Group: Group Therapy
Casper Kristiansson
Nicole Wijkman

RQ 3: Data-Intensive Computing

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 are both part of the Spark SQL model as high-level abstractions for working with structured (and unstructured?) data. They are both distributed table-like collections with well-defined rows and columns

Just like Resilient Distributed Datasets (RDDs), DataFrames are distributed collections of data. DataFrames are however organized into a series of rows and a number of named columns, similar to a table in a relational database. DataSets are an even newer addition to Spark and can be seen as an extension to DataFrames, designed to combine the strong typing of RDDs with the optimization benefits of DataFrames.

DataFrames does not provide compile-time type safety, meaning it does not throw an error during the compile time, only when the code is being executed. This differs from DataSets, which throws the error during the compile time. DataSets also provides advanced encoders, which can provide on-demand access to individual attributes. This feature is not found in DataFrames.

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}
```

The following script performs 4 different actions:

1. Reads the people.json file into a dataframe called people
2. Define a window of columns with the indexes of -1 to 1

3. A function is applied using the window spec on the column age to compute an average age of the rows -1 to 1 relative to the current column
4. The dataframe is transformed with three columns, name, age, and the average age computation using the select command.

Overall what this function does is that for each row it will compute an average age for the rows relative to the selective row with index -1 to 1. The final table is:

name	**age**	**avg_age**
Michael	15	(15+30)/2=22.5
Andy	30	(15+30+19)/3=21.33
Justin	19	(30+19+12)/3=20.33
Andy	12	(19+12+19)/3=16.67
Jim	19	(12+19+12)/3=14.33
Andy	12	(19+12)/2=15.5

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

A log-based message broker will store all events in a sequential log, while a typical message broker deletes the message once it has been consumed. The log is an append-only sequence of records where a producer appends messages to the end, and consumers read the records from a specified offset. This allows consumers to keep track of the history of all the events that have been processed.

A traditional message broker focuses more on delivering messages between producers and consumers. A traditional message broker is, therefore, preferred in scenarios that may require varied messaging patterns. They can be suitable for use cases where retaining the entire history of messages is not crucial.

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

Stream processing is the type of event that needs to be tracked. These types of events are not predetermined meaning that they are often the types of events like website clicks or IoT sensor readings. When these types of event data should be processed there are different ways to process them. For improvement's sake, windowing has been introduced where the goal is to group events together and process rather than process them one by one.

Processing time using a window refers to the process of grouping the events when they arrive in the system. Because the delivery time of the event to the main system can from time to time have different delays. The second type of processing is doing it by **event time.** This type of processing is usually used when the order of the events is important which means that the events will be grouped by the actual recorded date of the event.

The role of watermarks play are crucial part in dealing with late events when using a window processing structure. A watermark is the process of declaring a time when no more stream of events should arrive earlier than it. If there are older events than the watermark time (late events) they need to be dealt with. In many cases, the events will either be disregarded or inserted into the desired stream (update the stream of events).