ETL Project Proposal & Report

Jason Gabunilas, Ryan Little, Jacqueline Sailer | [GitHub Repository](https://github.com/mrryanlittle/etl-project)

# Proposal

## Extract

We have obtained two datasets regarding video games. The first list contains a list video games released between 2011 and 2019, along with their publishers, detailed metacritic scores, and several other attributes. A second dataset contains a list of games sold on the video game distribution platform Steam. The table contains many interesting data points, including number of owners, number of customer recommendations, etc. Our goal will be to parse the metacritic table for games that are available on Steam, and display these games’ ratings (positive, neutral, negative scores from critics and players), game ratings specifically for RPG games, number of players that recommend the games.

## Transform

Both tables contain many columns, most of which will not be needed for our analysis. Our transformation will remove most of these columns, keeping only the ones that will be needed in our final display. Additionally, the datasets have duplicates that will need to be cleaned. Finally, Steam only sells games on the PC platform, whereas many games are available on multiple platforms. To avoid nonsense joins, the metacritic table will need to be filtered down for PC games only.

## Load

The cleaned tables will be uploaded to a SQL database and inner-joined to only focus on games featured in both sources. Each team member will perform a separate extract, join, and clean effort to generate final tables with the data of interest. See work breakdown below.

## Datasets

Metacritic Ratings Data: <https://datasetsearch.research.google.com/search?query=video%20game%20ratings&docid=Rji9ydv%2BudtZQuLIAAAAAA%3D%3D&filters=WyJbXCJpc19hY2Nlc3NpYmxlX2Zvcl9mcmVlXCIsW11dIl0%3D&property=aXNfYWNjZXNzaWJsZV9mb3JfZnJlZQ%3D%3D>

Steam Platform Data:

<https://data.world/craigkelly/steam-game-data>

## 

## Work Breakdown

[Jason](#_4xuqgyywlq7r) - All Steam games by genre, ESRB rating, and number of recommendations by players

[Jacqueline](#_wwiq5bka3sg9) - All Steam game ratings (positive, negative, neutral, critic and users)

[Ryan](#_dnyhaayppxyb) - Highest rated RPGs available on Steam by release date, according to metacritic and user scores

# Report

The original data sources were shared between the analysis of all group members, consisting of the following two CSV files:

- datasets\_165618\_377201\_metacritic\_games.csv

o This file contains records for over 5000 video games and documents various numerical ratings as chronicled by the rating service Metacritic from 2011 through 2019. Along with the ratings, the table also catalogues various other metadata including game genre, developer, ESRB rating, and so forth. The video games in this file span multiple platforms (PC, Xbox, PS4, etc.).

- games-features.csv:

o This file contains a list of over 13000 video games distributed by the video game distribution service Steam. It documents various pieces of information for these games, including long text game descriptions, game price, the number of players that recommend the game, system hardware requirements, and a series of Boolean attributes describing whether the game belongs to a particular genre.

# Jason - Recommendations Analysis

## Extract

For the recommendations analysis, the Metacritic and Games Features CSV files were first read into pandas as dataframes.

## Transform

Once imported, the dataframes were cleaned and transformed as follows:

- Metacritic dataframe

o All columns were dropped except for those describing the game name, gaming platform, genre, and esrb rating

o The dataframe was then filtered for games that are on the PC platform. This is because the Steam distribution service deals solely with PC games, whereas many games are available on multiple platforms. Thus this filtering was required to prevent nonsense joins.

o Any duplicate rows were dropped if present

o The above transformations resulted in many rows being removed, causing the index to become non-sequential. Thus the index was reset to create a sequential index. Then the newly-created index was added as a column to the dataframe called “id”, the intention being to use this column as the primary key for the SQL table. Then the original, non-sequential index was dropped from the dataframe

o With the dataframe having already been filtered for PC games, the platform column was no longer needed and was also dropped.

o The remaining columns were renamed to SQL-friendly names that would be used for the corresponding SQL tables.

o Finally, some rows contained unpopulated values for the ESRB ratings column. All such rows were dropped

- Steam games features dataframe

o All columns were dropped except for the those containing the game name and the number of recommendations

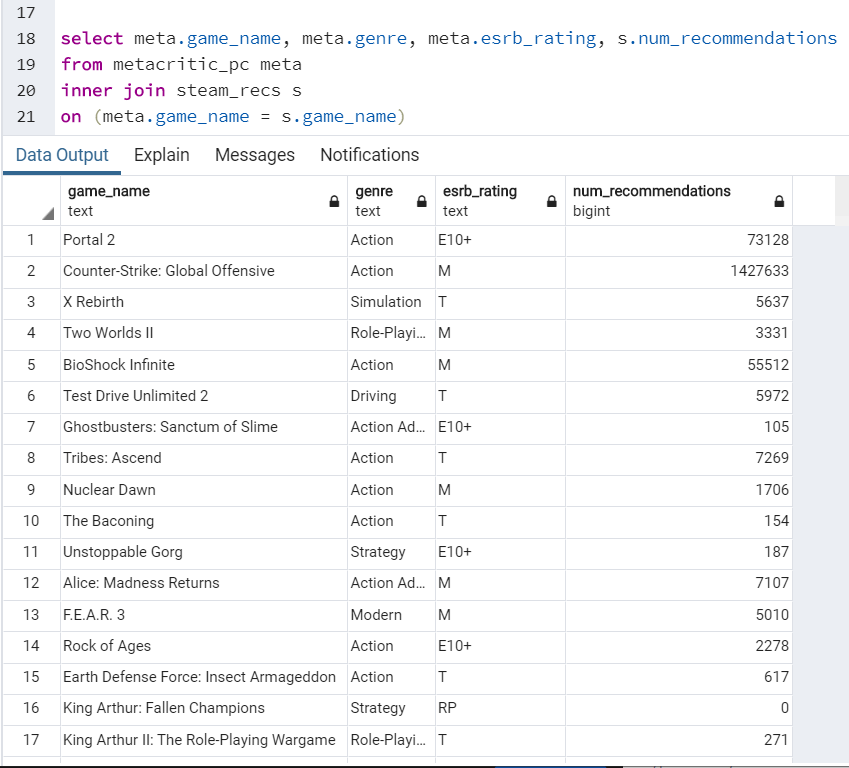
o Duplicate rows were dropped

o The columns were renamed to descriptive SQL-friendly names that would be used for the corresponding SQL tables

o The above transformations resulted in some rows being removed, causing the index to become non-sequential. Thus the index was reset to create a sequential index. Then the newly-created index was added as a column to the dataframe called “id”, the intention being to use this column as the primary key for the SQL table. Then the original, non-sequential index was dropped from the dataframe

## Load

Due to the relative simplicity of the final dataframes and the ease and versatility of performing table joins in SQL, the dataframes were loaded as separate tables into a SQL database created specifically for this project using sqlalchemy. Finally, a query was performed to join the two tables on the game name, yielding a final query result showing game\_name, genre, esrb\_rating, and number of recommendations. The final query contains about 300 rows

****

## 

# Jacqueline

## Extract

The ratings breakdown is primarily in the Metacritic Games CSV file, which contains columns for positive, neutral and negative rating counts for critics and users, respectively. However, for this database, we are only focusing on titles that were released on Steam. Thus, for reference, I extracted the list of games in the games-features CSV, which are all on Steam. Despite that spreadsheet’s plethora of columns, only the game name itself (denoted in the ResponseName column) will be relevant to our ETL process. In the Jupyter notebook, I used Pandas’ read\_csv() function to convert both CSV spreadsheets into data frames that could be tailored for upload.

## Transform

To transform the Metacritic Games rating breakdown spreadsheet, I first created a sub-data frame from the original CSV that only contained the columns pertaining to game, as well as positive/neutral/negative critics/users, respectively. Because many of the titles appear in the data frame multiple times, I cleaned the new data frame with the drop\_duplicates function applying to the game column. Since the game title is supposed to be our index, I set the index to the game column, thus eliminating the default ID number list that has no connection to the game-features CSV file. Finally, I renamed the columns to correspond to the column names being used in the SQL database we’re using (i.e., “positive\_critics” becomes “critics\_positive”, as I feel the database will look better-organized if the columns are ordered by critics and then users).

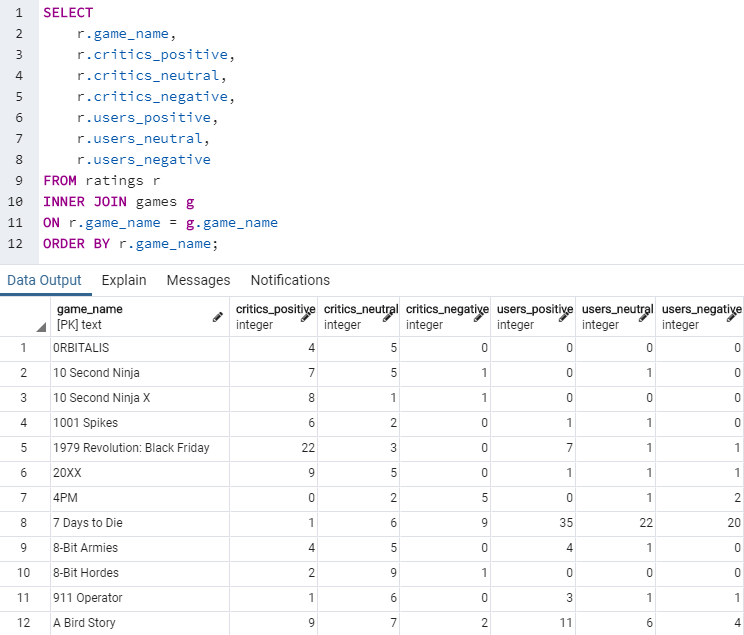
The game-features CSV was transformed through largely the same process as the Metacritics Games file. In this case, however, our only column to extract is ResponseName, which we will rename game\_name to correspond to the ratings breakdown data frame. Once I perform a SQL query, the two tables will be *inner*-joined on the game\_name column so that only titles featured in both original spreadsheets are represented in the final result.

## Load

Before uploading the transformed data frames into SQL, I first designed a schema to create the ratings and games tables (corresponding to the ratings breakdown and Steam games list data frames, respectively) into a new database, which I called game\_ratings\_db. Subsequently, I created a query to pull data from the two tables and inner join them on game\_name. I ordered this joined table alphabetically by game\_name, as this will make it much easier to navigate.

With the query now functional, I finally loaded the two data frames into game\_ratings\_db. I established the connection string as linking to games\_ratings\_db in the postgres server and used create\_engine to build an engine used to upload the data. First, I loaded the ratings breakdown data frame (named “ratings\_only\_df” after transformation) into the “ratings” table I had created with the SQL schema. For the list of Steam games (named “steam\_df\_relevant”), I loaded the data into the “games” table. It was imperative that I dropped duplicate games and set the index to game\_name in both data frames, as only then would they be successfully loaded into the database.

Upon executing the SQL query in pgAdmin4, I got a table consisting of 1,252 different Steam games and their rating breakdowns. For clarification, the transformed ratings\_only\_df (with duplicates removed) was 4,018 rows long, while steam\_df\_relevant was 13,199 rows long. Because of the inner join, only games present in both data frames were represented by the SQL query.



# Ryan Little

### Top Role-Playing Games (RPGs) According to Player and Metacritic Scores

### Extract

1. Rename csv files to something easier to identify and type.
   1. Datasets\_165618\_377201\_metacritic\_games.csv renamed to metacritic.csv
   2. Games-features.csv renamed to steam.csv
2. Imported csv files into a notebook using Python and Pandas.

### Transform

1. Transform csv file data into dataframes for each csv
2. Create new dataframe from metacritic dataframe which has only the columns which are relevant, rename columns
3. Create new dataframe from steam dataframe which has only the columns which are relevant, rename columns to match with metacritic dataframe
4. Using pd.merge I merged the two cleaned dataframes into a single dataframe
5. After noticing multiple release dates from steam dataframe, the steam release dates are dropped and the metacritic release dates are kept, renaming the column from Rlease Date\_x to Release Date.
6. Duplicate release dates are removed, keeping the first release date of the duplicates.
7. The Genre column is checked to remove all values that aren’t the string ‘Role-Playing’
8. With a clean database of just RPGs with game title, release date, only RPG, metacritic, and player scores, the database can be split into two tables to put into the SQL database: Metacritic Score and Player Score.
9. Metacritic score is sorted by Highest Metacritic Score, highest score at the top of the column
10. Player Score is sorted by Highest Player Score, highest score at the top of the column

### Load

1. Created a new database in PostgreSQL called rpg\_games using PGAdmin
2. Created a connection to postgres
3. Created engine to load into rpg\_games
4. Used df.to\_sql() function to add Player Score and Metacritic score to their own tables