1. **What is RDD?**

**Resilient Distributed Dataset** are the primary data abstraction in Apache Spark and the core of Spark that is referred to as "Spark Core".

Features:

* **Resilient**, i.e. fault-tolerant with the help of [RDD lineage graph](https://jaceklaskowski.gitbooks.io/mastering-apache-spark/content/spark-rdd.html#lineage) and so able to recompute missing or damaged partitions due to node failures.
* **Distributed** with data residing on multiple nodes in a [cluster](https://jaceklaskowski.gitbooks.io/mastering-apache-spark/content/spark-cluster.html).
* **Dataset** is a collection of [partitioned data](https://jaceklaskowski.gitbooks.io/mastering-apache-spark/content/spark-rdd-partitions.html) with primitive values or values of values, e.g. tuples or other objects (that represent records of the data you work with).

RDDs are Immutable and partitioned collection of records, which can only be created by coarse grained operationssuch as map, filter, group by etc. By coarse grained operations , it means that the operations are applied on all elements in a datasets. RDDs can only be created by reading data from a stable storage such as HDFS or by transformations on existing RDDs.

1. **Define Partitions.**

A partition/split is a logical chunk of a large distributed data set.

In Spark it is an atomic chunk of data it works with at a given time.

A partition is also a chunk of data that's stored on one node in the Cluster/Cloud.

Spark manages data using partitions that helps parallelize distributed data processing with minimal network traffic for sending data between executors.

**3. What operations does RDD support?**

* Create - from stable storage(HDFS)
* Transform -
* Generate RDD from other RDD (map, filter, groupBy)
* Lazy operations that builds a DAG (Directed Acyclic Graph)
* Once Spark knows our transformations, it starts building an efficient plan.
* Action - returns a result or write to storage (count, collect, reduce, save)

**4.What do you understand by Transformations in Spark?**

**Transformations** are lazy operations on a RDD that create one or many new RDDs.

In other words, transformations are functions that take a RDD as the input and produce one or many RDDs as the output. They do not change the input RDD (since [RDDs are immutable](https://jaceklaskowski.gitbooks.io/mastering-apache-spark/content/spark-rdd.html#introduction) and hence cannot be modified), but always produce one or more new RDDs by applying the computations they represent.

By applying transformations you incrementally build a [RDD lineage](https://jaceklaskowski.gitbooks.io/mastering-apache-spark/content/spark-rdd-lineage.html) with all the parent RDDs of the final RDD(s).

Transformations are lazy, i.e. are not executed immediately. Only after calling an action are transformations executed.

**5.Define Actions.**

Actions are [RDD operations](https://jaceklaskowski.gitbooks.io/mastering-apache-spark/content/spark-rdd-operations.html) that produce non-RDD values. They materialize a value in a Spark program. They trigger execution of [RDD transformations](https://jaceklaskowski.gitbooks.io/mastering-apache-spark/content/spark-rdd-actions.html#transformations) to return values. Simply put, an action evaluates the [RDD lineage graph](https://jaceklaskowski.gitbooks.io/mastering-apache-spark/content/spark-rdd-lineage.html).

You can think of actions as a valve and until action is fired, the data to be processed is not even in the pipes, i.e. transformations. Only actions can materialize the entire processing pipeline with real data.

Actions are one of two ways to send data from [executors](https://jaceklaskowski.gitbooks.io/mastering-apache-spark/content/spark-Executor.html) to the [driver](https://jaceklaskowski.gitbooks.io/mastering-apache-spark/content/spark-driver.html) (the other being [accumulators](https://jaceklaskowski.gitbooks.io/mastering-apache-spark/content/spark-accumulators.html)).