PER Calculation

1.Introduction

2.Formula

3.Practice

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

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.)

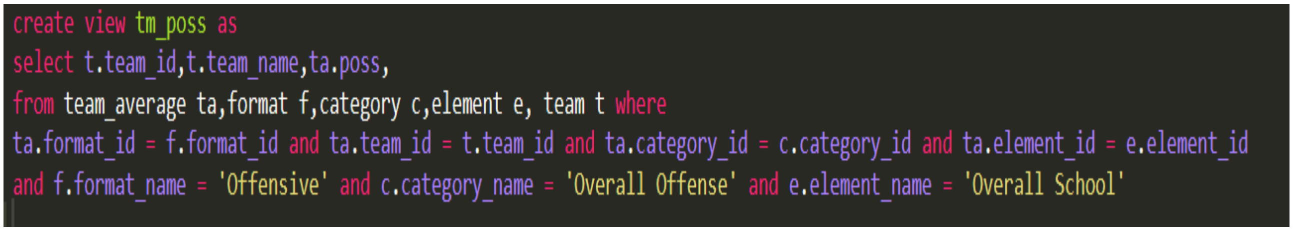
3.Practice

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”.

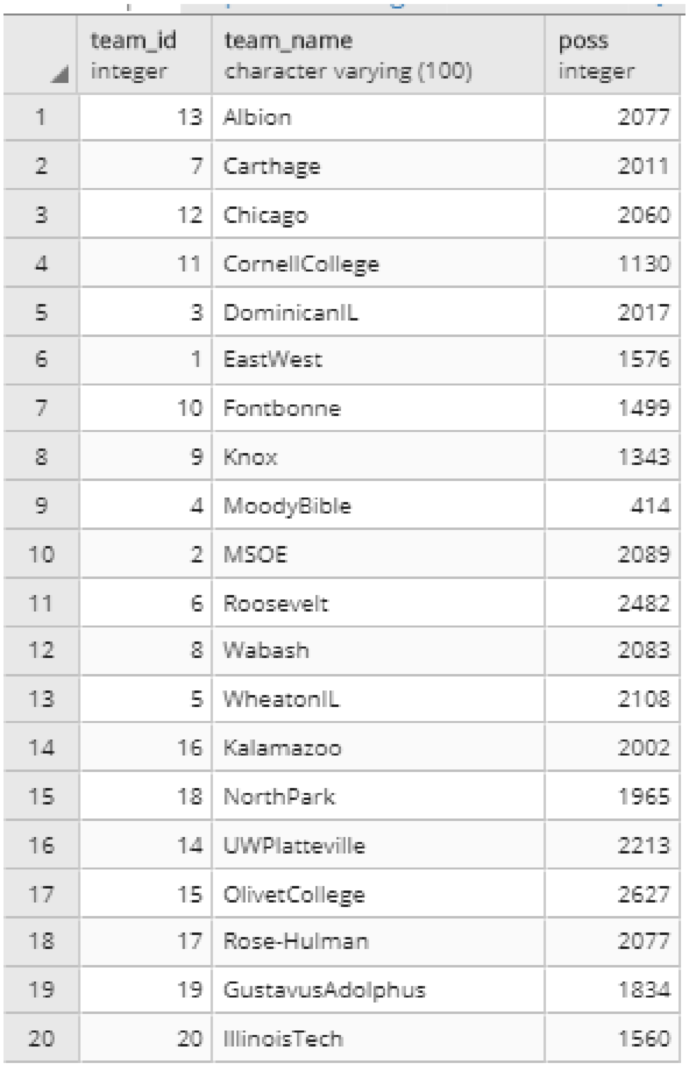
One thing to note is that since the formula includes the “league” information and IIT doesn’t have a formal kind of league, so I just created a so-called league which includes the teams that IIT played against in the last season.

3.1Get Pace factors

1)Get Tm_Poss

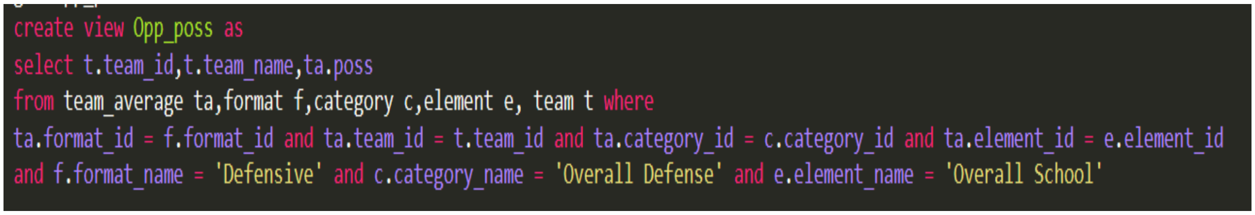


```
create view tm_poss as
select t.team_id,t.team_name,ta.poss,
from team_average ta,format f,category c,element e, team t where
ta.format_id = f.format_id and ta.team_id = t.team_id and ta.category_id = c.category_id and ta.element_id = e.element_id
and f.format_name = 'Offensive' and c.category_name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

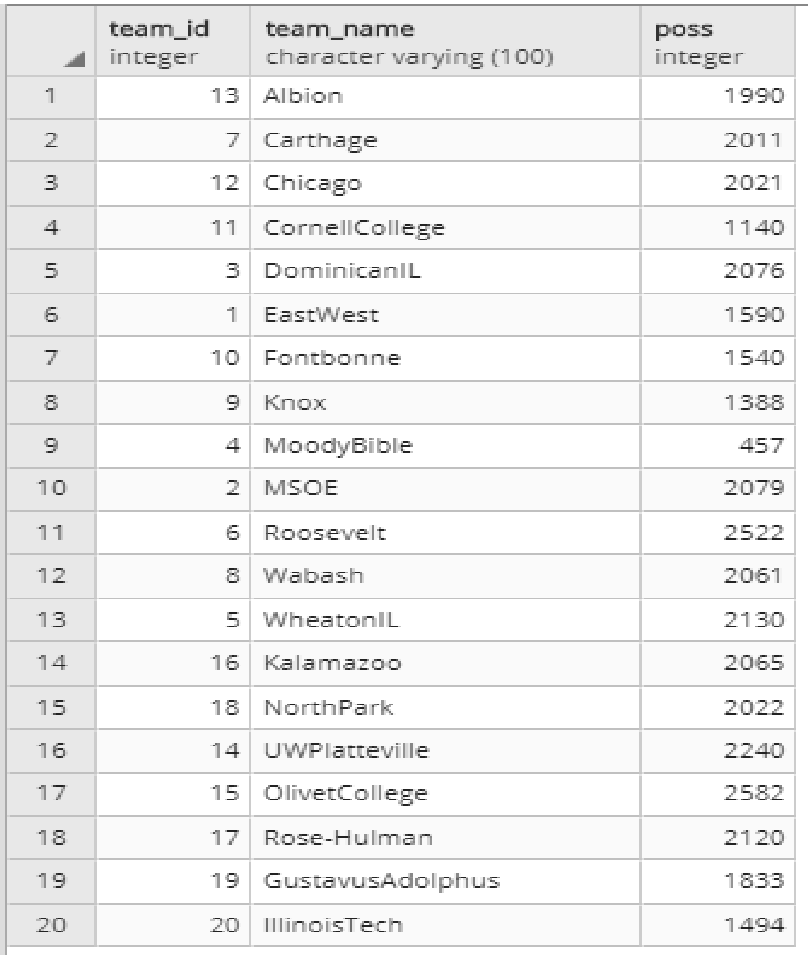


	team_id integer	team_name character varying (100)	poss integer
1	13	Albion	2077
2	7	Carthage	2011
3	12	Chicago	2060
4	11	CornellCollege	1130
5	3	DominicanIL	2017
6	1	EastWest	1576
7	10	Fontbonne	1499
8	9	Knox	1343
9	4	MoodyBible	414
10	2	MSOE	2089
11	6	Roosevelt	2482
12	8	Wabash	2083
13	5	WheatonIL	2108
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

2)Get Opp_Poss

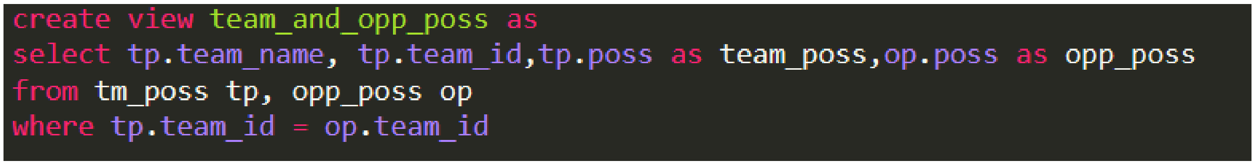


```
create view Opp_poss as
select t.team_id,t.team_name,ta.poss
from team_average ta,format f,category c,element e, team t where
ta.format_id = f.format_id and ta.team_id = t.team_id and ta.category_id = c.category_id and ta.element_id = e.element_id
and f.format_name = 'Defensive' and c.category_name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

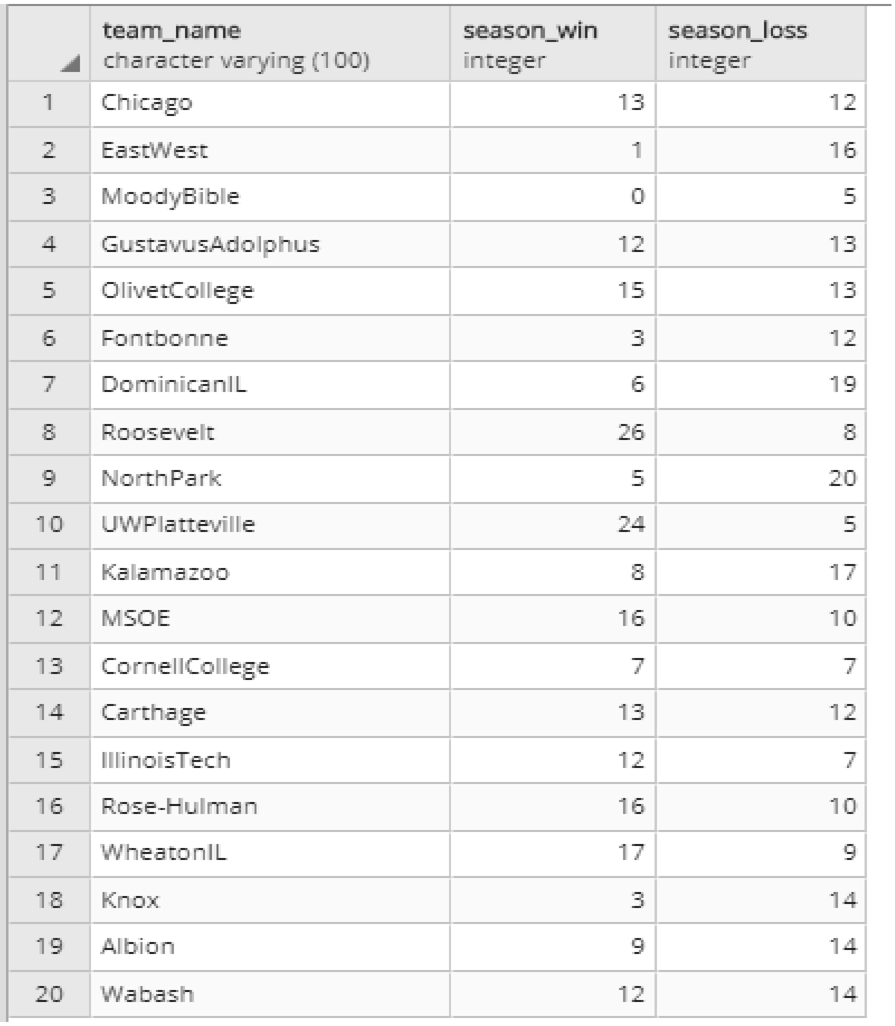


	team_id	team_name	poss
	integer	character varying (100)	integer
1	13	Albion	1990
2	7	Carthage	2011
3	12	Chicago	2021
4	11	CornellCollege	1140
5	3	DominicanIL	2076
6	1	EastWest	1590
7	10	Fontbonne	1540
8	9	Knox	1388
9	4	MoodyBible	457
10	2	MSOE	2079
11	6	Roosevelt	2522
12	8	Wabash	2061
13	5	WheatonIL	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

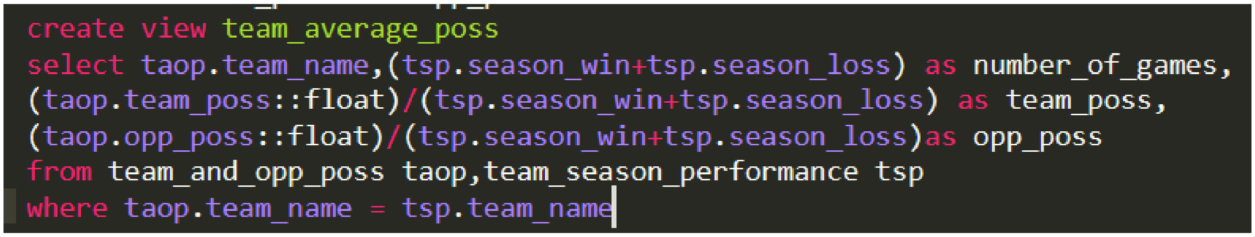


```
create view team_and_opp_poss as
select tp.team_name, tp.team_id,tp.poss as team_poss,op.poss as opp_poss
from tm_poss tp, opp_poss op
where tp.team_id = op.team_id
```

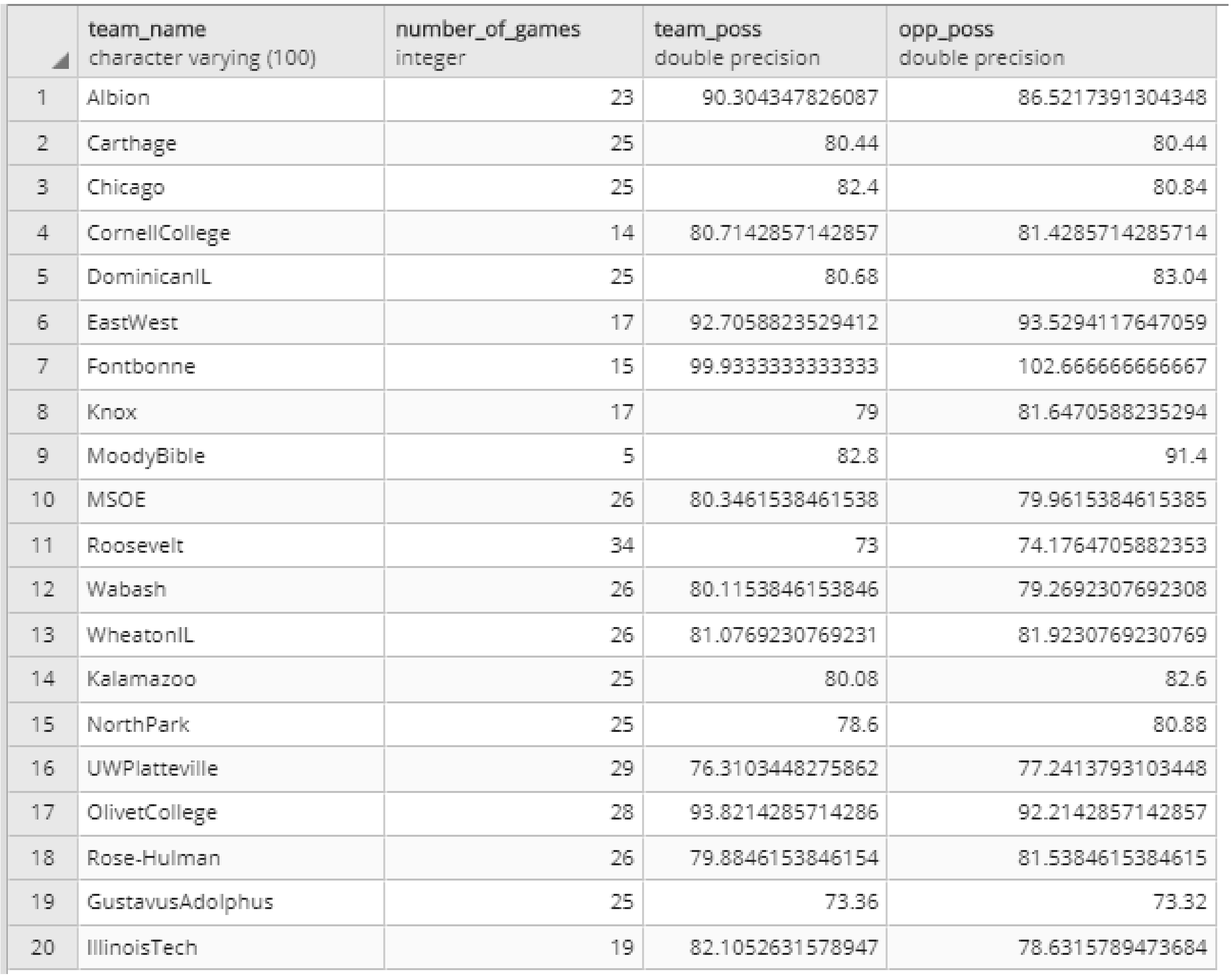


	team_name character varying (100)	season_win integer	season_loss integer
1	Chicago	13	12
2	EastWest	1	16
3	MoodyBible	0	5
4	GustavusAdolphus	12	13
5	OlivetCollege	15	13
6	Fontbonne	3	12
7	DominicanIL	6	19
8	Roosevelt	26	8
9	NorthPark	5	20
10	UWPlatteville	24	5
11	Kalamazoo	8	17
12	MSOE	16	10
13	CornellCollege	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

4) Calculate average team_poss and average opp_poss



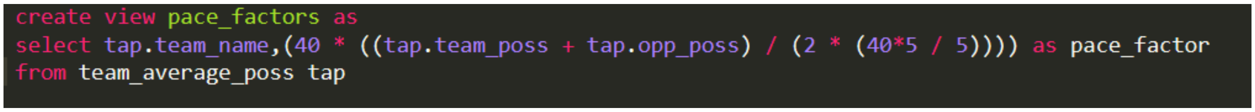
```
create view team_average_poss
select taop.team_name,(tsp.season_win+tsp.season_loss) as number_of_games,
(taop.team_poss::float)/(tsp.season_win+tsp.season_loss) as team_poss,
(taop.opp_poss::float)/(tsp.season_win+tsp.season_loss)as opp_poss
from team_and_opp_poss taop,team_season_performance tsp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



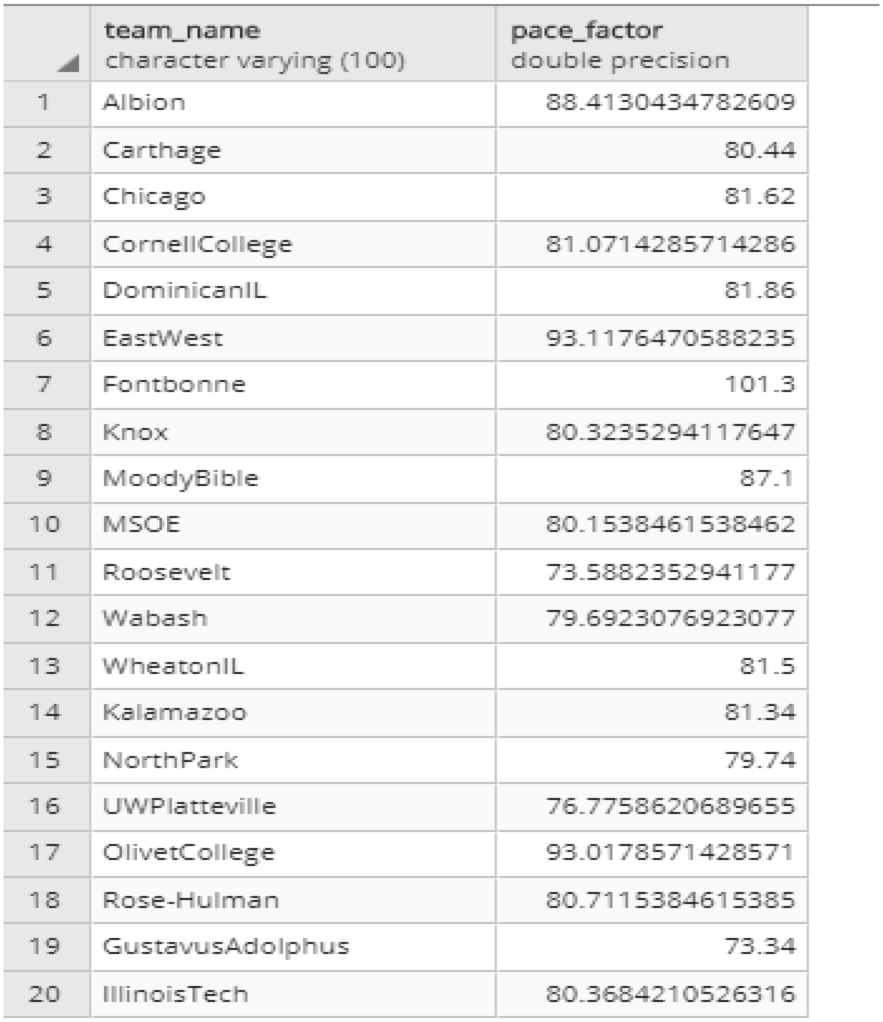
	team_name character varying (100)	number_of_games integer	team_poss double precision	opp_poss double precision
1	Albion	23	90.304347826087	86.5217391304348
2	Carthage	25	80.44	80.44
3	Chicago	25	82.4	80.84
4	CornellCollege	14	80.7142857142857	81.4285714285714
5	DominicanIL	25	80.68	83.04
6	EastWest	17	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

5)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)

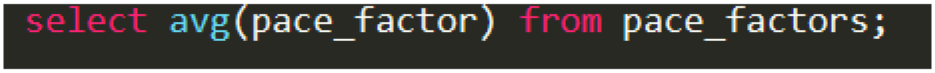


```
create view pace_factors as
select tap.team_name,(40 * ((tap.team_poss + tap.opp_poss) / (2 * (40*5 / 5)))) as pace_factor
from team_average_poss tap
```

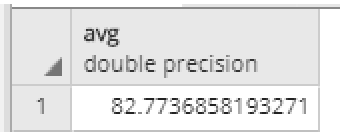


	team_name character varying (100)	pace_factor double precision
1	Albion	88.4130434782609
2	Carthage	80.44
3	Chicago	81.62
4	CornellCollege	81.0714285714286
5	DominicanIL	81.86
6	EastWest	93.1176470588235
7	Fontbonne	101.3
8	Knox	80.3235294117647
9	MoodyBible	87.1
10	MSOE	80.1538461538462
11	Roosevelt	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



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



	avg double precision
1	82.7736858193271

3.2 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:

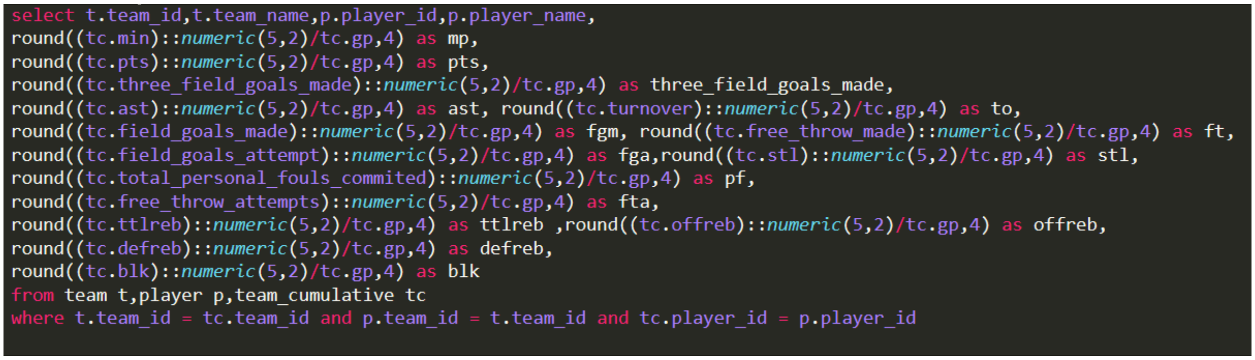
team_AST,team_FG

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

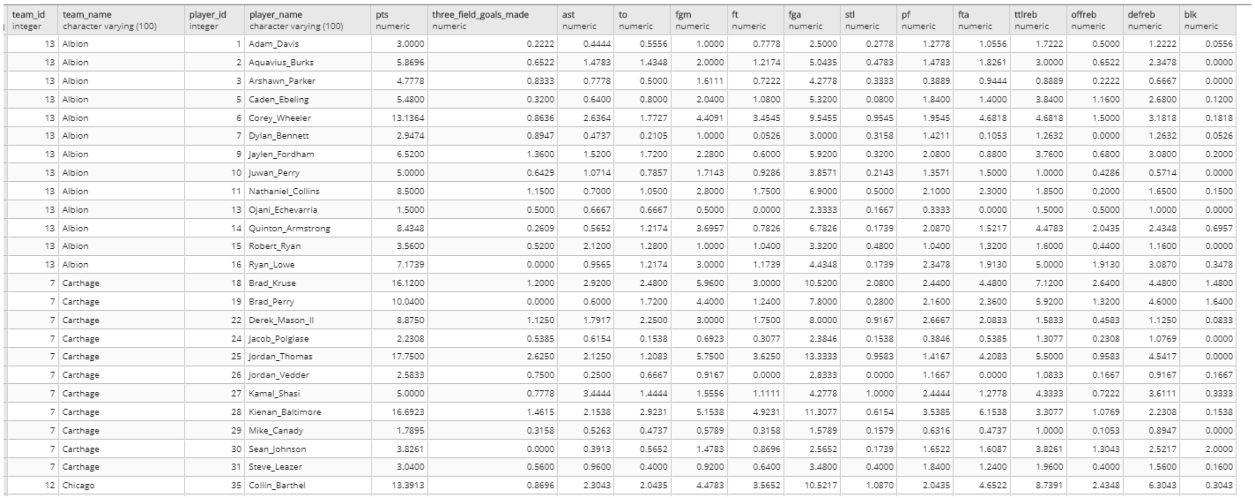
One thing to notice is that in our table ”team_cumulative”, we can calculate those metrics needed by dividing the existing metrics by the “game_played” column.

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:



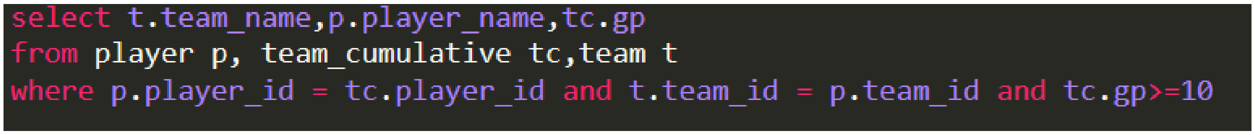
```
select t.team_id,t.team_name,p.player_id,p.player_name,
round((tc.min)::numeric(5,2)/tc.gp,4) as mp,
round((tc.pts)::numeric(5,2)/tc.gp,4) as pts,
round((tc.three_field_goals_made)::numeric(5,2)/tc.gp,4) as three_field_goals_made,
round((tc.ast)::numeric(5,2)/tc.gp,4) as ast, round((tc.turnover)::numeric(5,2)/tc.gp,4) 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	team_name	player_id	player_name	pts	three_field_goals_made	ast	to	fgm	ft	fga	stl	pf	fta	ttlreb	offreb	defreb	blk
integer	character varying (100)	integer	character varying (100)	numeric	numeric	numeric	numeric	numeric	numeric	numeric	numeric	numeric	numeric	numeric	numeric	numeric	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

3.3 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.



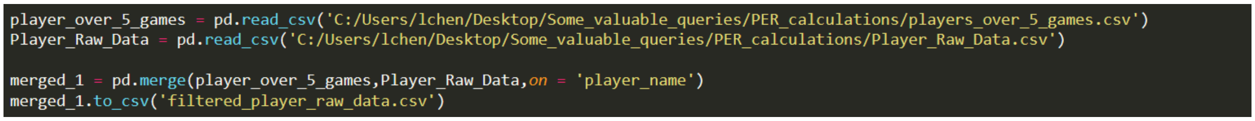
```
select t.team_name,p.player_name,tc.gp
from player p, team_cumulative tc,team t
where p.player_id = tc.player_id and t.team_id = p.team_id and tc.gp>=10
```

3.4 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

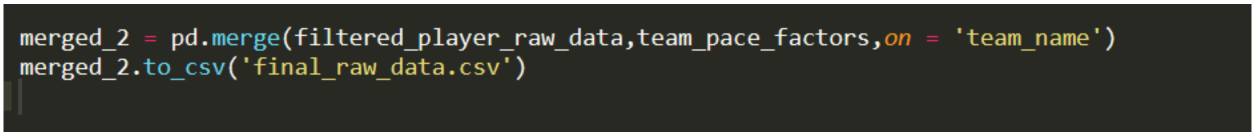
I used pandas module in Python to implement this step

First, merge “Player_Raw_Data” and “Player_Over_5_Games”



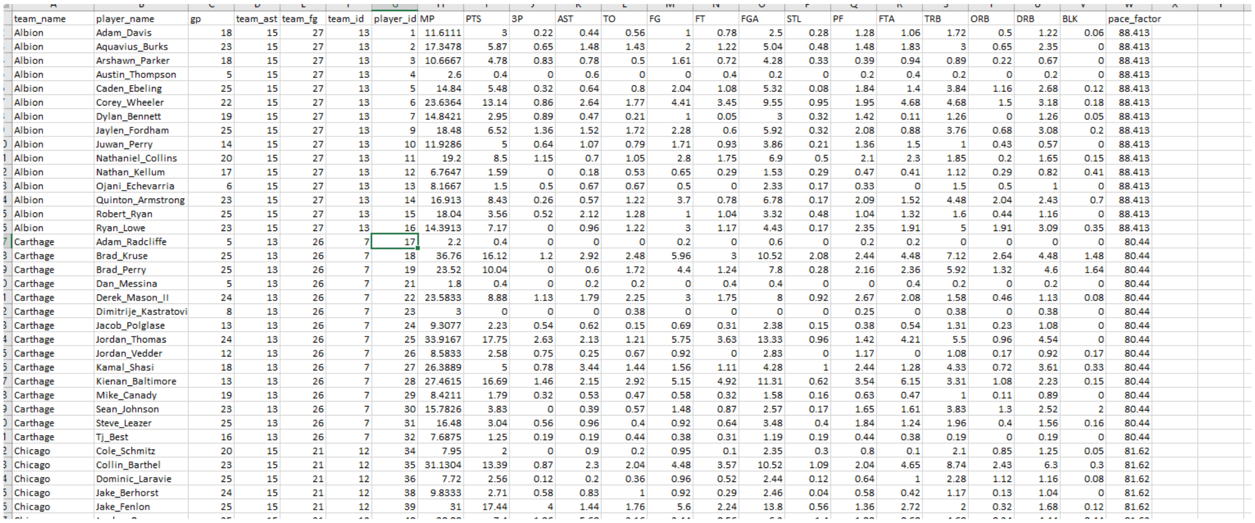
```
player_over_5_games = pd.read_csv('C:/Users/lchen/Desktop/Some_valuable_queries/PER_calculations/players_over_5_games.csv')
Player_Raw_Data = pd.read_csv('C:/Users/lchen/Desktop/Some_valuable_queries/PER_calculations/Player_Raw_Data.csv')

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”



```
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”



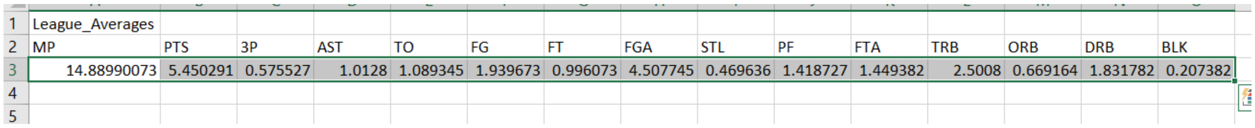
team_name	player_name	gp	team_ast	team_fg	team_id	player_id	MP	PTS	3P	AST	TO	FG	FT	FGA	STL	PF	FTA	TRB	ORB	DRB	BLK	pace_factor
Albion	Adam_Davis	18	15	27	13	1	11.6111	3	0.22	0.44	0.56	1	0.78	2.5	0.28	1.28	1.06	1.72	0.5	1.22	0.06	88.413
Albion	Aquavius_Burks	23	15	27	13	2	17.3478	5.87	0.65	1.48	1.43	2	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

3.5 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

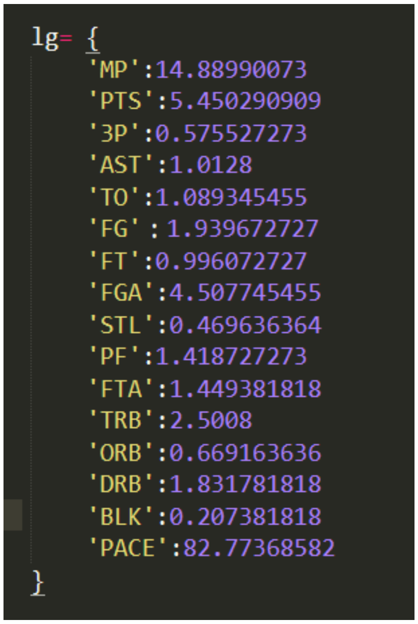
In this session, we continued to use the “Pandas” as tool to calculate the PER for each player.

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



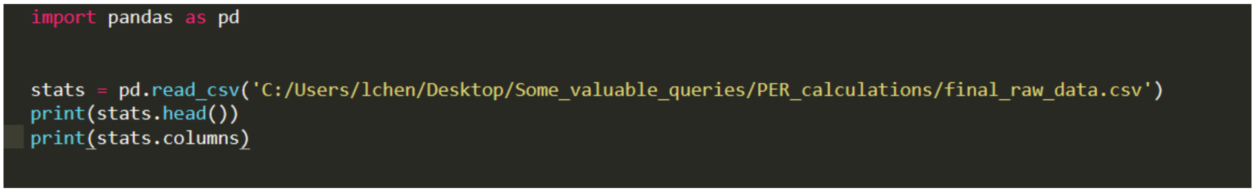
League_Averages														
MP	PTS	3P	AST	TO	FG	FT	FGA	STL	PF	FTA	TRB	ORB	DRB	BLK
14.88990073	5.450291	0.575527	1.0128	1.089345	1.939673	0.996073	4.507745	0.469636	1.418727	1.449382	2.5008	0.669164	1.831782	0.207382

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

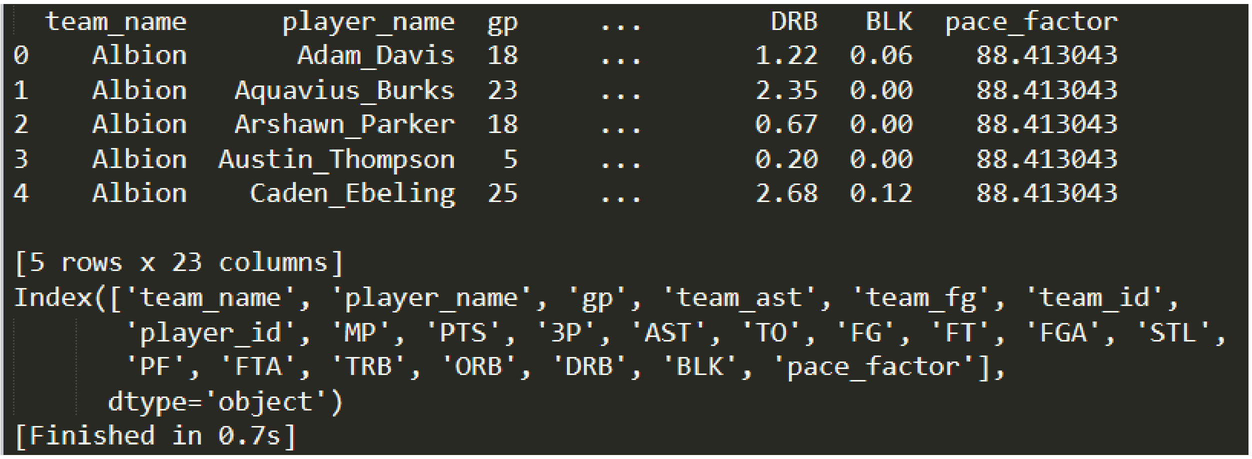


```
lg= {
    'MP':14.88990073
    'PTS':5.450290909
    '3P':0.575527273
    'AST':1.0128
    'TO':1.089345455
    'FG' : 1.939672727
    'FT':0.996072727
    'FGA':4.507745455
    'STL':0.469636364
    'PF':1.418727273
    'FTA':1.449381818
    'TRB':2.5008
    'ORB':0.669163636
    'DRB':1.831781818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

The we import the “final_raw_data” :

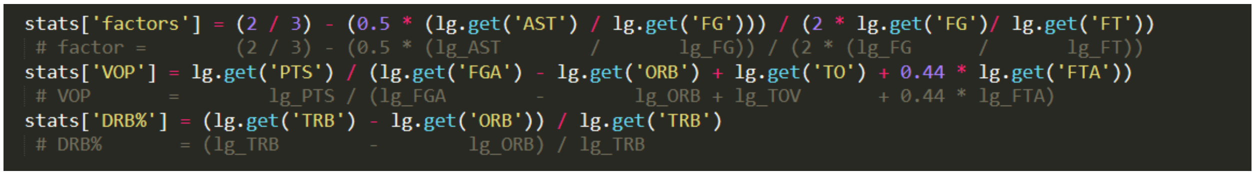


```
import pandas as pd


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



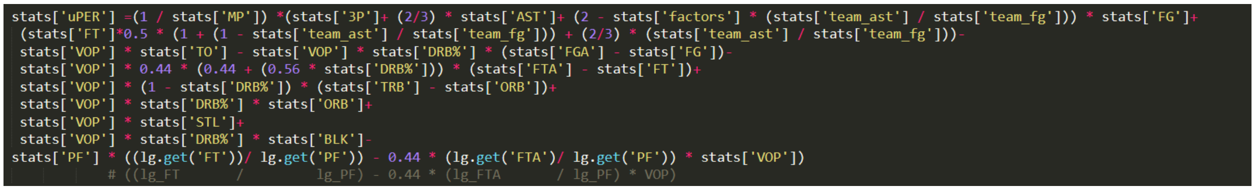
```
  team_name       player_name  gp      ...         DRB   BLK  pace_factor
0    Albion        Adam_Davis  18      ...        1.22  0.06    88.413043
1    Albion     Aquavius_Burks  23      ...        2.35  0.00    88.413043
2    Albion    Arshawn_Parker  18      ...        0.67  0.00    88.413043
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



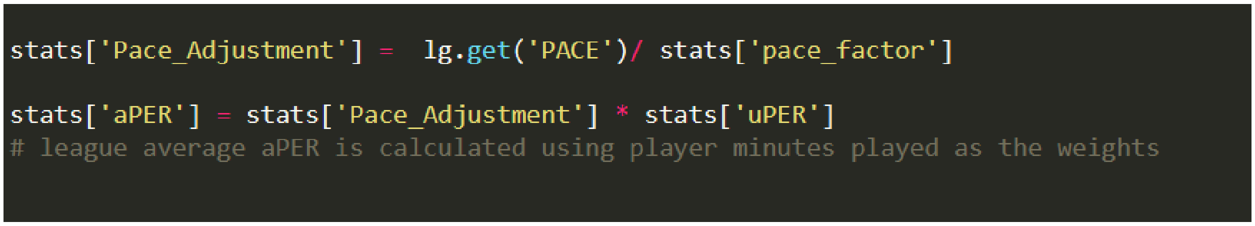
```
stats['factors'] = (2 / 3) - (0.5 * (lg.get('AST') / lg.get('FG'))) / (2 * lg.get('FG')/ lg.get('FT'))
 # factor =        (2 / 3) - (0.5 * (lg_AST        /       lg_FG)) / (2 * (lg_FG       /        lg_FT))
stats['VOP'] = lg.get('PTS') / (lg.get('FGA') - lg.get('ORB') + lg.get('TO') + 0.44 * lg.get('FTA'))
 # 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)
```

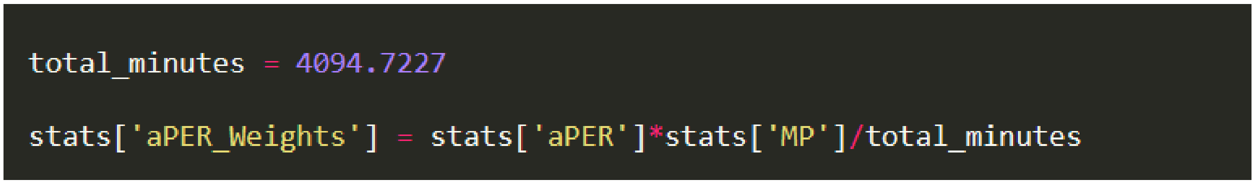
Pace adjustment:



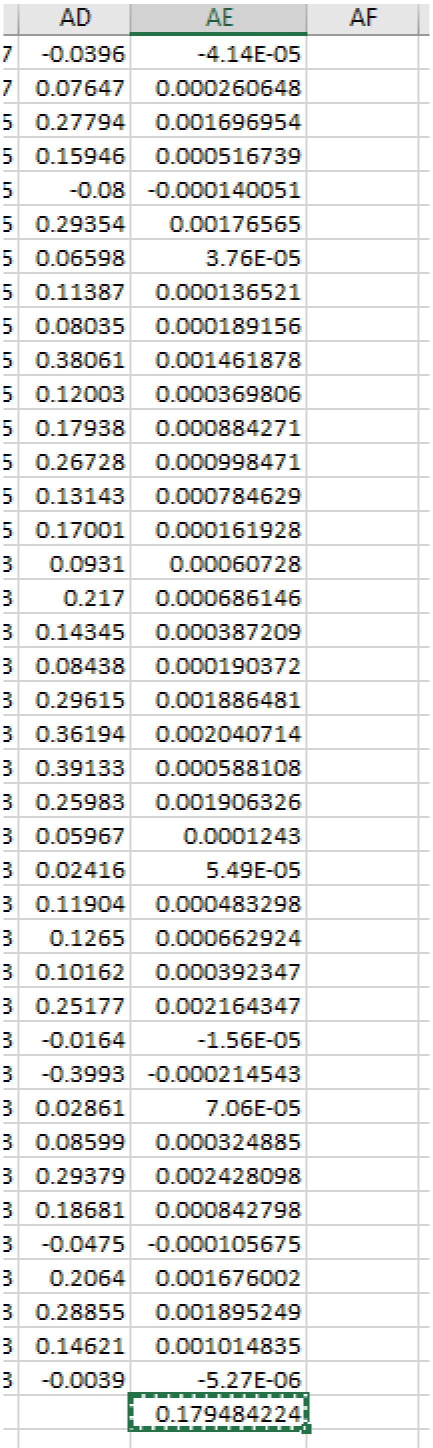
```
stats['Pace_Adjustment'] =  lg.get('PACE')/ stats['pace_factor']

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.

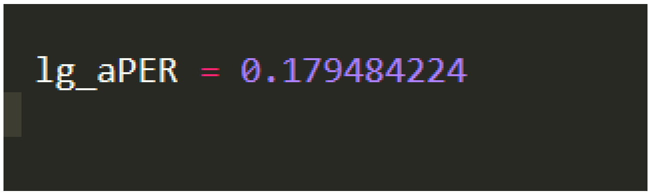


```
total_minutes = 4094.7227

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



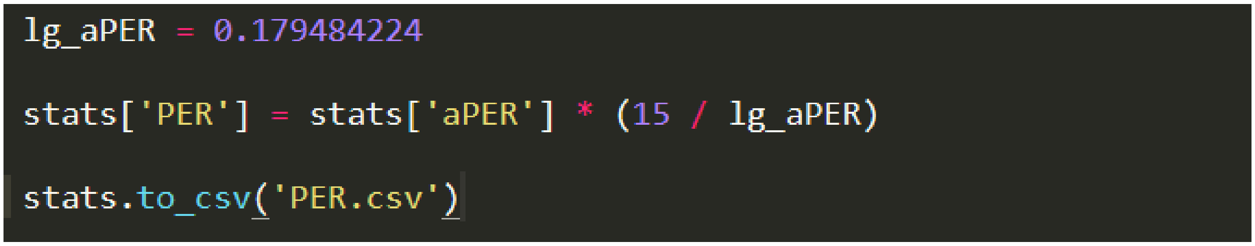
	AD	AE	AF
7	-0.0396	-4.14E-05	
7	0.07647	0.000260648	
5	0.27794	0.001696954	
5	0.15946	0.000516739	
5	-0.08	-0.000140051	
5	0.29354	0.00176565	
5	0.06598	3.76E-05	
5	0.11387	0.000136521	
5	0.08035	0.000189156	
5	0.38061	0.001461878	
5	0.12003	0.000369806	
5	0.17938	0.000884271	
5	0.26728	0.000998471	
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



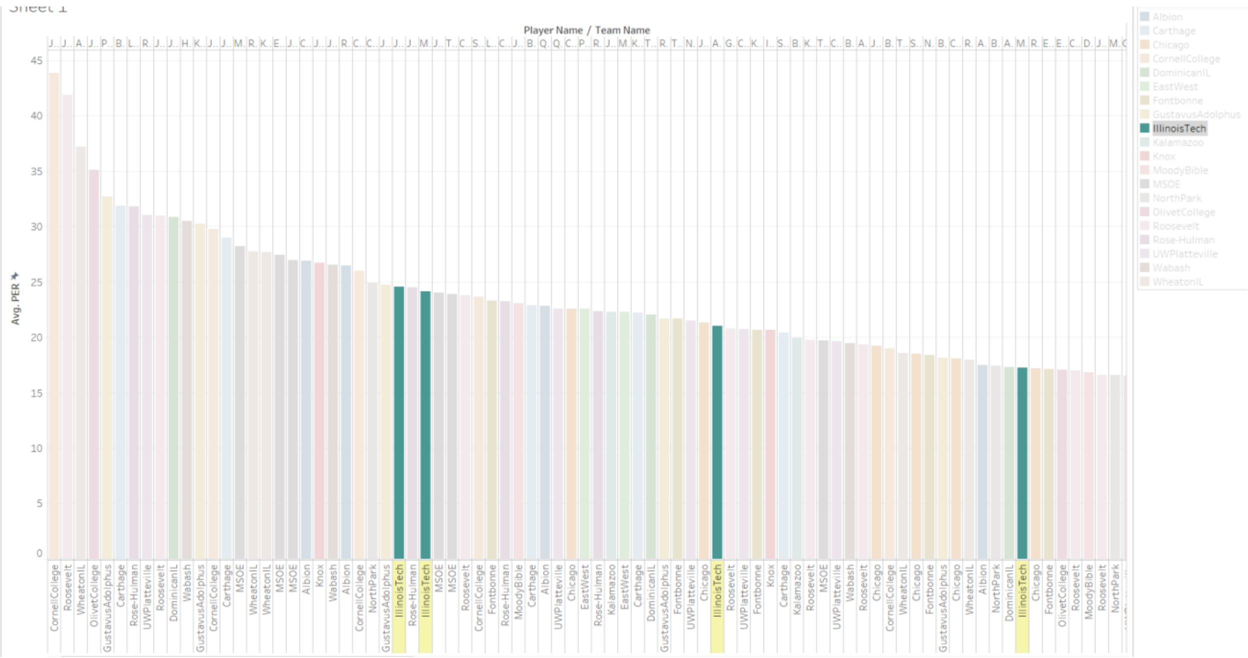
```
lg_aPER = 0.179484224
```

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!


Sheet 1
Player Name / Team Name
Avg. PER
CornellCollege
Roosevelt
WheatonIL
OlivetCollege
GustavusAdolphus
Carthage
Rose-Hulman
UWPlatteville
Roosevelt
DominicanIL
Wabash
GustavusAdolphus
CornellCollege
Carthage
MSOE
WheatonIL
WheatonIL
MSOE
MSOE
Albion
Knox
Wabash
Albion
CornellCollege
NorthPark
GustavusAdolphus
IllinoisTech
Rose-Hulman
IllinoisTech
MSOE
MSOE
Roosevelt
CornellCollege
Fontbonne
Rose-Hulman
MoodyBible
Carthage
Albion
UWPlatteville
Chicago
EastWest
Rose-Hulman
Kalamazoo
EastWest
Carthage
DominicanIL
GustavusAdolphus
Fontbonne
UWPlatteville
Chicago
IllinoisTech
Roosevelt
UWPlatteville
Fontbonne
Knox
Carthage
Kalamazoo
Roosevelt
MSOE
UWPlatteville
Wabash
Roosevelt
Chicago
CornellCollege
WheatonIL
Chicago
Fontbonne
GustavusAdolphus
Chicago
WheatonIL
Albion
NorthPark
DominicanIL
IllinoisTech
Chicago
Fontbonne
OlivetCollege
Roosevelt
MoodyBible
Roosevelt
NorthPark
Albion
Carthage
Chicago
CornellCollege
DominicanIL
EastWest
Fontbonne
GustavusAdolphus
IllinoisTech
Kalamazoo
Knox
MoodyBible
MSOE
NorthPark
OlivetCollege
Roosevelt
Rose-Hulman
UWPlatteville
Wabash
WheatonIL