Task 4 (25 points) Structured Streaming Processing

In Task 4, you will analyze the **NASA Logs dataset** using **Structured Streaming Processing**. Please refer to the **Week 8 lab** to gain an understanding of the NASA Logs dataset and the streaming process.

Please note that the example tables/pictures provided in this task are based on my results. Your results may differ depending on how you process and display the data. When a screenshot is required, in addition to the results, you must also include your running timestamp in your screenshot. A running timestamp is the recorded time (printed at the beginning of each line in the logs) during the execution of a Spark job; you can capture the running timestamp in the logs. The example tables/pictures provided in each question illustrate the expected format (including the running timestamp and your results) for your screenshot.

1) **(2 points)** In Week 8 lab, we retrieved the host and port values from environment variables to load the NASA logs dataset from a socket. Now, you need to explicitly specify the host value as **"stream-emulator.data-science-tools.svc.cluster.local"** and the port as **5551** to load the dataset. Please locate the **question 1** you need to complete in **task4_starter_kit.py**. Create a query to display the dataset in the **console** using **append** mode and ensure that **fields are not truncated**. You will need to provide a screenshot of the output for batch 0, 1 and 2 in your report. For example.

```
2024-10-25 13:34:24,454 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@1700f
-------------------------------------------
Batch: 0
-------------------------------------------
+-----+---------+
|value|timestamp|
+-----+---------+
+-----+---------+

-------------------------------------------
Batch: 1
-------------------------------------------
+-----------------------------------------------------------------------+-----------------------+
|value                                                                  |timestamp              |
+-----------------------------------------------------------------------+-----------------------+
|Server is working .,host,time,method,url,response,bytes                |2024-10-25 13:34:26.806|
|1,***.novo.dk,805465031,GET,/images/ksclogo-medium.gif,200,5866        |2024-10-25 13:34:28.808|
|0,***.novo.dk,805465029,GET,/ksc.html,200,7067                         |2024-10-25 13:34:27.807|
+-----------------------------------------------------------------------+-----------------------+

-------------------------------------------
Batch: 2
-------------------------------------------
+-----------------------------------------------------------------------+-----------------------+
|value                                                                  |timestamp              |
+-----------------------------------------------------------------------+-----------------------+
|2,***.novo.dk,805465051,GET,/images/MOSAIC-logosmall.gif,200,363       |2024-10-25 13:34:29.81 |
+-----------------------------------------------------------------------+-----------------------+
```

2) **(2 points)** Define a watermark on the **timestamp** column with a delay of **3** seconds. A **watermark** is a mechanism to handle late-arriving data by specifying a time threshold, after which delayed data is considered too late to be processed and is discarded. You need to explain the purpose of defining a watermark in your

report. Please locate the **question 2** you need to complete in **task4_starter_kit.py**.

3) **(3 points)** Create a query to display the dataset in the **console** using **append** mode and ensure that **fields are not truncated**. You will need to provide a screenshot of the output for batch 19 and batch 20 in your report. For example.

```
Batch: 19
-------------------------------------------
+------------------------------------------------------+-----------------------+---+------------+---------+------+-------------------+------------+-----+
|logs                                                  |timestamp              |idx|hostname    |time     |method|url                |responsecode|bytes|
+------------------------------------------------------+-----------------------+---+------------+---------+------+-------------------+------------+-----+
|18,***.novo.dk,807951832,GET,/shuttle/countdown/,200,4673|2024-10-15 14:46:31.727|18 |***.novo.dk|807951832|GET   |/shuttle/countdown/|200         |4673 |
+------------------------------------------------------+-----------------------+---+------------+---------+------+-------------------+------------+-----+

-------------------------------------------
Batch: 20
-------------------------------------------
+--------------------------------------------------------------------------+-----------------------+---+------------+---------+------+-------------------------------------+------------+-----+
|logs                                                                      |timestamp              |idx|hostname    |time     |method|url                                  |responsecode|bytes|
+--------------------------------------------------------------------------+-----------------------+---+------------+---------+------+-------------------------------------+------------+-----+
|19,***.novo.dk,807951848,GET,/shuttle/missions/sts-69/count69.gif,200,46053|2024-10-15 14:46:32.728|19 |***.novo.dk|807951848|GET   |/shuttle/missions/sts-69/count69.gif|200         |46053|
+--------------------------------------------------------------------------+-----------------------+---+------------+---------+------+-------------------------------------+------------+-----+
```

Note that the above figures/values may not represent the actual result.

4) **(6 points) (Window)** Create a query to count the number of lines in the **"URL"** field that contains **"gif"** in each time window, name the result column as **gif_count**, and write the output to the console in **update** mode, ensuring that **fields are not truncated**. Set the **windowDuration** to **'60 seconds'** and the **slideDuration** to **'30 seconds'**. You must include a screenshot of the output for **batches 1, 2, and 3** in your report. For example.

```
Batch: 1
-------------------------------------------
+------------------------------------------+---------+
|window                                    |gif_count|
+------------------------------------------+---------+
|[2024-10-15 15:07:00, 2024-10-15 15:08:00]|9        |
|[2024-10-15 15:07:30, 2024-10-15 15:08:30]|2        |
|[2024-10-15 15:06:30, 2024-10-15 15:07:30]|7        |
+------------------------------------------+---------+

-------------------------------------------
Batch: 2
-------------------------------------------
+------------------------------------------+---------+
|window                                    |gif_count|
+------------------------------------------+---------+
|[2024-10-15 15:07:00, 2024-10-15 15:08:00]|14       |
|[2024-10-15 15:07:30, 2024-10-15 15:08:30]|7        |
+------------------------------------------+---------+
```

Note that the above figures/values may not represent the actual result.

5) **(6 points) (Window)** group the dataset by **hostname** and aggregate the bytes transferred for each host, name the new column as **"total_bytes"** and sort the results by **total_bytes** in descending order**.** Write the output to the **console** in

Complete mode, ensuring that **fields are not truncated**. You must include a screenshot of the output for **batches 4** in your report. For example.

window	hostname	total_bytes
[2024-10-22 19:54:00, 2024-10-22 19:55:00]	***.novo.dk	50000
[2024-10-22 19:54:30, 2024-10-22 19:55:30]	***.novo.dk	40000
[2024-10-22 19:54:30, 2024-10-22 19:55:30]	007.thegap.com	30000
…….		

Note that the above figures/values may not represent the actual result.

6) **(6 points) (trigger)** Create a query that filters the incoming log data to include only GET requests with a response code of 200 (indicating successful requests). Group the data by hostname and count the number of successful GET requests for each host. The query should trigger every 10 seconds, outputting the results to the console in complete mode, ensuring that **fields are not truncated**. You must include a screenshot of the output for **batches 6** in your report. You do not need to include the timestamp for this screenshot. For example.

window	Correct_count
007.thegap.com	10
001.msy4.communique.net	8
***.novo.dk	15
…….	…….

Note that the above figures/values may not represent the actual result.