This project involves performing basic analytics on a large-scale dataset using Python on a single machine. The dataset is a subset of Yelp's businesses, reviews, and user data. It was originally put together for the Yelp Dataset Challenge, which is a chance for students to conduct research or analysis on Yelp's data and share their discoveries. The dataset consists of five JSON files, which contain information about businesses across 8 metropolitan areas in the USA and Canada. The dataset can be found and downloaded [here](https://www.kaggle.com/yelp-dataset/yelp-dataset/version/3) (registration to Kaggle is required).

In this project, you will write four Python scripts/programs. The first program (**dstats.py**) performs descriptive analytics of the dataset. The second (**dist.py)** computes useful frequency distributions. The third (**network.py**) constructs a social network of Yelp friends. The fourth (**graph.py**) performs basic network analytics.

* Write the dependencies of your implementation(solution) into *requirements.txt* in requirements format (using the [pip freeze](https://pip.pypa.io/en/stable/cli/pip_freeze/) command).
* All scripts are to be written using Python >= 3.7.0.
* **You should round your answer to at most 2 decimal places** (only if necessary) using the built-in round function

need to zip your repository and submit as one zip file with the name of project1.zip

*project1.zip* must look like this:

EECS4415\_Project\_1/

├─ dstats.py

├─ dist.py

├─ network.py

├─ graph.py

├─ requirements.txt

├─ info.txt

├─ README.md

├─ .gitignore

Task1)

Given a collection of businesses in a file /path-to-file/filename.json, a two-letter state/province abbreviation ST (case-sensitive), and a name of a city city (case-sensitive), write a Python script (*dstats.py*) that computes and writes the answer to a text file named ***Q1.out*** in the **current working directory**. The file *Q1.out* consists of six line-separated numbers as follows:

* The number of businesses in the city, ST
* The average number of stars of businesses in the city, ST
* The number of restaurants in the city, ST
* The average number of stars of restaurants in the city, ST
* The average number of reviews for all businesses in the city, ST
* The average number of reviews for restaurants in the city, ST

can use the original file provided by Kaggle (*[yelp\_academic\_dataset\_business.json](https://www.kaggle.com/yelp-dataset/yelp-dataset/version/3?select=yelp_academic_dataset_business.json)*) for evaluation. The variables **/path-to-file/filename.json** and **ST** need to be passed to the script as command line arguments. The *[argparse](https://docs.python.org/3/library/argparse.html)* module makes it easy to write user-friendly command-line interfaces.

Your script should be run as follows:

$ python3 dstats.py /path-to-file/filename.json city ST

For example:

$ python3 dstats.py yelp\_academic\_dataset\_business.json Vancouver BC

Sample output (*Q1.out*):

10299

3.6

4275

3.51

41.63

69.26

Task 2)

Given a collection of businesses in a file /path-to-file/filename.json, a two-letter state/province abbreviation ST (case-sensitive), and a name of a city city (case-sensitive), write a Python script (dist.py) that performs the following tasks:

1. For all restaurants in the city, ST, compute the frequency distribution of the number of restaurants in each category of restaurants (e.g., Japanese, Chinese, Canadian, Italian, etc.). Your script should only consider restaurant categories that are based on geographical origin. For example, "Mediterranean" is a legit restaurant category while "Sandwiches" is not. Please note that a restaurant can fall into multiple categories. Write the **top-10** categories to a text file named ***Q2\_part1.out*** in the **current working directory**. The output should be one line per pair of values as follows:

category:#restaurants

For example:

Japanese:525

Chinese:425

Canadian (New):345

Italian:230

Vietnamese:216

American (New):182

American (Traditional):181

Asian Fusion:177

Mediterranean:149

Indian:144

1. For all restaurants in the city, ST, compute the frequency distribution of the number of reviews submitted for each category of restaurants (e.g., Japanese, Chinese, Canadian, Italian, etc.). Your script should only consider restaurant categories that are based on geographical origin. For example, "Mediterranean" is a legit restaurant category while "Sandwiches" is not. Please note that a restaurant can fall into multiple categories. Write the top-10 most reviewed categories in descending order (from the most reviewed category to the least reviewed) to a text file named ***Q2\_part2.out*** in the **current working directory**. The output should be one line per triplet as follows:

category:#restaurants:average\_review\_count

For example:

Japanese:48181:91.77

Canadian (New):39046:113.18

Chinese:21924:51.59

American (New):21764:119.58

Italian:19729:85.78

American (Traditional):17222:95.15

Vietnamese:13829:64.02

Asian Fusion:12416:70.15

Middle Eastern:11849:101.27

French:11735:126.18

1. Create a bar chart that shows the **top-5** (NOT top-10) restaurant categories identified in part (1), where the x-axis represents the restaurant category, and the y-axis represents its frequency (#restaurants). The size of the bar chart should be **10-inch-by-10-inch**. The chart should be properly labeled. Save the plot as a **PDF** file named ***Q2\_part3.pdf*** in the **current working directory**. We use the original file provided by Kaggle (*[yelp\_academic\_dataset\_business.json](https://www.kaggle.com/yelp-dataset/yelp-dataset/version/3?select=yelp_academic_dataset_business.json)*) for evaluation. Your script should be run as follows:

$ python3 dist.py /path-to-file/filename.json city ST

For example:

$ python3 dist.py yelp\_academic\_dataset\_business.json Vancouver BC

**Hint:** Use the [*matplotlib*](https://matplotlib.org/stable/index.html) package to create the plot. You can follow this [tutorial](https://matplotlib.org/stable/plot_types/basic/bar.html#sphx-glr-plot-types-basic-bar-py) about using *matplotlib* to create a bar chart.

**Task3)**

The social network will be represented as a graph G(V,E), where V is a set of vertices representing the Yelp users and E is a set of edges representing friendships between Yelp users. The graph/network should be represented in a file using the edge list format. An edge list is a list that represents all the edges in a graph. Each edge is represented as a space-separated pair of vertices. For example, a small fully connected triangle-like graph between vertices a1, a2, a3 would be represented in the edge list as:

a1 a2

a2 a3

a3 a1

Note that the order of the lines does not matter, and edges are bidirectional (so either "a1 a2" or "a2 a1" should be listed but NOT both).

Given a collection of users in a file /path-to-file/filename.json and an integer n (n >= 100), write a Python script (network.py) that creates the social network of Yelp friends among Yelp users who sent **no less than n useful votes**, and writes the edge list of the created graph to a text file named **Q3.out** in the **current working directory**. Your script should only consider Yelp users who sent no less than n useful votes. For example, users a1 and a2 are friends, who sent n+1 and n-1 useful votes, respectively. In this case, neither "a1 a2" nor "a2 a1" should be listed.

We use the original file provided by Kaggle ([yelp\_academic\_dataset\_user.json](https://www.kaggle.com/yelp-dataset/yelp-dataset/version/3?select=yelp_academic_dataset_user.json)) for evaluation.

Your script should be run as follows:

$ python3 network.py /path-to-file/filename.json n

For example, the following command should create the social network of Yelp friends among Yelp users who sent no less than 100 useful votes.

$ python3 network.py yelp\_academic\_dataset\_user.json 100

**Task 4)**

Given a Yelp social network as an edge list in a text file /path-to-file/filename.txt (the format is the same as the Q3 output), write a Python script (*graph.py*) that computes the following network statistics and writes the answer to a text file named ***Q4.out*** in the **current working directory**.

* the number of vertices (*|V|*) and the number of edges (*|E|*) of the network. The output should be two space-separated integers.
* The average node degree of the graph (*avgNodeDegree*). The degree of a node is the number of edges that are incident to the node (i.e., #neighbors).
* The number of connected components in the network (*#components*). A connected component is a connected subgraph that is not part of any larger connected subgraph. The connected components of any graph partition its vertices into disjoint sets, and are the induced subgraphs of those sets. A graph that is itself connected has exactly one component, consisting of the whole graph.
* The number of triangles in the network (*#triangles*). For example, vertices *a1, a2, a3* (the order doesn't matter) form a triangle in the social network if *a1* and *a2* are friends, *a1* and *a3* are friends, and *a2* and *a3* are friends.

The output (Q4.out) should be one line per answer to the question as follows:

|V| |E|

avgNodeDegree

#components

#triangles

For example:

4415 402945

182.53

3

10116817

Your script should be run as follows:

$ python3 graph.py /path-to-file/filename.txt

For example:

$ python3 graph.py graph.txt