PR-7 BDA

Finding the average number of friends by age.

Input Data:
ID, name, age, number of friends
0,Will,33,385
1,Jean-Luc,33,2
2,Hugh,55,221
3,Deanna,40,465
4,Quark,68,21

For our task of finding the average number of friends by age, let us take an example: What is the average number of friends for the average 33 year old in our data set? The code for the same would look like below-

```
from pyspark import SparkConf, SparkContext
conf = SparkConf().setMaster("local").setAppName("FriendsByAge")
sc = SparkContext(conf = conf)
```

1. Importing Necessary Modules: from pyspark import SparkConf, SparkContext
Imports the required modules from PySpark, which is a Python API for Apache Spark.

2. Configuring the SparkContext: we create a SparkConf object to set configurations for the Spark application.

- We set the master node to "local" which means Spark will run locally on one machine.
- We set the application name to "FriendsByAge".
- Then, we create a SparkContext (sc) using the configuration.

conf = SparkConf().setMaster("local").setAppName("FriendsByAge")
sc = SparkContext(conf = conf)

3. Method to parse each Line: This function parseLine() takes a line of input as a parameter, splits it by commas (as it's a CSV file as shown above), extracts the age (as it's at index 2) and the number of friends (assuming it's at index 3), converts them to integers, and returns a tuple of (age, numFriends).

```
def parseLine(line):
 fields = line.split(',')
 age = int(fields[2])
 numFriends = int(fields[3])
 return (age, numFriends)
```

4. Loading Data from File: This line reads the text file containing the data. It creates an RDD (Resilient Distributed Dataset) named lines, where each element of the RDD represents a line from the text file.

lines = sc.textFile("dbfs:/FileStore/tables/socialCircle.csv")

5. Mapping the Data: This line applies the parseLine() function to each element (line) of the RDD lines.

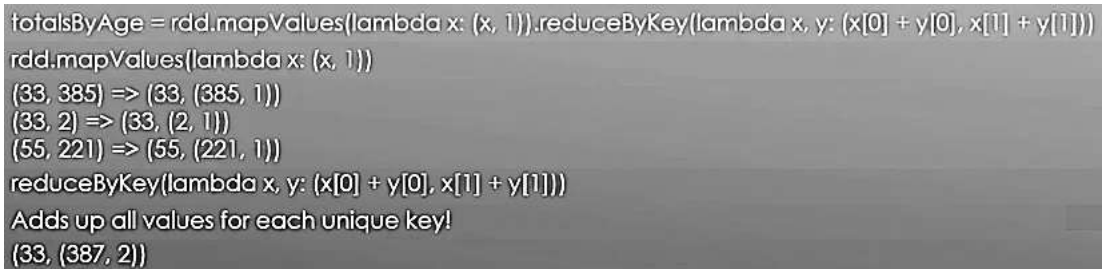
- It transforms each line of text into a tuple (age, numFriends).
- Here, Spark transformations like map() create a new RDD as a result. So, when you apply the map() operation on the lines RDD using parseLine function, It results in a new RDD named rdd.
- **Important concept:** Here, rdd is a key-value pair RDD where the keys are "**age"** and the values are the "**number of friends"**.

rdd = lines.map(parseLine)

6. Calculating Totals by Age: This line performs two operations:-

- First, it maps each value (age, numFriends) to (age, (numFriends, 1))
- Then, it reduces by key (age) by summing the values of friends and the count of records for each age.
- Note here that, totalsByAge is an RDD as well which has resulted from the transformation operations applied to the rdd RDD. It contains key-value pairs where each key is an "**age"**, and each value is a tuple (totalFriends, count).
-

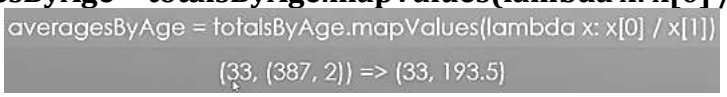
totalsByAge = rdd.mapValues(lambda x: (x, 1)).reduceByKey(lambda x, y: (x[0] + y[0], x[1] + y[1]))



7. Calculating Averages by Age: This line maps each value (age, (totalFriends, count)) to (age, totalFriends / count), which calculates the average number of friends for each age.

- In this line, totalsByAge is an RDD resulting from the previous transformation operations. It contains key-value pairs where each key is an age, and each value is a tuple (totalFriends, count).
- Now, the mapValues() transformation is applied to totalsByAge. It applies the provided lambda function (lambda x: x[0] / x[1]) to each value (tuple) in the key-value pairs. This lambda function calculates the average number of friends for each age by dividing the total number of friends by the count of records for that age.
- Thus, the RDD averagesByAge is contains the key-value pairs where each key is an "**age"**, and each value is the "**average number of friends"** for that age.

averagesByAge = totalsByAge.mapValues(lambda x: x[0] / x[1])



8. Collecting Results: This line collects all the results from the RDD averagesByAge and stores them in the variable results.

- In this line, collect() is an action that triggers the execution of all the previous transformations on the RDD averagesByAge. It collects all the elements of the RDD averagesByAge from the distributed nodes in the Spark cluster and brings them back to the driver program as a local Python list.
- So, results is a Python list that contains the collected results from the RDD averagesByAge. Each element in results is a tuple representing an age and its

corresponding average number of friends. This list is now available for further processing or analysis within the Python program.

results=averagesByAge.collect()

9. Printing Results: This loop iterates over the collected results and prints them. Each result is a tuple containing an age and its corresponding average number of friends.
For result in results: print(result)
The output of the above program looks something like this:-

```
(33, 325.3333333333333)
(26, 242.05882352941177)
(55, 295.53846153846155)
(40, 250.8235294117647)
(68, 269.6)
(59, 220.0)
(37, 249.33333333333334)
(54, 278.0769230769231)
(38, 193.53333333333333)
(27, 228.125)
(53, 222.85714285714286)
```

CODE:

```
def parseLine(line):
    fields = line.split(',')
    age = int(fields[2])
    numFriends = int(fields[3])
    return (age, numFriends)
lines = sc.textFile("dbfs:/FileStore/tables/avgfriends.csv")
rdd = lines.map(parseLine)
totalsByAge = rdd.mapValues(lambda x: (x, 1)).reduceByKey(lambda x, y: (x[0] + y[0], x[1]
+ y[1]))
averagesByAge = totalsByAge.mapValues(lambda x: x[0] / x[1])
results = averagesByAge.collect()
for result in results:
    print(result)
```