Session 18 : RDD'S CONTD. & INTRODUCTION TO DATAFRAMES

Assignment 2

**Problem Statement**

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.

Use the dataset given below:

<https://drive.google.com/drive/folders/0B_P3pWagdIrrVThBaUdVSUtzbms>

**Solution:-**

Starting Spark shell

Command - spark-shell



Loading data to dataframe:-

**Scala>** val holidaysRDD = sc.textFile("S18\_Dataset\_Holidays.txt");

**Scala>** val holidaysDF= holidaysRDD.map(lines=>lines.split(",")).map(arrays => (arrays(0),arrays(1),arrays(2),arrays(3),arrays(4),arrays(5))).toDF("Person\_ID","Source","Destination","Mode","Distance","Year");

**Scala>** val transportRDD = sc.textFile("S18\_Dataset\_Transport.txt");

**Scala>** val transportDF= transportRDD.map(lines=>lines.split(",")).map(arrays=>(arrays(0),arrays(1))).toDF("Transport\_Name","Fare");

**Scala>** val userRDD = sc.textFile("S18\_Dataset\_User\_details.txt");

**Scala>** val userDF= userRDD.map(lines=>lines.split(",")).map(arrays=>(arrays(0),arrays(1),arrays(2))).toDF("Person\_ID","Name","Age");



All the datasets have been loaded in to temporary tables.

These dataframes would be used to find solution to problem statements

Listing out dataframes and column names:-

holidaysDF -> "Person\_ID","Source","Destination","Mode","Distance","Year"

transportDF -> "Transport\_Name","Fare"

user DF -> "Person\_ID","Name","Age"

**Starting with problem statements now:-**

1. **Which route is generating the most revenue per year**

**scala>** val joinDF1 = holidaysDF.as("d1").join(transportDF.as("d2"), $"d1.Mode" === $"d2.Transport\_Name").select($"d1.Source", $"d1.Destination", $"d1.Year",$"d2.Fare");

**scala>** val problem1DF = joinDF1.groupBy("Source","Destination","Year").agg(sum("Fare")).orderBy($"sum(Fare)".desc);

problem1DF

**scala>** val yearMax = problem1DF.groupBy("Year").agg(max("sum(Fare)"));

yearMax: org.apache.spark.sql.DataFrame

**scala>** val mostRevenueRoute = problem1DF.as("d1").join(yearMax.as("d2"), $"d1.Year" === $"d2.Year" && $"d1.sum(Fare)" === $"d2.max(sum(Fare))").select($"d1.Source", $"d1.Destination", $"d1.Year",$"d1.sum(Fare)").orderBy($"d1.Year");





***Above is the list of most profitable routes per year.***

1. **What is the total amount spent by every user on air-travel per year**

**scala>** val joinDF = holidaysDF.as("d1").join(userDF.as("d2"), $"d1.Person\_ID" === $"d2.Person\_ID").select($"d2.Name", $"d1.Year", $"d1.Mode");

**scala>** val joinWithPrice = joinDF.as("d1").join(transportDF.as("d2"), $"d1.Mode" === $"d2.Transport\_Name").select($"d1.Name", $"d1.Year", $"d1.Mode",$"d2.Fare");

**scala>** val problem2DF= joinWithPrice.groupBy("Name","Year").agg(sum("Fare"));





***Above is the list of total amount spent by every user on air-travel per year***

1. **Considering age groups of < 20 , 20-35, 35 > ,Which age group is travelling the most every year**

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

**scala>** val userDFGrp = userDF.withColumn("AgeGrp", ageGrp($"Age"))

**scala>** val joinDF3 = holidaysDF.as("d1").join(userDFGrp.as("d2"), $"d1.Person\_ID" === $"d2.Person\_ID").select($"d2.AgeGrp", $"d1.Year", $"d1.Mode");

**scala>** val groupedJoin = joinDF3.groupBy("AgeGrp","Year").count();

**scala>** val yearMax = groupedJoin.groupBy("Year").agg(max("count"));



**scala>** val maxAgeGrp = groupedJoin.as("d1").join(yearMax.as("d2"), $"d1.Year" === $"d2.Year" && $"d1.count" === $"d2.max(count)").select($"d1.AgeGrp", $"d1.Year", $"d2.max(count)");

**scala>** maxAgeGrp.collect.foreach(println);



**In 1990 age group 20-35 travelled most with 5 trips**

**In 1991 age group 20-35 travelled most with 4 trips**

**In 1992 age group >35 travelled most with 4 trips**

**In 1993 age group <20 travelled most with 5 trips**

**In 1994 age group 20-35 travelled most with 1 trip**