1) MapReduce is a data processing model based mostly on Scheme's map and reduce functions. The core idea behind MapReduce is to organize data in terms of key-value pairs. In order to simplify this, the data is usually text or some binary representation As the name implies, MapReduce is composed of two major phases the Map Phase and the Reduce Phase. The Map Phase consists of an input of partitioned data, the Map phase then '*maps*' a function or a process on each of these data segments, producing intermediate results. These intermediate results then pass on to the Reduce phase which collects the intermediate results, sort them by key and stores them in files in a directory. Sometimes, you can include a step called a Combiner, where between the Map and Reduce phase you can merge results which share the same key in order to speed up the Reduce phase.

Examples where MapReduce is useful:

Large datasets where the data is well defined, but the task you're seeking to accomplish isn't particularly complicated. For example:

- Counting the frequency of a word.
- How many students go to X school.
- Group the name of people who share a title.

2) RDD's are a collection of partitioned records that are read-only. The Spark RDD stores the state of memory as an object which is shareable across the various jobs which are running. The key idea behind the Spark RDD is that it is stored in memory and not on disk (although you can if you needed to). Memory is faster than writing to disk, and since MapReduce spends a lot of its time writing to disk, the Spark RDD outperforms MapReduce in terms of computation speed.

Another key feature of RDD's is that they are recomputeable, so you are able to evaluate things on the fly - for example, you can morph RDDs into new RDDs or attempt to compute or extract a value from the RDD. Since in Spark this RDD is kept in memory, these on the fly actions are quick, much faster than doing an additional MapReduce stage in order to obtain the desired results.

Finally, since everything is computed in memory, there is no intermediate data being written to disk, this is what is known as materialization. Since materialization does not occur in RDDs, you have a performance gain since IO writes to disk are slow compared to memory operations. Apparently this is a big MapReduce drawback that database engineers and scientists knew.

3) Supervised learning is a specialization of machine learning where algorithm are developed that can 'learn' over time. Supervised learning algorithms make predictions after it is 'trained' by a set of examples. For example, suppose we have a function Y = f(x) - the objective of a

supervised learning algorithm would be to determine the relationship between the input x and the output Y in order to predict outputs based on non-training input data.

Supervised learning is broken down into two general categories: Classification and Regression models. In classification, you want the output to be a category, such as a color, a type of terrain, disease, gender, shape, etc. In essence, classifiers make a decision as to whether or not a particular set of data belongs to a class or not. The data belongs to a class if the value calculated by the algorithm surpasses a certain threshold determined by the user. In regression, you want a discrete value, a real value such a temperature, amount of money, number of people, price of an item, etc. Linear regression accomplishes this by attempting to fit a straight line through as many data points as possible in order to approximate what is known as a continuous value.

Examples of problems where SL can be used:

- Identifying cancer tissue among various pictures of patient tissue.
- Identifying rivers, land, buildings, etc from pictures.
- Attempting to predict patients weight given certain parameters, like height, gender, BMI, race, etc.

4) In Unsupervised learning, there is no training data-set provided to the algorithm. The algorithm is left to learn about the structure itself in order to provide interesting structures and observations in the data. There are two major groups in unsupervised learning, clusters and association problems. Clustering is when you seek to group data and association is when you want to understand the relationships and rules within your data-set.

I think the best way to describe the difference between supervised and unsupervised learning is that, very generally speaking, in supervised algorithms, there is such a thing as correct answers, and you want the algorithm to be able to provide the correct answers about a data-set as often as possible. As for unsupervised algorithms, they are generally used when there are no answers to the data that is being studied - you use the algorithm in order to provide unknown insights into the data.

Example for unsupervised algorithm:

Medicine - especially in the pursuit of cures for genetic diseases or diseases that are poorly understood. There are genetic diseases out there whose causes we do not understand, such as MS, ALS and various other diseases. The root causes and potential cures of these diseases are highly suspected to be hidden in our genes - but the human body has many genes, and those genes can interact with dozens of other genes in ways we are just beginning to understand. With unsupervised learning, we glean insight into the interaction of various genes that are highly suspected to be related to these diseases.