课程实验四：Spark SQL、Spark Streaming

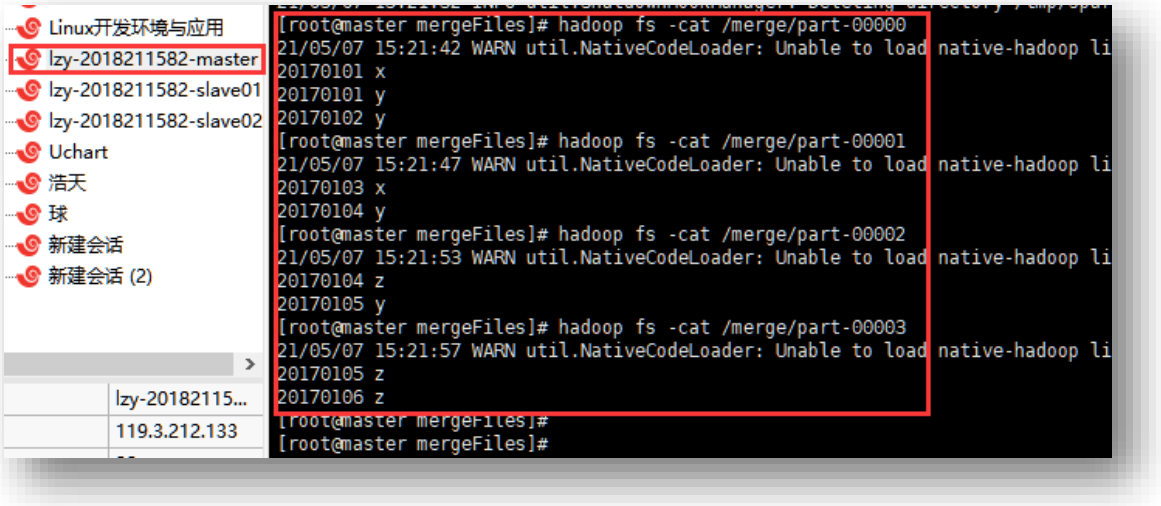
实验时间：2021 年 05 月 07 日

学生姓名：李志毅

学生班号、学号：2018211314 班 2018211582

一、实验结果截图

RDD 编程实验结果：



图一：文件合并去重结果

JDBC 连接数据库实验结果：

```
Linux开发环境与应用
lzy-2018211582-master
lzy-2018211582-slave01
lzy-2018211582-slave02
Uchart
浩天
球
新建会话
新建会话 (2)

scala> val jdbcDF = spark.read.format("jdbc").
     | option("url","jdbc:mysql://localhost:3306/spark").
     | option("driver","com.mysql.jdbc.Driver").
     | option("dbtable","student").
     | option("user","root").
     | option("password","Lzy1234..").
     | load()
jdbcDF: org.apache.spark.sql.DataFrame = [id: int, name: string ... 2 more fields]

scala> jdbcDF.show()
+---+----+------+---+
| id|name|gender|age|
+---+----+------+---+
|  1|  Li|     F| 23|
|  2|Wang|     M| 24|
+---+----+------+---+
```

图二：通过 jdbc 连接 MySQL 数据库

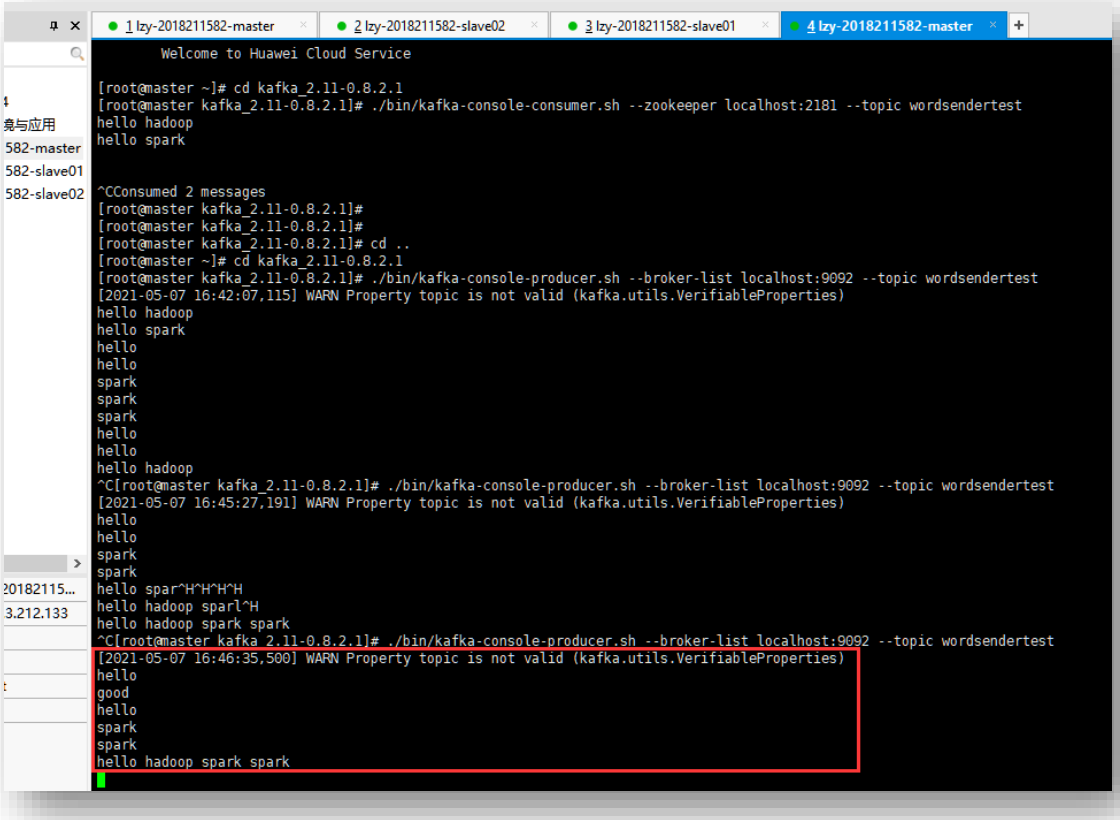
```
[root@master ]# spark-submit --class org.example.InsertStudent spark-test.jar
21/05/07 16:15:47 INFO spark.SparkContext: Running Spark version 2.1.1
21/05/07 16:15:47 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicabl
21/05/07 16:15:47 INFO spark.SecurityManager: Changing view acls to: root
21/05/07 16:15:47 INFO spark.SecurityManager: Changing modify acls to: root
21/05/07 16:15:47 INFO spark.SecurityManager: Changing view acls groups to:
21/05/07 16:15:47 INFO spark.SecurityManager: Changing modify acls groups to:
21/05/07 16:15:47 INFO spark.SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users  with view permissions: Set(root);
ps with view permissions: Set(); users  with modify permissions: Set(root); groups with modify permissions: Set()
21/05/07 16:15:48 INFO util.Utils: Successfully started service 'sparkDriver' on port 33923.
21/05/07 16:15:48 INFO spark.SparkEnv: Registering MapOutputTracker
21/05/07 16:15:48 INFO spark.SparkEnv: Registering BlockManagerMaster
21/05/07 16:15:48 INFO storage.BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopologyMapper for getting topology information
21/05/07 16:15:48 INFO storage.BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
21/05/07 16:15:48 INFO storage.DiskBlockManager: Created local directory at /tmp/blockmgr-d7689a79-1af1-4eb2-9e2f-895ca2a6b70a
21/05/07 16:15:48 INFO memory.MemoryStore: MemoryStore started with capacity 366.3 MB
21/05/07 16:15:48 INFO spark.SparkEnv: Registering OutputCommitCoordinator
21/05/07 16:15:48 INFO util.log: Logging initialized @1633ms
21/05/07 16:15:48 INFO server.Server: jetty-9.2.z-SNAPSHOT
21/05/07 16:15:48 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@6fff253c{/jobs,null,AVAILABLE,@Spark}
```

图三：执行编写好的 jar 包

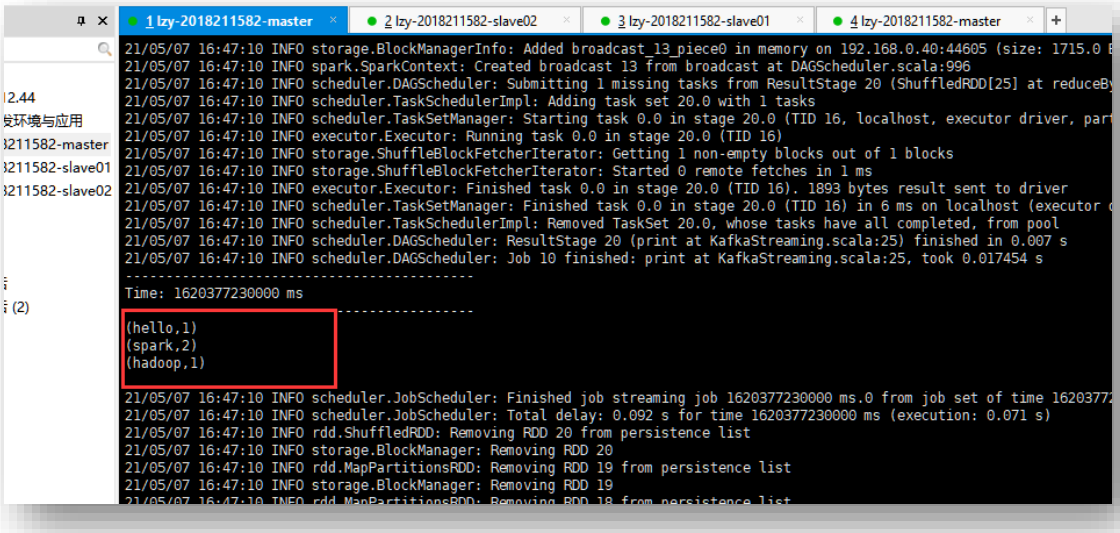
```
8.140.112.44
Linux开发环境与应用
lzy-2018211582-master
lzy-2018211582-slave01
lzy-2018211582-slave02
Uchart
浩天
球
新建会话
新建会话 (2)

Database changed
mysql> select * from student
    -> select * from student;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresp
ar 'select * from student' at line 2
mysql> select * from student;
+------+-------+--------+------+
| id   | name  | gender | age  |
+------+-------+--------+------+
|    1 | Li    | F      |   23 |
|    2 | Wang  | M      |   24 |
|    3 | Zhang | M      |   26 |
|    4 | Liu   | M      |   27 |
+------+-------+--------+------+
4 rows in set (0.00 sec)
```

图四：jar 包执行完毕后 student 表内容

Spark Streaming 实验结果：



图五：生产者终端展示



图六：消费者统计词频结果

二、简要描述实验做了哪些工作?

实验一：编写 Scala 程序完成了对于两个文本文件数据的合并与去重工作。

实验二：首先在服务器上安装并配置了 MySQL，之后通过编写程序完成了在 Spark 中通过 JDBC 连接 MySQL 数据库，编写 Scala 程序向 MySQL 中写入了部分数据。

实验三：首先安装并启动了 Kafka 程序，之后使用 Scala 编写了消费者方面 Spark Streaming 的程序(统计词频)，通过生产者输入词，消费者处理并打印结果，测试了 Spark Streaming 的工作原理和过程。

三.实验过程中遇到的问题和解决办法?

1.去重后排序问题

问题描述：在第一个实验中，去重之后的结果并没有根据第一列的顺序排序，虽然结果正确，但输出是杂乱无章的

```
[root@master mergeFiles]# hadoop fs -cat /mergeFiles/part-00000
21/05/07 17:12:29 WARN util.NativeCodeLoader: Unable to load native-hadoop library for y
20170105 z
20170102 y
20170103 x
[root@master mergeFiles]# hadoop fs -cat /mergeFiles/part-00001
21/05/07 17:12:48 WARN util.NativeCodeLoader: Unable to load native-hadoop library for y
20170106 z
[root@master mergeFiles]# hadoop fs -cat /mergeFiles/part-00002
21/05/07 17:12:52 WARN util.NativeCodeLoader: Unable to load native-hadoop library for y
20170101 x
20170104 y
[root@master mergeFiles]# hadoop fs -cat /mergeFiles/part-00003
21/05/07 17:12:56 WARN util.NativeCodeLoader: Unable to load native-hadoop library for y
20170104 z
20170101 y
20170105 y
[root@master mergeFiles]#
```

图-1 未排序结果

错误分析：使用 distinct()方法后并没有再次进行排序操作，修改成另一方法，可直接根据第一列进行排序

```
val sparkConf = new SparkConf().setAppName("MergeFiles").setMaster("yarn")
val sc = new SparkContext(sparkConf)
val file1 = sc.textFile( path = "words1.txt")
val file2 = sc.textFile( path = "words2.txt")
val file3 = file1.union(file2)
val result = file3.distinct()
```

图-2 直接使用 distinct()方法后不做任何操作

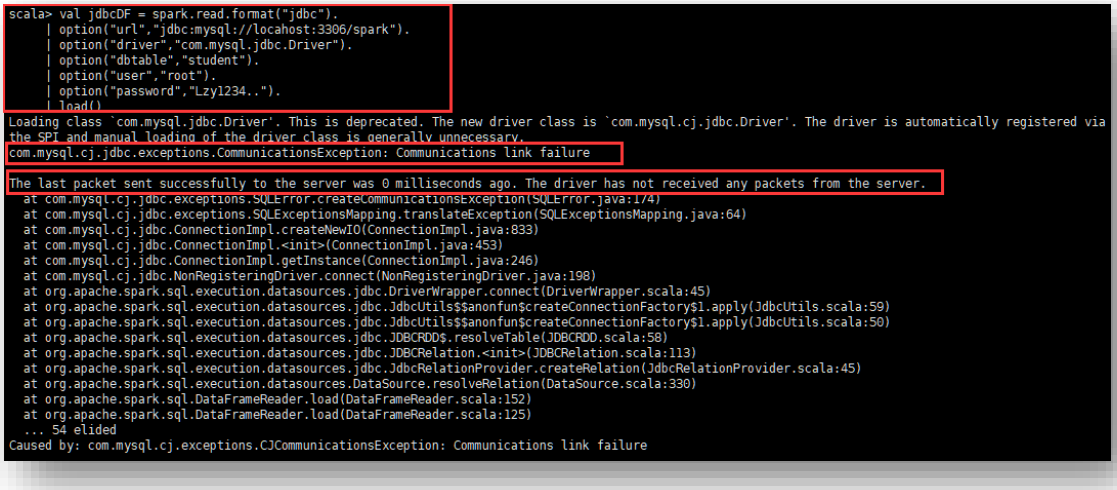
```
val sparkConf = new SparkConf().setAppName("MergeFiles").setMaster("yarn")
val sc = new SparkContext(sparkConf)
val file1 = sc.textFile( path = "words1.txt")
val file2 = sc.textFile( path = "words2.txt")
val file3 = file1.union(file2)
val result = file3.filter(_.trim().length()>0 ).map( line=>(line.trim,"") ).groupByKey()
  .sortByKey()
  .map(_._1)

result.saveAsTextFile( path = "hdfs://master:9000/merge")
```

图-3 修改后程序

问题思考：这个问题暴露出我在做实验时对于每一步操作的检查不够细致，虽然结果正确，但是排序操作耗费了我大量时间，究其原因是对于 Scala 代码语法的不熟悉，对于 Scala 程序编写的生疏，这也警示自己做实验前一定要做好充足的准备，在实验过程中也要步步为营，稳扎稳打的进行实验。

2.Spark 连接 Mysql 数据库出错

问题描述：按照实验指导书启动 spark shell 后并输入完命令后，报错



```
scala> val jdbcDF = spark.read.format("jdbc").
     | option("url","jdbc:mysql://locahost:3306/spark").
     | option("driver","com.mysql.jdbc.Driver").
     | option("dbtable","student").
     | option("user","root").
     | option("password","Lzy1234..").
     | load()
Loading class `com.mysql.jdbc.Driver'. This is deprecated. The new driver class is `com.mysql.cj.jdbc.Driver'. The driver is automatically registered via the SPI and manual loading of the driver class is generally unnecessary.
com.mysql.cj.jdbc.exceptions.CommunicationsException: Communications link failure

The last packet sent successfully to the server was 0 milliseconds ago. The driver has not received any packets from the server.
  at com.mysql.cj.jdbc.exceptions.SQLError.createCommunicationsException(SQLError.java:174)
  at com.mysql.cj.jdbc.exceptions.SQLExceptionsMapping.translateException(SQLExceptionsMapping.java:64)
  at com.mysql.cj.jdbc.ConnectionImpl.createNewIO(ConnectionImpl.java:833)
  at com.mysql.cj.jdbc.ConnectionImpl.<init>(ConnectionImpl.java:453)
  at com.mysql.cj.jdbc.ConnectionImpl.getInstance(ConnectionImpl.java:246)
  at com.mysql.cj.jdbc.NonRegisteringDriver.connect(NonRegisteringDriver.java:198)
  at org.apache.spark.sql.execution.datasources.jdbc.DriverWrapper.connect(DriverWrapper.scala:45)
  at org.apache.spark.sql.execution.datasources.jdbc.JdbcUtils$$anonfun$createConnectionFactory$1.apply(JdbcUtils.scala:59)
  at org.apache.spark.sql.execution.datasources.jdbc.JdbcUtils$$anonfun$createConnectionFactory$1.apply(JdbcUtils.scala:50)
  at org.apache.spark.sql.execution.datasources.jdbc.JDBCRDD$.resolveTable(JDBCRDD.scala:58)
  at org.apache.spark.sql.execution.datasources.jdbc.JDBCRelation.<init>(JDBCRelation.scala:113)
  at org.apache.spark.sql.execution.datasources.jdbc.JdbcRelationProvider.createRelation(JdbcRelationProvider.scala:45)
  at org.apache.spark.sql.execution.datasources.DataSource.resolveRelation(DataSource.scala:330)
  at org.apache.spark.sql.DataFrameReader.load(DataFrameReader.scala:152)
  at org.apache.spark.sql.DataFrameReader.load(DataFrameReader.scala:125)
  ... 54 elided
Caused by: com.mysql.cj.exceptions.CJCommunicationsException: Communications link failure
```

图-4 连接出错

错误分析：在查看群里的同样问题的解决方案后，发现是 option(“url”,” jdbc:mysql://locahost:3306/spark”) 打错了，localhost 少了一个 l，修改后连接成功

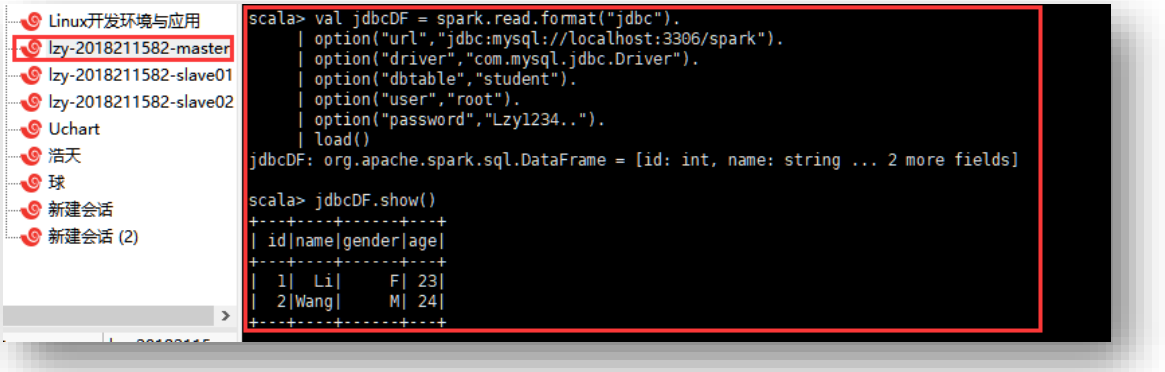


图-5 修改后成功

问题思考：实验步骤不够细致，只有需要问题了才反应过来，这种粗心的错误不应该出现，之后做实验时一定要注意。

四.实验代码

1. RDD 编程实验代码

pom.xml (spark-test) | KafkaStreaming.scala | ScalaWordCount.scala | ScalaMergeFiles.scala | InsertStudent.scala

```scala
package org.example

import org.apache.spark.{ SparkConf, SparkContext}

class ScalaMergeFiles {

}
object ScalaMergeFiles{
  def main(args: Array[String]): Unit = {
    val sparkConf = new SparkConf().setAppName("MergeFiles").setMaster("yarn")
    val sc = new SparkContext(sparkConf)
    val file1 = sc.textFile( path = "words1.txt")
    val file2 = sc.textFile( path = "words2.txt")
    val file3 = file1.union(file2)
    val result = file3.filter(_.trim().length()>0 ).map( line=>(line.trim,"") ).groupByKey()
      .sortByKey()
      .map(_._1)

    result.saveAsTextFile( path = "hdfs://master:9000/merge")
    sc.stop()
  }

}
```

```
package org.example


import org.apache.spark.{ SparkConf, SparkContext}


class ScalaMergeFiles {


}

object ScalaMergeFiles{

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

    val sparkConf = new SparkConf().setAppName("MergeFiles").setMaster("yarn
")

    val sc = new SparkContext(sparkConf)

    val file1 = sc.textFile("words1.txt")

    val file2 = sc.textFile("words2.txt")

    val file3 = file1.union(file2)

    val result = file3.filter(_.trim().length()>0 ).map( line=>(line.trim,"
") ).groupByKey()

      .sortByKey()

      .map(_._1)


    result.saveAsTextFile("hdfs://master:9000/merge")

    sc.stop()

  }


}
```

2.JDBC 连接数据库实验代码

pom.xml (spark-test) | KafkaStreaming.scala | ScalaWordCount.scala | ScalaMergeFiles.scala | InsertStudent.scala

```scala
import java.util.Properties

import org.apache.spark.sql.{Row, SQLContext}
import org.apache.spark.sql.types.{IntegerType, StringType, StructField, StructType}
import org.apache.spark.{SparkConf, SparkContext}

class InsertStudent {

}
object InsertStudent{
  def main(args: Array[String]): Unit = {
    val sparkConf = new SparkConf().setAppName("insert-student").setMaster("local")
    val sc = new SparkContext(sparkConf)
    val studentRDD = sc.parallelize(Array("3 Zhang M 26","4 Liu M 27")).map(_.split( regex = " "))
    val scheme = StructType(List(
      StructField("id",IntegerType,true),
      StructField("name",StringType,true),
      StructField("gender",StringType,true),
      StructField("age",IntegerType,true)
    ))
    val rowRDD = studentRDD.map( p => Row(p(0).toInt,p(1).trim,p(2).trim,p(3).toInt))
    val studentDF = new SQLContext(sc).createDataFrame(rowRDD,scheme)
    val prop = new Properties()
    prop.put("user","root")
    prop.put("password","Lzy1234..")
    prop.put("driver","com.mysql.jdbc.Driver")
    studentDF.write.mode( saveMode = "append").jdbc( url = "jdbc:mysql://localhost:3306/spark",
      table = "spark.student",prop)
  }
}
```

```scala
package org.example


import java.util.Properties


import org.apache.spark.sql.{Row, SQLContext}

import org.apache.spark.sql.types.{IntegerType, StringType, StructField, StructType}

import org.apache.spark.{SparkConf, SparkContext}


class InsertStudent {


}

object InsertStudent{

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

```
    val sparkConf = new SparkConf().setAppName("insert-student").setMaster("
local")

    val sc = new SparkContext(sparkConf)

    val studentRDD = sc.parallelize(Array("3 Zhang M 26","4 Liu M 27")).map
(_.split(" "))

    val scheme = StructType(List(

      StructField("id",IntegerType,true),

      StructField("name",StringType,true),

      StructField("gender",StringType,true),

      StructField("age",IntegerType,true)

    ))

    val rowRDD = studentRDD.map( p => Row(p(0).toInt,p(1).trim,p(2).trim,p
(3).toInt))

    val studentDF = new SQLContext(sc).createDataFrame(rowRDD,scheme)

    val prop = new Properties()

    prop.put("user","root")

    prop.put("password","Lzy1234..")

    prop.put("driver","com.mysql.jdbc.Driver")

    studentDF.write.mode("append").jdbc("jdbc:mysql://localhost:3306/spark
",

      "spark.student",prop)

  }

}
```

3.Spark Streaming 实验代码

pom.xml (spark-test) | KafkaStreaming.scala | ScalaWordCount.scala | ScalaMergeFiles.scala | InsertStudent.scala

```scala
package org.example

import org.apache.spark.streaming.kafka.KafkaUtils
import org.apache.spark.streaming.{Seconds, StreamingContext}
import org.apache.spark.{SparkConf, SparkContext}

class KafkaStreaming {

}
object KafkaStreaming{
  def main(args: Array[String]): Unit = {
    val sparkConf = new SparkConf().setAppName("kafka_test").setMaster("local[2]")
    val ssc = new StreamingContext(sparkConf,Seconds(10))

    val zkQuorum = "localhost:2181"
    val group = "1"
    val topics = "wordsendertest"
    val numThreads = 1
    val topicMap = topics.split(regex = ",").map((_,numThreads.toInt)).toMap
    val lineMap = KafkaUtils.createStream(ssc,zkQuorum,group,topicMap)

    val lines = lineMap.map(_._2)
    val words = lines.flatMap(_.split(regex = " "))
    val wordCounts = words.map((_,1)).reduceByKey(_+_)
    wordCounts.print()

    ssc.start()
    ssc.awaitTermination()
  }
}
```

```scala
package org.example



import org.apache.spark.streaming.kafka.KafkaUtils

import org.apache.spark.streaming.{Seconds, StreamingContext}

import org.apache.spark.{SparkConf, SparkContext}



class KafkaStreaming {



}

object KafkaStreaming{

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

    val sparkConf = new SparkConf().setAppName("kafka_test").setMaster("loca
l[2]")

    val ssc = new StreamingContext(sparkConf,Seconds(10))
```

```
    val zkQuorum = "localhost:2181"

    val group = "1"

    val topics = "wordsendertest"

    val numThreads = 1

    val topicMap = topics.split(",").map((_,numThreads.toInt)).toMap

    val lineMap = KafkaUtils.createStream(ssc,zkQuorum,group,topicMap)



    val lines = lineMap.map(_._2)

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

    val wordCounts = words.map((_,1)).reduceByKey(_+_)

    wordCounts.print()



    ssc.start()

    ssc.awaitTermination()

  }

}
```