ASSESSMENT

NAME-SWARNA LATHA GUVVALA

DATE-23/12/2023

PYSPARK COMMANDS-

1. The pyspark command is used to launch **Spark with Python** shell also call PySpark.
2. Use spark-shell command to work **Spark with Scala**.
3. PySpark shell default provides spark and sc variables. spark is an object of SparkSession and sc is an object of SparkContext.
4. In PySpark shell, you cannot create your own SparkContext.

PYSPARK SQL MODULE-

Some important classes of Spark SQL and DataFrames are the following:

- **pyspark.sql.SparkSession:** It represents the main entry point for **DataFrame** and SQL functionality.
- **pyspark.sql.DataFrame:** It represents a distributed collection of data grouped into named columns.
- **pyspark.sql.Column:** It represents a column expression in a **DataFrame.**
- **pyspark.sql.Row:** It represents a row of data in a **DataFrame.**
- **pyspark.sql.GroupedData:** Aggregation methods, returned by **DataFrame.groupBy().**
- **pyspark.sql.DataFrameNaFunctions:** It represents methods for handling missing data (null values).
- **pyspark.sql.DataFrameStatFunctions:** It represents methods for statistics functionality.
- **pysark.sql.functions:** It represents a list of built-in functions available for **DataFrame.**
- **pyspark.sql.types:** It represents a list of available data types.
- **pyspark.sql.Window:** It is used to work with Window functions.

INITIALIZING SPARKSESSION:

PySpark & Spark SQL Spark SQL is Apache Spark's module for working with structured data. A SparkSession can be used create DataFrame, register DataFrame as tables, execute SQL over tables, cache tables, and read parquet files.

```
>>> from pyspark.sql import SparkSession
 >>> spark = SparkSession \
              .builder \
              .appName( ) \
              .config( , ) \
              .getOrCreate() "Python Spark SQL basic example"
"spark.some.config.option" "some-value
```

INSPECT SPARKCONTEXT:

```
>>> sc.version #Retrieve SparkContext version
>>> sc.pythonVer #Retrieve Python version
>>> sc.master #Master URL to connect to
>>> str(sc.sparkHome) #Path where Spark is installed on worker nodes
>>> str(sc.sparkUser()) #Retrieve name of the Spark User running SparkContext
>>> sc.appName #Return application name
>>> sc.applicationId #Retrieve application ID
>>> sc.defaultParallelism #Return default level of parallelism
>>> sc.defaultMinPartitions #Default minimum number of partitions for RDDs
```

RETRIEVING RDD INFORMATION:

BASIC INFORMATION:

```
>>> rdd.getNumPartitions() #List the number of partitions
>>> rdd.count() #Count RDD instances 3
```

```
>>> rdd.countByKey() defaultdict(,{ :2, :1}) #Count RDD instances by value
```

```
>>> rdd.countByValue() defaultdict(,{( ,2):1,( ,2):1,( ,7):1}) #Return (key,value)
pairs as a dictionary
```

```
>>> rdd.collectAsMap() { : 2, : 2} >>> rdd3.sum() #Sum of RDD elements 4950
```

```
>>> sc.parallelize([]).isEmpty() #Check whether RDD is empty
```

INSPECTING DATA:

```
>>> df.dtypes              -- Returns df column names and data types
>>> df.show()                 -- Displays the content of df
>>> df.head()                 -- Returns first n rows
>>> df.first(n)              -- Returns the first n rows
>>> df.schema                 -- Returns the schema of df
>>> df.describe().show()        -- Computes the summary statistics
>>> df.columns                -- Returns the columns of df
>>> df.count()                -- Counts the number of rows in df
>>> df.distinct().count()       -- Counts the number of distinct rows in df
>>> df.printSchema()            -- Prints the schema of df
>>> df.explain()              -- Prints the (logical and physical) plans
```