Apache Spark

“In “God we trust, all others must bring data.”

-W. Edwards Deming

Dataset :

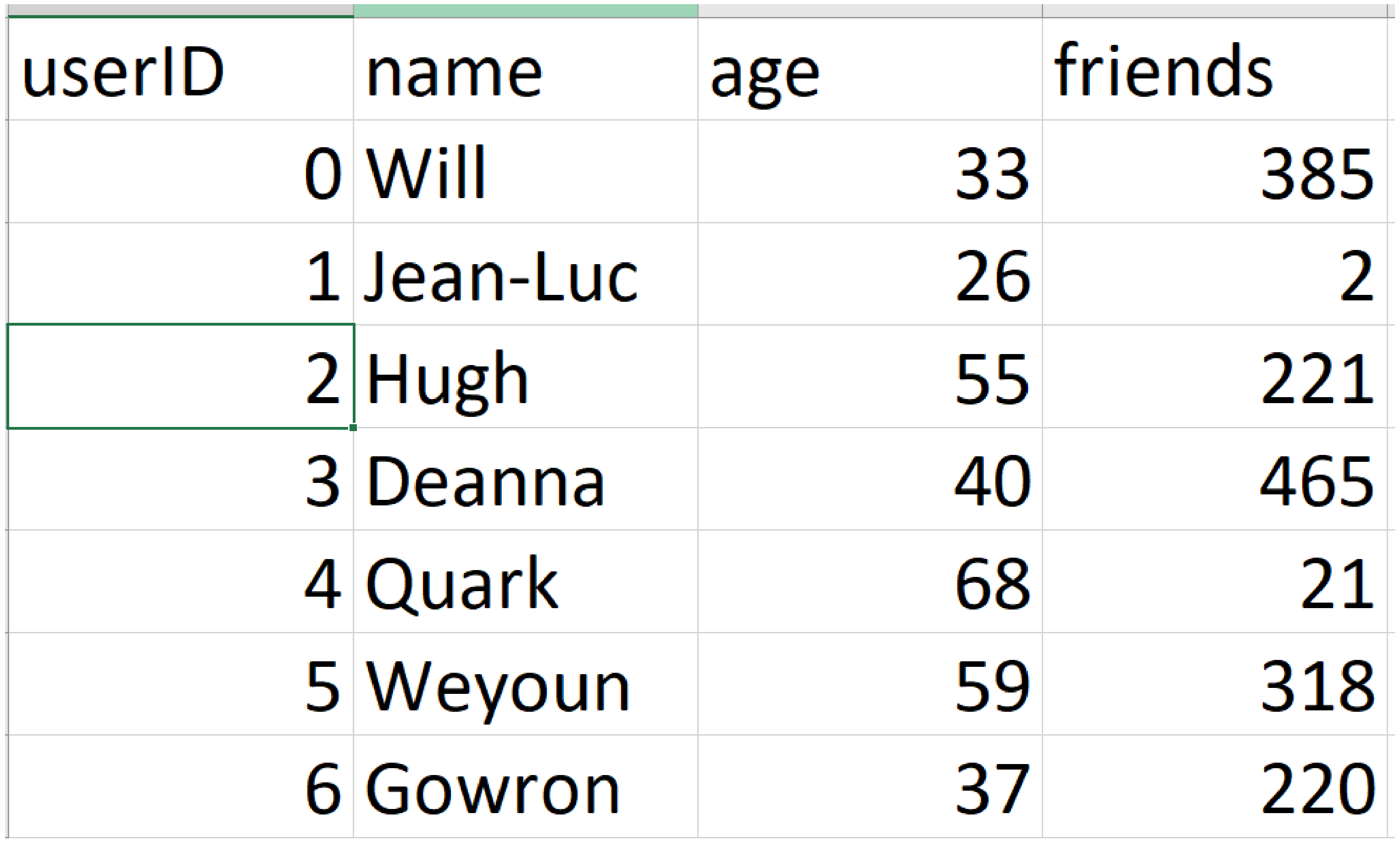
FAKEFRIENDS.CSV

WHAT IS IN IT ? LET'S SEE !!

Code :

```
from pyspark import SparkConf, SparkContext

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

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

lines = sc.textFile("/FileStore/tables/fakefriends.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)
```

Dataset :



userID	name	age	friends
0	Will	33	385
1	Jean-Luc	26	2
2	Hugh	55	221
3	Deanna	40	465
4	Quark	68	21
5	Weyoun	59	318
6	Gowron	37	220

Spark Functions:

01

redeceBykey(): combine values with same keys

02

GroupBykey(): group values with the same keys

03

SortBykey(): sort RDD by key values.

Setting up the spark configurations:

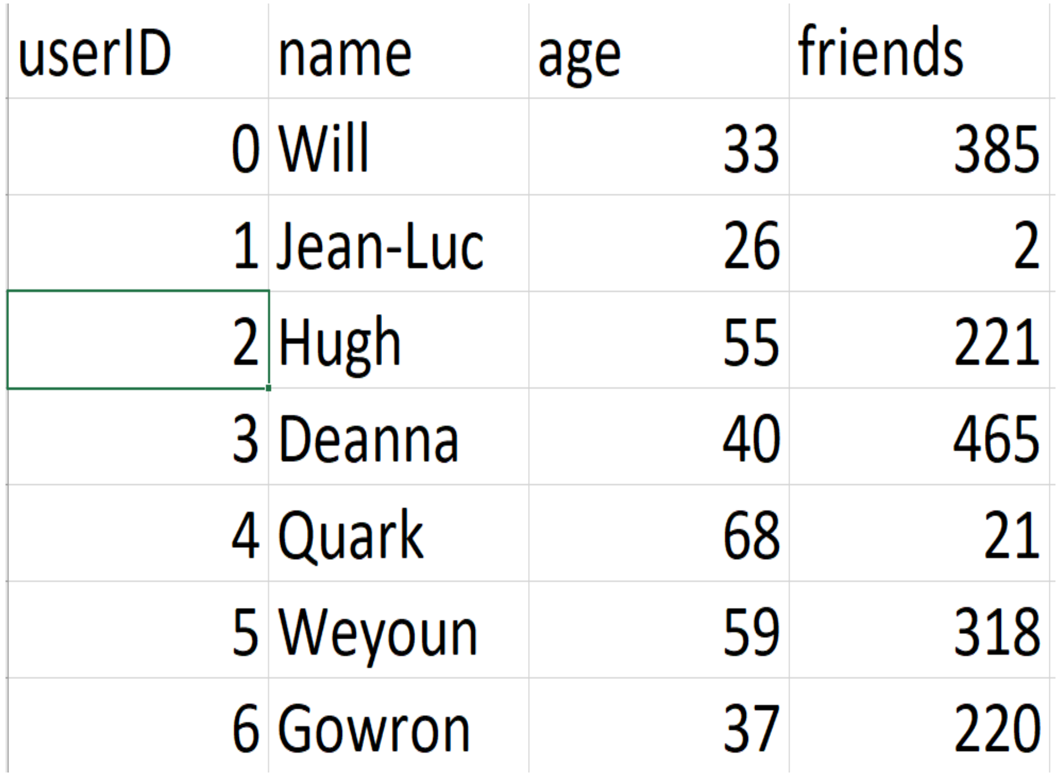
```
from pyspark import SparkConf, SparkContext

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

Code:

- *lines = sc.textFile("file:///Spark/fakefriends.csv")*

- *rdd = lines.map(parseLine)*

Code:



userID	name	age	friends
0	Will	33	385
1	Jean-Luc	26	2
2	Hugh	55	221
3	Deanna	40	465
4	Quark	68	21
5	Weyoun	59	318
6	Gowron	37	220

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

Output:

```
• 33,385
  33,2
  55,221
  40,465
……….
```

Code:

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

Line-1:

rdd.mapValues(lambda x: (x, 1)).

rdd.mapValues(lambda x: (x, 1))

- (33,385) => (33,(385,1))
- (33,2) => (33,(2,1))
- (55,221) => (55,(221,1))

reduceByKey(lambda x, y: (x[0] + y[0], x[1] + y[1]))

- Adds up all values for each unique key!
- (33,(387,2))

Code:

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

- (33,(387,2))=> (33,193.55)

Code:

```
results = averagesByAge.collect()
for result in results:
print(result)
```

Apache Spark :

- Nothing happens until we call action “reducebykey” in above program.

- Collect() is another action.

Computing an Average: (scala)

Computing an Average: (scala)

```
var rdd = sc.parallelize(array(1.0,2,3,4,5,6,7),3);

var rdd_count = rdd.map((_,1))

var(sum,count)=
rdd_count.reduce((x,y)=>x._1+y._1,x._2+y._2)

var avg  = sum/count
```

Q-1

- The metadata is stored on

A. Datanode

B. Namenode

Q-2

- Which of the following is not a metadata of file?

A. Name of the file

B. Folder name of the file

C. Permission attributes of the file

D. The contents of the file

Q-3

- Does datanode know the name and the parent folder name of the file?

A. Yes, of course

B. No, because name and foldername is the metadata

Q-4

- Which of the following is not true about replication in HDFS:

A. The default replication factor of is 3

B. The default replication factor can be specified in settings

C. We can change the default replication factor per file

D. The HDFS automatically decides the replication factor based on the demand of the file

Q-5

- What is not true about Apache spark?

a) It is fast large scale engine for data processing

b) Spark is faster than Apache Hadoop

c) It provides fast in-memory NoSQL datastore

d) It also has map-reduce paradigm

e) None

Q-6

- If you need to process data continuously, which library you would use:

A. MLlib

B. GraphX

C. Spark Streaming

D. SparkR