**Assignment 5.2**

**Problem Statement:**

**Implement the use case present in below blog link and share the complete steps along with screenshot(s) from your end.**

**NOTE: You must submit a word file containing steps and screenshots.**

[**https://acadgild.com/blog/aviation-data-analysis-using-apache-pig/**](https://acadgild.com/blog/aviation-data-analysis-using-apache-pig/)

The U.S. Department of Transportation’s (DOT) Bureau of Transportation Statistics (BTS) tracks the on-time performance of domestic flights operated by large air carriers. Summary information on *the number of on-time, delayed, canceled, and diverted flights* appears in DOT’s monthly Air Travel Consumer Report, published about 30 days after the month’s end, as well as in summary tables posted on this website. Summary statistics and raw data are made available to the public at the time the Air Travel Consumer Report is released.

These are 2 different datasets, i.e., Delayed\_Flights.csv and Airports.csv.

**Delayed\_Flights.csv Datasets**

There are 29 columns in this dataset. Some of them have been mentioned below:

* Year: 1987 – 2008
* Month: 1 – 12
* FlightNum: Flight number
* Canceled: Was the flight canceled?
* CancelleationCode: The reason for cancellation. (N=No cancellation,A = carrier, B = weather, C = NAS, D = security)
* Origin:Origin
* Dest: Destination

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Year | Month | FlightNum | Canceled | CancelleationCode | Origin | Dest |  |
| 2008 | 1 | 335 | 0 | N | IAD | TPA |  |
| 2008 | 1 | 3231 | 0 | N | IAD | TPA |  |
| 2008 | 1 | 448 | 0 | N | IND | BWI |  |
|  |  |  |  |  |  |  |  |

**Airports.csv Datasets**

* iata: the international airport abbreviation code
* name of the airport
* city and country in which airport is located.
* lat and long: the latitude and longitude of the airport

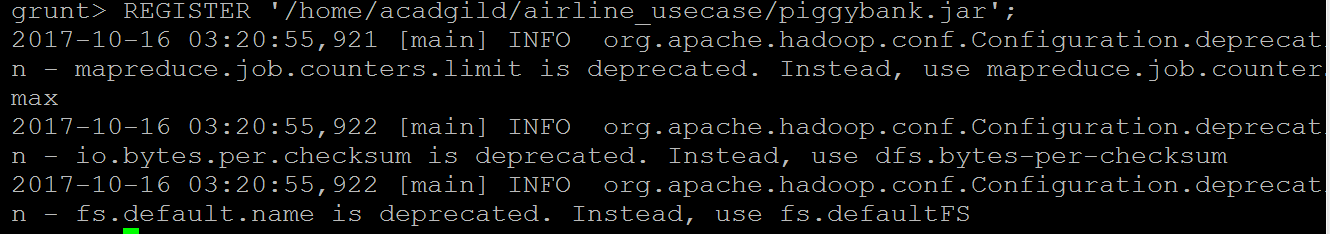
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| iata | Airport | city | state | country | lat | long |
| 00M | Thigpen | Bay Springs | MS | USA | 31.95376 | -89.2345 |
| 00R | Livingston Municipal | Livingston | TX | USA | 30.68586 | -95.0179 |
| 00V | Meadow Lake | Colorado Springs | CO | USA | 38.94575 | -104.57 |
|  |  |  |  |  |  |  |

## Problem Statement 1

Find out the top 5 most visited destinations.

1.Registering the piggybank jar in order to use the CSVExcelStorage class.

REGISTER '/home/acadgild/airline\_usecase/piggybank.jar';

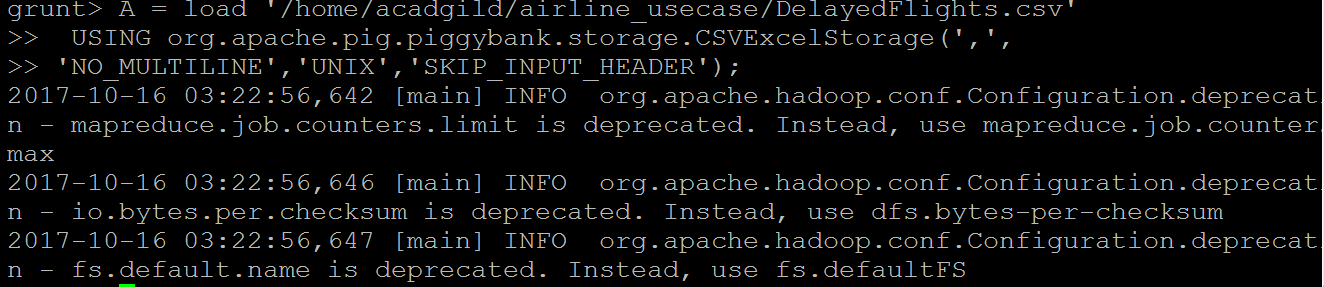


2. In relation **A**, we are loading the dataset using CSVExcelStorage because of its effective technique to handle double quotes and headers.

A = load '/home/acadgild/airline\_usecase/DelayedFlights.csv'

USING org.apache.pig.piggybank.storage.CSVExcelStorage(',',

'NO\_MULTILINE','UNIX','SKIP\_INPUT\_HEADER');

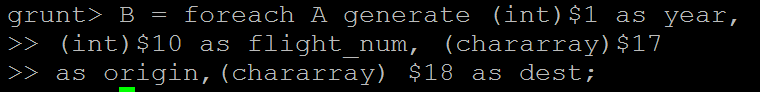


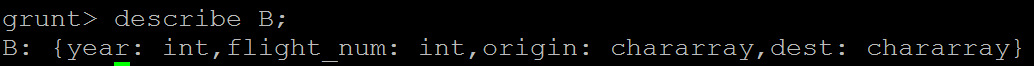
3. In relation **B**, we are generating the columns that are required for processing and explicitly typecasting each of them.

B = foreach A generate (int)$1 as year,

(int)$10 as flight\_num, (chararray)$17

as origin,(chararray) $18 as dest;





3. In relation C, we are filtering the null values from the “dest” column.

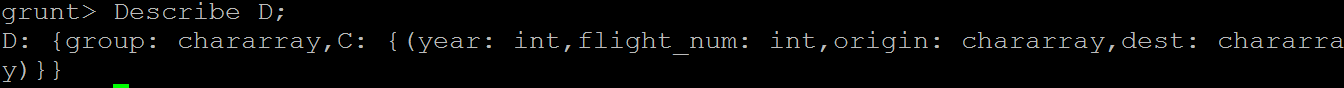
C = filter B by dest is not null; 



4. In relation D, we are grouping relation C by “dest.”

D = group C by dest;





5. In relation E, we are generating the grouped column and the count of each.

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





6. Relation **F** and **Result** is used to order and limit the result to top 5.

F = order E by $1 DESC;

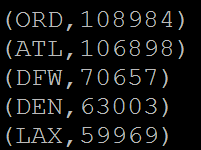




Result = LIMIT F 5;



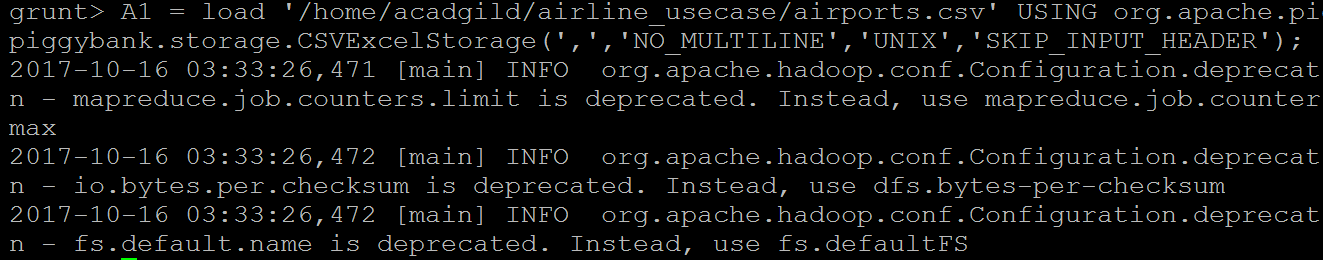




These are the steps to find the top 5 most visited destinations. However, adding few more steps in this process, we will be using another table to find the city name and country as well.

7. 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');



8. In relation A2, we are generating dest, city, and country from the previous relation.

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





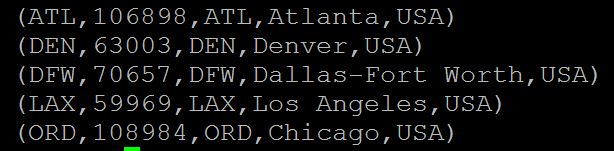
9. In relation joined\_table, we are joining Result and A2 based on a common column, i.e., “dest”

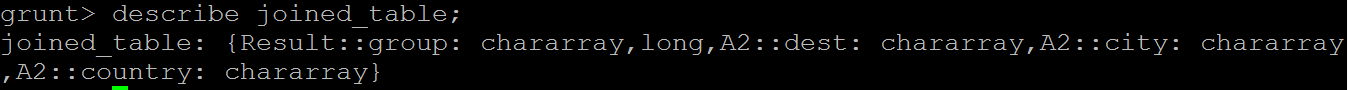
joined\_table = join Result by $0, A2 by dest;



dump joined\_table;





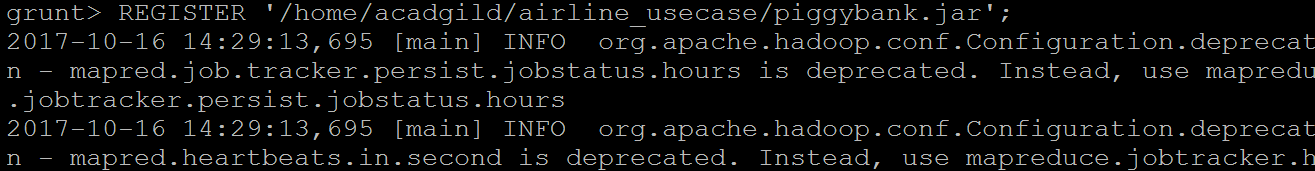


## Problem Statement 2

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

1. We are registering piggybank jar in order to use the CSVExcelStorage class.

REGISTER '/home/acadgild/airline\_usecase/piggybank.jar';

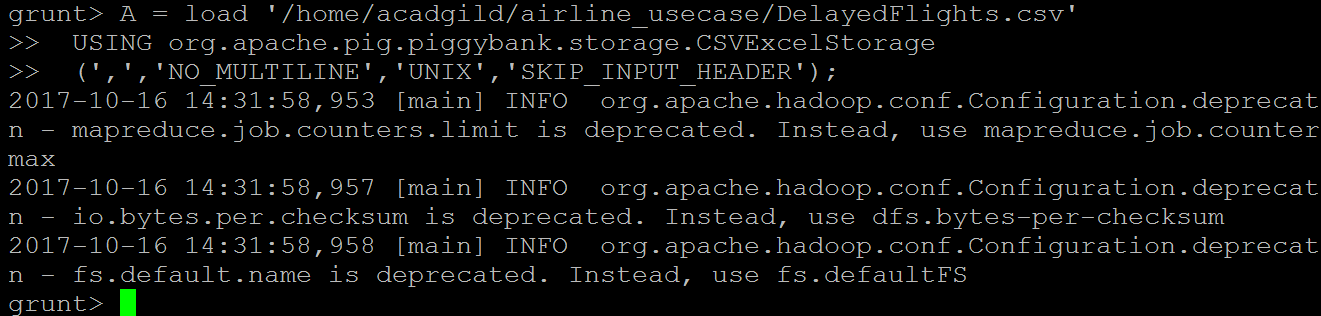


2. In relation **A**, we are loading the dataset using CSVExcelStorage because of its effective technique to handle double quotes and header.

A = load '/home/acadgild/airline\_usecase/DelayedFlights.csv'

USING org.apache.pig.piggybank.storage.CSVExcelStorage

(',','NO\_MULTILINE','UNIX','SKIP\_INPUT\_HEADER');



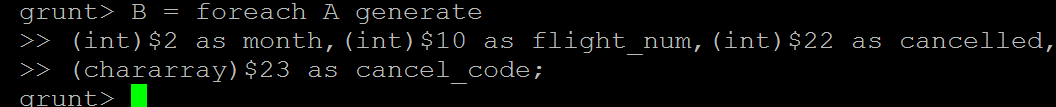


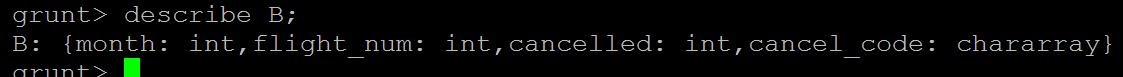
3. In relation **B**, we are generating the columns which are required for processing and explicitly typecasting each of them.

.B = foreach A generate

(int)$2 as month,(int)$10 as flight\_num,(int)$22 as cancelled,

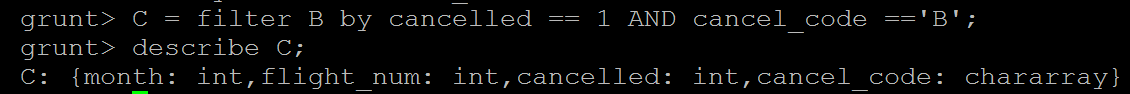
(chararray)$23 as cancel\_code;





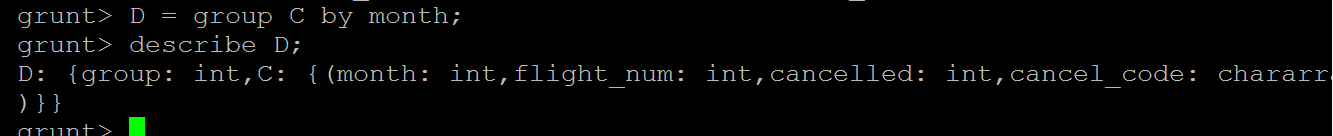
4. In relation **C**, we are filtering the data based on cancellation and cancellation code, i.e.,  canceled = 1 means flight have been canceled and cancel\_code = ‘B’ means the reason for cancellation is “weather.” So relation C will point to the data which consists of canceled flights due to bad weather.

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



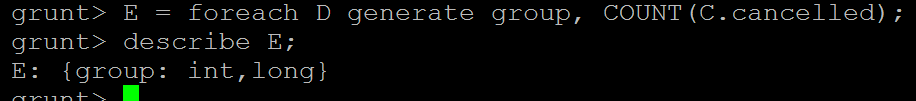
5. In relation**D**, we are grouping the relation C based on every month.

D = group C by month;



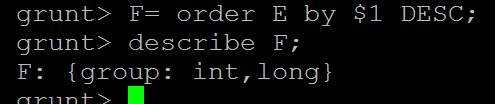
6. In relation **E**, we are finding the count of canceled flights every month.

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

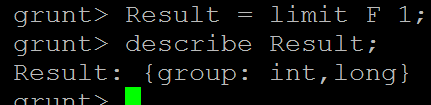


7. Relation **F** and **Result** is for ordering and finding the top month based on cancellation.

F= order E by $1 DESC;



Result = limit F 1;



dump Result;



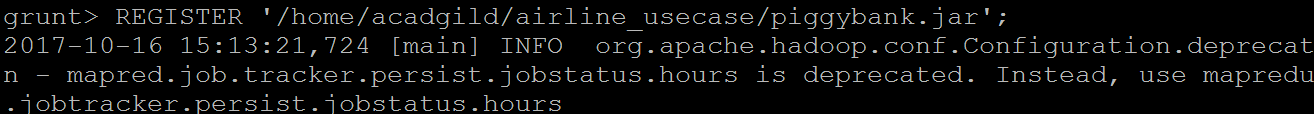


## Problem Statement 3

Top ten origins with the highest AVG departure delay.

1. We are registering piggybank jar in order to use the CSVExcelStorage class

REGISTER '/home/acadgild/airline\_usecase/piggybank.jar';

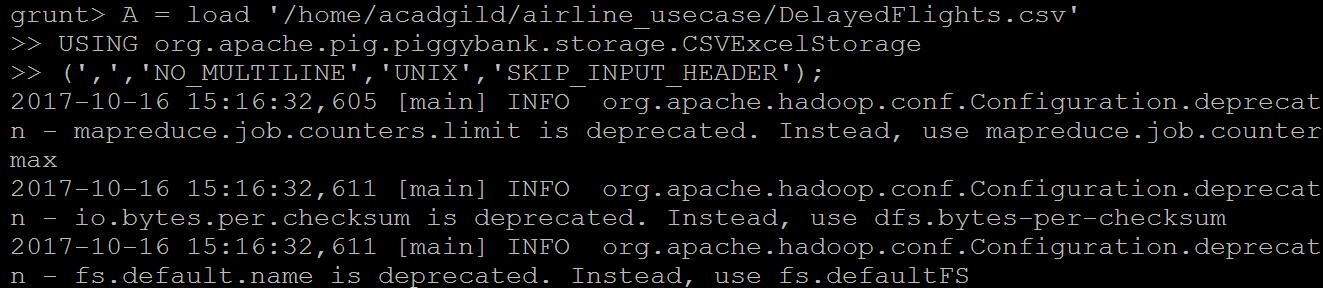


2. In relation **A**, we are loading the dataset using CSVExcelStorage because of its effective technique to handle double quotes and header.

A = load '/home/acadgild/airline\_usecase/DelayedFlights.csv'

USING org.apache.pig.piggybank.storage.CSVExcelStorage

(',','NO\_MULTILINE','UNIX','SKIP\_INPUT\_HEADER');



3. In relation **B1**, we are generating the columns which are required for processing and explicitly typecasting each of them.

B1 = foreach A generate

(int)$16 as dep\_delay, (chararray)$17 as origin;

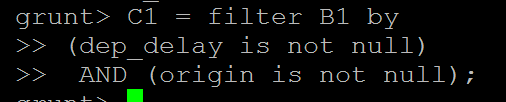
 

4. In relation C1, we are removing the null values fields present if any.

C1 = filter B1 by

(dep\_delay is not null)

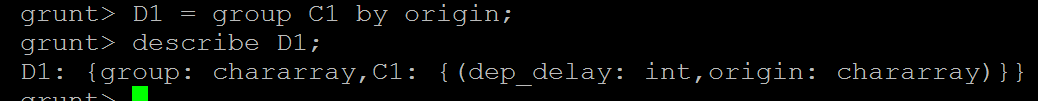
AND (origin is not null);





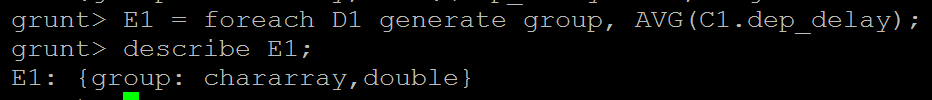
## 5. In relation ****D1****, we are grouping the data based on column “origin.”

D1 = group C1 by origin;



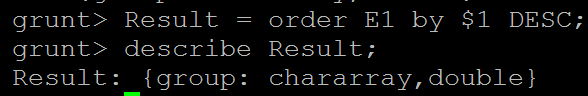
6. In relation **E1**, we are finding average delay from each unique origin.

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



7. Relations named **Result** and **Top\_ten** are ordering the results in descending order and printing the top ten values.

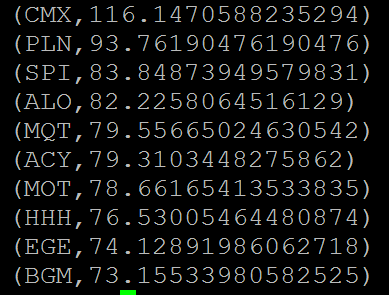
Result = order E1 by $1 DESC;



Top\_ten = limit Result 10;







These steps are good enough to find the top ten origins with the highest average departure delay.

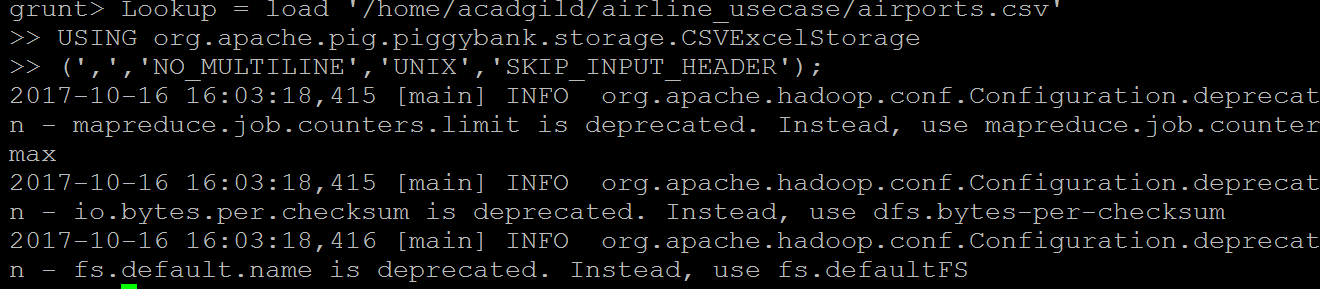
However, rather than generating just the code of origin, we will be following a few more steps to find some more details like **country and city**.

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

Lookup = load '/home/acadgild/airline\_usecase/airports.csv'

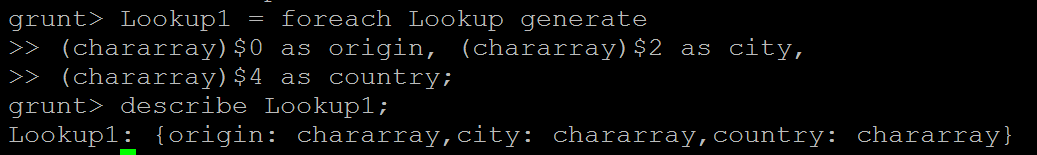
USING org.apache.pig.piggybank.storage.CSVExcelStorage

(',','NO\_MULTILINE','UNIX','SKIP\_INPUT\_HEADER');



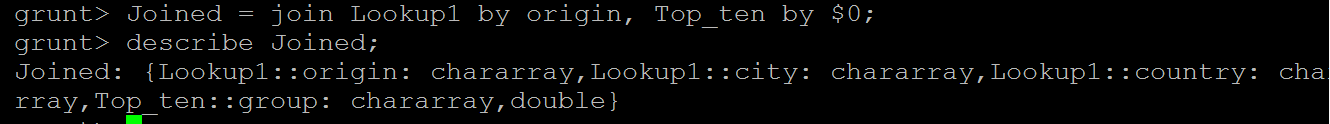


9. In the relation L**ookup1,** we are generating the destination, city, and country from the previous relation Lookup



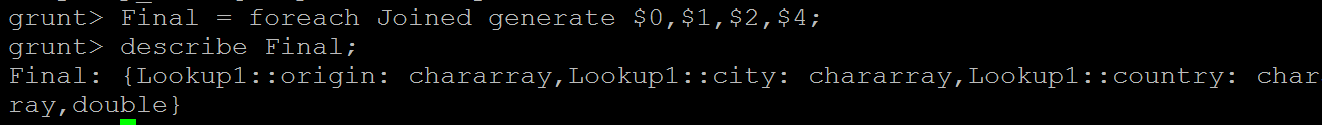
10. In the relation **Joined**, we are joining relation Top\_ten and Lookup1 based on common a column, i.e., “origin.”

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

.

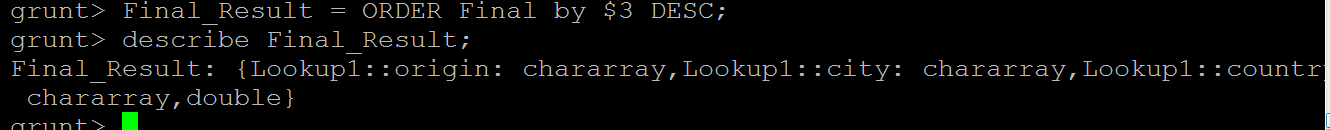
11. In the relation**Final,**we are generating required columns from the Joined table.

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

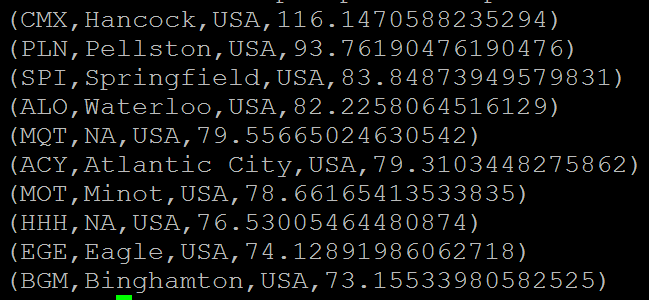


12. Finally, we are ordering and printing the results.

Final\_Result = ORDER Final by $3 DESC;





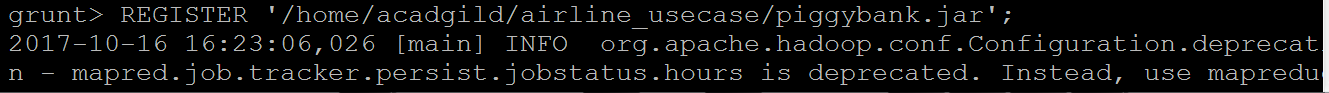


## Problem Statement 4

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

1. We are registering *piggybank* jar in order to use CSVExcelStorage class.

REGISTER '/home/acadgild/airline\_usecase/piggybank.jar';

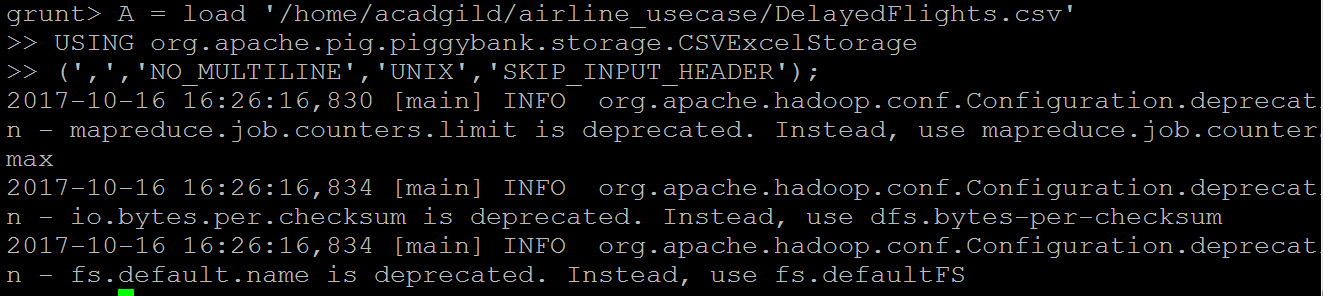


2.In relation **A**, we are loading the dataset using CSVExcelStorage because of its effective technique to handle double quotes and headers.

A = load '/home/acadgild/airline\_usecase/DelayedFlights.csv'

USING org.apache.pig.piggybank.storage.CSVExcelStorage

(',','NO\_MULTILINE','UNIX','SKIP\_INPUT\_HEADER');

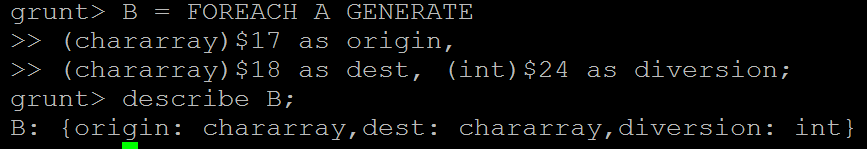


3. In relation **B**, we are generating the columns which are required for processing and explicitly type-casting each of them.

B = FOREACH A GENERATE

(chararray)$17 as origin,

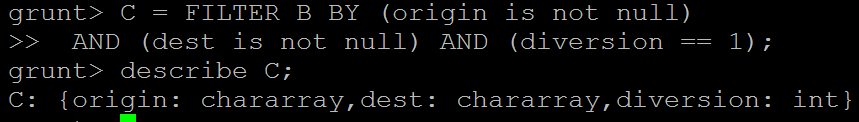
(chararray)$18 as dest, (int)$24 as diversion;



4. 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.

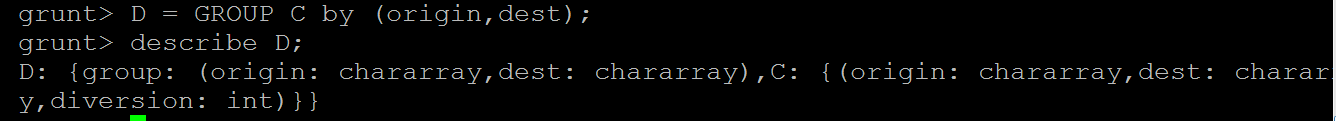
C = FILTER B BY (origin is not null)

AND (dest is not null) AND (diversion == 1);



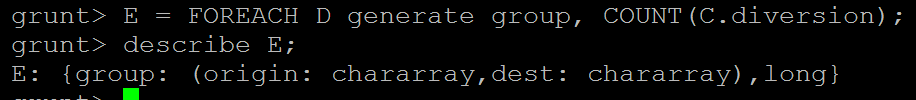
5. In relation **D**, we are grouping the data based on origin and destination.

D = GROUP C by (origin,dest);



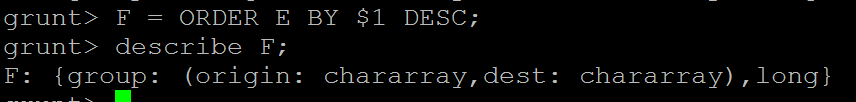
6. Relation **E** finds the count of diversion taken per unique origin and destination.

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

****

7. Relations **F** and **Result** orders the result and produces top 10 results.

F = ORDER E BY $1 DESC;



Result = limit F 10;

dump Result;

