Ebrahim Moosa

Malachai Cravens

Cody Gunter

ETL Project

**Inspiration:**

This project took inspiration from the fact that as we have grown older, we have had less time to enjoy the pleasure of playing video games. As a result, we tend to play games that tend to be rated the best. We decided that analyzing video games in the scope of review scores given by both critics and users, as well as how successful a game was based on units sold. The final database will look at video games, the genres, units sold, and what platforms they were on.

**Sources of Data:**

Being a major entertainment source, the largest in terms of revenue, there are a plethora of sources to look at. Traditional gaming outlets have a deep well of data, specifically in terms of critic reviews and lists of best games. There are also companies dedicated at looking at the sales of video games. The first data set we pulled from is from Kaggle. The data is a web scrap of sales of video games, per game. The next data set we used was scraped from Business Insider looking at the best games with both critic and user scores.

<https://www.businessinsider.com/best-video-games-metacritic-2017-11#50-world-of-goo-2>

<https://vgsales.fandom.com/wiki/List_of_best-selling_video_games>

<https://www.kaggle.com/datasets/gregorut/videogamesales?select=vgsales.csv>

**Extraction:**

Web scraping was done on both the Business Insider and the VGSales pages. The Business Insider scrape gave us the top 50 games, along with the critic reviews and user reviews. Due to the page being static, we did not have to use Splinter to navigate through the web page. Cleaning of the text had to be done to extract the rankings. For the VGSales scrape, we had to loop through the page, as well as utilize try catches to pull the sales data from the page. This page was also static and the information was presented in table form already. Reading in the CSV was of no issue, and had given us more sales data, that was also scraped from VGSales. As a result of the CSV being cleaner than our own scrape, we decided to drop the scrape during the construction of our database.

**Transformation**

Post extraction of the data, cleaning of the data took place. Starting with the VGSales CSV from Kaggle, we removed the second index column along with the region sales. This allowed us to isolate global sales and keep just the one indexing column. We followed this up with isolating the Platform column and saving it off to a CSV. The same was done to the Genre column. This was done as both columns have their own tables in the database that will be referenced when creating the junction tables.

**DATA SCRAPING**

Graphical user interface, text, application, email

Description automatically generated

This is the html format of the Business Insider webpage containing the top 50 video games of all time ranked by Metacritic scores. From the site we needed to scrape the game’s name, critic score, and user score. The name was located in the second header, and the scores were in the first and second paragraph in the div which was located in a second div containing the entire table. Once each text entry was extracted, a few extra cleaning practices were applied including removing quotes, ranking, and surrounding text in the score paragraphs.

Once the data was scraped from Business Insider, the VGSales data was narrowed down to reflect the same games. The two CSVs were then merged so that all unique game data was on the same dataframe.

The two junction tables were created using the Primary Keys from the three base tables we created, videogames (which had the names, sales data, year released, and both critic and user scores), genres (stored the different genres and giving them the serial primary keys), and platforms (stored the data for the different platforms games were release on and giving them their serial primary keys). The junction tables give us the genre per game and the platforms on which you can play said game. The platforms you can play each game was a unique challenge as it was a many to many table and we were unsure on how to approach this. We ended up having to join the genre Foreign Key to our original merged dataframe.

**Load**

Diagram

Description automatically generated

The ERD diagram was created using the quickdatabasediagram website (https://www.quickdatabasediagrams.com/). We created the original tables using the diagram but realized that some of them will need to be edited so that we could import our data in without getting errors. The loading was completed in jupyter notebook using the sqlaclchemy library and pandas. Once each CSV matched the table that it corresponded to, we wrote our data into postgresql.

When all the tables were populated, we began to do our querying and data visualizations.

**Querying**

Table

Description automatically generated

Here is an example of querying the data to mold our visualizations.

**Conclusion**

Chart

Description automatically generatedChart, histogram

Description automatically generated

Once we were able to join the genre and the platform into the videogame table, we wanted to find the results of how each genre and platform performed. We found that almost half of the games were action, followed by shooter and sports games so nothing all too surprising about that. For the platform chart, there again wasn’t anything all that surprising with PC being at the top followed by PS3 and the Xbox360.

**Limitations**

The time constraint was the biggest limitation that we have had. There were plenty of sources out there but due to the time constraint we had to pick sources that were smaller in scope and easier to clean and scrape. We would also take the time to do more exploratory data analysis in terms of what games sell the best broken down by both platform and genre per platform. In order to do that, we would have needed to pull data on platform sales as well.

**Further References:**

<https://stackoverflow.com/questions/18172851/deleting-dataframe-row-in-pandas-based-on-column-value>

<https://stackoverflow.com/questions/49574817/fill-a-new-pandas-column-with-row-numbers>

<https://stackoverflow.com/questions/63992639/pandas-to-sql-append-vs-replace>