Cleaning

We used R to clean the datasets. We will first describe the cleaning process used in R for the datasets bike shops and bicycle parking spots. We used the tidyverse, and readr libraries in R to clean the data, and saved all ensuing datasets to CSV. Note we used the write_csv function from the readr library, as it does not write the row names to the csv file. For the bike shop dataset, we first selected the relevant attributes and rename them to their corresponding names in our schema. This allowed us to ensure that we are aligning the correct attributes in our schema with the correct attributes in the CSV file. We converted the number to integers instead of character, and the names to lower case. This ensures that if there were some naming choices by the different capitalization choices by the different datasets, we would still be able to join on these attributes. Some of the street numbers had non-numeric characters, and we removed these observations to ensure that the data conforms to our design. For the remaining datasets that relate to bike parking, which was three different datasets, we treated them similarly regarding renaming and type casting. For these datasets we treated bike type of parking as a factor and forced "capacity" to be an integer. For the datasets that were not the high-capacity parking dataset, we did not have capacity information in those datasets and hence we treated each observation as a default capacity of one. Again, like the bike shop dataset we removed all observations that had a non-numeric character in their street number. Afterwards we joined the three parking datasets, by using row bind, and removed any duplicates. For some of the datasets the parking spot type had null values, so when saving we encoded NA as "", to ensure that psql interprets it as null when using \copy.

The traffic volume dataset contained number of different types of vehicles that passed a traffic light at a certain intersection. We first summed over the different type of vehicles to get total number of vehicles that passed through each intersection. Then we divided this number for each intersection to get by the number of times a measurement was taken at that intersection. This gave the average traffic volume at every intersection. From this we got an estimate for the average traffic volume per street. We split the intersection into two columns, one for each street

name. Then for each street that appeared in the columns we summed up the averages to get an approximate average of traffic volume per street. To count the traffic lights per street, we counted how many different times one street name occurred in different intersection. Since each intersection is a traffic light

For the BikeShare data, there were several datasets that recorded the data for each quarter over the years. Since there many of the datasets did not have consistent naming conventions (for example “blvd” is sometimes written as “boulevard” in another dataset) and different column names. We only picked the dataset containing the info on the first quarter of 2021. Each row in this dataset represented a trip, with the important columns being with station ids, date and station name present. First, we grouped the dataset by “station_id” we counted how many times it appeared as the “end station” for trips and created a column called checkoutTotal to record it. This gave us the total number a times a someone checked out a bike from the station. We did a similar process to get the total number of times a bike was checked for each station. We also split the location into 2 columns street1 and street2 since the location name was usually an intersection. However, we also noticed that some of the bike stations were also located at ttc subway stations. Hence, we created another column called “at_station” to indicate if this bike station was at a ttc subway station or not. Whenever street1 or street2 contained the word “station” we set this column value to TRUE.

For the TTC subway dataset. It contained the total usage per station. We took the bikeStation csv file and searched which TTC station matched the name of location of the bikestation that was located at a station. We recorded this value in the column as hasBikeStation. We also got the location of all bike repair stands that were located at a subway station and by matching the names, we found which stations has a bike repair stand we stored this information in hasBikeStand column

Lastly to create the streets table csv, we first got the unique names from the streets in the intersection dataset. We also got the average traffic volume and traffic light data from this dataset. Finally, we matched the name of the subway stations to street name to record how many subway stations each street has. We did similar process for the other variables to.

Finally, we removed rows from other datasets that recorded streets that was not in the streets dataset to uphold the integrity constraints.

Design and Changes

We will discuss in this section the comments from the T.A. from part1 and changes we made from the original design. The comment from the T.A from part 1 was about ensuring that we can use Street name and Street number as the primary key for our bike stations and parking spots. After investigation, we had noticed that these would not be possible primary keys, and therefore we decided to use an id from the respective original datasets as our primary keys.

Further, we decided to remove one of our relations, repair stand, as it was not particularly meaningful for investigation and instead, we included information related to repair stand in our subway station relation. Lastly, we removed bike lane information from the streets relation as we where unable to properly deal with that attribute. Further, for our design we decided to use a user defined type for the type of parking spot in parking spot type. Because the parking spot type can only take on a few values, we thought this would make the design better. Further, we used a default value of 'Unknown' instead of null to avoid the potential problems that can arise from null. Further, we changed the type for our street number to make it an integer instead of a text. This will allow us to do important calculations on these values. Further, we designed our schema in such a way that we do not have to use null values. For example, for the parking type attribute, we used a default value of 'unknown' instead of null which avoids complications from the null values. Furthermore, when examining our schema, we were unable to find any functional dependencies implying that our schema does not have redundancies. Lastly, there are some attributes in the streets relation that queries, and joins can calculate with other relations; however, such operations can be computationally expensive, and hence should be avoided.