Case study IV

```
import org.apache.spark.{SparkConf, SparkContext}
import org.apache.spark.rdd.RDD
import org.apache.spark.streaming.{Seconds, StreamingContext, Time}
import org.apache.spark.sql.SparkSession

object SqlNetworkWordCount {

  def main(args: Array[String]): Unit = {
    println("hey Spark SQL Streaming")

    val conf = new
SparkConf().setMaster("local[2]").setAppName("SparkSteamingExample")
    val sc = new SparkContext(conf)
    println("hey Spark Streaming ---> 1")

    sc.setLogLevel("WARN")
    //val sparkConf = new SparkConf().setAppName("NetworkWordCount")
    println("hey Spark Streaming ---> 2")
    val ssc = new StreamingContext(sc, Seconds(20))
    val lines = ssc.socketTextStream("localhost", 9999)
    println("hey Spark Streaming ---> 3")
```

```
val words = lines.flatMap(_.split(" "))

 println("hey Spark Streaming ---> 4")

// Convert RDDs of the words DStream to DataFrame and run SQL query
words.foreachRDD { (rdd: RDD[String], time: Time) =>
  val spark = SparkSessionSingleton.getInstance(rdd.sparkContext.getConf)
  import spark.implicits._

   println("hey Spark Streaming ---> 6")

  // Convert RDD[String] to RDD[case class] to DataFrame
  val wordsDataFrame = rdd.map(w => Record(w)).toDF()

   println("hey Spark Streaming ---> 7")
  // Creates a temporary view using the DataFrame
  wordsDataFrame.createOrReplaceTempView("words")

   println("hey Spark Streaming ---> 8")

  // Do word count on table using SQL and print it
  val wordCountsDataFrame =
    spark.sql("select word, count(*) as total from words group by word")
   println("hey Spark Streaming ---> 9"+time)
  println(s"========= $time =========")
  wordCountsDataFrame.show()
```

```
    println("hey Spark Streaming ---> 10")
  }


  ssc.start()
  ssc.awaitTermination()


}

/** Case class for converting RDD to DataFrame */
case class Record(word: String)

/** Lazily instantiated singleton instance of SparkSession */
object SparkSessionSingleton {

  @transient private var instance: SparkSession = _

  def getInstance(sparkConf: SparkConf): SparkSession = {
    if (instance == null) {
      instance = SparkSession
        .builder
        .config(sparkConf)
        .getOrCreate()
    }
    instance
```

```
   }
  }
}
```

Network word count

```
import org.apache.spark.{SparkConf, SparkContext}
import org.apache.spark.rdd.RDD
import org.apache.spark.streaming.{Seconds, StreamingContext, Time}
import org.apache.spark.sql.SparkSession


object NetworkWordCount {

  def main(args: Array[String]): Unit = {
   println("hey Spark SQL Streaming")

   /***val spark = SparkSession.builder().master("local").appName("Network Word Count App").config("spark.some.config.option", "some-value").getOrCreate()
   println("Session Object created")

   spark.sparkContext.setLogLevel("WARN")
   val sc = spark.sparkContext***/

   val conf = new
```

```
SparkConf().setMaster("local[2]").setAppName("SparkSteamingExample")
    val sc = new SparkContext(conf)

    sc.setLogLevel("WARN")

    println("hey Spark Streaming ---> 1")
    //val sparkConf = new SparkConf().setAppName("NetworkWordCount")
    println("hey Spark Streaming ---> 2")
    val ssc = new StreamingContext(sc, Seconds(15))
    println("Spark Streaming Context Created !")

    val lines = ssc.socketTextStream("localhost", 9999)
    val words = lines.flatMap(_.split(" "))
    val wordCounts = words.map(x => (x, 1)).reduceByKey(_ + _)
    wordCounts.print()
    ssc.start()
    ssc.awaitTermination()
```

```
  }



}
```