Goals -

One goal we had that was a little hard to achieve was creating a map with each of city we had data on from both websites. It became very crowded and hard to read, therefore we ended up just displaying a map with the top three cities. We also began the project thinking we would use Altair for our visuals, however stuck with matplotlib, given the resources on in online and the experience we got from using and seeing it in class. Another original goal we had was to see if the population density affected the average hourly salary. We achieved this and found that population density has a larger correlation than simple just the amount of people in the city. Another goal we completed was being able to see what size range generated the highest average salary. Interesting the middle, not the largest size range, had the highest average income.

Problems -

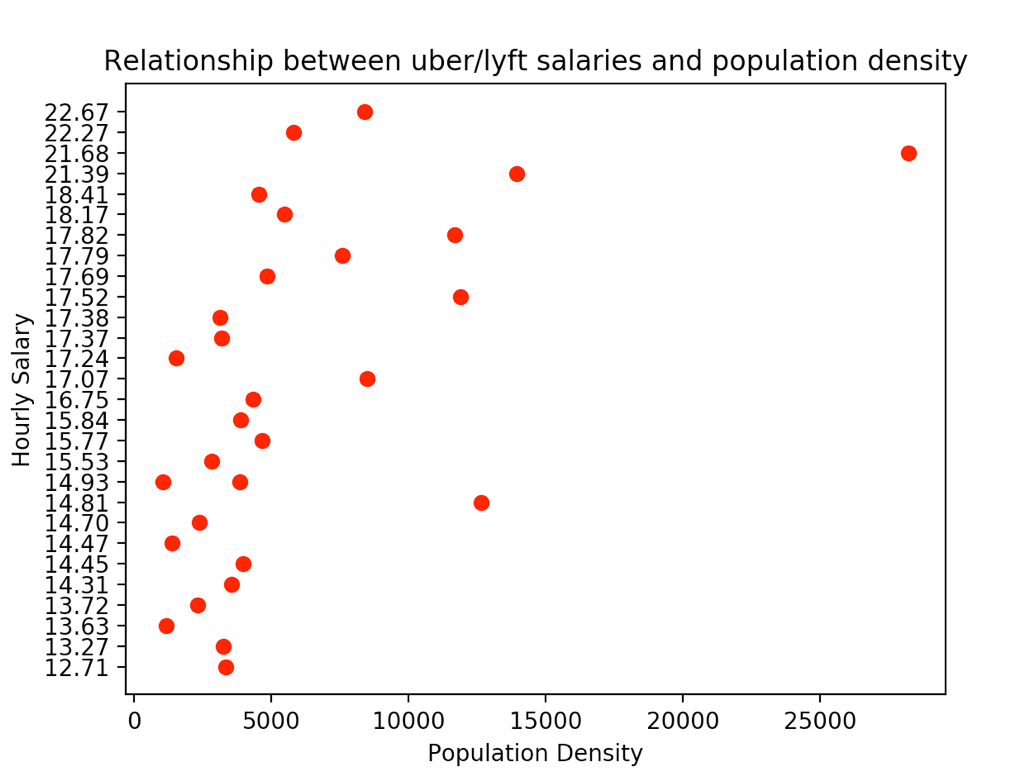
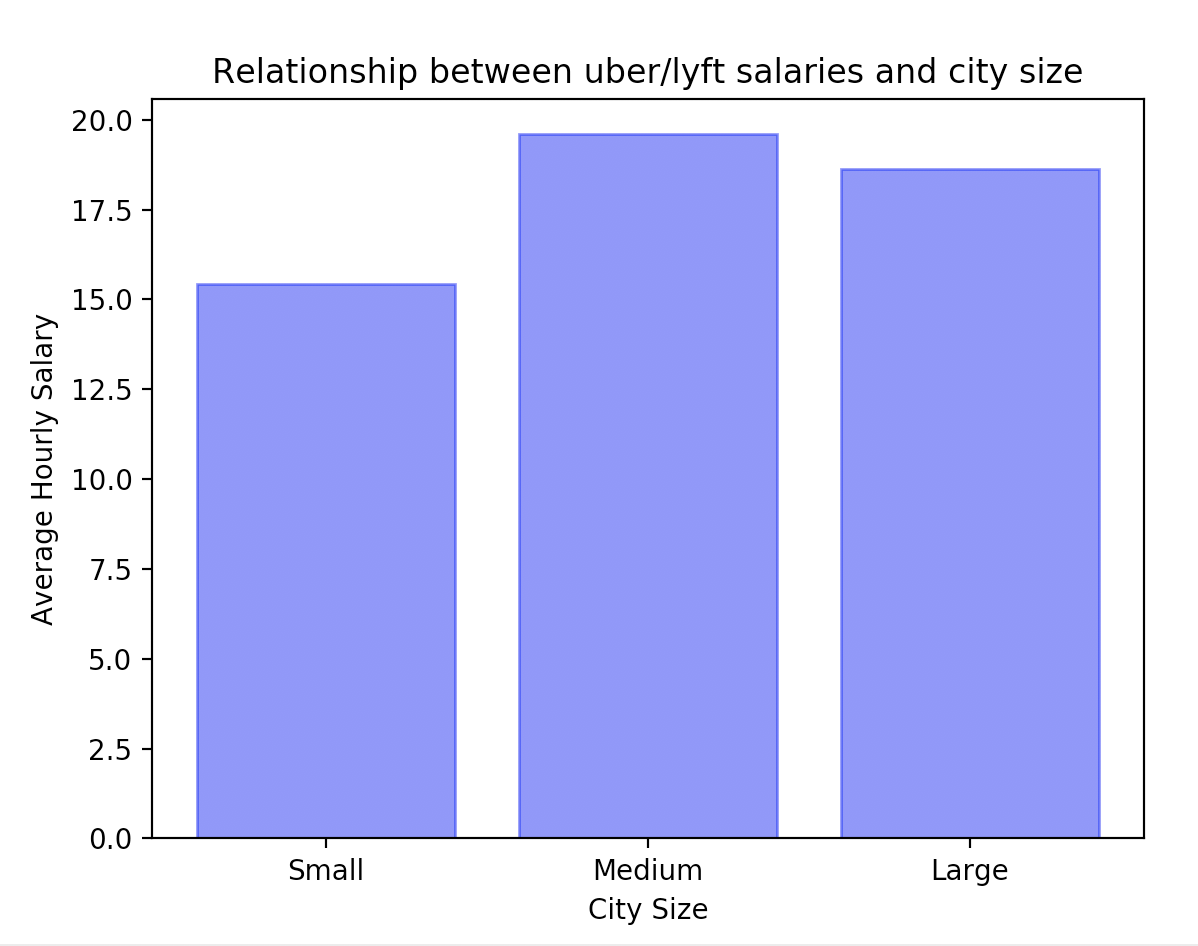
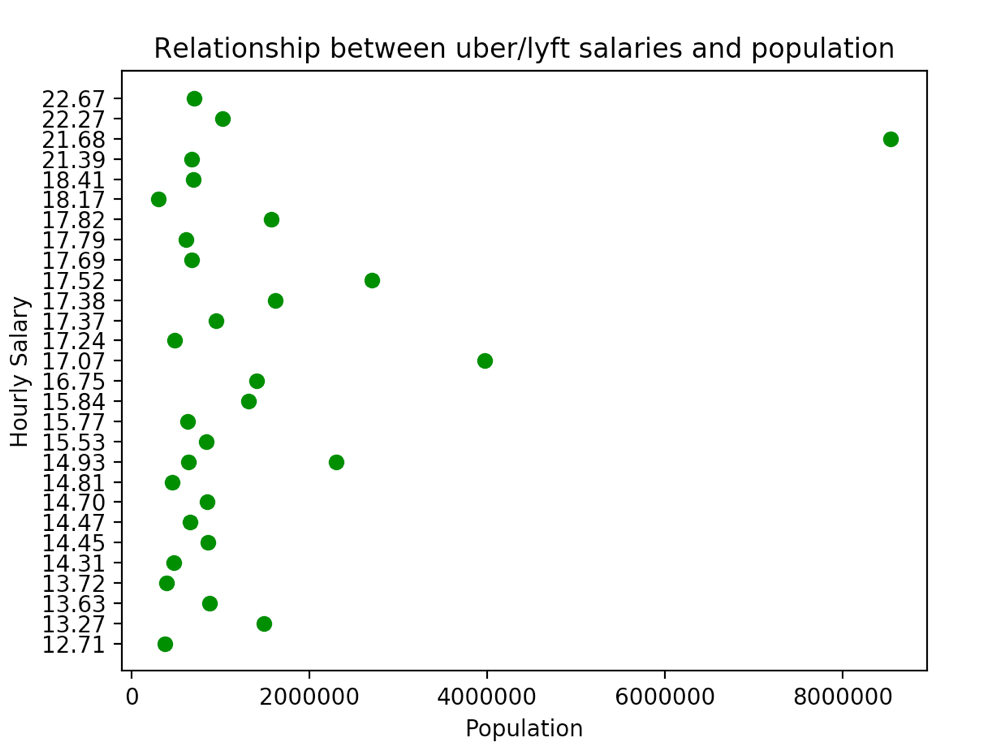
Some problems we faced while creating this code was the how to display the data in order to create meaning from it. Eventually we thought it was interesting to see how not only population density, but also populations as a whole effect the average income earned by uber and lyft drivers. Another thing we struggled with for a little was not using the drop table requirement, but in order to work past this we changed our create table line to have a requirement of only if it does not exist.

Running the code -

In order to run this code, first you must run the gettingdata.py file, this grabs the information and creates a database with three different tables. Then the user must run the file, finalproject.py, this first makes calculations and then with those creates and shows the four visuals. After that is done it as writes to a csv file as shown below with the information, we were able to get and make conclusions out of.

Data represented in the calculations file

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| City and State | Population Density | Ranking | Hourly Salary | Population |
| Seattle, Washington | 8,391 | 2 | $22.67 | 704,352 |
| San Jose, California | 5,808 | 3 | $22.27 | 1,025,350 |
| New York, New York | 28,211 | 4 | $21.68 | 8,537,673 |
| Boston, Massachusetts | 13,943 | 5 | $21.39 | 673,184 |
| Denver, Colorado | 4,530 | 7 | $18.41 | 693,060 |
| Pittsburgh, Pennsylvania | 5,484 | 9 | $18.17 | 303,625 |
| Philadelphia, Pennsylvania | 11,692 | 10 | $17.82 | 1,567,872 |
| Baltimore, Maryland | 7,594 | 11 | $17.79 | 614,664 |
| Detroit, Michigan | 4,849 | 12 | $17.69 | 672,795 |
| Chicago, Illinois | 11,883 | 13 | $17.52 | 2,704,958 |
| Phoenix, Arizona | 3,126 | 14 | $17.38 | 1,615,017 |
| Austin, Texas | 3,182 | 15 | $17.37 | 947,890 |
| Kansas City, Missouri | 1,529 | 16 | $17.24 | 481,420 |
| Los Angeles, California | 8,484 | 17 | $17.07 | 3,976,322 |
| San Diego, California | 4,326 | 18 | $16.75 | 1,406,630 |
| Dallas, Texas | 3,870 | 19 | $15.84 | 1,317,929 |
| Las Vegas, Nevada | 4,660 | 20 | $15.77 | 632,912 |
| Charlotte, North Carolina | 2,829 | 21 | $15.53 | 842,051 |
| Houston, Texas | 3,842 | 22 | $14.93 | 2,303,482 |
| Oklahoma City, Oklahoma | 1,053 | 23 | $14.93 | 638,367 |
| Miami, Florida | 12,645 | 24 | $14.81 | 453,579 |
| Indianapolis, Indiana | 2,366 | 25 | $14.70 | 855,164 |
| Nashville, Tennessee | 1,390 | 26 | $14.47 | 660,388 |
| Columbus, Ohio | 3,960 | 27 | $14.45 | 860,090 |
| Atlanta, Georgia | 3,549 | 28 | $14.31 | 472,522 |
| New Orleans, Louisiana | 2,311 | 29 | $13.72 | 391,495 |
| Jacksonville, Florida | 1,179 | 30 | $13.63 | 880,619 |
| San Antonio, Texas | 3,238 | 31 | $13.27 | 1,492,510 |
| Tampa, Florida | 3,326 | 32 | $12.71 | 377,165 |

Code Documentation -

We have separated our code into two different Python files. The first being a series of functions to gather data from both of our websites. For our function we input the city populations website URL in order to be able to sort through the data and return a list of tuples containing information for each city. This sets us up to then use a function to make our first table. For this table function we call the previous function on the same URL, so our input is the article URL in addition to the cur and conn to be able to use SQL methods within our table. This function does not return anything but commits our changes to the table in our database. Our next function follows the same pattern because it is creating an additional table for the website. This table is using a bit of different information, but the same process. We then have second data gathering function which takes as input our second website. This outputs the same format of information. A list of tuples per city containing information about the after Uber salaries in each. Next, we use a function to call on this data gathering function and input the data into our third table. We use the second URL along with cur and conn to achieve this. Similarly, we do not return any output, but we commit that changes to the database. Our main function runs all of these in the order explained in order to process our data.

Our next python file deals with our visualization. Our main function first runs the function that sets up our database. A database name is needed for input and our data is returned by the function. We then use this data to create our four figures. These functions do not take any input but do call on the set\_up function in order to use our data. Lastly, we export our calculations into a csv file. This function does not take in any output, but opens a csv files, writes our data to it, and proceeds by closing this file. This is our last function called in our main function.

Documentation of Resources -

Throughout the project we reviewed lecture slides and discussion posts to help us with small questions. Some of the larger and more time-consuming problems we came across required some of the resources listed below.

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Issue Description | Location of Resource | Result |
| April 20th | Using a map in a figure | Online matplotlib.org | Resolved our issue |
| April 22nd | Writing to a csv file review | CSV files lecture slides | Resolved our issue |
| April 21st | Selecting/checking for data already in a database | SQLcourse2.com | Resolved our issue |