This report describes the process of data cleaning and completion using the source provided. This report also provides details on extracting data and plotting it using graphic libraries.

Note: The program code is provided individually for each question in the root folder. The program can be executed by following the README.md instructions. The questions from the root directory contain method calls from the other files in the "mongo" subfolder to make the code more manageable and readable. The global settings are provided in "globals.py" inside the root folder.

Question 1: Briefly describe the information stored in the file.

The file contains external data for users, which is presumably shared across different Stack Exchange websites using the account (ID). The code to analyze the data is provided in "q1.py". Running the file stores the results in the "file_info.txt" file in the root directory. The following table explains the analysis of data present in the given file. It has data from 155 unique sites for 429 unique accounts.

Total Records: 7140

Sr.	Key	Type	Nulls	Unique	Example Value
1	account_id	Integer	0	429	102248
2	answer_count	Integer	0	253	16755
3	badge_counts.bronze	Integer	0	189	2917
4	badge_counts.gold	Integer	0	66	268
5	badge_counts.silver	Integer	0	158	2584
6	creation_date	Timestamp	0	7044	1218724709
7	last_access_date	Timestamp	0	7105	1705674547
8	question_count	Integer	0	130	1163
9	reputation	Integer	0	1190	1037034
10	site_name	Text	0	155	Long text
11	site_url	Text	0	155	Long text
12	user_id	Integer	0	6507	1037104

Question 2: Modifying your MongoDB Users collection to include the Account Id for each user from the original dataset. Then, add the sites information in users collection (specifically, include the reputation, number of each class of badge received for each other site, along with the site name). Provide a program to perform this operation and report how many successful updates you were able to perform with the given data.

The code to modify "users" collection can be run by executing "q2.py" from the root folder. The code first modifies the collection by inserting the account Id key for all users. As all users contain the account Ids therefore the operation updates all the users with account Ids. I map this data by reading Ids and Account Ids from the "users" dataset and map to the Ids from the MongoDB collection to make the accurate updates.

Secondly, to update "users" collection with the sites data provided as "extra data". As it is relevant to match records based on the account Ids, therefore, I devise the plan to map on that. I first read the data from the given file particularly account Id, site name, and reputation and badges for each site name for each account Id. To ensure the accurate insertion, I first find the unique accounts, and then select unique sites for each account Id. This ensures that no duplicate sites are added for any user (account Id). Then, I map the selected data for the account id read from the given data to the account Id of "users" collection in the database. As the provided data is unique for each site for all accounts, the total number of updates records (Sites inserted) are **7140**.

Question 3: Assume that the documents in the file did not contain the account IDs, perform the matching process. Analyze the issues and consider how many values in the new data set match existing documents and the cases where new values match more than one existing document. Include the results in your report.

If there is no account id provided in the new documents. There is only one accurate way to match records is by use the site name "Ask Ubuntu". This filters out all the records to select only the data where the site name is "Ask Ubuntu". As we are certain that the MongoDB "users" collection contains data for users from "Ask

Ubuntu" subset from Stack Exchange data. This will ensure that a correct matching process is carried out. However, this will greatly reduce the number of updates as only to include data from one site. This is the same for matching records if I map on the site URL to insert data for "ask Ubuntu's URL" based on user id. **(Matched 364 records)**

There is no logical way to map the data based on other attributes provided in the given dataset. For instance,

- The closest attribute to check is user ID. However, it is clear that each site assigns a different user ID to each user. Hence, no more mappings other than the specified above (e.g., using site name or site URL) can be accurately checked. **(Matched 4440 records; however incorrect matching)**
- The attributes answers count, badges counts, questions count, and reputation does not add any information to check against.
- The attributes creation date, and last access date can be used to match records. But in they represent the timestamp when each time record is updated. This is also potentially illogical except outside the same site (Ask Ubuntu). Hence, no records can be accurately mapped. **(No records matches; however incorrect matching)**

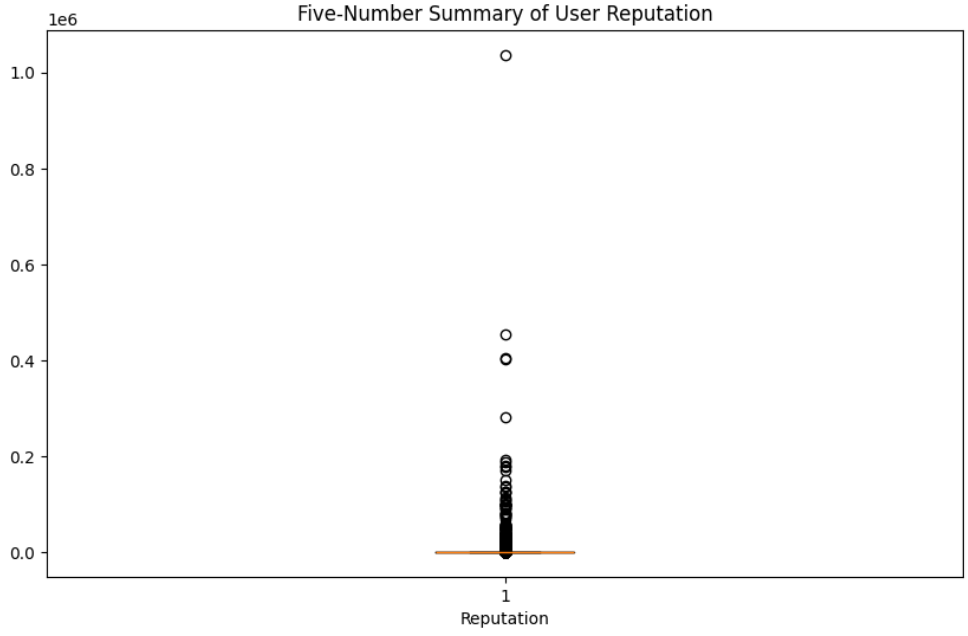
The code analyze and report matching process can be run by executing "q3.py" from the root folder. The code first reports count of matches based on existing account IDs to provide the benchmark. Then, the code execution reports matching on the each of the above attributes. The output can be observed on the terminal for each case.

Question 4: Provide programs that extract the following data to be plotted from MongoDB. Report your results and provide plots in the report.

The code to generate and save plots can be run by executing "q2.py" from the root folder. The reports the data on the console and saves the plots in the root folder as PNG files.

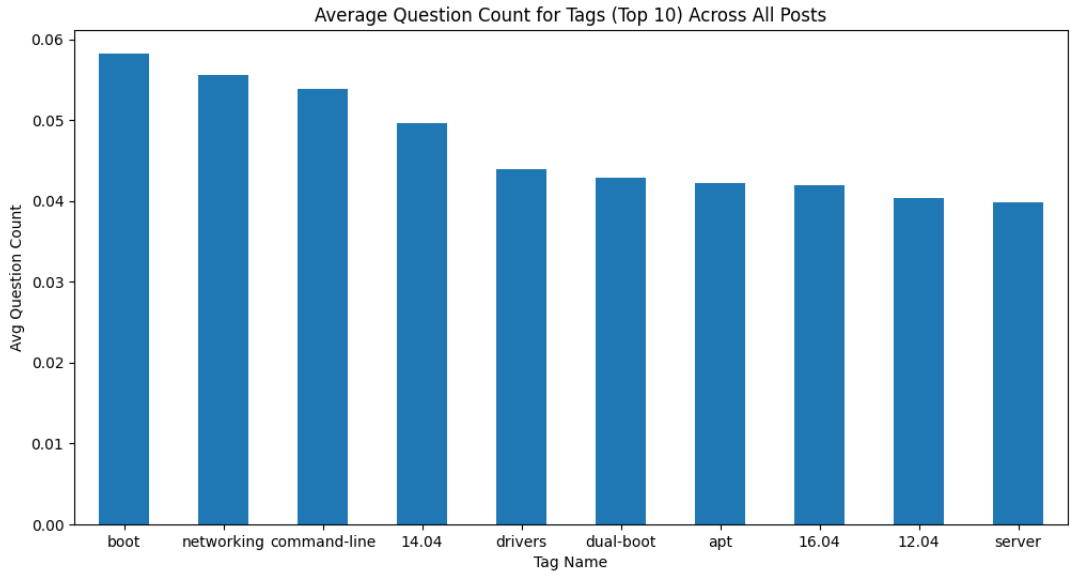
Plot 1. A five-number summary (minimum, first quartile, median, second quartile, maximum) of the reputation of all users.

As the reputation is added for each site. The code iterates through each site for each users to get reputation statistics and then reports the five-number summary on the console and saves the plot as "q4_1_plot.png".



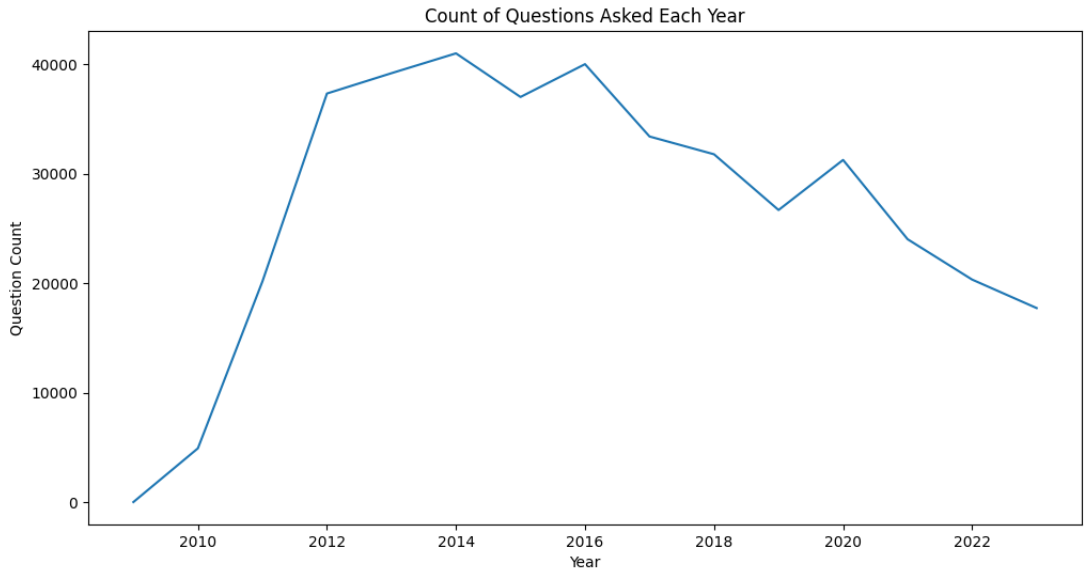
Plot 2. For the top 10 most used tags, the average number of questions for each tag as a bar chart.

The code fetches tags for questions (post type = 1) then reports the count of top 10 tags on the console and saves the plot as "q4_2_plot.png".



Plot 3. Number of questions asked each year as a time series plot.

The code fetches creation date for questions (post type = 1) then reports the count of questions by each year on the console and saves the plot as "q4_3_plot.png".



Note: Instructions to run the code are provided in the README.md file in the root folder.