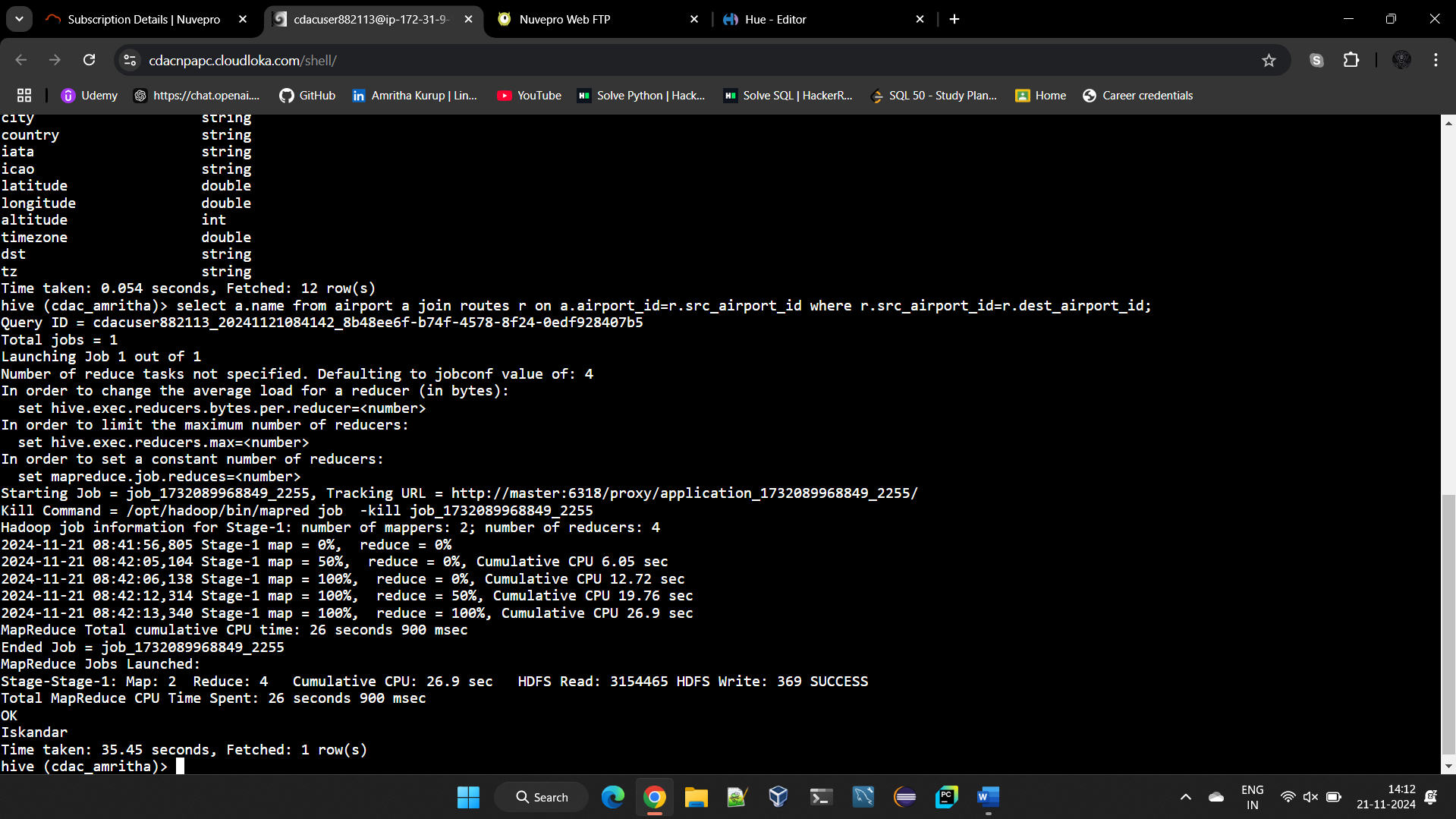
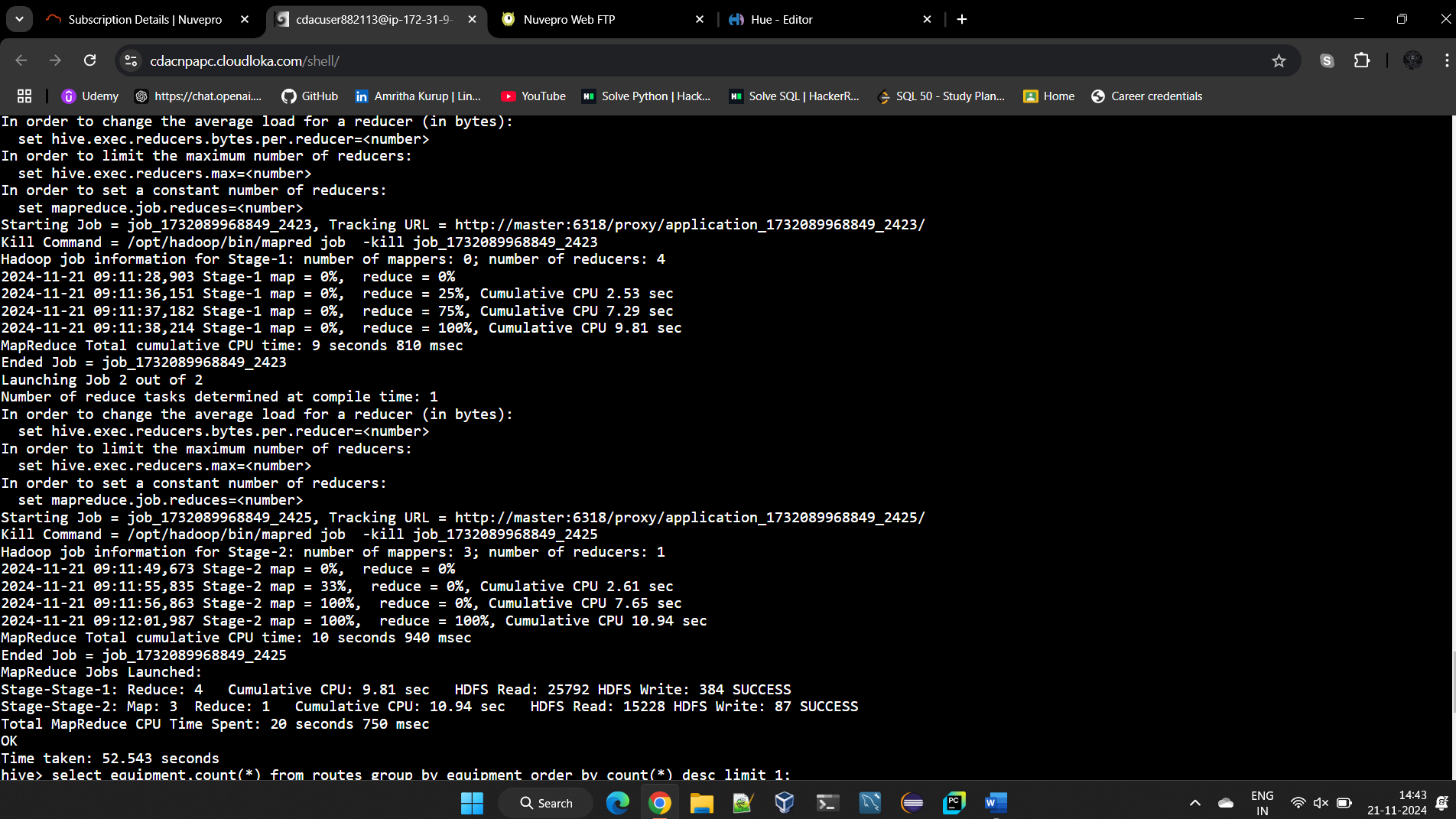
HIVE

Question 1.

1. Select a.name from airport a join routes r on a.airport\_id=r.src\_airport\_id where r.src\_airport\_id=r.dest\_airport\_id;



1. select equipment,count(\*) from routes group by equipment order by count(\*) desc limit 1;



1. select a.name from airline join r.routes on a.airline\_id=r.airline\_id group by a.name order by count(a.name) desc limit 1;

SPARK

Question 1:

1. from pyspark import SparkContext, SparkConf

sc=SparkContext()

rdd=sc.textFile("/user/cdacuser882113/airlines.csv")

for i in rdd.take(5):

... print(i)

header=rdd.first()

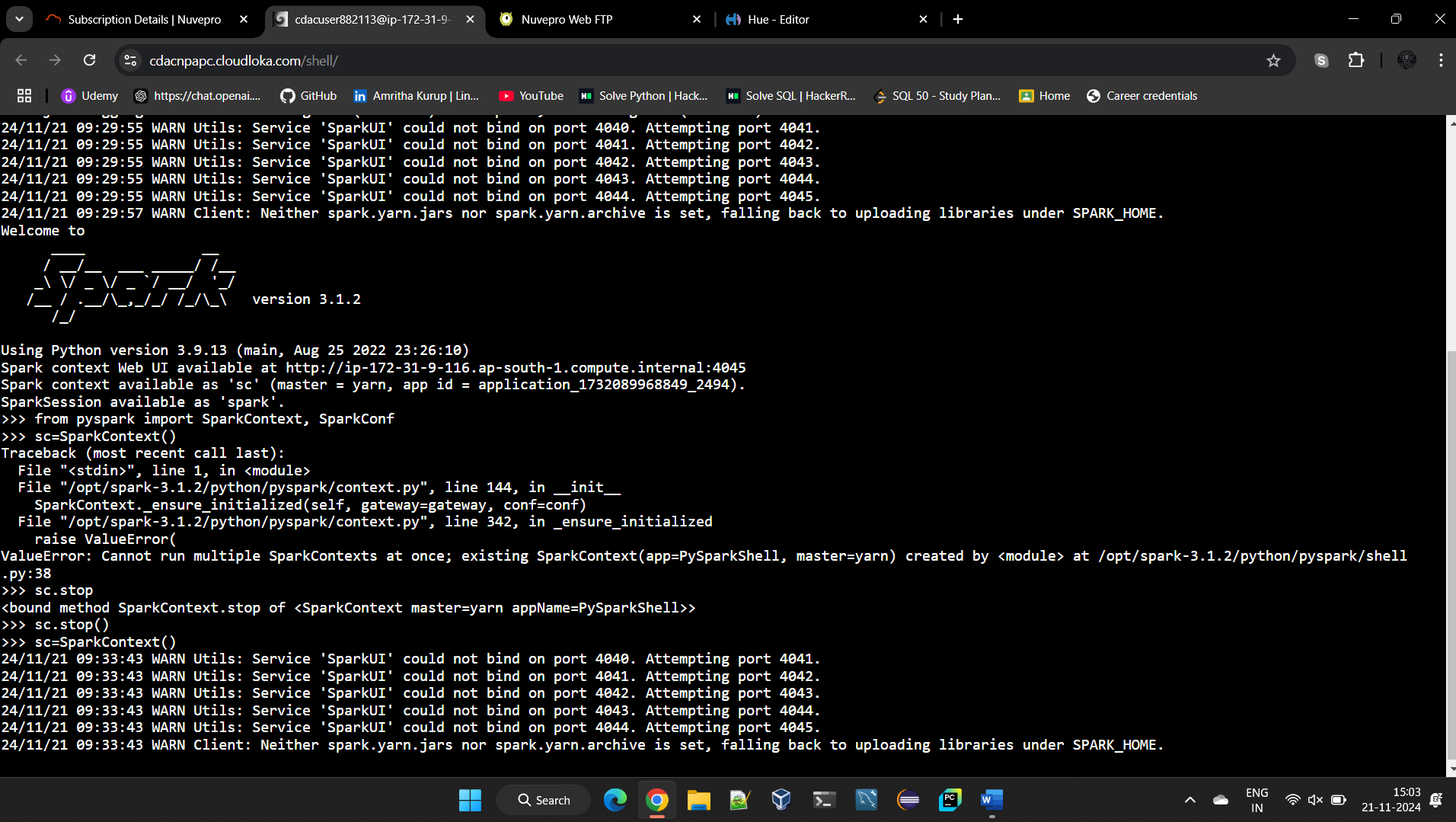
rdd1 = rdd.filter(lambda a:a != header)

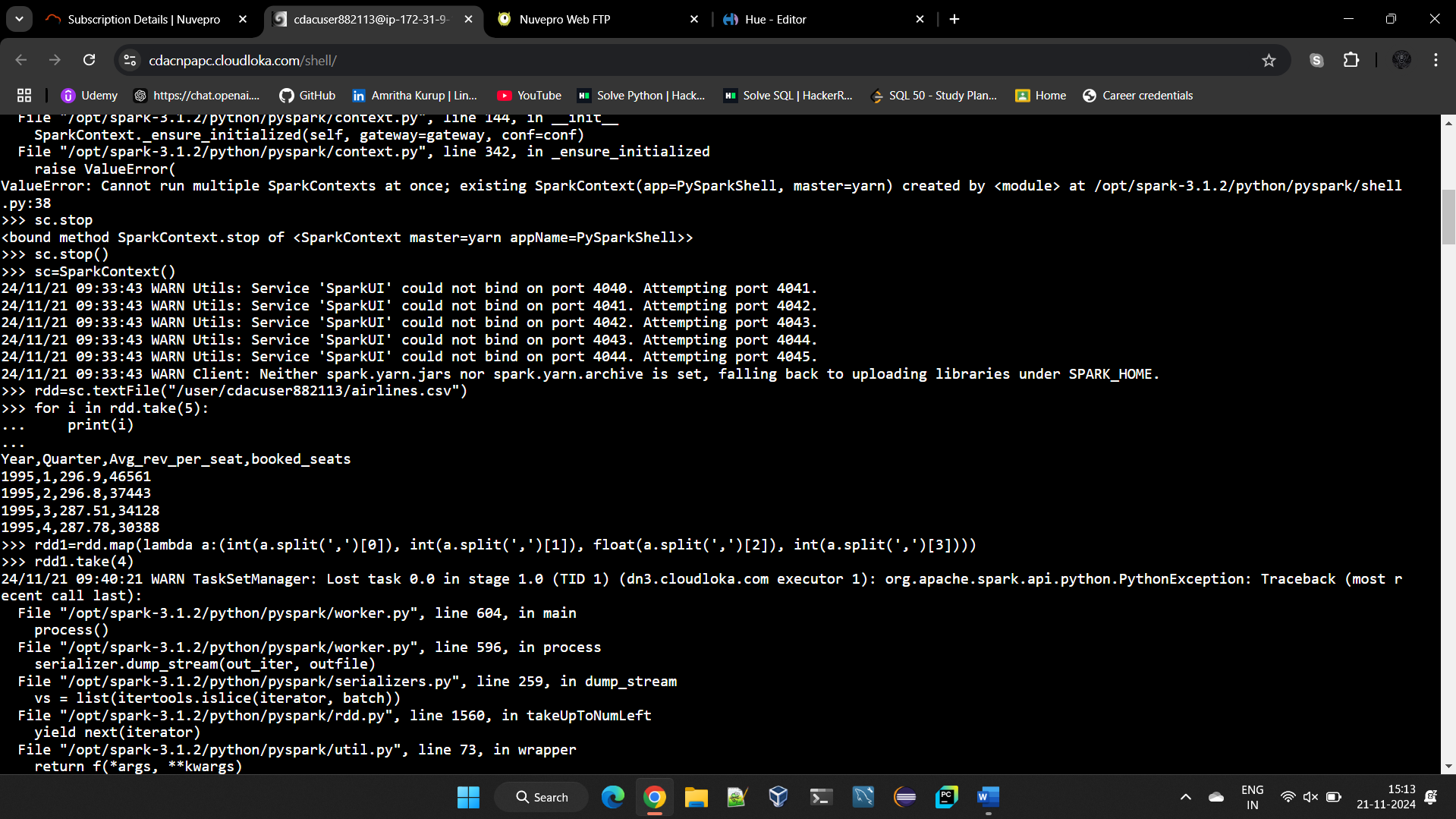
rdd2=rdd1.map(lambda a:(a.split(',')[0], a.split(',')[1], a.split(',')[2], a.split(',')[3]))

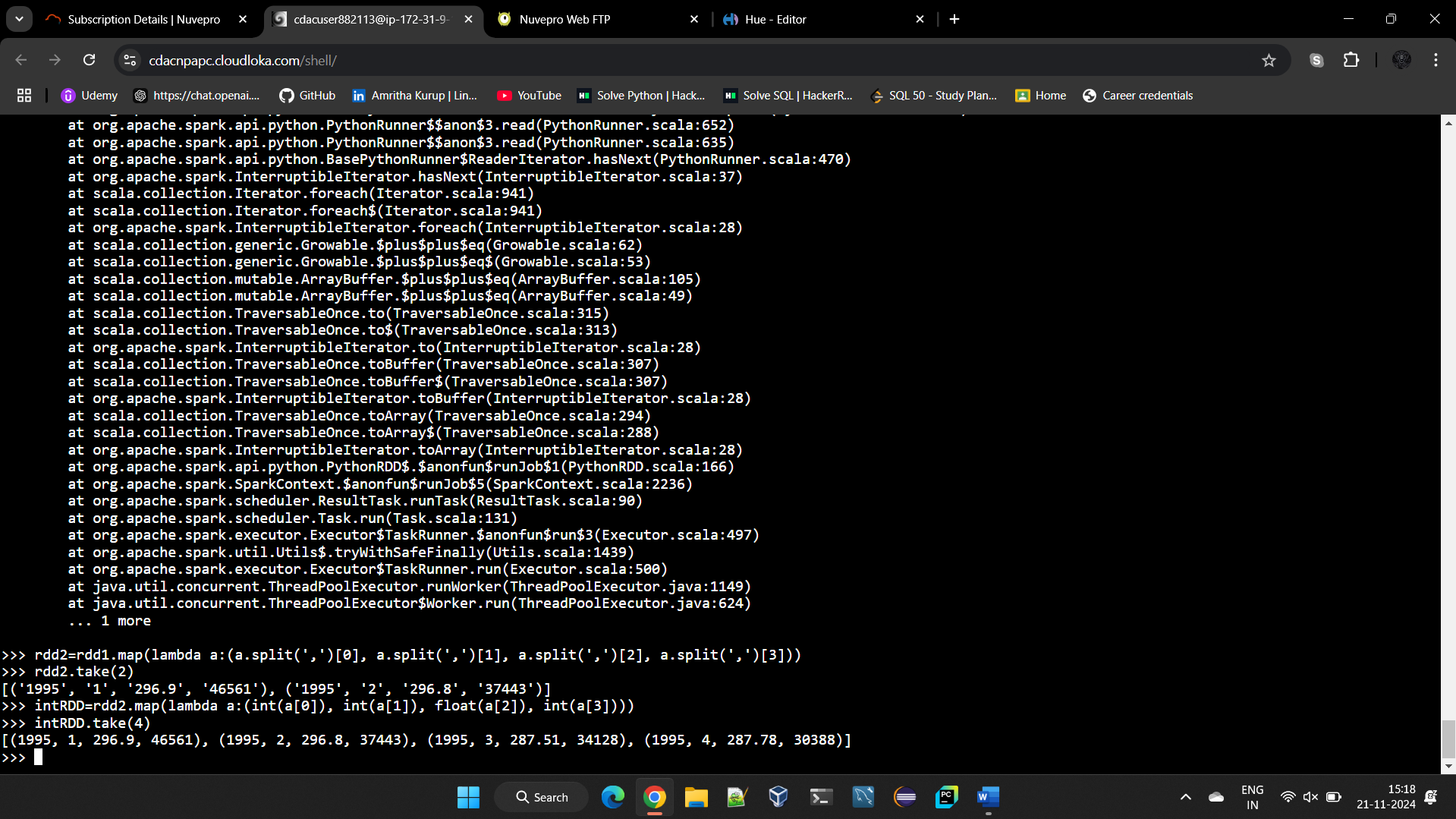
intRDD=rdd2.map(lambda a:(int(a[0]), int(a[1]), float(a[2]), int(a[3])))

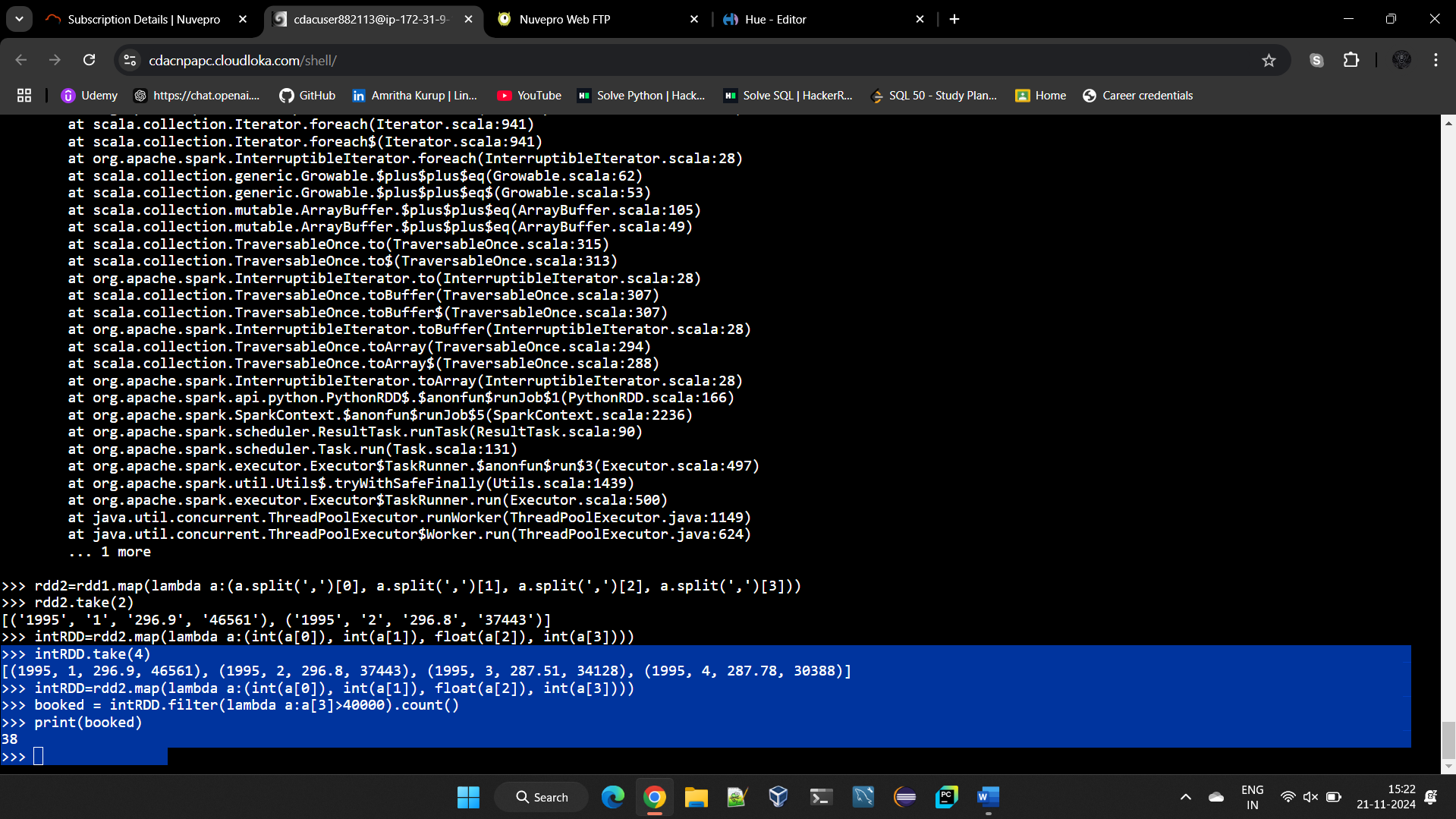
no of book exeeds 40000

booked = intRDD.filter(lambda a:a[3]>40000).count()







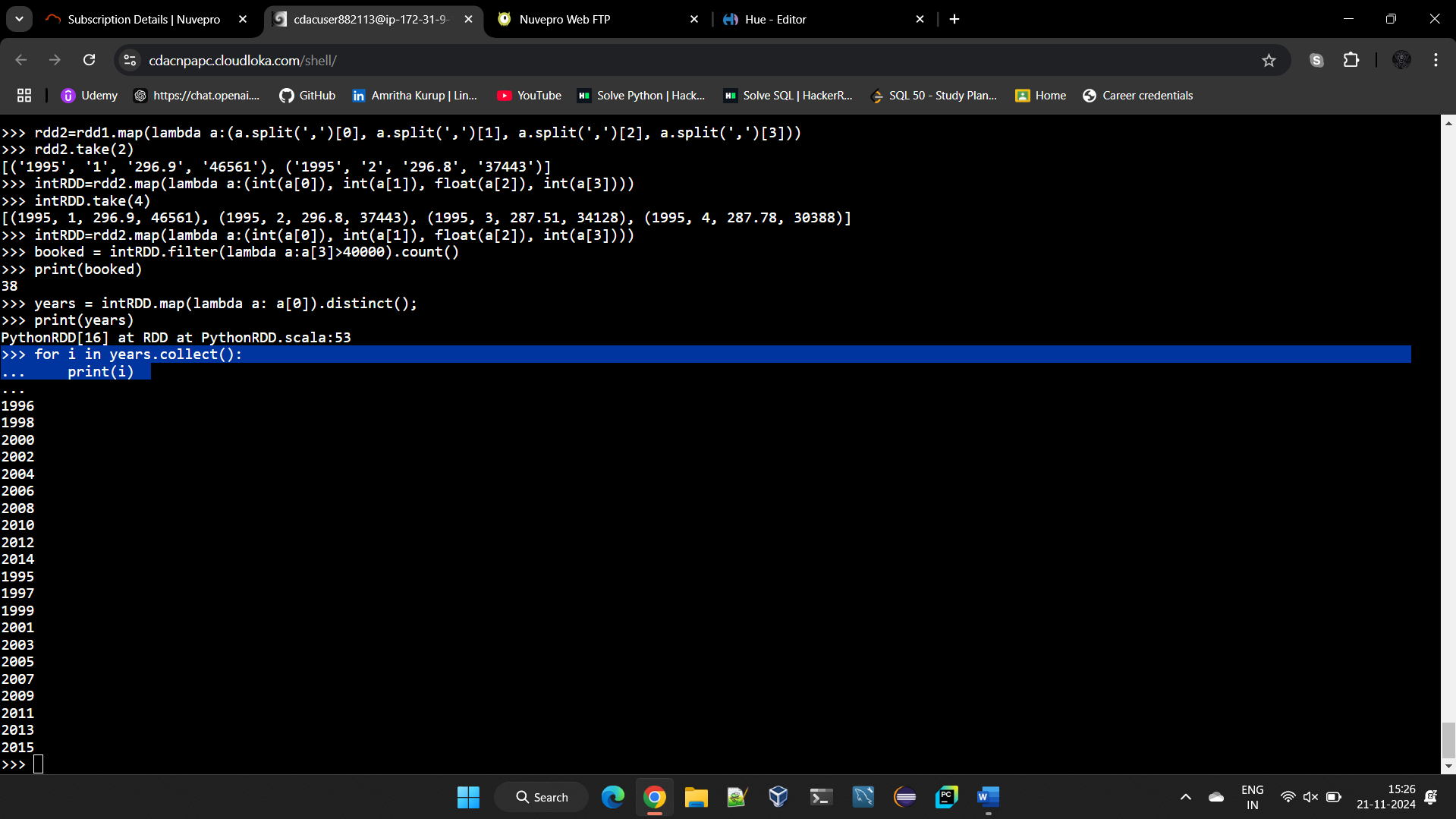




years = intRDD.map(lambda a: a[0]).distinct()

>>> for i in years.collect():

... print(i)



Question 2.

orderdes=intRDD.map(lambda a: -a[2])

Minimum revenue per seat = orderdes.take(1)

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

Minimum revenue per seat = orderasc.take(1)

Average = intRDD.map(lambda a: a[2]).mean()

avgRevPerSeat = intRDD.filter(lambda a:a[2]>290).count()

print(avgRevPerSeat)

Combined = intRDD.map(lambda a : (a[0], a[3]))

quaterCombined= Combined.reduceByKey(lambda a,b=a+b)

for i in quaterCombined.collect():

print(i)

years = intRDD.map(lambda a: a[0]).distinct()

for i in years.collect():

... print(i)

Rev = intRDD.map(lambda a : (a[0], a[2]\*a[3]))

cumulative= Rev.reduceByKey(lambda a,b=a+b)

for i in cumulative.collect():

print(i)

a[2]\*a[3] will give the total revenue and a[0] denotes the year