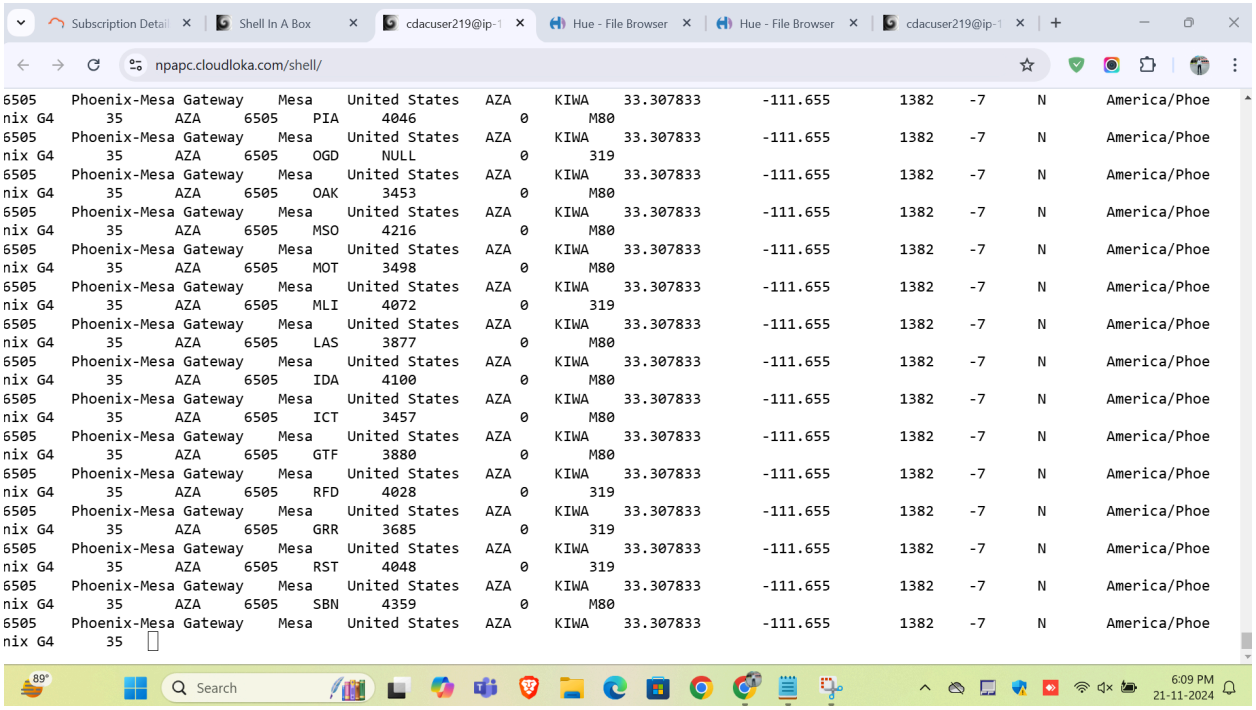
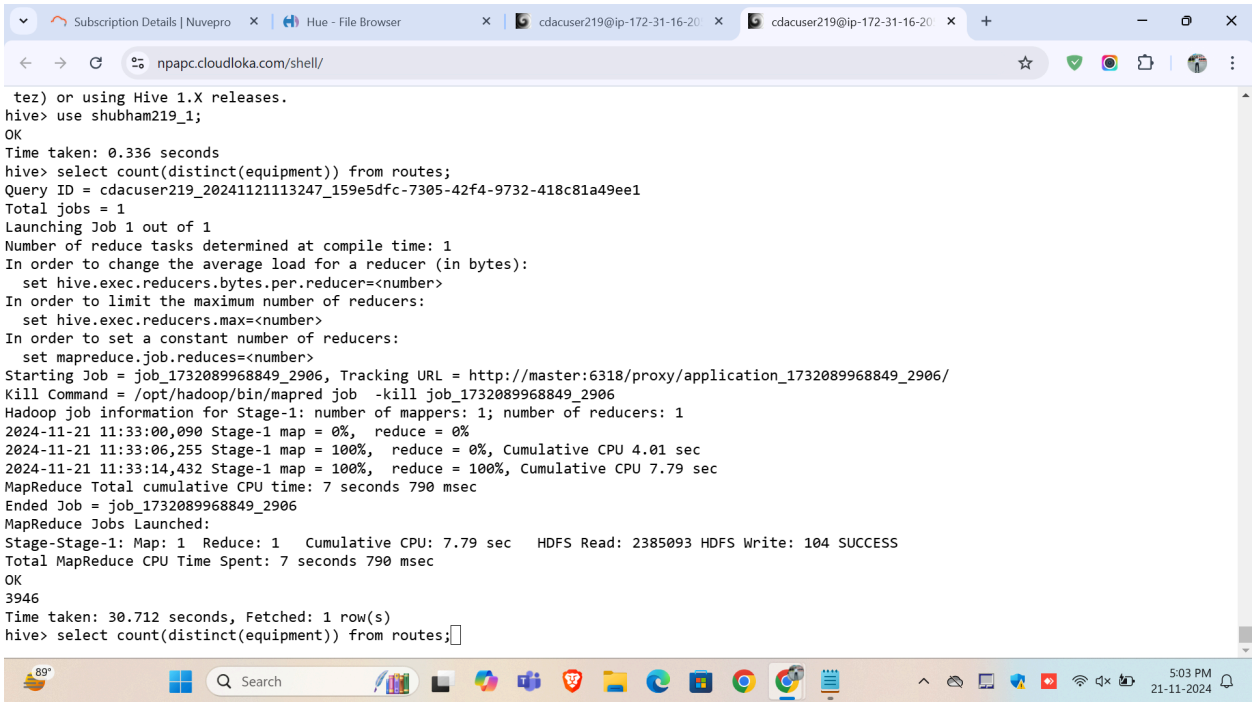
QUESTION 1 ;

1) `select * from airport_1 a  join routes r on a.iata = r.source_iata;`



2) `select count(airlne) from routes group by (source_iata,destination_iata) order by (source_iata,destination_iata)limit 3;`

3) `hive> select count(distinct(equipment)) from routes;`



OUTPUT : 3946

Question 2 :

1)hive (shubham219_1)> Create table route_parittion(airline string , airline_id int , source_iata string , source_id string , destination_id string , codeshare string ,stops int) partitioned by(destination_iata string) row format delimited fields terminated by "," stored as textfile;

2) insert overwrite table route_parittion select rp.airlne,rp.airline_id ,rp.source_iata, rp.source_id , rp.destination_id,rp.codeshare , rp.stops , rp.destination_iata from routes rp;

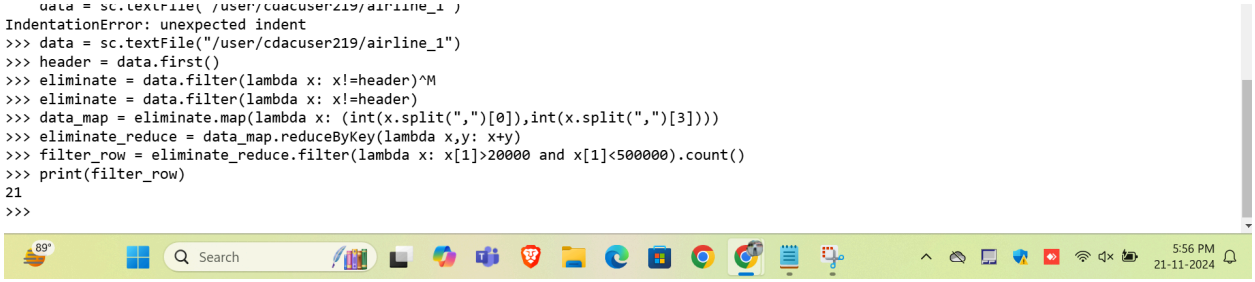
3) select * from route_parittion where destination_iata = "ORD";

4)

SPARK

Question 1 :

```
>>> data = sc.textFile("/user/cdacuser219/airline_1")
>>> header = data.first()
>>> eliminate = data.filter(lambda x: x!=header)
>>> data_map = eliminate.map(lambda x:
(int(x.split(",")[0]),int(x.split(",")[3])))
>>> for x in data_map.collect():
...     print(x)
...
>>> eliminate_reduce = data_map.reduceByKey(lambda x,y: x+y)
>>> for x in eliminate_reduce.collect():
...     print(x)
>>> filter_row = eliminate_reduce.filter(lambda x: x[1]>20000
and x[1]<50000).count()
>>> print(filter_row)
21
```




```
>>> data = sc.textFile("/user/cdacuser219/airline_1")
>>> header = data.first()
>>> eliminate = data.filter(lambda x: x!=header)^M
>>> eliminate = data.filter(lambda x: x!=header)
>>> data_map = eliminate.map(lambda x:
(int(x.split(",")[0]),int(x.split(",")[3])))
>>> eliminate_reduce = data_map.reduceByKey(lambda x,y: x+y)
>>> filter_row = eliminate_reduce.filter(lambda x: x[1]>20000
and x[1]<500000).count()
>>> print(filter_row)
21
>>> data_map = eliminate.map(lambda x:
(int(x.split(",")[1]),int(x.split(",")[0])))
>>> for x in data_map.take(5):
...     print(x)
...
```

```
      (1, 1995)
      (2, 1995)
      (3, 1995)
      (4, 1995)
      (1, 1996)


Part 2
Question 1
>>> spark = SparkSession.builder.appName("exam").getOrCreate()
>>> data = spark.read.format("csv").option("header",
True).option("inferSchema", True).load("/user/cdacuser219/airline_1")
>>> data.show()
+----+-------+---------------+------------+
|Year|Quarter|Avg_rev_per_seat|booked_seats|
+----+-------+---------------+------------+
|1995|      1|           296.9|       46561|
|1995|      2|           296.8|       37443|
|1995|      3|          287.51|       34128|
|1995|      4|          287.78|       30388|
|1996|      1|          283.97|       47808|
|1996|      2|          275.78|       43020|
|1996|      3|          269.49|       38952|
|1996|      4|          278.33|       37443|
|1997|      1|           283.4|       35067|
|1997|      2|          289.44|       46565|
|1997|      3|          282.27|       38886|
|1997|      4|          293.51|       37454|
|1998|      1|          304.74|       31315|
|1998|      2|          300.97|       30852|
|1998|      3|          315.25|       38118|
|1998|      4|          316.18|       35393|
|1999|      1|          331.74|       47453|
|1999|      2|          329.34|       38243|
|1999|      3|          317.22|       33048|
|1999|      4|          317.93|       31256|
+----+-------+---------------+------------+
only showing top 20 rows

>>> data.describe()
DataFrame[summary: string, Year: string, Quarter: string,
Avg_rev_per_seat: string, booked_seats: string]
>>> data.describe().show()
+-------+----------------+----------------+------------------+-----
------------+
```

```
|summary|              Year|          Quarter|  Avg_rev_per_seat|
booked_seats|
+-------+------------------+-----------------+------------------+-----
------------+
|  count|                84|               84|                84|
84|
|   mean|            2005.0|
2.5|329.74750000000006|39640.70238095238|
| stddev|6.091669207609634|1.124748967976346|
32.64232664586615|5424.069182884482|
|    min|              1995|                1|            269.49|
30103|
|    max|              2015|                4|            396.37|
49678|
+-------+------------------+-----------------+------------------+-----
------------+
```

3) data.groupBy('booked_seat').agg({'avg','booked_seats'}).show()