Bigdata Assignment 6.6

1) Which route is generating the most revenue per year
2) What is the total amount spent by every user on air-travel per year
3) Considering age groups of < 20 , 20-35, 35 > ,Which age group is travelling the most every year.

Solution -

- Loaded the datasets into dataframes using RDD.

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

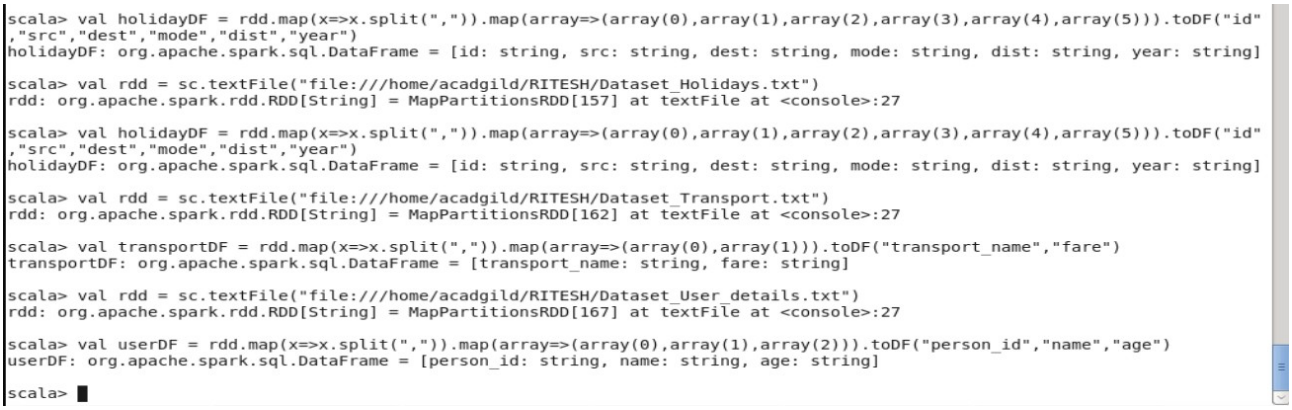
val holidayDF = rdd.map(x=>x.split(",")).map(array=>(array(0),array(1),array(2),array(3),array(4),array(5))).toDF("id","src","dest","mode","dist","year")

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")



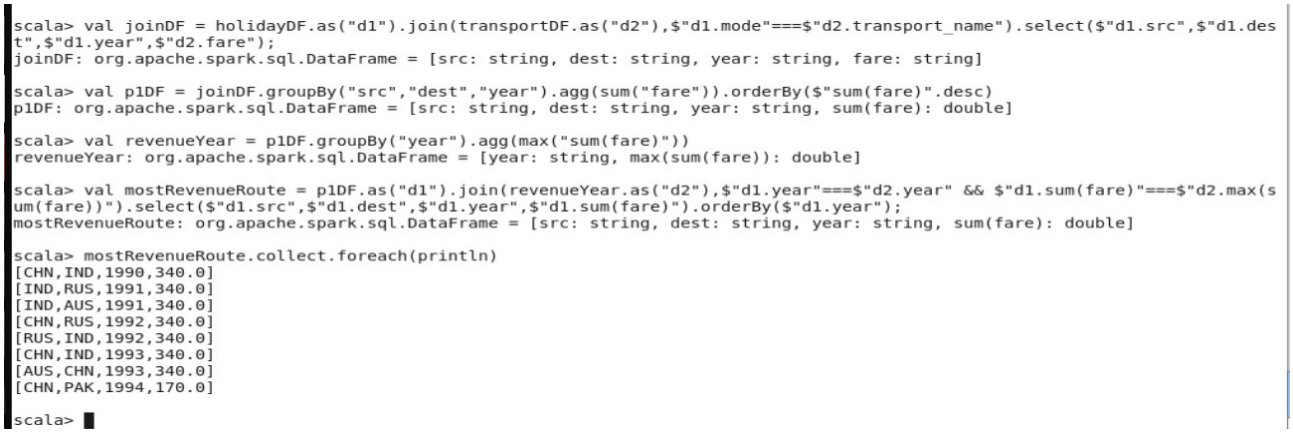
1.
//A joined data frame is made of holiday and transport dataframes on mode.
val joinDF = holidayDF.as(“d1”).join(transportDF.as(“d2”), $”d1.mode”===$”d2.transport_name”).select($”d1.src”,$”d1.dest”, $”d1.year”,$”d2.fare”);

//Dataframe is grouped by source , destination and year, corresponding fare is aggregated and it sorted in descending order.
val p1DF = joinDF.groupBy(“src”,”dest”,year”).agg(sum(“fare”)).orderBy($”sum (fare)”.desc)

//Grouped the dataframe by year and summation of its fare in a year is calculated
val revenueYear = p1DF.groupBy(“year”).agg(max(“sum(fare)”)

//Joined both the dataframes on year and sum of fares and columns like source,dest,year and revenue in year.
val mostRevenueRoute = p1DF.as(“d1”).join(revenueYear.as(“d2”). $”d1.year”===$”d2.year” && $”d1.sum(fare)===$”d2.max(sum(fare))”).select($”d1.src”,$”d1.dest”, $”d1.year”,$”d1.sum(fare)”).orderBy($”d1.year”);

//**Max** revenue collection in a year for a route.
mostRevenueRoute.collect.foreacch(println)



2.

//A joined data frame is made of holiday and user dataframes on mode.
val joinDF = holidayDF.as(“d1”).join(userDF.as(“d2”), $”d1.id”===$”d2.person_id”).select($”d1.name”,$”d1.year”, $”d1.mode”);

//A joined data frame is made of holiday and transport dataframes on mode.
val joinFare = holidayDF.as(“d1”).join(transportDF.as(“d2”), $”d1.mode”===$”d2.transport_name”).select($”d1.name”, $”d1.year”,$”d1.mode”,$”d2.fare”);

//Grouped the dataframe by name and year and its summation of fare in a year calculated.
val p2DF = joinDF.groupBy(“name”,year”).agg(sum(“fare”))

//Total amount spent by every user on air-travel per year is shown
p2DF.collect.foreach(println)

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

scala> val joinFare = joinDF.as("d1").join(transportDF.as("d2"),$"d1.mode"===$"d2.transport_name").select($"d1.name",$"d1.yea
r",$"d1.mode",$"d2.fare");
joinFare: org.apache.spark.sql.DataFrame = [name: string, year: string, mode: string, fare: string]

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

scala> p2DF.collect.foreach(println)
[lisa,1990,340.0]
[lisa,1991,170.0]
[mark,1990,170.0]
[mark,1991,170.0]
[mark,1992,340.0]
[mark,1993,510.0]
[mark,1994,170.0]
[luke,1991,170.0]
[luke,1992,170.0]
[luke,1993,170.0]
[peter,1991,340.0]
[peter,1993,170.0]
[john,1991,340.0]
[john,1993,170.0]
[james,1990,510.0]
[annie,1990,170.0]
[annie,1992,170.0]
[annie,1993,170.0]
[andrew,1990,170.0]
[andrew,1991,170.0]
[andrew,1992,170.0]
[thomas,1991,170.0]
[thomas,1992,340.0]

scala>
```

3.

//udf function is created for assigning age grp given on the condition.

val ageGrp = udf((age: String) => { if(age.toInt < 20) { "<20"; } else { if(age.toInt > 35) { ">35"; } else { "20-35"; }}});

//New column is created depending on the value of age.

val userDFGrp = userDF.withColumn(“AgeGrp”,”ageGrp($”age”))

//Grouped the dataframe by Agegrp and year and its count.

val grpJoin = joinDF.groupBy(“AgeGrp”,year”).count

//Grouped the dataframe by year and its aggregate of maximum of count is calculated.

val yearMax = grpJoin.groupBy(“year”).agg(max(“count”))

//A joined data frame is made of grpJoin and yearMAX dataframes on year and count

val maxGrp = grpJoin.as(“d1”).join(yearMAX.as(“d2”), $”d1.year”===$”d2.year”&&$”d1.count”===”$”d2.max(count)”).select($”d1.AgeGrp”,$”d1.year”,$”d2.max(count));

//The result is shown

maxGrp.collect.foreach(println)

```
scala> val ageGrp = udf((age:String)=> { if(age.toInt< 20) { "<20" ;} else { if(age.toInt> 35){ ">35" ;} else { "20-35";}}})
ageGrp: org.apache.spark.sql.UserDefinedFunction = UserDefinedFunction(<function1>,StringType,List(StringType))

scala> val userDFGrp = userDF.withColumn("AgeGrp",ageGrp($"age"))
userDFGrp: org.apache.spark.sql.DataFrame = [person_id: string, name: string, age: string, AgeGrp: string]

scala> val grpJoin = joinDF.groupBy("AgeGrp","year").count
grpJoin: org.apache.spark.sql.DataFrame = [AgeGrp: string, year: string, count: bigint]

scala> val yearMax = grpJoin.groupBy("year").agg(max("count"))
yearMax: org.apache.spark.sql.DataFrame = [year: string, max(count): bigint]

scala> val maxGrp = grpJoin.as("d1").join(yearMax.as("d2"),$"d1.year"===$"d2.year" && $"d1.count"===$"d2.max(count)").select(
$"d1.AgeGrp",$"d1.year",$"d2.max(count)");
maxGrp: org.apache.spark.sql.DataFrame = [AgeGrp: string, year: string, max(count): bigint]

scala> maxGrp.collect.foreach(println)
[<20,1993,5]
[20-35,1994,1]
[20-35,1990,5]
[20-35,1991,4]
[>35,1992,4]

scala>
```