Note:

To enter spark shell use the following: spark-shell --packages com.databricks:spark-avro\_2.11:4.0.0

**Code for step d.**

val categories = spark.read.format(“com.databricks.spark.avro”).load(“/home/vagrant/week-13/categories.avro”)

categories.take(20).foreach(println)

val products = spark.read.format(“com.databricks.spark.avro”).load(“/home/vagrant/week-13/products.avro”)

products.take(20).foreach(println)

**Code for step e.**

import com.databricks.spark.avro.\_

import org.apache.spark.sql.functions

val sqlContext = new org.apache.spark.sql.SQLContext(sc)

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

val a = sqlContext.read.avro("/home/vagrant/week-13/products.avro")

val b = sqlContext.read.avro("/home/vagrant/week-13/categories.avro")

val boserdd = a.as[(String,String,String,String,String)].rdd

val firstrdd = boserdd.filter(\_.\_4.toString!="")

val secondrdd = firstrdd.filter(\_.\_4.toFloat < 100.00)

val result\_0 = secondrdd.saveAsTextFile("Result\_0")

**Code for step f.**

import com.databricks.spark.avro.\_

import org.apache.spark.sql.functions

val sqlContext = new org.apache.spark.sql.SQLContext(sc)

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

val a1 = sqlContext.read.avro("/home/vagrant/week-13/products.avro")

val a = a1.withColumn("\_4", '\_4.cast("Float"))

val b = sqlContext.read.avro("/home/vagrant/week-13/categories.avro")

val ab = a.join(b, a.col("\_2") === b.col("category\_id"))

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

val rankTest = dense\_rank().over(partitionWindow)

ab.select($"\*")

val ass = ab.select($"\*")

val asstop = ass.withColumn("rn", row\_number.over(partitionWindow)).where($"rn" <= 10).drop("rn")

val result = asstop.select($"category\_name",$"\_3".alias("product\_name"),$"\_4".alias("product\_price"))

result.show()

result.coalesce(1).write.option("header","true").csv("Result\_1")

**Code for step g.**

import com.databricks.spark.avro.\_

import org.apache.spark.sql.functions

val sqlContext = new org.apache.spark.sql.SQLContext(sc)

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

sqlContext.setConf("spark.sql.avro.compression.codec","snappy")

val a1 = sqlContext.read.avro("/home/vagrant/week-13/products.avro")

val a = a1.withColumn("\_4", '\_4.cast("Float"))

val b = sqlContext.read.avro("/home/vagrant/week-13/categories.avro")

val ab = a.join(b, a.col("\_2") === b.col("category\_id"))

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

val rankTest = dense\_rank().over(partitionWindow)

ab.select($"\*")

val ass = ab.select($"\*")

val asstop = ass.withColumn("rn", row\_number.over(partitionWindow)).where($"rn" === 1).drop("rn")

val assbottom = ass.withColumn("rn",row\_number.over(partitionWindow)).where($"rn" === 24).drop("rn")

val top = asstop.select($"category\_name",$"\_3".alias("highest\_product\_name"),$"\_4".alias("Highest\_product\_price"))

val bottom = assbottom.select($"category\_name",$"\_3".alias("lowest\_product\_name"),$"\_4".alias("Lowest\_product\_price"))

val result = top.join(bottom, "category\_name")

result.show()

result.coalesce(1).write.avro("/home/vagrant/Result\_2")