**Project Report**

**Extract**

Data sources include:

[**https://data.world/mathiasburton/beer-curious/workspace/file?filename=styles.csv**](https://data.world/mathiasburton/beer-curious/workspace/file?filename=styles.csv)

[**https://data.world/socialmediadata/beeradvocate**](https://data.world/socialmediadata/beeradvocate)

The first data source “beer-curious” had extensive information on beers, breweries, styles, locations, and various other metrics. The second data source included social media data from “BeerAdvocate” which had over 1 million beer reviews on ratings from aroma, appearance, taste, reviewer palate, and overall score. The goal was to clean the data and create tables for interesting analysis. Data was formatted from excel cvs’s and put into pgAdmin 4. Problems occurred from using the review data set because it included over a million rows of data. How it was cleaned and transformed is discussed below. Data was formatted from excel cvs’s and put into pgAdmin 4.

**Transform**

First, we imported beer\_review and breweries into our postgres database beer\_lovers. We only wanted to focus on the beers that we had in breweries so we did a left join onto breweries to successfully do this. After we successfully did a left join, we filtered down the fields that we believed were necessary for the viewer to see. By doing this, it left a good amount of empty spaces in our dataset for beers that were not found. To eliminate these empty spaces we chose to drop all rows that had a null value in place for the brewery name. My logic for this was there would be no value in having those rows there if they did not have a brewery name. In addition, we added a new column in the reviews\_breweries that took the average of the four fields used to review the beers. Afterward, we also transformed the breweries to cleanded\_breweires by using ‘case when’ functions to replace null values with either a ‘no description’ or ‘no value’. In respect of creating new tables from the original tables, we were left with the task of adding primary keys and foreign keys to our dataset. To do this we chose to use the ‘ALTER TABLE’ function and the constraint, foreign key reference function to formally give the brewery\_id and id fields their correct key.

**Load**.

The Database used for loading was PostgresSQL due to are knowledge and background with SQL. It also allowed us to store the large data sets (over 1 million rows) easily. We chose these tables because breweries\_cleaned gave an overview of each beer that was in the data set. The reviews\_breweries gives only the beers that are in the breweires\_cleaned data set and thus gives us an easy way to connect these tables through their brewery\_id. Additionally, the categories allows us to see what “category” each beer would fall into.