**SUMMARY-DAY11**

**Name:**Tejaswini Gokanakonda

**Roll no:**DE142

**Date:**20-11-2024

**Functions of Spark**

**DataFrames Basics**

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

python

df = spark.read.load("file.csv", format="csv", sep=",", inferSchema="true", header="true")

* + format: Specifies the file type (e.g., CSV, Parquet).
  + header: Indicates if the file contains column headers.
  + inferSchema: Infers data types automatically.
* **Viewing Data**:
  + .show(): Displays rows in a tabular format.
  + .toPandas(): Converts a Spark DataFrame to a Pandas DataFrame for better visualization.

**Data Manipulations in DataFrames**

1. **Rename Columns**:

Python

df = df.withColumnRenamed("old\_name", "new\_name")

1. **Select Columns**:

Python

df.select("col1", "col2").show()

1. **Sorting**:
   * Ascending order:

python

df.sort("col\_name").show()

* + Descending order:

python

from pyspark.sql import functions as F

df.sort(F.desc("col\_name")).show()

1. **Filtering Rows**:

Python

df.filter((df.col1 > value) & (df.col2 == "value")).show()

1. **GroupBy and Aggregations**:
   * Perform grouping and aggregate operations:

Python

from pyspark.sql import functions as F

df.groupBy("col1").agg(F.sum("col2"), F.max("col3")).show()

**Joins in Spark**

* Joins combine datasets based on common columns:

python

df1.join(df2, ["common\_col"], how="inner").show()

* + Types of joins: inner, left, right, outer.

**Using SQL with DataFrames**

* Register a DataFrame as a SQL table:

python

df.createOrReplaceTempView("table\_name")

result = spark.sql("SELECT \* FROM table\_name WHERE col > 100")

result.show()

**Adding or Modifying Columns**

* Use .withColumn() to create or transform columns:

python

from pyspark.sql import functions as F

df = df.withColumn("new\_col", F.col("existing\_col") \* 2)

**2. Transforming Data with PySpark RDDs**

**RDD Basics**

* **Definition**: RDD (Resilient Distributed Dataset) is the foundational data structure in Spark. It is immutable and distributed across a cluster.
* **Creation**:

python

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

**Transformations in RDDs**

* **Definition**: Transformations are operations that produce a new RDD from an existing one. They are lazy and execute only when an action is performed.

1. **map()**:
   * Applies a function to each element:

python

rdd.map(lambda x: x + 1).collect()

1. **filter()**:
   * Filters elements based on a condition:

python

rdd.filter(lambda x: x % 2 == 0).collect()

1. **flatMap()**:
   * Similar to map(), but flattens the output:

python

rdd.flatMap(lambda x: x.split(" ")).collect()

1. **union()**:
   * Combines two RDDs:

python

rdd1.union(rdd2).collect()

**Actions in RDDs**

* **Definition**: Actions trigger the execution of transformations and return results to the driver program.

1. **collect()**:
   * Returns all elements in the RDD as a Python list:

python

rdd.collect()

1. **count()**:
   * Counts the number of elements:

python

rdd.count()

1. **first()**:
   * Returns the first element:

python

rdd.first()

1. **take(n)**:
   * Returns the first n elements:

python

rdd.take(3)

1. **reduce()**:
   * Aggregates elements using a lambda function:

python

rdd.reduce(lambda x, y: x + y)

1. **saveAsTextFile()**:
   * Saves the RDD to a text file:

python

rdd.saveAsTextFile("output\_dir")

**3. Working with Pair RDDs**

**Overview**

* **Definition**: Pair RDDs store data as key-value pairs, ideal for operations like grouping, aggregation, and sorting.

**Transformations in Pair RDDs**

1. **reduceByKey()**:
   * Aggregates values by key:

python

pair\_rdd.reduceByKey(lambda x, y: x + y).collect()

1. **sortByKey()**:
   * Sorts the RDD by keys:

python

pair\_rdd.sortByKey().collect()

1. **groupByKey()**:
   * Groups values with the same key:

python

pair\_rdd.groupByKey().mapValues(list).collect()

**Actions in Pair RDDs**

1. **countByKey()**:
   * Counts occurrences of each key:

python

pair\_rdd.countByKey()

**4. Key Advantages of Spark**

* **Speed**: Fast in-memory processing.
* **Scalability**: Handles large datasets across clusters.
* **Flexibility**: Supports batch, streaming, and real-time data processing.
* **Integration**: Works seamlessly with Hadoop, Hive, and other big data tools.