Capstone Mini-Project: Data Wrangling
The code used for this process is broken down step by step at the link below:
https://github.com/dgeihsler15/Springboard/blob/master/Capstone1/Data_Wrangling.ipynb

Dataset: 2017 Divvy bike sharing data (https://www.divvybikes.com/system-data):
The 2017 Divvy bike sharing data is available on their website in two main parts; trip data, and bike station data. The trip data contains a unique trip id for each trip taken and contains information on the timing of trips, stations used, and what type of user took the ride. The two user types refer to a “customer” who purchased a 24 hour pass, and a “subscriber” who purchased an annual membership. For Subscribers there is gender and birth year information as well. In the station data the main piece of information that we’re interested in is the station capacity (how many docking stations are available at each station). There is also information on the location of all the bike stations and the date they went online as well.

Loading the Data: The trip data for 2017 came in four csv files, one for each quarter. These were loaded into dataframes individually so that the shape and the column names could be looked at closer. Although each of the four dataframes had the same shape and column information, we’ll find that the Q4 information used a different format for the start and end times. Before combining the four quarters together, I first had to make sure the times in Q4 were uniform with the format in the other quarters. This will make it easier to convert these columns into datetime objects later in the process.

After using the .concat() method to combine the trip data for 2017, the next step was to load in the station data and merge it with the trip information. The station information came in two csv files, one for Q1 & Q2 of 2017 and one for Q3 & Q4. The Q3 & Q4 file was selected so the most up to date station information would be merged with the trip data.

Two merges were performed in total. One matching the station id in the station file to the starting station id in the trip file, and the second matching the station id in the station file to the ending station id in the trip file. This will allow us to view both the information for the starting and ending stations on the same line for each trip.

Cleaning the data:
After the merge I found some duplicate columns as there was some identical information in both the trip and station data. The first step was to drop the unneeded columns. Afterwards the remaining columns were renamed so they were clearly labeled and had a consistent style. This step was followed by filling in any missing values. Inspecting the the dataframe closer, the only columns that contained missing information were the gender and birth_year columns. As only “subscribers” gender and birth year information were tracked, this missing information was filled in with a “Non Subscriber” designation in the gender column to make the lack of gender clear. I wanted to hold off on filling in birth_year until it was converted to a numeric data type.

Moving on to the data types in each column, it made sense to change the way a couple of the variables were stored. The gender, and user_type, were both changed from strings to categories in order to save memory. Birth_year was transformed to the float data type using pd.to_numeric(), but after converting birth_year to number, and taking a closer look at the data, I noticed an issue. There was a fair number of riders whose birth years listed would make them over 100 years old. As people of this age are not likely to be riding a bike around Chicago it

does not seem that all of this information is accurate. This may be caused from users entering incorrect birth years while signing up, but because of the inaccuracies and that it is only available for subscribers, it will not be very helpful in grouping all riders by age. Because this project's main focus is on station usage and proper stocking requirements, the birth_year column was dropped from the dataset as well. The last adjustment to column data types was changing the start and end time columns to datetime objects. This will allow for more flexibility in grouping this information by different time frames later in the project.

As each row in the dataframe represents an individual trip, it was clear there were a few duplicate rows. These duplicates were dropped and the index of the dataframe was then set to trip_id.

Checking for outliers:

Categories: The first step I wanted to take while checking for outliers is to make sure the category columns all contained information that made sense. Although the gender column did not need any adjustments, I found something to investigate in the user_type column. Customer and Subscriber were the only expected user types, but using groupby on usertype made it clear that there were a very small number of rides marked as "Dependent" for user_type. As this was not mentioned in the README files associated with the data, I took a closer look at all the rides marked with a Dependent user_type. I noticed that some had gender info and some did not. Because the gender information was only tracked for "subscriber," the dependents with that information were changed to "subscriber," and those that didn't were changed to "customer" to avoid any confusion.

Trip Duration: The README file for the trip information states that any trips shorter than one minute or longer than 24 hours have been excluded. Based on those parameters it isn't likely that we'll find any outliers in the trip_duration column, but this was checked using the .describe() method to make sure the trip durations correctly fell into the parameters expected. No outliers were found.

Station Capacity: Looking closer at the station capacity using .describe() I found that the min capacity was 0. As it's not possible to have a station with a 0 capacity, I looked into what stations were listed that way. It turns out two of the stations on the Q3 & Q4 stations file were listed with 0 capacity. This potentially could have been because those stations were closed down in the second half of 2017, but because we have trip data for all of 2017, it's important to find the correct capacity of those stations when the rides were completed. Our best option was to load in the Q1 & Q2 station file for 2017, and use that to find the correct capacities for those stations. I was then able to find the capacities for those stations, and update that information in the combined dataframe.

Rides per day:Another potential outlier could be the number of rides in a single day. If any count of daily rides was dramatically different from the rest, it may warrant further investigation. The dataframe is currently indexed by trip_id, so my first step was to create a new dataframe with a timeseries index based off the start time of all the trips. Then I could groupy index.date and get a count of all the data points each day. Looking at these counts and then using the .describe() method I could see that the min value being on Christmas and the max values being found in the middle of summer made sense. The data is now in good shape for further exploration.