Understanding RDDs, DataFrames and Datasets

1. What does RDD stand for?

RDD stands for Resilient Distributed Dataset.

It is the core data structure of Apache Spark. RDD is an immutable distributed collection of objects that can be processed in parallel. It is fault-tolerant, distributed across the nodes in a cluster, and supports in-memory computations.

Key properties of RDD:

- Resilient: Fault-tolerant using lineage
- Distributed: Data is split across the cluster
- Dataset: Collection of objects

2. How does a DataFrame differ from an RDD?

A DataFrame is a distributed collection of data organized into named columns, similar to a table in a relational database or a DataFrame in Python's pandas.

Differences between DataFrame and RDD:

- **Schema**: DataFrame has a schema (columns with names and types), RDD does not.
- **Ease of Use**: DataFrames provide a higher-level API with SQL-like operations, whereas RDDs require functional programming.
- **Optimization**: DataFrames are optimized by Catalyst optimizer and Tungsten engine, while RDDs lack such optimization.
- **Performance**: DataFrames are generally faster due to optimization and better memory usage.

Understanding RDDs, DataFrames and Datasets

3. What is the role of a Dataset in Spark?

A Dataset is a distributed collection of data that combines the benefits of RDDs (type safety, object-oriented programming) and DataFrames (optimized execution).

- Introduced in Spark 1.6 (in Java and Scala only)
- Provides compile-time type safety
- Allows use of custom classes and functional transformations
- Uses Catalyst optimizer and Tungsten engine for performance

Datasets are ideal when you want both strong typing and optimized performance.