Data Scraping and Analysis : Build your NBA Dream Team

You want to create the perfect 5 Major ? You want to know who players will be the next NBA stars ? You want to predict what will be the american olympic team at the next Olympic Games ?

- In this project, I will use NBA players datas in order to create a function that may turn out to be very useful in your team making decision process.

- A Five Major is the five players of a team that start games, oftentimes the best players of the team.

In this notebook I will use :

- Web scraping for collecting data on a website.
- Object oriented programming.
- SQL commands.
- Data cleaning for making the datas standardized and usable.
- Data analysis for checking data quality and getting insights.

Packages and librairies

In [124…

```
from bs4 import BeautifulSoup
import requests
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from matplotlib.font_manager import FontProperties

import warnings;
warnings.filterwarnings('ignore')

%matplotlib inline
```

2) Web Scraping

We will scrape each season datas from 1980 to 2020. A season table contain the players average statistics per game.

In [125…

```
### 1) Create the framework of the dataframe

page = requests.get('https://www.basketball-reference.com/leagues/NBA_2020_per_game.html')
soup = BeautifulSoup(page.content,'html.parser')
tableau = soup.find_all(class_="full_table")

head = soup.find(class_='thead')
column_name = [head.text for item in head][0]
column_name = column_name.split('\n')
del(column_name[:2])
del(column_name[-1])
df = pd.DataFrame(columns=column_name)
```

In [3]:

```
df
```

Out[3]:

Player	Pos	Age	Tm	G	GS	MP	FG	FGA	FG%	...	FT%	ORB	DRB	TRB	AST	STL	BLK	TOV	PF	PTS

0 rows × 29 columns

In [5]:

```
### 2) Fill the dataframe

for season in range(1980,2021):

    url_players = 'https://www.basketball-reference.com/leagues/NBA_{}_per_game.html'.format(season)

    page = requests.get(url_players)
    soup = BeautifulSoup(page.content,'html.parser')
    tableau = soup.find_all(class_="full_table")

    players = []


    for i in range(len(tableau)):

        player = []
        for j in tableau[i].find_all('td'):
            player.append(j.text)
        players.append(player)



    head2 = soup.find(class_='thead')
    column_name2 = [head2.text for item in head][0]
    column_name2 = column_name2.split('\n')
    del(column_name2[:2])
    del(column_name2[-1])
    df2 = pd.DataFrame(players, columns=column_name2)
    df2['Season'] = season
    df = df.append(df2,ignore_index=True,sort=False)
```

3) Data quality check & data cleaning

In [14]:

```
df.head()
```

Out[14]:

	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	FG%	...	ORB	DRB	TRB	AST	STL	BLK	TOV	PF	PTS	Season
0	Kareem Abdul-Jabbar*	C	32	LAL	82		38.3	10.2	16.9	.604	...	2.3	8.5	10.8	4.5	1.0	3.4	3.6	2.6	24.8	1980.0
1	Tom Abernethy	PF	25	GSW	67		18.2	2.3	4.7	.481	...	0.9	1.9	2.9	1.3	0.5	0.2	0.6	1.8	5.4	1980.0
2	Alvan Adams	C	25	PHO	75		28.9	6.2	11.7	.531	...	2.1	6.0	8.1	4.3	1.4	0.7	2.9	3.2	14.9	1980.0
3	Tiny Archibald*	PG	31	BOS	80	80	35.8	4.8	9.9	.482	...	0.7	1.7	2.5	8.4	1.3	0.1	3.0	2.7	14.1	1980.0
4	Dennis Awtrey	C	31	CHI	26		21.5	1.0	2.3	.450	...	1.1	3.3	4.4	1.5	0.5	0.6	1.0	2.5	3.3	1980.0

5 rows × 30 columns

In [21]:

```
for col in  df.columns[4:30]:
    df[col] = pd.to_numeric(df[col], errors='coerce')

df['Age'] = df['Age'].astype(int)
df['Season'] = df['Season'].astype(int)
```

In [22]:

```
df.head()
```

Out[22]:

	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	FG%	...	ORB	DRB	TRB	AST	STL	BLK	TOV	PF	PTS	Season
0	Kareem Abdul-Jabbar*	C	32	LAL	82	NaN	38.3	10.2	16.9	0.604	...	2.3	8.5	10.8	4.5	1.0	3.4	3.6	2.6	24.8	1980
1	Tom Abernethy	PF	25	GSW	67	NaN	18.2	2.3	4.7	0.481	...	0.9	1.9	2.9	1.3	0.5	0.2	0.6	1.8	5.4	1980
2	Alvan Adams	C	25	PHO	75	NaN	28.9	6.2	11.7	0.531	...	2.1	6.0	8.1	4.3	1.4	0.7	2.9	3.2	14.9	1980
3	Tiny Archibald*	PG	31	BOS	80	80.0	35.8	4.8	9.9	0.482	...	0.7	1.7	2.5	8.4	1.3	0.1	3.0	2.7	14.1	1980
4	Dennis Awtrey	C	31	CHI	26	NaN	21.5	1.0	2.3	0.450	...	1.1	3.3	4.4	1.5	0.5	0.6	1.0	2.5	3.3	1980

5 rows × 30 columns

Check the NaN values percentage in each column. Fortunately columns with NaN will not be used in our case.

In [29]:

```
df.isna().sum()/df.shape[0]
```

Out[29]:

```
Player    0.000000
Pos       0.000000
Age       0.000000
Tm        0.000000
G         0.000000
GS        0.061689
MP        0.000000
FG        0.000000
FGA       0.000000
FG%       0.002989
3P        0.000000
3PA       0.000000
3P%       0.158340
2P        0.000000
2PA       0.000000
2P%       0.004793
eFG%      0.002989
FT        0.000000
FTA       0.000000
FT%       0.029266
ORB       0.000000
DRB       0.000000
TRB       0.000000
AST       0.000000
STL       0.000000
BLK       0.000000
TOV       0.000000
PF        0.000000
PTS       0.000000
Season    0.000000
dtype: float64
```

Visualize the composition of non numeric columns

- A '*' after a player's name means that the players is not in the NBA anymore
- A basketball team is composed of five players. Each of them has a position : 'C' means 'Center, 'PF' means 'Power Forward', 'PG' means 'Point Guard', 'SG' means 'Shooting Guard', 'SF' means 'Small Forward'. There are also players that get play at two different positions.
- NBA teams are often reported by their acronym, for instance 'LAL' means 'Los Angeles Lakers'.

In [37]:

```
for col in df.select_dtypes('object'):
    print(f'{col :-<40} {df[col].unique()}')
```

```
Player---------------------------------- ['Kareem Abdul-Jabbar*' 'Tom Abernethy' 'Alvan Adams' ...
 'Nigel Williams-Goss' 'Zion Williamson' 'Justin Wright-Foreman']
Pos------------------------------------- ['C' 'PF' 'PG' 'SG' 'SF' 'SG-PG' 'SF-SG' 'SG-SF' 'C-PF' 'PF-C' 'SF-PF'
 'PG-SG' 'PF-SF' 'PG-SF' 'SG-PF' 'SF-C']
Tm-------------------------------------- ['LAL' 'GSW' 'PHO' 'BOS' 'CHI' 'WSB' 'SEA' 'IND' 'SDC' 'HOU' 'TOT' 'POR'
 'PHI' 'KCK' 'NJN' 'MIL' 'UTA' 'CLE' 'NYK' 'ATL' 'SAS' 'DET' 'DEN' 'DAL'
 'LAC' 'SAC' 'CHH' 'MIA' 'ORL' 'MIN' 'VAN' 'TOR' 'WAS' 'MEM' 'NOH' 'CHA'
 'NOK' 'OKC' 'BRK' 'NOP' 'CHO']
```

In [31]:

```
df.describe().transpose()
```

Out[31]:

	count	mean	std	min	25%	50%	75%	max
Age	17734.0	26.671479	4.049753	18.0	24.000	26.000	29.000	44.0
G	17734.0	54.691948	25.276153	1.0	36.000	63.000	77.000	85.0
GS	16640.0	26.503005	29.692342	0.0	1.000	11.000	53.000	83.0
MP	17734.0	20.583066	10.112226	0.0	12.100	20.100	29.100	43.7
FG	17734.0	3.229892	2.291933	0.0	1.400	2.700	4.600	13.4
FGA	17734.0	7.070971	4.712975	0.0	3.300	5.900	10.000	27.8
FG%	17681.0	0.443537	0.090097	0.0	0.407	0.448	0.489	1.0
3P	17734.0	0.418473	0.612239	0.0	0.000	0.100	0.700	5.1
3PA	17734.0	1.217159	1.622814	0.0	0.000	0.400	2.000	13.2
3P%	14926.0	0.251542	0.172184	0.0	0.125	0.296	0.364	1.0
2P	17734.0	2.810500	2.149860	0.0	1.125	2.200	4.000	13.2
2PA	17734.0	5.852650	4.223069	0.0	2.600	4.700	8.300	27.0
2P%	17649.0	0.464969	0.093403	0.0	0.430	0.472	0.508	1.0
eFG%	17681.0	0.471554	0.091737	0.0	0.440	0.480	0.515	1.5
FT	17734.0	1.615146	1.437898	0.0	0.600	1.200	2.200	10.3
FTA	17734.0	2.160697	1.816570	0.0	0.900	1.600	2.900	13.1
FT%	17215.0	0.725363	0.139363	0.0	0.667	0.750	0.813	1.0
ORB	17734.0	1.062310	0.875555	0.0	0.400	0.800	1.500	7.0
DRB	17734.0	2.567007	1.822295	0.0	1.200	2.100	3.400	12.3
TRB	17734.0	3.627963	2.576345	0.0	1.700	3.000	4.900	18.7
AST	17734.0	1.933771	1.873434	0.0	0.600	1.300	2.600	14.5
STL	17734.0	0.685491	0.482607	0.0	0.300	0.600	0.900	3.7
BLK	17734.0	0.427179	0.515082	0.0	0.100	0.300	0.500	5.6
TOV	17734.0	1.292162	0.832343	0.0	0.700	1.100	1.800	5.7
PF	17734.0	1.971050	0.863359	0.0	1.300	2.000	2.600	6.0
PTS	17734.0	8.492032	6.099345	0.0	3.700	7.000	12.000	37.1
Season	17734.0	2001.089207	12.005391	1980.0	1991.000	2002.000	2012.000	2020.0

4) Database Storage

In [103…

```
import sqlite3

conn = sqlite3.connect("projects.db")
df.to_sql("nba_players_stats", conn, if_exists="replace")
conn.commit()
conn.close()
```

In [126…

```
import sqlite3

conn = sqlite3.connect("projects.db")
df = pd.read_sql("SELECT * FROM nba_players_stats", conn)
conn.close()
```

In [64]:

```
df.head()
```

Out[64]:

	index	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	...	ORB	DRB	TRB	AST	STL	BLK	TOV	PF	PTS	Season
0	0	Kareem Abdul-Jabbar*	C	32	LAL	82	NaN	38.3	10.2	16.9	...	2.3	8.5	10.8	4.5	1.0	3.4	3.6	2.6	24.8	1980
1	1	Tom Abernethy	PF	25	GSW	67	NaN	18.2	2.3	4.7	...	0.9	1.9	2.9	1.3	0.5	0.2	0.6	1.8	5.4	1980
2	2	Alvan Adams	C	25	PHO	75	NaN	28.9	6.2	11.7	...	2.1	6.0	8.1	4.3	1.4	0.7	2.9	3.2	14.9	1980
3	3	Tiny Archibald*	PG	31	BOS	80	80.0	35.8	4.8	9.9	...	0.7	1.7	2.5	8.4	1.3	0.1	3.0	2.7	14.1	1980
4	4	Dennis Awtrey	C	31	CHI	26	NaN	21.5	1.0	2.3	...	1.1	3.3	4.4	1.5	0.5	0.6	1.0	2.5	3.3	1980

5 rows × 31 columns

5) Data analysis

Best scorer per season

In [127…

```
df_pts = pd.DataFrame(columns=column_name)

for season in range(1980,2021):
    df2 = df.loc[df['Season']==season].nlargest(1, ['PTS'])
    df_pts = df_pts.append(df2)

df_pts[['Player','PTS','Season']].sort_values(by='PTS',ascending=False)
```

Out[127…

	Player	PTS	Season
2909	Michael Jordan*	37.1	1987.0
16880	James Harden	36.1	2019.0
10604	Kobe Bryant*	35.4	2006.0
3249	Michael Jordan*	35.0	1988.0
17402	James Harden	34.3	2020.0
3957	Michael Jordan*	33.6	1990.0
97	George Gervin*	33.1	1980.0
2272	Bernard King*	32.9	1985.0
5119	Michael Jordan*	32.6	1993.0
3598	Michael Jordan*	32.5	1989.0
1276	George Gervin*	32.3	1982.0
9459	Tracy McGrady*	32.1	2003.0
14332	Kevin Durant	32.0	2014.0
11067	Kobe Bryant*	31.6	2007.0
16106	Russell Westbrook	31.6	2017.0
4344	Michael Jordan*	31.5	1991.0
8962	Allen Iverson*	31.4	2002.0
8516	Allen Iverson*	31.1	2001.0
10286	Allen Iverson*	30.7	2005.0
1555	Adrian Dantley*	30.7	1983.0
346	Adrian Dantley*	30.7	1981.0
1869	Adrian Dantley*	30.6	1984.0
6332	Michael Jordan*	30.4	1996.0
16328	James Harden	30.4	2018.0
2748	Dominique Wilkins*	30.3	1986.0
12318	Dwyane Wade	30.2	2009.0
15277	Stephen Curry	30.1	2016.0
4736	Michael Jordan*	30.1	1992.0
12482	Kevin Durant	30.1	2010.0
11672	LeBron James	30.0	2008.0
5645	David Robinson*	29.8	1994.0
8185	Shaquille O'Neal*	29.7	2000.0
6763	Michael Jordan*	29.6	1997.0
5999	Shaquille O'Neal*	29.3	1995.0
13745	Carmelo Anthony	28.7	2013.0
7215	Michael Jordan*	28.7	1998.0
15150	Russell Westbrook	28.1	2015.0
13377	Kevin Durant	28.0	2012.0
9901	Tracy McGrady*	28.0	2004.0
12927	Kevin Durant	27.7	2011.0
7629	Allen Iverson*	26.8	1999.0

In [128…

```
best_scorers = {}

for i in range(len(df_pts)):

    if df_pts.iat[i,0] not in best_scorers:
        best_scorers.update({df_pts.iat[i,0] : 1 })
    else:
        best_scorers[df_pts.iat[i,0]] += 1

print(best_scorers)
```

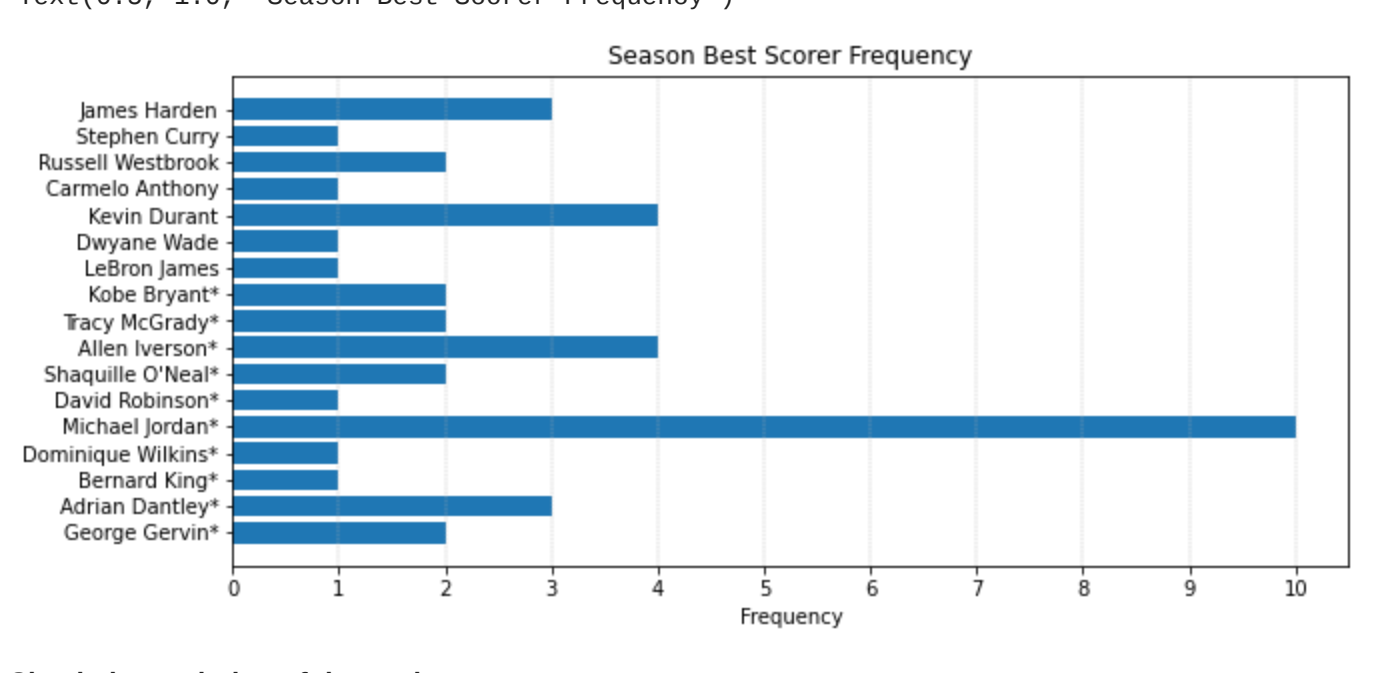
```
{'George Gervin*': 2, 'Adrian Dantley*': 3, 'Bernard King*': 1, 'Dominique Wilkins*': 1, 'Michael Jordan*': 10, 'David Robinson*': 1, "Shaquille O'Neal*": 2, 'Allen Iverson*': 4, 'Tracy McGrady*': 2, 'Kobe Bryant*': 2, 'LeBron James': 1, 'Dwyane Wade': 1, 'Kevin Durant': 4, 'Carmelo Anthony': 1, 'Russell Westbrook': 2, 'Stephen Curry': 1, 'James Harden': 3}
```

In [129…

```
plt.figure(figsize=(10,4.5))
plt.grid(axis = 'x', linestyle = '--', linewidth = 0.4)
plt.xticks(list(range(max(best_scorers.values())+1)))
plt.xlabel('Frequency')
plt.barh(*zip(*best_scorers.items()))
plt.title('Season Best Scorer Frequency')
```

Out[129…

```
Text(0.5, 1.0, 'Season Best Scorer Frequency')
```



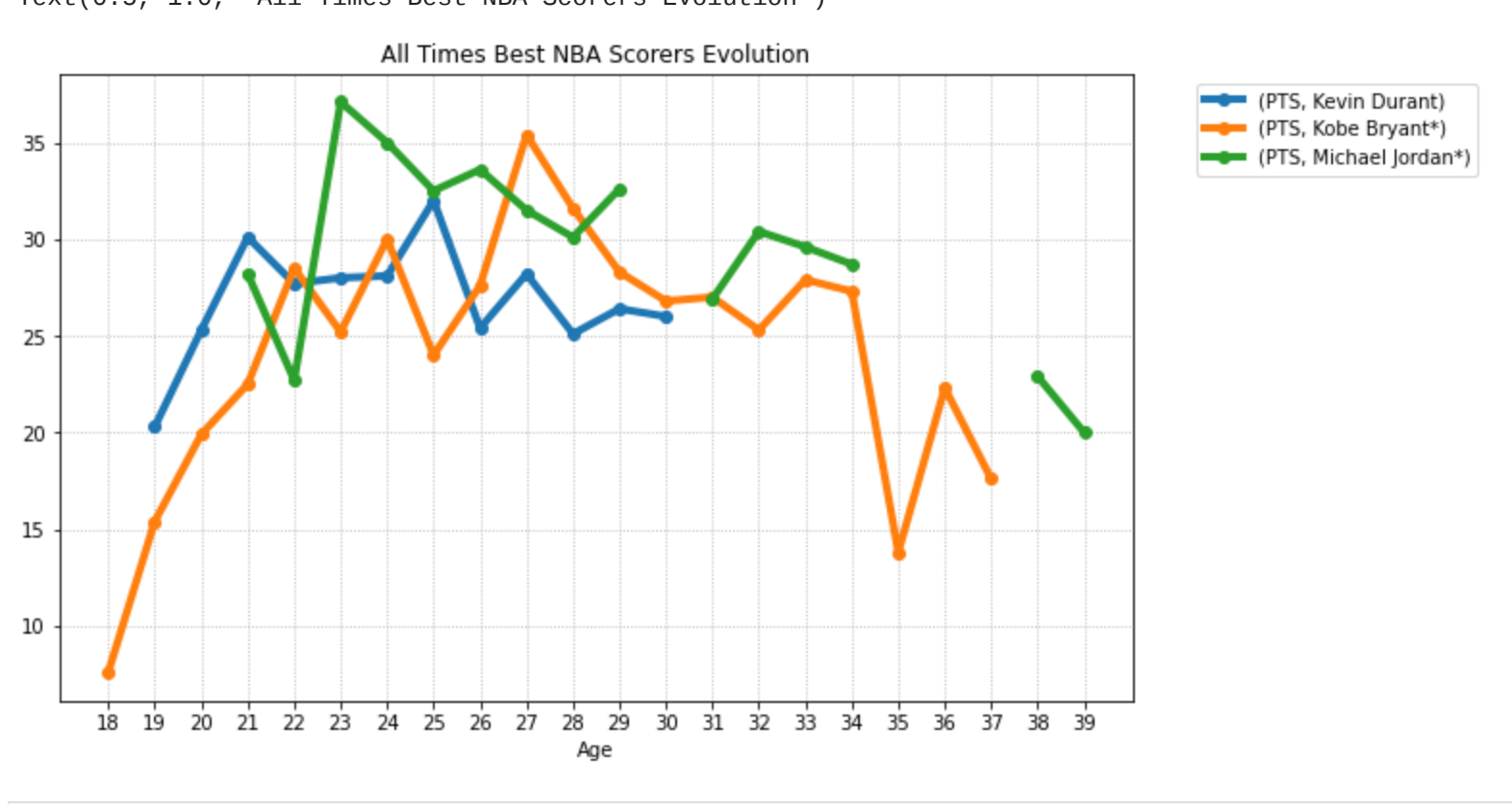
Check the evolution of these players

In [130…

```
df2 = df[['Player','Age','PTS']].loc[df['Player'].isin(['Michael Jordan*','Kevin Durant','Kobe Bryant*'])]
df2 = df2.groupby(['Age','Player']).mean()


fontP = FontProperties()



df2.unstack(1).plot(figsize=(10,6),linewidth=4, marker='o')
plt.xticks(list(range(18,40)))
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', prop=fontP)
plt.grid(linestyle = ':')
plt.title('All Times Best NBA Scorers Evolution')
```

Out[130…

```
Text(0.5, 1.0, 'All Times Best NBA Scorers Evolution')
```



In [69]:

```
df_pts.describe().transpose()
```

Out[69]:

	count	mean	std	min	25%	50%	75%	max
GS	39.0	74.025641	11.717030	22.000	71.5000	79.0000	81.0000	82.000
MP	41.0	38.812195	2.236313	34.200	37.5000	38.9000	40.3000	43.700
FG	41.0	11.014634	1.181220	9.000	10.2000	10.8000	11.9000	13.400
FGA	41.0	22.724390	2.345184	18.200	20.8000	22.6000	24.2000	27.800
FG%	41.0	0.486805	0.049087	0.398	0.4490	0.4860	0.5260	0.583
3P	41.0	1.370732	1.343362	0.000	0.2000	1.3000	2.0000	5.100
3PA	41.0	3.885366	3.467532	0.000	0.9000	3.6000	5.3000	13.200
3P%	40.0	0.300350	0.100869	0.000	0.2775	0.3185	0.3655	0.454
2P	41.0	9.643902	2.191124	5.100	8.1000	9.8000	11.5000	13.200
2PA	41.0	18.846341	4.123172	9.000	16.8000	19.5000	21.3000	27.000
2P%	41.0	0.513293	0.042227	0.419	0.4820	0.5140	0.5480	0.585
eFG%	41.0	0.517268	0.043167	0.422	0.4910	0.5160	0.5460	0.630
FT	41.0	7.704878	1.376400	4.600	6.7000	7.7000	8.7000	10.300
FTA	41.0	9.463415	1.533583	5.100	8.2000	9.8000	10.4000	12.000
FT%	41.0	0.817561	0.078004	0.524	0.8060	0.8370	0.8580	0.908
ORB	41.0	1.648780	0.881227	0.600	1.0000	1.6000	2.0000	4.300
DRB	41.0	5.017073	1.539952	3.100	3.8000	4.7000	5.8000	9.400
TRB	41.0	6.668293	1.975404	3.800	5.4000	6.4000	7.3000	13.600
AST	41.0	5.221951	1.967805	2.400	3.8000	4.8000	6.3000	10.400
STL	41.0	1.817073	0.673388	0.500	1.4000	1.8000	2.3000	3.200
BLK	41.0	0.809756	0.714774	0.000	0.4000	0.7000	1.0000	3.300
TOV	41.0	3.324390	0.743902	2.000	2.7000	3.3000	3.6000	5.400
PF	41.0	2.526829	0.496500	1.700	2.1000	2.5000	2.9000	3.500
PTS	41.0	31.095122	2.307266	26.800	29.8000	30.7000	32.3000	37.100
index	41.0	8356.926829	5111.208812	97.000	3957.0000	8185.0000	12482.0000	17402.000
Season	41.0	2000.000000	11.979149	1980.000	1990.0000	2000.0000	2010.0000	2020.000

6) Application : Create the perfect Five Major

In [70]:

```
df.columns
```

Out[70]:

```
Index(['index', 'Player', 'Pos', 'Age', 'Tm', 'G', 'GS', 'MP', 'FG', 'FGA',
       'FG%', '3P', '3PA', '3P%', '2P', '2PA', '2P%', 'eFG%', 'FT', 'FTA',
       'FT%', 'ORB', 'DRB', 'TRB', 'AST', 'STL', 'BLK', 'TOV', 'PF', 'PTS',
       'Season'],
      dtype='object')
```

In [131…

```
df3 = df.copy()
```

'TRB' means 'Total Rebounds', 'AST' means Assistances, 'STL' means 'Steals', 'BLK' means 'Blocks', 'PF' means Personal Faults, 'PTS' means 'Points'.
We create a KPI 'all' that synthesizes the players perfomances.

In [132…

```
df3['all'] = df3['TRB']+df3['AST']+df3['STL']+df3['BLK']-df3['PF']+df3['PTS']
```

In [133…

```
df3[['all']].head()
```

Out[133…

	all
0	41.9
1	8.5
2	26.2
3	23.7
4	7.8

In [134…

```
df3['Pos'].unique()
```

Out[134…

```
array(['C', 'PF', 'PG', 'SG', 'SF', 'SG-PG', 'SF-SG', 'SG-SF', 'C-PF',
       'PF-C', 'SF-PF', 'PG-SG', 'PF-SF', 'PG-SF', 'SG-PF', 'SF-C'],
      dtype=object)
```

In [135…

```
pos_list = list(df3['Pos'].unique())[:5]
```

In [76]:

```
pos_list
```

Out[76]:

```
['C', 'PF', 'PG', 'SG', 'SF']
```

In [136…

```
pos_list
L = []

for pos in pos_list:

    l = [pos]

    for pos2 in list(df3['Pos'].unique())[5:]:

        if pos in pos2:
            l.append(pos2)

    L.append(l)
```

Thanks to this list we can make selections per position category

In [137…

```
L
```

Out[137…

```
[['C', 'C-PF', 'PF-C', 'SF-C'],
 ['PF', 'C-PF', 'PF-C', 'SF-PF', 'PF-SF', 'SG-PF'],
 ['PG', 'SG-PG', 'PG-SG', 'PG-SF'],
 ['SG', 'SG-PG', 'SF-SG', 'SG-SF', 'PG-SG', 'SG-PF'],
 ['SF', 'SF-SG', 'SG-SF', 'SF-PF', 'PF-SF', 'PG-SF', 'SF-C']]
```

Application : We are in 2011 and I want to have an idea of what will the NBA Oympic Team be in 2012.

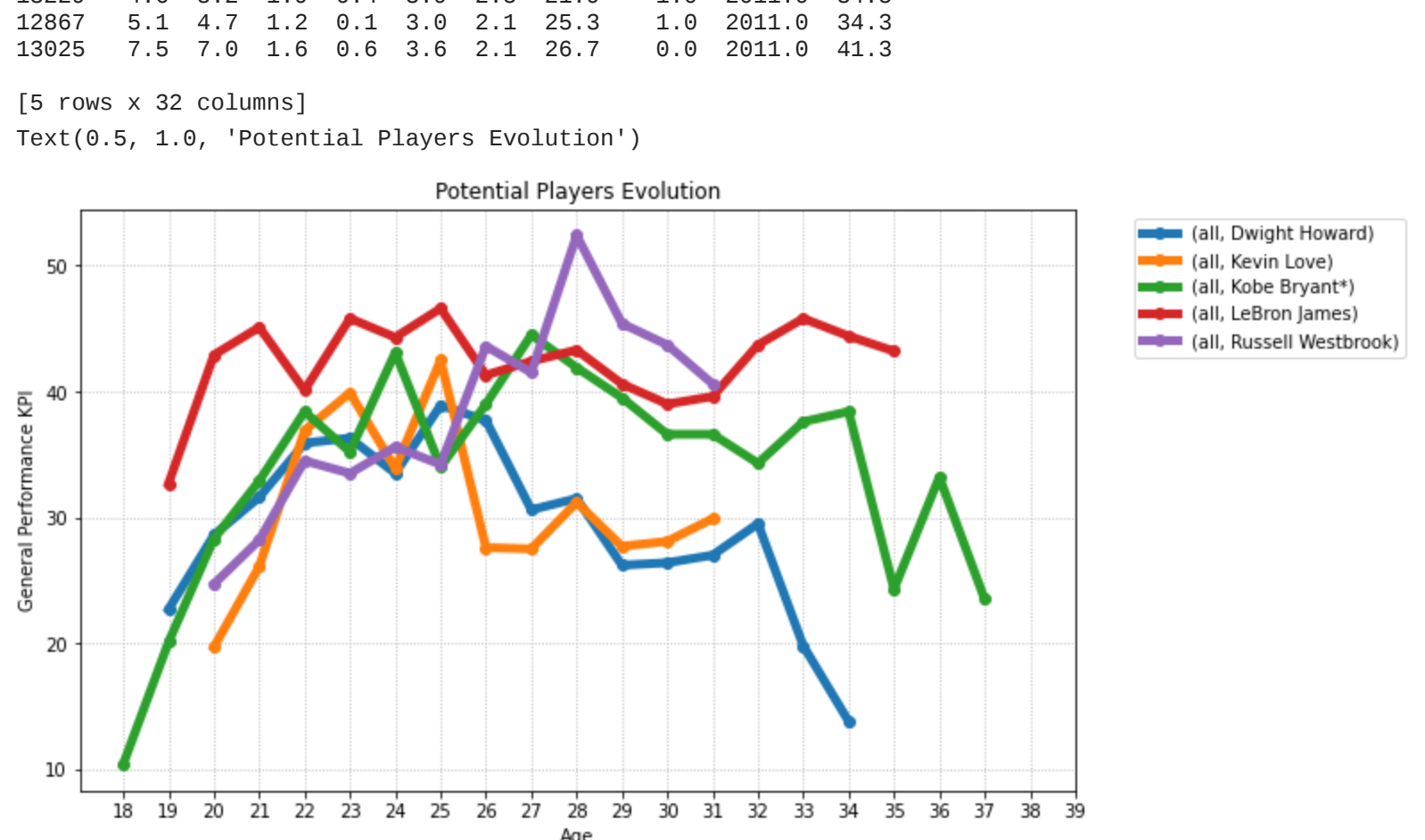
In [141…

```
df_all = pd.DataFrame(columns=column_name)

players_range = 3
year = 2011
age_max = 40
age_min = 18

for item in L:

    df4 = df3.loc[df3['Season'] == year]
    df4 = df4.loc[df4['Pos'].isin(item)]
    df4 = df4.loc[df4['Age'] <= age_max]
    df4 = df4.loc[df4['Age'] >= age_min]
    df4 = df4.nlargest(players_range, ['all'])
    df4['index'] = list(range(players_range))
    df4 = df4.loc[df4['index']==np.random.randint(low=min(df4['index']), high=max(df4['index'])+1)]
    df_all = df_all.append(df4)

print(df_all)

df5 = df3[['Player','Age','all']].loc[df['Player'].isin(list(df_all['Player']))]
df5 = df5.groupby(['Age','Player']).mean()


fontP = FontProperties()



df5.unstack(1).plot(figsize=(10,6),linewidth=5, marker='o',ylabel='General Performance KPI')
plt.xticks(list(range(18,40)))
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', prop=fontP)
plt.grid(linestyle = ':')
plt.title('Potential Players Evolution')
```

```
                 Player Pos Age   Tm   G    GS    MP   FG   FGA    FG%  ...  \
13010      Dwight Howard   C  25  ORL  78  78.0  37.6  7.9  13.4  0.593  ...
13065         Kevin Love  PF  22  MIN  73  73.0  35.8  6.6  14.1  0.470  ...
13229  Russell Westbrook  PG  22  OKC  82  82.0  34.7  7.5  17.0  0.442  ...
12867       Kobe Bryant*  SG  32  LAL  82  82.0  33.9  9.0  20.0  0.451  ...
13025       LeBron James  SF  26  MIA  79  79.0  38.8  9.6  18.8  0.510  ...

        TRB  AST  STL  BLK  TOV   PF   PTS  index  Season   all
13010  14.1  1.4  1.4  2.4  3.6  3.3  22.9    0.0  2011.0  38.9
13065  15.2  2.5  0.6  0.4  2.1  2.0  20.2    0.0  2011.0  36.9
13229   4.6  8.2  1.9  0.4  3.9  2.5  21.9    1.0  2011.0  34.5
12867   5.1  4.7  1.2  0.1  3.0  2.1  25.3    1.0  2011.0  34.3
13025   7.5  7.0  1.6  0.6  3.6  2.1  26.7    0.0  2011.0  41.3

[5 rows x 32 columns]
```

Out[141…

```
Text(0.5, 1.0, 'Potential Players Evolution')
```



Official 2012 Olympic Team :

- Chris Paul
- Deron Williams
- Russell Westbrook
- Kobe Bryant
- James Harden
- LeBron James
- Andre Iguodala
- Kevin Durant
- Carmelo Anthony
- Blake Griffin
- Kevin Love
- Tyson Chandler