

Task 1: (Process Student Dataset)

Problem Statement 1:

1. Read the text file, and create a tupled rdd.

Steps: Read the dataset, get header, Filter the records which is not header

```
val student_rdd = sc.readFile("/home/acadgild/assignment/student_dataset")
```

```
val header = student_rdd.first()
```

```
val student_records = student_rdd.map(records => records != header)
```

2. Find the count of total number of rows present.

- Get the count

```
println(student_rdd.count())
```

3. What is the distinct number of subjects present in the entire school

Steps:

- Create schema for Student
- Register Temporary Table Student
- Use SQL query to get list of distinct subject

```
case class Student(name: String, subject: String, grade: String, marks: Int, age: Int)
```

```
student_records.registerTempTable("Student")
```

```
spark.sqlContext.sql("SELECT COUNT(*) FROM (SELECT DISTINCT subject FROM Student)").show()
```

4. What is the count of the number of students in the school, whose name is Mathew and marks is 55

- Create SQL with query criteria of name is Mathew and marks is 55 and get the count

```
spark.sqlContext.sql("SELECT COUNT(*) FROM (SELECT name FROM Student WHERE name='Mathew' and marks=55)").show()
```

```
acadgild@localhost:~/Desktop
File Edit View Search Terminal Tabs Help
acadgild@localhost:~/Desktop acadgild@localhost:~/assignment
scala> import org.apache.spark.sql.types.{StringType,IntegerType}
import org.apache.spark.sql.types.{StringType, IntegerType}

scala> val student_rdd = sc.textFile("/home/acadgild/assignment/student_dataset")
student_rdd: org.apache.spark.rdd.RDD[String] = /home/acadgild/assignment/student_dataset MapPartitionsRDD[108] at textFile
at <console>:41

scala> val header = student_rdd.first()
header: String = name,subject,grade,marks,age

scala> val student_records = student_rdd.map(x=> x != header)
student_records: org.apache.spark.rdd.RDD[Boolean] = MapPartitionsRDD[109] at map at <console>:44

scala> println(student_records.count())
23

scala> case class Student(name:String, subject:String, grade:String, marks:Int, age:Int)
defined class Student

scala> val student_df = student_records.map(_._split(",")).map(x => Student(x(0).toString, x(1).toString, x(2).toString, x(3)
.toInt, x(4).toInt)).toDF
<console>:44: error: value split is not a member of Boolean
    val student_df = student_records.map(_._split(",")).map(x => Student(x(0).toString, x(1).toString, x(2).toString, x(3)
.toInt, x(4).toInt)).toDF
                                         ^

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

scala> spark.sqlContext.sql("SELECT COUNT(*) from (SELECT DISTINCT subject FROM Student)").show()
+-----+
|count(1)|
+-----+
|        3|
+-----+

scala> spark.sqlContext.sql("SELECT COUNT(*) from (SELECT name FROM Student WHERE name='Mathew' and marks=55)").show()
+-----+
|count(1)|
+-----+
|        2|
+-----+

scala>
```

Task 2: (Process Air Travelers dataset)

Step1: Create a temporary table User

Read dataset from /home/acadgild/assignment_18.1/S18_Dataset_User_details.txt and create RDD user_rdd. Create case class User with field user_id, name. Create dataframe user_df by mapping records splitting fields by and populating the User class object. Next create temporary table User

Code is as below:

```
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")

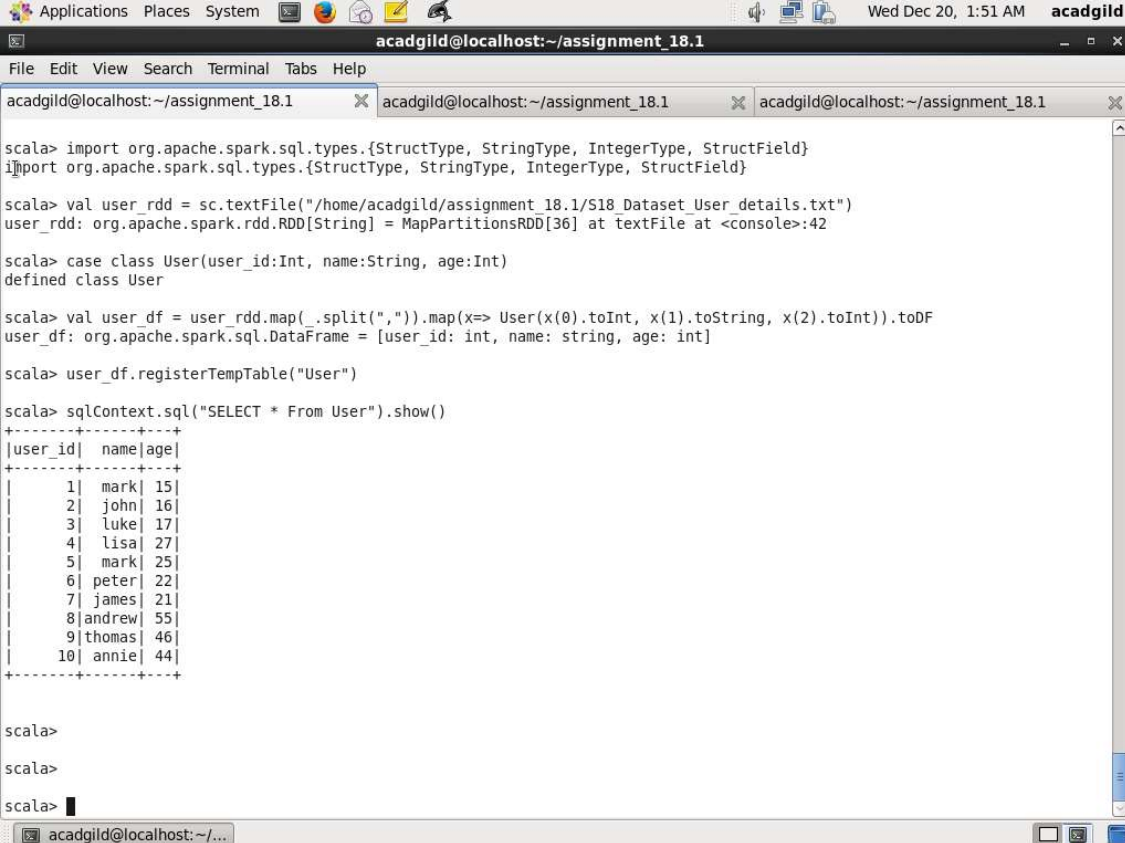
case class User(user_id:Int, name:String, age:Int)

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

user_df.registerTempTable("User")

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

Screenshot is as below:



The screenshot shows a terminal window titled "acadgild@localhost: ~/assignment_18.1". The terminal displays the following code and output:

```
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/assignment_18.1/S18_Dataset_User_details.txt")
user_rdd: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[36] at textFile at <console>:42

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, age: int]

scala> user_df.registerTempTable("User")

scala> 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>
scala>
scala>
```

Step2: Create a temporary table Travel

Read dataset from /home/acadgild/assignment_18.1/S18_Dataset_Holidays.txt and create RDD user_rdd. Create case class Travel with field user_id, src, dest, travel_mode distance, year_of_travel Create dataframe travel_df by mapping records splitting fields by , and populating the Travel class object. Next create temporary table Travel

Code is as below:

```
val travel_rdd = sc.textFile("/home/acadgild/assignment_18.1/S18_Dataset_Holidays.txt")

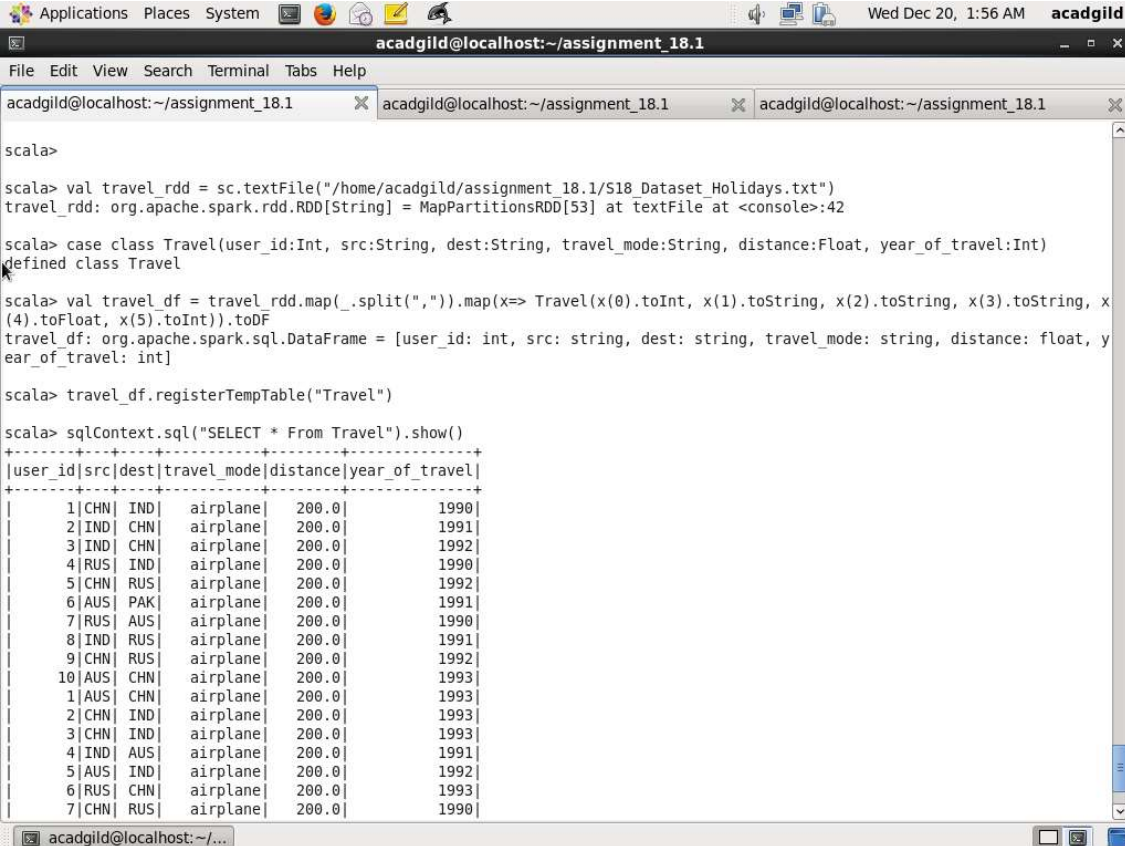
case class Travel(user_id:Int, src:String, dest:String, travel_mode:String, distance:Float,
year_of_travel:Int)

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")

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

Screenshot is as below:



The screenshot shows a terminal window with the following content:

```
scala>
scala> val travel_rdd = sc.textFile("/home/acadgild/assignment_18.1/S18_Dataset_Holidays.txt")
travel_rdd: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[53] at TextFile at <console>:42

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, dest: string, travel_mode: string, distance: float, y
ear_of_travel: int]

scala> travel_df.registerTempTable("Travel")

scala> 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|
```

Step3:

Read dataset from /home/acadgild/assignment_18.1/S18_Dataset_Transport.txt and create RDD transport_rdd. Create case class Transport with fields travel_mode, cost_per_unit. Create dataframe transport_df by mapping records splitting fields by , and populating the Transport class object. Next create temporary table Transport

```
val transport_rdd = sc.textFile("/home/acadgild/assignment_18.1/S18_Dataset_Transport.txt")

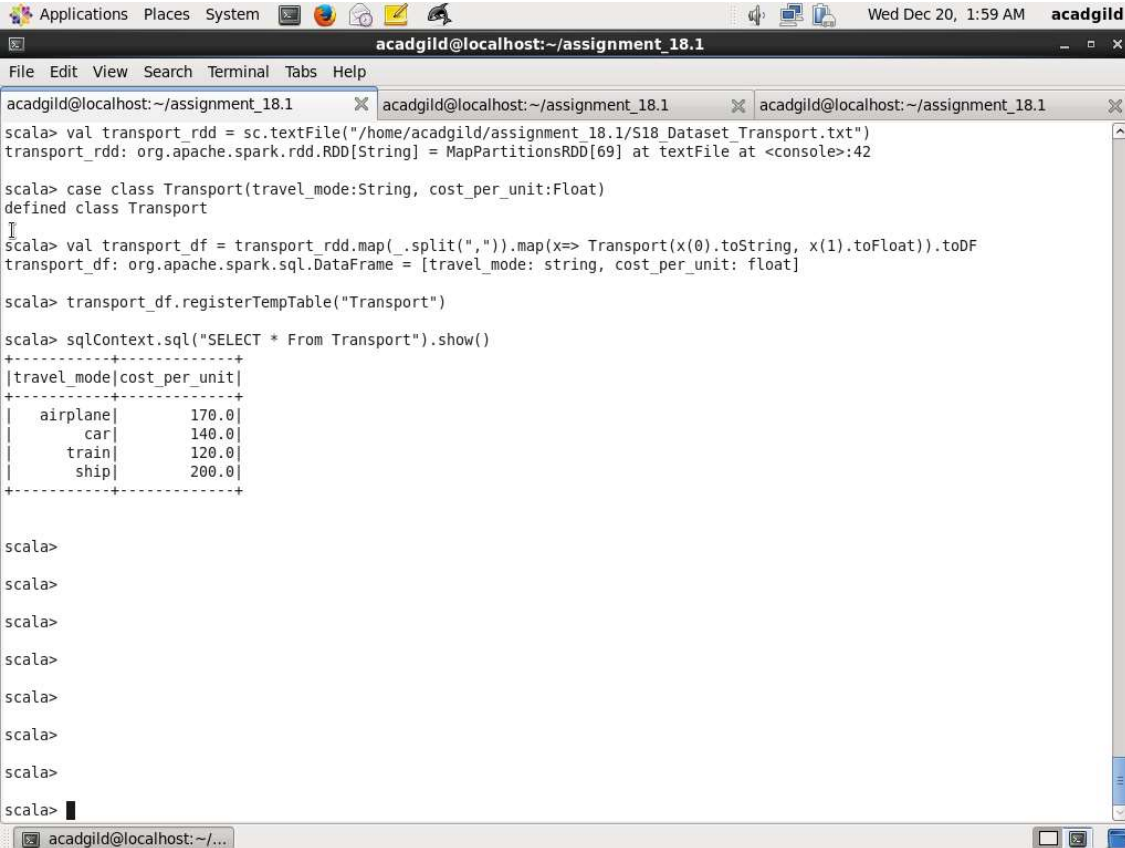
case class Transport(travel_mode:String, cost_per_unit:Float)

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

transport_df.registerTempTable("Transport")

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

Screenshot is as below:



The screenshot shows a terminal window titled 'acadgild@localhost:~/assignment_18.1'. The terminal displays the following Scala code and its output:

```
scala> val transport_rdd = sc.textFile("/home/acadgild/assignment_18.1/S18_Dataset_Transport.txt")
transport_rdd: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[69] at TextFile at <console>:42

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")

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

Below the output, there are several empty prompts for the Scala shell:

```
scala>
scala>
scala>
scala>
scala>
scala>
scala>
```

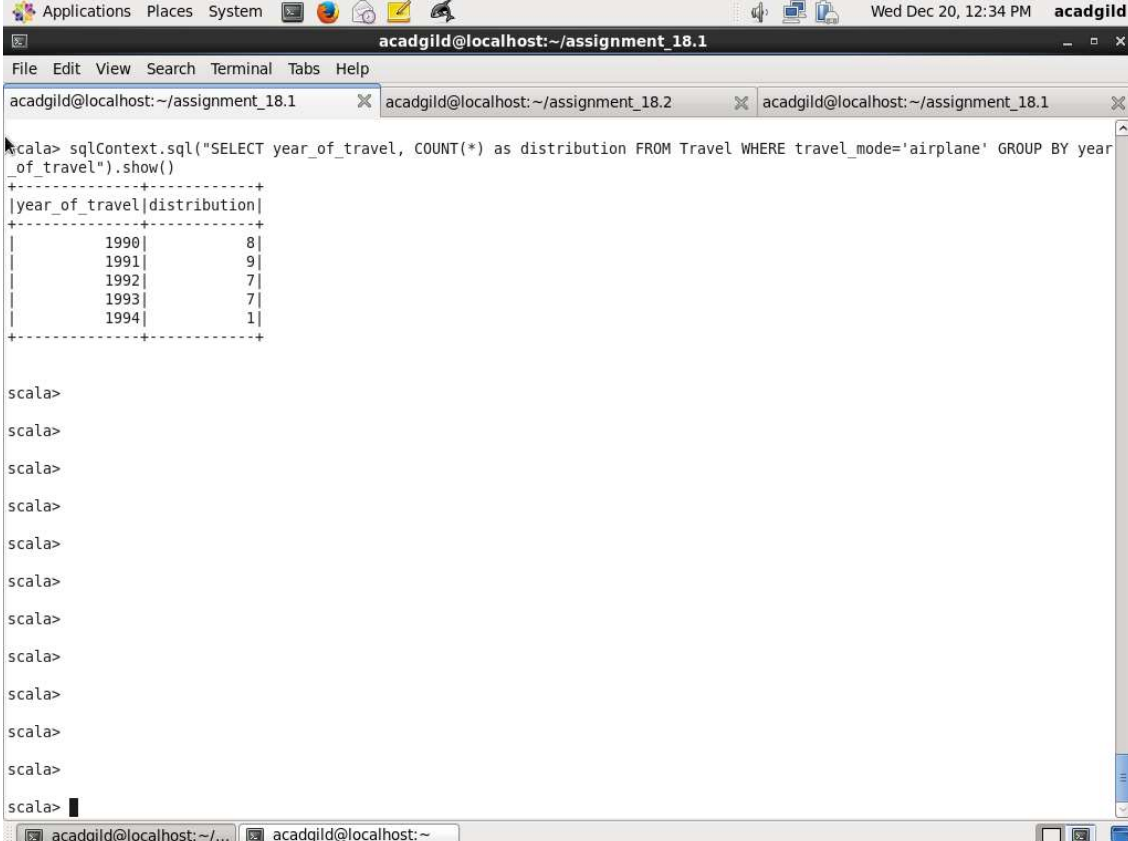
Task1: Distribution of total number of air travelers per year

Here using count(*) number of travels grouped by year calculated

Code is as below:

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

Screenshot is as below:



The screenshot shows a Scala REPL window titled 'acadgild@localhost:~/assignment_18.1'. The window contains the following text:

```
scala> 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
1991	9
1992	7
1993	7
1994	1

Below the table, the prompt 'scala>' is repeated multiple times, indicating the REPL is ready for further input.

Task2: Total air distance covered by each user per year

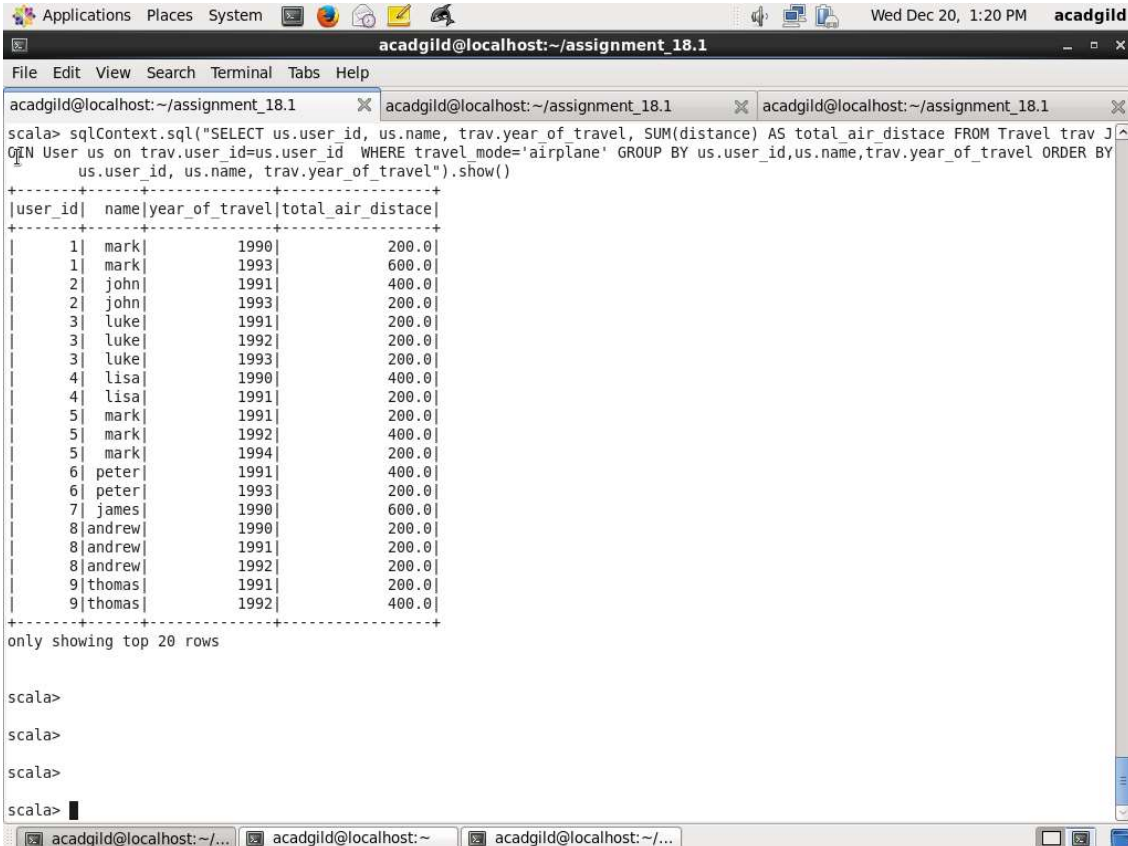
Here all the distance are summed using SUM function group by user_id by joining the temporary tables Travel and User on user_id column

Code is as below:

```
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()
```

(NOTE: I have used both user_id and user_name as there are two different users who have same name but different id. For example, user_id 1 and 5 both have name as Mark)

Screenshot is as below:



The screenshot shows a Scala REPL window titled 'acadgild@localhost: ~/assignment_18.1'. The user has executed a SQL query to calculate the total air distance for each user, grouped by user_id, name, and year of travel. The results are displayed in a table format, showing the first 20 rows. The table has four columns: user_id, name, year_of_travel, and total_air_distace. The data shows that users with the same name (like Mark) have different user_ids and travel in different years, resulting in different total distances.

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>
scala>
scala>
scala>

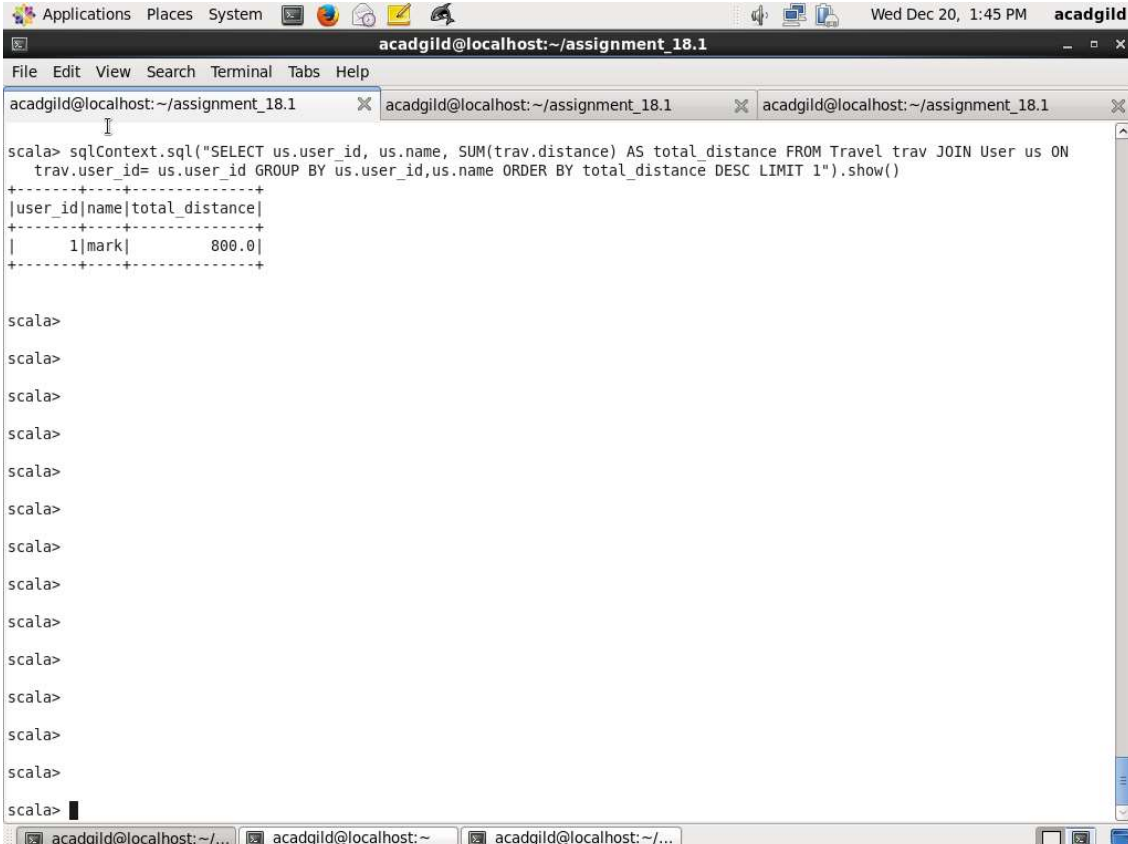
Task3: Which user has travelled largest distance till date

Here all the distance are summed using SUM function group by name, user_id by joining tables Travel, User column user_id then sorted in descending order by total_distance and first record is taken using LIMIT 1

Code is as below:

```
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()
```

Screenshot is as below:



The screenshot shows a Scala REPL window titled "acadgild@localhost: ~/assignment_18.1". The window contains the following code and output:

```
scala> 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

Below the output, the prompt "scala>" is repeated multiple times, indicating the REPL is ready for further input.

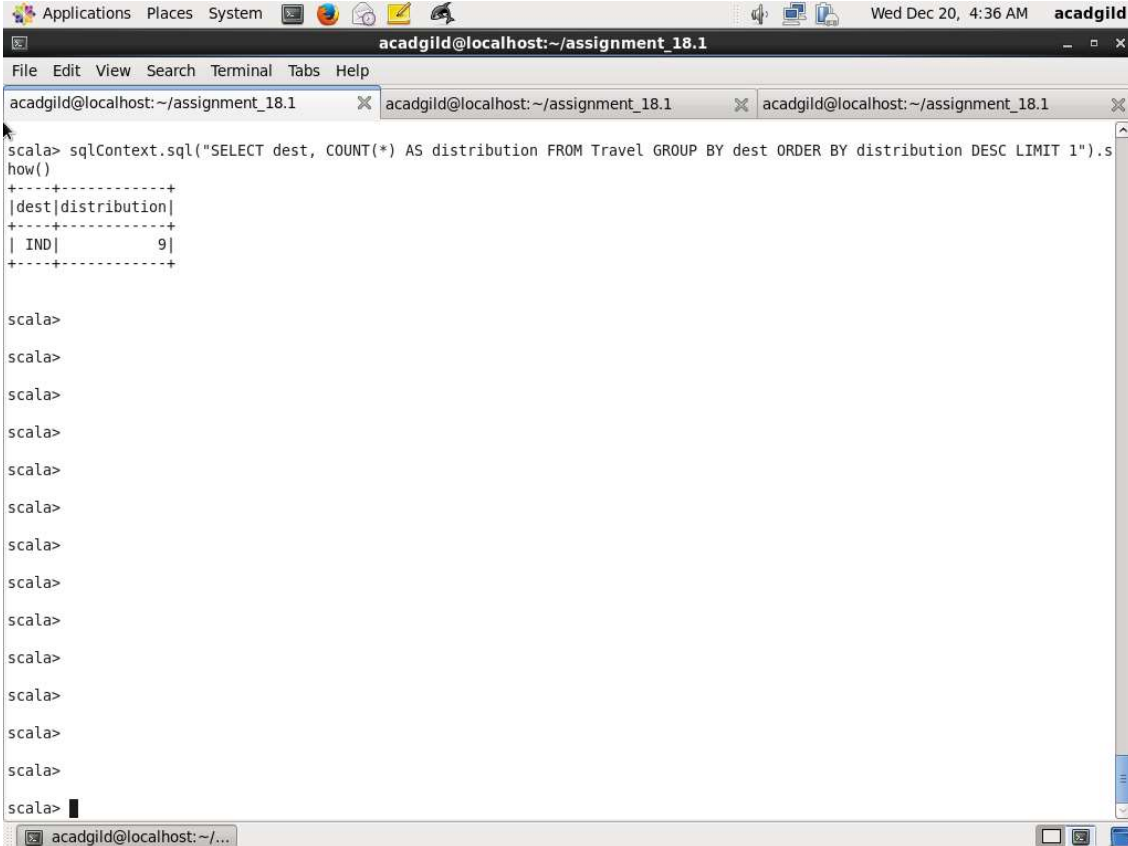
Task4: What is the most preferred destination of all users

Here using count(*) number of travels as distributed grouped by dest calculated and sorted by distribution and first record is taken

Code is as below:

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

Screenshot is as below:



The screenshot shows a terminal window titled "acadgild@localhost: ~/assignment_18.1". The terminal displays the following Scala code and its output:

```
scala> 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, there are several lines of "scala>" prompts, indicating that the user has entered multiple commands but only the first one has been executed and shown.

Task 3 (Process Holiday dataset)

Initial Steps:

Step1: Create a temporary table User

Read dataset from /home/acadgild/assignment_18.1/S18_Dataset_User_details.txt and create RDD user_rdd. Create case class User with field user_id, name. Create dataframe user_df by mapping records splitting fields by and populating the User class object. Next create temporary table User

Code is as below:

```
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")

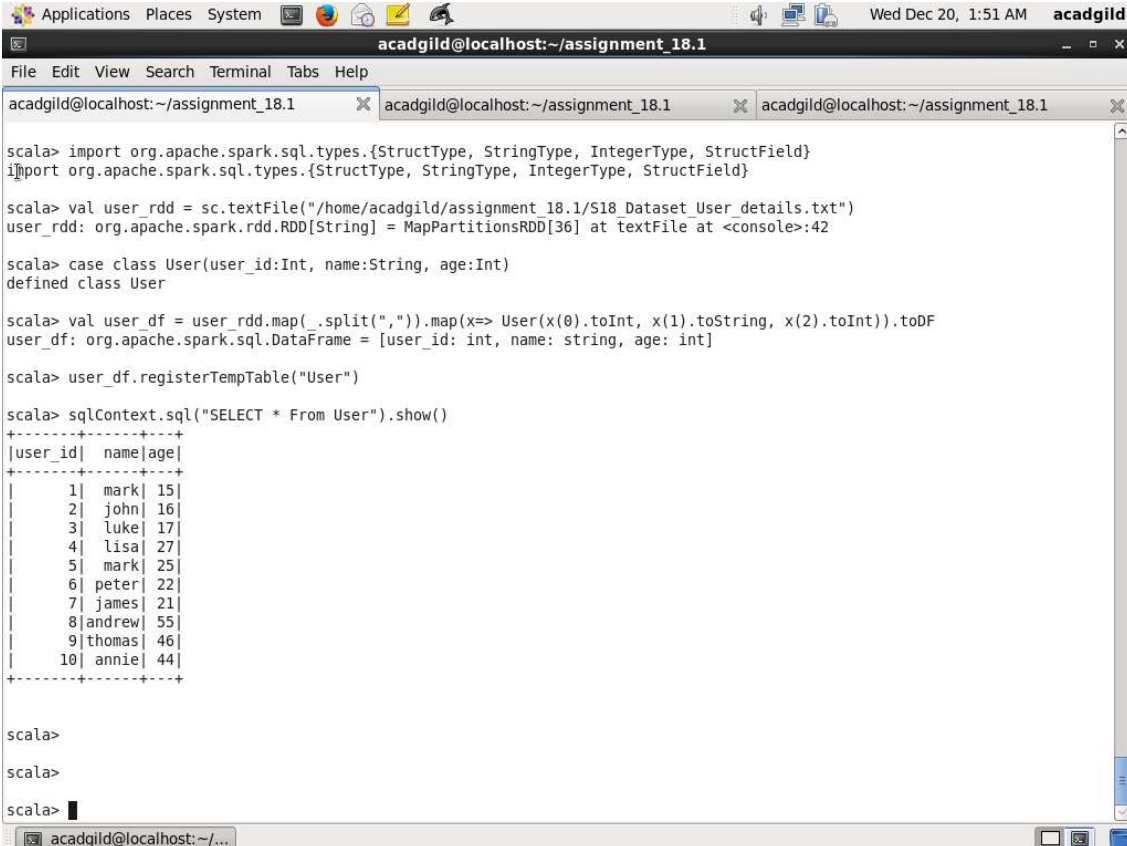
case class User(user_id:Int, name:String, age:Int)

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

user_df.registerTempTable("User")

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

Screenshot is as below:



The screenshot shows a terminal window titled 'acadgild@localhost:~/assignment_18.1'. The terminal displays the following Scala code and its output:

```
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/assignment_18.1/S18_Dataset_User_details.txt")
user_rdd: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[36] at textFile at <console>:42

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, age: int]

scala> user_df.registerTempTable("User")

scala> 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|
+-----+-----+
```

The terminal window also shows the Scala prompt 'scala>' at the bottom.

Step2: Create a temporary table Travel

Read dataset from /home/acadgild/assignment_18.1/S18_Dataset_Holidays.txt and create RDD user_rdd. Create case class Travel with field user_id, src, dest, travel_mode distance, year_of_travel. Create dataframe travel_df by mapping records splitting fields by , and populating the Travel class object. Next create temporary table Travel

Code is as below:

```
val travel_rdd = sc.textFile("/home/acadgild/assignment_18.1/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")

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

Screenshot is as below:

```
scala>
scala> val travel_rdd = sc.textFile("/home/acadgild/assignment_18.1/S18_Dataset_Holidays.txt")
travel_rdd: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[53] at TextFile at <console>:42

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, dest: string, travel_mode: string, distance: float, year_of_travel: int]

scala> travel_df.registerTempTable("Travel")

scala> 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|
```

Step3:

Read dataset from /home/acadgild/assignment_18.1/S18_Dataset_Transport.txt and create RDD transport_rdd. Create case class Transport with fields travel_mode, cost_per_unit. Create dataframe transport_df by mapping records splitting fields by , and populating the Transport class object. Next create temporary table Transport

```
val transport_rdd = sc.textFile("/home/acadgild/assignment_18.1/S18_Dataset_Transport.txt")

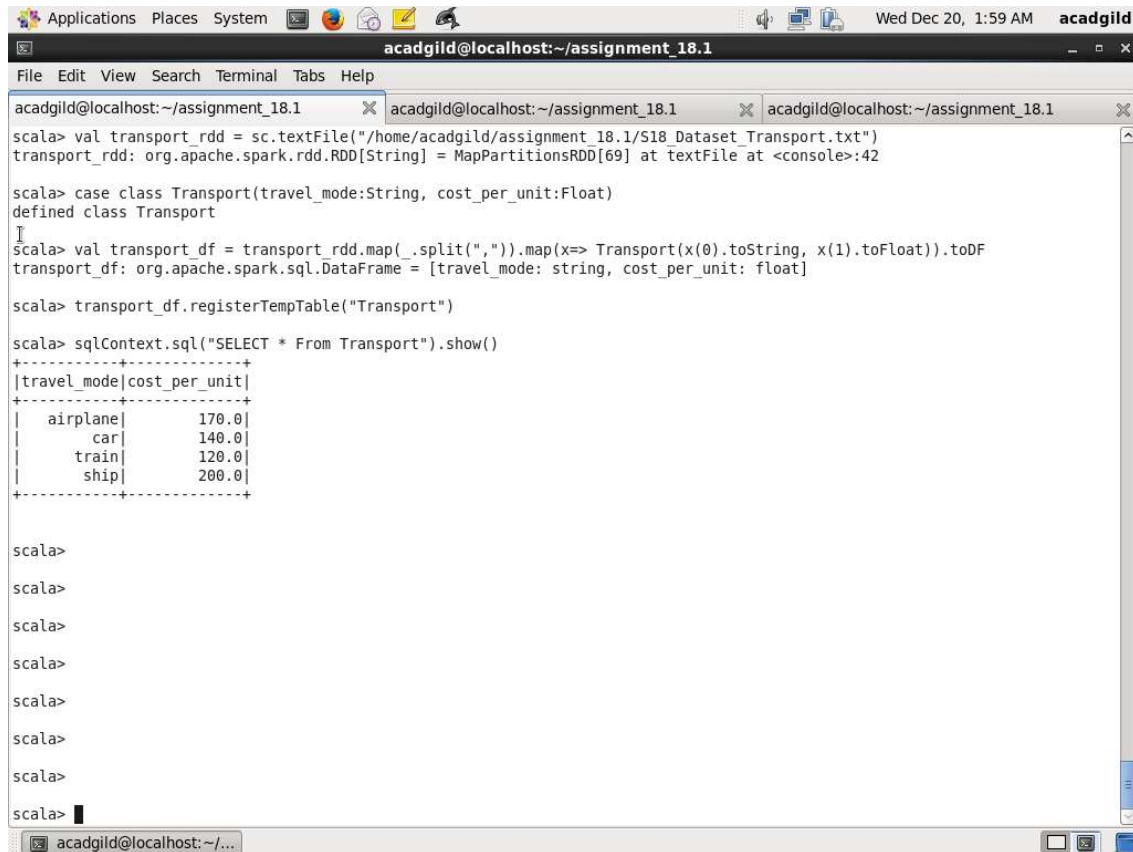
case class Transport(travel_mode:String, cost_per_unit:Float)

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

transport_df.registerTempTable("Transport")

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

Screenshot is as below:



```
acadgild@localhost:~/assignment_18.1
File Edit View Search Terminal Tabs Help
acadgild@localhost:~/assignment_18.1
scala> val transport_rdd = sc.textFile("/home/acadgild/assignment_18.1/S18 Dataset_Transport.txt")
transport_rdd: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[69] at textFile at <console>:42

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")

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

scala>
scala>
scala>
scala>
scala>
scala>
scala>
scala>
```

Task1: Which route is generating the most revenue per year

Note: This is a complex query as view was not working in this installation of acadgild VM. I first find the revenue by route per year and create a alias revenue_by_route_per_year. Next I find the maximum revenue per year and create alias max_revenue_per_year. Next I join both the table aliases revenue_by_route_per_year and max_revenue_per_year on year_of_travel and total_revenue and order them by year

Code is as below:

```
sqlContext.sql("SELECT revenue_by_route_per_year.year_of_travel,
revenue_by_route_per_year.src, revenue_by_route_per_year.dest,
revenue_by_route_per_year.total_revenue FROM (SELECT trav.year_of_travel, trav.src, trav.dest,
sum(trans.cost_per_unit) AS total_revenue FROM Travel trav JOIN Transport trans ON
trav.travel_mode= trans.travel_mode GROUP BY trav.year_of_travel, trav.src, trav.dest)
revenue_by_route_per_year, (SELECT year_of_travel, max(total_revenue) AS total_revenue FROM (SELECT
trav.year_of_travel, trav.src, trav.dest, sum(trans.cost_per_unit) AS total_revenue FROM
Travel trav JOIN Transport trans ON trav.travel_mode= trans.travel_mode GROUP BY
trav.year_of_travel, trav.src, trav.dest) travel_revenue_per_year2 GROUP BY year_of_travel)
max_revenue_per_year WHERE revenue_by_route_per_year.year_of_travel =
max_revenue_per_year.year_of_travel AND
revenue_by_route_per_year.total_revenue=max_revenue_per_year.total_revenue ORDER BY
revenue_by_route_per_year.year_of_travel").show()
```

Screenshot is as below:

Screenshot is as below:

```
scala> sqlContext.sql("SELECT revenue_by_route_per_year.year_of_travel, revenue_by_route_per_year.src, revenue_by_route_per_year.dest, revenue_by_route_per_year.total_revenue FROM (SELECT trav.year_of_travel, trav.src, trav.dest, sum(trans.cost_per_unit) AS total_revenue FROM Travel trav JOIN Transport trans ON trav.travel_mode= trans.travel_mode GROUP BY trav.year_of_travel, trav.src, trav.dest) revenue_by_route_per_year, (SELECT year_of_travel, max(total_revenue) AS total_revenue FROM (SELECT trav.year_of_travel, trav.src, trav.dest, sum(trans.cost_per_unit) AS total_revenue FROM Travel trav JOIN Transport trans ON trav.travel_mode= trans.travel_mode GROUP BY trav.year_of_travel, trav.src, trav.dest) travel_revenue_per_year2 GROUP BY year_of_travel) max_revenue_per_year WHERE revenue_by_route_per_year.year_of_travel = max_revenue_per_year.year_of_travel AND revenue_by_route_per_year.total_revenue=max_revenue_per_year.total_revenue ORDER BY revenue_by_route_per_year.year_of_travel").show()
+-----+-----+-----+-----+
|year_of_travel|src|dest|total_revenue|
+-----+-----+-----+-----+
|1990|CHN|IND|340.0|
|1991|IND|RUS|340.0|
|1991|IND|AUS|340.0|
|1992|CHN|RUS|340.0|
|1992|RUS|IND|340.0|
|1993|CHN|IND|340.0|
|1993|AUS|CHN|340.0|
|1994|CHN|PAK|170.0|
+-----+-----+-----+-----+

scala>
scala>
scala>
scala>
scala>
scala>
scala>
```

Task2: What is the total amount spent by each user on air travel per year

Here I have joined temporary tables Travel, User, Transport using the common fields (travel_mode between Travel and Transport, user_id between Travel and User) and using sum function on cost_per_unit group by user_id, name, with travel_mode as airplane

Code is as below:

```
sqlContext.sql("SELECT us.user_id, us.name, trav.year_of_travel, sum(trans.cost_per_unit) AS total_amount_spent FROM Travel trav JOIN Transport trans ON trav.travel_mode=trans.travel_mode JOIN user us ON trav.user_id=us.user_id WHERE trav.travel_mode= 'airplane' GROUP BY us.user_id, us.name, trav.year_of_travel ORDER BY us.user_id").show()
```

(NOTE: I have used both user_id and user_name as there are two different users who have same name but different id. For example, user_id 1 and 5 both have name as Mark)

Screenshot is as below:

```
acadgild@localhost: ~/assignment_18.1
File Edit View Search Terminal Tabs Help
acadgild@localhost: ~/assignment_18.1 acadgild@localhost: ~/assignment_18.2 acadgild@localhost: ~/assignment_18.1
scala> sqlContext.sql("SELECT us.user_id, us.name, trav.year_of_travel, sum(trans.cost_per_unit) AS total_amount_spent FROM T
ravel trav JOIN Transport trans ON trav.travel_mode= trans.travel_mode JOIN user us ON trav.user_id=us.user_id WHERE
trav.travel_mode= 'airplane' GROUP BY us.user_id, us.name, trav.year_of_travel ORDER BY us.user_id").show()
+-----+-----+-----+-----+
|user_id| name|year_of_travel|total_amount_spent|
+-----+-----+-----+-----+
|1| mark|1990|170.0|
|1| mark|1993|510.0|
|2| john|1991|340.0|
|2| john|1993|170.0|
|3| luke|1993|170.0|
|3| luke|1991|170.0|
|3| luke|1992|170.0|
|4| lisa|1991|170.0|
|4| lisa|1990|340.0|
|5| mark|1994|170.0|
|5| mark|1991|170.0|
|5| mark|1992|340.0|
|6| peter|1991|340.0|
|6| peter|1993|170.0|
|7| james|1990|510.0|
|8| andrew|1990|170.0|
|8| andrew|1992|170.0|
|8| andrew|1991|170.0|
|9| thomas|1992|340.0|
|9| thomas|1991|170.0|
+-----+-----+-----+-----+
only showing top 20 rows

scala>
scala>
scala>
scala>
```

Task3: Considering age group < 20, 20-35 , > 35. Which age group has travelled most every yeat

Using CASE WHEN first filter the age groups (<20, 20-35, > 35) joining tables User and Travel on common field user_id. Next find count number of travels per year per age group and create a table alias just_travel_count. Also found the maximum count of number of travels per year and create table alias max_travel_count. Next join these two aliases on field on travel_count and year_of_travel

(Note: As view is not working in this setup, I have to use complex query to solve this problem)

Code is as below:

```
sqlContext.sql("SELECT DISTINCT just_travel_count.year_of_travel, just_travel_count.age_group
FROM (SELECT age_group_count.year_of_travel, age_group_count.age_group, COUNT(*) AS
travel_count FROM (SELECT trav.year_of_travel, CASE WHEN us.age < 20 THEN '< 20' WHEN age >=
20 AND age <= 35 THEN '20-35' WHEN age >35 THEN '> 35' END AS age_group FROM Travel trav
JOIN User us ON trav.user_id=us.user_id) age_group_count GROUP BY
age_group_count.year_of_travel, age_group_count.age_group) just_travel_count, (SELECT
year_agegroup_travel_count.year_of_travel, max(year_agegroup_travel_count.travel_count) AS
travel_count FROM (SELECT age_group_count.year_of_travel AS
year_of_travel, age_group_count.age_group, COUNT(*) travel_count FROM (SELECT
trav.year_of_travel, CASE WHEN us.age < 20 THEN '< 20' WHEN age >= 20 AND age <= 35 THEN '20-
35' WHEN age >35 THEN '> 35' END AS age_group FROM Travel trav JOIN User us ON
trav.user_id=us.user_id) age_group_count GROUP BY age_group_count.year_of_travel,
age_group_count.age_group) year_agegroup_travel_count GROUP BY
year_agegroup_travel_count.year_of_travel) max_travel_count WHERE
just_travel_count.year_of_travel = max_travel_count.year_of_travel AND
just_travel_count.travel_count = max_travel_count.travel_count ORDER BY
just_travel_count.year_of_travel").show()
```

Screenshot is as below:

