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

object Class21Task {
  case class sportsDataClass(firstname: String,lastname: String ,sports:
String ,medal_type: String ,age: Int ,year: Int ,country: String)
  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 sportsData = sparkContext.textFile("F:\\PDF
Architect\\Sports_data.txt").map(_.split(","))
                        .map( x => sportsDataClass( x(0), x(1),
x(2),x(3),x(4).toInt, x(5).toInt, x(6)))
    val sportsDataDF =   sportsData.toDF()
    sportsDataDF.createOrReplaceTempView("Sports_Data")
    sportsDataDF.show()

    val usaSilver = sportsDataDF.filter("medal_type = 'silver' and
country = 'USA'")
    usaSilver.show()
    println("Total number of silver medals won by USA is : " +
usaSilver.count())

    /*    val goldmedal = sportsDataDF.filter("medal_type = 'gold'")
        val goldCount = goldmedal.groupBy("year").count()
        goldCount.show()
        println( "Total number of gold medals is : "
+goldmedal.count())*/
/*    val usaSilver = sportsDataDF.filter("medal_type = 'silver' and
country = 'USA'")
    usaSilver.show()
    println("Total number of silver medals won by USA is : " +
usaSilver.count()) */
/*val Name = udf((First_Name: String, Last_Name: String)=>
"Mr.".concat(First_Name.substring(0,2).concat(" ").concat(Last_Name)))
    sportsDataDF.withColumn("Concat_First_Last",Name(
$"firstname",$"lastname"))
      .select("Concat_First_Last","sports","medal_type",
"age","year","country").show()*/

/*    def Rank( Age:Int, Medal: String):String= (Age, Medal) match{
      case( Age, Medal) if Medal == "gold" && Age >= 32  => "pro"
      case( Age, Medal) if Medal  == "gold" && Age <= 31 => "amateur"
      case( Age, Medal) if Medal == "silver" &&  Age >= 32 => "expert"
      case( Age, Medal) if Medal == "silver" && Age <= 31 => "rookie"
      case _ => " "
    }

    val Ranks = udf( Rank(_:Int,_:String))
    sportsDataDF.withColumn("Ranking",Ranks($"age",$"medal_type"))
      .select("ranking","firstname", "lastname", "sports", "medal_type",
"age", "year", "country").show()*/
  }
}
```