Spark Programs

Big Data Lab

1. Demonstrate case classes and tuples in spark by creating an object, copying it, and checking for equality.

CaseClassExample.scala

case class CaseClassExample(v1: String, v2: String)  
  
object Test {  
 def main(args: Array[String]) : Unit = {  
 val caseClassExampleTest = CaseClassExample("abc", "def")  
  
 println(caseClassExampleTest.v1)  
 println(caseClassExampleTest.v2)  
 }  
}

build.sbt

ThisBuild / *version* := "0.1.0-SNAPSHOT"  
  
ThisBuild / *scalaVersion* := "2.11.8"  
  
lazy val root = (project in *file*("."))  
 .settings(  
 *name* := "Case Class & Tuples"  
 )

2. Demonstrate WordCount using spark with scala.

WordCount.scala

import org.apache.spark.sql.SparkSession  
import org.apache.spark.rdd.RDD  
  
object WordCount {  
 def main(args: Array[String]) {  
 val file = "src/main/words.txt"  
 val spark: SparkSession = SparkSession.builder  
 .appName("Word Count")  
 .config("spark.master", "local")  
 .getOrCreate()  
 val fileRdd: RDD[String] = spark.sparkContext.textFile(file)  
  
 // create the counts  
 val counts = fileRdd.map(\_.replaceAll("[.,]", ""))  
 .map(\_.replace("—", " "))  
 .flatMap(line => line.split(" "))  
 .map(word => (word, 1))  
 .reduceByKey(\_ + \_)  
 .sortBy(\_.\_2)  
 .collect  
  
 println( "------------------------------------------")  
 counts.foreach(println)  
 println( "------------------------------------------")  
  
 spark.stop()  
 }  
  
}

build.sbt

ThisBuild / version := "0.1.0-SNAPSHOT"  
  
ThisBuild / scalaVersion := "2.11.8"  
  
lazy val root = (project in file("."))  
 .settings(  
 name := "WordCount"  
 )  
  
val sparkVersion = "2.4.0"  
  
libraryDependencies ++= Seq(  
 "org.apache.spark" %% "spark-core" % sparkVersion,  
 "org.apache.spark" %% "spark-sql" % sparkVersion,  
)

words.txt

hi  
hello  
hey  
hi  
hi

3. Demonstrate use of aggregate functions (approx\_count\_distinct(), collect\_list(), collect\_set(), avg(), count(), countDistinct()) by using SQL libraries in spark.

AggregateFunctions.scala

import org.apache.spark.sql.SparkSession  
import org.apache.spark.sql.functions.\_  
  
object AggregateFunctions extends App {  
 val *spark*: SparkSession = SparkSession.*builder*()  
 .master("local[1]")  
 .appName("SparkByExamples.com")  
 .getOrCreate()  
  
 *spark*.sparkContext.setLogLevel("ERROR")  
  
 import *spark*.implicits.\_  
  
 val *simpleData* = *Seq*(("James", "Sales", 3000),  
 ("Michael", "Sales", 4600),  
 ("Robert", "Sales", 4100),  
 ("Maria", "Finance", 3000),  
 ("James", "Sales", 3000),  
 ("Scott", "Finance", 3300),  
 ("Jen", "Finance", 3900),  
 ("Jeff", "Marketing", 3000),  
 ("Kumar", "Marketing", 2000),  
 ("Saif", "Sales", 4100)  
 )  
 val *df* = *simpleData*.toDF("employee\_name", "department", "salary")  
  
 *println*("approx\_count\_distinct: "+  
 *df*.select(*approx\_count\_distinct*("salary")).collect()(0)(0))  
  
 *println*("avg: "+ *df*.select(*avg*("salary")).collect()(0)(0))  
  
 *df*.select(*collect\_list*("salary")).show(false)  
  
 *df*.select(*collect\_set*("salary")).show(false)  
  
 val *df2* = *df*.select(*countDistinct*("department", "salary"))  
 *df2*.show(false)  
 *println*("Distinct Count of Department & Salary: "+*df2*.collect()(0)(0))  
  
 *println*("count: "+  
 *df*.select(*count*("salary")).collect()(0))  
}

build.sbt

ThisBuild / *version* := "0.1.0-SNAPSHOT"  
  
ThisBuild / *scalaVersion* := "2.11.8"  
  
lazy val root = (project in *file*("."))  
 .settings(  
 *name* := "Aggregate Functions"  
 )  
  
val sparkVersion = "2.4.0"  
  
*libraryDependencies* ++= *Seq*(  
 "org.apache.spark" %% "spark-core" % sparkVersion, "org.apache.spark" %% "spark-sql" % sparkVersion,  
)

4. Demonstrate basic dataframe transformation functions by using SQL libraries in spark.

TransformationFunctions.scala

import org.apache.spark.sql.SparkSession  
import org.apache.spark.sql.functions.\_  
  
object TransformationFunctions extends App {  
 val spark: SparkSession = SparkSession.builder()  
 .master("local[1]")  
 .appName("SparkByExamples.com")  
 .getOrCreate()  
  
 spark.sparkContext.setLogLevel("ERROR")  
  
 import spark.implicits.\_  
  
 val simpleData = Seq(("James", "Sales", 3000),  
 ("Michael", "Sales", 4600),  
 ("Robert", "Sales", 4100),  
 ("Maria", "Finance", 3000),  
 ("James", "Sales", 3000),  
 ("Scott", "Finance", 3300),  
 ("Jen", "Finance", 3900),  
 ("Jeff", "Marketing", 3000),  
 ("Kumar", "Marketing", 2000),  
 ("Saif", "Sales", 4100)  
 )  
 val df = simpleData.toDF("employee\_name", "department", "salary")  
 //show all  
 df.select("\*").show(truncate = false)  
  
 //first row  
 df.select(first("salary")).show(false)  
  
 //last row  
 df.select(last("salary")).show(false)  
  
 //Select first 2 columns.  
 df.select(df.columns.slice(0,2).map(m=>col(m)):\_\*).show()  
  
  
 //Selects 3rd column (index starts from zero)  
 df.select(df.columns(2)).show()  
  
 //Selects columns from index 1 to 3  
 df.select(df.columns.slice(1,3).map(m=>col(m)):\_\*).show()  
  
}

build.sbt

ThisBuild / version := "0.1.0-SNAPSHOT"  
  
ThisBuild / scalaVersion := "2.11.8"  
  
lazy val root = (project in file("."))  
 .settings(  
 name := "Transformation Functions"  
 )  
  
val sparkVersion = "2.4.0"  
libraryDependencies ++= Seq(  
 "org.apache.spark" %% "spark-core" % sparkVersion,  
 "org.apache.spark" %% "spark-sql" % sparkVersion,  
)

5. Demonstrate basic math functions on a dataframe using SQL libraries in spark.

MathFunctions.scala

import org.apache.spark.sql.SparkSession  
import org.apache.spark.sql.functions.\_  
  
object MathFunctions extends App {  
 val spark: SparkSession = SparkSession.builder()  
 .master("local[1]")  
 .appName("SparkByExamples.com")  
 .getOrCreate()  
  
 spark.sparkContext.setLogLevel("ERROR")  
  
 import spark.implicits.\_  
  
 val simpleData = Seq(("James", "Sales", 3000),  
 ("Michael", "Sales", 4600),  
 ("Robert", "Sales", 4100),  
 ("Maria", "Finance", 3000),  
 ("James", "Sales", 3000),  
 ("Scott", "Finance", 3300),  
 ("Jen", "Finance", 3900),  
 ("Jeff", "Marketing", 3000),  
 ("Kumar", "Marketing", 2000),  
 ("Saif", "Sales", 4100)  
 )  
 val df = simpleData.toDF("employee\_name", "department", "salary")  
  
 df.select(max("salary")).show(false)  
  
 df.select(min("salary")).show(false)  
  
 df.select(mean("salary")).show(false)  
  
 df.select(skewness("salary")).show(false)  
  
 df.select(stddev("salary"), stddev\_samp("salary"),  
 stddev\_pop("salary")).show(false)  
  
 df.select(sum("salary")).show(false)  
  
 df.select(sumDistinct("salary")).show(false)  
  
 df.select(variance("salary"),var\_samp("salary"),  
 var\_pop("salary")).show(false)  
  
}

build.sbt

ThisBuild / *version* := "0.1.0-SNAPSHOT"  
  
ThisBuild / *scalaVersion* := "2.11.8"  
  
lazy val root = (project in *file*("."))  
 .settings(  
 *name* := "Math Functions"  
 )  
  
val sparkVersion = "2.4.0"  
*libraryDependencies* ++= *Seq*(  
 "org.apache.spark" %% "spark-core" % sparkVersion,  
 "org.apache.spark" %% "spark-sql" % sparkVersion,  
)