Week 5 - bonus homework

Description

We implemented a simplified version for the sessions generation. Let's try to go even further and do the same but with Apache Spark Structured Streaming! Unfortunately, this implementation can be implemented only in Scala/Java Spark because of PySpark's limitations.

But no worries, I also prepared 2 open questions where 1 is about the code you wrote previously in batch! Please take a look at both of them in the 2 points below.

Happy coding 🍀

Part 1 - coding exercise

We'll still use the output schema defined in the main exercise but this time we'll do that with Scala/Java Spark:

Implement a consumer that will:

- Continuously read data from *valid_data* topic
- Generate sessions for every visit with `mapGroupsWithState`
    - in the mapping function, accumulate all logs seen so far but only with the attributes that are useful for the output generation
    - configure the timeout with one of `GroupState's set*` methods
- Write the generated sessions into a topic called sessions in Apache Kafka. The topic is already created if you use our usual Docker image.
    - before writing, convert the terminated sessions (see next point's rules) into the format defined in the main-homework exercise.

Below you can find some business rules for the sessions implementation:

- use visit_id field to group one session's events
- you should terminate a session if you don't see any activity for given visit_id 2 minutes after the most recent received event, eg. if the most recent event for given visit is 9:50, I should terminate the session at 9:52
  Hint#1: you can do that with `org.apache.spark.sql.streaming.GroupState#setTimeoutTimestamp`
  Hint#2: `setTimeoutTimestap` requires a watermark definition so you will also have to use `.withWatermark(...)` we saw in the last week's exercise

Expected outcome

- create a new branch called week5_sessionization_streaming_app and push the corresponding homework code there
- create new Pull Request and send me the link

Setup

1. Start a new IntelliJ Scala/Java project project. You will find the examples in **Week 0 >> PyCharm and IntelliJ setup** examples lesson.
   Add required dependencies, depending on the technology you use (Apache Spark, Apache Kafka, other).
2. Start Apache Kafka broker and data generator:
   - Download or clone https://github.com/bartosz25/course-data-homework
   - Execute:
     ```
     git pull origin master
     cd docker-images/week5/bonus
     docker-compose down --volumes
     docker-compose up
     ```

     If you want to reduce the volume of generated data, you can change `all_users` property in the `configuration.yaml` file.
3. Write your sessionization logic.

   If you want to keep only the Kafka broker up and running:
   - stop launched images
   - edit `docker-compose.yaml` and comment all lines for `data_generator` entry
   - start Kafka broker again with `docker-compose up` - do not call `docker-compose down --volumes` because it will remove any data already produced

Homework tips

- to check what data is arriving to *valid_data* topic, you can start new console consumers

  -
    ```
    > docker exec -ti bonus_kafka_1 bash
    > kafka-console-consumer.sh --bootstrap-server
    localhost:9092 --topic valid_data
    ```
- to facilitate the implementation of your code you can connect to the *valid_data* consumer (above point), save some records in a file and later write your business logic based on that, without the Kafka broker running

    - you can also create a new topic from the Kafka CLI, like *valid_data_dev* and ingest there the records you put aside or that you created for tests - during the development you will read from *valid_data_dev* and during the validation, from *valid_data*, on the really generated data
- to check what data is arriving to *sessions* topic, you can start new console consumers

    -
      ```
      > docker exec -ti bonus_kafka_1 bash
      > kafka-console-consumer.sh --bootstrap-server
      localhost:9092 --topic sessions
      ```

Part 2 - open questions

1. Let's suppose that you have just joined a team and you got a task to fix a sessionization application. The application is running on production, it doesn't have unit tests, the logs don't allow you to identify the errors, and the errors are located in the business logic. Your debugging process cannot impact the production application, so you cannot shut it down and debug. Please explain what will be your debugging process? What will be your first step, what you will do next? Explain why you chose these steps.
2. In the previous exercise we generated the sessions only for 1 partition-window. Remember, we're synchronizing data from Kafka to our file system and put it into hourly partitions based on the event_time field.

The problem with our code is that we can generate only partially valid sessions. For example, if one visit starts at 9:30 and terminates at 10:40, we'll generate it into 2 files, one generated in the batch for 9 o'clock, and one in the batch for 10 o'clock.

Please propose a solution to generate the session only in the last processing window (10 o'clock for our example). Choose the format you're the most comfortable with, it can be a graphical representation, textual explanation, pseudo-code … .

Expected outcome

- add your answers to the files in week5_questions branch