# **ETL con PIG**

# 1 Base de datos

La base de datos que voy a usar la he sacado de: <https://www.kaggle.com/datasets> , para ser más exactos, he escogido una en la que se tratan eventos que pasan en partidos de futbol: <https://www.kaggle.com/secareanualin/football-events>

Enlace a mi Dropbox: <https://www.dropbox.com/sh/qwmh56qagnjblx1/AABWw4d-xNamPMjJH-_gE2Sta?dl=0>

Este dataset se divide en 3 ficheros:

* **Dictionary.txt**: en el que se especifican los detalles de algunos valores numéricos de las variables.
* **Ginf.csv:** Un fichero con algunos detalles sobre los partidos que luego se verán en los eventos.
* **Events.csv:** Dataset principal en el que tenemos 941009 instancias, con 22 atributos y en el que se tienen eventos que suceden durante los partidos. Los atributos y su descripción se pueden ver en la web de descarga del dataset, pero además lo adjunto en el Anexo 1, al final de documento.

Por lo tanto, he subido al servidor este dataset para realizar nuestras pruebas con pig.

# 2 Pig

## 2.1 Carga del dataset

Como podemos ver en el script, lo primero que hacemos es, tras abrir pig con “pig –x local”, es cargar el fichero csv con el siguiente comando:

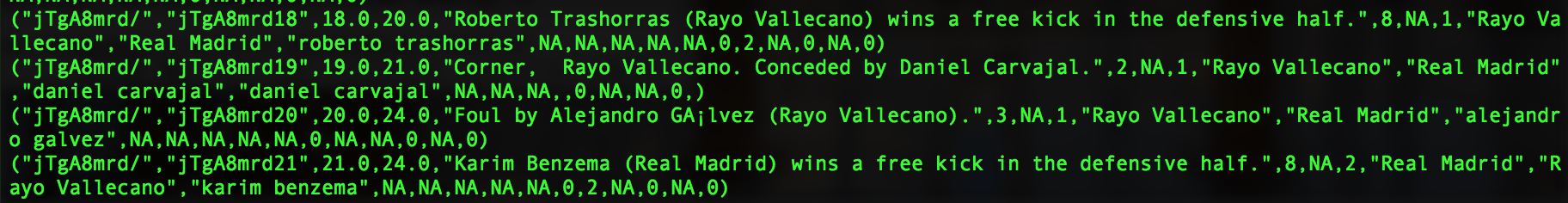
dt\_event = load 'input/events.csv' using PigStorage(',') AS (id\_odsp:chararray, id\_event:chararray, sort\_order:float, time:float, text:chararray, event\_type:chararray, event\_type2:chararray, side:chararray, event\_team:chararray, opponent:chararray, player:chararray, player2:chararray, player\_in:chararray, player\_out:chararray, shot\_place:chararray, shot\_outcome:chararray, is\_goal:int, location:chararray, bodypart:chararray, assist\_method:chararray, situation:chararray, fast\_break:int);

Donde a cada variable le asignamos el tipo de dato que es.

Si lo que queremos es mostrar esta lectura podemos hacer:

dump dt\_event;

Quedando algo como lo que vemos a continuación:



Esta lectura la podríamos almacenar con: (estará al final del script)

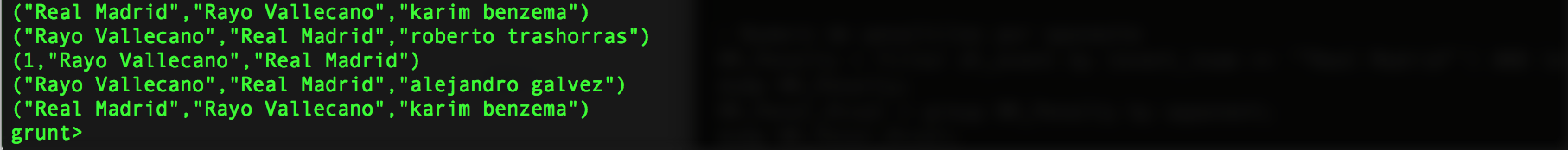
store dt\_event into 'pigResults/EventsProcessed';

## 2.2 Proyección

Si queremos una proyección simple para ver cómo funciona esta, podemos quedarnos con los equipos que realizan cada evento y el jugador principal del evento con:

Equipos\_y\_Jugador = foreach dt\_event generate event\_team, opponent, player;

Si lo mostramos con dump, podemos ver el resultado como el siguiente:



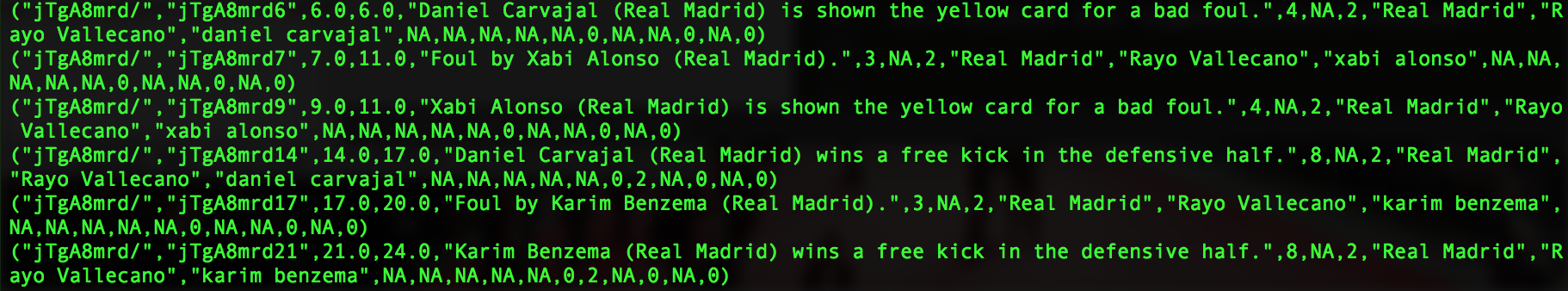
Donde vemos el resultado de las variables que hemos decido obtener de nuestro dataset.

## 2.3 Selección

He realizado 3 selecciones distintas, para poder hacer 3 agrupamientos y resúmenes distintos para ver distintos resultados. Veamos por lo tanto estas selecciones.

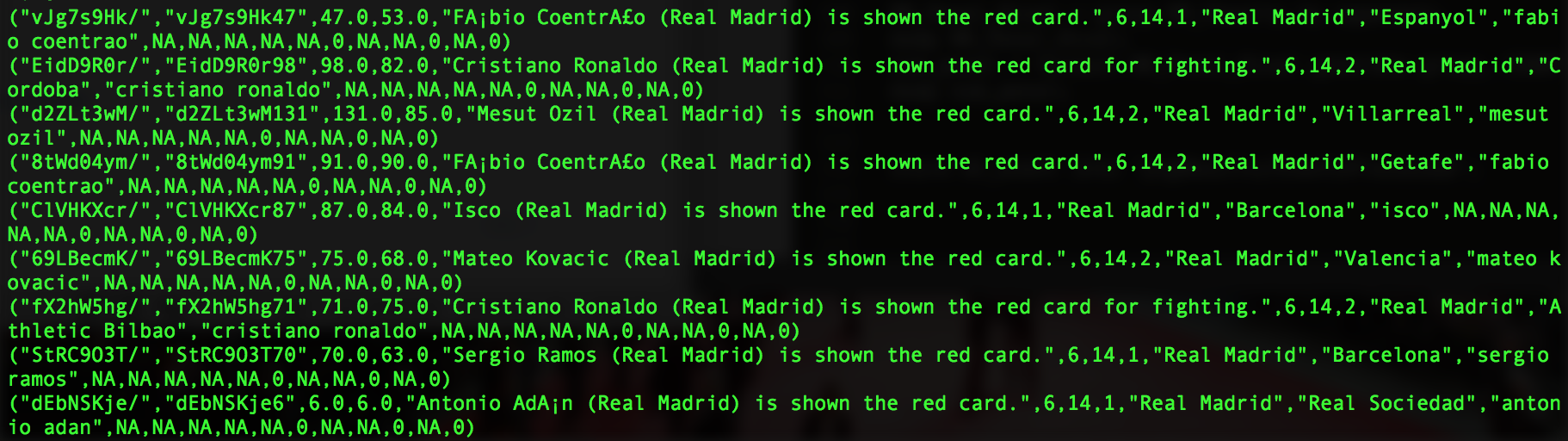
Seleccionaremos los eventos en los que el equipo local ha sido el Real Madrid o la Juventus:

RM\_J\_Local = filter dt\_event by (event\_team == '"Real Madrid"') OR (event\_team == '"Juventus"');



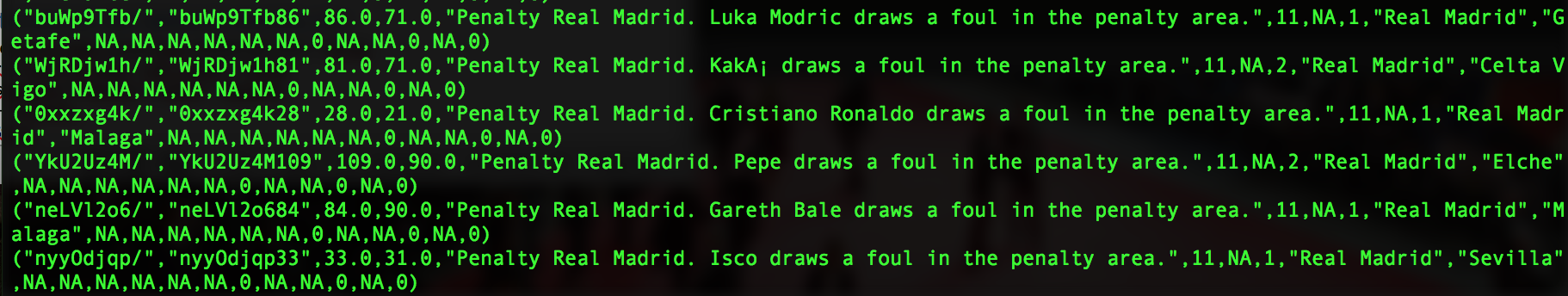
Seleccionaremos los eventos en los que el equipo local es el Real Madrid y el evento ha sido una tarjeta roja, con el atributo event\_type igual a 6:

RM\_RedCard = filter dt\_event by (event\_team == '"Real Madrid"') AND (event\_type == '6');



Seleccionaremos los eventos en los que el equipo local es el Real Madrid y el evento ha sido un penalti, con el atributo event\_type igual a 11:

RM\_Penalty = filter dt\_event by (event\_team == '"Real Madrid"') AND (event\_type == '11');



## 2.4 Agrupamientos y resúmenes

En base a nuestras selecciones del apartado anterior vamos a agrupar las 3 selecciones por el equipo rival y vamos a realizar un resumen para cada caso.

Si queremos ver la media del minuto en el que ocurren los eventos, para la primera selección, podemos ver como la mayoría de los resultados es el minuto 45, como era de esperar ya que los eventos ocurren entre el minuto 0 y el 90:

RM\_J\_Rival = group RM\_J\_Local by opponent;

minuto\_medio = foreach RM\_J\_Rival generate group, AVG(RM\_J\_Local.time) as dt\_event;



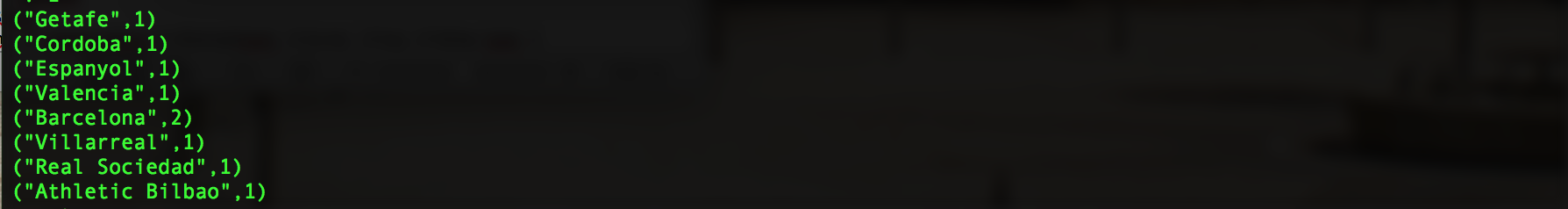
Si nos centramos en la segunda selección, vamos a contar cuantas tarjetas rojas se tienen según el rival al que se enfrenta:

RM\_RC\_Rival = group RM\_RedCard by opponent;



Ahora veamos la cuenta:

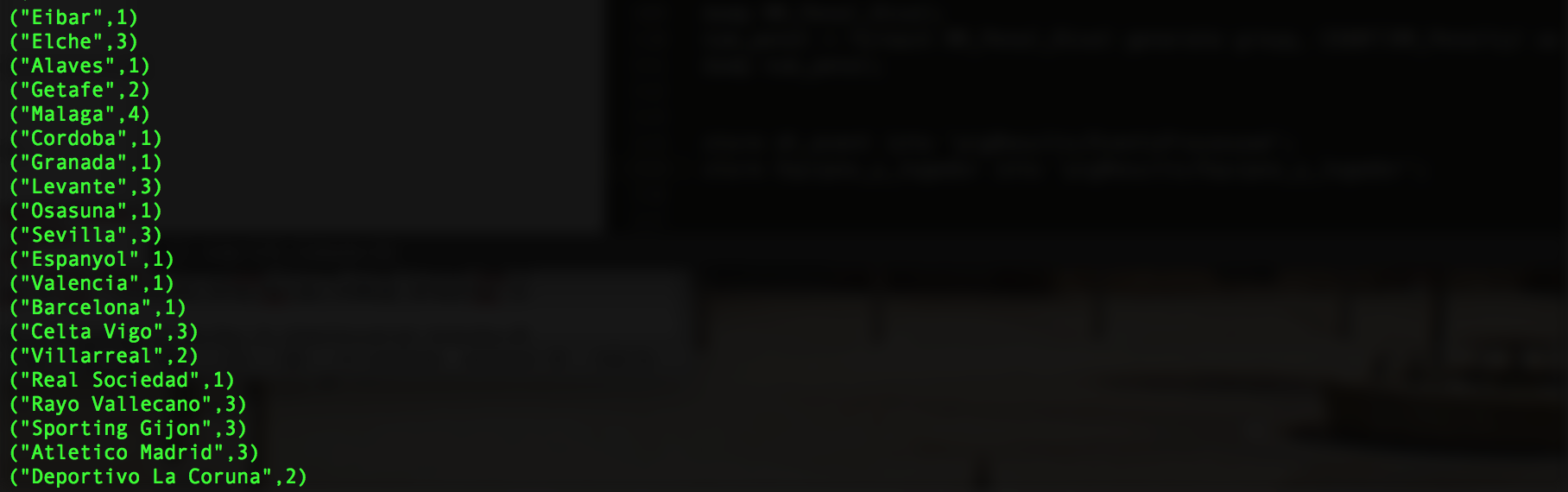
num\_rojas = foreach RM\_RC\_Rival generate group, COUNT(RM\_RedCard) as dt\_event;



SI hacemos lo mismo, para la tercera selección, vemos el número de penaltis en función del oponente:

RM\_Penal\_Rival = group RM\_Penalty by opponent;

num\_penal = foreach RM\_Penal\_Rival generate group, COUNT(RM\_Penalty) as dt\_event;



# 3 Conclusiones

Hemos realizado pruebas en Pig en modo local. Hemos probado las proyecciones, selecciones, agrupamientos y resúmenes. Se ha visto lo fácil que es realizar este tipo de acciones con esta herramienta y como se puede tratar con muchos tipos de datos.

# Anexo 1: 22 características del dataset events.csv

id\_odsp -> unique identifier of game (odsp stands from oddsportal.com) -> String

id\_event -> unique identifier of event (id\_odsp + sort\_order) -> String

sort\_order -> chronological sequence of events in a game -> Numeric

time -> minute of the game -> Numeric

text -> text commentary -> String

event\_type -> primary event. 11 unique events (1-Attempt(shot), 2-Corner, 3-Foul, 4-Yellow Card, 5-Second yellow card, 6-(Straight) red card, 7-Substitution, 8-Free kick won, 9-Offside, 10-Hand Ball, 11-Penalty conceded) -> String

event\_type2 -> secondary event. 4 unique events (12 - Key Pass, 13 - Failed through ball, 14-Sending off, 15-Own goal) -> String

side -> 1-Home, 2-Away -> String

event\_team -> team that produced the event. In case of Own goals, event team is the team that beneficiated from the own goal -> String

opponent -> Help us describe this column -> String

player -> name of the player involved in main event (converted to lowercase and special chars were removed) -> String

player2 -> name of player involved in secondary event -> String

player\_in -> player that came in (only applies to substitutions) -> String

player\_out -> player substituted (only applies to substitutions) -> String

shot\_place -> placement of the shot (13 possible placement locations, available in the dictionary, only applies to shots) -> String

shot\_outcome -> 4 possible outcomes (1-On target, 2-Off target, 3-Blocked, 4-Hit the post) -> String

is\_goal -> binary variable if the shot resulted in a goal (own goals included) -> Boolean

location -> location on the pitch where the event happened (19 possible locations, available in the dictionary) -> String

bodypart -> (1- right foot, 2-left foot, 3-head) -> String

assist\_method -> in case of an assisted shot, 5 possible assist methods (details in the dictionary) -> String

situation -> 4 types: 1-Open Play, 2-Set piece (excluding Direct Free kicks), 3-Corner, 4-Free kick -> String

fast\_break -> binary -> Boolean