**Give a brief answers to the questions below:**

1. **What is the purpose of RecordReader in Hadoop?**

RecordReader reads <key, value> pairs from an InputSplit.

RecordReader, typically, converts the byte-oriented view of the input, provided by the InputSplit, and presents a record-oriented view for the Mapper & Reducer tasks for processing. It thus assumes the responsibility of processing record boundaries and presenting the tasks with keys and values.

1. **What happens if the number of reducers is 0?**

If the number of reducers is 0 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?**

Joins performed by Mapper are called as Map-side joins. The inputs to each map must be partitioned and sorted in a specific way. 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 and which is mandatory. 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 no bigger than the HDFS block size.

Joins performed by Reducer can be treated as Reduce-side joins. Reduce-Side joins are simpler than Map-Side joins since the input datasets need not to be structured. But it is less efficient as both datasets have to go through the MapReduce shuffle phase. The records with the same key are brought together in the reducer. We can also use the Secondary Sort technique to control the order of the records.

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

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

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

Hadoop MapReduce Counter provides a way to measure the progress or the number of operations that occur within MapReduce programs.

These counters are very useful especially when you evaluate some MapReduce programs.

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

Customized partitioner is used in MapReduce which cannot be used in Pig. Let us take an example to understand it:

Let us consider a scenario where we need to count the number of students in a group of colleges. We have a data set and a list of different colleges. So we need just one MapReduce program to count the number of students in two colleges. Let us assume that one college is ‘A’ and the other is ‘X’.

The key of both the colleges need to be same so that we can map the number of students to one reducer. Hence in this case we need to write a custom partitioner based on the key which is the college name. In our case we will create a common key for both the colleges and for other colleges there will be other keys. Hence, when the map reduce program finds the name of these two colleges it has to consider a common key for both of these.

So whenever we get college = A or X then we need to pass the similar hashcode.

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

As pig is not a framework, so we cannot direct the execution engine to customize the partitioner. In such scenarios, MapReduce jobs are preferred.

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

A combiner is also known as a semi-reducer. It is used between the map class and the reduce class to reduce the volume of data transfer between the map and reduce programs.

So basically the main function of a combiner is to summarize the map output records with the same key. The output which is key-value pair is sent to the reducer as an input