Big Data Final Report

Group B_8

Na An 474995

Jareau Liang 473915

Edward Liu 473935

Yuhan Xu 474154

Contents

Introduction of PUBG

PUBG is a player versus player shooter game in which over 95 players fight in a battle royale, a type of large-scale last man standing deathmatch where players fight to remain the last alive. Each match starts with players parachuting from a plane onto one of the four maps where they are to scavenge towns and buildings for weapons, ammo, armor and first-aid. Players will then decide to either fight or hide with the ultimate goal of being the last one standing. Every few minutes, the playable area of the map begins to shrink down towards a random location, with any player caught outside the safe area taking damage incrementally, and to corral players closer and closer together.

Description of the data

In this Kaggle Dataset, KP provided over 720,000 competitive matches from the popular game PlayerUnknown's Battlegrounds. The data was extracted from pubg.op.gg, a game tracker website. The data is from 20th Oct. 2017 to 10th Jan. 2018. We selected the first two files in the original file for analysis, they contains 299984 competitive matches data. The data is structured.
This dataset provides two zips: aggregate and deaths:

- In deaths, the files record every death that occurred within the 720k matches. That is, each ro w documents an event where a player has died in the match. It has 12 features columns, and 2 6867235 rows records. The size of the data is 4 GB.
- In aggregate, each match's meta information and player statistics are summarized (as provided by pubg). It includes various aggregate statistics such as player kills, damage, distance walke d, etc as well as metadata on the match itself such as queue size, fpp/tpp, date, etc. It has 15 fe atures columns, and 27693560 rows records.

The link of the data is as below:
https://www.kaggle.com/skihikingkevin/pubg-match-deaths#agg_match_stats_4.csv

Problem statement

From the data of matches, We come up with 8 questions to learn about the effects of some strategic decisions made by players during the game so that we can offer players with various recommendations for their future games. Besides, we also try to find out some specific players, like the one who has killed most people per match, and the one who has participated most games among all the matches. Those are the fun facts about PUBG and PUBG fans may want to make friends with those players or keep away from those players in their future games. Our specific questions are listed below:
1. Which weapon is the best one?
2. What is the most dangerous location in each PUBG map?
3. What is the best location to kill another player in each PUBG map?
4. Who is the best player in this game? Which player has the highest average kills per match?
5. Will moving more in the games leads to a better rank?
6. Will a higher number of total kills of a team leads to a better rank for that team?
7. Which player has participated in most games?
8. Which player has highest average DBNO per match? Which player has the highest average assist per match? You can say they are the luckiest and the most helpful players.

In the fourth section of this report, we will interpret and solve those questions using big data analytics tools, and present the visualization using Tableau and Python.

Why big data

The dataset we found contains 5 csv files for aggregate match information and 5 csv files for killing information in each match. Among those files, we chose the first 2 files for each segment, which contains about 8GB of data in total. We chose this dataset because some of our group members were huge fans of PUBG. We wanted to discover some secrets from the PUBG game profile.

Methods & results

Question 1. Which weapon is the best one?

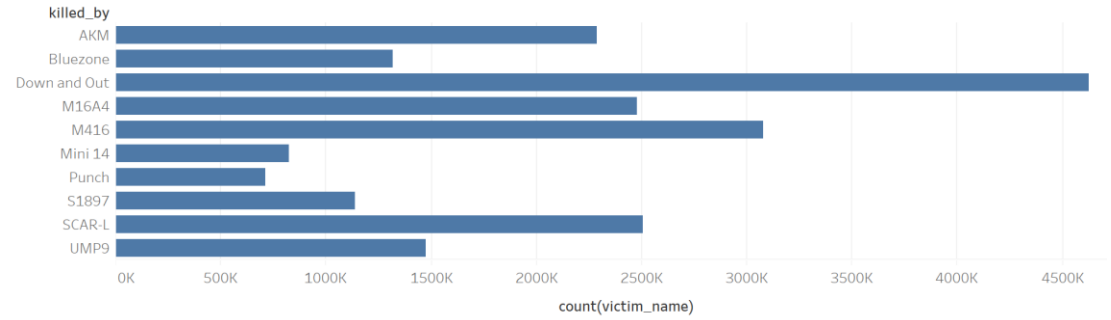
We used PySpark to solve this problem.

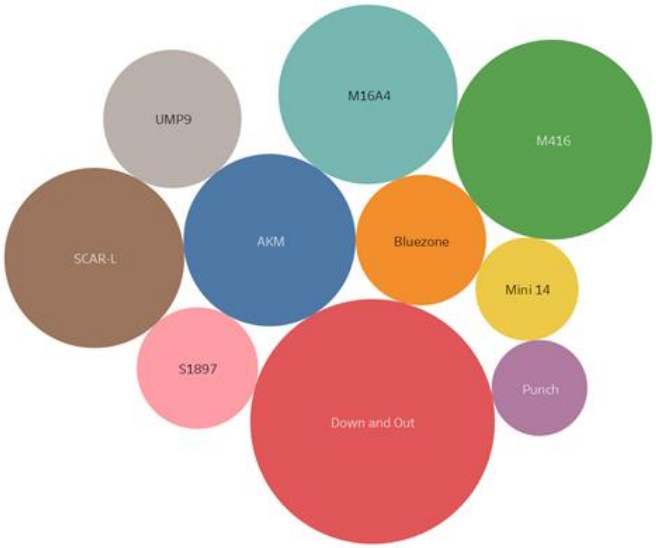
We assume that a better weapon can kill more people. Therefore, we count the number of people each type of weapon kills. We group by “killed_by” column, which contains the different types of weapons.

The top10 types of weapon and the number of victims are:

killed_by	count(victim_name)
Down and Out	4623813
M416	3079123
SCAR-L	2504612
M16A4	2479867
AKM	2289607
UMP9	1477190
Bluezone	1314890
S1897	1139445
Mini 14	821572

Then we use Tableau to visualize the result:





From the results, we can see that most useful weapon except “down and out” is M416.

Question 2. What is the most dangerous location in each PUBG map?

The method we used was Impala.

First, we removed the header of kill_match_stats_final_0.csv and kill_match_stats_final_1.csv, then merge them into one csv. We put it into hdfs and changed the mode accordingly.

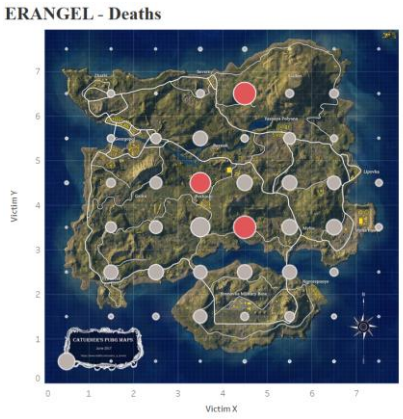
Second, we created the table called Kills and loaded the data into the table via Hive (I only used Hive in this part).

Third, we found out that there are two maps in this data set, which are ‘MIRAMAR’ and ‘ERANGEL’.
Fourth, since the author of this data stated that in both maps, the range of both x-axis and y-axis is (0,800000), so we decided to divide each map into 64 sub-sections, each sub-section covers an area of 100000*100000. We used case statement to create two new attributes, victim_x, victim_y, each of them having values from (0.5, 1.5, 2.5, 3.5, 4.5, 5.5, 6.5, 7.5) and each combination of victim_x and victim_y corresponding to a sub-section in the map. For example, when victim_x =1.5, victim_y =2.5, it’s related to the original area whose victim_position_x values is in (100000,200000) and whose victim_position_y value is in (200000,300000). The reason that we used ‘.5’ format because we need the (victim_x,victim_y) to correspond to the center of the area it’s related to in the original map, and also for visualization reasons.
Last, for each map, we group by (victim_x, victim_y) and show the results in descending order of the total deaths (defined by count(match_id)). So we have a list of locations in an order of how dangerous they are.

	victim_x	victim_y	total_deaths
1	3.5	4.5	833503
2	4.5	2.5	432558
3	4.5	5.5	310599
4	3.5	3.5	302414
5	3.5	2.5	267621
6	4.5	3.5	229572
7	0.5	0.5	226759
8	5.5	3.5	162674
9	4.5	4.5	145557
10	1.5	2.5	137012
11	2.5	5.5	123485

The picture above shows part of the results for map 'MIRAMAR’.
We used Tableau to visualize the results we got and used the real map in the game as the background so people who are familiar with the game will know which place is most dangerous. The size of each circle indicates how many deaths happened in that place, and the three red circle indicates the top 3 dangerous places.
For map 'MIRAMAR’:

For map ‘ERANGEL’:



Question 3. What is the best location to kill another player in each PUBG map?

Generally, this question is similar to question two. The only difference is that we used killer_x and killer_y to correspond to original attributes killer_position_x and killer_position_y. In the figure below, the size of each circle indicates how many kills happened in that place, and the three green circle indicates the top 3 most desirable killing places. For your information, the result may not be the same as question 2 since the location of a player who died is not the same as where the killer stands.

For map 'MIRAMAR':

For map 'ERANGEL':

Question 4. Who is the best player in this game? Which player has the highest average kills per match?

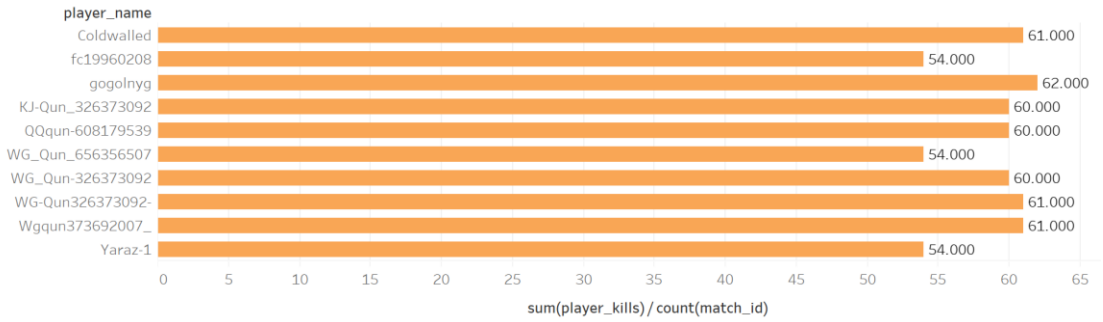
We used Impala to solve this problem.

The best player should kill more people in this game. So, we counted the number of victims killed each math and group by "player_name".

The top10 best players and the average kills are:

	player_name	sum(player_kills) / count(match_id)
1	gogolnyg	62
2	Wgqun373692007_	61
3	Coldwalled	61
4	WG-Qun326373092-	61
5	WG_Qun-326373092	60
6	KJ-Qun_326373092	60
7	QQqun-608179539	60
8	WG_Qun_656356507	54
9	fc19960208	54
10	Yaraz-1	54

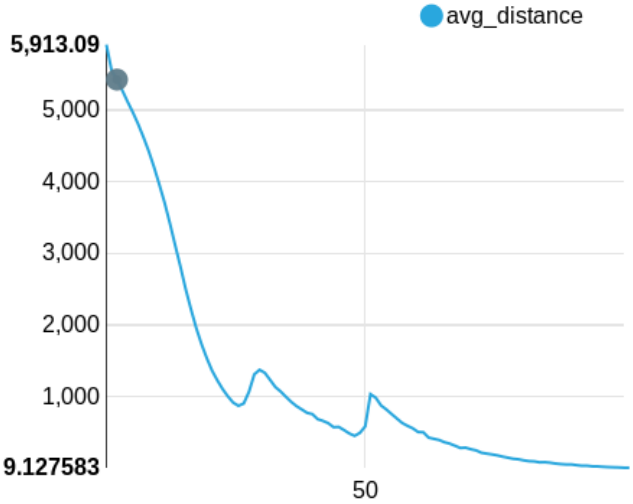
Then use Tableau to visualize the result:



Question 5. Will moving more in the games leads to a better rank?

We used Impala to solve this problem.

The total distance equals to the walk distance plus the ride distance. We sum the average total distance of the team that has the same rank in each match. Then we let rank to be x=axis, average distance to be y-axis and plot the graph.

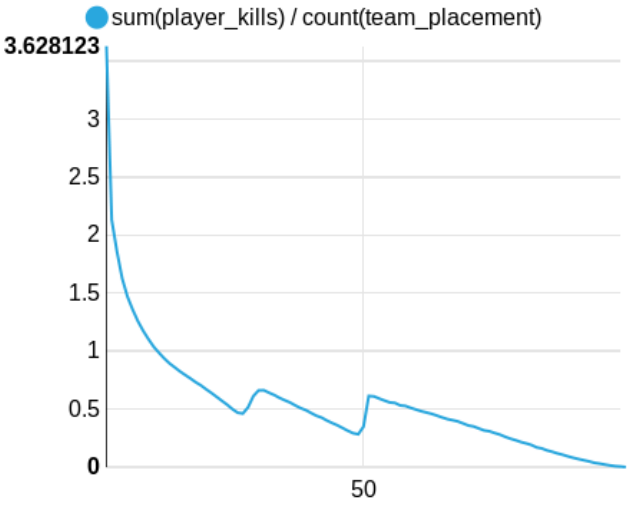


From the graph, we can see a negative correlation between x and y. Therefore, moving more in the game will lead to a better rank.

Question 6. Will a higher number of total kills of a team leads to a better rank for that team?

We used Impala to solve this problem.

We count the average number of kills of the team that has the same rank in each match. Then, we let rank to be x-axis, average number of kills to be y-axis and plot the graph.



From the graph, we can see a negative correlation between x and y. Thus, higher total number of kills leads to a better rank.

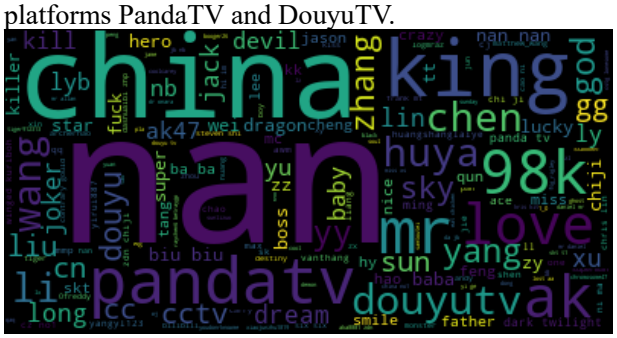
Question 7. Which player has participated in most games?

The method we used was pyspark.

Using the merged data, we counted distinct match id for each player name and order it in descending sequence. Since there are some blanks in the player_name column, we removed those rows using having function. We found that VanThang was the most addicted player.

Frequency	Player
688	VanThang
577	Matthew_wang
570	coolcarey
567	JZalan
542	hzxiaobin
502	Slh_Bunny
448	huangshanglaiye
440	jonelycai
440	SDFSADFASDF
434	AKA8881

We also created a wordcloud image to see the most frequently used phrases in player names. We can tell that most PUBG players were from China, especially from the most popular live stream platforms PandaTV and DouyuTV.



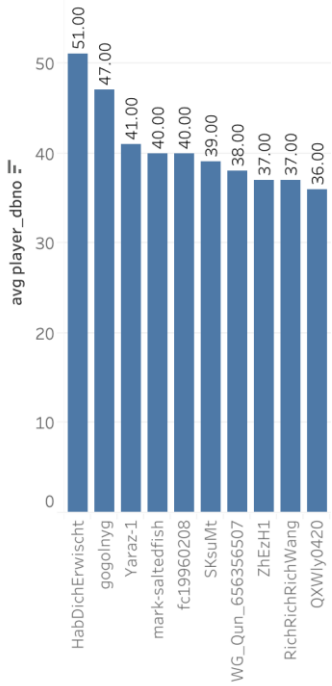
Question 8. Which player has highest average DBNO per match? Which player has the highest average assist per match?

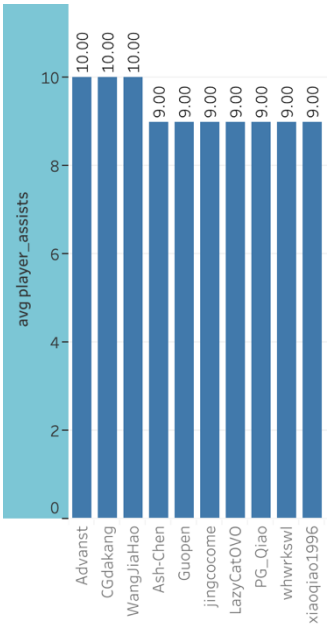
The method we used was Impala.

We calculated the average DBNO (down but no out) and average assists per match according to player name. Also we ignored those players with no player name. Surprisingly, we found that the luckiest player was saved 51 times on average, while the most helpful player assisted team members 10 times on average.

	player_name	avg_saved
1	HabDichErwischt	51
2	gogolnyg	47
3	Yaraz-1	41
4	fc19960208	40
5	mark-saltedfish	40
6	SKsuMt	39
7	WG_Qun_656356507	38
8	ZhEzH1	37
9	RichRichRichWang	37
10	QXWly0420	36

	player_name	avg_save
1	WangJiaHao	10
2	CGdakang	10
3	Advanst	10
4	jingcocome	9
5	whwrkswl	9
6	Guopen	9
7	LazyCatOVO	9
8	PG_Qiao	9
9	Ash-Chen	9
10	xiaoqiao1996	9





Conclusion

From the result of question 1, we know that the most useful weapon except “down and out” is M416. We recommend players can try to practice M416 more to kill more players in the game and get higher ranking.

From the results of question 2 and 3, we can see that for map 'MIRAMAR’, the areas (300000-400000, 400000-500000), (400000-500000, 200000-300000), and (400000-500000,500000-600000) are where most deaths and kills happened. For map ‘ERANGEL’, the areas (400000-500000,600000-700000), (400000-500000,300000-400000) and (300000-400000,400000-500000) are where most deaths and kills happened. The competition in those areas are most severe, so aggressive players may enjoy the competition there, while for those who are used to spend more time hiding and waiting for opportunities, you’d better stay away from those areas.

From the result of question 4, we know that the best player is gogolnyg. He also has the second largest average saved times, which means that he kills more and is saved more.

From the result of question 5 and 6, we know that to have a higher rank, players should move more and kills more.

From the results of question 7 and 8, we can see that China was one of the largest markets for PUBG while many players were from live stream platforms. Besides, the luckiest player was HabDichErwischt and the most helpful players were WangJiaHao, CGdakang and Advanst.

Appendix

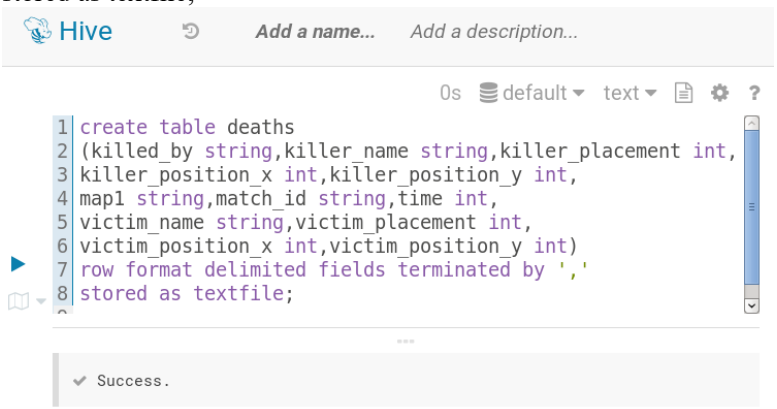
Preprocessing

#Merge two aggregate databases and load databases into impala

cat kill_match_stats_final_0.csv <(tail -n+2 kill_match_stats_final_1.csv) > deaths.csv
hdfs dfs -put deaths.csv
hdfs dfs -chmod 777 deaths.csv

```
[cloudera@quickstart deaths]$ cat kill_match_stats_final_0.csv <(tail -n+2 kill_
match_stats_final_1.csv) > deaths.csv
[cloudera@quickstart deaths]$ hdfs dfs -put deaths.csv
[cloudera@quickstart deaths]$ hdfs dfs -chmod 777 deaths.csv
```

create table deaths
(killed_by string,killer_name string,killer_placement int,
killer_position_x int,killer_position_y int,
map1 string,match_id string,time int,
victim_name string,victim_placement int,
victim_position_x int,victim_position_y int)
row format delimited fields terminated by ','
stored as textfile;



LOAD DATA INPATH "/user/cloudera/deaths.csv" INTO TABLE deaths;

cat agg_match_stats_0.csv <(tail -n+2 agg_match_stats_1.csv) > aggregate.csv

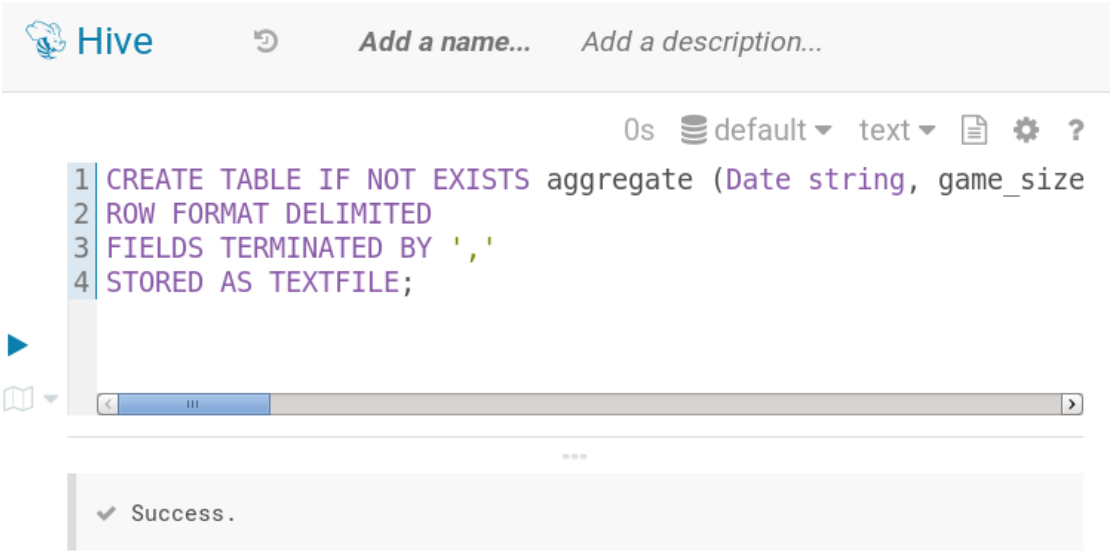
```
[cloudera@quickstart aggregate]$ cat agg_match_stats_0.csv <(tail -n+2  agg_matc
h_stats_1.csv) > aggregate.csv
```

hdfs dfs -put aggregate.csv
hdfs dfs -chmod 777 aggregate.csv

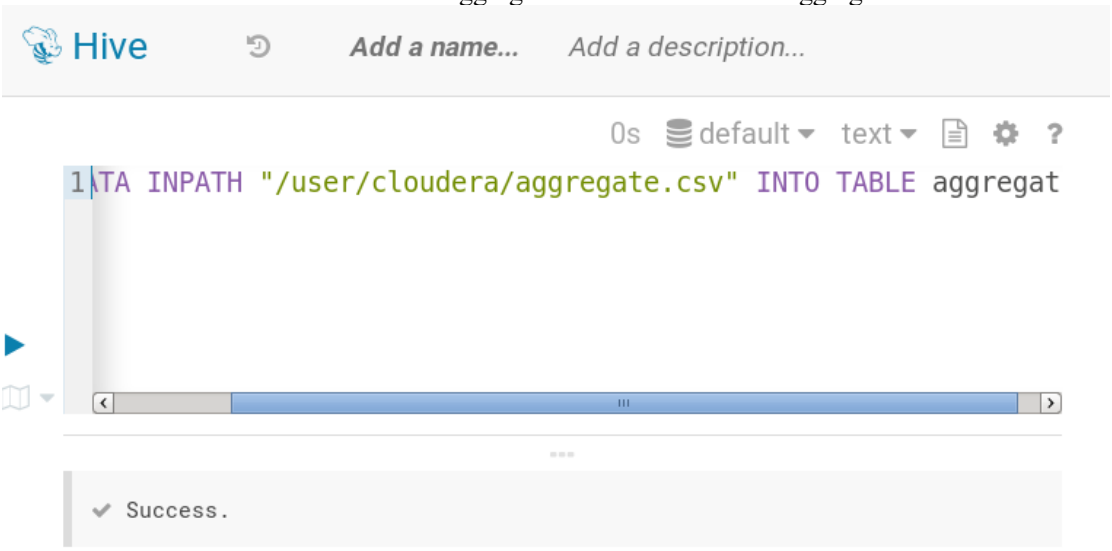
```
[cloudera@quickstart Desktop]$ hdfs dfs -put aggregate.csv
[cloudera@quickstart Desktop]$ hdfs dfs -chmod 777 aggregate.csv
[cloudera@quickstart Desktop]$
```

CREATE TABLE IF NOT EXISTS aggregate (Date string, game_size int, match_id string,

match_mode string, party_size int, player_assists int, player_dbno int, player_dist_ride float, player_dist_walk float, player_dmg int, player_kills int, player_name string, player_survive_time int, team_id int, team_placement int)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;



LOAD DATA INPATH "/user/cloudera/aggregate.csv" INTO TABLE aggregate



For questions 2 and 3, one of our team member created tables with different table names, and his codes are shown as follow:

Remove header + merge csvs + upload into hdfs and change mode

tail -n+2 kill_match_stats_final_1.csv > noheader_1.csv
tail -n+2 kill_match_stats_final_0.csv > noheader_0.csv
cat noheader_0.csv noheader_1.csv > merge_kill.csv
hdfs dfs -put merge_kill.csv
hdfs dfs -chmod 777 merge_kill.csv

Create table + load data into table + show the first 5 result of the table

create table kills (killed_by string,killer_name string,killer_placement float,

killer_position_x float,killer_position_y float,map_name string,match_id string,time int, victim_name string,victim_placement float,victim_position_x float, victim_position_y float) row format delimited fields terminated by ','

load data inpath '/user/cloudera/merge_kill.csv' into table kills
select * from kills limit 5

Query History Saved Queries Results (5)

	kills.killed_by	kills.killer_name	kills.killer_placement	kills.killer_position_x	kills.killer_position_y	kills.map_name
1	Grenade	KrazyPortuguese	5	657725.125	146275.203125	MIRAMAR
2	SCAR-L	nide2Bxiaojiejie	31	93091.3671875	722236.375	MIRAMAR
3	S686	Ascholes	43	366921.40625	421623.90625	MIRAMAR
4	Down and Out	Weirdo7777	9	472014.1875	313274.8125	MIRAMAR
5	M416	Solayuki1	9	473357.8125	318340.5	MIRAMAR

Questions

Question 1. Which weapon is the best one?

hdfs dfs -put deaths.csv
pyspark
line_rdd = sc.textFile("deaths.csv")
header = line_rdd.first()
rows_rdd = line_rdd.filter(lambda line: line != header)
data_rdd = rows_rdd.map(lambda x: s.split(",")).map(lambda x: (x[0].encode("ascii"), x[8].encode("ascii")))
df = sqlContext.createDataFrame(data_rdd, ["killed_by", "victim_name"])
df2 = df.groupBy("killed_by").agg({"victim_name":"count"})
from pyspark.sql.functions import desc
df3 = df2.sort(desc("count(victim_name)"))
df3.show(10)

```
>>> df3 = df2.sort(desc("count(victim_name)"))
>>> df3.show(10)
+------------+------------------+
|   killed_by|count(victim_name)|
+------------+------------------+
|Down and Out|           4623813|
|        M416|           3079123|
|      SCAR-L|           2504612|
|       M16A4|           2479867|
|         AKM|           2289607|
|        UMP9|           1477190|
|    Bluezone|           1314890|
|       S1897|           1139445|
|     Mini 14|            821572|
|       Punch|            709619|
+------------+------------------+
only showing top 10 rows
```

Question 2. What is the most dangerous location in each PUBG map?

 # find distinct map names

select distinct map_name from kills where map_name is not null;

```
1 select distinct map_name from kills where map_name is not null;
```

Query History | Saved Queries | Results (3)

	map_name
1	
2	MIRAMAR
3	ERANGEL

 # calculate the total deaths in each sub-sections for map MIRAMAR

select victim_x, victim_y, count(match_id) as total_deaths from
(select match_id,
(Case
When victim_position_x>=0 and victim_position_x <100000 then '0.5'
When victim_position_x >= 100000 and victim_position_x <200000 then '1.5'
When victim_position_x >= 200000 and victim_position_x < 300000 then '2.5'
When victim_position_x >= 300000 and victim_position_x < 400000 then '3.5'
When victim_position_x >= 400000 and victim_position_x < 500000 then '4.5'
When victim_position_x >= 500000 and victim_position_x <600000 then '5.5'
When victim_position_x >= 600000 and victim_position_x < 700000 then '6.5'

```
When victim_position_x >= 700000 and victim_position_x <= 800000 then '7.5'
End) as victim_x,
(Case
When victim_position_y >=0 and victim_position_y <100000 then '0.5'
When victim_position_y >= 100000 and victim_position_y <200000 then '1.5'
When victim_position_y >= 200000 and victim_position_y < 300000 then '2.5'
When victim_position_y >= 300000 and victim_position_y < 400000 then '3.5'
When victim_position_y >= 400000 and victim_position_y < 500000 then '4.5'
When victim_position_y >= 500000 and victim_position_y <600000 then '5.5'
When victim_position_y >= 600000 and victim_position_y < 700000 then '6.5'
When victim_position_y >= 700000 and victim_position_y <= 800000 then '7.5'
End) as victim_y from kills where map_name='MIRAMAR') as victim_table
where victim_x is not null and victim_y is not null
group by victim_x,victim_y
order by total_deaths desc;
```

	victim_x	victim_y	total_deaths
1	3.5	4.5	833503
2	4.5	2.5	432558
3	4.5	5.5	310599
4	3.5	3.5	302414
5	3.5	2.5	267621
6	4.5	3.5	229572
7	0.5	0.5	226759
8	5.5	3.5	162674
9	4.5	4.5	145557
10	1.5	2.5	137012
11	2.5	5.5	123485

 # calculate the total deaths in each sub-sections for map ERANGEL

```
select victim_x, victim_y, count(match_id) as total_deaths from
(select match_id,
(Case
When victim_position_x>=0 and victim_position_x <100000 then '0.5'
When victim_position_x >= 100000 and victim_position_x <200000 then '1.5'
When victim_position_x >= 200000 and victim_position_x < 300000 then '2.5'
When victim_position_x >= 300000 and victim_position_x < 400000 then '3.5'
When victim_position_x >= 400000 and victim_position_x < 500000 then '4.5'
When victim_position_x >= 500000 and victim_position_x <600000 then '5.5'
When victim_position_x >= 600000 and victim_position_x < 700000 then '6.5'
When victim_position_x >= 700000 and victim_position_x <= 800000 then '7.5'
End) as victim_x,
(Case
When victim_position_y >=0 and victim_position_y <100000 then '0.5'
When victim_position_y >= 100000 and victim_position_y <200000 then '1.5'
When victim_position_y >= 200000 and victim_position_y < 300000 then '2.5'
When victim_position_y >= 300000 and victim_position_y < 400000 then '3.5'
When victim_position_y >= 400000 and victim_position_y < 500000 then '4.5'
When victim_position_y >= 500000 and victim_position_y <600000 then '5.5'
When victim_position_y >= 600000 and victim_position_y < 700000 then '6.5'
When victim_position_y >= 700000 and victim_position_y <= 800000 then '7.5'
End) as victim_y from kills where map_name='ERANGEL') as victim_table
where victim_x is not null and victim_y is not null
group by victim_x,victim_y
order by total_deaths desc;
```

	victim_x	victim_y	total_deaths
1	4.5	6.5	1843935
2	4.5	3.5	1823058
3	3.5	4.5	1519836
4	3.5	3.5	1320054
5	0.5	0.5	979776
6	6.5	3.5	861852
7	4.5	4.5	851617
8	5.5	3.5	844341
9	2.5	2.5	800442
10	3.5	5.5	774490
11	5.5	4.5	761679

Question 3. What is the best location to kill another player in each PUBG map?

 # calculate the total kills in each sub-sections for map MIRAMAR

```
select killer_x, killer_y, count(match_id) as total_kills from
(select match_id,
(Case
When killer_position_x>=0 and killer_position_x<100000 then '0.5'
When killer_position_x >= 100000 and killer_position_x <200000 then '1.5'
When killer_position_x >= 200000 and killer_position_x < 300000 then '2.5'
When killer_position_x >= 300000 and killer_position_x < 400000 then '3.5'
When killer_position_x >= 400000 and killer_position_x < 500000 then '4.5'
When killer_position_x >= 500000 and killer_position_x <600000 then '5.5'
When killer_position_x >= 600000 and killer_position_x < 700000 then '6.5'
When killer_position_x >= 700000 and killer_position_x <= 800000 then '7.5'
End) as killer_x,
(Case
When killer_position_y >=0 and killer_position_y<100000 then '0.5'
When killer_position_y >= 100000 and killer_position_y <200000 then '1.5'
When killer_position_y >= 200000 and killer_position_y < 300000 then '2.5'
When killer_position_y >= 300000 and killer_position_y < 400000 then '3.5'
When killer_position_y >= 400000 and killer_position_y < 500000 then '4.5'
When killer_position_y >= 500000 and killer_position_y <600000 then '5.5'
When killer_position_y >= 600000 and killer_position_y < 700000 then '6.5'
When killer_position_y >= 700000 and killer_position_y <= 800000 then '7.5'
End) as killer_y from kills where map_name='MIRAMAR') as killer_table
where killer_x is not null and killer_y is not null
group by killer_x,killer_y
order by total_kills desc;
```

	killer_x	killer_y	total_kills
1	3.5	4.5	820978
2	4.5	2.5	425683
3	4.5	5.5	299917
4	3.5	3.5	291346
5	3.5	2.5	260256
6	4.5	3.5	214000
7	5.5	3.5	156069
8	1.5	2.5	137067
9	4.5	4.5	134477
10	2.5	5.5	116651
11	6.5	4.5	106065
12	2.5	4.5	101700

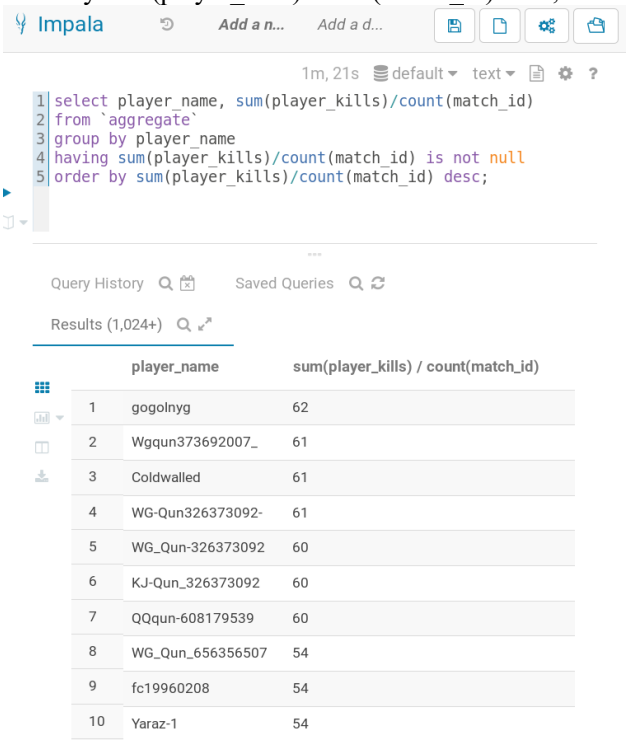
 # calculate the total kills in each sub-sections for map ERANGEL

```
select killer_x, killer_y, count(match_id) as total_kills from
(select match_id,
(Case
When killer_position_x>=0 and killer_position_x<100000 then '0.5'
When killer_position_x >= 100000 and killer_position_x <200000 then '1.5'
When killer_position_x >= 200000 and killer_position_x < 300000 then '2.5'
When killer_position_x >= 300000 and killer_position_x < 400000 then '3.5'
When killer_position_x >= 400000 and killer_position_x < 500000 then '4.5'
When killer_position_x >= 500000 and killer_position_x <600000 then '5.5'
When killer_position_x >= 600000 and killer_position_x < 700000 then '6.5'
When killer_position_x >= 700000 and killer_position_x <= 800000 then '7.5'
End) as killer_x,
(Case
When killer_position_y >=0 and killer_position_y<100000 then '0.5'
When killer_position_y >= 100000 and killer_position_y <200000 then '1.5'
When killer_position_y >= 200000 and killer_position_y < 300000 then '2.5'
When killer_position_y >= 300000 and killer_position_y < 400000 then '3.5'
When killer_position_y >= 400000 and killer_position_y < 500000 then '4.5'
When killer_position_y >= 500000 and killer_position_y <600000 then '5.5'
When killer_position_y >= 600000 and killer_position_y < 700000 then '6.5'
When killer_position_y >= 700000 and killer_position_y <= 800000 then '7.5'
End) as killer_y from kills where map_name='ERANGEL') as killer_table
where killer_x is not null and killer_y is not null
group by killer_x,killer_y
order by total_kills desc;
```

	killer_x	killer_y	total_kills
1	4.5	6.5	1849667
2	4.5	3.5	1787896
3	3.5	4.5	1469163
4	3.5	3.5	1276814
5	6.5	3.5	854877
6	5.5	3.5	812687
7	4.5	4.5	801942
8	2.5	2.5	772529
9	3.5	5.5	728688
10	5.5	4.5	722926
11	4.5	2.5	711075
12	1.5	2.5	709397

Question 4. Who is the best player in this game? Which player has the highest average kills per match?

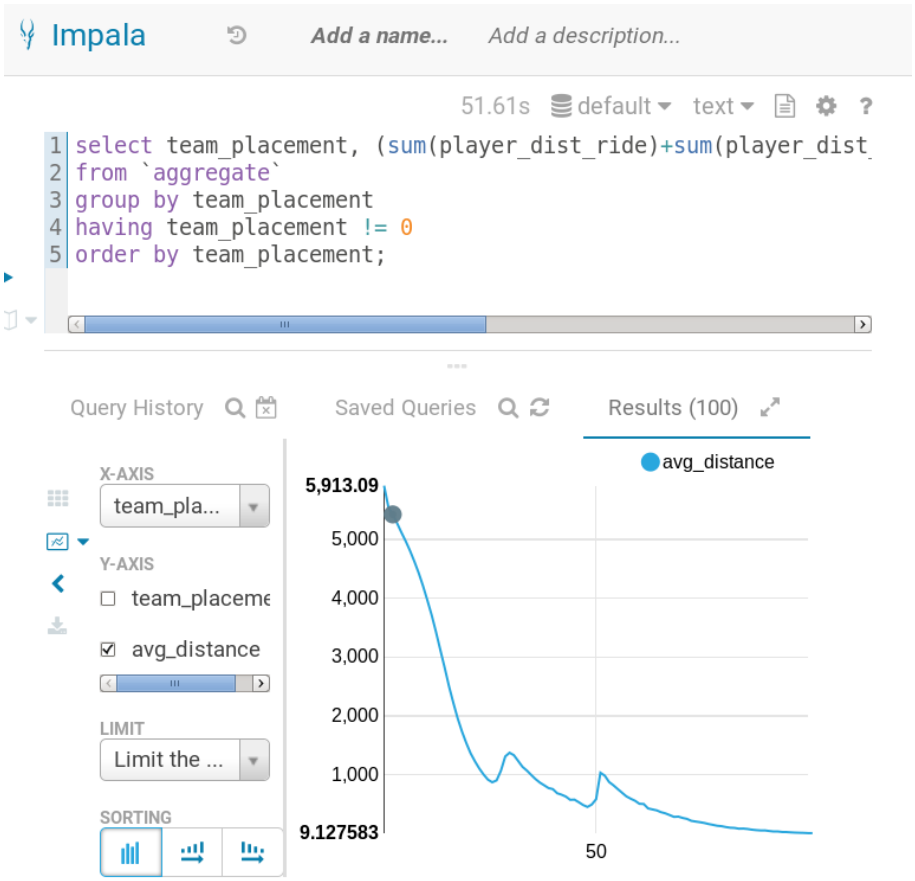
select player_name, sum(player_kills)/count(match_id)
from `aggregate`
group by player_name
having sum(player_kills)/count(match_id) is not null
order by sum(player_kills)/count(match_id) desc;



	player_name	sum(player_kills) / count(match_id)
1	gogolnyg	62
2	Wgqun373692007_	61
3	Coldwalled	61
4	WG-Qun326373092-	61
5	WG_Qun-326373092	60
6	KJ-Qun_326373092	60
7	QQqun-608179539	60
8	WG_Qun_656356507	54
9	fc19960208	54
10	Yaraz-1	54

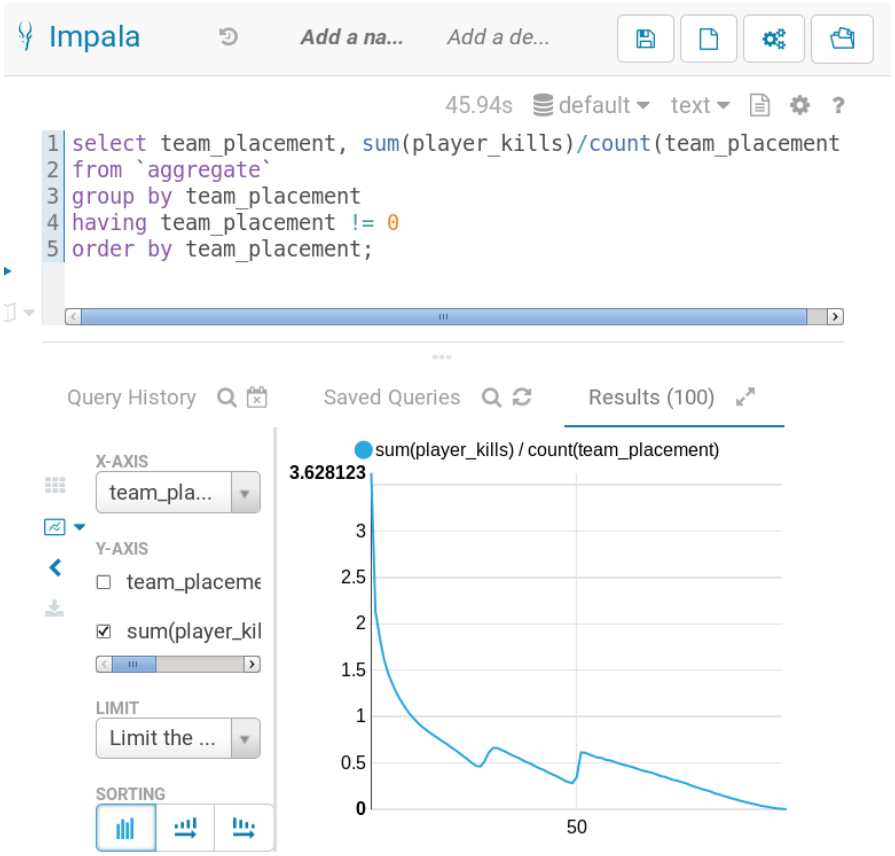
Question 5. Will moving more in the games leads to a better rank?

```
select team_placement, (sum(player_dist_ride)+sum(player_dist_walk))/
count(team_placement) as avg_distance
from `aggregate`
group by team_placement
having team_placement != 0
order by team_placement;
```



Question 6. Will a higher number of total kills of a team leads to a better rank for that team?

```
select team_placement, sum(player_kills)/count(team_placement)
from `aggregate`
group by team_placement
having team_placement != 0
order by team_placement;
```



Question 7. Which player has participated in most games?

 # find the top 10 players with the highest game frequency

We used pyspark here.

```
rdd_1=sc.textFile('aggregate.csv')
header=rdd_1.first()
rdd_data=rdd_1.filter(lambda x:x!=header)
rdd_name=rdd_data.map(lambda x:x.split(',')).map(lambda x: (x[12].encode('ascii'),1)).filter(lambda x: (x[0]!=''))
rdd_result=rdd_name.reduceByKey(lambda x,y:x+y).map(lambda x:(x[1],x[0])).sortByKey(False)
rdd_top_result=rdd_result.take(10)
df=sqlContext.createDataFrame(rdd_top_result,['Frequency','Player'])
df.show()
```

```
+---------+---------------+
|Frequency|         Player|
+---------+---------------+
|      688|        VanThang|
|      577|   Matthew_wang|
|      570|      coolcarey|
|      567|         JZalan|
|      542|      hzxiaobin|
|      502|      Slh_Bunny|
|      448|huangshanglaiye|
|      440|      jonelycai|
|      440|    SDFSADFASDF|
|      434|        AKA8881|
+---------+---------------+
```

 # find the most frequent words in player_name and create WordCloud with Python

```
import pandas as pd
df=pd.read_csv("aggregate.csv")
from os import path
from PIL import Image
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
import matplotlib.pyplot as plt
def name(df):
    names=''
    for index,row in df.iterrows():
        name=row['player_name']
        if name != '':
            names+=str(name).lower()
            names+=' '
    return names
names=name(df)
wordcloud = WordCloud().generate(names)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
wordcloud.to_file('Desktop: pic.png')
```

8. Which player has highest average DBNO per match? Which player has the highest average assist per match?

 # find the top 10 players with the highest average saved times

SELECT player_name, avg(player_dbno) as avg_saved
from aggregate group by player_name having player_name != ''
order by avg_saved desc limit 10;

	player_name	avg_saved
1	HabDichErwischt	51
2	gogolnyg	47
3	Yaraz-1	41
4	fc19960208	40
5	mark-saltedfish	40
6	SKsuMt	39
7	WG_Qun_656356507	38
8	ZhEzH1	37
9	RichRichRichWang	37
10	QXWly0420	36

 # find the top 10 players with the highest average saved times

SELECT player_name, avg(player_assists) as avg_save
from aggregate group by player_name having player_name != ''
order by avg_save desc limit 10;

	player_name	avg_save
1	WangJiaHao	10
2	CGdakang	10
3	Advanst	10
4	jingcocome	9
5	whwrkswl	9
6	Guopen	9
7	LazyCatOVO	9
8	PG_Qiao	9
9	Ash-Chen	9
10	xiaoqiao1996	9