# Input Data Sets:

<https://acadgild.com/blog/pig-use-case-pokemon-data-analysis/>

PIG Use Case: Pokemon Data Analysis

# Loading The Input Data set into the PIG Storage:

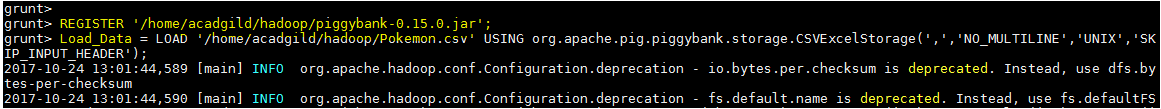
Code:

We are registering the piggybank jar in order to use the CSVExcelStorage class and loading the data into the PIG storage with respective headers have been generated.

**REGISTER '/home/acadgild/hadoop/piggybank-0.15.0.jar';**

**Load\_Data = LOAD '/home/acadgild/hadoop/Pokemon.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO\_MULTILINE','UNIX','SKIP\_INPUT\_HEADER');**

**Random = FOREACH Load\_Data GENERATE(chararray)$1 as Name,(chararray)$2 as Type1,(chararray)$3 as Type2,(int)$4 as Total, (int)$5 as HP,(int)$6 as Attack,(int)$7 as Defense,(int)$8 as SpAtk,(int)$9 as SpDef,(int)$10 as Speed;**



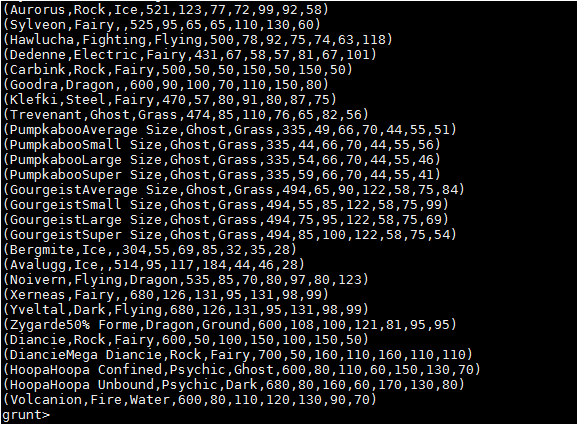
# Ques 1: Find the list of players that have been selected in the qualifying round (DEFENCE>55).

Code:

**selected\_List = FILTER Random BY Defense>55;**



**Output:**

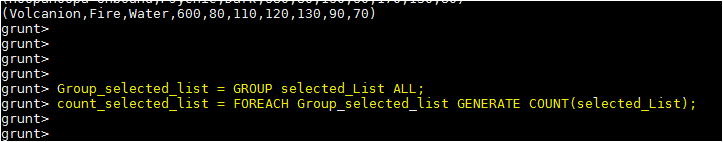


# Ques 2: State the number of players taking part in the competition after getting selected in the qualifying round.

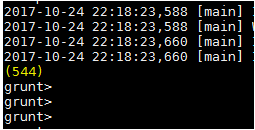
Code:

**Group\_selected\_list = GROUP selected\_List ALL;**

**count\_selected\_list = FOREACH Group\_selected\_list GENERATE COUNT(selected\_List);**



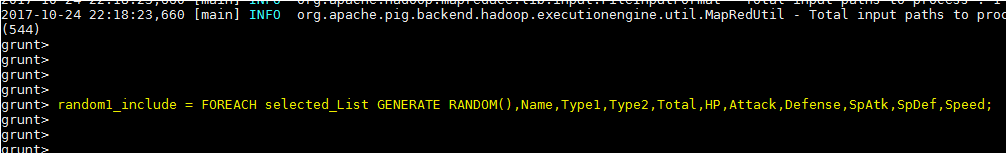
**Output:**



# Ques 3: Using random() generate random numbers for each Pokémon on the selected list.

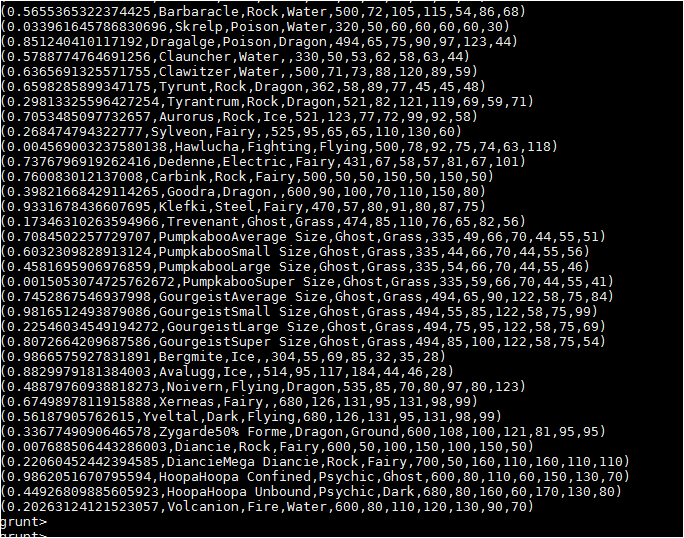
Code:

**random1\_include = FOREACH selected\_List GENERATE RANDOM(),Name,Type1,Type2,Total,HP,Attack,Defense,SpAtk,SpDef,Speed;**



Output:

Hence sample for the list after adding random numbers:

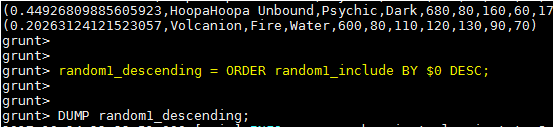


# Ques 4: Arrange the new list in a descending order according to a column randomly.

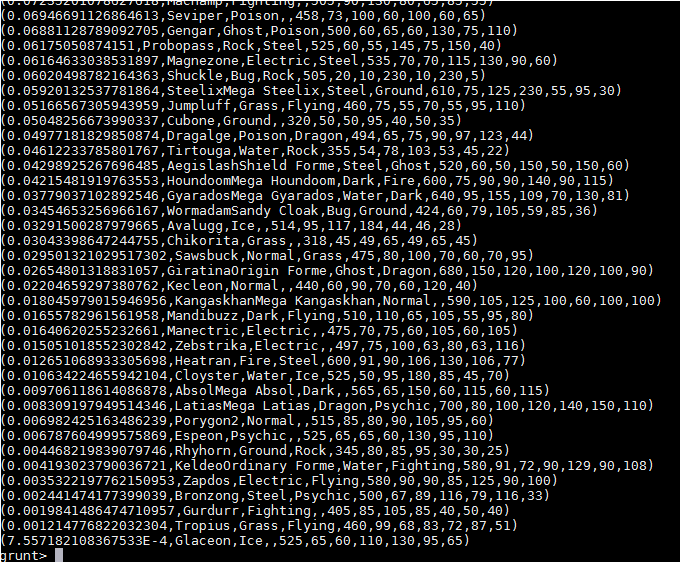
This will give us consequently a layer arranged to pick the random list which 1st player will choose.

Code:

**random1\_descending = ORDER random1\_include BY $0 DESC;**



Output



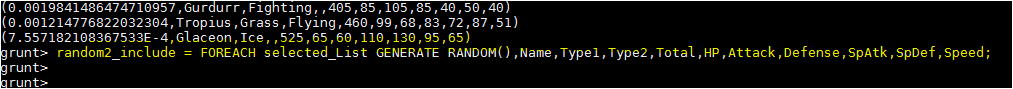
**Yet we want 1 more list with random arrangements of Pokemons which will be therefore chosen by the 2nd player later on.**

# Ques 5: Now on a new relation again associate random numbers for each Pokémon and arrange in descending order according to column random.

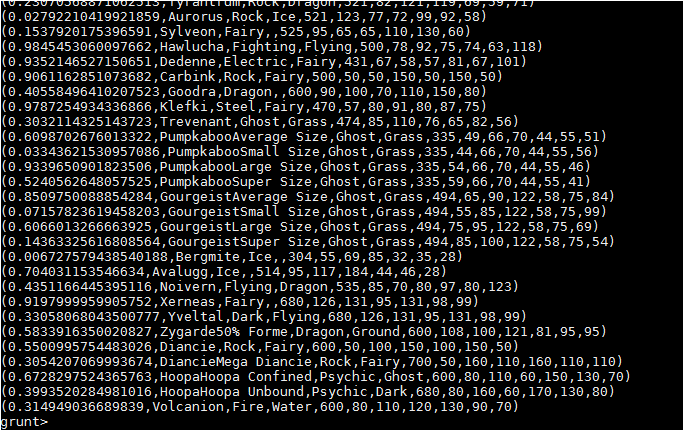
Code:

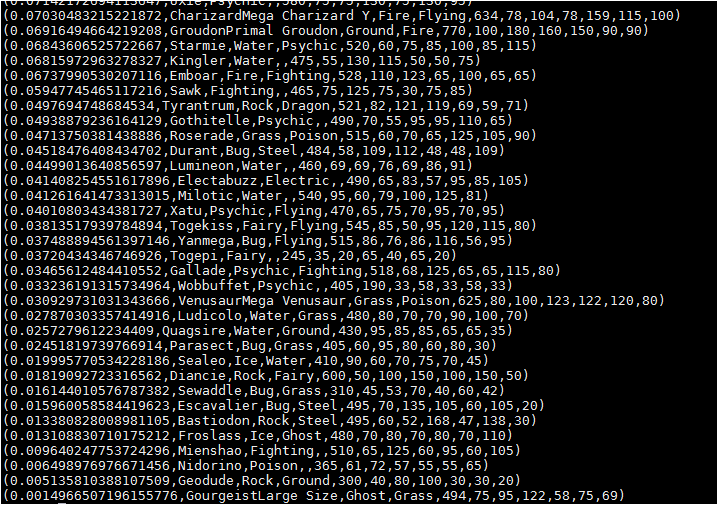
**random2\_include = FOREACH selected\_List GENERATE RANDOM(),Name,Type1,Type2,Total,HP,Attack,Defense,SpAtk,SpDef,Speed;**

**random2\_descending = ORDER random2\_include BY $0 DESC;**



Output:



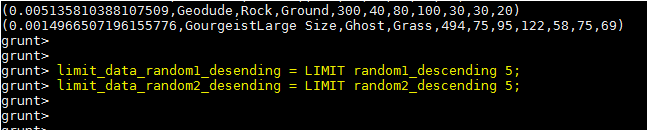


# Ques: From the two different descending lists of random Pokémons, select the top 5 Pokémons for 2 different players.

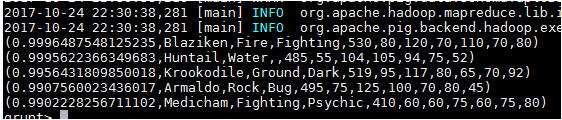
Code:

**limit\_data\_random1\_desending = LIMIT random1\_descending 5;**

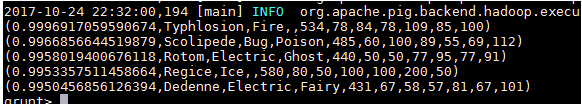
**limit\_data\_random2\_desending = LIMIT random2\_descending 5;**



Output: limit\_data\_random1\_desending



Output: limit\_data\_random2\_desending

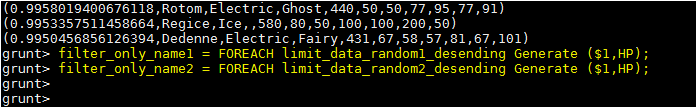


# Ques: Store the data on a local drive to announce for the final match. By the name player1 and player2 (only show the NAME and HP).

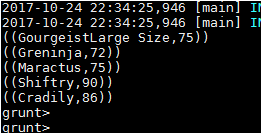
Code:

**filter\_only\_name1 = FOREACH limit\_data\_random1\_desending Generate ($1,HP);**

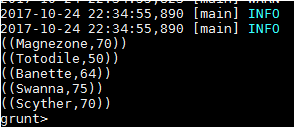
**filter\_only\_name2 = FOREACH limit\_data\_random2\_desending Generate ($1,HP);**



Output: filter\_only\_name1:



Output: filter\_only\_name2



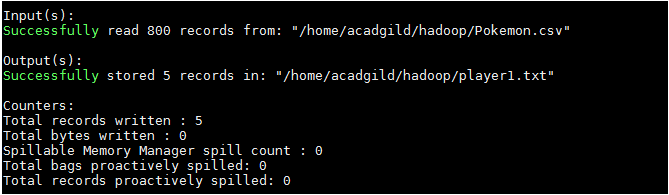
Therefore querying is over using some simple pre-defined functions to get 2 sets of 5 Pokémons, which get select randomly.

In conclusion, let’s store this result in our local system so we can use it as input to our next blog. Especially relevant where we will see UDF using PIG and calculations will be done through user-defined formulas.

Code to store:

***STORE limit\_data\_random1\_desending INTO '/home/acadgild/hadoop/player1.txt';***

As a result



***STORE limit\_data\_random2\_desending INTO ‘/home/acadgild/hadoop/player2.txt';***

