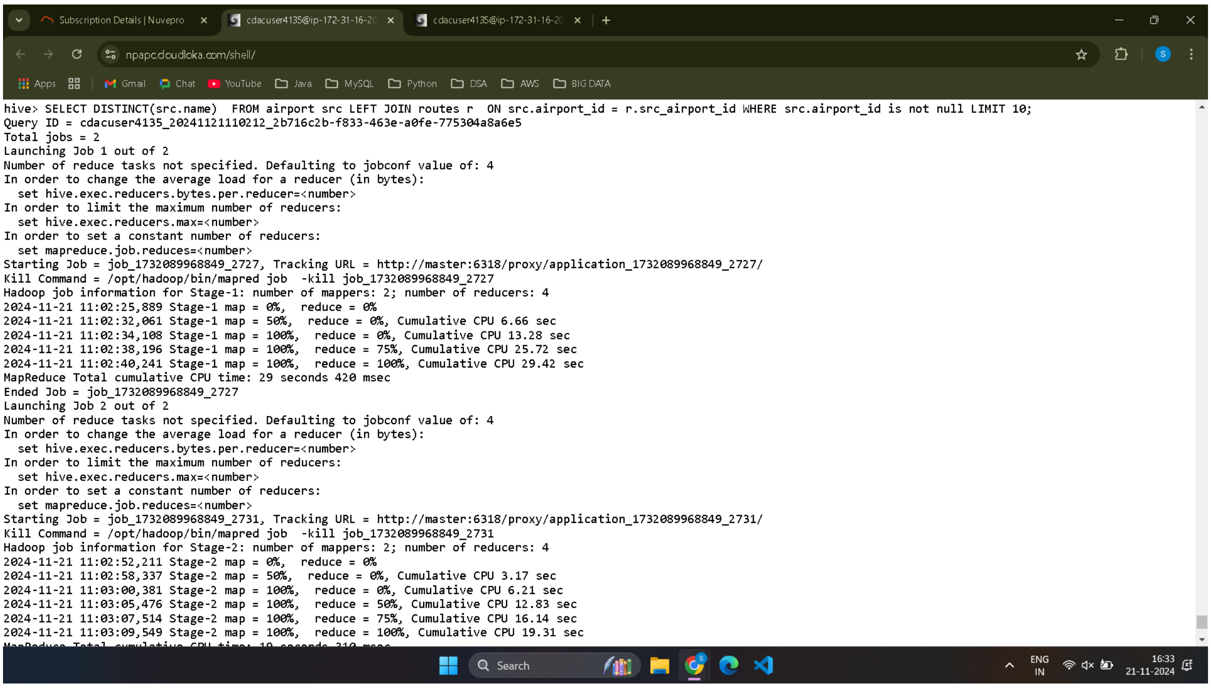
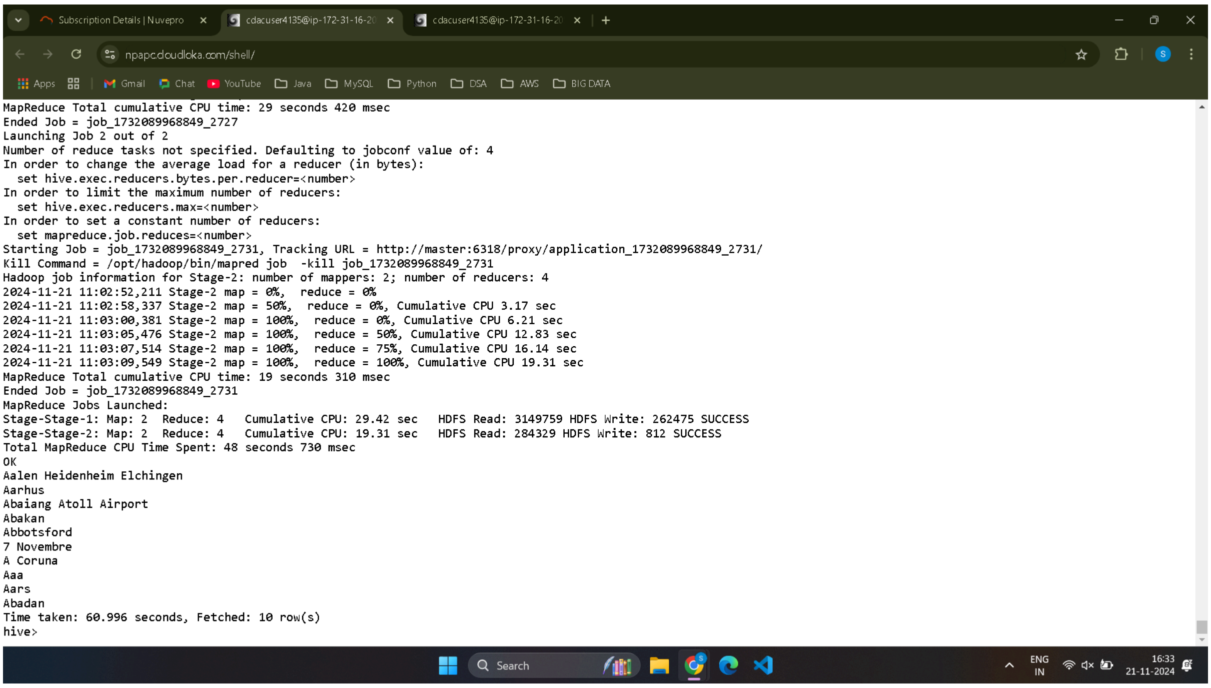
HIVE:
Question 1.

1. FInd the airports that are only listed as sources but not as destinations in the routes table.

```
SELECT
       DISTINCT(src.name)
FROM airport src
LEFT JOIN routes r
ON src.airport_id = r.src_airport_id
WHERE src.airport_id IS NOT NULL;
```


Subscription Details | Nuvepro
cdacuser4135@ip-172-31-16-20
cdacuser4135@ip-172-31-16-20
npapc.cloudloka.com/shell/
Apps
Gmail
Chat
YouTube
Java
MySQL
Python
DSA
AWS
BIG DATA
Search
ENG
IN
16:33
21-11-2024

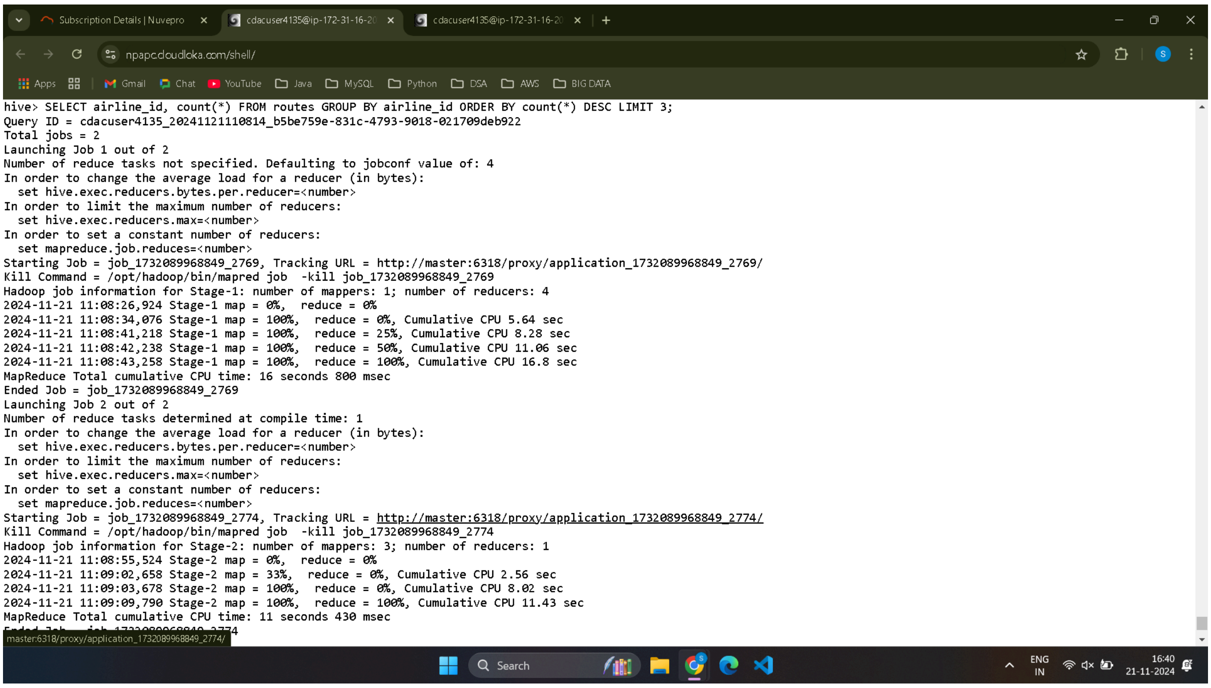
```
hive> SELECT DISTINCT(src.name)  FROM airport src LEFT JOIN routes r  ON src.airport_id = r.src_airport_id WHERE src.airport_id is not null LIMIT 10;
Query ID = cdacuser4135_20241121110212_2b716c2b-f833-463e-a0fe-775304a8a6e5
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Defaulting to jobconf value of: 4
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_2727, Tracking URL = http://master:6318/proxy/application_1732089968849_2727/
Kill Command = /opt/hadoop/bin/mapred job  -kill job_1732089968849_2727
Hadoop job information for Stage-1: number of mappers: 2; number of reducers: 4
2024-11-21 11:02:25,889 Stage-1 map = 0%,  reduce = 0%
2024-11-21 11:02:32,061 Stage-1 map = 50%,  reduce = 0%, Cumulative CPU 6.66 sec
2024-11-21 11:02:34,108 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 13.28 sec
2024-11-21 11:02:38,196 Stage-1 map = 100%,  reduce = 75%, Cumulative CPU 25.72 sec
2024-11-21 11:02:40,241 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 29.42 sec
MapReduce Total cumulative CPU time: 29 seconds 420 msec
Ended Job = job_1732089968849_2727
Launching Job 2 out of 2
Number of reduce tasks not specified. Defaulting to jobconf value of: 4
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_2731, Tracking URL = http://master:6318/proxy/application_1732089968849_2731/
Kill Command = /opt/hadoop/bin/mapred job  -kill job_1732089968849_2731
Hadoop job information for Stage-2: number of mappers: 2; number of reducers: 4
2024-11-21 11:02:52,211 Stage-2 map = 0%,  reduce = 0%
2024-11-21 11:02:58,337 Stage-2 map = 50%,  reduce = 0%, Cumulative CPU 3.17 sec
2024-11-21 11:03:00,381 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 6.21 sec
2024-11-21 11:03:05,476 Stage-2 map = 100%,  reduce = 50%, Cumulative CPU 12.83 sec
2024-11-21 11:03:07,514 Stage-2 map = 100%,  reduce = 75%, Cumulative CPU 16.14 sec
2024-11-21 11:03:09,549 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 19.31 sec
MapReduce Total cumulative CPU time: 19 seconds 310 msec
```


Subscription Details | Nuvepro
cdacuser4135@ip-172-31-16-20
cdacuser4135@ip-172-31-16-20
npapc.cloudloka.com/shell/
Apps
Gmail
Chat
YouTube
Java
MySQL
Python
DSA
AWS
BIG DATA
Search
ENG
IN
16:33
21-11-2024

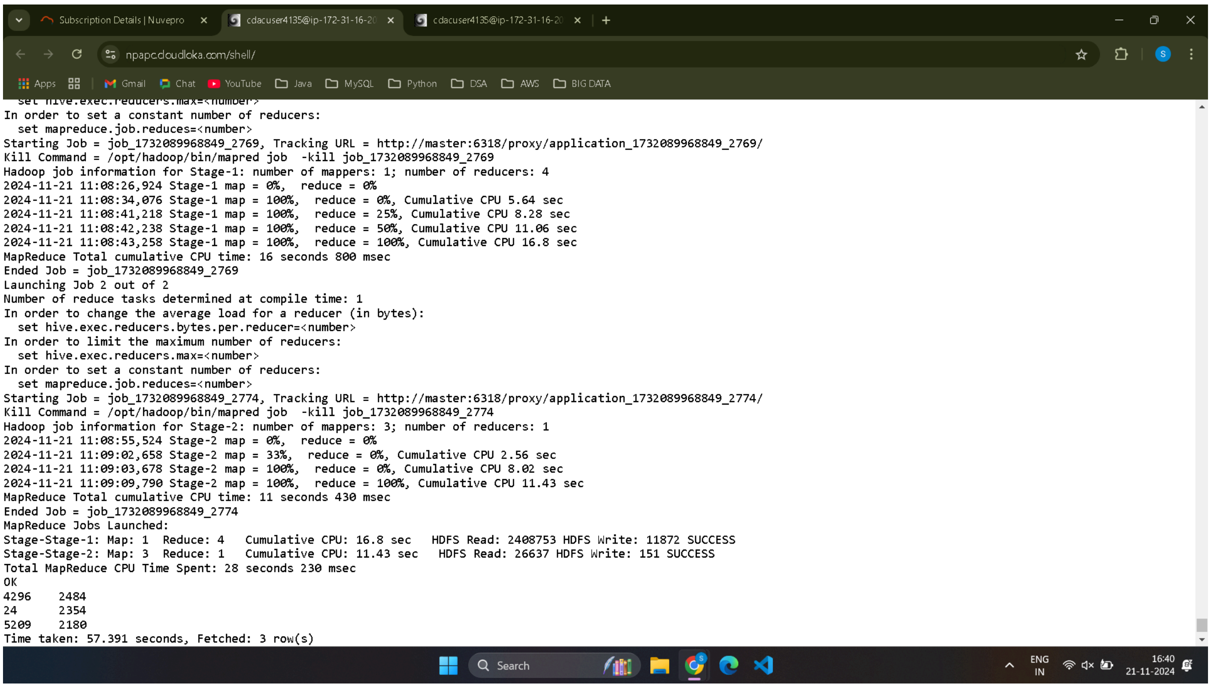
```
MapReduce Total cumulative CPU time: 29 seconds 420 msec
Ended Job = job_1732089968849_2727
Launching Job 2 out of 2
Number of reduce tasks not specified. Defaulting to jobconf value of: 4
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
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_2731, Tracking URL = http://master:6318/proxy/application_1732089968849_2731/
Kill Command = /opt/hadoop/bin/mapred job  -kill job_1732089968849_2731
Hadoop job information for Stage-2: number of mappers: 2; number of reducers: 4
2024-11-21 11:02:52,211 Stage-2 map = 0%,  reduce = 0%
2024-11-21 11:02:58,337 Stage-2 map = 50%,  reduce = 0%, Cumulative CPU 3.17 sec
2024-11-21 11:03:00,381 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 6.21 sec
2024-11-21 11:03:05,476 Stage-2 map = 100%,  reduce = 50%, Cumulative CPU 12.83 sec
2024-11-21 11:03:07,514 Stage-2 map = 100%,  reduce = 75%, Cumulative CPU 16.14 sec
2024-11-21 11:03:09,549 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 19.31 sec
MapReduce Total cumulative CPU time: 19 seconds 310 msec
Ended Job = job_1732089968849_2731
MapReduce Jobs Launched:
Stage-Stage-1: Map: 2  Reduce: 4   Cumulative CPU: 29.42 sec   HDFS Read: 3149759 HDFS Write: 262475 SUCCESS
Stage-Stage-2: Map: 2  Reduce: 4   Cumulative CPU: 19.31 sec   HDFS Read: 284329 HDFS Write: 812 SUCCESS
Total MapReduce CPU Time Spent: 48 seconds 730 msec
OK
Aalen Heidenheim Elchingen
Aarhus
Abaiang Atoll Airport
Abakan
Abbotsford
7 Novembre
A Coruna
Aaa
Aars
Abadan
Time taken: 60.996 seconds, Fetched: 10 row(s)
hive>
```

2. Determine the top 3 airlines that operate on the highest number of distinct routes.

```
SELECT
      airline_id, count(*)
FROM routes
GROUP BY airline_id
ORDER BY count(*) DESC
LIMIT 3;
```


Subscription Details | Nuvepro
cdacuser4135@ip-172-31-16-20
cdacuser4135@ip-172-31-16-20
npapc.cloudloka.com/shell/
Apps
Gmail
Chat
YouTube
Java
MySQL
Python
DSA
AWS
BIG DATA
master:6318/proxy/application_1732089968849_2774/
Search
ENG
IN
16:40
21-11-2024

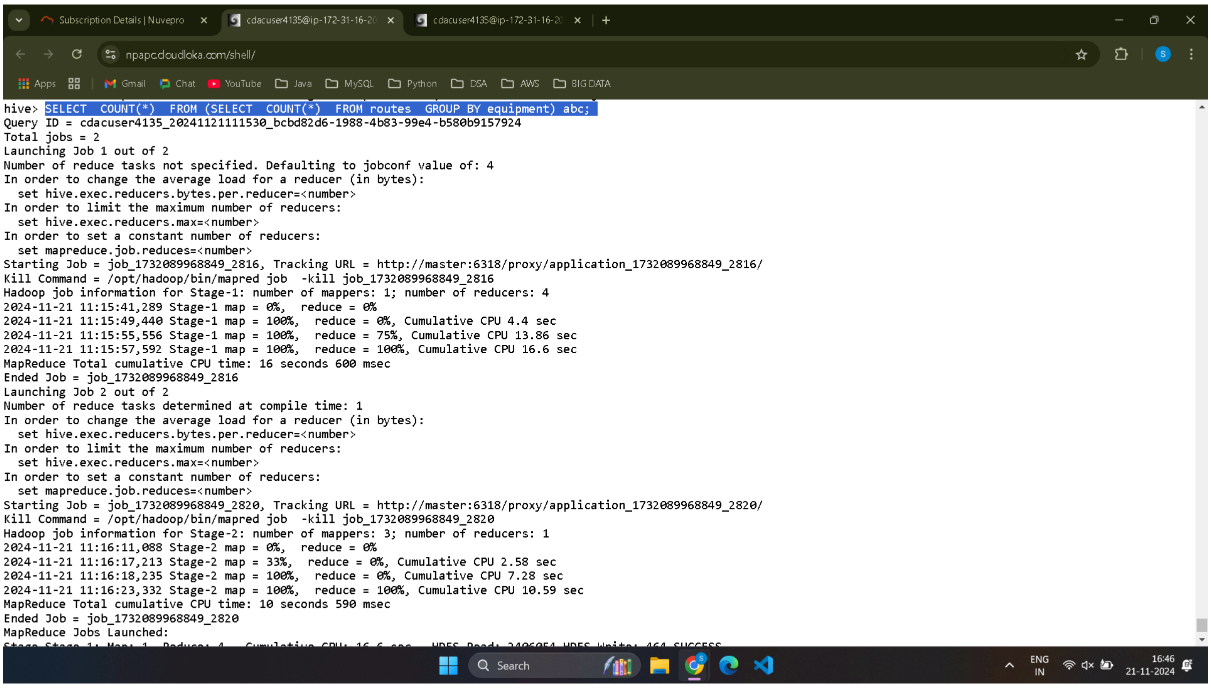
```
hive> SELECT airline_id, count(*) FROM routes GROUP BY airline_id ORDER BY count(*) DESC LIMIT 3;
Query ID = cdacuser4135_20241121110814_b5be759e-831c-4793-9018-021709deb922
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Defaulting to jobconf value of: 4
In order to 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_2769, Tracking URL = http://master:6318/proxy/application_1732089968849_2769/
Kill Command = /opt/hadoop/bin/mapred job  -kill job_1732089968849_2769
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 4
2024-11-21 11:08:26,924 Stage-1 map = 0%,  reduce = 0%
2024-11-21 11:08:34,076 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 5.64 sec
2024-11-21 11:08:41,218 Stage-1 map = 100%,  reduce = 25%, Cumulative CPU 8.28 sec
2024-11-21 11:08:42,238 Stage-1 map = 100%,  reduce = 50%, Cumulative CPU 11.06 sec
2024-11-21 11:08:43,258 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 16.8 sec
MapReduce Total cumulative CPU time: 16 seconds 800 msec
Ended Job = job_1732089968849_2769
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_2774, Tracking URL = http://master:6318/proxy/application_1732089968849_2774/
Kill Command = /opt/hadoop/bin/mapred job  -kill job_1732089968849_2774
Hadoop job information for Stage-2: number of mappers: 3; number of reducers: 1
2024-11-21 11:08:55,524 Stage-2 map = 0%,  reduce = 0%
2024-11-21 11:09:02,658 Stage-2 map = 33%,  reduce = 0%, Cumulative CPU 2.56 sec
2024-11-21 11:09:03,678 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 8.02 sec
2024-11-21 11:09:09,790 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 11.43 sec
MapReduce Total cumulative CPU time: 11 seconds 430 msec
Ended Job = job_1732089968849_2774
```


Subscription Details | Nuvepro
cdacuser4135@ip-172-31-16-20
cdacuser4135@ip-172-31-16-20
npapc.cloudloka.com/shell/
Apps
Gmail
Chat
YouTube
Java
MySQL
Python
DSA
AWS
BIG DATA
Search
ENG
IN
16:40
21-11-2024

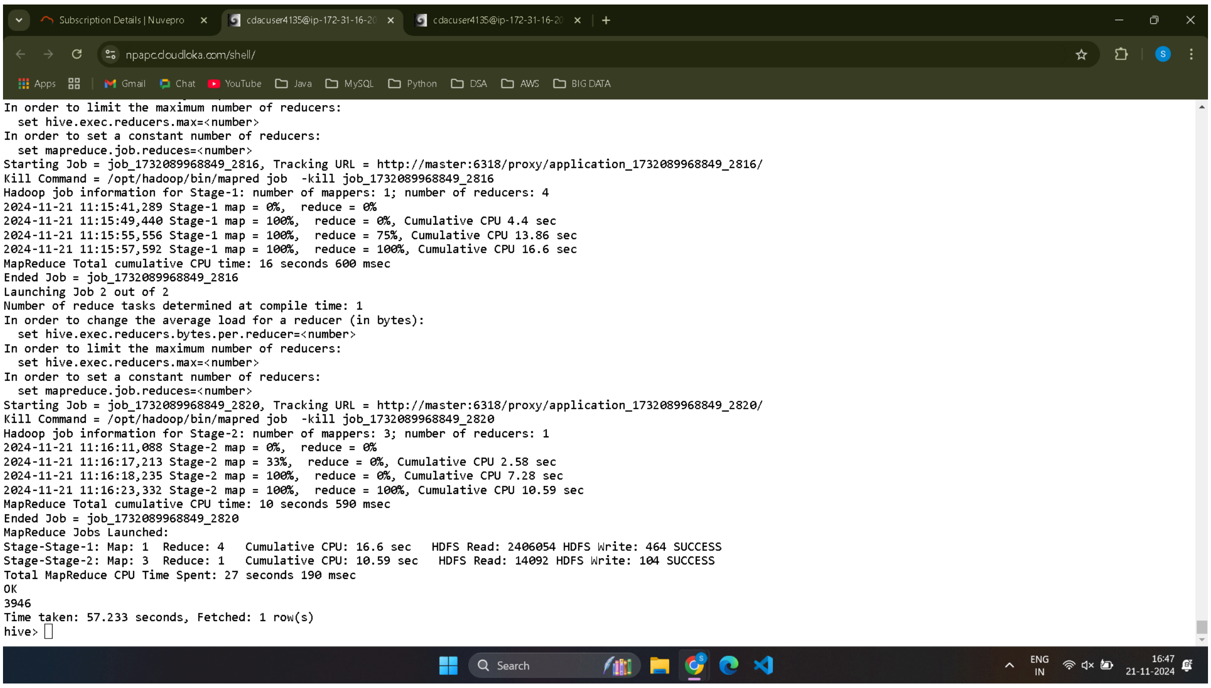
```
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1732089968849_2769, Tracking URL = http://master:6318/proxy/application_1732089968849_2769/
Kill Command = /opt/hadoop/bin/mapred job  -kill job_1732089968849_2769
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 4
2024-11-21 11:08:26,924 Stage-1 map = 0%,  reduce = 0%
2024-11-21 11:08:34,076 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 5.64 sec
2024-11-21 11:08:41,218 Stage-1 map = 100%,  reduce = 25%, Cumulative CPU 8.28 sec
2024-11-21 11:08:42,238 Stage-1 map = 100%,  reduce = 50%, Cumulative CPU 11.06 sec
2024-11-21 11:08:43,258 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 16.8 sec
MapReduce Total cumulative CPU time: 16 seconds 800 msec
Ended Job = job_1732089968849_2769
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_2774, Tracking URL = http://master:6318/proxy/application_1732089968849_2774/
Kill Command = /opt/hadoop/bin/mapred job  -kill job_1732089968849_2774
Hadoop job information for Stage-2: number of mappers: 3; number of reducers: 1
2024-11-21 11:08:55,524 Stage-2 map = 0%,  reduce = 0%
2024-11-21 11:09:02,658 Stage-2 map = 33%,  reduce = 0%, Cumulative CPU 2.56 sec
2024-11-21 11:09:03,678 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 8.02 sec
2024-11-21 11:09:09,790 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 11.43 sec
MapReduce Total cumulative CPU time: 11 seconds 430 msec
Ended Job = job_1732089968849_2774
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 4   Cumulative CPU: 16.8 sec   HDFS Read: 2408753 HDFS Write: 11872 SUCCESS
Stage-Stage-2: Map: 3  Reduce: 1   Cumulative CPU: 11.43 sec   HDFS Read: 26637 HDFS Write: 151 SUCCESS
Total MapReduce CPU Time Spent: 28 seconds 230 msec
OK
4296    2484
24      2354
5209    2180
Time taken: 57.391 seconds, Fetched: 3 row(s)
```

3. Find the total number of distinct aircraft types (Equipment) used in the routes table.

```
SELECT
       COUNT(*)
FROM (SELECT
       COUNT(*)
FROM routes
GROUP BY equipment)abc;
```



```
hive> SELECT  COUNT(*)  FROM (SELECT  COUNT(*)  FROM routes  GROUP BY equipment) abc;
Query ID = cdacuser4135_20241121111530_bcbd82d6-1988-4b83-99e4-b580b9157924
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Defaulting to jobconf value of: 4
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_2816, Tracking URL = http://master:6318/proxy/application_1732089968849_2816/
Kill Command = /opt/hadoop/bin/mapred job  -kill job_1732089968849_2816
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 4
2024-11-21 11:15:41,289 Stage-1 map = 0%,  reduce = 0%
2024-11-21 11:15:49,440 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 4.4 sec
2024-11-21 11:15:55,556 Stage-1 map = 100%,  reduce = 75%, Cumulative CPU 13.86 sec
2024-11-21 11:15:57,592 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 16.6 sec
MapReduce Total cumulative CPU time: 16 seconds 600 msec
Ended Job = job_1732089968849_2816
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_2820, Tracking URL = http://master:6318/proxy/application_1732089968849_2820/
Kill Command = /opt/hadoop/bin/mapred job  -kill job_1732089968849_2820
Hadoop job information for Stage-2: number of mappers: 3; number of reducers: 1
2024-11-21 11:16:11,088 Stage-2 map = 0%,  reduce = 0%
2024-11-21 11:16:17,213 Stage-2 map = 33%,  reduce = 0%, Cumulative CPU 2.58 sec
2024-11-21 11:16:18,235 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 7.28 sec
2024-11-21 11:16:23,332 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 10.59 sec
MapReduce Total cumulative CPU time: 10 seconds 590 msec
Ended Job = job_1732089968849_2820
MapReduce Jobs Launched:
```



```
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_2816, Tracking URL = http://master:6318/proxy/application_1732089968849_2816/
Kill Command = /opt/hadoop/bin/mapred job  -kill job_1732089968849_2816
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 4
2024-11-21 11:15:41,289 Stage-1 map = 0%,  reduce = 0%
2024-11-21 11:15:49,440 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 4.4 sec
2024-11-21 11:15:55,556 Stage-1 map = 100%,  reduce = 75%, Cumulative CPU 13.86 sec
2024-11-21 11:15:57,592 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 16.6 sec
MapReduce Total cumulative CPU time: 16 seconds 600 msec
Ended Job = job_1732089968849_2816
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_2820, Tracking URL = http://master:6318/proxy/application_1732089968849_2820/
Kill Command = /opt/hadoop/bin/mapred job  -kill job_1732089968849_2820
Hadoop job information for Stage-2: number of mappers: 3; number of reducers: 1
2024-11-21 11:16:11,088 Stage-2 map = 0%,  reduce = 0%
2024-11-21 11:16:17,213 Stage-2 map = 33%,  reduce = 0%, Cumulative CPU 2.58 sec
2024-11-21 11:16:18,235 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 7.28 sec
2024-11-21 11:16:23,332 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 10.59 sec
MapReduce Total cumulative CPU time: 10 seconds 590 msec
Ended Job = job_1732089968849_2820
MapReduce Jobs Launched:
```

```
Stage-Stage-1: Map: 1  Reduce: 4   Cumulative CPU: 16.6 sec   HDFS Read: 2406054 HDFS Write: 464 SUCCESS
Stage-Stage-2: Map: 3  Reduce: 1   Cumulative CPU: 10.59 sec   HDFS Read: 14092 HDFS Write: 104 SUCCESS
Total MapReduce CPU Time Spent: 27 seconds 190 msec
OK
3946
Time taken: 57.233 seconds, Fetched: 1 row(s)
hive> 
```

Question 2.

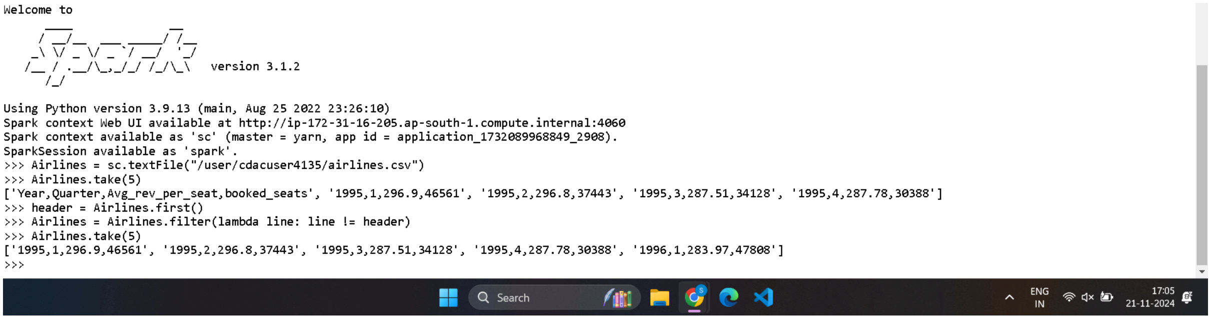
1. Write an SQL query to create a partitioned table named routes_partitioned to store route details, partitioned by the Destination_Airport Column

```
CREATE TABLE routes_partioned
     (STRING airline_iata,
     INT airline_id,
     STRING src_airport_iata,
     INT  src_airport_id,
     INT dest_airport_id,
     STRING codeshare,
     INT stops,
     STRING equipment)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ‘,’
STORED AS TEXTFILE
PARTITIONED BY(STRING dest_airport_iata)
‘/user/cdacuser4135/routes.csv’
```

2. Insert data into the routes_partitioned table for a specific Destination_Airport
3. To query data for a specific Destination_Airport
4. Write a query to display all available partitions in the routes_partitioned table.

SPARK:

```
>>> Airlines = sc.textFile("/user/cdacuser4135/airlines.csv")
>>> Airlines.take(5)
>>> header = Airlines.first()
>>> Airlines = Airlines.filter(lambda line: line != header)
>>> Airlines.take(5)
```



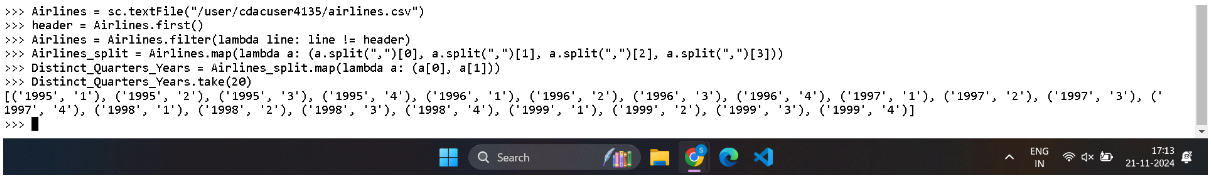
```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.1.2
      /_/

Using Python version 3.9.13 (main, Aug 25 2022 23:26:10)
Spark context Web UI available at http://ip-172-31-16-205.ap-south-1.compute.internal:4060
Spark context available as 'sc' (master = yarn, app id = application_1732089968849_2908).
SparkSession available as 'spark'.
>>> Airlines = sc.textFile("/user/cdacuser4135/airlines.csv")
>>> Airlines.take(5)
['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']
>>> header = Airlines.first()
>>> Airlines = Airlines.filter(lambda line: line != header)
>>> Airlines.take(5)
['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']
>>>
```

Question 1.

1. Count the number of rows where the total no. of booked seats is between 20,000 and 50,000

2. Find and list all the distinct quarters and years (combined as tuples) present in the dataset. Ensure no duplicates are included in the result.

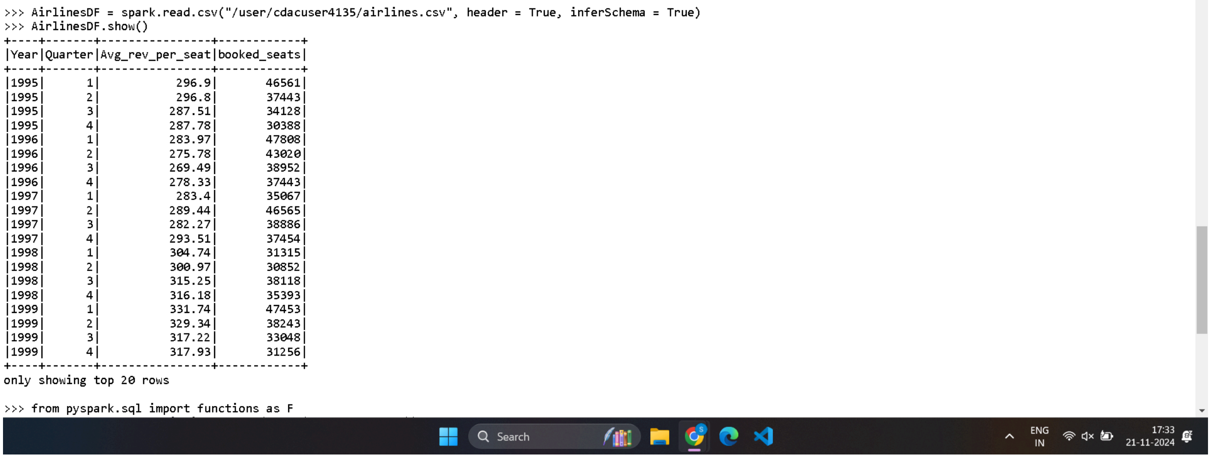
```
>>> Distinct_Quarters_Years = Airlines_split.map(lambda a: (a[0], a[1]))
>>> Distinct_Quarters_Years.take(20)
```



```
>>> Airlines = sc.textFile("/user/cdacuser4135/airlines.csv")
>>> header = Airlines.first()
>>> Airlines = Airlines.filter(lambda line: line != header)
>>> Airlines_split = Airlines.map(lambda a: (a.split(",")[0], a.split(",")[1], a.split(",")[2], a.split(",")[3]))
>>> Distinct_Quarters_Years = Airlines_split.map(lambda a: (a[0], a[1]))
>>> Distinct_Quarters_Years.take(20)
[('1995', '1'), ('1995', '2'), ('1995', '3'), ('1995', '4'), ('1996', '1'), ('1996', '2'), ('1996', '3'), ('1996', '4'), ('1997', '1'), ('1997', '2'), ('1997', '3'), ('
1997', '4'), ('1998', '1'), ('1998', '2'), ('1998', '3'), ('1998', '4'), ('1999', '1'), ('1999', '2'), ('1999', '3'), ('1999', '4')]
>>> █
```

Question 2.

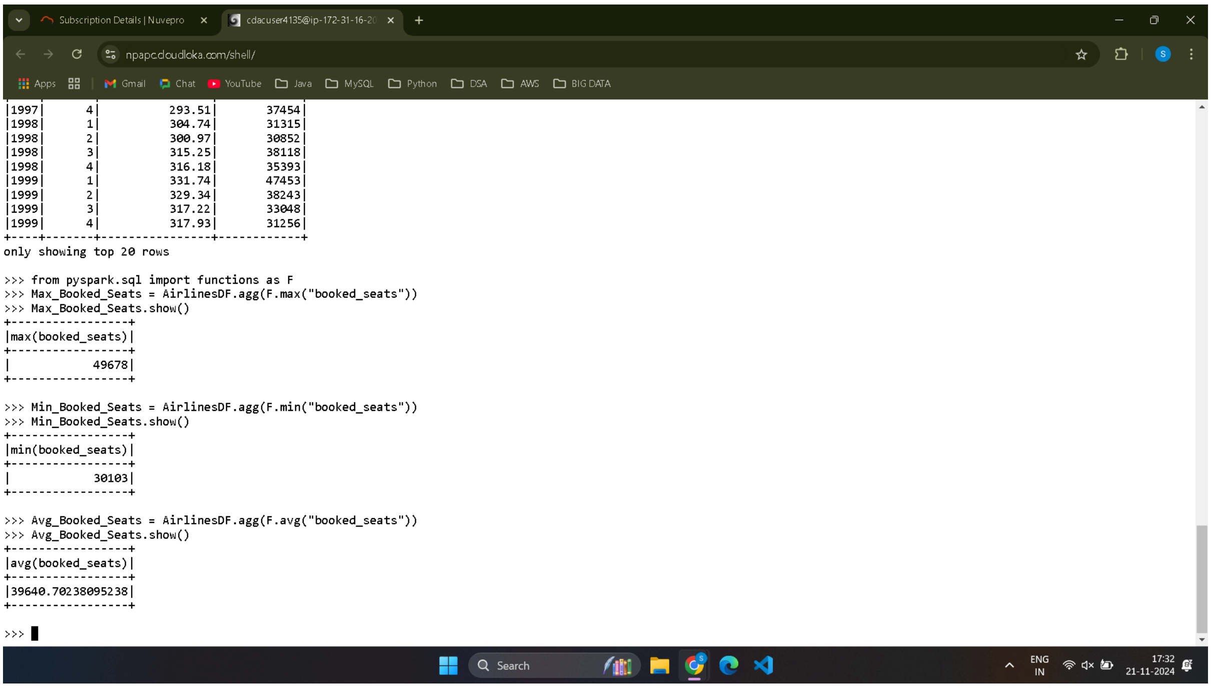
```
>>> AirlinesDF = spark.read.csv("/user/cdacuser4135/airlines.csv", header = True, inferSchema = True)
>>> AirlinesDF.show()
>>> from pyspark.sql import functions as F
```



```
>>> AirlinesDF = spark.read.csv("/user/cdacuser4135/airlines.csv", header = True, inferSchema = True)
>>> AirlinesDF.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

>>> from pyspark.sql import functions as F
```

1. Find the minimum, maximum and average of the total number of booked seats.

```
>>> Max_Booked_Seats = AirlinesDF.agg(F.max("booked_seats"))
>>> Max_Booked_Seats.show()
+-----------------+
|max(booked_seats)|
+-----------------+
|            49678|
+-----------------+

>>> Min_Booked_Seats = AirlinesDF.agg(F.min("booked_seats"))
>>> Min_Booked_Seats.show()
+-----------------+
|min(booked_seats)|
+-----------------+
|            30103|
+-----------------+

>>> Avg_Booked_Seats = AirlinesDF.agg(F.avg("booked_seats"))
>>> Avg_Booked_Seats.show()
+-----------------+
|avg(booked_seats)|
+-----------------+
|39640.70238095238|
+-----------------+
```

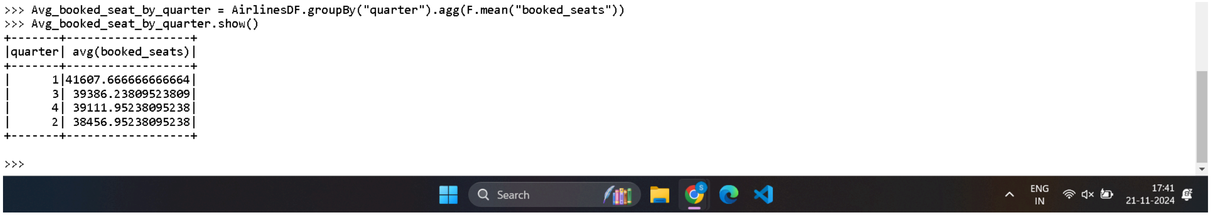

Subscription Details | Nuvepro
cdacuser4135@ip-172-31-16-20
npapc.cloudloka.com/shell/
Apps
Gmail
Chat
YouTube
Java
MySQL
Python
DSA
AWS
BIG DATA
Search
ENG
IN
17:32
21-11-2024

```
|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

>>> from pyspark.sql import functions as F
>>> Max_Booked_Seats = AirlinesDF.agg(F.max("booked_seats"))
>>> Max_Booked_Seats.show()
+-----------------+
|max(booked_seats)|
+-----------------+
|            49678|
+-----------------+

>>> Min_Booked_Seats = AirlinesDF.agg(F.min("booked_seats"))
>>> Min_Booked_Seats.show()
+-----------------+
|min(booked_seats)|
+-----------------+
|            30103|
+-----------------+

>>> Avg_Booked_Seats = AirlinesDF.agg(F.avg("booked_seats"))
>>> Avg_Booked_Seats.show()
+-----------------+
|avg(booked_seats)|
+-----------------+
|39640.70238095238|
+-----------------+

>>> 
```

2. Count the number of rows where the average revenue per seat is less than $290.

3. Find the average number of booked seats grouped by quarter.

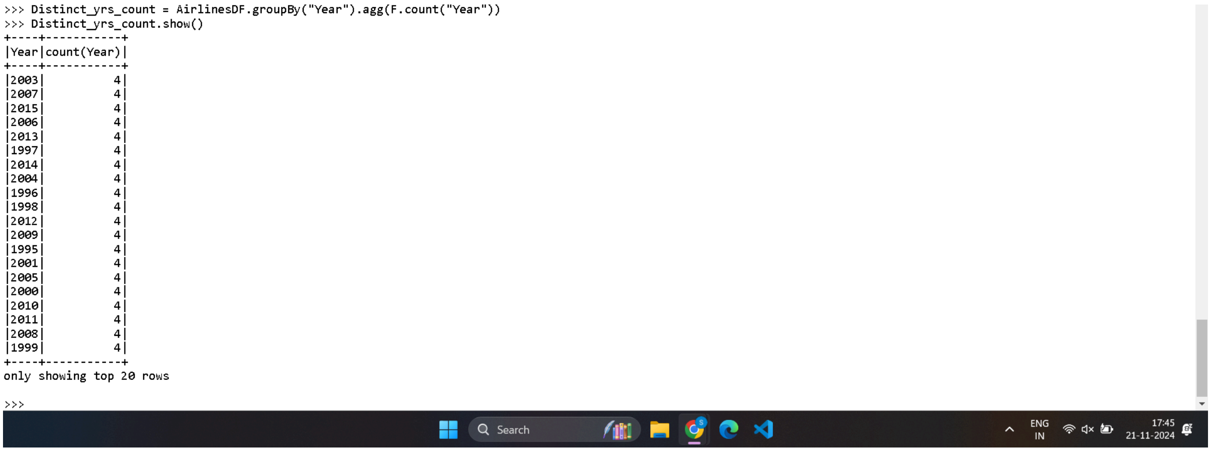
```
>>> Avg_booked_seat_by_quarter = 
AirlinesDF.groupBy("quarter").agg(F.mean("booked_seats"))
>>> Avg_booked_seat_by_quarter.show()
```



```
>>> Avg_booked_seat_by_quarter = AirlinesDF.groupBy("quarter").agg(F.mean("booked_seats"))
>>> Avg_booked_seat_by_quarter.show()
+-------+-----------------+
|quarter| avg(booked_seats)|
+-------+-----------------+
|      1|41607.666666666664|
|      3| 39386.23809523809|
|      4| 39111.95238095238|
|      2| 38456.95238095238|
+-------+-----------------+

>>>
```

4. List all distinct years in the dataset along with the count of rows for each year.

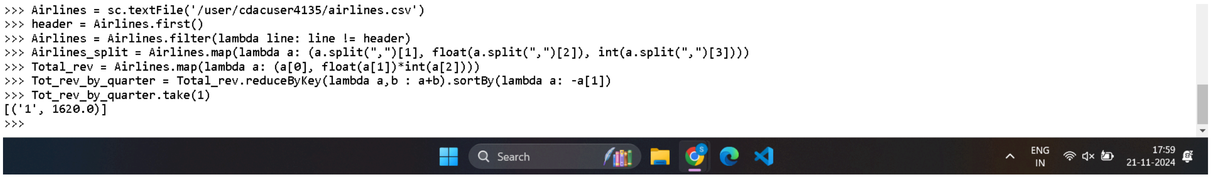
```
>>> Distinct_yrs_count = 
AirlinesDF.groupBy("Year").agg(F.count("Year"))
>>> Distinct_yrs_count.show()
```



```
>>> Distinct_yrs_count = AirlinesDF.groupBy("Year").agg(F.count("Year"))
>>> Distinct_yrs_count.show()
+----+-----------+
|Year|count(Year)|
+----+-----------+
|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

>>>
```

5. Find the quarter with the highest total revenue across the years and its value.

```
>>> Airlines = sc.textFile('/user/cdacuser4135/airlines.csv')
>>> header = Airlines.first()
>>> Airlines = Airlines.filter(lambda line: line != header)
>>> Airlines_split = Airlines.map(lambda a: (a.split(",")[1],
float(a.split(",")[2]), int(a.split(",")[3])))
>>> Total_rev = Airlines.map(lambda a: (a[0],
float(a[1])*int(a[2])))
>>> Tot_rev_by_quarter = Total_rev.reduceByKey(lambda a,b :
a+b).sortBy(lambda a: -a[1])
>>> Tot_rev_by_quarter.take(1)
[('1', 1620.0)]
```



```
>>> Airlines = sc.textFile('/user/cdacuser4135/airlines.csv')
>>> header = Airlines.first()
>>> Airlines = Airlines.filter(lambda line: line != header)
>>> Airlines_split = Airlines.map(lambda a: (a.split(",")[1], float(a.split(",")[2]), int(a.split(",")[3])))
>>> Total_rev = Airlines.map(lambda a: (a[0], float(a[1])*int(a[2])))
>>> Tot_rev_by_quarter = Total_rev.reduceByKey(lambda a,b : a+b).sortBy(lambda a: -a[1])
>>> Tot_rev_by_quarter.take(1)
[('1', 1620.0)]
>>>
```