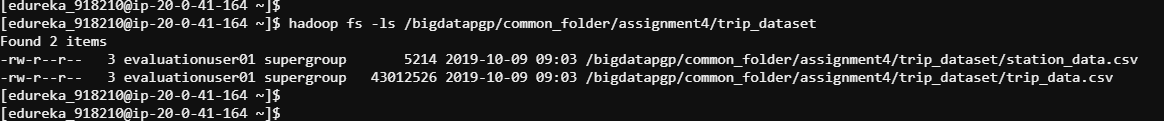
**TRIP DATASETS**

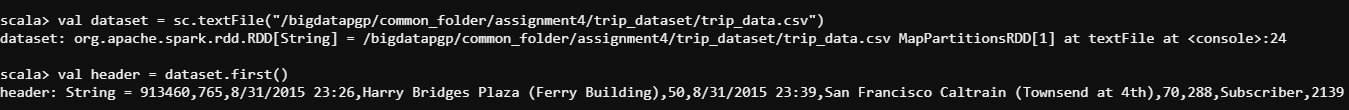
Please find the data stored in HDFS :

**/bigdatapgp/common\_folder/assignment4/trip\_dataset/**



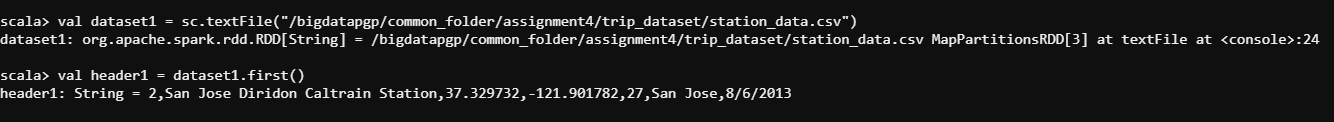
val dataset = sc.textFile("/bigdatapgp/common\_folder/assignment4/trip\_dataset/trip\_data.csv")

val header = dataset.first()



val dataset1 = sc.textFile("/bigdatapgp/common\_folder/assignment4/trip\_dataset/station\_data.csv")

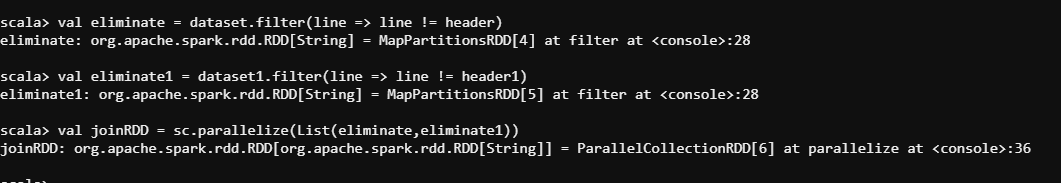
val header1 = dataset1.first()



val eliminate = dataset.filter(line => line != header)

val eliminate1 = dataset1.filter(line => line != header1)

val joinRDD = sc.parallelize(List(eliminate,eliminate1))



**Read the Station dataset and create an RDD. Now print the number of bike stations in each city.**

val arrange = split.reduceByKey(\_+\_).map(split => split.swap).groupby(x(11)).collect.foreach(println)

**Read the trip dataset and create an RDD**

val eliminate = dataset.filter(line => line != header)

val arrange = split.reduceByKey(\_+\_).map(split => split.swap).collect.foreach(println)

**Print the number of trips (bike rides) starting from San Jose.**

val eliminate = dataset.filter(line => line != header)

val arrange = split.reduceByKey(\_+\_).map(split => split.swap).filter(x(2)=='San Jose')collect.foreach(println)

**Find the city with max number of bike rides.**

val arrange = split.reduceByKey(\_+\_).map(split => split.swap).groupby(x(11)).max(count(x(9))).collect.foreach(println)

hdfs:///bigdatapgp/common\_folder/assignment4/trip\_dataset/

trip\_data.csv: This dataset contains the trip details

ID,Duration,Start\_Date,Start\_Station,Start\_Station\_Code,End\_Date,End\_Station,End\_Station\_Code,Bike\_Number,Subscriber,Zip\_Code

station\_data.csv: It contains geographical information about each bike station

Station\_Id,Start\_Station,Latitude,Longitude,Dock\_Count,Landmark,Installation

val dataset = sc.textFile("bigdatapgp/common\_folder/assignment4/trip\_dataset/trip\_data.csv")

val header = dataset.first()

val dataset1 = sc.textFile("bigdatapgp/common\_folder/assignment4/trip\_dataset/station\_data.csv")

val header1 = dataset1.first()

val eliminate = dataset.filter(line => line != header)

val eliminate1 = dataset1.filter(line => line != header1)

val joinRDD = sc.parallelize(List(eliminate,eliminate1))

Max duration of a bike trip .

Using accumulators find the number of rides for ’Subscriber ’ and ‘Customer’. Will this give better performance than grouping by subscriber\_type?

Find the number of times zip\_code is NULL using accumulators.

**Max duration of a bike trip .**

val arrange = split.reduceByKey(\_+\_).map(split => split.swap).groupby(x(11)).max(count(x(9))).collect.foreach(println)

Using accumulators find the number of rides for ’Subscriber ’ and ‘Customer’. Will this give better performance than grouping by subscriber\_type?

val arrange = split.reduceByKey(\_+\_).map(split => split.swap).groupby(x(10)).countBy(x(9))).collect.foreach(println)

**Find the number of times zip\_code is NULL using accumulators.**

val arrange = split.reduceByKey(\_+\_).map(split => split.swap).filter(x(11).equalto("NULL")).collect.foreach(println)