Spark Structured Streaming provides two types of stream processing: stateful and stateless

**Stateless Streaming:**

Stateless streaming treats each micro-batch of data independently and processes it without any reference to previous batches. This means that the output of a micro-batch is only based on the data in that batch. Examples of stateless operations include filtering, mapping, and aggregations that do not require state. Stateless operations are typically faster and consume less memory than stateful operations. Stateless streaming is also easy to scale horizontally since there is no need to keep track of state across multiple nodes.

**Stateful Streaming:**

Stateful streaming maintains state across multiple batches of data. This means that the output of a micro-batch depends not only on the data in that batch, but also on the state of the system accumulated from previous batches. Examples of stateful operations include windowed operations, which compute aggregates over a sliding window of data etc. Stateful operations are typically slower and consume more memory than stateless operations. Stateful streaming is also harder to scale horizontally since there is a need to keep track of state across multiple nodes.

*Choosing the Right Processing Mode:*

When choosing between stateful and stateless processing modes in Spark Streaming, you should consider the following factors:

* Memory Requirements:

Stateful operations require more memory than stateless operations since they need to maintain state across batches. If you have limited memory resources, stateless processing may be the better option.

* Processing Speed:

Stateless operations are typically faster than stateful operations since they do not require any memory of previous batches. If you need real-time processing, stateless processing may be the better option.

* Data Complexity:

Stateful operations are better suited for applications that require more complex processing, such as windowed computations or stateful aggregations. Stateless operations are better suited for simpler operations that do not require state.

* Scalability:

Stateless operations are easier to scale horizontally since there is no need to keep track of state across multiple nodes. Stateful operations require more complex coordination across nodes, which can make scaling more challenging.

* Fault Tolerance:

Stateful operations are more prone to failures since they require maintaining state across multiple batches. Spark Streaming provides fault-tolerance mechanisms to recover from node failures, but the recovery process can be slower and more complex for stateful operations.

In the Stateful Streaming section, one of the examples provided was windowed operations that compute aggregates over a sliding window of data. Primarily, there are 2 types of windows:

In *tumbling windows*, the time windows are non-overlapping. Useful in scenarios like what is the max trade\_value in the time period 10:45 to 11:00.

In *sliding windows*, the time windows are overlapping. Useful in scenarios like what is the max trade\_value in the last 15 minutes.