Question 1.

**import** java.io.IOException;

**import** java.util.StringTokenizer;

**import** org.apache.hadoop.conf.Configuration;

**import** org.apache.hadoop.fs.Path;

**import** org.apache.hadoop.io.IntWritable;

**import** org.apache.hadoop.io.LongWritable;

**import** org.apache.hadoop.io.DoubleWritable;

**import** org.apache.hadoop.io.Text;

**import** org.apache.hadoop.mapreduce.Job;

**import** org.apache.hadoop.mapreduce.Mapper;

**import** org.apache.hadoop.mapreduce.Reducer;

**import** org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

**import** org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

**public** **class** HighPrice {

**public** **static** **class** MapClass **extends** Mapper<LongWritable, Text, Text,DoubleWritable>

{

**public** **void** map(LongWritable key,Text value,Context context)

{

**try** {

String[] str = value.toString().split(",");

**double** price = Double.*parseDouble*(str[4]);

context.write(**new** Text(str[1]),**new** DoubleWritable(price));

}

**catch**(Exception e)

{

System.***out***.println(e.getMessage());

}

}

}

**public** **static** **class** ReduceClass **extends** Reducer<Text,DoubleWritable,Text,DoubleWritable> {

**private** DoubleWritable result = **new** DoubleWritable();

**public** **void** reduce(Text key, Iterable<DoubleWritable> values, Context context)**throws** IOException,InterruptedException

{

**double** price = 0;

**for**(DoubleWritable val:values)

{

**if**(val.get()> price)

price = val.get();

}

result.set(price);

context.write(key,result);

}

}

**public** **static** **void** main(String[] args) **throws** Exception {

Configuration conf = **new** Configuration();

Job job = Job.*getInstance*(conf, "HighPrice");

job.setJarByClass(HighPrice.**class**);

job.setMapperClass(MapClass.**class**);

job.setReducerClass(ReduceClass.**class**);

job.setNumReduceTasks(1);

job.setOutputKeyClass(Text.**class**);

job.setOutputValueClass(DoubleWritable.**class**);

FileInputFormat.*addInputPath*(job, **new** Path(args[0]));

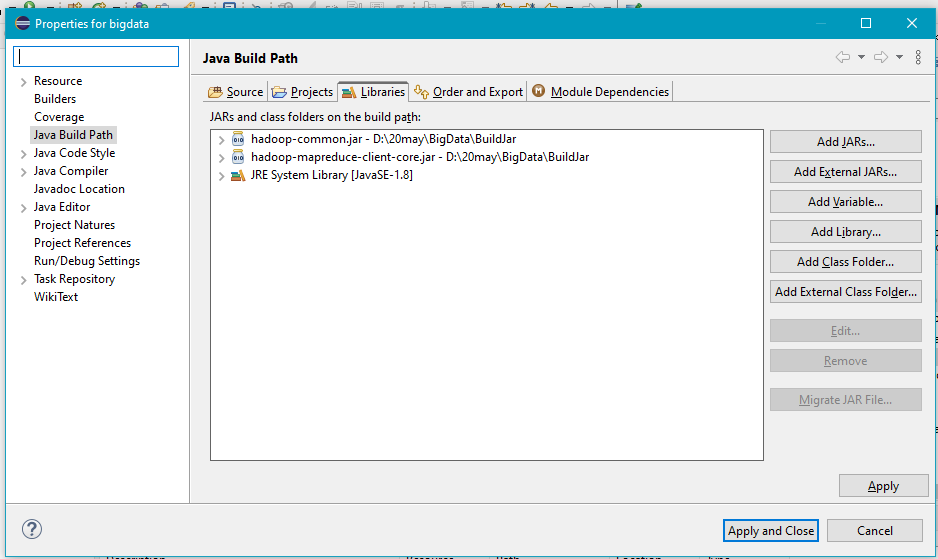
FileOutputFormat.*setOutputPath*(job, **new** Path(args[1]));

System.*exit*(job.waitForCompletion(**true**) ? 0 : 1);

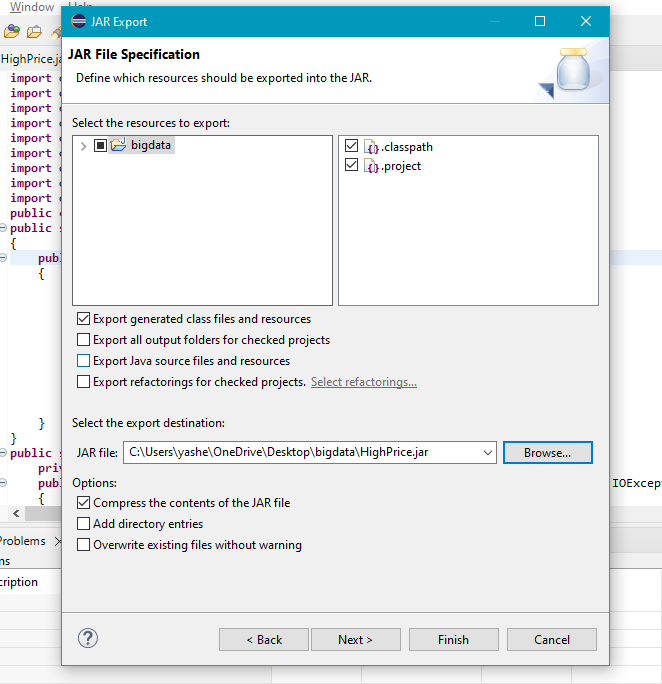
}

}

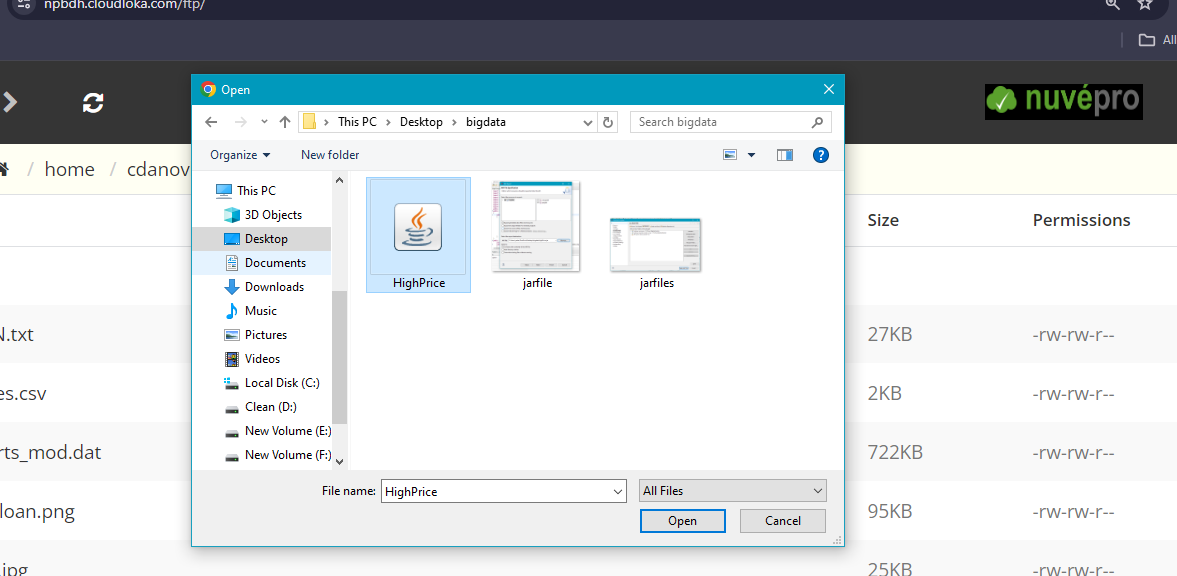
Add jar files



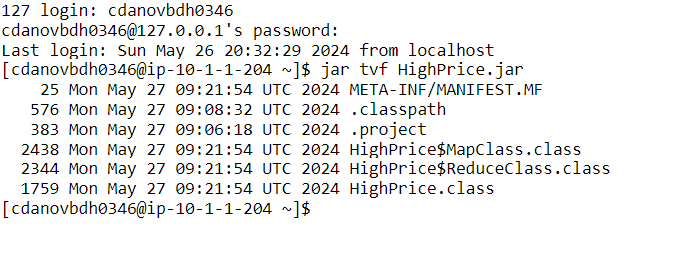
Export the java file as jar file



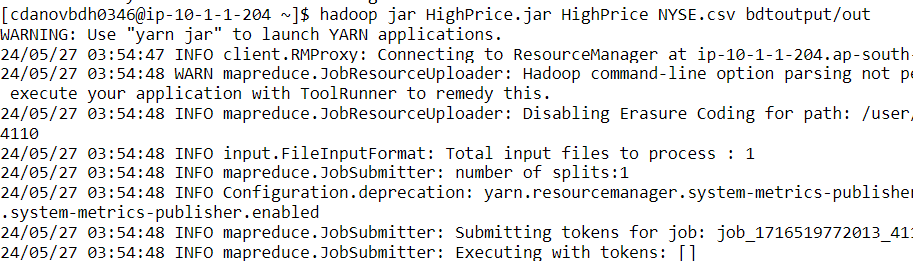
Upload the created jar file on ftp



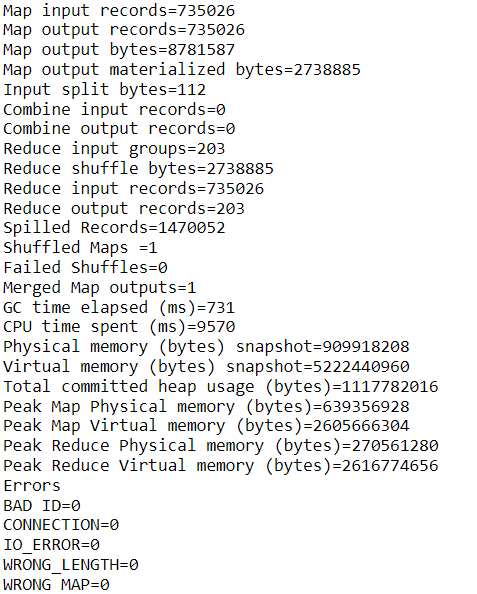
Check the jar file on web shell



Run the jar file using Hadoop command



Message after running the jar file



Output file on the Hadoop hdfs



Question 2

1.

select state,sum(amount) from txnrecords group by state limit 10;

2.

select city,avg(amount) from txnrecords group by city limit 10;

3.

select product, sum(amount) from txnrecords group by products where month(txndate) = 2;

4.

select count(custno),city from txnrecords group by city where year(txnrecords) = 2011;

Question 3

1.

baseRDD= sc.textFile("airlines.csv")

header=baseRDD.first()

dataRDD=baseRDD.filter(lambda a:a!=header)

kvRDD=dataRDD.map(lambda a:(a.split(',')[0],float(a.split(',')[2])\*float(a.split(',')[3])))

reduceRDD=kvRDD.reduceByKey(lambda a,b:a+b)

for line in kvRDD.take(10):

print(line)

2.

baseRDD= sc.textFile("airlines.csv")

header=baseRDD.first()

dataRDD=baseRDD.filter(lambda a:a!=header)

kvRDD=dataRDD.map(lambda a:(a.split(',')[0]+','+a.split(',')[1],float(a.split(',')[2])\*float(a.split(',')[3])))

filterRDD=kvRDD.filter(lambda a:a[0]=="2000,1" or a[0]=="2000,2" or a[0]=="2000,3" or a[0]=="2000,4")

sortbyRDD=filterRDD.sortBy(lambda a:-a[1])

for line in kvRDD.take(1):

print(line)

3.

baseRDD= sc.textFile("airlines.csv")

header=baseRDD.first()

dataRDD=baseRDD.filter(lambda a:a!=header)

kvRDD=dataRDD.map(lambda a:(a.split(',')[0]+','+a.split(',')[1],float(a.split(',')[3])))

sortbyRDD=filterRDD.sortBy(lambda a:-a[1])

for line in kvRDD.take(1):

print(line)