DATA INGESTION

* 1. 1. Create the necessary structure in a MySQL database using the steps mentioned below: a. Create a new database in MySQL with the name midproject
  2. b. Create a table in this database with the name census\_adult to store the input dataset

mysql -u edu\_labuser -pedureka -h dbserver.edu.cloudlab.com;

use labuser\_database;

CREATE TABLE new\_census\_adult(

age int

,workclass VARCHAR(50)

,fnlwgt int

,education VARCHAR(20)

,education\_num int

,marital\_status VARCHAR(20)

,occupation VARCHAR(50)

,relationship VARCHAR(50)

,race VARCHAR(50)

,sex VARCHAR(20)

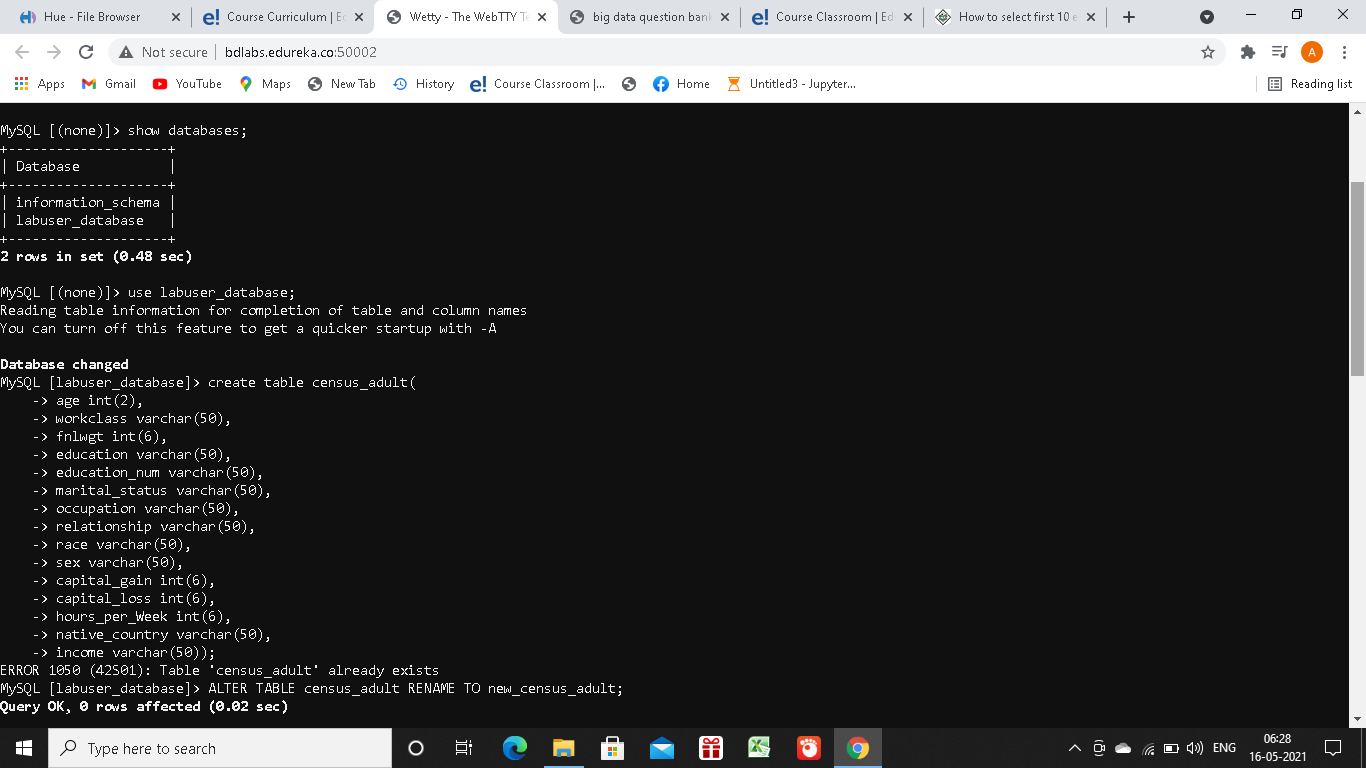
,capital\_gain int

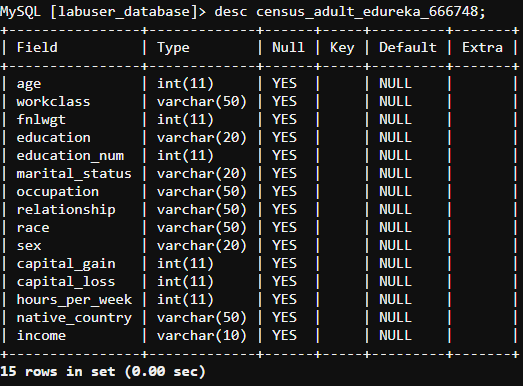
,capital\_loss int

,hours\_per\_week int

,native\_country VARCHAR(50)

,income VARCHAR(10));

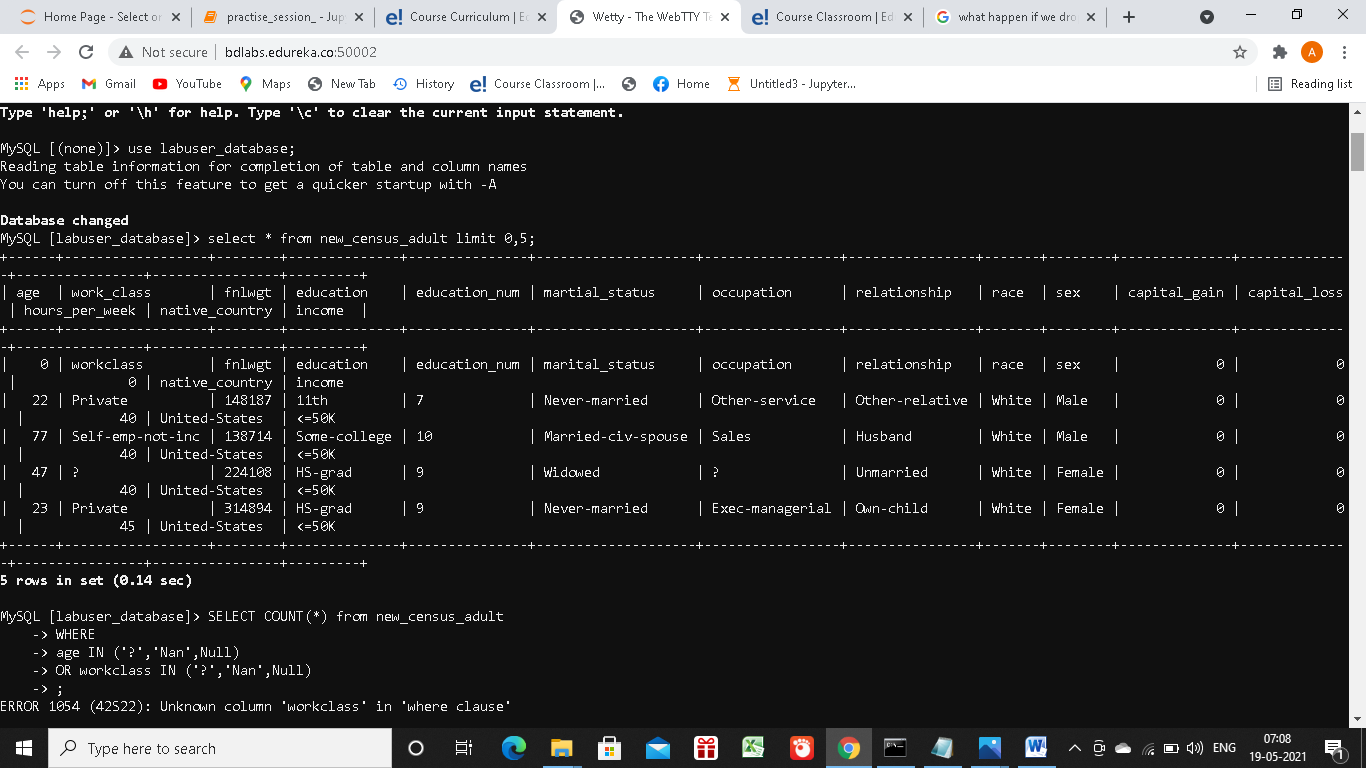
; 



* 1. c. Load the dataset into the table

LOAD DATA LOCAL INFILE '’/home/edureka\_/censusdata.csv' INTO TABLE census\_adult\_edureka\_666748 FIELDS TERMINATED BY ',';

* 1. d. Verify whether data is loaded properly
  2. select \* from new\_census\_adult limit 0,5;



* 1. e. Verify the table for unwanted data such as ‘?’,’Nan’ and ‘Null’

SELECT COUNT(\*) from new\_census\_adult

-> WHERE

-> age IN ('?','Nan','Null')

-> OR work\_class IN ('?','Nan','Null')

-> OR fnlwgt IN ('?','Nan','Null')

-> OR education IN ('?','Nan','Null')

-> OR education\_num IN ('?','Nan','Null')

-> OR martial\_status IN ('?','Nan','Null')

-> OR occupation IN ('?','Nan','Null')

-> OR relationship IN ('?','Nan','Null')

-> OR race IN ('?','Nan','Null')

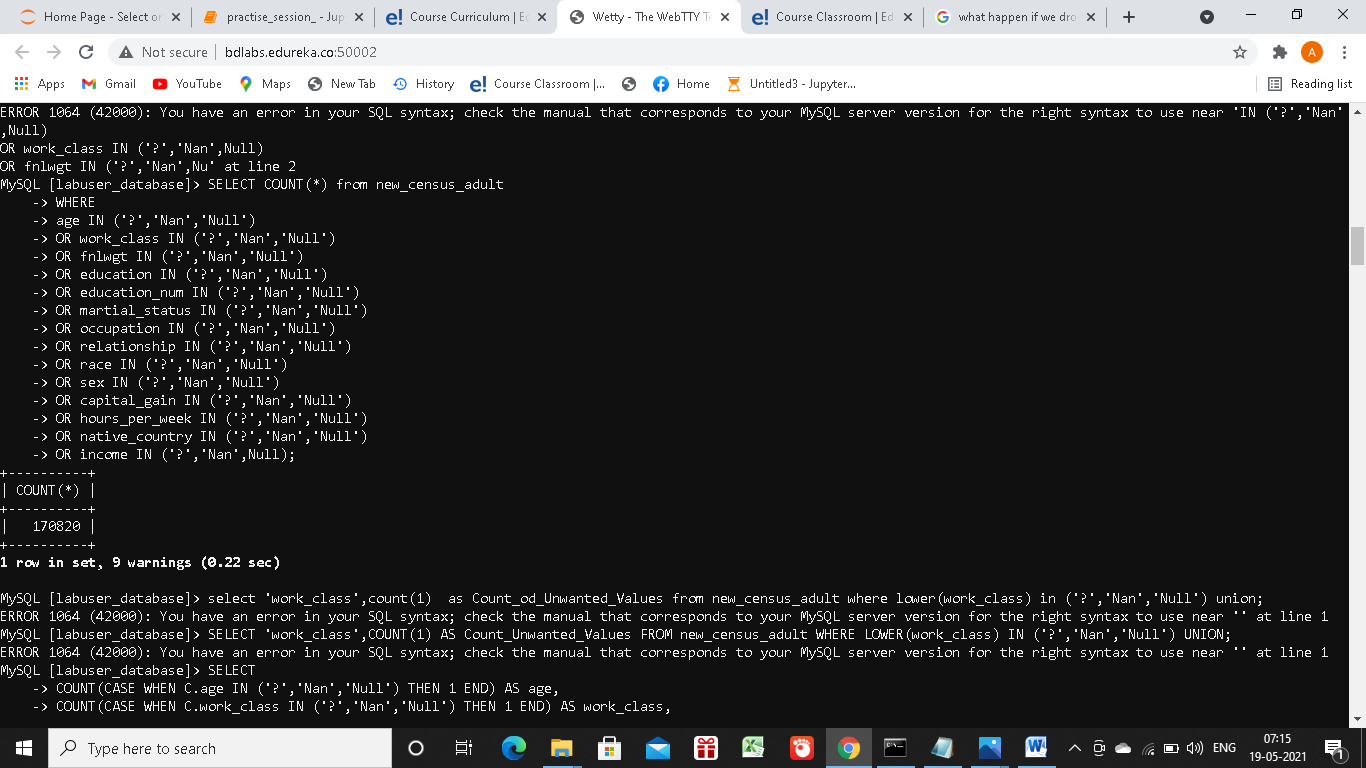
-> OR sex IN ('?','Nan','Null')

-> OR capital\_gain IN ('?','Nan','Null')

-> OR hours\_per\_week IN ('?','Nan','Null')

-> OR native\_country IN ('?','Nan','Null')

-> OR income IN ('?','Nan',Null);



* 1. f. Get the counts for the columns which contain unwanted data

SELECT

-> COUNT(CASE WHEN C.age IN ('?','Nan','Null') THEN 1 END) AS age,

-> COUNT(CASE WHEN C.work\_class IN ('?','Nan','Null') THEN 1 END) AS work\_class

, -> COUNT(CASE WHEN C.fnlwgt IN ('?','Nan','Null') THEN 1 END) AS fnlwgt,

-> COUNT(CASE WHEN C.education IN ('?','Nan','Null') THEN 1 END) AS education,

-> COUNT(CASE WHEN C.education\_num IN ('?','Nan','Null') THEN 1 END) AS education\_num,

-> COUNT(CASE WHEN C.martial\_status IN ('?','Nan','Null') THEN 1 END) AS martial\_status,

-> COUNT(CASE WHEN C.occupation IN ('?','Nan','Null') THEN 1 END) AS occupation,

-> COUNT(CASE WHEN C.relationship IN ('?','Nan','Null') THEN 1 END) AS relationship,

-> COUNT(CASE WHEN C.race IN ('?','Nan','Null') THEN 1 END) AS race,

-> COUNT(CASE WHEN C.sex IN ('?','Nan','Null') THEN 1 END) AS sex,

-> COUNT(CASE WHEN C.capital\_gain IN ('?','Nan','Null') AND C.capital\_gain !=0 THEN 1 END) AS capital\_gain,

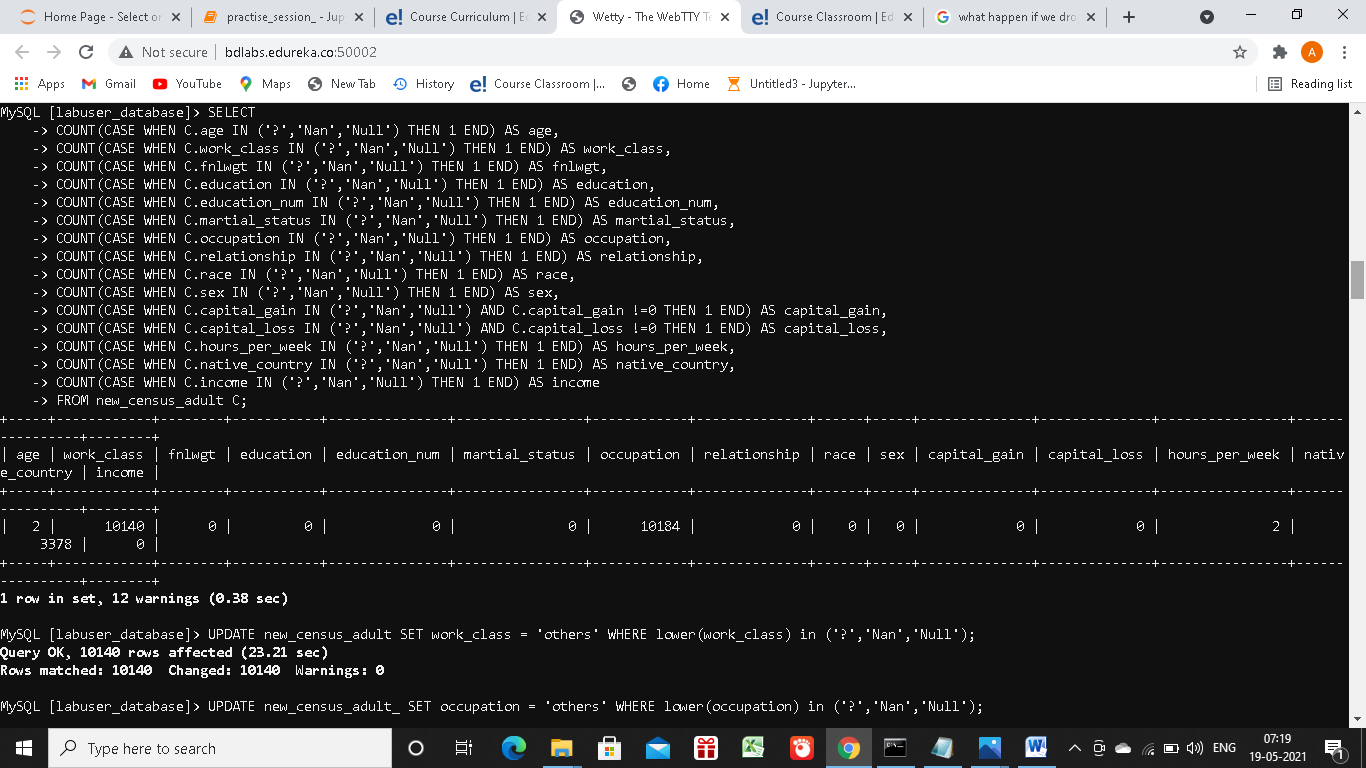
-> COUNT(CASE WHEN C.capital\_loss IN ('?','Nan','Null') AND C.capital\_loss !=0 THEN 1 END) AS capital\_loss,

-> COUNT(CASE WHEN C.hours\_per\_week IN ('?','Nan','Null') THEN 1 END) AS hours\_per\_week,

-> COUNT(CASE WHEN C.native\_country IN ('?','Nan','Null') THEN 1 END) AS native\_country,

-> COUNT(CASE WHEN C.income IN ('?','Nan','Null') THEN 1 END) AS income

-> FROM new\_census\_adult C;

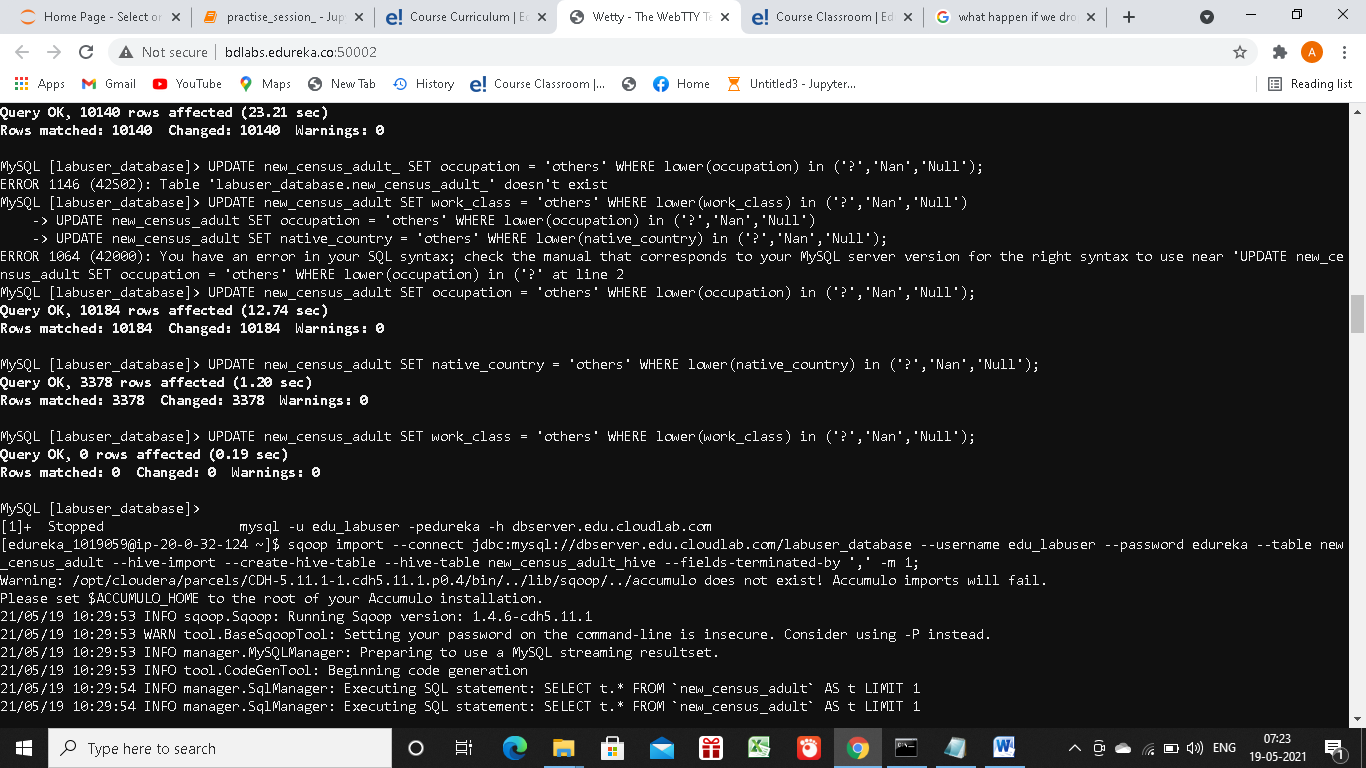


* 1. g. Clean the data by replacing the unwanted data with others

UPDATE new\_census\_adult SET occupation = 'others' WHERE lower(occupation) in ('?','Nan','Null');

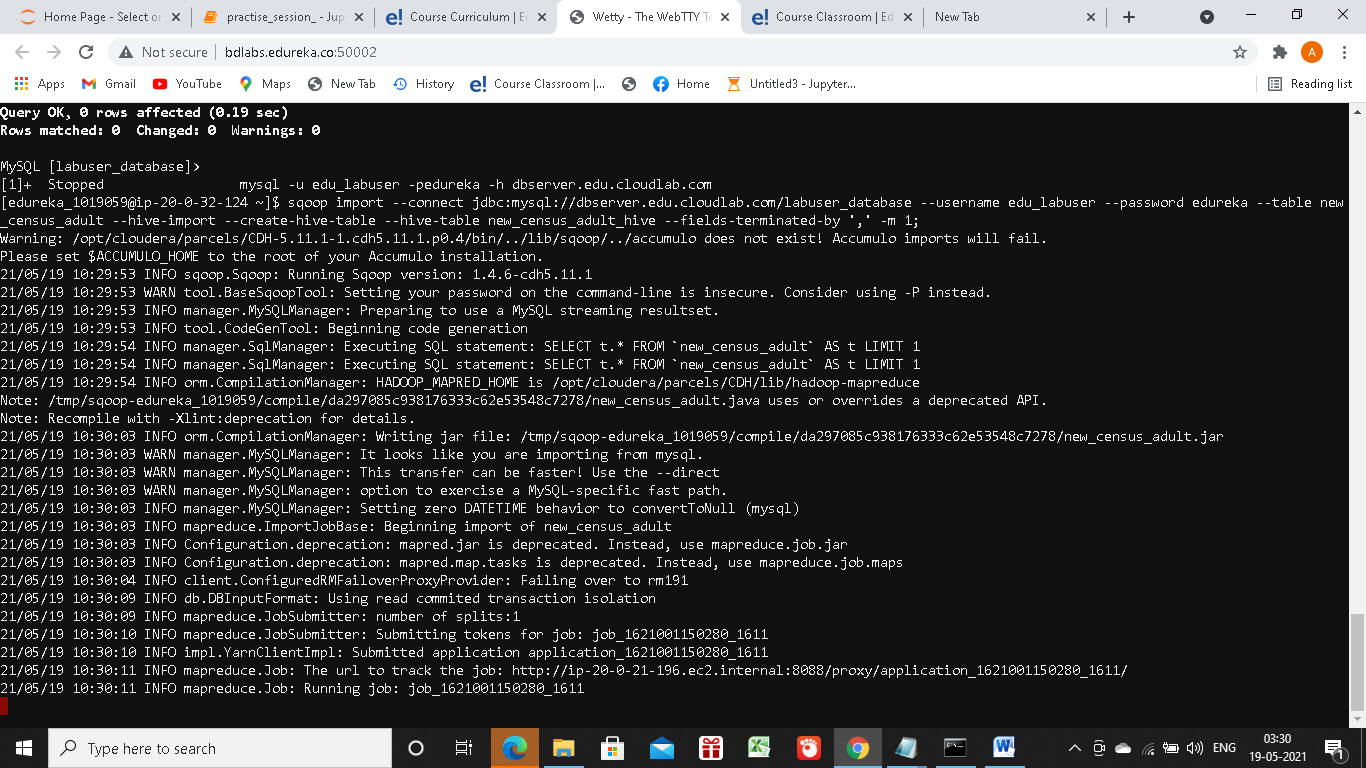
UPDATE new\_census\_adult SET native\_country = 'others' WHERE lower(native\_country) in ('?','Nan','Null');

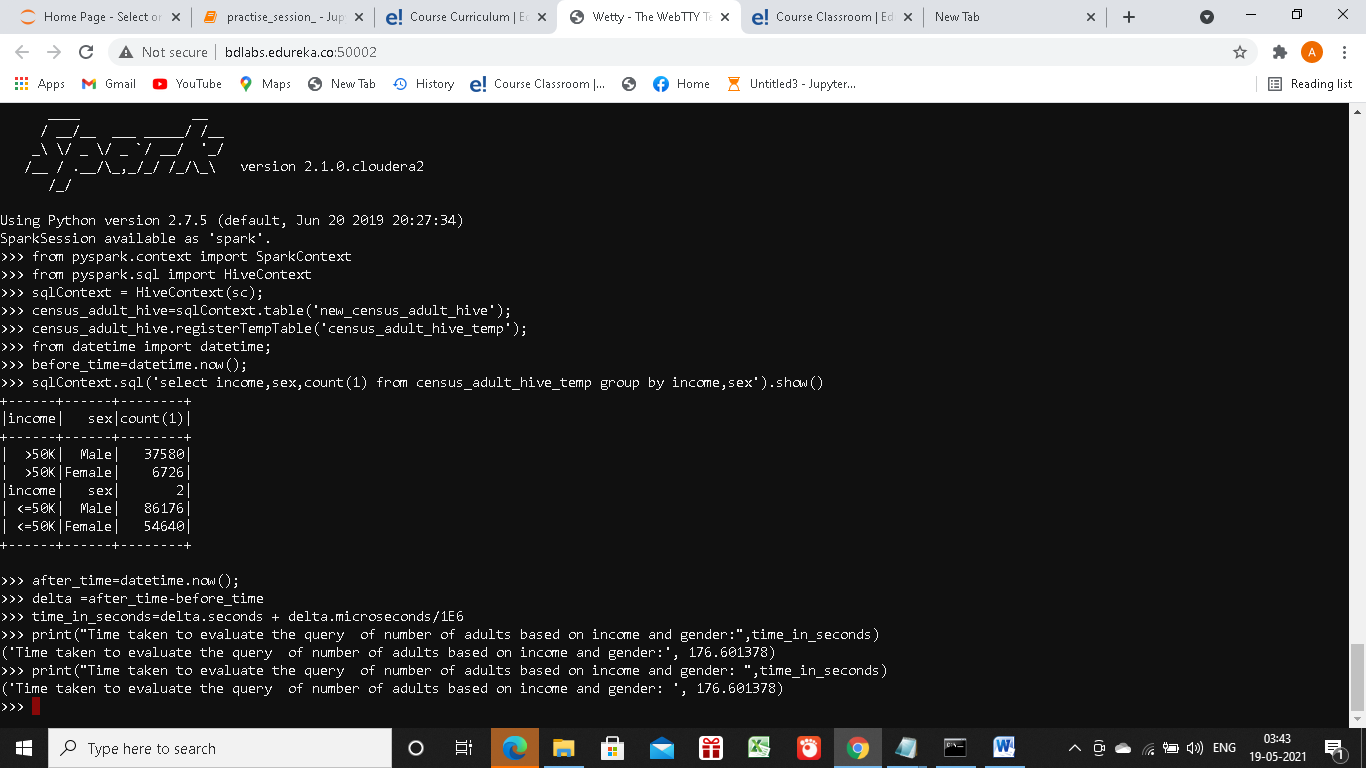
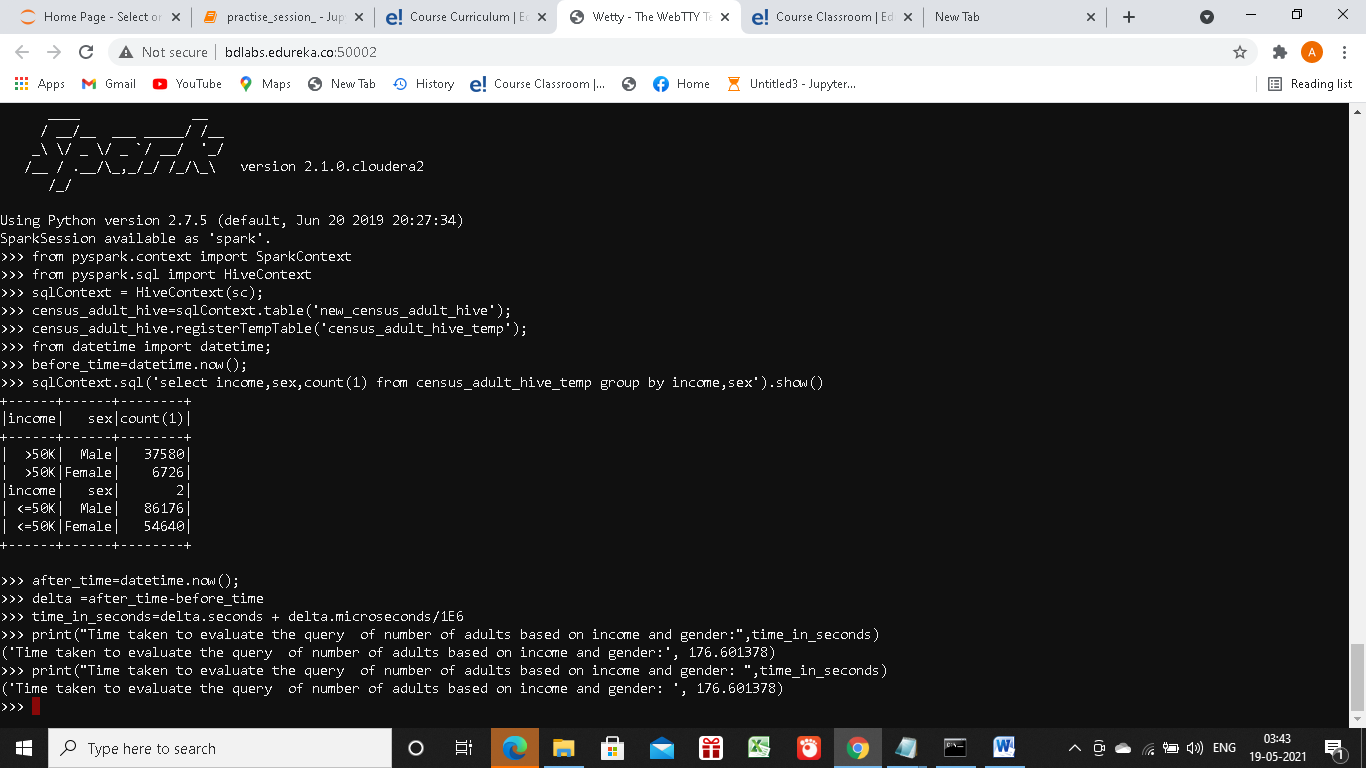
UPDATE new\_census\_adult SET work\_class = 'others' WHERE lower(work\_class) in ('?','Nan','Null');

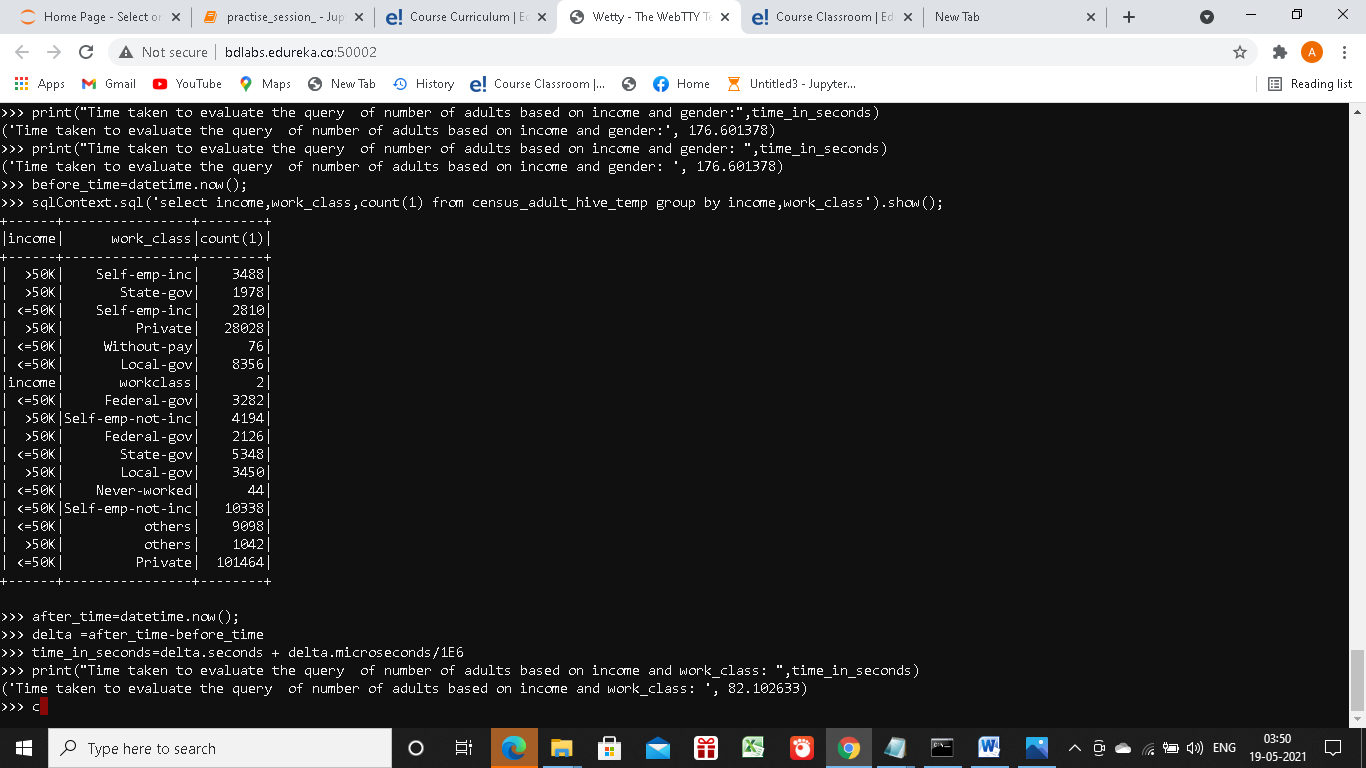


2. Import the above data from MySQL into a Hive table using Sqoop

sqoop import --connect jdbc:mysql://dbserver.edu.cloudlab.com/labuser\_database --username edu\_labuser --password edureka --table new\_census\_adult --hive-import --create-hive-table --hive-table new\_census\_adult\_hive --fields-terminated-by ',' -m 1;



* 1. 3. Connect to PySpark using web console to access the created Hive table. Perform the following queries and note the time taken for execution in each of the queries.
  2. from pyspark.context import SparkContext;
  3. from pyspark.sql import HiveContext;
  4. sqlContext = HiveContext(sc);
  5. census\_adult\_hive=sqlContext.table('new\_census\_adult\_hive');
  6. census\_adult\_hive.registerTempTable('census\_adult\_hive\_temp');
  7. 
  8. a. Query the table to get the number of adults based on income and gender
  9. from datetime import datetime;
  10. before\_time=datetime.now();
      + 1. sqlContext.sql('select income,sex,count(1) from census\_adult\_hive\_temp group by
        2. income,sex').show()
        3. after\_time=datetime.now();
        4. delta =after\_time-before\_time
        5. time\_in\_seconds=delta.seconds + delta.microseconds/1E6
        6. print("Time taken to evaluate the query of number of adults based on income and
        7. gender:",time\_in\_seconds)
        8. ('Time taken to evaluate the query of number of adults based on income and gender:',
        9. 176.601378)
  11. 
  12. b. Query the table to get the number of adults based on income and workclass
  13. before\_time=datetime.now();
  14. sqlContext.sql('select income,work\_class,count(1) from census\_adult\_hive\_temp group
  15. by income,work\_class').show();
  16. after\_time=datetime.now();
  17. delta =after\_time-before\_time
  18. time\_in\_seconds=delta.seconds + delta.microseconds/1E6
  19. print("Time taken to evaluate the query of number of adults based on income and
  20. work\_class: ",time\_in\_seconds)
  21. ('Time taken to evaluate the query of number of adults
  22. based on income and work\_class: ', 82.102633)



>>pyspark2

>>from pyspark.context import SparkContext

>>from pyspark.sql import HiveContext

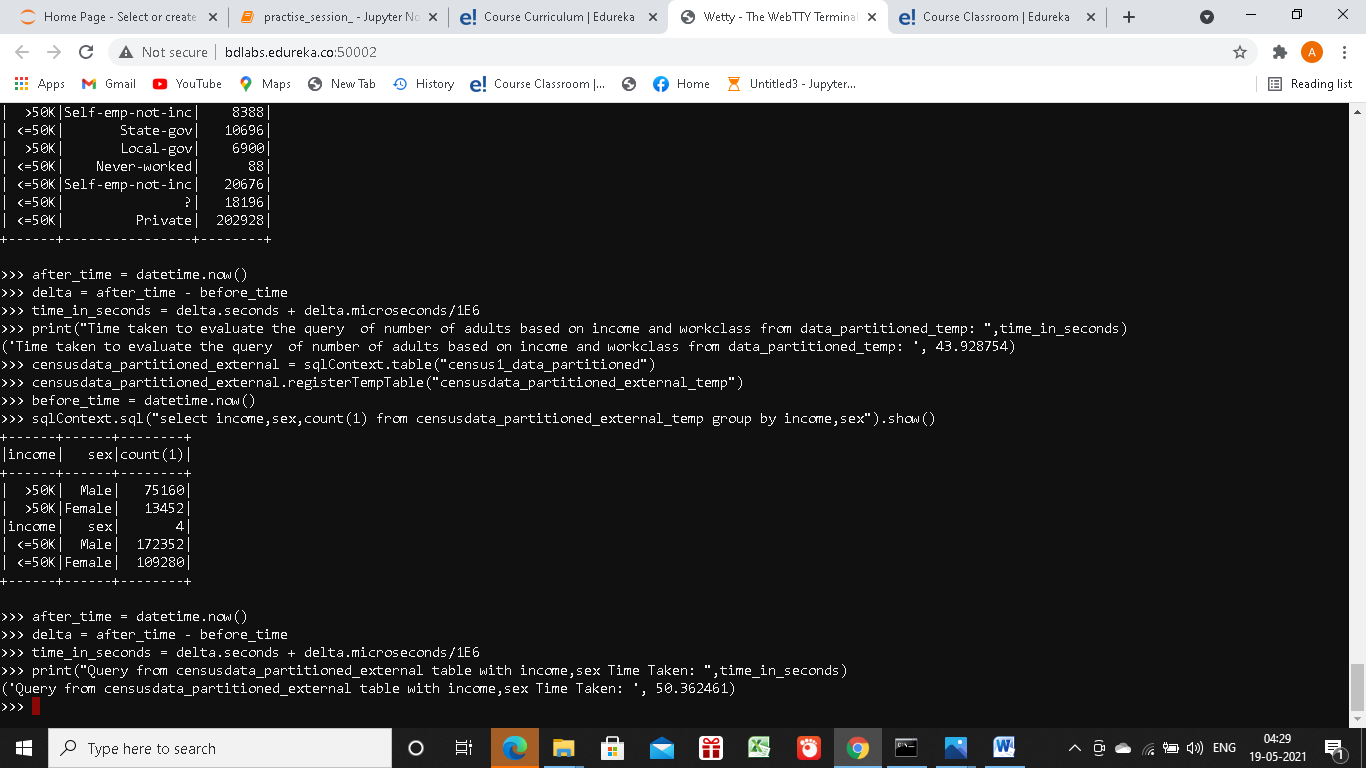
>>sqlContext = HiveContext(sc)

4. Access the following two tables created as part of Problem 1 (HDFS and Hive) and perform the steps as mentioned below:

a. Access Hive External Table with partition

censusdata\_partitioned\_external = sqlContext.table("census1\_data\_partitioned")

censusdata\_partitioned\_external.registerTempTable("censusdata\_partitioned\_external\_temp")



i. Query the table to get the number of adults based on income and gender

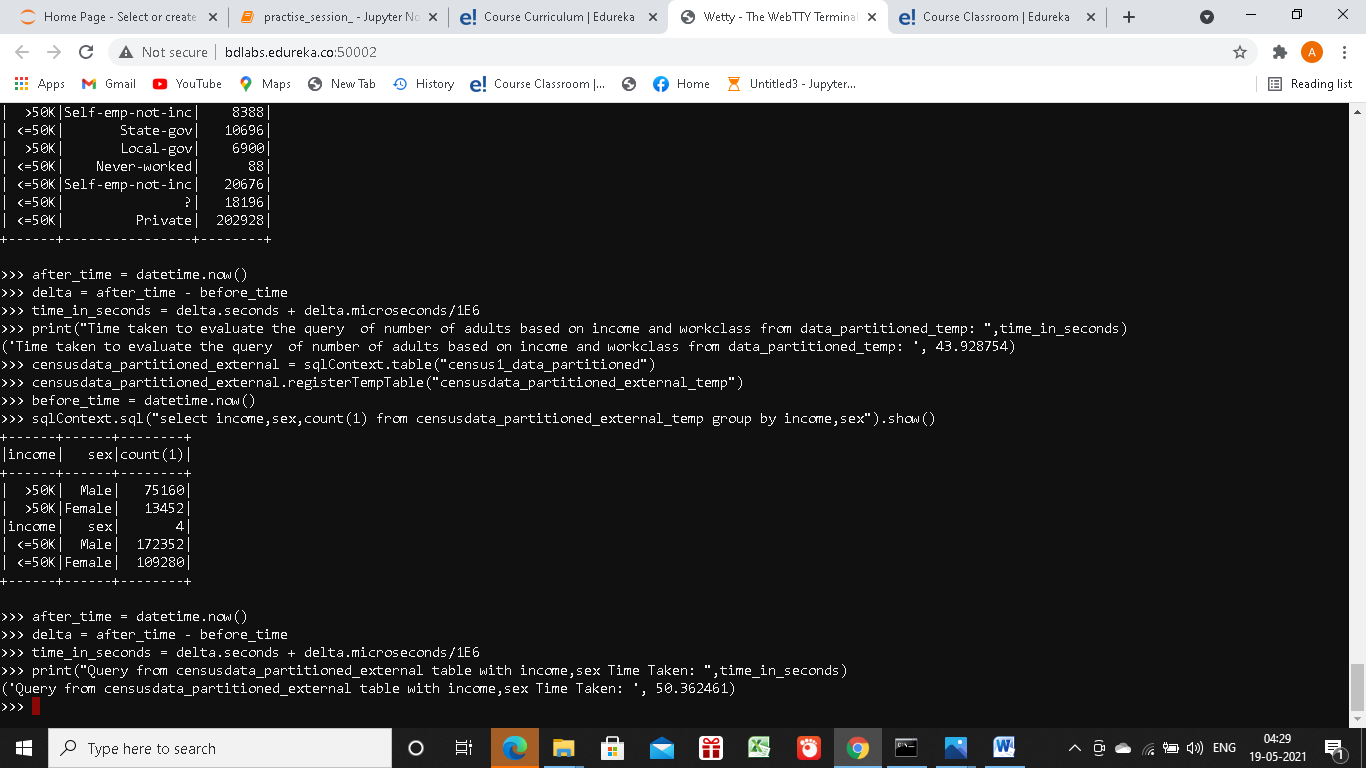
before\_time = datetime.now()

sqlContext.sql("select income,sex,count(1) from censusdata\_partitioned\_external\_temp group by income,sex").show

after\_time = datetime.now()

delta = after\_time - before\_time

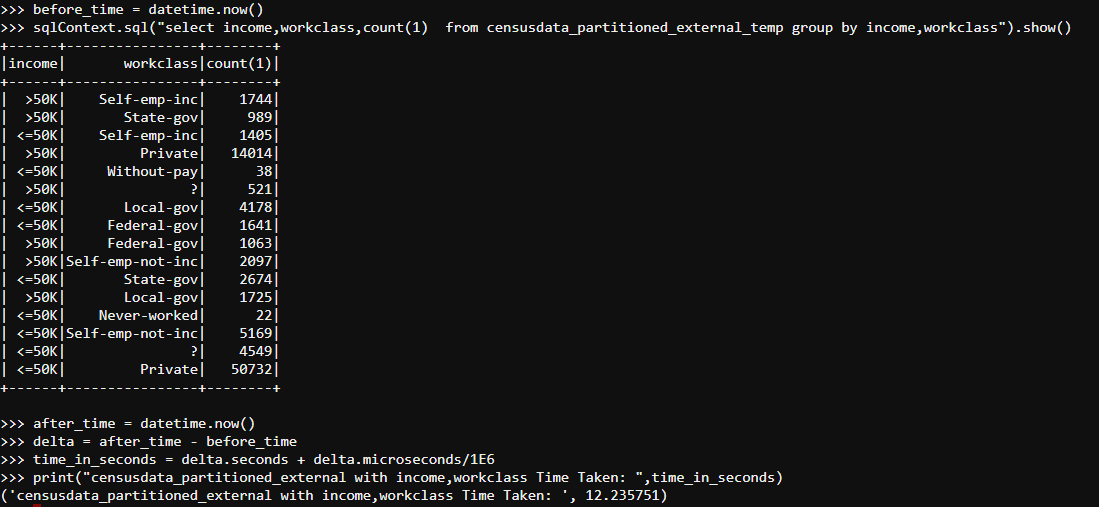
time\_in\_seconds = delta.seconds + delta.microseconds/1E6 print("censusdata\_partitioned\_external with income,sex Time Taken: ",time\_in\_seconds)



ii. Query the table to get the number of adults based on income and workclass

before\_time = datetime.now()

sqlContext.sql("select income,workclass,count(1) from censusdata\_partitioned\_external\_temp group by income,workclass").show()

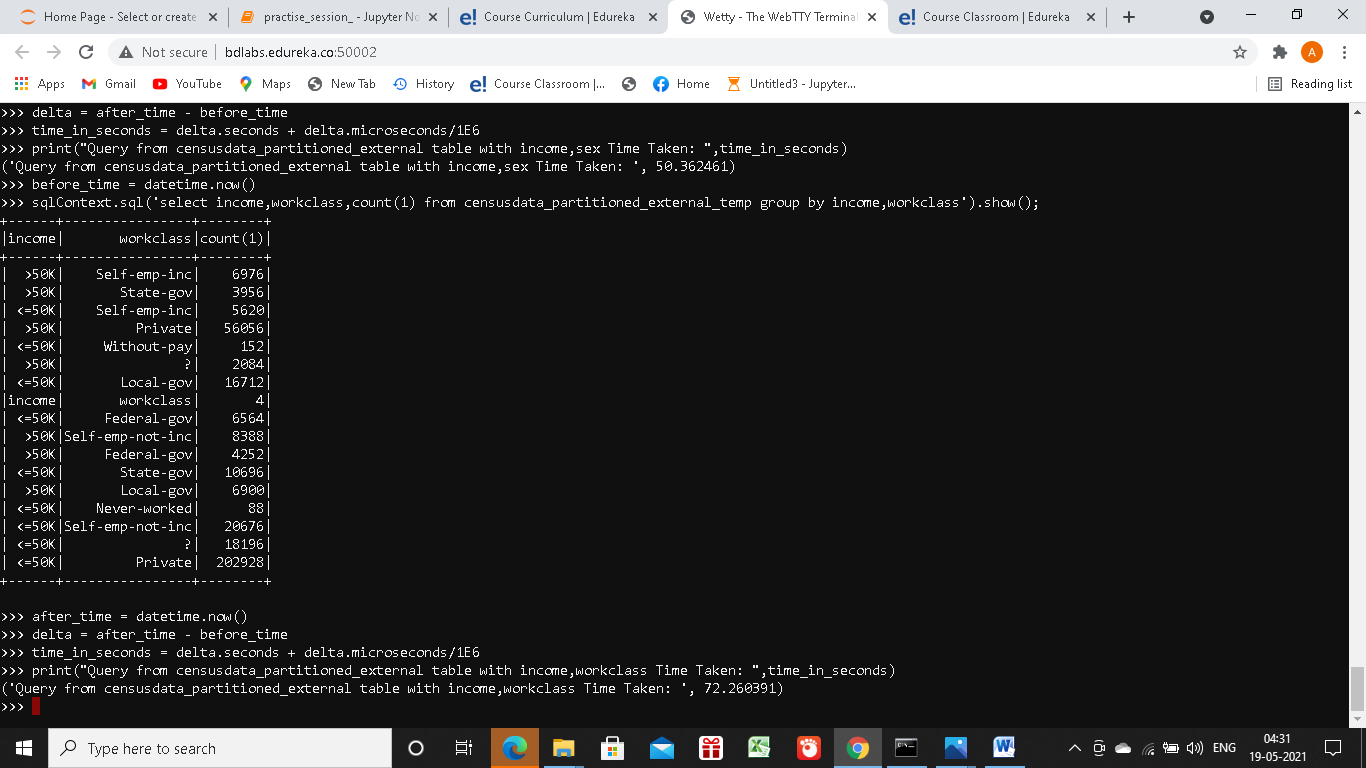


after\_time = datetime.now()

delta = after\_time - before\_time

time\_in\_seconds = delta.seconds + delta.microseconds/1E6

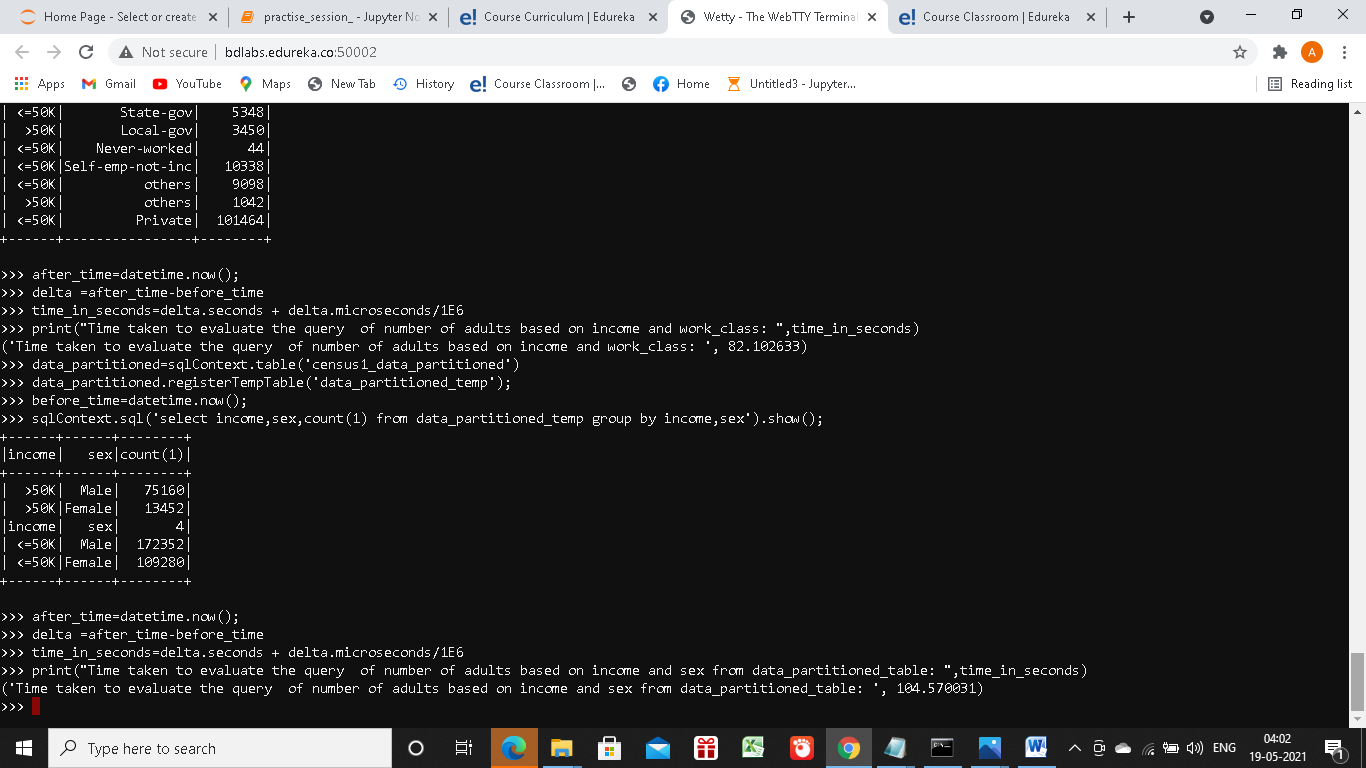
print("censusdata\_partitioned\_external with income,workclass Time Taken: ",time\_in\_seconds)



b. Access Hive Internal Table with Partition

data\_partitioned=sqlContext.table('census1\_data\_partitioned')

data\_partitioned.registerTempTable('data\_partitioned\_temp');



i. Query the table to get the number of adults based on income and gender

before\_time = datetime.now()

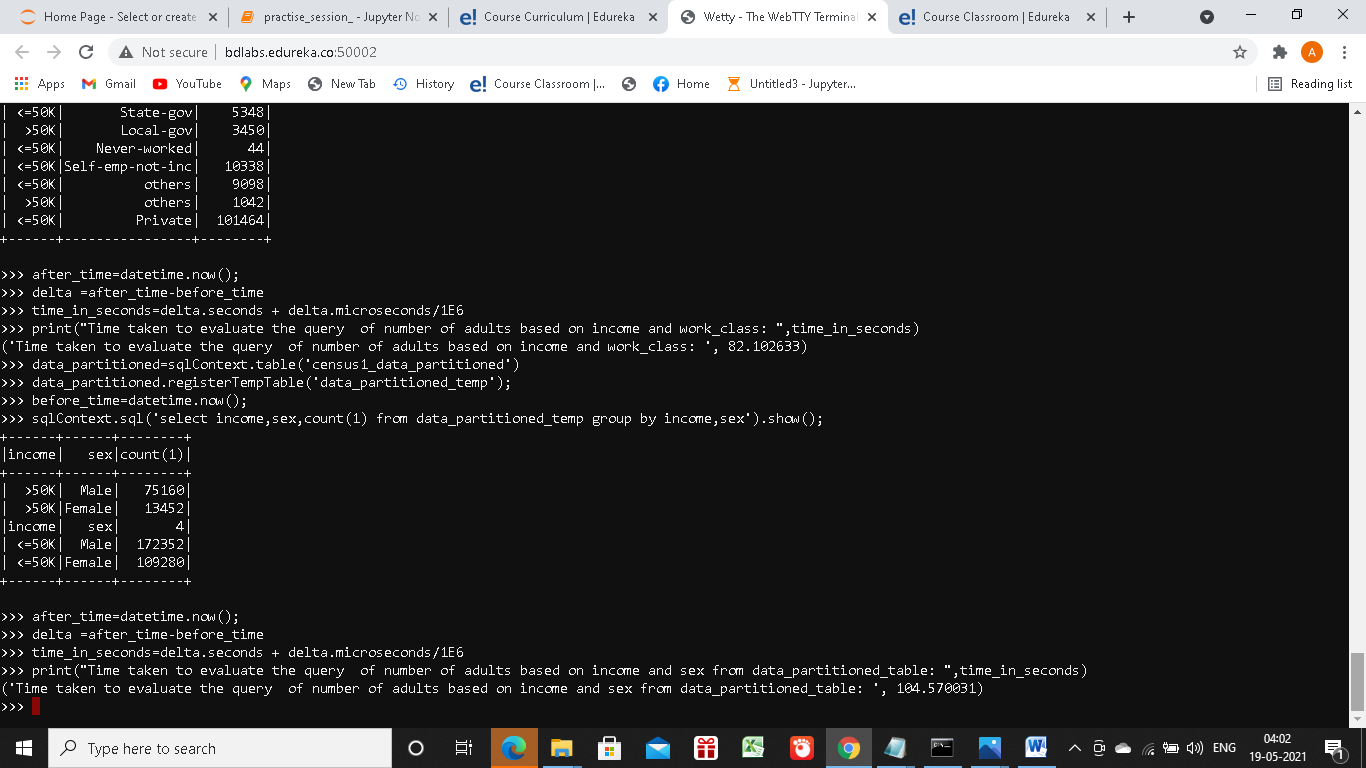
sqlContext.sql("select income,sex,count(1) data\_partitioned\_temp group by income,sex").show()

after\_time = datetime.now()

delta = after\_time - before\_time

time\_in\_seconds = delta.seconds + delta.microseconds/1E6

print("censusdata\_partitioned with income,sex Time Taken: ",time\_in\_seconds)



ii. Query the table to get the number of adults based on income and workclass

before\_time = datetime.now()

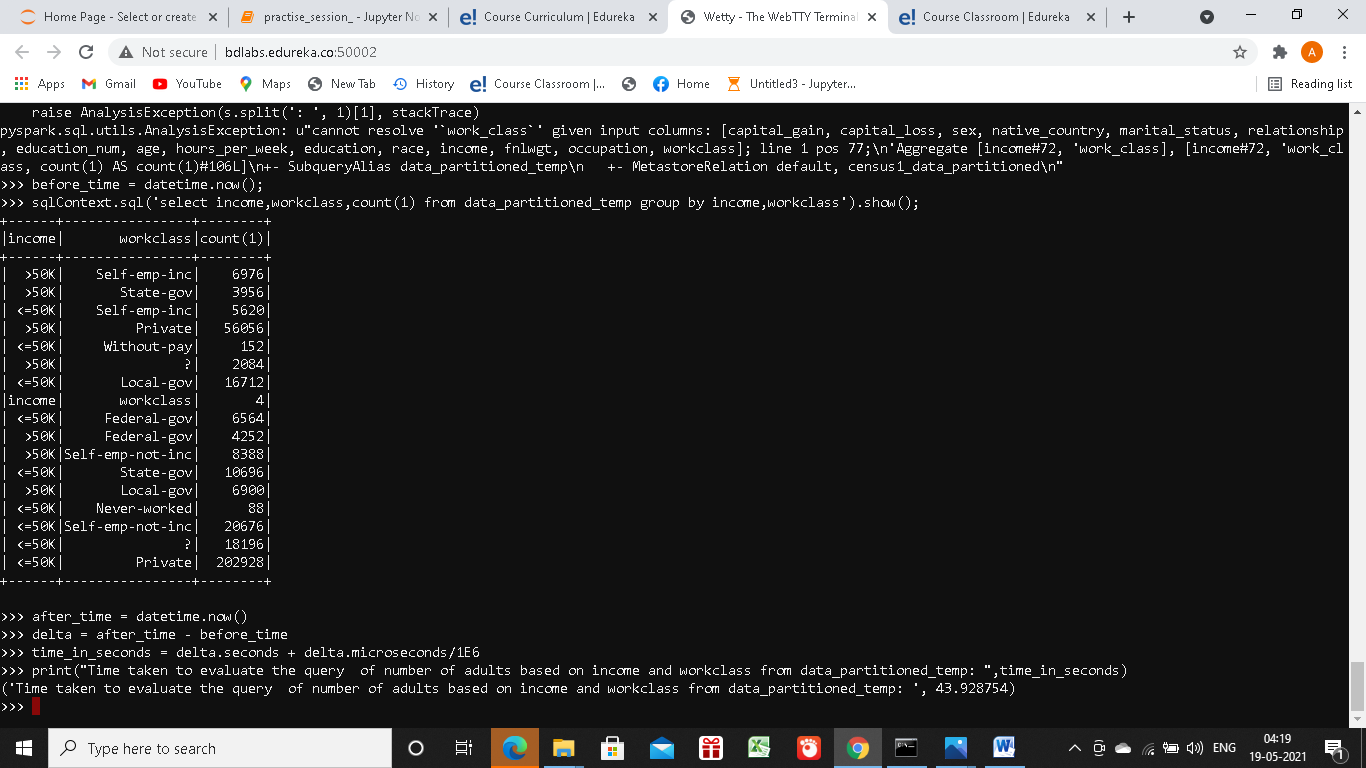
sqlContext.sql("select income,workclass,count(1) from data\_partitioned\_temp group by income,workclass").show()

after\_time = datetime.now()

delta = after\_time - before\_time

time\_in\_seconds = delta.seconds + delta.microseconds/1E6

print("censusdata\_partitioned income,workclass Time Taken: ",time\_in\_seconds)



Make a note of the time taken for getting the result in comparison with the time taken to get results with Hive.

|  |  |  |  |
| --- | --- | --- | --- |
| OBSERVATIONS IN BOTH QUERIES WRT TIME TAKEN IN THE 4 TABLES IN SECONDS | | | |
| Query | SQL Table | External Partitioned Table | Internal Partitioned Table |
| Query1 | **176.06** | **50.36** | **104.57** |
| Query2 | **82.03** | **72.26** | **44.00** |

5. Comment on the time taken for executing these commands using Spark as compared to the time taken for execution in Hive (Problem Statement 1).

Comment :

1. When we use Partition column in Query, that reduces query time significantly in case of partitioned tables.

2.Query on Hive tables is faster than sql tables.(Especially when the query is performed on a partition column)