MDSC-102-Final_Lab_Exam

1. Dataset Description

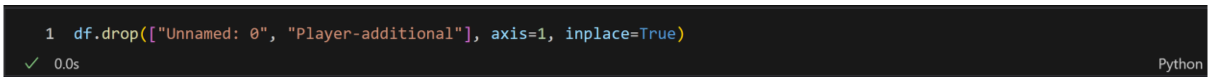
This dataset consists of various statistical aspects and information about the NBA players of the season 2022-2023 along with their salary data. This dataset gives us the opportunity to get a comprehensive understanding of the relation between a professional basketball player's performance and monetary gain.

The features present in the dataset are:

- Player Name
- Salary – Salary in US Dollars
- Position – Position Played
- Age – Age of the Player
- Team – Teams played in 2022-23 season.
- GP - Games Played
- GS – Game Started
- MP – Minutes Per Game
- FG – Field Goals Per Game
- FGA – Field Goals Attempts Per Game
- FG% - Field Goal Percentage
- 3P – Three Field Goals Made Per Game
- 3PA – Three Point Field Goal Attempts Per Game
- 3P% - Three Point Percentage
- 2P – Two Points Field Goals Made Per Game
- 2PA – Two Point Field Goal Attempts Per Game
- 2P% - Two Point Percentage
- eFG% - Effective Field Goal Percentage
- FT – Free Throws Made Per Game
- FTA – Free Throw Attempts Per Game
- FT% - Free Throw Percentage
- ORB – Offensive Rebounds Per Game
- DRB – Defensive Rebounds Per Game
- TRB – Total Rebounds Per Game
- AST – Assists Per Game
- STL – Steals Per Game
- BLK – Blocks Per Game
- TOV – Turnovers Per Game
- PF – Personal Fouls Per Game
- PTS – Points Per Game
- Player Additional

2. Data Cleaning

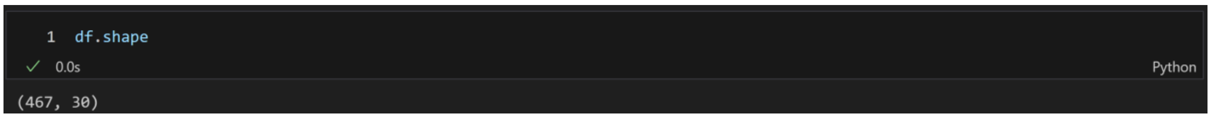
Removal of Redundant/Unnecessary features



```
df.drop(["Unnamed: 0", "Player-additional"], axis=1, inplace=True)
✓ 0.0s                                                                Python
```

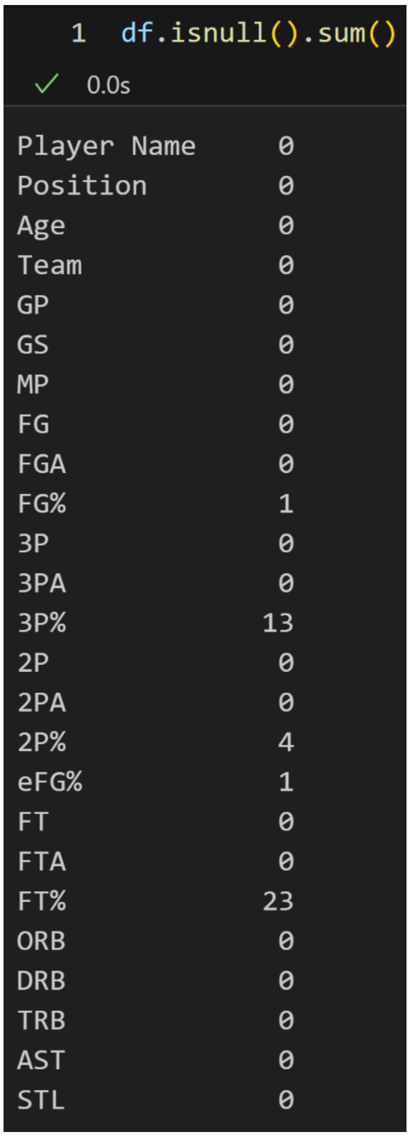
The feature “Unnamed: 0” was a redundant feature which didn’t really have any meaning to it. The feature “Player-additional” has no relevance to the subject at hand.

Therefore, now, the data frame consists of 30 columns and 467 rows.



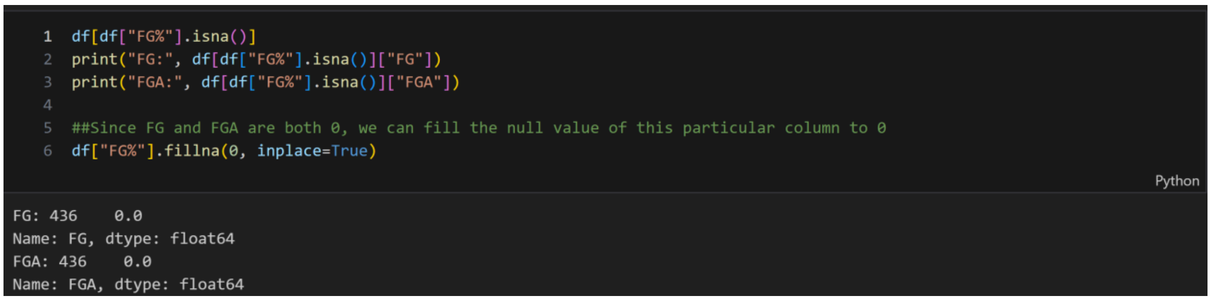
```
df.shape
✓ 0.0s                                                                Python
(467, 30)
```

The NBA Salary dataset was check for missing values and it turned out that it has very few missing values present.



```
df.isnull().sum()
✓ 0.0s

Player Name     0
Position        0
Age             0
Team            0
GP              0
GS              0
MP              0
FG              0
FGA             0
FG%             1
3P              0
3PA             0
3P%            13
2P              0
2PA             0
2P%             4
eFG%            1
FT              0
FTA             0
FT%            23
ORB             0
DRB             0
TRB             0
AST             0
STL             0
```



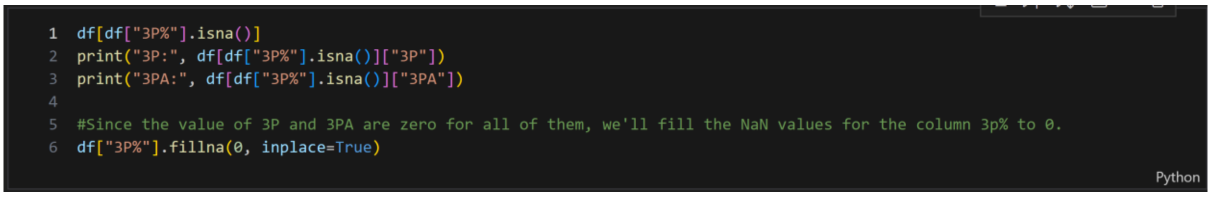
```python
df[df["FG%"].isna()]
print("FG:", df[df["FG%"].isna()]["FG"])
print("FGA:", df[df["FG%"].isna()]["FGA"])

##Since FG and FGA are both 0, we can fill the null value of this particular column to 0
df["FG%"].fillna(0, inplace=True)
```

Python

```
FG: 436    0.0
Name: FG, dtype: float64
FGA: 436    0.0
Name: FGA, dtype: float64
```

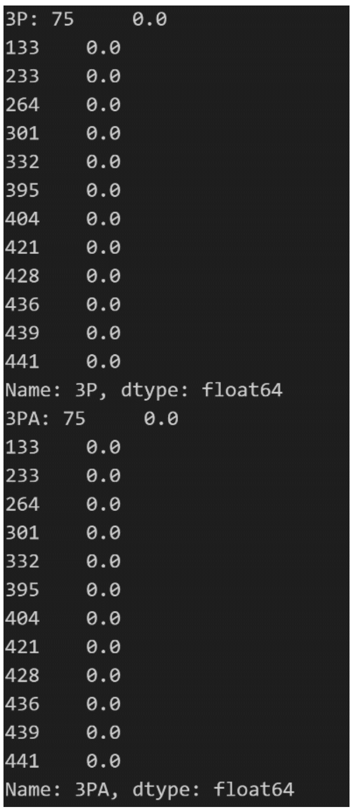
Since only 1 cell of the Field Goal Percentage (FG%) has a null value, we're checking what are the corresponding values of Field Goals Made Per Game (FG) and Field Goal Attempts Per Game (FGA) along that particular row. Since the values of FG and FGA are both zero, the null value of FG% is replaced with zero.



```python
df[df["3P%"].isna()]
print("3P:", df[df["3P%"].isna()]["3P"])
print("3PA:", df[df["3P%"].isna()]["3PA"])

#Since the value of 3P and 3PA are zero for all of them, we'll fill the NaN values for the column 3p% to 0.
df["3P%"].fillna(0, inplace=True)
```

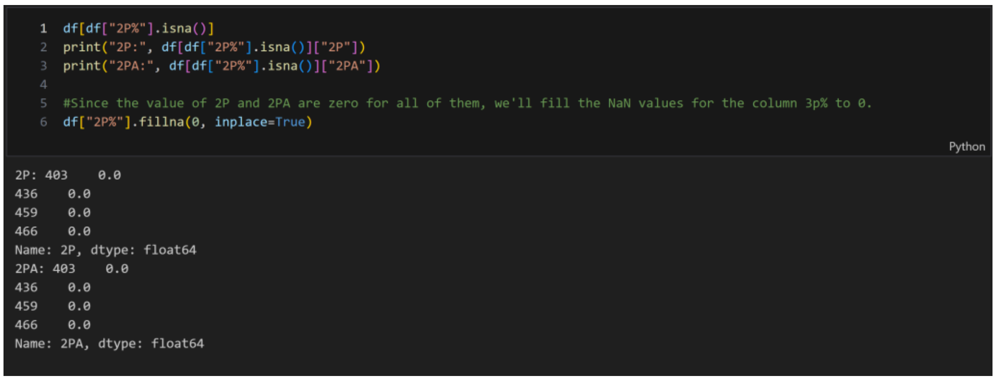
Python



```
3P: 75     0.0
133    0.0
233    0.0
264    0.0
301    0.0
332    0.0
395    0.0
404    0.0
421    0.0
428    0.0
436    0.0
439    0.0
441    0.0
Name: 3P, dtype: float64
3PA: 75     0.0
133    0.0
233    0.0
264    0.0
301    0.0
332    0.0
395    0.0
404    0.0
421    0.0
428    0.0
436    0.0
439    0.0
441    0.0
Name: 3PA, dtype: float64
```

Similarly, for the Three Point Percentage (3P%), we're checking the corresponding values of Three Field Points Per Game (3P) and Three Point Field Goal Attempts Per Game (3PA) along the rows of the null values. Since all of them came out to be zero, the null values in the 3P% column is filled up with zeros.

Similarly, the same procedure is followed for the null values present in the 2P%, eFG% and FT% columns.

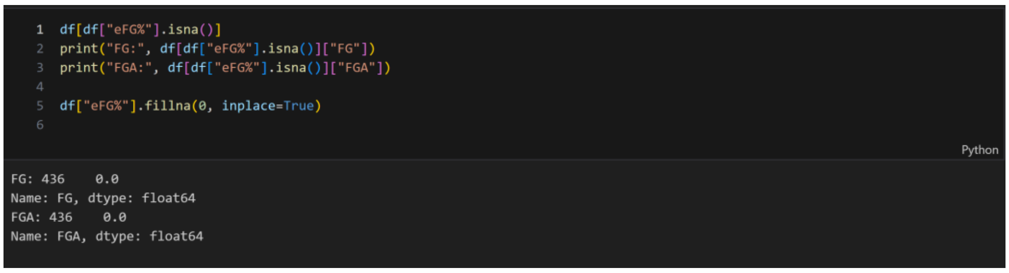


```python
df[df["2P%"].isna()]
print("2P:", df[df["2P%"].isna()]["2P"])
print("2PA:", df[df["2P%"].isna()]["2PA"])

#Since the value of 2P and 2PA are zero for all of them, we'll fill the NaN values for the column 3p% to 0.
df["2P%"].fillna(0, inplace=True)
```

Python

```
2P: 403    0.0
436    0.0
459    0.0
466    0.0
Name: 2P, dtype: float64
2PA: 403    0.0
436    0.0
459    0.0
466    0.0
Name: 2PA, dtype: float64
```



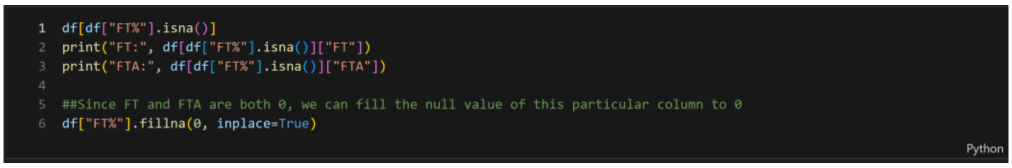
```python
df[df["eFG%"].isna()]
print("FG:", df[df["eFG%"].isna()]["FG"])
print("FGA:", df[df["eFG%"].isna()]["FGA"])

df["eFG%"].fillna(0, inplace=True)

```

Python

```
FG: 436    0.0
Name: FG, dtype: float64
FGA: 436    0.0
Name: FGA, dtype: float64
```



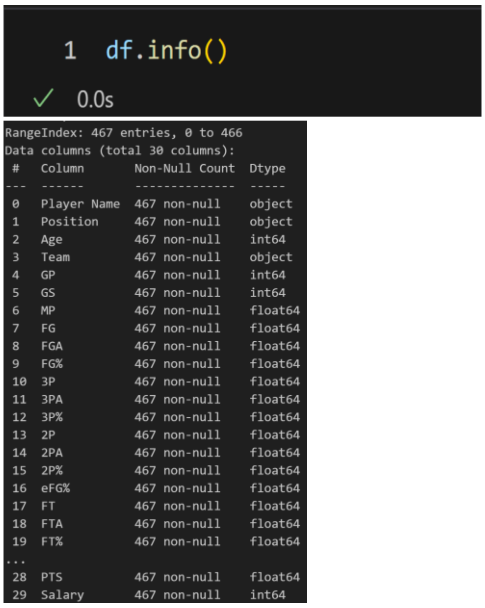
```python
df[df["FT%"].isna()]
print("FT:", df[df["FT%"].isna()]["FT"])
print("FTA:", df[df["FT%"].isna()]["FTA"])

##Since FT and FTA are both 0, we can fill the null value of this particular column to 0
df["FT%"].fillna(0, inplace=True)
```

Python

Apart from the missing values, the dataset was pretty clean. Now that the data cleaning is done, we'll move on to the summarization of the data.

3. Summary and Statistics



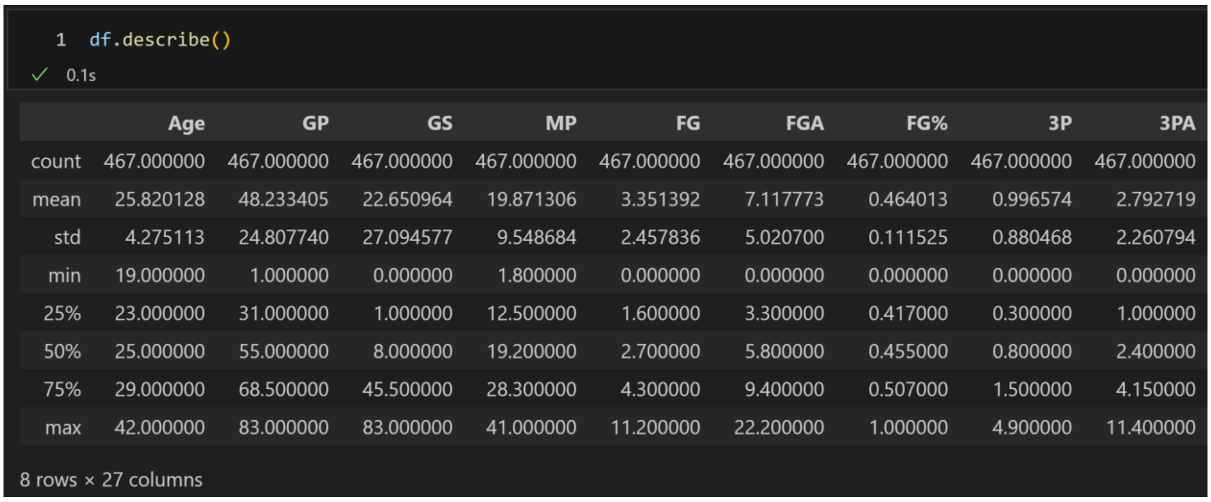
```python
df.info()
```

✓ 0.0s

```
RangeIndex: 467 entries, 0 to 466
Data columns (total 30 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Player Name  467 non-null    object
 1   Position     467 non-null    object
 2   Age          467 non-null    int64
 3   Team         467 non-null    object
 4   GP           467 non-null    int64
 5   GS           467 non-null    int64
 6   MP           467 non-null    float64
 7   FG           467 non-null    float64
 8   FGA          467 non-null    float64
 9   FG%          467 non-null    float64
 10  3P           467 non-null    float64
 11  3PA          467 non-null    float64
 12  3P%          467 non-null    float64
 13  2P           467 non-null    float64
 14  2PA          467 non-null    float64
 15  2P%          467 non-null    float64
 16  eFG%         467 non-null    float64
 17  FT           467 non-null    float64
 18  FTA          467 non-null    float64
 19  FT%          467 non-null    float64
...
 28  PTS          467 non-null    float64
 29  Salary       467 non-null    int64
```

df.info() gives a few details about the data type of the data present in each column and also the amount of null values present in each column.

The describe() function gives a very comprehensive and descriptive statistics about each column as shown below:



```
df.describe()
```

✓ 0.1s

```
             Age          GP          GS          MP          FG         FGA         FG%          3P         3PA
count 467.000000  467.000000  467.000000  467.000000  467.000000  467.000000  467.000000  467.000000  467.000000
mean   25.820128   48.233405   22.650964   19.871306    3.351392    7.117773    0.464013    0.996574    2.792719
std     4.275113   24.807740   27.094577    9.548684    2.457836    5.020700    0.111525    0.880468    2.260794
min    19.000000    1.000000    0.000000    1.800000    0.000000    0.000000    0.000000    0.000000    0.000000
25%    23.000000   31.000000    1.000000   12.500000    1.600000    3.300000    0.417000    0.300000    1.000000
50%    25.000000   55.000000    8.000000   19.200000    2.700000    5.800000    0.455000    0.800000    2.400000
75%    29.000000   68.500000   45.500000   28.300000    4.300000    9.400000    0.507000    1.500000    4.150000
max    42.000000   83.000000   83.000000   41.000000   11.200000   22.200000    1.000000    4.900000   11.400000

8 rows × 27 columns
```

The above statistics will come in handy when we're trying the find the confidence interval of certain parameters or even when testing hypotheses.

4. Data Visualization and Inferences

Now, we'll try to visualize the data at hand and find relations between the various features that we have. Furthermore, we'll try to make inferences out of them to get a more vivid understanding of how the features are behaving.

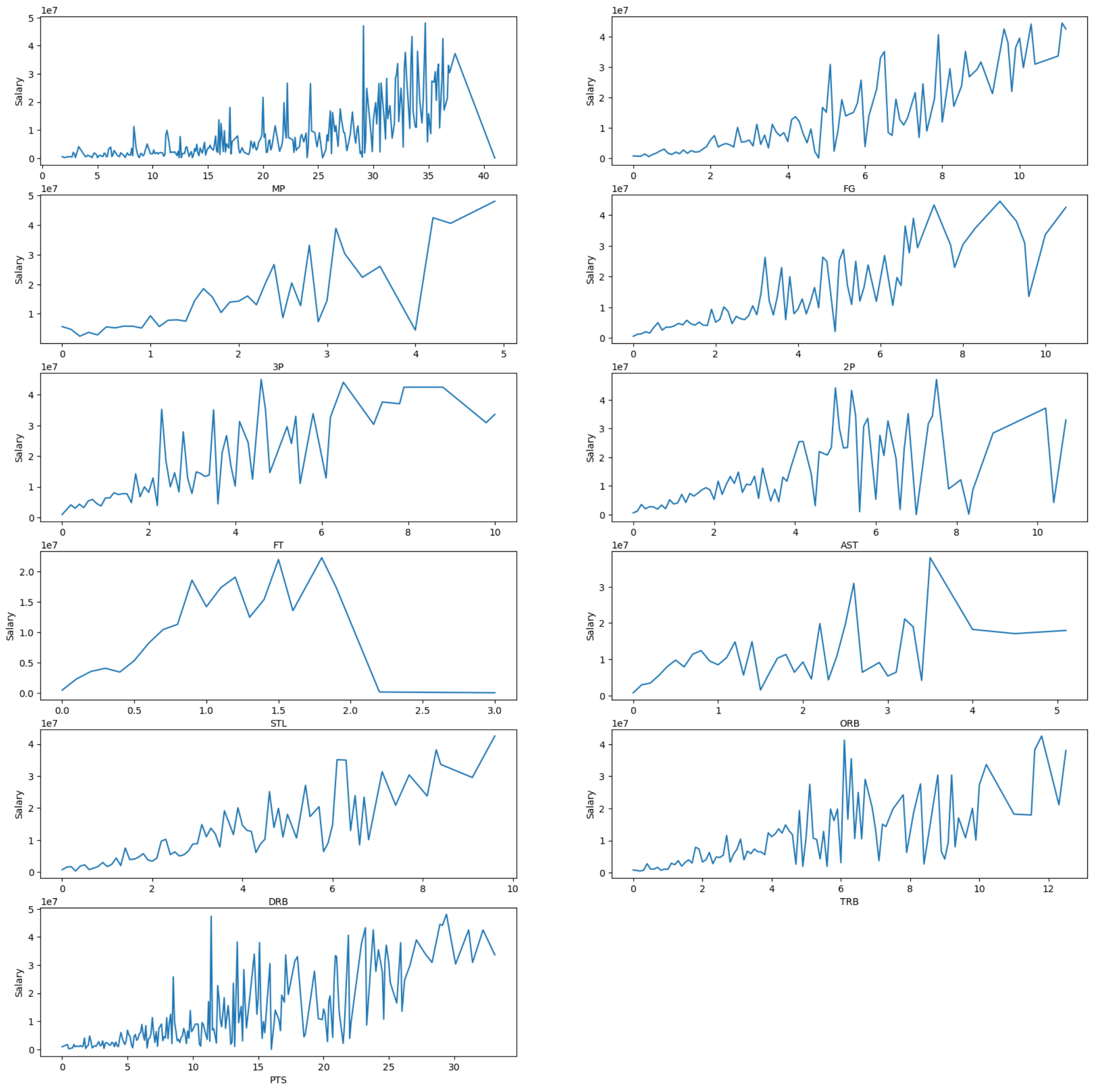
(i) How does the salary of the players vary with respect to some selected and relevant features.

We're trying to find and understand what all features affect the growth of the salary of the players.

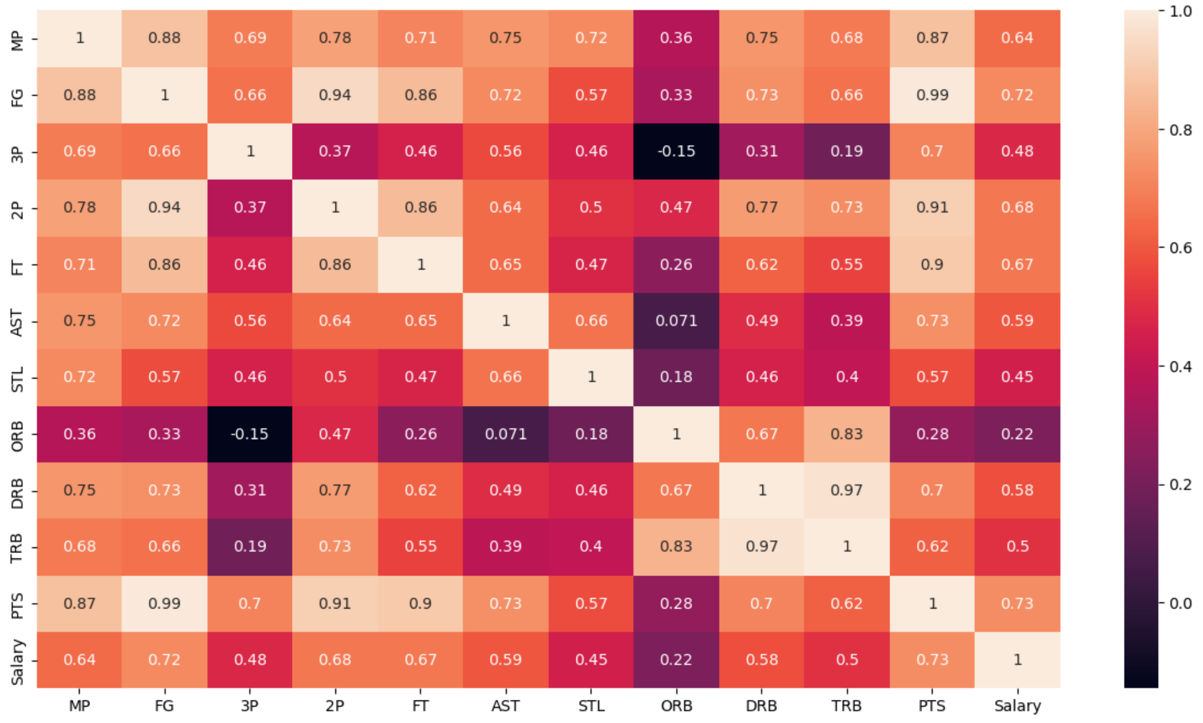
- **MP vs Salary:** Even though there's a lot of fluctuations in the relationship between MP and Salary, there's somewhat of a general increase in the salary as MP increases.

- **FG vs Salary:** Similarly, there might be a certain degree of fluctuations between the features FG and Salary, but we can see a general increase in the Salary as FG increases.
- **3P vs Salary:** From 0 to 3, there's a general increase in the salary of the players as the number of three points made per game increase, but from 3 to 4, there's considerable decrease in the salary. From 4 onwards, there's a sharp increase in the salary.
- **2P vs Salary:** From 0 to 7 on the 2 pointers made, there's a general increase in the salary of the players. From 7 onwards, we can see various sharp increases and decreases in the salary of the players.

- **FT vs Salary:** The salary increases as the number of free throws made per game increases.
- **AST vs Salary:** We can’t really find a proper relationship between the number of assists made per game and the salary of the players since there’s way too much of fluctuations in the line plot.
- **STL vs Salary:** From 0 to around 2.75, the Salary increases as the number of steals increases. But afterwards, there’s a steep decrease in the value of the Salary.
- **ORB vs Salary:**
- **DRB vs Salary:** We can infer from the line plot that there’s a general increase in the salary when the number of Defensive Rebounds per Game increases.
- **TRB vs Salary:** Similarly, there’s a general increase in the salary when the number of Total Rebounds per Game increases.
- **PTS vs Salary:** Even though there’s a lot of fluctuations in the relationship between PTS and Salary, there’s somewhat of a general increase in the salary as the number of points increases.



(ii) Use of Heatmap to find the correlation between features.



	MP	FG	3P	2P	FT	AST	STL	ORB	DRB	TRB	PTS	Salary
MP	1	0.88	0.69	0.78	0.71	0.75	0.72	0.36	0.75	0.68	0.87	0.64
FG	0.88	1	0.66	0.94	0.86	0.72	0.57	0.33	0.73	0.66	0.99	0.72
3P	0.69	0.66	1	0.37	0.46	0.56	0.46	-0.15	0.31	0.19	0.7	0.48
2P	0.78	0.94	0.37	1	0.86	0.64	0.5	0.47	0.77	0.73	0.91	0.68
FT	0.71	0.86	0.46	0.86	1	0.65	0.47	0.26	0.62	0.55	0.9	0.67
AST	0.75	0.72	0.56	0.64	0.65	1	0.66	0.071	0.49	0.39	0.73	0.59
STL	0.72	0.57	0.46	0.5	0.47	0.66	1	0.18	0.46	0.4	0.57	0.45
ORB	0.36	0.33	-0.15	0.47	0.26	0.071	0.18	1	0.67	0.83	0.28	0.22
DRB	0.75	0.73	0.31	0.77	0.62	0.49	0.46	0.67	1	0.97	0.7	0.58
TRB	0.68	0.66	0.19	0.73	0.55	0.39	0.4	0.83	0.97	1	0.62	0.5
PTS	0.87	0.99	0.7	0.91	0.9	0.73	0.57	0.28	0.7	0.62	1	0.73
Salary	0.64	0.72	0.48	0.68	0.67	0.59	0.45	0.22	0.58	0.5	0.73	1

- **PTS**
  We can clearly see that there's a positive correlation between the PTS (points) features and the other features which means that as the other features increase, the points also increase. A notable example is as field goal per game of a player increases, then obviously the number of points scored will also increase. They have a correlation coefficient of 0.99.

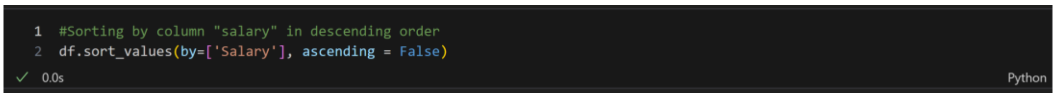
- **ORB (Offensive Rebounds) vs 3P (# Points) (-0.15):**
  Since the correlation coefficient is negative, we can deduce that as a player makes more offensive rebounds, they tend to score less 3 pointers. Therefore, we can infer that those players mostly focus on 2 pointers and stay near the loop for any offensive rebounds. To back this up, we can see that the **correlation coefficient between ORB and 2P is positive (0.47),** which means that as ORB increases, the number of 2 pointers also increases.

- **FG (Field Goal per Game) vs 2P (2 Points per Game) (0.94):**
  The relation between FG and 2P has one of the highest correlation coefficient, which means that as compared to the other features, 2 pointers contribute more to the field goal percentage of a player.

- **Salary vs (Other features)**
  All the features are positively correlated to the salary of a player, but we can see that FG (Field Goal per game) and PTS (Points) are more correlated to

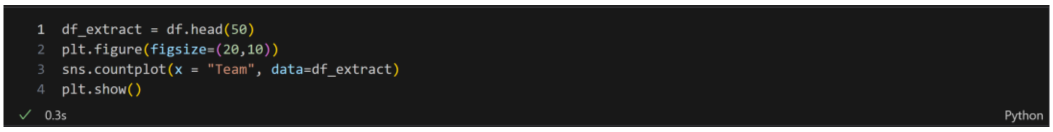
salary 0.72 and 0.73 respectively. Therefore, a player having more FG or PTS will mostly have a higher salary as compared to the other players.

(iii) Among the top 50 most paid players, we're finding which position is the most sought out and seems to be paid more.

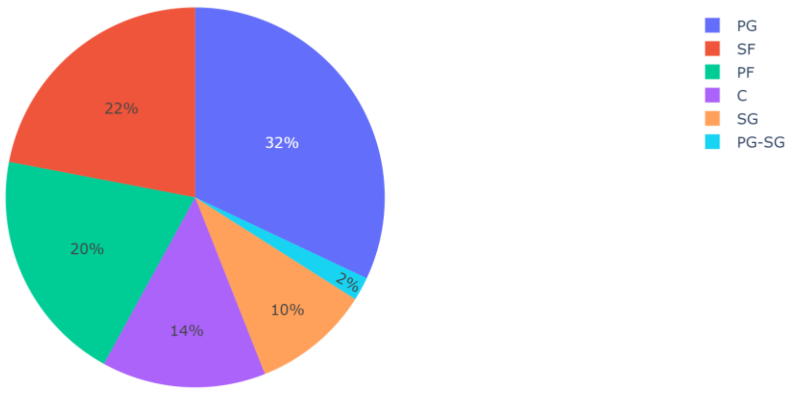


```python
#Sorting by column "salary" in descending order
df.sort_values(by=['Salary'], ascending = False)
```

First, we'll sort the values in the salary feature in descending order. Then we'll extract the first 50 rows of that particular column.

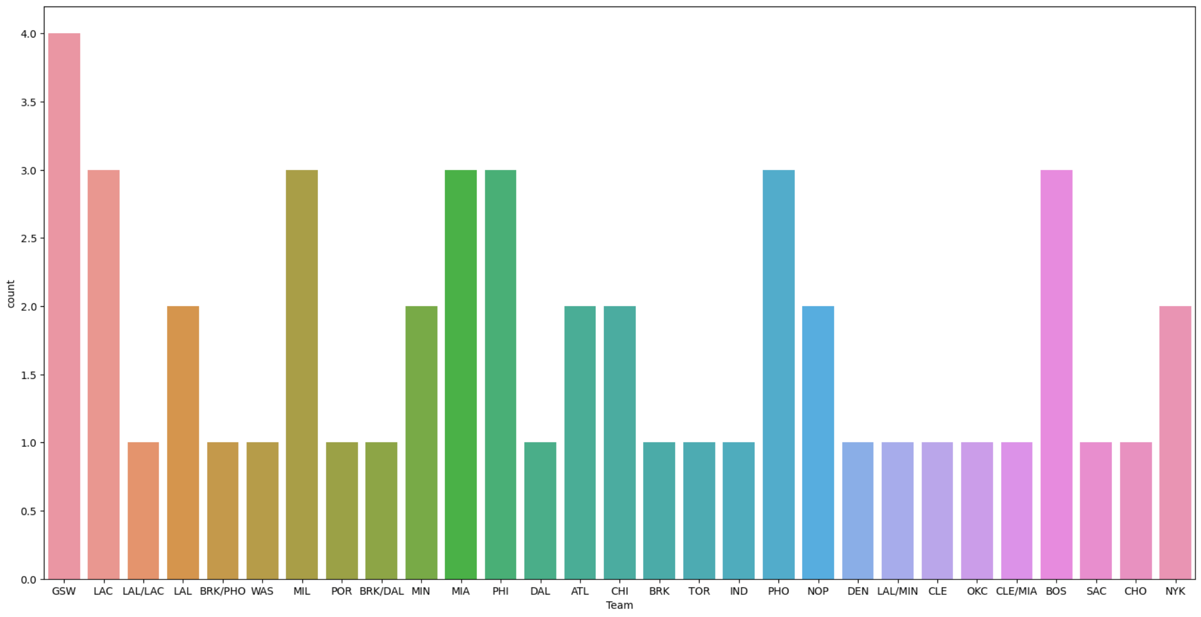


```python
df_extract = df.head(50)
plt.figure(figsize=(20,10))
sns.countplot(x = "Team", data=df_extract)
plt.show()
```



We can see that 32% of the top paid players play in the Point Guard(PG) position which is the highest percentage of all positions. The subsequent breakdown of the percentage is as shown below:
22% - Small Forward
20% - Power Forward
14% - Center
10% - Shooting Guard
2% - Athlete playing both as a Point Guard and a Shooting Guard

(iv) Which team has the most players featured in the top 50 most paid NBA athletes?



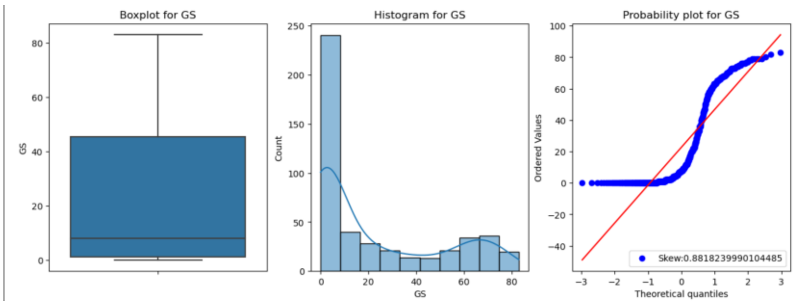
We can clearly see that the Golden State Warriors (GSW) have the most players (4 in total) in the list of the top 50 most paid players in the NBA.

Afterwards, there's a tie between 6 teams, namely LA Clippers (LAC), Milwaukee Bucks (MIL), Miami Heat (MIA), Philadelphia 76ers(PHI), Phoenix Suns (PHO) and Boston Celtics (BOS) which each have 3 players in the list of top 50 most paid players.

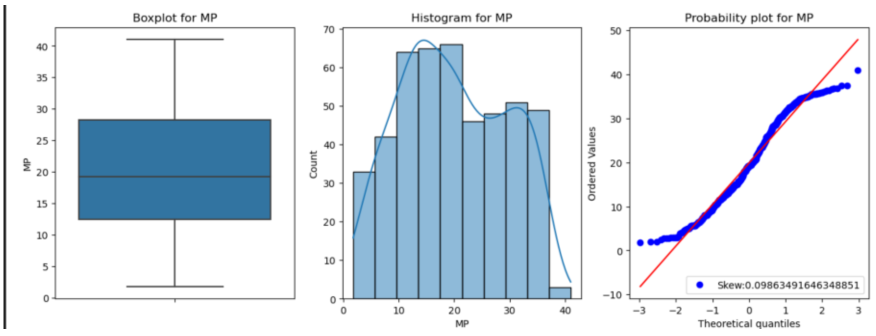
Subsequently, the Los Angeles Lakers (LAL), Minnesota Timberwolves (MIN), Atlanta Hawks (ATL), Chicago Bulls (CHI), New Orleans Pelicans (NOP), and the New York Knicks (NYK) all have only 2 players in the mentioned list.

Finally, all the remaining teams not mentioned which is present in the bar plot have only 1 player in the list of top 50 most paid players.

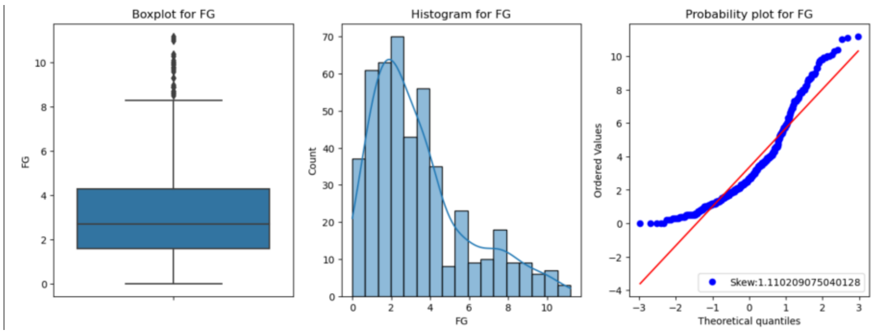
(v) Distribution of the features



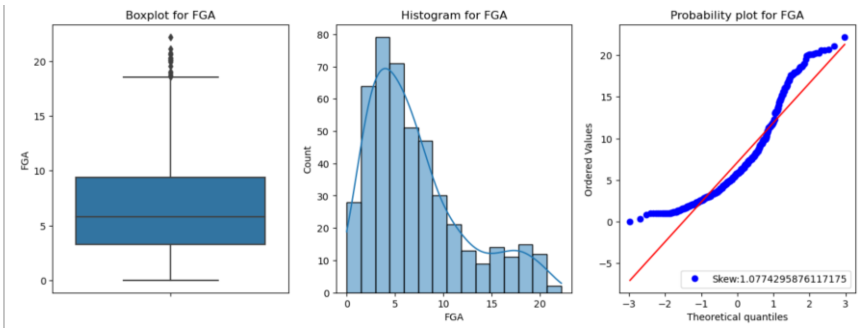
➢ Skewness of GS – positively skewed since most of the data is lying towards the right of the boxplot and the right tail is longer. Also, the skew value is positive.



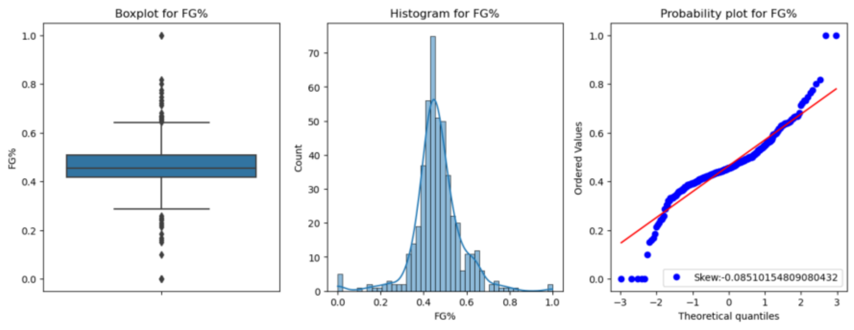
- Skewness of MP – almost normally skewed since the skew value is near to 0 and the length of both tails in the boxplot are almost the same.



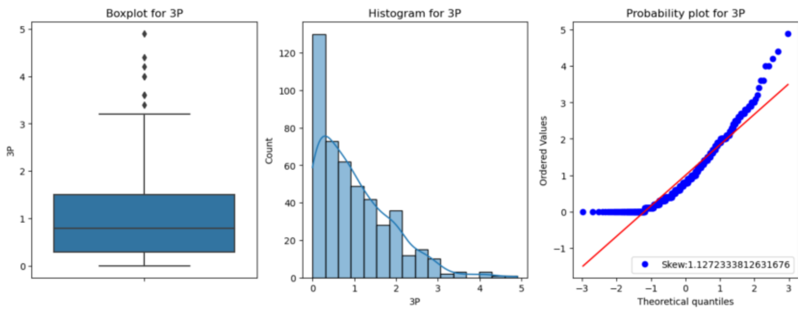
- Skewness of FG – positively skewed and contains outliers since the right tail of the boxplot is longer, the tail of the histogram is more skewed towards the right and the skew value is positive.



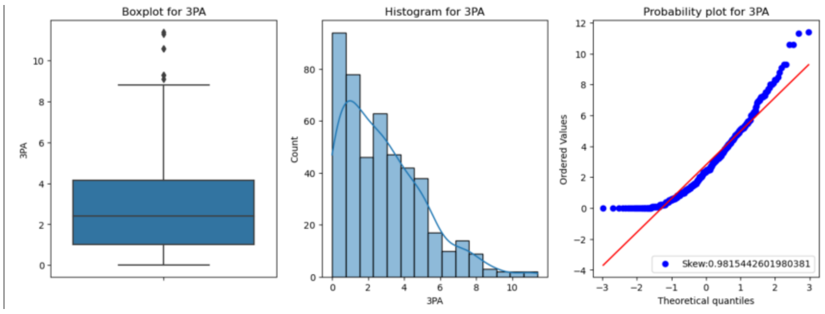
- Skewness of FGA – positively skewed and contains outliers since the right tail of the boxplot is longer, the tail of the histogram is more skewed towards the right and the skew value is positive.



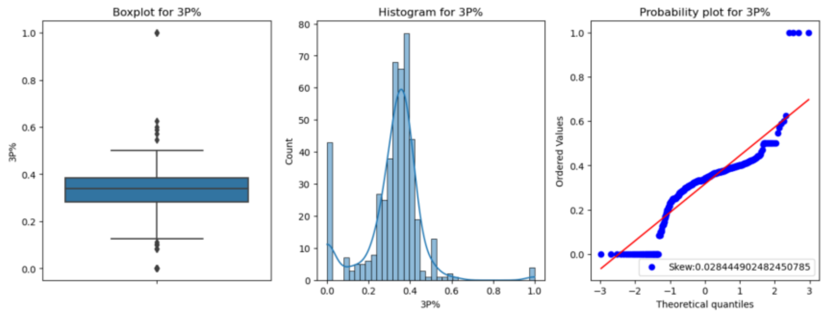
- Skewness of FG% – almost normal but have a lot of outliers on both sides of the boxplot. The histogram looks bell-shaped and centered around the mean. Also, the skew value is almost 0.



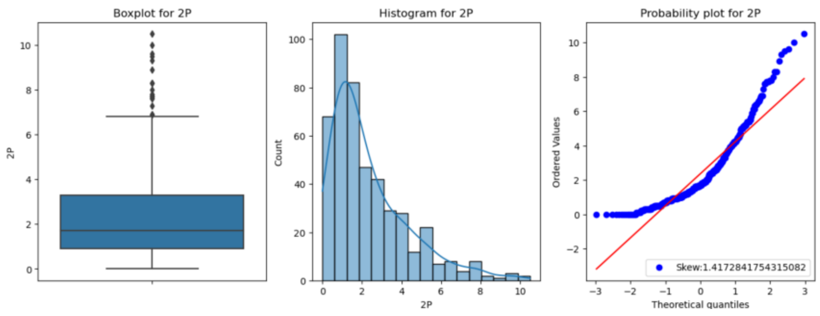
- Skewness of 3P – positively skewed with some outliers since the right tail of the boxplot is longer, the tail of the histogram is more skewed towards the right and the skew value is positive.



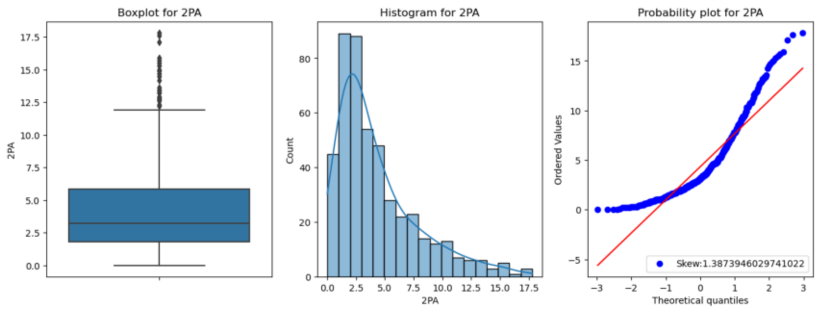
- Skewness of 3PA – positively skewed with some outliers since the right tail of the boxplot is longer, the tail of the histogram is more skewed towards the right and the skew value is positive.



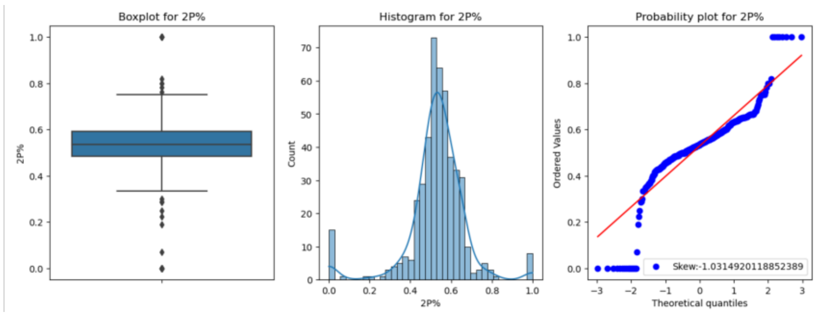
- Skewness of 3P% – almost normal but have a lot of outliers on both sides of the boxplot. The histogram looks bell-shaped and centered around the mean. Also, the skew value is almost 0.



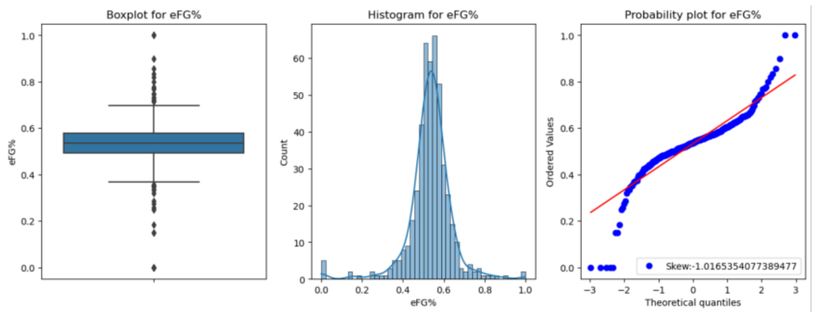
- Skewness of 2P – positively skewed with outliers since the right tail of the boxplot is longer, the tail of the histogram is more skewed towards the right and the skew value is positive.



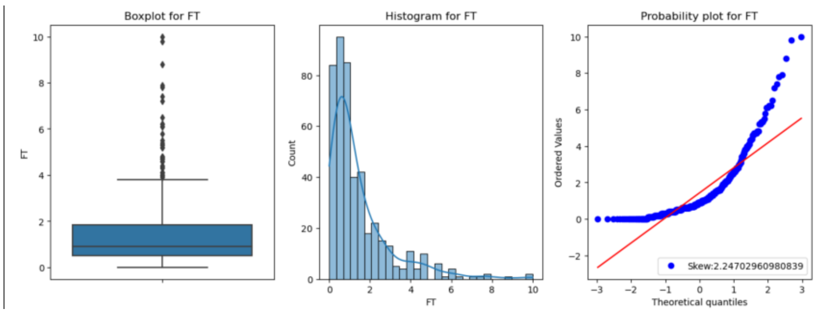
- Skewness of 2PA – positively skewed with outliers since the right tail of the boxplot is longer, the tail of the histogram is more skewed towards the right and the skew value is positive.



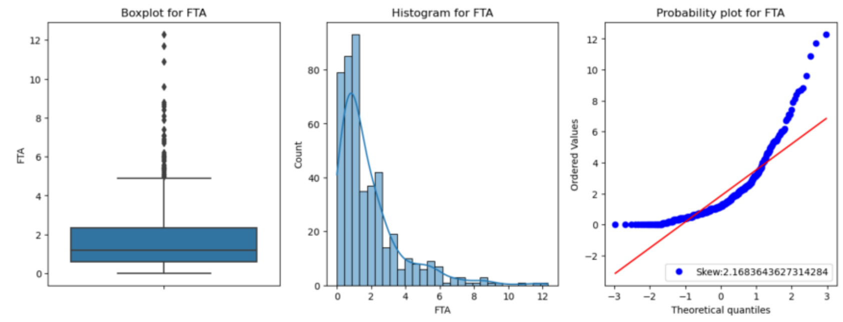
- Skewness of 2P% – almost normal with some outliers since the length of both the tails are the same and the histogram looks bell-shaped and centered around the mean.



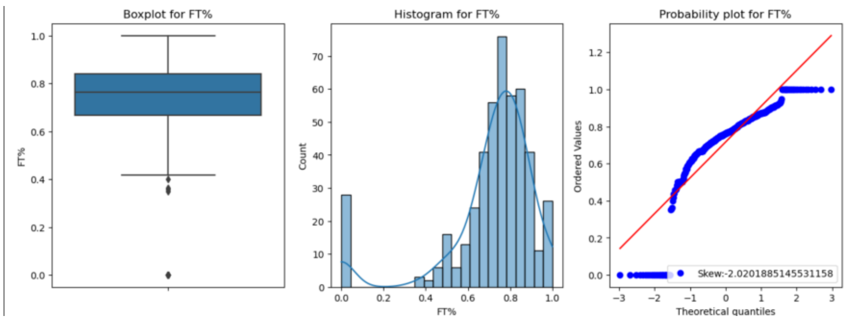
- Skewness of eFG% - almost normal with some outliers since the length of both the tails are the same and the histogram looks bell-shaped and centered around the mean.



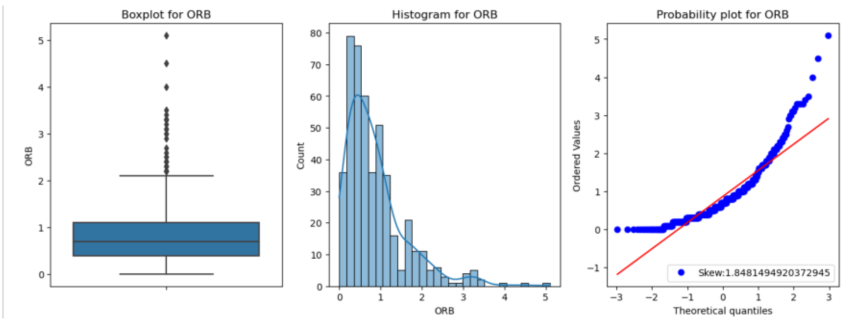
- Skewness of FT – positively skewed with lots of outliers since the right tail of the boxplot is longer, the tail of the histogram is more skewed towards the right and the skew value is positive.



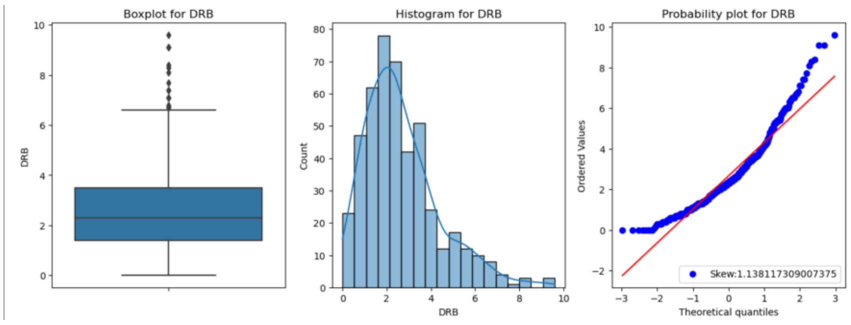
- Skewness of FTA – positively skewed with lots of outliers since the right tail of the boxplot is longer, the tail of the histogram is more skewed towards the right and the skew value is positive.



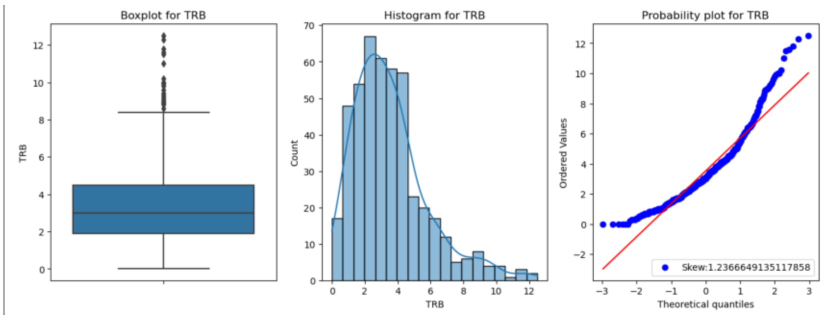
- Skewness of FT% – negatively skewed with few outliers since the length of the left tail of the boxplot is longer and the histogram is more skewed towards the left. Furthermore, the skew value is negative.



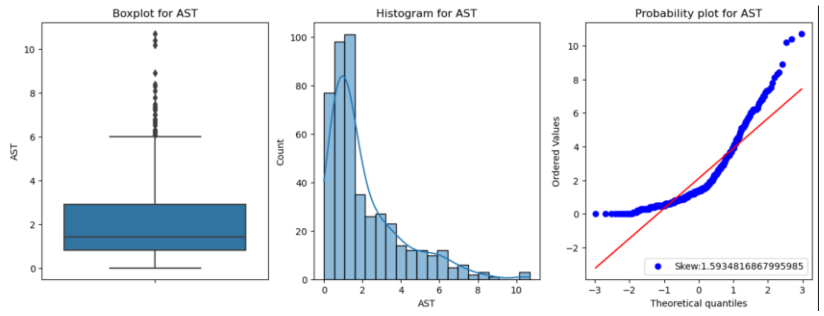
- Skewness of ORB – positively skewed with lots of outliers since the right tail of the boxplot is longer, the tail of the histogram is more skewed towards the right and the skew value is positive.



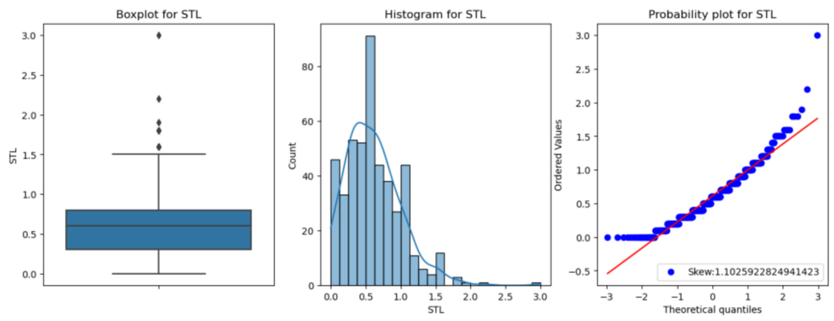
- Skewness of DRB – positively skewed with few outliers since the right tail of the boxplot is longer, the tail of the histogram is more skewed towards the right and the skew value is positive.



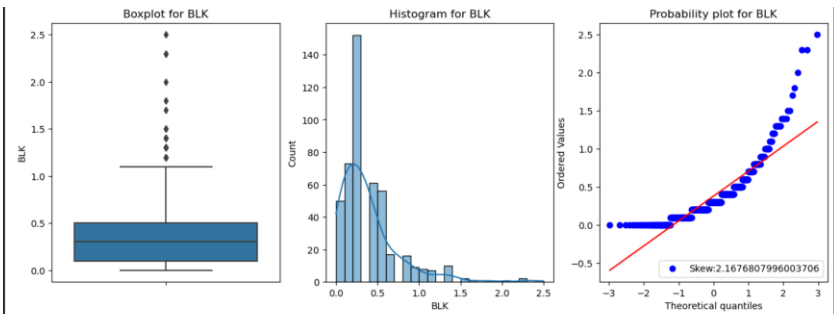
- Skewness of TRB – positively skewed with lots of outliers since the right tail of the boxplot is longer, the tail of the histogram is more skewed towards the right and the skew value is positive.



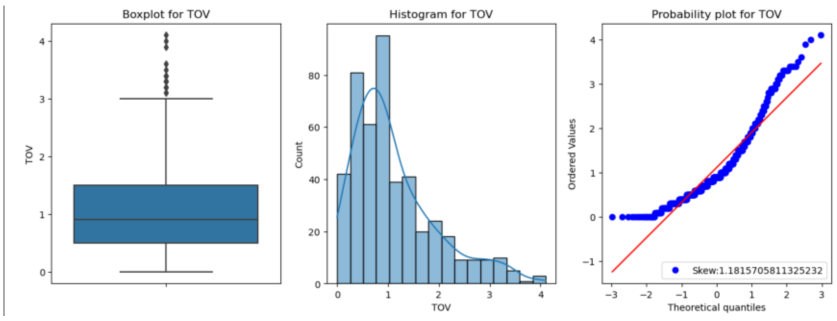
- Skewness of AST – positively skewed with lots of outliers since the right tail of the boxplot is longer, the tail of the histogram is more skewed towards the right and the skew value is positive.



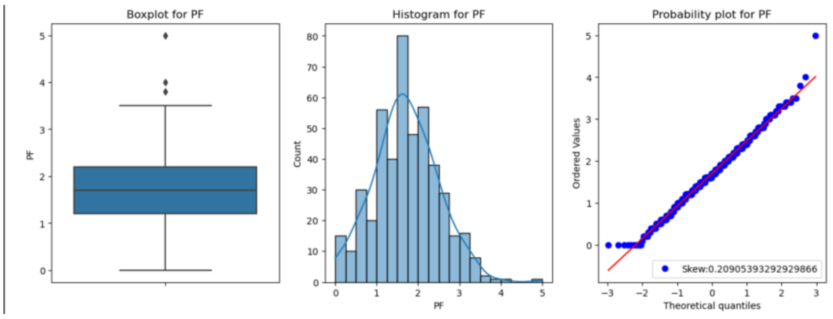
- ➢ Skewness of STL – positively skewed with few outliers since the right tail of the boxplot is longer, the tail of the histogram is more skewed towards the right and the skew value is positive.



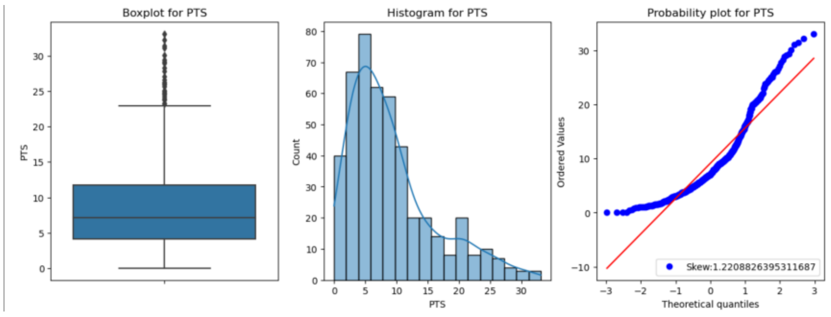
- ➢ Skewness of BLK – positively skewed with some outliers since the right tail of the boxplot is longer, the tail of the histogram is more skewed towards the right and the skew value is positive.



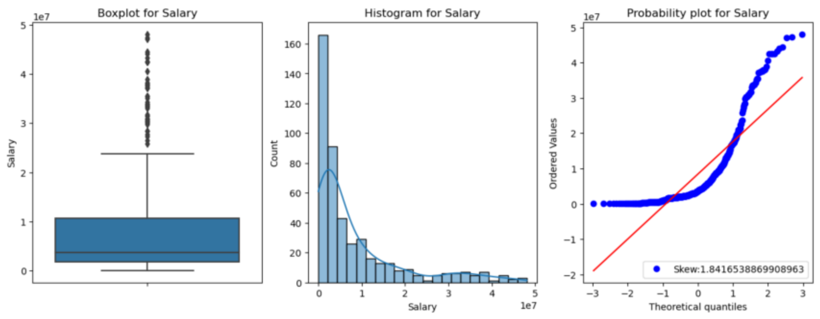
- ➢ Skewness of TOV – positively skewed with some outliers since the right tail of the boxplot is longer, the tail of the histogram is more skewed towards the right and the skew value is positive.



- ➢ Skewness of PF – almost normal with some outliers since the length of both the tails are the same and the histogram looks bell-shaped and centered around the mean. Also, all the points lies along the line in the probability plot and skew value is close to 0.



- Skewness of PTS – positively skewed with lots of outliers since the right tail of the boxplot is longer, the tail of the histogram is more skewed towards the right and the skew value is positive.



- Skewness of Salary – positively skewed with lots of outliers since the right tail of the boxplot is longer, the tail of the histogram is more skewed towards the right and the skew value is positive.

5. Testing Hypothesis

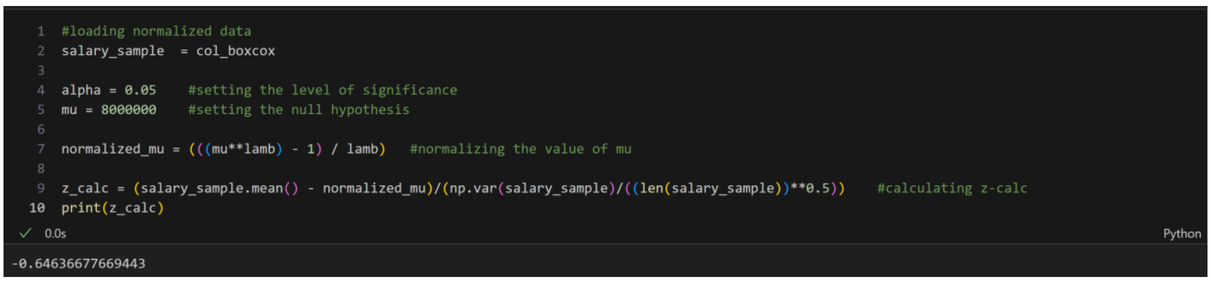
After standardizing the data, we're trying to predict the mean salary of the NBA players.

A testing hypothesis will be employed to check whether it is actually true or not.

We're assuming that the average salary of an NBA player is around 8,000,000.

H_0: $\mu = 8{,}000{,}000$ VS H_1: $\mu \neq 8{,}000{,}000$

$\alpha = 0.05$ (level of significance)



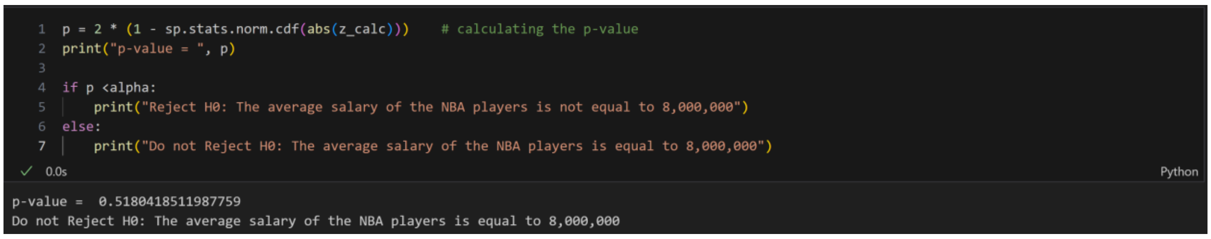
```
#loading normalized data
salary_sample  = col_boxcox

alpha = 0.05    #setting the level of significance
mu = 8000000    #setting the null hypothesis

normalized_mu = (((mu**lamb) - 1) / lamb)   #normalizing the value of mu

z_calc = (salary_sample.mean() - normalized_mu)/(np.var(salary_sample)/((len(salary_sample))**0.5))    #calculating z-calc
print(z_calc)
```

```
-0.64636677669443
```



```python
p = 2 * (1 - sp.stats.norm.cdf(abs(z_calc)))     # calculating the p-value
print("p-value = ", p)

if p <alpha:
    print("Reject H0: The average salary of the NBA players is not equal to 8,000,000")
else:
    print("Do not Reject H0: The average salary of the NBA players is equal to 8,000,000")
```

```
p-value =  0.5180418511987759
Do not Reject H0: The average salary of the NBA players is equal to 8,000,000
```