**Q1**.

MapReduce

List the files in local machine

[bigdatamind43825@ip-10-1-1-204 ~]$ ls

airlines.csv data3.txt **mysql-connector-java-5.1.47-bin.jar** spark415.py

airports\_mod.dat file3 name spark41.py

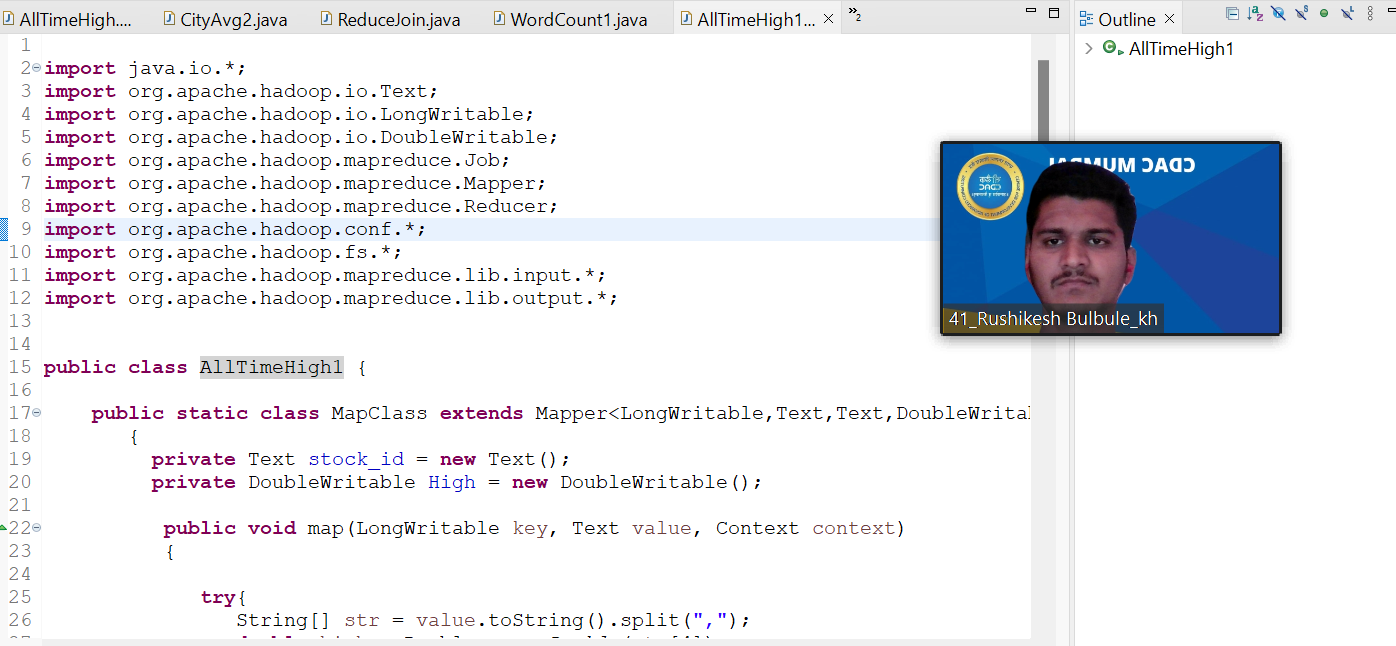
counter.txt file4 NYSE\_2.csv **spark-streaming-kafka-0-8-assembly\_2.11-2.4.8.jar**

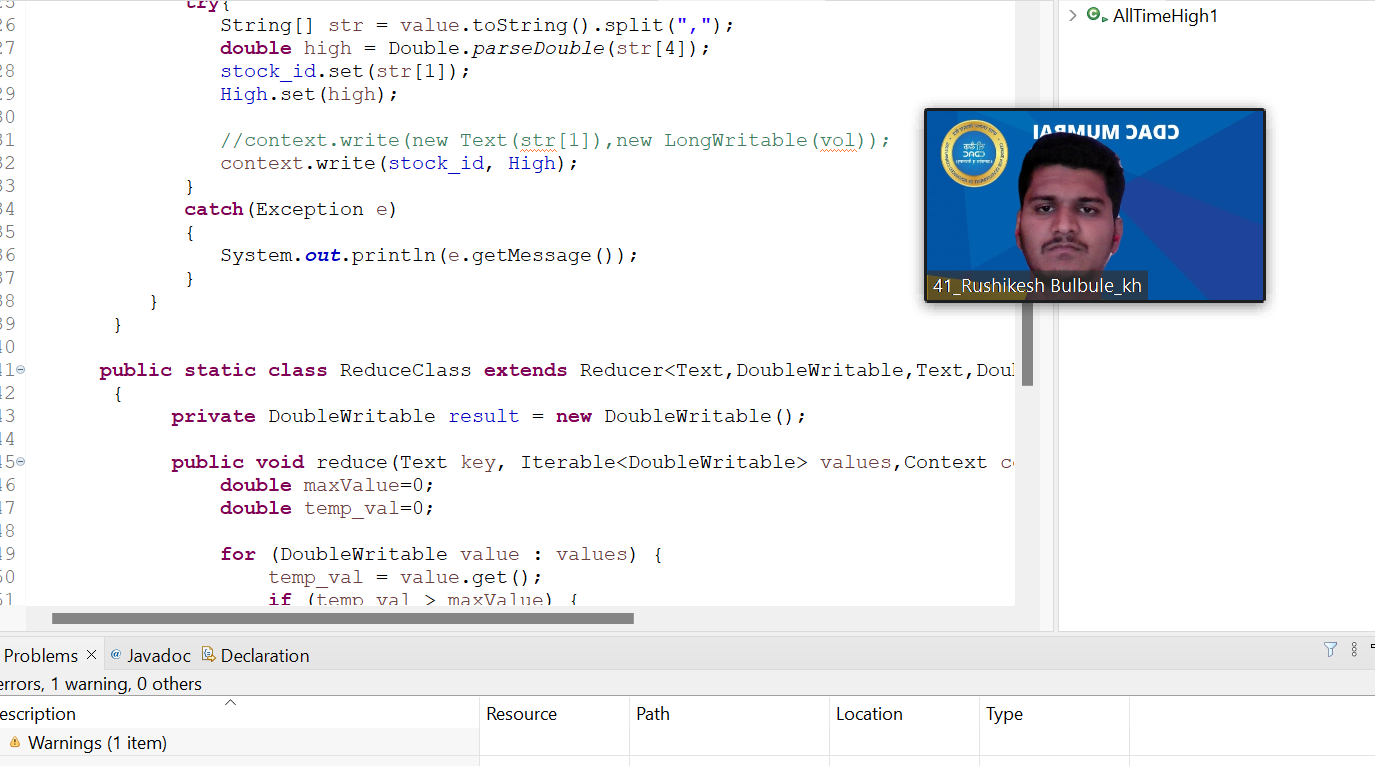
custs.txt Final\_airlines NYSE.csv Speed-data.txt

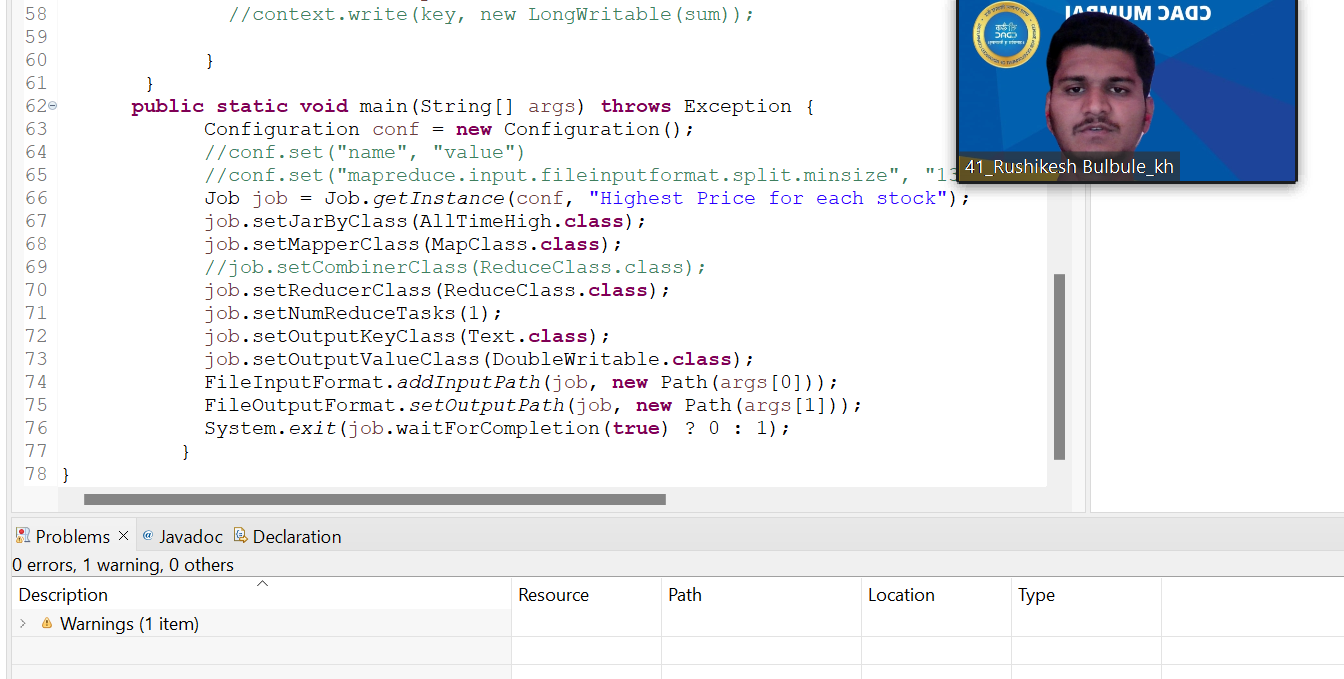
D01 header\_data.txt nyse table command.txt student1

-to upload file on Hadoop cluster

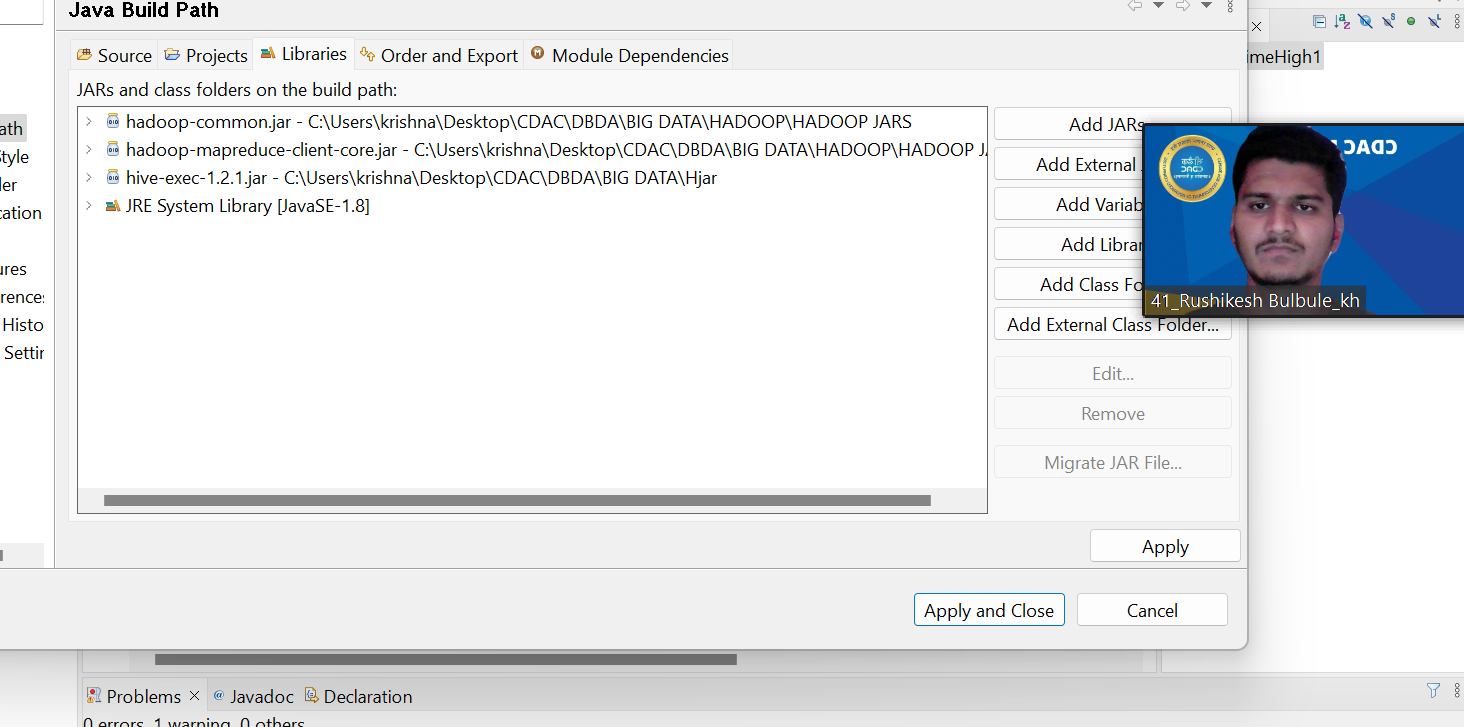
[bigdatamind43825@ip-10-1-1-204 ~]$ hadoop fs -put NYSE.csv cdac



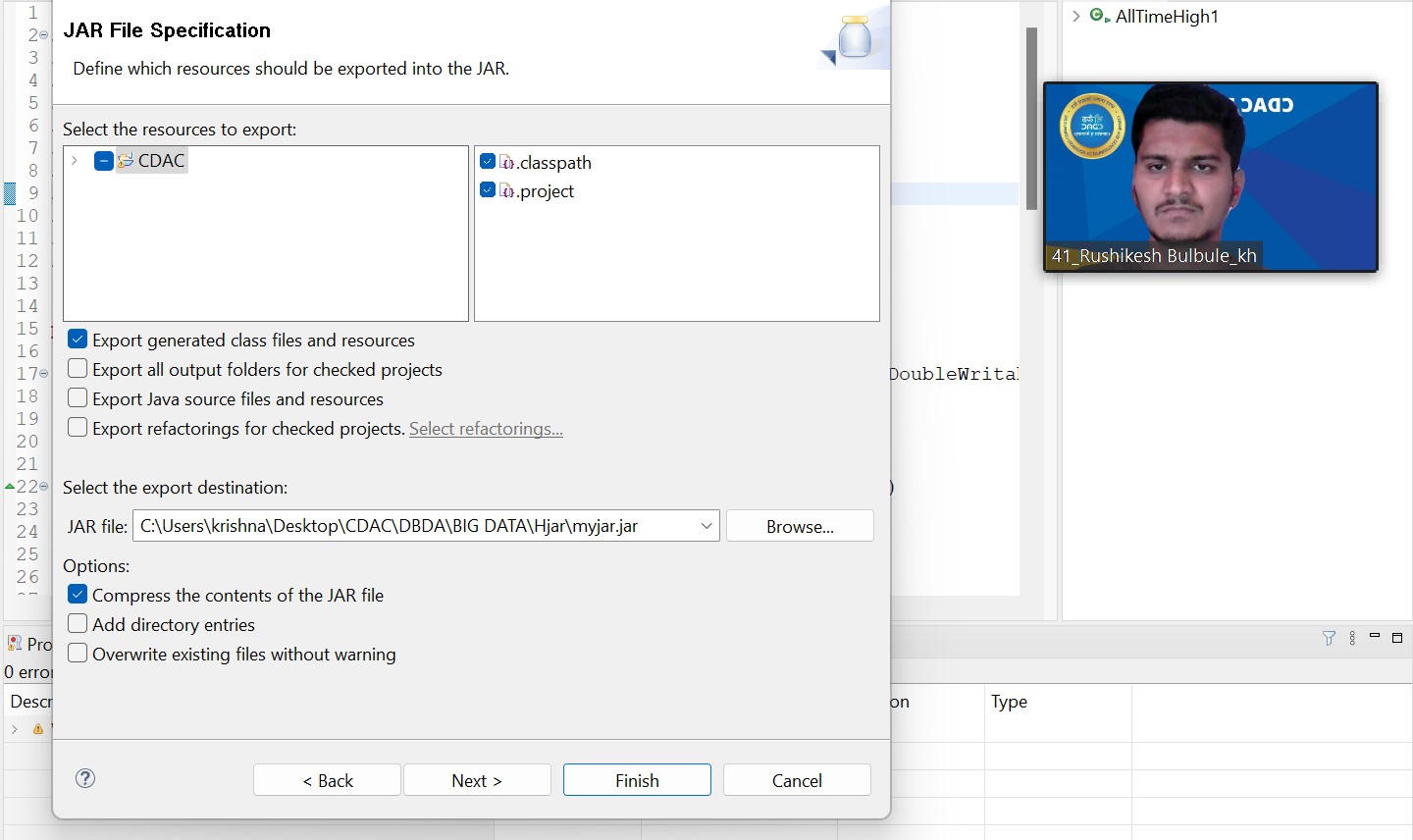


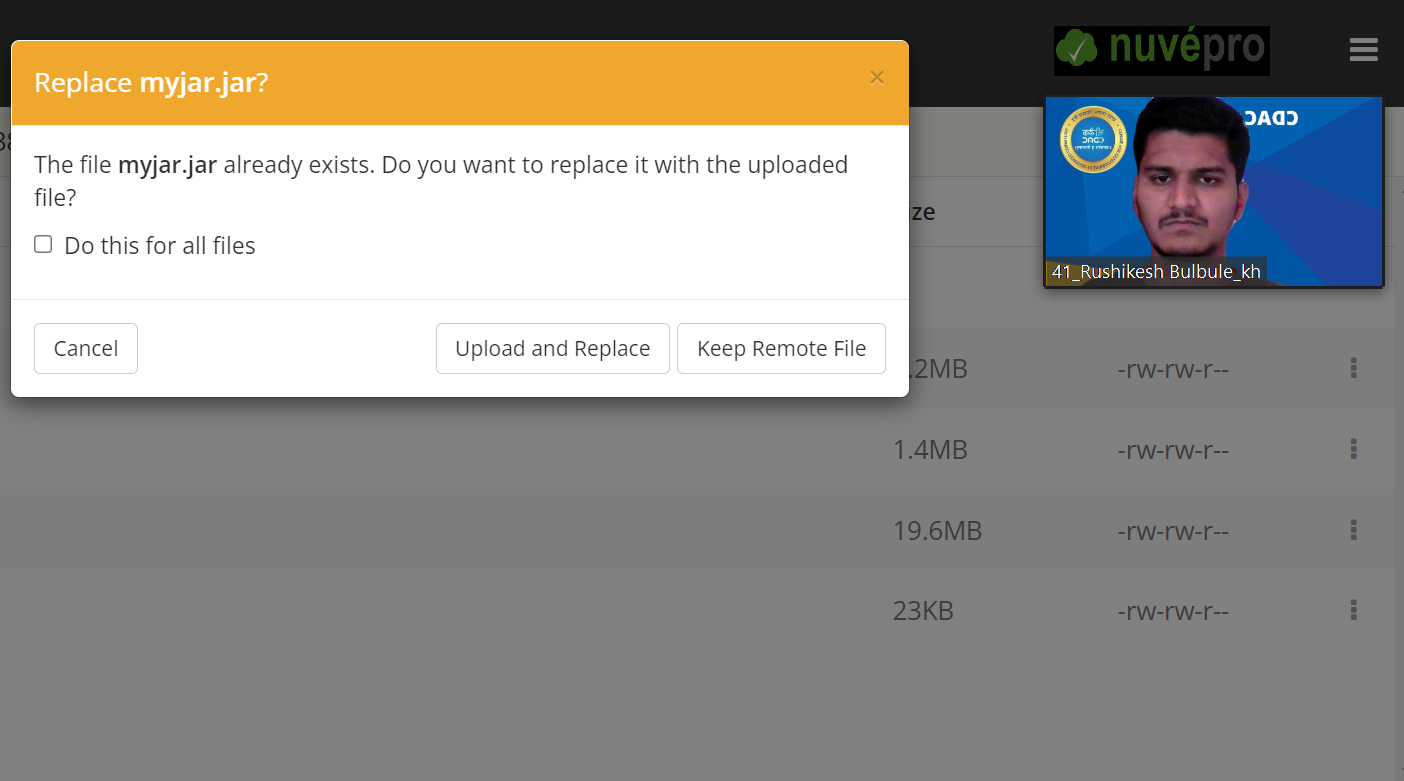


To upload external jars 🡪



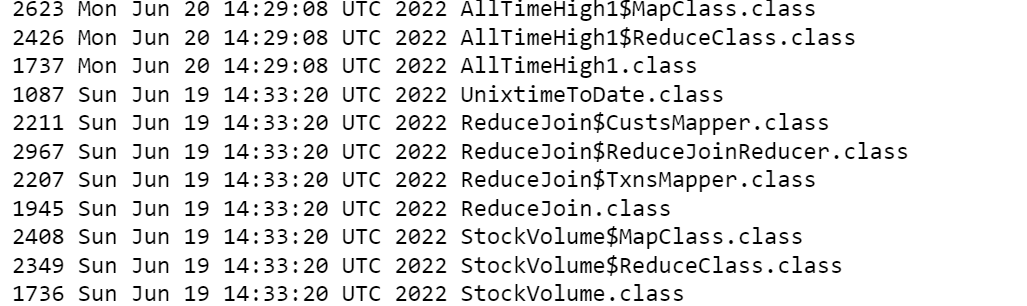
Create jar files files





To display the class file

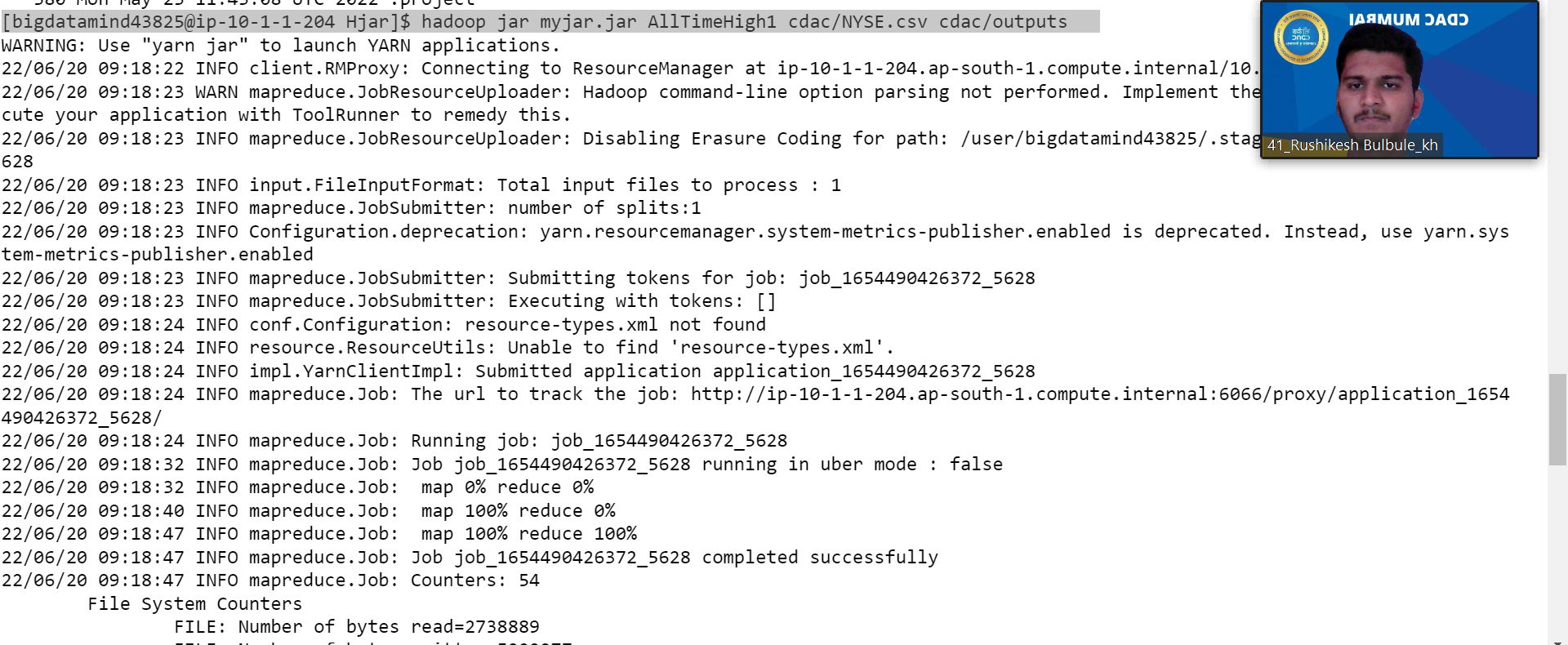
[bigdatamind43825@ip-10-1-1-204 Hjar]$ jar tvf myjar.jar



**Q2.**

Command to find all time high price for each stock--

[bigdatamind43825@ip-10-1-1-204 Hjar]$ hadoop jar myjar.jar AllTimeHigh1 cdac/NYSE.csv cdac/outputs







**Q3.hive**

hive> use training41;

OK

Time taken: 1.496 seconds

hive> show tables;

OK

nyse

stkvol

Time taken: 0.195 seconds, Fetched: 2 row(s)

hive> create table customer(custno int,firstname string,lastname string,age int,profession string

> )row format delimited

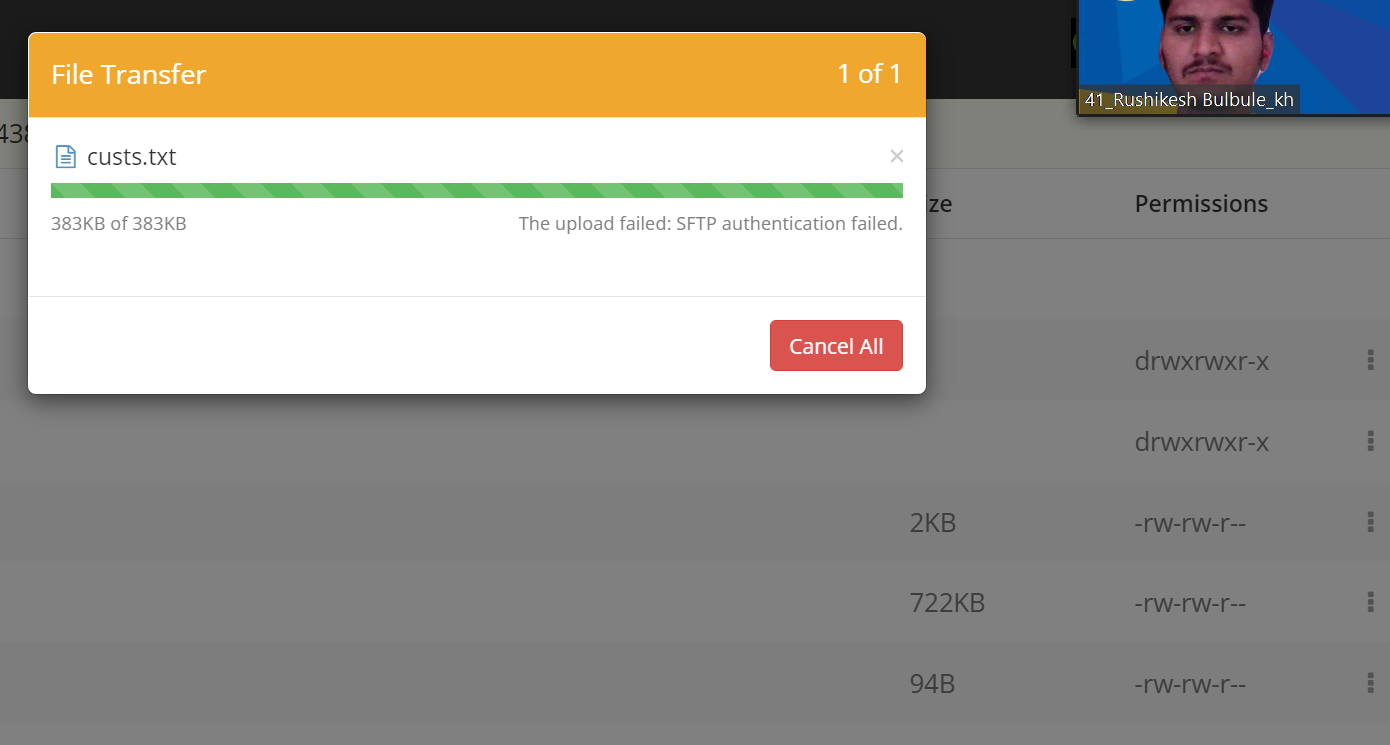
> fields terminated by ','

> stored as textfile;

OK

Time taken: 0.476 seconds

**Upload file on local machine**

****

hive> load data local inpath 'custs.txt' into table customer;

Loading data to table training41.customer

OK

Time taken: 1.118 seconds

hive> describe customer;

OK

custno int

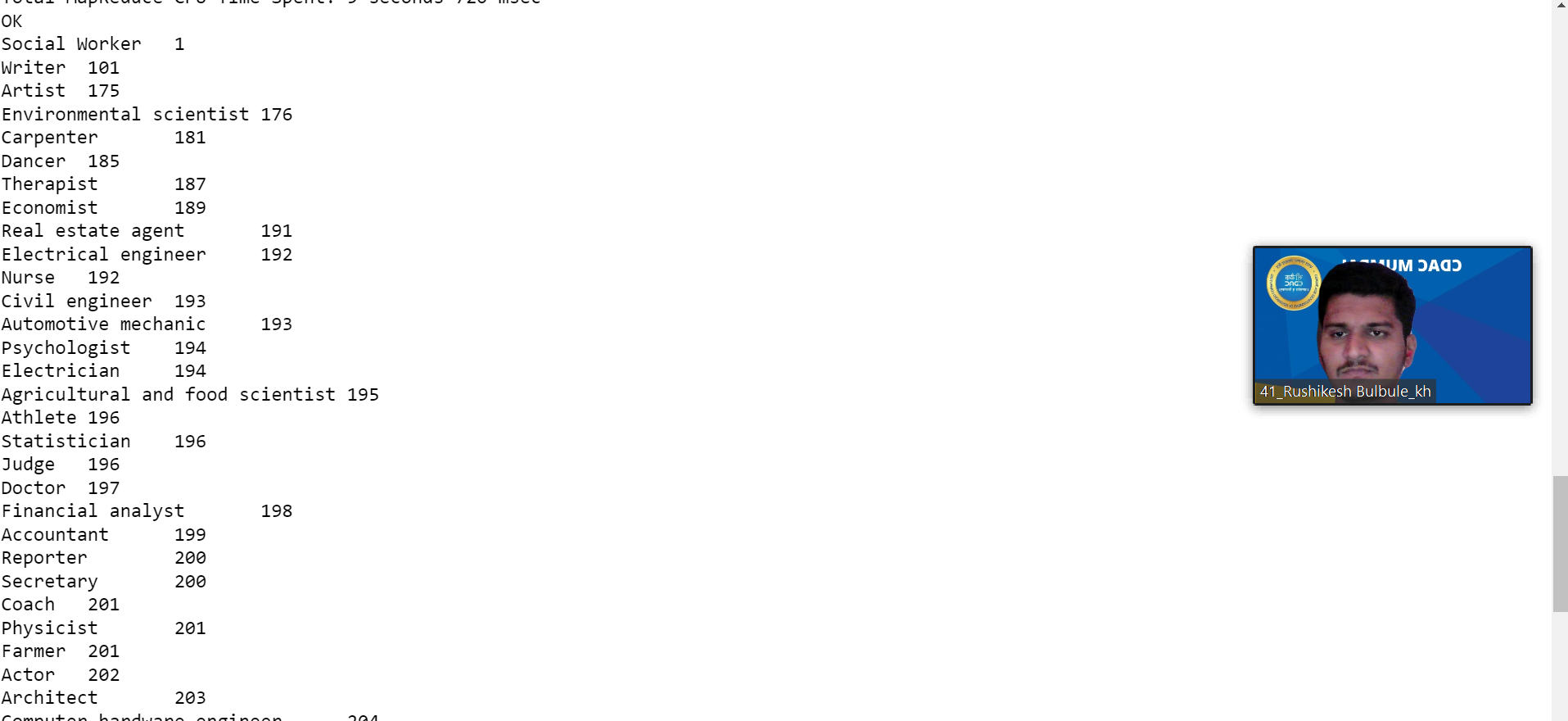
firstname string

lastname string

age int

profession string

hive> select profession, count(\*) as headcount from customer group by profession order by headeecount;





**Q2.HIVE**

hive> create table txnrecords(txnno int,txndate string,custno int,amount double,category string,product string,city string,state string,spen

dby string)

> row format delimited

> fields terminated by ','

> stored as textfile;

OK

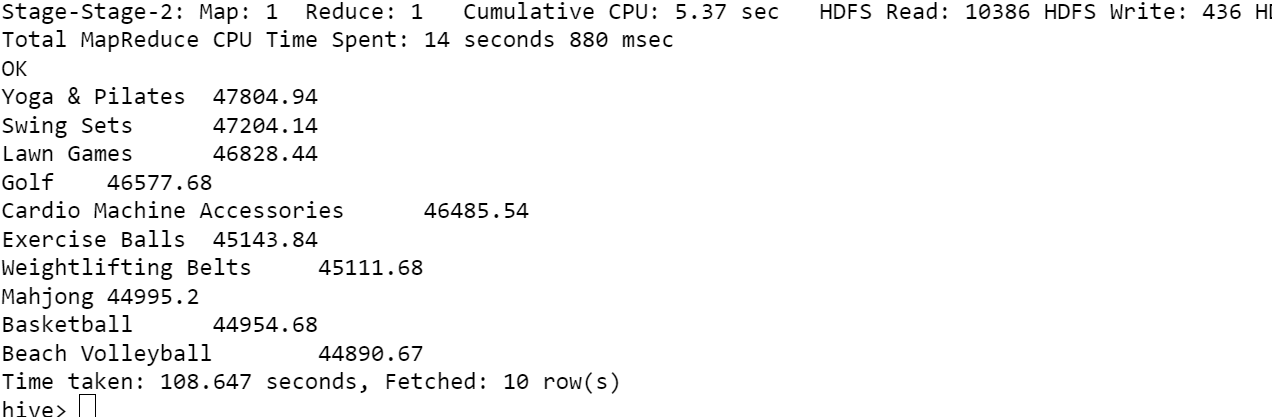
Time taken: 0.095 seconds

hive> load data local inpath 'txns1.txt' overwrite into table txnrecords;

Loading data to table training41.txnrecords

OK

hive> select product,round(sum(amount),2) as amt from txnrecords group by product order by amt desc limit 10;



**Q3. HIVE**

hive> create table txnrecordsbycat(txnno int,txndate string,custno int,amount double,product string,city string,state string,spendby string)

partitioned by (category string)

> row format delimited

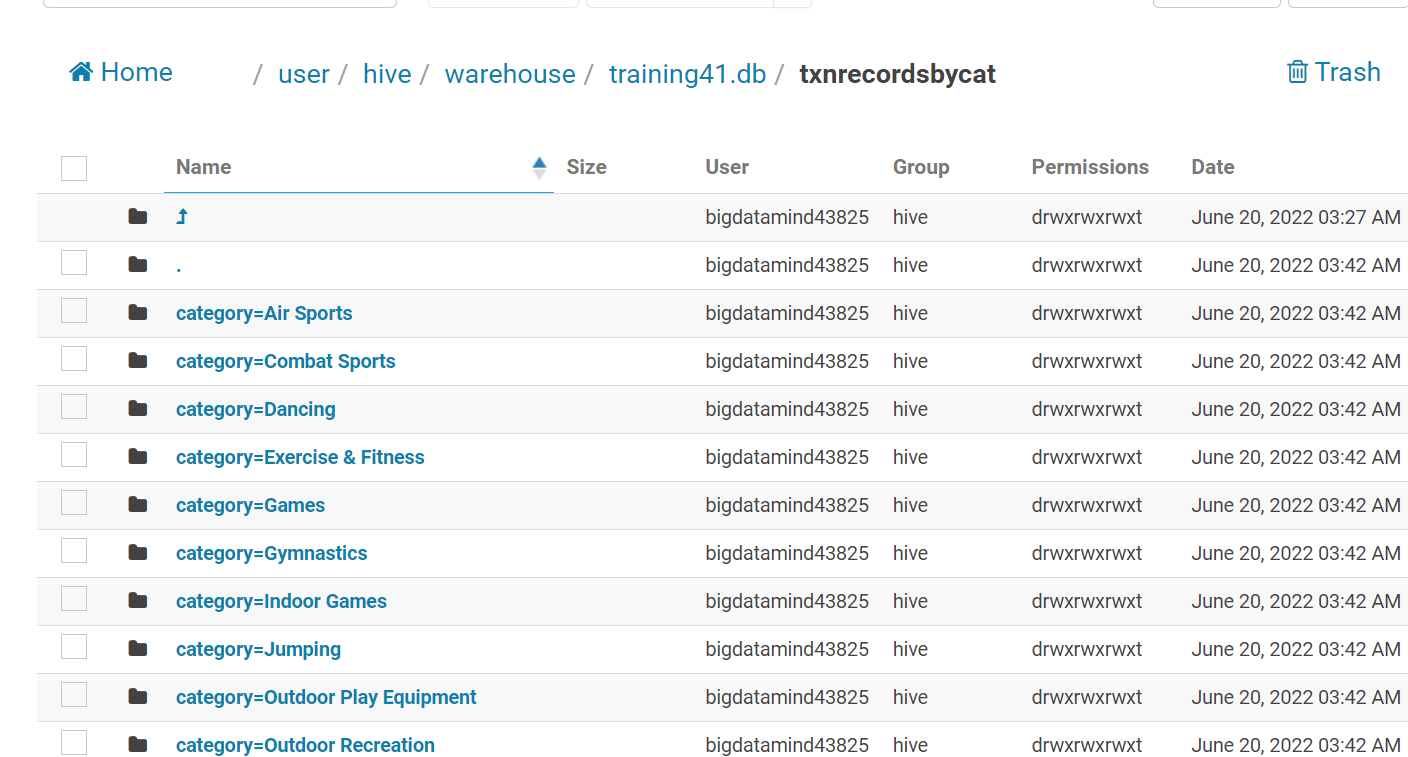
> fields terminated by ','

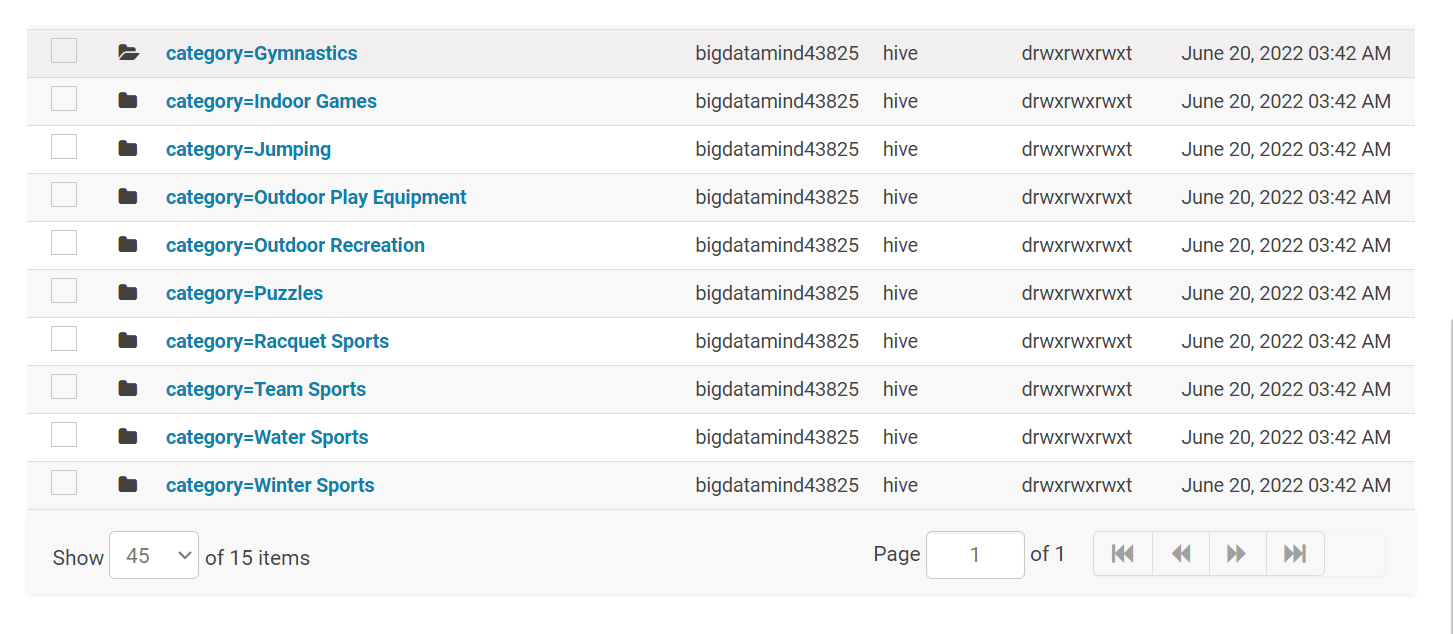
> stored as textfile;

OK

hive> INSERT OVERWRITE TABLE txnrecordsbycat PARTITION(category) select txn.txnno, txn.txndate,txn.custno, txn.amount,txn.product,txn.city,t

xn.state, txn.spendby, txn.category from txnrecords txn DISTRIBUTE By category;

****

****

**PYSPARK**

**Load the data airlines.csv**

**airlineRDD = sc.textFile("/user/bigdatamind43825/airlines.csv")**

**>>> airlineRDD.count()**

**85**

**>>> for i in airlineRDD.take(1):**

**print(i)**

**output-**

**Year,Quarter,Average revenue per seat,total no. of booked seats**

**TO REMOVE THE UNICODE**

**airlineRDD1 = airlineRDD.map(lambda a: a.encode("ascii","ignore"))**

**TO STORE THE header Line into variable**

**header = airlineRDD1.first()**

**>>> print(type(header))**

**<type 'str'>**

**to FILTER the HEADER ( first line ) from the RDD**

**airlineRDD2=airlineRDD1.filter(lambda a:a!=header)**

**>>> for i in airlineRDD2.collect():**

**print(i)**

**it will show all the lines without header string**

**In Spark, the First function always returns the FIRST LINE of the RDD TO THE driver node...**

**[ first element of the dataset ]. It is similar to take(1)**

**to split the RDD to get required KEY VALUE pair for processing.**

**airlineRDD3 = airlineRDD2.map(lambda a: a.split(','))**

**for i in airlineRDD3.take(5):**

**print(i)**

**Output :**

**['1995', '1', '296.9', '46561']**

**['1995', '2', '296.8', '37443']**

**['1995', '3', '287.51', '34128']**

**['1995', '4', '287.78', '30388']**

**To take KEY VALUE pair from the RDD as it already converted into ARRAY through SPLIT FUNCTION**

**airlineRDD4 = airlineRDD3.map(lambda a: (a[0], float(a[2])\*int(a[3])))**

**airlineRDD4 = airlineRDD3.map(lambda a : (a[0],float(a[2])\* float(a[3])))**

**addtotal = airlineRDD4.reduceByKey(lambda a,b:(a+b))**

**sortbyval = addtotal.sortBy(lambda a:-a[1])**

**>>> for i in sortbyval.take(10):**

**... print(i)**