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

object Class20Task {
  case class holidayClass( userId: Int, source: String, destination: 
String, transport_mode: String, distance:Int, year:Int )
  case class transportClass(transport_mode: String, cost_per_unit: Int )
  case class user_details(id: Int, name: String, age: Int)

  def main(args: Array[String]): Unit = {
    val sparkSession = 
SparkSession.builder.master("local").appName("spark session 
example").getOrCreate()
    val sparkContext = sparkSession.sparkContext

    import sparkSession.implicits._

    val holidaysData = sparkContext.textFile("F:\\PDF 
Architect\\S20_Dataset_Holidays.txt")
    val holidaysDF = holidaysData.map(_.split(",")).map(x => 
holidayClass(userId = x(0).toInt, source = x(1),
                  destination = x(2),transport_mode = x(3),distance = 
x(4).toInt, year =  x(5).toInt )).toDF()
    holidaysDF.createOrReplaceTempView("Holiday_Data")
    holidaysDF.show()

    val userDetails = sparkContext.textFile("F:\\PDF 
Architect\\S20_Dataset_User_details.txt")
    val userDetailsDF = userDetails.map(_.split(",")).map(x => 
user_details(id = x(0).toInt, name = x(1), age = x(2).toInt)).toDF()
    userDetailsDF.createOrReplaceTempView("User_Details")
    userDetailsDF.show()

    val transportData = sparkContext.textFile("F:\\PDF 
Architect\\S20_Dataset_Transport.txt")
    val transportDataDF = transportData.map(_.split(",")).map(x => 
transportClass(transport_mode = x(0), cost_per_unit = x(1).toInt 
)).toDF()
    transportDataDF.createOrReplaceTempView("Transport_Data")
    transportDataDF.show()

    val totAirTravellers = sparkSession.sql("select year, 
count(transport_mode) from Holiday_Data where transport_mode = 'airplane' 
group by year ")
    totAirTravellers.show()

    val joinedDF = holidaysDF.join(userDetailsDF, holidaysDF("userId") 
=== userDetailsDF("id"), "inner")
    joinedDF.createOrReplaceTempView("Joined_View")

    val totDistancePerYear = sparkSession.sql("select id, name, year, 
sum( distance ) from Joined_View group by year,id, name" )
    totDistancePerYear.show()
/*    val maxTravelUser = sparkSession.sql( "select name, year, sum(
distance ) from Joined_View " +
                            "group by id, year, name order by sum(
distance ) desc").take(1).mkString(",")
    println(maxTravelUser)*/

/*    val favDestination = sparkSession.sql( "select destination, count(
destination ) from Holiday_Data group by destination " +
                                          "order by  count( destination )
desc")
    favDestination.show(1)*/

/*    val routesDF = holidaysDF.withColumn("Route", struct(
"destination", "source" )).toDF()
    routesDF.createOrReplaceTempView("Routes")
    routesDF.show()

    val maxRoute = sparkSession.sql("select Route, Sum( cost_per_unit )
from Routes JOIN Transport_Data on " +
                    "Routes.transport_mode =
Transport_Data.transport_mode group by Route order by Sum( cost_per_unit)
desc")

    maxRoute.show(1) */

    val totalTravel =
holidaysDF.as("HD").join(userDetailsDF.as("UD"),$"HD.userId" === $"UD.id"
).
      join(transportDataDF.as("TD"), $"HD.transport_mode" ===
$"TD.transport_mode").groupBy(
      "UD.id", "UD.name", "HD.year").sum("cost_per_unit").sort("UD.id",
"year")
    totalTravel.show()

  /*  val travelByAge = sparkSession.sql("select age, count( userId )
from Joined_View where age >=20 and age <= 35 " +
                      "group by age, userId order by count( userId )
desc")

    travelByAge.show()*/
  }

}
```