sqoop import --connect jdbc:mysql://nn01.itversity.com/retail\_db --username retail\_dba --password itversity --table products --target-dir /user/rss161030/cloudera/products --fields-terminated-by \|

hadoop fs -mv /user/rss161030/cloudera/products/\* /user/rss161030/cloudera/problem2/products

chmod 4 read 2 write 1 execute

chmod 765

hadoop fs -chmod -R 765 /user/rss161030/cloudera/problem2/products

product\_id | int(11) | NO | PRI | NULL | auto\_increment |

| product\_category\_id | int(11) | NO | | NULL | |

| product\_name | varchar(45) | NO | | NULL | |

| product\_description | varchar(255) | NO | | NULL | |

| product\_price | float | NO | | NULL | |

| product\_image | varchar(255)

val productsRDD = sc.textFile("/user/rss161030/cloudera/problem2/products")

case class Products(product\_id: Int , product\_category\_id: Int, product\_name: String, product\_description : String , product\_price: Double, product\_image : String)

val productsDF = productsRDD.map( rec => {val r = rec.split('|') Products(r(0).toInt,r(1).toInt,r(2),r(3),r(4).toDouble,r(5))}).toDF()

var products = sc.textFile("/user/cloudera/products").map(x=> {var d = x.split('|'); (d(0).toInt,d(1).toInt,d(2).toString,d(3).toString,d(4).toFloat,d(5).toString)});

case class Product(productID:Integer, productCatID: Integer, productName: String, productDesc:String, productPrice:Float, productImage:String);

var productsDF = products.map(x=> Product(x.\_1,x.\_2,x.\_3,x.\_4,x.\_5,x.\_6)).toDF();

val productsRDD = sc.textFile("/user/rss161030/cloudera/problem2/products").map( rec => {val r = rec.split('|'); (r(0).toInt,r(1).toInt,r(2).toString,r(3).toString,r(4).toDouble,r(5).toString)})

productsDF.registerTempTable("products")

val filteredProductsDF = sqlContext.sql("select \* from products where product\_price < 100")

filteredProductsDF.registerTempTable("filteredProductsDF")

val filteredProductsDF1 = sqlContext.sql("select product\_category\_id, MAX(product\_price) max\_price from filteredProductsDF group by product\_category\_id")

val filteredProductsDF2 = sqlContext.sql("select product\_category\_id, COUNT(distinct(product\_id)) num\_products from filteredProductsDF group by product\_category\_id")

val filteredProductsDF2 = sqlContext.sql("select product\_category\_id, AVG(product\_price) avg\_price from filteredProductsDF group by product\_category\_id")

val filteredProductsDF4 = sqlContext.sql("select product\_category\_id, MIN(product\_price) min\_price from filteredProductsDF group by product\_category\_id")

val resultSQL = sqlContext.sql("select product\_category\_id,MAX(product\_price) max\_price, COUNT(distinct(product\_id)) num\_products, AVG(product\_price) avg\_price, MIN(product\_price) min\_price from products where product\_price < 100 group by product\_category\_id order by product\_category\_id")

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

import com.databricks.spark.avro.\_

resultSQL.write.avro("/user/rss161030/cloudera/problem2/products/result-sql")