Data Science Summer Project

IIT Men’s Basketball Team

Chenjie Li
Dilruba Palabıyık

2018 Summer

Table Of Contents:

1.Introduction

This work is a further development of Larry Layne and Denis Bajic's work in 2016 summer, who did a comprehensive analysis not only on the single players but also for the team as a whole.

Compared with two years ago, we have a more detailed and more normalized source of data – Synergy Sports website which have a comprehensive archive of last years' stats of our team and every player's performance.

But, soon after we started, we found out that it was a problem to gather the data from Synergy website, we tried writing scrapers to directly get data from the website but couldn't do it. After discussion with professor Shlomo, we decided to first copy and paste as ".txt" from Synergy and do some data cleaning, put them in our self- designed Database.

After several weeks of discussions and modifications, we finally successfully set the system up in our PostgresSQL server.

In the next one and a half month, we focused on the following stats and aspects for our analysis:

PER(Player Efficiency Rating), Strength and Weakness (For players and teams), Line up analysis based on Play-By-Play Data, Plus Minus(for single player and the line up)

Our work's explanation and results are available in GitHub:

https://github.com/JayLi2018/School-Basketball-Team-Analysis

2.Get data

2.1 Synergy website

Unlike 2016 summer, during which Larry et al. can only work with our school website, we can now refer to this sports technology website called Synergy: https://corp.synergysportstech.com/. Using coach Kelly's website, we would be able to log in and explore all the data that are currently available.

At first, we throw ourselves into developing a web scraper which can help facilitate the data gathering process. But after about two weeks, we were still stuck at getting over the log in page, and since most parts of the website are developed using Javascript, It made our work a lot harder.

So, after discussion with Professor, we decided to change our strategy: build up a database using the data about our team available on Synergy website. When it comes to how to get data, I suggested we directly copy paste the page we need from the website and then do some data cleaning to generate a nicely formatted .csv file so that we can put them to our database.

After exploring the Synergy and discussion within the team, we decided to get data from "Game"," Team", "Player" tabs.

"Game" includes the details of every game's stats of both teams.

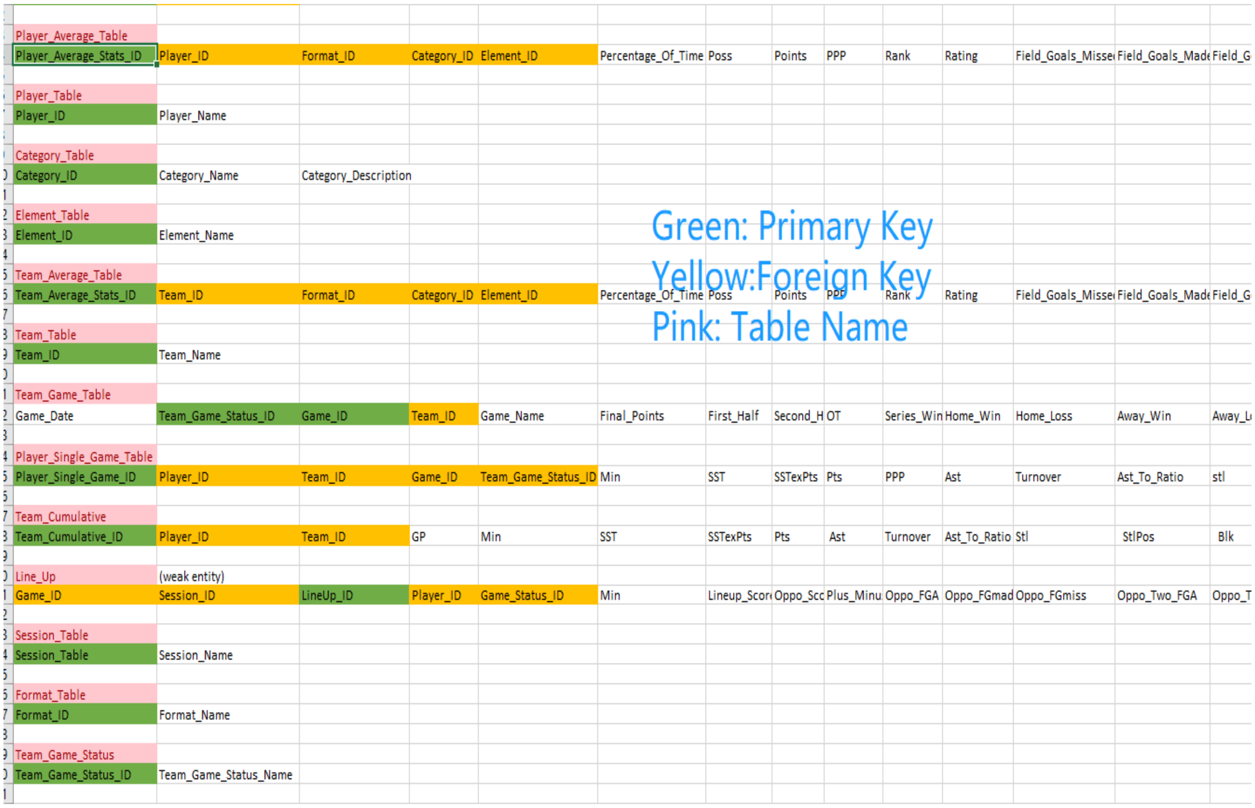
"Team" consists of the season overall average performance of the team.

"Player" covers the individual player's overall average performance of the team.

2.2 Database Design

We decide to design the database schema based on the way the data presented on Synergy, but since Synergy does not show how those data are connected with each other, we need to design our own way to be able to join those data in the future to help us run some interesting queries.

Here is our database schema (Please check GitHub if you need details):



Player_Average_Table													
Player_Average_Stats_ID	Player_ID	Format_ID	Category_ID	Element_ID	Percentage_Of_Time	Poss	Points	PPP	Rank	Rating	Field_Goals_Missei	Field_Goals_Made	Field_G
Player_Table													
Player_ID	Player_Name												
Category_Table													
Category_ID	Category_Name	Category_Description											
Element_Table													
Element_ID	Element_Name												
Team_Average_Table													
Team_Average_Stats_ID	Team_ID	Format_ID	Category_ID	Element_ID	Percentage_Of_Time	Poss	Points	PPP	Rank	Rating	Field_Goals_Missei	Field_Goals_Made	Field_G
Team_Table													
Team_ID	Team_Name												
Team_Game_Table													
Game_Date	Team_Game_Status_ID	Game_ID	Team_ID	Game_Name	Final_Points	First_Half	Second_H	OT	Series_Win	Home_Win	Home_Loss	Away_Win	Away_L
Player_Single_Game_Table													
Player_Single_Game_ID	Player_ID	Team_ID	Game_ID	Team_Game_Status_ID	Min	SST	SSTexPts	Pts	PPP	Ast	Turnover	Ast_To_Ratio	stl
Team_Cumulative													
Team_Cumulative_ID	Player_ID	Team_ID	GP	Min	SST	SSTexPts	Pts	Ast	Turnover	Ast_To_Ratio	Stl	StlPos	Blk
Line_Up	(weak entity)												
Game_ID	Session_ID	LineUp_ID	Player_ID	Game_Status_ID	Min	Lineup_Scor	Oppo_Scc	Plus_Minu	Oppo_FGA	Oppo_FGmad	Oppo_FGmiss	Oppo_Two_FGA	Oppo_T
Session_Table													
Session_Table	Session_Name												
Format_Table													
Format_ID	Format_Name												
Team_Game_Status													
Team_Game_Status_ID	Team_Game_Status_Name												

Green: Primary Key

Yellow:Foreign Key

Pink: Table Name

Player_Average_Table:

Corresponding to Synergy's "Player", it has all the info of every player in the "league"(Teams which IIT played against last season, will explain further later.)

Team_Average_Table:

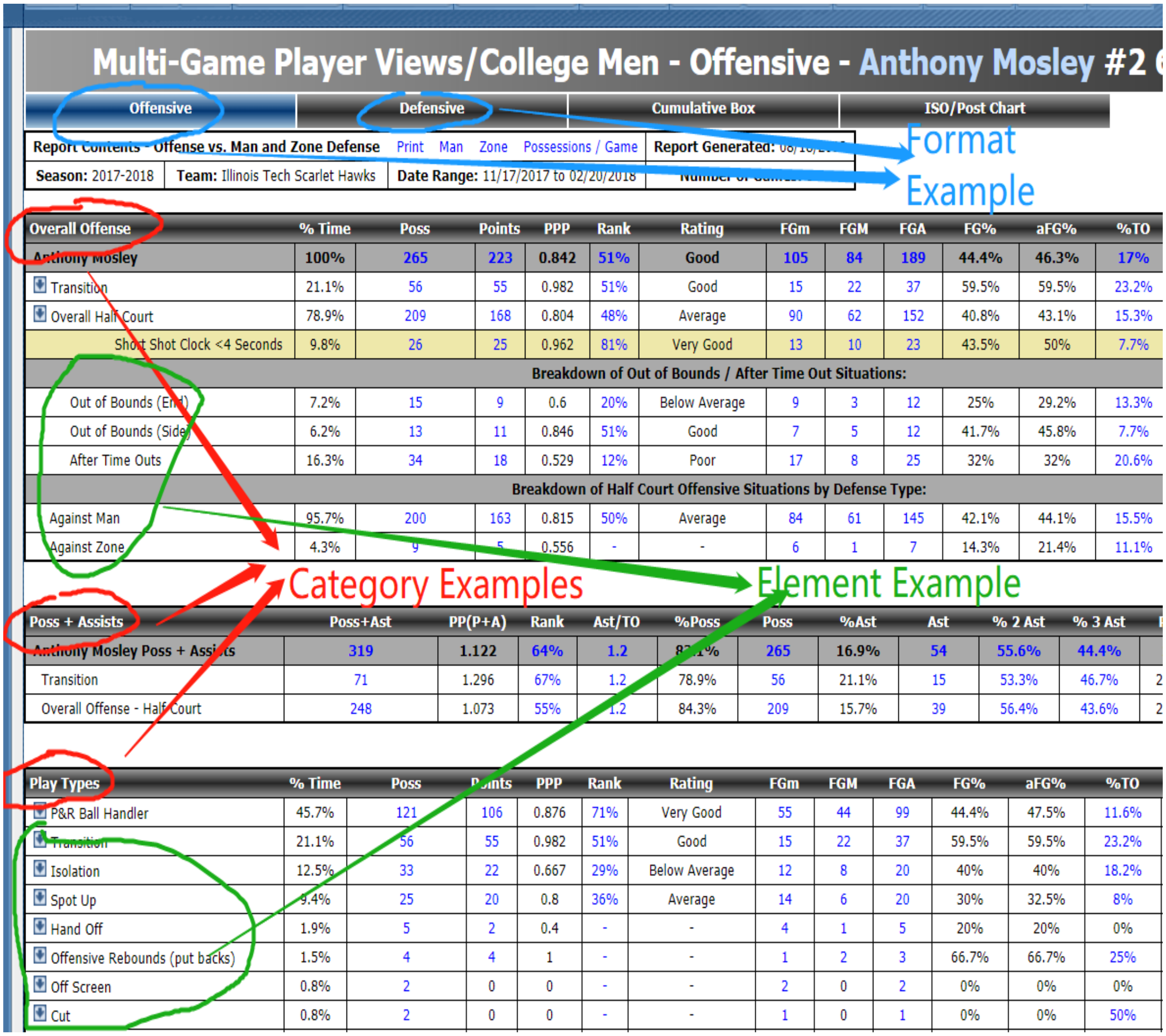
Similar to the Player_Average_Table

Player_Table:

Just player name and player_id. Note: player's First Name and Last Name are connected using "_" since we were afraid that we might run into some problems if there were some spaces(you can't make sure you will only leave one space)

Category_Table, Format_Table, Element_Table:

We designed this way to solve the "Two dimensional Table" Problems in "Team Average" and "Player Average" table in Synergy



Multi-Game Player Views/College Men - Offensive - Anthony Mosley #2

Report Contents - Offense vs. Man and Zone Defense | Print | Man | Zone | Possessions / Game | Report Generated: 08/10/20

Season: 2017-2018 | Team: Illinois Tech Scarlet Hawks | Date Range: 11/17/2017 to 02/20/2018 | Number of Games:

Overall Offense	% Time	Poss	Points	PPP	Rank	Rating	FGm	FGM	FGA	FG%	aFG%	%TO
Anthony Mosley	100%	265	223	0.842	51%	Good	105	84	189	44.4%	46.3%	17%
Transition	21.1%	56	55	0.982	51%	Good	15	22	37	59.5%	59.5%	23.2%
Overall Half Court	78.9%	209	168	0.804	48%	Average	90	62	152	40.8%	43.1%	15.3%
Short Shot Clock <4 Seconds	9.8%	26	25	0.962	81%	Very Good	13	10	23	43.5%	50%	7.7%
Breakdown of Out of Bounds / After Time Out Situations:												
Out of Bounds (End)	7.2%	15	9	0.6	20%	Below Average	9	3	12	25%	29.2%	13.3%
Out of Bounds (Side)	6.2%	13	11	0.846	51%	Good	7	5	12	41.7%	45.8%	7.7%
After Time Outs	16.3%	34	18	0.529	12%	Poor	17	8	25	32%	32%	20.6%
Breakdown of Half Court Offensive Situations by Defense Type:												
Against Man	95.7%	200	163	0.815	50%	Average	84	61	145	42.1%	44.1%	15.5%
Against Zone	4.3%	9	5	0.556	-	-	6	1	7	14.3%	21.4%	11.1%

Poss + Assists	Poss+Ast	PP(P+A)	Rank	Ast/TO	%Poss	Poss	%Ast	Ast	% 2 Ast	% 3 Ast	P
Anthony Mosley Poss + Assists	319	1.122	64%	1.2	83.1%	265	16.9%	54	55.6%	44.4%	
Transition	71	1.296	67%	1.2	78.9%	56	21.1%	15	53.3%	46.7%	2
Overall Offense - Half Court	248	1.073	55%	1.2	84.3%	209	15.7%	39	56.4%	43.6%	2

Play Types	% Time	Poss	Points	PPP	Rank	Rating	FGm	FGM	FGA	FG%	aFG%	%TO
P&R Ball Handler	45.7%	121	106	0.876	71%	Very Good	55	44	99	44.4%	47.5%	11.6%
Transition	21.1%	56	55	0.982	51%	Good	15	22	37	59.5%	59.5%	23.2%
Isolation	12.5%	33	22	0.667	29%	Below Average	12	8	20	40%	40%	18.2%
Spot Up	9.4%	25	20	0.8	36%	Average	14	6	20	30%	32.5%	8%
Hand Off	1.9%	5	2	0.4	-	-	4	1	5	20%	20%	0%
Offensive Rebounds (put backs)	1.5%	4	4	1	-	-	1	2	3	66.7%	66.7%	25%
Off Screen	0.8%	2	0	0	-	-	2	0	2	0%	0%	0%
Cut	0.8%	2	0	0	-	-	1	0	1	0%	0%	50%

This is what we call “Two Dimensional Table”, we need to split this up since we can’t put the exact same format in our database. Instead, we come up with the idea of “Category, Format, Element” combination to identify a single row.

We defined “Offensive”, ”Defensive” as format; Define” Overall offense”,”Play Types”(we didn’t include “poss+assists” because it was the only one that has an unique structure) as a part of category. Define “Anthony Mosley”(we call this “Player” in our database since every player has their own name),”Transition”,”Out Of Bounds” as a part of element.

So for a single row in Player_Average and Team_average, the structure looks like this:

Format_ID	**Category_ID**	**Element_ID**	**%Time**	**Poss**	**Points**	**PPP**	**…..**	**….**	**…..**

Team_Table:

In team_table, we put in the teams that IIT played agains last season, and treat them as a "League"(we need a league to calculate the PER later).

Team_Game_Table

We created two separate rows for the two teams in a game, and created team_game_status table to differentiate one row from another (one game has to have a "home_team":id = 1,and a "guest team": id= 2)

Player_Single_Game_Table

We created a table for the individual stats for each IIT player in each game.

Team_Cumulative_Tabke:

Team_Cumulative table is corresponding to the "Cumulative Box" in Synergy and it's a summary of each player in cumulative way.

Lineupinfo_Table:

This we designed it as a "weak entity" and (identifying relationship with game table, session table, player table)

Session_Table:

We define session table as "first half" or "second" half.

For more details regarding SQL for table creations and formatted CSVs for each table, python data cleaning codes, please visit my GitHub.

3.Data Exploration and Analysis Methodology

In this section we present our analysis part, which mainly focuses on PER (Player Efficiency Rating), Plus/Minus, Lineup Analysis, Strength and weakness.

3.1 Strength and Weakness (Players&Teams)

3.1.1. Introduction

Besides the PER, we were also trying to think of a way to find a player's Strength and Weakness Offensively. And I noticed that there is an interesting metric available in database called "PPP" (Points Per Possession).

Here is the definition of PPP:

An efficiency rating that calculates how many points on average a player or team is scoring in a specific play type or category, per possession. A sample PPP situation is if a team or player has three consecutive possessions:

1) A turnover

2) Gets fouled in the act of shooting and makes both the basket and the free throw

3) Makes a 3-point shot

That result is the team or player scoring 6 points on 3 possessions. The Points Per Possession for this situation would equal 2.0.

After that, I also noticed that there are some kinds of "Play Types" available in database

Here is the list of the play-type kinds and their corresponding definitions:

1)Spot-Ups:

When the possession ending event is a catch and shoot or catch and drive play. Spot-up players typically have a defender closing out on them during a defensive rotation and the spot-up player has the option of shooting the ball before the defender gets to them, letting the defender fly by and then shooting or driving, or they can use the defender's forward motion against them and drive to the basket. This situation also occurs during an off screen play. Therefore, if there was no screen, we then log the play as a spot-up. Other times spot-up players seem to simply get open, perhaps because their defender sagged towards the basket or simply went to sleep. In this case, the offensive player still has the option to shoot or drive.

2)Transition:

When the possession ending event comes before the defense sets following a possession change and a transition from one end of the court to the other. Unlike some other fast-break definitions, here "Transition" allows for a considerable period of time to pass before the possession ends – for example the center can trail down the floor after all the other transition options have been explored and if his defender is back guarding the paint and so is open to shoot a perimeter shot, then it is still logged as a transition. It can also include press breaks as the ball is quickly moved from the back court to the front court.

3) P&R Ball Handler

A screen is set on the ball handler's defender out on the perimeter. The offensive player can use the screen or go away from it and as long as the play yields a possession ending event, it is tagged as a pick and roll. It is important to note that pick and rolls frequently cause the defense to rotate resulting in ball movement with spot-ups, cuts or pick and pop/pick and rolls. These situations are defined in the Synergy system according to the ultimate play types that occur, but are also credited back to the ball handler as long as the defense never gets "whole" (in position to defend all 5 players).

4) Post-Up

When an offensive player receives the ball with their back to the basket and is less than 15' from the rim when the possession ending event occurs. Post players can attack over either shoulder, they can back their player down off the dribble, they can turn and face up their defender and there are a variety of moves they can make in all of these situations, yet they are all logged as post-ups. We also categorize flash posts in the paint as post-ups. Post-ups are handled like pick and rolls in the sense that if a double team occurs or, less extreme, if the defense commits and a rotation ensues, the post player is given credit for the ultimate outcome (a cut or a spot-up) as long as the defense never gets "whole" (in position to defend all 5 players).

5) Cuts

An interior play where the finisher catches a pass while moving toward, parallel to or slightly away from the basket. This will include back screen and flash cuts as well as times when the player is left open near the basket when his defender leaves him to help with a penetrating offensive player such as in a draw and kick situation.

6)Isolation

When the possession ending event is created during a "one on one" matchup. The defender needs to be set and have all of his defensive options at the initiation of the play. If the defender is closing out on an offensive player such as in a spot up situation or an off screen situation and cannot get "whole" (on balance and set to defend), then it is not an Isolation play.

7)P&R Man

When a screen is set for the ball handler, and the screen setter then receives the ball for a possession ending event. This action can include:

- Pick and rolls
  - The ball handler comes off the pick and the roll man rolls toward the basket before receiving the ball.
- Pick and pops
  - The ball handler comes off the pick and the roll man pops out away from the basket to receive the ball.
- Slips Pick
  - The roll man slips the pick prior to the ball handler using him as a screen, then receives the ball.

7)Off-Screens

Identifies players coming off of screens (typically downs screens) going away from the basket toward the perimeter. This includes curl, fades, and coming off straight. Selected options in this regard are typically dictated by the path the player's defender takes. For example, when a defender chases, the offensive player may curl to the middle to create space. Or when the defender cuts inside the screen, the offensive player may fade towards the corner. Flare screens are also "off screen" but differ in that the screen is set on the perimeter and the offensive player uses the screen to get separation from their defender using a path that is somewhat parallel to the basket.

8) Offensive Rebounds – Put Backs

When the rebounder attempts to score before passing the ball or establishing themselves in another play type.

9) Handoffs

The screen setter starts with the ball and hands the ball to a player cutting close by. This enables the player handing the ball off to effectively screen off a defender creating space for the player receiving the ball.

10) Miscellaneous

When the action doesn't fit any of the other play types. This includes, but is not limited to, last second full court shots, fouls in the backcourt, or errant passes not out of a different play type, etc.

3.1.2.Evaluation method

By comparing the specific play type’s PPP with the overall average PPP, we may clearly see in which kind of play a is more “efficiency” for a player, so that we may give suggestions on he should use this kind of play more in the future because he is “good at it”, or try to reduce the amount of play of some kind because he is “not that good” when he choose to attack the basket in this way (Or work harder on those aspects if he insists).

I used the same logic to evaluate the teams, too.

3.1.3. Step-By-Step Operations

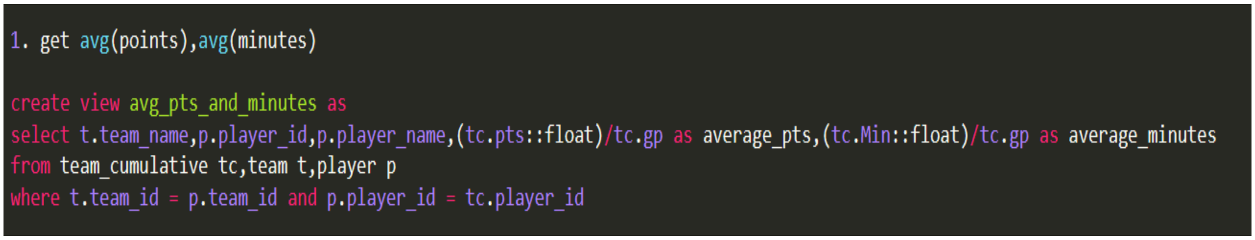
Players

- ***“Filter” for Players***

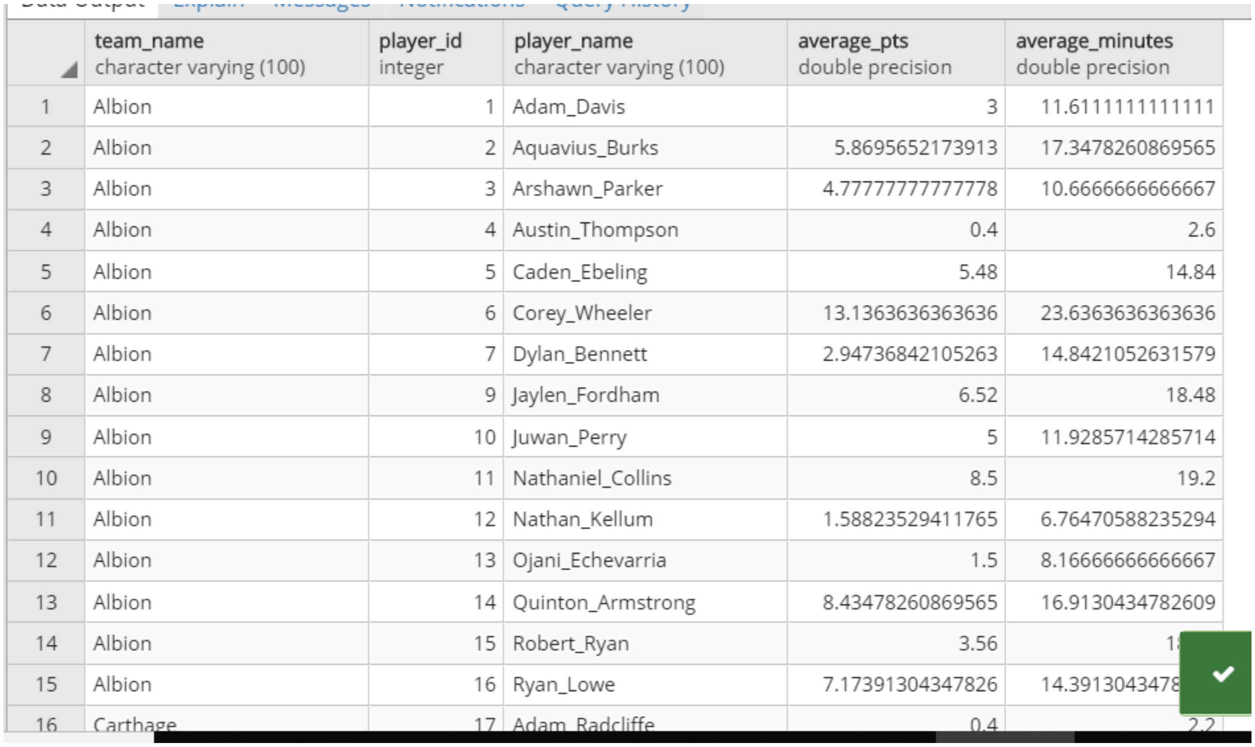
My goal is letting coach to be able to filter out players based on some key aspects such as by “Minutes Played”,” Points Per Game”,” School Name”, ” Overall Average PPP”, ”Play Type PPP”, ”Element Name”, ”Format Name” and so forth.

- ***Steps for players:***

1) get players’ average points and average minutes:

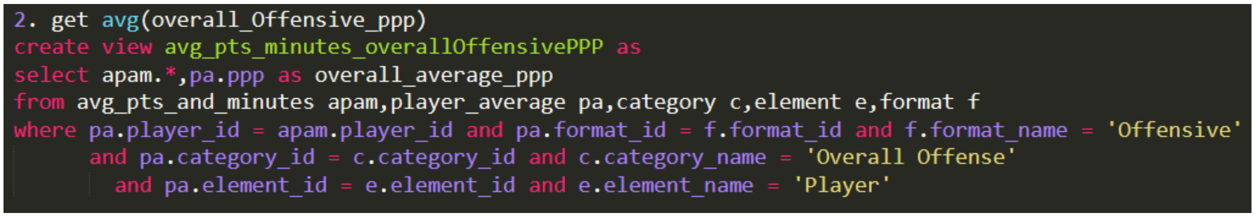


```
1. get avg(points),avg(minutes)

create view avg_pts_and_minutes as
select t.team_name,p.player_id,p.player_name,(tc.pts::float)/tc.gp as average_pts,(tc.Min::float)/tc.gp as average_minutes
from team_cumulative tc,team t,player p
where t.team_id = p.team_id and p.player_id = tc.player_id
```

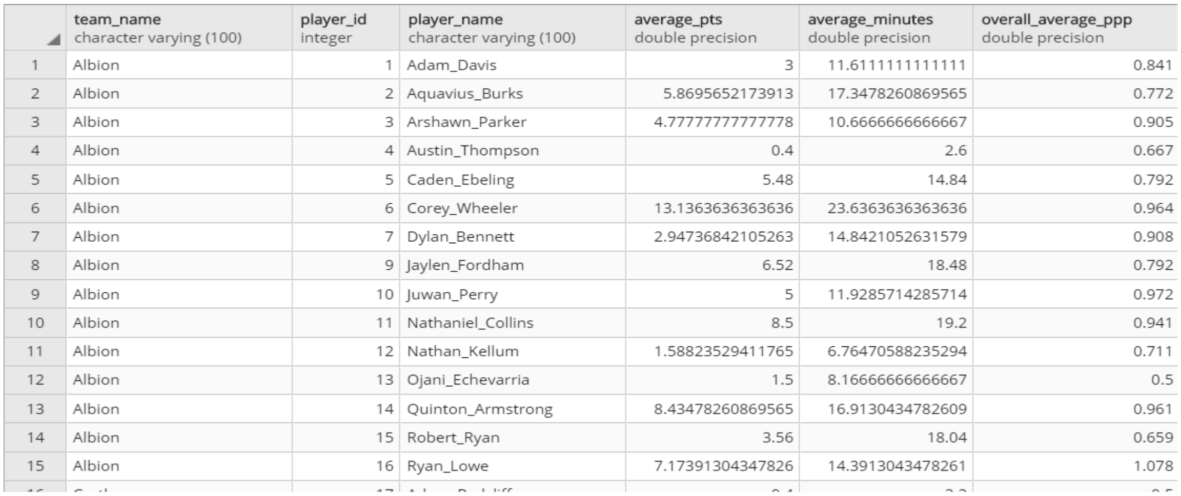


	team_name character varying (100)	player_id integer	player_name character varying (100)	average_pts double precision	average_minutes double precision
1	Albion	1	Adam_Davis	3	11.6111111111111
2	Albion	2	Aquavius_Burks	5.8695652173913	17.3478260869565
3	Albion	3	Arshawn_Parker	4.77777777777778	10.6666666666667
4	Albion	4	Austin_Thompson	0.4	2.6
5	Albion	5	Caden_Ebeling	5.48	14.84
6	Albion	6	Corey_Wheeler	13.1363636363636	23.6363636363636
7	Albion	7	Dylan_Bennett	2.94736842105263	14.8421052631579
8	Albion	9	Jaylen_Fordham	6.52	18.48
9	Albion	10	Juwan_Perry	5	11.9285714285714
10	Albion	11	Nathaniel_Collins	8.5	19.2
11	Albion	12	Nathan_Kellum	1.58823529411765	6.76470588235294
12	Albion	13	Ojani_Echevarria	1.5	8.16666666666667
13	Albion	14	Quinton_Armstrong	8.43478260869565	16.9130434782609
14	Albion	15	Robert_Ryan	3.56	1
15	Albion	16	Ryan_Lowe	7.17391304347826	14.3913043478
16	Carthage	17	Adam_Radcliffe	0.4	2.2

2)get the "overall offensive PPP" of each player

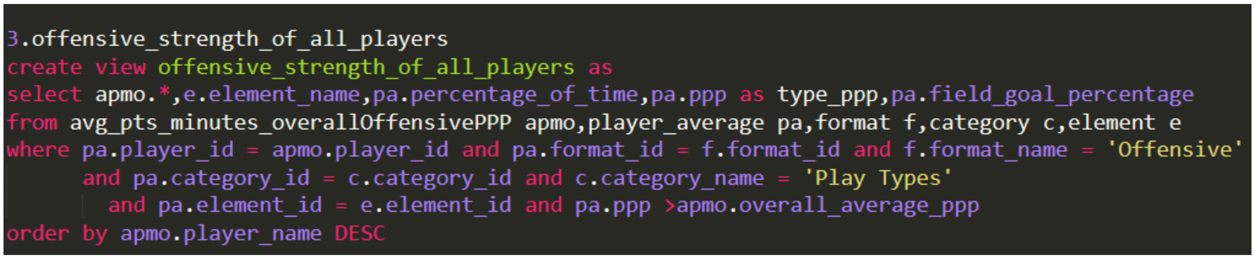


```
2. get avg(overall_Offensive_ppp)
create view avg_pts_minutes_overallOffensivePPP as
select apam.*,pa.ppp as overall_average_ppp
from avg_pts_and_minutes apam,player_average pa,category c,element e,format f
where pa.player_id = apam.player_id and pa.format_id = f.format_id and f.format_name = 'Offensive'
      and pa.category_id = c.category_id and c.category_name = 'Overall Offense'
       and pa.element_id = e.element_id and e.element_name = 'Player'
```

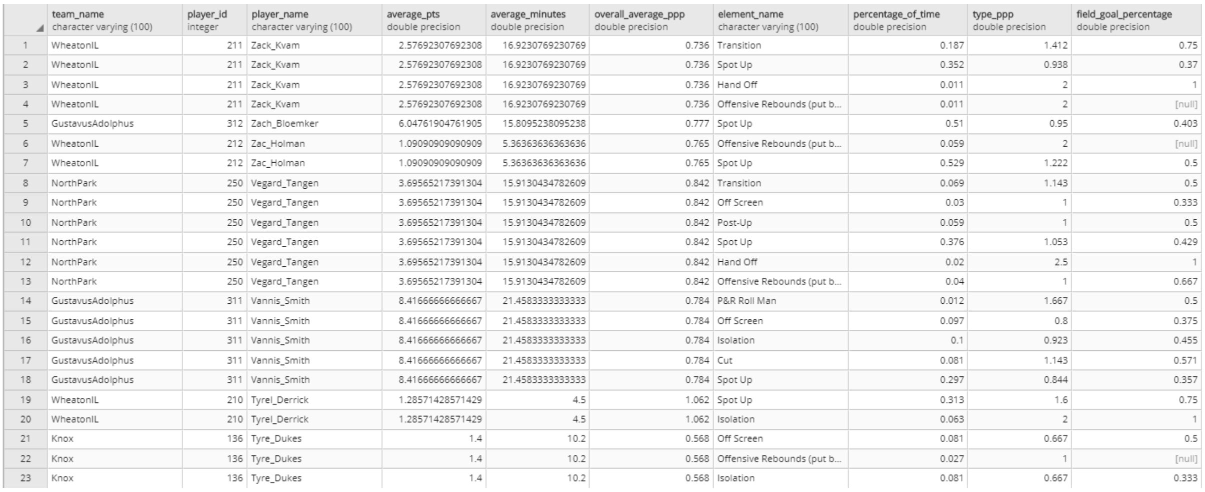


	team_name character varying (100)	player_id integer	player_name character varying (100)	average_pts double precision	average_minutes double precision	overall_average_ppp double precision
1	Albion	1	Adam_Davis	3	11.6111111111111	0.841
2	Albion	2	Aquavius_Burks	5.8695652173913	17.3478260869565	0.772
3	Albion	3	Arshawn_Parker	4.77777777777778	10.6666666666667	0.905
4	Albion	4	Austin_Thompson	0.4	2.6	0.667
5	Albion	5	Caden_Ebeling	5.48	14.84	0.792
6	Albion	6	Corey_Wheeler	13.1363636363636	23.6363636363636	0.964
7	Albion	7	Dylan_Bennett	2.94736842105263	14.8421052631579	0.908
8	Albion	9	Jaylen_Fordham	6.52	18.48	0.792
9	Albion	10	Juwan_Perry	5	11.9285714285714	0.972
10	Albion	11	Nathaniel_Collins	8.5	19.2	0.941
11	Albion	12	Nathan_Kellum	1.58823529411765	6.76470588235294	0.711
12	Albion	13	Ojani_Echevarria	1.5	8.16666666666667	0.5
13	Albion	14	Quinton_Armstrong	8.43478260869565	16.9130434782609	0.961
14	Albion	15	Robert_Ryan	3.56	18.04	0.659
15	Albion	16	Ryan_Lowe	7.17391304347826	14.3913043478261	1.078
16	Carthage	17	Adam_Radcliffe	0.4	2.2	0.5

3)get the "offensive strength" of players.

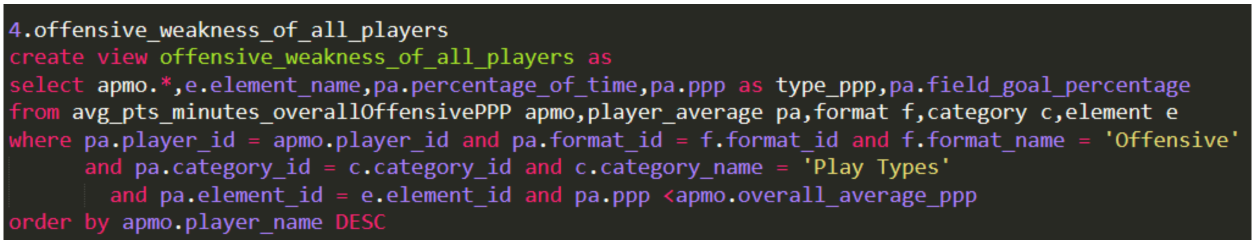


```
3.offensive_strength_of_all_players
create view offensive_strength_of_all_players as
select apmo.*,e.element_name,pa.percentage_of_time,pa.ppp as type_ppp,pa.field_goal_percentage
from avg_pts_minutes_overallOffensivePPP apmo,player_average pa,format f,category c,element e
where pa.player_id = apmo.player_id and pa.format_id = f.format_id and f.format_name = 'Offensive'
      and pa.category_id = c.category_id and c.category_name = 'Play Types'
        and pa.element_id = e.element_id and pa.ppp >apmo.overall_average_ppp
order by apmo.player_name DESC
```

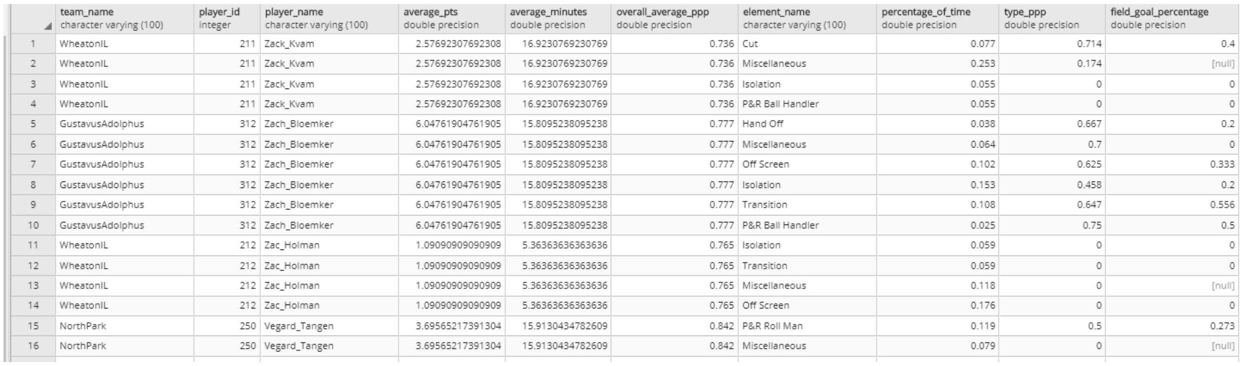


	team_name character varying (100)	player_id integer	player_name character varying (100)	average_pts double precision	average_minutes double precision	overall_average_ppp double precision	element_name character varying (100)	percentage_of_time double precision	type_ppp double precision	field_goal_percentage double precision
1	WheatonIL	211	Zack_Kvam	2.57692307692308	16.9230769230769	0.736	Transition	0.187	1.412	0.75
2	WheatonIL	211	Zack_Kvam	2.57692307692308	16.9230769230769	0.736	Spot Up	0.352	0.938	0.37
3	WheatonIL	211	Zack_Kvam	2.57692307692308	16.9230769230769	0.736	Hand Off	0.011	2	1
4	WheatonIL	211	Zack_Kvam	2.57692307692308	16.9230769230769	0.736	Offensive Rebounds (put b...	0.011	2	[null]
5	GustavusAdolphus	312	Zach_Bloemker	6.04761904761905	15.8095238095238	0.777	Spot Up	0.51	0.95	0.403
6	WheatonIL	212	Zac_Holman	1.09090909090909	5.36363636363636	0.765	Offensive Rebounds (put b...	0.059	2	[null]
7	WheatonIL	212	Zac_Holman	1.09090909090909	5.36363636363636	0.765	Spot Up	0.529	1.222	0.5
8	NorthPark	250	Vegard_Tangen	3.69565217391304	15.9130434782609	0.842	Transition	0.069	1.143	0.5
9	NorthPark	250	Vegard_Tangen	3.69565217391304	15.9130434782609	0.842	Off Screen	0.03	1	0.333
10	NorthPark	250	Vegard_Tangen	3.69565217391304	15.9130434782609	0.842	Post-Up	0.059	1	0.5
11	NorthPark	250	Vegard_Tangen	3.69565217391304	15.9130434782609	0.842	Spot Up	0.376	1.053	0.429
12	NorthPark	250	Vegard_Tangen	3.69565217391304	15.9130434782609	0.842	Hand Off	0.02	2.5	1
13	NorthPark	250	Vegard_Tangen	3.69565217391304	15.9130434782609	0.842	Offensive Rebounds (put b...	0.04	1	0.667
14	GustavusAdolphus	311	Vannis_Smith	8.41666666666667	21.4583333333333	0.784	P&R Roll Man	0.012	1.667	0.5
15	GustavusAdolphus	311	Vannis_Smith	8.41666666666667	21.4583333333333	0.784	Off Screen	0.097	0.8	0.375
16	GustavusAdolphus	311	Vannis_Smith	8.41666666666667	21.4583333333333	0.784	Isolation	0.1	0.923	0.455
17	GustavusAdolphus	311	Vannis_Smith	8.41666666666667	21.4583333333333	0.784	Cut	0.081	1.143	0.571
18	GustavusAdolphus	311	Vannis_Smith	8.41666666666667	21.4583333333333	0.784	Spot Up	0.297	0.844	0.357
19	WheatonIL	210	Tyrel_Derrick	1.28571428571429	4.5	1.062	Spot Up	0.313	1.6	0.75
20	WheatonIL	210	Tyrel_Derrick	1.28571428571429	4.5	1.062	Isolation	0.063	2	1
21	Knox	136	Tyre_Dukes	1.4	10.2	0.568	Off Screen	0.081	0.667	0.5
22	Knox	136	Tyre_Dukes	1.4	10.2	0.568	Offensive Rebounds (put b...	0.027	1	[null]
23	Knox	136	Tyre_Dukes	1.4	10.2	0.568	Isolation	0.081	0.667	0.333

4)get the "offensive weakness" of players

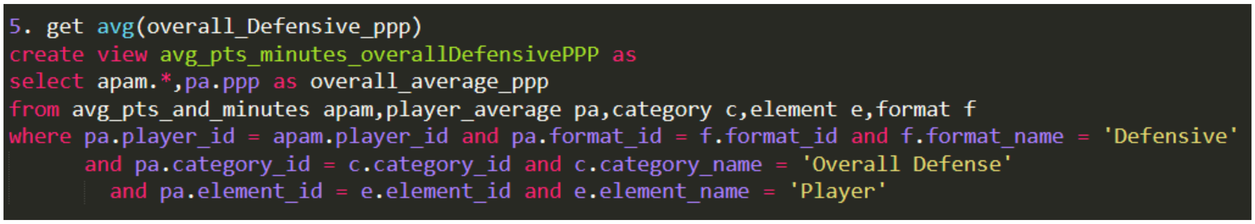


```
4.offensive_weakness_of_all_players
create view offensive_weakness_of_all_players as
select apmo.*,e.element_name,pa.percentage_of_time,pa.ppp as type_ppp,pa.field_goal_percentage
from avg_pts_minutes_overallOffensivePPP apmo,player_average pa,format f,category c,element e
where pa.player_id = apmo.player_id and pa.format_id = f.format_id and f.format_name = 'Offensive'
      and pa.category_id = c.category_id and c.category_name = 'Play Types'
        and pa.element_id = e.element_id and pa.ppp <apmo.overall_average_ppp
order by apmo.player_name DESC
```

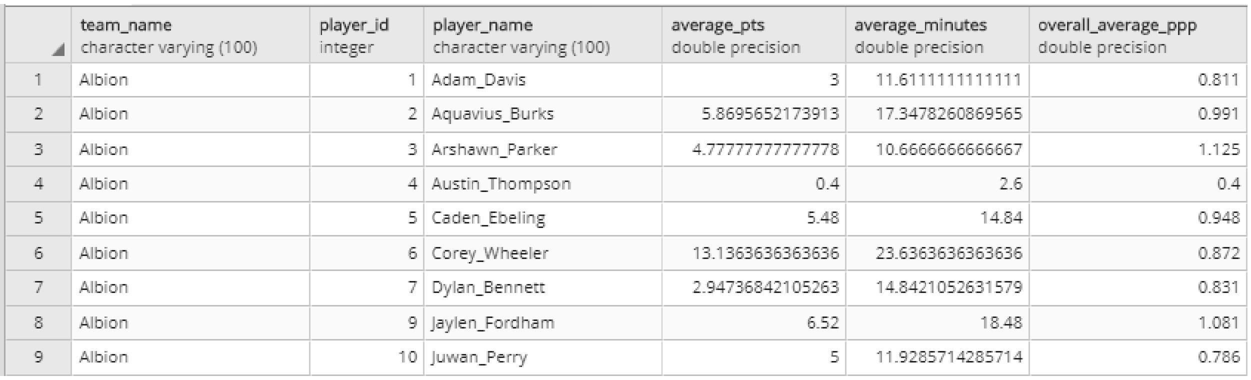


	team_name character varying (100)	player_id integer	player_name character varying (100)	average_pts double precision	average_minutes double precision	overall_average_ppp double precision	element_name character varying (100)	percentage_of_time double precision	type_ppp double precision	field_goal_percentage double precision
1	WheatonIL	211	Zack_Kvam	2.57692307692308	16.9230769230769	0.736	Cut	0.077	0.714	0.4
2	WheatonIL	211	Zack_Kvam	2.57692307692308	16.9230769230769	0.736	Miscellaneous	0.253	0.174	[null]
3	WheatonIL	211	Zack_Kvam	2.57692307692308	16.9230769230769	0.736	Isolation	0.055	0	0
4	WheatonIL	211	Zack_Kvam	2.57692307692308	16.9230769230769	0.736	P&R Ball Handler	0.055	0	0
5	GustavusAdolphus	312	Zach_Bloemker	6.04761904761905	15.8095238095238	0.777	Hand Off	0.038	0.667	0.2
6	GustavusAdolphus	312	Zach_Bloemker	6.04761904761905	15.8095238095238	0.777	Miscellaneous	0.064	0.7	0
7	GustavusAdolphus	312	Zach_Bloemker	6.04761904761905	15.8095238095238	0.777	Off Screen	0.102	0.625	0.333
8	GustavusAdolphus	312	Zach_Bloemker	6.04761904761905	15.8095238095238	0.777	Isolation	0.153	0.458	0.2
9	GustavusAdolphus	312	Zach_Bloemker	6.04761904761905	15.8095238095238	0.777	Transition	0.108	0.647	0.556
10	GustavusAdolphus	312	Zach_Bloemker	6.04761904761905	15.8095238095238	0.777	P&R Ball Handler	0.025	0.75	0.5
11	WheatonIL	212	Zac_Holman	1.09090909090909	5.36363636363636	0.765	Isolation	0.059	0	0
12	WheatonIL	212	Zac_Holman	1.09090909090909	5.36363636363636	0.765	Transition	0.059	0	0
13	WheatonIL	212	Zac_Holman	1.09090909090909	5.36363636363636	0.765	Miscellaneous	0.118	0	[null]
14	WheatonIL	212	Zac_Holman	1.09090909090909	5.36363636363636	0.765	Off Screen	0.176	0	0
15	NorthPark	250	Vegard_Tangen	3.69565217391304	15.9130434782609	0.842	P&R Roll Man	0.119	0.5	0.273
16	NorthPark	250	Vegard_Tangen	3.69565217391304	15.9130434782609	0.842	Miscellaneous	0.079	0	[null]

5)get the "overall defensive PPP" for each player.

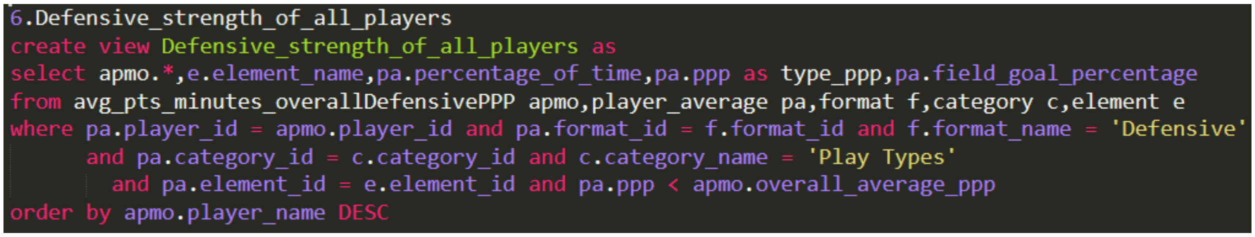


```
5. get avg(overall_Defensive_ppp)
create view avg_pts_minutes_overallDefensivePPP as
select apam.*,pa.ppp as overall_average_ppp
from avg_pts_and_minutes apam,player_average pa,category c,element e,format f
where pa.player_id = apam.player_id and pa.format_id = f.format_id and f.format_name = 'Defensive'
      and pa.category_id = c.category_id and c.category_name = 'Overall Defense'
        and pa.element_id = e.element_id and e.element_name = 'Player'
```

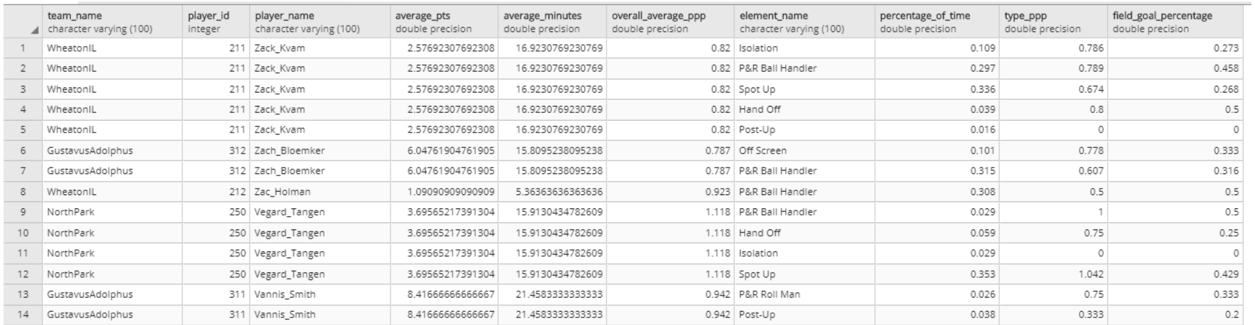


	team_name character varying (100)	player_id integer	player_name character varying (100)	average_pts double precision	average_minutes double precision	overall_average_ppp double precision
1	Albion	1	Adam_Davis	3	11.6111111111111	0.811
2	Albion	2	Aquavius_Burks	5.8695652173913	17.3478260869565	0.991
3	Albion	3	Arshawn_Parker	4.77777777777778	10.6666666666667	1.125
4	Albion	4	Austin_Thompson	0.4	2.6	0.4
5	Albion	5	Caden_Ebeling	5.48	14.84	0.948
6	Albion	6	Corey_Wheeler	13.1363636363636	23.6363636363636	0.872
7	Albion	7	Dylan_Bennett	2.94736842105263	14.8421052631579	0.831
8	Albion	9	Jaylen_Fordham	6.52	18.48	1.081
9	Albion	10	Juwan_Perry	5	11.9285714285714	0.786

6). get "Defensive Strength" for all players

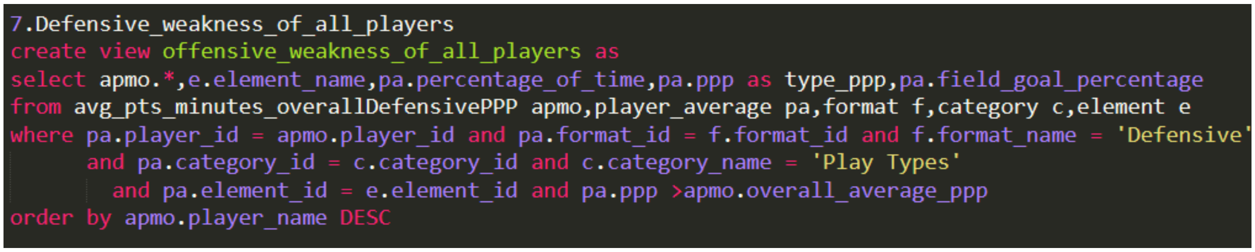


```
6.Defensive_strength_of_all_players
create view Defensive_strength_of_all_players as
select apmo.*,e.element_name,pa.percentage_of_time,pa.ppp as type_ppp,pa.field_goal_percentage
from avg_pts_minutes_overallDefensivePPP apmo,player_average pa,format f,category c,element e
where pa.player_id = apmo.player_id and pa.format_id = f.format_id and f.format_name = 'Defensive'
      and pa.category_id = c.category_id and c.category_name = 'Play Types'
        and pa.element_id = e.element_id and pa.ppp < apmo.overall_average_ppp
order by apmo.player_name DESC
```

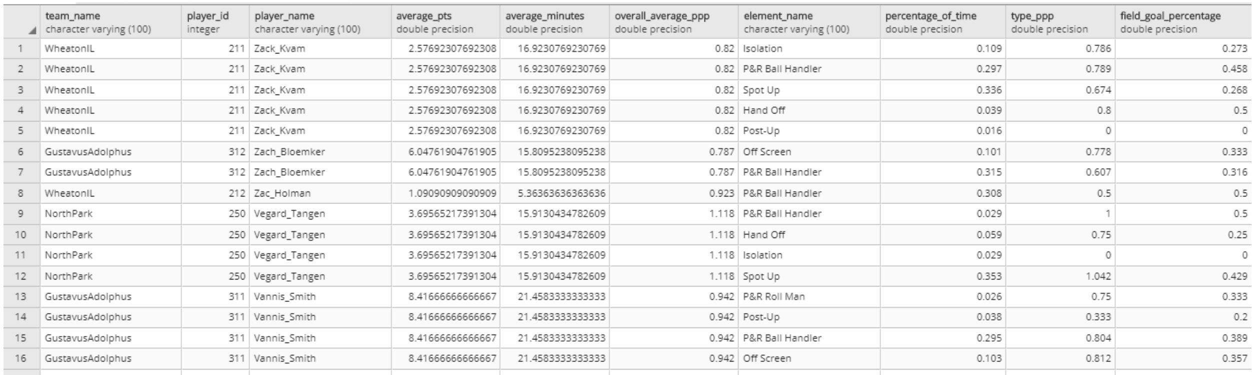


	team_name character varying (100)	player_id integer	player_name character varying (100)	average_pts double precision	average_minutes double precision	overall_average_ppp double precision	element_name character varying (100)	percentage_of_time double precision	type_ppp double precision	field_goal_percentage double precision
1	WheatonIL	211	Zack_Kvam	2.57692307692308	16.9230769230769	0.82	Isolation	0.109	0.786	0.273
2	WheatonIL	211	Zack_Kvam	2.57692307692308	16.9230769230769	0.82	P&R Ball Handler	0.297	0.789	0.458
3	WheatonIL	211	Zack_Kvam	2.57692307692308	16.9230769230769	0.82	Spot Up	0.336	0.674	0.268
4	WheatonIL	211	Zack_Kvam	2.57692307692308	16.9230769230769	0.82	Hand Off	0.039	0.8	0.5
5	WheatonIL	211	Zack_Kvam	2.57692307692308	16.9230769230769	0.82	Post-Up	0.016	0	0
6	GustavusAdolphus	312	Zach_Bloemker	6.04761904761905	15.8095238095238	0.787	Off Screen	0.101	0.778	0.333
7	GustavusAdolphus	312	Zach_Bloemker	6.04761904761905	15.8095238095238	0.787	P&R Ball Handler	0.315	0.607	0.316
8	WheatonIL	212	Zac_Holman	1.09090909090909	5.36363636363636	0.923	P&R Ball Handler	0.308	0.5	0.5
9	NorthPark	250	Vegard_Tangen	3.69565217391304	15.9130434782609	1.118	P&R Ball Handler	0.029	1	0.5
10	NorthPark	250	Vegard_Tangen	3.69565217391304	15.9130434782609	1.118	Hand Off	0.059	0.75	0.25
11	NorthPark	250	Vegard_Tangen	3.69565217391304	15.9130434782609	1.118	Isolation	0.029	0	0
12	NorthPark	250	Vegard_Tangen	3.69565217391304	15.9130434782609	1.118	Spot Up	0.353	1.042	0.429
13	GustavusAdolphus	311	Vannis_Smith	8.41666666666667	21.4583333333333	0.942	P&R Roll Man	0.026	0.75	0.333
14	GustavusAdolphus	311	Vannis_Smith	8.41666666666667	21.4583333333333	0.942	Post-Up	0.038	0.333	0.2

7)get "Defensive Weakness" for all players



```
7.Defensive_weakness_of_all_players
create view offensive_weakness_of_all_players as
select apmo.*,e.element_name,pa.percentage_of_time,pa.ppp as type_ppp,pa.field_goal_percentage
from avg_pts_minutes_overallDefensivePPP apmo,player_average pa,format f,category c,element e
where pa.player_id = apmo.player_id and pa.format_id = f.format_id and f.format_name = 'Defensive'
      and pa.category_id = c.category_id and c.category_name = 'Play Types'
        and pa.element_id = e.element_id and pa.ppp >apmo.overall_average_ppp
order by apmo.player_name DESC
```



	team_name	player_id	player_name	average_pts	average_minutes	overall_average_ppp	element_name	percentage_of_time	type_ppp	field_goal_percentage
	character varying (100)	integer	character varying (100)	double precision	double precision	double precision	character varying (100)	double precision	double precision	double precision
1	WheatonIL	211	Zack_Kvam	2.57692307692308	16.9230769230769	0.82	Isolation	0.109	0.786	0.273
2	WheatonIL	211	Zack_Kvam	2.57692307692308	16.9230769230769	0.82	P&R Ball Handler	0.297	0.789	0.458
3	WheatonIL	211	Zack_Kvam	2.57692307692308	16.9230769230769	0.82	Spot Up	0.336	0.674	0.268
4	WheatonIL	211	Zack_Kvam	2.57692307692308	16.9230769230769	0.82	Hand Off	0.039	0.8	0.5
5	WheatonIL	211	Zack_Kvam	2.57692307692308	16.9230769230769	0.82	Post-Up	0.016	0	0
6	GustavusAdolphus	312	Zach_Bloemker	6.04761904761905	15.8095238095238	0.787	Off Screen	0.101	0.778	0.333
7	GustavusAdolphus	312	Zach_Bloemker	6.04761904761905	15.8095238095238	0.787	P&R Ball Handler	0.315	0.607	0.316
8	WheatonIL	212	Zac_Holman	1.09090909090909	5.36363636363636	0.923	P&R Ball Handler	0.308	0.5	0.5
9	NorthPark	250	Vegard_Tangen	3.69565217391304	15.9130434782609	1.118	P&R Ball Handler	0.029	1	0.5
10	NorthPark	250	Vegard_Tangen	3.69565217391304	15.9130434782609	1.118	Hand Off	0.059	0.75	0.25
11	NorthPark	250	Vegard_Tangen	3.69565217391304	15.9130434782609	1.118	Isolation	0.029	0	0
12	NorthPark	250	Vegard_Tangen	3.69565217391304	15.9130434782609	1.118	Spot Up	0.353	1.042	0.429
13	GustavusAdolphus	311	Vannis_Smith	8.41666666666667	21.4583333333333	0.942	P&R Roll Man	0.026	0.75	0.333
14	GustavusAdolphus	311	Vannis_Smith	8.41666666666667	21.4583333333333	0.942	Post-Up	0.038	0.333	0.2
15	GustavusAdolphus	311	Vannis_Smith	8.41666666666667	21.4583333333333	0.942	P&R Ball Handler	0.295	0.804	0.389
16	GustavusAdolphus	311	Vannis_Smith	8.41666666666667	21.4583333333333	0.942	Off Screen	0.103	0.812	0.357

5) Combine strength and weakness in csv file

Create a new field in each results' row as "type", field_name = 'strength'

Field_name = 'weakness'

6)Normalization using "Adjustment Factor"

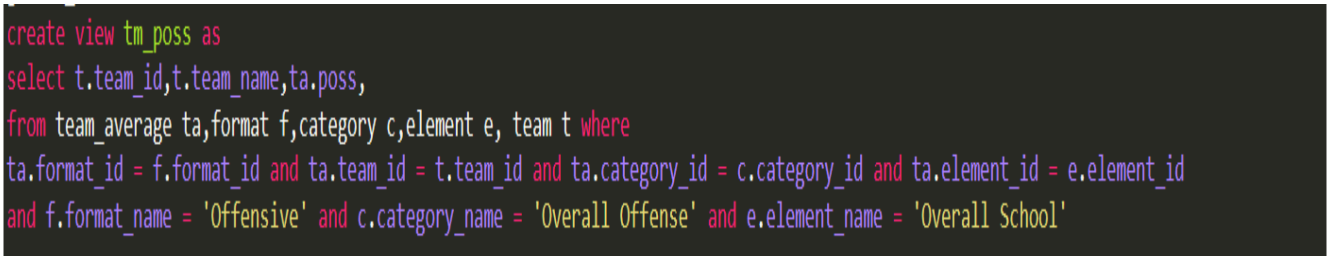
After getting these stats, I noticed that it is very necessary and makes more sense to "normalize" them by comparing "league" average Possesions (Offensively and Defensively) with each team's possession (Offensively and Defensively).

The normalization formula is:

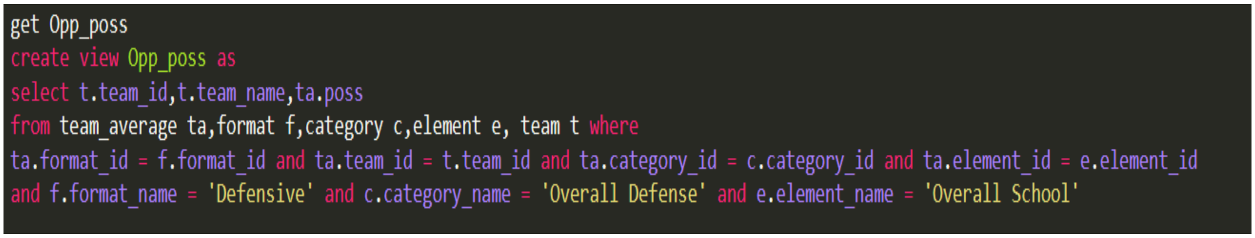
*Adjusted_PPP = Unadjusted PPP(which we got above) *normalize_factor*

(normalize factor = team_average_possesion/league_average_possesion)

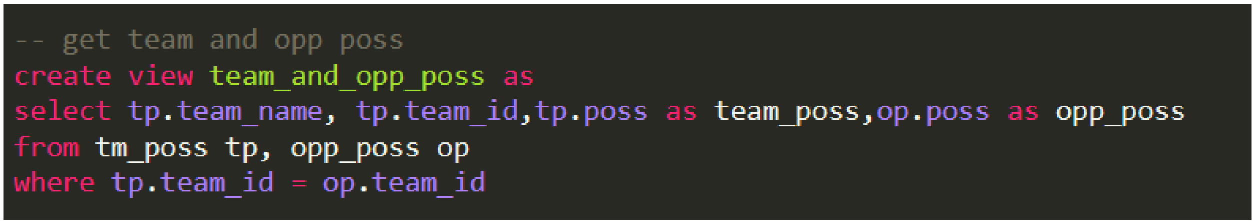
We use SQL below to get the average team_possesion and opponent_possesion:



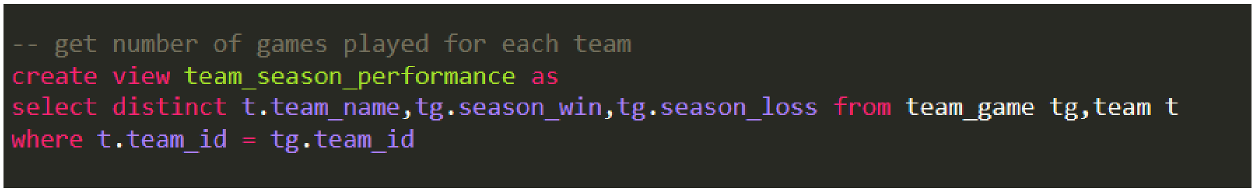
```
create view tm_poss as
select t.team_id,t.team_name,ta.poss,
from team_average ta,format f,category c,element e, team t where
ta.format_id = f.format_id and ta.team_id = t.team_id and ta.category_id = c.category_id and ta.element_id = e.element_id
and f.format_name = 'Offensive' and c.category_name = 'Overall Offense' and e.element_name = 'Overall School'
```



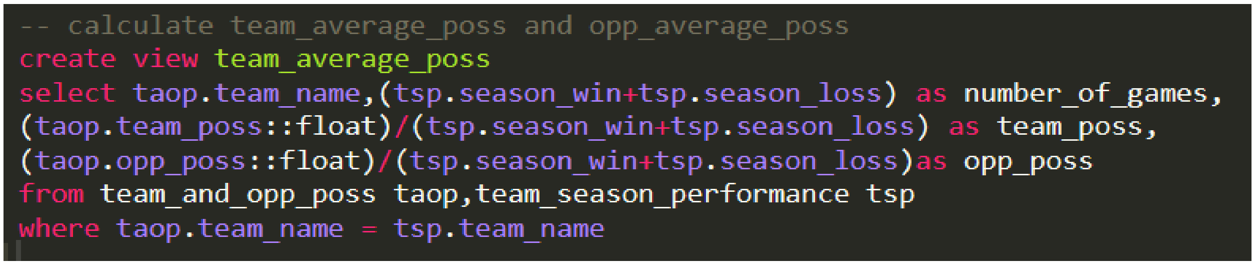
```
get Opp_poss
create view Opp_poss as
select t.team_id,t.team_name,ta.poss
from team_average ta,format f,category c,element e, team t where
ta.format_id = f.format_id and ta.team_id = t.team_id and ta.category_id = c.category_id and ta.element_id = e.element_id
and f.format_name = 'Defensive' and c.category_name = 'Overall Defense' and e.element_name = 'Overall School'
```



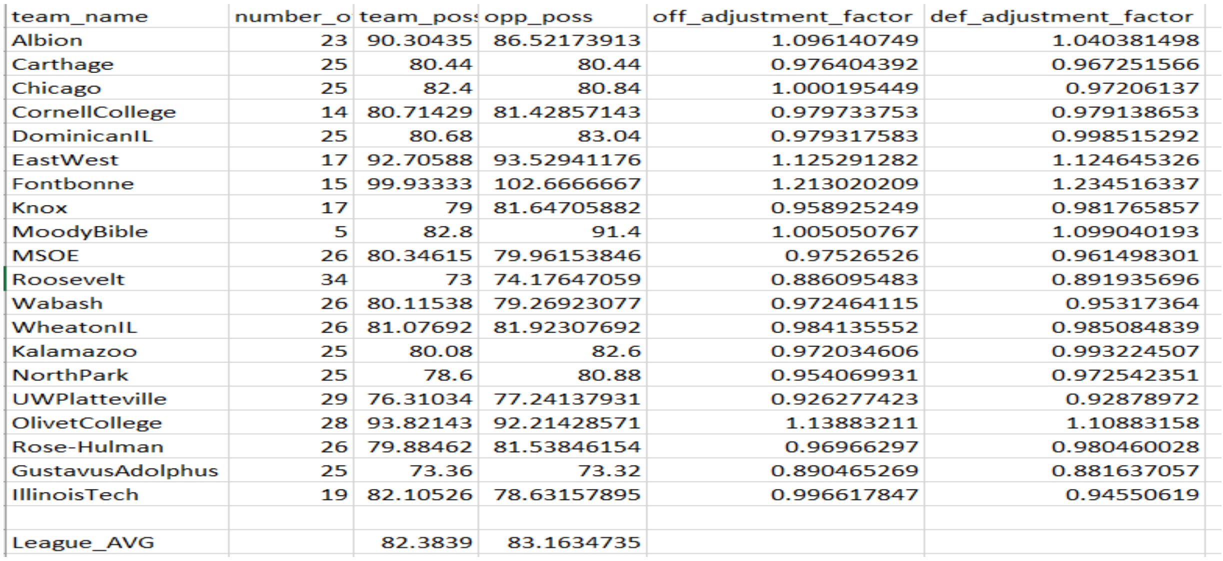
```
-- get team and opp poss
create view team_and_opp_poss as
select tp.team_name, tp.team_id,tp.poss as team_poss,op.poss as opp_poss
from tm_poss tp, opp_poss op
where tp.team_id = op.team_id
```



```
-- get number of games played for each team
create view team_season_performance as
select distinct t.team_name,tg.season_win,tg.season_loss from team_game tg,team t
where t.team_id = tg.team_id
```



```
-- calculate team_average_poss and opp_average_poss
create view team_average_poss
select taop.team_name,(tsp.season_win+tsp.season_loss) as number_of_games,
(taop.team_poss::float)/(tsp.season_win+tsp.season_loss) as team_poss,
(taop.opp_poss::float)/(tsp.season_win+tsp.season_loss)as opp_poss
from team_and_opp_poss taop,team_season_performance tsp
where taop.team_name = tsp.team_name
```

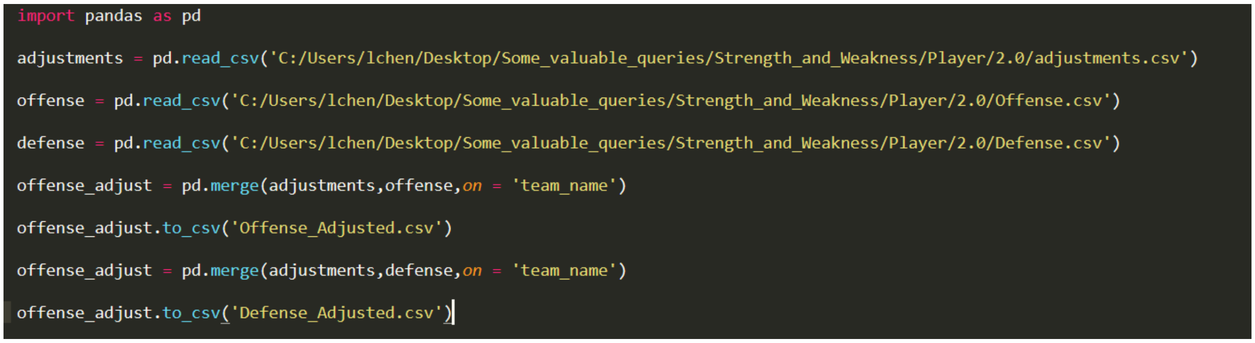


team_name	number_o	team_pos	opp_poss	off_adjustment_factor	def_adjustment_factor
Albion	23	90.30435	86.52173913	1.096140749	1.040381498
Carthage	25	80.44	80.44	0.976404392	0.967251566
Chicago	25	82.4	80.84	1.000195449	0.97206137
CornellCollege	14	80.71429	81.42857143	0.979733753	0.979138653
DominicanIL	25	80.68	83.04	0.979317583	0.998515292
EastWest	17	92.70588	93.52941176	1.125291282	1.124645326
Fontbonne	15	99.93333	102.6666667	1.213020209	1.234516337
Knox	17	79	81.64705882	0.958925249	0.981765857
MoodyBible	5	82.8	91.4	1.005050767	1.099040193
MSOE	26	80.34615	79.96153846	0.97526526	0.961498301
Roosevelt	34	73	74.17647059	0.886095483	0.891935696
Wabash	26	80.11538	79.26923077	0.972464115	0.95317364
WheatonIL	26	81.07692	81.92307692	0.984135552	0.985084839
Kalamazoo	25	80.08	82.6	0.972034606	0.993224507
NorthPark	25	78.6	80.88	0.954069931	0.972542351
UWPlatteville	29	76.31034	77.24137931	0.926277423	0.92878972
OlivetCollege	28	93.82143	92.21428571	1.13883211	1.10883158
Rose-Hulman	26	79.88462	81.53846154	0.96966297	0.980460028
GustavusAdolphus	25	73.36	73.32	0.890465269	0.881637057
IllinoisTech	19	82.10526	78.63157895	0.996617847	0.94550619
League_AVG		82.3839	83.1634735		

7) format the final version of CSV and visualize in Tableau

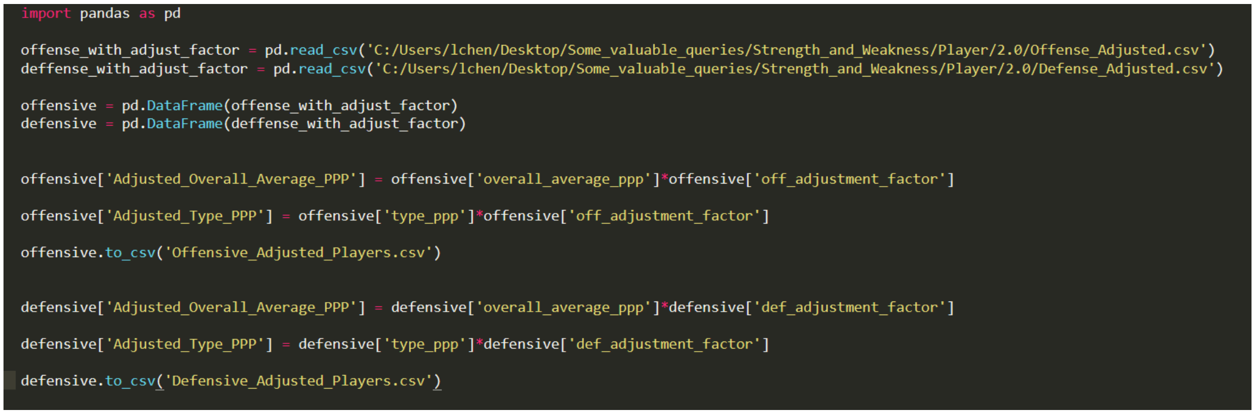
After getting those adjustment_factors, we can merge them with the csv in which we have the unadjusted PPPs, and then use ”Pandas” module in python to get the final individual “adjusted PPPs”

Merge:



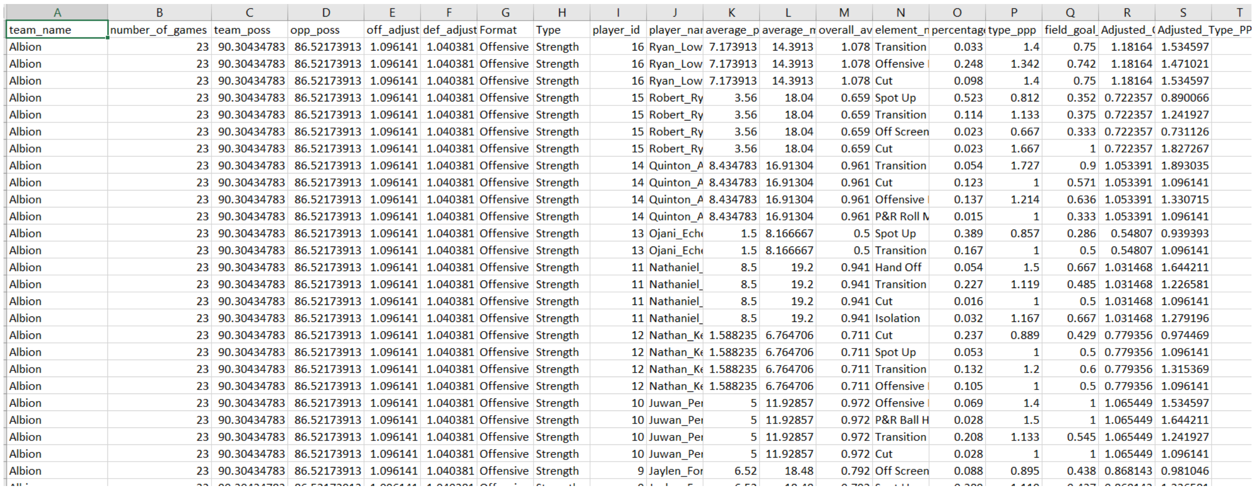
```
import pandas as pd

adjustments = pd.read_csv('C:/Users/lchen/Desktop/Some_valuable_queries/Strength_and_Weakness/Player/2.0/adjustments.csv')

offense = pd.read_csv('C:/Users/lchen/Desktop/Some_valuable_queries/Strength_and_Weakness/Player/2.0/Offense.csv')

defense = pd.read_csv('C:/Users/lchen/Desktop/Some_valuable_queries/Strength_and_Weakness/Player/2.0/Defense.csv')

offense_adjust = pd.merge(adjustments,offense,on = 'team_name')

offense_adjust.to_csv('Offense_Adjusted.csv')

offense_adjust = pd.merge(adjustments,defense,on = 'team_name')

offense_adjust.to_csv('Defense_Adjusted.csv')
```

Calculate the adjusted_PPPs:



```
import pandas as pd

offense_with_adjust_factor = pd.read_csv('C:/Users/lchen/Desktop/Some_valuable_queries/Strength_and_Weakness/Player/2.0/Offense_Adjusted.csv')
deffense_with_adjust_factor = pd.read_csv('C:/Users/lchen/Desktop/Some_valuable_queries/Strength_and_Weakness/Player/2.0/Defense_Adjusted.csv')

offensive = pd.DataFrame(offense_with_adjust_factor)
defensive = pd.DataFrame(deffense_with_adjust_factor)


offensive['Adjusted_Overall_Average_PPP'] = offensive['overall_average_ppp']*offensive['off_adjustment_factor']

offensive['Adjusted_Type_PPP'] = offensive['type_ppp']*offensive['off_adjustment_factor']

offensive.to_csv('Offensive_Adjusted_Players.csv')


defensive['Adjusted_Overall_Average_PPP'] = defensive['overall_average_ppp']*defensive['def_adjustment_factor']

defensive['Adjusted_Type_PPP'] = defensive['type_ppp']*defensive['def_adjustment_factor']

defensive.to_csv('Defensive_Adjusted_Players.csv')
```

Final version table:



A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
team_name	number_of_games	team_poss	opp_poss	off_adjust	def_adjust	Format	Type	player_id	player_nar	average_p	average_n	overall_av	element_n	percentage	type_ppp	field_goal_	Adjusted_(	Adjusted_Type_PP	
Albion	23	90.30434783	86.52173913	1.096141	1.040381	Offensive	Strength	16	Ryan_Low	7.173913	14.3913	1.078	Transition	0.033	1.4	0.75	1.18164	1.534597	
Albion	23	90.30434783	86.52173913	1.096141	1.040381	Offensive	Strength	16	Ryan_Low	7.173913	14.3913	1.078	Offensive	0.248	1.342	0.742	1.18164	1.471021	
Albion	23	90.30434783	86.52173913	1.096141	1.040381	Offensive	Strength	16	Ryan_Low	7.173913	14.3913	1.078	Cut	0.098	1.4	0.75	1.18164	1.534597	
Albion	23	90.30434783	86.52173913	1.096141	1.040381	Offensive	Strength	15	Robert_Ry	3.56	18.04	0.659	Spot Up	0.523	0.812	0.352	0.722357	0.890066	
Albion	23	90.30434783	86.52173913	1.096141	1.040381	Offensive	Strength	15	Robert_Ry	3.56	18.04	0.659	Transition	0.114	1.133	0.375	0.722357	1.241927	
Albion	23	90.30434783	86.52173913	1.096141	1.040381	Offensive	Strength	15	Robert_Ry	3.56	18.04	0.659	Off Screen	0.023	0.667	0.333	0.722357	0.731126	
Albion	23	90.30434783	86.52173913	1.096141	1.040381	Offensive	Strength	15	Robert_Ry	3.56	18.04	0.659	Cut	0.023	1.667	1	0.722357	1.827267	
Albion	23	90.30434783	86.52173913	1.096141	1.040381	Offensive	Strength	14	Quinton_A	8.434783	16.91304	0.961	Transition	0.054	1.727	0.9	1.053391	1.893035	
Albion	23	90.30434783	86.52173913	1.096141	1.040381	Offensive	Strength	14	Quinton_A	8.434783	16.91304	0.961	Cut	0.123	1	0.571	1.053391	1.096141	
Albion	23	90.30434783	86.52173913	1.096141	1.040381	Offensive	Strength	14	Quinton_A	8.434783	16.91304	0.961	Offensive	0.137	1.214	0.636	1.053391	1.330715	
Albion	23	90.30434783	86.52173913	1.096141	1.040381	Offensive	Strength	14	Quinton_A	8.434783	16.91304	0.961	P&R Roll M	0.015	1	0.333	1.053391	1.096141	
Albion	23	90.30434783	86.52173913	1.096141	1.040381	Offensive	Strength	13	Ojani_Eche	1.5	8.166667	0.5	Spot Up	0.389	0.857	0.286	0.54807	0.939393	
Albion	23	90.30434783	86.52173913	1.096141	1.040381	Offensive	Strength	13	Ojani_Eche	1.5	8.166667	0.5	Transition	0.167	1	0.5	0.54807	1.096141	
Albion	23	90.30434783	86.52173913	1.096141	1.040381	Offensive	Strength	11	Nathaniel_	8.5	19.2	0.941	Hand Off	0.054	1.5	0.667	1.031468	1.644211	
Albion	23	90.30434783	86.52173913	1.096141	1.040381	Offensive	Strength	11	Nathaniel_	8.5	19.2	0.941	Transition	0.227	1.119	0.485	1.031468	1.226581	
Albion	23	90.30434783	86.52173913	1.096141	1.040381	Offensive	Strength	11	Nathaniel_	8.5	19.2	0.941	Cut	0.016	1	0.5	1.031468	1.096141	
Albion	23	90.30434783	86.52173913	1.096141	1.040381	Offensive	Strength	11	Nathaniel_	8.5	19.2	0.941	Isolation	0.032	1.167	0.667	1.031468	1.279196	
Albion	23	90.30434783	86.52173913	1.096141	1.040381	Offensive	Strength	12	Nathan_Ke	1.588235	6.764706	0.711	Cut	0.237	0.889	0.429	0.779356	0.974469	
Albion	23	90.30434783	86.52173913	1.096141	1.040381	Offensive	Strength	12	Nathan_Ke	1.588235	6.764706	0.711	Spot Up	0.053	1	0.5	0.779356	1.096141	
Albion	23	90.30434783	86.52173913	1.096141	1.040381	Offensive	Strength	12	Nathan_Ke	1.588235	6.764706	0.711	Transition	0.132	1.2	0.6	0.779356	1.315369	
Albion	23	90.30434783	86.52173913	1.096141	1.040381	Offensive	Strength	12	Nathan_Ke	1.588235	6.764706	0.711	Offensive	0.105	1	0.5	0.779356	1.096141	
Albion	23	90.30434783	86.52173913	1.096141	1.040381	Offensive	Strength	10	Juwan_Per	5	11.92857	0.972	Offensive	0.069	1.4	1	1.065449	1.534597	
Albion	23	90.30434783	86.52173913	1.096141	1.040381	Offensive	Strength	10	Juwan_Per	5	11.92857	0.972	P&R Ball H	0.028	1.5	1	1.065449	1.644211	
Albion	23	90.30434783	86.52173913	1.096141	1.040381	Offensive	Strength	10	Juwan_Per	5	11.92857	0.972	Transition	0.208	1.133	0.545	1.065449	1.241927	
Albion	23	90.30434783	86.52173913	1.096141	1.040381	Offensive	Strength	10	Juwan_Per	5	11.92857	0.972	Cut	0.028	1	1	1.065449	1.096141	
Albion	23	90.30434783	86.52173913	1.096141	1.040381	Offensive	Strength	9	Jaylen_For	6.52	18.48	0.792	Off Screen	0.088	0.895	0.438	0.868143	0.981046	

Tableau visualization:

https://public.tableau.com/profile/chenjie.li#!/vizhome/Adjusted_Player_Strength_Weakness/Dashboard1

3.1.4. Conclusions

Here are some of (not all of them) the conclusions we got for IIT players specifically.

Player:

Anthony Mosley:

He should reduce the number of ISOs.

He is good at transition, probably coach should give more chances to him when in a transition.

Jake_Bruns:

He should reduce the number of "Off Screens"

He should be given more chances of P&Rs and Transitions.

Jake Digiorgio:

He should reduce the number of "Spot-Ups"

He should be given more chances of "Cuts" and "Post-Ups"

Malick Howze:

He is not good at P&R

Do more "ISO" and "Transition" is a good idea for him

Max Hisatake:

He is not very efficient in post-up

He can do more on "Cut"

3.2 Teams

With the same logic, We got the similar visualizations for Teams. The only difference we used different filter : score difference, so that we will be able to filter opponents by last season's game results.

Here is the visualization of Teams:

https://public.tableau.com/profile/chenjie.li#!/vizhome/AdjustedTeamsStrengthAndWeakness/Dashboard1

More details and the codes used please refer to my GitHub,

3.2 PER Calculation

3.2.1.Introduction

The Player Efficiency Rating (PER) is a per-minute rating developed by ESPN.com columnist John Hollinger. In John's words, "The PER sums up all a player's positive accomplishments, subtracts the negative accomplishments, and returns a per-minute rating of a player's performance." It appears from his books that John's database only goes back to the 1988-89 season. I decided to expand on John's work and calculate PER for all players since minutes played were first recorded (1951-52).

Some pros of PER

1)Can give you a straightforward idea of how good a player is

2)Useful in comparing seasons

3)A universal recognized valuable metric to evaluate a player using a “Single number”

Some cons of PER:

1)Doesn’t give enough credits to a Player’s defensive value

2)Overrate the Rebounds a little bit.

3)Doesn’t value FT enough

4)Overvalue the “volume” players

5)Undervalues the MPG

3.2.2. Formula

All calculations begin with what we call an unadjusted PER (uPER). The formula is:

```
uPER = (1 / MP) *
   [ 3P
   + (2/3) * AST
   + (2 - factor * (team_AST / team_FG)) * FG
   + (FT *0.5 * (1 + (1 - (team_AST / team_FG)) + (2/3) * (team_AST / team_FG)))
   - VOP * TOV
   - VOP * DRB% * (FGA - FG)
   - VOP * 0.44 * (0.44 + (0.56 * DRB%)) * (FTA - FT)
   + VOP * (1 - DRB%) * (TRB - ORB)
   + VOP * DRB% * ORB
   + VOP * STL
   + VOP * DRB% * BLK
   - PF * ((lg_FT / lg_PF) - 0.44 * (lg_FTA / lg_PF) * VOP) ]
```

Most of the terms in the formula above should be clear, some of the factors are defined by Mr. Hollinger:

```
factor = (2 / 3) - (0.5 * (lg_AST / lg_FG)) / (2 * (lg_FG / lg_FT))
VOP    = lg_PTS / (lg_FGA - lg_ORB + lg_TOV + 0.44 * lg_FTA)
DRB%   = (lg_TRB - lg_ORB) / lg_TRB
```

Note here that “lg” is not log10 but means “league”

After uPER is calculated, an adjustment must be made for the team's pace. The pace adjustment is:

```
pace adjustment = lg_Pace / team_Pace
```

Pace

Pace Factor (available since the 1973-74 season in the NBA); the formula is 48 * ((Tm Poss + Opp Poss) / (2 * (Tm MP / 5))). Pace factor is an estimate of the number of possessions per 48 minutes by a team. (Note: 40 minutes is used in the calculation for the College Basketball.)

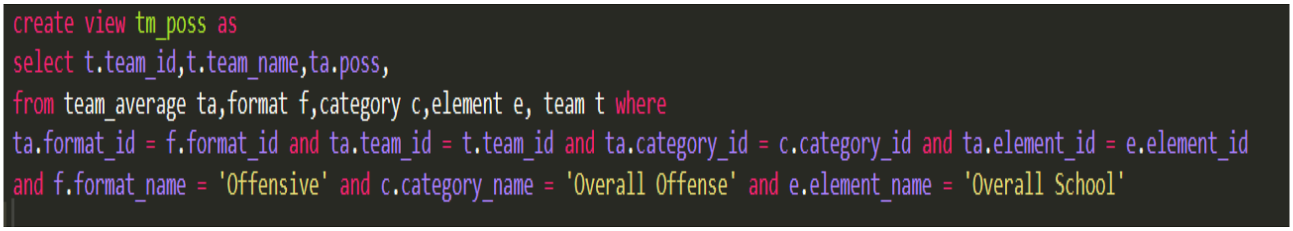
3.2.3.Practice

In this section I will show you how did I implement each step in the formula using Postgresql Database ”Basketball” I created and Python module “Pandas”.

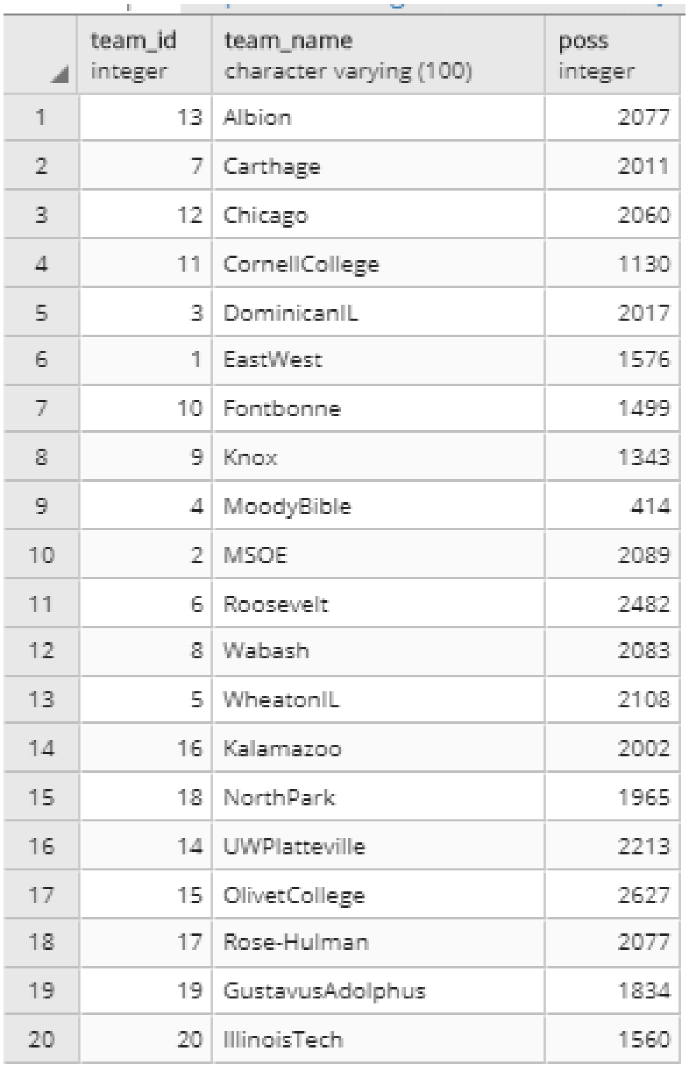
One thing to note is that since the formula includes the “league” information and IIT doesn’t have a formal kind of league, so I just created a so-called league which includes the teams that IIT played against in the last season.

1)Get Pace factors

Get Tm_Poss

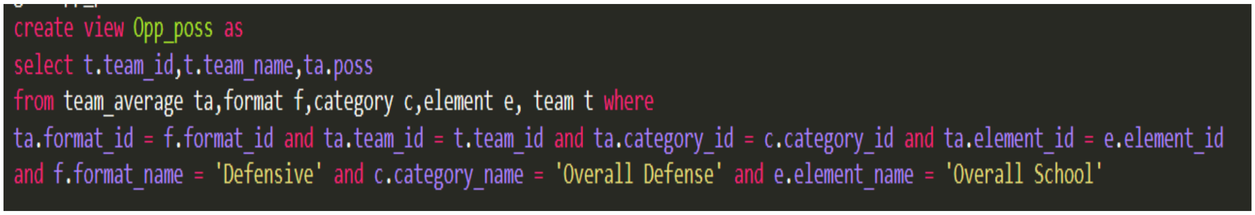


```
create view tm_poss as
select t.team_id,t.team_name,ta.poss,
from team_average ta,format f,category c,element e, team t where
ta.format_id = f.format_id and ta.team_id = t.team_id and ta.category_id = c.category_id and ta.element_id = e.element_id
and f.format_name = 'Offensive' and c.category_name = 'Overall Offense' and e.element_name = 'Overall School'
```

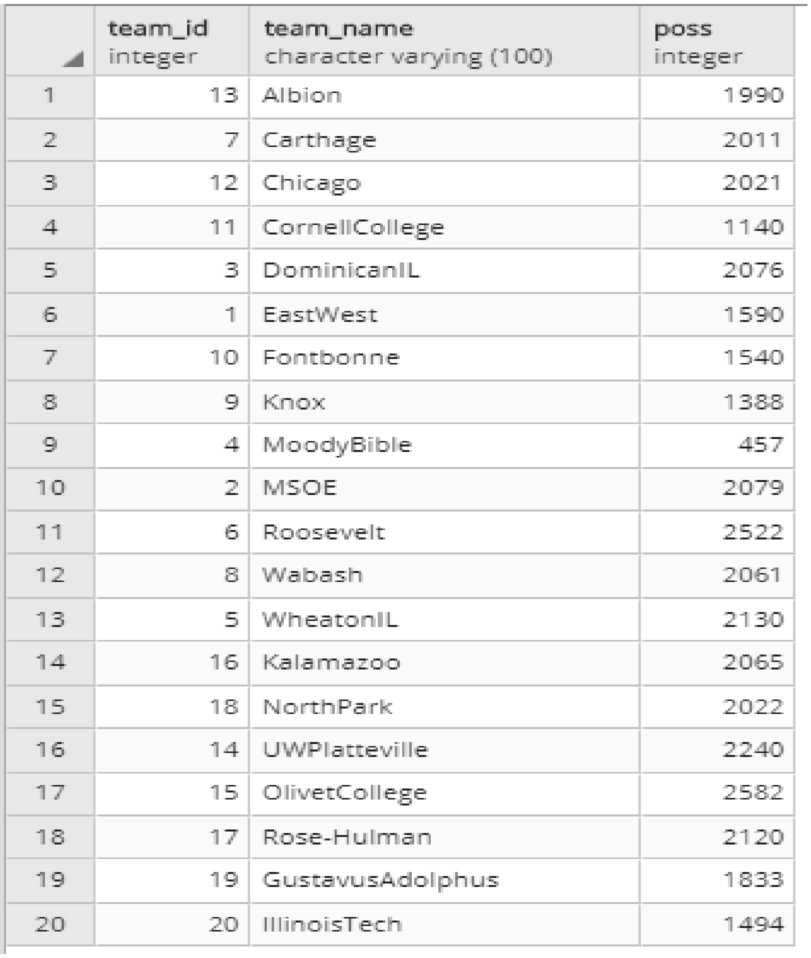


	team_id integer	team_name character varying (100)	poss integer
1	13	Albion	2077
2	7	Carthage	2011
3	12	Chicago	2060
4	11	CornellCollege	1130
5	3	DominicanIL	2017
6	1	EastWest	1576
7	10	Fontbonne	1499
8	9	Knox	1343
9	4	MoodyBible	414
10	2	MSOE	2089
11	6	Roosevelt	2482
12	8	Wabash	2083
13	5	WheatonIL	2108
14	16	Kalamazoo	2002
15	18	NorthPark	1965
16	14	UWPlatteville	2213
17	15	OlivetCollege	2627
18	17	Rose-Hulman	2077
19	19	GustavusAdolphus	1834
20	20	IllinoisTech	1560

Get Opp_Poss

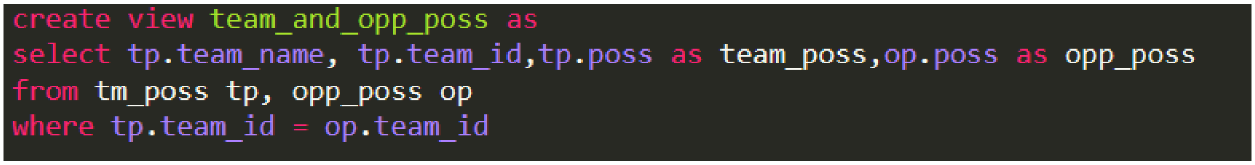


```
create view Opp_poss as
select t.team_id,t.team_name,ta.poss
from team_average ta,format f,category c,element e, team t where
ta.format_id = f.format_id and ta.team_id = t.team_id and ta.category_id = c.category_id and ta.element_id = e.element_id
and f.format_name = 'Defensive' and c.category_name = 'Overall Defense' and e.element_name = 'Overall School'
```

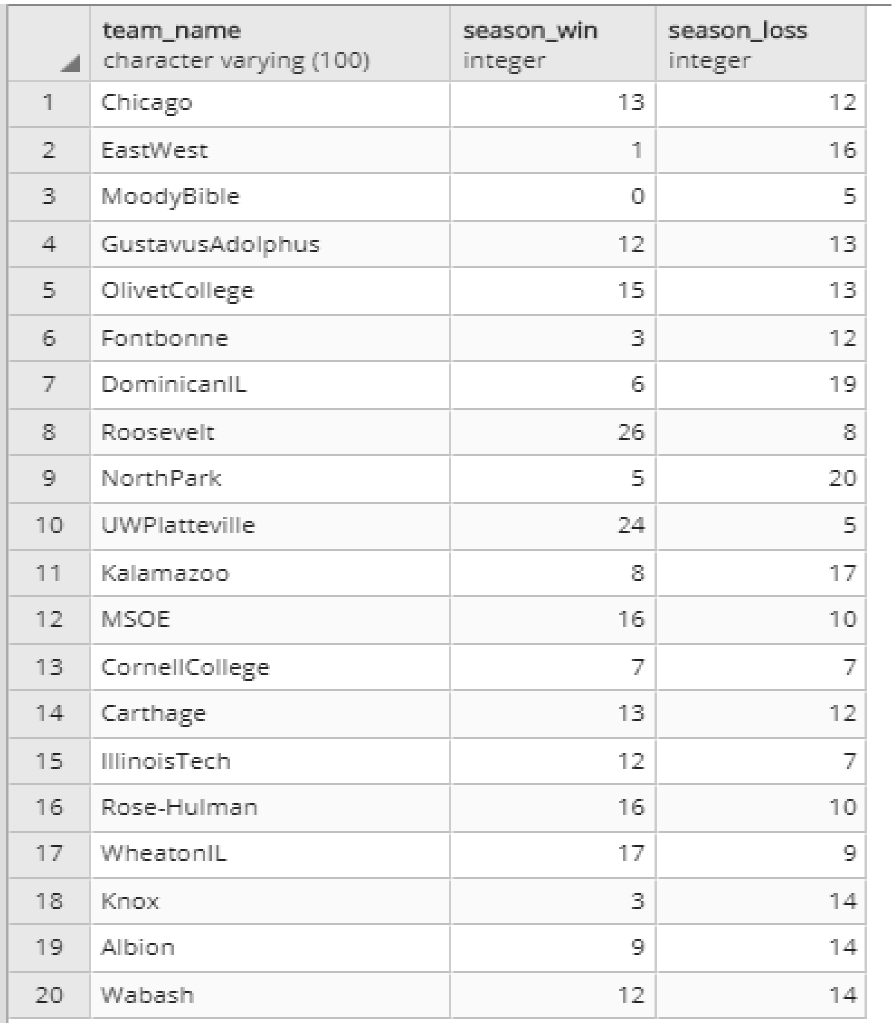


	team_id integer	team_name character varying (100)	poss integer
1	13	Albion	1990
2	7	Carthage	2011
3	12	Chicago	2021
4	11	CornellCollege	1140
5	3	DominicanIL	2076
6	1	EastWest	1590
7	10	Fontbonne	1540
8	9	Knox	1388
9	4	MoodyBible	457
10	2	MSOE	2079
11	6	Roosevelt	2522
12	8	Wabash	2061
13	5	WheatonIL	2130
14	16	Kalamazoo	2065
15	18	NorthPark	2022
16	14	UWPlatteville	2240
17	15	OlivetCollege	2582
18	17	Rose-Hulman	2120
19	19	GustavusAdolphus	1833
20	20	IllinoisTech	1494

3) Combine 1) and 2) get team and opp poss

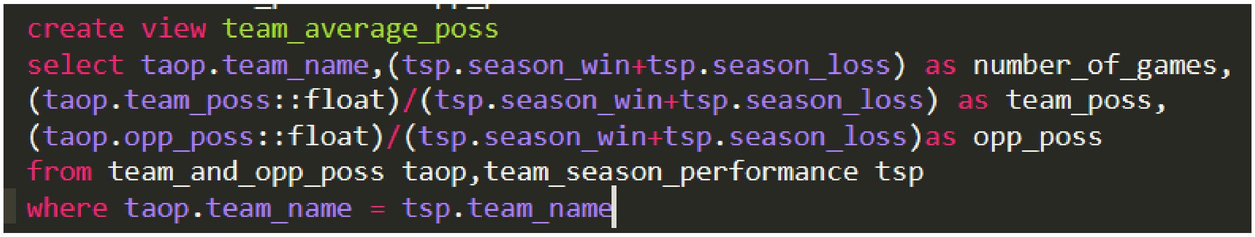


```
create view team_and_opp_poss as
select tp.team_name, tp.team_id,tp.poss as team_poss,op.poss as opp_poss
from tm_poss tp, opp_poss op
where tp.team_id = op.team_id
```

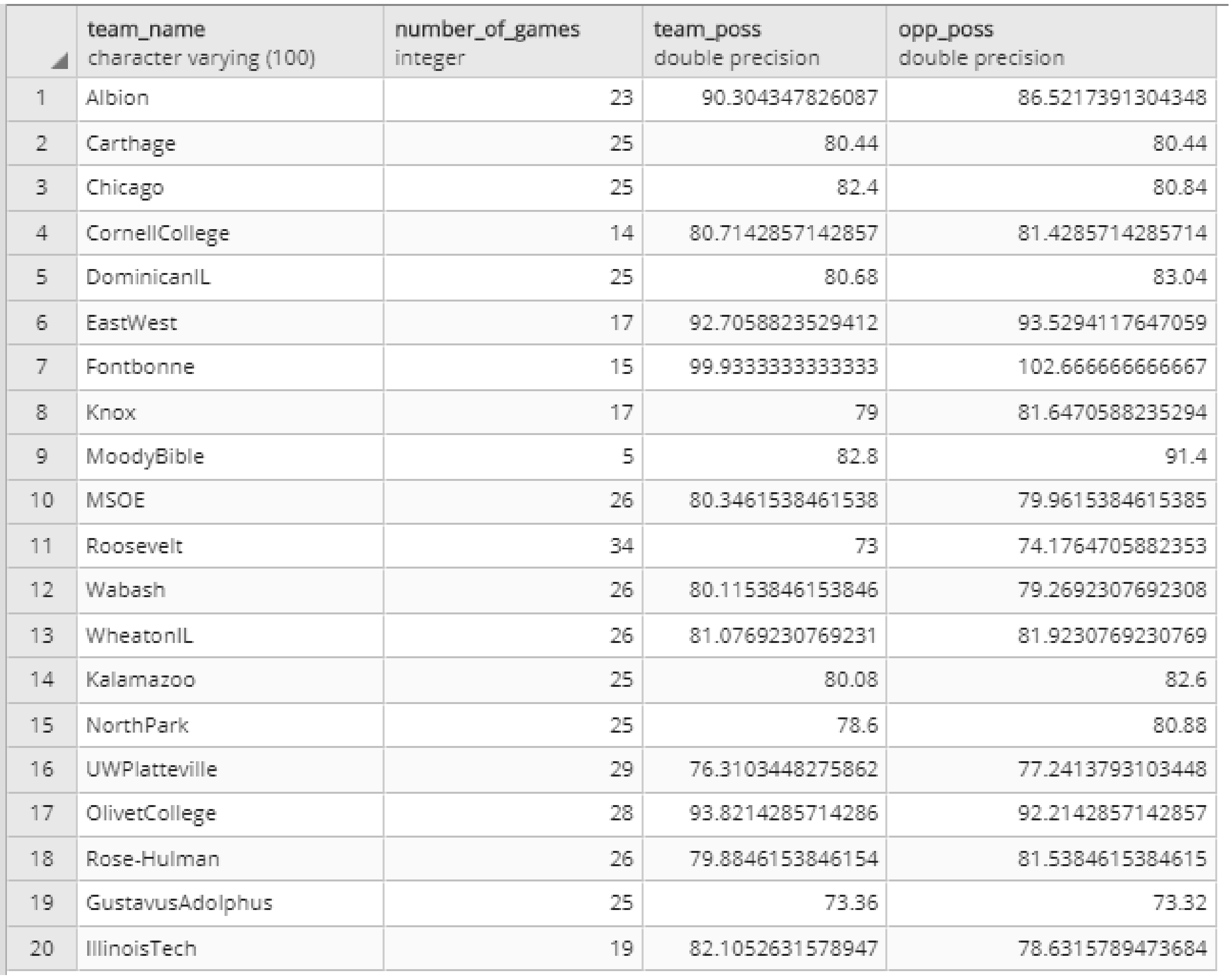


	team_name character varying (100)	season_win integer	season_loss integer
1	Chicago	13	12
2	EastWest	1	16
3	MoodyBible	0	5
4	GustavusAdolphus	12	13
5	OlivetCollege	15	13
6	Fontbonne	3	12
7	DominicanIL	6	19
8	Roosevelt	26	8
9	NorthPark	5	20
10	UWPlatteville	24	5
11	Kalamazoo	8	17
12	MSOE	16	10
13	CornellCollege	7	7
14	Carthage	13	12
15	IllinoisTech	12	7
16	Rose-Hulman	16	10
17	WheatonIL	17	9
18	Knox	3	14
19	Albion	9	14
20	Wabash	12	14

Calculate average team_poss and average opp_poss



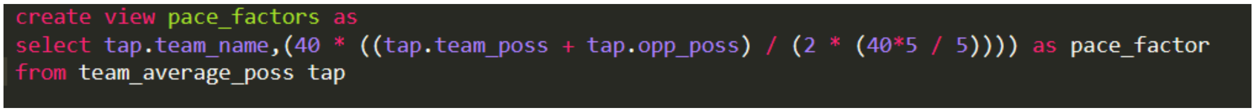
```
create view team_average_poss
select taop.team_name,(tsp.season_win+tsp.season_loss) as number_of_games,
(taop.team_poss::float)/(tsp.season_win+tsp.season_loss) as team_poss,
(taop.opp_poss::float)/(tsp.season_win+tsp.season_loss)as opp_poss
from team_and_opp_poss taop,team_season_performance tsp
where taop.team_name = tsp.team_name
```



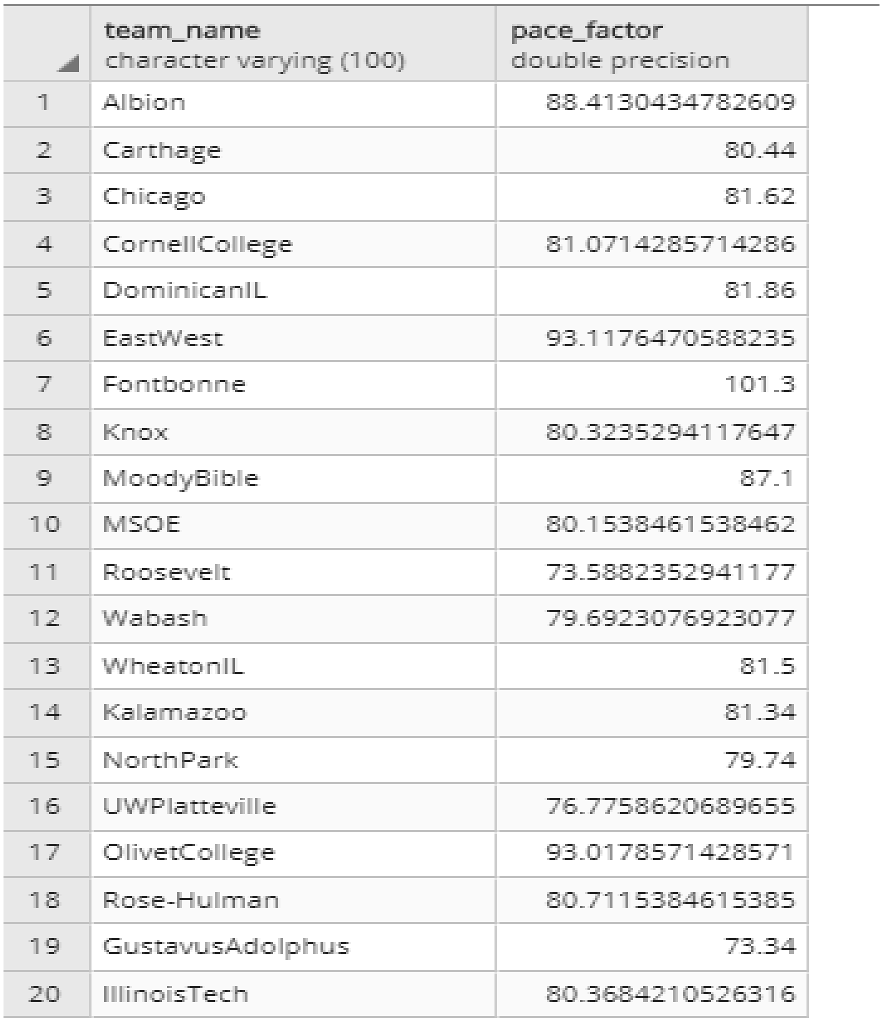
	team_name character varying (100)	number_of_games integer	team_poss double precision	opp_poss double precision
1	Albion	23	90.304347826087	86.5217391304348
2	Carthage	25	80.44	80.44
3	Chicago	25	82.4	80.84
4	CornellCollege	14	80.7142857142857	81.4285714285714
5	DominicanIL	25	80.68	83.04
6	EastWest	17	92.7058823529412	93.5294117647059
7	Fontbonne	15	99.9333333333333	102.666666666667
8	Knox	17	79	81.6470588235294
9	MoodyBible	5	82.8	91.4
10	MSOE	26	80.3461538461538	79.9615384615385
11	Roosevelt	34	73	74.1764705882353
12	Wabash	26	80.1153846153846	79.2692307692308
13	WheatonIL	26	81.0769230769231	81.9230769230769
14	Kalamazoo	25	80.08	82.6
15	NorthPark	25	78.6	80.88
16	UWPlatteville	29	76.3103448275862	77.2413793103448
17	OlivetCollege	28	93.8214285714286	92.2142857142857
18	Rose-Hulman	26	79.8846153846154	81.5384615384615
19	GustavusAdolphus	25	73.36	73.32
20	IllinoisTech	19	82.1052631578947	78.6315789473684

calculate Pace factors:

The original formula is 48 * ((Tm_Poss + Opp_Poss) / (2 * (Tm_MP / 5))).Since PER was invented based on NBA stats, but now we are analyzing the stats of College basketball, so we replace “48” in the formula with 40 and Tm_MP = 5*40 (instead of 5*48)

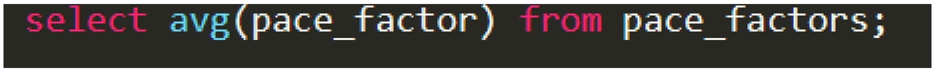


```
create view pace_factors as
select tap.team_name,(40 * ((tap.team_poss + tap.opp_poss) / (2 * (40*5 / 5)))) as pace_factor
from team_average_poss tap
```

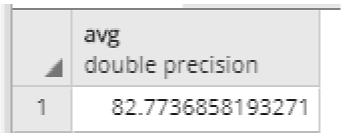


	team_name character varying (100)	pace_factor double precision
1	Albion	88.4130434782609
2	Carthage	80.44
3	Chicago	81.62
4	CornellCollege	81.0714285714286
5	DominicanIL	81.86
6	EastWest	93.1176470588235
7	Fontbonne	101.3
8	Knox	80.3235294117647
9	MoodyBible	87.1
10	MSOE	80.1538461538462
11	Roosevelt	73.5882352941177
12	Wabash	79.6923076923077
13	WheatonIL	81.5
14	Kalamazoo	81.34
15	NorthPark	79.74
16	UWPlatteville	76.7758620689655
17	OlivetCollege	93.0178571428571
18	Rose-Hulman	80.7115384615385
19	GustavusAdolphus	73.34
20	IllinoisTech	80.3684210526316

6)get league average Pace_Factor: lg_Pace



```
select avg(pace_factor) from pace_factors;
```



	avg double precision
1	82.7736858193271

2) Get player_stats and team_stats used in the formula.

According to the formula of the ‘uPER’,

the stats needed for players are :

PTS,3P,AST,turnover,FG,FT,FGA,STL,PF,FTA,TRB,ORB,DRB,BLK.

The stats needed for teams are:

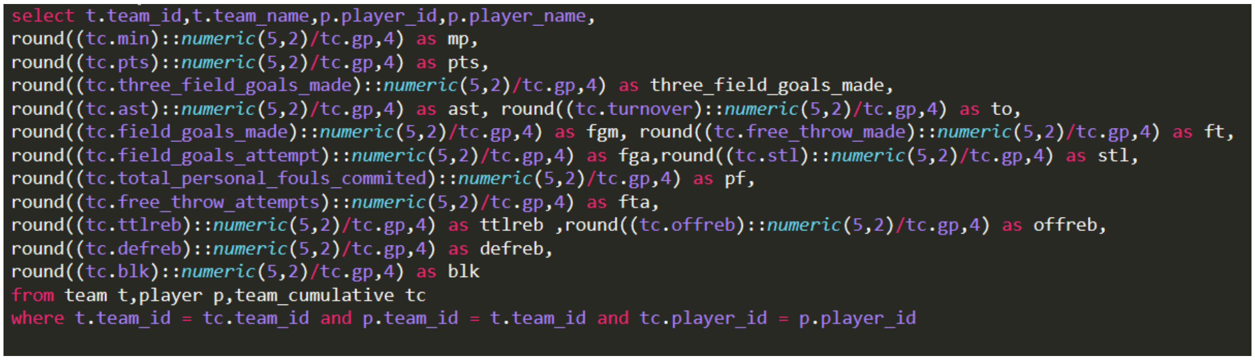
team_AST,team_FG

Since in our database”basketball”, we don’t have “average stats” of these metrics, we need to calculate those using queries.

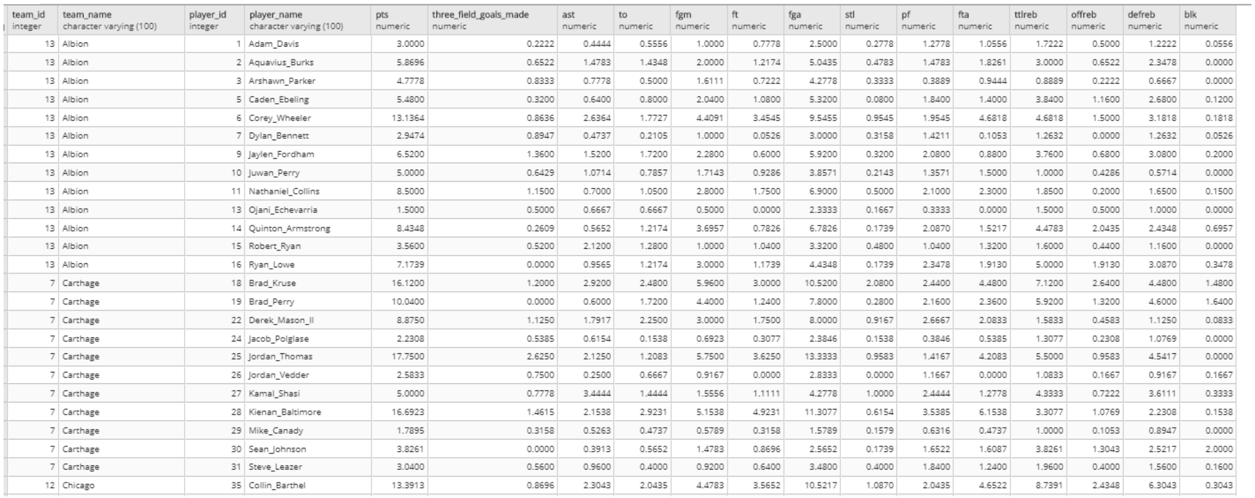
One thing to notice is that in our table ”team_cumulative”, we can calculate those metrics needed by dividing the existing metrics by the “game_played” column.

Another thing we need to notice is that since initially we created the fields types as “integers” for the most of our columns, we need to change it to float when we do averages.

Here is the view I created for player_raw_stats needed for calculating PER:



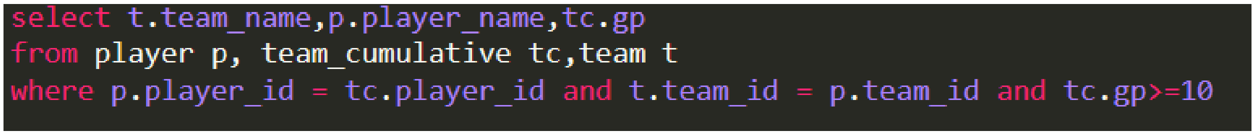
```
select t.team_id,t.team_name,p.player_id,p.player_name,
round((tc.min)::numeric(5,2)/tc.gp,4) as mp,
round((tc.pts)::numeric(5,2)/tc.gp,4) as pts,
round((tc.three_field_goals_made)::numeric(5,2)/tc.gp,4) as three_field_goals_made,
round((tc.ast)::numeric(5,2)/tc.gp,4) as ast, round((tc.turnover)::numeric(5,2)/tc.gp,4) as to,
round((tc.field_goals_made)::numeric(5,2)/tc.gp,4) as fgm, round((tc.free_throw_made)::numeric(5,2)/tc.gp,4) as ft,
round((tc.field_goals_attempt)::numeric(5,2)/tc.gp,4) as fga,round((tc.stl)::numeric(5,2)/tc.gp,4) as stl,
round((tc.total_personal_fouls_commited)::numeric(5,2)/tc.gp,4) as pf,
round((tc.free_throw_attempts)::numeric(5,2)/tc.gp,4) as fta,
round((tc.ttlreb)::numeric(5,2)/tc.gp,4) as ttlreb ,round((tc.offreb)::numeric(5,2)/tc.gp,4) as offreb,
round((tc.defreb)::numeric(5,2)/tc.gp,4) as defreb,
round((tc.blk)::numeric(5,2)/tc.gp,4) as blk
from team t,player p,team_cumulative tc
where t.team_id = tc.team_id and p.team_id = t.team_id and tc.player_id = p.player_id
```



team_id integer	team_name character varying (100)	player_id integer	player_name character varying (100)	pts numeric	three_field_goals_made numeric	ast numeric	to numeric	fgm numeric	ft numeric	fga numeric	stl numeric	pf numeric	fta numeric	ttlreb numeric	offreb numeric	defreb numeric	blk numeric
13	Albion	1	Adam_Davis	3.0000	0.2222	0.4444	0.5556	1.0000	0.7778	2.5000	0.2778	1.2778	1.0556	1.7222	0.5000	1.2222	0.0556
13	Albion	2	Aquavius_Burks	5.8696	0.6522	1.4783	1.4348	2.0000	1.2174	5.0435	0.4783	1.4783	1.8261	3.0000	0.6522	2.3478	0.0000
13	Albion	3	Arshawn_Parker	4.7778	0.8333	0.7778	0.5000	1.6111	0.7222	4.2778	0.3333	0.3889	0.9444	0.8889	0.2222	0.6667	0.0000
13	Albion	5	Caden_Ebeling	5.4800	0.3200	0.6400	0.8000	2.0400	1.0800	5.3200	0.0800	1.8400	1.4000	3.8400	1.1600	2.6800	0.1200
13	Albion	6	Corey_Wheeler	13.1364	0.8636	2.6364	1.7727	4.4091	3.4545	9.5455	0.9545	1.9545	4.6818	4.6818	1.5000	3.1818	0.1818
13	Albion	7	Dylan_Bennett	2.9474	0.8947	0.4737	0.2105	1.0000	0.0526	3.0000	0.3158	1.4211	0.1053	1.2632	0.0000	1.2632	0.0526
13	Albion	9	Jaylen_Fordham	6.5200	1.3600	1.5200	1.7200	2.2800	0.6000	5.9200	0.3200	2.0800	0.8800	3.7600	0.6800	3.0800	0.2000
13	Albion	10	Juwan_Perry	5.0000	0.6429	1.0714	0.7857	1.7143	0.9286	3.8571	0.2143	1.3571	1.5000	1.0000	0.4286	0.5714	0.0000
13	Albion	11	Nathaniel_Collins	8.5000	1.1500	0.7000	1.0500	2.8000	1.7500	6.9000	0.5000	2.1000	2.3000	1.8500	0.2000	1.6500	0.1500
13	Albion	13	Ojani_Echevarria	1.5000	0.5000	0.6667	0.6667	0.5000	0.0000	2.3333	0.1667	0.3333	0.0000	1.5000	0.5000	1.0000	0.0000
13	Albion	14	Quinton_Armstrong	8.4348	0.2609	0.5652	1.2174	3.6957	0.7826	6.7826	0.1739	2.0870	1.5217	4.4783	2.0435	2.4348	0.6957
13	Albion	15	Robert_Ryan	3.5600	0.5200	2.1200	1.2800	1.0000	1.0400	3.3200	0.4800	1.0400	1.3200	1.6000	0.4400	1.1600	0.0000
13	Albion	16	Ryan_Lowe	7.1739	0.0000	0.9565	1.2174	3.0000	1.1739	4.4348	0.1739	2.3478	1.9130	5.0000	1.9130	3.0870	0.3478
7	Carthage	18	Brad_Kruse	16.1200	1.2000	2.9200	2.4800	5.9600	3.0000	10.5200	2.0800	2.4400	4.4800	7.1200	2.6400	4.4800	1.4800
7	Carthage	19	Brad_Perry	10.0400	0.0000	0.6000	1.7200	4.4000	1.2400	7.8000	0.2800	2.1600	2.3600	5.9200	1.3200	4.6000	1.6400
7	Carthage	22	Derek_Mason_II	8.8750	1.1250	1.7917	2.2500	3.0000	1.7500	8.0000	0.9167	2.6667	2.0833	1.5833	0.4583	1.1250	0.0833
7	Carthage	24	Jacob_Polglase	2.2308	0.5385	0.6154	0.1538	0.6923	0.3077	2.3846	0.1538	0.3846	0.5385	1.3077	0.2308	1.0769	0.0000
7	Carthage	25	Jordan_Thomas	17.7500	2.6250	2.1250	1.2083	5.7500	3.6250	13.3333	0.9583	1.4167	4.2083	5.5000	0.9583	4.5417	0.0000
7	Carthage	26	Jordan_Vedder	2.5833	0.7500	0.2500	0.6667	0.9167	0.0000	2.8333	0.0000	1.1667	0.0000	1.0833	0.1667	0.9167	0.1667
7	Carthage	27	Kamal_Shasi	5.0000	0.7778	3.4444	1.4444	1.5556	1.1111	4.2778	1.0000	2.4444	1.2778	4.3333	0.7222	3.6111	0.3333
7	Carthage	28	Kienan_Baltimore	16.6923	1.4615	2.1538	2.9231	5.1538	4.9231	11.3077	0.6154	3.5385	6.1538	3.3077	1.0769	2.2308	0.1538
7	Carthage	29	Mike_Canady	1.7895	0.3158	0.5263	0.4737	0.5789	0.3158	1.5789	0.1579	0.6316	0.4737	1.0000	0.1053	0.8947	0.0000
7	Carthage	30	Sean_Johnson	3.8261	0.0000	0.3913	0.5652	1.4783	0.8696	2.5652	0.1739	1.6522	1.6087	3.8261	1.3043	2.5217	2.0000
7	Carthage	31	Steve_Leazer	3.0400	0.5600	0.9600	0.4000	0.9200	0.6400	3.4800	0.4000	1.8400	1.2400	1.9600	0.4000	1.5600	0.1600
12	Chicago	35	Collin_Barthel	13.3913	0.8696	2.3043	2.0435	4.4783	3.5652	10.5217	1.0870	2.0435	4.6522	8.7391	2.4348	6.3043	0.3043

3) Set a filter for the players included

I calculated the PER once and found some “end of bench” players can get an unexpected high PER since they only played 1 or 2 games and happened to player well in their limited minutes. So I think it is necessary to set a “filter” to rule those players out. I set the minimum number of game a player played as 10.Out of 321 players, I finally got 235 players.



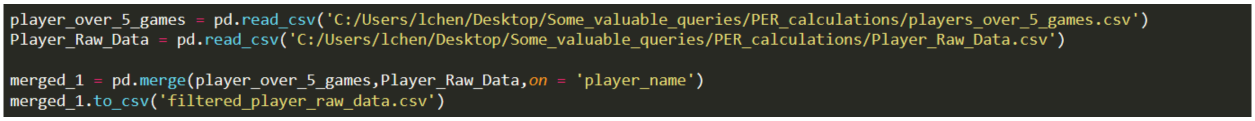
```sql
select t.team_name,p.player_name,tc.gp
from player p, team_cumulative tc,team t
where p.player_id = tc.player_id and t.team_id = p.team_id and tc.gp>=10
```

4) Merge “Filtered_Player_Raw_Stats” with”Player_Raw_Stats”and “Team_Pace_Factors”

Taking the results of 3.1, 3.2 and 3.3, now we need to “merge” those 3 so that it will facilitate us to calculate the PERs

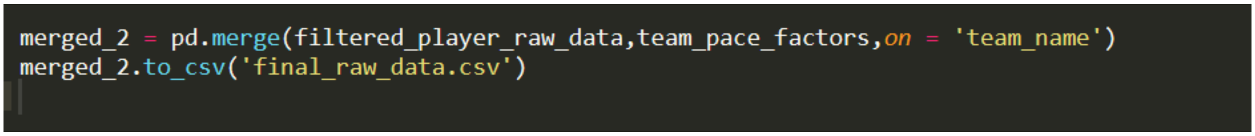
I used pandas module in Python to implement this step

First, merge “Player_Raw_Data” and “Player_Over_5_Games”



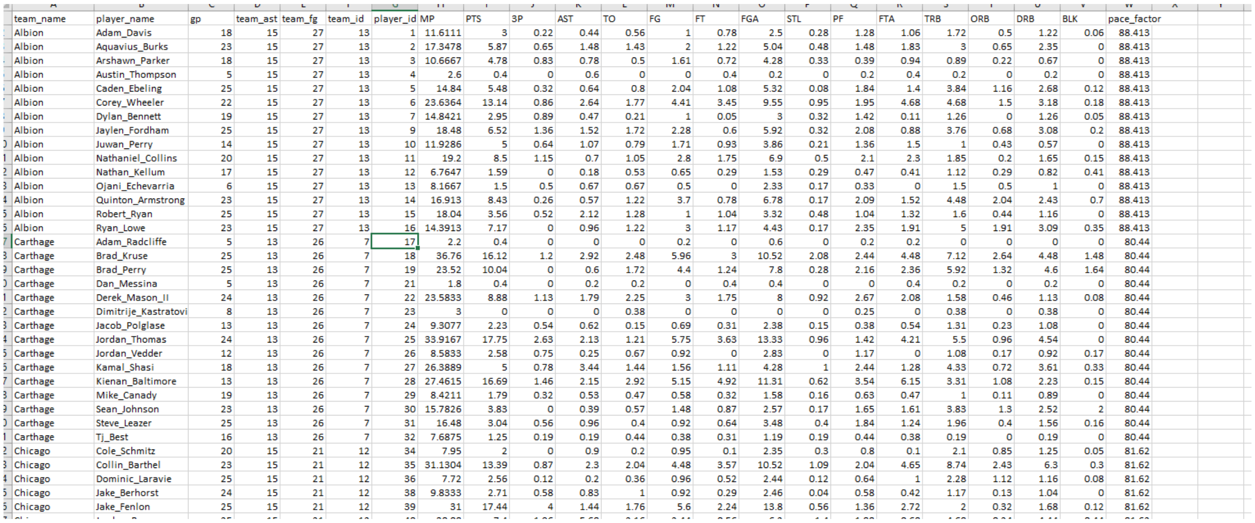
```python
player_over_5_games = pd.read_csv('C:/Users/lchen/Desktop/Some_valuable_queries/PER_calculations/players_over_5_games.csv')
Player_Raw_Data = pd.read_csv('C:/Users/lchen/Desktop/Some_valuable_queries/PER_calculations/Player_Raw_Data.csv')

merged_1 = pd.merge(player_over_5_games,Player_Raw_Data,on = 'player_name')
merged_1.to_csv('filtered_player_raw_data.csv')
```

Then, merge”filtered_player_raw_data” and “team_pace_factors”



```python
merged_2 = pd.merge(filtered_player_raw_data,team_pace_factors,on = 'team_name')
merged_2.to_csv('final_raw_data.csv')
```

Now, we are set to move to next step: Calculating PER “final raw data”



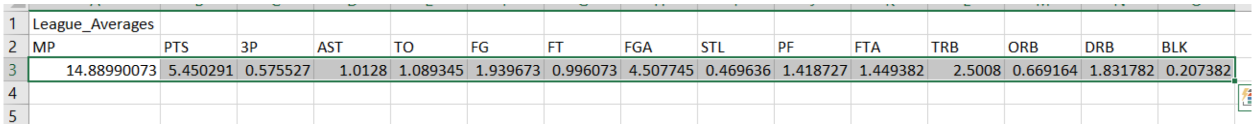
team_name	player_name	gp	team_ast	team_fg	team_id	player_id	MP	PTS	3P	AST	TO	FG	FT	FGA	STL	PF	FTA	TRB	ORB	DRB	BLK	pace_factor
Albion	Adam_Davis	18	15	27	13	1	11.6111	3	0.22	0.44	0.56	1	0.78	2.5	0.28	1.28	1.06	1.72	0.5	1.22	0.06	88.413
Albion	Aquavius_Burks	23	15	27	13	2	17.3478	5.87	0.65	1.48	1.43	2	1.22	5.04	0.48	1.48	1.83	3	0.65	2.35	0	88.413
Albion	Arshawn_Parker	18	15	27	13	3	10.6667	4.78	0.83	0.78	0.5	1.61	0.72	4.28	0.33	0.39	0.94	0.89	0.22	0.67	0	88.413
Albion	Austin_Thompson	5	15	27	13	4	2.6	0.4	0	0.6	0	0	0.4	0.2	0	0.2	0.4	0.2	0	0.2	0	88.413
Albion	Caden_Ebeling	25	15	27	13	5	14.84	5.48	0.32	0.64	0.8	2.04	1.08	5.32	0.08	1.84	1.4	3.84	1.16	2.68	0.12	88.413
Albion	Corey_Wheeler	22	15	27	13	6	23.6364	13.14	0.86	2.64	1.77	4.41	3.45	9.55	0.95	1.95	4.68	4.68	1.5	3.18	0.18	88.413
Albion	Dylan_Bennett	19	15	27	13	7	14.8421	2.95	0.89	0.47	0.21	1	0.05	3	0.32	1.42	0.11	1.26	0	1.26	0.05	88.413
Albion	Jaylen_Fordham	25	15	27	13	9	18.48	6.52	1.36	1.52	1.72	2.28	0.6	5.92	0.32	2.08	0.88	3.76	0.68	3.08	0.2	88.413
Albion	Juwan_Perry	14	15	27	13	10	11.9286	5	0.64	1.07	0.79	1.71	0.93	3.86	0.21	1.36	1.5	1	0.43	0.57	0	88.413
Albion	Nathaniel_Collins	20	15	27	13	11	19.2	8.5	1.15	0.7	1.05	2.8	1.75	6.9	0.5	2.1	2.3	1.85	0.2	1.65	0.15	88.413
Albion	Nathan_Kellum	17	15	27	13	12	6.7647	1.59	0	0.18	0.53	0.65	0.29	1.53	0.29	0.47	0.41	1.12	0.29	0.82	0.41	88.413
Albion	Ojani_Echevarria	6	15	27	13	13	8.1667	1.5	0.5	0.67	0.67	0.5	0	2.33	0.17	0.33	0	1.5	0.5	1	0	88.413
Albion	Quinton_Armstrong	23	15	27	13	14	16.913	8.43	0.26	0.57	1.22	3.7	0.78	6.78	0.17	2.09	1.52	4.48	2.04	2.43	0.7	88.413
Albion	Robert_Ryan	25	15	27	13	15	18.04	3.56	0.52	2.12	1.28	1	1.04	3.32	0.48	1.04	1.32	1.6	0.44	1.16	0	88.413
Albion	Ryan_Lowe	23	15	27	13	16	14.3913	7.17	0	0.96	1.22	3	1.17	4.43	0.17	2.35	1.91	5	1.91	3.09	0.35	88.413
Carthage	Adam_Radcliffe	5	13	26	7	17	2.2	0.4	0	0	0	0.2	0	0.6	0	0.2	0.2	0	0	0	0	80.44
Carthage	Brad_Kruse	25	13	26	7	18	36.76	16.12	1.2	2.92	2.48	5.96	3	10.52	2.08	2.44	4.48	7.12	2.64	4.48	1.48	80.44
Carthage	Brad_Perry	25	13	26	7	19	23.52	10.04	0	0.6	1.72	4.4	1.24	7.8	0.28	2.16	2.36	5.92	1.32	4.6	1.64	80.44
Carthage	Dan_Messina	5	13	26	7	21	1.8	0.4	0	0.2	0.2	0	0.4	0.4	0	0	0.4	0.2	0	0.2	0	80.44
Carthage	Derek_Mason_II	24	13	26	7	22	23.5833	8.88	1.13	1.79	2.25	3	1.75	8	0.92	2.67	2.08	1.58	0.46	1.13	0.08	80.44
Carthage	Dimitrije_Kastratovi	8	13	26	7	23	3	0	0	0	0.38	0	0	0	0	0.25	0	0.38	0	0.38	0	80.44
Carthage	Jacob_Polglase	13	13	26	7	24	9.3077	2.23	0.54	0.62	0.15	0.69	0.31	2.38	0.15	0.38	0.54	1.31	0.23	1.08	0	80.44
Carthage	Jordan_Thomas	24	13	26	7	25	33.9167	17.75	2.63	2.13	1.21	5.75	3.63	13.33	0.96	1.42	4.21	5.5	0.96	4.54	0	80.44
Carthage	Jordan_Vedder	12	13	26	7	26	8.5833	2.58	0.75	0.25	0.67	0.92	0	2.83	0	1.17	0	1.08	0.17	0.92	0.17	80.44
Carthage	Kamal_Shasi	18	13	26	7	27	26.3889	5	0.78	3.44	1.44	1.56	1.11	4.28	1	2.44	1.28	4.33	0.72	3.61	0.33	80.44
Carthage	Kienan_Baltimore	13	13	26	7	28	27.4615	16.69	1.46	2.15	2.92	5.15	4.92	11.31	0.62	3.54	6.15	3.31	1.08	2.23	0.15	80.44
Carthage	Mike_Canady	19	13	26	7	29	8.4211	1.79	0.32	0.53	0.47	0.58	0.32	1.58	0.16	0.63	0.47	1	0.11	0.89	0	80.44
Carthage	Sean_Johnson	23	13	26	7	30	15.7826	3.83	0	0.39	0.57	1.48	0.87	2.57	0.17	1.65	1.61	3.83	1.3	2.52	2	80.44
Carthage	Steve_Leazer	25	13	26	7	31	16.48	3.04	0.56	0.96	0.4	0.92	0.64	3.48	0.4	1.84	1.24	1.96	0.4	1.56	0.16	80.44
Carthage	Tj_Best	16	13	26	7	32	7.6875	1.25	0.19	0.19	0.44	0.38	0.31	1.19	0.19	0.44	0.38	0.19	0	0.19	0	80.44
Chicago	Cole_Schmitz	20	15	21	12	34	7.95	2	0	0.9	0.2	0.95	0.1	2.35	0.3	0.8	0.1	2.1	0.85	1.25	0.05	81.62
Chicago	Collin_Barthel	23	15	21	12	35	31.1304	13.39	0.87	2.3	2.04	4.48	3.57	10.52	1.09	2.04	4.65	8.74	2.43	6.3	0.3	81.62
Chicago	Dominic_Laravie	25	15	21	12	36	7.72	2.56	0.12	0.2	0.36	0.96	0.52	2.44	0.12	0.64	1	2.28	1.12	1.16	0.08	81.62
Chicago	Jake_Berhorst	24	15	21	12	38	9.8333	2.71	0.58	0.83	1	0.92	0.29	2.46	0.04	0.58	0.42	1.17	0.13	1.04	0	81.62
Chicago	Jake_Fenlon	25	15	21	12	39	31	17.44	4	1.44	1.76	5.6	2.24	13.8	0.56	1.36	2.72	2	0.32	1.68	0.12	81.62

5) Calculating PER

Now we have all the stats in a table called “final_raw_data”, now we can do our long-anticipated step: calculating the PER

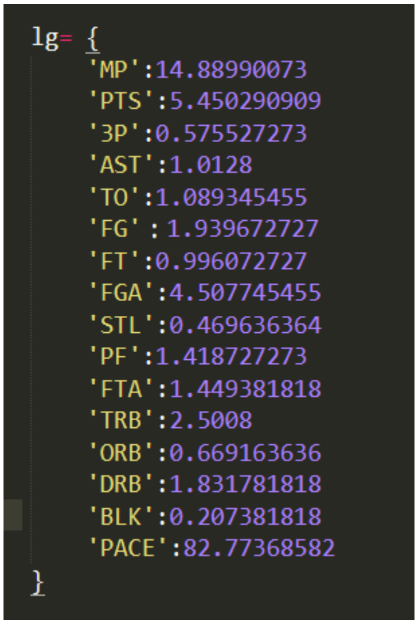
In this session, we continued to use the “Pandas” as tool to calculate the PER for each player.

First, in EXCEL sheet, we can get the average stats of the so-called “League”(includes 275 players).



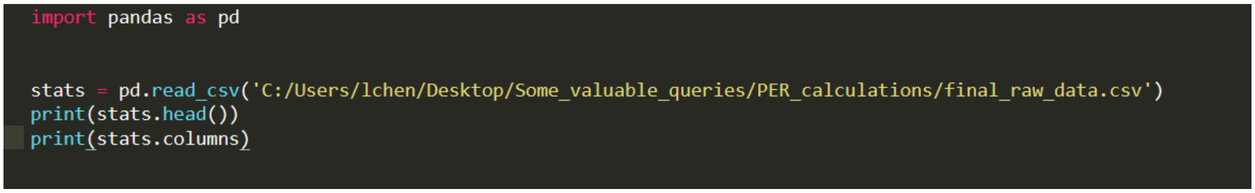
League_Averages														
MP	PTS	3P	AST	TO	FG	FT	FGA	STL	PF	FTA	TRB	ORB	DRB	BLK
14.88990073	5.450291	0.575527	1.0128	1.089345	1.939673	0.996073	4.507745	0.469636	1.418727	1.449382	2.5008	0.669164	1.831782	0.207382

Based on this, we create a dictionary called “lg” to store those average metrics.

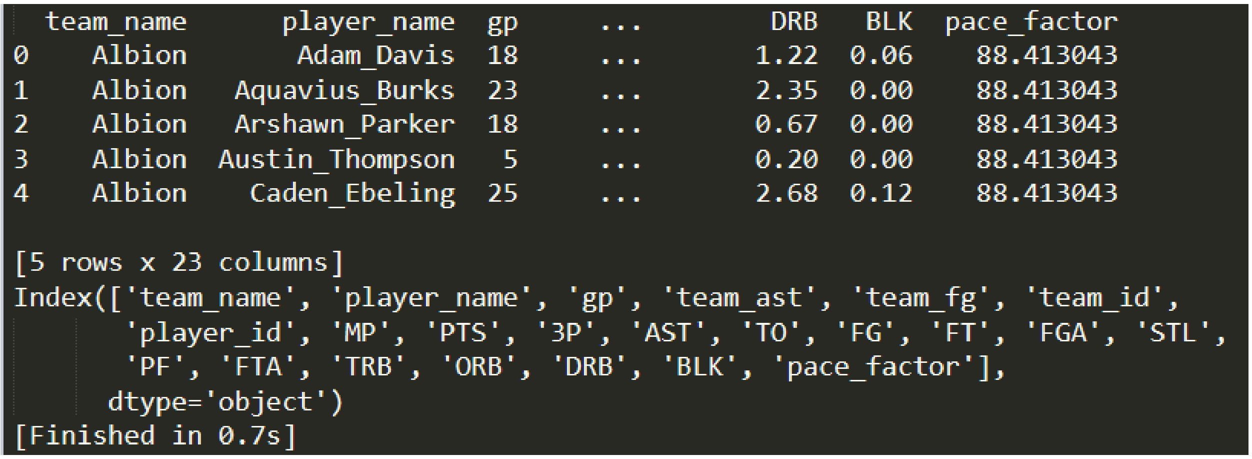


```
lg= {
    'MP':14.88990073
    'PTS':5.450290909
    '3P':0.575527273
    'AST':1.0128
    'TO':1.089345455
    'FG' : 1.939672727
    'FT':0.996072727
    'FGA':4.507745455
    'STL':0.469636364
    'PF':1.418727273
    'FTA':1.449381818
    'TRB':2.5008
    'ORB':0.669163636
    'DRB':1.831781818
    'BLK':0.207381818
    'PACE':82.77368582
}
```

The we import the “final_raw_data” :

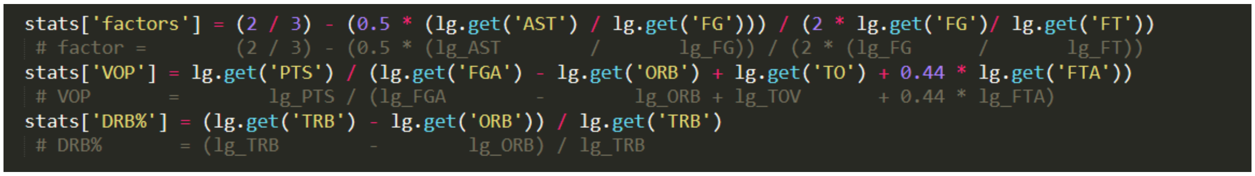


```
import pandas as pd


stats = pd.read_csv('C:/Users/lchen/Desktop/Some_valuable_queries/PER_calculations/final_raw_data.csv')
print(stats.head())
print(stats.columns)
```



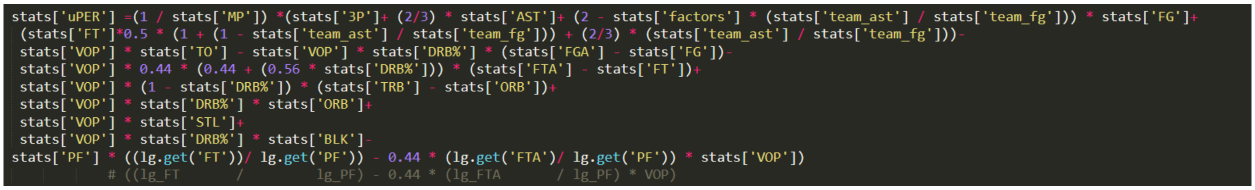
```
  team_name       player_name  gp      ...         DRB   BLK  pace_factor
0    Albion        Adam_Davis  18      ...        1.22  0.06    88.413043
1    Albion    Aquavius_Burks  23      ...        2.35  0.00    88.413043
2    Albion    Arshawn_Parker  18      ...        0.67  0.00    88.413043
3    Albion   Austin_Thompson   5      ...        0.20  0.00    88.413043
4    Albion     Caden_Ebeling  25      ...        2.68  0.12    88.413043

[5 rows x 23 columns]
Index(['team_name', 'player_name', 'gp', 'team_ast', 'team_fg', 'team_id',
       'player_id', 'MP', 'PTS', '3P', 'AST', 'TO', 'FG', 'FT', 'FGA', 'STL',
       'PF', 'FTA', 'TRB', 'ORB', 'DRB', 'BLK', 'pace_factor'],
      dtype='object')
[Finished in 0.7s]
```

Corresponding codes for 3 factors author introduced



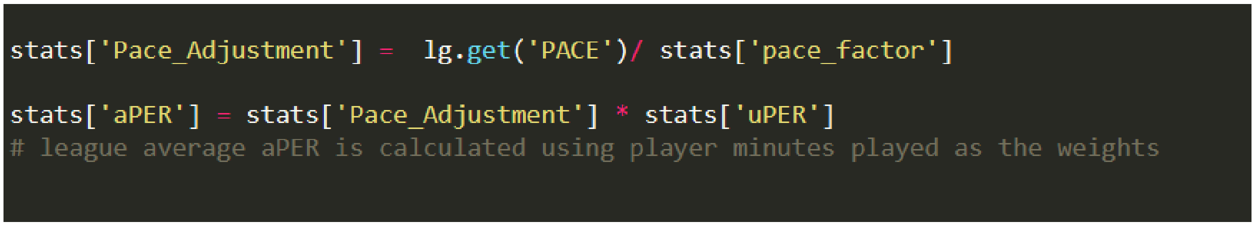
```
stats['factors'] = (2 / 3) - (0.5 * (lg.get('AST') / lg.get('FG'))) / (2 * lg.get('FG')/ lg.get('FT'))
 # factor =        (2 / 3) - (0.5 * (lg_AST        /        lg_FG)) / (2 * (lg_FG      /        lg_FT))
stats['VOP'] = lg.get('PTS') / (lg.get('FGA') - lg.get('ORB') + lg.get('TO') + 0.44 * lg.get('FTA'))
 # VOP        =        lg_PTS / (lg_FGA        -        lg_ORB + lg_TOV        + 0.44 * lg_FTA)
stats['DRB%'] = (lg.get('TRB') - lg.get('ORB')) / lg.get('TRB')
 # DRB%        = (lg_TRB        -        lg_ORB) / lg_TRB
```

Unadjusted PER:



```
stats['uPER'] =(1 / stats['MP']) *(stats['3P']+ (2/3) * stats['AST']+ (2 - stats['factors'] * (stats['team_ast'] / stats['team_fg'])) * stats['FG']+
 (stats['FT']*0.5 * (1 + (1 - stats['team_ast'] / stats['team_fg'])) + (2/3) * (stats['team_ast'] / stats['team_fg']))-
 stats['VOP'] * stats['TO'] - stats['VOP'] * stats['DRB%'] * (stats['FGA'] - stats['FG'])-
 stats['VOP'] * 0.44 * (0.44 + (0.56 * stats['DRB%'])) * (stats['FTA'] - stats['FT'])+
 stats['VOP'] * (1 - stats['DRB%']) * (stats['TRB'] - stats['ORB'])+
 stats['VOP'] * stats['DRB%'] * stats['ORB']+
 stats['VOP'] * stats['STL']+
 stats['VOP'] * stats['DRB%'] * stats['BLK']-
stats['PF'] * ((lg.get('FT'))/ lg.get('PF')) - 0.44 * (lg.get('FTA')/ lg.get('PF')) * stats['VOP'])
            # ((lg_FT       /         lg_PF) - 0.44 * (lg_FTA       / lg_PF) * VOP)
```

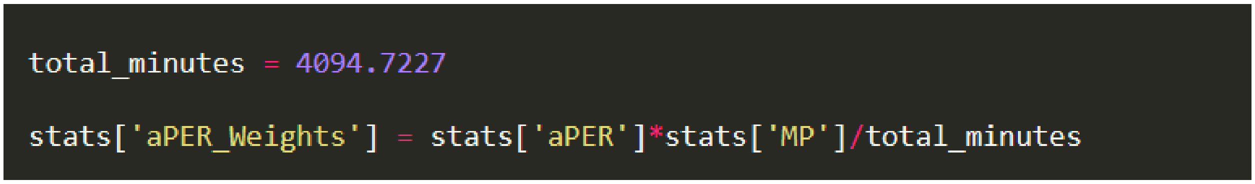
Pace adjustment:



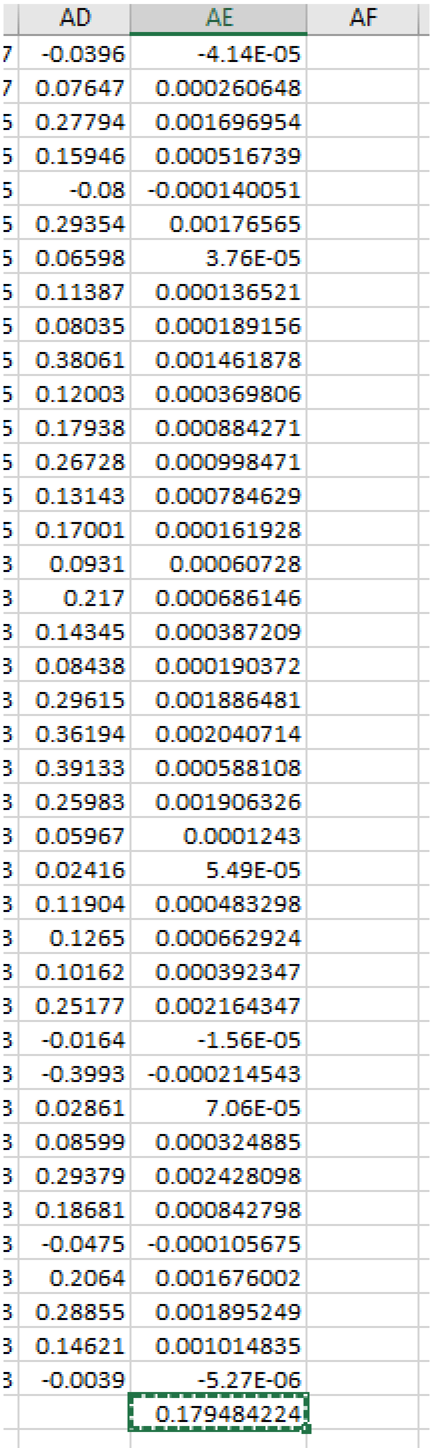
```
stats['Pace_Adjustment'] =  lg.get('PACE')/ stats['pace_factor']

stats['aPER'] = stats['Pace_Adjustment'] * stats['uPER']
# league average aPER is calculated using player minutes played as the weights
```

The final step is to calculate the lg_aPER which use player minutes played as the weights.

To do that I use this equation to get each players “contribution” to the lg_aPER and then sum them up in EXCEL sheet.

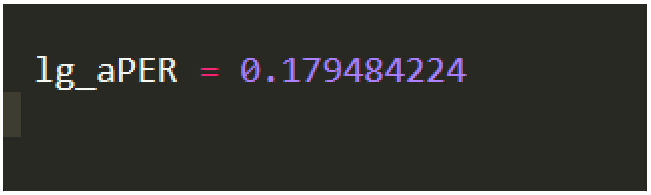


```
total_minutes = 4094.7227

stats['aPER_Weights'] = stats['aPER']*stats['MP']/total_minutes
```



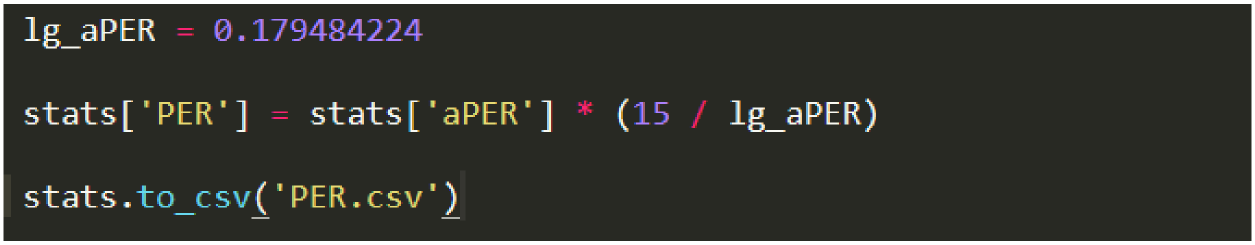
	AD	AE	AF
7	-0.0396	-4.14E-05	
7	0.07647	0.000260648	
5	0.27794	0.001696954	
5	0.15946	0.000516739	
5	-0.08	-0.000140051	
5	0.29354	0.00176565	
5	0.06598	3.76E-05	
5	0.11387	0.000136521	
5	0.08035	0.000189156	
5	0.38061	0.001461878	
5	0.12003	0.000369806	
5	0.17938	0.000884271	
5	0.26728	0.000998471	
5	0.13143	0.000784629	
5	0.17001	0.000161928	
3	0.0931	0.00060728	
3	0.217	0.000686146	
3	0.14345	0.000387209	
3	0.08438	0.000190372	
3	0.29615	0.001886481	
3	0.36194	0.002040714	
3	0.39133	0.000588108	
3	0.25983	0.001906326	
3	0.05967	0.0001243	
3	0.02416	5.49E-05	
3	0.11904	0.000483298	
3	0.1265	0.000662924	
3	0.10162	0.000392347	
3	0.25177	0.002164347	
3	-0.0164	-1.56E-05	
3	-0.3993	-0.000214543	
3	0.02861	7.06E-05	
3	0.08599	0.000324885	
3	0.29379	0.002428098	
3	0.18681	0.000842798	
3	-0.0475	-0.000105675	
3	0.2064	0.001676002	
3	0.28855	0.001895249	
3	0.14621	0.001014835	
3	-0.0039	-5.27E-06	
		0.179484224	

So we get lg_aPER is



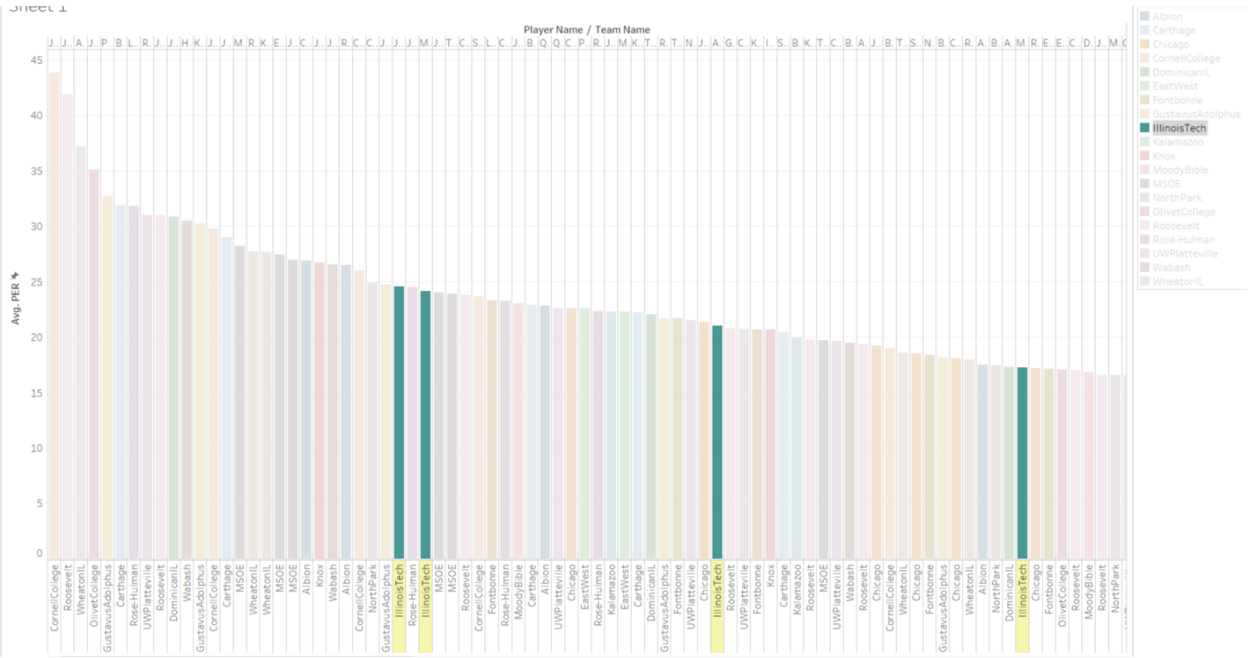
```
lg_aPER = 0.179484224
```

Finally, we get what we want!



```
lg_aPER = 0.179484224

stats['PER'] = stats['aPER'] * (15 / lg_aPER)

stats.to_csv('PER.csv')
```

Now we can manipulate the results in Tableau!



For more details, please refer to my GitHub.

3.3 Play - By - Play data

3.3.1 Introduction

In this section, we present the way we approached the “Play By Play” data, which is available in both IIT athletics website and Synergy website. It’s a log-like file that reports details of every phase of the match as it happens.

By processing the play-by-play data, we can get a lot of interesting data like Plus-Minus, Best Squads (in different aspects, we haven’t cover this yet), pair-wise synergy(haven’t covered yet) and more.

Since Synergy has much more information that IIT websites (we can grab other teams’ stats too!), we paid our attention more on the Synergy and tried to write Python codes to process every game’s play by play data.

Our program code is available in GitHub called “pbp_get_squads_stats.py”, we also created two classes called Player and Lineup in “BBall_Classes.py”, we call this functions in “using_pbp_get_squads_stats.py”. **Note that you need to change the starting lineup in “pbp_get_squads_stats.py” accordingly before you run the program.**

(Unfortunately, although we have total 19 games, but in 4 of them, the game’s play by play data was not right, so we have to omit those 4 games.)

3.3.2 From PBP to further

- ***Plus – Minus***

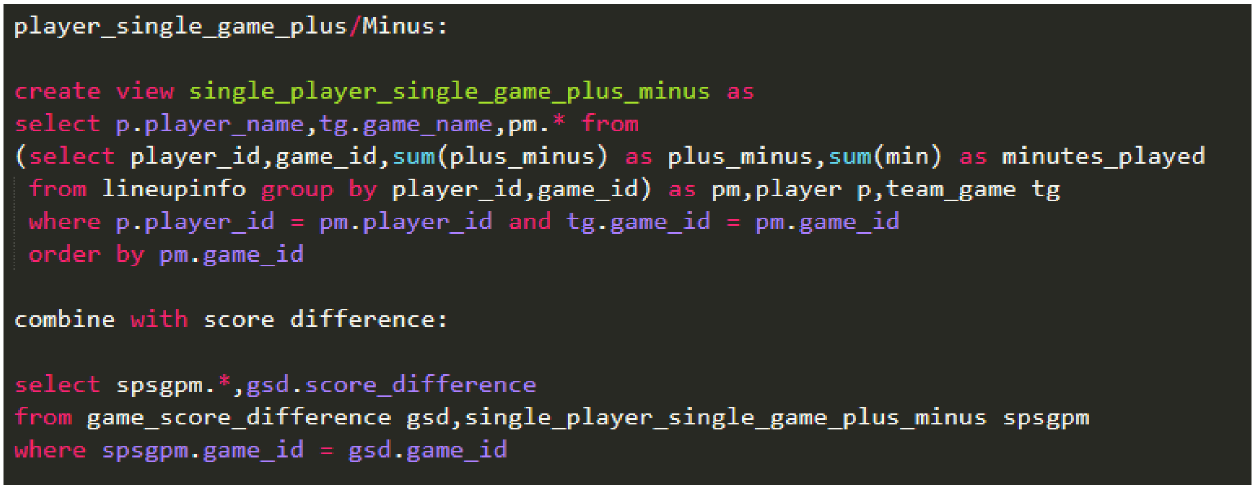
1) Plus- Minus is one of the most commonly used stats in basketball analytics world.

The formula is pretty simple: Plus Minus reflects how the team did while that player is on the court. If a player has a +5 PM, it means his team outscored the opponent by 5 points while he was on the court. If he has a -3, then the opposing team outscored his team by 3 points while he was on the court.

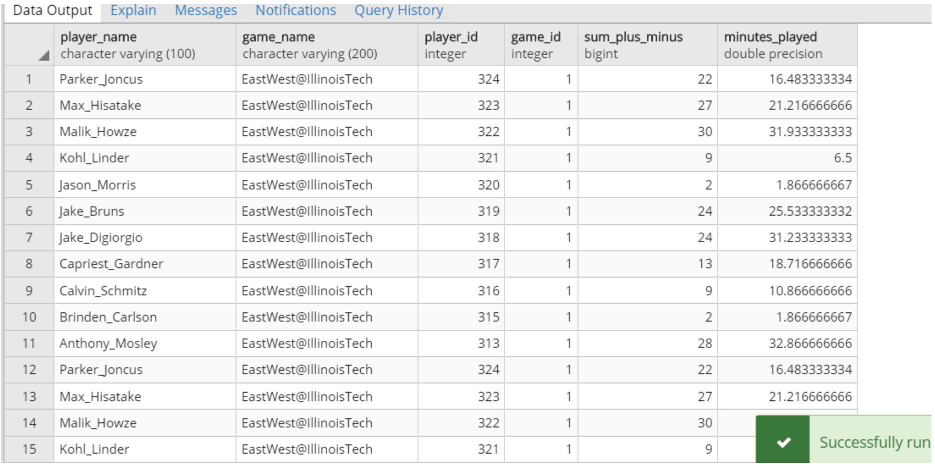
2) Calculation:

Since we can use our program directly get the “Lineup_Score” and “Oppo_Score” when certain player was on the court, we can just use **Plus/Minus = Lineup_Score - Oppo_Score** get the plus/minus, so we put this stats directly into our database.

To get the Plus/Minus for single_player_single_game, , combined with the view we created above and “score difference”(this view is not neccessary) view we created, Run the query below:



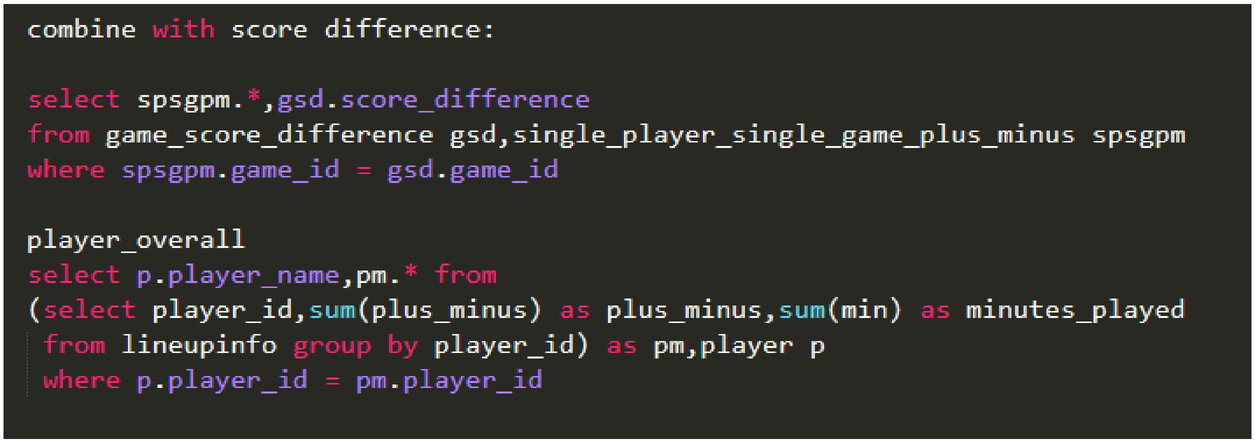
```
player_single_game_plus/Minus:

create view single_player_single_game_plus_minus as
select p.player_name,tg.game_name,pm.* from
(select player_id,game_id,sum(plus_minus) as plus_minus,sum(min) as minutes_played
 from lineupinfo group by player_id,game_id) as pm,player p,team_game tg
 where p.player_id = pm.player_id and tg.game_id = pm.game_id
 order by pm.game_id

combine with score difference:

select spsgpm.*,gsd.score_difference
from game_score_difference gsd,single_player_single_game_plus_minus spsgpm
where spsgpm.game_id = gsd.game_id
```



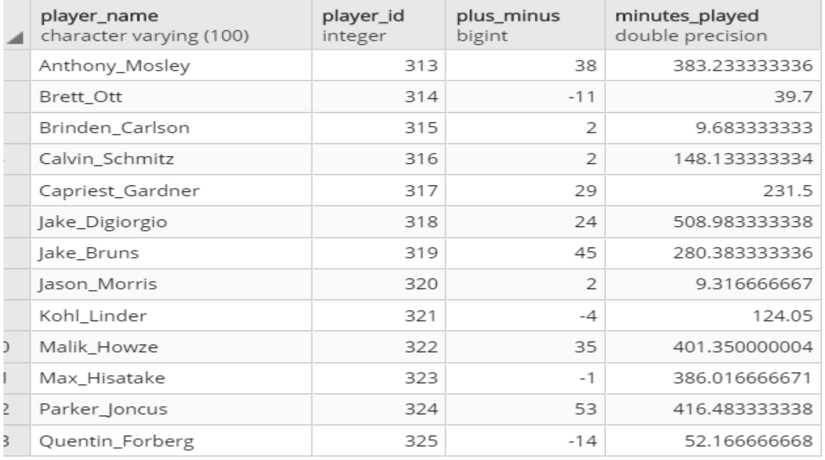
	player_name character varying (100)	game_name character varying (200)	player_id integer	game_id integer	sum_plus_minus bigint	minutes_played double precision
1	Parker_Joncus	EastWest@IllinoisTech	324	1	22	16.483333334
2	Max_Hisatake	EastWest@IllinoisTech	323	1	27	21.216666666
3	Malik_Howze	EastWest@IllinoisTech	322	1	30	31.933333333
4	Kohl_Linder	EastWest@IllinoisTech	321	1	9	6.5
5	Jason_Morris	EastWest@IllinoisTech	320	1	2	1.866666667
6	Jake_Bruns	EastWest@IllinoisTech	319	1	24	25.533333332
7	Jake_Digiorgio	EastWest@IllinoisTech	318	1	24	31.233333333
8	Capriest_Gardner	EastWest@IllinoisTech	317	1	13	18.716666666
9	Calvin_Schmitz	EastWest@IllinoisTech	316	1	9	10.866666666
10	Brinden_Carlson	EastWest@IllinoisTech	315	1	2	1.866666667
11	Anthony_Mosley	EastWest@IllinoisTech	313	1	28	32.866666666
12	Parker_Joncus	EastWest@IllinoisTech	324	1	22	16.483333334
13	Max_Hisatake	EastWest@IllinoisTech	323	1	27	21.216666666
14	Malik_Howze	EastWest@IllinoisTech	322	1	30	
15	Kohl_Linder	EastWest@IllinoisTech	321	1	9	

✓ Successfully run

To get the "cumulative plus minus"(season total) of each player, we can run the query below:



```
combine with score difference:

select spsgpm.*,gsd.score_difference
from game_score_difference gsd,single_player_single_game_plus_minus spsgpm
where spsgpm.game_id = gsd.game_id

player_overall
select p.player_name,pm.* from
(select player_id,sum(plus_minus) as plus_minus,sum(min) as minutes_played
 from lineupinfo group by player_id) as pm,player p
 where p.player_id = pm.player_id
```



player_name character varying (100)	player_id integer	plus_minus bigint	minutes_played double precision
Anthony_Mosley	313	38	383.233333336
Brett_Ott	314	-11	39.7
Brinden_Carlson	315	2	9.683333333
Calvin_Schmitz	316	2	148.133333334
Capriest_Gardner	317	29	231.5
Jake_Digiorgio	318	24	508.983333338
Jake_Bruns	319	45	280.383333336
Jason_Morris	320	2	9.316666667
Kohl_Linder	321	-4	124.05
Malik_Howze	322	35	401.350000004
Max_Hisatake	323	-1	386.016666671
Parker_Joncus	324	53	416.483333338
Quentin_Forberg	325	-14	52.166666668

Visualizations of the above two query results are available in my Tableau Profile:

Season Player Cumulative PM:

https://public.tableau.com/profile/chenjie.li#!/vizhome/SeasonPlayerCumulativePlusMinus/Dashboard1

Single Game Single Player PM:

https://public.tableau.com/profile/chenjie.li#!/vizhome/SingleGamePlusMinusSinglePlayer/Sheet1

- ***Lineup – Plus Minus***

We are working on it….

- ***"Best Five" in different categories***

We are working on it……

4.Conclusions and Implications

Since we had more detailed and advanced stats at hand this time, we could be able to get some stats more detailed compared with Larry and Denis, but some of their work we didn't cover such as game simulations because of the limited time.

We believe by having this pretty mature database in hand, we will have a higher starting point in the future and do more detailed, valuable analysis for Coach Kelly and Basketball staff team.

As data science field grows faster and faster, the analysis of games are also becoming more and more important in the contest. We really hope some day not only school basketball team, but in all of the sports departments, we can have our own "Sports Analytics" team some day.

Let's go Scarlet Hawks!