Bigdata Assignment 6.5

- Loaded the datasets into dataframes using RDD

val rdd = sc.textFile(“file:///home/acadgild/RITESH/Dataset_Holidays.txt”)

val holidayDF = rdd.map(x=>x.split(“,”)).map(array=>(array(0),array(1),array(2),array(3),array(4),array(5))).toDF(“id”,”src”,”dest”,”mode”,”dist”,”year”)

val rdd = sc.textFile(“file:///home/acadgild/RITESH/Dataset_Transport.txt”)

val transportDF = rdd.map(x=>x.split(“,”)).map(array=>(array(0),array(1))).toDF(“transport_name”,”transport_id”)

val rdd = sc.textFile(“file:///home/acadgild/RITESH/Dataset_User_details.txt”)

val userDF = rdd.map(x=>x.split(“,”)).map(array=>(array(0),array(1),array(2)).toDF(”person_id”,”name”,”age”)

```
scala> val rdd = sc.textFile("file:///home/acadgild/RITESH/Dataset_Holidays.txt")
rdd: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[92] at textFile at <console>:27

scala> val holidayDF = rdd.map(x=>x.split(",")).map(array=>(array(0),array(1),array(2),array(3),array(4),array(5))).toDF("id"
,"src","dest","mode","dist","year")
holidayDF: org.apache.spark.sql.DataFrame = [id: string, src: string, dest: string, mode: string, dist: string, year: string]

scala> val rdd = sc.textFile("file:///home/acadgild/RITESH/Dataset_Transport.txt")
rdd: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[97] at textFile at <console>:27

scala> val transportDF = rdd.map(x=>x.split(",")).map(array=>(array(0),array(1))).toDF("transport_name","transport_id")
transportDF: org.apache.spark.sql.DataFrame = [transport_name: string, transport_id: string]

scala> val rdd = sc.textFile("file:///home/acadgild/RITESH/Dataset_User_details.txt")
rdd: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[102] at textFile at <console>:27

scala> val userDF = rdd.map(x=>x.split(",")).map(array=>(array(0),array(1),array(2))).toDF("person_id","name","age")
userDF: org.apache.spark.sql.DataFrame = [person_id: string, name: string, age: string]
```

1) What is the distribution of the total number of air-travelers per year

Ans - holiday dataframe is grouped by year

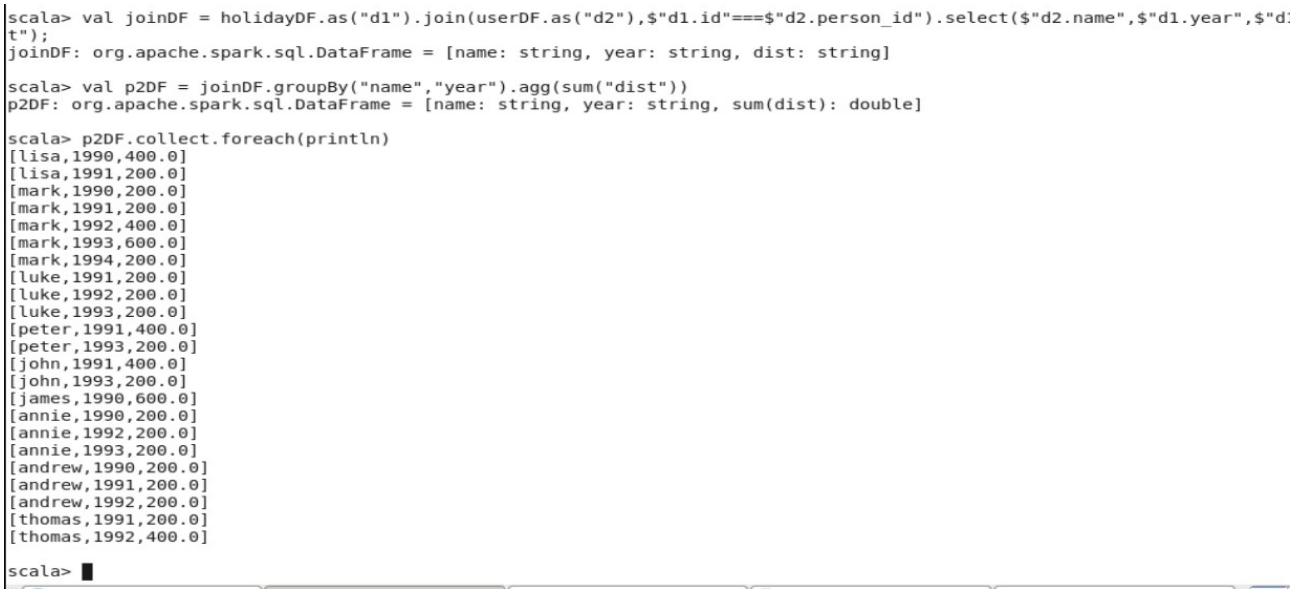
holidayDF.groupBy(“year”).count.show

```
scala> holidayDF.groupBy("year").count.show
+----+-----+
|year|count|
+----+-----+
|1990|    8|
|1991|    9|
|1992|    7|
|1993|    7|
|1994|    1|
+----+-----+
```

2) What is the total air distance covered by each user per year

Ans - Joined the user data and holiday data , then grouped by name and year to find the aggregate of the dist

val joinDF = holidayDF.as(“d1”).join(userDF.as(“d2”), $”s1.id”===$”d2.person_id”).select($”d2.name”,$”d1.year”, $”d1.dist”);
val p2DF = joinDF.groupBy(“name”,”year”).agg(sum(“dist”))



```
scala> val joinDF = holidayDF.as("d1").join(userDF.as("d2"),$"d1.id"===$"d2.person_id").select($"d2.name",$"d1.year",$"d1
t");
joinDF: org.apache.spark.sql.DataFrame = [name: string, year: string, dist: string]

scala> val p2DF = joinDF.groupBy("name","year").agg(sum("dist"))
p2DF: org.apache.spark.sql.DataFrame = [name: string, year: string, sum(dist): double]

scala> p2DF.collect.foreach(println)
[lisa,1990,400.0]
[lisa,1991,200.0]
[mark,1990,200.0]
[mark,1991,200.0]
[mark,1992,400.0]
[mark,1993,600.0]
[mark,1994,200.0]
[luke,1991,200.0]
[luke,1992,200.0]
[luke,1993,200.0]
[peter,1991,400.0]
[peter,1993,200.0]
[john,1991,400.0]
[john,1993,200.0]
[james,1990,600.0]
[annie,1990,200.0]
[annie,1992,200.0]
[annie,1993,200.0]
[andrew,1990,200.0]
[andrew,1991,200.0]
[andrew,1992,200.0]
[thomas,1991,200.0]
[thomas,1992,400.0]

scala>
```

3) Which user has travelled the largest distance till date

Ans - joinDFis grouped by name and its aggregate sum of the distance is ordered in descending and its top value is displayed.

val p3DF = joinDF.groupBy(“name”).agg(sum(“dist”)).orderBy($”sum(dist)”.desc).show(1)

```
scala> val p3DF = joinDF.groupBy("name").agg(sum("dist")).orderBy($"sum(dist)".desc).show(1)
+----+---------+
|name|sum(dist)|
+----+---------+
|mark|   1600.0|
+----+---------+
only showing top 1 row

p3DF: Unit = ()

scala>
```

4) What is the most preferred destination for all users.

Ans - Hoiday dataframe is grouped by destination and its frequency is calculated , then sorted in descending order and its top value is shown

val p4DF = holidayDF.groupBy(“dest”).count().orderBy($”count”.desc).show(1)

```
scala> val p4DF = holidayDF.groupBy("dest").count().orderBy($"count".desc).show(1)
+----+-----+
|dest|count|
+----+-----+
| IND|    9|
+----+-----+
only showing top 1 row

p4DF: Unit = ()

scala>
```