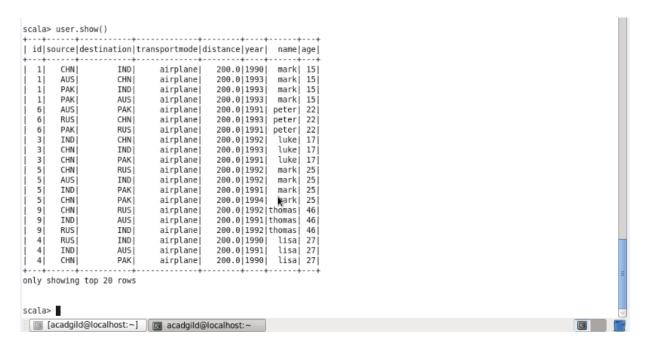
TASK 1: What is the distribution of the total number of air-travelers per year

```
💸 Applications Places System 🍓 ි 🗾 属
                                                                                        🌞 🍁 🌉 📋
                                                                                                          Fri Jul 6, 12:14 AM
                                                                                                                             Acadaild
                                                      acadgild@localhost:~
 Browse and run installed applications
 File Edit View Search Terminal Help
[acadgild@localhost ~1$ spark-shell
Setting default log level to "WARN"
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
18/07/05 23:57:02 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
18/07/05 23:57:04 WARN util.Utils: Your hostname, localhost.localdomain resolves to a loopback address: 127.0.0.1; using 10.0
 .3.15 instead (on interface eth16)
18/07/05 23:57:04 WARN util.Utils: Set SPARK_LOCAL_IP if you need to bind to another address
18/07/05 23:57:11 WARN util.Utils: Service 'SparkUT' could not bind on port 4040. Attempting port 4041.
Spark context Web UI available at http://l0.0.3.15:4041
Spark context available as 'sc' (master = local[*], app id = local-1530815233447).
Spark session available as 'spark'.
Welcome to
                               version 2.2.1
Using Scala version 2.11.8 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0 151)
Type in expressions to have them evaluated.
Type :help for more information.
scala> val lines = sc.textFile("/user/spark/Holidays.txt").map( .split(","))
lines: org.apache.spark.rdd.RDD[Array[String]] = MapPartitionsRDD[2] at map at <console>:24
scala> case class Holidays(id:Int,source:String,destination:String,transportmode:String,distance:Float,year:Int)
defined class Holidavs
scala> val holidayDF = lines.map(att => Holidays(att(0).toInt,att(1),att(2),att(3),att(4).toFloat,att(5).toInt)).toDF;
18/07/06 00:14:13 WARN metastore.ObjectStore: Failed to get database global_temp, returning NoSuchObjectException
holidayDF: org.apache.spark.sql.DataFrame = [id: int, source: string ... 4 more fields]
scala> holidayDF.registerTempTable("TASK1")
warning: there was one deprecation warning; re-run with -deprecation for details
scala> val sql =spark.sql("select * from TASK1")
sql: org.apache.spark.sql.DataFrame = [id: int, source: string ... 4 more fields]
scala> sql.show()
                  -----
| id|source|destination|transportmode|distance|year|
  --+-----+-----
                                           200.0|1990|
        CHNI
                     INDI
                               airplanel
  11
                     CHN
        IND
                               airplane|
                                            200.0|1991
   21
        IND
                     CHN
                               airplanel
                                            200.0 | 1992
   31
        RUS
                     IND
                               airplane|
                                            200.0 | 1990
   4
                               airplanel
        CHN
                     RUS
                                            200.0 | 1992
   51
                               airplane|
   61
        AUS
                     PAK
                                            200.0 | 1991
   7 |
        RUS
                     AUS
                               airplane
                                            200.0 | 1990
   8
        IND
                     RUS
                               airplane|
                                            200.0 | 1991
                     RUS
                               airplane
   9|
        CHN
                                            200.0|1992
  10 İ
        AUS
                     CHN
                               airplane
                                            200.0 | 1993
                               airplane
        AUS
                     CHN
                                            200.0|1993
   11
        CHN
                     IND
                               airplane
   21
                                            200.0 | 1993
        CHN
                               airplane
                                            200.0 | 1993
                     IND
   31
   4
        IND
                     AUS
                               airplanel
                                            200.0 | 1991
                               airplane
                                            200.0 | 1992
   51
        AUS
                     IND
        RUS
   6|
                     CHN
                               airplane
                                            200.0 | 1993
                               airplane
        CHN
                     RUS
                                            200.0 | 1990
        AUS
                     CHN
                               airplane|
                                            200.0 1990
   8
   9 |
        IND
                     AUS
                               airplane
                                            200.0 1991
 10
        RUS
                     CHN
                               airplane|
                                           200.0 | 1992 |
only showing top 20 rows
```

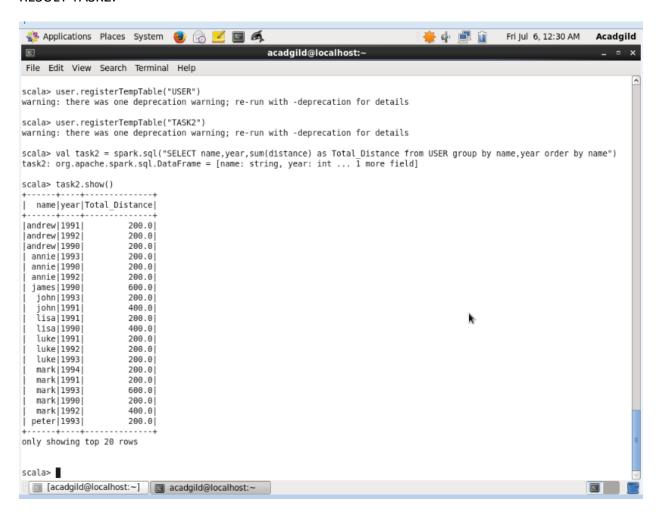
RESULT OF TASK 1:

TASK2: What is the total air distance covered by each user per year

```
scala> val userDetail = sc.textFile("/user/spark/User_Details.txt").map(_.split(","))
userDetail: org.apache.spark.rdd.RDD[Array[String]] = MapPartitionsRDD[28] at map at <console>:24
scala> case class UserDetail(id:Int,name:String,age:Int)
defined class UserDetail
scala> val userDetailDF = userDetail.map(attr => UserDetail(attr(0).toInt,attr(1),attr(2).toInt)).toDF;
userDetailDF: org.apache.spark.sql.DataFrame = [id: int, name: string ... 1 more field]
scala> userDetailDF.show()
| 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> val user = holidayDF.join(wserDetailDF, "id")
user: org.apache.spark.sql.DataFrame = [id: int, source: string ... 6 more fields]
[acadgild@localhost:~]
[acadgild@localhost:~]
```



RESULT TASK2:

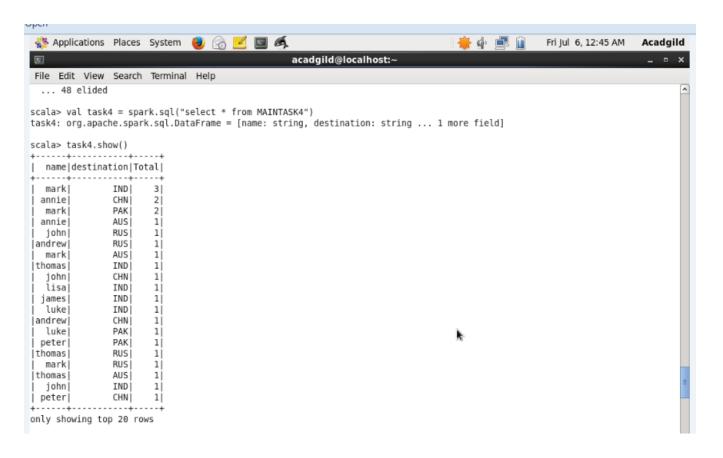


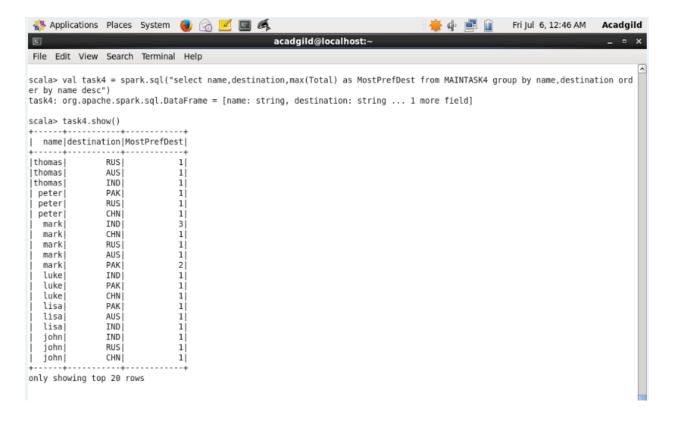
TASK 3: Which user has travelled the largest distance till date

```
scala> val task3 = spark.sql("SELECT name,sum(distance) as Total_Distance from USER group by name order by sum(distance) desc
task3: org.apache.spark.sql.DataFrame = [name: string, Total_Distance: double]
scala> task3.show(1)
|name|Total Distance|
|mark|
          1600.0
only showing top 1 row
scala> task3.show()
 name|Total_Distance|
 markl
              1600.0
              600.0
 peter
              600.0
 annie
               600.0
  lisal
|andrew|
               600.0
  john
               600.0
  luke
               600.0
thomas
               600.0
| james|
scala>
```

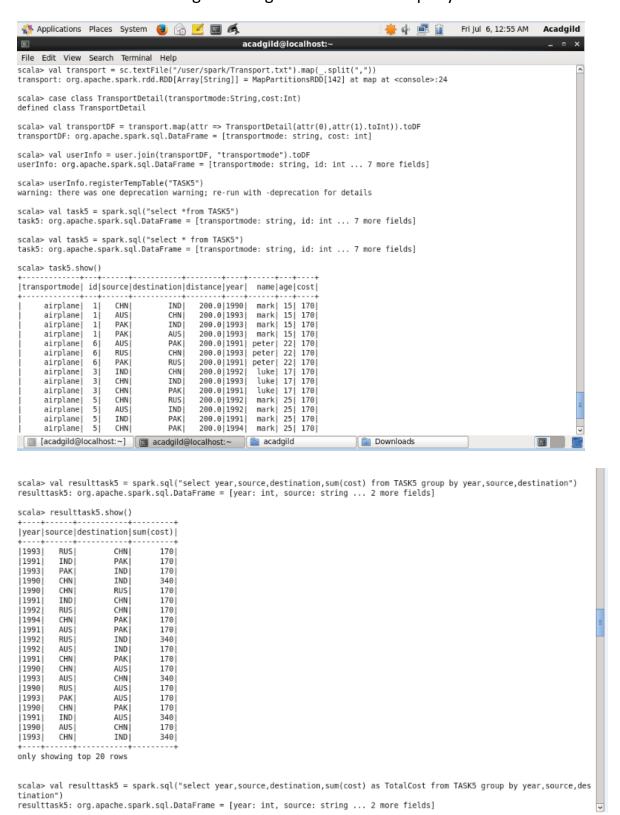
TASK 4: What is the most preferred destination for all users.

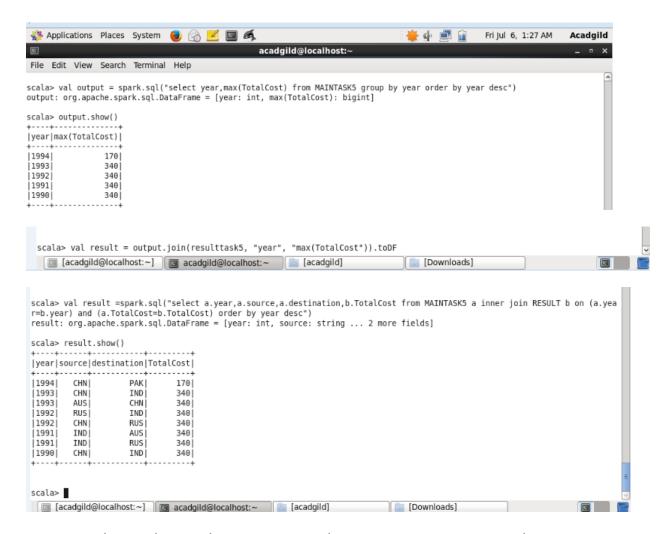
```
scala> val task4 = spark.sql("SELECT name, destination, count(destination) as Total from USER group by name, destination order
by count(destination) desc")
task4: org.apache.spark.sql.DataFrame = [name: string, destination: string ... 1 more field]
scala> task4.registerTempTable("MAINTASK4")
warning: there was one deprecation warning; re-run with -deprecation for details
```





TASK 5: Which route is generating the most revenue per year





TASK 6: What is the total amount spent by every user on air-travel per year

```
scala> val task6 = spark.sql("select id,year,sum(cost) from TASK5 group by id,year order by id")
task6: org.apache.spark.sql.DataFrame = [id: int, year: int ... 1 more field]
scala> task6.show()
| id|year|sum(cost)|
   1|1990|
                     170|
   1 | 1993 |
                    510
   2 | 1991
                    340
   2 1993
                     170
   3 1991
                    170
   3 | 1992
                    170
   3 | 1993
                    170
    4 1990
                     340
    4 | 1991
                     170
   5 | 1992 |
                    340 İ
                    170
   5 | 1991
    5 1994
                     170
   6 | 1991
                    340
   6 1993
                    170
    7 1990
                    5101
   8 1991
                    170
    8 1990
                     170
   8 | 1992 |
                    170
   9|1991
                    170
   9 1992
                    340
only showing top 20 rows
```

TASK 7: Considering age groups of < 20 , 20-35, 35 > ,Which age group is travelling the most every year.

scala> val task7 =spark.sql("select id,age,year,count(id) as TotalCount from TASK5 where age >35 or age <20 or age between 20
and 35 group by id,age,year order by age,year desc")
task7: org.apache.spark.sql.DataFrame = [id: int, age: int ... 2 more fields]</pre>

scala> task7.show()

++	++	+
id age year TotalCount		
++	+	+
1	15 1993	3
1	15 1990	1
2	16 1993	1
2	16 1991	2
3	17 1993	1
3	17 1992	1
3	17 1991	1
7	21 1990	3
6	22 1993	1
6	22 1991	2
5	25 1994	1
5	25 1992	2
5	25 1991	1
4	27 1991	1
4	27 1990	2
10	44 1993	1
10	44 1992	1
10	44 1990	1
9	46 1992	2
j 9j	46 1991	1
++	+	

only showing top 20 rows