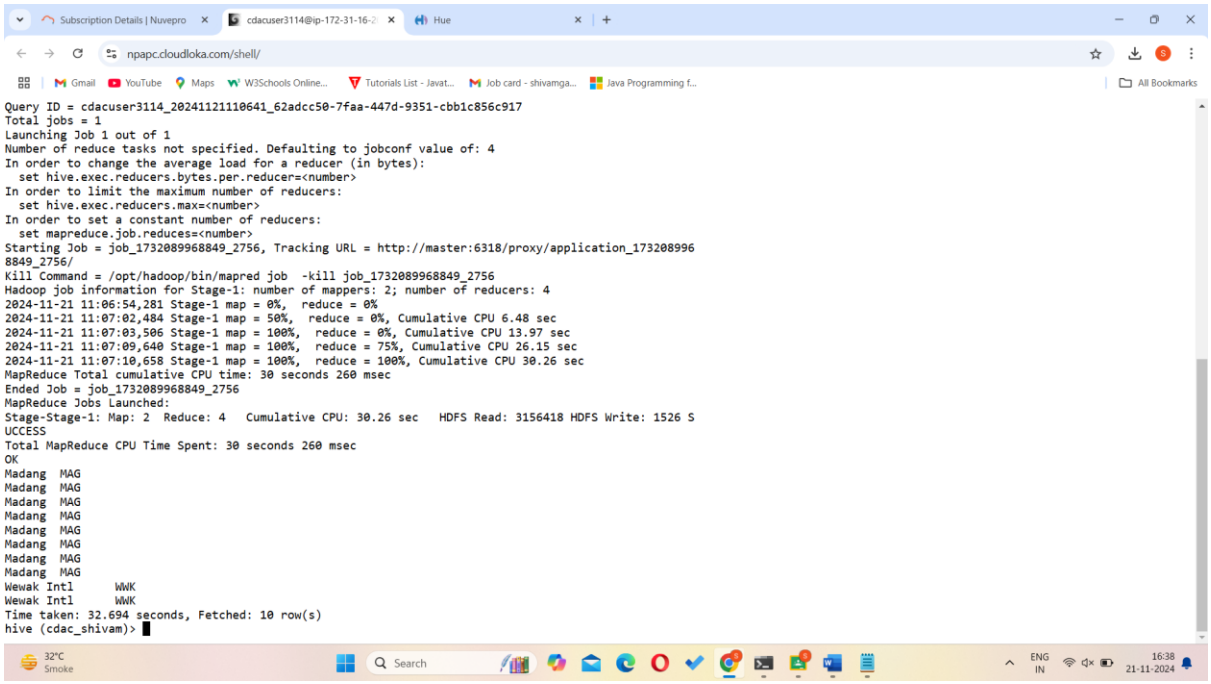
AQ1. select ap.name ,r.src_airport_iata from airports ap

join routes r

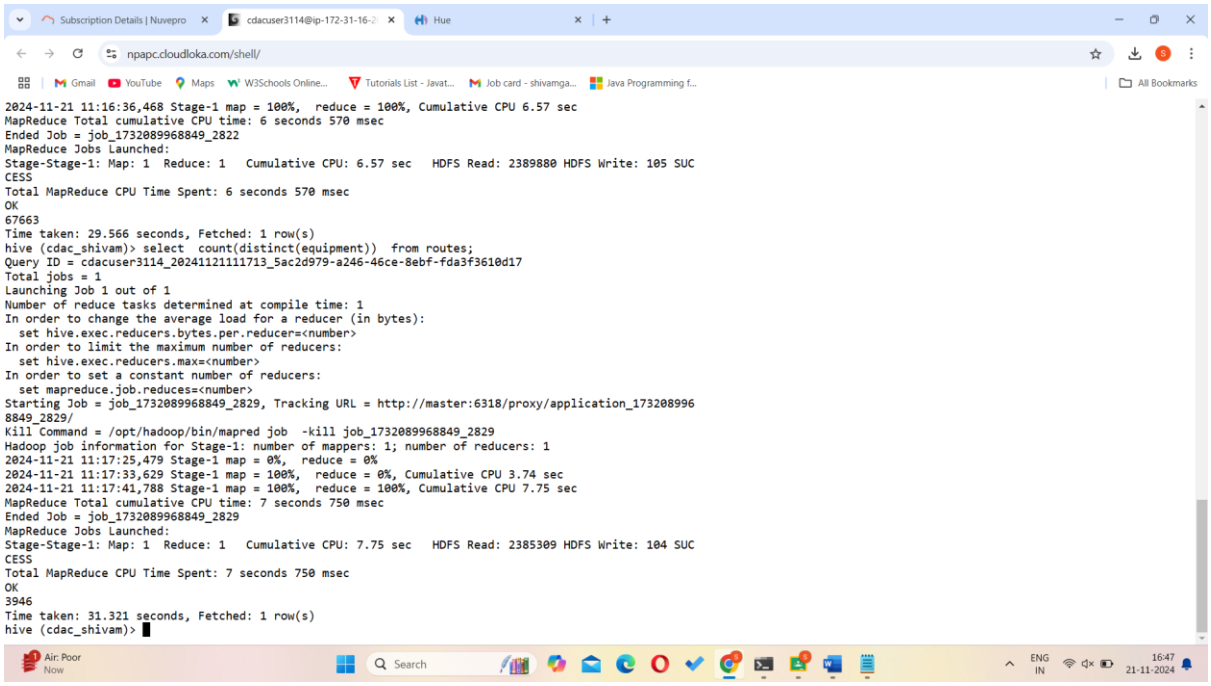
on r.src_airport_id=ap.airport_id

where r.src_airport_iata != dest_airport_iata limit 10;



AQ2.

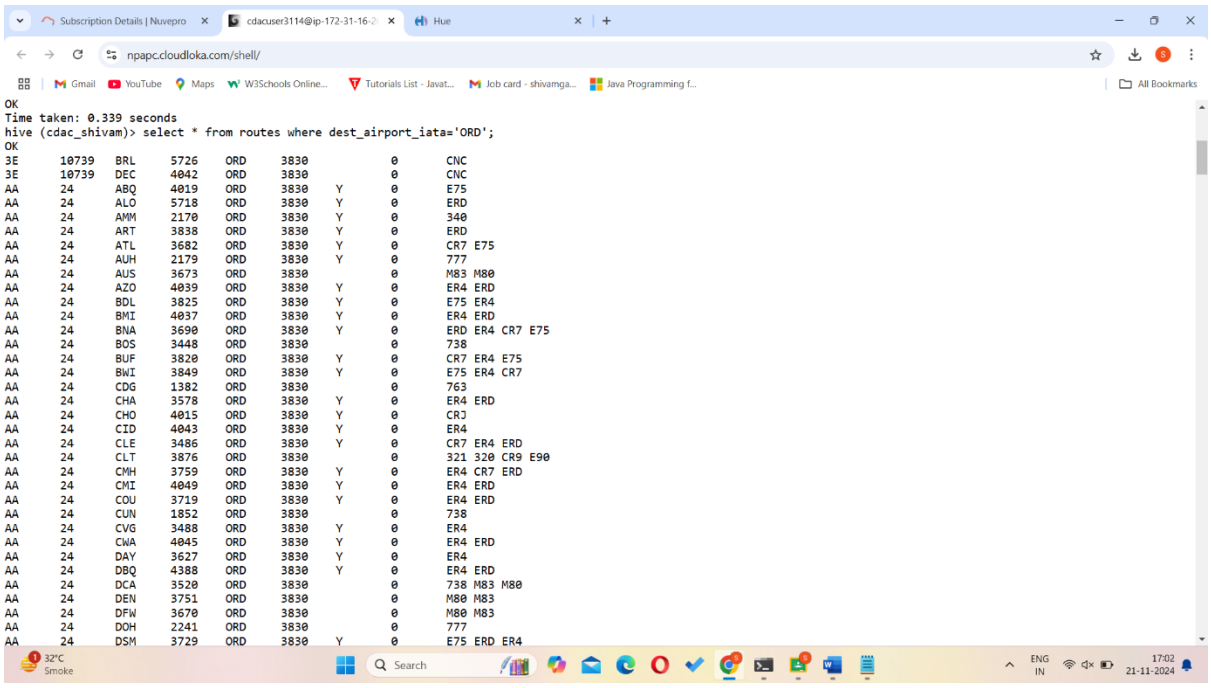
AQ3. select count(distinct(equipment)) from routes;



B.Q1

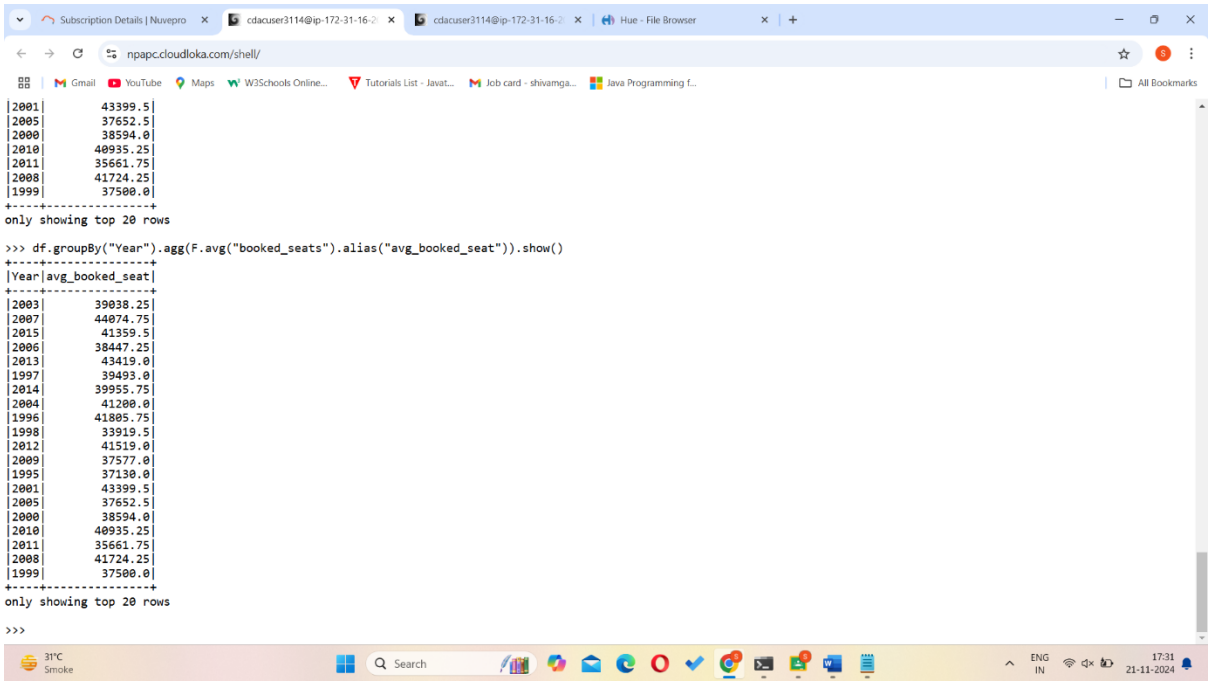
B.Q2

BQ3. select * from routes where dest_airport_iata='ORD';



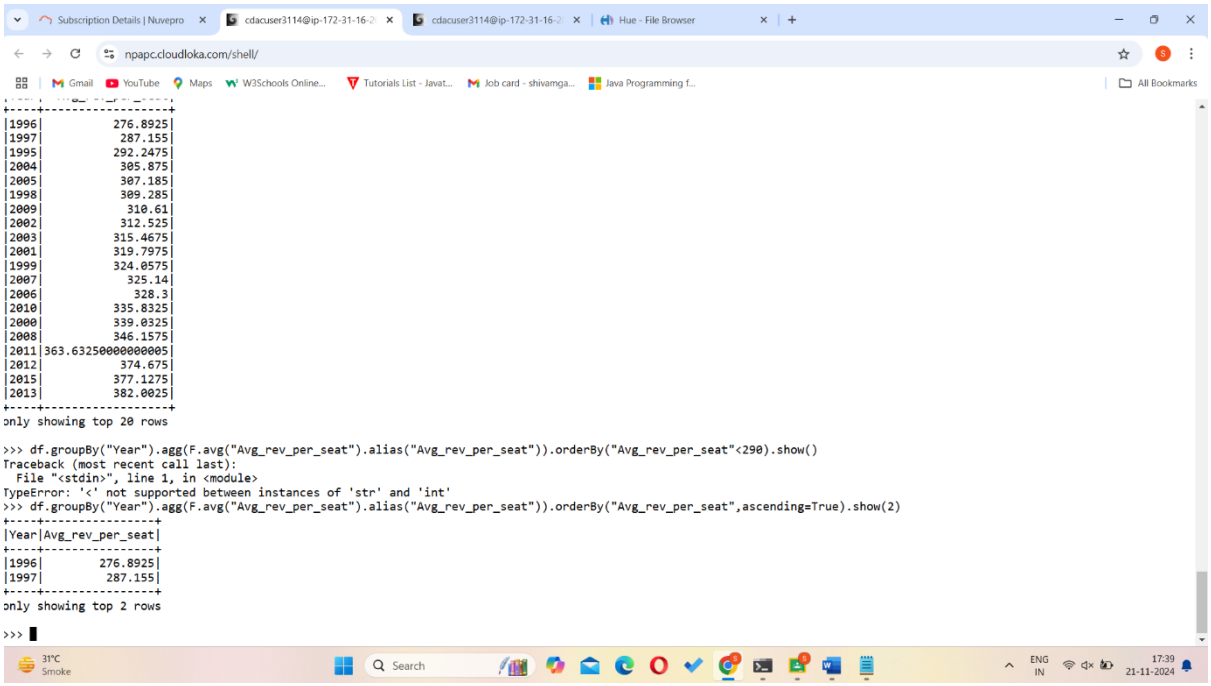
BQ4.

Spark Q2 -1




```
>>> df.groupBy("Year").agg(F.min("booked_seats").alias("min_booked_seat")).orderBy("min_booked_seat",ascending=True).show(10)
+----+---------------+
|Year|min_booked_seat|
+----+---------------+
|2000|          30103|
|1995|          30388|
|2011|          30562|
|1998|          30852|
|2004|          30877|
|1999|          31256|
|2005|          32003|
|2002|          32406|
|2009|          32491|
|2006|          32621|
+----+---------------+
only showing top 10 rows

>>> df.groupBy("Year").agg(F.max("booked_seats").alias("max_booked_seat")).orderBy("max_booked_seat",ascending=False).show(10)
+----+---------------+
|Year|max_booked_seat|
+----+---------------+
|2010|          49678|
|2013|          49143|
|2004|          49022|
|2000|          48159|
|2014|          47928|
|1996|          47808|
|2007|          47758|
|2005|          47608|
|1999|          47453|
|2008|          46885|
+----+---------------+
only showing top 10 rows

>>>
```

Q2-2

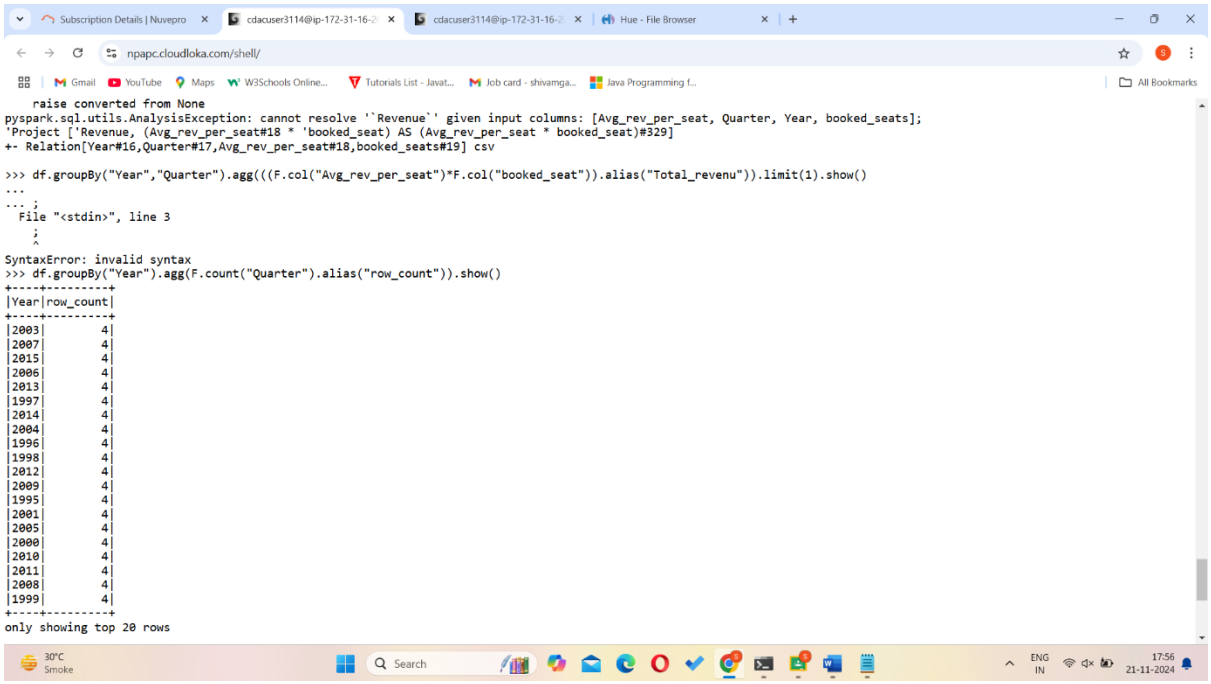


```
+----+------------------+
|1996|          276.8925|
|1997|           287.155|
|1995|          292.2475|
|2004|           305.875|
|2005|           307.185|
|1998|           309.285|
|2009|            310.61|
|2002|           312.525|
|2003|          315.4675|
|2001|          319.7975|
|1999|          324.0575|
|2007|            325.14|
|2006|             328.3|
|2010|          335.8325|
|2000|          339.0325|
|2008|          346.1575|
|2011|363.63250000000005|
|2012|           374.675|
|2015|          377.1275|
|2013|          382.0025|
+----+------------------+
only showing top 20 rows

>>> df.groupBy("Year").agg(F.avg("Avg_rev_per_seat").alias("Avg_rev_per_seat")).orderBy("Avg_rev_per_seat"<290).show()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: '<' not supported between instances of 'str' and 'int'
>>> df.groupBy("Year").agg(F.avg("Avg_rev_per_seat").alias("Avg_rev_per_seat")).orderBy("Avg_rev_per_seat",ascending=True).show(2)
+----+----------------+
|Year|Avg_rev_per_seat|
+----+----------------+
|1996|        276.8925|
|1997|         287.155|
+----+----------------+
only showing top 2 rows

>>>
```

Q2.3

```
+----+----------------+
only showing top 2 rows

>>> df.groupBy("Quaeter").agg(F.avg("booked_seat").alias("total_booked_seat")).show()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/opt/spark-3.1.2/python/pyspark/sql/group.py", line 118, in agg
    jdf = self._jgd.agg(exprs[0]._jc,
  File "/opt/spark-3.1.2/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py", line 1304, in __call__
  File "/opt/spark-3.1.2/python/pyspark/sql/utils.py", line 117, in deco
    raise converted from None
pyspark.sql.utils.AnalysisException: cannot resolve '`Quaeter`' given input columns: [Avg_rev_per_seat, Quarter, Year, booked_seats];
'Aggregate ['Quaeter], ['Quaeter, avg('booked_seat) AS total_booked_seat#294]
+- Relation[Year#16,Quarter#17,Avg_rev_per_seat#18,booked_seats#19] csv

>>> df.groupBy("Quarter").agg(F.avg("booked_seat").alias("total_booked_seat")).show()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/opt/spark-3.1.2/python/pyspark/sql/group.py", line 118, in agg
    jdf = self._jgd.agg(exprs[0]._jc,
  File "/opt/spark-3.1.2/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py", line 1304, in __call__
  File "/opt/spark-3.1.2/python/pyspark/sql/utils.py", line 117, in deco
    raise converted from None
pyspark.sql.utils.AnalysisException: cannot resolve '`booked_seat`' given input columns: [Avg_rev_per_seat, Quarter, Year, booked_seats];
'Aggregate [Quarter#17], [Quarter#17, avg('booked_seat) AS total_booked_seat#300]
+- Relation[Year#16,Quarter#17,Avg_rev_per_seat#18,booked_seats#19] csv

>>> df.groupBy("Quarter").agg(F.avg("booked_seats").alias("avg_booked_seat")).show()
+-------+-----------------+
|Quarter|   avg_booked_seat|
+-------+-----------------+
|      1|41607.666666666664|
|      3| 39386.23809523809|
|      4| 39111.95238095238|
|      2| 38456.95238095238|
+-------+-----------------+

>>>
```

Q2.4



```
   raise converted from None
pyspark.sql.utils.AnalysisException: cannot resolve '`Revenue`' given input columns: [Avg_rev_per_seat, Quarter, Year, booked_seats];
'Project ['Revenue, (Avg_rev_per_seat#18 * 'booked_seat) AS (Avg_rev_per_seat * booked_seat)#329]
+- Relation[Year#16,Quarter#17,Avg_rev_per_seat#18,booked_seats#19] csv

>>> df.groupBy("Year","Quarter").agg(((F.col("Avg_rev_per_seat")*F.col("booked_seat")).alias("Total_revenu")).limit(1).show()
...
... ;
  File "<stdin>", line 3
    ;
    ^
SyntaxError: invalid syntax
>>> df.groupBy("Year").agg(F.count("Quarter").alias("row_count")).show()
+----+---------+
|Year|row_count|
+----+---------+
|2003|        4|
|2007|        4|
|2015|        4|
|2006|        4|
|2013|        4|
|1997|        4|
|2014|        4|
|2004|        4|
|1996|        4|
|1998|        4|
|2012|        4|
|2009|        4|
|1995|        4|
|2001|        4|
|2005|        4|
|2000|        4|
|2010|        4|
|2011|        4|
|2008|        4|
|1999|        4|
+----+---------+
only showing top 20 rows
```

Q2.5

```
SyntaxError: invalid syntax
>>> df.groupBy("Year","Quarter").agg(((F.col("Avg_rev_per_seat")*F.col("booked_seat")).alias("Total_revenu")).limit(1).show()
... ;
```