NBA Stats analysis

We are going to delve into some interesting NBA statistics in this notebook. There are numerous NBA stats we can explore. For this project i am going to work with the NBA dataset provided in the Kaggle website. The link is given below.

NBA stats dataset (https://www.kaggle.com/nathanlauga/nba-games)

Motivation for this project

I started playing basketball at the age 10. I fell in love with this awesome game ever since. I used to watch NBA videos a lot back then, trying to simulate the moves of my idol, MJ, sticking my tongue out going hard towards the rim trying hang in the air before releasing ball into the hoop expecting to hear a swish. As i grew up i started watching NBA games waking up early in the morning(due to time-zone difference). Initially I never understood the NBA stat terms like assists, triple double, defense ratings etc. The more I spent my time watching and playing, I found myself intruiged by the volume of data being generated in each and every minute of the game.

With this project I am intending to explore some interesting stats and trying to find answers to questions that are always discussed in mainstream NBA. I am excited to explore stats and visualise them with all the knowledge that i have gained taking this course. Lets get started.

This notebook will explore

1. Best defensive player of the decade (2010-2020)
2. Best offensive player of the decade (2010-2020)
3. Best player of the decade (2010-2020)
4. Popular player comparisons
5. How has NBA transformed from last decade(2000-2010) to this decade(2010-2020)

Familiarising dataset

In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
from collections import defaultdict
from datetime import datetime
from IPython.display import display, Markdown
import seaborn as sns
from math import pi
import collections
from scipy.stats import rankdata


#path = 'D:/NIKHIL/ANACONDA PYTHON/Git repository/Python/Exercise files/Basic Data processsing and Visualisatio
n/Final Project/Dataset/games_details.csv'
path = 'M:/Git repository/Python/Exercise files/Basic Data processsing and Visualisation/Final Project/Dataset/
games_details.csv'
#path_2 = 'D:/NIKHIL/ANACONDA PYTHON/Git repository/Python/Exercise files/Basic Data processsing and Visualisat
ion/Final Project/Dataset/games.csv'
path_2 = 'M:/Git repository/Python/Exercise files/Basic Data processsing and Visualisation/Final Project/Datase
t/games.csv'
df_games1 = pd.read_csv(path)
df_gms = pd.read_csv(path_2)
df_gms.head()
```

Out[1]:

	GAME_DATE_EST	GAME_ID	GAME_STATUS_TEXT	HOME_TEAM_ID	VISITOR_TEAM_ID	SEASON	TEAM_ID_home	PTS_home	FG_PC
0	2020-03-01	21900895	Final	1610612766	1610612749	2019	1610612766	85.0	
1	2020-03-01	21900896	Final	1610612750	1610612742	2019	1610612750	91.0	
2	2020-03-01	21900897	Final	1610612746	1610612755	2019	1610612746	136.0	
3	2020-03-01	21900898	Final	1610612743	1610612761	2019	1610612743	133.0	
4	2020-03-01	21900899	Final	1610612758	1610612765	2019	1610612758	106.0	

5 rows × 21 columns

In [2]:
```python
df_games1.head()
```

Out[2]:

	GAME_ID	TEAM_ID	TEAM_ABBREVIATION	TEAM_CITY	PLAYER_ID	PLAYER_NAME	START_POSITION	COMMENT	MIN	FGM	..
0	21900895	1610612749	MIL	Milwaukee	202083	Wesley Matthews	F	NaN	27:08	3.0	..
1	21900895	1610612749	MIL	Milwaukee	203507	Giannis Antetokounmpo	F	NaN	34:55	17.0	..
2	21900895	1610612749	MIL	Milwaukee	201572	Brook Lopez	C	NaN	26:25	4.0	..
3	21900895	1610612749	MIL	Milwaukee	1628978	Donte DiVincenzo	G	NaN	27:35	1.0	..
4	21900895	1610612749	MIL	Milwaukee	202339	Eric Bledsoe	G	NaN	22:17	2.0	..

5 rows × 28 columns

In [3]:
```python
display(Markdown(f"Dataset rows and columns : {df_games1.shape}"))
```

Dataset rows and columns : (576782, 28)

Thats a whole lot of data

In [4]:
```python
No_of_games = [d for d in df_games1.groupby('GAME_ID',axis=0)]
display(Markdown(f"Number of games : {len(No_of_games)}"))
```

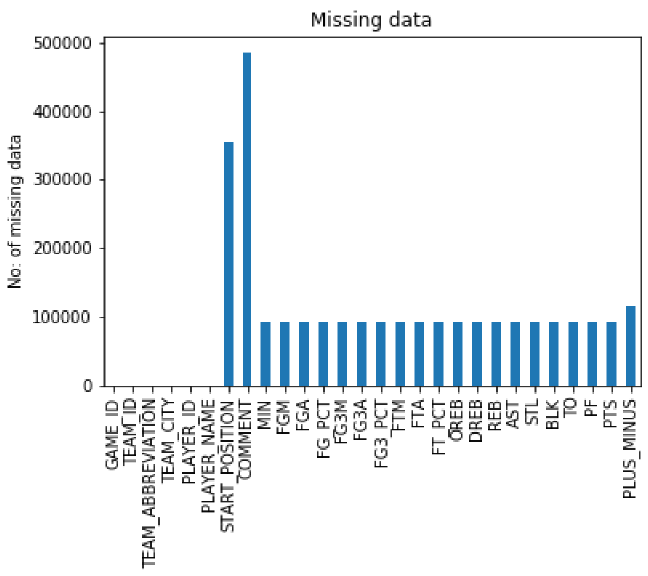
Number of games : 23096

Missing data

```
In [5]: nullvals = df_games1.isnull().sum()
        display(Markdown(f"Number of cells with missing data : {nullvals.sum()}"))
        nullvals.plot(kind='bar')
        plt.ylabel('No: of missing data')
        plt.title('Missing data')
```

Number of cells with missing data : 2708811

```
Out[5]: Text(0.5, 1.0, 'Missing data')
```



We can clearly see that the maximum missing data is in the Comments column followed by Starting Position column. For the analysis in this notebook we are going to fill all the missing data with 0 and further reprocess if required

Cleaning the data

```
In [6]: df_games1 = df_games1.fillna(0)
        nullvals = df_games1.isnull().sum()
        display(Markdown(f"Number of cells with missing data : {nullvals.sum()}"))
```

Number of cells with missing data : 0

```
In [8]: display(Markdown(f"Cleaning the Minutes played data to extract no: of games played"))
        df_games1['MINS'] = df_games1['MIN']
        zero_integer = (df_games1['MIN'] == 0)
        df_games1.loc[zero_integer,'MIN'] = '00:00'
        double_digit = df_games1['MIN'].str.len()==2
        df_games1.loc[double_digit,'MIN'] = df_games1['MIN'].astype(str)+':00'
        df_games1.loc[df_games1.index,'MIN'] = df_games1['MIN'].str.replace("60","59")
        single_digit = df_games1['MIN'].str.len() ==1
        df_games1.loc[single_digit,'MIN'] = '0'+df_games1['MIN'].astype(str)+':00'
        df_games1.loc[df_games1.index,'MIN'] = df_games1['MIN'].str.replace("-","")
        df_games1['MIN']
        display(Markdown(f"The Minutes played column is now in clean operable format to extract number of seconds playe
        d"))
```

Cleaning the Minutes played data to extract no: of games played

```
Out[8]: 0         27:08
        1         34:55
        2         26:25
        3         27:35
        4         22:17
                  ...
        576777    19:00
        576778    23:00
        576779    15:00
        576780    19:00
        576781    27:00
        Name: MIN, Length: 576782, dtype: object
```

The Minutes played column is now in clean operable format to extract number of seconds played

Win team

Combining the winning team data of each game from games.csv file to the pandas dataframe

In [9]:
```
df_win_team = {}
for ind in df_gms.index:
    if df_gms['HOME_TEAM_WINS'][ind] >0:
        df_win_team[df_gms['GAME_ID'][ind]] = df_gms['TEAM_ID_home'][ind]
    else:
        df_win_team[df_gms['GAME_ID'][ind]] = df_gms['TEAM_ID_away'][ind]
df_games1['WIN_TEAM_ID'] = df_games1['GAME_ID'].map(df_win_team)
```

Time played by each player

In [10]:
```
cc=list(df_games1['MIN'])
time_list = []

for ind in cc:
    try:
        pt = datetime.strptime(ind,'%M:%S')
        total_seconds = pt.second + pt.minute*60
        time_list.append(total_seconds)
    except:
        time_list.append(0)
        k = df_games1.loc[df_games1['MIN'] == ind ].index[0]
        print(df_games1['PLAYER_NAME'][k],'-',ind)

df_games1['TIME_PLAYED']= time_list
```

```
Anthony Roberson - 78:00
Keith McLeod - 86:00
Matt Barnes - 96:00
Kevin Burleson - 62:00
Matt Carroll - 93:00
Primoz Brezec - 92:00
Raymond Felton - 79:00
Ryan Hollins - 84:00
Walter Herrmann - 72:00
Will Conroy - 65:00
Charlie Villanueva - 89:00
Chris McCray - 70:00
Nikola Jokic - 64:58
```

The players listed above has recorded their time of play in an errounous format which we are going to disregard for our further analysis

Ranking system

We are creating a ranking system to analyse the top players in each parameter like points, rebounds, assists etc. Based on individual rankings a consolidated ranking is created to analyse the best player in specific category like defensive, offensive, allround player etc.

In [12]:
```
condition = df_gms['SEASON'] > 2010
df_gms_decade_2020 = df_gms[condition]
df = df_games1[df_games1['GAME_ID'].isin(df_gms_decade_2020['GAME_ID'])]
def Rank(df_,x):
    df = df_
    df = df.set_index('PLAYER_NAME')
    players = defaultdict()
    for d in df_['PLAYER_NAME']:
        players[d] = +1
    list_players = list(players.keys())

    Tot_rebound_ = defaultdict()
    for name in list_players:
        Tot_rebound_[name] = df.loc[name][x].sum()

    Max = max(Tot_rebound_, key=Tot_rebound_.get)

    ff = dict(zip(Tot_rebound_.keys(), rankdata([-i for i in Tot_rebound_.values()], method='min')))
    rank = [(ff[p],p) for p in ff]
    rank.sort()
    rank = rank[:50]

    rank_dict = {}
    for i in rank:
        rank_dict[i[1]] = i[0]

    return rank_dict
def dict_to_tup(dict_):
    list_tup =[(dict_[p],p) for p in dict_]
    return list_tup.sort()

def NO_of_win(rank,df_,df2):
    player_win = {}
    for i in range(0,len(rank)):
        cond_1 = df_['PLAYER_NAME'] == rank[i][1]
        df_player_win = df_[cond_1]

        wins = 0
        for ind in df_player_win.index:
            if df_player_win['TEAM_ID'][ind] == df_player_win['WIN_TEAM_ID'][ind]:
                wins = wins + 1
        player_win[rank[i][1]] = wins
    return player_win

def ranking_dict(v1,v2,v3):
    new_dict = {}
    for i,j in v1.items():
        for x,y in v2.items():
            for a,b in v3.items():
                if i==x==a:
                    new_dict[i]=j+y+b
    new_rank = [(new_dict[p],p) for p in new_dict]
    new_rank.sort()
    return new_rank
def new_ranking_dict(v1,v2,v3,v4,v5):
    new_dict = {}
    for i,j in v1.items():
        for x,y in v2.items():
            for a,b in v3.items():
                for g,f in v4.items():
                    for k,l in v5.items():
                        if i==x==a==g==k:
                            new_dict[i]=j+y+b+f+l
    new_rank = [(new_dict[p],p) for p in new_dict]
    new_rank.sort()
    return new_rank
```

In [13]:
```
Reb_rank = Rank(df,'REB')
DReb_rank = Rank(df,'DREB')
OReb_rank = Rank(df,'OREB')
Pts_rank = Rank(df,'PTS')
Ast_rank = Rank(df,'AST')
Stl_rank = Rank(df,'STL')
Blk_rank = Rank(df,'BLK')
FG_pct_rank = Rank(df,'FG_PCT')
FG3_pct_rank = Rank(df,'FG3_PCT')
```

Defensive beast

A defensive player is evaluated on the basis of top defensive rebounds, steals and blocks.

In [14]:
```
defensive_list = ranking_dict(DReb_rank,Blk_rank,Stl_rank)
defensive_list[:5]
```

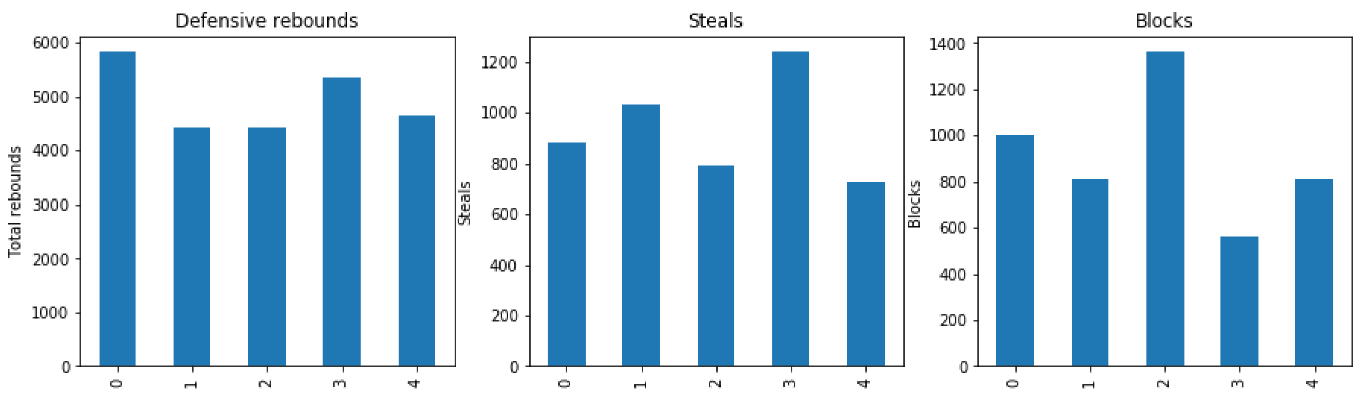
Out[14]:
```
[(32, 'Andre Drummond'),
 (38, 'Draymond Green'),
 (43, 'Anthony Davis'),
 (45, 'LeBron James'),
 (59, 'Kevin Durant')]
```

Andre Drummond is in the top of the Defensive Beast list with a consolidated rank of 32

In [45]:
```
df_Defensive_beast = pd.DataFrame(columns=['PLAYER_NAME', 'TOT_DREB', 'TOT_STL','TOT_BLK'])
pl_nme = []
tot_dreb = []
tot_stl = []
tot_blk = []
for rank,player in defensive_list[:5]:
    PL = df['PLAYER_NAME'] == player
    df_player = df[PL]
    pl_nme.append(player)
    tot_dreb.append(df_player['DREB'].sum())
    tot_stl.append(df_player['STL'].sum())
    tot_blk.append(df_player['BLK'].sum())

df_Defensive_beast['PLAYER_NAME'] = pl_nme
df_Defensive_beast['TOT_DREB'] = tot_dreb
df_Defensive_beast['TOT_STL'] = tot_stl
df_Defensive_beast['TOT_BLK'] = tot_blk
df_Defensive_beast = df_Defensive_beast.set_index('PLAYER_NAME')
```

In [53]:
```
fig,ax = plt.subplots(1,3)
fig1 = df_Defensive_beast['TOT_DREB'].plot(kind='bar',ax=ax[0],title='Defensive rebounds',figsize=(15, 4))
fig1.set_ylabel('Total rebounds')
fig2 = df_Defensive_beast['TOT_STL'].plot(kind='bar',ax=ax[1],title='Steals',figsize=(15, 4))
fig2.set_ylabel('Steals')
fig3 = df_Defensive_beast['TOT_BLK'].plot(kind='bar',ax=ax[2],title='Blocks',figsize=(15, 4))
fig3.set_ylabel('Blocks')
```

Out[53]:
```
Text(0, 0.5, 'Blocks')
```



Considering the ranking of Defensive rebounds, Blocks and Steals, Andre Drummond is the defensive beast in 2010-2020 decade. Let us see the contribution of each of the top 5 Defensive beasts on the taking the team to victory.

```
In [49]: df_Defensive_beast = df_Defensive_beast.reset_index('PLAYER_NAME')
         player_wins = NO_of_win(defensive_list,df,df_gms_decade_2020)
         df_Defensive_beast['TOT_WINS'] = df_Defensive_beast['PLAYER_NAME'].map(player_wins)
         df_Defensive_beast.set_index('PLAYER_NAME')
```

Out[49]:

PLAYER_NAME	TOT_DREB	TOT_STL	TOT_BLK	TOT_WINS
Andre Drummond	5822.0	883.0	1002.0	287
Draymond Green	4437.0	1030.0	812.0	522
Anthony Davis	4436.0	790.0	1361.0	311
LeBron James	5366.0	1238.0	564.0	565
Kevin Durant	4638.0	726.0	814.0	495

Considering the total wins contribution we can clearly say that Draymond Green has been a brut force in the defence with providing 522 wins for his team with 4K defensive rebounds 1000+ steals and 800+ blocks from 2010-2020.

Defensive Beast : Draymond Green

Best offensive player

An offensive player is evaluated based on the points he has scored and how consistent was he with it. Maximum number of 30 point games also considering the number of wins he was able to deliver his team.

```
In [18]: cond_3 = df['PTS']>=30
         df_thirty_plus_gms = df[cond_3]
         thirty_plus_gms = df_thirty_plus_gms['PLAYER_NAME'].value_counts()

         thirty_plus_rank = {}
         a=1
         for i in  thirty_plus_gms.index:
             thirty_plus_rank[i] = a
             a = a+1

         offensive_list = ranking_dict(thirty_plus_rank,Pts_rank,FG_pct_rank)
         offensive_list[:5]
```

```
Out[18]: [(6, 'LeBron James'),
          (17, 'James Harden'),
          (19, 'Kevin Durant'),
          (38, 'Stephen Curry'),
          (40, 'Klay Thompson')]
```

Lebron James is in the Top ranks as the Best offensive player with a consolidated rank of 6

```
In [42]: df_Ofensive_beast = pd.DataFrame(columns=['PLAYER_NAME', '30_PLUS_GAMES', 'TOT_PTS','FG_PCT'])
         pl_nme = []
         tot_30_plus_games = []
         tot_pts = []
         tot_fg_pct = []
         for rank,player in offensive_list[:5]:
             PL = df['PLAYER_NAME'] == player
             df_player = df[PL]
             pl_nme.append(player)
             tot_pts.append(int(df_player['PTS'].sum()))
             tot_fg_pct.append(round(df_player['FG_PCT'].mean(),3))
             for i in thirty_plus_gms.index:
                 if i == player:
                     tot_30_plus_games.append(thirty_plus_gms[i])
         df_Ofensive_beast['PLAYER_NAME'] = pl_nme
         df_Ofensive_beast['30_PLUS_GAMES'] = tot_30_plus_games
         df_Ofensive_beast['FG_PCT'] = tot_fg_pct
         df_Ofensive_beast['TOT_PTS'] = tot_pts
         df_Ofensive_beast = df_Ofensive_beast.set_index('PLAYER_NAME')
```

In [43]:
```
df_Ofensive_beast
```

Out[43]:

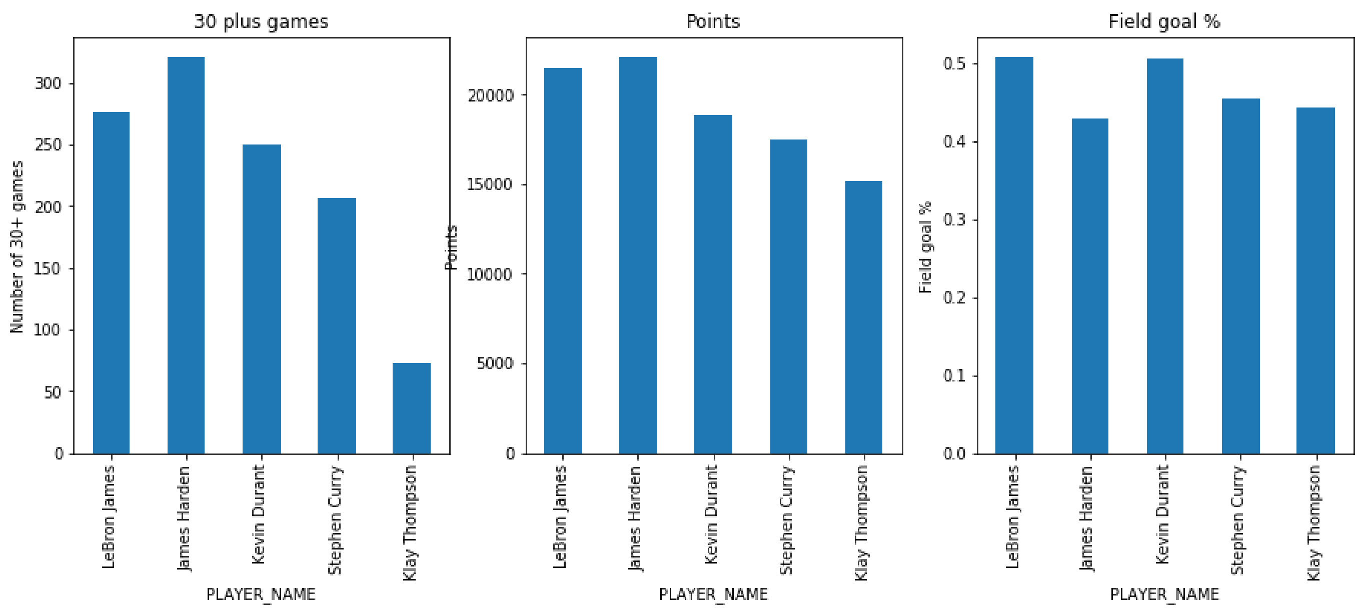
PLAYER_NAME	30_PLUS_GAMES	TOT_PTS	FG_PCT
LeBron James	276	21431	0.508
James Harden	321	22086	0.429
Kevin Durant	250	18819	0.505
Stephen Curry	206	17432	0.454
Klay Thompson	73	15127	0.443

In [20]:
```
display(Markdown(f"It sure looks like {offensive_list[0][1]} is in the lead!"))
```

It sure looks like LeBron James is in the lead!

In [44]:
```
fig,ax = plt.subplots(1,3)
fig1 = df_Ofensive_beast['30_PLUS_GAMES'].plot(kind='bar',ax=ax[0],title='30 plus games',figsize=(15, 5))
fig1.set_ylabel('Number of 30+ games')
fig2 = df_Ofensive_beast['TOT_PTS'].plot(kind='bar',ax=ax[1],title='Points',figsize=(15, 5))
fig2.set_ylabel('Points')
fig3 = df_Ofensive_beast['FG_PCT'].plot(kind='bar',ax=ax[2],title='Field goal %',figsize=(15, 5))
fig3.set_ylabel('Field goal %')
```

Out[44]:
```
Text(0, 0.5, 'Field goal %')
```



In [22]:
```
player_win = NO_of_win(offensive_list,df,df_gms_decade_2020)
df_Ofensive_beast['TOT_WINS'] = df_Ofensive_beast['PLAYER_NAME'].map(player_win)
df_Ofensive_beast.set_index('PLAYER_NAME')
```

Out[22]:

PLAYER_NAME	30_PLUS_GAMES	TOT_PTS	FG_PCT	TOT_WINS
LeBron James	276	21431	0.508	565
James Harden	321	22086	0.429	528
Kevin Durant	250	18819	0.505	495
Stephen Curry	206	17432	0.454	515
Klay Thompson	73	15127	0.443	550

Best offensive player is tough competition between Lebron, Harden and Durant. It comes down to the number of wins produced by the individual. With 565 wins Lebron James wins the best ofensive player of 2010-2020 decade.

Best Offensive player : Lebron James

Best player of the decade (2010-2020)

A best player is evaluated based on lots of criteria. Today we are going to explore the following stats to arrive at our conclusion of best player. We are going to take into consideration the efficiency of a player in deciding the same. That is essentially the stats considering the number of minutes he played. The following parameters are going to be looked into.

1. Points scored
2. Offensive and defensive rebounds
3. Steals
4. Blocks
5. Assists
6. Player efficiency i- ((PTS + REB + AST + STL + BLK − Missed FG − Missed FT - TO) / GP)

In [23]:
```
top_10_Reb = list(n for n,i in Reb_rank.items())[:10]
top_10_Pts = list(n for n,i in Pts_rank.items())[:10]
top_10_ast = list(n for n,i in Ast_rank.items())[:10]
top_10_Stl = list(n for n,i in Stl_rank.items())[:10]
top_10_Blk = list(n for n,i in Blk_rank.items())[:10]
df_Best_player = pd.DataFrame(columns=['REB rank', 'PTS rank', 'STL rank','BLK rank','AST rank'])
df_Best_player['REB rank'] = top_10_Reb
df_Best_player['PTS rank'] = top_10_Pts
df_Best_player['STL rank'] = top_10_ast
df_Best_player['BLK rank'] = top_10_Stl
df_Best_player['AST rank'] = top_10_Blk
df_Best_player
```

Out[23]:

	REB rank	PTS rank	STL rank	BLK rank	AST rank
0	DeAndre Jordan	James Harden	Russell Westbrook	Chris Paul	Serge Ibaka
1	Andre Drummond	LeBron James	Chris Paul	James Harden	Anthony Davis
2	Dwight Howard	Kevin Durant	LeBron James	Russell Westbrook	DeAndre Jordan
3	LeBron James	Russell Westbrook	James Harden	Paul George	Rudy Gobert
4	Nikola Vucevic	Stephen Curry	John Wall	LeBron James	Brook Lopez
5	Tristan Thompson	Damian Lillard	Rajon Rondo	Stephen Curry	Hassan Whiteside
6	LaMarcus Aldridge	DeMar DeRozan	Kyle Lowry	Thaddeus Young	Dwight Howard
7	Kevin Love	Klay Thompson	Jeff Teague	Kawhi Leonard	Andre Drummond
8	Anthony Davis	LaMarcus Aldridge	Stephen Curry	Trevor Ariza	Marc Gasol
9	Marcin Gortat	Paul George	Ricky Rubio	Ricky Rubio	Roy Hibbert

In [56]:
```
Best_Player = new_ranking_dict(Reb_rank,Pts_rank,Ast_rank,Stl_rank,Blk_rank)
Best_Player
```

Out[56]:
```
[(51, 'LeBron James'), (98, 'Kevin Durant'), (147, 'Giannis Antetokounmpo')]
```

LeBron James Leads in the ranking list considering Points, Assists, Steals, Rebounds and Blocks with a consolidated rank of 51. Let us also check out the player efficiency rating for the three contenders for Best player of the decade 2010-2020.

In [39]:
```
pl_nme_e = []
tot_dreb_e = []
tot_pts_e = []
tot_stl_e = []
tot_blk_e = []
tot_ast_e = []
tot_missed_fg_e = []
tot_missed_ft_e = []
tot_TO = []
tot_MTS = []
tot_fg_pct = []
tot_fg3_pct = []
tot_ft_pct = []


for i,player in Best_Player:
    PL_e = df['PLAYER_NAME'] == player
    df_player = df[PL_e]
    pl_nme_e.append(player)
    tot_pts_e.append(df_player['PTS'].sum())
    tot_dreb_e.append(df_player['DREB'].sum())
    tot_stl_e.append(df_player['STL'].sum())
    tot_blk_e.append(df_player['BLK'].sum())
    tot_ast_e.append(df_player['AST'].sum())
    tot_missed_fg_e.append(df_player['FGA'].sum()-df_player['FGM'].sum())
    tot_missed_ft_e.append(df_player['FTA'].sum()-df_player['FTM'].sum())
    tot_TO.append(df_player['TO'].sum())
    tot_fg_pct.append(round(df_player['FG_PCT'].mean(),3))
    tot_fg3_pct.append(round(df_player['FG3_PCT'].mean(),3))
    tot_ft_pct.append(round(df_player['FT_PCT'].mean(),3))
#    time_played = df_player['TIME_PLAYED'].sum()
#    tot_MTS.append(round(time_played/gt))
    GP = 0
    for ind in df_player.index:
        if df_player['TIME_PLAYED'][ind]>0:
            GP+=1
    tot_MTS.append(GP)

df_Best_player = pd.DataFrame(columns=['PLAYER_NAME','REB', 'PTS', 'STL','BLK','AST','M_FG','M_FT','TO','GP','P
_EFF'])
df_Best_player['PLAYER_NAME'] = pl_nme_e
df_Best_player['REB'] = tot_dreb_e
df_Best_player['PTS'] = tot_pts_e
df_Best_player['STL'] = tot_stl_e
df_Best_player['BLK'] = tot_blk_e
df_Best_player['AST'] = tot_ast_e
df_Best_player['M_FG'] = tot_missed_fg_e
df_Best_player['M_FT'] = tot_missed_ft_e
df_Best_player['TO'] = tot_TO
df_Best_player['FG_PCT'] = tot_fg_pct
df_Best_player['FG3_PCT'] = tot_fg3_pct
df_Best_player['FT_PCT'] = tot_ft_pct
df_Best_player['GP'] = tot_MTS
df_Best_player['P_EFF'] =   round((df_Best_player['REB']+
                            df_Best_player['PTS']+
                            df_Best_player['STL']+
                            df_Best_player['BLK']+
                            df_Best_player['AST']-
                            df_Best_player['M_FG']-
                            df_Best_player['M_FT']-
                            df_Best_player['TO'])/df_Best_player['GP'],2)
df_Best_player1 = df_Best_player.set_index('PLAYER_NAME')
df_Best_player1['P_EFF']

player_wins = NO_of_win(Best_Player,df,df_gms_decade_2020)
df_Best_player['TOT_WINS'] = df_Best_player['PLAYER_NAME'].map(player_wins)
df_Best_player['WIN_PCT'] =  round((df_Best_player['TOT_WINS']/df_Best_player['GP'])*100 ,1)
```

```
In [57]: df_Best_player.set_index('PLAYER_NAME')
```

Out[57]:

PLAYER_NAME	REB	PTS	STL	BLK	AST	M_FG	M_FT	TO	GP	P_EFF	FG_PCT	FG3_PCT	FT_PCT	TOT_WINS	WI
LeBron James	5366.0	21431.0	1238.0	564.0	6086.0	7185.0	1589.0	2913.0	807	28.50	0.508	0.303	0.678	565	
Kevin Durant	4638.0	18819.0	726.0	814.0	3253.0	6278.0	608.0	2149.0	684	28.09	0.505	0.386	0.842	495	
Giannis Antetokounmpo	4175.0	11585.0	691.0	780.0	2473.0	3836.0	1105.0	1606.0	585	22.49	0.487	0.192	0.645	309	

With higher Total wins,player efficiency rating, Rebounds, Points, Steals, and assists Lebron leaps over Durant to be the winner in all aspects.

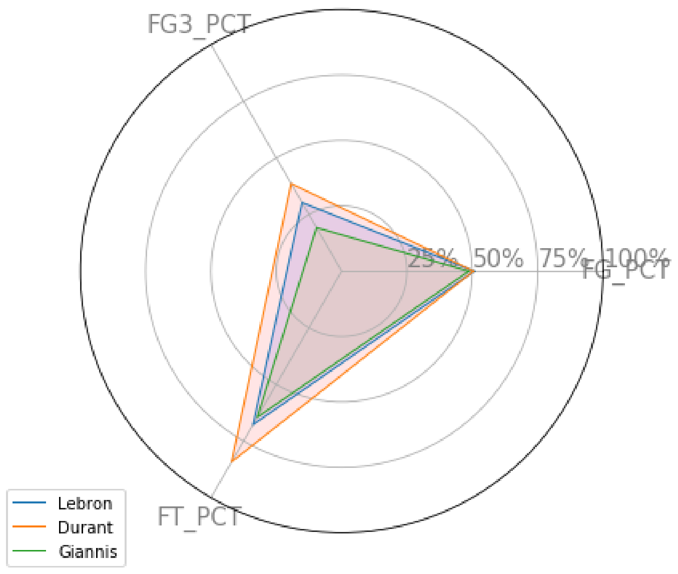
The clear winner is Lebron James with a Player efficiency rating of 28.5 during 2010-2020, with most wins produced for his team. He sure is the most valuable player. The numbers generated by Lebron is truly amazing.

Best Player of the Decade : Lebron James

Let us look into an interesting statistic

```
In [41]: categories = list(df_Best_player)[11:14]
         N = len(categories)

         angles = [n / float(len(categories)) * 2 * pi for n in range(len(categories))]
         angles += angles[:1]
         plt.figure(figsize=(20,6))
         ax = plt.subplot(111, polar=True)
         plt.xticks(angles[:-1], categories, color='grey', size=15)
         plt.yticks([0.25,0.5,0.75,1], ["25%","50%","75%","100%"], color="grey", size=15)
         plt.ylim(0,1)
         ax.set_rlabel_position(0)

         values = df_Best_player.loc[0].drop('PLAYER_NAME').values.flatten().tolist()
         values = values[10:13]
         values += values[:1]

         ax.plot(angles, values, linewidth=1, linestyle='solid',label = "Lebron")
         ax.fill(angles, values, 'b', alpha=0.1)

         values = df_Best_player.loc[1].drop('PLAYER_NAME').values.flatten().tolist()
         values = values[10:13]
         values += values[:1]

         ax.plot(angles, values, linewidth=1, linestyle='solid',label = "Durant")
         ax.fill(angles, values, 'r', alpha=0.1)

         values = df_Best_player.loc[2].drop('PLAYER_NAME').values.flatten().tolist()
         values = values[10:13]
         values += values[:1]

         ax.plot(angles, values, linewidth=1, linestyle='solid',label = "Giannis")
         ax.fill(angles, values, 'y', alpha=0.1)
         plt.legend(loc='upper right', bbox_to_anchor=(0.1, 0.1))
```

```
Out[41]: <matplotlib.legend.Legend at 0x1808caef088>
```



Above plot shows the Field goal, 3pointer and Free throw percentages. It is interesting to see that Kevin Durant surpasses James in all the 3 departments, proving to be a sharper shooter compared to Lebron James. We have also seen that Kevin Durant's Winning percentage is 2% higher compared to Lebron James. Will Kevin Durant be able to match up to Lebron James in the coming years as he enters his peak career stage we will see in the future.