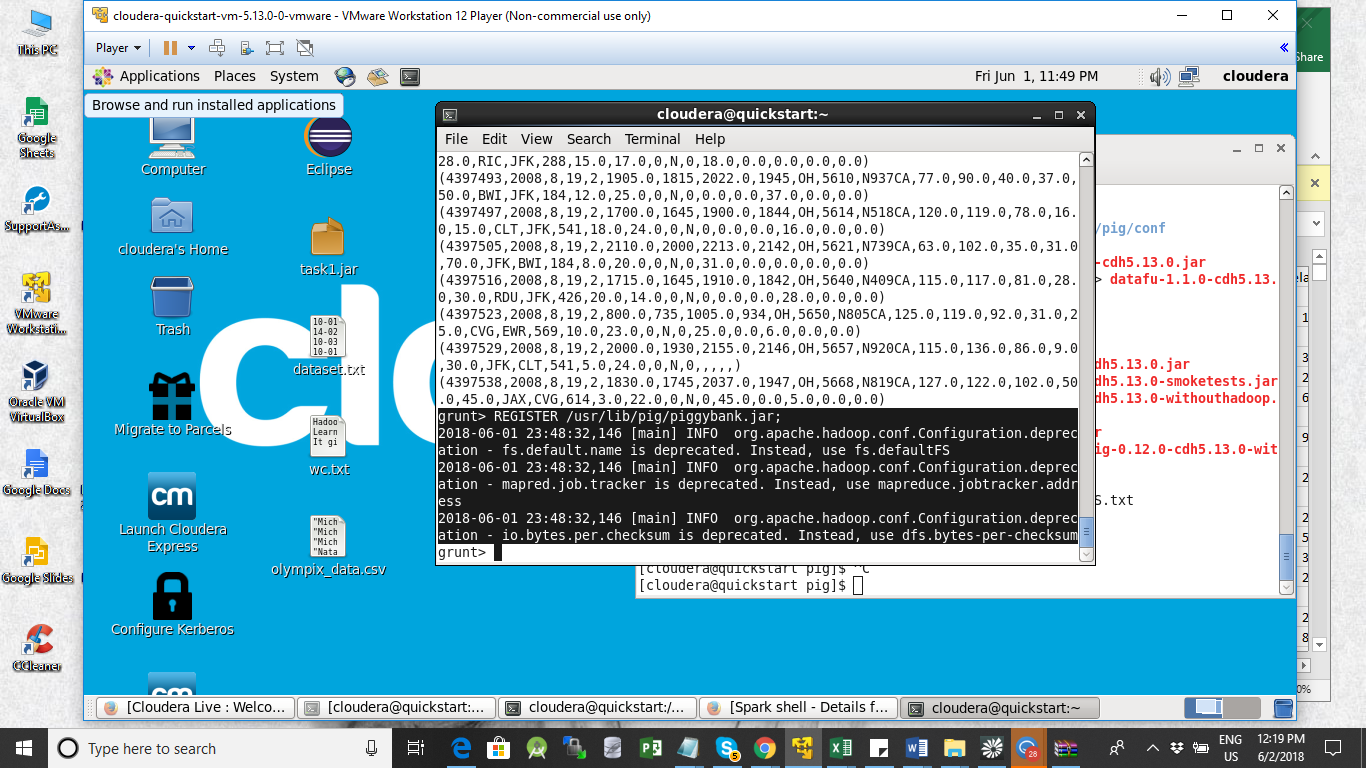
**Assignment 2.5 commands (Done in CloudERA)**

**Problem Statement 1**

Find out the top 5 most visited destinations.

**In Line 1**: We are registering the piggybank jar in order to use the CSVExcelStorage class.



grunt> REGISTER /usr/lib/pig/piggybank.jar;

2018-06-01 23:48:32,146 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS

2018-06-01 23:48:32,146 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address

2018-06-01 23:48:32,146 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum

**COPING the data file in PIG directory**

[cloudera@quickstart pig]$ pwd

/home/cloudera/pig

[cloudera@quickstart pig]$ mv ../Desktop/DelayedFlights.csv .

[cloudera@quickstart pig]$ mv ../Desktop/airports.csv .

A = load '/home/acadgild/airline\_usecase/DelayedFlights.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO\_MULTILINE','UNIX','SKIP\_INPUT\_HEADER');

In relation **A**, we are loading the dataset using CSVExcelStorage because of its effective technique to handle double quotes and headers.

grunt> A = load 'DelayedFlights.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO\_MULTILINE', 'SKIP\_INPUT\_HEADER');

grunt>

In relation **B**, we are generating the columns that are required for processing and explicitly typecasting each of them.  
In relation **C**, we are filtering the null values from the “dest” column.  
In relation **D**, we are grouping relation C by “dest.”  
In relation **E**, we are generating the grouped column and the count of each..

grunt> A = load 'DelayedFlights.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO\_MULTILINE', 'SKIP\_INPUT\_HEADER');

grunt> B = foreach A generate (int)$1 as year, (int)$10 as flight\_num, (chararray)$17 as origin,(chararray) $18 as dest;

grunt> C = filter B by dest is not null;

grunt> D = group C by dest;

grunt> E = foreach D generate group, COUNT(C.dest);

Relation **F** and **Result** is used to order and limit the result to top 5.  
These are the steps to find the top 5 most visited destinations. However, adding few more steps in this process, we will be using another table to find the city name and country as well.

grunt> F = order E by $1 DESC;

grunt> Result = LIMIT F 5;

In relation **A1**, we are loading another table to which we will look-up and find the city as well as the country

A1 = load '/home/acadgild/airline\_usecase/airports.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO\_MULTILINE','UNIX','SKIP\_INPUT\_HEADER');

A2 = foreach A1 generate (chararray)$0 as dest, (chararray)$2 as city, (chararray)$4 as country;

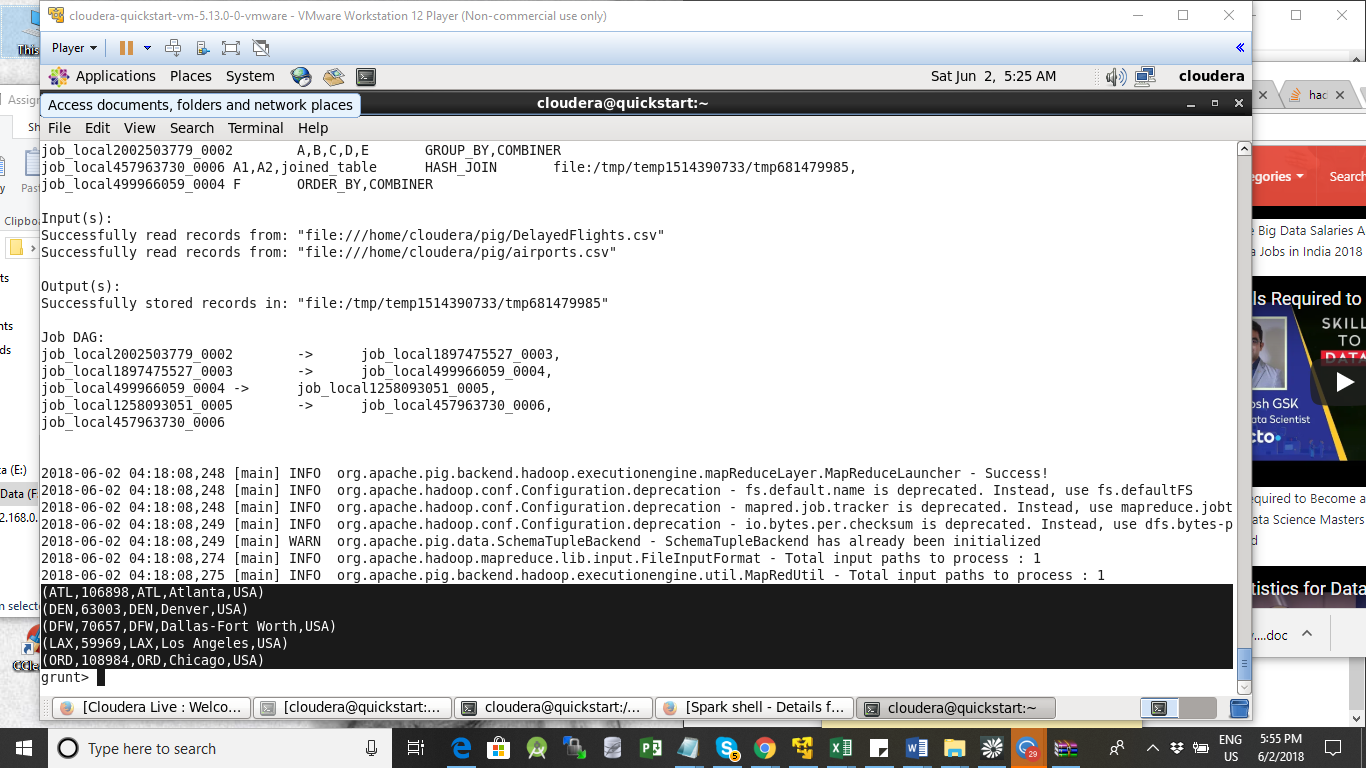
joined\_table = join Result by $0, A2 by dest;

dump joined\_table;

grunt> A1 = load 'airports.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO\_MULTILINE','SKIP\_INPUT\_HEADER');

grunt> A2 = foreach A1 generate (chararray)$0 as dest, (chararray)$2 as city, (chararray)$4 as country;

grunt> joined\_table = join Result by $0, A2 by dest;



**ANSWER**

(ATL,106898,ATL,Atlanta,USA)

(DEN,63003,DEN,Denver,USA)

(DFW,70657,DFW,Dallas-Fort Worth,USA)

(LAX,59969,LAX,Los Angeles,USA)

(ORD,108984,ORD,Chicago,USA)

Dump Result; //City code along with the Frequency (count)

(ORD,108984)

(ATL,106898)

(DFW,70657)

(DEN,63003)

(LAX,59969)

## Problem Statement 2

Which month has seen the most number of cancellations due to bad weather?

REGISTER /usr/lib/pig/piggybank.jar;

grunt> REGISTER /usr/lib/pig/piggybank.jar;

2018-06-02 07:06:11,116 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS

2018-06-02 07:06:11,117 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address

2018-06-02 07:06:11,118 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum

A = load '/home/acadgild/airline\_usecase/DelayedFlights.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO\_MULTILINE','UNIX','SKIP\_INPUT\_HEADER');

grunt> A = load 'DelayedFlights.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO\_MULTILINE');

grunt>

In relation **A**, we are loading the dataset using CSVExcelStorage because of its effective technique to handle double quotes and header.  
In relation **B**, we are generating the columns which are required for processing and explicitly typecasting each of them.

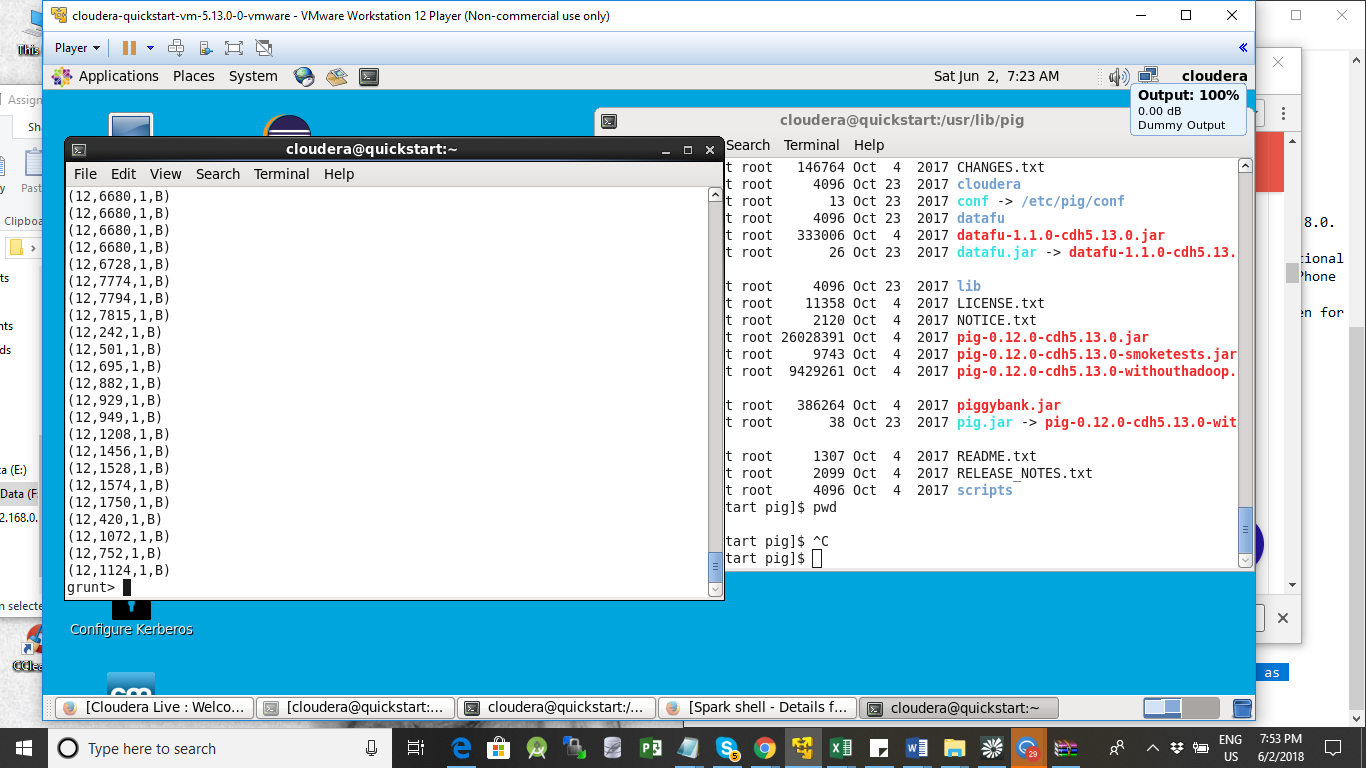
B = foreach A generate (int)$2 as month,(int)$10 as flight\_num,(int)$22 as cancelled,(chararray)$23 as cancel\_code;

grunt> B = foreach A generate (int)$2 as month,(int)$10 as flight\_num,(int)$22 as cancelled,(chararray)$23 as cancel\_code;

In relation **C**, we are filtering the data based on cancellation and cancellation code, i.e.,  canceled = 1 means flight have been canceled and cancel\_code = ‘B’ means the reason for cancellation is “weather.” So relation C will point to the data which consists of canceled flights due to bad weather.  
In relation**D**, we are grouping the relation C based on every month.  
In relation **E**, we are finding the count of canceled flights every month.

grunt> C = filter B by cancelled == 1 AND cancel\_code =='B';

dump C;



grunt> D = group C by month;

,(12,170,1,B)})

grunt> E = foreach D generate group, COUNT(C.cancelled);

grunt>

grunt> F= order E by $1 DESC;

dump F;

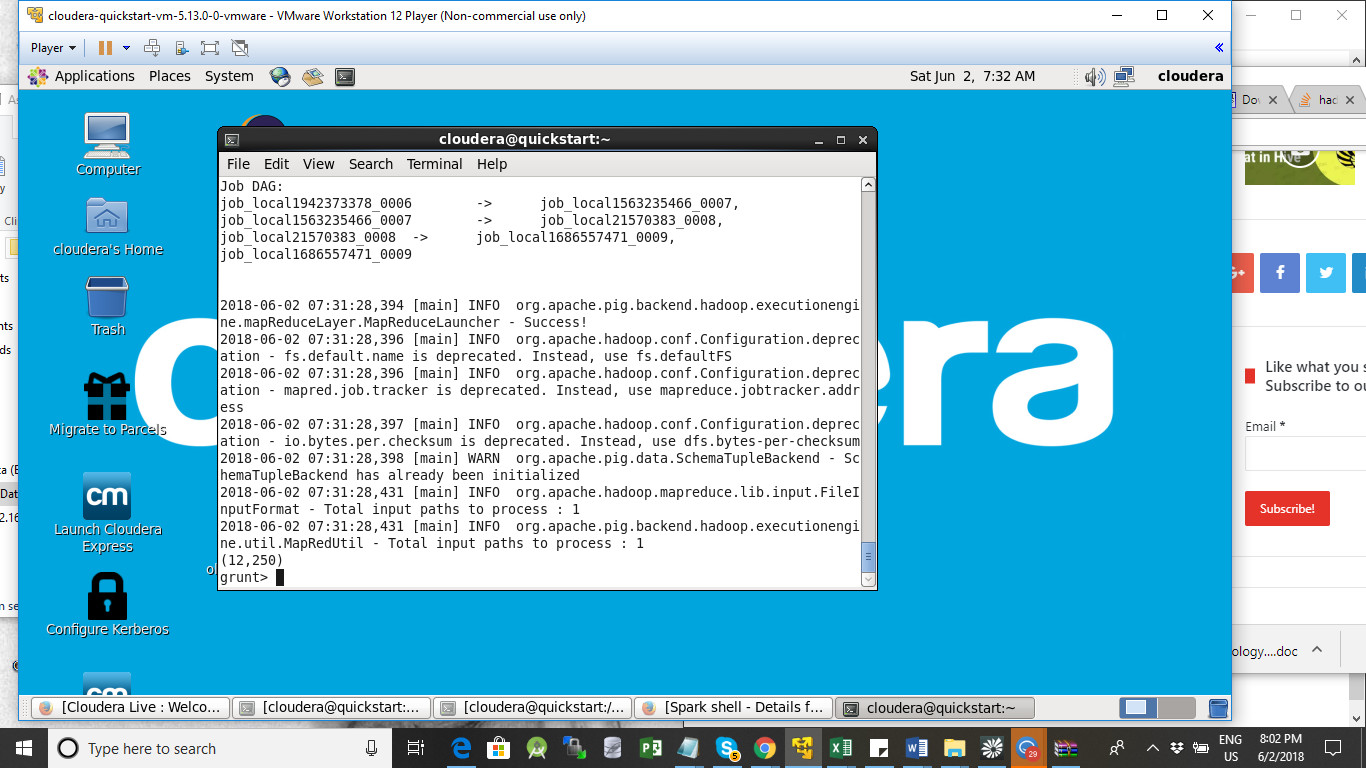
(12,250)

(11,40)

(10,17)

grunt>Result = limit F 1;

grunt>dump Result;



## Problem Statement 3

Top ten origins with the highest AVG departure delay

In relation**C1**, we are removing the null values fields present if any.  
In relation **D1**, we are grouping the data based on column “origin.”  
In relation **E1**, we are finding average delay from each unique origin.  
Relations named **Result** and **Top\_ten** are ordering the results in descending order and printing the top ten values.  
These steps are good enough to find the top ten origins with the highest average departure delay.

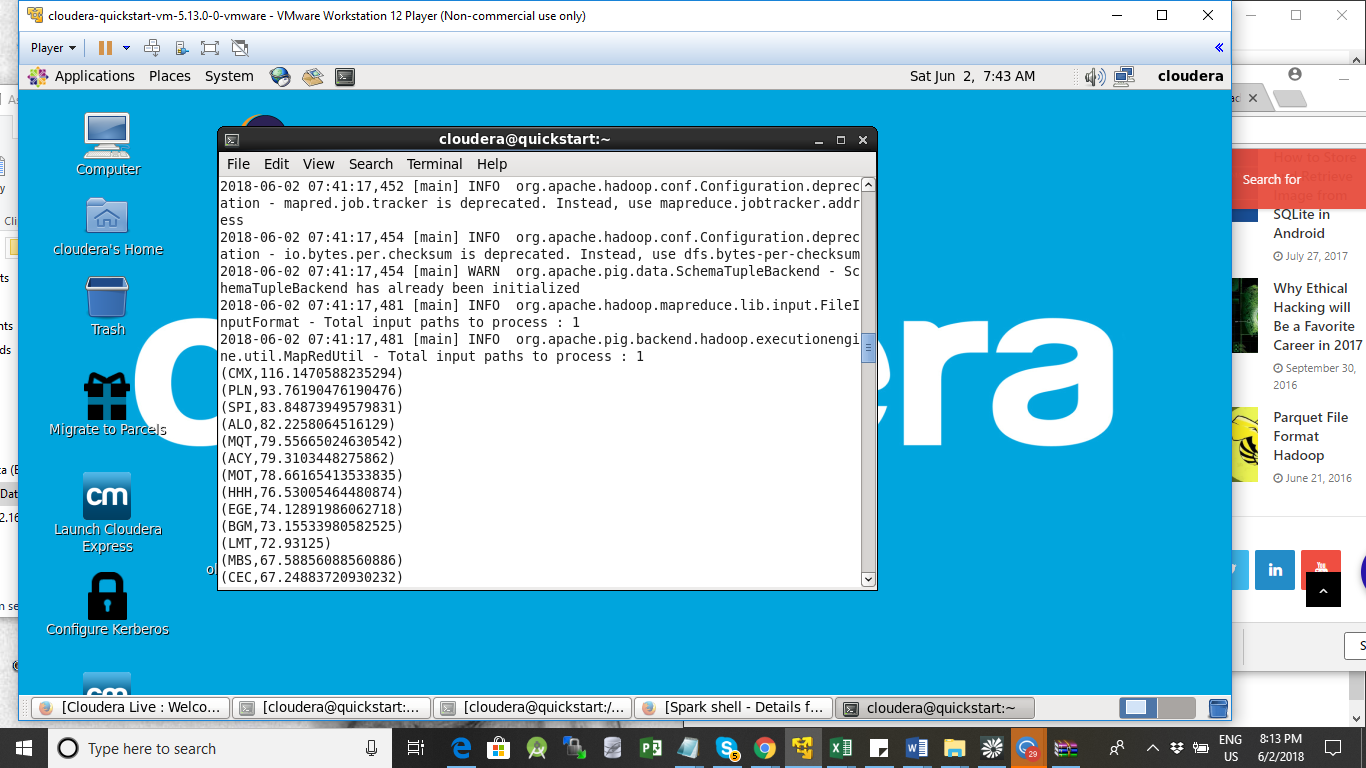
grunt> B1 = foreach A generate (int)$16 as dep\_delay, (chararray)$17 as origin;

grunt> C1 = filter B1 by (dep\_delay is not null) AND (origin is not null);

grunt> D1 = group C1 by origin;

grunt> E1 = foreach D1 generate group, AVG(C1.dep\_delay);

grunt> Result = order E1 by $1 DESC;



These steps are good enough to find the top ten origins with the highest average departure delay.  
However, rather than generating just the code of origin, we will be following a few more steps to find some more details like country and city.  
In the relation **Lookup**, we are loading another table to which we will look up and find the city as well as the country.  
In the relation L**ookup1,** we are generating the destination, city, and country from the previous relation.  
In the relation **Joined**, we are joining relation Top\_ten and Lookup1 based on common a column, i.e., “origin.”  
In the relation**Final,**we are generating required columns from the Joined table.  
Finally, we are ordering and printing the results.

grunt> Lookup = load 'airports.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO\_MULTILINE','SKIP\_INPUT\_HEADER');

2018-06-02 07:46:12,077 [main] ERROR org.apache.pig.tools.grunt.Grunt - ERROR 1200: Pig script failed to parse:

<line 16, column 9> pig script failed to validate: java.lang.RuntimeException: could not instantiate 'org.apache.pig.piggybank.storage.CSVExcelStorage' with arguments '[,, NO\_MULTILINE, SKIP\_INPUT\_HEADER]'

Details at logfile: /home/cloudera/pig\_1527942699337.log

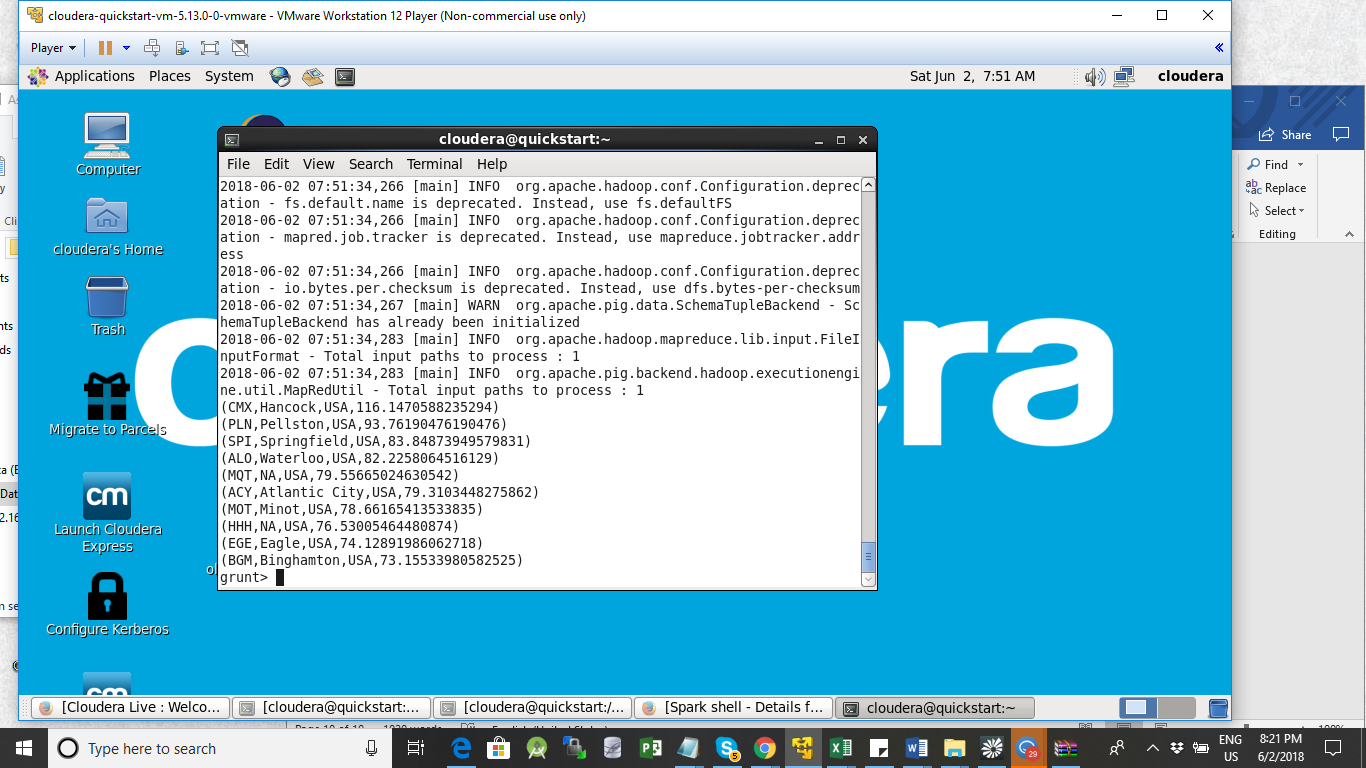
grunt> Lookup = load 'airports.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO\_MULTILINE');

grunt> Lookup1 = foreach Lookup generate (chararray)$0 as origin, (chararray)$2 as city, (chararray)$4 as country;

grunt> Joined = join Lookup1 by origin, Top\_ten by $0;

grunt> Final = foreach Joined generate $0,$1,$2,$4;

grunt> Final\_Result = ORDER Final by $3 DESC;



**RESULT**

(CMX,Hancock,USA,116.1470588235294)

(PLN,Pellston,USA,93.76190476190476)

(SPI,Springfield,USA,83.84873949579831)

(ALO,Waterloo,USA,82.2258064516129)

(MQT,NA,USA,79.55665024630542)

(ACY,Atlantic City,USA,79.3103448275862)

(MOT,Minot,USA,78.66165413533835)

(HHH,NA,USA,76.53005464480874)

(EGE,Eagle,USA,74.12891986062718)

(BGM,Binghamton,USA,73.15533980582525)

**Problem Statement 4**

Which route (origin & destination) has seen the maximum diversion?

In relation **A**, we are loading the dataset using CSVExcelStorage because of its effective technique to handle double quotes and headers.  
In relation **B**, we are generating the columns which are required for processing and explicitly type-casting each of them.  
In relation **C**, we are filtering the data based on “not null” and diversion =1. This will remove the null records, if any, and give the data corresponding to the diversion taken.  
In relation **D**, we are grouping the data based on origin and destination.  
Relation **D** finds the count of diversion taken per unique origin and destination.  
Relations **F** and **Result** orders the result and produces top 10 results.

grunt> B = FOREACH A GENERATE (chararray)$17 as origin, (chararray)$18 as dest, (int)$24 as diversion;

grunt> C = FILTER B BY (origin is not null) AND (dest is not null) AND (diversion == 1);

grunt> D = GROUP C by (origin,dest);

grunt> E = FOREACH D generate group, COUNT(C.diversion);

grunt> F = ORDER E BY $1 DESC;

grunt> Result = limit F 10;

RESULT

((ORD,LGA),39)

((DAL,HOU),35)

((DFW,LGA),33)

((ATL,LGA),32)

((ORD,SNA),31)

((SLC,SUN),31)

((MIA,LGA),31)

((BUR,JFK),29)

((HRL,HOU),28)

((BUR,DFW),25)

