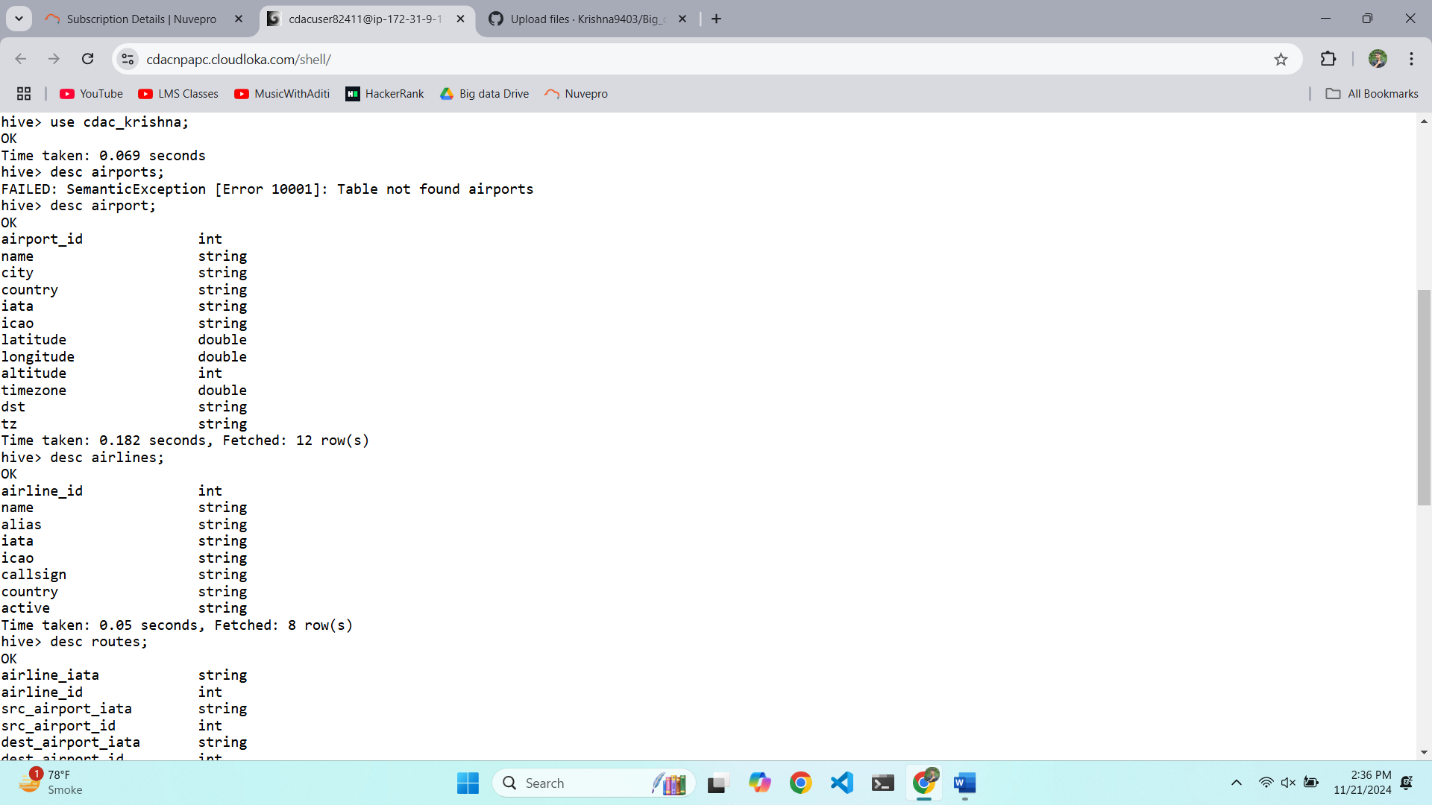
Q2 use cdac\_krishna;

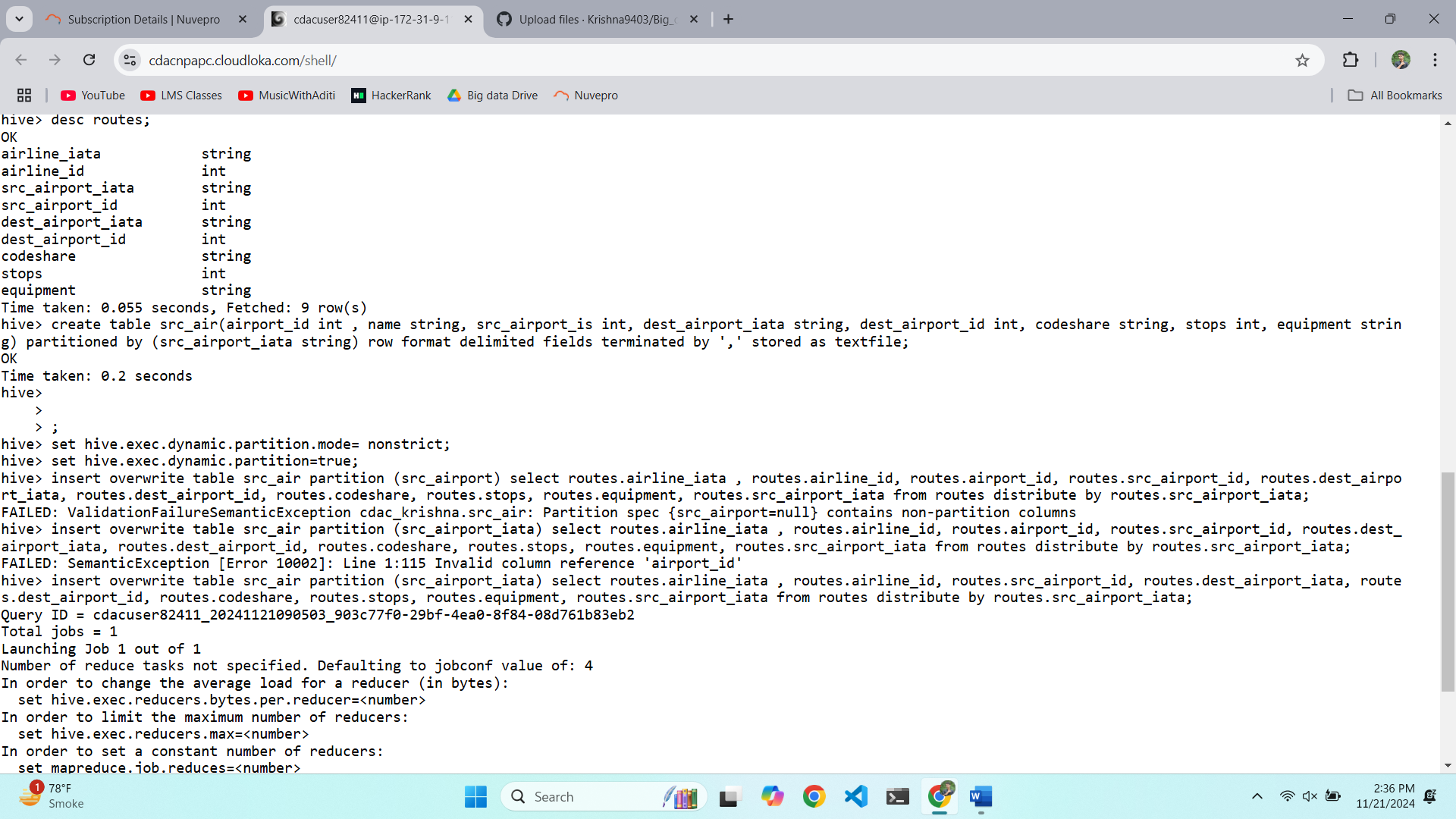


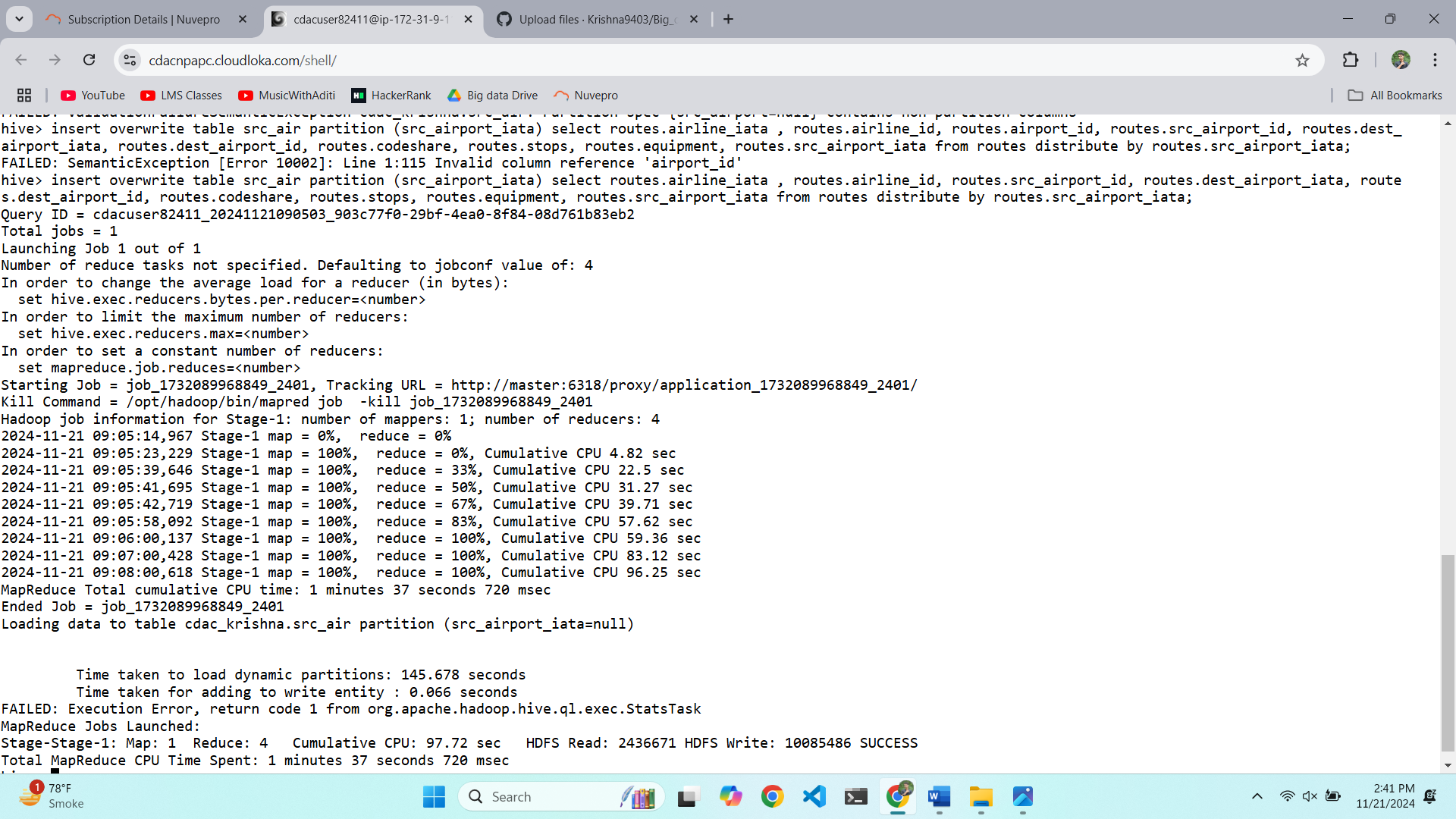
Creating table - create table src\_air(airport\_id int , name string, src\_airport\_is int, dest\_airport\_iata string, dest\_airport\_id int, codeshare string, stops int, equipment strin

g) partitioned by (src\_airport\_iata string) row format delimited fields terminated by ',' stored as textfile;

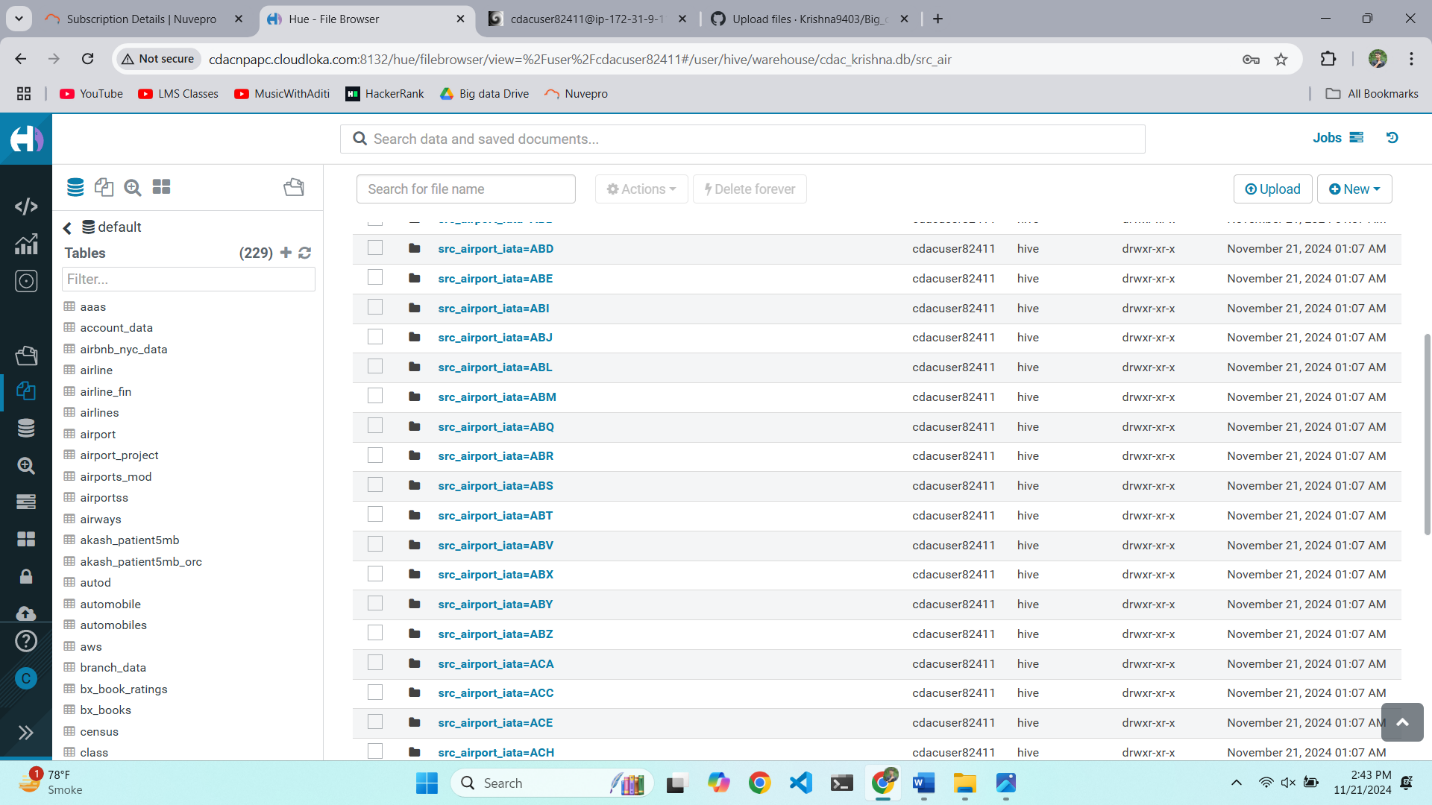
Inserting data - insert overwrite table src\_air partition (src\_airport\_iata) select routes.airline\_iata , routes.airline\_id, routes.src\_airport\_id, routes.dest\_airport\_iata, route

s.dest\_airport\_id, routes.codeshare, routes.stops, routes.equipment, routes.src\_airport\_iata from routes distribute by routes.src\_airport\_iata;





Partition created -



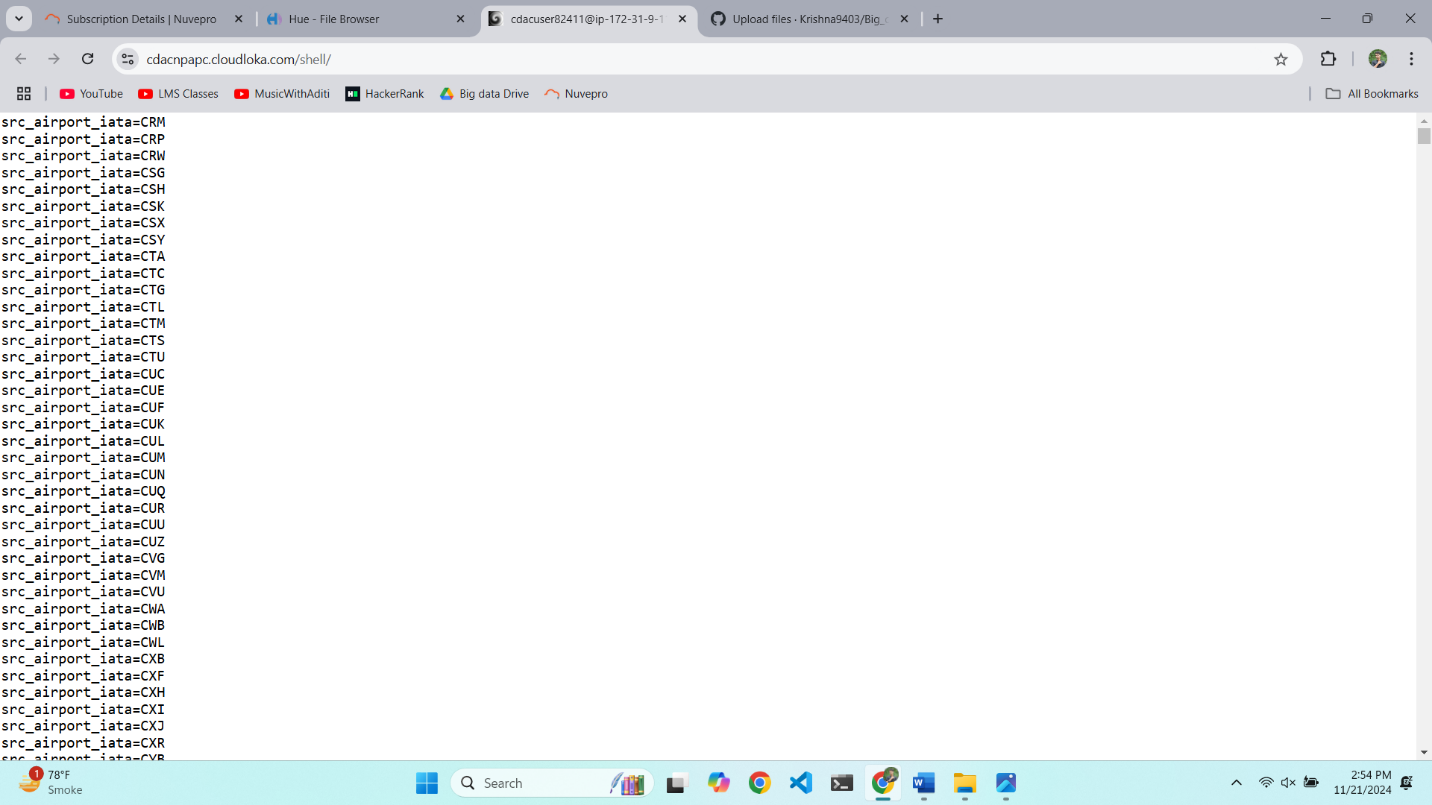
set hive.vectorized.execution.reduce.enabled=false;

set hive.vectorized.execution.enabled=false;

set hive.enforce.bucketing=true;

4.

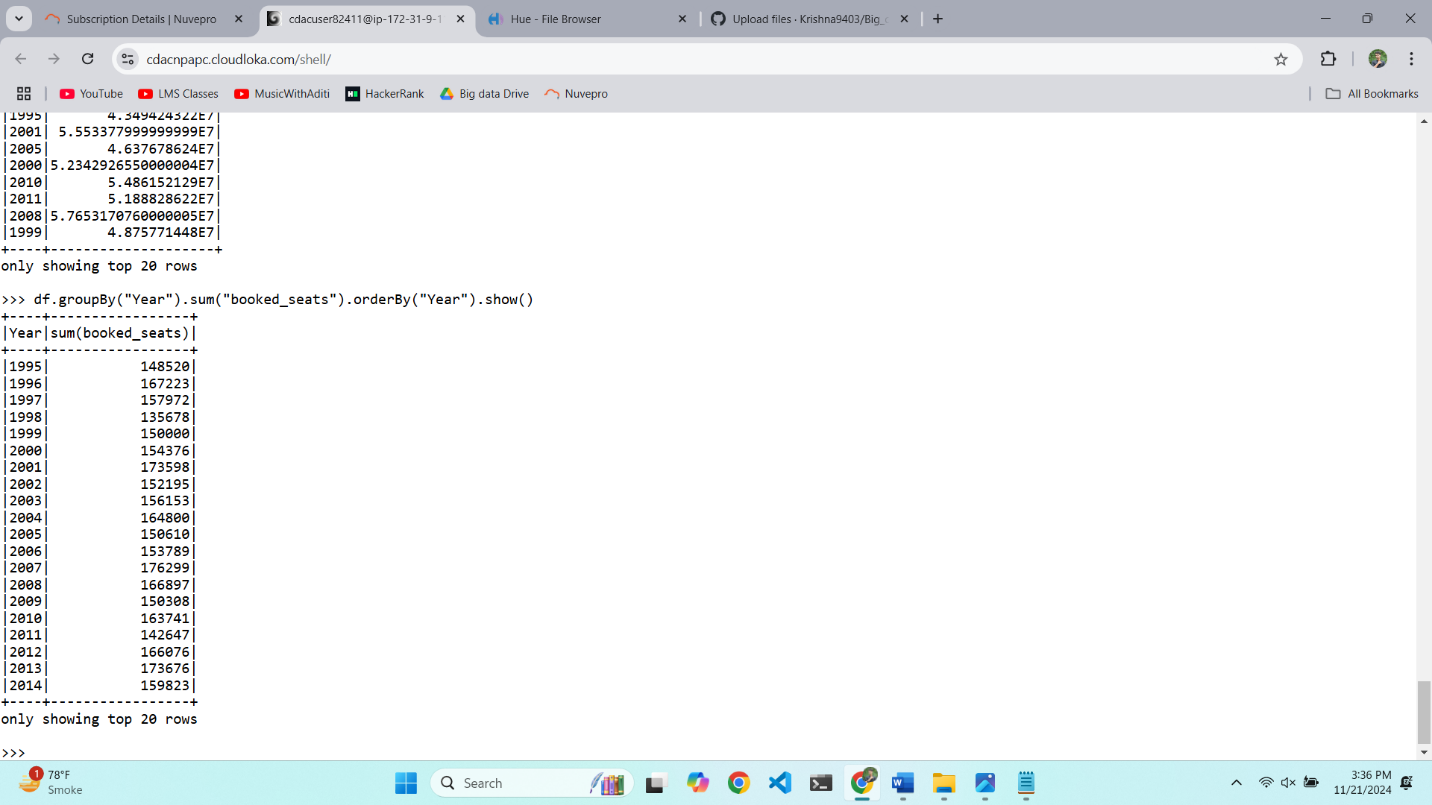
Show partitions src\_air;



Spark

Q1

1. >>> df.groupBy("Quarter").sum("booked\_seats").show()

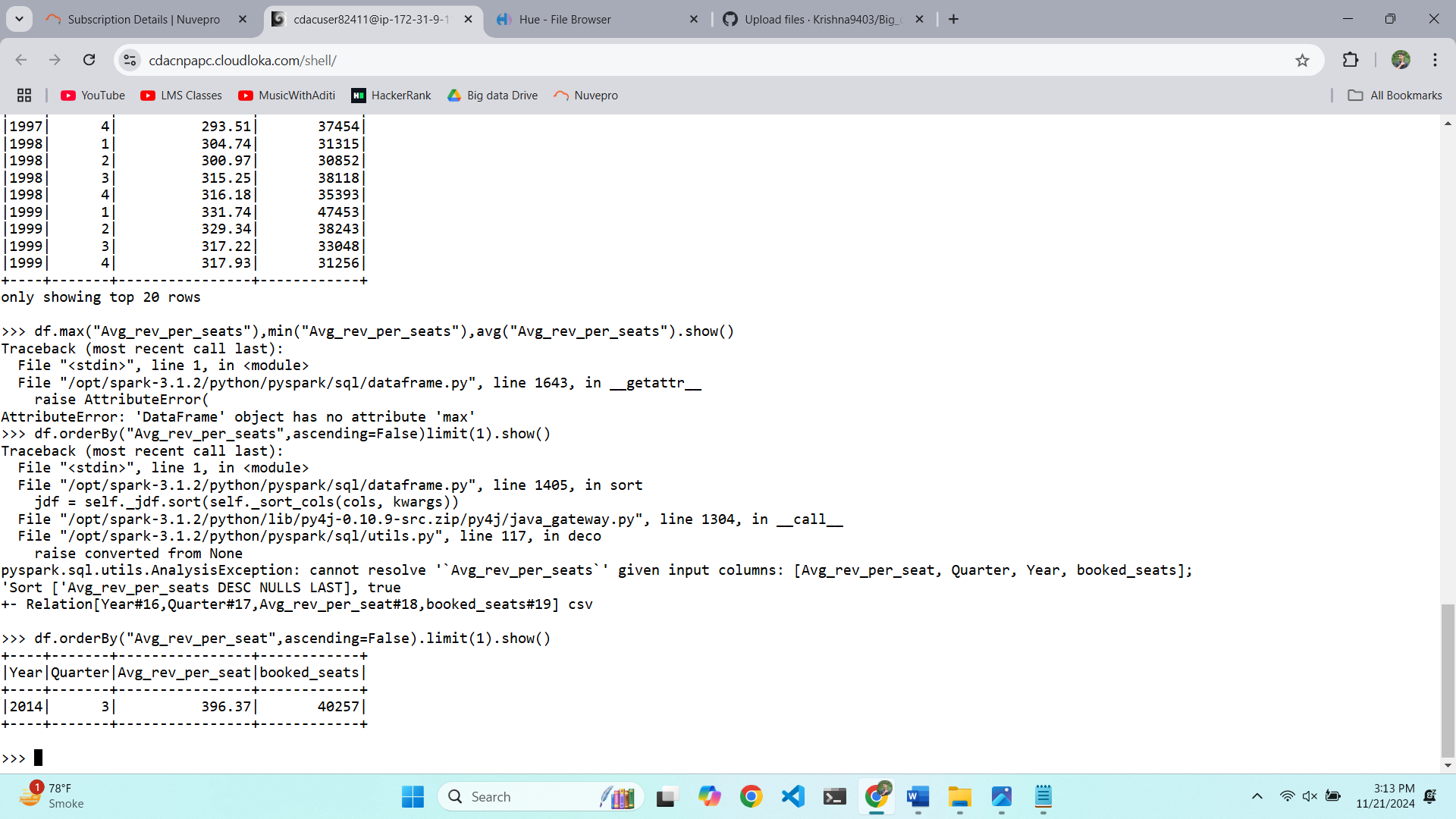


Q2.

1.

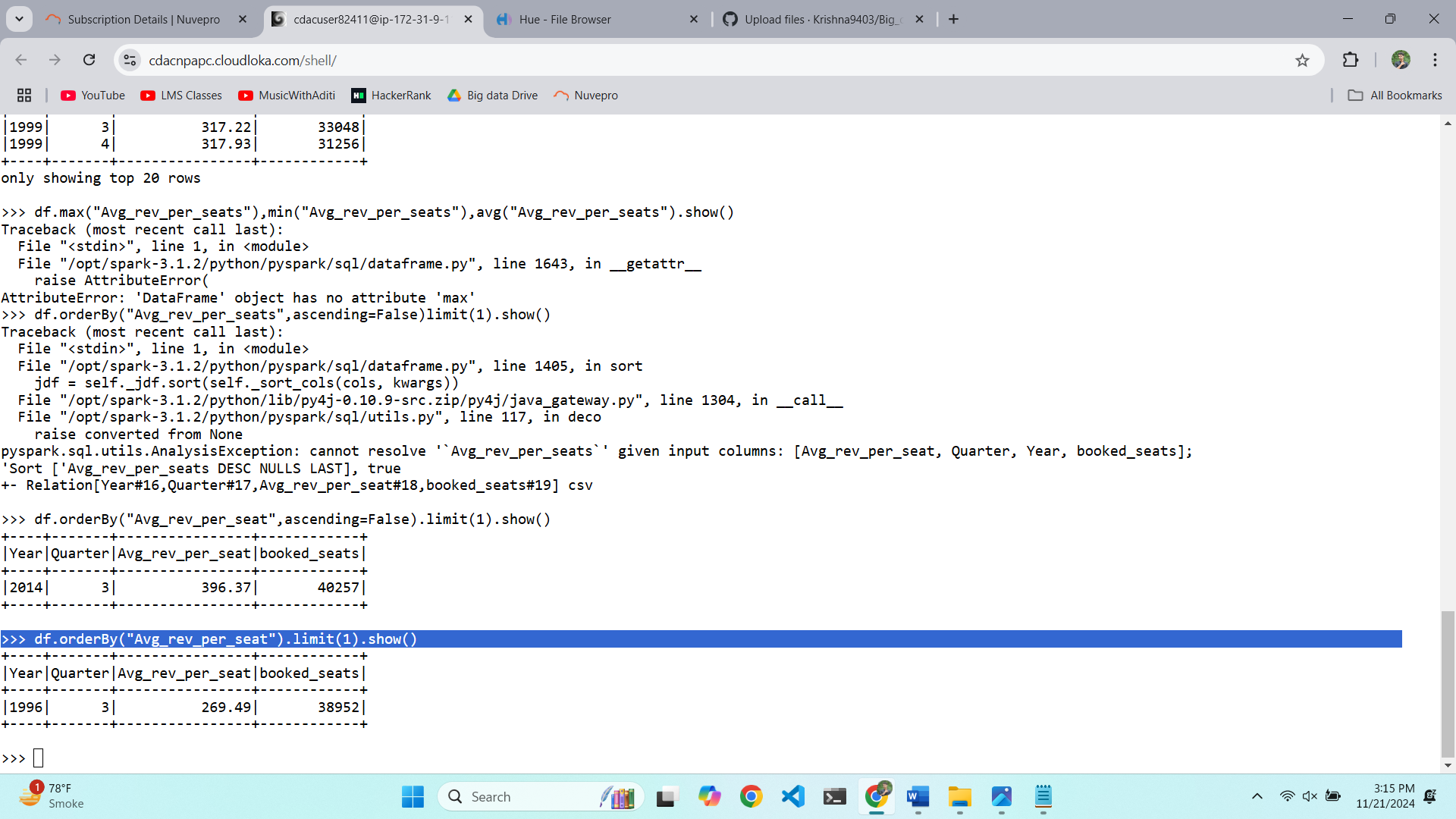
Max of Average revenue per seat

>>> df.orderBy("Avg\_rev\_per\_seat",ascending=False).limit(1).show()



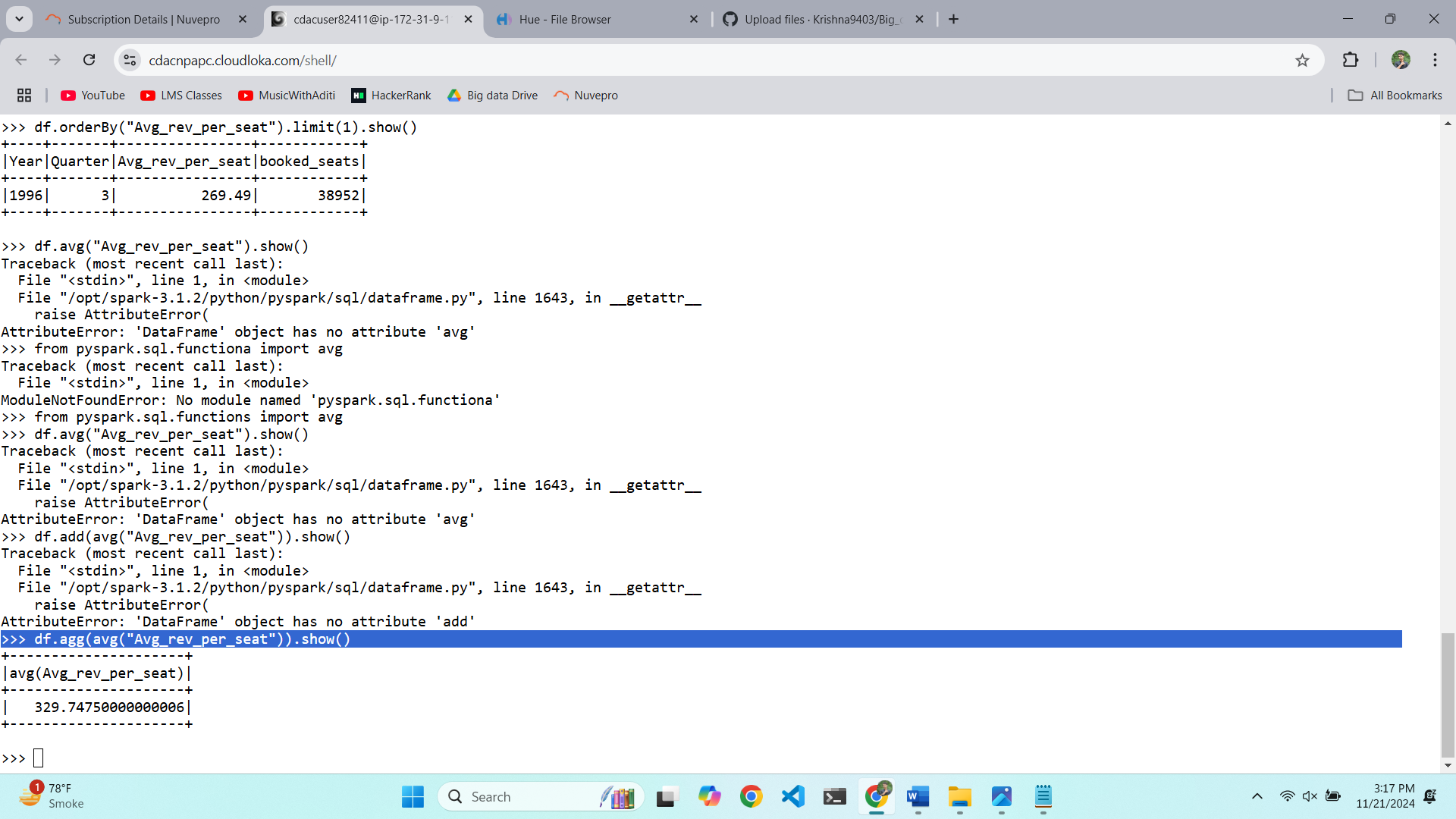
Max of Average revenue per seat

>>> df.orderBy("Avg\_rev\_per\_seat").limit(1).show()



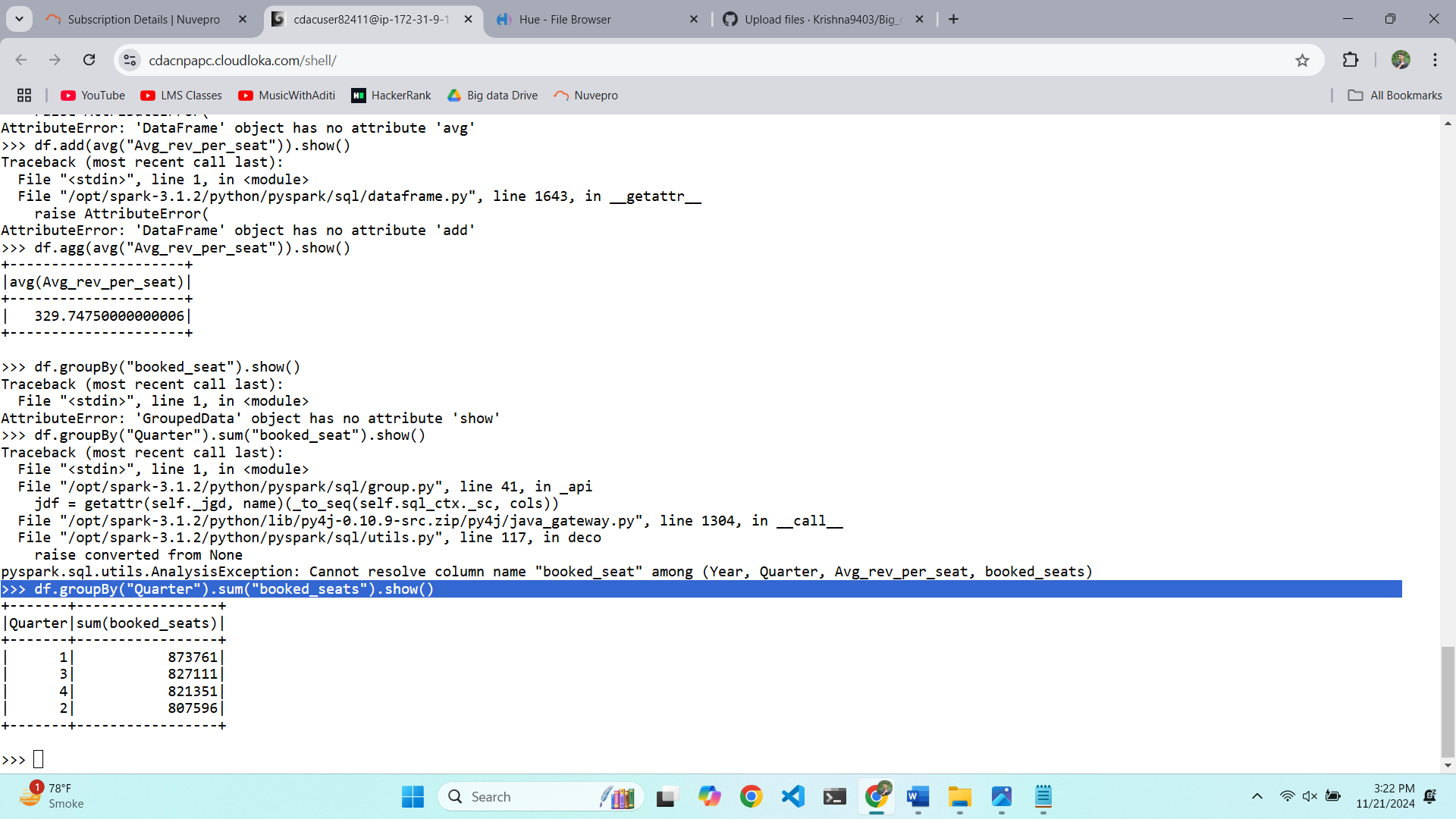
Max of Average revenue per seat

>>> df.agg(avg("Avg\_rev\_per\_seat")).show()



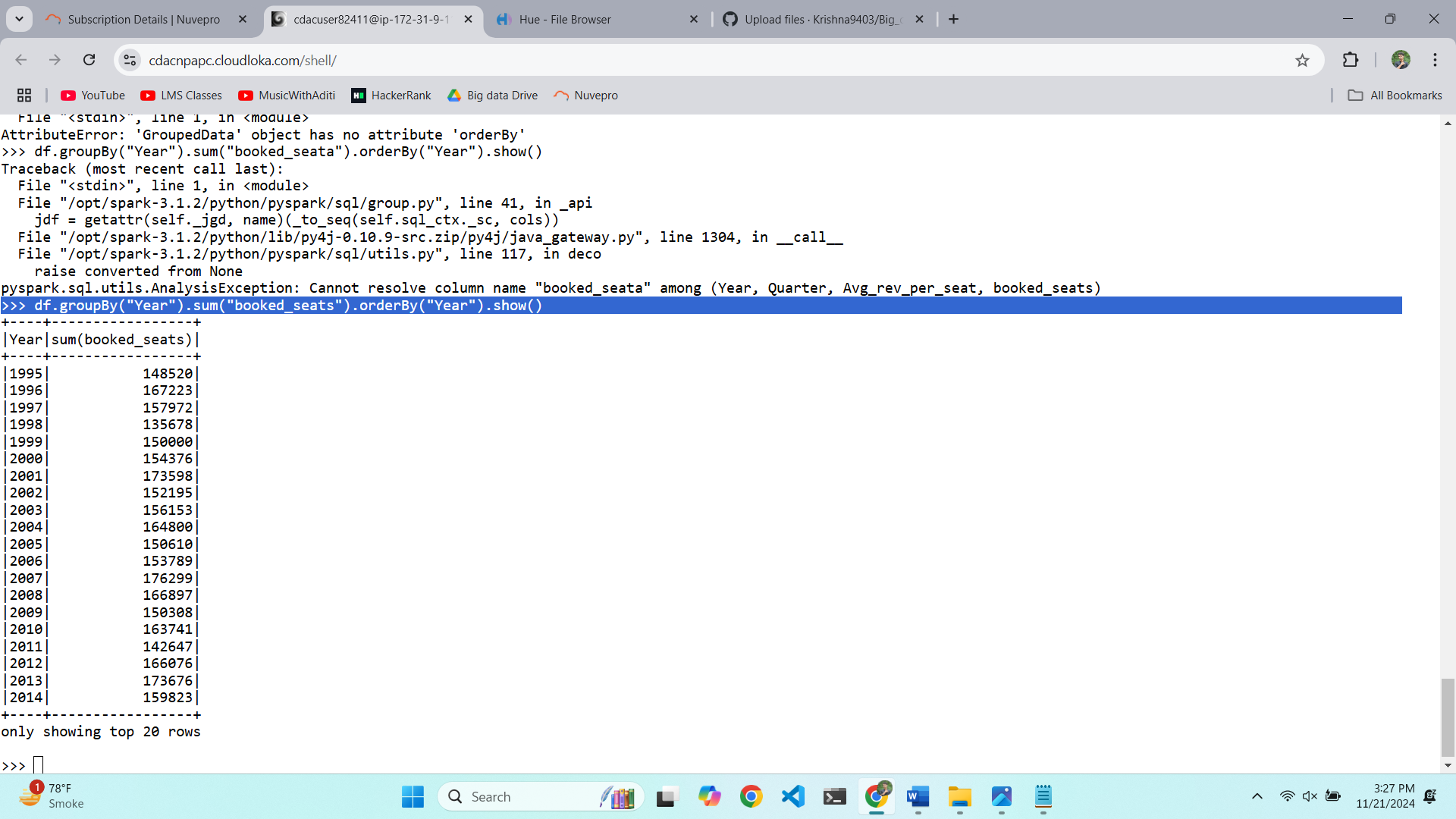
3.

>>> df.groupBy("Quarter").sum("booked\_seats").show()

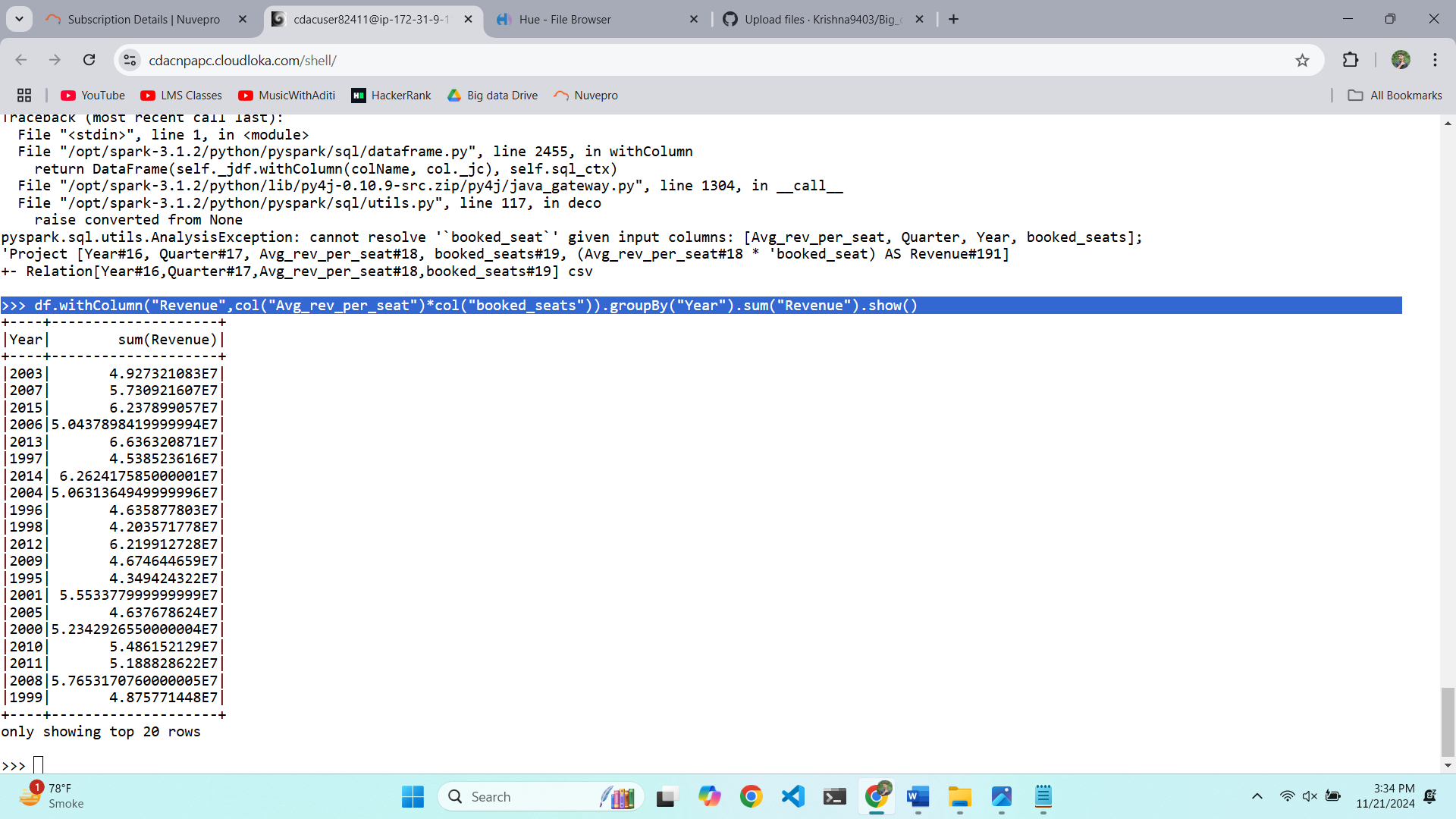


4.

>>> df.groupBy("Year").sum("booked\_seats").orderBy("Year").show()



5. >>> df.withColumn("Revenue",col("Avg\_rev\_per\_seat")\*col("booked\_seats")).groupBy("Year").sum("Revenue").show()



SELECT equipment\_id, COUNT(DISTINCT route\_id) AS route\_count FROM equipment\_usage GROUP BY equipment\_id ORDER BY route\_count DESC