2. DATA INGESTION

Problem Statement 2

In a similar scenario as above, the data is available in a MySQL database. Due to the inefficiency of RDBMS systems to store and analyze Big Data, it is recommended that we move the data to the Hadoop Ecosystem.

Ingest the data from MySQL database into Hive using Sqoop. Data pipeline needs to be created to ingest data from an RDBMS into Hadoop Cluster and then load data into Hive.

To make the analysis faster, use Spark on top of Hive after getting data into the Hadoop cluster. Using Spark, query different tables from Hive to analyze the dataset.

**Steps to be performed:**

1 Create the necessary structure in a MySQL database using the steps mentioned below:

1. Create a new database in MySQL with the name ***midproject***

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

create database midproject;

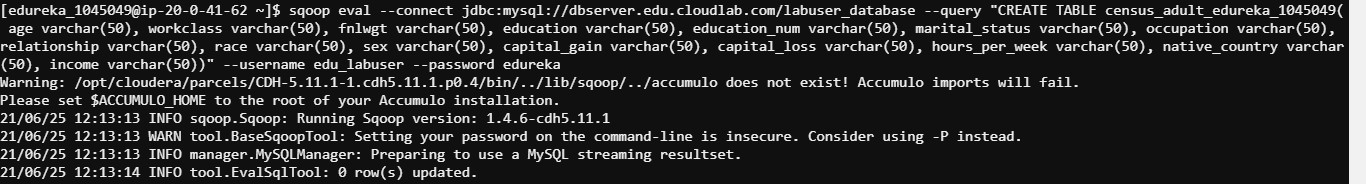
use midproject;

1. Create a table in this database with the name ***census\_adult*** to store the input dataset

sqoop eval --connect jdbc:mysql://dbserver.edu.cloudlab.com/labuser\_database --query "CREATE TABLE census\_adult\_edureka\_1045049(

age varchar(50), workclass varchar(50), fnlwgt varchar(50), education varchar(50), education\_num varchar(50), marital\_status varchar(50), occupation varchar(50),

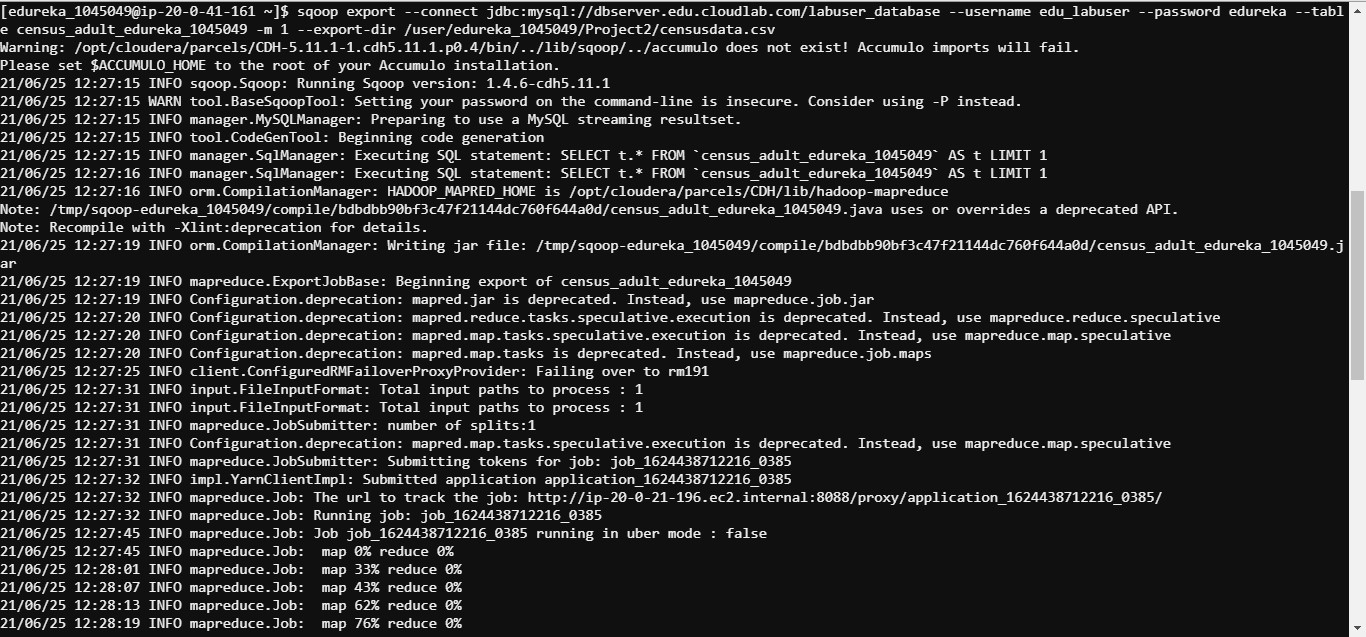
relationship varchar(50), race varchar(50), sex varchar(50), capital\_gain varchar(50), capital\_loss varchar(50), hours\_per\_week varchar(50), native\_country varchar(50), income varchar(50))" --username edu\_labuser --password edureka

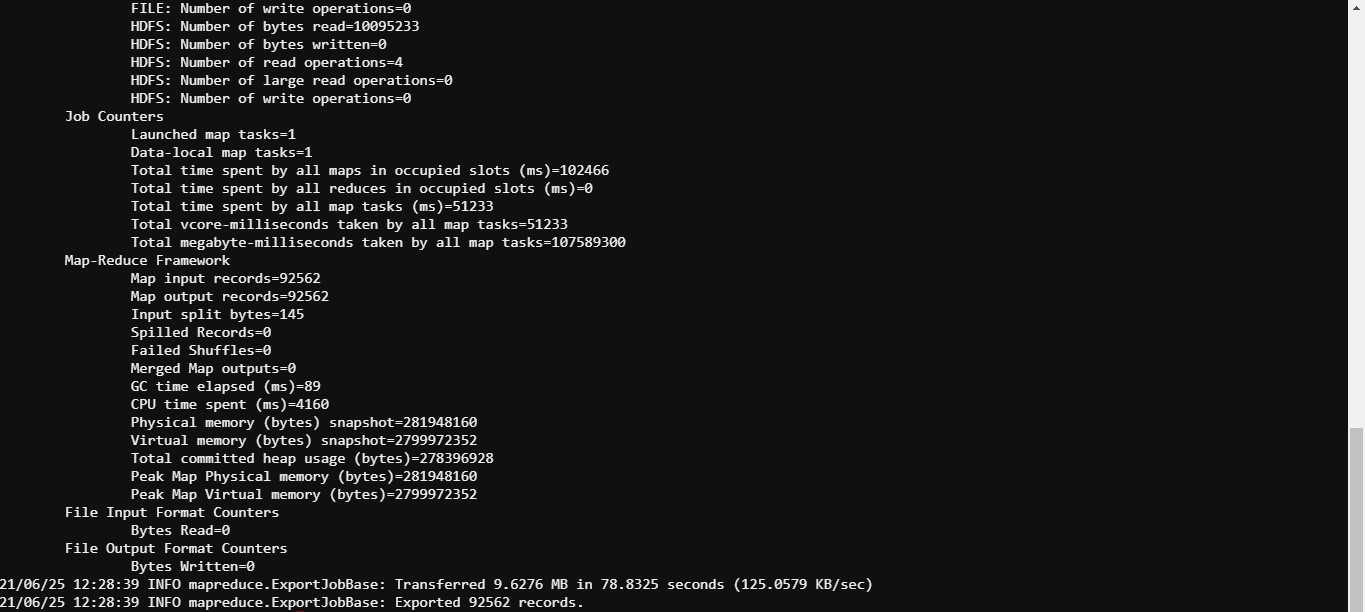


1. Load the dataset into the table

sqoop export --connect jdbc:mysql://dbserver.edu.cloudlab.com/labuser\_database --username edu\_labuser --password edureka --table

census\_adult\_edureka\_1045049 -m 1 --export-dir /user/edureka\_1045049/Project2/censusdata.csv





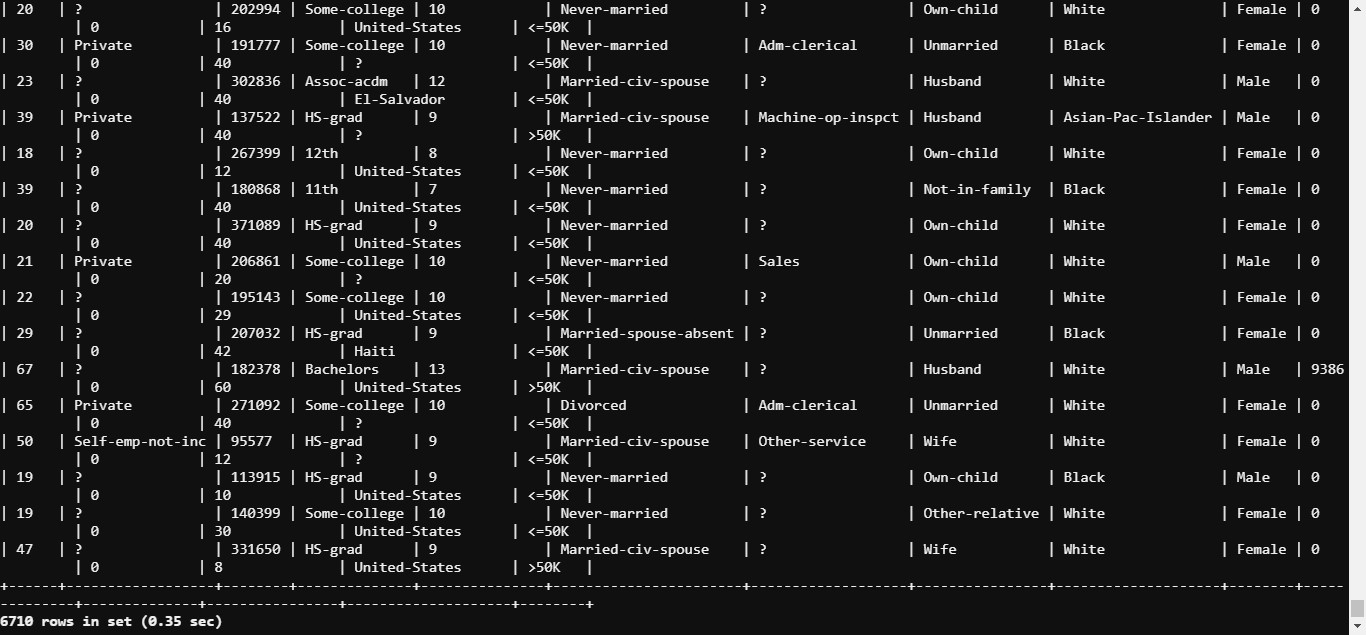
1. Verify whether data is loaded properly

select \* from census\_adult\_edureka\_1045049 limit 10;



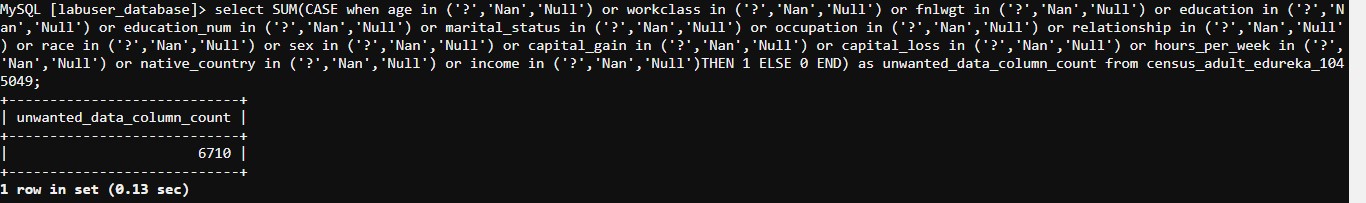
1. Verify the table for unwanted data such as ‘?’,’Nan’ and ‘Null’

select \* from census\_adult\_edureka\_1045049 where age in ('?','Nan','Null') or workclass in ('?','Nan','Null') or fnlwgt in ('?','Nan','Null') or education in ('?','Nan','Null') or education\_num in ('?','Nan','Null') or marital\_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 capital\_loss in ('?','Nan','Null') or hours\_per\_week in ('?','Nan','Null') or native\_country in ('?','Nan','Null') or income in ('?','Nan','Null');



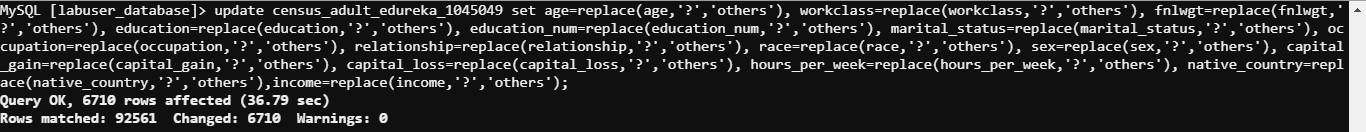
1. Get the counts for the columns which contain unwanted data

select SUM(CASE when age in ('?','Nan','Null') or workclass in ('?','Nan','Null') or fnlwgt in ('?','Nan','Null') or education in ('?','Nan','Null') or education\_num in ('?','Nan','Null') or marital\_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 capital\_loss in ('?','Nan','Null') or hours\_per\_week in ('?','Nan','Null') or native\_country in ('?','Nan','Null') or income in ('?','Nan','Null')THEN 1 ELSE 0 END) as unwanted\_data\_column\_count from census\_adult\_edureka\_1045049;



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

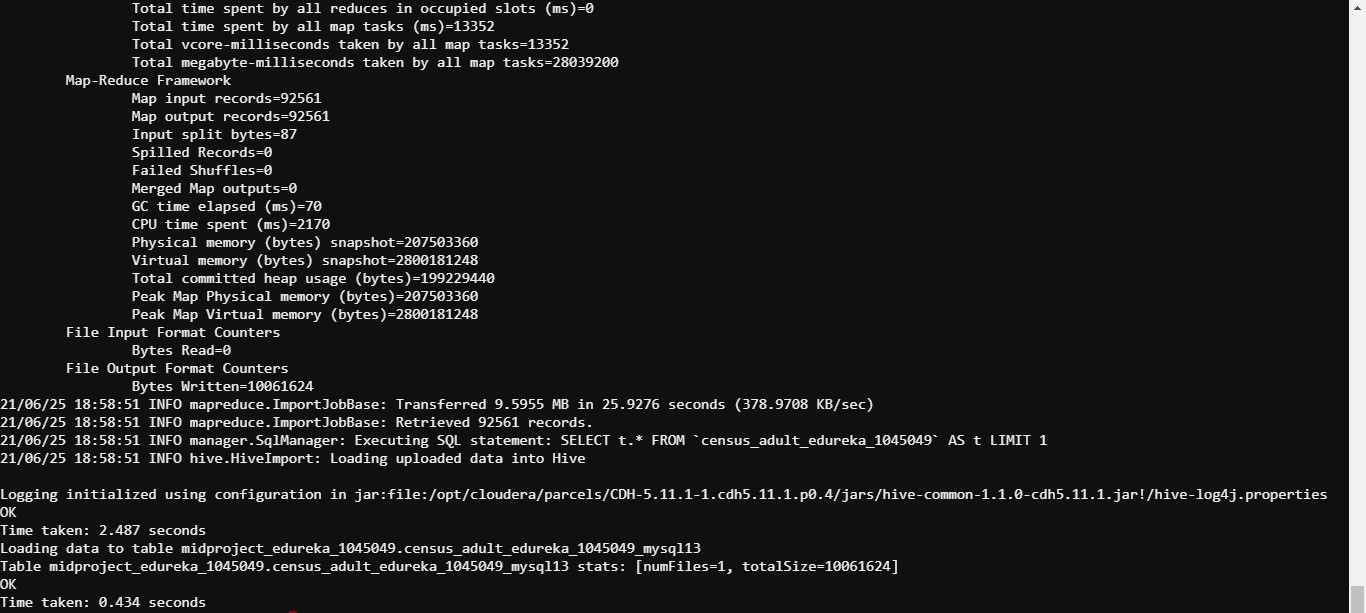
update census\_adult\_edureka\_1045049 set age=replace(age,'?','others'), workclass=replace(workclass,'?','others'), fnlwgt=replace(fnlwgt,'?','others'), education=replace(education,'?','others'), education\_num=replace(education\_num,'?','others'), marital\_status=replace(marital\_status,'?','others'), occupation=replace(occupation,'?','others'), relationship=replace(relationship,'?','others'), race=replace(race,'?','others'), sex=replace(sex,'?','others'), capital\_gain=replace(capital\_gain,'?','others'), capital\_loss=replace(capital\_loss,'?','others'), hours\_per\_week=replace(hours\_per\_week,'?','others'), native\_country=replace(native\_country,'?','others'), income=replace(income,'?','others');



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

census\_adult\_edureka\_1045049 --target-dir /user/edureka\_1045049/Project2/census\_adult --fields-terminated-by "," --hive-import --create-hive-table --hive-table midproject\_edureka\_1045049.census\_adult\_edureka\_1045049\_mysql13 -m 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.

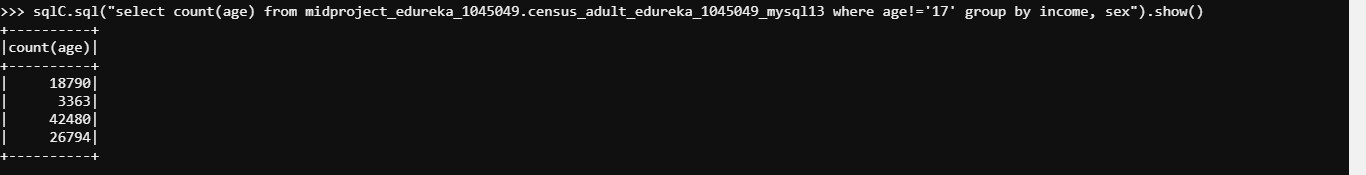
1. Query the table to get the number of adults based on income and gender

>>> from pyspark.context import SparkContext

>>> from pyspark.sql import HiveContext

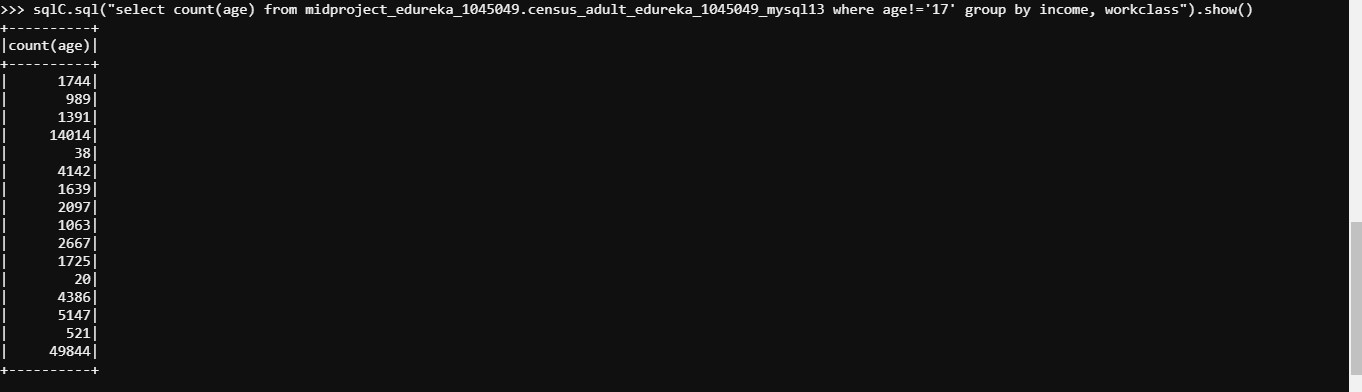
>>> sqlC = HiveContext(sc)

>>> sqlC.sql("select count(age) from midproject\_edureka\_1045049.census\_adult\_edureka\_1045049\_mysql13 where age!='17' group by income, sex").show()



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

>>> sqlC.sql("select count(age) from midproject\_edureka\_1045049.census\_adult\_edureka\_1045049\_mysql13 where age!='17' group by income, workclass").show()



**Hint:** To access Hive tables using Spark console, use the following commands:

*>>pyspark2*

*>>from pyspark.context import SparkContext*

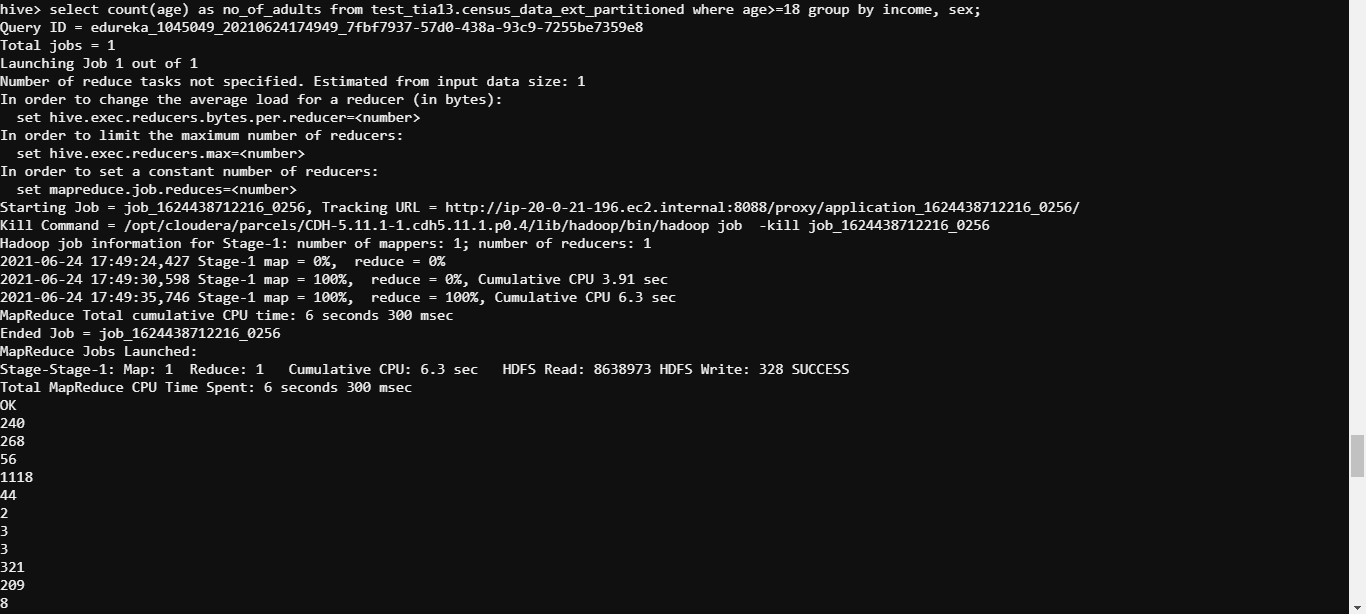
*>>from pyspark.sql import HiveContext*

*>>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:

1. Access Hive External Table with partition
   1. Query the table to get the number of adults based on income and gender

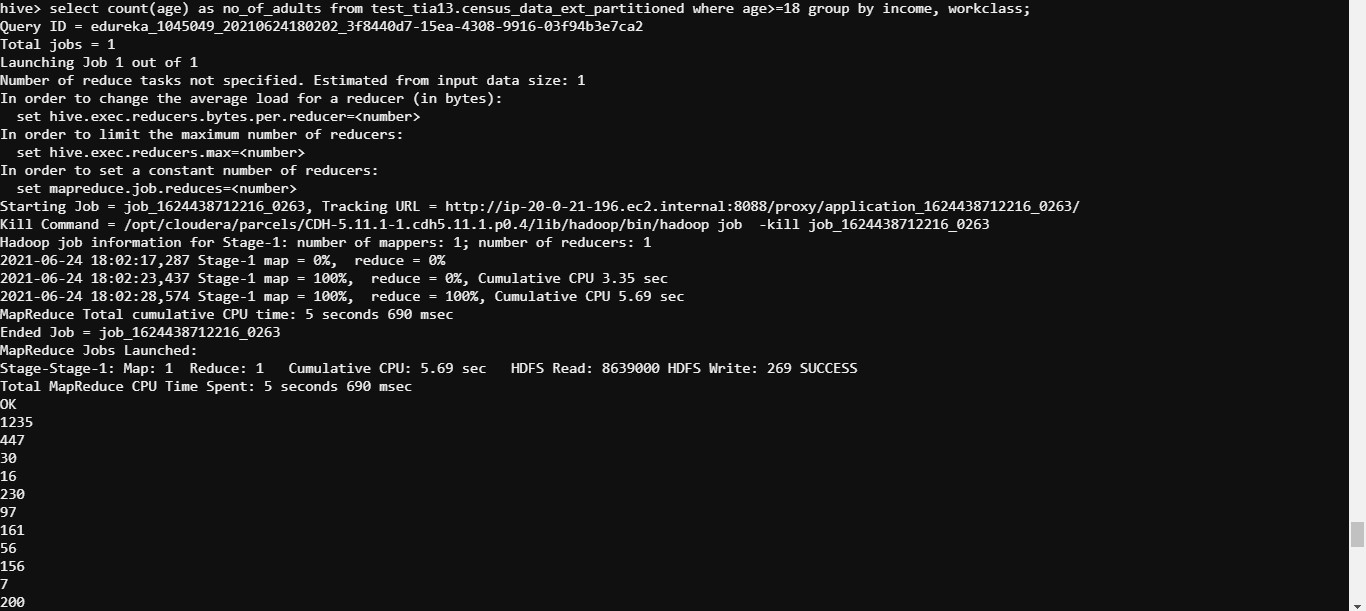
select count(age) as no\_of\_adults from test\_tia13.census\_data\_ext\_partitioned where age>=18 group by income, sex;





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

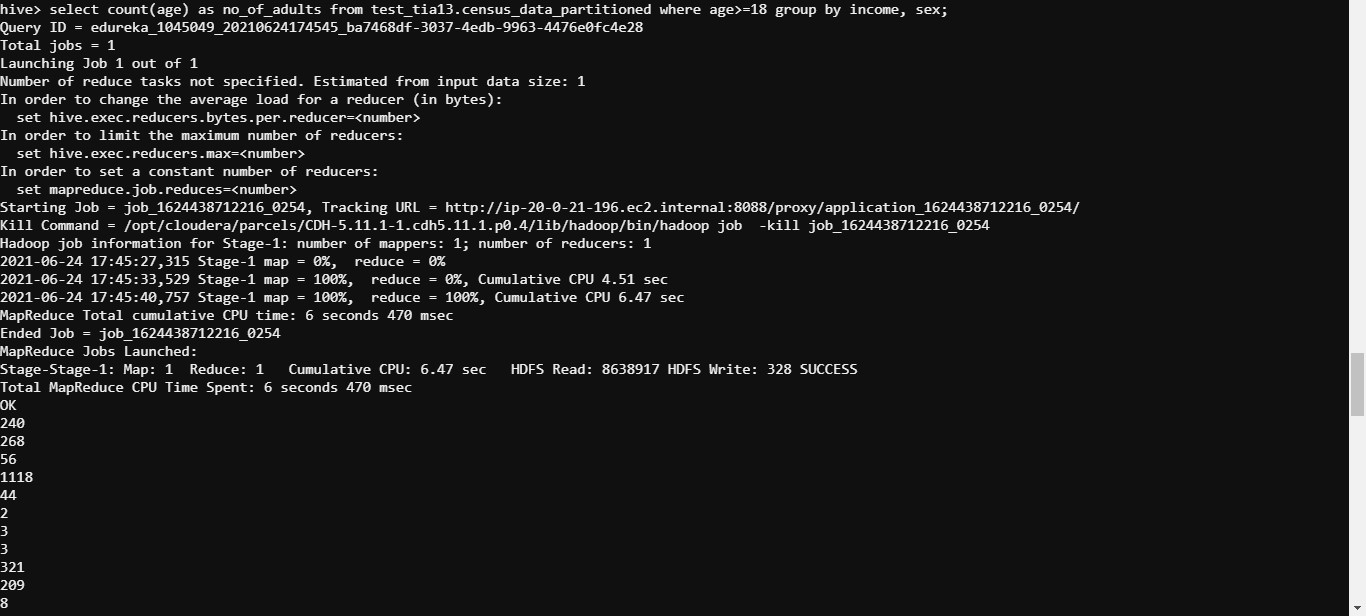
select count(age) as no\_of\_adults from test\_tia13.census\_data\_ext\_partitioned where age>=18 group by income, workclass;





1. Access Hive Internal Table with Partition
   1. Query the table to get the number of adults based on income and gender

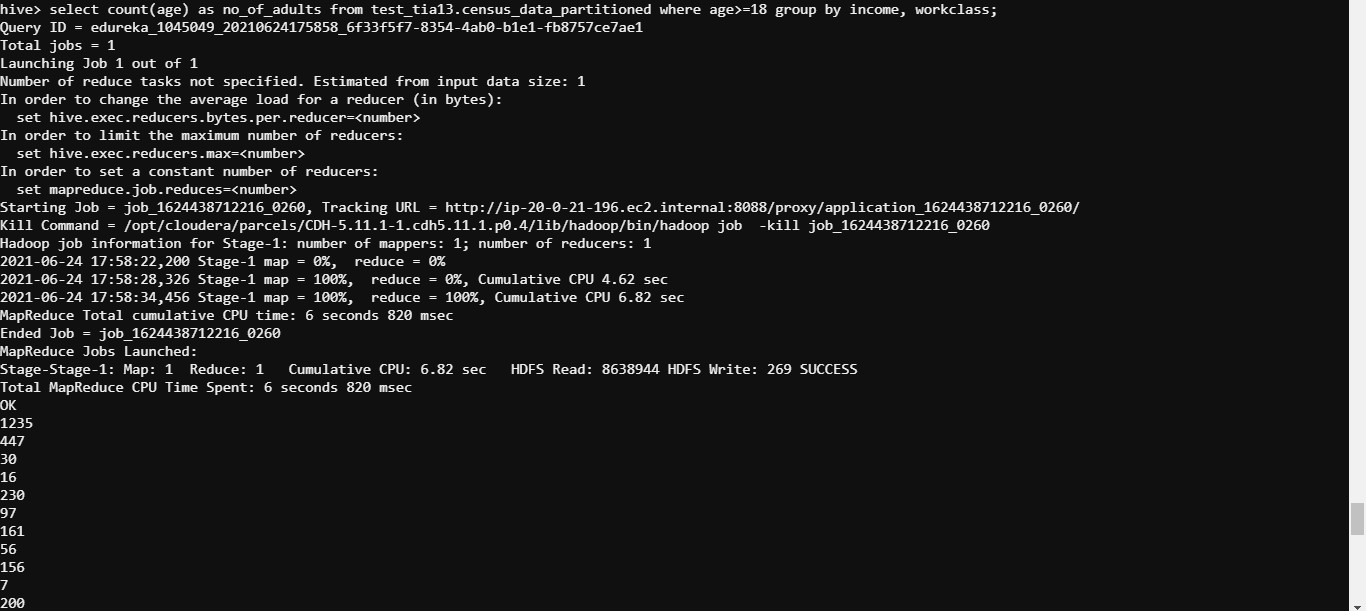
select count(age) as no\_of\_adults from test\_tia13.census\_data\_partitioned where age>=18 group by income, sex;





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

select count(age) as no\_of\_adults from test\_tia13.census\_data\_partitioned where age>=18 group by income, workclass;





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

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).

Time taken while working with External Partition Tables (gender, workclass) - 17.722secs, 21.24secs;

Avg time taken while working on External Partition Tables - 19.481secs;

Time taken while working with Internal Partition Tables (gender, workclass) - 19.501secs, 20.613secs;

Avg time taken while working on Internal Partition Tables - 20.057secs;

Time taken while working with spark sql - less than 10secs;

Working with spark sql is much time saving compared to hive.