NAME : DARSHAN AHIRE

CDAC MUMBAI PG-DBDA MARCH 2022 BATCH KHARGHAR MODULE: BIG DATA ANALYTICS DATE : 20TH JUNE 2022 MARKS : 40 MARKS ---------------------------------------------------------------------------------------------------------------------------

[10 marks]

Q1. MapReduce Problem Statement Here, we have chosen the stock market dataset on which we have performed map-reduce operations. Following is the structure of the data. Kindlyfind the solutions to the questions below.

Data Structure 1. Exchange Name

2 Stock symbol

3. Transaction date

4. Opening price of the stock

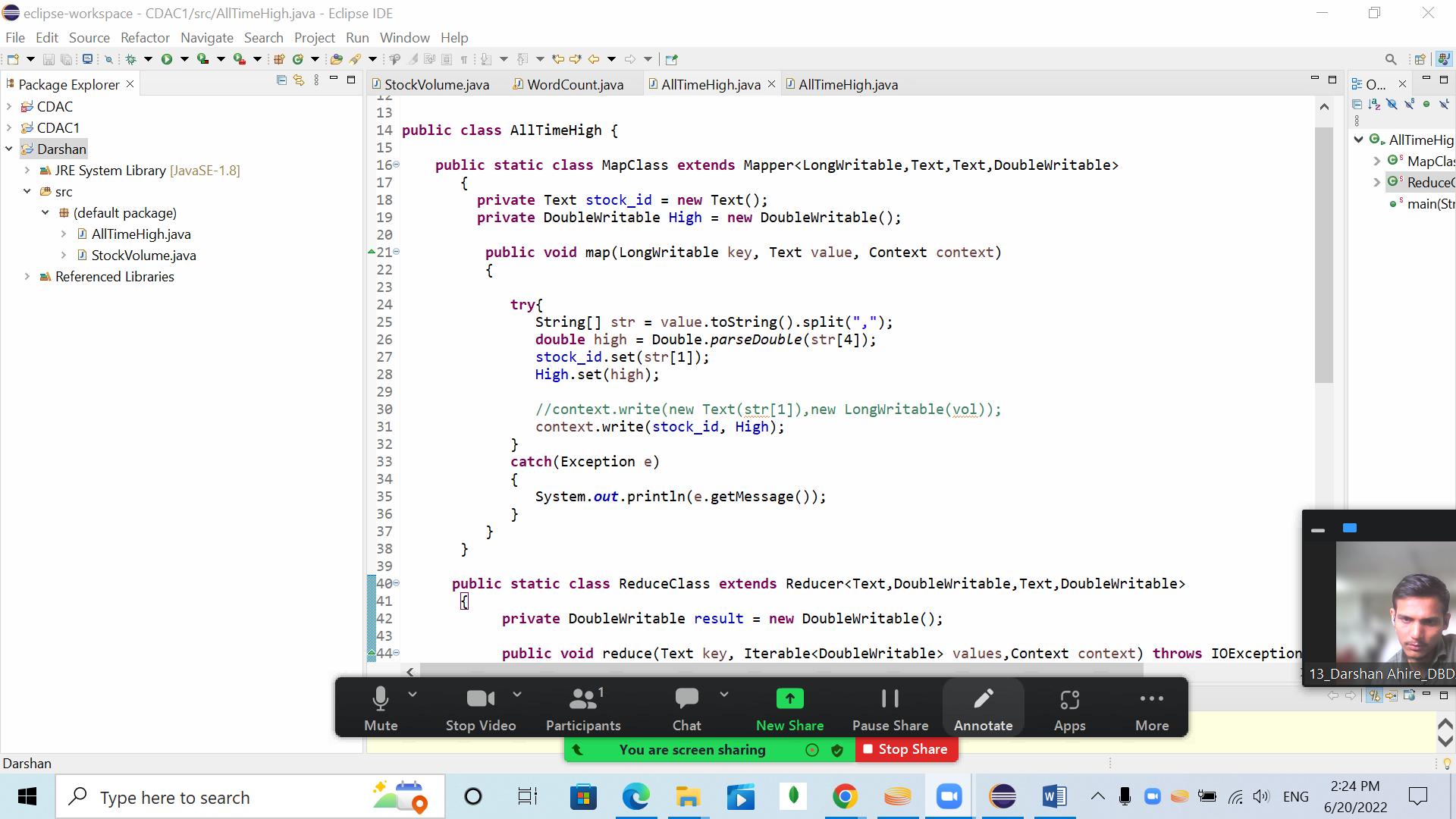
5. Intra day high price of the stock

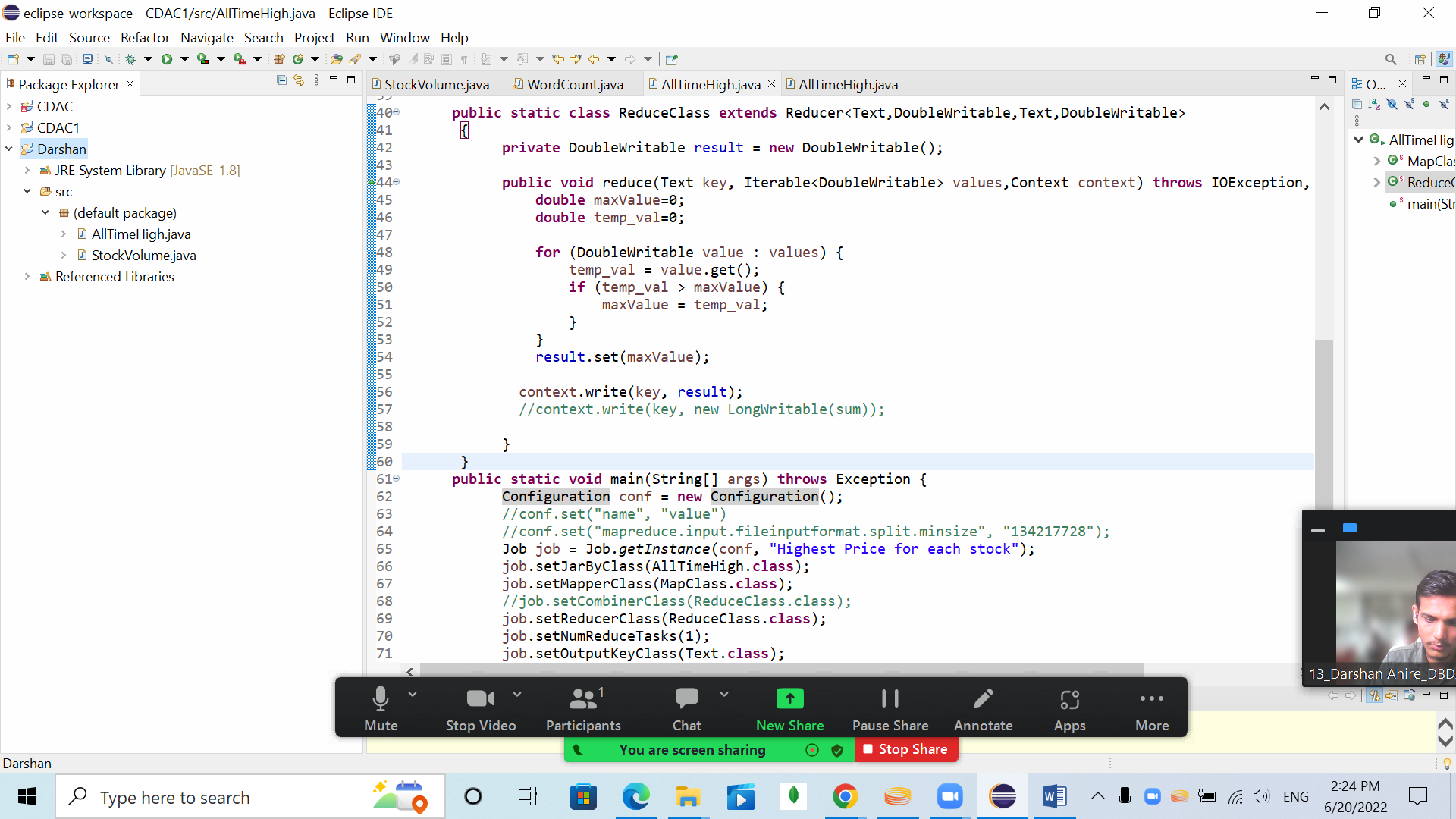
6. Intra day low price of the stock

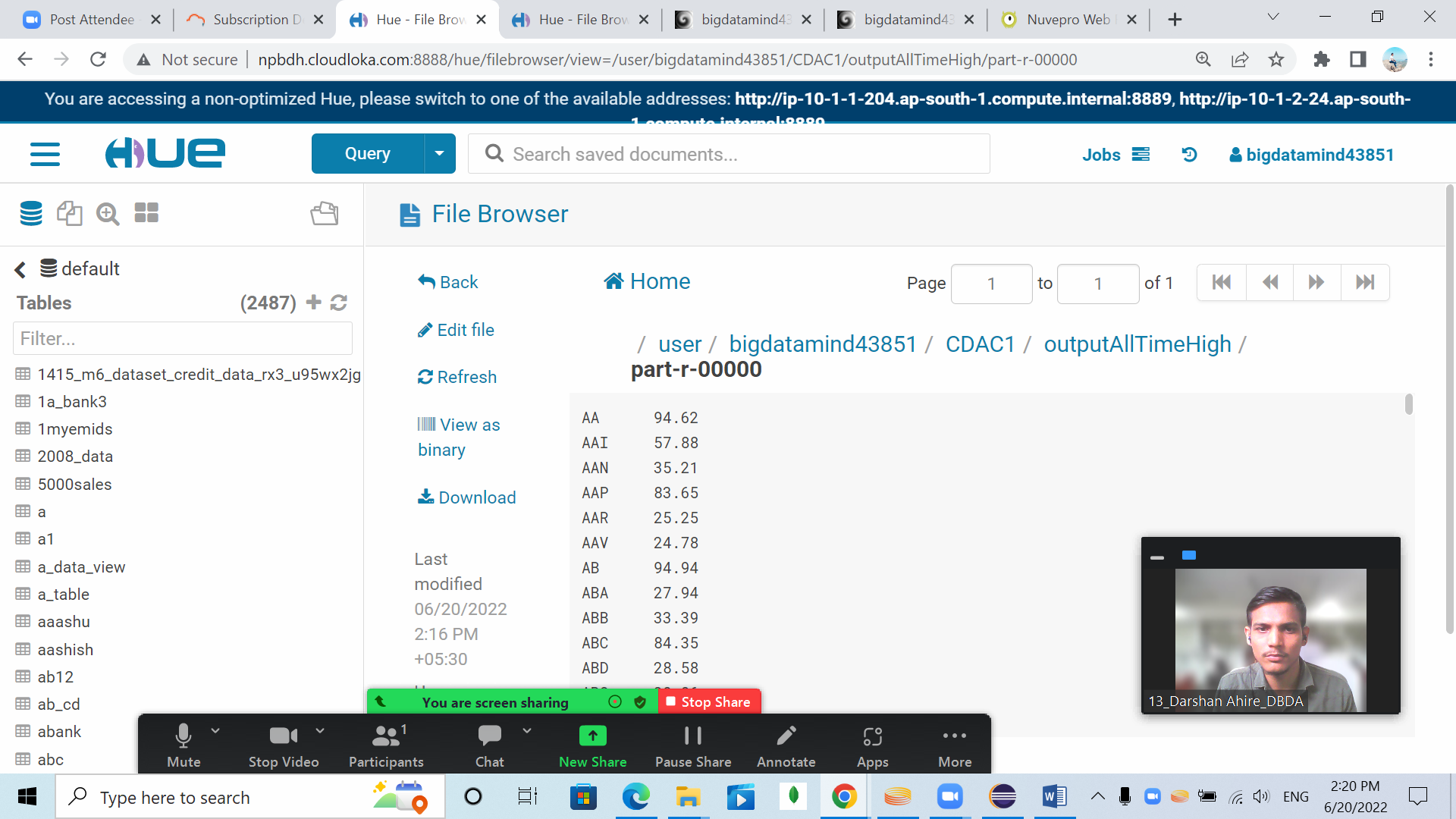
7. Closing price of the stock

8. Total Volume of the stock on the particular day

9. Adjustment Closing price of the stock Field Separator – comma







***Hive***

**Please find the customer data set. cust id firstname lastname age profession**

*ANS:*

hive> use hivedarshan;

hive> set hive.cli.print.current.db = true;

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

row format delimited fields terminated by ',' stored as textfile

> ;

OK

Time taken: 0.426 seconds

hive (hivedarshan)> show tables;

OK

airlines

customer

+

**1) Write a program to find the count of customers for each profession**

hive (hivedarshan)> select profession,count(custno) from customer group by profession;

Query ID = bigdatamind43851\_20220620091132\_ae6be093-8b27-4b5c-9196-e2a12e306276

Total jobs = 1

**35**

**Please find the sales data set. txn id txn date cust id amount category product city state spendb**

hive (hivedarshan)> create table txnrecords(txnno INT, txndate STRING, custno INT, amount DOUBLE, category STRING, product STR

ING, city STRING, state STRING, spendby STRING) row format delimited fields terminated by ',' stored as textfile

> ;

OK

Time taken: 0.094 seconds

hive (hivedarshan)> show tables;

OK

airlines

customer

txnrecords

**2) Write a program to find the top 10 products sales wise**

hive (hivedarshan)> select product, sum(amount)as total from txnrecords group by product order by total desc limit 10;

**3) Write a program to create partiioned table on category**

hive (hivedarshan)> create table txnrecsByCat(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 textfi

le;

OK

Time taken: 0.088 seconds

hive (hivedarshan)> create table txnrecsByCat2(txnno INT, txndate STRING, custno INT, amount DOUBLE, product STRING, city STRIN

G, state STRING, spendby STRING) partitioned by (category STRING) clustered by (state) into 10 buckets row format delimited fie

lds terminated by ',' stored as textfile;

OK

Time taken: 0.115 seconds

hive (hivedarshan)> create table txnrecsByCat4(txnno INT, txndate STRING, custno INT, amount DOUBLE, category String, product S

TRING, city STRING, state STRING, spendby STRING) partitioned by (month STRING) row format delimited fields terminated by ',' s

tored as textfile;

OK

Time taken: 0.088 seconds

hive (hivedarshan)> create table txnrecsByCat3(txnno INT, txndate STRING, custno INT, amount DOUBLE, product STRING, city STRIN

G, state STRING) partitioned by (category STRING,spendby STRING) row format delimited fields terminated by ',' stored as textfi

le;

OK

Time taken: 0.166 seconds

hive (hivedarshan)> set hive.exec.dynamic.partition.mode=nonstrict;

hive (hivedarshan)> set hive.exec.dynamic.partition=true;

hive (hivedarshan)> show tables;

OK

airlines

customer

txnrecords

txnrecsbycat

txnrecsbycat2

txnrecsbycat3

txnrecsbycat4

Time taken: 0.04 seconds, Fetched: 7 row(s)

**QUESTION 3**

**[15 marks]**

***PySpark***

**Please find the AIRLINES data set Year Quarter Average revenue per seat Total number of booked seats**

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

>>> header = airlineRDD1.first()

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

>>> arrayRDD = airlineRDD2.map(lambda a : a.split(","))

>>> from pyspark.sql.types import StructType, IntegerType, DoubleType, LongType, StringType

>>> schema2 = StructType().add("Year",StringType(),True).add("qtr",IntegerType(),True).add("revenue",DoubleType(),True).add("se

ats",LongType(),True)

>>> airlinesDF = spark.read.format("csv").option("header","true").schema(schema2).load("/user/bigdatamind43846/Spark/airlines.c

sv")

>>> rdd=sc.textFile("/user/bigdatamind43851/airlines.csv")

>>> rdd1 = rdd.map(lambda a : a.encode("ascii","ignore"))

>>> header = rdd1.first()

>>> rdd2 = rdd1.filter(lambda a : a!=header)

>>> rdd2.count()

84

>>> array = rdd2.map(lambda a : a.split(","))

>>> for i in array.take(4):

... print(i)

...

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

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

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

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

1. **What was the highest number of people travelled in which year?**

rdd = sc.textFile("/user/bigdatamind43850/airlines.csv")

>>> header = rdd.first()

>>> rdd2 = rdd.filter(lambda a : a != header)

>>> rdd3 = rdd2.map(lambda a : a.encode("ascii","ignore"))

>>> arrayrdd = rdd3.map(lambda a : a.split(","))

>>> kvrdd = arrayrdd.map(lambda a : (a[0],int(a[3])))

>> counts = kvrdd.reduceByKey(lambda a,b : a+b)

>>> sort = counts.sortBy(lambda a : -a[1])

>>> for i in sort.take(1):

... print(i)

...

('2007', 176299)

1. **Identifying the highest revenue generation for which year**

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

>>> total = key.reduceByKey(lambda a,b: a+b)

>>> total1 = total.sortBy(lambda a: -a[1])

>>> total1.first()

>>> for i in sort.take(1):

... print(i)

...

('2013', 66363208.71)

1. **Identifying the highest revenue generation for which year and quarter (Common group)**

kvrdd2 = arrayrdd.map(lambda a : (a[0]+" "+a[1],float(a[2])\*int(a[3])))

counts2 = kvrdd2.reduceByKey(lambda a,b : a+b)

sort2 = counts2.sortBy(lambda a : -a[1])

>>> for i in sort2.take(1):

... print(i)

...

**('2014 4', 18819408.48)**