|  |
| --- |
|  |
| RDD Functions |
|  |  |
|  | **\*\*Splitting rows of info\*\*** |
|  |  |
|  | import org.apache.spark.\_ |
|  |  |
|  | val data = sc.parallelize(Array("IN01 2016 01 00100000001 20160115", |
|  | "AU01 2016 01 00100000001 20170115")) |
|  | val pattern = """(\S+) (\S+) (\S+) (\S+) (\S+)""".r |
|  |  |
|  | //Case class for the document header |
|  |  |
|  | case class financialDocumentHeader(companyCode: String, |
|  | year: String, |
|  | month: String, |
|  | documentNumber: String, |
|  | postingDate: String |
|  | ) |
|  |  |
|  | def parseFinancialHeaderDocumentLine(line: String): financialDocumentHeader = { |
|  | val patternMatch = pattern.findFirstMatchIn(line) |
|  | if(patternMatch.isEmpty){ |
|  | throw new RuntimeException("cannot parse line:" + line) |
|  | } |
|  | val lineData = patternMatch.get |
|  | financialDocumentHeader(lineData.group(1), lineData.group(2), lineData.group(3), lineData.group(4), lineData.group(5)) |
|  | } |
|  |  |
|  | val dataHeaderDocs = data.map(parseFinancialHeaderDocumentLine) |
|  | println("Results \n -------") |
|  | println(dataHeaderDocs.collect().mkString("\n")) |
|  |  |
|  |  |
|  | //map |
|  | import org.apache.spark.\_ |
|  | val companyCodeRDDx = sc.parallelize(Array("GB01","US01","US01","UK01")) |
|  | val companyCodeRDDy = companyCodeRDDx.map(obj => (obj,1)) |
|  |  |
|  | println("\n Results \n -----------") |
|  | println("CompanyCodeRDDx : " + companyCodeRDDx.toDF.show()) |
|  | println("CompanyCodeRDDy : " + companyCodeRDDy.toDF.show()) |
|  |  |
|  | //filter |
|  | import org.apache.spark.\_ |
|  | val companyCodeRDDx = sc.parallelize(Array("GB01","US01","US01","UK01")) |
|  | val companyCodeRDDy = companyCodeRDDx.filter(obj => obj == "US01") |
|  |  |
|  | println("\n Results \n -----------") |
|  | println("CompanyCodeRDDx : " + companyCodeRDDx.toDF.show()) |
|  | println("CompanyCodeRDDy : " + companyCodeRDDy.toDF.show()) |
|  |  |
|  | //keyBy |
|  | import org.apache.spark.\_ |
|  | val companyCodeRDDx = sc.parallelize(Array("GB01","US01","US01","UK01")) |
|  | val companyCodeRDDy = companyCodeRDDx.keyBy(obj => obj.substring(0,2)) |
|  | println(companyCodeRDDy.toDF.show()) |
|  |  |
|  | //CountByKey Sample |
|  | import org.apache.spark.\_ |
|  | import scala.collection.\_ |
|  |  |
|  | val sqlContext= new org.apache.spark.sql.SQLContext(sc) |
|  | import sqlContext.implicits.\_ |
|  |  |
|  | val companyCodeRDDx = sc.parallelize(Array(("GB01",100),("US01",50),("US01",75),("UK01",100))) |
|  | val companyCodeRDDy = companyCodeRDDx.sample(false,0.5) |
|  | println("\n Results \n -----------") |
|  | println("CompanyCodeRDDx : " + companyCodeRDDx.toDF.show()) |
|  | println("CompanyCodeRDDy : " + companyCodeRDDy.toDF.show()) |
|  |  |
|  |  |
|  | //Distinct |
|  | import org.apache.spark.\_ |
|  |  |
|  | val companyCodeRDDx = sc.parallelize(Array("GB01","US01","UK01","US01")) |
|  | val companyCodeRDDy = companyCodeRDDx.distinct() |
|  |  |
|  | println("\n Results \n -----------") |
|  | println("CompanyCodeRDDx : " + companyCodeRDDx.toDF.show()) |
|  | println("CompanyCodeRDDy : " + companyCodeRDDy.toDF.show()) |
|  |  |
|  |  |
|  | //ReduceByKey |
|  | import org.apache.spark.\_ |
|  |  |
|  | val companyCodeRDDx = sc.parallelize(Array(("GB01",100),("US01",50),("US01",75),("UK01",100))) |
|  | val companyCodeRDDy = companyCodeRDDx.reduceByKey(\_+\_) |
|  |  |
|  | println("\n Results \n -----------") |
|  | println("CompanyCodeRDDx : " + companyCodeRDDx.toDF.show()) |
|  | println("CompanyCodeRDDy : " + companyCodeRDDy.toDF.show()) |
|  |  |
|  |  |
|  | //Union |
|  | import org.apache.spark.\_ |
|  |  |
|  | val companyCodeRDDx = sc.parallelize(Array(("GB01",100),("US01",50),("US01",75),("UK01",100))) |
|  | val companyCodeRDDy = sc.parallelize(Array(("AU01",100))) |
|  |  |
|  | val companyCodeRDDz = companyCodeRDDx.union(companyCodeRDDy) |
|  |  |
|  | println("\n Results \n ------------") |
|  | println("companyCodeRDDz : " + companyCodeRDDz.toDF.show()) |
|  |  |
|  |  |
|  |  |
|  | //Join |
|  | import org.apache.spark.\_ |
|  |  |
|  | val companyCodeRDD = sc.parallelize(Array(("GB01",100),("US01",50),("US01",75),("UK01",100))) |
|  | val companyCodeAttrRDD = sc.parallelize(Array(("AU01","AU"),("GB01","GB"),("US01","US"))) |
|  |  |
|  | val companyCodeJoinRDD = companyCodeRDD.join(companyCodeAttrRDD) |
|  | val companyCodeLeftJoinRDD = companyCodeRDD.leftOuterJoin(companyCodeAttrRDD) |
|  |  |
|  | println("\n Results \n ------------") |
|  | println("companyCodeRDD : " + companyCodeRDD.toDF.show()) |
|  | println("companyCodeAttrRDD : " + companyCodeAttrRDD.toDF.show()) |
|  | println("companyCodeJoinRDD : " + companyCodeJoinRDD.toDF.show()) |
|  | println("companyCodeLeftJoinRDD : " + companyCodeLeftJoinRDD.toDF.show()) |
|  |  |
|  |  |
|  |  |
|  | //FlatMap |
|  | import org.apache.spark.\_ |
|  |  |
|  | val companyCodeRDD = sc.parallelize(Array("GB,01","AU,01","IN,01","UK,01")) |
|  | val companyCodeRDDFlat = companyCodeRDD.flatMap(a => a.split(",")) |
|  |  |
|  | println("\n Results \n ------------") |
|  | println("companyCodeRDDFlat : " + companyCodeRDDFlat.toDF.show()) |
|  |  |
|  |  |
|  | //Map and Split |
|  | import org.apache.spark.\_ |
|  |  |
|  | val companyCodeRDD = sc.parallelize(Array(("GB01,100"),("US01,50"),("US01,75"),("UK01,100"))) |
|  | companyCodeRDD.map(x => { |
|  | var splitStr = x.split(",") |
|  | (splitStr(0),splitStr(1)) |
|  | }) |
|  |  |
|  | println("\n Results \n ------------") |
|  | println("companyCodeRDD : " + companyCodeRDD.toDF.show()) |
|  |  |
|  |  |
|  | //save a RDD to HDFS as a Textfile |
|  |  |
|  | import org.apache.hadoop.fs.FileSystem |
|  | import org.apache.hadoop.fs.Path |
|  | val fs = FileSystem.get(sc.hadoopConfiguration) |
|  |  |
|  | fs.delete(new Path("/user/vora/sapData"), true) |
|  | val sapDataRDD = sc.parallelize(Array("GB01,100","IN01,200","UK01 90")) |
|  | sapDataRDD.repartition(1).saveAsTextFile("sapData") |
|  |  |
|  |  |
|  | //import files to RDD from HDFS |
|  |  |
|  | import scala.sys.process.\_ |
|  |  |
|  | val hdfsFileRDD = sc.textFile("hdfs://vorahost.dummy.nodomain:8020/user/vora/sapData") |
|  | println("Result \n ------") |
|  | println(hdfsFileRDD.collect().mkString("\n")) |
|  |  |