Thor Ryan

Jonathan Tran

Data Analytics Bootcamp

ETL Project Report – Steam Games

**Extract**

For this ETL project, we wanted to gather data regarding games available on the popular platform, Steam. We sourced/downloaded our data from two sites: <https://www.kaggle.com/trolukovich/steam-games-complete-dataset> and <https://steamspy.com/> .

Both datasets were available in csv formats, although we chose to use the excel file from the second source (steamspy) to vary it a little and see how it would differ. With these formats, we determined that utilizing pandas would be most efficient. For the first dataset from Kaggle, we used the pandas.read\_csv() function to put the data into a dataframe. Similarly, we used the pandas.read\_excel() function to extract the second dataset from steamspy.

**Transform**

Starting with the original dataset from Kaggle, we cut down on unneeded columns, leaving only the name, review percentage, release date, developer, genre, and original price. For the steamspy dataset, we only wanted the name and number of owners columns. After getting only the desired columns, we merged the two dataframes on the game names using the pandas.merge() function. Luckily, the spelling and capitalization for the names were the same for both datasets, so there were no issues merging. The merging also cut down on the number of rows in the first dataset from about 40k to 239 (thanks to the smaller second dataset).

The first thing we saw that we wanted to change was the reviews column. We were only interested in the percentage while each row had extra text with it. To fix that, we iterated through the column, split the text, and set the value to only the percent portion.

Afterwards, we dropped any rows with incomplete data (using the dropna() function). This brought the number of rows to 223.

Next, we changed the non-dollar values in the price column to “$0.00” (since they were “Free”, “Free to Play”, or had other text that made it clear that they were free). For this, we used the mask function to replace those values.

At this point, we realized that all of the datatypes in the table were strings. We then changed the datatype of the release date column to datetime (using pandas.to\_datetime()), price column to float (first taking out the “$” using the replace function, then using the astype function).

Lastly, in the owners column, we replaced the “..” between the numbers (meant to indicate a range) with “-“ to make it easier to read/understand.

**Load**

For the load portion, we determined that a noSQL database would fit best since we only had a single table and no need for relationships. To load the dataframe into MongoDB, we first used the to\_dict() function to change it into a dictionary format, then simply used the insert\_many() function with pymongo to complete the task.