1. **What is the purpose of Record Reader in Hadoop?**

It is the interface between input split and reducer. It reads one line at a time (default) from input split and convert to (key, value) format for mapper input. We can write our own code for record reader to increase the no of lines read at a time.

1. **What happens if number of reducers are 0?**

In this case the outputs of the map-tasks go directly to the Filesystem, into the output path set by setOutputPath (Path). The framework does not sort the map-outputs before writing them out to the Filesystem.

1. **What is meant by Map-side and Reduce-side join in Hadoop?**

Map Side Joins: Are also called as replicated joins. Join operation between one large data and one or more small data sets that can be performed on the map-side. Maximum size limit of the cache is 2 GB. Completely eliminates the need to shuffle any data to the reduce phase. In Hadoop Map side joins are achieved via Distributed cache. All the data sets except the large one are read into the memory during the set up phase of each map task, which is limited by the JVM heap(2 GB). Join is done entirely in the Map phase, with the large data set as the input for the MapReduce job. Output is a number of files equivalent to the number of map tasks

**Reduce side join**

Reduce Side Joins are used when both data sets are huge. Joins of datasets done in the reduce phase are called reduce side joins. They are less efficient than maps-side joins because the datasets have to go through the sort and shuffle phase.

1. **What is the significance of conf.setMapper class?**

Conf.setMapperclass sets the mapper class and all the stuff related to map job such as reading data and generating a key-value pair out of the mapper

1. **Give an example scenario on the usage of counters.**

Counters are a useful channel for gathering statistics about the job for quality control or for application level-statistics. They are also useful for problem diagnosis. If you are tempted to put a log message into your map or reduce task, then it is often better to see whether you can use a counter instead to record that a particular condition occurred. In addition to counter values being much easier to retrieve than log output for large distributed jobs, you get a record of the number of times that condition occurred, which is more work to obtain from a set of logfiles.

1. **Elaborate some problems which can only be solved by MapReduce and cannot be solved by PIG?**

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 you could use pig, it also simplifies things like JOIN. Also Standard MapReduce gives you full control to minimize the number of MapReduce jobs that your data processing flow requires, which translates into performance. But it requires more time to code and introduce changes.

1. **In what kind of scenarios, MR jobs will be more useful than PIG?**

* When you need definite driver program control.
* Whenever the job requires implementing a custom partitioner.
* If there already exists pre-defined library of Java Mappers or Reducers for a job.
* If you require good amount of testability when combining lots of large data sets.
* If the application demands legacy code requirements that command physical structure.
* If the job requires optimization at a particular stage of processing by making the best use of tricks like in-mapper combining.
* If the job has some tricky usage of distributed cache (replicated join), cross products, groupings or joins.

1. **What are combiners and when are these used in a MapReduce job?**

A Combiner, 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. Usually, the output of the map task is large and the data transferred to the reduce task is high.

Here is a brief summary on how MapReduce Combiner works −

* A combiner does not have a predefined interface and it must implement the Reducer interface’s reduce() method.
* A combiner operates on each map output key. It must have the same output key-value types as the Reducer class.
* A combiner can produce summary information from a large dataset because it replaces the original Map output.

Although, Combiner is optional yet it helps segregating data into multiple groups for Reduce phase, which makes it easier to process.