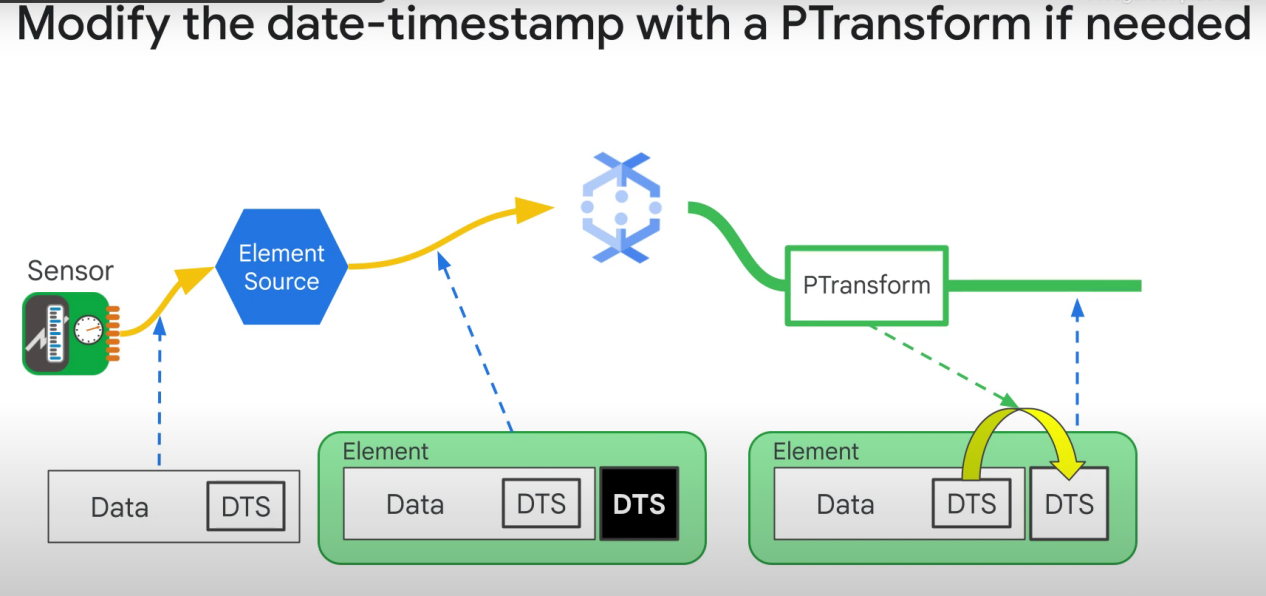
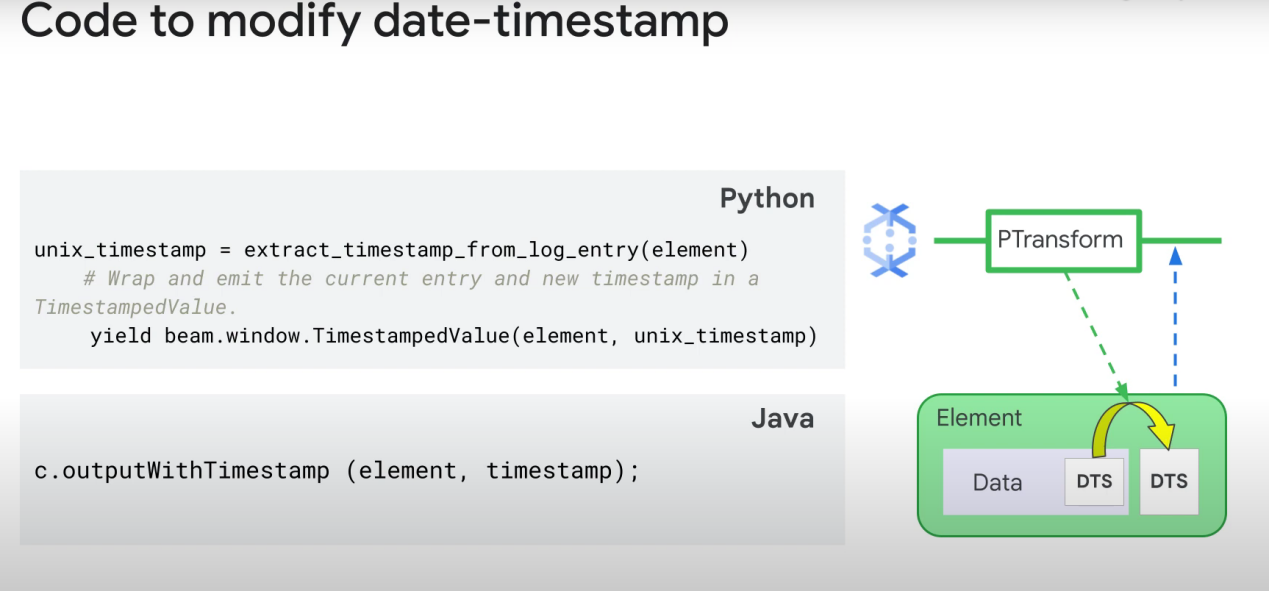
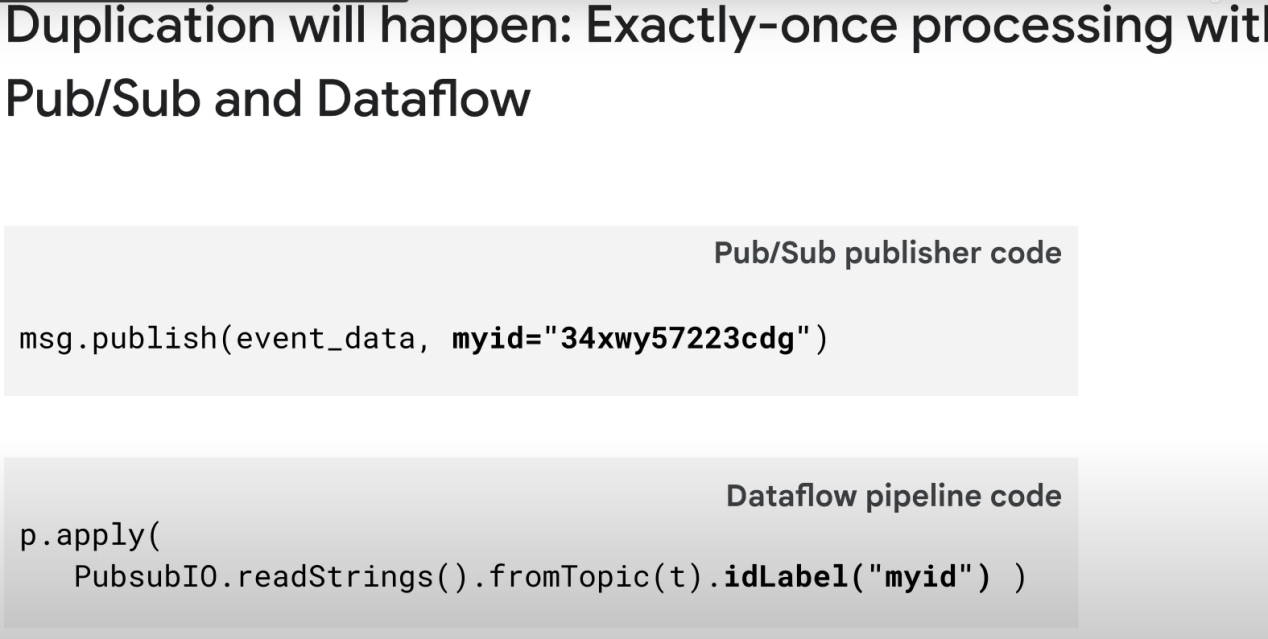


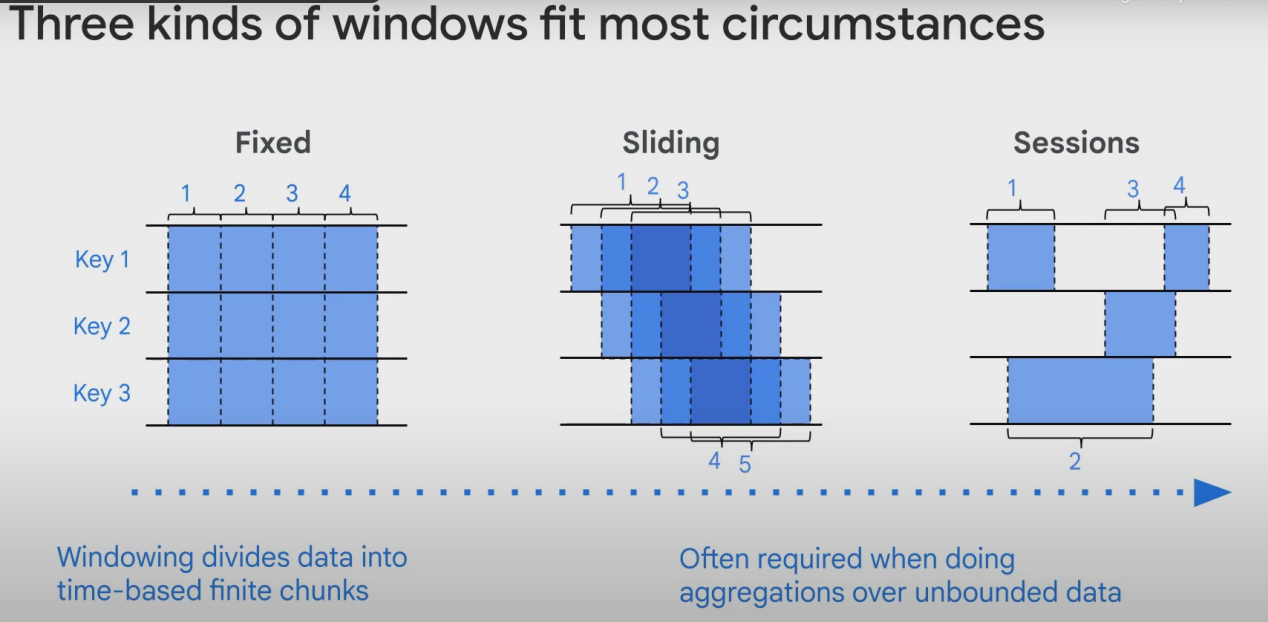
**By default, DTS (date time stamp) of each data is set to the time of entry of the data to the system, instead of the sensor’s capture time.** There is some latency between the time a sensor reads the data, and the time the publisher sends it. You can recalculate timestamps with Ptransforms if that latency is too big.





Note: If pub/sub IO is configured to use custom message IDs, dataflow will remove any duplicate automatically ==> solves the duplication of messages





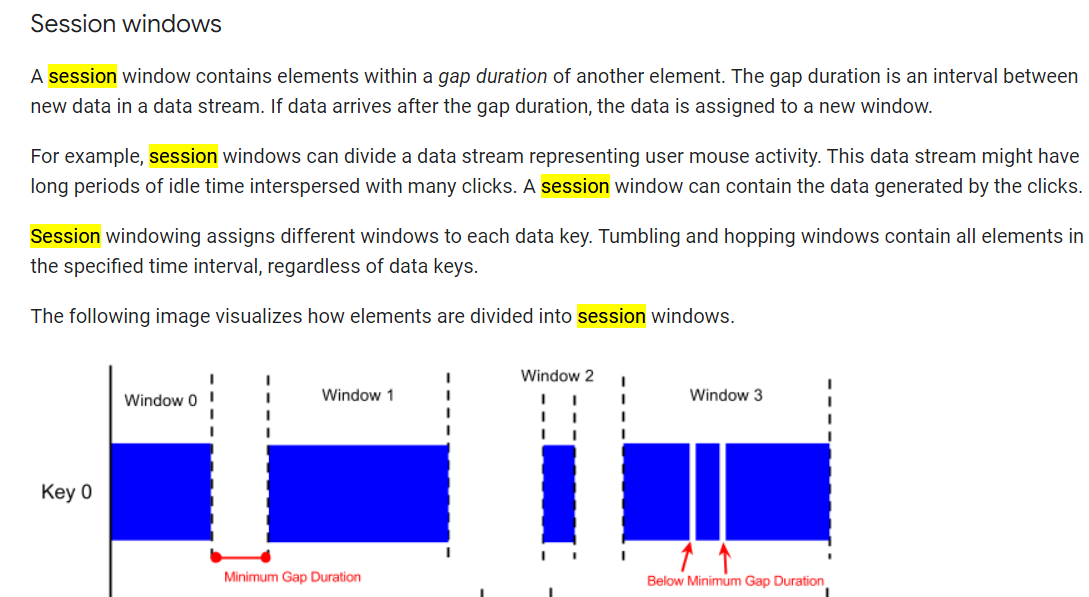
Examples:

Fixed: Just fixed... Consistent, non-overlaping intervals

Sliding: Generally used when we need computation/statistics. For instance, to calculate, every 5mins, the average and mean of 30min worth of data.

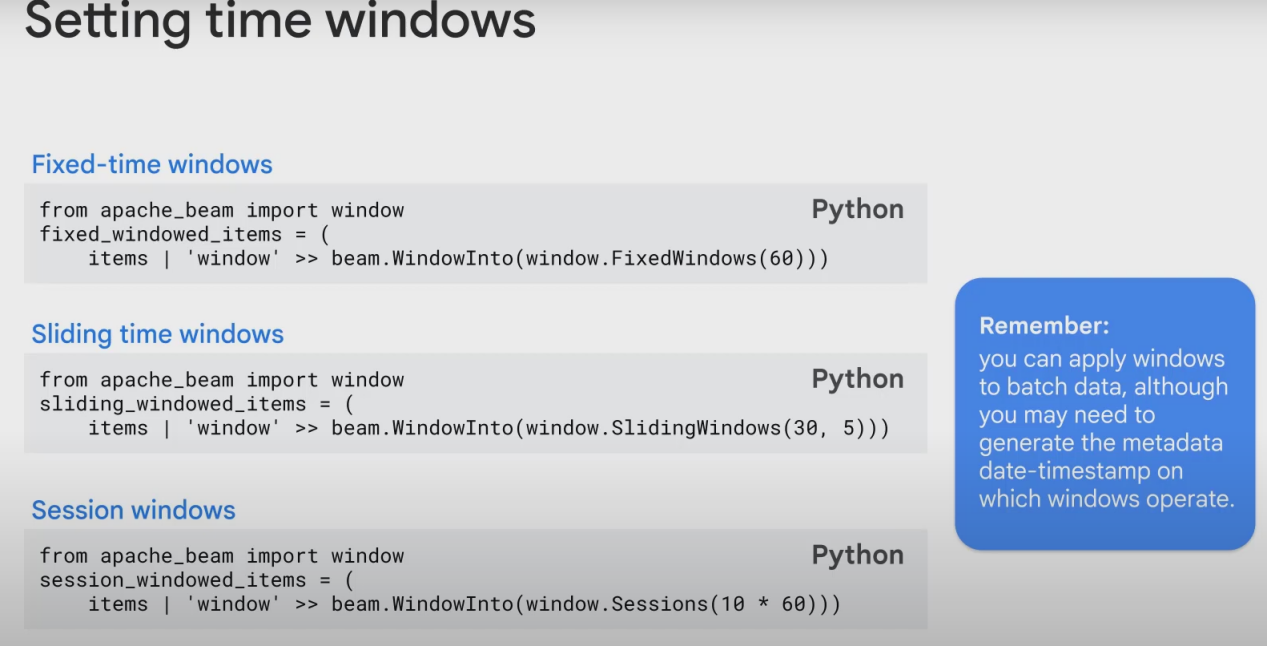
Sessions: The duration a user spends on a website can be considered a window. It usually has a «minimum» **gap duration**.

More on gap duration:

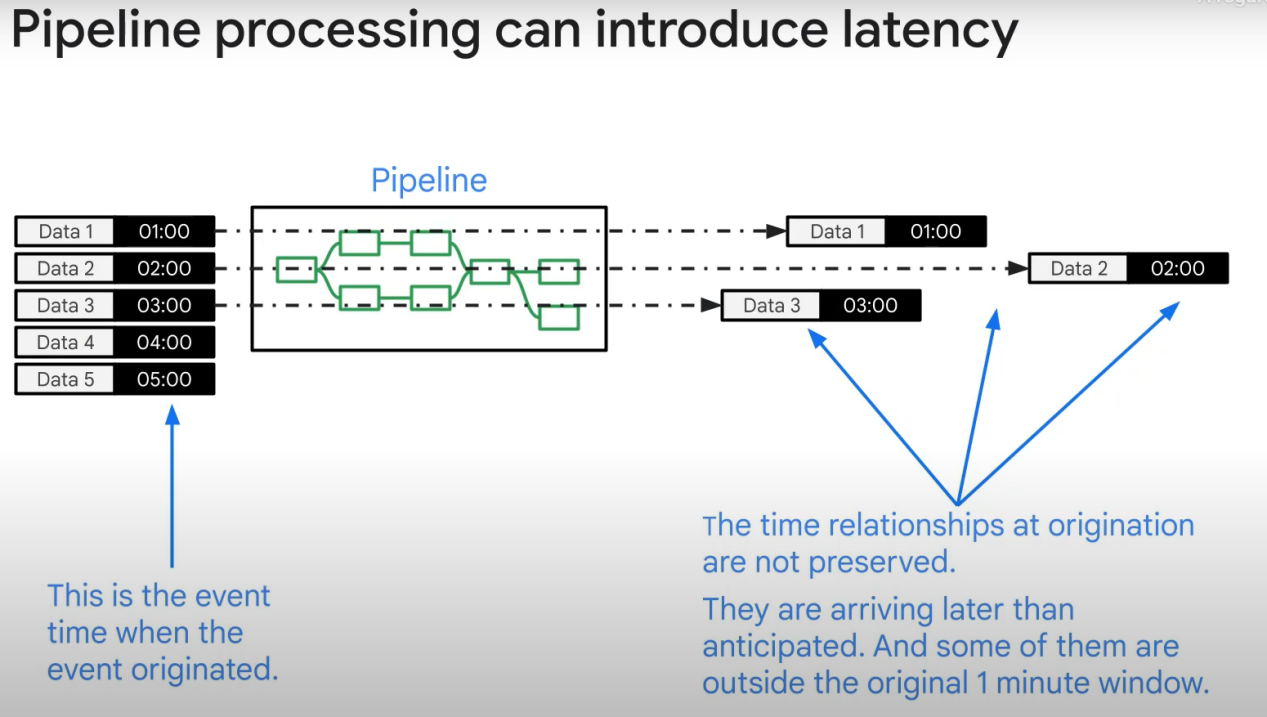




Any Key in the data can be used as a session key. It will have a timeout period, and it will flush the window at the end of that timeout.



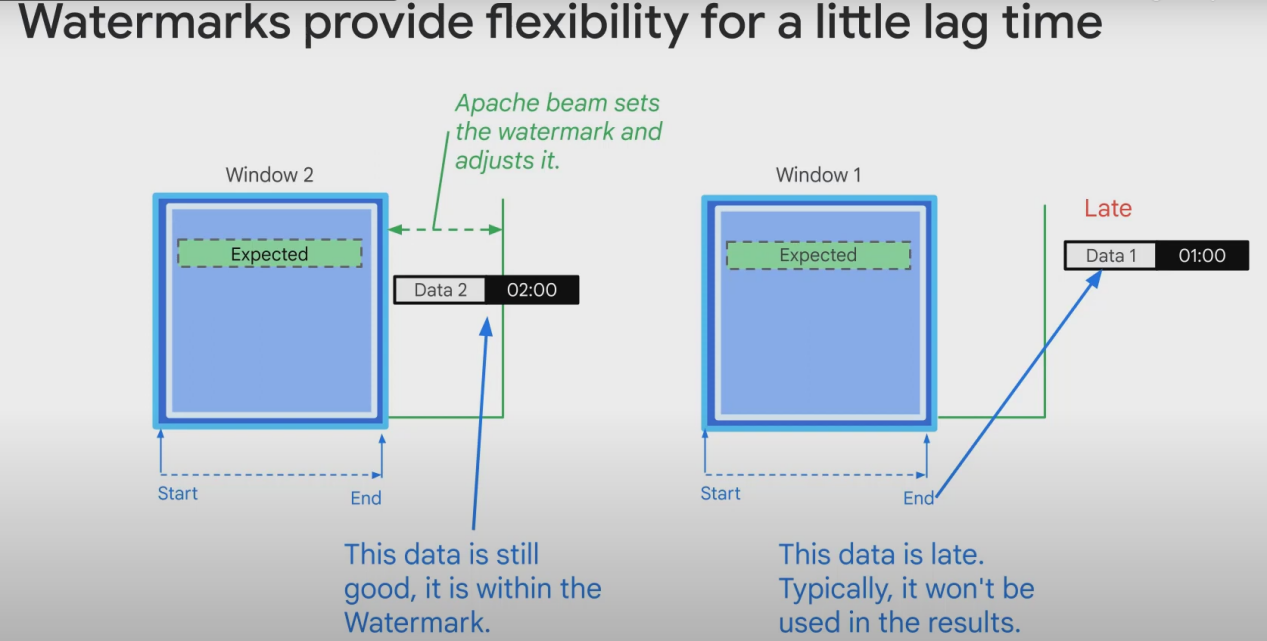
How to deal with latency? We may receive 1 piece of data after a window is closed, even though the data was originally created before the closure of our window!

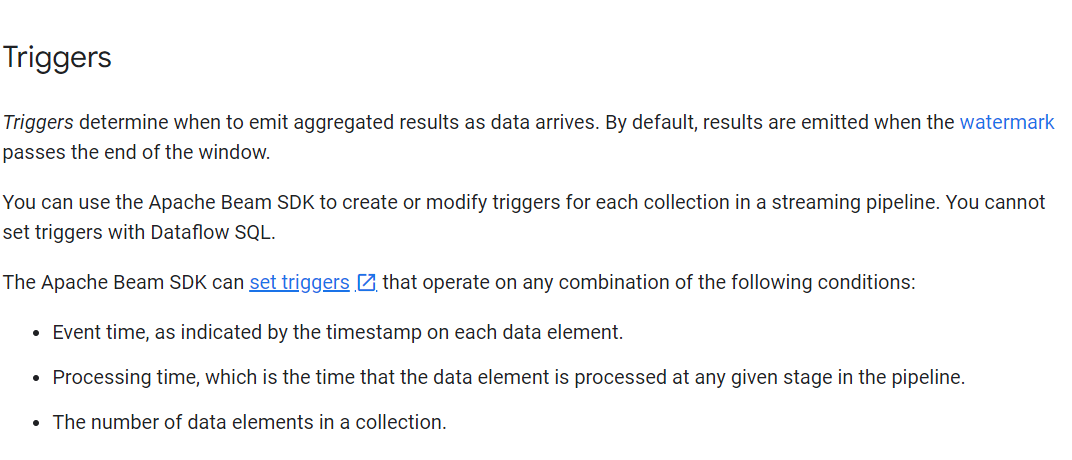


Solution:

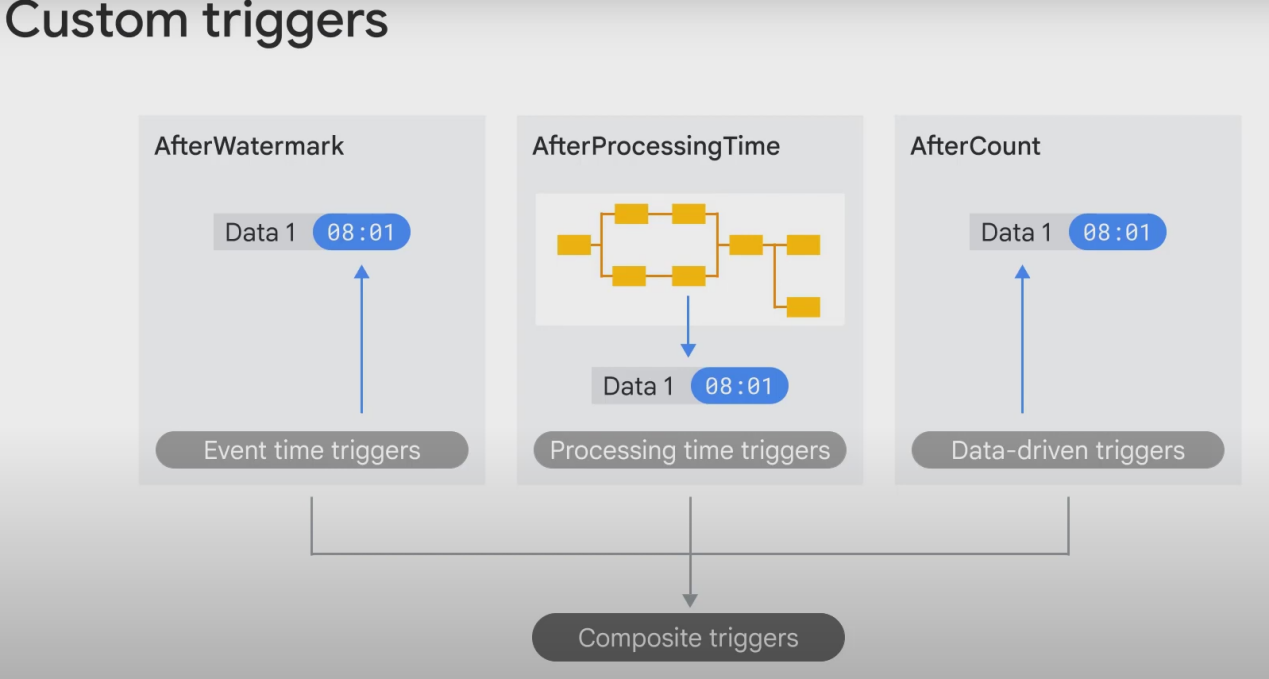
Dataflow calculates the **Watermark**: the lag/latency

It then extends each window by the current watermark. If you have 5mins windows, and a watermark of 5 secs, and start your window at 00:00, all data that arrives before 00:05:05 will still be considered to be part of that window.





More: https://beam.apache.org/documentation/programming-guide/#triggers



After Watermark (default): The messages timestamps are used to measure time.

After Count: emit the data after you receive a certain quantity of data

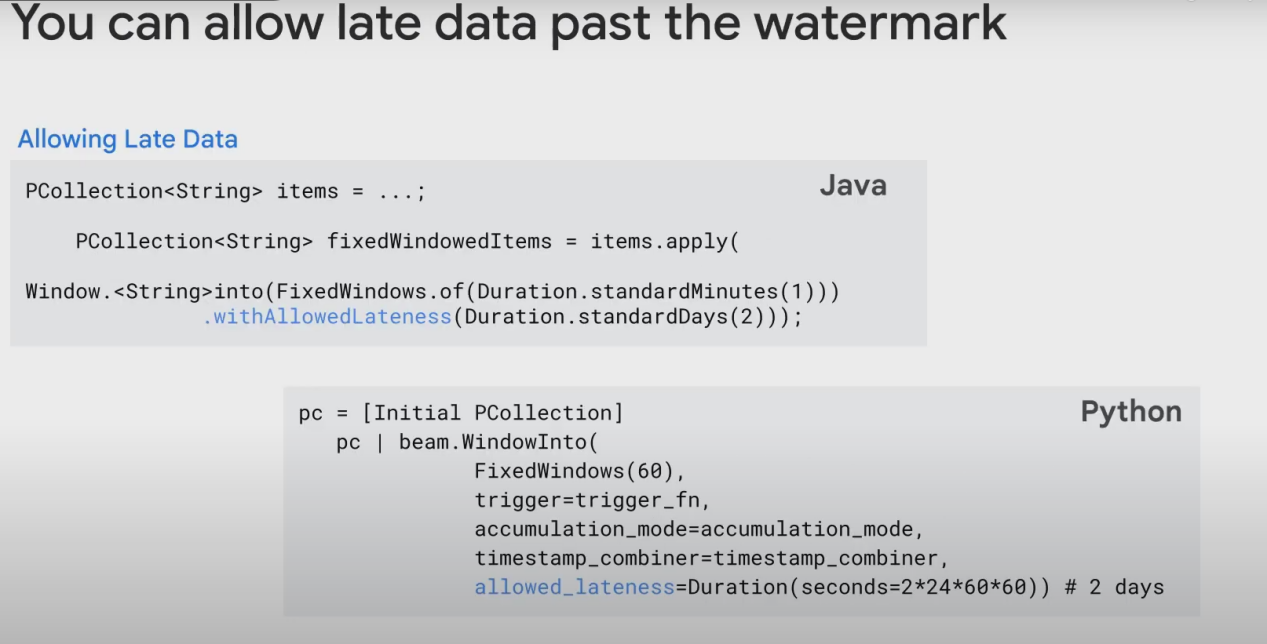
Processing time trigger:

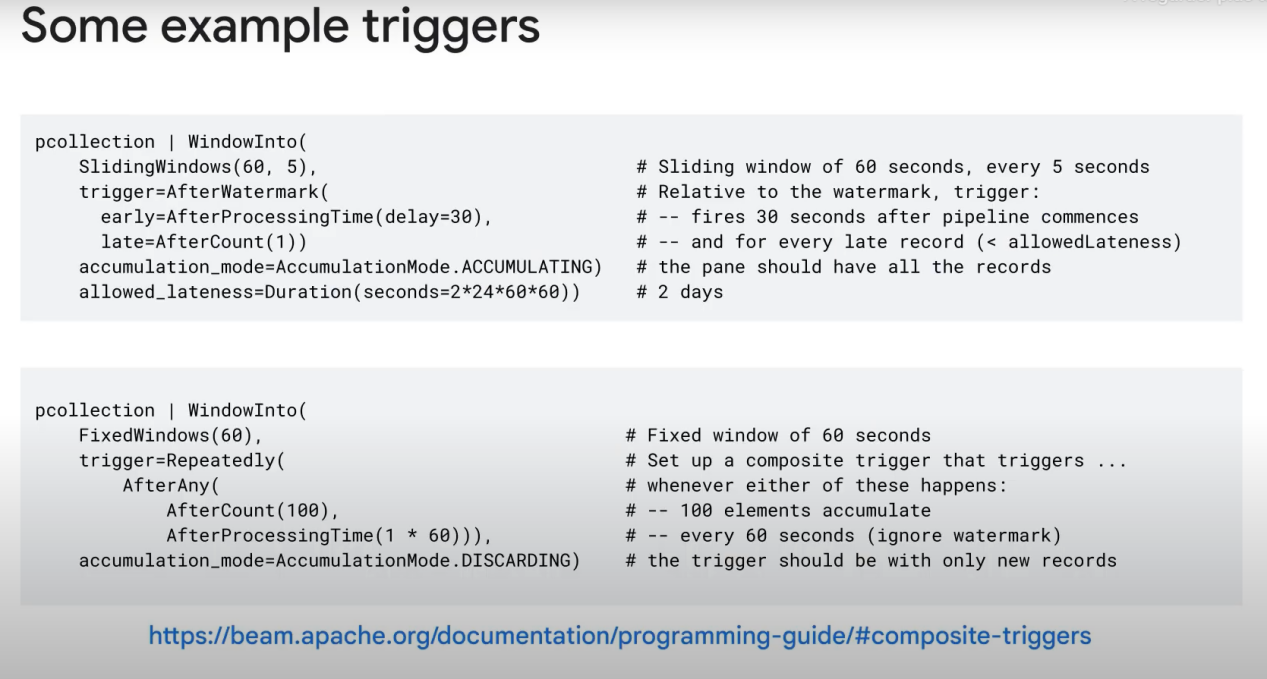
the actual clock (real time) is used to decide when to emit results, regardless of the timestamps of the data that arrives to the window.

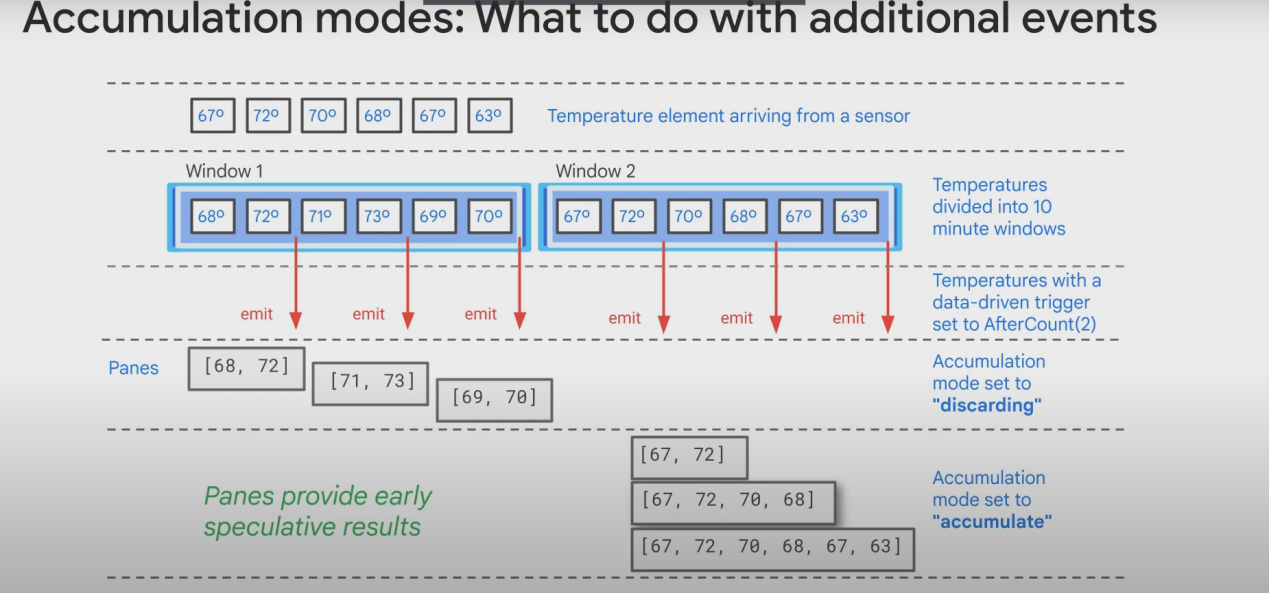
OR

These triggers operate on the processing time – the time when the data element is processed at any given stage in the pipeline. For example, the AfterProcessingTime.pastFirstElementInPane() trigger emits a window after a certain amount of processing time has passed since data was received. The processing time is determined by the system clock, rather than the data element’s timestamp. (I think this one is correct)

Composite triggers: many triggers combined.







Panes are sort of small subsets of data that you receive periodically:

