Spark Scala

READY

```
%sh
hdfs dfs -mkdir -p /sparklab/

hdfs dfs -put /home/cloudera/Downloads/Baby_Names__Beginning_2007.csv /sparkl
```

READY

```
%sh
hdfs dfs -ls /sparklab
```

```
Found 8 items
-rw-r--r--   1 root supergroup    1428841 2022-06-07 07:13 /sparklab/5000-8.txt
-rw-r--r--   1 root supergroup    5657962 2022-06-07 07:17 /sparklab/Baby_Names__Beginning_2007.csv
-rw-r--r--   1 root supergroup      41082 2022-05-24 10:49 /sparklab/Stations_2019.csv
-rw-r--r--   1 root supergroup        210 2022-05-24 11:01 /sparklab/bikers.txt
-rw-r--r--   1 root supergroup        477 2022-05-24 11:01 /sparklab/exits.txt
-rw-r--r--   1 root supergroup       1117 2022-05-24 11:01 /sparklab/hikings.json
-rw-r--r--   1 root supergroup         92 2022-05-24 11:01 /sparklab/state.txt
-rw-r--r--   1 root supergroup    1979226 2022-05-24 11:05 /sparklab/u.data
```

READY

```
%spark

val textFile = sc.textFile("/sparklab/Baby_Names__Beginning_2007.csv")
```

```
textFile: org.apache.spark.rdd.RDD[String] = /sparklab/Baby_Names__Beginning_2007.csv MapPartitionsRDD[67] at textFile at <console>:25
```

READY

```
%spark

val rows = textFile.map(line => line.split(","))
```

```
rows: org.apache.spark.rdd.RDD[Array[String]] = MapPartitionsRDD[68] at map at <console>:27
```

READY

```
%spark

rows.collect
```

```
res18: Array[Array[String]] = Array(Array(Year, First Name, County, Sex, Count), Array(2013, GAVIN, ST LAWRENCE, M, 9), Array(2013, LEVI, ST LAWRENCE, M, 9), Array(2013, LOGAN, NEW YORK, M, 44), Array(2013, HUDSON, NEW YORK, M, 49), Array(2013, GABRIEL, NEW YORK, M, 50), Array(2013, THEODORE, NEW YORK, M, 51), Array(2013, ELIZA, KINGS, F, 16), Array(2013, MADELEINE, KINGS, F, 16), Array(2013, ZARA, KINGS, F, 16), Array(2013, DAISY, KINGS, F, 16), Array(2013, JONATHAN, NEW YORK, M, 51), Array(2013, CHRISTOPHER, NEW YORK, M, 52), Array(2013, LUKE, SUFFOLK, M, 49), Array(2013, JACKSON, NEW YORK, M, 53), Array(2013, JACKSON, SUFFOLK, M, 49), Array(2013, JOSHUA, NEW YORK, M, 53), Array(2013, AIDEN, NEW YORK, M, 53), Array(2013, BRANDON, SUFFOLK, M, 50), Array(2013, JUDY, KINGS, F, 16), Array...
```

READY

Note: *No need to remove the csv header as the Filter Expression will exclude it*

READY

```
%spark

val female = rows.filter(row => row(3).contains("F"))
```

```
female: org.apache.spark.rdd.RDD[Array[String]] = MapPartitionsRDD[71] at filter at <console>:29
```

READY

```
%spark

val male = rows.filter(row => row(3).contains("M"))
```

```
male: org.apache.spark.rdd.RDD[Array[String]] = MapPartitionsRDD[72] at filter at <console>:29
```

Spark Scala

```
%spark
female.count
```

READY

```
res19: Long = 124424
```

```
%spark
male.count
```

READY

```
res20: Long = 111086
```

```
%spark
female.collect
```

READY

```
res21: Array[Array[String]] = Array(Array(2013, ELIZA, KINGS, F, 16), Array(2013, MADELEINE, KINGS, F, 16), Array(2013, ZARA, KINGS, F, 16), Array(2013, DAISY, KINGS, F, 16), Array(2013, JUDY, KINGS, F, 16), Array(2013, DEVORA, KINGS, F, 16), Array(2013, YEHUDIS, KINGS, F, 16), Array(2013, SABRINA, KINGS, F, 15), Array(2013, LUNA, KINGS, F, 15), Array(2013, MILAN, KINGS, F, 15), Array(2013, DANIELLE, KINGS, F, 15), Array(2013, ISLA, KINGS, F, 15), Array(2013, PARIS, KINGS, F, 15), Array(2013, LOLA, KINGS, F, 15), Array(2013, NYLAH, KINGS, F, 15), Array(2013, HELEN, KINGS, F, 15), Array(2013, ADELE, KINGS, F, 15), Array(2013, SURI, KINGS, F, 15), Array(2013, ZISSY, KINGS, F, 15), Array(2013, YIDES, KINGS, F, 15), Array(2013, COLETTE, NEW YORK, F, 10), Array(2013, CAMILLA, NEW YORK, F, 10...
```

```
%spark
male.collect
```

READY

```
res22: Array[Array[String]] = Array(Array(2013, GAVIN, ST LAWRENCE, M, 9), Array(2013, LEVI, ST LAWRENCE, M, 9), Array(2013, LOGAN, NEW YORK, M, 44), Array(2013, HUDSON, NEW YORK, M, 49), Array(2013, GABRIEL, NEW YORK, M, 50), Array(2013, THEODORE, NEW YORK, M, 51), Array(2013, JONATHAN, NEW YORK, M, 51), Array(2013, CHRISTOPHER, NEW YORK, M, 52), Array(2013, LUKE, SUFFOLK, M, 49), Array(2013, JACKSON, NEW YORK, M, 53), Array(2013, JACKSON, SUFFOLK, M, 49), Array(2013, JOSHUA, NEW YORK, M, 53), Array(2013, AIDEN, NEW YORK, M, 53), Array(2013, BRANDON, SUFFOLK, M, 50), Array(2013, MASON, ST LAWRENCE, M, 8), Array(2013, DAVID, NEW YORK, M, 53), Array(2013, NOAH, ST LAWRENCE, M, 8), Array(2013, AIDEN, ST LAWRENCE, M, 8), Array(2013, SEBASTIAN, NEW YORK, M, 57), Array(2013, SAMUEL, NEW YORK...
```

```
%spark
// saveAsTextFile(path)
// Purpose: Writes the content of RDD to a text file or a set of text files t

// val femaleSplit = female.saveAsTextFile("baby_females")
// val maleSplit = male.saveAsTextFile("baby_males")
```

READY

```
%spark

// convert to csv
female.map(_.toList).map(_.mkString(",")).saveAsTextFile("/female_baby_names"
```

READY

```
%spark

// convert to csv
male.map(_.toList).map(_.mkString(",")).saveAsTextFile("/male_baby_names")
```

READY

```
%spark
```

READY