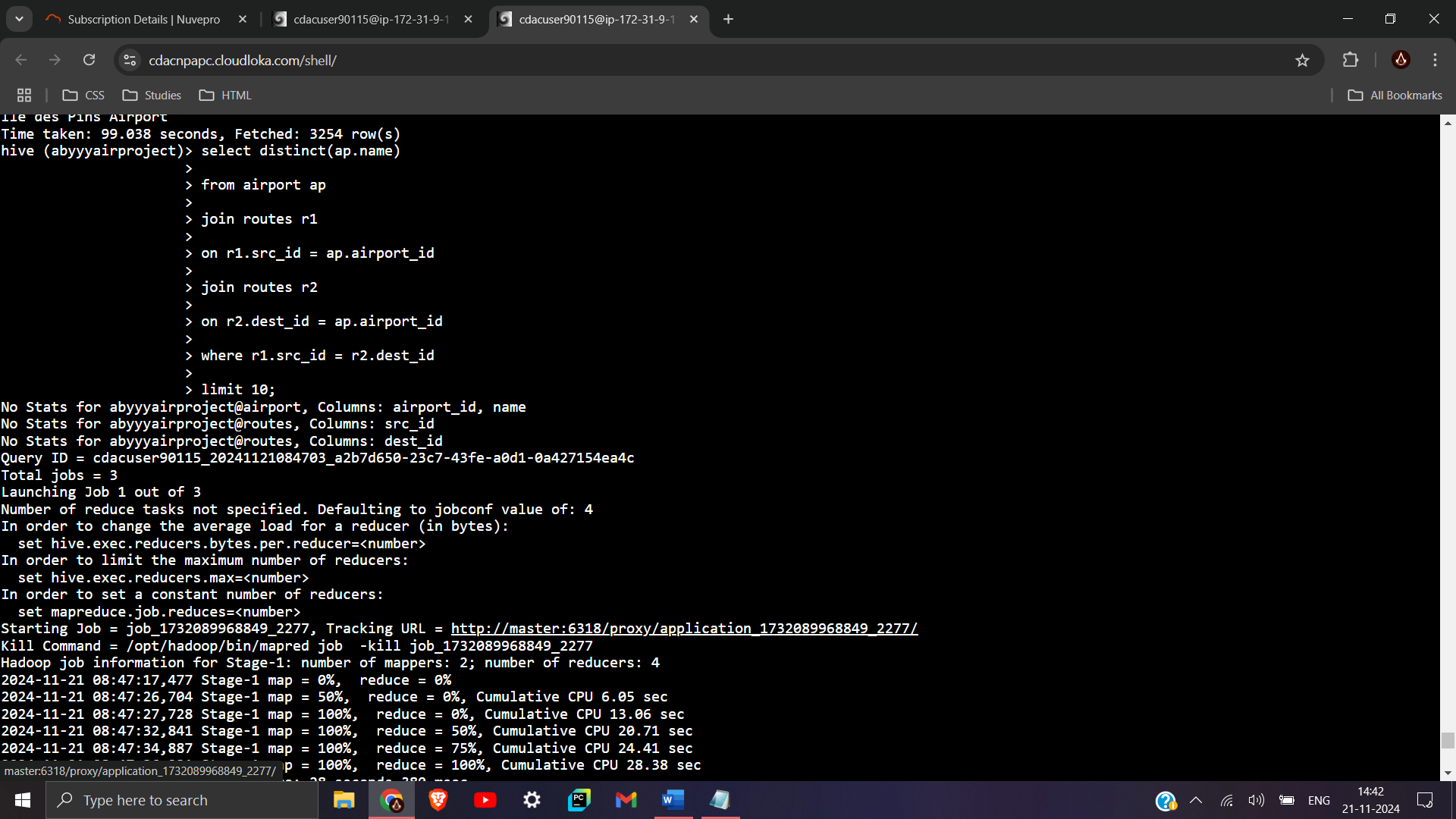
Abhay Patil Big Data Exam

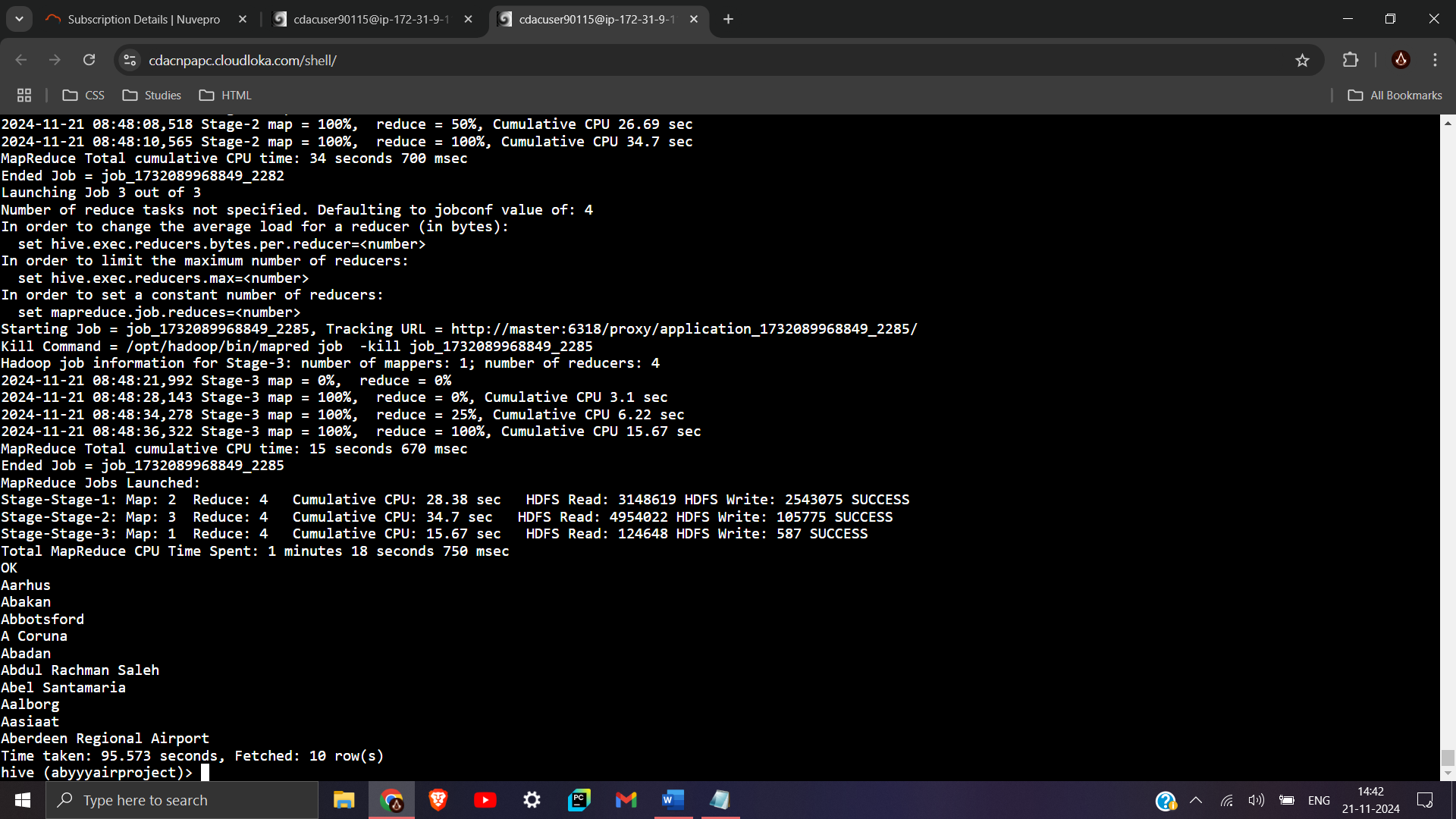
ID : 240840325002

**HIVE : (20 mrks)**

Q1.

A]





select distinct(ap.name)

from airport ap

join routes r1

on r1.src\_id = ap.airport\_id

join routes r2

on r2.dest\_id = ap.airport\_id

where r1.src\_id = r2.dest\_id

limit 10;

B]

with (select max(equipment) from routes) as man\_count,

(select equipment, count(equipment)

from routes

group by equipment

having count(equipment) = max\_count);

“Can't get the output due to crashing of the hive terminal.”

C]

select a.name, count(\*)

from airlines a

join routes r

on a.airline\_id = r.airline\_id

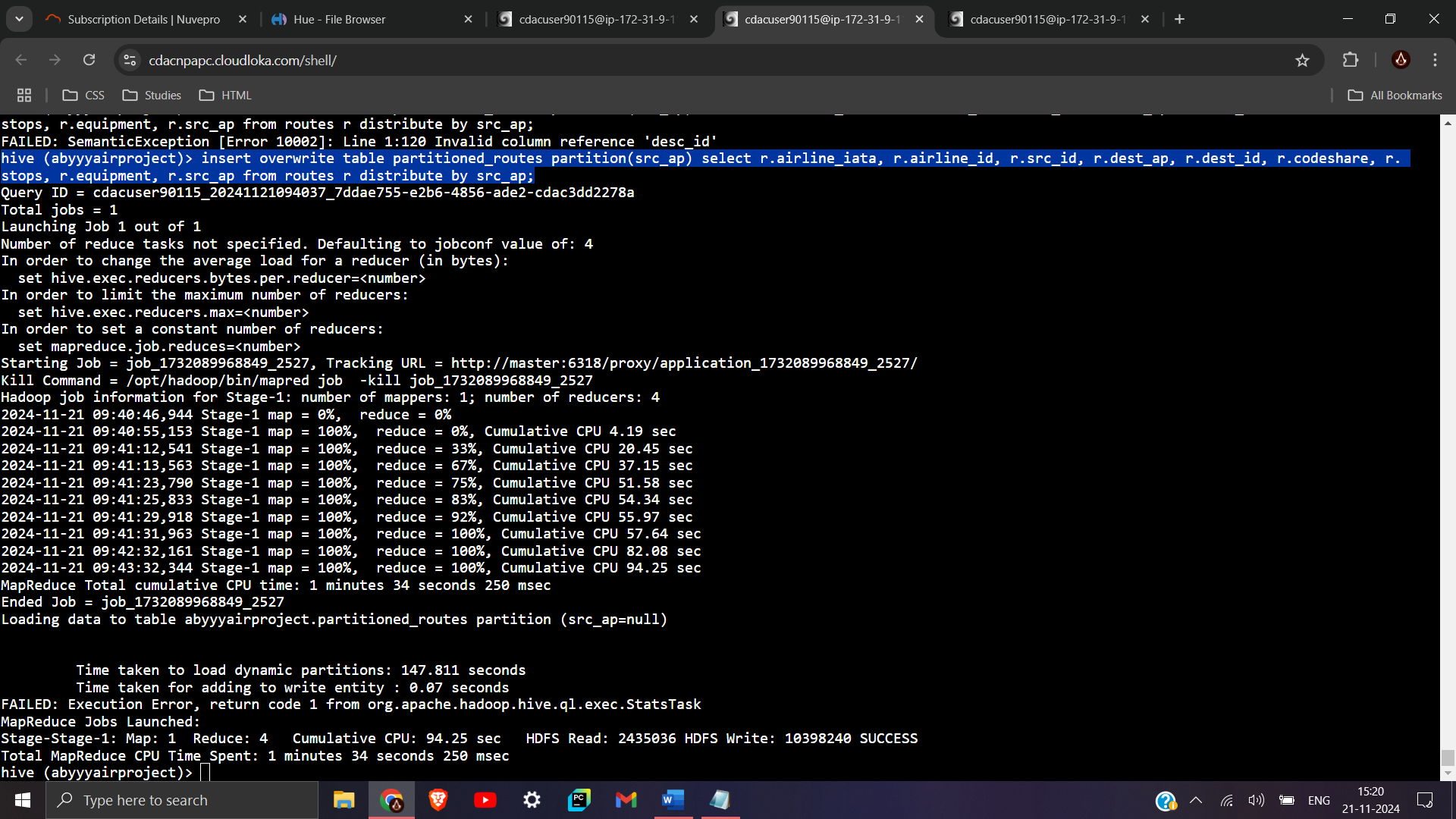
group by a.name

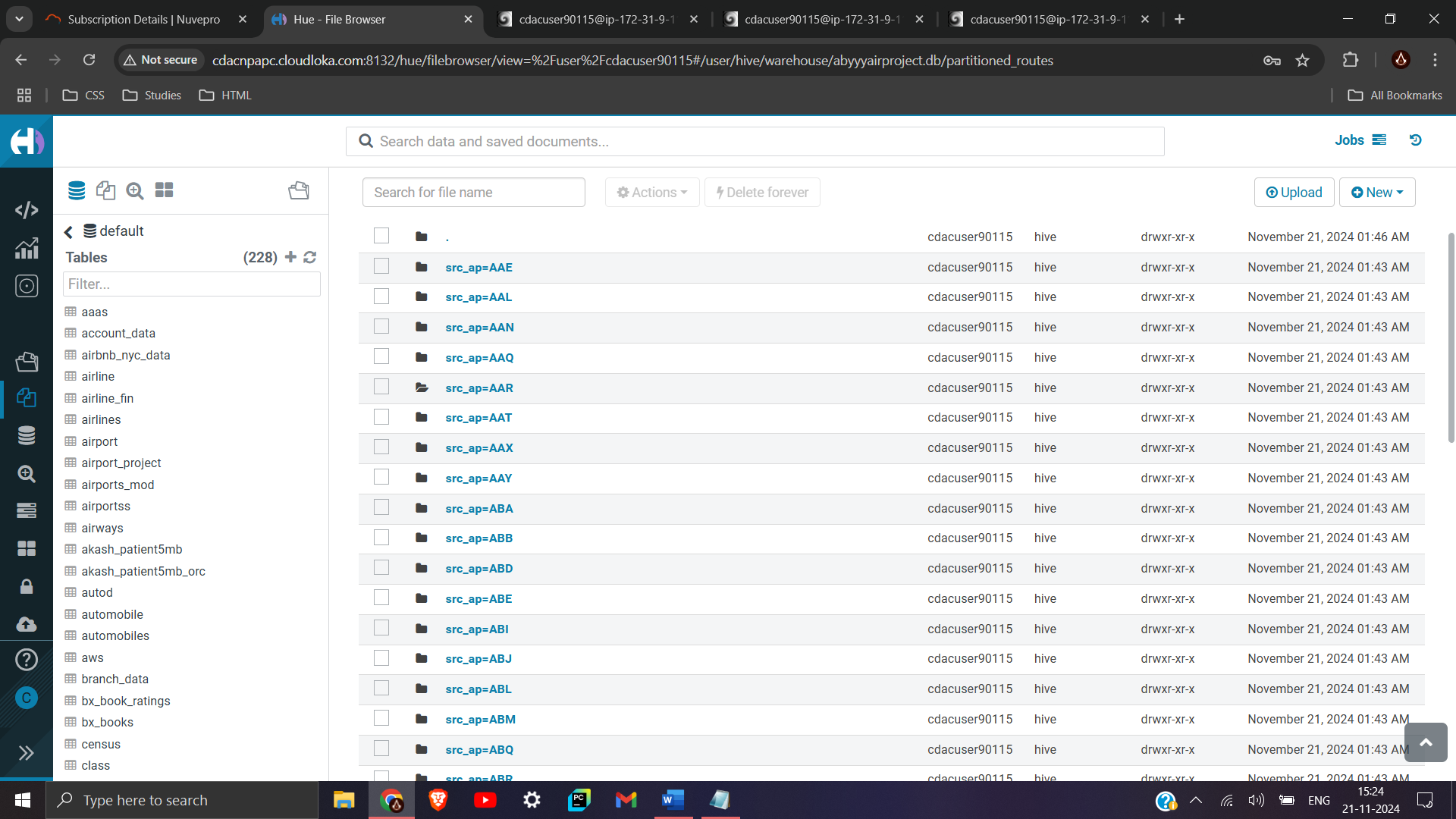
having max(count(\*)) = count(\*);

“Can't get the output due to crashing of the hive terminal.”

Q2.

A]





create table partitioned\_routes (airline\_iata string, airline\_id int, src\_id int, dest\_ap string, dest\_id int,

codeshare string, stops int, equipment string)

partitioned by (src\_ap string)

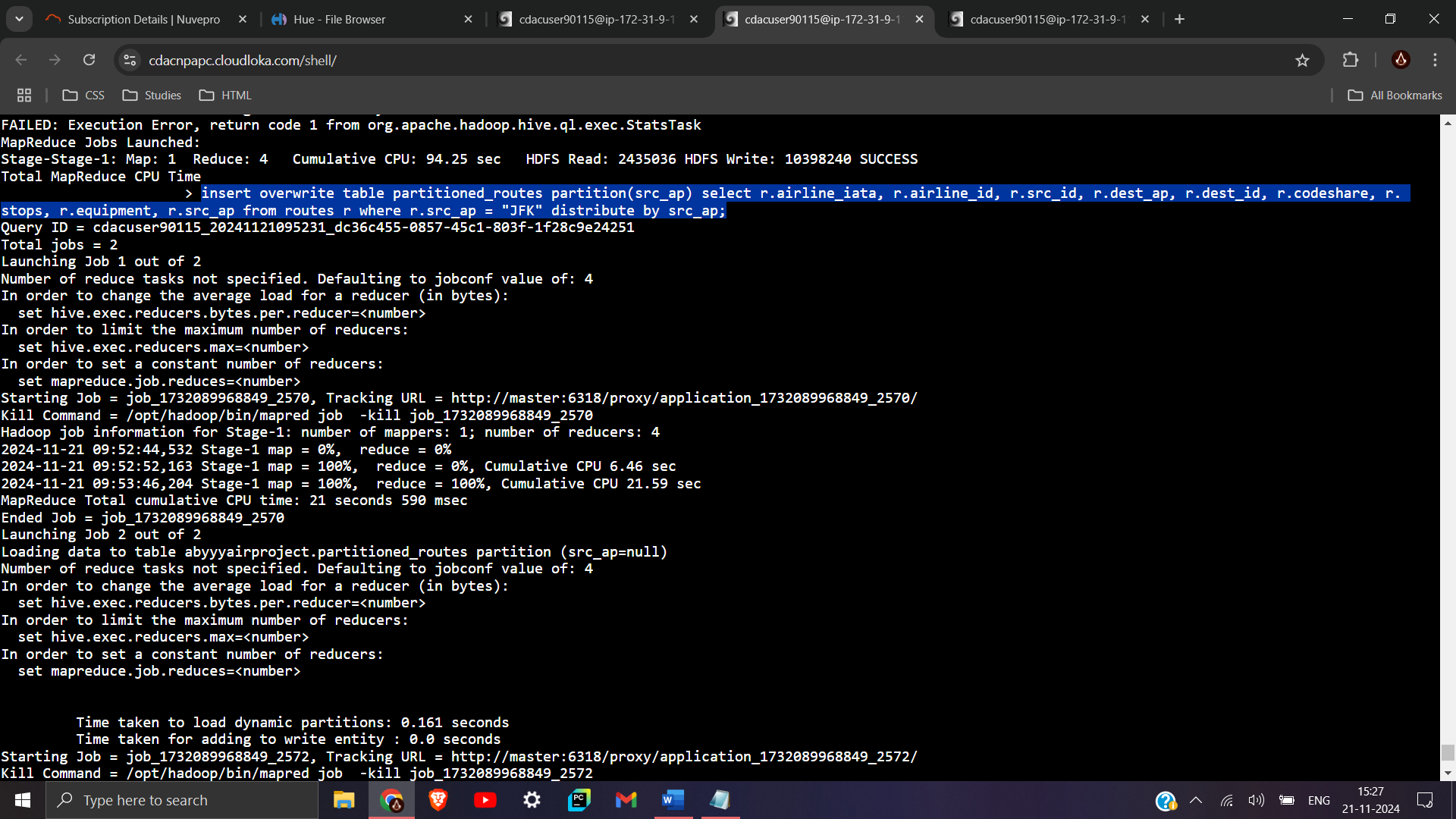
row format delimited

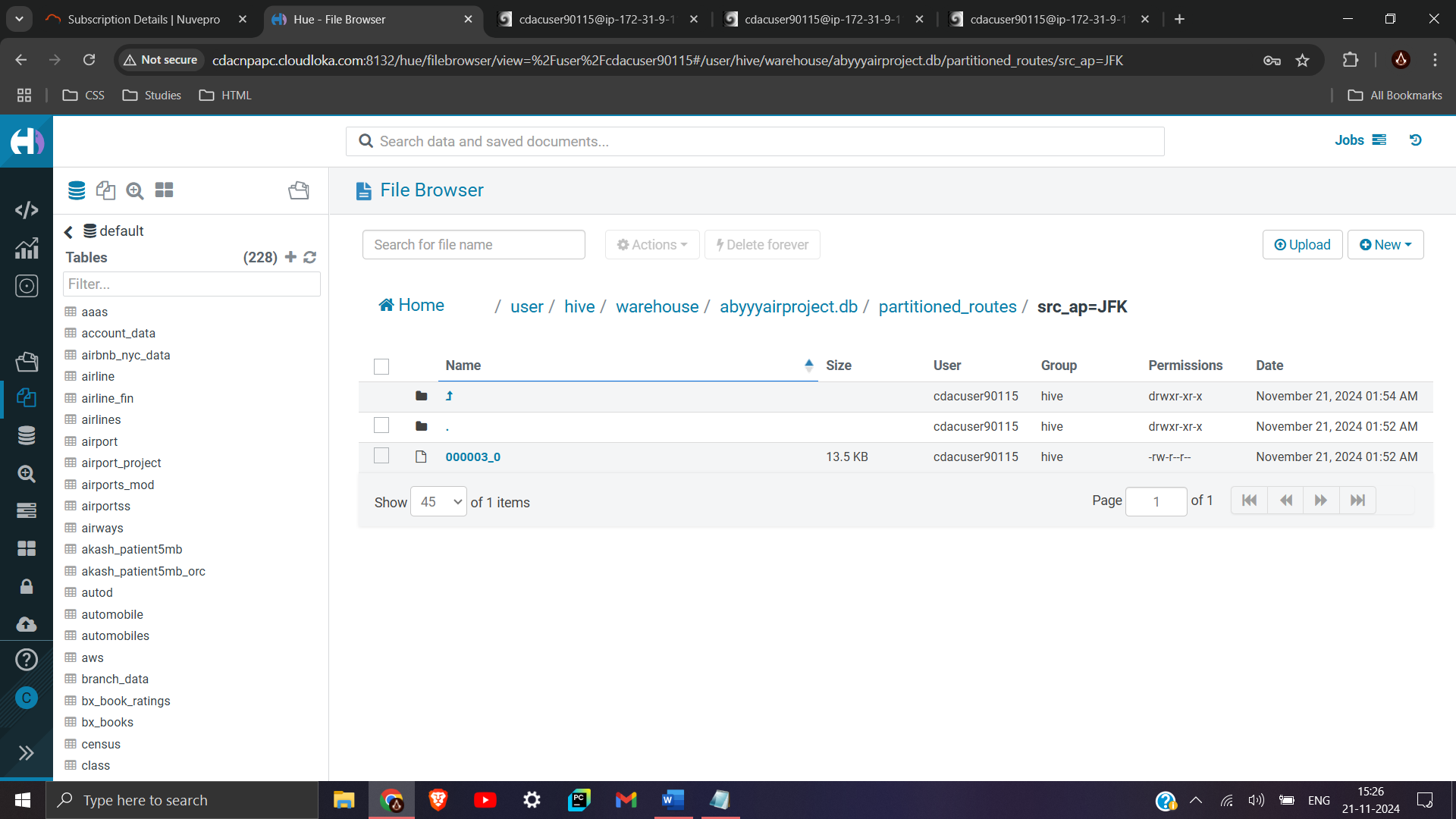
fields terminated by ','

stored as textfile;

insert overwrite table partitioned\_routes partition(src\_ap) select r.airline\_iata, r.airline\_id, r.src\_id, r.dest\_ap, r.dest\_id, r.codeshare, r.stops, r.equipment, r.src\_ap from routes r distribute by src\_ap;

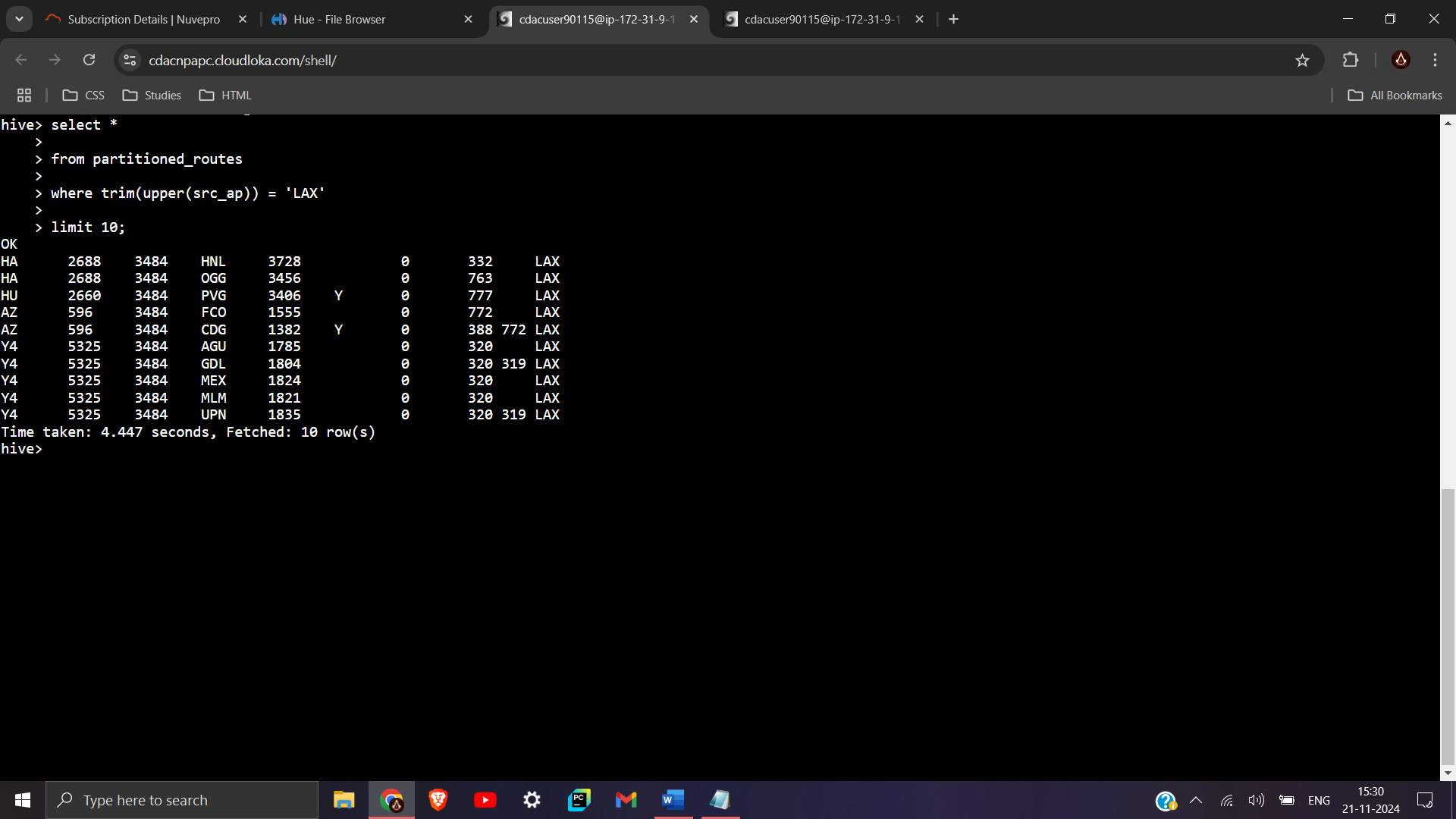
B]





insert overwrite table partitioned\_routes partition(src\_ap) select r.airline\_iata, r.airline\_id, r.src\_id, r.dest\_ap, r.dest\_id, r.codeshare, r.stops, r.equipment, r.src\_ap from routes r where r.src\_ap = "JFK" distribute by src\_ap;

C]



select \*

from partitioned\_routes

where trim(upper(src\_ap)) = 'LAX'

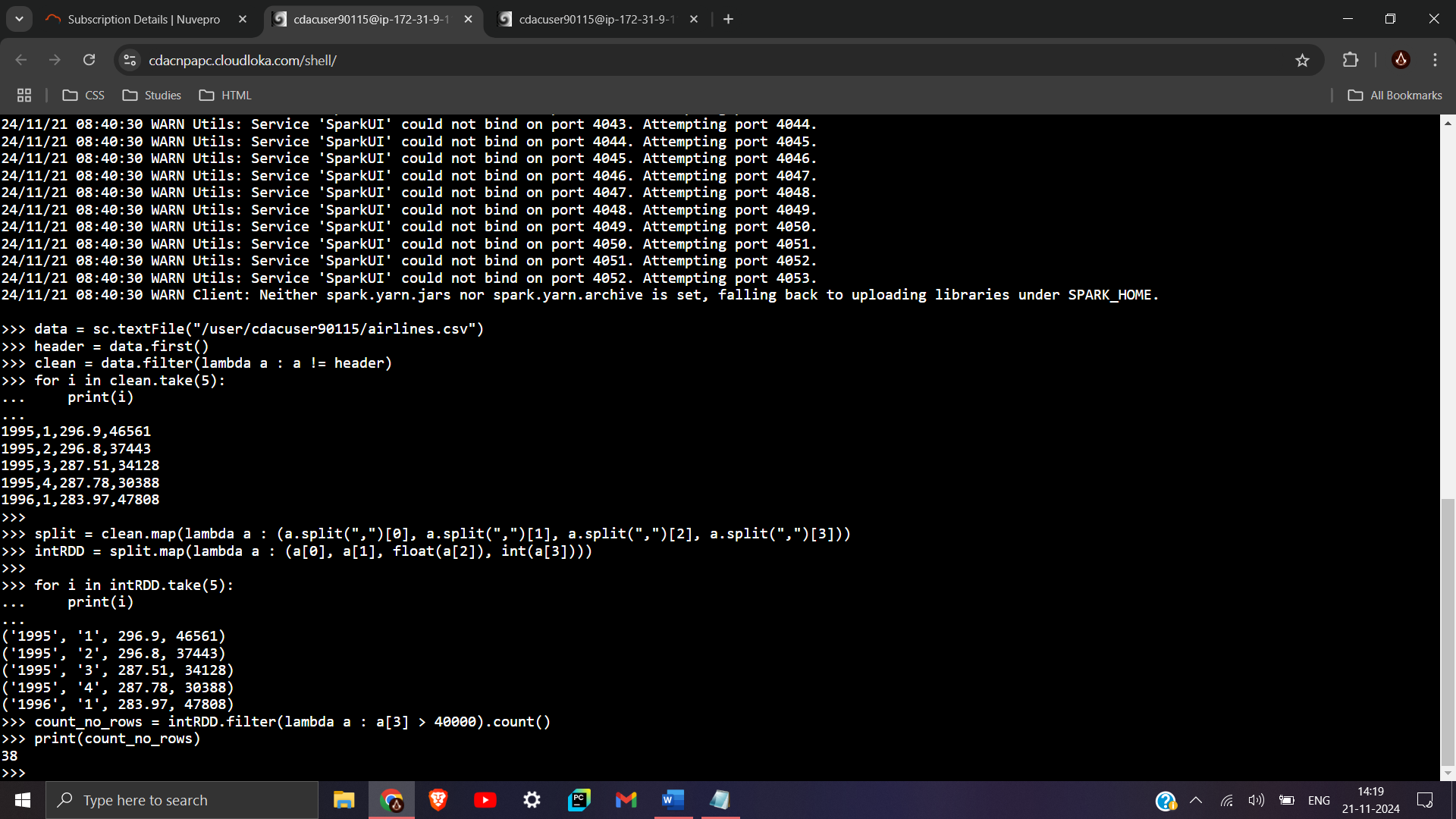
limit 10;

D]

**SPARK : (20 mrks)**

**Q1.**

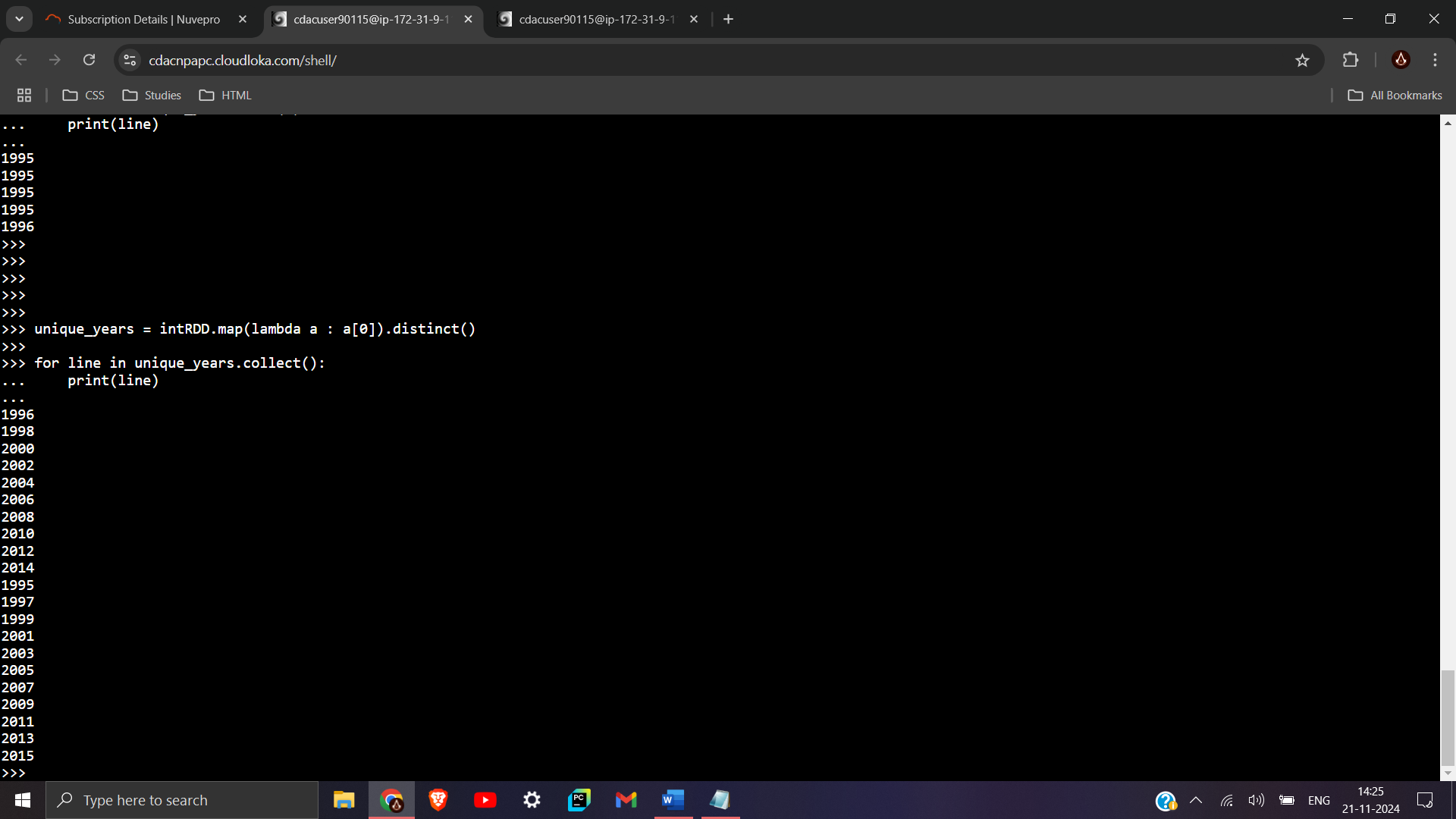
**A]**



>>> count\_no\_rows = intRDD.filter(lambda a : a[3] > 40000).count()

>>> print(count\_no\_rows)

B]



>>> unique\_years = intRDD.map(lambda a : a[0]).distinct()

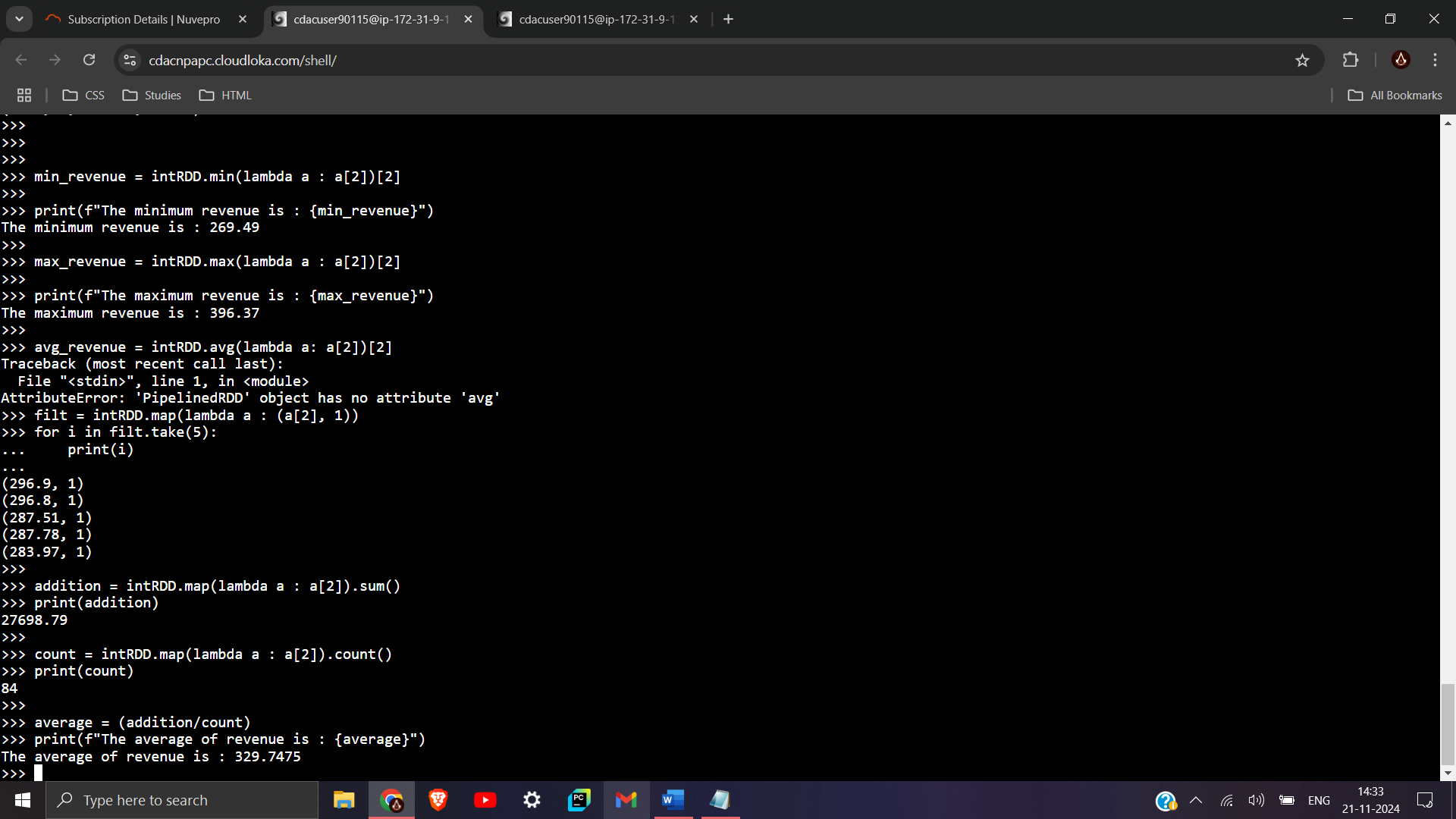
>>>

>>> for line in unique\_years.collect():

... print(line)

Q2.

A]



>>> min\_revenue = intRDD.min(lambda a : a[2])[2]

>>>

>>> print(f"The minimum revenue is : {min\_revenue}")

The minimum revenue is : 269.49

>>>

>>> max\_revenue = intRDD.max(lambda a : a[2])[2]

>>>

>>> print(f"The maximum revenue is : {max\_revenue}")

The maximum revenue is : 396.37

>>> addition = intRDD.map(lambda a : a[2]).sum()

>>> print(addition)

27698.79

>>>

>>> count = intRDD.map(lambda a : a[2]).count()

>>> print(count)

84

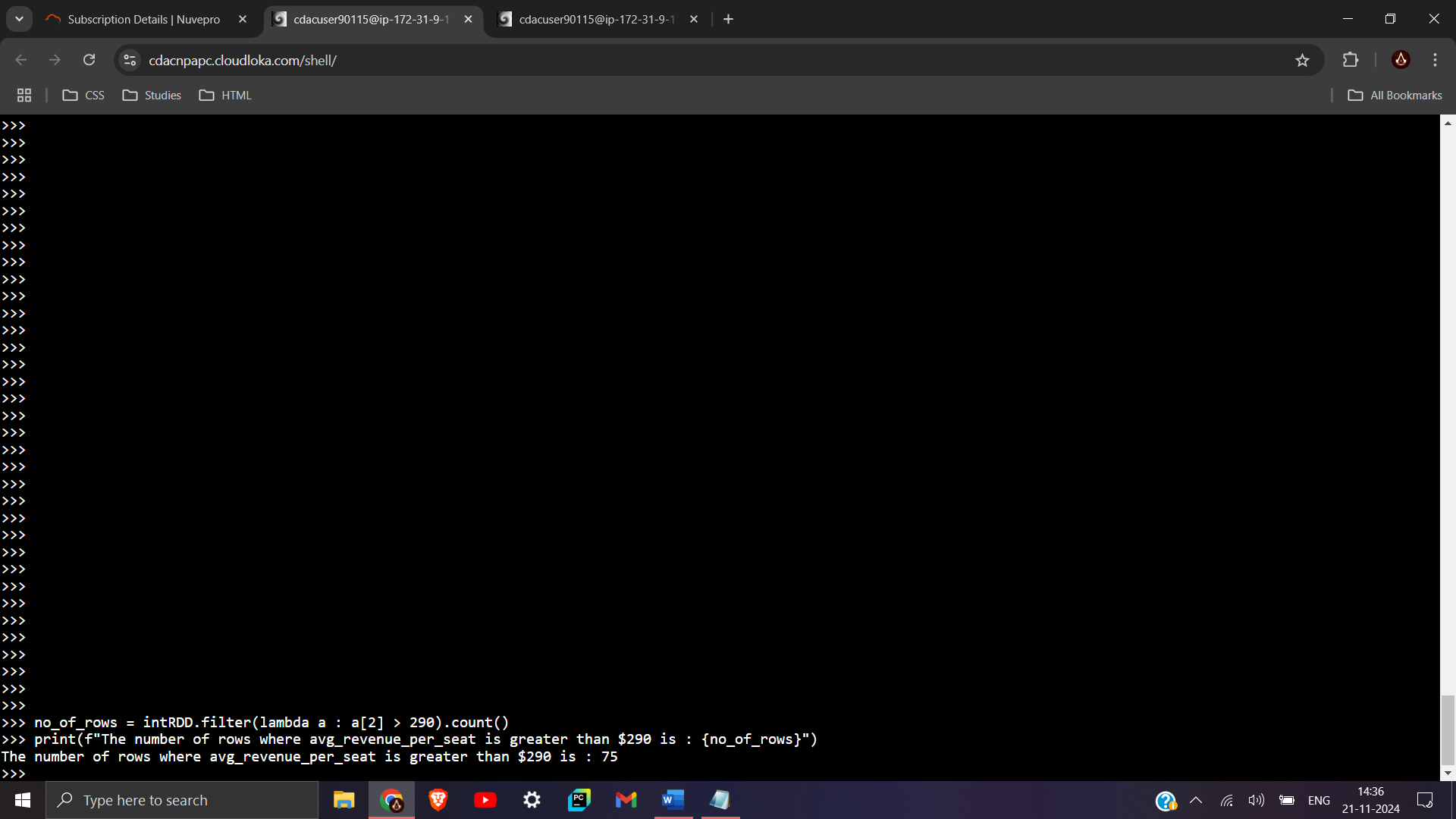
>>>

>>> average = (addition/count)

>>> print(f"The average of revenue is : {average}")

The average of revenue is : 329.7475

B]

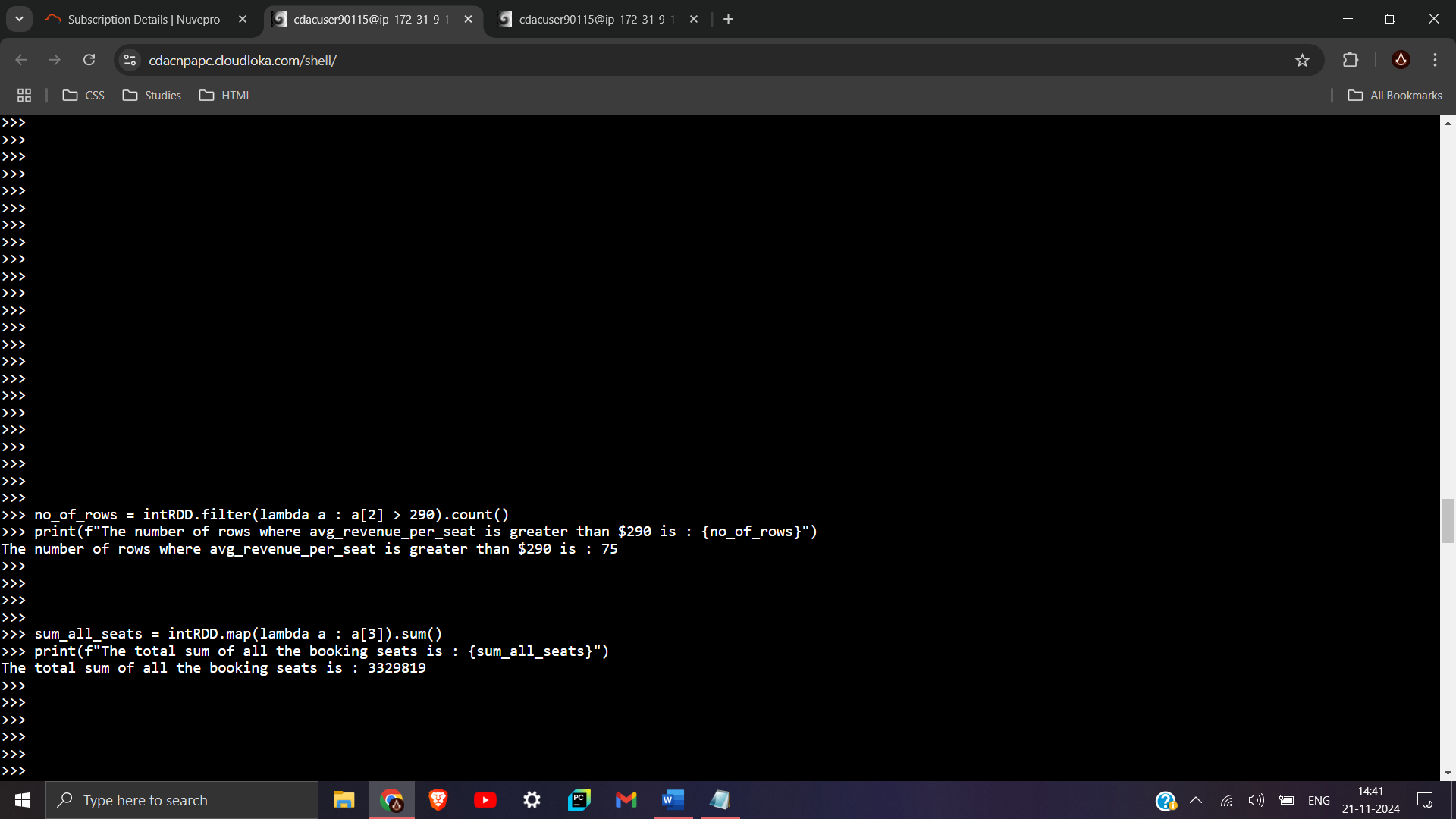


>>> no\_of\_rows = intRDD.filter(lambda a : a[2] > 290).count()

>>> print(f"The number of rows where avg\_revenue\_per\_seat is greater than $290 is : {no\_of\_rows}")

The number of rows where avg\_revenue\_per\_seat is greater than $290 is : 75

C]

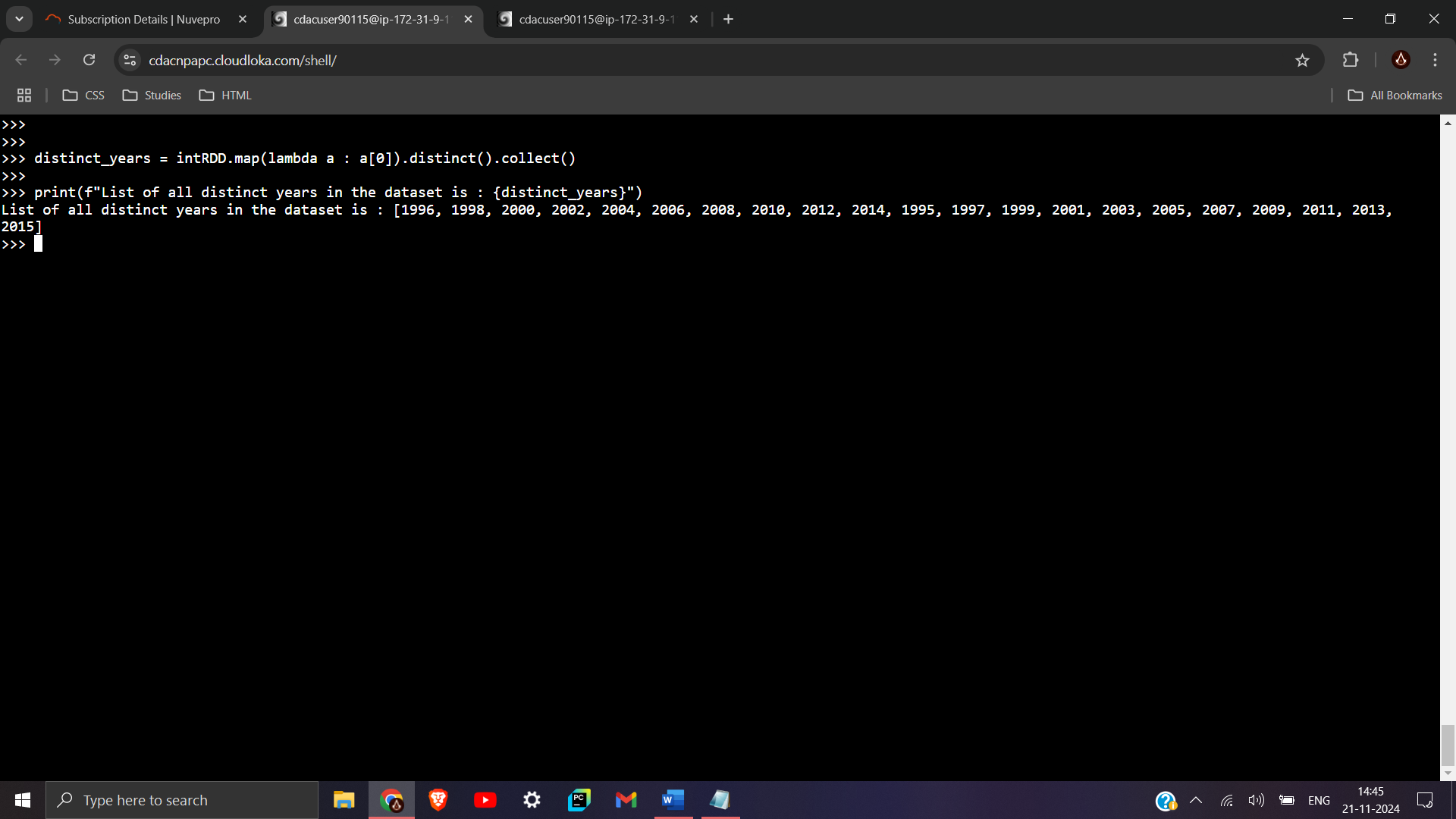


>>> sum\_all\_seats = intRDD.map(lambda a : a[3]).sum()

>>> print(f"The total sum of all the booking seats is : {sum\_all\_seats}")

The total sum of all the booking seats is : 3329819

D]



>>> distinct\_years = intRDD.map(lambda a : a[0]).distinct().collect()

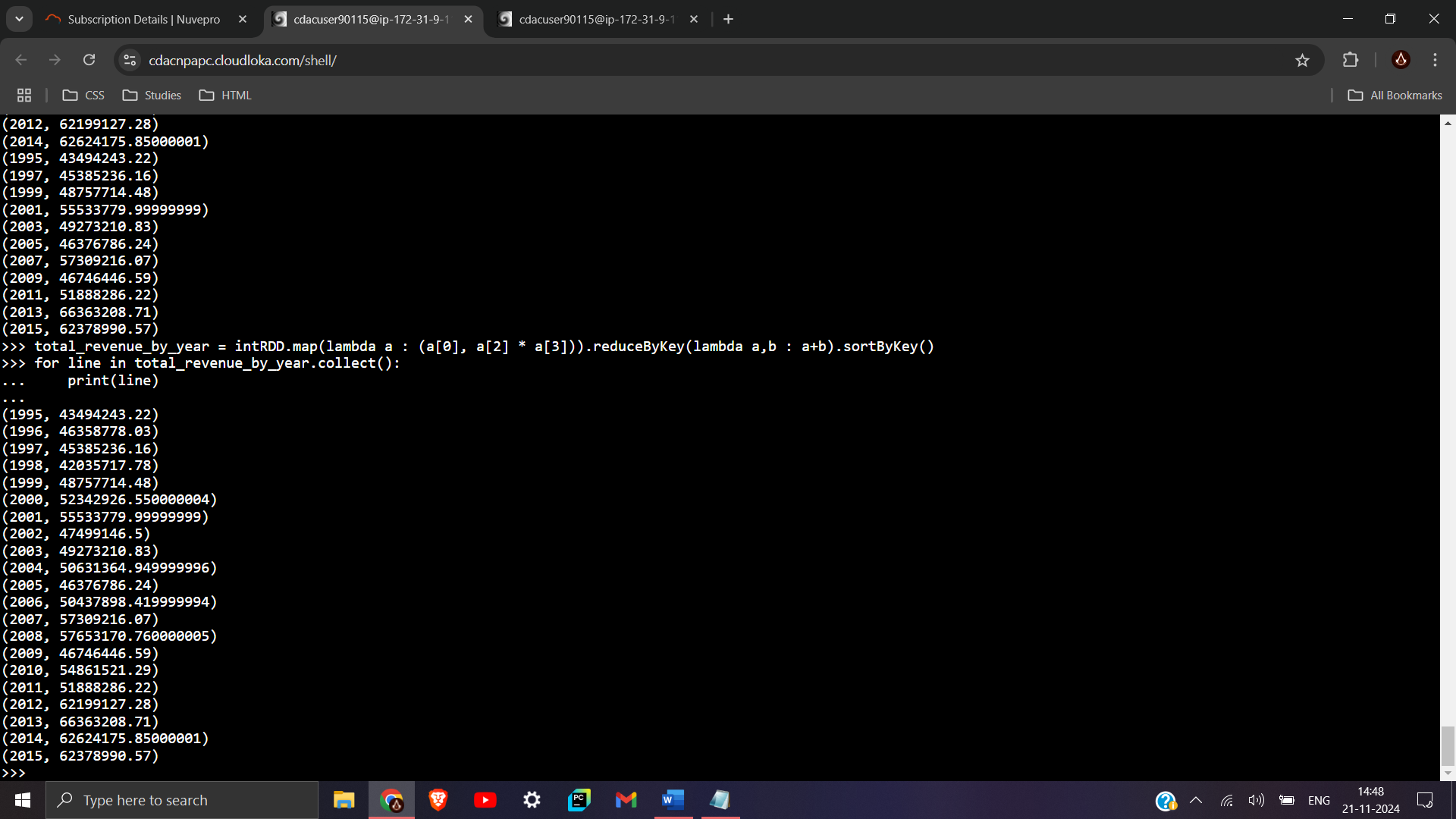
>>>

>>> print(f"List of all distinct years in the dataset is : {distinct\_years}")

List of all distinct years in the dataset is : [1996, 1998, 2000, 2002, 2004, 2006, 2008, 2010, 2012, 2014, 1995, 1997, 1999, 2001, 2003, 2005, 2007, 2009, 2011, 2013,

2015]

E]



>>> total\_revenue\_by\_year = intRDD.map(lambda a : (a[0], a[2] \* a[3])).reduceByKey(lambda a,b : a+b).sortByKey()

>>> for line in total\_revenue\_by\_year.collect():

... print(line)

...