## What is the purpose of RecordReader in Hadoop?

Ans: RecordReader is an interface. All mappers and reducers work only with key-value pairs, they are not aware of any other way. RecordReader take care of to convert different kinds of input format for example TextFile format, Sequence file format into key-value pairs. It reads one line at a time from corresponding input split and convert it into key-value pairs, and passing these key-value pairs as input. You can write your own RecordReader to read more than 1 line at a time.

## What happens if the number of reducers is 0?

Ans: There will not be any reducer even identity reducer. There will not be sorting, shuffling of map results and the job will be map only job. Reduce step will be skipped and mapper output will be final output.

## What is meant by Map-side and Reduce-side join in Hadoop?

Ans: Map-side Join: A map-side join between large inputs works by performing the join before the data reaches the map function. Each input dataset must be divided into the same number of partitions, and it must be sorted by the same key (the join key) in each source. All the records for a particular key must reside in the same partition. A map-side join can be used to join the outputs of several jobs that had the same number of reducers, the same keys, and output files that are not splittable (by virtue of being smaller than an HDFS block or being gzip compressed, for example). You use a CompositeInputFormat from the org.apache.hadoop.mapreduce.join package to run a map-side join.

Reduce Side Join:

A reduce-side join is the join in which input datasets don’t have to be structured in any particular way, but it is less efficient because both datasets have to go through the MapReduce shuffle. The basic idea is that the mapper tags each record with its source and uses the join key as the map output key, so that the records with the same key are brought together in the reducer.

The reducer will see the records from both sources that have the same key, but they are not guaranteed to be in any particular order. We can also use the Secondary Sort technique to control the order of the records.

The key of the map output, of datasets being joined, has to be the join key - so they reach the same reducer.

Each dataset has to be tagged with its identity, in the mapper- to help differentiate between the datasets in the reducer, so they can be processed accordingly.

In each reducer, the data values from both datasets, for keys assigned to the reducer, are available, to be processed as required.

A secondary sort needs to be done to ensure the ordering of the values sent to the reducer.

If the input files are of different formats, we would need separate mappers, and we would need to use MultipleInputs class in the driver to add the inputs and associate the specific mapper to the same.

## What is the significance of conf.setMapper class?

Ans: job.setmapperClass(MyJob.MyMapper.class). It is a class that extends Mapper. It sets the mapper class and all the stuff related to mao job such as reading data, generating key-value pair out of the mapper.

## Give an example scenario on the usage of counters.

Ans: Counters are used for gathering statistics about the job for quality control or for application level statistics. It is the easiest way to investigate internal behaviors of mapReduce jobs. Counters are used to track the number of malformed records. It is a simplest way to spot issue.

## Elaborate some problems which can only be solved by MapReduce and cannot be solved by PIG?

1. Answer: If there is large datasets then Pig will not be able to handle for instance key distribution so it can be solved through MapReduce.
2. If we need definite driver program control then we have to use MapReduce and it can not be solved through Pig.
3. Things like doing a bunch of DISTINCTs or loading the same dataset twice for trying to do a cross product or typical kind of joins.

## In what kind of scenarios, MR jobs will be more useful than PIG?

Answer: Complex branching logic which has a lot of nested if .. else .. structures is easier and quicker to implement in Standard MapReduce, for processing structured data, it also simplifies things like JOIN. Also Standard MapReduce gives full control to minimize the number of MapReduce jobs that your data processing flow requires, which translates into performance.

## What are combiners and when are these used in a MapReduce job?

Answer: A Combiner is also known as a semi-reducer, is an optional class that operates by accepting the inputs from the Map class and thereafter passing the output key-value pairs to the Reducer class. The main function of a Combiner is to summarize the map output records with the same key. The output (key-value collection) of the combiner will be sent over the network to the actual Reducer task as input.

The Combiner class is used in between the Map class and the Reduce class to reduce the volume of data transfer between Map and Reduce. Combiner reduces network congestion and optimized network bandwidth to a great extent. It also helps in improving the data processing speed

**Combiner should be used only when the function is both commutative and associative.**