As the data was gathered and stored every other day in a csv using a webscraper, the first approach to cleaning the data was bringing it all together in one single dataframe container. The code contains the script that was used for web scraping.

One quick detail to mention is that “address” was used as the source of geographic location. A new column “address\_check” was created as a boolean to indicate presence or absence of an address. Although it seems counterintuitive, False indicates an address and True indicates no address.

Duplicate values were dropped from the dataframe using “post id” column. This column contained a unique integer and so seemed an ideal column for identifying duplicate ads and dropping them from the dataset. The “post id” column did contain null values, so to account for this the “title” column was leveraged, as it was the next most unique column. However, a duplicate value in the “title” column does not necessarily indicate a duplicate post, be it due to different time, price or other attributes across posts with the same title. In order to drop the least amount of duplicates by title, two dataframes were created: one absent of null “post id” values and one with only null “post id” values. This way, only rows with null “post id” values had duplicates dropped according to title. This dataframe was then joined back with the larger dataframe.

Cleaning the data required the creation and elimination of numerous data columns. It should first be mentioned that craigslist ads do not follow a standardized format, as a lot of freedom is left to the user about what information they would like to include in their ad. For the following section, refer to the **Sample Webscraped Data csv file**.

. The “square feet” column contains, in the majority of cases, either a value of the square feet for the rental, or an availability date. The “beds\_baths” column more often than not contains the number of beds and baths of the rental. The “square feet 2” column more often than not contains the number of bedrooms and the square feet of the rental. The end goal of cleaning the data was to retain as much information on bedrooms and square feet as possible. Due to the fact that there were sometimes missing values from the three aforementioned columns, where in another one of the three columns this value was not missing, the script was written to capture a beds or square feet value where it was not present where expected, but present somewhere else in the data. This called for the creation of several boolean columns that indicated if either the bedrooms or square feet variable was available in one of the three aforementioned columns by using a regular expressions matching pattern. In addition the regular expression function findall was used to capture and store the value where it was present, so as to later fill the column containing final bedroom values or square feet values. This was done in a specific order, so as to make sure that the values from all three columns were investigated for the values.

In addition to ensuring that as much available data as possible as obtained, the columns were cleaned so as to appear as integers, without prefixes such as “$” or end-fixes such as “ft”. These include the “beds\_baths”, “square feet”, and “price” columns.

Due to the Superbowl occurring in Minneapolis at the same time that the data was being collected, there was potential for noise in the data with price values not representative of the normal rental market. These values were eliminated by creating the “check\_superbowl” column which contained a boolean using regular expressions to find superbowl-related ads using the “title” column.

As geospatial data was a focus of the analysis, many steps in the data cleaning process were taken so as to capture as precise and accurate geospatial data as possible. Firstly,, the “latitude”, “longitude” and “map accuracy” columns were removed. This information was initially included in the web scraper, but does not provide accurate data on the location of rentals, as the latitude and longitude coordinates come from an OpenStreetMap link in the ad itself, which frequently contains a randomly-dropped pin. Instead, values were geocoded using the Google Geocode API. Values “False” in the “address\_check” column were fed into the geocoding script (remember that False indicated an address present).

Geospatial information was available at the county level for each ad, as the“url” column contains a three-letter abbreviation identifying the county in which the rental is located. Assuming that the value in the three letter value in the url reflected the true county in which the ad was located, a “counties” column was created by extracting this value from the “url” column. After creating data columns with county information, a column called “county\_state” was created, which contains a string “(county), Minnesota”. It was attempted to insert this at the end of each address during geocoding so as to achieve a higher accuracy, but something went wrong in the code, and only some of the addresses were read by the geocoder with this extra bit of information. The lengthy process of geocoding due to a 2500 daily requests limit by the Google API made it impractical to go back and fix this. However, there were a number of addresses that were not geocoded as a location not within the state of Minnesota. After the geocoding process, these values were identified, and dropped from the data set.

While it is not fully apparent in the data cleaning code, not all csv files containing web-scraped craigslist listings were globbed all at once. This is due to various factors. The Google Geocode API only allowed 2,500 requests per day and so the cleaned csv was run daily through the geocoding script, incrementing the iterrows indexing line of code by 2,500 each time. Additionally, data collection and other data cleaning processes occurred alongside geocoding processes, so not all data was available to be geocoded during each iteration. There were three iterations of globbing csvs containing web-scraped data, cleaning, and geocoding.

The geocoded data underwent further cleaning. All address values that produced an error during the geocoding process or simply did not have an address were dropped from the dataset and excluded from the analysis. In addition, It was determined that bed values Studio-4 units would be included in the analysis. These 5 values correspond with the fair market rent, had the least proportions of identified outliers (read on for steps on identifying outliers) and had the greatest amounts of listings. The dataset was subsetted as such.

The data was also subsetted by county. It was ultimately decided after the geocoding process that only values from Hennepin and Ramsey counties with address locations would be included in the analysis.

Summary statistics were obtained for five groupings of the data. Groupings were Studio, 1, 2, 3 and 4 values in the “beds\_baths” column indicating the number of units an advertised listing contained. Related to units and which ads were classified in which category, some important examinations of the data were needed, specifically as it pertained to “beds\_baths” values that were “0” or null as indicated by a “False” value in the “beds boolean column”. In finding an appropriate way to assign these values to either “Studio” or “1”, first the “titles” column was searched for any information that would indicate these values as a Studio, 1, 2, 3 or 4 bedroom. Half of these values were matched with one of the aforementioned groupings. The other rows that remained unmatched were dropped from the data set.

All operations and analysis was performed regarding each possible value in the “beds\_baths” column as a separate distribution and dataset.

Accordingly, outliers in the “price” and “square feet” columns of the dataset were identified and managed respective of these five groupings.. Values in these two columns 1.5x the IQR below the 25th percentile, or 3x the IQR above the 75th percentile were considered outliers. The differing values were used as 3x the IQR below the 25th percentile produced a negative value, but it was desirable to eliminate values in the “price” column that were 1, which usually indicated an incomplete or spam ad, or a number like 25 dollars, that was likely advertising a nightly, not monthly cost.

After removing outliers from the dataset, null values in the “price” and “square feet” columns were filled with the mean values of their respective “bed\_bath” grouping.

At the end of the cleaning process, values in the “square feet”, “price” and “beds\_baths” column that still did not conform to the desirable pattern were set as the mean value of the respective column in which they were found. To get the means of the price and It was first necessary to create a separate dataframe without null values to calculate the mean value for each column, and then fill null values in the main dataframe with the calculated mean for each column