**E**xtract: When extracting data, we looked on Kaggle and found a csv file that had information about a bunch of different cocktails. We found the cities data in a csv format from the website called Simple Maps.

**T**ransform: For transforming the data we used pandas in Jupyter Notebook. By reading in the csv files as pandas dataframes we were able to clean the data and reformat some of the column names to better organize the database we created. One of the things we did was to create a city\_id table to create a reference that can link all of the tables based upon their given city id. This would help them look up the name of the city that was being described.

**L**oad: for loading the data coding was performed in VS Code and Spyder. Given the difference in what the two applications view as the current working directory we made code to toggle between the two. so, the file locations would not error out. Upon completing the newly transformed CSVs in the aforementioned section, they were made into the following:

* DF1 – is the “Cocktail” CSV and the life of the party. Not much was performed on this dataframe other than renaming columns and dropping some final NAN’s which were spotted.
* DF2 – is the “Cities” CSV again with minor changes to the column names. This dataframe is then bifurcated into two dataframes “City” and “tz” which hold the city information and timezone information, respectively.
* City- the child of DF2. The only modification was having the ‘Timezone’ column removed.
* Tz- the child of DF2. The modification was that all columns save for ‘CityID’ and ‘TimeZone’ were removed. It was then aggregated to eliminate redundancy.
* Cityjoin – is the ‘city\_id’ csv. This is the central spoke to how the other datasets will be connected. No modifications where made from the transform stage.

When all dataframes were read in we then coded the process to generate a .db file to store our information. On completing the read-in we joined the tables together based on “CityID” using DB Browser. Also we made some indexes manually to possibly help with data efficiency.