Project Code:
https://github.com/ArchdukeCoker/SI507Final

README:
Packages:
from bs4 import BeautifulSoup
import requests
import
time import pandas as pd
import numpy as np
import random as random
 import matplotlib.pyplot as plt
import matplotlib from matplotlib.collections
import PatchCollection from matplotlib.patches
import Polygon from matplotlib.path import Path
import secrets as secrets

Keys needed: Census API key

Key can be received from the census website. Users interact with the program by inputting numbers that then give users the desired outputs.

Data Sources:
Wikipedia
Links:
1. https://en.wikipedia.org/wiki/List_of_United_States_cities_by_population – scraped HTML
2. https://en.wikipedia.org/wiki List_of_U.S._state_and_territory_abbreviations – scraped HTML
3-102: https://en.wikipedia.org/wiki/New_York_City (or another city name) – crawled HTML

First, I scraped the table on link 1 and got the: city name, city state, population, size in square miles, population density and the city page URL. I then scraped the second link and got the state names and corresponding two letter abbreviations. I then looped over the collected state names for the city table and made a new list of city abbreviations for each city. I compiled all of this information into a dictionary. After these steps I had collected 7 columns with 100 rows for and 2 columns of 50 rows for 800 records total.

After that, I used the city page URL to crawl Wikipedia and get the HTML of the first paragraph of each city instance. This brought back one paragraph for each city for a total of 100 records.

Caching is used on all Wikipedia processes to speed the process. This is particularly essential for the crawling phase as that is very time consuming for the computer.

Richmond

Link:
https://dsl.richmond.edu/panorama/redlining– JSON for each city with redlining data
I crawled through the various URLs to see which cities have JSON on historic redlining and returned this information to the program. I would estimate 75% of the cities have redlining data so that is a further 75 records. Caching is used here as well to speed repeat processes.

FCC:
Link:
https://geo.fcc.gov/api/census/area?lat={latitude}'&lon{longitude point}&format=json - JSON

This was used to get exact coordinates for each redlined district so that census data could then be used to make new maps. I would continue with my estimate that 75% of cities had redline data so this would be another 75 records. Caching was very necessary here, I had to make the grid have miniscule differences to work with the varied maps and district sizes of every city so this part of the process is very time consuming and caching.

Census
Links:
1.
https://api.census.gov/data/2015/acs/acs5?get=NAME,B19013_001E&for=tract:*&in=state:{2 digit state code}&key={census key } - JSON
2.
https://api.census.gov/data/2015/acs/acs5?get=NAME,B08303_001E&for=tract:*&in=state:{2 digit state code}&key={census key } - JSON

With the coordinates and redlining polygons, we are able to fill in census data with new information. We then use this information to make new maps with the historically redlined districts to see if any valuable insights can be gleaned. I would continue with my estimate that 75% of cities had redline data so this would be another 75 records. Caching was used here to speed the processing.

Database:
I did not complete the database portion. I saw that email that said you were ok with one day extensions however and I plan to complete this tomorrow but I would still like to turn what I have in tonight in case of disaster.

Interaction and Presentation Options:
The user is limited in what data they are able to see. The user interacts with the program through the command line. The user is given options in the form of numbers on the screen and they enter the number that corresponds to what they want to see.

They can go into the aggregate section of the program and see the summary statistics for population, size, and population density or they have the option of seeing histograms of these distributions to get a better sense of the data. If the user enters a bad input, they are gently told so and given the opportunity to try again. When a user is finished, they can enter ‘back’ or ‘exit’ to either return to the first part of the program or leave entirely.

If the user goes to the individual city data, they will be able to see the Wikipedia introductory paragraph for each and every city in the list of 100 most populous US cities. If there is redlining data, that map will be shown simultaneously with the city intro paragraph. If there is not redlining data, the user it told so and given an opportunity to choose a different city. If there is redlining data, users can then choose to see a map of how historic redlining relates to current median income in the district or median drive time to work in minutes. The user can hit back to see different cities, return to the aggregate part of the program, or the user can enter exit if they wish to leave entirely.

Demo Link:
https://youtu.be/zZVmQOzok20
* I have also included the video in my git repo as the Youtube version is very compressed and difficult to read.