Data Engineering: Final Project

1. Knowledge required for this project

- Kafka
- Spark DataFrame transformation
- Spark Structured Streaming
- Impala
- Hdfs
- Kudu
- YARN

2. Create a Kafka Topic

Create a kafka topic named "rsvp_your_id" with 2 partitions and 2 replications.

3. Create Kafka Data Source

Use meetup RSVP event as the Kafka data source. Ingest meetup_rsvp event to the Kafka topic, use kafka-console-consumer to verify it.

Meetup provides a streaming API for retrieving the RSVP event.
https://stream.meetup.com/2/rsvps

Hint:
The easiest way is to use kafka-console-producer

Bonus:
Write a Kafka producer and a Kafka consumer using Scala, or Java, or Python to produce/consume events to/from the Kafka topic.

4. Write 5 Spark Structured Streaming Job

Write Spark Streaming jobs to process data from Kafka.

1. Save events in HDFS in text(json) format. Use "kafka" source and "file" sink. Set outputMode to "append".

2. Save events in HDFS in parquet format **with schema**. Use "kafka" source and "file" sink. Set outputMode to "append".

3. Show how many events are received, display in a 2-minute tumbling window. Show result at 1-minute interval. Use "kafka" source and "console" sink. Set outputMode to "complete". Here is a sample output.

```
+------------------------------------------+-----+
|window                                    |count|
+------------------------------------------+-----+
|[2020-10-07 23:30:00, 2020-10-07 23:32:00]|29   |
|[2020-10-07 23:32:00, 2020-10-07 23:34:00]|121  |
|[2020-10-07 23:34:00, 2020-10-07 23:36:00]|107  |
|[2020-10-07 23:36:00, 2020-10-07 23:38:00]|40   |
+------------------------------------------+-----+
```

4. Show how many events are received for each country, display it in a sliding window (set windowDuration to 3 minutes and slideDuration to 1 minutes). Show result at 1-minute interval. Use "kafka" source and "console" sink. Set outputMode to "complete". Here is a sample output.

```
+------------------------------------------+-------------+-----+
|window                                    |group_country|count|
+------------------------------------------+-------------+-----+
|[2020-10-09 01:07:00, 2020-10-09 01:10:00]|us           |8    |
|[2020-10-09 01:07:00, 2020-10-09 01:10:00]|sg           |1    |
|[2020-10-09 01:07:00, 2020-10-09 01:10:00]|gb           |2    |
|[2020-10-09 01:07:00, 2020-10-09 01:10:00]|au           |4    |
|[2020-10-09 01:07:00, 2020-10-09 01:10:00]|it           |2    |
|[2020-10-09 01:07:00, 2020-10-09 01:10:00]|za           |2    |
|[2020-10-09 01:07:00, 2020-10-09 01:10:00]|id           |1    |
|[2020-10-09 01:07:00, 2020-10-09 01:10:00]|in           |2    |
|[2020-10-09 01:08:00, 2020-10-09 01:11:00]|au           |4    |
|[2020-10-09 01:08:00, 2020-10-09 01:11:00]|za           |2    |
|[2020-10-09 01:08:00, 2020-10-09 01:11:00]|it           |2    |
|[2020-10-09 01:08:00, 2020-10-09 01:11:00]|id           |1    |
|[2020-10-09 01:08:00, 2020-10-09 01:11:00]|us           |8    |
|[2020-10-09 01:08:00, 2020-10-09 01:11:00]|in           |2    |
|[2020-10-09 01:08:00, 2020-10-09 01:11:00]|sg           |1    |
|[2020-10-09 01:08:00, 2020-10-09 01:11:00]|gb           |2    |
|[2020-10-09 01:09:00, 2020-10-09 01:12:00]|sg           |1    |
|[2020-10-09 01:09:00, 2020-10-09 01:12:00]|it           |2    |
|[2020-10-09 01:09:00, 2020-10-09 01:12:00]|id           |1    |
|[2020-10-09 01:09:00, 2020-10-09 01:12:00]|us           |8    |
+------------------------------------------+-------------+-----+
```

5. Use impala to create a KUDU table. Do dataframe transformation to extract information and write to the KUDU table. Use "kafka" source and "kudu" sink.

```
create table if not exists rsvp_db.rsvp_kudu_<your-id>
(
    rsvp_id         bigint primary key,
    member_id       bigint,
    member_name     string,
    group_id        bigint,
    group_name      string,
    group_city      string,
    group_country   string,
    event_name      string,
    event_time      bigint
)
PARTITION BY HASH PARTITIONS 2
STORED AS KUDU;
```

Include following kudu-specific code in writeStream:

```
.format("kudu")
.option("kudu.master", "the actual kudu master")
.option("kudu.table", "impala::rsvp_db.rsvp_kudu_<your-id>")
.option("kudu.operation", "upsert")
```

Use impala query to verify that the streaming job is inserting data correctly.

6. Use YARN web console to monitor the job.