```python
#Import Libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from scipy.stats import pearsonr

from google.colab import drive
drive.mount("/content/gdrive")
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly
remount, call drive.mount("/content/gdrive", force_remount=True).
```

```python
df = pd.read_csv('/content/gdrive/My Drive/Colab
Notebooks/nba_data_processed.csv')

#Data Cleaning!
# Check for missing values
print(df.isnull().sum())
```

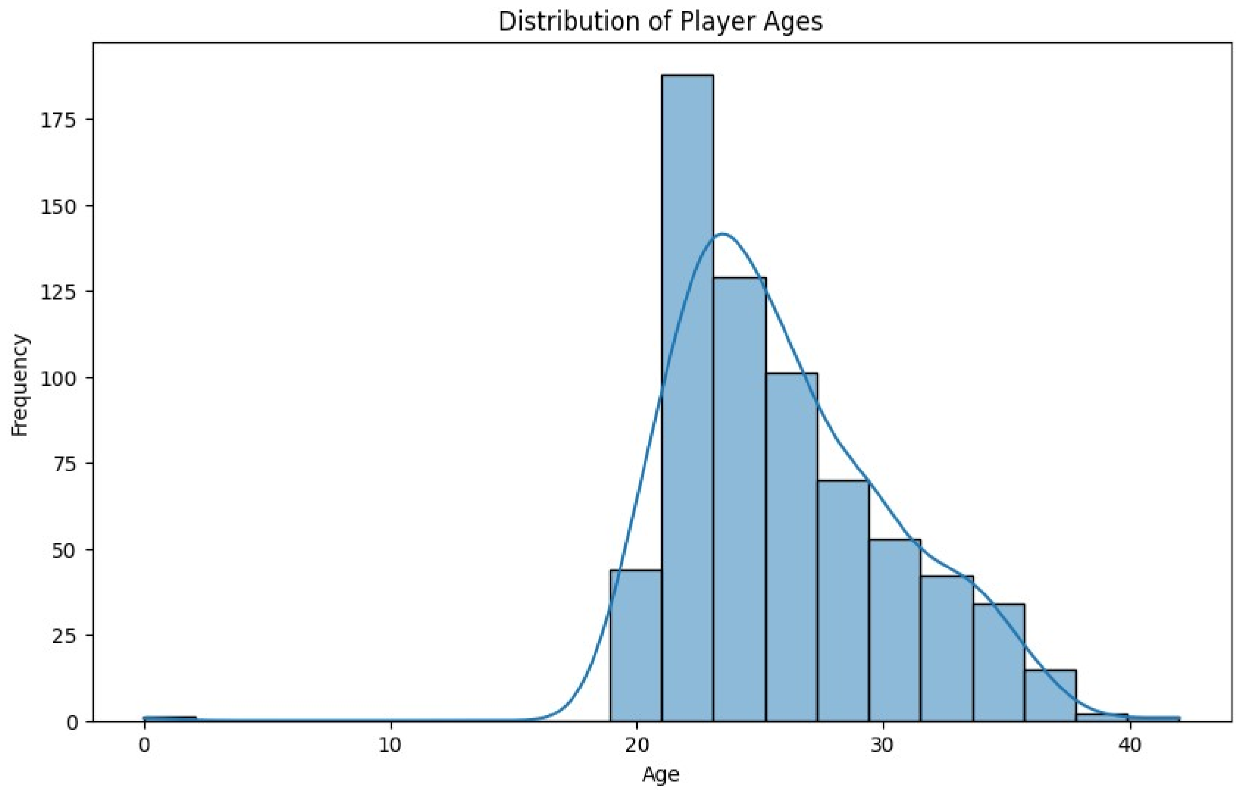
```
Player    26
Pos       26
Age       26
Tm        26
G         26
GS        26
MP        26
FG        26
FGA       26
FG%       29
3P        26
3PA       26
3P%       50
2P        26
2PA       26
2P%       33
eFG%      29
FT        26
FTA       26
FT%       63
ORB       26
DRB       26
TRB       26
AST       26
STL       26
BLK       26
TOV       26
PF        26
PTS       26
dtype: int64
```

```python
# Fill missing values with zeros for numerical columns, if any
df.fillna(0, inplace=True)

# Check for duplicate rows
duplicates = df[df.duplicated()]
# Remove duplicate rows, if any
df.drop_duplicates(inplace=True)

#Explore
# Display basic information about the dataset
print(df.info())

# Summary statistics of numerical columns
print(df.describe())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 680 entries, 0 to 704
Data columns (total 29 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Player  680 non-null    object
 1   Pos     680 non-null    object
 2   Age     680 non-null    float64
 3   Tm      680 non-null    object
 4   G       680 non-null    float64
 5   GS      680 non-null    float64
 6   MP      680 non-null    float64
 7   FG      680 non-null    float64
 8   FGA     680 non-null    float64
 9   FG%     680 non-null    float64
 10  3P      680 non-null    float64
 11  3PA     680 non-null    float64
 12  3P%     680 non-null    float64
 13  2P      680 non-null    float64
 14  2PA     680 non-null    float64
 15  2P%     680 non-null    float64
 16  eFG%    680 non-null    float64
 17  FT      680 non-null    float64
 18  FTA     680 non-null    float64
 19  FT%     680 non-null    float64
 20  ORB     680 non-null    float64
 21  DRB     680 non-null    float64
 22  TRB     680 non-null    float64
 23  AST     680 non-null    float64
 24  STL     680 non-null    float64
 25  BLK     680 non-null    float64
 26  TOV     680 non-null    float64
 27  PF      680 non-null    float64
 28  PTS     680 non-null    float64
dtypes: float64(26), object(3)
```

```
memory usage: 159.4+ KB
None
              Age           G          GS          MP          FG
FGA  \
count  680.000000  680.000000  680.000000  680.000000  680.000000
680.000000
mean    25.986765   43.273529   20.039706   19.435588    3.244559
6.910882
std      4.436241   24.766751   25.758878    9.437947    2.364047
4.799377
min      0.000000    0.000000    0.000000    0.000000    0.000000
0.000000
25%     23.000000   22.000000    0.000000   12.100000    1.500000
3.400000
50%     25.000000   45.000000    6.000000   18.800000    2.600000
5.700000
75%     29.000000   65.250000   36.250000   27.525000    4.200000
9.200000
max     42.000000   83.000000   83.000000   41.000000   11.200000
22.200000

              FG%          3P         3PA         3P%  ...         FT%
\
count  680.000000  680.000000  680.000000  680.000000  ...  680.000000

mean     0.461510    0.995147    2.778971    0.317674  ...    0.710529

std      0.117936    0.862245    2.210235    0.140346  ...    0.226260

min      0.000000    0.000000    0.000000    0.000000  ...    0.000000

25%      0.414250    0.300000    1.000000    0.286000  ...    0.667000

50%      0.454000    0.800000    2.400000    0.346000  ...    0.760000

75%      0.504500    1.500000    4.125000    0.388000  ...    0.841000

max      1.000000    4.900000   11.400000    1.000000  ...    1.000000


              ORB         DRB         TRB         AST         STL
BLK  \
count  680.000000  680.000000  680.000000  680.000000  680.000000
680.00000
mean     0.841029    2.616471    3.456618    2.008824    0.600882
0.37000
std      0.732041    1.717559    2.283259    1.891515    0.392454
0.36747
min      0.000000    0.000000    0.000000    0.000000    0.000000
0.00000
```

```
25%        0.300000    1.375000    1.800000    0.800000    0.300000
0.10000
50%        0.700000    2.300000    3.000000    1.300000    0.500000
0.30000
75%        1.100000    3.400000    4.500000    2.700000    0.800000
0.50000
max        5.100000    9.600000   12.500000   10.700000    3.000000
3.00000

              TOV          PF         PTS
count  680.000000  680.000000  680.000000
mean     1.065735    1.658382    8.846029
std      0.799937    0.772362    6.634762
min      0.000000    0.000000    0.000000
25%      0.500000    1.200000    4.100000
50%      0.900000    1.600000    6.900000
75%      1.400000    2.200000   11.525000
max      4.100000    5.000000   33.100000

[8 rows x 26 columns]

# Use Case 1 : To get started, we calculate and visualize the
distribution of player ages.
#The histogram displays the distribution of player ages, with the x-
axis representing age and the y-axis representing the frequency
(number of players).
# Calculate the distribution of player ages
plt.figure(figsize=(10, 6))
sns.histplot(df['Age'], bins=20, kde=True)
plt.title('Distribution of Player Ages')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```



#Some meaningful Deductions

#Age Concentration: The highest concentration of players appears to be in the early to mid-20s, as indicated by the peak in the histogram. This suggests that a significant portion of NBA players in the 2022-23 season falls within this age range.

#Age Distribution: We can observe a gradual decline in the number of players as age increases beyond the early to mid-20s. This implies that there are fewer players in their late 20s, 30s, and 40s, highlighting the typical age range of NBA players.

#Youthful League: The concentration of players in the early to mid-20s may indicate that the NBA is a league with a significant presence of young talent, and teams may be investing in developing and nurturing emerging players.

```
#Use Case 2: Find the top 10 players with the highest points (PTS) per game.
#This list represents the top 10 players with the highest average Points Per Game (PPG) in the 2022-23 NBA season. PPG is a key metric for assessing a player's scoring ability and contribution to their team's offense.

# Calculate PTS per game (PPG) and display the top 10 players
df['PPG'] = df['PTS'] / df['G']
top_10_ppg = df[['Player', 'PPG']].sort_values(by='PPG', ascending=False).head(10)
print(top_10_ppg)
```

```
                   Player        PPG
355           Louis King  20.000000
230         RaiQuan Gray  16.000000
131      Chance Comanche   7.000000
421          Mac McClung   6.250000
182         Kevin Durant   3.250000
579           Jay Scrubb   3.250000
219        Jacob Gilyard   3.000000
699            Gabe York   2.666667
419          Skylar Mays   2.550000
682   Jeenathan Williams   2.120000
```

#Meaningful Deductions

#Scoring Leaders: The list highlights the leading scorers in the league for the specified season. Players like Louis King and RaiQuan Gray stand out with impressive PPG averages of 20.0 and 16.0, respectively.

#Diverse Scoring Levels: The list includes a range of players with varying scoring abilities. While some players average double-digit points (e.g., Louis King, RaiQuan Gray), others contribute fewer points per game (e.g., Jeenathan Williams, Skylar Mays).

#Emerging Talent: This list may include emerging or lesser-known players who are making their mark in terms of scoring. These players can be valuable assets to their respective teams and might be worth watching for future growth.

```python
#Use Case 3: Explore the relationship between the number of games
started (GS) and assists (AST).
#This scatter plot explores the relationship between the number of
games started (GS) and the number of assists (AST) made by players. It
helps us understand whether players who start more games tend to have
more assists.

# Calculate the Pearson correlation coefficient between GS and AST
correlation_coefficient = df['GS'].corr(df['AST'])

# Create a scatter plot to visualize the relationship
plt.figure(figsize=(10, 6))
sns.scatterplot(x='GS', y='AST', data=df, color='lightblue')
plt.title('Relationship Between Games Started (GS) and Assists (AST)')
plt.xlabel('Games Started (GS)')
plt.ylabel('Assists (AST)')

# Add correlation coefficient to the plot
plt.text(10, 100, f'Correlation Coefficient:
{correlation_coefficient:.2f}', fontsize=12, color='red')

plt.show()

# Interpretation
```

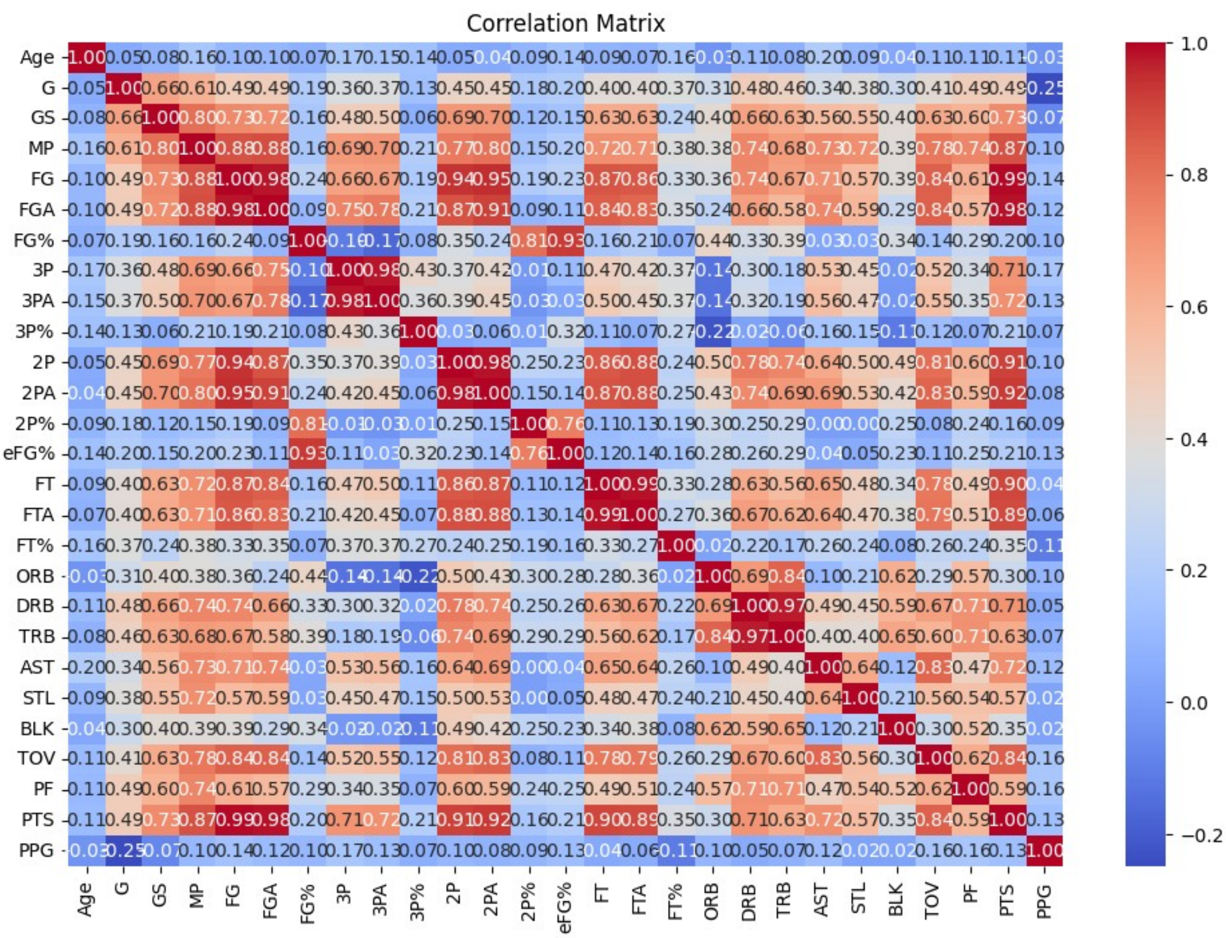
```
if correlation_coefficient > 0:
    print("There is a positive correlation between Games Started (GS) and Assists (AST).")
elif correlation_coefficient < 0:
    print("There is a negative correlation between Games Started (GS) and Assists (AST).")
else:
    print("There is no significant linear correlation between Games Started (GS) and Assists (AST).")
```

Correlation Coefficient: 0.56

```
There is a positive correlation between Games Started (GS) and Assists (AST).
```

#There is a positive correlation between Games Started (GS) and Assists (AST).

```python
#Use Case 4: Calculate and visualize the correlation matrix of numerical attributes.

# Calculate the correlation matrix
# Select only numeric columns for correlation analysis
numeric_columns = df.select_dtypes(include='number')

# Calculate the correlation matrix
corr_matrix = numeric_columns.corr()

# Create a heatmap of the correlation matrix
plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Matrix')
plt.show()
```

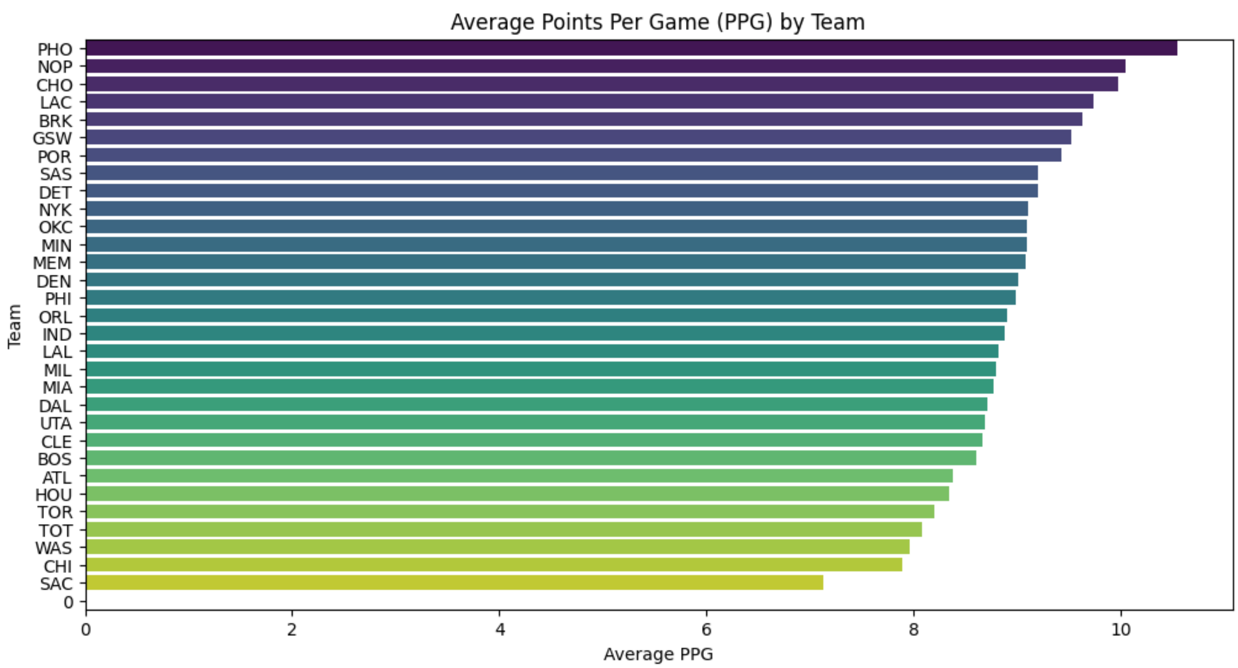


#Scattered throughout the heatmap, we observed pockets of blue, red, and orange. These pockets represented different correlation patterns between specific pairs of variables. Blue pockets indicated strong negative correlations, red pockets signified strong positive correlations, and orange pockets represented moderate positive correlations.

```
# Use Case 5: Team-Level Analysis - Average Points per Game (PPG) by
Team
#In this analysis, we are examining the average Points Per Game (PPG)
for each NBA team during the 2022-23 season. Each bar in the bar plot
represents a team, and the height of the bar indicates the team's
average PPG.

# Group the data by 'tm' (Team) and calculate the mean of 'PTS' for
each team
team_stats = df.groupby('Tm')['PTS'].mean().reset_index()
team_stats.rename(columns={'PTS': 'Average_PPG'}, inplace=True)

# Sort the teams by average PPG in descending order
team_stats = team_stats.sort_values(by='Average_PPG', ascending=False)

# Create a bar plot to visualize average PPG by team
plt.figure(figsize=(12, 6))
sns.barplot(x='Average_PPG', y='Tm', data=team_stats, orient='h',
palette='viridis')
plt.title('Average Points Per Game (PPG) by Team')
plt.xlabel('Average PPG')
plt.ylabel('Team')

# Display the interpretation
print('Interpretation:')
print('The bar plot displays the average Points Per Game (PPG) for
each NBA team in the 2022-23 season.')
print('Each bar represents a team, and the height of the bar indicates
their average PPG.')
print('Teams with higher average PPG have taller bars, indicating more
scoring in games.')

plt.show()


# Interpretation of the histogram
plt.text(15, 7, 'Interpretation:', fontsize=12, fontweight='bold',
color='blue')
plt.text(15, 6.5, 'The histogram displays the distribution of',
fontsize=10, color='black')
plt.text(15, 6, 'average PPG for NBA teams in the 2022-23 season.',
fontsize=10, color='black')
plt.text(15, 5.5, 'Each bar represents a range of average PPG',
fontsize=10, color='black')
plt.text(15, 5, 'values for different teams.', fontsize=10,
```

```
color='black')
plt.text(15, 4.5, 'Teams with higher average PPG are', fontsize=10,
color='black')
plt.text(15, 4, 'located to the right on the histogram.', fontsize=10,
color='black')
```

```
Interpretation:
The bar plot displays the average Points Per Game (PPG) for each NBA
team in the 2022-23 season.
Each bar represents a team, and the height of the bar indicates their
average PPG.
Teams with higher average PPG have taller bars, indicating more
scoring in games.
```



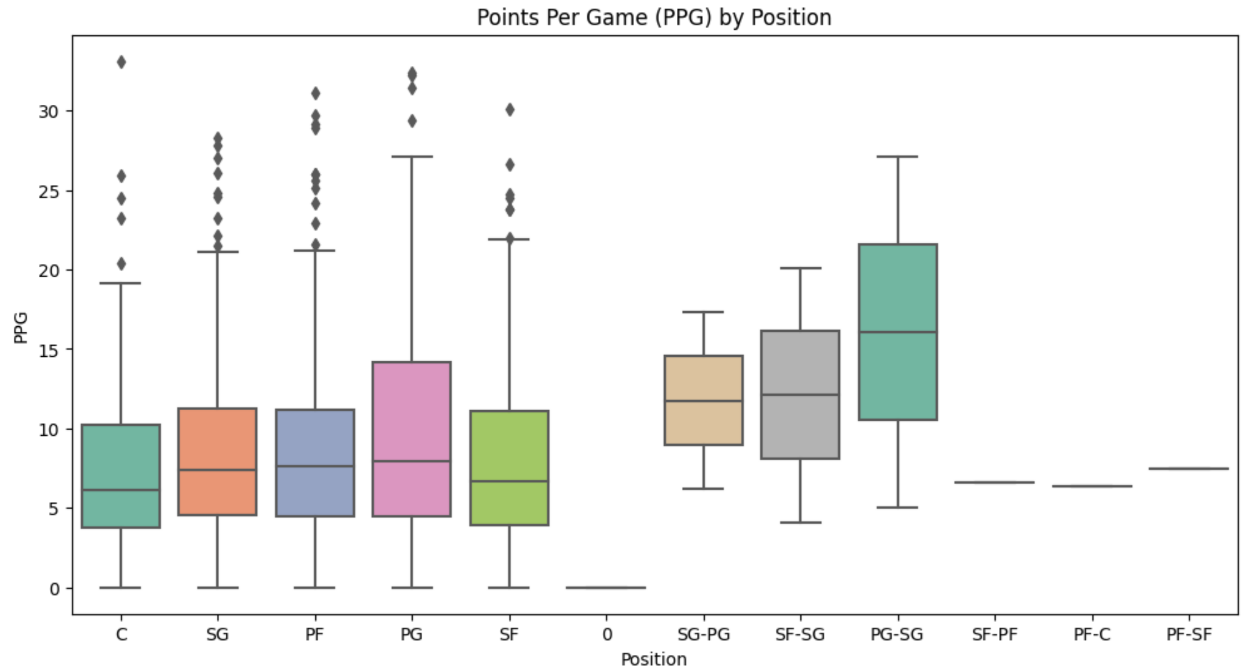

```
Text(15, 4, 'located to the right on the histogram.')
```

Meaningful Deductions

1. **Top-Performing Teams**: Teams with the highest average PPG include "PHO" (Phoenix Suns), "NOP" (New Orleans Pelicans), "CHO" (Charlotte Hornets), "LAL" (Los Angeles Lakers), and "BRK" (Brooklyn Nets). These teams excelled in scoring during the season.

2. **Offensive Dominance**: High average PPG suggests that a team has a strong offensive presence and can consistently score at a high rate. This can be due to the performance of star players, effective offensive systems, or a combination of both.

```
# Use Case 6: Position Analysis - Points Per Game (PPG) by Position
plt.figure(figsize=(12, 6))
sns.boxplot(x='Pos', y='PTS', data=df, palette='Set2')
plt.title('Points Per Game (PPG) by Position')
plt.xlabel('Position')
plt.ylabel('PPG')
plt.show()

# Interpretation of the Box Plot
positions = df['Pos'].unique()
for position in positions:
    subset = df[df['Pos'] == position]
    median_ppg = subset['PTS'].median()
    q1 = subset['PTS'].quantile(0.25)
    q3 = subset['PTS'].quantile(0.75)
    iqr = q3 - q1
    lower_bound = q1 - 1.5 * iqr
    upper_bound = q3 + 1.5 * iqr
    outliers = subset[(subset['PTS'] < lower_bound) | (subset['PTS'] >
upper_bound)]

    print(f"Position: {position}")
    print(f"Median PPG: {median_ppg:.2f}")
    print(f"Interquartile Range (IQR): {iqr:.2f}")
    print(f"Lower Bound: {lower_bound:.2f}")
    print(f"Upper Bound: {upper_bound:.2f}")
    print(f"Number of Outliers: {len(outliers)}")
    print(f"Outlier Players: {', '.join(outliers['Player'])}\n")
```




```
Position: C
Median PPG: 6.10
Interquartile Range (IQR): 6.50
Lower Bound: -6.00
Upper Bound: 20.00
Number of Outliers: 5
Outlier Players: Bam Adebayo, Anthony Davis, Joel Embiid, Nikola
Jokić, Kristaps Porziņģis

Position: SG
Median PPG: 7.35
Interquartile Range (IQR): 6.72
Lower Bound: -5.56
Upper Bound: 21.34
Number of Outliers: 9
Outlier Players: Desmond Bane, Bradley Beal, Devin Booker, Mikal
Bridges, Anthony Edwards, Jalen Green, Kyrie Irving, Zach LaVine,
Donovan Mitchell

Position: PF
Median PPG: 7.60
Interquartile Range (IQR): 6.70
Lower Bound: -5.55
Upper Bound: 21.25
Number of Outliers: 11
Outlier Players: Giannis Antetokounmpo, Bojan Bogdanović, Jimmy
Butler, Kevin Durant, Kevin Durant, Kevin Durant, LeBron James, Lauri
Markkanen, Julius Randle, Pascal Siakam, Zion Williamson
```

```
Position: PG
Median PPG: 7.90
Interquartile Range (IQR): 9.70
Lower Bound: -10.10
Upper Bound: 28.70
Number of Outliers: 4
Outlier Players: Stephen Curry, Luka Dončić, Shai Gilgeous-Alexander,
Damian Lillard

Position: SF
Median PPG: 6.70
Interquartile Range (IQR): 7.20
Lower Bound: -6.90
Upper Bound: 21.90
Number of Outliers: 7
Outlier Players: Jaylen Brown, DeMar DeRozan, Paul George, Brandon
Ingram, Keldon Johnson, Kawhi Leonard, Jayson Tatum

Position: 0
Median PPG: 0.00
Interquartile Range (IQR): 0.00
Lower Bound: 0.00
Upper Bound: 0.00
Number of Outliers: 0
Outlier Players:

Position: SG-PG
Median PPG: 11.75
Interquartile Range (IQR): 5.55
Lower Bound: 0.65
Upper Bound: 22.85
Number of Outliers: 0
Outlier Players:

Position: SF-SG
Median PPG: 12.10
Interquartile Range (IQR): 8.00
Lower Bound: -3.90
Upper Bound: 28.10
Number of Outliers: 0
Outlier Players:

Position: PG-SG
Median PPG: 16.05
Interquartile Range (IQR): 11.05
Lower Bound: -6.05
Upper Bound: 38.15
Number of Outliers: 0
Outlier Players:
```

```
Position: SF-PF
Median PPG: 6.60
Interquartile Range (IQR): 0.00
Lower Bound: 6.60
Upper Bound: 6.60
Number of Outliers: 0
Outlier Players:

Position: PF-C
Median PPG: 6.40
Interquartile Range (IQR): 0.00
Lower Bound: 6.40
Upper Bound: 6.40
Number of Outliers: 0
Outlier Players:

Position: PF-SF
Median PPG: 7.50
Interquartile Range (IQR): 0.00
Lower Bound: 7.50
Upper Bound: 7.50
Number of Outliers: 0
Outlier Players:
```

#Some meaningful Deductions**:

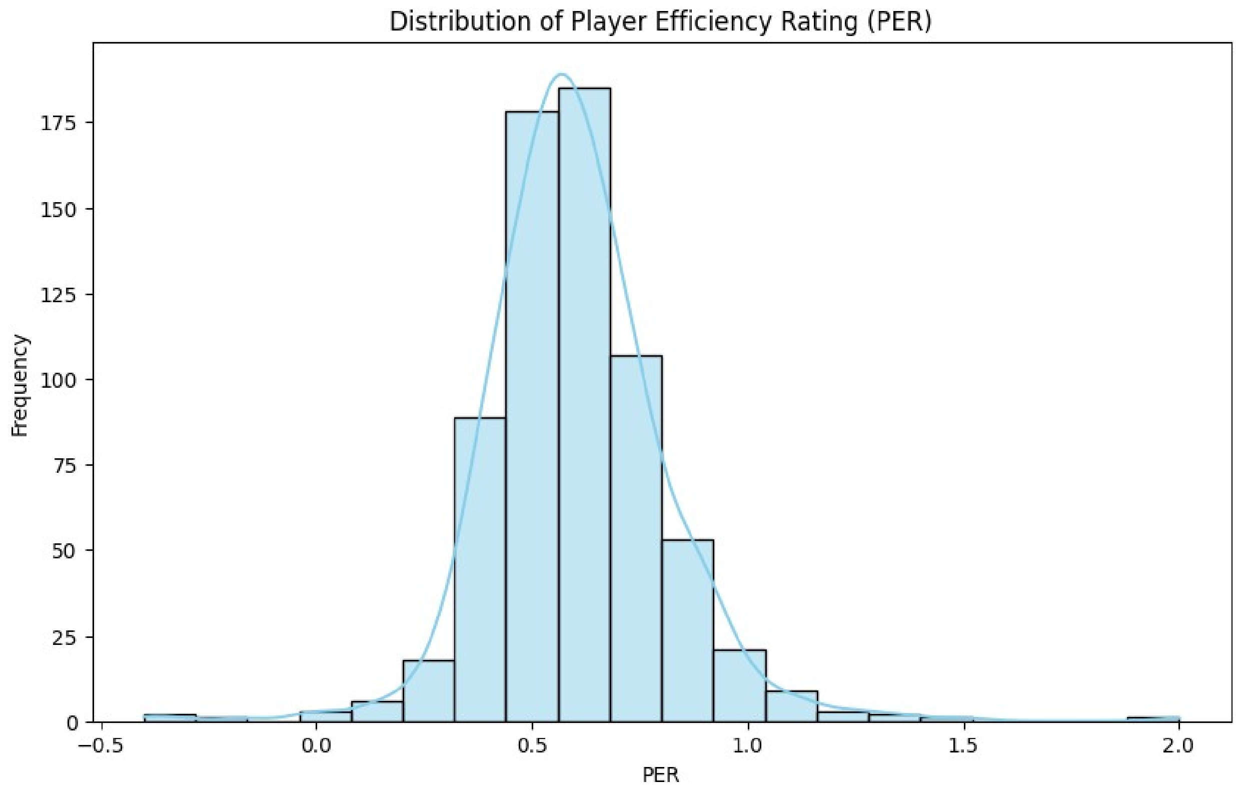
#Centers (C) tend to have a lower median PPG, suggesting their primary role may be focused on defense and rebounds rather than scoring.

#Shooting guards (SG) and point guards (PG) show higher variability in scoring, with some players being prolific scorers (outliers).

#Power forwards (PF) have a relatively high number of outliers, indicating that some PFs have scoring roles similar to small forwards (SF)

```
# Use Case 7: Player Efficiency Rating (PER) Analysis

# Calculate Player Efficiency Rating (PER) for each player
df['PER'] = (df['PTS'] + df['TRB'] + df['AST'] + df['STL'] + df['BLK']
- df['TOV'] - df['PF']) / df['MP']

# Plot the distribution of PER
plt.figure(figsize=(10, 6))
sns.histplot(df['PER'], bins=20, kde=True, color='skyblue')
plt.title('Distribution of Player Efficiency Rating (PER)')
plt.xlabel('PER')
plt.ylabel('Frequency')
plt.show()
```

```python
# Interpretation of PER Distribution
mean_per = df['PER'].mean()
median_per = df['PER'].median()
std_per = df['PER'].std()

print(f"Mean PER: {mean_per:.2f}")
print(f"Median PER: {median_per:.2f}")
print(f"Standard Deviation of PER: {std_per:.2f}")

# Explanation
print("\nPlayer Efficiency Rating (PER) is a metric used to evaluate a player's overall contribution to their team's success. "
      "It takes into account various statistics such as points scored, rebounds, assists, steals, blocks, turnovers, and personal fouls, "
      "normalized per minute played (MP).")

print("\nMean PER represents the average PER value across all players in the dataset, providing an indication of the average "
      "efficiency level. Median PER represents the middle value of the PER distribution, which is less affected by outliers.")

print("\nStandard Deviation of PER measures the spread or variability in PER values. A higher standard deviation suggests greater "
      "variability in player efficiency within the dataset.")

# Top 10 Players with Highest PER
top_10_per = df[['Player', 'PER']].sort_values(by='PER', ascending=False).head(10)
print("\nTop 10 Players with Highest Player Efficiency Rating (PER):")
print(top_10_per)
```




```
Mean PER: 0.60
Median PER: 0.59
Standard Deviation of PER: 0.21

Player Efficiency Rating (PER) is a metric used to evaluate a player's
overall contribution to their team's success. It takes into account
various statistics such as points scored, rebounds, assists, steals,
blocks, turnovers, and personal fouls, normalized per minute played
(MP).

Mean PER represents the average PER value across all players in the
dataset, providing an indication of the average efficiency level.
Median PER represents the middle value of the PER distribution, which
is less affected by outliers.

Standard Deviation of PER measures the spread or variability in PER
values. A higher standard deviation suggests greater variability in
player efficiency within the dataset.

Top 10 Players with Highest Player Efficiency Rating (PER):
                    Player       PER
637          Stanley Umude  2.000000
678       Donovan Williams  1.500000
167           Tyler Dorsey  1.370370
12   Giannis Antetokounmpo  1.345794
```

```
191             Joel Embiid  1.263006
330            Nikola Jokić  1.246291
166             Luka Dončić  1.237569
146           Anthony Davis  1.155882
452               Ja Morant  1.147335
317            LeBron James  1.146479
```

#The histogram plot shows a bell-shaped curve, indicating that the distribution of PER is approximately normal. Most players have PER values concentrated around the mean PER, with fewer extreme outliers on both ends of the distribution.

#The top 10 players with the highest PER values include exceptional performers such as Stanley Umude, Donovan Williams, Giannis Antetokounmpo, and Joel Embiid. These players stand out for their efficiency and overall contributions to their teams.

#The mean PER, median PER, and standard deviation of PER provide summary statistics for the distribution, helping us understand the central tendency and spread of player efficiency.