Qus:- Pen down the limitations of MapReduce.

Ans:- some limitations are given below:

1- **Issue with small files: -** it is now suited for small data. It lacks the ability to efficiently support the random reading of small files because of its high capacity design.

2- **Slow processing speed: -** it processes large dataset. There are tasks that need to performed Map & Reduce. And it requires a lot of time to perform these tasks.

3- **Support for batch processing only:** - it supports batch processing only. It does not process streamed data.

4- **No delta iteration:** - it is not so efficient for iterative operation.

5- **Latency:** - In MapReduce, Map takes a set of data and converts it into another set of data, where individual element is broken down into key value pair and Reduce takes the output from the map as input and process further and MapReduce requires a lot of time to perform these tasks thereby increasing latency.

6- **Not easy to use:** - MapReduce developers need to hand code for each and every operation which makes it very difficult to work. MapReduce has no interactive mode, but adding one such as hive and pig makes working with MapReduce a little easier for adopters.

7- **No caching:** - MapReduce cannot cache the intermediate data in memory for a further requirement which diminishes the performance of Hadoop.

Qus: What is RDD? Explain new features of RDD?

Ans: RDD (Resilient Distributed Dataset) is the fundamental data structure of Apache spark which are an immutable collection of objects which computes on the different node of the cluster. Each and every dataset in Spark RDD is logically partitioned across many servers so that they can be computed on different nodes of the cluster.
Decomposing the name RDD:

- **Resilient**, i.e. fault-tolerant with the help of RDD lineage graph(**DAG**) and so able to recompute missing or damaged partitions due to node failures.
- **Distributed**, since Data resides on multiple nodes.
- **Dataset** represents records of the data you work with. The user can load the data set externally which can be either JSON file, CSV file, text file or database via JDBC with no specific data structure.

Features of RDD: there are several features:

1- **In memory computation:** it stores intermediate results in distributed memory(RAM) instead of stable storage(disk).
2- **Lazy Evolution:** they do not compute their results right away. Instead, they just remember the transformations applied to some base data set.
3- **Fault tolerance:** RDDs are fault tolerant as they track data lineage information to rebuild lost data automatically on failure.

4- **Immutability:** Data is safe to share across processes. It can also be created or retrieved anytime which makes caching, sharing & replication easy.

5- **Partitioning:** Partitioning is the fundamental unit of parallelism in RDD. Each partition is one logical division of data which is mutable. One can create a partition through some transformations on existing partitions.

Qus: List down few Spark RDD operations and explain each of them?

Ans: Apache Spark RDD supports two types of Operations-

- Transformations
- Actions

Transformations: Spark Transformation is a function that produces new RDD from the existing RDDs. It takes RDD as input and produces one or more RDD as output. Each time it creates new RDD when we apply any transformation. Thus, the so input RDDs, cannot be changed since RDD are immutable in nature.

There are two types of transformations:

1- **Narrow transformation –** In Narrow transformation, all the elements that are required to compute the records in single partition live in the single partition of parent RDD. A limited subset of partition is used to calculate the result. Narrow transformations are the result of map(), filter().

2- **Wide transformation –** In wide transformation, all the elements that are required to compute the records in the single partition may live in many partitions of parent RDD. The partition may live in many partitions of parent RDD. Wide transformations are the result of groupbyKey() and reducebyKey().

I. **Map(func):** The map function iterates over every line in RDD and split into new RDD. Using **map()** transformation we take in any function, and that function is applied to every element of RDD.

II. **flatMap():** With the help of flatMap() function, to each input element, we have many elements in an output RDD. The most simple use of flatMap() is to split each input string into words.

III. **filter(func):** it returns a new RDD, containing only the elements that meet a predicate. It is a narrow *operation* because it does not shuffle data from one partition to many partitions.

IV. **MapPartitions(func):** The MapPartition converts each *partition* of the source RDD into many elements of the result (possibly none). In mapPartition(), the map() function is applied on each partitions simultaneously. MapPartition is like a map, but the difference is it runs separately on each partition(block) of the RDD.

V. **Union(dataset):** With the union() function, we get the elements of both the RDD in new RDD. The key rule of this function is that the two RDDs should be of the same type.

VI. **Intersection(dataset):** With this function, we get only the common element of both the RDD in new RDD. The key rule of this function is that the two RDDs should be of the same type.