**HIVE**

Please find the customer data set

1. **Write a program to find the count of customers for each profession**

**use training\_432538;**

**create table customer2(cust\_id int, firstname string,lastname string,age int, profession string)**

**>**

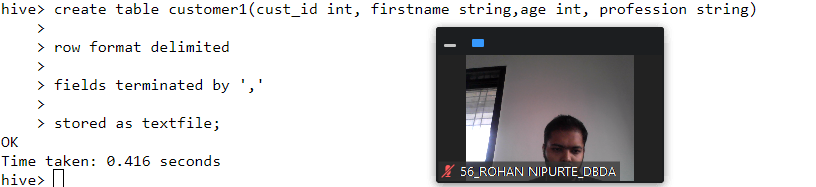
**> row format delimited**

**>**

**> fields terminated by ','**

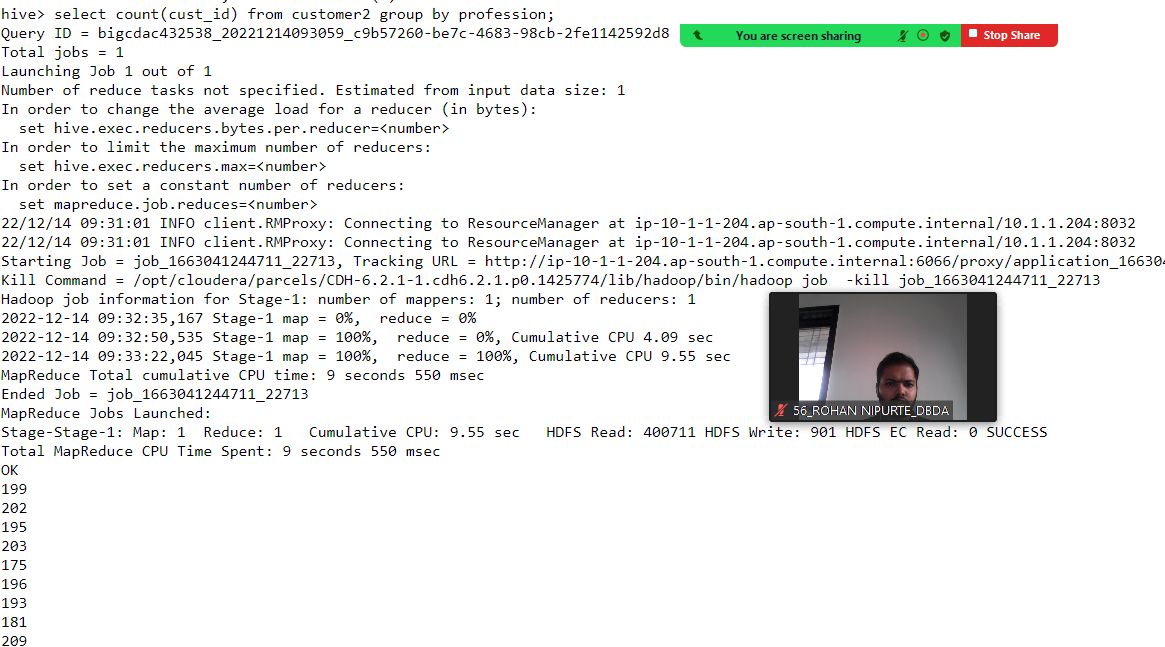
**>**

**> stored as textfile;**

****

load data local inpath 'custs1.txt' overwrite into table customer2;

**select count(cust\_id) from customer2 group by profession;**



**2) Write a program to find the top 10 products sales wise.**

create table transaction1(txn\_id int,txn\_date string,cust\_id int,amount double, category string, product string,city string,state string,spendb

y string)

>

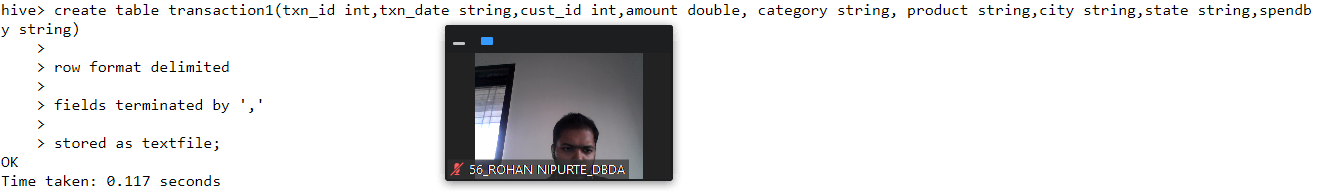
> row format delimited

>

> fields terminated by ','

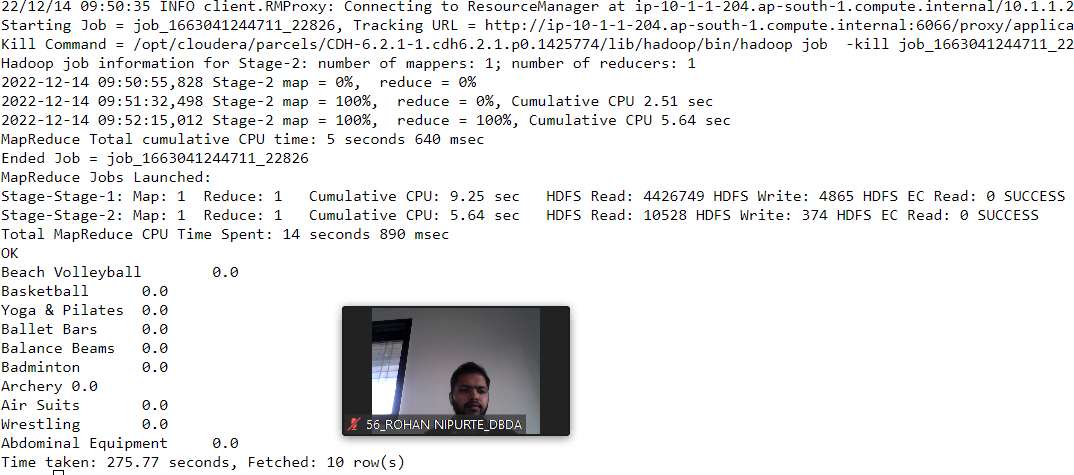
>

> stored as textfile;



load data local inpath 'txns1.txt' overwrite into table transaction1;

**select product,sum(spendby) as tot from transaction1 group by product order by tot desc limit 10;**

****

**3) Write a program to create partiioned table on category**

create table salesPartitioned(txn\_id int,txn\_date string,cust\_id int,amount double, product string,city string,state string,spendby string)

>

> partitioned by (category string)

>

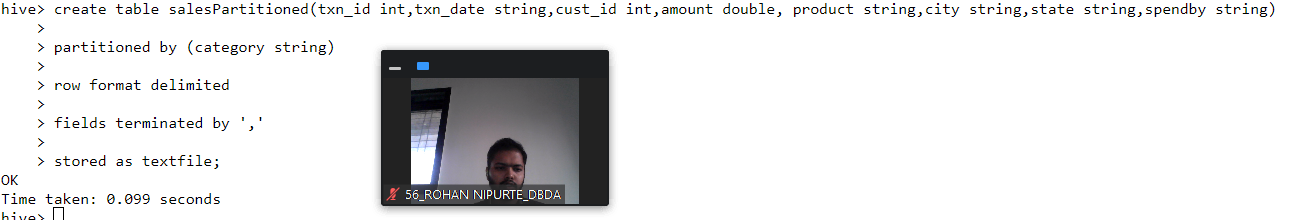
> row format delimited

>

> fields terminated by ','

>

> stored as textfile;



**SPARK**

**1) What was the highest number of people travelled in which year?**

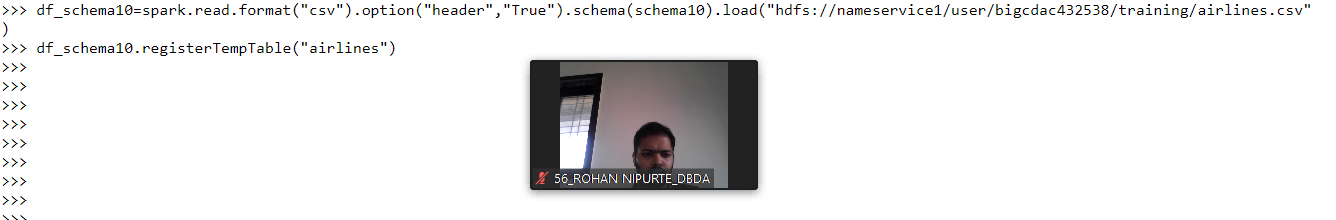
from pyspark.sql.types import StructType,StringType,IntegerType,DoubleType,LongType

schema10=StructType().add("Year",StringType(),True).add("Quarter",StringType(),True).add("Avg\_rev\_per\_seat",DoubleType(),True).add("Booked\_seats"

,IntegerType(),True)

df\_schema10=spark.read.format("csv").option("header","True").schema(schema10).load("hdfs://nameservice1/user/bigcdac432538/training/airlines.csv")

df\_schema10.registerTempTable("airlines")



df\_schema10.printSchema()

root

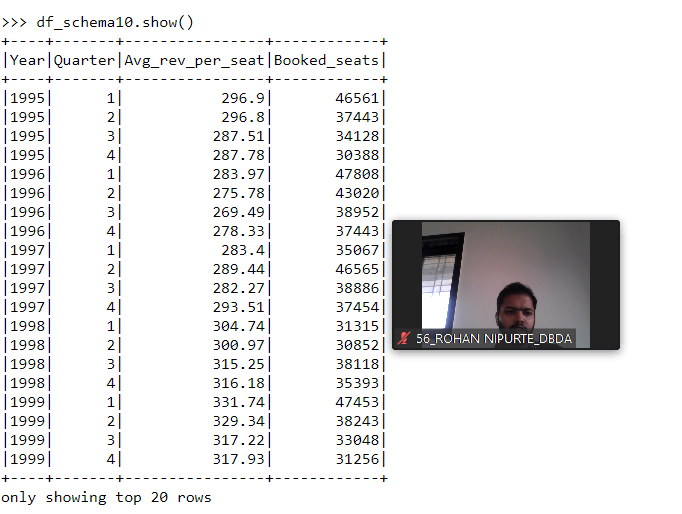
|-- Year: string (nullable = true)

|-- Quarter: string (nullable = true)

|-- Avg\_rev\_per\_seat: double (nullable = true)

|-- Booked\_seats: integer (nullable = true)

df\_schema10.show()

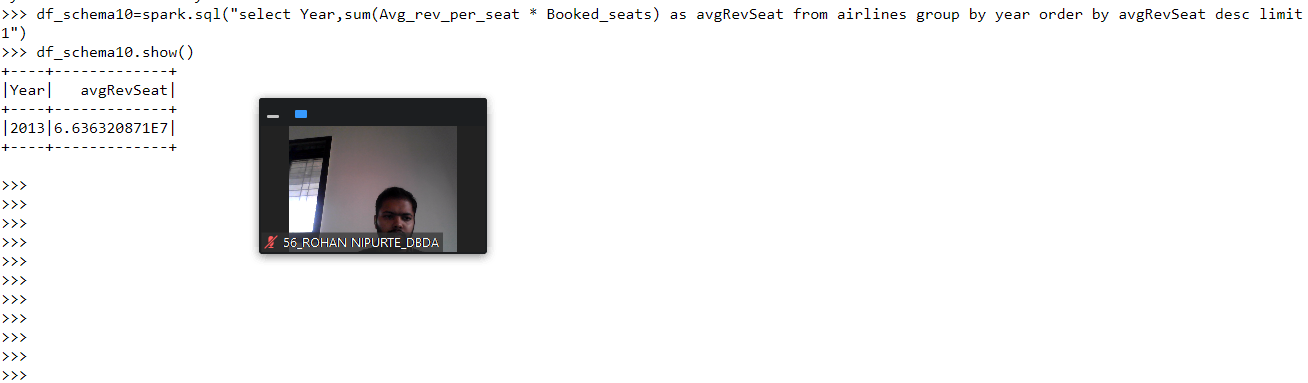


**df\_schema11=spark.sql("select year,sum(Avg\_rev\_per\_seat \* Booked\_seats) as avgRevSeat from airlines group by year order by avgRevSeat desc limit 1")**

**2) Identifying the highest revenue generation for which year**

**df\_schema10=spark.sql("select Year,sum(Avg\_rev\_per\_seat \* Booked\_seats) as avgRevSeat from airlines group by year order by avgRevSeat desc limit 1")**

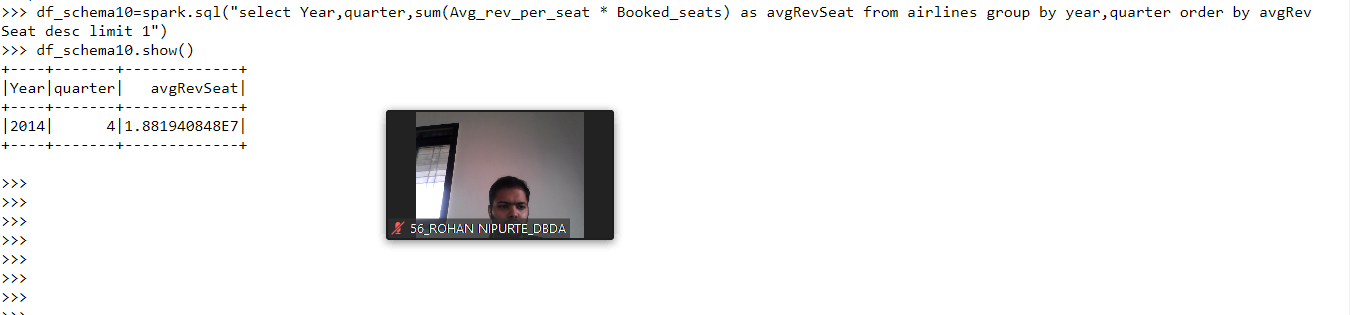
**df\_shema10.show()**

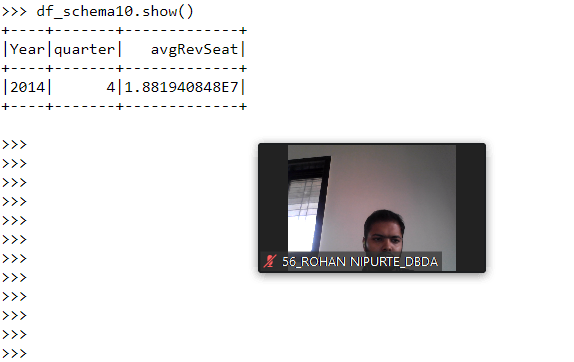
****

**3) Identifying the highest revenue generation for which year and quarter (Common group)**

**df\_schema10=spark.sql("select Year,quarter,sum(Avg\_rev\_per\_seat \* Booked\_seats) as avgRevSeat from airlines group by year,quarter order by avgRev**

**Seat desc limit 1")**

****

****

**MapReduce**