Task 1: (Process Student Dataset)

Problem Statement 1:

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

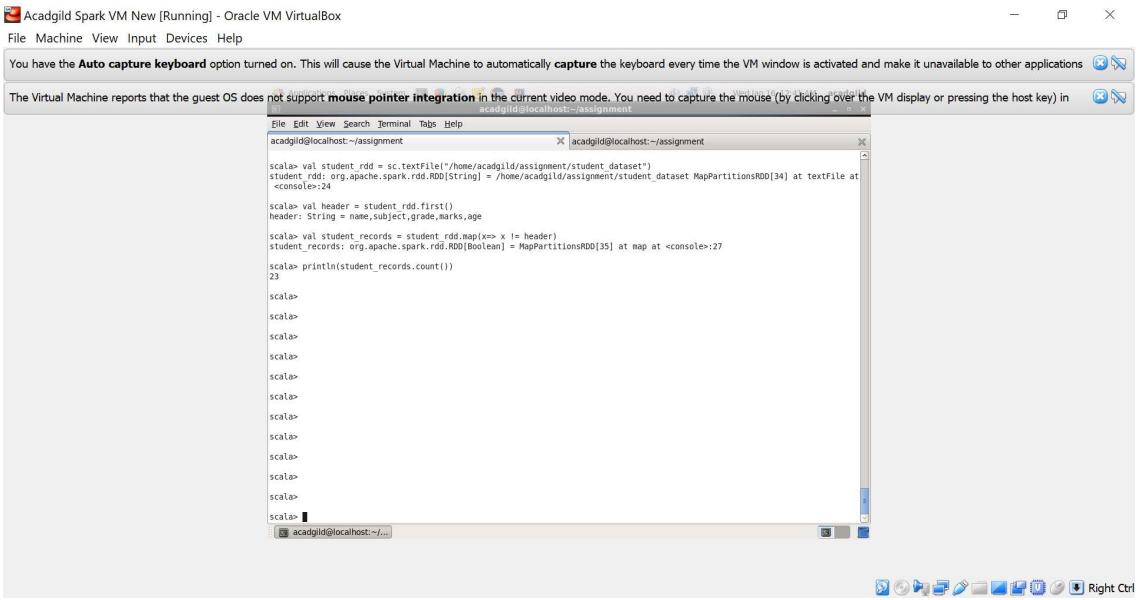
Steps: Read the dataset, get header, Filter the records which is not header

```
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())
```

Screenshots for 1 and 2 is as below:



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

Steps: Create a dataframe from the CSV file
Register a temporary table Student
Create SQL Query which will return count of distinct subjects

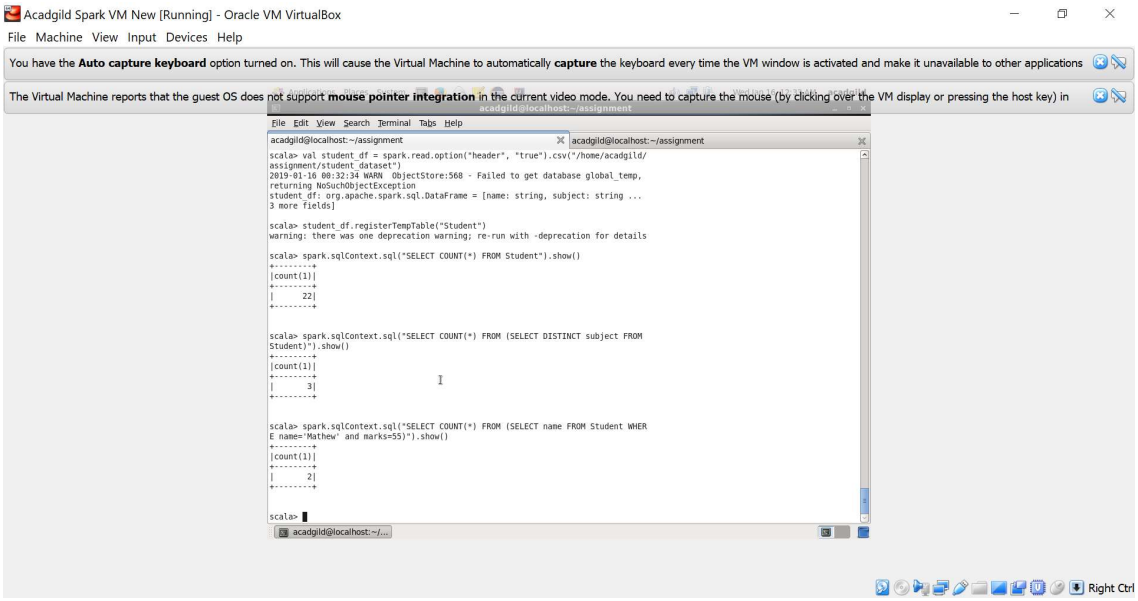
spark.sqlContext.sql("SELECT COUNT(*) FROM (SELECT DISTINCT subject FROM Student").show()

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()

Screenshots for 3 and 4 is as below



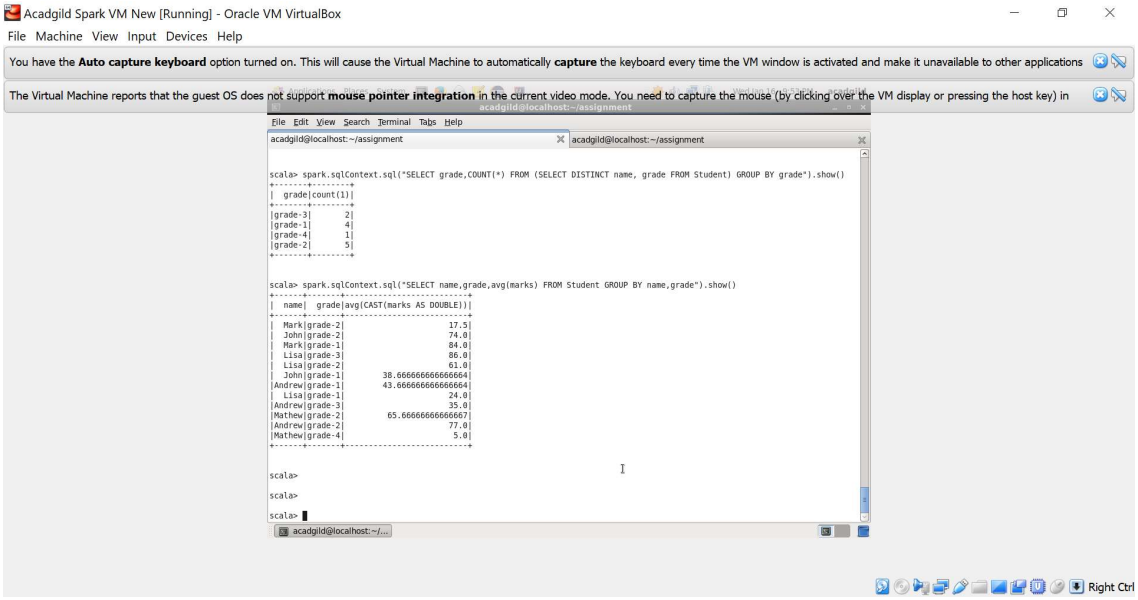
Problem Statement 2:

1. What is the count of students per grade in the school?
    - Create a inner query which will select distinct name, grade from Student
    - Create a outer query which will select count from the inner query group by grade

2. Find the average of each student (Note - Mathew is grade-1, is different from Mathew in some other grade!)

    - Create a query which will select average marks from Student group by name, grade

Screenshots of 1 and 2 are as below:

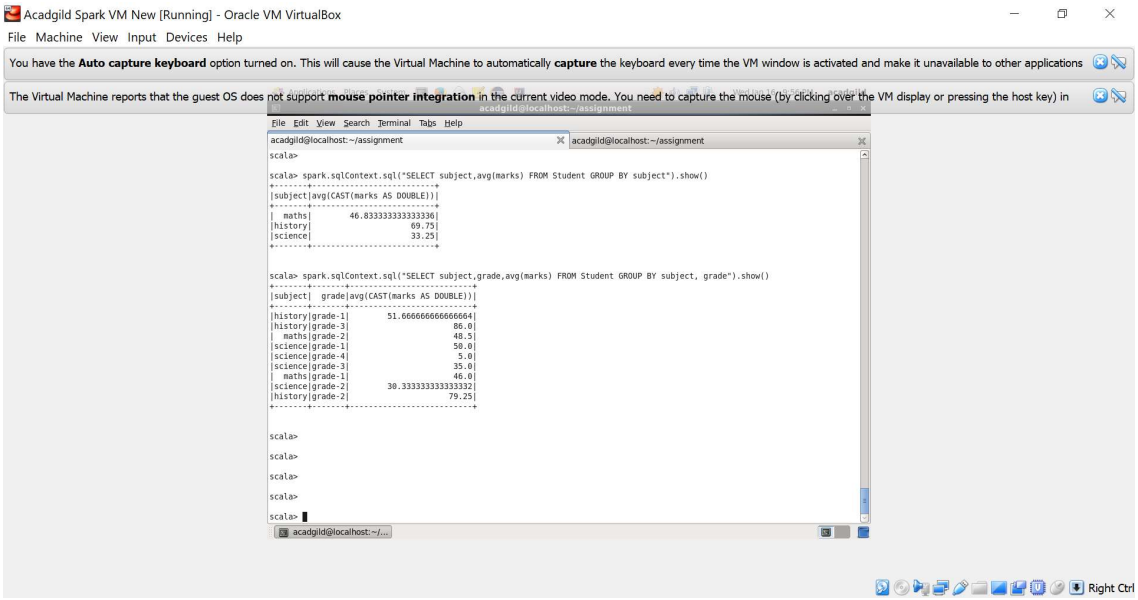


3. What is the average score of students in each subject across all grades?

- Create a query which will select average marks from Student group by subject

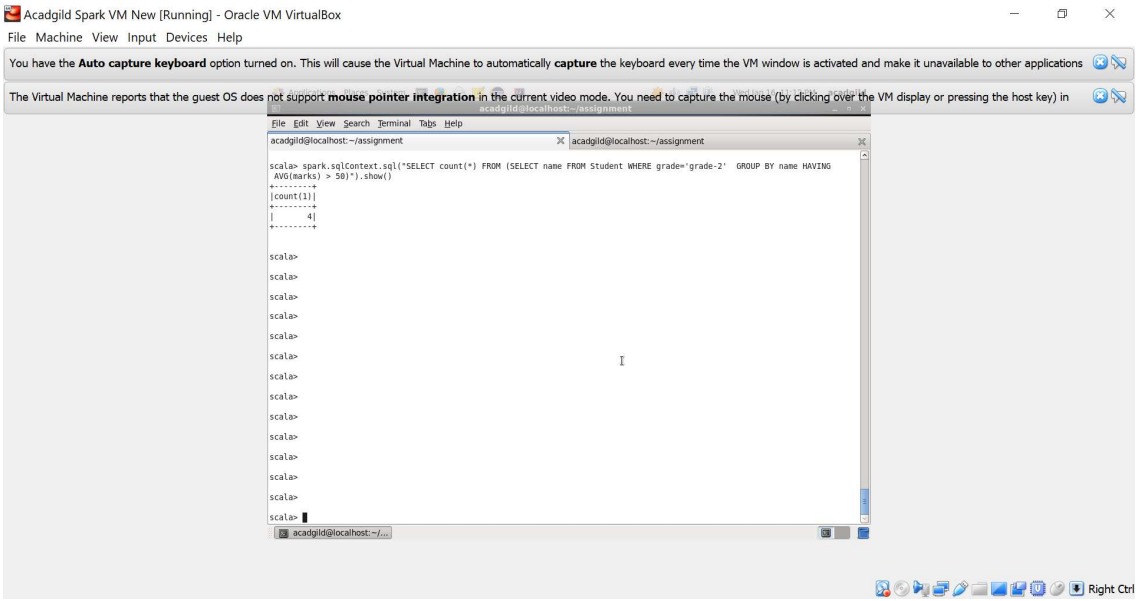
4. What is the average score of students in each subject per grade?
- Create a query which will select average marks from Student group by subject, grade

Screenshot for 3 and 4 are as below:



5. For all students in grade-2, how many have average score greater than 50?

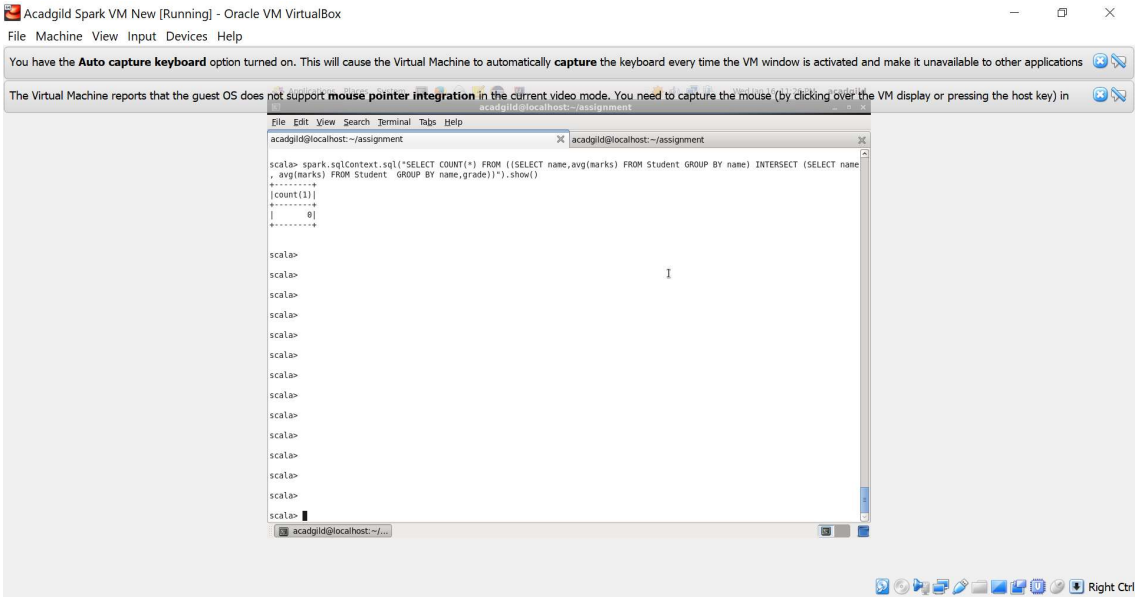
- Create a inner query which will select name from Student for grade-2 having marks > 50
- Create a outer query which will select count from the inner query
Screenshot for #5 is as below:



Problem Statement 3:
Are there any students in the college that satisfy the below criteria :
1. Average score per student_name across all grades is same as average score per student_name per grade
Hint - Use Intersection Property.

- There will be two inner query, first will select name,average marks group by name, the second inner query will select name,average marks group by name, grade
- Create a outer query which will intersect both the inner queries and select count

Screenshot is as below:



Task 2: (Process Air Travelers dataset)

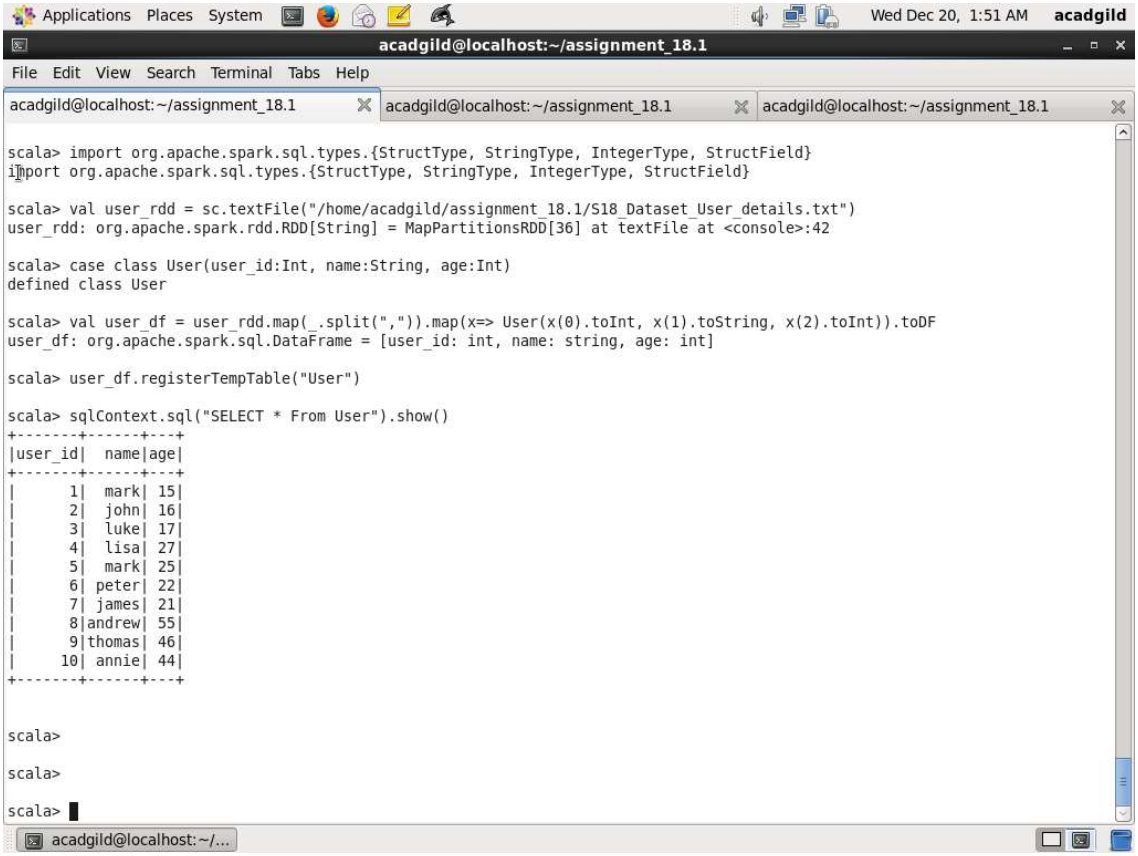
Step1: Create a temporary table User

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")

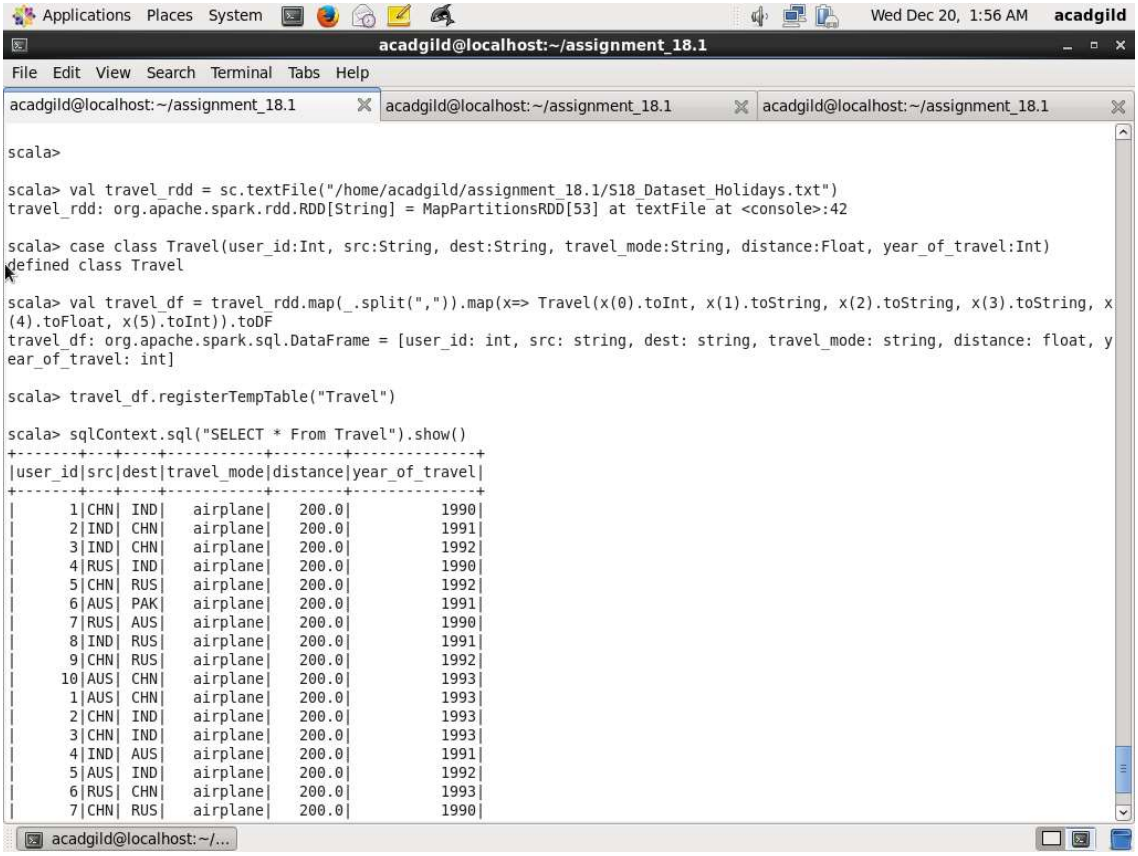
case class Travel(user_id:Int, src:String, dest:String, travel_mode:String, distance:Float, year_of_travel:Int)

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")

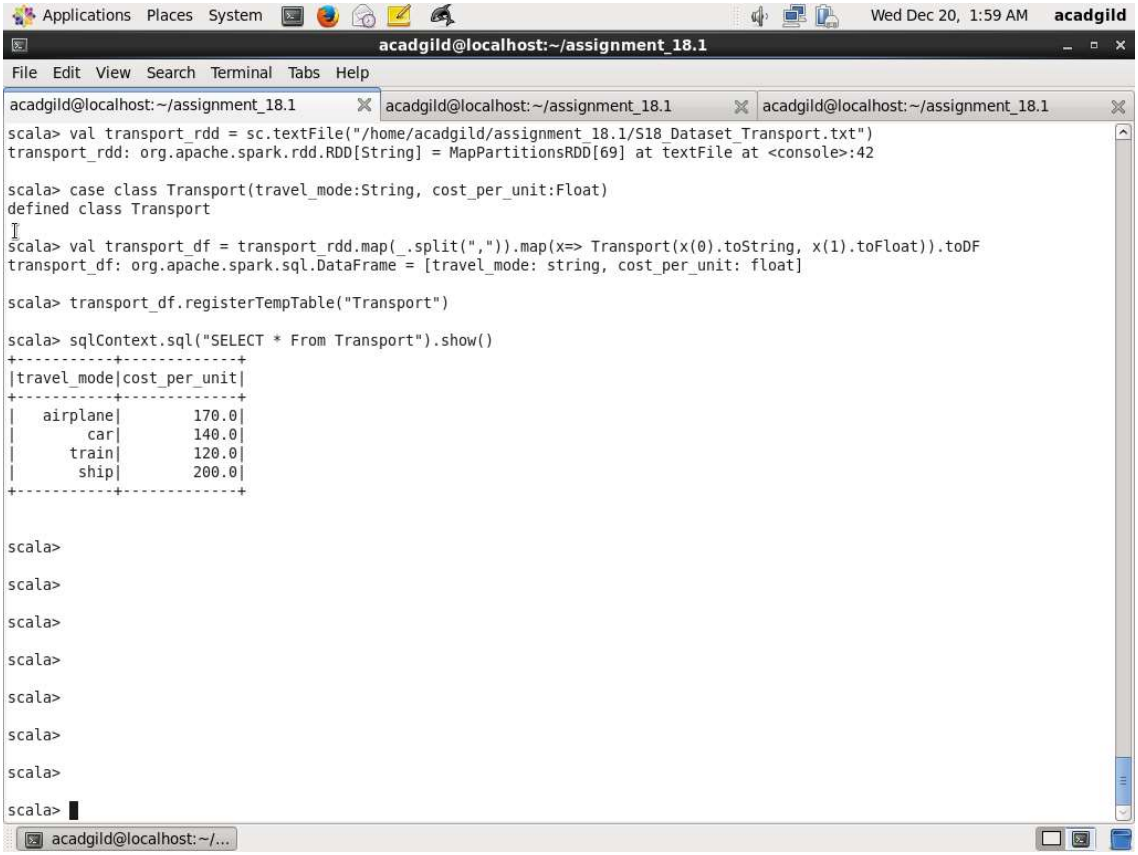
case class Transport(travel_mode:String, cost_per_unit:Float)

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:



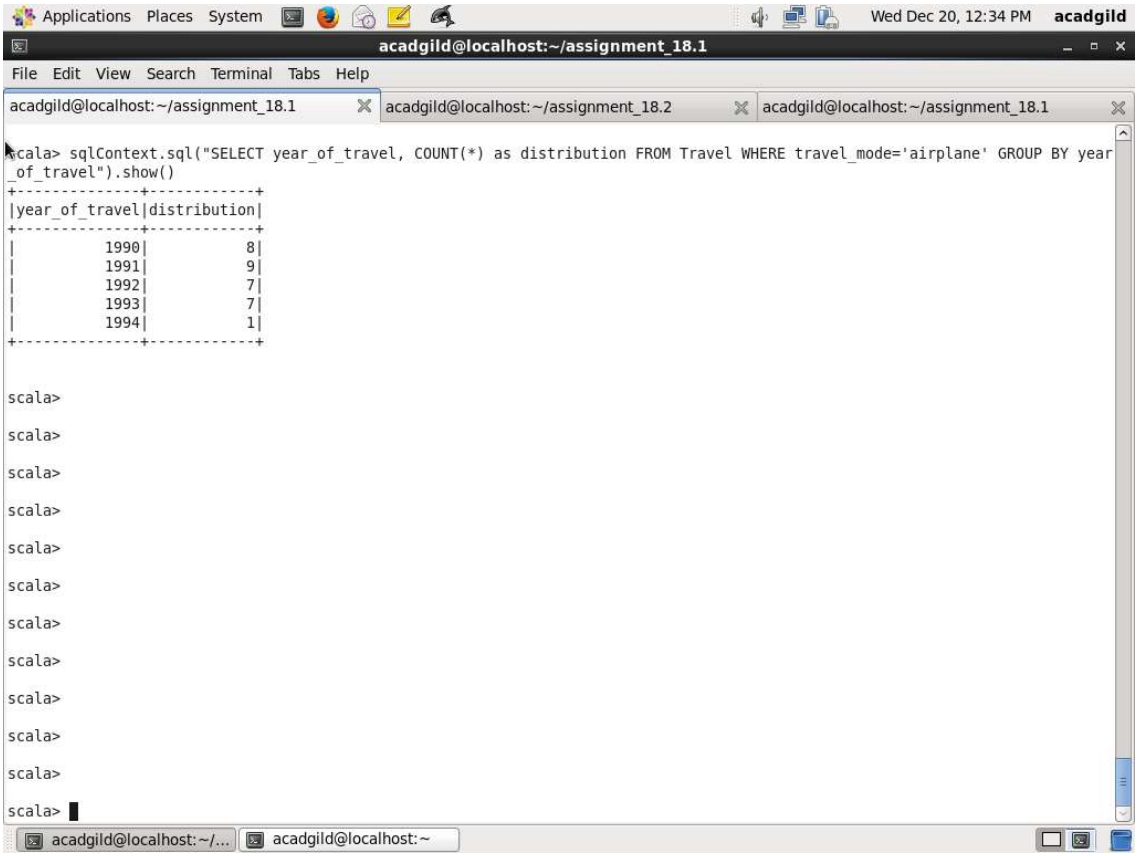
Task1: Distribution of total number of air travelers per year

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

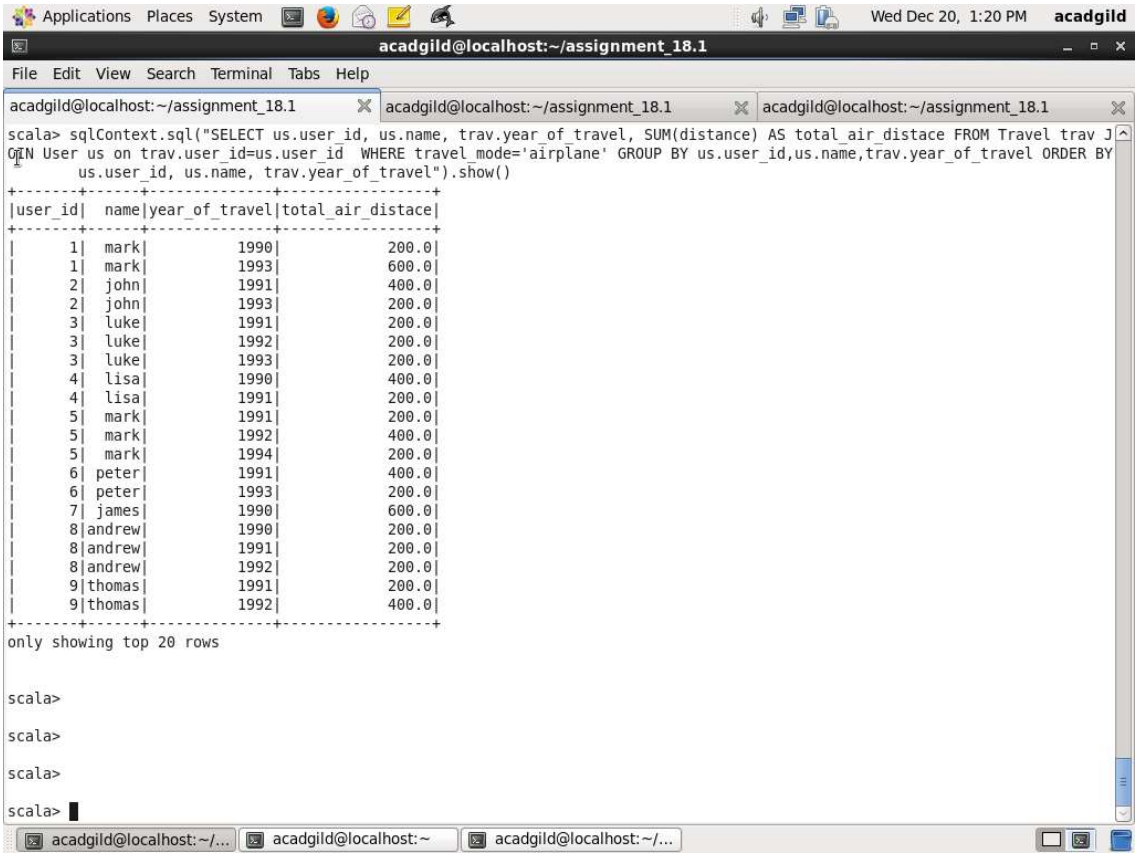
Code is as below:

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:



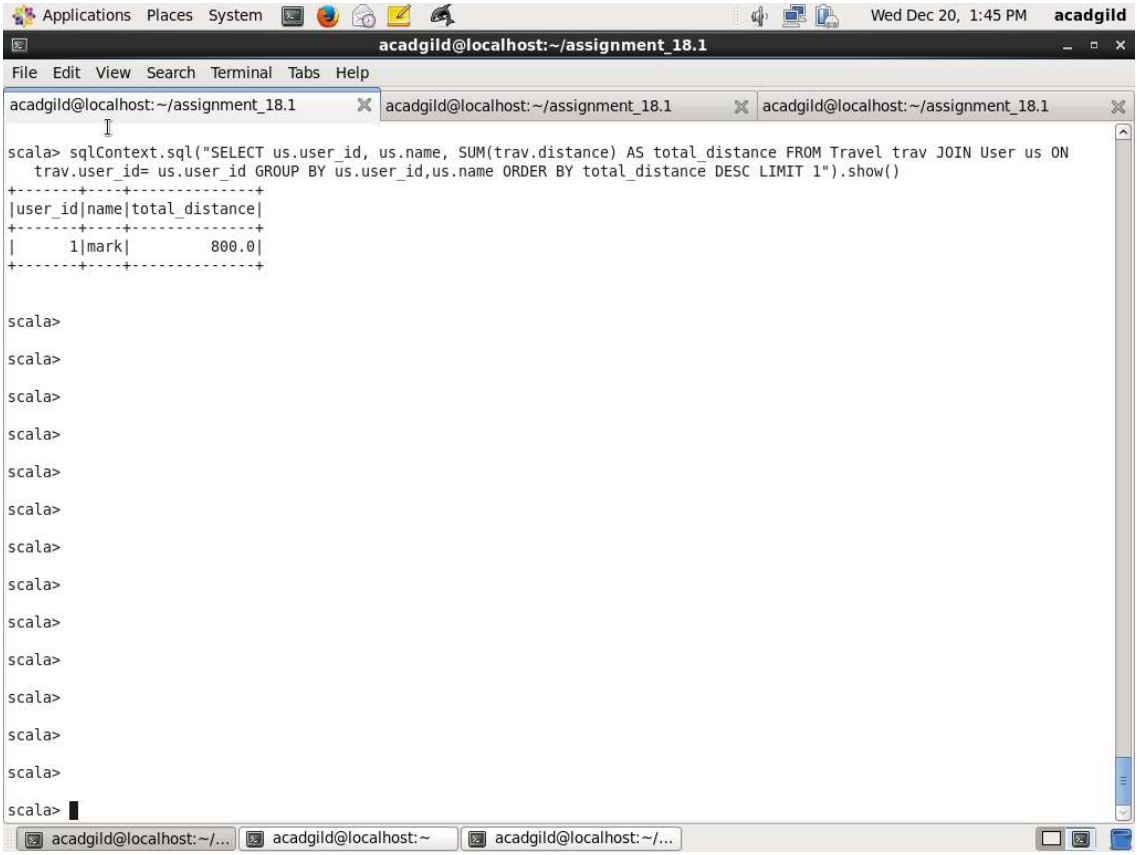
Task3: Which user has travelled largest distance till date

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:



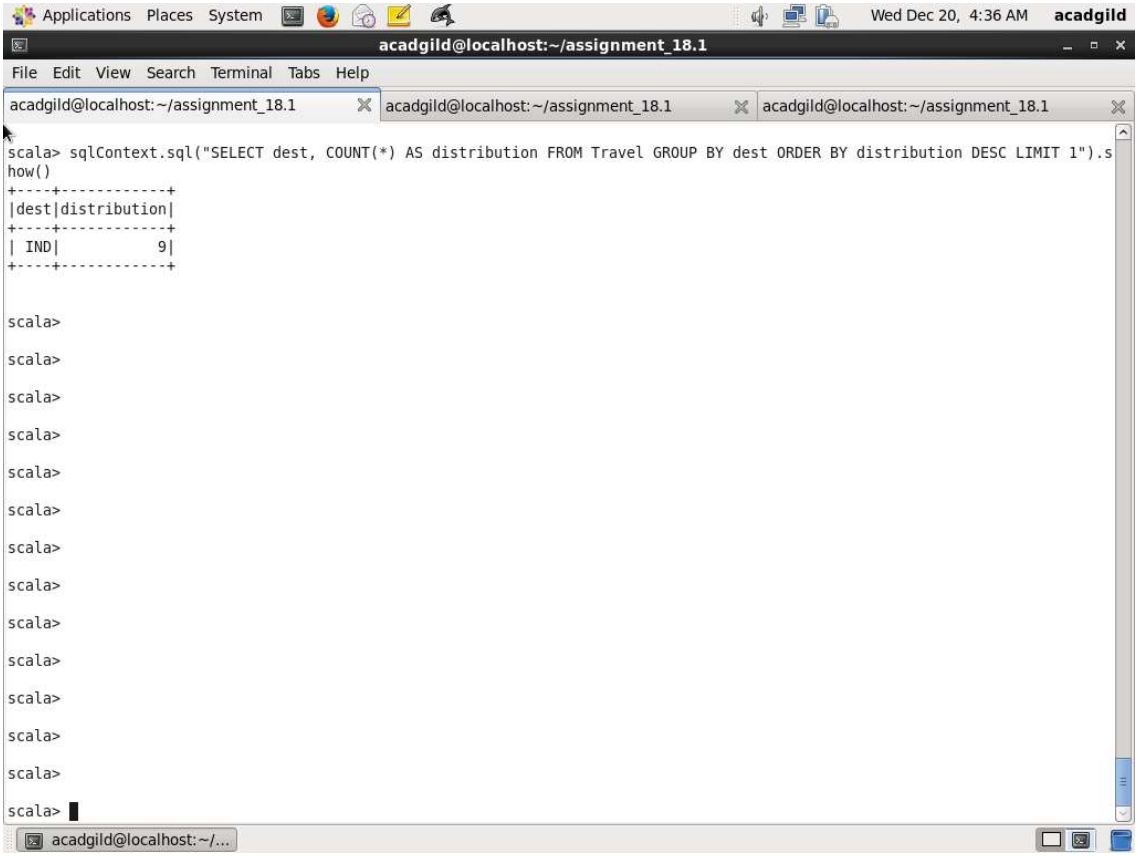
Task4: What is the most preferred destination of all users

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")

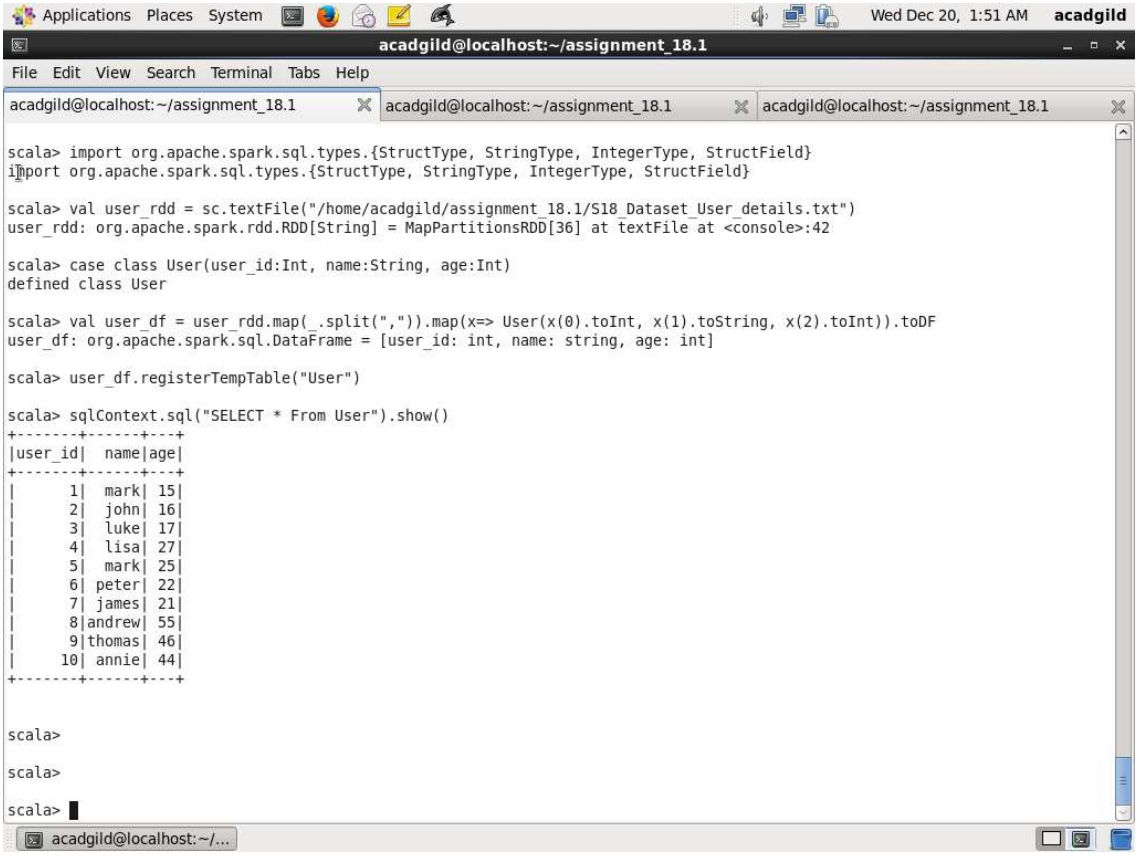
case class User(user_id:Int, name:String, age:Int)

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")

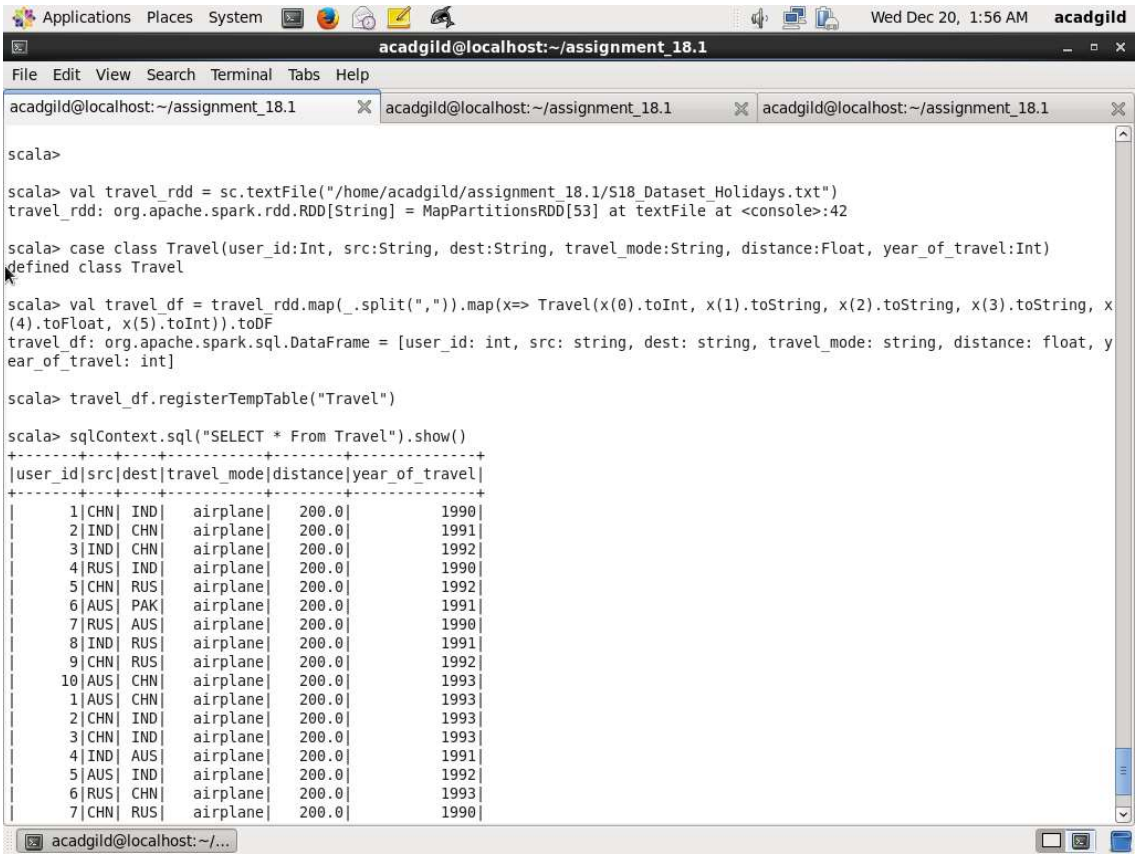
case class Travel(user_id:Int, src:String, dest:String, travel_mode:String, distance:Float, year_of_travel:Int)

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:



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")

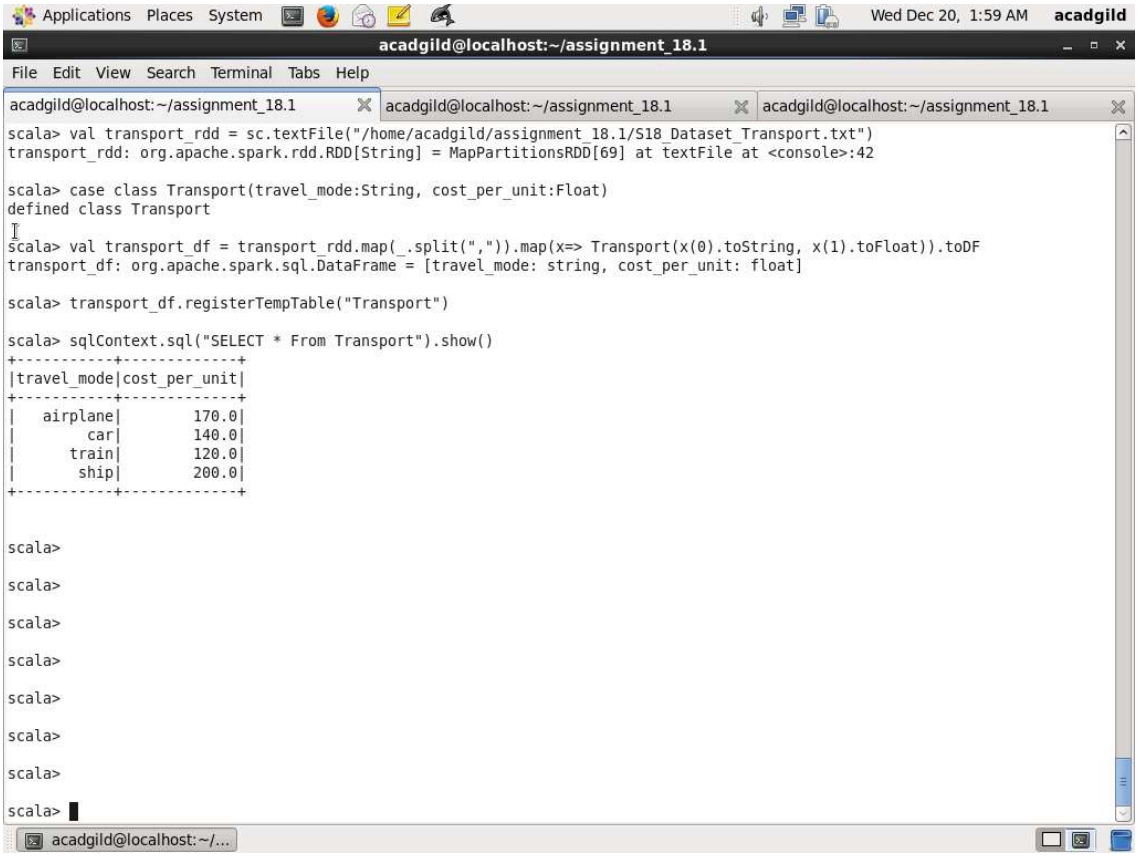
case class Transport(travel_mode:String, cost_per_unit:Float)

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:



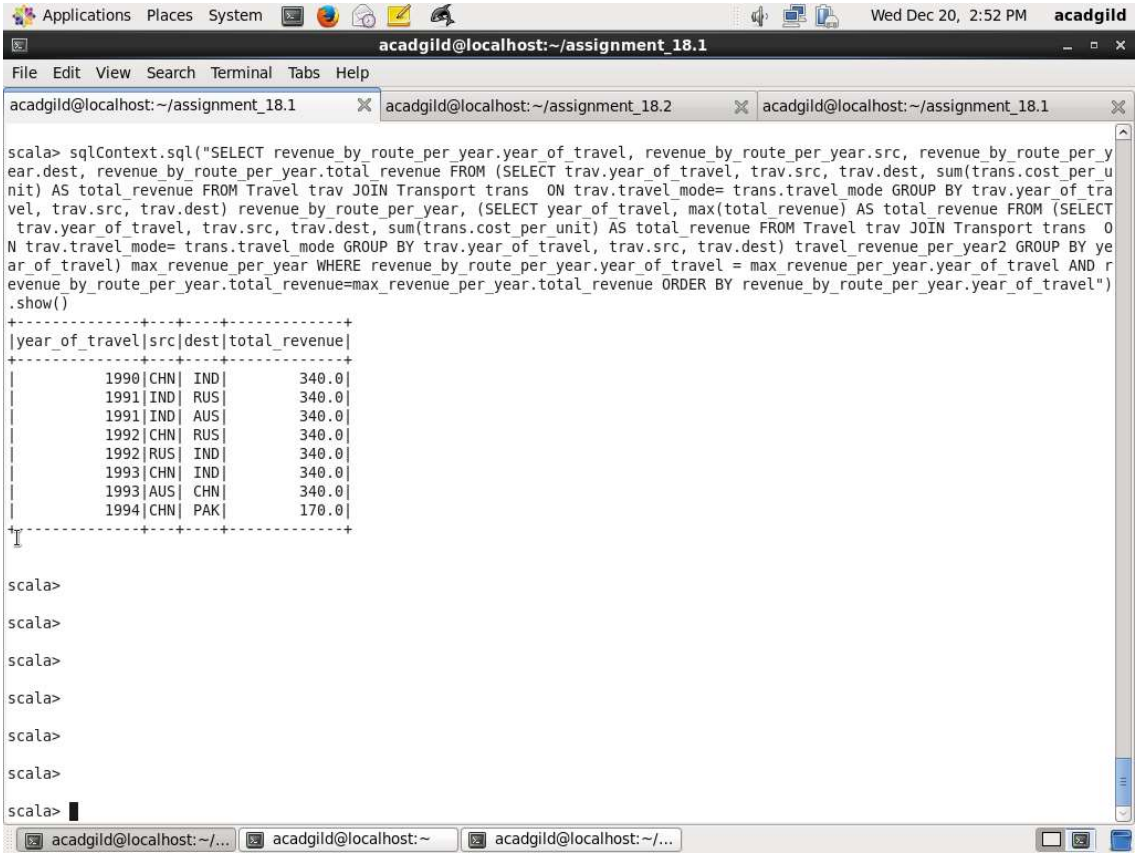
Task1: Which route is generating the most revenue per year

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

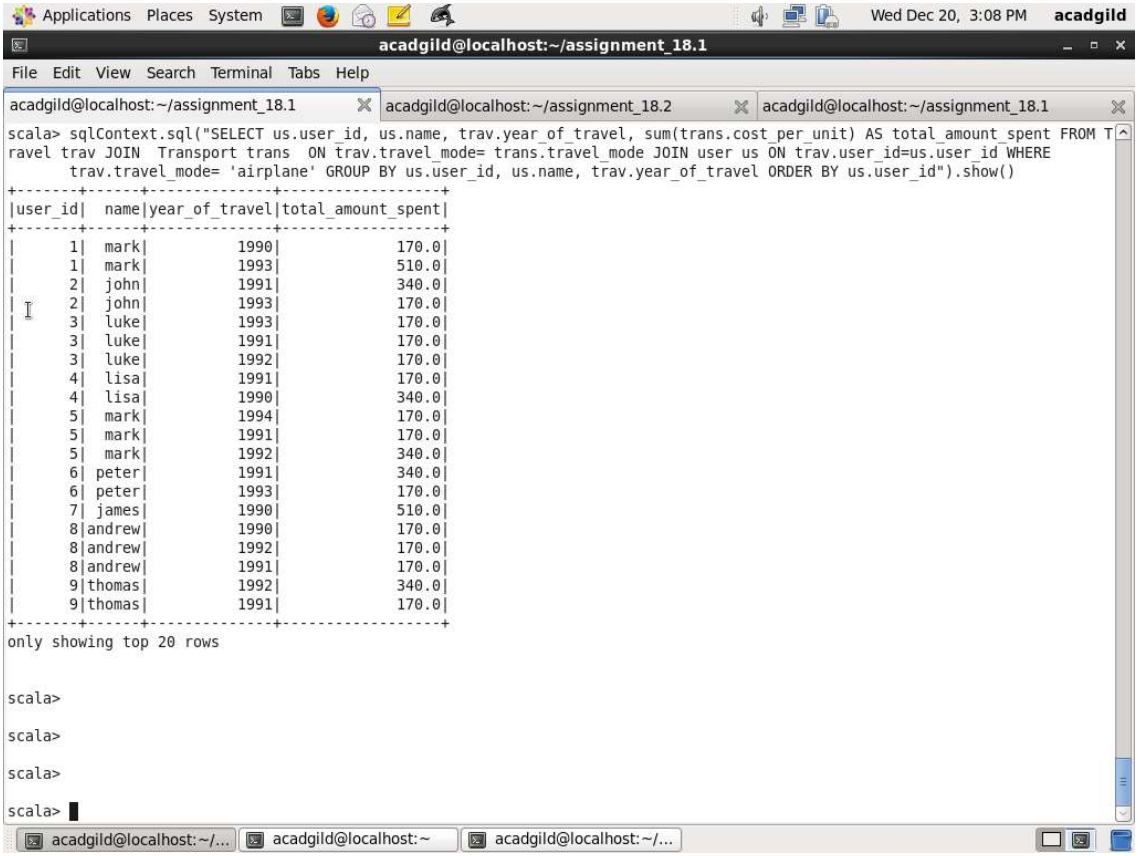
Here I have joined temporary tables Travel, User, Transport using the common fields (travel_mode between Travel and Transport, user_id between Travel and User) and using sum function on cost_per_unit group by user_id, name, with travel_mode as airplane

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: