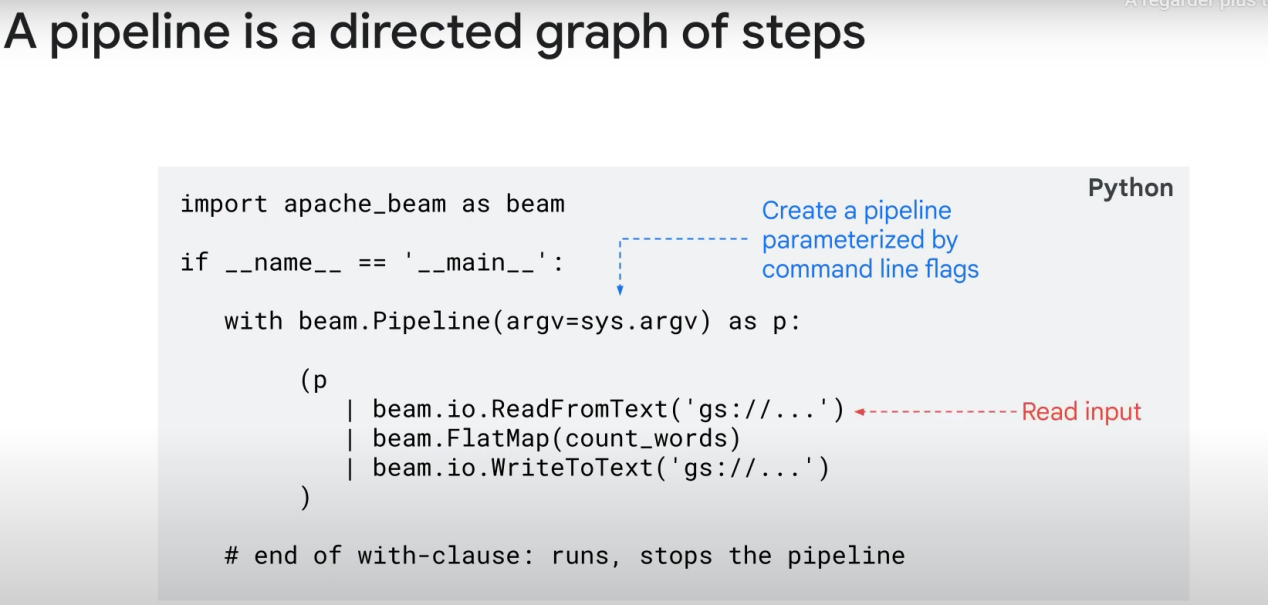


But how to get the initial Pcollection?

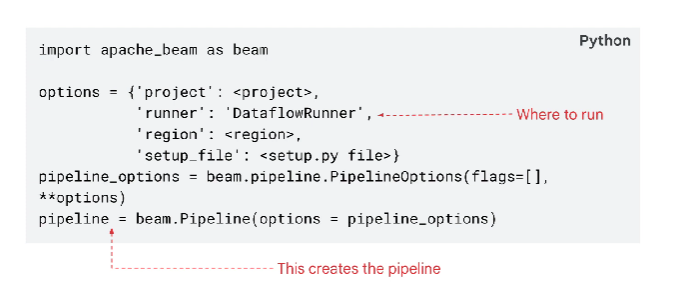
You just read it from your data lake



Note: here, «p» will be the name of our pipeline

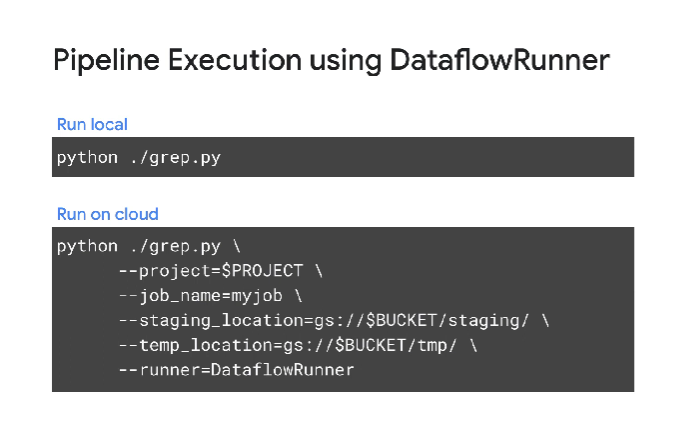
FlatMap is a ptransform: applies a function to each row in the input, and concatenate all the results

And then we run the pipeline!

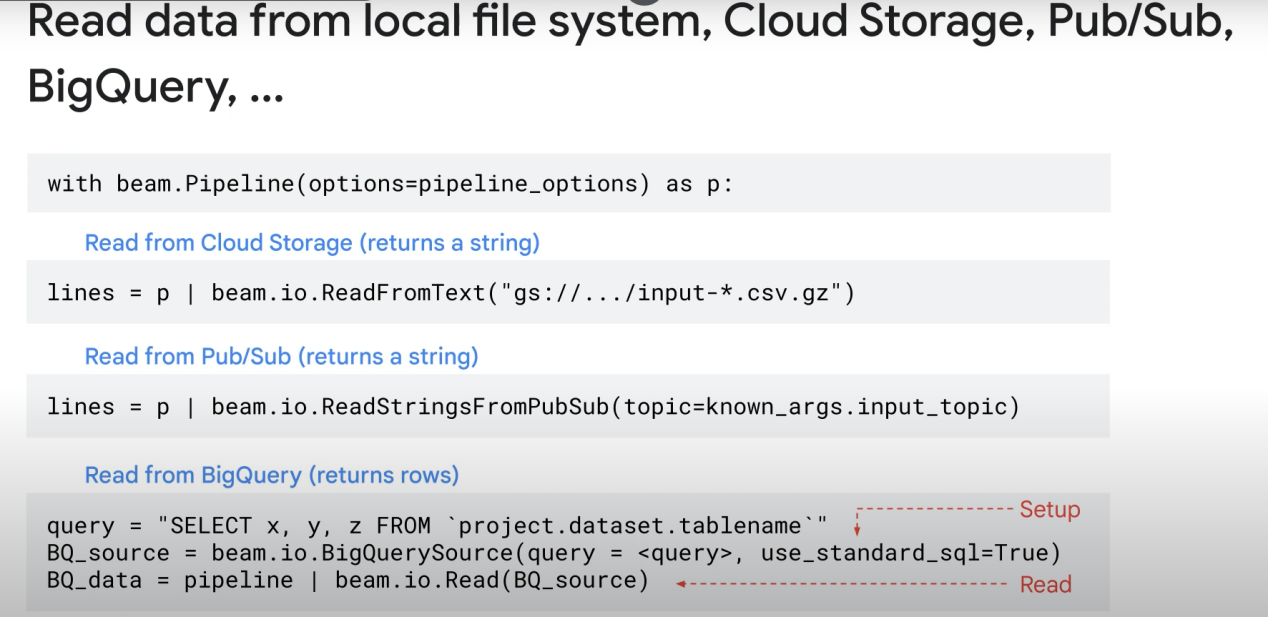


If you specify DataFlowRunner as a runner, the code will be run in google cloud.

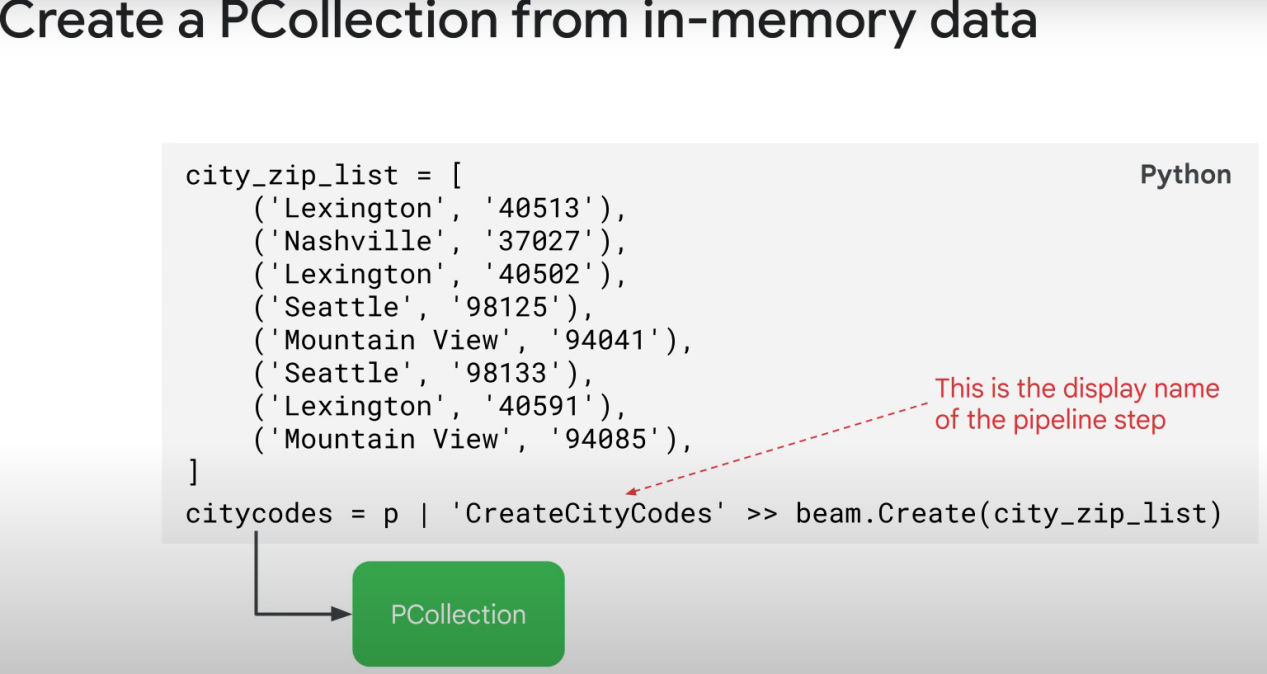
Tips for scalability: don’t use hard-coded variables



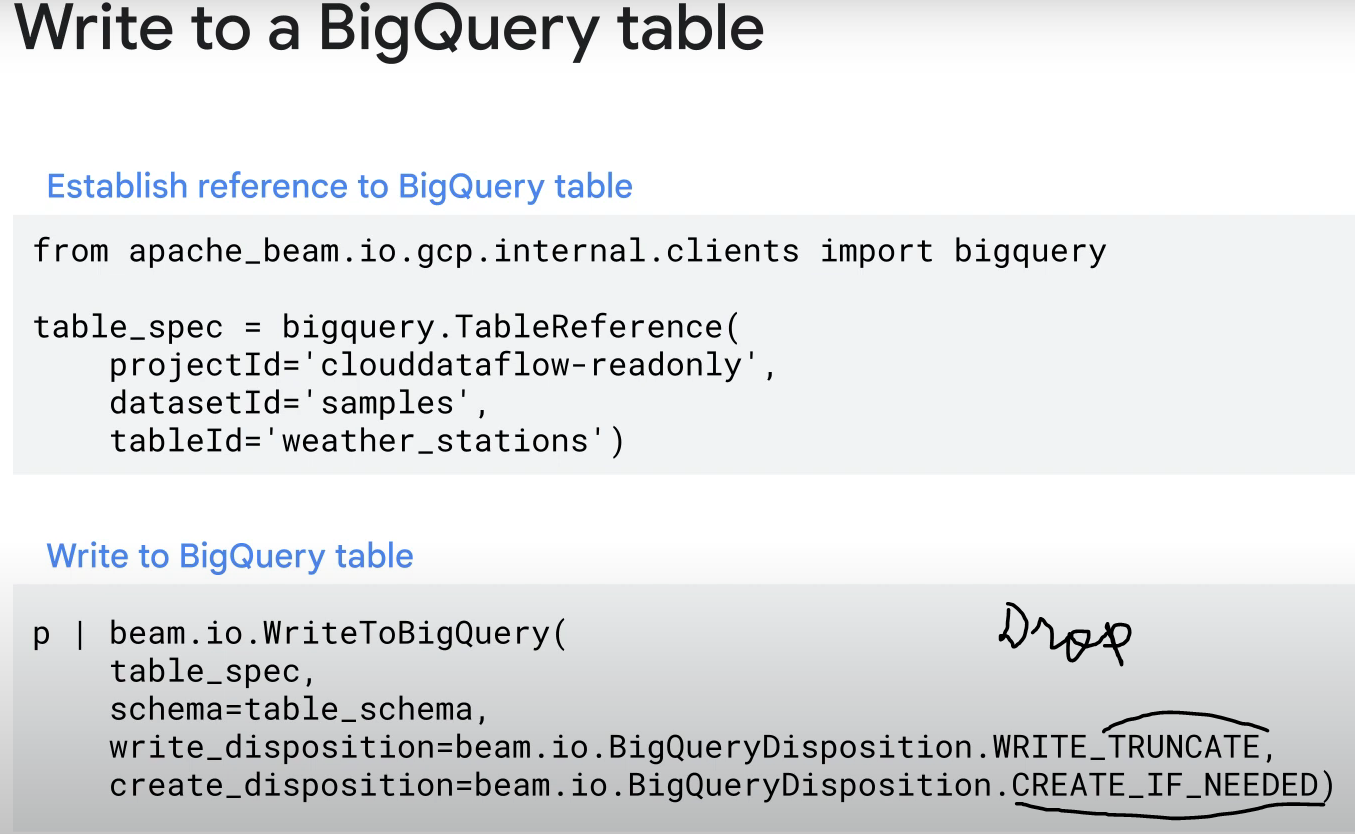
Reading the data is an art! :



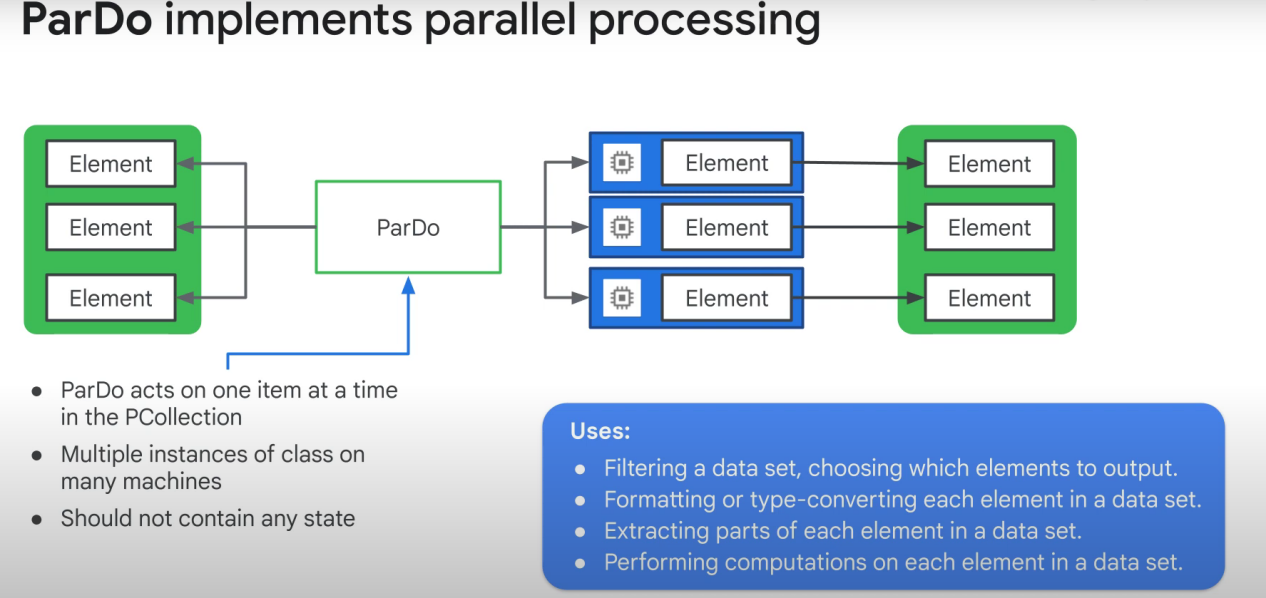
If you want to create a PCollection from local data:

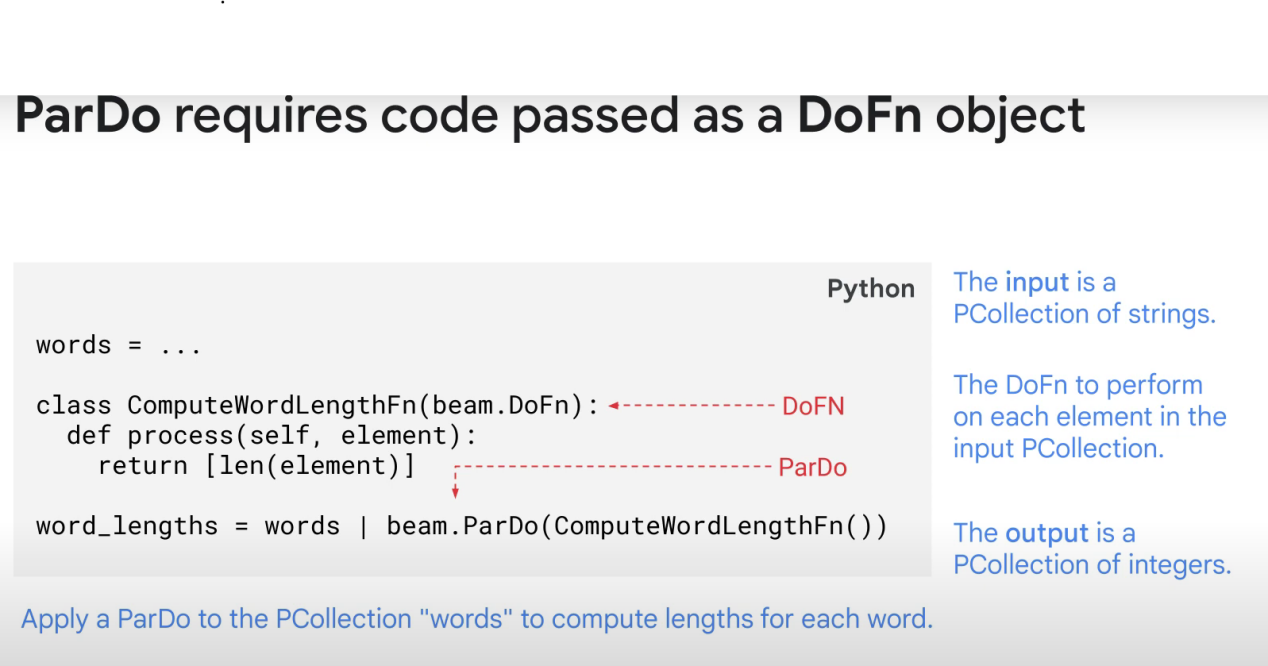


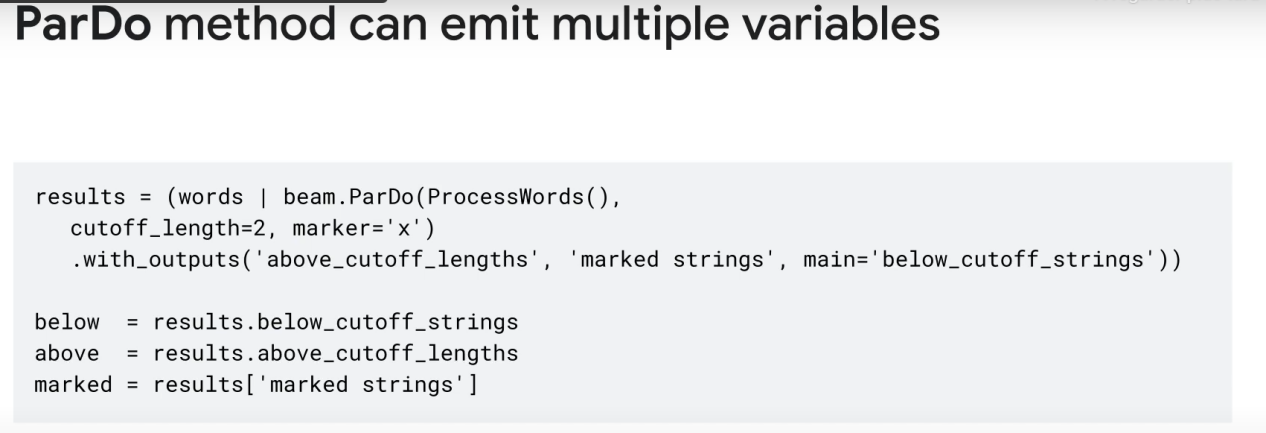
And writing it too!



ParDo Transforms:



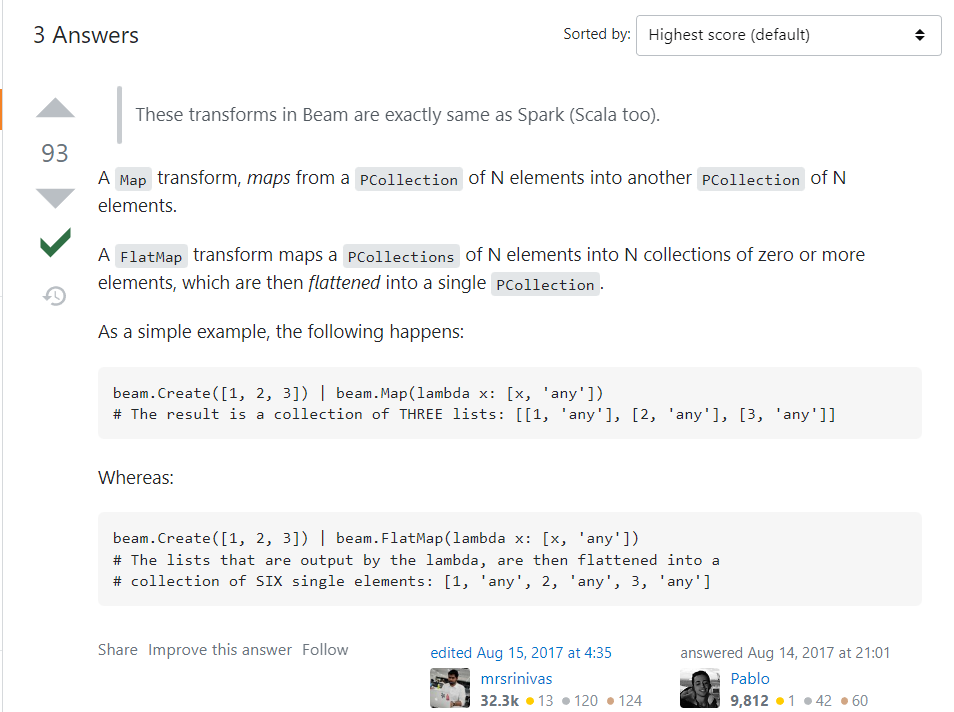


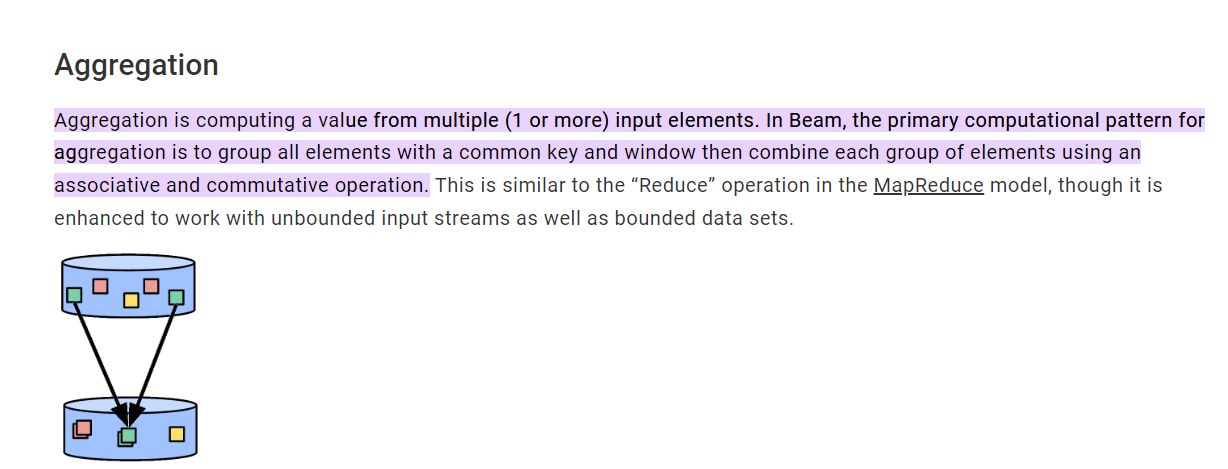


Note: A ParDo is just 1 type of PTransform

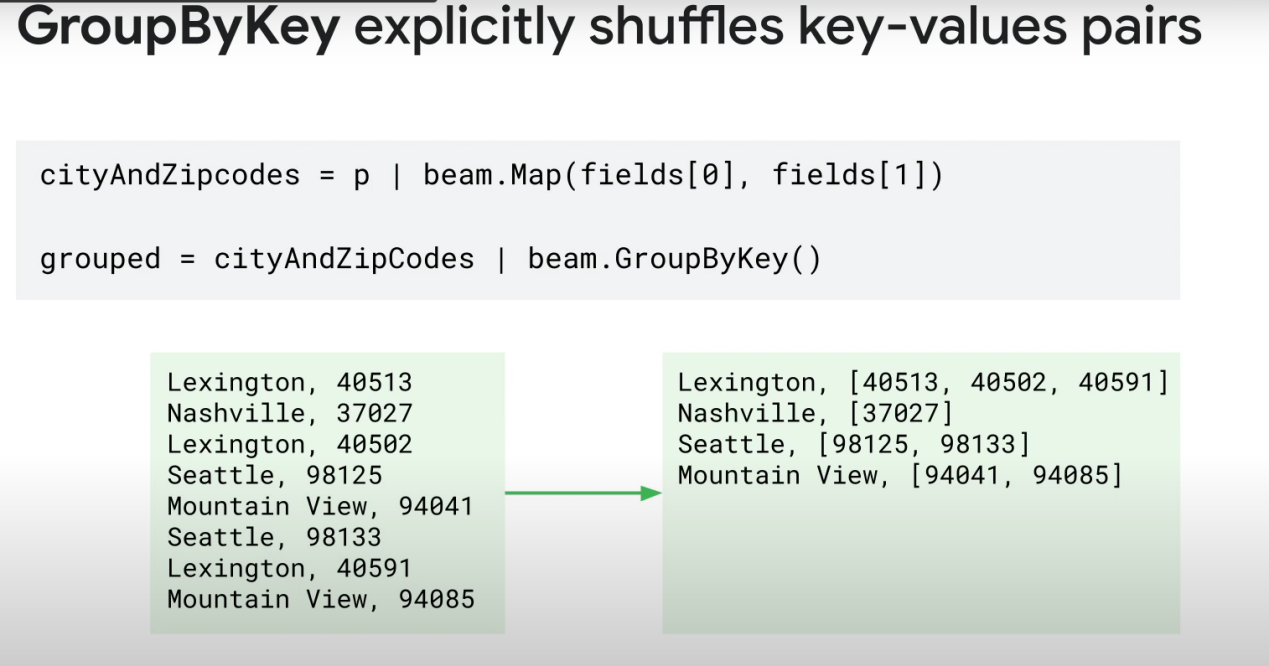
A parDo require a «DoFn» function that basically defines what transformation you want each element of your dataset to go through.

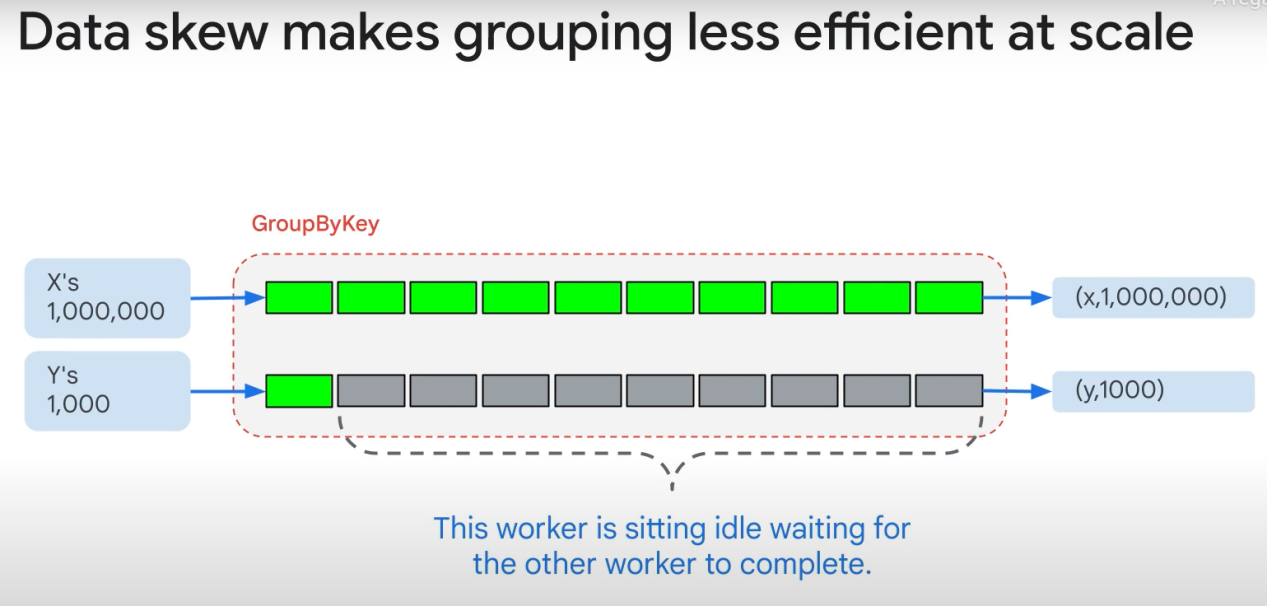
Map Vs FlatMap Ptransforms:





Grouping (it’s one type of aggregation):

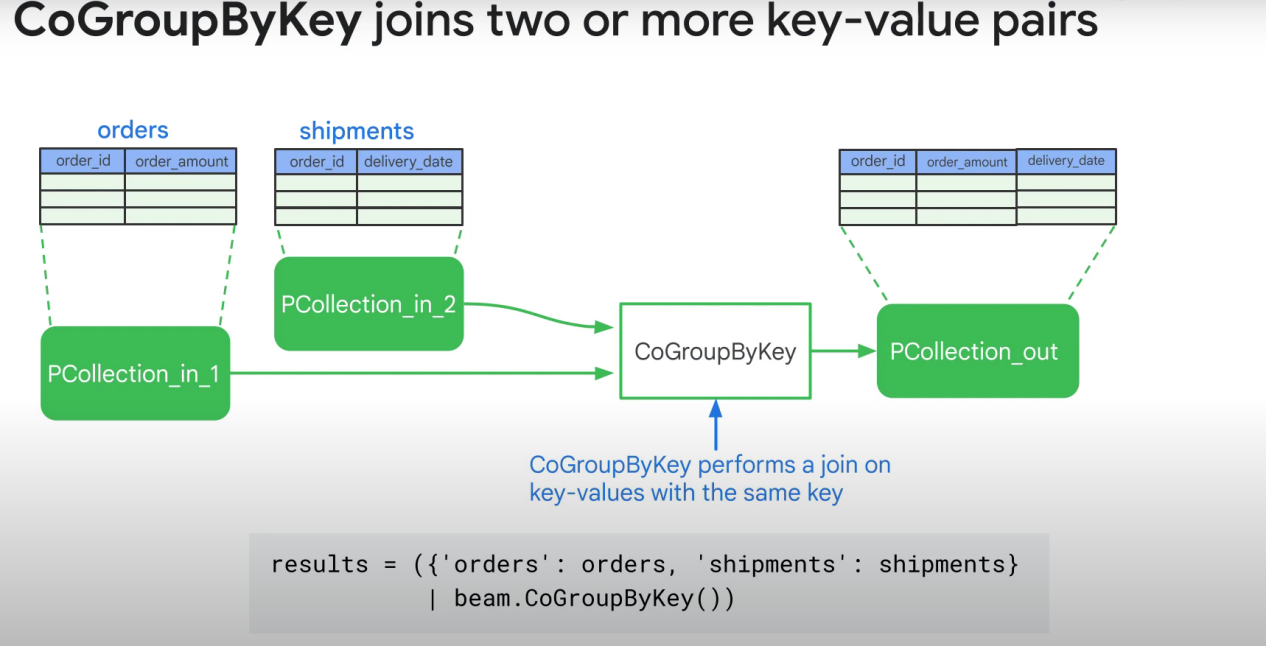




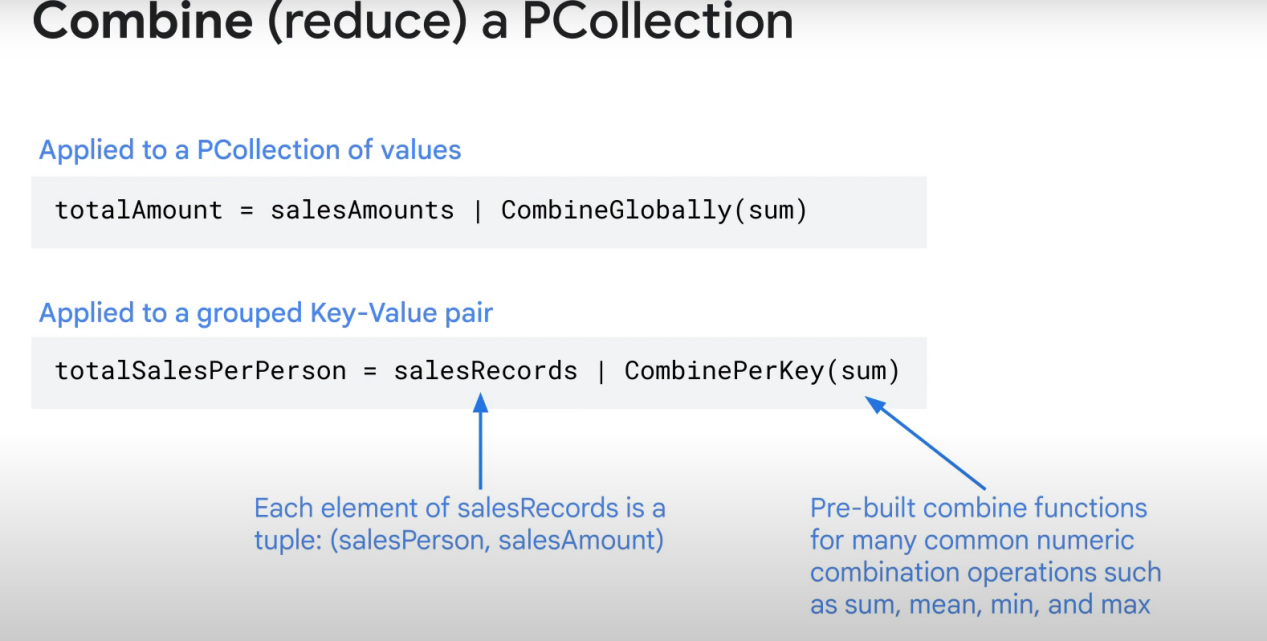
Note: the second worker taking care of Y’s will be active but idle until the other job (X’s process) is done. This will cost you money!

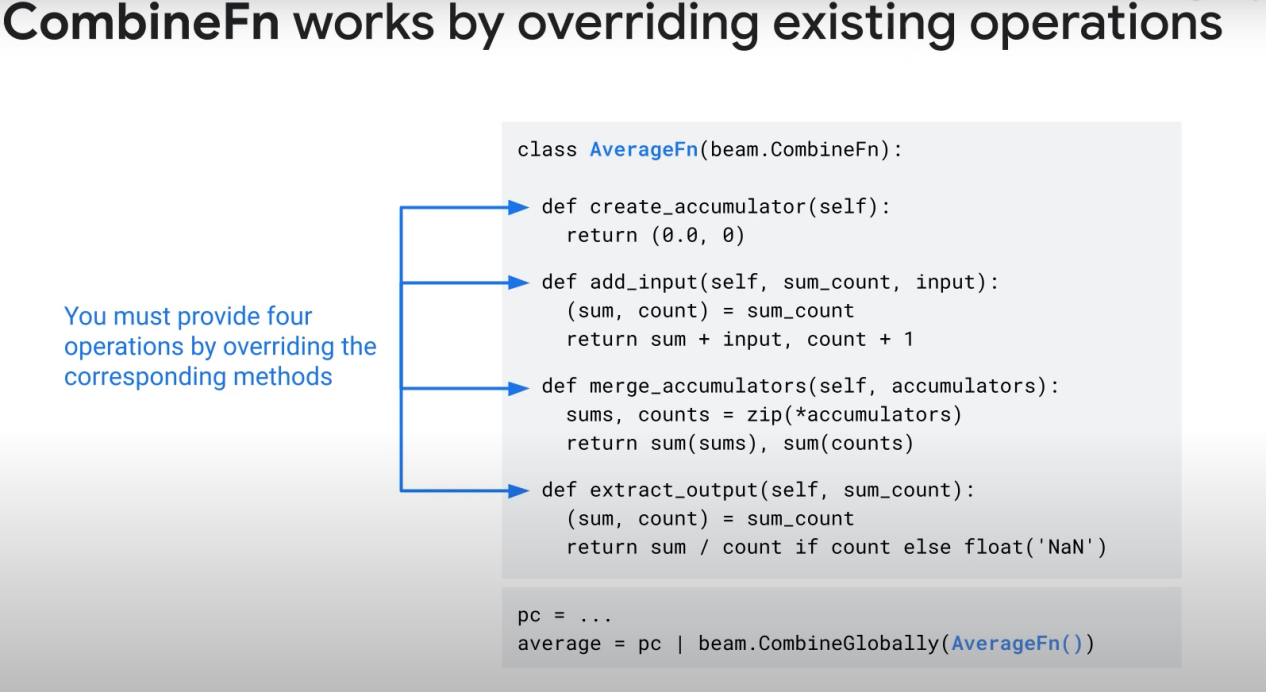
==> It’s better to group by at the end, after filetring elements, cleaning data, and reducing the size of the dataset if possible

For multiple tables (join)



Reducing data (sums, min, max, etc)

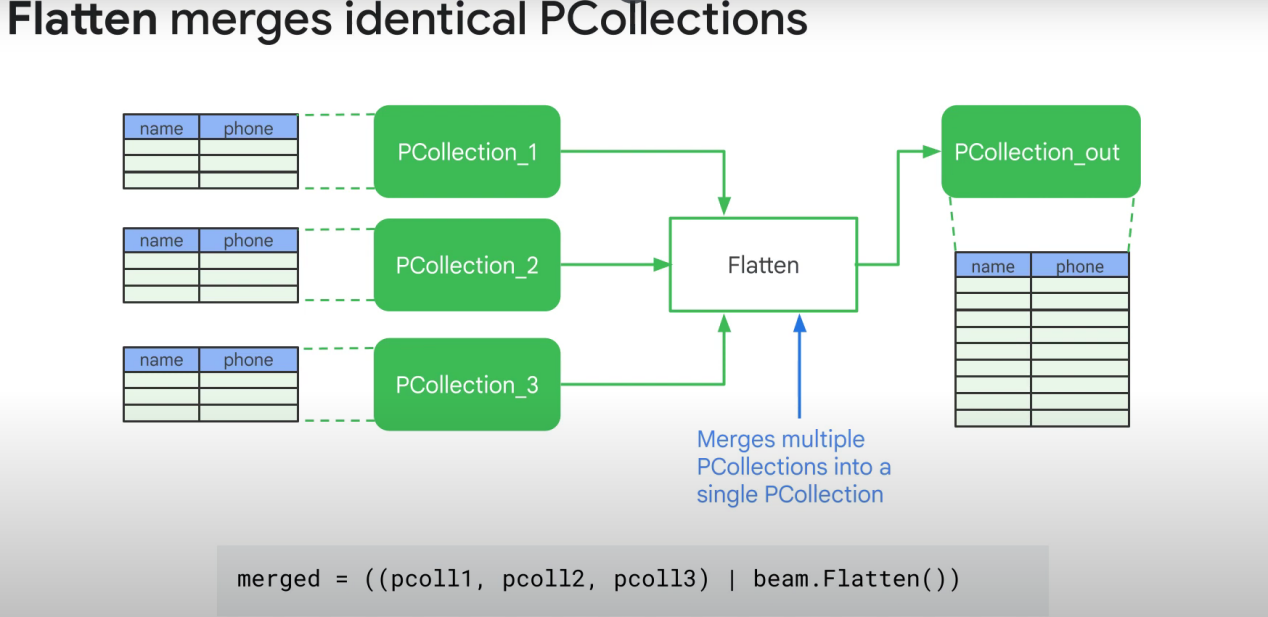


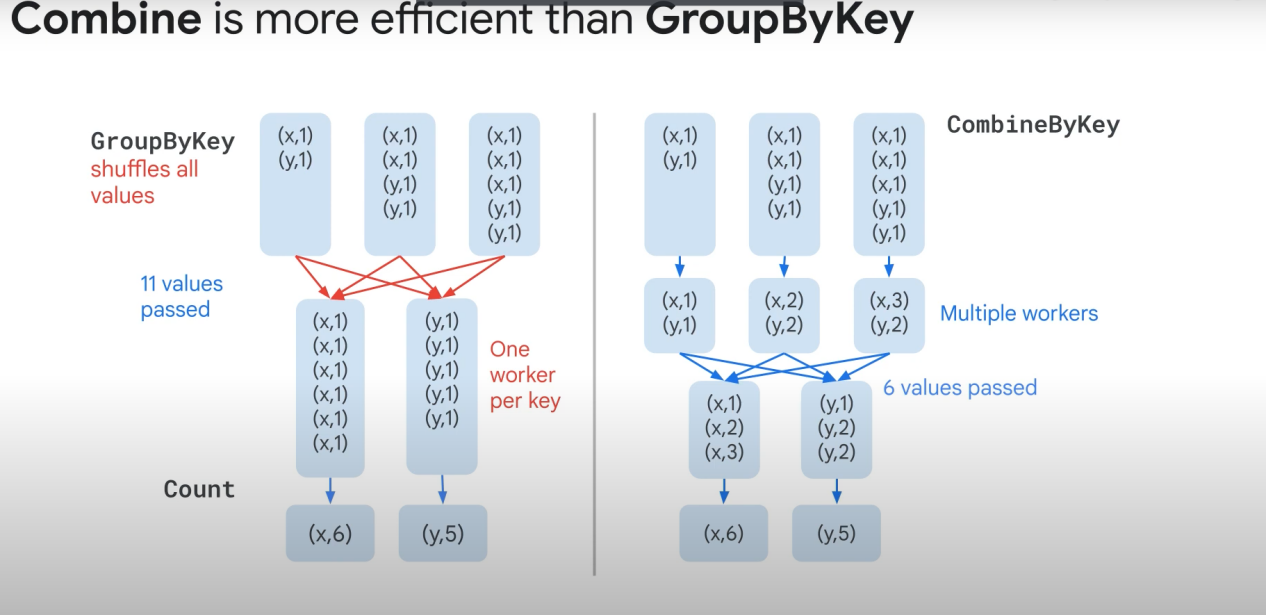
For a custom fn that reduces data:  


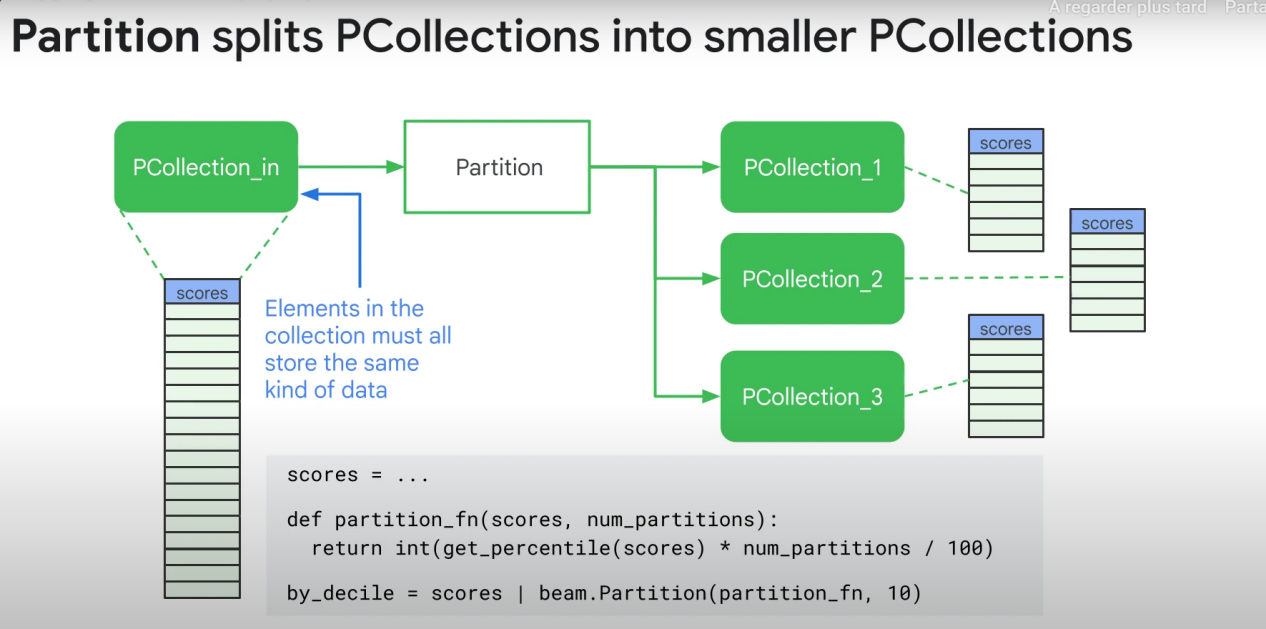
Documentation of combine: <https://beam.apache.org/releases/javadoc/2.0.0/org/apache/beam/sdk/transforms/Combine.CombineFn.html>

**COMBINE IS WAY MORE EFFICIENT THAN GROUPBYKEY**

Flatten:

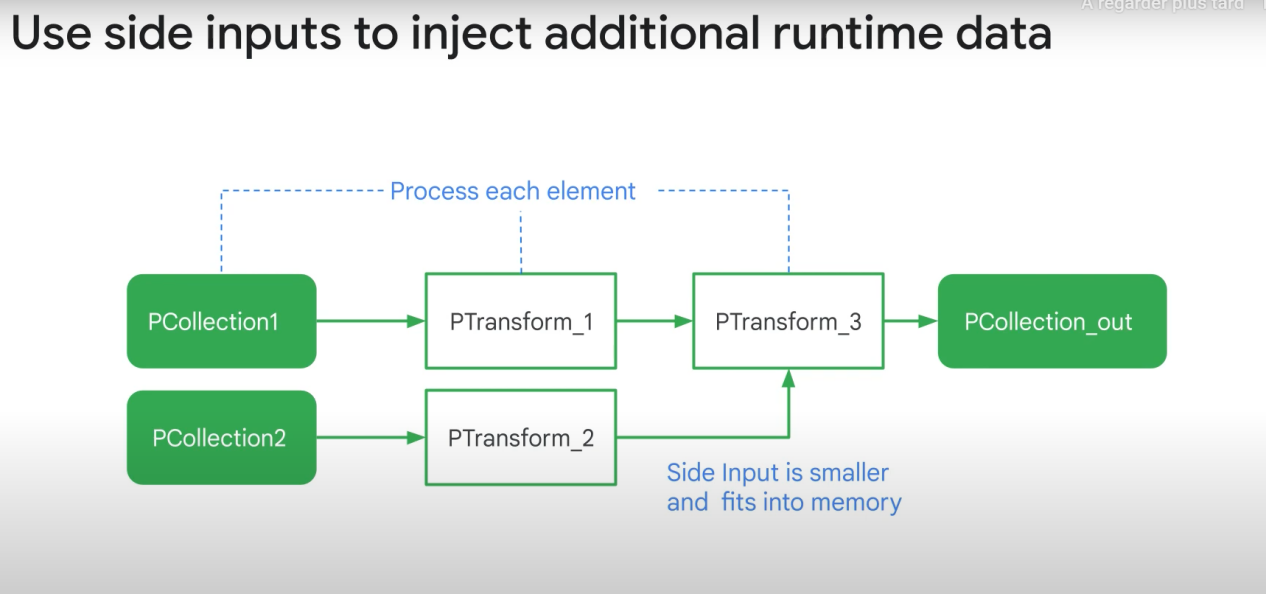






You might use partition if you want to do different ptransforms on certain parts of the data you have.

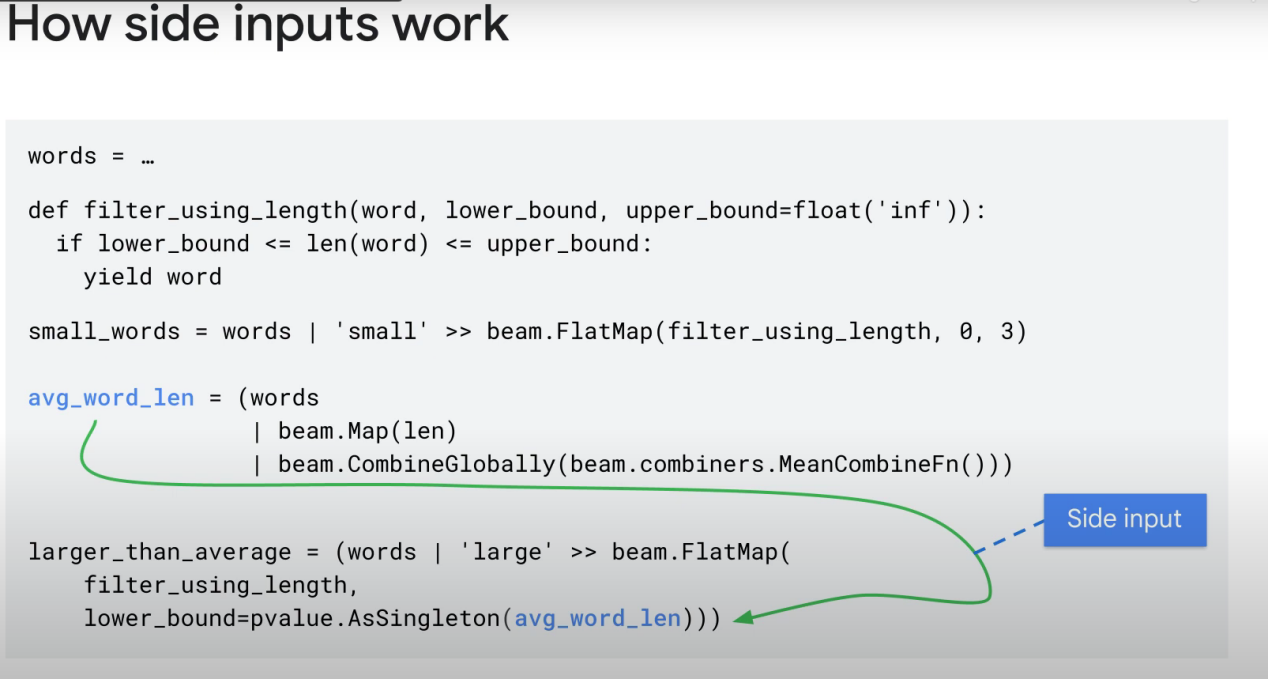
Personal note (not sure): I think all ptransforms happen in parallel except group by



The additional data needs to be determined at runtime and not hardcoded.

Useful if you need to user some other data in your ParDo transform for calculation purposes.

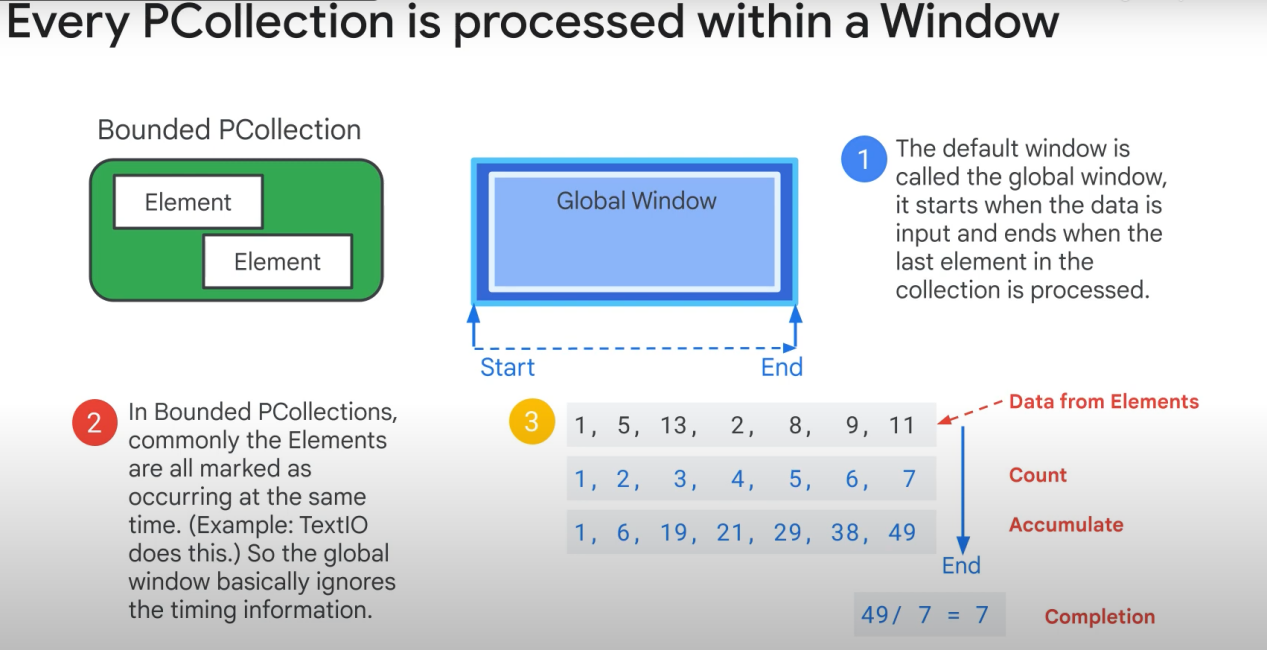
1. g:



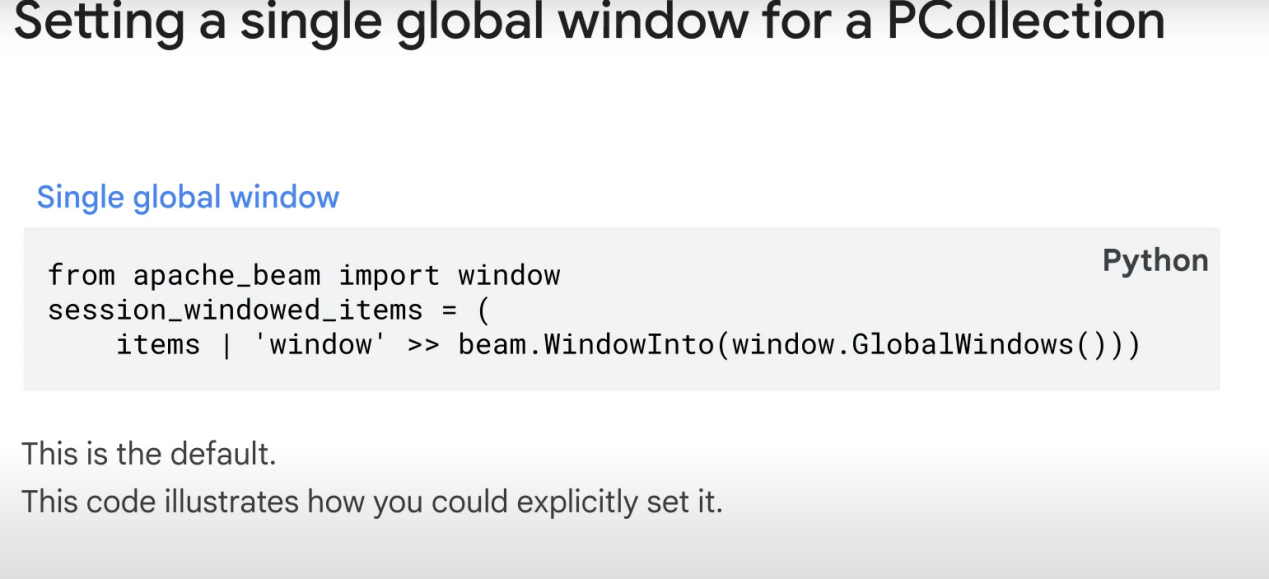
Other notes:

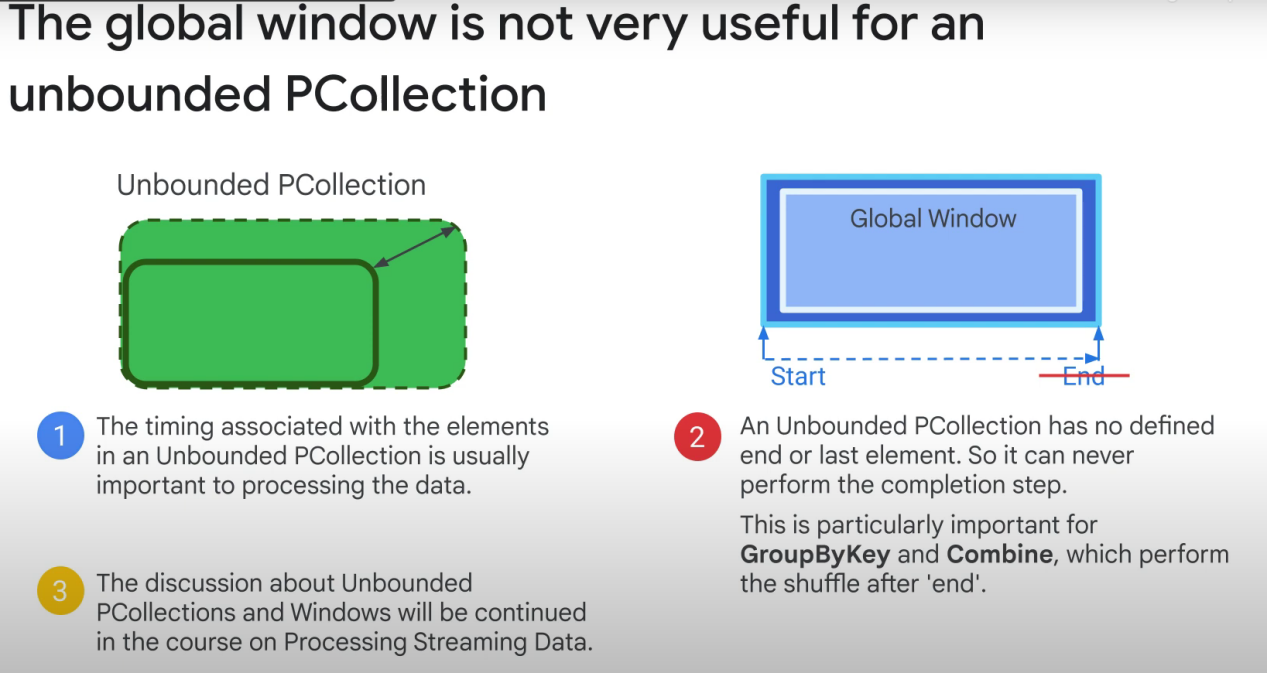
Ptransforms have 2 main steps:

1. The first step/process is started when the first element of your data is inputted, and ends when the last element is processed
2. Then, after the end of that process, we call a sort of «doAfterEnd» function. For instance, GroupByKey and Combine pTransforms perform the shuffle after the «end».

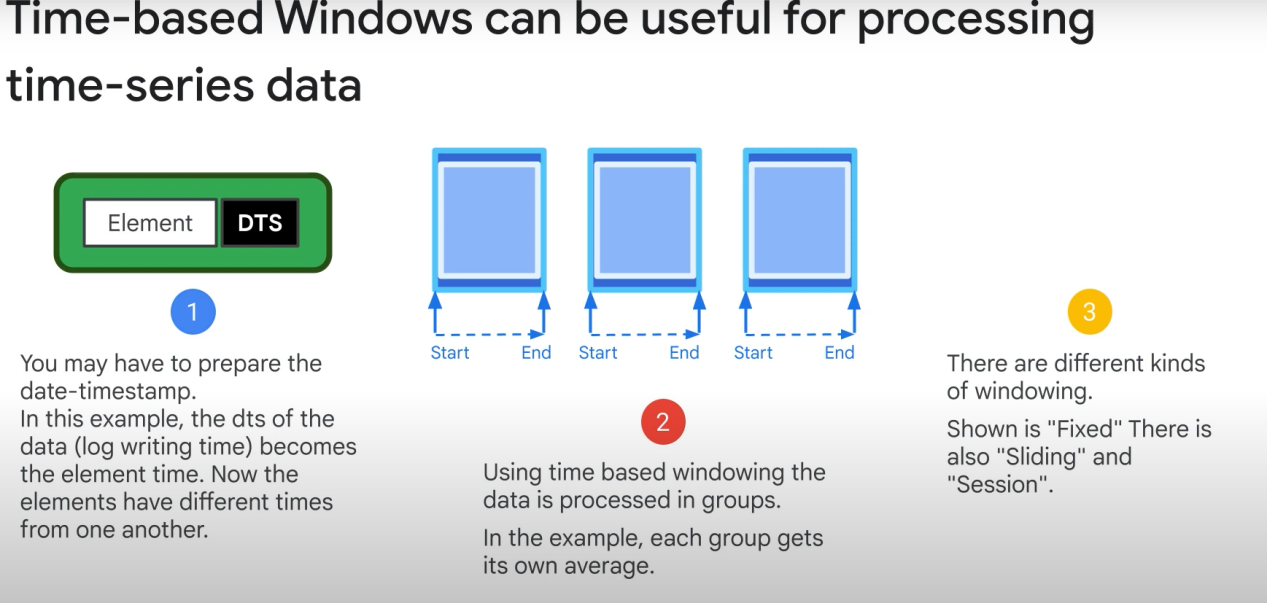


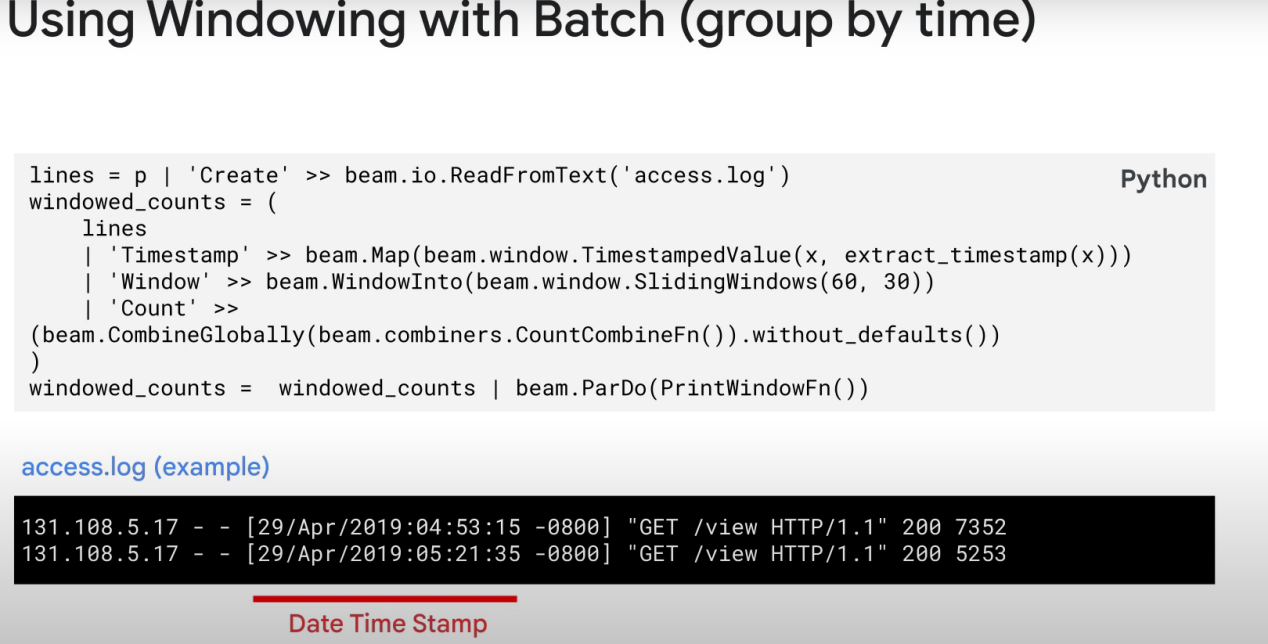
By default, dataflow uses global windows





For streaming data (real time)





SlidingWindows(60,30) means that the length of 1 window will be 60sec, and that dataflow will create a new window every 30s. The main thing to remember is that we can do this in batch.

TEMPLATES! NO NEED FOR CODE!

