# Database schema

* Database is not used in this case study, but if this is a real-case scenario where the data will not be used only once, there should be more consideration regarding data size and consistency. Database needs to be involved to tackle these issues.
* The data consists of CSV files that have different organization, also the data are separated by time period(season, year), and if all combined will be a rather huge dataset to work around with spreadsheet tools, so I will have to put them into a SQL database.
* For the convenience of data processing and reading, I shall organize the data in a more consistent & helpful way, also removing some columns that has little use for the data analysis.
* There will be 2 data tables for the data. One is about trips and one is about stations :
  + Trip
    - trip\_id (varchar)
    - start\_station\_id (varchar)
    - end\_station\_id (varchar)
    - trip\_date (date)
    - started\_at (time)
    - ended\_at (time)
    - rideable\_type (varchar)
    - user\_type(varchar)
    - trip\_duration (decimal)
    - day (SMALLINT)
  + Station
    - station\_id (varchar)
    - station\_name (varchar)
    - lat (decimal)
    - longi (decimal)

# Data Merging

* For if I need to merge all the data into one format in the database.
* The necessity of data merging in this case study context : convenience in data retrieval and manipulation, station-wise analysis, to store data in a more consistent manner for future use.
* Merge data of 2 different organizations into 2 SQL tables
* First of all, we need to address the inconsistencies between the 2 organizations of data :
  + Usertype : data from 2019 divides users into ‘Subscriber’ and ‘Customer’, whilst data from 2020 to 2023 divides users into ‘member’ and ‘casual’, so we need to map ‘Subcriber’ to ‘member’ and ‘Customer’ to ‘casual’.
  + There are some extra characters(asterisks) in the string of the station names, need to do the data cleaning for those name before moving into id-name mapping.
  + There are different station id in 2019 data compared to that of 2020-2023. To address this inconsistency, we have to first build a mapping between station id and station name for the data of 2020-2023, then traverse all the station name appeared in 2019 data, then change their station id to the corresponding ones in the mapping. If it occurs such that the station name is not found in the mapping, then add one new instance(with non-colliding station id) to the mapping.
  + For the data columns that were collected in 2019 but not in 2020-2023 and those vice versa, I will set the data cell as NULL.
  + Since I am merging the data into a SQL database, and some of the columns CAN’T be null due to the E-R constraints, I will remove the data entries that have empty values on these columns and put them separately in a CSV file(for possible future usage/reference).
  + The CSV file ‘Divvy\_Trips\_2019\_Q2.csv’ had different column names but same organization as data from the same year, so manually change the data column names.

# Data Cleaning

* All data cleaning is done using R language.
* I took one year of data to do analysis, namely from July 2022 to June 2023. But as the volume of the data is huge, I will do the cleaning and analysis as per season. I aggregated monthly data into four seasonal data, namely July-Sep 2022, Oct-Dec 2022, Jan-Mar 2023, Apr-Jun 2023.
* The data cleaning process for each season is done by same piece of R script, the process is as follows:
  + Read monthly data from 3 month’s CSV file of the corresponding season.
  + Remove duplicated data rows
  + Check for missing values and remove missing values, for the missing values only take up a really small portion of the data(<0.1%).
  + Clean the station names, some of the station have different names in the data, some contain extra asterisks, some contain extra remark such as “Public Rack”
    - Remove all asterisks
    - Extract & replace station name with format “Public Rack - XXX” with its substring.
    - Remove all brackets
    - Trim the data names(remove whitespaces)
  + Compute new column ride\_duration\_sec (difference between ride start time and ride end time)
  + Convert duration from second to hour, ride\_duration\_hour
  + Check for outliers, then remove rows with ride duration that is longer than 24 hours, since they take up a really small portion of the data(<0.1%).
  + Compute distance between starting and ending station, dist\_km using Haversine Formula.
  + Remove rows with dist\_km outliers, since there should not be any stations within Chicago that are over hundreds of KMs apart.
  + Separate the year, month, day, time into different columns by string slicing.
  + Remove rows with invalid values, such as negative distance, negative ride duration.