

## Assignment 18.1:

### Problem Statement:

#### Initial Steps:

#### Step1: Create a temporary table User

```
import org.apache.spark.sql.types.{StructType, StringType, IntegerType, StructField}

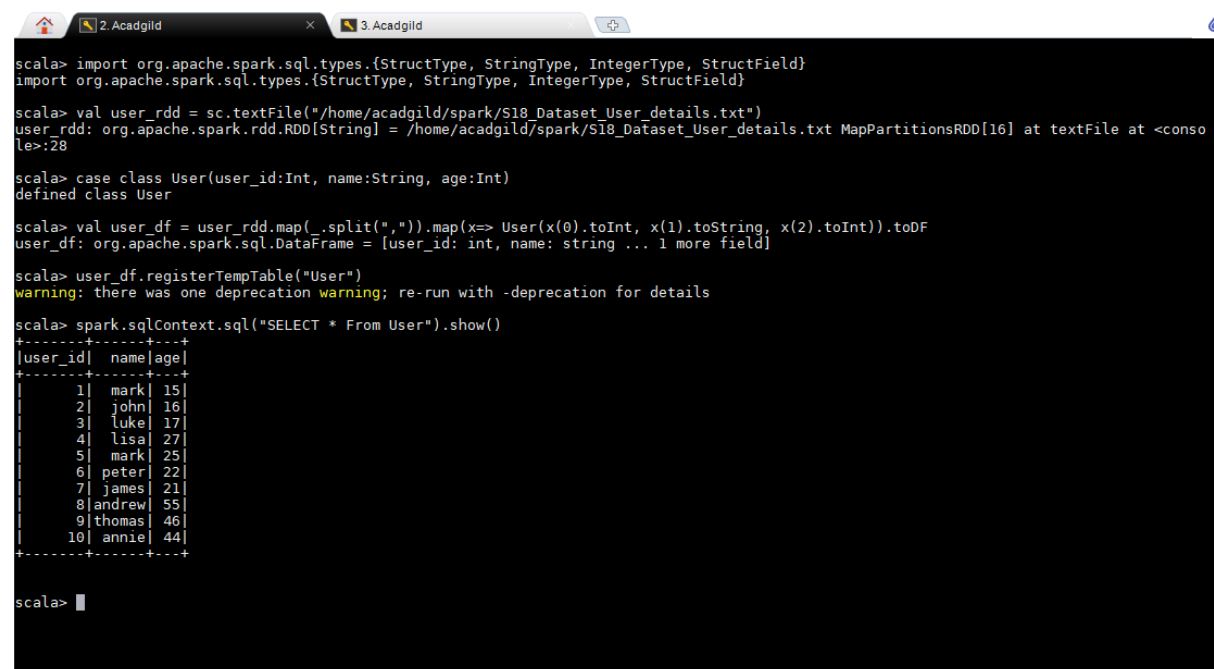
val user_rdd = sc.textFile("/home/acadgild/assignment_18.1/S18_Dataset_User_details.txt")

case class User(user_id:Int, name:String, age:Int)

val user_df = user_rdd.map(_.split(",")).map(x=> User(x(0).toInt, x(1).toString, x(2).toInt)).toDF

user_df.registerTempTable("User")

spark.sqlContext.sql("SELECT * From User").show()
```



```
scala> import org.apache.spark.sql.types.{StructType, StringType, IntegerType, StructField}
import org.apache.spark.sql.types.{StructType, StringType, IntegerType, StructField}

scala> val user_rdd = sc.textFile("/home/acadgild/spark/S18_Dataset_User_details.txt")
user_rdd: org.apache.spark.rdd.RDD[String] = /home/acadgild/spark/S18_Dataset_User_details.txt MapPartitionsRDD[16] at textFile at <console>:28

scala> case class User(user_id:Int, name:String, age:Int)
defined class User

scala> val user_df = user_rdd.map(_.split(",")).map(x=> User(x(0).toInt, x(1).toString, x(2).toInt)).toDF
user_df: org.apache.spark.sql.DataFrame = [user_id: int, name: string ... 1 more field]

scala> user_df.registerTempTable("User")
warning: there was one deprecation warning; re-run with -deprecation for details

scala> spark.sqlContext.sql("SELECT * From User").show()
+-----+-----+
|user_id| name|age|
+-----+-----+
|      1| mark| 15|
|      2| john| 16|
|      3| luke| 17|
|      4| lisa| 27|
|      5| mark| 25|
|      6| peter| 22|
|      7| james| 21|
|      8| andrew| 55|
|      9| thomas| 46|
|     10| annie| 44|
+-----+-----+

scala> █
```

#### Step2: Create a temporary table Travel

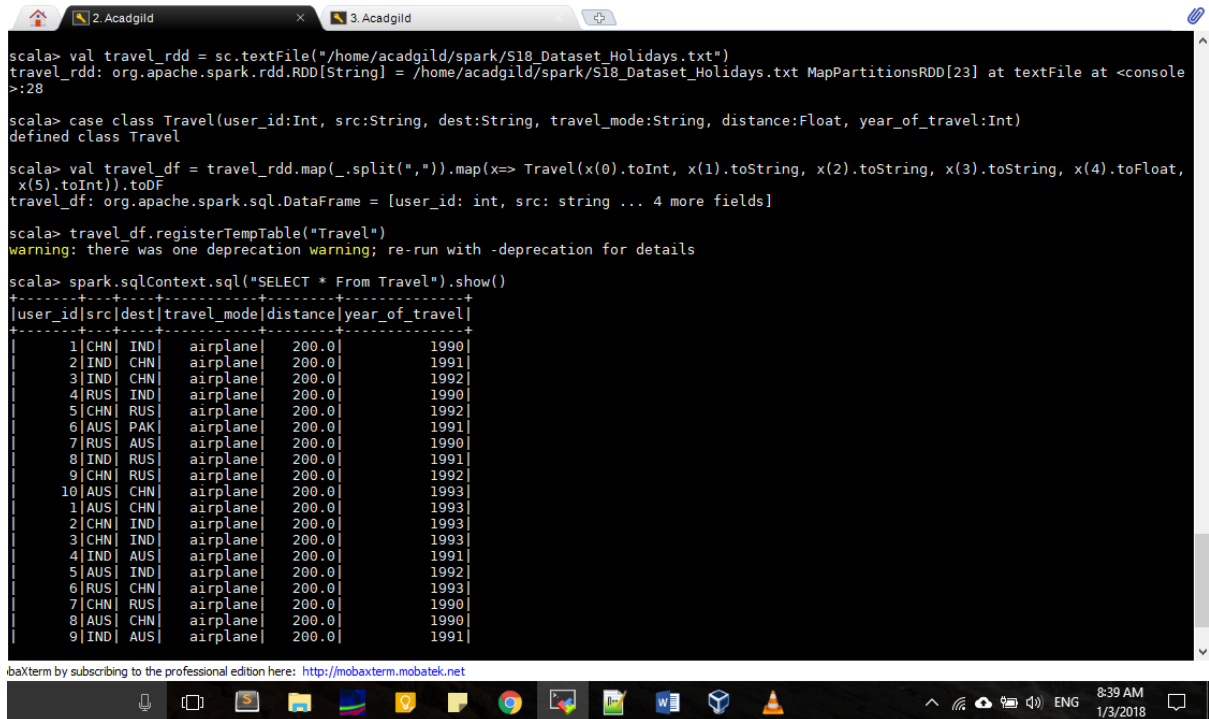
```
val travel_rdd = sc.textFile("/home/acadgild/spark/S18_Dataset_Holidays.txt")

case class Travel(user_id:Int, src:String, dest:String, travel_mode:String, distance:Float,
year_of_travel:Int)

val travel_df = travel_rdd.map(_.split(",")).map(x=> Travel(x(0).toInt, x(1).toString, x(2).toString,
x(3).toString, x(4).toFloat, x(5).toInt)).toDF

travel_df.registerTempTable("Travel")

spark.sqlContext.sql("SELECT * From Travel").show()
```



```
scala> val travel_rdd = sc.textFile("/home/acadgild/spark/S18_Dataset_Holidays.txt")
travel_rdd: org.apache.spark.rdd.RDD[String] = /home/acadgild/spark/S18_Dataset_Holidays.txt MapPartitionsRDD[23] at textFile at <console>:28

scala> case class Travel(user_id:Int, src:String, dest:String, travel_mode:String, distance:Float, year_of_travel:Int)
defined class Travel

scala> val travel_df = travel_rdd.map(_._split(",")).map(x=> Travel(x(0).toInt, x(1).toString, x(2).toString, x(3).toString, x(4).toFloat, x(5).toInt)).toDF
travel_df: org.apache.spark.sql.DataFrame = [user_id: int, src: string ... 4 more fields]

scala> travel_df.registerTempTable("Travel")
warning: there was one deprecation warning; re-run with -deprecation for details

scala> spark.sqlContext.sql("SELECT * From Travel").show()
+-----+-----+-----+-----+-----+-----+
|user_id|src|dest|travel_mode|distance|year_of_travel|
+-----+-----+-----+-----+-----+
|1|CHN|IND|airplane|200.0|1990|
|2|IND|CHN|airplane|200.0|1991|
|3|IND|CHN|airplane|200.0|1992|
|4|RUS|IND|airplane|200.0|1990|
|5|CHN|RUS|airplane|200.0|1992|
|6|AUS|PAK|airplane|200.0|1991|
|7|RUS|AUS|airplane|200.0|1990|
|8|IND|RUS|airplane|200.0|1991|
|9|CHN|RUS|airplane|200.0|1992|
|10|AUS|CHN|airplane|200.0|1993|
|1|AUS|CHN|airplane|200.0|1993|
|2|CHN|IND|airplane|200.0|1993|
|3|CHN|IND|airplane|200.0|1993|
|4|IND|AUS|airplane|200.0|1991|
|5|AUS|IND|airplane|200.0|1992|
|6|RUS|CHN|airplane|200.0|1993|
|7|CHN|RUS|airplane|200.0|1990|
|8|AUS|CHN|airplane|200.0|1990|
|9|IND|AUS|airplane|200.0|1991|
```

### Step3: Create temporary table Transport

```
val transport_rdd = sc.textFile("/home/acadgild/spark/S18_Dataset_Transport.txt")
```

```
case class Transport(travel_mode:String, cost_per_unit:Float)
```

```
val transport_df = transport_rdd.map(_._split(",")).map(x=> Transport(x(0).toString, x(1).toFloat)).toDF
```

```
transport_df.registerTempTable("Transport")
```

```
spark.sqlContext.sql("SELECT * From Transport").show()
```

```
scala> val transport_rdd = sc.textFile("/home/acadgild/spark/S18_Dataset_Transport.txt")
transport_rdd: org.apache.spark.rdd.RDD[String] = /home/acadgild/spark/S18_Dataset_Transport.txt MapPartitionsRDD[30] at textFile at <console>:28

scala> case class Transport(travel_mode:String, cost_per_unit:Float)
defined class Transport

scala> val transport_df = transport_rdd.map(_split(",")).map(x=> Transport(x(0).toString, x(1).toFloat)).toDF
transport_df: org.apache.spark.sql.DataFrame = [travel_mode: string, cost_per_unit: float]

scala> transport_df.registerTempTable("Transport")
warning: there was one deprecation warning; re-run with -deprecation for details

scala> spark.sqlContext.sql("SELECT * From Transport").show()
+-----+-----+
|travel_mode|cost_per_unit|
+-----+-----+
|airplane|170.0|
|car|140.0|
|train|120.0|
|ship|200.0|
+-----+-----+

scala> 
```

baXterm by subscribing to the professional edition here: <http://mobaxterm.mobatek.net>

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

```
spark.sqlContext.sql("SELECT year_of_travel, COUNT(*) as distribution FROM Travel WHERE travel_mode='airplane' GROUP BY year_of_travel").show()
```

```
scala> spark.sqlContext.sql("SELECT year_of_travel, COUNT(*) as distribution FROM Travel WHERE travel_mode='airplane' GROUP BY year_of_travel").show()
+-----+-----+
|year_of_travel|distribution|
+-----+-----+
|1990|8|
|1994|1|
|1991|9|
|1992|7|
|1993|7|
+-----+-----+

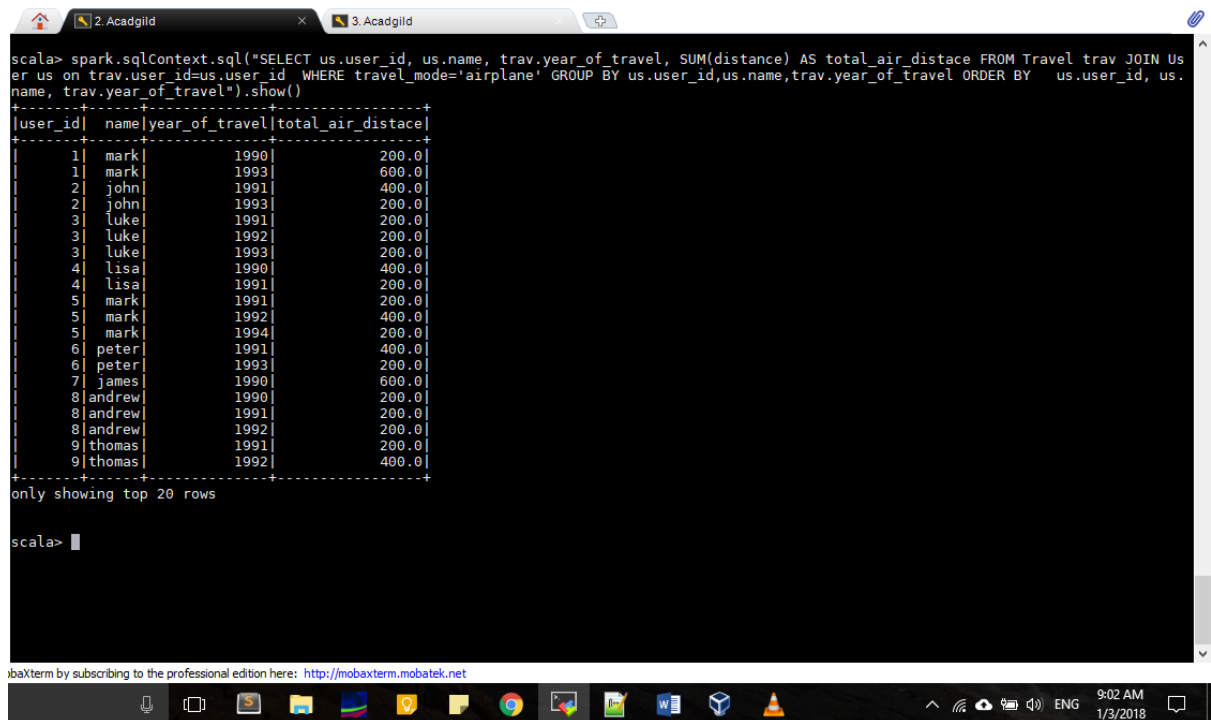
scala>
scala> 
```

baXterm by subscribing to the professional edition here: <http://mobaxterm.mobatek.net>

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

```
spark.sqlContext.sql("SELECT us.user_id, us.name, trav.year_of_travel, SUM(distance) AS total_air_distace FROM Travel trav JOIN User us on trav.user_id=us.user_id WHERE
```

travel\_mode='airplane' GROUP BY us.user\_id,us.name,trav.year\_of\_travel ORDER BY us.user\_id, us.name, trav.year\_of\_travel").show()

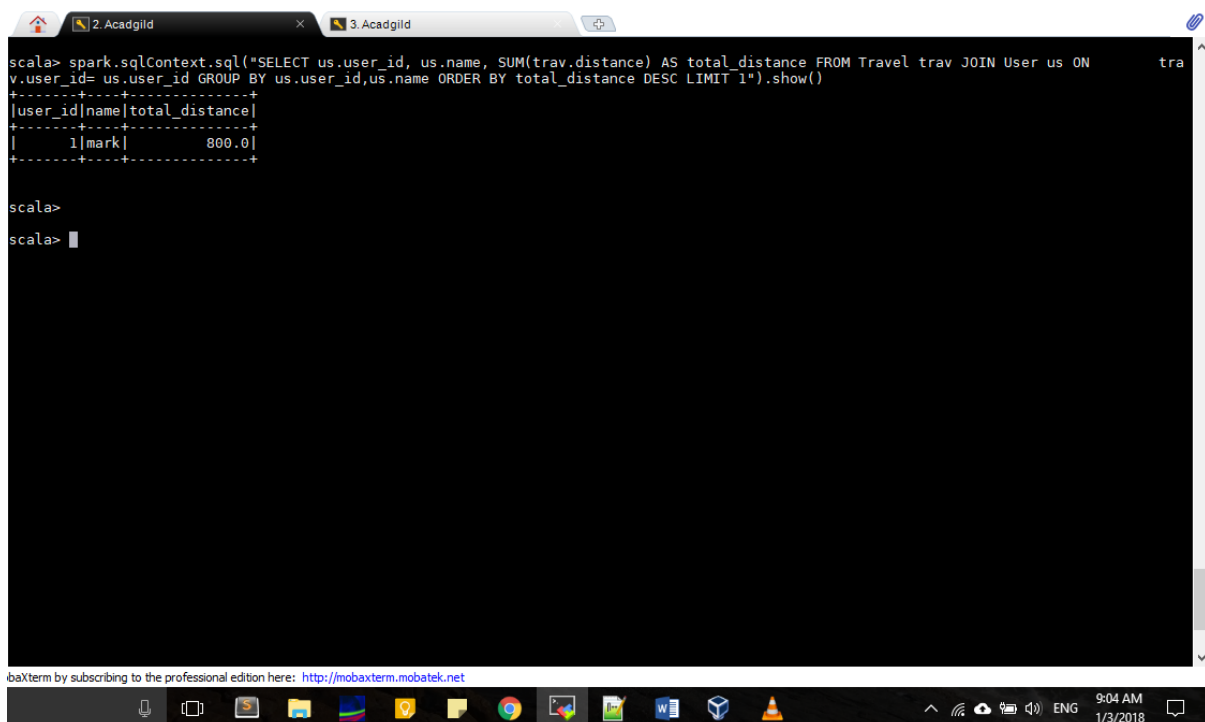


```
scala> spark.sqlContext.sql("SELECT us.user_id, us.name, trav.year_of_travel, SUM(distance) AS total_air_distace FROM Travel trav JOIN User us ON trav.user_id=us.user_id WHERE travel_mode='airplane' GROUP BY us.user_id,us.name,trav.year_of_travel ORDER BY us.user_id, us.name, trav.year_of_travel").show()
+-----+-----+-----+-----+
|user_id|name|year_of_travel|total_air_distace|
+-----+-----+-----+-----+
|1|mark|1990|200.0|
|1|mark|1993|600.0|
|2|john|1991|400.0|
|2|john|1993|200.0|
|3|luke|1991|200.0|
|3|luke|1992|200.0|
|3|luke|1993|200.0|
|4|lisa|1990|400.0|
|4|lisa|1991|200.0|
|5|mark|1991|200.0|
|5|mark|1992|400.0|
|5|mark|1994|200.0|
|6|peter|1991|400.0|
|6|peter|1993|200.0|
|7|james|1990|600.0|
|8|andrew|1990|200.0|
|8|andrew|1991|200.0|
|8|andrew|1992|200.0|
|9|thomas|1991|200.0|
|9|thomas|1992|400.0|
+-----+-----+-----+-----+
only showing top 20 rows

scala>
```

### 3) Which user has travelled the largest distance till date

spark.sqlContext.sql("SELECT us.user\_id, us.name, SUM(trav.distance) AS total\_distance FROM Travel trav JOIN User us ON trav.user\_id= us.user\_id GROUP BY us.user\_id,us.name ORDER BY total\_distance DESC LIMIT 1").show()

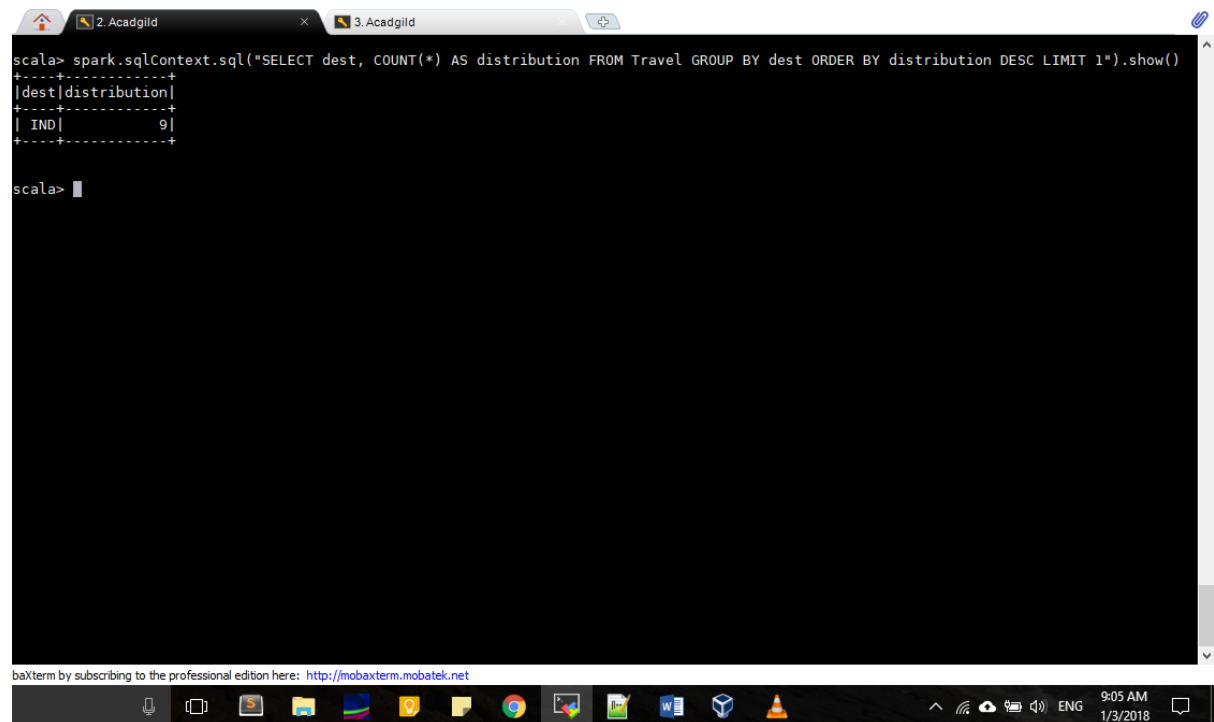


```
scala> spark.sqlContext.sql("SELECT us.user_id, us.name, SUM(trav.distance) AS total_distance FROM Travel trav JOIN User us ON trav.user_id= us.user_id GROUP BY us.user_id,us.name ORDER BY total_distance DESC LIMIT 1").show()
+-----+-----+-----+
|user_id|name|total_distance|
+-----+-----+-----+
|1|mark|800.0|
+-----+-----+-----+

scala>
scala>
```

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

```
spark.sqlContext.sql("SELECT dest, COUNT(*) AS distribution FROM Travel GROUP BY dest ORDER BY distribution DESC LIMIT 1").show()
```



The screenshot shows a terminal window with a dark background. At the top, there are two tabs labeled "2. Acadgild" and "3. Acadgild". The terminal displays the following text:

```
scala> spark.sqlContext.sql("SELECT dest, COUNT(*) AS distribution FROM Travel GROUP BY dest ORDER BY distribution DESC LIMIT 1").show()
+-----+
|dest|distribution|
+-----+
| IND|          9|
+-----+
```

Below the output, the prompt "scala>" is visible with a cursor. At the bottom of the terminal window, there is a small text link: "baXterm by subscribing to the professional edition here: <http://mobaxterm.mobatek.net>".

The Windows taskbar is visible at the bottom of the screen, showing various application icons and the system clock indicating 9:05 AM on 1/3/2018.