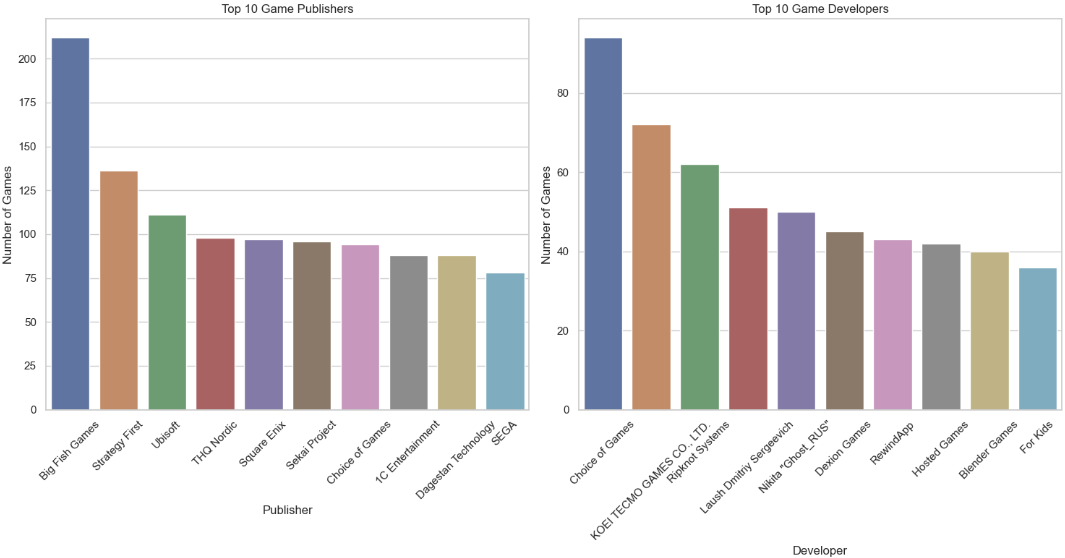
*We will keep our outliers as despite being really big, it seems after closer inspection that they are indeed valid values.*

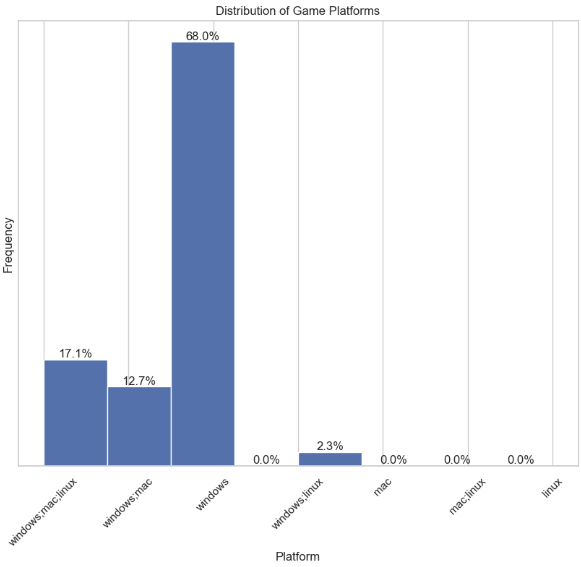


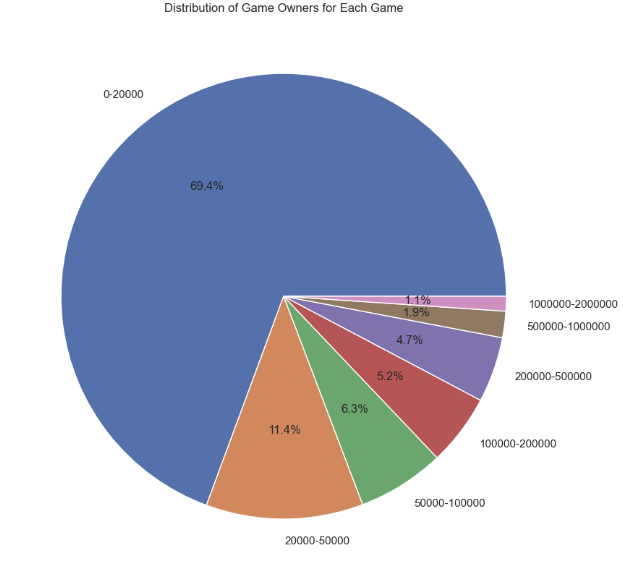
*A correlation map is another handy way for us to check our dataset numerical values*

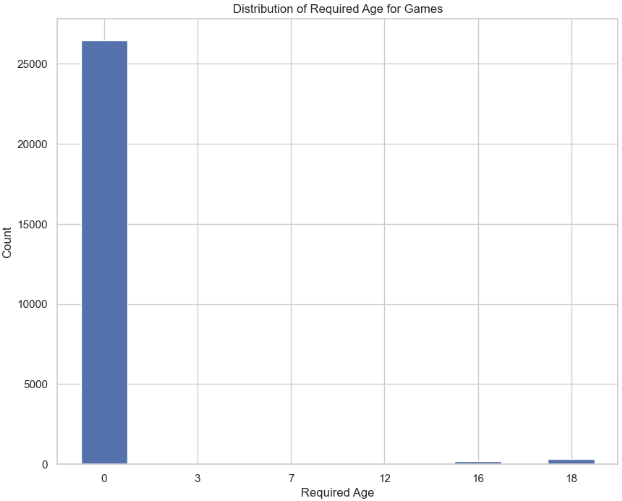


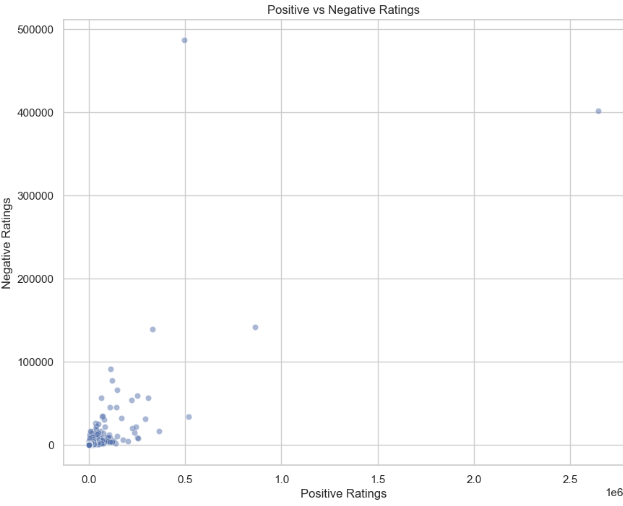
*Then we keep on creating a few plots for us to get a better understanding of the values:*



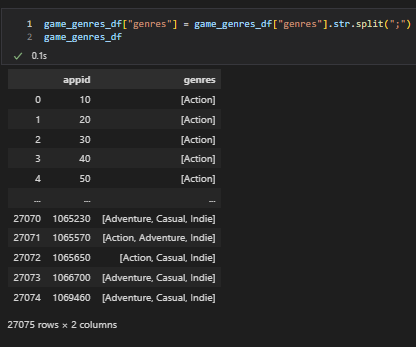
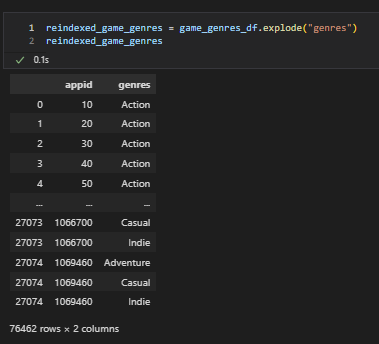
****

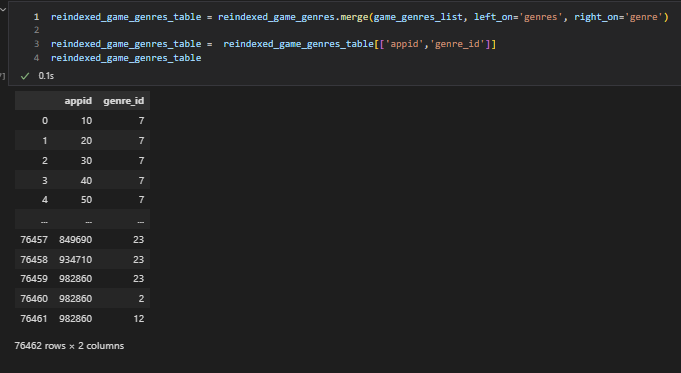
****

****

****

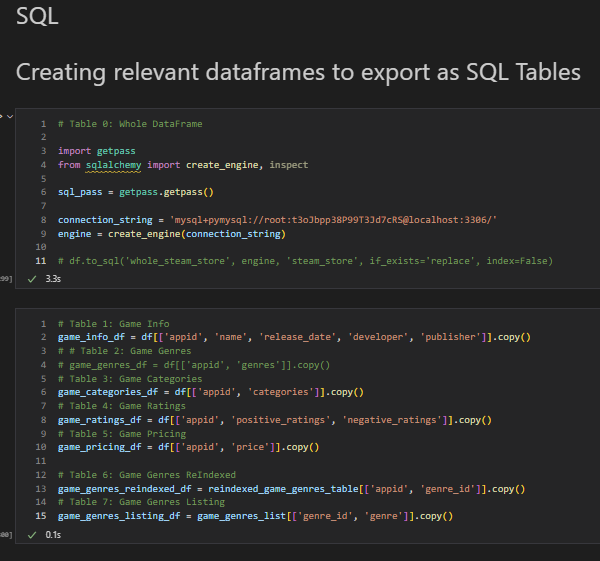
*We will then tackle an important part of the data tinkering as we have to breakdown our different game genres for the SQL part as follows:*

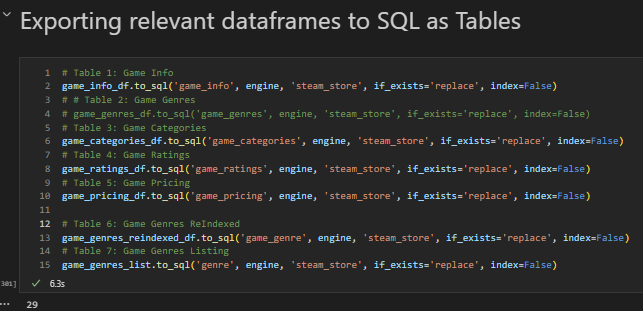
****

****

We then just need to create and export our different tables in SQL.

For this we create the corresponding dataframes before the export instruction.

****

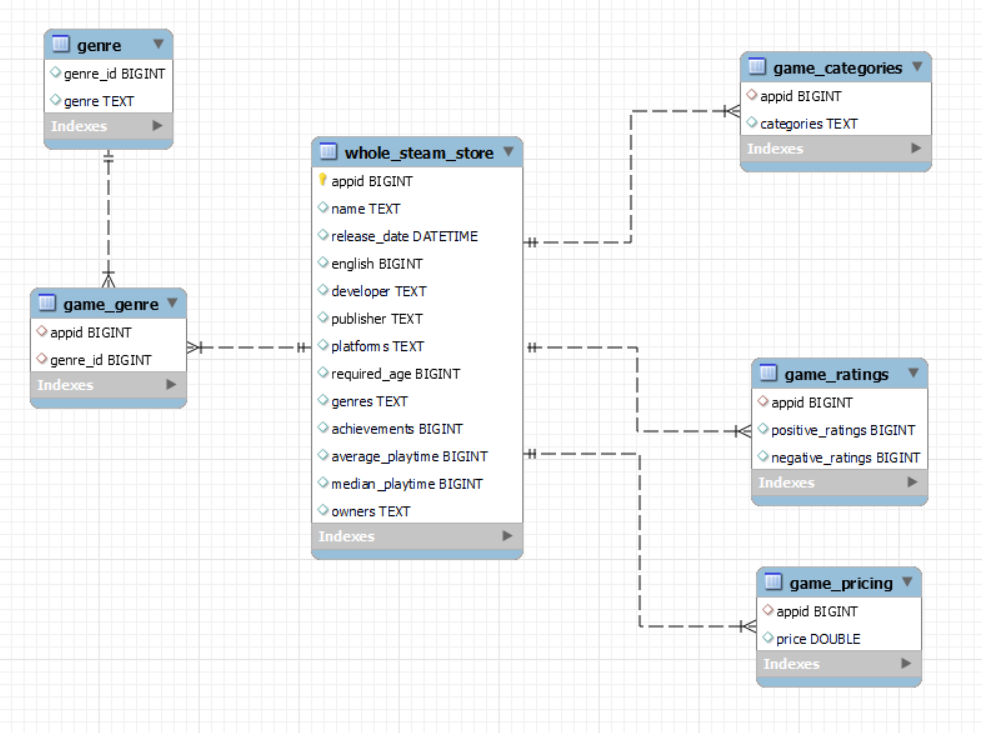
****

**SQL VS other data base types – the argument**

SQL and NoSQL are two different types of database management systems with their own set of characteristics and advantages.

* Relational databases organize data into tables with predefined schemas and use structured query language - SQL - for data manipulation.  
  They provide strong data consistency, integrity, and support for complex queries and transactions. Relational are then suitable for applications that require complex relationships between entities and need ACID - Atomicity, Consistency, Isolation, Durability - properties, such as financial systems, inventory management, and customer relationship management.
* They are characterized by tables with predefined schemas and support for complex relationships, transactions, and queries. Despite my project not being of that size for now, it still seems appropriate to use SQL databases since they are a tool I start to be familiar with.  
    
  Another argument I can add is that, as I expect to add more data or other sources later on to take this project to the next stage, SQL will be optimal due to its capacity for vertical scalability and also the ability to use multi-row transactions.

**Entities Relationship Diagram**



*Tables listing:*

* Table: whole\_steam\_store

Columns: «appid», «name», «release\_date», «english», «developer», «publisher», «platforms», «required\_age», «genres», «achievements», «average\_playtime», «median\_playtime», «owners», «price»

Description: This is our main table which contains our primary key: “appid”

* Table: game\_genre

Columns: «appid», «genres»

Description: This table represents the genres associated with each game. It includes the «appid» column as a foreign key referencing the “whole\_steam\_store” table and a column for the genres the game belongs to. Multiple genres can be stored as a comma-separated list or in a separate table if there are many-to-many relationships[.](https://file+.vscode-resource.vscode-cdn.net/c%3A/Users/theya/Mon%C2%A0Drive/iRonhack%20on%20G-Drive/n/n4.)

* Table: genre

Columns: «genre\_id», «genre»

Description: This table represents the different game genres we found in our database. It includes the «genre\_id» column as a foreign key referencing the “game\_genre” table and a column for the genres the game belongs to. Multiple genres can be stored as a comma-separated list or in a separate table if there are many-to-many relationships[.](https://file+.vscode-resource.vscode-cdn.net/c%3A/Users/theya/Mon%C2%A0Drive/iRonhack%20on%20G-Drive/n/n4.)

* Table: game\_categories

Columns: «appid», «categories»

Description: This table represents the categories associated with each game. It includes the «appid» column as a foreign key referencing the “whole\_steam\_store” table and a column for the genres the game belongs to. Multiple genres can be stored as a comma-separated list or in a separate table if there are many-to-many relationships.

* Table: game\_ratings

Columns: «appid», «positive\_ratings», «negative\_ratings

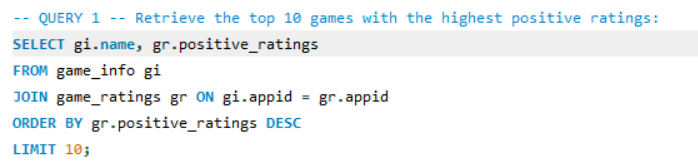
Description: This table stores the ratings for each game, including the number of positive ratings and negative ratings it has received. The «appid» column serves as a foreign key referencing the games unique identifier in the “whole\_steam\_store”.

* Table: game\_pricing

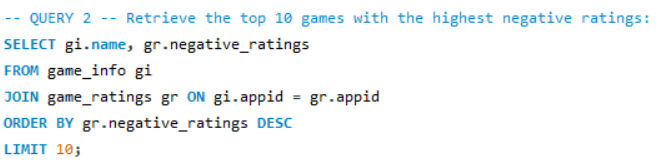
Columns: «appid», «price»

Description: This table contains the pricing information for each game, including the «appid» column as a foreign key referencing the “whole\_steam\_store” table and the price of the game.

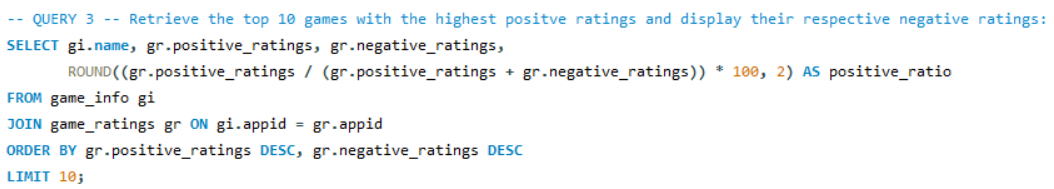
**MySQL queries**

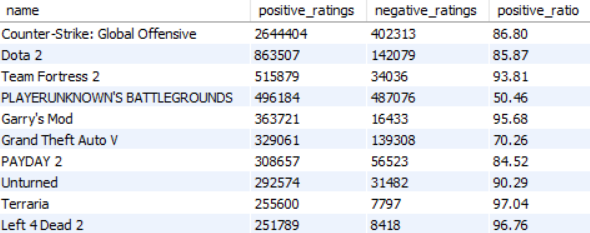
****

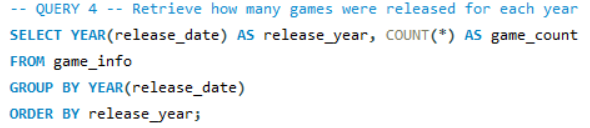
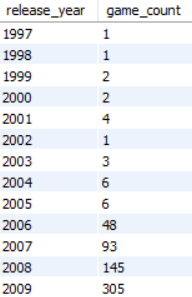
****

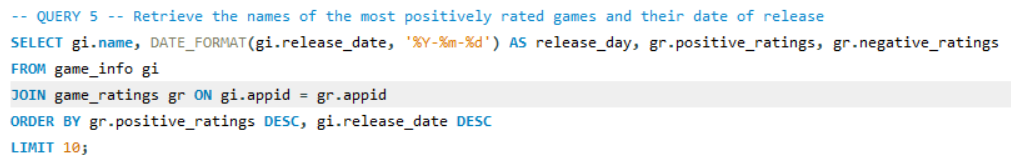
****

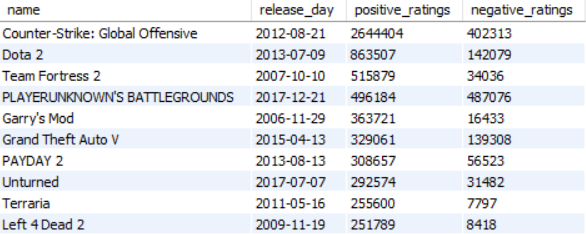
****

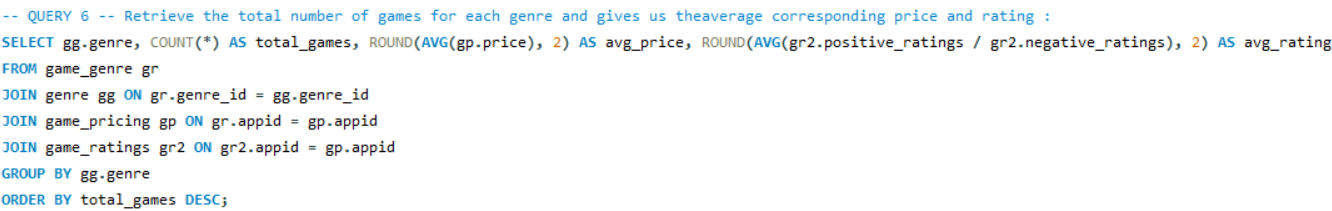
****

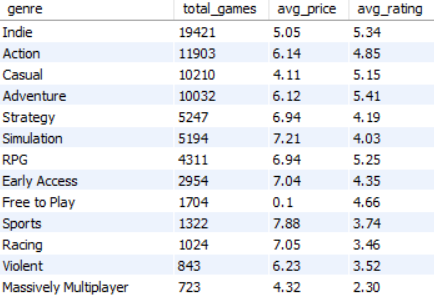
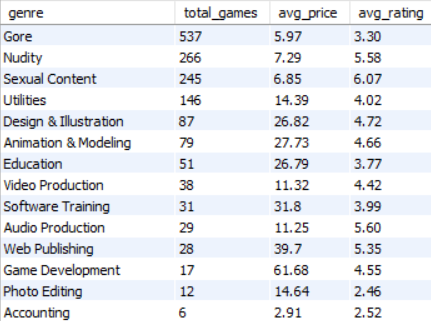
****

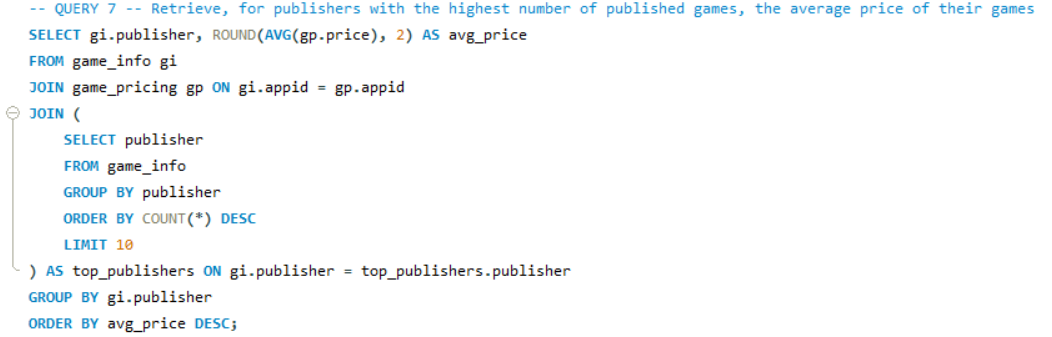
****

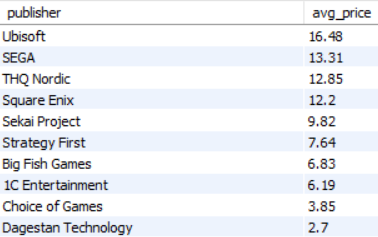
****

****

****

****

****

****

**Machine Learning**

Finally, the third notebook addresses the machine learning component of the project. Here, we will engage in model selection, training, evaluation, and prediction. The focus of this notebook will be to train a model that learns the patterns from the cleaned data to eventually recommend games that a user is likely to enjoy based on their gaming history and preferences.

For that we will address the machine learning aspect of the recommendation system.

This process involves training a model on the pre-processed data, enabling it to learn the patterns and characteristics that dictate user preferences.

Once the model will be adequately trained, it will be used to predict potential game recommendations for an existing user based on their games library, playtime history and therefore global preferences.

While this part of the project is not fully fleshed out yet, we can consider, as the data cleaning is now done, that the rest of the general process will follow these steps:

* Feature selection: This step will involve identifying the relevant features that will be used in the machine learning model.
* Normalization and scaling: Once the relevant features have been selected, they can be normalized or scaled to ensure that they are on a similar scale. (This is especially important for models that are sensitive to the scale of the input features, such as linear regression, support vector machines, or k-nearest neighbors).
* Model selection: After preprocessing the data, we will be able to select the appropriate machine learning model.
* Training and evaluation: The selected model will be trained on the preprocessed data, and its performance will be evaluated.
* Final implementation: Once the model's performance is satisfaying, it will be implemented for use in making predictions. This could involve deploying the model as part of a larger system or using the model to generate predictions for a new dataset.

I am optimistic that the completion of this stage will result in the elaboration of an efficient recommendation system, capable of enhancing user experience on the Steam platform by offering users a different approach when it comes to them finding their next gem.

**Conclusion**

The journey to creating the "Brocommender" algorithm in only a few days’ time has already been filled with great learning opportunities, especially in understanding and overcoming new technical concepts on top of the ones I learnt during my formation here at Ironhack.

This is just the beginning.  
  
The magic of coding and computer science residing mostly in automation and constant improvement, version after version.  
  
I really hope and want to finalize the recommender and make it somehow useful, if not only to me, to anybody willing to try it.