# ASSIGNMENT - 3

**This Assignment has two parts,**
**total points: Max 25 points,[ with Bonus: 30 points ]**

## Part 1 (Do the following using Spark): 4 points

**Note-1**: Please note that you need to skip the "stop" words such as 'the' – please see the list of stop words here: **https://gist.github.com/sebleier/554280**

**Note-2**: Make sure the text in the input datasets is converted to lowercase. (If the word is in upper case convert it into lower case)

1. Determine 100 most frequent/repeated words in the 16GB dataset.
2. Determine 100 most frequent/repeated words in the 16GB dataset considering only the words having more than 6 characters.

Dataset for Part 1:
https://drive.google.com/file/d/1kaVM15rD9_O9HsvzrUKkIZ4R6ETUAdo4/view?usp=share_link

## Part 2: (total points for Part-2 -> 21 points)

This part has 2 sections.

### Section A: 6 points

1. Solving and getting familiar with log analytics.
2. Follow, understand and implement log analytics from the following article - "https://towardsdatascience.com/scalable-log-analytics-with-apache-spark-a-comprehensive-case-study-2be3eb3be977". (You will have to submit the implemented code from here too)
3. Along with all the EDA (Exploratory data analysis) mentioned in the article implement '3.1' and '3.2':
   - 3.1. Generate the output presenting the endpoint that received the highest number of invocations on a specific day of the week, along with the corresponding count of invocations. (Summarize this for the entire dataset)
     - 3.1.1. Output:
       Day in a week: Day in a week
       Endpoint: endpoint which was invoked the most.
       Count: Number of times it's been invoked
     - 3.1.2. Eg:

| Day in a week | endpoint | count |
|---|---|---|

| | | |
|---|---|---|
| Monday | /images/NASA-logosmall.gif | 200 |

The above is just an example of output format. But the values for that year would be different.

3.2. Total number of 404 status codes generated in each day of a week(For Monday, Tuesday,..... Sunday) for the entire dataset.

## Section B: 15 points

1. In this section, we will be working with a combination of Big Data technologies, including Kafka (producer and consumer), Spark Streaming, Parquet files, and HDFS File System.
2. The objective is to process and analyze log files by establishing a flow that begins with a Kafka producer, followed by a Kafka consumer that performs transformations using Spark. The transformed data will then be converted into Parquet file format and stored in the HDFS.
    a. Flow:
        i. Log File -> Kafka(Producer) -> Kafka(Consumer which performs transformation using Spark) -> Create Parquet files ->Store the Parquet file in HDFS
3. To initiate the process, we need to feed the data, which is an input file, into the Kafka producer. This can be achieved through various approaches, one such approach is injecting the data line by line(Feel free to come up with other ideas). This producer is responsible for publishing the data to a specific topic to which it is subscribed.
4. Once the data is injected, the Kafka consumer, which is also subscribed to the same topic, will periodically check for new data. In this case, it will check for available data every 10 seconds.
5. Once the data is received by the consumer, we will utilize Spark to perform transformations and conduct exploratory data analysis (EDA), employing techniques similar to those discussed in the article.
6. Finally, the results(eg: dataframes or RDD's from final transformation) obtained from each analysis will be converted into the Parquet file format, and then these Parquet files will be stored in the HDFS.

## Dataset to use in Section B:

These datasets have the similar structural information as that of the reference NASA dataset in the article(Section A) but the data values in the below dataset are different.
Dataset Link

- This folder contains a small dataset(for testing purpose) and a large dataset. (Access the dataset on May 21st 2023, in the noon)
    - https://drive.google.com/drive/folders/17D0wSo5qGNr8XA5753vF2BxsCjt-kPSm?usp=sharing

## Bonus Points (Section-C) – It is Optional: 5 points

1. The bonus task requires the development of a clustering algorithm capable of grouping requests based on several factors, including the host that invoked it, the time at which the endpoint was accessed, the status code received, and the data size of the returned information.
2. Be creative in coming up with what information you want to use for clustering.
3. While a straightforward approach would involve filtering the data based on these criteria, it would result in numerous small clusters. The objective is to minimize the number of clusters while ensuring that each cluster contains similar information.
4. Use Kafka Streaming for batching the information to transfer the data from producer to consumer.
5. For starters, one approach to clustering the streaming messages is to create clusters during the initial two streaming windows(or two batches). Then subsequently, messages from subsequent windows(batches) can be mapped to these pre-established clusters based on their similarity.
6. Various methods can be employed to determine the similarities, feel free to use anything you see fit.

## Deliverables:

1. Source code including a readme file to run your code, Output screenshots and also a report (PDF format) should be included in a SINGLE tar/zip file uploaded on Camino. Naming convention for the zip/tar file is "LastName_FirstName.zip/tar"
2. PDF Report - Containing information on how you tackled the problem and its analysis.