Purpose

Often times in academic and corporate settings, engineers and professionals seek to derive meaningful information from data to meet a given objective. Before one can begin cleaning or analyzing data they first have to collect it. In some cases, data collection is as simple as downloading a CSV file, but this is infeasible for many applications. For example, an application that displays weather data should collect live data through an API instead of downloading a CSV file. Data exchange allows data to be shared between different computer programs. This assignment will give you experience in how data is exchanged by having you input, manipulate, and export data using different data exchange formats.

Skills

This assignment will give you practice with comma separated value (csv) files, Javascript Object Notation (json) files, scraping data from a website using the bs4 module, and accessing data stored in API using the requests module, all while transforming the raw data into the desired format. You will learn how to locate and make meaning from the information in these files and also create new files stored in each of these formats.

Knowledge

By working extensively with these types of data exchange formatted files, you will begin to see how data can be stored in many ways. In the future you will be better able to understand the differences between data exchange types and have a better understanding of how these data formats are selected to fit a particular purpose.

Important

1. Due Date: **10/09/2020 at 11:59 pm**
2. This homework is graded out of **100** points.
3. This is an individual assignment. You may collaborate with other students in this class. Collaboration means talking through problems, assisting with debugging, explaining a concept, etc. You should not exchange code or write code for others. For individual assignments, each student must turn in a unique program. Your submission must not be substantially similar to another student's submission. Collaboration at a reasonable level will not result in substantially similar code.
4. For Help:
    - TA Helpdesk / Piazza
    - Data Exchange Format Handout
    - Requests, RegEx, and BeautifulSoup Handout
5. Comment out or delete all your function calls. When your code is run, all it should do is run without any errors.
6. Do not wait until the last minute to do this assignment in case you run into problems.
7. **Read the entire specifications document before starting this assignment.**
8. **IF YOUR CODE CANNOT RUN BECAUSE OF AN ERROR, IT IS A 0%**

Introduction

Please read through the entire document before starting this homework. The goal of this homework is to demonstrate your understanding of data exchange formats (CSV and JSON), APIs, and HTMLs. You will be defining a few functions that deal with file input and output and data manipulation. You should test them out first on your own computer, then when you have one or more of them working, upload the entire file to GradeScope to see how well your code performs on the test cases we have created. The score shown on Gradescope will not reflect your final grade on this assignment, as we reserve the right to run different test cases after the deadline. You can submit the homework files as many times as you'd like before the deadline.

Allowed imports: `csv, json, requests, bs4, pprint, re`

pprint Module

When using Python to manipulate data, programmers will want to be able to see what is stored in their outputs and variables. While the traditional print statement will most likely suffice, there will be times where you need to visualize more complex data structures in a way that the traditional print statement fails to do. One way around this is to utilize pprint. Short for Pretty Printer, pprint is a native Python library which allows you to customize the formatting of your outputs. This is especially useful when handling large data, as it will format your output so it is more readable. Since you will work with large quantities of data for this assignment, we highly encourage you to import and utilize this library, as test cases will be showcased using this format.

```
>>> from pprint import pprint

>>> ta_list = [{'Manny': ['outgoing', 'courageous', 'caring'], 'major':
'ISYE'}, {'Josh': ['selfless', 'dependable', 'funny'], 'major': 'ISYE'},
{'Marylyn': ['gentle', 'kind-hearted', 'considerate'], 'major': 'ISYE'}]

>>> print(ta_list) # prints without formatting
[{'Manny': ['outgoing', 'courageous', 'caring'], 'major': 'ISYE'}, {'Josh':
['selfless', 'dependable', 'funny'], 'major': 'ISYE'}, {'Marylyn': ['gentle',
'kind-hearted', 'considerate'], 'major': 'ISYE'}]

>>> pprint(ta_list) # prints with pretty formatting
[
 {'Manny': ['outgoing', 'courageous', 'caring'], 'major': 'ISYE'},
 {'Josh': ['selfless', 'dependable', 'funny'], 'major': 'ISYE'},
 {'Marylyn': ['gentle', 'kind-hearted', 'considerate'], 'major': 'ISYE'}
]
```

Introduction to the API

For this assignment, you will be utilizing the *RESTCountries* API. This specific API allows users to access statistics such as currency codes, number of bordering countries, translations, region and subregions, and capitals pertaining to a specific country. The country and corresponding statistics can be accessed by utilizing various endpoints that API has to offer. You can access the *RESTCountries* API here:

[*RESTCountries*]

Since APIs often return very long and complicated dictionaries, it is advisable to use Python's built–in module called `pprint` in order to be able to better see the keys and values of your returned dictionary.

Google Chrome Inspect Tool

When dealing with web scraping, a handy tool to utilize is Google Chrome's Inspect Tool. This allows you to better visualize and read the source code for each HTML page.

There are 3 ways to access the Inspect Tool:

1. Click on the three vertical dots in the upper right hand corner → More tools → Developer Tools
2. Right–click on the page and choose "Inspect"
3. Ctrl + Shift + C (**Windows**) or Cmd + Opt + C (**Mac**)

You should now be able to see the page's source code include other various details. You can select the square with the cursor at the top left corner of the menu to bring up "Inspect Element" mode, which will highlight the specific HTML line when hovering over your page.

Data Sources

For this assignment, you will be utilizing four different data files to complete your functions. The `countries.csv` was pulled from **Fernanado's** Countries of the World dataset, `additional_stats.json` data were extracted and modified from **Samayo's** country–json project on GitHub, `companies.csv` was pulled and heavily filtered from **People Data Labs'** 7+ Million Company Dataset, and `foreign_trade.html` was extracted from **census.gov's** Foreign Trade–U.S. page source.

Functions

A grading rubric including point distribution is provided at the end of this document.

Function name: `csv_parser`
Parameter(s): `filename` (str)
Return Type: list
Description: Write a function that reads in and cleans the data from the CSV file, whose name is passed into the function as `filename`, and returns a list of lists containing the data. Each sub–list is a list representing the information of each row (data for a given country) in the CSV file. You should format each sub–list in the following order as directed:

Column	Expected Data Type	Other Requirements
`Country`	str	Strip any outer whitespaces
`Region`	str	Strip any outer whitespaces
`Population`	int	
`Pop. Density (per sq. mi.)`	float	Replace commas with periods before float conversion
`GDP ($ per capita)`	int	Set to `"Unknown"` if empty
`Literacy (%)`	float	Replace commas with periods before float conversion, set to `"Unknown"` if empty

Note the first list within your returned value should be a list of the specific header names. You should preserve the order of rows as found in the csv file.

You must include the file extension when passing in your opening your file using the `filename` parameter. If you are experiencing difficulty in reading the csv file and receive an encoding error, try specifying the `"utf8"` encoding:

```
open("example.csv", encoding = "utf8")
```

Below is a snippet of the returned value should look like:

Python Test Case:

```
>>> csv_parser('countries')
[
 ['Country',
  'Region',
  'Population',
  'Pop. Density (per sq. mi.)',
  'GDP ($ per capita)',
  'Literacy (%)'],
 ['Afghanistan', 'ASIA (EX. NEAR EAST)', 31056997, 48.0, 700, 36.0],
 ['Albania', 'EASTERN EUROPE', 3581655, 124.6, 4500, 86.5],
 ['Algeria', 'NORTHERN AFRICA', 32930091, 13.8, 6000, 70.0],
 ['American Samoa', 'OCEANIA', 57794, 290.4, 8000, 97.0],
 ['Andorra', 'WESTERN EUROPE', 71201, 152.1, 19000, 100.0],
 ['Angola', 'SUB-SAHARAN AFRICA', 12127071, 9.7, 1900, 42.0],
 ['Anguilla', 'LATIN AMER. & CARIB', 13477, 132.1, 8600, 95.0],
 ['Antigua And Barbuda', 'LATIN AMER. & CARIB', 69108, 156.0, 11000, 89.0],
 ['Argentina', 'LATIN AMER. & CARIB', 39921833, 14.4, 11200, 97.1],
 ['Armenia', 'C.W. OF IND. STATES', 2976372, 99.9, 3500, 98.6],
 ['Aruba', 'LATIN AMER. & CARIB', 71891, 372.5, 28000, 97.0],
 ['Australia', 'OCEANIA', 20264082, 2.6, 29000, 100.0],
 ['Austria', 'WESTERN EUROPE', 8192880, 97.7, 30000, 98.0],
 ['Azerbaijan', 'C.W. OF IND. STATES', 7961619, 91.9, 3400, 97.0],
 ['Bahamas', 'LATIN AMER. & CARIB', 303770, 21.8, 16700, 95.6],
 ['Bahrain', 'NEAR EAST', 698585, 1050.5, 16900, 89.1],
 ['Bangladesh', 'ASIA (EX. NEAR EAST)', 147365352, 1023.4, 1900, 43.1],
 ['Barbados', 'LATIN AMER. & CARIB', 279912, 649.5, 15700, 97.4],
 ['Belarus', 'C.W. OF IND. STATES', 10293011, 49.6, 6100, 99.6],
 ['Belgium', 'WESTERN EUROPE', 10379067, 340.0, 29100, 98.0],
 ['Belize', 'LATIN AMER. & CARIB', 287730, 12.5, 4900, 94.1],
 ['Benin', 'SUB-SAHARAN AFRICA', 7862944, 69.8, 1100, 40.9],
 ['Bermuda', 'NORTHERN AMERICA', 65773, 1241.0, 36000, 98.0],
 ['Bhutan', 'ASIA (EX. NEAR EAST)', 2279723, 48.5, 1300, 42.2],
 ['Bolivia', 'LATIN AMER. & CARIB', 8989046, 8.2, 2400, 87.2],
 ['Bosnia And Herzegovina', 'EASTERN EUROPE', 4498976, 88.0, 6100, 'Unknown'],
 ['Botswana', 'SUB-SAHARAN AFRICA', 1639833, 2.7, 9000, 79.8],
 ['Brazil', 'LATIN AMER. & CARIB', 188078227, 22.1, 7600, 86.4],
 ['British Virgin Islands', 'LATIN AMER. & CARIB', 23098, 151.0, 16000, 97.8],
 ['Brunei', 'ASIA (EX. NEAR EAST)', 379444, 65.8, 18600, 93.9],
 ['Bulgaria', 'EASTERN EUROPE', 7385367, 66.6, 7600, 98.6],
...
]
>>> len(csv_parser('countries'))
217
```

Function name: `json_parser`
Parameter(s): `filename` (str), `data` (list)
Return Type: list
Description: Write a function that reads in and cleans the data from the JSON file, whose name is passed in to the function as `filename`, and combines together with the cleaned list from `csv_parser` given as `data`. You should return a list of dictionaries, where each dictionary represents information for a given country. You should format each dictionary as listed:

Key	Value	Expected Data Type	Additional Info
`Country`	Name of Country	str	Given by `data`
`Region`	Name of Region	str	Given by `data`
`Population`	Population number	int	Given by `data`
`Pop. Density (per sq. mi.)`	Pop. Density number	float	Given by `data`
`GDP ($ per capita)`	GDP number	int	Given by `data`
`Literacy (%)`	Literacy number	float	Given by `data`
`Languages`	Split languages into different strings, Set to `None` if empty	list	Found in JSON file
`National Dish`	Set to "`Unknown`" if None	str	Found in JSON file
`Religion`	Set to "`Unknown`" if None	str	Found in JSON file
`Government`	Set to "`Unknown`" if None	str	Found in JSON file
`Currency Name`	Set to "`Unknown`" if None	str	Found in JSON file

Finally, you should sort your list based on the `Country` name in alphabetical order before returning. You **should not include** the header row in your returned list. Make sure to include the file extension when opening your file.
Note: there are more countries listed in the JSON file than in the `data` list. **Do not include** any countries listed in the JSON file that are not present within the `data` list.
Python Test Case:

```
>>> data = csv_parser('countries')
>>> json_parser('additional_stats', data)
[
 {'Country': 'Afghanistan',
  'Currency Name': 'Afghanistan Afghani',
  'GDP ($ per capita)': 700,
  'Government': 'Islamic Emirate',
  'Languages': ['Balochi', 'Dari', 'Pashto', 'Turkmenian', 'Uzbek'],
  'Literacy (%)': 36.0,
  'National Dish': 'Kabuli Palaw',
  'Pop. Density (per sq. mi.)': 48.0,
  'Population': 31056997,
  'Region': 'ASIA (EX. NEAR EAST)',
  'Religion': 'Islam'},
 {'Country': 'Albania',
  'Currency Name': 'Albanian Lek',
  'GDP ($ per capita)': 4500,
  'Government': 'Republic',
  'Languages': ['Albaniana', 'Greek', 'Macedonian'],
  'Literacy (%)': 86.5,
  'National Dish': 'Tavë kosi',
  'Pop. Density (per sq. mi.)': 124.6,
  'Population': 3581655,
  'Region': 'EASTERN EUROPE',
  'Religion': 'Islam'},
 {'Country': 'Algeria',
  'Currency Name': 'Algerian Dinar',
  'GDP ($ per capita)': 6000,
  'Government': 'Republic',
  'Languages': ['Arabic', 'Berberi'],
  'Literacy (%)': 70.0,
  'National Dish': 'Couscous',
  'Pop. Density (per sq. mi.)': 13.8,
  'Population': 32930091,
  'Region': 'NORTHERN AFRICA',
  'Religion': 'Islam'},
...
]
>>> len(json_parser('additional_stats', data))
216
```

Function name: `company_parser`
Parameter(s): `filename` (str), `data` (list)
Return Type: dict
Description: Write a function that passes in the `filename` of the CSV containing company information and the cleaned data from `json_parser` as `data`. You should parse through the company dataset and return a dictionary with the following keys and values:

Main Key	Main Value	Sub–dictionaries	
`Country` (Must **capitalize** each word of the country)	Dictionary containing specific information of each country	**Sub Key**	**Sub Value**
		`GDP ($ per capita)`	Number of GDP
		`businesses`	List of businesses, with each word of the business name **capitalized**
		`industries`	List of **unique** industries
		`estimated_employees`	Sum of total estimated employees for each country

Additional guidelines to follow if a business's home country does not appear within the given `data`:

- Create a new Country key called `"Unknown"`
- Set the `GDP ($ per capita)` to 0
- Add the `businesses`, `industries`, and `estimated_employees` to under `"Unknown"`

Return the manipulated dictionary after adding in the corresponding statistics listed above.
Hint: You can capitalize each word of a string in the following manner:

```
>>> country = "united states"
>>> country = " ".join([word.capitalize() for word in country.split()])
```

Note: The Python test case has been truncated. Instead, you have been provided a file titled `key.json` that will represent what the entire correct output will look like. Since your returned value is a dictionary, the keys do not have to be in any specific order.

Python Test Case:

```
>>> data = csv_parser('countries')
>>> data = json_parser('additional_stats', data)
>>> company_parser('companies', data)
{
 'Afghanistan': {'GDP ($ per capita)': 700,
                 'businesses': ['Etisalat Afghanistan',
                                'Roshan',
                                ...,
                                'Pajhwok Afghan News',
                                'Aria Target Logistics Services'],
                 'estimated_employees': 5560,
                 'industries': ['telecommunications',
                                'banking',
                                ...,
                                'program development',
                                'online media']},
 'Albania': {'GDP ($ per capita)': 4500,
             'businesses': ['Albtelecom Albania',
                            'Telekom Albania',
                            ...,
                            'Turgut Ozal Colleges',
                            'Albanian Armed Forces'],
             'estimated_employees': 8142,
             'industries': ['telecommunications',
                            'banking',
                            ...,
                            'education management',
                            'military']},
...,
'Unknown': {'GDP ($ per capita)': 0,
             'businesses': ['Nhs',
                            'Sap',
                             ...,
                             'Subway',
                            'Nessuna Azienda']
              'estimated_employees': 1187124,
              'industries': ['hospital & health care',
                            'computer software',
                             ...,
                             'restaurants',
                            'writing and editing']},
...
}
>>> print(len(company_parser('companies', data)))
217
```

Function name: `country_stats`
Parameter(s): `json_filename` (str), `txt_filename` (str), `data` (dict)
Return Type: str
Description: Write a function that passes in the `json_filename`, `txt_filename` and `data`, which represents the name of the JSON and TXT files you will be writing to and the data returned from `company_parser` that you will export. You should write the complete dictionary given by `data` to your JSON file first. After you are done writing to the JSON file, you should then write the following lines to the TXT file:

```
"{country} has a total of {number of businesses} businesses, an
estimated {number of employees} employees, a total of {number of
        industries} industries, and total GDP of {GDP}."
```

You should write this line for each country present in `data`, along with the appropriate stats. You should write each country in alphabetical order. After successfully writing your data to both files, you should return the string:

`"Data successfully exported."`

Python Test Case:

```
>>> data = csv_parser('countries')
>>> data = json_parser('additional_stats')
>>> data = company_information('companies', data)
>>> country_stats('country_stats', 'summary', data)
'Data successfully exported.'
```

The following screenshots are what your exported files may look like. You have been provided the files `key.txt` and `key.json` that will represent what your file should look like.

summary.txt

```
1  Afghanistan has a total of 28 businesses, an estimated 5560 employees, a total of 15 industries, and total GDP of 700.
2  Albania has a total of 28 businesses, an estimated 8142 employees, a total of 16 industries, and total GDP of 4500.
3  Algeria has a total of 28 businesses, an estimated 36466 employees, a total of 17 industries, and total GDP of 6000.
4  American Samoa has a total of 3 businesses, an estimated 396 employees, a total of 3 industries, and total GDP of 8000.
5  Andorra has a total of 28 businesses, an estimated 3044 employees, a total of 19 industries, and total GDP of 19000.
6  Angola has a total of 28 businesses, an estimated 7868 employees, a total of 12 industries, and total GDP of 1900.
7  Anguilla has a total of 4 businesses, an estimated 198 employees, a total of 4 industries, and total GDP of 8600.
8  Antigua And Barbuda has a total of 0 businesses, an estimated 0 employees, a total of 0 industries, and total GDP of 11000.
9  Argentina has a total of 28 businesses, an estimated 141471 employees, a total of 16 industries, and total GDP of 11200.
10 Armenia has a total of 28 businesses, an estimated 7297 employees, a total of 14 industries, and total GDP of 3500.
11 Aruba has a total of 17 businesses, an estimated 1029 employees, a total of 13 industries, and total GDP of 28000.
12 Australia has a total of 28 businesses, an estimated 657591 employees, a total of 17 industries, and total GDP of 29000.
13 Austria has a total of 28 businesses, an estimated 129735 employees, a total of 22 industries, and total GDP of 30000.
14 Azerbaijan has a total of 28 businesses, an estimated 15847 employees, a total of 13 industries, and total GDP of 3400.
15 Bahamas has a total of 28 businesses, an estimated 4616 employees, a total of 17 industries, and total GDP of 16700.
16 Bahrain has a total of 28 businesses, an estimated 20955 employees, a total of 15 industries, and total GDP of 16900.
17 Bangladesh has a total of 28 businesses, an estimated 46053 employees, a total of 12 industries, and total GDP of 1900.
18 Barbados has a total of 21 businesses, an estimated 918 employees, a total of 18 industries, and total GDP of 15700.
19 Belarus has a total of 28 businesses, an estimated 9094 employees, a total of 11 industries, and total GDP of 6100.
20 Belgium has a total of 28 businesses, an estimated 315170 employees, a total of 17 industries, and total GDP of 29100.
```

country_stats.json


```
{
    "Afghanistan": {
        "businesses": [
            "Etisalat Afghanistan",
            "Roshan",
            "Awcc",
            "Mtn Afghanistan",
            "Aib Bank",
            "Afghan United Bank",
            "Da Afghanistan Breshna Sherkat",
            "Salaam (afghan Telecom Gsm)",
            "Afgs",
            "Kam Air",
            "Afghanistan Research And Evaluation Unit",
            "State Corps",
            "Maiwand Bank",
            "Ace Hardware",
            "Azizi Bank",
            "Leighton Construction Company",
            "Afghanistan Holding Group",
            "Kardan University",
            "The First Microfinancebank - Afghanistan (fmfb-a)",
            "Afghanistan International Bank",
            "Finca Afghanistan",
            "Sanayee Development Organization",
            "Ministry Of Energy And Water Afghanistan",
            "Integrity Watch Afghanistan",
            "Capital Region Independent Development Authority-crida",
            "Dutch Committee For Afghanistan - Veterinary Programmes",
            "Pajhwok Afghan News",
            "Aria Target Logistics Services"
        ],
        "GDP ($ per capita)": 700,
        "industries": [
            "telecommunications",
            "banking",
            "utilities",
            "logistics and supply chain",
            "airlines/aviation",
            "research",
            "construction",
            "retail",
            "management consulting",
            "education management",
            "non-profit organization management",
            "civil engineering",
            "government relations",
            "program development",
            "online media"
        ],
        "estimated_employees": 5560
    },
```

Function name: `inequality`
Parameter(s): `region` (str) `gini_val` (float)
Return Type: dict
Description: Write a function to return a dictionary of countries mapped to their GINI coefficient within the specified region that exceed the given GINI coefficient threshold. You should use the REST Countries API in this function. If the specified region is not a valid region in the API, return the following string:
`"The given region was not found"`.
Python Test Cases:

```
>>> inequality('Europe', 40)
{'Macedonia (the foreigner Yugoslav Republic of)': 43.2, 'Russian Federation':
40.1}
>>> inequality('Africa', 63.9)
{'Comoros': 64.3, 'Seychelles': 65.8}
>>> inequality('Africa', 73.9)
{}
>>> inequality('Mars', 30)
'The given region was not found'
```

Function name: `html_parser`
Parameter(s): `filename` (str)
Return Type: tuple
Description: Write a function that reads in and cleans the data of specific tables from theHTML file, whose name is passed in to the function as `filename`, and returns a tuple containing data from each specified table. Your function should read and parse data from two different tables from the HTML file: the ***Year–to–Date Exports*** and the ***Year–to–Date Imports***.
For each table, you should scrape the data into a list of sub–lists, where each sub–list represents a row of data from that table. You should return the data while adhering to the following conditions:

- Include each header row for both tables
- Do not include the rows containing data pertaining to ***Total, All Countries*** or ***Total, Top 15 Countries***
- Convert the values of the ***Rank*** column to type `int`
- Convert the values of the ***Exports*** or ***Imports*** column to type `float`
- Strip any outer whitespaces from each text value
- Preserve the order of the rows as shown in the tables

Finally, you should return a tuple containing both lists, where the `0`th index of the tuple is the ***Foreign Trade Exports*** data and the `1`st index of the tuple is the ***Foreign Trade Imports*** data.

Although you should be scraping the data from the HTML file provided, you may view the live representative website, where the HTML page source was extracted: **Foreign Trade - US Trade**. We **strongly suggest** you utilize Google Chrome's Inspect Tool to inspect the page source so you can identify the parts of the page source code efficiently.

Python Test Case:

```
>>> exports, imports = html_parser('foreign_trade')
>>> exports
[
 ['Rank', 'Country', 'Exports', 'Percent of Total Exports'],
 [1, 'Canada', 140.1, '17.5%'],
 [2, 'Mexico', 117.5, '14.7%'],
 [3, 'China', 58.5, '7.3%'],
 [4, 'Japan', 38.0, '4.8%'],
 [5, 'United Kingdom', 33.8, '4.2%'],
 [6, 'Germany', 32.5, '4.1%'],
 [7, 'Korea, South', 30.6, '3.8%'],
 [8, 'Netherlands', 26.1, '3.3%'],
 [9, 'Brazil', 19.8, '2.5%'],
 [10, 'Taiwan', 17.4, '2.2%'],
 [11, 'France', 16.7, '2.1%'],
 [12, 'Belgium', 16.1, '2.0%'],
 [13, 'Singapore', 16.1, '2.0%'],
 [14, 'India', 15.0, '1.9%'],
 [15, 'Hong Kong', 13.5, '1.7%']
]
>>> imports
[
 ['Rank', 'Country', 'Imports', 'Percent of Total Imports'],
 [1, 'China', 221.9, '17.3%'],
 [2, 'Mexico', 173.1, '13.5%'],
 [3, 'Canada', 148.4, '11.6%'],
 [4, 'Japan', 65.4, '5.1%'],
 [5, 'Germany', 63.3, '5.0%'],
 [6, 'Switzerland', 48.8, '3.8%'],
 [7, 'Korea, South', 42.5, '3.3%'],
 [8, 'Vietnam', 40.6, '3.2%'],
 [9, 'Ireland', 38.0, '3.0%'],
 [10, 'Taiwan', 33.0, '2.6%'],
 [11, 'United Kingdom', 28.7, '2.2%'],
 [12, 'Italy', 26.9, '2.1%'],
 [13, 'India', 26.7, '2.1%'],
 [14, 'France', 24.9, '1.9%'],
 [15, 'Malaysia', 23.3, '1.8%']
]
```

Bonus

There are **10** points of total bonus that you can earn for this assignment. If you choose to do the bonus, then you must have one function titled `bonus` in which you must meet the following conditions in order to have your queries be eligible for the **10** bonus points.

Function name: `bonus`
Parameter(s): `any`
Return Type: any
Description: The goal of this function is to generate and answer a real–world insightful question or analysis by using the datasets in this assignment alongside any other related data sources. Since this function is open–ended, you may choose whichever topic that interests you as long as you include at least one data source provided in this assignment in your analysis. To be eligible to obtain the full credit, you must adhere to the following conditions:

- Function must not error
- The function must answer a question that is practical and meaningful while using non-trivial manipulation
    - For example, finding the maximum number in a given list may be meaningful but it is only one line of code
- Must provide a comment or a docstring that thoroughly explains why the question that the function answers is meaningful, insightful, and provides value to the stakeholders involved
- Must read in data from at least 1 additional file/source not provided (csv, txt, json, html, API)
    - **Please include any static files with your submission**
- Must have rigorous data cleaning and manipulation process to answer insightful question
- Must write a summary of the manipulated data that outputs to at least 1 file in the following formats: *csv, txt, json, html xml*
    - This summary file should be an aggregated output of the inputted data files showing quantitative summary statistics or high level numerical insights that can be derived from the given data
    - For example, if the input data is COVID cases for each county in Georgia from March to September then a summary file could be the total cases for each county across all time periods
- Must provide an example test case which calls the bonus function

Failure to meet any components of the conditions outlined above will result in partial or complete deduction of possible bonus points.

Gradescope Requirements

- **NO PRINT STATEMENTS.** this will break the autograder
- **NO FUNCTION CALLS** outside of function definitions. This will also break the autograder
- Make sure the file submitted is titled **HW02.py**
- **Only import csv, json, requests, bs4, re, and pprint modules**
- Up to **10 points of this grade may be lost** if you are randomly selected to do a Code Review with a TA and you don't do within one week of the homework's due date or are unable to explain your code.
- **All points of this grade will be lost** if it is shown that you have collaborated in an unauthorized manner on this assignment or otherwise violated the Georgia Tech honor code.

Grading Rubric

```
csv_parser                                  20  pts
json_parser                                 20  pts
company_parser                              20  pts
country_stats                               15  pts
inequality                                  10  pts
html_parser                                 15  pts
bonus                                       10  pts
_____________________________________________________
Total Possible                          110/100 pts
```