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)

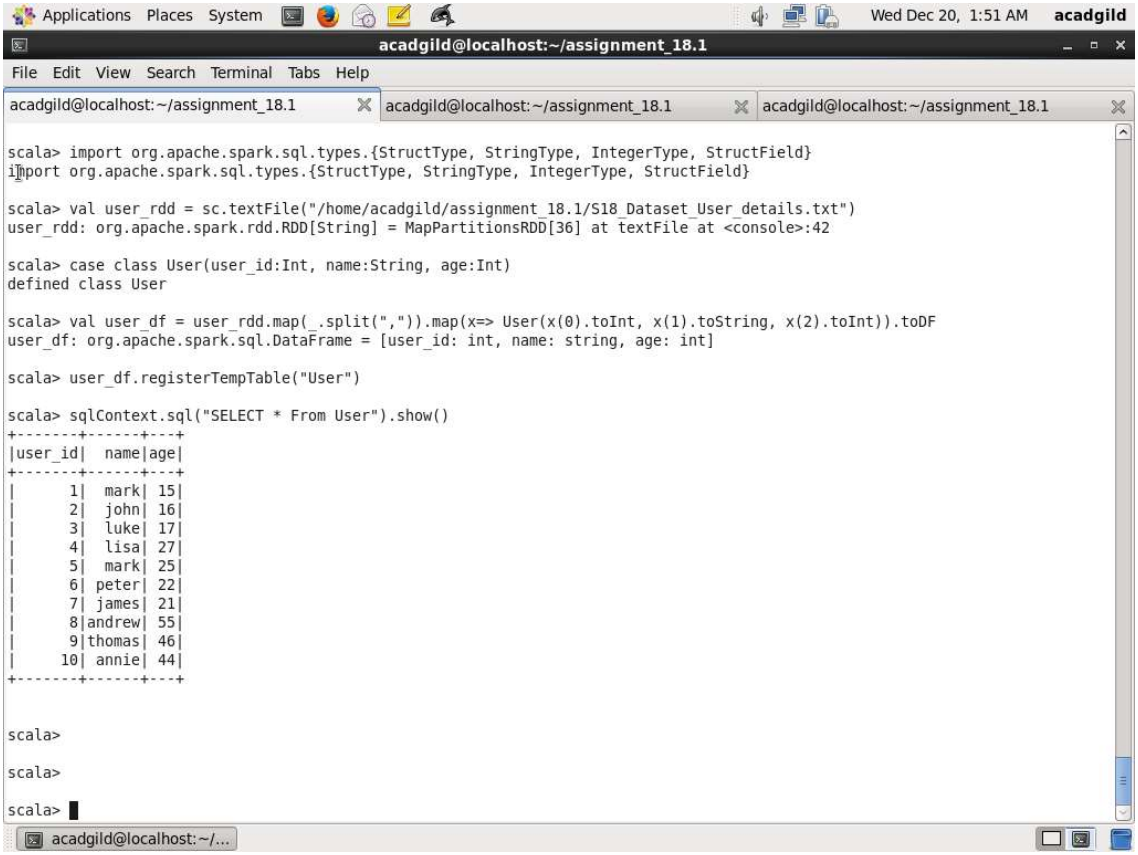
Step1: Create a temporary table User

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:



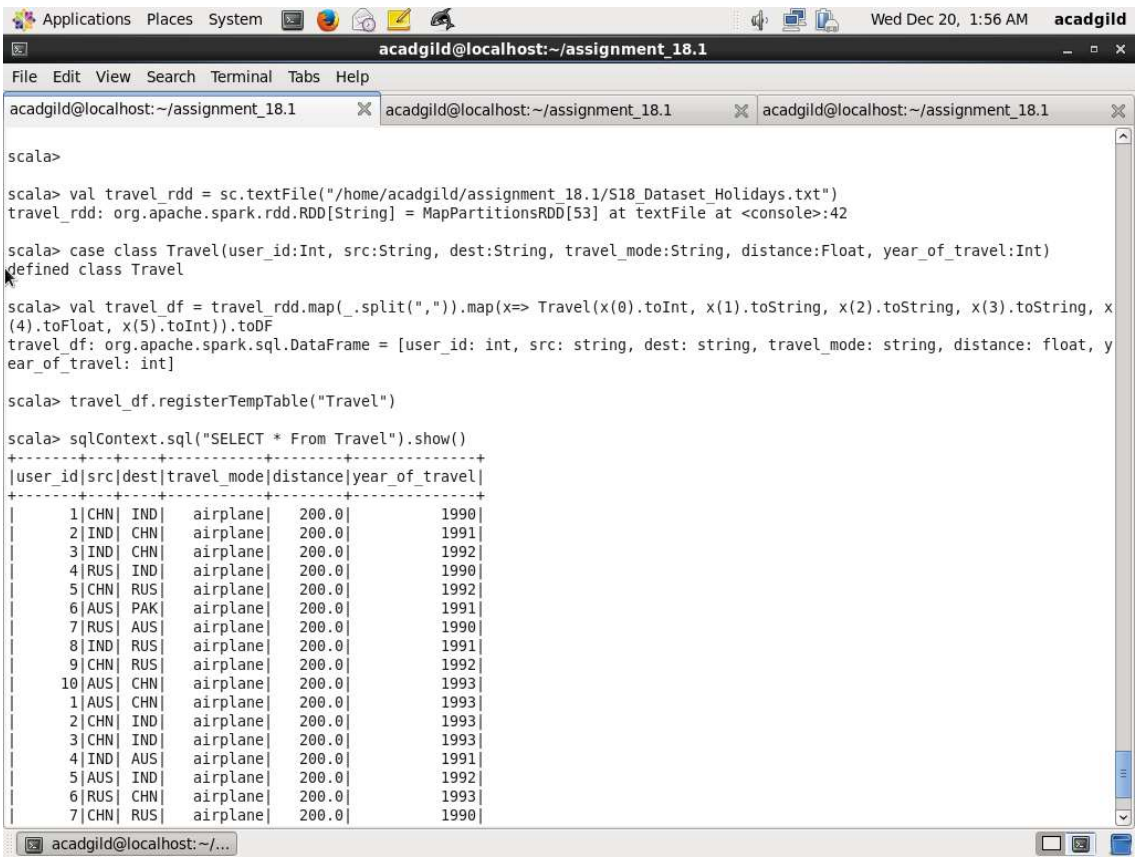
Step2: Create a temporary table Travel

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:

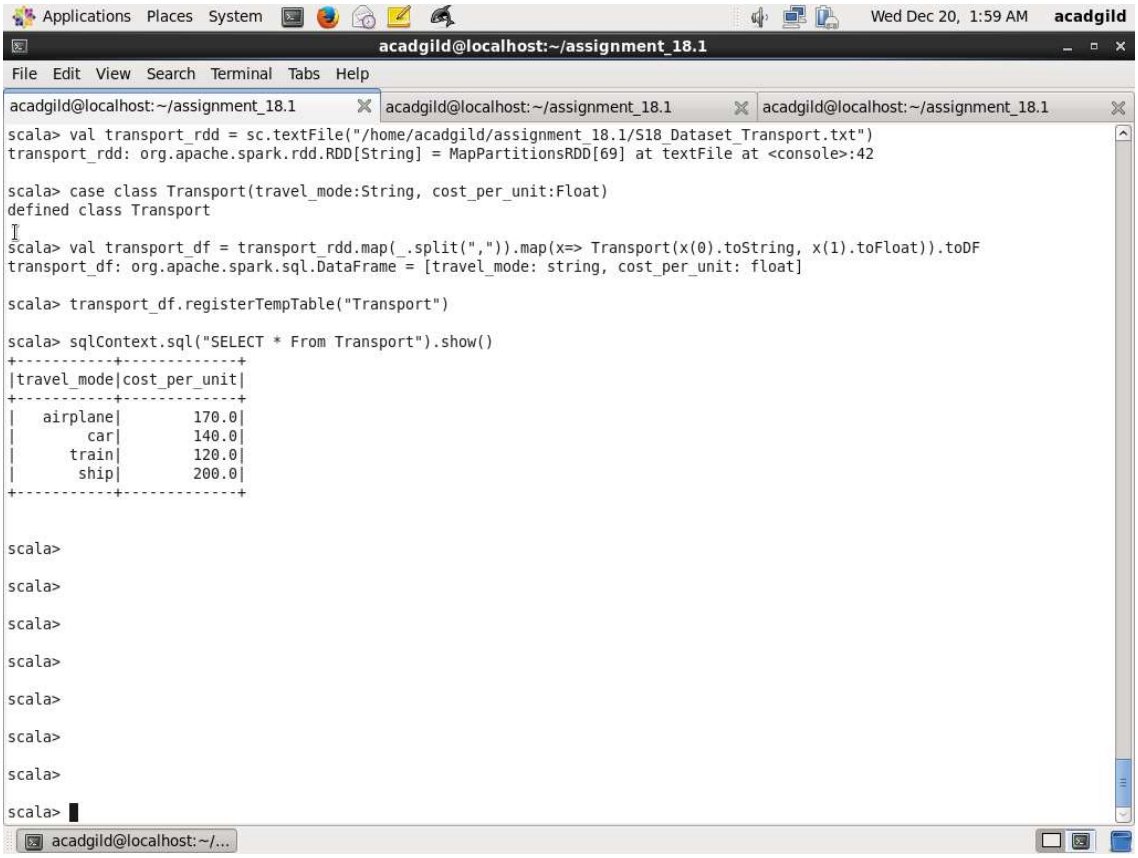


Step3:

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:



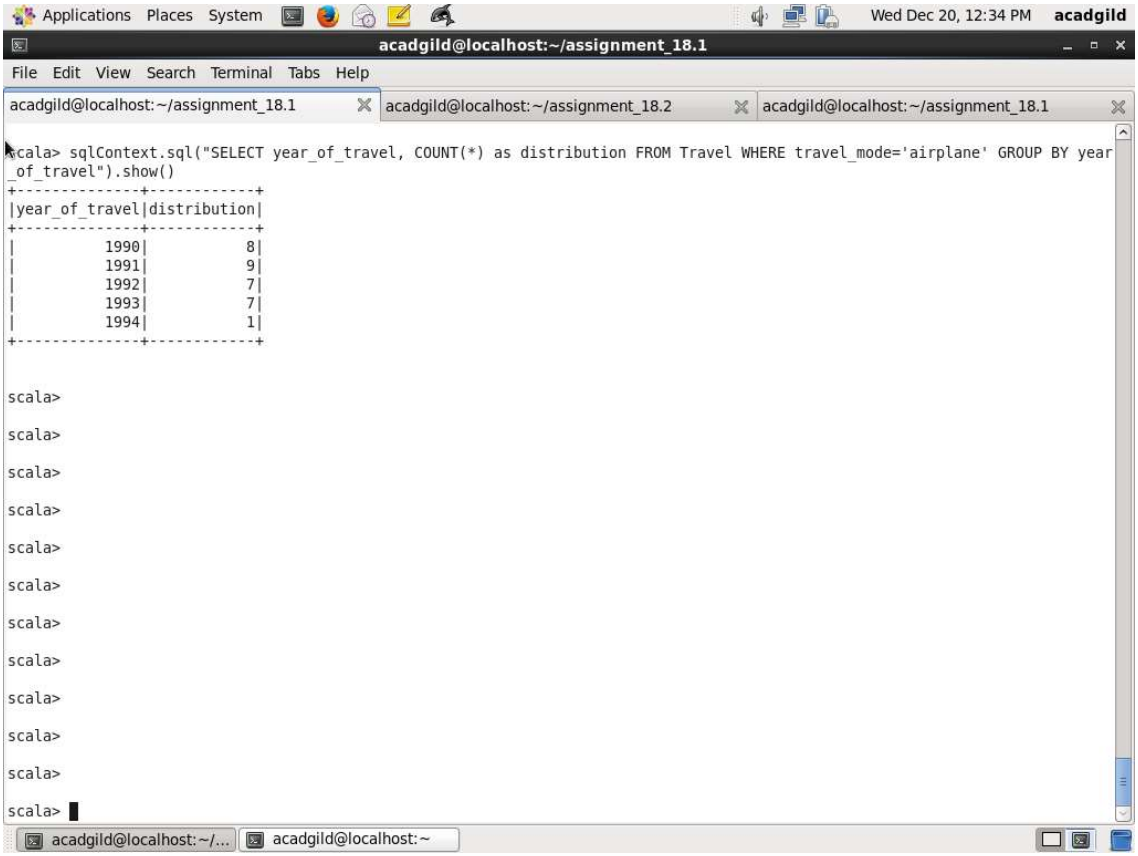
Task1: Distribution of total number of air travelers per year

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:



Task2: Total air distance covered by each user per year

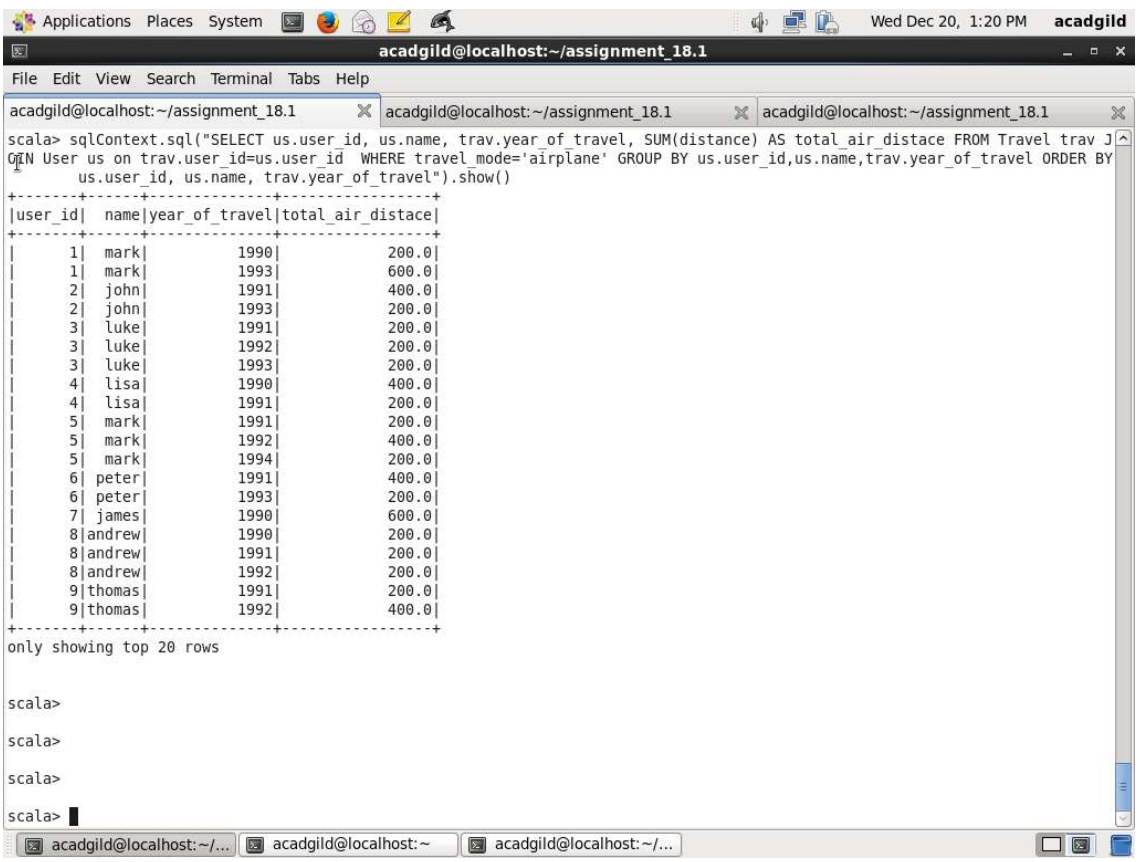
Here all the distance are summed using SUM function group by user_id by joining the temporary tables Travel and User on user_id column

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:



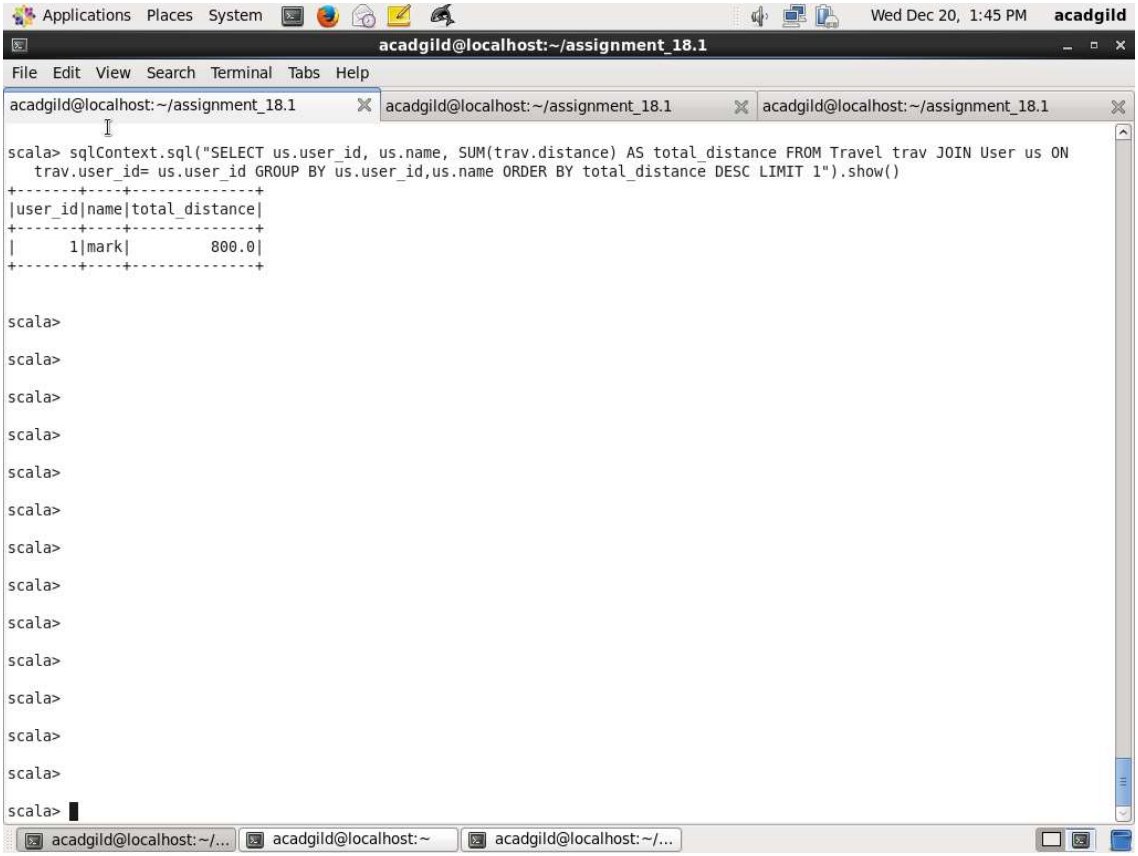
Task3: Which user has travelled largest distance till date

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_air_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:



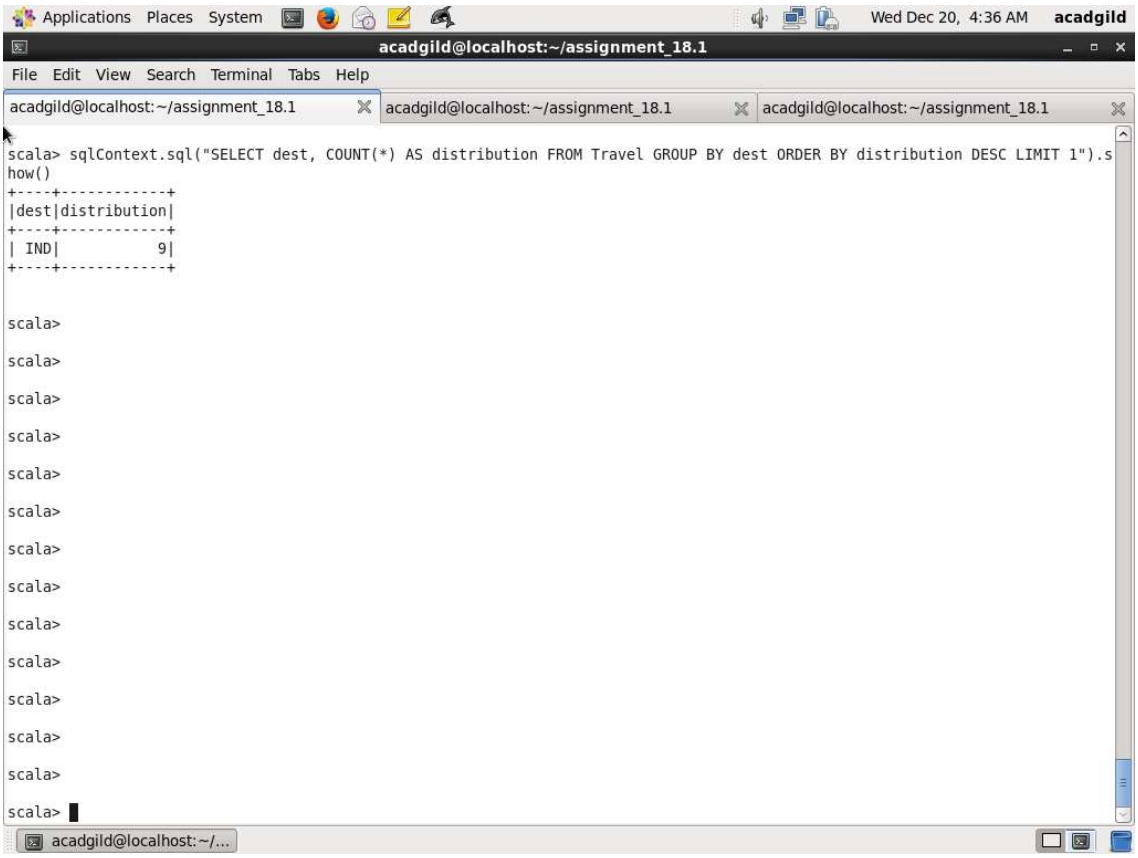
Task4: What is the most preferred destination of all users

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:



Task 3 (Process Holiday dataset)

Initial Steps:

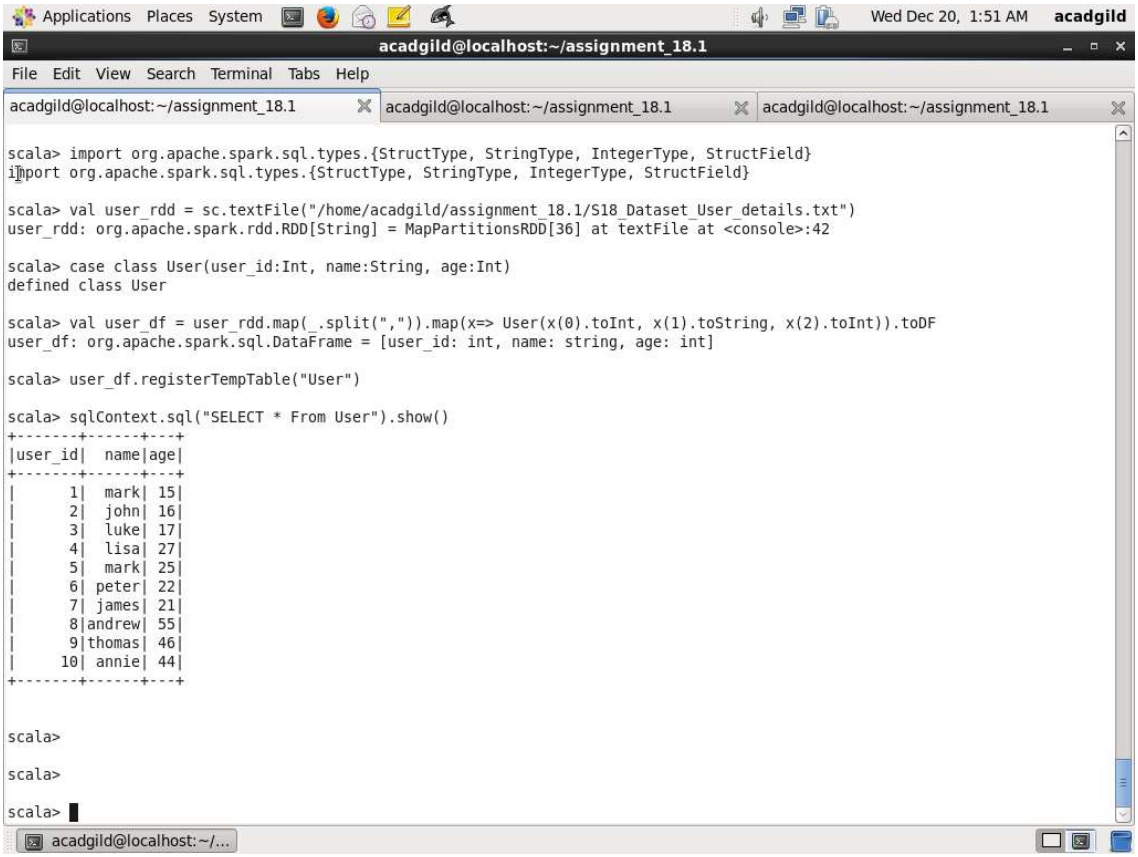
Step1: Create a temporary table User

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:



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, 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:

```
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>
```

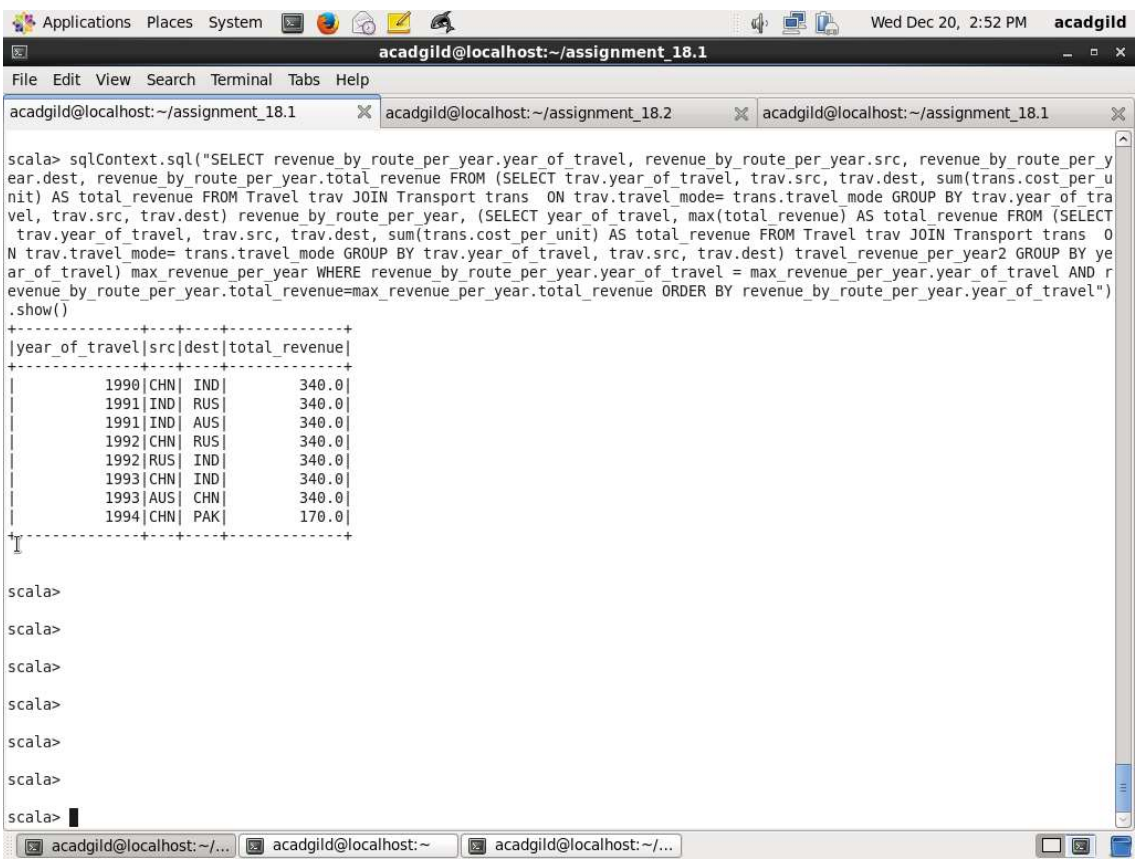
Task1: Which route is generating the most revenue per year

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:



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

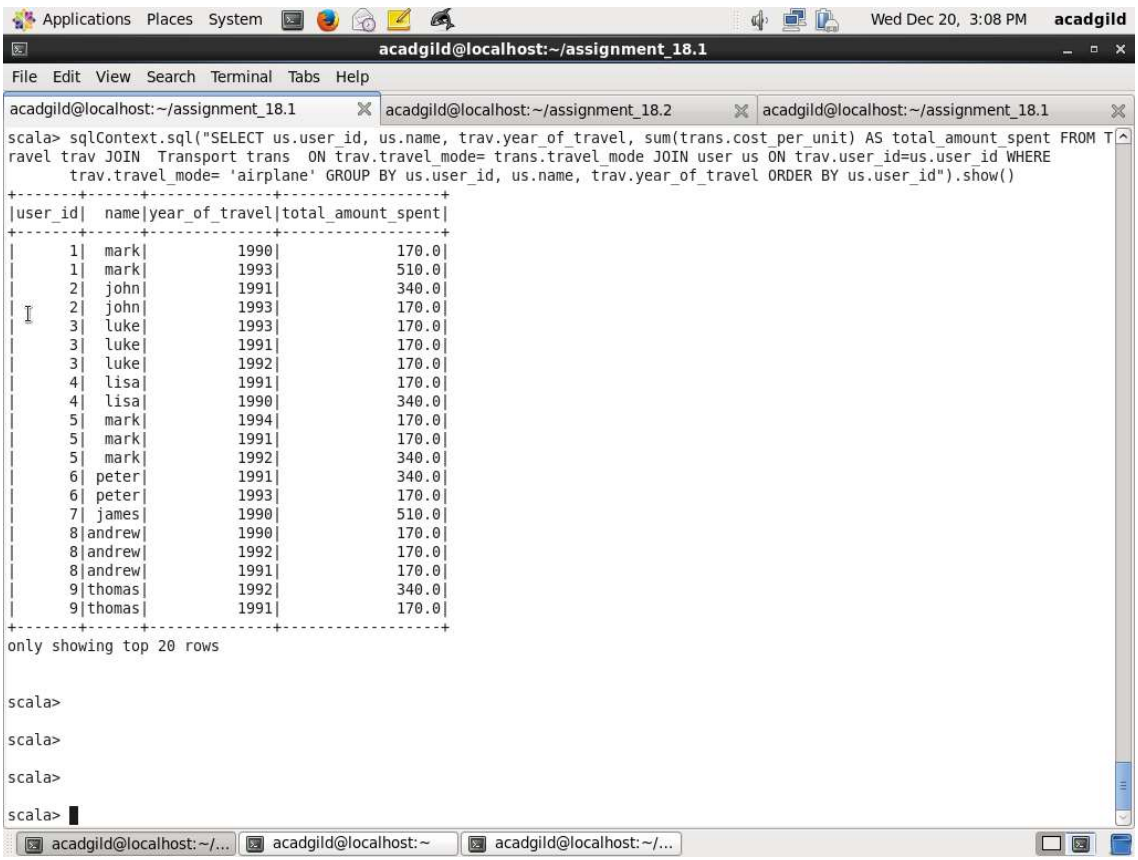
Here I have joined temporary tables Travel, User, Transport using the common fields (travel_mode between Travel and Transport, user_id between Travel and User) and using sum function on cost_per_unit group by user_id, name, with travel_mode as airplane

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:



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:

Applications Places System Wed Dec 20, 11:12 AM acadgild

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 DISTINCT just_travel_count.year_of_travel, just_travel_count.age_group FROM (SELECT age_group_c
ount.year_of_travel,age_group_count.age_group, COUNT(*) AS travel_count FROM (SELECT trav.year_of_travel,CASE WHEN us.age < 2
0 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_tra
vel_count, (SELECT year_agegroup_travel_count.year_of_travel, max(year_agegroup_travel_count.travel_count) AS  travel_count F
ROM (SELECT age_group_count.year_of_travel AS year_of_travel,age_group_count.age_group, COUNT(*) travel_count FROM (SELECT tr
av.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 A
S 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_trave
l, 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_cou
nt.travel_count ORDER BY just_travel_count.year_of_travel").show()
+--------------+---------+
|year_of_travel|age_group|
+--------------+---------+
|          1990|    20-35|
|          1991|    20-35|
|          1992|     > 35|
|          1993|     < 20|
|          1994|    20-35|
+--------------+---------+


scala>

scala>

scala>

scala>

scala>

scala>

scala>

scala>
```

acadgild@localhost:~/...