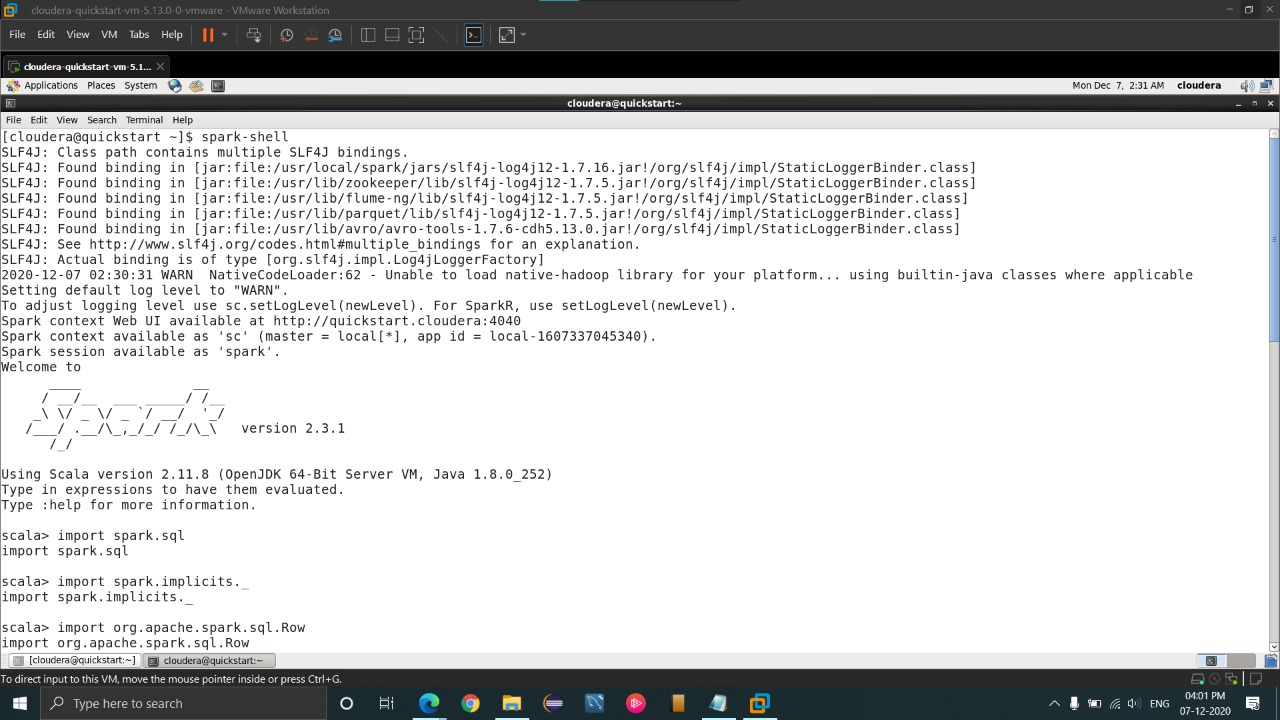
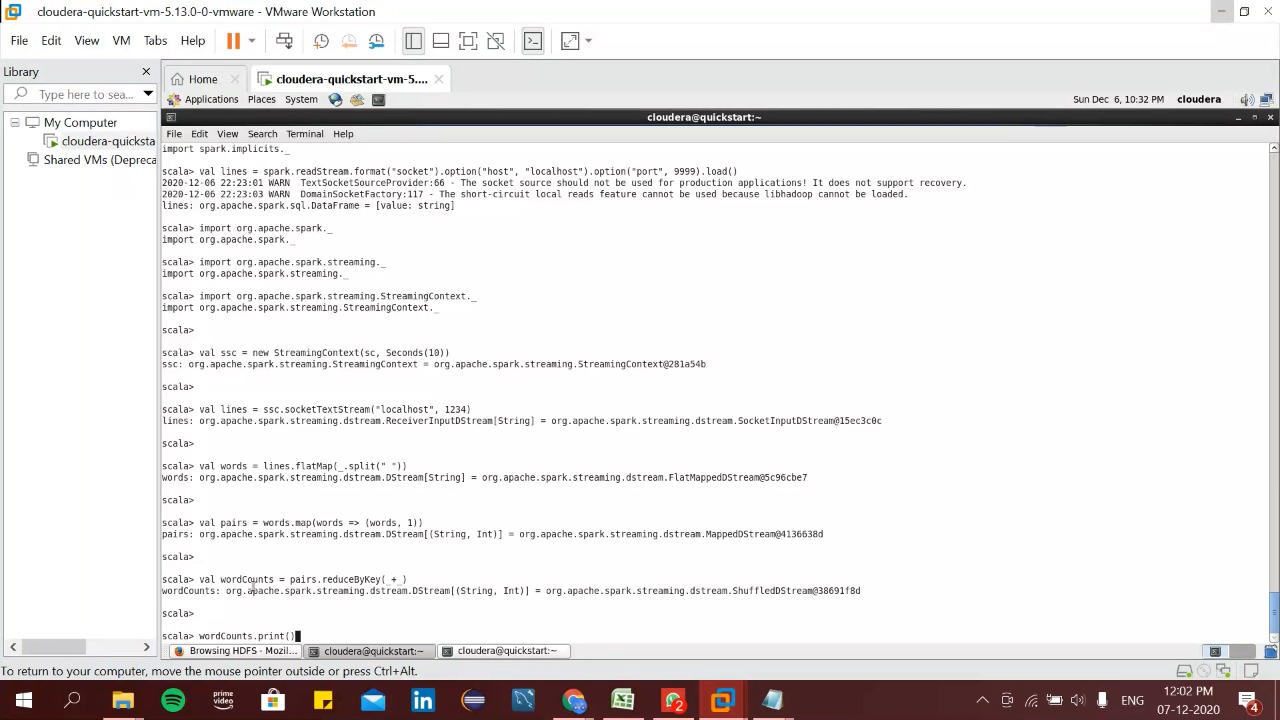
KO: Stream Analytics using Structured Streaming

Structured Streaming with localhost

Open the spark shell :



Import all the packages and use of word count program :



Import all the packages:

import org.apache.spark.sql.functions.\_

import org.apache.spark.sql.SparkSession

val spark = SparkSession.builder.appName("StructuredNetworkWordCount").getOrCreate()

import spark.implicits.\_

Now connect to localpost port 9999

val lines = spark.readStream.format("socket").option("host", "localhost").option("port", 9999).load()

Split the lines into words

val words = lines.as[String].flatMap(\_.split(" "))

Generate running word count

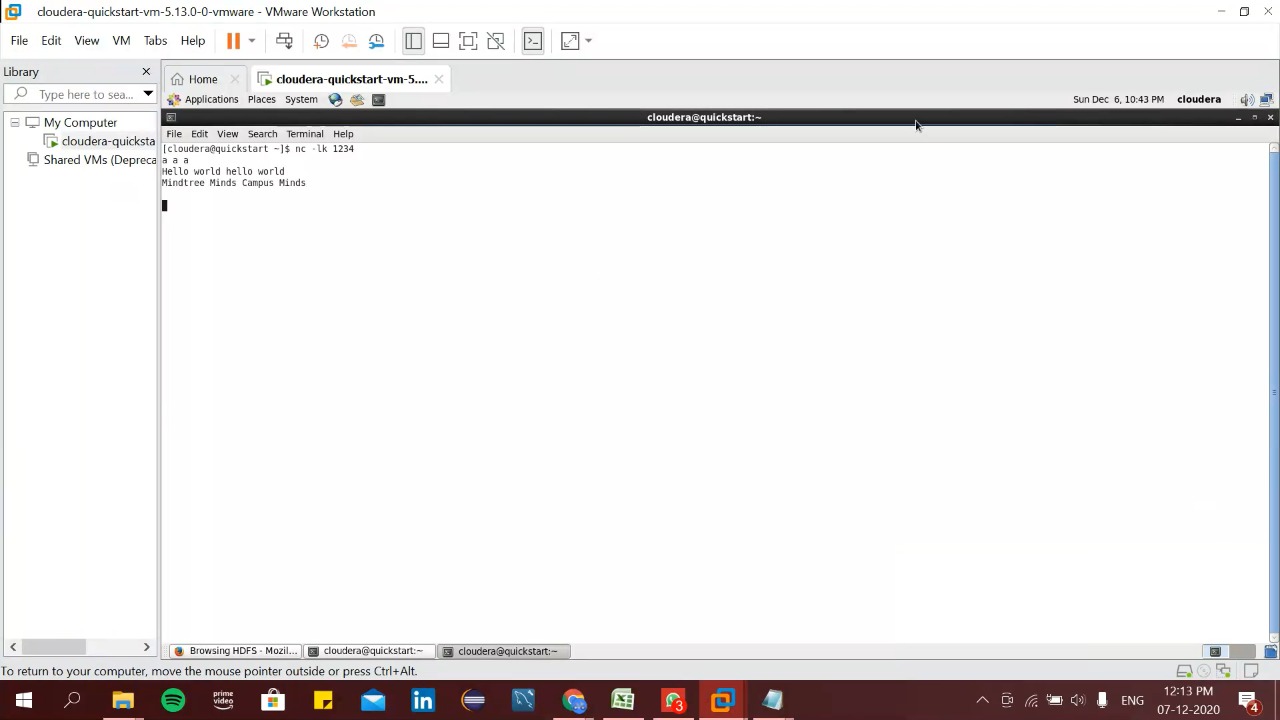
val wordCounts = words.groupBy("value").count()

Printing the output in the console

val query = wordCounts.writeStream.outputMode("complete").format("console").start()

query.awaitTermination()

Give the input to the socket streaming:

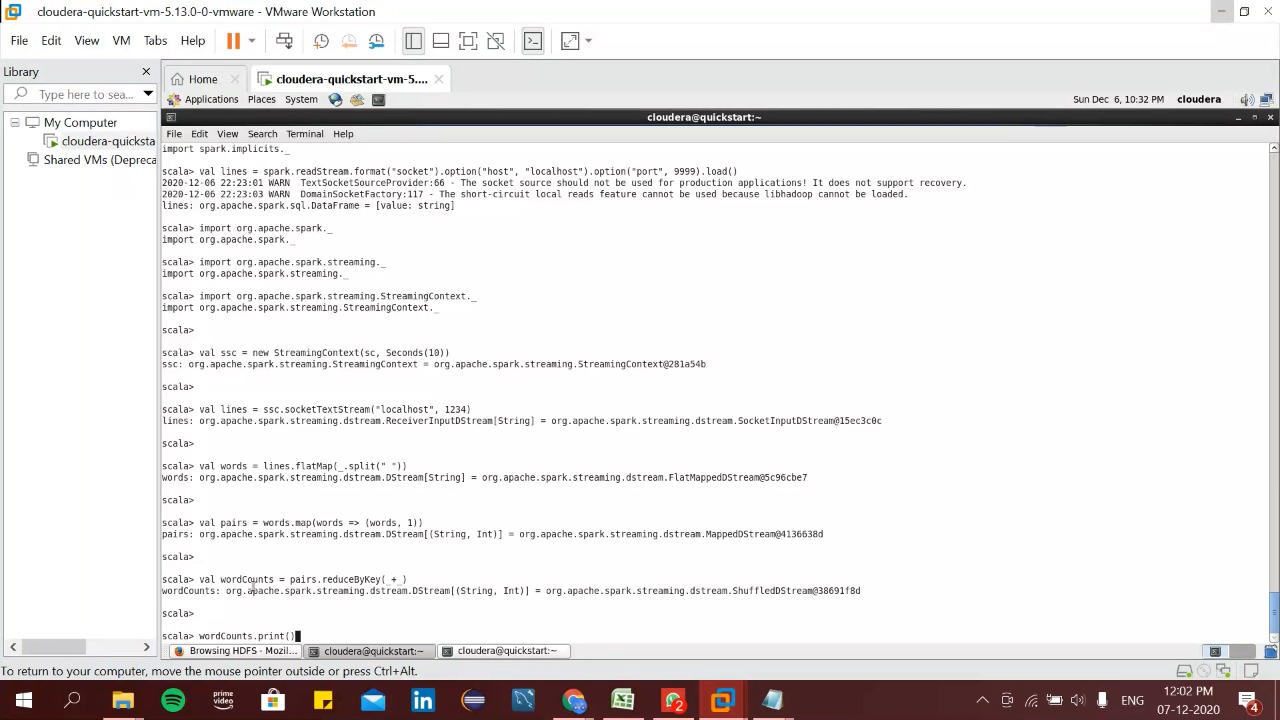


Connect to the local host using the command:

nc –lk 9999

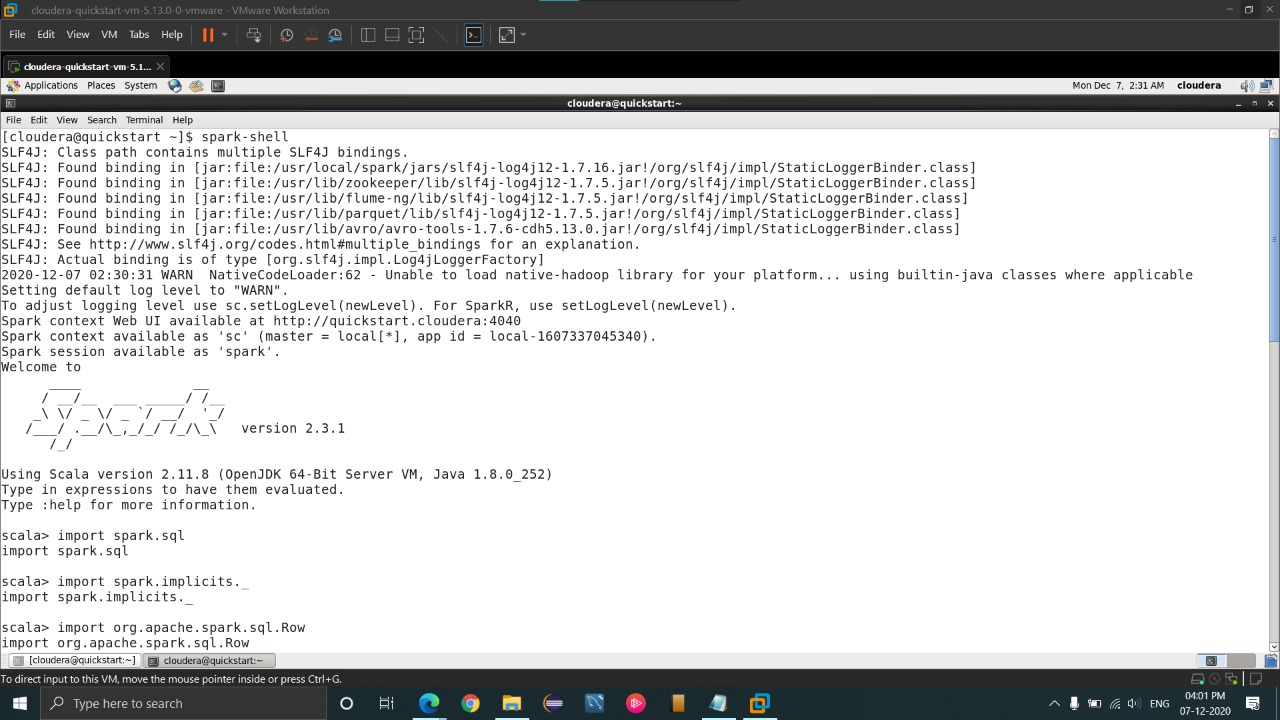
Type any word to get the output

Output of the program:

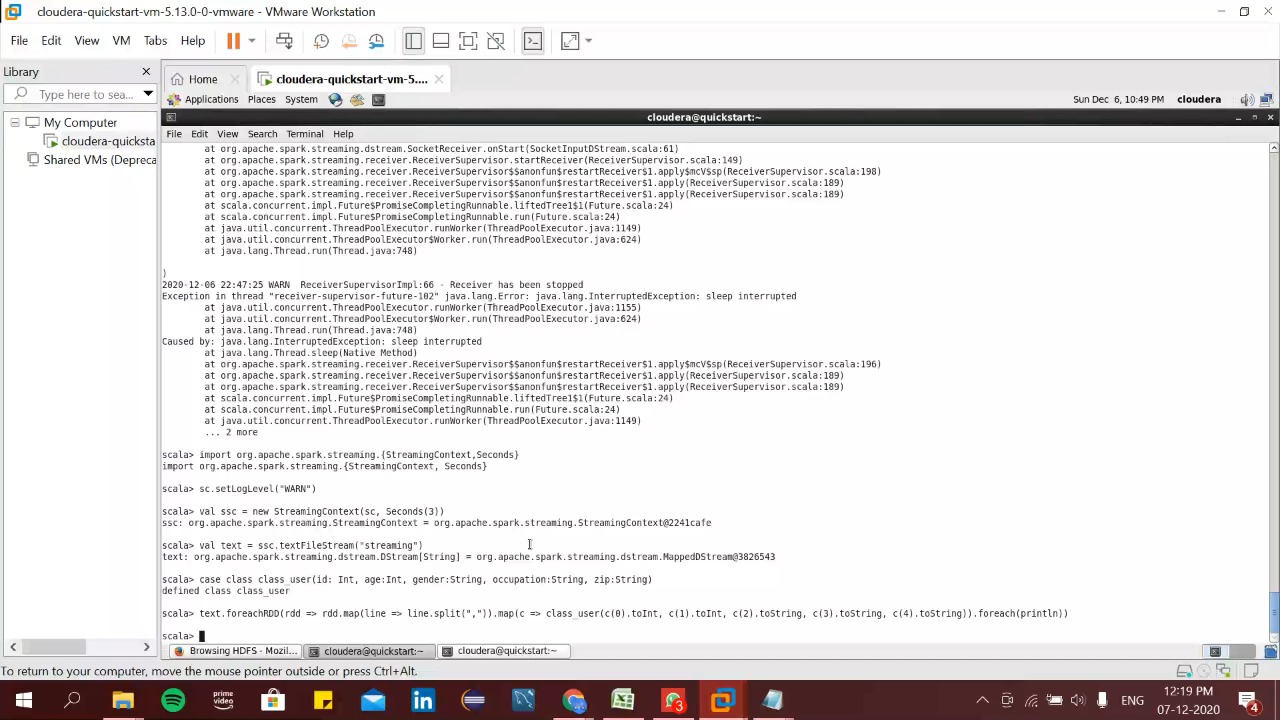


Textfile streaming:

Open the spark shell :



Create a class for textfile streaming:



Import the packages and create a class for the streaming using the following commands:

import org.apache.spark.streaming.{StreamingContext,Seconds}

sc.setLogLevel("WARN")

val ssc = new StreamingContext(sc, Seconds(3))

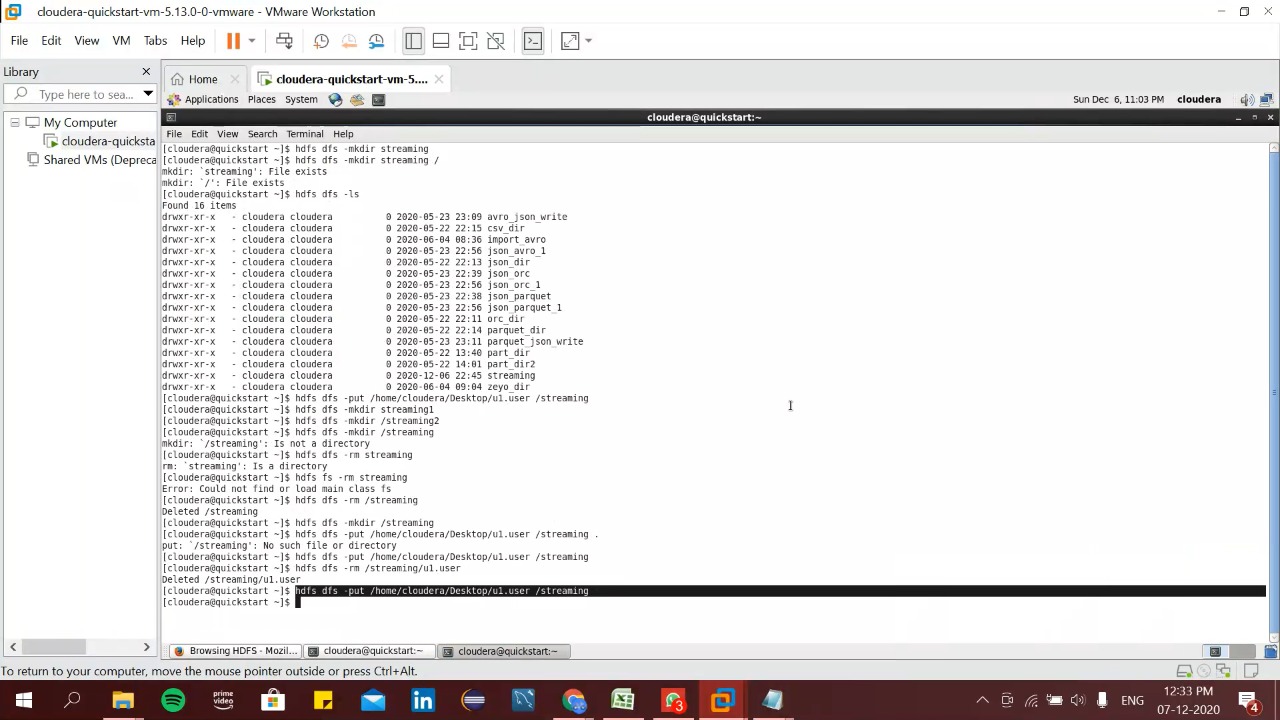
val soni = ssc.textFileStream("/streaming")

case class class\_user(id: Int, age:Int, gender:String, occupation:String, zip:String)

soni.foreachRDD(rdd => rdd.map(line => line.split(",")).map(c => class\_user(c(0).toInt, c(1).toInt, c(2).toString, c(3).toString, c(4).toString)).foreach(println))

ssc.start()

Creation of directory and putting file in it:

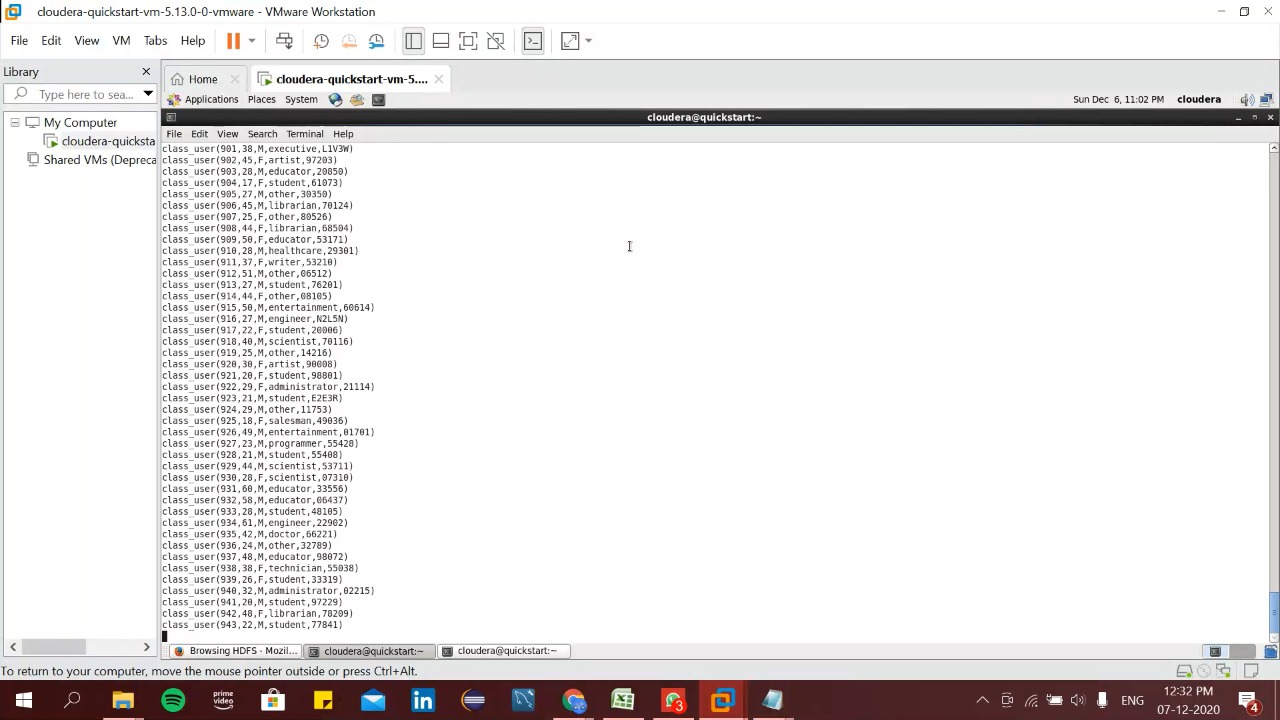


Use the command :

hdfs dfs –mkdir /streaming

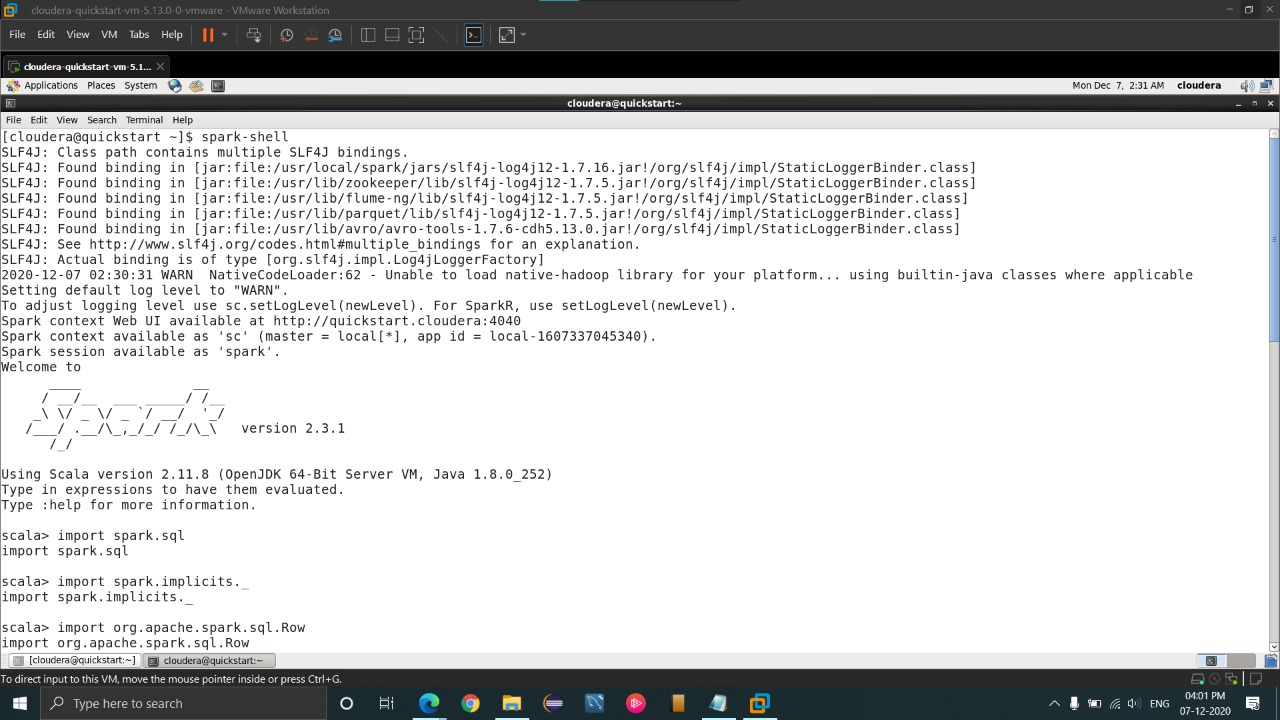
hdfs dfs –put /home/cloudera/Desktop/u1.user /streaming

Output of textfile streaming:



Using csv files and running query using schemas

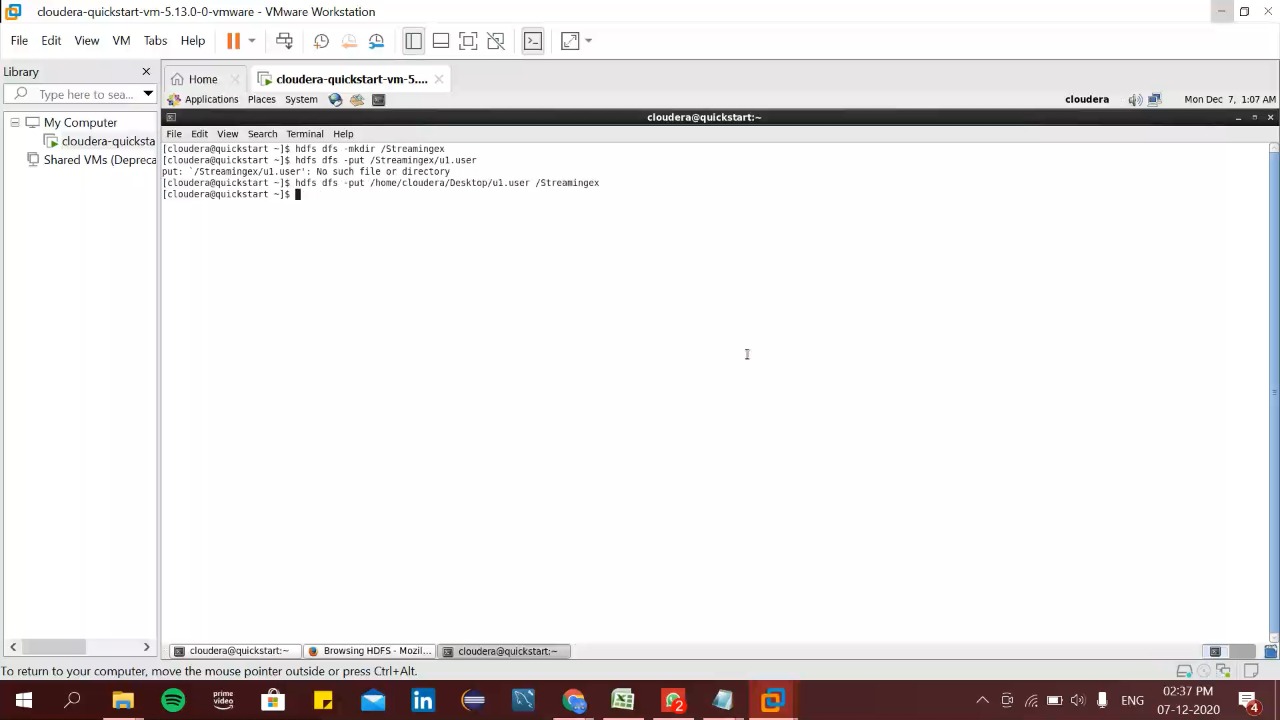
Open the spark shell :



Open the spark shell using the command:

spark-shell

Put the csv file in the hdfs in the streaming directory:

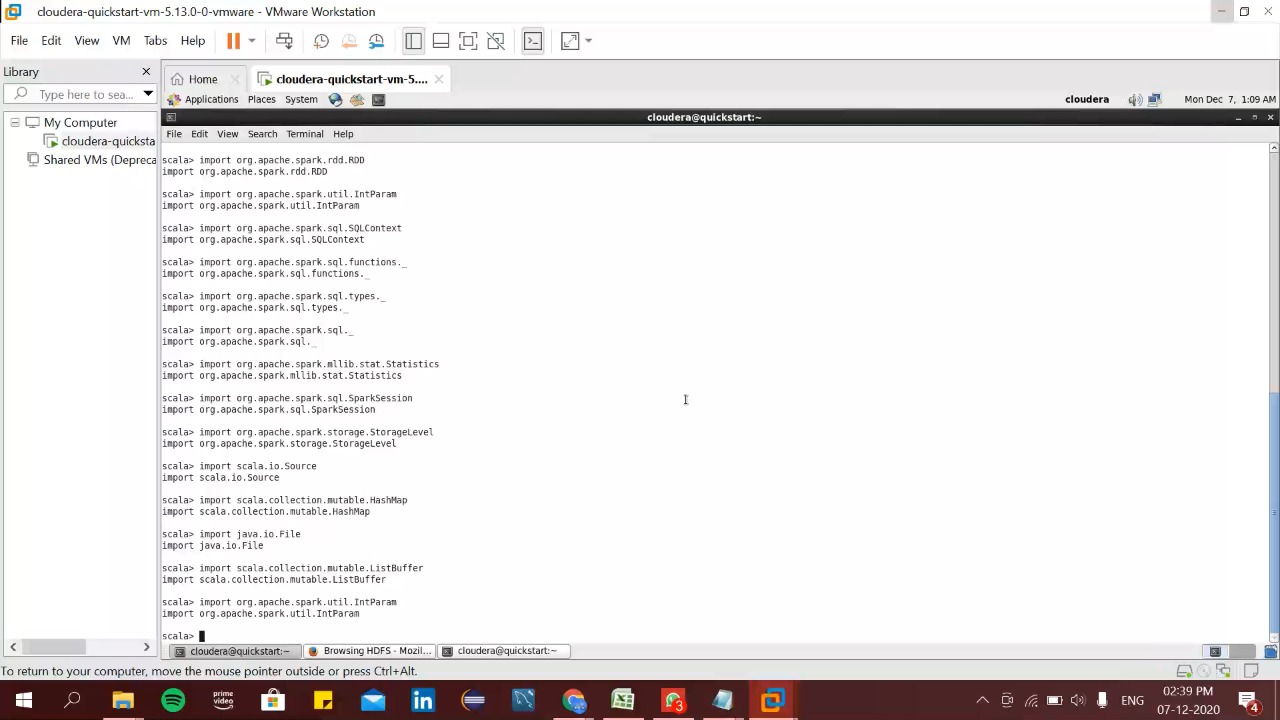


We have to first make a directory in hdfs and upload the required csv file in it using the command:

hdfs dfs –mkdir /Streamingex

hdfs dfs –put /Streamingex/u1.user

Import the necessary packages of the spark streaming:



We need to import all the following packages:

import spark.sql

import spark.implicits.\_

import org.apache.spark.sql.Row

import org.apache.spark.sql.SparkSession

import org.apache.spark.\_

import org.apache.spark.rdd.RDD

import org.apache.spark.util.IntParam

import org.apache.spark.sql.SQLContext

import org.apache.spark.sql.functions.\_

import org.apache.spark.sql.types.\_

import org.apache.spark.sql.\_

import org.apache.spark.mllib.stat.Statistics

import org.apache.spark.sql.SparkSession

import org.apache.spark.storage.StorageLevel

import scala.io.Source

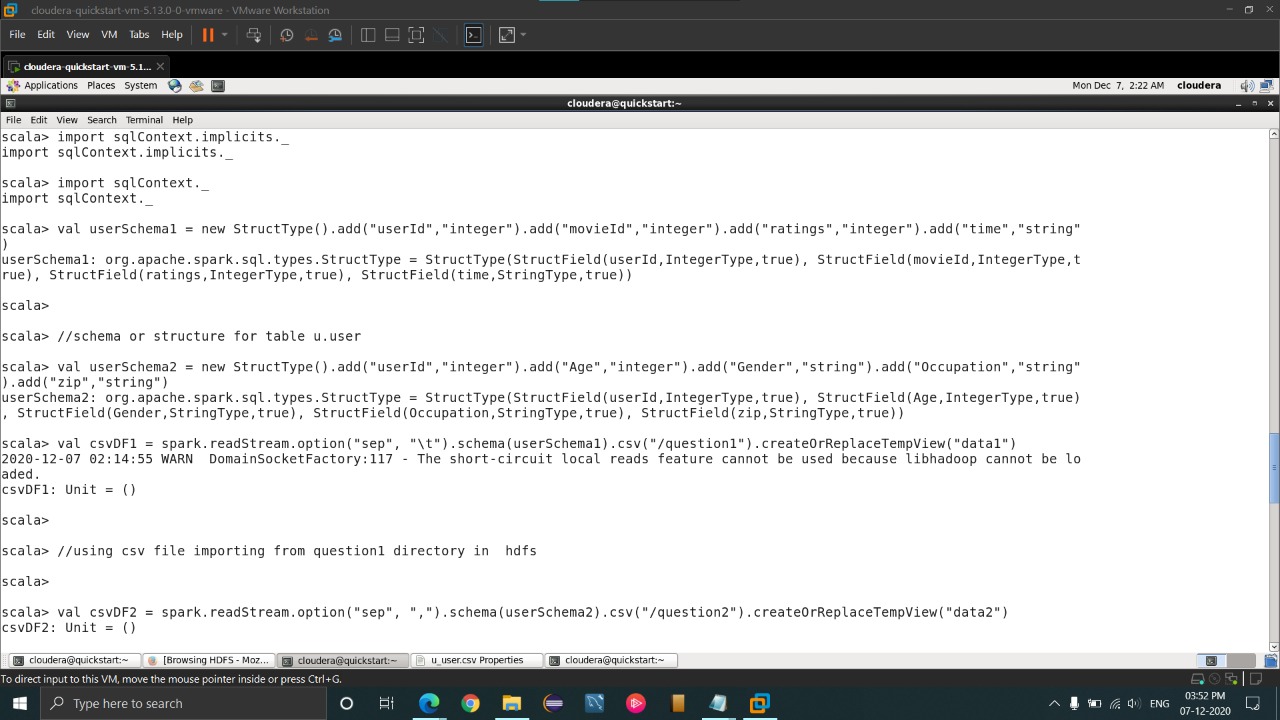
import scala.collection.mutable.HashMap

import java.io.File

import scala.collection.mutable.ListBuffer

import org.apache.spark.util.IntParam

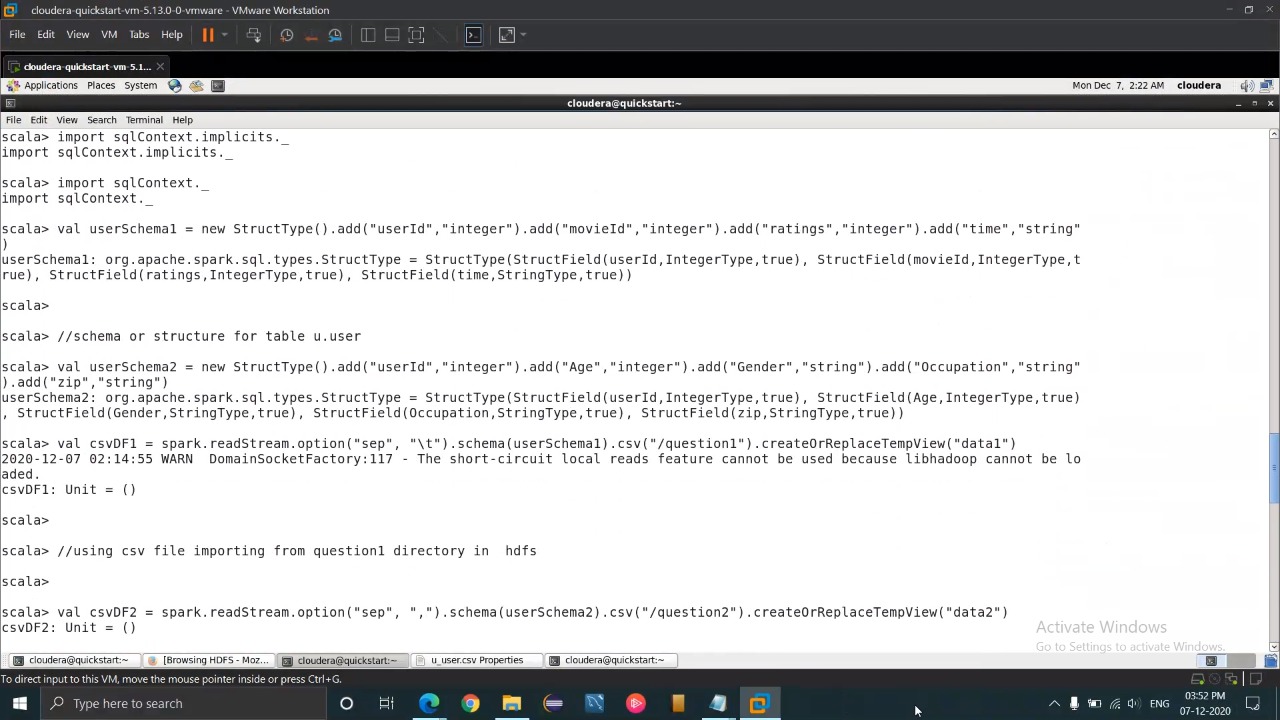
Creating schema and loading the data



Here we create the schema of the u.user using the command:

val userSchema1 = new StructType().add("userId","integer").add("movieId","integer").add("ratings", "integer").add("time","string")

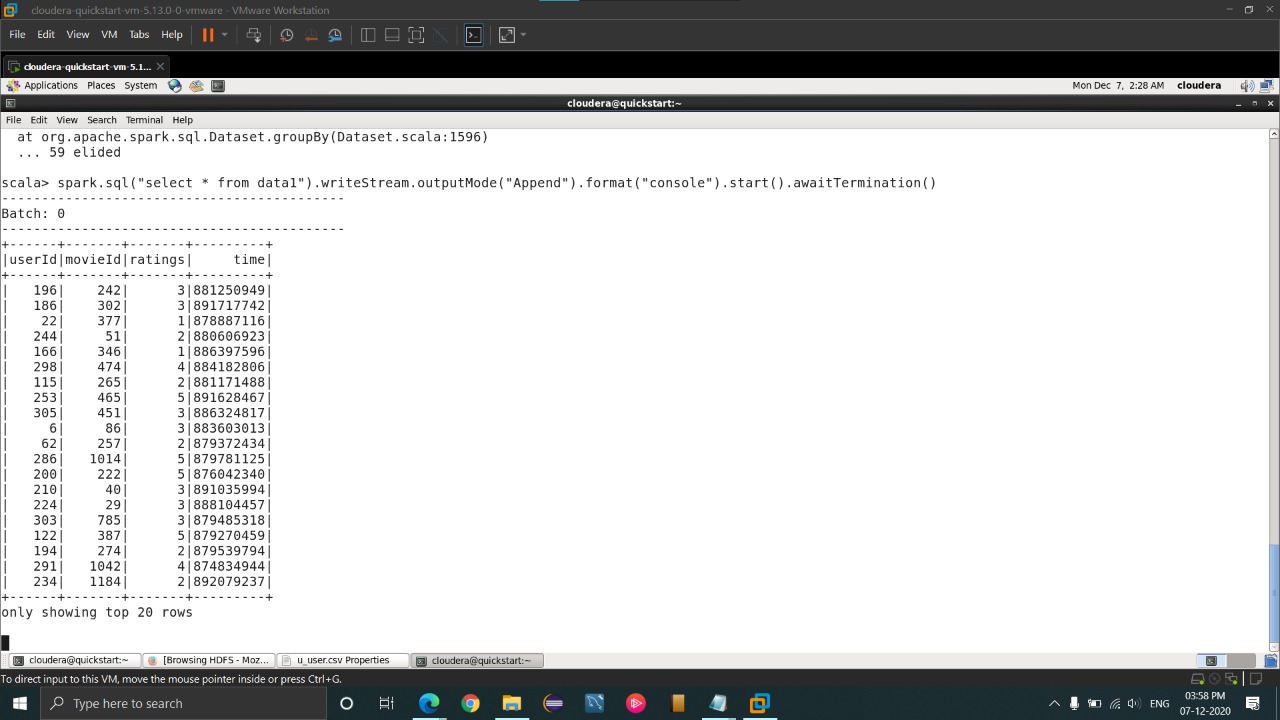
Create the temporary view of the data:



val csvDF2 = spark.readStream.option("sep", ",").schema(userSchema2).csv("/question2").createOrReplaceTempView("d ata2")

Using this command , we create the temporary view .

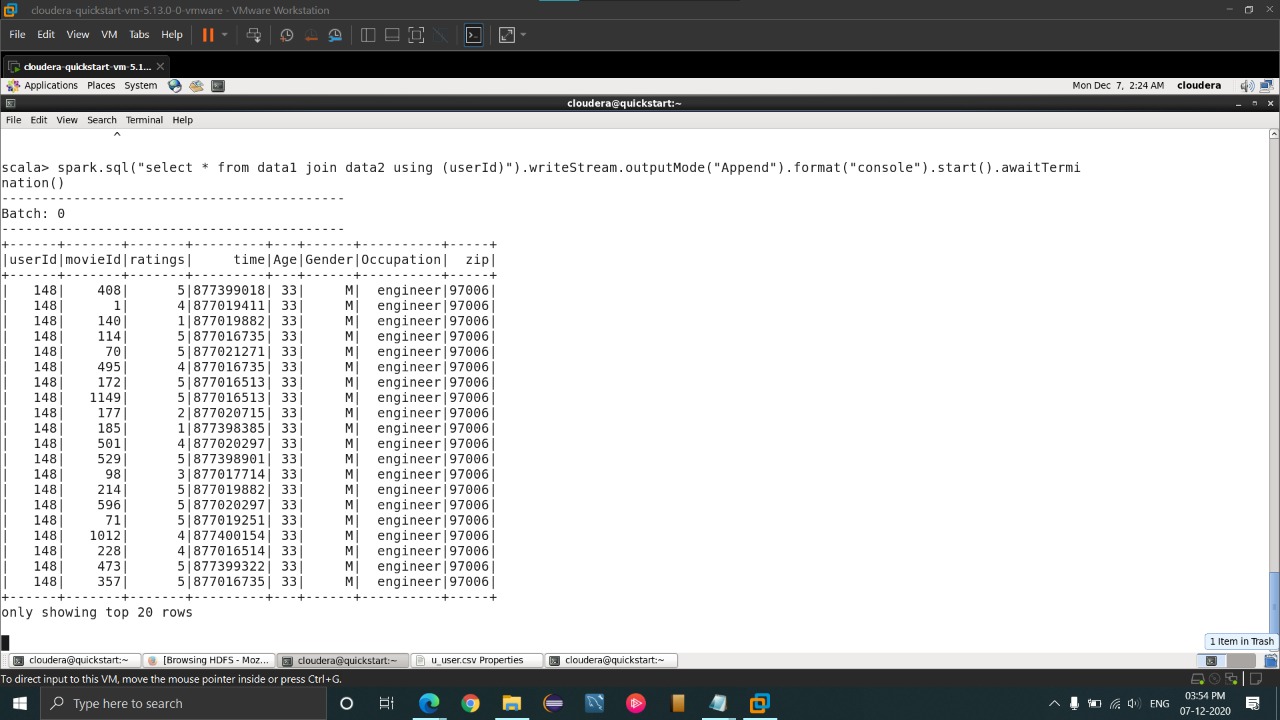
SQL query for the temporary view:



Here we do the sql query to get all the data of u.user using the command:

spark.sql("select \* from data1 ").writeStream.outputMode("Append").format("console").start().awaitTermination()

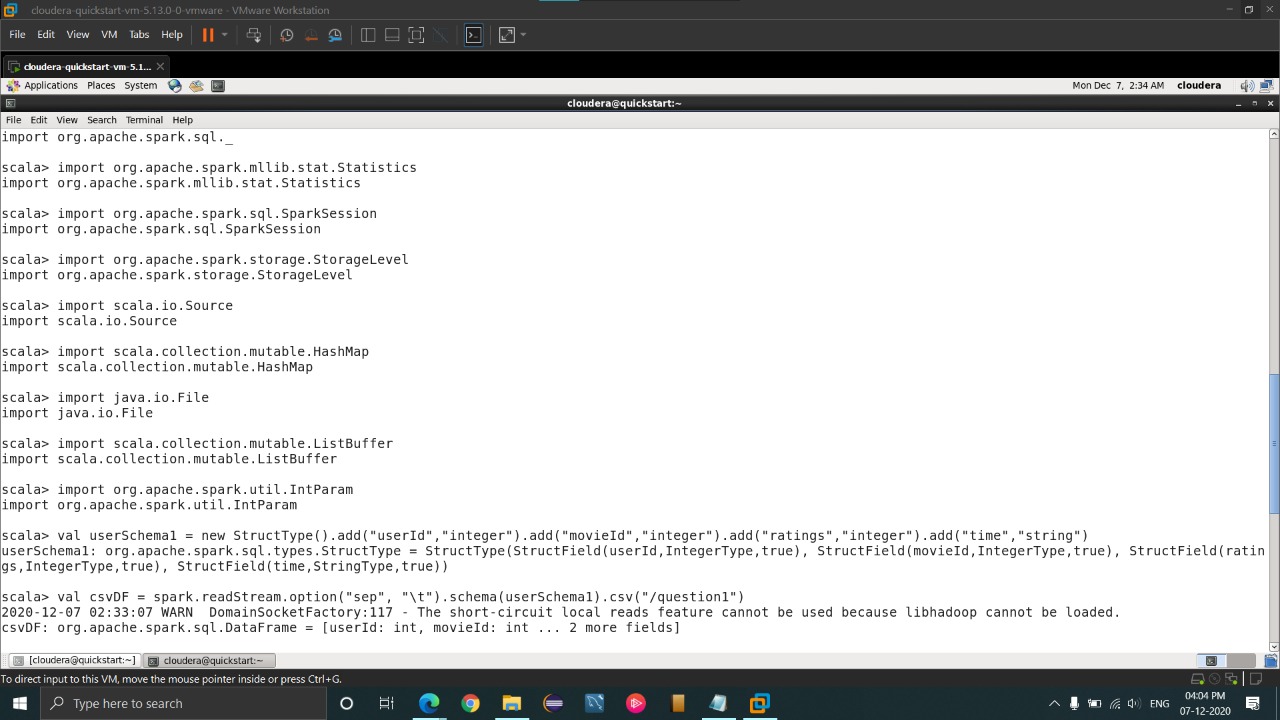
SQL query:



Here we do the sql query to get the join of u.user and u.data based on userId

spark.sql("select \* from data1 join data2 using (userId)").writeStream.outputMode("Append").format("console").start().awaitTerminatio n()

Structured streaming for the dataframe:

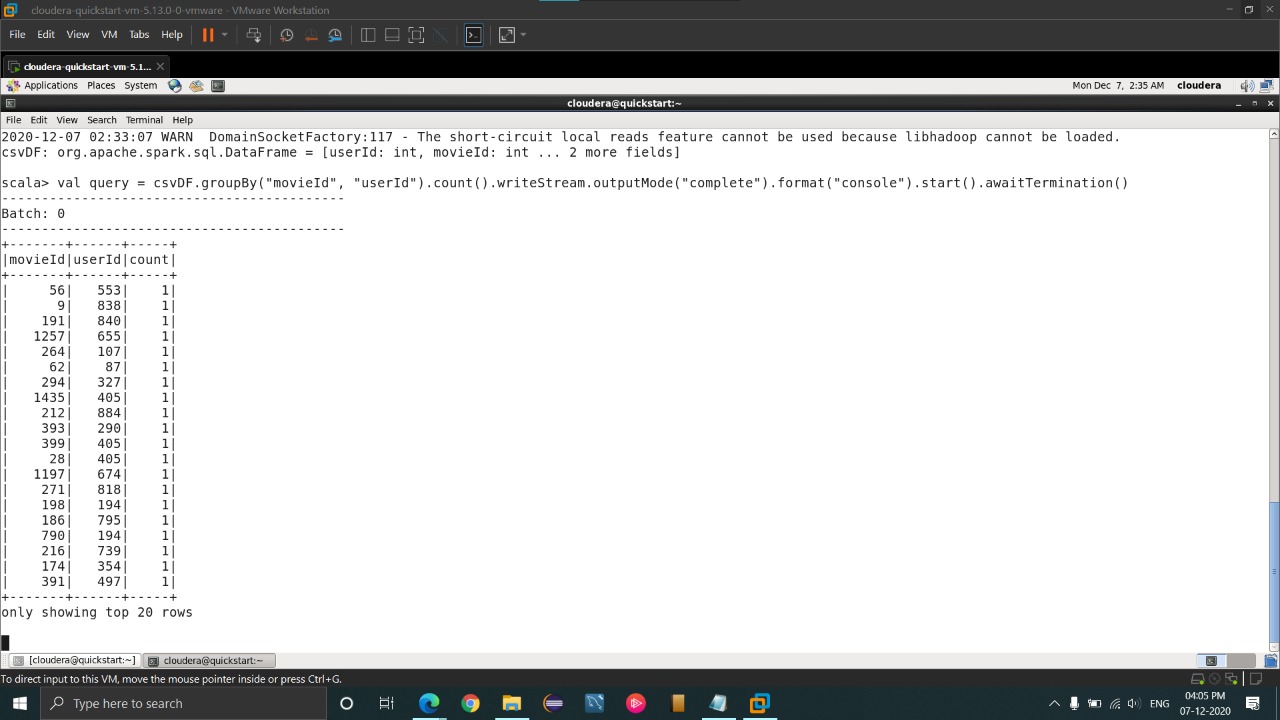


Here we create the dataframe using:

val userSchema1 = new StructType().add("userId","integer").add("movieId","integer").add("ratings","integer").ad d("time","string")

val csvDF = spark.readStream.option(“sep”,”\t”).schema(userSchema1).csv(“/question1”)

Output of the structured streaming for data frame:



Now we can do the query on the dataframe using the command:

val query = csvDF.groupBy("userId").count().writeStream.outputMode("complete").format("console ").start().awaitTermination()

Here the sql query will group both userId and the movieId and count them.