Transformation: Transformation refers to the operation applied on a RDD to create a new RDD. Filter, groupBy and map are the examples of transformations. It is also a function that can take an RDD as the input and produce one or many RDDs, but always produce one or more new RDDs by applying the computations they represent. There are two types of transformation: 1) Narrow transformation, 2)Wide transformation.

There are list of some common transformation supported by spark:
1)map(func): Return a new distributed dataset formed by passing each element of the source through a function func.
2)filter(func): Return a new dataset formed by selecting those elements of the source on which *func* returns true.
3)flatMap(func): Similar to map, but each input item can be mapped to 0 or more output items
4)mapPartitions(func): Similar to map, but runs separately on each partition (block) of the RDD, so *func* must be of type Iterator<T> => Iterator<U> when running on an RDD of type T.
5)union(otherDataset): Return a new dataset that contains the union of the elements in the source dataset and the argument.

Actions: Actions refer to an operation which also applies on RDD, that instructs Spark to perform computation and send the result back to the driver.
Transformations and Actions in Apache Spark are divided into 4 major categories: 1) General, 2) Mathematical and Statistical, 3) Set Theory and Relational, 4) Data-structure and IO.

There are list of some common transformation supported by spark:
1)count(): Return the number of elements in the dataset.
2)first(): Return the first element of the dataset (similar to take(1)).
3)take(n): Return an array with the first *n* elements of the dataset.
4)reduce(func): Aggregate the elements of the dataset using a function *func* (which takes two arguments and returns one). The function should be commutative and associative so that it can be computed correctly in parallel.

5)collect(): Return all the elements of the dataset as an array at the driver program. This is usually useful after a filter or other operation that returns a sufficiently small subset of the data.

The Spark RDD API also exposes asynchronous versions of some actions, like foreachAsync for foreach, which immediately return a FutureAction to the caller instead of blocking on completion of the action. This can be used to manage or wait for the asynchronous execution of the action.