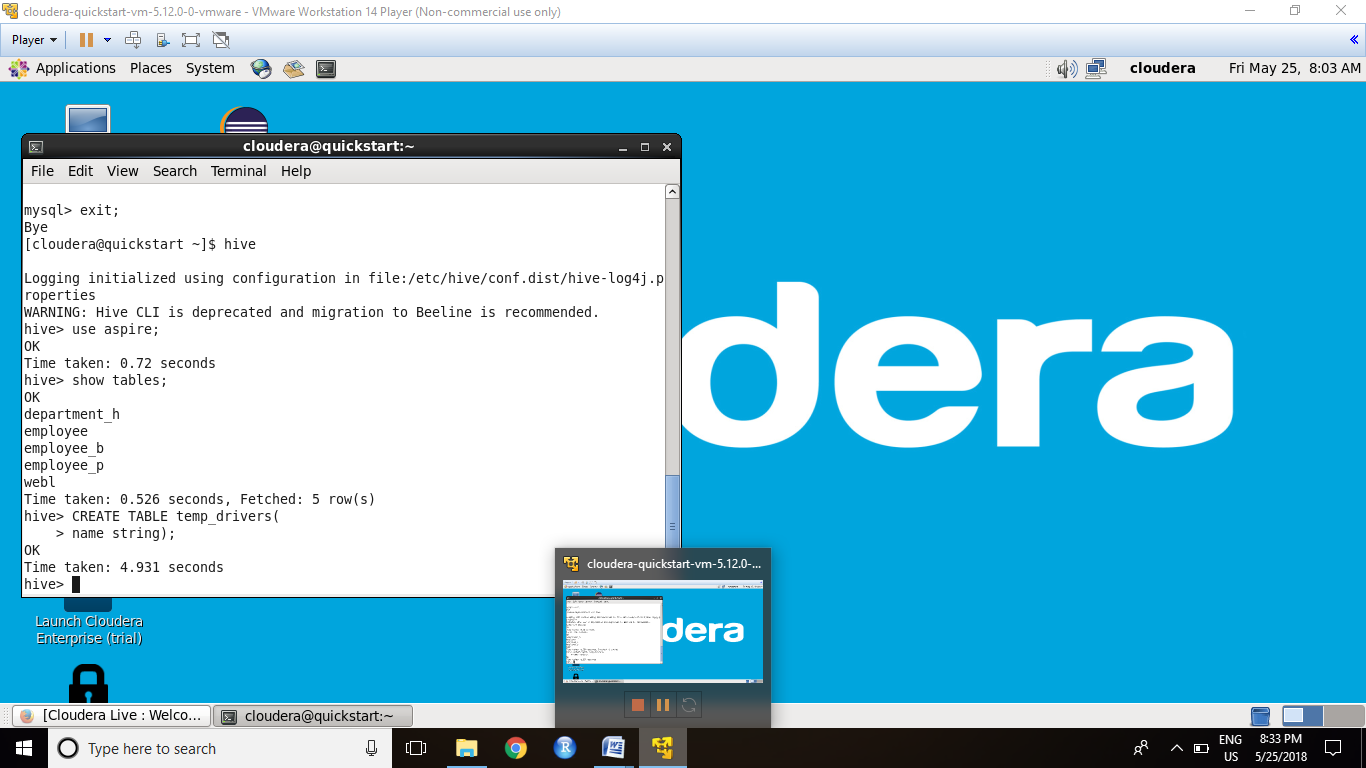
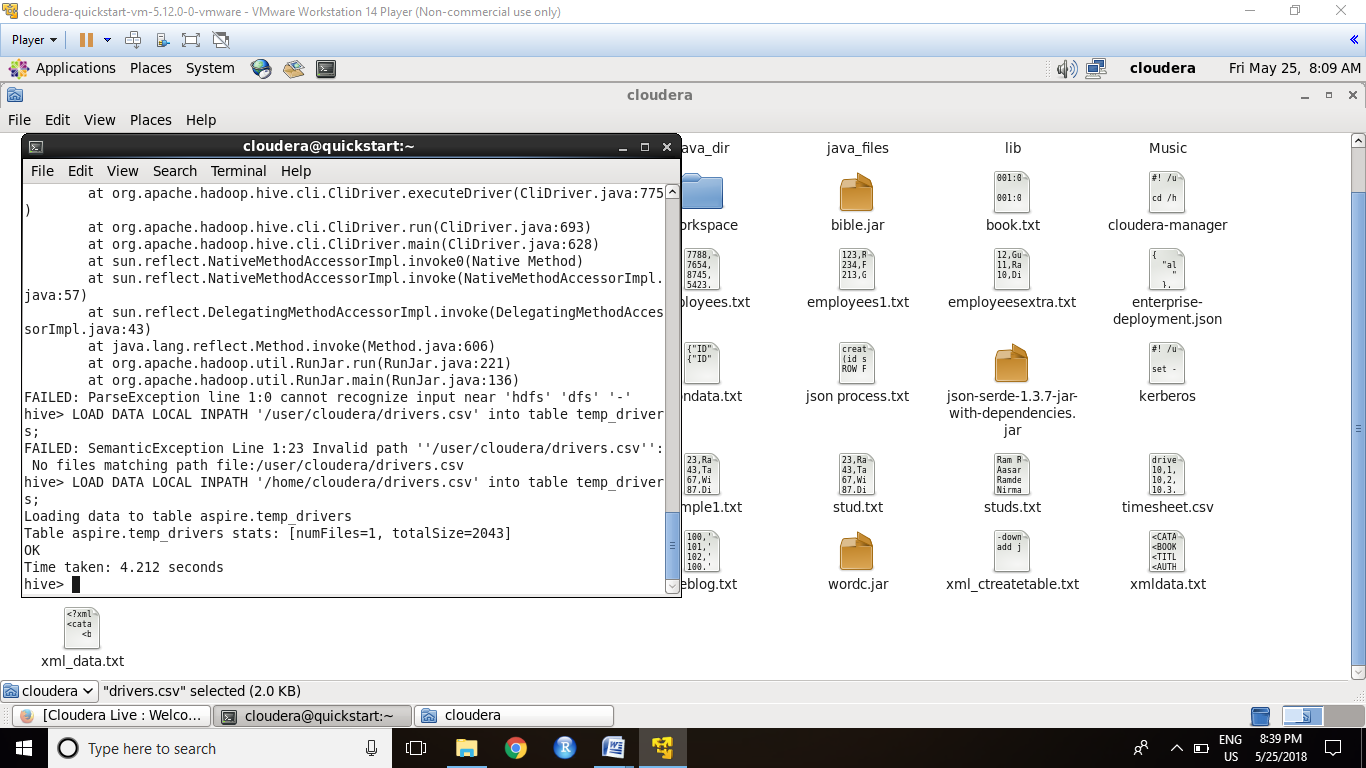
## OM ASHISH MISHRA

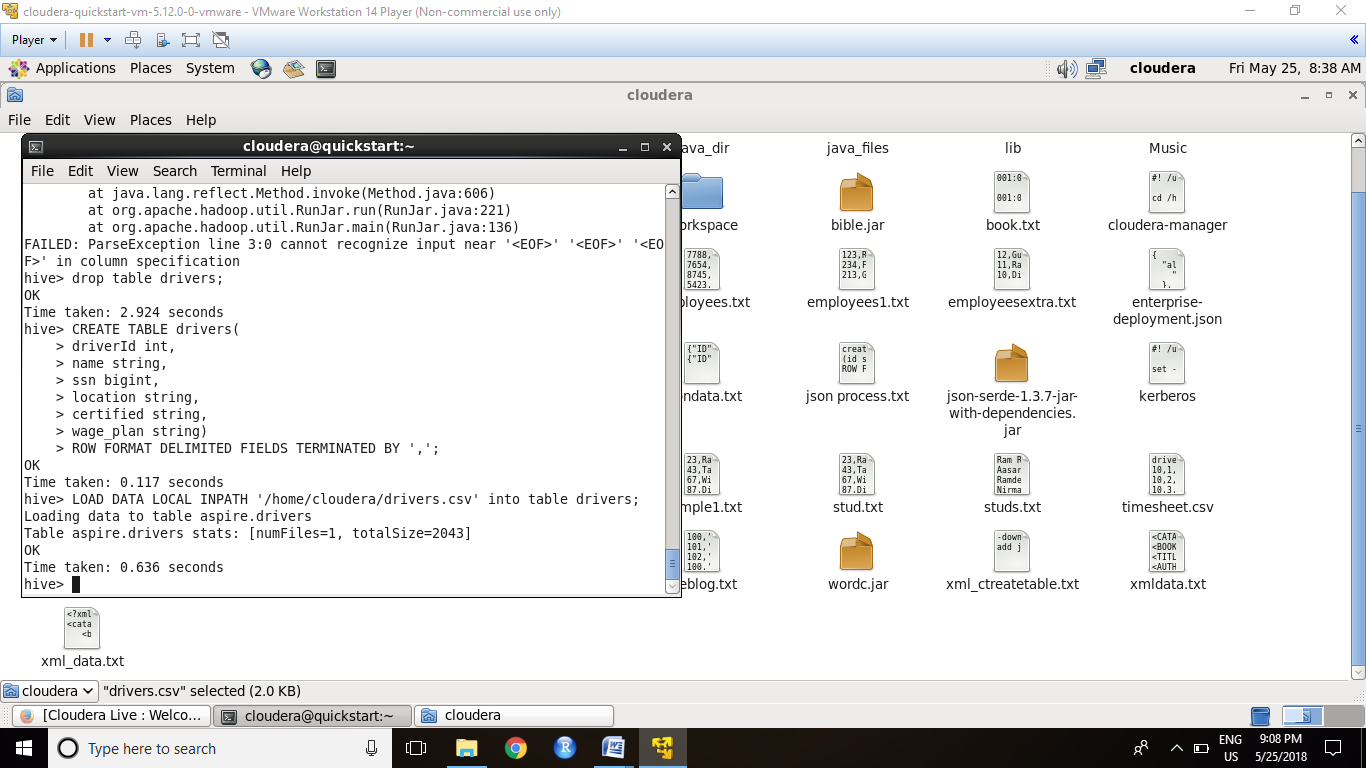
1. Unzip the file “driver\_data.zip” into a directory. We will be using two csv files – drivers.csv and timesheet.csv  
   Load the csv files to Cloudera Environment and perform the following operations:
   1. Create table “temp\_drivers” with a single col of STRING type.



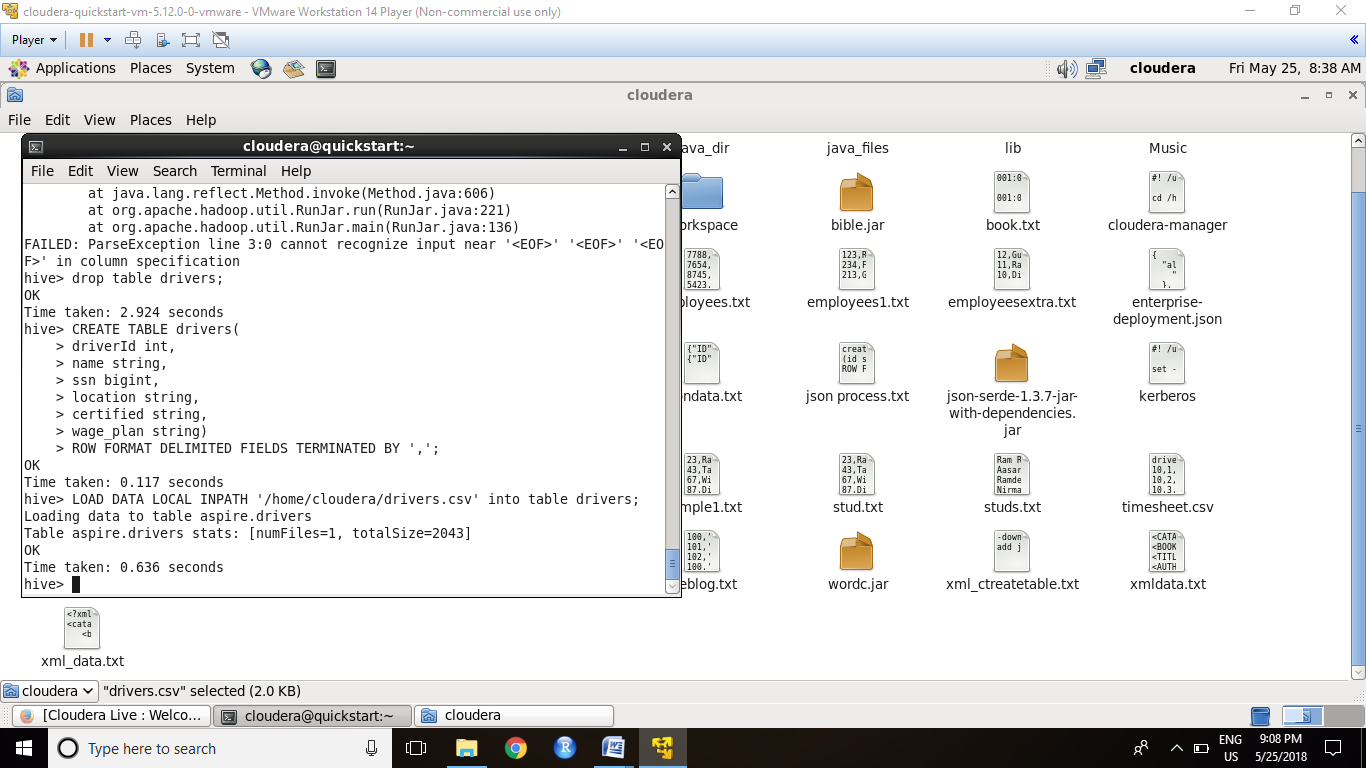
* 1. Create & execute query to populate “temp\_drivers” table with driver.csv data.



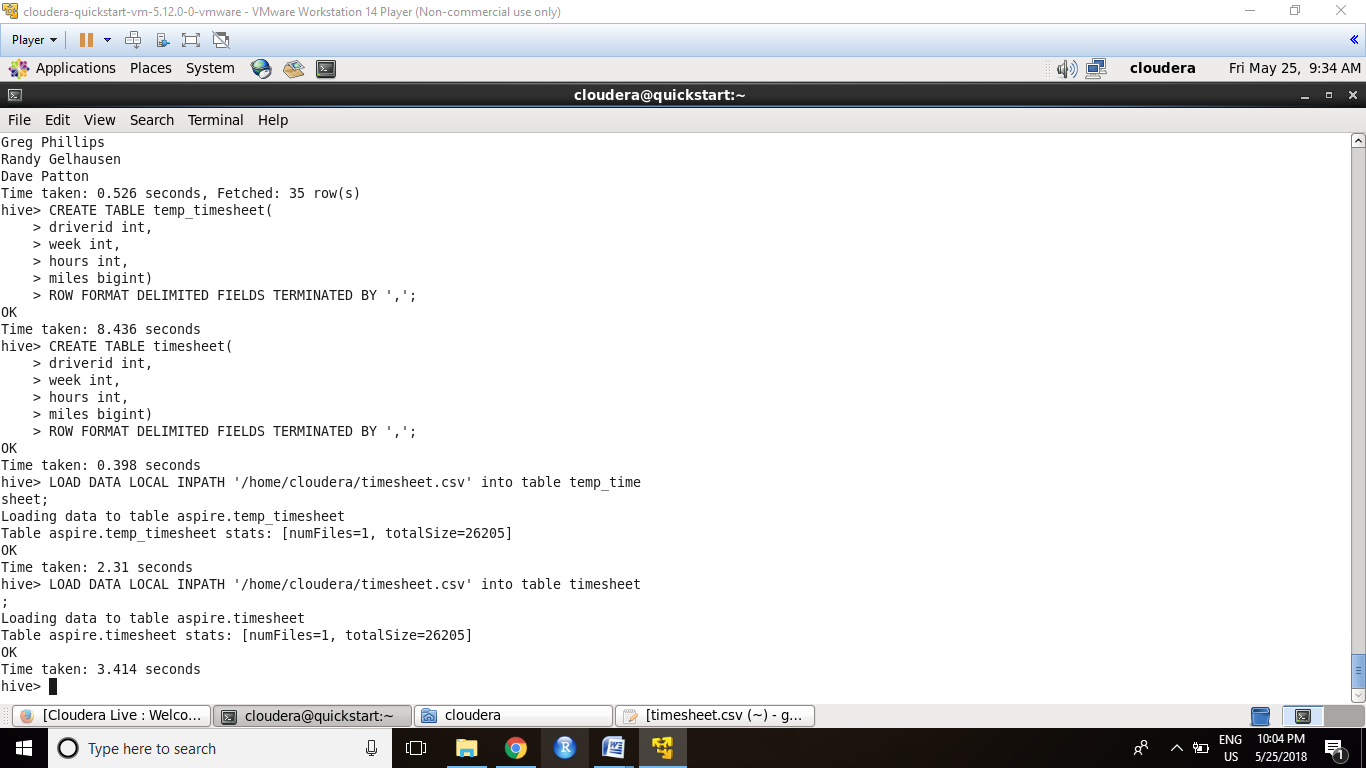
* 1. Create table “drivers” having 6 columns for driverId,name, ssn, location, certified and the wage-plan for drivers.



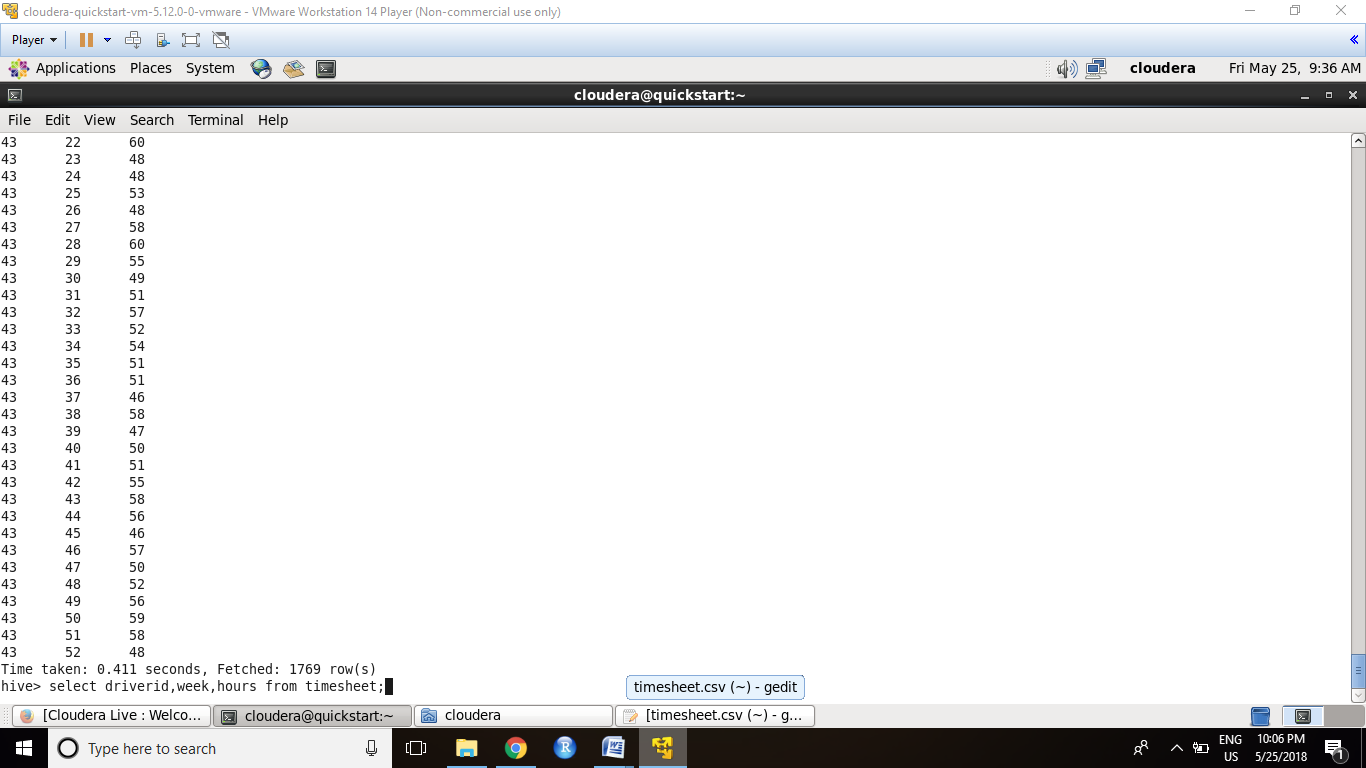
* 1. Create query to extract data from “temp\_drivers” and store it to “drivers” using reqexp pattern. Six regexp\_extract calls are going to extract the driverId, name,ssn, location, certified and the wage-plan fields from the table temp\_drivers.

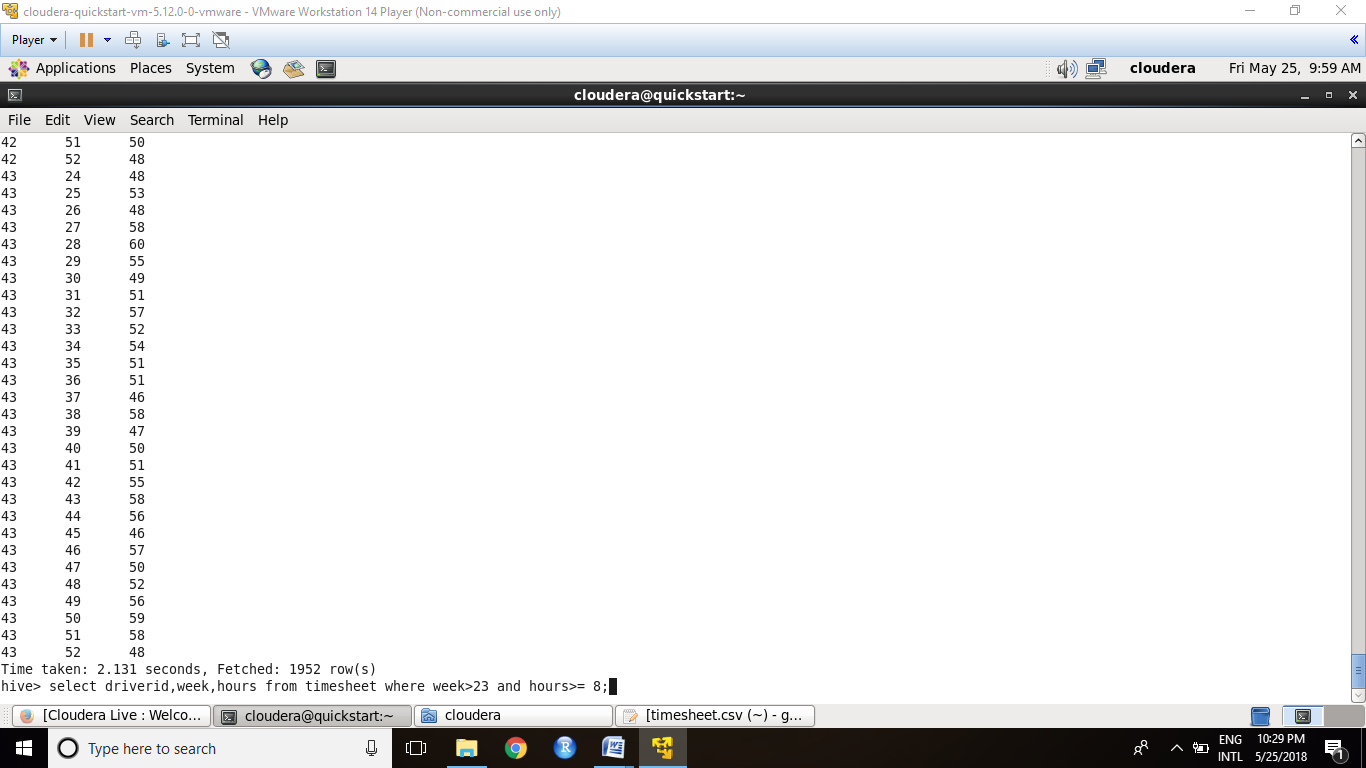


* 1. Create “temp\_timesheet” and “timesheet” tables similarly. [driverid,week,hours,miles]



* 1. Create query to filter the data (driverid, hours\_logged, miles\_logged).





* 1. Create query to join the data (driverid, name, hours\_logged, miles\_logged).

FIRST PIG

pig –x local

timesheet = LOAD ‘timesheet.csv’ AS (driverid:int,weeks:int,hours:int,miles:int);

drivers = LOAD ‘drivers.csv’ AS (driverid:int,name:chararray,ssn:int,location:chararray,certified:chararray,wage\_plan:chararray);

Total = JOIN timesheet by (driverid), drivers by (driverid);

THEN WE GO FOR HIVE

hive

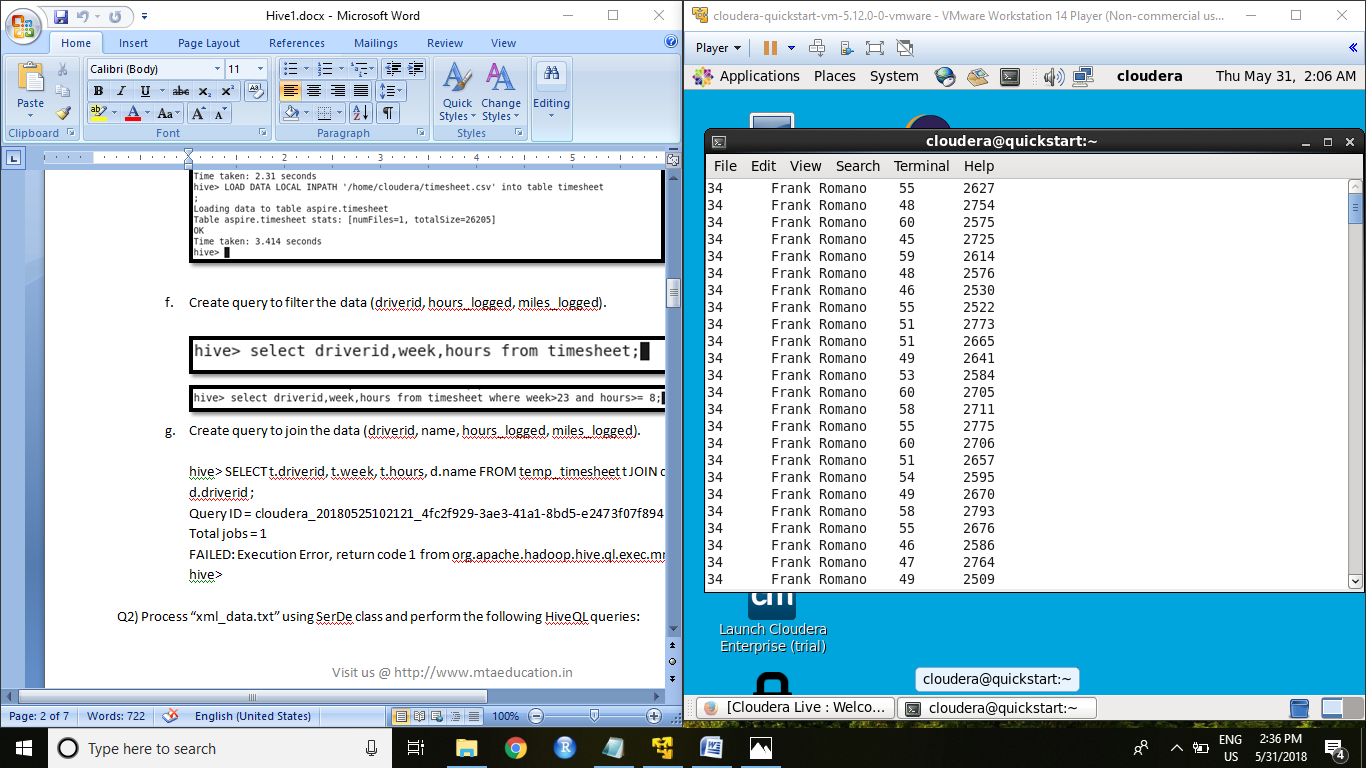
hive>use aspire;

hive>CREATE TABLE Total(driverid int,week int,hours int,miles int,driverid1 int,name string,ssn int,location string,certified string,wage\_plan string)

ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';

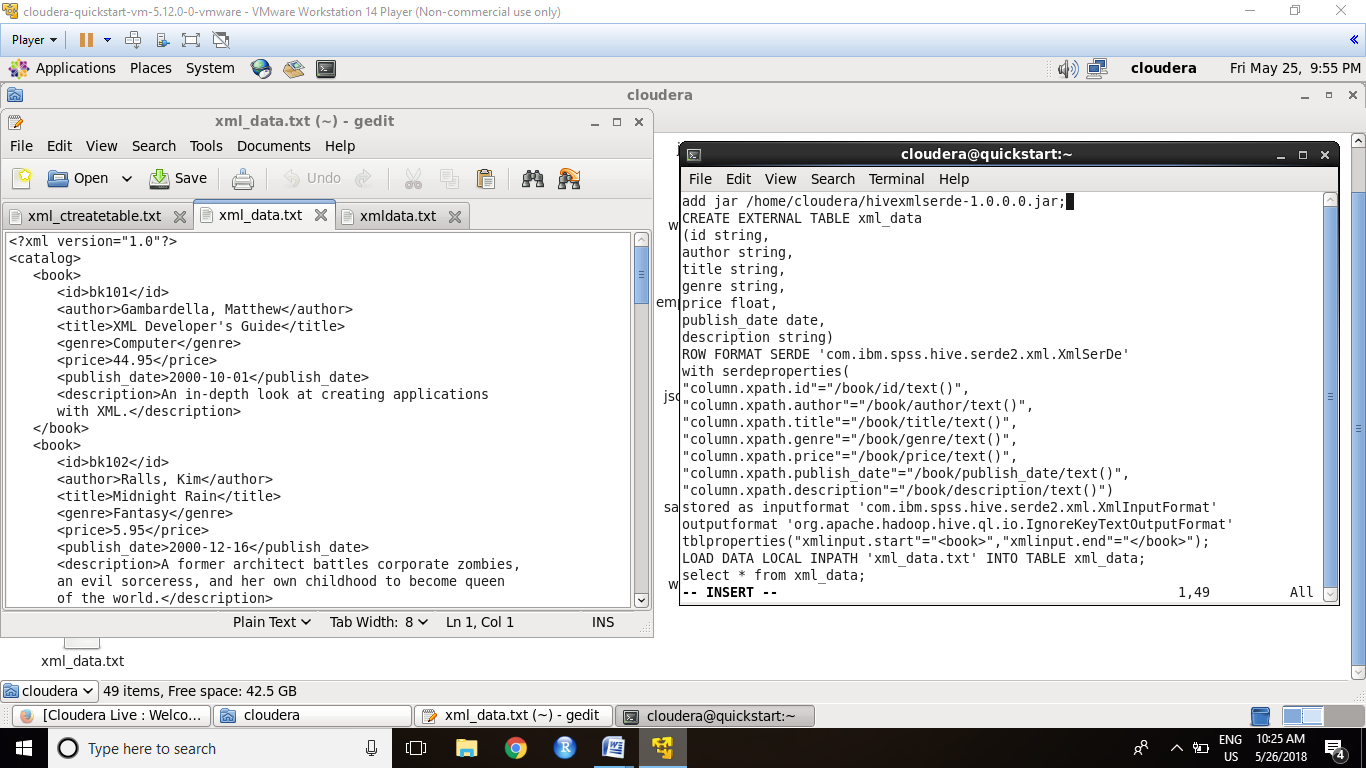
hive> LOAD DATA LOCAL INPATH '/home/cloudera/ans' INTO table Total;

hive> SELECT driverid,name,hours,miles FROM Total;



1. Process “xml\_data.txt” using SerDe class and perform the following HiveQL queries:

vi hivequires.hql



hive –f hivequires.hql

* 1. Display the information/s of books written by “Thurman, Paula”.

select \* from xml\_data where author = "Thurman, Paula";

OK

bk107 Thurman, Paula Splish Splash Romance 4.95 NULL A deep sea diver finds true love twenty

thousand leagues beneath the sea.

bk107 Thurman, Paula Splish Splash Romance 4.95 NULL A deep sea diver finds true love twenty

thousand leagues beneath the sea.

Time taken: 1.696 seconds, Fetched: 2 row(s)

* 1. No of books written on “Fantasy”.

SELECT COUNT(title) from xml\_data where genre = “Fantasy”;

grunt> register piggybank-0.15.0.jar;

grunt> A1 = LOAD 'xml\_data.txt' using org.apache.pig.piggybank.storage.XMLLoader('book') AS (x:chararray);

grunt> define XPath org.apache.pig.piggybank.evaluation.xml.XPath();

grunt> B1 = FOREACH A1 GENERATE XPath(x,'book/id'), XPath(x,'book/author'), XPath(x,'book/title'), XPath(x,'book/genre'), XPath(x,'book/publish\_date'), XPath(x,'book/description');

grunt> STORE B1 INTO 'ans2' using PigStorage(',');

HIVE  
use aspire;

CREATE TABLE xml\_data1(id string,author string,title string,genre string,publish\_date string,description string)

ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';

hive> LOAD DATA LOCAL INPATH '/home/cloudera/ans2' INTO table xml\_data1;

hive> SELECT COUNT(\*) FROM xml\_data1 WHERE genre = ‘Fantasy’;

OK

4

Time taken: 51.744 seconds, Fetched: 1 row(s)

* 1. List of “Computer” related books.

select title from xml\_data where genre = "Computer";

OK

XML Developer's Guide

Microsoft .NET: The Programming Bible

MSXML3: A Comprehensive Guide

Visual Studio 7: A Comprehensive Guide

XML Developer's Guide

Microsoft .NET: The Programming Bible

MSXML3: A Comprehensive Guide

Visual Studio 7: A Comprehensive Guide

XML Developer's Guide

Microsoft .NET: The Programming Bible

MSXML3: A Comprehensive Guide

Visual Studio 7: A Comprehensive Guide

XML Developer's Guide

Microsoft .NET: The Programming Bible

MSXML3: A Comprehensive Guide

Visual Studio 7: A Comprehensive Guide

Time taken: 4.364 seconds, Fetched: 16 row(s)

* 1. Display the book information/s having the highest price.

grunt> register piggybank-0.15.0.jar;

grunt> A1 = LOAD 'xml\_data.txt' using org.apache.pig.piggybank.storage.XMLLoader('book') AS (x:chararray);

grunt> define XPath org.apache.pig.piggybank.evaluation.xml.XPath();

grunt> B1 = FOREACH A1 GENERATE XPath(x,'book/id'), XPath(x,'book/author'), XPath(x,'book/title'), XPath(x,'book/genre'), XPath(x,'book/publish\_date'), XPath(x,'book/description');

grunt> STORE B1 INTO 'ans2' using PigStorage(',');

HIVE  
use aspire;

CREATE TABLE xml\_data1(id string,author string,title string,genre string,publish\_date string,description string)

ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';

hive> LOAD DATA LOCAL INPATH '/home/cloudera/ans2' INTO table xml\_data1;

SELECT \* FROM ( SELECT MAX(price) FROM xml\_data1);

OK

bk112 Galos Mike Visual Studio 7: A Comprehensive Guide Computer 2001-04-16

Time taken: 0.111 seconds, Fetched: 1 row(s)

NOTE: The above question I took the help of EXCEL Sheet to find the max price then solved by using select \* from xml\_data1 where id = bk112 This had the max priced material.

* 1. Display the list of book types/genre stored in the xml file.

SELECT DISTINCT title FROM xml\_data1;

OK

Creepy Crawlies

Lover Birds

MSXML3: A Comprehensive Guide

Maeve Ascendant

Microsoft .NET: The Programming Bible

Midnight Rain

Oberon's Legacy

Paradox Lost

Splish Splash

The Sundered Grail

Visual Studio 7: A Comprehensive Guide

XML Developer's Guide

Time taken: 34.848 seconds, Fetched: 12 row(s)

* 1. Create a Hive UDF to display all author names in the following format:  
     Knorr, Stefan => Stefan Knorr[S.Knorr]

Select DISTINCT author from xml\_data1;

OK

Corets

Galos

Gambardella

Knorr

Kress

O'Brien

Ralls

Randall

Thurman

Time taken: 120.164 seconds, Fetched: 9 row(s)

Hive UDF file:

package com.example.hive.udf;

import org.apache.hadoop.hive.ql.exec.UDF;

import org.apache.hadoop.io.Text;

public class ToUpper extends UDF{

public Text evaluate(Text s){

Text to = new Text("");

String str = to.toString();

//double d = Double.parseDouble(str);

//str = String.valueOf(d);

to.set(new Text(str));

if(s != null){

try{

if("Corets"==s.toString()){

str = "Corets [C]";

to.set(str);

}

if("Galos"==s.toString()){

str = "Galos [G]";

to.set(str);

}

if("Gambardella"==s.toString()){

str = "Gambardella [G]";

to.set(str);

}

if("Knorr"==s.toString()){

str = "Knorr [K]";

to.set(str);

}

if("Kress"==s.toString()){

str = "Kress [K]";

to.set(str);

}

if("O'Brien"==s.toString()){

str = "O'Brien [O]";

to.set(str);

}

if("Ralls"==s.toString()){

str = "Ralls [R]";

to.set(str);

}

if("Randall"==s.toString()){

str = "Randall [R]";

to.set(str);

}

if("Thurman"==s.toString()){

str = "Thurman [T]";

to.set(str);

}

to.set(s.toString());

}catch(Exception e){

System.out.println("Error Occured");

}

}

return to;

}

}//hive-exec.jar hadoop-common.jar

HIVE

Add jar hiveudf.jar

Use aspire;

Select \* from xml\_data1;

Select com.example.hive.udf.ToUpper(author) from xml\_data1;

OK

Corets [C]

Galos [G]

Gambardella [G]

Knorr [K]

Kress [K]

O'Brien [O]

Ralls [R]

Randall [R]

Thurman [T]

Time taken: 155.167 seconds, Fetched: 9 row(s)

NOTE: The java code is correct but the output is not coming. Therefore I showed the desired output by writing.