Case Study

Data Engineering

Problem

On the Loco platform, we have a short form content offering known as **Clips**. You are given clickstream events data for Watch, Like and Unlike events related to the clips in JSON lines format. You are expected to write an ETL job that ingests the raw data and processes it into a meaningful aggregated form that can be used by other teams and jobs down the stream.

Data

Please familiarize yourself with the schema for each of the events in the files named `reel_watched_schema.json`, `reel_liked_schema.json` and `reel_unliked_schema.json`.

Events:

- **reel_watched**: Fired when a user watches a clip. It contains data such as duration for which the clip was watched, how many times it was replayed, etc. Since a clip keeps on looping, a user can have **watch** events with replay_count > 1, or he/she can scroll to other clips and come back to it, causing a new **reel_watched** event.
- **reel_liked**: Fired when a user likes a clip.
- **reel_unliked**: Fired when a user unlikes a clip.

Note: A user can have multiple watch/like/unlike events for the same clip.

Output: Your task is to help the other teams in figuring out whether a user is interested in a clip or not based on a number of factors:

1. If the user has disliked a clip, then he/she is definitely not interested in it.
2. If the user has liked a clip, then he/she is interested.
3. If the user hasn't performed any like/unlike action on a clip, then calculate the watched ratio of the user for that clip. If the watched ratio is more than 0.6, then mark the user as interested else mark him/her as not interested.

The output file should have aggregated data at user level with these fields for each entry:

1. **user_uid (string)**
2. **reel_uid (string)**
3. **liked (boolean)**
4. **unliked (boolean)**
5. **watched_ratio (decimal)**
6. **is_interested (boolean)**


```
{
  "user_uid": "ABCD1234Z",
  "reel_uid": "007bb793-d88e-4370-b7a2-a31fec0111db",
  "liked": "false",
  "unliked": "false",
  "watched_ratio": "1.1578947368421053",
  "is_interested": "true"
}
```


Things to keep in mind:

1. Take care of null attributes. You need to ignore the events where important fields are null while doing calculations.
2. The attribute **reel_watch_duration** can mean different things across different platform and app versions. Here are 3 cases to be handled:
    1. For **platform** == **"android"** and the **app_version** >= **"2.3.0"**, the frontend into account the replay count and reel duration. This means that the **reel_watch_duration** is the actual total watch duration.
    2. For **platform** == **"android"** and the **app_version** < **"2.3.0"**, the **reel_watch_duration** resets to 0 when a clip loops. This means that if **replay_count** is greater than 0, the total **reel_watch_duration** will be (**reel_duration** X **replay_count**) + **reel_watch_duration**.
    3. For **platform** == **"ios"**, only case **#1** is there. That is, the **reel_watch_duration** is the total watch duration.
3. Users can **like** or **unlike** any clip multiple times using the same ui button in the frontend. So assuming that the data given to you is exhaustive, you need to take timestamp into consideration while figuring out whether the final action was a like or unlike for a clip by a user, and only consider the final action.

Deliverables

- An ETL script that reads the input data from the files and writes the required output to a file.
- The aggregated output (in any format).

Must have

- Script for the ETL job
- A description of which cloud tools you would use for productionisation of this ETL job

Good to have

- Solution written using PySpark (As we use PySpark for our ETL workloads in production)
- Conversion to Apache Parquet format (Be ready for discussions around the parquet format and different configurations and tradeoffs around it)

Brownie Points

- Apache Airflow based workflow consisting of 2 jobs:
    - Another job that takes the aggregated output and writes it to a rdbms
    - Airflow code should have proper error handling for failures
- Unit tests for the logic

What we are looking for

We are looking for a good understanding of the fundamentals around data. The solutions to this assignment should be clean and well abstracted. We are looking for you to make architectural decisions on how to structure things, and be able to describe/justify those decisions and why you made them. Experience with AWS is a plus.

Our Tech Stack

As part of the data team at Loco, we use these tools in our day to day work:

- AWS Glue
- AWS Athena
- Python
- PySpark
- AWS EMR
- SQL
- Apache Airflow
- AWS Lambda