**Code:**

d)

spark-shell --packages com.databricks:spark-avro\_2.11:4.0.0

import com.databricks.spark.avro.\_

import org.apache.spark.sql.SparkSession

import org.apache.spark.sql.SQLContext

val sqlContext = new SQLContext(sc)

import sqlContext.implicits.\_

val products = sqlContext.read.avro("hdfs://localhost/user/root/spark\_demo/scala/")

products.show

products.toDF()

products.printSchema()

val prodrows: org.apache.spark.rdd.RDD[org.apache.spark.sql.Row] = products.rdd

val categories = sqlContext.read.avro("hdfs://localhost/user/root/spark\_demo/")

categories.show

categories.toDF()

categories.printSchema()

categories.take(20).foreach(println)

val catrows: org.apache.spark.rdd.RDD[org.apache.spark.sql.Row] = categories.rdd

e)

import com.databricks.spark.avro.\_

import org.apache.spark.sql.SparkSession

import org.apache.spark.sql.SQLContext

val sqlContext = new SQLContext(sc)

val products = sqlContext.read.avro("hdfs://localhost/user/root/spark\_demo/scala/")

val prodrows1 = products.as[(String,String,String,String,String)].rdd

val clean = prodrows1.filter(\_.\_4.toString != "")

val hundred = clean.filter(\_.\_4.toFloat < 100)

hundred.foreach(println)

hundred.saveAsTextFile("hdfs://localhost/user/root/spark\_demo/Result/Result\_0")

f)

import com.databricks.spark.avro.\_

import org.apache.spark.sql.SparkSession

import org.apache.spark.sql.SQLContext

val sqlContext = new SQLContext(sc)

val join = categories.join(products, categories("category\_id") === products("\_2"))

import org.apache.spark.sql.functions.{row\_number, max, broadcast}

import org.apache.spark.sql.expressions.Window

val prodorder = Window.partitionBy($"category\_id").orderBy($"\_4".desc)

val top10 = join.withColumn("rn", row\_number.over(prodorder)).where($"rn" <= 10).drop("rn").filter($"\_4" < 100)

scala> dfteens.printSchema

root

|-- category\_id: string (nullable = true)

|-- \_3: string (nullable = true)

|-- \_4: string (nullable = true)

scala> val newNames = Seq("c\_id", "name", "price")

newNames: Seq[String] = List(c\_id, name, price)

scala> val dfRenamed = dfteens.toDF(newNames: \_\*)

dfRenamed: org.apache.spark.sql.DataFrame = [c\_id: string, name: string ... 1 more field]

scala> dfRenamed.printSchema

root

|-- c\_id: string (nullable = true)

|-- name: string (nullable = true)

|-- price: string (nullable = true)

g)

import com.databricks.spark.avro.\_

import org.apache.spark.sql.SparkSession

import org.apache.spark.sql.SQLContext

val sqlContext = new SQLContext(sc)

import sqlContext.implicits.\_

import org.apache.spark.sql.functions.{row\_number, max, broadcast}

import org.apache.spark.sql.expressions.Window

val products = sqlContext.read.avro("hdfs://localhost/user/root/spark\_demo/scala/")

val categories = sqlContext.read.avro("hdfs://localhost/user/root/spark\_demo/")

val prodjoin = categories.join(products, categories("category\_id") === products("\_2"))

val prodcatorderdesc = Window.partitionBy($"category\_id").orderBy($"\_4".desc)

val prodcatorderasc = Window.partitionBy($"category\_id").orderBy($"\_4".asc)

val top1prodcat = prodjoin.withColumn("rn", row\_number.over(prodcatorderdesc)).where($"rn" === 1).drop("rn").filter($"\_4" < 100)

val down1prodcat = prodjoin.withColumn("rn", row\_number.over(prodcatorderasc)).where($"rn" === 1).drop("rn").filter($"\_4" < 100)

val one=top1prodcat.select($"category\_id",$"category\_name",$"\_1",$"\_2",$"\_3", $"\_4", $"\_5")

val cat1 = one.withColumnRenamed("\_1","number1")

val cat2 = cat1.withColumnRenamed("\_2","number2")

val cat3 = cat2.withColumnRenamed("\_3","highest\_product\_name")

val cat4 = cat3.withColumnRenamed("\_4","highest\_product\_price")

val cat5 = cat4.withColumnRenamed("\_5","images")

val fin1 = cat5.select($"category\_id", $"highest\_product\_name", $"highest\_product\_price")

val two=down1prodcat.select($"category\_id",$"category\_name",$"\_1",$"\_2",$"\_3", $"\_4", $"\_5")

val cat11 = two.withColumnRenamed("\_1","number1")

val cat22 = cat11.withColumnRenamed("\_2","number2")

val cat33 = cat22.withColumnRenamed("\_3","lowest\_product\_name")

val cat44 = cat33.withColumnRenamed("\_4","lowest\_product\_price")

val cat55 = cat44.withColumnRenamed("\_5","images")

val cat555 = cat55.withColumnRenamed("category\_id","cid")

val cat5555 = cat555.withColumnRenamed("category\_name","cname")

val fin2 = cat5555.select($"cid",$"cname", $"lowest\_product\_name", $"lowest\_product\_price")

val finaljoin = fin2.join(fin1, fin2("cid") === fin1("category\_id"))

val finaloutput = finaljoin.select($"cname",$"highest\_product\_name",$"highest\_product\_price",$"lowest\_product\_name",$"lowest\_product\_price")

finaloutput.write.format("com.databricks.spark.avro").save("hdfs://localhost/user/spark\_demo/Result\_2.avro")