SPARK ASSIGNMENT 18.3

==========

S18_Dataset_User_Details.txt -> userid,name,age

S18_Dataset_Transport.txt -> travel_mode, cost_per_unit

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

Problem Statement

18.3.1) Considering age groups of < 20 , 20-35, 35 > ,Which age group spends the most amount of money travelling.

18.3.2) What is the amount spent by each age-group, every year in travelling?

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")
```

---

1) Considering age groups of < 20 , 20-35, 35 > ,Which age group spends the most amount of money travelling.

```
val resdf = sqlContext.sql("select z.age,SUM(x.cost_per_unit) as
total_spending from userdetails z, holiday y ,transport x where z.userid =
```

y.userid and x.travel_mode = y.travel_mode group by z.age order by total_spending DESC").take(2)

resdf.show()

Output:

```
scala> val resdf = sqlContext.sql("select z.age,SUM(x.cost_per_unit) as total_spending from userdeta
ils z, holiday y ,transport x where z.userid = y.userid and x.travel_mode = y.travel_mode group by z
.age order by total_spending DESC").take(2)
resdf: Array[org.apache.spark.sql.Row] = Array([15,680], [25,680])

scala> resdf.foreach(println)
[15,680]
[25,680]
```

-

---

2) What is the amount spent by each age-group, every year in travelling?

---

val res1df = sqlContext.sql("select y.year_of_travel , SUM(x.cost_per_unit) as total_spending_of_age_below_20 from userdetails z, holiday y ,transport x where z.userid = y.userid and x.travel_mode = y.travel_mode and z.age < 20 group by y.year_of_travel order by y.year_of_travel DESC")

val res2df = sqlContext.sql("select y.year_of_travel , SUM(x.cost_per_unit) as total_spending_of_age_bracket_from_20_to_35 from userdetails z, holiday y ,transport x where z.userid = y.userid and x.travel_mode = y.travel_mode and z.age >= 20 and z.age<=35 group by y.year_of_travel order by y.year_of_travel DESC")

val res3df = sqlContext.sql("select y.year_of_travel , SUM(x.cost_per_unit) as total_spending_of_age_above_35 from userdetails z, holiday y ,transport x where z.userid = y.userid and x.travel_mode = y.travel_mode and z.age > 35 group by y.year_of_travel order by y.year_of_travel DESC")

res1df.show()

res2df.show()

res3df.show()

Output:

```
scala> val res1df = sqlContext.sql("select y.year_of_travel , SUM(x.cost_per_unit) as total_spending
_of_age_below_20 from userdetails z, holiday y ,transport x where z.userid = y.userid and x.travel_m
ode = y.travel_mode and z.age < 20 group by y.year_of_travel order by y.year_of_travel DESC")
res1df: org.apache.spark.sql.DataFrame = [year_of_travel: int, total_spending_of_age_below_20: bigin
t]

scala> val res2df = sqlContext.sql("select y.year_of_travel , SUM(x.cost_per_unit) as total_spending
_of_age_bracket_from_20_to_35 from userdetails z, holiday y ,transport x where z.userid = y.userid a
nd x.travel_mode = y.travel_mode and z.age >= 20 and z.age<=35 group by y.year_of_travel order by y.
year_of_travel DESC")
res2df: org.apache.spark.sql.DataFrame = [year_of_travel: int, total_spending_of_age_bracket_from_20
_to_35: bigint]

scala> val res3df = sqlContext.sql("select y.year_of_travel , SUM(x.cost_per_unit) as total_spending
_of_age_above_35 from userdetails z, holiday y ,transport x where z.userid = y.userid and x.travel_m
ode = y.travel_mode and z.age > 35 group by y.year_of_travel order by y.year_of_travel DESC")
res3df: org.apache.spark.sql.DataFrame = [year_of_travel: int, total_spending_of_age_above_35: bigin
t]


scala> res1df.show()
+--------------+-----------------------------+
|year_of_travel|total_spending_of_age_below_20|
+--------------+-----------------------------+
|          1993|                          850|
|          1992|                          170|
|          1991|                          510|
|          1990|                          170|
+--------------+-----------------------------+


scala> res2df.show()
+--------------+---------------------------------------------+
|year_of_travel|total_spending_of_age_bracket_from_20_to_35|
+--------------+---------------------------------------------+
|          1994|                                          170|
|          1993|                                          170|
|          1992|                                          340|
|          1991|                                          680|
|          1990|                                          850|
+--------------+---------------------------------------------+


scala> res3df.show()
+--------------+-----------------------------+
|year_of_travel|total_spending_of_age_above_35|
+--------------+-----------------------------+
|          1993|                          170|
|          1992|                          680|
|          1991|                          340|
|          1990|                          340|
+--------------+-----------------------------+
```