** Extraction:** your original data sources and how the data was formatted (CSV, JSON, pgAdmin 4, etc).

**What we did.**

After some research, our group decided to gather data of movie and tv show ratings. We gathered our data through Kaggle where we found a csv file (<https://www.kaggle.com/shivamb/netflix-shows>) containing data of tv shows and movies from the Netflix database. The csv file was read into Pandas and converted into a dataframe.

Similarly, we also found a JSON file on Kaggle, containing data from IMDB. The file was read into Pandas as a JSON file and then converted into a pandas dataframe and saved as a csv file.

Lastly, we scrapped the Rotten Tomatoes website (<https://www.rottentomatoes.com/browse/dvd-streaming-all>) for data containing movies and tv shows and their ratings. We employed the use of BeautifulSoup and splinter to append the data into lists. The data was then converted into a pandas dataframe and saved as a csv file.

**Problems Encountered.**

During the scrapping process of the Rotten Tomatoes website, we formulated a code which would gather the data from the website. However, the website had more than 18500 separate pieces of data. The process took over 2 and a half hours and the Rotten Tomatoes website kicked us out after we had gathered data for 9500 titles.

Additionally, the webscrapping of the Rotten Tomatoes website was a little different to what we had learn in class. Instead of having a ‘Next Page’ button, the website employed the use of a ‘show more’ button. The show more button added more titles, but did not change the page of the website. Therefore we researched and found (‘browser.find\_by\_text('Show More').click()’) function.

Our group initially wanted to gather data in regards to mobile games from both the Google Playstore and the Apple Appstore. We decided that we wanted to obtain the data from Kaggle and by scrapping both the Google Playstore and Apple Appstore websites.

We soon realised that our knowledge on webscrapping was not sufficient to handle the complex scrapping methods required for the Appstore. Similarly, the JSON file from the playstore data was so big that it kept crashing our Jupyter Notebooks. We quickly decided to cut our losses and find a topic that was more realistic for our skillset level.

** Transformation:** what data cleaning or transformation was required.

We needed to transform all our data into a pandas dataframe. Therefore, we read the csv, JSON and converted the webscraped data into pandas.

After converting the data files, we cleaned the data by dropping columns that were of no use to us and by filtering and deleting any rows which had N/A values.

We decided to filter the Netflix data by movies, as 80% of the data related to movies.

Additionally, as some data only had the release year, we decided to convert all release dates to reflect the year in which it was released. This was to ensure that the data was consistent across the dataframe.

We merged all 3 dataframes together through a inner join on title, to create a nf\_t\_imdb dataframe.

** Loading:** the final database, tables/collections, and why this was chosen.

We created a table on PostgreSQL and then connected the database from Pandas into PostgresSQL.

The reason we chose to push the data into PostgreSQL was because the IMDB ,NETFLIX, Rotten tomatoes data were relational data. The data was relational as they share similar columns such as movie title, year and ratings.

**What we can do with this data:**

With this data, we are able to find information of movies on Netflix. By obtaining data from IMDB and Rotten Tomatoes, we are able to compare the rating scores of individual movies.

Alternatively, the database could support analysis for movies by Release Year and the Year that it was uploaded onto Netflix.