

## Hive

1.

```
Subscription Details | Nuvepro x cdacuser87125@ip-172-31-9-1 x Big_data_23 - Google Docs x +
cdacnpapc.cloudloka.com/shell/
ip-172-31-9-116 login: cdacuser87125
Password:
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-1072-aws x86_64)

System information as of Thu Nov 21 08:30:47 UTC 2024
System load: 1.31 Processes: 419
Usage of /: 85.4% of 30.83GB Users logged in: 24
Memory usage: 16% IPv4 address for ens5: 172.31.9.116
Swap usage: 0%

=> / is using 85.4% of 30.83GB
=> There is 1 zombie process.

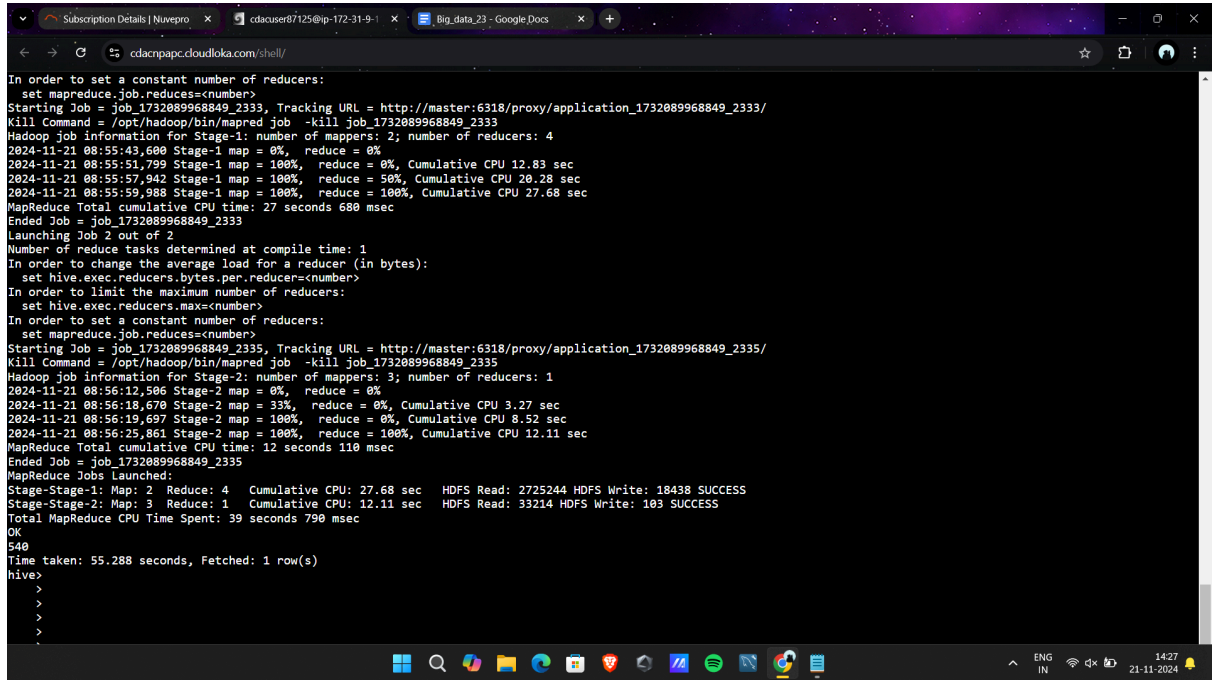
Last login: Thu Nov 21 08:28:59 UTC 2024 from 127.0.0.1, 14.139.112.69 on pts/18
cdacuser87125@ip-172-31-9-116:~$ sc.stop()
> ;
-bash: syntax error near unexpected token `;'
cdacuser87125@ip-172-31-9-116:~$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hive-3.1.1/lib/log4j-slf4j-impl-2.10.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Hive Session ID = d87fe638-a9ba-4a61-a9eb-b26985a40ae5

Logging initialized using configuration in jar:file:/opt/hive-3.1.1/lib/hive-common-3.1.1.jar!/hive-log4j2.properties Async: true
Hive Session ID = 1f2fe991-9eed-4e6c-ac8b-645b77486e1c
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
hive> use hd;
OK
Time taken: 0.679 seconds
hive> █
```

2.

3

```
select count(distinct(a.name)) from airlines a
join routes r on a.a_id =r.a_id ;
```

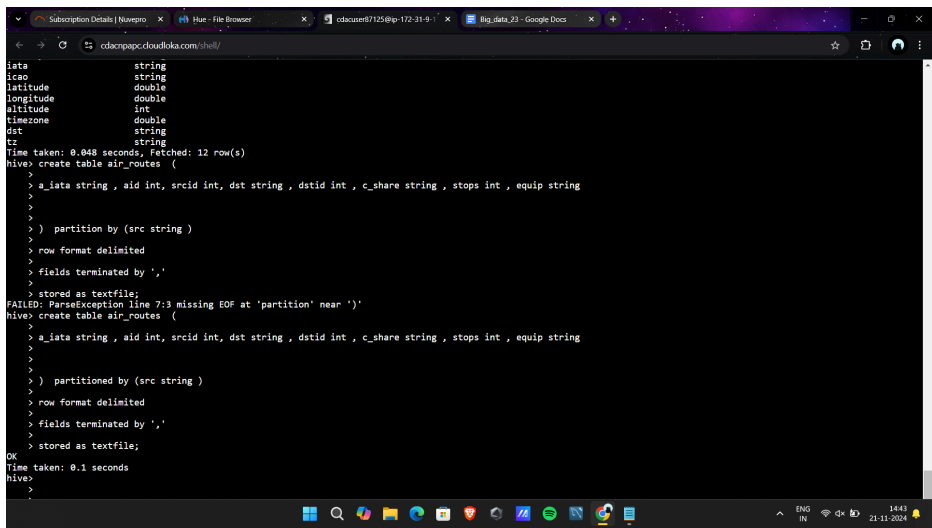


The screenshot shows a terminal window with a dark background. The top of the window displays browser tabs for 'Subscription Details | Nuvepro', 'cdacuser@7125@ip-172-31-9-1', and 'Big\_data\_23 - Google Docs'. The terminal output shows the execution of a Hive query. It starts with instructions on how to set the number of reducers for a MapReduce job. Then, it shows the execution of a Hive job (job\_1732089968849\_2333). The logs include the tracking URL, kill command, and Hadoop job information for Stage-1. It shows the progress of the map and reduce tasks, including cumulative CPU time. The job is then launched, and the logs show the progress of Stage-2. Finally, the job is completed, and the results are fetched. The terminal output is as follows:

```
In order to set a constant number of reducers:
set mapreduce.job.reduces=<number>
Starting Job = job_1732089968849_2333, Tracking URL = http://master:6318/proxy/application_1732089968849_2333/
Kill Command = /opt/hadoop/bin/mapred job -kill job_1732089968849_2333
Hadoop job information for Stage-1: number of mappers: 2; number of reducers: 4
2024-11-21 08:55:43,600 Stage-1 map = 0%, reduce = 0%
2024-11-21 08:55:51,799 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 12.83 sec
2024-11-21 08:55:57,942 Stage-1 map = 100%, reduce = 50%, Cumulative CPU 20.28 sec
2024-11-21 08:55:59,988 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 27.68 sec
MapReduce Total cumulative CPU time: 27 seconds 680 msec
Ended Job = job_1732089968849_2333
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
set mapreduce.job.reduces=<number>
Starting Job = job_1732089968849_2335, Tracking URL = http://master:6318/proxy/application_1732089968849_2335/
Kill Command = /opt/hadoop/bin/mapred job -kill job_1732089968849_2335
Hadoop job information for Stage-2: number of mappers: 3; number of reducers: 1
2024-11-21 08:56:12,506 Stage-2 map = 0%, reduce = 0%
2024-11-21 08:56:18,670 Stage-2 map = 33%, reduce = 0%, Cumulative CPU 3.27 sec
2024-11-21 08:56:19,697 Stage-2 map = 100%, reduce = 0%, Cumulative CPU 8.52 sec
2024-11-21 08:56:25,861 Stage-2 map = 100%, reduce = 100%, Cumulative CPU 12.11 sec
MapReduce Total cumulative CPU time: 12 seconds 110 msec
Ended Job = job_1732089968849_2335
MapReduce Jobs Launched:
Stage-Stage-1: Map: 2 Reduce: 4 Cumulative CPU: 27.68 sec HDFS Read: 2725244 HDFS Write: 18438 SUCCESS
Stage-Stage-2: Map: 3 Reduce: 1 Cumulative CPU: 12.11 sec HDFS Read: 33214 HDFS Write: 103 SUCCESS
Total MapReduce CPU Time Spent: 39 seconds 790 msec
OK
540
Time taken: 55.288 seconds, Fetched: 1 row(s)
hive>
>
>
>
```

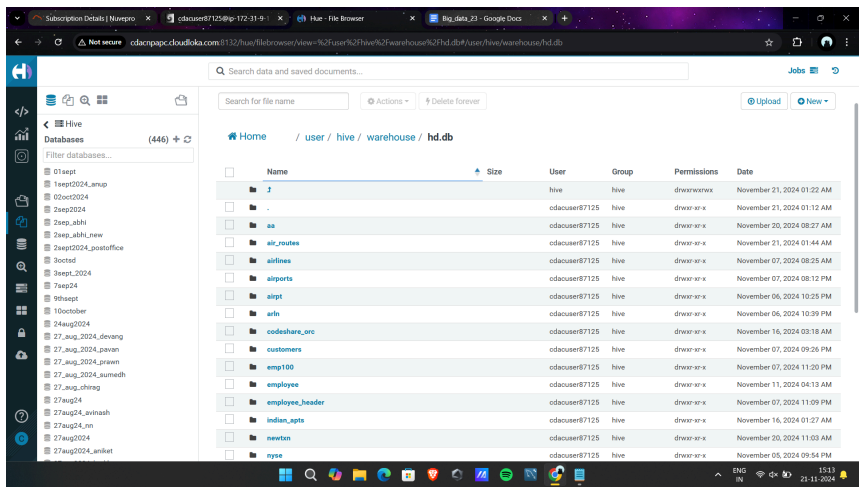
Question2

1.



create table air\_routes (
a\_iata string , aid int, srcid int, dst string , dstid int , c\_share string , stops int , equip
string

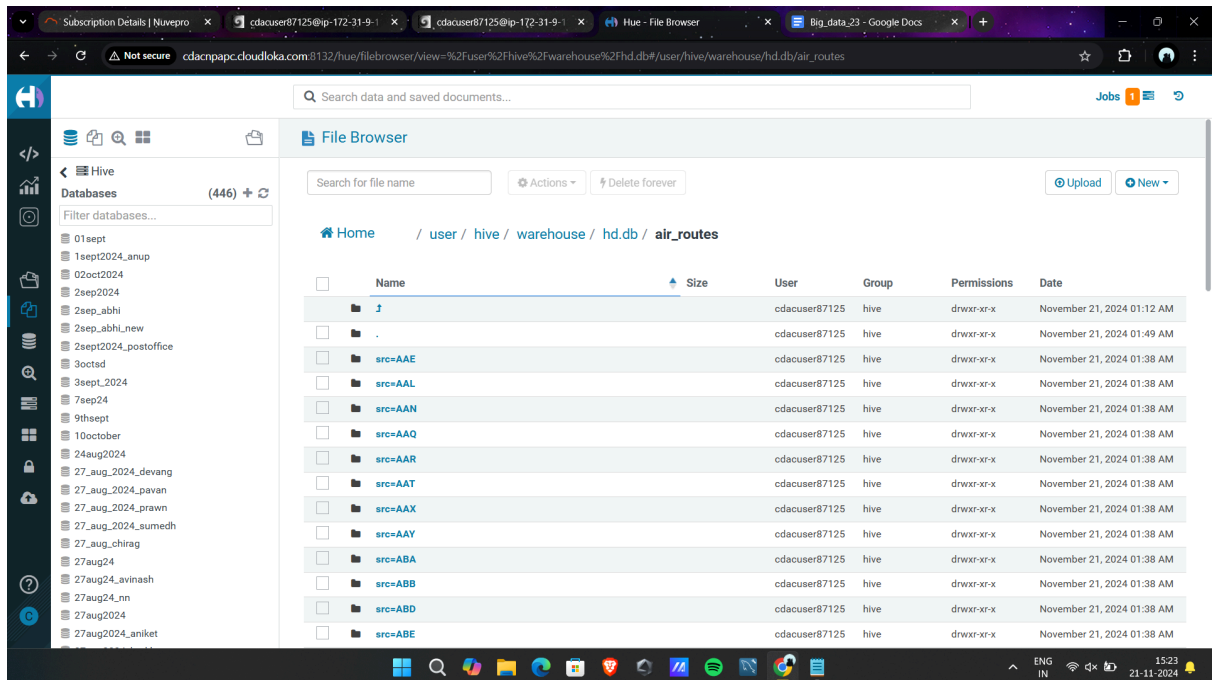
) partitioned by (src string )
row format delimited
fields terminated by ','
stored as textfile;



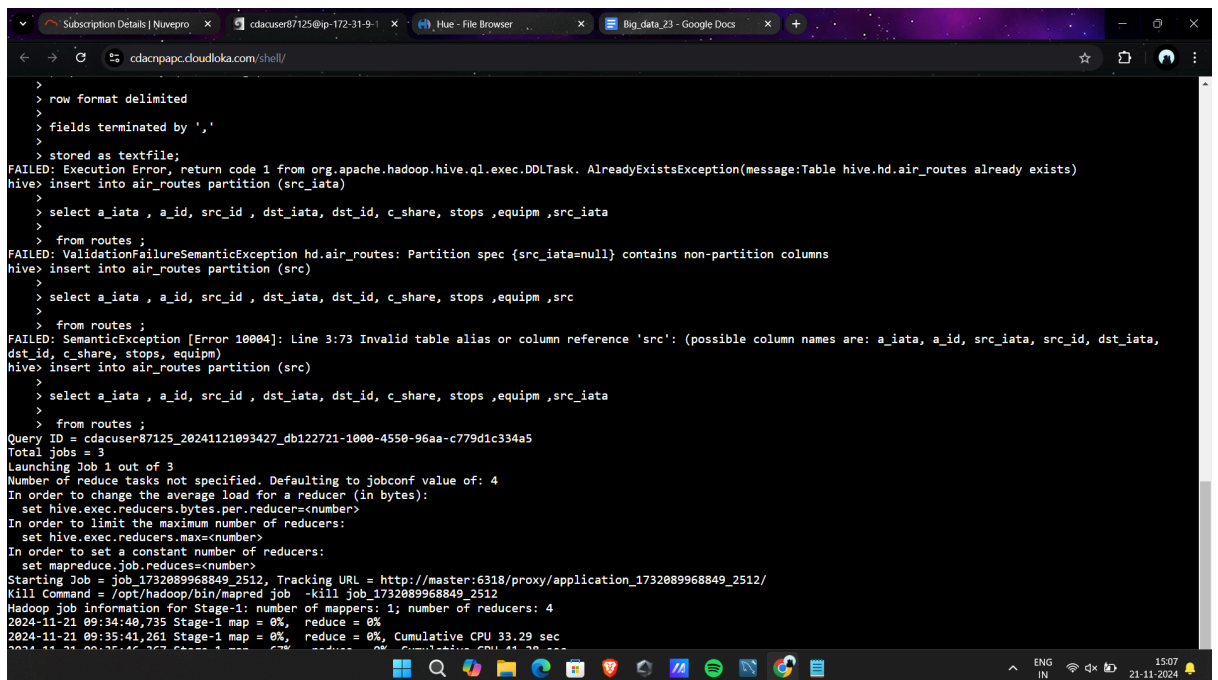
```

insert into air_routes partition (src)
select a_iata , a_id, src_id , dst_iata, dst_id, c_share, stops ,equipm ,src_iata
from routes ;

```



## Hive lagged



2.

```
insert into air_routes select * from routes where src_iata="JFK" ;
```

3.

```
Select src , dst from air_routes where src='LAX' limit 10;
```

4.

```
explain air_routes;
```

## Spark

### Question 1

1.

```
aird=sc.textfile("/user/cdacuser87125/spark/airline.csv")  
aird.count()
```

```
header=aird.first()  
el=aird.filter(lambda a :a != header)
```

```
split=el.map(lambda a:a.split(","))
```

```
aa=split.map(lambda a: str(a[0])+" "+ a[1], float(a[2]) )
```

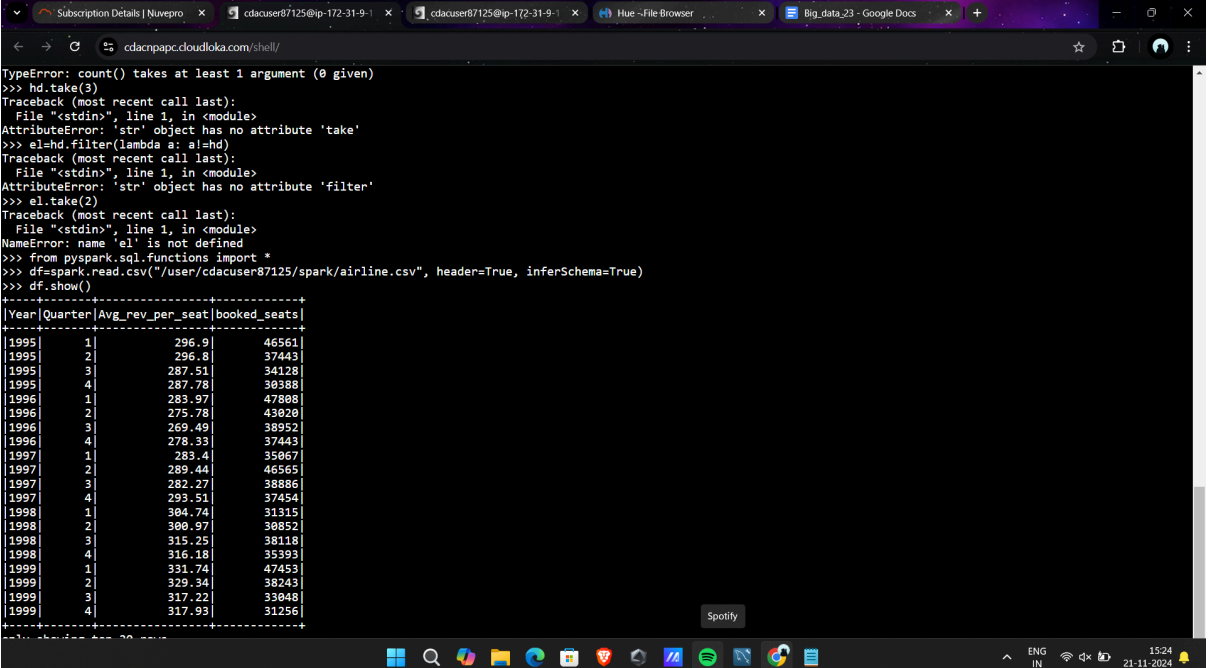
```
arrange=aa.reduceByKey( lambda a,b :a+b)
```

```
main=aa.sortByKey(lambda a: a[0])
```

## Question 2

1.

```
>>> from pyspark.sql.functions import *
>>>
df=spark.read.csv("/user/cdacuser87125/spark/airline.csv",
header=True, inferSchema=True)
```



```
TypeError: count() takes at least 1 argument (0 given)
>>> hd.take(3)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'str' object has no attribute 'take'
>>> el=hd.filter(lambda a: a!=hd)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'str' object has no attribute 'filter'
>>> el.take(2)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'el' is not defined
>>> from pyspark.sql.functions import *
>>> df=spark.read.csv("/user/cdacuser87125/spark/airline.csv", header=True, inferSchema=True)
>>> df.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|
-----+-----+
+-----+
|1999|4|317.93|31256|
+-----+
```

```
Subscription Details | Nuvepro x cdacuser87125@ip-172-31-9-1 x cdacuser87125@ip-172-31-9-1 x Hue -File Browser x Big_data_23 - Google Docs x
cdacnpapc.cloudloka.com/shell/

|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

>>> df.select(min("Avg_rev_per_seat"))
DataFrame[min(Avg_rev_per_seat): double]
>>> df.select(max("Avg_rev_per_seat"))
DataFrame[max(Avg_rev_per_seat): double]
>>> df["Avg_rev_per_seat"].min()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'Column' object is not callable
>>> df["Avg_rev_per_seat"].max()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'Column' object is not callable
>>> df.groupBy("Year").agg(col("Avg_rev_per_seat") * col("booked_seats")).alias("total_rev").show()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can't multiply sequence by non-int of type 'function'
>>> df.groupBy("Year").agg(col("Avg_rev_per_seat") * ("booked_seats")).alias("total_rev").show()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can't multiply sequence by non-int of type 'function'
>>> df.groupBy("Year").agg(("Avg_rev_per_seat") * ("booked_seats")).alias("total_rev").show()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can't multiply sequence by non-int of type 'str'
>>> df.groupBy("Quarter").sum("booked_seats").show()
  File "<stdin>", line 1
    df.groupBy("Quarter").sum("booked_seats").show()
    ^
SyntaxError: invalid syntax
```

Connect

2.

```
df.filter("Avg_rev_per_seat" > 290)
df.agg(count("Avg_rev_per_seat"))
```

3.

```
df.groupBy("Year").sum("booked_seats")
```

4.

```
df.select("Year").distinct()
```