Final Project on Basketball

In [8]:
```
import numpy as np
import pandas as pd
import sqlite3
from pandas.api.types import CategoricalDtype


import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

In [9]:
```
pip install cufflinks
```

```
Requirement already satisfied: cufflinks in /anaconda3/lib/python3.7/si
te-packages (0.17.3)
Requirement already satisfied: ipython>=5.3.0 in /anaconda3/lib/python
3.7/site-packages (from cufflinks) (7.6.1)
Requirement already satisfied: six>=1.9.0 in /anaconda3/lib/python3.7/s
ite-packages (from cufflinks) (1.12.0)
Requirement already satisfied: ipywidgets>=7.0.0 in /anaconda3/lib/pyth
on3.7/site-packages (from cufflinks) (7.5.0)
Requirement already satisfied: pandas>=0.19.2 in /anaconda3/lib/python
3.7/site-packages (from cufflinks) (0.24.2)
Requirement already satisfied: colorlover>=0.2.1 in /anaconda3/lib/pyth
on3.7/site-packages (from cufflinks) (0.3.0)
Requirement already satisfied: numpy>=1.9.2 in /anaconda3/lib/python3.
7/site-packages (from cufflinks) (1.16.4)
Requirement already satisfied: plotly>=4.1.1 in /anaconda3/lib/python3.
7/site-packages (from cufflinks) (4.7.1)
Requirement already satisfied: setuptools>=34.4.1 in /anaconda3/lib/pyt
hon3.7/site-packages (from cufflinks) (41.0.1)
Requirement already satisfied: decorator in /anaconda3/lib/python3.7/si
te-packages (from ipython>=5.3.0->cufflinks) (4.4.0)
Requirement already satisfied: pexpect; sys_platform != "win32" in /ana
conda3/lib/python3.7/site-packages (from ipython>=5.3.0->cufflinks) (4.
7.0)
Requirement already satisfied: traitlets>=4.2 in /anaconda3/lib/python
3.7/site-packages (from ipython>=5.3.0->cufflinks) (4.3.2)
Requirement already satisfied: pygments in /anaconda3/lib/python3.7/sit
e-packages (from ipython>=5.3.0->cufflinks) (2.4.2)
Requirement already satisfied: pickleshare in /anaconda3/lib/python3.7/
site-packages (from ipython>=5.3.0->cufflinks) (0.7.5)
Requirement already satisfied: backcall in /anaconda3/lib/python3.7/sit
e-packages (from ipython>=5.3.0->cufflinks) (0.1.0)
Requirement already satisfied: appnope; sys_platform == "darwin" in /an
aconda3/lib/python3.7/site-packages (from ipython>=5.3.0->cufflinks)
(0.1.0)
Requirement already satisfied: jedi>=0.10 in /anaconda3/lib/python3.7/s
ite-packages (from ipython>=5.3.0->cufflinks) (0.13.3)
Requirement already satisfied: prompt-toolkit<2.1.0,>=2.0.0 in /anacond
a3/lib/python3.7/site-packages (from ipython>=5.3.0->cufflinks) (2.0.9)
Requirement already satisfied: nbformat>=4.2.0 in /anaconda3/lib/python
3.7/site-packages (from ipywidgets>=7.0.0->cufflinks) (4.4.0)
Requirement already satisfied: ipykernel>=4.5.1 in /anaconda3/lib/pytho
n3.7/site-packages (from ipywidgets>=7.0.0->cufflinks) (5.1.1)
Requirement already satisfied: widgetsnbextension~=3.5.0 in /anaconda3/
lib/python3.7/site-packages (from ipywidgets>=7.0.0->cufflinks) (3.5.0)
Requirement already satisfied: pytz>=2011k in /anaconda3/lib/python3.7/
site-packages (from pandas>=0.19.2->cufflinks) (2019.1)
Requirement already satisfied: python-dateutil>=2.5.0 in /anaconda3/li
b/python3.7/site-packages (from pandas>=0.19.2->cufflinks) (2.8.0)
Requirement already satisfied: retrying>=1.3.3 in /anaconda3/lib/python
3.7/site-packages (from plotly>=4.1.1->cufflinks) (1.3.3)
Requirement already satisfied: ptyprocess>=0.5 in /anaconda3/lib/python
3.7/site-packages (from pexpect; sys_platform != "win32"->ipython>=5.3.
0->cufflinks) (0.6.0)
Requirement already satisfied: ipython-genutils in /anaconda3/lib/pytho
n3.7/site-packages (from traitlets>=4.2->ipython>=5.3.0->cufflinks) (0.
2.0)
Requirement already satisfied: parso>=0.3.0 in /anaconda3/lib/python3.
```

```
7/site-packages (from jedi>=0.10->ipython>=5.3.0->cufflinks) (0.5.0)
Requirement already satisfied: wcwidth in /anaconda3/lib/python3.7/site
-packages (from prompt-toolkit<2.1.0,>=2.0.0->ipython>=5.3.0->cufflink
s) (0.1.7)
Requirement already satisfied: jupyter-core in /anaconda3/lib/python3.
7/site-packages (from nbformat>=4.2.0->ipywidgets>=7.0.0->cufflinks)
(4.5.0)
Requirement already satisfied: jsonschema!=2.5.0,>=2.4 in /anaconda3/li
b/python3.7/site-packages (from nbformat>=4.2.0->ipywidgets>=7.0.0->cuf
flinks) (3.0.1)
Requirement already satisfied: tornado>=4.2 in /anaconda3/lib/python3.
7/site-packages (from ipykernel>=4.5.1->ipywidgets>=7.0.0->cufflinks)
(6.0.3)
Requirement already satisfied: jupyter-client in /anaconda3/lib/python
3.7/site-packages (from ipykernel>=4.5.1->ipywidgets>=7.0.0->cufflinks)
(5.3.1)
Requirement already satisfied: notebook>=4.4.1 in /anaconda3/lib/python
3.7/site-packages (from widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->c
ufflinks) (6.0.0)
Requirement already satisfied: attrs>=17.4.0 in /anaconda3/lib/python3.
7/site-packages (from jsonschema!=2.5.0,>=2.4->nbformat>=4.2.0->ipywidg
ets>=7.0.0->cufflinks) (19.1.0)
Requirement already satisfied: pyrsistent>=0.14.0 in /anaconda3/lib/pyt
hon3.7/site-packages (from jsonschema!=2.5.0,>=2.4->nbformat>=4.2.0->ip
ywidgets>=7.0.0->cufflinks) (0.14.11)
Requirement already satisfied: pyzmq>=13 in /anaconda3/lib/python3.7/si
te-packages (from jupyter-client->ipykernel>=4.5.1->ipywidgets>=7.0.0->
cufflinks) (18.0.0)
Requirement already satisfied: nbconvert in /anaconda3/lib/python3.7/si
te-packages (from notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidget
s>=7.0.0->cufflinks) (5.5.0)
Requirement already satisfied: jinja2 in /anaconda3/lib/python3.7/site-
packages (from notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=
7.0.0->cufflinks) (2.10.1)
Requirement already satisfied: terminado>=0.8.1 in /anaconda3/lib/pytho
n3.7/site-packages (from notebook>=4.4.1->widgetsnbextension~=3.5.0->ip
ywidgets>=7.0.0->cufflinks) (0.8.2)
Requirement already satisfied: Send2Trash in /anaconda3/lib/python3.7/s
ite-packages (from notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidge
ts>=7.0.0->cufflinks) (1.5.0)
Requirement already satisfied: prometheus-client in /anaconda3/lib/pyth
on3.7/site-packages (from notebook>=4.4.1->widgetsnbextension~=3.5.0->i
pywidgets>=7.0.0->cufflinks) (0.7.1)
Requirement already satisfied: defusedxml in /anaconda3/lib/python3.7/s
ite-packages (from nbconvert->notebook>=4.4.1->widgetsnbextension~=3.5.
0->ipywidgets>=7.0.0->cufflinks) (0.6.0)
Requirement already satisfied: mistune>=0.8.1 in /anaconda3/lib/python
3.7/site-packages (from nbconvert->notebook>=4.4.1->widgetsnbextension~
=3.5.0->ipywidgets>=7.0.0->cufflinks) (0.8.4)
Requirement already satisfied: testpath in /anaconda3/lib/python3.7/sit
e-packages (from nbconvert->notebook>=4.4.1->widgetsnbextension~=3.5.0-
>ipywidgets>=7.0.0->cufflinks) (0.4.2)
Requirement already satisfied: entrypoints>=0.2.2 in /anaconda3/lib/pyt
hon3.7/site-packages (from nbconvert->notebook>=4.4.1->widgetsnbextensi
on~=3.5.0->ipywidgets>=7.0.0->cufflinks) (0.3)
Requirement already satisfied: pandocfilters>=1.4.1 in /anaconda3/lib/p
ython3.7/site-packages (from nbconvert->notebook>=4.4.1->widgetsnbexten
```

```
sion~=3.5.0->ipywidgets>=7.0.0->cufflinks) (1.4.2)
Requirement already satisfied: bleach in /anaconda3/lib/python3.7/site-
packages (from nbconvert->notebook>=4.4.1->widgetsnbextension~=3.5.0->i
pywidgets>=7.0.0->cufflinks) (3.1.0)
Requirement already satisfied: MarkupSafe>=0.23 in /anaconda3/lib/pytho
n3.7/site-packages (from jinja2->notebook>=4.4.1->widgetsnbextension~=
3.5.0->ipywidgets>=7.0.0->cufflinks) (1.1.1)
Requirement already satisfied: webencodings in /anaconda3/lib/python3.
7/site-packages (from bleach->nbconvert->notebook>=4.4.1->widgetsnbexte
nsion~=3.5.0->ipywidgets>=7.0.0->cufflinks) (0.5.1)
Note: you may need to restart the kernel to use updated packages.
```

In [10]:
```
%matplotlib inline
%pprint
```

```
Pretty printing has been turned OFF
```

In [11]:
```
import plotly
import plotly.graph_objs as go
import numpy as np
```

In [12]:
```
import plotly.offline as py
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import plotly.figure_factory as ff
import cufflinks as cf
cf.set_config_file(offline=True, sharing=False, theme='ggplot');
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.colors import ListedColormap

import matplotlib.gridspec as gridspec
```

In [13]:
```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression, LogisticRegressionC
V
```

In [15]:
```
TeamBoxScore = pd.read_csv('/Users/feliciapramudji/Desktop/2012-18_teamB
oxScore.csv')
TeamBoxScore.head()
```

Out[15]:

	gmDate	gmTime	seasTyp	offLNm1	offFNm1	offLNm2	offFNm2	offLNm3	offFNm3	te
0	10/30/12	19:00	Regular	Brothers	Tony	Smith	Michael	Workman	Haywoode	
1	10/30/12	19:00	Regular	Brothers	Tony	Smith	Michael	Workman	Haywoode	
2	10/30/12	20:00	Regular	McCutchen	Monty	Wright	Sean	Fitzgerald	Kane	
3	10/30/12	20:00	Regular	McCutchen	Monty	Wright	Sean	Fitzgerald	Kane	
4	10/30/12	22:30	Regular	Foster	Scott	Zielinski	Gary	Dalen	Eric	

5 rows × 123 columns

In [9]:
```
Official_Box_Score = pd.read_csv('/Users/feliciapramudji/Desktop/2012-18
_officialBoxScore (1).csv')
```

In [10]:
```
Standings = pd.read_csv('/Users/feliciapramudji/Desktop/2012-18_standing
s.csv')
Standings.head()
```

Out[10]:

	stDate	teamAbbr	rank	rankOrd	gameWon	gameLost	stk	stkType	stkTot	gameBack	..
0	10/30/12	ATL	3	3rd	0	0	-	-	0	0.5	..
1	10/30/12	BKN	3	3rd	0	0	-	-	0	0.5	..
2	10/30/12	BOS	14	14th	0	1	L1	loss	1	1.0	..
3	10/30/12	CHA	3	3rd	0	0	-	-	0	0.5	..
4	10/30/12	CHI	3	3rd	0	0	-	-	0	0.5	..

5 rows × 39 columns

In [11]:
```
College = pd.read_csv('/Users/feliciapramudji/Desktop/college.csv')
College.head()
```

Out[11]:

	Unnamed: 0	active_from	active_to	birth_date	college	height	name	position	
0	0	1991	1995	24-Jun-68	Duke University	10-Jun	Alaa Abdelnaby	F-C	/players/
1	1	1969	1978	7-Apr-46	Iowa State University	9-Jun	Zaid Abdul-Aziz	C-F	/players/a
2	2	1970	1989	16-Apr-47	University of California, Los Angeles	2-Jul	Kareem Abdul-Jabbar	C	/players/a
3	3	1991	2001	9-Mar-69	Louisiana State University	1-Jun	Mahmoud Abdul-Rauf	G	/players/a/
4	4	1998	2003	3-Nov-74	University of Michigan, San Jose State University	6-Jun	Tariq Abdul-Wahad	F	/players/a

5 rows × 34 columns

In [12]:
```
playerBoxScore = pd.read_csv('2012-18_playerBoxScore.csv')
playerBoxScore.head()
```

Out[12]:

	gmDate	gmTime	seasTyp	playLNm	playFNm	teamAbbr	teamConf	teamDiv	teamLoc	te
0	2012-10-30	19:00	Regular	Price	A.J.	WAS	East	Southeast	Away	
1	2012-10-30	19:00	Regular	Ariza	Trevor	WAS	East	Southeast	Away	
2	2012-10-30	19:00	Regular	Okafor	Emeka	WAS	East	Southeast	Away	
3	2012-10-30	19:00	Regular	Beal	Bradley	WAS	East	Southeast	Away	
4	2012-10-30	19:00	Regular	Booker	Trevor	WAS	East	Southeast	Away	

5 rows × 51 columns

In [13]:
```
Official_Box_Score.head()
```

Out[13]:

	gmDate	gmTime	seasTyp	offLNm	offFNm	teamAbbr	teamConf	teamDiv	teamLoc
0	10/30/12	19:00	Regular	Brothers	Tony	WAS	East	Southeast	Away
1	10/30/12	19:00	Regular	Smith	Michael	WAS	East	Southeast	Away
2	10/30/12	19:00	Regular	Workman	Haywoode	WAS	East	Southeast	Away
3	10/30/12	19:00	Regular	Brothers	Tony	CLE	East	Central	Home
4	10/30/12	19:00	Regular	Smith	Michael	CLE	East	Central	Home

5 rows × 119 columns

In [14]:
```
factors_in_game_result = Official_Box_Score.groupby("teamRslt").count()
factors_in_game_result
```

Out[14]:

teamRslt	gmDate	gmTime	seasTyp	offLNm	offFNm	teamAbbr	teamConf	teamDiv	teamLoc
Loss	22142	22142	22142	22142	22142	22142	22142	22142	22142
Win	22142	22142	22142	22142	22142	22142	22142	22142	22142

2 rows × 118 columns

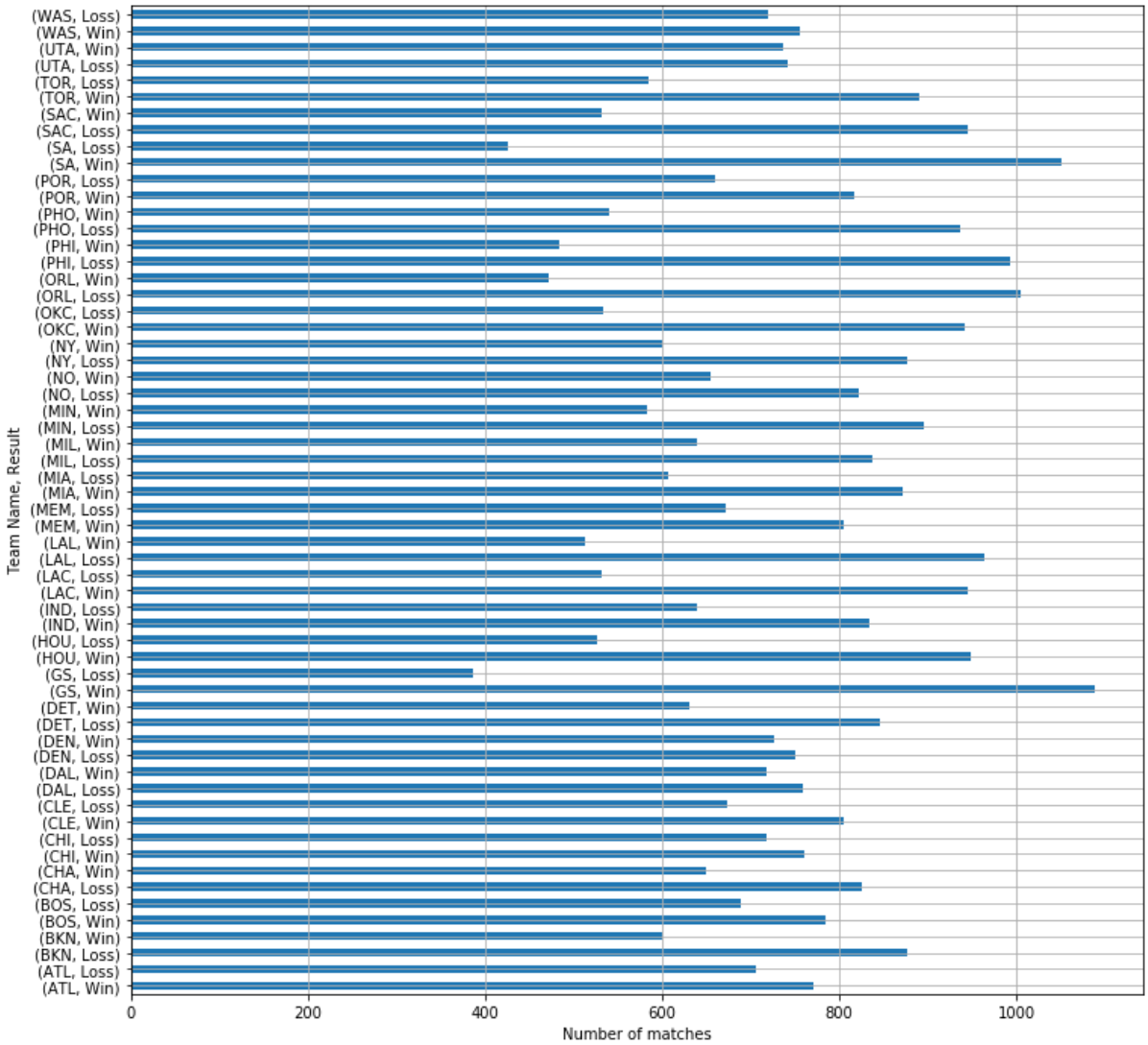
In [15]:
```
Counted_result_based_on_teamLoc = Official_Box_Score.groupby('teamRslt')
.teamLoc.value_counts()
Counted_result_based_on_teamLoc
```

Out[15]:
```
teamRslt  teamLoc
Loss      Away       12983
          Home        9159
Win       Home       12983
          Away        9159
Name: teamLoc, dtype: int64
```

Since we want to find out which team has the highest number of matches win/ lost, we are going to plot a bar chart

In [16]:
```
plt.figure(figsize=(12,12))
Official_Box_Score.groupby('teamAbbr').teamRslt.value_counts().plot.barh
()
plt.grid(True)
plt.xlabel("Number of matches")
plt.ylabel("Team Name, Result")
```

Out[16]: Text(0, 0.5, 'Team Name, Result')



To make sure that the data is cleaned and there is no null value, we run the code below to find out the sum of the null values in each corresponding feature

In [17]:
```
Official_Box_Score.isnull().sum()
```

Out[17]:
```
gmDate        0
gmTime        0
seasTyp       0
offLNm        0
offFNm        0
teamAbbr      0
teamConf      0
teamDiv       0
teamLoc       0
teamRslt      0
teamMin       0
teamDayOff    0
teamPTS       0
teamAST       0
teamTO        0
teamSTL       0
teamBLK       0
teamPF        0
teamFGA       0
teamFGM       0
teamFG%       0
team2PA       0
team2PM       0
team2P%       0
team3PA       0
team3PM       0
team3P%       0
teamFTA       0
teamFTM       0
teamFT%       0
              ..
opptPTS1      0
opptPTS2      0
opptPTS3      0
opptPTS4      0
opptPTS5      0
opptPTS6      0
opptPTS7      0
opptPTS8      0
opptTREB%     0
opptASST%     0
opptTS%       0
opptEFG%      0
opptOREB%     0
opptDREB%     0
opptTO%       0
opptSTL%      0
opptBLK%      0
opptBLKR      0
opptPPS       0
opptFIC       0
opptFIC40     0
opptOrtg      0
opptDrtg      0
opptEDiff     0
opptPlay%     0
opptAR        0
```

```
opptAST/TO     0
opptSTL/TO     0
poss           0
pace           0
Length: 119, dtype: int64
```

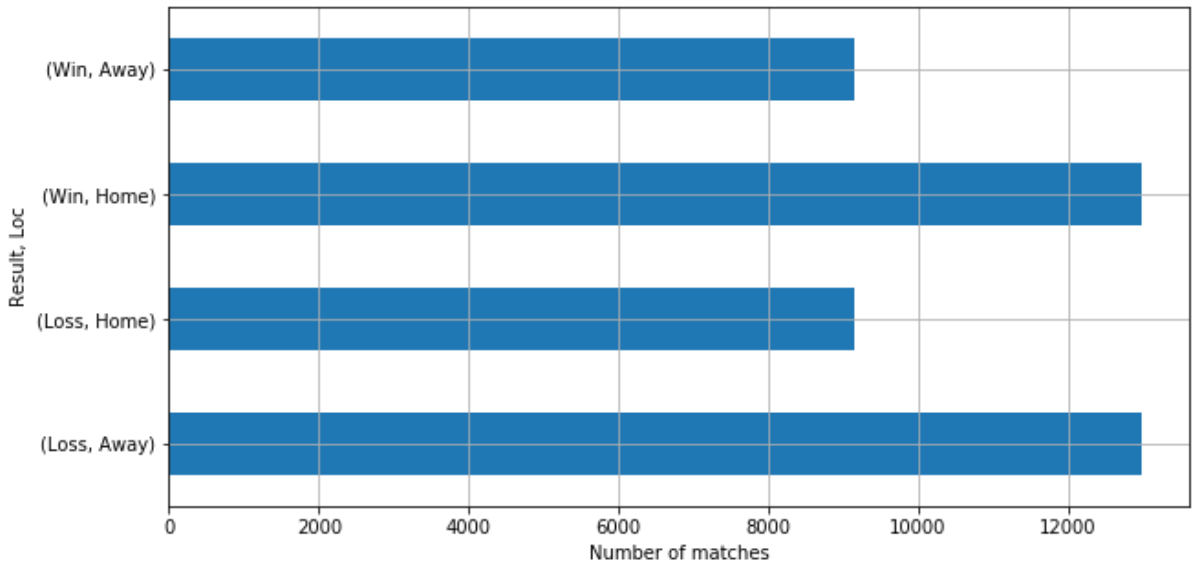
To see the number of matches win or lost by the home and away teams:

In [18]:
```
Official_Box_Score.groupby('teamRslt').teamLoc.value_counts()
```

Out[18]:
```
teamRslt  teamLoc
Loss      Away       12983
          Home        9159
Win       Home       12983
          Away        9159
Name: teamLoc, dtype: int64
```

In [19]:
```
plt.figure(figsize=(10,5))
Official_Box_Score.groupby('teamRslt').teamLoc.value_counts().plot.barh
()


plt.grid(True)
plt.xlabel("Number of matches")
plt.ylabel("Result, Loc")
```

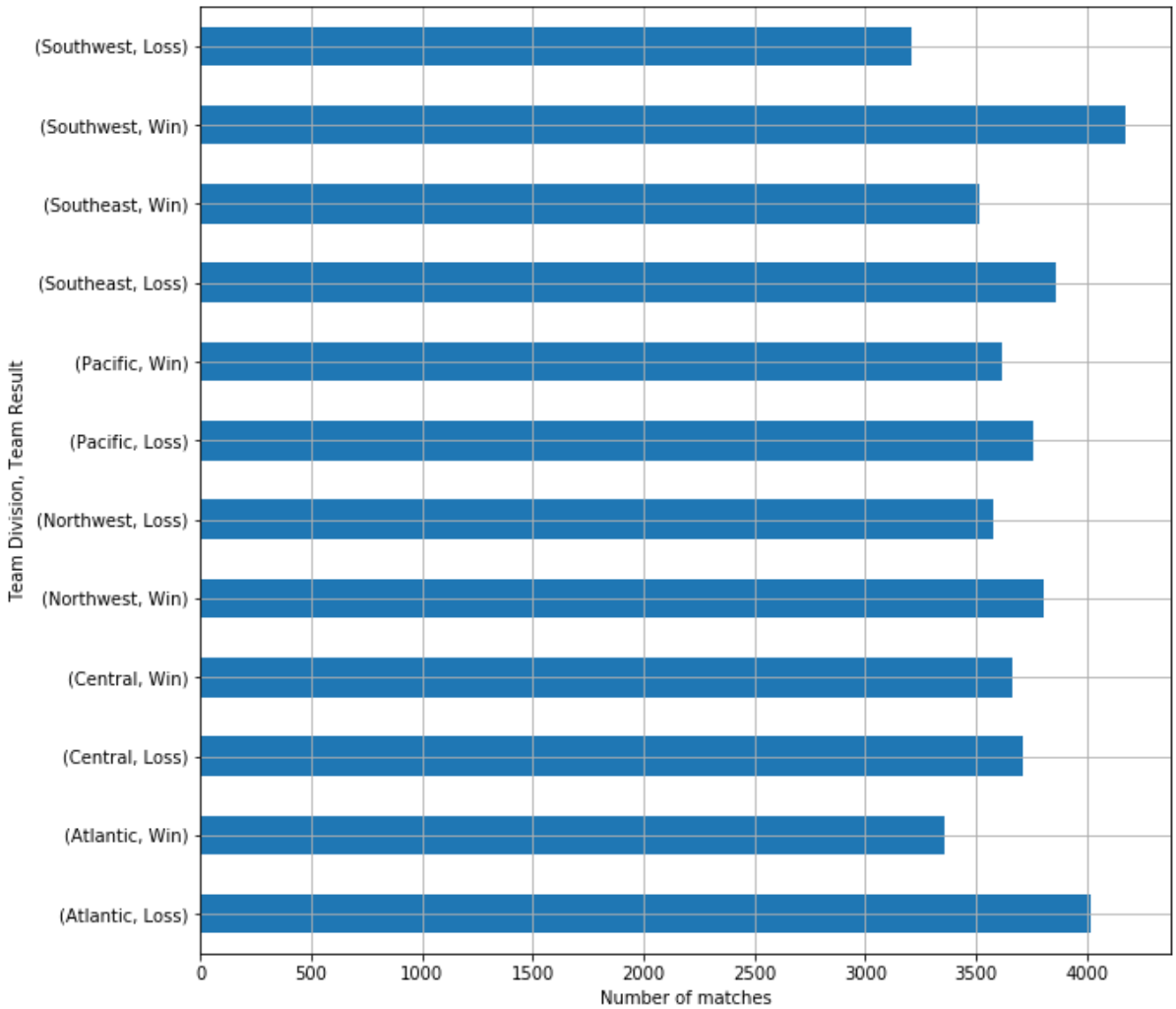
Out[19]:
```
Text(0, 0.5, 'Result, Loc')
```



To see the number of matches won or lost made by the teams from different division:

In [20]:
```
plt.figure(figsize=(10,10))
Official_Box_Score.groupby('teamDiv').teamRslt.value_counts().plot.barh
()
plt.grid(True)
plt.xlabel("Number of matches")
plt.ylabel("Team Division, Team Result")
```

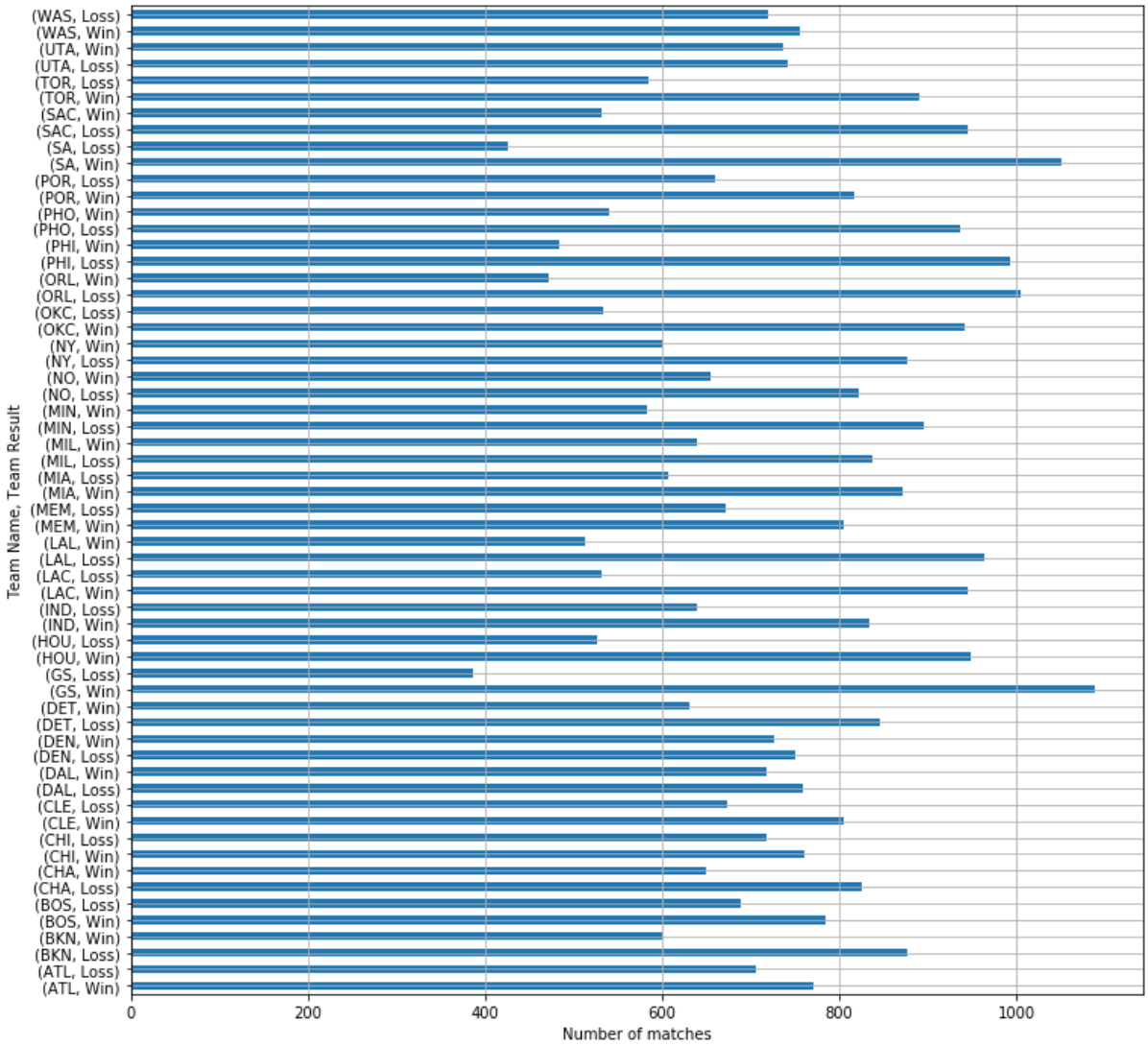
Out[20]: Text(0, 0.5, 'Team Division, Team Result')



From here we can observe that the highest number of matches won are made by the teams from Central division, and the highest number of the matches lost are made by the teams from Atlantic divison.

In [21]:
```
plt.figure(figsize=(12,12))
Official_Box_Score.groupby('teamAbbr').teamRslt.value_counts().plot.barh
()

plt.grid(True)
plt.xlabel("Number of matches")
plt.ylabel("Team Name, Team Result")
```

Out[21]: Text(0, 0.5, 'Team Name, Team Result')



From the above bar plot, we can see that the highest number of matches won is by made by team GS, while the second highest number is made by team SA

In [22]:
```
df_gs = Official_Box_Score.loc[Official_Box_Score["teamAbbr"]=='GS']
df_gs.head()
```

Out[22]:

	gmDate	gmTime	seasTyp	offLNm	offFNm	teamAbbr	teamConf	teamDiv	teamLoc	tea
54	10/31/12	22:00	Regular	Adams	Bennie	GS	West	Pacific	Away	
55	10/31/12	22:00	Regular	Callahan	Mike	GS	West	Pacific	Away	
56	10/31/12	22:00	Regular	Cutler	Kevin	GS	West	Pacific	Away	
147	11/2/12	22:30	Regular	Mauer	Ken	GS	West	Pacific	Home	
148	11/2/12	22:30	Regular	Wood	Leon	GS	West	Pacific	Home	

5 rows × 119 columns

In [23]:
```
# total number of matches won and lost
Result = df_gs["teamRslt"].value_counts()
print(Result)
```

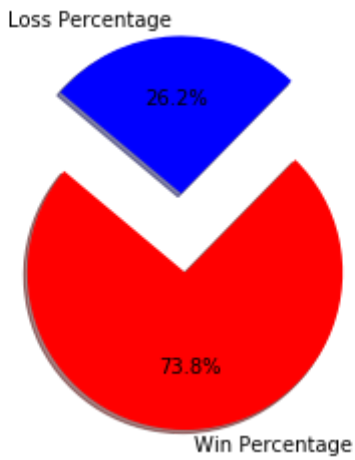
```
Win     1089
Loss     387
Name: teamRslt, dtype: int64
```

The total no. of matches won and lost by team GS are 1089 and 387 respectively

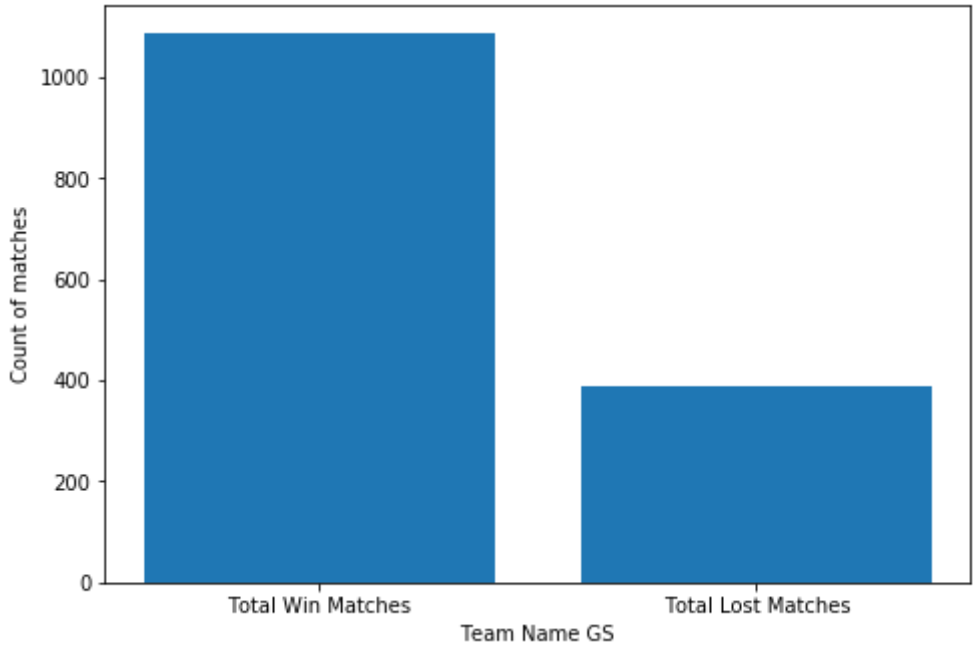
In [24]:
```
# number of matches played by the team
Total=Result.sum()
print('Total=',Total)


Win_per=(Result[0]/Total)*100
Loss_per=(Result[1]/Total)*100
print("Win Percentage=",Win_per,'%')
print("Loss Percentage=",Loss_per,'%')
```

```
Total= 1476
Win Percentage= 73.78048780487805 %
Loss Percentage= 26.21951219512195 %
```

In [25]:
```
# plot a pie-chart for the number of matches won and lost by the team GS

# Data to plot
labels = 'Win Percentage', 'Loss Percentage'
sizes = [Win_per, Loss_per]
colors = ['red', 'blue']
explode = (0.3, 0.2)  # explode 1st slice

# Plot
plt.pie(sizes, explode=explode, labels=labels, colors=colors,
autopct='%1.1f%%', shadow=True, startangle=140)

plt.axis('equal')
plt.show()
```



In [26]:
```
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
labels = ['Total Win Matches', 'Total Lost Matches']
count_matches = [Result[0],Result[1]]
plt.xlabel("Team Name GS")
plt.ylabel("Count of matches")
ax.bar(labels,count_matches)

plt.show()
```



In [27]:
```
Total_win_gs_per=(Result[0]/44284)*100
Total_loss_gs_per=(Result[1]/44284)*100

print("The total percentage of the matche won by GS is",Total_win_gs_per
,'%')
print("The total percentage of the matche loss by GS is",Total_loss_gs_p
er,'%')
```

```
The total percentage of the matche won by GS is 2.4591274500948423 %
The total percentage of the matche loss by GS is 0.8739047963146961 %
```

In [28]:
```
#group and sum the matches won and lost by the home and away teams.

df2=df_gs.groupby("teamLoc").teamRslt.value_counts()
df2
```

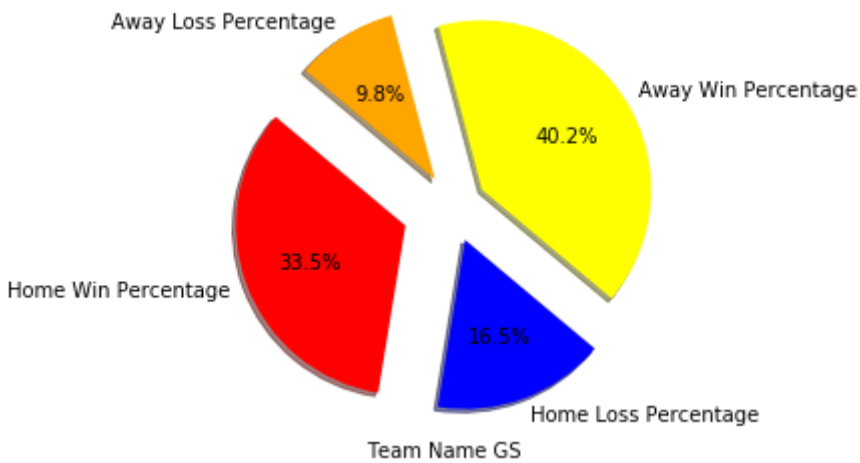
Out[28]:
```
teamLoc  teamRslt
Away     Win         495
         Loss        243
Home     Win         594
         Loss        144
Name: teamRslt, dtype: int64
```

From the result, we observe that teams Away have won and lost 495 and 243 matches respectively. Also, From the above result we can observe that the teams at home have won and lost 594 and 144 matches respectively.

In [29]:
```
df2=np.asarray(df2)
df2
```

Out[29]:
```
array([495, 243, 594, 144])
```

In [30]:
```
#Plotting a pie chart according to the above data

# Data to plot
labels = 'Home Win Percentage', 'Home Loss Percentage','Away Win Percent
age', 'Away Loss Percentage'
sizes = [df2[0],df2[1],df2[2],df2[3]]
colors = ['red', 'blue','yellow','orange']
explode = (0.3, 0.2,0.2,0.2)  # explode 1st slice

# Plot
plt.pie(sizes, explode=explode, labels=labels, colors=colors,
autopct='%1.1f%%', shadow=True, startangle=140)
plt.xlabel("Team Name GS")
plt.axis('equal')
plt.show()
```



Now we look at team SA

In [31]:
```
df_sa = Official_Box_Score.loc[Official_Box_Score["teamAbbr"]=='SA']
df_sa.head()
```

Out[31]:

	gmDate	gmTime	seasTyp	offLNm	offFNm	teamAbbr	teamConf	teamDiv	teamLoc	te
42	10/31/12	20:00	Regular	Fraher	Pat	SA	West	Southwest	Away	
43	10/31/12	20:00	Regular	Goble	John	SA	West	Southwest	Away	
44	10/31/12	20:00	Regular	Capers	James	SA	West	Southwest	Away	
75	11/1/12	21:30	Regular	Crawford	Dan	SA	West	Southwest	Home	
76	11/1/12	21:30	Regular	Collins	Derrick	SA	West	Southwest	Home	

5 rows × 119 columns

In [32]:
```
Result_sa=df_sa["teamRslt"].value_counts()
print(Result_sa)
Result=np.asarray(Result)
```

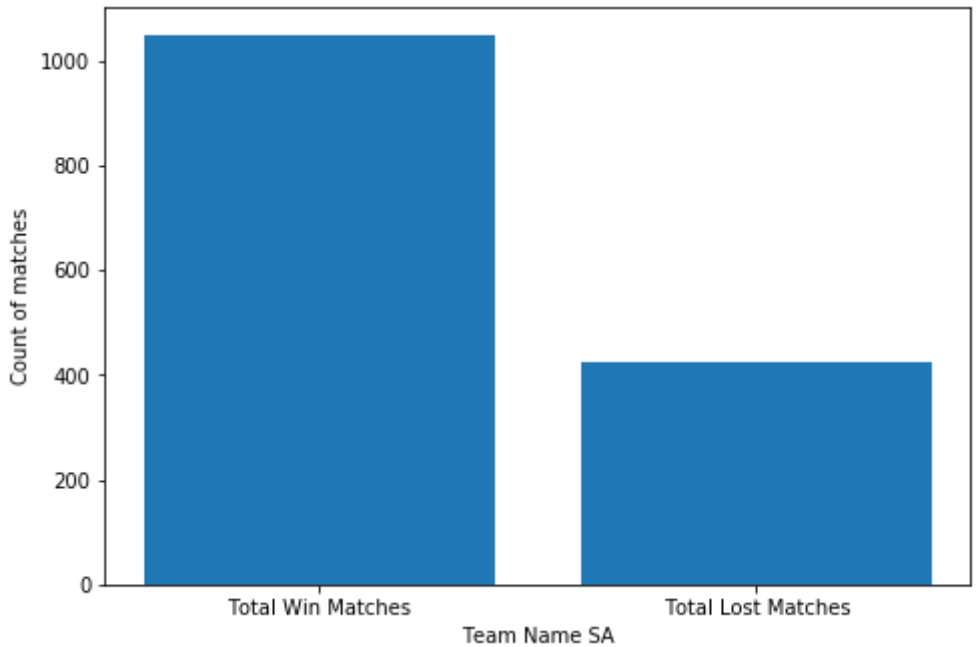
```
Win     1050
Loss     426
Name: teamRslt, dtype: int64
```

The SA team has won 1050 matches and lost 426 matches.

In [33]:
```
Total_sa=Result_sa.sum()
print('Total=',Total)


Win_per_sa=(Result_sa[0]/Total)*100
Loss_per_sa=(Result_sa[1]/Total)*100


print("Win Percentage=",Win_per_sa,'%')
print("Loss Percentage=",Loss_per_sa,'%')
```

```
Total= 1476
Win Percentage= 71.13821138211382 %
Loss Percentage= 28.86178861788618 %
```

In [34]:
```
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
labels_sa = ['Total Win Matches', 'Total Lost Matches']
count_matches_sa = [Result_sa[0],Result_sa[1]]
plt.xlabel("Team Name SA")
plt.ylabel("Count of matches")
ax.bar(labels_sa,count_matches_sa)
plt.show()
```



In [35]:
```
Total_win_sa_per=(Result_sa[0]/44284)*100
Total_loss_sa_per=(Result_sa[1]/44284)*100

print("The total percentage of the matche won by SA is",Total_win_sa_per
,'%')
print("The total percentage of the matche loss by SA is",Total_loss_sa_p
er,'%')
```

```
The total percentage of the matche won by SA is 2.371059524884834 %
The total percentage of the matche loss by SA is 0.9619727215247043 %
```

The team SA has won 2.37% and lost 0.96% matches.

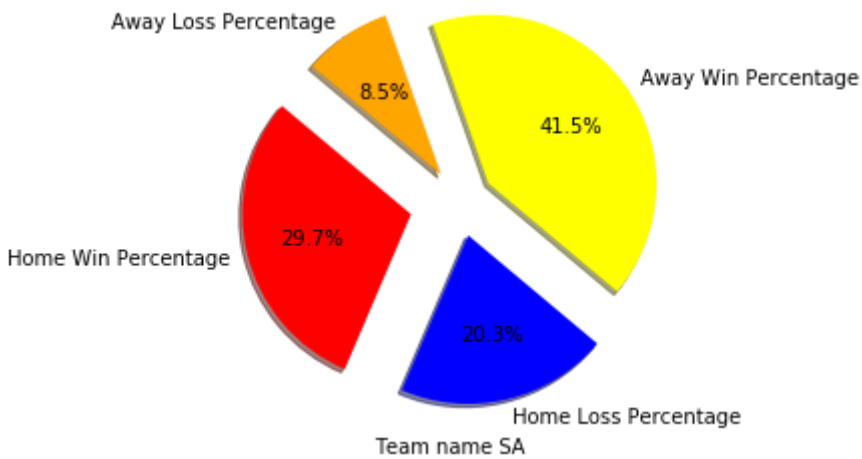
In [36]:
```
df3=df_sa.groupby("teamLoc").teamRslt.value_counts()
df3
```

Out[36]:
```
teamLoc  teamRslt
Away     Win         438
         Loss        300
Home     Win         612
         Loss        126
Name: teamRslt, dtype: int64
```

In [37]:
```
df3=np.asarray(df3)
df3
```

Out[37]:
```
array([438, 300, 612, 126])
```

In [38]:
```
labels = 'Home Win Percentage', 'Home Loss Percentage','Away Win Percent
age', 'Away Loss Percentage'
sizes = [df3[0],df3[1],df3[2],df3[3]]
colors = ['red', 'blue','yellow','orange']
explode = (0.3, 0.2,0.2,0.2)  # explode 1st slice

# Plot
plt.pie(sizes, explode=explode, labels=labels, colors=colors,
autopct='%1.1f%%', shadow=True, startangle=140)
plt.xlabel("Team name SA")
plt.axis('equal')
plt.show()
```



If we look into the details of different games we can observe that whether a team will win or lose is dependent on many factors. We all know that people say that winning or losing is dependent on how the players perform but still statistics tell us a different story. According to the statistics, the performance of a team is based upon many other factors.

Now, we will try to build a model which predicts whether the team will lose or win on the basis of various features to see which features have a greater impact on the result of the game

In [39]:
```
# Since our target feature is 'teamRslt' and we need to convert it into
 0 or 1 for our model to work
from sklearn.preprocessing import LabelEncoder

#This method will convert into 0 or 1 for us
encoder=LabelEncoder() #Creating instance of the method
Official_Box_Score["teamRslt"] = encoder.fit_transform(Official_Box_Scor
e["teamRslt"]) #Calling the instance and passing the data
Official_Box_Score["teamRslt"]
```

Out[39]:

```
0        0
1        0
2        0
3        1
4        1
5        1
6        0
7        0
8        0
9        1
10       1
11       1
12       1
13       1
14       1
15       0
16       0
17       0
18       0
19       0
20       0
21       1
22       1
23       1
24       1
25       1
26       1
27       0
28       0
29       0
        ..
44254    0
44255    0
44256    0
44257    1
44258    1
44259    1
44260    0
44261    0
44262    0
44263    1
44264    1
44265    1
44266    1
44267    1
44268    1
44269    0
44270    0
44271    0
44272    0
44273    0
44274    0
44275    1
44276    1
44277    1
44278    0
44279    0
```

```
44280    0
44281    1
44282    1
44283    1
Name: teamRslt, Length: 44284, dtype: int64
```

In [40]:
```
import seaborn as sns
```

In [41]:
```
corr=Official_Box_Score.corr()
corr.sort_values(by = 'teamRslt', ascending=False)
```

Out[41]:

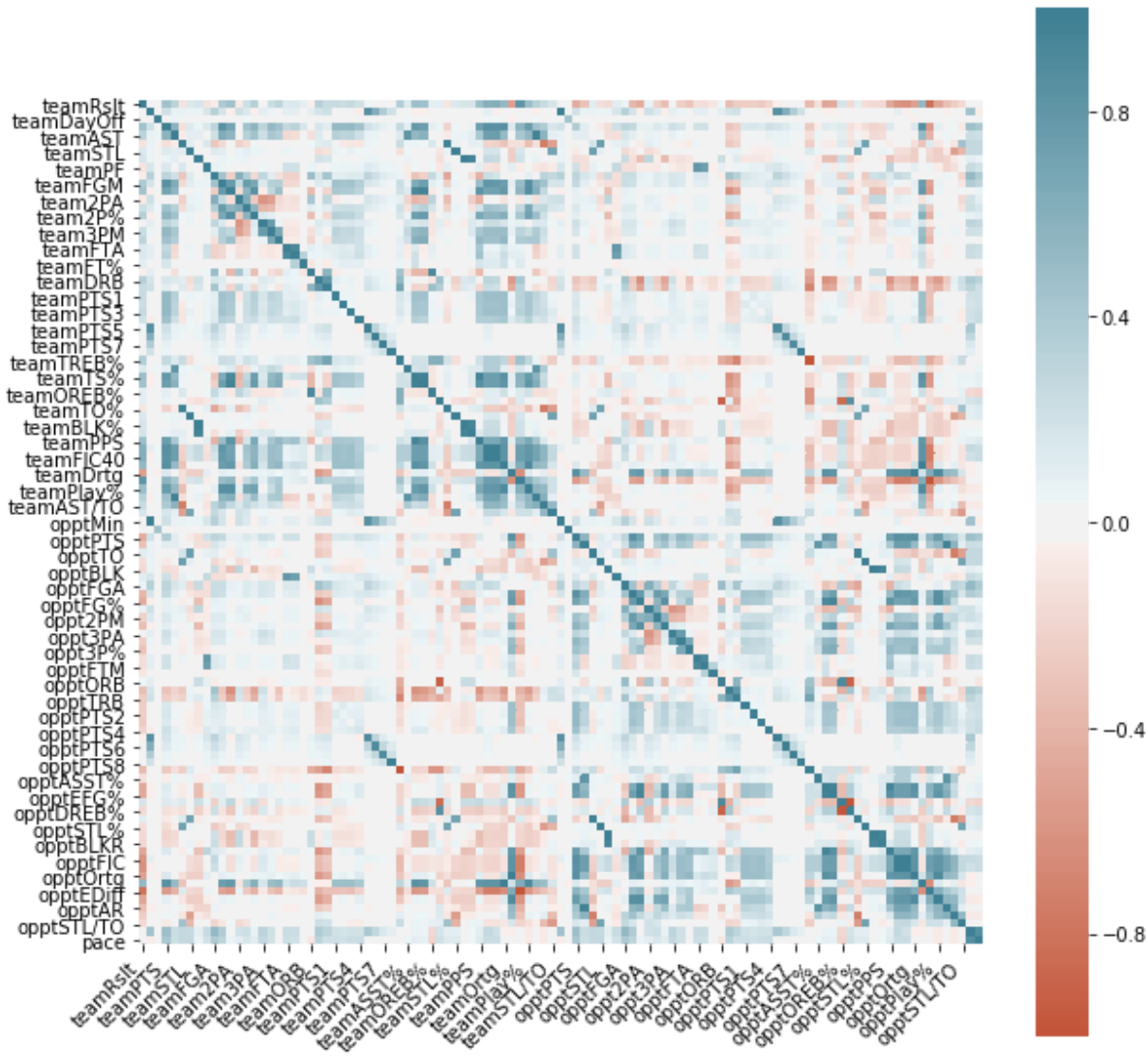
	teamRslt	teamMin	teamDayOff	teamPTS	teamAST	teamTO	teamSTL	teaml
teamRslt	1.000000	0.000270	0.031149	0.453929	0.318070	-0.115121	0.142429	0.174
teamEDiff	0.809808	0.000687	0.037213	0.559446	0.410487	-0.129487	0.170080	0.189
teamFIC40	0.609847	-0.045065	0.039260	0.830936	0.738901	-0.221243	0.177654	0.253
teamFIC	0.607026	0.096734	0.040415	0.858108	0.747806	-0.210217	0.183426	0.261
opptDrtg	0.516026	-0.021850	0.006466	0.859243	0.509720	-0.297283	-0.015803	0.013
teamOrtg	0.516026	-0.021850	0.006466	0.859243	0.509720	-0.297283	-0.015803	0.013
teamPlay%	0.484859	-0.030484	0.024907	0.750634	0.534933	-0.332741	0.000470	0.041
teamTS%	0.483764	-0.040056	0.008836	0.777093	0.516504	0.058689	0.022995	0.033
teamPPS	0.469018	-0.034793	0.010347	0.746601	0.398943	0.083271	0.025954	0.033
teamEFG%	0.460942	-0.042710	0.012953	0.739288	0.579160	0.044237	0.022844	0.034
teamFG%	0.454300	-0.045274	0.025356	0.696209	0.547068	0.032027	0.030943	0.045
teamPTS	0.453929	0.209798	0.015094	1.000000	0.571228	-0.118994	0.118359	0.060
teamFGM	0.383311	0.180612	0.027432	0.836210	0.639515	-0.157303	0.110908	0.065
teamTREB%	0.362426	0.000404	0.024175	0.258988	0.136836	0.242982	-0.134631	0.032
teamDRB	0.339497	0.186723	0.025328	0.157737	0.121016	0.116595	-0.154103	0.192
team2P%	0.337290	-0.041503	0.019073	0.564665	0.438961	0.043138	0.051030	0.035
teamAR	0.332062	-0.042985	0.041159	0.460727	0.955524	-0.123824	0.033091	0.071
teamAST	0.318070	0.077213	0.047634	0.571228	1.000000	-0.047037	0.096791	0.091
team3P%	0.312128	-0.017449	-0.002655	0.459319	0.365226	0.012489	-0.019279	0.013
teamPTS3	0.260623	-0.021499	0.007132	0.515751	0.314043	-0.090483	0.082537	0.023
teamPTS1	0.256625	-0.001236	0.024680	0.515298	0.340081	-0.021013	0.053825	0.045
teamTRB	0.252486	0.211488	0.031006	0.115988	0.029134	0.123472	-0.114640	0.168
teamAST/TO	0.251495	-0.017940	0.010243	0.390057	0.570802	-0.740172	-0.026182	0.027
teamPTS2	0.241888	-0.005956	0.011367	0.514053	0.308605	-0.075379	0.045160	0.030
team3PM	0.229267	0.056003	-0.023577	0.531300	0.408334	-0.005787	0.016618	-0.004
team2PM	0.218945	0.139774	0.043806	0.455724	0.347115	-0.151985	0.098336	0.067
teamPTS4	0.188470	-0.050488	-0.015613	0.464721	0.211709	-0.101639	0.044610	-0.006
teamBLKR	0.183974	0.004152	0.017768	0.059968	0.091997	0.093077	-0.013341	0.948
teamSTL/TO	0.178694	-0.011754	0.016045	0.160827	0.094850	-0.527328	0.696195	-0.013
opptOREB%	0.176312	0.008151	0.023033	0.126963	0.095550	0.132893	-0.001803	0.222

	teamRslt	teamMin	teamDayOff	teamPTS	teamAST	teamTO	teamSTL	teaml
...	...	...	...	...	...	...	...	
opptBLK%	-0.174762	0.010703	-0.016924	-0.209936	-0.159298	-0.031086	-0.008883	0.021
opptDREB%	-0.176312	-0.008151	-0.023033	-0.126963	-0.095550	-0.132893	0.001803	-0.222
opptSTL/TO	-0.178694	-0.010357	0.008259	-0.072487	-0.032688	0.460215	-0.382004	0.012
opptBLKR	-0.183974	0.003436	-0.023308	-0.167286	-0.144546	-0.021498	0.063137	-0.049
opptPTS4	-0.188470	-0.050412	-0.015744	0.217312	0.085183	0.018413	-0.047313	-0.068
oppt2PM	-0.218945	0.139483	0.002908	0.149643	0.070228	0.054715	-0.093282	-0.046
oppt3PM	-0.229267	0.055978	-0.026838	0.172247	0.042718	-0.044187	0.003122	-0.131
opptPTS2	-0.241888	-0.007004	-0.005616	0.168704	0.039252	-0.011279	-0.043710	-0.097
opptAST/TO	-0.251495	-0.017320	-0.026360	0.041040	0.029230	-0.102194	-0.533204	-0.074
opptTRB	-0.252486	0.210808	-0.001655	-0.245569	-0.159628	-0.214748	0.072971	0.123
opptPTS1	-0.256625	-0.001575	-0.015706	0.153336	0.028359	-0.020582	-0.023677	-0.096
opptPTS3	-0.260623	-0.021911	-0.025016	0.138460	0.036716	0.025417	-0.061864	-0.099
oppt3P%	-0.312128	-0.017911	-0.011170	0.030654	-0.026020	-0.067866	-0.006272	-0.097
opptAST	-0.318070	0.076684	-0.012954	0.109944	0.100140	0.032212	-0.010981	-0.132
opptAR	-0.332062	-0.043606	-0.020713	-0.006759	0.041303	-0.034473	-0.079066	-0.204
oppt2P%	-0.337290	-0.042642	-0.035538	0.101642	0.008169	-0.023742	0.041258	-0.360
opptDRB	-0.339497	0.185864	-0.015830	-0.304578	-0.205223	-0.277501	0.068064	-0.022
opptTREB%	-0.362426	-0.000404	-0.024175	-0.258988	-0.136836	-0.242982	0.134631	-0.032
opptFGM	-0.383311	0.180302	-0.016105	0.273012	0.101088	0.023810	-0.091812	-0.140
opptPTS	-0.453929	0.209028	-0.026611	0.372021	0.109944	0.025093	-0.072167	-0.154
opptFG%	-0.454300	-0.046441	-0.029778	0.080683	-0.011460	-0.054819	0.019486	-0.334
opptEFG%	-0.460942	-0.043804	-0.034902	0.103641	-0.007354	-0.068257	0.028540	-0.341
opptPPS	-0.469018	-0.036348	-0.036263	0.145532	-0.012124	-0.053874	0.046185	-0.322
opptTS%	-0.483764	-0.041254	-0.037495	0.127199	-0.009778	-0.065864	0.034255	-0.341
opptPlay%	-0.484859	-0.031248	-0.031688	0.086179	-0.011815	-0.107825	-0.249878	-0.244
teamDrtg	-0.516026	-0.022726	-0.040960	0.146262	-0.013421	-0.132259	-0.232560	-0.228
opptOrtg	-0.516026	-0.022726	-0.040960	0.146262	-0.013421	-0.132259	-0.232560	-0.228
opptFIC	-0.607026	0.095693	-0.030671	0.035850	-0.008424	-0.000369	-0.142337	-0.187
opptFIC40	-0.609847	-0.045321	-0.032033	0.003587	-0.021477	-0.010214	-0.149906	-0.196
opptEDiff	-0.809808	-0.000687	-0.037213	-0.559446	-0.410487	0.129487	-0.170080	-0.189

105 rows × 105 columns

In [42]:
```
corr=Official_Box_Score.corr()


ax = plt.figure(figsize=(10,10))
ax = sns.heatmap(
corr,
vmin=-1, vmax=1, center=0,
cmap=sns.diverging_palette(20, 220, n=200),
square=True
)
ax.set_xticklabels(
ax.get_xticklabels(),
rotation=45,
horizontalalignment='right'
)
```

Out[42]:
```
[Text(0.5, 0, 'teamRslt'), Text(3.5, 0, 'teamPTS'), Text(6.5, 0, 'teamS
TL'), Text(9.5, 0, 'teamFGA'), Text(12.5, 0, 'team2PA'), Text(15.5, 0,
'team3PA'), Text(18.5, 0, 'teamFTA'), Text(21.5, 0, 'teamORB'), Text(2
4.5, 0, 'teamPTS1'), Text(27.5, 0, 'teamPTS4'), Text(30.5, 0, 'teamPTS
7'), Text(33.5, 0, 'teamASST%'), Text(36.5, 0, 'teamOREB%'), Text(39.5,
0, 'teamSTL%'), Text(42.5, 0, 'teamPPS'), Text(45.5, 0, 'teamOrtg'), Te
xt(48.5, 0, 'teamPlay%'), Text(51.5, 0, 'teamSTL/TO'), Text(54.5, 0, 'o
pptPTS'), Text(57.5, 0, 'opptSTL'), Text(60.5, 0, 'opptFGA'), Text(63.
5, 0, 'oppt2PA'), Text(66.5, 0, 'oppt3PA'), Text(69.5, 0, 'opptFTA'), T
ext(72.5, 0, 'opptORB'), Text(75.5, 0, 'opptPTS1'), Text(78.5, 0, 'oppt
PTS4'), Text(81.5, 0, 'opptPTS7'), Text(84.5, 0, 'opptASST%'), Text(87.
5, 0, 'opptOREB%'), Text(90.5, 0, 'opptSTL%'), Text(93.5, 0, 'opptPP
S'), Text(96.5, 0, 'opptOrtg'), Text(99.5, 0, 'opptPlay%'), Text(102.5,
0, 'opptSTL/TO')]
```



As it can be seen from above, 'team Rslt' is highly correlated with features, such as teamRslt, teamEDiff, teamFIC40, teamFIC, opptDrtg, teamOrtg, teamPlay%, teamTS%, teamPPS, teamEFG%. Therefore, we are using these features to predict

First, we declare our input and output variable

In [43]:
```
X = Official_Box_Score[['teamEDiff','teamFIC40', 'teamFIC', 'opptDrtg',
'teamOrtg', 'teamPlay%', 'teamTS%', 'teamPPS', 'teamEFG%']]
y = Official_Box_Score[['teamRslt']]
```

Then we normalize the data

In [44]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler


X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2, rand
om_state=83)

print("Training Data Size: ", len(X_train))
print("Test Data Size: ", len(X_test))

print("Training Data Size: ", len(y_train))
print("Test Data Size: ", len(y_test))



X_train_normalized = StandardScaler().fit_transform(X_train)
X_test_normalized = StandardScaler().fit_transform(X_test)
```

```
Training Data Size:  35427
Test Data Size:  8857
Training Data Size:  35427
Test Data Size:  8857
```

In [45]:
```
X_train_normalized = StandardScaler().fit_transform(X_train)
X_test_normalized = StandardScaler().fit_transform(X_test)
```

In [46]:
```
from sklearn.linear_model import LogisticRegression

lr=LogisticRegression()
```

In [47]:
```
#Training our model using the training data
lr.fit(X_train_normalized,y_train)
# Predict housing prices on the test set
y_pred=lr.predict(X_test_normalized)
```

```
//anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.p
y:432: FutureWarning:

Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to
silence this warning.

//anaconda3/lib/python3.7/site-packages/sklearn/utils/validation.py:72
4: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please chang
e the shape of y to (n_samples, ), for example using ravel().
```

In [ ]:

In [48]:
```
from sklearn.metrics import accuracy_score
```

In [49]:
```
#Evaluating the accuracy
accuracy = accuracy_score(y_pred,y_test)
print("The accuracy score is",accuracy*100)
```

```
The accuracy score is 100.0
```

This is suprising as we know that a model, no matter how good, should not have an accuracy score of 100%--especially given that there are so many datas in this dataset. Therefore, we tried trials and errors on why this might be the case. All our efforts came to being fruitful, as we figured that we should not have put teamEDiff in our input

In [50]:
```
X = Official_Box_Score[['teamFIC40', 'teamFIC', 'opptDrtg', 'teamOrtg',
'teamPlay%', 'teamTS%', 'teamPPS', 'teamEFG%']]
y = Official_Box_Score[['teamRslt']]
```

In [51]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler


X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2, rand
om_state=83)

print("Training Data Size: ", len(X_train))
print("Test Data Size: ", len(X_test))

print("Training Data Size: ", len(y_train))
print("Test Data Size: ", len(y_test))



X_train_normalized = StandardScaler().fit_transform(X_train)
X_test_normalized = StandardScaler().fit_transform(X_test)
```

```
Training Data Size:  35427
Test Data Size:  8857
Training Data Size:  35427
Test Data Size:  8857
```

In [52]:
```
X_train_normalized = StandardScaler().fit_transform(X_train)
X_test_normalized = StandardScaler().fit_transform(X_test)
```

In [53]:
```
from sklearn.linear_model import LogisticRegression

lr=LogisticRegression()
#Training our model using the training data
lr.fit(X_train_normalized,y_train)
# Predict housing prices on the test set
y_pred=lr.predict(X_test_normalized)
```

```
//anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.p
y:432: FutureWarning:

Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to
silence this warning.

//anaconda3/lib/python3.7/site-packages/sklearn/utils/validation.py:72
4: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please chang
e the shape of y to (n_samples, ), for example using ravel().
```

In [54]:
```
#Evaluating the accuracy
accuracy = accuracy_score(y_pred,y_test)
print("The accuracy score is",accuracy*100)
```

```
The accuracy score is 78.76256068646269
```

Now, we will be exploring the playerBoxScore data

In [55]:
```
#Giving a look to our data
playerBoxScore.head()
```

Out[55]:

	gmDate	gmTime	seasTyp	playLNm	playFNm	teamAbbr	teamConf	teamDiv	teamLoc	te
0	2012-10-30	19:00	Regular	Price	A.J.	WAS	East	Southeast	Away	
1	2012-10-30	19:00	Regular	Ariza	Trevor	WAS	East	Southeast	Away	
2	2012-10-30	19:00	Regular	Okafor	Emeka	WAS	East	Southeast	Away	
3	2012-10-30	19:00	Regular	Beal	Bradley	WAS	East	Southeast	Away	
4	2012-10-30	19:00	Regular	Booker	Trevor	WAS	East	Southeast	Away	

5 rows × 51 columns

In [56]:
```
playerBoxScore.columns
```

Out[56]:
```
Index(['gmDate', 'gmTime', 'seasTyp', 'playLNm', 'playFNm', 'teamAbbr',
       'teamConf', 'teamDiv', 'teamLoc', 'teamRslt', 'teamDayOff', 'off
LNm1',
       'offFNm1', 'offLNm2', 'offFNm2', 'offLNm3', 'offFNm3', 'playDisp
Nm',
       'playStat', 'playMin', 'playPos', 'playHeight', 'playWeight',
       'playBDate', 'playPTS', 'playAST', 'playTO', 'playSTL', 'playBL
K',
       'playPF', 'playFGA', 'playFGM', 'playFG%', 'play2PA', 'play2PM',
       'play2P%', 'play3PA', 'play3PM', 'play3P%', 'playFTA', 'playFT
M',
       'playFT%', 'playORB', 'playDRB', 'playTRB', 'opptAbbr', 'opptCon
f',
       'opptDiv', 'opptLoc', 'opptRslt', 'opptDayOff'],
      dtype='object')
```

In [57]:
```
#Selecting the features which are relevant at the moment
player = playerBoxScore[['teamAbbr','teamConf', 'teamDiv', 'teamLoc', 't
eamRslt','playFNm',
       'playStat', 'playMin', 'playPos', 'playHeight', 'playWeight',
       'playBDate', 'playPTS', 'playAST', 'playTO', 'playSTL', 'playBLK'
,
       'playPF', 'playFGA', 'playFGM', 'playFG%', 'play2PA', 'play2PM',
       'play2P%', 'play3PA', 'play3PM', 'play3P%', 'playFTA', 'playFTM',
       'playFT%', 'playORB', 'playDRB', 'playTRB', ]]
```

In [58]:
```
player.head()
```

Out[58]:

	teamAbbr	teamConf	teamDiv	teamLoc	teamRslt	playFNm	playStat	playMin	playPos	pl
0	WAS	East	Southeast	Away	Loss	A.J.	Starter	29	PG	
1	WAS	East	Southeast	Away	Loss	Trevor	Starter	25	SG	
2	WAS	East	Southeast	Away	Loss	Emeka	Starter	25	C	
3	WAS	East	Southeast	Away	Loss	Bradley	Starter	22	SG	
4	WAS	East	Southeast	Away	Loss	Trevor	Starter	17	PF	

5 rows × 33 columns

Since we want to find out what are the unique positions are in a basketball, we run the code below:

In [59]:
```
player["playPos"].unique()
```

Out[59]:
```
array(['PG', 'SG', 'C', 'PF', 'SF', 'F', 'G'], dtype=object)
```

There are five positions in basketball. These positions are employed by all basketball teams. They are defined as the point guard (PG), the shooting guard (SG), the small forward (SF), the power forward (PF), and the center (C).

In [60]:
```
#displays the title of the columnns
player.columns
```

Out[60]:
```
Index(['teamAbbr', 'teamConf', 'teamDiv', 'teamLoc', 'teamRslt', 'playF
Nm',
       'playStat', 'playMin', 'playPos', 'playHeight', 'playWeight',
       'playBDate', 'playPTS', 'playAST', 'playTO', 'playSTL', 'playBL
K',
       'playPF', 'playFGA', 'playFGM', 'playFG%', 'play2PA', 'play2PM',
       'play2P%', 'play3PA', 'play3PM', 'play3P%', 'playFTA', 'playFT
M',
       'playFT%', 'playORB', 'playDRB', 'playTRB'],
      dtype='object')
```

In [61]:
```
# Grouping the data by the player first name and then totalling the numb
er of fouls
df_fouls = player.groupby("playFNm").playPF.sum()

#Looking at the player names having fouls less than 5
df_fouls[df_fouls<5]
```

Out[61]:
```
playFNm
Athanasios    2
Branden       2
Carrick       4
Cliff         1
Coty          0
DaJuan        0
Dalenta       0
Darko         1
Duje          2
Edmond        1
Elias         0
Gursimran     0
Jacob         4
Jaleel        1
Jameel        2
Jerrelle      0
Kalin         1
Kristopher    3
Lazar         3
Lester        4
Monte         1
Mustafa       2
Nazareth      0
Othyus        3
Patricio      4
Quentin       1
Scott         0
Name: playPF, dtype: int64
```

From the above, we can see that these are the name of the players who have fouls counts less than 5

In [62]:
```
df_fouls[df_fouls>1000]
```

Out[62]:
```
playFNm
Aaron       1413
Al          1664
Amir        1414
Andre       2450
Andrew      2368
Anthony     3946
Austin      1017
Bismack     1044
Blake       1083
Brandon     2573
Brook       1012
Carmelo     1091
Chris       2622
Corey       1019
Dante       1097
David       1835
DeAndre     1518
DeMarcus    1477
Derrick     1910
Devin       1365
Draymond    1285
Dwight      1871
Earl        1385
Edward      1034
Enes        1042
Eric        1715
Ersan       1008
Evan        1886
Gerald      1562
Giannis     1162
            ...
Matt        1301
Maurice     1345
Michael     2661
Mike        1811
Nick        1498
Nikola      2650
Patrick     2554
Paul        3160
Quincy      1010
Reggie      1239
Ricky       1029
Robert      1344
Rodney      1255
Roy         1155
Rudy        1665
Russell     1067
Ryan        1395
Serge       1275
Spencer     1208
Steven      1058
Taj         1096
Terrence    1153
Thaddeus    1067
Tony        2232
Trevor      1689
```

```
Tristan     1092
Tyler       2341
Wardell     1008
Wesley      1868
Zach        1099
Name: playPF, Length: 88, dtype: int64
```

From the above, we can see that these are the list of the players who have fouls counts more than 1000

In [63]:
```
# Grouping the data by the first name of the player and then summing the
number of 3-point attempts made
df_3PA = player.groupby("playFNm").play3PA.sum()
df_3PA_arr=np.asarray(df_3PA)
#Looking at the players having number of three-points-attempt greater th
an 3000
df_3PA[df_3PA>3000]
```

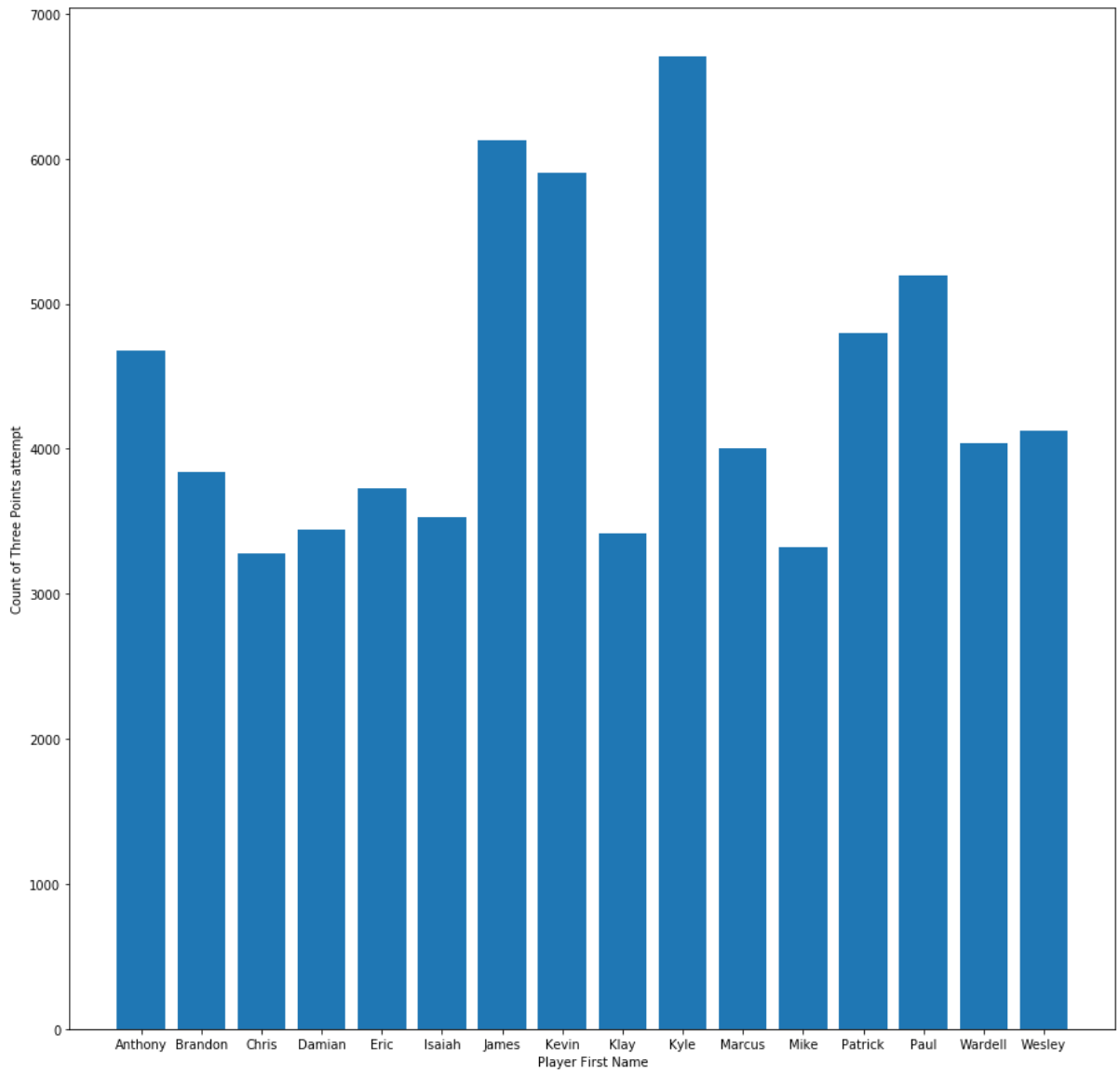
Out[63]:
```
playFNm
Anthony    4676
Brandon    3836
Chris      3278
Damian     3444
Eric       3729
Isaiah     3532
James      6127
Kevin      5905
Klay       3420
Kyle       6704
Marcus     4001
Mike       3321
Patrick    4801
Paul       5192
Wardell    4038
Wesley     4122
Name: play3PA, dtype: int64
```

From the above, we can see that these are the list of the players who have the highest number of 3-points-attempt

In [64]:
```
# Plotting the bar graph for the better visualiztion of the attempts mad
e by the players
ax = fig.add_axes([0,0,1,1])
labels_players = ['Anthony','Brandon','Chris','Damian','Eric','Isaiah',
'James','Kevin',
            'Klay','Kyle','Marcus','Mike','Marcus','Patrick','Paul','War
dell',
            'Wesley']
count_3PA = []
for p in labels_players:
    count=df_3PA[p]
    count_3PA.append(count)

import matplotlib.pyplot as plt
plt.figure(figsize=(15,15))
plt.bar(labels_players,count_3PA)
plt.xlabel("Player First Name")
plt.ylabel("Count of Three Points attempt")
```

```
//anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:2: Matplo
tlibDeprecationWarning:

Adding an axes using the same arguments as a previous axes currently re
uses the earlier instance.  In a future version, a new instance will al
ways be created and returned.  Meanwhile, this warning can be suppresse
d, and the future behavior ensured, by passing a unique label to each a
xes instance.
```

Out[64]: Text(0, 0.5, 'Count of Three Points attempt')



From the above bar graph it can be said that the player whose first name is Kyle has the highest number of 3-points-attempts among all the players

In [65]:
```
# Grouping the data by the first name of the player and then summing the
number of 3-point shots
#made
df_3PM=player.groupby("playFNm").play3PM.sum()
df_3PM_arr=np.asarray(df_3PA)
df_3PM
```

Out[65]:
```
playFNm
A.J.            91
Aaron          657
Abdel           23
Adonis           1
Adreian         15
Al             338
Al-Farouq      372
Alan           278
Alec           231
Alejandro      178
Aleksander      18
Alex            23
Alexander        0
Alexey         162
Alexis           0
Alfonso        392
Alfonzo          3
Allen          483
Alonzo         118
Amar'e           0
Amir            91
Anderson         0
Andray          18
Andre          426
Andrea          86
Andrei          36
Andrew         509
Andris           0
André          108
Antawn          92
               ...
Ty             315
Tyler          503
Tyreke         324
Tyrone          10
Tyrus            3
Tyshawn          9
Tyson            0
Tyus            87
Udonis           4
Vander           2
Viacheslav       0
Victor          34
Vincent        605
Vitor           12
Vladimir         5
Wade             7
Walter           1
Wardell       1757
Wayne          737
Wesley        1501
Will             2
William        486
Willie         159
Wilson         511
Xavier          42
```

```
Zach           124
Zachary        342
Zaza             0
Zhou             2
Zoran            3
Name: play3PM, Length: 592, dtype: int64
```

In [66]:
```
#Max number of three-points-shot made by any player
df_3PM.max()
```

Out[66]:
```
2683
```

In [67]:
```
#player with the highest number of 3-points-shot
df_3PM[df_3PM==2683]
```

Out[67]:
```
playFNm
Kyle    2683
Name: play3PM, dtype: int64
```

Player whose first name is Kyle is the player to make max number of 3-points-shot

In [68]:
```
df_3PM.min()
```

Out[68]:
```
0
```

In [69]:
```
#player with 0 3-points shot
df_3PM[df_3PM==0]
```

Out[69]:
```
playFNm
Alexander     0
Alexis        0
Amar'e        0
Anderson      0
Andris        0
Ante          0
Arinze        0
Arnett        0
Athanasios    0
Bernard       0
Bismack       0
Boban         0
Brandan       0
Branden       0
Brendan       0
Cheick        0
Chinanu       0
Cliff         0
Clint         0
Cole          0
Cristiano     0
DaJuan        0
Dakari        0
Dalenta       0
Darko         0
DeJuan        0
DeSagana      0
Dexter        0
Deyonta       0
Diamond       0
              ..
Kwame         0
Lazar         0
London        0
Mangok        0
Markelle      0
Marshall      0
Mason         0
Melvin        0
Miles         0
Miroslav      0
Mitchell      0
Monte         0
Mustafa       0
Nazr          0
Ognjen        0
Omer          0
Othyus        0
Patricio      0
Rakeem        0
Ronny         0
Samardo       0
Scott         0
Shavlik       0
Steven        0
Terrel        0
```

```
Tibor         0
Tristan       0
Tyson         0
Viacheslav    0
Zaza          0
Name: play3PM, Length: 103, dtype: int64
```

As it is depicted above, there are 103 players who have 0 , three-points-shot

In [70]:
```
#tallest player
maxi = player["playHeight"].max()
player['playFNm'].loc[player["playHeight"]==89]
```

Out[70]:
```
75868    Gursimran
76114    Gursimran
76393    Gursimran
Name: playFNm, dtype: object
```

The tallest player is Gursimran

In [71]:
```
#shortest player
mini = player["playHeight"].min()
player['playFNm'].loc[player["playHeight"]==mini]
```

Out[71]:
```
122       Isaiah
136         Nate
339         Nate
424       Isaiah
554       Isaiah
628         Nate
1014      Isaiah
1057        Nate
1335      Isaiah
1401        Nate
1645      Isaiah
1806        Nate
1990      Isaiah
2080        Nate
2449        Nate
2717      Isaiah
2797        Nate
3023      Isaiah
3080        Nate
3506        Nate
3598      Isaiah
3981        Nate
4014      Isaiah
4278        Nate
4423      Isaiah
4618        Nate
4937      Isaiah
5018        Nate
5128      Isaiah
5423        Nate
            ...
143346    Isaiah
143746    Isaiah
143952    Isaiah
144443    Isaiah
144825    Isaiah
145229    Isaiah
145489    Isaiah
145709    Isaiah
146179    Isaiah
146422    Kahlil
146555    Isaiah
146785    Isaiah
147241    Isaiah
147921    Isaiah
148004    Isaiah
148376    Isaiah
148538    Isaiah
148669    Isaiah
149194    Isaiah
149561    Isaiah
149905    Isaiah
150223    Isaiah
150473    Isaiah
150851    Isaiah
151178    Isaiah
151243    Isaiah
```

```
151566    Isaiah
151907    Isaiah
152436    Isaiah
155485    Kahlil
Name: playFNm, Length: 637, dtype: object
```

As it is depicted from above, Isaiah, Nate, and Kahlil are the shortest among the players

In [72]:
```
#heaviest player
maxi=player["playWeight"].max()
player['playFNm'].loc[player["playWeight"]==maxi]
```

Out[72]:
```
75868    Gursimran
76114    Gursimran
76393    Gursimran
Name: playFNm, dtype: object
```

In [73]:
```
#lightest player
mini = player["playWeight"].min()
player['playFNm'].loc[player["playWeight"]==mini]
```

Out[73]:

103717	Tyler
103923	Tyler
104291	Tyler
104430	Tyler
106479	Tyler
107253	Tyler
107558	Tyler
107851	Tyler
108117	Tyler
108592	Tyler
108809	Tyler
109915	Tyler
110555	Tyler
111181	Tyler
111784	Tyler
112032	Tyler
112795	Tyler
113466	Tyler
113811	Tyler
113961	Tyler
114337	Tyler
114601	Tyler
114755	Tyler
116073	Tyler
116355	Tyler
116728	Tyler
117059	Tyler
117467	Tyler
117620	Tyler
117968	Tyler
	...
142943	Tyler
143153	Tyler
143480	Tyler
143880	Tyler
144301	Tyler
144551	Tyler
145002	Tyler
145187	Tyler
145348	Tyler
145785	Tyler
146060	Tyler
146312	Tyler
146719	Tyler
146829	Tyler
147943	Tyler
150557	Tyler
151159	Tyler
151413	Tyler
151772	Tyler
152168	Tyler
152533	Tyler
152742	Tyler
153101	Tyler
153415	Tyler
153613	Tyler
154052	Tyler

```
154352    Tyler
154826    Tyler
155117    Tyler
155396    Tyler
Name: playFNm, Length: 132, dtype: object
```

In [74]:
```
College
```

Out[74]:

	Unnamed: 0	active_from	active_to	birth_date	college	height	name	position
0	0	1991	1995	24-Jun-68	Duke University	10-Jun	Alaa Abdelnaby	F-C
1	1	1969	1978	7-Apr-46	Iowa State University	9-Jun	Zaid Abdul-Aziz	C-F
2	2	1970	1989	16-Apr-47	University of California, Los Angeles	2-Jul	Kareem Abdul-Jabbar	C
3	3	1991	2001	9-Mar-69	Louisiana State University	1-Jun	Mahmoud Abdul-Rauf	G
4	4	1998	2003	3-Nov-74	University of Michigan, San Jose State University	6-Jun	Tariq Abdul-Wahad	F
5	5	1997	2008	11-Dec-76	University of California	9-Jun	Shareef Abdur-Rahim	F
6	6	1977	1981	6-May-54	Indiana University	7-Jun	Tom Abernethy	F
7	7	1957	1957	27-Jul-32	Western Kentucky University	3-Jun	Forest Able	G
8	8	1947	1948	9-Feb-19	Salem International University	3-Jun	John Abramovic	F
9	9	2017	2018	1-Aug-93	NaN	6-Jun	Alex Abrines	G-F
10	10	2006	2009	21-Jan-83	Pepperdine University	5-Jun	Alex Acker	G
11	11	1954	1954	4-Sep-30	Long Island University	Jun-00	Don Ackerman	G
12	12	1988	1993	15-Nov-62	Oral Roberts University	11-Jun	Mark Acres	F-C
13	13	1968	1968	11-Jan-42	Hillsdale College	6-Jun	Bud Acton	F
14	14	2013	2018	6-Oct-90	Baylor University	7-Jun	Quincy Acy	F
15	15	1976	1988	19-Jul-54	University of Oklahoma	9-Jun	Alvan Adams	C-F
16	16	1971	1977	27-Nov-47	Northwestern University	6-Jun	Don Adams	F
17	17	1973	1975	15-May-49	Gardner-Webb University	5-Jun	George Adams	F-G
18	18	2007	2009	20-Jun-84	University of Arizona	4-Jun	Hassan Adams	G
19	19	2015	2016	8-Jul-94	University of California, Los Angeles	5-Jun	Jordan Adams	G

	Unnamed: 0	active_from	active_to	birth_date	college	height	name	position
20	20	1986	1996	19-Jan-63	Boston College	10-May	Michael Adams	G
21	21	2014	2018	20-Jul-93	University of Pittsburgh	Jul-00	Steven Adams	C
22	22	1987	1997	22-Jul-64	Syracuse University	7-Jun	Rafael Addison	F-G
23	23	2018	2018	18-Jul-97	University of Kentucky	10-Jun	Bam Adebayo	C-F
24	24	1969	1975	16-Jun-46	Loyola Marymount University	1-Jun	Rick Adelman	G
25	25	2011	2015	10-Feb-86	University of Connecticut	7-Jun	Jeff Adrien	F
26	26	2008	2018	15-Oct-85	University of California, Los Angeles	5-Jun	Arron Afflalo	G
27	27	2007	2011	9-Feb-84	Michigan State University	5-Jun	Maurice Ager	G
28	28	1982	1994	10-Dec-59	DePaul University	6-Jun	Mark Aguirre	F-G
29	29	2008	2012	27-May-84	Missouri State University	2-Jun	Blake Ahearn	G
...	...	...	...	...	...	...	...	...
4546	4546	2015	2018	16-Aug-95	University of Kentucky	6-Jun	James Young	G-F
4547	4547	2016	2018	27-Jun-92	University of Houston, University of Oregon	2-Jun	Joe Young	G
4548	4548	1999	1999	31-Dec-78	NaN	7-Jun	Korleone Young	F
4549	4549	1985	1990	2-Jan-61	University of Houston	7-Jun	Michael Young	F-G
4550	4550	2008	2018	1-Jun-85	University of Southern California	7-Jun	Nick Young	G-F
4551	4551	1987	1987	4-Aug-63	Virginia Polytechnic Institute and State Unive...	5-Jun	Perry Young	G
4552	4552	2010	2013	1-Jun-85	University of Pittsburgh	6-Jun	Sam Young	F
4553	4553	2008	2018	21-Jun-88	Georgia Institute of Technology	8-Jun	Thaddeus Young	F
4554	4554	2000	2000	6-Feb-76	Stanford University	Jul-00	Tim Young	C

	Unnamed: 0	active_from	active_to	birth_date	college	height	name	position
4555	4555	2009	2009	6-Nov-85	NaN	9-Jun	Sun Yue	G
4556	4556	1947	1956	7-Dec-25	St. John's University	2-Jun	Max Zaslofsky	G-F
4557	4557	1953	1955	13-Oct-30	St. John's University	7-Jun	Zeke Zawoluk	F-C
4558	4558	2014	2018	5-Oct-92	Indiana University	Jul-00	Cody Zeller	C-F
4559	4559	1962	1962	8-Jun-39	Miami University	1-Jun	Dave Zeller	G
4560	4560	1971	1972	20-Nov-47	Drake University	3-Jun	Gary Zeller	G
4561	4561	1947	1947	10-Jul-19	Washington & Jefferson College	4-Jun	Harry Zeller	C-F
4562	4562	2013	2013	7-Apr-87	University of Notre Dame	11-Jun	Luke Zeller	C
4563	4563	2013	2018	17-Jan-90	University of North Carolina	Jul-00	Tyler Zeller	F-C
4564	4564	1980	1980	1-Oct-57	Arizona State University	8-Jun	Tony Zeno	F
4565	4565	1988	1988	13-Apr-64	Seattle Pacific University, University of Wash...	10-Jun	Phil Zevenbergen	C
4566	4566	2001	2005	8-Jul-77	NaN	Jul-00	Wang Zhizhi	C
4567	4567	1996	1998	2-Aug-73	University of California, Los Angeles	Jul-00	George Zidek	C
4568	4568	2006	2006	2-Dec-81	Mississippi State University	3-Jun	Derrick Zimmerman	G
4569	4569	2017	2017	9-Sep-96	University of Nevada, Las Vegas	Jul-00	Stephen Zimmerman	C
4570	4570	2017	2018	18-Feb-94	NaN	8-Jun	Paul Zipser	G-F
4571	4571	2018	2018	4-Jan-97	NaN	11-Jun	Ante Zizic	F-C
4572	4572	1983	1983	20-Dec-53	Kent State University	1-Jul	Jim Zoet	C
4573	4573	1971	1971	7-Jun-48	Duquesne University	1-Jun	Bill Zopf	G
4574	4574	2017	2018	18-Mar-97	NaN	1-Jul	Ivica Zubac	C
4575	4575	1949	1949	19-Dec-19	George Washington University	3-Jun	Matt Zunic	G-F

4576 rows × 34 columns

From the above, we kind of get the max and min attributes of some of the players

The Data

After exploring the data, we will be using the player dataset for our notebook. Our goal is to predict the player position based on measurements of their height and weight.

In [75]:
```
#These are the original data, displaying the fist name along with the pl
ayer position
player[["playFNm", "playPos"]]
```

Out[75]:

	playFNm	playPos
0	A.J.	PG
1	Trevor	SG
2	Emeka	C
3	Bradley	SG
4	Trevor	PF
5	Jordan	SG
6	Martell	SG
7	Jan	SF
8	Chris	SF
9	Earl	PF
10	Jannero	PG
11	Anderson	C
12	Kyrie	PG
13	Alonzo	SF
14	Tristan	SF
15	Dion	SG
16	Calvin	SF
17	Daniel	PG
18	Tyler	C
19	Donald	PG
20	Luke	SF
21	Rajon	PG
22	Paul	SF
23	Kevin	PF
24	Brandon	PF
25	Courtney	SG
26	Jason	SG
27	Jeff	PF
28	Leandro	SG
29	Jared	PF
...	...	...
155683	Ekpe	C
155684	Damian	PG
155685	Christian	SG

	playFNm	playPos
155686	Al-Farouq	PF
155687	Jusuf	C
155688	Evan	SF
155689	Zach	C
155690	Edward	PF
155691	Patrick	SG
155692	Shabazz	PG
155693	Wade	PG
155694	Caleb	PF
155695	Jake	SF
155696	Tarik	PF
155697	Ronald	SG
155698	Joe	SG
155699	Anthony	SF
155700	Gerald	SG
155701	Aaron	F
155702	Zhou	PF
155703	Chinanu	C
155704	Bogdan	SG
155705	Justin	SF
155706	Willie	C
155707	De'Aaron	PG
155708	JaKarr	SF
155709	Chavano	SG
155710	Nigel	SF
155711	Vincent	SG
155712	Jack	PF

155713 rows × 2 columns

In [76]:
```
#Now we will be using label encoder to change the player position to the
number
encoder=LabelEncoder()
player["playPos"]=encoder.fit_transform(player["playPos"])
player["playPos"]
```

Out[76]:

```
0          4
1          6
2          0
3          6
4          3
5          6
6          6
7          5
8          5
9          3
10         4
11         0
12         4
13         5
14         5
15         6
16         5
17         4
18         0
19         4
20         5
21         4
22         5
23         3
24         3
25         6
26         6
27         3
28         6
29         3
          ..
155683     0
155684     4
155685     6
155686     3
155687     0
155688     5
155689     0
155690     3
155691     6
155692     4
155693     4
155694     3
155695     5
155696     3
155697     6
155698     6
155699     5
155700     6
155701     1
155702     3
155703     0
155704     6
155705     5
155706     0
155707     4
155708     5
```

```
155709    6
155710    5
155711    6
155712    3
Name: playPos, Length: 155713, dtype: int64
```

Since the label encoder gives a unique number to each of the player position, we can deduce from the above that these are the numbers that define the positions: 0=C, 1=F, 2=G ,3=PF, 4=PG, 6=SG, 5=SF

Next, we want to check and see how the players' weight and height are correlated with the players position, we build a correlation matrix

In [77]:
```
player.corr()
```

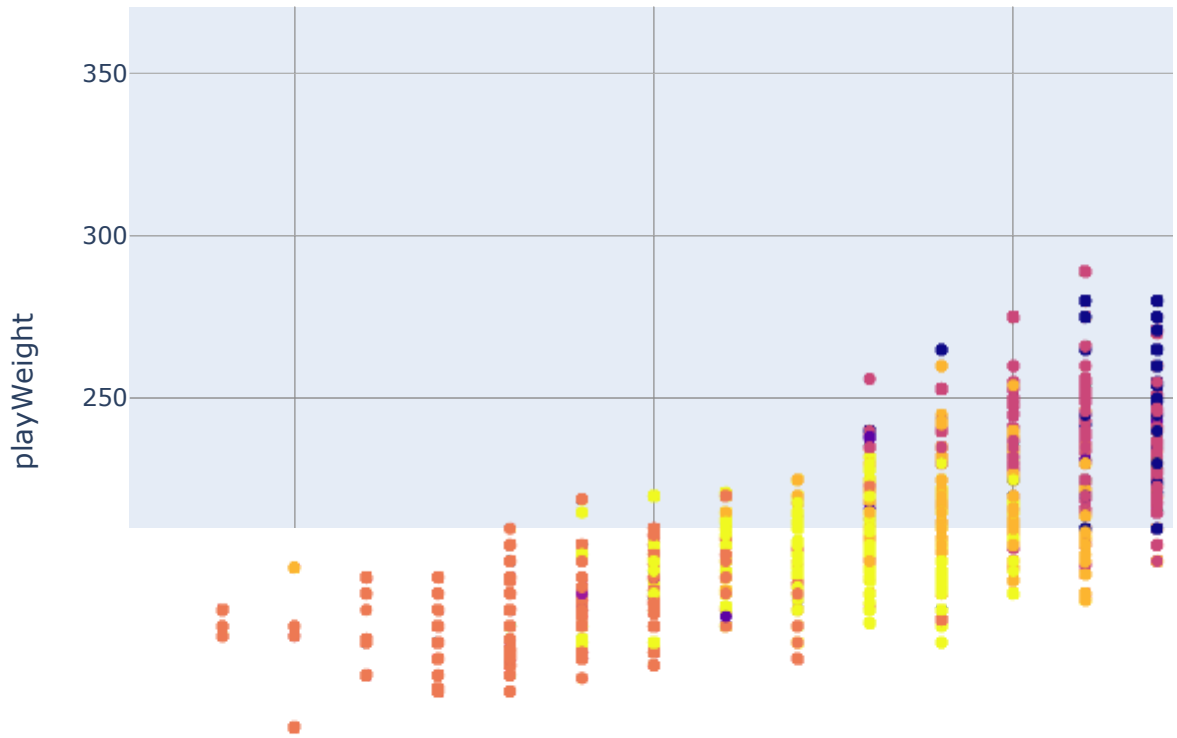
Out[77]:

	playMin	playPos	playHeight	playWeight	playPTS	playAST	playTO	playS
playMin	1.000000	0.068984	-0.070542	-0.069286	0.730469	0.510342	0.457095	0.3792
playPos	0.068984	1.000000	-0.568039	-0.602517	0.051176	0.126777	0.002840	0.0913
playHeight	-0.070542	-0.568039	1.000000	0.801663	-0.062331	-0.366652	-0.108370	-0.1269
playWeight	-0.069286	-0.602517	0.801663	1.000000	-0.048047	-0.319195	-0.084105	-0.1206
playPTS	0.730469	0.051176	-0.062331	-0.048047	1.000000	0.421939	0.408070	0.3039
playAST	0.510342	0.126777	-0.366652	-0.319195	0.421939	1.000000	0.402056	0.2960
playTO	0.457095	0.002840	-0.108370	-0.084105	0.408070	0.402056	1.000000	0.2328
playSTL	0.379251	0.091371	-0.126910	-0.120608	0.303973	0.296016	0.232820	1.0000
playBLK	0.228371	-0.270558	0.294576	0.248021	0.164073	0.001522	0.097196	0.0590
playPF	0.390977	-0.134638	0.113959	0.122741	0.266305	0.141878	0.251215	0.1481
playFGA	0.787700	0.092512	-0.107087	-0.086542	0.876450	0.464264	0.425958	0.3228
playFGM	0.708921	0.005800	-0.021846	-0.008912	0.959514	0.389152	0.382984	0.2881
playFG%	0.266308	-0.125916	0.106705	0.098274	0.449040	0.082087	0.118033	0.0980
play2PA	0.688677	-0.102769	0.047554	0.073291	0.772304	0.381799	0.391453	0.2742
play2PM	0.622564	-0.131652	0.088336	0.103891	0.822757	0.325093	0.351951	0.2509
play2P%	0.274893	-0.090636	0.076812	0.068661	0.395848	0.101785	0.124653	0.1129
play3PA	0.477807	0.385646	-0.318279	-0.319766	0.520816	0.324864	0.224210	0.2101
play3PM	0.389071	0.292505	-0.239479	-0.241257	0.568648	0.248812	0.177141	0.1625
play3P%	0.243900	0.278103	-0.226963	-0.235674	0.374447	0.175363	0.106058	0.1089
playFTA	0.484163	-0.041364	0.013183	0.032041	0.641344	0.299398	0.325404	0.2194
playFTM	0.473001	0.003505	-0.029261	-0.012432	0.656778	0.313661	0.320683	0.2146
playFT%	0.423301	0.005935	-0.031905	-0.026475	0.507450	0.245724	0.240177	0.1774
playORB	0.288008	-0.368682	0.344896	0.350872	0.231334	-0.009019	0.127095	0.0742
playDRB	0.544876	-0.258675	0.286574	0.284087	0.427482	0.204056	0.285279	0.1730
playTRB	0.535008	-0.346347	0.358398	0.358859	0.421881	0.153823	0.270677	0.1630

25 rows × 25 columns

From the above table we can conclude that there is a high negative correlation between playPos and playHeight & playWeight, with -0.568039 and -0.602517 respectively

In [78]:
```
px.scatter(player, x="playHeight", y="playWeight", color="playPos")
```



We can see using just player height and player weight as our features, we see some kind of cluster separation. That is, PG, SG, C, PF, SF, F, G appear in different regions of the plot even though there are some overlapped.

In [79]:
```
def plot_decision_boundaries(model, X, n=50):
    categories, z_int = np.unique(Y, return_inverse=True)
    # Make contour plot
    u = np.linspace(X[:,0].min()-0.5, X[:,0].max()+0.5, n)
    v = np.linspace(X[:,1].min()-0.5, X[:,1].max()+0.5, n)
    us,vs = np.meshgrid(u, v)
    X_test = np.c_[us.ravel(), vs.ravel()]
    z_str = model.predict(X_test)
    categories, z_int = np.unique(z_str, return_inverse=True)
    return go.Contour(x=X_test[:,0], y=X_test[:,1], z=z_int,
#                      contours=dict(start=0,end=2,size=1),
                      colorscale=px.colors.qualitative.Plotly[:3],
                      showscale=False,
                      )
```

In the following plot we can see the data points and the predicted class assignment for all combinations of player height and player weight.

One VS Rest Logistic Regression

In [80]:
```
nba_train, nba_test = train_test_split(player, test_size=0.25, random_st
ate=100)
nba_train = nba_train.sort_values(by='playPos')
nba_test = nba_test.sort_values(by='playPos')
```

In [81]:
```
a = LogisticRegression(multi_class='ovr')
logistic_regression_model = a.fit(player[['playHeight','playWeight']], p
layer['playPos'])
```

```
//anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.p
y:432: FutureWarning:

Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to
silence this warning.
```

In [82]:
```
nba_train['Predicted Position'] = logistic_regression_model.predict(nba_
train[['playHeight','playWeight']])
nba_train[['playHeight','playWeight','playPos', 'Predicted Position']].h
ead()
```

Out[82]:

	playHeight	playWeight	playPos	Predicted Position
25257	84	245	0	0
110738	82	245	0	3
40223	83	235	0	0
155655	84	260	0	0
114899	82	255	0	0

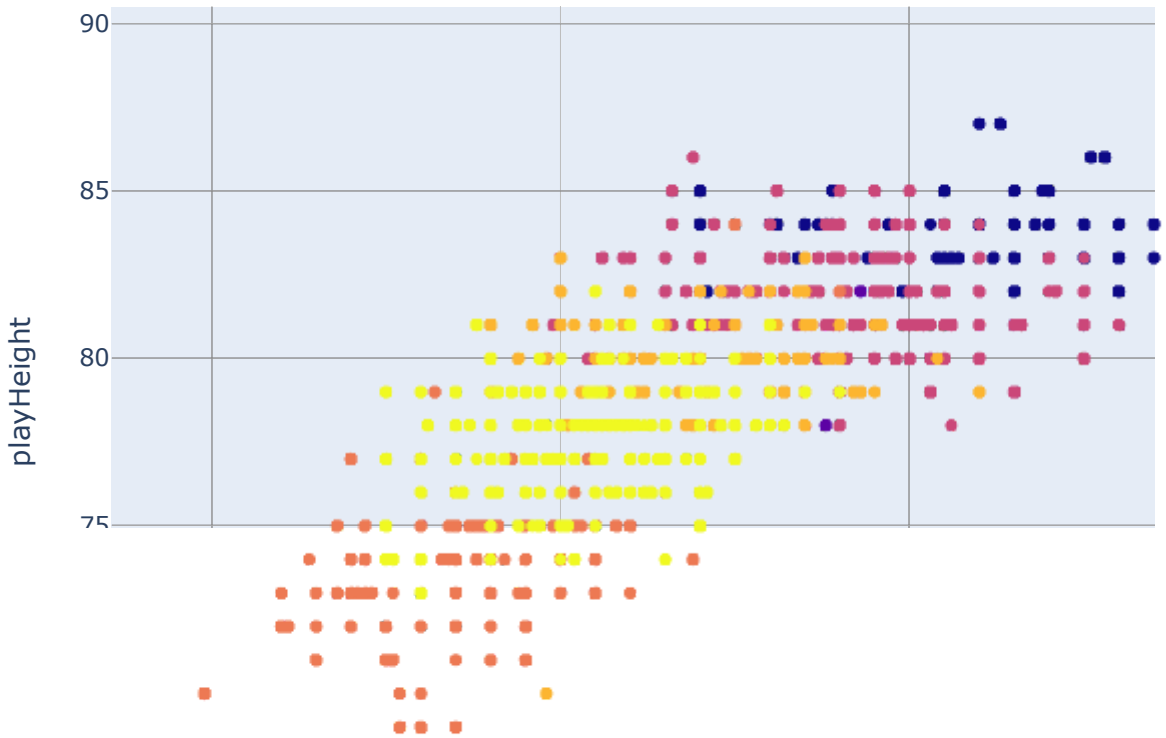
In [83]:
```
lr_training_accuracy = logistic_regression_model.score(nba_train[['playH
eight','playWeight']], nba_train['playPos'])
lr_training_accuracy
```

Out[83]: 0.6399421153582683

In [84]:
```
lr_test_accuracy = logistic_regression_model.score(nba_test[['playHeigh
t','playWeight']], nba_test['playPos'])
lr_test_accuracy
```

Out[84]: 0.6379305915898174

In [144]:
```
px.scatter(nba_train, x="playWeight", y="playHeight", color="playPos")
```



From here we can see that the training set and test set would be 0.6399 and 0.6379 respectively

Decision Trees

Notice that there are seven classes of player position. This is a not a binary classification problem but instead a multiclass classification problem. We can see that

In [134]:
```
x = player[["playHeight","playWeight"]]
Y = player[["playPos"]]
```

In [135]:
```
x_train,x_test,Y_train,Y_test=train_test_split(x,Y,test_size=0.2)
```

In [136]:
```
scaler=StandardScaler()
scaler_train=scaler.fit_transform(x_train)
```

In [86]:
```
scaler=StandardScaler()
scaler_train=scaler.fit_transform(x_train)
```

In [137]:
```
#We will be using Decision trees algorithm for the prediction because th
is we can't do the
# binary classification
from sklearn import tree
```

In [138]:
```
#Training the model and then predicting
decision_tree=tree.DecisionTreeClassifier(random_state=42)
decision_tree=decision_tree.fit(x_train,Y_train)
Y_pred=decision_tree.predict(x_test)
```

In [90]:
```
lr_training_accuracy = decision_tree.score(nba_train[['playHeight','play
Weight']], nba_train['playPos'])
print("The training accuracy of our decision tree model is",lr_training_
accuracy*100,"%")
```

```
The training accuracy of our decision tree model is 79.64447184545828 %
```

In [91]:
```
lr_test_accuracy = decision_tree.score(nba_test[['playHeight','playWeigh
t']], nba_test['playPos'])
print("The test accuracy of our decision tree model is",lr_test_accuracy
*100,"%")
```

```
The test accuracy of our decision tree model is 79.23399008451283 %
```

Our Decision Tree model has performed nice keeping in mind that the correlation values were -0.56 and -0.60

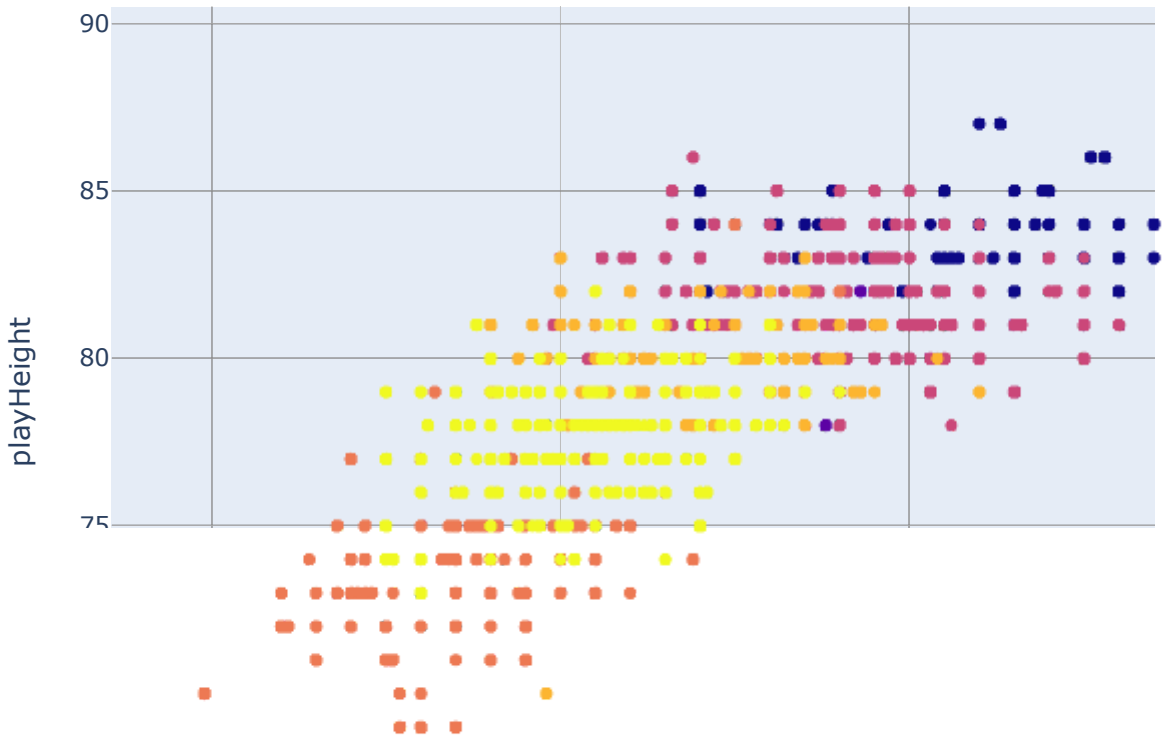
Random Forests

In [104]:
```
from sklearn import ensemble
```

In [105]:
```
train_forest,test_forest=train_test_split(player, test_size=0.25, random
_state=42)

train_forest = train_forest.sort_values(by='playPos')
test_forest = test_forest.sort_values(by='playPos')
```

In [106]:
```
random_forest_model = ensemble.RandomForestClassifier(n_estimators=20,ra
ndom_state=42)
random_forest_model= b.fit(player[['playHeight','playWeight']], player[
'playPos'])
```

In [110]:
```
px.scatter(nba_train, x="playWeight", y="playHeight", color="playPos")
```



In [108]:
```
lr_training_accuracy_forest = random_forest_model.score(train_forest[['p
layHeight','playWeight']], train_forest['playPos'])
print("The training accuracy of our random_forest_model is",lr_training_
accuracy_forest*100,"%")
```

```
The training accuracy of our random_forest_model is 79.4637964104672 %
```

In [109]:
```
lr_test_accuracy_forest = random_forest_model.score(test_forest[['playHe
ight','playWeight']], test_forest['playPos'])
print("The test accuracy of our random forest model is",lr_test_accuracy
_forest*100,"%")
```

```
The test accuracy of our random forest model is 79.79912147756171 %
```

In [111]:
```
College
```

Out[111]:

	Unnamed: 0	active_from	active_to	birth_date	college	height	name	position
0	0	1991	1995	24-Jun-68	Duke University	10-Jun	Alaa Abdelnaby	F-C
1	1	1969	1978	7-Apr-46	Iowa State University	9-Jun	Zaid Abdul-Aziz	C-F
2	2	1970	1989	16-Apr-47	University of California, Los Angeles	2-Jul	Kareem Abdul-Jabbar	C
3	3	1991	2001	9-Mar-69	Louisiana State University	1-Jun	Mahmoud Abdul-Rauf	G
4	4	1998	2003	3-Nov-74	University of Michigan, San Jose State University	6-Jun	Tariq Abdul-Wahad	F
5	5	1997	2008	11-Dec-76	University of California	9-Jun	Shareef Abdur-Rahim	F
6	6	1977	1981	6-May-54	Indiana University	7-Jun	Tom Abernethy	F
7	7	1957	1957	27-Jul-32	Western Kentucky University	3-Jun	Forest Able	G
8	8	1947	1948	9-Feb-19	Salem International University	3-Jun	John Abramovic	F
9	9	2017	2018	1-Aug-93	NaN	6-Jun	Alex Abrines	G-F
10	10	2006	2009	21-Jan-83	Pepperdine University	5-Jun	Alex Acker	G
11	11	1954	1954	4-Sep-30	Long Island University	Jun-00	Don Ackerman	G
12	12	1988	1993	15-Nov-62	Oral Roberts University	11-Jun	Mark Acres	F-C
13	13	1968	1968	11-Jan-42	Hillsdale College	6-Jun	Bud Acton	F
14	14	2013	2018	6-Oct-90	Baylor University	7-Jun	Quincy Acy	F
15	15	1976	1988	19-Jul-54	University of Oklahoma	9-Jun	Alvan Adams	C-F
16	16	1971	1977	27-Nov-47	Northwestern University	6-Jun	Don Adams	F
17	17	1973	1975	15-May-49	Gardner-Webb University	5-Jun	George Adams	F-G
18	18	2007	2009	20-Jun-84	University of Arizona	4-Jun	Hassan Adams	G
19	19	2015	2016	8-Jul-94	University of California, Los Angeles	5-Jun	Jordan Adams	G

	Unnamed: 0	active_from	active_to	birth_date	college	height	name	position
20	20	1986	1996	19-Jan-63	Boston College	10-May	Michael Adams	G
21	21	2014	2018	20-Jul-93	University of Pittsburgh	Jul-00	Steven Adams	C
22	22	1987	1997	22-Jul-64	Syracuse University	7-Jun	Rafael Addison	F-G
23	23	2018	2018	18-Jul-97	University of Kentucky	10-Jun	Bam Adebayo	C-F
24	24	1969	1975	16-Jun-46	Loyola Marymount University	1-Jun	Rick Adelman	G
25	25	2011	2015	10-Feb-86	University of Connecticut	7-Jun	Jeff Adrien	F
26	26	2008	2018	15-Oct-85	University of California, Los Angeles	5-Jun	Arron Afflalo	G
27	27	2007	2011	9-Feb-84	Michigan State University	5-Jun	Maurice Ager	G
28	28	1982	1994	10-Dec-59	DePaul University	6-Jun	Mark Aguirre	F-G
29	29	2008	2012	27-May-84	Missouri State University	2-Jun	Blake Ahearn	G
...	...	...	...	...	...	...	...	...
4546	4546	2015	2018	16-Aug-95	University of Kentucky	6-Jun	James Young	G-F
4547	4547	2016	2018	27-Jun-92	University of Houston, University of Oregon	2-Jun	Joe Young	G
4548	4548	1999	1999	31-Dec-78	NaN	7-Jun	Korleone Young	F
4549	4549	1985	1990	2-Jan-61	University of Houston	7-Jun	Michael Young	F-G
4550	4550	2008	2018	1-Jun-85	University of Southern California	7-Jun	Nick Young	G-F
4551	4551	1987	1987	4-Aug-63	Virginia Polytechnic Institute and State Unive...	5-Jun	Perry Young	G
4552	4552	2010	2013	1-Jun-85	University of Pittsburgh	6-Jun	Sam Young	F
4553	4553	2008	2018	21-Jun-88	Georgia Institute of Technology	8-Jun	Thaddeus Young	F
4554	4554	2000	2000	6-Feb-76	Stanford University	Jul-00	Tim Young	C

	Unnamed: 0	active_from	active_to	birth_date	college	height	name	position
4555	4555	2009	2009	6-Nov-85	NaN	9-Jun	Sun Yue	G
4556	4556	1947	1956	7-Dec-25	St. John's University	2-Jun	Max Zaslofsky	G-F
4557	4557	1953	1955	13-Oct-30	St. John's University	7-Jun	Zeke Zawoluk	F-C
4558	4558	2014	2018	5-Oct-92	Indiana University	Jul-00	Cody Zeller	C-F
4559	4559	1962	1962	8-Jun-39	Miami University	1-Jun	Dave Zeller	G
4560	4560	1971	1972	20-Nov-47	Drake University	3-Jun	Gary Zeller	G
4561	4561	1947	1947	10-Jul-19	Washington & Jefferson College	4-Jun	Harry Zeller	C-F
4562	4562	2013	2013	7-Apr-87	University of Notre Dame	11-Jun	Luke Zeller	C
4563	4563	2013	2018	17-Jan-90	University of North Carolina	Jul-00	Tyler Zeller	F-C
4564	4564	1980	1980	1-Oct-57	Arizona State University	8-Jun	Tony Zeno	F
4565	4565	1988	1988	13-Apr-64	Seattle Pacific University, University of Wash...	10-Jun	Phil Zevenbergen	C
4566	4566	2001	2005	8-Jul-77	NaN	Jul-00	Wang Zhizhi	C
4567	4567	1996	1998	2-Aug-73	University of California, Los Angeles	Jul-00	George Zidek	C
4568	4568	2006	2006	2-Dec-81	Mississippi State University	3-Jun	Derrick Zimmerman	G
4569	4569	2017	2017	9-Sep-96	University of Nevada, Las Vegas	Jul-00	Stephen Zimmerman	C
4570	4570	2017	2018	18-Feb-94	NaN	8-Jun	Paul Zipser	G-F
4571	4571	2018	2018	4-Jan-97	NaN	11-Jun	Ante Zizic	F-C
4572	4572	1983	1983	20-Dec-53	Kent State University	1-Jul	Jim Zoet	C
4573	4573	1971	1971	7-Jun-48	Duquesne University	1-Jun	Bill Zopf	G
4574	4574	2017	2018	18-Mar-97	NaN	1-Jul	Ivica Zubac	C
4575	4575	1949	1949	19-Dec-19	George Washington University	3-Jun	Matt Zunic	G-F

4576 rows × 34 columns

In [112]:
```
College.shape
```

Out[112]:
```
(4576, 34)
```

In [113]:
```
College.columns
```

Out[113]:
```
Index(['Unnamed: 0', 'active_from', 'active_to', 'birth_date', 'colleg
e',
       'height', 'name', 'position', 'url', 'weight', 'NBA__3ptapg',
       'NBA__3ptpct', 'NBA__3ptpg', 'NBA_efgpct', 'NBA_fg%', 'NBA_fg_pe
r_game',
       'NBA_fga_per_game', 'NBA_ft%', 'NBA_ft_per_g', 'NBA_fta_p_g',
       'NBA_g_played', 'NBA_ppg', 'NCAA__3ptapg', 'NCAA__3ptpct',
       'NCAA__3ptpg', 'NCAA_efgpct', 'NCAA_fgapg', 'NCAA_fgpct', 'NCAA_
fgpg',
       'NCAA_ft', 'NCAA_ftapg', 'NCAA_ftpg', 'NCAA_games', 'NCAA_ppg'],
      dtype='object')
```

In [114]:
```
# We have many null values but we won't replaced th NaN with mean or mod
e because it would lead
#to incorrect predictions
College.isnull().sum()
```

Out[114]:
```
Unnamed: 0             0
active_from            0
active_to              0
birth_date            29
college              302
height                 1
name                   0
position               1
url                    0
weight                 6
NBA__3ptapg         1128
NBA__3ptpct         1623
NBA__3ptpg          1128
NBA_efgpct          1150
NBA_fg%               28
NBA_fg_per_game        0
NBA_fga_per_game       0
NBA_ft%              198
NBA_ft_per_g           0
NBA_fta_p_g            0
NBA_g_played           0
NBA_ppg                0
NCAA__3ptapg        2708
NCAA__3ptpct        2850
NCAA__3ptpg         2705
NCAA_efgpct         4576
NCAA_fgapg          1994
NCAA_fgpct          1993
NCAA_fgpg           1978
NCAA_ft             1982
NCAA_ftapg          1984
NCAA_ftpg           1978
NCAA_games          1978
NCAA_ppg            1978
dtype: int64
```

We are grouping our data by url because it is the only feature which is unique to a player. In this dataset there are many players with single name and same data of birth.

In [115]:
```
df = College.groupby("url").NBA__3ptapg.sum()
df
```

Out[115]:
```
url
/players/a/abdelal01.html    0.0
/players/a/abdulka01.html    0.0
/players/a/abdulma01.html    0.0
/players/a/abdulma02.html    2.3
/players/a/abdulta01.html    0.3
/players/a/abdulza01.html    0.0
/players/a/abdursh01.html    0.6
/players/a/abernto01.html    0.0
/players/a/ablefo01.html     0.0
/players/a/abramjo01.html    0.0
/players/a/abrinal01.html    3.3
/players/a/ackeral01.html    0.8
/players/a/ackerdo01.html    0.0
/players/a/acresma01.html    0.0
/players/a/actonbu01.html    0.0
/players/a/acyqu01.html      1.5
/players/a/adamsal01.html    0.0
/players/a/adamsdo01.html    0.1
/players/a/adamsge01.html    0.1
/players/a/adamsha01.html    0.0
/players/a/adamsjo01.html    0.8
/players/a/adamsmi01.html    4.4
/players/a/adamsst01.html    0.0
/players/a/addisra01.html    0.6
/players/a/adebaba01.html    0.1
/players/a/adelmri01.html    0.0
/players/a/adrieje01.html    0.0
/players/a/afflaar01.html    2.9
/players/a/agerma01.html     0.5
/players/a/aguirma01.html    1.4
                            ...
/players/y/youngja01.html    1.4
/players/y/youngjo01.html    1.0
/players/y/youngko01.html    1.3
/players/y/youngmi01.html    0.6
/players/y/youngni01.html    3.9
/players/y/youngpe01.html    0.0
/players/y/youngsa01.html    0.7
/players/y/youngth01.html    1.3
/players/y/youngti01.html    0.0
/players/y/yuesu01.html      0.6
/players/z/zasloma01.html    0.0
/players/z/zawolze01.html    0.0
/players/z/zelleco01.html    0.1
/players/z/zelleda01.html    0.0
/players/z/zellega01.html    0.1
/players/z/zelleha01.html    0.0
/players/z/zellelu01.html    0.3
/players/z/zellety01.html    0.1
/players/z/zenoto01.html     0.0
/players/z/zevenph01.html    0.0
/players/z/zhizhwa01.html    1.3
/players/z/zidekge01.html    0.0
/players/z/zimmede01.html    0.0
/players/z/zimmest01.html    0.0
/players/z/zipsepa01.html    2.1
```

```
/players/z/zizican01.html    0.0
/players/z/zoetji01.html     0.0
/players/z/zopfbi01.html     0.0
/players/z/zubaciv01.html    0.0
/players/z/zunicma01.html    0.0
Name: NBA__3ptapg, Length: 4576, dtype: float64
```

In [116]:
```
maxi=np.asarray(df).max()
print(maxi)
```

```
7.8
```

In [117]:
```
df[df==7.8]
```

Out[117]:
```
url
/players/c/curryst01.html    7.8
Name: NBA__3ptapg, dtype: float64
```

In [118]:
```
College.loc[College['url']=='/players/c/curryst01.html']
```

Out[118]:

	Unnamed: 0	active_from	active_to	birth_date	college	height	name	position	
869	869	2010	2018	14-Mar-88	Davidson College	3-Jun	Stephen Curry	G	/players/c/

1 rows × 34 columns

Stephen Curry from the college, "Davidson College" has the maximum number of 3-points-goal-field attempts made per game in NBA

In [119]:
```
df = College.groupby("url").NBA__3ptpg.sum()
maxi=np.asarray(df).max()
url_x=df[df==maxi]
url_x
```

Out[119]:
```
url
/players/c/curryst01.html    3.4
Name: NBA__3ptpg, dtype: float64
```

In [120]:
```
College.loc[College['url']=="/players/c/curryst01.html"]
```

Out[120]:

	Unnamed: 0	active_from	active_to	birth_date	college	height	name	position	
869	869	2010	2018	14-Mar-88	Davidson College	3-Jun	Stephen Curry	G	/players/c/

1 rows × 34 columns

Stephen Curry from the college, "Davidson College" has the maximum number of 3-points-field-goals made per game in NBA.

In [121]:
```
df = College.groupby("url").NCAA__3ptapg.sum()
maxi=np.asarray(df).max()
url_x=df[df==maxi]
url_x
```

Out[121]:
```
url
/players/h/hudsotr01.html    10.8
Name: NCAA__3ptapg, dtype: float64
```

In [122]:
```
College.loc[College['url']=="/players/h/hudsotr01.html"]
```

Out[122]:

	Unnamed: 0	active_from	active_to	birth_date	college	height	name	position	
1862	1862	1998	2008	13-Mar-76	University of Missouri, Southern Illinois Univ...	1-Jun	Troy Hudson	G	/players/h

1 rows × 34 columns

Troy Hudson from the college, "University of Missouri" has the maximum number of 3-points-field-goals attempts made per game in NFCAA.

In [123]:
```
df=College.groupby("url").NCAA__3ptpg.sum()
maxi=np.asarray(df).max()
url_x=df[df==maxi]
url_x
```

Out[123]:
```
url
/players/c/curryst01.html    4.0
/players/h/hudsotr01.html    4.0
Name: NCAA__3ptpg, dtype: float64
```

Troy Hudson from the college, "University of Missouri" and Stephen Curray from the college,"Davidson College" has the maximum number of 3-points-field-goals made per game in NFCAA.

In [124]:
```
df = College.groupby("url").NBA_g_played.sum()
maxi=np.asarray(df).max()
url_x=df[df==maxi]
url_x
```

Out[124]:
```
url
/players/p/parisro01.html    1611
Name: NBA_g_played, dtype: int64
```

In [125]:
```
College.loc[College['url']=="/players/p/parisro01.html"]
```

Out[125]:

	Unnamed: 0	active_from	active_to	birth_date	college	height	name	position	
3119	3119	1977	1997	30-Aug-53	Centenary College of Louisiana	Jul-00	Robert Parish	C	/players/p/

1 rows × 34 columns

Rober Parish from ,"Centenary College of Louisiana" has played the highest number of the games in NBA

In [126]:
```
df = College.groupby("url").NCAA_games.sum()
maxi=np.asarray(df).max()
url_x=df[df==maxi]
url_x
```

Out[126]:
```
url
/players/m/milleda01.html    152.0
Name: NCAA_games, dtype: float64
```

In [127]:
```
College.loc[College['url']=="/players/m/milleda01.html"]
```

Out[127]:

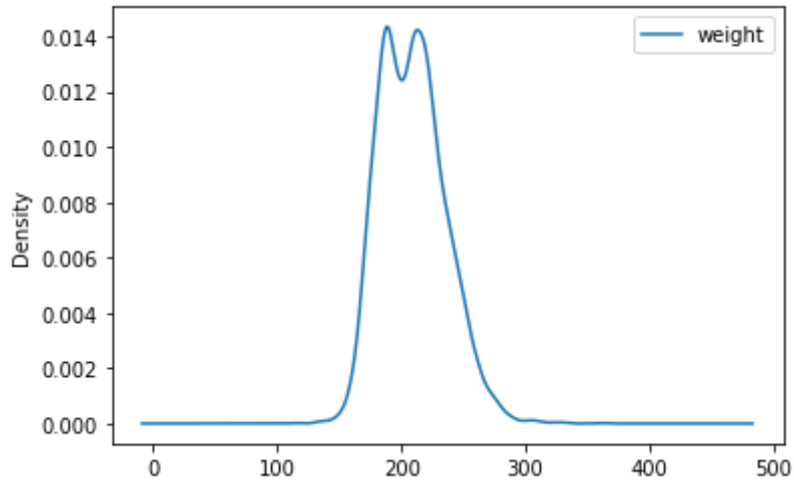
	Unnamed: 0	active_from	active_to	birth_date	college	height	name	position	
2768	2768	2013	2018	21-Mar-90	University of Kentucky	8-Jun	Darius Miller	F	/players/m/

1 rows × 34 columns

Rober Darius Miller ,"University of Kentucky" has played the highest number of the games in NCAA

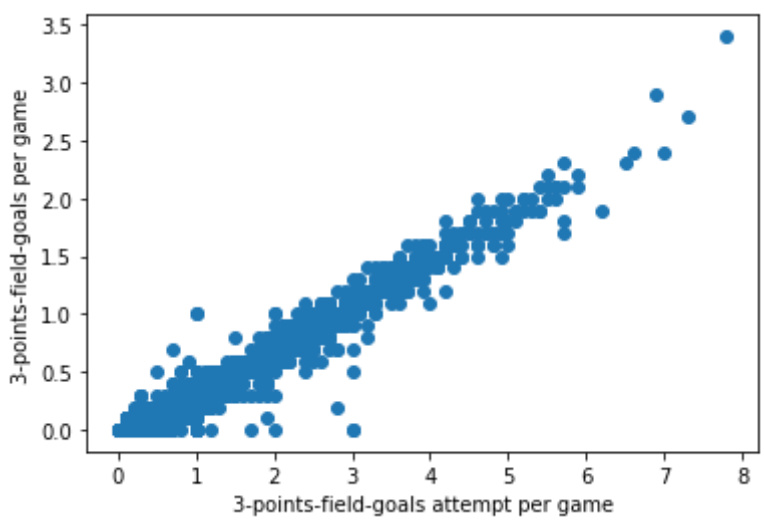
From the above, we can conclude that University of Kentucky, Davidson College, Centenary College of Louisiana, and University of Missouri produced some of the best players.

In [128]:
```
data = College[["height","weight"]]

data.plot(kind='density',layout = (1,1), subplots=True, sharex=False)
plt.show()
```



In [129]:
```
#This code will be used to plot a scatter plat
#to display values for typically two variables for a set of data and how
they are related.

plt.scatter(College["NBA__3ptapg"],College["NBA__3ptpg"])
plt.xlabel("3-points-field-goals attempt per game")
plt.ylabel("3-points-field-goals per game")
```

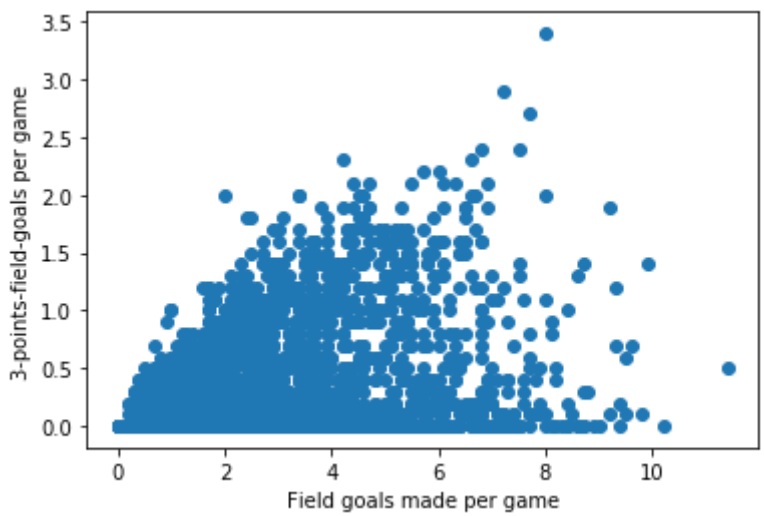
Out[129]: Text(0, 0.5, '3-points-field-goals per game')



It is clear from the above scatter plot that "3-points-field-goals attempt per game" and "3-points-field-goals per game". Now, this is logically true too.

In [130]:
```
#This code will be used to plot a scatter plat
#A scatter plot is a type of plot or mathematical diagram using
#cartesian coordinates to display values for typically two variables for
a set of data and how they are related.

plt.scatter(College["NBA_fg_per_game"],College["NBA__3ptpg"])
plt.xlabel("Field goals made per game")
plt.ylabel("3-points-field-goals per game")
```

Out[130]: Text(0, 0.5, '3-points-field-goals per game')

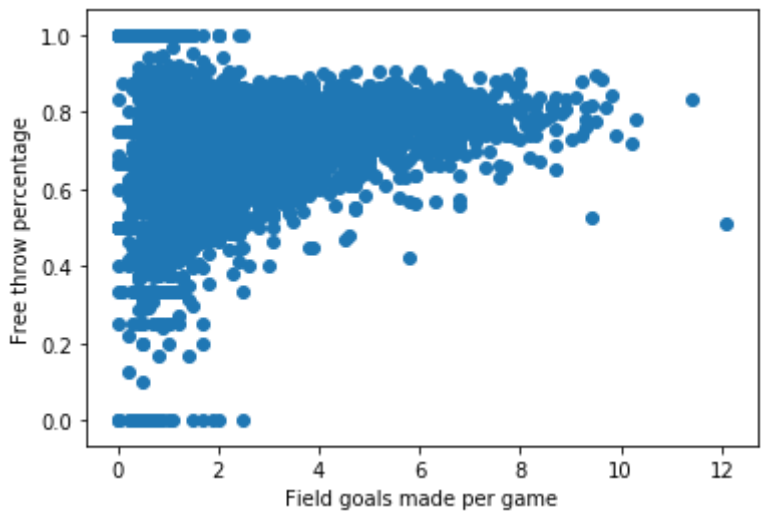


Scatter plot is used to show that total number of successful shots does not depend upon the 3-point-shoot as some of the experts say.

It is clear that there is almost no relationship between "Field goals made per game" and "3-points-field-goals per game"

In [131]:
```
#This code will be used to plot a scatter plat
#A scatter plot is a type of plot or mathematical diagram using
#cartesian coordinates to display values for typically two variables for
a set of data and how they are related.

plt.scatter(College["NBA_fg_per_game"],College["NBA_ft%"])
plt.xlabel("Field goals made per game")
plt.ylabel("Free throw percentage")
```

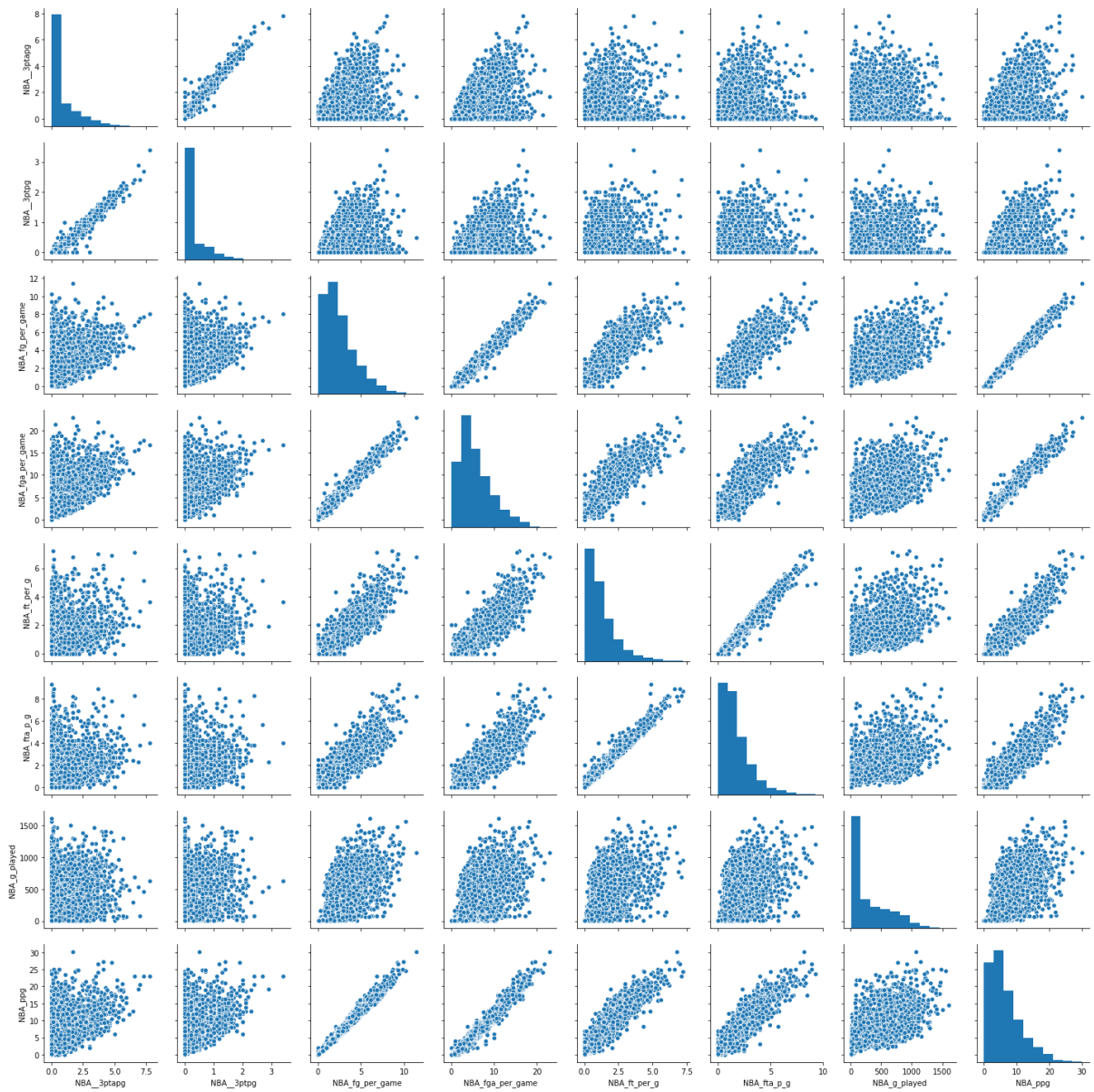
Out[131]: Text(0, 0.5, 'Free throw percentage')



Also, number of three throws don't impact the field goals made per game

In [132]:
```
#This code will be used to plot a scatter plat matrix
#A scatter plot is a type of plot or mathematical diagram using
#cartesian coordinates to display values for typically two variables for
a set of data and how they are related.
import seaborn as sns
sns.pairplot(College[['NBA__3ptapg',
       'NBA__3ptpg', 'NBA_fg_per_game',
       'NBA_fga_per_game', 'NBA_ft_per_g', 'NBA_fta_p_g',
       'NBA_g_played', 'NBA_ppg',]].dropna()) #dropna() method drops the
rows having NaN values
```

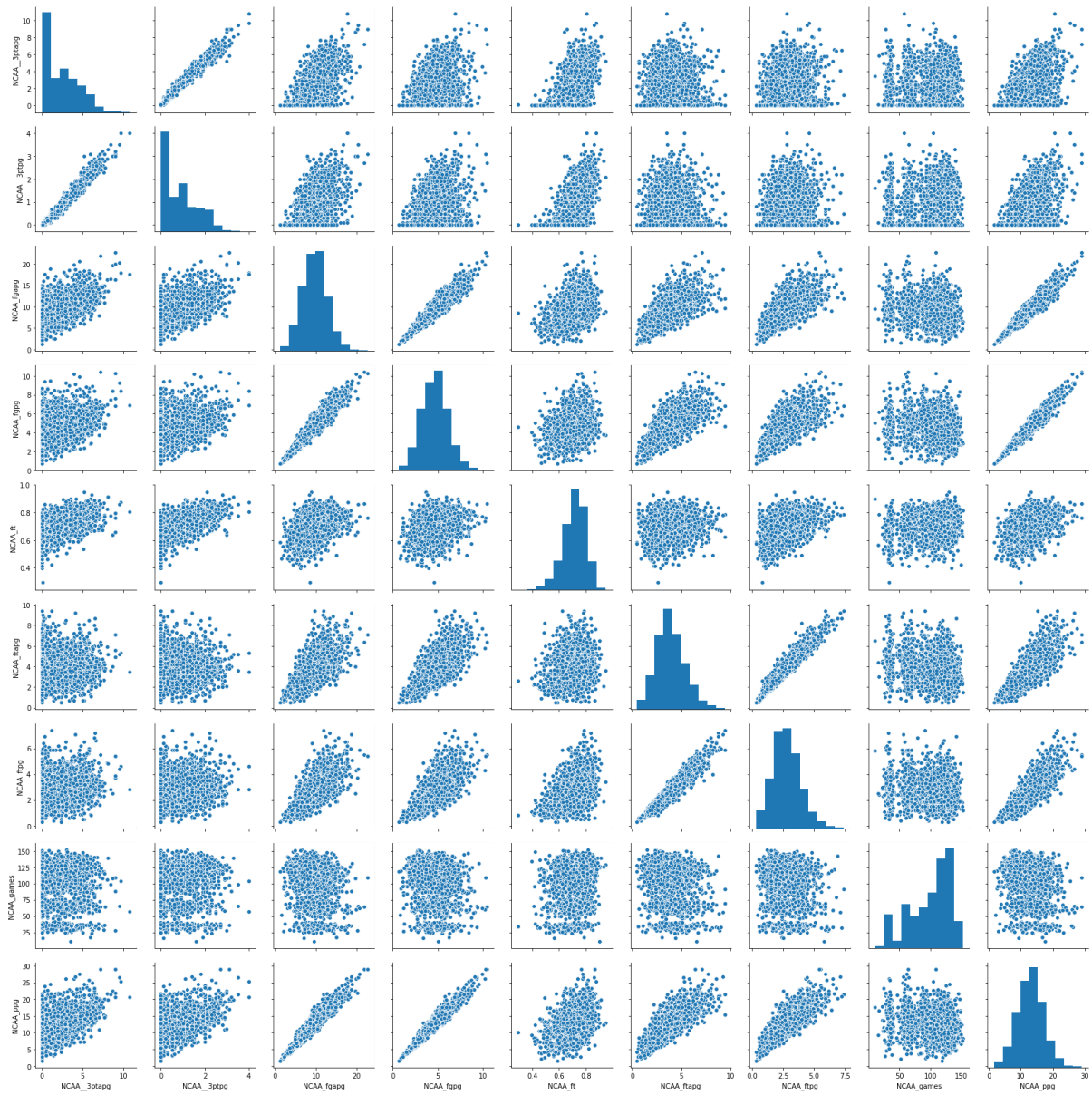
Out[132]: <seaborn.axisgrid.PairGrid object at 0x1c1f70eda0>



Now, we will check the stats between the feature for NCAA

In [133]:
```
#This code will be used to plot a scatter plat matrix
#A scatter plot is a type of plot or mathematical diagram using
#cartesian coordinates to display values for typically two variables for
a set of data and how they are related.

import seaborn as sns

sns.pairplot(College[['NCAA__3ptapg',
       'NCAA__3ptpg', 'NCAA_fgapg', 'NCAA_fgpg',
       'NCAA_ft', 'NCAA_ftapg', 'NCAA_ftpg', 'NCAA_games', 'NCAA_ppg']].
dropna()) #dropna() method drops the rows having NaN values
```

Out[133]: <seaborn.axisgrid.PairGrid object at 0x1c32c47438>



We can visualize the relationship using the scatter matrices

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: