SPARK ASSIGNMENT

BIG DATA ANALYTICS

Performed on Red Hat (64 bit) CentOS 2019

Submitted To: Prof. Aditya Bharadwaj
Big Data Analytics

Submitted By: SID - 16103104
Name - Lovedeep Singh
BTech- CSE 4th Year

Ensure that **Spark** is setup in the linux system you are operating on.

Note: All these commands have been executed on **Red Hat (64 - bit) CentOS**.

Start the terminal and run *spark-shell* command

```
To adjust logging level use sc.setLogLevel(newLevel).
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/lib/zookeeper/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBi
SLF4J: Found binding in [jar:file:/usr/lib/flume-ng/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBin
SLF4J: Found binding in [jar:file:/usr/lib/parquet/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBind
SLF4J: Found binding in [jar:file:/usr/lib/avro/avro-tools-1.7.6-cdh5.13.0.jar!/org/slf4j/impl/StaticLoggerBind
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 1.6.0
      /_/

Using Scala version 2.10.5 (Java HotSpot(TM) 64-Bit Server VM, Java 1.7.0_67)
Type in expressions to have them evaluated.
Type :help for more information.
19/11/03 04:26:28 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using b
Spark context available as sc (master = local[*], app id = local-1572783992196).
19/11/03 04:26:37 WARN shortcircuit.DomainSocketFactory: The short-circuit local reads feature cannot be used b
SQL context available as sqlContext.

scala>
```

Before proceeding further, ensure that mobile user.csv is in hdfs, use the following commands for this:

hadoop fs -put mobileusers.csv /user/cloudera

Now load the file using the command below:

val mobiledt = sc.textFile("/user/cloudera/mobileusers.csv")

```
scala> val mobiledt = sc.textFile("/user/cloudera/mobileusers.csv")
mobiledt: org.apache.spark.rdd.RDD[String] = /user/cloudera/mobileusers.csv MapPartitionsRDD[5
] at textFile at <console>:27
```

We shall do some preprocessing before proceeding further

val header = mobiledt.take(1)(0)

```
scala> val header = mobiledt.take(1)(0)
header: String = state,account length,area code,phone number,international plan,voice mail pla
n,number vmail messages,total day minutes,total day calls,total day charge,total eve minutes,t
otal eve calls,total eve charge,total night minutes,total night calls,total night charge,total
 intl minutes,total intl calls,total intl charge,customer service calls
```

val mobiled = mobiledt.filter(x=>x != header)

```
scala> val mobiled = mobiledt.filter(x=>x != header)
mobiled: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[6] at filter at <console>:31
```

Now *mobiled* contains data which we will use in the questions

Q1. Verify that state consist of two character only.

Ans : Run the following command

mobiled.map(x=>x.split(“,”)).filter(x=>x(0).length()==2).count==mobiled.count

```
scala> mobiled.map(x=>x.split(",")).filter(x=>x(0).length()==2).count == mobiled.count
res10: Boolean = true
```

Q2. Find out maximum number of vmail messages for states OH, OK, and RI.

Ans : For state OH alone

mobiled.map(x=>x.split(“,”)).filter(x=>x(0)==“OH”).map(x=>x(6).toInt).max

```
scala> mobiled.map(x=>x.split(",")).filter(x=>x(0)=="OH").map(x=>x(6).toInt).max
res18: Int = 46
```

For state OK alone

mobiled.map(x=>x.split(“,”)).filter(x=>x(0)==“OK”).map(x=>x(6).toInt).max

```
scala> mobiled.map(x=>x.split(",")).filter(x=>x(0)=="OK").map(x=>x(6).toInt).max
res20: Int = 43
```

For state RI alone

mobiled.map(x=>x.split(“,”)).filter(x=>x(0)==“RI”).map(x=>x(6).toInt).max

```
scala> mobiled.map(x=>x.split(",")).filter(x=>x(0)=="RI").map(x=>x(6).toInt).max
res23: Int = 43
```

For all OH, OK and RI together

mobiled.map(x=>x.split(“,”)).filter(x=>x(0)==“OH” || x(0)==“OK” || x(0)==“RI”).map(x=>x(6).toInt).max

```
scala> mobiled.map(x=>x.split(",")).filter(x=>x(0)=="OH" || x(0)=="OK" || x(0)=="RI").map(x=>x
(6).toInt).max
res32: Int = 46
```

Q3. Find out the sum of total day calls for customers who belong to OH and NY State.

Ans : For state OH alone

mobiled.map(x=>x.split(“,”)).filter(x=>x(0)==“OH”).map(x=>x(8).toDouble).sum

```
scala> mobiled.map(x=>x.split(",")).filter(x=>x(0)=="OH").map(x=>x(8).toDouble).sum
res33: Double = 7771.0
```

For state NY alone

mobiled.map(x=>x.split(“,”)).filter(x=>x(0)==“OH”).map(x=>x(8).toDouble).sum

```
scala> mobiled.map(x=>x.split(",")).filter(x=>x(0)=="NY").map(x=>x(8).toDouble).sum
res34: Double = 8154.0
```

For both states taken together

mobiled.map(x=>x.split(“,”)).filter(x=>x(0)==“OH” || x(0)==“NY”).map(x=>x(8).toDouble).sum

```
scala> mobiled.map(x=>x.split(",")).filter(x=>x(0)=="OH" || x(0)=="NY").map(x=>x(8).toDouble).
sum
res35: Double = 15925.0
```

Q4. Verify that international field contains yes and no entries.

Ans : Run the following command

mobiled.map(x=>x.split(“,”)).filter(x=>x(4)==“yes” || x(4)==“no”).count==mobiled.count

```
scala> mobiled.map(x=>x.split(",")).filter(x=>x(4)=="yes" || x(4)=="no").count == mobiled.count
res39: Boolean = true
```

Q5. How many customers have both international and voice mail plan.

Ans : Run the following command

mobiled.map(x=>x.split(“,”)).filter(x=>x(4)==“yes” && x(5)==“yes”).count

```
scala> mobiled.map(x=>x.split(",")).filter(x=>x(4)=="yes" && x(5)=="yes").count
res48: Long = 92
```