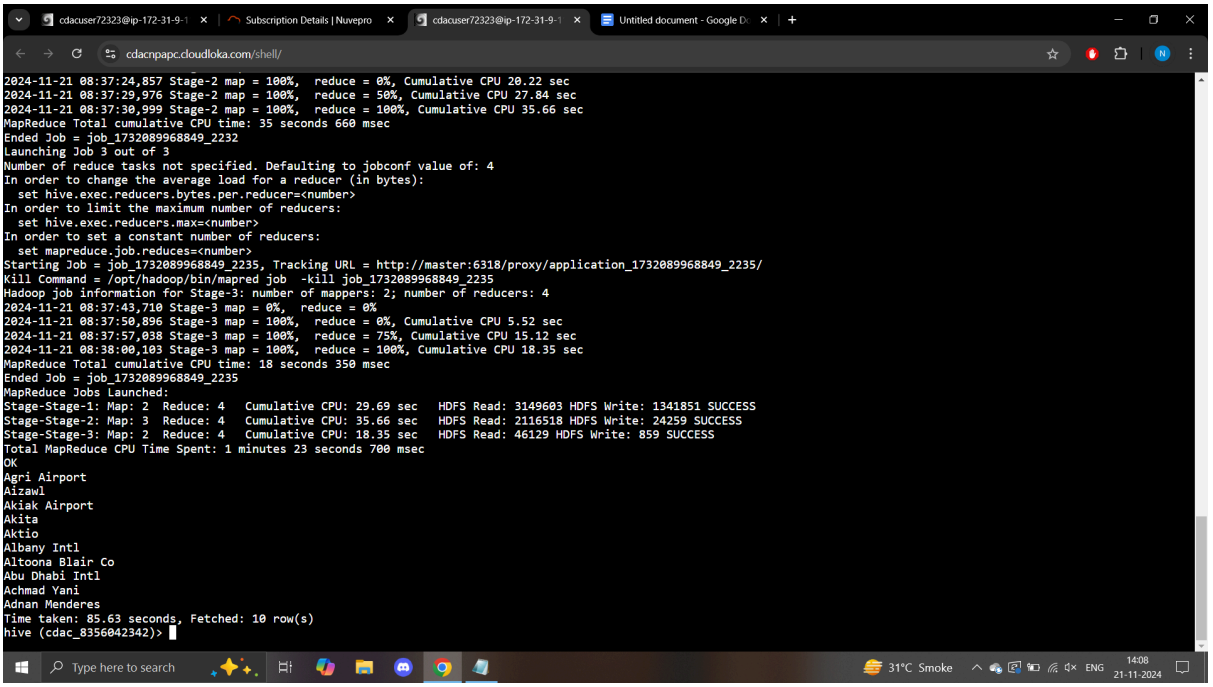
Hive

Question 1:

1:

Select distinct a.name from airport a join routes r on a.airport_id = r.src_airport_id join airport a1 on a1.airport_id = r.dest_airport_id wher
e a1.airport_id = r.dest_airport_id and a.airport_id = r.src_airport_id limit 10;



```
2024-11-21 08:37:24,857 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 20.22 sec
2024-11-21 08:37:29,976 Stage-2 map = 100%,  reduce = 50%, Cumulative CPU 27.84 sec
2024-11-21 08:37:30,999 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 35.66 sec
MapReduce Total cumulative CPU time: 35 seconds 660 msec
Ended Job = job_1732089968849_2232
Launching Job 3 out of 3
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_2235, Tracking URL = http://master:6318/proxy/application_1732089968849_2235/
Kill Command = /opt/hadoop/bin/mapred job  -kill job_1732089968849_2235
Hadoop job information for Stage-3: number of mappers: 2; number of reducers: 4
2024-11-21 08:37:43,710 Stage-3 map = 0%,  reduce = 0%
2024-11-21 08:37:50,896 Stage-3 map = 100%,  reduce = 0%, Cumulative CPU 5.52 sec
2024-11-21 08:37:57,038 Stage-3 map = 100%,  reduce = 75%, Cumulative CPU 15.12 sec
2024-11-21 08:38:00,103 Stage-3 map = 100%,  reduce = 100%, Cumulative CPU 18.35 sec
MapReduce Total cumulative CPU time: 18 seconds 350 msec
Ended Job = job_1732089968849_2235
MapReduce Jobs Launched:
Stage-Stage-1: Map: 2  Reduce: 4   Cumulative CPU: 29.69 sec   HDFS Read: 3149603 HDFS Write: 1341851 SUCCESS
Stage-Stage-2: Map: 3  Reduce: 4   Cumulative CPU: 35.66 sec   HDFS Read: 2116518 HDFS Write: 24259 SUCCESS
Stage-Stage-3: Map: 2  Reduce: 4   Cumulative CPU: 18.35 sec   HDFS Read: 46129 HDFS Write: 859 SUCCESS
Total MapReduce CPU Time Spent: 1 minutes 23 seconds 700 msec
OK
Agri Airport
Aizawl
Akiak Airport
Akita
Aktio
Albany Intl
Altoona Blair Co
Abu Dhabi Intl
Achmad Yani
Adnan Menderes
Time taken: 85.63 seconds, Fetched: 10 row(s)
hive (cdac_8356042342)>
```

2:

select concat(src_airport_id," ",dest_airport_id) ,equipment,count(*) from routes group by concat(src_airport_id," ",dest_airport_id),equipment order by count(*) d
esc limit 1;

(Hive stopped working)

```
-bash: export: `/opt/anaconda3/bin/python': not a valid identifier
-bash: export: `/opt/anaconda3/bin/python': not a valid identifier
cdacuser72323@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 = 29aa4e6b-d619-4ae2-ab43-60cac93a2c83

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 = 7a8df13a-1217-4392-bcd9-a65f9939e7d7
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 rele
ases.
hive> use cdac_8356042342;
OK
Time taken: 61.204 seconds
hive> select concat(src_airport_id," ",dest_airport_id) ,equipment,count(*) from routes group by concat(src_airport_id," ",dest_airport_id),equipment order by count(*)
d esc limit 1;
FAILED: ParseException line 1:162 missing EOF at 'd' near ')'
hive> select concat(src_airport_id," ",dest_airport_id) ,equipment,count(*) from routes group by concat(src_airport_id," ",dest_airport_id),equipment order by count(*)
desc limit 1;
Query ID = cdacuser72323_20241121100012_9c19893b-46d1-4ea7-95ff-45b61785dae4
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_2605, Tracking URL = http://master:6318/proxy/application_1732089968849_2605/
Kill Command = /opt/hadoop/bin/mapred job  -kill job_1732089968849_2605

[1]+  Stopped                 hive
cdacuser72323@ip-172-31-9-116:~$ pysparkhive
pysparkhive: command not found
cdacuser72323@ip-172-31-9-116:~$
```

3:

select a.name , count(*) from airlines a join routes r on a.airline_id = r.airline_id group by concat(src_airport_id," ",dest_airport_id),name
order by count(*) desc limit 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_2389, Tracking URL = http://master:6318/proxy/application_1732089968849_2389/
Kill Command = /opt/hadoop/bin/mapred job  -kill job_1732089968849_2389
Hadoop job information for Stage-2: number of mappers: 2; number of reducers: 4
2024-11-21 09:02:51,752 Stage-2 map = 0%,  reduce = 0%
2024-11-21 09:02:58,986 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 10.07 sec
2024-11-21 09:03:06,157 Stage-2 map = 100%,  reduce = 75%, Cumulative CPU 24.15 sec
2024-11-21 09:03:08,202 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 28.52 sec
MapReduce Total cumulative CPU time: 28 seconds 520 msec
Ended Job = job_1732089968849_2389
Launching Job 3 out of 3
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_2394, Tracking URL = http://master:6318/proxy/application_1732089968849_2394/
Kill Command = /opt/hadoop/bin/mapred job  -kill job_1732089968849_2394
Hadoop job information for Stage-3: number of mappers: 3; number of reducers: 1
2024-11-21 09:03:20,877 Stage-3 map = 0%,  reduce = 0%
2024-11-21 09:03:29,079 Stage-3 map = 100%,  reduce = 0%, Cumulative CPU 11.81 sec
2024-11-21 09:03:35,220 Stage-3 map = 100%,  reduce = 100%, Cumulative CPU 15.21 sec
MapReduce Total cumulative CPU time: 15 seconds 210 msec
Ended Job = job_1732089968849_2394
MapReduce Jobs Launched:
Stage-Stage-1: Map: 2  Reduce: 4   Cumulative CPU: 40.75 sec   HDFS Read: 2729160 HDFS Write: 2862459 SUCCESS
Stage-Stage-2: Map: 2  Reduce: 4   Cumulative CPU: 28.52 sec   HDFS Read: 2884739 HDFS Write: 2220498 SUCCESS
Stage-Stage-3: Map: 3  Reduce: 1   Cumulative CPU: 15.21 sec   HDFS Read: 2235311 HDFS Write: 116 SUCCESS
Total MapReduce CPU Time Spent: 1 minutes 24 seconds 480 msec
OK
Air Greenland   62
Time taken: 87.23 seconds, Fetched: 1 row(s)
hive (cdac_8356042342)>
```

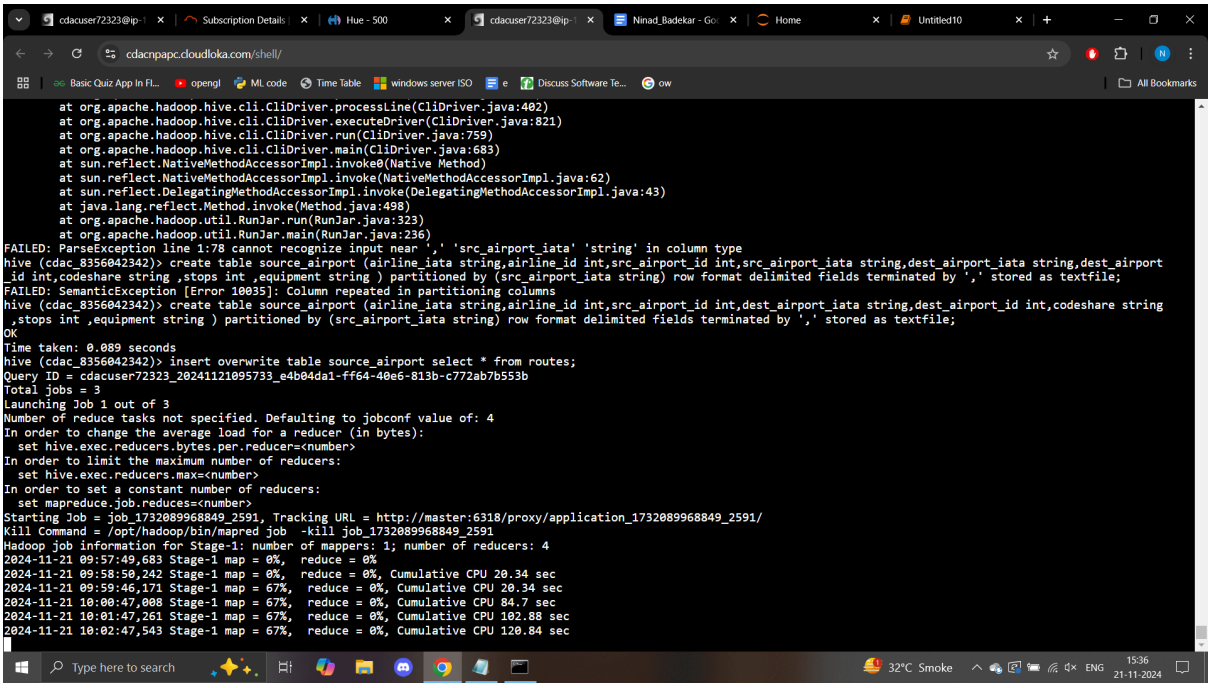
Question 2:

1.

create table source_airport (airline_iata string,airline_id int,src_airport_id,src_airport_iata string,dest_airport_iata string,dest_airport_id int,codeshare string ,stops int ,equipment string

) partitioned by (src_airport_iata int) row format delimited fields terminated by ',' stored as textfile;
insert overwrite table source_airport select * from routes;

(Hive stopped working)



2.
Select * from source_airport where src_aiport_iata = ‘JFK’
3.
Select * from source_airport where src_aiport_iata = ‘LAX’
4.

Spark

Question 1:

1.
```
rdd = sc.textFile("air/airseat.csv")
header = rdd.first()
rdd = rdd.filter(lambda a: a!=header)
rdd = rdd.map(lambda a: int(a.split(',')[0]),int(a.split(',')[1]),float(a.split(',')[2]),int(a.split(',')[3]))
excRdd = rdd.filter(lambda a : a>40000).count()
print(excRDD)
```

```
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-172-31-9-116.ap-south-1.compute.internal:4053
Spark context available as 'sc' (master = yarn, app id = application_1732089968849_2468).
SparkSession available as 'spark'.
>>> rdd = sc.textFile("air/airseat.csv")
>>> rdd.take 2
  File "<stdin>", line 1
    rdd.take 2
            ^
SyntaxError: invalid syntax
>>> rdd.take(2)
['Year,Quarter,Avg_rev_per_seat,booked_seats', '1995,1,296.9,46561']
>>> header = rdd.first()
>>> header
'Year,Quarter,Avg_rev_per_seat,booked_seats'
>>> rdd = rdd.filter(lambda a : a!=header)
>>> rdd.take(2)
['1995,1,296.9,46561', '1995,2,296.8,37443']
>>> rdd = rdd.map(lambda a: int(a.split(',')[0]),int(a.split(',')[1]),float(a.split(',')[2]),int(a.split(',')[3]))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'a' is not defined
>>> rdd = rdd.map(lambda a: (int(a.split(',')[0]),int(a.split(',')[1]),float(a.split(',')[2]),int(a.split(',')[3])))
>>> rdd.take(2)
[(1995, 1, 296.9, 46561), (1995, 2, 296.8, 37443)]
>>> excRows = rdd.filter(lambda a:a[3]>40000).count()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'PipelinedRDD' object has no attribute 'coount'
>>> excRows = rdd.filter(lambda a:a[3]>40000).count()
>>> print(excRows)
38
>>>
```

2.

```
>>> years = rdd.map(lambda a:a[0]).distinct()
>>> for item in years.collect():
...     print(item)
...
```

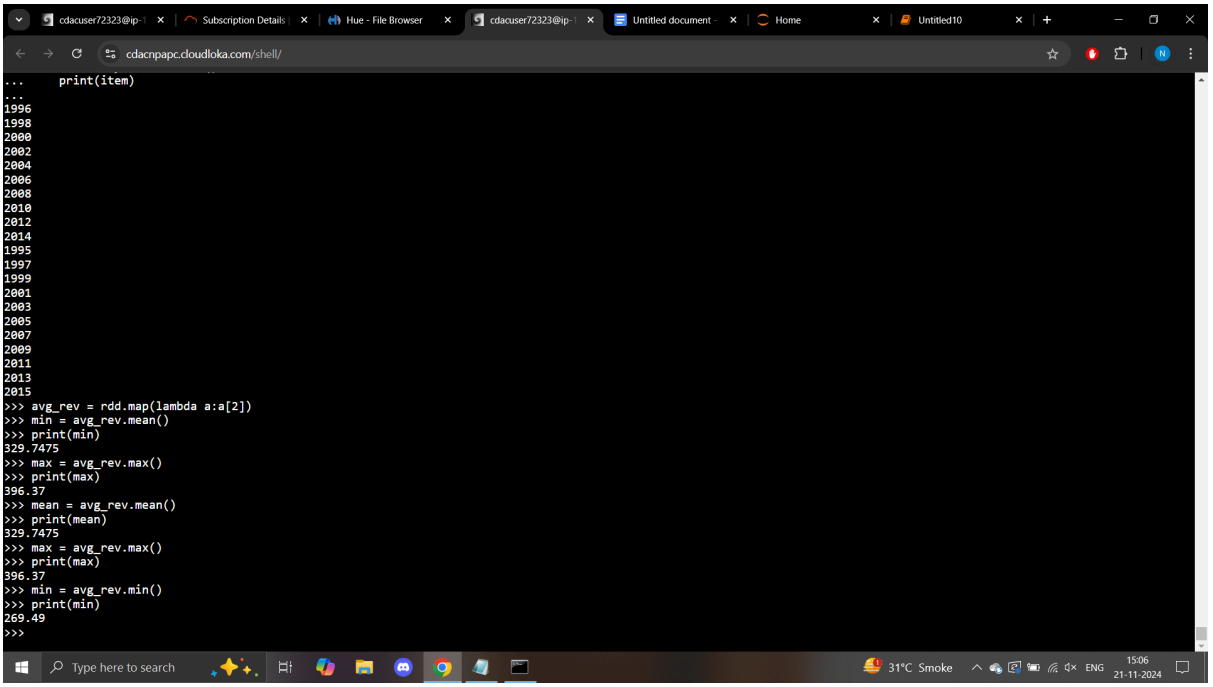
```
38
>>> years = rdd.map(lambda a: a[0]).distinct()
>>> print(year)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'year' is not defined
>>> print(years)
PythonRDD[11] at RDD at PythonRDD.scala:53
>>> for item in years:
... nt(item)
  File "<stdin>", line 2
    nt(item)
    ^
IndentationError: expected an indented block
>>> years = rdd.map(lambda a:a[0]).distinct()
>>> for item in years.collect():
...     print(item)
...
1996
1998
2000
2002
2004
2006
2008
2010
2012
2014
1995
1997
1999
2001
2003
2005
2007
2009
2011
2013
2015
>>>
```

Question 2:

1.

```
>>> avg_rev = rdd.map(lambda a:a[2])
```

```
>>> mean = avg_rev.mean()
>>> print(mean)
329.7475
>>> max = avg_rev.max()
>>> print(max)
396.37
>>> min = avg_rev.min()
>>> print(min)
269.49
>>>
```



2.

```
>>> avg_rev.take(5)
[296.9, 296.8, 287.51, 287.78, 283.97]
>>> avg_290 = avg_rev.filter(lambda a : a > 290)
>>> avg_290.take(5)
[296.9, 296.8, 293.51, 304.74, 300.97]
>>>
```

```
TypeError: 'float' object is not subscriptable

        at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:517)
        at org.apache.spark.api.python.PythonRunner$$anon$3.read(PythonRunner.scala:652)
        at org.apache.spark.api.python.PythonRunner$$anon$3.read(PythonRunner.scala:635)
        at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:470)
        at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)
        at scala.collection.Iterator.foreach(Iterator.scala:941)
        at scala.collection.Iterator.foreach$(Iterator.scala:941)
        at org.apache.spark.InterruptibleIterator.foreach(InterruptibleIterator.scala:28)
        at scala.collection.generic.Growable.$plus$plus$eq(Growable.scala:62)
        at scala.collection.generic.Growable.$plus$plus$eq$(Growable.scala:53)
        at scala.collection.mutable.ArrayBuffer.$plus$plus$eq(ArrayBuffer.scala:105)
        at scala.collection.mutable.ArrayBuffer.$plus$plus$eq(ArrayBuffer.scala:49)
        at scala.collection.TraversableOnce.to(TraversableOnce.scala:315)
        at scala.collection.TraversableOnce.to$(TraversableOnce.scala:313)
        at org.apache.spark.InterruptibleIterator.to(InterruptibleIterator.scala:28)
        at scala.collection.TraversableOnce.toBuffer(TraversableOnce.scala:307)
        at scala.collection.TraversableOnce.toBuffer$(TraversableOnce.scala:307)
        at org.apache.spark.InterruptibleIterator.toBuffer(InterruptibleIterator.scala:28)
        at scala.collection.TraversableOnce.toArray(TraversableOnce.scala:294)
        at scala.collection.TraversableOnce.toArray$(TraversableOnce.scala:288)
        at org.apache.spark.InterruptibleIterator.toArray(InterruptibleIterator.scala:28)
        at org.apache.spark.api.python.PythonRDD$.$anonfun$runJob$1(PythonRDD.scala:166)
        at org.apache.spark.SparkContext.$anonfun$runJob$5(SparkContext.scala:2236)
        at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)
        at org.apache.spark.scheduler.Task.run(Task.scala:131)
        at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:497)
        at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1439)
        at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:500)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
        ... 1 more

>>> avg_rev.take(5)
[296.9, 296.8, 287.51, 287.78, 283.97]
>>> avg_290 = avg_rev.filter(lambda a : a > 290)
>>> avg_290.take(5)
[296.9, 296.8, 293.51, 304.74, 300.97]
>>>
```

3.

```
>>> tot = rdd.map(lambda a: (a[0],a[3]))
>>> year_seat = tot.reduceByKey(lambda a,b : a+b)
>>> year_seat.take(5)
[(1996, 167223), (1998, 135678), (2000, 154376), (2002, 152195),
(2004, 164800)]
>>> for item in year_seat.take(5):
...     print(item)
...
(1996, 167223)
(1998, 135678)
(2000, 154376)
(2002, 152195)
(2004, 164800)
```

```
    return f(iterator)
  File "/opt/spark-3.1.2/python/pyspark/rdd.py", line 2144, in combineLocally
    merger.mergeValues(iterator)
  File "/opt/spark-3.1.2/python/pyspark/shuffle.py", line 240, in mergeValues
    for k, v in iterator:
ValueError: too many values to unpack (expected 2)

        at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:517)
        at org.apache.spark.api.python.PythonRunner$$anon$3.read(PythonRunner.scala:652)
        at org.apache.spark.api.python.PythonRunner$$anon$3.read(PythonRunner.scala:635)
        at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:470)
        at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)
        at scala.collection.Iterator$GroupedIterator.fill(Iterator.scala:1209)
        at scala.collection.Iterator$GroupedIterator.hasNext(Iterator.scala:1215)
        at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:458)
        at org.apache.spark.shuffle.sort.BypassMergeSortShuffleWriter.write(BypassMergeSortShuffleWriter.java:132)
        at org.apache.spark.shuffle.ShuffleWriteProcessor.write(ShuffleWriteProcessor.scala:59)
        at org.apache.spark.scheduler.ShuffleMapTask.runTask(ShuffleMapTask.scala:99)
        at org.apache.spark.scheduler.ShuffleMapTask.runTask(ShuffleMapTask.scala:52)
        at org.apache.spark.scheduler.Task.run(Task.scala:131)
        at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:497)
        at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1439)
        at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:500)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
        ... 1 more

>>> tot = rdd.map(lambda a: (a[0],a[3]))
>>> year_seat = tot.reduceByKey(lambda a,b : a+b)
>>> year_seat.take(5)
[(1996, 167223), (1998, 135678), (2000, 154376), (2002, 152195), (2004, 164800)]
>>> for item in year_seat.take(5):
...     print(item)
...
(1996, 167223)
(1998, 135678)
(2000, 154376)
(2002, 152195)
(2004, 164800)
>>>
```

4.

```
>>> years = rdd.map(lambda a:a[0]).distinct()
>>> for item in years.collect():
...     print(item)
...
```

```
        ... 1 more

>>> tot = rdd.map(lambda a: (a[0],a[3]))
>>> year_seat = tot.reduceByKey(lambda a,b : a+b)
>>> year_seat.take(5)
[(1996, 167223), (1998, 135678), (2000, 154376), (2002, 152195), (2004, 164800)]
>>> for item in year_seat.take(5):
...     print(item)
...
(1996, 167223)
(1998, 135678)
(2000, 154376)
(2002, 152195)
(2004, 164800)
>>> years = rdd.map(lambda a:a[0]).distinct()
>>> for item in years.collect():
...     print(item)
...
1996
1998
2000
2002
2004
2006
2008
2010
2012
2014
1995
1997
1999
2001
2003
2005
2007
2009
2011
2013
2015
>>>
```

5.

```
>>> cumu = rdd.map(lambda a: (a[0],a[2]))
>>> cumu.take(5)
```

```
[(1995, 296.9), (1995, 296.8), (1995, 287.51), (1995, 287.78),
(1996, 283.97)]
>>> cumulative_per_year = cumu.reduceByKey(lambda a,b:a+b)
>>> cumulative_per_year.take(5)
[(1996, 1107.57), (1998, 1237.14), (2000, 1356.13), (2002,
1250.1), (2004, 1223.5)]
>>> for item in cumulative_per_year.collect():
...     print(item)
...
```