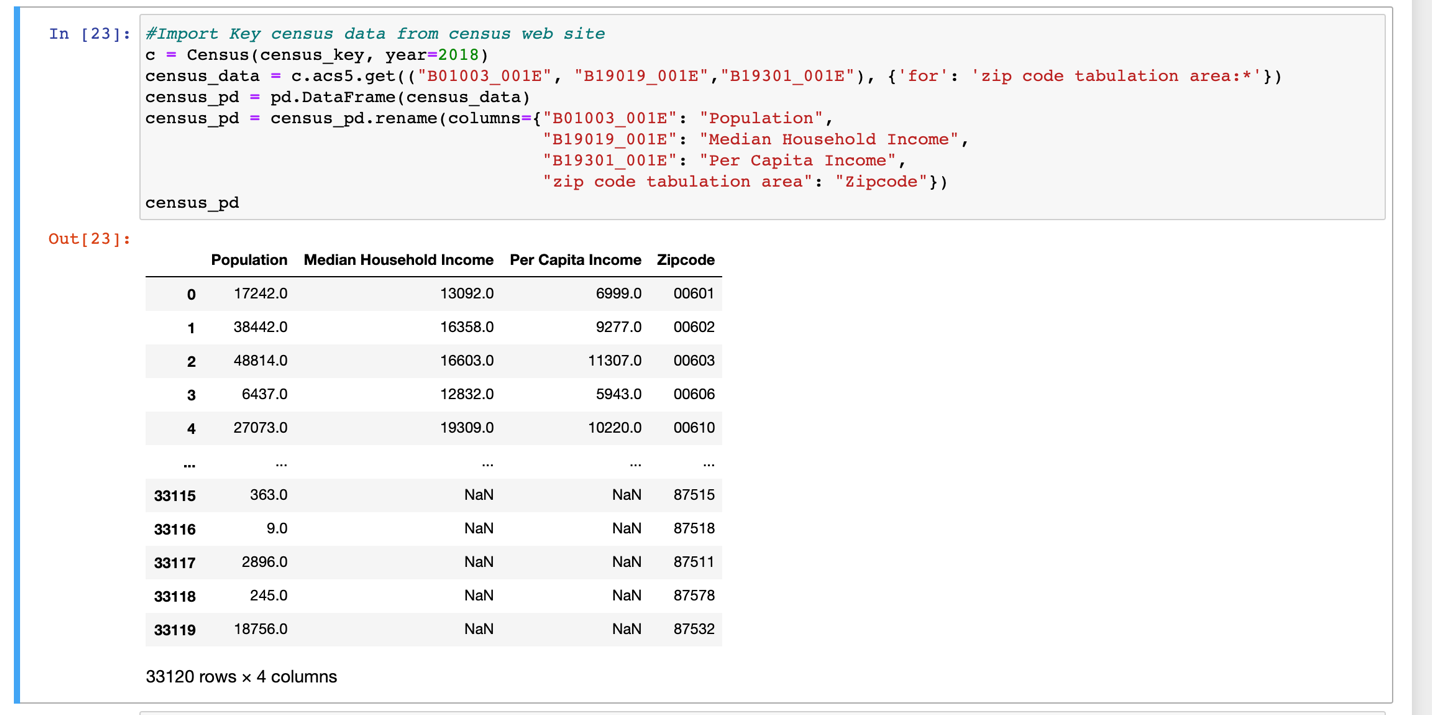
**Census Demographics Data Sources**

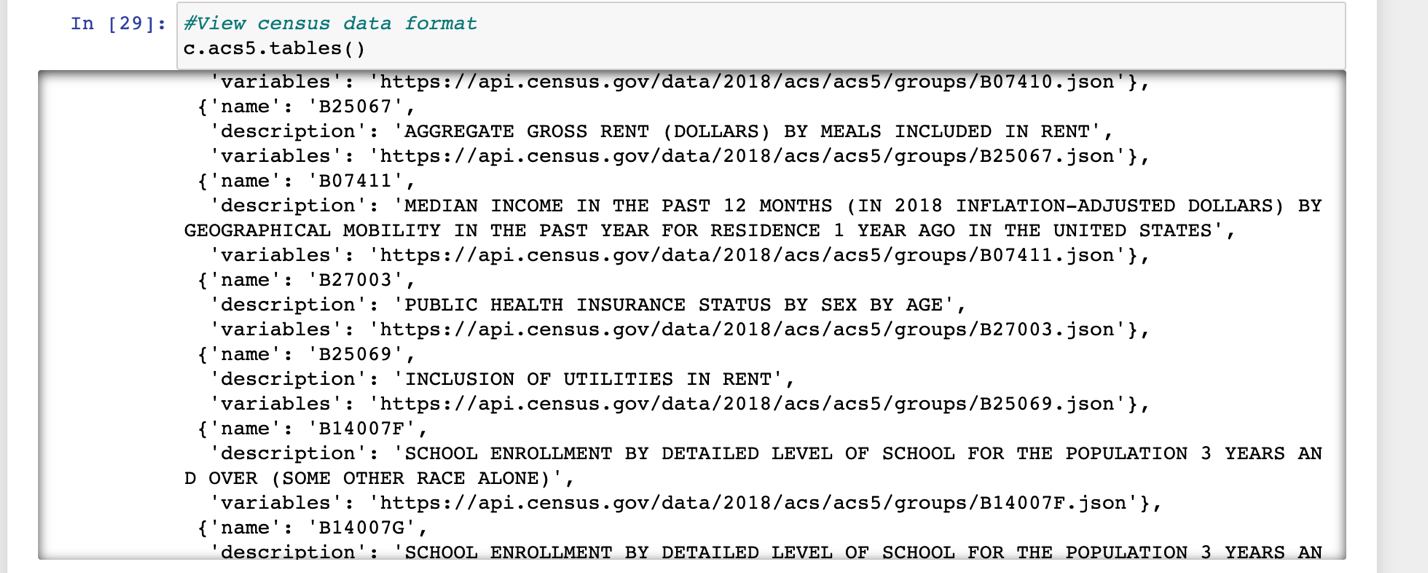
First, we load all the libraries we need.



We then pull Census data including Population, Median Household Income, Per Capita Income by Zip Code and set up a dataframe.



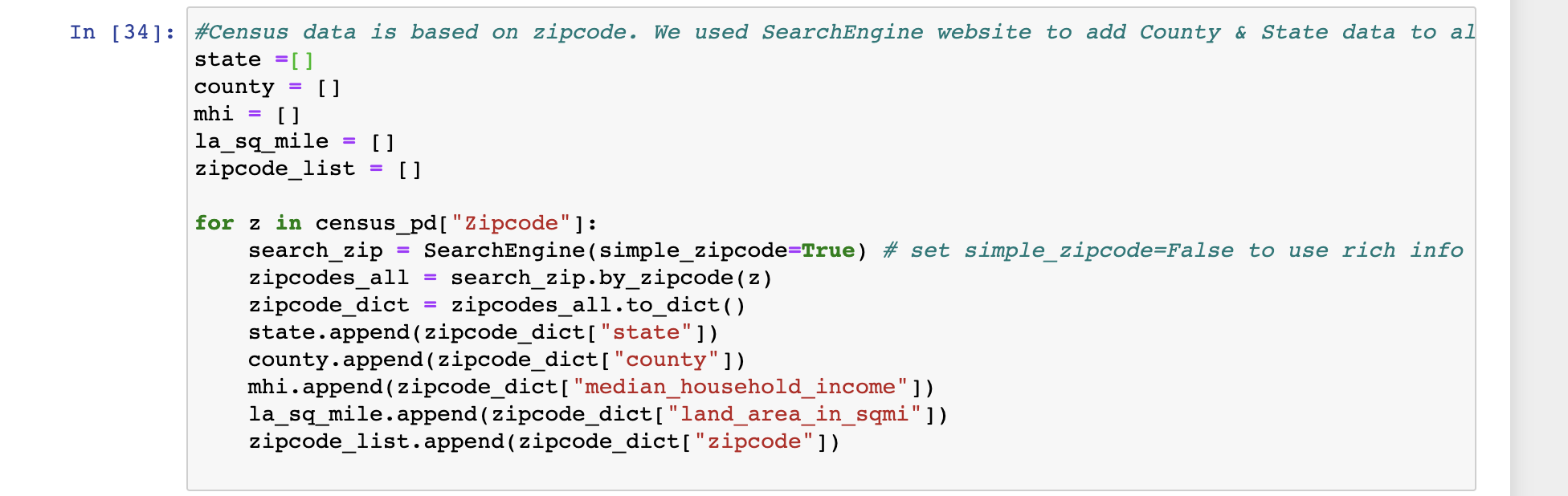
A quick view of the json to see what fields we need to pull



Export the file as a CSV to Excel



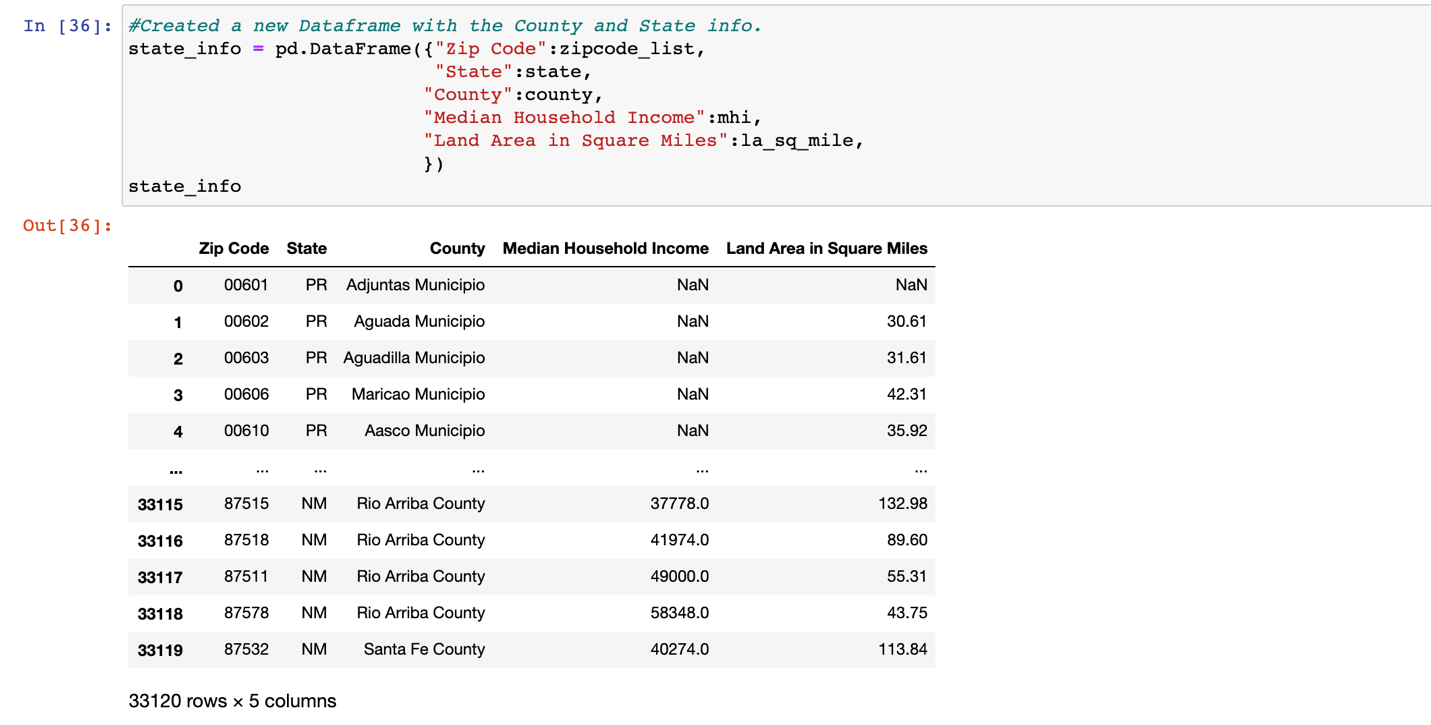
Since the Census data is grouped by Zip code, and we want the data in County and State, we used the Pypi api’s SearchEngine to run all 33 thousand plus zip codes through a loop to associate each zip code with its corresponding County and State.



I quick look at the json to confirm the fields we needed.



Here’s the new dataframe which now includes County and State.



And finally, export the csv.



**Further Cleanup and Merging the 2 Census Data Sources**

We started with dependencies and setup

A screenshot of a cell phone

Description automatically generated

We then loaded the file with land area

A screenshot of a cell phone

Description automatically generated

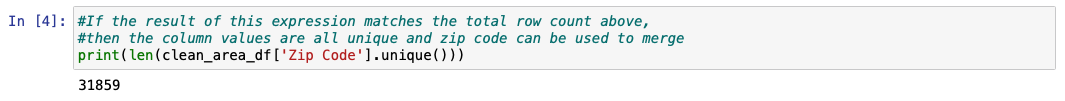
We then did some clean-up:

* Remove any rows missing land area (cannot figure population density without that)
* Remove any rows missing Median Household Income
* Oklahoma has 1 zip code without a county name, but since we’re not doing a by-county analysis, it doesn’t matter
* Remove any rows with a negative Median Household Income (some commercial-only zip codes have this)
* Rename Land Area in Square Miles to Land Area (m2) for brevity
* The results show the same number of rows in all columns:

A screenshot of a social media post

Description automatically generated

To ensure this data frame is ready for joining by zip code, we ensure that the number of unique zip codes is the same as the total number of zip codes above



Next, we loaded the census information described above

A screenshot of a cell phone

Description automatically generated

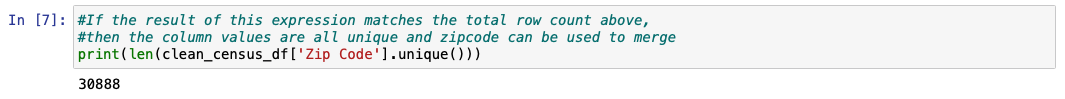
We did some cleanup on this data source as well:

* Eliminate any rows missing Median Household Income
* Eliminate any rows missing Per Capita Income
* Create Total Income by multiplying Per Capita Income by Population; this will be needed later to aggregate to the state level
* Renamed Zip Code and Population Density colums for consistency
* The results show the same number of rows in all columns:

A screenshot of a social media post

Description automatically generated

Again, we checked to make sure the data frame is ready for merging by zip code by verifying the unique list of zip codes is the same length as the total list of zip codes



Merge the 2 data sources above on zip code.

* Inner join; any missing data makes the whole row irrelevant
* Drop any rows in the resulting data frame that are missing a state; we won’t be able to aggregate those into the state level
* Drop any rows missing Population
* The results show the same number of rows in all columns:

A screenshot of a social media post

Description automatically generated

Group the data frame by state and aggregate our measures by median, sum, or custom function as needed. Also rename land area column to correctly reflect the unit

A screenshot of a social media post

Description automatically generated

Finally, save the results to a CSV for further processing



**Cases and Deaths Data Source**

Import necessary libraries

A screenshot of a cell phone

Description automatically generated

Prepare parameters and a result data frame for the results

A screenshot of a social media post

Description automatically generated



There are 2 alternate versions of this data frame:

* An original “bi-monthly” data frame that has the 1st and 15th of every month since cases and deaths have been counted (March 2020). This data source omits the 5 “Diff” columns seen above, since the figures are presented as daily differentials and do not make sense in a bi-monthly context.
* An enhanced “daily” data frame that has all dates since March 1, 2020, to date. This frame includes the daily differentials since they make sense with daily data.

Analyze a sample of the results returned by the URL query. For this, we just said USA and a recent date

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

The data source returns a list, data, in which each element represents a “state” (more on this later). Each data element has a region dictionary with information about the state, includes its counties, packed in the list labeled as cities above. The details of that list have been omitted because we are not using county-level data because our census data by zip code only aggregates cleanly to the state level (zip codes often span multiple counties).

Execute the query for each date, depending on the version being run, and pack each returned relevant data element into the data frame from above.

A screenshot of a social media post

Description automatically generated

A screenshot of a cell phone

Description automatically generated

Here, we kept the following fields, as relevant measures to compare against wealth, population density, and age:

* Active
* Confirmed
* Date
* Deaths
* Fatality Rate (for convenience; figured as Deaths / Confirmed)
* Recovered (for convenience; figured as Confirmed – Active – Deaths)
* State
* State’s abbreviation (from the py US library)
* Active Diff\*
* Confirmed Diff\*
* Deaths Diff\*
* Recovered Diff\*
* Lat/Lon for each state, as provided by this data source

\*Differential fields were only kept in the “daily” version of this data source, since they represent daily changes.

We also observed there to be a lot things reported in the data as “states” that were not actually states. Some examples are a lot of the cruise ships at sea that were reporting cases. Since those are not real states that we do not have real census data to compare with, we used the py US library to only extract data pertaining to real states, as seen in the “if” block above.

To retrieve all the data correctly, we needed to loop once for each day needed. This was the reason we initially created a bi-monthly data source to be faster to work with, and then later created a daily data source that was more detailed, but much bigger and taking longer to create.

Within each date, we had to loop through all the data elements returned, determine if each represented a real state, and if so, extract its information needed for the data frame as described above. A pair of nested try/except blocks attempt to catch KeyErrors so that the script can keep processing if a particular date or data element does not contain the dictionary keys expected.

We determined this would be a more optimal way to retrieve all the data than blindly loading everything returned into the data frame and then attempting the drop the junk data out of it (i.e., the rows that would not represent real states)

Afterwards, we checked to ensure all columns have the same amount of data to ensure no row is missing any important measures

A screenshot of a social media post

Description automatically generated

Check to make sure we only have 50 unique states in the data frame

A screenshot of a cell phone

Description automatically generated

Finally, save the results to a csv for convenient access merging to census information and plotting, since especially the daily version takes a long time to load



**Fully Combined Data Source**

Import necessary libraries

A screenshot of a cell phone

Description automatically generated

Read the cases-by-state data source saved in the CSV file described above

A screenshot of a cell phone

Description automatically generated

Rename State and State Abbr columns to be more consistent with the census data source, especially since we’ll be joining on State abbreviation

A screenshot of a cell phone

Description automatically generated

Load the data source with census demographic information from its CSV described above

A screenshot of a cell phone

Description automatically generated

Ensure the census demographics data source is unique by state (51 includes DC)

A screenshot of a social media post

Description automatically generated

Merge cases to census demographics on state using an inner join since we cannot use rows with missing information on either side. Validate many-to-one because the cases data frame has a list of all 50 states for each date within it.

A screenshot of a cell phone

Description automatically generated

Finally, save the resulting data set in a CSV file for use with plotting



**Age Groups**

For Age groups, we first download the libraries.

A screenshot of a cell phone

Description automatically generated

Load a csv from the CDC website containing info on number of deaths by age bracket for nationwide and state level.

A screenshot of a cell phone

Description automatically generated

Delete Columns that were no needed

A screenshot of a cell phone

Description automatically generated

Locate just the rows pertaining to national data.

A screenshot of a cell phone

Description automatically generated

Clean up Column headings

A screenshot of a cell phone

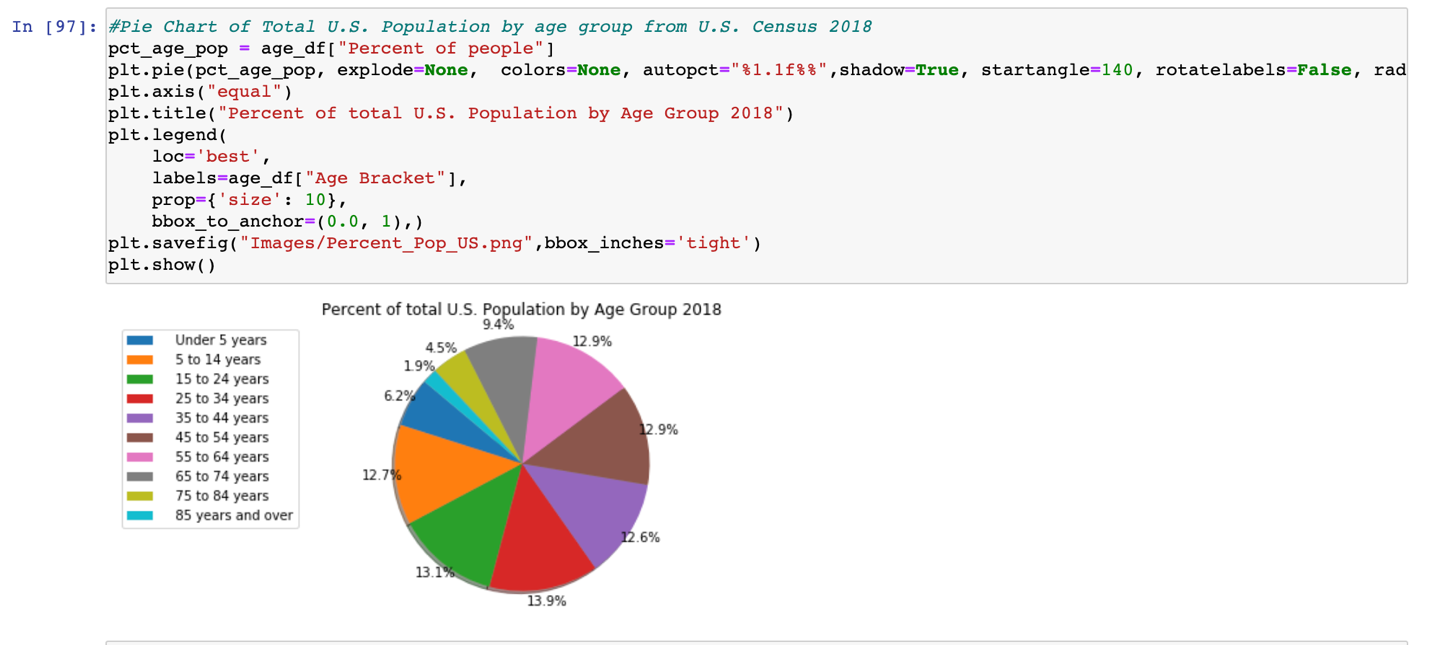
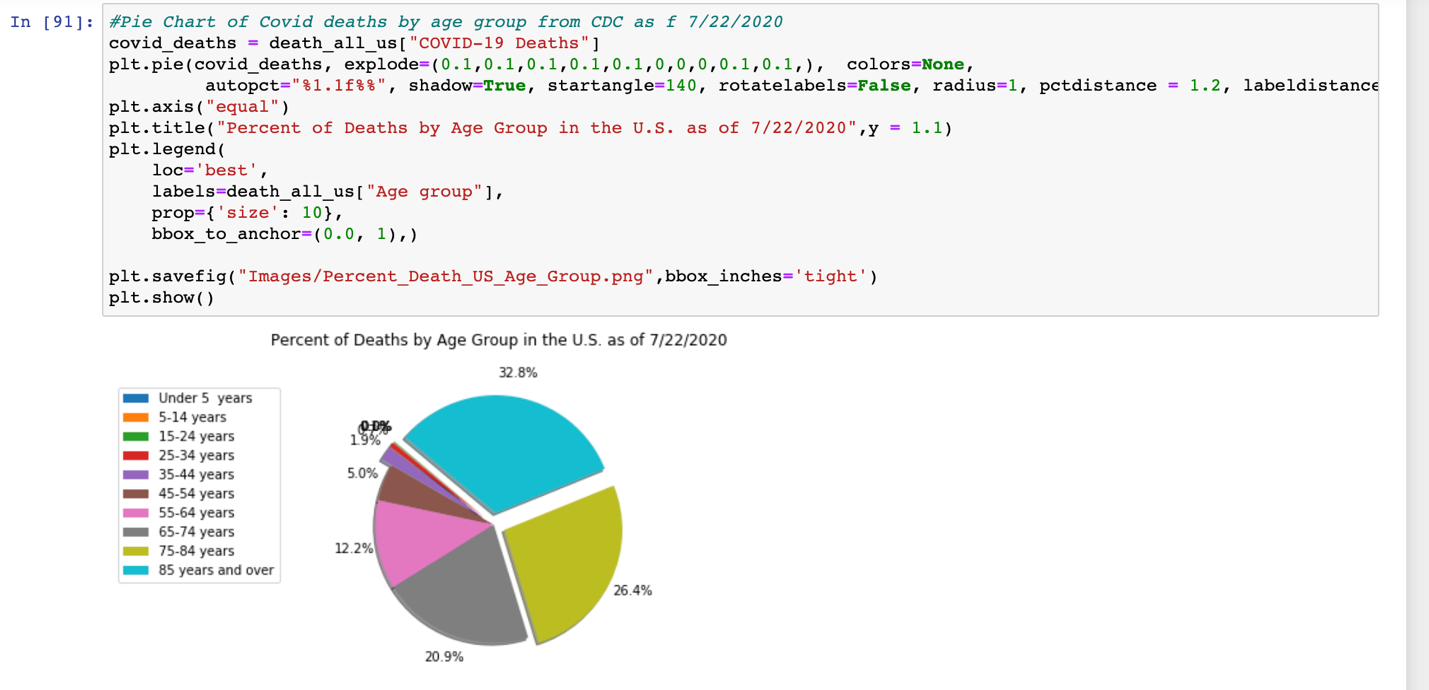
Description automatically generated

Load Census data downloaded from the Census website.

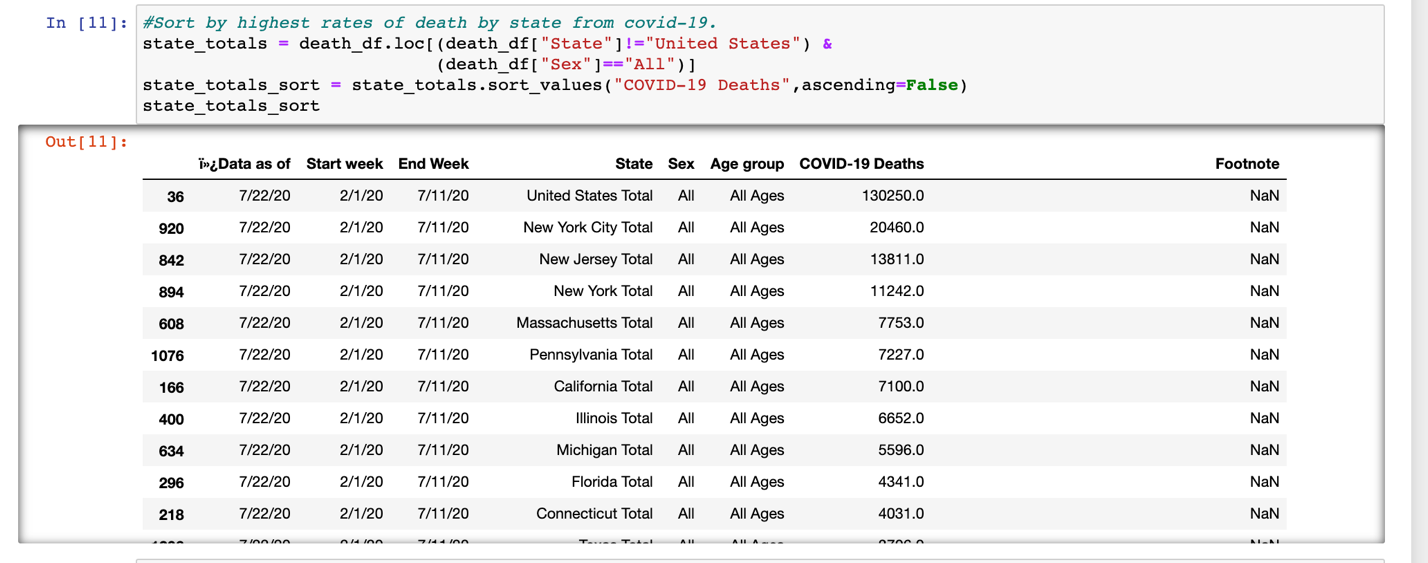
A screenshot of a social media post

Description automatically generated

Create two pie charts showing Populations by age bracket and Covid-19 Deaths by age bracket.



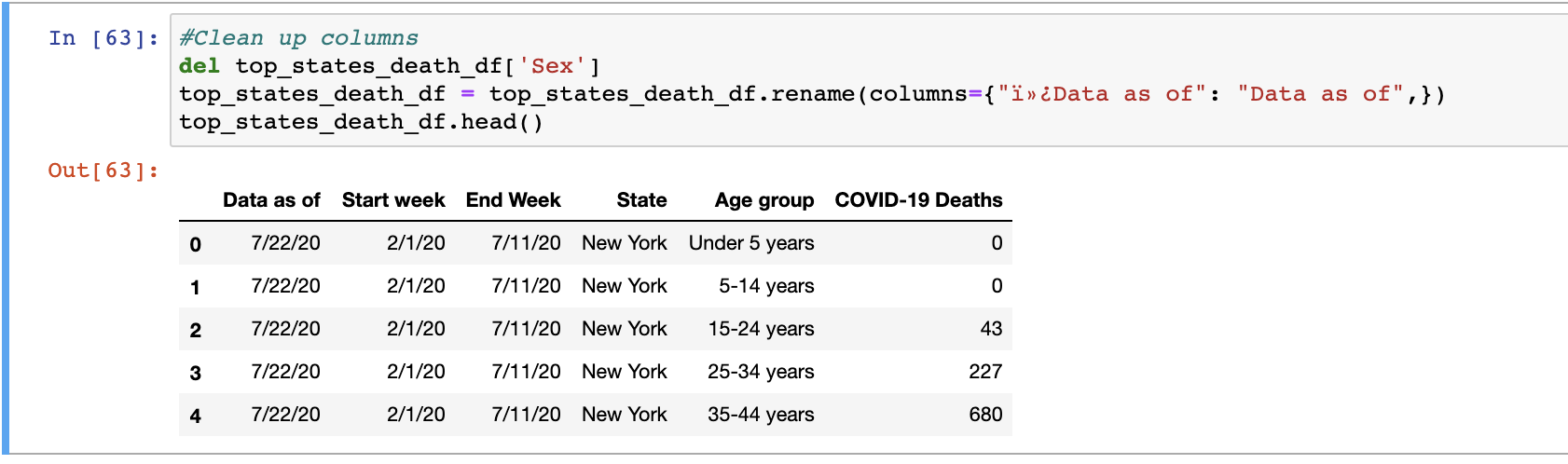
We searched for the states with the five highest rates of death from Covid-19



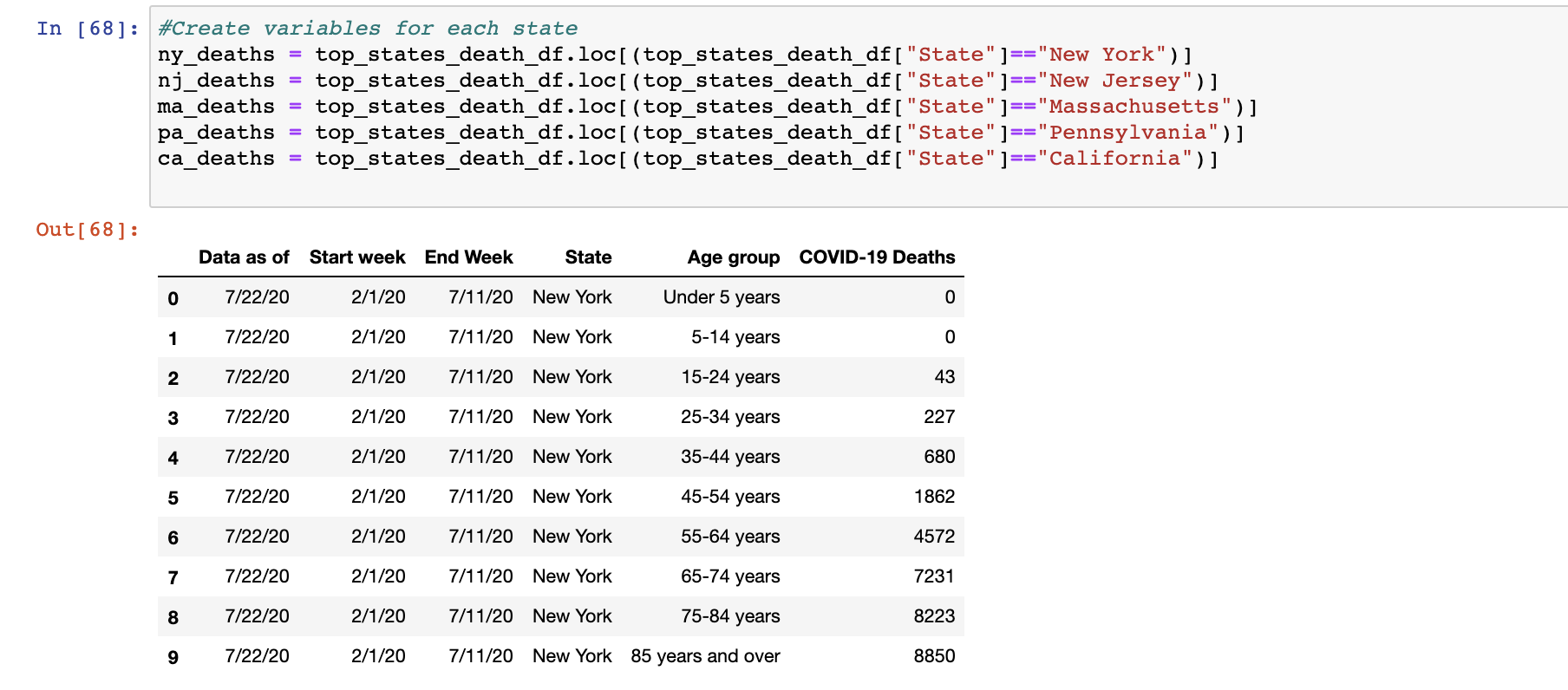
To save time we manually updated the csv file to combine data and reimported it.



Cleaned up the column headings.



Created variables for each of the top five states.



And created pie charts for each of the five states with New York given as an example.

