PySpark Comprehensive Guide

➢ Introduction to PySpark

PySpark is the Python API for Apache Spark, an open-source, distributed computing system used for big data processing. It enables the parallel processing of data and is known for its speed and ease of use

➢ SparkContext

Definition: The primary entry point for Spark functionality.

Role: Handles connections to the cluster manager, distributed storage, and cluster resource management.

➢ SparkSession

Definition: Introduced in Spark 2.0, it encapsulates SparkContext and is the entry point for DataFrame and Dataset API.

Usage: Simplifies the codebase by replacing the need for multiple contexts (SQLContext, HiveContext, etc.).

➢ RDD (Resilient Distributed Dataset)

Definition: The core data structure, representing a fault-tolerant, distributed collection of elements.

Properties:

- Once created, it cannot be altered.
- Operations are not executed immediately; instead, they are queued until an action is called.
- Capable of recovering from node failures.

```
rdd = sc.parallelize([1, 2, 3, 4, 5])
```

➢ DataFrame

Definition: A distributed collection of data organized into named columns, similar to a table in a relational database.

Features:

- Leverages Catalyst for query optimization and Tungsten for efficient execution.
- DataFrames have a schema, making it easy to understand the data structure.
- Provides higher-level abstraction over RDDs for structured data.

```
df = spark.read.csv("/content/btw.csv", header=True, inferSchema=True)
```

➢ Dataset

Definition: A combination of RDD and DataFrame, offering both strong typing and the benefits of the Catalyst optimizer.

Type-Safe Operations: Enables compile-time type checks, which helps catch errors early in the development process.

Usage: Typically, Datasets are more prevalent in Scala and Java APIs, but in PySpark, DataFrames often serve a similar role.