ID2221 Data Intensive Computing - Review Questions 3 - Group 19

by Artem Sliusarenko, Mihailo Cvetkovic and Eugen Lucchiari Hartz

1.) Briefly compare the DataFrame and DataSet in SparkSQL and via one example show when it is beneficial to use DataSet instead of DataFrame.

DataFrames and Datasets in SparkSQL are high-level abstractions for working with structured data. DataFrames are similar to tables in a relational database. They offer an organized collection of rows and named columns, but they do not provide compile-time type safety. Datasets on the other hand, extend DataFrames by adding type safety and by allowing errors to be caught at compile time. They also provide advanced encoders for efficient serialization and deserialization of data.

When working with complex data transformations where type safety is crucial, such as performing operations on a structured dataset with specific schema requirements, it is beneficial to use a Dataset. For example, if you are performing type-specific transformations on a collection of custom objects, a Dataset will allow you to catch type mismatches at compile time, reducing runtime errors and improving code safety. DataFrames do not provide compile-time type safety which means that they only throw an error after executing the code but not during compilation.

2.) What will be the result of running the following code on the table people.json, shown below? Explain how each value in the final table is calculated.

```
val people = spark.read.format("json").load("people.json")
val windowSpec = Window.rowsBetween(-1, 1)
val avgAge = avg(col("age")).over(windowSpec)
people.select(col("name"), col("age"), avgAge.alias("avg_age")).show
```

people.json

```
{"name":"Michael", "age":15, "id":12}
{"name":"Andy", "age":30, "id":15}
{"name":"Justin", "age":19, "id":20}
{"name":"Andy", "age":12, "id":15}
{"name":"Jim", "age":19, "id":20}
{"name":"Andy", "age":12, "id":10}
```

In PySpark the **Window.rowsBetween(start, end)** method defines a window frame for window function. The frame specifies a range of rows to consider when performing an operation.

The expression **Window.rowsBetween(-1, 1)** defines **-1** as the start row(inclusive) and **1** as the end row(inclusive) of the frame to be considered. Basically it means the frame includes the current row, 1 row before current row, and 1 row after current row.

The window frame moves processing from row to row and it considers the defined window frame while performing an operation on the current row.

The result of the operation above is as follows:

Michael	15	22,5
Andy	30	21,3
Justin	19	20,3
Andy	12	16,7
Jim	19	14,3
Andy	12	15,5

Each value in the final table is evaluated by calculating an average age of the current row(age), rows(age) above(if any) and row(age) below(if any).

Ex: Michael's(row 1) age is calculated by summing ages (15 + 30(row below)) /2 = 22,5
Andy(row 2)is calculated by summing (30 + 15(row above) + 19(row below)) / 3 = 21,3

3.) What is the main difference between the log-based broker systems (such as Kafka), and the other broker systems.

The main difference between log-based broker systems like Kafka and traditional message broker systems is how they handle message storage and consumption. In a log-based broker, all events are stored in a sequential log and messages are retained even after being consumed. Producers append messages to the log and consumers read them from a specified offset which allows them to replay and track the history of events.

In contrast, traditional message brokers typically delete messages once they have been consumed. They focus on delivering messages between producers and consumers and are preferred in scenarios where retaining the entire history of messages is not necessary. This makes them suitable for use cases with varied messaging patterns where message retention isn't a priority.

4.) Compare the windowing by processing time and the windowing by event time, and explain how watermarks help streaming processing systems to deal with late events.

In windowing by processing time events are grouped based on the time they are processed by the system. If you set a 1-minute processing-time window, the system will create a new window every minute. Any event processed within that time will be part of that window even if the event occurred much later or earlier. An advantage of this is its simplicity as it is very

easy to implement since it relies on the system's own clock. One drawback may be that events are not grouped correctly.

In windowing by event time events are grouped by actual time the event occurs. In a 1 minute time window, any event within that time is grouped together regardless of when the event is processed. An advantage is that it is more accurate as it reflects on actual real-world time that the event occurs. A drawback is that handling late-arriving data becomes a challenge.

Watermarks are the mechanism that helps streaming processing systems deal with late events in event-time windowing. A watermark is a threshold that tells the system how far behind the current event time it should expect to see late events. Watermarks help deal with late events by reducing inaccuracies as it waits for late events. They also help with graceful handlings, i.e Once the watermark passes, the system can finalize the window and treat any further late events as truly late and act according to the selected policy.