Task 1

1) What is the distribution of the total number of air-travelers per year
2) What is the total air distance covered by each user per year
3) Which user has travelled the largest distance till date
4) What is the most preferred destination for all users.
5)Which route is generating the most revenue per year

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

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

Dataset Used

File Edit Format View Help

```
1,CHN,IND,airplane,200,1990
2,IND,CHN,airplane,200,1991
3,IND,CHN,airplane,200,1992
4,RUS,IND,airplane,200,1990
5,CHN,RUS,airplane,200,1992
6,AUS,PAK,airplane,200,1991
7,RUS,AUS,airplane,200,1990
8,IND,RUS,airplane,200,1991
9,CHN,RUS,airplane,200,1992
10,AUS,CHN,airplane,200,1993
1,AUS,CHN,airplane,200,1993
2,CHN,IND,airplane,200,1993
3,CHN,IND,airplane,200,1993
4,IND,AUS,airplane,200,1991
5,AUS,IND,airplane,200,1992
6,RUS,CHN,airplane,200,1993
7,CHN,RUS,airplane,200,1990
8,AUS,CHN,airplane,200,1990
9,IND,AUS,airplane,200,1991
10,RUS,CHN,airplane,200,1992
1,PAK,IND,airplane,200,1993
2,IND,RUS,airplane,200,1991
3,CHN,PAK,airplane,200,1991
4,CHN,PAK,airplane,200,1990
5,IND,PAK,airplane,200,1991
6,PAK,RUS,airplane,200,1991
7,CHN,IND,airplane,200,1990
8,RUS,IND,airplane,200,1992
9,RUS,IND,airplane,200,1992
10,CHN,AUS,airplane,200,1990
1,PAK,AUS,airplane,200,1993
5,CHN,PAK,airplane,200,1994
```

Code :

```
package Test

import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.functions._

object Task567 {

  //Create case classes globally to be used inside the main method for different
dataset
  case class TRAVEL(id: String, origin: String, destination: String, transport:
String, distance: Int, year: Int)

  case class TRANSPORT(transport: String, amount: Int)

  case class USER(id: String, name: String, age: Int)

  def main(args: Array[String]): Unit =
```

```
  {
    println("Assignment Number 20 !!!")

    // Use new SparkSession interface in Spark
    val spark = SparkSession
      .builder()
      .master("local[*]")
      .appName("Assignment 20 task 1 to 5 ")
      .config("spark.some.config.option", "some-value")
      .getOrCreate()

    // For implicit conversions like converting RDDs and sequences  to DataFrames
    import spark.implicits._

    // Create an RDD of TRAVEL objects from a text file S20_Dataset_Holidays.txt.
    val travel =
spark.sparkContext.textFile("C:/Users/admin/Desktop/Assignment_to_be
submitted/s20/Dataset_Holidays.txt")
    val travelDF = travel.map(_.split(",")).map(line => TRAVEL(line(0).toString,
line(1).toString,
      line(2).toString, line(3).toString, line(4).toInt, line(5).toInt))

    //convert the RDD travelDF to a Dataframe
    val transportByAirplane = travelDF.filter(x => x.transport == "airplane").toDF


    // Create an RDD of TRAVEL objects from a text file S20_Dataset_Transport.txt
and convert the RDD transportDF to a Dataframe
    val transportMode =
spark.sparkContext.textFile("C:/Users/admin/Desktop/Assignment_to_be
submitted/s20/Dataset_Transport.txt")
    val transportDF = transportMode.map(_.split(",")).map(line =>
TRANSPORT(line(0).toString, line(1).toInt)).toDF

    // Create an RDD of TRAVEL objects from a text file
S20_Dataset_User_details.txt and convert the RDD userDF to a Dataframe
    val user = spark.sparkContext.textFile("C:/Users/admin/Desktop/Assignment_to_be
submitted/s20/Dataset_User_details.txt")
    val userDF = user.map(_.split(",")).map(line => USER(line(0).toString,
line(1).toString, line(2).toInt)).toDF



    //Use transportByAirplane dataframe,group the year and count the value
    val air_travelers_per_year =
transportByAirplane.groupBy("year").count().sort("year").show()
    println(" Total no. of air travelers per year")


    //Use transportByAirplane dataframe,group it by user and year apply summation
for distance column.
    val Total_Distance_Cover_per_year =
transportByAirplane.groupBy("id","year").sum("distance").orderBy("id").show()
    println("Total air distance cover by each user per year")

    //Use transportByAirplane dataframe,group it by id and add all the distance
with respect to it.
    val largest_Distance_By_User =
transportByAirplane.groupBy("id").sum("distance").orderBy("id").show(1)
   println("Largest distance travel by user till date ")


    //Use transportByAirplane dataframe,group it by destination column and count
its value.
    val preferred_destination =
transportByAirplane.groupBy("destination").count().orderBy(desc("count")).show(1)
    println("Most preferred destination for all users")
```

```
    //Join transportByAirplane and transportDF dataframes ,group it by year,origin
and destination column and
    //add all the amount with repect to this column
    val revenue_per_year =
transportByAirplane.join(transportDF,transportByAirplane("transport")===
transportDF("transport")).

groupBy("year","origin","destination").sum("amount").sort(desc("sum(amount)"))show(
10)
    println("Route generating most revenue per year")

    //Join transportByAirplane and transportDF dataframes ,group it by id,year and
add all the amount with repect to this columns
   val amount_spent_per_year =
transportByAirplane.join(transportDF,transportByAirplane("transport")===
transportDF("transport")).
      groupBy("id","year").sum("amount").orderBy("id","year").show()
    print("total amount spent by every user on air travel per year")

  }

}
```

Output:

```
18/09/16 14:16:07 INFO CodeGenerator: Code generated in 17.67324 ms
+----+-----+
|year|count|
+----+-----+
|1990|    8|
|1991|    9|
|1992|    7|
|1993|    7|
|1994|    1|
+----+-----+

 Total no. of air travelers per year
18/09/16 14:16:07 INFO SparkContext: Invoking stop() from shutdown hook
```

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

```
18/09/16 14:17:22 INFO CodeGenerator: Code generated in 11.95334 ms
+---+----+-------------+
| id|year|sum(distance)|
+---+----+-------------+
|  1|1993|          600|
| 10|1990|          200|
| 10|1993|          200|
| 10|1992|          200|
|  2|1993|          200|
|  2|1991|          400|
|  3|1992|          200|
|  3|1993|          200|
|  3|1991|          200|
|  4|1990|          400|
|  4|1991|          200|
|  5|1991|          200|
|  5|1992|          400|
|  5|1994|          200|
|  6|1993|          200|
|  6|1991|          400|
|  7|1990|          600|
|  8|1992|          200|
|  8|1991|          200|
|  8|1990|          200|
+---+----+-------------+
only showing top 20 rows

Total air distance cover by each user per year
18/09/16 14:17:22 INFO SparkContext: Invoking stop() from shutdown hook
```

3) Which user has travelled the largest distance till date

Output:

```
18/09/16 14:19:21 INFO DAGScheduler: Job 2 finished: show at Task567.scala:59, took 1.007218 s
18/09/16 14:19:21 INFO CodeGenerator: Code generated in 8.36523 ms
+---+-------------+
| id|sum(distance)|
+---+-------------+
|  1|          600|
+---+-------------+
only showing top 1 row

Largest distance travel by user till date
18/09/16 14:19:21 INFO SparkContext: Invoking stop() from shutdown hook
```

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

Output:

```
18/09/16 14:20:59 INFO DAGScheduler: Job 3 finished: show at Task567.scala:64, took 0.906662 s
18/09/16 14:20:59 INFO CodeGenerator: Code generated in 8.669551 ms
+-----------+-----+
|destination|count|
+-----------+-----+
|        IND|    9|
+-----------+-----+
only showing top 1 row

Most preferred destination for all users
18/09/16 14:20:59 INFO SparkContext: Invoking stop() from shutdown hook
18/09/16 14:20:59 INFO SparkUI: Stopped Spark web UI at http://192.168.100.3:4040
```

5)Which route is generating the most revenue per year

```
+----+------+-----------+-----------+
.8/09/19 23:36:49 INFO SparkContext: Invoking stop() from shutdown hook
 year|origin|destination|sum(amount)|
+----+------+-----------+-----------+
+----+------+-----------+-----------+

loute generating most revenue per year
.8/09/19 23:36:49 INFO SparkUI: Stopped Spark web UI at http://192.168.100.4:4040
.8/09/19 23:36:49 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint s
```

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

```
18/09/19 23:38:43 INFO DAGScheduler: Job 5 finished: show at Task567.scala:77, took 4.429220 s
+---+----+-----------+
| id|year|sum(amount)|
+---+----+-----------+
+---+----+-----------+

total amount spent by every user on air travel per year18/09/19 23:38:43 INFO SparkContext: Invoking stop() from shutdown hc
18/09/19 23:38:43 INFO SparkUI: Stopped Spark web UI at http://192.168.100.4:4040
18/09/19 23:38:43 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
```

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

Code :

```
package Test

import org.apache.spark.sql.SparkSession

object  Task7
{

  def main(args: Array[String]): Unit =

  {   println("Assignment Number 20 !!!")

    // Use new SparkSession interface in Spark
    val spark = SparkSession
      .builder()
      .master("local[*]")
      .appName("Assignment 20 task no 7 ")
      .config("spark.some.config.option", "some-value")
      .getOrCreate()

    // For implicit conversions like converting RDDs and sequences  to DataFrames
    import spark.implicits._

    // Create an RDD of  from a text file S20_Dataset_Holidays.txt.
    val TRAVEL =
spark.sparkContext.textFile("C:/Users/admin/Desktop/Assignment_to_be
submitted/s20/Dataset_Holidays.txt")
    val travel = TRAVEL.map(x =>
```

```
(x.split(",")(0),x.split(",")(1),x.split(",")(2),x.split(",")(3),
      x.split(",")(4).toInt,x.split(",")(5).toInt))

    // Create an RDD of  from a text file S20_Dataset_Transport.txt.
    val TRANSPORT =
spark.sparkContext.textFile("C:/Users/admin/Desktop/Assignment_to_be
submitted/s20/Dataset_Transport.txt")
    val transport = TRANSPORT.map(x => (x.split(",")(0),x.split(",")(1).toInt))

    // Create an RDD of  from a text file S20_Dataset_User_details.txt.
    val USER =spark.sparkContext.textFile("C:/Users/admin/Desktop/Assignment_to_be
submitted/s20/Dataset_User_details.txt")
    val user = USER.map(x =>
(x.split(",")(0),x.split(",")(1),x.split(",")(2).toInt))


    // create an RDD AgeGroup from user to get different age-groups from age
column.
    val AgeGroup = user.map(x => x._1 -> {if(x._3<20) "20" else if(x._3>35) "35"
else "20-35" })

    // create an RDD travelMap from travel to map id as key and (distance and year)
as value
    val travelMap = travel.map(x => (x._1 -> (x._6,x._5)))

    // create an RDD ageTravelJoin to join AgeGroup and travelMap
    val ageTravelJoin = AgeGroup.join(travelMap)

    // create an RDD ageTravelMap to map (year and age-groups) as key and distance
as a value
    val ageTravelMap = ageTravelJoin.map(x => (x._2._1 ,x._2._2._1) -> x._2._2._2)

    // create an RDD to aggregate the keys year and age-groups
    val  ageTravelReduce = ageTravelMap.reduceByKey((x,y)=> x+y).sortByKey()


    //convert the RDD yearGroupSort to a Dataframe
    val yearGroupSort = ageTravelReduce.map( x => (x._1._2,x._1._1,x._2)).toDF

    /* Now we have a dataframe yearGroupSort with data in below format
   |  _1|   _2|  _3|
      +----+-----+----+
      |1990|   20| 200|
      |1990|20-35|1000|
      |1991|20-35| 800|    */

    //Now we use spaek-sql to get the output....

    val newName = Seq("year","ageGroup","Distance")
    //Schema of yearGroupSort is (._1,._2,._3),convert it into
(year,ageGroup,Distance)  in yearGroupSortNew
    val yearGroupSortNew = yearGroupSort.toDF(newName: _*)

    // to check the new shema of yearGroupSortNew Data Frame
    yearGroupSortNew.printSchema()
    // Register the DataFrame as a temporary view AGEGROUP
    yearGroupSortNew.createOrReplaceTempView("AGEGROUP")

    // RUN SQL statements by using the sql methods provided by Spark to get the
desired result from view AGEGROUP
    val max_distance_per_year = spark.sql("SELECT a.*FROM AGEGROUP a " +
      "INNER JOIN " +
      "(SELECT year, MAX(distance) AS max FROM AGEGROUP  " +
      "GROUP BY year) b " +
      "ON a.year = b.year " +
      "AND a.distance = b.max ").show()

    println("Above results shows the age group travelling the most every year")
```

```
    }
}
```

Output:

```
18/09/15 16:11:20 INFO CodeGenerator: Code generated in 12.332714 ms
+----+--------+--------+
|year|ageGroup|Distance|
+----+--------+--------+
|1990|   20-35|    1000|
|1993|      20|     400|
|1992|      35|     800|
|1994|   20-35|     200|
|1991|   20-35|     800|
+----+--------+--------+

Above results shows the age group travelling the most every year
18/09/15 16:11:20 INFO SparkContext: Invoking stop() from shutdown hook
18/09/15 16:11:20 INFO SparkUI: Stopped Spark web UI at http://192.168.100.3:4040
```