SPARK ASSIGNMENT 18.2

==========

S18_Dataset_User_Details.txt -> userid,name,age

S18_Dataset_Transport.txt -> holiday_mode, cost_per_unit

S18_Dataset_Holidays.txt -> userid,source, destination , travel_mode, distance , year_of_travel

Problem Statement

18.2.1) Which route is generating the most revenue per year

18.2.2) What is the total amount spent by every user on air-holiday per year

18.2.3) Considering age groups of < 20 , 20-35, 35 > ,Which age group is holidaying the most every year.

Input Commands:

//putting the input files to HDFS

[acadgild@localhost]$cd Downloads

[acadgild@localhost Downloads]$ hadoop fs -put S18_Dataset_User_details.txt /user/acadgild/spark/

[acadgild@localhost Downloads]$ hadoop fs -put S18_Dataset_Transport.txt /user/acadgild/spark/

[acadgild@localhost Downloads]$ hadoop fs -put S18_Dataset_Holidays.txt /user/acadgild/spark/

//Initiating the spark-shell to run spark

[acadgild@localhost Downloads]$ spark-shell

```
//importing the spark packages

import org.apache.spark.sql.Row;

import
org.apache.spark.sql.types.{StructType,StructField,StringType,NumericTyp
e,IntegerType};

import org.apache.spark.sql._

import sqlContext.implicits._

//Loading the input files to respective RDDs

val userdetailsRDD =
sc.textFile("hdfs://localhost:9000//user/acadgild/spark/S18_Dataset_User_d
etails.txt")

val transportRDD =
sc.textFile("hdfs://localhost:9000//user/acadgild/spark/S18_Dataset_Transp
ort.txt")

val holidaysRDD =
sc.textFile("hdfs://localhost:9000//user/acadgild/spark/S18_Dataset_Holiday
s.txt")

//Defining the schemas for each of the 3 files mentioned above respectively

val schemaStringu = "userid:integer,name:string,age:integer"

val schemaStringt = "travel_mode:string,cost_per_unit:integer"

val schemaStringh =
"userid:integer,source:string,destination:string,travel_mode:string,distanc
e:integer,year_of_travel:integer"

//Defining the Structtype and StructField for each DB2 Schema

val schemau = StructType(schemaStringu.split(",").map(fieldInfo =>
StructField(fieldInfo.split(":")(0),

 if (fieldInfo.split(":")(1).equals("integer")) IntegerType else
StringType,true)))
```

```
val schemat = StructType(schemaStringt.split(",").map(fieldInfo =>
StructField(fieldInfo.split(":")(0), if (fieldInfo.split(":")(1).equals("string"))
StringType else IntegerType, true)))

val schemah = StructType(schemaStringh.split(",").map(fieldInfo =>
StructField(fieldInfo.split(":")(0), if (fieldInfo.split(":")(1).equals("string"))
StringType else IntegerType, true)))

//Mapping the data present in TEXT files at HDFS

val RDDu = userdetailsRDD.map(_.split(",")).map(r => Row(r(0).toInt, r(1),
r(2).toInt ))

val RDDt = transportRDD.map(_.split(",")).map(r => Row(r(0), r(1).toInt ))

val RDDh = holidaysRDD.map(_.split(",")).map(r => Row(r(0).toInt, r(1) ,
r(2), r(3) , r(4).toInt, r(5).toInt ))

//Defining the SQLCONTEXT object with the help of Spark Context object

val sqlContext = new org.apache.spark.sql.SQLContext(sc);

//Creating the Dataframe with the help of schema and data in text files

val uDF =sqlContext.createDataFrame(RDDu, schemau)

val tDF =sqlContext.createDataFrame(RDDt, schemat)

val hDF =sqlContext.createDataFrame(RDDh, schemah)

//Defining the temporary tables  with the newly created dataframes

uDF.registerTempTable("userdetails")

tDF.registerTempTable("transport")

hDF.registerTempTable("holiday")

============================================================

18.2.1) Which route is generating the most revenue per year

val resdf = sqlContext.sql("SELECT  y.source , y.destination
,y.year_of_travel FROM transport x, holiday y where x.travel_mode =
y.travel_mode GROUP BY y.source , y.destination ,y.year_of_travel ORDER
BY SUM(x.cost_per_unit) DESC").take(6)

resdf.foreach(println)
```

Output:

```
scala> val resdf = sqlContext.sql("SELECT  y.source , y.destination ,y.year_of_travel FROM transport
 x, holiday y where x.travel_mode = y.travel_mode GROUP BY y.source , y.destination ,y.year_of_trave
l ORDER BY SUM(x.cost_per_unit) DESC").take(6)
resdf: Array[org.apache.spark.sql.Row] = Array([CHN,RUS,1992], [IND,RUS,1991], [AUS,CHN,1993], [IND,
AUS,1991], [RUS,IND,1992], [CHN,IND,1990])

scala> resdf.foreach(println)
[CHN,RUS,1992]
[IND,RUS,1991]
[AUS,CHN,1993]
[IND,AUS,1991]
[RUS,IND,1992]
[CHN,IND,1990]

scala>
```

18.2.2) What is the total amount spent by every user on air-holiday per year

```
sqlContext.sql("SELECT z.userid,z.name,y.year_of_travel ,
SUM(x.cost_per_unit) as total_amount FROM transport x, holiday y
,userdetails z where x.travel_mode = y.travel_mode and x.travel_mode =
'airplane' and y.userid = z.userid group by z.userid,z.name,y.year_of_travel
ORDER by z.userid,z.name,y.year_of_travel").show()
```

Output:

```
scala> sqlContext.sql("SELECT z.userid,z.name,y.year_of_travel , SUM(x.cost_per_unit) as total_amoun
t FROM transport x, holiday y ,userdetails z where x.travel_mode = y.travel_mode and x.travel_mode =
 'airplane' and y.userid = z.userid group by z.userid,z.name,y.year_of_travel ORDER by z.userid,z.na
me,y.year_of_travel").show()
+------+------+--------------+------------+
|userid|  name|year_of_travel|total_amount|
+------+------+--------------+------------+
|     1|  mark|          1990|         170|
|     1|  mark|          1993|         510|
|     2|  john|          1991|         340|
|     2|  john|          1993|         170|
|     3|  luke|          1991|         170|
|     3|  luke|          1992|         170|
|     3|  luke|          1993|         170|
|     4|  lisa|          1990|         340|
|     4|  lisa|          1991|         170|
|     5|  mark|          1991|         170|
|     5|  mark|          1992|         340|
|     5|  mark|          1994|         170|
|     6| peter|          1991|         340|
|     6| peter|          1993|         170|
|     7| james|          1990|         510|
|     8|andrew|          1990|         170|
|     8|andrew|          1991|         170|
|     8|andrew|          1992|         170|
|     9|thomas|          1991|         170|
|     9|thomas|          1992|         340|
+------+------+--------------+------------+
only showing top 20 rows
```

18.2.3) Considering age groups of < 20 , 20-35, 35 > ,Which age group is holidaying the most every year.

val resdf = sqlContext.sql("select age from userdetails z, holiday y where z.userid = y.userid group by age order by count(age) desc").take(2)

resdf.foreach(println)

```
scala> val resdf = sqlContext.sql("select age from userdetails z, holiday y where z.userid = y.useri
d group by age order by count(age) desc").take(2)
resdf: Array[org.apache.spark.sql.Row] = Array([15], [25])

scala> resdf.foreach(println)
[15]
[25]
```

==========