**Goals:**

We intend to investigate the relationship between population distribution by race, income, and restaurants in Ann Arbor and Ypsilanti. We want to see if there is a correlation between the race distribution and percentage of different ethnic restaurants, and we also would like to know if there is a correlation between income level of each city/race and the average pricing of its restaurants.

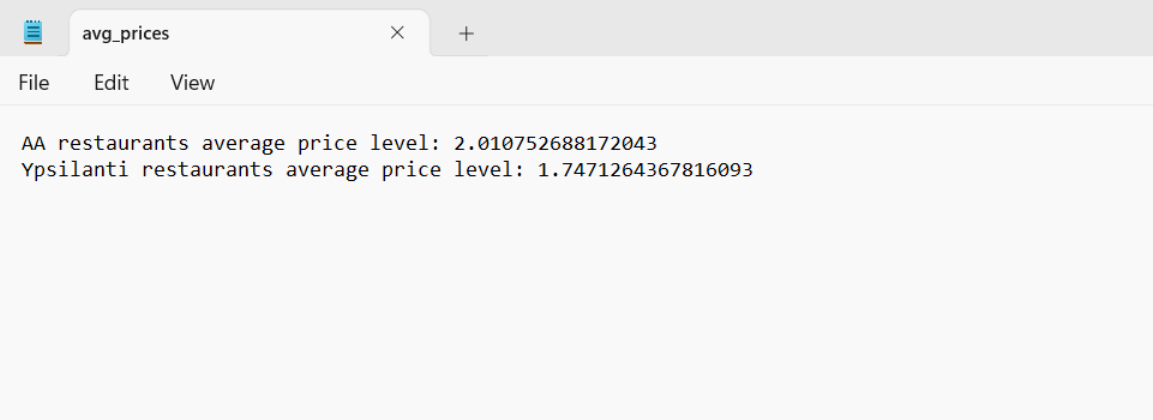
Our achieved goal is that there is a correlation between income and restaurant pricing. Since we found it hard to associate each restaurant with a specific race or ethnicity and we cannot assume the race of the customers, we only plotted the categories of the restaurants rather than calculating the proportion of each race within the restaurants (see “Problems we faced”).

**Problems we faced:**

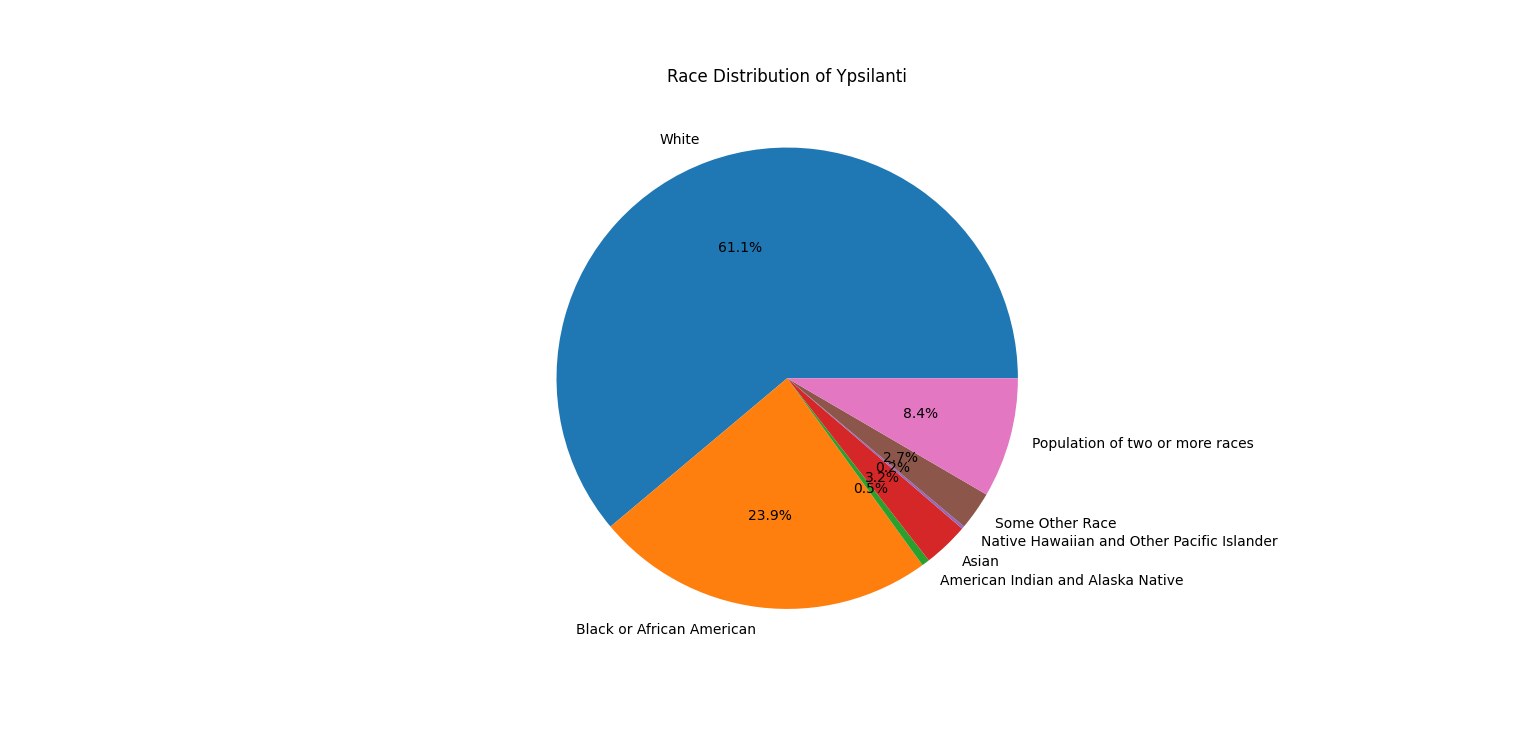
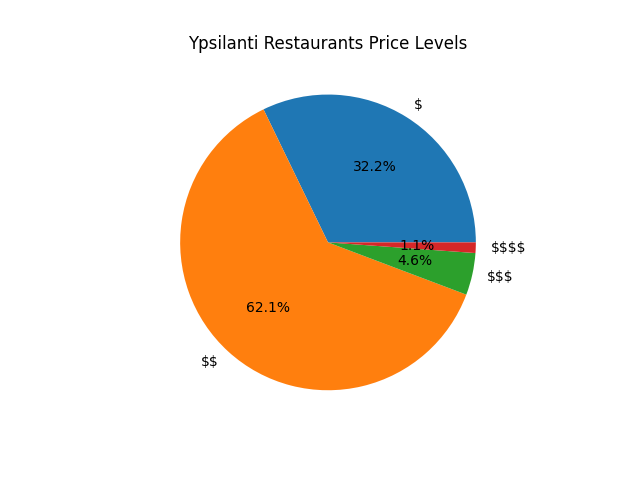
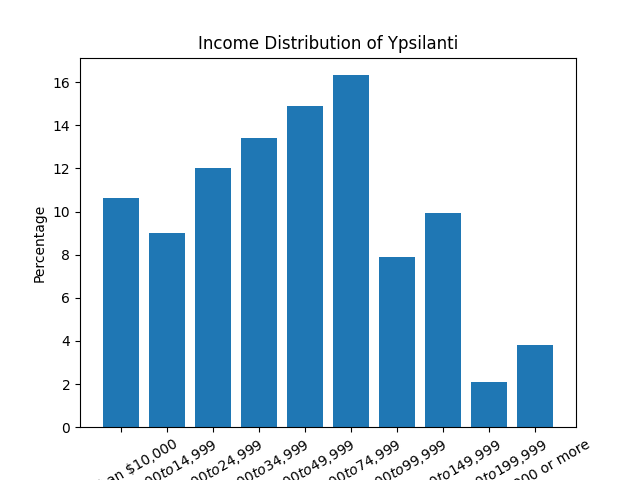
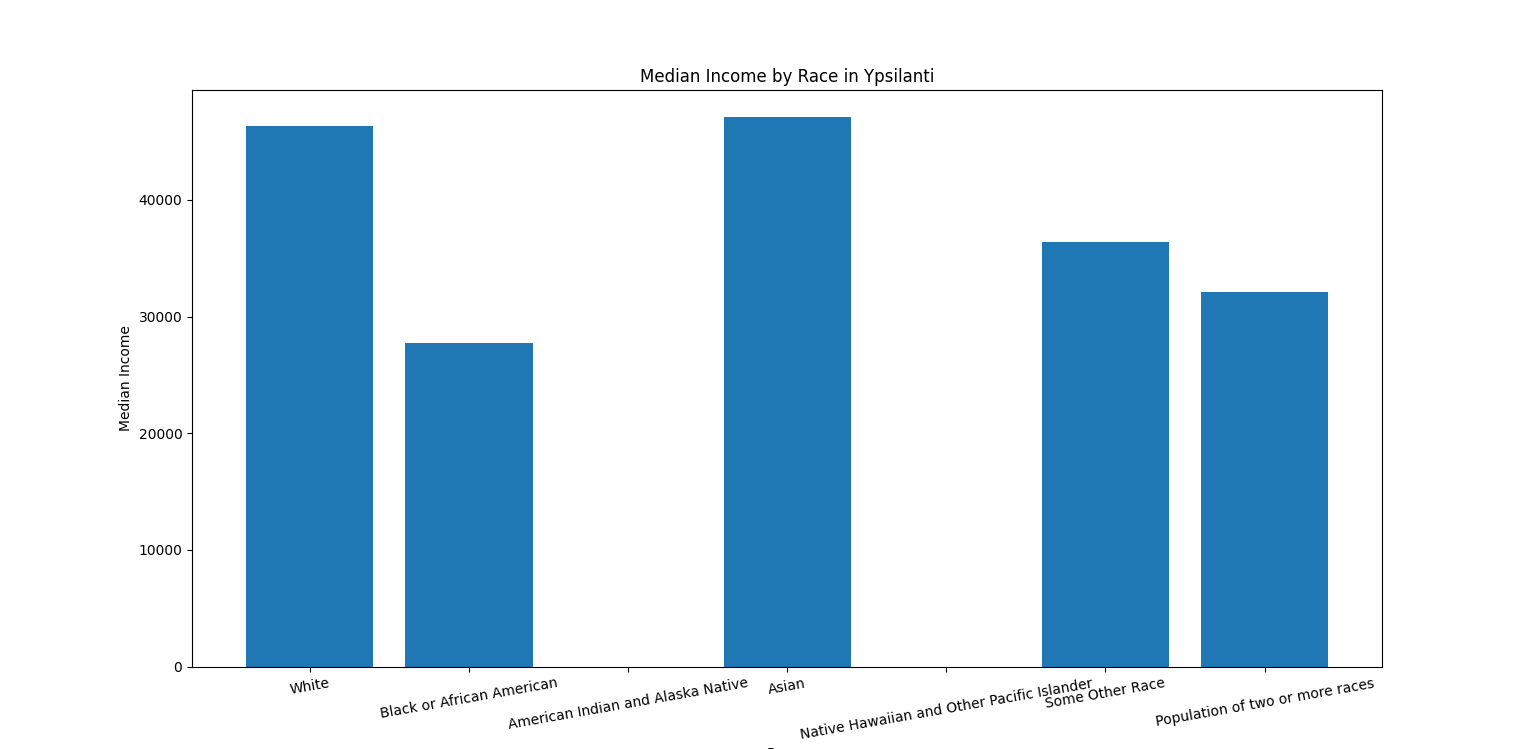
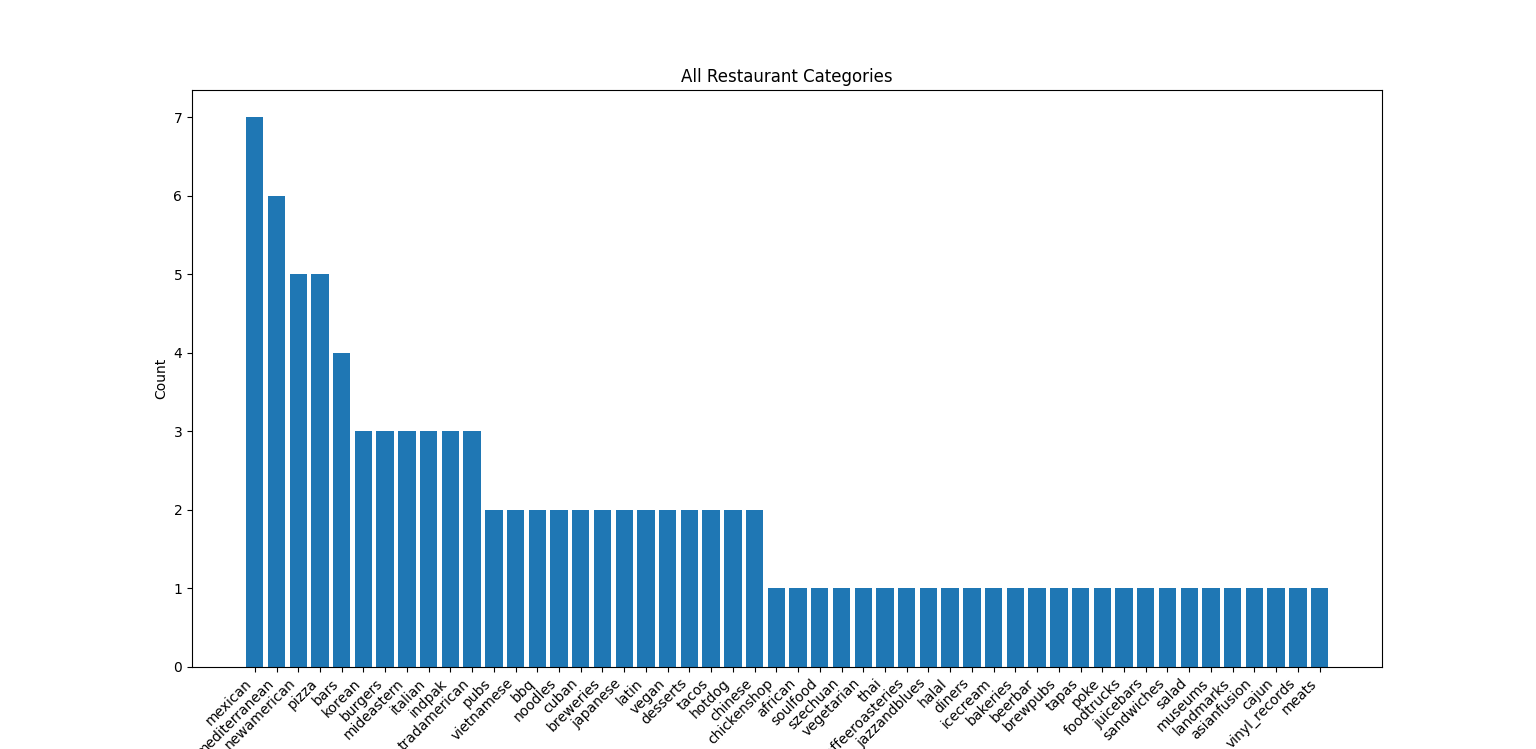
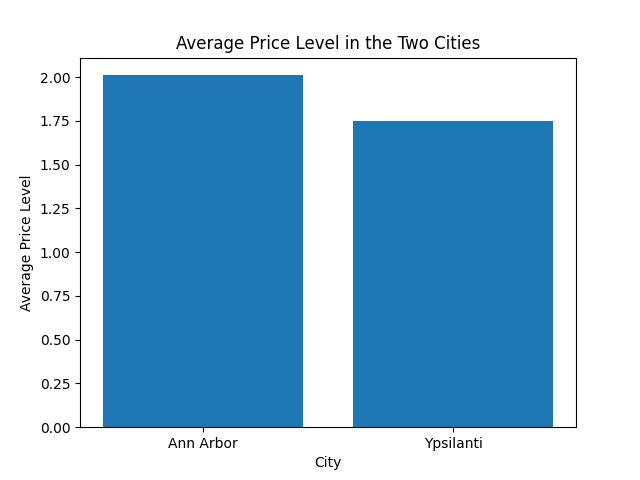
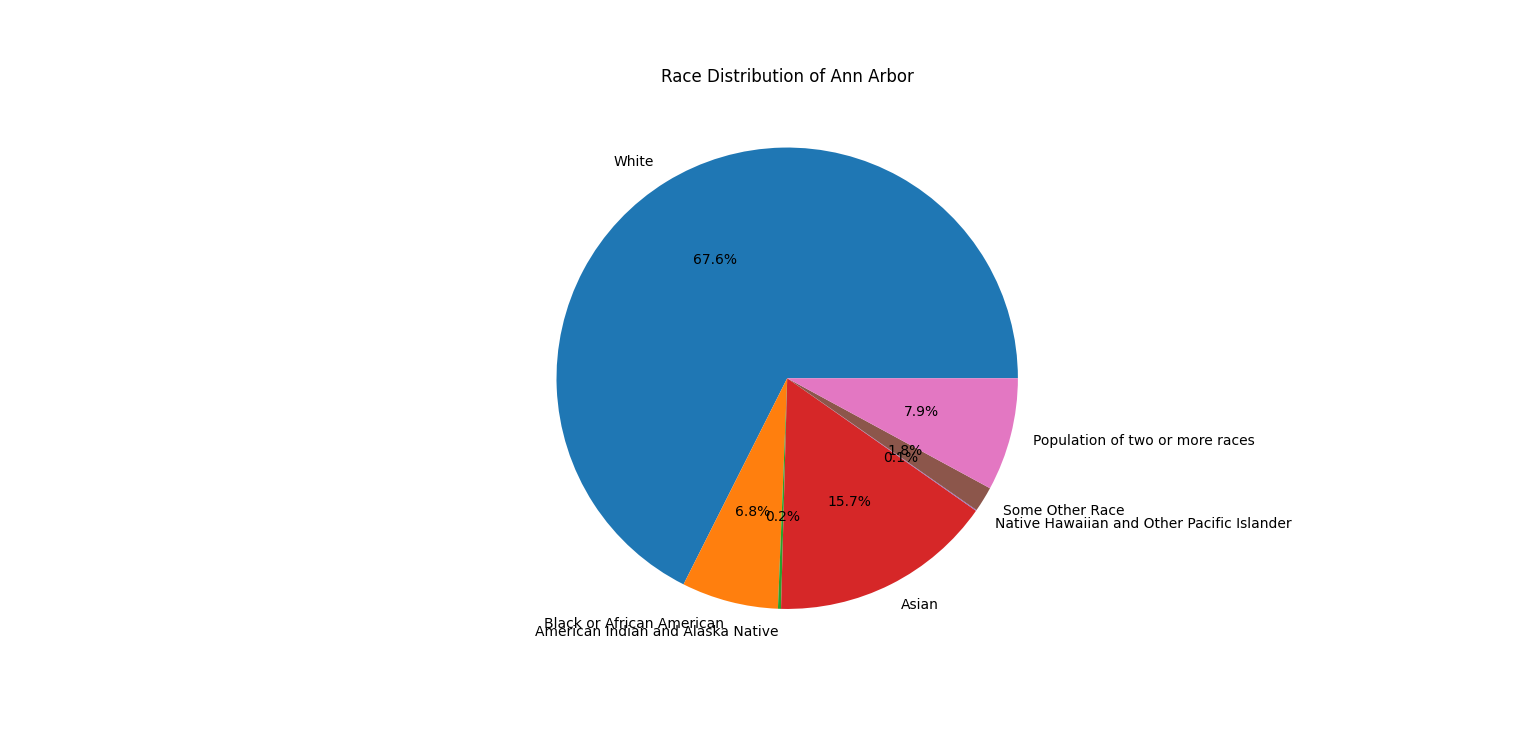
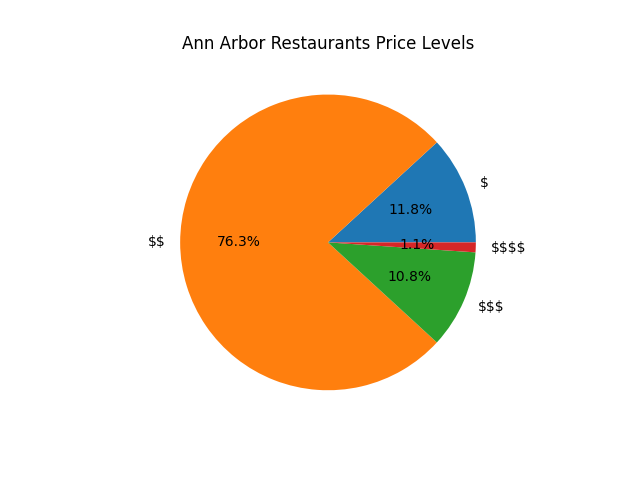
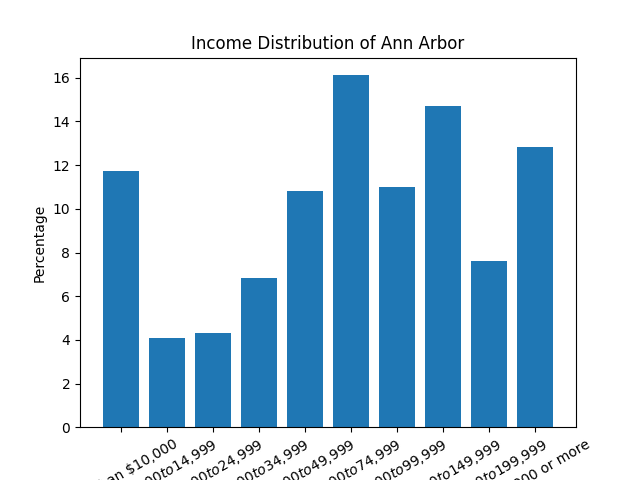
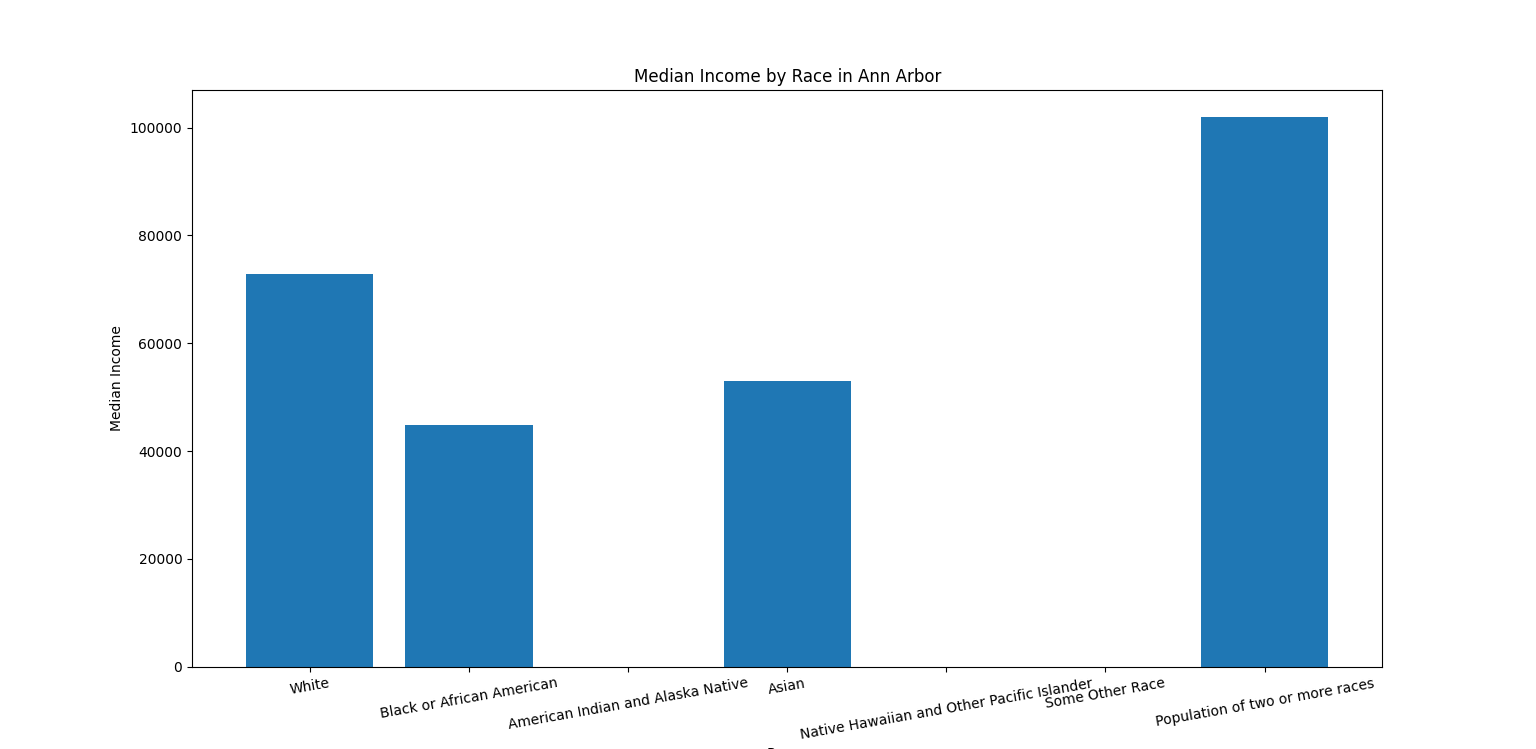
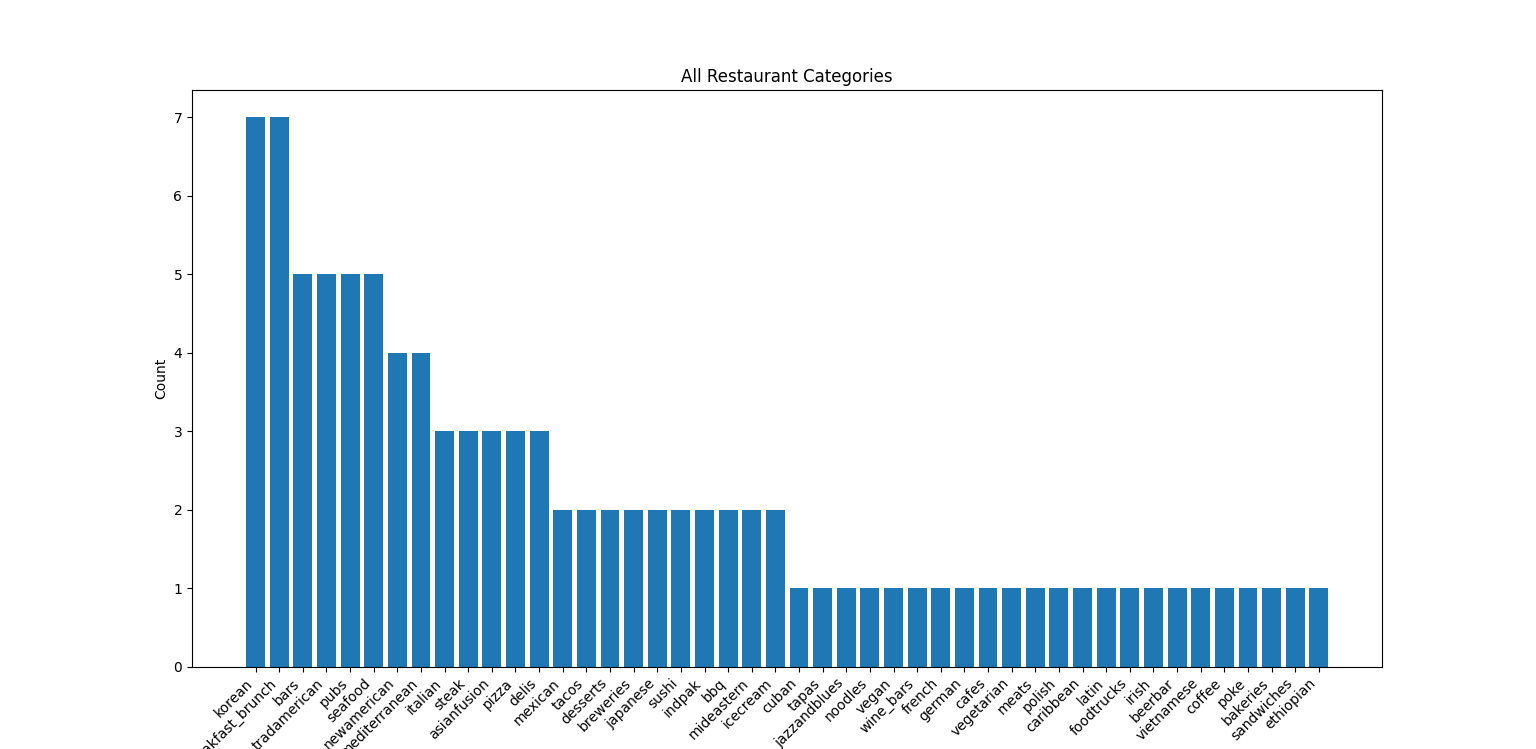
First problem we faced is that the U.S. The Census Bureau API key we obtained was only able to retrieve Ann Arbor’s data but not Ypsilanti’s (Ypsilanti works for some census results but not for the income data). After investigating, we found this was a bug or an internal mechanism of the API, so we had to hard code the data from the website for our use. Second problem is that initially we thought that restaurants would be categorized by their ethnic characters on Yelp, for example, we thought that a lot of the restaurants would be labeled as “Italian”, “Chinese”, “Japanese”, “Mideastern”, etc. However, many of the restaurants actually have very neutral categories like “Breakfast”, “Pub”, “Burger”, etc. This made it difficult to explore the relationship between the distribution of race in this city and the percentage of different ethnic restaurants.

**Calculation file:**

See calculation.py



**Visualizations**



**Instructions for running the code:**

Run Data\_Collection.py to store the data into a database restaurants\_data.db. Data\_Collection.py contains the code that collects data from the two APIs and the SQL prompts to create the tables and insert the data into the tables.

Run calculation.py to select and clean the data from the database and draw the charts. You might need to manually save the charts.

**Code Documentation:**

Data\_Collection.py:

**get\_race\_pop(city):** This function takes in the id of a city in Washtenaw City (for example "03000" stands for Ann Arbor) and returns a dictionary whose keys are different races and values are the population of each race.

**get\_income\_distribution(city):** This function takes in the id of a city in Washtenaw City (for example "03000" stands for Ann Arbor) and returns a dictionary whose keys are different income ranges and values are the proportion of each range.

**scale\_to\_100(income\_distribution):** The proportions in the dictionary generated by get\_income\_distribution() are not completely accurate because there are errors according to the original census data. This means the sum of all the percentages may not exactly be 1. This function re-weigh the numbers and makes their sum equal to 1.

**get\_median\_income\_by\_race(city):** This function takes in the id of a city in Washtenaw City (for example "03000" stands for Ann Arbor) and returns a dictionary whose keys are different races and values are the median household income of each race.

**get\_100\_restaurants(location):** This function takes in a location as a string formatted to fit in an url (for example “Ann Arbor” should be “Ann%20Arbor” since “%20” stands for space). Returns a nested dictionary.

**insert\_batch(data):** This function inserts a batch of rows into a database. It is used to meet the 25 limit requirement. Later I will set the batch size to 25 and insert 25 rows at a time (albeit I am still not sure if the requirement is met).

calculation.py:

**draw\_bar\_chart(categories):** This function takes in a list that contains all the categories to be put on the x-axis and draws a bar chart based on the list.

**price\_level\_to\_number(price\_list):** The price levels directly pulled from Yelp are “$”,”$$”, “$$$”, and “$$$$”. This function converts these into 1, 2, 3, and 4 so the average price level can be calculated.

**Documentation of resources used:**

<https://docs.developer.yelp.com/docs/fusion-intro>

<https://docs.developer.yelp.com/reference/v3_business_search>

<https://www.census.gov/data/developers/data-sets/acs-1year.html>

https://www.census.gov/data/developers/data-sets/decennial-census.html